UNIVERSITY OF CALIFORNIA,
IRVINE

Pixel-Level Prediction: Models and Applications

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Shu Kong

2019

DEDICATION

To the FAMILY.

TABLE OF CONTENTS

Page

LIST OF FIGURES **vii**

LIST OF TABLES **xiii**

ACKNOWLEDGMENTS **xv**

CURRICULUM VITAE **xvi**

ABSTRACT OF THE DISSERTATION **xxii**

1 Introduction **1**
- 1.1 Pixel-Level Prediction and Applications 2
- 1.2 Learning based Method for Pixel-Level Prediction 3
  - 1.2.1 Model architecture 4
  - 1.2.2 Learning protocols 7
- 1.3 Data and supervision 9
  - 1.3.1 Data Acquisition Dilemma and Solutions 10
- 1.4 Contributions 12
- 1.5 Thesis Organization 13

2 Learning for Pixel-level Prediction **17**
- 2.1 Learning with Unified Framework 18
- 2.2 Example Problems with Loss Function for Training 20
- 2.3 Experimental Result 24
  - 2.3.1 Datasets 25
  - 2.3.2 Results 26
  - 2.3.3 Comprehensive Benchmark Comparison 26
  - 2.3.4 Qualitative Visualization 28

3 Depth-aware Gating for Contextual Pooling **31**
- 3.1 Background 32
- 3.2 Related work 35
- 3.3 Depth-aware Gating Module 36
- 3.4 Recurrent Refinement Module 39
- 3.5 Implementation 40
- 3.6 Experiments 41
  - 3.6.1 Datasets and Benchmarks 41
  - 3.6.2 Depth Prediction 42

3.6.3 Semantic Segmentation 44
3.6.4 Analysis of Gating Architectures Alternatives 47
3.6.5 Qualitative Results 49
3.7 Conclusion and Discussion 50

4 Pixel-wise Attentional Gating for Dynamic Computation 53
4.1 Background 54
4.2 Related Work on Dynamic and Spatially Varying Computation 57
4.3 Pixel-wise Attentional Gating 58
4.3.1 Plug-in PAG inside a Residual Block 58
4.3.2 Learning Discrete Attention Maps 60
4.3.3 Dynamic Per-Pixel Computation Routing 61
4.3.4 Dynamic Spatial Pooling 63
4.4 Implementation and Training 64
4.5 Experiments 65
4.5.1 Analysis of Pixel-wise Attentional Gating 65
4.5.2 Comprehensive Benchmark Comparison 68
4.5.3 Qualitative Visualization 70
4.6 Conclusion and Future Work 71

5 Pixel Embedding for Instance Grouping 73
5.1 Background 74
5.2 Related Work 76
5.3 Pairwise Loss for Pixel Embeddings 78
5.3.1 Instance-aware Pixel Weighting 80
5.3.2 Margin Selection 81
5.4 Recurrent Mean-Shift Grouping 82
5.4.1 Mean Shift Clustering 83
5.4.2 End-to-end training 85
5.5 Experiments 85
5.5.1 Tasks, Datasets and Implementation 86
5.5.2 Boundary Detection 87
5.5.3 Object Proposal Detection 88
5.5.4 Semantic Instance Detection 89
5.6 Conclusion and Future Work 92

6 Image Reconstruction with Simulated Data 98
6.1 Background 99
6.2 Related Work 102
6.3 Predictive Filter Flow 104
6.3.1 Learning to predict filter flow 104
6.3.2 Model Architecture and Training 106
6.4 Experiments 108
6.4.1 Datasets and Metrics 108
6.4.2 Non-Uniform Motion Blur Removal 109

6.4.3 JPEG Compression Artifact Reduction 112
6.4.4 Single Image Super-Resolution 113
6.5 Visualization and Analysis 115
6.6 Conclusion and Future Work 118

7 Self-Supervised Learning with Videos **119**
7.1 Background 119
7.2 Related Work 123
7.2.1 Multigrid PFF 124
7.2.2 Predictive Filter Flow (PFF) on Video 124
7.2.3 Predictive Filter Flow (PFF) vs. Optical Flow 125
7.2.4 Multigrid PFF 127
7.2.5 Constraints, Regularization and Loss 130
7.2.6 Architecture and Implementation 132
7.3 Experiments 132
7.3.1 Experimental Setup 133
7.3.2 Video Object Segmentation and Tracking 135
7.3.3 Long-Range Frame Reconstruction 138
7.3.4 Interactive Photo Editing 140
7.4 Conclusion 141

8 Concluding Remarks **142**
8.1 Summary and Contributions 142
8.2 Suggestions and Future Research 144

Bibliography **148**

A Appendix to Chapter 3 **178**
A.1 Results on the SUN-RGBD dataset 180
A.2 Visualization on Large Perspective Images 182

B Appendix to Chapter 4 **186**
B.1 Local Surface Normal in Panoramas 186
B.2 Further Analysis of Loss Functions and MultiPool Module 189
B.2.1 Boundary Detection on BSDS500 Dataset 189
B.2.2 Monocular Depth Estimation on NYUv2 Dataset 191
B.2.3 Surface Normal Estimation on NYUv2 Dataset 191
B.3 More Qualitative Visualization 193

C Appendix to Chapter 5 **199**
C.1 Details of Recurrent Mean Shift Grouping 200
C.1.1 Gradient Calculation for Recurrent Mean Shift 202
C.1.2 Toy Example of Mean Shift Backpropagation 204
C.2 Additional Boundary Detection Results 206
C.3 Additional Results on Instance-Level Semantic Segmentation 207

D Appendix to Chapter 6 **217**
D.1 Visualization of Per-Pixel Loading Factors 217
D.2 Iteratively Removing Motion Blur 218
D.3 More Qualitative Results 219

E Appendix to Chapter 7 **225**
E.1 Appendix 225
E.2 Intermediate Reconstruction by mgPFF 225
E.3 Graph Visualization of mgPFF architecture 226
E.4 Pixel Embedding in mgPFF 226
E.5 Video Demos 227

LIST OF FIGURES

Page

2.1 Visualization of some pixel-level prediction tasks. Given the images shown in first row, these tasks require per-pixel prediction for boundary, semantic segmentation, depth estimation and surface normal estimation, as shown in the second row. As we can see, these per-pixel prediction problems are ubiquitous, as agnostic to scene styles and image content. Our learning based models with the CNN architecture perform well in solving these tasks, as visualized in the third row. 17

2.2 Extending the U-shape architecture as shown in (a) enables finer-grained prediction, while with computation overhead. (b) adds in skip connections for better training and is widely adopted in pixel-level prediction tasks. (c) adds in another stream which is of full resolution for fine pixel level predictions, but demands more computation cost and memory utilization. 18

2.3 Comparison of cosine loss and inverse cosine loss. Inverse cosine loss has a constant gradient, while the gradient of the widely used cosine loss decreases as the prediction errors become small, preventing further model refinement. 24

2.4 Visualization of some failure cases in semantic segmentation, such as the messy predictions in the Cityscapes image and the "holes" on the wall in Stanford-2D-3D image. We note the model fails more easily in such perspective images, where both small and large objects exist in a single image. This observation motivates us to investigate how to better exploit contextual information for each pixel. 29

3.1 **Upper**: depth-aware gating spatially modulates the selected pooling scale using a depth map predicted from monocular input. In the chapter, we also evaluate related architectures where scene depth is provided directly at test time as a gating signal, and where spatially adaptive attentional gating is learned without any depth supervision. **Lower**: example ground-truth compared to predictions with and without the depth gating module. Rectangles overlayed on the image indicate pooling field sizes which are adapted based on the local depth estimate. We quantize the depth map into five discrete scales in our experiments. Using depth-gated pooling yields more accurate segment label predictions by avoiding pooling across small multiple distant objects while simultaneously allowing using sufficiently large pooling fields for nearby objects. 32

3.2 The input to our recurrent module is the concatenation (denoted by $\oslash$) of the feature map from an intermediate layer of the feed-forward pathway with the prior recurrent prediction. Our recurrent module utilizes depth-aware gating which carries out both depth regression and quantized prediction. Updated depth predictions at each iteration gate pooling fields used for semantic segmentation. This recurrent update of depth estimation increases the flexibility and representation power of our system yielding improved segmentation. We illustrate the prediction prior to, and after two recurrent iterations for a particular image and visualize the difference in predictions between consecutive iterations which yield small but notable gains as measured by average intersection-over-union (IoU) benchmark performance. 37
3.3 Examples of monocular depth predictions. First row: the input RGB image; second row: ground-truth; third row: our result. In our visualizations, all depth maps use the same fixed (absolute) colormap to represent metric depth. 43
3.4 Performance comparisons across gating architectures including tied vs untied parameters across different branches, averaging vs gating branch predictions, using monocular predicted vs ground-truth depth for the gating signal, gating pooling region size (MultiPool) or rescaling input image (MultiScale), and gating without depth supervision during training (attention). 47
3.5 Visualization of the attention maps on random images from Cityscapes and Stanford-2D-3D. The raw disparity/depth maps and the quantized versions are also shown for reference. Though we train the attention branch with randomly initialized weights, we can see that the learned attention maps capture some depth information as well as encoding distance to object boundaries. . 51
3.6 Visualization of the output on NYU-depth-v2. We show four randomly selected testing images with ground-truth and predicted disparity (first row), quantized disparity (second row) and segmentation (third row) at each iteration of the recurrent computation. 52
3.7 Visualization of randomly selected validation images from Cityscapes and Stanford-2D-3D with the segmentation output and the predicted quantized disparity at each iteration of the recurrent loop. We depict “ground-truth” continuous and quantized disparity beneath the input image. Our monocular disparity estimate makes predictions for reflective surfaces where stereo fails and recurrent iteration further improves estimates, particularly for featureless areas such as the pavement. Note that Cityscapes shows disparity while Stanford-2D-3D shows depth so the colormaps are reversed. 52

4.1 Pixel-wise Attentional Gating units (PAG) achieve parsimonious inference by learning a dynamic computation path for each pixel under limited computation budget. The "ponder cost" maps shown in the last row provide a visualization of the amount of computation allocated to each location (generated by accumulating binary masks from PAG units across all layers). We apply PAG to a variety of per-pixel labeling tasks (boundary detection, semantic segmentation, monocular geometry) and evaluate over diverse image datasets (indoor/outdoor scenes, narrow/wide field-of-view). 55
4.2 (a) A standard residual block. (b) Pixel-wise Attentional Gating unit (PAG) integrated into a residual block. Boxes/arrows denote activations/computations. **G** is a sparse, binary map that modulates what processing applied to each spatial location. "$\oslash$" means the perforated convolution [84], which assembles only active pixels for computation. 60
4.3 PAG-based MultiPool module learns to select the pooling size for each spatial location so that contextual information can be better aggregated. This can be implemented efficiently using perforated convolution [84], denoted by $\oslash$, which assembles only active pixels for computation in each pooling branch and thus avoids computing all pooled versions. 63
4.4 (a) Performance vs. computation budget (controlled by sparsity ρ) for boundary detection and segmentation tasks, % indicate computational savings relative to non-gated model. (b) Benchmark comparison of our PAG Multi-pool (MP@Res4) with state-of-the-art methods for boundary detection. 68
4.5 Visualization of sparse binary attention maps at each layer for boundary detection, together with the output and ponder map accumulating all binary maps. PAG-based MultiPool module is inserted at layer Res4-2, which is not included in the ponder map. 68
4.6 Visualization of semantic segmentation, monocular depth and surface normal estimation over the three datasets. Besides the overall ponder map, we also show the partial ponder map for each macro residual block by squeezing the sparse binary attentional maps. Random images are chosen from the three datasets, and results are warped for visualization. 72

5.1 Our framework embeds pixels into a hyper-sphere where recurrent mean-shift dynamics groups pixels into a variable number of object instances. Here we visualize random projections of a 64-dim embeddings into 3-dimensions. 75
5.2 Loss as a function of calibrated similarity score Eq. 5.1 with $\alpha = 0.5$. The gradient is constant, limiting the effect of noisy ground-truth labels (i.e., near an object boundary) 79
5.3 Recurrent mean shift grouping module is unrolled during training. 82
5.4 Demonstration of mean-shift grouping on a synthetic image and with ground-truth instance identities (left panel). Right panel: the pixel embedding visualization at 3-dimensional embedding sphere (upper row) and after 10 iterations of recurrent mean-shift grouping (bottom row). 84

5.5 To analyze the recurrent mean shift grouping module, we compare the embedding vector gradients with and without one loop of grouping. The length of arrows in the projection demonstrates the gradient magnitude, which are also depicted in maps as the second column. Backpropagating the loss through the grouping module serves to focus updates on embeddings of ambiguous pixels near boundaries while ignoring pixels with small errors which will be corrected by the subsequent grouping process. 93
5.6 Boundary detection performance on BSDS500 94
5.7 Visualization of boundary detection embeddings. We show the 3D embedding as RGB images (more examples in appendix). The upper and lower row in the right panel show embedding vectors at different layers from the model before and after fine-tuning using logistic loss. After fine-tuning not only predict the boundary pixels, but also encode boundary orientation and signed distance to the boundary, similar to supervised embedding approaches [305, 322, 12] . . 95
5.8 Segmented object proposals evaluation on PASCAL VOC 2012 validation set measured by Average Recall (AR) at IoU from 0.5 to 0.95 and step size as 0.5. We also include the curve for our method at IoU=0.5. 95
5.9 Semantic segmentation performance as a function of distance from ground-truth object boundaries comparing a baseline model trained with cross-entropy loss versus a model which also includes embedding loss. 96
5.10 The proposed embedding loss improves semantic segmentation by forcing the pixel feature vectors to be similar within the segments. Randomly selected images from PASCAL VOC2012 validation set. 96
5.11 Visualization of generic/instance-level semantic segmentation on random PASCAL VOC 2012 validation images. 97

6.1 Overview of the proposed *Predictive Filter Flow* which is readily applicable to various low-level vision problems, yielding state-of-the-art performance for non-uniform motion blur removal, compression artifact reduction and single image super-resolution. Given a single corrupted image, a two-stream CNN analyzes the image and synthesizes the weights of spatially-varying linear filters at each pixel. The per-pixel filter is then convolved with the corresponding local region to produce a deblurred/denoised pixel prediction. The whole framework is end-to-end trainable in a self-supervised way for the three aforementioned tasks where corrupted images can be simulated automatically. The predicted filters are easily constrained for different tasks and interpretable (here visualized in the center column by the mean flow displacement, see Fig. 6.6). 99
6.2 Visual comparison of our method (*PFF*) to *[Sun, et al.]* [310] and *patch-optim [Bahat, et al.]* [11] on testing images released by [11]. Please be guided with the strong edges in the filter flow maps to compare visual details in the deblurred images by different methods. Also note that the bottom two rows display images from the real-world, meaning they are not synthesized and there is no blur ground-truth. Best view in color and zoom-in. 107

6.3 Visual comparison of our methods (PFF and CNN). Strong edges in the expected flow map (right) highlight areas where most apparent artifacts are removed. Best viewed in color and zoomed-in. 110
6.4 Performance vs. training data with different compression quality factors measured by PSNR and SSIM and their performance gains, over the LIVE1 dataset. The original JPEG compression is plotted for baseline. 114
6.5 Visual comparison of our method (PFF) to CNN, each image is super-resolved ($\times$4). More results can be found in the supplementary material. Best view in color and zoom-in. 115
6.6 Three row-wise panels: (1) We run K-means (K=400) on all filters synthesized by the model over the test set, and visualize the 400 centroid kernels using t-SNE on a 2D plane; (2) top ten principal components of the synthesized filters; (3) visualizing the color coded filter flow along with input and quality image. Each pixels filter is assigned to the nearest centroid and the color for the centroid is based on the 2D t-SNE embedding using the color chart shown at top left. 117

7.1 The flowchart of multigrid Predictive Filter Flow framework (mgPFF). Conceptually we draw a single scale for demonstrating how we train our model in an unsupervised way with the photometric reconstruction loss along with constraints imposed on the filter flow maps. The multigrid strategy is illustrated in Fig. 7.3. 122
7.2 **Comparison between Predictive Filter Flow (PFF) and optical flow**, both of which are trained over the Sintel movie and the optical flow dataset [33]. We evaluate training with a mix of supervised and unsupervised data controlled by a hyperparameter α where α=1 is fully-supervised optical flow regression. We note >96% frames are without flow annotations, while only <4% have ground-truth flows. By varying α, we plot the image reconstruction error and endpoint error (EPE) in (a) and (b), respectively; and visualize the flows and reconstructed frames in (c). See details in Section 7.2.3. 126
7.3 Illustration of how multigrid Predictive Filter Flow (mgPFF) performs progressively by warping images from one to the other at multiple resolution scales from coarse to fine. 128
7.4 Visualization of unsupervised learning for video segmentation on DAVIS2017: *soccerball*, *dog* and *bear*. We show the tracking results with temporal window size K=3 for *soccerball* (otherwise it loses track due to heavy occlusion) and K=1 for others. Note that in *soccerball*, there are heavy occlusions but our mgPFF model can still track the ball. In *dog*, we can see how each pixel moves along with the dog: when the dog turns from right side to left side, the colors from the neck are propagated for tracking. In *bear*, we see an error where the shadow appearing in the bottom of the image is explained by flow from the bear's leg. 137

7.5 **Long-range frame reconstruction** (rightmost column) reflects long-range flow learning that warps $\mathbf{I}_t$ (1st column) with the coordinate flow (2nd column) which is transformed from the predicted multigrid filter flow. The target frames $\mathbf{I}_{t+10}$ are shown in the 3rd column. 139
7.6 **Interactive Photo Editing**. We compare mgPFF with two classic blending algorithms (alpha blending and Poisson blending) and the unsupervised optical flow method. We can see the self-regularization enables our mgPFF preserve both the structure in the source image patch and target image style, allowing for more "realistic" restoration. The upper-left flow maps visualize the optical/filter flows. 140

LIST OF TABLES

Page

2.1 Benchmark comparison of our baseline model with state-of-the-art methods We report the standard F-measure at Optimal Dataset Scale (odsF), Optimal Image Scale (oisF), and the Average Precision (AP) for boundary detection. 26

2.2 Semantic segmentation is measured by Intersection over Union (IoU), pixel accuracy (acc), and iIoU that leverages the size of segments w.r.t categories. Results marked by † are from our trained models with the released code. . . 27

2.3 Depth estimation is measured by standard threshold accuracy, *i.e.* the percentage (%) of predicted pixel depths d_i s.t. $\delta = \max(\frac{d_i}{\hat{d}_i}, \frac{\hat{d}_i}{d_i}) < \tau$, where $\tau = \{1.25, 1.25^2, 1.25^3\}$. Methods with * use ∼100k extra images to train. . 27

2.4 Surface normal estimation is measured by mean angular error and the percentage of prediction error within t° degree where $t = \{11.25, 22.50, 30.00\}$. Smaller ang. err. mean better performance as marked by ↓. 28

2.5 Study of loss functions and PAG unit for monocular depth estimation on NYUv2 dataset. Our MP@Res5 model is the base model, unless specified, all the models are trained with softmax weighted average in the MultiPool module. The performance is measured by standard threshold accuracy, *i.e.* the percentage of predicted pixel depths d_i s.t. $\delta = \max(\frac{d_i}{\hat{d}_i}, \frac{\hat{d}_i}{d_i}) < \tau$, where $\tau = \{1.25, 1.25^2, 1.25^3\}$. 28

2.6 Study of the loss functions for surface normal estimation over NYUv2 dataset. Performance is measured by mean angular error (ang. err.) and the portion of prediction error within t° degree where $t = \{11.25, 22.50, 30.00\}$. Smaller ang. err. means better performance as marked by ↓. 29

3.1 Depth prediction on NYU-depth-v2 dataset. 42

3.2 Performance of semantic segmentation on different datasets. Results marked by † are from our trained model models with the released code, and results marked by * are evaluated by the dataset server on test set. Note that we train our models based on ResNet50 architecture on indoor datasets NYU-depth-v2 and SUN-RGBD, and ResNet101 on the large perspective datasets Cityscapes and Stanford-2D-3D. 45

4.1 Ablation study for where to insert the PAG-based MultiPool module. Experiments are from boundary detection and semantic segmentation on BSDS500 and NYUv2 dataset, measured by F-score ($F_{bnd.}$) and IoU(IoU$_{seg.}$) respectively. Numbers are in % (higher is better). 65

4.2 Computational parsimony compared with truncated ResNet and models learning to drop/skip whole layers. Evaluation is performed on NYUv2 dataset for semantic segmentation. 66
4.3 Semantic segmentation is measured by Intersection over Union (IoU), pixel accuracy (acc), and iIoU that leverages the size of segments w.r.t categories. Results marked by † are from our trained models with the released code. . . 70
4.4 Depth estimation is measured by standard threshold accuracy, *i.e.* the percentage (%) of predicted pixel depths d_i s.t. $\delta = \max(\frac{d_i}{\hat{d}_i}, \frac{\hat{d}_i}{d_i}) < \tau$, where $\tau = \{1.25, 1.25^2, 1.25^3\}$. Methods with * use $\sim$100k extra images to train. . 71
4.5 Surface normal estimation is measured by mean angular error and the percentage of prediction error within t° degree where $t = \{11.25, 22.50, 30.00\}$. Smaller ang. err. mean better performance as marked by $\downarrow$. 71

5.1 Object proposal detection on PASCAL VOC 2012 validation set measured by total Average Recall (AR) at IoU=0.50 and various number of proposals per image. 89
5.2 Instance-level segmentation comparison using APr metric at 0.5 IoU on the PASCAL VOC 2012 validation set. 90

6.1 Comparison on motion blur removal over the non-uniform motion blur dataset [11]. Moderate/large blur are from motion up to 16/38 pixels. For the two metrics, the larger value means better performance of the model. 109
6.2 Comparison on JPEG compression artifact reduction over LIVE1 dataset [344]. PSNR and SSIM are used as metrics listed on two rows respectively in each macro row grid (the larger the better). 112
6.3 Comparison on single image super-resolution ($\times$4) over the classic Set5 [24], Set14 [373] and Urban [131] datasets. The metrics used here are PSNR/SSIM ($\times 10^{-2}$). . . . 116

7.1 **Tracking Segmentation** on the DAVIS2017 validation set. Methods marked with ^{1st} always use the first frame and its mask (provided) for tracking. The number in bracket indicates the estimated number of frames used for training. 136
7.2 **Long-range flow for frame Reconstruction**: We compute the long-range flow vector from the filter flows by mgPFF on two frames $\mathbf{I}_t$ and $\mathbf{I}_{t+m}$, where m=5 or m=10. We use the flow to warp $\mathbf{I}_t$ towards $\mathbf{I}_{t+m}$, and report the averaged pixel difference under the L_1 norm between the warped frame and $\mathbf{I}_{t+m}$. 138

ACKNOWLEDGMENTS

Compared to the PhD journey, this thesis is simply a pollen grain, evidencing the journey while expecting the future; but it would not have existed without Charless Fowlkes, who means so much to me that drives me to be a “little Char<-:”.

In this journey to a degree of scientific training, I would not shape myself any better without the tweaks from the “elders” in the academic family (what a shame I couldn’t learn every A-to-Z from them)-: Pierre Baldi, Alex Berg, Olivier Cinquin, Larry S. Davis, Alyosha Efros, Alex Ihler, Ramesh Jain, Aaron Kuan, G.P. Li, Kevin P. Murphy, Surangi Punyasena, Andrew Owens, Deva Ramanan, Amit Roy-Chowdhury, Cordelia Schmid, Erik Sudderth, Rahul Sukthankar, Rick Szeliski, Christian Theobalt, Donghui Wang, Qiang Yang, Yezhou Yang, Shuang Zhao. At the same time, this journey becomes so sweet and colorful with the company of my “gang”: here and there, in and out, near and far, junior and senior.

However, this journey is more about life, enjoyable and memorable. In life, my family have witnessed my growth; my spiritual family grow together under his arrangement. So I acknowledge them for the fine life we have been enjoying, past, present and the eternal.

How amazing! Life can be more! That becomes especially true in this journey when I have the very fortune to marry my beloved wife and welcome our seed, crying “Hello, world!”

Disclaimer&Achknowledgement Everything included in this thesis has been either submitted or published in the form of conference or journal papers. No companies hold any copyright on any these research projects presented in this thesis. The research is supported by multiple funding sources: NSF grants IIS-1618806, IIS-1253538, DBI-1262547, the NIH grant NIA R01AG057748, UCI Multi-disciplinary Design Program, hardware donation from NVIDIA, Google Graduate Research Award, Adobe Research Gift, HHMI Janelia Junior Scientist Workshop Travel Grant, CVPR2019 PhD Consortium Travel Grant, WACV2019 PhD Consortium Travel Grant. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agents.

Further Acknowledgement. Much research in this thesis was initially motivated by the pollen and worm projects; some were much inspired from the lessons taught by Mr. Etthew Kong, who was a zero-year-old boy showing off the great learning ability.

CURRICULUM VITAE

Shu Kong

EDUCATION

Doctor of Philosophy in Computer Science	**2019**
University of California, Irvine	*Irvine, California*
Master of Science in Computer Science	**2013**
Zhejiang University	*Hangzhou*
Bachelor of Science in Computer Science	**2010**
Donghua University	*Shanghai*

RESEARCH EXPERIENCE

Graduate Research Assistant	**2015 – 2019**
University of California, Irvine	*Irvine, California*
Research Intern	**06/2019 – 09/2019**
Facebook Inc.	*Menlo Park, California*
Software Engineering Intern	**06/2017 – 09/2017**
Google Inc.	*Mountain View, California*
Research Intern	**06/2015 – 09/2015**
Adobe Systems Incorporated	*San Jose, California*
Research Assistant	**2013 – 2014**
Hong Kong University of Science and Technology	*Hong Kong*
Research Assistant	**2010 – 2013**
Zhejiang University	*Hangzhou*

TEACHING

Big Data Image Processing and Analysis	Fall 2017
Computational Photography and Vision (CS116)	Spring 2017
Big Data Image Processing and Analysis	Fall 2016
Graph Algorithms (CS163)	Spring 2016
Introduction to Machine Learning (CS273)	Winter 2016
Graph Algorithms (CS163)	Spring 2015
Machine Learning and Data Mining (CS178)	Winter 2015
Introduction to Graphical Models (CS179)	Fall 2015

PUBLICATIONS

Z. Fang, **S. Kong**, C. Fowlkes, Y. Yang, "Modularized Textual Grounding for Counterfactual Resilience", Computer Vision and Pattern Recognition (CVPR), 2019.
S. Kong, C. Fowlkes, "Pixel-wise Attentional Gating for Scene Parsing", Winter Conference on Applications of Computer Vision (WACV), 2019

S. Kong, C. Fowlkes, "Recurrent Pixel Embedding for Instance Grouping", Computer Vision and Pattern Recognition (CVPR), 2018

S. Kong, C. Fowlkes, "Recurrent Scene Parsing with Perspective Understanding in the Loop", Computer Vision and Pattern Recognition (CVPR), 2018.

S. Kong, C. Fowlkes, "Low-rank Bilinear Pooling for Fine-Grained Classification", Computer Vision and Pattern Recognition (CVPR), 2017.

S. Kong, X. Shen, Z. Lin, R. Mech, C. Fowlkes, "Photo Aesthetics Ranking Network with Attributes and Content Adaptation", European Conference on Computer Vision (ECCV), 2016.

S. Kong, S. Punyasena, C. Fowlkes, "Spatially Aware Dictionary Learning and Coding for Fossil Pollen Identification", Computer Vision for Microscopy Image Analysis (CVMI), 2016.

S. Kong, Z. Jiang, Q. Yang, "Modeling Neuron Selectivity over Simple Mid-Level Features for Image Classification", IEEE Trans. on Image Processing, 2015

Yuetan Lin, **S. Kong**, Donghui Wang, Yueting Zhuang, "Saliency Detection within a Deep Convolutional Architecture", AAAI Workshop on Cognitive Computing for Augmented Human Intelligence, 2014.

S. Kong*, D. Wang* "A Classification-Oriented Dictionary Learning Model: Explicitly Learning the Particularity and Commonality Across Categories", Pattern Recognition, 2014.

S. Kong, D. Wang, "Learning Exemplar-Represented Manifolds in Latent Space for Classification", European Conference on Machine Learning and European Conference on Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD), 2013.

D. Wang, X. Wang, **S. Kong**, "Integration of Multi-Feature Fusion and Dictionary Learning for Face Recognition", Image and Vision Computing (IVC), 2013.

S. Kong, D. Wang, "Learning Individual-Specific Dictionaries with Fused Multiple Features for Face Recognition", IEEE Conference on Automatic Face and Gesture Recognition (FG), 2013.

S. Kong, X. Wang, D. Wang, "Multiple Feature Fusion for Face Recognition", IEEE Conference on Automatic Face and Gesture Recognition (FG), 2013.

S. Kong, D. Wang, "A Dictionary Learning Approach for Classification: Separating the Particularity and the commonality", European Conference on Computer Vision (ECCV), 2012.

S. Kong, D. Wang, "Transfer Heterogeneous Unlabeled Data for Unsupervised Clustering", International Conference on Pattern Recognition (ICPR), 2012.

S. Kong, D. Wang, "A Multi-task Learning Strategy for Unsupervised Clustering via Explicitly Separating the Commonality", International Conference on Pattern Recognition (ICPR), 2012.

D. Wang, **S. Kong**, "Learning Class-Specific Dictionaries for Digit Recognition from Spherical Surface of a 3D Ball", Machine Vision and Applications (MVA), 2012.

D. Wang, **S. Kong**, "Feature Selection from High-Order Tensorial Data via Sparse Decomposition", Pattern Recognition Letters, 2012.

I. Romero, **S. Kong**, C. Fowlkes, M. Urban, F. Oboh-Ikuenobe, C. D'Apolitoe, C. Jaramillo, S. Punyasena, "Reconstructing the Cenozoic expansion and diversification of the pollen morphospecies Striatopollis catatumbus (Amherstieae, Fabaceae) using convolutional neural nets," 2019 under review.

D. Haselhorst, **S. Kong**, C. Fowlkes, S. Punyasena, "Automated identification of diverse palynological samples using convolutional neural nets", 2019 udner review.

S. Kong, C. Fowlkes, "Image Reconstruction with Predictive Filter Flow", 2019 under review.

S. Kong*, F. Zhou*, C. Fowlkes, T. Chen, B. Lei, "Fine-Grained Facial Expression Analysis Using Dimensional Emotion Model", 2018 under review.

Z. Fang, **S. Kong**, C. Fowlkes, Y. Yang, "WSRA: Weak Supervision and Referring Attention for Temporal-Textual Association Learning over Videos", 2018 under review.

Y. Zhao, **S. Kong**, D. Shin, C. Fowlkes, "Fine-Grained Facial Expression Analysis Using Dimensional Emotion Model", 2018 under review.

S. Kong, C. Fowlkes, "Multigrid Predictive Filter Flow for Unsupervised Learning on Videos", 2019 under review.

ABSTRACT

Z. Fang, **S. Kong**, C. Fowlkes, Y. Yang, "Modularized Textual Grounding for Counterfactual Resilience", Language And Vision workshop joint with CVPR, 2019.
S. Punyasena, **S. Kong**, C. Fowlkes, "Improving the taxonomic accuracy and precision of fossil pollen identifications", North American Paleontological Convention, Riverside, USA, 2019.

I. Romero, **S. Kong**, C. Fowlkes, M. Urban, S. Punyasena, "Automated Neotropical Fossil Pollen Fabaceae Analysis Using Convolutional Neural Networks", GSA Annual Meeting in Indianapolis, Indiana, USA, 2018.

Z. Fang, **S. Kong**, T. Yu, Y. Yang, "Weakly Supervised Attention Learning for Textual Phrases Grounding", Language and Vision Workshop joint with CVPR, 2018.

S. Kong, C. Fowlkes, “Low-rank Bilinear Pooling for Fine-Grained Classification”, the Fourth Workshop on Fine-grained Visual Categorization joint with CVPR, 2017.

S. Kong, C. Fowlkes, “Recurrent Scene Parsing with Perspective Understanding in the Loop”, Southern California Machine Learning Symposium, 2017.

I. Romero, **S. Kong**, C. Fowlkes, M. Urban, C. D‘Apolito, C. Jaramillo, F. Oboh-Ikuenobea, S. Punyasena, “Novel Morphological Analysis of a Fossil Fabaceae Pollen Type, Striatopollis Catatumbus (Tribe Detariae)”, GSA, 2017.

I. Romero, **S. Kong**, C. Fowlkes, M. Urban, C. D‘Apolito, C. Jaramillo, F. Oboh-Ikuenobe, and S. Punyasena, “Cenozoic biogeography of Striatopollis catatumbus (Fabaceae Detariae)”, AASP-The Palynological Society, 2017.

D. Haselhorst, **S. Kong**, C. Fowlkes, J. Enrique Moreno, D. Tcheng, S. Punyasena, “Automating tropical pollen counts using convolutional neural nets: from image acquisition to identification”, the iDigBio inaugural conference, 2017.

S. Punyasena, **S. Kong**, C. Fowlkes, S. Jackson, “Reconstructing the extinction dynamics of Picea critchfieldii - the application of computer vision to fossil pollen analysis”, the iDigBio inaugural conference, 2017.

PATENT

S. Shen, Z. Lin, **S. Kong**, R. Mech, “Utilizing deep learning to rate attributes of digital images”, US 2018 / 0268535 A1.
S. Shen, Z. Lin, **S. Kong**, R. Mech, “Utilizing deep learning for rating aesthetics of digital images”, US20170294010A1.

CONSULTING

AlpineReplay, Inc.	2018–2019
US Cabinets Online LLC.	2018
Paralian Tech	2017

PRESENTATION

Pixel-level Learning and Prediction for Fine Vision 11/2019
Max Planck Institute for Informatics *host: Christian Theobalt*

The Best Practice in Unsupervised Depth Learning over Videos 8/2019
Facebook Research *Menlo Park*

Attending to Pixels, Embedding Pixels, Predicting Pixels	08/2019
CMU VASC Seminar	*host: Deva Ramanan*
Attending Pixels, Embedding Pixels, Predicting Pixels	01/2019
CVPR PhD Consortium	*Long Beach, USA*
Attention to Pixels, embed pixels, track pixels	06/2019
Caltech Computational Vision	*host: Pietro Perona and Oisin Mac Aodha*
Attention to Pixels, embed pixels, track pixels	05/2019
UC Berkeley BAIR Seminar	*host: Alexei Efros and Hany Farid*
Video Mining by Weakly/Un-supervised Learning	05/2019
USC CLVR	*host: Joseph Lim*
Video Mining: from Sub-pixel to Causality	04/2019
Video Computing Group at UC Reverside	*host: Amit Roy-Chowdhury*
Predictive Filter Flow: Diving into (Sub)pixels	02/2019
UC Berkeley BAIR	*host: Alexei Efros and Andrew Owens*
Fine-Grained Visual Understanding and Learning	01/2019
WACV PhD Consortium	*Waikoloa Village, HI*
Fine-Grained Image Understanding	09/2018
Trace AlpineReplay, Inc	*host: David Lokshin*
Pay Attention to the Pixel, Understand the Scene Better	05/2018
UCI Center for Machine Learning and Intelligent Systems	*host: Sameer Singh*
Scene Parsing through Per-Pixel Labeling: a better and faster way	03/2018
ASU Active Perception Group Seminar	*host: Yezhou Yang*
Predicting Real-World Distance between 360 Photos	09/2017
Google Geo	*Mountain View, CA*
Low-rank Bilinear Pooling for Fine-Grained Classification	11/2016
Southern California Machine Learning Symposium	*Caltech*
Automated Biological Image Analysis with CV/ML	10/2016
Janelia Research Junior Scientist Workshop	*Ashburn, VA*
Spatially Aware Dictionary Learning for Fossil Pollen Recognition	07/2016
CVPR CVMI Workshop	*Las Vegas, NV*

Geographically Aware Knowledge Mining on Mobile Data	05/2016
UCI Data Hackathon	*Irvine, CA*
Image Quality and Aesthetics Estimation	09/2015
Adobe Research	*San Jose, CA*
Automated Biological Image Analysis with CV/ML	05/2015
Multi-Disciplinary Project Research Symposium	*UCI*

Selected Honors

CVPR PhD Consortium	2019
WACV PhD Consortium	2019
Google Graduate Student Award	2017
Janelia Travel Support for Junior Scientist Workshop on ML&CV	2016
Multi-disciplinary Design Project Grant	2014
Computer Science Department Fellowship	2014
Excellent Graduate Student of Zhejiang Province	2013
Excellent Graduate Student of Zhejiang University	2013
ECCV Travel Grant	2012
National Graduate Scholarship	2012
Kefa Cen Scholarship	2012
The Star of eVideo – Innovation Scholarship	2011
Kwang-Hua Scholarship	2011
Graduate of Merit/Triple-A Graduate, Zhejiang University	2011, 2012
First-Class Award of Honor for Graduate, Zhejiang University	2011, 2012
Outstanding Bachelor Thesis Award	2010
Yangbang Redhat Scholarship	2009
Undergraduate Merit Scholarship	2007, 2008

ABSTRACT OF THE DISSERTATION

Pixel-Level Prediction: Models and Applications

By

Shu Kong

Doctor of Philosophy in Computer Science

University of California, Irvine, 2019

Prof. Charless C. Fowlkes, Chair

Pixel-level prediction enables visual understanding at finer granularity, such as segmenting all the persons and vehicles and estimating their 3D shapes as well as distances from the camera. It distinguishes itself from image-level prediction, which is relatively coarse such as simply telling an image of a person from that of a car.

The solutions to pixel-level prediction are core to many real-world applications, spanning a variety of vision tasks from the low-level vision like image deblurring, to the mid- and high-level such as understanding scene geometry and all the objects' 3D shape and motion. It has been greatly advanced since the past decade and fostering new challenges and opportunities, owing to the fast development of hardware, the availability of large-scale dataset, and the resurgence of deep convolutional neural networks.

In this thesis, we study pixel-level prediction with new algorithms, innovative model architectures, and novel applications. We begin with training convolutional neural networks (CNN) for different pixel-level prediction tasks, and demonstrate that CNN acts as a unified framework. Within the framework, we propose novel modules to encode perceptual or cognitive principles, such as 1) objects appearing larger when closer to the camera, and 2) cognitive mechanism allowing one for perceiving the world with dynamical attention. We show the proposed modules not only achieve the state-of-the-art performance on different tasks, but also enables dynamic and parsimonious computation.

As the dataset for per-pixel labeling tasks requires painstaking per-pixel annotations, we propose the Predictive Filter Flow (PFF) framework to train over simulated images for image reconstruction tasks. PFF generates per-pixel kernels for warping the input towards the output, thus has better interpretability w.r.t decision making. We further present its multigrid extension (dubbed mgPFF) to train over unconstrained videos. We show successful applications of mgPFF to visual tracking and flow learning, as well as a unique interactive application for photo editing.

Chapter 1

Introduction

Computer vision has been greatly advanced in the past decade. One evidence is that many image-level prediction tasks are nearly solved, such as image classification that distinguishes an image of person from that of a car. This is largely owing to the fast development of hardware enabling high-performance computation, the availability of big data allowing for learning powerful models, and the resurgence of deep convolutional neural networks providing tremendous learning capacity. At the same time, these factors also open up research in pixel-level prediction, which offers the opportunity for finer-grained visual understanding, such as segmenting all the persons and vehicles and estimating their 3D shapes as well as motions and distance from one another.

Pixel-level prediction is more challenging than image classification problems since it requires substantial attention to both image details at pixel level and holistic image understanding; yet it is of great significance because of its core role in various real-world applications. To understand its significance, in the following, we first present some applications linking some pixel-level prediction tasks, divided by the "level"[*]. Then, we introduce the state-of-the-art practice to approach pixel-level vision problems, before giving an overview of our work and

[*]Traditionally, computer vision has three stages: 1) the low level vision for image-to-image transform, like image deblurring; 2) the mid-level about image-to-features computation, such as grouping pixels into segments; and 3) the high-level meaning feature-to-analysis, e.g., semantic segmentation. However, such division is largely blurred nowadays, especially when many pixel-level tasks can be approached by training a supervised model with sufficient annotated dataset.

contributions in the thesis.

1.1 Pixel-Level Prediction and Applications

Low-level vision includes the general category of image reconstruction, which predicts every pixel directly based on the input image, aiming at obtaining high-quality reconstructed image from a low-quality input image. The low-quality of images arises from the imperfection of real-world images. Practical engineering trade-offs entail that consumer photos are often blurry due to low-light, camera shake or object motion, limited in resolution and further degraded by image compression artifacts introduced for the sake of affordable transmission and storage. Scientific applications such as microscopy or astronomy, which push the fundamental physical limitations of light, lenses and sensors, face similar challenges. Recovering high-quality images from degraded measurements has been a long-standing problem for image analysis and spans a range of tasks such as blind-image deblurring [21, 182, 81, 295], compression artifact reduction [298, 216], and single image super-resolution [258, 363]. Solutions to image reconstruction problems not only improve people's daily life (e.g., improved photography by cellphones), but also boost scientific research, e.g., through enhancing the microscopic images in biology and medical science.

The mid- and high-level computer vision also contain many pixel-level prediction problems, such as semantic segmentation that classifies every pixel into one of defined categories, monocular depth estimation that predicts distance to the camera for each pixel of the single input image, surface normal estimation that explains the 3D shape of the scene or object, optical flow estimation that tells how each pixel evolves over time, and so forth. These tasks are important components in autonomous vehicle, assistive robotics for the elderly and spacecraft autonomy, through helping the robots move around freely and safely. In AR/VR and gaming, they can help enhance visualization by understanding and augmenting the scene geometry. They can also play an important role in video editing and summarization, surveillance and security. For example, through visual grounding that localizes the object or video frames with a referring expressions (e.g., short natural language phrase and NLP sentence), one can answer questions like “the moment when the new couple are kissing at the wedding”, or “the moment when the man in yellow shirt appeared yesterday around the hall”. Last but not the least, solutions to these generic pixel-level prediction vision tasks can accelerate scientific research, such as predicting pixel identities to group them into neurons for studying brain connectomics, segmenting and counting C. elegans nematodes for studying lifespan, healthspan, and hypothesis testing of genetic interactions with aging.

Realizing the importance and broad impact of pixel-level prediction vision, we introduce the nowadays' practice for providing state-of-the-art solutions to these tasks.

■ 1.2 Learning based Method for Pixel-Level Prediction

As mentioned earlier, pixel-level prediction vision has been greatly advanced thanks to the development of hardware (espeically GPU), availability of large-scale annotated dataset, and the resurgence of deep convolutional neural networks (CNN). In particular, these factors lay the ground for supervised learning CNN models at scale. Though we note that supervised training CNN is only one of many learning based methods, it achieves the state-of-the-art performance in almost all the pixel-level prediction tasks. Therefore, we focus on CNN models for learning and briefly review the history and components.

Learning based methods essentially learn a mapping function, or a model, that maps input image into the output space defined by the task, fulfilling pixel-level prediction. As we know, pixels exist altogether in a local neighborhood and thus are not i.i.d. For better pixel level prediction, it is a common practice to design mechanisms for encoding informative connections among pixels [192, 224, 60, 274, 143, 371, 28, 372]. The convolution operation achieves this by convolving every patch with kernels in a dense manner, to make predictions for the pixel centered at the patch [78]. The kernels are the parameters to be learned.

We can see that the function should have enough capacity for good mapping from the raw input images into the desired output [124]. In terms of learning, one would also require the function to generalize to unseen data and avoid overfitting the training data. In this sense, the more data, the more sensible regularization and the larger capacity of the model we can have, the better model we can learn: fitting the training data sufficiently and generalizing to unseen data well. Among various learning models, the convolutional neural networks (CNN) shows excellent performance, not only having sufficient capacities [15], but also being regularized well for generalization from its convolutional locality property, weight sharing [190], weight decay [180], and so on. To understand the evolvement of CNN and gain a better view how other important factors developed over time related to CNN (e.g., data scale and learning protocols), we briefly dive into these aspects individually as below, which motivate our research included in this thesis.

1.2.1 Model architecture

Convolutional Neural Networks (CNN) appeared decades ago. LeCun et al. for the first time show the success of applying CNN to the real-world problem, hand-written digit recognition [193], which was then widely used in U.S. Postal Service. CNN became a principled model since its great success in natural image classification. In 2012, Krizhevsky et al. published the seminal work [179] that trains a CNN model (so-called AlexNet to mark the milestone in literature) on ImageNet dataset [63], remarkably outperforming all the others in the image classification challenge. In brief, AlexNet stacks multiple types of layers to form a deep architecture, and train the whole model with the classifiers in an end-to-end manner: the convolutional layers computing over local patches in a dense way to save computation and avoid overfitting; the non-linear transform layers as Rectified Linear Unit (ReLU) [247] which eases the overall training through error-backpropagation; the local-contrast normalization layers for scale the intermediate features values into a more stable range; the pooling layers that reduce computation and enable large receptive field for long-range pixel computation; and fully connected layers that project the intermediate features into the output space and thus for the final classification.

After AlexNet, many other model architectures have been designed that show success in image classification tasks. We review only a few architectures, while many more are available in literature but not covered in this thesis. Following AlexNet, VGG was proposed that stacks small 3x3 convolution kernels for more convolution layers [304], outperforming AlexNet in image classification on ImageNet dataset. It showed that large convolutional kernels are not critical, and the local-contrast normalization layers are not necessary for better training through normalizing features. Some later research shows the convolution kernels can be decomposed even further into smaller ones like 2x3 or 1x3 [314, 313]. Network-in-networks (NIN) shows deep models are better [210], and the fully-connected layers are not critical as used in AlexNet. ResNet [115] and DenseNet [129] are proposed that show skip-connections are important in learning very deep models, through enabling more reliable gradient back-propagation, achieving the state-of-the-art in image classification on ImageNet dataset with fewer computation FLOPS than VGG. More recently, researchers realize hand-designing the model architectures is not optimal for specific tasks [392]. Therefore there emerges a research direction called Neural Architecture Search (NAS) that trains over dataset for building better models in terms of performance [392], though other recent research shows the model with NAS overfits the target task easily and lacks generalization [174].

At the same time, with the inspiration of the architecture design for image classification, researchers began to modify CNN for dense prediction, i.e., pixel-level prediction. The modifications usually focus on the pooling layers in CNN. As we know, the pooling layers are important that pool the intermediate feature maps spatially into low-resolution ones, reducing computation and memory cost and also allows for capturing holistic information from long-range connections. However, pooling layers make the output size much smaller than the input image. For example, if the CNN model contains five non-overlapping pooling layers with 2x2 strides, the direct output is $2^5 = 32$ times smaller in resolution than the input.

Therefore, to increase the output resolution, some work proposes various techniques to improve the output resolution, while keeping low memory cost and computation and allowing for large receptive field to capture long-range information. For example, the Fully Convolutional Networks (FCN) [221] replaced the fully connected layers as used in VGG, and add upsampling layers to increase the output resolution. The upsampling layers can be either the simplistic bilinear interpolation which is differentiable for training using back-propagation, or a parametric layer with learnable weights. To make the receptive field even larger, researchers begin to utilize the dilated convolution, or atrous convolution, which convolves the input with kernels skipping input values and thus capture longer range of pixels [368]. Moreover, the U-shape architecture [285] is proposed that include an encoder, which acts like a standard ResNet or VGG, and a decoder which has skip connections to the encoder layers and upsampling layers, both help produce high-resolution output and boost gradient descent within the deep model during training.

Besides the micro building blocks, there are macro strategies, like the multi-scale [32] that process the input image at different scales and merge the multiple output towards a single one for better per-pixel prediction [356, 99, 45], and the multi-stream which incorporates another small but full-resolution network to provide more precise complement/residual prediction on every pixel [269, 164, 168].

With all the aforementioned development in CNN for per-pixel predictions, one can train the CNN models for solving specific tasks, with dataset and the choice of loss functions. We dive into training in Chpater 2, and next, review the learning protocols which make the whole training procedure faster, more reliable, and more stable.

■ 1.2.2 Learning protocols

In the learning world, research in optimization is of great value [29], especially in training CNN models which are effective in solving complex problems but are nonlinear in nature and containing huge number of parameters (in the scale of 10^7). Therefore, high-order optimization algorithms like Newton's method are unlikely to be used; instead the first-order optimization methods are widely studied and used in CNN training.

Training the deep CNN models is commonly done through back-propagation [287], i.e., back-propagating the errors through gradient from the loss function to all the layers and update the layers' weight accordingly. Therefore, the requirement for back-propagation or end-to-end training is that all layers have to be "differentiable[†]". As training CNN models involves large-scale training set, it is impossible to feed all the training data into the model at the same time. Therefore in practice, small, random mini-batches of data are fetched and fed into the model for training, and the Stochastic Gradient Descent (SGD) optimization is widely used in large-scale learning [27].

Many variants of SGD algorithms are studied in literature. Such research work can be categorized into two approaches: (1) adaptive learning rate schemes, such as AdaGrad [73] and Adam [156], and (2) accelerated schemes, such as heavy-ball and Nesterov momentum [311]. In practice, the combination of these attempts usually lead to stable learning and fast convergence [72], and thus is widely adopted in practice, embedded in deep learning toolboxes. We do not dive in the learning protocols any further as this is not the focus of this thesis.

[†]Technically speaking, models we typically train are "differentiable almost everywhere". For example, the derivative of ReLU is not defined at 0.

However, we would like to introduce and acknowledge some deep learning toolboxes that help us carry out our research, serving as an optimization package.

Before describing the toolboxes, we would like to mention normalization techniques along with optimization. Because normalization typically serve the purpose of making CNN models more amenable to optimization, allowing the training of very deep models without the use of careful initialization schemes [304, 375], customized nonlinear activation functions [158], or other more tricks [357].

Normalization for Optimization

Normalization methods commonly perform as individual layers. Among these methods, the Batch Normalization is perhaps the best known in recent literature, since it is the very first published method that show a sensible normalization can greatly boost the training, with demonstration of more stable training, faster convergence, better performance, etc [137]. In brief, to normalize intermediate features during training, Batch Normalization learns a “global mean” and “standard variance” accumulated over time, uses them to normalize spatial features within each training batch, which goes through mean subtraction and standard variance division using statistics of this training batch. As Batch Normalization comes with mini-batches during training, the batch size matters. If the batch size is 1, meaning using a single data sample at each time for updating the CNN parameters, it boils down to instance normalization [324], which is shown very effective in transferring the styles of an image independently, but less effective in other more domains. Group Normalization is another normalization method proposed to deal with small batch size [353], which could happen in training CNN with large input images, e.g., when one GPU can only hold one high-resolution image and its deep features. However, if the batch size is very large with the support of multi-gpu configurations, Ghost Batch Normalization is proposed to bridge the feature statistical gaps between mini-batches distributed in different machines [120]. In this thesis, we present research with training over medium-size data, so we do not face problems in small size or large size. Interested readers are referred to these papers for more details of different normalization methods.

Software and Toolbox

As deep learning has become so influential nowadays [191], there emerge many excellent toolboxes that highly wrap useful techniques, layers, optimizers for easy use in training customized models on novel dataset. Projects presented herein benefit from multiple toolboxes we would like to acknowledge. They are Decaf [67], Theano [317], MatConvNet [326], Tensorflow [1] and PyTorch [260]. We also note other great open-source packages, but we do not introduce them one by one while encourage interested readers to refer to them.

1.3 Data and supervision

Data with annotations are required for learning-based methods as the annotations provide the supervision for training. However, the overwhelming significance of data scale has been gradually revealed in recent years.

This can also date back to the very success of training AlexNet on the ImageNet dataset. Some follow-up research work, which aims at scrutinizing AlexNet, find that: 1) truncated CNN model can act as a better feature extraction than hand-designed visual features (e.g., SIFT [224], HOG [60]); 2) such learning-based features can be directly transferred to other tasks [67, 296], even though the new task has totally different domain statistics [169, 162]. 3) such pre-trained features or models can act as loss functions or metric functions [194, 380].

Because of these findings, it is a common practice to fine-tune ImageNet-pretrained model to train for other tasks, as well as for pixel-level prediction tasks [45, 74]. All these advances are due to the large-scale data used for training the initial model.

This is especially verified by recent research that the data scale is more important than what was thought a decade ago [97, 57, 246, 309, 250]: larger-scale dataset help learn better features, more robust and more generalizable, which can be transferred to diverse data domains for various applications.

▩ 1.3.1 Data Acquisition Dilemma and Solutions

Understanding the overwhelming significance of data scale in the deep learning era, one would annotate large-scale data for a curate training set to train the CNN models. While this is sometimes easy, most of the time it is very difficult, expensive in terms of human effort,

For some tasks, one can easily download large amount of images from the internet (e.g., flickr [170] and instagram [228]), along with the search keyword, or the tags or even the comments from users. By doing so, it shows that such web-scale dataset can remarkably boost the performance of specific tasks. For example, Krause et al. show that training over web-scale noisy bird dataset can outperform the state-of-the-art easily [178]. Mahajan et al. shows by training over web-scale data with only loose annotations from the user tags, one can obtain an even better model than the ImageNet pretrained model, in terms of image classification and model generalization [228]. Murray et al. download from professional photography website million of images and their comments, and train a model for image aesthetic estimation [246].

For other tasks, annotating the data for a training set is hard: time-consuming, expensive, and requires expertise. For example, annotating image aesthetics and the attributes takes moderate time and labor force [170]; but per-pixel annotation in autonomous driving scenario is very expensive. For Kitti [96] and Cityscapes [57], which include detailed annotations of every pixel, it takes one annotator almost three hours to annotate a single image. In other cases, annotating requires domain expertise, which ordinary people rarely have. For example, annotating pollen grain species (both modern and fossilized ones) for automated fine-grained identification [231, 169], annotating the C. elegans mask in microscope and their age for automated analysis system also requires expert knowledge [51, 265]. For some other tasks, it is impossible to annotate data for ground-truth. For example, optical flow requires pixel-level correspondence between two video frames; but annotating all pixel in a video is prohibitively time-consuming. For outdoor depth annotation, the LiDAR sensor can only provide the uncalibrated true depth in a very sparse way [97]; the kinect sensor can only return true depth on some surfaces, not possible in face of glasses or mirrors [75]; the surface normal generated from the depth annotation is inevitably noisy and inaccurate [116].

Fortunately, in some scenarios, we can synthesize data to create a training set. It is worth noting that synthetic data have different domain distributions, so that the domain gap becomes a serious problem and motivate the study of domain adaptation in the community [257, 319]. We note that, for low-level vision tasks like learning to estimate optical flow, models trained over such synthetic datasets often generalize very well to real-world images [236, 71]. To leverage both synthetic data and the small annotated real-world data, one can achieve even better performance on various vision tasks, especially those requiring pixel-level predictions. For example, by synthesizing scene images (outdoor or indoor) and their semantic segment masks, depth and normals at pixel level, one can train a better model that generalizes to real-world images [307, 382, 286]. As another example, one can also synthesize human poses using 3D human models [25], or background [325]; by doing so we can generate large-scale training data for human keypoint detection and thus for human pose estimation.

To attack the challenges of data annotation and collection, the community witnesses other efforts. Among them, self-supervised learning methods show great promise in training model with the supervision provided by the data itself, without any manual supervision.

Self-supervised methods can exploit unlimited real-world data, and do not have much domain discrepancy (unlike synthetic vs. real-world domains). There are some attempts in self-supervised learning, such as training the model for un-shuffling image patches or image jigsaw [65], image/video colorization [378, 66, 333], predicting the temporal order of frames [241, 345], learning for image inpainting [262], learning to predict image rotation [100], learning to reconstruct frames for pixel correspondence [168, 339], learning with multi-modal signals for cross supervision [6, 256, 255, 153, 254], learning to mimic the offline algorithms output (e.g., disparity [149], tracker [337], etc.).

1.4 Contributions

We briefly summarize the main contributions, which will be elaborated in the next section that outlines the thesis.

We study pixel-level prediction with new algorithms, innovative model architectures, and novel applications. We begin with training convolutional neural networks (CNN) for different pixel-level prediction tasks, and demonstrate that CNN acts as a unified framework. Within the framework, we propose novel modules to encode perceptual or cognitive principles, such as 1) objects appearing larger when closer to the camera, and 2) cognitive mechanism allowing one for perceiving the world with dynamical attention. We show the proposed modules not only achieve the state-of-the-art performance on different tasks, but also enables dynamic and parsimonious computation.

As the dataset for per-pixel labeling tasks requires painstaking per-pixel annotations, we propose the Predictive Filter Flow (PFF) framework to train over simulated images for image reconstruction tasks. PFF generates per-pixel kernels for warping the input towards the output, thus has better interpretability w.r.t decision making. We further present its multigrid extension (dubbed mgPFF) to train over unconstrained videos. We show successful applications of mgPFF to visual tracking, flow learning and interactive photo editing.

▣ 1.5 Thesis Organization

Before concluding this Chapter 1 as an introduction and history survey, we now give an overview of the thesis with the subsequent chapters. The introductory paragraphs of each chapter provide more detailed outlines.

Chapter 2

We describe some fundamental pixel-level computer vision tasks in this chapter, with some proposed loss functions and the trained models as a baseline. Through the effort, we show the CNN model acts as a unified architecture for learning to solve pixel-level prediction problems. For the different tasks, with respect to the model architecture, the last layer and the loss function are what one should pay attention to most. We show with experimental results, and thus solicit more sensible loss functions in practice. This chapter comes as a part of our published paper [165].

Chapter 3

We introduces our novel depth-aware gating module to encode the empirical observation that objects appear larger in size when closer to the camera, while smaller when further away. To study this module, we specifically apply it to semantic segmentation that classifies each pixel into one of pre-defined categories. We show this module improves semantic segmentation performance notably, especially over perspective images (e.g., images from large field-of-view camera). This study also demonstrates the benefit of using one task (depth estimation) to help with the other (semantic segmentation). Moreover, we propose to refine the semantic segmentation results along with updating the depth estimate in a recurrent way. We show such a practice leads to even better performance in semantic segmentation. This chapter is based on our published paper [167].

Chapter 4

We extend the previous depth-aware gating module to a more general one, called Pixel-wise Attentional Gating (PAG), which learns to assign a pooling size at each pixel for dynamic aggregating contextual information, showing better performance in not only semantic segmentation, but also many others include boundary detection, monocular depth estimation and surface normal estimation. Furthermore, we apply PAG to the CNN model's computation graph, and demonstrate that PAG allows for dynamic computation at each pixel, achieving better performance in all these tasks with a fixed computation budget (measured by FLOPS) than other baselines, such as a truncated model, layer-wise skipping model, etc. This chapter comes as a part of our published paper [165].

Chapter 5

We present a different high-level methodology called pixel embedding, projecting every pixel into an embedding space in which one can group pixels into meaningful regions. We apply our pixel embedding model to object instance segmentation, achieving remarkably better performance than other state-of-the-art methods in detecting object proposals with limited number of proposal candidates. We demonstrate our method, through other published work built over the idea of pixel embedding or inspired directly by our method, is very promising in segmenting biological matters, like C. elegans [181], bacteria [181], neurons [197], cells [43], etc. We also test it on other tasks, and show it can act as an auxiliary practice for semantic segmentation and boundary detection. This chapter comes as a part of our published paper [166].

Chapter 6

We introduces the Predictive Filter Flow (PFF) framework that learns to predict per-pixel kernels used for warping input image patches centered at each pixel towards the output pixels. Moreover, we train PFF models over simulated data, though it can be also trained with supervision from manual annotations, and demonstrate a notable success in the world of self-supervised learning. It is worth noting that PFF not only regularizes the output by the input itself, but also explicitly explains how the output pixels are generated from the input image; therefore we see PFF has better interpretability in terms of decision making. We demonstrate its effectiveness through a series of image reconstruction tasks, including non-uniform blur removal, single-image super-resolution, and JPEG compression artifact reduction. Besides, we also attempt to train it in an unsupervised way, i.e., even without simulation, and demonstrate the promise of using it in biological and medical image enhancement, in which an interpretable algorithm is much favored over blackbox models, e.g., an output directly from a CNN model is assumed to be the enhanced image. This chapter is based on our submitted paper [164].

Chapter 7

We extend the PFF idea to learning over unconstrained videos in a self-supervised way. The key idea is to train PFF models, taking as input two frames and generating per-pixel kernels for each frame to reconstruct the other frames. The only supervision is from the temporal coherence of videos. To process video frames and output high-resolution filter flows, we adopt a multigrid strategy, so we name our framework multigrid PFF (mgPFF). We demonstrate mgPFF learns to capture pixel correspondence, which can be directly used for tracking objects through propagating masks while also allowing for tracking every pixels along time. We show mgPFF reconstructs consecutive frames almost perfectly, and has the power to warp one image towards another. This strong ability of reconstruction and fast computing speed enable a novel interactive photo editing application, i.e., transplanting visual content (say a nose of one person) to a target image (e.g., the face of another person) while keeping the harmony of the target image. This chapter is based on our submitted paper [168].

Chapter 8

This is the last chapter of this thesis. We revisit our contributions and outline directions for future research.

Chapter 2

Learning for Pixel-level Prediction

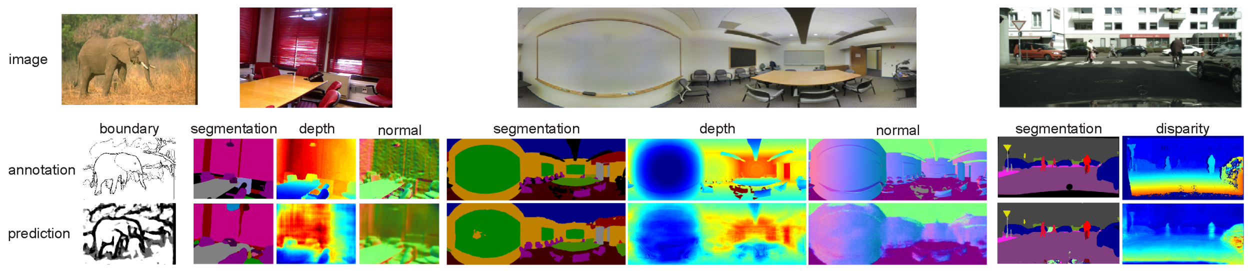


Figure 2.1: Visualization of some pixel-level prediction tasks. Given the images shown in first row, these tasks require per-pixel prediction for boundary, semantic segmentation, depth estimation and surface normal estimation, as shown in the second row. As we can see, these per-pixel prediction problems are ubiquitous, as agnostic to scene styles and image content. Our learning based models with the CNN architecture perform well in solving these tasks, as visualized in the third row.

The development of deep convolutional neural networks (CNN) has allowed remarkable progress in wide range of image pixel-labeling tasks such as boundary detection [232, 358, 161], semantic segmentation [166, 167, 45], monocular depth estimation [167, 200, 188, 219, 75], and surface normal estimation [336, 17, 74]. Figure 2.1 depicts visualizations for these tasks, with the results by our baseline models presented in this chapter soon.

As the focus of this thesis is about studying learning-based methods for solving pixel-level prediction vision tasks, we present in this chapter some fundamental pixel-level prediction problems, and introduce how we train the unified convolutional neural networks (CNN) over annotated datasets. In this sense, this chapter serves as a technical introduction. Therefore,

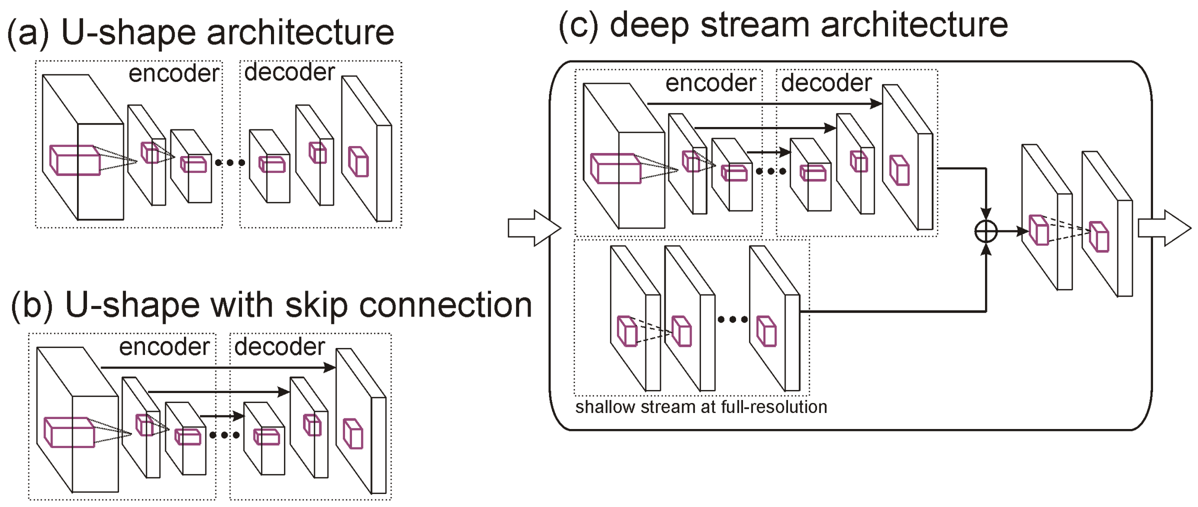

(a) U-shape architecture
encoder
decoder
(b) U-shape with skip connection
encoder
decoder
(c) deep stream architecture
encoder
decoder
shallow stream at full-resolution

Figure 2.2: Extending the U-shape architecture as shown in (a) enables finer-grained prediction, while with computation overhead. (b) adds in skip connections for better training and is widely adopted in pixel-level prediction tasks. (c) adds in another stream which is of full resolution for fine pixel level predictions, but demands more computation cost and memory utilization.

we start with CNN architecture design and implementation details, then describe novel objective functions for training.

2.1 Learning with Unified Framework

There are many learning based methods in the community of machine learning and computer vision. Convolutional neural networks (CNN) have become one of the best choice as it is elegant in architecture design, easy to train, fast in computation and prototyping, and able to achieve state-of-the-art performance by leveraging large-scale annotated dataset. Now we choose CNN architecture as the unified framework, train the models for some per-level prediction tasks as a review on the technical background.

As we know, per-pixel labeling tasks require pixel-level predictions; so it is desired that the output from the CNN models are of high-resolution, or even with the same size as the input. However, CNN architectures contain some pooling operations (either pooling layers or convolution layers with strides) to reduce memory cost and capture spatially long-range information. For example, the ResNet architecture [115] contains five 2×2 pooling layers, meaning that the output is $2^5 = 32$ smaller than the input in resolution. We can clearly see a dilemma caused by the two combating aspects. To solve this discrepancy, the first remedy is to upsample the output by removing some pooling layers. Following [45, 167], we increase the output resolution of ResNet by removing the top global 7×7 pooling layer and the last two 2×2 pooling layers, replacing them with atrous convolution with dilation rate 2 and 4, respectively, to maintain the same spatial sampling rate. Then such modifications produce output at 1/8 the input resolution. Moreover, we find it better to include another convolutional layer(s) after upsampling, smoothing the output and reducing the grid effects.

