UNIVERSITY OF CALIFORNIA SAN DIEGO

Adaptive Computations and Model Structures in Object and Scene Understanding Systems

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Intelligent Systems, Robotics and Control)

by

Yongxi Lu

Committee in charge:

Professor Tara Javidi, Chair
Professor Dinesh Bharadia
Professor Kenneth Kreutz-Delgado
Professor Hao Su
Professor Nuno M. Vasconcelos
Professor Angela J. Yu

2019

DEDICATION

To my father and mother, and the random realization of the evolutionary process on planet Earth that makes us a curious species.

EPIGRAPH

If I should hear the Way in the morning, I would feel all right to die in the evening.

— Confucius

TABLE OF CONTENTS

Signature Page iii

Dedication iv

Epigraph v

Table of Contents vi

List of Figures ix

List of Tables xi

Acknowledgements xii

Vita xiv

Abstract of the Dissertation xv

Chapter 1 Introduction 1

Chapter 2 Adaptive Object Detection Using Adjacency and Zoom Prediction 5
- 2.1 Introduction 5
- 2.2 Previous Work 9
  - 2.2.1 Overview of the Adaptive Search 11
- 2.3 Design of the Algorithm 12
  - 2.3.1 Design of Building Blocks 13
  - 2.3.2 Implementation 14
- 2.4 Experiments 16
  - 2.4.1 Results on VOC 2007 16
  - 2.4.2 Quality of Region Proposals 17
  - 2.4.3 Efficiency of Adaptive Search 21
  - 2.4.4 Results on MSCOCO 21
- 2.5 Conclusion and Future Work 24

Chapter 3 Target Localization with Drones using Mobile CNNs 26
- 3.1 Introduction 26
- 3.2 Target Search Problem 28
  - 3.2.1 Visibility-Aware POMDP 30
  - 3.2.2 Control System 32
- 3.3 Simulations 33
  - 3.3.1 Advantage of Considering Structured Noise 35
  - 3.3.2 Comparing Policies 37

3.3.3 Risk of Over-Modeling 40
3.4 Data Collection And Perception Module Design 42
3.4.1 Data Collection and Model Training 42
3.4.2 Sensing Modes and Error Characteristics 45
3.5 Conclusion 47

Chapter 4 Fully-adaptive Feature Sharing in Multi-Task Networks with Applications in Person Attribute Classification 48
4.1 Introduction 48
4.2 Related Work 50
4.3 Methodology 52
4.3.1 Thin Networks and Filter Selection using Simultaneous Orthogonal Matching Pursuit 55
4.3.2 Top-Down Layer-wise Model Widening 56
4.3.3 Task Grouping based on the Probability of Concurrently Simple or Difficult Examples 58
4.3.4 Complexity-aware Width Selection 60
4.4 Experiments 61
4.4.1 Comparison with the State of the art 61
4.4.2 Understanding the Task Grouping 63
4.4.3 Cross-domain Training 64
4.4.4 Ablation Studies 64
4.4.5 Training Details 66
4.5 Conclusion and Future Works 67

Chapter 5 Implicit Label Augmentation on Partially Annotated Clips via Temporally-Adaptive Features Learning 69
5.1 Introduction 69
5.2 Methods 72
5.2.1 Temporally Adaptive Feature Learning 72
5.2.2 Efficient Frame Sampling 74
5.2.3 Application to Semantic Segmentation 76
5.3 Related Works 77
5.4 Experiments 79
5.4.1 Datasets and Evaluation Metrics 79
5.4.2 Comparison to Baselines 80
5.4.3 Ablation Studies 81
5.5 Conclusion and Future Works 84
5.6 Appendix 84
5.6.1 Temporal Change Rates of Semantic Classes 84
5.6.2 Details of Training Procedures 85
5.6.3 Choice of Temporal Change Rates 86
5.6.4 Measure Temporal Change Rates Relative to Input 87

5.6.5 Choice of Loss Functions 87

Chapter 6 Efficient Video Understanding via Layered Multi Frame-Rate Analysis . . 89
6.1 Introduction 89
6.2 Related Work 91
6.3 Methods 92
6.3.1 Two-Stream Architecture 93
6.3.2 Domain Calibration Modulator 94
6.3.3 Model Training and Loss Functions 96
6.3.4 Model Inference 97
6.4 Experiments 98
6.4.1 Dataset and Preprocessing 99
6.4.2 Network Architectures 100
6.4.3 Training and Testing Procedures 101
6.4.4 Comparison with Baselines 102
6.4.5 Ablation Studies 104
6.5 Conclusion and Future Directions 107

Chapter 7 SIGNet: Semantic Instance Aided Unsupervised 3D Geometry Perception 109
7.1 Introduction 109
7.2 Related Work 113
7.3 State-of-the-art Unsupervised Geometry Prediction 114
7.4 Methods 116
7.4.1 Semantic Input Augmentation 118
7.4.2 Semantic Guided Loss Functions 119
7.5 Experiments 121
7.5.1 Implementation Details 122
7.5.2 Monocular Depth Evaluation on KITTI 122
7.5.3 Ablation Studies 124
7.5.4 Optical Flow Estimation on KITTI 126
7.5.5 Category-Specific Metrics Evaluation 127
7.6 Conclusion 128

Chapter 8 Conclusion and Future Directions 130

Bibliography 132

LIST OF FIGURES

Figure 2.1: Comparison of our proposed adaptive search algorithm with the non-adaptive RPN method. 7
Figure 2.2: As illustrated, a given region is divided into 5 sub-regions (numbered). Each of these sub-regions is recursively divided if its zoom indicator is above a threshold. 10
Figure 2.3: Illustration of desired zoom indicator for common situations. 10
Figure 2.4: Illustration of sub-region priors. 11
Figure 2.5: Illustration of the AZ-Net architecture. 12
Figure 2.6: Illustration of the inverse matching procedure. 14
Figure 2.7: Example outputs of our algorithm. 18
Figure 2.8: Distribution of the number of anchor regions evaluated on VOC 2007 test set. 20
Figure 2.9: Comparison of recall of region proposals generated by AZ-Net and RPN at different intersection over union thresholds on VOC 2007 test. 22
Figure 2.10: Number of proposals matched to ground truth (with IoU= 0.5). 22
Figure 2.11: Comparison of recall of region proposals generated by AZ-Net and RPN at different number of region proposals on VOC 2007 test. 23
Figure 2.12: Comparison of recall of region proposals generated by AZ-Net and RPN for objects of different sizes on VOC 2007 test. 23

Figure 3.1: The control interface. 32
Figure 3.2: Example images collected in this project. 39
Figure 3.3: The field of view of the drone. 39
Figure 3.4: Two sensing modes. 40
Figure 3.5: False positive and false negative rates in two sensing modes. Quadratic fittings are provided to illustrate the general trends, while the original data points are summarized as scatter points. 43
Figure 3.6: Illustration of varying maximum visibility levels caused by persistent factors in the scene. 44

Figure 4.1: Comparing the thin model with VGG-16. 54
Figure 4.2: Illustration of the widening procedure. 56
Figure 4.3: The reduction in accuracy when changing the task grouping to favor grouping of dissimilar tasks. 64
Figure 4.4: The actual task grouping in the Branch-32-2.0 model on CelebA. Upper: fc7 layer. Lower: fc6 layer. Other layers are omitted. 65
Figure 4.5: Comparison of training progress with and without SOMP initialization. The model using SOMP initialization clearly converges faster and better. . . . 66

Figure 5.1: The sampling procedure of TAF learning for a single pair of images, in our implementation for semantic segmentation where Φ_i are defined on features extracted from a shared backbone. 75

Figure 5.2: Effect of varying the change rate. 81
Figure 5.3: Effect of varying temporal contexts 81
Figure 5.4: Prediction mIOU as a function of feature swapping. 82
Figure 5.5: Visualization of predictions with feature swapping. 83
Figure 5.6: Change in prediction IOU after attenuating the stride-32 features. 84
Figure 5.7: Changing rate of each class in Camvid train set. The figures show the ratio of prediction accuracy on x_t, between results using features from $x_{t+\Delta}$ and x_t, measured in per-class IOU. 85

Figure 6.1: The pipeline of our proposed framework, using semantic segmentation as an example. 91
Figure 6.2: Proposed modulator design in networks without BN layers (left) and with BN layers (right). 94
Figure 6.3: The sampling of training batches. 96
Figure 6.4: The modified convolution layer from AlexNet (left) and modified inverted residual unit from MobileNet V2 (right). 100

Figure 7.1: On the right, state-of-the-art unsupervised learning approach relies on pixel-wise information only, while SIGNet on the left utilizes the semantic information to encode the spatial constraints hence further enhances the geometry prediction. 110
Figure 7.2: Our unsupervised architecture contains DepthNet, PoseNet and ResFlowNet to predict depth, poses and motion using semantic-level and instance-level segmentation concatenated along the input channel dimension. 115
Figure 7.3: Top to bottom: RGB image, semantic segmentation, instance class segmentation and instance edge map. They are used for the full prediction architecture. The semantic segmentation provides accurate segments grouped by classes, but it fails to differentiate neighboring cars. 117
Figure 7.4: Infer semantic labels from depth predictions. The transfer function uses RGB and predicted depth as input. We experimented the variants with and without semantic input. 120
Figure 7.5: Comparisons of depth evaluations on KITTI. 123
Figure 7.6: The abs errs change as lighting condition drops. Our model can still be better than baseline even if the lighting intensity drops to 0.30 of the original ones. 125
Figure 7.7: Performance gains in depth (left) and flow (right) among different classes of dynamic objects. 128

LIST OF TABLES

Table 2.1: Comparison on VOC 2007 test set using VGG-16 for convolutional layers. 17
Table 2.2: Numbers related to the efficiency of the object detection methods listed in Table 2.1. 20
Table 2.3: The detection mAP on MSCOCO 2015 test-dev set. 24

Table 3.1: Improved Search Time using Visibility 35
Table 3.2: Time at Varying Realization of Visibility 37
Table 3.3: Acquisitions at Varying Heights 38
Table 3.4: Accuracy of Visibility Estimator 40
Table 3.5: Risk of Over-Modeling 41

Table 4.1: Comparison of accuracy, speed and compactness on CelebA test set. 62
Table 4.2: Comparison of accuracy, speed and compactness on Deepfashion test set. . 62
Table 4.3: Accuracy drop of Branch-32.2.0 compared to VGG-16 baseline, with and without initialization from pre-trained model. 65

Table 5.1: Overall results on Camvid and Cityscapes (CS) datasets. 80
Table 5.2: Hyperparameters used for main results. 86

Table 6.1: Result for AlexNet-FCN32s experiment. 101
Table 6.2: Result for MobileNet-FCN32s experiment. 102
Table 6.3: Use starting frames in a video instead of randomly selected images for DCM input. 105
Table 6.4: Effects of inputs with wrong domains. 105
Table 6.5: Effects of adding channel-wise multiplication parameters in the AlexNet. . . 106
Table 6.6: Effect of domain mismatches in DCM training. 107

Table 7.1: Monocular depth results on KITTI 2015 [1] by the split of Eigen *et al.* [2] (Our model used scale normalization.) 123
Table 7.2: Depth prediction performance gains due to different semantic sources and forms. (Scale normalization was used.) 124
Table 7.3: Each module's contribution toward performance gain from semantics. (Scale normalization was used.) 126
Table 7.4: Gains in depth prediction using our proposed Transfer Network. (**+sn**: "using scale normalization".) 126
Table 7.5: Rigid flow prediction from first stage on KITTI on non-occluded regions(Noc) and overall regions(All). 127
Table 7.6: Full flow prediction on KITTI 2015 on non-occluded regions(Noc) and overall regions(All). Results from DirFlowNetS are shown in [3] 127

ACKNOWLEDGEMENTS

This thesis is impossible without the many stimulating discussions with my advisor Professor Tara Javidi. I am grateful to her continued support and advice throughout my PhD study. My gratitude is also extended to my committee members for their invaluable suggestions.

I am fortunate to have many collaborators and mentors over the years, who have added deep insights and broad perspectives to my works. They include Svetlana Lazebnik, Rogerio Feris, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Dinesh Bharadia and Gaurav Bansal. I have learned a lot from them.

Finally, I would like to thank colleagues and fellow students at UC San Diego who I have closely collaborated with, in particular Ziyao Tang, Zeyangyi Wang, Anusha Lalitha and Yue Meng. Our close collaborations and frequent discussions have made me a better researcher.

Chapter 2, in full, is a reprint of the material that has been accepted for publication by *29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, authored by Yongxi Lu, Tara Javidi and Svetlana Lazebnik. The dissertation author was the primary investigator and author of this material.

Chapter 3, in full, is a reprint of the material that has been accepted for publication by *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2018)*, authored by Yongxi Lu, Zeyangyi Wang, Ziyao Tang and Tara Javidi. The dissertation author was the primary investigator and author of this material.

Chapter 4, in full, is a reprint of the material that has been accepted for publication by *30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, authored by Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi and Rogerio Feris. The dissertation author was the primary investigator and author of this material.

Chapter 5, in full, is a reprint of the material that has been submitted for publication to *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, authored by Yongxi Lu, Ziyao Tang and Tara Javidi. The dissertation author was the primary investigator and author of this material.

Chapter 6 is substantially based on unpublished material that is available as an arXiv preprint, authored by Ziyao Tang, Yongxi Lu and Tara Javidi. The dissertation author was a joint primary investigator and author of this material.

Chapter 7 is substantially based on material that has been accepted for publication by *32nd IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2019)*, authored by Yue Meng, Yongxi Lu, Aman Raj, Samuel Sunarjo, Rui Guo, Tara Javidi, Gaurav Bansal and Dinesh Bharadia. The dissertation author was a joint primary investigator and author of this material.

VITA

2013	B. Eng. in Electronic Engineering *Honors Research*, Hong Kong University of Science and Technology
2019	Ph. D. in Electrical Engineering (Intelligent Systems, Robotics and Control), University of California, San Diego

PUBLICATIONS

Yongxi Lu and Tara Javidi, "Efficient Object Detection for High Resolution Images", *in 53rd Annual Allerton Conference on Communication, Control and Computing*, 2015.

Yongxi Lu, Tara Javidi and Svetlana Lazebnik, "Adaptive Object Detection Using Adjacency and Zoom Prediction", *spotlight in 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi and Rogerio Feris, "Fully-adaptive Feature Sharing in Multi-Task Networks with Applications in Person Attribute Classification", *spotlight in 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Yongxi Lu, Zeyangyi Wang, Ziyao Tang and Tara Javidi, "Target Localization with Drones using Mobile CNNs", *in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

Yue Meng, Yongxi Lu, Aman Raj, Samuel Sunarjo, Rui Guo, Tara Javidi, Gaurav Bansal and Dinesh Bharadia, "SIGNet: Semantic Instance Aided Unsupervised 3D Geometry Perception", *in 32nd IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

ABSTRACT OF THE DISSERTATION

Adaptive Computations and Model Structures in Object and Scene Understanding Systems

by

Yongxi Lu

Doctor of Philosophy in Electrical Engineering (Intelligent Systems, Robotics and Control)

University of California San Diego, 2019

Professor Tara Javidi, Chair

This thesis presents a set of novel algorithms that address practical limitations in existing object and scene understanding systems. These limitations include high computational demands of the systems, the lack of unified models that can model multiple tasks accurately in an efficient manner, and the high dependency on output labels which are in many cases difficult and expensive to collect. Our strategy is as follows: We first identify important and intuitive structures in the respective problems. Then, we analyze the existing architectures and introduce additional dimensions of variations in the computational and model structures. The proposed algorithms are more "adaptive" than their respective baselines, in the sense that they can now use the new degrees of freedoms to address the limitations in the latter. These algorithms are practical because they demonstrate significant empirical successes in addressing the limitations of existing methods. They have pushed the state-of-the-art and inspired follow-up studies in designing better object and scene understanding systems for real-world challenges.

Chapter 1

Introduction

Humans can understand the visual world with little effort. The same cannot be said for man-made machines. Despite decades of scientific investigations into this very topic, the dream of fully replicating the visual recognition capabilities of human beings using artificial agents remains elusive. There is a silver lining. In the past few years, significant progress has been made in some specific instances [4–18, 18–36, 36–43, 43–45]. These include the particularly intriguing cases of assigning semantic labels or estimating the physical locations of every pixels on images. In some simpler tasks [31], state-of-the-art algorithms can reach parity or even surpass human performances. While these algorithms are still filled with limitations and failure modes, they are starting to become practical for many applications. The most notable examples include autonomous motion control for drones and cars, smart security camera systems and medical image processing.

Statistical or "data-driven" approaches are central to these promising developments. This methodology demands that the recognition problem in question can be described as a function defined from input images to output prediction labels. An output label can take various forms, depending on the task. For example, it can be an integer for image classification, a fixed-length real vector for pose estimation, or a multi-channel image itself for semantic segmentation. Regardless of the form of the outputs, it is usually assumed that there is no ambiguity in the true definition of this function for all practical purposes. This assumption ensures that a large-scale dataset can be collected and annotated. The annotated dataset can be used to define an empirical risk minimization problem. The solution of this problem amounts to selecting a function from a class of functions that are optimal w.r.t. a selection criterion. The criterion is defined by a differentiable surrogate of an underlying performance metric that quantifies the deviation from the ideal output of the underlying labeling function defined implicitly via the annotations of the collected dataset. Increasingly, "deep" convolutional neural networks (ConvNets) [31, 32] with up to a few hundred layers are used as the function classes. These architectures are differentiable w.r.t. to the model parameters, so an "end-to-end" optimization procedure using gradient-based methods, such as stochastic gradient descent and its variants are typically used as the approximate solver.

Despite the great empirical successes achieved by this paradigm, even state-of-the-art recognition systems still fall short in many aspects when used to support complex real-world applications such as autonomous driving cars. Some of the most significant limitations include: The high computational demands of the systems, the lack of unified models that can model multiple tasks accurately in an efficient manner, and the high dependency on output labels which are in many cases difficult and expensive to collect. This thesis presents a set of novel algorithms that address the aforementioned challenges, with a special focus on object and scene understanding systems. In all the instances, we reveal important structures in the respective problems then utilize them by explicitly introducing additional dimensions of variations in the computational and model structures. The proposed algorithms are in this sense an "adaptive" version of the baseline methods as the latter fail to introduce this dimension and thus cannot model these useful structures exhibited in the data distribution. These algorithms are practical because they demonstrate significant empirical successes in addressing the limitations of current state-of-the-arts.

We now give an overview of the methods covered in this thesis, their technical connections and their respective contributions.

Chapter 2 presents a novel object detection algorithm called “AZ-Net” [1]. It is built on the observation that most images have a small number of objects, yet existing algorithms fails to utilize this structure so can result in excessive computational complexity. Motivated by group testing which is computationally efficient under a similar sparsity structure, the proposed algorithm directs more computationally resources to image regions that are likely to contain more interesting image structures by recursively processing any given image in a coarse-to-fine manner, starting globally then “zoom-in” to specific sub-regions. This method is particularly interesting as the model can adjust its computational complexity based on the difficulty of the current sample. On standard datasets and evaluation metrics, this algorithm is shown to significantly reduce region processing in object detection without sacrificing accuracy. The proposed method is extended to robotics applications where we formulate the problem as Partially-observable Markov Decision Processes (POMDPs) and consider flight time and communication delays in addition to image processing time. This extension is discussed in Chapter 3. [2]

Chapter 4 discusses an algorithm that also addresses computational complexity issue [3]. The algorithm recognizes that a successful recognition system should be one that can simultaneously handle multiple tasks. However, combining tasks via separate recognition modules could result in excessive complexity. On the other hand, combining all the tasks prematurely could result in sub-optimal models due to “negative transfer”. This algorithm addresses the issue of designing efficient multi-task networks by explicitly modeling the affinity between tasks then use this to dynamically change the underlying ConvNet architecture. In particular, the proposed algorithm selects an optimal branching and merging pattern in the design of the deep ConvNets in an automatic way without manual intervention. It is shown in this work that such a design can lead to substantial reduction of computations with only minor degradation of accuracy.

[1]Materials presented in this chapter are based on [46]
[2]Materials presented in this chapter are based on [47]
[3]Materials presented in this chapter are based on [48]

Chapter 5 proposes an algorithm that can utilize partially annotated clips for frame level prediction tasks, such as semantic segmentation [4]. The proposed method is a novel way to significantly reduce labeling cost in building object and scene understanding systems. The structure identified in this work is the temporal correlation between frames in video clips. It is built on the observation that features for different types of structures can have distinct temporal change rates. The proposed strategy is to separate "fast features" from "slow features" in the model and training procedure design which then enable learning from unlabeled frames without sacrificing the ability of the ConvNet in modeling fine-grained details. It is shown that such a design can successfully utilize partially annotated clips to train better models compared to fully supervised baselines.

The idea of separating "slow features" from "fast features" can also result in computationally efficient video prediction algorithm. In Chapter 6 we discuss a design that is technically similar to Chapter 5, but in the fully supervised, video prediction setting [5]. Since features that changes slowly in time will not affect the quality of the prediction significantly after delays, the proposed system can use different multiple frame rates in the feature computation to reduce computational costs.

Chapter 7 again addresses the label efficiency problem, but from a multi-task learning and transfer learning perspective [6]. It is built on the observation that there is a significant difference in labeling cost between geometry and semantic labels. In particular, semantic labels are easier and less expensive to collect. Yet these two tasks of recognition tasks are complementary and could benefit from joint training. In recognition of this structure, the proposed SIGNet combines supervised semantic models with self-supervised geometry training and show that this leads to large improvements over previous state-of-the-arts.

[4]Materials presented in this chapter are based on [49]
[5]Materials presented in this chapter are based on [50]
[6]Materials presented in this chapter are based on [51]

Chapter 2

Adaptive Object Detection Using Adjacency and Zoom Prediction

2.1 Introduction

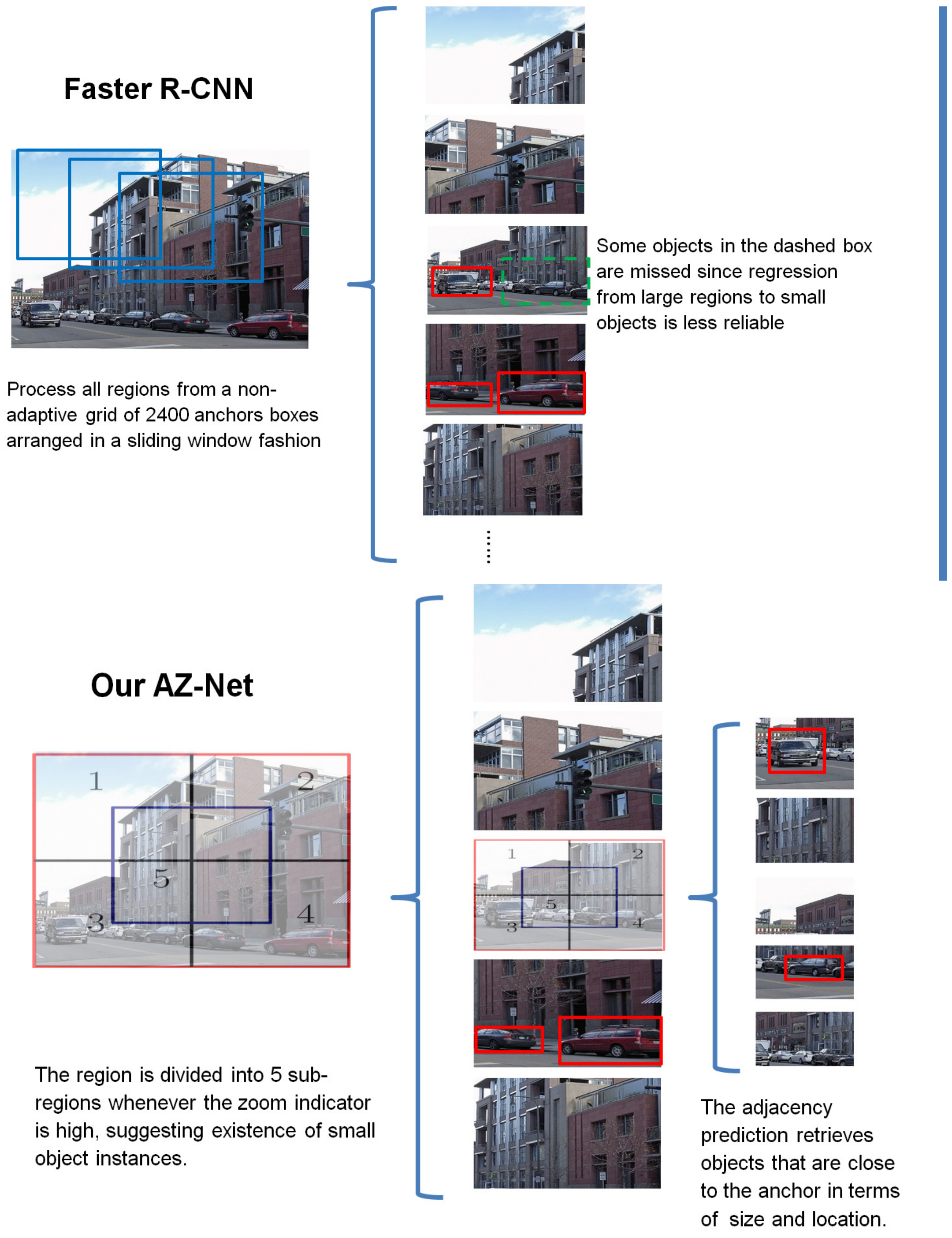
Object detection is an important computer vision problem for its intriguing challenges and large variety of applications. Significant recent progress in this area has been achieved by incorporating deep convolutional neural networks (DCNN) [31] into object detection systems [52–58].

An object detection algorithm with state-of-the-art accuracy typically has the following two-step cascade: a set of class-independent region proposals are hypothesized and are then used as input to a detector that gives each region a class label. The role of region proposals is to reduce the complexity through limiting the number of regions that need be evaluated by the detector. However, with recently introduced techniques that enable sharing of convolutional features [55, 59], traditional region proposal algorithms such as selective search [60] and EdgeBoxes [11] become the bottleneck of the detection pipeline.

An emerging class of efficient region proposal methods are based on end-to-end trained deep neural networks [52, 61]. The common idea in these approaches is to train a class-independent regressor on a small set of pre-defined anchor regions. More specifically, each anchor region is assigned the task of deciding whether an object is in its neighborhood (in terms of center location, scale and aspect ratio), and predicting a bounding box for that object through regression if that is the case. The design of anchors differs for each method. For example, MultiBox [52] uses 800 anchors from clustering, YOLO [62] uses a non-overlapping 7 by 7 grid, RPN [61] uses overlapping sliding windows. In these prior works the test-time anchors are not adaptive to the actual content of the images, thus to further improve accuracy for detecting small object instances a denser grid of anchors is required for all images, resulting in longer test time and a more complex network model.

We alternatively consider the following adaptive search strategy. Instead of fixing *a priori* a set of anchor regions, our algorithm starts with the entire image. It then recursively divides the image into sub-regions (see Figure 2.2) until it decides that a given region is unlikely to enclose any small objects. The regions that are visited in the process effectively serve as anchors that are assigned the task of predicting bounding boxes for objects nearby. A salient feature of our algorithm is that the decision of whether to divide a region further is based on features extracted from that particular region. As a result, the generation of the set of anchor regions is conditioned on the image content. For an image with only a few small objects most regions are pruned early in the search, leaving a few small anchor regions near the objects. For images that contain exclusively large instances, our approach gracefully falls back to existing methods that rely on a small number of large anchor regions. In this manner, our algorithm adaptively directs its computational resources to regions that are likely to contain objects. Figure 2.1 compares our algorithm with RPN.

To support our adaptive search algorithm, we train a deep neural network we call *Adjacency and Zoom Network (AZ-Net)*. Given an input anchor region, the AZ-Net outputs a scalar *zoom indicator* which is used to decide whether to further zoom into (divide) the region and a set


Faster R-CNN
Process all regions from a non-adaptive grid of 2400 anchors boxes arranged in a sliding window fashion
Some objects in the dashed box are missed since regression from large regions to small objects is less reliable
Our AZ-Net
1
2
5
3
4
The region is divided into 5 sub-regions whenever the zoom indicator is high, suggesting existence of small object instances.
The adjacency prediction retrieves objects that are close to the anchor in terms of size and location.

The red boxes show region proposals from adjacency predictions. Note that for small objects, RPN is forced to perform regression from much larger anchors, while our AZ-Net approach can adaptively use features from small regions.

Figure 2.1: Comparison of our proposed adaptive search algorithm with the non-adaptive RPN method.

of bounding boxes with confidence scores, or *adjacency predictions*. The adjacency predictions with high confidence scores are then used as region proposals for a subsequent object detector. The network is applied recursively starting from the whole image to generate an adaptive set of proposals.

To intuitively motivate the design of our network, consider a situation in which one needs to perform a quick search for a car. A good strategy is to first look for larger structures that could provide evidence for existence of smaller structures in related categories. A search agent could, for example, look for roads and use that to reason about where cars should be. Once the search nears the car, one could use the fact that seeing certain parts is highly predictive of the spatial support of the whole. For instance, the wheels provide strong evidence for a tight box of the car. In our design, the zoom indicator mimics the process of searching for larger structures, while the adjacency predictions mimic the process of neighborhood inference.

To validate this design we extensively evaluate our algorithm on Pascal VOC 2007 [63] with fine-grained analysis. We also report baseline results on the recently introduced MSCOCO [64] dataset. Our algorithm achieves detection mAP that is close to state-of-the-art methods at a fast frame rate. Code has been made publicly available at `https://github.com/luyongxi/az-net`.

In summary, we make the following contributions:

- We design a search strategy for object detection that adaptively focuses computational resources on image regions that contain objects.
- We evaluate our approach on Pascal VOC 2007 and MSCOCO datasets and demonstrate it is comparable to Fast R-CNN and Faster R-CNN with fewer anchor and proposal regions.
- We provide a fine-grained analysis that shows intriguing features of our approach. Namely, our proposal strategy has better recall for higher intersection-over-union thresholds, higher recall for smaller numbers of top proposals, and for smaller object instances.

This chapter is organized as follows. In section 2.2 we survey existing literature highlighting the novelty of our approach. In Section 2.3 we introduce the design of our algorithm. Section 2.4 presents an empirical comparison to existing object detection methods on standard evaluation benchmarks, and Section 2.5 discusses possible future directions.

2.2 Previous Work

Lampert *et al.*[65] first proposed an adaptive branch-and-bound approach. More recently, Gonzeles-Garcia *et al.*[66], Caicedo and Lazebnik [67], and Yoo *et al.*[68] explored active object detection with DCNN features. While these approaches show the promise of using an adaptive algorithm for object detection, their detectors are class-wise and their methods cannot achieve competitive accuracy. Our approach, on the other hand, is multi-class and is comparable to state-of-the-art approaches in both accuracy and test speed.

The idea of using spatial context has been previously explored in the literature. Previous work by Torralba *et al.*[69] used a biologically inspired visual attention model [70], but our focus is on efficient engineering design. Divvala *et al.*[71] evaluated the use of context for localization, but their empirical study was performed on hand-crafted features and needs to be reexamined in combination with more accurate recent approaches.

Our method is closely related to recent approaches that use anchor regions for proposal generation or detection. For example, Erhan *et al.*[52] use 800 data-driven anchors for region proposals and Redmon *et al.*[62] use a fixed grid of 49 non-overlapping regions to provide class-wise detections. The former has the concern that these anchors could overfit the data, while the latter cannot achieve state-of-the-art performance without model ensembles. Our work is most related to the recent work by Ren *et al.*[61], which uses a set of heuristically designed 2400 overlapping anchor regions. Our approach uses a similar regression technique to predict multiple bounding boxes from an anchor region. However, our anchor regions are generated adaptively, making them intrinsically more efficient. In particular, we show that it is possible to detect small object instances in the scene without an excessive number of anchor regions. We propose to grow a tree of finer-grained anchor regions based on local image evidence, and design the regression model strategically on top of it. We extensively compare the output of our method against [61] in our experimental section and show the unique advantages of our approach.

This chapter is a follow-up to the work published in the 53rd Annual Allerton Conference [72]. Here, we introduce a substantially improved algorithm and add extensive evaluations on standard benchmarks.

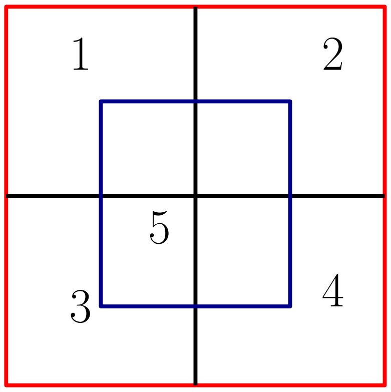
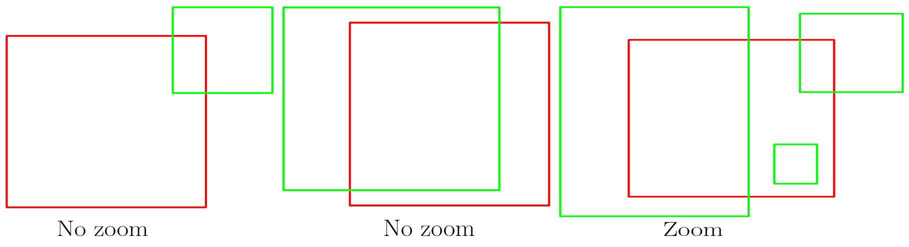
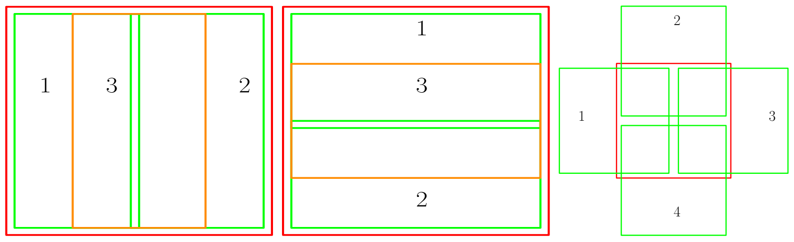


Figure 2.2: As illustrated, a given region is divided into 5 sub-regions (numbered). Each of these sub-regions is recursively divided if its zoom indicator is above a threshold.



The green boxes are objects, and the red boxes are regions. Left: the object is small but it is mostly outside the region – there is no gain in zooming in. Middle: the object is mostly inside but its size is large relative to the region – there is no gain in zooming in. Right: there is a small object that is completely inside the region. In this case further division of the region greatly increases the chance of detection for that object.

Figure 2.3: Illustration of desired zoom indicator for common situations.



From left to right: vertical stripes, horizontal stripes, neighboring squares. The red rectangular box is the image. In the figure the numbered regions are template sub-regions. The gaps between sub-regions are exaggerated for better visualization. The vertical stripes are used to detect tall objects, the horizontal stripes are used to detect fat objects, while the neighboring squares are used to detect objects that fall in the gaps between anchor regions generated in the search process.

Figure 2.4: Illustration of sub-region priors.

2.2.1 Overview of the Adaptive Search

Our object detection algorithm consists of two steps. In step 1, a set of class-independent region proposals are generated using Adaptive Search with AZ-Net (see Algorithm 1). In step 2, an object detector evaluates each region proposed in step 1 to provide class-wise detections. In our experiments the detector is Fast R-CNN.

Algorithm 1: Adaptive search with AZ-Net.

```
Data: Input image $x$ (the whole image region $b_x$). $Y_k$ is the region proposed at step $k$.
      $Y^k$ are the accumulated region proposals up to step $k$. $Z_k$ are the regions to
      further zoom in to at step $k$. $B_k$ are anchor regions at step $k$.
Result: Region proposals at termination $Y^K$.
Initialization: $B_0 \leftarrow \{b_x\}$. $Y^0 \leftarrow \emptyset$, $k \leftarrow 0$
while ($B_k$ is not an empty set) do
    Initialize $Y_k$ and $Z_k$ as empty sets.
    foreach $b \in B_k$ do
        Compute adjacency predictions $A_b$ and the zoom indicator $z_b$ using AZ-Net.
        Include all $a \in A_b$ with high confidence scores into $Y_k$.
        Include $b$ into $Z_k$ if $z_b$ is above threshold.
    end
    $Y^k \leftarrow Y^{k-1} \cup Y_k$
    $B_{k+1} \leftarrow$ Divide-Regions$(Z_k)$
    $k \leftarrow k+1$
end
$K \leftarrow k-1$
```

2.3 Design of the Algorithm

Our focus is on improving step 1. We consider a recursive search strategy, starting from the entire image as the root region. For any region encountered in the search procedure, the algorithm extracts features from this region to compute the zoom indicator and the adjacency predictions. The adjacency predictions with confidence scores above a threshold are included in the set of output region proposals. If the zoom indicator is above a threshold, this indicates that the current region is likely to contain small objects. To detect these embedded small objects, the current region is divided into sub-regions in the manner shown in Figure 2.2. Each of these sub-regions is then recursively processed in the same manner as its parent region, until either its area or its zoom indicator is too small. Figure 2.1 illustrates this procedure.

