Lightweight Deep Learning Architectures for Semantic Scene Segmentation for Applications in Autonomous Driving

A thesis submitted in fulfilment of the requirements

for the degree of Doctor of Philosophy

by

Saquib Mazhar

DEPARTMENT OF ELECTRONICS AND ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

June 2025

Certificate

It is certified that the work contained in this thesis entitled **"Lightweight Deep Learning Architectures for Semantic Scene Segmentation for Applications in Autonomous Driving"** by **Saquib Mazhar** has been carried out under my supervision and that it has not been submitted elsewhere for a degree.

Prof. M.K. Bhuyan
Professor
EEE Department
Indian Institute of Technology Guwahati

Prof. Shaik Rafi Ahamed
Professor
EEE Department
Indian Institute of Technology Guwahati

Declaration

This is to certify that the thesis titled **"Lightweight Deep Learning Architectures for Semantic Scene Segmentation for Applications in Autonomous Driving"** has been authored by me. It presents the research conducted by me under the supervision of **Prof. M.K. Bhuyan** and **Prof. Shaik Rafi Ahamed**.

To the best of my knowledge, it is an original work, both in terms of research content and narrative, and has not been submitted elsewhere, in part or in full, for a degree. Further, due credit has been attributed to the relevant state-of-the-art and collaborations with appropriate citations and acknowledgments, in line with established norms and practices.

Saquib Mazhar
Roll No. 186102113
EEE Department
Indian Institute of Technology Guwahati

Abstract

Semantic segmentation, a critical task in computer vision, involves assigning a meaningful label to every pixel in an image, allowing machines to understand visual data at a granular level. It is vital in various applications, including medical imaging, video surveillance, and augmented reality. However, it is particularly critical in autonomous driving, where vehicles rely on accurate pixel-level predictions to navigate and make real-time decisions. Autonomous systems must process complex scenes and identify and differentiate between road users, obstacles, traffic signals, and other environmental elements in diverse and dynamic settings. Achieving this pixel-wise accuracy in real-time, however, presents several key challenges.

One of the primary challenges is the need for a balance between high segmentation accuracy and computational efficiency. Autonomous vehicles operate with limited onboard computational resources, making it difficult to deploy large, complex models in real-time environments. Additionally, road scenes often include small and infrequent objects, such as pedestrians, traffic lights, and cyclists, which are crucial for safe navigation but are typically harder to segment accurately. Existing methods have sought to address these issues primarily through the use of deep learning, particularly CNN-based architectures. Several techniques have been proposed both at macro-architecture level such as multi-branch network, encoder-decoder networks, vision transformers, as well as at fundamental block level such as spatial pyramid pooling modules, attention modules, and context learning modules. However, many of these models rely on large, pre-trained networks that are computationally expensive and unsuitable for real-time applications. Additionally, several modules are not optimised for lightweight models, leading to higher latency and memory usage. As a result, the trade-off between accuracy and real-time processing capabilities

remains challenging, particularly for resource-constrained environments like autonomous vehicles.

To address these challenges, novel solutions have been developed in this thesis to improve both segmentation accuracy and computational efficiency. One of the significant contributions is the introduction of a composite loss function that tackles class imbalance and poor boundary accuracy. By combining traditional cross-entropy loss with boundary and region-based losses, this approach effectively enhances the segmentation of small objects and improves accuracy at object boundaries. Extensive experimentation on datasets like Cityscapes demonstrates that this composite loss function leads to substantial accuracy gains, particularly for smaller and infrequent classes.

In addition to optimizing the loss function, new network architectures have been designed to meet the demands of real-time segmentation. A notable contribution is the inverse-residual dilation pyramid network (IRDPNet), a lightweight CNN specifically developed for resource-constrained environments. By employing an inverse-residual bottleneck and a multi-scale dilation pyramid, IRDPNet achieves a significant reduction in parameters while maintaining high segmentation accuracy. The network reduces model size by half compared to existing architectures, yet it delivers competitive or superior performance, making it ideal for real-time autonomous driving applications. Further advancements include the integration of attention mechanisms within the network architecture. The Block Attention Network (BANet) incorporates a modified attention method to efficiently capture global context and long-range dependencies. This enhances the model's ability to interpret complex scenes while maintaining low computational overhead, ensuring that the network can operate in real time without sacrificing accuracy.

Another challenge lies in the efficient incorporation of context awareness in the model, which enhances the ability to distinguish between similar objects in a scene and reduces confusion. The existing techniques majorly exploit high-level semantic features through attention mechanisms, which may not give optimal segmentation as they lack crucial spatial details. To address this issue, the Context-Guided Multi-scale Attention Network

(CGMANet) is proposed to simultaneously use low-level and high-level features to generate context-aware attention weights. The network further proposes a hybrid basic module that combines local semantics and contextual information at the block level. By combining these techniques, CGMANet ensures precise segmentation while preserving computational efficiency. This architecture is optimized for real-time deployment on embedded GPUs, demonstrating its practical applicability in autonomous vehicles that require fast inference and decision-making.

Acknowledgements

First and foremost, I express my deepest gratitude to the Almighty for His countless blessings, guidance, and strength throughout this journey.

I extend my heartfelt appreciation to my supervisors, Prof. M.K. Bhuyan and Prof. Shaik Rafi Ahamed, whose invaluable guidance, unwavering support, and encouragement have been instrumental in shaping this research. I am deeply grateful for their insightful advice and support. Their mentorship has been an inspiration and has significantly contributed to my academic and personal growth. I wish to extend my sincere thanks to Prof. Ekram Khan of Aligarh Muslim University for imbibing early research motivation in me.

I am grateful to the doctoral committee chairman Dr. Kannan Karthik and members Dr. Rishikesh Dilip Kulkarni, Prof. Samit Bhattacharya and Dr. Teena Sharma for the insightful comments and constructive criticisms, which has shaped the thesis work in the current form.

A special note of gratitude goes to my mother, whose immense dedication, sacrifices, and unconditional love have been my greatest source of strength. Her unwavering belief in me has been a pillar of support throughout this journey. I would also like to express my gratitude to my father and wife for their continuous motivation and for having stood by me.

I would also like to extend my sincere thanks to my friends Nadeem Atif, Riyaz Khan, Rijuban Rangslang, H. Pallab Jyoti Dutta and Rajarshi Goswami, who have been a constant source of motivation, knowledge-sharing, and companionship. I am also thankful to my lab mates Allen Patnaik, Harhsal Chaudhari, Brij Nandan Tripathy, Vineet Kumar, Bibek Goswami, Kamal Kumar and Nitin Yadav for their help and support. My seniors, Nayan Moni Baishya, Aniruddha Mazumdar, Pradipta Sasmal, and Rahul Sharma, deserve a special mention for their motivation and technical guidance. Their encouragement and camaraderie have made this journey at IIT Guwahati both enriching and enjoyable.

Finally, I acknowledge everyone who has, in any way, contributed to this thesis and my academic growth. Thank you all.

Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial intelligence
BDD	Berkeley deep drive
BN	Batch normalization
CE	Cross Entropy
CNN	Convolutional neural network
CPU	Central processing unit
Conv2d	2 dimensional convolution
DL	Deep learning
FC	Fully connected
GAN	Generative adversarial network
GAP	Global average pooling
GPU	Graphical processing unit
HD	High definition
IoU	Intersection over union
KITTI	Karlsruhe institute of technology and Toyota technological institute
mIoU	Mean IoU

MSE	Mean square error
PReLU	Parametric rectified linear unit
ReLU	Rectified linear unit
RNN	Recurrent neural network

List of Publications

Publications from Thesis

Journals

1. Saquib Mazhar, N. Atif, M.K. Bhuyan, S.R. Ahamed, "Block attention network: A lightweight deep network for real-time semantic segmentation of road scenes in resource-constrained devices", Engineering Applications of Artificial Intelligence, Vol. 126, Part C, 2023, doi: *10.1016/j.engappai.2023.107086*.