As a second improvement, the U-architecture is widely adopted that treats normal CNN model as an encoder, and adds in a decoder to enlarge the output progressively towards a full-resolution output. Within the U-shape architecture, there are skip connections and upsampling layers to ensure the smooth (anti-aliasing) operation, and boost gradient flow for more effective training.

The encoder architecture is typically based on existing CNN models for image classification which makes it possible to initialize with weights that have been pre-trained (e.g., using ImageNet). It has systematically studied and in literature various CNN architectures, in terms of comparisons w.r.t performance and computation FLOPS. Though they achieve different performance, ResNet [115] is shown to be one of the top-performing one, achieving quite good performance and stable across different vision tasks. Therefore, we choose in this thesis the ResNet as encoder, which is also pre-trained on ImageNet [63].

Besides upsampling layers, it is also common to apply skip connection formulated as convolution layers to bridge long-range layers, through layer concatenation or sum without further parameters to train. This is helpful in multiple aspects, such as better backpropagating the gradient from very top layers to bottom ones [115, 365, 195], using low-level features to smoothen the high-level layers' upsampled output [99, 285, 101], allowing for intermediate loss [195, 211], etc.

Note that the U-shape architecture may still be far from sufficient to deal with grid/aliasing effect induced by upsampling (e.g., bilinear interpolation). As the third improvement, we also consider a full-resolution stream to augment the output from the U-net [164, 269, 168]. We note this is quite useful in high-precision output especially on the boundaries, but the computation burden becomes expensive in both wall-clock time and memory usage.

Note that we adopt the ResNet architecture as our basic degin throughout this thesis, but the exact ResNet-based models in the rest chapters vary slightly depending on the specific tasks. In this chapter, we simply train the modified ResNet model (with the first improvement only) without U-shape structure. This model produces 8x smaller output in resolution. We postpone other macro design choices until the last two chapters, which demand high-precision reconstruction at pixel level.

◻ 2.2 Example Problems with Loss Function for Training

We train the CNN model for the following four pixel-level prediction tasks. Besides describing each task briefly, we include the loss functions used for training, some of which are novel ones proposed by us. We analyze why these new loss functions are better than other widely used in literature.

Semantic Segmentation

Semantic segmentation refers to the process of linking each pixel in an image to a class label. The class label can be person, vehicle, building, trees, etc. Essentially, we can think of semantic segmentation as image classification at pixel level. As semantic segmentation groups pixels according to the class label for understanding where and what in the image, it becomes very useful in applications like autonomous vehicles, human-computer interaction, robotics, and photo editing/creativity tools. It also enables applications requiring counting, e.g., the number of people at a metro entry for security and surveillance, the nematodes in biological images for accelerated analysis, etc.

For semantic segmentation, we, and others in literature, commonly train a CNN model using cross-entropy loss as in [45, 167]:

$$\ell_{semantic} = -\sum_{i}\sum_{c}^{K} 1_{[y_i=c]} \cdot \log(C_i) \tag{2.1}$$

where C_i is the class prediction (from a softmax transform) at pixel i, and y_i is the ground-truth class label.

Boundary Detection

The great importance of boundary detection as an early stage for more complicated image processing or computer vision tasks is known since many years [42, 98, 234, 7]. Boundary detection complements image segmentation in that: image segmentation fails to provide accurate information about the shape and the position of the objects in the image, while boundary can accurately determine object contours.

We train a base model using (binary) logistic loss. Following [358, 232, 166], we include four prediction branches at macro building blocks in ResNet [115] and a final fusion layer.

Therefore, we have five losses in total. To handle class imbalance (more than 80% pixels are non-boundaries), we utilize a weighted loss accumulated over the prediction losses given by:

$$\ell_{boundary} = \sum_{b \in \mathcal{B}} \Big(-\beta \sum_{j \in Y_+} \log(P(y_j = 1|\theta_b)) - (1-\beta) \sum_{j \in Y_-} \log(P(y_j = 0|\theta_b)) \Big) \tag{2.2}$$

where b indexes the branches, $\beta = |Y_-|/|Y_- \cup Y_+|$, and Y_+ and Y_- denote the set of boundary and non-boundary annotations, respectively.

Monocular Depth Estimation

Monocular depth estimation is to estimate per-pixel depth to the camera for the single 2D image. It is a crucial step in scene reconstruction, 3D object recognition, segmentation and detection. While depth estimation can be done with stereo camera or the light-field, or sensors like LiDAR for outdoor or kinect for indoor, monocular depth estimation can augment sparse sensor data (e.g., from LiDAR) and occluded regions where stereo/ligh-field cannot capture. Moreover, the model learned for monocular depth estimation can also be used as initialized model for downstream tasks [149].

For monocular depth estimation, we use combined L_2 and L_1 losses to compare the predicted and ground-truth depth maps $\mathbf{D}$ and $\hat{\mathbf{D}}$ on a log scale:

$$\ell_{depth} = \sum_{i=1} \| \log(\mathbf{D}_i) - \log(\hat{\mathbf{D}}_i)\|_2^2 + \gamma \| \log(\mathbf{D}_i) - \log(\hat{\mathbf{D}}_i)\|_1 \tag{2.3}$$

where $\gamma = 2$ controls the relative importance of the two losses. This mixed loss penalizes large errors quadratically (the $\mathcal{L}_2$ term) while still assuring a non-vanishing gradient that continues to drive down small errors (the $\mathcal{L}_1$ term). The idea behind our loss is similar to the reverse Huber loss as used in [188], which in fact can be understood as concatenation of truncated $\mathcal{L}_2$ and $\mathcal{L}_1$ loss. However, the reverse Huber loss requires specifying a hyperparameter for the boundary between $\mathcal{L}_2$ and $\mathcal{L}_1$; we find our mixed loss is robust and performs well with $\gamma \in [1, 5]$.

We note that there are other loss functions transforming the depth regression problem into a classification problem, and show better results [90, 64]. We do not explore the technical details specific to monocular depth estimation, but focus on generalization of the unified framework.

Surface Normal Estimation

In computer vision or graphics, surface normal, or normal for short, to a surface at a point is a vector perpendicular to the tangent plane of the surface at this point. The normal is often used in 3D computer graphics to determine a surface's orientation toward a light source for flat shading; or in robotics to determine the angle for a robot to approach to an object. Strictly speaking, surface normal is the derivative of depth at each pixel, and given a perfect depth map, one can derive the surface normal of the scene or object. However, depth maps are not perfect and often come with holes (missing values due to sensor limitations) [116], so we would like to learn to predict per-pixel normal over a single image. Moreover, it also shows surface normal is "easier" to predict through learning [370], thus has the potential to boost other tasks.

To predict surface normals, we insert a final L_2 normalization layer so that predicted normals have unit Euclidean length. In the literature, cosine distance is often used in the loss function to train the model (implemented with inner product), while performance metrics for normal estimation measure the angular difference between prediction $\mathbf{n}$ and the target normal $\hat{\mathbf{n}}$ [88, 75]. We note that this discrepancy is largely overlooked in literature. To address this, we propose to incorporate inverse cosine distance along with cosine distance as our objective

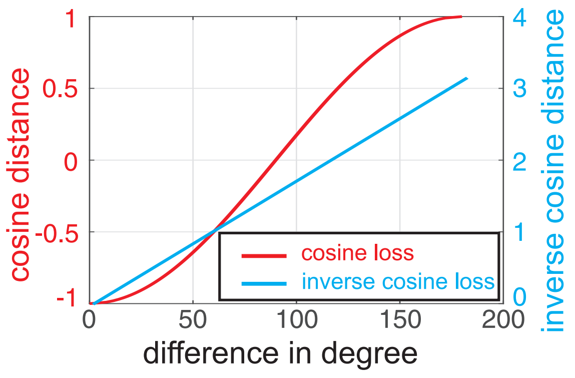


Figure 2.3: Comparison of cosine loss and inverse cosine loss. Inverse cosine loss has a constant gradient, while the gradient of the widely used cosine loss decreases as the prediction errors become small, preventing further model refinement.

function:

$$\ell_{normal} = \sum_i -\mathbf{n}_i^T \hat{\mathbf{n}}_i + \lambda \cos^{-1}(\mathbf{n}_i^T \hat{\mathbf{n}}_i) \tag{2.4}$$

where λ controls the importance of the two part and we set $\lambda = 4$ throughout our experiments. Fig. 2.3 compares the curves of the two losses, and we can clearly see that the inverse cosine loss always produces meaningful gradients, whereas the popular cosine loss has "vanishing gradient" issue when prediction errors become small (analogous to the mixed L_1/L_2 loss for depth).

▢ 2.3 Experimental Result

We train the CNN models for these tasks over some public benchmark datasets. We now describe the datasets, and then show the training models and some quantitative results with brief comparison to baselines with our novel loss functions, as well as the state-of-the-art. Finally, we visualize the results and understand the limitations, which motivate our research in next chapters.

▨ 2.3.1 Datasets

BSDS500 [7] is the most popular dataset for boundary detection. It provides a standard split [7, 358] of 300 train images and 200 test images.

NYUv2 [301] consists of 1,449 RGB-D indoor scene images of the resolution 640×480 which include color and pixel-wise depth obtained by a Kinect sensor. We use the ground-truth segmentation into 40 classes provided in [105] and a standard train/test split into 795 and 654 images, respectively. For surface normal estimation, we compute the normal as target from depth using the method in [301] by fitting least-squares planes to neighboring sets of points in the point cloud.

Stanford-2D-3D [9] contains 1,559 RGB panoramic images with depths, surface normal and semantic annotations covering six large-scale indoor areas from three different buildings. We use area 3 and 4 as a validation set (489 panoramas) and the remaining four areas for training (1,070 panoramas). The panoramas are very large (2,048×4,096) and contain black void regions at top and bottom due to the spherical panoramic topology. We rescale them by 0.5 and crop out the central two-thirds ($y \in [160, 863]$) resulting in final images of size 704×2,048-pixels. We randomly crop out sub-images of 704×704 resolution for training. Note that the surface normals in panoramic images are relative to the global coordinate system which cannot be determined from the image alone. Thus we transformed this global normal into local normal specified relative to the camera viewing direction (details in appendix). Note that such relative normals are also useful in scene understanding and reconstruction.

Cityscapes [57] contains high-quality pixel-level annotations of images collected in street scenes from 50 different cities. We use the standard split of training set (2,975 images) and validation set (500 images) for testing, respectively, labeled for 19 semantic classes as well as depth obtained by disparity. The images are of high resolution (1,024×2,048), and we randomly crop out sub-images of 800×800 resolution during training.

Table 2.1: Benchmark comparison of our baseline model with state-of-the-art methods We report the standard F-measure at Optimal Dataset Scale (odsF), Optimal Image Scale (oisF), and the Average Precision (AP) for boundary detection.

methods	odsF	oisF	AP
baseline	0.790	0.806	0.826
HED [358]	0.780	0.790	0.834
COB [232]	0.793	0.820	0.859
LEP [248]	0.757	0.793	0.828
MCG [8]	0.747	0.779	0.759
MShift [56]	0.601	0.644	0.493
gPb-UCM [7]	0.726	0.760	0.727
NCut [299]	0.641	0.674	0.447
EGB [80]	0.636	0.674	0.581

2.3.2 Results

We evaluate the trained baseline models for different tasks over different datasets. The goal of these experiments is to establish:

1. the baseline model achieves comparable performance to the state-of-the-art methods on diverse tasks;
2. (ablation) study on our designed loss functions that perform better than the popular ones used in literature.

2.3.3 Comprehensive Benchmark Comparison

We carry out experimental comparisons with a few state-of-the-art methods, demonstrating that our baselines achieve comparable results. We will show more results in the rest chapters for detailed analysis of our specific methods for specific tasks.

Taking boundary detection as the first task, we quantitatively compare our model to COB [232], HED [358], LEP [248], UCM [7], NCuts [299], EGB [80], MCG [8] and the mean shift (MShift) algorithm [56]. Fig. 2.1 shows comparison to all the methods, demonstrating our model

Table 2.2: Semantic segmentation is measured by Intersection over Union (IoU), pixel accuracy (acc), and iIoU that leverages the size of segments w.r.t categories. Results marked by † are from our trained models with the released code.

	NYUv2 [301]		Stanford-2D-3D [9]		Cityscapes [301]	
methods/metrics	IoU	pixel acc.	IoU	pixel acc.	IoU	iIoU
baseline	42.1	71.1	79.5	92.1	73.8	54.7
DeepLab [45]	—	—	$69.8^\dagger$	$88.0^\dagger$	71.4	51.6

Table 2.3: Depth estimation is measured by standard threshold accuracy, *i.e.* the percentage (%) of predicted pixel depths d_i s.t. $\delta = \max(\frac{d_i}{\hat{d}_i}, \frac{\hat{d}_i}{d_i}) < \tau$, where $\tau = \{1.25, 1.25^2, 1.25^3\}$. Methods with * use $\sim$100k extra images to train.

	NYUv2 [301]			Stanford-2D-3D [9]			Cityscapes [301]		
methods/metric ($\delta < \tau$)	1.25	1.25^2	1.25^3	1.25	1.25^2	1.25^3	1.25	1.25^2	1.25^3
baseline	71.1	93.2	98.5	73.1	92.1	97.5	29.0	53.8	75.8
Eigen* [75]	61.4	88.8	97.2	—	—	—	—	—	—
Eigen* [74]	76.9	95.0	98.8	—	—	—	—	—	—

achieves comparable performance to the state-of-the-art methods. Note that our model has the same backbone architecture of HED [358], indicating our implementation is correct. Our model performs on par with COB [232], which also has the same backbone architecture but uses auxiliary losses for oriented boundary detection. Note that it is possible to surpass human performance with sophisticated techniques [161], but we don't pursue this as it is out the scope of this thesis.

Table 2.2, 2.3 and 2.4 show the comprehensive comparisons on the tasks of semantic segmentation, monocular depth and surface normal estimation, respectively. Our baseline models achieve performance on par with recent methods for all tasks. For depth and surface normal estimation tasks, our baseline models also perform very well. This is notable since we don't leverage multi-task learning (unlike Eigen [74]) and do not use extra images to augment training set (unlike most methods for depth estimation using $\sim$100k extra images to augment the training set as shown in Table 2.3). We attribute this to the carefully designed losses for depth and surface normal estimation.

Table 2.4: Surface normal estimation is measured by mean angular error and the percentage of prediction error within t° degree where $t = \{11.25, 22.50, 30.00\}$. Smaller ang. err. mean better performance as marked by $\downarrow$.

	NYUv2 [301]				Stanford-2D-3D [9]			
methods/metrics	ang. err.$\downarrow$	11.25°	22.50°	30.00°	ang. err.$\downarrow$	11.25°	22.50°	30.00°
baseline	22.3	34.4	62.5	74.4	19.0	51.5	68.6	76.3
Eigen [74]	22.2	38.6	64.0	73.9	—	—	—	—

Table 2.5: Study of loss functions and PAG unit for monocular depth estimation on NYUv2 dataset. Our MP@Res5 model is the base model, unless specified, all the models are trained with softmax weighted average in the MultiPool module. The performance is measured by standard threshold accuracy, *i.e.* the percentage of predicted pixel depths d_i s.t. $\delta = \max(\frac{d_i}{\hat{d}_i}, \frac{\hat{d}_i}{d_i}) < \tau$, where $\tau = \{1.25, 1.25^2, 1.25^3\}$.

metrics	$L2$ loss	$L1$ loss	$L1$+$L2$ loss	$L1$+$L2$ loss
1.25	0.737	0.743	0.745	**0.751**
1.25^2	0.939	0.942	0.944	**0.944**
1.25^3	0.986	0.987	0.988	**0.988**

Then, we study the loss function mixing $L1$ and $L2$. We train the models using different loss functions, and report the results in Table 2.5. It's clear to see the mixed $L1$ and $L2$ loss leads to the best performance, especially on the metric of < 1.25, focusing on the range of small prediction errors where $L2$ loss alone is unable to provide a meaningful gradient.

In Table 2.6, we compare the results from models trained with different loss functions. We can see the combination of cosine distance loss and the inverse cosine loss achieves the best performance. From the table, we clearly see that the improvement on metric 11.25° is more remarkable, which focuses on small prediction errors. This is because the combined loss function provides meaningful gradient "everywhere", whereas the cosine distance loss alone has "vanishing gradient" issue when the prediction errors become small.

■ 2.3.4 Qualitative Visualization

We briefly visualize the prediction and attentional maps in Figure 2.1, the very beginning of this chapter. From the figure, we can have a sense on how well the baseline models

Table 2.6: Study of the loss functions for surface normal estimation over NYUv2 dataset. Performance is measured by mean angular error (ang. err.) and the portion of prediction error within $t°$ degree where $t = \{11.25, 22.50, 30.00\}$. Smaller ang. err. means better performance as marked by ↓.

metrics	cosine distance $(-\mathbf{n}^T\hat{\mathbf{n}})$	inverse cosine $(\cos^{-1}\mathbf{n}^T\hat{\mathbf{n}})$	cosine and inverse cosine
ang. err.↓	23.3462	23.1191	**22.7170**
11.25°	0.3163	0.3279	**0.3382**
22.50°	0.5995	0.6093	**0.6195**
30.00°	0.7240	0.7302	**0.7383**

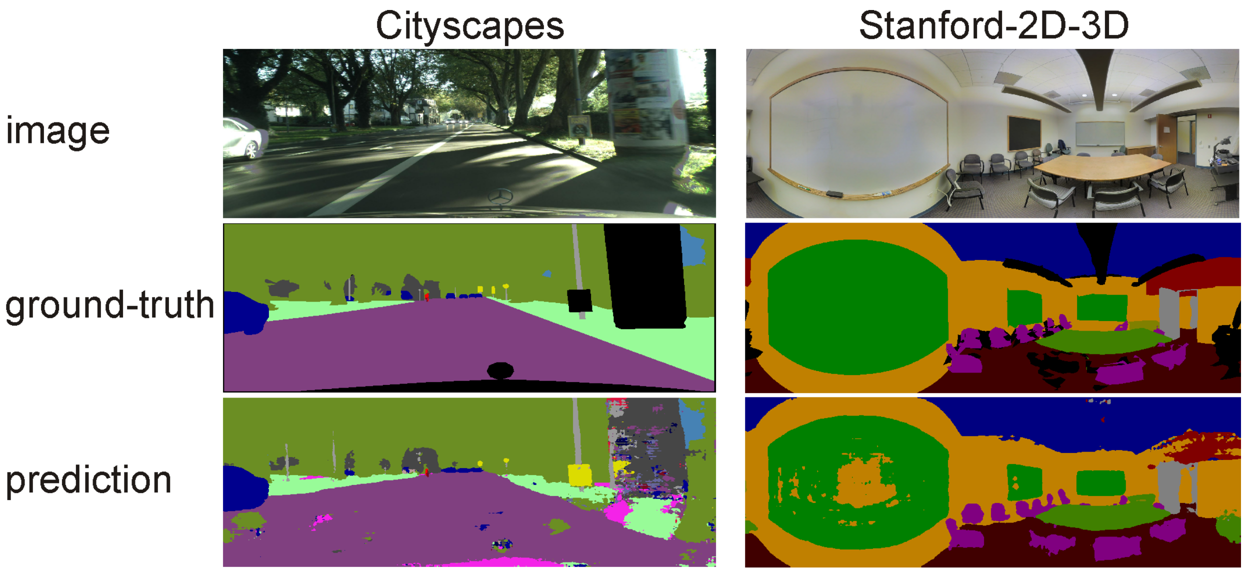


Figure 2.4: Visualization of some failure cases in semantic segmentation, such as the messy predictions in the Cityscapes image and the "holes" on the wall in Stanford-2D-3D image. We note the model fails more easily in such perspective images, where both small and large objects exist in a single image. This observation motivates us to investigate how to better exploit contextual information for each pixel.

performance, and where they fail. To have a closer look at the failure cases, we show two examples from semantic segmentation in Figure 2.4. From the two perspective images, we can see the model easily fail when both small and large objects exist in the image. This makes sense as objects of varying sizes pose challenges to the model which uses the same receptive field for every pixel. Therefore we hypothesize that pixels from different objects should have different pooling sizes, i.e., receptive fields, to better aggregate contextual information for better prediction. To this end, we introduce our next chapter for how to allocate dynamic receptive field to each pixel.

Chapter 3

Depth-aware Gating for Contextual Pooling

Objects may appear at arbitrary scales in perspective images of a scene, posing a challenge for recognition systems that process images at a fixed resolution, see Figure 2.4 in Chapter 2. In this chapter, we propose a depth-aware gating module that adaptively selects the pooling field size in a convolutional network architecture according to the object scale (inversely proportional to the depth) so that small details are preserved for distant objects while larger receptive fields are used for those nearby. The depth gating signal is provided by stereo disparity or estimated directly from monocular input. We integrate this depth-aware gating into a recurrent convolutional neural network to perform semantic segmentation. Our recurrent module iteratively refines the segmentation results, leveraging the depth and semantic predictions from the previous iterations.

Through extensive experiments on four popular large-scale RGB-D datasets, we demonstrate this approach achieves competitive semantic segmentation performance with a model which is substantially more compact. We carry out extensive analysis of this architecture including variants that operate on monocular RGB but use depth as side-information during training, unsupervised gating as a generic attentional mechanism, and multi-resolution gating. We find that gated pooling for joint semantic segmentation and depth yields state-of-the-art

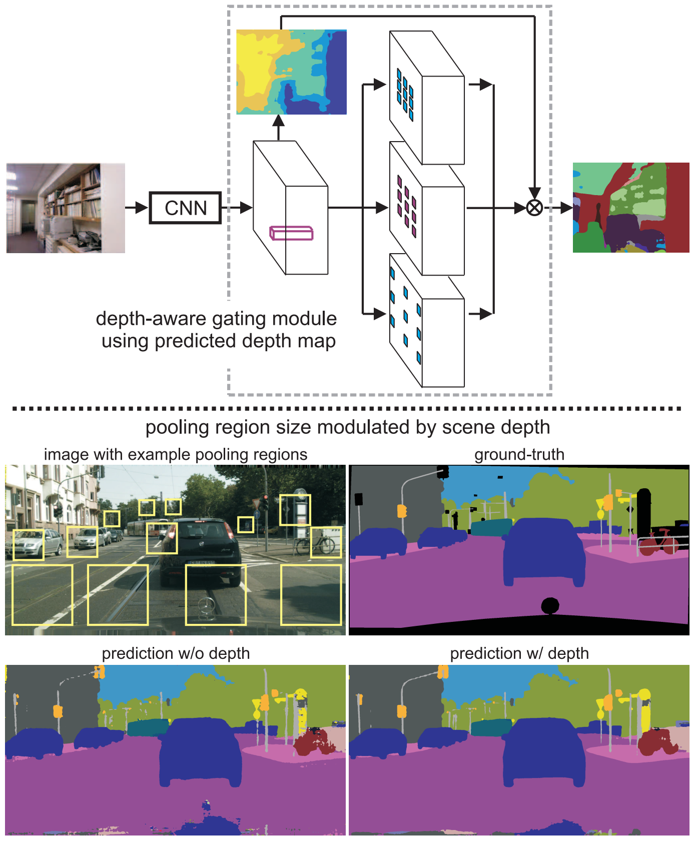

CNN
depth-aware gating module
using predicted depth map
pooling region size modulated by scene depth
image with example pooling regions
ground-truth
prediction w/o depth
prediction w/ depth

Figure 3.1: **Upper**: depth-aware gating spatially modulates the selected pooling scale using a depth map predicted from monocular input. In the chapter, we also evaluate related architectures where scene depth is provided directly at test time as a gating signal, and where spatially adaptive attentional gating is learned without any depth supervision. **Lower**: example ground-truth compared to predictions with and without the depth gating module. Rectangles overlayed on the image indicate pooling field sizes which are adapted based on the local depth estimate. We quantize the depth map into five discrete scales in our experiments. Using depth-gated pooling yields more accurate segment label predictions by avoiding pooling across small multiple distant objects while simultaneously allowing using sufficiently large pooling fields for nearby objects.

results for quantitative monocular depth estimation.

3.1 Background

An intrinsic challenge of parsing rich scenes is understanding object layout relative to the camera. Roughly speaking, the scales of the objects in the image frame are inversely proportional to the distance to the camera. Humans easily recognize objects even when they range over many octaves of spatial resolution, e.g., the cars near the camera in urban scene can appear a dozen times larger than those at distance as shown by the lower panel in Figure 3.1. However, the huge range and arbitrary scale at which objects appear pose difficulties for machine image understanding. Although individual local features (e.g., in a deep neural network) can exhibit some degree of scale-invariance, it is not obvious this invariance covers the range scale variation that exists in images.

In this chapter, we investigate how cues to perspective geometry conveyed by image content (estimated from stereo disparity, or measured directly via specialized sensors) might be exploited to improve recognition and scene understanding. We focus specifically on the task of semantic segmentation which seeks to produce per-pixel category labels.

One straightforward approach is to stack the depth map with RGB image as a four-channel input tensor which can then be processed using standard architectures. In practice, this RGB-D input has not proven successful and sometimes even results in worse performance [113, 225]. We conjecture including depth as a per-pixel input doesn't adequately address scale-invariance in learning; such models lack an explicit mechanism to generalize to depths not observed during training and hence still require training examples with object instances at many different scales to learn a multiscale appearance model.

Instead, our method takes inspiration from the work of [184], who propose using depth estimates to rescale local image patches to a pre-defined canonical depth prior to analysis. For patches contained within a fronto-parallel surface, this can provide true depth-invariance over a range of scales (limited by sensor resolution for small objects) while effectively augmenting the training data available for the canonical depth. Rather than rescaling the input image, we propose a depth gating module that adaptively selects pooling field sizes over higher-level feature activation layers in a convolutional neural network (CNN). Adaptive pooling works with a more abstract notion of scale than standard multiscale image pyramids which operate on input pixels. This gating mechanism allows spatially varying processing over the visual field which can capture context for semantic segmentation that is not too large or small, but “just right”, maintaining details for objects at distance while simultaneously using much larger receptive fields for objects near the camera. This gating architecture is trained with a loss that encourages selection of target pooling scales derived from “ground-truth” depth but at test time makes accurate inferences about scene depth using only monocular cues.

Inspired by studies of human visual processing (e.g., [54]) that suggest dynamic allocation of computation depending on the task and image content (background clutter, occlusion, object scale), we propose embedding gated pooling inside a recurrent refinement module that takes initial estimates of high-level scene semantics as a top-down signal to reprocess feed-forward representations and refine the final scene segmentation (similar to the recurrent module proposed in [20] for human pose). This provides a simple implementation of “Biased Competition Theory” [19] which allows top-down feedback to suppress irrelevant stimuli or incorrect interpretations, an effect we observe qualitatively in our recurrent model near object boundaries and in cluttered regions with many small objects.

We train this recurrent adaptive pooling CNN architecture end-to-end and evaluate its performance on several scene parsing datasets. The monocular depth estimates produced by our gating channel yield state-of-the-art performance on the NYU-depth-v2 benchmark [301]. We also find that using this gating signal to modulate pooling inside the recurrent refinement architecture results in improved semantic segmentation performance over fixed multiresolution pooling. We also compare to gating models trained without depth supervision where the gating signal acts as a generic attentional signal that modulates spatially adaptive pooling. While this works well, we find that depth supervision results in best performance. The resulting system matches state-of-the-art segmentation performance on four large-scale datasets using a model which, thanks to recurrent computation, is substantially more compact than many existing approaches.

3.2 Related work

Starting from the "fully convolutional" architecture of [221], there has been a flurry of recent work exploring CNN architectures for semantic segmentation and other pixel-labeling tasks [166]. The seminal DeepLab [45] model modifies the very deep residual neural network [115] for semantic segmentation using dilated or atrous convolution operators to maintain spatial resolution in high-level feature maps. To leverage features conveying finer granularity lower in the CNN hierarchy, it has proven useful to combine features across multiple layers (see e.g., FCN [221], LRR [99] and RefineNet [208]). To simultaneously cover larger fields-of-view and incorporate more contextual information, [385] concatenates features pooled over different scales.

Starting from the work of [122, 289], estimating depth from (monocular) scene semantics has been examined in a variety of indoor and outdoor settings (see e.g., [196]). Accurate monocular depth estimation using a multiscale deep CNN architecture was demonstrated by [75] using a geometrically inspired regression loss. Follow-on work [74] showed that depth, surface orientation and semantic labeling predictions can benefit each other in a multi-task setting using a shared network model for feature extraction.

The role of perspective geometry and geometric context in object detection was emphasized by a line of work starting with [123] and others (e.g., [18]) and has played an increasingly important role, particularly for scene understanding in urban environments [95]. We were inspired by [184], who showed reliable depth recovery from image patches (i.e., without vanishing point estimation) and that the resulting depths could be used to estimate object scale and improve segmentation in turn. Chen et al. [47] used an attention gating mechanism to combine predictions from CNN branches run on rescaled images (multi-resolution), a natural but computationally expensive approach that we compare experimentally to our proposal (multi-pool).

Finally, there have been a number of proposals to carry out high-level recognition tasks such as human pose estimation [202, 20] and semantic segmentation [284] using recurrent or iterative processing. As pixel-wise labelling tasks are essentially a structured prediction problem, there has also been a related line of work that aims to embed unrolled conditional random fields or mean shift into differentiable CNN architectures to allow for more tractable learning and inference (e.g., [386, 166]).

3.3 Depth-aware Gating Module

Our depth-aware gating module utilizes estimated depth at each image location as a proxy for object scale in order to select the appropriate spatial extent over which to pool features. Informally speaking, for a given object category (e.g., cars) the size of an object in the image is inversely proportional to the distance from the camera. Thus, if a region of an image has a larger depth values, the windows over which features are pooled (pooling field size) should be smaller in order to avoid pooling responses over many small objects and capture details needed to precisely segment small objects. For regions with small depth values, the same object will appear much larger and the pooling field size should be scaled up in a covariant manner to capture sufficient contextual appearance information in the vicinity of the object.

This depth-aware gating can readily utilize depth maps derived from stereo disparity or specialized time-of-flight sensors. Such depth maps typically contain missing data and measurement noise due to oblique view angle, reflective surface and occlusion boundary. While these estimates can be improved using more extensive off-line processing (e.g., [307]), in our

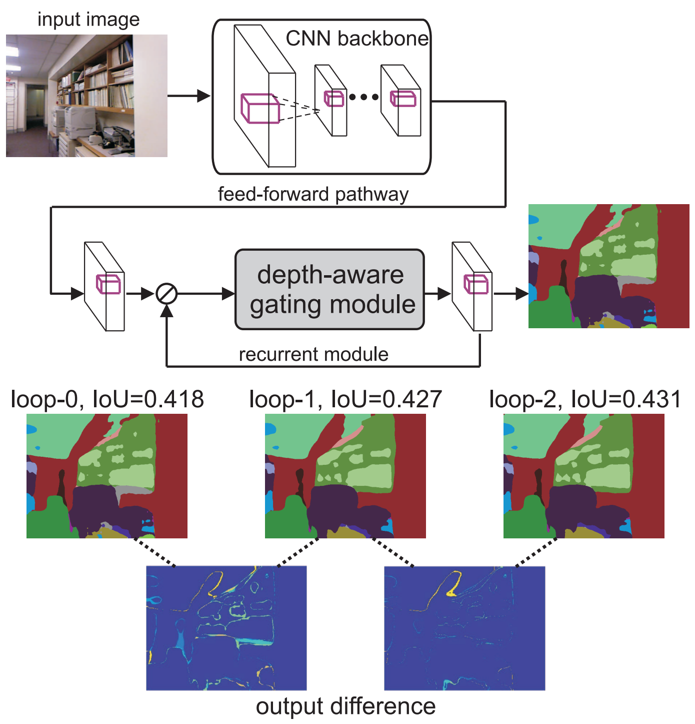


Figure 3.2: The input to our recurrent module is the concatenation (denoted by ⊘) of the feature map from an intermediate layer of the feed-forward pathway with the prior recurrent prediction. Our recurrent module utilizes depth-aware gating which carries out both depth regression and quantized prediction. Updated depth predictions at each iteration gate pooling fields used for semantic segmentation. This recurrent update of depth estimation increases the flexibility and representation power of our system yielding improved segmentation. We illustrate the prediction prior to, and after two recurrent iterations for a particular image and visualize the difference in predictions between consecutive iterations which yield small but notable gains as measured by average intersection-over-union (IoU) benchmark performance.

experiments we use these "raw" measurements. When depth measurements are not available, the depth-aware gating can instead exploit depth estimated directly from monocular cues. The upper panel of Figure 3.1 illustrates the architecture of our depth-aware gating module using monocular depth predictions derived from the same front-end feature extractor.

Regardless of the source of the depth map, we quantize the depth into a discrete set of predicted scales (5 in our experiments). The scale prediction at each image location is then used to multiplicatively gate between a set of feature maps computed with corresponding pooling regions and summed to produce the final feature representation for classification [170, 163]. In the depth gating module, we use atrous convolution with different dilation rates to produce the desired pooling field size on each branch.

When training a monocular depth prediction branch, we quantize the ground-truth depth and treat it as a five-way classification using a softmax loss. For the purpose of quantitatively evaluating the accuracy of such monocular depth prediction, we also train a depth regressor over the input feature of the module using a simple Euclidean loss for the depth map $\mathbf{D}$ in log-space:

$$\ell_{depthReg}(\mathbf{D}, \mathbf{D}^*) = \frac{1}{|M|} \sum_{(i,j)\in M} \| \log(\mathbf{D}_{ij}) - \log(\mathbf{D}_{ij})^* \|_2^2,$$

where $\mathbf{D}^*$ is the ground-truth depth. Since our "ground-truth" depth may have missing entries, we only compute the loss over pixels inside a mask M which indicates locations with valid ground-truth depth. For benchmarking we convert the log-depth predictions back to depths using an element-wise exponential. Although more specific depth-oriented losses have been explored [75, 74], we show in experiment that this simplistic Euclidean loss on log-depth achieves state-of-the-art monocular depth estimation when combined with our architecture for semantic segmentation.

In our experiments, we evaluate models based on RGB-D images (where the depth channel is used for gating) and on RGB images using the monocular depth estimation branch. We also evaluated a variant which is trained monocularly (without the depth loss) where the gating can be viewed as a generic attentional mechanism. In general, we find that using predicted (monocular) depth to gate segmentation feature maps yields better performance than models using the ground-truth depth input. This is a surprising, but desirable outcome, as it avoids the need for extra sensor hardware and/or additional computation for refining depth estimates from multiple video frames (e.g., [307]).

3.4 Recurrent Refinement Module

It is natural that scene semantics and depth may be helpful in inferring each other. To achieve this, our recurrent refinement module takes as input feature maps extracted from a feed-forward CNN model along with current segmentation predictions available from previous iterations of the recurrent module. These are concatenated into a single feature map. This allows the recurrent module to provide an anytime segmentation prediction which can be dynamically refined in future iterations. The recurrent refinement module has multiple convolution layers, each of which is followed by a ReLU and batch normalization layers. We also use depth-aware gating in the recurrent module, allowing the refined depth output to serve as a top-down signal for use in refining the segmentation (as shown in experiments below). Figure 3.2 depicts our final recurrent architecture using the depth-aware gating module inside.

For a semantic segmentation problem with K semantic classes, we use a K-way softmax classifier on individual pixels to train our network. Our final multi-task learning objective function utilizes multiple losses weighted by hyperparameters:

$$\ell = \sum_{l=0}^{L} (\lambda_s \ell^l_{segCls} + \lambda_r \ell^l_{depthReg} + \lambda_c \ell^l_{depthCls}), \tag{3.1}$$

where L means we unroll the recurrent module into L loops and $l = 0$ denotes the prediction from the feed-forward pathway. The three losses ℓ^l_{segCls}, $\ell^l_{depthReg}$ and $\ell^l_{depthCls}$ correspond to the semantic segmentation, depth regression and quantized depth classification loss at iteration l, respectively. We train our system in a stage-wise procedure by varying the hyper-parameters λ_s, λ_r and λ_c, as detailed in Section 3.5, culminating in end-to-end training using the full objective. As our primary task is improving semantic segmentation, in the final training stage we optimize only ℓ^l_{segCls} and drop the depth side-information (setting $\lambda_r = 0$ and $\lambda_c = 0$).

▢ 3.5 Implementation

We implement our model with the MatConvNet toolbox [326] and train using SGD on a single Titan X GPU. We use the pre-trained ResNet50 and ResNet101 models [115] as the backbone of our models*. To increase the output resolution of ResNet, like [45], we remove the top global 7×7 pooling layer and the last two 2×2 pooling layers. Instead we apply atrous convolution with dilation rate 2 and 4, respectively to maintain a spatial sampling rate which is of 1/8 resolution to the original image size (rather than 1/32 resolution if all pooling layers are kept). To obtain a final full resolution segmentation prediction, we simply apply bilinear interpolation on the softmax class scores to upsample the output by a factor of eight.

We train our models in a stage-wise procedure. First, we train a feed-forward baseline model for segmentation. The feed-forward module is similar to DeepLab [45], but we add two additional 3×3-kernel layers (without atrous convolution) on top of the ResNet backbone. Starting from this baseline, we train depth estimation branch and replace the second 3×3-kernel layer with the depth prediction and depth-aware gating module. We train the recurrent refinement module (containing the depth-aware gating), unrolling one layer at a time, and fine-tune the whole system using the objective function of Eq. 3.1.

We augment the training set using random data transforms. Specifically, we use random scaling by $s \in [0.5, 2]$, in-plate rotation by degrees in $[-10°, 10°]$, random left-right flip with 0.5 probability, random crop with sizes around 700×700 divisible by 8, and color jittering. Note that when scaling the image by s, we also divide the depth values by s. All these data transforms can be performed in-place with minimal computational cost.

Throughout training, we set batch size to one where the batch is a single input image (or a crop of a very high-resolution image). Due to this small batch size, we freeze the batch

*Code and models are available here: `http://www.ics.uci.edu/~skong2/recurrentDepthSeg`.

normalization in ResNet backbone during training, using the same constant global moments in both training and testing. We use the “poly” learning rate policy [45] with a base learning rate of $2.5e-4$ scaled as a function of iteration by $(1-\frac{iter}{maxiter})^{0.9}$.

3.6 Experiments

To show the effectiveness of our approach, we carry out comprehensive experiments on four large-scale RGB-D datasets (introduced below). We start with a quantitative evaluation of our monocular depth predictions, which achieve state-of-the-art performance. We then compare our complete model with the existing methods for semantic segmentation on these datasets, followed by ablation experiments to determine whether our depth-aware gating module improves semantic segmentation, validate the benefit of our recurrent module, and compare among using ground-truth depth, predicted depth, and unsupervised attentional gating. Finally, we show some qualitative results.

3.6.1 Datasets and Benchmarks

For our primary task of semantic segmentation, we use the standard Intersection-over-Union (IoU) criteria to measure the performance. We also report the per-pixel prediction accuracy for the first three datasets to facilitate comparison to existing approaches.

NYUD-depth-v2 [301] consists of 1,449 RGB-D indoor scene images of the resolution 640×480 which include color and pixel-wise depth obtained by a Kinect sensor. We use the ground-truth segmentation into 40 classes provided in [105] and a standard train/test split into 795 and 654 images respectively.

SUN-RGBD [307] is an extension of NYUD-depth-v2 [301], containing 5,285 training images and 5,050 testing images. It provides pixel labelling masks for 37 classes, and depth

Table 3.1: Depth prediction on NYU-depth-v2 dataset.

Metric $\delta <$	Ladicky [184]	Liu [219]	Eigen [75]	Eigen [74]	Laina [188]	Ours	Ours -blur
1.25	0.542	0.614	0.614	0.769	0.811	0.809	0.816
1.25^2	0.829	0.883	0.888	0.950	0.953	0.945	0.950
1.25^3	0.940	0.971	0.972	0.988	0.988	0.986	0.989

maps using different depth cameras. While this dataset provides refined depth maps (exploiting depth from the neighborhood video frames), the ground-truth depth maps still have significant noisy/mislabled depth (examples can be found in our supplemental material).

Cityscapes [57] contains high quality pixel-level annotations of images collected in street scenes from 50 different cities. The training, validation, and test sets contain 2,975, 500, and 1,525 images respectively labeled for 19 semantic classes. The images of Cityscapes are of high resolution (1024×2048), which makes training challenging due to limited GPU memory. We randomly crop out sub-images of 800×800 resolution for training.

Stanford-2D-3D [9] contains 1,559 panoramas as well as depth and semantic annotations covering six large-scale indoor areas from three different buildings. We use area 3 and 4 as a validation set (489 panoramas) and the remaining four areas for training (1,070 panoramas). The panoramas are very large (2048×4096) and contain black void regions at top and bottom due to the spherical panoramic topology. For the task of semantic segmentation, we rescale them by 0.5 and crop out the central two-thirds ($y \in [160, 863]$) resulting in final images of size 704×2048-pixels.

▢ 3.6.2 Depth Prediction

In developing our approach, accurate depth prediction was not the primary goal, but rather generating a quantized gating signal to select the pooling field size. However, to validate our depth prediction, we also trained a depth regressor over the segmentation backbone

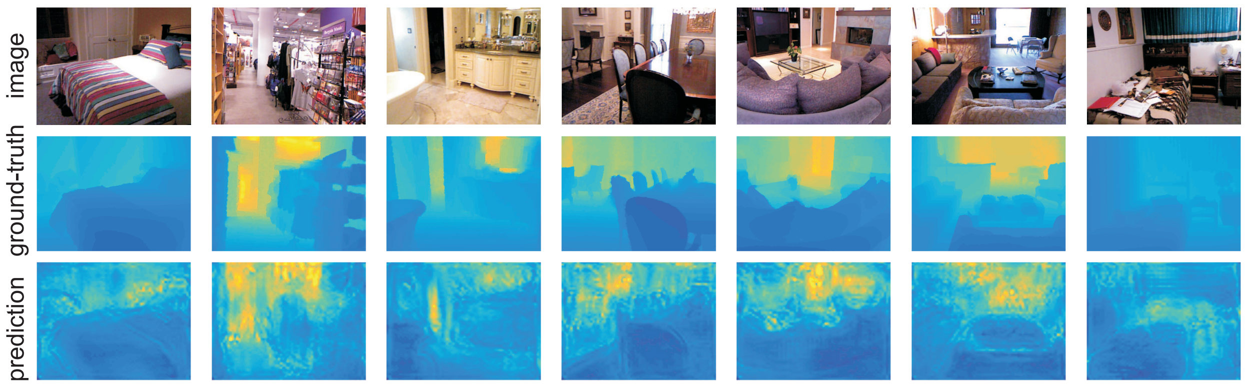


Figure 3.3: Examples of monocular depth predictions. First row: the input RGB image; second row: ground-truth; third row: our result. In our visualizations, all depth maps use the same fixed (absolute) colormap to represent metric depth.

and compared the resulting predictions with previous work. We evaluated our model on NYU-depth-v2 dataset, on which a variety of depth prediction methods have been tested. We report performance using the standard threshold accuracy metrics, i.e., the percentage of predicted pixel depths d_i s.t. $\delta = \max(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}) < \tau$, evaluated at multiple thresholds $\tau = \{1.25, 1.25^2, 1.25^3\}$.

Table 3.1 provides a quantitative comparison of our predictions with several published methods. We can see our model trained with the Euclidean loss on log-depth is quite competitive and achieves significantly better performance in the $\delta < 1.25$ metric. This simplistic loss compares well to, e.g., [74] who develop a scale-invariant loss and use first-order matching term which compares image gradients of the prediction with the ground-truth, and [188] who develop a set of sophisticated upsampling layers over a ResNet50 model.

In Figure 3.3, we visualize our estimated depth maps on the NYU-depth-v2 dataset[†]. Visually, we can see our predicted depth maps tend to be noticeably less smooth than true depth. Inspired by [74] who advocate modeling smoothness in the local prediction, we also

[†]We also evaluate our depth prediction on SUN-RGBD dataset, and achieve 0.754, 0.899 and 0.961 by the three threshold metrics. As SUN-RGBD is an extension of NYU-depth-v2 dataset, it has similar data statistics resulting in similar prediction performance. Examples of depth prediction on SUN-RGBD dataset can be found in the supplementary material.

apply Gaussian smoothing on our predicted depth map. This simple post-process is sufficient to outperform the state-of-the-art. We attribute the success of our depth estimator to two factors. First, we use a deeper architecture (ResNet50) than that in [74] which has generally been shown to improve performance on a variety vision tasks as reported in literature. Second, we train our depth prediction branch jointly with features used for semantic segmentation. This is essentially a multi-task problem and the supervision provided by semantic segmentation may understandably help depth prediction, explaining why our blurred predictions are as good or better than a similar ResNet50-based approach which utilized a set of sophisticated upsampling layers [188].

▢ 3.6.3 Semantic Segmentation

To validate the proposed depth-aware gating module and the recurrent refinement module, we evaluate several variants over our baseline model. We list the performance details in the first group of rows in Table 3.2. The results are consistent across models trained independently on the four datasets. Adding depth maps for gating feature pooling brings noticeable boost in segmentation performance, with greatest improvements especially on the large-perspective datasets Cityscapes and Stanford-2D-3D.

Interestingly, we achieve slightly better performance using the predicted depth map rather than the provided ground-truth depth. We attribute this to three explanations. Firstly, the predicted depth is smooth without holes or invalid entries. When using raw depth, say on Cityscapes and Stanford-2D-3D[‡], we assign equal weight on the missing entries so that the gating actually averages the information at different scales. This average pooling might be harmful in some cases such as a very small object at a distance. Secondly, the predicted depth maps show some object-aware patterns (e.g., car region shown in the visualization in Figure 3.7), which might be helpful for class-specific segmentation. Thirdly, the model is

[‡]NYU-depth-v2 and SUN-RGBD datasets provide improved depth maps without invalid entries.

Table 3.2: Performance of semantic segmentation on different datasets. Results marked by † are from our trained model models with the released code, and results marked by * are evaluated by the dataset server on test set. Note that we train our models based on ResNet50 architecture on indoor datasets NYU-depth-v2 and SUN-RGBD, and ResNet101 on the large perspective datasets Cityscapes and Stanford-2D-3D.

	NYU-depth-v2 [301]		SUN-RGBD [301]		Stanford-2D-3D [9]		Cityscapes [57]
	IoU	pixel acc.	IoU	pixel acc.	IoU	pixel acc.	IoU
baseline	0.406	0.703	0.402	0.776	0.644	0.866	0.738
w/ gt-depth	0.413	0.708	0.422	0.787	0.730	0.897	0.753
w/ pred-depth	0.418	0.711	0.423	0.789	0.742	0.900	0.759
loop1 w/o depth	0.419	0.706	0.432	0.793	0.744	0.901	0.762
loop1 w/ gt-depth	0.425	0.711	0.439	0.798	0.747	0.902	0.769
loop1 w/ pred-depth	0.427	0.712	0.440	0.798	0.753	0.906	0.772
loop2	0.431	0.713	0.443	0.799	0.760	0.908	0.776
loop2 (test-aug)	0.445	0.721	0.451	0.803	0.765	0.910	0.791 / 0.782*
DeepLab [45]	-	-	-	-	0.698†	0.880†	0.704 / 0.704*
LRR [99]	-	-	-	-	-	-	0.700 / 0.697*
Context [209]	0.406	0.700	0.423	0.784	-	-	- / 0.716*
PSPNet [385]	-	-	-	-	0.674†	0.876†	- / 0.784*
RefineNet-Res50 [208]	0.438	-	-	-	-	-	- / -
RefineNet-Res101 [208]	0.447	-	0.457	0.804	-	-	- / 0.736*
RefineNet-Res152 [208]	0.465	0.736	0.459	0.806	-	-	- / -

trained end-to-end so co-adaption of the depth prediction and segmentation branches may increase the overall representation power and flexibility of the whole model, benefiting the final predictions.

Table 3.2 also shows the benefit of the recurrent refinement module as shown by improved performance from baseline to loop1 and loop2. Equipped with depth in the recurrent module, the improvement is more notable. As with the pure feed-forward model, using predicted depth maps in the recurrent module yields slight gains over the ground-truth depth. We observe that performance improves using a depth 2 unrolling (third group of rows in Table 3.2) but saturates/converges after two iterations.

In comparing with state-of-the-art methods, we follow common practice of augmenting images at test time by running the model on flipped and rescaled variants and average the class scores to produce the final segmentation output (compare loop2 and loop2 (test-aug)). We can see our model performs on par or better than recently published results listed in Table 3.2.

Note that for NYU-depth-v2 and SUN-RGBD, our backbone architecture is ResNet50, whereas RefineNet reports the results using a much deeper models (ResNet101 and ResNet152) which typically outperform shallower networks in vision tasks. For the Cityscapes, we also submitted our final result for held-out benchmark images which were evaluated by the Cityscapes benchmark server. Our model achieves IoU 0.782, on par with the best published result, IoU 0.784, by PSPNet.[§] We did not perform any extensive performance tuning and only utilized the fine-annotation training images for training (without the twenty thousand coarse-annotation images and the validation set). We also didn't utilize any post-processing (such as the widely used fully-connected CRF [175] which typical yields additional performance increments).

[§]We compare to performance using train only rather than train+val which improved PSPNet performance to 0.813.

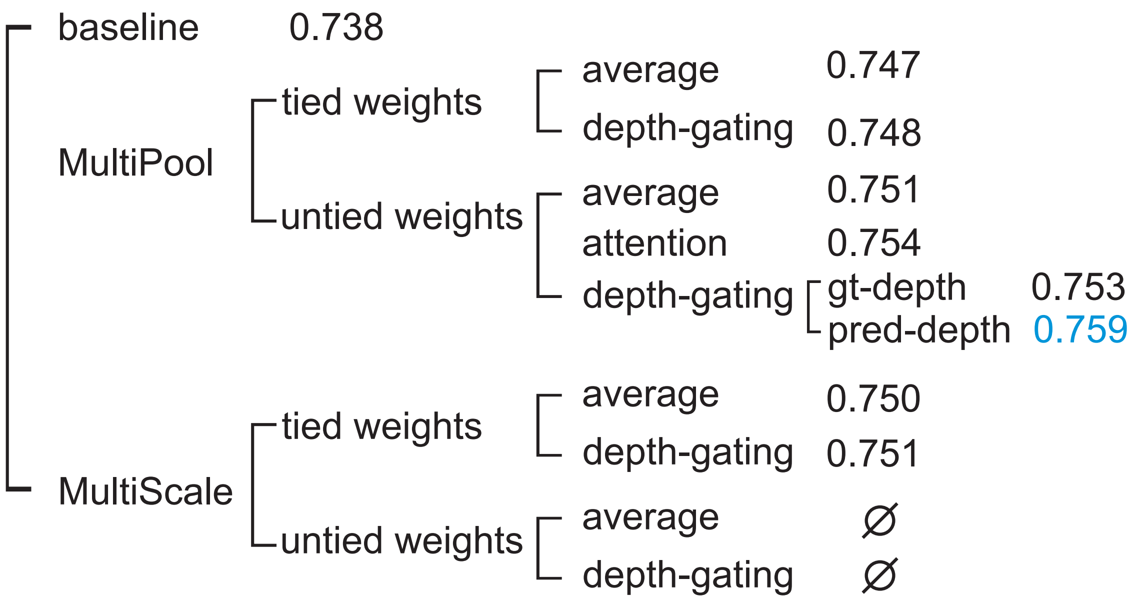


Figure 3.4: Performance comparisons across gating architectures including tied vs untied parameters across different branches, averaging vs gating branch predictions, using monocular predicted vs ground-truth depth for the gating signal, gating pooling region size (MultiPool) or rescaling input image (MultiScale), and gating without depth supervision during training (attention).

One key advantage of recurrent refinement is that it allows richer computation (and better performance) without additional model parameters. Our ResNet50 model (used on the NYU-depth-v2 dataset) is relatively compact (221MB) compared to RefineNet-Res101 which achieves similar performance but is nearly double the size (426MB). Our model architecture is similar to DeepLab which also adopts pyramid atrous convolution at multiple scales of inputs (but simply averages output feature maps without any depth-guided adaptive pooling). However, the final DeepLab model utilizes an ensemble which yields a much larger model (530MB). PSPNet concatenates the intermediate features into 4,096 dimension before classification while our model operates on small 512-dimension feature maps.

■ 3.6.4 Analysis of Gating Architectures Alternatives

We discuss the important question of whether depth-aware gating is really responsible for improved performance over baseline, or if gains are simply attributable to training a larger, richer architecture. We also contrast our approach to a number of related proposals in the literature. We summarize our experiments exploring these alternatives in Figure 3.4 (more details can be found in supplementary material).

We use the term *MultiPool* to denote the family of models (like our proposed model) which process the input image at a single fixed scale, but perform pooling at multiple convolutional dilate rate at high level layers. For a multi-pool architecture, we may choose to learn independent *untied* weights across the scale-specific branches or use the same *tied* weights. As an alternative to our gating function, which selects a spatially varying weighted combination of the scale-specific branches, we can simply *average* the branches (identical at all spatial locations).

We can contrast MultiPool with the *MultiScale* approach, which combines representations or predictions from multiple branches where each branch is applied to a scaled version of the input image[¶]. Many have adopted this strategy as a test time heuristic to boost performance by simply running the same model (tied) on different scaled versions of the input and then averaging the predictions. Others, such as DeepLab [45], train multiple (untied) models and use the average ensemble output.

In practice, we found that both MultiPool and MultiScale architectures outperform baseline and achieve similar performance. While MultiScale processing is conceptually appealing, it has a substantial computational overhead relative to MultiPool processing (where early computation is shared among branches). As a result, it was not feasible to train untied MultiScale models end-to-end on a single GPU memory constraints. As a result, we found that the untied, depth-gated model performed the best (and was adopted in our final approach).

Finally, we explored use of the gated pooling where the gating was trained without the

[¶]The roots of this idea can be traced back to early work on scale-space for edge detection (see, e.g. [23, 214])

depth loss. We refer to this as an *attention* model after the work of [47]. The attention model achieves surprisingly good performance, even outperform gating using ground-truth depth. We show the learned attention map in Figure 3.5, which behaves quite differently from depth gating. Instead, the gating signal appears to encode the distance from object boundaries. We hypothesize this selection mechanism serves to avoid pooling features across different semantic segments while still utilizing large pooling regions within each region. Our fine-tuned model using predicted depth-gating (instead of ground-truth depth) likely benefits from this adaption.

3.6.5 Qualitative Results

In Figures 3.6 and 3.7, we depict several randomly selected examples from the test set of NYU-depth-v2, Cityscapes and Stanford-2D-3D. We visualize both the segmentation results and the depth maps updated across multiple recurrent iterations. Interestingly, the depth maps on Cityscapes and Stanford-2D-3D change more noticeably than those on NYU-depth-v2 dataset. In Cityscapes, regions in the predicted depth map corresponding to objects, such as the car, are grouped together and disparity estimates on texture-less regions such as the street surface improve across iterations, while in Stanford-2D-3D, depth estimate for adaptation suggests that the recurrent module is performing coarse-to-fine segmentation (where later iterations shift towards a smaller pooling regions as semantic confidence increases).

Gains for the NYU-depth-v2 data are less apparent. We conjecture this is because images in NYU-depth-v2 are more varied in overall layout and often have less texture and fewer objects from which the model can infer semantics and subsequently depth. In all datasets, we can see that our model is able to exploit recurrence to correct misclassified regions/pixels "in the loop", visually demonstrating the effectiveness of the recurrent refinement module.

3.7 Conclusion and Discussion

In this chapter, we have proposed a depth-aware gating module that uses depth estimates to adaptively modify the pooling field size at a high level layer of neural network for better segmentation performance. The adaptive pooling can use large pooling fields to include more contextual information for labeling large nearby objects, while maintaining fine-scale detail for objects further from the camera. While our model can utilize stereo disparity directly, we find that using such data to train a depth predictor which is subsequently used for adaptation at test-time in place of stereo ultimately yields better performance. We also demonstrate the utility of performing recurrent refinement which yields improved prediction accuracy for semantic segmentation without adding additional model parameters.

We envision that the recurrent refinement module can capture object shape priors, contour smoothness and region continuity. However, our current approach converges after a few iterations and performance saturates. This leaves open future work in exploring other training objectives that might push the recurrent computation towards producing more varied outputs. This might be further enriched in the setting of video where the recurrent component could be extended to incorporate memory of previous frames.

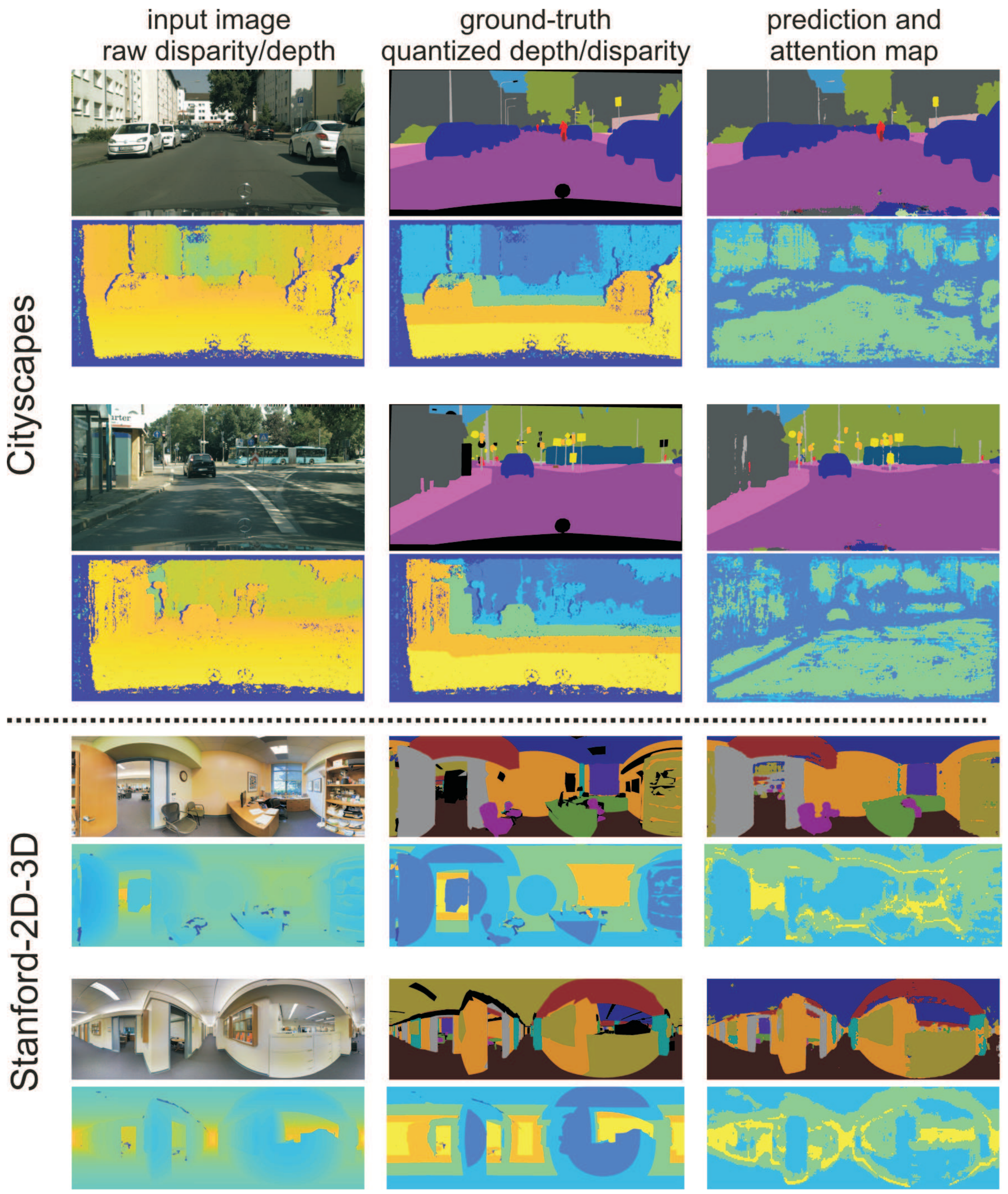


Figure 3.5: Visualization of the attention maps on random images from Cityscapes and Stanford-2D-3D. The raw disparity/depth maps and the quantized versions are also shown for reference. Though we train the attention branch with randomly initialized weights, we can see that the learned attention maps capture some depth information as well as encoding distance to object boundaries.

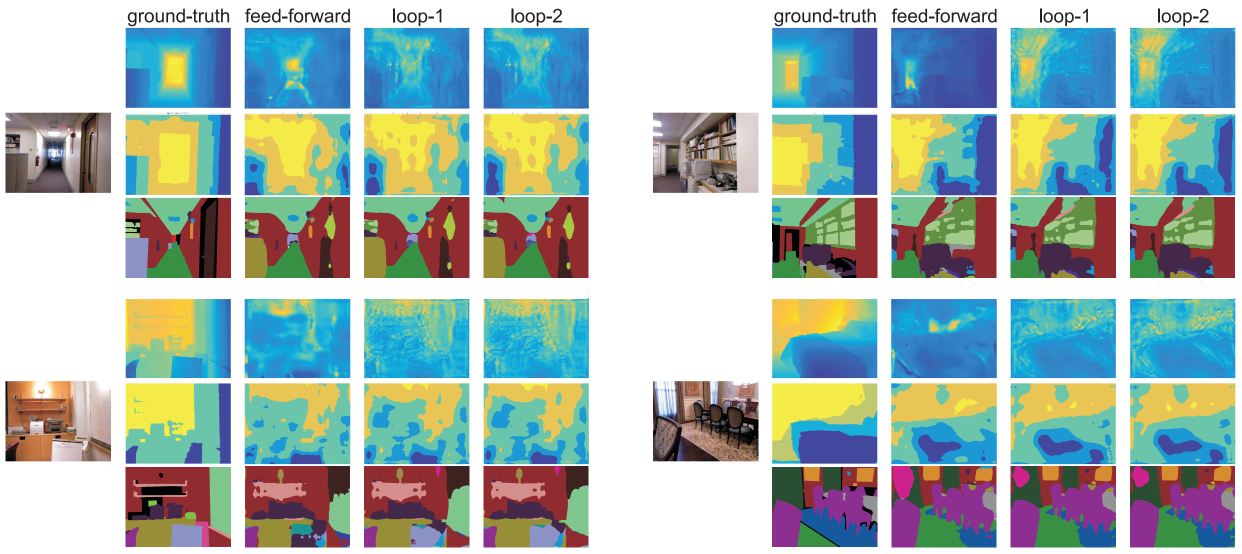


Figure 3.6: Visualization of the output on NYU-depth-v2. We show four randomly selected testing images with ground-truth and predicted disparity (first row), quantized disparity (second row) and segmentation (third row) at each iteration of the recurrent computation.

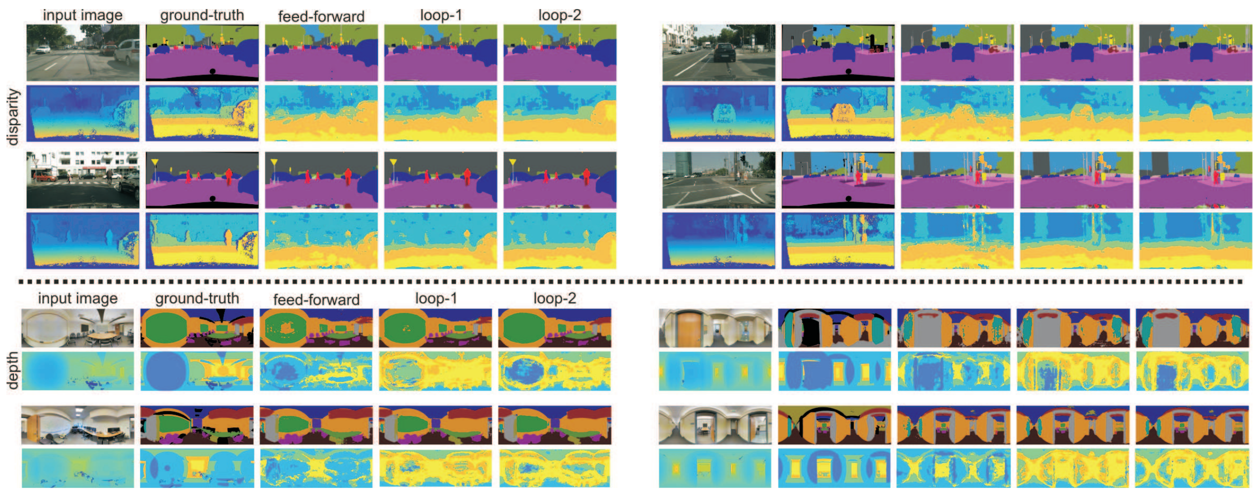


Figure 3.7: Visualization of randomly selected validation images from Cityscapes and Stanford-2D-3D with the segmentation output and the predicted quantized disparity at each iteration of the recurrent loop. We depict "ground-truth" continuous and quantized disparity beneath the input image. Our monocular disparity estimate makes predictions for reflective surfaces where stereo fails and recurrent iteration further improves estimates, particularly for featureless areas such as the pavement. Note that Cityscapes shows disparity while Stanford-2D-3D shows depth so the colormaps are reversed.

Chapter 4

Pixel-wise Attentional Gating for Dynamic Computation

In this chapter, we extend the depth-aware gating module to a more general one, called *Pixel-wise Attentional Gating* unit (*PAG*). PAG learns to allocate per-pixel computation route without heuristical supervision. As a benefit, PAG enables not only spatially dynamic computation for pooling contextual information at each pixel, but also allows for parsimonious computation.

PAG is a generic, architecture-independent, problem-agnostic mechanism that can be readily "plugged in" to an existing model with fine-tuning. We utilize PAG in two ways: 1) learning spatially varying pooling fields that improve model performance without the extra computation cost associated with multi-scale pooling, and 2) learning a dynamic computation policy for each pixel to decrease total computation while maintaining accuracy.

We extensively evaluate PAG on a variety of per-pixel labeling tasks, including semantic segmentation, boundary detection, monocular depth and surface normal estimation. We demonstrate that PAG allows competitive or state-of-the-art performance on these tasks. Our experiments show that PAG learns dynamic spatial allocation of computation over the input image which provides better performance trade-offs compared to related approaches (e.g., truncating deep models or dynamically skipping whole layers). Generally, we observe PAG can reduce computation by 10% without noticeable loss in accuracy and performance degrades gracefully when imposing stronger computational constraints.