In the following section, we discuss the design of the zoom indicator and adjacency prediction.

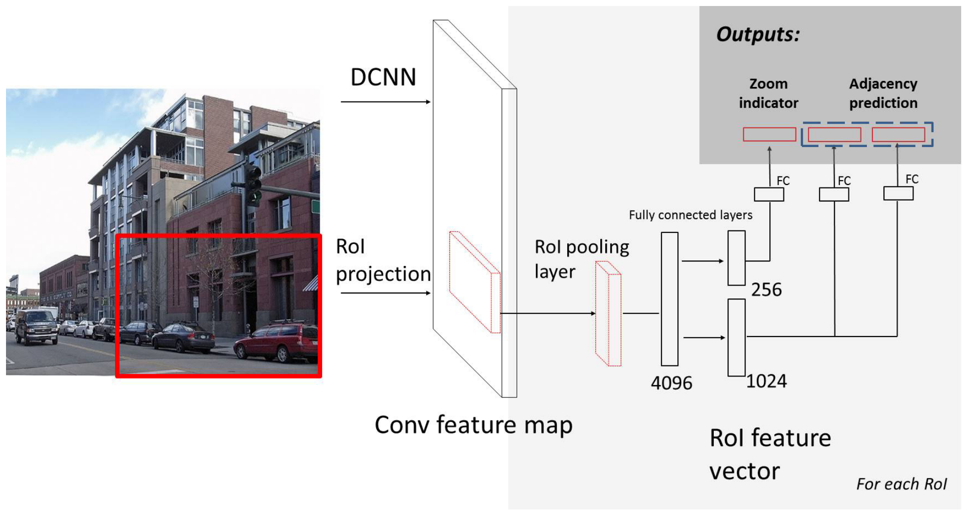
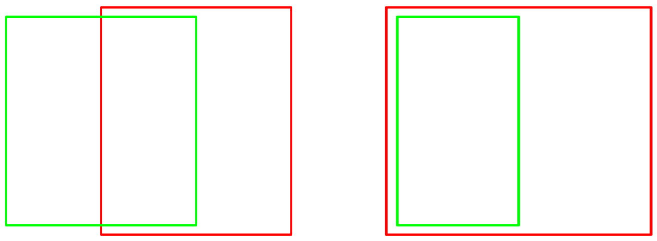


Figure 2.5: Illustration of the AZ-Net architecture.

2.3.1 Design of Building Blocks

The zoom indicator should be large for a region only when there exists at least one object whose spatial support mostly lies within the region, and whose size is sufficiently small compared to the region. The reasoning is that we should zoom in to a region only when it substantially increases the chance of detection. For example, if an object is mostly outside the region, dividing the region further is unlikely to increase the chance of detecting that object. Similarly, if an object is large compared to the current region, the task of detecting this object should be handled by this region or its parents. In the latter case, further division of the region not only wastes computational resources, but also introduces false positives in the region proposals. Figure 2.3 shows common situations and the desirable behavior of the zoom indicator.

The role of adjacency prediction is to detect one or multiple objects that overlap with the anchor region sufficiently by providing tight bounding boxes. The adjacency prediction should be aware of the search geometry induced by the zoom indicator. More specifically, the adjacency prediction should perform well on the effective anchor regions induced by the search algorithm. For this purpose we propose a training procedure that is aware of the adaptive search scheme (discussed in Section 2.3.2). On the other hand, its design should explicitly account for typical geometric configurations of objects that fall inside the region, so that the training can be performed in a consistent fashion. For this reason, we propose to make predictions based on a set of sub-region priors as shown in Figure 2.4. Note that we also include the anchor region itself as an additional prior. We make sub-region priors large compared to the anchor under the intuition that if an object is small, it is best to wait until the features extracted are at the right scale to make bounding box predictions.



The red box is the inverse match for the object (green box). The left figure shows inverse matching of a neighboring square, the right figure shows inverse matching of a vertical stripe.

Figure 2.6: Illustration of the inverse matching procedure.

2.3.2 Implementation

We implement our algorithm using the Caffe [73] framework, utilizing the open source infrastructure provided by the Fast R-CNN repository [59]. In this section we introduce the implementation details of our approach. We use the Fast R-CNN detector since it is a fast and accurate recent approach. Our method should in principle work for a broad class of object detectors that use region proposals.

We train a deep neural network as illustrated in Figure 2.5. Note that in addition to the sub-region priors as shown in Figure 2.4, we also add the region itself as a special prior region, making in total 11 adjacency predictions per anchor. For the convolutional layers, we use the VGG16 model [57] pre-trained on ImageNet data. The fully-connected layers are on top of a region pooling layer introduced in [59] which allows efficient sharing of convolutional layer features.

The training is performed as a three-step procedure. First, a set of regions is sampled from the image. These samples should contain hard positive and negative examples for both the zoom indicator and the adjacency prediction. Finally, the tuples of samples and labels are used in standard stochastic gradient descent training. We now discuss how the regions are sampled and labeled, and the loss function we choose.

Region Sampling and Labeling

Since a typical image only has a few object instances, to provide sufficient positive examples for adjacency predictions our method inversely finds regions that will see a ground truth object as a perfect fit to its prior sub-regions (see Figure 2.6 for illustration). This provides $k \times 11$ training examples for each image, where k is the number of objects.

To mine for negative examples and hard positive examples, we search the input image as in Algorithm 1. Note that the algorithm uses zoom indicators from the AZ-Net. Instead of optimizing AZ-Net with an on-policy approach (that uses the intermediate AZ-Net model to sample regions), which might cause training to diverge, we replace the zoom prediction with the zoom indicator label. However, we note that using the zoom label directly could cause overfitting, since at test time the algorithm might encounter situations where a previous zoom prediction is wrong. To improve the robustness of the model, we add noise to the zoom label by flipping the ground truth with a probability of 0.3. We found that models trained without random flipping are significantly less accurate. For each input image we initiate this procedure with five sub-images and repeat it multiple times. We also append horizontally flipped images to the dataset for data augmentation.

Assignment of labels for the zoom indicator follows the discussion of Section 2.3. The label is 1 if there exists an object with 50% of its area inside the region and the area is at most 25% of the size of the region. Note that here we use a loose definition of inclusion to add robustness for objects falling between boundaries of anchors. For adjacency prediction, we set a threshold in the intersection-over-union (IoU) score between an object and a region. A region is assigned to detect objects with which it has sufficient overlap. The assigned objects are then greedily matched to one of the sub-regions defined by the priors shown in Figure 2.4. The priority in the matching is determined by the IoU score between the objects and the sub-regions. We note that in this manner multiple predictions from a region are possible.

Loss Function

As shown in Figure 2.5, the AZ-Net has three output layers. The zoom indicator outputs from a sigmoid activation function. To train it we use the cross-entropy loss function popular for binary classification. For the adjacency predictions, the bounding boxes are parameterized as in Fast R-CNN [61]. Unlike in Fast R-CNN, to provide multiple predictions from any region, the confidence scores are not normalized to a probability vector. Correspondingly we use smooth L1-loss for bounding box output and element-wise cross-entropy loss for confidence score output. The three losses are summed together to form a multi-task loss function.

Fast R-CNN Detectors

The detectors we use to evaluate proposal regions are Fast R-CNN detectors trained using AZ-Net proposals. As in [61], we implement two versions: one with unshared convolutional features and the other that shares convolutional features with AZ-Net. The shared version is trained using alternating optimization.

2.4 Experiments

We evaluate our approach on Pascal VOC 2007 [63] and MSCOCO [64] datasets. In addition to evaluating the accuracy of the final detectors, we also perform detailed comparisons between the RPN approach adopted in Faster R-CNN and our AZ-Net on VOC 2007. At the end of the section, we give an analysis of the efficiency of our adaptive search strategy.

2.4.1 Results on VOC 2007

To set up a baseline comparison, we evaluate our approach using the standard average precision (AP) metric for object detection. For AP evaluation we use the development kit provided by the VOC 2007 object detection challenge. We compare our approach against the recently

Table 2.1: Comparison on VOC 2007 test set using VGG-16 for convolutional layers.

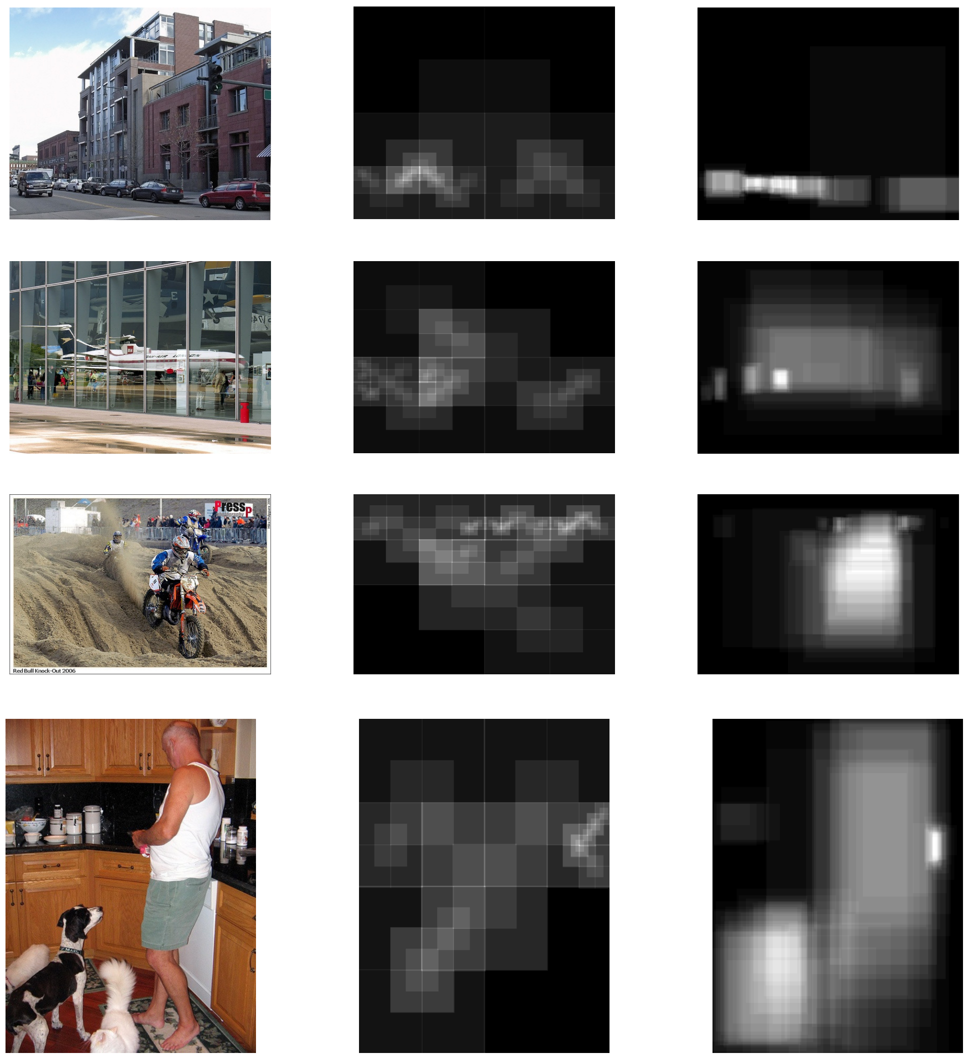
The results of RPN are reported in [61]. The results for Fast R-CNN are reported in [59]. The AZ-Net and RPN results are reported for top-300 region proposals, but in AZ-Net many images have too few anchors to generate 300 proposals. * indicates results without shared convolutional features. All listed methods use DCNN models trained on VOC 2007 trainval.

Method	**AZ-Net**	**AZ-Net***	**RPN**	**RPN***	**FRCNN**
boxes	231	228	300	300	2000
aero	73.3	73.9	70.0	74.1	**74.6**
bike	78.8	79.9	**80.6**	77.2	79.0
bird	69.2	68.8	**70.1**	67.7	68.6
boat	**59.9**	58.9	57.3	53.9	57.0
bottle	48.7	49.1	49.9	**51.0**	39.3
bus	**81.4**	80.8	78.2	75.1	79.5
car	82.8	**83.3**	80.4	79.2	78.6
cat	83.6	**83.7**	82.0	78.9	81.9
chair	47.5	47.2	**52.2**	50.7	48.0
cow	77.3	75.8	75.3	**78.0**	74.0
table	62.9	63.8	**67.2**	61.1	67.4
dog	**81.1**	80.6	80.3	79.1	80.5
horse	83.5	**84.4**	79.8	81.9	80.7
mbike	78.0	**78.9**	75.0	72.2	74.1
person	75.8	75.8	**76.3**	75.9	69.6
plant	38.0	**39.2**	39.1	37.2	31.8
sheep	68.7	70.2	68.3	**71.4**	67.1
sofa	67.2	67.4	67.3	62.5	**68.4**
train	79.0	78.4	**81.1**	77.4	75.3
tv	66.4	**68.3**	67.6	66.4	65.5
mAP	70.2	**70.4**	69.9	68.5	68.1

introduced Fast R-CNN [59] and Faster R-CNN [61] systems, which achieve state-of-the-art performance in standard benchmarks, such as VOC 2007 [63] and VOC 2012 [74]. A comparison is shown in Table 2.1. The results suggest that our approach is comparable to or better than these methods.

2.4.2 Quality of Region Proposals

We preform a detailed analysis of the quality of region proposals from our AZ-Net, highlighting a comparison to the RPN network used in Faster R-CNN. For all our experiments,



The left column shows the original image. The middle column shows the anchor regions induced by our adaptive search. The right column shows the top 100 adjacency predictions made around the anchor regions. The anchor regions and the adjacency predictions are superimposed into a figure at the same resolution of the original image. We note that the anchor regions and the region proposals in our approach are shared across object categories. For example, for the last image, the algorithm generates anchor regions at proper scales near the dogs, the person, and the bottles.

Figure 2.7: Example outputs of our algorithm.

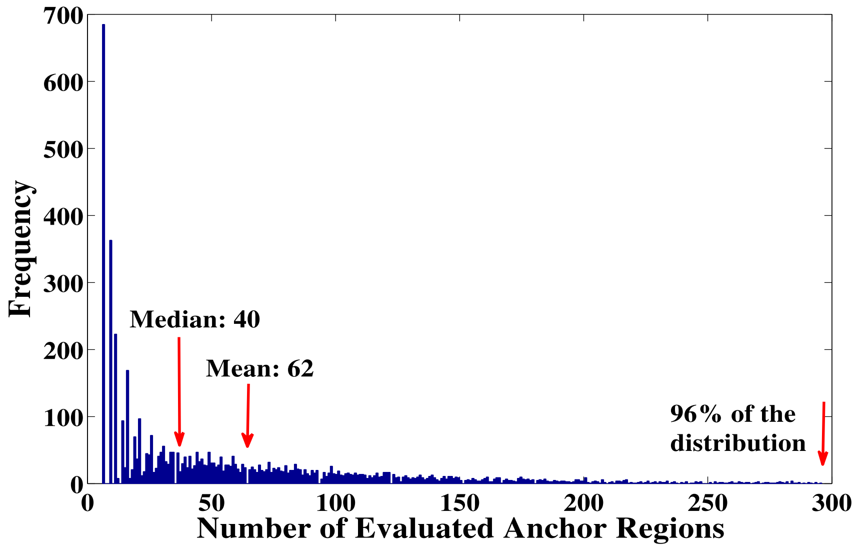
we analyze the recall on Pascal VOC 2007 test set using the following definition: An object is counted as retrieved if there exists a region proposal with an above-threshold IoU with it. The recall is then calculated as the proportion of the retrieved objects among all ground truth object instances. To accurately reproduce the RPN approach, we downloaded the region proposals provided on the Faster R-CNN repository [1]. We used the results from a model reportedly trained on VOC 2007 trainval. Correspondingly we compare it against our model trained on VOC 2007 trainval set. The comparisons concerning top-N regions are performed by ranking the region proposals in order of their confidence scores.

Figure 2.9 shows a comparison of recall at different IoU thresholds. Our AZ-net has consistently higher recall than RPN, and the advantage is larger at higher IoU thresholds. This suggests our method generates bounding boxes that in general overlap with the ground truth objects better. The proposals are also more concentrated around objects, as shown in Figure 2.10.

Figure 2.11 shows a plot of recall as a function of the number of proposals. A region proposal algorithm is more efficient in covering objects if its area under the curve is larger. Our experiment suggests that our AZ-Net approach has a better early recall than RPN. That means our algorithm in general can recover more objects with the same number of region proposals.

Figure 2.12 shows a comparison of recall for objects with different sizes. The "small object" has an area less than 32^2, a "medium object" has an area between 32^2 and 96^2, and a "large object" has an area greater than 96^2, same as the definition in MSCOCO [64]. Our approach achieves higher recall on the small object subset. This is because when small objects are present in the scene our adaptive search strategy generates small anchor regions around them, as shown in Figure 2.7.

[1]`https://github.com/ShaoqingRen/faster_rcnn`



For most images a few dozen anchor regions are required. Note that anchors are shared across categories.

Figure 2.8: Distribution of the number of anchor regions evaluated on VOC 2007 test set.

Table 2.2: Numbers related to the efficiency of the object detection methods listed in Table 2.1.

The runtimes for RPN and Fast R-CNN are reported for a K40 GPU [61]. Our runtime experiment is performed on a GTX 980Ti GPU. The K40 GPU has larger GPU memory, while the GTX 980Ti has higher clock rate. * indicates unshared convolutional feature version.

Method	Anchor Regions	Region proposals	Runtime (ms)
AZ-Net	62	231	171
AZ-Net*	44	228	237
RPN	2400	300	198
RPN*	2400	300	342
FRCNN	N/A	2000	1830

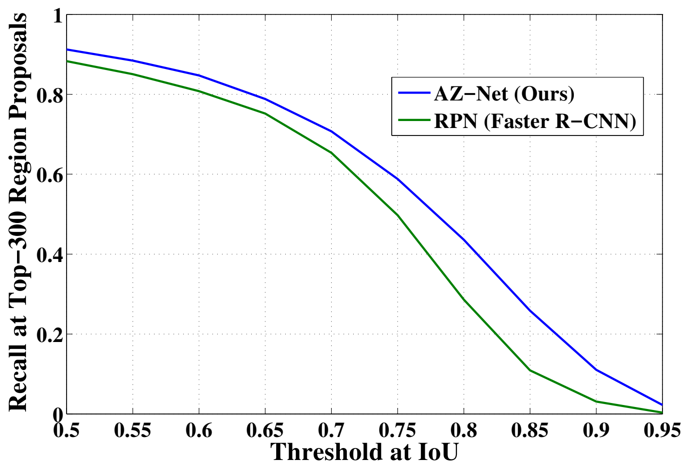
2.4.3 Efficiency of Adaptive Search

Our approach is efficient in runtime, as shown in Table 2.2. We note that this is achieved even with several severe inefficiencies in our implementation. First, for each image our algorithm requires several rounds of fully connected layer evaluation, which induces expensive memory transfer between GPU and CPU. Secondly, the Faster R-CNN approach uses convolutional computation for the evaluation of anchor regions, which is highly optimized compared to the RoI pooling technique we adopted. Despite these inefficiencies, our approach still achieves high accuracy at a state-of-the-art frame rate, using lower-end hardware. With improved implementation and model design we expect our algorithm to be significantly faster.

An interesting aspect that highlights the advantages of our approach is the small number of anchor regions to evaluate. To further understand this aspect of our algorithm, we show in Figure 2.8 the distribution of anchor regions evaluated for each image. For most images our method only requires a few dozen anchor regions. This number is much smaller than the 2400 anchor regions used in RPN [61] and the 800 used in MultiBox [52]. Future work could further capitalize on this advantage by using an expensive but more accurate per-anchor step, or by exploring applications to very high-resolution images, for which traditional non-adaptive approaches will face intrinsic difficulties due to scalability issues. Our experiment also demonstrates the possibility of designing a class-generic search. Unlike per-class search methods widely used in previous adaptive object detection schemes [67, 68] our anchor regions are shared among object classes, making it efficient for multi-class detection.

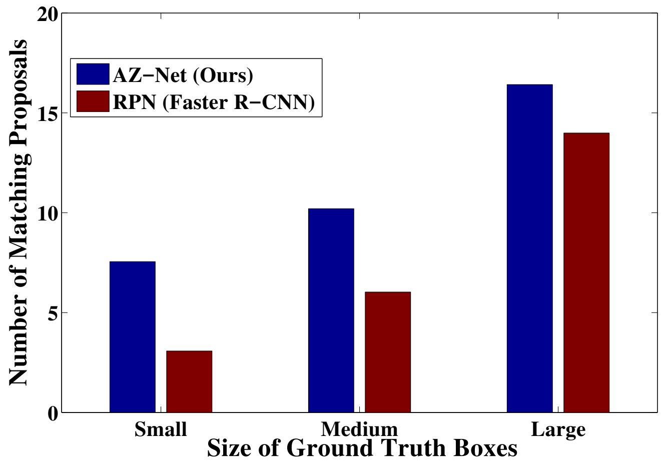
2.4.4 Results on MSCOCO

We also evaluated our method on MSCOCO dataset and submitted a “UCSD” entry to the MSCOCO 2015 detection challenge. Our post-competition work greatly improved accuracy with more training iterations. A comparison with other recent methods is shown in Table 2.3.



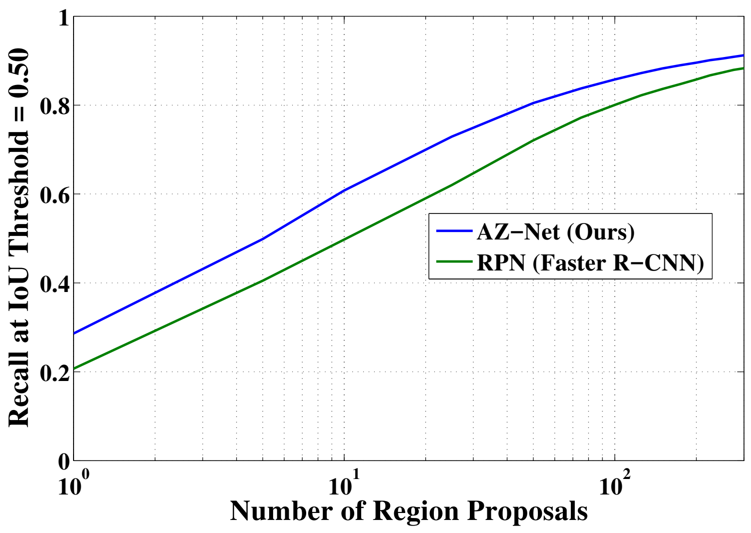
The comparison is performed at top-300 region proposals. Our approach has better recall at large IoU thresholds, which suggests that AZ-Net proposals are more accurate in localizing the objects.

Figure 2.9: Comparison of recall of region proposals generated by AZ-Net and RPN at different intersection over union thresholds on VOC 2007 test.



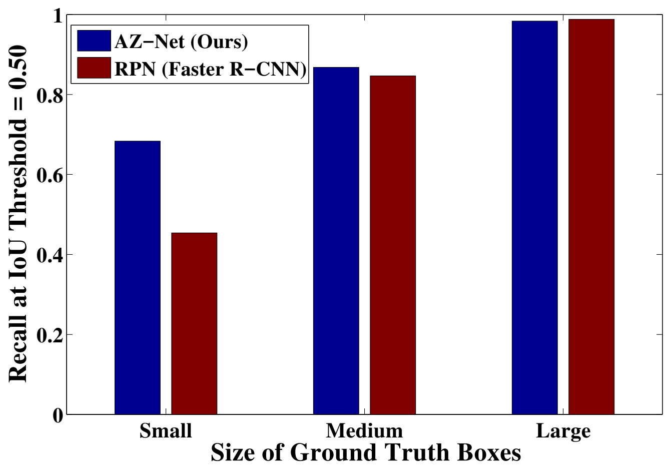
This shows proposals from AZ-Net are more concentrated around true object locations.

Figure 2.10: Number of proposals matched to ground truth (with IoU= 0.5).



The comparison is performed at IoU threshold 0.5. Our approach has better early recall. In particular, it reaches 0.6 recall with only 10 proposals.

Figure 2.11: Comparison of recall of region proposals generated by AZ-Net and RPN at different number of region proposals on VOC 2007 test.



The comparison is performed at IoU threshold 0.5 with top-300 proposals. Our approach has significantly better recall for small objects.

Figure 2.12: Comparison of recall of region proposals generated by AZ-Net and RPN for objects of different sizes on VOC 2007 test.

Table 2.3: The detection mAP on MSCOCO 2015 test-dev set.

The RPN (ResNet) entry won the MSCOCO 2015 detection challenge. Updated leaderboard can be found in `http://mscoco.org`.

Method	AP	AP IoU=0.50
FRCNN (VGG16) [59]	19.7	35.9
FRCNN (VGG16) [61]	19.3	39.3
RPN (VGG16)	21.9	42.7
RPN (ResNet)	37.4	59.0
AZ-Net (VGG16)	22.3	41.0

Our model is trained with minibatches consisting of 256 regions sampled from one image, and 720k iterations in total. The results for RPN(VGG16) reported in [61] were obtained with an 8-GPU implementation that effectively has 8 and 16 images per minibatch for RPN and Fast R-CNN respectively, each trained at 320k training iterations. Despite the much shorter effective training iterations, our AZ-Net achieves similar mAP with RPN(VGG16) and is more accurate when evaluated on the MSCOCO mAP metric that rewards accurate localization.

Our best post-competition model is still significantly outperformed by the winning "MSRA" entry. Their approach is a Faster-R-CNN-style detection pipeline, replacing the VGG-16 network with an ultra-deep architecture called Deep Residual Network [32]. They also report significant improvement from using model ensembles and global contextual information. We note that these developments are complementary to our contribution.

2.5 Conclusion and Future Work

This chapter has introduced an adaptive object detection system using adjacency and zoom predictions. Our algorithm adaptively focuses its computational resources on small regions likely to contain objects, and demonstrates state-of-the-art accuracy at a fast frame rate.

The current method can be further extended and improved in many aspects. Better pre-trained models [32] can be incorporated into the current system for even better accuracy. Further refining the model to allow single-pipeline detection that directly predicts class labels, as in

YOLO [62] and the more recent SSD [75] method, could significantly boost testing frame rate. Recent techniques that improve small object detection, such as the contextual model and skip layers adopted in Inside-Outside Net [76], suggest additional promising directions. It is also interesting to consider more aggressive extensions. For instance, it might be advantageous to use our search structure to focus high-resolution convolutional layer computation on smaller regions, especially for very high-resolution images.

Acknowledgment

Chapter 2, in full, is a reprint of the material that has been accepted for publication by *29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, authored by Yongxi Lu, Tara Javidi and Svetlana Lazebnik. The dissertation author was the primary investigator and author of this material.

Chapter 3

Target Localization with Drones using Mobile CNNs

3.1 Introduction

In recent years commercial quadcopters, or drones, have become widely available. These flying machines perceive the world through a unique perspective and unlike ground vehicles, are not subject to the usual traffic patterns. This has motivated a wide range of applications using drones. Recently, drones have been applied to search and rescue [77–80], active classifications [81, 82], aerial surveillance [83, 84] and agriculture [85], to name few. An enabler of many such applications is an algorithm that allows the drone to recognize and approach a pre-defined target quickly. Similar problems have been investigated in prior works [86–91]. Closest to our setting is [90, 91]. However, Sudevan *et al.*[90] uses a classical perception module, which is less effective for complex applications. Gupta *et al.*[91] assumes a simple observation model without validation on the physical environment. In view of these limitations of prior works, this chapter makes the following contributions:

- A dual-mode image processing mechanism is fully integrated into our Parrto Bebop 2 drone and the accompanying Samsung S8 phone. The first mode is faster but with less accuracy, while the second mode is more expensive but significantly more accurate. This is in line with prior work in detection and tracking on drones [80, 92, 93] which performs target search through image processing. Here our contribution is to extend the prior work to scenarios in which the search area is large, making it necessary to collect images at different flight locations.

- An image classifier trained on images collected from actually operating our drone. The classifier is built on state-of-the-art computer vision algorithms, a MobileNet CNN [94]. From the data collected, we note that the performance of the trained classifier is a function of flight altitude as well as random but persistent environmental factors that affects visibility. The former is in line with the assumptions in prior works, but we observe interesting differences. The latter is a novel aspect that has not been considered before.
- A partially observable Markov decision problem formulation. The proposed formulation incorporates the acquisition process as well as the classifiers' characteristics. Our work is in line with a large body of research on motion planning for drones using POMDPs [86–89, 91, 95], but our focus is on data collection and analysis that leads to a realistic model. Another novelty is the adoption of latest computer vision algorithms.
- A thorough comparative analysis of the impacts of the noise and the robustness to modeling artifacts. Through elaborate simulations, we investigate the impact of the persistent environmental factors in search using drones.

The chapter is organized as follows: Section 3.2 presents the problem formulation, highlights its difference to prior works and introduces our testbed. Section 3.3 discusses our simulation which shows the importance of considering the persistent factors in the environment, such as visibility. Section 3.4 introduces our dataset collection process and the perception module design. Section 3.5 concludes the chapter and discusses interesting future directions. The attached demo video shows a successful test flight of our system.

3.2 Target Search Problem

We consider the problem of optimization the flight path of a drone which is tasked with localizing a single static target on the ground in the face of uncertainty and physical constraints of the system. The components of our model include

- **Search Area** We consider the location of a single target of interest in a two-dimensional plane on the ground, denoted as $I_{\text{target}} = [I^{\text{la}}_{\min}, I^{\text{la}}_{\max}] \times [I^{\text{lo}}_{\min}, I^{\text{lo}}_{\max}]$. The static target is denoted as $Y_{\text{target}} \in I_{\text{target}}$.

- **Flight Space** We consider a three-dimensional region $I_{\text{flight}} = [I^{\text{la}}_{\min}, I^{\text{la}}_{\max}] \times [I^{\text{lo}}_{\min}, I^{\text{lo}}_{\max}] \times [I^{\text{alt}}_{\min}, I^{\text{alt}}_{\max}]$. This denotes the allowable space of flight for the drone. The location of the drone at time t is denoted as $X_t \in I_{\text{flight}}$.

- **Actions** At every time step, the drone is allowed to move to one of the immediate neighbors of its current location, denoted as $N(x_t)$. It will also choose a sensing mode $a_{\text{sense},t}$ from a finite set of sensing modes. The sensing modes differ in their cost and reliability. The drone can also choose to stop its operation and land at any time.

- **Observations** At any given time t and flight location $X_t \in I_{\text{flight}}$, the drone can acquire and process and image to arrive at a binary observation $O_t \in \{0,1\}$. This observation is an indication of whether the target is visible in the field-of-view (FOV) of the drone at this time step. We denote $Y_t \in \{0,1\}$ be the random variable denoting the "true" inclusion indicator, which depends on X_t and Y_{target}. O_t is a noisy version of Y_t.

- **Observation Noise** We take a probabilistic model to describe O_t as a function of the altitude of the drone (denoted as H_t), the visibility V that reflects environmental conditions, and the true inclusion indicator Y_t:

$$\begin{aligned}&\mathbb{P}(O_t = 1 | Y_t = y, H_t = h, V = v) \\ &= \begin{cases} 0 & \text{if } h \geq v \\ p_{0\to1}(h) & \text{if } h < v, y = 0 \\ 1 - p_{1\to0}(h) & \text{if } h < v, y = 1 \end{cases}\end{aligned} \tag{3.1}$$

We emphasize that $p_{0\to1}(h)$ and $p_{0\to1}(h)$ is more precisely a function of the sensing mode selected. We omit this in the notation for simplicity.

- **Prior distribution** We assume the target is uniformly distributed in I_{target}. We also assign a Bayesian prior to V, denoted as p_V. In practice this prior shall be estimated from data. For multiple sensing modes, a prior is assigned for each independently.

- **Reward** The drone receives a step-wise movement cost and sensing cost. The former is dependent on the speed of the drone. The latter is dependent on the sensing mode design. Both will be discussed in greater details later. If the drone chooses to stop, it receives a positive reward if: (a) $H_t = I^{\text{alt}}_{\text{min}}$ (flying at minimum altitude). (b) $Y_t = 1$ (the current FOV includes the target).

We note that our problem formulation is similar to the one investigated in the recent work [91]. However, our model is significantly different in its observation model. In particular, it has a notion of "visibility" which is not considered in prior works. Our model says that for each flight session, there is an intrinsic maximum visibility level of the current conditions. If the drone is flying on or above this level, the perception module will output a trivial "0" consistently. This is motivated by our observations from field tests and data collection practice (see Section 3.4 for more details). Our formulation models the effects of "persistent factors" in the environment. Those factors could be lighting conditions, background and weather. Presented with these conditions, the data-driven perception module can lose its ability to successfully detect the target, either due to reduced image quality or a lack of data points in the training set that represents the current condition. We note that the latter is particularly prevalent in a modern, data-driven perception module, such as the mobile CNN algorithm we adopted. While such algorithms are much more successful in complex tasks compared to classical algorithms, they could be more susceptible to the kind of catastrophic failures we consider in this work due to their internal complexity and requirement for large amount of data. In contrast, prior work assumes the observation noise is conditionally independent given the height and the target location. It does not consider persistent factors in the observation model. Their simplified model is unrealistic. From our simulations to be discussed in Section 3.3, ignoring those factors could lead to sub-optimal search algorithms.

3.2.1 Visibility-Aware POMDP

Our formulation can be easily casted into a POMDP, more precisely a mixed observability Markov decision processes (MOMDPs) [96]. The problem is specified by a tuple $\mathbf{P} = (S, \mathbb{P}_0, A, T, \Omega, H, R, \gamma)$, denoting states, initial state distributions, actions, transition function, set of observations, observation function, reward function and discount factor, respectively.

- **States:** The states $S = (\mathcal{X} \cup \{\xi, \Xi\}) \times \mathcal{Y} \times V$. We use quantization to simplify the problem domain. In particular, $\mathcal{X} \subseteq I_{\text{flight}}$, $\mathcal{Y} \subseteq I_{\text{target}}$ are discrete grid points within the flight space and the target space, respectively. We use a $N_{\text{la}} \times N_{\text{lo}} \times N_{\text{alt}}$ grid for $\mathcal{X}$, and its $N_{\text{la}} \times N_{\text{lo}}$ projection onto the ground plane for $\mathcal{Y}$. V is the space of visibility levels. Since $\mathcal{X}$ has N_{alt} distinct altitudes, V can be described by $(N_{\text{alt}}+1)^{N_{\text{sense}}}$ intervals covering $[0, \infty)$ for each sensing mode (N_{sense} denotes the number of sensing modes). ξ and Ξ is the starting and ending states, respectively. The $\mathcal{X} \cup \{\xi, \Xi\}$ factor is observable, while $\mathcal{Y}$ and V are hidden parts of the state space.

- **Initial Distributions:** The observable state is initialized at ξ, the starting state. The target $Y_{\text{target}} \in \mathcal{Y}$ follows uniform distribution, the visibility has a prior p_V.

- **Actions:** Since $\mathcal{X}$ is a grid point, the neighbor $N(X_t)$ is the four neighbors at the same altitude and the location directly above and below the current location. Thus the movement action can be specified by $\{\text{left}, \text{right}, \text{forward}, \text{backward}, \text{up}, \text{down}, \text{closing}\}$, denoted as A_{fly}. The drone also decides on the sensing modes for the next location, denoted as A_{sense}. The stopping action is denoted as Δ. The action space is thus $A = A_{\text{fly}} \times A_{\text{sense}} \cup \{\Delta\}$.

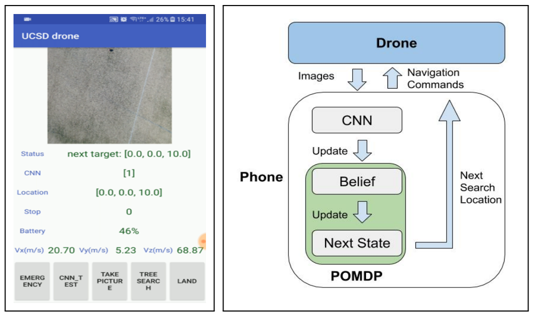
- **Transition Function:** If $s = \xi$, the drone will move to the highest location at the center of the flight space. If $a = \Delta$ or if a_{fly} leads to a location outside $\mathcal{X}$ the next state is Ξ (the end state). The drone will stay at the end state once entered. Otherwise the drone will move to the location specified by a_{fly} (thus changing the observable state). The hidden states are always static.

- **Observations:** The observation set $\Omega = \{0, 1, nil\}$.

- **Observation Function:** The observation is *nil* if and only if the next state is the stopping state *nil*. Otherwise the observation distributes according to Eqns. 3.1.

- **Reward Function:** The reward function is introduced in the problem formulation. We use realistic measurements to acquire movement and sensing costs. The positive reward for successful search is a hyperparameter.