2. Saquib Mazhar, N. Atif, M.K. Bhuyan and S.R. Ahamed, "Rethinking DABNet: Lightweight Network for Real-time Semantic Segmentation of Road Scenes," in IEEE Transactions on Artificial Intelligence, vol. 5, no. 6, pp. 3098-3108, June 2024, doi: *10.1109/TAI.2023.3341976*.

3. Saquib Mazhar, N. Atif, M.K. Bhuyan and S.R. Ahamed, "Context Guided Multiscale Attention for Realtime Semantic Segmentation of Road-scene," (under review) in IEEE Transactions on AI.

Conferences

1. Saquib Mazhar, N. Atif, M.K. Bhuyan and S.R. Ahamed, "HCINet: A Hierarchical Approach of Context Integration in Real-time Semantic Segmentation for Autonomous Driving," 2023 IEEE 20th India Council International Conference (INDICON), Hyderabad, India, 2023, pp. 1174-1179, doi: *10.1109/INDICON59947.2023.10440773*.

2. Saquib Mazhar, N. Atif, M.K. Bhuyan and S.R. Ahamed, "Improving Semantic Segmentation Performance of Deep Networks for Autonomous Driving through Loss Functions," 2023 IEEE Guwahati Subsection Conference (GCON), Guwahati, India, 2023, pp. 1-6, doi: *10.1109/GCON58516.2023.10183416*.

3. Saquib Mazhar, N. Atif, M.K. Bhuyan and S.R. Ahamed, "S-Net: A Lightweight Real-time Semantic Segmentation Network for Autonomous Driving," in Computer Vision and Image Processing - 8th International Conference, CVIP 2023. Communications in Computer and Information Science, Part II, vol 2010, pp.147-159, Springer, 2023, doi: *10.1007/978-3-031-58174-8_14*.

Dedicated to my mother.

Chapter 1

Introduction

1.1 Background

Semantic segmentation is a pivotal task in the field of computer vision, aimed at assigning a meaningful label to every pixel in an image [3]. Unlike traditional image classification [4] that assigns a single label to the entire image or object detection [5] that localizes objects with bounding boxes, semantic segmentation provides a pixel-level understanding of the scene. This technique plays a critical role in various real-world applications such as autonomous driving, medical image analysis, video surveillance, and augmented reality.

At its core, semantic segmentation involves partitioning an image into multiple segments, each representing a distinct object or region with semantic meaning. For instance, in a street scene, semantic segmentation can differentiate between pedestrians, vehicles, road surfaces, and buildings, assigning a unique label to every pixel belonging to these categories. This detailed understanding enables machines to comprehend visual scenes with a level of granularity similar to human perception. Modern day semantic segmentation algorithm typically leverages deep learning architectures, particularly convolutional neural networks (CNNs) [6, 7, 8], owing to their remarkable capability to learn complex features from raw pixel data. These networks are trained on large datasets annotated with pixel-wise labels, allowing them to capture intricate patterns and relationships within images.

Furthermore, recent advancements in semantic segmentation focus on achieving real-time performance and efficiency, especially in resource-constrained environments like embedded systems and mobile devices [7, 9]. Development of lightweight architectures in addition to techniques such as network pruning and knowledge distillation enable the deployment of

semantic segmentation models on edge devices without compromising performance. Autonomous driving is one such application that requires lightweight architecture pertaining to the availability of limited computation power. Moreover, the time-critical nature further mandates the algorithm to work in real-time.

In general, semantic segmentation stands as a fundamental task in computer vision, enabling machines to comprehend visual scenes at a pixel-level granularity. With ongoing research and innovation, semantic segmentation continues to evolve, driving advancements across a myriad of domains and fostering the development of intelligent systems with enhanced perception capabilities [10].

1.2 Semantic Segmentation in Autonomous Driving

Scene perception is an important and challenging task for intelligent vehicles due to the wide variety of environments in which they operate. While complex systems rely on multimodal sensor data to comprehend the surroundings, image-based techniques have gained traction due to the lower capture cost and advancements in computer vision. Semantic segmentation is among the core techniques of autonomous driving for detailed understanding of the surrounding environment, which is essential for safe and reliable navigation. By understanding the semantic meaning of different objects and regions in the scene, the vehicle can make more intelligent decisions regarding navigation, trajectory planning, and interaction with other road users. It is a pixel-level image classification task requiring complex processing algorithms that can be used to improve various aspects of autonomous driving some of which are:

1. Scene Understanding: Semantic segmentation enables the vehicle to understand the scene by segmenting the raw sensor data, typically from cameras mounted on the vehicle, into meaningful categories such as roads, vehicles, pedestrians, cyclists, traffic signs, and buildings. This pixel-level understanding helps the autonomous vehicle make informed decisions based on the detected objects and their spatial relationships.

2. Obstacle Detection and Tracking: By segmenting the scene into different semantic classes, the autonomous vehicle can accurately detect and track obstacles such as other vehicles, pedestrians, and cyclists. This information is crucial for path planning and obstacle avoidance to ensure safe navigation through complex urban environments.

3. Lane Detection and Lane-Keeping Assistance: Semantic segmentation assists in lane detection by segmenting the road surface from the surrounding environment. This allows the autonomous vehicle to precisely identify lane boundaries and maintain its position within the lane, thereby providing lane-keeping assistance to the driver or autonomously controlling the steering to stay in the lane.

4. Traffic Sign Recognition: Semantic segmentation helps in recognizing and understanding traffic signs by segmenting them from the background scene. This allows the autonomous vehicle to interpret traffic regulations, such as speed limits, stop signs, yield signs, and traffic signals, and adjust its behavior accordingly.

5. Semantic Mapping: Semantic segmentation contributes to building high-definition semantic maps of the environment, which contain detailed information about the semantic classes and their spatial distribution. These maps serve as a valuable resource for localization, path planning, and decision-making, enabling the autonomous vehicle to navigate efficiently and safely.

Overall, semantic segmentation plays a critical role in enhancing the perception capabilities of autonomous driving systems, enabling them to interpret and navigate complex real-world environments safely and efficiently.

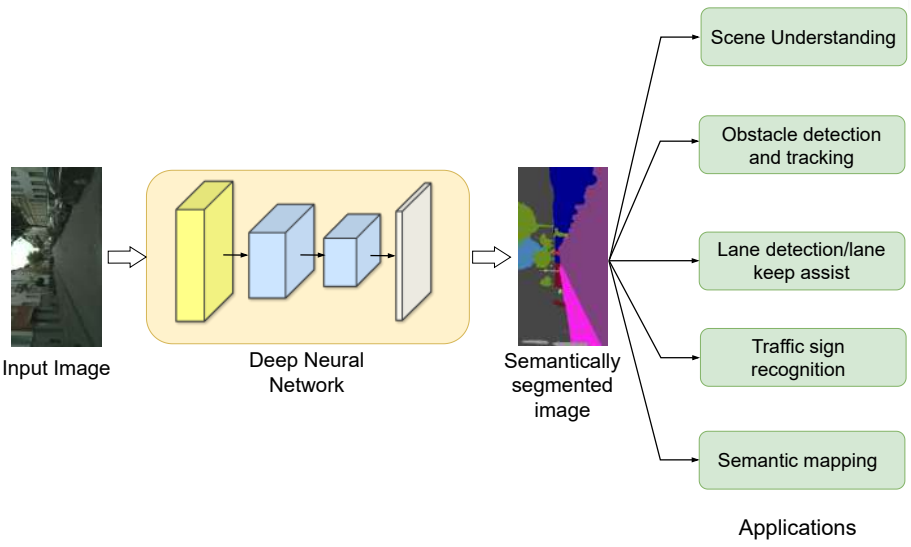


Figure 1.1: Applications of semantic segmentation in autonomous driving.