4.1 Background

The development of deep convolutional neural networks (CNN) has allowed remarkable progress in wide range of image pixel-labeling tasks such as boundary detection [232, 358, 161], semantic segmentation [165, 167, 45], monocular depth estimation [167, 200, 188, 219, 75], and surface normal estimation [336, 17, 74]. Architectures that enable training of increasingly deeper networks have resulted in corresponding improvements in prediction accuracy [304, 115]. However, with great depth comes great computational burden. This hinders deployment of such deep models in applications such as mobile and robotics, which have significant power constraints and real-time processing requirements.

To make deep models more practically applicable, a flurry of recent work has focused on reducing these storage and computational costs [108, 242, 135, 230, 40, 163]. Static offline techniques like network distillation [117], pruning [242], and model compression [40] take a trained network as input and synthesize a new network that approximates the same functionality with reduced memory footprint and test-time execution cost. Our approach is inspired by a complementary family of techniques that learn to vary the network computation depth adaptively, depending on the input data [327, 354, 341, 83].

In this chapter, we study the problem of achieving parsimonious inference for per-pixel labeling tasks with a deep CNN model under limited computational budget. For image classification, dynamic allocation of computational "attention" can be interpreted as expending more computation on ambiguous images (e.g., [327, 354, 341]) or limiting processing to infor-

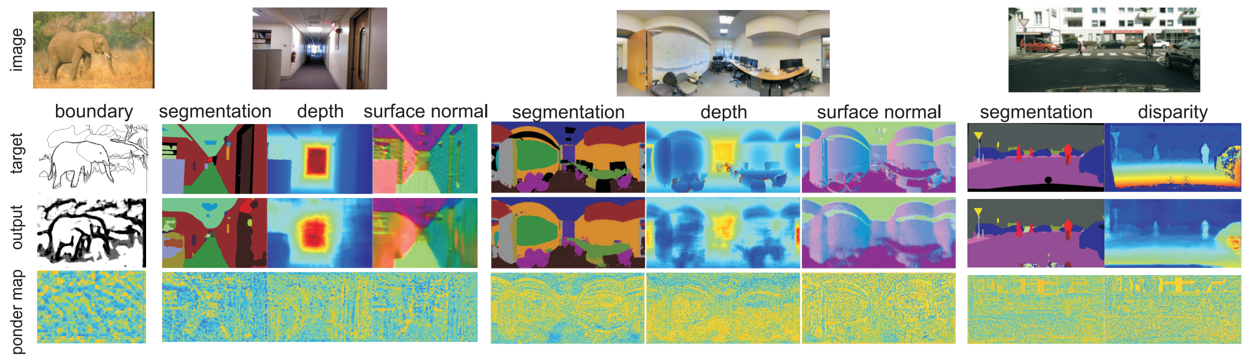


Figure 4.1: Pixel-wise Attentional Gating units (PAG) achieve parsimonious inference by learning a dynamic computation path for each pixel under limited computation budget. The "ponder cost" maps shown in the last row provide a visualization of the amount of computation allocated to each location (generated by accumulating binary masks from PAG units across all layers). We apply PAG to a variety of per-pixel labeling tasks (boundary detection, semantic segmentation, monocular geometry) and evaluate over diverse image datasets (indoor/outdoor scenes, narrow/wide field-of-view).

mative image regions (e.g., [83]). However, understanding the role of dynamic computation in pixel labeling tasks has not been explored. Pixel-level labeling requires analyzing fine-grained image details and making predictions at every spatial location, so it is not obvious that dynamically allocating computation to different image regions is useful. Unlike classification, labeling locally uninformative regions would seem to demand more computation rather than less (e.g., to incorporate long-range context).

To explore these questions, we introduce a *Pixel-wise Attentional Gating* (*PAG*) unit that selects a sparse subset of spatial locations to process based on the input feature map. We utilize the Gumbel sampling trick [104, 142, 227] to allow differentiable, end-to-end latent training of PAG units inserted across multiple computational blocks of a given task-specific architecture. We exploit this generic PAG unit in two ways: bypassing sequential (residual) processing layers and dynamically selecting between multiple parallel network branches.

Dynamic computation depth: Inserting PAG at multiple layers of a Residual Network enables learning a dynamic, feed-forward computation path for each pixel that is conditional on the input image. We introduce a sparsity hyperparameter that provides control over the average total and per-layer computation per-pixel. For a fixed computational budget, we show this dynamic, per-pixel gating outperforms architectures that meet the budget by using a smaller number of fixed layers or that learn to dynamically bypass whole layers (Section 4.3.3).

Dynamic spatial pooling: We exploit PAG to dynamically select the extent of pooling regions at each spatial image location. Previous work has demonstrated the benefits of averaging features from multiple pooling scales using either learned weights [45], or spatially varying weights based on attention [47] or scene depth [167]. However, such multi-scale pooling requires substantially more computation. We show the proposed PAG unit can learn to select appropriate spatially-varying pooling, outperforming the recent work of [167] without the computational burden of multiple parallel branches (Section 4.3.4).

We carry out an extensive evaluation of pixel-wise attentional gating over diverse datasets for a variety of per-pixel labeling tasks including boundary detection, semantic segmentation, monocular depth estimation and surface normal estimation (see Fig. 4.1). We demonstrate that PAG helps deliver state-of-the-art performance on these tasks by dynamically allocating computation. In general, we observe that the introduction of PAG units can reduce total computation by 10% without noticeable drop in accuracy and shows graceful degradation in performance even with substantial budget constraints (e.g., a 30% budget cut).

To summarize our primary contribution: (1) we introduce a pixel-wise attentional gating unit which is problem-agnostic, architecture-independent and provides a simple method to allow user-specified control computational parsimony with standard training techniques; (2) we investigate the role of dynamic computation in pixel-labeling tasks and demonstrate improved prediction performance while maintaining or reducing overall compute cost.

▢ 4.2 Related Work on Dynamic and Spatially Varying Computation

Deep CNN models with residual or "skip" connections have yielded substantial performance improvements with increased depth [115, 129], but also introduced redundant parameters and computation [108, 242]. In interpreting the success of residual networks (ResNet) [115], it has been suggested that ResNet can be seen as an ensemble of many small networks [328], each defined by a path through the network topology. This is supported by the observation that ResNet still performs well even when some layers are removed after training [130, 83]. This indicates it may be possible to reduce test-time computation by dynamically choosing only a subset of these paths to evaluate [327, 354, 341, 83].

This can be achieved by learning a halting policy that stops computation after evaluation of a particular layer [83], or a more flexible routing policy trained through reinforcement learning [354, 341]. Our method is most closely related to [327], which utilizes the "Gumbel sampling trick" [104, 142, 227] to learn binary gating that determines whether each layer is computed. The Gumbel sampling technique allows one to perform gradient descent on models that include a discrete argmax operation without resorting to approximation by softmax or reinforcement learning techniques.

The PerforatedCNN [84] demonstrated that convolution operations could be accelerated by learning static masks that skip computation at a subset of spatial positions. This was used in [83] to achieve spatially varying dynamic depth. Our approach is simpler (it uses a simple sparsity regularization to directly control amount per-pixel or per-layer computation rather than ponder cost) and more flexible (allowing more flexible routing policies than early halting*).

Finally, our use of dynamic computation to choose between branches is related to [167],

*Results in [354, 341] suggest general routing offers better performance than truncating computation at a particular depth.

which improves semantic segmentation by fusing features from multiple branches with various pooling sizes using a spatially varying weighted average. Unlike [45, 47, 167] which require computing the outputs of parallel pooling branches, our PAG-based learns to select a pooling size for each spatial location and only computes the necessary pooled features. This is similar in spirit to the work of [297], which demonstrated that sparsely-gated mixture-of-experts can dramatically increase model capacity using multi-branch configuration with only minor losses in computational efficiency.

▢ 4.3 Pixel-wise Attentional Gating

We first describe our design of Pixel-wise Attentional Gating (PAG) unit and its relation to the ResNet architecture [115]. Then, we elaborate how we exploit the Gumbel sampling technique to make learning PAG differentiable even when generating binary masks. Finally we describe how the PAG unit can be used to perform parsimonious inference by (1) selecting the subset of layers in the computational path for each spatial location, and (2) selecting the correct pooling size at each spatial location.

▢ 4.3.1 Plug-in PAG inside a Residual Block

Consider a block that computes output $\mathbf{O}$ using a residual update $\mathbf{Z} = \mathcal{F}(\mathbf{I})$ to some input $\mathbf{I}$. To reduce computation, one can learn a gating function $\mathcal{G}(\mathbf{I})$ that selects a subset of spatial locations (pixels) to process conditional on the input. We represent the output of $\mathcal{G}$ as a binary spatial mask $\mathbf{G}$ which is replicated along feature channel dimension as needed to match dimension of $\mathbf{O}$ and $\mathbf{I}$. The spatially gated residual update can be written as:

$$\begin{aligned}\mathbf{G} =& \mathcal{G}(\mathbf{I}) \\ \mathbf{O} =& \bar{\mathbf{G}} \odot \mathbf{I} + \mathbf{G} \odot (\mathcal{F}_{\mathbf{G}}(\mathbf{I}) + \mathbf{I}) \\ =& \mathbf{I} + \mathbf{G} \odot \mathcal{F}_{\mathbf{G}}(\mathbf{I})\end{aligned} \tag{4.1}$$

where $\odot$ is element-wise product, $\bar{\mathbf{G}} = 1 - \mathbf{G}$, and the notation $\mathcal{F}_{\mathbf{G}}$ indicates that we only evaluate $\mathcal{F}$ at the locations specified by $\mathbf{G}$. An alternative to spatially varying computation is for the gating function to predict a single binary value that determines whether or not the residual is calculated at this layer [327] in which case $\mathcal{F}_{\mathbf{G}}$ is only computed if $\mathbf{G} = 1$.

Both pixel-wise and layer-wise gating have the intrinsic limitation that the gating function $\mathcal{G}$ must be evaluated prior to $\mathcal{F}$. To overcome this limitation we integrate the gating function more carefully within the ResNet block. We demonstrate our approach in the equations below, comparing a standard residual block (left) and a PAG residual block (right) with corresponding illustrations in Fig. 4.2.

$$\begin{aligned}\mathbf{X} =& \mathcal{F}^1(\mathbf{I}) & \mathbf{X} =& \mathcal{F}^1(\mathbf{I}), \quad \mathbf{G} = \mathcal{G}(\mathbf{I}) \\ \mathbf{Y} =& \mathcal{F}^2(\mathbf{X}) & \mathbf{Y} =& \mathcal{F}^2_{\mathbf{G}}(\mathbf{X}) \\ \mathbf{Z} =& \mathcal{F}^3(\mathbf{Y}) & \mathbf{Z} =& \mathcal{F}^3_{\mathbf{G}}(\bar{\mathbf{G}} \odot \mathbf{X} + \mathbf{G} \odot \mathbf{Y}) \\ \mathbf{O} =& \mathbf{I} + \mathbf{Z} & \mathbf{O} =& \mathbf{I} + \mathbf{Z}\end{aligned} \tag{4.2}$$

The transformation functions $\mathcal{F}$'s consist of convolution, batch normalization [137] and ReLU [247] layers. As seen from the right set of equations, our design advocates computing the gating mask on the input $\mathbf{I}$ to the current building block in parallel with $\mathbf{X} = \mathcal{F}_{\mathbf{X}}(\mathbf{I})$. ResNet adopts bottleneck structure so the first transformation $\mathcal{F}^1$ performs dimensionality reduction with a set of 1×1 kernels, $\mathcal{F}^2$ utilizes 3×3 kernels, and $\mathcal{F}^3$ is another transform with 1×1 kernels that restores dimensionality. As a result, the most costly computation is

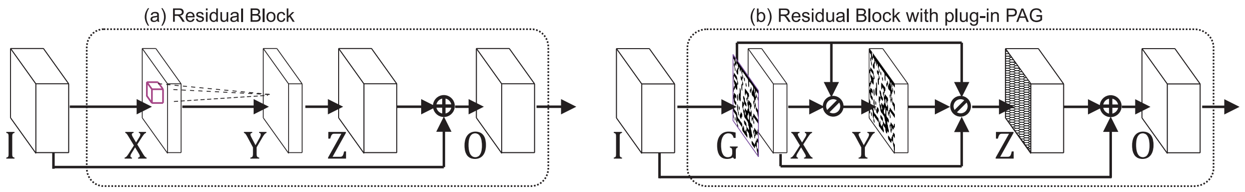


Figure 4.2: (a) A standard residual block. (b) Pixel-wise Attentional Gating unit (PAG) integrated into a residual block. Boxes/arrows denote activations/computations. **G** is a sparse, binary map that modulates what processing applied to each spatial location. "$\oslash$" means the perforated convolution [84], which assembles only active pixels for computation.

in the second transformation $\mathcal{F}^2$ which is mitigated by gating the computation. We show in our ablation study (Section 4.5.1) that for per-pixel labeling tasks, this design outperforms layer-wise gating.

4.3.2 Learning Discrete Attention Maps

The key to the proposed PAG is the gating function $\mathcal{G}$ that produces a discrete (binary) mask which allows for reduced computation. However, producing the binary mask using hard thresholding is non-differentiable, and thus cannot be simply incorporated in CNN where gradient descent is used for training. To bridge the gap, we exploit the Gumbel-Max trick [104] and its recent continuous relaxation [227, 142].

A random variable m follows a Gumbel distribution if $m \equiv -\log(-\log(u))$, where u is a sample from the uniform distribution $u \sim \mathcal{U}[0,1]$. Let g be a discrete random variable with probabilities $P(g=k) \propto a_k$, and let $\{m_k\}_{k=1,\ldots,K}$ be a sequence of i.i.d. Gumbel random variables. Then we can sample from the discrete variable with:

$$g = \underset{k=1,\ldots,K}{\text{argmax}}(\log \alpha_k + m_k) \tag{4.3}$$

The drawback of this approach is that the argmax operation is not continuous when mapping the Gumbel samples to the realizations of discrete distribution. To address this issue, a continuous relaxation the Gumbel Sampling Trick, proposed in [227, 142], replaces the argmax operation with a softmax. Using a one-hot vector $\mathbf{g} = [g_1, \dots, g_K]$ to encode g, a sample from the Gumbel softmax relaxation can be expressed by the vector:

$$\mathbf{g} = softmax((\log(\boldsymbol{\alpha} + \mathbf{m}))/\tau) \tag{4.4}$$

where $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_K]$, $\mathbf{m} = [m_1, \dots, m_K]$, and τ is the "temperature" parameter. In the limit as $\tau \to 0$, the softmax function approaches the argmax function and Eq. (4.4) becomes equivalent to the discrete sampler. Since the softmax function is differentiable and $\mathbf{m}$ contains i.i.d Gumbel random variables which are independent to input activation $\boldsymbol{\alpha}$, we can easily propagate gradients to the probability vector $\boldsymbol{\alpha}$, which is treated as the gating mask for a single pixel in the per-pixel labeling tasks.

As suggested in [327], we employ the straight-through version [227] of Eq. (4.4) during training. In particular, for the forward pass, we use discrete samples from Eq. (4.3), but during the backwards pass, we compute the gradient of the softmax relaxation in Eq. (4.4). Based on our empirical observation as well as that reported in [227], such greedy straight-through estimator performs slightly better than strictly following Eq. (4.4), even though there is a mismatch between forward and backward pass. In our work, we initialize $\tau = 1$ and decrease it to 0.1 gradually during training. We find this works even better than training with a constant small τ.

4.3.3 Dynamic Per-Pixel Computation Routing

By stacking multiple PAG residual blocks, we can construct a model in which the subset of layers used to a compute an output varies for each spatial location based the collection of binary masks. We allow the user to specify the computational budget in terms of a target sparsity ρ. For a binary mask $\mathbf{G} \in \{0, 1\}^{H \times W}$, we compute the empirical sparsity $g = \frac{1}{H*W} \sum_{h,w}^{H,W} \mathbf{G}_{h,w}$ (smaller values indicate sparser computation) and measure how well it matches the target ρ using the KL divergence:

$$KL(\rho\|g) \equiv \rho \log(\frac{\rho}{g}) + (1-\rho)\log(\frac{1-\rho}{1-g}) \tag{4.5}$$

To train the model, we jointly minimize the sum of a task-specific loss ℓ_{task} and the per-layer sparsity loss summed over all layers of interest:

$$\ell = \ell_{task} + \lambda \sum_{l=1}^{L} KL(\rho\|g_l) \tag{4.6}$$

where l indexes one of L layers which have PAG inserted for parsimonious computation and λ controls the weight for the constraints. In our experiments we set $\lambda = 10^{-4}$ but found performance is stable over a wide range of penalties ($\lambda \in [10^{-5}, 10^{-2}]$). To visualize the spatial distribution of computation, we accumulate the binary gating masks from all to produce a "ponder map". This reveals that trained models do not allocate computation uniformly, but instead responds to image content (*e.g.* focusing computation on boundaries between objects where semantic labels, depths or surface normals undergo sharp changes).

An alternative to per-layer sparsity is to compute the total sparsity $g = \frac{1}{L}\sum_{l=1}^{L} g_l$ and penalize g with $KL(\rho\|g)$. However, training in this way does not effectively learn dynamic computational paths and results in trivial, non-dynamic solutions, *e.g.* completely skipping a subset of layers and always using the remaining ones. Similar phenomenon is reported in [327]. In training models we typically start from a pre-trained model and insert sparsity constraints layer-by-layer. We found this incremental construction produces better diversity in the PAG computation paths. We also observe that when targeting reduced computation budget, fine-tuning a model which has already been trained with larger ρ consistently brings better performance than fine-tuning a pre-trained model directly with a small ρ.

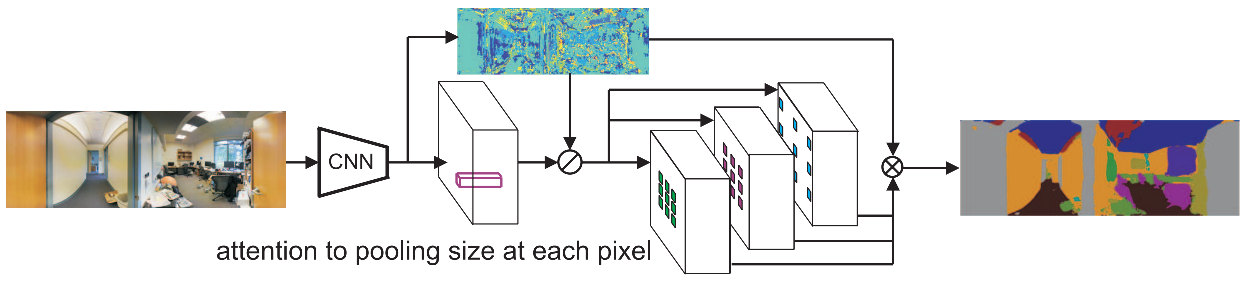


Figure 4.3: PAG-based MultiPool module learns to select the pooling size for each spatial location so that contextual information can be better aggregated. This can be implemented efficiently using perforated convolution [84], denoted by $\oslash$, which assembles only active pixels for computation in each pooling branch and thus avoids computing all pooled versions.

4.3.4 Dynamic Spatial Pooling

In pixel-labeling tasks, the ideal spatial support for analyzing a pixel can vary over the visual field in order to simultaneously maintain fine-grained details and capture context. This suggests an adaptive pooling mechanism at pixel level, or multi-scale pooling module (*MultiPool*) that chooses the appropriate pooling size for each pixel (see e.g., [167]). Given a collection of P pooled feature maps $\{\mathbf{M}_i\}_{i=1,\dots,P}$ computed with different pooling sizes, we can generate a MultiPool feature map $\mathbf{O} = \sum_i \mathbf{W}_i \odot \mathbf{M}_i$, where $\{\mathbf{W}_i\}_{i=1,\dots,P}$ are spatial selection masks, and $\odot$ indicates element-wise product between $\mathbf{W}_i$ and each channel of $\mathbf{M}_i$. We utilize the PAG to select the "correct" pooling region at each spatial location by applying Eq. (4.4). This MultiPool module, illustrated in Fig. 4.3, can be inserted in place of regular pooling with little computational overhead and learned in a latent manner using the task-specific loss (no additional sparsity loss).

We implement pooling using a set of 3×3-kernels applied at a set of user-specified dilation rates ($[0, 1, 2, 4, 6, 8, 10]$) [368]. A dilation rate of 0 means the input feature is simply copied into the output feature map. In our experiments, we observe that only a small portion of pixels are exactly copied for the final representation without being fed into any multi-pooling branches. Note that in Fig. 4.3, a multiplicative gating operation is shown for clarity, but an efficient implementation would utilize masking to directly select pixels in a matrix multiplication implementation of the convolutional layers in GPU or FPGA kernel [50, 253]. Our MultiPool module is thus distinct from [167] which use weighted averages over all intermediate feature activations from all branches for the final feature representation. Our approach selects a single pooling size for each pixel and hence does not require overhead of computing all branches.

4.4 Implementation and Training

As our PAG unit is agnostic to network architectures, the implementation is exactly the same as what described in Chapter 2.2. We briefly review the implementation here.

We utilize ResNet [115] pre-trained on ImageNet [63] as the base model. We follow [45, 167] and increase the output resolution of ResNet by removing the top global 7×7 pooling layer and the last two 2×2 pooling layers, replacing them with atrous convolution with dilation rate 2 and 4, respectively, to maintain a spatial sampling rate. Such a modification thus outputs predictions at 1/8 the input resolution. Rather than up-sampling the output (or downsampling the ground-truth) 8× for benchmarking [45, 167], we find it better to apply a deconvolution/upsampling layer followed by two or more convolutional layers before the final output. However, in [46], it trains with upsampled logits (the final output) with intact ground-truth annotation.

We augment the training sets with random left-right flips and random crops with 20-pixel margin and of size divisible by 8. When training the model, we fix the batch normalization, using the same constant global moments in both training and testing. This modification does not impact the performance and allows a batch size of one during training (a single input image per batch). We use the "poly" learning rate policy [45] with a base learning rate of 0.0002 scaled as a function of iteration by $(1 - \frac{iter}{maxiter})^{0.9}$. We adopt a stage-wise training

Table 4.1: Ablation study for where to insert the PAG-based MultiPool module. Experiments are from boundary detection and semantic segmentation on BSDS500 and NYUv2 dataset, measured by F-score ($F_{bnd.}$) and IoU(IoU$_{seg.}$) respectively. Numbers are in % (higher is better).

metrics	base.	Res3	Res4	Res5	Res6	Res4-5	Res3-5	Res4-6	Res5-6	Res3-6
$F_{bnd.}$	79.00	79.19	**79.19**	79.14	—	79.18	79.07	—	—	—
IoU$_{seg.}$	42.05	44.13	45.67	**46.52**	45.99	45.48	44.83	44.97	46.44	44.02

strategy over all tasks, *i.e.* training a base model, adding PAG-based MultiPool, inserting PAG for dynamic computation layer by layer, and finally decreasing ρ to achieve target computational budget. Since our goal is to explore computational parsimony in per-pixel labeling tasks, we implement our models without "bells and whistles", *e.g.* no utilization of ensembles, no CRF as post-processing, and no external training data. We implement our approach using the toolbox MatConvNet [326], and train using SGD on a single Titan X GPU[†].

4.5 Experiments

To evaluate our method based on PAG, we choose datasets that span a variety of per-pixel labeling tasks, including boundary detection, semantic segmentation, depth and surface normal estimation. As we have described the details of the tasks, datasets and loss functions in Chapter 2.2, we do not reiterate these in this chapter.

4.5.1 Analysis of Pixel-wise Attentional Gating

We evaluate different configurations of PAG on the BSDS500 and NYUv2 datasets for boundary detection and semantic segmentation (similar observations hold on other tasks, see appendix). The goal of these experiments is to establish:

[†]As MatConvNet itself does not provide perforated convolution, we release the code and models implemented with multiplicative gating at *https://github.com/aimerykong/Pixel-Attentional-Gating*.

Table 4.2: Computational parsimony compared with truncated ResNet and models learning to drop/skip whole layers. Evaluation is performed on NYUv2 dataset for semantic segmentation.

hyper param.	FLOPs	consumption	truncated		layer-skipping		MP@Res5 (PAG)	
ρ	$1e10$	%	IoU	acc.	IoU	acc.	IoU	acc.
$\rho = 0.5$	6.29	67.69	36.30	67.36	37.78	67.31	40.89	69.44
$\rho = 0.7$	8.27	86.20	37.69	67.44	39.84	69.00	43.61	71.41
$\rho = 0.9$	8.95	93.36	40.29	69.66	41.27	70.01	45.75	72.93
$\rho = 1.0$	9.63	100.00	—	—	—	—	46.52	73.50

1. whether our base model is comparable to state-of-the-art methods;
2. where to insert the PAG-based MultiPool module for the best performance;
3. how our PAG-based method for computational parsimony impacts performance, and how it performs compared with other related methods, *e.g.* truncated ResNet and methods learning to skip/drop layers.

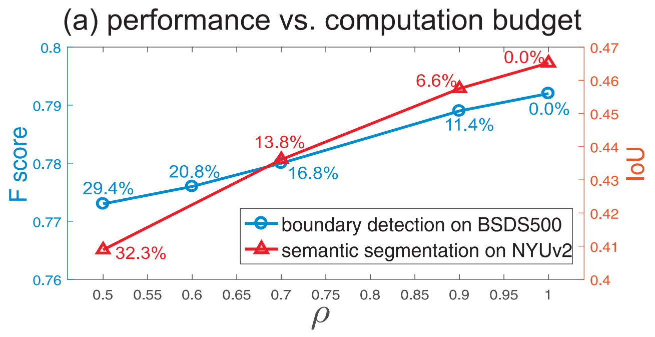
Base models: We train our base models as described in Section 4.4 without PAG units (the same as described in Chapter 2). The performance of our base model is on-par with state-of-the-art systems, achieving IoU=42.05% on NYUv2 for semantic segmentation (RefineNet [208] achieves IoU=44.5 with multi-resolution input), and $F = 0.79$ on BSDS500 for boundary detection (HED [358] achieves $F = 0.78$). More comprehensive comparisons with other related methods are shown later in Section 4.5.2.

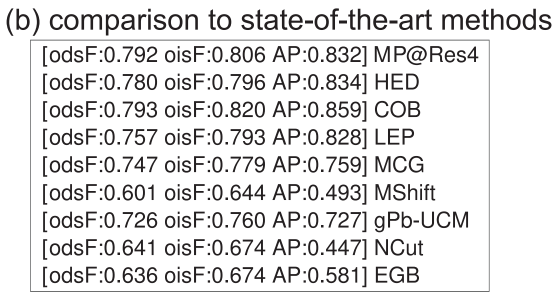
MultiPool: Table 4.1 explores the effect of inserting the MultiPool operation at different layers in the base model. In Table 4.1, Res6 means that we insert MultiPool module in the additional convolutional layers above the ResNet50 base. For boundary detection, we do not initialize more convolutional layers above the backbone, so there is no Res6. For both tasks, we observe that including a PAG-based MultiPool module improves performance, but including more than one MultiPool module does not offer further improvements. We find inserting MultiPool module at second last macro residual block (Res4 or Res5 depending on task) yields the largest gain.

For semantic segmentation, our MultiPool module also outperforms the weighted pooling in [167], which uses the same ResNet50 backbone. We conjecture this is due to three reasons. First, we apply the deconvolutional layer way before the last convolutional layer for softmax input as explained in Section 4.4. This increase resolution that enables the model to see better the fine details. Additionally, our set of pooling regions includes finer scales (rather than using powers of 2). Finally, the results in Table 4.3 show that PAG with binary masks performs slightly better (IoU=46.5 vs. IoU=46.3) than the (softmax) weighted average operation used in [167].

Computation-Performance Tradeoffs: Lastly, we evaluate how our dynamic parsimonious computation setup impacts performance and compare with other baselines. We show results on semantic segmentation on NYUv2 dataset in Table 4.2, comparing different baselines and our models with MultiPool at macro block Res5, *MP@Res5 (PAG)* for short, which are trained with different target computational budgets (specified by ρ). The "truncated" baseline means we simple remove top layers of ResNet to save computation, while "layer-skipping" is an implementation of [327] that learns to dynamically skip a subset of layer. For fair comparison, we insert MultiPool module at the top of both baselines. These results clearly suggest that the PAG approach outperforms the two baselines, demonstrating that learning dynamic computation path at the pixel level is helpful for per-pixel labeling tasks.

Fig. 4.4 (a) shows that as we decrease the computation budget, the performance of the PAG-based method degrades gracefully even as the amount of computation is scaled back to 70%, merely inducing 2.4% performance degradation (F score of 0.773 compared to the full model $F = 0.792$). Table 4.2 highlights the comparison to truncation and layer-skipping models adjusted to match the same computational budget as PAG. For these approaches, performance decays much more sharply with decreasing budget. These results also highlight that




(b) comparison to state-of-the-art methods

[odsF:0.792	oisF:0.806	AP:0.832]	MP@Res4
[odsF:0.780	oisF:0.796	AP:0.834]	HED
[odsF:0.793	oisF:0.820	AP:0.859]	COB
[odsF:0.757	oisF:0.793	AP:0.828]	LEP
[odsF:0.747	oisF:0.779	AP:0.759]	MCG
[odsF:0.601	oisF:0.644	AP:0.493]	MShift
[odsF:0.726	oisF:0.760	AP:0.727]	gPb-UCM
[odsF:0.641	oisF:0.674	AP:0.447]	NCut
[odsF:0.636	oisF:0.674	AP:0.581]	EGB

Figure 4.4: (a) Performance vs. computation budget (controlled by sparsity ρ) for boundary detection and segmentation tasks, % indicate computational savings relative to non-gated model. (b) Benchmark comparison of our PAG Multi-pool (MP@Res4) with state-of-the-art methods for boundary detection.

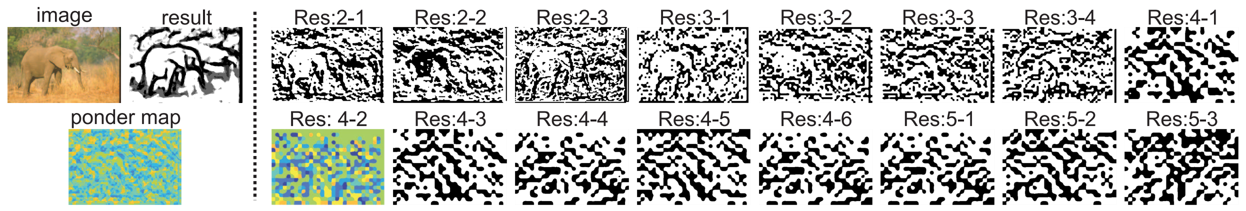


Figure 4.5: Visualization of sparse binary attention maps at each layer for boundary detection, together with the output and ponder map accumulating all binary maps. PAG-based MultiPool module is inserted at layer Res4-2, which is not included in the ponder map.

the target sparsity parameter ρ provides tight control over the actual average computation of the model.

Fig. 4.5 shows example binary masks at each layer for boundary detection (more results can be found in the appendix), as well as the final output and ponder map which shows cumulative computation per pixel. We observe qualitatively that the model learns to allocate more computation around boundaries, which aligns well with how humans perform boundary annotation.

4.5.2 Comprehensive Benchmark Comparison

We now compare our models under different degrees of computational parsimony ($\rho = 0.5, 0.7, 0.9, 1.0$) with other state-of-the-art systems for pixel labeling.

Taking boundary detection as the first task, we quantitatively compare our model to COB [232], HED [358], LEP [248], UCM [7], NCuts [299], EGB [80], MCG [8] and the mean shift (MShift) algorithm [56]. Fig. 4.4 (b) shows comparison to all the methods (PR curves in appendix), demonstrating our model achieves state-of-the-art performance. Note that our model has the same backbone architecture of HED [358], but outperforms it with our MultiPool module which increases receptive fields at higher levels. Our model performs on par with COB [232], which uses auxiliary losses for oriented boundary detection. Note that it is possible to surpass human performance with sophisticated techniques [161], but we don't pursue this as it is out the scope of this thesis.

Table 4.3, 4.4 and 4.5 show the comprehensive comparisons on the tasks of semantic segmentation, monocular depth and surface normal estimation, respectively. In addition to comparing with state-of-the-art methods, we also show the result of MultiPool module with softmax weighted average operation, termed by *MP@Res5 (w-Avg.)*. Interestingly, MultiPool performs slightly better when equipped with PAG than the weighted average fusion. We attribute this to the facts that, longer training has been done in the stage-wise training strategy, and PAG unit also constrains the information flow to train specific branches.

Our baseline model achieves performance on par with recent methods for all tasks. When inserting the MultiPool module, we improve even further and surpass the compared methods for most tasks and datasets. In particular, on datasets with large perspective images, *i.e.* Stanford-2D-3D and Cityscapes, the MultiPool module shows greater improvement. Reducing the computation 20-30% only yields a performance drop of 3-5% across most tasks and benchmarks.

For depth and surface normal estimation tasks, our baseline models also perform very well. This is notable since we don't leverage multi-task learning (unlike Eigen [74]) and do not use extra images to augment training set (unlike most methods for depth estimation using

Table 4.3: Semantic segmentation is measured by Intersection over Union (IoU), pixel accuracy (acc), and iIoU that leverages the size of segments w.r.t categories. Results marked by † are from our trained models with the released code.

	NYUv2 [301]		Stanford-2D-3D [9]		Cityscapes [301]	
methods/metrics	IoU	pixel acc.	IoU	pixel acc.	IoU	iIoU
baseline	42.1	71.1	79.5	92.1	73.8	54.7
MP@Res5 (w-Avg.)	46.3	73.4	83.7	93.6	75.8	56.9
MP@Res5 (PAG)	46.5	73.5	83.7	93.7	75.7	55.8
MP@Res5 ($\rho = 0.9$)	45.8	72.9	82.8	93.3	75.0	55.4
MP@Res5 ($\rho = 0.7$)	43.6	71.4	82.4	93.2	72.6	55.1
MP@Res5 ($\rho = 0.5$)	40.9	69.4	81.8	92.9	70.8	53.2
PerspectiveParsing [167]	44.5	72.1	76.5	91.0	75.4	56.8
DeepLab [45]	—	—	69.8†	88.0†	71.4	51.6
LRR [99]	—	—	—	—	70.0	48.0
PSPNet [385]	—	—	67.4†	87.6†	78.7	60.4
RefineNet-Res50 [208]	43.8	—	—	—	—	—
RefineNet-Res152 [208]	46.5	73.6	—	—	—	—

$\sim$100k extra images to augment the training set as shown in Table 4.4). We attribute this to the combination of the proposed PAG MultiPool and carefully designed losses for depth and surface normal estimation.

■ 4.5.3 Qualitative Visualization

We visualize the prediction and attentional maps in Fig. 4.6 for the four datasets, respectively. As we notice that the binary attention maps are similar w.r.t sparse property, we squeeze over macro building blocks (Res) the attention maps for visualization, as well as the overall ponder map. From the figures, we can see our models allocate more computation on the regions/pixels which are likely the sharp change region, *e.g.* boundary between semantic segments, regions between two depth layer, places around normal changes like between wall and ceiling.

Table 4.4: Depth estimation is measured by standard threshold accuracy, *i.e.* the percentage (%) of predicted pixel depths d_i s.t. $\delta = \max(\frac{d_i}{\hat{d}_i}, \frac{\hat{d}_i}{d_i}) < \tau$, where $\tau = \{1.25, 1.25^2, 1.25^3\}$. Methods with * use ∼100k extra images to train.

	NYUv2 [301]			Stanford-2D-3D [9]			Cityscapes [301]		
methods/metric ($\delta < \tau$)	1.25	1.25^2	1.25^3	1.25	1.25^2	1.25^3	1.25	1.25^2	1.25^3
baseline	71.1	93.2	98.5	73.1	92.1	97.5	29.0	53.8	75.8
MP@Res5 (w-Avg.)	74.5	94.4	98.8	77.5	94.1	97.9	33.7	65.9	76.9
MP@Res5 (PAG)	75.1	94.4	98.8	77.6	94.1	97.9	34.6	66.2	77.2
MultiPool ($\rho = 0.9$)	74.5	94.4	98.8	77.3	93.9	97.8	34.5	65.7	76.9
MultiPool ($\rho = 0.7$)	71.0	93.3	98.5	75.4	92.8	97.6	32.0	63.5	75.8
MultiPool ($\rho = 0.5$)	67.3	91.0	97.7	72.7	91.3	97.1	28.7	58.7	71.6
Liu [188]	61.4	88.3	97.1	—	—	—	—	—	—
Ladicky* [184]	54.2	82.9	94.0	—	—	—	—	—	—
Eigen* [75]	61.4	88.8	97.2	—	—	—	—	—	—
Eigen* [74]	76.9	95.0	98.8	—	—	—	—	—	—
Laina* [188]	81.1	95.3	98.8	—	—	—	—	—	—

Table 4.5: Surface normal estimation is measured by mean angular error and the percentage of prediction error within t° degree where $t = \{11.25, 22.50, 30.00\}$. Smaller ang. err. mean better performance as marked by ↓.

	NYUv2 [301]				Stanford-2D-3D [9]			
methods/metrics	ang. err.↓	11.25°	22.50°	30.00°	ang. err.↓	11.25°	22.50°	30.00°
baseline	22.3	34.4	62.5	74.4	19.0	51.5	68.6	76.3
MP@Res5 (w-Avg.)	21.9	35.9	63.8	75.3	16.5	58.2	74.2	80.4
MP@Res5 (PAG)	21.7	36.1	64.2	75.5	16.5	58.3	74.2	80.4
MP@Res5 ($\rho = 0.9$)	21.9	35.9	63.9	75.4	16.7	57.5	73.7	80.1
MP@Res5 ($\rho = 0.7$)	22.5	34.7	62.5	74.1	17.0	56.5	73.1	79.7
MP@Res5 ($\rho = 0.5$)	23.6	31.9	59.7	71.8	17.7	54.7	71.4	78.5
Fouhey [88]	35.3	16.4	36.6	48.2	—	—	—	—
Ladicky [186]	35.5	24.0	45.6	55.9	—	—	—	—
Wang [336]	28.8	35.2	57.1	65.5	—	—	—	—
Eigen [74]	22.2	38.6	64.0	73.9	—	—	—	—

4.6 Conclusion and Future Work

In this chapter, we have studied the problem of parsimonious inference for pixel labeling tasks under limited computation budget with a deep CNN network. To achieve this, we propose a Pixel-wise Attentional Gating unit (PAG) that learns to generate sparse binary masks that control computation at each layer on a per-pixel basis. Our approach differs

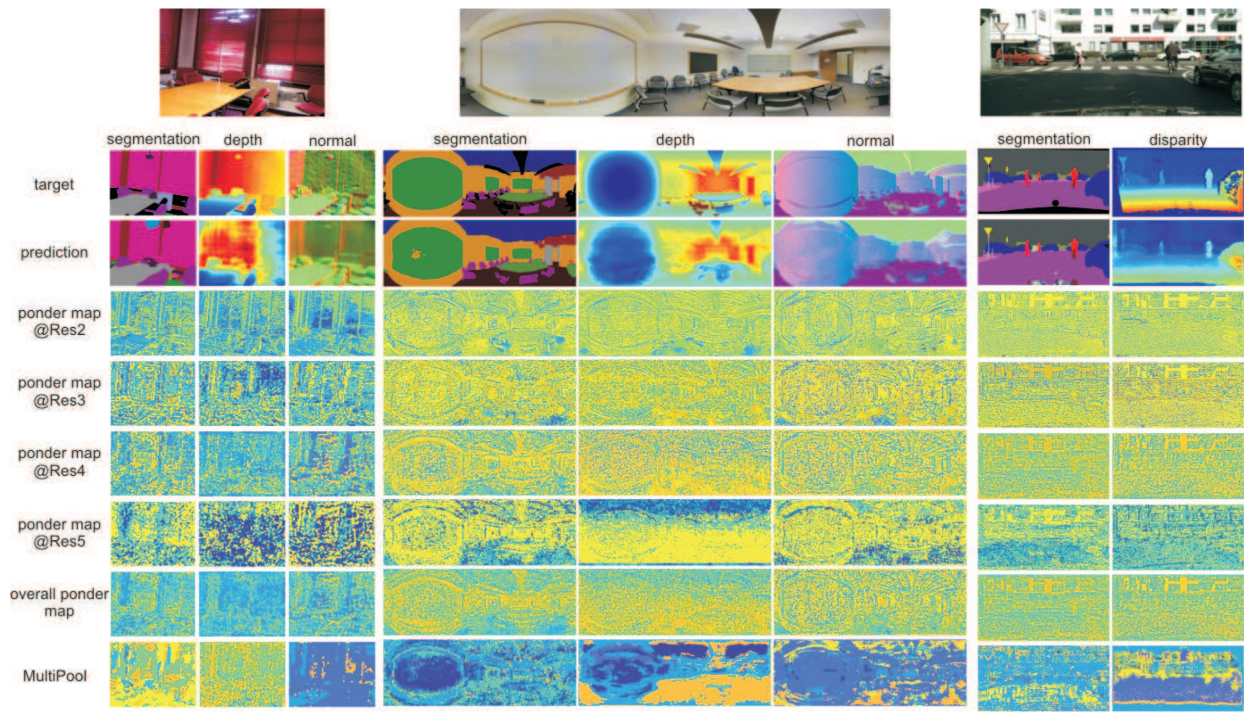


Figure 4.6: Visualization of semantic segmentation, monocular depth and surface normal estimation over the three datasets. Besides the overall ponder map, we also show the partial ponder map for each macro residual block by squeezing the sparse binary attentional maps. Random images are chosen from the three datasets, and results are warped for visualization.

from previous approaches in demonstrating improved performance on pixel labeling tasks using spatially varying computation trained with simple task-specific loss. This makes our approach a good candidate for general use as it is task and architecture agnostic and avoids more complicated reinforcement learning-style approaches, instead relying a simple, easy-to-set sparsity target that correlates closely with empirical computational cost. While our PAG is based on a generic attention mechanism, we anticipate future work might explore further improvements that integrate task-driven constraints to achieve further savings.

Chapter 5

Pixel Embedding for Instance Grouping

We introduce a differentiable, end-to-end trainable framework for solving pixel-level grouping problems such as instance segmentation consisting of two novel components. First, we regress pixels into a hyper-spherical embedding space so that pixels from the same group have high cosine similarity while those from different groups have similarity below a specified margin. We analyze the choice of embedding dimension and margin, relating them to theoretical results on the problem of distributing points uniformly on the sphere. Second, to group instances, we utilize a variant of mean-shift clustering, implemented as a recurrent neural network parameterized by kernel bandwidth. This recurrent grouping module is differentiable, enjoys convergent dynamics and probabilistic interpretability. Backpropagating the group-weighted loss through this module allows learning to focus on only correcting embedding errors that won't be resolved during subsequent clustering. Our framework, while conceptually simple and theoretically abundant, is also practically effective and computationally efficient. We demonstrate substantial improvements over state-of-the-art instance segmentation for object proposal generation, as well as demonstrating the benefits of grouping loss for classification tasks such as boundary detection and semantic segmentation.

5.1 Background

The successes of deep convolutional neural nets (CNNs) at image classification has spawned a flurry of work in computer vision on adapting these models to pixel-level image understanding tasks, such as boundary detection [7, 358, 232], semantic segmentation [221, 45, 167], optical flow [349, 71], and pose estimation [346, 36]. The key ideas that have enabled this adaption thus far are: (1) deconvolution schemes that allow for upsampling coarse pooled feature maps to make detailed predictions at the spatial resolution of individual pixels [358, 99], (2) skip connections and hyper-columns which concatenate representations across multi-resolution feature maps [111, 44], (3) atrous convolution which allows efficient computation with large receptive fields while maintaining spatial resolution [45, 167], and (4) fully convolutional operation which handles variable sized input images.

In contrast, there has been less innovation in the development of specialized loss functions for training. Pixel-level labeling tasks fall into the category of structured output prediction [13], where the model outputs a structured object (e.g., a whole image parse) rather than a scalar or categorical variable. However, most CNN pixel-labeling architectures are simply trained with loss functions that decompose into a simple (weighted) sum of classification or regression losses over individual pixel labels.

The need to address the output space structure is more apparent when considering problems where the set of output labels isn't fixed. Our motivating example is object instance segmentation, where the model generates a collection of segments corresponding to object instances. This problem can't be treated as k-way classification since the number of objects isn't known in advance. Further, the loss should be invariant to permutations of the instance labels within the same semantic category.

As a result, most recent successful approaches to instance segmentation have adopted more

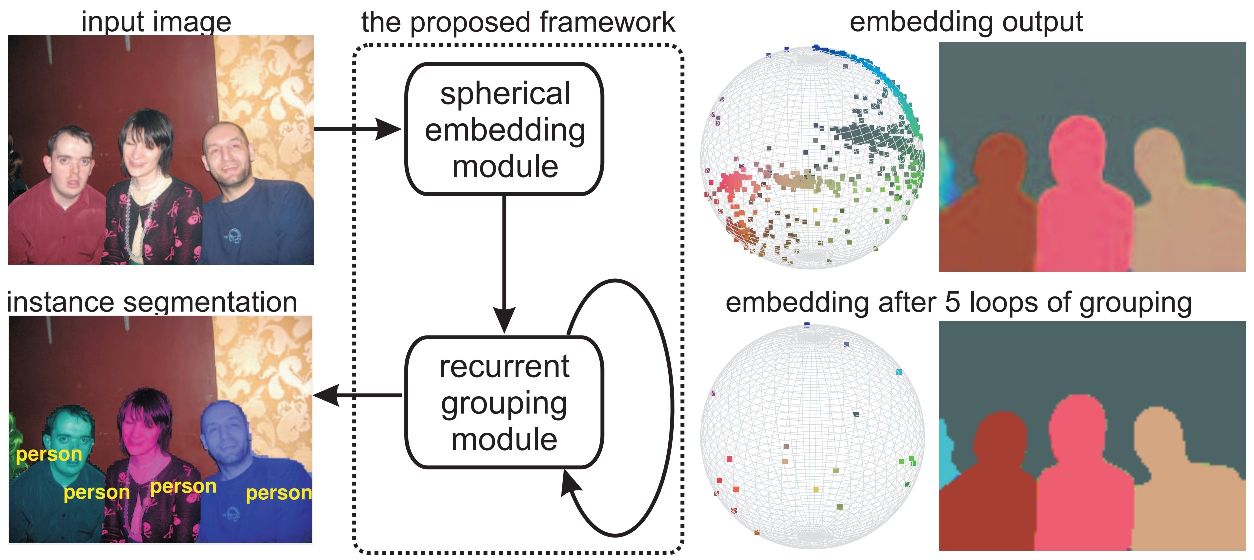


Figure 5.1: Our framework embeds pixels into a hyper-sphere where recurrent mean-shift dynamics groups pixels into a variable number of object instances. Here we visualize random projections of a 64-dim embeddings into 3-dimensions.

heuristic approaches that first use an object detector to enumerate candidate instances and then perform pixel-level segmentation of each instance [206, 59, 204, 205, 10]. Alternately one can generate generic proposal segments and then label each one with a semantic detector [110, 48, 111, 58, 322, 114]. In either case the detection and segmentation steps can both be mapped to standard binary classification losses. While effective, these approaches are somewhat unsatisfying since: (1) they rely on the object detector and non-maximum suppression heuristics to accurately "count" the number of instances, (2) they are difficult to train in an end-to-end manner since the interface between instance segmentation and detection is non-differentiable, and (3) they underperform in cluttered scenes as the assignment of pixels to detections is carried out independently for each detection*.

Here we propose to directly tackle the instance grouping problem in a unified architecture by training a model that labels pixels with unit-length vectors that live in some fixed-dimension embedding space (Fig. 5.1). Unlike k-way classification where the target vectors for each pixel

*This is less a problem for object proposals that are jointly estimated by bottom-up segmentation (e.g., MCG [270] and COB [232]). However, such generic proposal generation is not informed by the top-down semantics.

are specified in advance (i.e., one-hot vectors at the vertices of a k-1 dimensional simplex) we allow each instance to be labeled with an arbitrary embedding vector on the sphere. Our loss function simply enforces the constraint that the embedding vectors used to label different instances are far apart. Since neither the number of labels, nor the target label vectors are specified in advance, we can't use standard soft-max thresholding to produce a discrete labeling. Instead, we utilize a variant of mean-shift clustering which can be viewed as a recurrent network whose fixed point identifies a small, discrete set of instance label vectors and concurrently labels each pixel with one of the vectors from this set.

This framework is largely agnostic to the underlying CNN architecture and can be applied to a range of low, mid and high level visual tasks. Specifically, we carry out experiments showing how this method can be used for boundary detection, object proposal generation and semantic instance segmentation. Even when a task can be modeled by a binary pixel classification loss (e.g., boundary detection) we find that the grouping loss guides the model towards higher-quality feature representations that yield superior performance to classification loss alone. The model really shines for instance segmentation, where we demonstrate a substantial boost in object proposal generation (improving the state-of-the-art average recall for 10 proposals per image from 0.56 to 0.77). To summarize our contributions: (1) we introduce a simple, easily interpreted end-to-end model for pixel-level instance labeling which is widely applicable and highly effective, (2) we provide theoretical analysis that offers guidelines on setting hyperparameters, and (3) benchmark results show substantial improvements over existing approaches.

5.2 Related Work

Common approaches to instance segmentation first generate region proposals or class-agnostic bounding boxes, segment the foreground objects within each proposal and classify the objects in the bounding box [366, 185, 110, 48, 59, 205, 114]. [204] introduce a fully convolutional approach that includes bounding box proposal generation in end-to-end training. Recently, "box-free" methods [266, 267, 206, 127] avoid some limitations of box proposals (e.g. for wiry or articulated objects). They commonly use Faster RCNN [279] to produce "centeredness" score on each pixel and then predict binary instance masks and class labels. Other approaches have been explored for modeling joint segmentation and instance labeling jointly in a combinatorial framework (e.g., [157]) but typically don't address end-to-end learning. Alternately, recurrent models that sequentially produce a list of instances [284, 278] offer another approach to address variable sized output structures in a unified manner.

The most closely related to ours is the associative embedding work of [249], which demonstrated strong results for grouping multi-person keypoints, and unpublished work from [79] on metric learning for instance segmentation. Our approach extends on these ideas substantially by integrating recurrent mean-shift to directly generate the final instances (rather than heuristic decoding or thresholding distance to seed proposals). There is also an important and interesting connection to work that has used embedding to separate instances where the embedding is directly learned using a supervised regression loss rather than a pairwise associative loss. [305] train a regressor that predicts the distance to the contour centerline for boundary detection, while [12] predict the distance transform of the instance masks which is then post-processed with watershed transform to generate segments. [322] predict an embedding based on scene depth and direction towards the instance center (like Hough voting).

Finally, we note that these ideas are related to work on using embedding for solving pairwise clustering problems. For example, normalized cuts clusters embedding vectors given by the eigenvectors of the normalized graph Laplacian [299] and the spatial gradient of these embedding vectors was used in [7] as a feature for boundary detection. Rather than learning pairwise similarity from data and then embedding prior to clustering (e.g., [229]), we use a pairwise loss but learn the embedding directly. Our recurrent mean-shift grouping is reminiscent of other efforts that use unrolled implementations of iterative algorithms such as CRF inference [386] or bilateral filtering [140, 92]. Unlike general RNNs [22, 259] which are often difficult to train, our recurrent model has fixed parameters that assure interpretable convergent dynamics and meaningful gradients during learning.

5.3 Pairwise Loss for Pixel Embeddings

In this section we introduce and analyze the loss we use for learning pixel embeddings. This problem is broadly related to supervised distance metric learning [348, 171, 173] and clustering [172] but adapted to the specifics of instance labeling where the embedding vectors are treated as labels for a variable number of objects in each image.

Our goal is to learn a mapping from an input image to a set of D-dimensional embedding vectors (one for each pixel). Let $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^D$ be the embeddings of pixels i and j respectively with corresponding labels y_i and y_j that denote ground-truth instance-level semantic labels (e.g., *car.1* and *car.2*). We will measure the similarity of the embedding vectors using the cosine similarity, been scaled and offset to lie in the interval $[0, 1]$ for notational convenience:

$$s_{ij} = \frac{1}{2}\left(1 + \frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\|_2 \|\mathbf{x}_j\|_2}\right) \tag{5.1}$$

In the discussion that follows we think of the similarity in terms of the inner product between the projected embedding vectors (e.g., $\frac{x_i}{\|x_i\|}$) which live on the surface of a $(D-1)$ dimensional sphere. Other common similarity metrics utilize Euclidean distance with a squared exponential kernel or sigmoid function [249, 79]. We prefer the cosine metric since it is invariant to the scale of the embedding vectors, decoupling the loss from model design choices such as weight decay or regularization that limit the dynamic range of Euclidean distances.

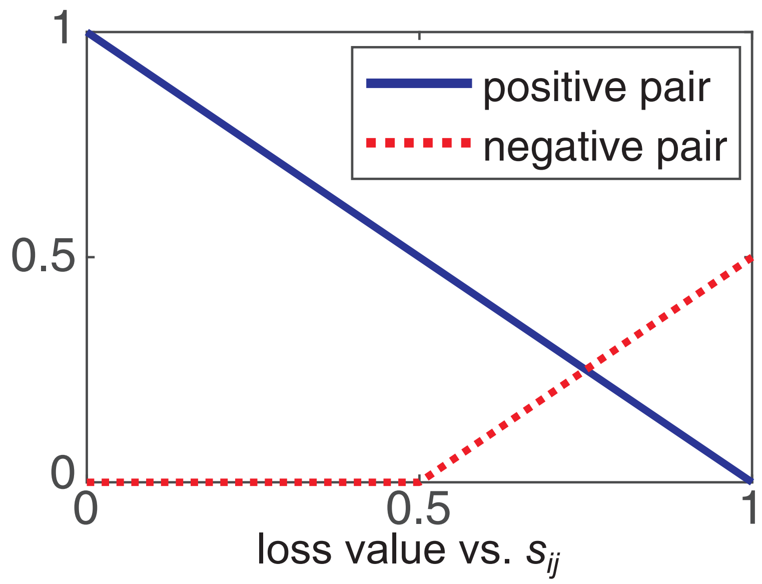


Figure 5.2: Loss as a function of calibrated similarity score Eq. 5.1 with $\alpha = 0.5$. The gradient is constant, limiting the effect of noisy ground-truth labels (i.e., near an object boundary)

Our goal is to learn an embedding so that pixels with the same label (positive pairs with $y_i = y_j$) have the same embedding (i.e. $s_{ij} = 1$). To avoid a trivial solution where all the embedding vectors are the same, we impose the additional constraint that pairs from different instances (negative pairs with $y_i \neq y_j$) are placed far apart. To provide additional flexibility, we include a weight w_i in the definition of the loss which specifies the importance of a given pixel. The total loss over all pairs and training images is:

$$\ell = \sum_{k=1}^{M} \sum_{i,j=1}^{N_k} \frac{w_i^k w_j^k}{N_k} \Big(\mathbf{1}_{\{y_i = y_j\}} (1 - s_{ij}) + \mathbf{1}_{\{y_i \neq y_j\}} [s_{ij} - \alpha]_+ \Big) \tag{5.2}$$

where N_k is the number of pixels in the k-th image (M images in total), and w_i^k is the pixel pair weight associated with pixel i in image k. The hyper-parameter α controls the maximum margin for negative pairs of pixels, incurring a penalty if the embeddings for pixels belonging to the same group have an angular separation of less than $\cos^{-1}(\alpha)$. Positive pairs pay a penalty if they have a similarity less than 1. Fig. 5.2 shows a graph of the loss function. [351] argue that the constant slope of the margin loss is more robust, e.g., than squared loss.

We carry out a simple theoretical analysis which provides a guide for setting the weights w_i and margin hyperparameter α in the loss function. Proofs can be found in the appendix.

5.3.1 Instance-aware Pixel Weighting

We first examine the role of embedding dimension and instance size on the training loss.

Proposition 5.1. *For n vectors $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, the total intra-pixel similarity is bounded as $\sum_{i \neq j} \mathbf{x}_i^T \mathbf{x}_j \geq -\sum_{i=1}^n \|\mathbf{x}_i\|_2^2$. In particular, for n vectors on the hypersphere where $\|\mathbf{x}_i\|_2 = 1$, we have $\sum_{i \neq j} \mathbf{x}_i^T \mathbf{x}_j \geq -n$.*

This proposition indicates that the total cosine similarity (and hence the loss) for a set of embedding vectors has a constant lower bound that does not depend on the dimension of the embedding space (a feature lacking in Euclidean embeddings). In particular, this type of analysis suggests a natural choice of pixel weighting w_i. Suppose a training example contains Q instances and $\mathcal{I}_q$ denotes the set of pixels belonging to a particular ground-truth instance q. We can write

$$\begin{aligned} \| \sum_{q=1}^{Q} \sum_{i \in \mathcal{I}_q} w_i \mathbf{x}_i \|^2 = & \sum_{q=1}^{Q} \| \sum_{i \in \mathcal{I}_q} w_i \mathbf{x}_i \|^2 + \\ & \sum_{p \neq q} \big(\sum_{i \in \mathcal{I}_p} w_i \mathbf{x}_i \big)^T \big(\sum_{j \in \mathcal{I}_q} w_j \mathbf{x}_j \big) \end{aligned}$$

where the first term on the r.h.s. corresponds to contributions to the loss function for positive pairs while the second corresponds to contributions from negative pairs. Setting $w_i = \frac{1}{|\mathcal{I}_q|}$ for pixels i belonging to ground-truth instance q assures that each instance contributes equally to the loss independent of size. Furthermore, when the embedding dimension $D \geq Q$, we can simply embed the data so that the instance means $\mu_k = \frac{1}{|\mathcal{I}_q|} \sum_{i \in \mathcal{I}_q} \mathbf{x}_i$ are along orthogonal axes on the sphere. This zeros out the second term on the r.h.s., leaving only the first term which is bounded $0 \leq \sum_{q=1}^{Q} \left\| \frac{1}{|\mathcal{I}_q|} \sum_{i \in \mathcal{I}_q} \mathbf{x}_i \right\|^2 \leq Q$, and translates to corresponding upper and lower bounds on the loss that are independent of the number of pixels and embedding dimension (so long as $D \geq Q$).

Pairwise weighting schemes have been shown important empirically [79] and class imbalance can have a substantial effect on the performance of different architectures (see e.g., [212]). While other work has advocated online bootstrapping methods for hard-pixel mining or mini-batch selection [222, 170, 300, 355], our approach is much simpler. Guided by this result we simply use uniform random sampling of pixels during training, appropriately weighted by instance size in order to estimate the loss.

5.3.2 Margin Selection

To analyze the appropriate margin, let's first consider the problem of distributing labels for different instances as far apart as possible on a 3D sphere, sometimes referred to as Tammes's problem, or the hard-spheres problem [288]. This can be formalized as maximizing the smallest distance among n points on a sphere: $\max_{\mathbf{x}_i \in \mathbb{R}^3} \min_{i \neq j} \|\mathbf{x}_i - \mathbf{x}_j\|_2$. Asymptotic results in [106] provide the following proposition (see proof in the appendix):

Proposition 5.2. *Given N vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ on a 2-sphere, i.e. $\mathbf{x}_i \in \mathbb{R}^3$, $\|\mathbf{x}_i\|_2 = 1, \forall i = 1 \ldots n$, choosing $\alpha \leq 1 - \left(\frac{2\pi}{\sqrt{3}N}\right)$, guarantees that $[s_{ij} - \alpha]_+ \geq 0$ for some pair $i \neq j$. Choosing $\alpha > 1 - \frac{1}{4}\left(\left(\frac{8\pi}{\sqrt{3}N}\right)^{\frac{1}{2}} - CN^{-\frac{2}{3}}\right)^2$, guarantees the existence of an embedding with $[s_{ij} - \alpha]_+ = 0$ for all pairs $i \neq j$.*

Proposition 5.2 gives the maximum margin for a separation of n groups of pixels in a three dimensional embedding space (sphere). For example, if an image has at most $\{4, 5, 6, 7\}$ instances, α can be set as small as $\{0.093, 0.274, 0.395, 0.482\}$, respectively.

For points in a higher dimension embedding space, it is a non-trivial problem to establish a tight analytic bound for the margin α. Despite its simple description, distributing n points on a $(D-1)$-dimensional hypersphere is considered a serious mathematical challenge for which there is no general solutions [288, 223]. We adopt a safe (trivial) strategy. For n instances embedded in $n/2$ dimensions one can use value of $\alpha = 0.5$ which allows for zero loss by placing a pair of groups antipodally along each of the $n/2$ orthogonal axes. We adopt

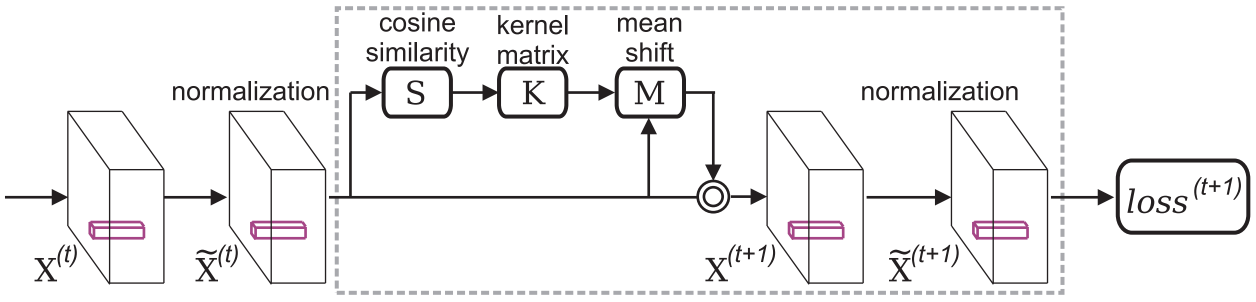


Figure 5.3: Recurrent mean shift grouping module is unrolled during training.

this setting for the majority of experiments in the paper where the embedding dimension is set to 64.

5.4 Recurrent Mean-Shift Grouping

While we can directly train a model to predict embeddings as described in the previous section, it is not clear how to generate the final instance segmentation from the resulting (imperfect) embeddings. One can utilize heuristic post-processing [62] or utilize clustering algorithms that estimate the number of instances [206], but these are not differentiable and thus unsatisfying. Instead, we introduce a mean-shift grouping model (Fig. 5.3) which operates recurrently on the embedding space in order to congeal the embedding vectors into a small number of instance labels.

Mean-shift and closely related algorithms [91, 49, 55, 56] use kernel density estimation to approximate the probability density from a set of samples and then perform clustering on the input data by assigning or moving each sample to the nearest mode (local maxima). From our perspective, the advantages of this approach are (1) the final instance labels (modes) live in the same embedding space as the initial data, (2) the recurrent dynamics of the clustering process depend smoothing on the input allowing for easy backpropagation, (3) the behavior depends on a single parameter, the kernel bandwidth, which is easily interpretable and can be related to the margin used for the embedding loss.

5.4.1 Mean Shift Clustering

A common choice for non-parametric density estimation is to use the isotropic multivariate normal kernel $K(\mathbf{x}, \mathbf{x}_i) = (2\pi)^{-D/2} \exp\Big(-\frac{\delta^2}{2}\|\mathbf{x} - \mathbf{x}_i\|_2^2\Big)$ and approximate the data density non-parametrically as $p(x) = \frac{1}{N}\sum K(x, x_i)$. Since our embedding vectors are unit norm, we instead use the von Mises-Fisher distribution which is the natural extension of the multivariate normal to the hypersphere [85, 16, 233, 159], and is given by $K(\mathbf{x}, \mathbf{x}_i) \propto \exp(\delta \mathbf{x}^T \mathbf{x}_i)$. The kernel bandwidth, δ determines the smoothness of the kernel density estimate and is closely related to the margin used for learning the embedding space. While it is straightforward to learn δ during training, we instead set it to satisfy $\frac{1}{\delta} = \frac{1-\alpha}{3}$ throughout our experiments, such that the cluster separation (margin) in the learned embedding space is three standard deviations.

We formulate the mean shift algorithm in a matrix form. Let $\mathbf{X} \in \mathbb{R}^{D\times N}$ denote the stacked N pixel embedding vectors of an image. The kernel matrix is given by $\mathbf{K} = \exp(\delta \mathbf{X}^T\mathbf{X}) \in \mathbb{R}^{N\times N}$. Let $\mathbf{D} = \mathsf{diag}(\mathbf{K}^T\mathbf{1})$ denote the diagonal matrix of total affinities, referred to as the degree when $\mathbf{K}$ is viewed as a weighted graph adjacency matrix. At each iteration, we compute the mean shift $\mathbf{M} = \mathbf{XKD}^{-1} - \mathbf{X}$, which is the difference vector between $\mathbf{X}$ and the kernel weighted average of $\mathbf{X}$. We then modify the embedding vectors by moving them in the mean shift direction with step size η:

$$\begin{aligned} \mathbf{X} \leftarrow & \mathbf{X} + \eta(\mathbf{XKD}^{-1} - \mathbf{X}) \\ \leftarrow & \mathbf{X}(\eta\mathbf{KD}^{-1} + (1-\eta)\mathbf{I}) \end{aligned} \tag{5.3}$$

Note that unlike standard mean-shift mode finding, we recompute $\mathbf{K}$ at each iteration. These update dynamics are termed the explicit-η method and were analyzed by [39]. When $\eta = 1$ and the kernel is Gaussian, this is also referred to as Gaussian Blurring Mean Shift (GBMS) and has been shown to have cubic convergence [39] under appropriate conditions. Unlike deep

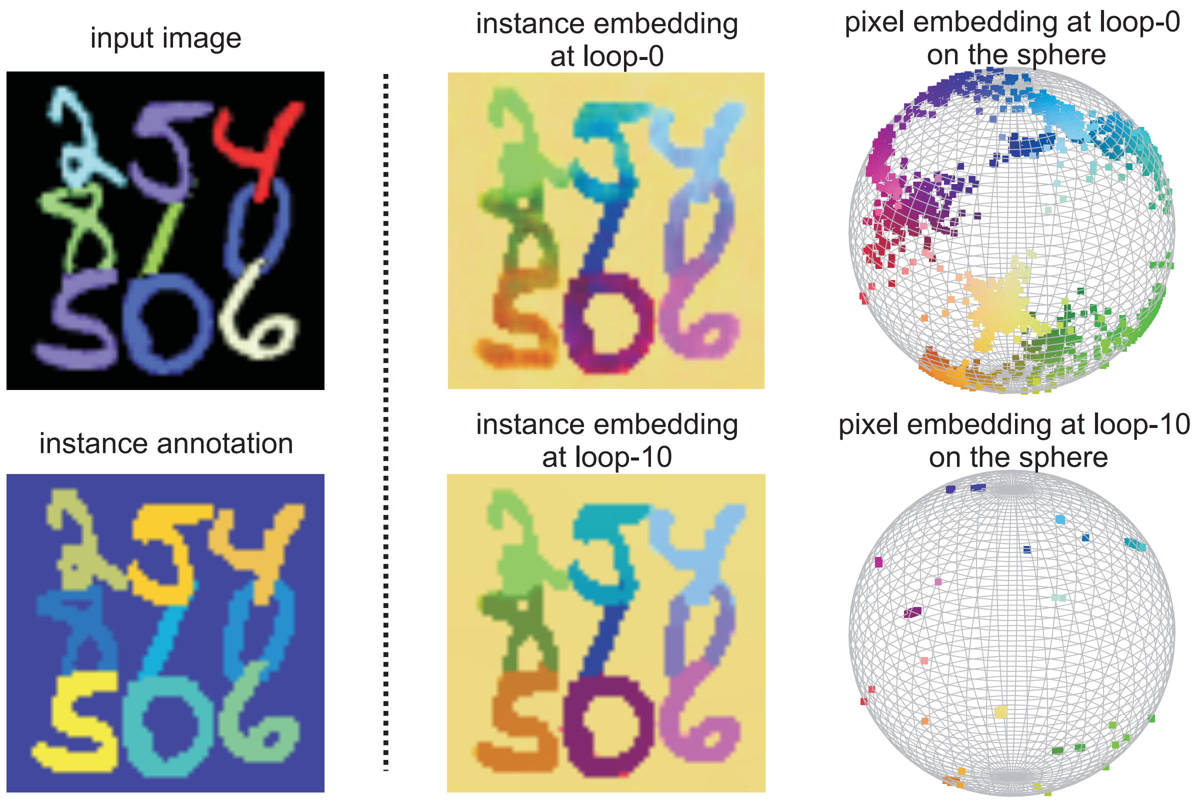


Figure 5.4: Demonstration of mean-shift grouping on a synthetic image and with ground-truth instance identities (left panel). Right panel: the pixel embedding visualization at 3-dimensional embedding sphere (upper row) and after 10 iterations of recurrent mean-shift grouping (bottom row).

