UNIVERSITY OF CALIFORNIA SAN DIEGO

Object Segmentation and Tracking in Videos

A thesis submitted in partial satisfaction of the
requirements for the degree of Master of Science

in

Computer Science and Engineering

by

Vijay Viswanath

Committee in charge:

Professor Garrison Cottrell, Chair
Professor Manmohan Chandraker
Professor Lawrence Saul

2020

The Thesis of Vijay Viswanath is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California San Diego

2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to acknowledge Professor Garrison Cottrell for his support as the chair of my committee as well as his guidance throughout the masters program. Providing advice and motivation even when exploring off topic areas, he was instrumental in providing me an opportunity as well as resources to learn and work on Deep Learning projects. The projects in this thesis were motivated by the works of Celene Gonzalez, the only person with confidence to wear eye tracking camera in public. I would like to thank her for helping me learn about the eye tracking camera as well as for allowing me to use the recordings for my project.

Thank you Michael Liu and Sumit Binnani for your ideas and advice on the projects which formed this thesis. I would also like to thank other members of the GURU team for their support, advice and help in setting up the environment and coding. Finally I owe a lot to my parents and my brother, who motivated me to keep going and in giving the best effort.

ABSTRACT OF THE THESIS

Object Segmentation and Tracking in Videos

by

Vijay Viswanath

Master of Science in Computer Science and Engineering

University of California San Diego, 2020

Professor Garrison Cottrell, Chair

Object detection and segmentation are some of the key components of Computer Vision, which have wide ranging real world applications. The current state of the art techniques in computer vision are based on Deep Neural Networks and one of the key challenges is using the state of the art techniques in these fields on novel images, and videos in different environments, and classes. These methods require expensive manual annotations and transfer learning to make them work on domains different from their training data sets. In this thesis, we explore both domain adaptation, and deep learning techniques that don't necessarily rely on the idea of a class, to help with the annotation of private videos. We implemented the initial idea of domain adaptation for directly annotating objects and followed with using video object segmentation

(VOS) tracking methods for propagating annotations. Their application to a novel video acquired in the GURU lab is explored as well as ways to improve their performance.

Chapter 1

Introduction

This work was motivated by the need to identify the objects that individuals focused on during human visual experience videos. These videos were recordings from a head mounted camera with eye tracking. It recorded what the person was viewing and the position in the image where the person was fixating at, for each frame. In order to make meaningful interpretations from these videos, it was important to identify the object being fixated at in each frame. For this purpose, we initially explored object detectors like YOLO[43], RCNN [12][44]. One challenge with these object detectors were in identifying objects like trees, roads, etc for which pixel level segmentation made more sense than a bounding box. The segmentation task involves labelling each pixel based on which object they belong to. One key difference between object segmentation and object detection is that, each pixel has a unique label in the segmentation task, while the bounding boxes can overlap in object detection. Both these tasks can serve in identifying objects fixated by humans, by recording the scene using a head mounted camera and identifying the location the person was fixating at, within the images taken.

Generally, producing annotations for object segmentation tasks is harder than object detection tasks, and this can be seen in public datasets that are currently available. MS COCO [26] is the most used dataset for object detection and has 2,500,000 labeled instances in 328,000 images and 82 object categories. The SUN dataset[26], [52] is another example with 908 scene categories and 3819 object categories. While a typical dataset for object segmentation like

Cityscapes [7] contain 25000 annotated images and 30 categories. The number of classes of objects in training data heavily favours object detection tasks.

However, we had the following reasons to the explore object segmentation networks and use domain adaptation to improve them. The videos we wanted to annotate were collected as part of the 'Day in the life' project, whose goal is to record the visual experience in the daily life of a person. The videos were collected by students on campus, with the chosen video being a student driving around the campus. Thus the video mostly features a city landscape with frequent outdoor settings. This heavily correlates with the Cityscapes dataset and also the need to identify roads, trees and shrubs in the videos. During manual annotations, we observed that trees were frequently fixated at along with pavements and when attention was focused on people, the gaze tended to frequently shift between different people. This prompted an interest in exploring the object segmentation tasks, which can cover trees and roads well. For this, we explored one of the state of the art segmentation network. To improve the performance in custom videos, a discriminator based domain adaptation technique was performed. Chapter 2 details this work. The chapter covers the working of an object segmentation network and a domain adaptation method used to improve the performance of the network in our novel videos. We chose HRNet[46], one of the SOTA methods in Cityscapes dataset, whose code is public and in Pytorch. HRNet uses multiple Fully Convolutional Blocks and features at multiple resolutions to build a segmentation map.To adapt to new videos, Domain Adversarial Neural Network (DANN)[11] was used. The advantage of DANN is, it can be jointly trained with an already trained classifier, is fully unsupervised and works on using the same classifier network trained on initial domain.

1.1 Video Object Segmentation Tracking

The interest in identifying whether attention is changing between two objects of same category led towards exploring video object segmentation (VOS) tracking techniques. The

existing public datasets cover a lot of day-to-day objects and it could be useful to use domain adaptation with different object segmentation/detection networks to produce initial annotations on private videos. But the challenge gets quickly complicated by the need to use an ensemble of models on the same video and picking one label for the gaze location. The objects in the video may not be covered by a single dataset/network or may even have new object classes needing tracking. In this case, its useful to have a network which can track any generic object across the video from one training example.

Video Object Segmentation (VOS) directl tracks an object across frames using pixel level annotation. This semi-supervised task involves using an annotation from the first frame annotation and then tracking the object across the video frames. This is different from object detection/object segmentation in that the class of the tracked object is no longer important and only the object of interest should be tracked even if there are other objects of the same class present in the video. For example, if the annotated object in the first frame is a car, the same car needs to be tracked in subsequent frames and the network should not mark other cars present in the video. The other challenging aspect is occlusion and imperfect information in the first annotated frame. This is very useful in reducing the annotation efforts as a good tracking network should be able to adapt to new object classes. Also, an ensemble of models would no longer be needed if the same network can be adapted to different videos and objects, either through domain adaptation or one shot learning methods.

The work in this chapter involves using RANet[51], a state of the art VOS network on DAVIS 2016 (Densly Annotated VIdeo Segmentation)[6, 40, 38] dataset which uses single object tracking. Even though its an end-to-end network and relatively fast, it had the highest performance in DAVIS 2016 dataset. But its performance is much less than the best models in DAVIS 2017 dataset. The DAVIS 2017 dataset has single as well multi object tracking videos, which is an area where RANet could use some improvement. To build upon previous work, we explored the network and based on the initial analysis, we implemented 3 ideas to improve the

network's performance on DAVIS2017 dataset : using better loss function in terms of IOU loss, using a discriminant score and non maximal suppression to select pixels in Ranking Attention module of RANet. The loss function improved the performance by 4% while the other ideas didn't improve the performance much and led to over fitting. Further analysis on RANet revealed certain features of the network which might make the updates to RAM module less useful. The details and results are mentioned in Chapter 3.

1.2 Unsupervised methods in Video Object Segmentation Tracking

The VOS task does not require the models to learn class identities. A good distinguisher for objects should work well with the task without any need to learn what each object corresponds to. This makes unsupervised learning attractive for the task. An unsupervised method using only raw videos can not only make use of large video datasets, but can also be fine tuned further on specific videos.

The unsupervised methods typically have much lower performance than supervised methods. But a recent paper proposed Memory Augmented Self Supervised Tracker(MAST)[54], which can train on raw un-annotated videos and claims performance on-par with some of the supervised methods. This prompted an interest in exploring this method and trying to reproduce the results. The code for the model was not available in public and so we attempted to implement it from scratch. References were made from publicly available code for Corrflow[23], the predecessor to MAST. The performance comparable to the original paper could not be achieved, which may have been related to certain compromises we made during training. The results achieved and further details are mentioned in Chapter 4.

1.3 Related Work

1.3.1 Object segmentation and domain adaptation

Object segmentation and detection has been some of the hottest topics in computer vision. Various methods have been devised to do object detection tasks. RCNN [13][12][44] is one of the most famous methods, using region proposals and classification of objects. Starting from a mix of neural networks and other machine learning techniques in RCNN, the faster RCNN is an end to end neural network. A fully convolutional network outputs anchor boxes at different scales for region proposal, with a box-regression layer outputting box coordinates and box-classification layer outputting the object label, or a ”no-confidence” vote. The detector network outputs the class for these boxes, and various techniques like non maximal suppression are used to select good boxes. YOLO[42] and YOLOv2 9000[43] use fully convolutional neural networks, with YOLOv2 using anchor boxes similar to Faster RCNN. Instead of two separate networks, YOLOv2 outputs a prediction for each pixel in the final convolutional layer output. The network down samples the input by 32 when the final layer is reached and each pixel in the final feature map makes predictions for 5 anchor boxes. And each anchor box makes prediction for the bounding box (bbox), the objectness score and Logistic output for 80 classes. Using pass through layers and up sampling, predictions are also made as higher resolutions. Single Shot Detector (SSD) [29] is another network which is fully convolutional and makes predictions per pixel in the final output feature map. SSD performed better and faster than YOLO, but in turn was overcome by YOLOv2.

In object segmentation, the network is expected to label each pixel based on which object they belong to. Thus the output needs to have accurate object boundaries in addition to size and location. This is especially useful for identifying the object being fixated at, as there will be no overlap between object predictions. Mask RCNN[16] is an improvement over faster RCNN for the instance segmentation task in MS-COCO. It decouples mask and class predictions, making class agnostic binary pixel masks in regions of interest. In contrast, several

networks use an encoder followed by up-sample decoder to generate high resolution segmentation masks. UNet[45] uses skip connections from encoder to decoder and up convolutions. Hourglass networks[37] proposes using stacks of modules with a down-sample and up-sample pathways to process information at low and high resolution. Segnet[1] passes on pooling indexes instead of skip connections to improve memory utilization. GridNet[9] uses both high resolution and low resolution pathways, with the network pathways similar to UNet and Full Resolution Residual Network[39] inside it.This idea is refined by HRNet.

Domain adaptation involves using a trained model in one domain to do classification in another(domain 2). This is in contrast to transfer learning, where the labelled data in domain 2 is used to fine tune the network trained in domain 1. The aim of domain adaptation is to achieve performance close to transfer learning while using fewer or no labelled data in domain 2. Generally there are 3 approaches to domain adaptation: Reducing divergence between feature vectors in different domain through statistics, reducing divergence between feature vectors using adversarial learning and generating new training samples using reconstruction. In Statistical divergence reduction, Maximum Mean Discrepancy is a good example which is used in[30][50], to bring the mean feature vector of different domains at different layers to be close to each other. In [50][19], the authors use different forms of entropy losses to make the prediction probabilities peaky over one class.Domain Adaptive Neural Network (DANN) [11] and its improvement Adversarial Discriminative Domain Adaptation(ADDA)[47] are examples of adversarial learning, where a discriminator network tries to identify the domain from feature vectors and the classification network is forced to learn domain invariant features. In [3], a generator network is used to adapt images from one domain to another and learn grasping policies in the new domain. For our particular task, the videos are unlabelled and the labels closely align with the labels used in Cityscapes dataset. Hence we decided to go with ADDA.

1.3.2 Video Object Segmentation tracking

The VOS challenge has been approached with end-to-end predictors as well as ensemble of models, with each model doing a particular defined task. Among the end-to-end predictors, [22] proposed a single Deep Network which will take current frame, previous object mask and predict the current frame's object mask. [5] uses a single network on DAVIS2016 dataset and then fine tunes the network for each of the test videos using one shot learning on the first annotated frame. CINM [2] also takes in a 4 channel input (RGB image + mask) and uses spatio-temporal Markov Random Field Model to output a refined mask. Lucid [21] proposes a single network which uses a warped previous mask using flow estimation as well the flow estimation itself as inputs along with the target frame to predict the new mask. Their main contribution was introducing new data augmentation methods to produce synthetic data from annotated first frame of target videos. Additionally, they used pixel level semantic labelling as an extra input. For temporal consistency, they propose refining the current mask by removing all connected components which do not overlap with connected components in previous frame.