1.3 Datasets

1.3.1 Cityscapes

The Cityscapes dataset [1] offers a comprehensive benchmark for evaluating semantic segmentation algorithms in urban road-scene environments. Its large scale, fine-grained annotations, diversity, and benchmarking infrastructure make it a valuable resource for advancing research in computer vision tasks related to urban scene analysis and autonomous driving.

It contains over 5,000 finely annotated images, each with a resolution of 2048×1024 pixels captured in various urban environments across multiple cities of Germany. Every image is densely annotated with pixel-level semantic labels, providing detailed information about the various objects and regions present in the scene. The images are annotated with 30 different semantic classes, including common objects such as road, sidewalk, building, car, pedestrian, cyclist, traffic sign, and vegetation. However, only 19 classes are used for training and evaluation on the official server.

The dataset encompasses a wide range of urban scenes, capturing diverse environmental conditions, lighting variations, weather conditions, and architectural styles. This diversity makes Cityscapes well-suited for evaluating the robustness and generalization capabilities of semantic segmentation models across different real-world scenarios.

1.3.2 Berkeley Deep Drive

The Berkeley DeepDrive (BDD) 100K dataset [11] is a large-scale diverse driving dataset designed for various computer vision tasks, including semantic segmentation. It consists of over 100,000 high-resolution images captured from dashcams mounted on vehicles driving in urban environments of New York, Berkeley, San Francisco and Bay Area. These images are of size 1280×720 pixels and provide rich visual information about different traffic scenarios, road conditions, and environmental factors encountered in real-world driving scenarios.

Each image in the BDD 100K dataset is densely annotated with pixel-level semantic labels, similar to the Cityscapes dataset. The annotations provide detailed information about the various objects and regions present in the road-scene. In addition to the semantic segmentaion task, BDD dataset is annotated for driving tasks such as road object detection, lane markings, and drivable area segmentation.

1.3.3 CamVid

The Cambridge-driving Labeled Video Database (CamVid) dataset [12] is a widely used benchmark dataset comprising video sequences captured from the perspective of a vehicle navigating through urban environments. It consists of 701 images extracted from video sequences, each with a resolution of 960×720 pixels. While smaller in scale compared to some other datasets, CamVid remains valuable for research due to its detailed annotations and specific focus on urban scenes.

The images in the dataset includes 32 semantic classes, covering a wide range of urban elements such as roads, sidewalks, buildings, vehicles, pedestrians, cyclists, trees, and sky. However only a subset of 12 classes is used for training and evaluation of the model.

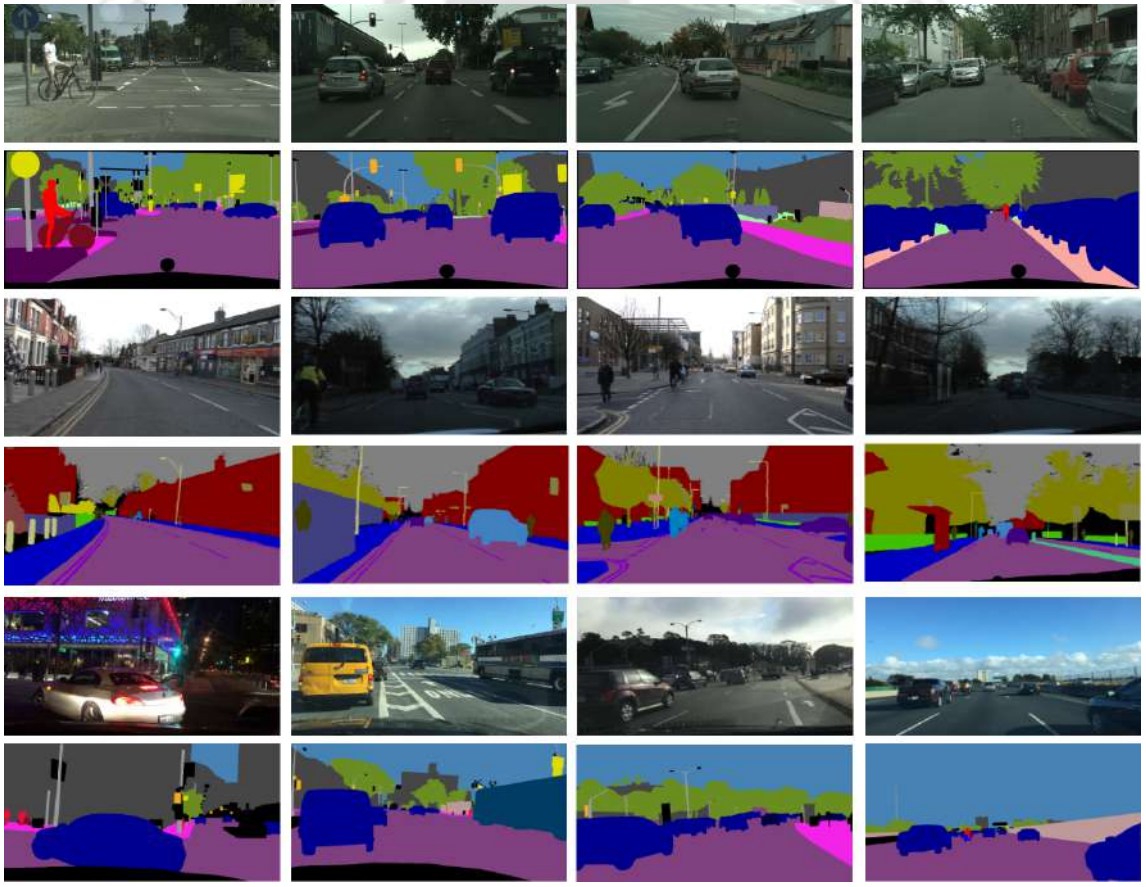


Figure 1.2: A few sample images and their labels from the datasets used. Row 1: Cityscapes; Row 3: CamVid; Row 5: BDD100K.

1.3.4 KITTI

The KITTI Semantic Dataset [13] is a widely-used benchmark consisting of high-resolution images captured from a car-mounted stereo camera system, with a native resolution of 1242 × 375 pixels. It provides pixel-level semantic annotations for diverse environments, including urban, rural, and highway scenes. The dataset contains 200 training images and 200 testing images, with a total of 34 semantic classes, though commonly only 19 classes are used for evaluation. Additionally, the dataset offers depth maps, 3D point clouds, and GPS data, enabling multimodal approaches to semantic segmentation and object detection.

Figure 1.3: A few sample images and their labels from the KITTI Semantic Segmentation benchmark

1.4 Literature Survey

Semantic segmentation is fundamental for several computer vision applications, including intelligent vehicles. Consequently, it has drawn the attention of the research community for several decades. Initial research was primarily focused on improving the accuracy of the segmentation algorithms, as the real-time application was limited due to technological bottlenecks. Semantic segmentation algorithms can be broadly divided into classical and deep learning (DL) based.