RNNs, the parameters of our recurrent module are not learned and the forward dynamics are convergent under general conditions. In practice, we do not observe issues with exploding or vanishing gradients during back-propagation through a finite number of iterations [†].

Fig. 5.4 demonstrates a toy example of applying the method to perform digit instance segmentation on synthetic images from MNIST [193]. We learn 3-dimensional embedding in order to visualize the results before and after the mean shift grouping module. From the figure, we can see the mean shift grouping transforms the initial embedding vectors to yield a small set of instance labels which are distinct (for negative pairs) and compact (for positive pairs).

[†]Some intuition about stability may be gained by noting that the eigenvalues of $\mathbf{KD}^{-1}$ lie in the interval $[0,1]$, but we have not been able to prove useful corresponding bounds on the spectrum of the Jacobian.

5.4.2 End-to-end training

It's straightforward to compute the derivatives of the recurrent mean shift grouping module w.r.t $\mathbf{X}$ based on the the chain rule so our whole system is end-to-end trainable through back-propagation. Details about the derivative computation can be found in the appendix. To understand the benefit of end-to-end training, we visualize the embedding gradient with and without the grouping module (Fig. 5.5). Interestingly, we observe that the gradient backpropagated through mean shift focuses on fixing the embedding in uncertain regions, e.g. instance boundaries, while suggesting small magnitude updates for those errors which will be easily fixed by the mean-shift iteration.

While we could simply apply the pairwise embedding loss to the final output of the mean-shift grouping, in practice we accumulate the loss over all iterations (including the initial embedding regression). We unroll the recurrent grouping module into T loops, and accumulate the same loss function at the unrolled loop-t:

$$\begin{aligned}\ell^t =& \sum_{k=1}^{M} \sum_{i,j \in S_k} \frac{w_i^k w_j^k}{|S_k|} \Big(\mathbf{1}_{\{y_i = y_j\}} (1 - s_{ij}^t) + \mathbf{1}_{\{y_i \neq y_j\}} [s_{ij}^t - \alpha]_+ \Big) \\ \ell =& \sum_{t=1}^{T} \ell^t\end{aligned}$$

5.5 Experiments

We now describe experiments in training our framework to deal a variety of pixel-labeling problems, including boundary detection, object proposal detection, semantic segmentation and instance-level semantic segmentation.

▢ 5.5.1 Tasks, Datasets and Implementation

We illustrate the advantages of the proposed modules on several large-scale datasets. First, to illustrate the ability of the instance-aware weighting and uniform sampling mechanism to handle imbalanced data and low embedding dimension, we use the BSDS500 [7] dataset to train a boundary detector for boundary detection ($> 90\%$ pixels are non-boundary pixels). We train with the standard split [7, 358], using 300 train-val images to train our model based on ResNet50 [115] and evaluate on the remaining 200 test images. Second, to explore instance segmentation and object proposal generation, we use PASCAL VOC 2012 dataset [77] with additional instance mask annotations provided by [109]. This provides 10,582 and 1,449 images for training and evaluation, respectively.

We implement our approach using the toolbox MatConvNet [326], and train using SGD on a single Titan X GPU. [‡]. To compute calibrated cosine similarity, we utilize an L2-normalization layer before matrix multiplication [163], which also contains random sampling with a hyper-parameter to control the ratio of pixels to be sampled for an image. In practice, we observe that performance does not depend strongly on this ratio and hence set it based on available (GPU) memory.

While our modules are architecture agnostic, we use the ResNet50 and ResNet101 models [115] pre-trained over ImageNet [63] as the backbone. Similar to [45], we increase the output resolution of ResNet by removing the top global 7×7 pooling layer and the last two 2×2 pooling layers, replacing them with atrous convolution with dilation rate 2 and 4, respectively to maintain a spatial sampling rate. Our model thus outputs predictions at 1/8 the input resolution which are upsampled for benchmarking.

We augment the training set using random scaling by $s \in [0.5, 1.5]$, in-plane rotation by

[‡]The code and trained models can be found at *https://github.com/aimerykong/Recurrent-Pixel-Embedding-for-Instance-Grouping*

$[-10°, 10°]$ degrees, random left-right flips, random crops with 20-pixel margin and of size divisible by 8, and color jittering. When training the model, we fix the batch normalization in ResNet backbone, using the same constant global moments in both training and testing. Throughout training, we set batch size to one where the batch is a single input image. We use the "poly" learning rate policy [45] with a base learning rate of $2.5e-4$ scaled as a function of iteration by $(1-\frac{iter}{maxiter})^{0.9}$.

▢ 5.5.2 Boundary Detection

For boundary detection, we first train a model to group the pixels into boundary or non-boundary groups. Similar to COB [232] and HED [358], we include multiple branches over ResBlock $2, 3, 4, 5$ for training. Since the number of instances labels is 2, we learn a simple 3-dimensional embedding space which has the advantage of easy visualization as an RGB image. Fig. 5.7 shows the resulting embeddings in the first row of each panel. Note that even though we didn't utilize mean-shift grouping, the trained embedding already produces compact clusters. To compare quantitatively to the state-of-the-art, we learn a fusion layer that combines predictions from multiple levels of the feature hierarchy fine-tuned with a logistic loss to match the binary output. Fig. 5.7 shows the results in the second row. Interestingly, we can see that the fine-tuned model embeddings encode not only boundary presence/absence but also the orientation and signed distance to nearby boundaries.

Quantitatively, we compare our model to COB [232], HED [358], CEDN [362], LEP [248], UCM [7], ISCRA [280], NCuts [299], EGB [80], and the original mean shift (MShift) segmentation algorithm [56]. Fig. 5.6 shows standard benchmark precision-recall for all the methods, demonstrating our model achieves state-of-the-art performance. Note that our model has the same architecture of COB [232] except with a different loss functions and no explicit branches to compute boundary orientation. Our embedding loss by naturally pushes boundary pixel embeddings to be similar which is also the desirable property for detecting boundaries using logistic loss. Note that it is possible to surpass human performance with several sophisticated techniques [161], we don't pursue this as it is out the scope of this paper.

■ 5.5.3 Object Proposal Detection

Object proposals are an integral part of current object detection and semantic segmentation pipelines [279, 114], as they provide a reduced search space of locations, scales and shapes for subsequent recognition. State-of-the-art methods usually involve training models that output large numbers of proposals, particularly those based on bounding boxes. Here we demonstrate that by training our framework with 64-dimensional embedding space on the object instance level annotations, we are able to produce very high quality object proposals by grouping the pixels into instances. It is worth noting that due to the nature of our grouping module, far fewer number of proposals are produced with much higher quality. We compare against the most recent techniques including POISE [133], LPO [177], CPMC [38], GOP [176], SeSe [323], GLS [275], RIGOR [132].

Fig. 5.8 shows the Average Recall (AR) [125] with respect to the number of object proposals[§]. Our model performs remarkably well compared to other methods, achieving high average recall of ground-truth objects with two orders of magnitude fewer proposals. We also plot the curves for SharpMask [266] and DeepMask [267] using the proposals released by the authors. Despite only training on PASCAL, we outperform these models which were trained on the much larger COCO dataset [213]. In Table 5.1 we report the total average recall at IoU$= 0.5$ for some recently proposed proposal detection methods, including unpublished work inst-DML [79] which is similar in spirit to our model but learns a Euclidean distance based metric to group pixels. We can clearly see that our method achieves significantly

[§]Our basic model produces ~ 10 proposals per image. In order to plot a curve for our model for larger numbers of proposals, we run the mean shift grouping with multiple smaller bandwidth parameters, pool the results, and remove redundant proposals.

#prop.	SCG [270]	MCG [270]	COB [232]	inst-DML [79]	Ours
10	-	-	-	0.558	0.769
60	0.624	0.652	0.738	0.667	0.814

Table 5.1: Object proposal detection on PASCAL VOC 2012 validation set measured by total Average Recall (AR) at IoU=0.50 and various number of proposals per image.

better results than existing methods.

5.5.4 Semantic Instance Detection

As a final test of our method, we also train it to produce semantic labels which are combined with our instance proposal method to recognize the detected proposals.

For semantic segmentation which is a k-way classification problem, we train a model using cross-entropy loss alongside our embedding loss. Similar to our proposal detection model, we use a 64-dimension embedding space on top of DeepLab-v3 [46] as our base model. While there are more complex methods in literature such as PSPNet [385] and which augment training with additional data (e.g., COCO [213] or JFT-300M dataset [309]) and utilize ensembles and post-processing, we focus on a simple experiment training the base model with/without the proposed pixel pair embedding loss to demonstrate the effectiveness.

In addition to reporting mean intersection over union (mIoU) over all classes, we also computed mIoU restricted to a narrow band of pixels around the ground-truth boundaries. This partition into figure/boundary/background is sometimes referred to as a tri-map in the matting literature and has been previously utilized in analyzing semantic segmentation performance [160, 45, 99]. Fig. 5.9 shows the mIoU as a function of the width of the tri-map boundary zone. This demonstrates that with embedding loss yields performance gains over cross-entropy primarily far from ground-truth boundaries where it successfully fills in holes in the segments output (see also qualitative results in Fig. 5.10). This is in spirit similar to the model in [112], which considers local consistency to improve spatial precision. However,

Method	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	mean
SDS [110]	58.8	0.5	60.1	34.4	29.5	60.6	40.0	73.6	6.5	52.4	31.7	62.0	49.1	45.6	47.9	22.6	43.5	26.9	66.2	66.1	43.8
Chen et al. [48]	63.6	0.3	61.5	43.9	33.8	67.3	46.9	74.4	8.6	52.3	31.3	63.5	48.8	47.9	48.3	26.3	40.1	33.5	66.7	67.8	46.3
PFN [206]	76.4	15.6	74.2	54.1	26.3	73.8	31.4	92.1	17.4	73.7	48.1	82.2	81.7	72.0	48.4	23.7	57.7	64.4	88.9	72.3	58.7
MNC [59]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	63.5
Li et al. [204]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	65.7
R2-IOS [205]	87.0	6.1	90.3	67.9	48.4	86.2	68.3	90.3	24.5	84.2	29.6	91.0	71.2	79.9	60.4	42.4	67.4	61.7	94.3	82.1	66.7
Assoc. Embed. [249]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	35.1
inst-DML [79]	69.7	1.2	78.2	53.8	42.2	80.1	57.4	88.8	16.0	73.2	57.9	88.4	78.9	80.0	68.0	28.0	61.5	61.3	87.5	70.4	62.1
Ours	85.9	10.0	74.3	54.6	43.7	81.3	64.1	86.1	17.5	77.5	57.0	89.2	77.8	83.7	67.9	31.2	62.5	63.3	88.6	74.2	64.5

Table 5.2: Instance-level segmentation comparison using APr metric at 0.5 IoU on the PASCAL VOC 2012 validation set.

our uniform sampling allows for long-range interactions between pixels.

To label detected instances with semantic labels, we use the semantic segmentation model described above to generate labels and then use a simple voting strategy to transfer these predictions to the instance proposals. In order to produce a final confidence score associated with each proposed object, we train a linear regressor to score each object instance based on its morphology (e.g., size, connectedness) and the consistency w.r.t. the semantic segmentation prediction. We note this is substantially simpler than approaches based, e.g. on Faster-RCNN [279] which use much richer convolutional features to rescore segmented instances [114].

Comparison of instance detection performance are displayed in Table 5.2. We use a standard IoU threshold of 0.5 to identify true positives, unless an ground-truth instance has already been detected by a higher scoring proposal in which case it is a false positive. We report the average precision per-class as well as the average all classes (as in [109]). Our approach yields competitive performance on VOC validation despite our simple re-scoring. Among the competing methods, the one closest to our model is inst-DML [79], that learns Euclidean distance based metric with logistic loss. The inst-DML approach relies on generating pixel seeds to derive instance masks. The pixel seeds may fail to correctly detect thin structures which perhaps explains why this method performs 10x worse than our method on the bike category. In contrast, our mean-shift grouping approach doesn't make strong assumptions about the object shape or topology.

For visualization purposes, we generate three random matrices projections of the 64-dimensional embedding and display them in the spatial domain as RGB images. Fig. 5.11 shows the embedding visualization, as well as predicted semantic segmentation and instance-level segmentation. From the visualization, we can see the instance-level semantic segmentation outputs complete object instances even though semantic segmentation results are noisy, such as the bike in the first image in Fig. 5.11. The instance embedding provides important details that resolve both inter- and intra-class instance overlap which are not emphasized in the semantic segmentation loss.

5.6 Conclusion and Future Work

We have presented an end-to-end trainable framework for solving pixel-labeling vision problems based on two novel contributions: a pixel-pairwise loss based on spherical max-margin embedding and a variant of mean shift grouping embedded in a recurrent architecture. These two components mesh closely to provide a framework for robustly recognizing variable numbers of instances without requiring heuristic post-processing or hyperparameter tuning to account for widely varying instance size or class-imbalance. The approach is simple and amenable to theoretical analysis, and when coupled with standard architectures yields instance proposal generation which substantially outperforms state-of-the-art. Our experiments demonstrate the potential for instance embedding and open many opportunities for future work including learn-able variants of mean-shift grouping, extension to other pixel-level domains such as encoding surface shape, depth and figure-ground and multi-task embeddings.

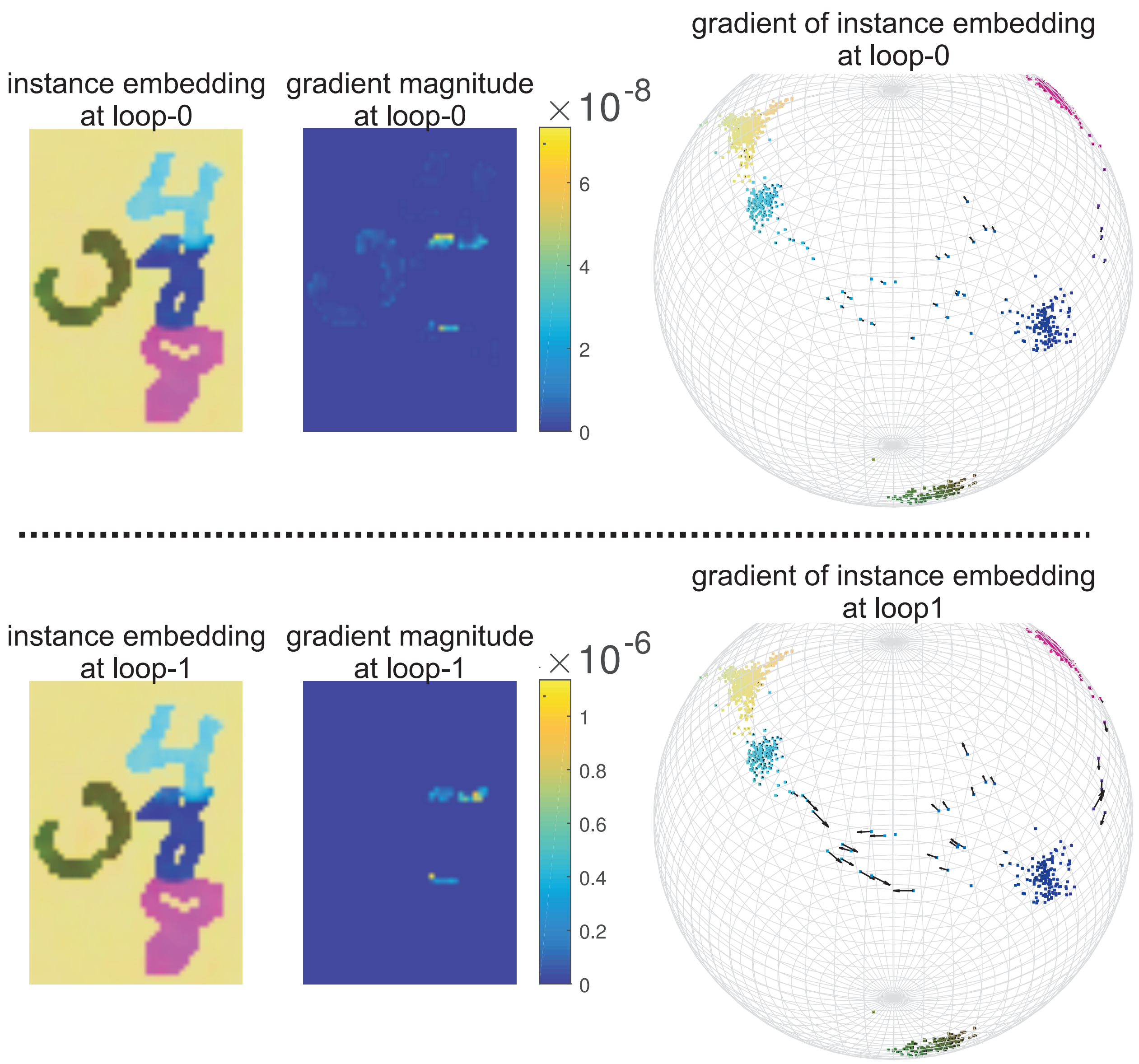


Figure 5.5: To analyze the recurrent mean shift grouping module, we compare the embedding vector gradients with and without one loop of grouping. The length of arrows in the projection demonstrates the gradient magnitude, which are also depicted in maps as the second column. Backpropagating the loss through the grouping module serves to focus updates on embeddings of ambiguous pixels near boundaries while ignoring pixels with small errors which will be corrected by the subsequent grouping process.

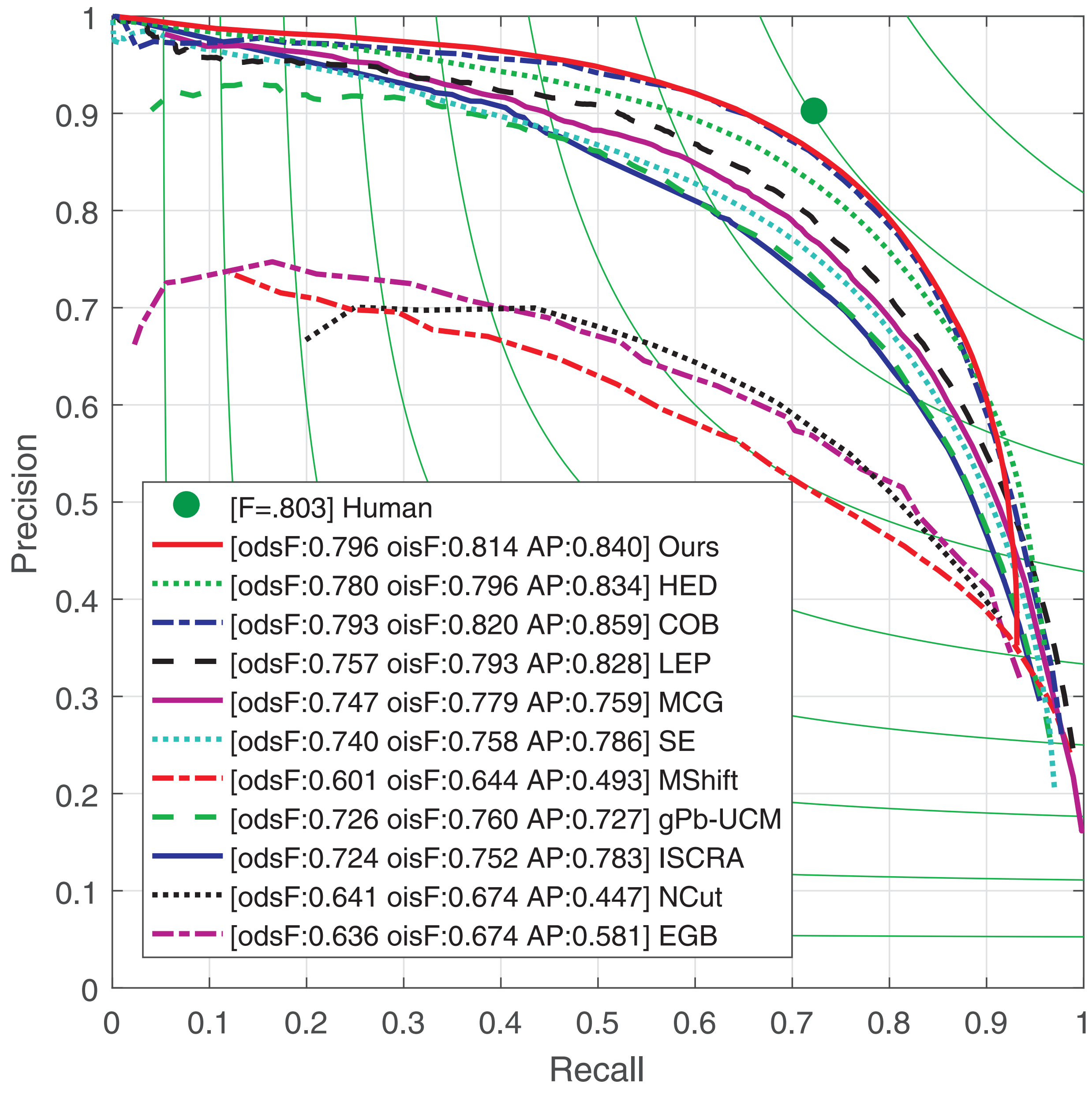


Figure 5.6: Boundary detection performance on BSDS500

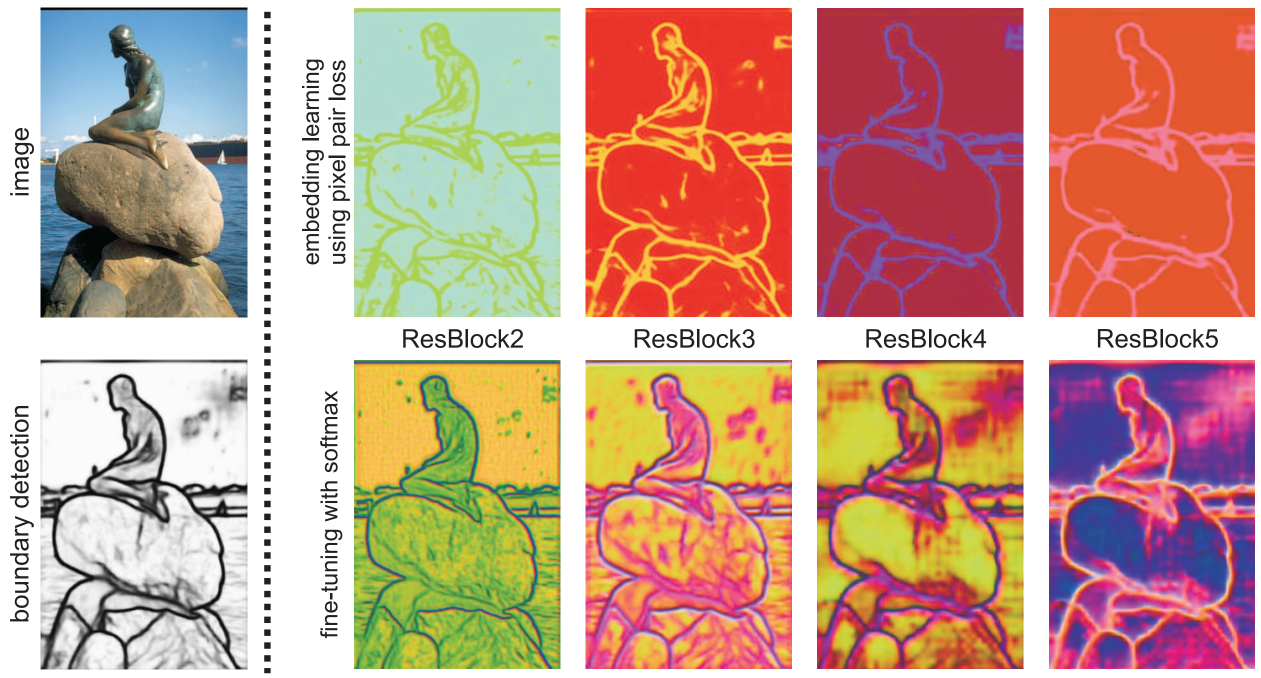


Figure 5.7: Visualization of boundary detection embeddings. We show the 3D embedding as RGB images (more examples in appendix). The upper and lower row in the right panel show embedding vectors at different layers from the model before and after fine-tuning using logistic loss. After fine-tuning not only predict the boundary pixels, but also encode boundary orientation and signed distance to the boundary, similar to supervised embedding approaches [305, 322, 12]

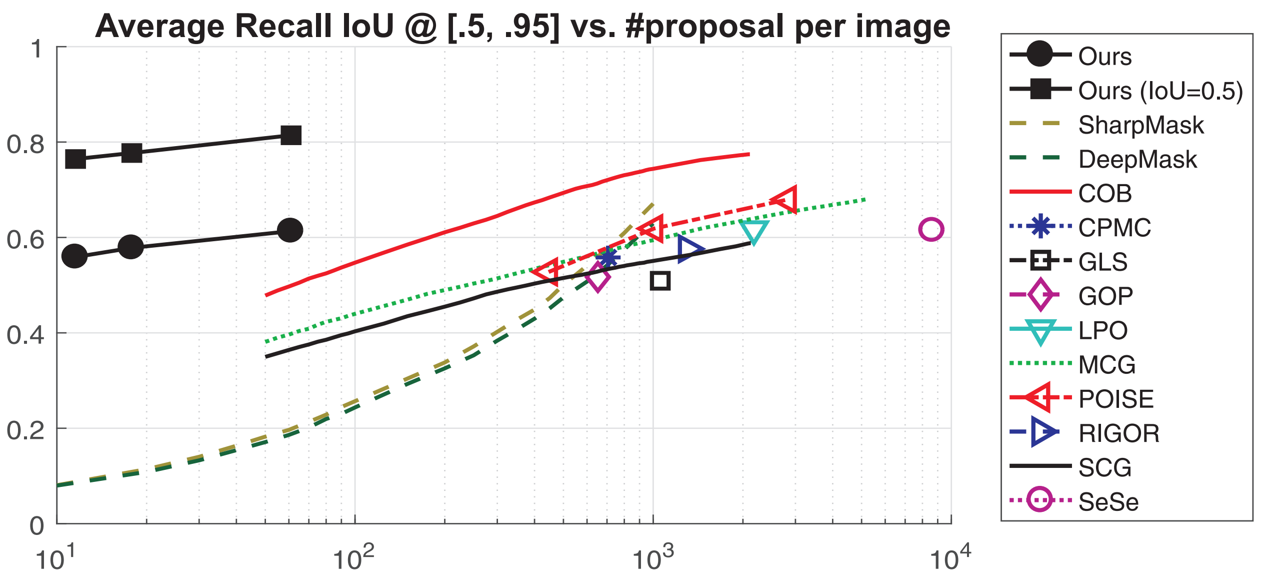


Figure 5.8: Segmented object proposals evaluation on PASCAL VOC 2012 validation set measured by Average Recall (AR) at IoU from 0.5 to 0.95 and step size as 0.5. We also include the curve for our method at IoU=0.5.

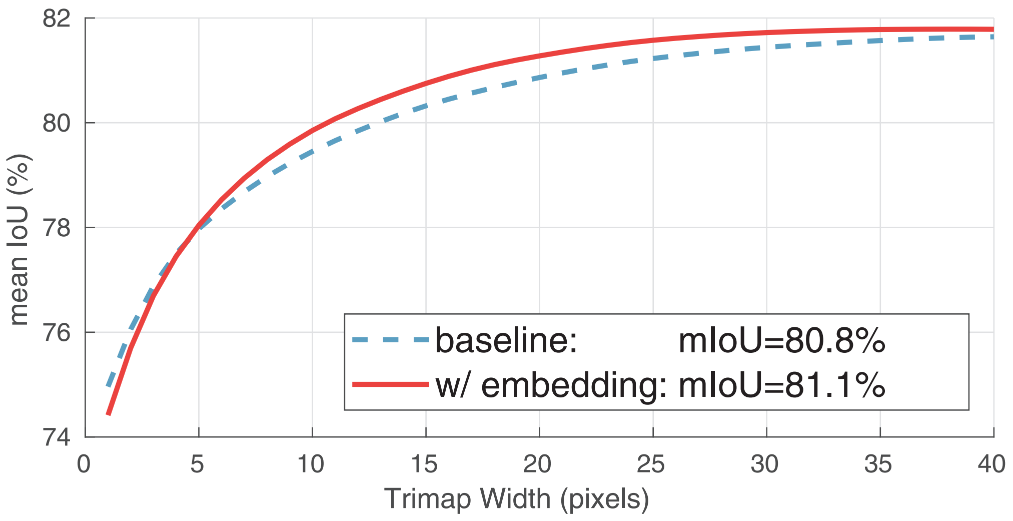


Figure 5.9: Semantic segmentation performance as a function of distance from ground-truth object boundaries comparing a baseline model trained with cross-entropy loss versus a model which also includes embedding loss.

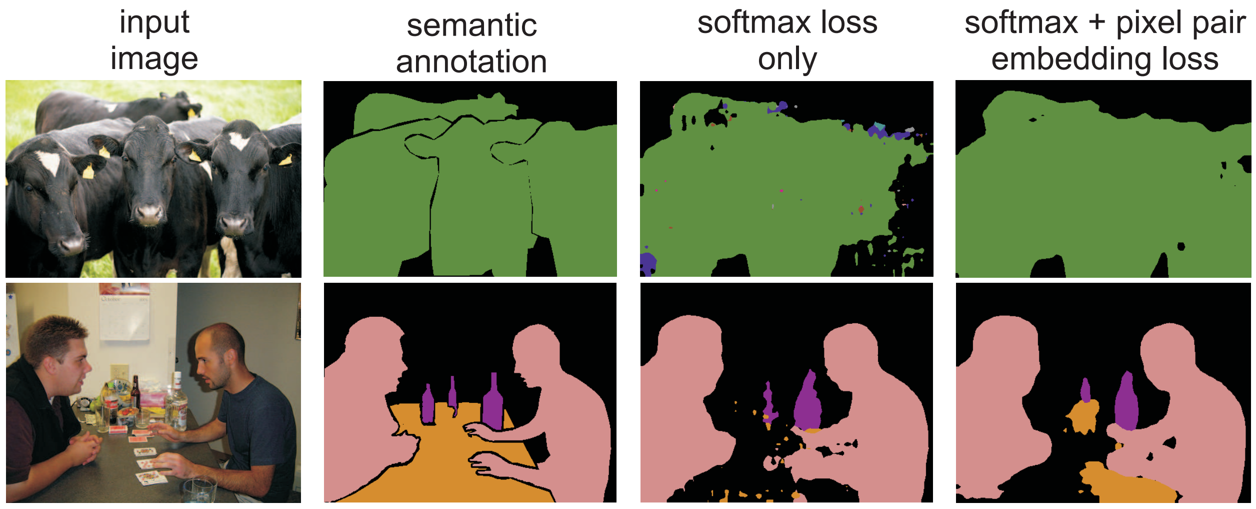


Figure 5.10: The proposed embedding loss improves semantic segmentation by forcing the pixel feature vectors to be similar within the segments. Randomly selected images from PASCAL VOC2012 validation set.

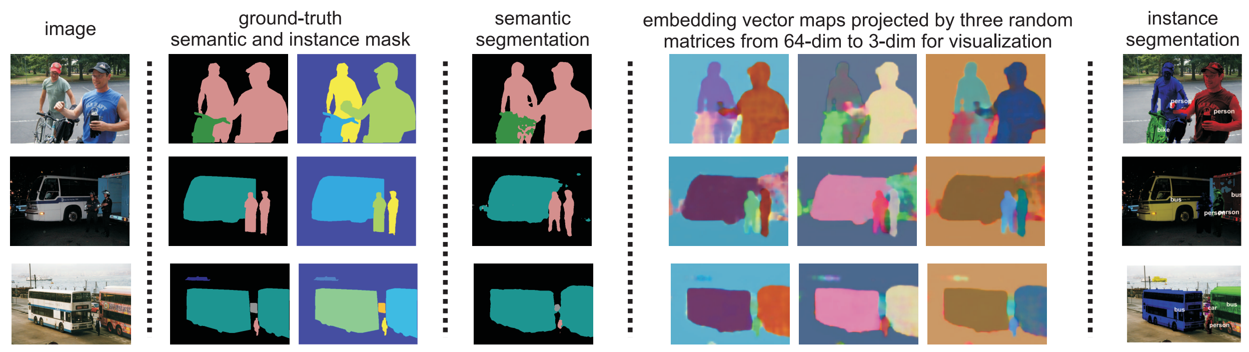


image
ground-truth semantic and instance mask
semantic segmentation
embedding vector maps projected by three random matrices from 64-dim to 3-dim for visualization
instance segmentation
person
person
bike
bus
bus
person
person
bus
car
bus
person

Figure 5.11: Visualization of generic/instance-level semantic segmentation on random PASCAL VOC 2012 validation images.

Chapter 6

Image Reconstruction with Simulated Data

We propose a simple, interpretable framework for solving a wide range of image reconstruction problems such as denoising and deconvolution. Given a single corrupted input image, the model synthesizes a spatially varying linear filter which, when applied to the input image, reconstructs the desired output. The model parameters are learned through supervised or self-supervised training over pairs of clean images and their corrupted counterparts through simulation. We test this model on three tasks: non-uniform motion blur removal, lossy-compression artifact reduction and single image super resolution. We demonstrate that our model performs substantially better or on-par compared to the state-of-the-art methods on all the three tasks, and is significantly faster than optimization-based approaches to deconvolution. Unlike models that directly predict output pixel values or residuals, the predicted filter flow regularizes the output with the input image itself, and is controllable and interpretable, which we demonstrate by visualizing the space of predicted filters for different tasks.

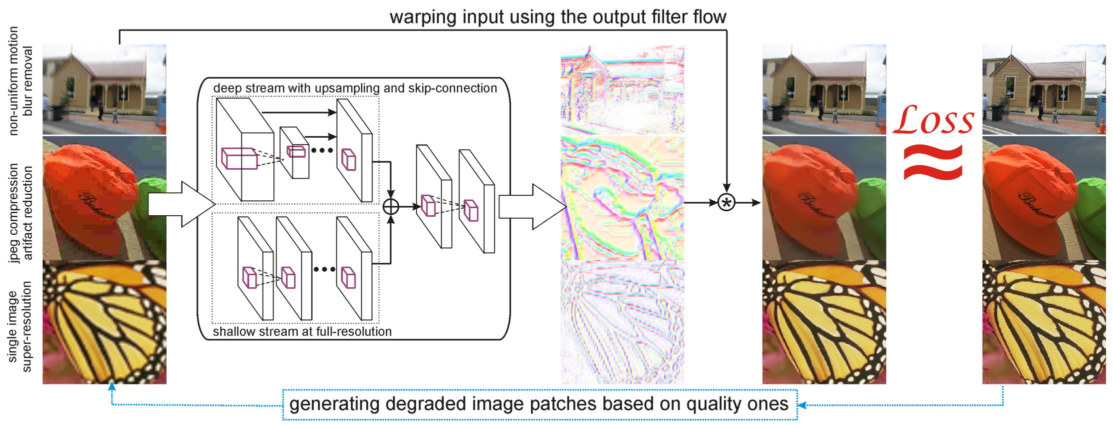

warping input using the output filter flow
non-uniform motion blur removal
jpeg compression artifact reduction
single image super-resolution
deep stream with upsampling and skip-connection
shallow stream at full-resolution
Loss
≈
generating degraded image patches based on quality ones

Figure 6.1: Overview of the proposed *Predictive Filter Flow* which is readily applicable to various low-level vision problems, yielding state-of-the-art performance for non-uniform motion blur removal, compression artifact reduction and single image super-resolution. Given a single corrupted image, a two-stream CNN analyzes the image and synthesizes the weights of spatially-varying linear filters at each pixel. The per-pixel filter is then convolved with the corresponding local region to produce a deblurred/denoised pixel prediction. The whole framework is end-to-end trainable in a self-supervised way for the three aforementioned tasks where corrupted images can be simulated automatically. The predicted filters are easily constrained for different tasks and interpretable (here visualized in the center column by the mean flow displacement, see Fig. 6.6).

6.1 Background

Real-world images are seldom perfect. Practical engineering trade-offs entail that consumer photos are often blurry due to low-light, camera shake or object motion, limited in resolution and further degraded by image compression artifacts introduced for the sake of affordable transmission and storage. Scientific applications such as microscopy or astronomy, which push the fundamental physical limitations of light, lenses and sensors, face similar challenges. Recovering high-quality images from degraded measurements has been a long-standing problem for image analysis and spans a range of tasks such as blind-image deblurring [21, 182, 81, 295], compression artifact reduction [298, 216], and single image super-resolution [258, 363].

Such image reconstruction tasks can be viewed mathematically as inverse problems [316, 150], which are typically ill-posed and massively under-constrained. Many contemporary techniques to inverse problems have focused on regularization techniques which are amenable to computational optimization. While such approaches are interpretable as Bayesian estimators with particular choice of priors, they are often computationally expensive in practice [81, 295, 393, 11, 147]. Alternately, data-driven methods based on training deep convolutional neural networks yield fast inference but lack interpretability and guarantees of robustness [310, 383]. In this paper, we propose a new framework called *Predictive Filter Flow* that retains interpretability and control over the resulting reconstruction while allowing fast inference. The proposed framework is directly applicable to a variety of low-level computer vision problems involving local pixel transformations.

As the name suggests, our approach is built on the notion of Filter Flow introduced by Seitz and Baker [293]. In filter flow, pixels in a local neighborhood of the input image are linearly combined to reconstruct the center pixel in the output image. However, unlike convolution, the filter weights are allowed to vary from one spatial location to the next. Filter flows are a flexible class of image transformations that can model a wide range of imaging effects (including optical flow, lighting changes, non-uniform blur, non-parametric distortion). The original work on filter flow [293] focused on the problem of estimating an appropriately regularized/constrained flow between a given pair of images. This yielded convex but impractically large optimization problems (e.g., hours of computation to compute a single flow [276]). Instead of solving for an optimal filter flow, we propose to directly predict a filter flow given an input image using a convolutional neural net (CNN) to regress the filter weights. Using a CNN to directly predict a well regularized solution is orders of magnitude faster than expensive iterative optimization.

Fig. 6.1 provides an illustration of our overall framework. Instead of estimating the flow between a pair of input images, we focus on applications where the model predicts both the flow and the transformed image. This can be viewed as "blind" filter flow estimation, in analogy with blind deconvolution. During training, we use a loss defined over the transformed image (rather than the predicted flow). This is closely related to so-called self-supervised techniques that learn to predict optical flow and depth from unlabeled video data [93, 101, 144]. Specifically, for the reconstruction tasks we consider such as single image super-resolution, the forward degradation process can be easily simulated to generate a large quantity of training data without manual collection or annotation.

The lack of interpretability in deep image-to-image regression models makes it hard to provide guarantees of robustness in the presence of adversarial input [183], and confer reliability needed for researchers in biology and medical science [217]. Predictive filter flow differs from other CNN-based approaches in this regard since the intermediate filter flows are interpretable and transparent [329, 70, 215], providing an explicit description of how the input is transformed into output [240, 251, 148]. It is also straightforward to inject constraints on the reconstruction (e.g., local brightness conservation) which would be nearly impossible to guarantee for deep image-to-image regression models.

To evaluate our model, we carry out extensive experiments on three different low-level vision tasks: non-uniform motion blur removal, JPEG compression artifact reduction and single image super-resolution. We show that our model achieves substantially better or on-part performance compared to the state-of-the-art methods. We also visualize the predicted filters which reveals filtering operators reminiscent of classic unsharp masking filters and anisotropic diffusion along boundaries.

To summarize our contribution: (1) we propose a novel, end-to-end trainable, learning framework for solving various low-level image reconstruction tasks; (2) we show this framework is highly interpretable and controllable, enabling direct post-hoc analysis of how the reconstructed image is generated from the degraded input; (3) we show experimentally that predictive filter flow achieves better or on-par performance than the state-of-the-art methods on the three different tasks, non-uniform motion blur removal, compression artifact reduction and single image super-resolution.

6.2 Related Work

Our work is inspired by filter flow [293], which is an optimization based method for finding a linear transformation relating nearby pixel values in a pair of images. By imposing additional constraints on certain structural properties of these filters, it serves as a general framework for understanding a wide variety of low-level vision problems. However, filter flow as originally formulated has some obvious shortcomings. First, it requires a set of constraints based on prior knowledge to produce good results. It is not always straightforward to model or even come up with such knowledge-based constraints helpful for prediction. Second, solving for an optimal filter flow is compute-intensive; it takes up to 20 hours to compute over a pair of 500$\times$500 images [293, 276]. We address these by directly predicting flows from images. We leverage predictive filter flow for targeting three specific image reconstruction tasks which can be framed as performing spatially variant filtering over local image patches.

There are several recent works embracing the similar technical idea of predicting spatially variant filters. Zhang *et al.* train recurrent networks to produce spatially variant kernels [376] for image deblur in a fully supervised fashion. Wang *et al.* train a CNN model for dynamic computation inside the model with segmentation probability maps for image super-resolution [342]. The most similar to our work are [240] and [251], which predict per-pixel kernels for burst denoising and video frame interpolation, respectively. Both of them produce per-pixel, per-frame kernels conditioning on two or more frames as input. Our model goes beyond this by predicting the per-pixel filters conditioned on *a single corrupted image* and demonstrating that reconstruction with predicted per-pixel filters performs across a wider range tasks.

Non-Uniform Blind Motion Blur Removal is an extremely challenging yet practically significant task of removing blur caused by object motion or camera shake on a blurry photo. The blur kernel is unknown and may vary over the image. Recent methods estimate blur kernels locally at patch level, and adopt an optimization method for deblurring the patches [310, 11]. Other works [350, 126, 310] leverage prior information about smooth motion by selecting from a predefine discretized set of linear blur kernels. These methods are computationally expensive as an iterative solver is required for deconvolution after estimating the blur kernel [52] while deep learning approaches have difficulties generalizing well to novel motion kernels [359, 310, 126, 291].

Compression Artifact Reduction is of significance as lossy image compression is ubiquitous for reducing the size of images transmitted over the web and recorded on data storage media. However, high compression rates come with visual artifacts that degrade the image quality and thus user experience. Among various compression algorithms, JPEG has become the most widely accepted standard in lossy image compression with several (non-invertible) transforms [334], i.e., downsampling and DCT quantization. Removing artifacts from jpeg compression can be viewed as a practical variant of natural image denoising problems [31, 139]. Recent methods based on deep convolutional neural networks trained to take as input the compressed image and output the denoised image directly achieve good performance [68, 312, 41].

Single Image Super-Resolution aims at recovering a high-resolution image from a single low-resolution image. This problem is inherently ill-posed as a multiplicity of solutions exists for any given low-resolution input. Many methods adopt an example-based strategy [361] requiring an optimization solver, others are based on deep convolutional neural nets [69, 194] which achieve the state-of-the-art and real-time performance. The deep learning methods take as input the low-resolution image (usually upsampled one using bicubic interpolation), and produce residual maps added with input image for the final high-resolution output.

▢ 6.3 Predictive Filter Flow

Filter flow models image transformations $\mathbf{I}_1 \rightarrow \mathbf{I}_2$ as a linear mapping where each output pixel only depends on a local neighborhood of the input. Find such a flow can be framed as solving a constrained linear system

$$\mathbf{I}_2 = \mathbf{T}\mathbf{I}_1, \quad \mathbf{T} \in \mathbf{\Gamma}. \tag{6.1}$$

where $\mathbf{T}$ is a matrix whose rows act separately on a vectorized version of the source image $\mathbf{I}_1$. For the model (6.1) to make sense, $\mathbf{T} \in \mathbf{\Gamma}$ must serve as a placeholder for the entire set of additional constraints on the operator which enables a unique solution that satisfies our expectations for particular problems of interest. For example, standard convolution corresponds to $\mathbf{T}$ being a circulant matrix whose rows are cyclic permutations of a single set of filter weights which are typically constrained to have compact localized non-zero support. For a theoretical perspective, Filter Flow model is simple and elegant, but directly solving it is intractable for image sizes we typically encounter in practice, particularly when the filters are allowed to vary spatially.

▢ 6.3.1 Learning to predict filter flow

Instead of optimizing over $\mathbf{T}$ directly, we seek for a learnable function $f_{\mathbf{w}}(\cdot)$ parameterized by $\mathbf{w}$ that predicts the transformation $\hat{\mathbf{T}}$ specific to image $\mathbf{I}_1$ taken as input:

$$\mathbf{I}_2 \approx \hat{\mathbf{T}}\mathbf{I}_1, \quad \hat{\mathbf{T}} \equiv f_{\mathbf{w}}(\mathbf{I}_1), \tag{6.2}$$

We call this model Predictive Filter Flow. Manually designing such a function $f_{\mathbf{w}}(\cdot)$ isn't feasible in general, therefore we learn a specific $f_{\mathbf{w}}$ under the assumption that $\mathbf{I}_1, \mathbf{I}_2$ are drawn from some fixed joint distribution.

Given sampled image pairs, $\{(\mathbf{I}_1^i, \mathbf{I}_2^i)\}_{i=1}^N$, we seek parameters $\mathbf{w}$ to minimize the difference between a recovered image $\hat{\mathbf{I}}_2$ and the real one $\mathbf{I}_2$ measured by the loss ℓ:

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \ell(\mathbf{I}_2^i - f_{\mathbf{w}}(\mathbf{I}_1^i) \cdot \mathbf{I}_1^i) + \mathcal{R}(f_{\mathbf{w}}(\mathbf{I}_1^i)) \tag{6.3}$$

In practice, we enforce hard constraints via our choice of the architecture/functional form of f along with soft-constraints via additional regularization term $\mathcal{R}$, as detailed below. There are a range of possible choices for measuring the difference between two images (including e.g., adversarial losses [194]). In our experiments, we simply use the robust L_1 norm to measure the pixel-level difference.

Filter locality In principle, each pixel output $\mathbf{I}_2$ in Eq. 6.3 can depend on all input pixels $\mathbf{I}_1$. We introduce the structural constraint that each output pixel only depends on a corresponding local neighborhood of the input. The size of this neighborhood is thus a hyper-parameter of the model. We note that while the predicted filter flow $\hat{\mathbf{T}}$ acts locally, the output of local filter flow within a patch is determined with global context captured by large receptive fields in the predictor $f_{\mathbf{w}}(\cdot)$.

In practice, we implement this constraint with the "im2col" operation which vectorizes the local neighborhood patch centered at each pixel and computes the inner product of this vector with the corresponding predicted filter. Note that the im2col operation and the follow-up inner product are highly optimized for available hardware architectures in most deep learning libraries, *exactly the same* used in modern convolution operation; thus our model is quite efficient in computation. For example, if the filter size is 20×20, the last layer of the CNN model $f_{\mathbf{w}}(\cdot)$ outputs a three-dimensional array with a channel dimension of 400, which is comparable in size to feature activations at a single layer of typical CNN architectures [179, 304, 115]. Note that, in image debluring, the per-pixel filter size is largely independent to the blur kernel size — our model learns to reconstruct pixels with even bigger receptive field telling it how to reconstruct the pixels with limited budget (i.e., neighborhood size specified by the per-pixel filter size).

Other filter constraints Various priori constraints on the filter flow $\hat{\mathbf{T}} \equiv f_{\mathbf{w}}(\mathbf{I}_1)$ can be added easily to enable better model training. For example, if smoothness is desired, an L_2 regularization on the (1st order or 2nd order) derivative of the filter flow maps can be inserted during training; if sparsity is desired, an L_1 regularization on the filter flows can be added easily. In literature of image deblur/denoising, the sum-to-one and non-negative constraints are usually exploited corresponding to a reasonable assumption that blurring an image with a slowly varying kernel does not change the overall brightness [128, 14]. In our work, we add these constraints on the filters for the task of non-uniform motion blur removal. This can be easily done by inserting a softmax transform across channels of the predicted filter weights. For other tasks, we simply let the model output free-form filters with no further constraints on the weights, so that it is able to enhance over block effects (e.g., in JPEG compressed images).

Self-Supervision Though the proposed framework for training Predictive Filter Flow requires paired inputs and target outputs, we note that generating training data for many reconstruction tasks can be accomplished automatically without manual labeling. Given a pool of high quality images, we can automatically generate low-resolution, blurred or JPEG degraded counterparts to use in training (see Section 6.4). This can also be generalized to so-called self-supervised training for predicting flows between video frames or stereo pairs.

◻ 6.3.2 Model Architecture and Training

Our basic framework is largely agnostic to the choice of architectures, learning method, and loss functions. In our experiments, we utilize to a two-stream architecture as shown in Fig. 6.1. The first stream is a simple ResNet18 [115] network with 3×3 convolutional

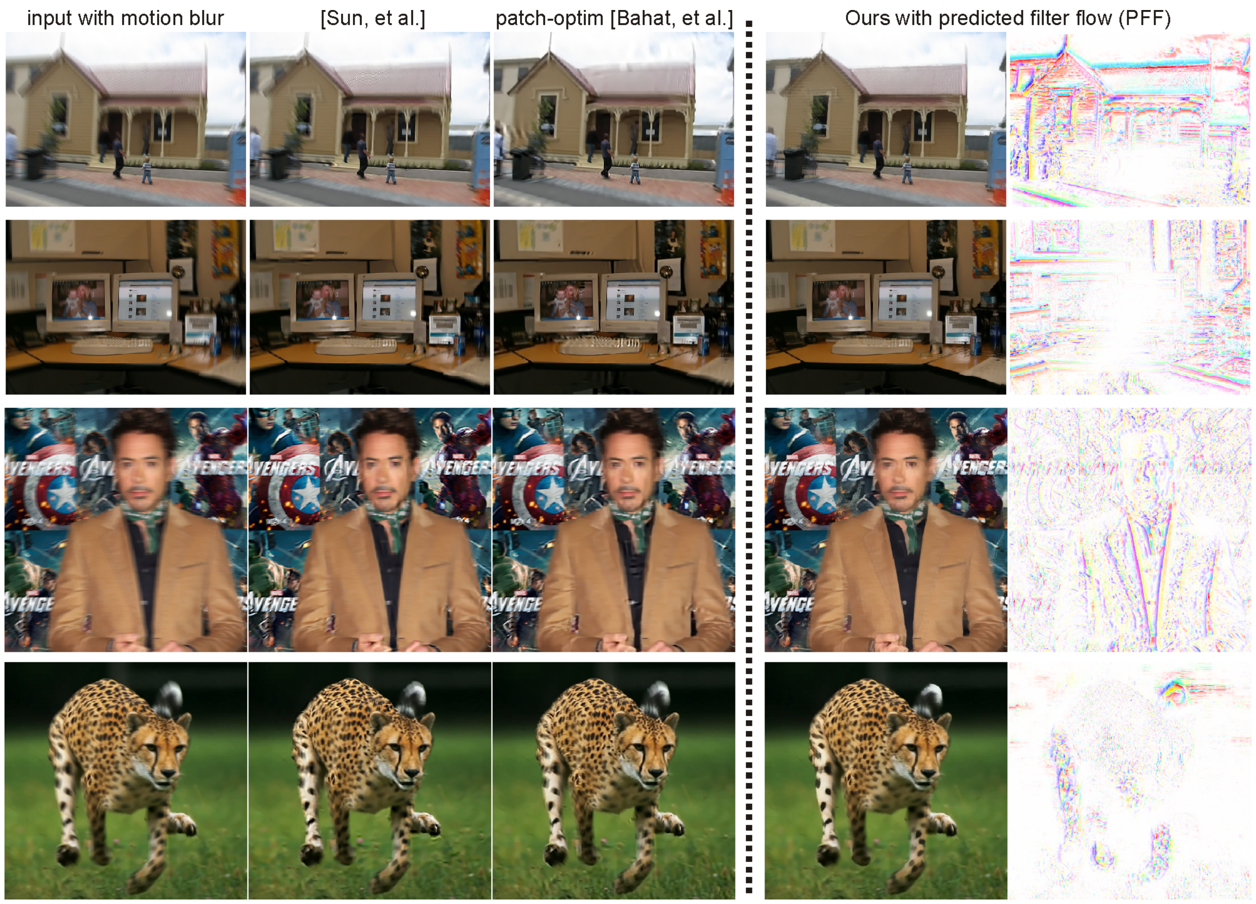

input with motion blur
[Sun, et al.]
patch-optim [Bahat, et al.]
Ours with predicted filter flow (PFF)
VENGERS
AVENGERS

Figure 6.2: Visual comparison of our method (*PFF*) to *[Sun, et al.]* [310] and *patch-optim [Bahat, et al.]* [11] on testing images released by [11]. Please be guided with the strong edges in the filter flow maps to compare visual details in the deblurred images by different methods. Also note that the bottom two rows display images from the real-world, meaning they are not synthesized and there is no blur ground-truth. Best view in color and zoom-in.

layers, skip connections [115], pooling layers and upsampling layers; the second stream is a shallow but full-resolution network with no pooling. The first stream has larger receptive fields for estimating per-pixel filters by considering long-range contextual information, while the second stream keeps same resolution as input image that helps compensate the aliasing effect [377] caused by the (un)pooling operation in the first stream. Batch normalization [137] is also inserted between a convolution layer and ReLU [247]. The Predictive Filter Flow is self-supervised so we could generate an unlimited amount of image pairs for training large models. However, we find a small ResNet18-based model trained over moderate-scale training set performs quite well. Since our architecture is different from other feed-forward image-to-image regression CNNs, we also report the baseline performance of the two-stream architecture trained to directly predict the image rather than the filter coefficients.

For training, we crop 64×64-resolution patches to form a batch of size 56. Since the model adapts to patch boundary effects seen during training, at test time we apply it to non-overlapping tiles of the input image; as a benefit, this makes our approach very amenable to parallelization. Moreover, we note that the model is fully convolutional so it could be trained over larger patches to avoid boundary effects and applied to arbitrary size inputs.

We use ADAM optimization method during training [156], with initial learning 0.0005 and coefficients 0.9 and 0.999 for computing running averages of gradient and its square. As for the training loss, we simply use the ℓ_1-norm loss measuring absolute difference over pixel intensities. We train our model from scratch using PyTorch [260] on a single NVIDIA TITAN X GPU, and terminate after several hundred epochs (100K update iteration)*.

■ 6.4 Experiments

We evaluate the proposed Predictive Filter Flow framework (PFF) on three low-level vision tasks: non-uniform motion blur removal, JPEG compression artifact reduction and single image super-resolution. We first describe the datasets and evaluation metrics, and then compare with state-of-the-art methods on the three tasks in separate subsections, respectively.

■ 6.4.1 Datasets and Metrics

We train all our models from scratch without pre-training on ImageNet [?]. For image super-resolution, we follow standard practice in the literature of using the 800 HR images

*Each model is trained within 2 days. The code and models can be found in https://github.com/aimerykong/predictive-filter-flow

Table 6.1: Comparison on motion blur removal over the non-uniform motion blur dataset [11]. Moderate/large blur are from motion up to 16/38 pixels. For the two metrics, the larger value means better performance of the model.

	Moderate Blur				
metric	[360]	[310]	[11]	CNN	**PFF**
PSNR	22.88	24.14	24.87	24.29	**25.18**
SSIM	0.68	0.714	0.743	0.708	**0.767**
	Large Blur				
metric	[360]	[310]	[11]	CNN	**PFF**
PSNR	20.47	20.84	22.01	20.87	**22.12**
SSIM	0.54	0.56	0.624	0.539	**0.617**

in the DIV2K dataset [5] to train our model. For the other two tasks, we additionally include the BSDS500 training set [234] to train the models (1,200 training images in total). We evaluate each model over different datasets specific to the task. Concretely, we test our model for non-uniform motion blur removal over the dataset introduced in [11], which contains large motion blur up to 38 pixels. We evaluate over the classic LIVE1 dataset [344] for JPEG compression artifacts reduction; Set5 [24], Set14 [373] and Urban [131] for single image super-resolution.

To quantitatively measure performance, we use Peak-Signal-to-Noise-Ratio (PSNR) and Structural Similarity Index (SSIM) [344] between the output quality image and the original image (over the RGB for image deblurring task and the Y channel in YCbCr color space for the other two tasks). This is a standard practice in literature of the three tasks for quantitatively measuring the recovered image quality.

6.4.2 Non-Uniform Motion Blur Removal

To train models for non-uniform motion blur removal, we generate the 64×64-resolution blurry patches from clear ones using random linear kernels [310], which are of size 30×30 and have motion vector with random orientation in $[0, 180°]$ degrees and random length in

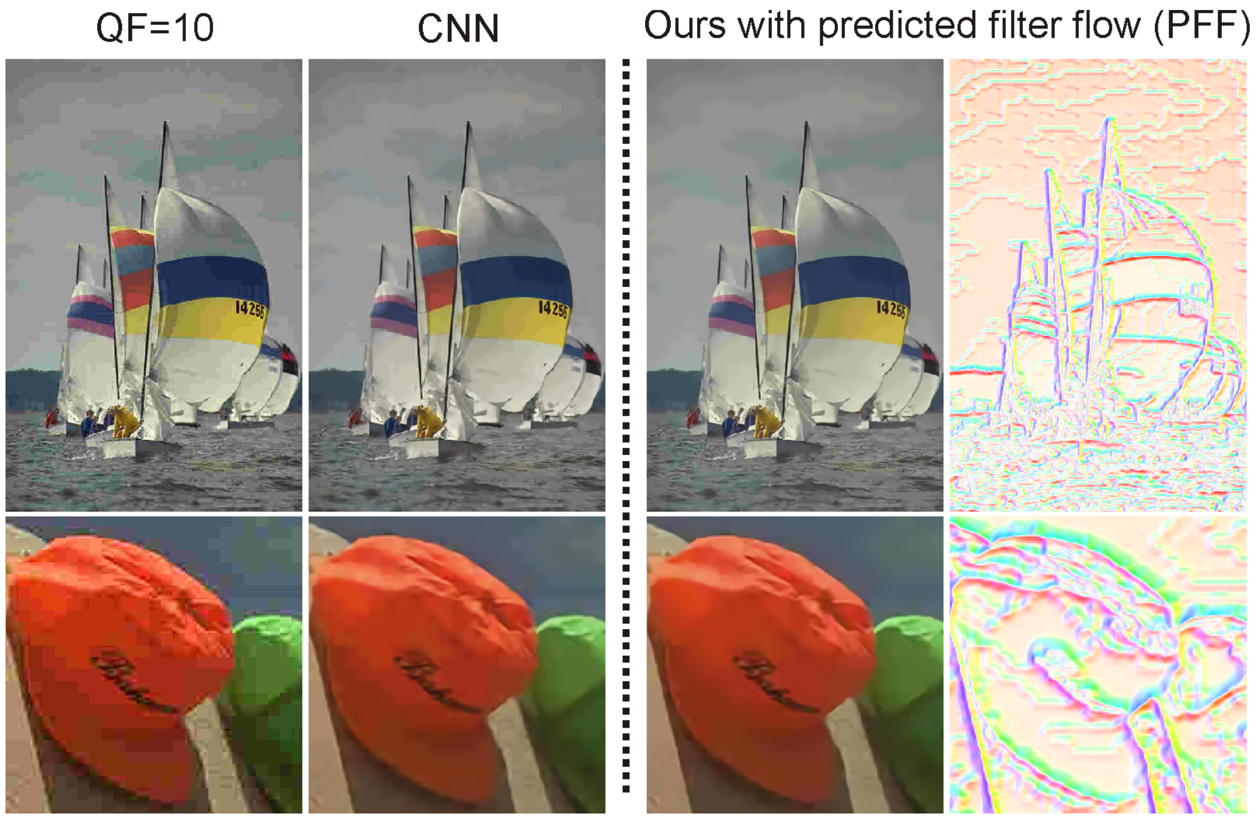


Figure 6.3: Visual comparison of our methods (PFF and CNN). Strong edges in the expected flow map (right) highlight areas where most apparent artifacts are removed. Best viewed in color and zoomed-in.

$[1, 30]$ pixels. We set the predicted filter size to be 17×17 so the model outputs 17×17=289 filter weights at each image location. Note that we generate training pairs on the fly during training, so our model can deal with a wide range of motion blurs. This is advantageous over methods in [310, 11] which require a predefined set of blur kernels used for deconvolution through some offline algorithm.

In Table 6.1, we list the comparison with the state-of-the-art methods over the released test set by [11]. There are two subsets in the dataset, one with moderate motion blur and the other with large blur. We also report our CNN models based on the proposed two-stream architecture that outputs the quality images directly by taking as input the blurry ones. Our CNN model outperforms the one in [310] which trains a CNN for predicting the blur kernel over a patch, but carries out non-blind deconvolution with the estimated kernel for the final quality image. We attribute our better performance to two reasons. First, our CNN model learns a direct inverse mapping from blurry patch to its clear counterpart based on the learned image distribution, whereas [310] only estimates the blur kernel for the patch and uses an offline optimization for non-blind deblurring, resulting in some artifacts such as ringing. Second, our CNN architecture is higher fidelity than the one used in [310], as ours outputs full-resolution result and learns internally to minimize artifacts, e.g., aliasing and ringing effect.

From the table, we can see our PFF model outperforms all the other methods by a fair margin. To understand where our model performs better, we visualize the qualitative results in Fig. 6.2, along with the filter flow maps as output from PFF. We can't easily visualize the 289 dimensional filters. However, since the predicted weights $\hat{T}$ are positive and L_1 normalized, we can treat them as a distribution which we summarize by computing the expected flow vector

$$\begin{bmatrix} v_x(i,j) \\ v_y(i,j) \end{bmatrix} = \sum_{x,y} \hat{T}_{ij,xy} \begin{bmatrix} x-i \\ y-j \end{bmatrix}$$

where ij is a particular output pixel and xy indexes the input pixels. This can be interpreted as the optical flow (delta filter) which most closely approximates the predicted filter flow. We use the the color legend shown in top-left of Fig. 6.6.

The last two rows of Fig. 6.2 show the results over real-world blurry images for which there is no “blur-free” ground-truth. We can clearly see that images produced by PFF have less artifacts such as ringing artifacts around sharp edges [310, 11]. Interestingly, from the filter flow maps, we can see that the expected flow vectors are large near high contrast boundaries and smaller in regions that are already in sharp focus or which are uniform in color.

Although we define the filter size as 17×17, which is much smaller than the maximum shift in the largest blur (up to 30 pixels), our model still handles large motion blur and performs

Table 6.2: Comparison on JPEG compression artifact reduction over LIVE1 dataset [344]. PSNR and SSIM are used as metrics listed on two rows respectively in each macro row grid (the larger the better).

QF	JPEG	SA-DCT [87]	AR-CNN [68]	L4 [312]	CAS-CNN [41]	MWCNN [220]	**PFF**
10	27.77	28.65	29.13	29.08	29.44	29.69	**29.82**
	0.791	0.809	0.823	0.824	0.833	0.825	**0.836**
20	30.07	30.81	31.40	31.42	31.70	32.04	**32.14**
	0.868	0.878	0.890	0.890	0.895	0.889	**0.905**
40	32.35	32.99	33.63	33.77	34.10	34.45	**34.67**
	0.917	0.940	0.931	—	0.937	0.930	**0.949**

better than [11]. While this is owing to the fact that the model is trained to reconstruct pixels with limited budget (the filter size), we assume it should be possible to utilize larger filter sizes but we did not observe further improvements when training models to synthesize larger per-pixel kernels. This suggests that a larger blurry dataset is needed to validate this point in future work.

We also considered an iterative variant of our model in which we feed the resulting deblurred image back as input to the model. However, we found relatively little improvement with additional iterations (results shown in the supplementary material). We conjecture that, although the model was trained with a wide range of blurred examples, the statistics of the transformed image from the first iteration are sufficiently different than the blurred training inputs. One solution could be inserting adversarial loss to push the model to generate more fine-grained textures (as done in [194] for image super-resolution).

6.4.3 JPEG Compression Artifact Reduction

Similar to training for image deblurring, we generate JPEG compressed image patches from original non-compressed ones on the fly during training. This can be easily done using JPEG compression function by varying the quality factor (QF) of interest.

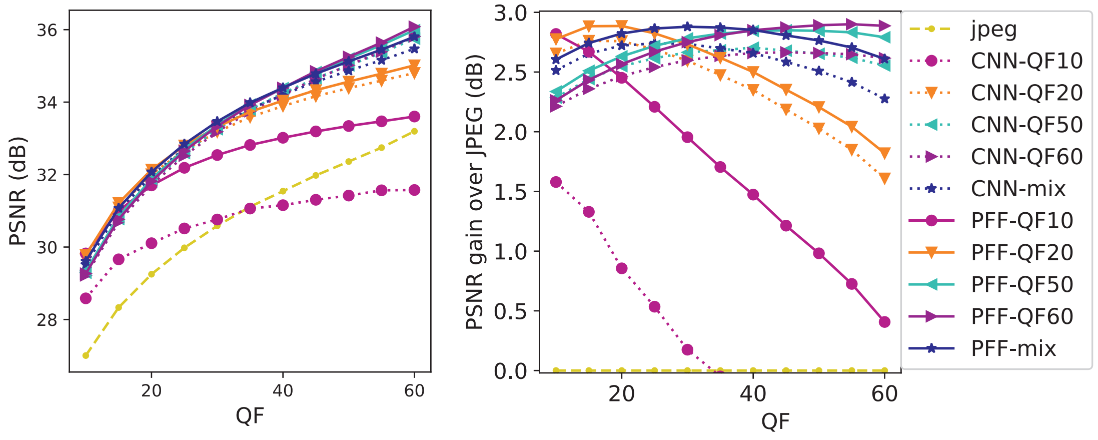
In Table 6.2, we list the performance of our model and compare to the state-of-the-art methods. We note that our final PFF achieves the best among all the methods. Our CNN baseline model also achieves on-par performance with state-of-the-art, though we do not show in the table, we draw the performance under the ablation study in Fig. 6.4. Specifically, we study how our model trained with single or a mixed QFs affect the performance when tested on image compressed with a range of different QFs. We plot the detailed performances of our CNN and PFF in terms of absolute measurements by PSNR and SSIM, and their performance gains of reconstructed images compared to the JPEG compressed image.