Some of the multi-network models include PreMVOS [31] and [24]. [24] uses a Re-identification module which proposes a rough bounding box for target object and refines it into a mask using attention mechanism. These masks are temporally refined using a module which does bi-directional mask propagation. This is done by using flow guided warping of a "memory" from the nearest key frame. [25] also proposes similar mask propagation and re-identification modules. PreMVos [31] is currently the best performing model for DAVIS2017 test dataset with mean J&F score of 71.6. Similar to RCNN, it uses a Proposal generator for bounding box proposal, a proposal refinement network, mask propagation using optical flow computed by Flownet2.0[17], re-identification network and finally a proposal merger to estimate the masks. PreMVOS also uses online learning, where it updates its networks on 2500 augmented images generated for each of the first annotated frames of val/test videos. OSVOS-S [33] builds on top

of OSVOS, using semantic segmentation networks such as Masked RCNN to improve the result of OSVOS.

1.3.3 Loss functions for segmentation

All the networks mentioned above use cross entropy loss for losses associated with predicted mask. Some papers mention using weight balanced Binary Cross Entropy (BCE) loss like[5],[33] while others don't explicitly specify. Tsung et al. [27] proposed focal loss, modifying the regular Cross Entropy (CE) loss to focus more on hard to classify examples. When regular CE uses the term $\sum_t y_t log(p_t)$, focal loss proposes adding the term $(1-p_t)^\gamma, \gamma \in [0,5]$ to CE loss. This is to weight the highly miss-classified examples more. The final loss becomes$\sum_t y_t(1-p_t)^\gamma log(p_t)$. In UNet [45], authors proposed a weight term along with CE loss which adds weight based on class imbalance as well as to force the network to learn the small separation borders. Various papers in the literature have also suggested using alternative loss functions to directly optimize for IOU score.

There have been various attempts to directly maximize IOU. The challenge is, the direct IOU computation is non-differentiable. This has been worked around by directly approximates the IOU loss to a differentiable form, called soft IOU[41] and [35] . Consider X to denote the final output of the network just before sigmoid layer and M to denote the true mask for image I. Let $x_i \in X$ be the output for pixel i in the final layer, then the soft IOU loss in [41] is defined as:

$$L = 1 - I(X,M)/U(X,M), where \tag{1.1}$$

$$I(X,M) = \sum_{i \in I} sig(x_i) * M_i \tag{1.2}$$

$$U(X,M) = \sum_{i \in I} sig(x_i) + \sum_{i \in I} M_i - \sum_{i \in I} sig(x_i) * M_i \tag{1.3}$$

where $sig(x_i)$ is sigmoid function on x_i.

Similar IOU loss has also been proposed for bounding box regression[53]. Instead of per pixel

summation, the differentiable IOU is computed as a function of predicted and true bounding box dimensions. Nagendar et al.[36] proposes using a Neural Network to approximate the non differentiable IOU loss function. The neural network takes as input the sums of probabilities of pixels in TP, FP & FN and predicts the IOU loss value. Fausto et al. [34] used differentiable version of Dice coefficient as loss, similar to differentiable IOU loss in eqn (1). Similarly another author proposed using negative log of soft IOU as loss as in eqn (4):

$$L = -log(\frac{I(X,M)}{U(X,M)}) \tag{1.4}$$

where I(X,M) and U(X,M) are defined as in eqn (2) and (3).

1.3.4 Unsupervised Learning for Object Segmentation

Vondrick et al. [49] was one of the first authors to propose using a pixel-to-pixel matching learning. The authors proposed using grey scale images and psuedo label for each pixel during training. The network was trained to match pixels with same pseudo label. The label for each pixel was generated from K-means clustering on LAB color space of original images. Corrflow[23] improved upon this, using RGB input with channel dropout. UnOVOST [32] proposed a different approach of using Mask RCNN to get region proposals and then tracked the object through frames using optical flow, temporal information, and feature vector associated with each region proposal. Tengda et al.[15] took a generative approach, using self supervision through input video blocks to predict future video blocks.

1.4 Summary

In this thesis, we aim to use existing models to generate annotations for novel videos. First we explore using a pretrained object segmentation network for generating annotation. A state of the art network, HRNet was picked because of the similarity between the dataset on which it was trained and the novel videos on which it needs to be applied. Due to lack of

annotations in the novel videos, unsupervised domain adaptation is explored to improve its performance on these videos.

The lack of object class variety in pretrained models led to an interest in techniques which can track new object classes from one or few annotations. For this, Video Object Segmentation (VOS) tracking networks were explored. The networks in this domain use a single annotation of an object to track that specific object for the rest of the video and does not rely on being trained on object classes. We picked RANet for its state of the art performance in single object tracking and then improved its performance in multi object tracking. Also we explore using the network to track objects on our novel video of a car driving around the University campus.

Finally, we explored unsupervised learning methods for VOS tracking. Unsupervised learning methods have the added advantage of being trainable directly on our novel videos while not requiring any annotations, but usually have much lower performance than supervised methods. A recent paper proposed an unsupervised model whose performance is on par with several supervised models. The code for the model is not available online. So we attempted to replicate the results and discuss our results in chapter 4.

Chapter 2

Object Segmentation and domain adaptation

Semantic segmentation involves identifying the objects in the scene at a pixel level detail, with each pixel belonging to one of the object classes. Unlike object detection with bounding boxes, there is no overlap between objects as a pixel can only belong to one class. This makes it possible for having object segmentation masks for vegetation and roads. These classes can be thin, long, and curvy in the image, so a bounding box encompassing them may end up also covering a lot of objects of other categories. In addition, a pixel level annotation would reduce ambiguity of identifying which object the wearer of eye tracker was fixated at. The gaze location identified by the eye tracker is a pixel coordinate in the image and by having a measure of the error in gaze location, we can have an accurate idea of possible objects being looked at. As a example, a person might be looking at a toy under a table. In pixel level annotation, the pixels near toy might that of background, with table quite far away. But a bounding box for the table would end up encompassing the bounding box of the toy and create ambiguity.

HRNet is one of the state of the art network architecture in semantic segmentation network, with the model also being used for other tasks such as pose estimation, object detection and facial landmark detection. We specifically selected the pretrained model for Cityscapes dataset, because of the similarity of the dataset with our videos. The videos were taken around the university and thus features an urban settings, with building, roads, vegetation and cars

being prominent. We picked a video of a person driving the car and generated annotations using the pretrained model. Seeing that the generated annotations were not good, we used Domain Adversarial Neural Network (DANN), an unsupervised domain adaptation technique. The use of domain adaptation was found to be quite helpful and the new annotations generated appear to have more consistent object boundaries and generally better identification of objects. Since we did not have ground truth annotation for the videos, visual inspection was used for comparing the results.

2.1 Method

HRNet[46] relies on using both high resolution and low resolution channels to construct a feature map, which is used by the classification layer. The model comprises of 4 stages, with each stage having various number of layers working at different resolutions. This can be seen in Figure2.1, where the 4 stages have 1, 2, 3, and 4 layers. The top layers in each stage works at input image resolution while each subsequent layer has half the resolution of the layer above it. Each layer consists of a fully convolutional block with residual connections.

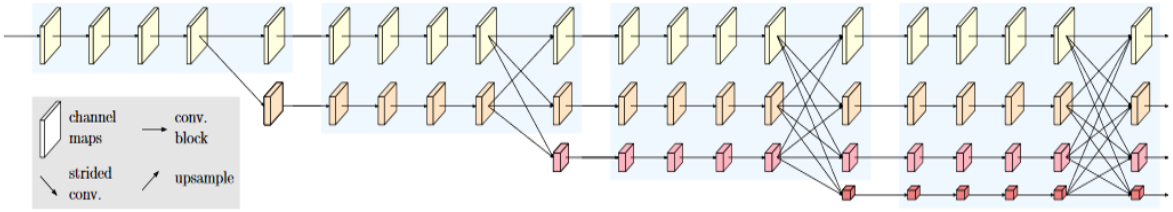

Figure 2.1. [46] An example of a high-resolution network with four stages. Please note that in the connections between stage, a branch in next stage receives input from all branches in the previous stage.

To make full use of processing at different resolutions, the input to one layer of a stage uses all the outputs from the previous stage. For example, in Figure2.1, the output of 3 layers in stage 3 are fed into the 4 layers of stage 4. For input to layer 2 in stage 4, the outputs of layer 1,2 and 3 in stage 3 are summed and fed. The output of layers can have different resolution and also different number of channels which cannot be directly summed. This is resolved by having

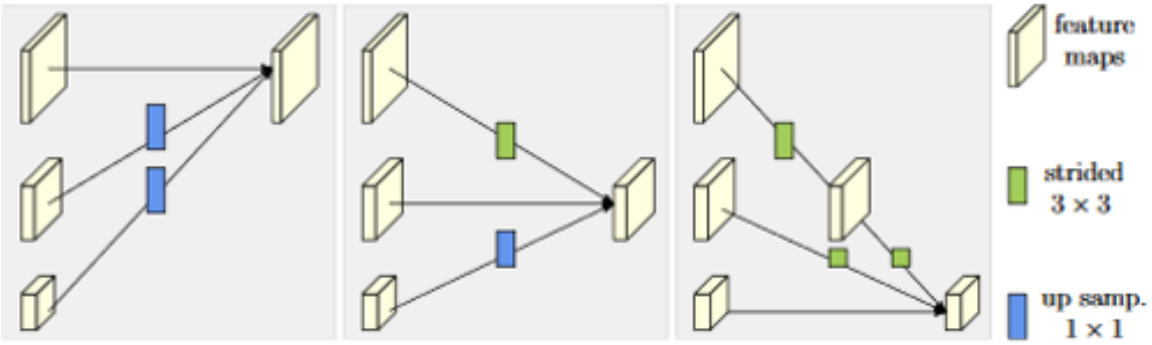

Figure 2.2. [46] Connections from one stage to the next stage.

bi-linear up-sampling with convolutions and strided convolutions for down sampling. This is shown in Figure 2.2. The CNN block doing the up-sampling or down sampling also takes care of adjusting the channel numbers.

2.1.1 Domain adaptation through Adversarial Training

We adapted the HRNet trained on cityscapes to our videos using a simple convolutional network classifier as discriminator. The Domain Adversarial Neural Network (DANN) method involves the discriminator trained to binary classify images based on whether they are from source domain or target domain. The input to the discriminator is the feature vector from HRNet and the target label is 1 (source domain) or 0 (target domain). Thus the simplified loss for the discriminant is L_D [11]:

$$L_D = -\frac{1}{|X_s|}\sum_{x_s \in X_s}[log(G_d(G_f(x_s)))] - \frac{1}{|X_t|}\sum_{x_t \in X_t}[log(1 - G_d(G_f(x_t)))] \tag{2.1}$$

where X_s and X_t are source and target domains respectively, G_d is the discriminator network and G_f is the feature extractor network. Since we want the feature extractor G_f to increase the L_D, one way is to train the feature extractor and discriminator alternately. But both can be trained together by using a gradient reversal layer[11] in between the feature extractor and the discriminator. The gradient reversal layer has no learn-able parameter. During forward pass,

it simply passes on the input as its output (identity transformation). During backward pass, it reverses the sign of the gradient before passing it back. With this, the whole network can be jointly trained using back prop using a single loss L:

$$L = L_c ls(X_s, y_s) + \lambda L_D \tag{2.2}$$

where $L_c ls(X_s, y_s)$ is the regular supervised classification loss in source domain. λ is a hyper parameter which controls the relative importance of each loss.

