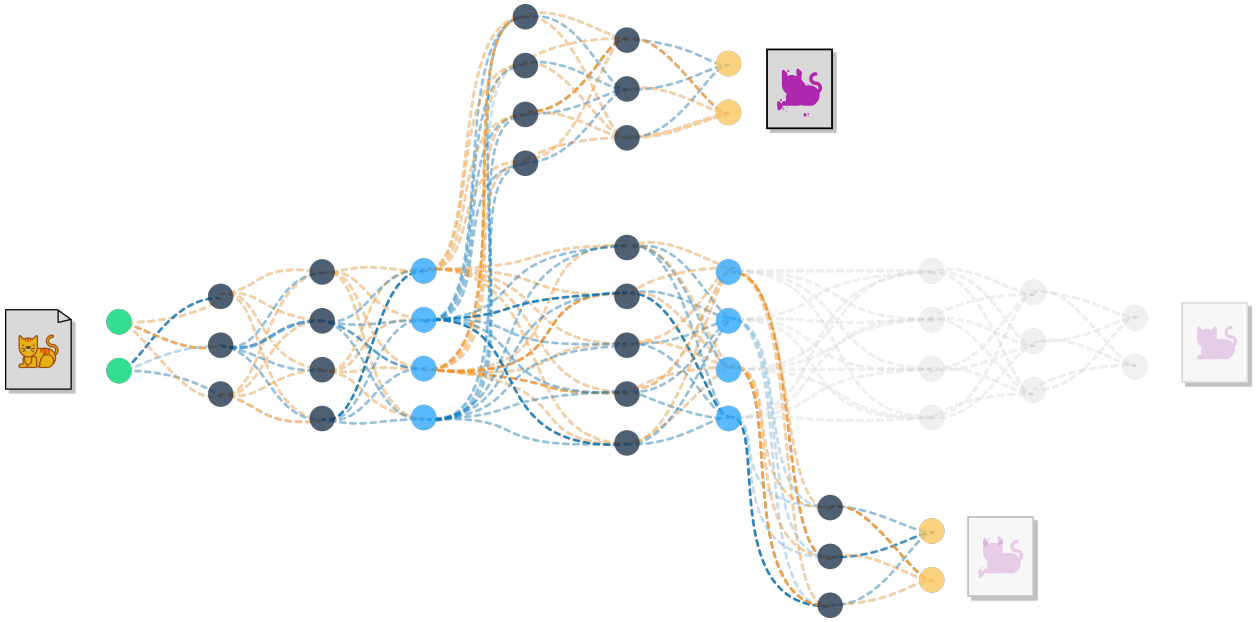
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



Low latency video analytics system with multi-exit neural networks

Input adaptive inference optimization for real-time semantic segmentation

Master's thesis in Computer science and engineering

NEETHU HARINDRAN

BHARATH POOJARY

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2022

Master's thesis 2022

Low latency video analytics system with multi-exit neural networks

Input adaptive inference optimization for real-time semantic segmentation

NEETHU HARINDRAN

BHARATH POOJARY

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2022

Multi-exit neural networks
Input adaptive inference optimization for
real-time semantic segmentation
NEETHU HARINDRAN
BHARATH POOJARY

Supervisor: Ahmed Ali-Eldin Hassan, Department of Computer Science
Advisor: Per-Lage Götvall and Erik Brorsson, Volvo Lastvagnar AB
Examiner: Philippas Tsigas, Department of Computer Science

Master's Thesis 2022
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Description of the picture on the cover page (if applicable)

Typeset in LaTeX
Gothenburg, Sweden 2022

Multi-exit neural networks
Input adaptive inference optimization for
real-time semantic segmentation
NEETHU HARINDRAN
BHARATH POOJARY
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Computer vision-based control systems have become increasingly powerful and promising in tackling real-world problems. This can be accredited to the use of deep learning methods in these systems with state-of-the-art performance sometimes outperforming humans in tasks which require subjective decision making. This has resulted in increased interest in these systems from Swedish industry, including Volvo. One example system where these systems are used is the Volvo GPSS system, where semantic segmentation is used to perform real-time decisions based on pixel level classification of a monitored area. However, such systems frequently deal with a trade-off between latency and accuracy. This is primarily due to the increasing number of model layers being used to develop Deep-Neural-Network models for vision systems, resulting in equal resource utilization regardless of input complexity. In this thesis, we develop an approach that employs input adaptive multi-exit strategy to exploit latency benefits of dynamic processing based on the input complexity. The proposed approach aims to have a reduced average inference time as the simple input samples takes an early exit and only the complex samples need more computation offered by all the model layers. The open source CityScapes dataset and the Volvo dataset were used in a number of multi-exit semantic segmentation experiments with HRNet architecture chosen as the backbone. The thesis work studies three novel exit strategies, including reinforcement learning, auxiliary models, and fast Fourier transform. Out of all the methods examined, the reinforcement learning-based exit strategy displayed the best performance advantages, with accuracy on par with unbranched HRNet and a significant decrease in latency and computation.

Keywords: Multi-exit Neural Networks, Input Adaptive Inference, Semantic Segmentation, Inference Optimization

Acknowledgements

We want to thank Ahmed Ali-Eldin Hassan, Erik Brorsson and Per-Lage Götvall for the great support and guidance they've provided throughout the project. We also want to thank Volvo GTO for providing us with resources to carry out our research. A special thanks to Bassam Kaddoura and Cao Do for providing annotated Volvo factory dataset for our experiments.

Neethu Harindran, Gothenburg, November 2022
Bharath Poojary, Gothenburg, November 2022

Contents

List of Figures **xi**

List of Tables **xiii**

1 Introduction **1**
1.1 Overview . . . 1
1.2 Problem Statement . . . 2
1.3 Challenges . . . 3
1.4 Related works . . . 3
1.5 Our contribution . . . 4

2 Background **6**
2.1 Multi-exit neural networks . . . 6
2.2 Semantic Segmentation . . . 7
2.3 Multi-exit approach on Semantic Segmentation . . . 9
2.4 Reinforcement Learning . . . 10
2.5 Fourier Transform for Images . . . 11

3 Methods **13**
3.1 Evaluation Metrics . . . 13
3.1.1 Accuracy . . . 13
3.1.2 Latency . . . 14
3.1.3 Floating Point Operations . . . 14
3.2 Multi-exit Classification Model . . . 14
3.3 Semantic Segmentation . . . 16
3.4 Multi-exit Semantic Segmentation . . . 18
3.4.1 Choosing the exit points . . . 18
3.4.2 Design of segmentation heads and the additional trainable blocks 18
3.4.3 Selection and implementation of exit methodology . . . 19
3.4.4 Exit strategy based on Entropy . . . 19
3.5 Proposed novel Exit Strategies . . . 20
3.5.1 Exit strategy based on Fast Fourier Transform . . . 20
3.5.2 Exit strategy with an Auxiliary Network . . . 21
3.5.3 Exit strategy through Reinforcement Learning . . . 24
3.6 Effects of Dataset Variation . . . 25

4 Results **26**

4.1 Multi-exit Classification Model 26
4.2 Multi-exit Semantic Segmentation 27
4.2.1 Entropy based exit strategy 30
4.2.2 Analysis of exits with entropy 31
4.2.3 Analysis of exits with FFT 32
4.2.4 FFT based exit strategy 33
4.2.5 Auxiliary model based exit strategy 34
4.2.5.1 Classification model 34
4.2.5.2 Regression model 35
4.2.6 Reinforcement learning based exit strategy 36
4.2.6.1 Entropy states 37
4.2.6.2 FFT states 37
4.2.7 Comparison of exit strategies 38

5 Conclusion **41**
5.1 Discussion 41
5.1.1 Multi-exit Classification Models 41
5.1.2 Multi-exit Semantic Segmentation Models 41
5.1.2.1 Entropy based exit strategy 41
5.1.2.2 FFT based exit strategy 42
5.1.2.3 Auxiliary model-based exit strategy 42
5.1.2.4 Reinforcement learning based exit strategy 43
5.2 Conclusion 43
5.2.1 Future works 43

Bibliography **45**

List of Figures

2.1 Multi-exit neural network with two exit branches [27] 7
2.2 Multi-exit semantic segmentation architecture 10
2.3 Reinforcement Learning setup 11

3.1 Intersection Over Union 13
3.2 Branched LeNet architecture 15
3.3 HRNet architecture [28] 17
3.4 HRNetV2 representation [28] 17
3.5 Multi-exit HRNet Architecture 19
3.6 Entropy based early exit 20
3.7 FFT based early exit 21
3.8 Multi-exit model with auxiliary network 22
3.9 Auxiliary classification model 22
3.10 Auxiliary regression model 23
3.11 Multi-exit reinforcement learning model pipeline 24
3.12 states in RL based Multi-exit network 25

4.1 Accuracy vs Entropy 26
4.2 Results of classification models 27
4.3 Results of HRNet-18 for CityScapes dataset 28
4.4 Results of HRNet-18 for Volvo dataset 28
4.5 Branched HRNet Inference on Cityscapes 29
4.6 Branched HRNet Inference on Volvo Dataset 30
4.7 Results of entropy based exit strategy for Cityscape dataset 31
4.8 Results of entropy based exit strategy for Volvo dataset 31
4.9 Analysis of exits with entropy for Cityscape dataset 32
4.10 Analysis of exits with entropy for Volvo dataset 32
4.11 Analysis of exits with FFT for Cityscape dataset 33
4.12 Analysis of exits with FFT for Volvo dataset 33
4.13 Results of FFT based exit strategy for Cityscape dataset 34
4.14 Results of FFT based exit strategy for Volvo dataset 34
4.15 Results of auxiliary classification model based exit strategy for Cityscape dataset 35
4.16 Results of auxiliary classification model based exit strategy for Volvo dataset 35
4.17 results of auxiliary regression model based exit strategy for Cityscape dataset 36

4.18 Results of auxiliary regression model based exit strategy for Volvo dataset 36
4.19 Results of RL based exit strategy with entropy states for Cityscapes dataset 37
4.20 Results of RL based exit strategy with entropy states for Volvo dataset 37
4.21 Results of RL based exit strategy with FFT states for Cityscapes dataset 38
4.22 Results of RL based exit strategy with FFT states for Volvo dataset . 38
4.23 Results Comparison for CityScapes dataset 39
4.24 Results Comparison for Volvo dataset 40

List of Tables

3.1 Comparison between semantic segmentation models 16

4.1 Comparison between branched and un-branched classification models 27
4.2 Performance comparison for CityScapes dataset 39
4.3 Performance comparison for Volvo dataset 40

1
Introduction

In the ever-growing world of data, there is a need for algorithms that learn to perform tasks with minimal human intervention. These can uncover complex relations in the data that might go unnoticed by the manual engineering of the solutions. Considering the latest advancements in artificial intelligence, machine learning and deep learning appears to be the most popular concepts. However, Deep Learning has recently gained prominence as a result of its supremacy in terms of performance.

1.1 Overview

Deep learning is a subset of the larger machine learning family. It is built on neural networks, which attempts to emulate the functioning of the human brain by learning from large amounts of data. Amongst machine learning algorithms, deep learning techniques take the lead in terms of performance, due to the minimal effort required in engineering the features that have good correlation with the parameters affecting the decision. Deep learning encompasses these feature extractions automatically by learning and building the features using the provided data. Another reason for the popularity of deep learning is the recent advancements in hardware and computing power. The advent of Graphics Processing Units (GPUs) with parallel computing capabilities has allowed deep learning algorithms to be trained much faster and with copious amounts of data. This has made deep learning even more powerful than it was before and has increased its demand. Deep learning methods can scale well when compared to traditional learning algorithms. Deep learning is also remarkably successful in fields where there are enormous amounts of unlabelled data, such as computer vision. This is because deep learning algorithms can learn the features automatically from the data with little human intervention. Deep learning is also successful when there is a need to learn from a large amount of data in a short amount of time.

But there is another side to this story, the improved performance comes at the cost of increased number of parameters and model complexity. Most state-of-the-art deep learning models use a lot of parameters and excessive computational resources. What started with models with couple of parameters has now reached models like GPT3 [3] having billions of parameters in the race to achieve human level performance. This approach is not scalable in the long term and is not very efficient. When discussing these topics, there are a few important questions that needs to be considered: are these tremendous amounts of computation justified? Do we require the same amount of computation for every example, regardless of if it is simple or

complex?
This thesis project is an industrial thesis that was done in Volvo Group. Volvo's vision is to create a future factory where robots and humans can work and collaborate efficiently on equal terms - a Unified Environment [2]. This is crucial from the perspective of flexible manufacturing and optimal usage of skill sets of both humans and robots. A critical component of this system is a central sensory system that monitors the movement of humans and robots in the environment and ensures humans' safety. This will be accomplished by analyzing video streams from potentially hundreds of ceiling-mounted cameras that monitor the area in real-time. Deep learning-based approaches dominate the core implementation of this vision [12] data processing system, with performance comparable to humans. Though DNN systems have superior performance in terms of accuracy, they tend to suffer [8] [15] from latency issues due to their complex architecture, which limits their use in real-time applications. Furthermore, these systems are typically deployed on edge cloud servers [4] [1] at the factory to optimize the sharing of computational resources (e.g., GPUs). As a result, it is critical to develop a solution that meets the low-latency requirements while utilizing minimal computational resources.
At Volvo, semantic segmentation [30] is a primary component of the video analytic system that help to identify humans, obstacles, and the drivable part of the factory floor that help to guide the movements of the Autonomous Transport Robots (ATRs). This thesis will focus on the implementation of the early exit architecture on semantic segmentation algorithms and aims to provide solutions for semantic segmentation architectures tailored for stringent latency constraints and efficient resources sharing problems without any major degradation in the prediction accuracy.

1.2 Problem Statement

Most deep neural networks designed to perform semantic segmentation achieves impressive accuracy at the cost of increased inference time as well as energy requirements hindering them from being used in many real-world time-critical applications. As semantic segmentation plays a significant role in identifying the barriers on the factory floor, optimisation of the semantic segmentation architectures for speed becomes crucial, typically when the system must be implemented on a resource constrained platform. The key objectives of this thesis project are as follows.

- The primary objective of the thesis is to develop a multi-exit architecture for semantic segmentation. This entails choosing a publicly available dataset as well as a cutting-edge semantic segmentation framework. The goal is to revamp the present architecture by adding intermediate branches to the main branch and integrating it into a multi-exit workflow.
- The next objective is to fine-tune the multi-exit semantic segmentation model that has been created upon the availability of synthetic/simulated data that represents real-time data from the factory's ceiling-mounted cameras. This involves measuring the impact and evaluating the robustness of the model with respect to the change in dataset.
- Finally, we intend to provide a standard training procedure for multi-exit architecture in order to improve the approach's adaptability in terms of ap-

plication to any semantic segmentation architecture. This is accomplished by fine-tuning the training process iteratively until it can be utilized to convert any single-branched model to a multi-exit model.

1.3 Challenges

Semantic segmentation is one of the fine grained vision tasks for understanding the scene and parsing it. It provides a pixel level approach to classify objects and recognizes their shape in the scene. State-of-the-art semantic segmentation models involve the challenge of maintaining high resolution feature maps throughout the network due to the every pixel nature of segmented output. At the same time, it needs to maintain a large receptive field on the output to capture more spatial context. Thus, it is required to have down sampling steps to capture semantic/contextual information as well as up sampling steps to recover spatial information. Thus, the semantic segmentation architectures require numerous layers which resulted in it's bulky nature as well as latency intensive inference.
We aim to improve this latency burden through a multi exit semantic segmentation architecture. Input dependent adaptive inference is an area widely explored in researches in the context of image classification. However, due to the heavy weight nature of segmentation heads, applying a multi-exit approach to a semantic segmentation model is rather challenging. The addition of a single segmentation head has been seen to result in a greater overhead. In addition, the dense output of semantic segmentation models makes the exit strategy more difficult. Through this work, we intend to choose a base semantic segmentation architecture and add multiple exit branches to it. We train the model in such a way that the workload is pushed to the earlier layers of the network, allowing for efficient inference depending on the complexity of the input sample. The resulting design should allow the speed-accuracy trade off to be tailored to the need of the system.

