Multi-task Learning for Visual Perception in Automated Driving

by

Sumanth Chennupati

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical, Electronics, and Computer Engineering)
in the University of Michigan-Dearborn
2021

Doctoral Committee:

Associate Professor Samir Rawashdeh, Chair
Assistant Professor Abdallah Chehade
Assistant Professor Alireza Mohammadi
Associate Professor Paul Watta

Sumanth Chennupati

schenn@umich.edu

ORCID iD: 0000-0002-3382-540X

DEDICATION

I want to dedicate this dissertation

to my wife, who somehow managed to be nothing but supportive,

to my daughter,

to my parents,

to my in-laws,

to my family,

to my teachers,

to my friends,

to my colleagues,

and finally to whoever that might find this dissertation exciting and useful.

ACKNOWLEDGEMENTS

I feel fortunate enough to have an opportunity to pursue research while working as a full-time employee. I want to thank everyone for their help, support, and encouragement in making this journey possible, challenging, and fun.

Firstly, I would like to thank Dr. Raymond Ptucha, who inspired me to achieve what seemed impossible. I am indebted to Davian Larente for his trust and encouragement. Without your support and initiatives, this journey would not have been possible. I would also like to thank Vira Mourovapin, Dr. Mark Beeler, Roland Richardson, and Dr. Lin Chen for being flexible and supportive. I sincerely thank Mo Poorsartep, who introduced me to my advisor Dr. Samir Rawashdeh.

I want to thank Dr. Samir Rawashdeh for his excellent guidance and support. Your substantial advising and feedback have helped me in improving the quality of my research. I sincerely thank my dissertation committee, In alphabetical order: Dr. Abdallah Chehade, Dr. Alireza Mohammadi, and Dr. Paul Watta, for their valuable time and advice. Thank you, Amanda Donovan, Debi Butler, for your administrative support.

I want to thank Senthil Yogamani for his extraordinary mentorship. His enthusiasm for new ideas is never-ending. Thank you for the wonderful opportunities you opened for me. I want to thank several colleagues I was fortunate to learn from: Dr. Philip Chen, Ganesh Sistu, Dr. Kalyani Chaganti, Gireesh Suresh, and Venkatraman Narayanan.

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGEMENTS iii

LIST OF FIGURES ix

LIST OF TABLES xiii

ABSTRACT xvi

CHAPTER

I. Introduction 1

1.1 Motivation 1

1.2 Approach 2

1.3 Visual Perception 3

1.3.1 Semantic Segmentation 3

1.3.2 Object Detection 5

1.3.3 Instance Segmentation 5

1.3.4 Panoptic Segmentation 6

1.3.5 General Object Detection 7

1.3.6 Monocular Depth Estimation 9

1.3.7 Motion Segmentation 11

1.3.8 Soiling Detection 12

1.4 Multi-task Learning . . . 13
1.5 Contributions . . . 14
II. Panoptic Segmentation . . . 20
2.1 Introduction . . . 20
2.1.1 Instance Contour Segmentation . . . 22
2.1.2 Challenges . . . 22
2.2 Method . . . 23
2.2.1 Model Architecture . . . 24
2.2.2 Loss Functions . . . 25
2.2.3 Instance Segmentation . . . 26
2.2.4 Refining Instance Segmentation . . . 27
2.2.5 Panoptic Segmentation . . . 28
2.3 Experiments and Results . . . 28
2.3.1 Experimental Setup . . . 28
2.3.2 Ablation Studies . . . 30
2.3.2.1 Instance Contour Segmentation Loss Function . . . 30
2.3.2.2 Instance Contour Ground Truth Dilation Rate . . . 31
2.3.2.3 Refining Instance Segmentation . . . 32
2.3.2.4 Network Ablation . . . 33
2.3.3 State of the Art Comparison . . . 33
2.3.3.1 Comparison with Two-stage Methods . . . 33
2.3.3.2 Comparison with Instance Clustering . . . 35
2.3.3.3 Comparison with Single-stage Object Detection and Others 35
2.4 Discussions . . . 35
2.5 Conclusion . . . 36

III. Multi-stream Learning . . . 37
3.1 Introduction . . . 37
3.2 Extending Semantic Segmentation to Videos . . . 38
3.2.1 Single Frame Baseline . . . 40
3.2.2 Detect and Track Approach . . . 40
3.2.3 Temporal Post Processing . . . 40
3.2.4 Recurrent Encoder Model . . . 41
3.2.5 Fused Multi-stream Encoder Model . . . 41
3.3 Method . . . 41
3.3.1 Single Stream Architecture . . . 42
3.3.2 Multi-stream Fused Architectures . . . 42
3.3.3 Multi-stream Recurrent Architecture . . . 43
3.4 Experiments and Results . . . 44
3.4.1 Experimental Setup . . . 44
3.4.2 Ablation Studies . . . 45
3.4.2.1 Temporal Depth . . . 46
3.4.2.2 Shared Weights . . . 49
3.4.2.3 Non Automotive Datasets . . . 51
3.5 Discussions . . . 51
3.5.1 MSFCN vs FCN . . . 52
3.5.2 MSFCN-2 vs MSFCN-3 . . . 53
3.5.3 MSFCN-2 vs RFCN . . . 53
3.5.4 Weight Sharing . . . 53
3.6 Conclusion . . . 53
IV. Multi-task Learning . . . 55

4.1 Introduction . . . 55
4.2 Universality of CNN Features . . . 56
4.2.1 Adaptation . . . 57
4.3 Pros and Cons of Multi-task Learning . . . 58
4.3.1 Pros . . . 58
4.3.2 Cons . . . 59
4.4 Unified Visual Perception Model . . . 59
4.4.1 Two Task Model . . . 60
4.4.2 Three Task Model . . . 64
4.5 Conclusion . . . 66
V. Auxiliary Learning . . . 67
5.1 Introduction . . . 67
5.2 Motivation . . . 68
5.3 Methods . . . 69
5.3.1 Architecture Design . . . 69
5.3.2 Loss Function . . . 70
5.4 Experiments and Results . . . 72
5.4.1 Experimental Setup . . . 72
5.4.2 Results and Discussion . . . 73
5.5 Conclusion . . . 76
VI. Multi-stream Multi-task Learning . . . 77
6.1 Introduction . . . 77
6.1.1 Feature Aggregation . . . 79
6.1.2 Multi-task Loss . . . 80
6.2 Methods . . . 81

6.2.1 Multi-stream Multi-task Architecture 82
6.2.2 Geometric Loss Strategy 83
6.3 Experiments and Results 84
6.3.1 Datasets 84
6.3.2 Model Analysis 85
6.3.3 Optimization 88
6.3.4 Results 89
6.4 Conclusion 96
VII.Conclusions 97
7.1 Findings, Limitations and Future Work 98
7.2 Broader Impacts 100
BIBLIOGRAPHY 101

LIST OF FIGURES

Figure

1.1 Top: RGB input image, Bottom: Semantic segmentation. 4

1.2 Illustration of a fully convolutional neural network for semantic segmentation. 4

1.3 Top: RGB input image, Bottom: Object detection. 6

1.4 Top: RGB input image, Bottom: Instance segmentation. 7

1.5 Top: RGB input image, Bottom: Instance offset regression. 8

1.6 Top: RGB input image, Bottom: Panoptic segmentation. 9

1.7 Example of general object detection. 10

1.8 Top: RGB input image, Bottom: Ground truth depth from a velodyne. . . . 10

1.9 Example of motion segmentation. Moving objects are marked in green. . . 11

1.10 Illustration of a soiling on an automotive camera. 12

1.11 Illustration of a multi-task learning network for joint semantic segmentation and object detection. 13

2.1 Illustration of (a) Semantic segmentation, (b) Instance contour segmentation, (c) Instance center regression and (d) Instance segmentation. 21

2.2 Illustration of our method that learns panoptic segmentation from instance contours. 23

2.3 Our panoptic segmentation model architecture with CNN backbone. 24

2.4 Illustrative flow diagram of our algorithm that learns panoptic segmentation from semantic segmentation and instance contours. 27

2.5 Qualitative results on Cityscapes val dataset obtained with ResNet-50 encoder using separate necks, wBCE + Huber loss combination, split and merge refinement with min Instance area = 300 pixels. Instance contours ground truth are generated with dilation rate = 2. From left to right: Semantic segmentation, instance contour segmentation, center regression, instance segmentation. Top: ground truth, Bottom: prediction. Predicted contours are thicker than ground truth. 29
2.6 Panoptic segmentation results on Cityscapes val dataset. Results obtained with ResNet-50 encoder using a separate neck architecture, wBCE + Huber loss combination, split and merge refinement with min Instance area = 300 pixels. Instance contours ground truth are generated with dilation rate = 2. . 30
3.1 Top: Recurrent FCN, Bottom: Multi-stream FCN. 38
3.2 Comparison of different approaches to extend semantic segmentation to videos - a) Frame-level output b) Detect and track c) Temporal post processing d) Recurrent encoder model and e) Fused multi-stream encoder model. 39
3.3 FCN: Single encoder baseline. 41
3.4 MSFCN-2: Two stream fusion architecture. 42
3.5 MSFCN-3: Three stream fusion architecture. 43
3.6 RFCN-2: Two stream LSTM architecture. 44
3.7 Accuracy over epochs for SYNTHIA dataset. 45
3.8 Qualitative results of experiments with SYNTHIA dataset. Left to right: RGB image, single encoder (FCN), two stream encoder (MSFCN-2), ground truth, two stream encoder + LSTM (RFCN-2) and three stream encoder (MSFCN-3). 47

3.9 Results on KITTI dataset. 49
3.10 Results over DAVIS dataset. Left to right: RGB image, ground truth, single encoder (FCN), two stream encoder (MSFCN-2), two stream encoder + LSTM (RFCN-2), three stream encoder (MSFCN-3). 51
4.1 Unified model for the important visual perception tasks in automated driving. 60
4.2 Qualitative results of two-task model performing segmentation and detection. 63
4.3 Illustration of three-task model architecture comprising of object detection, semantic segmentation and soiling detection tasks. 64
5.1 Illustration of several auxiliary visual perception tasks in an automated driving dataset KITTI. First row shows RGB and semantic segmentation, second row shows dense optical flow and depth, third row shows visual SLAM and meta-data for steering angle, location and condition. 68
5.2 AuxNet: Auxiliary learning network with segmentation as main task and depth estimation as auxiliary task. 70
5.3 Results on KITTI (Top) and SYNTHIA (Bottom) datasets. 75
6.1 Illustration of MultiNet++ where feature aggregation is performed to combine intermediate output data obtained from a shared encoder that operates on multiple input streams (frames ‘t’ and ‘t-1’). The aggregated features are later processed by task specific decoders. 78
6.2 Illustration of the MultiNet++ network operating on consecutive frames of input video sequence. Consecutive frames are processed by a shared siamese-style encoder and extracted features are concatenated and processed by task specific segmentation, depth estimation and moving object detection decoders. 81

6.3 Change of validation loss (X-axis) over several epochs (Y-axis) during training phase for 1-Task model vs 3-Task models for segmentation, depth and motion tasks on KITTI dataset. 91
6.4 Change of validation loss (X-axis) over several epochs (Y-axis) during training phase for 1-Task model vs 3-Task models for segmentation, depth and motion tasks on Cityscapes dataset. 92
6.5 Change of validation loss (X-axis) over several epochs (Y-axis) during training phase for 1-Task model vs 2-Task models for segmentation and depth on SYNTHIA dataset. 93
6.6 Left to right: Input image, single task network outputs, MultiNet++ output, ground truth. More qualitative results of MultiNet++ model can be accessed via this link https://youtu.be/E378PzLq7lQ. 94
6.7 Left to right: Input image, semantic segmentation output from single task, 3-Task with equal weights, 3-Task GLS, 3-Task MultiNet++ networks, ground truth. 95

LIST OF TABLES

Table

2.1 Instance and Panoptic Segmentation results on Cityscapes val dataset for different loss functions used to represent instance contour loss. wBCE = weighted binary cross entropy, AP = average precision, PQ = panoptic quality. PQ^{Th}, SQ^{Th}, and RQ^{Th} represent panoptic, segmentation and recognition qualities of instance objects. 31

2.2 Performance of instance and panoptic segmentation on Cityscapes val dataset when different dilation rates were used to generate ground truth instance contours. Increasing the dilation rate, increases the thickness of the ground truth instance contours. 31

2.3 Evaluation of instance and panoptic segmentation on Cityscapes val dataset before and after refinement using offsets predicted by center regression results. 32

2.4 Impact of minimum instance area threshold during instance refinement on Cityscapes val dataset. 32

2.5 Performance of semantic, instance and panoptic segmentation using different network architecture choices on Cityscapes val dataset. 33

2.6 Comparison with other state-of-the art methods on Cityscapes dataset (val split). † Performance reported on test split. *Evaluated on image of size 416×832. 34

3.1 Semantic segmentation results on SYNTHIA sequences. We split the test sequences into two parts, one is Highway for high speeds and the other is City for medium speeds. 48
3.2 Semantic segmentation Results on KITTI video sequence. 50
3.3 Semantic segmentation Results on SYNTHIA video sequence. 50
3.4 Comparison of multi-stream network with its baseline counterpart on SegTrack and DAVIS. 52
4.1 Comparison study: Single-task vs two-task, JI: Jaccard index, AP: Average precision. IOU: Intersection over union. 62
4.2 Comparison study: Single-task vs. three-task models. 65
5.1 Comparison study : Single task vs auxiliary learning. AuxNet_{400} and AuxNet_{1000} weighs segmentation loss 400 and 1000 times compared to depth loss. $\text{AuxNet}_{\text{TWB}}$ is constructed by expressing total loss as product of task losses. 74
5.2 Comparison between SegNet, FuseNet and AuxNet in terms of performance and parameters. 76
6.1 Summary of the automotive datasets used in our experiments. 85
6.2 Comparative study: Parameters needed to construct 1-task segmentation, depth and motion, 2-task segmentation and depth, 2-task segmentation and motion and 3-task segmentation, depth and motion models. We compare 2-task and 3-task models that operate on 1-frame and 2-frames. 2-frame models required relatively minimal additional computational complexity compared to 1-frame models. 87

6.3 Comparative Study: Performance of 1-Task, equal weights, 3-task uncertainty, Dynamic Weight Average (DWA) and geometric loss strategy (GLS) on KITTI and Cityscapes datasets. 89
6.4 Improvements in learning segmentation, depth estimation and motion detection as multiple tasks using equal weights, geometric loss strategy (GLS) and 2 stream feature aggregation with GLS (MultiNet++) vs independent networks (1-Task) on KITTI, Cityscapes and SYNTHIA datasets. 90

ABSTRACT

Every year, 1.2 million people die, and up to 50 million people are injured in accidents worldwide. Automated driving can significantly reduce that number. Automated driving also has several economic and societal benefits that include convenient and efficient transportation, enhanced mobility for the disabled and elderly population, etc.

Visual perception is the ability to perceive the environment, which is a critical component in decision-making that builds safer automated driving. Recent progress in computer vision and deep learning paired with high-quality sensors like cameras and LiDARs fueled mature visual perception solutions. The main bottleneck for these solutions is the limited processing power available to build real-time applications. This bottleneck often leads to a trade-off between performance and run-time efficiency.

To address these bottlenecks, we focus on: 1) building optimized architectures for different visual perception tasks like semantic segmentation, panoptic segmentation, etc. using convolutional neural networks that have high performance and low computational complexity, 2) using multi-task learning to overcome computational bottlenecks by sharing the initial convolutional layers between different tasks while developing advanced learning strategies that achieve balanced learning between tasks.

CHAPTER I

Introduction

1.1 Motivation

Automated driving comprises key components like perception, localization & mapping, sensor fusion, path planning, and decision control. Perception involves geometric and semantic understanding of the environment. For perception, cameras are a dominant sensor as the roadway infrastructure is typically created for human visual perception. Semantic tasks such as object detection [1, 2, 3, 4] (detecting cars, pedestrians etc with bounding boxes), semantic segmentation [5, 6, 7] (pixel-wise labeling of road, lane markings etc) and geometric tasks like depth estimation [8, 9, 10, 11, 12] (distance in real-world from ego vehicle), motion estimation [13, 14, 15, 16, 17, 18] (detect motion and estimate velocities of moving objects) etc are some major tasks that help build a visual perception system.

Before the success of deep learning, traditional computer vision-based engineered feature descriptors coupled with lightweight machine learning classifiers were used to solve several visual perception tasks. Convolutional Neural Networks (CNNs) [19, 20, 21] dominate state of the art in visual perception and are the standard models used in the latest generation of vehicles for automated driving and advanced driver assistance applications. Deep Neural Networks require dedicated hardware often equipped with GPUs or specific hardware accelerators to meet automated driving's real-time requirements. Such expensive hardware limits the amount of processing power available. Thus, there is significant

importance in optimizing deep neural network architectures.

1.2 Approach

Optimization of deep neural network architectures can be addressed in two methods: 1) developing lightweight models that reduce computational complexity, memory footprint while maintaining the performance compared to a large model, 2) developing a single model that can perform multiple tasks by reusing the parameters that posses common knowledge across different tasks. In the first method, we try to build a smaller model with reduced complexity, whereas in the second method, we may build a single complex model to solve multiple tasks. Still, we reduce the complexity when compared to the total complexity of multiple tasks. The latter is referred to as Multi-task Learning (MTL).

This dissertation shows how to formulate deep learning models for visual perception applications in automated driving using the above two methods. The main applications that are addressed in this dissertation include:

1. Semantic Segmentation (What type of an object at a pixel level in an image)
2. Object Detection (Where is or what type of an object in an image)
3. Monocular Depth Estimation (How far is an object at a pixel level in an image)
4. Motion Segmentation (Is the object moving)
5. Instance Segmentation (What type of an object at a pixel level along with an id)
6. Panoptic Segmentation (Joint semantic (uncountable) and instance (countable) segmentation)
7. Video Semantic Segmentation (What type of an object at a pixel level in a video)

8. Multi-task task learning (Learn multiple tasks using a single input)

9. Auxiliary Learning (Learn multiple tasks with more focus on a single task)

10. Multi-stream Multi-task Learning (Learn multiple tasks using multiple inputs)

1.3 Visual Perception

Visual Perception in automated driving has witnessed tremendous progress over the past decade with the introduction of Convolution Neural Networks that aided in developing the scene and geometric understanding. Panoptic segmentation [22, 23, 24, 25], a joint semantic [5, 6, 7] and instance segmentation [26, 27, 28, 29, 30] has provided complete scene understanding by categorizing a pixel into distinct categories and instances. Monocular depth [8, 9, 10, 11, 12] and motion estimation [13, 14, 15, 16, 17, 18] provide the understanding of geometry in the scene. In the following subsections, we introduce several visual perception tasks.

1.3.1 Semantic Segmentation

Semantic segmentation refers to a pixel-wise classification of a scene as shown in Figure 1.1. It provides dense pixel-wise labeling of the image, which leads to scene understanding. A few years ago, semantic segmentation was considered a challenging problem. With the help of fully convolutional neural networks (FCNs), [6], the development of accurate and efficient solutions were made possible. The level of maturity of semantic segmentation has rapidly grown recently, and the computational power of embedded systems has increased to enable commercial deployment.

A Fully convolutional neural network is a CNN based encoder-decoder network (shown in Figure 1.2), where the encoder performs feature extraction of input image, which is

Figure 1.1: Top: RGB input image, Bottom: Semantic segmentation.

decoded by an up-sampling network to generate pixel-wise classification result.

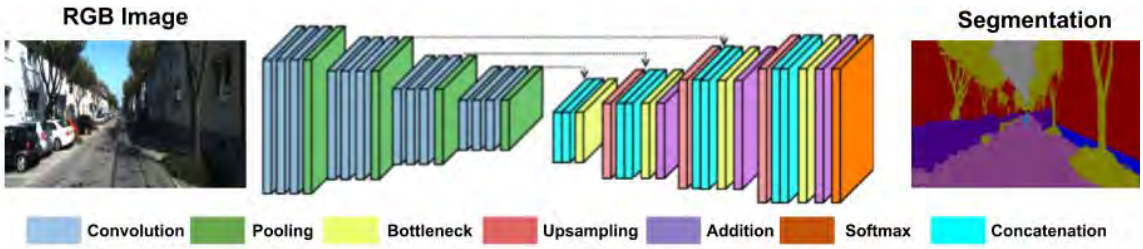


Figure 1.2: Illustration of a fully convolutional neural network for semantic segmentation.

A detailed survey of semantic segmentation for automated driving is presented in [31]. Several enhancements were made to push the performance of semantic segmentation higher by making improvements to encoder and decoder in FCNs [7]. Dilated residual convolutions [32, 33], Feature pyramid networks [4, 24], Spatial pyramid pooling [34] etc. are

examples of improvements made to encoder while U-Net [35], Densely connected CRFs [36] are examples of improvements made to decoder.

1.3.2 Object Detection

Object detection (Figure 1.3) involves recognizing category and localizing by position different objects in an input image. Fully convolution neural networks are successfully used to solve object detection task. The CNN based bounding box detection can be broadly categorized into two groups, single-stage and two-stage approaches. Single-stage approaches regress for box co-ordinates and class categories in one shot. YOLO [3] and SSD [37] are pioneering works in single-stage methods. On the other hand, two-stage networks involve explicit loss functions for class agnostic region proposals followed by accurate box co-ordinates regression. The R-CNN family of algorithms [38] fall into this category.

1.3.3 Instance Segmentation

In instance segmentation (shown in Figure 1.4), an object instance(id) is assigned to every pixel for every known object within an image.

The majority of instance segmentation networks are two-stage methods. Two-stage methods like MaskR-CNN [39] involves proposal generation from object detection followed by mask generation using a foreground/background binary segmentation network. These methods dominate state of the art in instance segmentation but incur a relatively higher computational cost. Using YOLO [3], SSD [37] and other lightweight object detector compared to Faster R-CNN [38] may seem promising. However, they still possess inevitable additional compute in generating object proposals followed by mask generation.

Other approaches in instance segmentation range from clustering of instance embedding [29] to prediction of instance centers using offset regression (shown in Figure 1.5)

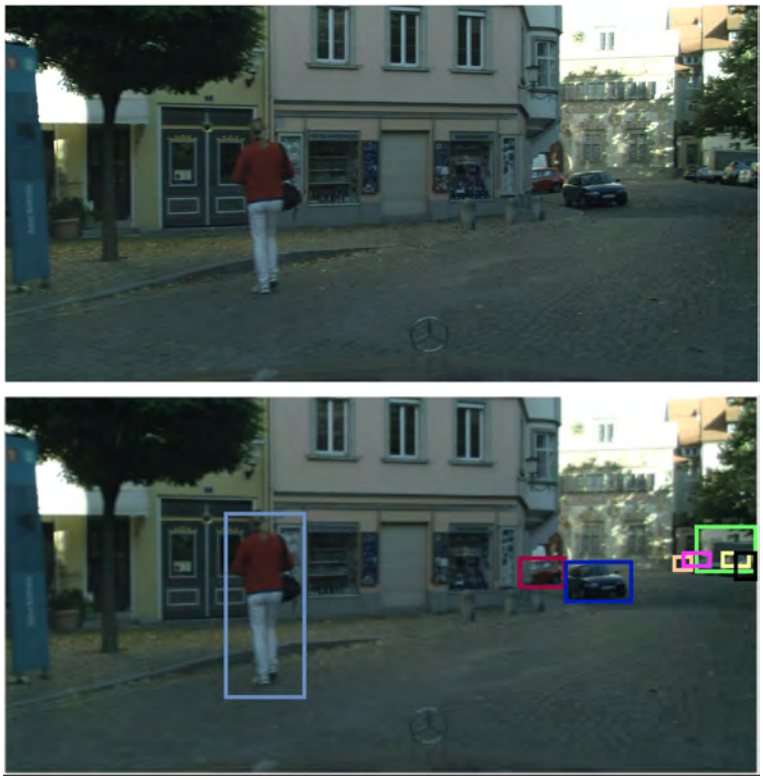


Figure 1.3: Top: RGB input image, Bottom: Object detection.

[40, 23]. These methods appear logically straightforward but are lagging in terms of accuracy and computational efficiency. The major drawback with these methods is usage of compute-intensive clustering methods like OPTICS [41], DBSCAN [42] etc.

1.3.4 Panoptic Segmentation

Panoptic segmentation [22] shown in Figure 1.6 combines semantic segmentation and instance segmentation to provide the class category and instance id for every pixel within an image. Recent works [24, 23, 43] use a shared backbone and predict panoptic segmentation by fusing output from semantic and instance segmentation branches. Almost every work so far uses an FCN based semantic segmentation branch with variations including usage of dilated convolutions [23] or feature pyramid networks [24]. However, choices of instance

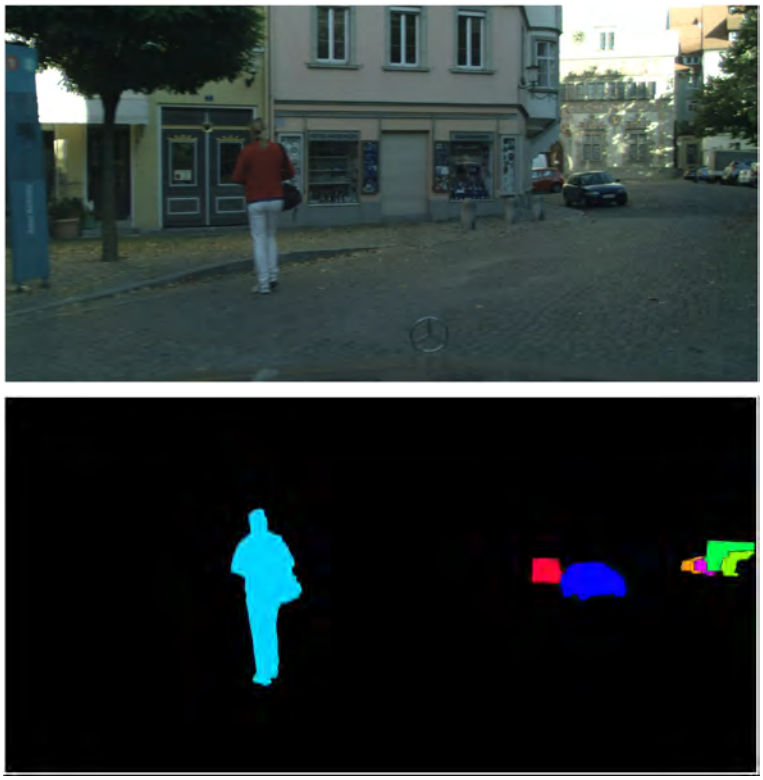


Figure 1.4: Top: RGB input image, Bottom: Instance segmentation.

segmentation branches can vary, as discussed earlier.

1.3.5 General Object Detection

It is impossible to list the entire set of objects in an automotive scenario as this set can be considered infinite cardinality for all practical purposes. Thus, there will always be objects that are not trained to use CNN-based object detection discussed earlier (e.g., kangaroos, moose, or obscure construction vehicles with distinctly unfamiliar visual appearances). In some cases, trucks or buses with ads can confuse the object detection model. Thus appearance agnostic object detection is critical for automated driving systems, and alternate geometric cues of motion and depth are required. Even in standard objects like pedestrians and vehicles, such cues will aid the robustness of detection.