2.2 Experimentation and Results

We took the HRNet training code and the pretrained model in the Cityscapes from the author's site. The training procedure involved various data augmentations of which scaling was one. The input and therefore feature vector output of the network could vary between mini batches. Thus, the discriminator network consisted of 3 layers of CNN followed by an adaptive max pool layer and a fully connected classification layer. The adaptive max pool layer is a max pool layer with dynamic kernel size such that the output feature map size would be constant irrespective of the input feature map size. The adaptive max pool layer was configured to always provide output of dimension 1x1x64. This relatively simple network was enough to adapt the network. The lambda was chosen to be 0.01.

Since there was no true annotation available in the target domain, we checked the performance by comparing the outputs of the base network(pretrained model) and the adjusted network. The results can be seen in Figure 2.3. Overall, the predictions are more consistent and accurate compared to the initial network. For example, the adapted network frequently classifies the dashboard of the car as a road. The rear view mirror is classified as either a road or a car, depending on the frame. Both the rear view mirror and the dashboard are not present in the classification list of cityscapes and thus the network does not know of their existence. One thing to note is that, the trained HRnet model uses only 19 of the 30 classes in the Cityscapes dataset

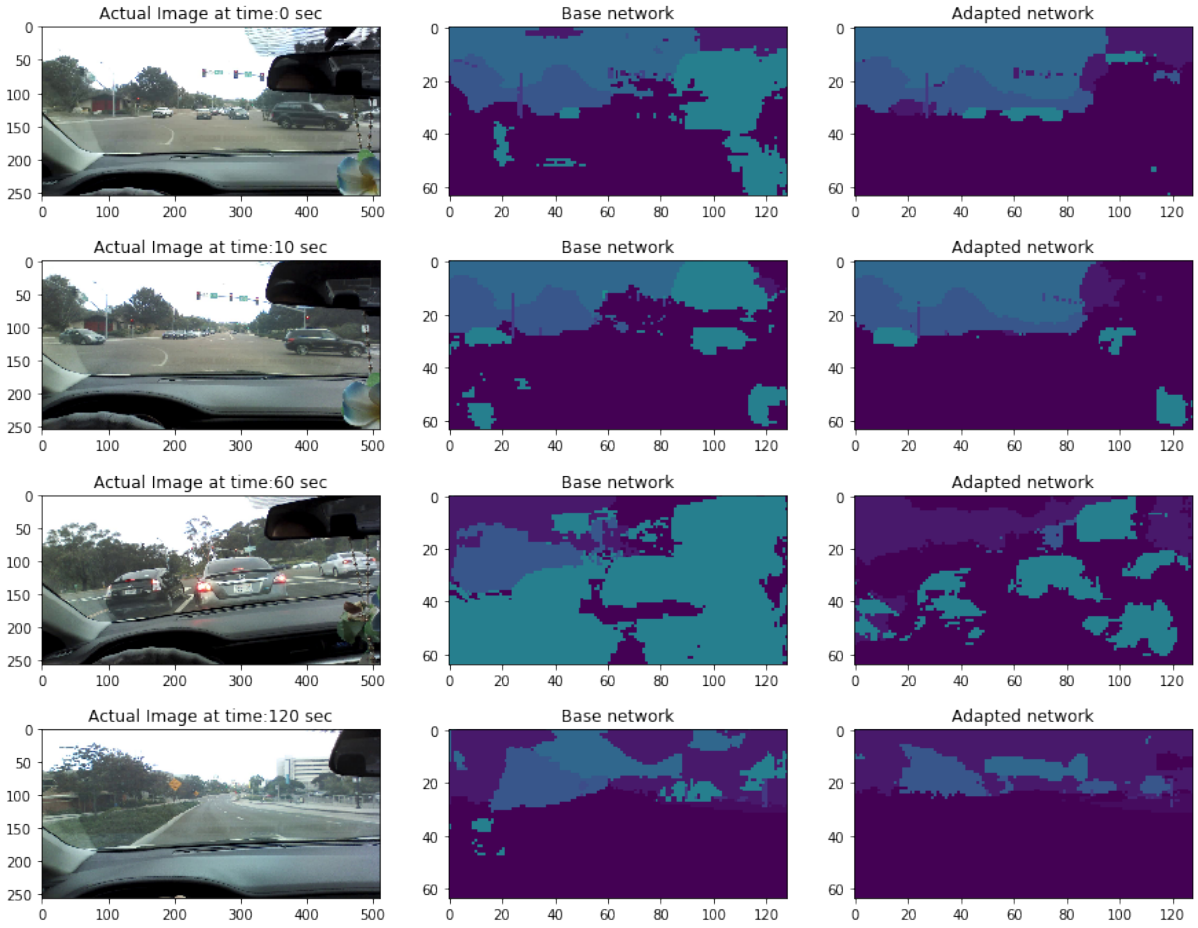

Figure 2.3. Comparison of base model and adapted model on the private video. The video is from the head mounted camera recording the scene seen by a person driving. As can be seen, the adapted network generates smoother and consistent predictions even if it miss classifies a lot of pixels.

and the label numbers associated with class names differ from the list provided by cityscapes. Even though the domain adaptation is improving the results, its still far from a good detector. In many instances, the network is miss-classifying vegetation as buildings or completely not detecting some cars in the road. This can be seen in Figure 2.4, where the network misses a lot of vegetation area in the time frame at 85 sec and a car in the time frame at 63 sec.

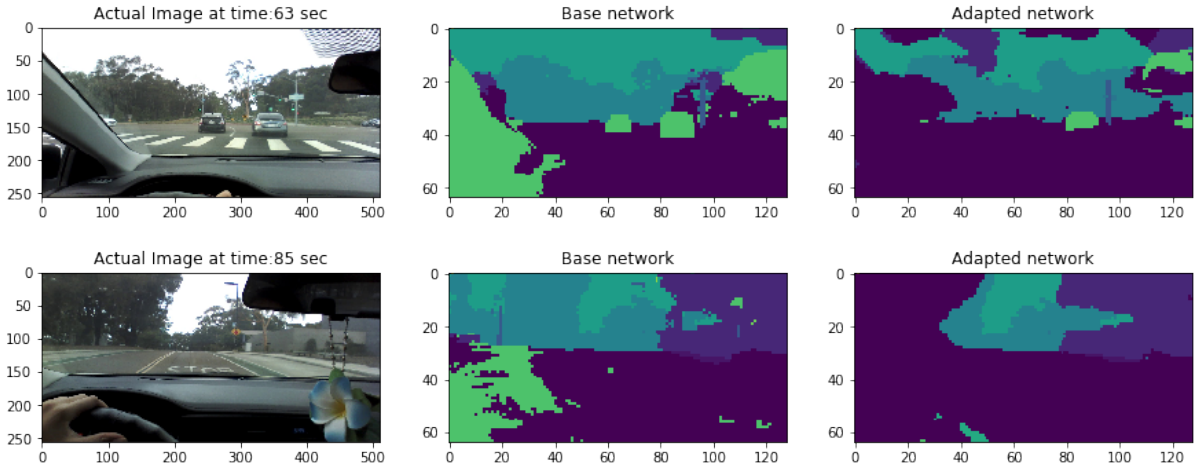

Figure 2.4. Instances where adapted network could be performing worse than base network.

2.3 Conclusion

The domain adaptation can be used with a pretrained model to generate initial annotations which can then be refined manually. The Figures 2.3 and 2.4 show that the prediction results from domain adaptation are smoother and usually more accurate than the unrefined network. We did not have true annotations to give out a measure in IOU or accuracy improvement. Nevertheless, domain adaptation reduces erratic annotations which should reduce the effort for manual refining. One thing which can be seen is that when the network is forced to classify new classes into some existing classes, it does not always classify it to the nearest trained class. For example, in Figures 2.3 2.4, the flower hanging inside the car is a new class of object not present in the annotated data trained by HRnet. It could be considered as vegetation. But the network is either classifying it as road or another car, because of how contrasting it is with existing vegetation. The network could be trained to identify it as separate class using some manually annotated data, using transfer learning or domain adaptation. But this could be expensive for short videos and for objects with sparse occurrence. A technique which can track the marked object from one example would be immensely helpful for this task. This is explored in the next chapter.

Chapter 3

Object Segmentation and Tracking

The experiments with domain adaptation and image segmentation made us realize that manual annotation might be necessary to accurately identify the fixated object as well as to train future object detection and segmentation networks to identify new object classes. The output from adapted HRNet could be used as an initial annotation which can then be manually refined. But these do not make use of the annotations after manual refinement. The frames in a video are correlated with each other and thus we should be able to use the refined annotation in frame 1 to improve the initial annotation in subsequent frames. This prompted us to explore the literature and select a semi-supervised Video Object Segmentation tracking task. The methods for this task rely on first frame annotation to propagate those annotations across the whole video. They usually have very good performance on frames close to annotated frame and the performance decays as the temporal gap increases. We selected RANet[51] for its state of the art performance in single object tracking in DAVIS 2016 dataset and its good performance as a single end-to-end network in DAVIS 2017 multi-object tracking dataset. We analyzed the network, tried improving its performance and then evaluated its use in the task of propagating annotation on a video.

3.1 Method

RANet uses the first (template) frame I_0, its mask M_0 and the current (target) frame I_t to compute the correlation map. The correlation map consists of 512 maps, 256 of which belongs to

foreground and rest 256 for background. All 512 maps are constructed by finding the correlation of feature map of I_t with feature vectors of selected pixels from I_0. Thus, 256 FG pixels and 256 BG pixels needs to be selected from I_0 and this is done by using Ranking Attention Module (RAM). The correlation map is used to refine the previous mask M_{t-1} to create current mask M_t.

To create feature vectors for pixels, ResNet 101 is used with batch norm layers replaced by

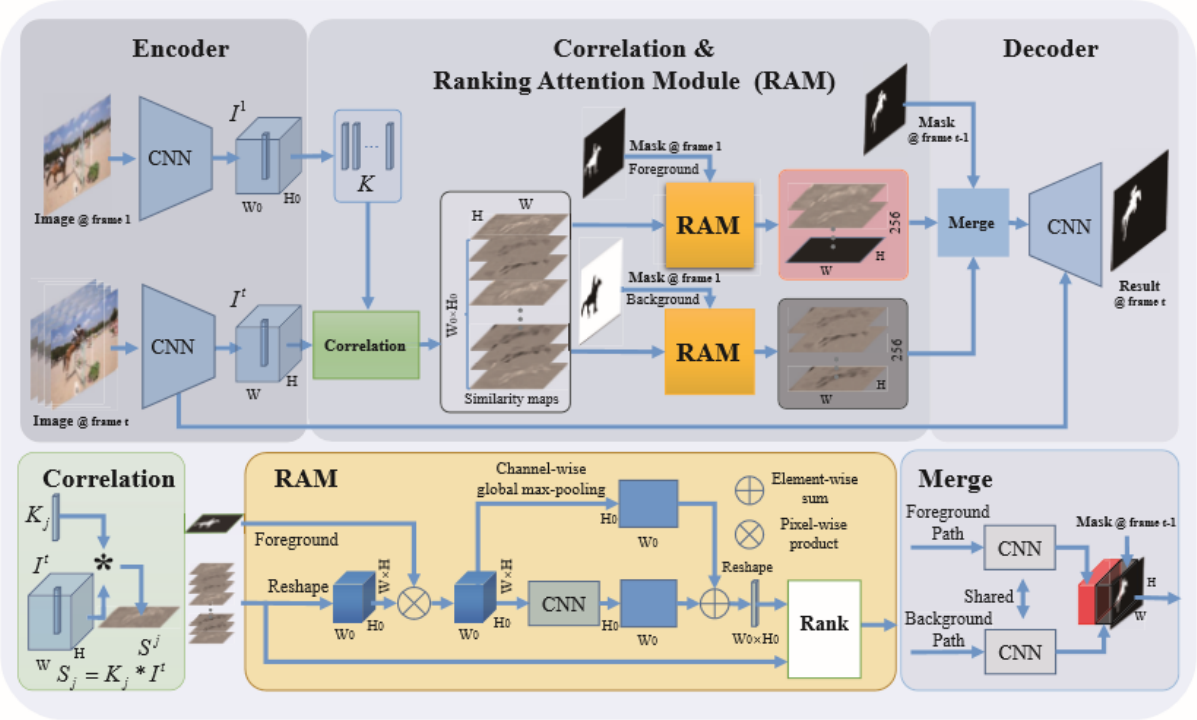

Figure 3.1. [51] Illustration of the RANet. The correlation of the features extracted by Siamese networks is used as a similarity map. The output similarity map and template mask are fed into the RAM module to rank and select the foreground/background similarity maps.

instance norm based on idea from [48]. The authors also used Fully Convolution Block to convert the 2048 sized feature vectors into 512 size for each pixel. Each feature vector is then normalized.