We can see that, though a model trained with QF=10 overfits the dataset, all the other models achieve generalizable and stable performance. Basically, a model trained on a single QF brings the largest performance gain over images compressed with the same QF. Moreover, when our model is trained with mixed quality factors, its performance is quite stable and competitive with quality-specific models across different compression quality factors. This shows our model is of practical value in real-world applications.

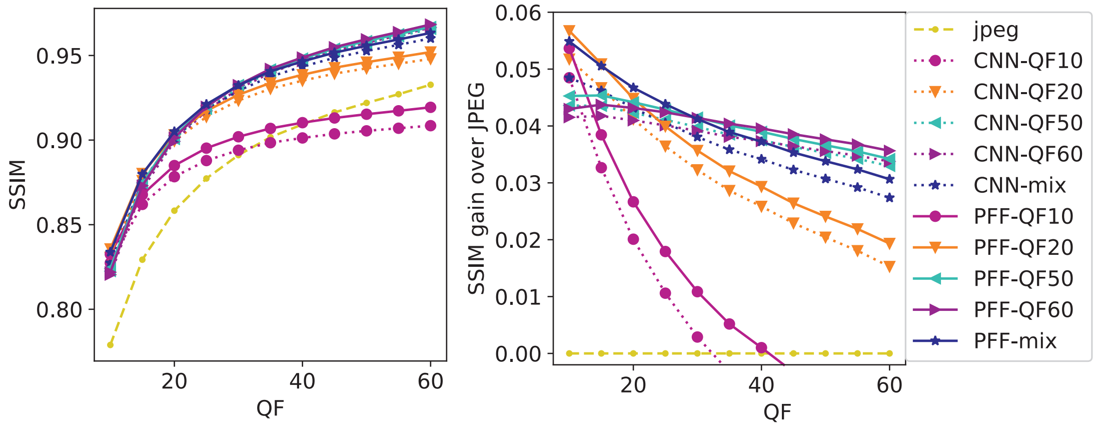
In Fig. 6.3, we demonstrate qualitative comparison between CNN and PFF. The output filter flow maps indicate from the colorful edges how the pixels are warped from the neighborhood in the input image. This also clearly shows where the JPEG image degrades most, e.g., the large sky region is quantized by JPEG compression. Though CNN makes the block effect smooth to some extent, our PFF produces the best visual quality, smoothing the block artifact while maintaining both high- and low-frequency details.

6.4.4 Single Image Super-Resolution

To generate training image pairs, for each original image, we downsample $\frac{1}{4}\times$ and upsample $4\times$ again using bicubic interpolation (with anti-aliasing). The $4\times$ upsampled image from the low-resolution is the input to our model, which is thus expected to learn to sharpen and interpolate residual detail lost due to the downsampling process.



PSNR improvements.



SSIM improvements.

Figure 6.4: Performance vs. training data with different compression quality factors measured by PSNR and SSIM and their performance gains, over the LIVE1 dataset. The original JPEG compression is plotted for baseline.

From the quantitative comparison listed in Table 6.3, we can see that our PFF model with the same (un-optimized) architecture used for all tasks achieves comparable performance with the state-of-the-art methods. While our approach could be combined with other techniques such self-ensemble [207, 383], multiscale convolution [201] and attentional mechanism [381], we did not exploit them as it is beyond the scope of this paper. In Fig. 6.5, we compare visually the outputs of bicubic interpolation, CNN and PFF. We can see from the zoom-in regions that our PFF model generates sharper boundaries and delivers an anti-aliasing functionality. The filter flow visualization maps illustrate where the smoothing happens and

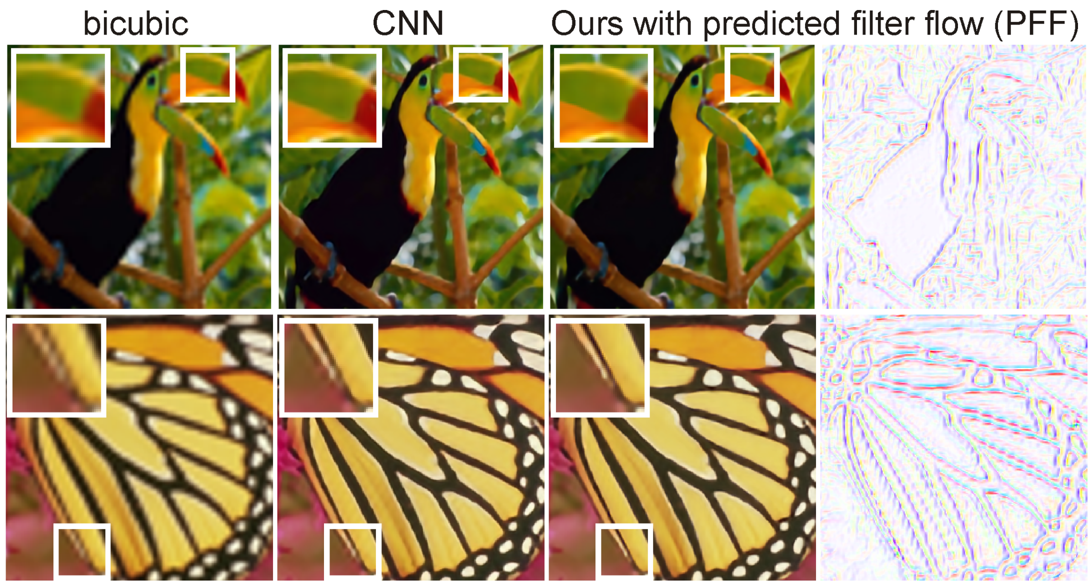


Figure 6.5: Visual comparison of our method (PFF) to CNN, each image is super-resolved ($\times 4$). More results can be found in the supplementary material. Best view in color and zoom-in.

where sharpening happens (strong colorful edges in the visualization indicate where pixels undergo larger transformations). In next section, we visualize the per-pixel kernels to provide a more in-depth analysis.

6.5 Visualization and Analysis

We have explored a number of techniques to visualize the predicted filter flows for different tasks. First, we ran k-means on predicted filters from the set of test images for each the three tasks, respectively, to cluster the kernels into K=400 groups. Then we run t-SNE [226] over the 400 mean filters to display them in the image plane, shown by the scatter plots in top row of Fig. 6.6. Qualitative inspection shows filters that can be interpreted as performing translation or integration along lines of different orientation (non-uniform blur), filling in high-frequency detail (jpeg artifact reduction) and deformed Laplacian-like filters (super-resolution).

Table 6.3: Comparison on single image super-resolution ($\times 4$) over the classic Set5 [24], Set14 [373] and Urban [131] datasets. The metrics used here are PSNR/SSIM ($\times 10^{-2}$).

	Bicubic	KK	A+	SRCNN	SelfEx	RDN+	VDSR	LapSRN	RCAN	MSRN	**PFF**
		[155]	[318]	[69]	[131]	[383]	[154]	[187]	[381]	[201]	*(Ours)*
Set5	28.42/81.04	29.69/84.19	30.28/86.03	30.49/86.28	30.34/85.93	32.61/90.03	31.36/87.96	31.54/88.11	32.63/90.02	32.07/89.03	32.17/89.46
Set14	26.00/70.19	26.85/73.52	27.32/74.91	27.50/75.13	27.55/75.11	28.92/78.93	28.11/76.24	28.19/76.35	28.87/78.89	28.60/77.51	28.43/78.21
Urban	23.14/65.73	24.20/71.04	24.34/71.95	24.52/72.21	24.83/74.03	26.61/80.28	25.18/75.43	25.21/75.64	26.82/80.87	26.04/78.96	25.34/78.14

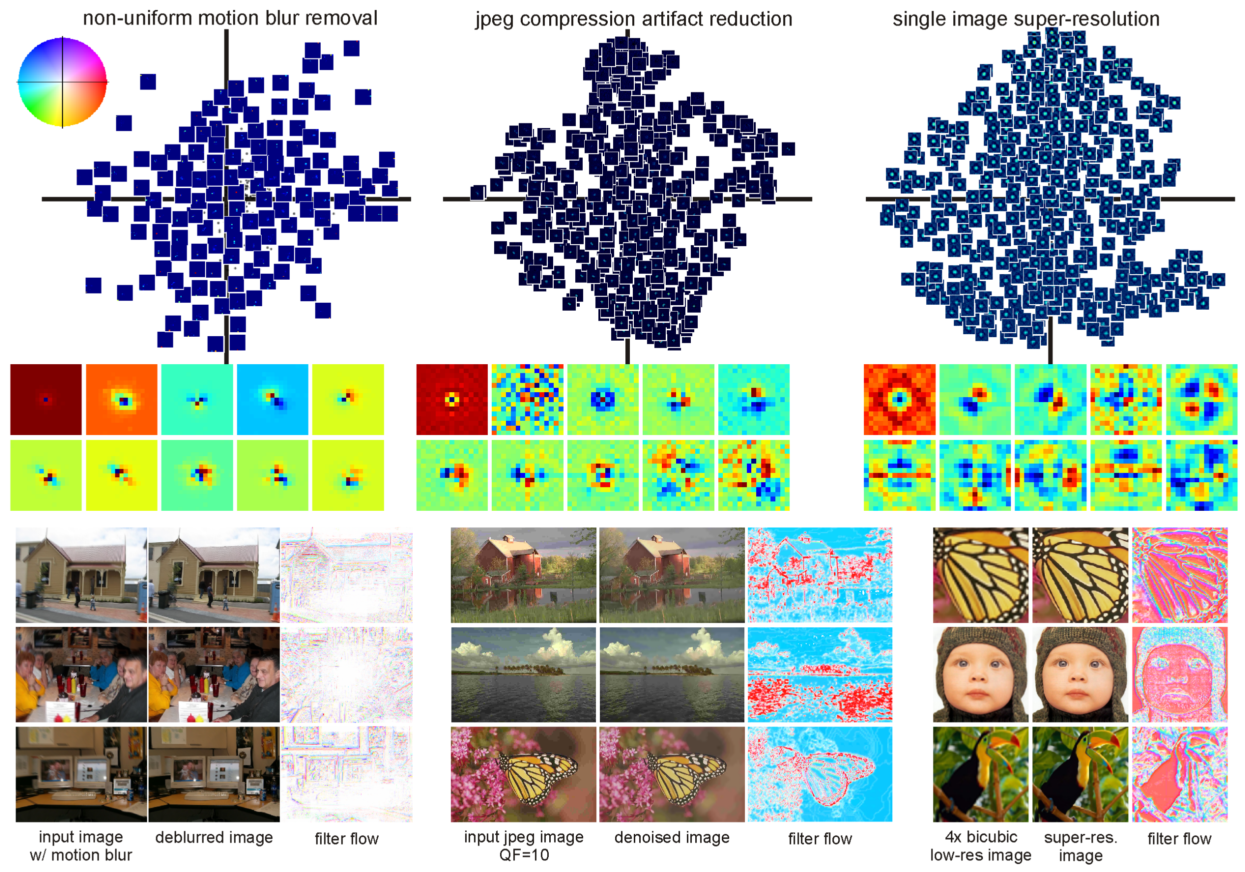


Figure 6.6: Three row-wise panels: (1) We run K-means (K=400) on all filters synthesized by the model over the test set, and visualize the 400 centroid kernels using t-SNE on a 2D plane; (2) top ten principal components of the synthesized filters; (3) visualizing the color coded filter flow along with input and quality image. Each pixels filter is assigned to the nearest centroid and the color for the centroid is based on the 2D t-SNE embedding using the color chart shown at top left.

We also examined the top 10 principal components of the predicted filters (shown in the second row grid in Fig. 6.6). The 10D principal subspace capture 99.65%, 99.99% and 99.99% of the filter energy for non-uniform blur, artifact removal and super resolution respectively. PCA reveals smooth, symmetric harmonic structure for super-resolution with some intriguing vertical and horizontal features.

Finally, in order to summarize the spatially varying structure of the filters, we use the 2D t-SNE embedding to assign a color to each centroid (as given by the reference color chart shown top-left), and visualize the nearest centroid for the filter at each filter location in the third row grid in Fig. 6.6. This visualization demonstrates the filters as output by our model generally vary smoothly over the image with discontinuities along salient edges and textured regions reminiscent of anisotropic diffusion or bilateral filtering.

In summary, these visualizations provide a transparent view of how each reconstructed pixel is assembled from the degraded input image. We view this as a notable advantage over other CNN-based models which simply perform image-to-image regression. Unlike activations of intermediate layers of a CNN, linear filter weights have a well defined semantics that can be visualized and analyzed using well developed tools of linear signal processing.

6.6 Conclusion and Future Work

We propose a general, elegant and simple framework, Predictive Filter Flow, which has direct applications to a broad range of image reconstruction tasks. Our framework generates space-variant per-pixel filters which are easy to interpret and fast to compute at test time. Through extensive experiments over three different low-level vision tasks, we demonstrate this approach outperforms the state-of-the-art methods.

In our experiments, we only train models over patches. However, we believe global image context is also important for these tasks and is an obvious direction for future work. For example, the global blur structure conveys information about camera shake; super-resolution and compression reduction can benefit from long-range interactions to reconstruct high-frequency detail (as in non-local means). Finally, we expect that the interpretability of the output will be particularly appealing for interactive and scientific applications such as medical imaging and biological microscopy where predicted filters could be directly compared to physical models of the imaging process.

Chapter 7

Self-Supervised Learning with Videos

We introduce multigrid Predictive Filter Flow (mgPFF), a low-level vision framework for unsupervised learning on free-form videos. It takes as input a pair of frames and outputs per-pixel filters to warp one frame to the other. Compared to optical flow, filter flow is more expressive in handling sub-pixel movement and transparency (e.g., motion blur). We develop a multigrid coarse-to-fine strategy that avoids the requirement of learning big filters to capture large displacement. This allows us to train a very compact model that operates progressively over multiple resolutions with shared weights. Through experiments, we show mgPFF not only reconstructs frames almost perfectly, but also is readily amendable for video object segmentation/tracking and long-range flow learning, where it outperforms other mid-level state-of-the-art by a notable margin. Moreover, we can use the predicted filters to track the correspondence of individual pixels across time and perform novel interactive photo editing-by-reconstruction.

7.1 Background

Videos contain rich information for humans to understand the scene and interpret the world. However, providing detailed per-frame ground-truth labels is challenging for large-scale video datasets, prompting work on leveraging weak supervision such as video-level labels to learn visual features for various tasks [34, 3, 151, 89]. Video constrained to contain primarily ego-motion has also been leveraged for unsupervised learning of stereo, depth, odometry, and optical flow [330, 101, 387, 294, 374].

Cognitively, a newborn baby can easily track an object without understanding any high-level semantics after immersed in the ambient environment for only one month [103, 245, 243, 331]. However, until recently few works have demonstrated effective unsupervised learning on free-form videos*. For example, Wei *et al.* exploit the arrow-of-time property [272, 102] to learn visual features by predicting frames' temporal order [345], and show its usefulness in action classification and video forensic analysis. Vondrick *et al.* use video colorization as a proxy task and show that the learned features capture objects and parts that are useful for tracking [333]. Wang *et al.* propose to reconstruct mid-level features of frames with cost volume, and apply the trained model to visual tracking [340].

In this chapter we introduce a low-level vision approach for training on unsupervised, free-form videos, which we call multigrid *Predictive Filter Flow* (*mgPFF*), as illustrated by the conceptual flowchart in Fig. 7.1. The mgPFF makes direct, fine-grained predictions of per-pixel filters, as opposed to per-pixel offsets by optical flow, deciding how to reconstruct a video frame from pixels in the previous frame. While mgPFF is trained using a simple photometric reconstruction error and thus expected to perform well in low-level vision tasks (*e.g.* , image reconstruction and editing), we find these per-pixel filters can generate flows that are accurate enough to carry out high-level tasks such as video object segmentation and tracking.

A popular approach for learning flow from reconstruction in video is to employ the differentiable spatial transform (ST) layer (a.k.a grid sampling) [138]. ST-layer learns to output

*By "Free-form", we emphasize videos are unlabled, long (versus short synthetic ones [71, 141]) and camera-agnostic, and without either structural patterns (e.g., ego-motion [96, 57, 367]) or restricted background [321, 283].

per-pixel coordinate offset for sampling pixels with bilinear interpolation, and applies the transform to reconstructing frames. While it works quite well on global transformation and has been widely used in unsupervised optical flow learning [281, 141, 238, 144, 343], we and others observe that unsupervised learning on free-form videos with a simple ST-layer is challenging. Detlefsen *et al.* give an excellent explanation on why it is hard to train the ST-layer [306]. Briefly, training with ST-layer requires the invertibility of the spatial transform which is not guaranteed during training. Additionally, we note that fixed grids for sampling (usually 2×2 for bilinear interpolation) typically only provide meaningful gradients once the predicted flow is nearly correct (*i.e.* within 1 pixel of the correct flow). This necessitates the use of spatial smoothness constraint and the multi-scale losses to avoid getting caught in bad local-minima. We formally study this with experimental comparison between filter flow and optical flow learning, as detailed in Section 7.2.3 and demonstrated in Fig. 7.2.

Inspired by the conceptual framework Filter Flow [293], we propose to diretly learn per-pixel filters instead of the per-pixel offset as in the ST-layer. For each output pixel, we predict the weights of a filter kernel that when applied to the input frame reconstructs the output. In this sense, we reverse the order of operations from the ST-layer: rather than predicting an offset and then constructing filter weights (*i.e.* bilinear kernel), mgPFF directly predicts filters which can vote for the offset vector. We observe that training mgPFF is substantially easier since mgPFF provides useful gradient for all possible flow vectors rather than just those near the current prediction, as verified in Fig. 7.2 and Section 7.2.1.

Since the Filter Flow [293] outputs per-pixel kernels, capturing large displacements with big kernels is computationally expensive. We address this using a multigrid strategy [320, 107] to approximate the kernel. Concretely, we run the model over multi-resolution inputs with a fixed filter size (11×11 used in this work unless specified) and compose the filters generated at multiple scales to produce the final flow fields, as detailed in Section 7.2.4 and illustrated by Fig. 7.3. The model thus only outputs 11*11=121 per-pixel filter weights at each resolution

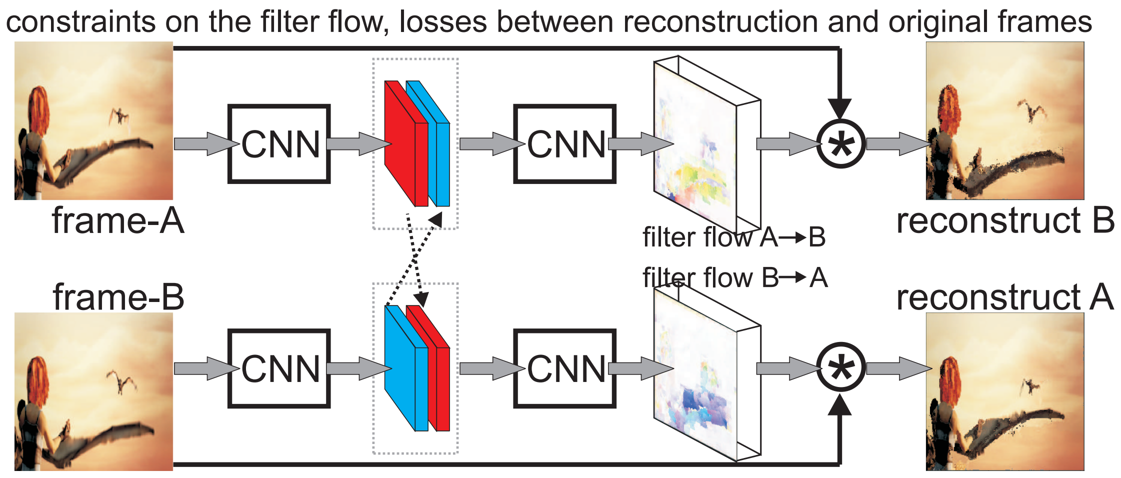


Figure 7.1: The flowchart of multigrid Predictive Filter Flow framework (mgPFF). Conceptually we draw a single scale for demonstrating how we train our model in an unsupervised way with the photometric reconstruction loss along with constraints imposed on the filter flow maps. The multigrid strategy is illustrated in Fig. 7.3.

scale (smaller than the channel dimension in modern CNN architectures). We further assume self-similarity across scales and learn only a single set of shared model weights. This makes mgPFF quite efficient w.r.t running time and model size. Our final (un-optimized) model is only **4.6MB** in size and takes 0.1sec to process a pair of 256×256-pixel resolution images.

To summarize our contributions: (1) conceptually, we introduce a simple, low-level framework, multigrid Predictive Filter Flow (mgPFF), which allows for unsupervised learning on free-form videos; (2) technically, we show the filter flow overcomes the limitation of spatial-transform layer and the multigrid strategy significantly reduces model size; (3) practically, we show that mgPFF substantially outperforms other mid-level state-of-the-art methods on frame reconstruction, video object segmentation and tracking.

▢ 7.2 Related Work

Unsupervised Learning for Vision: Our work builds upon a flurry of recent work that trains visual models without human supervision. A common approach is to leverage the natural context in images and video for learning the visual features [65, 256, 145, 66, 338, 379, 189, 262, 337, 332, 252, 261], which can be transferred to down-stream tasks. Other approaches include interaction with an environment to learn visual features [268, 4, 352], which are useful for robotics. A related but different line of work explores how to learn geometric properties or cycle consistencies with self-supervision, for example for motion capture or correspondence [321, 387, 388, 136, 389, 340]. Ours also develop an unsupervised model, but with the signal from temporal consistency between consecutive frames in free-form videos, without the requirement of synthetic data [388, 136].

Unsupervised Learning on Free-Form Videos: Though there are many methods for unsupervised optical flow learning [238, 343] on videos (either synthetic [71, 141] or structured [57, 96]), there is very few on unsupervised learning over free-form videos: [337] uses an offline tracker to provide signal to guide feature learning; [345, 241, 82] learn to verify whether frames come with the correct order, and transfer the feature to action classification; [261] learns for region segmentation by considering the moving pattern of rigid objects; [333] learns for video colorization and applies to object tracking; [340] learns dense correspondence by reconstructing mid-level features and applies the model to visual tracking as well.

Filter Flow [293] is a powerful framework which models a wide range of low-level vision problems as estimating spatially varying linear filters, including optical flow [282, 239, 369], deconvolution [198, 264, 119], stereo [290, 237], blur removal [118], etc. However, as it requires an optimization-based solver, it is very expensive to optimize, demanding several hours to compute filters for a pair of medium-size images [293, 276]. Therefore, learning to predict per-pixel kernels attracts more attention, as demonstrated by its applications such as stereo [148], frame interpolation [251], burst denoising [240] and image reconstruction [164].

▢ 7.2.1 Multigrid PFF

Our multigrid Predictive Filter Flow (mgPFF) is rooted in the Filter Flow framework [293], which models the image transformations $\mathbf{I}_B \rightarrow \mathbf{I}_A$ as a linear mapping where each pixel in $\mathbf{I}_A$ only depends on the local neighborhood centered at same place in $\mathbf{I}_B$. Finding such a flow of per-pixel filter can be framed as solving a constrained linear system:

$$\mathbf{I}_A = \mathbf{T}_{B\rightarrow A} \cdot \mathbf{I}_B, \quad \mathbf{T}_{B\rightarrow A} \in \mathbf{\Gamma}. \tag{7.1}$$

where $\mathbf{T}_{B\rightarrow A}$ is a matrix whose rows act separately on a vectorized version of the source image $\mathbf{I}_B$. $\mathbf{T}_{B\rightarrow A}\in\mathbf{\Gamma}$ serves as a placeholder for the entire set of additional constraints on the operator which enables a unique solution that satisfies our expectations for particular problems of interest. For example, standard convolution corresponds to $\mathbf{T}_{B\rightarrow A}$ being a circulant matrix whose rows are cyclic permutations of a single set of filter weights which are typically constrained to have compact localized non-zero support. For a theoretical perspective, Filter Flow expressed in Eq. 7.1 is simple and elegant, but directly solving it is intractable for image sizes we typically encounter in practice, particularly when the filters are big and allowed to vary spatially.

▢ 7.2.2 Predictive Filter Flow (PFF) on Video

Instead of optimizing over $\mathbf{T}$, some recent work coincidentally proposes the predictive version of Filter Flow (PFF) that learns a function $f_{\mathbf{w}}(\cdot)$ parameterized by $\mathbf{w}$ to generate the transformation $\mathbf{T}$ over input image $\mathbf{I}_B$ [148, 164]:

$$\mathbf{I}_A \approx \mathbf{T}_{B\rightarrow A} \cdot \mathbf{I}_B, \quad \mathbf{T}_{B\rightarrow A} \equiv f_{\mathbf{w}}(\mathbf{I}_B). \tag{7.2}$$

The function $f_{\mathbf{w}}(\cdot)$ is learned with a CNN model under the assumption that $(\mathbf{I}_A, \mathbf{I}_B)$ are drawn from some fixed joint distribution. Therefore, given sampled image pairs, $\{(\mathbf{I}_A^i, \mathbf{I}_B^i)\}$, where $i = 1, \dots, N$, we can learn parameters $\mathbf{w}$ that minimize the difference between a recovered image $\hat{\mathbf{I}}_A$ and the real one $\mathbf{I}_A$ measured by some loss ℓ.

In this work, to tailor the PFF idea to unsupervised learning on videos, under the same assumption that $(\mathbf{I}_A, \mathbf{I}_B)$ are drawn from some fixed joint distribution, we can have the predictable transform $\mathbf{T}_{B\to A} \equiv f_{\mathbf{w}}(\mathbf{I}_B, \mathbf{I}_A)$, parametrized by $\mathbf{w}$. To learn the function $f_{\mathbf{w}}(\cdot)$, we use the Charbonnier function [30] to measure the pixel-level reconstruction error averaged spatially over the image, defined as $\phi(\mathbf{s}) = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} \sqrt{s_i^2 + 0.001^2}$, and minimize the following objective function over parameters $\mathbf{w}$ through $T_{B\to A}$:

$$\ell_{rec}(\mathbf{I}_B, \mathbf{I}_A) = \phi(\mathbf{I}_A - \mathbf{T}_{B\to A} \cdot \mathbf{I}_B). \tag{7.3}$$

Note that the above loss can also take image pairs in a reverse order, *i.e.* $\phi(\mathbf{I}_B - \mathbf{T}_{A\to B} \cdot \mathbf{I}_A)$, where $\mathbf{T}_{A\to B}$ is generated over the concatenated features in the reversed order by our design, as demonstrated in Fig. 7.1.

When exploiting the locality constraints (similar to convolution), we implement the operation $\mathbf{T}_{B\to A} \cdot \mathbf{I}_B$ with the "im2col" function which vectorizes the local neighborhood patch centered at each pixel and computes its inner product with the corresponding predicted filter. Note that "im2col" and the follow-up inner product are highly optimized for available hardware architectures in modern deep learning libraries, *exactly the same* as convolution operation. Therefore our model is quite efficient in computation.

7.2.3 Predictive Filter Flow (PFF) vs. Optical Flow

To understand the merits of Predictive Filter Flow compared to standard optical flow with bilinear interpolation, we carry out a diagnostic comparison which evaluates reconstruction

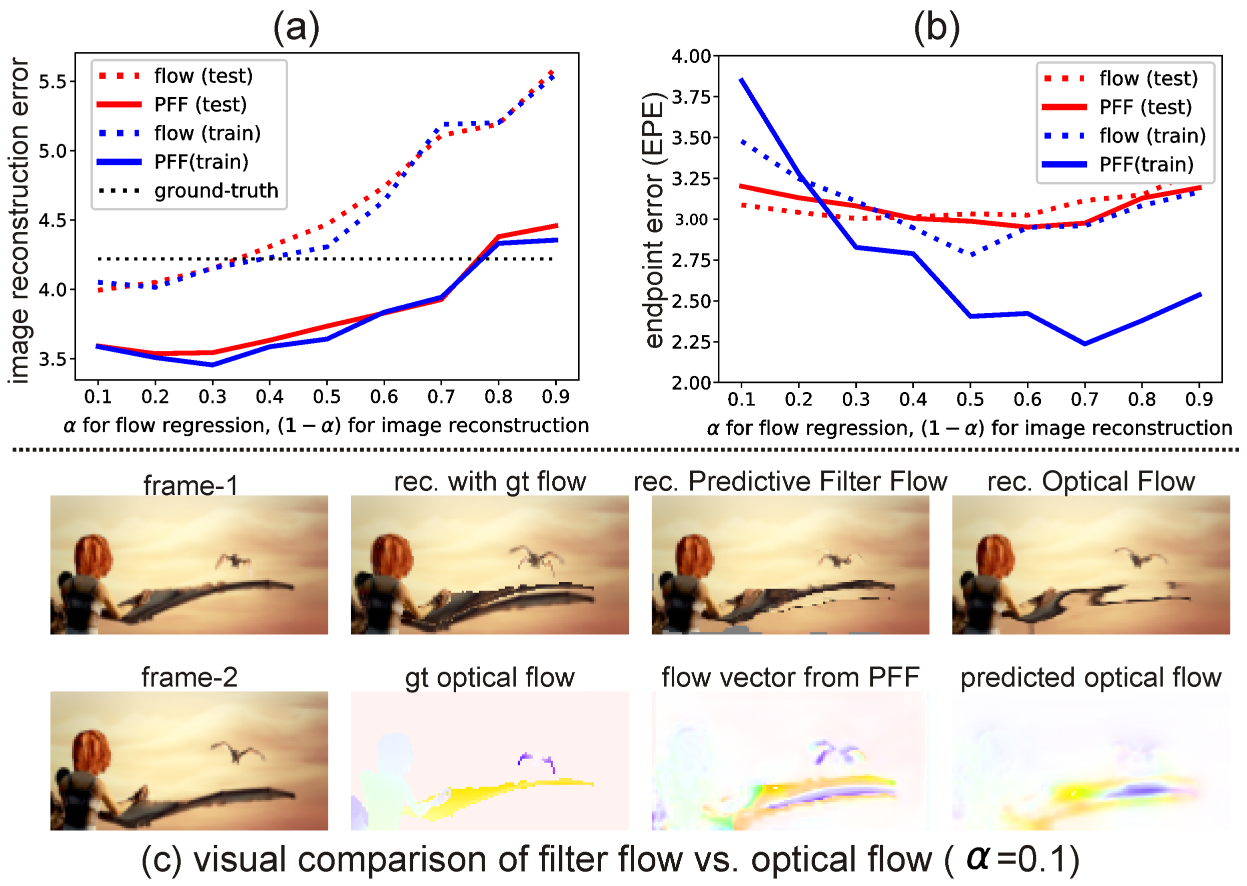


Figure 7.2: **Comparison between Predictive Filter Flow (PFF) and optical flow**, both of which are trained over the Sintel movie and the optical flow dataset [33]. We evaluate training with a mix of supervised and unsupervised data controlled by a hyperparameter α where α=1 is fully-supervised optical flow regression. We note >96% frames are without flow annotations, while only <4% have ground-truth flows. By varying α, we plot the image reconstruction error and endpoint error (EPE) in (a) and (b), respectively; and visualize the flows and reconstructed frames in (c). See details in Section 7.2.3.

and motion estimates at varying levels of supervision. For fair comparison, we choose the same backbone architecture (a modified ResNet18 [115] as detailed in Section 7.2.6) where the top layer outputs either per-pixel filters (PFF) or per-pixel offsets (OF) respectively. We train both models with the self-sueprvised photometric loss $\phi(\cdot)$ for frame reconstruction, without any further regularization terms (*e.g.* , smoothness and occlusion handling). We use the optical flow dataset [33] which provides 1,064 frames with ground-truth optical flow annotations, and split them into our training (574 frames) and testing (490 frames) sets. We also extract all frames in the the Sintel movie, exclude the validation frames, and treat them as unlabeled data (∼17,712 frames). We resize all frames into 64×128, and set the filter size in PFF as 25×25 with a single scale.

We weigh the relative contribution of the supervised (flow regression) and unsupervised (frame reconstruction) loss terms with weights α and 1-α repectively. In Fig. 7.2 we plot the image reconstruction error and endpoint error (EPE) as a function of α. We also depict the flows generated by PFF and the optical flow model (α=0.5) in Fig. 7.2 (c), along with the reconstructed frames compared to using the ground-truth flow.

We observe that: (1) PFF has significantly lower reconstruction error compared to OF across all settings of α. (2) Using large amount of unlabeled video frames improves flow estimates (e.g., at α=0.5) compared to only using supervised training data (at α=1). (3) While PFF does not perform as well as flow regression in flow learning with small α, it begins to outperform optical flow learning when α increases. (4) Visualizing the flows generated by PFF and optical flow, we see the fixed kernel size automatically regularizes the flow scale, thus producing sharper boundaries of the flows. These observations demonstrate the potential of PFF in flow learning and the benefit of frame reconstruction loss to leverage unsupervised data which will be further highlighted through experiments in Section 7.3.

■ 7.2.4 Multigrid PFF

While the PFF described above is elegant, effective and simple for unsupervised learning over videos, it faces a substantial challenge that, to capture large displacement, it must predict big filters to provide large spatial support. To address this problem, we take inspiration from multigrid which seeks to solve high-dimensional systems of equations using hierarchical, multiscale discretizations of linear operators [320, 107], to produce a coarse-to-fine series of smaller, more easily solved problems.

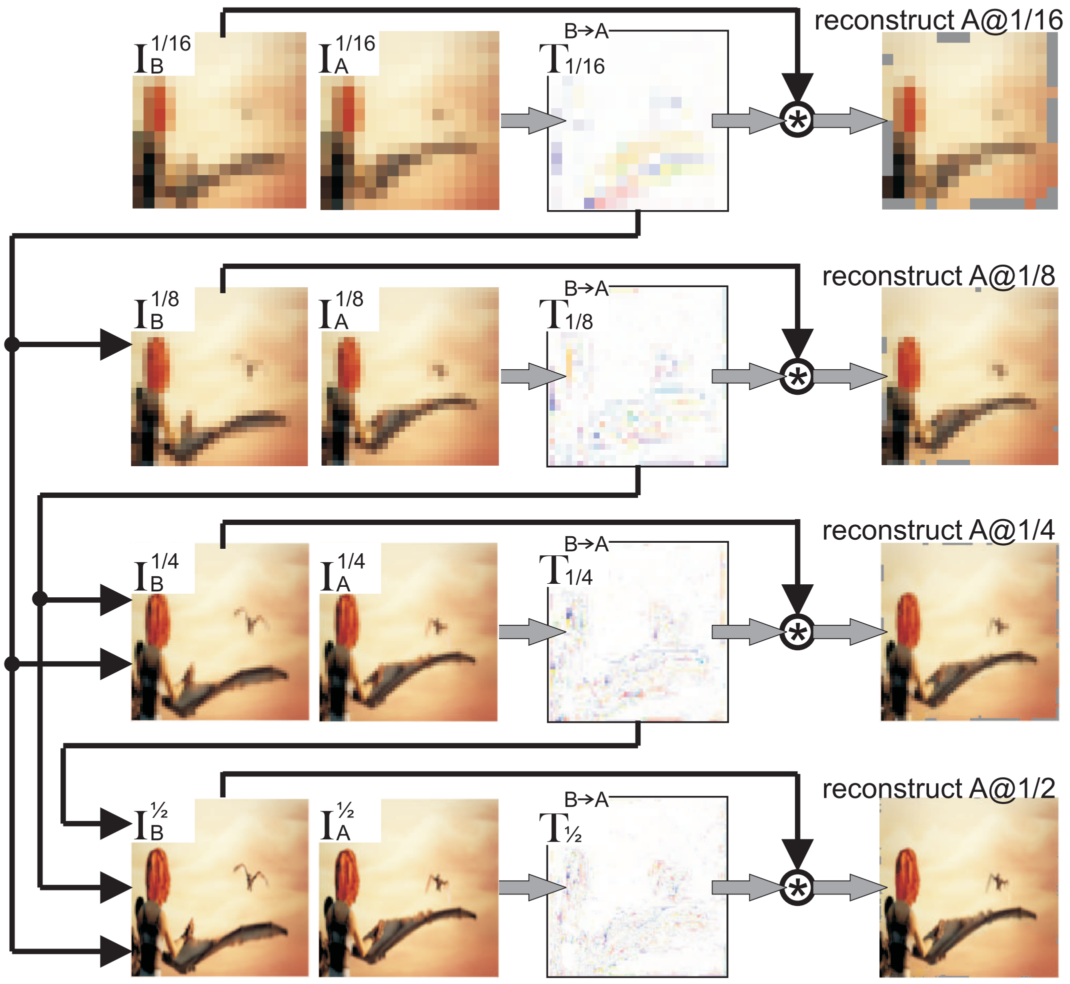


Figure 7.3: Illustration of how multigrid Predictive Filter Flow (mgPFF) performs progressively by warping images from one to the other at multiple resolution scales from coarse to fine.

To explain this, mathematically, suppose we have filter flow $\mathbf{T}$ in original resolution that maps from $\mathbf{X}$ to $\mathbf{Y}$, *i.e.* $\mathbf{Y} = \mathbf{T} \cdot \mathbf{X}$. Then if resizing $\mathbf{X}$ and $\mathbf{Y}$ by half, we have

$$\mathbf{D}_{\frac{1}{2}}\mathbf{Y} = \mathbf{D}_{\frac{1}{2}}\mathbf{T} \cdot \mathbf{X} \approx (\mathbf{D}_{\frac{1}{2}}\mathbf{T}) \cdot (\mathbf{U}_{2\times}\mathbf{D}_{\frac{1}{2}}\mathbf{X}), \tag{7.4}$$

where the upsampling $\mathbf{U}_{2\times}$ and downsampling $\mathbf{D}_{\frac{1}{2}}$ operators are approximately inverse to each other. Then we write a reduced system:

$$\mathbf{Y}_{\frac{1}{2}} \approx (\mathbf{D}_{\frac{1}{2}}\mathbf{T}\mathbf{U}_{2\times}) \cdot (\mathbf{D}_{\frac{1}{2}}\mathbf{X}) = \mathbf{T}_{\frac{1}{2}}\mathbf{X}_{\frac{1}{2}} \tag{7.5}$$

The above derivation implies we can solve a smaller system for $\mathbf{T}_{\frac{1}{2}}$ on the input $\mathbf{X}_{\frac{1}{2}}$, *e.g.* , an image with half the resolution and then upsample $\mathbf{T}_{\frac{1}{2}}$ to get an approximate solution to the original problem.

In practice, to avoid assembling the full resolution $\mathbf{T}$, we always represent it as a composition of residual transformations at each scale. $\mathbf{T} = \mathbf{T}_1 \cdot \mathbf{U}_{2\times} \cdot \mathbf{T}_{\frac{1}{2}} \ldots \mathbf{U}_{2\times} \cdot \mathbf{T}_{\frac{1}{2^L}}$, where $\mathbf{T}_{\frac{1}{2^l}}$ is estimated filter flow over frames at resolution scale $1/2^{l-1}$. In our work, we set L=5. Each individual transformation has a fixed filter support (sparse). By construction, the effective filter "sizes" grow spatially larger as it goes up in the pyramid but the same filter weight is simply applied to larger area (we use nearest-neighbor upsampling). Then the total number of filter coefficients to be predicted for the pyramid would be just 4/3 more than just the finest level (ref. geometric series $4/3 = 1 + \frac{1}{2^2} + \frac{1}{4^2} + \frac{1}{8^2} + \ldots$).

Concretely, suppose we need the kernel size as 80×80 to capture large displacement, we can work on coarse scale of 8× smaller input region with kernel size 11×11, this will reflect on the original image of receptive field as large as 88×88. But merely working on such coarse scale introduces checkerboard effect if we resize the filters 8× larger. Therefore, we let the model progressively generate a series of 11×11 filters at smaller scales of [8×,4×,2×,1×], as demonstrated by Fig. 7.3. Finally, we can accumulate all the generated filter flows towards the single map, which can be a long-range flow (studied in Section 7.3.3). We train our system with the same model at all these scales. We have also trained scale-specific models, but we do not observe any obvious improvements in our experiments. We conjecture that in diverse, free-form videos there is substantial self-similarity in the (residual, low-level) flow across scales.

We note that coarse-to-fine estimation of residual motion is a classic approach to optical flow estimation (*e.g.* , [86]). It has also been used to handle temporal aliasing [302] and acted as a technique for imposing a prior smoothness constraint [315]. Framing flow as a linear operator draws a close connection to multigrid methods in numerical analysis [320, 107]. However, in literature there is primarily focused on solving for $\mathbf{X}$ where the residuals are additive, rather than $\mathbf{T}$ where the residuals are naturally multiplicative as derived in Eq. 7.5.

7.2.5 Constraints, Regularization and Loss

Training with the photometric loss alone produces almost perfect reconstruction performance. But to utilize mgPFF on video segmentation and tracking, we should regularize training to restrain the reconstruction power and encourage unimodal filter (for offset flow) with further constraints, as described as below.

Non-negativity and Sum-to-One. With the mgPFF framework, it is straightforward to impose the non-negativity and sum-to-one constraints by using the softmax layer to output the per-pixel filters, as softmax operation on the kernels naturally provides a transformation on the weights into the range of [0,1], and sum-to-one constraint mimics the brightness constancy assumption [128, 14].

Warping with Flow Vector. To encourage the estimated filter kernels to behave like optical flow (*i.e.* a translated delta function) we define a projection of the filter weights on to the best approximate flow vector by treating the (positive) weights as a distribution and computing an expectation. Given a filter flow $\mathbf{T}$ we define the nearest optical flow as

$$\mathbf{F}(\mathbf{T}) \equiv \begin{bmatrix} v_x(i,j) \\ v_y(i,j) \end{bmatrix} = \sum_{x,y} T_{ij,xy} \begin{bmatrix} x-i \\ y-j \end{bmatrix} \tag{7.6}$$

As discussed in Section 7.1 and 7.2.3, directly learning to predict $\mathbf{F}$ is difficult but when keeping $\mathbf{T}$ as an intermediate representation, learning becomes much easier. To force the predicted $\mathbf{T}$ towards a unimodal offset, we add a loss term based on the optical flow $\mathbf{F}$ with grid sampling layer just as done in literature of unsupervised optical flow learning [281, 343, 141]. We denote the loss term $\ell_{flow}(\mathbf{I}_B, \mathbf{I}_A)$, meaning the reconstruction loss computed by warping with optical flow $\mathbf{F}(\mathbf{T}_{B\rightarrow A})$ from $\mathbf{I}_B$ to $\mathbf{I}_A$.

Forward-Backward Flow Consistency. As we know, there are many solutions to the reconstruction problem. For more robust learning, we adopt a forward-backward consistency constraint as below:

$$\ell_{fb}(\mathbf{f}, \mathbf{b}) \equiv \frac{1}{|\mathcal{I}|} \sum_{\mathbf{i} \in \mathcal{I}} \phi(\mathbf{p}_i - \mathbf{b}(\mathbf{f}(\mathbf{p}_i))) \tag{7.7}$$

where forward and backward flow are $\mathbf{f} = \mathbf{F}(\mathbf{T}_{B\rightarrow A})$ and $\mathbf{b} = \mathbf{F}(\mathbf{T}_{A\rightarrow B})$, and $\mathbf{p}_i \equiv [x_i, y_i]^T$ is a spatial coordinate at i. Note that this constraint helps address the chicken-and-egg problem related to optical flow and occlusion [134, 238]. But we do not use it for handling occlusion with additional thresholding mechanisms. We find it crucial to train the mgPFF model with this constraint when applying the model later for video segmentation; otherwise mask pixels would diffuse to the background easily.

Smoothness constraint can be imposed simply through penalizing the norm of the flow field gradient, *i.e.* $\ell_{sm} \equiv \|\nabla \mathbf{F}(\mathbf{T})\|_1$. The smoothness term helps avoid abrupt transitions on flow field, especially at coarse scales where very few big flows are expected.

Our overall loss for training mgPFF model minimizes the following combination of the terms across multiple scales $l = 1, \ldots, L$:

$$\begin{aligned} \min_{\mathbf{w}} \sum_{l=1}^{L} & \ell_{rec}(\mathbf{I}_B^l, \mathbf{I}_A^l) + \lambda_{flow} \cdot \ell_{flow}(\mathbf{I}_B^l, \mathbf{I}_A^l) \\ & + \lambda_{fb} \cdot \ell_{fb}(\mathbf{f}^l, \mathbf{b}^l) + \lambda_{sm} \cdot \ell_{sm}(\mathbf{f}^l) \end{aligned} \tag{7.8}$$

We set the weights λ_{flow}=λ_{fb}=λ_{sm}=1, as we find the training quite stable over large range of choices on these weights. Note that we only write the loss containing flow from B to A; but in practice we also include those from A to B.

7.2.6 Architecture and Implementation

The mgPFF framework is largely agnostic to the architectures design. In this chapter, we modify the ResNet18 [115] by removing res$_4$ and res$_5$ (the top 9 residual blocks, see supplementary material) and reducing the unique channel size from $[64, 128, 256, 512]$ to $[32, 64, 128, 196]$. We also add in bilateral connection and upsampling layers to make it a U-shape [285], whose output is at the original resolution. Furthermore, we build another shallow stream but in full-resolution manner with batch normalization [137] between a convolution layer and ReLU layer [247] that learns to take care of aliasing effect caused by (un-)pooling layers in the first steam. The mgPFF model is very compact that the overall model is only **4.6MB**; it also performs fast that the wall-clock time for processing a pair of 256×256 frames is 0.1 seconds. Note that the two-stream architecture is popular in multiple domain learning [303], but we note that such design on a single domain was first used in [269] which is more computationally expensive that the two streams interact along the whole information flow, while ours is cheaper that they only interact at the top layer. We note that our mgPFF is also different from FlowNetS and FlowNetC [71] in that, 1) unlike FlowNetS, ours produces pixel embeddings which could be transferred to downstream tasks (though we do not explore this under the scope of this thesis); 2) unlike FlowNetC, ours does not have the expensive correlation layer for cost volume.

7.3 Experiments

We carry out extensive experiments to verify the following:

▨ 7.3.1 Experimental Setup

Dataset. In fact, mgPFF can be trained on any videos owing to its low-level vision nature. Without introducing any new video datasets, we combine three diverse video datasets available to public: the whole Sintel movie [199], DAVIS2017 [271] and JHMDB [146]. Therefore, we construct a training set with a total number of $\sim 6\times 10^4$ video frames resized to 256×256-pixel resolution, among which 17,712 frames are extracted from the Sintel movie. Note that our compared state-of-the-art methods train over orders magnitude larger dataset as explained later. For evaluation on tasks of video object segmentation/tracking and long-range frame reconstruction, we use the DAVIS2017 validation set [271], as done in [333, 340]. In principle, we could combine more videos to train our mgPFF model; however we observe that more data does not necessarily improve mgPFF. We conjecture the reason is due to the low-level nature of our mgPFF approach. This is also noted in [340], which further reports that fine-tuning on the DAVIS2017 validation set barely improves performance either.

Training. We use ADAM optimization [156] for training our model from scratch (*i.e.* random initialization), with initial learning 0.0005 and coefficients 0.9 and 0.999 for computing running averages of gradient and its square. We train our model using PyTorch [260] on a single NVIDIA TITAN X GPU, and terminate after 500K iteration updates[†]. During training, we randomly sample frame pairs (resized to 256×256-pixel resolution) within five consecutive frames. We also augment the training data by randomly flipping and rotating the frame pairs. After training, we directly evaluate the mgPFF model on the DAVIS2017 validation set for video object segmentation/tracking, long-range frame reconstruction and photo editing. We note, as also reported by [340], unsupervised fine-tuning specifically on the validation set does not necessarily improve the performance, due to the low-level nature of our mgPFF model.

[†] The code and models can be found in `https://github.com/aimerykong/predictive-filter-flow`

It is also worth noting that, in training our mgPFF on the Sintel movie, mgPFF offers the ability of detecting video shot/transition [26, 94] purely based on the reconstruction errors between frames. This helps develop a stage-wise training pipeline that we could train mgPFF first on the whole movie, and then simply threshold the reconstruction errors for shot detection and get discrete groups for finer training.

Inference. For video object segmentation/tracking, we propagate the binary mask given at the first frame along the time using the output filter flows by mgPFF. As mgPFF produces per-pixel kernels indicating a probability (owing to the sum-to-one property) on where to move, the propagated mask contains continuous values with range [0,1]. We threshold with 0.8 the propagated masks at each tracking update. This helps prevent foreground mask pixels from diffusing to the background especially at the true boundary. As mask propagation can also accumulate history masks instead of the only one from the previous frame, we set the temporal window size K, meaning we warp towards the target frame using previous K frames. For mgPFF, we find K=3 works the best with an ablation study reported in Table 7.1 as shown later; the compared methods also adopt different window size K for their best performance.

Compared methods include the following:

- **Identity** always copies the mask given at the $1st$ frame.
- **Optical Flow** (FlowNet2 [136]): a state-of-the-art method supervised trained for predicting optical flow. Particularly, for a target frame $\mathbf{I}_t$, we compute the optical flow from $\mathbf{I}_{t-1}$ to $\mathbf{I}_t$ and warp the mask at $\mathbf{I}_{t-1}$ using the flow.
- **SIFT Flow** [218]: an off-the-shelf tool for estimating dense flow, which is used to propagate masks similar to the above Optical Flow method.
- **ColorPointer** [333] is a self-supervised method with the proxy task of video frame

colorization over the Kinetics dataset [151] ($\sim 9 \times 10^7$ frames). The architecture is based on 3D ResNet18. We report the results from [333].

- **CycleTime** [340] is another unsupervised method trained for reconstructing deep features with cost volume between frames. It is trained on VLOG dataset [89] (344-hour recording, $\sim 3.7 \times 10^7$ frames). We report the results from [340] with the window size K=7.

- **DeepCluster** [37] is a self-supervised approach by using a K-means objective to iteratively update targets and learn a mapping from images to targets. It is trained on the ImageNet dataset [63] without using annotations. The trained model with VGG16 [304] adopts the same inference procedure as CycleTime, with window size K=7.

- **Fully-Supervised Methods** serve as reference [364, 35], which are trained specifically on DAVIS2017 dataset from an ImageNet pre-trained model in a supervised way.

■ 7.3.2 Video Object Segmentation and Tracking

We analyze our model on video object segmentation and tracking over the DAVIS 2017 validation set [271], where the initial segmentation mask is given and the task is to predict the segmentation in the rest of the video. This is a very challenging task as the videos contain multiple objects that undergo significant occlusion, deformation and scale change with clutter background, as shown in Fig. 7.4. We use the provided code and report two metrics that score segment overlap and boundary accuracy. The Jacaard index $\mathcal{J}$ is defined as the intersection-over-union of the estimated segmentation and the ground-truth mask, measuring how well the pixels of two masks match [77]. The $\mathcal{J}$ recall measures the fraction of sequences with IoU>0.5. The F-measure denoted by $\mathcal{F}$ considers both contour-based precision and recall that measure the accuracy of the segment contours [235].

Table 7.1: **Tracking Segmentation** on the DAVIS2017 validation set. Methods marked with ^{1st} always use the first frame and its mask (provided) for tracking. The number in bracket indicates the estimated number of frames used for training.

Method	Supervision	$\mathcal{J}$ (segments)		$\mathcal{F}$ (boundaries)	
		mean↑	recall↑	mean↑	recall↑
OSVOS [35]	ImageNet, DAVIS	55.1	60.2	62.1	71.3
MaskTrack [152]	ImageNet, DAVIS	51.2	59.7	57.3	65.5
OSVOS-B [35]	ImageNet	18.5	15.9	30.0	20.0
MaskTrack-B [152]	ImageNet	35.3	37.8	36.4	36.0
OSVOS-M [364]	ImageNet	36.4	34.8	39.5	35.3
Identity	None	22.1	15.9	23.6	11.7
SIFTflow [218]	None	13.0	7.9	15.1	5.5
SIFTflow^{1st} [218]	None	33.0	–	35.0	–
FlowNet2 [136]	Synthetic	16.7	9.5	19.7	7.6
FlowNet2^{1st} [136]	Synthetic	26.7	–	25.2	–
DeepCluster^{1st} [37]	Self (1.3×10^6)	37.5	–	33.2	–
ColorPointer [333]	Self (9.0×10^7)	34.6	34.1	32.7	26.8
CycleTime^{1st} [340]	Self (3.7×10^7)	40.1	–	38.3	–
mgPFF (1st only)	Self (6.0×10^4)	31.6	29.5	36.2	30.8
mgPFF (K=1)		38.9	38.5	41.1	38.6
mgPFF^{1st} (K=1)		41.9	41.4	45.2	43.9
mgPFF^{1st} (K=3)		**42.2**	**41.8**	**46.9**	**44.4**

We compare our mgPFF with other unsupervised methods as well as some supervised ones [364, 35] in Table 7.1. As in literature there are methods always using the given mask at the first frame to aid tracking, we also follow this practice with mgPFF to report the performance. But before doing so, we ablate how much gain we can get from using only the given mask for the tracking. To this end, we setup the mgPFF by always propagating the given mask for tracking, as noted by mgPFF (1st only) in Table 7.1. Surprisingly, this simple setup works quite well, even surpassing almost all the compared flow-based methods, *e.g.* , SIFTflow^{1st} and FlowNet2^{1st}, both of which use the given mask at the first frame in addition to K=7 predicted masks in the past. This suggests the mgPFF is able to capture long-range flow even though we did not train our model with frames across large intervals. We explicitly study this long-range flow in Section 7.3.3 quantitatively.

When tracking with the *only one* previous propagated mask (K=1), our mgPFF outperforms

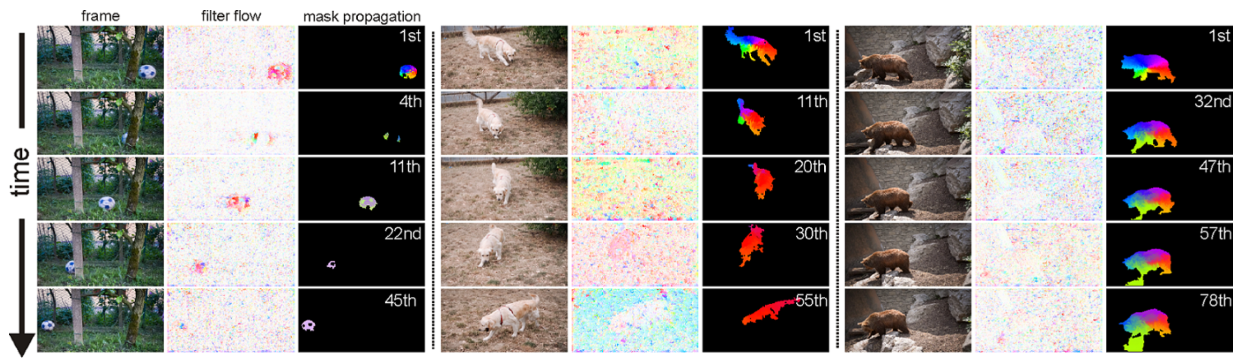


Figure 7.4: Visualization of unsupervised learning for video segmentation on DAVIS2017: *soccerball*, *dog* and *bear*. We show the tracking results with temporal window size K=3 for *soccerball* (otherwise it loses track due to heavy occlusion) and K=1 for others. Note that in *soccerball*, there are heavy occlusions but our mgPFF model can still track the ball. In *dog*, we can see how each pixel moves along with the dog: when the dog turns from right side to left side, the colors from the neck are propagated for tracking. In *bear*, we see an error where the shadow appearing in the bottom of the image is explained by flow from the bear's leg.

all the other unsupervised methods, except for CycleTime on the $\mathcal{J}$ measure, which is explicitly trained at patch level thus captures better object segment. When additionally using the mask given at the first frame for tracking in subsequent frames, mgPFF^{1st}(K=1) outperforms notably all other unsupervised methods, and our mgPFF^{1st}(K=3) achieves the best performance. In particular, in terms of the boundary measure, we can see mgPFF performs significantly better than the other unsupervised methods. This demonstrates the benefit of propagating masks with fine-grained pixel-level flows instead of flows learned at patch level through mid-level feature activations [333, 340].

Overall, our mgPFF even outperforms several supervised methods, but only worse than the first two supervised models in Table 7.1 which are explicitly trained with DAVIS pixel-level annotations at all training frames. It is also worth noting that our mgPFF model is trained over two orders magnitude less frames than other unsupervised methods, *e.g.* , DeepCluster, ColorPointer and CycleTime. This demonstrates the benefit of the low-vision nature of mgPFF that it does not demand overly large-scale training data.

Table 7.2: **Long-range flow for frame Reconstruction**: We compute the long-range flow vector from the filter flows by mgPFF on two frames $\mathbf{I}_t$ and $\mathbf{I}_{t+m}$, where m=5 or m=10. We use the flow to warp $\mathbf{I}_t$ towards $\mathbf{I}_{t+m}$, and report the averaged pixel difference under the L_1 norm between the warped frame and $\mathbf{I}_{t+m}$.

method/error↓	$m = 5$	$m = 10$
Identity	82.0	97.7
Optical Flow (FlowNet2) [136]	62.4	90.3
CycleTime [340]	60.4	76.4
mgPFF	**7.32**	**8.83**

We note that the most recent work [203] achieves the best reported performance by combining the low-level reconstruction learning (*i.e.* pixel correspondence as done in our mgPFF) and mid-level region correspondence learning as done in CycleTime [340], with additional sophisticated regularization terms. While not aiming at competing with [203], we note our mgPFF not only complements all the mid-level methods (*e.g.* CycleTime and its extension [203]), but also exclusively embraces the benefit from the fine-grained pixel reconstruction, which can be highlighted by long-range frame reconstruction and interactive photo editing as demonstrated in the next Section 7.3.3 and 7.3.4.

In Fig. 7.4, we visualize the tracking results and the predicted filter flow (from previous one frame only). Specifically, we transform the filter flow into the flow vector (Eq. 7.6) for visualization. As mgPFF performs at pixel level, we are able to visualize the tracking through more fine-grained details. We paint on the mask with the color chart from optical flow, and visualize to see how the pixels evolve over time. Interestingly, we can see how tracking is accomplished in face of heavy occlusion, big deformation and similar background (see the caption of Fig. 7.4).

7.3.3 Long-Range Frame Reconstruction

We highlight our mgPFF is particularly good at learning long-range flow through the task of frame reconstruction, as suggested by [340]. To validate this, specifically, given two testing

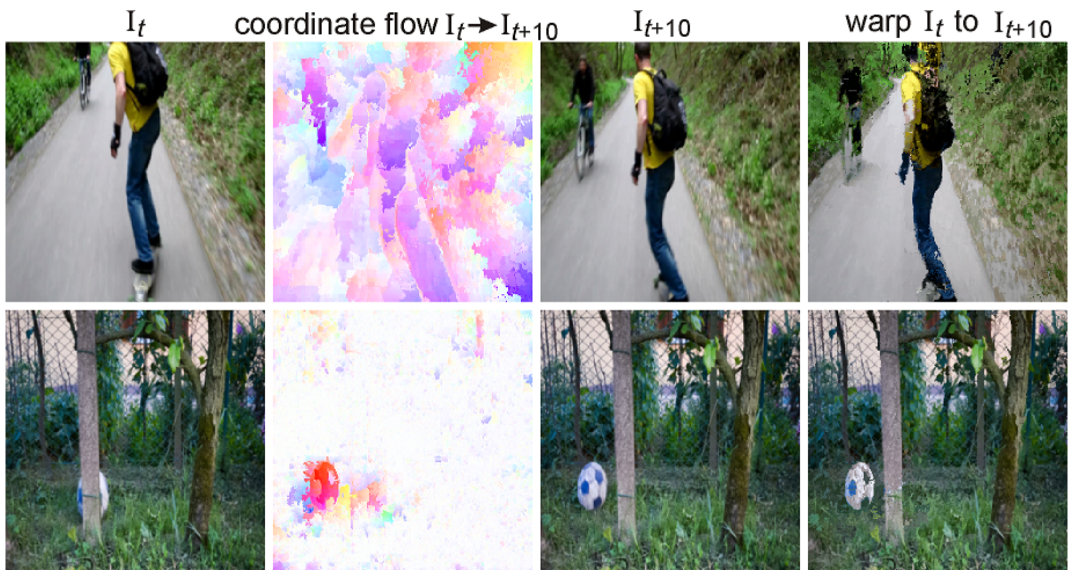


Figure 7.5: **Long-range frame reconstruction** (rightmost column) reflects long-range flow learning that warps $\mathbf{I}_t$ (1st column) with the coordinate flow (2nd column) which is transformed from the predicted multigrid filter flow. The target frames $\mathbf{I}_{t+10}$ are shown in the 3rd column.

frames $\mathbf{I}_t$ and $\mathbf{I}_{t+m}$ distant in time from a video, we predict the filter flow between them, and then transform the filter flow into offset (*i.e.* optical flow) according to Eq. 7.6 to indicate where to copy pixels from $\mathbf{I}_t$. With the flow, we warp frame $\mathbf{I}_t$ to generate a new frame $\hat{\mathbf{I}}_{t+m}$. Therefore, we round the warped image to the integer values, and compare the averaged pixel difference under the L_1 norm between $\mathbf{I}_t$ and $\hat{\mathbf{I}}_{t+m}$ in the original [0,255] scale.

We perform this experiment on DAVIS2017 validation set, and report the performance in Table 7.2, We set the time gap as m=5 or m=10, meaning the pair of testing frames are m frames apart from each other. In both frame gaps, our mgPFF significantly outperforms the compared methods, demonstrating the powerfulness of mgPFF in modeling pixel-level movement, even though our model is trained over frame pairs within 5-frame interval without seeing any frames far away from 5 frames. In Fig. 7.5, we clearly see that mgPFF performs quite well visually on long-range frame reconstruction. In the next section, we demonstrate further its powerfulness in image reconstruction through a novel photo editing application enabled by mgPFF.

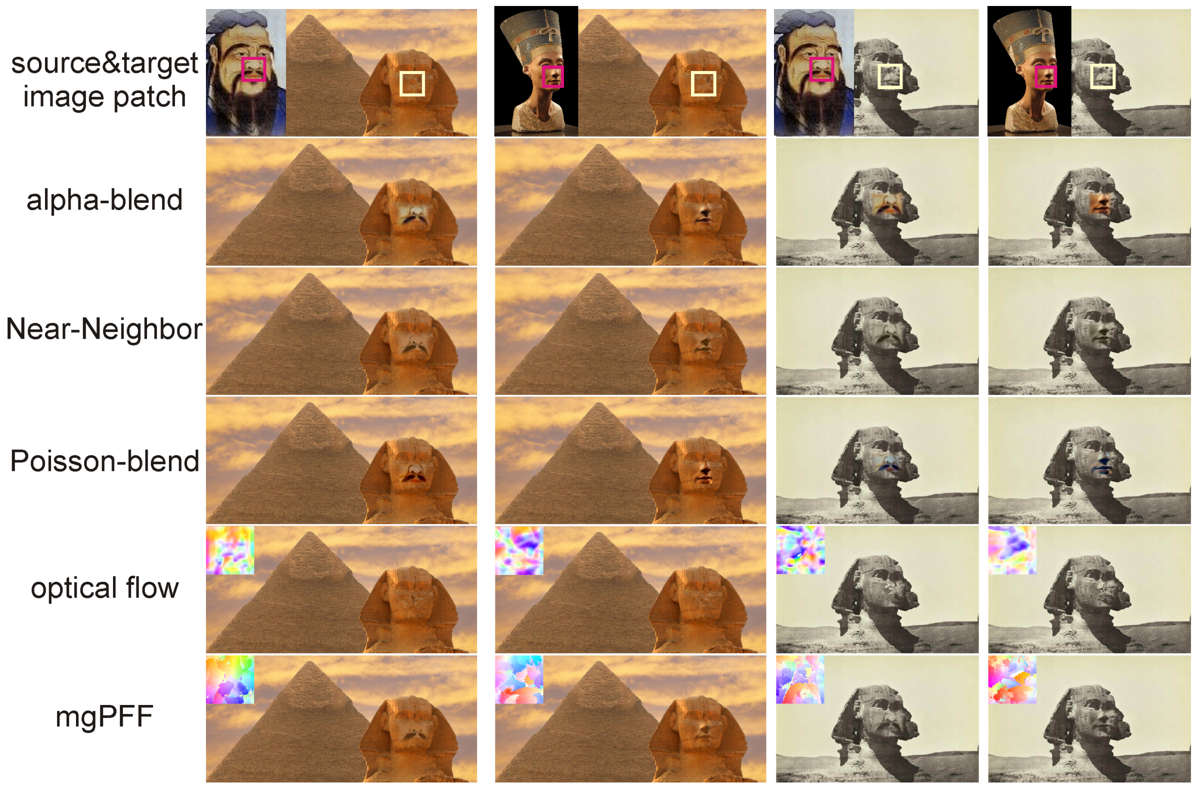


Figure 7.6: **Interactive Photo Editing**. We compare mgPFF with two classic blending algorithms (alpha blending and Poisson blending) and the unsupervised optical flow method. We can see the self-regularization enables our mgPFF preserve both the structure in the source image patch and target image style, allowing for more "realistic" restoration. The upper-left flow maps visualize the optical/filter flows.

▢ 7.3.4 Interactive Photo Editing

We use the same trained mgPFF model in the previous section for a novel interactive photo editing application, which requires blending a source image patch to a target image and preserving both the source texture/structure and the target image style. We demonstrate this application in Fig. 7.6, where we transplant a nose from a reference image to the missing nose part in the Sphinx statue. We compare with alpha blending [273], Poisson blending [263], the nearest neighbor approach that reconstruct pixels of target patch using all the source pixels, and the unsupervised optical flow method as trained in Section 7.2.3.