1.4.1 Classical methods

While deep learning has revolutionized this field, classical methods laid the groundwork and remain valuable for specific scenarios. These algorithms are based on feature extraction and representation used as image descriptors. While the local feature extractors search for regions or local patches of specific features like corners and edges, the global features are modelled in the form of colours and textures. The classical methods can be broadly classified as follows:

1. Thresholding [14]: This is the simplest and fastest method. It assumes objects have distinct intensity values compared to the background. A threshold value is chosen, and pixels brighter (or darker) than the threshold are assigned to a specific class, while the rest belong to another. This works well for high-contrast images with well-separated objects, but struggles with variations in lighting or overlapping objects.

2. Edge-Based Segmentation [15]: Edges often mark object boundaries. This method identifies edges using filters like Sobel or Canny edge detectors. The Canny edge detector is another popular edge detector for segmentation. It computes the intensity gradients of the image to detect edges, then finds the maximal gradients among neighbours via non-maximum suppression. Finally, two thresholds are applied to select the real edges. Edges are then connected to form closed contours that enclose objects. However, noise can lead to spurious edges, and weak edges at object boundaries can cause segmentation errors.

3. Region-Based Segmentation: This approach groups pixels with similar characteristics (intensity, color) into regions. Common methods include:

Region Growing [16] Starting from a seed point, pixels with similar properties are added to the region iteratively until a stopping criterion is met. This can be sensitive to initial seed placement and can struggle with irregular object shapes. Region Merging: The image is initially divided into small regions. Adjacent regions with similar properties are merged iteratively to form larger, more coherent object regions. This can be computationally expensive for large images. Split and Merge [17] This method combines both approaches. The image is initially over-segmented (divided into many small regions) and then merged based on similarity until desired regions are formed. This offers more control but requires careful parameter tuning.

4. Clustering-Based Segmentation [18]: This treats pixels as data points in a feature space (e.g., intensity, color). Clustering algorithms [19] like K-means or hierarchical clustering group pixels into a predefined number of clusters, each potentially representing an object class. This is effective for segmenting objects with distinct color or intensity properties but may not work well for complex shapes or overlapping objects.

5. Watershed Transformation [20]: This method treats the image as a topographic landscape, with intensity representing elevation. Markers are placed at object minima (like valleys). The image is flooded from these markers, with pixels assigned to the watershed basin (region) they belong to. This is well-suited for segmenting objects separated by clear intensity gradients but can be sensitive to noise and over-segmentation in flat regions.

All these methods require the selection and handcrafting of low-level features, which becomes more cumbersome as the number of segmentation classes increases. Moreover, choosing optimal parameters for these algorithms can be challenging and image specific. These algorithms often struggle with complex images with noise and variations in lightning, or intricate shapes The recent advancements in Deep Learning has led to above human-level performance in the field of computer vision.

1.4.2 Deep Learning methods

The success of deep neural networks in object detection motivated researchers to explore the possibility of applying them in semantic segmentation tasks. Towards this end, fully convolutional network (FCN) [3] was the first attempt to design an end-to-end fully trainable deep network for semantic segmentation (Figure 1.4). It can take an arbitrary size input to produce a segmentation map, thus relaxing the earlier condition of fixed input size. FCN uses a pre-trained VGG-16 [6] backbone developed for ImageNet classification. The fully connected layers of VGG are replaced with convolutional layers to produce dense 2D class-activation maps. Following the success of FCN, most of the ensuing algorithms were formulated to solve the structured pixel-wise labelling problems of semantic segmentation based on CNN. These networks acquire a coarse label map by classifying individual local regions within the image. Subsequently, they employ bilinear interpolation to achieve pixel-level labelling refinement. However, FCN-based semantic segmentation has some critical limitations. Firstly, due to the fixed-size receptive field, it can only learn single-scale semantics. This results in label prediction with only local context, ignoring the non-local semantics. Secondly, the input to the upsampling layers to restore the image resolution is very coarse. In the absence of any learnable upsampling layers, the output lacks detailed object structures. Restoring the original resolution and preserving the spatial information simultaneously still remained a challenge.

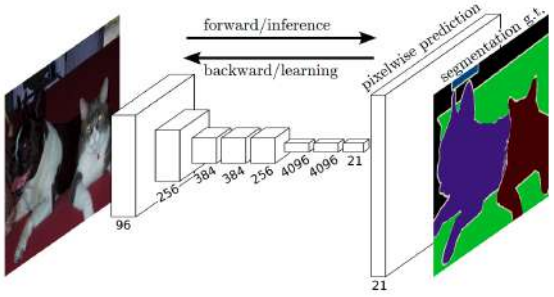


Figure 1.4: Architecture of FCN. The blocks represent the feature maps. The number of channels are mentioned at the bottom of each block.

To solve this problem to an extent, DeconvNet [21] proposed a symmetric encoder-decoder architecture with a learnable deconvolution network (Figure 1.5). The decoder uses learned filters along with unpooling operations to reconstruct the shape of an input object. Following the DeconvNet, encoder-decoder architecture became a popular method to restore the lost spatial details. SegNet [22] removed the fully connected layers in DeconvNet to improve computational speed; however, its accuracy suffered due to limited receptive field.

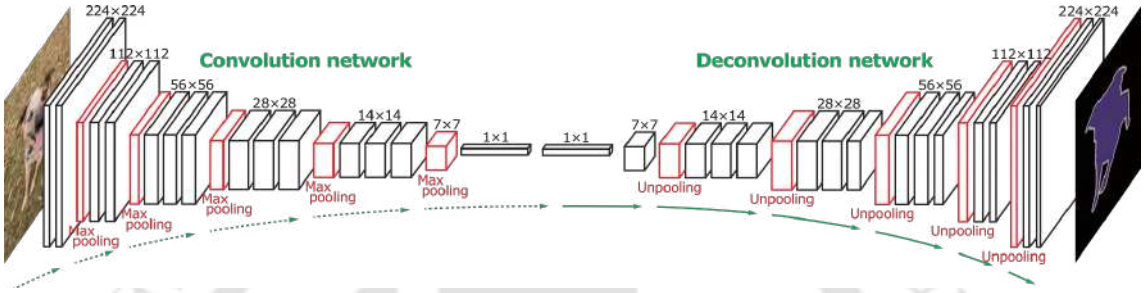


Figure 1.5: Architecture of DeconvNet. The feature map dimension is mentioned at the top of each block.

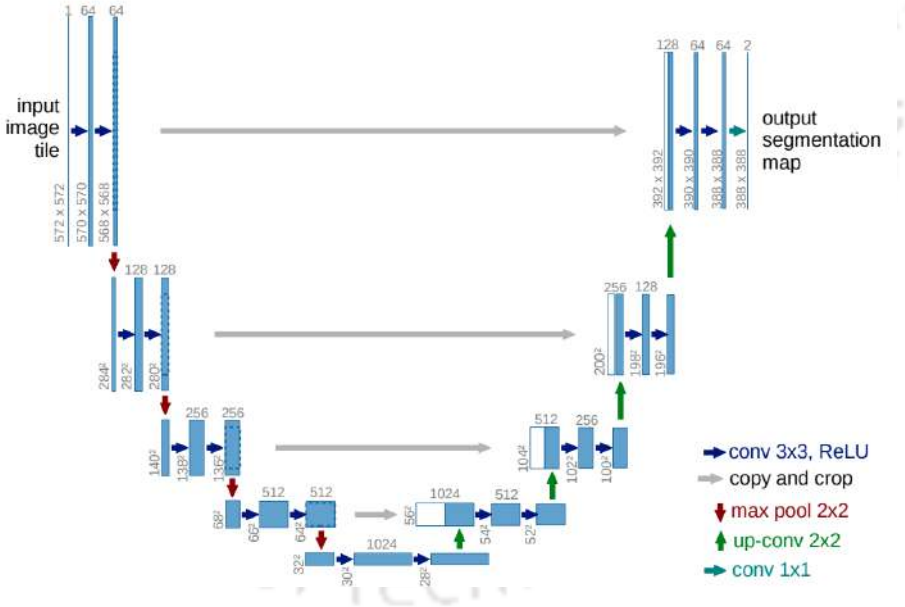


Figure 1.6: Architecture of UNet. The number of channels is mentioned at the top and the feature map dimensions on the left of each box.