The overall architecture of RANet is shown in Figure 3.1. Using the ResNet 101 encoder layer, a WxHx512 sized feature map is generated for both I_0 and I_t, called F_0 and F_t respectively. For each pixel in F_0, the highest correlation it has with any of the pixel in F_t is found. This is used to create WxH correlation score matrix C_t. Then the feature map (WxHx512) is given

to a Fully Convolution Block called Ranking module which again generates a scalar score for each pixel and gives score matrix R_t. Initial mask M_0 is used to get the pixels corresponding to foreground(FG) and background(BG) in F_0. Among the FG pixels, the top 256 pixels are selected based on the total score. If there are less than 256 pixels, dummy pixels with 0 valued feature vectors are used as fillers. The same is done for BG pixels as well. Then these maps and the previous frame's mask are fed into the decoder, which generates the final segmentation mask.

3.2 Experimentation and Results

3.2.1 Analysis of RAM module

As can be seen from Figure 3.2, the output after RAM module has values very close to one. The image used was taken from 'India' validation set, where we have to track 3 people in a crowded street. The normalized correlation maps show a rough silhouette of objects and the RANet uses a decoder layer which is a Fully Convolution Block with Instance normalization and residual connections to refine these correlation maps. This is demonstrated in Figure 3.3. The object to be tracked is much more prominent and the values are no longer close to each other.

Still, it can be seen that the correlation maps have a lot of activation for areas with no object (trees) and with similar objects (other people). These parts are refined by later layers which also uses the previous mask as temporal information.

3.2.2 Use of Temporal Information in RANet

RANet uses two methods for temporal information transfer, of which only one is mentioned in the original paper. The first method (and the one mentioned in the paper) is to use the previous mask as an input along with the correlation maps for the current frame. When the template mask was used instead, the mean IOU score reduced by 4% in the validation set. Thus, this step provides a significant boost to performance.

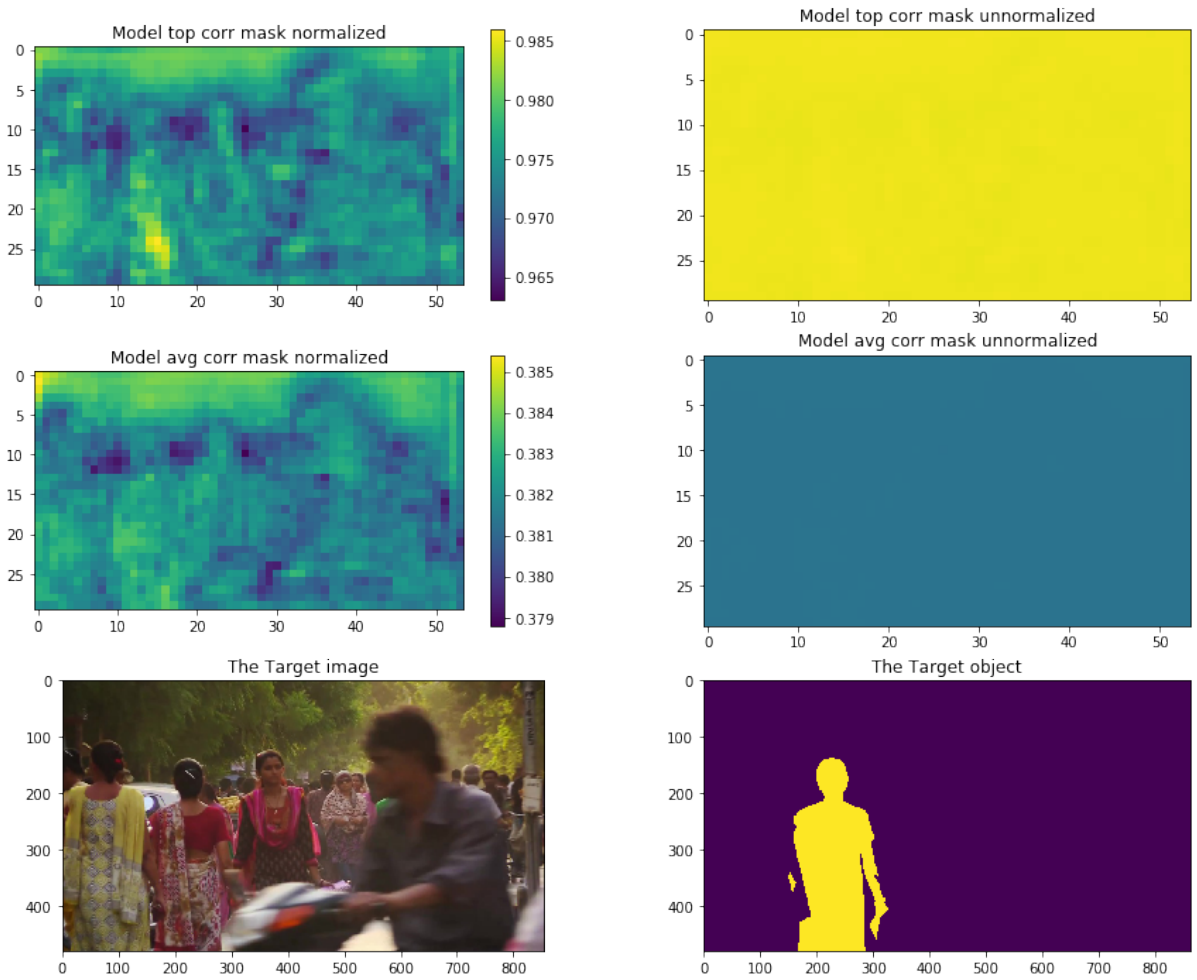

Figure 3.2. Correlation Map from RAM Module. By normalization, the values are scaled from (min, max) to (0,1). By unnormalized values, it means the values are kept as it is and clipped with min=0 and max=1 for displaying.

The second method is bounding box cropping. A tight bounding box is computed on the previous frame mask M_{t-1} and loose bounding box of 1.5 times the size of this tight bounding box is used to crop the current frame I_t. The object segmentation is done only within the cropped image. For multi object scenario, the tight bounding box is built to encompass all the objects in the previous frame. Removing this step reduced the performance by a further 4% in IOU score. The results are shown in Table:3.1. The J here is Jaccard distance, measuring the IOU for region similarity. F score is a measure of contour accuracy, that is how aligned the boundaries of true and predicted masks are. For both J and F mean, the higher values indicate better prediction. The decay term

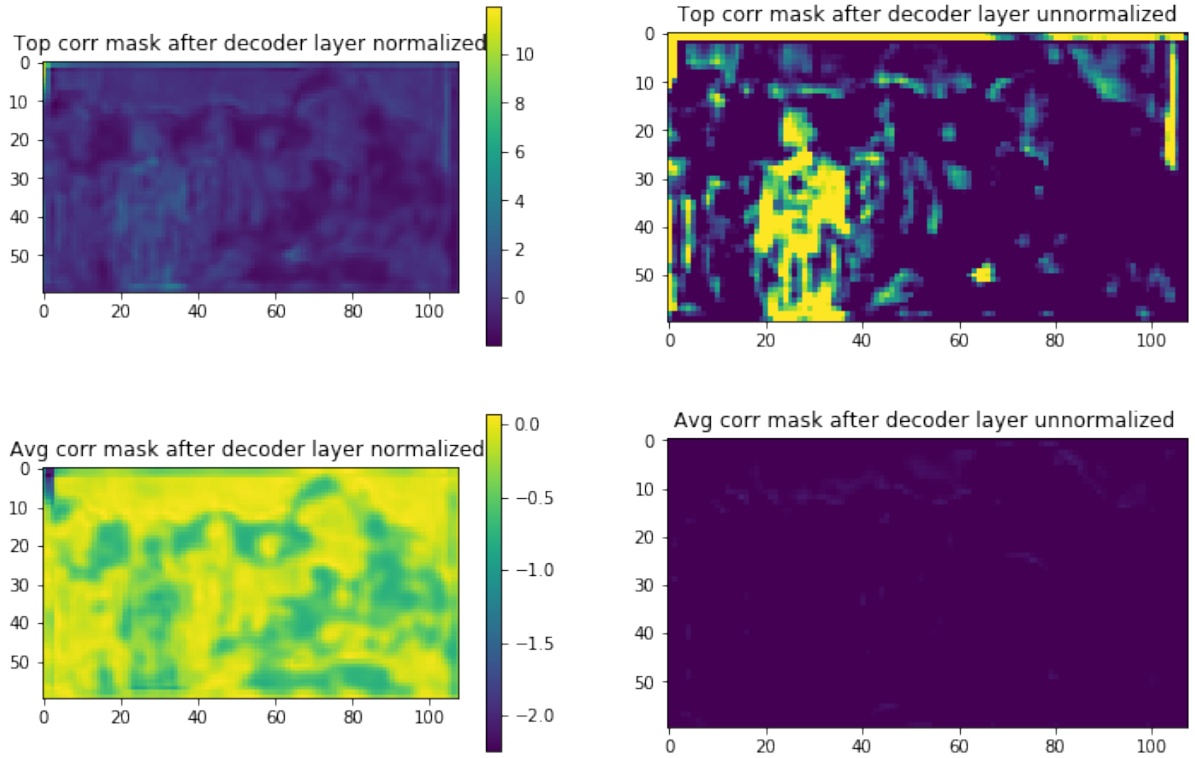

Figure 3.3. Correlation Map after decoder layer. By normalization, the values are scaled from (min, max) to (0,1). By unnormalized values, it means the values are kept as it is and clipped with min=0 and max=1 for displaying.

is a measure of how fast the J or F value degrades as we predict for later frames of the video, with lower the better. Using Bounding Box (BBox) for prediction in training dataset reduced the performance by 1%. This might indicate a slight over-fit, since the BBox method is not used during training.

Table 3.1. Ablation study on modifications using temporal information. M_{t-1} denote using the predicted mask for previous frame as initial estimate for current frame t.

Model	**J&F Mean**	**J Mean**	**J Decay**	**F Mean**	**F Decay**
RANet	0.677	0.65	0.22	0.70	0.23
RANet without M_{t-1}	0.632	0.60	0.29	0.66	0.28
RANet without M_{t-1} and Bbox crop	0.591	0.55	0.31	0.629	0.309

3.2.3 RANet Modifications

Compared to training of RANet by the authors, the common data augmentations we used were cropping with scaling and random rotations of +-30^0. The thin spline transform was dropped and instead random horizontal/vertical flips were used. Whatever transform was applied on an image, the exact transform was applied on its mask. The flip transforms were applied uniformly on the template image/mask and target image/mask while rest of the transforms applied could differ between the template and the target. That is, if template image was decided to be vertically flipped, then the template mask, target image and target mask would also be vertically flipped. But if the template image was chosen to be rotated by $+15^o$, then the template mask is also rotated by $+15^o$ degree but the target image and target mask could be rotated by a different angle. The vertical flip was integrated in the hope of making the network less reliant on class information and make the network learn similarity between images. The limits for rotation were based on the limits used in original RANet paper.