We can see the mgPFF performs quite well in such an interactive photo editing task, transferring the structure of the source patch while observing the target style. However, the optical flow method fails to make visible edits, and the classic blending methods fail to either adjust the colors or induce artifact colors due to color leakage. It is worth noting that nearest the neighbor approach also reconstructs the target patch quit well. However it is slower than mgPFF as it requires iterating over all the source pixels to reconstruct the target pixel. Moreover nearest-neighbor is too flexible to introduce reasonable diversity; in contrast, mgPFF uses a little region (defined by the per-pixel kernel size) from source patch to reconstruct the target pixel which is the centered of the corresponding target patch. This small yet novel photo editing task benefits from the mgPFF's reconstruction power, while the self-regularization (*i.e.* sum-to-one property) guarantees consistent color, tones and brightness in the reconstruction, fulfilling satisfactory blending with fast computation by a single forward pass of the mgPFF model (<0.1sec for blending).

7.4 Conclusion

We propose a simple, compact framework for unsupervised learning on free-form videos, named multigrid Predictive Filter Flow (mgPFF). Through experiments, we show mgPFF outperforms other state-of-the-art methods notably in video object segmentation under the unsupervised learning setup; it also exhibits great power in long-range flow learning and enables a novel photo editing application owing to its reconstruction power. We expect future research based on mgPFF for, *e.g.* , flow-based applications, pixel embedding transfer to downstream tasks, higher-level learning for video understanding, etc.

Chapter 8

Concluding Remarks

To conclude this thesis, we start with a high-level summary of the major themes and contributions of the thesis. As we realize a plethora of open questions which are raised by, or simply indicated by, our work, we include more discussions and the literature, and suggest some directions or topics for future research.

8.1 Summary and Contributions

In previous chapters, we review the literature that advances pixel-level prediction, understanding the important ingredients involved in the learning based methods, specifically the convolutional neural networks (CNN), which achieve state-of-the-art over various per-pixel vision tasks. Then we show the CNN framework serves as an elegant model for different per-pixel prediction tasks. We propose several loss functions in training, and show performance gains over the state-of-the-art.

Since the CNN model directly produces the output in the target space, making itself vulnerable to complex tasks, such as segmenting C. elegans worms intersection each other, we propose the pixel embedding methodology that trains CNN models to project all pixels into an embedding space, in which pixels become easier to be grouped according to the defined metric. We apply the pixel embedding idea to object instance segmentation, and show great success that it outperforms the state-of-the-art remarkably. Furthermore, we show the metric learning among pixel pairs enables richer regularization, improving over the direct-prediction methodology.

Noticing all the aforementioned models are trained in a fully supervised way, i.e., requiring (manual) annotations for supervision, we study how to relax the annotation level. To this end, we propose the Predictive Filter Flow (PFF) framework that trains over simulated data. Specifically, we train for image reconstruction tasks, including single image super-resolution, non-uniform blur removal and JPEG compression artifacts reduction. We simulate degraded images through simulation, or off-the-shelf algorithms, so that we form training image pairs. Besides, PFF learns to output per-pixel kernels which are used to warp input patches centered at each pixel towards the output pixel. In this sense, PFF has better interpretability as the decision making process is transparent to users, endowing controllable property to the system. Just because of the nature of PFF, it also regularizes the output by the input itself, through warping the input to the output without a blackbox-like transformation. Over the mentioned image reconstruction tasks, we show PFF achieve better performance than the state-of-the-art, enjoying better interpretable and controllable properties.

We further extend PFF to the multigrid version (mgPFF) to process high-resolution images and train over unconstrained videos. As videos has temporal coherency which acts as supervision, we train mgPFF in a fully self-supervised way, taking as input the two frames nearby temporally, learning to output per-pixel kernels used to warping one frame towards the other. We show mgPFF not only achieves almost perfect reconstruction between frames, but also learns to capture pixel correspondence, which can be used for visual tracking and enables one to track every pixel to understand how the pixels evolves over time. With its the reconstruction power and self-regularization, we apply it to transplanting image patches to a target image, and demonstrate an innovative and unique photo editing experience.

▢ 8.2 Suggestions and Future Research

From the proposed depth-aware gating module that improves semantic segmentation, we see coupling multiple tasks in a smart way may help learning for a particular task. While it is a well-holden belief that multi-task learning may boost all tasks, recent study finds that tasks have their own hierarchies, in terms of learning for better performance [66, 370, 308]. Moreover, some tasks are tightly coupled with physical relation or mathematical dependence. For example, surface normal is the derivative of depth, while curvature is the derivative of surface normal and second order derivative of depth; instance segments exactly carry the object boundary, and are over-segments for semantic segmentation. Therefore, the study in task-wise relationship and how to use such relations for better learning will be a rewarding research direction.

In the Pixel-wise Attention Module (PAG) study, we show how we can achieve dynamic computation at pixel level that achieves better performance than other design choices for pixel-level prediction, while with reduced computation budget measured by FLOPS. However, we find the computation FLOPS is not tightly coupled with wall-clock time. In other word, fewer computation FLOPS does not necessarily indicate less computation time. This is largely due to the highly optimized hardware for specific computations, like CUDA for matrix multiplication. Recent literature has witnessed different attempts towards real-world pixel-level predictions, like teacher-student training strategy that trains compact model to mimic a very deep model [244], multigrid strategy for quickly capture long-range information at coarse resolution [384, 277], flow-based idea for fast propagating features used for prediction [391], and so on. We suggest further investigation in this direction for both real-time inference and fewer computation FLOPS. This is not only of great interest to industry requiring real-time calculation, but also significant in computation with green energy [292], useful for low-power deployment and scientific understanding of information flow in deep learning.

The pixel embedding idea proposed in Chapter 5 show very promising results in various pixel-level prediction/grouping problems. However, through the mentioned limitations of the method, improving the cost volume in terms of computation and memory consumption is prerequisite to broad deployment. Luckily, we see some recent literature publishing some attempts on this, e.g., exploiting locality instead of computing holistic cost volume [43, 197]. Such an improvement enables grouping pixels or voxels into meaningful parts, like neurons in 3D scanning and cells in 2D image. Therefore, we suggest more endeavor invested in this direction, learning to embedding pixels into meaningful groups and even better into hierarchical clusters. At the same time, we note such learning for pixel embedding, as well as supervised methods presented earlier, requires extensively manual annotations at pixel level. So we expect more research in unsupervised or self-supervised learning with specific applications, especially in inter-disciplinary applications.

From the PFF method which has interpretable and controllable advantage over blackbox mechanism, we expect its more utilizations in broad fields, such as brain imaging registration [61], the simple image enhancement in biological images and medical images [53, 347, 335, 390]. The barrier is still in how to obtain training data, as we do not have clear images as target for training. There are some recent work in collecting data through two different microscopes: one is low-power for low-resolution imaging, the other is energy costly but for high-resolution imaging. It shows that the image enhancement model can be trained over with these data pairs [53, 347, 335, 390]. However, we notice that collecting such training data is also expensive in term of operating complex machines and register the resulting images from different microscopy techniques. Motivated by this, we would recommend using generative model and a discriminator loss to reconstruct images, like the preliminary result we have shown in Chapter 6. Therefore, we expect more fruitful research in this direction in the near future.

The multigrid PFF framework (mgPFF) shows exciting and promising results in self-supervised learning on videos, and broad applications in visual tracking and photo editing. However we note the mgPFF essentially only learns some low-level features without high-level understanding of the video frames. Therefore, we recommending more investigation in this direction with a few suggestions. First, multi-modal signals can be exploited along with mgPFF learning. Such signals can be audio, captions, LiDAR, GPS, etc. Second, using some features as input along with RGB frames may also help regularize training and capture higher level information. Such features can be from a pre-trained model, as a reflection of knowledge distillation [117]. Third, it may also be valuable to train mgPFF on domain specific data for particular application, like neuron images for neuron tracing, tissue scanning for tracking cells, etc.

Beyond purely exploiting temporal consistency knowledge in learning over unconstrained videos (like our mgPFF), we would also expect learning methods with geometry-aware constraints. For example, one can exploit the projective geometry rule to learn both camera pose and single-image depth [387]. Such camera pose estimation can be used for SLAM, augmenting pixel-level prediction, and fast rendering predictions at previous frame(s) for real-time processing in videos. We also expect investigation in how to make use of such self-supervised trained models in other tasks, through fine-tuning or other formats to regularize other downstream tasks learning.

Lastly, we have not covered another important venue, leveraging synthetic data with the real-world data in the domain of interest for pixel-level prediction. As one can render unlimited synthetic data, it is no reason not to exploit them, optionally combined with small amount of annotated real-world data, if available. Research in this amounts to domain adaptation, which aims at closing domain gap. We would like to mention that, the current practice that using GAN method (e.g., CyCADA [121]) is limited as it only changes color and tones between images of two domains. It is common that different domains have different stuffs, for example it is not possible to render diaper and clothes in the synthetic indoor data, while these stuffs can be common in real-world scenes. Therefore, we recommend a study in how to translate images between different domains with a domain-aware zero-out strategy, offering more interpretable and sensible mechanism in closing domain gaps caused by scene content instead of the simple low-level differences (say color and tone).

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] W. Abdulla and P. Ferriere. Neural network graphs and training metrics for pytorch and tensorflow. 2018.

[3] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[4] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, pages 5074–5082, 2016.

[5] E. Agustsson and R. Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[6] R. Arandjelovic and A. Zisserman. Objects that sound. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 435–451, 2018.

[7] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011.

[8] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 328–335, 2014.

[9] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese. Joint 2d-3d-semantic data for indoor scene understanding. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[10] A. Arnab and P. H. Torr. Pixelwise instance segmentation with a dynamically instantiated network. *arXiv preprint arXiv:1704.02386*, 2017.

[11] Y. Bahat, N. Efrat, and M. Irani. Non-uniform blind deblurring by reblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3286–3294, 2017.

[12] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. *arXiv preprint arXiv:1611.08303*, 2016.

[13] G. BakIr, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. Vishwanathan. *Predicting structured data.* MIT press, 2007.

[14] S. Bako, T. Vogels, B. McWilliams, M. Meyer, J. Novák, A. Harvill, P. Sen, T. Derose, and F. Rousselle. Kernel-predicting convolutional networks for denoising monte carlo renderings. *ACM Trans. Graph.*, 36(4):97–1, 2017.

[15] P. Baldi and R. Vershynin. The capacity of feedforward neural networks. *Neural Networks*, 116:288–311, 2019.

[16] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research*, 6(Sep):1345–1382, 2005.

[17] A. Bansal, B. Russell, and A. Gupta. Marr revisited: 2d-3d alignment via surface normal prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5965–5974, 2016.

[18] S. Y. Bao, M. Sun, and S. Savarese. Toward coherent object detection and scene layout understanding. *Image and Vision Computing*, 29(9):569–579, 2011.

[19] D. M. Beck and S. Kastner. Top-down and bottom-up mechanisms in biasing competition in the human brain. *Vision research*, 49(10):1154–1165, 2009.

[20] V. Belagiannis and A. Zisserman. Recurrent human pose estimation. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pages 468–475. IEEE, 2017.

[21] A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159, 1995.

[22] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[23] F. Bergholm. Edge focusing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):726–741, 1987.

[24] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *British Machine Vision Conference*, 2012.

[25] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *European Conference on Computer Vision*, pages 561–578. Springer, 2016.

[26] J. S. Boreczky and L. A. Rowe. Comparison of video shot boundary detection techniques. *Journal of Electronic Imaging*, 5(2):122–129, 1996.

[27] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[28] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2559–2566. Citeseer, 2010.

[29] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[30] A. Bruhn and J. Weickert. Towards ultimate motion estimation: Combining highest accuracy with real-time performance. In *10th IEEE International Conference on Computer Vision (ICCV 2005), 17-20 October 2005, Beijing, China*, pages 749–755, 2005.

[31] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 60–65. IEEE, 2005.

[32] P. Burt and E. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on communications*, 31(4):532–540, 1983.

[33] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012.

[34] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, 2015.

[35] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 221–230, 2017.

[36] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[37] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018.

[38] J. Carreira and C. Sminchisescu. Cpmc: Automatic object segmentation using constrained parametric min-cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1312–1328, 2012.

[39] M. A. Carreira-Perpinán. Generalised blurring mean-shift algorithms for nonparametric clustering. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[40] M. A. Carreira-Perpinán and Y. Idelbayev. Model compression as constrained optimization, with application to neural nets. *Part III: Pruning. arXiv*, 2017.

[41] L. Cavigelli, P. Hager, and L. Benini. Cas-cnn: A deep convolutional neural network for image compression artifact suppression. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 752–759. IEEE, 2017.

[42] C.-C. Chen et al. Fast boundary detection: A generalization and a new algorithm. *IEEE Transactions on Computers*, 100(10):988–998, 1977.

[43] L. Chen, M. Strauch, and D. Merhof. Instance segmentation of biomedical images with an object-aware embedding learned with local constraints. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 451–459. Springer, 2019.

[44] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[45] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[46] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[47] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3640–3649, 2016.

[48] Y.-T. Chen, X. Liu, and M.-H. Yang. Multi-instance object segmentation with occlusion handling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3470–3478, 2015.

[49] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

[50] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

[51] M. Chiang, S. Hallman, A. Cinquin, N. R. de Mochel, A. Paz, S. Kawauchi, A. L. Calof, K. W. Cho, C. C. Fowlkes, and O. Cinquin. Analysis of in vivo single cell behavior by high throughput, human-in-the-loop segmentation of three-dimensional images. *BMC bioinformatics*, 16(1):397, 2015.

[52] S. Cho, J. Wang, and S. Lee. Handling outliers in non-blind image deconvolution. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 495–502. IEEE, 2011.

[53] E. M. Christiansen, S. J. Yang, D. M. Ando, A. Javaherian, G. Skibinski, S. Lipnick, E. Mount, A. ONeil, K. Shah, A. K. Lee, et al. In silico labeling: predicting fluorescent labels in unlabeled images. *Cell*, 173(3):792–803, 2018.

[54] R. M. Cichy, D. Pantazis, and A. Oliva. Resolving human object recognition in space and time. *Nature neuroscience*, 17(3):455–462, 2014.

[55] D. Comaniciu and P. Meer. Mean shift analysis and applications. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1197–1203. IEEE, 1999.

[56] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(5):603–619, 2002.

[57] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[58] J. Dai, K. He, and J. Sun. Convolutional feature masking for joint object and stuff segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3992–4000, 2015.

[59] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016.

[60] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. 2005.

[61] A. V. Dalca, G. Balakrishnan, J. Guttag, and M. R. Sabuncu. Unsupervised learning of probabilistic diffeomorphic registration for images and surfaces. *arXiv preprint arXiv:1903.03545*, 2019.

[62] B. De Brabandere, D. Neven, and L. Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017.

[63] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE, 2009.

[64] R. Diaz and A. Marathe. Soft labels for ordinal regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4738–4747, 2019.

[65] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.

[66] C. Doersch and A. Zisserman. Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2051–2060, 2017.

[67] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[68] C. Dong, Y. Deng, C. Change Loy, and X. Tang. Compression artifacts reduction by a deep convolutional network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 576–584, 2015.

[69] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2016.

[70] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[71] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[72] T. Dozat. Incorporating nesterov momentum into adam. 2016.

[73] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[74] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015.

[75] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[76] V. A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1):153–158, 1969.

[77] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[78] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Scene parsing with multiscale feature learning, purity trees, and optimal covers. *arXiv preprint arXiv:1202.2160*, 2012.

[79] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy. Semantic instance segmentation via deep metric learning. *arXiv preprint arXiv:1703.10277*, 2017.

[80] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)*, 59(2):167–181, 2004.

[81] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. In *ACM transactions on graphics (TOG)*, volume 25, pages 787–794. ACM, 2006.

[82] B. Fernando, H. Bilen, E. Gavves, and S. Gould. Self-supervised video representation learning with odd-one-out networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3636–3645, 2017.

[83] M. Figurnov, M. D. Collins, Y. Zhu, L. Zhang, J. Huang, D. Vetrov, and R. Salakhutdinov. Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[84] M. Figurnov, A. Ibraimova, D. P. Vetrov, and P. Kohli. Perforatedcnns: Acceleration through elimination of redundant convolutions. In *Advances in Neural Information Processing Systems (NIPS)*, pages 947–955, 2016.

[85] R. Fisher. Dispersion on a sphere. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 217, pages 295–305. The Royal Society, 1953.

[86] D. Fleet and Y. Weiss. Optical flow estimation. In *Handbook of mathematical models in computer vision*, pages 237–257. Springer, 2006.

[87] A. Foi, V. Katkovnik, and K. Egiazarian. Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images. *IEEE Transactions on Image Processing*, 16(5):1395–1411, 2007.

[88] D. F. Fouhey, A. Gupta, and M. Hebert. Data-driven 3d primitives for single image understanding. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3392–3399. IEEE, 2013.

[89] D. F. Fouhey, W.-c. Kuo, A. A. Efros, and J. Malik. From lifestyle vlogs to everyday interactions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4991–5000, 2018.

[90] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018.

[91] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1):32–40, 1975.

[92] R. Gadde, V. Jampani, M. Kiefel, D. Kappler, and P. V. Gehler. Superpixel convolutional networks using bilateral inceptions. *arXiv preprint arXiv:1511.06739*, 2015.

[93] R. Garg, V. K. BG, G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016.

[94] U. Gargi, R. Kasturi, and S. H. Strayer. Performance characterization of video-shot-change detection methods. *IEEE transactions on circuits and systems for video technology*, 10(1):1–13, 2000.

[95] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun. 3d traffic scene understanding from movable platforms. *IEEE transactions on pattern analysis and machine intelligence*, 36(5):1012–1025, 2014.

[96] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[97] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[98] D. Geman, S. Geman, C. Graffigne, and P. Dong. Boundary detection by constrained optimization. *IEEE transactions on pattern analysis and machine intelligence*, 12(7):609–628, 1990.

[99] G. Ghiasi and C. C. Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *European Conference on Computer Vision*, pages 519–534. Springer, 2016.

[100] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[101] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017.

[102] T. Gold. The arrow of time. *American Journal of Physics*, 30(6):403–410, 1962.

[103] C. C. Goren, M. Sarty, and P. Y. Wu. Visual following and pattern discrimination of face-like stimuli by newborn infants. *Pediatrics*, 56(4):544–549, 1975.

[104] E. J. Gumbel. *Statistics of extremes*. Courier Corporation, 2012.

[105] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[106] W. Habicht and B. Van der Waerden. Lagerung von punkten auf der kugel. *Mathematische Annalen*, 123(1):223–234, 1951.

[107] W. Hackbusch. *Multi-grid methods and applications*, volume 4. Springer Science & Business Media, 2013.

[108] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR)*, 2016.

[109] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 991–998. IEEE, 2011.

[110] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, pages 297–312. Springer, 2014.

[111] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 447–456, 2015.

[112] A. W. Harley, K. G. Derpanis, and I. Kokkinos. Segmentation-aware convolutional networks using local attention masks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5038–5047, 2017.

[113] C. Hazirbas, L. Ma, C. Domokos, and D. Cremers. Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture. In *ACCV*, 2016.

[114] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017.

[115] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[116] S. Hickson, K. Raveendran, A. Fathi, K. Murphy, and I. Essa. Floors are flat: Leveraging semantics for real-time surface normal prediction. *arXiv preprint arXiv:1906.06792*, 2019.

[117] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[118] M. Hirsch, C. J. Schuler, S. Harmeling, and B. Schölkopf. Fast removal of non-uniform camera shake. In *2011 International Conference on Computer Vision*, pages 463–470. IEEE, 2011.

[119] M. Hirsch, S. Sra, B. Schölkopf, and S. Harmeling. Efficient filter flow for space-variant multiframe blind deconvolution. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 607–614. IEEE, 2010.

[120] E. Hoffer, I. Hubara, and D. Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pages 1731–1741, 2017.

[121] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017.

[122] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *Proceedings of the IEEE International Conference on Computer Vision*, 2005.

[123] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. *International Journal of Computer Vision*, 80(1):3–15, 2008.

[124] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[125] J. Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *IEEE transactions on pattern analysis and machine intelligence*, 38(4):814–830, 2016.

[126] M. Hradis, J. Kotera, P. Zemcík, and F. Sroubek. Convolutional neural networks for direct text deblurring. In *Proceedings of BMVC*, volume 10, page 2, 2015.

[127] H. Hu, S. Lan, Y. Jiang, Z. Cao, and F. Sha. Fastmask: Segment multi-scale object candidates in one shot. *arXiv preprint arXiv:1612.08843*, 2016.

[128] Z. Hu and M.-H. Yang. Good regions to deblur. In *European conference on computer vision*, pages 59–72. Springer, 2012.

[129] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[130] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision (ECCV)*, pages 646–661. Springer, 2016.

[131] J.-B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5197–5206, 2015.

[132] A. Humayun, F. Li, and J. M. Rehg. Rigor: Reusing inference in graph cuts for generating object regions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–343, 2014.

[133] A. Humayun, F. Li, and J. M. Rehg. The middle child problem: Revisiting parametric min-cut and seeds for object proposals. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1600–1608, 2015.

[134] J. Hur and S. Roth. Mirrorflow: Exploiting symmetries in joint optical flow and occlusion estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 312–321, 2017.

[135] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[136] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2462–2470, 2017.

[137] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[138] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[139] V. Jain and S. Seung. Natural image denoising with convolutional networks. In *Advances in Neural Information Processing Systems*, pages 769–776, 2009.

[140] V. Jampani, M. Kiefel, and P. V. Gehler. Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4452–4461, 2016.

[141] J. Janai, F. Guney, A. Ranjan, M. Black, and A. Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 690–706, 2018.

[142] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations (ICLR)*, 2017.

[143] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th international conference on computer vision*, pages 2146–2153. IEEE, 2009.

[144] J. Y. Jason, A. W. Harley, and K. G. Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *European Conference on Computer Vision*, pages 3–10. Springer, 2016.

[145] D. Jayaraman and K. Grauman. Learning image representations tied to ego-motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1413–1421, 2015.

[146] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 3192–3199, 2013.

[147] G. Ji, M. C. Hughes, and E. B. Sudderth. From patches to images: a nonparametric generative model. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1675–1683. JMLR. org, 2017.

[148] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool. Dynamic filter networks. In *Advances in Neural Information Processing Systems*, pages 667–675, 2016.

[149] H. Jiang, G. Larsson, M. Maire Greg Shakhnarovich, and E. Learned-Miller. Self-supervised relative depth learning for urban scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–35, 2018.

[150] J. Kaipio and E. Somersalo. *Statistical and computational inverse problems*, volume 160. Springer Science & Business Media, 2006.

[151] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[152] A. Khoreva, F. Perazzi, R. Benenson, B. Schiele, and A. Sorkine-Hornung. Learning video object segmentation from static images. *CoRR*, abs/1612.02646, 2016.

[153] E. Kidron, Y. Y. Schechner, and M. Elad. Pixels that sound. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 88–95. IEEE, 2005.

[154] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 1646–1654, 2016.

[155] K. I. Kim and Y. Kwon. Single-image super-resolution using sparse regression and natural image prior. *IEEE transactions on pattern analysis & machine intelligence*, (6):1127–1133, 2010.

[156] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[157] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother. Instancecut: from edges to instances with multicut. *arXiv preprint arXiv:1611.08272*, 2016.

[158] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980, 2017.

[159] T. Kobayashi and N. Otsu. Von mises-fisher mean shift for clustering on a hypersphere. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2130–2133. IEEE, 2010.

[160] P. Kohli, P. H. Torr, et al. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.

[161] I. Kokkinos. Pushing the boundaries of boundary detection using deep learning. In *International Conference on Learning Representations (ICLR)*, 2016.

[162] B. Kong, J. Supancic, D. Ramanan, and C. Fowlkes. Cross-domain forensic shoeprint matching. In *British Machine Vision Conference (BMVC)*, pages 1–5, 2017.

[163] S. Kong and C. Fowlkes. Low-rank bilinear pooling for fine-grained classification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7025–7034. IEEE, 2017.

[164] S. Kong and C. Fowlkes. Image reconstruction with predictive filter flow. *arXiv preprint arXiv:1811.11482*, 2018.

[165] S. Kong and C. Fowlkes. Pixel-wise attentional gating for parsimonious pixel labeling. *arXiv preprint arXiv:1805.01556*, 2018.

[166] S. Kong and C. Fowlkes. Recurrent pixel embedding for instance grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9018–9028, 2018.

[167] S. Kong and C. Fowlkes. Recurrent scene parsing with perspective understanding in the loop. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[168] S. Kong and C. Fowlkes. Multigrid predictive filter flow for unsupervised learning on videos. *arXiv preprint arXiv:1904.01693*, 2019.

[169] S. Kong, S. Punyasena, and C. Fowlkes. Spatially aware dictionary learning and coding for fossil pollen identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–10, 2016.

[170] S. Kong, X. Shen, Z. Lin, R. Mech, and C. Fowlkes. Photo aesthetics ranking network with attributes and content adaptation. In *European Conference on Computer Vision (ECCV)*, pages 662–679. Springer, 2016.

[171] S. Kong and D. Wang. A dictionary learning approach for classification: Separating the particularity and the commonality. *Computer Vision–ECCV 2012*, pages 186–199, 2012.

[172] S. Kong and D. Wang. A multi-task learning strategy for unsupervised clustering via explicitly separating the commonality. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 771–774. IEEE, 2012.

[173] S. Kong and D. Wang. Learning exemplar-represented manifolds in latent space for classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 240–255. Springer, 2013.

[174] S. Kornblith, J. Shlens, and Q. V. Le. Do better imagenet models transfer better? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2661–2671, 2019.

[175] P. Krahenbuhl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *NIPS*, 2011.

[176] P. Krähenbühl and V. Koltun. Geodesic object proposals. In *European Conference on Computer Vision*, pages 725–739. Springer, 2014.

[177] P. Krahenbuhl and V. Koltun. Learning to propose objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1574–1582, 2015.

[178] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition. In *European Conference on Computer Vision*, pages 301–320. Springer, 2016.

[179] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[180] A. Krogh and J. A. Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992.

[181] V. Kulikov and V. Lempitsky. Instance segmentation of biological images using harmonic embeddings. *arXiv preprint arXiv:1904.05257*, 2019.

[182] D. Kundur and D. Hatzinakos. Blind image deconvolution. *IEEE signal processing magazine*, 13(3):43–64, 1996.

[183] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[184] L. Ladicky, J. Shi, and M. Pollefeys. Pulling things out of perspective. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[185] L. Ladickỳ, P. Sturgess, K. Alahari, C. Russell, and P. H. Torr. What, where and how many? combining object detectors and crfs. In *European conference on computer vision*, pages 424–437. Springer, 2010.

[186] L. Ladicky, B. Zeisl, and M. Pollefeys. Discriminatively trained dense surface normal estimation. In *European Conference on Computer Vision (ECCV)*, pages 468–484. Springer, 2014.

[187] W. Lai, J. Huang, N. Ahuja, and M. Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5835–5843, 2017.

[188] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *International Conference on 3D Vision (3DV)*, pages 239–248. IEEE, 2016.

[189] G. Larsson, M. Maire, and G. Shakhnarovich. Colorization as a proxy task for visual understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6874–6883, 2017.

[190] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[191] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[192] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[193] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[194] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, volume 2, page 4, 2017.

[195] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *Artificial intelligence and statistics*, pages 562–570, 2015.

[196] D. C. Lee, M. Hebert, and T. Kanade. Geometric reasoning for single image structure recovery. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2136–2143. IEEE, 2009.

[197] K. Lee, R. Lu, K. Luther, and H. S. Seung. Learning dense voxel embeddings for 3d neuron reconstruction. *arXiv preprint arXiv:1909.09872*, 2019.

[198] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1964–1971. IEEE, 2009.

[199] C. Levy and T. Roosendaal. Sintel. In *ACM SIGGRAPH ASIA 2010 Computer Animation Festival, Seoul, Republic of Korea, December 15 - 18, 2010*, page 82:1, 2010.

[200] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1119–1127, 2015.

[201] J. Li, F. Fang, K. Mei, and G. Zhang. Multi-scale residual network for image super-resolution. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VIII*, pages 527–542, 2018.

[202] K. Li, B. Hariharan, and J. Malik. Iterative instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[203] X. Li, S. Liu, S. De Mello, X. Wang, J. Kautz, and M.-H. Yang. Joint-task self-supervised learning for temporal correspondence. In *NeurIPS*, 2019.

[204] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. *arXiv preprint arXiv:1611.07709*, 2016.

[205] X. Liang, Y. Wei, X. Shen, Z. Jie, J. Feng, L. Lin, and S. Yan. Reversible recursive instance-level object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 633–641, 2016.

[206] X. Liang, Y. Wei, X. Shen, J. Yang, L. Lin, and S. Yan. Proposal-free network for instance-level object segmentation. *arXiv preprint arXiv:1509.02636*, 2015.

[207] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1132–1140, 2017.

[208] G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[209] G. Lin, C. Shen, A. van den Hengel, and I. Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[210] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[211] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[212] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017.

[213] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014.

[214] T. Lindeberg. Feature detection with automatic scale selection. *International journal of computer vision*, 30(2):79–116, 1998.

[215] Z. C. Lipton. The mythos of model interpretability. *Commun. ACM*, 61(10):36–43, 2018.

[216] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz. Adaptive deblocking filter. *IEEE transactions on circuits and systems for video technology*, 13(7):614–619, 2003.

[217] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. Van Ginneken, and C. I. Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[218] C. Liu et al. *Beyond pixels: exploring new representations and applications for motion analysis*. PhD thesis, Massachusetts Institute of Technology, 2009.

[219] F. Liu, C. Shen, and G. Lin. Deep convolutional neural fields for depth estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5162–5170, 2015.

[220] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo. Multi-level wavelet-cnn for image restoration. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[221] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[222] I. Loshchilov and F. Hutter. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015.

[223] L. Lovisolo and E. Da Silva. Uniform distribution of points on a hyper-sphere with applications to vector bit-plane encoding. *IEE Proceedings-Vision, Image and Signal Processing*, 148(3):187–193, 2001.

[224] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[225] Z. Luo, B. Peng, D.-A. Huang, A. Alahi, and L. Fei-Fei. Unsupervised learning of long-term motion dynamics for videos. *arXiv:1701.01821*, 2017.

[226] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[227] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations (ICLR)*, 2017.

[228] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 181–196, 2018.

[229] M. Maire, T. Narihira, and S. X. Yu. Affinity cnn: Learning pixel-centric pairwise relations for figure/ground embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 174–182, 2016.

[230] A. Mallya and S. Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. *arXiv preprint arXiv:1711.05769*, 2017.

[231] L. Mander, M. Li, W. Mio, C. C. Fowlkes, and S. W. Punyasena. Classification of grass pollen through the quantitative analysis of surface ornamentation and texture. *Proceedings of the Royal Society B: Biological Sciences*, 280(1770):20131905, 2013.

[232] K.-K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool. Convolutional oriented boundaries: From image segmentation to high-level tasks. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):819–833, 2018.

[233] K. V. Mardia and P. E. Jupp. *Directional statistics*, volume 494. John Wiley & Sons, 2009.

[234] D. Martin, C. Fowlkes, D. Tal, J. Malik, et al. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. Iccv Vancouver:, 2001.

[235] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (5):530–549, 2004.

[236] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016.

[237] X. Mei, X. Sun, W. Dong, H. Wang, and X. Zhang. Segment-tree based cost aggregation for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 313–320, 2013.

[238] S. Meister, J. Hur, and S. Roth. Unflow: Unsupervised learning of optical flow with a bidirectional census loss. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[239] M. Menze, C. Heipke, and A. Geiger. Discrete optimization for optical flow. In *German Conference on Pattern Recognition*, pages 16–28. Springer, 2015.

[240] B. Mildenhall, J. T. Barron, J. Chen, D. Sharlet, R. Ng, and R. Carroll. Burst denoising with kernel prediction networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2502–2510, 2018.

[241] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016.

[242] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations (ICLR)*, 2017.

[243] M. K. Moore, R. Borton, and B. L. Darby. Visual tracking in young infants: Evidence for object identity or object permanence? *Journal of Experimental Child Psychology*, 25(2):183–198, 1978.

[244] R. T. Mullapudi, S. Chen, K. Zhang, D. Ramanan, and K. Fatahalian. Online model distillation for efficient video inference. *arXiv preprint arXiv:1812.02699*, 2018.

[245] A. A. Muller and R. N. Aslin. Visual tracking as an index of the object concept. *Infant Behavior and Development*, 1:309–319, 1978.

[246] N. Murray, L. Marchesotti, and F. Perronnin. Ava: A large-scale database for aesthetic visual analysis. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2408–2415. IEEE, 2012.

[247] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[248] L. Najman and M. Schmitt. Geodesic saliency of watershed contours and hierarchical segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 18(12):1163–1173, 1996.

[249] A. Newell, Z. Huang, and J. Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in Neural Information Processing Systems*, pages 2277–2287, 2017.

[250] P. X. Nguyen, G. Rogez, C. Fowlkes, and D. Ramanan. The open world of micro-videos. *arXiv preprint arXiv:1603.09439*, 2016.

[251] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 670–679, 2017.

[252] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016.

[253] K. Ovtcharov, O. Ruwase, J.-Y. Kim, J. Fowers, K. Strauss, and E. S. Chung. Accelerating deep convolutional neural networks using specialized hardware. *Microsoft Research Whitepaper*, 2(11), 2015.

[254] A. Owens and A. A. Efros. Audio-visual scene analysis with self-supervised multisensory features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 631–648, 2018.

[255] A. Owens, P. Isola, J. McDermott, A. Torralba, E. H. Adelson, and W. T. Freeman. Visually indicated sounds. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2405–2413, 2016.

[256] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba. Ambient sound provides supervision for visual learning. In *European conference on computer vision*, pages 801–816. Springer, 2016.

[257] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[258] S. C. Park, M. K. Park, and M. G. Kang. Super-resolution image reconstruction: a technical overview. *IEEE signal processing magazine*, 20(3):21–36, 2003.

[259] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[260] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[261] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2701–2710, 2017.

[262] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[263] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Transactions on graphics (TOG)*, 22(3):313–318, 2003.

[264] D. Perrone and P. Favaro. A clearer picture of total variation blind deconvolution. *IEEE transactions on pattern analysis and machine intelligence*, 38(6):1041–1055, 2016.

[265] Z. Pincus. Ageing: A stretch in time. *Nature*, 530(7588):37, 2016.

[266] P. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. In *Advances in Neural Information Processing Systems*, pages 1990–1998, 2015.

[267] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *European Conference on Computer Vision*, pages 75–91. Springer, 2016.

[268] L. Pinto, D. Gandhi, Y. Han, Y.-L. Park, and A. Gupta. The curious robot: Learning visual representations via physical interactions. In *European Conference on Computer Vision*, pages 3–18. Springer, 2016.

[269] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. Full-resolution residual networks for semantic segmentation in street scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4151–4160, 2017.

[270] J. Pont-Tuset, P. Arbelaez, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. *IEEE transactions on pattern analysis and machine intelligence*, 39(1):128–140, 2017.

[271] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

[272] K. R. Popper. The arrow of time. *Nature*, 177(4507):538, 1956.

[273] T. Porter and T. Duff. Compositing digital images. In *ACM Siggraph Computer Graphics*, volume 18, pages 253–259. ACM, 1984.

[274] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766. ACM, 2007.

[275] P. Rantalankila, J. Kannala, and E. Rahtu. Generating object segmentation proposals using global and local search. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2417–2424, 2014.

[276] S. N. Ravi, Y. Xiong, L. Mukherjee, and V. Singh. Filter flow made practical: Massively parallel and lock-free. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3549–3558, 2017.

[277] M. Ren, A. Pokrovsky, B. Yang, and R. Urtasun. Sbnet: Sparse blocks network for fast inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8711–8720, 2018.

[278] M. Ren and R. S. Zemel. End-to-end instance segmentation with recurrent attention. In *CVPR*, 2017.

[279] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[280] Z. Ren and G. Shakhnarovich. Image segmentation by cascaded region agglomeration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2011–2018, 2013.

[281] Z. Ren, J. Yan, B. Ni, B. Liu, X. Yang, and H. Zha. Unsupervised deep learning for optical flow estimation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[282] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1164–1172, 2015.

[283] H. Rhodin, M. Salzmann, and P. Fua. Unsupervised geometry-aware representation for 3d human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 750–767, 2018.

[284] B. Romera-Paredes and P. H. S. Torr. Recurrent instance segmentation. In *ECCV*, 2016.

[285] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[286] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243, 2016.

[287] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[288] E. B. Saff and A. B. Kuijlaars. Distributing many points on a sphere. *The mathematical intelligencer*, 19(1):5–11, 1997.

[289] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *NIPS*, 2005.

[290] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002.

[291] C. Schuler, M. Hirsch, S. Harmeling, and B. Scholkopf. Learning to deblur. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (1):1–1, 2016.

[292] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni. Green ai. *arXiv preprint arXiv:1907.10597*, 2019.

[293] S. M. Seitz and S. Baker. Filter flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[294] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[295] Q. Shan, J. Jia, and A. Agarwala. High-quality motion deblurring from a single image. In *Acm transactions on graphics (tog)*, volume 27, page 73. ACM, 2008.

[296] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[297] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

[298] M.-Y. Shen and C.-C. J. Kuo. Review of postprocessing techniques for compression artifact removal. *Journal of visual communication and image representation*, 9(1):2–14, 1998.

[299] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(8):888–905, 2000.

[300] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–769, 2016.

[301] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision (ECCV)*, pages 746–760. Springer, 2012.

[302] E. P. Simoncelli. Bayesian multi-scale differential optical flow. 1999.

[303] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[304] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[305] A. Sironi, V. Lepetit, and P. Fua. Multiscale centerline detection by learning a scale-space distance transform. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2697–2704, 2014.

[306] N. Skafte Detlefsen, O. Freifeld, and S. Hauberg. Deep diffeomorphic transformer networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4403–4412, 2018.

[307] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[308] T. Standley, A. R. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese. Which tasks should be learned together in multi-task learning? *arXiv preprint arXiv:1905.07553*, 2019.

[309] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[310] J. Sun, W. Cao, Z. Xu, and J. Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 769–777, 2015.

[311] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[312] P. Svoboda, M. Hradis, D. Barina, and P. Zemcik. Compression artifacts removal using convolutional neural networks. *arXiv preprint arXiv:1605.00366*, 2016.

[313] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[314] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[315] R. Szeliski. *Bayesian modeling of uncertainty in low-level vision*, volume 79. Springer Science & Business Media, 2012.

[316] A. Tarantola. *Inverse problem theory and methods for model parameter estimation*, volume 89. siam, 2005.

[317] T. T. D. Team, R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

[318] R. Timofte, V. De Smet, and L. Van Gool. Anchored neighborhood regression for fast example-based super-resolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1920–1927, 2013.

[319] A. Torralba, A. A. Efros, et al. Unbiased look at dataset bias. In *CVPR*, volume 1, page 7. Citeseer, 2011.

[320] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Elsevier, 2000.

[321] H.-Y. Tung, H.-W. Tung, E. Yumer, and K. Fragkiadaki. Self-supervised learning of motion capture. In *Advances in Neural Information Processing Systems*, pages 5236–5246, 2017.

[322] J. Uhrig, M. Cordts, U. Franke, and T. Brox. Pixel-level encoding and depth layering for instance-level semantic labeling. In *German Conference on Pattern Recognition*, pages 14–25. Springer, 2016.

[323] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[324] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[325] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid. Learning from synthetic humans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 109–117, 2017.

[326] A. Vedaldi and K. Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 689–692. ACM, 2015.

[327] A. Veit and S. Belongie. Convolutional networks with adaptive computation graphs. *arXiv preprint arXiv:1711.11503*, 2017.

[328] A. Veit, M. J. Wilber, and S. Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 550–558, 2016.

[329] A. Vellido, J. D. Martín-Guerrero, and P. J. Lisboa. Making machine learning models interpretable. In *ESANN*, volume 12, pages 163–172. Citeseer, 2012.

[330] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017.

[331] C. Von Hofsten, O. Kochukhova, and K. Rosander. Predictive tracking over occlusions by 4-month-old infants. *Developmental Science*, 10(5):625–640, 2007.

[332] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, pages 613–621, 2016.

[333] C. Vondrick, A. Shrivastava, A. Fathi, S. Guadarrama, and K. Murphy. Tracking emerges by colorizing videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 391–408, 2018.

[334] G. K. Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[335] H. Wang, Y. Rivenson, Y. Jin, Z. Wei, R. Gao, H. Günaydın, L. A. Bentolila, C. Kural, and A. Ozcan. Deep learning enables cross-modality super-resolution in fluorescence microscopy. *Nat. Methods*, 16:103–110, 2019.

[336] X. Wang, D. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 539–547, 2015.

[337] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2015.

[338] X. Wang, K. He, and A. Gupta. Transitive invariance for self-supervised visual representation learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1329–1338, 2017.

[339] X. Wang, A. Jabri, and A. A. Efros. Learning correspondence from the cycle-consistency of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2566–2576, 2019.

[340] X. Wang, A. Jabri, and A. A. Efros. Learning correspondence from the cycle-consistency of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[341] X. Wang, F. Yu, Z.-Y. Dou, and J. E. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. *arXiv preprint arXiv:1711.09485*, 2017.

[342] X. Wang, K. Yu, C. Dong, and C. C. Loy. Recovering realistic texture in image super-resolution by deep spatial feature transform. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 606–615, 2018.

[343] Y. Wang, Y. Yang, Z. Yang, L. Zhao, P. Wang, and W. Xu. Occlusion aware unsupervised learning of optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4884–4893, 2018.

[344] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[345] D. Wei, J. J. Lim, A. Zisserman, and W. T. Freeman. Learning and using the arrow of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8052–8060, 2018.

[346] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *CVPR*, 2016.

[347] M. Weigert, U. Schmidt, T. Boothe, A. Müller, A. Dibrov, A. Jain, B. Wilhelm, D. Schmidt, C. Broaddus, S. Culley, et al. Content-aware image restoration: pushing the limits of fluorescence microscopy. *Nature methods*, 15(12):1090, 2018.

[348] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[349] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1385–1392, 2013.

[350] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce. Non-uniform deblurring for shaken images. *International journal of computer vision*, 98(2):168–186, 2012.

[351] C.-Y. Wu, R. Manmatha, A. Smola, and P. K. Sampling matters in deep embedding learning. *arXiv preprint arXiv:1706.07567*, 2017.

[352] J. Wu, J. J. Lim, H. Zhang, J. B. Tenenbaum, and W. T. Freeman. Physics 101: Learning physical object properties from unlabeled videos. In *BMVC*, volume 2, page 7, 2016.

[353] Y. Wu and K. He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[354] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris. Blockdrop: Dynamic inference paths in residual networks. *arXiv preprint arXiv:1711.08393*, 2017.

[355] Z. Wu, C. Shen, and A. v. d. Hengel. Bridging category-level and instance-level semantic image segmentation. *arXiv preprint arXiv:1605.06885*, 2016.

[356] F. Xia, P. Wang, L.-C. Chen, and A. L. Yuille. Zoom better to see clearer: Human and object parsing with hierarchical auto-zoom net. In *European Conference on Computer Vision*, pages 648–663. Springer, 2016.

[357] L. Xiao, Y. Bahri, J. Sohl-Dickstein, S. S. Schoenholz, and J. Pennington. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. *arXiv preprint arXiv:1806.05393*, 2018.

[358] S. Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1395–1403, 2015.

[359] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In *Advances in Neural Information Processing Systems*, pages 1790–1798, 2014.

[360] L. Xu, S. Zheng, and J. Jia. Unnatural l0 sparse representation for natural image deblurring. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1107–1114, 2013.

[361] C.-Y. Yang, C. Ma, and M.-H. Yang. Single-image super-resolution: A benchmark. In *European Conference on Computer Vision*, pages 372–386. Springer, 2014.

[362] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang. Object contour detection with a fully convolutional encoder-decoder network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 193–202, 2016.

[363] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE transactions on image processing*, 19(11):2861–2873, 2010.

[364] L. Yang, Y. Wang, X. Xiong, J. Yang, and A. K. Katsaggelos. Efficient video object segmentation via network modulation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6499–6507, 2018.

[365] S. Yang and D. Ramanan. Multi-scale recognition with dag-cnns. In *Proceedings of the IEEE international conference on computer vision*, pages 1215–1223, 2015.

[366] Y. Yang, S. Hallman, D. Ramanan, and C. C. Fowlkes. Layered object models for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1731–1743, 2012.

[367] Z. Yang, P. Wang, Y. Wang, W. Xu, and R. Nevatia. Every pixel counts: Unsupervised geometry learning with holistic 3d motion understanding. In *European Conference on Computer Vision*, pages 691–709. Springer, 2018.

[368] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[369] R. Yu, C. Russell, N. D. Campbell, and L. Agapito. Direct, dense, and deformable: Template-based non-rigid 3d reconstruction from rgb video. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 918–926, 2015.

[370] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3712–3722, 2018.

[371] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pages 2528–2535. IEEE, 2010.

[372] M. D. Zeiler, G. W. Taylor, R. Fergus, et al. Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, volume 1, page 6, 2011.

[373] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *International conference on curves and surfaces*, pages 711–730. Springer, 2010.

[374] H. Zhan, R. Garg, C. Saroj Weerasekera, K. Li, H. Agarwal, and I. Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 340–349, 2018.

[375] H. Zhang, Y. N. Dauphin, and T. Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019.

[376] J. Zhang, J. Pan, J. S. J. Ren, Y. Song, L. Bao, R. W. H. Lau, and M. Yang. Dynamic scene deblurring using spatially variant recurrent neural networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 2521–2529, 2018.

[377] R. Zhang. Making convolutional networks shift-invariant again. 2019.

[378] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[379] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1058–1067, 2017.

[380] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018.

[381] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu. Image super-resolution using very deep residual channel attention networks. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, pages 294–310, 2018.

[382] Y. Zhang, S. Song, E. Yumer, M. Savva, J.-Y. Lee, H. Jin, and T. Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5287–5295, 2017.

[383] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu. Residual dense network for image super-resolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[384] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia. Icnet for real-time semantic segmentation on high-resolution images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 405–420, 2018.

[385] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[386] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015.

[387] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1851–1858, 2017.

[388] T. Zhou, P. Krahenbuhl, M. Aubry, Q. Huang, and A. A. Efros. Learning dense correspondence via 3d-guided cycle consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 117–126, 2016.

[389] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *European conference on computer vision*, pages 286–301. Springer, 2016.

[390] B. Zhu, J. Z. Liu, S. F. Cauley, B. R. Rosen, and M. S. Rosen. Image reconstruction by domain-transform manifold learning. *Nature*, 555(7697):487, 2018.

[391] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. Deep feature flow for video recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2349–2358, 2017.

[392] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[393] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *2011 International Conference on Computer Vision*, pages 479–486. IEEE, 2011.

Appendix A

Appendix to Chapter 3

We analyze the proposed depth-aware gating module with detailed results in Table A.1. We perform the ablation study on the Cityscapes dataset [57]. Specifically, we train the following models in order (except the fourth model which learns attention to gate).

1. "baseline" is our DeepLab-like baseline model by training two convolutional (with 3×3 kernels) layers above the ResNet101 backbone.

2. "tied, avg." is the model we train based on "baseline" by using the same 3×3 kernel but different dilate rates equal to $\{1, 2, 4, 8, 16\}$, respectively. So there are five branches and each of them has the same kernels which are tied to make processing scale-invariant. We average the feature maps for the final output prior to classification.

3. "gt-depth, tied, gating" is the model using the ground-truth depth map to select the branch; the pooling window size is determined according to the ground-truth depth value.

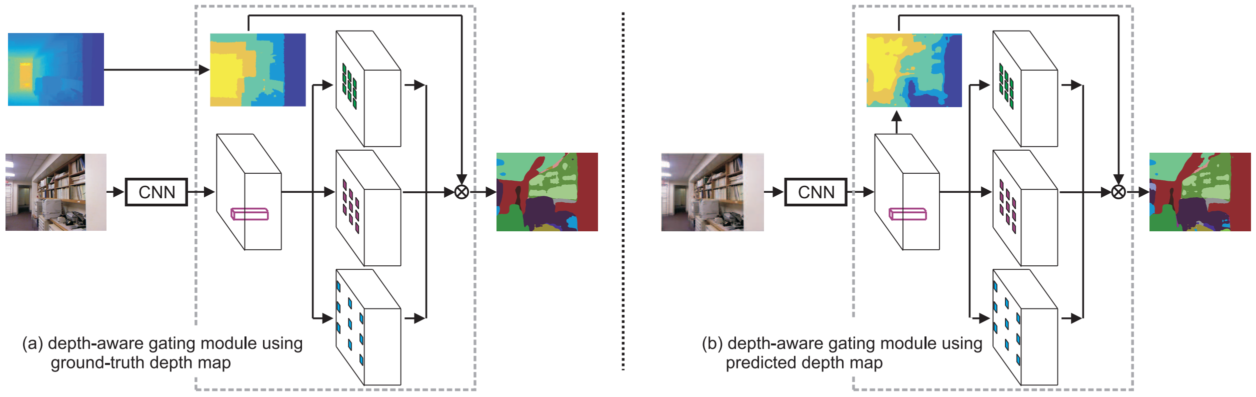
4. "gt-depth, untied, gating" is the model based on "gt-depth, tied, gating" by unleashing the tied kernels in the five branches. These untied kernels improve the flexibility and representation power of the network. Figure A.1 (a) depicts this model.

5. "attention, untied, gating" is an independent model to the previous ones that is trained

without depth supervision where the gating signal acts as a generic attentional signal that modulates spatially adaptive pooling. Specifically, we train an attention branch to produce the soft weight mask after softmax to gate the features from multiple pooling at different scales. We also adopt untied weights for the scale-specific pooling branches. The architecture is similar to what depicted in Figure A.1 (b), but without depth supervision.

6. "pred-depth, untied, gating" is our final model in which we learn a quantized depth predictor to gate the five branches. This model determines the size of pooling window based on its predicted depth map. Figure A.1 (b) shows the architecture of this model.

Through Table A.1, we can see that increasing the dilate rate with our model "tied, avg." improves the performance noticeably. This is consistent with the observation in [45], in which the large view-of-field version of DeepLab performs better. The benefit can be explained by the large dilation rate increasing the size of the receptive field, allowing more contextual information to be captured at higher levels of the network. With the gating mechanism, either using ground-truth depth map or the predicted one, the performance is improved further over non-adaptive pooling. The depth-aware gating module helps determine the pooling window size wisely, which is better than averaging all branches equally as in our "tied, avg." model and DeepLab. Moreover, by unleashing the tied kernels, the "gt-depth untied, gating" improves over "gt-depth, tied, gating" remarkably. We conjecture that this is because the untied kernels provide more flexibility to distinguish features at different scales and allow selection of the appropriate non-invariant features from lower in the network. Interestingly, the attention-gating model performs well and using the predicted depth map achieves the best among all these compared models. We attribute this to three reasons. Firstly, the predicted depth is smooth without holes or invalid entries. When using ground-truth depth on Cityscapes dataset, we assign equal weight on the missing entries so that the gating actually averages the information at different scales. This average pooling might be harmful


CNN
(a) depth-aware gating module using ground-truth depth map
CNN
(b) depth-aware gating module using predicted depth map
Figure A.1: (a) Depth-aware gating module using the ground-truth depth map, and (b) depth-aware gating module using the predicted depth map. The grids within the feature map blocks distinguish different pooling field sizes. Here we depict three different pooling window sizes while in our actual experiments we quantize the depth map into five scale bins.

in some cases such as very small object at distance. This can be taken as complementary evidence that the blindly averaging all branches achieves inferior performance to using the depth-aware gating. Secondly, the predicted depth maps have some object-aware pattern structure, which might be helpful for segmentation. From the visualization shown later in Figure A.4, we can observe such patterns, e.g. for cars. Thirdly, the depth prediction branch, as well as the attention branch, generally increases the representation power and flexibility of the whole model; this can be beneficial for segmentation.

A.1 Results on the SUN-RGBD dataset

In Figure A.2, we show the depth prediction results of several images randomly picked from the test set of SUN-RGBD dataset. Note that the there are unnatural regions in the ground-truth depth maps, which are the result of refined depth completion by the algorithm in [307]. Visually, these regions do not always make sense and constitute bad depth completions. In contrast, our predicted depth maps are much smoother. We also evaluate our depth prediction on SUN-RGBD dataset, and achieve 0.754, 0.899 and 0.961 by the three threshold metrics respectively. As SUN-RGBD is an extension of NYU-depth-v2 dataset, it has similar

Table A.1: Result of different depth-aware gating module deployments on Cityscapes dataset. IoU is short for intersection over union averaged over all classes, and nIoU is the weighted IoU through the pre-defined class weights provided by the benchmark.

	baseline		tied, avg.		gt-depth, tied, gating		gt-depth, untied, gating		attention, untied, gating		pred-depth, untied, gating	
	IoU	nIoU	IoU	nIoU	IoU	nIoU	IoU	nIoU	IoU	nIoU		
Score Avg.	**0.738**	**0.547**	**0.747**	**0.554**	**0.748**	**0.556**	**0.753**	**0.561**	**0.754**	**0.558**	**0.759**	**0.571**
road	0.980	–	0.981	–	0.981	–	0.982	–	0.982	–	0.982	–
sidewalk	0.849	–	0.847	–	0.849	–	0.982	–	0.853	–	0.857	–
building	0.916	–	0.917	–	0.918	–	0.919	–	0.923	–	0.920	–
wall	0.475	–	0.499	–	0.506	–	0.511	–	0.527	–	0.512	–
fence	0.596	–	0.605	–	0.605	–	0.611	–	0.618	–	0.614	–
pole	0.598	–	0.599	–	0.604	–	0.616	–	0.615	–	0.624	–
traffic light	0.684	–	0.674	–	0.678	–	0.692	–	0.689	–	0.699	–
traffic sign	0.780	–	0.776	–	0.775	–	0.782	–	0.783	–	0.790	–
vegetation	0.918	–	0.917	–	0.918	–	0.920	–	0.920	–	0.922	–
terrain	0.619	–	0.620	–	0.627	–	0.632	–	0.625	–	0.638	–
sky	0.941	–	0.937	–	0.940	–	0.942	–	0.943	–	0.944	–
person	0.803	0.635	0.803	0.631	0.804	0.639	0.808	0.648	0.804	0.641	0.814	0.659
rider	0.594	0.448	0.595	0.462	0.602	0.460	0.612	0.461	0.602	0.443	0.616	0.473
car	0.939	0.859	0.942	0.854	0.942	0.863	0.942	0.867	0.943	0.862	0.944	0.871
truck	0.631	0.398	0.666	0.421	0.679	0.407	0.679	0.417	0.666	0.424	0.674	0.421
bus	0.759	0.595	0.802	0.607	0.787	0.612	0.786	0.609	0.798	0.602	0.799	0.615
train	0.621	0.467	0.683	0.494	0.656	0.489	0.655	0.487	0.684	0.508	0.687	0.507
motorcycle	0.562	0.396	0.587	0.387	0.591	0.398	0.602	0.410	0.583	0.407	0.610	0.425
bicycle	0.755	0.582	0.747	0.387	0.753	0.575	0.761	0.586	0.760	0.575	0.765	0.594

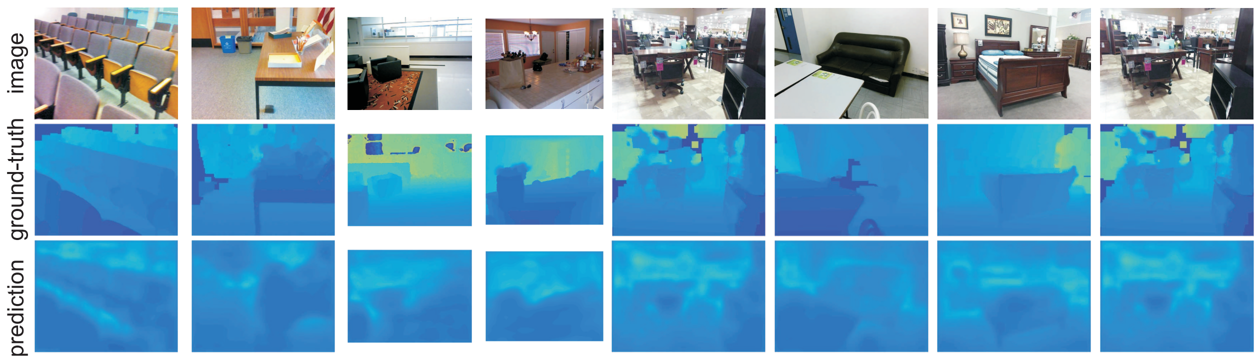


Figure A.2: Visualization of images from SUN-RGBD dataset and their ground-truth depth and our predicted depth on the three rows, respectively. We scale all the depth maps into a fixed range of $[0, 10^5]$. In this sense, the color of the depth maps directly reflect the absolute physical depth. Note that there are unnatural regions in the ground-truth depth maps, which have been refined by the algorithm in [307]. Visually, these refined region do not always make sense and are incorrect depth completions. In contrast, our monocular predictions are quite smooth.

data statistics resulting in similar prediction performance.

In Figure A.3, we randomly show fourteen images and their segmentation results at loops of the recurrent refining module. Visually, we can see that the our recurrent module refines the segmentation result in the loops.

A.2 Visualization on Large Perspective Images

In Figure A.4 and A.5, we visualize more results on Cityscapes and Stanford-2D-3D datasets, respectively. First, we show the segmentation prediction and the attention map after training with the unsupervised attentional mechanism in the third column. We can see the attention map appears to encode the distance from object boundaries. We hypothesize this selection mechanism serves to avoid pooling features across different semantic segments while still utilizing large pooling regions within each region. This is understandable and desirable in practice, as per-pixel feature vectors have different feature statistics for different categories. Then, we compare the segmentation results and depth estimate for adaptation in the recur-

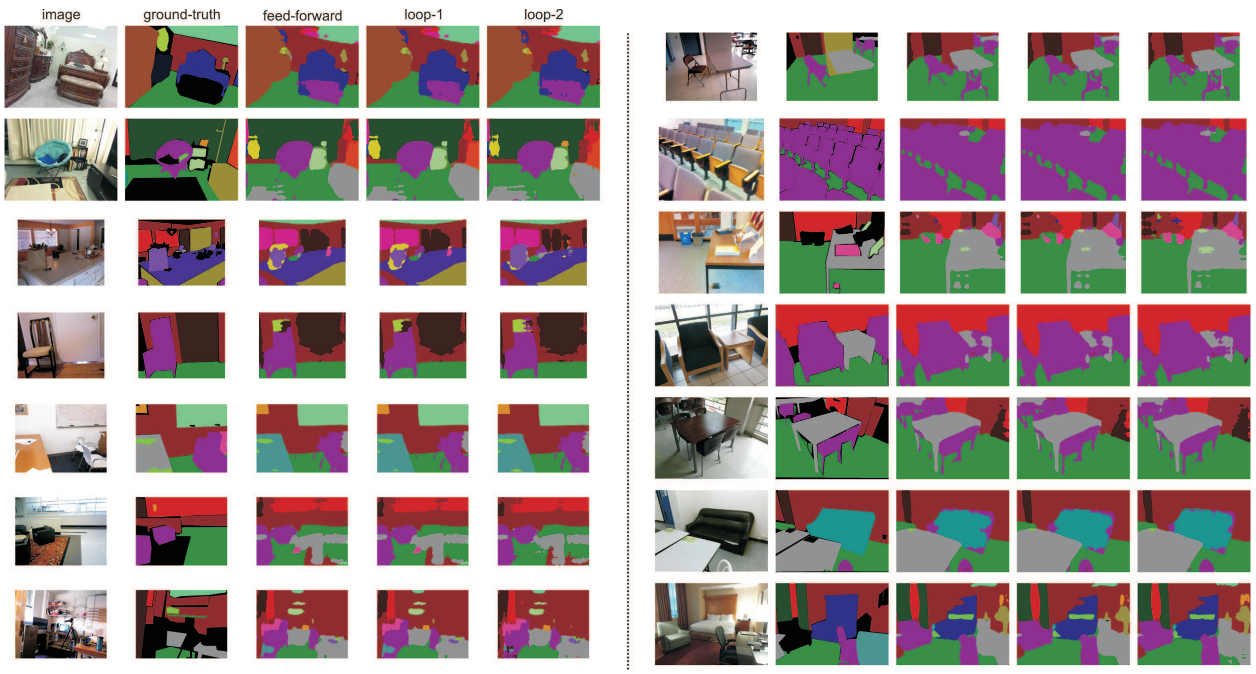


Figure A.3: Visualization of the output on SUN-RGBD dataset. We randomly show fourteen images from validation set with their segmentation output from both feed-forward pathway and recurrent loops. In the ground-truth segmentation annotation, we can see that there are many regions (with black color) not annotated.

rent refinement loops (last three columns in Figure A.4 and A.5). We notice that the depth estimate for adaptation changes remarkably in the loop (the depth module is fine-tuned using the segmentation loss only in training). While the depth estimate captures some object shapes in Cityscapes (e.g. car), it becomes more noticeable that the depth prediction helps the model perform coarse-to-fine refinement in the loop by using smaller receptive fields in Stanford-2D-3D dataset. We conjecture that this is owing to the top-down signal from the depth estimate at the previous loop. The recurrent refinement module also fills the holes in large areas, like light reflection regions on the car in street scene (Cityscapes) and white board in the second image (row 3 and 4) of panoramic photos (Stanford-2D-3D).

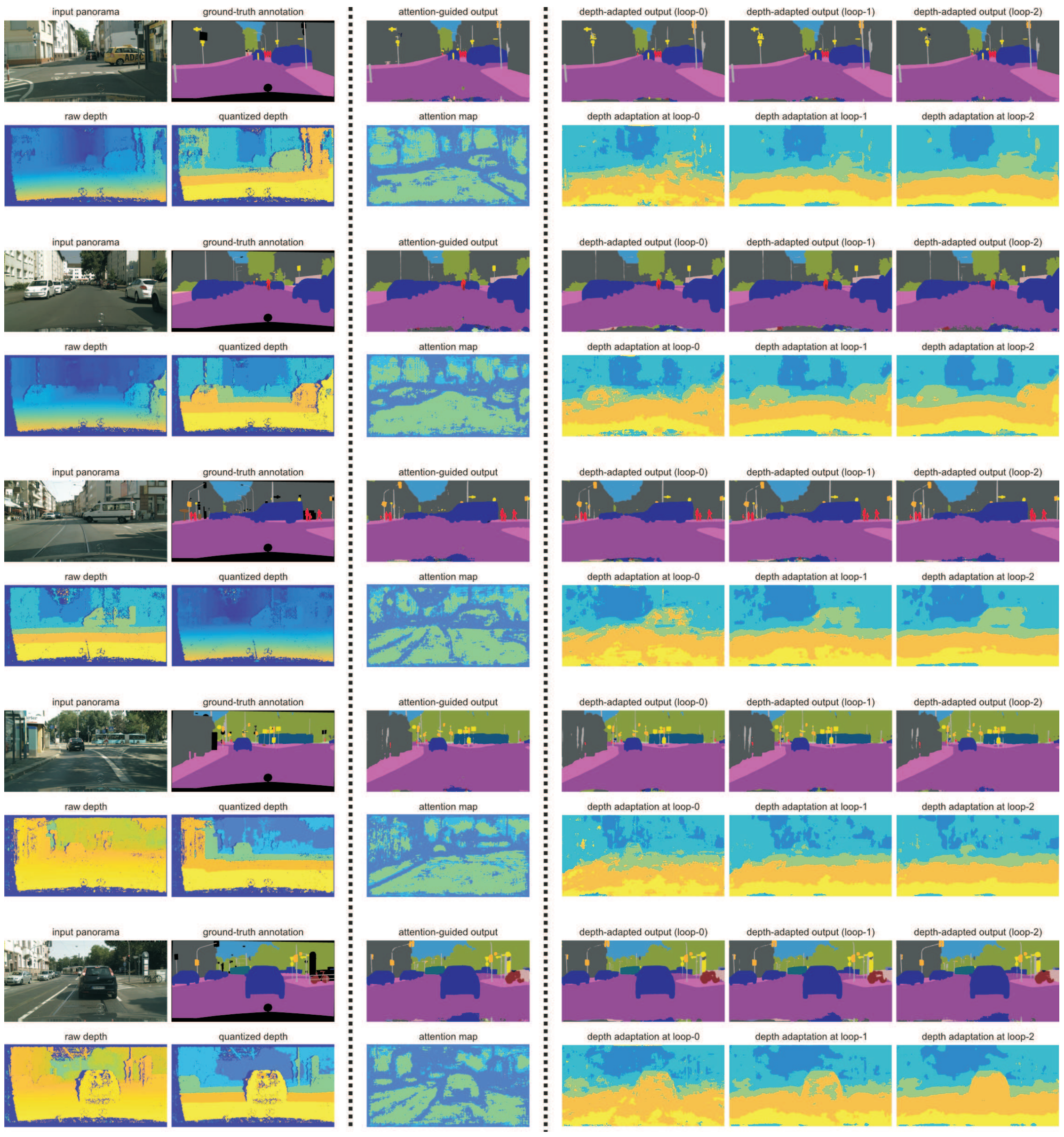

input panorama
ground-truth annotation
attention-guided output
depth-adapted output (loop-0)
depth-adapted output (loop-1)
depth-adapted output (loop-2)
raw depth
quantized depth
attention map
depth adaptation at loop-0
depth adaptation at loop-1
depth adaptation at loop-2

Figure A.4: Visualization of the results on Cityscapes dataset. For five random images from the validation set, we show the input perspective street scene photos, ground-truth annotation, raw disparity and the five-scale quantized depth map in the leftmost two columns. Then, we show the segmentation prediction and the attention map using our unsupervised attentional mechanism in the third column. In the rest three columns, we show the output of our depth-aware adaptation within recurrent refinement, from loop-0 to loop-2. Note that the more yellowish the color is, the closer the object is to camera and the finer scale of the feature maps the model adopts to process. From the visualization, we can see 1) the attention map helps the model avoid pooling across semantic segments; 2) the depth-adaptation in the recurrent refinement loops gradually captures objects like the cars, we attribute this to to the top-down signal from previous loops.