- **Discount** We set the discount factor γ to 0.99.


UCSD drone

Status next target: [0.0, 0.0, 10.0]
CNN [1]
Location [0.0, 0.0, 10.0]
Stop 0
Battery 46%
Vx(m/s) 20.70 Vy(m/s) 5.23 Vz(m/s) 68.87

Left: Screenshot of the Android app. The app supports both manual control for data collection as well as autonomous search using the POMDP policies. **Right**: Control loop of the drone.

Figure 3.1: The control interface.

3.2.2 Control System

Approximately optimal solutions are found using the SARSOP solver [97]. We use the APPL toolkit [1]. Model searches are performed in 2-hour sessions following the practice in [91]. The solution of the POMDP problem using SARSOP is a piece-wise linear function that approximates the value function. This function is represented as a set of α-vectors with the length of the hidden states. Following the MOMDP formulation [96] each observable state s_{obs} is associated with a set of α-vector denoted as $\Gamma(s_{\text{obs}})$, each in turn is associated with a unique action $a(\alpha)$. Let the belief vector on the hidden states be b_t, the action is selected by

$$a(\alpha) = \arg\max_{\alpha \in \Gamma(s_{\text{obs}})} (\alpha \cdot b_t) \tag{3.2}$$

Given the current state, the controller first finds the next action using Eqns. 3.2. As in our model the next observable states is a deterministic function of the current observable state and the action, the observable state can be updated after the action is known. The drone can then move to the next position (or stop and land), and acquire an observation using the selected sensing mode at the new location. The observation is then used to update its internal belief on the hidden states. This process repeats until a timeout is issued by the meta-controller (to avoid excessively long search sessions), or a stop action is chosen. We implement this control loop on an Android

[1] `http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl`

device that communicates with the drone and its onboard camera through a wireless connection. All the control modules as well as the perception modules (to be discussed in Section 3.4) are implemented using TensorFlow [98] for convenient deployment. Figure 3.1 shows the UI design of the Android app and the illustration of the control loop for target search.

3.3 Simulations

We build a simulation environment to investigate the impact of the visibility level in our model. In our simulations, we assume the drone can only fly to the center points of a $7 \times 7 \times 7$ grid of the location space $\mathcal{X}$. The intervals defining the latitude, longitude and altitude ranges are $[-14, 14]$, $[-21, 21]$, $[4, 16]$ respectively, in meters. The drone is assumed to move at a speed of 2 m/s along arbitrary directions. The perception module supports up to two sensing modes, with delays of 0.4 seconds and 2.4 seconds per frame respectively. A 1.4-second delay is added to the cost of each sensing modes to model the communication between the drone and the android device. These movement and sensing delay parameters are realistic measurement in our testbed, and will be discussed in greater details in later sections. To highlight the effect of this structured noise, we perform simulations using a wide range of noise parameters. We set those numbers in a way that is qualitatively similar to real measurements (detailed in Section 3.4). To evaluate a given policy, we collect two statistics:

- **Localization accuracy** measured as the percentage of test sessions in which the drone stops at 4 meters, and its field-of-view includes the target.
- **Average search time** spent on the sessions. The time is measured using the aforementioned parameters.

An important detail is the sampling of the random variables, such as target locations, maximum visibility levels and observations. In our simulations, the target locations are sampled uniformly in the projection of the location grid onto the ground. Our simulator distinguishes two sampling modes

- **Structured persistent noise**: the maximum visibility level V is sampled first according to the supplied prior distribution, and then the observations are sampled according to Eqns. 3.1 given the realization of V.
- **Independent noise**: the observations has cross-over probabilities that are set to mimic the average case of the corresponding persistent noise model. More precisely, following Eqns. 3.3.

$$\begin{aligned} \tilde{p}_{0\to 1}(h) &= p_{0\to 1}(h)(1-P_V(h)) \\ \tilde{p}_{1\to 0}(h) &= (P_V(h)+p_{1\to 0}(h)(1-P_V(h))) \end{aligned} \tag{3.3}$$

where P_V is the cdf. of V, and the $p_{0\to 1}(h)$ and $p_{1\to 0}(h)$ are the false positive and false negative rates at height h, same as in Eqns. 3.1.

Baseline Algorithms Prior works consider random and heuristic policies as baselines. Another interesting baseline is linear search at the minimum height. However, all of these are much slower and is not the focus of this work, thus we do not include them in our comparison. We mainly compare our work against a simplified POMDP formulation that removes V (the visibility level) from our visibility-aware POMDP formulation. This formulation is a strong baseline. It is the same as the state-of-the-art model described in [91] with empirically measured (rather than heuristic) observation models. We highlight the importance of considering visibility as a persistent factor in the observation model through comparing with this baseline algorithm.

Meta Parameters for Simulations Unless specified otherwise, all simulations are performed with a timeout period of 180 seconds and the statistics are compiled from 10,000 repetitions.

3.3.1 Advantage of Considering Structured Noise

We first consider a scenario in which the environment is filled with structured noise. The designer of the search algorithm may neglect this structure, and use the baseline algorithm instead. This designer still has access to the empirical observation of the error rates, which is the average of different visibility levels (see Eqns. 3.3). This is compared against a design based on accurate measurement on p_V and the adoption of visibility-aware algorithm.

Table 3.1: Improved Search Time using Visibility

The baseline model has access to the empirical erro r rates averaged over the prior on V. Simulations are performed for structured, persistent noise.

Single	**Time (seconds)**		**Success Rate**	
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.2$	44.09	46.76	99.82%	99.81%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.4$	45.72	48.35	99.88%	99.78%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.6$	46.04	52.30	99.92%	99.80%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.8$	46.85	46.91	99.93%	99.89%
Dual	**Time (seconds)**		**Success Rate**	
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.2$	39.76	42.82	99.88%	99.72%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.4$	40.08	43.53	99.87%	99.79%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.6$	40.07	47.47	99.79%	99.69%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.8$	40.07	67.02	99.76%	99.82%

We build two sets of test cases to identify situations in which the effects of structured persistent noise are largest. Set one concerns one sensing mode. We set the false positive rate to 0.01, and set the intrinsic false negative rate to a monotonically increasing linear function of height. At 4 meters the false negative rate is 0.025. The rate grows at a step of 0.025 per two meters.

These numbers are qualitatively similar to real measurements for the less expensive sensing mode, shown in Figure 3.5. We assume the maximum visibility level is either at 12 meters or above 16 meters, for the sake of simplicity. Thus the prior on the visibility level is completely specified by $p_v(12)$ where $p_v(12) + p_v(>16) = 1$. We set $p_v(12)$ to $\{0.2, 0.4, 0.6, 0.8\}$ to investigate different strength of the persistent noise. It is assumed that sensing takes 1.8 seconds per frame (the less expensive sensing mode in the simulator). For set two, we add a more expensive but less erroneous sensing mode to the problem (3.8 seconds per frame including communication cost). It is assumed to have the same intrinsic error rates with the sensing mode in set one, but it always has $p_v(>16) = 1$. This added sensing mode thus serves as an expensive "backup": if the less expensive sensing mode fails above 12 meters, the "clean" but more expensive sensing mode can kick in. Intuitively, this should enable the drone to use a more aggressive strategy in using the less expensive sensing mode.

Table 3.1 summarizes the results. In the single sensing mode case, the baseline algorithm results in longer search time except for when $p_V(12) = 0.8$. The gain is largest when $p_V(12) = 0.6$. Intuitively, when persistent visibility is insignificant (small $p_V(12)$), the advantage of using the visibility-aware formulation is expected to be small. Interestingly, when $p_V(12)$ is very large the gain also diminishes. We conjecture that at this particular situation, the two formulations result in similar policies. Intuitively, when the empirical error rates are large at high altitudes, the drone should quickly descend to avoid wasting time. Access to visibility level is not helpful in this scenario as in most realizations the visibility is poor. This suggests considering persistent factors when their effects are significant but moderate is most important for designing successful search strategies. In the dual sensing mode case however, the visibility-aware formulation yields largest gain when $p_V(12) = 0.8$. The knowledge of the visibility structure seems to allow the drone to evaluate the information from the two sensing modes more accurately.

Table 3.2: Time at Varying Realization of Visibility

Comparing search time at different realization of the visibility. $p_V(12)$ shows the prior on visibility that controls the noise model available to the policy search algorithm at the design phase. $V > 16$ can be seen as "good" realization of visibility, as the drone can make gain information from high altitudes. $V = 12$ is correspondingly the "bad" visibility realizations.

	$p_V(12) = 0.2$		$p_V(12) = 0.4$	
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$V > 16$	42.58	39.99	44.49	39.88
$V = 12$	50.27	73.67	47.61	60.96

	$p_V(12) = 0.6$		$p_V(12) = 0.8$	
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$V > 16$	43.68	40.10	43.82	44.06
$V = 12$	47.59	60.33	47.61	47.63

3.3.2 Comparing Policies

It is very interesting to understand why and how the baseline algorithm is inferior when the structured noise we consider is present. While it is in general difficult to understand the numerical solution of a complex POMDP, we provide insights through analysis on the sample path realizations taken by individual policies. For simplicity, we focus on the single sensing mode case.

We first look at the average time required for the search to stop at sessions at varying realization of visibility. We note that in our simple simulation case, the maximum visibility level can either be at 12 meters or above 16 meters. Thus there are only two cases, "good" or "bad" visibility. In Table 3.2 we compare the two types of policies. In general, policies from visibility-aware POMDP searches slightly longer in cases where the visibility is good (only invisible above 16 meters), but when visibility is poor baseline policies perform much longer search. This suggests that excessive sensing in cases where visibility is poor is the main reason for the longer overall search time of the baseline policies.

Table 3.3 shows the average number of acquisitions performed at different heights. Comparing the two types of policies, the baseline policies spend more time on or above 12 meters, as

Table 3.3: Acquisitions at Varying Heights

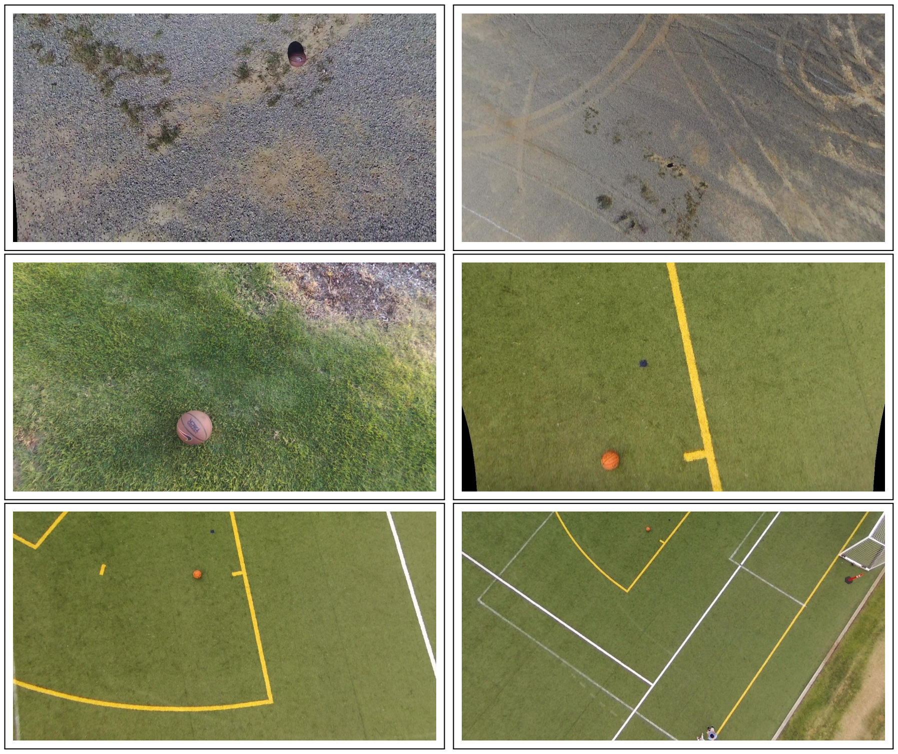
The number of acquisitions (observations) made at different heights. This table shows the simulation under the structured noise. It compares the behavior of the two types of policies in this realistic setting.

	$p_V(12) = 0.2$		$p_V(12) = 0.4$	
Height (meters)	*Visibility*	*Baseline*	*Visibility*	*Baseline*
4	3.72	2.89	2.58	3.11
6	1.44	1.35	2.66	1.56
8	1.70	1.78	1.56	1.58
10	2.35	1.38	3.46	2.21
12	1.03	2.00	1.01	1.01
14	1.00	1.01	1.03	1.00
16	1.74	3.17	1.00	3.58

	$p_V(12) = 0.6$		$p_V(12) = 0.8$	
Height (meters)	*Visibility*	*Baseline*	*Visibility*	*Baseline*
4	2.25	2.73	1.82	2.65
6	3.10	2.17	2.71	2.60
8	1.49	1.85	2.32	1.75
10	3.55	2.19	3.72	3.59
12	1.00	1.00	1.00	1.00
14	1.00	1.00	1.00	1.00
16	1.00	4.08	1.01	1.00

well as on the lowest height of 4 meters. From 6 to 10 meters the baseline policies spend less time. This suggests that the visibility-aware policies tend to avoid spending too much time on heights subject to structured noise. The knowledge of visibility level could lead to more accurate belief updates, since if the visibility is found to be poor the observations collected from 12 to 16 meters should be discarded. This could be the reason for the shorter search at the lowest height.

But could the visibility-aware POMDP collect information about the maximum visibility level, given that it spends less observations above 12 meters? To answer this question, we examine the belief vectors of the POMDPs after stopping. We use MAP decoding on the belief vectors and compare against ground truth maximum visibility levels. It seems at termination the POMDPs is in many cases aware of the true visibility levels, as in Table 3.4 which shows accuracies are well above random guessing (which has 50% accuracy).



The images are collected at different heights, and with different background. The figure only shows images with basketballs. Background images without basketballs are also included in the dataset.

Figure 3.2: Example images collected in this project.

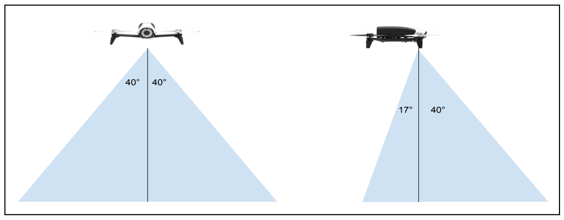
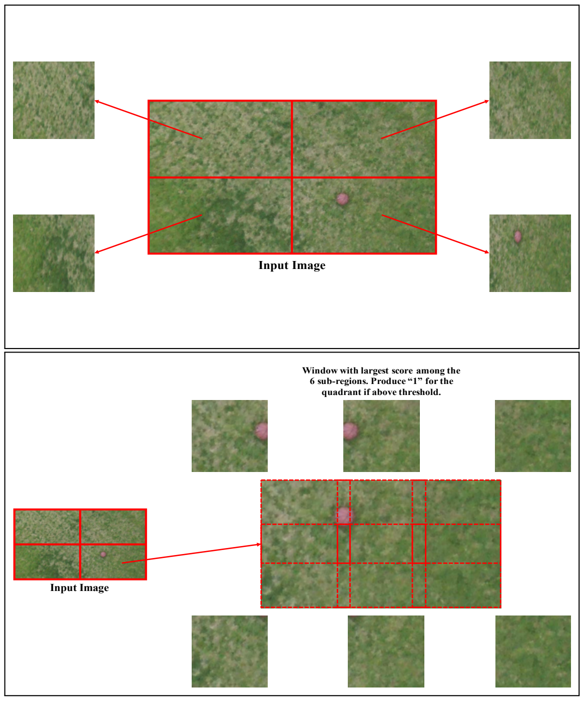


Figure 3.3: The field of view of the drone.

Table 3.4: Accuracy of Visibility Estimator

Accuracy of the visibility estimator, built using MAP decoding of the belief vector at the termination of the algorithm. This table shows that the proposed algorithm is indeed learning the true visibility level during the search.

$p_V(12) = 0.2$	$p_V(12) = 0.4$	$p_V(12)0.6$	$p_V(12) = 0.8$
89.92%	75.92%	79.01%	89.35%


Input Image
Window with largest score among the 6 sub-regions. Produce "1" for the quadrant if above threshold.
Input Image

Top: Sensing mode 1, resizing quadrants. **Bottom**: Sensing mode 2, process multiple crops at each quadrant.

Figure 3.4: Two sensing modes.

3.3.3 Risk of Over-Modeling

Another important issue is the potential cost of over-modeling. This is the opposite situation of Section 3.3.1. In this case, the designer assumes the perception module is subject to the persistent factors in the environment, while in reality such factors are not significant. To investigate this issue, we use the same policies but test them in a simulated environment with independent, non-structured noise. Table 3.5 summarizes the result which suggests that it is important not to over-model.

Table 3.5: Risk of Over-Modeling

The baseline model has access to the empirical error rates averaged over the prior on V. Simulations are performed for independent noise.

Single	**Time (seconds)**		**Success Rate**	
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.2$	42.99	40.73	99.80%	99.85%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.4$	44.89	42.12	99.81%	99.82%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.6$	45.23	44.76	99.89%	99.77%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.8$	46.83	46.09	99.84%	99.90%

Dual	**Time (seconds)**		**Success Rate**	
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.2$	41.20	42.56	99.89%	99.71%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.4$	43.62	43.46	99.71%	99.75%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.6$	48.85	44.44	99.26%	99.78%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.8$	58.18	45.01	99.11%	99.97%

3.4 Data Collection And Perception Module Design

A major goal of this work is to investigate target search with drones using realistic perception modules. There is a rich and growing literature in machine learning on efficient neural networks for mobile applications [94, 99, 100]. We choose to implement a perception module based on MobileNet, in particular the variant with width multiplier of 1.0 [94]. We fine-tune from a model pre-trained on ImageNet [2], on the dataset specifically collected for this project. The particular architecture variant is selected to balance the accuracy/complexity tradeoff for our application. From benchmarking on our Samsung S8 mobile phone, the inference speed is approximately 100 ms per frame with 224×224 input images [3]. The limited size of our dataset makes it necessary for us to fine-tune on an ImageNet pretrained model. As the network are trained for 224×224 resolutions, to alleviate overfitting we design our perception module in such a way that it takes multiple cropped and/or resized images to this resolution at inference.

3.4.1 Data Collection and Model Training

As an exemplary application, we consider searching for a single basketball on the ground. To train a mobile CNN model for this application we collect image data with one or more basketballs on various locations and at different heights. In total we collect 41 clips, each around 2-3 minutes (due to the limited flying time per charge). The height ranges from 2 meters to 64 meters. Due to safety and regulation reasons, the data collection is performed at only a few locations on and around campus with sufficient clearance from people, building and obstacles. There is a bias in the data collection process towards day time and good weathers. This bias is mainly due to daily routines at facilities where data collection is performed. Example images from our *basketball-search* dataset is shown in Figure 3.2.

The training and validation images are obtained by sampling frames extracted from the

[2]Available at `https://github.com/tensorflow/models/tree/master/research/slim`

[3]The phone is loaded with an Android system. Due to lack of API support, computations are performed on CPU.

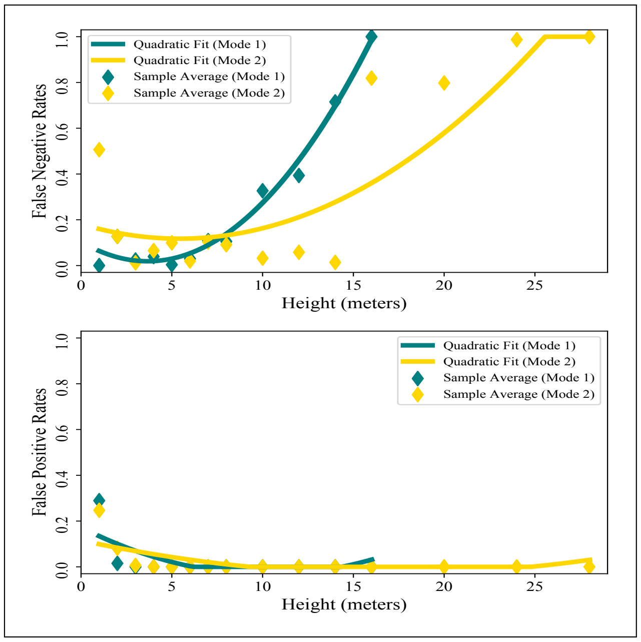
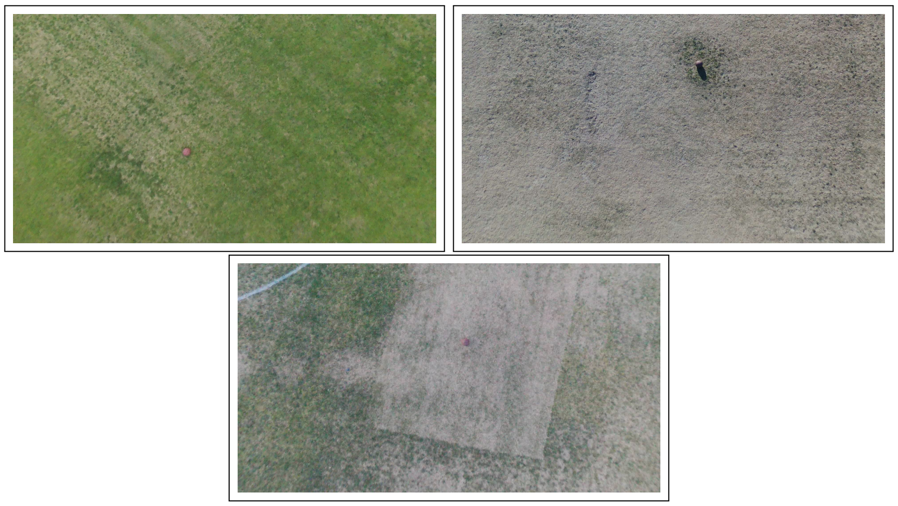


Figure 3.5: False positive and false negative rates in two sensing modes. Quadratic fittings are provided to illustrate the general trends, while the original data points are summarized as scatter points.

raw clips. The extracted frames are full-size 1280×720 images, the resolution of original videos saved onboard the drone. The field-of-view (FOV) of the drone is shown in Figure 3.3. The images are fully labeled with bounding box annotations, with 1,021 positive images (w/ basketballs) and 2,304 negative images (w/o basketballs). As discussed previously during inference images are processed at 224×224 resolution, thus we use this resolution for training. The training images are random crops of 224×224 from the full-size images. Crops are assigned a binary label with the following rules: On a positive image, if a random crop overlaps with a basketball with more 80% of the area of the latter, then the crop is labeled "1"; else if its overlapping with any basketball is less than 20% of the latter its label is "0". For negative images, all crops are labeled "0". We randomly discard crops to ensure crops labeled with "1" from positive images, those labeled with "0" from positive images and crops from negative images roughly follow the a 1:1:1 proportion. The crop sampling process is repeated 4 times. This results in 4,084 crops with label "1" and 13,300 negative crops with label "0". This sampling procedure is essential in ensuring a



All pictures are taken at 8 meters. **Left**: Image taken from the test clip at 8 meters. The image was taken at a condition in which the basketball has clear contrast with the background. Sensing mode 1 has only around 10% false negative rate on this clip. In similar conditions, sensing mode 1 will consistently produce "0" only on and above 16 meters. **Middle**: Image taken from a field test around 3pm in the afternoon. The plain background, the color of the sunlight as well as the long shadows make the basketball less visible. As a result, during repeated experiments during the field test the CNN classifier consistently reports negative ('no basketball') on and above 8 meters. **Right**: Image taken from a field test at dawn. The lighting condition is not ideal, causing large noise in the image.

Figure 3.6: Illustration of varying maximum visibility levels caused by persistent factors in the scene.

balanced training set. A random partition separates this set into a training set and a validation set with a 3:1 ratio.

We remove the output 1000-way fully-connected layer and replace it with a 2-way output. The model is trained with softmax loss. The model is first trained for 20 epochs during which all layers except for the output layer are frozen. Then the remaining 20 epochs are trained with all layers. The fine-tuning is performed with a learning rate of 0.02. Random horizontal flipping and random saturation are added for data augmentation. The resultant model has 98.94% accuracy on training and 98.78% on validation. This CNN model is then used as the backbone for our perception module.

3.4.2 Sensing Modes and Error Characteristics

We design two sensing modes using the mobile CNN. As an image captured by the drone has a 1280×720 resolution, instead of processing it in the original resolution we partition the image into four quadrants. Each quadrant is a region with 640×360 resolutions. The outputs of the perception module are conceptually similar to the observation presented in Section 3.2. However, instead of producing a single binary output each quadrant provides a $\{0,1\}$ indicator of whether the particular sub-region includes the basketball. In our simulations and field tests these observations are treated as observations taken from a smaller FOV compared to the entire image. Observations from the same image are assumed independent given the target location and the maximum visibility level. We implement two sensing modes. For the less expensive sensing mode 1, we directly resize (through interpolation) to a 224×224 image. This results in less accurate observations, but the inference time is only $4 \times 100 = 400$ ms per frame. For the more expensive sensing mode 2, we perform linear scan within the 640×360 sub-regions, and take the maximum confidence score as the prediction. The test time is thus $6 \times 4 \times 100 = 2400$ ms per frame. In both test modes, the resultant confidence scores from the neural network are thresholded to obtain binary predictions. Figure 3.4 illustrates the two sensing modes.

We collect a separate test set from manually operating the drone at 1-meter intervals from 1 to 8 meters, 2-meter intervals from 8 to 16 meters, and 4-meter intervals from 16 to 28 meters, to empirically test the error characteristics of the proposed sensing module. Using a similar cropping strategy as in the training procedure, at each height 150 positive and negative images are collected. However, the crops in this case are 640×360, matching the size of the four quadrants of the full-size images. We then test the false positive and false negative rates at each height, of the two sensing modes. The results are summarized in Figure 3.5. Interestingly, different from the assumptions made at prior works [91], the errors made by the CNN-based perception module are highly imbalanced. As expected, the false negative rates (or missing rates) grow as the height increases, however false positive rates tend to decrease to near zero values at higher altitudes. Both error rates see an increase when approaching the ground. This is caused by larger targets, which results in increased chance of cases in which the basketball is partially included in the FOV.

As discussed in Section 3.2, an important characteristic we observe from our field tests is that the predictions made by the neural network seems to have a “visibility level” structure. Figure 3.6 illustrates some conditions in which the perception module fails in this way, namely consistently producing “0” even if the basketball is present during repeated tests, for all images taken above a scene dependent level. This level tends to stay constant until after changing to another field test location or perform another field test at a different time of the day. Limited by available data, it is still impossible to make statistically significant conclusions. However, this phenomenon is intuitive for the search application using drones, and has large potential impact to the search strategy. In future works we plan to investigate this further through improved data collection or through video simulation.

3.5 Conclusion

In this work, we investigate the problem of target search using drones. We build a testbed that is used to investigate realistic conditions for this application. From testing perception modules using recent computer vision algorithms, we identify that persistent factors in the environment could have unexpected impact to the output of the observations. From our extensive studies through simulations, such factors have a significant influence on the design of the search algorithm. Important future directions include characterizing the impact of persistent factors more precisely, developing methods to automatically detect existence of those structures to avoid over-modeling, as well as to designing better computer vision algorithms that are more robust for target search and other related applications using drones.

Acknowledgment

Chapter 3, in full, is a reprint of the material that has been accepted for publication by *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2018)*, authored by Yongxi Lu, Zeyangyi Wang, Ziyao Tang and Tara Javidi. The dissertation author was the primary investigator and author of this material.

Chapter 4

Fully-adaptive Feature Sharing in Multi-Task Networks with Applications in Person Attribute Classification

4.1 Introduction

Humans possess a natural yet remarkable ability of seamlessly transferring and sharing knowledge across multiple related domains while doing inference for a given task. Effective mechanisms for sharing *relevant* information across multiple prediction tasks (referred as *multi-task learning*) are also arguably crucial for making significant advances towards machine intelligence. In this chapter, we propose a novel approach for multi-task learning in the context of deep neural networks for computer vision tasks. We particularly aim for two desirable characteristics in the proposed approach: (i) *automatic learning of multi-task architectures* based on branching, (ii) *selective sharing* among tasks with automated learning of whom to share with. In addition, we want our multi-task models to have low memory footprint and low latency during prediction (forward pass through the network).

A natural approach for enabling sharing across multiple tasks is to share model parameters (partially or fully) across the corresponding layers of the task-specific deep neural networks. Most of the multi-task deep architectures share the bottom layers till some layer l after which the sharing is blocked, resulting in task-specific sub-networks or branches beyond it [36, 101, 102]. This is motivated by the observation made by several earlier works that bottom layers capture low level detailed features, which can be shared across multiple tasks, whereas top layers capture features at a higher level of abstraction that are more task specific. It can be further extended to a more general tree-like architecture, e.g., a smaller group of tasks can share parameters even after the first break-point at layer l and breakup at a later layer. However, the space of such possible branching architectures is combinatorially large and current approaches largely make a decision based on limited manual exploration of this space, often biased by designer's perception of the relationship among different tasks [103]. Finding a data-driven approach to replace the manual exploration is an important and interesting academic question that has only been sparsely explored.

The proposed approach operates in a greedy top-down manner, making branching and task-grouping decisions at each layer of the network using a novel criterion that promotes the creation of separate branches for unrelated tasks (or groups of tasks) while penalizing for the model complexity. To address the issue of scalability to multiple tasks, the proposed approach starts with a *thin* network and dynamically grows it during the training phase by creating new branches based on the aforementioned criterion. We also propose a method based on simultaneous orthogonal matching pursuit (SOMP) [104] for initializing a thin network from a pretrained wider network (e.g., VGG-16) as a side contribution in this work.

We test this idea on facial (CelebA [41]), clothing (DeepFashion [105]) and person (facial combined with clothing) attribute classification, in all cases treating each attribute as a separate task. In these experiments, the models designed by our approach closely match state-of-the-art accuracy while being up to 90x more compact and 3x faster than the widely adopted VGG-16 architecture.

In summary, our main contributions are listed below:

- ◦ We propose to automate learning of multi-task deep network architectures through a novel dynamic branching procedure, which makes task grouping decisions at each layer of the network (deciding with whom each task should share features) by taking into account both task relatedness and complexity of the model.
- ◦ A novel method based on Simultaneous Orthogonal Matching Pursuit is proposed for initializing a thin network from a wider pre-trained network model, leading to faster convergence and higher accuracy.
- ◦ We demonstrate effectiveness of the proposed method on facial, clothing and joint person (facial+clothing) attribute classification, and investigate the behavior of the method through ablation studies.

4.2 Related Work

Multi-Task Learning There is a long history of research in multi-task learning [103, 106–109]. Most proposed techniques assume that all tasks are related and appropriate for joint training. A few methods have addressed the problem of "with whom" each task should share features [108–113]. These methods are generally designed for shallow classification models, while our work investigates feature sharing among tasks in hierarchical models such as deep neural networks.

Recently, several methods have been proposed for multi-task learning using deep neural networks. HyperFace [101] simultaneously learns to perform face detection, landmarks localization, pose estimation and gender recognition. UberNet [114] jointly learns low-, mid-, and high-level computer vision tasks using a compact network model. MultiNet [115] exploits recurrent networks for transferring information across tasks. Cross-ResNet [102] connects tasks through residual learning for knowledge transfer. However, all these methods rely on *hand-designed* network architectures composed of base layers that are shared across tasks and specialized branches that learn task-specific features.

As network architectures become deeper, defining the right level of feature sharing across tasks through handcrafted network branches is impractical. Cross-stitching networks [103] have been recently proposed to learn an optimal combination of shared and task-specific representations. Although cross-stitching units connecting task-specific sub-networks are designed to *learn* the feature sharing among tasks, the size of the network grows linearly with the number of tasks, causing scalability issues. We instead propose a novel algorithm that makes decisions about branching based on task relatedness, while optimizing for the efficiency of the model. We note that other techniques such as HD-CNN [116] and Network of Experts [117] also group related classes to perform hierarchical classification, but these methods are not applicable for the multi-label setting (where labels are not mutually exclusive).

Model Compression and Acceleration Our method achieves model compression and acceleration by considering task relatedness. It is complementary to existing task-agnostic approaches, such as knowledge distillation [118, 119], low-rank-factorization [120–122], pruning and quantization [123, 124], structured matrices [125–127], and dynamic capacity networks [128]. Many of these state-of-the-art compression techniques can be used to further reduce the size of our learned multi-task architectures.

Person Attribute Classification Methods for recognizing attributes of people, such as facial and clothing attributes, have received increased attention in the past few years. In the visual surveillance domain, person attributes serve as features for improving person re-identification [33] and enable search of suspects based on their description [34, 35]. In e-commerce applications, these attributes have proven effective in improving clothing retrieval [36], and fashion recommendation [37]. It has also been shown that facial attribute prediction is helpful as an auxiliary task for improving face detection [38] and face alignment [39].

State-of-the-art methods for person attribute prediction are based on deep convolutional neural networks [40–43]. Most methods either train separate classifiers per attribute [43] or perform joint learning with a fully shared network [44]. Multi-task networks have been used with base layers that are shared across all attributes, and branches to encode task-specific features for each attribute category [36, 45]. However, in contrast to our work, the network branches are hand-designed and do not exploit the fact that some attributes are more related than others in order to determine the level of sharing among tasks in the network. Moreover, we show that our approach produces a single compact network that can predict both facial and clothing attributes simultaneously.

4.3 Methodology

Let the linear operation in a layer l of the network be paramterized by W^l. Let $x^l \in \mathbb{R}^{c_l}$ be the input vector of layer l, and $y^l \in \mathbb{R}^{c_{l+1}}$ be the output vector. In feedforward networks that are of interest to this work, it is always the case that $x^l = y^{l-1}$. In other words, the output of a layer is the input to the layer above. In vision applications, the feature maps are often considered as three-way tensors and one should think of x^l and y^l as appropriately vectorized versions of the input and output feature tensors. The functional form of the network is a series of within-layer computations chained in a sequence linking the lowest to the highest (output) layer. The within-layer computation (for both convolutional and fully-connected layers) can be concisely represented by a simple linear operation parametrized by W^l, followed by a non-linearity $\sigma_l(\cdot)$ as

$$y^l = \sigma_l(\mathcal{P}_l(W^l)x^l), \tag{4.1}$$

where $\mathcal{P}_l$ is an operator that maps the parameters W^l to the appropriate matrix $\mathcal{P}_l(W^l) \in \mathbb{R}^{c_{l+1} \times c_l}$. For a fully connected layer $\mathcal{P}_l$ reduces to the identity operator, whereas for a convolutional layer with f_l filters, $W^l \in \mathbb{R}^{f_l \times d_l}$ contains the vectorized filter coefficients in each row and the operator $\mathcal{P}_l$ maps it to an appropriate matrix that represents convolution as matrix multiplication. We define the width of the network at layer l as c_l for the fully connected layers, and as f_l for the convolutional layers.

The widths at different layers are critical hyper-parameters for a network design. Successful deep convolutional network architectures, such as AlexNet [31], VGG [57], Inception [129] and ResNet [32] all use wider layers at the top of the network in what can be called an "inverted pyramid" pattern. From visualization of filters at different layers [130] it is observed that top level features tend to be task-dependent. More recently, researchers have noted that the width schedule (especially at the top layers) need to be tuned for the underlying set of tasks the network has to perform in order to achieve best accuracy [103]. Motivated by these findings, our approach starts with a "flat" architecture that has a similar width at all layers, then expands it to create explicit task-specific branches. The procedure is as follows:

Thin Model Initialization. Start with a thin neural network model, use random initialization or optionally initialize it from a pre-trained wider VGG-16 model by selecting a subset of filters using simultaneous orthogonal matching pursuit (ref. Section 4.3.1).

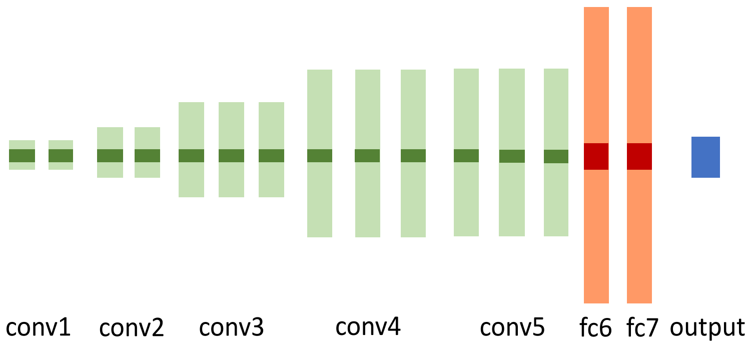
Adaptive Model Widening. The thin initialized model goes through a multi-round widening and training procedure. The widening is done in a greedy top-down layer-wise manner starting from the top layer. For the current layer to be widened, our algorithm makes a decision on the number of branches to be created at this layer along with task assignments for each branch. The network architecture is frozen when the algorithm decides to create no further branches (ref. Section 4.3.2).