First modification was using soft IOU loss based on [41] instead of BCE loss. Let X be the final output of the network just before sigmoid layer and M be the true mask for image I. Let $x_i \in X$ be the output for pixel i in the final layer, then the soft IOU loss is defined as:

$$L = 1 - I(X,M)/U(X,M), where \tag{3.1}$$

$$I(X,M) = \sum_{i \in I} sig(x_i) * M_i \tag{3.2}$$

$$U(X,M) = \sum_{i \in I} sig(x_i) + \sum_{i \in I} M_i - \sum_{i \in I} sig(x_i) * M_i \tag{3.3}$$

where $sig(x_i)$ is sigmoid function on x_i. Thus, there is no thresholding involved and the predicted probabilities are directly used to compute the loss, making it differentiable. This loss has to be applied on per object basis in an image pair.

The second and third modifications were based on improving the pixel selection in RAM

module. The foreground(fg) correlation maps needs to be matched with the true foreground in target map and vice versa for background (bg) correlation maps. For this, a discriminant score was added along with the scoring mechanism of RANet. This discriminant score was computed as follows: For each fg pixel in template frame, its correlation with other pixels in the template image was computed. The sum of top 8 pixel correlation values with other fg pixels were take and top 8 correlation values with bg pixels were subtracted. Thus fg pixels which match well with other fg pixels and less with bg pixels in template frame will get highlighted. Similar discriminant scoring was done for bg pixels as well. This discriminant score was added to the original score computed by RANet, with a relative weight of 0.5 (a hyper parameter which was not explored for tuning).

The other modification was using non maximal suppression. If we picked one pixel in the neighbourhood, selecting another pixel in the same neighbourhood might give less extra information. But since the number of pixels might be limited (esp for fg) and we are using a ranking method, the score of maximal pixel in the neighbourhood was increased instead of suppressing the score of other pixels. This way, if the number of maximal pixels are less than 256, other pixels with high scores would be picked who may not be maximal in the neighbourhood. But maximal pixels in the neighbourhood would be preferred first.

For all modifications, we initialized the RANet with the trained model from author's site. All networks were trained with the same data augmentation methods, same learning rate of 10^{-6} and early stopping on validation dataset. The results are shown in table: 3.2 and 3.3. The RANet corresponds to the original model trained further with BCE loss. It was over-fitting immediately and so the original weights were used for comparison. RANet + IOU corresponds to using original RANet with IOU loss. The network started over-fitting in one epoch. RANet + IOU + disc corresponds to using RANET with IOU loss and discriminant scoring in RAM module. It over-fit after 1 epoch. Finally, RANet + IOU + disc + nms corresponds to using RANEt with IOU loss, discriminant scoring and non maximal suppression. It also started over-fitting after 1

Table 3.2. Results of modification techniques in Val dataset

Model	J&F Mean	J Mean	J Decay	F Mean	F Decay
RANet	0.694	0.66	0.22	0.719	0.22
RANet + IOU 2 epoch M_{t-1}	0.718	0.692	0.20	0.74	0.21
RANet IOU + disc 1 epoch	0.719	0.69	0.20	0.74	0.21
RANet IOU + disc + nms 1 epoch	0.719	0.695	0.19	0.74	0.21

Table 3.3. Results of modification techniques in DAVIS 2017 test dev

Model	J&F Mean score
RANet	0.553
RANet + IOU 2 epoch M_{t-1}	0.596
RANet IOU + disc 1 epoch	0.591
RANet IOU + disc + nms 1 epoch	0.584

epoch. Separate applications of these modifications were not run.

As can be seen from the table, IOU loss improved the performance by 4.3% in test dataset. But other modifications didn't bring much change in val dataset performance and decreased performance in test dataset. Two possible reasons are that the modification is causing the model to learn quickly and overfit and that the modifications are not compatible with the RANet network architecture. As shown in section 3.2.1, the network is not reliant on sharp correlation maps and is instead using the maps to modify the previous frame mask. The learnt decoder on correlation maps also sharpens the correlation maps using some idea of object shapes.

3.3 RANet on private video annotation and tracking

One of the motivations for exploring RANet was to assist in generating annotations and track objects of new categories. To test the effectiveness of RANet, we picked the driving video, whose images are shown in Figure2.3. Unlike prediction models, tracking algorithm requires an initial annotation so that the network knows which objects to track. For this, we used the output annotations of HRNet on the the first frame. The images were extracted every 30 frames on a

30 FPS video. Thus there is a gap of 1 second between each frame and fast moving objects as well. The results are shown in figures 3.4 and 3.4. The domain adapted HRNet was used for the first frame annotation. The categories identified were road (darkest), vegetation, sky and car (brightest). This was provided as masks for RANet to track, with all pixels belonging to one category forming mask for that ”object”. Since there was no background, a softmax over all present classes were used to label the pixel. Thus RANet was able to identify new cars coming and going in the images. This can be seen in the figures where different cars pass by each second. The ability to identify new cars from the car in initial frame can be attributed to the feature extraction ability of RANet encoder. Since every pixel had a label in HRNet annotation and no ”background”, a softmax was used over all classes present in the first frame, to decide the label of a pixel when tracking with RANet. Occasionally the RANet encoder is able to better identify cars than HRNet as seen in Frame at sec 2 and frame at sec 6 in figure 3.4. But at the same time HRNet identifies the new car better in Frame at sec 3 in the same figure. As a limitation, the RANet cannot identify traffic signals and poles which were not annotated by HRNet in frame 1. A classification network is not constrained by the introduction of new object classes in subsequent frames, as long as the network was trained in some dataset containing these object classes.

To compare the effectiveness of tracking a new object class, the hanging flower in Frame 1 was picked up as test subject. The flower was occasionally labelled as road or car by the HRNet. We generated the annotation with a simple bounding box converted to dense binary mask as shown in Figure 3.6. Despite starting with a bounding box mask, the RANet is able to track the rough boundaries of the flower, identify when it disappears and reappears in the video.

3.4 Conclusion

We made a significant improvement in RANet performance by fine tuning with better loss functions. RANet also showed itself to be good at using propagating an initial annotation

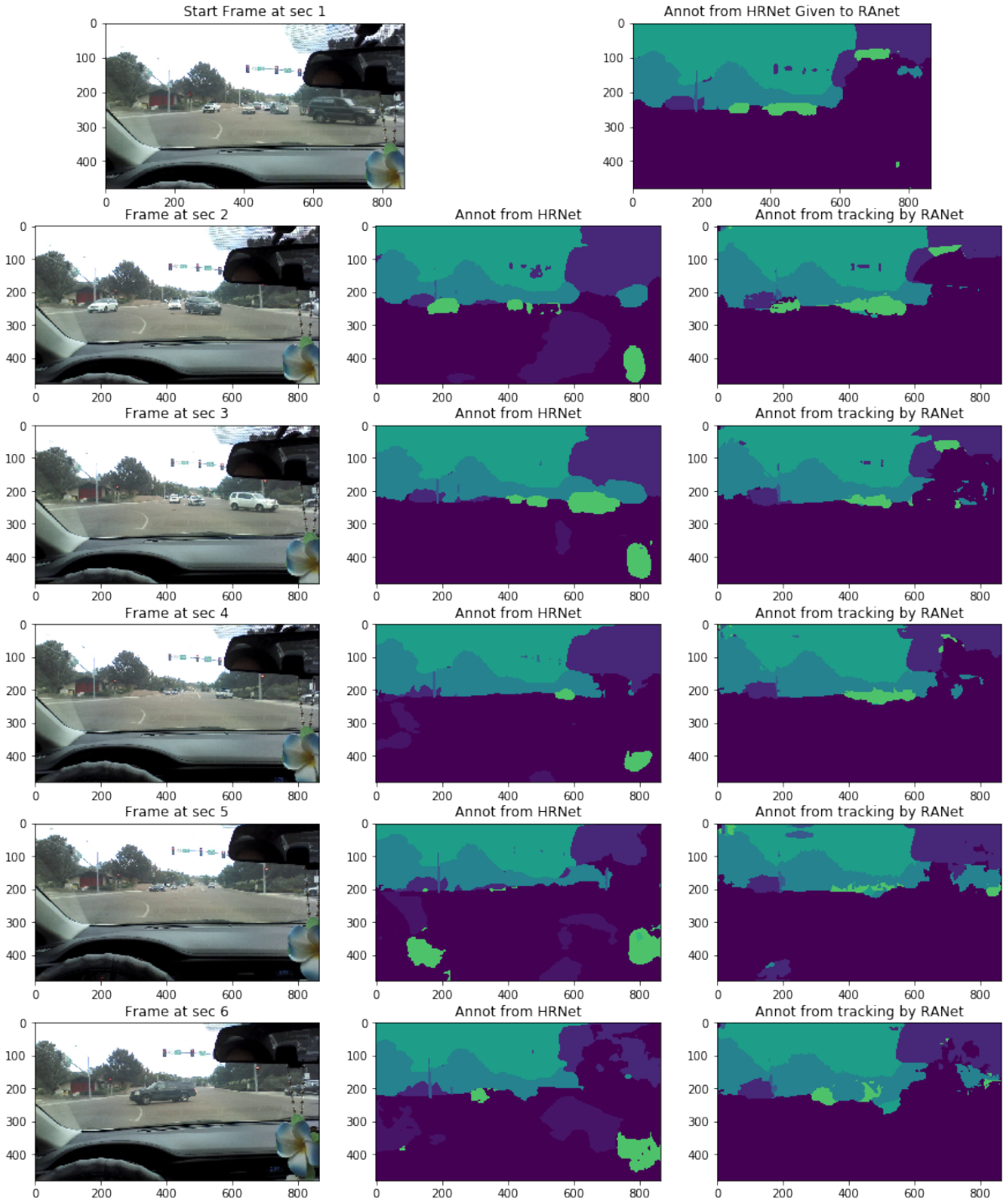

Figure 3.4. First frame annotated by domain adapted HRNet. Subsequent annotations are results of tracking using modified RANet.