1.4 Related works

Multi-exit neural network architectures for improving feed forward inference efficiency has received a lot of attention from the research community. Panda et al. proposed a conditional deep learning (CDL) technique in which the difficulty of identifying input samples are observed by iteratively adding linear classifiers to each convolutional layer [20]. The output of each linear classifier is observed to decide whether classification can be terminated at the current stage or not, thereby conditionally activating the deeper layers of the network. The proposed methodology enables to adjust the computational effort based on the difficulty of the input data while maintaining the required accuracy. Another novel architecture that works on a similar principle is Branchynet [27]. Branchynet is a multi exit deep neural network which has several side branches/exits added to the main branch to allow some of the test samples to exit early. The intermediate branch network is equipped with additional layers at each exit point. It is based on the observation that majority of the test samples can be correctly inferred by the features learnt at an earlier stage.

By eliminating layer-by-layer processing of the majority of test data through early exits, Branchynet helps to significantly reduce the inference time. Both Branchynet and CDL addresses classification task but the architecture is general and can be extended to semantic segmentation as well.
There has been several works focussing on light weight semantic segmentation models emerging as possible solutions to the stringent latency constraints and for efficient resource sharing. Kouris et al. [16] introduced a Multi-Exit Semantic Segmentation (MESS) network which adopts a similar principle like Branchynets by attaching several early exits (i.e. segmentation heads) on the CNN backbone. It allows efficient inference based on the difficulty of the input and takes into account the target device's capabilities. It adopts a two stage training mechanism which helps to boost the accuracy. In the first stage, the segmentation heads are attached to the exit points and the network is trained end to end. It is followed by the individual training of the early exit architectures connected to exit points with the back bone and the final exit kept frozen. The potential of knowledge distillation[21] [31] is also utilized in optimizing the training of early exit branches.
There are several works which adopt the principle of anytime inference which address the latency constraints by introducing flexibility to model computation. Such a model makes a number of intermediate predictions between the first and last exits. The prediction continues to the final one if time remains or stops at a point if the results are satisfactory or the task runs out of a preset time interval.. Liu et al. [17] proposes the anytime inference approach to pixel level visual recognition tasks like semantic segmentation reducing the total computations without degrading accuracy. The proposed model has multiple predictors that branch from the intermediate stages of the model. Both the original exit and intermediate exits are trained end-to-end. For each exit, there is an encoder-decoder architecture that combines pooling, convolution, and interpolation to enlarge receptive fields and smooth spatial noise. The work proposes spatially adaptive inference, taking advantage of spatial structure and makes predictions for different pixels at different points in time. It decides whether or not to continue computations at each exit and position. The entropy of class predictions can be used as a measure of confidence for semantic segmentation. The output of each exit is masked by thresholding the confidence of its predictions such that further computations are reduced for sufficiently confident pixels. This confidence adaptivity can drastically reduce the total computation while maintaining accuracy. But, to achieve the performance improvements, the technique uses pixel-wise exclusion, which necessitates hardware tuned for sparse convolution, which a standard GPU is not.

1.5 Our contribution

The works presented in the preceding section primarily represent the core ideas of the branching architecture's tasks. Our work in this thesis will be focused on semantic segmentation of classes relevant to the goals of Volvo's central vision system. Our implementation will be tailored to the system's real-time requirements, and to use inputs from a ceiling-mounted camera array.
Prior works also prioritize budgeted computing as their primary goal, intending to

exit the network early due to the resource constraints of the underlying system. Our primary goal, on the other hand, will be to dynamically modify the computational graph based on the complexity of the input samples. Together with this, our optimization will meet the requirements of the system specifications intended for deployment on edge cloud clusters.
Additionally, the thesis explores various approaches for designing an exit strategy to manage well the trade-off between latency and accuracy for multi-exit networks. As in the earlier works, the exit decisions are initially made using the most popular entropy metric. However, we choose the entropy threshold value dynamically while the model was being trained, taking into account the expected performance guaranteed by the system. In order to further refine the exit strategy, the prediction at each exit point was analysed in the frequency domain. The fast fourier transform of the prediction was utilized to make the exit decision which was proven to perform better than entropy based exit strategy. Furthermore, this work revealed two sophisticated methodologies for exit decision making: employing an auxiliary network and applying reinforcement learning. These methods make sure that the input is only processed further by the network when a performance gain is expected, minimising unnecessary computation expenses.

2
Background

The thesis aims to optimize multi-exit DNNs for semantic segmentation by dynamically choosing which exit should be taken based on the image, reducing the overall latency for semantic segmentation inference. The theoretical underpinning for the primary concepts explored in the thesis is presented in this chapter. The first two sections briefly cover the multi-exit network architecture and semantic segmentation models, which range from the earliest to the most modern. The multi exit approach on a semantic segmentation model is explained in the third section. The fourth and the fifth sections explains the concepts of reinforcement learning and fourier transform of images respectively which are the novel techniques employed for an efficient exit strategy in the multi exit model.

2.1 Multi-exit neural networks

Deep neural networks have shown state-of-the-art performance in computer vision tasks due to its ability to extract features more effectively with the increase in the number of layers. The expansion of neural networks from a few layers to hundreds of layers has been facilitated by hardware advancements and the emergence of novel learning algorithms. However, there is a downside to these performance benefits with the additional number of layers in terms of the increased latency and energy consumption. This drawback becomes critical when it comes to real-time resource constrained applications.

One of the strategies used to address this issue is to add multiple exits to a deep neural network such that it can make predictions at intermediate points of the stack of layers. In multi-exit architecture, side branches are added to the base neural network allowing the test samples to exit early. It is based on the notion that most of the data samples can be correctly inferred with the features learned by the earlier layers of the network. As a result of utilizing the earlier exits, all data samples do not need to be processed by all layers of the network resulting in a considerable reduction in average inference time. Figure 2.1 shows a multi-exit neural network. Furthermore, it has been observed that multi exit neural networks aids in prevention of overfitting and mitigation of vanishing gradients as well [27].

There are different training approaches adopted for multi-exit architectures. The common or straight forward way is to treat it as a joint optimization problem by combining the losses from all the exit points [27]. Another strategy is to train the backbone network first, followed by training the early exits independently [16]. Once the network is trained, data samples can make early exits depending on how

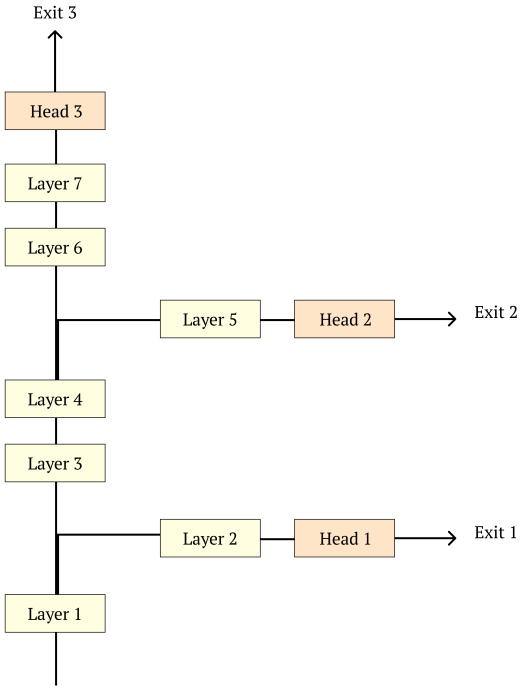


Figure 2.1: Multi-exit neural network with two exit branches [27]

confident the network is its predictions. For a classification problem, entropy of the classification result is considered as a measure of confidence. If the entropy value at an early exit is below a threshold value, the exit is confident in its predictions and the prediction results are returned without additional processing. If the entropy value at an early exit exceeds the threshold value, the exit is not confident in its predictions and the sample continues to the next exit. The predicted output is always returned when the sample reaches the last/main exit of network.

2.2 Semantic Segmentation

Image segmentation [7] [29] is a computer vision task in which each and every pixel in an image is assigned a label. It differs from image classification where a single label is assigned for an entire picture. Semantic segmentation [9] and instance segmentation [10] are the two types of image segmentation. Semantic segmentation treats multiple objects of the same class as a single entity whereas, instance segmentation treats multiple objects of the same class as distinct individual objects. Prior to the advent of deep learning, image processing techniques such as grey level segmentation [22], conditional random fields [25], and others were used to segment an image into separate regions of interest. Today, deep learning methods [19] are considered to be the most effective for semantic segmentation due to the impressive quality of the segmented outputs.
The purpose of semantic segmentation is to create a segmentation map from an RGB or greyscale image, with each pixel represented by a label. For image classification, the deep neural networks learn the low level concepts in the earlier layers while the

later layers develop the high level feature mappings by increasing the number of channels. The computational burden is reduced by down sampling the feature maps through pooling or strided convolutions [5]. However, for semantic segmentation, in order for the model to give a full resolution semantic prediction, the input image's resolution must be preserved. One of the popular architectures used for semantic segmentation is fully convolutional network (FCN) [18]. In FCN, the input image is downsampled into a smaller size through a series of convolution operations which is termed as encoding. The encoded output is then upsampled to a full resolution segmentation map either through bilinear interpolation [14] or a series of transpose-convolutions. This process is termed as decoding. Most of the segmentation models follow this encoder-decoder architecture.

U-Net [24] design was developed to increase the performance quality of basic FCN architecture by expanding the capacity of the decoder module. U-Net is a symmetric architecture with skip connections between downsampling and upsampling paths. There are connections from the output of the convolution blocks to the corresponding input of the transposed-convolution blocks at the same level. With these skip connections, the image at the decoding path is concatenated with the corresponding image at the encoding path which results in a more precise prediction. With the introduction of Densely Connected Convolutional Networks (DenseNets) [11], a novel CNN architecture that has shown good results on image classification tasks, a new model for semantic segmentation called Tiramisu [13] was introduced. A Dense Block consists of several layers of convolutions where each layer is directly connected to every other layer in a feed-forward fashion. This improves the network's accuracy and makes it easier to train. The Tiramisu model is created by replacing the regular convolution blocks of the U-Net model with these dense blocks. The resulting network is said to be parameter efficient and allows for efficient feature reuse [13].

For efficient semantic segmentation, several deep learning models make advantage of multi scale approaches. PSPNet (Pyramid Scene Parsing Network) [32] is one such example, in which the input image is fed to a CNN to get the feature map of the last convolutional layer. Then, a pyramid parsing module performs the pooling operation (max or average) using four different kernel sizes and strides to the output feature map of CNN. It is followed by upsampling and concatenation layers to form the final feature representation. Downsampling feature maps, as we know, broadens the receptive field, but the broader context comes at the expense of reduced spatial resolution. Dilated (atrous) convolutions is an alternative approach that provides a wide field of view while preserving the full spatial resolution. By adjusting the dilation rate, the same filter has its weight values spread out farther in space which enables it to learn more global context. DeepLabv3 [6] is a semantic segmentation model that combines atrous convolutions with varying dilation rates to extract information from several scales while preserving image resolution. They experiment with employing atrous convolutions in a cascaded manner and also in a parallel manner in the form of Atrous Spatial Pyramid Pooling [6]. However, it is still considered computationally expensive to completely replace pooling layers with dilated convolutions.

HRNet is a state-of-the-art neural network in the field of semantic segmentation. HRNet [28] stands for high resolution network referring to the high resolution of the

images processed. As we know, high resolution representation plays a key role in the task of semantic segmentation, HRNet maintains the high resolution representation of the input image by connecting high-to-low resolution convolutions in parallel, where there are repeated multiscale fusions across parallel convolutions. HRNet has four stages, each with high to low resolution convolutions running in parallel. There is a complete connection to the multi resolution group of the next level at the end of each stage. The biggest advantages of HRNet is that it is semantically strong and spatially precise.
Pixel-wise cross entropy loss is the most popular loss function for semantic segmentation. The semantic segmentation labels, we know, are the same size as the original image. As the labels can be represented in a one-hot encoded form, they can be used directly as the ground truth to calculate cross entropy. Pixel-wise cross entropy loss examines each and every pixel individually comparing the class predictions to the one hot encoded target vector. Pixel wise loss is calculated as the log loss, which is then averaged over all possible classes using the formula 2.1. This is repeated for all the pixels and then averaged.

$$Loss = \sum_{classes} y_{true} * log(y_{pred}) \tag{2.1}$$

2.3 Multi-exit approach on Semantic Segmentation

This thesis study aims to implement an efficient semantic segmentation architecture suitable for the use case at the Volvo factory for which we propose a framework that converts a state-of-the-art semantic segmentation model to a multi exit semantic segmentation model. By referring to and employing the multi exit concepts in the related works, early exits are employed along the depth of the original model in order to save computation during inference on easier samples. The approach includes selection of a base model compatible with the accuracy requirements of the system and using it to design a final prediction system with multiple exits.
The multi-exit semantic segmentation model is segregated into

1. **Segmentation backbone**: The segmentation backbone is the layer that is in charge of extracting the system's basic features or downsampling the input. These are often a collection of convolutional layers that produce a low-dimensional spatial feature mapping of the input image. The early departure branches will emerge from the backbone's intermediary points and the segmentation backbone makes a key contribution to the system's total latency.
2. **Segmentation head**: The processed data is upsampled to the resolution of the input image in this section of a segmentation network. The segmentation head attached to each early exit will be customised to generate outputs corresponding to the resolution of the input image.

The draft architecture of the model is depicted in the figure 2.2, where the model's backbone will be broken into multiple sections and individually customized exit branches are connected to these branching points. Main component of these branches

will be the segmentation heads. During the inference, a chosen confidence metric will be used to identify the exit strategy of the model based on predetermined thresholds.

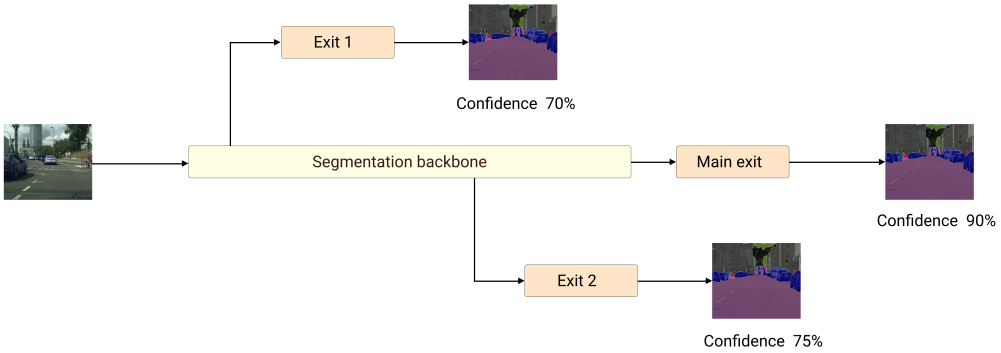


Figure 2.2: Multi-exit semantic segmentation architecture

2.4 Reinforcement Learning

Reinforcement learning is a type of machine learning that has gained popularity in recent years. It is a learning approach that falls between supervised and unsupervised learning. Reinforcement learning, unlike supervised learning, is not based on a set of labelled training data; instead, the learning algorithm focuses on maximising a reward. The agent and the environment are the two essential components of the reinforcement learning technique. The reinforcement learning algorithm is represented by the agent, and the environment is the context in which the agent is acting on. The agent placed in an unknown environment uses trial and error method in order to figure out the environment and come up with an outcome. In the absence of a training dataset, it is bound to learn from its experiences. To attain its ultimate goal, the agent must determine the "right" actions to take in different situations [26].