Figure 1.5: Top: RGB input image, Bottom: Instance offset regression.

In classical computer vision, motion is computed using optical flow, and depth is computed using structure from motion. Then post-processing algorithms, such as clustering, are performed to extract generic static and moving objects. CNN's can also be used to extract moving objects and static objects directly instead of the intermediate pixel-level flow or more complex depth estimation. Moving Object Detection Network (MODNet) [44] poses moving object detection as a binary segmentation problem and directly estimated. StixelNet [45] poses generic static obstacles represented as stixels and learned using a CNN as shown in Figure 1.7.

Figure 1.6: Top: RGB input image, Bottom: Panoptic segmentation.

1.3.6 Monocular Depth Estimation

Monocular Depth estimation involves estimating the distance two an object (or any plane) at a pixel level as shown in Figure 1.8. It is an essential task for detecting generic obstacles and also enables tasks such as localization, obstacle avoidance, safe interaction, and manipulation with objects in the environment, among many others. Traditionally, specialized vision sensors are used to obtain depth information along with color images. Stereo cameras, which apply binocular vision principles, are the first vision sensors used to get depth information. The literature on stereo vision is rich, and well-developed [46]. Stereo cameras have been used successfully to solve many robotics perception tasks such as visual odometry [47]. Other specialized vision sensors are ones based on structured light

Figure 1.7: Example of general object detection.

(RGB-D) that can measure depth directly. This type of sensor has been used successfully in mapping, reconstruction, and tracking applications [48].

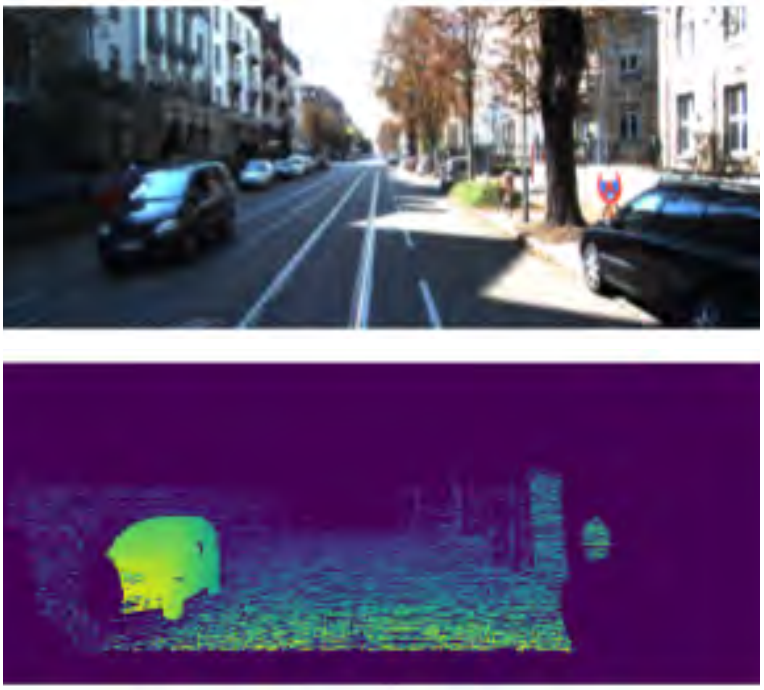


Figure 1.8: Top: RGB input image, Bottom: Ground truth depth from a velodyne.

Convolutional Neural Networks (CNNs) have been used successfully to perform monoc-

ular depth prediction or estimating depth from a single image [8, 9, 10, 11, 12, 49]. Moreover, recent works attempt to combine depth prediction in a unified visual perception framework where a single network can perform other tasks such as semantic segmentation and object detection as well [50, 51].

1.3.7 Motion Segmentation

In automotive driving, motion is a strong cue due to the cameras' ego-motion on the moving vehicle, and dynamic objects around the car are the critical interacting agents. Additionally, it helps detect generic objects based on motion cues rather than appearance cues as there will always be rare objects like kangaroos or construction trucks. Moving Object detection has been explored in [18, 44]. Motion segmentation is treated as a binary segmentation problem as shown in Figure 1.9, and IoU is used as the metric.

Figure 1.9: Example of motion segmentation. Moving objects are marked in green.

1.3.8 Soiling Detection

Cameras embedded within the vehicles are directly exposed to an external environment, and there is a good chance that they get soiled due to bad weather conditions such as rain, fog, snow, etc. [52, 53]. Moreover, dust and mud have a substantial effect on degraded computer vision performance. Compared to other types of sensors, cameras have much higher degradation in performance due to soiling. Thus, it is critical to robustly detect soiling on the cameras, especially for higher autonomous driving levels. In Figure 1.10, we show exemplary images with the corresponding annotations for the camera soiling detection task.

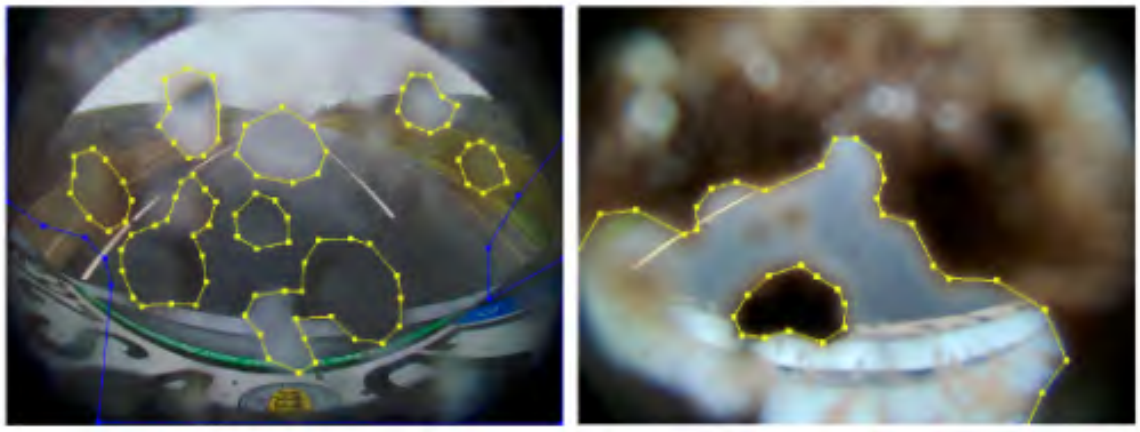


Figure 1.10: Illustration of a soiling on an automotive camera.

Soiling detection was first implemented to alarm the driver that there will be degraded performance in the environment perception system. There could be fatal consequences if information from soiled cameras is relied on in a high-level autonomous system, without having prior information that it is not correct. According to the SAE autonomous levels definition [54], adverse weather detection is a necessary functionality for achieving Level 5 autonomy.

1.4 Multi-task Learning

Multi-task learning [55] (shown in Figure 1.11) refers to joint training of multiple tasks or networks. In general, Multi-task Learning (MTL) aims to overcome the computational bottlenecks in Convolutional Neural Networks and improve computational efficiency by sharing the expensive layers between all tasks. This allowed deployment of MTL networks in various applications in computer vision (especially scene understanding) [56, 57, 50], natural language processing [58, 59], speech recognition [60, 61], reinforcement learning [62, 63], drug discovery [64, 65], etc.

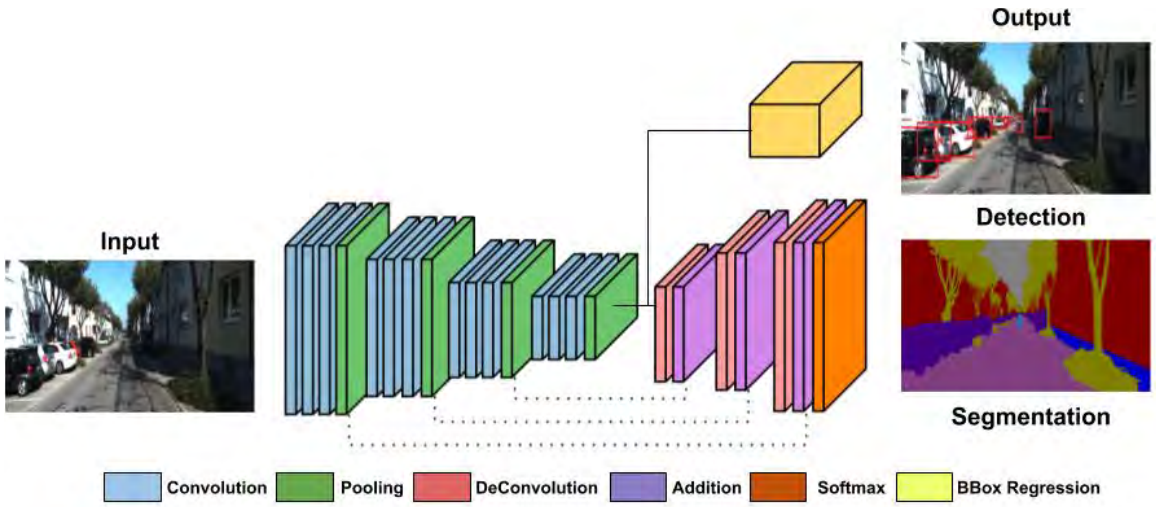


Figure 1.11: Illustration of a multi-task learning network for joint semantic segmentation and object detection.

Multi-task learning typically consists of two blocks, shared parameters and task-specific parameters. Shared parameters are learned to represent commonalities between several tasks, while task-specific parameters are learned to perform independent processing. In MTL networks built using CNNs, shared parameters are called encoders as they perform the key feature extraction, and the task-specific parameters are called decoders as they decode the information from encoders. MTL networks are classified into hard parameter

sharing or soft parameter sharing categories based on how they share their parameters. In hard parameter sharing, initial layers or parameters are shared between different tasks such that these parameters are common for all tasks. In soft parameter sharing, different tasks are allowed to have different initial layers with some extent of sharing between them. Cross stitch [66] and sluice networks [67] are examples of soft parameter sharing. The majority of the works in MTL use hard parameter sharing as it is easier to build and computationally less complex.

The performance of the MTL network is highly dependent on their shared parameters as they contain the knowledge learned from different tasks [55, 68]. Inappropriate learning of these parameters can induce biased representations for a particular task, which can hurt the performance of MTL networks. This phenomenon is referred to as negative transfer learning. In order to prevent it, balanced learning methods are required. An ideal loss function should enable the learning of multiple tasks with equal importance irrespective of loss magnitude, task complexity, etc. Manual tuning of task weights in a loss function is a tedious process, and it is prone to errors. Most of the work in multi-task learning uses a linear combination of multiple task losses. Recent works [40, 69, 70] have attempted to learn/assign the task weights based on task heuristics like uncertainty, change of loss rate etc. Modelling multi-task loss as a multi-objective function was proposed in [71], [72] and [73].

1.5 Contributions

In my dissertation, I will focus on building optimized deep learning architectures that improve individual perception tasks and develop a joint model that shares the available compute power to process multiple tasks. In the following subsection, we introduce each

chapter in this dissertation and then present the key contributions.

Chapter 2: Panoptic Segmentation

Panoptic segmentation aims to understand background (stuff) and instances of objects (things) at a pixel level. It combines the separate semantic segmentation (pixel-level classification) and instance segmentation tasks to build a single unified scene understanding task. Typically, panoptic segmentation is derived by combining semantic and instance segmentation tasks learned separately or jointly (multi-task networks). In general, instance segmentation networks are built by adding a foreground mask estimation layer on top of object detectors or using instance clustering methods that assign a pixel to an instance center. This chapter presents a fully convolution neural network that learns instance segmentation from semantic segmentation and instance contours (boundaries of things). Instance contours along with semantic segmentation yield a boundary aware semantic segmentation of things. Connected component labeling on these results produces instance segmentation. We merge semantic and instance segmentation results to output panoptic segmentation. We evaluate our proposed method on the CityScapes dataset to demonstrate qualitative and quantitative performances along with several ablation studies.

We hope that our idea encourages a new direction in panoptic segmentation research, which ultimately leads to the learning of instance separating contours within the segmentation task. The main contributions of this chapter include:

1. A novel method to learn panoptic segmentation and instance segmentation from semantic segmentation and instance contours.

2. An instance contour segmentation network that learns boundaries between objects of the same semantic category.

Chapter 3: Multi-stream Learning

The majority of visual perception algorithms like semantic segmentation operate on a single frame, even in videos. This chapter aims to exploit temporal information within the algorithm model for leveraging motion cues and temporal consistency. We propose two simple high-level architectures based on Recurrent FCN (RFCN) and Multi-Stream FCN (MSFCN) networks. In RFCN, a recurrent network, namely Long Short Term Memory network (LSTM), is inserted between the encoder and decoder. MSFCN combines the encoders of different frames into a fused encoder via 1x1 channel-wise convolution. We use a ResNet50 [21] network as the baseline encoder and construct three networks, namely MSFCN of order 2 & 3 and RFCN of order 2. MSFCN-3 produces the best results with an accuracy improvement of 9% and 15% for Highway and New York-like city scenarios in the SYNTHIA-CVPR'16 [74] dataset using the mean IoU metric. MSFCN-3 also produced 11% and 6% for SegTrack V2 [75] and DAVIS [76] datasets over the baseline FCN network. We also designed an efficient version of MSFCN-2 and RFCN-2 using weight sharing among the two encoders. The efficient MSFCN-2 provided an improvement of 11% and 5% for KITTI and SYNTHIA with a negligible increase in computational complexity compared to the baseline version.

The list of contributions include:

1. Design of Recurrent FCN (RFCN) and Multi-Stream FCN (MSFCN) architectures that extends semantic segmentation models for videos.

2. Implementation of a network for spatio-temporal video semantic segmentation.

3. Detailed experimental analysis of multi-class video semantic segmentation with automated driving dataset SYNTHIA [74] and binary video segmentation with SegTrack V2 [75] and DAVIS [76] datasets.

Chapter 4: Multi-task Learning

Convolutional Neural Networks (CNN) are successfully used for various visual perception tasks, including bounding box object detection, semantic segmentation, optical flow, depth estimation, visual SLAM, etc. Generally, these tasks are independently explored and modeled. In this chapter, we propose a joint multi-task network design for learning multiple tasks simultaneously. The goal is to use the CNN encoder as a generic feature extractor for all tasks to be computationally efficient, improve accuracy, and ease development effort. The main advantages are increased run time efficiency through shared network parameters across tasks, scalability to add more tasks leveraging previous features, and better generalization.

The major contributions of this chapter include:

1. Design of a two task network for joint semantic segmentation and objection and experimentation on various datasets to demonstrate that joint network provides the same accuracy as separate networks.

2. Design a three task network for object detection, semantic segmentation, and soiling detection.

3. Comparison of computational complexities between multi-task models vs. independent single-task models.

Chapter 5: Auxiliary Learning

Pixel level classification was once considered a challenging task, becoming mature to be productized in a car. However, semantic annotation is time-consuming and quite expensive. Synthetic datasets with domain adaptation techniques have been used to alleviate the lack of large annotated datasets. In this chapter, we explore an alternate approach to leveraging other tasks' annotations to improve semantic segmentation. Motivated by multi-task

learning, we use auxiliary tasks like depth estimation to improve semantic segmentation task performance. We propose adaptive task loss weighting techniques to address scale issues in multi-task loss functions, which become more crucial in auxiliary tasks. We experimented on automotive datasets including SYNTHIA [74], and KITTI [77] and obtained 3% and 5% improvement in accuracy, respectively.

The contributions of this chapter include:

1. Construction of auxiliary task learning architecture for semantic segmentation.
2. Novel loss function weighting strategy for one main task and one auxiliary task.
3. Experimentation on automotive datasets namely SYNTHIA [74] and KITTI [77].

Chapter 6: Multi-stream Multi-task Learning

Current work on multi-task learning networks focuses on processing a single input image, and there is no known implementation of multi-task learning handling a sequence of images. In this chapter, we propose a multi-stream multi-task network to take advantage of using feature representations from preceding frames in a video sequence for joint learning of segmentation, depth, and motion. The weights of the current and previous encoder are shared so that features computed in the previous frame can be leveraged without additional computation. We also propose using the geometric mean of task losses as a better alternative to the weighted average of task losses. The proposed loss function facilitates better handling of the difference in convergence rates of different tasks. Experimental results on KITTI [77], Cityscapes [78] and SYNTHIA [74] datasets demonstrate that the proposed strategies outperform various existing multi-task learning solutions

The contributions of this chapter include:

1. A multi-stream multi-task network to take advantage of temporal features from preceding frames in a video sequence.

2. Geometric mean of task losses as a better alternative to the weighted average of task losses.

3. Experimentation on three automotive datasets namely KITTI [77], Cityscapes [78] and SYNTHIA [74].

CHAPTER II

Panoptic Segmentation

2.1 Introduction

Panoptic segmentation [22, 24] offers the ultimate understanding of a scene by providing joint semantic and instance-level predictions of background and objects at a pixel level. Panoptic segmentation is usually achieved by combining outputs from semantic segmentation and instance segmentation. Examples where panoptic segmentation offers an unprecedented advantage over standalone semantic or instance segmentation solutions, include collective knowledge of distinct objects and the drivable area around a self-driving car [43, 79], semantic and instance-level details of cancerous cells in digital pathology [80], understanding of the background and different individuals in a frame to enhance smartphone photography. Multi-task learning networks [55] that jointly perform semantic, and instance segmentation [24, 43] accelerated progress of panoptic segmentation in terms of accuracy and computational efficiency compared to traditional methods that use a naive fusion of predictions from independent semantic and instance segmentation networks to derive panoptic segmentation output [22].

Instance segmentation is typically achieved in two major ways, 1) Foreground mask estimation of objects detected by an object detection model [24, 81, 39] or 2) Clustering-based instance assignment methods [23, 40]. Recently, single-stage instance segmentation methods have been developed [82, 27]. These major approaches use fully convolution networks so that they can be trained in an end-to-end fashion. Clustering-based instance

assignments appear logically straight forward but are lagging in terms of accuracy and computational efficiency. The major drawback with these methods is usage of compute-intensive clustering methods like OPTICS [41], DBSCAN [42] etc. In contrast to these methods, we derive instance segmentation from semantic segmentation using instance contours (boundaries of things).

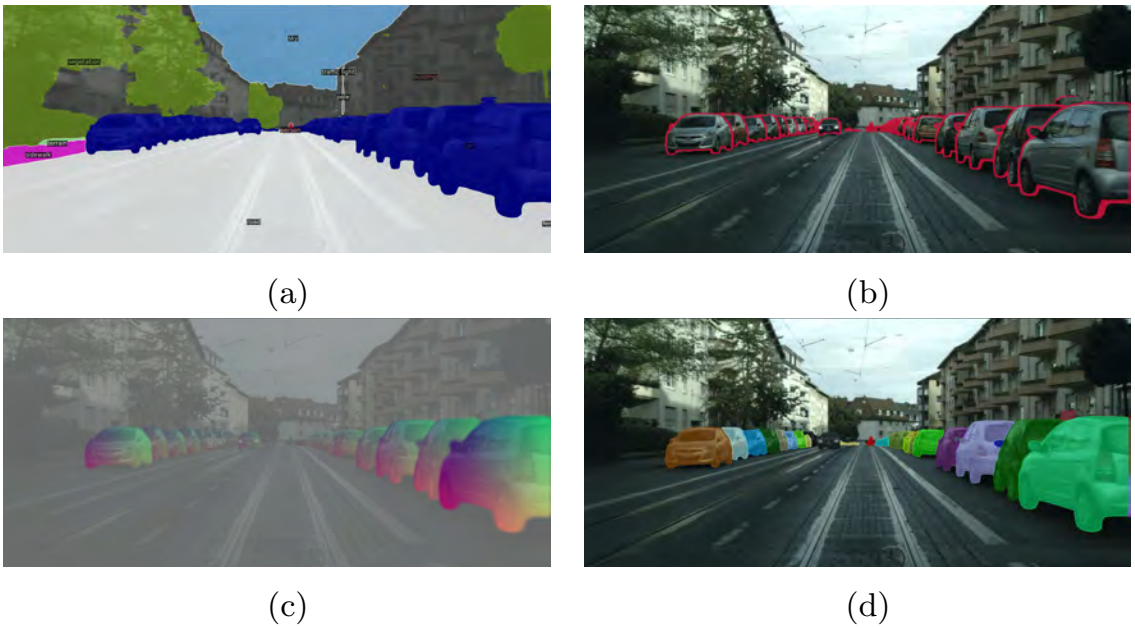


Figure 2.1: Illustration of (a) Semantic segmentation, (b) Instance contour segmentation, (c) Instance center regression and (d) Instance segmentation.

On the other hand, Semantic segmentation is a mature task that is well explored in the literature relative to panoptic segmentation. We observe that panoptic segmentation can be obtained from semantic segmentation by additionally estimating instance separating contours. Naively, the instance separating contours can be an additional class in the segmentation task. In practice, it isn't easy to get good performance for this class. It is illustrated in Figure 2.1 where segmentation (a) and instance contour segmentation (b) contains all the information to obtain panoptic segmentation. The minimal contours needed are contours that separate two instances of the same object. However, these contours do not

have sufficient information to be learned on their own, and thus, we use the entire instance contours.

2.1.1 Instance Contour Segmentation

Semantic edge detection (SED) [83, 84] differs from edge detection [85] by predicting edges that belong to semantic class boundaries. In SED, edges/boundaries that separate segments of one category from another are predicted, whereas, in edge detection, every edge is detected based on image gradients. The main idea in CNN based edge models is combining intermediate feature maps across different layers of network that contain edge information to form semantic boundaries that separate one class from another. Holistically-nested edge detection (HED) [86] is one of the first CNN based edge detection methods that proposed the usage of the above idea. Later, several methods were proposed to address different edge detection challenges that include prediction of crisp boundaries [87, 88], selection of intermediate feature maps, and choices of supervision on these feature maps [89, 90]. It is important to note that these methods ignore the boundaries between objects belonging to the same semantic category. Instance contour estimation as shown in Figure 2.1 (b) can overcome this challenge.

Deep contour [91] are used instance contours generate instance segmentation. Deep Snake [30] recently proposed to predict instance contours by learning contours from object detection. They replace foreground mask estimation for objects with contours to derive instance segmentation.

2.1.2 Challenges

The major challenge in generating panoptic segmentation output is merging conflicting outputs from semantic segmentation and instance branches. For example, semantic

segmentation can predict that a pixel might belong to the car class, while the instance segmentation branch may predict the same pixel as the person class. A naive way to resolve conflicts is by considering the semantic segmentation as the basis and using instance segmentation for instance identification only. Several methods [43] were proposed to handle the conflicts better and learned fashion. Our methods propose to derive instance segmentation from semantic segmentation using instance contours. Therefore, our method does not require a conflict resolution policy like other existing methods.

2.2 Method

Our method (shown in Figure 2.2) is a multi-task neural network with several shared convolution layers and multiple output heads that predict semantic segmentation, instance contours (boundaries of things), and center regression. Instance contours along with semantic segmentation yield a boundary aware semantic segmentation of things. Connected component labeling on these results produces instance segmentation and, eventually, panoptic segmentation. We also estimate a confidence score for each instance.

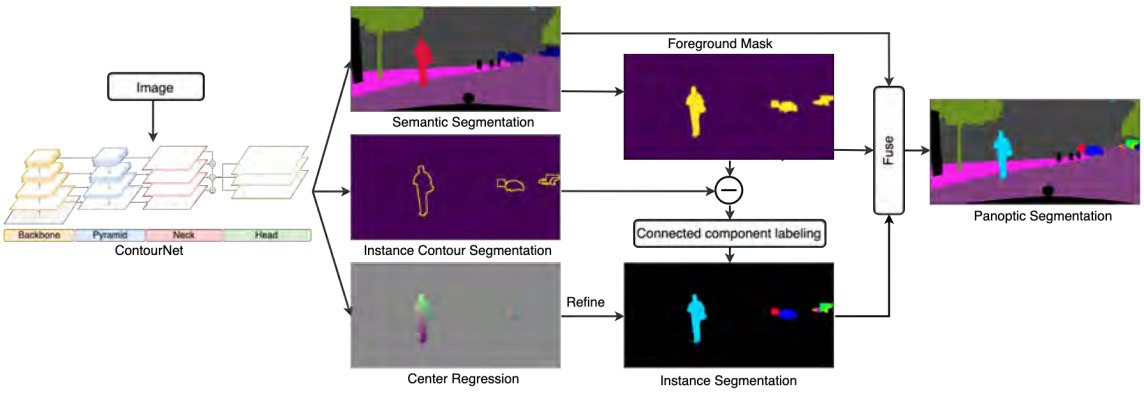


Figure 2.2: Illustration of our method that learns panoptic segmentation from instance contours.

Our instance contour segmentation network is a binary segmentation network that pre-

dicts instance boundaries between objects that belong to the same category. Compared to semantic edge detection networks [83, 87] our instance contour estimation does not ignore boundaries between instances of the same category.

2.2.1 Model Architecture

As shown in Figure 2.3, a common ResNet [21] backbone outputs multi-scale feature maps $\{1/4, 1/8, 1/16, 1/32\}$ w.r.t to input image. Our pyramid is built using Feature pyramid network (FPN) [4] which consumes feature maps (scales 1/4 to 1/32) from backbone in a top-down fashion and outputs feature maps with 256 channels maintaining their input scale. Feature maps from the pyramid are then passed through a series of 1x1 convolutions and are upsampled to 1/4 scale using 2-d bi-linear interpolation in the neck layers as proposed in [24]. These layers have 128 dims at each level. We add these feature maps from different levels and pass them to prediction heads. Our semantic segmentation head contains 1×1 convolution layer with k filters (k output maps for k classes) followed by a 4x upsampling.

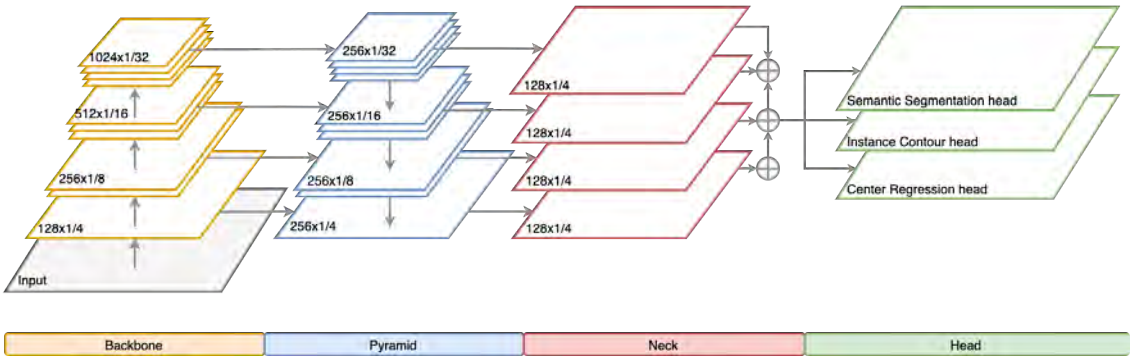


Figure 2.3: Our panoptic segmentation model architecture with CNN backbone.