U-Net [23] was another encoder-decoder model proposed for medical image segmentation. In addition to deconvolution, the decoder concatenates the corresponding feature maps from the encoder after each upsampling operation. This novel approach of skip connection between the encoder and decoder preserved the spatial information locally (refer Figure 1.6). Consequently, it has been adopted for numerous segmentation tasks, and many variants have been proposed [24, 25, 26]. The conventional convolution blocks in U-Net

was replaced with dense blocks in FC-DenseNet [27] to provide deep supervision with feature reuse. Similarly LinkNet [28] replaced the encoder backbone with a ResNet-18 and performed addition in place of concatenation of the feature maps in the decoder. RefineNet [29] introduced long-range residual connections between encoder and decoder for fusing multi-level semantic features. HyperSeg [30] adapted the UNet architecture to divide input image into patches. It used a specifically designed context head at the end of the encoder backbone to generate weights which are of different resolutions as they are generated after varying strides. The patches are multiplied by the context weights before upsampling.

However, due to the limited receptive field, U-Net-based architectures were ineffective in encoding information from the global context, which is vital for road-scene images. To this end, researchers proposed techniques for global context aggregation discussed subsequently.

The presence of multiple scales of objects in a traffic scene and the local connectivity of the convolution kernels pose a challenge for effective semantic segmentation. The size of the receptive field of a dense convolution kernel is often considered too small to capture significant object context in the image [31]. The straightforward solution can be thought of as the use of dilations in the kernel to gather the rich context of the objects in a larger feature map. However, it was found that the resulting feature maps lack details of isolated and smaller objects. DeepLab [32] solved this problem by exploiting multi-scale context features by using dilated convolutions in a pyramid structure. They named it the atrous pyramid pooling module (ASPP). The module consisted of multiple atrous convolution kernels in parallel with different dilation rates inside the pyramid. This led to feature extraction to varying scales along with an enlarged receptive field. The pyramid pooling structure was quickly adopted by several architectures that followed. Instead of employing atrous convolutions, PSPNet employs a sequence of parallel average pooling operations for downsampling the input feature maps and acquiring global contextual representations. DenseASPP [33] extended the DeepLab by adding more dilated convolutions to the ASPP module. This further improved the feature encoding capability of the module. ParseNet [34] designed a global pooling module to extract global context information and added it to local features after normalization. While LBN-AA [35] uses a lightweight backbone and a distinctive ASPP at the end of the encoder, it adds a parallel branch for spatial detail preserving.

Self-attention mechanism [36] is another technique introduced to model the context information. It works by aggregating context features through long-range dependencies

between the feature maps. The attention weights generated throught the aggregated context features are mutliplied to the higher order feature maps to emphasize import object features in the image. DANet [37] utilized a dual attention scheme to exploit spatial and channel interdependecies through respective attention modules. This scheme significantly improved the segmentation result by modelling rich global context over local features. OCNet [38] used pixels from the same object category rather than the whole image to calculate interlaced sparse self-attention. This reduced the computations, and the model could learn better about interdependencies among objects. CCNet [39] aggregated the features in both horizontal and vertical directions instead of collecting all the contextual information over the feature maps, which reduced space complexity and led to a faster inference. Asymmetric non-local neural net [40] designed non-local attention blocks, inspired from [41], by replacing the dense matrices for calculating the query and value with sparse pooling pyramids. Although these methods reduce the memory requirement and have faster inference speed, their accuracy drops significantly compared to the dense pyramid pooling-based methods.

Realtime semantic segmentation methods

Similar to recognition models, deeper and wider networks generally achieve improved semantic segmentation results, however at the expense of increased computational demands and greater memory usage. Given the constraints of computational resources in embedded devices and autonomous vehicles, researchers have been investigating the trade-off between accuracy and inference speed across various semantic segmentation methods. To this end numerous techniques have been proposed to reduce the computational complexity of the lightweight networks. These include depthwise separable convolutions [42, 43, 44, 45], convolution factorization [46, 47, 48], and zoomed convolution [49]. The architectures discussed subsequently incorporate these tecniques for efficient segmentation in real-time.

ENet [50] was the initial lightweight algorithm for real-time semantic segmentation of road scenes. It uses bottleneck basic blocks and follows an encoder-decoder architecture. The encoder introduces early downsampling that reduces the feature map dimensions of the intermediate layers and reduces the overall computational complexity. Consequently, ENet was significantly faster than other networks while it acheived considerable accuracy. ERFNet [47] replaced the bottleneck blocks in the ENet with non-bottleneck 1D. Moreover, it factorized the 2D convolution kernels into 1D asymmetric convolutions to reduce the parameters. The decoder was designed with deconvolution layers, which improved accuracy despite reducing the inference speed.

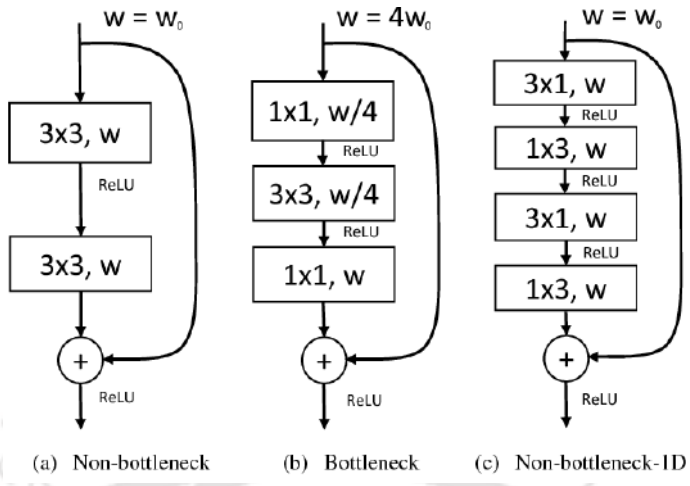


Figure 1.7: Basic blocks of ENet and ERFNet. (a) ENet non-bottleneck (b) ENet bottleneck (c) ERFNet non-bottleneck-1D.

A series of ESPNets [51, 52] with a series of parallel depth-wise convolutions were proposed to reduce the parameters and improve inference speed. The pyramid structure in basic block aggragated rich context information and improved accuracy. These models were significantly lightweight and had real-time inference, however, their accuracy was restricted due to limited receptive fields. A solution to this problem was proposed in ICNet [53] and ContextNet [54] in the form of multi-branch architecture. Images at multiple scales are given as input. Specifically, a low-resolution image is an input for the deeper branch to extract global context, and a large-resolution image is for the shallow branch to preserve spatial details. BiSeNet [55] proposed dual branch network in which same image is given as input to both semantic branch and spatial branch. An attention based refinement module is designed to enhance the spatial features prior to fusion with the semantic feature. BiSeNetv2 [56] merged the initial layers of the two branches by feature sharing. Consequently, the inference speed was increased with a significant reduction in computation. MLFNet [57] also follows a similar two-branch encoder structure. However, it varies in the decoder structure as it uses multi-branch fusion. Essentially, it fuses three-level features from the two encoder branches and the previous decoder block. CABiNet [58] introduced attention based pyramid pooling module in the context branch of BiSeNet along with attention based feature fusion.