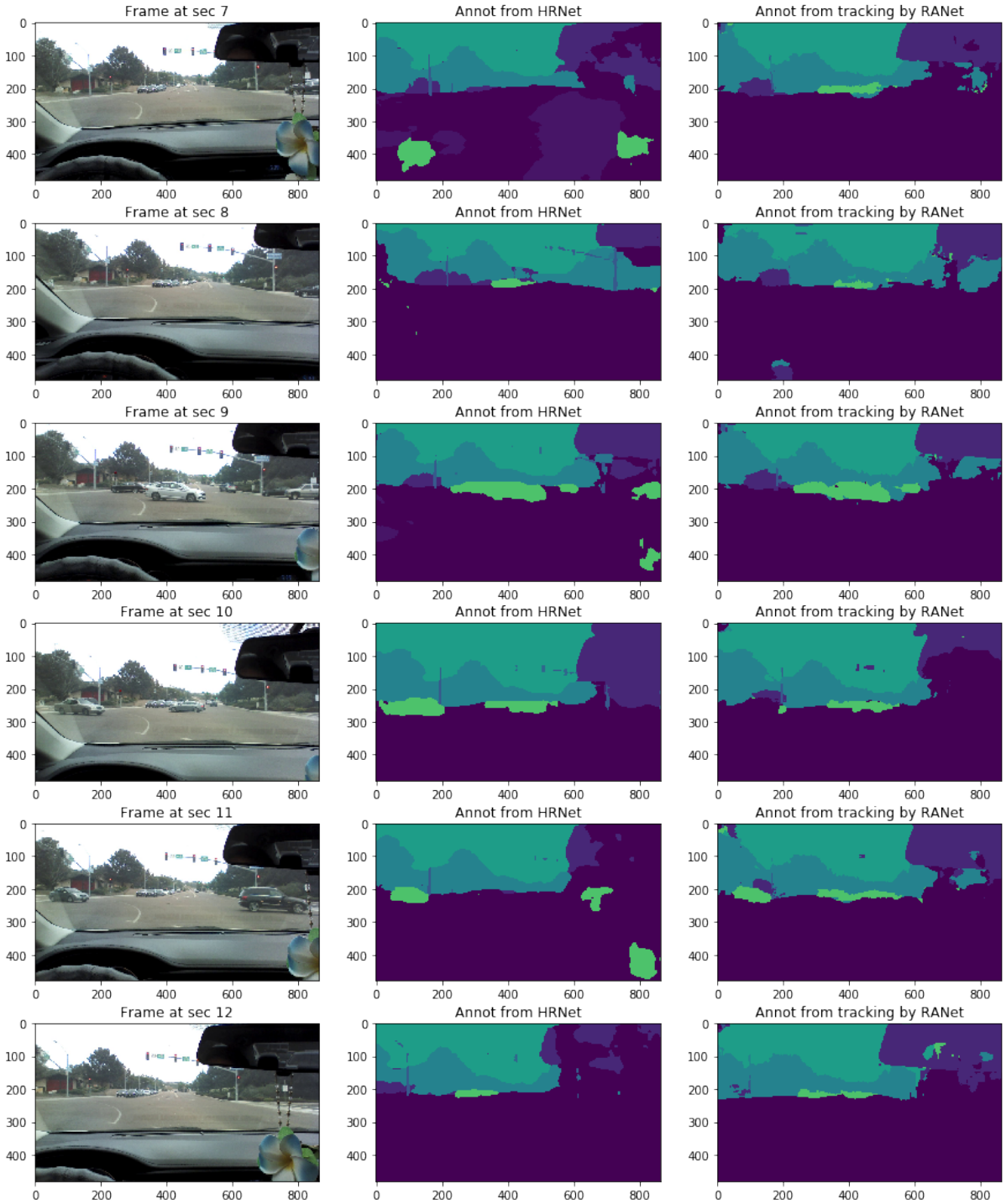

Figure 3.5. Continuation from figure 3.4.

across a video. From the initial annotation of cars by HRNet in the first frame, the RANet could identify different cars in the subsequent frames, with performance on par with domain adapted

Figure 3.6. Tracking of a new class of object. A crude mask was generated on Frame 1 and given to RANet to track. This object was identified as car or road by HRNet (both non-adapted and adapted).

HRNet. It also proved itself useful in the niche case of tracking a new object class, from a single annotation instance. A good strategy for annotating novel videos would be to use HRNet or similar detectors to extract an initial annotation, identify the new object classes needing tracking and using RANet to propagate those annotations further.

There were also certain limitations observed, with the network not recognizing some cars

or vegetation and instead identifying them as road. Also the tracking of a new object class has only been tried on a single video and the possibility of using an annotated frame in one video to track the same object in a separate video (with different background) has yet to be implemented. This needs to be further explored along with making the feature extractor more domain invariant.

Chapter 4

Unsupervised Learning for Object Segmentation

The supervised VOS method worked well for tracking new object classes as shown in previous chapter. This made us interested in networks which learn good feature extractors. The recent papers, CorrFlow[23] and Memory Augmented Self supervised Tracker (MAST)[54] proposed unsupervised training methods on raw videos and worked based on extracting good distinguishing feature vectors for pixels. Moreover the MAST claims a high performance comparable to supervised methods like RANet. A working MAST method might generalize better on new objects and can be fine-tuned on new videos directly using unsupervised training method. This made us interested in exploring the unsupervised method and try reproducing it. Since the code for MAST was not publicly available, we referred the code of Corrflow and implemented MAST from scratch.

4.1 Method

The MAST works on the idea of non-local means filters, by identifying similar pixels and reconstruction using them. The input to the model are a set of past frames (called reference frames), the current frame(called query frame) and the annotations associated with the past frames. The frames are in LAB color space and the annotations associated with them are one of the A or B channel in LAB space. The channel selected as annotation is dropped out in input

frames. The aim of the model is to reconstruct the annotation associated with the query frame using the reference frames and their annotations. This query annotation is constructed pixel by pixel. For a query pixel in query frame, similar pixels are found in reference frames. Theses pixels and their corresponding annotations form the key pixels and key values associated with that particular query pixel. The MAST module takes in the first frame, 5^{th} frame, t-5, t-3 and t-1 frames as reference frames when the query is frame t in the video.

For a particular query pixel, the search for key pixels in the past 5 frames is conducted in two stages: Region of Interest (ROI) identification in each reference frame and selection of pixels from the ROI with their importance. These processes can be described independently for each reference frame.

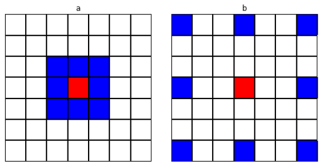

Figure 4.1. Using dilation for sparse region of interest searching. The Red box is the coordinate for the query pixel, while the blue boxes are the only pixels considered for correlation computation. The window size is 3x3 with (a) having dilation of 1 and (b) having dilation of 3.

Step 1: For each pixel in the query frame,the nearby pixels are sampled in the reference frame. For a query pixel at coordinates (i,j), the nearby pixels are the pixels in a window of size (w $\times$ w) around coordinate (i,j) in the reference frame. As the temporal gap between query and reference frame increases, the window is enlarged by use of dilation. The dilation factor used was: ceil($\frac{t}{15}$) where t is the difference in frame number. The amount of computation remains same while a larger field of view is searched, due to dilation. As an example consider a window size of 13$\times$13. For a frame with temporal gap of 1, the dilation factor is 1 and all pixels in a 13$\times$13 window would be sampled, giving 169 points for computing correlation. For a frame with temporal gap of 35, the dilation factor is 3. Thus a 39$\times$39 window centered around the query

pixel would be considered, but not every pixel in that window would be used for computing correlation with feature vector of query pixel. Instead, only 13*13 points would be considered from this 39×39 window, with a gap of 3 between the pixels. This is similar to the points picked by dilated convolution. Figure 4.1 gives an example for a 3x3 window with dilation 1 and 3. Even-though the window considered for ROI increased in Figure 4.1 (b), the points picked for correlation computation are still only 9 pixels. Based on the correlation score of candidate pixels w.r.t feature vector of query pixel, the center coordinate of region of interest for query pixel is computed in the reference frame using the below equation. For reference frame r, query frame q, for a pixel in q with coordinate (x_p, y_p):

$$C^{q,r}_{x_p,y_p} = \frac{\sum_{h\in\{-w,w\}} \sum_{l\in\{-w,w\}} exp(< v^{q}_{(x_p,y_p)}, v^{r}_{(x_{p+h*d},y_{p+l*d})} >)(x_{p+h*d}, y_{p+l*d})}{\sum_{h\in\{-w,w\}} \sum_{l\in\{-w,w\}} exp(< v^{q}_{(x_p,y_p)}, v^{r}_{(x_{p+h*d},y_{p+l*d})} >)} \tag{4.1}$$

Where <> denotes correlation and $v^{s}_{(i,j)}$ denotes feature vector for pixel (i,j) in frame s. d here is the dilation factor. In other words, softmax is used to predict the center coordinates of region of interest(ROI) for each pixel. From each frame, all the pixels in a w'xw' window around the ROI center as key pixels for that query pixel. Thus each reference frame provides w'^{2} number of key pixels for a particular query pixel. The values associated with the key pixels are called key values and they are taken from the value of pixel at same coordinate as key pixel, but in the annotation frame.

Step 2: Once the key pixels and their corresponding annotation values (key values) are found in all reference frames, they are all pooled together, giving 5 * w'^{2} number of key pixels and key values associated with the key pixels. The reconstruction is done similar to constructing the ROI location, but now using all key pixels from all frames. Let i be the query pixel and J corresponds to set of key pixels associated with it. Let $feat_k$ represents feature vector associated with pixel k and y_k represents the value associated with pixel k which might be in the query

frame or one of the reference frame. The equations for reconstructing the value of pixel i is[54]:

$$\hat{y}_i = \sum_{j \in J} A_{i,j} v_j \tag{4.2}$$

$$A_{i,j} = \frac{exp(< feat_i, feat_j >)}{\sum | j' \in J exp(< feat_i, feat_{j'} >)} \tag{4.3}$$

Where $\hat{y}_i$ is the predicted value for pixel i while y_i is the true value. The window size used by us for predicting center coordinates was 13x13 and the window size for selecting key pixels from ROI was 5x5. The window size value was chosen based on window size used for ROI prediction in Corrflow[23] for a similar operation while the window size used for selecting key pixel was based on computation and memory constrains.

In case of regression loss, a Huber loss is used with y_i being the target value. The y_i is either the A or B channel in the LAB channel space. Thus for regression loss, the input is image from LAB space and target is one of the channels in A-B. The channel picked as target is zeroed out in the input. Even though the original MAST paper proposes using regression loss, the model can also be trained with classification loss similar to Corrflow. For classification loss, each pixel must be associated with a particular class. y_i in this case becomes a binary value 0 or 1 depending on whether the pixel i belongs to the current reconstructed class (or object being tracked). The class for a pixel can be generated from supervision data or in an unsupervised way[23, 49]. In Corrflow[23], the authors proposed generating 16 cluster centers using K means on the whole dataset and then assigning each pixel to one of the clusters. RGB pixel values of images in the dataset were considered as x-y-z 3D coordinates and the clusters were generated using L2 distance. For classification training, the input would be RGB image and the annotations would be generated using clustering in LAB space. This method was followed during implementation. To avoid learning pixel values as feature vectors, both Corrflow and MAST proposes channel dropouts.

4.1.1 Loss functions for training

Common definitions: x_i refers to feature vector for pixel i. y_i refers to true annotation for pixel i, which could be one of the A/B channels in LAB space (for regression) or the cluster label/true annotation for classification.

Classification Loss

Consider query pixel i in frame t with class y_i. Using cross entropy loss for pixel i

$$L(i) = \sum y_i log(p_i^{y_i}), where \tag{4.4}$$

$$p_i^{y_i} = \frac{\sum_{j \in Key(x_i), y_j == y_i} e^{c_{ij}}}{\sum_{j \in key(x_i)} e^{c_{ij}}} \tag{4.5}$$

Here c_{ij} is the correlation between feature vectors of pixel i and pixel j. Note we are not considering frames of i and j here as that is not important once the key pixels are selected.

Also lets define y_i^{pred} as:

$$y_i^{pred} = \sum_{j \in Key(i)} p_{ij} y_j \tag{4.6}$$

$$c_{ij} = < x_i, x_j > \tag{4.7}$$

To simplify, we can consider the class y to be 0 or 1. With this, we can write the cross entropy loss as:

$$L(i) = -\sum y_i log(y_i^{pred}) - (1 - y_i) log(1 - y_i^{pred}), where \tag{4.8}$$

$$y_i^{pred} = \frac{\sum_{j \in Key(x_i)} e^{c_{ij}} y_j}{\sum_{j \in key(x_i)} e^{c_{ij}}} \tag{4.9}$$

$$y_i^{pred} i = \sum_j p_{ij} y_j where \tag{4.10}$$

$$p_{ij} = \frac{e^{c_{ij}}}{\sum_{j \in key(x_i)} e^{c_{ij}}} \tag{4.11}$$

Regression Loss

Consider pixel i in frame t with true value y_i. Even though the loss used is Huber loss, L2 loss can be considered for understanding the gradients and how back propagation helps. Using L2 loss for pixel i

$$L(i) = (y_i - \sum_{j \in Key(i)} p_{ij} y_j)^2, where \tag{4.12}$$

$$p_{ij} = \frac{e^{c_{ij}}}{\sum_{j' \in key(x_i)} e^{c_{ij'}}} \tag{4.13}$$

4.1.2 Gradient

Thus the loss for both can be written in terms of p_{ij} where i is the query pixel we want to reconstruct and j is one of the Key pixels for i. Before proceeding to gradient in case of both loss functions, lets compute gradient of p_{ij} w.r.t input x_k.
if j=k:

$$\frac{\partial p_{ij}}{\partial x_k} = p_{ik}(1 - p_{ik}) x_i \tag{4.14}$$

if j$\neq$ k:

$$\frac{\partial p_{ij}}{\partial x_k} = -p_{ij} p_{ik} x_i \tag{4.15}$$

This will be useful for deriving equations of gradients for both loss functions and comparing how the gradients differ for both loss functions.