We perform softmax activation followed by an argmax function on the k output maps to derive full resolution semantic segmentation output. Our instance contour estimation head is similar to the semantic segmentation head, except it has one output feature map

and a sigmoid activation instead of a softmax. Our center regression head has two output channels that predict offsets from the instance center in the x and y-axis and does not have any particular activation function.

2.2.2 Loss Functions

We discuss the explicit loss functions defined for semantic segmentation and instance contour branches. We chose cross-entropy loss for semantic segmentation. In Equation 2.1, $L_{semantic}$ is segmentation loss over k classes for all pixels in the image, where p_i is the prediction probability and $\hat{y}_i$ is ground truth that indicates whether pixel belongs to class i.

$$L_{semantic} = -\sum_{i}^{k} \hat{y}_i \cdot log(p_i) \tag{2.1}$$

For instance contours, we chose weighted binary cross entropy loss [83] as shown in Equation 2.2, where β is the ratio of non edge pixels to total pixels n in the image. p_i is the probability that the current pixel is an edge and $\hat{y}_i$ is ground truth, indicating whether pixel i is an edge.

$$L_{wBCE} = -\sum_{i}^{n} \Big\langle \beta \cdot \hat{y}_i \cdot log(p_i) + (1-\beta) \cdot (1-\hat{y}_i) \cdot log(1-p_i) \Big\rangle \tag{2.2}$$

We add Huber loss (δ = 0.3):

$$L_{Huber} = \begin{cases} 0.5 \cdot (p_i - \hat{y}_i)^2, & |p_i - \hat{y}_i| \leq \delta \\ \delta \cdot (p_i - \hat{y}_i) - 0.5 \cdot \delta^2, & otherwise \end{cases} \tag{2.3}$$

and NMS Loss $\{L_{NMS} = -\sum_c log(h)\}$ [87] terms to contour loss to predict thin and crisp boundaries. We compute softmax response h along the normal direction of boundary pixels

c as described in [87]. For center regression, we use Huber loss to compute error between y, predicted offsets and $\hat{y}$, ground truth offsets with $\delta = 1$.

Our total loss function is a weight combination of semantic loss, contour losses, and center regression loss. We chose λ_1, λ_2, and λ_3 as 1, 50 and 0.1 for our experiments.

$$L_{total} = \lambda_1 \cdot L_{semantic} + \lambda_2 \cdot L_{contour} + \lambda_3 \cdot L_{center} \tag{2.4}$$

where $L_{contour}$ is defined as:

$$L_{contour} = L_{wBCE} + L_{Huber} + L_{NMS} \tag{2.5}$$

2.2.3 Instance Segmentation

Our instance segmentation is derived from semantic segmentation, unlike any other instance segmentation methods as shown in Figure 2.4. As a first step, we generate a binary mask by searching for instance classes in semantic segmentation, which we refer to as instance class mask. We subtract instance contours (generated from instance contour segmentation head) from instance class mask to derive boundary aware instance class mask. Using connected component labeling [92], we derive unique instances from boundary aware instance class mask. We map the semantic segmentation output to the instance generated. We assign the most frequent label found inside an instance as its category and average the softmax predictions over the area of an instance to generate confidence for an instance.

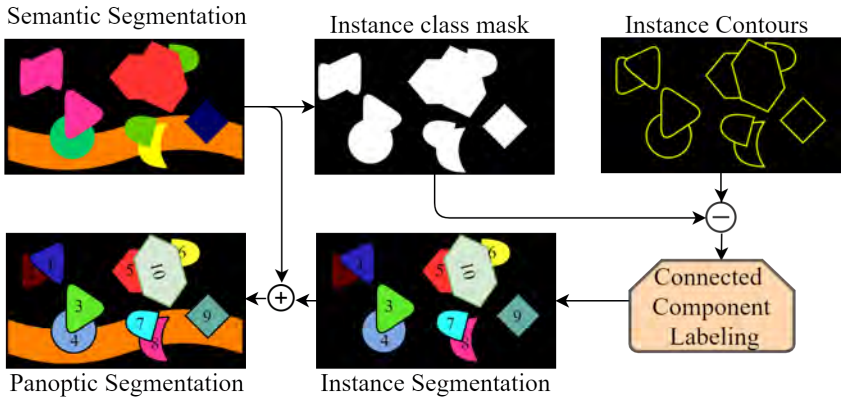


Figure 2.4: Illustrative flow diagram of our algorithm that learns panoptic segmentation from semantic segmentation and instance contours.

2.2.4 Refining Instance Segmentation

We refine instance segmentation output using center regression results. Our refinement consists of mainly two stages: Split and Merge. We estimate centroids predicted by the center regression head. We cluster the centroid predictions using DBSCAN in an instance and split them if distinct centroids are found. DBSCAN requires a fixed 'eps' parameter, which is the maximum distance between two samples for one to be considered in the other's neighborhood. Beyond this distance, two predicted centers will belong to two different instances. If the distance between two centroids is at least 20 pixels (eps), we declare them distinct. For a 1024×2048 image, we believed that 20 pixels are relatively enough to distinguish smaller instances. Our clustering stage does not require considerable computational complexity like other methods [23, 29, 40] since we perform clustering within instances that are much smaller compared to performing clustering on the entire image.

After the instances are split, we estimate mean centroids for every instance using offsets predicted by the center regression head. If the mean centroids are closer than 20 pixels in the euclidean distance, we merge those instances. Later, we remove all instances that have an area lower than a minimum area threshold. We assign these pixels to instances whose

centroids are closest to the centroids derived from offsets predicted by the center regression head.

2.2.5 Panoptic Segmentation

Panoptic segmentation is now obtained by simply merging output from semantic segmentation and instance segmentation. Our methods attempt to derive instance segmentation from semantic segmentation using instance contours. Therefore, our method does not require a conflict resolution policy like other existing methods.

2.3 Experiments and Results

In this section, we demonstrate the performance of our methods for panoptic segmentation on Cityscapes [78] dataset specifically on the validation split. We also present the performance of our semantic segmentation and instance segmentation results.

2.3.1 Experimental Setup

Cityscapes [78] is an automotive scene understanding dataset with 2975/500 train/val images at 1024×2048 resolution. This dataset contains labels for semantic, instance, and panoptic segmentation tasks. We derive labels for our instance contour task by applying a contour detection algorithm on instance ground truth masks. We dilate the resulting contours to derive thick contours and serve them as ground truth for our instance contour segmentation task. Cityscapes dataset has 19 semantic object categories, out of which eight categories are provided with instance masks.

We train our network on full resolution images with a batch size of 4 images. We use Group Normalization [93] which is effective for smaller batch sizes. We use an SGD

optimizer with learning rate = 0.005, momentum = 0.9, weight decay = 10^{-4}. We initialize our ResNet encoders with pre-trained ImageNet [94] weights and train our networks for 48000 iterations.

We measure the performance of semantic segmentation using mean intersection over union (mIoU), instance segmentation using mean average precision (mAP) and panoptic segmentation using panoptic quality (PQ) [22], segmentation quality (SQ), and recognition quality (RQ) metrics. Qualitative results in Figure 2.5 demonstrate that the contours generated are thin and crisp when the above combination is used. Figure 2.6 demonstrates more qualitative results of the panoptic segmentation.

Figure 2.5: Qualitative results on Cityscapes val dataset obtained with ResNet-50 encoder using separate necks, wBCE + Huber loss combination, split and merge refinement with min Instance area = 300 pixels. Instance contours ground truth are generated with dilation rate = 2. From left to right: Semantic segmentation, instance contour segmentation, center regression, instance segmentation. Top: ground truth, Bottom: prediction. Predicted contours are thicker than ground truth.

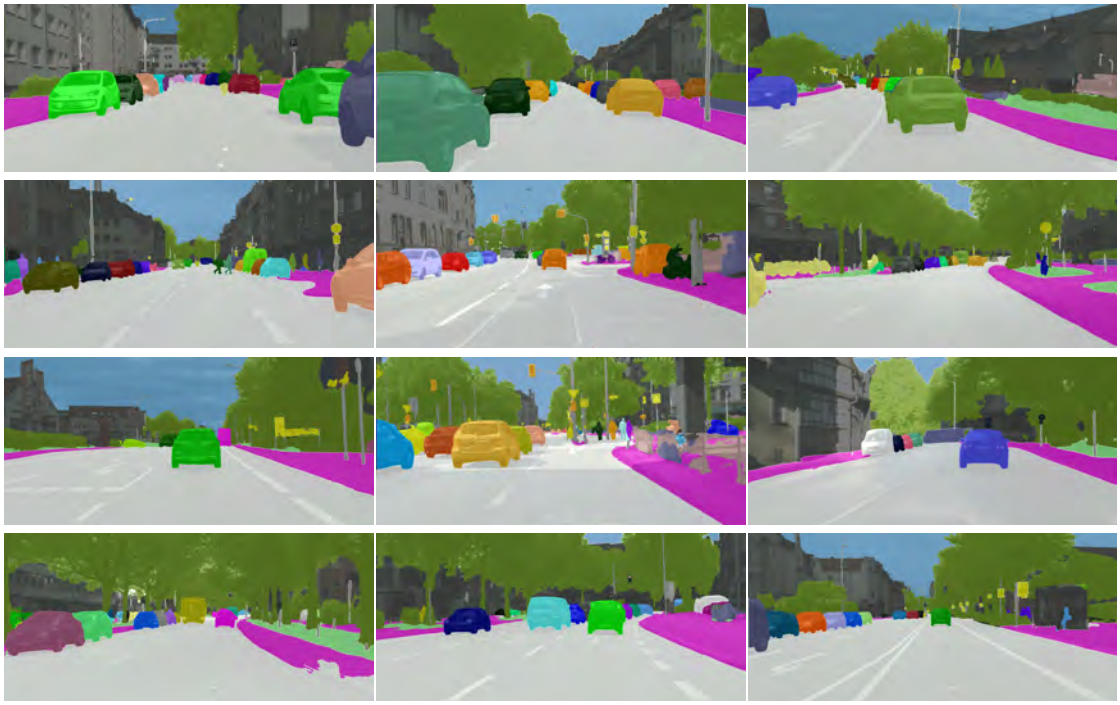


Figure 2.6: Panoptic segmentation results on Cityscapes val dataset. Results obtained with ResNet-50 encoder using a separate neck architecture, wBCE + Huber loss combination, split and merge refinement with min Instance area = 300 pixels. Instance contours ground truth are generated with dilation rate = 2.

2.3.2 Ablation Studies

2.3.2.1 Instance Contour Segmentation Loss Function

As mentioned before, we aim to predict thin and crisp instance contours. We study different loss functions discussed in Section 2.2.2 by evaluating the performance of instance and panoptic segmentation as shown in Table 2.1. We used ResNet-50 encoder as our backbone and separate heads with a common neck.

We observed that Huber and NMS loss function had improved the performance of instance and panoptic segmentation results. The weighted binary cross-entropy combined with the Huber loss is the best combination we found.

Contour Loss			Performance				
wBCE	Huber	NMS	AP	PQ	PQ^{Th}	SQ^{Th}	RQ^{Th}
✓			16.0	43.9	25.0	72.6	33.3
✓	✓		**24.3**	**47.8**	**33.2**	**76.3**	**42.9**
✓		✓	18.9	44.6	26.1	74.3	35.3
✓	✓	✓	23.3	46.7	32.4	76.1	42.1

Table 2.1: Instance and Panoptic Segmentation results on Cityscapes val dataset for different loss functions used to represent instance contour loss. wBCE = weighted binary cross entropy, AP = average precision, PQ = panoptic quality. PQ^{Th}, SQ^{Th}, and RQ^{Th} represent panoptic, segmentation and recognition qualities of instance objects.

2.3.2.2 Instance Contour Ground Truth Dilation Rate

We generate our ground truth instance contours by applying a contour detection algorithm on instance masks provided for different objects in the cityscapes dataset. The number of edge pixels is comparatively lower than non-edge pixels in our contour segmentation problem. We can alleviate this class imbalance using appropriate loss functions as discussed in Section 2.2.2 or by dilating the contours and increasing their thickness. In Table 2.2, we evaluate the performance of instance and panoptic segmentation for different dilation rates.

Dilation Rate	AP	PQ	PQ^{Th}	SQ^{Th}	RQ^{Th}
1	24.1	46.0	30.5	73.1	40.6
2	**24.3**	**47.8**	**33.2**	**76.3**	**42.9**
3	22.6	46.6	32.0	75.6	41.7

Table 2.2: Performance of instance and panoptic segmentation on Cityscapes val dataset when different dilation rates were used to generate ground truth instance contours. Increasing the dilation rate, increases the thickness of the ground truth instance contours.

We observed that when an appropriate loss combination is used, the dilation rate does not significantly impact the performance. However, increasing the dilation rate from 2 to 3 decreases the performance. We use a dilation rate of 2 to generate our ground truth contours for all other experiments.

2.3.2.3 Refining Instance Segmentation

As discussed in Section 2.2.4, we refine our instance segmentation output using center regression results. We evaluate the effects of split and merge components in our refinement process in Table 2.3 and evaluate the effect of min instance area in Table 2.4.

Refine		Performance				
Split	Merge	AP	PQ	PQ^{Th}	SQ^{Th}	RQ^{Th}
		24.0	47.1	33.0	75.6	42.7
✓		24.2	47.7	33.1	76.1	**42.9**
✓	✓	**24.3**	**47.8**	**33.2**	**76.3**	**42.9**

Table 2.3: Evaluation of instance and panoptic segmentation on Cityscapes val dataset before and after refinement using offsets predicted by center regression results.

We observed that refining the instance segmentation using offsets predicted by center regression marginally improves instance segmentation performance. However, the refinement is critical when a broken contour can miss the boundary between two instances that can wrongly be predicted as a single instance. Similarly, an occlusion by a pole or low width object can mislead connected component labeling to interpret resulting contours as separate instances.

min Instance Area	AP	PQ	PQ^{Th}	SQ^{Th}	RQ^{Th}
1	10.0	40.6	17.6	75.5	23.1
100	21.3	46.4	31.4	75.7	40.8
300	**24.3**	**47.8**	**33.2**	**76.3**	**42.9**
500	23.6	47.0	32.7	75.5	42.4

Table 2.4: Impact of minimum instance area threshold during instance refinement on Cityscapes val dataset.

We observed that choosing an appropriate minimum instance area threshold is critical in determining our method's performance. The lower instance area allows the removal of unwanted instances generated due to artifacts in contour estimation. Such artifacts could

result from false contours around mirrors of cars, convex hulls, occlusion, etc.

2.3.2.4 Network Ablation

We experimented with different network architecture choices. We studied the impact of using a shared neck vs. separate neck layer to upsample and add features from a common feature pyramid network. We also studied how the depth of ResNet encoder impacts our performance by using ResNet-50 and ResNet-101 encoders in Table 2.5. We report the performance of semantic, instance, and panoptic segmentation networks as the change in network architecture impacts the learning of different heads. We observed that higher ResNet depth and separate necks yield better performance.

Neck	Backbone	mIoU	PQ^{St}	AP	PQ^{Th}	PQ
Shared	ResNet-50	67.5	57.4	24.3	33.2	47.8
Separate	ResNet-50	**69.6**	58.6	**25.0**	**34.0**	48.3
Shared	ResNet-101	68.4	58.5	24.7	33.4	48.1
Separate	ResNet-101	68.7	**59.3**	24.9	33.2	**48.4**

Table 2.5: Performance of semantic, instance and panoptic segmentation using different network architecture choices on Cityscapes val dataset.

2.3.3 State of the Art Comparison

In Table 2.6, we compare our methods against other semantic, instance and panoptic segmentation methods.

2.3.3.1 Comparison with Two-stage Methods

Two-stage object detection methods [24, 39, 81] dominate state of the art in the instance and panoptic segmentation. However, they have incurred additional compute costs in generating object detection followed by foreground mask generation. Mask R-CNN [39] for

Method	mIoU	PQ^{St}	AP	PQ^{Th}	PQ
Two-stage Object detection					
Mask R-CNN [39]	-	-	31.5	-	-
Weakly Supervised [81]	71.6	52.9	24.3	39.6	47.3
Panoptic-FPN [24]	74.5	62.4	32.2	51.3	57.7
UPSNet [25]	75.2	62.7	33.3	54.6	59.3
DeepSnake [30]	-	-	37.4	-	-
Instance Clustering					
Kendall et al [40] †	78.5	-	21.6	-	-
Panoptic-DeepLab [23]	78.2	-	32.7	-	60.3
Single-stage Object detection					
Poly YOLO [27]*	-	-	8.7	-	-
Others					
Deep Contour [91] †	-	-	2.3	-	-
Uhrig et al. [95]	-	-	9.9	-	-
Deep Watershed [96]	-	-	21.2	-	-
SGN [97]	-	-	29.2	-	-
Ours [ResNet-50]	69.6	58.6	25.0	34.0	48.3
Ours [ResNet-101]	68.7	59.3	24.9	33.2	48.4

Table 2.6: Comparison with other state-of-the art methods on Cityscapes dataset (val split).
† Performance reported on test split. *Evaluated on image of size 416×832.

instance segmentation on a high end GPU like Nvidia Titan X runs at ∼5 and ∼2 fps on 1024×1024 and 1024×2048 images respectively. Other two-stage methods UPSNet [25], and DeepSnake [30] are lighter compared to Mask R-CNN and operate at ∼4 fps, for instance segmentation task. When semantic segmentation task is executed in parallel with instance segmentation to compute panoptic segmentation, these methods' run time speed will further decline. These increased latencies make the two-stage object detection based methods not suitable for real-time applications. Our method with ResNet-50 encoder outputs panoptic segmentation at ∼3 fps and ∼5 fps on a mid-grade Nvidia GTX 1080 GPU (∼8.8 Tflops) on a 1024×2048 image with and without instance refinement function. We expect higher frame rates when our connected component labeling and instance refinement

functions are optimized for GPU operation instead of its current CPU based implementation.

2.3.3.2 Comparison with Instance Clustering

Kendall et al. [40] was one of the early works that used multi-task learning to simultaneously learn semantic and instance segmentation. Panoptic-DeepLab [23] recently proposed a strong baseline for center regression-based methods by exploiting the effectiveness of Atrous Spatial Pyramid Pooling (ASPP) modules. We believe that using ASPP module in our network will improve our semantic segmentation performance and eventually lead us to a better instance and panoptic segmentation results. However, ASPP modules are computationally very expensive compared to Feature pyramid networks [24]. Panoptic-DeepLab with ResNet50 achieves $\sim$5fps on Tesla V-100 SMX2 GPU ($\sim$14 Tflops).

2.3.3.3 Comparison with Single-stage Object Detection and Others

Poly YOLO [27] reported $\sim$22 fps on a 416$\times$832 image with an AP score of 8.7 while Deep Contour [91] reported $\sim$5fps on a mid grade GTX 1070 with AP score of 2.3†. Other methods like Deep Watershed [96] and SGN [97] ($\sim$0.6 fps) incur a huge computation complexity. Our methods are outperform faster methods like [27] and [91] while improving run-time efficiency compared to [96, 97].

2.4 Discussions

Two-stage methods are dominating state of the art due to the main reason that segmentation is performed on objects detected in the first stage. These objects serve as a great prior (while reducing False positives) and reduce foreground-background segmentation into a

very small subproblem. In contrast, one-stage methods aim to predict instance segmentation in a single step where no object proposal is used. Our method is a single-stage method that relies on processing the entire image to generate instance segmentation results. Relevant single-stage methods were compared against our method in Table 2.6, both in terms of computational complexity and quantitative performance. In section 2.3.3, we briefly discussed the method's significant shortcomings and how we tried to tackle them.

On the other hand, If one wants to pay more attention to larger instances over smaller instances, PQ may not be an ideal metric. During the estimation of PQ, the size of instances is not considered. For example, instances with 10×10 pixels contribute equally to the metric as instances with 1000×1000 pixels. Therefore, PQ is sensitive to false positives. Such Examples include applications in automotive driving, where nearby objects are more important than farther objects. Our method qualitatively demonstrated better performance on near-range objects compared to farther objects. In the future, we would like to compare our methods against others at different distances.

2.5 Conclusion

In this chapter, we presented a new approach to panoptic segmentation using instance contours. Our method is one of the first approaches where instance segmentation is generated as a byproduct in a semantic segmentation network. We evaluated the performance of our semantic, instance, and panoptic segmentation results on the Cityscapes dataset. We presented several ablation studies that help understand the impact of architecture and training choices that we made. We believe that our methods open a new direction in the research of instance and panoptic segmentation and serve as a baseline for contour-based methods.

CHAPTER III

Multi-stream Learning

3.1 Introduction

Semantic segmentation provides complete semantic scene understanding wherein each pixel in an image is assigned a class label. It has applications in various fields, including automated driving [98, 99], augmented reality, and medical image processing. This algorithm has recently matured in terms of accuracy, sufficient for commercial deployment due to advancements in deep learning. Most of the standard architectures use a single frame even when the algorithm is run on a video sequence. There are a strong temporal continuity and constant ego-motion of the camera for automated driving videos, which can be exploited within the semantic segmentation model. This inspired us to explore temporal based video semantic segmentation.

In this chapter, we present two types of architectures, namely Recurrent FCN (RFCN) and Multi-stream FCN (MSFCN) (as shown in Figure 3.1) inspired by FCN and Long short-term memory (LSTM) networks. Multi-stream Architectures were first introduced in [100] in which a two-stream CNN was proposed for action recognition. They were also successfully used for other applications like Optical Flow [17], moving object detection [44] and depth estimation [101]. The main motivation is to leverage temporal continuity in video streams. In RFCN, we temporally processed FCN encoders using the LSTM network. In MSFCN architecture, we combine the current and previous frames' encoder to produce a new fused encoder of the same feature map dimension. This would enable keeping the

same decoder.

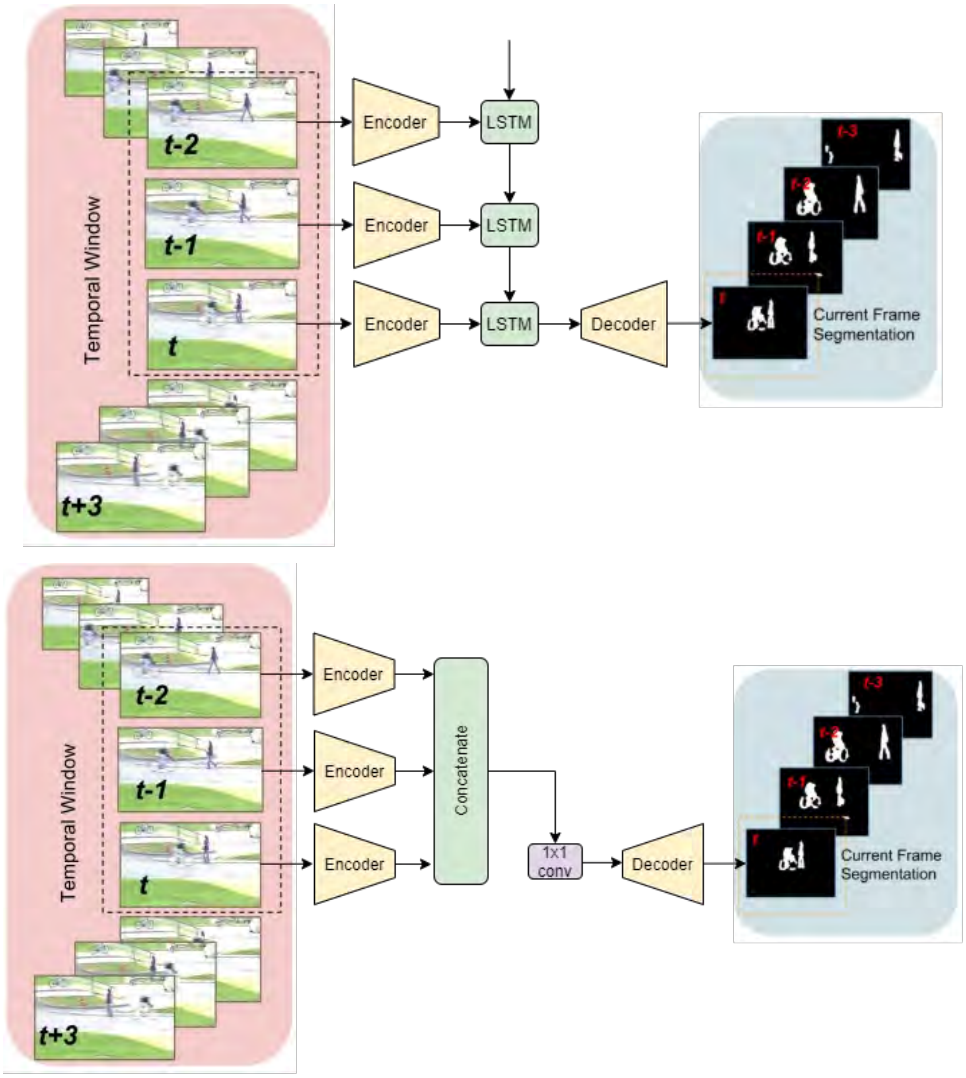


Figure 3.1: Top: Recurrent FCN, Bottom: Multi-stream FCN.

3.2 Extending Semantic Segmentation to Videos

In this section, we motivate incorporating temporal models in automated driving and explain different high-level methods to accomplish the same. Motion is a dominant cue in automated driving due to the vehicle's continuous motion on which the camera is mounted.

The objects of interest in automotive are split into static infrastructures like roads, traffic signs, etc., and dynamic objects, which are interacting like vehicles and pedestrians. The main challenges are posed due to the uncertain behavior of dynamic objects. Dense optical flow is commonly used to detect moving objects purely based on motion cues. Recently, HD maps are becoming a widely used cue that enables detecting static infrastructure, which is previously mapped and encoded. In this work, we explore temporal continuity usage to improve accuracy by implicitly learning motion cues and tracking. We discuss the various types of temporal models in Figure 3.2 which illustrates the different ways to extend image-based segmentation algorithm to videos.