DDRNet [59] adapted a multipath approach based on HRNet [60]. However, for a lightweight structure, it removed two branches out of four. DDRNet proposed a bidirectional feature enhancement strategy to improve the accuracy of lightweight backbones in which low level context features from high-resolution branch are added to the semantic branch. Similarly, the high level semantic features from the low resolution branch are added to the context

branch at multiple stages. This extensive fusion strategy helps the network to learn rich feature representations and improve accuracy.

Though vanilla UNet-based architectures were deemed to be inefficient for resource-constrained environments, methods like FASSD-Net [61] and HyperSeg [30] follow UNet to design a dense segmentation network by carefully redesigning the backbone. FASSD-Net extensively uses the HarDBlock from HarDNet [25] in its core encoder backbone due to the highly optimised memory traffic. In addition, a lightweight pyramid fusion technique is employed to improve the context information. HyperSeg uses a nested UNet design to process images in small pathces. The backbone and a specifically designed context head both follow the UNet. The context head generates weights for the image patches of multiple resolution. This patch-wise prcessing scheme reduces the computation load.

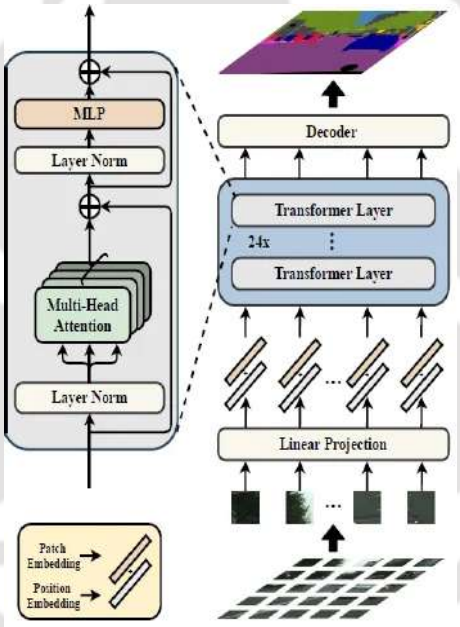


Figure 1.8: Semantic segmentation transformer (SETR). The multi-head attention and MLP (multi-layer perceptron) layers consist of fully connected layers.

Transformer-based architectures

Transformers [41] are increasingly finding applications in various machine translation tasks due to their high accuracy while they are regarded as next breakthrough in machine learning after CNNs. Vision transformer (ViT) [62] adapts it to computer vision task such as segmentation and detection. ViT uses a language model to represent an image as a succession of 16×16 image patches and uses multi-head self-attention modules with FC layers to learn the sequence dependencies. Segmenter [63] employs transformers for

encoding and decoding, whereas SETR [64] segments images using a transformer encoder and a CNN decoder (Figure 1.8). Trans4trans [65] uses a transformer parsing module to jointly learn features between a self-attention encoder and a decoder. Compared to models that only use CNN, these models are highly accurate. However, due to the fully connected layers in their architecture and having quadratic complexity, dense operations are required between the sequences. Transformer-based networks are, therefore, computationally costly since they need a lot of floating-point operations (FLOPs). This makes deployment difficult for embedded devices with limited resources.

1.5 Major challenges

In spite of the remarkable success in deep learning, several challenges remain unaddressed while designing a lightweight neural network for semantic segmentation. The first challenge is preserving spatial details while extracting high-level features in CNNs. CNNs typically use pooling and downsampling operations to increase receptive fields and reduce computational costs. However, these operations often lead to a loss of spatial information, which is crucial for accurate segmentation. Recent research [26] has highlighted the importance of retaining spatial details for producing high-quality segmentation results. Traditional CNN architectures [3, 27, 66] tend to prioritize encoding high-level contextual information with consecutive layers and large receptive fields, potentially neglecting low-level spatial details necessary for precise segmentation. This imbalance can result in segmentation errors, especially in scenarios where fine-grained distinctions are essential, such as distinguishing between different types of road markings or recognizing small obstacles.

To address this challenge, modern semantic segmentation approaches have explored various strategies given below:

- Multi-scale Feature Fusion: Some approaches [55, 56, 59] leverage inputs at different resolutions to extract features at multiple levels of abstraction. By incorporating features from both high and low resolutions, these methods aim to capture both high-level semantics and fine-grained spatial details. However, implementing such architectures often requires complex designs and increased computational resources.
- Feature Pyramid Networks (FPN): FPNs [67, 33, 68] have gained popularity for their ability to generate feature pyramids with multiple scales while maintaining computational efficiency. These networks use lateral connections to fuse features

from different stages of the CNN architecture, enabling the model to leverage both high-level semantic information and low-level spatial details.

- Encoder-Decoder Architectures: Another approach involves using encoder-decoder architectures [47, 51, 52, 23], where the encoder extracts high-level features, and the decoder reconstructs the segmentation map at the original resolution by incorporating low-level features. This design helps preserve spatial information while benefiting from the high-level context learned by the encoder. Variants such as U-Net and its derivatives have demonstrated effectiveness in semantic segmentation tasks by explicitly addressing the spatial detail preservation challenge.

- Dilated Convolutions: Dilated convolutions [69, 70] also known as atrous convolutions, offer an alternative to downsampling operations by introducing holes in the convolutional filters. This allows the network to expand its receptive field without reducing spatial resolution, thereby preserving fine-grained details. Dilated convolutions have been successfully incorporated into CNN architectures for semantic segmentation, contributing to improved performance in capturing both local and global context.

Addressing the challenge of preserving spatial details in semantic segmentation requires careful architectural design and innovation. By incorporating techniques such as multi-scale feature fusion, encoder-decoder architectures, and dilated convolutions, researchers aim to strike a balance between capturing high-level semantics and preserving fine-grained spatial information, ultimately leading to more accurate and reliable segmentation results for applications like autonomous driving.

The second challenge is handling multi-scale objects in input images, which poses a significant hurdle in semantic segmentation tasks, particularly when using Deep Neural Networks. Traditional CNN architectures, which rely on local connectivity properties of convolutional filters, struggle to capture sufficient contextual information to recognize large objects effectively. While dilated convolutions have been employed to enlarge receptive fields and capture broader context, they often lead to dominant representations of large objects, neglecting details of small or isolated objects and resulting in subpar performance. To address this challenge, researchers have explored several strategies as follows:

- Multi-Scale Feature Encoding [47, 51, 71, 72]: Some approaches utilize a combination of dilated and dense convolutions operations in the basic block structure to locally encode multi-scale features. By aggregating features from different scales,

these methods aim to capture local semantics of objects of varying sizes. However, balancing the representations of large and small objects remains crucial while designing the network.

- Aggregated Multi-Scale Context: Several methods [73, 54, 58, 60] have been developed to combine features generated by various dilated convolutions or pooling operations to yield aggregated multi-scale contextual information. By integrating features from different scales, these methods seek to enhance the network's ability to recognize objects across a range of sizes. However, achieving an optimal balance between capturing fine details and contextual information remains a delicate trade-off.

- Self-Attention Mechanism: Recent advancements in semantic segmentation have explored the use of self-attention mechanisms [37, 74, 75] to model pixel relations and capture long-range dependencies within feature maps. By attending to relevant spatial locations, these methods aim to improve the network's ability to incorporate global context while preserving local details. However, implementing conventional self-attention mechanisms often requires substantial GPU memory and incurs high computational costs, limiting their practical applicability, especially in resource-constrained environments.