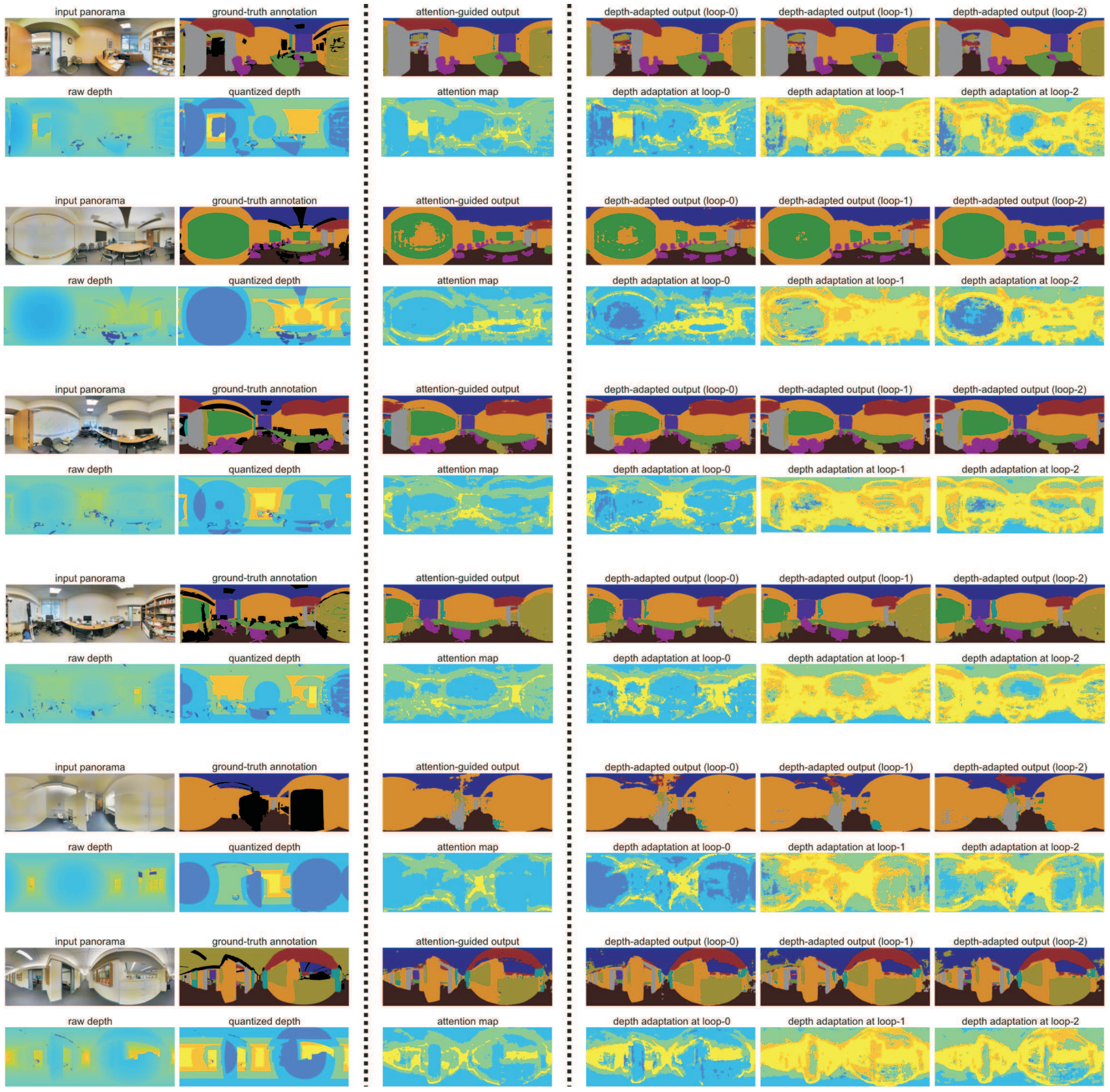

input panorama
ground-truth annotation
raw depth
quantized depth
attention-guided output
attention map
depth-adapted output (loop-0)
depth-adapted output (loop-1)
depth-adapted output (loop-2)
depth adaptation at loop-0
depth adaptation at loop-1
depth adaptation at loop-2

Figure A.5: Visualization of the results on Stanford-2D-3D dataset. For six random images from the validation set, we show the input panorama, ground-truth annotation, raw depth map and the five-scale quantized depth map in the leftmost two columns. Then, we show the segmentation prediction and the attention map using our unsupervised attentional mechanism in the third column. In the rest three columns, we show the output of our depth-aware adaptation within recurrent refinement, from loop-0 to loop-2. Note that the more yellowish the color is, the further away the object is to camera and the finer scale of the feature maps the model adopts to process. From the visualization, we can see 1) the attention map helps the model avoid pooling across semantic segments; 2) the depth-adaptation in the recurrent refinement loops fulfill coarse-to-fine processing as smaller receptive fields are used, due to the top-down signal from previous loops.

Appendix B

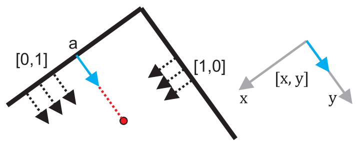
Appendix to Chapter 4

In the supplementary material, we first present in detail how to transform the (unpredictable) global surface normals into (predictable) local normals in panoramic images. We then show more ablation studies on the loss functions introduced in Chapter 4 and MultiPool module with/without PAG unit. Finally, we provide more qualitative visualization of the results for various pixel-labeling tasks, as well as the attentional ponder maps and MultiPool maps.

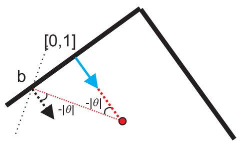
B.1 Local Surface Normal in Panoramas

Stanford-2D-3D [9] provides cylindrical panoramas with global surface normals, which are in a global Earth-Centered-Earth-Fixed coordinate system (ECEF). For example, the normals for a wall have the same direction pointing to the true north. However, such global coordinate system is impossible to determine from a single image, and thus the global normals are unpredictable purely based on panoramic image data alone. For this reason, we propose to transform the global normals into "local normals" which are relative to the camera viewing direction. We note that such a predictability makes relative normals more useful in scene understanding and reconstruction.

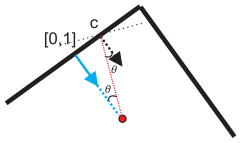
For a cylindrical panorama, we assume the vertical axis of the panorama is aligned with the



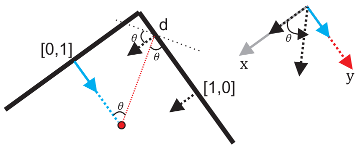
(a) Global normals in ECEF coordinate system. We define a canonical direction at point a to transform global normal to local normal relative to the camera viewing direction. Normal of point a is [0,1], exactly on the camera view direction.



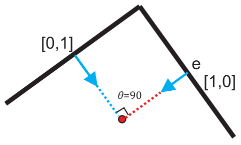
(b) For point b, the local normal can be obtained by rotating counter-clockwise its global normal [0,1] by -|θ| degree.



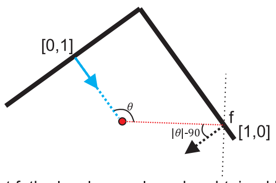
(c) For point c, the local normal can be understood by rotating counter-clockwise its global normal [0,1] by θ degree.



(d) For point d, the local normal can be obtained by rotating counter-clockwise its global normal [0,1] by θ



(e) For point e, the local normal [0,1] can be understood by rotating counter-clockwise its global normal [1,0] by θ=90 degree.



(f) For point f, the local normal can be obtained by rotating counter-clockwise [0,1] (the canonical normal) by |θ|-90 degree; equivalent to rotating its global normal [1,0] by 90 degree first (like e) and then |θ|-90 degree; equivalent to rotating counter-clockwise its global normal [1,0] by θ degree directly.

Figure B.1: An illustration of how to transform global normals into local normals specified relative to the camera viewing direction. Global normals in Stanford-2D-3D dataset [9] are in Earth-Centered-Earth-Fixed (ECEF) coordinate system.

global coordinate frame. Given a global normal at a pixel $\mathbf{n} = [x, y, z]^T$, we can apply a rotation matrix $\mathbf{R}$ in the horizontal plane (x and y) to obtain its local normal $[x', y', z]^T$ in the "camera viewing" coordinate system:

$$\left[\begin{array}{c} x' \\ y' \\ \hline z \end{array}\right] = \begin{bmatrix} \mathbf{R}, \mathbf{0} \\ \mathbf{0}, 1 \end{bmatrix} \left[\begin{array}{c} x \\ y \\ \hline z \end{array}\right] \tag{B.1}$$

where z is the variable for vertical direction.

We would like to determine the appropriate rotation matrices for all pixels where each pixel has its own rotation matrix which is controlled by a single signed angle parameter θ. For a cylindrical panorama, the relative difference in viewing direction between two image locations is completely specified by their horizontal separation in image coordinates. Therefore, to determine the set of rotations, we simply need to specify an origin for which the rotation

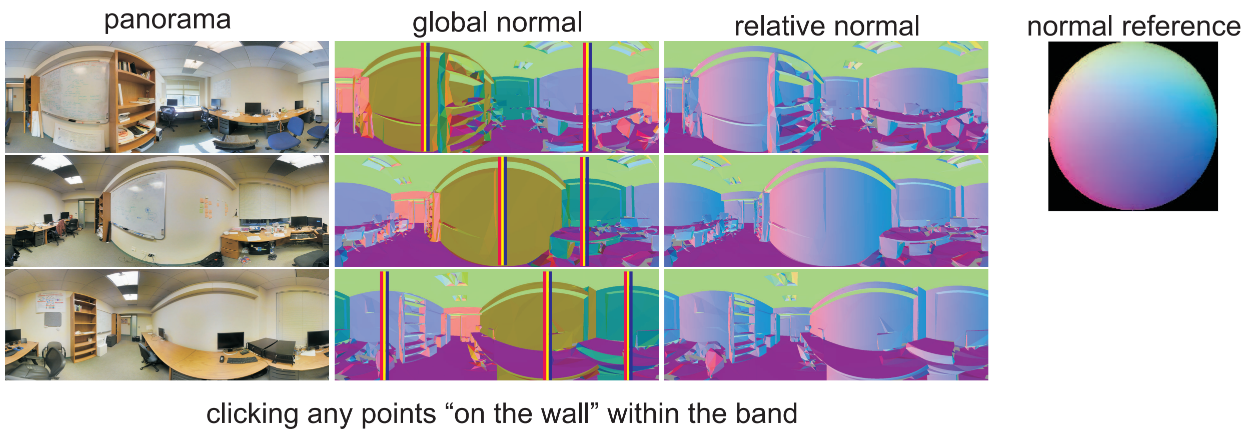


Figure B.2: We draw the color bands on global normal maps to indicate points of the wall within these bands can be treated as canonical points, whose normals exactly face towards the camera. For a human annotator, these points can be easily detected by looking at the shape of the room. Our interface allows an annotator to click a point supposedly within any one of the bands, and through the coordinate transform, such global normals can be transformed into predictable, relative normals. The rightmost pie chart provides reference on the local surface normal relative to the camera viewing direction.

is $\theta = 0$, i.e., a canonical point whose surface normal points exactly to the camera. Given the rotation matrix for the canonical point, the rotation angle for remaining points can be calculated as $2\pi\frac{\triangle W}{W}$, where W is the width of panorama and $\triangle W$ is the offset from the target pixel to the canonical pixel (with sign). Fig. B.1 illustrates the principle behind our methodology.

We note that it is straightforward to identify canonical points manually by choosing a flat vertical surface (e.g., a wall) and selecting the point on it which is nearest to the camera (e.g. shape of panoramic topology). An automated method can be built with the same rule based on semantic annotation and depth map. However, the automated method may suffer from cluttered scene (e.g. boards and bookcase on the wall), yet such manual annotation enables us to visualize what the local normals would look like if we clicked the "wrong" points*. From the three random panoramic images as shown in Fig. B.2, we can see such canonical points lie in the color bands (manually drawn for demonstration) noted in the figure. They are

*Clicking on the "wrong" points will leads to some normals pointing outwards the camera.

easy to detect by eye based on the warping effect due to panoramic topology. We made an easy-to-use tool to click a canonical point for each panoramic image. Fig. B.2 demonstrates the resulting transformed normals after an annotator has clicked on some point in the color band. We note that annotating each panoramic image costs less than 5 seconds, and it required less than three hours to carefully annotate all 1,559 panoramas in the dataset. We also compare the annotation when clicking on different canonical points (at different color bands) for the same image, and the maximum difference of normals for all spatial locations is only less than 8° degree. This means the annotation is easy and robust. We will release to public all the transformed local surface normals as an extension to Stanford-2D-3D dataset, as well as our interactive tool for annotation.

B.2 Further Analysis of Loss Functions and MultiPool Module

In this section, we describe further analysis on the architectural choices of where to insert the MultiPool module, as well as new loss functions introduced in Chapter 4. We conduct experiments on BSDS500 [7] and NYUv2 [74] datasets for boundary detection, depth and surface normal estimation, to complement the analysis in Chapter 4.

B.2.1 Boundary Detection on BSDS500 Dataset

In Fig. B.3, we show the precision-recall curves for boundary detection on BSDS500 dataset [7]. First, Fig. B.3 (a) summarizes our ablation study on where to insert the MultiPool module to obtain the largest performance gain. We observe that the best performance is achieved when the MultiPool module is inserted at the fourth macro building block (Res4), *i.e.* the second last macro block or the Resnet50 architecture. This supports our conclusion that MultiPool inserted at the second last macro building block leads to the largest performance gain.

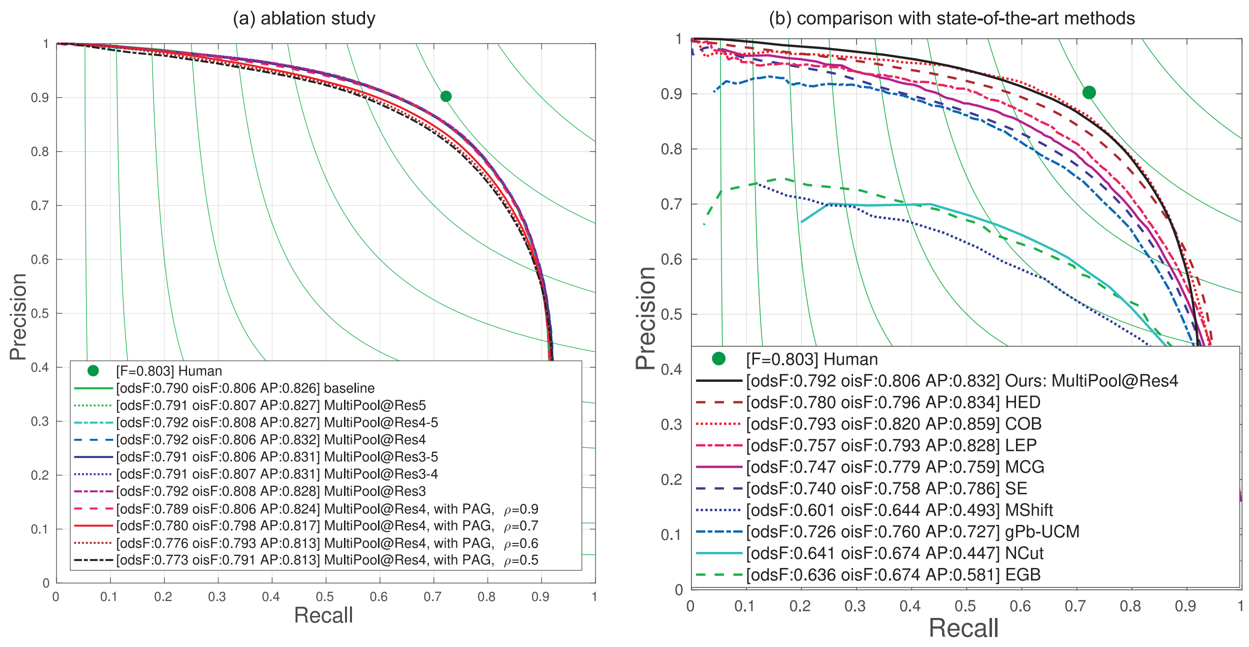


Figure B.3: Precision-Recall curves on boundary detection on BSDS500 dataset. (a) Ablation study. (b) Comparison with state-of-the-art methods.

Moreover, based on the "MultiPool@Res4" model, we gradually insert PAG units for parsimonious inference, and decrease the hyper-parameter ρ which controls the sparsity degree of binary masks (the sparser the masks are, the more parsimonious constraint is imposed). In particular, note that $\rho = 0.5$ means $\sim$30% computation has been saved, at which point our model achieves $F = 0.773$, only degrading by 2.4% performance. This shows that the ResNet50 model has sufficient capacity for boundary detection, and more parsimonious constraint does not harm the performance too much. Perhaps due to this reason, the MultiPool module does not improve performance remarkably for boundary detection.

Finally, by comparing to state-of-the-art methods as shown in Fig. B.3 (b), we note our "MultiPool@Res4" model outperforms HED [358] which shares the same architecture but without MultiPool module, and performs similarly with COB [232] which further exploits auxiliary loss for oriented boundary detection. This validates that the PAG-based MultiPool module improves performance by providing each pixel the "correct" size of pooling field.

Table B.1: Ablation study of where to insert the MultiPool module to obtain the largest performance gain for monocular depth estimation on NYUv2 dataset. All models are evaluated over the standard split of NYUv2 dataset, with $L2$ loss only, and with softmax weighted average in the MultiPool module. The performance is measured by standard threshold accuracy, *i.e.* the percentage of predicted pixel depths d_i s.t. $\delta = \max(\frac{d_i}{\hat{d}_i}, \frac{\hat{d}_i}{d_i}) < \tau$, where $\tau = \{1.25, 1.25^2, 1.25^3\}$.

metrics	base.	MP@Res3	MP@Res4	MP@Res5	MP@Res6	MP@Res3-4	MP@Res5-6
1.25	0.711	0.721	0.726	**0.737**	0.725	0.726	0.726
1.25^2	0.932	0.935	0.939	**0.939**	0.936	0.938	0.939
1.25^3	0.985	0.986	0.986	**0.986**	0.985	0.986	0.985

B.2.2 Monocular Depth Estimation on NYUv2 Dataset

We provide complementary ablation study on the task of monocular depth estimation on NYUv2 dataset.

First, we study where to insert the MultiPool module to obtain the largest performance gain. We train our base model using $L2$ loss function only, and insert the MultiPool module (without PAG but the softmax weighted average operation) at each macro building block one by one. We list the performance of these models in Table B.1. From the table, we observe that no matter where to insert the MultiPool module, it consistently improves the performance; while when MultiPool module is inserted at Res5, which is the second last macro building block, we obtain the largest performance gain. These observations, along with what reported in Chapter 4, support our conclusion that one is able to get the best performance when inserting the MultiPool module at the second last macro building block of a ResNet model.

B.2.3 Surface Normal Estimation on NYUv2 Dataset

Similar to the ablation study for monocular depth estimation task, we study firstly how the proposed loss function improves performance, then where to insert the MultiPool module for the best performance, and lastly performance comparison between PAG-based and weighted-

Table B.2: Ablation study of at which layer to insert *MultiPool* module (with softmax weighted average, w-Avg.) for surface normal estimation on NYUv2 dataset. Performance is measured by mean angular error (ang. err.) and the portion of prediction error within t° degree where $t = \{11.25, 22.50, 30.00\}$. Smaller ang. err. means better performance as marked by $\downarrow$.

metrics	base.	MP@Res3	MP@Res4	MP@Res5	MP@Res6	MP@Res3-4	MP@Res3-5
ang. err.↓	22.7170	21.9951	22.5506	**21.9556**	22.1661	22.5183	22.5051
11.25°	0.3382	0.3560	0.3366	**0.3567**	0.3514	0.3375	0.3389
22.50°	0.6195	0.6362	0.6188	**0.6374**	0.6323	0.6198	0.6209
30.00°	0.7383	0.7514	0.7386	**0.7526**	0.7482	0.7392	0.7394

Table B.3: Comparison of MultiPool module with PAG and softmax weighted average (w-Avg.) over surface normal estimation on NYUv2 dataset. Performance is measured by mean angular error (ang. err.) and the portion of prediction error within t° degree where $t = \{11.25, 22.50, 30.00\}$. Smaller ang. err. means better performance as marked by $\downarrow$.

metrics	MP@Res3 (w-Avg.)	MP@Res3 (PAG)	MP@Res5 (w-Avg.)	MP@Res5 (PAG)
ang. err.↓	21.9951	21.9793	21.9556	21.9226
11.25°	0.3560	0.3591	0.3567	0.3587
22.50°	0.6362	0.6396	0.6374	0.6384
30.00°	0.7514	0.7523	0.7526	0.7532

average MultiPool.

We then study where to insert the MultiPool module to get the best performance in Table B.2. Note that, in this ablation study, we didn't use PAG for binary masks, but instead using weighted average based on softmax operator. Consistent to previous discovery, when inserting MultiPool at the second last macro block, we achieve the best performance. In Table B.3, we compare the results with PAG-based MultiPool and weighted-average MultiPool. We conclude with consistent observation that PAG unit does not harm the performance compared to the softmax weighted average fusion, but instead achieves better performance with the same computation overhead, thanks to the perforated convolution.

▢ B.3 More Qualitative Visualization

In this section, we visualize more results of boundary detection, semantic segmentation, monocular depth estimation and surface normal estimation, over the four datasets used in Chapter 4, BSDS500 (Fig. B.4), NYUv2 (Fig. B.5), Stanford-2D-3D (Fig. B.6) and Cityscapes (Fig. B.7). In the figures, we show the ponder map for each macro building block, as well as the overall ponder map. From these ponder maps, we can see our model learns to dynamically allocate computation at different spatial location, primarily expending more computation on the regions/pixels which are regions with sharp changes, e.g. boundary between semantic segments, regions between two depth layer, locations around normal changes like between wall and ceiling. We also show all the binary maps produced by PAG units in Fig. B.8 over a random image from Stanford2D3D dataset for semantic segmentation, surface normal estimation and depth estimation.

In Fig. B.9, we visualize the learned binary masks by PerforatedCNN [84] on NYUv2 dataset for semantic segmentation. We also accumulate all the binary masks towards the ponder map, from which we can see that the active pixels largely follow uniform distribution. This is different from what reported in [84] that the masks mainly highlight central region in image classification, which is due to the fact that images for the classification task mainly contain object in the central region; whereas for scene images, it is hard for PerforatedCNN to focus on any specific location of the image. Note again that PerforatedCNN does not support either dynamic pixel routing or fully convolutional computation, requiring that the input image have a fixed size in order to learn fixed computation paths over the image. In contrast, our method is fully convolutional that is able to take as input images of arbitrary size and perform computing with input-dependent dynamic paths.

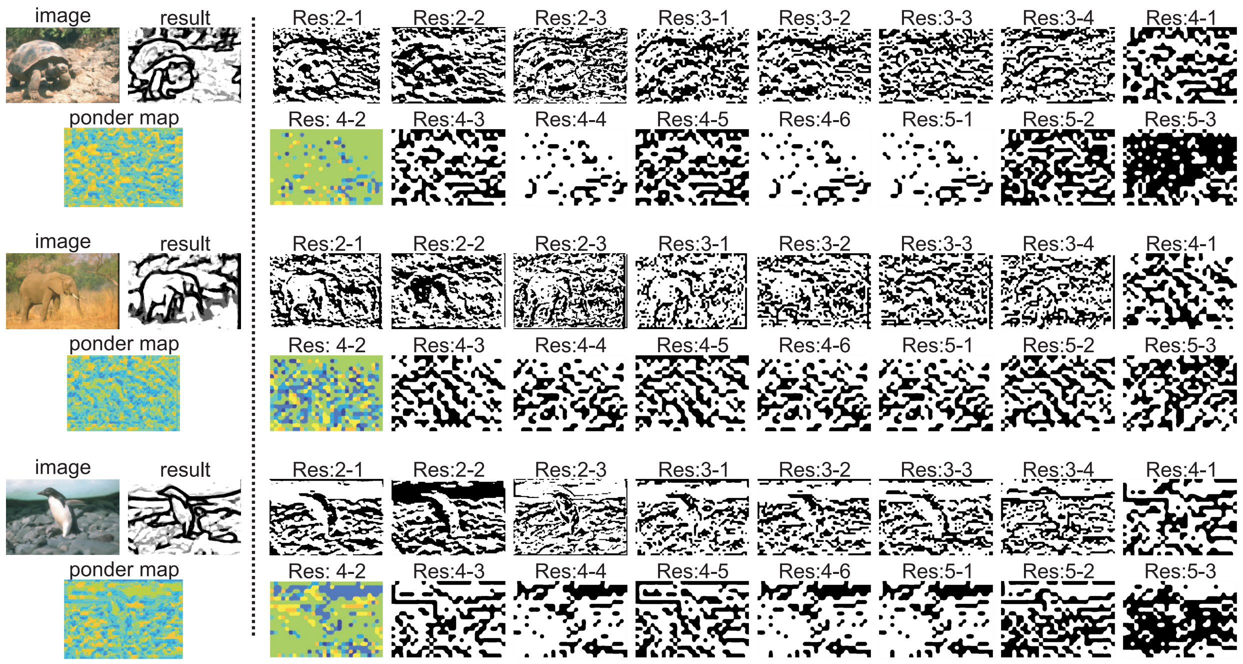


Figure B.4: Visualization on BSDS500 dataset [7] of sparse binary attention maps at each layer for boundary detection, together with the output and ponder map accumulating all binary maps. PAG-based MultiPool module is inserted at layer Res4-2, which is not included in the ponder map.

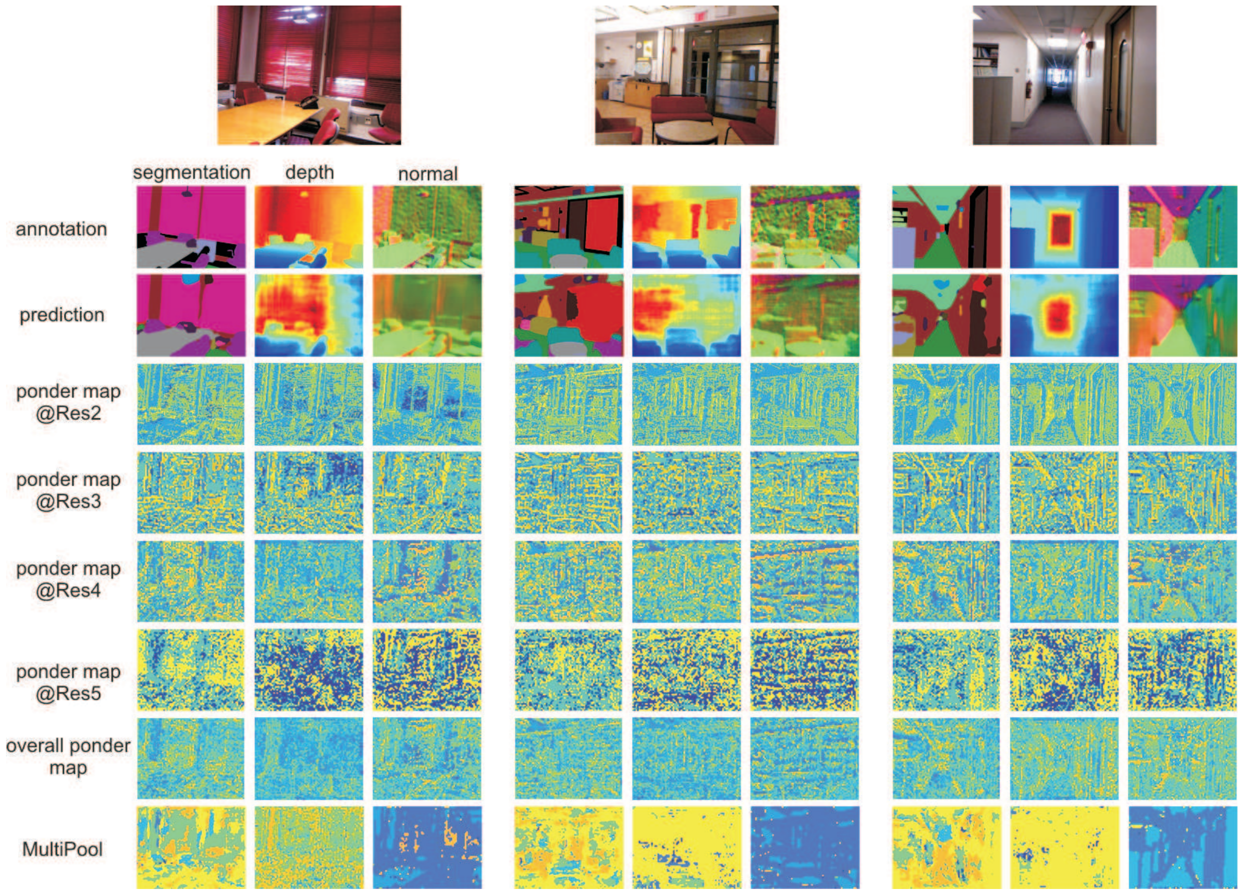


Figure B.5: Visualization on NYUv2 dataset [74] for semantic segmentation, depth estimation and surface normal estimation. Besides the overall ponder map, we also show the partial ponder map for each macro residual block by summing the sparse binary attentional maps. The MultiPool binary masks are not included in the ponder maps.

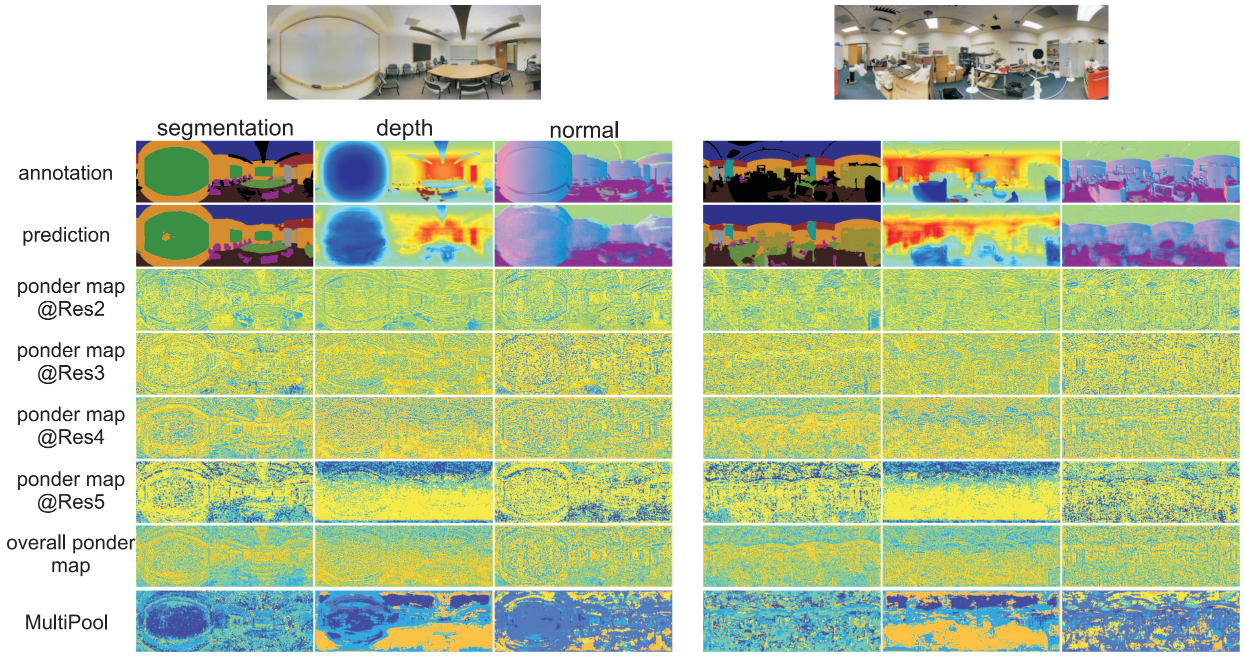


Figure B.6: Visualization on Stanford2D3D [9] for semantic segmentation, depth estimation and surface normal estimation. Besides the overall ponder map, we also show the partial ponder map for each macro residual block by summing the sparse binary attentional maps. The MultiPool binary masks are not included in the ponder maps.

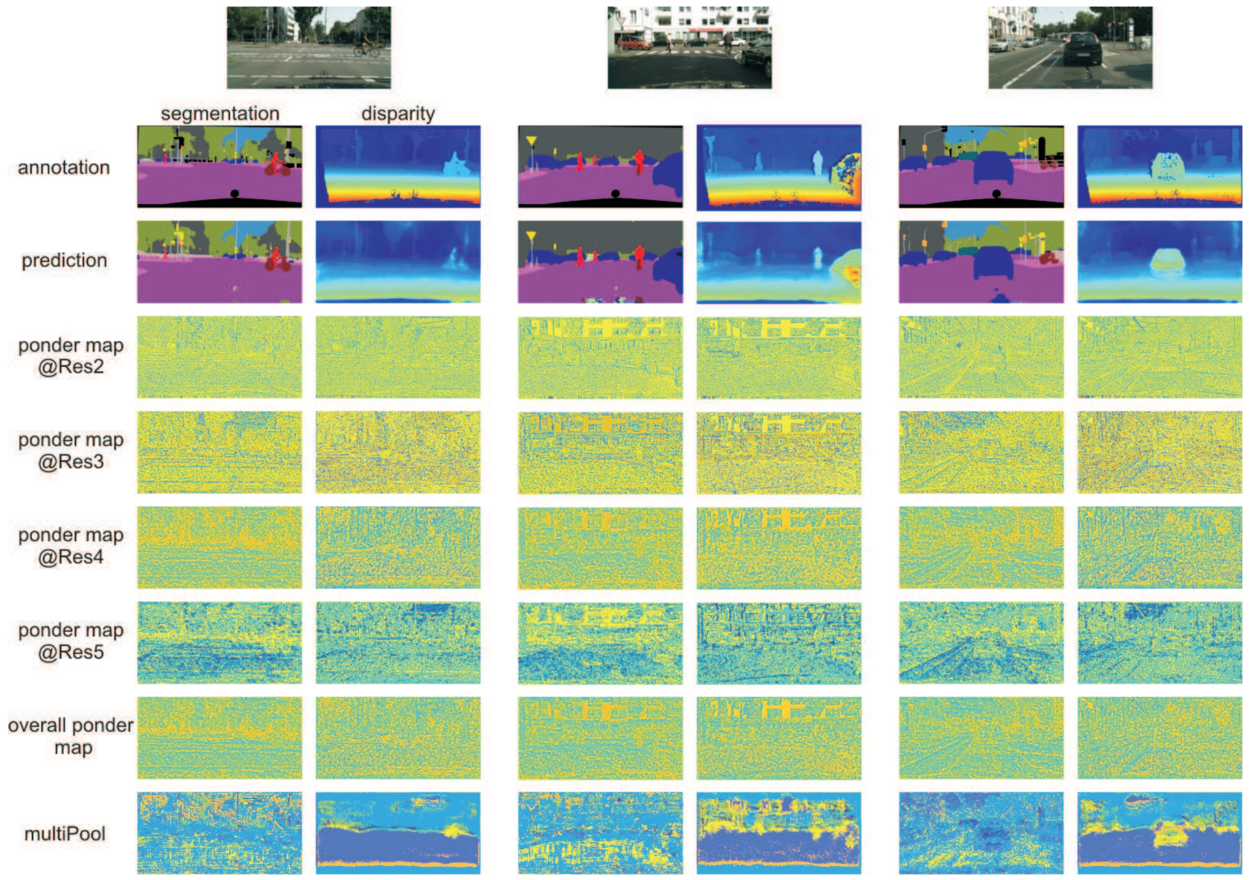


Figure B.7: Visualization on Cityscapes dataset [57] for semantic segmentation and depth estimation. Besides the overall ponder map, we also show the partial ponder map for each macro residual block by summing the sparse binary attentional maps. The MultiPool binary masks are not included in the ponder maps.

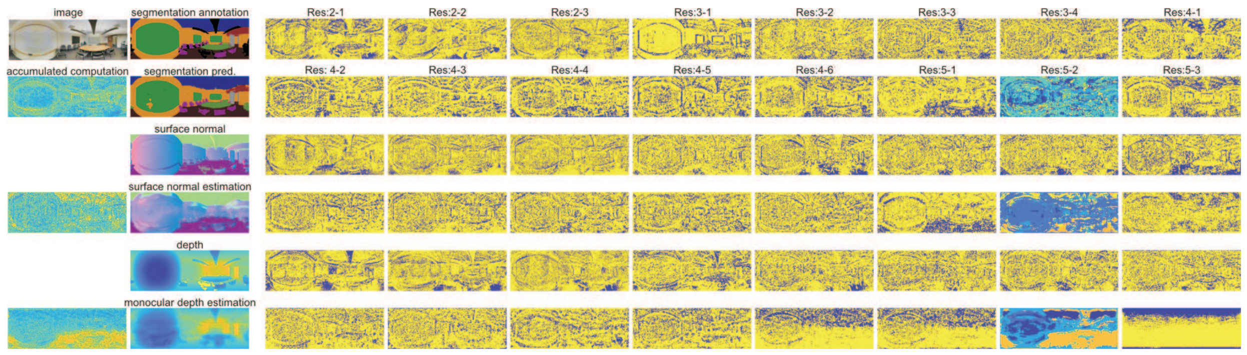


Figure B.8: Visualization on Stanford2D3D [9] for semantic segmentation, surface normal estimation and depth estimation. Besides the overall ponder map (accumulated computation), we show all the binary maps produced by PAG, as well as the one in the MultiPool module at layer 5-2.

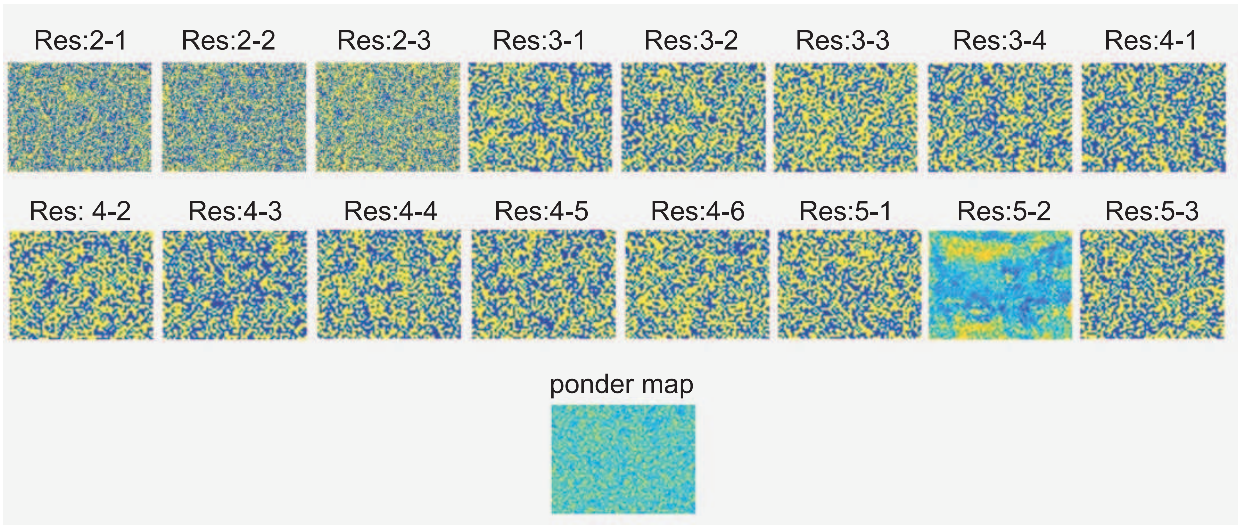


Figure B.9: Visualization on binary masks trained by PerforatedCNN [84] on NYUv2 dataset for semantic segmentation. Note that we also insert a MultiPool module at Res5-2 block. This makes it fair to compare between our method and PerforatedCNN. We also accumulate all the binary masks towards the ponder map, from which we can see that the active pixels largely follow uniform distribution. This is different from what reported in [84] that the masks mainly highlight central region in image classification, which is due to the fact that images for the classification task mainly contain object in the central region; whereas for scene images, it is hard for PerforatedCNN to focus on any specific location of the image.

Appendix C

Appendix to Chapter 5

We provide proofs for the propositions presented in Chapter 5 which provide some analytical understanding of our proposed objective function, and the mechanism for subsequent pixel grouping mechanism.

Proposition C.1. *For n vectors $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, the total intra-pixel similarity is bounded as $\sum_{i \neq j} \mathbf{x}_i^T \mathbf{x}_j \geq -\sum_{i=1}^n \|\mathbf{x}_i\|_2^2$. In particular, for n vectors on the hypersphere where $\|\mathbf{x}_i\|_2 = 1$, we have $\sum_{i \neq j} \mathbf{x}_i^T \mathbf{x}_j \geq -n$.*

Proof. First note that $\|\mathbf{x}_1 + \cdots + \mathbf{x}_n\|_2^2 \geq 0$. We expand the square and collect all the cross terms so we have $\sum_i \mathbf{x}_i^T \mathbf{x}_i + \sum_{i \neq j} \mathbf{x}_i^T \mathbf{x}_j \geq 0$. Therefore, $\sum_{i \neq j} \mathbf{x}_i^T \mathbf{x}_j \geq -\sum_{i=1}^n \|\mathbf{x}_i\|_2^2$. When all the vectors are on the hyper-sphere, i.e. $\|\mathbf{x}_i\|_2 = 1$, then $\sum_{i \neq j} \mathbf{x}_i^T \mathbf{x}_j \geq -\sum_{i=1}^n \|\mathbf{x}_i\|_2^2 = -n$. ■ □

Proposition C.2. *If n vectors $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ are distributed on a 2-sphere (i.e. $\mathbf{x}_i \in \mathbb{R}^3$ with $\|\mathbf{x}_i\|_2 = 1, \forall i = 1 \dots n$) then the similarity between any pair is lower-bounded by $s_{ij} \geq 1 - \left(\frac{2\pi}{\sqrt{3}n}\right)$. Therefore, choosing the parameter α in the maximum margin term in objective function to be less than $1 - \left(\frac{2\pi}{\sqrt{3}n}\right)$ results in positive loss even for a perfect embedding of n instances.*

We treat all the n vectors as representatives of n different instances in the image and seek to minimize pairwise similarity, or equivalently maximize pairwise distance (referred to as Tammes's problem, or the hard-spheres problem [288]).

Proof. Let $d = \max_{\{\mathbf{x}_i\}} \min_{i \neq j} \|\mathbf{x}_i - \mathbf{x}_j\|_2$ be the distance between the closest point pair of the optimally distributed points. Asymptotic results in [106] show that, for some constant $C > 0$,

$$\left(\frac{8\pi}{\sqrt{3}n}\right)^{\frac{1}{2}} - Cn^{-\frac{2}{3}} \leq d \leq \left(\frac{8\pi}{\sqrt{3}n}\right)^{\frac{1}{2}} \tag{C.1}$$

Since $\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 = 2 - 2\mathbf{x}_i^T\mathbf{x}_j$, we can rewrite this bound in terms of the similarity $s_{ij} = \frac{1}{2}\left(1 + \frac{\mathbf{x}_i^T\mathbf{x}_j}{\|\mathbf{x}_i\|_2\|\mathbf{x}_j\|_2}\right)$, so that for any $i \neq j$:

$$1 - \left(\frac{2\pi}{\sqrt{3}N}\right) \leq s_{ij} \leq 1 - \frac{1}{4}\left(\left(\frac{8\pi}{\sqrt{3}N}\right)^{\frac{1}{2}} - CN^{-\frac{2}{3}}\right)^2 \tag{C.2}$$

Therefore, choosing $\alpha \leq 1 - \left(\frac{2\pi}{\sqrt{3}N}\right)$, guarantees that $[s_{ij} - \alpha]_+ \geq 0$ for some pair $i \neq j$. Choosing $\alpha > 1 - \frac{1}{4}\left(\left(\frac{8\pi}{\sqrt{3}N}\right)^{\frac{1}{2}} - CN^{-\frac{2}{3}}\right)^2$, guarantees the existence of an embedding with $[s_{ij} - \alpha]_+ = 0$. ■ □

C.1 Details of Recurrent Mean Shift Grouping

There are two commonly used multivariate kernels in mean shift algorithm. The first, Epanechnikov kernel [76, 49], has the following profile

$$K_E(\mathbf{x}) = \begin{cases} \frac{1}{2}c_d^{-1}(d+2)(1 - \|\mathbf{x}\|_2^2), & \text{if } \|\mathbf{x}\|_2 \leq 1 \\ 0, & \text{otherwise} \end{cases} \tag{C.3}$$

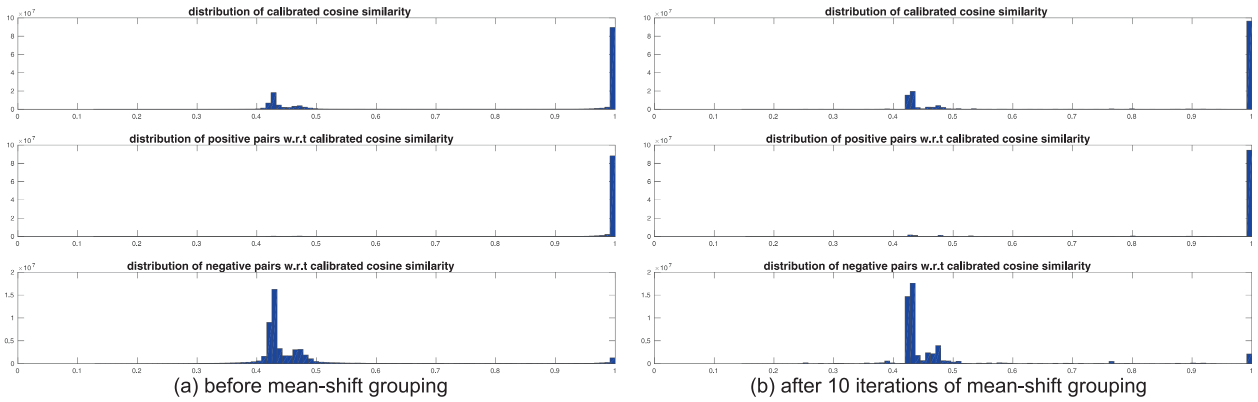


Figure C.1: Distribution of calibrated cosine similarity between pairs of pixels. After 10 iterations of mean-shift grouping. Margin is 0.5 for negative pairs. From the figures, we believe that the mean shift grouping mechanism forces learning to focus on those pixel pairs that will not be corrected by mean shift grouping itself if running offline, and thus pushing down to parameters in the the deep neural network to learn how to correct them during training.

where c_d is the volume of the unit d-dimensional sphere. The standard mean-shift algorithm computes the gradient of the kernel density estimate given by

$$p(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} K_E(\frac{\mathbf{x}-\mathbf{x}_i}{b})$$

and identifies modes (local maxima) where $\nabla p(\mathbf{x}) = 0$. The scale parameter b is known as the kernel bandwidth and determines the smoothness of the estimator. The gradient of $p(\mathbf{x})$ can be elegantly computed as the difference between $\mathbf{x}$ and the mean of all data points with $\|\mathbf{x}-\mathbf{x}_i\| \leq b$, hence the name “mean-shift” for performing gradient ascent.

Since the Epanechnikov profile is not differentiable at the boundary, we use the squared exponential kernel adapted to vectors on the sphere:

$$K(\mathbf{x}, \mathbf{x}_i) \propto \exp(\delta^2 \mathbf{x}^T \mathbf{x}_i) \tag{C.4}$$

which can be viewed as a natural extension of the Gaussian to spherical data (known as the von Mises Fisher (vMF) distribution [85, 16, 233, 159]). In our experiments we set the bandwidth δ based on the margin α so that $\frac{1}{\delta} = \frac{1-\alpha}{3}$.

Our proposed algorithm also differs from the standard mean-shift clustering (i.e., [55]) in that rather than performing gradient ascent on a fixed kernel density estimate $p(\mathbf{x})$, at every iteration we alternate between updating the embedding vectors $\{\mathbf{x}_i\}$ using gradient ascent on $p(\mathbf{x})$ and re-estimating the density $p(\mathbf{x})$ for the updated vectors. This approach is termed Gaussian Blurring Mean Shift (GBMS) in [39] and has converge rate guarantees for data which starts in compact clusters.

In Chapter 5 we visualized embedding vectors after GBMS for specific examples. Figure C.1 shows aggregate statistics over a collection of images (in the experiment of instance segmentation). We plot the distribution of pairwise similarities for positive and negative pairs during forward propagation through 10 iterations. We can observe that the mean shift module produces sharper distributions, driving the similarity between positive pairs to 1 making it trivial to identify instances.

C.1.1 Gradient Calculation for Recurrent Mean Shift

To backpropagate gradients through an iteration of GBMS, we break the calculation into a sequence of steps below where we assume the vectors in the data matrix X have already been normalized to unit length.

$$
\begin{aligned}
\boxed{\mathbf{S} =\mathbf{X}^T\mathbf{X}} \\
\boxed{\mathbf{K} = \exp(\delta^2\mathbf{S})}, \\
\boxed{\mathbf{d} =\mathbf{K}^T\mathbf{1}} \\
\boxed{\mathbf{q} =\mathbf{d}^{-1}} \\
\boxed{\mathbf{P} = (1-\eta)\mathbf{I} + \eta\mathbf{K}\mathsf{diag}(\mathbf{q})} \\
\boxed{\mathbf{Y} =\mathbf{X}\mathbf{P}}
\end{aligned}
\tag{C.5}
$$

where $\mathbf{Y}$ is the updated data after one iteration which is subsequently renormalized to project back onto the sphere. Let ℓ denote the loss and $\odot$ denote element-wise product.

Backpropagation gradients are then given by:

$$
\begin{gathered}
\boxed{\frac{\partial \ell}{\partial \mathbf{X}} = 2\mathbf{X}\frac{\partial \ell}{\partial \mathbf{S}}}\\
\boxed{\begin{aligned}
\frac{\partial \ell}{\partial \mathbf{S}} &= \delta^2 \exp(\delta^2 \mathbf{S}) \odot \frac{\partial \ell}{\partial \mathbf{K}}\\
\frac{\partial \ell}{\partial \delta} &= 2\delta \sum_{ij} \Big((s_{ij}) \odot \exp(\delta^2 s_{ij}) \odot \frac{\partial \ell}{\partial k_{ij}} \Big)
\end{aligned}}\\
\boxed{\frac{\partial \ell}{\partial \mathbf{K}} = \mathbf{1}\Big(\frac{\partial \ell}{\partial \mathbf{d}}\Big)^T}\\
\boxed{\frac{\partial \ell}{\partial \mathbf{d}} = \frac{\partial \ell}{\partial \mathbf{q}} \odot (-\mathbf{d}^{-2})}\\
\boxed{\begin{aligned}
\frac{\partial \ell}{\partial \mathbf{K}} &= \eta\Big(\frac{\partial \ell}{\partial \mathbf{P}}\Big)(\mathbf{q}\mathbf{1}^T)\\
\frac{\partial \ell}{\partial \mathbf{q}} &= \eta\Big(\frac{\partial \ell}{\partial \mathbf{P}}\Big)^T \mathbf{K}\mathbf{1}
\end{aligned}}\\
\boxed{\begin{aligned}
\frac{\partial \ell}{\partial \mathbf{X}} &= \frac{\partial \ell}{\partial \mathbf{Y}}\mathbf{P}^T\\
\frac{\partial \ell}{\partial \mathbf{P}} &= \mathbf{X}^T\frac{\partial \ell}{\partial \mathbf{Y}}
\end{aligned}}
\end{gathered}
\tag{C.6}
$$

C.1.2 Toy Example of Mean Shift Backpropagation

In Chapter 5 we show examples of the gradient vectors backpropagated through recurrent mean shift to the initial embedding space. Backpropagation through this fixed model modulates the loss on the learned embedding, increasing the gradient for initial embedding vectors whose instance membership is ambiguous and decreasing the gradient for embedding vectors that will be correctly resolved by the recurrent grouping phase.

Figure C.2 shows a toy example highlighting the difference between supervised and unsupervised clustering. We generate a set of 1-D data points drawn from three Gaussian distributions with mean and standard deviation as ($\mu = 3, \sigma = 0.2$), ($\mu = 4, \sigma = 0.3$) and ($\mu = 5, \sigma = 0.1$), respectively, as shown in Figure C.2 (a). We use mean squared error for the loss with a fixed linear regressor $y_i = 0.5 * x_i - 0.5$ and fixed target labels. The optimal embedding would set $x_i = 3$ if $y_i = 1$, and $x_i = 5$ if $y_i = 2$. We perform 30 gradient updates of the embedding vectors $x_i \leftarrow x_i - \alpha \nabla_{x_i} \ell$ with a step size α as 0.1. We analyze the behavior of Gaussian Blurring Mean Shift (GBMS) with bandwidth as 0.2.

If running GBMS for unsupervised clustering on these data with the default setting (bandwidth is 0.2), we can see they are grouped into three piles, as shown in Figure C.2 (b). If updating the data using gradient descent without GBMS inserted, we end up with three visible clusters even though the data move towards the ideal embedding in terms of classification. Figure C.2 (c) and (d) depict the trajectories of 100 random data points during the 30 updates and the final result, respectively.

Now we insert the GBMS module to update these data with different loops, and compare how this effects the performance. We show the updated data distributions and those after five loops of GBMS grouping in column (e) and (f) of Figure C.2, respectively. We notice that, with GBMS, all the data are grouped into two clusters; while with GBMS grouping they become more compact and are located exactly on the "ideal spot" for mapping into label space (i.e. 3 and 5) and achieving zero loss. On the other hand, we also observe that, even though these settings incorporates different number of GBMS loops, they achieve similar visual results in terms of clustering the data. To dive into the subtle difference, we randomly select 100 data and depict their trajectories in column (g) and (h) of Figure C.2, using a single loss on top of the last GBMS loop or multiple losses over every GBMS loops, respectively. We have the following observations:

1. By comparing with Figure C.2 (c), which depicts update trajectories without GBMS, GBMS module provides larger gradient to update those data further from their "ideal spot" under both scenarios.

2. From (g), we can see the final data are not updated into tight groups. This is because

that the updating mechanism only sees data after (some loops of) GBMS, and knows that these data will be clustered into tight groups through GBMS.

3. A single loss with more loops of GBMS provides greater gradient than that with fewer loops to update data, as seen in (g).

4. With more losses over every loops of GBMS, the gradients become even larger that the data are grouped more tightly and more quickly. This is because that the updating mechanism also incorporates the gradients from the loss over the original data, along with those through these loops of GBMS.

To summarize, our GBMS based recurrent grouping module indeed provides meaningful gradient during training with back-propagation. With the convergent dynamics of GBMS, our grouping module becomes especially more powerful in learning to group data with suitable supervision.

■ C.2 Additional Boundary Detection Results

We show additional boundary detection results on BSDS500 dataset [7] based on our model in Figure C.4, C.5, C.6 and C.7. Specifically, besides showing the boundary detection result, we also show 3-dimensional pixel embeddings as RGB images before and after fine-tuning using logistic loss. From the consistent colors, we can see (1) our model essentially carries out binary classification even using the pixel pair embedding loss; (2) after fine-tuning with logistic loss, our model captures also boundary orientation and signed distance to the boundary. Figure C.3 highlights this observation for an example image containing round objects. By zooming in one plate, we can observe a "colorful Mobius ring", indicating the embedding features for the boundary also capture boundary orientation and the signed distance to the boundary.

▢ C.3 Additional Results on Instance-Level Semantic Segmentation

We show more instance-level semantic segmentation results on PASCAL VOC 2012 dataset [77] based on our model in Figure C.8, C.9 and C.10. As we learn 64-dimensional embedding (hyper-sphere) space, to visualize the results, we randomly generate three matrices to project the embeddings to 3-dimension vectors to be treated as RGB images. Besides showing the randomly projected embedding results, we also visualize the semantic segmentation results used to product instance-level segmentation. From these figures, we observe the embedding for background pixels are consistent, as the backgrounds have almost the same color. Moreover, we can see the embeddings (e.g. in Figure C.8, the horses in row-7 and row-13, and the motorbike in row-14) are able to connect the disconnected regions belonging to the same instance. Dealing with disconnected regions of one instance is an unsolved problem for many methods, e.g. [12, 157], yet our approach has no problem with this situation.

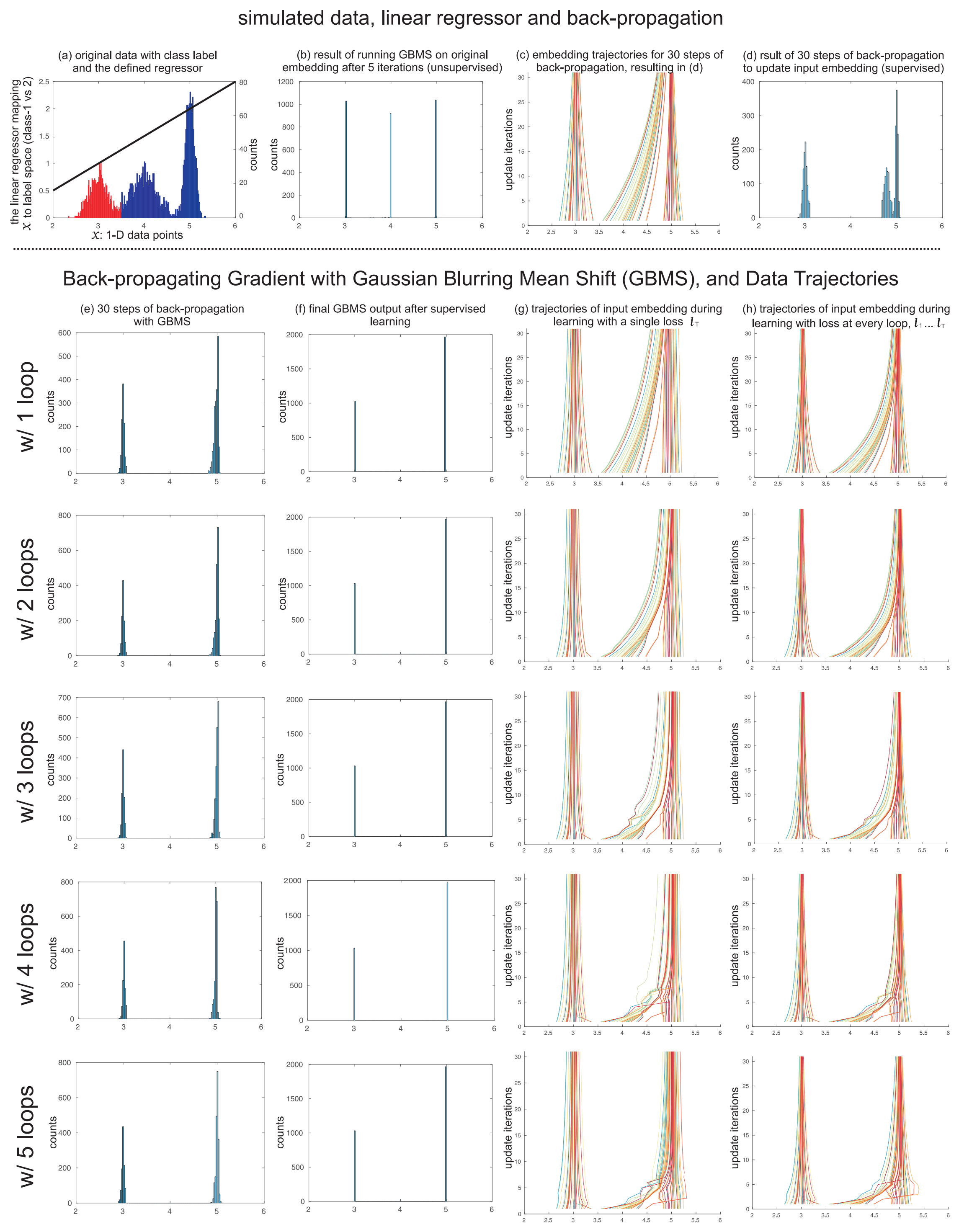


Figure C.2: Trajectory of updating data using back-propagation without mean shift module (top row), and with the Gaussian Blurring Mean Shift (GBMS). To compare the results, we vary the number of GBMS loops in the grouping module, and use either a single loss at the final GBMS loop or multiple losses on all GBMS loops. All the configurations can shift data towards the "ideal spots" (3 or 5 depending on the label) in terms of the fixed regressor.

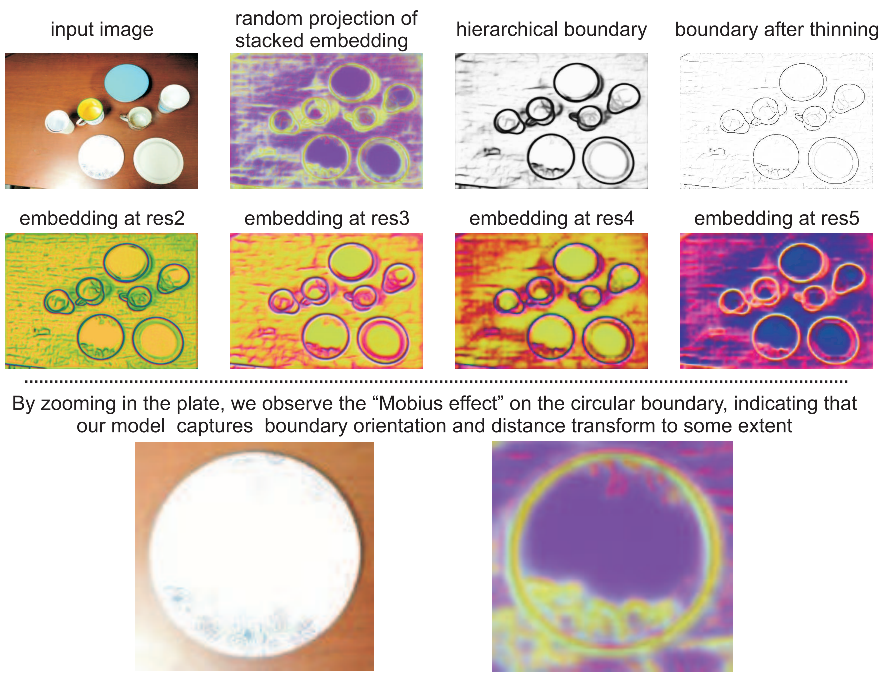

input image
random projection of stacked embedding
hierarchical boundary
boundary after thinning
embedding at res2
embedding at res3
embedding at res4
embedding at res5
By zooming in the plate, we observe the "Mobius effect" on the circular boundary, indicating that our model captures boundary orientation and distance transform to some extent

Figure C.3: An image highlighting the structure of the embedding for an image with circular boundaries. We observe a "Mobius effect" where the embedding encodes both the orientation and distance to the boundary.

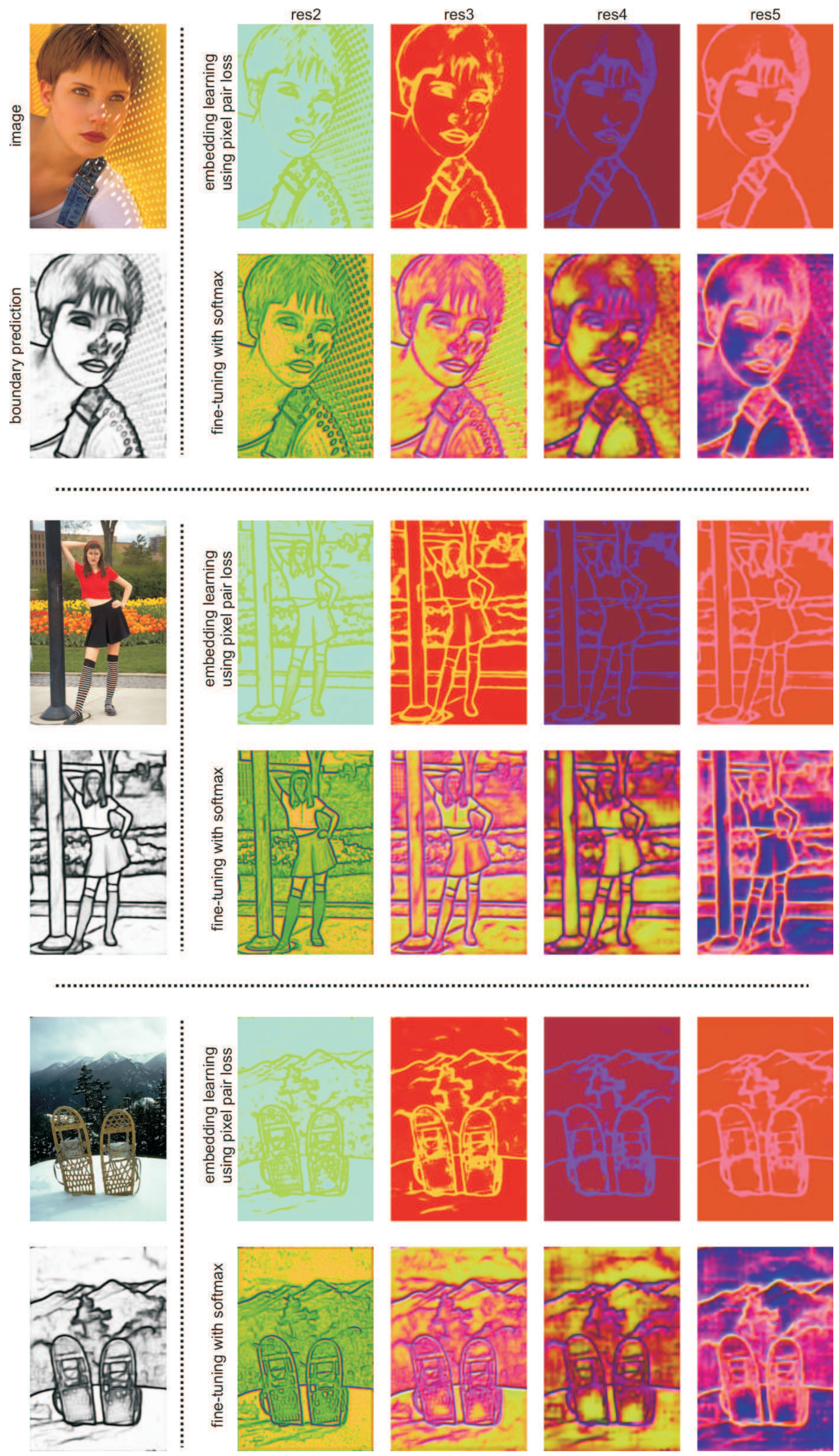


Figure C.4: Visualization for boundary detection (part-1/5). Images are randomly selected from BSDS500 test set. For each image, we show the embedding vectors at different layers from the model before and after fine-tuning using logistic loss. We can see that the boundary embedding vectors after fine-tuning not only highlights the boundary pixels, but also captures to some extent the edge orientation and distance from the colors conveyed.