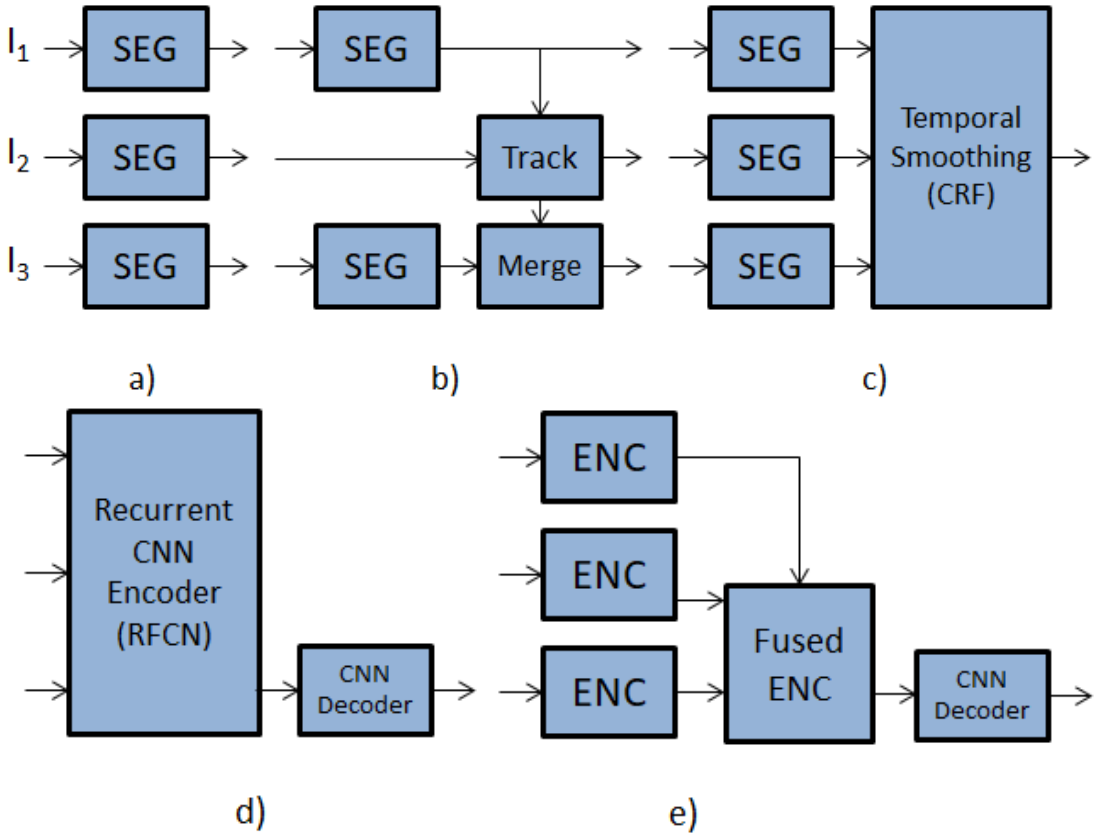


Figure 3.2: Comparison of different approaches to extend semantic segmentation to videos - a) Frame-level output b) Detect and track c) Temporal post processing d) Recurrent encoder model and e) Fused multi-stream encoder model.

3.2.1 Single Frame Baseline

Figure 3.2 (a) illustrates the typical way the detector is run every frame independently. This would be the reference baseline for comparing the accuracy of improvements by other methods.

3.2.2 Detect and Track Approach

This approach's premise is to leverage the previously obtained estimate of semantic segmentation as the next frame has only incrementally changed. This approach can significantly reduce the computational complexity as a lighter model can refine the previous semantic segmentation output for the current frame. The high level block diagram is illustrated in Figure 3.2 (b). This approach has been successfully used to detect bounding box objects where tracking could even help when the detector fails in certain frames. However, it isn't easy to model it for semantic segmentation as the output representation is quite complex. It is challenging to handle the appearance of new regions in the next frame.

3.2.3 Temporal Post Processing

The third approach is to use a post-processing filter on output estimates to smooth out the noise. Probabilistic Graphical Models (PGM) like Conditional Random Fields (CRF) are commonly used to accomplish this. The block diagram of this method is shown in Figure 3.2 (c), where recurrence is built on the output. This step is computationally complex because the recurrence operation is on the image dimension, which is large.

3.2.4 Recurrent Encoder Model

In this approach, the intermediate feature maps from the encoders are fed into a recurrent unit. The recurrent unit in the network can be an RNN, LSTM, or a GRU. Then the resulting features are fed to a decoder, which outputs semantic labels. For instance, In a ResNet50 encoder, conv5 layer features from consecutive image streams can be passed as temporal features for the LSTM network.

3.2.5 Fused Multi-stream Encoder Model

This method can be seen as a special case of the Recurrent model in some sense. But the perspective of multi-stream encoder will enable the design of new architectures. As this is the main contribution of this work, we will describe it in more detail in the next section.

3.3 Method

In this section, we discuss the details of our multi-stream networks. Multi-stream fused architectures (MSFCN-2 & MSFCN-3) concatenate the output from each encoder and fuse them via 1×1 channel-wise convolutions to obtain a fused encoder, which is then fed to the decoder. Recurrent based architecture (RFCN) uses an LSTM unit to feed the decoder.

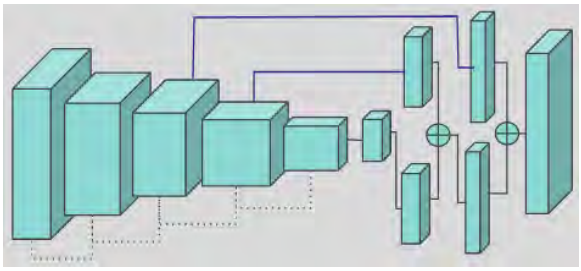


Figure 3.3: FCN: Single encoder baseline.

3.3.1 Single Stream Architecture

A fully convolution network (FCN) shown in Figure 3.3 is inspired from [5] is used as the baseline architecture. We used ResNet50 [21] as the encoder and conventional up-sampling with skip-connections to predict pixel-wise labels. Initializing model weights by pre-trained ResNet50 weights alleviates over-fitting problems as these weights result from training on a much larger dataset, namely ImageNet.

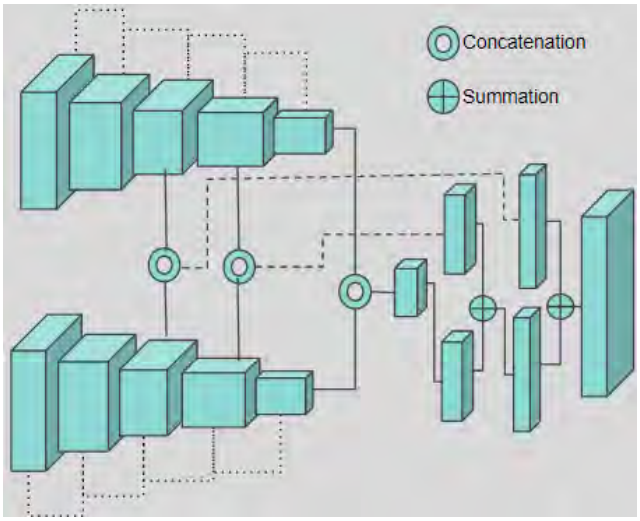


Figure 3.4: MSFCN-2: Two stream fusion architecture.

3.3.2 Multi-stream Fused Architectures

Multi-stream FCN architecture is illustrated in Fig 3.4 & 3.5. We used multiple ResNet50 encoders to construct the multi-stream architectures. Consecutive input frames are processed by multiple ResNet50 encoders independently. The intermediate feature maps obtained at three different stages (conv3, conv4, and conv5) of the encoder are concatenated and added to the decoder's up-sampling layers. MSFCN-2 is constructed using 2 encoders while MSFCN-3 uses 3 encoders. A channel-wise 1x1 convolution is applied to fuse the

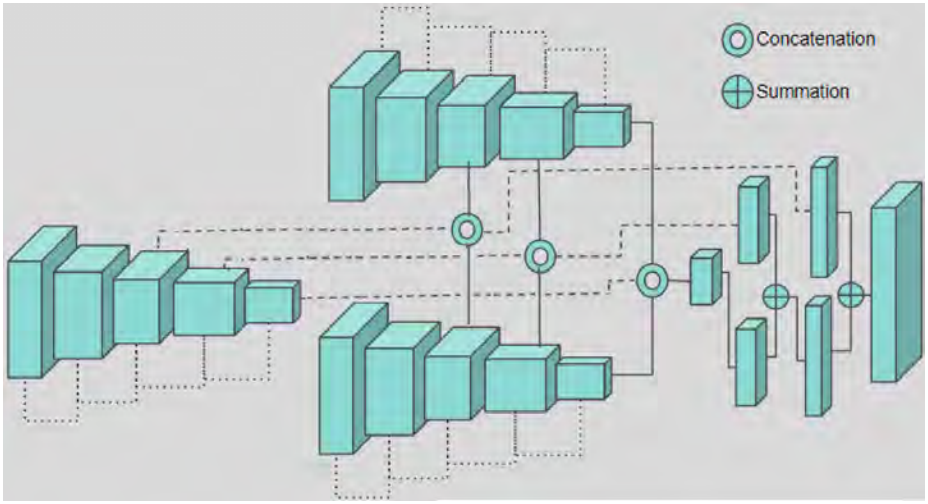


Figure 3.5: MSFCN-3: Three stream fusion architecture.

multiple encoder streams into a single one of the same dimension. This strategy will enable the usage of the same decoder.

3.3.3 Multi-stream Recurrent Architecture

A recurrent fully convolutional network (RFCN) is designed to incorporate a recurrent network into a convolutional encoder-decoder architecture. It is illustrated in Figure 3.6. We use the generic recurrent unit LSTM, which can specialize to simpler RNNs and GRUs. LSTM operates over the previous N frames' encoder and produces a filtered encoder of the same dimension, which is then fed to the decoder.

The generic form of multi-stream architectures has different weights for the different encoders. In Figure 3.2 (e), the three encoders can be different, and they have to be re-computed each frame. Thus the computational complexity of the encoder increases by a factor of three. However, if the weights are shared between the encoders, there is no need to recompute each frame. One encoder feature extraction per frame suffices, and a combination of previously computed encoders computes the fused encoder. This weight sharing

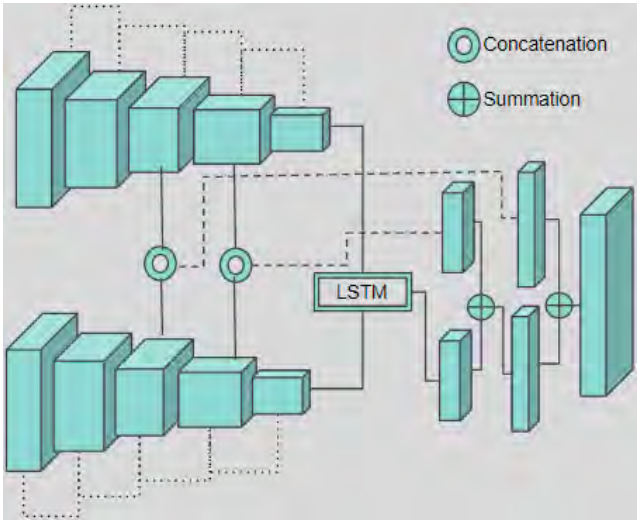


Figure 3.6: RFCN-2: Two stream LSTM architecture.

approach drastically brings down the complexity with negligible additional computation relative to the single-stream encoder.

3.4 Experiments and Results

This section explains the experimental setting, including the datasets used, training algorithm details, etc., and discusses the results.

3.4.1 Experimental Setup

In most datasets, the frames in a video sequence are sparsely sampled temporally to have better diversity of objects. Thus consecutive video frames are not provided for training our multi-stream algorithm. Synthetic datasets have no cost for annotation, and ground truth annotation is available for all consecutive frames. Hence, we used the synthetic autonomous driving dataset SYNTHIA [74] for our experiments. We also used DAVIS [76], and SegTrack V2 [75] which provides consecutive frames. They are not automotive

datasets but realistic.

We implemented the different multi-stream architectures using Keras [102]. We used ADAM optimizer as it provided faster convergence. The maximum order (number of consecutive frames) used in training is three (MSFCN-3) because of the limitation of memory needed for training. Categorical cross-entropy is used as a loss function for the optimizer. The maximum number of training epochs is set to 30, and early stopping with patience of 10 epochs monitoring the gains is added. Mean class IoU and per-class IoU were used as accuracy metrics. All input images were resized to 224x384 because of the memory requirements needed for multiple streams.

3.4.2 Ablation Studies

We performed four sets of experiments summarized in four tables. Qualitative results are provided in Figure 3.9 for KITTI, Figure 3.10 for DAVIS and Figure 3.8 for SYNTHIA.

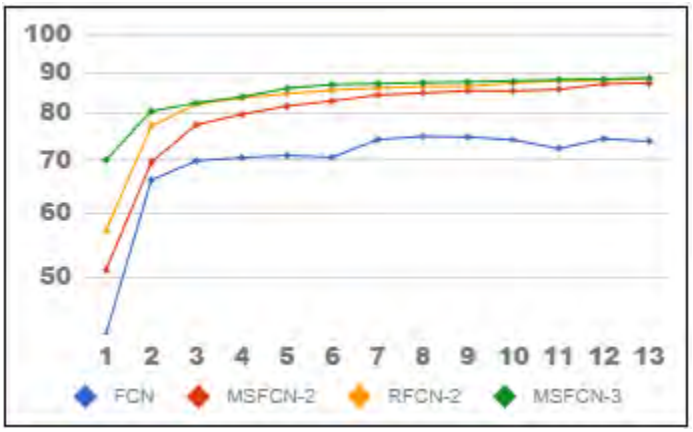


Figure 3.7: Accuracy over epochs for SYNTHIA dataset.

3.4.2.1 Temporal Depth

Firstly, we wanted to evaluate different orders on multi-stream and understand the impact. We also wanted to understand the implications for high speed and medium speed scenarios. SYNTHIA dataset was used for this experiment as it had separation of various speed sequences, and it was also a relatively larger dataset. Table 3.1, Two-stream networks provided a considerable increase in accuracy compared to the baseline. MSFCN-2 provided an accuracy improvement of 8% for Highway and 14% for City sequence. RFCN-2 provided a slightly better accuracy relative to MSFCN-2. MSFCN-3 provided a marginal improvement over MSFCN-2, and thus we did not explore higher orders. We show the performance of methods over different training epochs on the SYNTHIA dataset in Figure 3.7. We also present qualitative results in Figure 3.8.

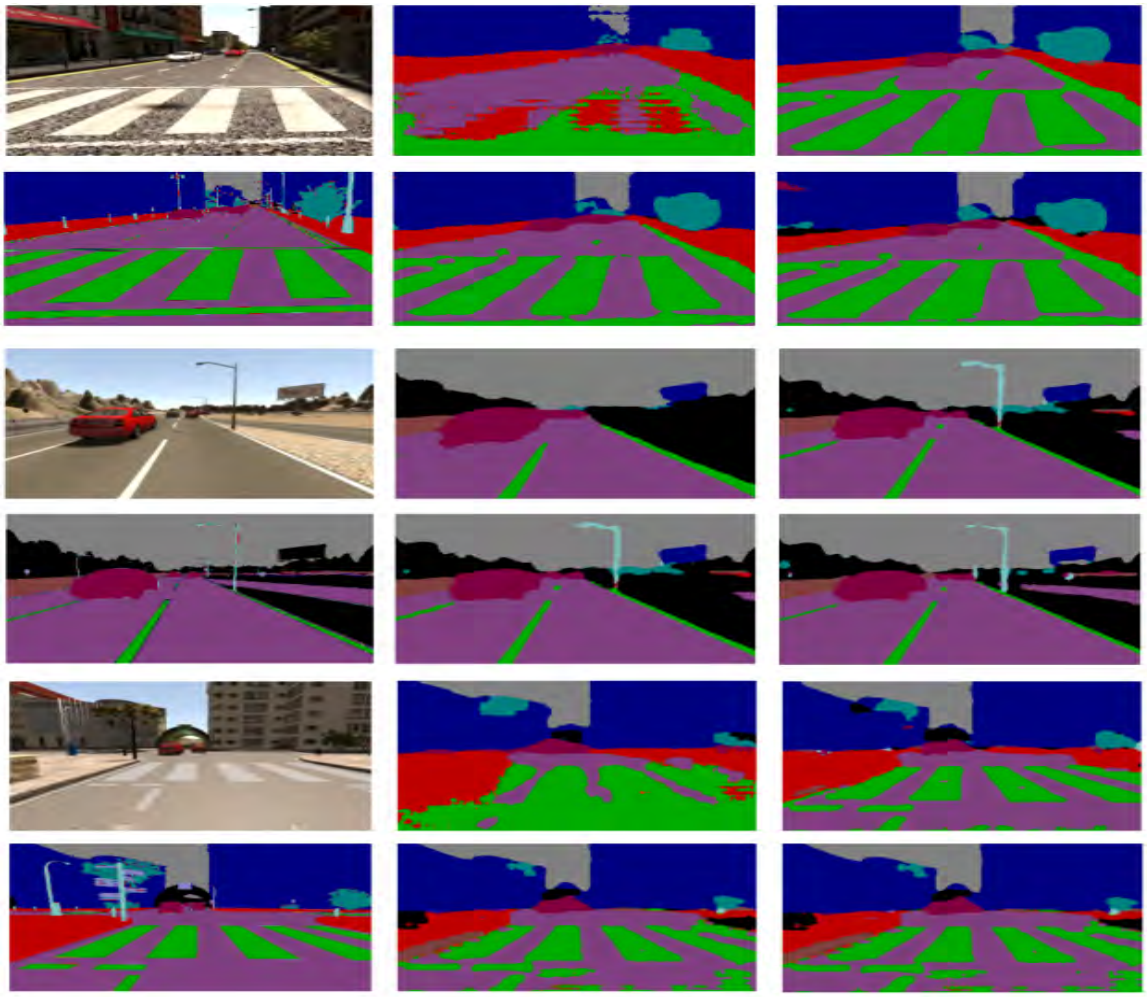


Figure 3.8: Qualitative results of experiments with SYNTHIA dataset. Left to right: RGB image, single encoder (FCN), two stream encoder (MSFCN-2), ground truth, two stream encoder + LSTM (RFCN-2) and three stream encoder (MSFCN-3).

Highway										
Architecture	**Mean IoU**	**Sky**	**Building**	**Road**	**Sidewalk**	**Fence**	**Vegetation**	**Pole**	**Car**	**Lane**
FCN	85.42	0.91	0.67	0.89	0.02	0.71	0.79	0.01	0.81	**0.72**
MSFCN-2	93.44	0.92	0.66	0.94	0.28	0.85	0.78	0.11	0.82	0.71
RFCN-2	94.17	**0.93**	**0.71**	0.95	**0.31**	0.82	**0.83**	**0.13**	**0.87**	0.7
MSFCN-3	**94.38**	**0.93**	0.69	**0.96**	**0.31**	**0.87**	0.81	0.12	**0.87**	**0.72**
City										
Architecture	**Mean IoU**	**Sky**	**Building**	**Road**	**Sidewalk**	**Fence**	**Vegetation**	**Pole**	**Car**	**Lane**
FCN	73.88	**0.94**	**0.94**	0.72	0.78	0.34	0.54	0	0.69	0.56
MSFCN-2	87.77	0.87	**0.94**	0.84	**0.83**	**0.68**	0.64	0	**0.8**	**0.8**
RFCN-2	88.24	0.91	0.92	**0.87**	0.78	0.56	**0.67**	0	**0.8**	0.74
MSFCN-3	**88.89**	0.88	0.89	0.86	0.74	0.64	0.53	0	0.71	0.72

Table 3.1: Semantic segmentation results on SYNTHIA sequences. We split the test sequences into two parts, one is Highway for high speeds and the other is City for medium speeds.

3.4.2.2 Shared Weights

We reduced our experiments to MSFCN-2 and RFCN-2, but we added shared weight versions of the same. In Table 3.2 MSFCN-2 provided an accuracy improvement of 11% on KITTI dataset[77], and the shared weight version only lagged slightly. We demonstrate qualitative results of our methods on the KITTI dataset in Figure 3.9.

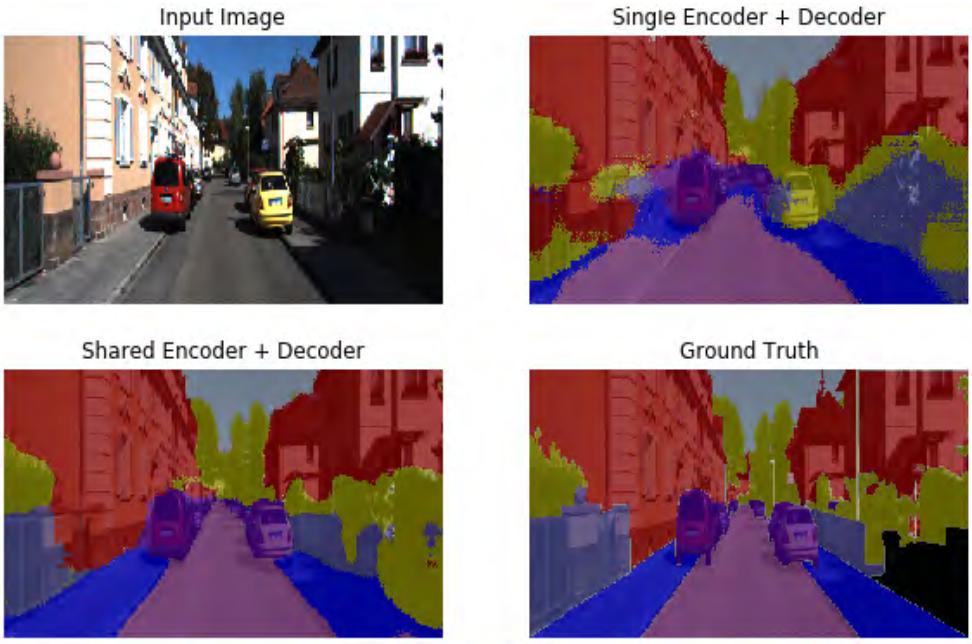


Figure 3.9: Results on KITTI dataset.

We repeated the experiments of the same networks used in Table 3.2 on a more extensive SYNTHIA sequence and present in Table 3.3. MSFCN-2 provided an accuracy improvement of 6% in Mean IoU. MSFCN-2 with shared weights lagged by 1%. RFCN-2 versions had slightly lesser accuracy compared to their MSFCN-2 counterparts with and without weight sharing.

Architecture	NumParams	Mean IoU	Sky	Building	Road	Sidewalk	Fence	Vegetation	Car	Sign
FCN	23,668,680	74.00	46.18	86.50	80.60	69.10	37.25	81.94	74.35	35.11
MSFCN-2 (shared)	23,715,272	85.31	47.89	91.08	**97.58**	88.02	**62.60**	92.01	**90.26**	58.11
RFCN-2 (shared)	31,847,828	84.19	**50.20**	**93.74**	94.90	88.17	59.73	87.73	87.66	55.55
MSFCN-2	47,302,984	**85.47**	48.72	92.29	96.36	90.21	59.60	**92.43**	89.27	**70.47**
RFCN-2	55,435,540	83.38	44.80	92.84	91.77	**91.67**	58.53	86.01	87.25	52.87

Table 3.2: Semantic segmentation Results on KITTI video sequence.

Architecture	Mean IoU	Sky	Building	Road	Sidewalk	Fence	Vegetation	Pole	Car	Sign	Pedestrian	Cyclist	Lane
FCN	84.08	97.2	92.97	87.74	81.58	34.44	62	1.87	72.75	0.21	0.01	0.33	93.08
MSFCN-2 (shared)	88.88	97.08	93.14	93.58	**86.81**	47.47	75.11	46.78	**88.22**	0.27	**32.12**	2.27	95.26
RFCN-2 (shared)	88.16	96.85	91.07	**94.17**	85.62	28.29	**83.2**	**47.28**	87.6	**19.12**	16.89	**3.01**	93.97
MSFCN-2	**90.01**	**97.34**	**95.97**	93.14	86.76	73.52	73.63	35.02	87.86	3.62	27.57	1.11	**95.35**
RFCN-2	89.48	97.15	94.01	93.76	85.88	**76.26**	70.35	39.86	87.5	8.16	28.05	1.28	94.67

Table 3.3: Semantic segmentation Results on SYNTHIA video sequence.

3.4.2.3 Non Automotive Datasets

As most automotive semantic segmentation datasets do not provide consecutive frames for temporal models, we tested in real non-auomotive datasets namely SegTrack [75] and DAVIS [76] in Table 3.4. MSFCN-3 provided an accuracy improvement of 11% in SegTrack [75] and 6% in DAVIS [76]. This demonstrates that the constructed networks provide consistent improvements in various datasets.

Figure 3.10: Results over DAVIS dataset. Left to right: RGB image, ground truth, single encoder (FCN), two stream encoder (MSFCN-2), two stream encoder + LSTM (RFCN-2), three stream encoder (MSFCN-3).

3.5 Discussions

We have chosen a moderately sized-based encoder, namely ResNet50, and we will be experimenting with various sizes like ResNet10, ResNet101, etc., for future work. In general, multi-stream provides a significant improvement in accuracy for this moderately

SegTrack v2	
Architecture	**Mean IoU**
FCN	83.82
MSFCN-3	**94.61**
DAVIS	
FCN	77.64
MSFCN-3	**83.42**
BVS[103]	66.52
FCP[104]	63.14

Table 3.4: Comparison of multi-stream network with its baseline counterpart on SegTrack and DAVIS.

sized encoder. The improvements might be more extensive for smaller networks, which are less accurate. With a shared weights encoder, an increase in computational complexity is minimal. However, it increases memory usage and memory bandwidth significantly due to additional encoder feature maps' maintenance. It also increases the latency of output by 33 ms for a 30 fps video sequence. From visual inspection, the improvements are seen mainly in refining the boundaries and detecting smaller regions. It is likely due to the temporal aggregation of feature maps for each pixel from past frames.

3.5.1 MSFCN vs FCN

The single-frame based FCN suffers to segment weaker classes like poles and objects at further distances. Table 3.3 shows IoU metrics for weaker classes like Pole, Fence, and Sidewalk have significantly improved in the case of multi-stream networks compared to single-stream FCN. Fig 4 visually demonstrates that the temporal encoder modules help preserve the small structures and boundaries in segmentation.

3.5.2 MSFCN-2 vs MSFCN-3

The increase in the temporal information has increased the performance of the semantic segmentation. But this brings an extra latency for real-time applications.

3.5.3 MSFCN-2 vs RFCN

The recurrent encoder feature fusion has shown quite a decent improvement for a multi-stream network compared to the feature concatenation technique. It is also observed that the recurrent networks helped in preserving the boundaries of the weaker classes like poles and lane markings. However, RFCN demands more parameters and takes longer training time for convergence, as shown in Figure 3.7.

3.5.4 Weight Sharing

In most of the experiments, MSFCN-2 with shared weights provided an excellent improvement over the baseline, and its performance deficit relative to the generic MSFCN-2 is usually small, around 1%. However, the shared weights version provides a drastic improvement in computational complexity, as shown by the number of parameters in Table 3.2. Shared weights MSFCN-2 has a negligible increase in the number of parameters and computational complexity, whereas the generic MSFCN-2 has double the parameters. Thus it is vital to make use of weight sharing.

3.6 Conclusion

In this chapter, we designed and evaluated two video semantic segmentation architectures, namely Recurrent FCN (RFCN) and Multi-stream FCN (MSFCN) networks, to exploit temporal information. We implemented three architectures, namely RFCN-2, MSFCN-

2, and MSFCN-3, using ResNet50 as a base encoder and evaluated on SYNTHIA sequences. We obtain promising improvements of 9% and 15% for Highway and New York-like city scenarios over the baseline network. We also tested MSFCN-3 on real datasets like SegTrack V2 and DAVIS datasets where 11% and 6% accuracy improvement was achieved, respectively. We also explored weight sharing among encoders for better efficiency and produced an improvement of 11% and 5% for KITTI and SYNTHIA using MSFCN-2 with roughly the same complexity as the baseline encoder. In future work, we plan to explore more sophisticated encoder fusion techniques.