However, ongoing research efforts continue to explore novel approaches to address the multi-scale object recognition problem in semantic segmentation. By innovating in architectural design, feature aggregation techniques, and attention mechanisms, researchers aim to develop more effective and efficient solutions capable of accurately segmenting objects of varying sizes in complex visual scenes. Achieving a balance between capturing fine-grained details and contextual information remains a key objective in advancing the state-of-the-art in semantic segmentation.

Reducing computational complexity without compromising performance is third crucial challenge in semantic segmentation, especially for applications where real-time processing and resource constraints are critical factors. Many existing methods rely on deep and wide CNN architectures, such as ResNet101 [66] and DenseNet [33], as backbone networks, which often have millions of parameters and high computational demands. Additionally, complex decoders are sometimes employed to recover the resolution of segmentation results, further increasing computational complexity. To deal with this challenge, several optimizations have been proposed in the literature. The key techniques are disussed below:

- Architectural Design Optimization: Researchers have explored novel architectural designs tailored for efficient semantic segmentation. These designs often involve carefully balancing the trade-off between model complexity and performance. For example, lightweight encoder-decoder architectures or depth-wise separable convolutions can achieve good segmentation accuracy while requiring fewer parameters and computations.

- Efficient Semantic Segmentation Architectures: Some recent approaches focus specifically on designing architectures optimized for low computational complexity. These architectures prioritize efficiency by leveraging lightweight building blocks, reducing redundant operations, and minimizing memory footprint. Examples include MobileNet [76], EfficientNet [77], and ENet [50], which demonstrate competitive performance with significantly lower computational requirements compared to traditional CNNs.

- Network Pruning and Compression: Techniques such as network pruning [78] and weight quantization [79] aim to reduce the number of parameters and operations in the model without significantly affecting performance. Pruning methods identify and remove redundant or less important connections or neurons, while compression techniques quantize weights to reduce precision and memory footprint. These approaches can lead to more lightweight models suitable for resource-constrained devices.

- Model Distillation: Model distillation [80, 81] involves training a smaller, more lightweight model to mimic the behavior of a larger, more complex model (teacher model). By transferring knowledge from the teacher model to the student model, distillation can produce compact networks with comparable performance to their larger counterparts. This approach enables the deployment of efficient segmentation models with reduced computational complexity.

- Hardware Acceleration: Hardware accelerators [82, 83], such as embedded GPUs, FPGAs , and ASICs can be used to speed up inference for semantic segmentation tasks. These accelerators are designed to efficiently execute neural network computations, enabling real-time processing even on resource-constrained devices.

By leveraging a combination of these techniques, researchers aim to develop efficient semantic segmentation models capable of running in real-time on resource-constrained devices, without sacrificing accuracy or requiring excessive computational resources. Achieving low latency and low memory footprint segmentation is essential for enabling intelligent systems to operate efficiently and effectively in real-world environments.

1.6 Problem Definition

Motivated by the challenges in the previous section, the objectives of the research work undertaken in this thesis can be defined as follows:

- Propose a custom loss function to improve the performance of the semantic segmentation around small objects and boundaries.
- Design a real-time and lightweight CNN with simple encoder-decoder architecture for semantic segmentation of road-scenes.
- Develop an efficient DNN with attention based basic building block for efficient feature learning for semantic segmentation.
- Develop a fast CNN for semantic segmentation with competitive accuracy achived through efficient incorporation of attention guided contextual features and minimise the computation requirements.

1.7 Contributions of the Work

To address the challenges and achieve the objectives mentioned in the previous sections, this dissertation introduces several deep neural networks and various computational modules for semantic segmentation. The key contributions are summarized below.

- The first contribution is the inverse-residual dilation pyramid network (IRDPNet), which proposes a lightweight encoder-decoder-based architecture without using any ImageNet backbone. It takes inspiration from the DABNet [46] and further improves the segmentation score and reduces the network size by redesigning the basic block. The basic IRDP module utilizes groups of depth-wise dilated convolutions with varying rates to form a spatial pyramid, effectively reducing computational complexity. The module can efficiently learn contextual features at multiple dilations within the block. The usage of the inverted residual structure with an expansion layer prevents information loss due to the dimensionality reduction of the feature space. The proposed network can achieve real-time performance with improved accuracy over the DABNet with less than half of the parameters.

- The second contribution is the block attention network (BANet). The spatial attention mechanism has been shown to effectively aggregate global contexts and capture long-range dependencies, making it advantageous for semantic segmentation. However, its implementation incurs significant computational costs, even on high-end GPU-based platforms. In BANet, a simplified attention method is designed and integrated in the basic block of the model. The block is repeated to build an efficient encoder-decoder network which can run in real-time on resource-constrained devices.

- The presence of large number of classes and class imbalance in the autonomous driving road-scene mak accurate labeling of pixels challenging around smaller classes and object boundaries. Inspired by the extensive use of custom deep learning training losses in bio-medical image segmentation, a composite loss function is proposed to mitigate the issues discussed above. The composite loss function has been formulated by combining distribution-based and boundary/region-based loss functions. The emperical results prove the efficacy of the proposed task-specific loss function.

- Another significant contribution is the context-guided multi-scale attention network (CGMANet), which builds on the extensive knowledge gained through the above contributions. A fast-hybrid module (FHM) is designed to enhance the joint learning of semantic and contextual features through stacked depth-wise dilated convolutions. The FHM further adds a downsampling kernel with a learnable convolution filter to increase the effective field of view. To further combine global contexts and long-range dependencies, a context-guided attention method is proposed. The unique idea behind this method is the use of low-level features, often rich in details like object boundaries and textures, to generate weights for high-level semantic features.

1.8 Thesis Organization

The rest of the thesis has been divided into five chapters based on the publications, followed by a concluding chapter presenting future directions.

Chapter 2 proposes a custom loss function for semantic segemntation of road-scenes. It explores the various boundary and region-based functions in context of autonomous driving. Based on the performance a composite function is proposed which improves the segmentation performace around smaller objects and boundaries.

In chapter 3, an efficient, lightweight CNN is proposed based on the state-of-the-art DAB-Net architecture. By redesigning the basic block with inverse-residual bottleneck and dilation pyramid, improved performance has been achieved with half the model size.

Chapter 4 leverages the self-attention mechanism in the basic block of the deep network to model the relevant features at every stage. A block-attention module is proposed with simplified attention added to a multi-branch structure to learn features at multiple dilations. A fast and lightweight decoder is also designed to fuse the encoded features and restore the resolution.

Chapter 5 further delves into the joint learning of semantic and contextual features by designing a fast-hybrid basic module. In addition, a context-guided multiscale attention module has also been proposed, which leverages the low-level feature details to selectively restore the feature map resolution in the decoder.

In chapter 6, a hierarchical approach for context integration in a multi-path deep neural network has been proposed. The network integrates features at multiple depth and resolution to obtain a rich representation of the semantically distinct objects.

Chapter 7 concludes the thesis and discusses the future direction of the work.

Chapter 2

Custom Loss Functions for Semantic Segmentation

Semantic segmentation of images for autonomous driving is a dense pixel-level prediction task. Present-day algorithms make extensive use of deep learning models for parsing the image. A large number of classes and class imbalance in the autonomous driving dataset make accurate labeling of pixels challenging. Cross-Entropy (CE) loss function is mainly used to optimize the network parameters during training. The overall performance gain resulting from CE loss comes from large object classes. In contrast, the segmentation accuracy of smaller classes and object boundaries remains inferior, even with increased overall accuracy. To address this limitation, task-specific custom loss functions have been developed and used for various applications like biomedical image processing. However, they have rarely been used for road scene segmentation in an autonomous driving context. To fill this gap, we develop a composite loss function by linearly combining the CE loss and boundary/region-based loss to improve the existing state-of-the-art model. We give empirical evidence that our custom loss function improves the segmentation scores on smaller object categories and object boundaries.