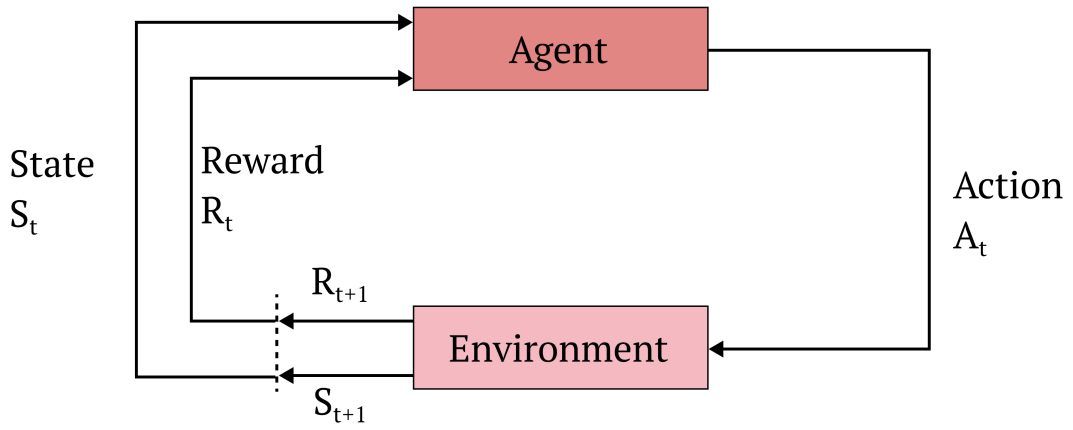


Figure 2.3: Reinforcement Learning setup

Reinforcement learning starts when the environment sends its state to an agent and then, the agent takes some actions based on the observations. The environment sends the updated state and a respective reward back to the agent. Reward is an instant return from the environment to appraise the last action taken by the agent. The right action depends on the reward which indicates how well the agent is doing at a particular point of time. The agent updates its knowledge from the reward returned by the environment and uses it to evaluate its previous action. It makes a decision on which actions to take based on the reward and the observed state. The loop keeps executing with the overall goal of maximising total future reward. Reward maximisation theory states that an RL agent must be trained in such a way that it takes the best action so that the reward is maximum. So, reinforcement learning needs to consider exploitation and exploration with a balance between exploring new information and using known information to maximise reward [26].

2.5 Fourier Transform for Images

Fourier transform is a mathematical transform that converts a function or signal to an alternate representation of sine and cosine functions of varying frequencies. It decomposes functions depending on space or time into functions depending on spatial frequency or temporal frequency. An image can be considered as a function that varies across the two dimensional space. Using fourier transform, it can be decomposed into it's sine and cosine components. The input image which is represented in spatial domain is transformed into its equivalent frequency domain representation through Fourier transform, i.e. after the transformation, each point on the image represents a particular frequency contained in the spatial domain image. The discrete fourier transform for a digital image of size N×N is given by formula 2.2 where f(i,j) is the image in the spatial domain and the exponential term is the function that corresponds to each point F(k,l) in the fourier space [23].

$$F(k,l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j) e^{-t2\pi(\frac{ki}{N} + \frac{lj}{N})} \tag{2.2}$$

The Fourier transform represents the output image as a complex number having magnitude and phase components that can be displayed as two images . In the context of image processing, only the magnitude of the Fourier Transform plays an important role , as it contains information of the geometric structure of the image in spatial domain. It is extremely useful for identifying the features of the image which are not clearly visible in the spatial domain [23].

3
Methods

This chapter explains the main methodologies adopted in this thesis, organized based on the experiments performed. It begins with the evaluation metrics chosen to assess the performance of the implementation. The technical specifics of a multi-exit classification model designed to evaluate the performance advantages offered by the multi-exit strategy are presented in the next section. The architecture of the chosen semantic segmentation model is explained next, followed by the multi-exit approach on the chosen model. The final sections describe the implications of using the Volvo factory data, and the standard branching pipeline that can be used on any deep learning model.

3.1 Evaluation Metrics

Evaluation metrics are used to quantify and compare the performance of deep learning models. It is crucial to choose multiple appropriate evaluation metrics to measure the model's performance to ensure it is operating at its best. This section describes the selected metrics for evaluating the models in this thesis and the appropriate methods to measure these parameters without impacting the neural network's performance.

3.1.1 Accuracy

Accuracy is an important metric for evaluating classification models. It is the ratio of the number of correct predictions made by the model to the total number of predictions. To measure the accuracy of a semantic segmentation model, the best way is to check the similarity of the output generated with the ground truth. Mathematically, it is done by calculating Intersection Over Union (IoU) between the two as shown in figure 3.1.

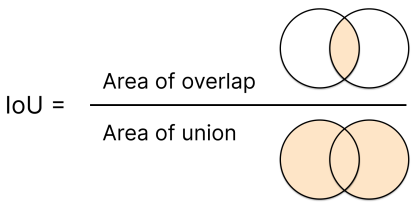


Figure 3.1: Intersection Over Union

3.1.2 Latency

Latency/inference time is the time taken by the model to make predictions. For the application at Volvo factory, the focus of our is to reduce the inference time as much as possible without degrading accuracy, for which an accurate measure of latency / inference time is extremely important. In the deep learning context, as the GPU operations are asynchronous by default, the regular approach of using python time library results in inaccurate inference time measurement. So, CUDA events are used to measure time accurately on the GPU. CUDA events are synchronization markers that can be used to monitor the device's progress and accurately measure model inference time. It aids to perform synchronization between GPU and CPU so that the time measurement takes place only after the process running on the GPU finishes, which overcomes the issue of unsynchronized execution. Thus, CUDA events are used to measure the inference time in all the experiments conducted.

3.1.3 Floating Point Operations

Floating point operations describe how many operations are required to run a single instance of a given deep learning model. It gives a hardware independent measure of the complexity of the model. A light weight core library fvcore is used to accurately measure the floating point operations of a model. fvcore contains a flop counting tool for pytorch models which provides both operator level and module level flop counts. It displays the flop counts in a hierarchical way and helps to analyse the model easily. Given a model and the inputs passed to the model, fvore can return the total flops, flop counts for all sub modules and flop counts over different operator types. fvcore is used to claculate the flops count for all the experiments in this thesis.

3.2 Multi-exit Classification Model

To study the performance benefits of multi-exit approaches, we first studied their performance benefits when applied to two classification models, LeNet and Branched Lenet. The dataset used for the experiments is MNIST dataset. Lenet architecture consists of two convolution layers followed by a Rectified Liner Unit (ReLU) and max pooling layers and 3 fully connected dense layers. For branched Lenet, an early exit is added after the first convolution, ReLU and max pooling layer. The early exit branch consists of one fully connected dense layer which generates the predictions. Branched Lenet is shown in figure 3.2.
During training the models, we use the most common loss function for classification task, softmax cross entropy. Let y be the one-hot ground-truth label vector, x be an input sample, $\hat{y}$ be the predicted vector and C be the set of all possible labels, the loss function is given by the formula 3.1.

$$L(\hat{y}, y) = -\frac{1}{|C|}\sum_{c \in C} y_c log\hat{y}_c \tag{3.1}$$

To train the branched Lenet, we treat it as a joint optimization problem as a weighted sum of loss functions of each exit branch given by formula 3.2, where w_1 and w_2 are

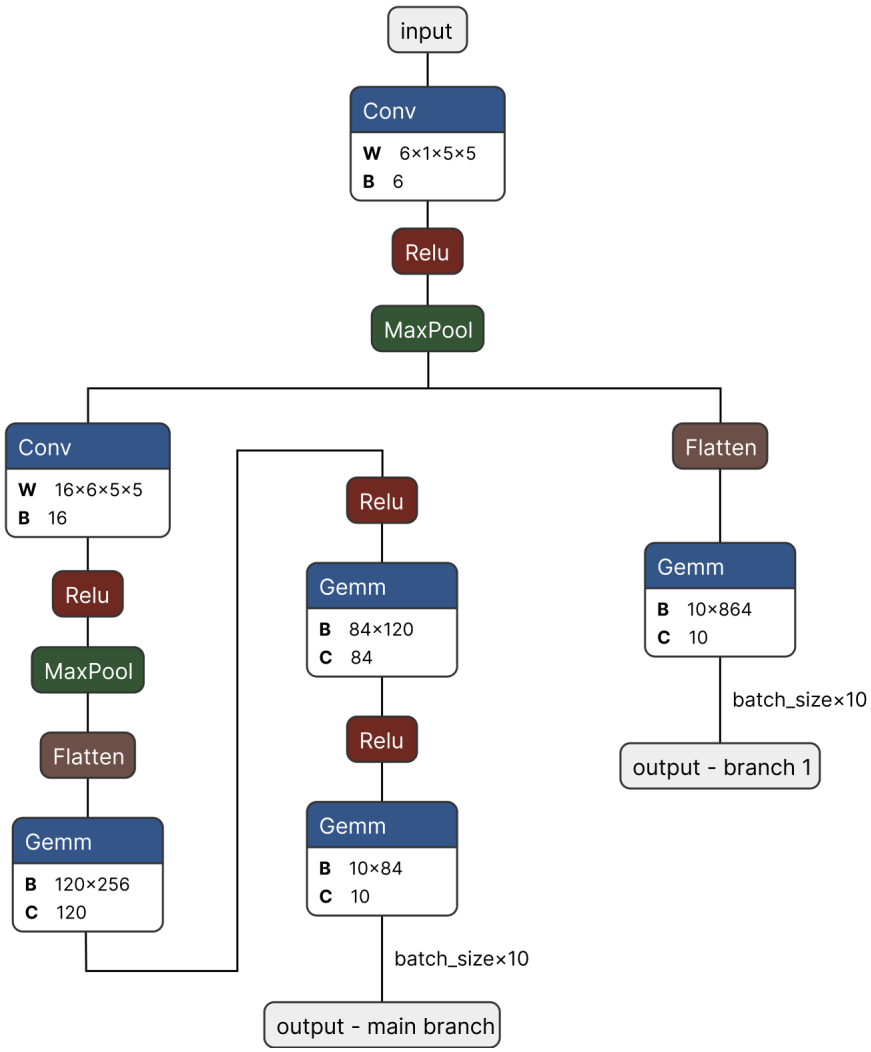


Figure 3.2: Branched LeNet architecture

weights assigned to each exit and L_1 and L_2 represents the loss functions for each exit. The training algorithm involves a forward pass in which the training data is passed through the network, output from all the exits are recorded and the loss is calculated as,

$$Loss_{branched} = w_1 * L_1 + w_2 * L_2 \tag{3.2}$$

This loss is back propagated through the network and weights are updated using gradient descent. During inference, if the classifier at the early exit is confident in it's predictions about correctly labeling a test sample, it is exited and returns the predicted output. Entropy is used as a measure of this confidence. Entropy for a prediction is given by,

$$Entropy(y) = -\sum_{c \in C} y_c log y_c \tag{3.3}$$

Where y is a vector containing computed probabilities for all possible class labels and C is a set of all possible labels. We need to set a threshold value for entropy at the early exit such that the sample makes an early exit if the entropy of the prediction is less than the threshold entropy. If not, it is processed further and exits through the main exit.

3.3 Semantic Segmentation

The choice of a suitable backbone semantic segmentation architecture for converting it to a multi-exit model is critical. We chose HRNet architecture (High Resolution Network) is chosen as the base model for the multi-exit experiments, as it preserves the high resolution representation of the input image throughout the network, addressing the main challenge of recovering spatial information in semantic segmentation tasks. Furthermore, a quick comparision of state-of-the-art accuracies of some popular semantic segmentation models shows HRNet has superior performance.

Model	**Accuracy(in %)**
HRNet V2-W48	81.6
DeepLab v3 (ResNet-101)	81
BiSeNet (ResNet-101)	78.9

Table 3.1: Comparison between semantic segmentation models

HRNet architecture is also better in terms of the number of parameters and computation complexity which is one of the key reasons for it to be used on resource and time constrained platforms and applications. HRNet architecture is being used as the backbone for a variety of vision applications such as object detection, semantic segmentation, image classification, and human pose estimation. Thus, multi-exit version of the same can further be utilized for all these tasks. We chose a lower scale of HRNet - W18, for easier training and repeatability of the experiments.
High Resolution Networks maintains the high resolution representation by connecting high to low resolution convolutions in parallel along with multi-scale fusions

[28]. The architecture is depicted in figure 3.3. There are four stages with the first stage having only high resolution convolutions. The second, third and fourth stages are formed by repeating modularized two resolution, three resolution and four resolution blocks. Each stage can be referred as a multi-resolution block which consists of multi-resolution group convolution and fusions. The multi-resolution group convolution divides the input channels into several subsets and performs a regular convolution over each subset over different spatial resolutions separately. Multi-scale fusions are performed in two ways to address resolution decrease and increase. The resolution decrease is implemented 2-strided 3×3 convolutions. The resolution increase is simply implemented by bilinear upsampling [28].

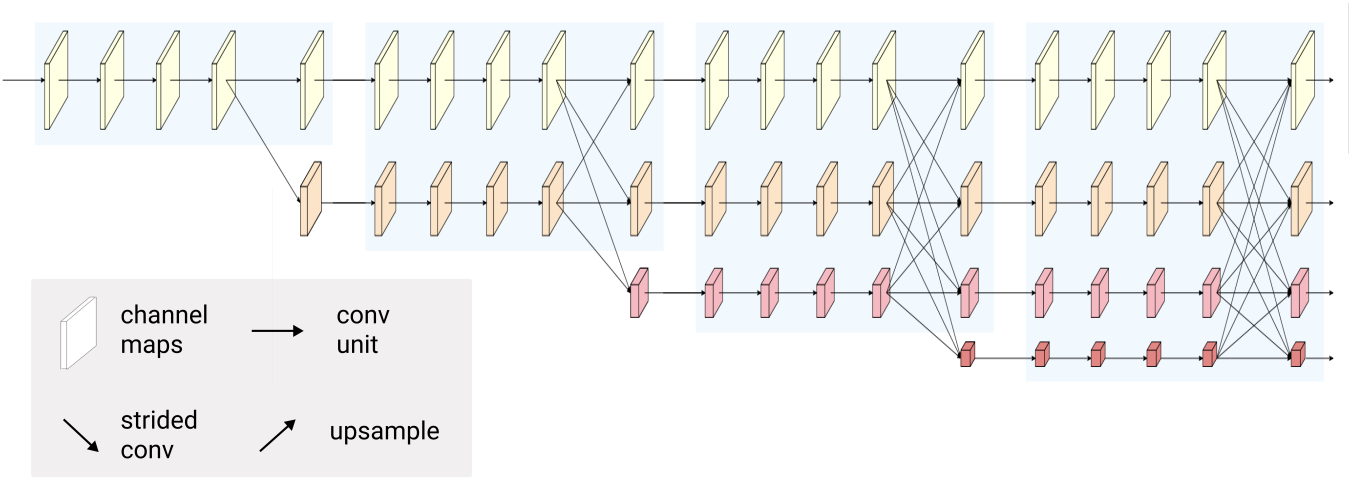


Figure 3.3: HRNet architecture [28]

In the initial HRNet version V1, only the representation from the high resolution convolution at the fourth stage is considered as the output and the low resolution convolution feature maps are lost. An effective modification has been done to this by fully exploring the capacity of multi resolution convolutions. The low-resolution representations are rescaled through bilinear upsampling to the high resolution, and concatenated resulting in the high-resolution representation, as illustrated in figure 3.4 which is used for estimating segmentation maps. The modified version is called HRNetV2 and it only adds small parameter and computation overhead to HRNetV1.