Training with the Final Model. In this last phase, the fixed final network is trained until convergence.

More technical details are discussed in the next few sections. Algorithm 2 provides a summary of the procedure.

Algorithm 2: Training with Adaptive Widening

```
Data: Input data $D = (x_i, y_i)_{i=1}^N$. The labels $y$ are for a set of $T$ tasks.
Input: Branch factor $\alpha$, and thinness factor $\omega$. Optionally, a pre-trained network $M_p$
       with parameters $\Theta_p$.
Result: A trained network $M_f$ with parameters $\Theta_f$.
Initialization: $M_0$ is a thin-$\omega$ model with $L$ layers.
if exist $M_p, \Theta_p$ then
    $\Theta_0 \leftarrow SompInit(M_0, M_p, \Theta_p)$. $t \leftarrow 1$, $d \leftarrow T$. (Sec. 4.3.1)
else
    $\Theta_0 \leftarrow$ Random initialization
end
while $(t \leq L)$ and $(d > 1)$ do
    $\Theta_t, A_t \leftarrow TrainAndGetAffinity(D, M_t, \Theta_t)$ (Sec. 4.3.3)
    $d \leftarrow FindNumberBranches(M_t, A_t, \alpha)$ (Sec. 4.3.4)
    $M_{t+1}, \Theta_{t+1} \leftarrow WidenModel(M_t, \Theta_t, A_t, d)$ (Sec. 4.3.2)
    $t \leftarrow t + 1$
end
Train model $M_t$ with sufficient iterations, update $\Theta_t$. $M_f \leftarrow M_t$, $\Theta_f \leftarrow \Theta_t$.
```



The light color blobs shows the layers in the VGG-16 architecture. It has an inverted pyramid structure with a width plan of 64-128-256-512-512-4096-4096. The dark color blobs shows a thin network with $\omega = 32$. The convolutional layers all have widths of 32, and the fully connected layers have widths of 64.

Figure 4.1: Comparing the thin model with VGG-16.

4.3.1 Thin Networks and Filter Selection using Simultaneous Orthogonal Matching Pursuit

The initial model we use is a thin version of the VGG-16 network. It has the same structure as VGG-16 except for the widths at each layer. We experiment with a range of thin models that are denoted as thin-ω models. The width of a convolutional layer of the thin-ω model is the minimum between ω and the width of the corresponding layer of the VGG-16 network. The width of the fully connected layers are set to 2ω. We shall call ω the "thinness factor". Figure 4.1 illustrates a thin model side by side with VGG-16.

Using weights from pre-trained models is known to speed up training and improve model generalization. However, the standard direct copy method is only suitable when the source and the target networks have the same architecture (at least for most of the layers). Our adoption of a thin initial model forbids the use of direct copy, as there is a mismatch in the dimension of the weight matrix (for both the input and output dimensions, see Equation 4.1 and discussions). In the literature a set of general methods for training arbitrarily small networks using an existing larger network and the training data are known as "knowledge distillation' [118, 119]. However, for the limited use case of this work we propose a faster, data-free, and simple yet reasonably effective method. Let $W^{p,l}$ be the parameters of the pre-trained model at layer l with d rows. For convolutional layers, each row of $W^{p,l}$ represents a vectorized filter kernel. The initialization procedure aims to identify a subset of $d'(< d)$ rows of $W^{p,l}$ to form $W^{0,l}$ (the superscript 0 denotes initialized parameters for the thin model). We would like the selected rows that minimize the following objective:

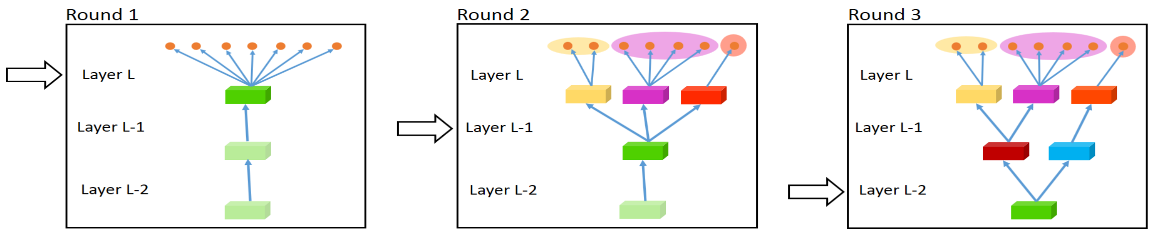
$$A^\star, \omega^\star(l) = \underset{A\in\mathbb{R}^{d\times d'}, |\omega|=d'}{\arg\min} \; ||W^{p,l} - AW^{p,l}_{\omega:}||_F, \tag{4.2}$$

where $W^{p,l}_{\omega:}$ is a truncated weight matrix that only keeps the rows indexed by the set ω. This problem is NP-hard, however, there exist approaches based on convex relaxation [131] and

```
Algorithm 3: SompInit($M_0$, $M_p$, $\Theta_p$)
Input: The architecture of the thin network $M_0$ with $L$ layers. The pretrained network and its
        parameters $M_p$, $\Theta_p$. Denote the weight matrix at layer $l$ as $W^{p,l} \in \Theta_p$.
Result: The initial parmaeters of the thin network $\Theta_0$.
foreach $l \in 1, 2, \cdots, L$ do
    Find $\omega^\star(l)$ in Equation 4.2 by SOMP, using $W^{p,l}$ as weight matrix.
    $W^{0,l} \leftarrow W^{p,l}_{\omega^\star(l):}$
    $W^{p,l+1} \leftarrow \left((W^{p,l+1})^T_{\omega^\star(l):}\right)^T$
end
Aggregate $W^{0,l}$ for $l \in \{1, 2, \cdots, L\}$ to form $\Theta_0$.
```

greedy simultaneous orthogonal matching pursuit (SOMP) [104] which can produce approximate solutions. We use the greedy SOMP to find the approximate solution $\omega^\star(l)$ which is then used to initialize the parameter matrix of the thin model as $W^{0,l} \leftarrow W^{p,l}_{\omega^\star(l):}$. We run this procedure layer by layer, starting from the input layer. At layer l, after initializing $W^{0,l}$, we replace $W^{p,l+1}$ with a column-truncated version that only keeps the columns indexed by $\omega^\star(l)$ to keep the input dimensions consistent. This initialization procedure is applicable for both convolutional and fully connected layers. See Algorithm 3.

4.3.2 Top-Down Layer-wise Model Widening



Left: the active layer is at layer L, there is one junction with 7 branches at the top. *Middle:* The seven branches are clustered into three groups. Three branches are created at layer L, resulting in a junction at layer $L-1$. Layer $L-1$ is now the active layer. *Right:* Two branches are created at layer $L-1$, making layer $L-2$ now the active layer. At each branch creation, the filters at the newly created junction are initialized by direct copy from the old filter.

Figure 4.2: Illustration of the widening procedure.

At the core of our training algorithm is a procedure that incrementally widens the current design in a layer-wise fashion. Let us introduce the concept of a "junction". A junction is a point at which the network splits into two or more independent sub-networks. We shall call such a sub-network a "branch". The leaves of each branch are outputs of a subset of tasks performed by this network. In person attribute classification each task is a *sigmoid* unit that produces a normalized confidence score on the existence of an attribute. We propose to widen the network only at these junctions. More formally, consider a junction at layer l with input x^l and d outputs $\{y_i^l\}_{i=1}^d$. Note that each output is the input to one of the d top sub-networks. Similar to Equation 4.1 the within-layer computation is given as

$$y_i^l = \sigma_l(\mathcal{P}_l(W_i^l)x^l) \qquad \text{for} \quad i \in [d], \tag{4.3}$$

where W_i^l parameterizes the connection from input x^l to the i'th output y_i^l at layer l. The set $[d]$ is the indexing set $\{1, 2, \cdots, d\}$. A junction is widened by creating new outputs at the layer below. To widen layer l by a factor of c, we make layer $l-1$ a junction with $2 \leq c \leq d$ outputs. We use y_j^{l-1} to denote an output in layer $l-1$ (each is an input for layer l) and W_j^{l-1} to denote its parameter matrix. All of the newly-created parameter matrices have the same shape as W^{l-1} (the parameter matrix before widening). The single output $y^{l-1} = x^l$ is replaced by a set of outputs $\{y_j^{l-1}\}_{j=1}^c$ where

$$y_j^{l-1} = \sigma_{l-1}(\mathcal{P}_{l-1}(W_j^{l-1})x^{l-1}) \qquad \text{for} \quad j \in [c]. \tag{4.4}$$

Let $g^l : [d] \to [c]$ be a given grouping function at layer l. After widening, the within-layer computation at layer l is given as (cf. Equation 4.3)

$$\begin{aligned} y_i^l &= \sigma_l(\mathcal{P}_l(W_i^l)x_{g^l(i)}^l \\ &= \sigma_l\left(\mathcal{P}_l(W_i^l)\sigma_{l-1}(\mathcal{P}_{l-1}(W_{g^l(i)}^{l-1})x^{l-1})\right) \end{aligned} \tag{4.5}$$

where the latter equality is a consequence of Equation 4.3. The widening operation sets the initial weight for W_j^{l-1} to be equal to the original weight of W^{l-1}. It allows the widened network to preserve the functional form of the smaller network, enabling faster training.

To put the widening of one junction into the context of the multi-round progressive model widening procedure, consider a situation where there are T tasks. Before any widening, the output layer of the initial thin multi-task network has a junction with T outputs, each is the output of a sub-network (branch). It is also the only junction at initialization. The widening operation naturally starts from the output layer (denoted as layer l). It will cluster the T branches into t groups where $t \leq T$. In this manner the widening operation creates t branches at layer $l-1$. The operation is performed recursively in a top-down manner towards the lower layers. Note that each branch will be associated with a sub-set of tasks. There is a 1-1 correspondence between tasks and branches at the output layer, but the granularity goes coarser at lower layers. An illustration of this procedure can be found in Figure 4.2.

4.3.3 Task Grouping based on the Probability of Concurrently Simple or Difficult Examples

Ideally, dissimilar tasks are separated starting from a low layer, resulting in less sharing of features. For similar tasks the situation is the opposite. We observe that if an easy example for one task is typically a difficult example for another, intuitively a distinctive set of filters are required for each task to accurately model both in a single network. Thus we define the affinity between a pair of tasks as the probability of observing concurrently simple or difficult examples for the underlying pair of tasks from a random sample of the training data.

To make it mathematically concrete, we need to properly define the notion of a "difficult" and a "simple" example. Consider an arbitrary attribute classification task i. Denote the prediction of the task for example n as s_i^n, and the error margin as $m_i^n = |t_i^n - s_i^n|$, where t_i^n is the binary label for task i at sample n. Following the previous discussion, it seems natural to set a fixed threshold on m_i^n to decide whether example n is simple or difficult. However, we observe that this is problematic since as the training progresses most of the examples will become simple as the error rate decreases, rendering this measure of affinity useless. An adaptive but universal (across all tasks) threshold is also problematic as it creates a bias that makes intrinsically easier tasks less related to all the other tasks.

These observations lead us to the following approach. Instead of setting a fixed threshold, we estimate the average margin for each task, $\mathbb{E}\{m_i\}$. We define the indicator variable for a difficult example for task i as $e_i^n = \mathbf{1}_{m_i^n \geq \mathbb{E}\{m_i\}}$. For a pair of tasks i, j, we define their affinity as

$$\begin{aligned} A(i,j) &= \mathbb{P}(e_i^n = 1, e_j^n = 1) + \mathbb{P}(e_i^n = 0, e_j^n = 0) \\ &= \mathbb{E}\{e_i^n e_j^n + (1-e_i^n)(1-e_j^n)\}. \end{aligned} \tag{4.6}$$

Both $\mathbb{E}\{m_i\}$ and the expectation on Equation 4.6 can be estimated by their sample averages. Since these expectations are functions of the current neural network model, a naive implementation would require a large number of time consuming forward passes after every training iterations. As a much more efficient implementation, we alternatively collect the sample averages from each training mini-batches. The expectations are estimated by computing a weighted average of the within-batch sample averages. To make the estimation closer to the true expectations from the current model, an exponentially decaying weight is used.

The estimated task affinity is used directly for the clustering at the output layer. It is natural as branches at the output layer has a 1-1 map to the tasks. But at lower layers the mapping is one to many, as a branch can be associated with more than one tasks. In this case, affinity is computed to reflect groups of tasks. In particular, let k, l denote two branches at the current layer, where i_k and j_l denotes the i-th and j-th task associated with each branch respectively. The affinity of the two branches are defined by

$$\tilde{A_b}(k,l) = \operatorname*{mean}_{i_k}\left(\min_{j_l} \quad A(i_k, j_l)\right) \tag{4.7}$$

$$\tilde{A_b}(l,k) = \operatorname*{mean}_{j_l}\left(\min_{i_k} \quad A(i_k, j_l)\right) \tag{4.8}$$

The final affinity score is computed as $A_b(k,l) = (\tilde{A_b}(k,l) + \tilde{A_b}(l,k))/2$. Note that if branches and tasks form a 1-1 map (the situation at the output layer), this reduces to the definition in Equation 4.6. For branches with coarser task granularity, $A_b(k,l)$ measures the affinity between two branches by looking at the largest distance (smallest affinity) between their associated tasks.

4.3.4 Complexity-aware Width Selection

The number of branches to be created determines how much wider the network becomes after a widening operation. This number is determined by a loss function that balances complexity and the separation of dissimilar tasks to different branches. For each number of clusters $1 \leq d \leq c$, we perform spectral clustering to get a grouping function $g_d : [d] \to [c]$ that associates the newly created branches with the c old branches at one layer above. At layer l the loss function is given by

$$L^l(g_d) = (d-1)L_0 2^{p_l} + \alpha L_s(g_d) \tag{4.9}$$

where $(d-1)L_0 2^{p_l}$ is a penalty term for creating branches at layer l, $L_s(g_d)$ is a penalty for separation. p_l is defined as the number of pooling layers above the layer l and L_0 is the unit cost for branch creation. The first term grows linearly with the number of branches, with a scalar that defines how expensive it is to create a branch at the current layer (which is heuristically set to double after every pooling layers). Note that in this formulation a larger α encourages the creation of more branches. We call α the branching factor. The network is widened by creating the number of branches that minimizes the loss function, or $g_d^{l\star} = \underset{g_d}{\arg\min} \quad L^l(g_d)$.

The separation term is a function of the branch affinity matrix A_b. For each $i \in [d]$, we have

$$L_s^i(g_d) = 1 - \underset{k \in g^{-1}(i)}{\text{mean}} \left(\min_{l \in g^{-1}(i)} A_b(k,l) \right), \tag{4.10}$$

and the separation cost is the average across each newly created branches

$$L_s(g_d) = \frac{1}{d} \sum_{i \in [d]} L_s^i(g_d). \tag{4.11}$$

Note Equation 4.10 measures the maximum distances (minimum affinity) between the tasks within the same group. It penalizes cases where very dissimilar tasks are included in the same branch.

4.4 Experiments

We test our approach on person attribute classification tasks. We use CelebA [41] dataset for facial attribute classification tasks and Deepfashion [105] for clothing category classification tasks. CelebA consists of images of celebrities labeled with 40 attribute classes. Most images also include the torso region in addition to the face. DeepFashion is richly labeled with 50 categories of clothes, such as “shorts”, “jeans”, “coats”, etc. (the labels are mutually exclusive). Faces are often visible on these images.

4.4.1 Comparison with the State of the art

A successful multi-task architecture should reach a good balance of speed, model complexity and accuracy. In this section we discuss how well the proposed approach work in these aspects compared to recent state-of-the-art methods in person attribute classifications. To facilitate fair comparison in a controlled setting, we also list baselines trained in comparable conditions with the proposed branching method. For baselines we prepare vanilla VGG-16 models, low-rank models that factorizes all layers and thins models. We summarize our results on Table 4.1 and 4.2. See training and model details in Section 4.4.5.

Accuracy Following established protocols, we report attribute classification accuracy and

Table 4.1: Comparison of accuracy, speed and compactness on CelebA test set.

LNet+ANet and Walk and Learn results are cited from [40]. MOON results are cited from [44]. $+$: There is no reported number to cite. $*$: MOON uses the VGG16 architecture, thus its test time should be similar to our VGG-16 baseline. We use the convention Branch-x-c to represent a model from the proposed branching method, with initial width of x and branching factor of c. Similar, Baseline-thin-x can be seen as Branch-x-0 (no branching). The details of architectural changes on low-rank and thin models are shown in Section 4.3.1.

Method	**Acc. (%)**	**Top-10 Recall (%)**	**Speed (ms)**	**Params. (M)**	**Jointly?**
LNet+ANet	87	N/A	$+$	$+$	No
Walk and Learn	88	N/A	$+$	$+$	No
MOON	90.94	N/A	$\approx 33^*$	119.73	No
Our VGG-16 Baseline	91.44	73.55	33.2	134.41	No
Our Low-rank Baseline	90.88	69.82	16.0	4.52	No
Our Baseline-thin-32	89.96	65.95	5.1	0.22	No
Our Branch-32-1.0	90.74	69.95	9.6	1.49	No
Our Branch-32-2.0	90.90	71.08	15.7	2.09	No
Our Branch-64-1.0	91.26	72.03	15.2	4.99	No
Our Joint Branch-32-2.0	90.4	68.72	10.01	3.25	Yes
Our Joint Branch-64-2.0	91.02	71.38	16.28	10.53	Yes

Table 4.2: Comparison of accuracy, speed and compactness on Deepfashion test set.

WTBI and DARN results are cited from [105]. The experiments are reportedly performed in the same condition on the FashionNet method and tested on the DeepFashion test set. $+$: There is no reported number to cite. $*$: There is no reported number, but based on the adoption of VGG-16 network as base architecture they should be similar to those of our VGG-16 baseline. #: The results are from a network jointly trained for clothing landmark, clothing attribute and clothing categories predictions. We cite the reported results for clothing category [105]. See captions of Table 4.1 for naming conventions.

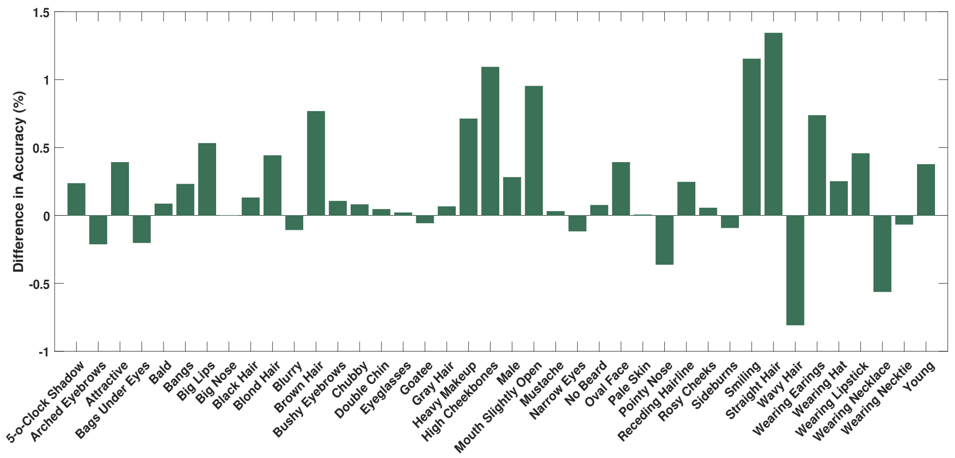
Method	**Top-3 Acc. (%)**	**Top-5 Acc. (%)**	**Speed (ms)**	**Params. (M)**	**Jointly?**
WTBI	43.73	66.26	$+$	$+$	No
DARN	59.48	79.58	$+$	$+$	No
FashionNet	82.58#	90.17#	$\approx 34^*$	$\approx 134^*$	No
Our VGG-16 Baseline	86.72	92.51	34.0	134.45	No
Our Low-rank Baseline	84.14	90.96	16.34	4.52	No
Our Joint Branch-32-2.0	79.91	88.09	10.01	3.25	Yes
Our Joint Branch-64-2.0	83.24	90.39	16.28	10.53	Yes

top-10 recall rate on CelebA dataset, and top-3 and top-5 classification accuracy on DeepFashion dataset. The evaluations are performed on respective test partitions. We conclude that while being more compact/faster, models generated by the proposed branching method can closely match the state-of-the-art results, including the strong vanilla VGG-16 baselines prepared for this work.

Model Complexity/Speed At a comparable accuracy level, our models are significantly faster and more compact than the vanilla baseline (and closest competitors in the literature, MOON and FashionNet, both based on VGG-16). Compared to the thin baseline, a steady improvement of accuracy as model complexity increases is observed. This improvement starts to saturate as we make the thinness factor larger (creating more branches). On the other hand, making the initial network wider becomes more effective when at higher accuracy levels. On the facial attribute tasks on CelebA, the Branch-64-1.0 model is more accurate than the baseline from low-rank factorization while being slightly faster and slightly less compact. While we only explore small initial width to demonstrate the speed-up/model compression effects of the proposed method, future work should more thoroughly investigate different thinness setting and/or widening individual branches.

4.4.2 Understanding the Task Grouping

It is interesting to see if the automated procedure is learning an intuitive grouping of tasks, or rather a surprising grouping that contradicts our own intuition. To this end we visualize task groupings in the top layer of the Branch-32-2.0 model (for CelebA facial attribute tasks). Figure 4.4 shows the visualization. We observe an intuitive set of groupings. For instance, “5-o-clock Shadow”, “Bushy Eyebrows” and “No Beard”, which all describe some forms of facial hairs, are grouped. The cluster with “Heavy Makeup”, “Pale Skin” and “Wearing Lipstick” is clearly related. Groupings at lower layers are also sensible. For instance, the group “Bags Under Eyes”, “Big Nose” and “Young” are joined by “Attractive” and “Receding Hairline” at fc6, probably because they all describe age cues.



A positive number suggests a reduction in accuracy when changing from original to the new grouping. This figure shows our automatic grouping strategy improves accuracy for most tasks.

Figure 4.3: The reduction in accuracy when changing the task grouping to favor grouping of dissimilar tasks.

4.4.3 Cross-domain Training

We examine the method's ability to handle cross-domain tasks by training a network that jointly predict facial and clothing attributes. The model is trained on the union of the two training sets. Note that the CelebA dataset is not annotated with clothing labels, and the Deepfashion dataset is not annotated with facial attribute labels. To augment the annotations for both datasets, we use the predictions provided by the baseline VGG-16 models as soft training targets. We demonstrate that the joint model is comparable to the state-of-the-art on both facial and clothing tasks, while being a much more efficient combined model rather than two separate models. The comparison between the joint models with the baselines is shown in Table 4.1 and 4.2.

4.4.4 Ablation Studies

Significance of grouping We swap the 20 tasks shown at the left side of Figure 4.4 with the 20 tasks at the right and retrain the model (using the "Branch-32-2.0" model illustrated in the figure). In this particular case this shuffling separates a large number of tasks originally in

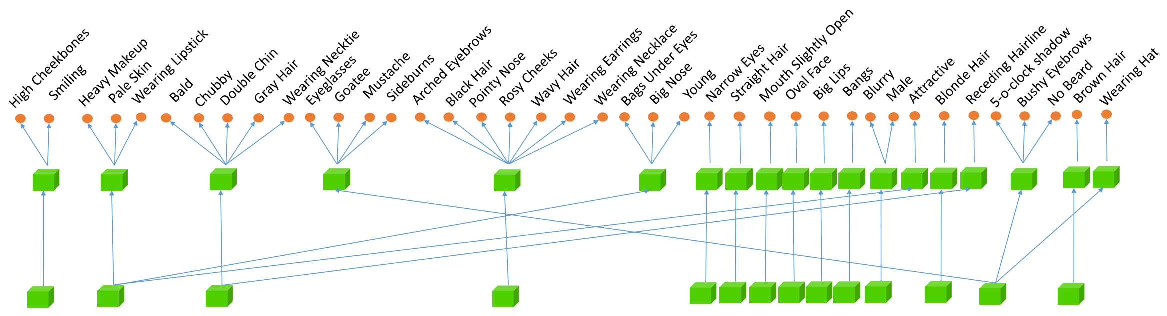


Figure 4.4: The actual task grouping in the Branch-32-2.0 model on CelebA. Upper: fc7 layer. Lower: fc6 layer. Other layers are omitted.

Table 4.3: Accuracy drop of Branch-32.2.0 compared to VGG-16 baseline, with and without initialization from pre-trained model.

Method	**Accuracy (%)**	**Top-10 Recall (%)**
w/ pre-trained	-0.54	-2.47
w/o pre-trained	-0.65	-3.77

the same branch, creating a helpful scenario to diagnose our approach. Figure 4.3 summarizes the reduction in accuracy due to reshuffling. In this case, grouping tasks according to similarity improves accuracy for most tasks. We observe similar behaviors from multiple random reshuffling, however due to the randomness the overall gain are less clear cut at times.

Cause of accuracy drop The drop in accuracy of the proposed method compared to VGG-16 baseline could to caused by sub-optimal use of the pretrained model (as the initial model is thinner), or the smaller capacity. We diagnose the issue by comparing the accuracy of Branch-32-2.0 and VGG-16 baseline with and without the initialization. If poor initialization is the main cause, removing the initialization should narrow the gap between the two models. This is not the case, as shown in Table 4.3. In this light, future work should probably prioritize better automated model design over more effective initialization.

Effects of SOMP initialization We compare training with and without this initialization using the Baseline-thin-32 model on CelebA, under identical training conditions. The evolution

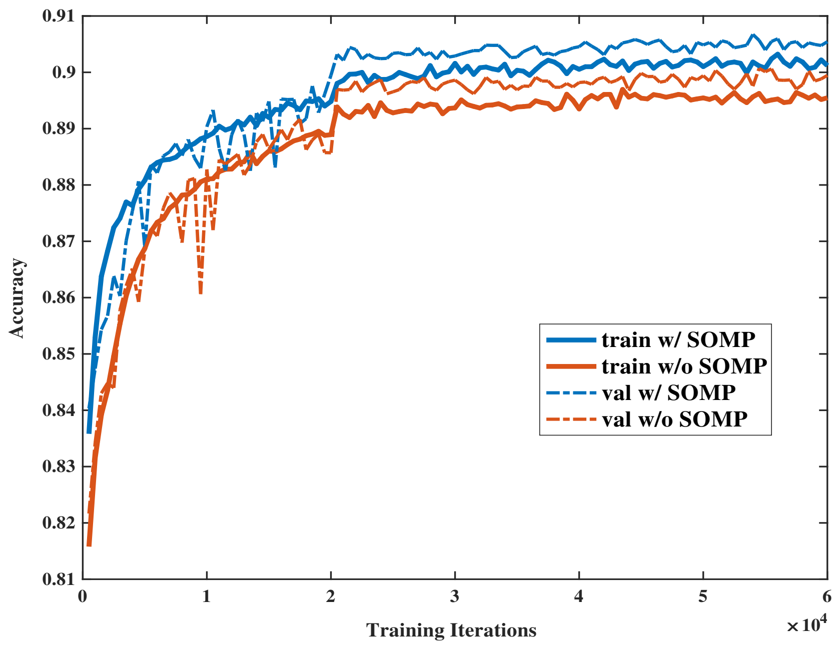


Figure 4.5: Comparison of training progress with and without SOMP initialization. The model using SOMP initialization clearly converges faster and better.

of training and validation accuracies are shown in Figure 4.5. Clearly, the network initialized with SOMP initialization converges faster and better than the one without SOMP initialization.

4.4.5 Training Details

For all experiments, the attribute outputs are sigmoid units. Loss function is sigmoid cross-entropy loss weighting attributes uniformly. Each attribute is treated as a task. We use the original partitions in the two datasets. For CelebA images without face alignment (unlike reported MOON results) are used as their contents provide contexts useful for clothing. Mini-batch size is 32. The training iterations is 60000, the learning rate is initialized to 0.001 and decayed by a factor of 10 every 20000 iterations. The initial model branching phase updates the model after every 1000 iterations. These are changed to 100000, 40000 and 2000 respectively for joint models. The branching factor and the initial width of thin models used to train various models are summarized in Table 1 and Table 2. The error decay factor is set to 0.99. Experiments are performed on a single K40 GPU. With branching (factor=1.0) the training time on CelebA is reduced to 17 from 40 hours (VGG16 baseline). All training uses Batch Normalization (BN) [132]. BN layers are removed and the corresponding convolution layers are scaled and shifted accordingly for faster inference.

Baselines The vanilla VGG-16 model is initialized from imdb-wiki gender VGG-16 models by replacing the output layers [133]. The low-rank model factorizes all layers. It is initialized using truncated SVD [134] from the imdb-wiki model (the number of basis filters is 8-16-32-64-64-64-64-16). The thin models are initialized with SOMP (See Section 4.3.1 for more details).

4.5 Conclusion and Future Works

We have proposed a novel method for learning the structure of compact multi-task deep neural networks. Our method starts with a thin network model and expands it during training by means of a novel multi-round branching mechanism, which determines with whom each task shares features in each layer of the network, while penalizing for the complexity of the model. We demonstrated promising results of the proposed approach on the problem of person attribute classification.

The method itself is independent of the underlying tasks since it only require a notion of correlation between them. It thus can be applied to multi-task problem beyond person attribute classifications in future works. We also plan to adapt this method to other related problems, such as incremental learning and domain adaptation.

Acknowledgment

Chapter 4, in full, is a reprint of the material that has been accepted for publication by *30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, authored by Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi and Rogerio Feris. The dissertation author was the primary investigator and author of this material.

Chapter 5

Implicit Label Augmentation on Partially Annotated Clips via Temporally-Adaptive Features Learning

5.1 Introduction

The success of modern machine learning techniques in solving challenging problems such as image recognition depends on the availability of large-scale, well-annotated datasets. Unfortunately, the most complex and useful tasks (e.g. semantic segmentation) are usually also the ones that require the most labeling efforts. This is arguably a major obstacle for large-scale applications to real-world scenarios, such as autonomous driving, where model performance is critical due to safety concerns. In this work, we focus on methods that can utilize partially annotated clip data, more precisely short video sequences with annotations only at key frames, to improve model performance. Datasets in this format are natural byproducts of typical data collection procedures. From clips, a large number of unlabeled frames is available at virtually no additional cost. But clip data can nevertheless encode rich temporal contexts useful for training more accurate models. Fully utilizing partially annotated clips in learning is an interesting problem not only for its practical relevance, but also because it provides partial answers to an interesting scientific question: Humans can naturally learn from continuous evolution of sensing signals without much "labels", can machines do the same?

We investigate a particularly intriguing case: To train a model that benefits from temporal information during training but is used to make predictions on independent frames at inference. This is in contrast to video prediction models [135–144] where video clips are used at both training and inference. The main intuition of our approach is to decouple fast-changing factors and slow-changing factors in data. Fast-changing factors reflect rapid temporal dynamics and can only be learned from a labeled frame or its immediate neighbors, while slow-changing factors can be learned from data points within a larger temporal context. Our method utilizes the temporal context provided by the partially annotated clips to learn better features without diminishing the ability to learn fine-grained features with rapid temporal changes. This is achieved by allowing different parts of the model to adapt to distinct temporal change rates in data, a.k.a. **Temporally Adaptive Features (TAF)** learning. We propose a principled approach to formalize this intuition by introducing temporal change rate constraints in the learning problem and show that the resultant optimization problem can be efficiently approximated by a feature swapping procedure with contrastive loss. The TAF paradigm generalizes the well-motivated "slow feature" learning methods [145, 146] for self-supervised learning. In this regard, ours is the first to demonstrate significant empirical gains on a challenging real-world application via imposing temporal coherence regularization. It can also be seen as a form of implicit label augmentation and is related to explicit pseudo label generation techniques [147–150] which also show promising improvements in practice. But ours is a more principled treatment that handles the important issue of label uncertainty automatically. Interestingly, our work is the first to combine these two seemingly unrelated line of research. It thus sheds new light on the theory and practice of the important problem of learning from partially annotated clips and can benefit future explorations on this topic.

The TAF framework can in theory be applied to any recognition tasks with partially annotated clip data. However, the advantage in doing so will well depend on the task. We identify **semantic segmentation**, the task of assigning class labels to every pixel in an image, as a good test case due to the necessity of multi-scale modeling. Natural images usually feature structures with a great variety of sizes, functions and perspectives. This results in different intrinsic spatial and temporal change rates of different structures. A useful semantic segmentation model needs to provide comprehensive understanding of all these different structures. TAF can address this challenge by allowing different parts of the model to learn features with varying temporal change rates, rather than forcing all the features to vary slowly, as is the case of slow feature learning [145, 146]. Beyond this particular task, semantic segmentation is also a good example of the broader set of "dense prediction tasks" in computer vision, such as object detection [12–18], pose estimation [19], monocular depth estimation [20–25], instance segmentation [18, 26–28] as well as panoptic segmentation [29, 30], to name a few. Dense prediction tasks all share the key properties of laborious annotation and multi-scale features thus it is likely that our finding from semantic segmentation can directly benefit these tasks.

While the most exciting aspect of this design is to enable learning from partially annotated clips for single frame models, preliminary work show that a similar design can lead to computational savings in video prediction by allowing different parts of the model to run at different frame rates [50]. A shortened version of this exploration is documented in Chapter 6. This suggests another interesting benefit of making different parts of the models adaptive to a distinct temporal change rate.

This chapter is organized as follows. Section 5.2 presents our method. Section 5.3 compares our method to related works. Section 5.4 presents our empirical findings and ablation studies. Section 5.5 concludes the chapter and discusses future directions.

5.2 Methods

We first introduce notations useful to our presentation. We denote the dataset as $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$. Each input and its associated labels can be finely indexed as $d_t^{(k)} = (x_t^{(k)}, y_t^{(k)})$, where k, t denotes the clip index and the time index within the clip, respectively. Whenever it is clear from the context, we use (x_t, y_t) to denote input-label tuples at time t for any particular clip.

5.2.1 Temporally Adaptive Feature Learning

Our method decouples the fast and slow changing factors in data by forcing the model to learn features that are adaptive to the varying temporal change rates. To be applicable to our framework, we assume the labeling function y can be factorized as

$$y(x;\Theta) = \Omega(\Phi_1(x;\theta_1), \cdots, \Phi_m(x;\theta_m);\omega)$$

We can quantify how fast the labeling function changes w.r.t. time by taking its time derivative.

$$\left\| \frac{dy(x_t;\Theta)}{dt} \right\| = \left\| \sum_{i=1}^{m} \frac{\partial \Omega}{\partial \Phi_i(x_t)} \frac{d\Phi_i(x_t)}{dt} \right\| \triangleq \left\| \sum_{i=1}^{m} \Psi_i(x_t) \right\| \tag{5.1}$$

Note that y can be seen as a function with m-dimensional input where $\Phi_i(x_t;\theta_i)$ represents one of its dimensions. $\Psi_i(x_t)$ quantifies the variation of y w.r.t. time through this dimension. The "fast" and "slow" factors are characterized by the degree at which they contribute to temporal variations in the predictive model y. To instantiate this idea, our TAF frameworks solves the following empirical risk minimization problem with temporal change rates constraints.

$$\min_{\Theta} \quad \mathbf{E}_{\mathcal{D}}\left[\ell(y(x;\Theta),y)\right] \tag{5.2a}$$

$$\text{subject to} \quad \|\Psi_i(x_t)\| \leq c_i, \quad \text{where } 0 \leq c_1 \leq c_2 \leq \cdots \leq c_m. \tag{5.2b}$$

For a differentiable model, the analytical form of the constraints is available if $\frac{dx_t}{dt}$ is provided. In applications where x_t is a high dimensional vector (such as an image), this may not be possible. Thus, we propose to use first-order finite difference to approximate the constraints via neighboring samples.

$$\|\Psi_i(x_t)\| \approx \frac{\left\|\Omega(\{\Phi_j(x_t)\}_{j\neq i}),\Phi_i(x_{t+\Delta}) - \Omega(\{\Phi_j(x_t)\}_{j\neq i}),\Phi_i(x_t)\right\|}{|\Delta|} \triangleq \frac{\delta y(x_t,i,\Delta)}{|\Delta|} \tag{5.3}$$

The constrained optimization problem itself is difficult to solve. We can convert it into an unconstrained optimization with the following regularization term which approximate the original constraints. This permits the use of gradient-based solvers if the model is differentiable.

$$R_+(x_t,i,\Delta) = (\delta y(x_t,i,\Delta) - |\Delta|c_i)_+ = \max\left(0,\delta y(x_t,i,\Delta) - |\Delta|c_i\right) \tag{5.4}$$

The proposed regularization is defined on any pairs of samples separated by known interval Δ, even if their labels are unknown. This construction thus enables learning from unlabeled data. Needless to say, TAF learning is limited to short clips as the first order approximation is valid only for small Δ. The slack term $|\Delta|c_i$ promotes features that adapts to a distinct temporal change rate. When c_i is small, that dimension is forced to model slow-changing factors shared within a large temporal context, which is an implicit form of data augmentation. The dimensions with large c_i on the other hand can still model rapid motions in data important for the task. As we will discuss in ablation studies and supplementary materials, c_i and m are important hyper-parameters.