CHAPTER IV

Multi-task Learning

4.1 Introduction

Convolutional Neural Networks (CNNs) are successfully used for important automotive visual perception tasks, including object recognition, motion, depth estimation, visual SLAM, etc. However, these tasks are typically independently explored and modeled. In this chapter, we present a joint multi-task network design for learning several tasks simultaneously. Our primary motivation is the computational efficiency achieved by sharing the expensive initial convolutional layers between all tasks. Indeed, the main bottleneck in automated driving systems is the limited processing power available on deployment hardware. There is also some evidence for other benefits in improving accuracy for some tasks and easing development effort. It also offers scalability to add more tasks leveraging existing features and achieving better generalization.

Multi-task learning [56, 105, 106] has been gaining significant popularity over the past few years as it has proven to be very efficient for embedded deployment. Multiple tasks like object detection, semantic segmentation, depth estimation, etc., can be solved simultaneously using a single model. A typical multi-task learning framework consists of a shared encoder coupled with multiple task-dependent decoders. An encoder extracts feature vectors from an input image after series of convolution and pooling operations. Individual decoders then process these feature vectors to solve different problems. [57] is an example of three task-specific decoders used for scene classification, object detection, and

road segmentation of an automotive scene. The main advantages of multi-task learning are improved computational efficiency, regularization, and scalability. [107] discusses other benefits and applications of multi-task learning in various domains.

4.2 Universality of CNN Features

The universality of CNN features allows them to transfer to other tasks when previously learned for a different task, a widespread practice known as Transfer Learning. For example, weights obtained from a pre-trained network on the Imagenet [94] classification task is usually used as an initialization step for fine-tuning more complex vision tasks. Transfer learning allows knowledge transfer from one task to another, mostly when the data lacks the targeted task. Indeed, the low-level features are mostly task agnostic to be reused by other vision tasks. But how far can this practice be applied? To better understand this concept of transfer learning, Zamir et al. [108] characterized the relationship between several visual tasks and found an order of dependency for task transfer learning. Other papers [68, 109] demonstrated that universality applies to different domains and different modalities. Kaiser et al. [110] jointly learned tasks in other modalities (speech recognition, image classification, and text translation) using a single model. Therefore, feature sharing is possible for very different tasks across a shared network.

Learning a universal representation to solve multiple tasks is crucial in developing efficient algorithms in terms of performance, generalization, and computational complexity instead of having several separate networks for different tasks. Recently, several works [56, 57, 106] proposed a joint multi-task network to solve several tasks simultaneously. However, the universality of CNN features is only possible up to a certain extent. One limitation of the CNN is their ease of specialization to a domain or task, preventing their

generalization to other domains or tasks. To overcome this limitation, Bilen et al. [109] normalized the network's information using domain-specific scaling factors or normalization layers. Rebuffi et al. [68] built universal parametric families of networks for efficient sharing of parameters across domains. Tamaazousti et al. [111] proposed universalizing methods to force a network to learn a representation capable of handling various tasks. These signs of progress suggest CNN features offer a strong possibility to represent multiple tasks through a unified model.

4.2.1 Adaptation

Universal feature representation can fail for several reasons. For example, the model's representation capacity for tasks with different complexities can be insufficient or oversized. In some cases, too generic representations might prevent the specialization to a certain task or domain. The complexity of automotive tasks can have large variability in practice, and thus it is necessary to have adaptation mechanisms to obtain optimal feature representation. Designing a balanced dataset satisfying the complexities of different tasks is challenging. In general, complex tasks require bigger models, while simple tasks need smaller models. An undersized representation for a complex task can be compensated by augmenting specialized feature maps for more complex tasks. In this case, these augmented features are used only for complex tasks. On the other hand, an oversized representation for a simpler task can be avoided by pruning to simplify the model. Pruning methods compress the model by removing redundant filters and keeping only the most relevant ones [112, 113]. Here we propose to perform task-specific feature pruning for simpler tasks. Thus a shared model constructed with these adaptations can then be effectively utilized by tasks with varying complexities.

4.3 Pros and Cons of Multi-task Learning

4.3.1 Pros

The main advantage of a unified model is improving computational efficiency. Say there are two problems with two equivalent independent networks utilizing 50% of available processing power. A unified model with 30% sharing across the two networks can offer 15% of additional resources to each network for computing a slightly larger problem. This approach allows unified models to offer scalability for adding new tasks at a minimal computation complexity. On the other hand, these models reduce development effort and training time as shared layers minimize the need to learn multiple sets of parameters in different models. Unified models learn features jointly for all tasks, making them robust to over-fitting by acting as a regularizer, as demonstrated in various multi-task networks [56, 57, 106].

Computational power in embedded systems for deploying automated driving solutions is rapidly growing. In particular, there are specialized accelerators available for CNNs. Convolution is the main compute-intensive operation in a CNN, offering heavy parallelism suitable for specialized hardware. The majority of the hardware vendors for automated driving have custom hardware accelerators for CNNs. For example, Nvidia Xavier [114] provides $\sim$30 TOPS (Tera Operations per second) for CNNs. Relatively, compute power available for general-purpose processing is much lower, and the trend shows that it will reduce even further. Thus, even if a classical algorithm is more optimal for calibration tasks, a CNN-based approach will be a more efficient mapping to specialized hardware.

4.3.2 Cons

In the case of separate models, the algorithms are entirely independent. This could make the dataset design, architecture design, tuning, hard negative mining, etc., simpler and easier to manage. Debugging a unified model can be quite challenging. These models are often less fault-tolerant since features are shared for all tasks leading to a single point of failure. Such failures in learning features for a particular scenario might negatively impact other tasks. This is often called negative transfer in multi-task learning. Another practical disadvantage is that unified models assume a fixed input format for all tasks. Some tasks might need a different setting like camera field-of-view, color format, or pixel resolution.

4.4 Unified Visual Perception Model

Figure 4.1 illustrates a simple unified model architecture for five main automated driving tasks. We refer to the unified model's shared layers as CNN encoder and task-dependent layers as decoders throughout this paper. It is straight forward to add other tasks like calibration or depth estimation via additional task-specific decoders. This architecture has a shared CNN encoder and multiple parallel task-dependent decoders.

In this unified model, object detection decoders like YOLO [115], SSD [37] can be used to predict bounding boxes and categories of objects, while segmentation decoder like FCN8 [5] can be used to perform pixel-wise semantic segmentation. Motion and depth decoders can be constructed to learn simpler representations for generic object detection. A motion decoder can perform binary segmentation of moving objects and depth decoder to estimate variable height stixels of static objects. Finally, a localization decoder can predict the pose of a camera similar to PoseNet [116]. Multinet by Teichmann et al. [57], and fast scene understanding by Neven et al. [106] share similar design but for fewer tasks.

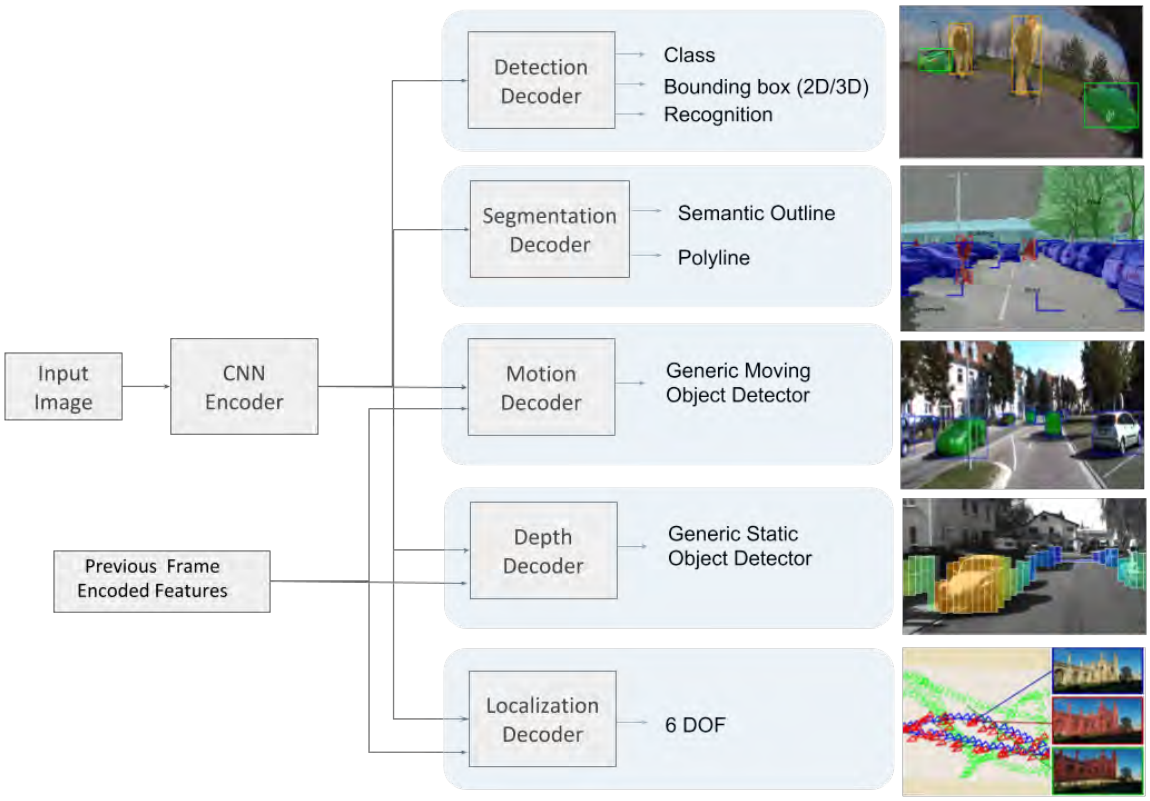


Figure 4.1: Unified model for the important visual perception tasks in automated driving.

To facilitate joint training of the unified model, tasks of varying complexities and datasets must be balanced. It is challenging to collect annotated data for all tasks. To alleviate this problem in a heterogeneous dataset, UberNet [56] proposes an asynchronous variant of backpropagation where training data are sequentially read and task-specific parameters are updated only after accumulating sufficient training examples for this particular task.

4.4.1 Two Task Model

There are many challenges in getting the full multi-task learning network presented in Section 4.4 implemented. Firstly, no datasets provide simultaneous annotation for all the

five tasks proposed in Fig. 4.1. Secondly, there are practical limitations with GPU memory for training a complex model. Thus we implement a two-task model as a first step towards building a unified visual perception model.

We implemented a two task model with three segmentation classes (background, road, sidewalk) and three object classes (car, person, cyclist). We built a small encoder with ten layers and residual connections to enable feasible deployment on a low power embedded system. This encoder is fully shared between the two tasks. FCN8 [5] and YOLO [115] are used as the decoders for semantic segmentation and object detection. The weight parameters are randomly initialized. Semantic segmentation loss (L_{seg}) is defined by the average of pixel-wise cross-entropy for each predicted label and ground-truth label. Object detection loss (L_{det}) is defined as the sum of categorical cross-entropy (between object label and prediction) and squared loss of average precision for object localization. We used ADAM optimizer as it provided faster convergence, with a learning rate of 0.0005. The maximum number of training epochs is set to 30, and early stopping with the patience of 3 epochs monitoring the gains is added. Input images were resized to 1280x384 to minimize the memory requirements needed for multiple tasks.

The total loss for the two-task unified model is a weighted sum of the task losses:

$$L_{total} = w_{seg} * L_{seg} + w_{det} * L_{det} \tag{4.1}$$

We trained and evaluated the two-task unified model (TTM) and the two single-task models (STM) on different driving datasets: two publicly available datasets KITTI [77], and Cityscapes [78]. In our experiments, we refer:

- STM$_{seg}$ is the single-task model for segmentation
- STM$_{det}$ the single-task model for detection.

- TTM the two-task model with $w_{det} = 1$ and $w_{seg} = 1$
- TTM_{10} the two-task model with $w_{det} = 1$ and $w_{seg} = 10$
- TTM_{100} the two-task model with $w_{det} = 1$ and $w_{seg} = 100$

Mean class IoU (Intersection over Union) and per-class IoU were used as accuracy metrics for semantic segmentation, mean average precision (mAP), and per-class average precision for object detection.

Datasets	Metrics	STM_{seg}	STM_{det}	TTM	TTM_{10}	TTM_{100}
KITTI	JI background	0.9706		0.9621	0.9663	0.9673
	JI road	0.8603		0.8046	0.8418	0.8565
	JI sidewalk	0.6387		0.5045	0.5736	0.6277
	mean IOU	**0.8232**		**0.757**	**0.7939**	**0.8172**
	AP car		0.801	0.7932	0.7746	0.7814
	AP person		0.469	0.5337	0.518	0.468
	AP cyclist		0.5398	0.4928	0.5107	0.5844
	mean AP		**0.6033**	**0.6066**	**0.6011**	**0.6112**
Cityscapes	JI road	0.9045		0.8273	0.8497	0.8815
	JI sidewalk	0.5434		0.3658	0.4223	0.4335
	JI building	0.7408		0.6363	0.6737	0.6947
	JI vegetation	0.8085		0.6949	0.7417	0.7363
	JI sky	0.7544		0.6228	0.652	0.6873
	JI person+rider	0.3916		0.3225	0.3218	0.3613
	JI car	0.695		0.5237	0.5918	0.6579
	JI bicycle	0.3906		0.2911	0.4123	0.3506
	mean IOU	**0.5971**		**0.4918**	**0.5213**	**0.5555**
	AP car		0.3691	0.411	0.398	0.3711
	AP person		0.1623	0.1931	0.1694	0.1845
	AP bicycle		0.1279	0.1898	0.1422	0.1509
	mean AP		**0.2198**	**0.2647**	**0.2365**	**0.2355**
	Params (Million)	**4.91**	**4.92**	**4.93**	**4.93**	**4.93**

Table 4.1: Comparison study: Single-task vs two-task, JI: Jaccard index, AP: Average precision. IOU: Intersection over union.

Table 4.1 summarizes the obtained results for STM networks and TTM networks on

KITTI and Cityscapes datasets. This is intended to provide a baseline accuracy for incorporating more complex multi-task learning techniques. We compare a segmentation network (STM_{seg}) and a detection network (STM_{det}) to a two task network performing segmentation and detection (TTM, TTM_{10} and TTM_{100}). We tested three configurations of the multi-task loss, the first one (TTM) uses a simple sum of the segmentation loss and detection loss ($w_{seg} = w_{det} = 1$). The two other configurations TTM_{10} and TTM_{100}, use a weighted sum of the task losses where the segmentation loss is weighted with a weight $w_{seg} = 10$ and $w_{seg} = 100$ respectively. This compensates for task loss scaling: the segmentation loss is 10-100 times higher than the detection loss during the training.

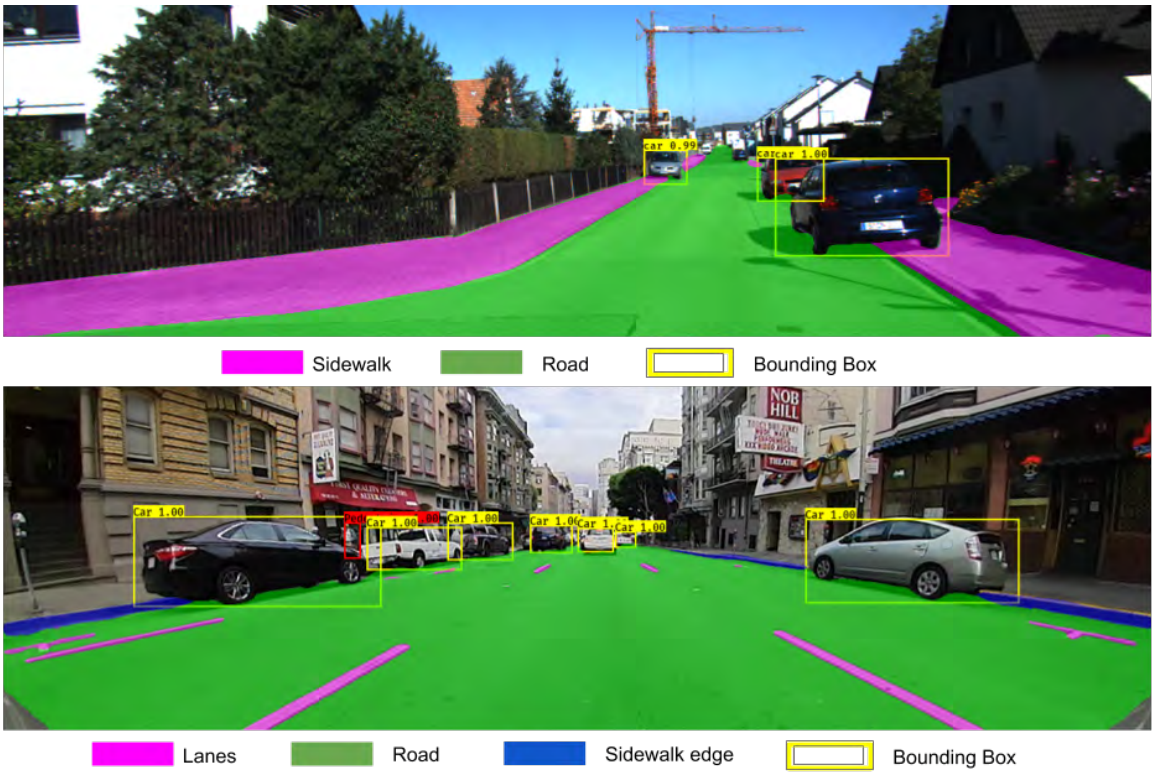


Figure 4.2: Qualitative results of two-task model performing segmentation and detection.

Figure 4.2 illustrates the qualitative output of the two-task network. TTM outperforms STM in object detection but has a slight degradation in segmentation accuracy on both

KITTI and Cityscapes datasets. This experiment shows the capacity of the unified model TTM to learn multiple tasks with similar accuracies compared to STM having two times fewer parameters but only by choosing the appropriate task loss weighting. STM_{seg} and STM_{det} would require 4.91M and 4.92M parameters (9.83M together) while TTM would only require 4.93M parameters. This shows that we have a drastic gain in terms of memory and computational efficiency.

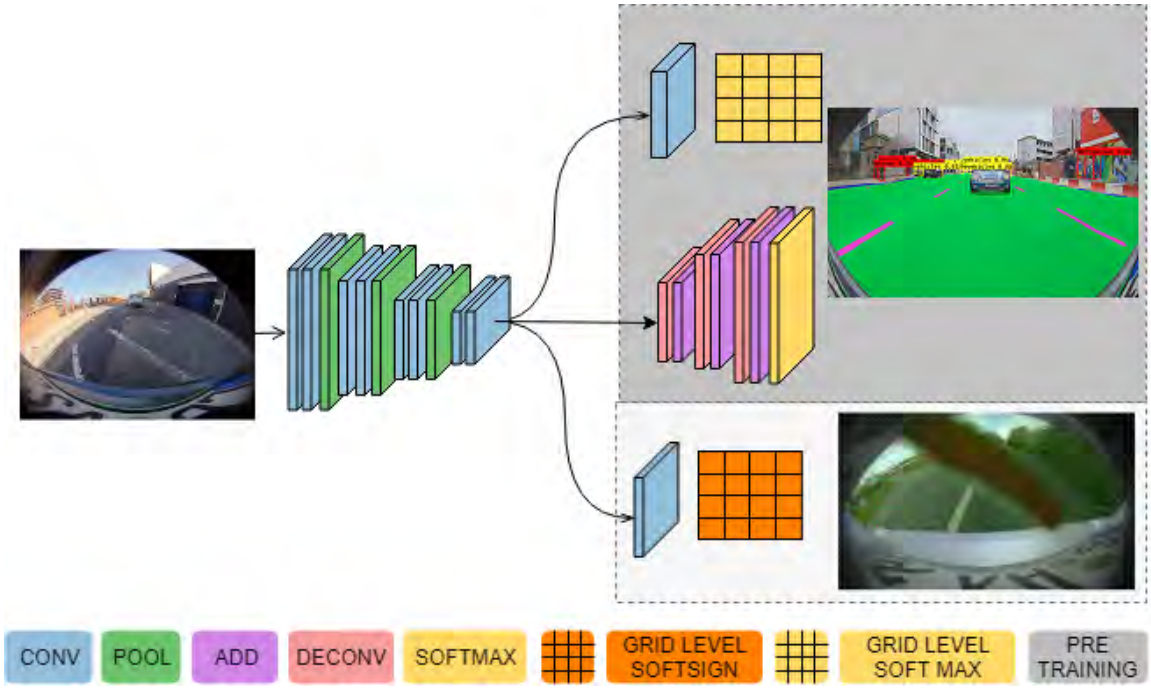


Figure 4.3: Illustration of three-task model architecture comprising of object detection, semantic segmentation and soiling detection tasks.

4.4.2 Three Task Model

We implemented a three-task model, having a shared encoder and three independent decoders that perform joint semantic segmentation, object detection, and soiling detection as shown in Figure 4.3. Semantic segmentation decoder and object detection decoder are designed similar to the two-task model. The soiling detection decoder outputs the presence

of external contamination on the camera lens, providing classification per tile for obtaining the localization of soiling in the image.

Datasets	Metrics	STM_{seg}	STM_{det}	STM_{soil}	TTM_{3task}
Segmentation [53]	JI road	0.9574			0.9514
	JI lane	0.6517			0.6424
	JI curb	0.5960			0.5850
	mean IOU	**0.7350**			**0.7263**
Object Detection [53]	AP Vehicle		0.6910		0.7016
	AP person		0.3620		0.3609
	AP cyclist		0.3682		0.3817
	mean AP		**0.4737**		**0.4814**
Soiling Detection [53]	TPR			0.5581	0.5532
	FPR			0.1432	0.1443

Table 4.2: Comparison study: Single-task vs. three-task models.

We treat the camera soiling detection task as a mixed multilabel-categorical classification problem focusing on a classifier, which jointly classifies a single image with a binary indicator array, where each 0 or 1 corresponds to a missing or present class respectively and simultaneously assigns a categorical label. The classes to detect are $\{\text{opaque}, \text{transparent}\}$. Typically, opaque soiling arises from mud and dust, and transparent soiling arises from water and ice.

We evaluate the performance of the our three-task model on the WoodScape dataset [53] comprising $10,000$ images. All input images were resized to 1280×384 because of memory requirements needed for multiple tasks. Table 4.2 summarizes the obtained results for the single-task (STM) independent networks and three-task (TTM) networks on the WoodScape dataset.

4.5 Conclusion

CNN's have become the standard model for semantic tasks like object detection and semantic segmentation, geometric tasks like depth estimation and visual SLAM. This brings an opportunity for CNNs to become a unifying model for all visual perception tasks for automated driving. In this chapter, we argue for moving towards a unified model and use current literature to propose how to achieve it. We also discuss the pros and cons of having a unified model. Finally, we perform experiments on a simpler scenario with two, three tasks and demonstrate results to support our argument.

CHAPTER V

Auxiliary Learning

5.1 Introduction

Learning a side or auxiliary task jointly during the training phase to enhance the main task's performance is usually referred to as auxiliary learning. Auxiliary learning is similar to multi-task learning, except the auxiliary task is nonoperational during inference. This auxiliary task is typically selected to have much larger annotated data to act as a regularizer for the main task. In [117] semantic segmentation is performed using auxiliary tasks like time, weather, etc. In [118], end2end speech recognition training uses auxiliary task phoneme recognition for the initial stages. [119] uses unsupervised aux tasks for audio-based emotion recognition. It is often believed that auxiliary tasks can help focus attention on specific parts of the input. Predictions of road characteristics like markings as an auxiliary task in [55] to improve the main task for steering prediction is one instance of such behavior.

Figure 5.1 illustrates auxiliary tasks in a popular automated driving dataset KITTI. It contains various output tasks like dense optical flow, depth estimation, and visual SLAM. It also contains meta-data like steering angle, location, and external condition. These meta-data comes for free without any annotation task. Depth could be obtained for free using a Velodyne depth map or disparity from a stereo pair of cameras.

This chapter explores an alternate approach of leveraging the annotations of monocular depth estimation to improve semantic segmentation. We present adaptive task loss weight-

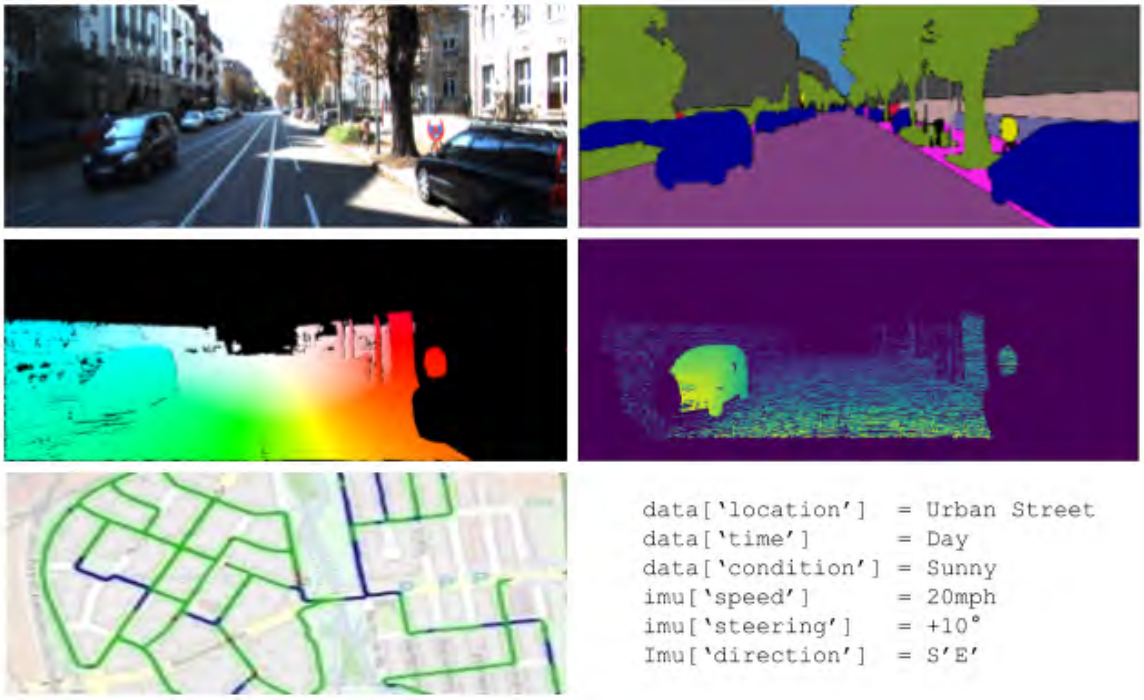


Figure 5.1: Illustration of several auxiliary visual perception tasks in an automated driving dataset KITTI. First row shows RGB and semantic segmentation, second row shows dense optical flow and depth, third row shows visual SLAM and meta-data for steering angle, location and condition.

ing techniques to address scale issues in multi-task loss functions, which become more crucial in auxiliary tasks.