Gradient for L2 loss

The x denote the feature vector associated with pixels while y denote the annotation associated with the pixel. For key pixel feature vector x_k, the gradient to reduce loss on pixel i

will be :

$$-\frac{\partial L(i)}{\partial x_k} = -\frac{\partial (y_i - \sum_{j \in Key(i)} p_{ij} y_j)^2}{\partial x_k} \tag{4.16}$$

$$= -(y_i^{pred} - y_i) \frac{\partial \sum_{j \in Key(i)} p_{ij} y_j}{\partial x_k} \tag{4.17}$$

$$= -(y_i^{pred} - y_i) x_i [p_{ik}(1 - p_{ik}) y_k - \sum_{j \in Key(i), j \neq k} p_{ij} p_{ik} y_j] \tag{4.18}$$

$$= -(y_i^{pred} - y_i) x_i p_{ik} [y_k - p_{ik} y_k - \sum_{j \in Key(i), j \neq k} p_{ij} y_j] \tag{4.19}$$

$$= -(y_i^{pred} - y_i) x_i p_{ik} (y_k - \sum_{j \in Key(i)} p_{ij} y_j) \tag{4.20}$$

$$= (y_i - y_i^{pred})(y_k - y_i^{pred}) p_{ik} x_i \tag{4.21}$$

When training with L2 loss, the annotations(values) y are scalar pixel value themselves from the A or B channel in the LAB color space. If true query value y_i and key value y_k are on the same side of the predicted query value y_i^{pred}, the gradient shifts the key pixel feature vector x_k towards query pixel feature vector x_i. If they are opposite, the gradient moves x_k away from x_i.

Gradient for Classification loss

For key pixel x_k, the gradient to reduce loss on pixel i will be :

$$-\frac{\partial L(i)}{\partial x_k} = \frac{\partial \sum y_i log(y_i^{pred}) + (1 - y_i) log(1 - y_i^{pred})}{\partial x_k} \tag{4.22}$$

If y_i = 1:

$$-\frac{\partial L(i)}{\partial x_k} = \frac{\partial log(y_i^{pred})}{\partial x_k} \tag{4.23}$$

$$= \frac{y_i}{y_i^{pred}} \frac{\partial \sum_{j \in Key(i)} p_{ij} y_j}{\partial x_k} \tag{4.24}$$

$$= \frac{y_i}{y_i^{pred}} (y_k - y_i^{pred}) p_{ik} x_i \tag{4.25}$$

$$= \frac{(y_k - y_i^{pred}) p_{ik} x_i}{y_i^{pred}} \tag{4.26}$$

If $y_i = 0$:

$$-\frac{\partial L(i)}{\partial x_k} = \frac{\partial \sum log(1 - y_i^{pred})}{\partial x_k} \tag{4.27}$$

$$= \frac{1}{1 - y_i^{pred}} \frac{\partial - y_i^{pred})}{\partial x_k} \tag{4.28}$$

$$= \frac{(y_i^{pred} - y_k) p_{ik} x_i}{1 - y_i^{pred}} \tag{4.29}$$

Thus, the gradient pushes x_k towards x_i if their labels match and pushes away if labels don't match.

4.2 Experimentation and Results

To compare between the the 3 ways of training the feature extractor, we trained three networks, starting from same initial model and using same data augmentation and learning rates. Since the 3 MAST models and Corrflow use the same feature extractor, the MAST models were initialized with Corrflow pretrained model on OxuVa dataset. This was done to reduce amount of training required and whether it was a good idea will be discussed later in this section. The learning rate used was 1e-04 and data augmentations only included random vertical/horizontal flips and 90 degree rotations. All models were trained for 30 epochs. Table 4.1 shows the results of training. As expected, the model with true supervision performs best. One interesting result was that Unsupervised Cls (clustering in LAB space and using that as annotations) performed better than L2 loss. This is in contradiction to the original MAST paper. One reason why this happened might be because the Corrflow pretrained model ran on RGB space and thus fine tuning was not enough to adjust to LAB space. This is supported by the results of Unsupervised $L2^{RGB}$, where giving the input in RGB space produced better results with an L2 loss trained

network. Here, the same model as Unsupervised L2 was used (with training in LAB space), but during testing, RGB input images were given, instead of LAB color space.

Table 4.1. Comparison of different training losses. For normalized models, a temperature of 0.1 was used in the softmax to scale up the cosine distances

Model	J mean unnorm	F mean unnorm	J mean norm	F mean norm
Supervised	50.2	51.9	49.8	52.3
Unsupervised Cls	43.4	46.4	41.2	42.3
Unsupervised L2	35.7	38.4	30.5	26.9
Unsupervised $L2^{RGB}$	37.5	40.0	-	-

Another dimension we explored was whether to use correlation or cosine distances in equations 3.1 and 3.3 . In table 4.1, the results are shown by columns with ”norm”. Overall, the values are close but lesser than using direct correlation values. Using cosine distances required two additional hyper parameters: temperature on softmax operations for ROI location prediction and for key score computation in equations 4.1 and 4.3 repectively. This is because, cosine distances are in the range of [-1,1] and thus a highly correlated pixel can get suppressed by the presence of large number of uncorrelated pixels. This can be seen in figure 4.2. The temperature factor in softmax sharpens the weighted sum and lets the computation focus on regions with higher correlation. The query frame was 20^{th} frame of video and the model is trying to estimate the ROI in frame 1. The cosine distance value for the selected query pixel, blue box in (a), is an ideal example: high activation near the place where the true head of the person is present and little to no activation everywhere else. Still, the ROI estimated with regular softmax is far from the region of activations. We needed a scaling up of activations to correctly compute the ROI location. This is not needed if we use correlation values as its not constrained to a range. As seen from the table 4.1, having correlation instead of cosine distances is giving better results. It might be due to the model having flexibility to scale up or down the feature vectors. Using cosine distances is equivalent to having normalized feature vectors as output from the feature extractor.

Overall, the results of the original MAST paper were not achieved by either supervised or unsupervised training. The results are shown in Table: 4.2. Although a variety of factors in

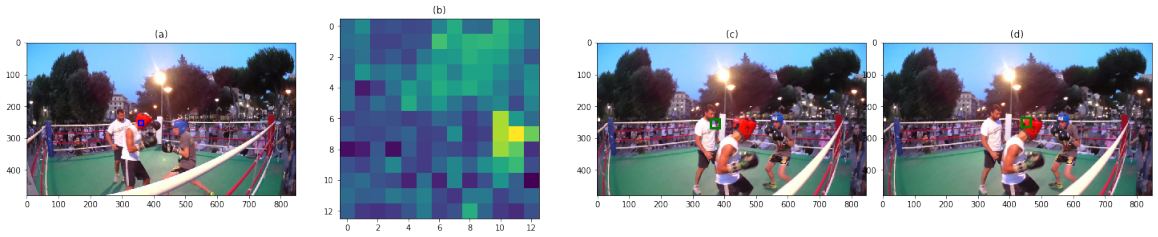

Figure 4.2. ROI prediction using cosine distances and softmax. a) The Query frame with blue box as query. b) The 13x13 window cosine distance values corresponding to the query pixel in first frame. c) The estimated ROI using softmax. d) The estimated ROI using softmax with 0.1 temperature.

the trained models differ from the original MAST model, using a Corrflow pretrained model might be the biggest factor. Although 30 epochs is a bit low and training losses were still sharply reducing, DAVIS 2017 train set is small and the validation dataset performance was saturating by 30 epochs.

Table 4.2. Comparison of reproduced model with original models

Model	J mean	F mean
Corrflow Best Results	48.4	52.2
Mast paper Results	63.9	64.9
Corrflow pretrain on Mast	16.8	11.8
Superv Mast best result	50.2	51.9
Unsuperv Mast best result	45.7	47.8

Another point of difference is in how the feature vectors are extracted for the pixels sampled in dilated ROI search window. With high dilation, important features can be missed in the gap between dilated samples. This can be seen in figure 4.3. The white part of the beak is present in between some sampled windows. In the current implementation, only the feature vectors of sampled pixels are picked, which can hamper training and performance. The original paper proposes using bi-linear sampling similar to Spatial transformer networks[18], although its not clear what they meant by that. Having the feature vector of green sampled points as the average feature vector within red boxes in figure4.3(c) might give better results.

The prediction capability of trained MAST model (unsupervised classification) was visualized by replacing the pseudo label of first frame with true annotation. The results can be

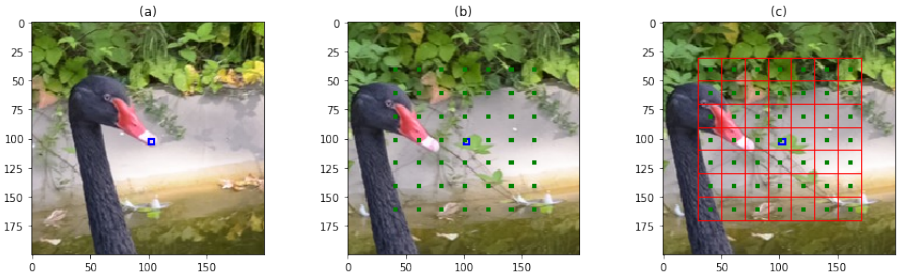

Figure 4.3. Area scanned by dilated window. a) The Query frame with query pixel marked in blue box. b) The 7x7 window dilated by a rate of 20. The sampled pixels are showing in green boxes c) The area represented by each dilated window sample (red box)

seen in Figure 4.4. The predictions for initial frame are good, with performance decaying as we progress through the frames. Sometimes the prediction improves in later frames like the predicted annotation of human in frame 51 vs 61, where the network is able to detect the change in pose, and the predicted outline of the person is much smoother in the later frame. At the same time, the network missed tracking movement of small objects with high similarity to background like the cycle tyre. This shows that the network feature extractor is still not perfect and might still be correlated with the color information of the pixel.

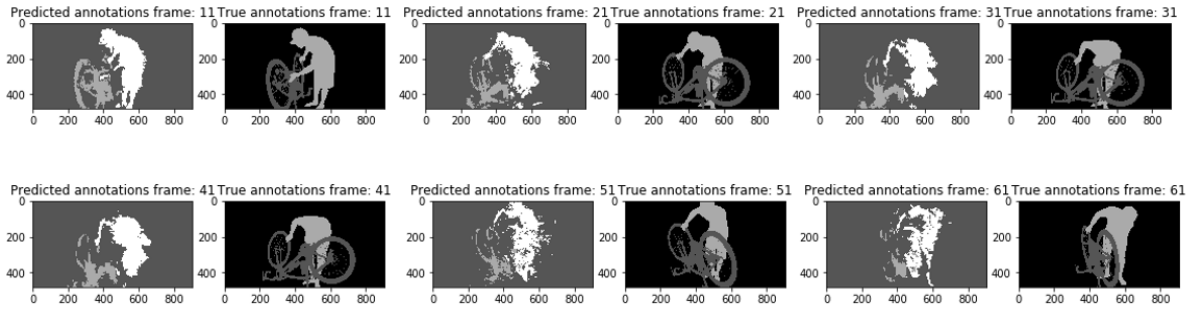

Figure 4.4. Sample annotation prediction by a trained unsupervised MAST model on a video in validation dataset. Only the first frame annotation was provided.