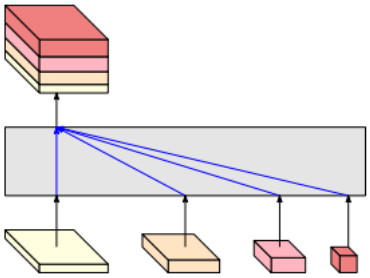


Figure 3.4: HRNetV2 representation [28]

3.4 Multi-exit Semantic Segmentation

We modified the regular deep learning models to multi-exit architecture with following steps:

1. Choosing the optimal exit points.
2. Design of segmentation heads and the additional trainable blocks.
3. Selection and implementation of exit methodology.

3.4.1 Choosing the exit points

The selection of the exit points is a crucial step in creating a multi-exit architecture. While an increased number of exit points allows a finer criteria selection, it also leads to increased model size and training time, which is undesirable when the model has to be deployed on a system with memory constraints. Also, it would have a larger footprint when accelerated processing units like GPU, TPU, etc. are used, which would require loading the entire model into those devices before making the predictions. With these constraints, a fair number of the exits can be formulated by dividing the model into groups of processing blocks based on encompassing sets of the layer with a similar intention of prediction. In terms of HRNet, these can be chosen to match the number of multi-resolution blocks.

Once the number of the exits is chosen they have to be placed at optimal points in the computation graph. The groups of layers used to determine the exit count can be used as a draft for these placements. Placing the exits equidistant along the network is also a reasonable approach but inferring these positions based on the network architecture and the data set complexity will yield optimal results. In the case of HRNet, once again multi-resolution blocks are a very good draft for these placements.

3.4.2 Design of segmentation heads and the additional trainable blocks

Because the exit points are located across the network, the default segmentation heads may be appropriate to reuse because the dimension of the feature maps obtained from these exit points and the executed input to the segmentation heads may differ. However, in the case of HRNet, there is a feature map with the same resolution as the input at the end of each multi-resolution block, so the processing required can be kept to a minimum by employing simpler blocks in the segmenting head. However, to compensate for the decreased receptive Fields, a few additional trainable blocks would be required. These can be a few convolution blocks placed between the exit point and segmentation head as represented in figure 3.5. Furthermore, because the segmentation head may come across distinct features that are distributed differently, retraining these specific to these feature maps could be beneficial.

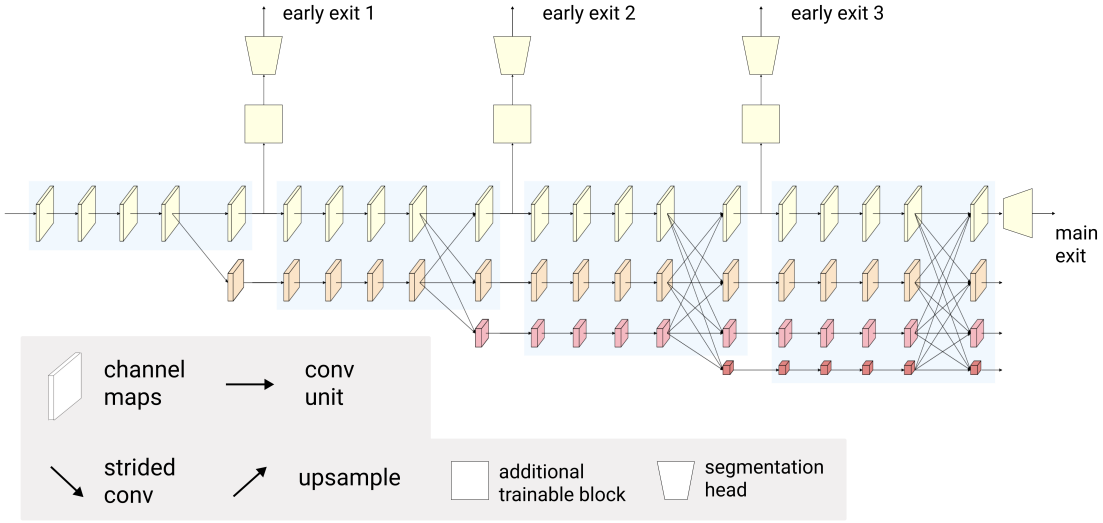


Figure 3.5: Multi-exit HRNet Architecture

3.4.3 Selection and implementation of exit methodology

Selection of the strategy that will decide if the network prediction is good enough to halt further processing of the input data. This would be directly responsible for handling the latency vs accuracy trade-off in the system. Strategies are mainly based on the confidence metrics. At the end of each block of processing that is separated by exit points, the network analyzes the output and rates the confidence in its prediction. When the confidence level is higher than the pre-defined thresholds, the network halts the further processing and returns the prediction. We have studied different exit strategies to identify the best one in terms of performance which are discussed in detail in the next section.

In the context of multi-exit networks, determining whether to opt for an early exit or process the image through the whole network remains a challenging task. Without the right exit strategy in place, there is a risk that easy examples will be processed by the whole network (which is wasteful) or that complex examples will make an early exit (which may harm the quality of the output). To make the correct decision, the multi-exit network must be able to determine what performance/accuracy gain can be expected if the image is processed further, even at the time when the image has only been processed by parts of the network. This thesis studies different approaches to design an exit strategy that efficiently handles the potential trade-off between inference speed and accuracy for multi-exit networks.

3.4.4 Exit strategy based on Entropy

A common way of determining the likelihood of the prediction without knowing the ground truth is to use entropy, which measures the amount of uncertainty in the prediction. If the entropy is low, it means that the prediction is very certain. If the entropy is high, it means that the prediction is very uncertain. while if the

entropy is low, the network should interrupt the execution and make an early exit. For semantic segmentation, the entropy is calculated using the predicted output at each exit of the model. The per-pixel entropy is given by formula 3.3 and the total entropy for the segmented output is calculated by taking an average of the per pixel entropies.
During the network training, values for entropy and accuracy at different branches are recorded, which is utilized to identify the threshold entropy involved in the decision-making. The task of choosing these threshold entropies can be thought of as an optimization problem in which we aim to maximize the entropy threshold value, which corresponds to the maximum early exits, while keeping the bare minimum accuracy criteria. The exit strategies based on the entropy of the predictions at each exit branch build on the assumption that the entropy has a negative correlation with the accuracy of the predictions, which is mostly true in the case of a neural network since the loss functions used in the training makes similar assumptions. Figure 3.6 shows the model pipeline where an exit condition is placed at the end of each stage and the input sample takes the earliest exit if the entropy of its prediction is below the set threshold entropy for that exit.

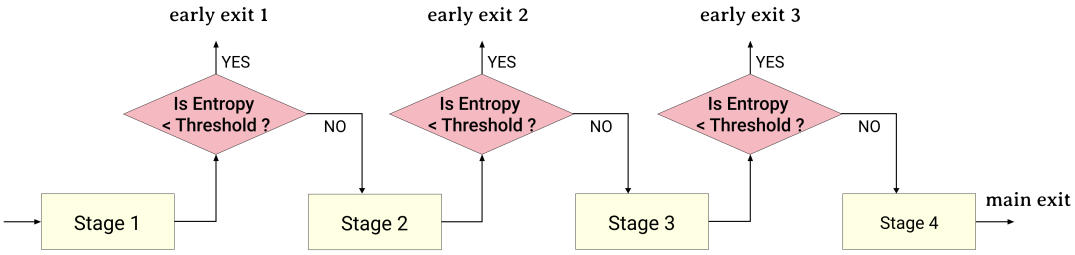


Figure 3.6: Entropy based early exit

However, this strategy makes no effort in estimating the expected performance gain of processing the image further. Therefore, the network would avoid making an early exit if the entropy of the prediction of an image is large at each exit branch, but the prediction performance might not be any better than that of the first exit branch. In general, one would only want to process an image further if doing so would result in a performance benefit, which is not expressly considered in this method.

3.5 Proposed novel Exit Strategies

3.5.1 Exit strategy based on Fast Fourier Transform

We developed an innovative approach to making the exit decision in the multi-exit semantic segmentation model by analyzing the prediction at each exit point in the frequency domain. Fast Fourier transform is used to transform the prediction from the spatial domain to the frequency domain. An image in the frequency domain helps to identify certain features that are not easily identifiable in the spatial domain. The exit criteria are determined by the range of expected mean magnitude of the prediction in the frequency domain.

Intuitively less accurate predictions from a semantic segmentation will have either higher jitter in the output or outputs predicting only one of the classes for the entire image. The mean magnitude of the prediction in the frequency domain acts as an interesting proxy that represents this information. During training, we can set a range of the mean FFT magnitude where we expect to achieve the desired accuracy and use this as a criterion for making an early exit in a neural network 3.6.

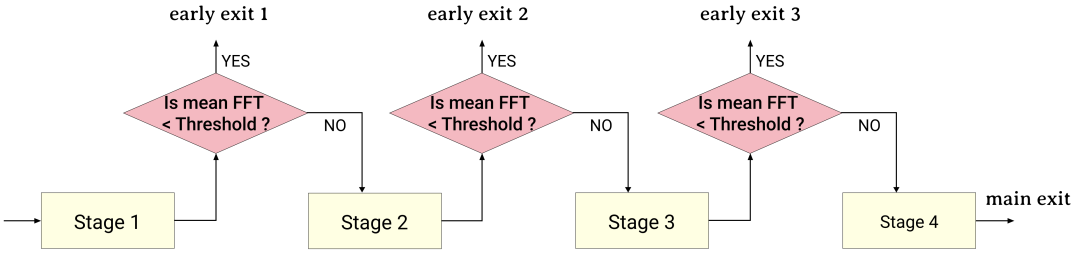


Figure 3.7: FFT based early exit

During inference, if the mean magnitude of the prediction in the frequency domain is not in the expected range, it indicates the presence of jitter/incorrectly classified pixels in the output which is prominent in the earlier exits due to lower receptive fields. We propose the exit thresholds to be determined by analyzing the magnitude of the Fast Fourier transform on the prediction from training data and by objective calculation by setting a minimum size of the segment to be determined in the image coordinate.

3.5.2 Exit strategy with an Auxiliary Network

A more sophisticated way of deciding the exit criteria is to use an auxiliary deep neural network for decision making, where an additional network is used to represent the early exit preferences. The auxiliary network is trained and supervised to learn an optimal exit strategy comparing the predictions from the multiple exits of the semantic segmentation model and the ground truth. It is connected to all the early exits in the main model. The trained auxiliary model 3.8 processes the prediction from each exit and tries to makes the optimal exit decision. This ensures that the image will be further processed by the network only when a performance gain is guaranteed. The prediction from each stage is fed to an auxiliary network which decides if the output has the expected accuracy or if it is required to be processed by the subsequent stages.

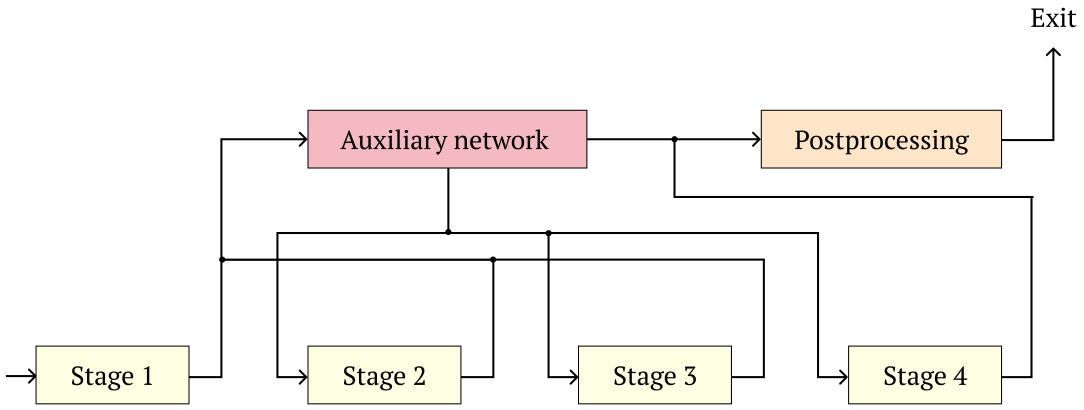


Figure 3.8: Multi-exit model with auxiliary network

The auxiliary network can be modeled in two different ways: As a classification model or as a regression model.

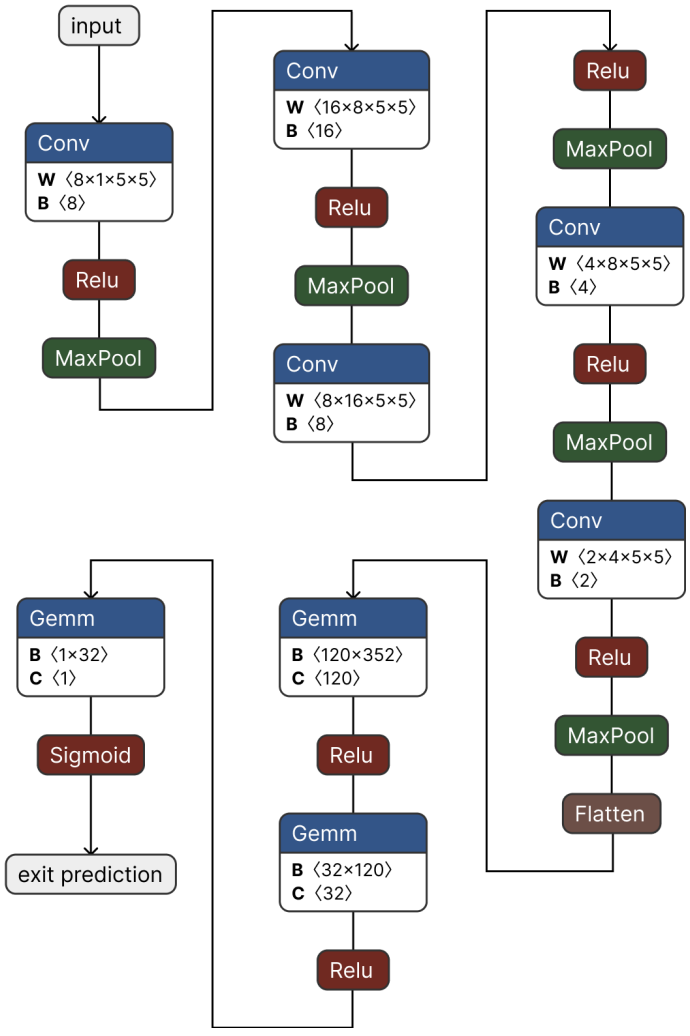


Figure 3.9: Auxiliary classification model

- As a **classification model**, the model uses binary classification to decide if the prediction is good enough to take the current exit or multi-class classification problem to decide the expected exit index. The auxiliary network 3.9 tackles the decision-making as a binary classification problem predicting True/ False in correspondence with the if the image should take the early exit or not.
  The target variables during the training are formulated based on setting a range for minimum expected gain that qualifies for not taking an early exit. This range is compared with each training sample to mark binary target values. And during the inference, these binary predictions are directly considered to take or to skip an early exit.
- As a **regression model** to predict the expected value of mIoU for the output segments. The auxiliary network 3.10 is chosen to predict the one variable that corresponds to the accuracy of the prediction at the output with a suitable activation function.