As a remark on related methods, we note that Eqns. 5.4 generalizes the temporal coherence regularization in [145] to multiple change rates, making it more suitable for real-world applications such as semantic segmentation where multi-scale features are essential. This regularization can also be seen as implicitly assuming constant labels across the entire clip, with the slack term acknowledging the uncertainty introduced by this approximation. In this regard, c_i measures the growth rate of uncertainty in time of the implicit pseudo labels. Prior works suggests that properly modeling the uncertainty in pseudo labels to be important in the final task performance [147]. While TAF learning is motivated differently, it leads to a similar construction.

Finally, $\mathcal{D}_{\text{key}} \subseteq \mathcal{D}$ denotes the subset of the data with annotations, the half-length of each clip is n_h, Δ_0 is the sampling period and $\mathbf{U}$ denotes the uniform distribution defined on integers. The regularized optimization problem becomes

$$\min_{\Theta} \quad \mathbf{E}_{(x,y)\sim\mathcal{D}_{\text{key}}}\left[\ell(y(x;\Theta),y)\right] + \lambda \mathbf{E}_{\substack{(x_t,-)\sim\mathcal{D},\\ n\sim\mathbf{U}(-t\Delta_0,2n_h-t\Delta_0)}}\left[\sum_{i=1}^{m}\left[R_+(x_t,i,n\Delta_0)\right]\right]. \tag{5.5}$$

5.2.2 Efficient Frame Sampling

The proposed objective function in Eqns. 5.5 is computationally inefficient when combined with mini-batch SGD or its variants. First of all, the computation of the sample averages requires two separate sampling streams: One for the key frames with annotations for the loss function, and the other for the regularization term using pairs of frames. In general, there is no ensured overlapping in these two streams of samples. As a result, we usually cannot use features computed from an image to update both terms. This inefficiency is exacerbated by the fact that the regularizer requires pair inputs, making the training even less efficient. Secondly, the regularization term requires separate feature exchanges for each feature dimension. When m is

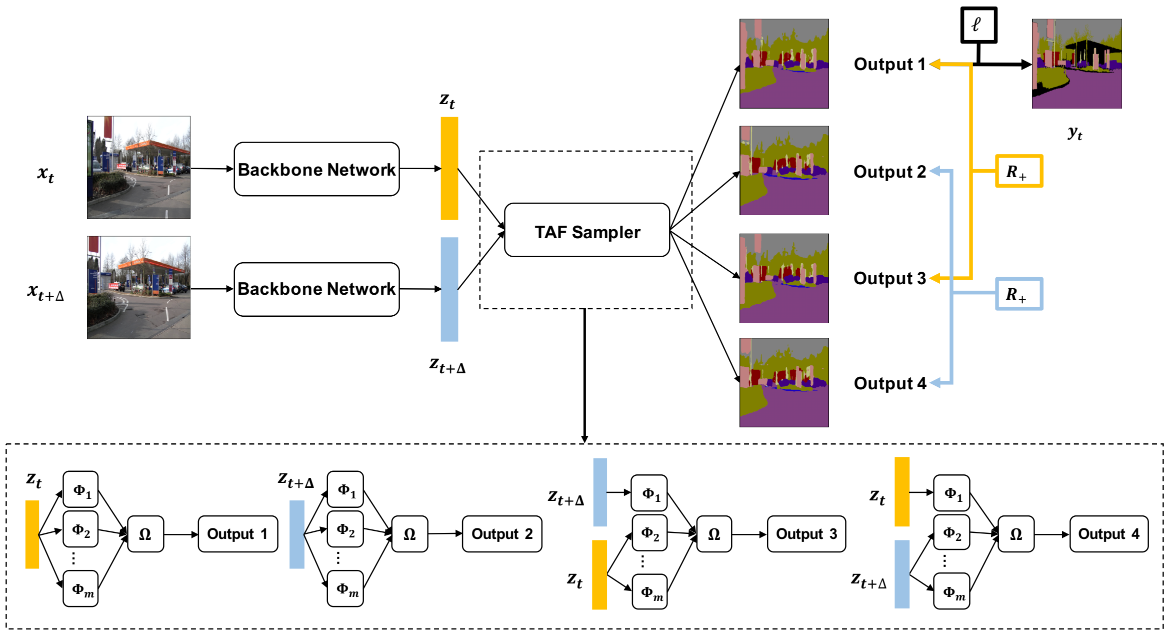


Figure 5.1: The sampling procedure of TAF learning for a single pair of images, in our implementation for semantic segmentation where Φ_i are defined on features extracted from a shared backbone.

large and the model decoupling requires re-computation of a significant portion of the model, this strategy is highly inefficient.

Our proposal is as follows: Within each mini-batch, a set of image-label tuples are first sampled from the annotated key frame subset $\mathcal{D}_{\text{key}}$. Each of these tuples are associated with a clip. Then, for each key frame sampled a random (unlabeled) pairing image is selected from the same clip by sampling the index difference between the key frame and the unlabeled pairing frame. In this improved procedure, all feature computations contribute to all terms in the objective function. To further make use of cached features, the regularization term is also made symmetric. To ensure tractable mini-batch updates, the summation over the dimensions are replaced by a uniform sampling of the dimension index i at each training example. The efficient TAF procedure solves the following problem

$$\min_{\Theta} \quad \mathbf{E}_{\substack{(x_t,y_t)\sim\mathcal{D}_{\text{key}},\\ n\sim\mathbf{U}(-n_h,n_h),\\ i_1,i_2\sim\mathbf{U}(1,m)}} \left[\ell(y(x_t;\Theta),y_t)+\lambda\left[R_+(x_t,i_1,n\Delta_0)+R_+(x_{t+n\Delta_0},i_2,-n\Delta_0)\right]\right] \tag{5.6}$$

where we simplify the notation by assuming that the key frame is always at the center of each clip. Figure 5.1 illustrates how the training objective is computed between a pair of sampled images.

The number of training iterations of TAF learning is two times of the baseline as only half of the mini-batch have ground truth labels. In order to compute the pair-wise loss, the aggregation function Ω has to be evaluated twice in each forward pass [1]. Thus for tractable training, Ω should be chosen to be a lightweight function. Figure 5.1 illustrates the proposed sampling procedure in the application of semantic segmentation, where we use a Siamese network for the backbone feature extractor and apply the TAF procedure only at the encoder layers (details in Section 5.2.3).

5.2.3 Application to Semantic Segmentation

We now provide a brief overview of two of the most popular semantic segmentation models and explain how our framework can be applied. The multi-branch structure that enable TAF learning for FCNs and DeepLab v3+ is used in a broader set of architectures for semantic segmentation [7, 8, 10, 15] and we expect similar modifications to be feasible.

FCNs Fully convolutional networks (FCNs) [6] is one of the earliest and most popular deep-learning based architecture for semantic segmentation. It follows a straightforward multi-scale design: Feature maps at the output of three different stages of a backbone convolutional network are extracted. Due to the downsampling operators between stages, feature maps have a decreasing spatial resolution (in the case of FCN8s that we consider the output stride equals to 8, 16 and 32, respectively). The features maps are converted into class logits maps via a single layer

[1]The first time using the original features, the second time using features after swapping.

of convolutions. The three predictions are aggregated via a cascade of upsampling and addition operations. In our modification of FCNs we assign $\Phi_1(x), \Phi_2(x)$ and $\Phi_3(x)$ to represent three feature maps, where $\Phi_1(x), \Phi_2(x), \Phi_3(x)$ represents the stride-32, stride-16 and stride-8 feature maps respectively. The aggregation function is the single convolution layer and the following cascaded addition operations. In practice, we find swapping $\Phi_1(x)$ is sufficient for improved accuracies over the baselines.

DeepLab v3+ DeepLab v3+ [4] is a recent semantic segmentation algorithm that has achieved state-of-the-art accuracy in challenging datasets such as Pascal VOC and Cityscapes. It follows an encoder-decoder structure, where the encoder is an ASPP module [5] that consists of five branches with different receptive fields (modeling structures at different scales): Four branches with varying dilation rates and an additional image pooling branch. Similar in spirit to the FCNs case, we assign Φ_1 to the image pooling branch, and $\Phi_2 \cdots, \Phi_5$ to the remaining branches starting from the one with largest dilation rate. The decoder is the aggregation function $\Omega(\cdot)$ in our formulation.

5.3 Related Works

Regularization and Data Augmentation in Deep Neural Networks There is a rich literature of generic regularization and data augmentation techniques sharing our goal of improving generalization, e.g. norm regularization [151, 152], reduction of co-adaptation [153–155] and pooling [156–158, 158]. For semantic segmentation, data augmentations techniques based on simple image transformations [2] are standard practices. Recently, [159–162] learn optimal transformations. These techniques are limited by not using video information but are complementary to our approach. We follow the default choice of regularization and random transformations when comparing TAF learning with corresponding baselines. Another effective solution is to

[2]such as horizontal flipping, random cropping, random jittering, random scaling and rotation

use generative models for data and label synthesis [163–166]. Similar to ours, these methods can improve model accuracy using unlabeled data. However, it could be intrinsically difficult to generate realistic and diverse data for complicated applications, while our method can directly utilize the large amount of real video clips.

Single Image and Video Semantic Segmentation We use semantic segmentation [4–10] as an example to verify our method as discussed in Section 5.3. Importantly, our method is quite different from the related literature of video semantic segmentation [135–144], where video clips are utilized at both training and inference. Our work learns models using video clips at training, but the model can be used on independent frames at inference. In semantic segmentation, unlabeled frames can be used via future frame predictions [141–144, 167] and label propagation [147, 148]. The former is only shown to improve video prediction results but not on single image predictions (as expected as future frame prediction is difficult from a single frame due to the lack of temporal context at test time). A few preliminary works suggest the latter can bring promising improvements to single frame predictions by generating pseudo labels [147–150]. But video propagation notably relies on manual screening and careful hyper-parameter tuning to reject low quality labels [147], otherwise it could surprisingly lead to performance degradation after including pseudo-labels in some cases [148]. Our method has the advantage of not requiring manual intervention. More importantly, our work suggests that regularizing the temporal behavior of features is an implicit form of video augmentation without explicit modeling of temporal dynamics, which compared to video propagation is a simpler pipeline and could be more transferable to other tasks.

Self-Supervised Learning Self-supervised learning utilizes the large amount of unlabeled data via carefully designed "pretext" tasks or constraints that aim at capturing meaningful real-world invariance structures in the data. The goal is to learn more robust features. Future frame prediction [141–144, 167], patch consistency via tracking [140], transitive invariance [168], temporal order verification [169] and motion consistency [170–172] have been proposed as useful pretext tasks. Recently, consistency across tasks are also explored [173, 174], although these methods do not consider videos. However, as pretext tasks usually differ from the target task, a separate transfer learning step is required. Ours in contrast can be used directly on the target task. Via imposing geometric constraints, several recent works use self-supervised learning to directly address real-world tasks, most notably in depth and motion predictions [20, 21, 23, 24, 51, 175, 175, 176]. However, these methods cannot transfer easily outside of their intended geometry application. Among them, SIGNet [51] points to a unified framework for self-supervised learning of both semantic and geometric tasks which would broaden the applications of this line of works, but the existing work can only improve on geometric tasks. In contrast, TAF is not restricted to any particular task by design. Our TAF framework generalizes the temporal coherence regularization in [145, 146] to multiple change rates and is the first to validate the utility of this form of regularization on challenging real-world applications. In contrast, the prior works focus on theoretical insights and are not rigorously validated.

5.4 Experiments

5.4.1 Datasets and Evaluation Metrics

We test our approach on two widely-used datasets for semantic segmentation: Camvid [177] and Cityscapes [178]. The images of both datasets are frames captured from videos. Detailed annotations are provided on key frames. The meta-data of the datasets include the source frame ids of the annotated frames which makes unlabeled frames within the same clips available. The availability of unlabeled frames in the said clip format makes these two datasets ideal for testing our TAF framework. In particular, Camvid consists of 367 clips for training and 101/233 images for val/test. Key frames from training and test set are captured at 1Hz and annotated with 11 object classes. We capture extra frames around the key frames at 30Hz using the provided raw video. Cityscapes consists of 2975 training key frames and 500 validation images. The

Table 5.1: Overall results on Camvid and Cityscapes (CS) datasets.

Method	**Output stride**	**Training set**	**Backbone**	**TAF**	**mIOU (%)**	**Pixel acc. (%)**
			Camvid test set			
FCN8s	8,16,32	Camvid	ResNet-50		65.3	91.0
FCN8s	8,16,32	Camvid	ResNet-50	✓	67.1	91.8
DeepLabV3+	16	Camvid	MobileNetV2		65.0	91.7
DeepLabV3+	16	Camvid	MobileNetV2	✓	66.9	91.9
DeepLabV3+	16	Camvid	ResNet-50		68.2	92.3
DeepLabV3+	16	Camvid	ResNet-50	✓	**69.1**	**92.3**
			Cityscapes validation set			
DeepLabV3+	16	CS-0.2	MobileNetV2		62.1	95.0
DeepLabV3+	16	CS-0.2	MobileNetV2	✓	63.8	95.3
DeepLabV3+	16	CS-0.2	ResNet50		67.7	95.7
DeepLabV3+	16	CS-0.2	ResNet50	✓	**69.4**	**96.0**
DeepLabV3+	16	CS-0.5	MobileNetV2		67.0	95.6
DeepLabV3+	16	CS-0.5	MobileNetV2	✓	68.1	95.8
DeepLabV3+	16	CS-0.5	ResNet50		71.7	96.2
DeepLabV3+	16	CS-0.5	ResNet50	✓	**73.0**	**96.3**

key frames are the 20-th frames in the provided 30-frame clips (30Hz) annotated with 19 object classes. In the interest of fast experimentation and to test our methods on small datasets, we sample 20% and 50% of the clips from Cityscapes training set, creating customary datasets with 595 and 1488 training clips respectively. We follow standard evaluation protocols and report mIOU and pixel accuracy on the held-out set.

5.4.2 Comparison to Baselines

To understand the advantage of the proposed method, we train FCNs and DeepLab v3+ models using the TAF learning paradigm on partially labelled clips and compare against fully supervised training using only key frames, on both Camvid and Cityscapes dataset. We use mean-average-error (L1 norm) for the contrastive loss as it is a common choice of image applications

[3]. We find it important to first normalize the per-pixel prediction via softmax function to avoid learning degenerate features. Our training hyper-parameters are detailed in the supplementary material. We ensure to use a comparable set of hyper-parameters for both the baseline and our method whenever is applicable. For FCN8s, we show results for performing feature swapping only on the stride-32 branch as this leads to better accuracy, while for DeepLab v3+ all branches are swapped with equal probabilities. Our main results are summarized in Table 5.1. The main finding is that our method improves over the respective baseline methods using both segmentation algorithms and on both datasets. We note that TAF learning only affects the training time procedures. At inference time models from TAF has exactly the same complexity as the baselines, ensuring that the improvements from TAF is not resultant from increased complexity.

5.4.3 Ablation Studies

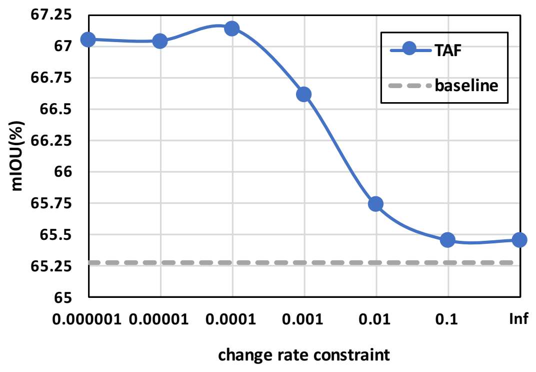


Figure 5.2: Effect of varying the change rate.

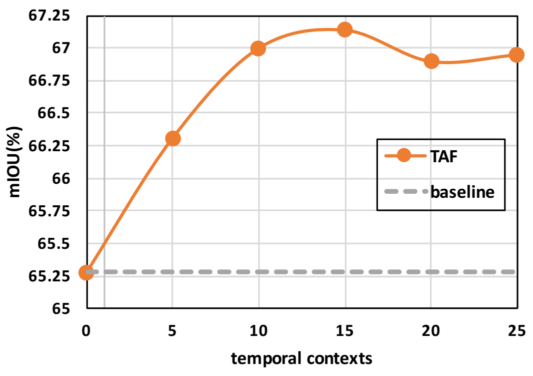
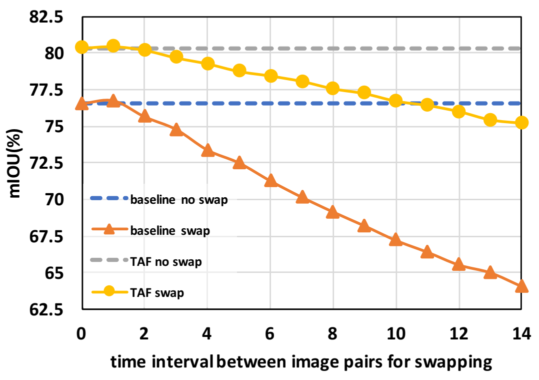


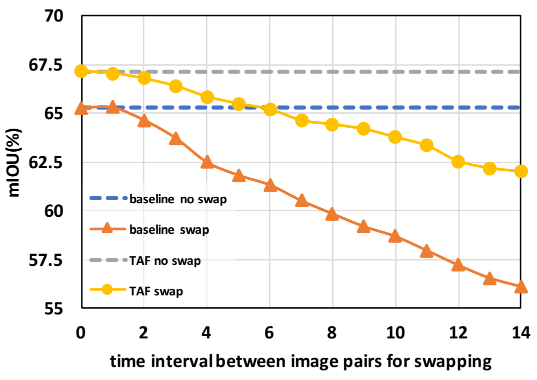
Figure 5.3: Effect of varying temporal contexts

To further understand the proposed method, we perform ablation studies on Camvid dataset using FCN8s models trained with TAF. Feature swapping is only performed on the stride-32 branch for simplicity. We choose to perform ablation studies on this model as its simple design can lead to clearer insights. We use $c_1 = 0.0001$ and the half size of each data clip (a.k.a. length of temporal context) to 15 unless specified otherwise in particular studies.

[3]This choice is also discussed in the supplementary material



(a) Result on train set



(b) Result on test set

Figure 5.4: Prediction mIOU as a function of feature swapping.

Study on Change Rates Constraint In our formulation c_i controls change rates of a particular feature dimension. It is interesting to observe the model performance as a function of c_i as this directly informs us on whether the change rate constraint is effective or not. When c_i is 0, the feature dimension in question will be forced to stay constant across frames. This should force it to learn features that are not informative to the final prediction, effectively reducing the model capacity and consequently, the prediction accuracy. On the other hand, as c_i goes to infinity the regularization term is effectively ignored, leading to sub-optimal results if the proposed regularization is indeed effective. Our finding as summarized in Figure 5.2 is as expected, verifying that the temporal change rate constraints are not trivially imposed.

Study on Temporal Context Temporal context refers to how far apart in time a pair of training examples can be. Note that in our derivation, we assume that the pair of frames used in the constraints are sufficiently close. This is important as when the two data points are too far apart, the first order approximation become ineffective. On the other extreme, when setting the temporal context to near zero, the regularization effect is diminished. Results from varying the temporal context are summarized in Figure 5.3. The mIOU reaches maximum when the temporal context is roughly 15 examples (the same setting used in our main results). Interestingly, while a larger temporal context leads to sub-optimal results, the degradation remains relatively mild, suggesting that precise approximation is not critical.

Study on Feature Swapping It is particularly interesting to see how the swapping of features between image pairs affects the prediction results. There are two trivial cases that deserve careful consideration: a) If the performance of the model (especially the baseline model) does not show a decrease in accuracy even with feature swapping, then our proposed constraints are not useful as this would suggest a natural tendency for part of the model to learn features that are insensitive to temporal changes. b) If the performance of the model does decrease after feature swapping, but both the baseline model and the TAF models demonstrate similar rate of degradation, then it would cast questions on whether the proposed constraint can actually be successfully imposed in the optimization. Furthermore, whether these constraints imposed on the training set can generalize to a test set. In Figure 5.4 we show that TAF learning does not result in the aforementioned trivial cases and can indeed generalize to held-out sets. Figure 5.5 further illustrates the effects of feature swapping.

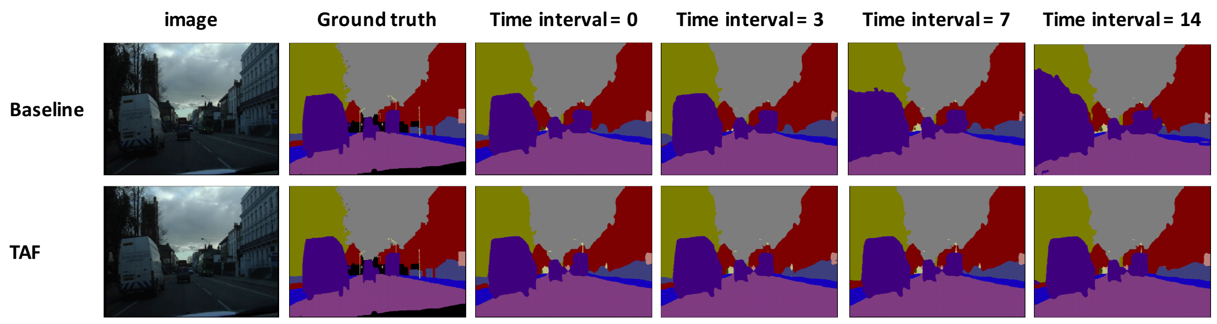


Figure 5.5: Visualization of predictions with feature swapping.

Study on Feature Attenuation Constraining the temporal change rate in features could lead to trivial solutions that are not discriminative [145]. In Figure 5.6 we show the effect of replacing the stride-32 branch either with its sample mean or zeros. The large resultant reduction in per-class accuracy suggests that TAF learning is not producing constant, trivial features as feared. However, this issue can be a function of model architectures and should be investigated further in future works.

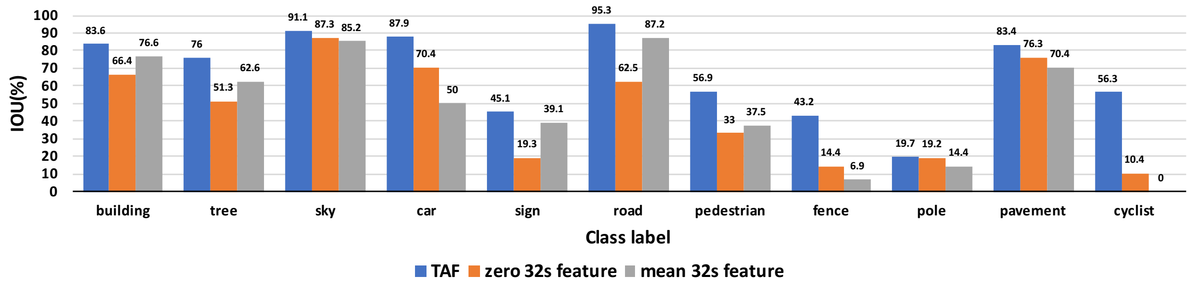


Figure 5.6: Change in prediction IOU after attenuating the stride-32 features.

5.5 Conclusion and Future Works

In this work, we propose to learn temporally-adaptive features to utilize partially annotated clips. Our proposed framework has demonstrated convincing gains on the challenging task of semantic segmentation. The ablation studies verify that our approach is learning non-trivial features that reflect the proposed temporal rate change constraints, validating our design choices. Our finding suggests the potential of such constraints in enabling self-supervised learning from clip data. It would be interesting to further validate the utility of this approach in related applications. Dense prediction tasks are natural starting points. Another interesting direction is to explore data-driven metrics, such as perceptual loss [179–181] and adversarial training [182], in constructing the contrastive loss, replacing the current heuristic choice of L1 norm. It is also interesting to further explore existing ideas from slow feature learning and video propagation (explicit label augmentation) to better model problem structures, which may in return lead to stronger results.

5.6 Appendix

5.6.1 Temporal Change Rates of Semantic Classes

We expect different semantic classes to demonstrate different temporal change rates. This as we discussed is a motivation for testing our method on semantic segmentation. To verify

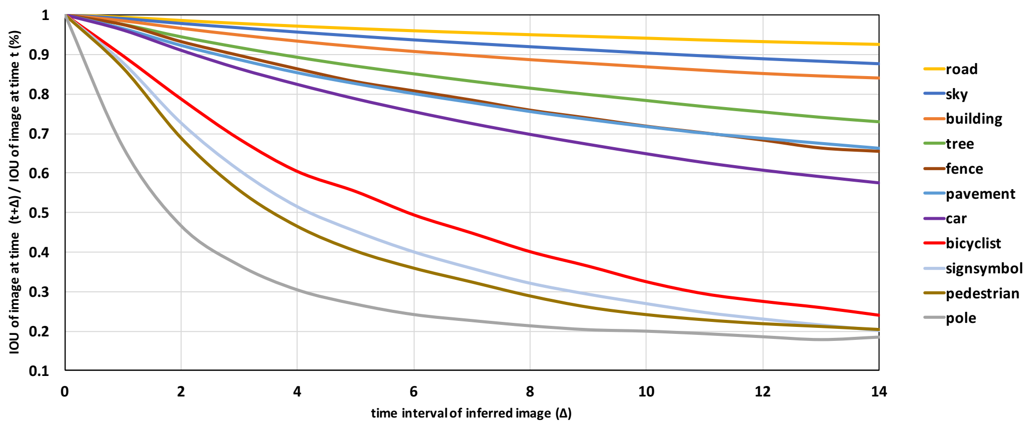


Figure 5.7: Changing rate of each class in Camvid train set. The figures show the ratio of prediction accuracy on x_t, between results using features from $x_{t+\Delta}$ and x_t, measured in per-class IOU.

it empirically, we compare the predicted segmentation labels at $x_{t+\Delta}$ against the ground truth label at x_t. Accuracy are reported using IOU normalized by the prediction accuracy at x_t, as shown in Figure 5.7. This normalization is necessary as different semantic classes have different intrinsic difficulties. Since labels are not available beyond the key frames, we use the model prediction instead in our study. Interestingly, there is a clear differentiation in the temporal change rates among different classes, as demonstrated by the large differences in change rates of the normalized IOU. Notably, the accuracies of larger or static objects such as road, sky, tree, fence, pavement tend to decrease slowly with time, suggesting low temporal change rates for those structures. On the other hand, smaller or moving objects like car, bicyclist, sign-symbol, pedestrian, pole tend change much faster with time.

5.6.2 Details of Training Procedures

We use mini-batch SGD optimizer with Nesterov momentum. We set momentum to 0.9 and weight decay to 0.0001. The batch size is 16 except for training models with ResNet50 backbone on Cityscapes, in which case due to GPU memory constraints we use batch size of 8. The training are performed on 4 Nvidia GTX 1080Ti GPUs. Synchronized batch normalization [4]

[4]Implementation: `https://github.com/vacancy/Synchronized-BatchNorm-PyTorch`

is used since the number of images per GPU is small in our setting. The ResNet-50 [5] [183] and MobileNet v2 [6] [184] models are pre-trained on ImageNet [185]. We adopt a learning schedule with polynomial decay with power set to 0.9, following standard practice in semantic segmentation. This schedule multiplies the initial learning rate by the factor $(1 - \frac{current\ epoch}{max\ epoch})^{power}$. During training, we apply random horizontal flip, random scales between 0.5 and 2 and random cropping. For both baselines and the TAF models, we report the best results among the initial learning rate from $\{0.005, 0.01, 0.02, 0.025, 0.05\}$ and additionally for ATF learning λ from $\{0.5, 1, 2\}$. Additional details are summarized in Table 5.2.

Table 5.2: Hyperparameters used for main results.

Method	Training set	Backbone	TAF	Init. lr	λ	Init. Img size	Crop size	Epoch	Val size
FCN8s	Camvid	ResNet-50		0.02	N/A	360×480	360×360	600	360×480
FCN8s	Camvid	ResNet-50	✓	0.02	1.0	360×480	360×360	600	360×480
DeepLabV3+	Camvid	MobileNetV2		0.02	N/A	360×480	360×360	600	360×480
DeepLabV3+	Camvid	MobileNetV2	✓	0.05	1.0	360×480	360×360	600	360×480
DeepLabV3+	Camvid	ResNet-50		0.02	N/A	360×480	360×360	600	360×480
DeepLabV3+	Camvid	ResNet-50	✓	0.05	0.5	360×480	360×360	600	360×480
DeepLabV3+	CS-0.2	MobileNetV2		0.02	N/A	1024×2048	720×720	300	1024×2048
DeepLabV3+	CS-0.2	MobileNetV2	✓	0.05	1.0	1024×2048	720×720	300	1024×2048
DeepLabV3+	CS-0.2	ResNet50		0.02	N/A	1024×2048	720×720	300	1024×2048
DeepLabV3+	CS-0.2	ResNet50	✓	0.05	1.0	1024×2048	720×720	300	1024×2048
DeepLabV3+	CS-0.5	MobileNetV2		0.02	N/A	1024×2048	720×720	300	1024×2048
DeepLabV3+	CS-0.5	MobileNetV2	✓	0.02	1.0	1024×2048	720×720	300	1024×2048
DeepLabV3+	CS-0.5	ResNet50		0.01	N/A	1024×2048	720×720	300	1024×2048
DeepLabV3+	CS-0.5	ResNet50	✓	0.01	1.0	1024×2048	720×720	300	1024×2048

5.6.3 Choice of Temporal Change Rates

For FCN8s, our preliminary studies suggest that assigning $c_1 = 0.0001$ and c_2, c_3 to inf leads to best performance. We note that this design effectively disables TAF learning on the stride-16 and stride-8 branches. This design is necessary to allow the two high resolution features to model structures with fast temporal change rates sufficiently. For DeepLab v3+, we assign

[5] Downloaded from `https://download.pytorch.org/models/resnet50-19c8e357.pth`

[6] Downloaded from `http://jeff95.me/models/mobilenet_v2-6a65762b.pth`

$c_1 = 0.000001, c_2 = 0.000001, c_3 = 0.0001, c_4 = 0.001, c_5 = 0.01$. There is no advantage in disabling TAF learning on any branch. In fact, our study suggests that it leads to worse accuracy. We think that can be attributable to the decoder (Ω function) design of the DeepLab v3+, which provides a skip connection with output stride of 4 from low level features and can model fast features sufficiently by itself.

5.6.4 Measure Temporal Change Rates Relative to Input

In our formulation the temporal change rates are directly measured by the variations in the predictive model y. However, different clips can have intrinsically different rates of motions, thus it might be wise to impose the temporal change rate constraints relative to the change rates in input images. This, as we also discuss in Section 5.2, is not trivial since $\frac{dx}{dt}$ is not available. In our preliminary studies, we empirically test using $L1$ norm as a measure of the change rate, using the first order approximation as we do for y. Then, we set the constraints as the proportion between the change rates in y and those in x. We find that this does not lead to improvement over the design we presented and the training is usually less stable. We conjecture that this is attributable to our heuristic method in measuring differences between images, a point worth revisiting in future works.

5.6.5 Choice of Loss Functions

The loss function consists of two parts: The semantic loss function and the contrastive loss. The former compares the prediction of the model against the ground truth annotations at the key frames, while the latter compares the prediction from the model before and after feature swapping (our regularization term). For the semantic loss function, we use cross-entropy loss for both the baseline and TAF learning, per standard practice. For the contrastive loss, in our preliminary studies we experiment with a few different metrics, including mean-squared-error

(MSE), mean-average-error (MAE, or L1 loss) as well as symmetric cross entropy. Among them, L1 loss leads to more stable training and best results. We note that it is important to first normalize the prediction at every pixel via a softmax function, as applying L1 norm regularization directly on the logits (before normalization) leads to degenerate solutions. We note that L1 norm is by no means the optimal choice of the metric to compare images, as it does not reflect the rich semantic structures encoded in natural images. We believe that perceptual loss or even adversarial loss (via a learnable model) could lead to better performance and are interesting future directions to explore.

Acknowledgment

Chapter 5, in full, is a reprint of the material that has been submitted for publication to *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, authored by Yongxi Lu, Ziyao Tang and Tara Javidi. The dissertation author was the primary investigator and author of this material.

Chapter 6

Efficient Video Understanding via Layered Multi Frame-Rate Analysis

6.1 Introduction

This chapter discusses a video prediction algorithm called *Domain Calibration Modulator*. It is an algorithm designed to produce per-frame outputs from a continuous stream of video. This algorithm is conceptually and technically closely related to the TAF framework, as discussed in detail in Chapter 5. Its similarity and differences with the TAF frameworks are summarized as follows.

The similarities with TAF:

- TAF are technically similar to DCM. The former can be seen as a generalization to the latter. The DCM network and the light-weight prediction network can be seen as Φ_1 and Φ_2 in the notation of the TAF framework. The temporal change rate constraints are $c_1 = 0$ and $c_2 = \inf$, respectively.
- DCM can be seen as an extension of TAF training procedure to cases where both training and testing can use videos. Both methods share the motivation adapting different parts of

the model to a distinct temporal change rate.

The differences with TAF:

- While TAF focus on the case where the video clips are partially annotated, in DCM all frames are fully labeled. As a result, the contrastive loss is not necessary in DCM. However, even in the video prediction application considered in this chapter, we expect the contrastive loss used in TAF framework be effective in cases where labels are missing for some frames.
- The DCM framework uses videos in both training and testing. TAF only considers per-frame prediction at testing, and does not affect the computational structure.
- The main focus of DCM is computational efficiency, while TAF is a framework designed for utilizing partially annotated clips.

The material covered in this chapter thus complements those presented in Chapter 5. It shows the benefit of making different parts of the models adaptive to a distinct temporal change rate is not restricted to enabling learning for partially annotated clips. With careful parameterization, this design principle could also lead to reduction in computational complexity when applied to video prediction tasks.

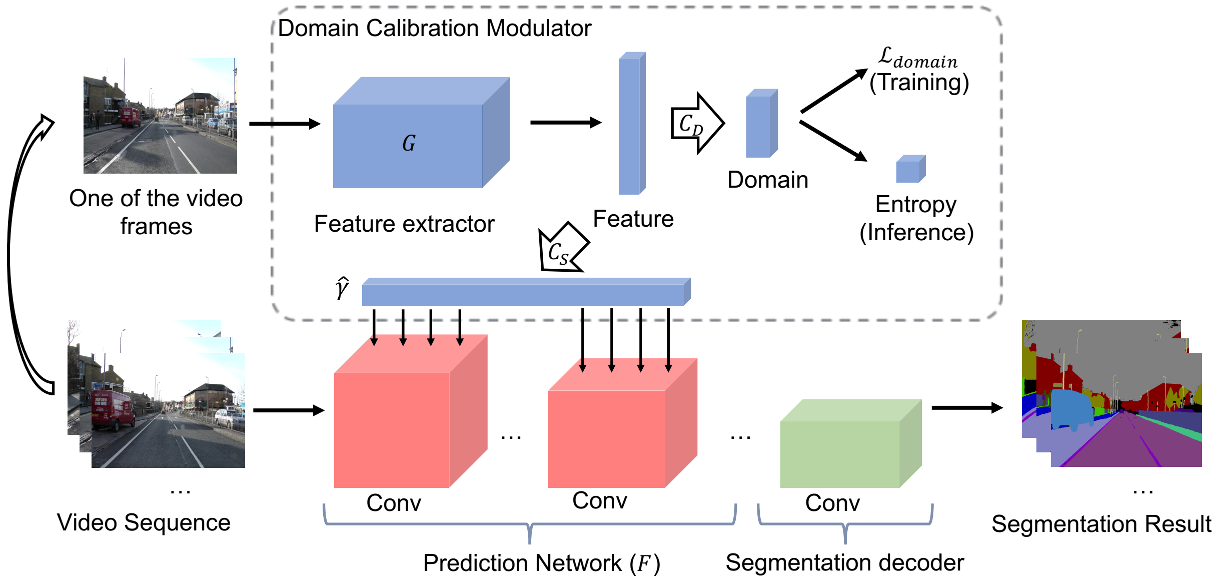


Figure 6.1: The pipeline of our proposed framework, using semantic segmentation as an example.

6.2 Related Work

Fast Neural Networks Architecture search [48, 186–190], network compression [191–193] and novel manual design [194–196] are popular methods in designing fast neural network architectures. These methods can find efficient architectures, but by focusing on single-frames they cannot utilize the strong correlation between nearby frames which are usually present in video understanding applications. Spatially adaptive processing [46, 197] and dynamic layer dropping [198–200] achieves efficient execution through conditional execution on part of the model or image, but these methods introduce additional overheads in decision makings and are less efficient in practice. Our solution is a simpler static architecture. Our method is also related to [201]. Similar to our method, they utilize the correlation between frames to achieve efficient processing. However, they focus on modeling the temporal consistency of labels. Our method focuses on separating environment modeling and per-frame modeling in a dual frame-rate system.