4.2.1 List of differences in Corrflow/MAST and training done:

- The size of ROI search window. The size in Corrflow was used and might differ from the one used in MAST.

- ROI size. A size of 5x5 was used and might differ from the original MAST.
- Using bi-linear sampling with dilation in ROI search window.
- The reference frame selection: MAST used 0, 5, t-5, t-3 and t-1 frames as reference frames for frame t while the current implementation used 0, 5, t-5 and t-1 frames. Even though one extra frame may not look that impactful, the short term memory outweighs the long term memory by 3:2 in original MAST because of this. This might have an impact as the authors mention using long term memory brings an improvement in the order of 4%, with the short term memory having the most impact.
- The Corrflow model predicts at 1/4 the image size and this prediction is bi-linear up-sampled to image resolution before applying loss. This might provide a better loss than down sampling the target prediction to lower resolution.

As an experiment, the MAST model was trained again with same hyper-parameters, but with changing ROI window size from 5x5 to 7x7 and then including some of the differences listed above. The results are shown in table 4.3. The training method was only unsupervised classification loss. In Up sampling with short term memory model, the same changes as the Unsuperv Classification initial model was made, but during training, only the short term memory frames were included (t-5 and t-1). The Up-sampling model result shows that the bi-linear sampling plays a key role in training and improved the performance by a significant amount. While the second experiment with using only short term memory showed that the current training is not able to make the network rely much on long term memory. Initializing feature extractor with a Corrflow pretrained model might be the most significant factor in limiting the training performance.

Table 4.3. Comparison of initial trained model with fixing some of the stated differences. Up sampling model indicates using bi-linear sampling with dilation in ROI search window and up sampling prediction to original image size before applying loss.

Model	J mean	F mean
Unsuperv Classification initial	45.7	47.8
Up sampling	48.9	51.0
Up sampling with short term memory	48.2	51.3

4.3 Conclusion

We attempted to reproduce the results of MAST[54]. To reduce training time, the pretrained OxuVa model of Corrflow was used as initialization for feature extractor of MAST model and the model was trained only on DAVIS training dataset. The initial results are not good, with performance peaking before even reaching the performance of Corrflow. Improving the code for memory usage and fixing some of the identified differences with the original paper improved the performance of the model, with performance of unsupervised classification training now reaching the performance of Corrflow model. Still it is not close to the performance stated in the original paper. Training from scratch in a bigger video dataset like OxuVa might be the key in reaching the benchmark performance.

Chapter 5

Conclusion

Annotation of novel videos motivated the exploration of object segmentation and domain adaptation networks. We picked a state of the art object segmentation network and domain adapted it to the novel video for identifying objects. The domain adaptation improved the quality of predicted annotation. For assisting with the annotation of new object classes, we explored video object segmentation (VOS) tracking. RANet, a state of the art network in single object tracking was picked. Out of academic curiosity, we attempted several modifications to improve its performance in multi object tracking . One modification, using soft IOU loss, improved the performance on DAVIS 2017 test dataset by 4%. The network was also used as a tracker for new object class in the private video and it showed promising results. The effectiveness of feature extractor of the network was shown, when it could track new cars from the annotation of old cars despite the network never having been trained to identify car as a class of objects. The importance of feature extracted and usefulness of object tracking led us to explore the unsupervised video object segmentation tracking. We selected the state of the art unsupervised network, MAST, and a partially successful model was implemented. Currently, the model hasn't achieved the performance shown in the paper. We identified actual and possible differences with the implementation in paper which could bridge the gap in performance.

There are currently better domain adaptation methods than DANN used for HRNet like ADDA[47] and UNIT[28]. Also new works exploring adaptation for new object categories are

coming up. These methods have good potential to make object detectors and segmentation networks work for object identification in new videos without going through time consuming manual annotations. With the potential of VOS tracking for assisting in annotation shown, getting a working unsupervised method could go a long way in automatic annotation. The existing implementation of MAST needs to be improved to reach the original paper's results. Exploring new ways to improve the unsupervised training method is another avenue. Currently the method relies on per pixel reconstruction, which may not help the network learn the structure of objects in the video. A way of area by area reconstruction (box reconstruction) could help network learn this. This also introduces new problems such as the problem of orientation adjustment between matched areas, optimal size of the areas etc. Having a working unsupervised VOS tracker network can simplify the annotation generation as well as provide a very good feature extractor which could be used in other computer vision areas as well.

Bibliography

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

[2] L. Bao, B. Wu, and W. Liu. Cnn in mrf: Video object segmentation via inference in a cnn-based higher-order spatio-temporal mrf. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5977–5986, 2018.

[3] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke. Using simulation and domain adaptation to improve efficiency of deep robotic grasping, 2017, arXiv:1709.07857.

[4] A. Bulling and K. Kunze. Eyewear computers for human-computer interaction. *ACM Interactions*, 23(3):70–73, 2016. doi:10.1145/2912886.

[5] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. V. Gool. One-shot video object segmentation, 2016, arXiv:1611.05198.

[6] S. Caelles, J. Pont-Tuset, F. Perazzi, A. Montes, K.-K. Maninis, and L. Van Gool. The 2019 davis challenge on vos: Unsupervised multi-object segmentation. *arXiv:1905.00737*, 2019.

[7] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. *CoRR*, abs/1504.06852, 2015, 1504.06852. URL http://arxiv.org/abs/1504.06852.

[9] D. Fourure, R. Emonet, É. Fromont, D. Muselet, A. Trémeau, and C. Wolf. Residual conv-deconv grid network for semantic segmentation. *ArXiv*, abs/1707.07958, 2017.

[10] W. Fuhl, T. Santini, G. Kasneci, and E. Kasneci. Pupilnet: Convolutional neural networks for robust pupil detection. arxiv, 2016. *arXiv preprint arXiv:1601.04902*.

[11] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030, Jan. 2016.

[12] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, page 580–587, USA, 2014. IEEE Computer Society. doi:10.1109/CVPR.2014.81.

[14] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013, 1311.2524. URL http://arxiv.org/abs/1311.2524.

[15] T. Han, W. Xie, and A. Zisserman. Video representation learning by dense predictive coding. *CoRR*, abs/1909.04656, 2019, 1909.04656. URL http://arxiv.org/abs/1909.04656.

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[17] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *CoRR*, abs/1612.01925, 2016, 1612.01925. URL http://arxiv.org/abs/1612.01925.

[18] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks, 2015, arXiv:1506.02025.

[19] T. Kalluri, G. Varma, M. Chandraker, and C. V. Jawahar. Universal semi-supervised semantic segmentation. 2018, arXiv:1811.10323.

[20] M. Kassner, W. Patera, and A. Bulling. Pupil: an open source platform for pervasive eye tracking and mobile gaze-based interaction. In *Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing: Adjunct publication*, pages 1151–1160. ACM, 2014.

[21] A. Khoreva, R. Benenson, E. Ilg, T. Brox, and B. Schiele. Lucid data dreaming for object tracking. *CoRR*, abs/1703.09554, 2017, 1703.09554. URL http://arxiv.org/abs/1703.09554.

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

[23] Z. Lai and W. Xie. Self-supervised learning for video correspondence flow. *CoRR*, abs/1905.00875, 2019, 1905.00875. URL http://arxiv.org/abs/1905.00875.

[24] X. Li and C. C. Loy. Video object segmentation with joint re-identification and attention-aware mask propagation. *CoRR*, abs/1803.04242, 2018, 1803.04242. URL http://arxiv.org/abs/1803.04242.

[25] X. Li, Y. Qi, Z. Wang, K. Chen, Z. Liu, J. Shi, P. Luo, X. Tang, and C. C. Loy. Video object segmentation with re-identification, 2017, arXiv:1708.00197.

[26] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014, 1405.0312. URL http://arxiv.org/abs/1405.0312.

[27] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection, 2017, arXiv:1708.02002.

[28] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. 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 700–708. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/6672-unsupervised-image-to-image-translation-networks.pdf.

[29] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016. doi:10.1007/978-3-319-46448-0_2.

[30] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 97–105. JMLR.org, 2015.

[31] J. Luiten, P. Voigtlaender, and B. Leibe. Premvos: Proposal-generation, refinement and merging for video object segmentation. *CoRR*, abs/1807.09190, 2018, 1807.09190. URL http://arxiv.org/abs/1807.09190.

[32] J. Luiten, I. E. Zulfikar, and B. Leibe. Unovost: Unsupervised offline video object segmentation and tracking, 2020, arXiv:2001.05425.

[33] K.-K. Maninis, S. Caelles, Y. Chen, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. V. Gool. Video object segmentation without temporal information, 2017, arXiv:1709.06031.

[34] F. Milletari, N. Navab, and S.-A. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation, 2016, arXiv:1606.04797.

[35] G. Máttyus, W. Luo, and R. Urtasun. Deeproadmapper: Extracting road topology from aerial images. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3458–3466, Oct 2017. doi:10.1109/ICCV.2017.372.

[36] G. Nagendar, D. Singh, V. N. Balasubramanian, and C. V. Jawahar. Neuro-iou: Learning a surrogate loss for semantic segmentation. In *BMVC*, 2018.

[37] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation, 2016, arXiv:1603.06937.

[38] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*, 2016.

[39] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. Full-resolution residual networks for semantic segmentation in street scenes, 2016, arXiv:1611.08323.

[40] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017.

[41] M. A. Rahman and Y. Wang. Optimizing intersection-over-union in deep neural networks for image segmentation. In G. Bebis, R. Boyle, B. Parvin, D. Koracin, F. Porikli, S. Skaff, A. Entezari, J. Min, D. Iwai, A. Sadagic, C. Scheidegger, and T. Isenberg, editors, *Advances in Visual Computing*, pages 234–244, Cham, 2016. Springer International Publishing.

[42] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection, 2015, arXiv:1506.02640.

[43] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016, 1612.08242. URL http://arxiv.org/abs/1612.08242.

[44] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[45] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015, arXiv:1505.04597.

[46] K. Sun, Y. Zhao, B. Jiang, T. Cheng, B. Xiao, D. Liu, Y. Mu, X. Wang, W. Liu, and J. Wang. High-resolution representations for labeling pixels and regions, 2019, arXiv:1904.04514.

[47] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation, 2017, arXiv:1702.05464.

[48] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016, 1607.08022. URL http://arxiv.org/abs/1607.08022.

[49] C. Vondrick, A. Shrivastava, A. Fathi, S. Guadarrama, and K. Murphy. Tracking emerges by colorizing videos, 2018, arXiv:1806.09594.

[50] M. Wang, W. Deng, J. Hu, J. Peng, X. Tao, and Y. Huang. Racial faces in-the-wild: Reducing racial bias by deep unsupervised domain adaptation. *CoRR*, abs/1812.00194, 2018, 1812.00194. URL http://arxiv.org/abs/1812.00194.

[51] Z. Wang, J. Xu, L. Liu, F. Zhu, and L. Shao. Ranet: Ranking attention network for fast video object segmentation, 2019, arXiv:1908.06647.

[52] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492. IEEE, 2010.

[53] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. Huang. Unitbox. *Proceedings of the 2016 ACM on Multimedia Conference - MM '16*, 2016. doi:10.1145/2964284.2967274.

[54] W. X. Zihang Lai, Erika Lu. Mast: A memory-augmented self-supervised tracker. 2020, 2002.07793. URL https://arxiv.org/abs/2002.07793.