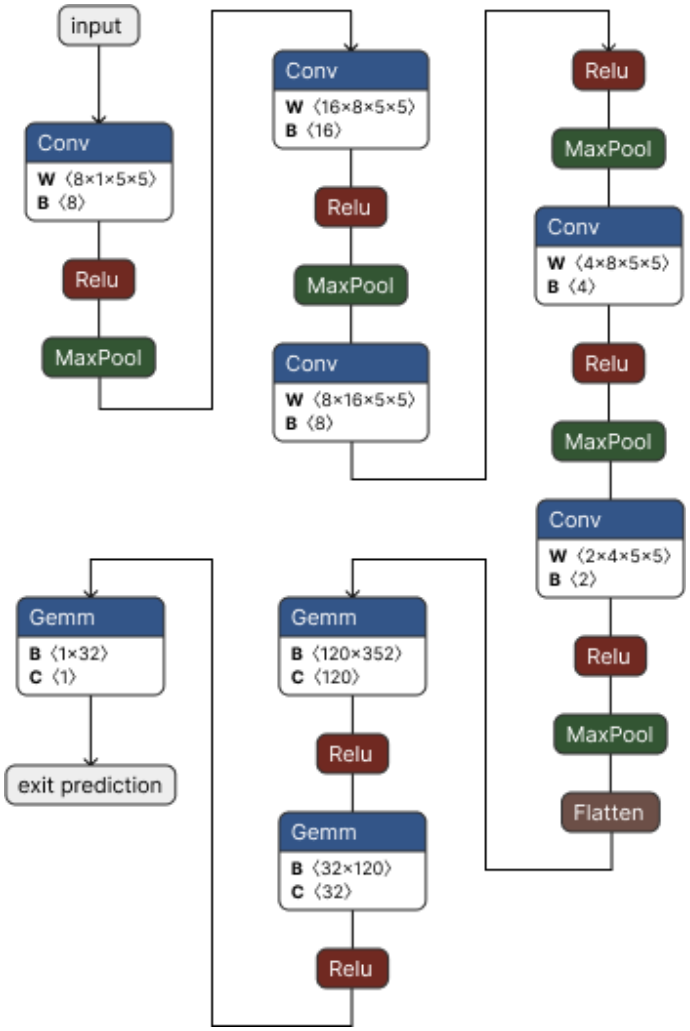


Figure 3.10: Auxiliary regression model

At the time of training the ground truth from inputs are used to calculate actual mIoU, which is used as the target variable for the auxiliary network.

After this a threshold for this value can be set based on the requirements of the system, which will be used in the early exit strategy in the course of inference.

3.5.3 Exit strategy through Reinforcement Learning

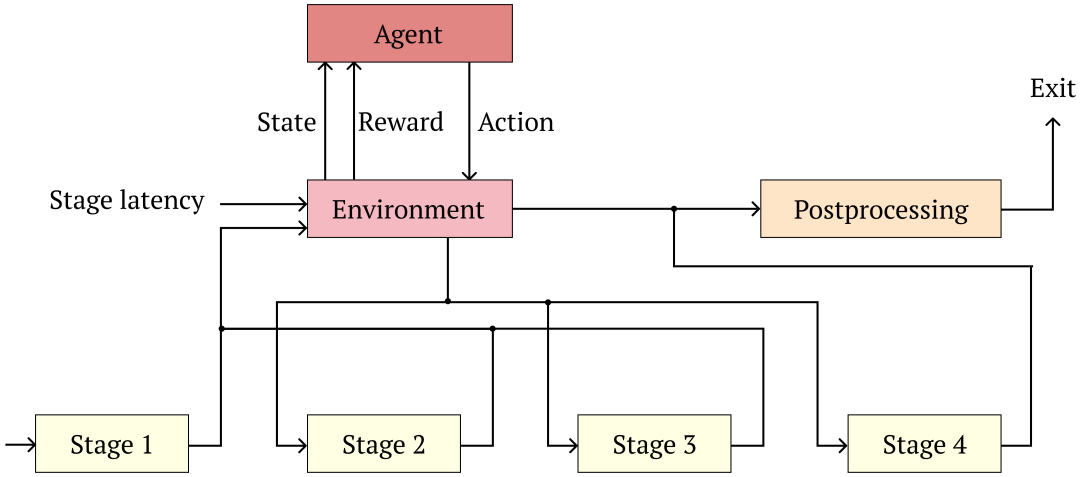


Figure 3.11: Multi-exit reinforcement learning model pipeline

One of the primary challenges of training an auxiliary decision network is to define objective goals that maximize the chance of optimal exit selection. The features that are input might have different distributions at each exit point, which might lead to inaccurate exits. In addition, the expected latency penalty of the subsequent processing layers also plays a key role in choosing the current exit or skipping it. To encompass expected latency as one of the deciding factors in the exit strategy, the thesis introduces a reinforcement learning approach. A transient environment is created based on prediction and prediction likelihood, which is used to train an agent that learns to choose the exits. The thesis suggests a technique for creating states based on predefined sets of bins predicated on the input metric distribution as shown in the figure 3.12, to reduce the number of state action pairs that agents must comprehend.

The reward given to the agent is determined by both accuracy and the latency of the processing block. Through this technique, the agent would be encouraged to learn the policy that optimizes the latency of the system while maintaining the required accuracy.

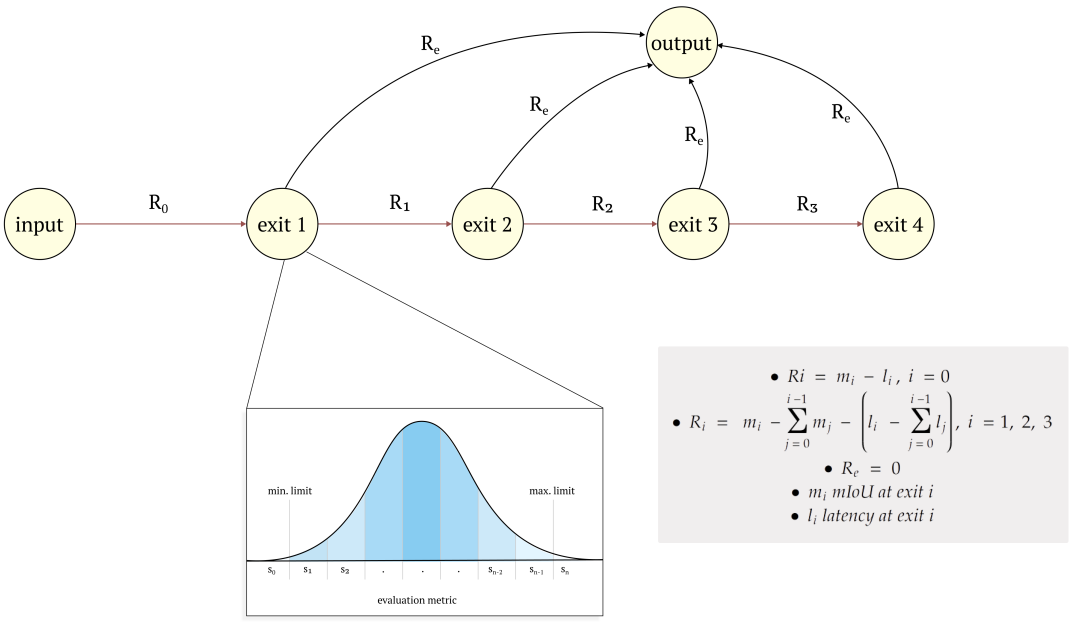


Figure 3.12: states in RL based Multi-exit network

The flexibility in choosing the inputs for the reward function would also allow encoding certain domain knowledge in the reward, this would allow the exit strategy to be tuned based on the handcrafted features. An example of this knowledge could be the use of the previous prediction in deciding the confidence of the current prediction, which would have a significant impact on dealing with the prediction on video feed with sequential frames.

3.6 Effects of Dataset Variation

For the initial validation of various exit tactics, we have run all the experiments using an open-source CityScapes dataset. To demonstrate the generality of exit options, these findings are confirmed using a different dataset. The second dataset is based on the binary classification job that will be employed by Volvo's hybrid work environment's primary vision systems.
The experiment setups are kept the same with those used for CityScapes evaluation, with a few minor adjustments made to better replicate deployment conditions for edge cloud operations.

4

Results

The primary findings of the experiments conducted in the thesis are explained in this chapter. The sub sections describe the results of multi-exit classification model and the multi-exit semantic segmentation model.

4.1 Multi-exit Classification Model

Two deep learning classification models LeNet and branched LeNet were trained and evaluated using the popular MNIST dataset and this section describes the key observations. To train the branched LeNet, the loss is calculated as a weighted sum of losses from early exit and main exit. The weights assigned are $w_1 = 0.75$ for the early exit and $w_2 = 0.5$ for the main exit. Higher weight is assigned to early exit for more feature learning in the early stage of the network which allows more samples to exit early with high confidence. The threshold entropy set for performing inference is chosen dynamically by noting down the entropy of predictions during training. Figure 4.1 shows the inverse relationship between entropy and accuracy and the threshold entropy for attaining an accuracy of 99.4 % can be easily observed from this graph.

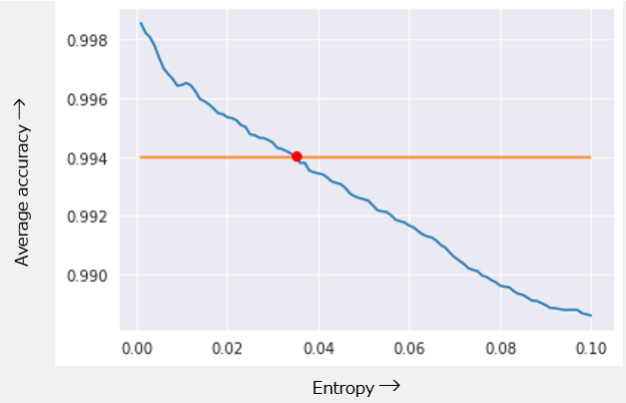
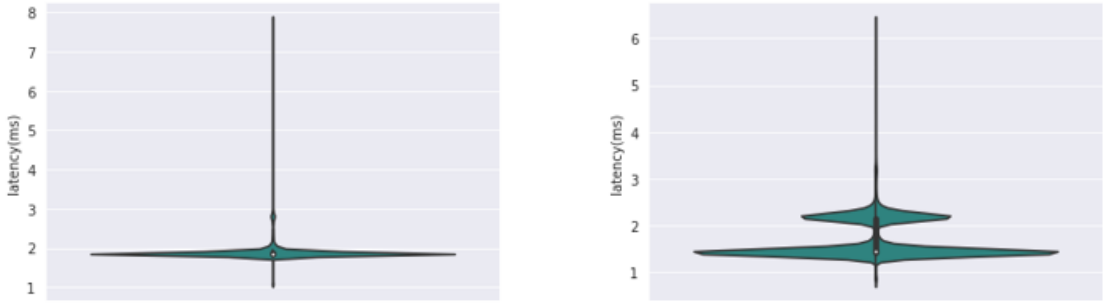


Figure 4.1: Accuracy vs Entropy

Figure 4.2 shows the plots of inference time for LeNet and branched LeNet. For branched LeNet, it shows the distribution of samples taking early exit and main exit and the average inference time is reduced for the branched version with more samples taking the early exit. Table 4.1 compares both models in terms of accuracy, average flops and latency. It has been observed that the branched model has considerably

reduced number of average floating point operations and a reduced latency with a very slight drop in accuracy.



(a) LeNet **(b)** Branched LeNet

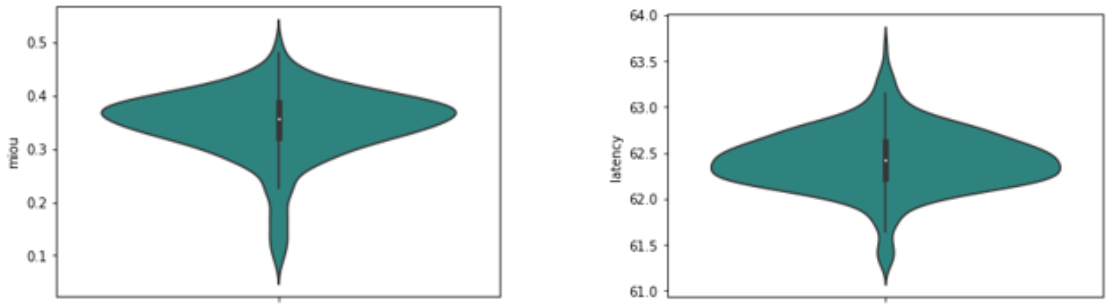
Figure 4.2: Results of classification models

Model	Accuracy(in %)	Average FLOPs	Latency (in ms)
LeNet	97.57	281640	1.944
Branched LeNet	97.56	101287	1.692

Table 4.1: Comparison between branched and un-branched classification models

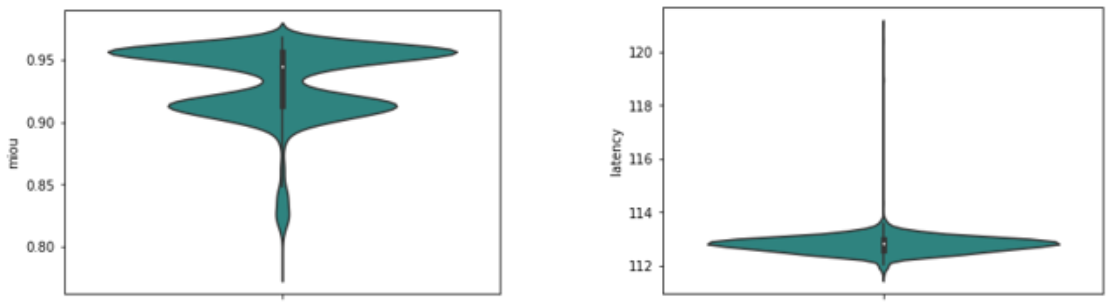
4.2 Multi-exit Semantic Segmentation

All the semantic segmentation experiments were performed on the base architecture of HRNet-18 which was chosen among the HRNet family for the ease of training and carrying out multiple experiments. The HRNet-18 backbone was trained using the open source CityScpaes dataset and the Volvo dataset. Figures 4.3 and 4.4 represents the mIoU and latency distributions of HRNet-18 for CityScapes dataset and Volvo dataset respectively. The average mIoU and average latency observed for the CityScapes dataset were 34.8 % and 62.42 ms. Similarly, the average mIoU and average latency observed for the Volvo dataset were 93.19 % and 114.79 ms. These values are used as references for the further experiments conducted for the task of semantic segmentation.