5.2 Motivation

The main challenge for constructing large datasets for semantic segmentation is that the pixel-wise annotation is very labor-intensive. Annotation for semantic segmentation is a tedious and expensive process. An average experienced annotator takes anywhere around 10 to 20 minutes for one image, and it takes three iterations for correct annotations. This process limits the availability of large scale accurately annotated datasets. Popular semantic segmentation automotive datasets like CamVid [120], Cityscapes [78], KITTI

[77] are relatively smaller when compared to classification datasets like ImageNet [94]. Synthetic datasets like Synthia [74], Virtual KITTI [121], Synscapes [122] offer larger annotated synthetic data for semantic segmentation. Efforts like Berkley Deep Drive [123], Mapillary Vistas [124] and Toronto City [125] have provided larger datasets to facilitate training a deep learning model for segmentation but are expensive to construct.

There is a lot of research to reduce the sample complexity of segmentation networks by incorporating domain knowledge and other cues wherever possible. One way to overcome this is via using synthetic datasets and domain adaptation techniques [126]. Another way is to use multiple clues or annotations to learn efficient representations for the task with limited or expensive annotations [117].

5.3 Methods

Semantic segmentation and depth estimation have common feature representations. Joint learning of these tasks have shown significant performance gains in [127], [8], [128], [129] and [130]. Learning underlying representations between these tasks help the multi-task network alleviate the confusion in predicting semantic boundaries of depth estimation. Inspired by these papers, we present a multi-task network with semantic segmentation as the main task and depth estimation as an auxiliary task. As the accuracy of the auxiliary task is not important, weighting its loss function appropriately is important.

5.3.1 Architecture Design

Our network takes input RGB image and outputs semantic, and depth maps together. Figure 5.2 shows two task-specific decoders coupled to a shared encoder to perform semantic segmentation and depth estimation. The shared encoder is built using ResNet-50

[21] by removing the fully connected layers from the end. The encoded feature vectors are now passed to two parallel stages for independent task decoding. Semantic segmentation decoder is constructed similar to FCN8 [5] architecture with transposed convolutions, up-sampling, and skip connections. The final output is made up of a softmax layer to output probabilistic scores for each semantic class. Depth estimation decoder is also constructed similar to segmentation decoder, except the final output is replaced with a regression layer to estimate scalar depth.

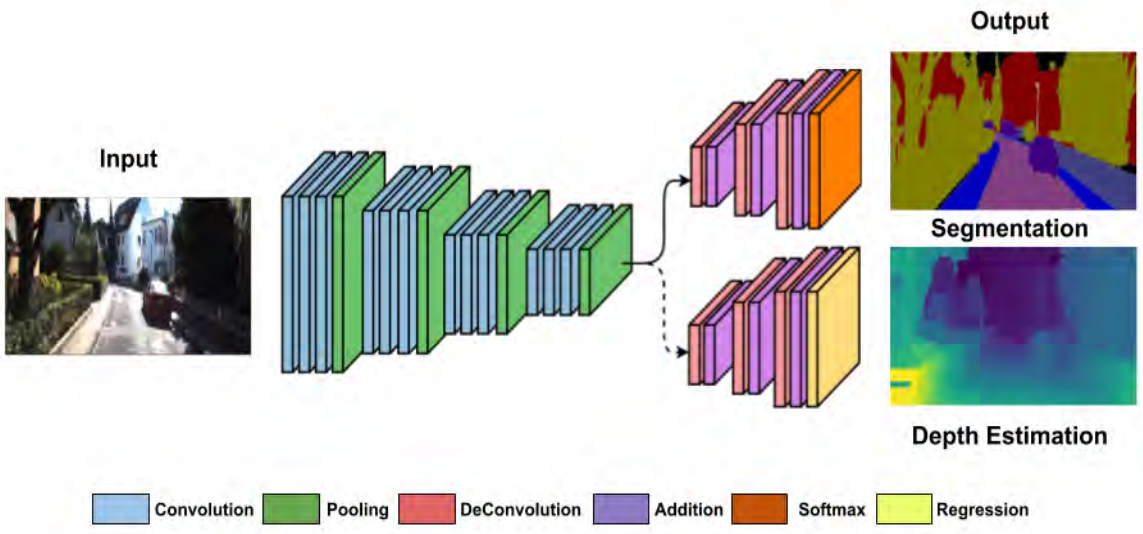


Figure 5.2: AuxNet: Auxiliary learning network with segmentation as main task and depth estimation as auxiliary task.

5.3.2 Loss Function

In general, a multi-task loss function is expressed as weighted combination of multiple task losses where L_i is loss and λ_i is associated weight for task i.

$$L_{Total} = \sum_{i=1}^{t} \lambda_i L_i \tag{5.1}$$

For the 2-task architecture we express loss as:

$$L_{Total} = \lambda_{Seg} L_{Seg} + \lambda_{Depth} L_{Depth} \tag{5.2}$$

L_{Seg} is semantic segmentation loss expressed as an average pixel-wise cross-entropy for each predicted label and ground-truth label. L_{Depth} is depth estimation loss described as the mean absolute error between estimated depth and real depth for all pixels. To overcome the significant scale difference between semantic segmentation and depth estimation losses, we perform task weight balancing as shown in Algorithm 1.

for $epoch \leftarrow 1$ **to** n **do**
  for $batch \leftarrow 1$ **to** s **do**
    $\lambda_{Seg} = L_{Depth}$
    $\lambda_{Depth} = L_{Seg}$
    $L_{Total} = L_{Depth} L_{Seg} + L_{Seg} L_{Depth}$
    $L_{Total} = 2 \times L_{Seg} L_{Depth}$
  end
end

Algorithm 1: Weight balancing for 2-task semantic segmentation and depth estimation.

Expressing the multi-task loss function as a product of task losses forces each task to optimize so that the total loss reaches a minimal value. This expression ensures no task is left in a stale mode while other tasks are making progress. By making an update after every batch in an epoch, we dynamically change the loss weights. We also add a moving average to the loss weights to smoothen the rapid changes in loss values at the end of every batch.

In Algorithm 2, we introduced focused task weight balancing to prioritize the main task's loss in auxiliary learning networks. We introduce an additional term to increase the importance of the main task. This term could be a fixed value to scale up the main task weight or the magnitude of task loss.

for $epoch \leftarrow 1$ **to** n **do**
  for $batch \leftarrow 1$ **to** s **do**
    $\lambda_{Seg} = L_{Seg} \times L_{Depth}$
    $\lambda_{Depth} = L_{Seg}$
    $L_{Total} = L_{Seg}^2 L_{Depth} + L_{Seg} L_{Depth}$
    $L_{Total} = (L_{Seg} + 1) \times L_{Seg} L_{Depth}$
  end
end

Algorithm 2: Focused task weight balancing for auxiliary learning.

5.4 Experiments and Results

This section presents details about the experimental setup used and discusses the observations on the results obtained.

5.4.1 Experimental Setup

We implemented the auxiliary learning network as discussed in section 5.3.1 to perform semantic segmentation and depth estimation. We chose ResNet-50 [21] as the shared encoder which is pre-trained on ImageNet [94]. We used segmentation and depth estimation decoders with random weight initialization. Semantic segmentation decoder is built using FCN8 [5]. Depth regression decoder is also constructed similar to segmentation decoder, except the final layer is replaced with regression units instead of softmax to estimate depth.

We refer to our baseline semantic segmentation network as SegNet and auxiliary learning network with depth estimation auxiliary task as AuxNet. We performed all our experiments on KITTI [77] semantic segmentation and SYNTHIA [74] datasets. These datasets contain RGB image data, ground truth semantic labels, and depth data represented as disparity values in 16-bit png format. We re-sized all the input images to a size 224x384.

The loss function is expressed as detailed in section 5.3.2. Categorical cross-entropy was used to compute semantic segmentation loss, and the mean absolute error is used to

calculate depth estimation loss.

5.4.2 Results and Discussion

In Table 5.1, we compare our auxiliary learning networks (AuxNet) against a simple semantic segmentation network (SegNet) constructed using an encoder-decoder combination. It is observed that auxiliary networks perform better than the baseline semantic segmentation in Figure 5.3. It is evident that incorporating depth information improves the performance of segmentation task. It is also observed that depth-dependent categories like sky, sidewalk, pole, and car have shown better improvements than other categories due to depth cues' availability.

We experimented with different hand-weighted and adaptive weighted task loss networks to understand the behavior of auxiliary learning. We expressed total loss for adaptive weighted task loss as a geometric mean of individual task losses. This expression adds a constraint for join minimization. We implemented four different auxiliary learning networks by changing the expression of the loss function. AuxNet_{400} and AuxNet_{1000} weighs segmentation loss 400 and 1000 times compared to depth estimation loss. $\text{AuxNet}_{\text{TWB}}$ and $\text{AuxNet}_{\text{FTWB}}$ are built based on Algorithms 1 and 2 respectively. These networks are trained with ADAM [131] optimizer for 200 epochs. The best model for each network was saved by monitoring the validation loss of the semantic segmentation task. Mean IoU and categorical IoU were used for comparing the performance. Auxiliary network achieved 4% and 3% IoU improvement on KITTI [77] and SYNTHIA [74] validation sets.

KITTI										
Model	**Sky**	**Building**	**Road**	**Sidewalk**	**Fence**	**Vegetation**	**Pole**	**Car**	**Lane**	**IoU**
SegNet	46.79	87.32	89.05	60.69	**22.96**	85.99	-	74.04	-	74.52
AuxNet_{400}	49.11	88.55	**93.17**	69.65	22.93	**87.12**	-	74.63	-	78.32
AuxNet_{1000}	49.17	89.81	90.77	64.16	14.77	86.52	-	71.40	-	76.58
$\text{AuxNet}_{\text{TWB}}$	**49.73**	**91.10**	92.30	**70.55**	18.64	86.01	-	**77.32**	-	**78.64**
$\text{AuxNet}_{\text{FTWB}}$	48.43	89.50	**92.71**	**71.58**	15.37	**88.31**	-	**79.55**	-	**79.24**
SYNTHIA										
Model	**Sky**	**Building**	**Road**	**Sidewalk**	**Fence**	**Vegetation**	**Pole**	**Car**	**Lane**	**IoU**
SegNet	**95.41**	58.18	93.46	09.82	76.04	80.95	08.79	85.73	90.28	89.70
AuxNet_{400}	95.12	**69.82**	92.95	21.38	77.61	84.23	51.31	**90.42**	91.20	91.44
AuxNet_{1000}	**95.41**	59.57	**96.83**	28.65	**81.23**	82.48	**56.43**	88.93	94.19	**92.60**
$\text{AuxNet}_{\text{TWB}}$	94.88	66.41	94.81	**31.24**	77.01	**86.04**	21.83	90.16	**94.47**	91.67
$\text{AuxNet}_{\text{FTWB}}$	**95.82**	56.19	96.68	21.09	81.19	83.26	55.86	89.01	92.11	92.05

Table 5.1: Comparison study : Single task vs auxiliary learning. AuxNet_{400} and AuxNet_{1000} weighs segmentation loss 400 and 1000 times compared to depth loss. $\text{AuxNet}_{\text{TWB}}$ is constructed by expressing total loss as product of task losses.

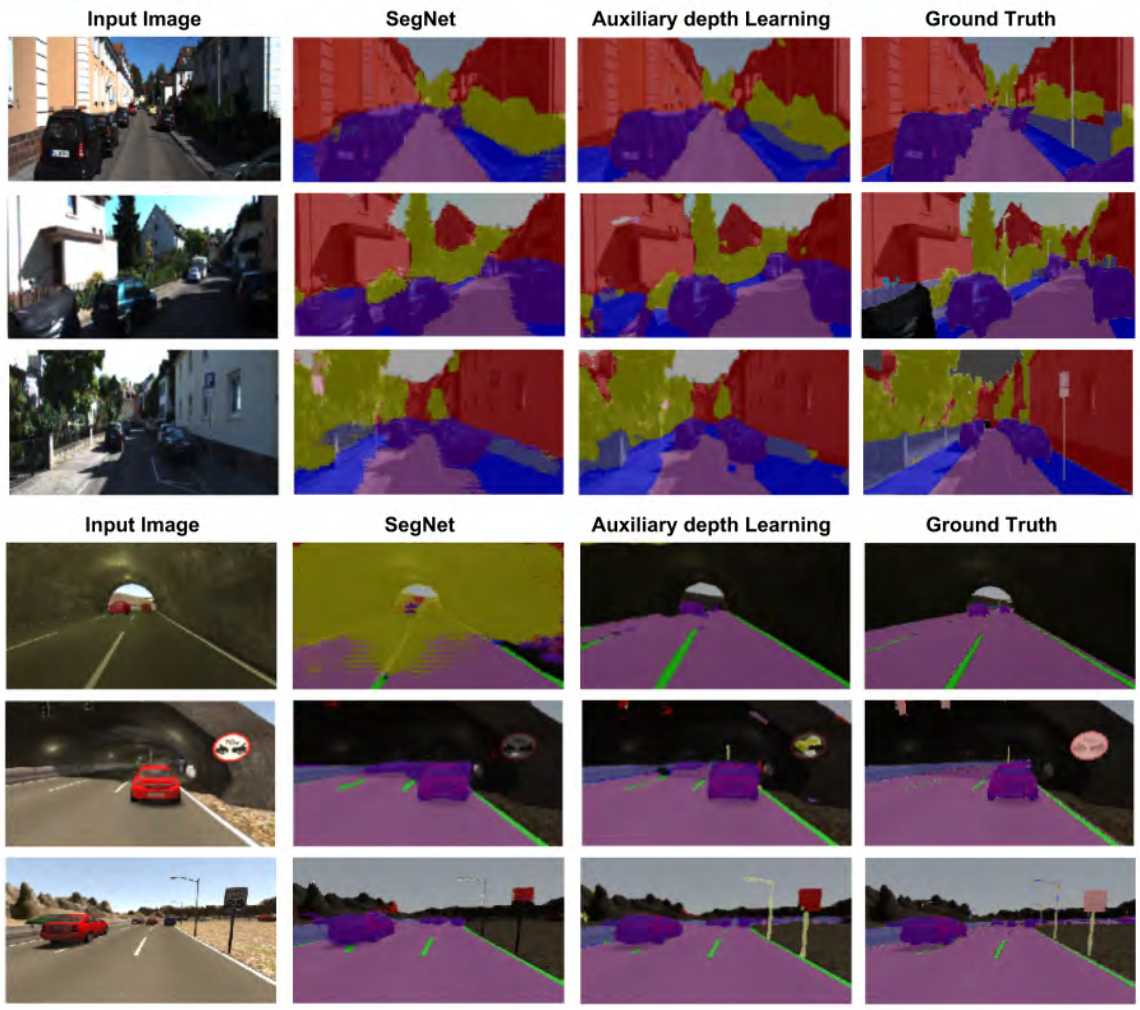


Figure 5.3: Results on KITTI (Top) and SYNTHIA (Bottom) datasets.

We compare the performances of SegNet, AuxNet with FuseNet [132] in Table 5.2. FuseNet is another semantic segmentation network (FuseNet) that takes RGB images and a depth map. We compare the mean IoU of each network and the number of parameters needed to construct the network. AuxNet requires a negligible increase in parameters, while FuseNet almost needs twice the parameters compared to SegNet. It is observed that AuxNet can be chosen as a suitable low-cost replacement to FuseNet as the shared encoder learns the needed depth information.

KITTI		
Model	**IoU**	**Params**
SegNet	74.52	23,672,264
FuseNet	**80.99**	47,259,976
AuxNet	79.24	23,676,142
SYNTHIA		
Model	**IoU**	**Params**
SegNet	89.70	23,683,054
FuseNet [132]	92.52	47,270,766
AuxNet	**92.60**	23,686,932

Table 5.2: Comparison between SegNet, FuseNet and AuxNet in terms of performance and parameters.

5.5 Conclusion

Semantic segmentation is a critical task to enable fully automated driving. It is also a complicated task and requires large amounts of annotated data, which is expensive. Large annotated datasets are currently the bottleneck for achieving high accuracy for deployment. In this chapter, we looked into an alternate mechanism of using auxiliary tasks to alleviate the lack of large datasets. We discussed how there are many auxiliary tasks in automated driving that can be used to improve accuracy. We implement a prototype and used depth estimation as an auxiliary task and show 5% improvement on KITTI and 3% improvement on SYNTHIA datasets. We also experimented with various weight balancing strategies, which are critical to solving for enabling more auxiliary tasks.

CHAPTER VI

Multi-stream Multi-task Learning

6.1 Introduction

Multi-task learning (MTL) [55] networks built using Convolution Neural Networks (CNNs) were usually limited to operate on a single stream of input data. Numerous works demonstrated using multiple streams of data as input to CNNs can improve performance drastically compared to using a single stream of input data. Recent attempts that use consecutive frames in a video sequence for semantic segmentation [133, 134], activity recognition [135, 100], optical flow estimation [136], moving object detection [44, 18] are examples demonstrating the benefits of using multiple streams of input data. Similarly, a pair of images from stereo vision cameras [137], or multiple images from different cameras of a car's surround-view system can also be processed as multiple streams of input to CNNs. Some works considered processing input data from different domains [138] to solve certain tasks that require multi-modal data representations.

These significant benefits demand the construction of a multi-task learning network that can operate on multiple streams of input data. Thus, we present MultiNet++, a novel multi-task network using simple feature aggregation methods as shown in Figure 6.1 to combine multiple streams of input data, which task-specific decoders can further process. Figure 6.1 illustrates a generic way to aggregate features temporally, and we make use of a simple summation junction to combine temporal features in our experiments. MultiNet++ would be ideal for processing video sequences for semantic segmentation, depth estima-

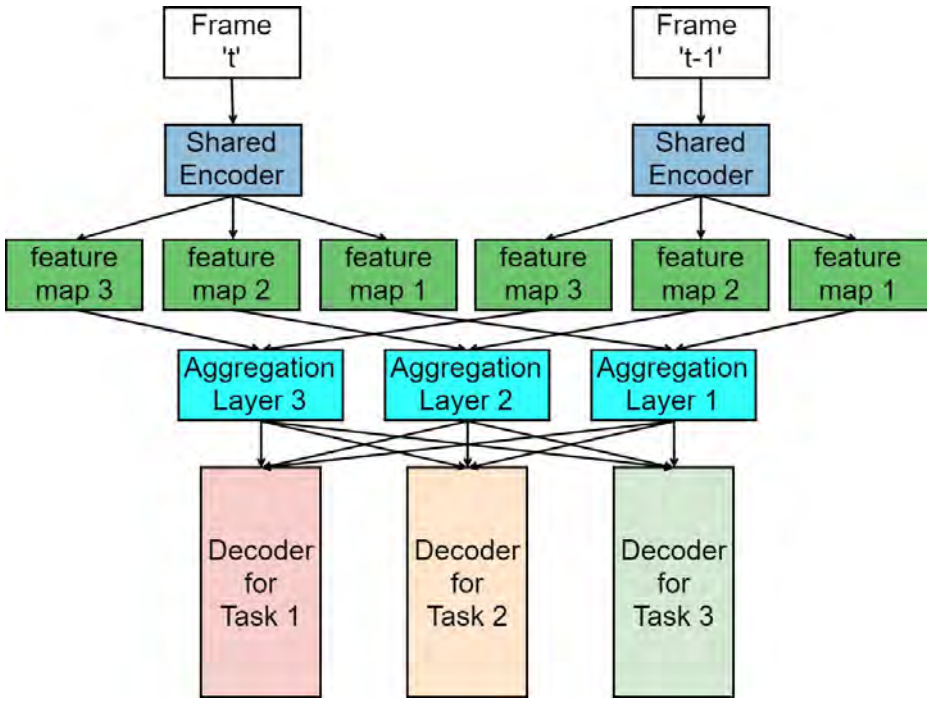


Figure 6.1: Illustration of MultiNet++ where feature aggregation is performed to combine intermediate output data obtained from a shared encoder that operates on multiple input streams (frames 't' and 't-1'). The aggregated features are later processed by task specific decoders.

tion, optical flow estimation, object detection, and tracking, etc., with improved efficiency. We also introduce a novel loss strategy for multi-task learning based on geometric mean representation to prioritize the learning of all tasks equally. We suggest using three diverse tasks: segmentation, depth estimation, and motion segmentation, which use appearance, geometry, and motion cues, respectively.

The rest of the contents in this chapter are structured as follows. Section 6.1.1 reviews related work using feature aggregation for multiple streams of inputs to CNNs. Section 6.2 discusses in detail the MultiNet++ network along with the geometric loss strategy introduced in this paper. Section 6.3 presents the experimental results on automotive datasets mainly KITTI [77], Cityscapes [78] and SYNTHIA [74]. Finally, Section 6.4 summarizes the chapter with key observations and concluding remarks.

6.1.1 Feature Aggregation

Different outputs from initial or mid-level convolution layers from CNNs (referred to as extracted features) are forwarded to the next processing stage using feature aggregation. Feature aggregation is a meaningful way to combine these extracted features. These features can be extracted from different CNNs operating on different input data [139, 140] or from a CNN operating on different resolutions of input [141]. Ranjan et al. [142] combines intermediate outputs from a CNN and passes them to the next processing stages. Yu et al. [143] proposed several possibilities of feature aggregation.

There are plenty of choices to perform feature aggregation. These choices range from using simple concatenation techniques to complex Long Short Term Memory Units (LSTMs) [144] or recurrent units. Simple concatenation or addition layers can capture short term temporal cues from a video sequence. Sun et al. [145] combine spatial and temporal features from video sequences for human activity recognition, and Karpathy et al. [135] combine features from inputs separated by 15 frames in a video for classification. Hei Ng et al. [146] proposed several convolution and pooling operations to combine features for video classification while Sistu et al. [134] used simple 1×1 bottleneck convolutions to combine features from consecutive frames for video segmentation.

In automotive or indoor robotic visual perception problems, simple concatenation techniques perform well. Still, they fall short in some applications like video captioning [147, 148] or summarization [149] where long term dependencies are required. LSTMs in such cases offer a better alternative [150, 151]. Convolution-LSTMs (Conv-LSTMs) [152, 153] and 3D convolutions [154] are other options. However, these options incur additional computational complexity, and they are needed mainly for the aggregation of features that are significant for long term dependencies.

6.1.2 Multi-task Loss

With the growing popularity of MTL, it is worth considering the possibility of imbalances in training an MTL network. It is often observed that some tasks dominate others during the training phase [70]. This dominance can be attributed to variations in task heuristics like complexities, uncertainties, magnitudes of losses, etc. Therefore an appropriate loss or prioritization strategy for all tasks in an MTL is a necessity.

Early works in MTL [56, 57, 106], use a weighted arithmetic sum of individual task losses. Later, several works attempted to balance the task weights using specific task heuristics discussed earlier. Kendall et al. [40] proposed to use homoscedastic uncertainty of tasks to weigh them. This work presents a multi-task learning problem as a joint probabilistic model with zero mean and variance expressed as task uncertainty. Minimizing the negative log-likelihood of this joint probabilistic model yields an optimal set of uncertainties that are used to weigh the task losses. This approach requires explicit modeling of uncertainty, and more importantly, the task weights remain constant.

GradNorm [69] is another notable work in which Chen et al. propose to normalize gradients from all tasks to a standard scale during backpropagation. Lui et al. [127] proposed Dynamic Weight Average (DWA), which uses an average of task losses overtime to weigh the task losses. Guo et al. [70] on the other hand, proposed dynamic task prioritization (DTP), where the changes in the difficulty of tasks adjust the task weights. DTP allows distributing focus on harder problems first and then on less challenging tasks. On the other hand, Liu et al. devised a different strategy to use a reinforcement learning-based approach to learn optimal task weights. However, this method isn't simple, and it brings additional complexity to the training phase.

In contrast to modeling multi-task problem as a single objective problem, Sener and Koltun [72] proposed to model it as a multi-optimization problem. Zhang and Yeung

[71] proposed a convex formulation for multi-task learning, and Desideri [73] proposed a multiple-gradient descent algorithm. In summary, these strategies either involve an explicit definition of loss function using task heuristics or require complex optimization techniques. Therefore, a loss strategy with minimal design complexities will be well suited for multi-task learning to accommodate a virtually unlimited number of joint tasks.

6.2 Methods

We introduce our novel multi-task network, MultiNet++, capable of processing multiple streams of input data. The proposed architecture is scalable and can be readily applied in any multi-task problem. In the following subsection, we discuss how we built our MultiNet++ network shown in Figure 6.2.

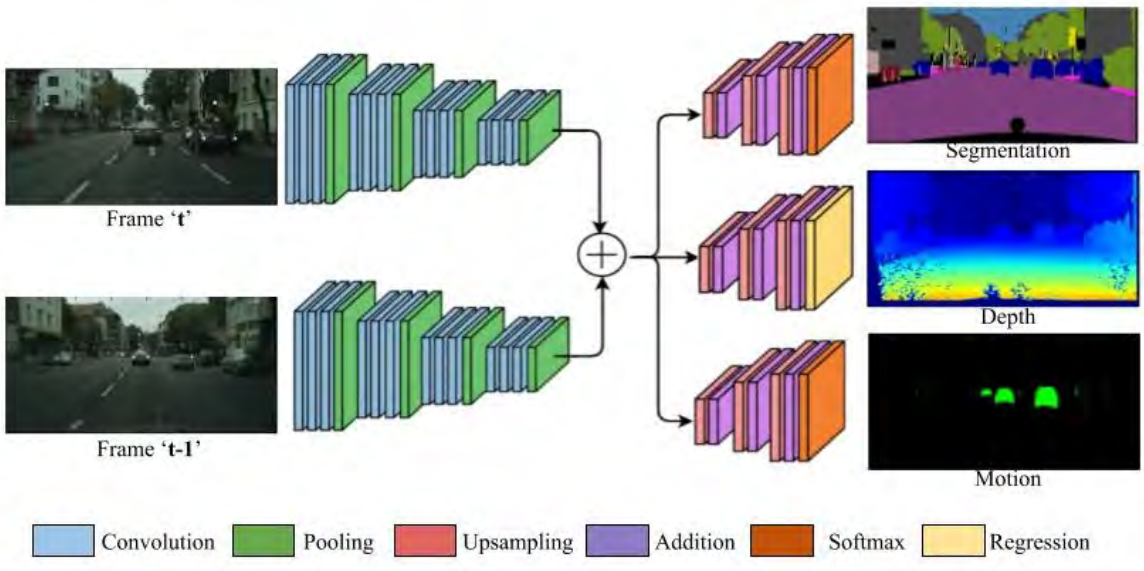


Figure 6.2: Illustration of the MultiNet++ network operating on consecutive frames of input video sequence. Consecutive frames are processed by a shared siamese-style encoder and extracted features are concatenated and processed by task specific segmentation, depth estimation and moving object detection decoders.

6.2.1 Multi-stream Multi-task Architecture

MultiNet++ is a simple multi-task network with the ability to process multiple streams of input data. It is built using three main components, 1) Encoders that feed multiple streams of input into the network, 2) Feature aggregation layers that concatenate the encoded feature vectors from multiple streams, and 3) Task-specific decoders that operate on aggregated feature space to perform task-specific operations. This chapter uses MultiNet++ for joint semantic segmentation, depth estimation, and moving object detection (or simply motion) on video sequences. We share the encoder between two consecutive frames from a given video sequence as shown in Figure 6.2. This network can significantly reduce the computational load as the encoders require a daunting number of parameters. These input frames can be selected sparsely or densely from a video sequence by observing its motion histogram. One can also choose to pass keyframes as proposed by Kulhare et al. [155].