Domain Adaptation and Network Modulation Domain adaptation aims at improving generalization of models across different domains [202–212]. “Environment factors” we model in this work can be seen as a particular way to partition images into different domains. However, our work does not aim at improving generalization per se. Instead, it acknowledges that a light-weight model is unlikely to generalize well for all domains. To battle domain-shifts, an expensive model extracts robust features to augment the light-weight model through modulation. Our modulation algorithm is similar to Conditional Batch Normalization (CBN) [213–215], which has shown to be effective in controlling domain distributions, in applications such as style transfer and question answering. In our case, CBN is used to influence the feature distribution of the prediction network. A similar modulation pipeline is introduced in [216]. Similar to ours, this work modulates the feature extraction process at the current frame using features from a different frame. However, their approach is designed for one-shot learning. Our method addresses a different problem.

Attention Mechanisms Self-attention mechanisms [217, 218] leads to better generalization by learning automatic recalibration of features. Our method similarly calibrates the features of the light-weight network to improve its generalization. Unlike a self-attention mechanism, the features used to perform calibration in our case are not from the current image or model. They are extracted from a nearby frame using a different model.

Semantic Segmentation Our method is evaluated in a semantic segmentation task using the classical FCN decoder [219] as our baseline. Our contribution is complementary to state-of-the-art methods in segmentation, such as [219–221]. We expect to see similar accuracy gains from using our method on top of the latest segmentation models.

6.3 Methods

In this section we introduce our proposed system: A light-weight per-frame prediction model guided by a heavy-lifting domain calibration modulator (DCM). The per-frame prediction model is deployed at high frame-rate to catch all the subtle changes in each frame. It is trained to output the predicted labels (e.g. classification, detection or semantic segmentation etc.). The heavy-lifting domain calibration modulator (DCM) is deployed at a much lower frame-rate to reduce latency and power consumption, but it predicts reliable features which are fed into the prediction model through a "modulation" mechanism to improve task prediction robustness. Interestingly, this proposed framework consistently improve the test accuracy while adding only negligible amount of complexity per frame. We will present our empirical findings in the next section. The current section presents the basic structures of our framework.

6.3.1 Two-Stream Architecture

The goal of the proposed framework is to learn a function $\hat{y}(x;\theta)$ so as to minimize $\mathbb{E}[\mathcal{L}_D(y,\hat{y}(x;\theta))]$, where (x,y) are pairs of images and the associated task labels. The data consists of a number of domains $\{D_1,\cdots,D_k\}$. Images within the same domain are visually similar but could have very different labels.

In applications that use streaming inputs, short-term contiguous sequence of frames naturally falls into the same "domain". In view of this structure, to reduce complexity while improving prediction accuracy we decouple the predictor $\hat{y}$ into two components: A domain calibration modulator (DCM) and a prediction model. Figure 6.1 illustrates the framework using semantic segmentation task as an example. The DCM model extracts features shared across images in the same domain using an arbitrary sample from the domain. It then uses this feature to predict a set of calibration parameters $\hat{\gamma}_c$. The prediction model extracts fine-grained features from each frame. In the interest of fast processing, the prediction model is compact and as a result its features are less robust. This is compensated by a modulation mechanism which applies the calibration parameters to the prediction model. This process "calibrates" the inaccurate features extracted from the prediction model and turn them into robust features for the current domain.

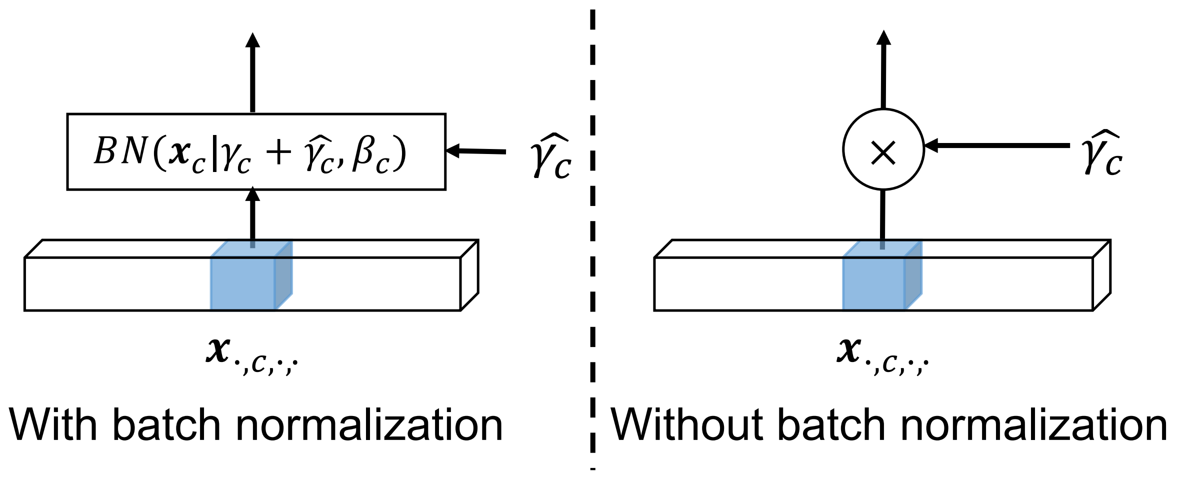

$BN(\boldsymbol{x}_c|\gamma_c + \widehat{\gamma_c}, \beta_c)$
$\widehat{\gamma_c}$
$\boldsymbol{x}_{\cdot,c,\cdot,\cdot}$
With batch normalization
$\times$
$\widehat{\gamma_c}$
$\boldsymbol{x}_{\cdot,c,\cdot,\cdot}$
Without batch normalization

Figure 6.2: Proposed modulator design in networks without BN layers (left) and with BN layers (right).

6.3.2 Domain Calibration Modulator

There are two important choices in the design of a DCM architecture, namely how are calibration features extracted and how can it be used to modulate the prediction network.

How are features extracted? The intermediate and output features extracted by a ConvNet architecture are multi-scale and dense. However, in DCM the features are out-of-sync with the current frame. Spatial details extracted should be suppressed or modulation based on it will be noisy and confusing since it is describing the incorrect contents. For this reason, we propose to use global average pooling from the last convolution layer to perform calibration feature extraction. The features extracted in this way are global descriptors, making them more suitable and stable for describing the environmental information.

How to modulate the prediction network using calibration features? The main consideration in this design choice is the trade-off between complexity and accuracy. If the modulation function is introducing large extra complexity, it will defeat the purpose of our design. We propose a modulation mechanism based on channel-wise scaling. This mechanism is orders of magnitude lighter than a convolution operator, yet it is shown to be very effective in other related controlling and re-calibrating the distribution of features [213–216, 218]. With this mechanism, our method attenuates or amplifies the contribution of feature channels based on the decoding of the environment by the DCM network. The DCM in effect adaptively selects the most useful features from the prediction network for the current domain. In our design, the modulation scalings are produced by a simple linear transformation of the extracted calibration features (from global average pooling).

The final design can be described in simple formulas. As shown in Figure 6.1, the proposed method first extracts features from a sample frame, then predict per-channel scales through a linear transformation

$$\hat{\gamma} = C_S(G(\mathbf{x})) \tag{6.1}$$

The per-channel scales are applied to the prediction network through element-wise modulation with a few selected intermediate layers. For prediction networks without Batch Normalization, the modulation mechanism is applied after every convolution layer. It can be written as

$$F(\mathbf{x}_{i,c,h,w}|\hat{\gamma}_c) = \hat{\gamma}_c \mathbf{x}_{\cdot,c,\cdot,\cdot} \tag{6.2}$$

For prediction networks with Batch Normalization (BN), these BN layers are replaced by

$$BN(\mathbf{x}_{i,c,h,w}|\gamma_c, \hat{\gamma}_c, \beta_c) = (\gamma_c + \hat{\gamma}_c)\frac{\mathbf{x}_{\cdot,c,\cdot,\cdot} - E[\mathbf{x}_{\cdot,c,\cdot,\cdot}]}{Var[\mathbf{x}_{\cdot,c,\cdot,\cdot}] + \varepsilon} + \beta_c \tag{6.3}$$

where γ_c is the original scale factor of the BN layer. In this case, the predicted scale is the residual of the modified BN scale factor. Using a residual scale is particularly useful when a pretrained model is used, as this avoid destructing the original BN parameters. The differences in design with and without BN are summarized in Figure 6.2.

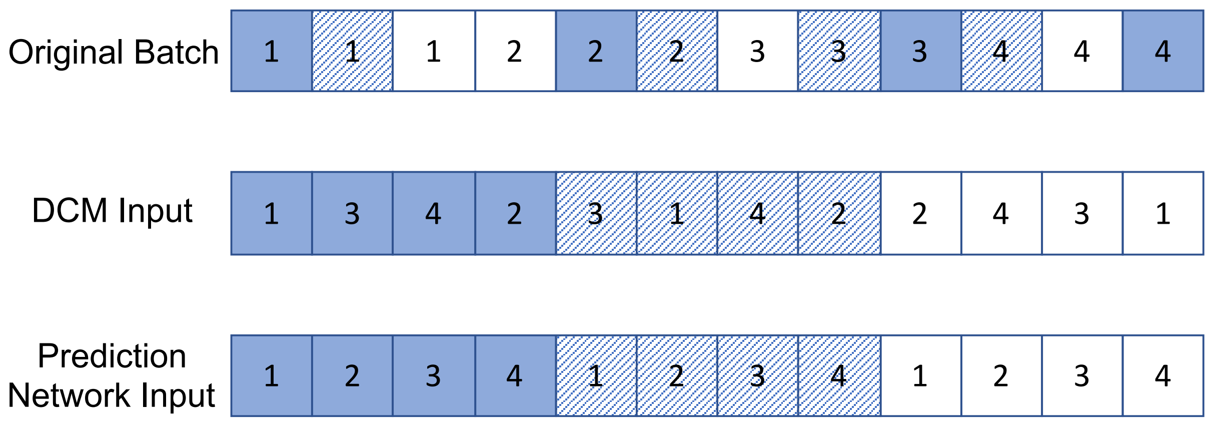


Figure 6.3: The sampling of training batches.

6.3.3 Model Training and Loss Functions

Training of our system requires image pairs from the same domains. In the applications we consider, the datasets are separated into video clips, each containing frames taken at similar time and locations. In both the training and testing procedures, these groups are considered "domains". This partition of data does not guarantee that all similar image pairs are classified into the same domain. But it suffices in providing image pairs that are likely to be visually similar. Based on this domain partition, our training procedure is essentially a two-step sampling process. It fills a training mini-batch in the following fashion (also see Figure 6.3):

1. Sample a set of images from the entire datasets. These images are used as input to the DCM. Their domain ids (video clip names) are recorded.

2. Group sampled images in step 1 according to domain ids. Re-shuffle samples within each group. The images are used as input to the prediction network using this new ordering.

The system can be trained end-to-end via a loss function that penalizes incorrect predictions at the output head of the prediction network. The DCM model is trained jointly with the prediction network as our proposed modulation mechanism is differentiable.

Features extracted for calibration should intuitively be similar if they are from two images in the same domain. To regularize the calibration features, in our framework, if there are k video clips, we train a k-way domain classifier using a simple linear layer from calibration features (shown as C_D in Figure 6.1). The video clip id is the domain label in this case. The entropy loss serves to regularize the calibration features. In addition, the entropy of the domain classifier prediction can also indicate the quality of the extracted domain features. This results in a more robust testing procedure for unexpected changes in the environment. Details of our testing procedure can be found in Section 6.3.4.

We denote the per-frame prediction network as F, and the domain label as d. The total loss for our framework is the sum of the task loss and the domain loss presented. The total loss function is

$$\begin{aligned}
\mathcal{L}_{total} &= \mathcal{L}_{domain} + \mathcal{L}_{task} \\
&= \frac{1}{N} \sum_{i,j \in [1,N]} \{\mathcal{L}(C_D(G(\mathbf{x}_j^d)), d) \\
&\quad + \mathcal{L}(F(\mathbf{x}_i^d), \mathbf{y}_i^d; \hat{\gamma})\} \\
&= \frac{1}{N} \sum_{i,j \in [1,N]} \{\mathcal{L}(C_D(G(\mathbf{x}_j^d)), d) \\
&\quad + \mathcal{L}(F(\mathbf{x}_i^d), \mathbf{y}_i^d; C_S(G(\mathbf{x}_j^d)))\}
\end{aligned} \tag{6.4}$$

6.3.4 Model Inference

A practical application of such a system is to use the first few frames at any operating sessions to extract “calibration” features using the DCM model. Assuming a slowly changing environment, the extracted calibration features are then re-used by the prediction model until the end of the particular deployment session. However, by the design of our training procedure as described in Section 6.3.3, our system should work equally well with out-of-order stream of inputs. It does not rely on the particular ordering of frames as long the inputs to the DCM and the prediction network are in the same domain. To confirm this conjecture, in our experiments we evaluate our system in both cases: (1) Feeding the DCM model with randomly selected inputs from the same domain (2) Feed the DCM model with the first few frames within the video clip. In both cases, our system demonstrates similarly strong improvements over the baselines methods.

An interesting observation from our empirical study is that if the domain prediction has high entropy, it strongly indicates a reduction in final test accuracy using this particular image for calibration. This could be caused by model failure, changes in environmental factors or a frame being not representational for a particular domain. In view of this, we propose a robust version of our naive testing procedure. In this alternative entropy-based testing, we perform a rejection sampling. If an incoming candidate input to DCM results in high entropy in domain prediction, we disqualify this candidate and sample another example. Our testing procedure can further improve average and worst-case performance of our system while using only 1-2 additional samples for the DCM, which is only a small increase in complexity.

Complexity at Inference We conclude this section with details in the computation of complexity of our testing procedure. Assuming that the DCM needs to process K images before extracting the final calibration features and a particular operation session (video clip) with N frames. Denote the computational cost per frame as M_{Pred} for the prediction network and M_{DCM} for the DCM. The average computational cost per frame (M_{frame}) is given by

$$M_{frame} = M_{Seg} + \frac{K}{N} M_{DCM} \tag{6.5}$$

In practical applications, $K \ll N$ in Equation 6.5. In such cases, our framework does not bring in much increase in average complexity. In our experiment sections, the complexity metrics reported are numbers of multiply-adds.

6.4 Experiments

To validate our design and test its applicability in real-world scenarios, we evaluate our framework on CamVid [222, 223] dataset. We test the robustness of our approach under a variety of combinations of popular ConvNet architectures. In all test cases we observe a consistent accuracy gain with negligible increase in complexity against baseline methods. This suggests our method can be applied in a wide variety of settings. Through extensive ablation studies, we conclude that the gain from our approach is not a trivial result of naively adding parameters. Nor does it result from learning the dataset average. Our analysis also shows that the proposed rejection sampling using entropy further improves the robustness of the system.

In the remainder of this section we will first discuss the choice of the dataset and data preprocessing details. We then discuss the details of our training and testing procedures and the exact modulation mechanisms used for different architectures and problem setups. It is followed by our main results on a variety of architectures, comparing our results against baselines available in the literature and our own replications. This section is concluded with ablation studies that sheds lights on how our system improves the performance and its most salient failure modes.

6.4.1 Dataset and Preprocessing

CamVid [222, 223] is a segmentation dataset with 11 ground truth classes including sky, building, pole, road, pavement, tree, sign-symbol, fence, car, pedestrian, bicyclist. The images are all taken at street-level. It consists of four video sequences taken in different locations and times. We use the same training, testing and validation split as in [224]. We use the training split for training and the testing split for evaluation.

This dataset is ideally suited for evaluating our method: (1) It is separated into a few videos, ideally simulating the scenario in which environmental factors are persistent within each video clip. (2) It has full annotations of every frames of the videos, making it suitable for performing extensive ablation studies of our proposed framework. While there are other large-scale datasets in similar applications, their configurations are not suitable for testing our particular framework. However, conclusions drawn from CamVid should still be informative for future works on large-scale evaluations.

Baseline methods [224, 225] processes images at 360×480 and 720×960. In our

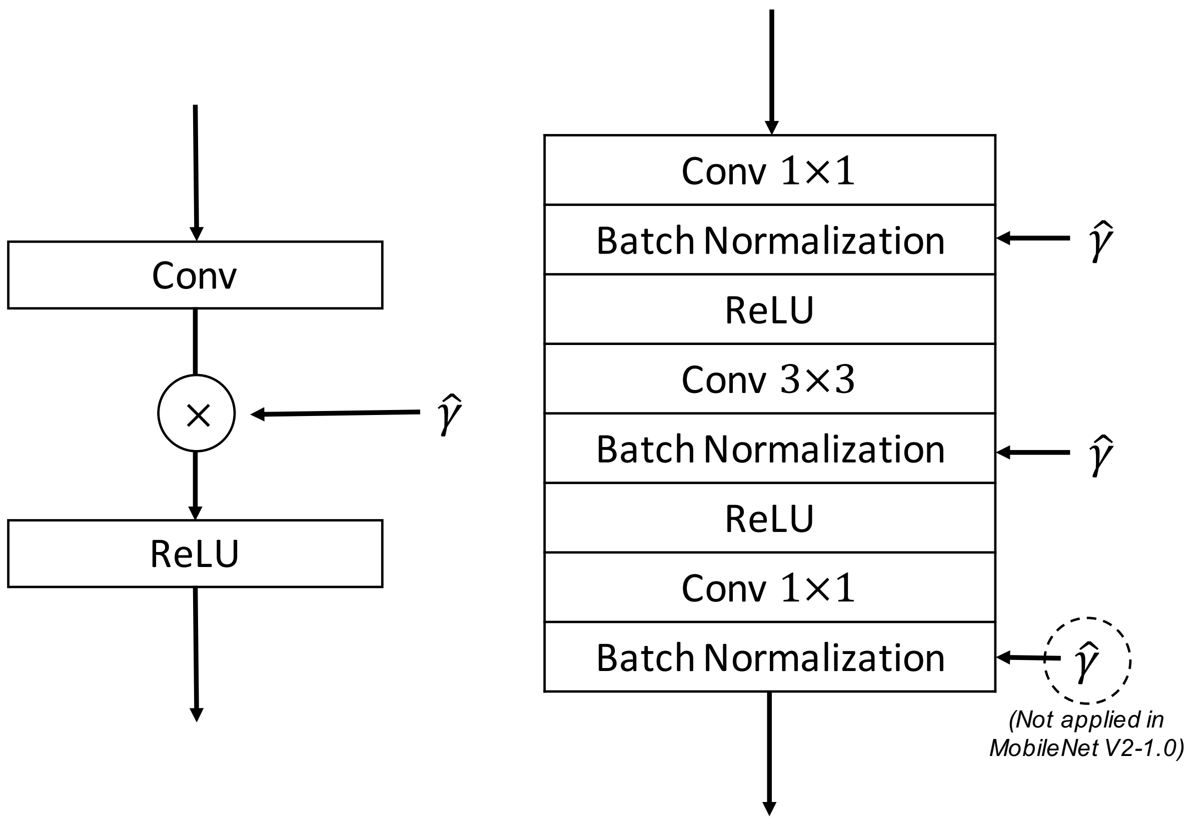


Figure 6.4: The modified convolution layer from AlexNet (left) and modified inverted residual unit from MobileNet V2 (right).

evaluations both settings are tested. In training time, we apply random crops of 352×352 for 360×480 images, and 704×704 for 720×960 images, respectively. Random horizontal flip is used as in standard practices. Inputs to DCM are scaled to 224×224.

6.4.2 Network Architectures

In experiments on CamVid, the DCM backbones are either ResNet50 [226] or MobileNet V2-1.0 [196]. We perform global average pooling at the top convolution layer to extract calibration features, in a manner described in Section 6.3.2 and Figure 6.1.

The prediction networks use fully convolution networks (FCNs) [219] decoders. The decoders follow the FCN32s configuration. To test performance of compact networks, the backbone architectures are AlexNet [227] and MobileNet V2 with different width factor. For

Table 6.1: Result for AlexNet-FCN32s experiment.

Eval Img. denotes to average number of images the DCM processed.

Backbone	Img. Size	Pretrained	mIOU(%)	Eval Img.	Mult-Adds/Img.
AlexNet(baseline)			41.2	–	16.02G
AlexNet+DCM(Ours)	360×480	$\times$	44.1($\pm$0.42)	2	16.05G
AlexNet+DCM+Entropy(Ours)			**44.4**($\pm$0.43)	4.55	16.10G
AlexNet(baseline)			50.6	–	16.02G
AlexNet+DCM(Ours)	360×480	✓	53.0($\pm$0.18)	2	16.05G
AlexNet+DCM+Entropy(Ours)			**53.1**($\pm$0.15)	3.55	16.08G
AlexNet(baseline)			44.0 [224]	–	54.79G
AlexNet(baseline, Ours)	720×960	$\times$	48.7	–	54.79G
AlexNet+DCM(Ours)			51.7($\pm$0.75)	2	54.82G
AlexNet+DCM+Entropy(Ours)			**51.8**($\pm$0.90)	3.75	54.89G
AlexNet(baseline)			57.4 [224]	–	54.79G
AlexNet(baseline, Ours)	720×960	✓	57.2	–	54.79G
AlexNet+DCM(Ours)			60.8($\pm$0.38)	2	54.82G
AlexNet+DCM+Entropy(Ours)			**60.9**($\pm$0.37)	3.1	54.84G

AlexNet backbones, channel-wise modulation is applied after every convolution layers. For MobileNet V2 backbones, by default DCN modulates every BN layers in the inverted residual blocks. However, for MobileNet V2-1.0 we omit modulation for the last convolution layer in each inverted residual block. Figure 6.4 summarizes the modifications.

6.4.3 Training and Testing Procedures

Training Procedure and Hyperparameters We train all models with a batch size of 12 on 3 Nvidia GTX 1080Ti GPUs. We use Adam optimizer [228] with weight decay of 0.0005. Unless specified otherwise, we train for 600 epochs. When the DCM model uses ResNet50 backbones, we use a step-wise learning rate schedule with initial learning rate of 0.0001 and reduces it to 0.00001 at epoch 400. When the DCM model uses MobileNet v2-1.0 backbone, the initial learning rate is 0.00005 with reduction to 0.000005. The training mini-batches are sampled according to the procedure described in Section 6.3.3.

The scale prediction layer of the DCM (C_s) is a fully connected layer initialized with all-zero weights and the bias is set to 1. This ensure that the initial output of the DCM is always 1. In MobileNet based prediction networks, if a pretrained model is used, the scaling factor in the

Table 6.2: Result for MobileNet-FCN32s experiment.

Eval Img. denotes to average number of images the DCM processed. MBNet denotes MobileNet.

Backbone	Img. Size	Pretrained	mIOU(%)	Eval Img.	Mult-Adds/Img.
MBNet-1.0(baseline)			58.4	–	2747.49M
MBNet-1.0+DCM(Ours)	360×480	✓	59.2($\pm$0.70)	2	2750.38M
MBNet-1.0+DCM+Entropy(Ours)			**59.6**($\pm$0.08)	2.45	2751.04M
MBNet-0.75(baseline)			58.1	–	1980.49M
MBNet-0.75+DCM(Ours)	360×480	✓	59.4($\pm$0.23)	2	1983.36M
MBNet-0.75+DCM+Entropy(Ours)			**59.5**($\pm$0.16)	2.35	1983.87M
MBNet-0.5(baseline)			54.4	–	1069.92M
MBNet-0.5+DCM(Ours)	360×480	✓	56.6($\pm$0.40)	2	1072.75M
MBNet-0.5+DCM+Entropy(Ours)			**56.9**($\pm$0.07)	3.15	1074.37M
MBNet-0.35(baseline)			51.8	–	770.34M
MBNet-0.35+DCM(Ours)	360×480	✓	52.0($\pm$1.92)	2	773.14M
MBNet-0.35+DCM+Entropy(Ours)			**53.0**($\pm$0.21)	2.5	773.85M

BN layers are subtracted by 1 at initialization. This modification ensures the BN layers are not changed initially. We find it important to fix the scaling factor term in BN layer during training or the DCM could learn trivial solutions.

Testing Procedure As discussed in Section 6.3.4, we use random as well as the first few images from the same video clip as input to the DCM model. For random sampling the procedure we report both the mean and the std of the mIOU from 20 repetitions. When rejection sampling with entropy is used, the number of DCM evaluations is dynamic. In those cases, we also report the average number of evaluations and computational complexity (in number of Multiply-Adds). Although both testing procedures are evaluated, we report the result from random shuffling by default as it results in a larger test set, making the accuracy numbers more robust. However, our ablation study shows that as expected, there is no significant difference in the two testing procedures.

6.4.4 Comparison with Baselines

AlexNet FCN32s AlexNet is used to build baseline methods on CamVid segmentation [224]. In this set of experiments, we use a DCM model based on ResNet-50 backbones. The prediction network is AlexNet FCN32s. The DCM model is always initialized from weights pretrained on ImageNet. For our experiments using the entropy trick, the threshold for rejecting a DCM input is set to 0.1.

Table 6.1 summarizes our results. We carefully compare against best available baseline results reported in the literature. For pretrained AlexNet FCN32s model on 720×960 images, our baseline is comparable to the reported number in [224]. When training from scratch, our baseline is better than their reported number in the corresponding setting. The proposed method outperforms the baselines by 2-3% with negligible increase in average complexity in all settings. Large gains are observed for both small picture input as well as large picture input. Although we use ImageNet pretrained models in DCM, we observe that the gain is similar regardless of the initialization method used in the AlexNet backbone. This suggests that the gains are not the result of feature transfer from ImageNet.

In this experimental setting, the quality of calibration features extracted by DCM is high. As long as the DCM input is from the correct domain, the modulation process almost always leads to improvements in testing accuracy. For this reason, the entropy trick for DCM input selection is not effective in improving the testing accuracy. This is in contrast in our findings from using MobileNet V2 backbones, in which case the entropy brings in significant gains.

MobileNet FCN32s To further evaluate our method in mobile applications, we experiment with models using MobileNet V2 [196] as backbones. MobileNet V2 is a recently proposed light-weight architecture designed for mobile applications. It significantly reduces model complexity by adoption of depth-wise convolution and inverted residual units. In our experiments, we use a MobileNet V2-1.0 model as the backbone network of DCM. The prediction networks at FCN32s models using MobileNet V2 models with different width factors. The DCM model is initialized using weights pretrained on CamVid, while the prediction network model uses weights pretrained on ImageNet.

Table 6.2 compares our algorithm against baselines. Notably, MobileNet V2 baseline models out-performs AlexNet models despite its much smaller complexity. Our approach further improves the mIOU by 1-2% with insignificant increase in complexity. The gain seems to increase when the backbone network has smaller width. In particular, the gain using $\{1.0, 0.75, 0.5, 0.35\}$ are 1.0%, 1.4%, 2.5% and 1.2% respectively. Intuitively, as the prediction model becomes smaller, it should be more difficult for the model to generalize to different domains, thus the benefit from using the DCM model is larger. Our result suggests that this is the general trend. However, when the model is too small (in the case of MobileNet V2-0.35), the benefit starts to diminish. The authors conjectures that when the prediction network is too small, our channel-wise feature modulation mechanism is not powerful enough to significantly improve the feature quality.

Another interesting observation for is that in experiments using MobileNet V2 backbones, the standard deviations in the prediction accuracy is larger than those in AlexNet. In particular, the standard deviation in prediction accuracy reaches 1.92% for the model based on MobileNet V2-0.35. This is likely caused by the fact that MobileNet V2-0.35 is a smaller architecture. Thus, an incorrect modulation signal from DCM could have a greater impact to its prediction accuracy. When sub-optimal DCM inputs are used, modulating the prediction networks leads to accuracy numbers that are below the baselines. In this context, we find that high entropy of the domain classifier (C_D in Figure 6.1) can reliably indicate a non-ideal input to DCM. We test the rejection sampling procedure using an entropy threshold of 0.05. This improves accuracy in all settings, but the gain for MobileNet V2-0.35 is a particularly large at 1%. It also leads to a more predictable algorithm at test time as can be concluded from the largely reduced standard deviation in mIOU numbers.

6.4.5 Ablation Studies

To further understand the proposed method, we use ablation studies to answer the following questions.

Can DCM leads to performance gain if the input are the first few frames? To answer this question, we compare the two types of inputs (first frames versus random frames).

Table 6.3: Use starting frames in a video instead of randomly selected images for DCM input.

The numbers enclosed in parenthesis show the relative change in numbers against the random selection counterparts. Eval Img. denotes to number of images the DCM processed.

Backbone	Img. Size	Pretrained	Eval Img.	mIOU(%)
AlexNet	360×480	×	$4(-0.55)$	$44.7(+0.3)$
AlexNet	360×480	✓	$3(-0.55)$	$53.1(+0.0)$
AlexNet	720×960	×	$2(-1.75)$	$50.8(-1.0)$
AlexNet	720×960	✓	$3(-0.1)$	$61.1(+0.2)$
MobileV2-1.0	360×480	✓	$3(+0.55)$	$59.5(-0.1)$
MobileV2-0.75	360×480	✓	$2(-0.35)$	$59.6(+0.1)$
MobileV2-0.5	360×480	✓	$3(-0.15)$	$56.9(+0.0)$
MobileV2-0.35	360×480	✓	$3(+0.5)$	$52.9(-0.1)$

Table 6.4: Effects of inputs with wrong domains.

Δ denotes to the difference between ablation study result and our method with using backbone Network+DCM+entropy.

Backbone	Img. Size	Pretrained	mIOU(%)	Δ(%)
AlexNet	360×480	×	$37.1(\pm 1.31)$	-7.3
AlexNet	360×480	✓	$51.2(\pm 0.21)$	-1.9
AlexNet	720×960	×	$41.8(\pm 1.30)$	-10.0
AlexNet	720×960	✓	$59.6(\pm 0.31)$	-1.3
MobileV2-1.0	360×480	✓	$54.4(\pm 1.05)$	-5.2
MobileV2-0.75	360×480	✓	$52.6(\pm 1.20)$	-6.9
MobileV2-0.5	360×480	✓	$50.2(\pm 0.72)$	-6.7
MobileV2-0.35	360×480	✓	$44.3(\pm 1.23)$	-8.7

Table 6.3 summarizes the comparison. It is clear that the choice of input type does not result in significantly different accuracy or complexity.

What happens if the DCM input is from the wrong domain? This is an interesting sanity check: If wrong inputs to the DCM does not lead to significant reduction in accuracy, then DCM might be learning producing trivial modulation signals. From Table 6.4, using input images with wrong domain ids results in up to 10% drop in mIOU, dispriving this possibility.

What happens if we add additional scaling parameters on AlexNet? Since AlexNet does not have BN layers, our modulation mechanism effectively adds new parameters to the network. To rule out a trivial gain resultant from added parameters, we perform a comparison. As can be seen in Table 6.5, adding parameters alone does not improve accuracy significantly.

What happens if there is domain mismatches in training? It is reasonable to suspect that the system is simply benefiting from added parameters from the DCM model (although the inputs are out-of-sync to the current frame). To rule out this trivial case, we purposely create mismatches in domain ids between the input to DCM and the prediction network. Our experiment summarized in 6.6 shows that our method significantly outperforms models trained with domain mismatches, disproving another trivial case.

The answer to the last three questions strongly suggests that our method is indeed using the domain information in a non-trivial manner. Based on these results, we conclude that: (1) Gains from our approach are not a trivial result of naively adding parameters. (2) Nor does it result from learning the degenerate solutions such as dataset means.

Table 6.5: Effects of adding channel-wise multiplication parameters in the AlexNet. Δ denotes to the difference between ablation study result and our method with AlexNet+DCM+entropy.

Backbone	Img. Size	Pretrained	mIOU(%)	Δ(%)
AlexNet+param	360×480	×	42.3	-2.2
AlexNet+param	360×480	✓	49.8	-3.3
AlexNet+param	720×960	×	49.5	-2.3
AlexNet+param	720×960	✓	55.2	-5.7

Table 6.6: Effect of domain mismatches in DCM training.

Δ denotes to the difference between ablation study result and our method with using backbone Network+DCM+entropy.

Backbone	Img. Size	Pretrained	mIOU(%)	Δ(%)
AlexNet	360×480	$\times$	43.3($\pm$0.05)	-1.1
AlexNet	360×480	✓	51.8($\pm$0.02)	-1.3
MobileV2-1.0	360×480	✓	59.7($\pm$0.16)	$+0.1$
MobileV2-0.75	360×480	✓	58.1($\pm$0.05)	-1.4
MobileV2-0.5	360×480	✓	55.1($\pm$0.10)	-1.8
MobileV2-0.35	360×480	✓	52.2($\pm$0.10)	-0.8

6.5 Conclusion and Future Directions

In this work, we propose and empirically investigate a novel dual frame-rate architecture for efficient video understanding. This strategy has demonstrated consistent gains over baselines, over a wide variety of settings. Through ablation studies, we show that the success is due to accurate modeling of the environment. We also propose practical solutions to improve the robustness of our algorithm when the environmental modeling is inaccurate.

The current work is limited by the size of the dataset used for evaluation. An important future work is to curate a large-scale dataset to evaluate similar design principles. It is also interesting to test its applicability to a wider variety of applications. In this work we use semantic segmentation on video clips as an example application. The authors expect the same strategy would lead to improvements in related applications such as object detection and instance level segmentation, as the problem structures and constraints are similar. Another interesting direction is to find the “optimal” modulation strategy and a potential generalization of the two-stream design. In fact, temporally visual signals usually exhibits multi-scale structures, just as they do spatially. It would be interesting to go beyond the hand-crafted dual frame-rate design to a truly adaptive multi frame-rate system.

Acknowledgement

Chapter 6 is substantially based on unpublished material that is available as an arXiv preprint, authored by Ziyao Tang, Yongxi Lu and Tara Javidi. The dissertation author was a joint primary investigator and author of this material.

Chapter 7

SIGNet: Semantic Instance Aided Unsupervised 3D Geometry Perception

7.1 Introduction

Visual perception of 3D scene geometry using a monocular camera is a fundamental problem with numerous applications, like autonomous driving and space exploration. We focus on the ability to infer accurate geometry (depth and flow) of static and moving objects in a 3D scene. Supervised deep learning models have been proposed for geometry predictions, yielding "robust" and favorable results against the traditional approaches (SfM) [229–234]. However, supervised models require a dataset labeled with geometrically informative annotations, which is extremely challenging as the collection of geometrically annotated ground truth (e.g. depth, flow) requires expensive equipment (e.g. LIDAR) and careful calibration procedures.

Recent works combine the geometric-based SfM methods with end-to-end unsupervised trainable deep models to utilize abundantly available unlabeled monocular camera data. In [3, 235–237] deep models predict depth and flow per pixel simultaneously from a short sequence of images and typically use photo-metric reconstruction loss of a target scene from neighboring

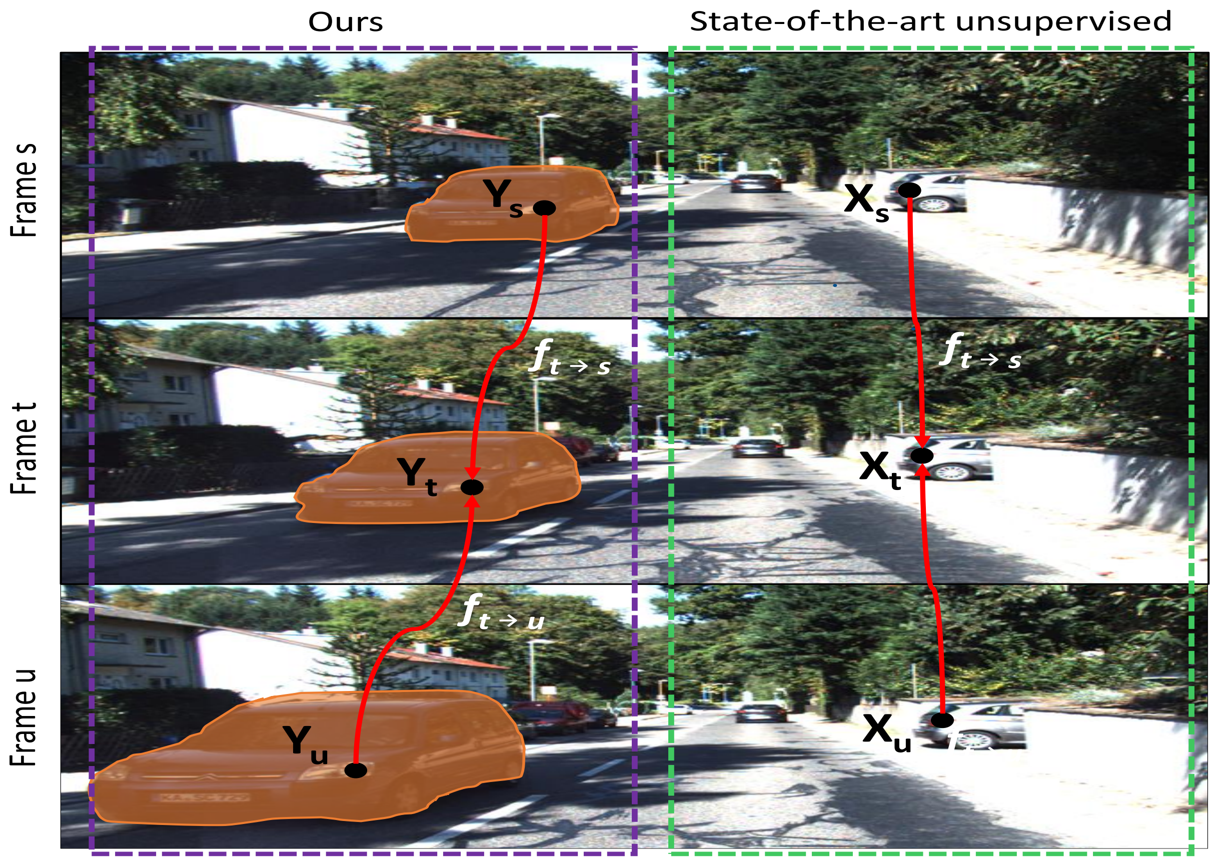


Figure 7.1: On the right, state-of-the-art unsupervised learning approach relies on pixel-wise information only, while SIGNet on the left utilizes the semantic information to encode the spatial constraints hence further enhances the geometry prediction.

scenes as the surrogate task. However, these solutions often fail when dealing with dynamic objects[1]. *Furthermore, the prediction quality is negatively affected by the imperfections like Lambertian reflectance and varying intensity which occur in the real world. In short, no robust solution is known.*

In Fig 7.1, we highlight the innovation of our system (on the left) comparing to the existing unsupervised frameworks (on the right) for geometry perception. Traditional unsupervised models learn from the pixel-level feedback (i.e. photo-metric reconstruction loss), whereas SIGNet relies on the key observation that inherent spatial constraints exist in the visual perception problem as shown in Fig 7.1. Specifically, we exploit the fact that pixels belonging to the same object have

[1]Section 7.5 presents empirical results that explicitly illustrate this shortcoming of state-of-the-art unsupervised approaches.

additional constraints for the depth and flow prediction.