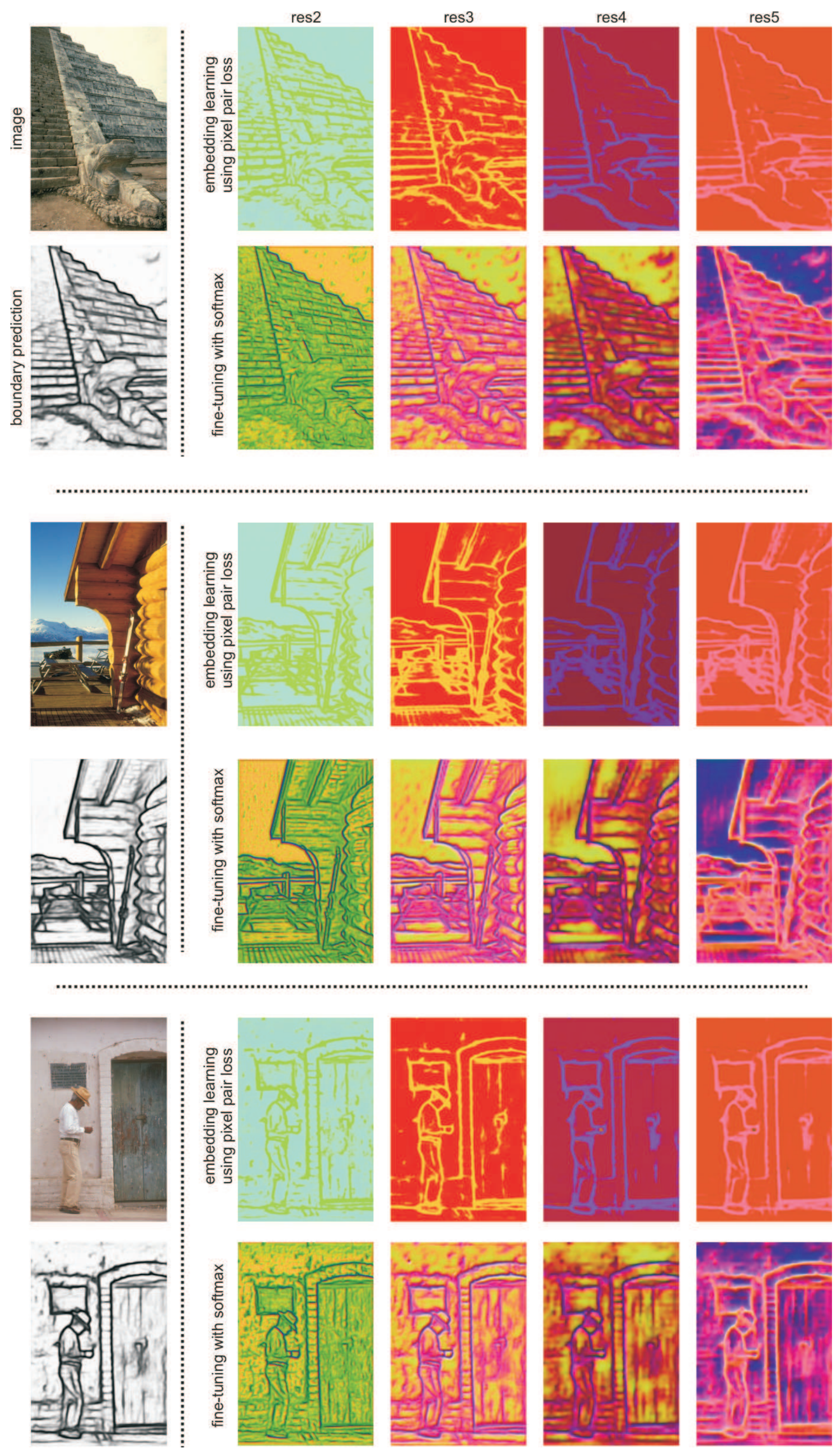


Figure C.5: Visualization for boundary detection (3/5). Images are randomly selected from BSDS500 test set. For each image, we show the embedding vectors at different layers from the model before and after fine-tuning using logistic loss. We can see that the boundary embedding vectors after fine-tuning not only highlights the boundary pixels, but also captures to some extent the edge orientation and distance from the colors conveyed.

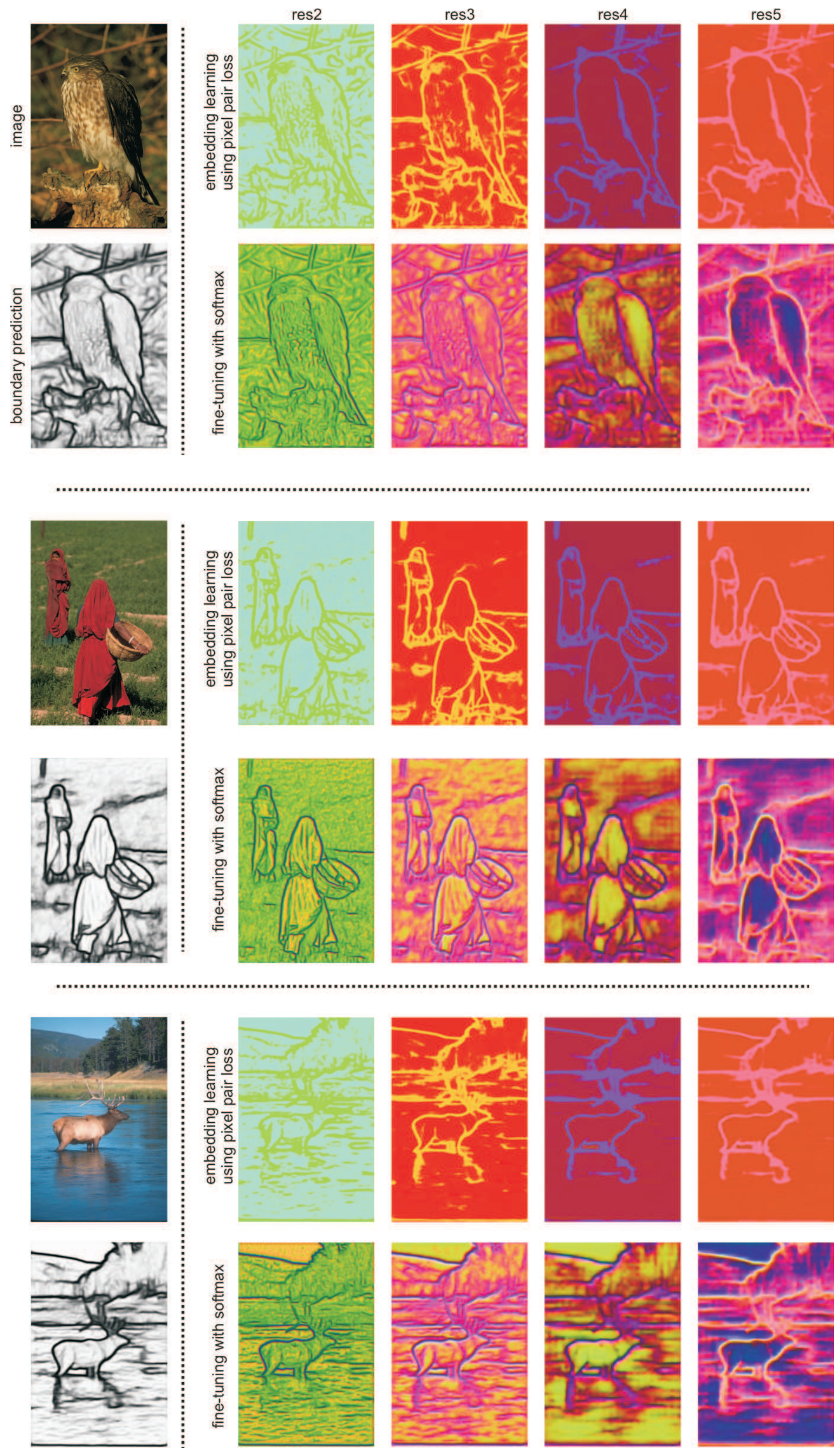


Figure C.6: Visualization for boundary detection (4/5). Images are randomly selected from BSDS500 test set. For each image, we show the embedding vectors at different layers from the model before and after fine-tuning using logistic loss. We can see that the boundary embedding vectors after fine-tuning not only highlights the boundary pixels, but also captures to some extent the edge orientation and distance from the colors conveyed.

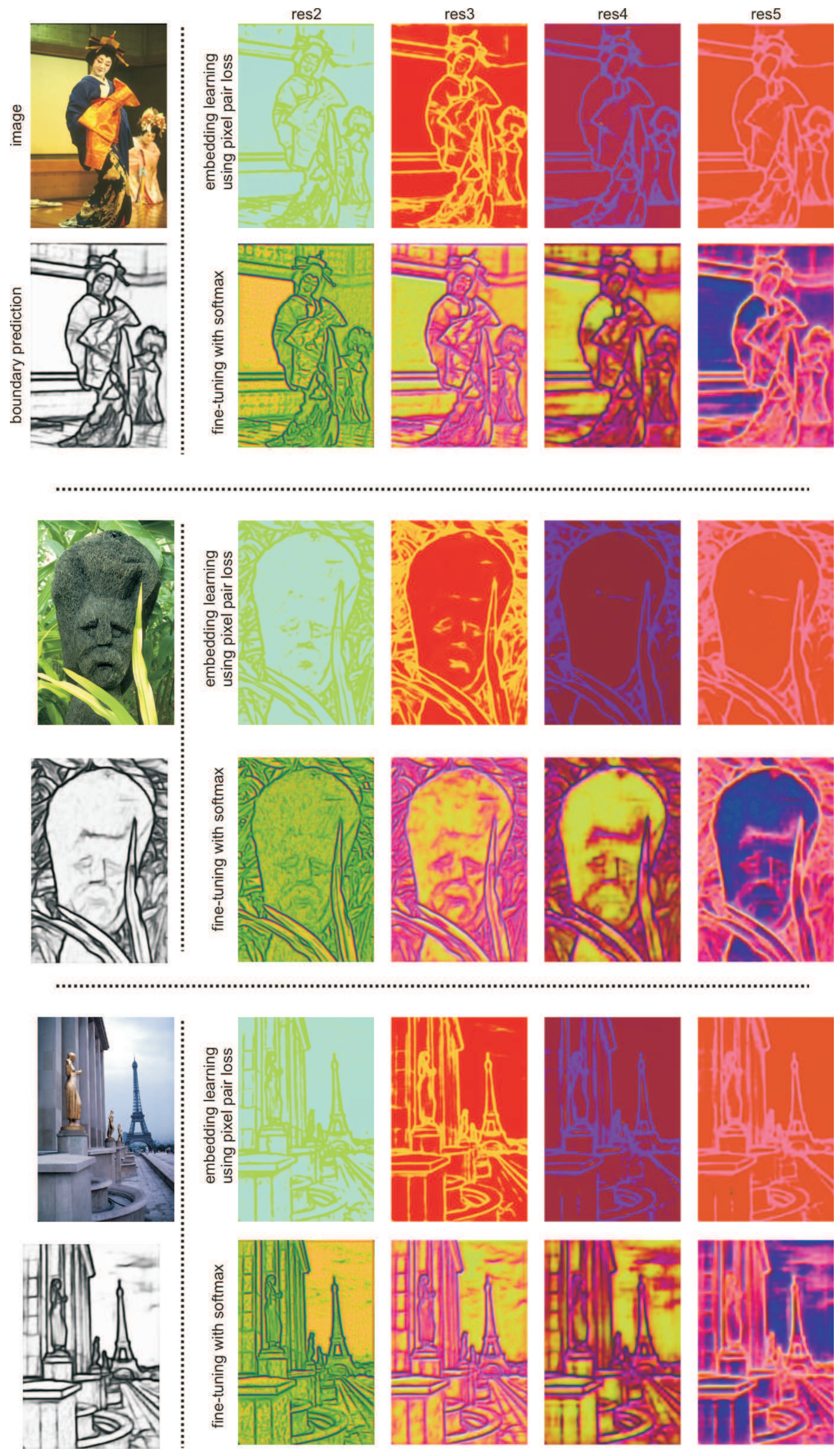


Figure C.7: Visualization for boundary detection (5/5). Images are randomly selected from BSDS500 test set. For each image, we show the embedding vectors at different layers from the model before and after fine-tuning using logistic loss. We can see that the boundary embedding vectors after fine-tuning not only highlights the boundary pixels, but also captures to some extent the edge orientation and distance from the colors conveyed.

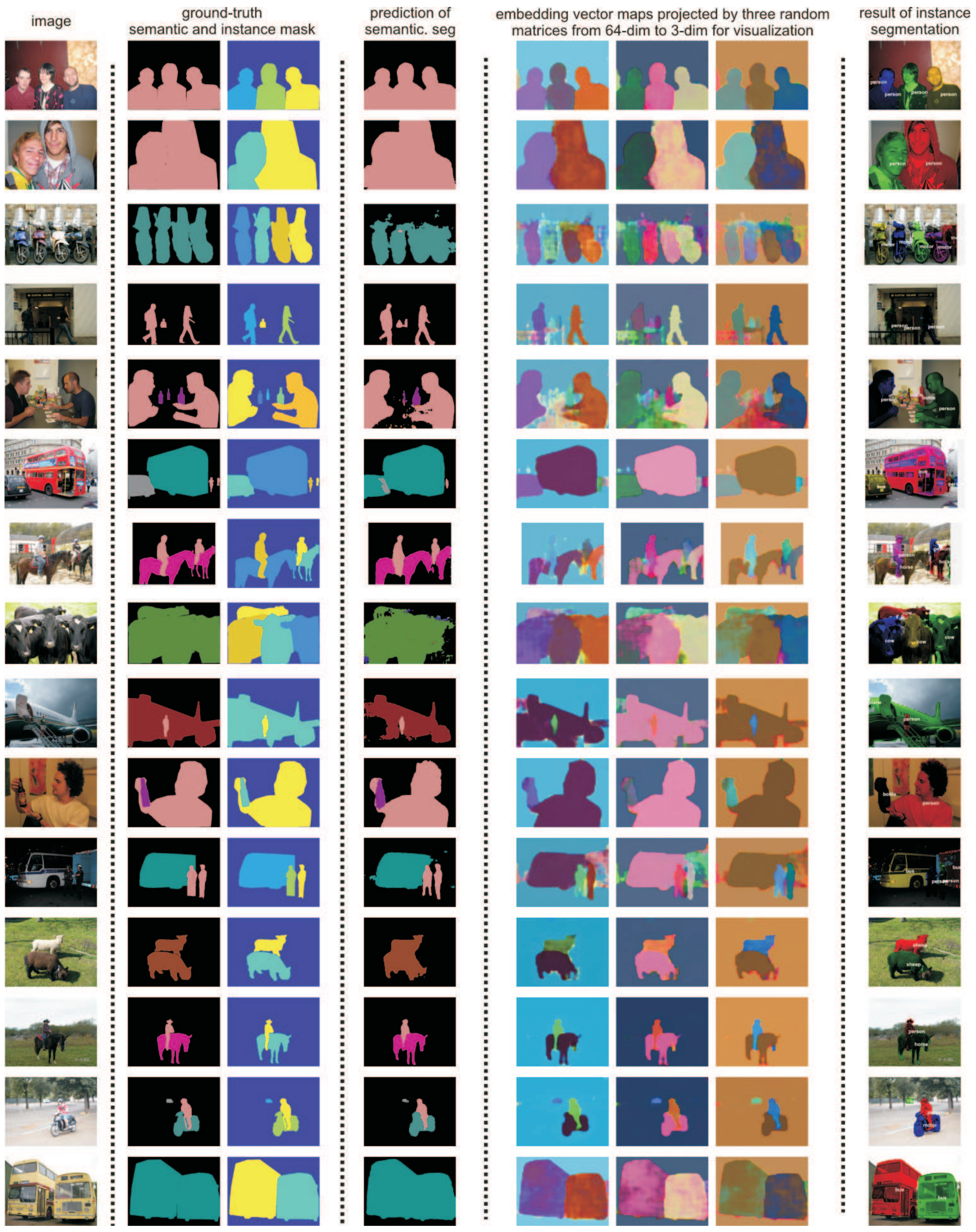


image
ground-truth semantic and instance mask
prediction of semantic. seg
embedding vector maps projected by three random matrices from 64-dim to 3-dim for visualization
result of instance segmentation

Figure C.8: Visualization of generic and instance-level semantic segmentation with random projection of the embedding vectors (part-1/3).

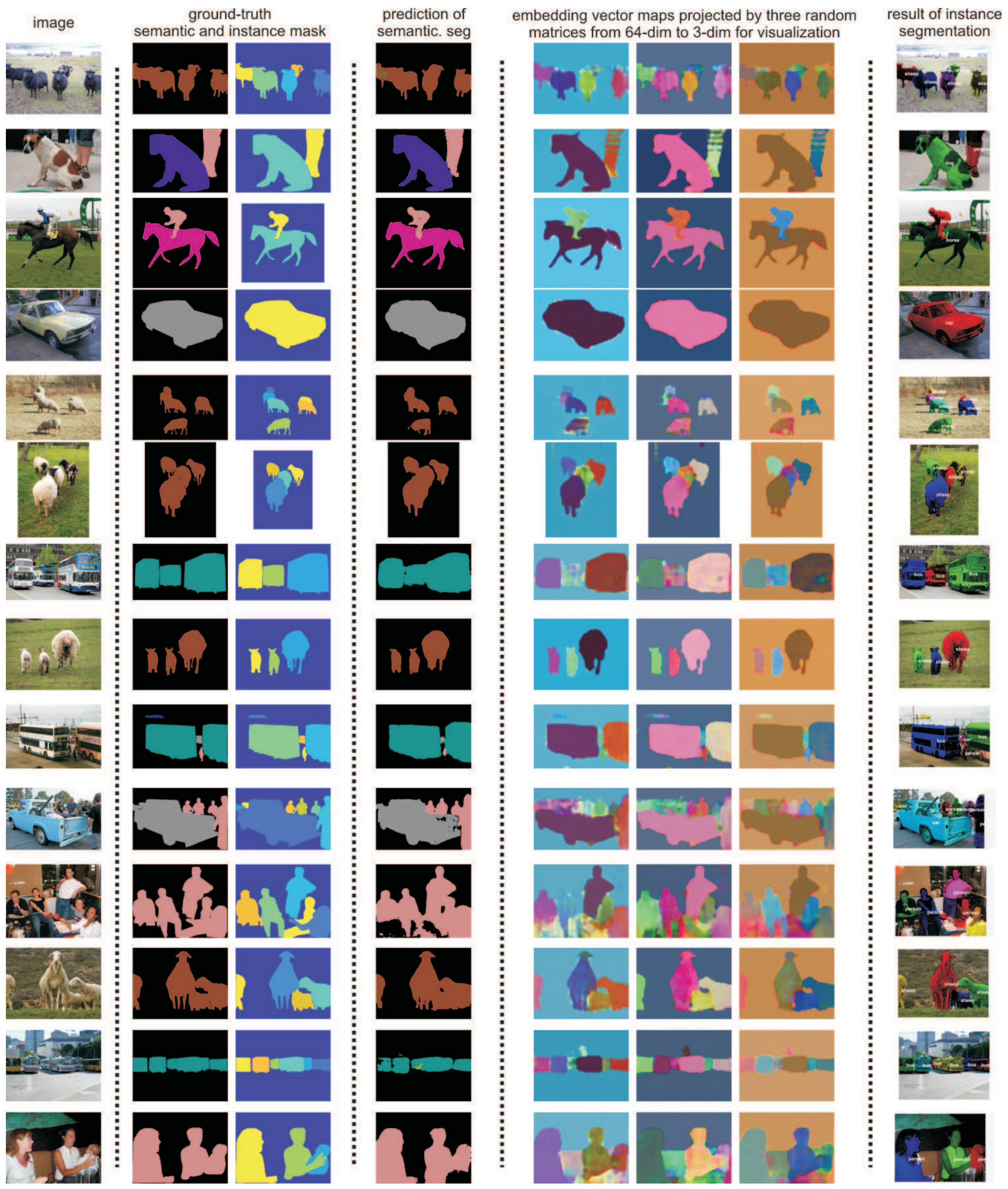


image
ground-truth
semantic and instance mask
prediction of
semantic. seg
embedding vector maps projected by three random
matrices from 64-dim to 3-dim for visualization
result of instance
segmentation

Figure C.9: Visualization of generic and instance-level semantic segmentation with random projection of the embedding vectors (part-2/3).

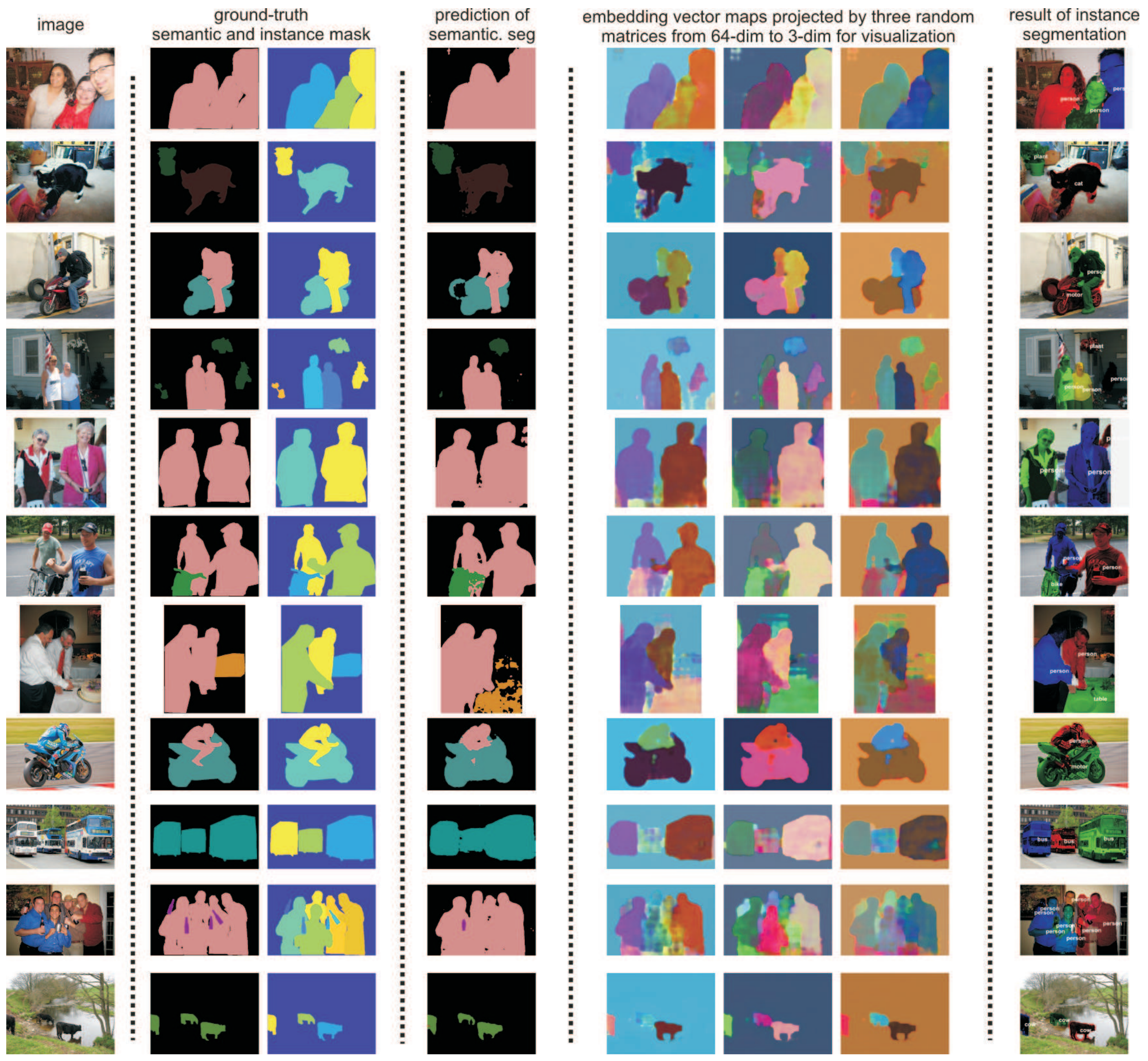


image
ground-truth
semantic and instance mask
prediction of
semantic. seg
embedding vector maps projected by three random
matrices from 64-dim to 3-dim for visualization
result of instance
segmentation

Figure C.10: Visualization of generic and instance-level semantic segmentation with random projection of the embedding vectors (part-3/3).

Appendix D

Appendix to Chapter 6

In the supplementary material, we first show more visualizations to understand the predicted filter flows, then show if it is possible to refine the results by iteratively feeding deblurred image to the same model for the task of non-uniform motion blur removal. We finally present more qualitative results for all the three tasks studied in Chapter 6.

▢ D.1 Visualization of Per-Pixel Loading Factors

As a supplementary visualization to the principal components by PCA shown in Chapter 6, we can also visualize the per-pixel loading factors corresponding to each principal component. We run PCA over testing set and show the first six principal components and the corresponding per-pixel loading factors as a heatmap in Figure D.1. With this visualization technique, we can know what region has higher response to which component kernels. Moreover, given that the first ten principal components capture $\geq 99\%$ filter energy (stated in Chapter 6), we expect future work to predict compact per-pixel filters using low-rank technique, which allows for incorporating long-range pixels through large predictive filters while with compact features (thus memory consumption is reduced largely).

Table D.1: Comparison on motion blur removal over the non-uniform motion blur dataset [11]. PFF+1 means we perform PFF one more time by taking as input the deblurred image by the same model.

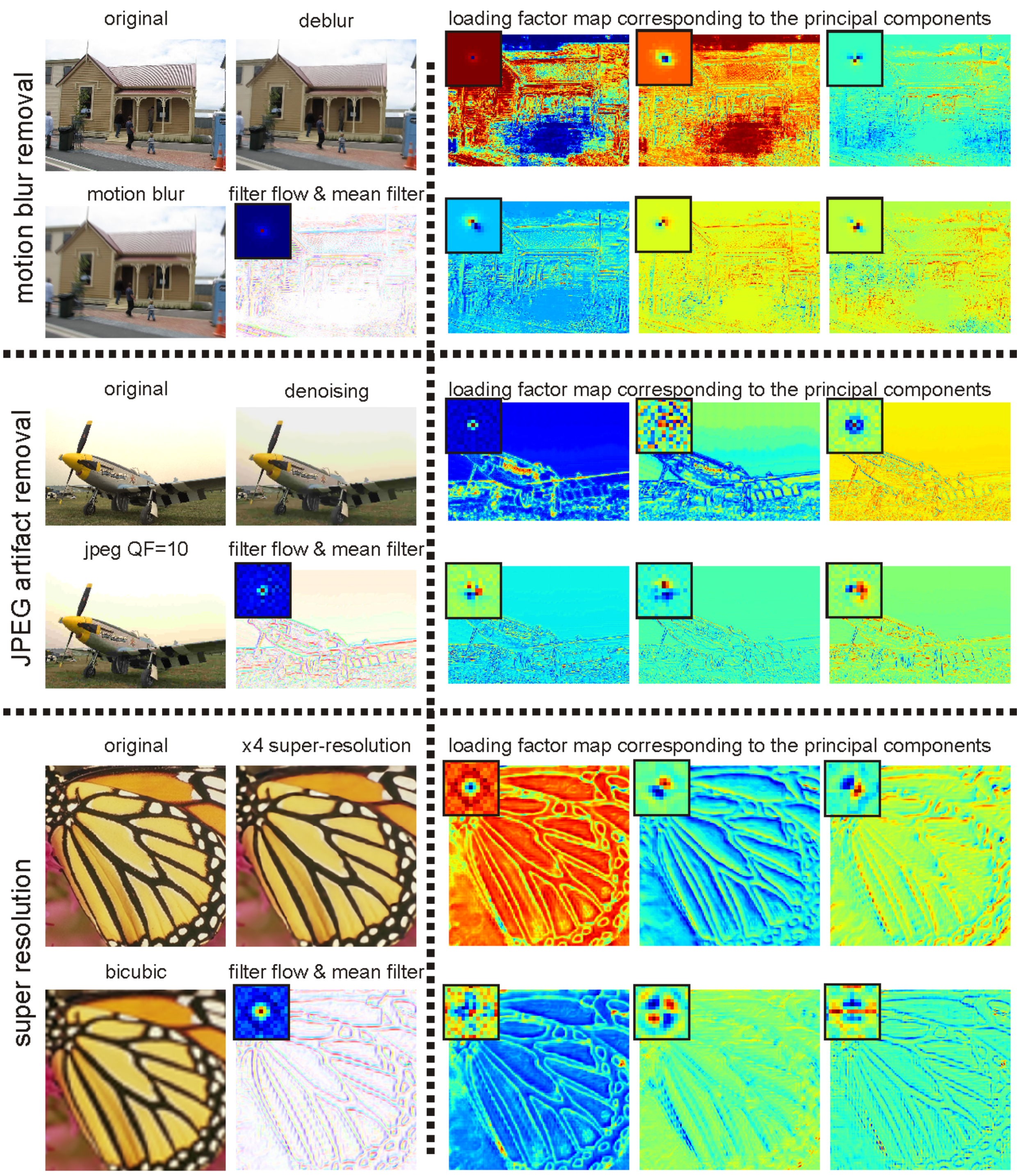
	Moderate Blur					
metric	[360]	[310]	[11]	CNN	**PFF**	PFF+1
PSNR	22.88	24.14	24.87	24.29	**25.18**	25.07
SSIM	0.68	0.714	0.743	0.708	**0.767**	0.761
	Large Blur					
metric	[360]	[310]	[11]	CNN	**PFF**	PFF+1
PSNR	20.47	20.84	22.01	20.87	**22.12**	22.00
SSIM	0.54	0.56	0.624	0.539	**0.617**	0.624

D.2 Iteratively Removing Motion Blur

As the deblurred images are still not perfect, we are interested in studying if we can improve performance by iteratively running the model, i.e., feeding the deblurred image as input to the same model one more time to get the result. We denote this method as PFF+1. Not much surprisingly, we do not observe further improvement as listed in Figure D.1, instead, such a practice even hurts performance slightly. The qualitative results are shown in Figure D.2, from which we can see the second run does not generate much change through the filter flow maps. We believe the reason is that, the deblurred images have different statistics from the original blurry input, and the model is not trained with such deblurred images. Therefore, it suggests two natural directions as future work for improving the results, 1) training explicitly with recurrent loops with multiple losses to improve the performance, similar to [20, 202, 284, 167], or 2) simultaneously inserting an adversarial loss to force the model to hallucinate details for realistic output, which can be useful in practice as done in [194].

▢ D.3 More Qualitative Results

In Figure D.3, D.4 and D.5, we show more qualitative results for non-uniform motion blur removal, JPEG compression artifact reduction and single image super-resolution, respectively. From these comparisons and with the guide of filter flow maps, we can see at what regions our PFF pays attention to and how it outperforms the other methods.



motion blur removal
original
deblur
motion blur
filter flow & mean filter
loading factor map corresponding to the principal components
JPEG artifact removal
original
denoising
jpeg QF=10
filter flow & mean filter
loading factor map corresponding to the principal components
super resolution
original
x4 super-resolution
bicubic
filter flow & mean filter
loading factor map corresponding to the principal components

Figure D.1: We show the original image, low-quality input and the high-quality output by our model as well as the mean kernel and filter flow maps on the left panel, and the first six principal components and the corresponding loading factors as heatmap on the right panel. Best seen in color and zoom-in.

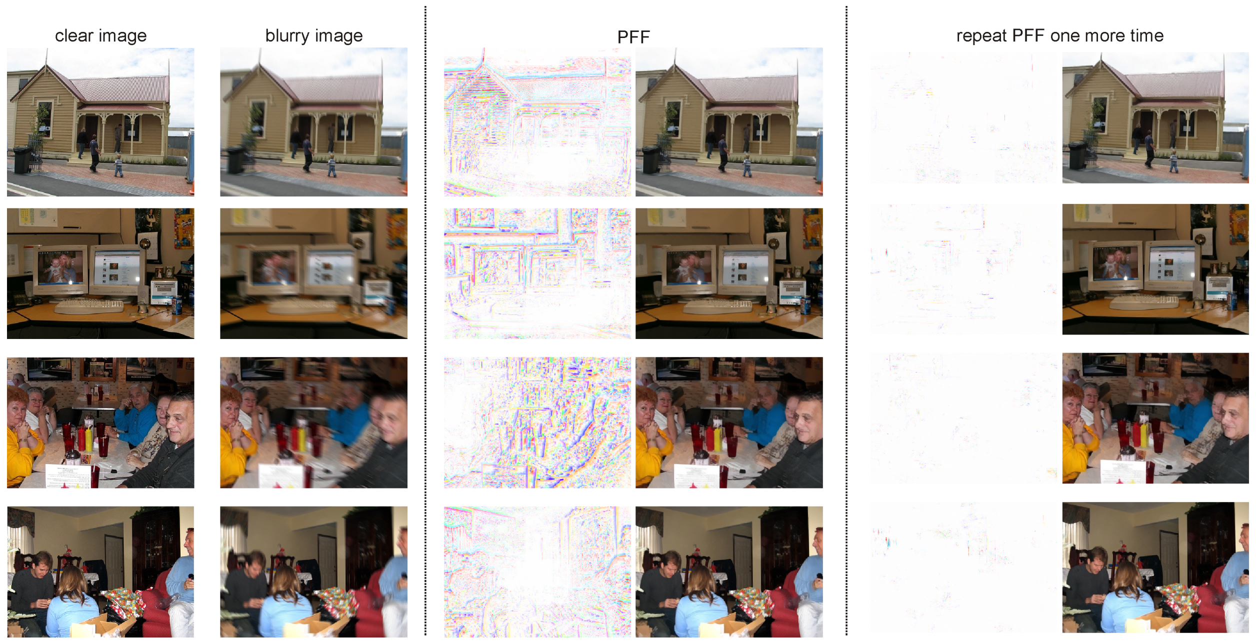


Figure D.2: We show deblurring results over some random testing images from the dataset released by [11]. We first feed the blurry images to PFF model, and obtain deblurred images; then we feed such deblurred images into the same PFF model again to see if this iterative practice refines the output. However, through the visualization that iteratively running the model changes very little as seen from the second filter flow maps. This helps qualitatively explain why iteratively running the model does not improve deblurring performance further.

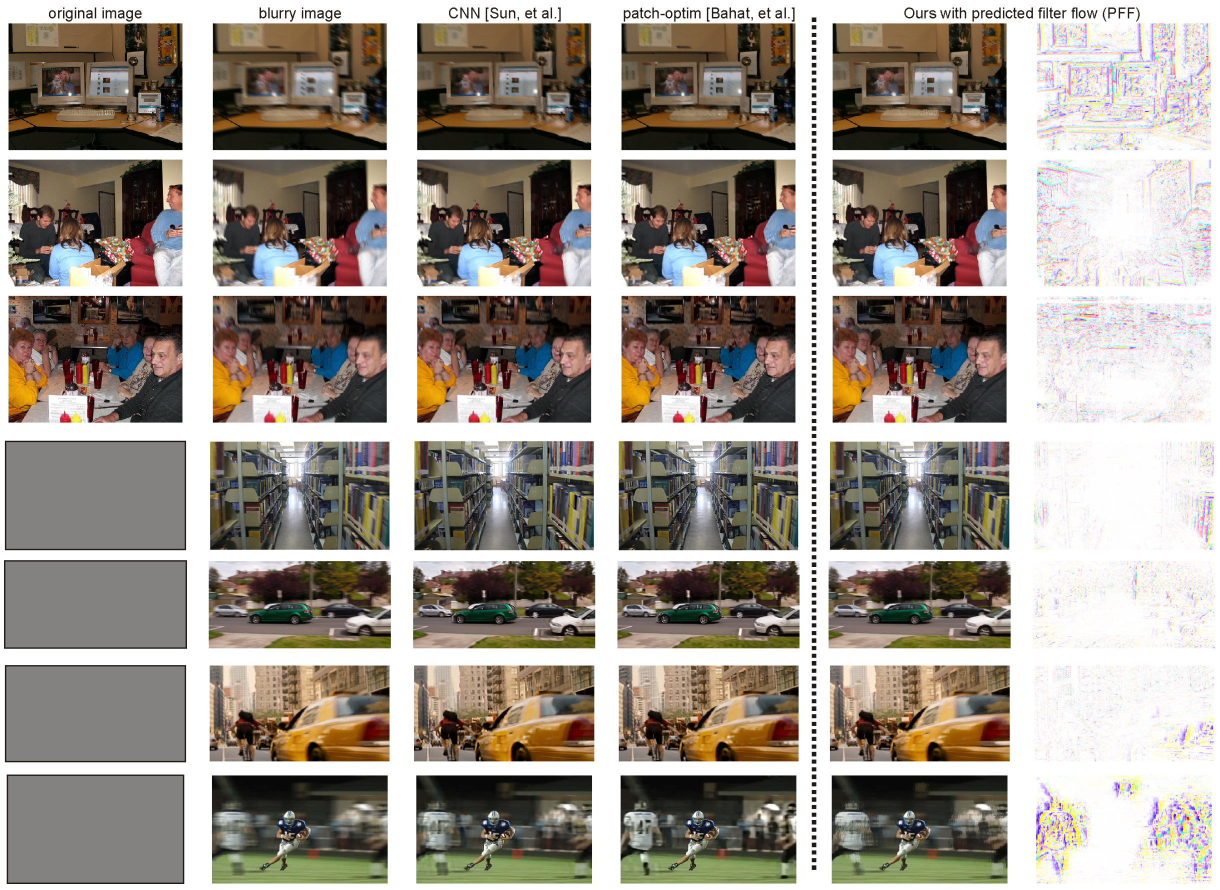

original image
blurry image
CNN [Sun, et al.]
patch-optim [Bahat, et al.]
Ours with predicted filter flow (PFF)

Figure D.3: Visual comparison of our method (*PFF*) to *CNN [Sun, et al.]* [310] and *patch-optim [Bahat, et al.]* [11] on more testing images released by [11]. Please be guided with the strong edges in the filter flow maps to compare visual details in the deblurred images by different methods. The last four rows show real-world blurry images without "ground-truth" blur. Note that for the last image, there is very large blur caused by the motion of football players. As our model is not trained on larger kernels which should be able to cover the size of blur, it does not perform as well as *patch-optim [Bahat, et al.]* [11]. But it is clear that our model generates sharp edges in this task. Best view in color and zoom-in.

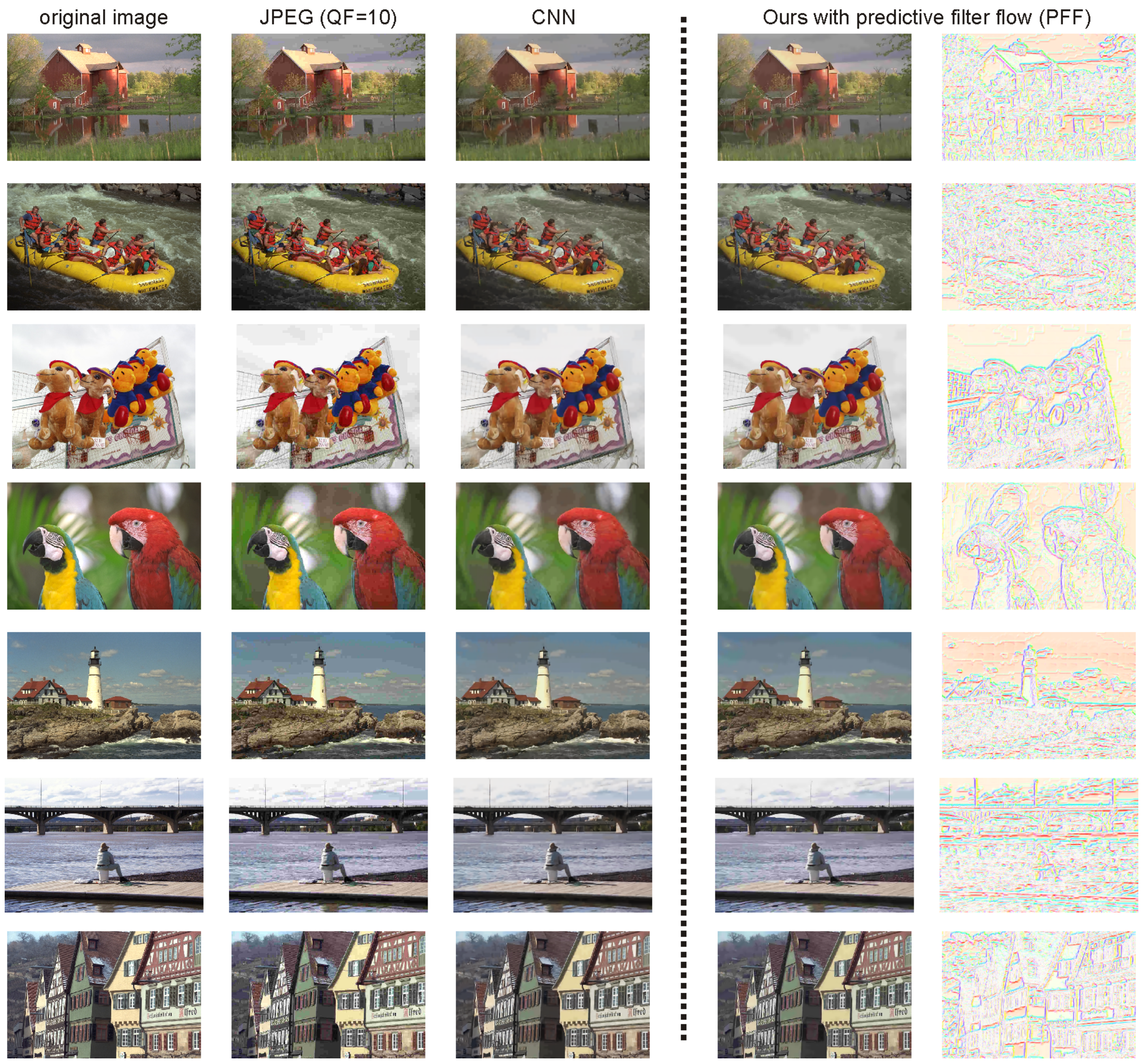


Figure D.4: Visual comparison between CNN and our method (*PFF*) for JPEG compression artifact reduction. Here we compress the original images using JPEG method with quality factor (QF) as 10. Best view in color and zoom-in.

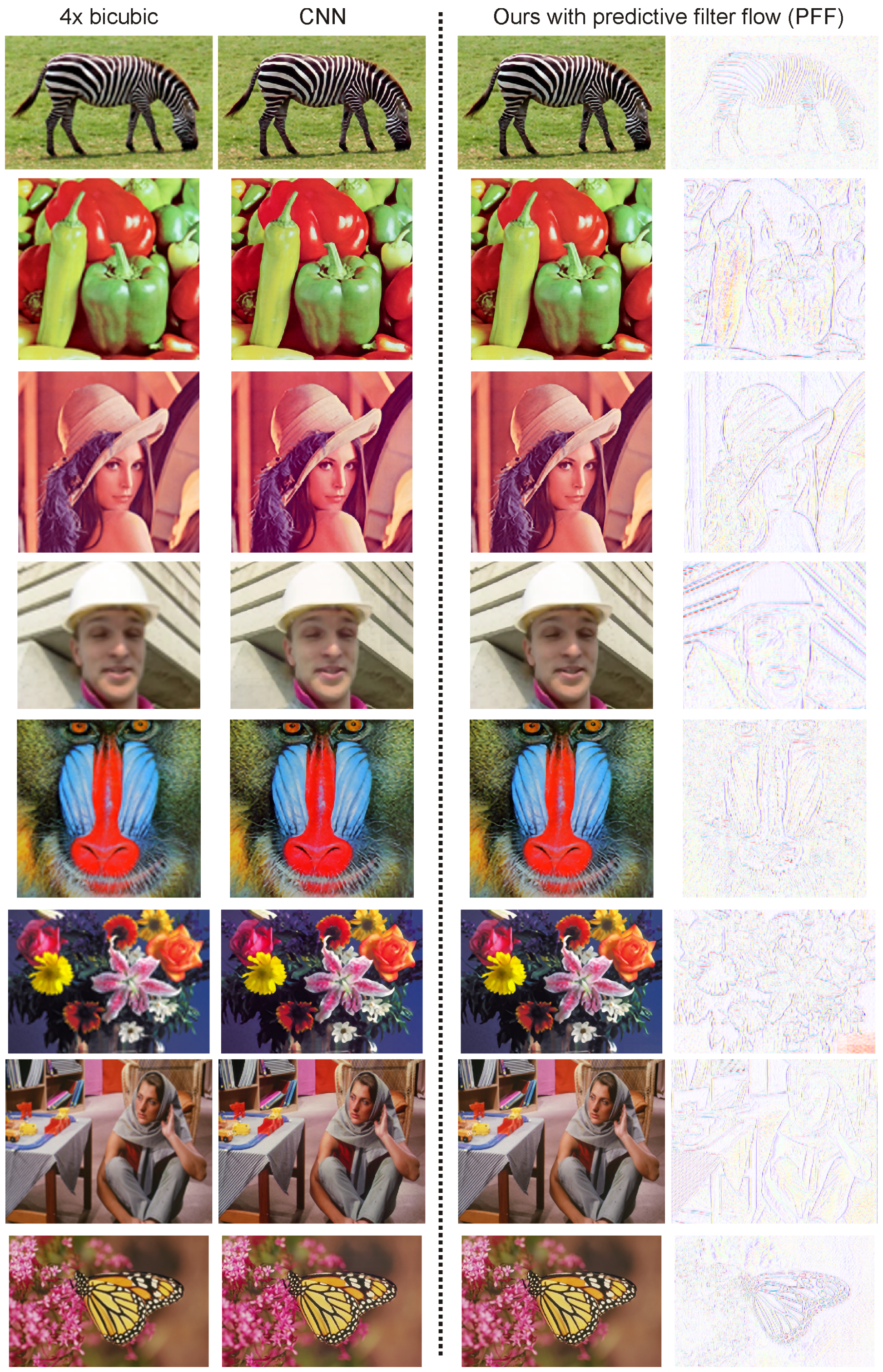


Figure D.5: Visual comparison between CNN and our method (*PFF*) for single image super-resolution. Here all images are super-resolved by $4\times$ larger. We show in the first column the results by bicubic interpolation. Best view in color and zoom-in.

Appendix E

Appendix to Chapter 7

▢ E.1 Appendix

In the appendix, we first show all intermediate results of multigrid Predictive Filter Flow (mgPFF) from multi-resolution inputs, to have an idea how these outputs look like in terms of frame reconstruction. Then, we plot the graph visualization of our model architecture with detailed design of the two stream architecture. Furthermore, we visualize pixel embedding generated by our architecture to understand what the model learns. Finally, along this document, we provide some demo videos of the object segmentation/tracking results with different setup.

▢ E.2 Intermediate Reconstruction by mgPFF

As our mgPFF performs progressively from coarse to fine, it produces the predicted filter flows and reconstruction frames at each resolution scale. We visualize all the intermediate results in Fig. E.1. We also accumulate the filter flow maps at all scales and convert it into the coordinate flow, which can be thought as optical flow. We use this coordinate flow to warp masks for propagating the track results in our experiments. Please pay attention to how mgPFF achieves excellent reconstruction results from coarse to fine, like resolving the aliasing and block effects, refining reconstruction at finer scales, etc.

▢ E.3 Graph Visualization of mgPFF architecture

In Fig. E.2, we plot the architecture of our model using the HiddenLayer toolbox [2]. As the visualization is too "long" to display, we chop it into four parts. We modify the ResNet18 [115] by removing res_4 and res_5 (the top 9 residual blocks, and reducing the unique channel size from $[64, 128, 256, 512]$ to $[32, 64, 128, 196]$. The two macro towers take the two frames, respectively; in each tower, there are two streams, one is of U-shape [285] with pooling and upsampling layers to increase the receptive fields, the other is full-resolution yet shallow in channel depth. The two-stream architecture is popular in multiple domain learning [303], but we note that such design on a single domain was first used in [269] which is more computationally expensive that the two streams talk to each other along the whole network flow; whereas ours is cheaper that they only talk at the top layer. Our mgPFF is very compact that the overall model size is only **4.6MB**; it also performs fast that the wall-clock time for processing a pair of 256x256 frames is 0.1 seconds.

As we did not search over architecture design in our work, tt is worth exploring other sophisticated modules to make it more compact for deploying in mobile devices, e.g., using meta-learning for architecture search [392].

▢ E.4 Pixel Embedding in mgPFF

As our model produces per-image pixel embeddings [249, 166] (the output before "concatenation layer" as shown in the architecture Fig. E.2), we are interested in visualizing the pixel embeddings to see what the model learns. To visualize the pixel embeddings, we use PCA to project the embedding feature map H×W×D at each resolution/grid into an H×W×3 array, and visualize the projection as an RGB image. We also concatenate the embedding maps at all the resolutions/grids for visualization (with necessary nearest neighbor upsampling). Fig. E.3 lists these visualizations, from which we can see the embedding colors largely come from the original RGB intensities. We conjecture this is due to two reasons. First, we use a simplistic photometric loss on the RGB values, this explains why the visualization colors group the pixels which have similar RGB values in local neighborhood. Second, our mgPFF by nature is based on low-level vision, i.e., flow field, and in such a way it does not necessarily depend on mid/high-level understanding of the frames. Therefore, part/instance grouping does not appear in the embedding visualization, which is shown in mid-level methods [333, 339]. This suggests further exploration of using other losses and combining other mid/high-level cues to force the model to learn more abstract features.

■ E.5 Video Demos

The attached videos demonstrate how mgPFF performs with different setup[*]. Note how it improves performance with different setup in terms of dealing with occlusion and large displacement.

Among the videos, it is worth noting how far the model can go with tracking correctly. As we adopt the multigrid computing strategy, the filter of size 11x11 on the coarsest grid (16x downsample) implies the largest displacement we can represent is D=88. If we simply warp from the first frame to the t^{th} frame, it only works well when the total displacement is less that D. This can be seen from video *soccerball, K=1, frame-*[1] as an example. When the soccerball moves further than D from its initial location at the first frame, the model suddenly fails in tracking that the mask is no longer correctly warped. We show the relevant frames in Fig. E.4. It is clear that not only the tracking is missing, but also the filter flow

[*]Here is a Youtube list

changes abruptly and the reconstruction becomes very different. It turns out that in the reconstruction, the soccerball's color is from the grass and tree trunk.

Here is the list of videos with brief description:

1. *soccerball, K=3, frame-*$[1, t-2, t-1]$: this video shows the results on soccerball from DAVIS dataset when we feed the **first frame-1 and two previous frame (**$t-2$ **and** $t-1$**)** to predict the filter flow, warp frame and track the object at current frame-t. (video url `https://youtu.be/M49nLtT1UmY`).

2. *soccerball, K=3, frame-*$[t-3, t-2, t-1]$: this video shows the results on soccerball from DAVIS dataset when we feed the **previous three frames (**$t-3$**,** $t-2$ **and** $t-1$**)** to predict the filter flow, warp frame and track the object at current frame-t (video url `https://youtu.be/q_FNk-3lh3g`).

3. *soccerball, K=2, frame-*$[1, t-1]$, this video shows the results on soccerball from DAVIS dataset when we feed the **first frame-1 and one previous frame-(**$t-1$**)** to predict the filter flow, warp frame and track the object. (video url `https://youtu.be/u6IdVS2L7-M`).

4. *soccerball, K=1, frame-*$[1]$, this video shows the results on soccerball from DAVIS dataset when we feed the **first frame only** at which the mask is given to predict the filter flow, warp frame and track the object. (video url `https://youtu.be/vsXZgdR4XEY`)

5. *soccerball, K=1, frame-*$[t-1]$, this video shows the results on soccerball from DAVIS dataset when we feed the **the previous frame-(**$t-1$) to predict the filter flow, warp frame and track the object at current frame-t. (video url `https://youtu.be/8AZ9wPF15QE`)

6. *dog, K=3, frame*-$[1, t-2, t-1]$: this video shows the results on dog from DAVIS dataset when we feed the **first frame-1 and two previous frame ($t-2$ and $t-1$)** to predict the filter flow, warp frame and track the object at current frame-t. (video url `https://youtu.be/seg5tFSMFX8`).

7. *dog, K=3, frame*-$[t-3, t-2, t-1]$: this video shows the results on dog from DAVIS dataset when we feed the **previous three frames ($t-3$, $t-2$ and $t-1$)** to predict the filter flow, warp frame and track the object at current frame-t (video url `https://youtu.be/BqM4-OctYwA`).

8. *dog, K=2, frame*-$[1, t-1]$, this video shows the results on dog from DAVIS dataset when we feed the **first frame-1 and one previous frame-($t-1$)** to predict the filter flow, warp frame and track the object. (video url `https://youtu.be/dOao8qQMsv0`).

9. *dog, K=1, frame*-$[1]$, this video shows the results on dog from DAVIS dataset when we feed the **first frame only** at which the mask is given to predict the filter flow, warp frame and track the object. (video url `https://youtu.be/xNMuMlcvfJY`)

10. *dog, K=1, frame*-$[t-1]$, this video shows the results on dog from DAVIS dataset when we feed the **the previous frame-**($t-1$) to predict the filter flow, warp frame and track the object at current frame-t. (video url `https://youtu.be/Yu5amZf1KEc`)

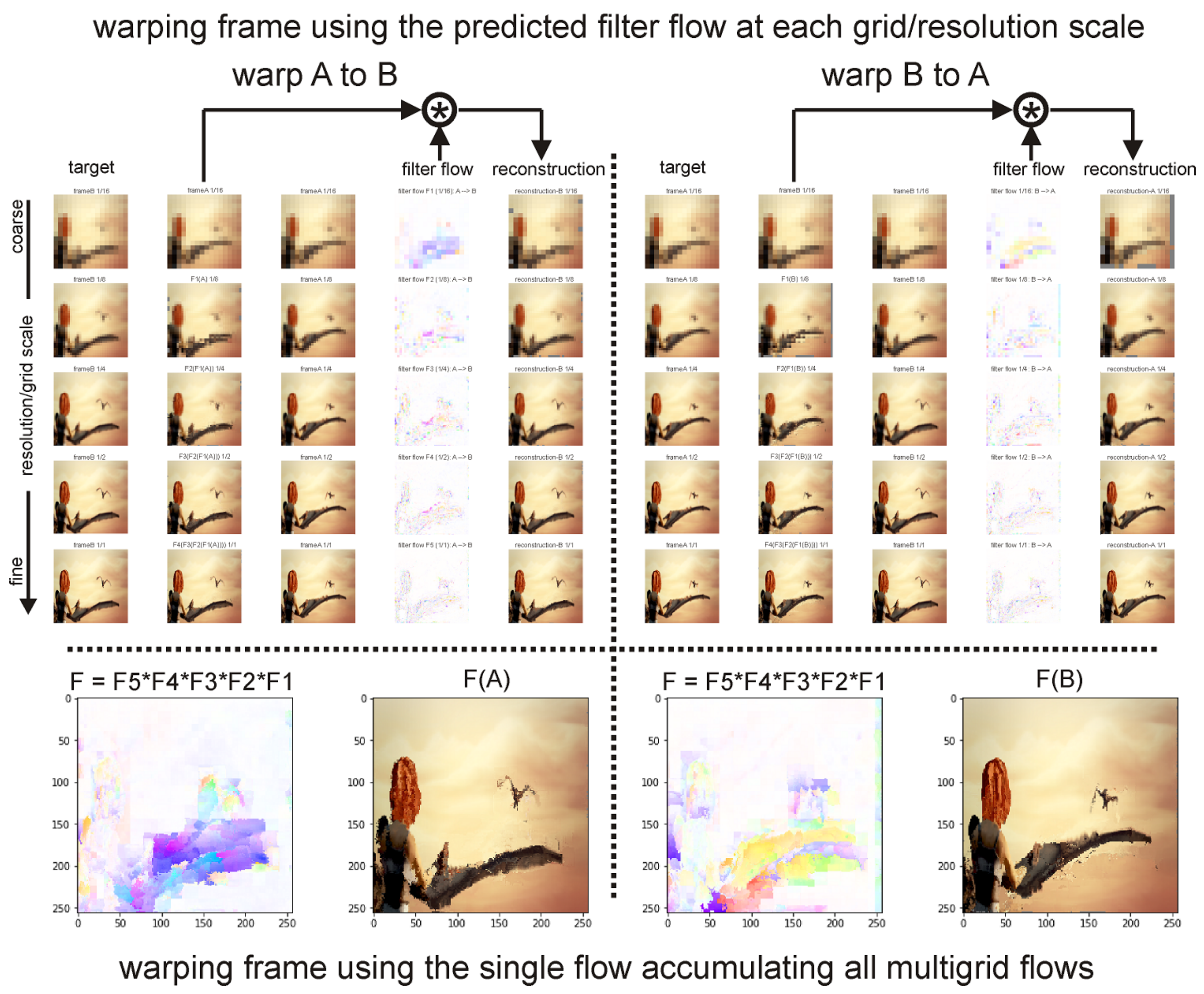

warping frame using the predicted filter flow at each grid/resolution scale
warp A to B
warp B to A
target
filter flow
reconstruction
coarse
resolution/grid scale
fine
F = F5*F4*F3*F2*F1
F(A)
F(B)
warping frame using the single flow accumulating all multigrid flows

Figure E.1: **Visualization of intermediate results at each resolution scale (grid).** Top: we show the predicted filter flows and the reconstruction results from warping A to B, or B to A. Note how mgPFF resolves the aliasing effect reflected by the blocks in the reconstruction images. Bottom: we accumulate all the filter flows (with necessary upsampling using nearest neighbor interpolation), and transform into a coordinate flow which can be thought as optical flow. Then we use the overall flow to warp from one frame to the other. This introduces some artifacts due to information loss, but the reconstruction appears good generally, e.g., capturing the bird wings' movement. In our experiment of tracking, we use the coordinate flow in the same way to warp the given masks (or the predicted mask at previous frames) to propagate the track results.

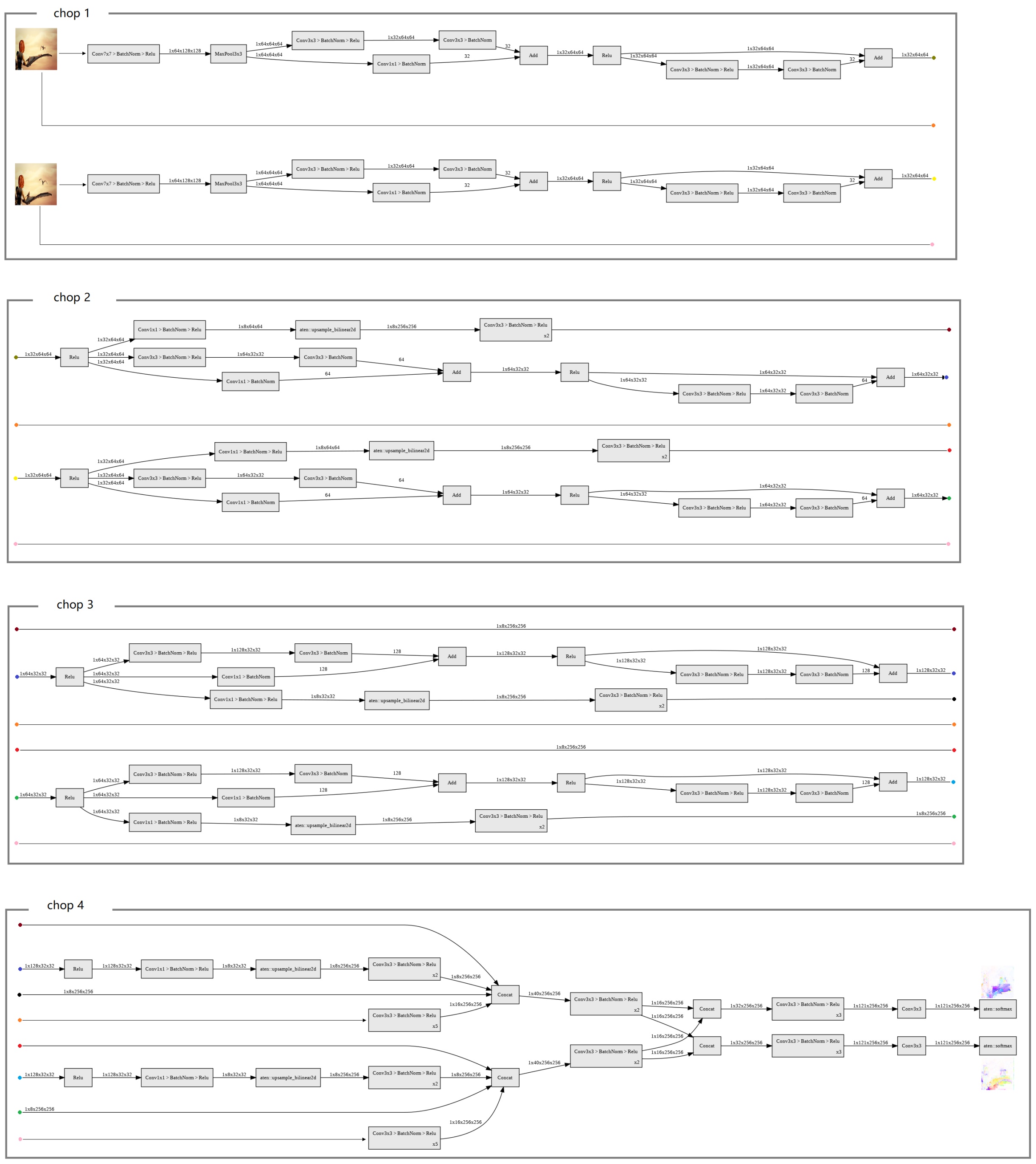


Figure E.2: Graph visualization of mgPFF architecture using HiddenLayer toolbox [2]. Zoom in to see clearly.

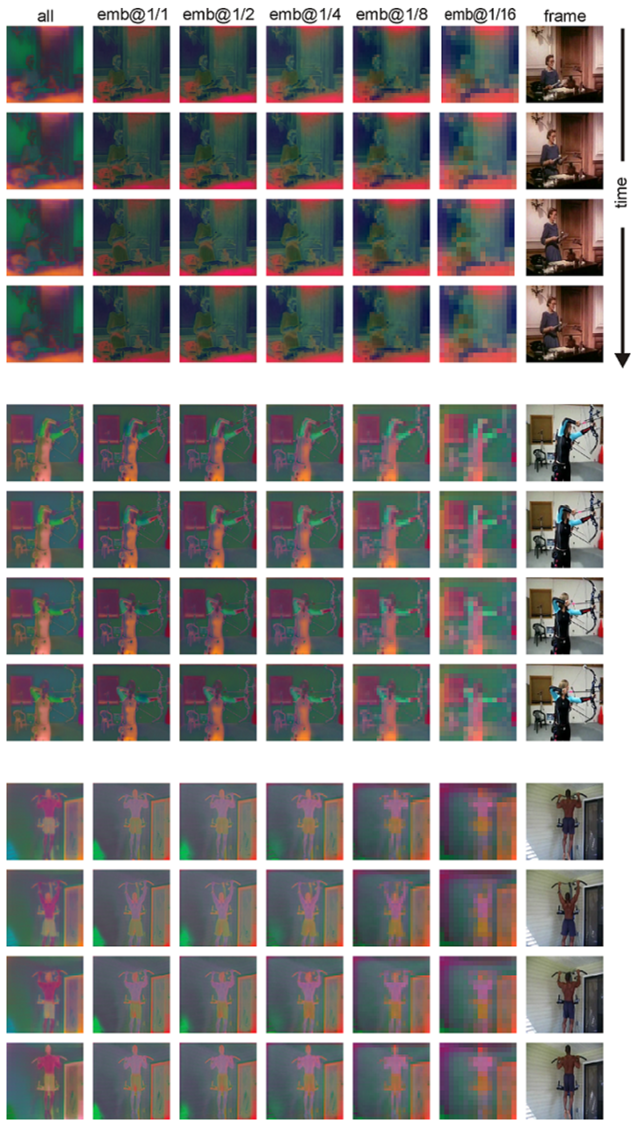


Figure E.3: **Visualization of learned pixel embedding**: We use PCA to project the pixel embedding (3D array of size H$\times$W$\times D$) into H$\times$W$\times$3, and visualize it as an RGB image. Individual embedding map has $D = 16$ in channel dimension. We also concatenate the pixel embeddings of all resolutions and apply PCA, in which case $D = 16 * 5 = 80$. From the visualization, we can see that the visualization colors largely come from the RGB intensities. This is largely due to two reasons: 1) the photometric loss we are using during training is based on RGB intensities, 2) our mgPFF by nature is based on low-level vision that it does not need understanding of mid/high-level perspective of the frames.

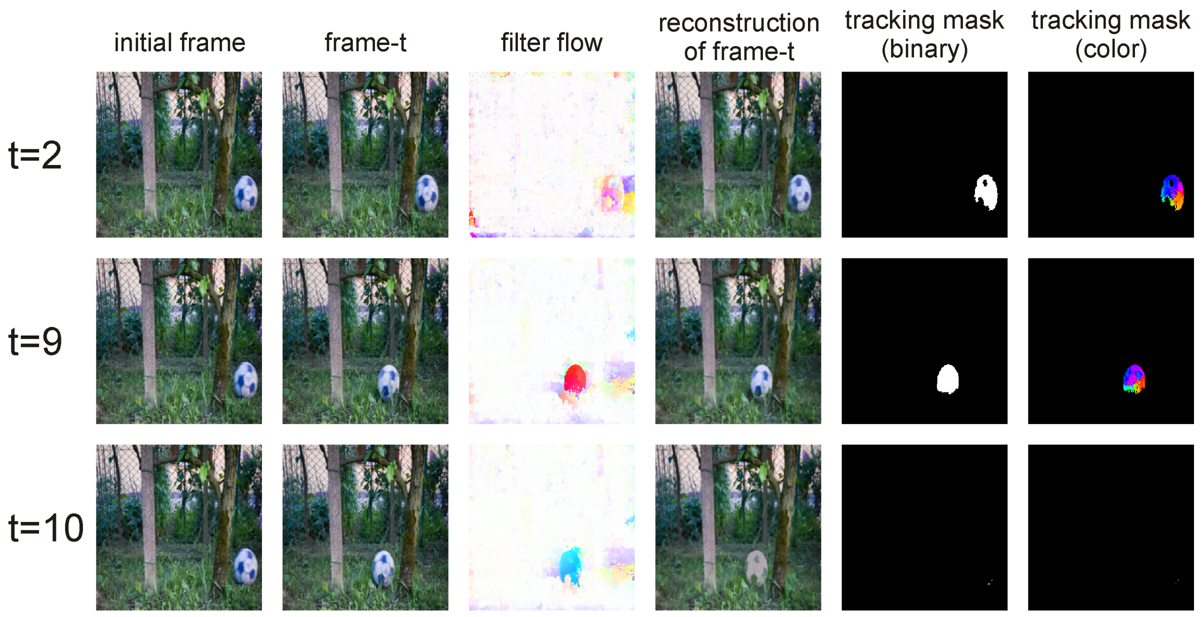


Figure E.4: How far the model can track the object correctly? As we adopt the multigrid computing strategy, the filter of size 11x11 on the coarsest grid (16x downsample) implies the largest displacement we can represent is D=88. If the object moves further than D from its last location, the model fails in tracking it. This happens at frame-10, in which we can see that not only the tracking is missing, but also the filter flow changes abruptly and the reconstruction becomes very different. It turns out that in the reconstruction, the soccerball's color is from the grass and tree trunk.