(a) mIoU **(b)** latency

Figure 4.3: Results of HRNet-18 for CityScapes dataset

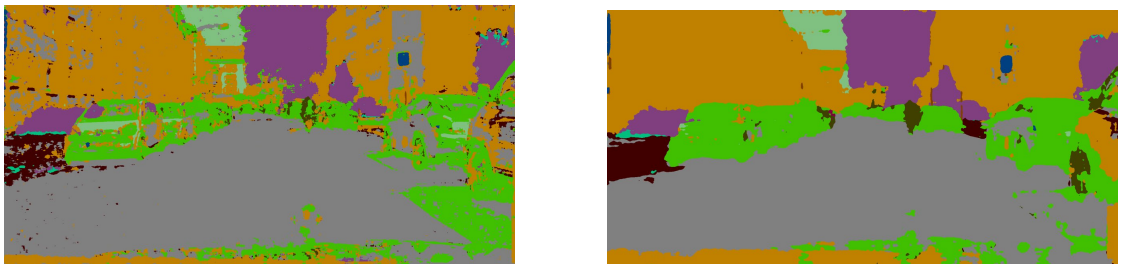


(a) mIoU **(b)** latency

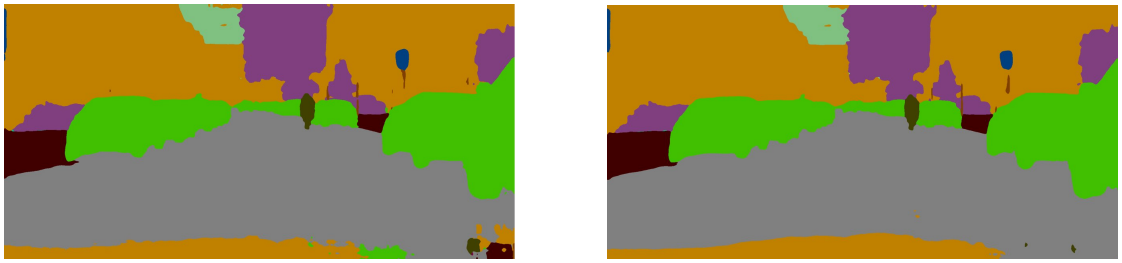
Figure 4.4: Results of HRNet-18 for Volvo dataset

We developed a multi-exit semantic segmentation model using HRNet-18 as backbone by adding 3 exits at the end of each stages. Additional blocks and segmentation head was designed for each exit to imrpove the quality of prediction. We trained the model on CityScapes dataset to predict dense classes on an image. The loss function is calculated as a weighted sum of loss functions from the three early exits and the main exit. The weights assigned are $w_1 = 0.4$ for the early exit 1, $w_2 = 0.3$ for the early exit 2, $w_3 = 0.2$ for the early exit 3 and $w_m = 0.1$ for the main exit, which were chosen to provide importance to the earlier exits with inverse correlation to its position in the network. Figure 4.5 shows the input image, ground truth mask and the predictions from multiple exits. Exits one through four show varying degrees of prediction quality, which shows with each added layer there is an increase in the quality of the output. The initial exits have jitter in the prediction, which could be due to the lower receptive fields in these branches. However, certain areas of the image have clear and accurate predictions from the first exit. The same multi-exit model was trained using the Volvo dataset and the figure 4.6 shows the results. The task is binary segmentation and the characteristics of the predictions from each exit is similar to that of CityScapes dataset.

(a) input image **(b)** ground truth

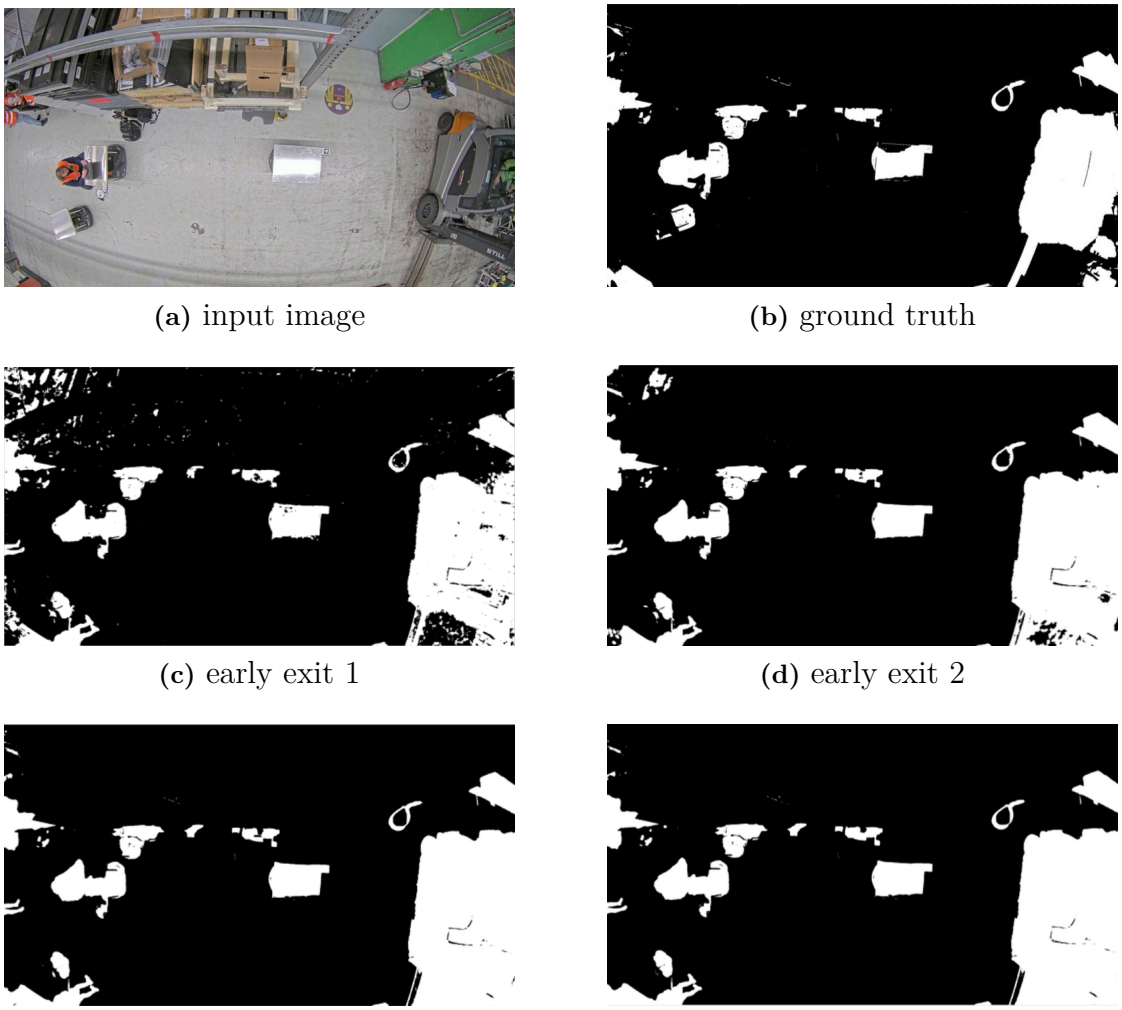


(c) early exit 1 **(d)** early exit 2



(e) early exit 3 **(f)** early exit 4

Figure 4.5: Branched HRNet Inference on Cityscapes



(a) input image

(b) ground truth

(c) early exit 1

(d) early exit 2

(e) early exit 3

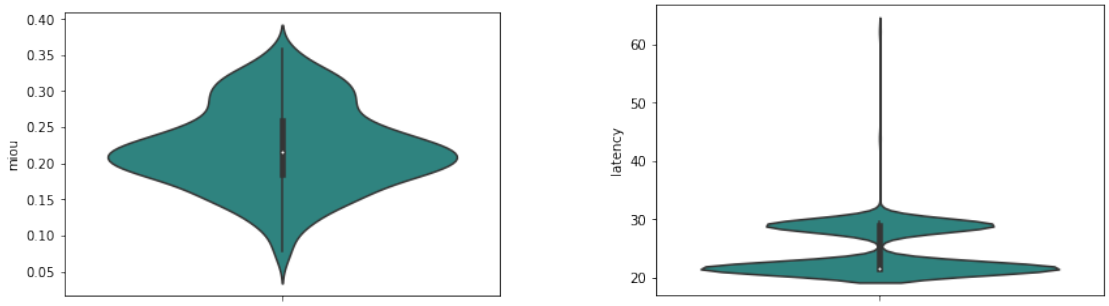
(f) early exit 4

Figure 4.6: Branched HRNet Inference on Volvo Dataset

4.2.1 Entropy based exit strategy

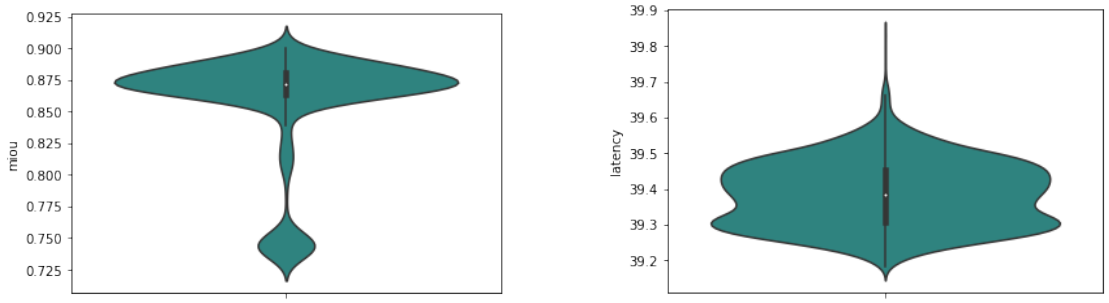
The initial experiment conducted was to study the performance of the multi-exit model by considering entropy of the predictions at each exit point to make decisions to exit the network. We chose entropy thresholds for each of the four exits by setting the target accuracy to be 30 % for CityScapes dataset and 90 % for Volvo dataset. The resultant plots have been similar for both CityScapes and Volvo dataset shown by figures 4.7 and 4.8 respectively. The average mIoU dropped to 22.12 % for CityScapes and 85.82 % for Volvo datasets which are less than the traget mIoUs set for both cases. The average latency was observed to be 24.63 ms and 39.38 ms for CityScapes and Volvo datasets, respectively. The average latency has been reduced significantly for both datasets as a large number of samples took earlier exits. The subsequent section studies the entropy in detail for each of the four exits and the unexpected deviation from the target accuracy set for both datasets. The negative correlation between entropy and accuracy did not work as expected which is due to semantic segmentation being a pixel-wise classification task that can have

extremities incorrect and wrong predictions within the image. This might have lead to the unexpected accuracy drop from the target value in case of both datasets.



(a) mIoU **(b)** latency

Figure 4.7: Results of entropy based exit strategy for Cityscape dataset



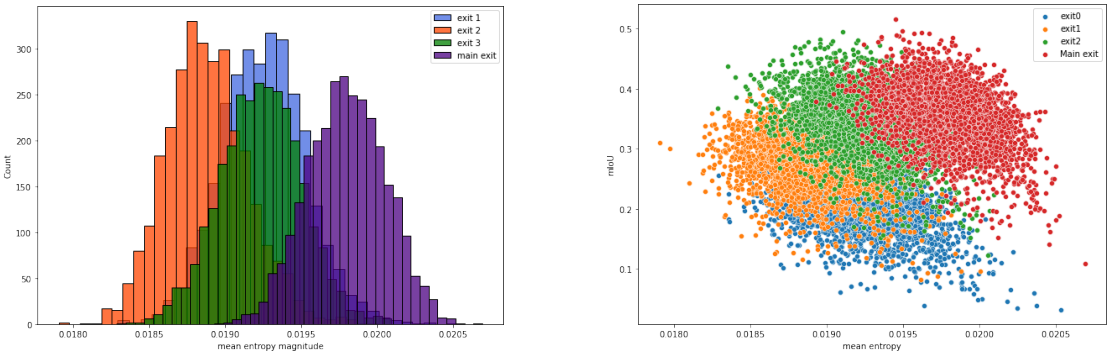
(a) mIoU **(b)** latency

Figure 4.8: Results of entropy based exit strategy for Volvo dataset

4.2.2 Analysis of exits with entropy

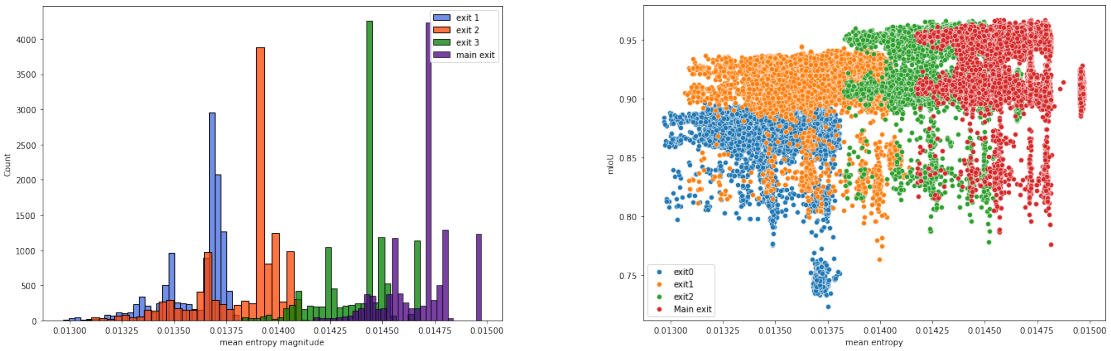
When the accuracy of the samples was analyzed with respect to the entropy at the exit points, the distributions showed an overlapping pattern. There is not a clear threshold that would distinguish the entropy patterns in the histograms 4.9(a) and 4.10(a), the distribution of samples taking the earlier exits is sometimes completely shadowed by the entropy distribution of subsequent exits.

The scatter plots 4.9(b) and 4.10(b) between the accuracy and mIoU does not show much of clear distinguishing traits or linear trends like the one that was observed in entropy-based early exits in the classification tasks. The observations were consistent among the two different datasets used with some contrast due to their individual scale.



(a) (b)

Figure 4.9: Analysis of exits with entropy for Cityscape dataset

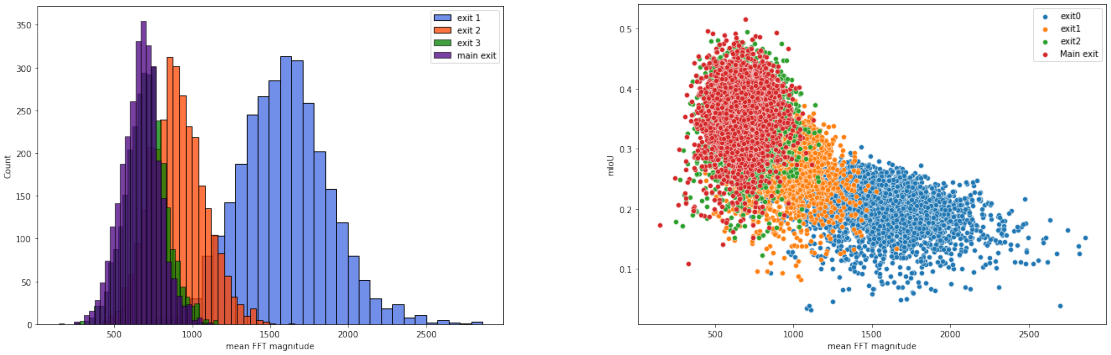


(a) (b)

Figure 4.10: Analysis of exits with entropy for Volvo dataset

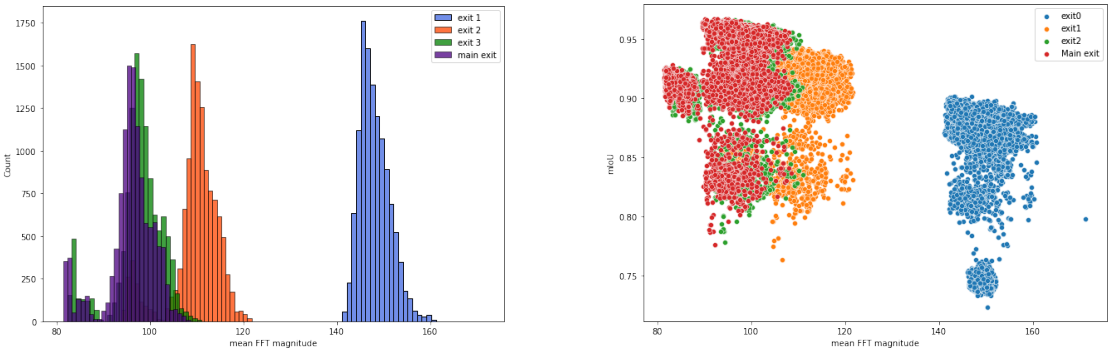
4.2.3 Analysis of exits with FFT

The values (mean FFT magnitude) of FFT-based early exits were studied with respect to the accuracy of each sample with a similar setting as the entropy-based exit strategy. This showed a similar trend to that seen in the branched classification results. The histograms 4.11(a) and 4.12(a) showed a distribution order that depicts the arrangement of the exits, starting from the last exit to the earliest one. There is a significant overlap seen between exit 3 and the main exit (exit 4), which could be attributed to the similarity in the results. The plots 4.11(b) and 4.12(b) between the accuracy and the FFT values shows an inverse relationship, the increased accuracy can be seen with a decrease in the FFT values.