How can those spatial constraints of the pixels be encoded? We leverage the semantic information as seen in Fig 7.1 for unsupervised frameworks. Intuitively, semantic information can be interpreted as defining boundaries around a group of pixels whose geometry is closely related. The knowledge of semantic information between different segments of a scene could allow us to easily learn which pixels are correlated, while the object edges could imply sharp depth transition. Furthermore, note that this learning paradigm is practical [2] as annotations for semantic prediction tasks such as semantic segmentation are relatively cheaper and easier to acquire. To the best of our knowledge, our work is the first to utilize semantic information in the context of unsupervised learning for geometry perception.

A natural question is how do we combine semantic information with an unsupervised geometric prediction? Our approach to combine the semantic information with RGB input is two-fold: First, we propose a novel way to augment RGB images with semantic information. Second, we propose new loss functions, architecture, and training method. The two-fold approach precisely accounts for spatial constraints in making geometric predictions:

Feature Augmentation We concatenate the RGB input data with both per-pixel class predictions and instance-level predictions. We use per pixel class predictions to define semantic mask which serves as a guidance signal that eases unsupervised geometric predictions. Moreover, we use the instance-level prediction and split them into two inputs, instance edges and object masks. Instance edges and object masks enable the network to learn the object edges and sharp depth transitions.

Loss Function Augmentation Second, we augment the loss function to include various semantic losses, which reduces the reliance on semantic features in the evaluation phase. This is crucial when the environment contains less common contextual elements (like in dessert navigation or mining exploitation). We design and experiment with various semantic losses, such

[2] Semantic labels can be easily curated on demand on unlabeled data. On the contrary, geometrically informative labels such as flow and depth require additional sensors and careful annotation at the data collection stage.

as semantic warp loss, masked reconstruction loss, and semantic-aware edge smoothness loss. However, manually designing a loss term which can improve the performance over the feature augmentation technique turns out to be very difficult. The challenge comes from the lack of understanding of error distributions because we are generally biased towards simple, interpretable loss functions that can be sub-optimal in unsupervised learning. Hence, we propose an alternative approach of incorporating a transfer network that learns how to predict semantic mask via a semantic reconstruction loss and provides feedback to improve the depth and pose estimations, which shows considerable improvements in depth and flow prediction.

We empirically evaluate the feature and loss function augmentations on KITTI dataset [238] and compare them with the state-of-the-art unsupervised learning framework [3]. In our experiments we use class-level predictions from DeepLabv3+ [239] trained on Cityscapes [178] and Mask R-CNN [240] trained on MSCOCO [64]. Our key findings:

- By using semantic segmentation for both feature and loss augmentation, our proposed algorithms improves squared relative error in depth estimation by 28% compared to the strong baseline set by state-of-the-art unsupervised GeoNet [3].
- Feature augmentation alone, combining semantic with instance-level information, leads to larger gains. With both class-level and instance-level features, the squared relative error of the depth predictions improves by 30% compared to the baseline.
- Finally, as for common dynamic object classes (e.g. vehicles) SIGNet shows 39% improvement (in squared relative error) for depth predictions and 29% improvement in the flow prediction, thereby showing that semantic information is very useful for improving the performance in the dynamic categories of objects. Furthermore, SIGNet is robust to noise in image intensity compared to the baseline.

7.2 Related Work

Deep Models for Understanding Geometry Deep models have been widely used in supervised depth estimation [2, 241–248], tracking, and pose estimation [232, 249–251] , as well as optical flow predictions [252–255]. These models have demonstrated superior accuracy and typically faster speed in modern hardware platforms (especially in the case of optical flow estimation) compared to traditional methods. However, achieving good performance with supervised learning requires a large amount of geometry-related labels. The current work addresses this challenge by adopting an unsupervised learning framework for depth, pose, and optical flow estimations.

Deep Models for Semantic Predictions Deep models are widely applied in semantic prediction tasks, such as image classification [256], semantic segmentation [239], and instance segmentation [240]. In this work, we utilize the effectiveness of the semantic predictions provided by DeepLab v3+ [239] and Mask R-CNN [240] in encoding spatial constraints to accurately predict geometric attributes such as depth and flow. While we particularly choose [239] and [240] for our SIGNet, similar gains can be obtained by using other state-of-the-art semantic prediction methods.

Unsupervised Deep Models for Understanding Geometry Several recent methods propose to use unsupervised learning for geometry understanding. In particular, Garg *et al.*[24] uses a warping method based on Taylor expansion. In the context of unsupervised flow prediction, Yu *et al.*[257] and Ren *et al.*[258] introduce image reconstruction loss with spatial smoothness constraints. Similar methods are used in Zhou *et al.*[235] for learning depth and camera egomotions by ignoring object motions. This is partially addressed by Vijayanarasimhan *et al.*[236], despite the fact, we note, that the modeling of motion is difficult without introducing semantic information. This framework is further improved with better modeling of the geometry. Geometric consistency loss is introduced to handle occluded regions, in binocular depth learning [259], flow prediction [260] and joint depth, ego-motion and optical flow learning [3]. Mahjourian *et al.*[175] focuses on improved geometric constraints, Godard *et al.*[261] proposes several architectural and loss innovations, while Zhan *et al.*[262] uses reconstruction in the feature space rather than the image space. In contrast, the current work explores using semantic information to resolve ambiguities that are difficult for pure geometric modeling. Methods proposed in the current work are complementary to these recent methods, but we choose to validate our approach on a state-of-the-art framework known as GeoNet [3].

Multi-Task Learning for Semantic and Depth Multi-task learning [263] achieves better generalization by allowing the system to learn features that are robust across different tasks. Recent methods focus on designing efficient architectures that can predict related tasks using shared features while avoiding negative transfers [48, 264–268]. In this context, several prior works report promising results combining scene geometry with semantics. For instance, similar to our method Liu *et al.*[269] uses semantic predictions to provide depth. However, this work is fully supervised and only uses sub-optimal traditional methods. Wang *et al.*[270], Cross-Stitching [264], UberNet [267] and NDDR-CNN [268] all report improved performance over single-task baselines. But they have not addressed outdoor scenes and unsupervised geometry understanding. Our work is also related to PAD-Net [271]. PAD-Net reports improvements by combining intermediate tasks as inputs to final depth and segmentation tasks. Our method of using semantic input similarly introduces an intermediate prediction task as input to the depth and pose predictions, but we tackle the problem setting where depth labels are not provided.

7.3 State-of-the-art Unsupervised Geometry Prediction

Prior to presenting our technical approach, we provide a brief overview of state-of-the-art unsupervised depth and motion estimation framework, which is based on image reconstruction from geometric predictions [3, 235]. It trains the geometric prediction models through the recon-

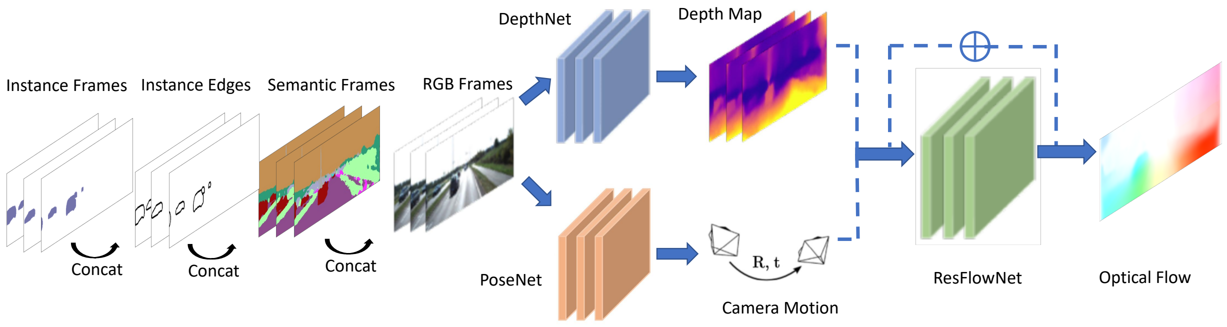


Figure 7.2: Our unsupervised architecture contains DepthNet, PoseNet and ResFlowNet to predict depth, poses and motion using semantic-level and instance-level segmentation concatenated along the input channel dimension.

structions of a target image from source images. The target and source images are neighboring frames in a video sequence. Note that such a reconstruction is possible only when certain elements of the 3D geometry of the scene are understood: (1) The relative 3D location (and thus the distance) between the camera and each pixel. (2) The camera ego-motion. (3) The motion of pixels. Thus this framework can be used to train a depth estimator and an ego-motion estimator, as well as a optical flow predictor.

Technically, each training sample $I = \{I_i\}_{i=1}^{n}$ consists of n contiguous video frames $I_i \in \mathbb{R}^{H\times W\times 3}$ where the center frame I_t is the "target frame" and the other frames serve as the "source frame". In training, a differentiable warping function $f_{t\to s}$ is constructed from the geometry predictions. The warping function is used to reconstruct the target frame $\tilde{I}_s \in \mathbb{R}^{H\times W\times 3}$ from source frame I_s via bilinear sampling. The level of success in this reconstruction provides training signals through backpropagation to the various ConvNets in the system. A standard loss function to measure reconstruction success is as follows:

$$\mathcal{L}_{rw} = \alpha\frac{1-\text{SSIM}(I_t,\tilde{I}_s)}{2} + (1-\alpha)||I_t - \tilde{I}_s||_1 \tag{7.1}$$

where SSIM denotes the structural similarity index [272] and α is set to 0.85 in [3].

To filter out erroneous predictions while preserving sharp details, the standard practice is to include an edge-aware depth smoothness loss $\mathcal{L}_{ds}$ weighted by image gradients

$$\mathcal{L}_{ds} = \sum_{p_t} |\nabla D(p_t)| \cdot (e^{-|\nabla I(p_t)|})^T \tag{7.2}$$

where $|\cdot|$ denotes element-wise absolute operation, ∇ is the vector differential operator, and T denotes transpose of gradients. These losses are usually computed from a pyramid of multi-scale predictions. The sum is used as the training target.

While the reconstruction of RGB images is an effective surrogate task for unsupervised learning, it is limited by the lack of semantic information as supervision signals. For example, the system cannot learn the difference between the car and the road if they have similar colors or two neighboring cars with similar colors. When object motion is considered in the models, the learning can mistakenly assign motion to non-moving objects as the geometric constraints are ill-posed. We augment and improve this system by leveraging semantic information.

7.4 Methods

In this section, we present solutions to enhance geometry predictions with semantic information. Semantic labels can provide rich information on 3D scene geometry. Important details such as 3D location of pixels and their movements can be inferred from a dense representation of the scene semantics. The proposed methods are applicable to a wide variety of recently proposed unsupervised geometry learning frameworks based on photometric reconstruction [3, 235, 259] represented by our baseline framework introduced in Section 7.3. Our complemented pipeline in test time is illustrated in Fig 7.2.

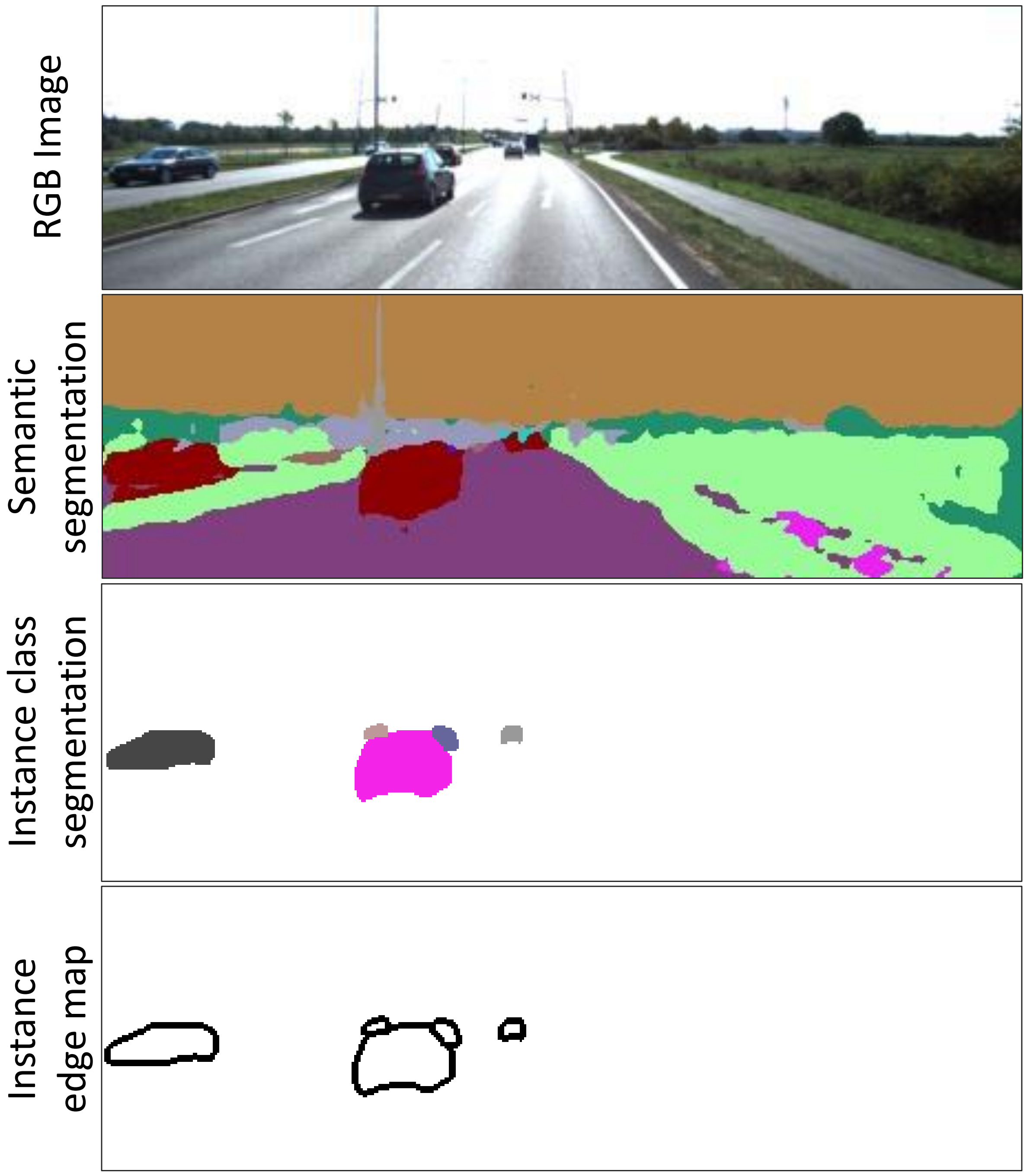


Figure 7.3: Top to bottom: RGB image, semantic segmentation, instance class segmentation and instance edge map. They are used for the full prediction architecture. The semantic segmentation provides accurate segments grouped by classes, but it fails to differentiate neighboring cars.

7.4.1 Semantic Input Augmentation

Semantic predictions can improve geometry prediction models when serving as input features. Unlike RGB images, semantic predictions mark objects and contiguous structures with consistent blobs, which provide important information for the learning problem. However, it is uncertain that using semantic labels as input could indeed improve depth and flow predictions since training labels are not available. Semantic information could be lost or distorted, which would end up being a noisy training signal. An important finding of our work is that using semantic predictions as inputs significantly improves the accuracy in geometry predictions, despite the presence of noisy training signal. Input representation and the type of semantic labels have a large impact on the performance of the system. We further illustrate this by Fig 7.3, where we show various semantic labels (semantic segmentation, instance segmentation, and instance edge) that we use to augment the input. This imposes additional constraints such as depth of the pixels belonging to a particular object (e.g. a vehicle) which helps the learning process. Furthermore, sudden changes in the depth predictions can be inferred from the boundary of vehicles. The semantic labels of the pixels can provide important information to associate pixels across frames.

Encoding Pixel-wise Class Labels We explored two input encoding techniques for class labels: dense encoding and one-hot encoding. In dense encoding, dense class labels are concatenated along the input channel dimension. The added semantic features are centralized to the range of $[-1, 1]$ to be consistent with RGB inputs. In the case of one-hot encoding, the class-level semantic predictions are first expanded to one-hot encoding and then concatenated along the input channel dimension. The labels are represented as one-hot sparse vectors. In this variant, semantic features are not normalized since they have similar value range as the RGB inputs,

Encoding Instance-level Semantic Information Both dense and one-hot encoding are natural to class-level semantic prediction, where each pixel is only assigned a class label rather than an instance label. Our conjecture is that instance-level semantic information is particular well-suited to improve unsupervised geometric predictions, as it provides accurate information on the boundary between individual objects of the same type. Unlike class-level label, the instance label itself does not have a well-defined meaning. Across different frames, the same label could refer to different object instances. To efficiently represent the instance-level information, we compute the gradient map of a dense instance map and use it as an additional feature channel concatenating to the class label input (dense/one-hot encoding).

Direct Input versus Residual Correction Complementary to the choice of encoding, we also experiment with different architectures to feed semantic information to the geometry prediction model. In particular, we make a residual prediction using a separate branch that takes in only semantic inputs. Notably, using residual depth prediction leads to further improvement on top of the gains from the direct input methods.

7.4.2 Semantic Guided Loss Functions

The information from semantic predictions could be diminished due to noisy semantic labels and very deep architectures. Hence, we design training loss functions that are guided by semantic information. In such design, the semantic predictions provide additional loss constraints to the network. In this subsection, we introduce a set of semantic guided loss functions to improve depth and flow predictions.

Semantic Warp Loss Semantic predictions can help learn scenarios where reconstruction of the RGB image is correct in terms of pixel values but violates obvious semantic correspondences, e.g. matching pixels to incorrect semantic classes and/or instances. In light of this, we propose to reconstruct the semantic predictions in addition of doing so for RGB images. We call this “semantic warping loss” as it is based on warping of the semantic predictions from source frames to the target frame. Let S_s be the source frame semantic prediction and $\tilde{S}_s^{rig}$ be the warped semantic image, we define semantic warp loss as:

$$\mathcal{L}_{sem} = ||\tilde{S}_s^{rig} - S_t||_2 \tag{7.3}$$

The warped loss is added to the baseline framework using a hyper-tuned value of the weight w.

Masking of Reconstruction Loss via Semantics As described in Section 7.3, the ambiguity in object motion can lead to sub-optimal learning. Semantic labels can partially resolve this by separating each class of region. Motivated by this observation, we mask the foreground region out to form a set of new images $J^k_{t,c} = I_{t,c} \odot S_{t,k}$ for $c = 0,1,2$ and $k = 0,\dots,K-1$ where c represents the RGB-channel index, $\odot$ is the element-wise multiplication operator and $S_{s,k}$ is the k-th channel of the binary semantic segmentation (K classes in total). Similarly we can obtain $\tilde{J}^{rig,k}_{s,c} = \tilde{I}^{rig}_{s,c} \odot S_{t,k}$ for $c = 0,1,2$ and $k = 0,\dots,K-1$. Finally, the image similarity loss is defined as:

$$\mathcal{L}'_{rw} = \sum_{k=0}^{K-1} \alpha \frac{1 - \mathrm{SSIM}(J^k_t, \tilde{J}^{rig,k}_s)}{2} + (1-\alpha)||J^k_t - \tilde{J}^{rig,k}_s||_1 \tag{7.4}$$

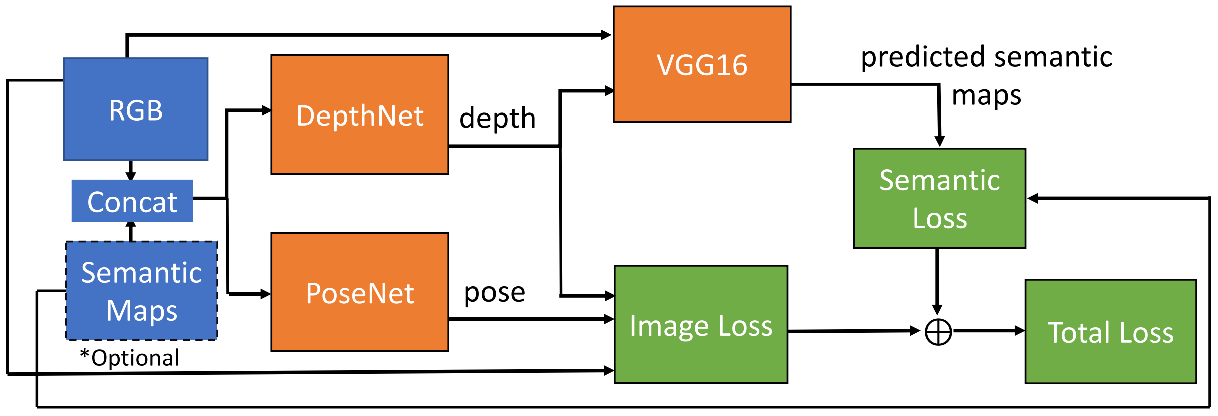


Figure 7.4: Infer semantic labels from depth predictions. The transfer function uses RGB and predicted depth as input. We experimented the variants with and without semantic input.

Semantic-Aware Edge Smoothness Loss Equation 7.2 uses RGB to infer edge locations when enforcing smooth regions of depth. This could be improved by including an edge map computed from semantic predictions. Given a semantic segmentation result S_t, we define a weight matrix $M_t \in [0,1]^{H\times W}$ where the weight is low (close to zero) on class boundary regions and high (close to one) on other regions. We propose a new image similarity loss as:

$$
\begin{aligned}
\mathcal{L}''_{rw} = \sum_{k=0}^{K-1} \alpha \frac{1 - \mathrm{SSIM}(I_t \odot M_t, \tilde{I}_s^{rig} \odot M_t)}{2} \\
+ (1-\alpha)||I_t \odot M_t - \tilde{I}_s^{rig} \odot M_t||_1
\end{aligned}
\tag{7.5}
$$

Semantic Loss by Transfer Network Motivated by the observation that high-quality depth maps usually depict object classes and background region, we designed *a novel transfer network architecture*. As shown in Fig 7.4 the transfer network block receives predicted depth maps along with the original RGB images and outputs semantic labels. The transfer network introduces a semantic reconstruction loss term to the objective function to force the predicted depth maps to be richer in contextual sense, hence refines the depth estimation. For implementation, we choose the ResNet-50 as the backbone and alter the dimensions for the input and output convolutional layers to be consistent with the segmentation task. The network generates one-hot encoded heatmaps and use cross-entropy as the semantic similarity measure.

7.5 Experiments

To quantify the benefits that semantic information brings to geometry-based learning, we designed experiments similar to [3]. First, we showed our model's depth prediction performance on KITTI dataset [238], which outperformed state-of-the-art unsupervised and supervised models. Then we designed ablation studies to analyze each individual component's contribution. Finally, we presented improvements in flow predictions and revisited the performance gains using a category-specific evaluation.

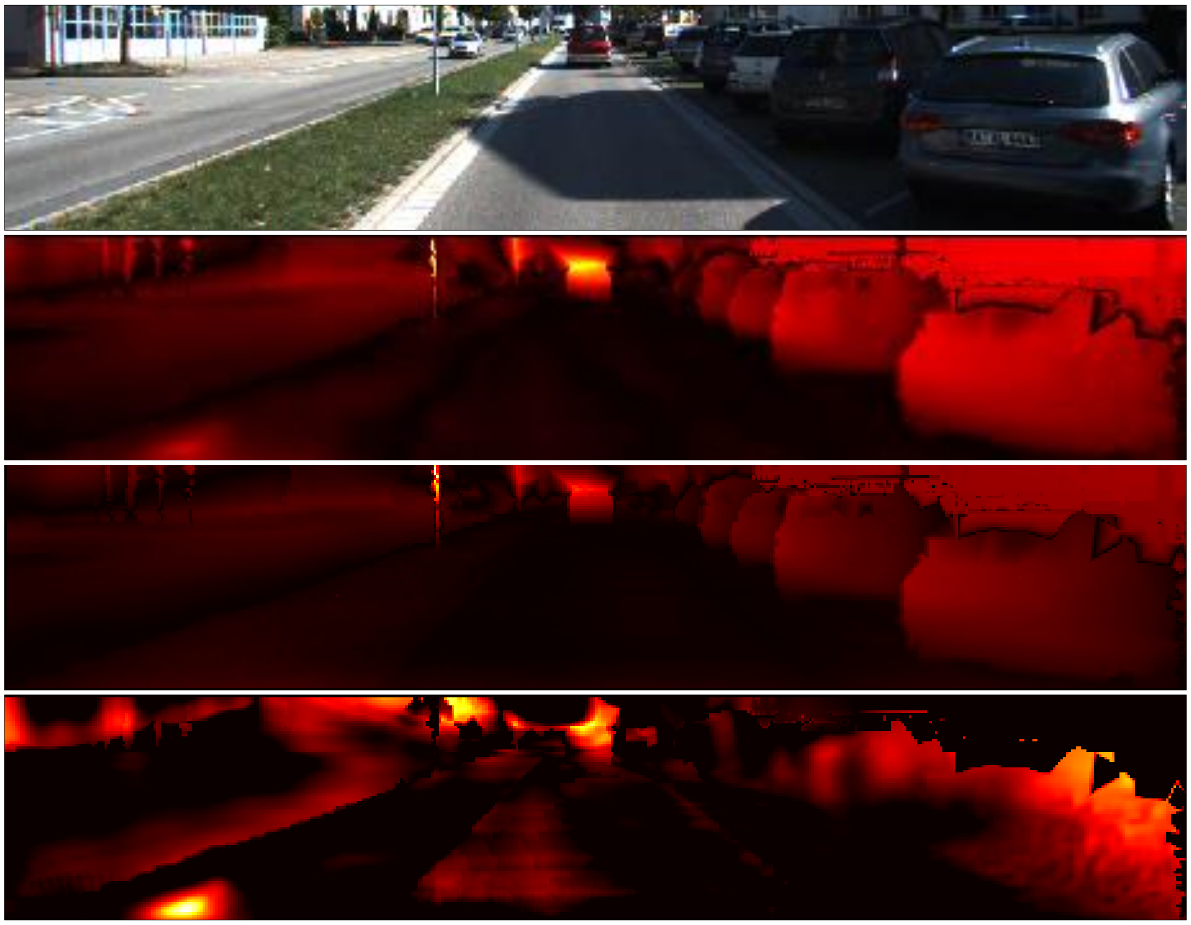
7.5.1 Implementation Details

To make a fair comparison with state-of-the-art models [2, 3, 235], we divided KITTI 2015 dataset into train set (40238 images) and test set (697 images) according to the rules from Eigen *et al* [2]. We used DeepLabv3+ [239] (pretrained on [178]) for semantic segmentation and Mask-RCNN [240] (pretrained on [64]) for instance segmentation. Similar to the hyper-parameter settings in [3], we used Adam optimizer [273] with initial learning rate as 2e-4, set batch size to 4 per GPU and trained our modified DepthNet and PoseNet modules for 250000 iterations with random shuffling and data augmentation (random scaling, cropping and RGB perturbation). The training took 10 hours on two GTX1080Ti.

7.5.2 Monocular Depth Evaluation on KITTI

We augmented the image sequences with corresponding semantic and instance segmentation sequences and adopted the scale normalization suggested in [274]. In the evaluation stage, the ground truth depth maps were generated by projecting 3D Velodyne LiDAR points to the image plane. Followed by [3], we clipped our depth predictions within 0.001m to 80m and calibrated the scale by the medium number of the ground truth. The evaluation results are shown in Table 7.1, where all the metrics are introduced in [2]. Our model benefits significantly from feature augmentation and surpasses the state-of-the-art methods substantially in both supervised and unsupervised fields.

Moreover, we found a correlation between the improvement region and object classes. We visualized the absolute relative error (AbsRel) among image plane from our model and from the baseline. As shown in Fig 7.5, most of the improvements come from regions containing objects. This indicates that the network is able to learn the concept of objects to improve the depth prediction by rendering extra semantic information.



Top to bottom: Input RGB image, AbsRel error map of [3], AbsRel error map of ours, and improvements of ours on AbsRel map compared to [3]. The ground truth is interpolated to enhance visualization. Lighter color in those heatmaps corresponds to larger errors or improvements.

Figure 7.5: Comparisons of depth evaluations on KITTI.

Table 7.1: Monocular depth results on KITTI 2015 [1] by the split of Eigen *et al.* [2] (Our model used scale normalization.)

Method	Supervised	Error-related metrics				Accuracy-related metrics		
		Abs Rel	Sq Rel	RSME	RSME log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Eigen *et al.* [2] Coarse	Depth	0.214	1.605	6.653	0.292	0.673	0.884	0.957
Eigen *et al.* [2] Fine	Depth	0.203	1.548	6.307	0.282	0.702	0.890	0.957
Liu *et al.* [241]	Depth	0.202	1.614	6.523	0.275	0.678	0.895	0.965
Godard *et al.* [259]	Pose	0.148	1.344	5.927	0.247	0.803	0.922	0.964
Zhou *et al.* [235] updated	No	0.183	1.595	6.709	0.270	0.734	0.902	0.959
Yin *et al.* [3]	No	0.155	1.296	5.857	0.233	0.793	0.931	0.973
Ours	No	**0.133**	**0.905**	**5.181**	**0.208**	**0.825**	**0.947**	**0.981**
(improved by)		**14.04%**	**30.19%**	**11.55%**	**10.85%**	**3.14%**	**1.53%**	**0.80%**

7.5.3 Ablation Studies

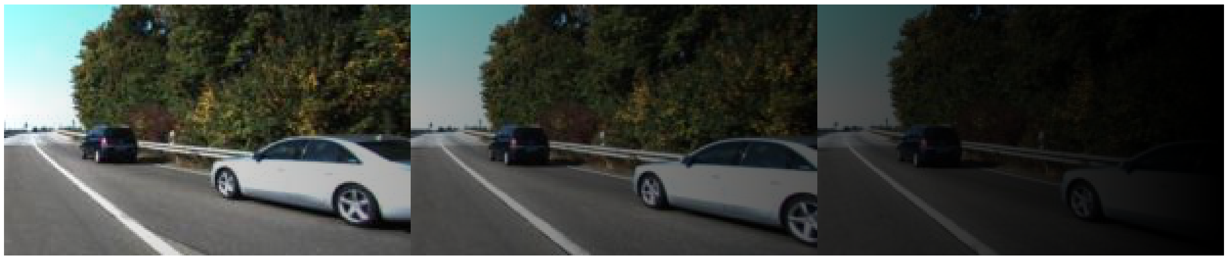
Here we took a deeper look of our model, testified its robustness under noise from observations, and presented variations of our framework to show promising explorations for future researchers. In the following experiments, we kept all the other parameters the same in [3] and applied the same training/evaluation strategies mentioned in Section 7.5.2

How much gain from various feature augmentation? We tried out different combinations and forms of semantic/instance-level inputs based on "Yin *et al*" [3] with scale normalization. From Table 7.2, our first conclusion is that any meaningful form of extra input can ameliorate the model, which is straightforward. Secondly, when we use "Semantic" and "Instance class" for feature augmentation, one-hot encoding tends to outperform the dense map form. Conceivably one-hot encoding stores richer information in its structural formation, whereas dense map only contains discrete labels which may be more difficult for learning. Moreover, using both "Semantic" and "Instance class" can provide further gain, possibly due to the different label distributions of the two datasets. Labels from Cityscape [178] cover both background and foreground concepts, while the COCO dataset [64] focuses more on objects. At last, when we combined one-hot encoded "Semantic" and "Instance class" along with "Instance id" edge features, the network exploited the most from scene understanding, hence greatly enhanced the performance.

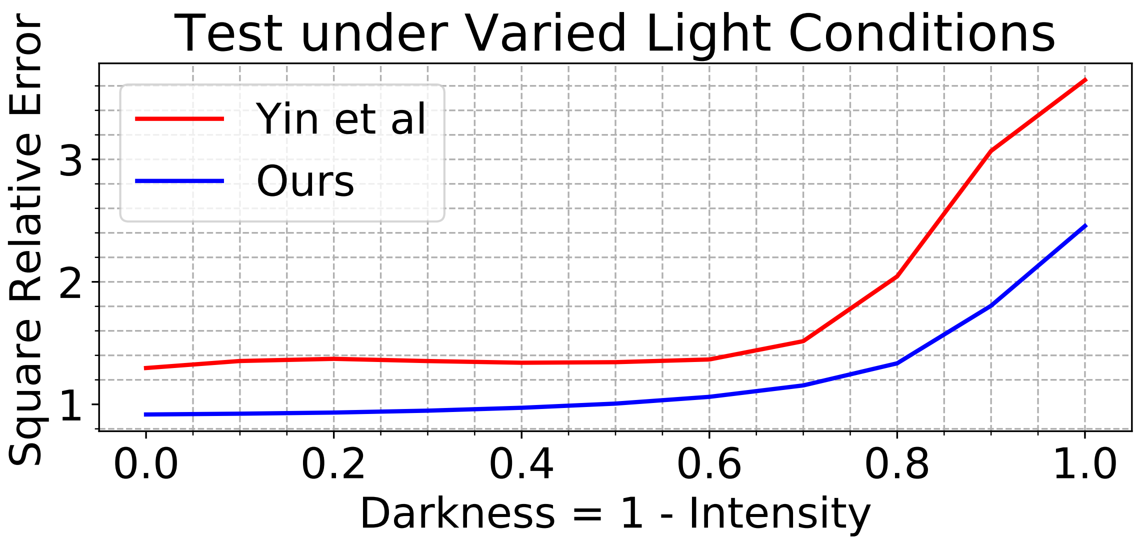
Table 7.2: Depth prediction performance gains due to different semantic sources and forms. (Scale normalization was used.)

Semantic	Instance class	Instance id	Error-related metrics				Accuracy-related metrics		
			Abs Rel	Sq Rel	RSME	RSME log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
			0.149	1.060	5.567	0.226	0.796	0.935	0.975
Dense			0.142	0.991	5.309	0.216	0.814	0.943	0.980
One-hot			0.139	0.949	5.227	0.214	0.818	0.945	0.980
	Dense		0.142	0.986	5.325	0.218	0.812	0.943	0.978
	One-hot		0.141	0.976	5.272	0.215	0.811	0.942	0.979
		Edge	0.145	1.037	5.314	0.217	0.807	0.943	0.978
	Dense	Edge	0.142	0.969	5.447	0.219	0.808	0.941	0.978
One-hot	One-hot	Edge	**0.133**	**0.905**	**5.181**	**0.208**	**0.825**	**0.947**	**0.981**

Can our model survive under low lighting conditions? To testify our model's robustness for varied lighting conditions, we multiplied a scalar between 0 and 1 to RGB inputs in the evaluation. Fig 7.6 showed that our model still holds equal performance to [3] when the intensity drops to 30%.



(a) Observations under decreased light condition (left to right)



(b) Robustness under decreased light condition

Figure 7.6: The abs errs change as lighting condition drops. Our model can still be better than baseline even if the lighting intensity drops to 0.30 of the original ones.