Our encoders are selected by removing fully connected layers from ResNet-50 [21]. Outputs from ReLU [156] activation at layers 23, 39 and 46 from ResNet-50 [21] encoder are extracted and sent to feature aggregation layers. These feature maps extracted from different streams of inputs are concatenated and sent to task-specific decoders as shown in Figure 6.1. Segmentation decoder is built using FCN8 [5] architecture that comprises three upsampling layers and skip connections from aggregated feature maps as shown in Figure 6.2. The final layer consists of softmax [157] units to predict pixel-wise classification labels. Similarly, we construct a motion decoder by changing the number of output classes in softmax units. Depth decoder is built by replacing softmax with regression units.

6.2.2 Geometric Loss Strategy

We discussed the importance of a loss strategy that requires minimal effort during the design phase in Section 6.1.2. The commonly used loss combination function is an arithmetic mean, and it suffers from differences in the scale of the individual losses. A weighted average of the losses partially alleviates this, but it isn't easy to tune manually. We were motivated to explore the geometric loss combination, which is invariant to the scale of the individual losses. Thus we express the total loss of a multi-task learning problem as the geometric mean of individual task losses. We refer to this as Geometric Loss Strategy (GLS). For an n-task problem with task losses 'L_1','L_2' ... 'L_n', we express total loss as:

$$L_{Total} = \prod_{i=1}^{n} \sqrt[n]{L_i} \tag{6.1}$$

For example, in a 3-task problem with losses 'L_1','L_2' and 'L_3', we express total loss:

$$L_{Total} = \sqrt[3]{L_1 L_2 L_3} \tag{6.2}$$

Equations 6.1 and 6.2 are quite popular in geometric programming. This loss function is differentiable and can be optimized using an optimizer like Stochastic Gradient Descent (SGD). This definition makes sure that all tasks are making progress. We adapt our loss function to focus or give more attention to certain tasks by introducing Focused Loss Strategy (FLS) where we multiply geometric mean of losses of focused tasks to existing loss function. In this case, we define loss function with focus on m ($m \leq n$) important tasks as:

$$L_{Total} = \prod_{i=1}^{n} \sqrt[n]{L_i} \times \prod_{j=1}^{m} \sqrt[m]{L_j} \tag{6.3}$$

In a 3-task problem with focus on 2-tasks, we can express loss function as:

$$L_{total} = \sqrt[3]{L_1 L_2 L_3} \times \sqrt[2]{L_1 L_2} \tag{6.4}$$

Equation 6.3 and 6.4 provides an opportunity to focus on important tasks in a multi-task learning problem. Here we assume that the tasks are ordered in terms of priority so that the first m tasks out of the total n tasks get higher weightage. This expression is a simple extension that can be generalized to weighted-geometric mean, which would have more hyper-parameters to be learned.

Application of log function converts the product of losses to the sum of log of individual losses and can thus be interpreted as equivalent to normalizing individual losses and then adding them. However, it is computationally complex to make use of the log function.

6.3 Experiments and Results

In this section, we discuss the datasets used for evaluating the efficacy of our models. Later, we discuss how we constructed the proposed models and provide a complexity analysis of each. We also discuss the optimization strategies used during the training phase. Finally, we present the results obtained along with a discussion.

6.3.1 Datasets

KITTI [77], Cityscapes [78] and SYNTHIA [74] are popular automotive datasets. KITTI has annotations for several tasks, including semantic segmentation, depth estimation, object detection, etc. However, these annotations were done separately for each task, and the input is not always common across the tasks. KITTI Stereo 2015 [158, 159] dataset provides stereo images for depth estimation. A subset of these images is labeled for KITTI semantic segmentation [77]. This dataset consists of 200 train images and 200 test images.

Cityscapes [78] dataset provides both segmentation and depth estimation annotations for $\approx$ 3500 images. Motion labels for these datasets are provided by Vertens et al. [18]. SYNTHIA [74] is a synthetic dataset that provides segmentation and depth annotations for raw video sequences simulated in different weather, light conditions and road types. KITTI [77] and Cityscapes [78] provide segmentation labels for 20 categories while SYNTHIA [74] dataset provides segmentation labels for 13 categories.

In KITTI [77], and Cityscapes [78] datasets, images are sampled and annotated sparsely from raw videos. This poses a challenge to approaches that use temporal methods for segmentation or motion detection tasks in videos. In addition to KITTI [77], and Cityscapes [78] datasets, we use SEQS-02 (New York-like city) and SEQS-05 (New York-like city) from the SYNTHIA dataset for training and validation respectively in our experiments. These sequences provide segmentation and depth annotations for consecutive images in a video sequence. Thus they are more suitable for evaluating our multi-task model, which operates on multiple input data streams. Table 6.1 provides a summary of different properties of the three datasets discussed so far.

Annotations	KITTI[77]	Cityscapes[78]	SYNTHIA[74]
Segmentation	✓	✓	✓
Depth	✓	✓	✓
Motion	✓	✓	×
# Train	200	2,975	888
# Validation	200	500	787
# Type	Real	Real	Synthetic

Table 6.1: Summary of the automotive datasets used in our experiments.

6.3.2 Model Analysis

We constructed several models to evaluate the benefits of the MultiNet++. We build 3 single-task baseline models for segmentation, depth and motion tasks using ResNet-50 [21]

as an encoder and different task-specific decoders as discussed in Section 6.2.1. Segmentation decoder predicts pixel-wise labels from 20 different categories for input in KITTI [77] & Cityscapes [78] datasets, while the decoder predicts from 13 categories in SYNTHIA [74] dataset. Depth decoder outputs a 16-bit integer at every pixel location to predict depth. The motion decoder predicts a binary classification label for every pixel to classify as a moving or static object. These models process one frame of input data. We also constructed 2-task and 3-task models that operate on a single frame and two consecutive frames of an input video sequence. MultiNet++ refers to models that operate on two consecutive frames built using feature aggregation as discussed in Section 6.2.1. Table 6.2 provides details about number parameters required to construct different models.

Majority of computational load arises from ResNet-50 [21] encoder. Due to this property, 2-task and 3-task models required the almost same number of parameters as the 1-task model. This is one of the main reasons why multi-task networks are computationally efficient and favor embedded deployment. We build our 2-frame models with relatively very little increase in complexity ($\approx$ 100K parameters) by reusing the encoder between 2-frames. In the 2-frames model, the aggregated features are larger when compared to the 1-frame model. It increased the parameters needed for our 2-frame model.

Method	KITTI & Cityscapes					SYNTHIA			
	Encoder	Segmentation	Depth	Motion	Total	Encoder	Segmentation	Depth	Total
1-Task Segmentation, Depth or Motion									
1-Task	23.58M	0.18M	-	-	23.77M	23.58M	0.14M	-	23.68M
1-Task	23.58M	-	3.88K	-	23.59M	23.58M	-	3.87K	23.59M
1-Task	23.58M	-	-	8.33K	23.60M	-	-	-	-
2-Task Segmentation and Depth									
1-Frame	23.58M	0.18M	3.88K	-	23.77M	23.58M	95.34K	3.88K	23.69M
2-Frames	23.58M	0.26M	7.46K	-	23.86M	23.58M	0.14M	7.46K	23.74M
2-Task Segmentation and Motion									
1-Frame	23.58M	0.18M	-	8.33K	23.78M	-	-	-	-
2-Frames	23.58M	0.26M	-	15.50K	23.86M	-	-	-	-
3-Task Segmentation, Depth and Motion									
1-Frame	23.58M	0.18M	3.88K	8.33K	23.79M	-	-	-	-
2-Frames	23.58M	0.26M	7.46K	15.50K	23.87M	-	-	-	-

Table 6.2: Comparative study: Parameters needed to construct 1-task segmentation, depth and motion, 2-task segmentation and depth, 2-task segmentation and motion and 3-task segmentation, depth and motion models. We compare 2-task and 3-task models that operate on 1-frame and 2-frames. 2-frame models required relatively minimal additional computational complexity compared to 1-frame models.

6.3.3 Optimization

We implemented our models using Keras [102]. In all our experiments, we re-size the input images to 224×384. We used only 2-frames for feature aggregation because adding more frames would increase computational complexity with insignificant performance gains, as demonstrated by Sistu et al. [134] In our multi-task learning networks, we define each task's loss functions separately and feed them to our geometric loss strategy (GLS) introduced in Section 6.1.2. For semantic segmentation and motion, we use pixel-wise cross-entropy loss for C classes averaged over a mini-batch with N samples as shown in Equation 6.5.

$$L_{Seg} \text{ or } L_{Motion} = -\sum_{j=1}^{N}\sum_{i=1}^{C} y_{i,j} log(p_{i,j}) \tag{6.5}$$

For depth estimation, we use Huber loss as defined in Equation 6.6 with δ =250.

$$L_{Depth} = \begin{cases} \frac{1}{2}[y-\hat{y}]^2 & : |y-\hat{y}| \leq \delta \\ \delta(|y-\hat{y}| - \delta/2) & : otherwise \end{cases} \tag{6.6}$$

The total loss L_{Total} is defined as:

$$L_{Total} = \sqrt[3]{L_{Seg} L_{Depth} L_{Motion}} \tag{6.7}$$

We optimize this loss function in our training phase using Adam optimizer [131]. Accuracy is used as an evaluation metric for segmentation and motion tasks, while regression accuracy is used for depth estimation.

6.3.4 Results

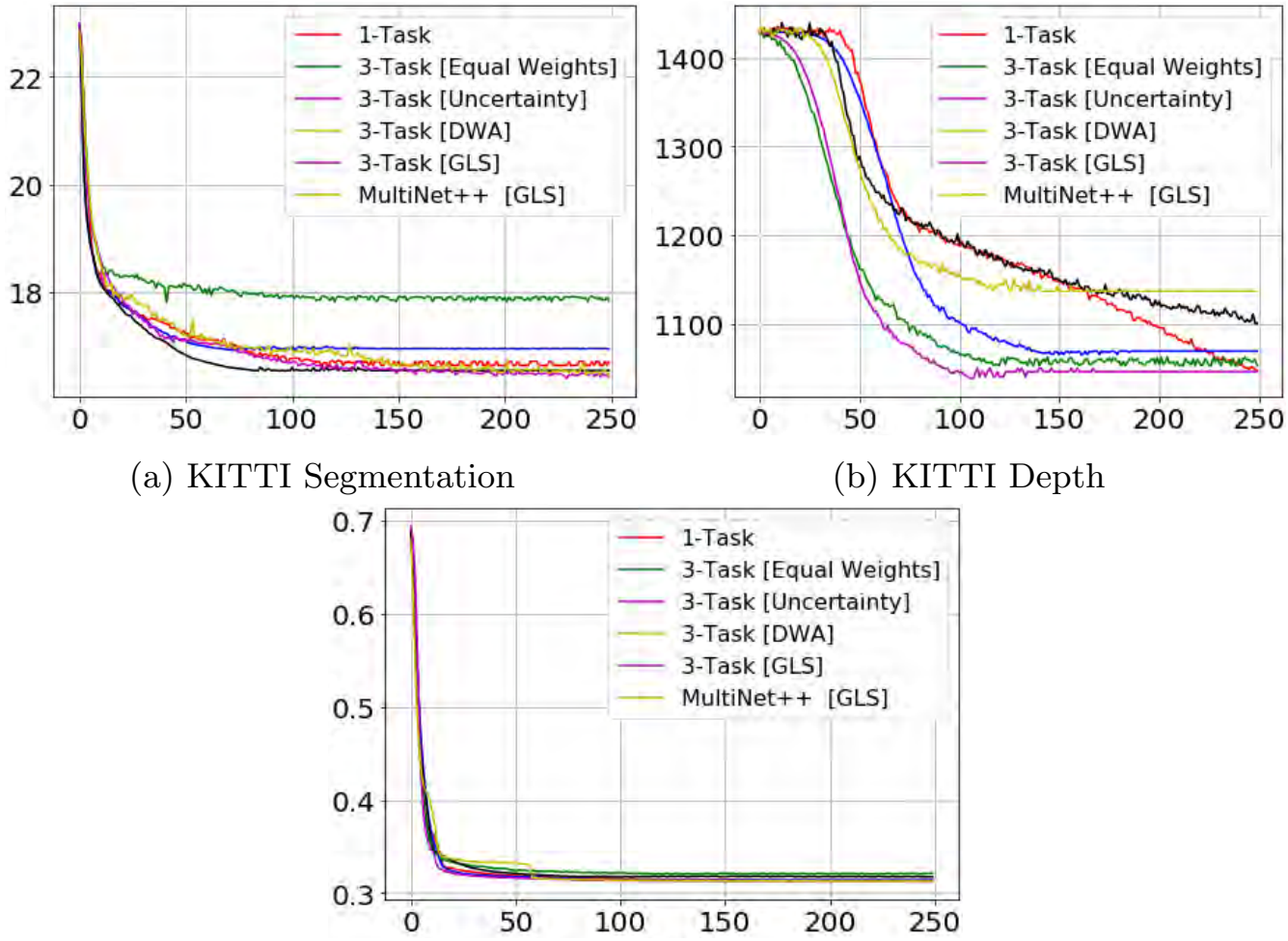
In Table 6.4, we compare the results of 2-task models and 3-task models using our geometric loss strategy (GLS) against naive equal task weight method. We also compare their performances with 1-task segmentation, depth, and motion models. Our GLS method shows significant improvements in performance over equal weights method in both 2-task and 3-task models. In Table 6.3, we compare the results of 3-task models using our geometric loss strategy (GLS) against naive equal task weights, uncertainty weight method proposed by Kendal et al. [40] and Dynamic Weight Average (DWA) proposed by Liu et al. [127]. In Figures 6.3 and 6.4, we show how validation loss for these models change over time during training phase on KITTI [77] and Cityscapes [78] datasets. In Figure 6.5, we show the same for segmentation and depth tasks on SYNTHIA [74] dataset. Our models using GLS demonstrated faster convergence on all tasks.

Method	Segmentation	Depth	Motion
KITTI			
1-Task	81.74%	75.91%	**98.49%**
Equal weights	77.14%	76.15%	97.83%
Uncertainty [40]	78.93%	75.73%	98.00%
DWA [127]	80.05%	74.48%	97.78%
GLS (ours)	**82.20**%	**76.54**%	97.92%
MultiNet++ (ours)	80.06%	73.94%	97.94%
Cityscapes			
1-Task	**78.95**%	60.13%	**98.72**%
Equal weights	72.71%	60.97%	98.20%
Uncertainty [40]	77.32%	60.44%	98.63%
DWA [127]	78.05%	59.34%	98.45%
GLS (ours)	77.38%	**61.56**%	**98.72**%
MultiNet++ (ours)	**82.36**%	**62.74**%	98.21%

Table 6.3: Comparative Study: Performance of 1-Task, equal weights, 3-task uncertainty, Dynamic Weight Average (DWA) and geometric loss strategy (GLS) on KITTI and Cityscapes datasets.

Method	KITTI			Cityscapes			SYNTHIA	
	Segmentation	Depth	Motion	Segmentation	Depth	Motion	Segmentation	Depth
1-Task Segmentation, Depth or Motion								
1-Task	81.74%	-	-	78.95%	-	-	84.08%	-
1-Task	-	75.91%	-	-	60.13%	-	-	73.19%
1-Task	-	-	**98.49**%	-	-	**98.72**%	-	-
2-Task Segmentation and Depth								
Equal weights	74.30%	74.47%	-	73.76%	59.38%	-	63.45%	71.84%
GLS (ours)	81.50%	74.92%	-	79.14%	60.15%	-	86.87%	73.60%
MultiNet++	81.01%	73.95%	-	**83.07**%	60.15%	-	**88.15**%	**78.39**%
2-Task Segmentation and Motion								
Equal weights	80.14%	-	97.88%	78.46%	-	98.25%	-	-
GLS (ours)	81.52%	-	97.93%	77.63%	-	98.83%	-	-
MultiNet++	81.75%	-	98.15%	78.86%	-	98.65%	-	-
3-Task Segmentation, Depth and Motion								
Equal weights	77.14%	76.15%	97.83%	72.71%	60.97%	98.20%	-	-
GLS (ours)	**82.20**%	**76.54**%	97.92%	77.38%	61.56%	98.72%	-	-
MultiNet++	80.06%	73.94%	97.94%	82.36%	**62.74**%	98.21%	-	-

Table 6.4: Improvements in learning segmentation, depth estimation and motion detection as multiple tasks using equal weights, geometric loss strategy (GLS) and 2 stream feature aggregation with GLS (MultiNet++) vs independent networks (1-Task) on KITTI, Cityscapes and SYNTHIA datasets.

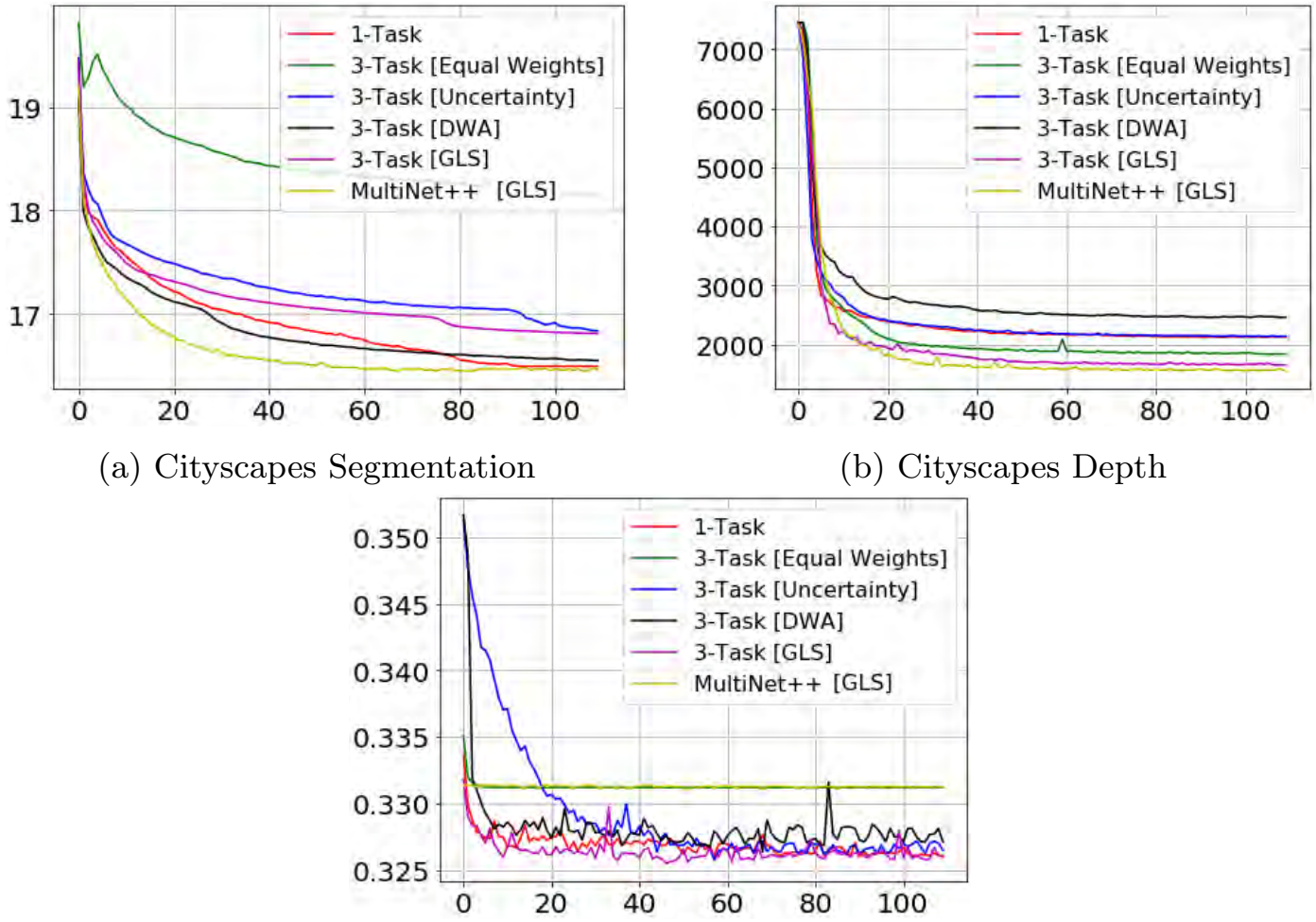


(a) KITTI Segmentation

(b) KITTI Depth

(c) KITTI Motion

Figure 6.3: Change of validation loss (X-axis) over several epochs (Y-axis) during training phase for 1-Task model vs 3-Task models for segmentation, depth and motion tasks on KITTI dataset.

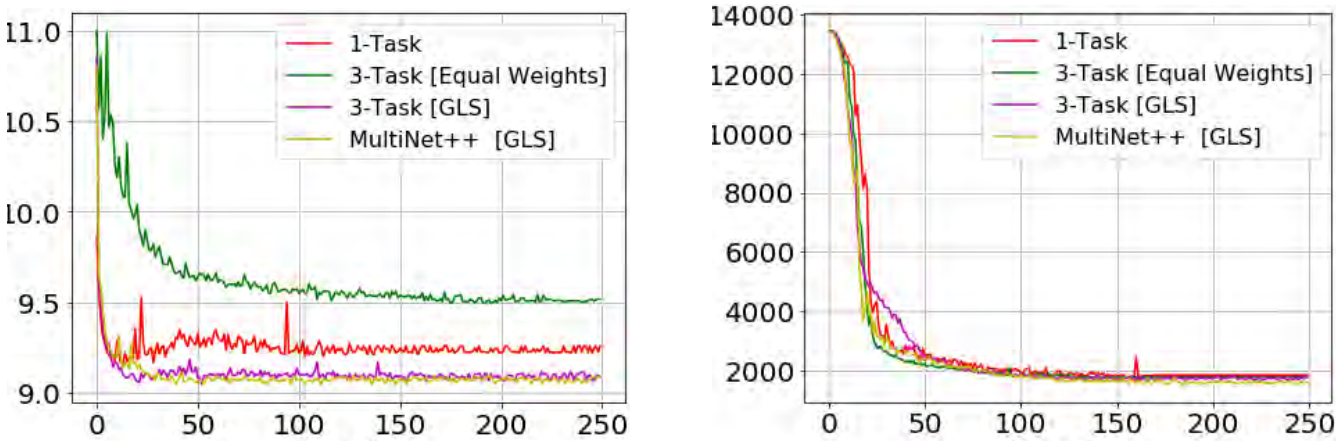


(a) Cityscapes Segmentation

(b) Cityscapes Depth

(c) Cityscapes Motion

Figure 6.4: Change of validation loss (X-axis) over several epochs (Y-axis) during training phase for 1-Task model vs 3-Task models for segmentation, depth and motion tasks on Cityscapes dataset.



(a) SYNTHIA Segmentation

(b) SYNTHIA Depth

Figure 6.5: Change of validation loss (X-axis) over several epochs (Y-axis) during training phase for 1-Task model vs 2-Task models for segmentation and depth on SYNTHIA dataset.

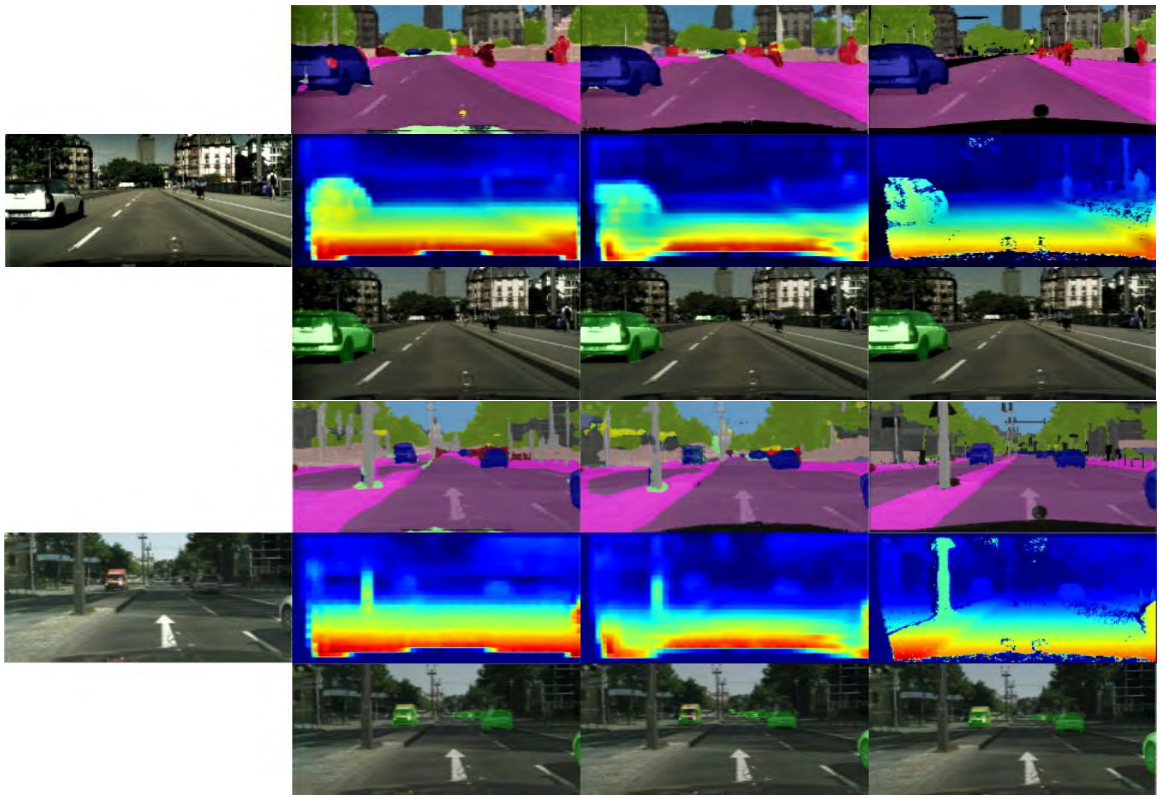


Figure 6.6: Left to right: Input image, single task network outputs, MultiNet++ output, ground truth. More qualitative results of MultiNet++ model can be accessed via this link https://youtu.be/E378PzLq7lQ.