(a) (b)

Figure 4.11: Analysis of exits with FFT for Cityscape dataset

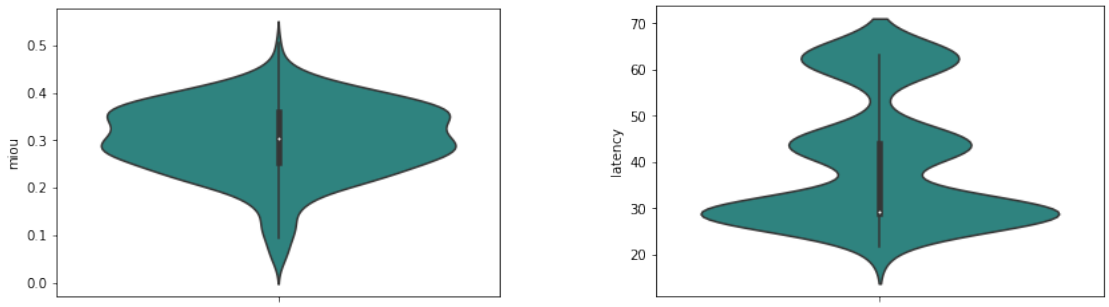


(a) (b)

Figure 4.12: Analysis of exits with FFT for Volvo dataset

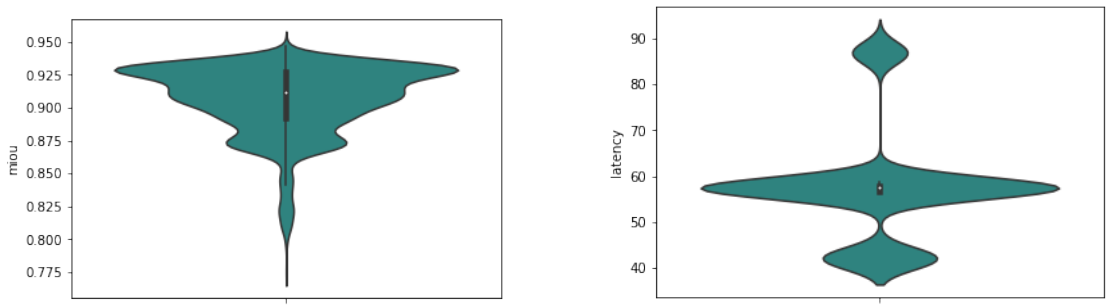
4.2.4 FFT based exit strategy

As the measure of frequency in the predictions shows an inverse relationship with the mIoU, the magnitude of FFT of the predictions at each exit point has been considered to make decisions to exit the network. we chose the thresholds for each of the four exits by setting the target mIoU (accuracy) to be 30 % for CityScapes dataset and 90 % for Volvo dataset. The resultant plots have been similar for both CityScapes and Volvo dataset shown by figures 4.13 and 4.14 respectively. The results were as expected with 30.2 % and 90.65 % of average mIoU and 39.97 ms and 57.40 ms of average latency for CityScapes and Volvo datasets, respectively. The average latency has been reduced significantly compared to the baseline values shown in figures 4.3 and 4.4.



(a) mIoU **(b)** latency

Figure 4.13: Results of FFT based exit strategy for Cityscape dataset



(a) mIoU **(b)** latency

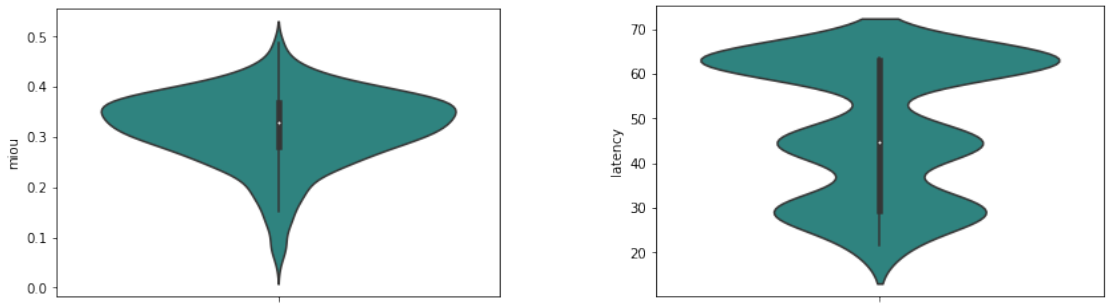
Figure 4.14: Results of FFT based exit strategy for Volvo dataset

4.2.5 Auxiliary model based exit strategy

Two auxiliary networks were trained using the ground truth and the prediction from each exit so that these can take more efficient exit decisions compared to the metric based exit strategies. The results of these approaches are explained in the following sub-sections.

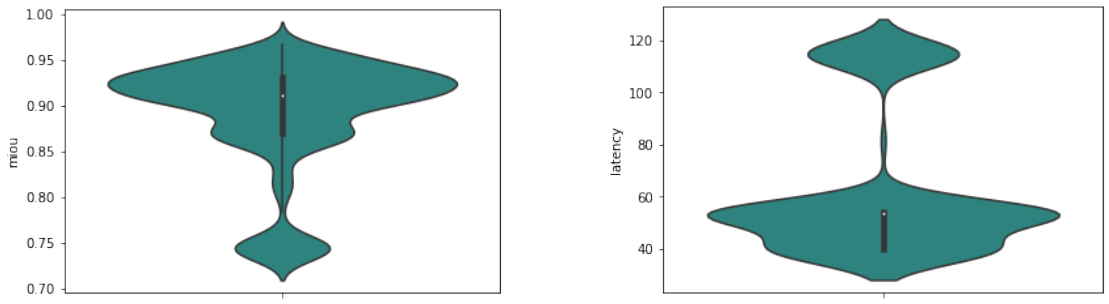
4.2.5.1 Classification model

A classification model is trained using the prediction from each exit and a binary target value. The binary target value for a sample is decided to be False if the expected accuracy gain from the later exits exceeds 0.1. Otherwise, it is set to True. This ensures that further processing decision is only taken if there is an expected performance gain. Figures 4.15 and 4.16 shows the results as expected with an average mIoU of 31.9% and 89.25% and an average latency of 47.86 ms and 62.64 ms for CityScapes and Volvo datasets, respectively.



(a) mIoU **(b)** latency

Figure 4.15: Results of auxiliary classification model based exit strategy for Cityscape dataset

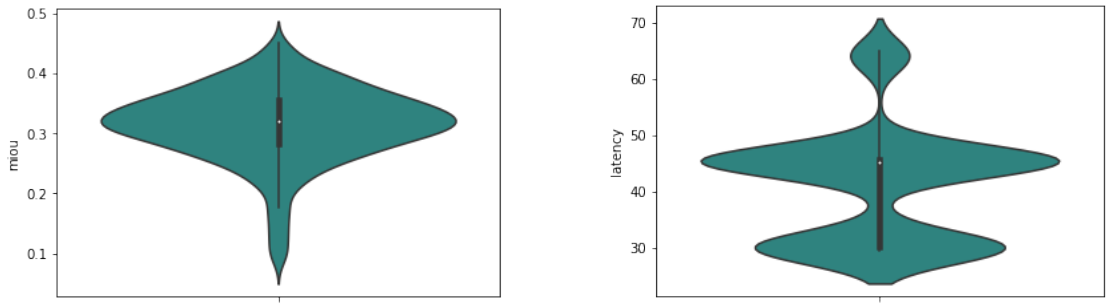


(a) mIoU **(b)** latency

Figure 4.16: Results of auxiliary classification model based exit strategy for Volvo dataset

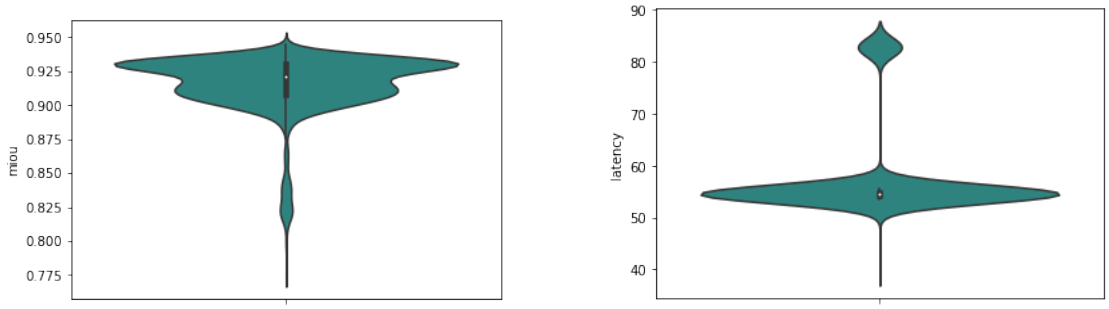
4.2.5.2 Regression model

A regression model is trained using the predictions from each exit and the mIoU calculated using the ground truth. During inference, a sample may make an exit if the predicted mIoU by the regression model is greater than the threshold value set. The threshold accuracy set for CityScapes and Volvo datasets are 30% and 90%, respectively. The average mIoU achieved are 31.48 % and 91.58% and the average latency is observed to be 41.36 ms and 57.64 ms for CityScapes and Volvo datasets, as shown in figures 4.17 and 4.18.



(a) mIoU **(b)** latency

Figure 4.17: results of auxiliary regression model based exit strategy for Cityscape dataset



(a) mIoU **(b)** latency

Figure 4.18: Results of auxiliary regression model based exit strategy for Volvo dataset

4.2.6 Reinforcement learning based exit strategy

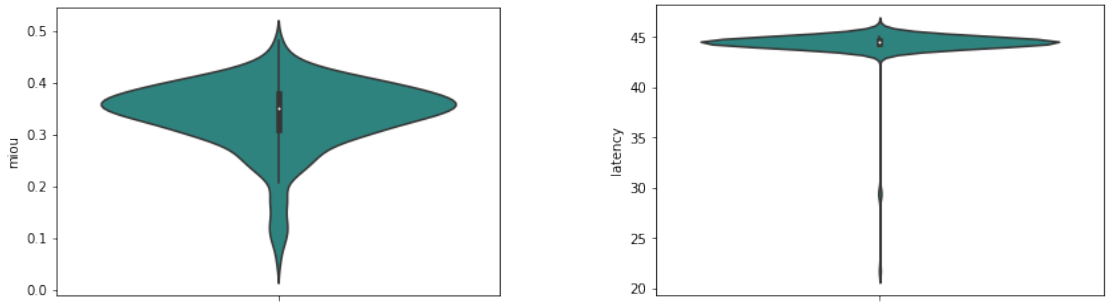
We trained a model-free reinforcement agent on the tabular states with a fixed number of bins. The agent used an epsilon-greedy policy while taking the action. The epsilon was set to zero using the training, which forced the agent to prioritize the exploration. This prioritization helps in acquiring maximum knowledge of the environment that the agent is dealing with. After the completion of training episodes, the agent's learning is frozen, and the agent is instructed only to exploit the learning that it had made previously. This was done by setting the epsilon parameter of the agent to one.

The study was conducted in environments with states formed by two different output exit metrics,

1. Entropy states (mean Entropy)
2. FFT states (mean FFT magnitude)

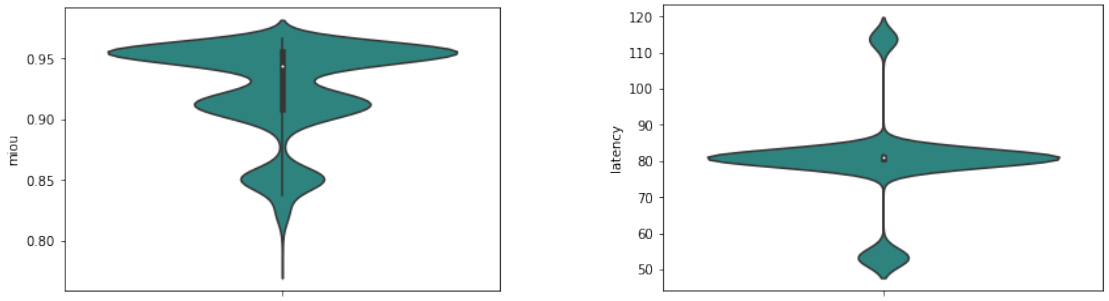
4.2.6.1 Entropy states

The entropy-based states used output entropy from exits as an input to the preprocessing unit that created states in the environment. The individual pixel entropies are averaged and normalized using a standard scaler before categorizing them using binning techniques.
The technique yielded an accuracy of 33.8% and 92.46% with an average latency of 44.3153ms and 79.8214ms on CityScapes and Volvo datasets, respectively. The detailed distribution of the results can be seen in figures 4.19.



(a) mIoU **(b)** latency

Figure 4.19: Results of RL based exit strategy with entropy states for Cityscapes dataset

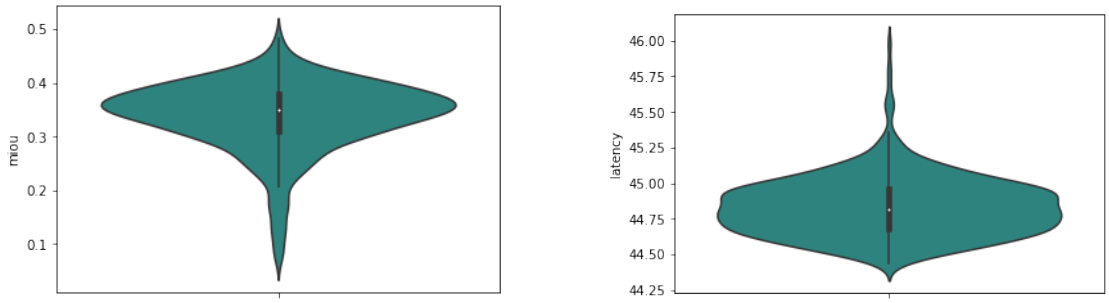


(a) mIoU **(b)** latency

Figure 4.20: Results of RL based exit strategy with entropy states for Volvo dataset

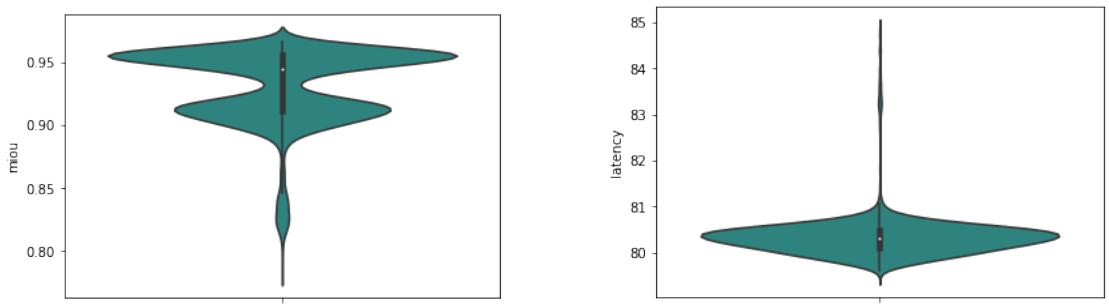
4.2.6.2 FFT states

The configurations of the environment created for FFT states are done like the entropy-based states. The FFT values from the segmentation prediction are considered as the input of the preprocessing block, which uses standard normalizer and similar state binning techniques like the entropy-based state creation.
The outcome of method can be seen in figure 4.21 . The technique had 33.9% and 93.08% of accuracy with an average latency of 44.8356ms and 80.3392ms on CityScapes and Volvo datasets, respectively.