Which module needs extra information the most? We fed semantics to only DepthNet or PoseNet to see the difference in their performance gain. From Table 7.3 we can see that compared to DepthNet, PoseNet learns little from the semantics to help depth prediction. Therefore we tried to feed the semantics to a new PoseNet with the same structure as the original one and compute the predicted poses by taking the sum from two different PoseNets, which led to performance gain; however, performance gain was not observed from applying the same method to DepthNet.

Table 7.3: Each module's contribution toward performance gain from semantics. (Scale normalization was used.)

DepthNet	PoseNet	Error-related metrics				Accuracy-related metrics		
		Abs Rel	Sq Rel	RSME	RSME log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
		0.149	1.060	5.567	0.226	0.796	0.935	0.975
Channel		0.145	0.957	5.291	0.216	0.805	0.943	0.980
	Channel	0.147	1.076	5.385	0.223	0.808	0.938	0.975
Channel	Channel	0.139	0.949	**5.227**	0.214	0.818	0.945	0.980
Extra Net	Channel	0.147	1.036	5.593	0.226	0.803	0.937	0.975
Channel	Extra Net	**0.135**	**0.932**	5.241	**0.211**	**0.821**	**0.945**	**0.980**

How to be "semantic-free" in evaluation? Though semantic helps depth prediction, this idea relies on semantic features during the evaluation phase. If semantic is only utilized in the loss, it would not be needed in evaluation. We attempted to introduce a handcrafted semantic loss term as a weight guidance among image plane but it didn't work well. Also we designed a transfer network which uses the predicted depth to predict semantic maps along with a reconstruction error to help in the training stage. The result in Table 7.4 shows a better result can be obtained by training from pretrained models.

Table 7.4: Gains in depth prediction using our proposed Transfer Network. (**+sn**: "using scale normalization".)

Checkpoint	Transfer Network	Error-related metrics				Accuracy-related metrics		
		Abs Rel	Sq Rel	RSME	RSME log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Yin *et al.* [3]		0.155	1.296	5.857	0.233	0.793	0.931	0.973
Yin *et al.* [3]	Yes	0.150	1.141	5.709	0.231	0.792	0.934	0.974
Yin *et al.* [3] +sn		0.149	1.060	5.567	0.226	0.796	0.935	0.975
Yin *et al.* [3] +sn	Yes	**0.145**	**0.994**	**5.422**	**0.222**	**0.806**	**0.939**	**0.976**

7.5.4 Optical Flow Estimation on KITTI

Using our best model for DepthNet and PoseNet in Section 7.5.2, we conducted rigid flow and full flow evaluation on KITTI [238]. We generated the rigid flow from estimated depth and pose, and compared with [3]. Our model performed better in all the metrics shown in Table 7.5.

Table 7.5: Rigid flow prediction from first stage on KITTI on non-occluded regions(Noc) and overall regions(All).

Method	End Point Error		Accuracy	
	Noc	All	Noc	All
Yin *et al.* [3]	23.5683	29.2295	0.2345	0.2237
Ours	22.3819	26.8465	0.2519	0.2376

Table 7.6: Full flow prediction on KITTI 2015 on non-occluded regions(Noc) and overall regions(All). Results from DirFlowNetS are shown in [3]

Method	End Point Error	
	Noc	All
DirFlowNetS	6.77	12.21
Yin *et al.* [3]	8.05	10.81
Ours	7.66	13.91

We further appended the semantic warping loss introduced in Section 7.4.2 to ResFlowNet in [3] and trained our model on KITTI stereo for 1600000 iterations. As demonstrated in Table 7.6, flow prediction got improved in non-occluded region compared to [3] and our model produced comparable results in overall regions.

7.5.5 Category-Specific Metrics Evaluation

This section will present the improvements by semantic categories. As shown in the bar-chart in Fig 7.7, most improvements were shown in “Vehicle” and “Dynamic” classes[3], where errors are generally large. Our network did not improve much for other less frequent categories, such as “Motorcycle”, which are generally more difficult to segment in images.

[3]For “Dynamic” classes, we choose “person”, “rider”, “car”, “truck”, “bus”, “train”, “motorcycle” and “bicycle” classes as defined in [178]

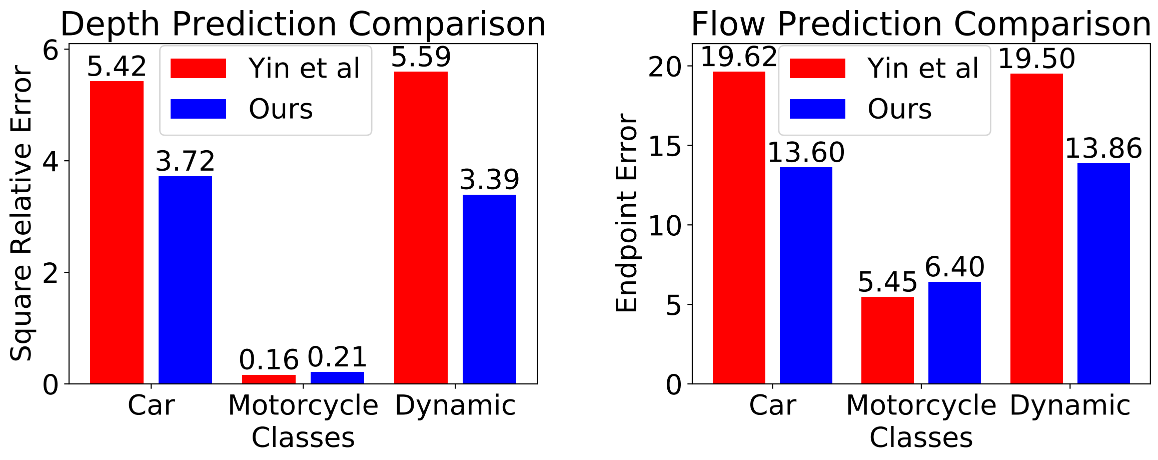


Figure 7.7: Performance gains in depth (left) and flow (right) among different classes of dynamic objects.

7.6 Conclusion

In SIGNet, we strive to achieve robust performance for depth and flow perception without using geometric labels. To achieve this goal, SIGNet utilizes semantic and instance segmentation to create spatial constraints on the geometric attributes of the pixels. We present novel methods of feature augmentation and loss augmentation to include semantic labels in the geometry predictions. This work presents a first of a kind approach which moves away from pixel-level to object-level depth and flow predictions. Most notably, our method significantly surpasses the state-of-the-art solution for monocular depth estimation. In the future, we would like to extend our SIGNet to various sensor modalities (IMU, LiDAR or thermal).

Acknowledgment

Chapter 7 is substantially based on material that has been accepted for publication by *32nd IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2019)*, authored by Yue Meng, Yongxi Lu, Aman Raj, Samuel Sunarjo, Rui Guo, Tara Javidi, Gaurav Bansal and Dinesh Bharadia. The dissertation author was a joint primary investigator and author of this

material.

Chapter 8

Conclusion and Future Directions

This thesis presents a set of novel algorithms that address practical limitations in existing object and scene understanding systems. These limitations include high computational demands of the systems, the lack of unified models that can model multiple tasks accurately in an efficient manner, and the high dependency on output labels which are in many cases difficult and expensive to collect. Our strategy is as follows: We first identify important and intuitive structures in the respective problems. Then, we analyze the existing architectures and introduce additional dimensions of variations in the computational and model structures. The proposed algorithms are more "adaptive" than their respective baselines, in the sense that they can now use the new degrees of freedoms to address the limitations in the latter. These algorithms are practical because they demonstrate significant empirical successes in addressing the limitations of existing methods. They have pushed the state-of-the-art and inspired follow-up studies in designing better object and scene understanding systems for real-world challenges.

There are many future directions. In addition to those discussed in respective chapters, one particularly interesting question is how to quantify the progress towards practical object and scene understanding systems, given the many practical considerations that are all good candidates for an evaluation of success. A successful design must have built-in mechanism to perform best under a given resource budget (computational, model-size, memory footprint), and should have not just good accuracy for a single task but also for a set of related tasks. The design must also take into account the labeling cost, especially the fine-grained breakdown which are usually embedded in the data collection process. Currently, systems are evaluated in simple metrics that sometimes could be detached from the realistic considerations. A thorough benchmark could complement existing efforts and expedite progress in this field.

Bibliography

[1] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3061–3070, 2015.

[2] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[3] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2018.

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan 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, April 2018.

[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, June 2015.

[7] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6230–6239, July 2017.

[8] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122, 2016.

[9] Fisher Yu, Vladlen Koltun, and Thomas A. Funkhouser. Dilated residual networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 636–644, 2017.

[10] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017.

[11] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *Computer Vision–ECCV 2014*, pages 391–405. Springer, 2014.

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *In NIPS*, 2015.

[13] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016.

[14] Yongxi Lu, Tara Javidi, and Svetlana Lazebnik. Adaptive object detection using adjacency and zoom prediction. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2351–2359, June 2016.

[15] Tsung-Yi Lin, Piotr Dollr, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, July 2017.

[16] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollr. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, Oct 2017.

[17] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[18] Kaiming He, Georgia Gkioxari, Piotr Dollr, and Ross Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, Oct 2017.

[19] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660, June 2014.

[20] Tinghui Zhou, Matthew Brown, Noah Snavely, and David Lowe. Unsupervised learning of depth and ego-motion from video. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6612–6619, July 2017.

[21] Yuliang Zou, Zelun Luo, and Jia-Bin Huang. Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[22] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1983–1992, 2018.

[23] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6602–6611, 2017.

[24] Ravi Garg, G VijayKumarB., and Ian D. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *ECCV*, 2016.

[25] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018.

[26] Liang-Chieh Chen, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. *In CVPR*, 2018.

[27] Shu Liu, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. Sgn: Sequential grouping networks for instance segmentation. *In ICCV*, 2017.

[28] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. *In CVPR*, 2018.

[29] Alexander Kirillov, Kaiming He, Ross B. Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. *CoRR*, abs/1801.00868, 2018.

[30] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. *CoRR*, abs/1901.03784, 2019.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[33] Chi Su, Shiliang Zhang, Junliang Xing, Wen Gao, and Qi Tian. Deep attributes driven multi-camera person re-identification. *ECCV*, 2016.

[34] Daniel A Vaquero, Rogerio S Feris, Duan Tran, Lisa Brown, Arun Hampapur, and Matthew Turk. Attribute-based people search in surveillance environments. In *WACV*, 2009.

[35] Rogerio Feris, Russel Bobbitt, Lisa Brown, and Sharath Pankanti. Attribute-based people search: Lessons learnt from a practical surveillance system. In *ICMR*, 2014.

[36] Junshi Huang, Rogerio S Feris, Qiang Chen, and Shuicheng Yan. Cross-domain image retrieval with a dual attribute-aware ranking network. In *ICCV*, pages 1062–1070, 2015.

[37] Luoqi Liu, Junliang Xing, Si Liu, Hui Xu, Xi Zhou, and Shuicheng Yan. Wow! you are so beautiful today! *ACM Transactions on Multimedia Computing, Communications, and Applications*, 11(1s):20, 2014.

[38] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. From facial parts responses to face detection: A deep learning approach. In *ICCV*, 2015.

[39] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Learning deep representation for face alignment with auxiliary attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(5), 2016.

[40] Jing Wang, Yu Cheng, and Rogerio Schmidt Feris. Walk and learn: Facial attribute representation learning from egocentric video and contextual data. *CVPR*, 2016.

[41] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[42] Qiang Chen, Junshi Huang, Rogerio Feris, Lisa M Brown, Jian Dong, and Shuicheng Yan. Deep domain adaptation for describing people based on fine-grained clothing attributes. In *CVPR*, 2015.

[43] Ning Zhang, Manohar Paluri, Marc'Aurelio Ranzato, Trevor Darrell, and Lubomir Bourdev. Panda: Pose aligned networks for deep attribute modeling. In *CVPR*, 2014.

[44] Ethan Rudd, Manuel Günther, and Terrance Boult. Moon: A mixed objective optimization network for the recognition of facial attributes. *ECCV*, 2016.

[45] Patrick Sudowe, Hannah Spitzer, and Bastian Leibe. Person attribute recognition with a jointly-trained holistic cnn model. In *ICCV ChaLearn Looking at People Workshop*, 2015.

[46] Yongxi Lu, Tara Javidi, and Svetlana Lazebnik. Adaptive object detection using adjacency and zoom prediction. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2351–2359, 2016.

[47] Yongxi Lu, Zeyangyi Wang, Ziyao Tang, and Tara Javidi. Target localization with drones using mobile cnns. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2566–2573, Oct 2018.

[48] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogério Schmidt Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1131–1140, 2017.

[49] Yongxi Lu, Ziyao Tang, and Tara Javidi. Implicit label augmentation on partially annotated clips via temporally-adaptive features learning. *CoRR*, abs/1905.10000, 2019.

[50] Ziyao Tang, Yongxi Lu, and Tara Javidi. Efficient video understanding via layered multi frame-rate analysis. *CoRR*, abs/1811.09834, 2018.

[51] Yue Meng, Yongxi Lu, Aman Raj, Samuel Sunarjo, Rui Guo, Tara Javidi, Gaurav Bansal, and Dinesh Bharadia. Signet: Semantic instance aided unsupervised 3d geometry perception. *In CVPR*, 2019.

[52] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2155–2162. IEEE, 2014.

[53] Spyros Gidaris and Nikos Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[54] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jagannath Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Computer Vision–ECCV 2014*, pages 346–361. Springer, 2014.

[56] Xiaodan Liang, Si Liu, Yunchao Wei, Luoqi Liu, Liang Lin, and Shuicheng Yan. Towards computational baby learning: A weakly-supervised approach for object detection. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[57] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[58] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Advances in Neural Information Processing Systems*, pages 2553–2561, 2013.

[59] Ross Girshick. Fast r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[60] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[61] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[62] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2015.

[63] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[64] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014*, pages 740–755. Springer, 2014.

[65] Christoph H Lampert, Matthew B Blaschko, and Thomas Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(12):2129–2142, 2009.

[66] Abel Gonzalez-Garcia, Alexander Vezhnevets, and Vittorio Ferrari. An active search strategy for efficient object class detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3022–3031, 2015.

[67] Juan C. Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[68] Donggeun Yoo, Sunggyun Park, Joon-Young Lee, Anthony S. Paek, and In So Kweon. Attentionnet: Aggregating weak directions for accurate object detection. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[69] Antonio Torralba, Aude Oliva, Monica S Castelhano, and John M Henderson. Contextual guidance of eye movements and attention in real-world scenes: the role of global features in object search. *Psychological review*, 113(4):766, 2006.

[70] Ali Borji and Laurent Itti. State-of-the-art in visual attention modeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):185–207, 2013.

[71] Santosh K Divvala, Derek Hoiem, James H Hays, Alexei Efros, Martial Hebert, et al. An empirical study of context in object detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1271–1278. IEEE, 2009.

[72] Yongxi Lu and Tara Javidi. Efficient object detection for high resolution images. *arXiv preprint arXiv:1510.01257*, 2015.

[73] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[74] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[75] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, and Scott Reed. Ssd: Single shot multibox detector. *arXiv preprint arXiv:1512.02325*, 2015.

[76] Sean Bell, C Lawrence Zitnick, Kavita Bala, and Ross Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. *arXiv preprint arXiv:1512.04143*, 2015.

[77] T. Furukawa, F. Bourgault, B. Lavis, and H. F. Durrant-Whyte. Recursive bayesian search-and-tracking using coordinated uavs for lost targets. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2521–2526, May 2006.

[78] S. Waharte and N. Trigoni. Supporting search and rescue operations with uavs. In *2010 International Conference on Emerging Security Technologies*, pages 142–147, Sept 2010.

[79] M. A. Goodrich, L. Lin, and B. S. Morse. Using camera-equipped mini-uavs to support collaborative wilderness search and rescue teams. In *2012 International Conference on Collaboration Technologies and Systems (CTS)*, pages 638–638, May 2012.

[80] D. Maturana, S. Arora, and S. Scherer. Looking forward: A semantic mapping system for scouting with micro-aerial vehicles. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6691–6698, Sept 2017.

[81] K. C. T. Vivaldini, V. Guizilini, M. D. C. Oliveira, T. H. Martinelli, D. F. Wolf, and F. Ramos. Route planning for active classification with uavs. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2563–2568, May 2016.

[82] M. Popovi, G. Hitz, J. Nieto, I. Sa, R. Siegwart, and E. Galceran. Online informative path planning for active classification using uavs. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5753–5758, May 2017.

[83] M. Coombes, W. H. Chen, and C. Liu. Boustrophedon coverage path planning for uav aerial surveys in wind. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1563–1571, June 2017.

[84] D. Meyer, E. Fraijo, E. Lo, D. Rissolo, and F. Kuester. Optimizing uav systems for rapid survey and reconstruction of large scale cultural heritage sites. In *2015 Digital Heritage*, volume 1, pages 151–154, Sept 2015.

[85] P. Roy and V. Isler. Active view planning for counting apples in orchards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6027–6032, Sept 2017.

[86] S. Waharte, A. Symington, and N. Trigoni. Probabilistic search with agile uavs. In *2010 IEEE International Conference on Robotics and Automation*, pages 2840–2845, May 2010.

[87] F. Bourgault, T. Furukawa, and H. F. Durrant-Whyte. Coordinated decentralized search for a lost target in a bayesian world. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 1, pages 48–53 vol.1, Oct 2003.

[88] T. H. Chung and J. W. Burdick. A decision-making framework for control strategies in probabilistic search. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4386–4393, April 2007.

[89] A. Symington, S. Waharte, S. Julier, and N. Trigoni. Probabilistic target detection by camera-equipped uavs. In *2010 IEEE International Conference on Robotics and Automation*, pages 4076–4081, May 2010.

[90] V. Sudevan, A. Shukla, and H. Karki. Vision based autonomous landing of an unmanned aerial vehicle on a stationary target. In *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, pages 362–367, Oct 2017.

[91] A. Gupta, D. Bessonov, and P. Li. A decision-theoretic approach to detection-based target search with a uav. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5304–5309, Sept 2017.

[92] J. Lee, J. Wang, D. Crandall, S. abanovi, and G. Fox. Real-time, cloud-based object detection for unmanned aerial vehicles. In *2017 First IEEE International Conference on Robotic Computing (IRC)*, pages 36–43, April 2017.

[93] K. R. Sapkota, S. Roelofsen, A. Rozantsev, V. Lepetit, D. Gillet, P. Fua, and A. Martinoli. Vision-based unmanned aerial vehicle detection and tracking for sense and avoid systems. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1556–1561, Oct 2016.

[94] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[95] F. Vanegas, D. Campbell, M. Eich, and F. Gonzalez. Uav based target finding and tracking in gps-denied and cluttered environments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2307–2313, Oct 2016.

[96] S. C. W. Ong, S. W. Png, D. Hsu, and W. S. Lee. POMDPs for robotic tasks with mixed observability. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[97] Wee Sun Lee Hanna Kurniawati, David Hsu. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland, June 2008.

[98] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[99] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$1mb model size. *CoRR*, abs/1602.07360, 2016.

[100] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *CoRR*, abs/1707.01083, 2017.

[101] R. Ranjan, V. Patel, and R. Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. In *arXiv preprint arXiv:1603.01249*, 2016.

[102] B. Jou and S. F. Chang. Deep cross residual learning for multi-task visual recognition. In *ACM Multimedia*, 2016.

[103] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *CVPR*, 2016.

[104] Joel A Tropp, Anna C Gilbert, and Martin J Strauss. Algorithms for simultaneous sparse approximation. part i: Greedy pursuit. *Signal Processing*, 86(3):572–588, 2006.

[105] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *CVPR*, 2016.

[106] R. Caruana. Multi-task learning. *Machine Learning Journal*, 28:41–75, 1997.

[107] S. Thrun and L. Pratt. Learning to learn. *Kluwer Academic Publishers*, 1998.

[108] Laurent Jacob, Jean-philippe Vert, and Francis R Bach. Clustered multi-task learning: A convex formulation. In *NIPS*, 2009.

[109] A. Kumar and H. Daume III. Learning task grouping and overlap in multi-task. In *ICML*, 2012.

[110] Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 8(Jan):35–63, 2007.

[111] Jiayu Zhou, Jianhui Chen, and Jieping Ye. Clustered multi-task learning via alternating structure optimization. In *NIPS*, 2011.

[112] Z. Kang, K. Grauman, and F. Sha. Learning with whom to share in multi-task feature learning. In *ICML*, 2011.

[113] Alexandre Passos, Piyush Rai, Jacques Wainer, and Hal Daume III. Flexible modeling of latent task structures in multitask learning. *arXiv preprint arXiv:1206.6486*, 2012.

[114] I. Kokkinos. Ubernet: Training a universal cnn for low-, mid-, and high- level vision using diverse datasets and limited memory. In *arXiv preprint arXiv:1609.02132*, 2016.

[115] H. Bilen and A. Vedaldi. Integrated perception with recurrent multi-task neural networks. In *NIPS*, 2016.

[116] Zhicheng Yan, Hao Zhang, Robinson Piramuthu, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Yizhou Yu. Hd-cnn: Hierarchical deep convolutional neural network for large scale visual recognition. In *ICCV*, 2015.

[117] Karim Ahmed, Mohammad Haris Baig, and Lorenzo Torresani. Network of experts for large-scale image categorization. *ECCV*, 2016.

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

[119] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[120] Yani Ioannou, Duncan Robertson, Jamie Shotton, Roberto Cipolla, and Antonio Criminisi. Training cnns with low-rank filters for efficient image classification. *ICLR*, 2016.

[121] Cheng Tai, Tong Xiao, Xiaogang Wang, et al. Convolutional neural networks with low-rank regularization. *ICLR*, 2016.

[122] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *ICASSP*, 2013.

[123] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *ICLR*, 2016.

[124] Adam Polyak and Lior Wolf. Channel-level acceleration of deep face representations. *IEEE Access*, 3:2163–2175, 2015.

[125] Y. Cheng, F. Yu, R. Feris, S. Kumar, A. Choudhary, and S. F. Chang. An exploration of parameter redundancy in deep networks with circulant projections. In *ICCV*, 2015.

[126] Vikas Sindhwani, Tara Sainath, and Sanjiv Kumar. Structured transforms for small-footprint deep learning. In *NIPS*, 2015.

[127] Bingchen Gong, Brendan Jou, Felix Yu, and Shih-Fu Chang. Tamp: A library for compact deep neural networks with structured matrices. In *ACM Multimedia*, 2016.

[128] Amjad Almahairi, Nicolas Ballas, Tim Cooijmans, Yin Zheng, Hugo Larochelle, and Aaron Courville. Dynamic capacity networks. *ICML*, 2016.

[129] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[130] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[131] Joel A Tropp. Algorithms for simultaneous sparse approximation. part ii: Convex relaxation. *Signal Processing*, 86(3):589–602, 2006.

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

[133] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision (IJCV)*, 2016.

[134] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 2014.

[135] Varun Jampani, Raghudeep Gadde, and Peter V. Gehler. Video propagation networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3154–3164, 2017.

[136] Mennatullah Siam, Sepehr Valipour, Martin Jagersand, and Nilanjan Ray. Convolutional gated recurrent networks for video segmentation. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3090–3094, Sep. 2017.

[137] Mohsen Fayyaz, Mohammad Hajizadeh Saffar, Mohammad Sabokrou, Mahmood Fathy, Fay Huang, and Reinhard Klette. Stfcn: Spatio-temporal fully convolutional neural network for semantic segmentation of street scenes. In Chu-Song Chen, Jiwen Lu, and Kai-Kuang Ma, editors, *Computer Vision – ACCV 2016 Workshops*, pages 493–509, Cham, 2017. Springer International Publishing.

[138] David Nilsson and Cristian Sminchisescu. Semantic video segmentation by gated recurrent flow propagation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6819–6828, 2018.

[139] Raghudeep Gadde, Varun Jampani, and Peter V. Gehler. Semantic video cnns through representation warping. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4463–4472, 2017.

[140] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[141] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 843–852. JMLR.org, 2015.

[142] Xiaojie Jin, Xin Li, Huaxin Xiao, Xiaohui Shen, Zhe Lin, Jimei Yang, Yunpeng Chen, Jian Dong, Luoqi Liu, Zequn Jie, Jiashi Feng, and Shuicheng Yan. Video scene parsing with

predictive feature learning. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5581–5589, Oct 2017.

[143] Michaël Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *CoRR*, abs/1511.05440, 2016.

[144] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 835–851, Cham, 2016. Springer International Publishing.

[145] Ross Goroshin, Joan Bruna, Jonathan Tompson, David Eigen, and Yann LeCun. Unsupervised learning of spatiotemporally coherent metrics. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4086–4093, Dec 2015.

[146] Laurenz Wiskott and Terrence J. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Comput.*, 14(4):715–770, April 2002.

[147] Siva Karthik Mustikovela, Michael Ying Yang, and Carsten Rother. Can ground truth label propagation from video help semantic segmentation? In *ECCV Workshops*, 2016.

[148] Ignas Budvytis, Patrick Sauer, Thomas Roddick, Kesar Breen, and Roberto Cipolla. Large scale labelled video data augmentation for semantic segmentation in driving scenarios. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 230–237, Oct 2017.

[149] Yi Zhu, Karan Sapra, Fitsum A. Reda, Kevin J. Shih, Shawn D. Newsam, Andrew Tao, and Bryan Catanzaro. Improving semantic segmentation via video propagation and label relaxation. *CoRR*, abs/1812.01593, 2018.

[150] Md. Alimoor Reza, Hui Zheng, Georgios Georgakis, and Jana Koeck. Label propagation in rgb-d video. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4917–4922, Sep. 2017.

[151] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 950–957. Morgan-Kaufmann, 1992.

[152] Andrew Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 78–, New York, NY, USA, 2004. ACM.

[153] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[154] Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1058–III–1066. JMLR.org, 2013.

[155] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[156] Jawad Nagi, Frederick Ducatelle, Gianni A. Di Caro, Dan Cirean, Ueli Meier, Alessandro Giusti, Farrukh Nagi, Jrgen Schmidhuber, and Luca Maria Gambardella. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 342–347, Nov 2011.

[157] Chen-Yu Lee, Patrick W. Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 464–472, Cadiz, Spain, 09–11 May 2016. PMLR.

[158] Shuangfei Zhai, Hui Wu, Abhishek Kumar, Yu Cheng, Yongxi Lu, Zhongfei Zhang, and Rogerio Feris. S3pool: Pooling with stochastic spatial sampling. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4003–4011, July 2017.

[159] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018.

[160] Søren Hauberg, Oren Freifeld, Anders Boesen Lindbo Larsen, John W. Fisher, and Lars Kai Hansen. Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation. In *AISTATS*, 2016.

[161] Amy Zhao, Guha Balakrishnan, Frédo Durand, John V. Guttag, and Adrian V. Dalca. Data augmentation using learned transforms for one-shot medical image segmentation. *CoRR*, abs/1902.09383, 2019.

[162] Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. Learning to compose domain-specific transformations for data augmentation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3236–3246. Curran Associates, Inc., 2017.

[163] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2242–2251, July 2017.

[164] Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Synthetic data augmentation using gan for improved liver lesion classification. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 289–293, April 2018.

[165] Antreas Antoniou, Amos J. Storkey, and Harrison A Edwards. Data augmentation generative adversarial networks. *CoRR*, abs/1711.04340, 2018.

[166] Christopher Bowles, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger N. Gunn, Alexander Hammers, David Alexander Dickie, Maria del C. Valdés Hernández, Joanna M. Wardlaw, and Daniel Rueckert. GAN augmentation: Augmenting training data using generative adversarial networks. *CoRR*, abs/1810.10863, 2018.

[167] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating the future by watching unlabeled video. *CoRR*, abs/1504.08023, 2015.

[168] Xiaolong Wang, Kaiming He, and Abhinav Gupta. Transitive invariance for self-supervised visual representation learning. In *ICCV*, pages 1338–1347, 10 2017.

[169] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *ECCV*, 2016.

[170] Dinesh Jayaraman and Kristen Grauman. Learning image representations tied to egomotion from unlabeled video. *International Journal of Computer Vision*, 125(1):136–161, Dec 2017.

[171] Yin Li, Manohar Paluri, James M. Rehg, and Piotr Dollr. Unsupervised learning of edges. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1619–1627, June 2016.

[172] Deepak Pathak, Ross Girshick, Piotr Dollr, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6024–6033, July 2017.

[173] Zhongzheng Ren and Yong Jae Lee. Cross-domain self-supervised multi-task feature learning using synthetic imagery. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 762–771, 2018.

[174] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2070–2079, 2017.

[175] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[176] Huaizu Jiang, Gustav Larsson, Michael Maire Greg Shakhnarovich, and Erik Learned-Miller. Self-supervised relative depth learning for urban scene understanding. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[177] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV (1)*, pages 44–57, 2008.

[178] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt 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.

[179] Richard Zhang, Phillip Isola, Alexei Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 06 2018.

[180] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[181] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.

[182] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[183] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[184] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[185] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, pages 1–42, April 2015.

[186] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016.

[187] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018.

[188] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[189] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[190] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[191] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[192] Changan Chen, Frederick Tung, Naveen Vedula, and Greg Mori. Constraint-aware deep neural network compression. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[193] Bo Peng, Wenming Tan, Zheyang Li, Shun Zhang, Di Xie, and Shiliang Pu. Extreme network compression via filter group approximation. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[194] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[195] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *CoRR*, abs/1707.01083, 2017.

[196] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *arXiv preprint arXiv:1801.04381*, 2018.

[197] Michael Figurnov, Maxwell D. Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[198] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S. Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[199] Xin Wang, Fisher Yu, Zi-Yi Dou, and Joseph Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. *CoRR*, abs/1711.09485, 2017.

[200] Andreas Veit and Serge J. Belongie. Convolutional networks with adaptive computation graphs. *CoRR*, abs/1711.11503, 2017.

[201] Gedas Bertasius, Lorenzo Torresani, and Jianbo Shi. Object detection in video with spatiotemporal sampling networks. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[202] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. *In ICML*, 2015.

[203] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. *In ICML*, 2015.

[204] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Unsupervised domain adaptation with residual transfer networks. *In NIPS*, 2016.

[205] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. *In CVPR*, 2017.

[206] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *In CVPR*, 2017.

[207] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto. Unified deep supervised domain adaptation and generalization. *In ICCV*, 2017.

[208] M. Mancini, L. Porzi, S. R. Bul, B. Caputo, and E. Ricci. Boosting domain adaptation by discovering latent domains. *In CVPR*, 2018.

[209] H. Zhao, S. Zhang, G. Wu, J. Moura, J. P. Costeira, and G. J. Gordon. Adversarial multiple source domain adaptation. *In NIPS*, 2018.

[210] Y. Zhang, P. David, and B. Gong. Curriculum domain adaptation for semantic segmentation of urban scenes. *In ICCV*, 2017.

[211] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. V. Gool. Domain adaptive faster r-cnn for object detection in the wild. *In CVPR*, 2018.

[212] M. Wulfmeier, A. Bewley, and I. Posner. Incremental adversarial domain adaptation for continually changing environments. *In ICRA*, 2018.

[213] H. de Vries, F. Strub, J. Mary, H. Larochelle, O.Pietquin, and A. C. Courville. Modulating early visual processing by language. *In NIPS*, 2017.

[214] X. Huang and S. J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *In ICCV*, 2017.

[215] E. Perez, F. Strub, H.de Vries, V. Dumoulin, and A. C. Courville. Film: Visual reasoning with a general conditioning layer. *In AAAI*, 2018.

[216] Linjie Yang, Yanran Wang, Xuehan Xiong, Jianchao Yang, and Aggelos K. Katsaggelos. Efficient video object segmentation via network modulation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[217] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6450–6458, 2017.

[218] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[219] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *In CVPR*, 2015.

[220] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *In CVPR*, 2017.

[221] L. C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv preprint arXiv:1802.02611*, 2018.

[222] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 2008.

[223] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. *In ECCV*, pages 44–57, 2008.

[224] H. Jiang, G. Larsson, M. Maire, G. Shakhnarovich, and E. Learned-Miller. Self-supervised relative depth learning for urban scene understanding. *In ECCV*, 2018.

[225] S. Jegou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. *arXiv preprint arXiv:1611.09326*, 2016.

[226] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *In CVPR*, 2016.

[227] K. Simonyan and A. Zisserman. Imagenet classification with deep convolutional neural networks. *In NIPS*, 2012.

[228] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *In International Conference on Learning Representations (ICLR)*, 2015.

[229] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. Learning depth from single monocular images. In *Advances in neural information processing systems*, pages 1161–1168, 2006.

[230] Ashutosh Saxena, Min Sun, and Andrew Y Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):824–840, 2009.

[231] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick Van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015.

[232] Arunkumar Byravan and Dieter Fox. Se3-nets: Learning rigid body motion using deep neural networks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 173–180. IEEE, 2017.

[233] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. Codeslam-learning a compact, optimisable representation for dense visual slam. *arXiv preprint arXiv:1804.00874*, 2018.

[234] Konstantinos-Nektarios Lianos, Johannes L Schönberger, Marc Pollefeys, and Torsten Sattler. Vso: Visual semantic odometry. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 234–250, 2018.

[235] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, volume 2, page 7, 2017.

[236] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017.

[237] Xiaohan Fei, Alex Wang, and Stefano Soatto. Geo-supervised visual depth prediction. *arXiv preprint arXiv:1807.11130*, 2018.

[238] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

[239] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv preprint arXiv:1802.02611*, 2018.

[240] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[241] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian D Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(10):2024–2039, 2016.

[242] N.Mayer, E.Ilg, P.Häusser, 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 *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134.

[243] Ziyu Zhang, Alexander G. Schwing, Sanja Fidler, and Raquel Urtasun. Monocular object instance segmentation and depth ordering with cnns. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2614–2622, 2015.

[244] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 730–738, USA, 2016. Curran Associates Inc.

[245] Dan Xu, Elisa Ricci, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Multi-scale continuous crfs as sequential deep networks for monocular depth estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[246] Dan Xu, Wei Wang, Hao Tang, Hong Liu, Nicu Sebe, and Elisa Ricci. Structured attention guided convolutional neural fields for monocular depth estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[247] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[248] Ke Xian, Chunhua Shen, Zhiguo Cao, Hao Lu, Yang Xiao, Ruibo Li, and Zhenbo Luo. Monocular relative depth perception with web stereo data supervision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[249] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 3119–3127, 2015.

[250] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[251] Daniel Gordon, Ali Farhadi, and Dieter Fox. Re ^{3}: Real-time recurrent regression networks for visual tracking of generic objects. *IEEE Robotics and Automation Letters*, 3(2):788–795, 2018.

[252] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[253] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[254] Wei-Sheng Lai, Jia-Bin Huang, and Ming-Hsuan Yang. Semi-supervised learning for optical flow with generative adversarial networks. In *NIPS*, 2017.

[255] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[256] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.

[257] J Yu Jason, Adam W Harley, and Konstantinos 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.

[258] Zhe Ren, Junchi Yan, Bingbing Ni, Bin Liu, Xiaokang Yang, and Hongyuan Zha. Unsupervised deep learning for optical flow estimation. In *AAAI*, volume 3, page 7, 2017.

[259] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2, page 7, 2017.

[260] Simon Meister, Junhwa Hur, and Stefan Roth. UnFlow: Unsupervised learning of optical flow with a bidirectional census loss. In *AAAI*, New Orleans, Louisiana, February 2018.

[261] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Digging into self-supervised monocular depth estimation. *CoRR*, abs/1806.01260, 2018.

[262] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[263] Rich Caruana. Multitask learning. *Mach. Learn.*, 28(1):41–75, July 1997.

[264] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3994–4003, 2016.

[265] Keke He, Zhanxiong Wang, Yanwei Fu, Rui Feng, Yu-Gang Jiang, and Xiangyang Xue. Adaptively weighted multi-task deep network for person attribute classification. In *Proceedings of the 25th ACM International Conference on Multimedia*, MM '17, pages 1636–1644, New York, NY, USA, 2017. ACM.

[266] Elliot Meyerson and Risto Miikkulainen. Beyond shared hierarchies: Deep multitask learning through soft layer ordering. *CoRR*, abs/1711.00108, 2017.

[267] I. Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5454–5463, July 2017.

[268] Yuan Gao, Qi She, Jiayi Ma, Mingbo Zhao, Wei Liu, and Alan L. Yuille. Nddr-cnn: Layer-wise feature fusing in multi-task cnn by neural discriminative dimensionality reduction. *CoRR*, abs/1801.08297, 2018.

[269] Beyang Liu, Stephen Gould, and Daphne Koller. Single image depth estimation from predicted semantic labels. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1253–1260, 2010.

[270] Peng Wang, Xiaohui Shen, Zhe Lin, Scott Cohen, Brian Price, and Alan L. Yuille. Towards unified depth and semantic prediction from a single image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[271] Dan Xu, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. *CoRR*, abs/1805.04409, 2018.

[272] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[273] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[274] Chaoyang Wang, José Miguel Buenaposada, Rui Zhu, and Simon Lucey. Learning depth from monocular videos using direct methods. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2022–2030, 2018.