In 3-task models solving for segmentation, depth, and motion, depth is usually the most complex task. Figures 6.3.(b) and 6.5.(b) and show that depth estimation on KITTI [77] and Synthia [74] requires longer convergence time compared to segmentation (Figure 6.3.(a) and 6.5.(a) and motion tasks (Figures 6.3.(c)). In these cases, our GLS method has shown faster convergence compared to uncertainty [40], and DWA [127] methods. While solving for multiple tasks, uncertainty [40] and DWA [127] weigh the tasks that converge quickly higher than the others. This led to faster convergence in segmentation and motion tasks but late convergence in depth tasks. In such circumstances, the encoder parameters might be biased towards segmentation and motion tasks. This can result in imbalanced

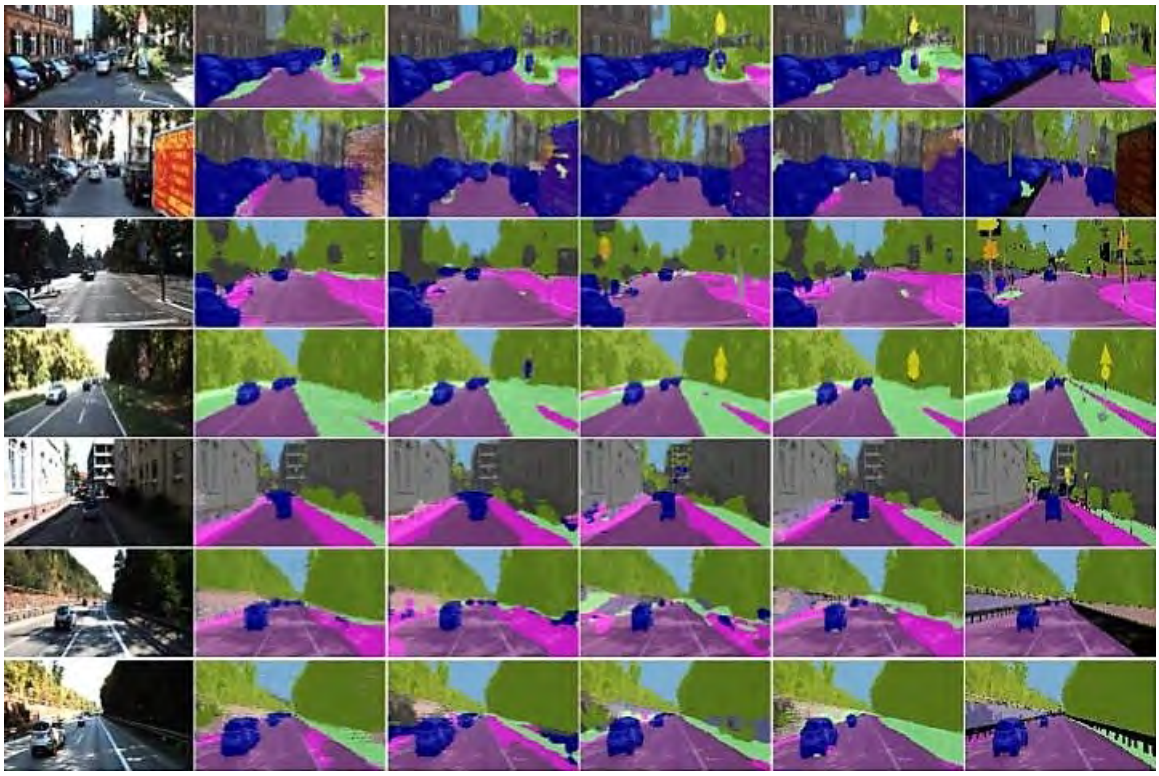


Figure 6.7: Left to right: Input image, semantic segmentation output from single task, 3-Task with equal weights, 3-Task GLS, 3-Task MultiNet++ networks, ground truth.

learning of depth tasks. Our GLS method expresses the total loss as the geometric mean of individual losses, so it doesn't prioritize one task higher than others.

In Table 6.4, we also compare 2-task and 3-task models with our novel MultiNet++ which uses both feature aggregation (for 2-frame input) and GLS. In KITTI [77] dataset, input images are sparsely sampled from raw video sequences, which hinder the performance gains of MultiNet++. In Cityscapes [78] dataset, MultiNet++ outperforms single-task models by 4% and 3% for segmentation and depth tasks, respectively, as they provide images sampled closely compared to the KITTI dataset. These improvements are much better in SYNTHIA [74] dataset (4% and 5% for segmentation and depth estimation tasks, respectively) as they provide continuous video frames sequences. We achieve similar per-

formances for motion task compared to 1-task models.

We compare qualitative results of MultiNet++ with 1-task segmentation model on Cityscapes [78] dataset in Figure 6.6. The main difference between 1-task models and 3-task models is that the latter have learned representations from other tasks using a common encoder. Knowledge acquired through these representations helps the 3-task model identify semantic boundaries better than the 1-task model. MultiNet++ model has improved performance. Our models detect traffic signs, lights, and other near-range objects better compared to other models on KITTI dataset [77] as shown in Figure 6.7.

6.4 Conclusion

We introduced an efficient way of constructing MultiNet++, a multi-task learning network that operates on multiple input data streams. We demonstrated that our geometric loss strategy (GLS) is robust to different task heuristics like complexity, magnitude, etc. We achieved balanced training and improved performances for a multi-task learning network solving different tasks, namely segmentation, depth estimation, and motion on automotive datasets KITTI, Cityscapes, and SYNTHIA. Our GLS strategy is easy to implement. Most importantly, it allows for balanced learning of a large number of tasks in multi-task learning without requiring explicit loss modeling compared to other multi-task learning loss strategies. In the future, we would like to explore the benefits of multi-task learning networks using our efficient feature aggregation and loss strategies for multi-modal data.

CHAPTER VII

Conclusions

This dissertation introduces novel approaches and methodologies to advance the state-of-the-art visual perception for automated driving.

End-to-end deep learning networks dominate the literature in computer vision and are widely used in consumer intelligent systems. First, we explored how to optimize these end-to-end deep learning network architectures to improve individual perception tasks' performance. We proposed to learn complete scene understanding in the form of panoptic segmentation using instance contour representation. We also introduced novel architectures that help fuse multiple input streams from a temporal input to learn video semantic segmentation in a computes efficient manner.

Later, we developed strategies to build a joint model that shares the available compute power to process multiple tasks primarily built using end-to-end deep learning networks. We argued that adaptation of end-to-end models for conventional tasks like calibration etc. benefit from multi-task learning as it is easier to incorporate these models into a joint network with minimal additional computational complexity. We later introduced auxiliary learning methods that learn better representations of given tasks with limited data when trained alongside an auxiliary/relevant task with surplus amounts of data. We also presented better replacements for total loss in a multi-task learning network like geometric representation instead of arithmetic counterparts. Finally, we combined our novel multi-stream fusion, multi-task network, and geometric loss strategies to build a multi-stream multi-task learning network that outperforms existing methods.

7.1 Findings, Limitations and Future Work

We briefly summarise the findings, limitations, and future work of each chapter in this dissertation.

- **Panoptic Segmentation**: Our methods provide a baseline for a single-stage panoptic segmentation network. These methods are lightweight compared to state-of-the-art two-stage object detection, instance clustering-based methods. They are better in terms of performance compared to single-stage instance segmentation methods. However, the performance of two-stage methods are superior compared to our methods and also single-stage methods in general. In future, we would like to update our instance segmentation refinement with dynamic clustering techniques instead of current DSBCAN method. Also, we would like to try different encoders with better performances to improve current results.

- **Multi-stream Learning**: An automated driving scene doesn't change rapidly; thus, Recurrent Neural networks like LSTM's or GRU's are not required as they are built to handle long term dependencies in temporal sequences. Our multi-stream CNN-based semantic segmentation provides an efficient alternative to Recurrent Neural networks while improving various datasets' quantitative performances. In future we would like to fuse the temporal features using optical flow or similar motion related cues. Also, we would like to learn how to fuse, what to fuse by adding a learnable block instead of feature concatenation.

- **Multi-task learning**: CNN's have become the standard model for semantic tasks like object detection and semantic segmentation, geometric tasks like depth estimation and visual SLAM. This brings an opportunity for CNNs to become a unifying model

for all visual perception tasks for automated driving. In this chapter, we argue for moving towards a unified model and use current literature to propose how to achieve it. We also discuss the pros and cons of having a unified model. Finally, we perform experiments on a simpler scenario with two, three tasks and demonstrate results to support our argument.

- **Auxiliary learning**: Large annotated datasets are currently the bottleneck for achieving high accuracy for deployment. An alternate mechanism of using auxiliary tasks to alleviate the lack of large datasets is presented in this work. A prototype that uses depth estimation as an auxiliary task to semantic segmentation is implemented to show 5% improvement on KITTI and 3% improvement in semantic segmentation performance on SYNTHIA datasets. Auxiliary learning may fail when the auxiliary task has very little similarity with main task. In future, we would like to learn how different task relate to each other and determine if they are suitable to be trained in an auxiliary learning setup. Another shortcoming of auxiliary learning is it requires to find right parameters for task weight balancing.

- **Multi-stream multi-task learning**: An efficient way of constructing a multi-task learning network that operates on multiple streams of input data is presented along with a geometric loss strategy (GLS) robust to different task heuristics like complexity, magnitude, etc. GLS strategy is easy to implement. Most importantly, it allows for balanced learning of many tasks in multi-task learning without requiring explicit loss modeling compared to other multi-task learning loss strategies. In future, we would like to extend the multi-stream multi-task learning to multiple modalities.

7.2 Broader Impacts

The presented research can improve perception tasks in other domains like medical imaging, smartphone photography, video surveillance, etc., where performance and efficiency are essential. Further, the presented multi-task learning strategies help in learning current perception solutions jointly. Applications in augmented reality, virtual 3d tours of real estate properties, etc., where recognition and reconstruction of the environment are essential, can significantly benefit from unified visual perception networks that learn recognition and reconstruction tasks using deep learning.

BIBLIOGRAPHY

[1] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, May 2004. [Online]. Available: https://doi.org/10.1023/B:VISI.0000013087.49260.fb

[2] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[4] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2117–2125.

[5] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[6] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1520–1528.

[7] 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*, vol. 39, no. 12, pp. 2481–2495, 2017.

[8] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE international Conference on Computer Vision*, 2015, pp. 2650–2658.

[9] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5162–5170.

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

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

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

[13] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scandinavian conference on Image analysis*, 2003, pp. 363–370.

[14] A. Kundu, K. M. Krishna, and J. Sivaswamy, "Moving object detection by multi-view geometric techniques from a single camera mounted robot," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 4306–4312.

[15] T.-H. Lin and C.-C. Wang, "Deep learning of spatio-temporal features with geometric-based moving point detection for motion segmentation," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3058–3065.

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

[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," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2017, p. 6.

[18] J. Vertens, A. Valada, and W. Burgard, "Smsnet: Semantic motion segmentation using deep convolutional neural networks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 582–589.

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

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[22] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9404–9413.

[23] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen, "Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 475–12 485.

[24] A. Kirillov, R. Girshick, K. He, and P. Dollár, "Panoptic feature pyramid networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6399–6408.

[25] Y. Xiong, R. Liao, H. Zhao, R. Hu, M. Bai, E. Yumer, and R. Urtasun, "Upsnet: A unified panoptic segmentation network," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8810–8818.

[26] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[27] P. Hurtik, V. Molek, J. Hula, M. Vajgl, P. Vlasanek, and T. Nejezchleba, "Poly-yolo: higher speed, more precise detection and instance segmentation for yolov3," *arXiv preprint arXiv:2005.13243*, 2020.

[28] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," *arXiv preprint arXiv:1611.07709*, 2016.

[29] X. Liang, L. Lin, Y. Wei, X. Shen, J. Yang, and S. Yan, "Proposal-free network for instance-level object segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2978–2991, 2017.

[30] S. Peng, W. Jiang, H. Pi, X. Li, H. Bao, and X. Zhou, "Deep snake for real-time instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[31] M. Siam, S. Elkerdawy, M. Jagersand, and S. Yogamani, "Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–8.

[32] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[33] I. Freeman, L. Roese-Koerner, and A. Kummert, "Effnet: An efficient structure for convolutional neural networks," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018, pp. 6–10.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[35] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: learning dense volumetric segmentation from sparse annotation," in *International conference on medical image computing and computer-assisted intervention*, 2016, pp. 424–432.

[36] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1529–1537.

[37] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision*, 2016, pp. 21–37.

[38] 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*, 2015, pp. 91–99.

[39] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international Conference on Computer Vision*, 2017, pp. 2961–2969.

[40] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7482–7491.

[41] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: ordering points to identify the clustering structure," *ACM Sigmod record*, vol. 28, no. 2, pp. 49–60, 1999.

[42] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96. AAAI Press, 1996, p. 226231.

[43] A. Petrovai and S. Nedevschi, "Multi-task network for panoptic segmentation in automated driving," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 2394–2401.

[44] M. Siam, H. Mahgoub, M. Zahran, S. Yogamani, M. Jagersand, and A. El-Sallab, "Modnet: Motion and appearance based moving object detection network for autonomous driving," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2859–2864.

[45] N. Garnett, S. Silberstein, S. Oron, E. Fetaya, U. Verner, A. Ayash, V. Goldner, R. Cohen, K. Horn, and D. Levi, "Real-time category-based and general obstacle detection for autonomous driving," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop*, 2017, pp. 198–205.

[46] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[47] M. Aladem and S. A. Rawashdeh, "Lightweight visual odometry for autonomous mobile robots," *Sensors*, vol. 18, no. 9, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/9/2837

[48] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE international symposium on mixed and augmented reality*, 2011, pp. 127–136.

[49] M. Aladem, S. Chennupati, Z. El-Shair, and S. A. Rawashdeh, "A comparative study of different cnn encoders for monocular depth prediction," in *2019 IEEE National Aerospace and Electronics Conference (NAECON)*, 2019, pp. 328–331.

[50] S. Chennupati, G. Sistu., S. Yogamani., and S. Rawashdeh., "Auxnet: Auxiliary tasks enhanced semantic segmentation for automated driving," in *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP,*. SciTePress, 2019, pp. 645–652.

[51] S. Chennupati, G. Sistu, S. Yogamani, and S. A Rawashdeh, "Multinet++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[52] M. Uřičář, P. Křížek, G. Sistu, and S. Yogamani, "Soilingnet: Soiling detection on automotive surround-view cameras," in *2019 22nd International Conference on Intelligent Transportation Systems (ITSC)*, 2019, to appear.

[53] S. Yogamani, C. Hughes, J. Horgan, G. Sistu, P. Varley, D. O'Dea, M. Uricár, S. Milz, M. Simon, K. Amende *et al.*, "Woodscape: A multi-task, multi-camera fisheye dataset for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9308–9318.

[54] S. A. E. International, "Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems J3016," 2018.

[55] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[56] 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)*, July 2017, pp. 5454–5463.

[57] M. Teichmann, M. Weber, M. Zllner, R. Cipolla, and R. Urtasun, "Multinet: Real-time joint semantic reasoning for autonomous driving," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, June 2018, pp. 1013–1020.

[58] V. Sanh, T. Wolf, and S. Ruder, "A hierarchical multi-task approach for learning embeddings from semantic tasks," 2018.

[59] D. Dong, H. Wu, W. He, D. Yu, and H. Wang, "Multi-task learning for multiple language translation," in *ACL*, 2015.

[60] Z. Wu, C. Valentini-Botinhao, O. Watts, and S. King, "Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 4460–4464.

[61] O. Siohan and D. Rybach, "Multitask learning and system combination for automatic speech recognition," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, Dec 2015, pp. 589–595.

[62] P. Dewangan, S. P. Teja, K. M. Krishna, A. Sarkar, and B. Ravindran, "Digrad: Multi-task reinforcement learning with shared actions," *CoRR*, vol. abs/1802.10463, 2018.

[63] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, "Learning modular neural network policies for multi-task and multi-robot transfer," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 2169–2176.

[64] B. Ramsundar, S. Kearnes, P. Riley, D. Webster, D. Konerding, and V. Pande, "Massively multitask networks for drug discovery. 2015," *arXiv preprint arXiv:1502.02072*, 2015.

[65] S. Liu, "Exploration on deep drug discovery: Representation and learning," Ph.D. dissertation, University of WisconsinMadison, 2018.

[66] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, "Cross-stitch networks for multi-task learning," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2016.433

[67] S. Ruder, J. Bingel, I. Augenstein, and A. Søgaard, "Learning what to share between loosely related tasks," *arXiv preprint arXiv:1705.08142*, 2017.

[68] S.-A. Rebuffi, H. Bilen, and A. Vedaldi, "Efficient parametrization of multi-domain deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8119–8127.

[69] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *International Conference on Machine Learning*, 2018, pp. 794–803.

[70] M. Guo, A. Haque, D.-A. Huang, S. Yeung, and L. Fei-Fei, "Dynamic task prioritization for multitask learning," in *European Conference on Computer Vision*, 2018, pp. 282–299.

[71] Y. Zhang and D.-Y. Yeung, "A convex formulation for learning task relationships in multi-task learning," in *UAI*, 2010.

[72] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," in *Advances in Neural Information Processing Systems*, 2018, pp. 525–536.

[73] J.-A. Désidéri, "Multiple-gradient descent algorithm (mgda) for multiobjective optimization," *Comptes Rendus Mathematique*, vol. 350, no. 5-6, pp. 313–318, 2012.

[74] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3234–3243.

[75] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg, "Video segmentation by tracking many figure-ground segments," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2192–2199.

[76] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool, "The 2017 davis challenge on video object segmentation," *The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*, 2017.

[77] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[78] 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 *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 3213–3223.

[79] D. de Geus, P. Meletis, and G. Dubbelman, "Single network panoptic segmentation for street scene understanding," in *IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 709–715.

[80] D. Zhang, Y. Song, D. Liu, H. Jia, S. Liu, Y. Xia, H. Huang, and W. Cai, "Panoptic segmentation with an end-to-end cell r-cnn for pathology image analysis," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2018, pp. 237–244.

[81] Q. Li, A. Arnab, and P. H. Torr, "Weakly-and semi-supervised panoptic segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 102–118.

[82] E. Xie, P. Sun, X. Song, W. Wang, X. Liu, D. Liang, C. Shen, and P. Luo, "Polarmask: Single shot instance segmentation with polar representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[83] Z. Yu, C. Feng, M.-Y. Liu, and S. Ramalingam, "Casenet: Deep category-aware semantic edge detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5964–5973.

[84] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang, "Object contour detection with a fully convolutional encoder-decoder network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 193–202.

[85] L. J. Van Vliet, I. T. Young, and G. L. Beckers, "A nonlinear laplace operator as edge detector in noisy images," *Computer vision, graphics, and image processing*, vol. 45, no. 2, pp. 167–195, 1989.

[86] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proceedings of the IEEE international Conference on Computer Vision*, 2015, pp. 1395–1403.

[87] D. Acuna, A. Kar, and S. Fidler, "Devil is in the edges: Learning semantic boundaries from noisy annotations," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[88] R. Deng, C. Shen, S. Liu, H. Wang, and X. Liu, "Learning to predict crisp boundaries," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[89] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai, "Richer convolutional features for edge detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3000–3009.

[90] G. Bertasius, J. Shi, and L. Torresani, "Deepedge: A multi-scale bifurcated deep network for top-down contour detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4380–4389.

[91] J. van den Brand, M. Ochs, and R. Mester, "Instance-level segmentation of vehicles by deep contours," in *Computer Vision – ACCV 2016 Workshops*, C.-S. Chen, J. Lu, and K.-K. Ma, Eds., Cham, 2017, pp. 477–492.

[92] H. Samet and M. Tamminen, "Efficient component labeling of images of arbitrary dimension represented by linear bintrees," *IEEE transactions on pattern analysis and machine intelligence*, vol. 10, no. 4, pp. 579–586, 1988.

[93] Y. Wu and K. He, "Group normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.

[94] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[95] J. Uhrig, M. Cordts, U. Franke, and T. Brox, "Pixel-level encoding and depth layering for instance-level semantic labeling," in *German Conference on Pattern Recognition*, 2016, pp. 14–25.

[96] M. Bai and R. Urtasun, "Deep watershed transform for instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5221–5229.

[97] S. Liu, J. Jia, S. Fidler, and R. Urtasun, "Sgn: Sequential grouping networks for instance segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3496–3504.

[98] J. Horgan, C. Hughes, J. McDonald, and S. Yogamani, "Vision-based driver assistance systems: Survey, taxonomy and advances," in *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, 2015, pp. 2032–2039.

[99] M. Heimberger, J. Horgan, C. Hughes, J. McDonald, and S. Yogamani, "Computer vision in automated parking systems: Design, implementation and challenges," *Image and Vision Computing*, vol. 68, pp. 88–101, 2017.

[100] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, 2014, pp. 568–576.

[101] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox, "Demon: Depth and motion network for learning monocular stereo," *arXiv preprint arXiv:1612.02401*, 2016.

[102] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[103] N. Märki, F. Perazzi, O. Wang, and A. Sorkine-Hornung, "Bilateral space video segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 743–751.

[104] F. Perazzi, O. Wang, M. Gross, and A. Sorkine-Hornung, "Fully connected object proposals for video segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3227–3234.

[105] L. Chen, Z. Yang, J. Ma, and Z. Luo, "Driving scene perception network: Real-time joint detection, depth estimation and semantic segmentation," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018, pp. 1283–1291.

[106] D. Neven, B. D. Brabandere, S. Georgoulis, M. Proesmans, and L. V. Gool, "Fast scene understanding for autonomous driving," 2017.

[107] S. Ruder, "An overview of multi-task learning in deep neural networks," 2017.

[108] A. R. Zamir, A. Sax, W. Shen, L. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3712–3722.

[109] H. Bilen and A. Vedaldi, "Universal representations: The missing link between faces, text, planktons, and cat breeds," *arXiv preprint arXiv:1701.07275*, 2017.

[110] L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit, "One model to learn them all," *arXiv preprint arXiv:1706.05137*, 2017.

[111] Y. Tamaazousti, "On the universality of visual and multimodal representations," Ph.D. dissertation, University College London, 2018.

[112] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017, pp. 2755–2763.

[113] G. Huang, S. Liu, L. Van der Maaten, and K. Q. Weinberger, "Condensenet: An efficient densenet using learned group convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2752–2761.

[114] "Nvidia xavier soc specification," https://en.wikichip.org/wiki/nvidia/tegra/xavier, 2018 (accessed Nov 22, 2018).

[115] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2017.690

[116] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE international Conference on Computer Vision*, 2015, pp. 2938–2946.

[117] L. Liebel and M. Körner, "Auxiliary tasks in multi-task learning," *arXiv preprint arXiv:1805.06334*, 2018.

[118] S. Toshniwal, H. Tang, L. Lu, and K. Livescu, "Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition," in *INTERSPEECH*, 2017.

[119] S. Parthasarathy and C. Busso, "Ladder networks for emotion recognition: Using unsupervised auxiliary tasks to improve predictions of emotional attributes," in *Interspeech*, 2018.

[120] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. 30, no. 2, pp. 88–97, 2009.

[121] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4340–4349.

[122] M. Wrenninge and J. Unger, "Synscapes: A photorealistic synthetic dataset for street scene parsing," *CoRR*, vol. abs/1810.08705, 2018.

[123] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3530–3538, 2017.

[124] G. Neuhold, T. Ollmann, S. R. Bulo, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes." in *ICCV*, 2017, pp. 5000–5009.

[125] S. Wang, M. Bai, G. Máttyus, H. Chu, W. Luo, B. Yang, J. Liang, J. Cheverie, S. Fidler, and R. Urtasun, "Torontocity: Seeing the world with a million eyes," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 3028–3036, 2017.

[126] S. Sankaranarayanan, Y. Balaji, A. Jain, S. N. Lim, and R. Chellappa, "Learning from synthetic data: Addressing domain shift for semantic segmentation," in *CVPR*, 2018.

[127] S. Liu, E. Johns, and A. J. Davison, "End-to-end multi-task learning with attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1871–1880.

[128] A. Mousavian, H. Pirsiavash, and J. Kosecka, "Joint semantic segmentation and depth estimation with deep convolutional networks," *2016 Fourth International Conference on 3D Vision (3DV)*, Oct 2016. [Online]. Available: http://dx.doi.org/10.1109/3DV.2016.69

[129] O. H. Jafari, O. Groth, A. Kirillov, M. Y. Yang, and C. Rother, "Analyzing modular cnn architectures for joint depth prediction and semantic segmentation," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2017.7989537

[130] A. Gurram, O. Urfalioglu, I. Halfaoui, F. Bouzaraa, and A. M. Lopez, "Monocular depth estimation by learning from heterogeneous datasets," *2018 IEEE Intelligent Vehicles Symposium (IV)*, Jun 2018. [Online]. Available: http://dx.doi.org/10.1109/IVS.2018.8500683

[131] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[132] C. Hazirbas, L. Ma, C. Domokos, and D. Cremers, "Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture," in *Asian Conference on Computer Vision*, 2016, pp. 213–228.

[133] M. Siam, S. Valipour, M. Jägersand, N. Ray, and S. Yogamani, "Convolutional gated recurrent networks for video semantic segmentation in automated driving," *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–7, 2017.

[134] G. Sistu., S. Chennupati, and S. Yogamani., "Multi-stream cnn based video semantic segmentation for automated driving," in *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP,*. SciTePress, 2019, pp. 173–180.

[135] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *CVPR*, 2014.

[136] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4161–4170.

[137] L. Ma, J. Stückler, C. Kerl, and D. Cremers, "Multi-view deep learning for consistent semantic mapping with rgb-d cameras," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 598–605.

[138] S. Sah, "Multi-modal deep learning to understand vision and language," Ph.D. dissertation, Rochester Institute of Technology., 2018.

[139] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei, "Flow-guided feature aggregation for video object detection," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[140] H. Rashed, A. El Sallab, S. Yogamani, and M. ElHelw, "Motion and depth augmented semantic segmentation for autonomous navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[141] J. Lee and J. Nam, "Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging," *IEEE signal processing letters*, vol. 24, no. 8, pp. 1208–1212, 2017.

[142] R. Ranjan, V. M. Patel, and R. Chellappa, "Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 1, p. 121135, Jan 2019. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2017.2781233

[143] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[144] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[145] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi, "Human action recognition using factorized spatio-temporal convolutional networks," *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2015.522

[146] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," *2015*

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun 2015. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2015.7299101

[147] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, p. 677691, Apr 2017. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2016.2599174

[148] R. M. Oruganti, S. Sah, S. Pillai, and R. Ptucha, "Image description through fusion based recurrent multi-modal learning," in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3613–3617.

[149] S. Sah, S. Kulhare, A. Gray, S. Venugopalan, E. Prud'Hommeaux, and R. Ptucha, "Semantic text summarization of long videos," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017, pp. 989–997.

[150] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville, "Describing videos by exploiting temporal structure," in *Advances in Neural Information Processing Systems*, 2015.

[151] S. Sharma, R. Kiros, and R. Salakhutdinov, "Action recognition using visual attention," in *Proceedings of the IEEE international Conference on Computer Vision*, 2015, pp. 4507–4515.

[152] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[153] H. Song, W. Wang, S. Zhao, J. Shen, and K.-M. Lam, "Pyramid dilated deeper convlstm for video salient object detection," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[154] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[155] S. Kulhare, S. Sah, S. Pillai, and R. Ptucha, "Key frame extraction for salient activity recognition," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, Dec 2016, pp. 835–840.

[156] R. H. Hahnloser and H. S. Seung, "Permitted and forbidden sets in symmetric threshold-linear networks," in *Advances in Neural Information Processing Systems*, 2001, pp. 217–223.

[157] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning," *MIT Press*, pp. 189–191, 2016, http://www.deeplearningbook.org.

[158] M. Menze, C. Heipke, and A. Geiger, "Joint 3d estimation of vehicles and scene flow," in *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015.

[159] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," pp. 3061–3070, 2015.