(a) mIoU **(b)** latency

Figure 4.21: Results of RL based exit strategy with FFT states for Cityscapes dataset

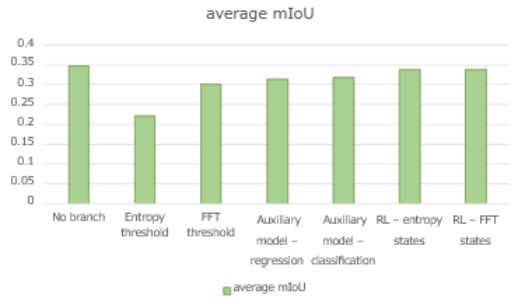


(a) mIoU **(b)** latency

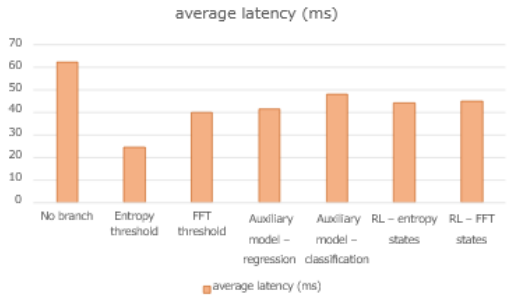
Figure 4.22: Results of RL based exit strategy with FFT states for Volvo dataset

4.2.7 Comparison of exit strategies

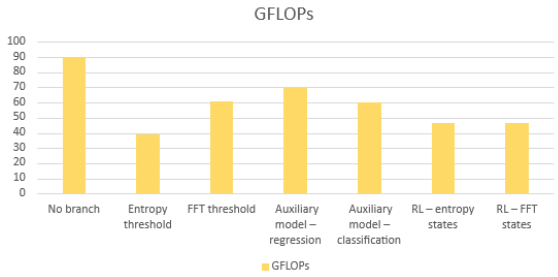
The compiled results from all the exit strategies are represented in the following tables and plots 4.23, 4.24 for the Cityscapes and Volvo datasets, respectively. The model without branching had the best performance in terms of mIoU but also had the worst performance in terms of latency and FLOPs. The branching architecture with RL-based exit strategies had accuracies (mIoU) close to the no branch models, with computations (GFLOPs) reduced to close to half of what is requested in no branch models. The performance gain in terms of latency is also significant in these methods. Following this, regression-based auxiliary model and FFT-based thresholding have commendable performances in terms of accuracy (mIoU) in the task of semantic segmentation. It is observed that the regression-based auxiliary model and FFT-based thresholding models are better in terms of latency but the required number of computations (GFLOPs) are much higher when compared to the RL-based exit startegies. This implies that the RL-based approaches are less complex and appears to be the most effective method.



(a) mIoU



(b) latency



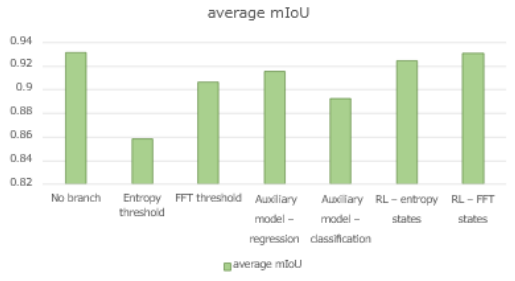
(c) GFLOPs

Figure 4.23: Results Comparison for CityScapes dataset

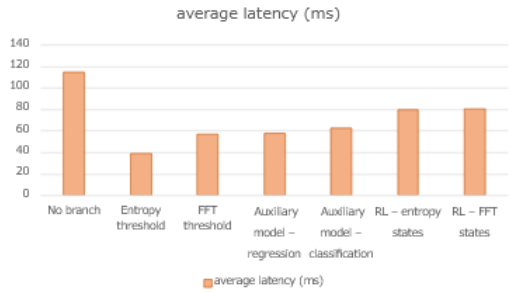
Exit Strategy	**mean mIoU**	**std. mIoU**	**mean Latency (ms)**	**std. Latency (ms)**	**GFLOP**
No branch	0.348	0.06505	62.4248	0.33385	90.2
Entropy threshold	0.2212	0.0579	24.6359	4.06192	39
FFT threshold	0.302	0.07741	39.9767	13.57318	60.5
Classification Aux. model	0.319	0.07253	47.8696	15.02366	69.8
Regression Aux. model	0.3148	0.06223	41.3675	10.13324	60
RL - entropy states	0.3381	0.06721	44.3153	1.97735	46.2
RL - FFT states	0.339	0.06518	44.8356	0.20765	46.2

Table 4.2: Performance comparison for CityScapes dataset

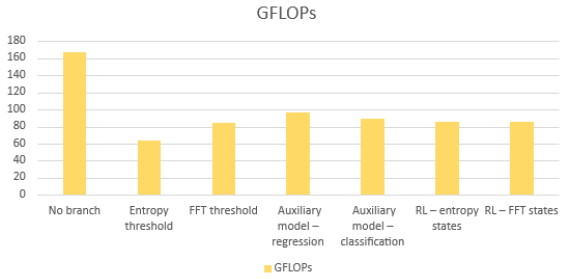
The relative performance of the exit strategies on a dataset (that is either Cityscapes or Volvo) was found to be comparable with the other dataset. However, the true values for accuracy were found to be higher in the case of the Volvo dataset because of lesser classes in the segmentation task and the latency was higher due to the resolution of the images used.



(a) mIoU



(b) latency



(c) GFLOPs

Figure 4.24: Results Comparison for Volvo dataset

Exit Strategy	**mean mIoU**	**std. mIoU**	**mean Latency (ms)**	**std. Latency (ms)**	**GFLOP**
No branch	0.9319	0.02988	114.7967	8.35199	167.3
Entropy threshold	0.8582	0.04294	39.3858	0.09231	63.4
FFT threshold	0.9065	0.02623	57.4016	15.6413	84.6
Classification Aux. model	0.8925	0.06086	62.6463	27.9698	96.8
Regression Aux. model	0.9158	0.02178	57.6474	8.83003	89.6
RL - entropy states	0.9246	0.03865	79.8214	12.56519	85.5
RL - FFT states	0.9308	0.03007	80.3392	0.45784	85.9

Table 4.3: Performance comparison for Volvo dataset

5
Conclusion

This chapter includes the key observations derived from the multi-exit approach applied on a classification and a semantic segmentation model. Finally, it presents a conclusion to the thesis work.

5.1 Discussion

This section provides an explanation and justification for the findings in the previous chapter's several experiments involving multi-exit neural networks. Beginning with the straightforward multi-exit classification model, it goes on to discuss the outcomes of the multi-exit semantic segmentation model that had a variety of exit strategies applied to it.

5.1.1 Multi-exit Classification Models

Experimenting with branching a deep learning model, adding multiple exits, and specifying appropriate exit criteria yields significant performance benefits in terms of average inference time. Considering the models evaluated, the multi-exit classification model shows a decreased inference time compared to the original unbranched model with little compromise on accuracy. The exit criteria based on entropy of the prediction at the exit point worked very well as the entropy clearly shows an inverse correlation with accuracy of the classification model. The observations of the initial experiment conducted with the simple multi-exit classification shows that there can be great performance benefits when it is replaced by a bigger model with numerous layers.

5.1.2 Multi-exit Semantic Segmentation Models

When the state-of-the-art semantic segmentation architecture HRNet was modified to a multi-exit model, the average inference time was observed to be reduced significantly with a slight reduction in accuracy. However, the criteria used for deciding the early exit plays an especially key role which is discussed in detail in the subsequent sections.

5.1.2.1 Entropy based exit strategy

Though the entropy metric is an effective strategy for the classification problem, it did not perform well with semantic segmentation as it is a pixel-wise prediction task.

The uneven trends in the plot between accuracy and entropy may result in providing high-confidence decisions for predictions that are incorrect. Furthermore, DNNs are typically also prone to overfitting training data and providing overconfident predictions for unseen data. In such cases, the assumption that entropy has a negative correlation with the accuracy of the prediction is violated, and therefore any exit strategy solely based on entropy would fail, which was evident in the experiments. It is also observed that the entropy-based exit strategy makes no effort in estimating the expected performance gain of processing the image further. Therefore, in the case that the entropy of the prediction of an image is high at each exit branch the network would refrain from making an early exit, but in the end, the performance of the prediction may be no better than that of the first exit branch. In general, one would only want to process the image further if a performance gain is expected by this, which is not explicitly considered in this approach.

5.1.2.2 FFT based exit strategy

The FFT based exit approach was found to be innovative and to be superior than entropy-based values in the results. The mean FFT magnitude seeks to capture the essence of the entire image in a single value and attempts to describe the image in the frequency domain. In the studies, it was shown that the predictions for the image with lower accuracy had more jitter than the predictions for the image with higher accuracy, and the FFT value of the accurate results will fall within a certain range of frequency. This is evident in early exit points and is caused by the earlier processing blocks' smaller receptive fields. The thesis attempted to quantify this jitter by measuring the frequency components in the image as part of the FFT-based exit strategy. These assumptions turned out to be accurate, and on both datasets, the exit strategy performed better than the entropy-based approach. However, the approach continues to fall short in addressing some of the problems with entropy-based exit strategies, such as estimating the projected performance improvement and doing away with manual threshold selection for accuracy.

5.1.2.3 Auxiliary model-based exit strategy

The sophisticated method of using an auxiliary neural network for the performance analysis of host neural networks had improved performance than the previously discussed exit strategies. The idea was developed to harness the power of neural networks as universal function approximate, in this case trying to approximate the function, if the exit is good enough. The method based on classification was successful in including the expected performance gains in the subsequent exits. But it made the model rigid, as the minimum expected performance gain was embedded in the model and could not be changed without retraining the model. The regression-based strategy performed superior to the classification but persisted with overhead of selection of the threshold for what is the minimum expected accuracy.

5.1.2.4 Reinforcement learning based exit strategy

Reinforcement learning techniques with an idea to encode the current exit index in the decision-making process were proven to be successful. The exit strategy outperformed all the previous exit strategies with accuracies comparable with the model with no branch. The techniques also offered significant latency improvement and reduced processing requirements. This is due to the RL method at its core trying to find the threshold for different exits and in different metric ranges while considering a complex reward that includes latency as one of the input parameters. It reduces the computational overhead by mocking away from complex neural networks in the decision-making process of exit selection. The thesis tried to evaluate two sets of input parameters in the creation of the states. Both performed well but the FFT metric takes the edge because it gives a better representation for the predicted segmentation after the transformation.

5.2 Conclusion

The thesis study focuses on using multi-exit architecture to improve the latency of deep learning neural networks, making them more suited for time-critical real-time applications. It began by branching a simple classification model and observed the performance improvements in the form of reduced average inference time when majority of the less complex samples took the early exit. The same strategy was applied on a bulky semantic segmentation architecture like HRNet and the results were as expected in terms of latency benefits. With a slight drop in accuracy, the model's latency was lowered to half that of an un-branched model. However, the use of entropy as an exit strategy has been found to be challenging because it does not necessarily have a negative correlation with accuracy/performance. Exit strategy based on a new metric which is the FFT of the prediction enhanced the performance in terms of improved accuracy along with reduced latency. More sophisticated exit strategies like employing an auxiliary model and a reinforcement learning approach has been studied and evaluated which has the added advantages of predicting the performance improvement with further processing and elimination of threshold selection. The reinforcement learning based exit strategy appears to be the most efficient with accuracy comparable to that of the reference unbranched model and notable latency and computation reduction.
Introducing the multi-exit strategy to a real-time computer vision system, like the central vision system in Volvo GPSS would significantly improve the throughput and efficiency of the resource handling. It allows a differential allocation of these resources as the resources get free from the samples taking the early exits.

5.2.1 Future works

As a future work individual branches can be trained using the knowledge distillation process with the output from the main branch as the master. This would provide the opportunity to early branched to mimic the main branch with granular target parameters.

An evolutionary neural architecture search can be explored where the exit points and the additional trainable blocks can be selected efficiently to improve performance. This would reduce the overhead of the selection of optimal exit points for arbitrary neural network architecture.
Possibility of using Bayesian neural networks and evidential deep learning to estimate the confidence of each prediction which can be used in the early exit criteria. This would allow the estimation of uncertainty in the model and the data to help decide better multi-exit strategies.

Bibliography

[1] The need for edge video analysis - embedded computing design. `https://www.embeddedcomputing.com/technology/iot/edge-computing/the-need-for-edge-video-analysis`. (Accessed on 02/11/2022).

[2] Industry 4.0 people and robots work together on equal terms | Chalmers, 07 2021.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020.

[4] H. Chang, A. Hari, S. Mukherjee, and T. Lakshman. Bringing the cloud to the edge. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 346–351. IEEE, 2014.

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

[6] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation, 2017.

[7] H.-D. Cheng, X. H. Jiang, Y. Sun, and J. Wang. Color image segmentation: advances and prospects. *Pattern recognition*, 34(12):2259–2281, 2001.

[8] J. Fung. Computer vision on the gpu. *GPU Gems*, 2(649-666):34, 2005.

[9] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017.

[10] A. M. Hafiz and G. M. Bhat. A survey on instance segmentation: state of the art. *International journal of multimedia information retrieval*, 9(3):171–189, 2020.

[11] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks, 2016.

[12] S. S. Islam, S. Rahman, M. M. Rahman, E. K. Dey, and M. Shoyaib. Application of deep learning to computer vision: A comprehensive study. In *2016 5th international conference on informatics, electronics and vision (ICIEV)*, pages 592–597. IEEE, 2016.

[13] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation, 2016.

[14] E. J. Kirkland. Bilinear interpolation. In *Advanced Computing in Electron Microscopy*, pages 261–263. Springer, 2010.

[15] B. Kisacanin, S. S. Bhattacharyya, and S. Chai. *Embedded computer vision*. Springer Science & Business Media, 2008.

[16] A. Kouris, S. I. Venieris, S. Laskaridis, and N. D. Lane. Multi-exit semantic segmentation networks. *arXiv preprint arXiv:2106.03527*, 2021.

[17] Z. Liu, T. Darrell, and E. Shelhamer. Confidence adaptive anytime pixel-level recognition, 2021.

[18] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation, 2014.

[19] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[20] P. Panda, A. Sengupta, and K. Roy. Conditional deep learning for energy-efficient and enhanced pattern recognition, 2016.

[21] M. Phuong and C. Lampert. Distillation-based training for multi-exit architectures. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1355–1364, 2019.

[22] T. Pun. A new method for grey-level picture thresholding using the entropy of the histogram. *Signal processing*, 2(3):223–237, 1980.

[23] R. E. W. Rafael C.Gonzalez. *Digital Image Processing*. 2008.

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

[25] F. Salzenstein and W. Pieczynski. Parameter estimation in hidden fuzzy markov random fields and image segmentation. *Graphical models and image processing*, 59(4):205–220, 1997.

[26] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction, 2014,2015.

[27] S. Teerapittayanon, B. McDanel, and H.-T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE, 2016.

[28] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao. Deep high-resolution representation learning for visual recognition, 2019.

[29] S. D. Yanowitz and A. M. Bruckstein. A new method for image segmentation. *Computer Vision, Graphics, and Image Processing*, 46(1):82–95, 1989.

[30] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation, 2018.

[31] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, and K. Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. *CoRR*, abs/1905.08094, 2019.

[32] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network, 2016.