2.1 Introduction

A loss function is utilized in deep learning networks to evaluate the model's performance and update the learned parameters. It quantifies the difference between the desired output and the model output. Modern state-of-the-art computer vision algorithms employ deep

networks for various tasks such as object detection, segmentation, and classification. Semantic image segmentation is one such task requiring labeling each pixel. It is extensively applied in bio-medical image segmentation, road-scene parsing, robotics, and aerial image segmentation. Over recent years, many deep-learning algorithms have been developed for these tasks. The general trend to improve model accuracy is increasing its complexity and size. However, this comes at increased hardware costs. Small network size becomes essential for resource-constrained devices such as embedded systems and mobile devices. Researchers in [84] showed that significant accuracy gains could be achieved over existing deep learning object detection algorithms by employing a custom loss function while training. Following this work, several custom loss functions were developed for computer vision tasks such as image segmentation [85, 86, 87]. However, the primary focus has been bio-medical image segmentation. With the recent paradigm shift in transportation towards autonomous driving, more accurate predictions are required at the image level. Since both the applications of bio-medical image segmentation and road-scene parsing are fundamentally pixel-labeling tasks, algorithms developed for the former have been applied to road-scene segmentation [23, 53]. Consequently, the custom loss functions designed for bio-medical applications can be used in autonomous driving, albeit with modifications.

2D Cross-entropy(CE) loss [88] is majorly used to optimize segmentation algorithms in the literature. It is an easily differentiable and simple loss function without much computational overhead. Overall, it performs well for the bulk region of the objects and large classes present in the images. However, it could still perform better on object boundaries and smaller classes. Researchers in [89] have identified class imbalance as one of the primary reasons for low performance in smaller classes. While being region agnostic, CE loss fares poorly at boundary pixels. Some of these limitations have been addressed by researchers in the bio-medical domain [90, 91] through custom loss functions. This eventually inspires us to adopt a similar approach for our field.

In this chapter, we adapt different custom loss functions for road-scene parsing tasks. We extensively analyze different hyper-parameters and loss function combinations on the Cityscapes dataset. Experiments show that our proposed combinations of custom loss functions can improve the existing network's performance on small classes and object boundaries. We have used the ERFNet model [47] for its simplicity and sequential structure. In the following sections, we briefly discuss the related works, followed by an introduction to various custom loss functions, the proposed method, and experimental results.

2.2 Related Works

To improve the standard CE loss for road scene segmentation tasks, researchers have broadly followed three approaches:

1. Introduce a class-weighing factor in the CE loss function to address the class imbalance.
2. Use the auxiliary loss function in conjunction with CE loss to improve image boundaries.
3. Employ sampling techniques such as hard example mining to filter out easy examples before calculating CE.

These approaches are briefly discussed in this section.

1. Weighted CE loss based

As stated in the previous section, these loss functions add a class probability-based weight in the cross entropy factor. The aim is to give large weights to the smaller classes like traffic signals, traffic signs, poles, boundaries, and bicycles. The dominant classes, like roads, cars, footpaths, pedestrians, and buses, get smaller weights. The most intuitive method of assigning weights is calculating the inverse class probabilities. However, as the class probabilities tend to zero, the factor becomes unbounded. To overcome this, ENet [50] introduced a custom class weighing scheme in which the weights are bounded as probabilities tend to 0. Works like [51], [47], [73], and [54] have used similar weighing schemes with different hyper-parameters. RetinaNet [84] developed a dynamically scaled CE loss in which the scaling factor reduces as the classification accuracy increases.

2. Auxiliary loss based

In addition to the primary n-class CE loss, where n is the total number of object classes in the image, [92], [59], [54] have used auxiliary loss functions. While the main branch outputs are fed into the CE loss function, an additional layer is used for the auxiliary loss. CE is mainly used as an auxiliary loss in the networks which follow a multi-branch structure. In these networks, one branch is used for contextual feature learning, whereas the other is used for spatial details. Since the outputs from both branches have n-channels,

CE loss is used. For improving the segmentation accuracy on object boundaries, [92] has introduced the dice loss [90]. The edge details are represented as a binary image; hence the auxiliary output has a single channel. The ground truth image is passed through a simple edge detector filter such as Canny [15] before calculating loss.

3. Sampling technique based

Authors in [93] have observed that in classification problems if the negative samples are not filtered during training, the network will be biased towards lower confidence scores. Sampling techniques like bootstraping [94] and online hard example mining (OHEM) [93] are readily utilized for a sampling of the pixels. After these pixels are sampled according to some minimum criteria, they are given to standard loss functions. Detectron [95] proposed the probabilistic OHEM-based CE loss function, which provided excellent results on problems like object detection, classification, and segmentation.

2.3 Loss Functions

This section briefly discusses some custom loss functions used for image segmentation. Based on the derivation, the loss functions can be broadly classified into distribution-based, region-based, and boundary-based.

2.3.1 Distribution-based loss

These loss functions tend to measure the dissimilarity between two statistical distributions. The classic function of CE is the most basic yet efficient example. CE and its variants are highly utilized for training a semantic segmentation model.

a. Cross entropy

The cross-entropy loss measures the distance or dissimilarity between two distributions, say A and B. In the semantic segmentation task, A is the ground truth label, and B is the predicted label by the deep network. The CE loss is thus defined by

$$L_{CE} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{H\times W} gt_i^n \log y_i^n \tag{2.1}$$

where N is the total number of classes, gt_i^n is the ground-truth label binary indicator of i^{th} pixel of class n and y_i^n is the corresponding predicted probability. $H \times W$ is the dimension of the final predicted feature map or input image.

b. Weighted cross entropy

To prevent training bias due to class imbalance, ENet [50] introduced a weighting factor based on inverse class probabilities. The class probabilities are calculated apriori from the training subset. The custom class weight is given as

$$w_{class} = \frac{1}{\ln(c + p_{class})} \tag{2.2}$$

where p_{class} is the class probability and c is a hyperparameter added to bound the weights.

c. Focal loss

Focal loss only factors the hard examples by adding a dynamic scaling factor to the CE loss. [84] argues that in the absence of any mechanism to filter out the easy examples, the model is biased toward these. As a result, the hard examples are not appropriately classified. The dynamic scaling factor decays to zero with increased confidence in the correct class. The focal loss is defined by

$$L_{Focal} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{H\times W}(1 - y_i^n)^{\gamma} gt_i^n \log y_i^n \tag{2.3}$$

here $(1 - y_i^n)^{\gamma}$ is the added factor. As y_i^n is the predicted probability which tends to 1 as confidence increases, the factor tends to 0. γ is the hyperparameter with a value of 2 for optimal performance.

2.3.2 Region-based loss

Region-based loss functions measure the overlap or intersection between two regions. In the case of segmentation, the regions are obtained from the ground truth image and the predicted model output. Jaccard and Dice [90] are commonly used region-based losses discussed here.

a. Jaccard loss

Also known as Intersection-over-Union(IoU), Jackard loss [96] measures the overlap between the segmented regions in the ground truth and predicted output. As it directly factors the segmented object region, class imbalance does not affect the metric. It is given as

$$L_{IoU} = 1 - \frac{\sum_{n=1}^{N}\sum_{i=1}^{H\times W} gt_i^n y_i^n}{\sum_{n=1}^{N}\sum_{i=1}^{H\times W}(gt_i^n + y_i^n - gt_i^n y_i^n)} \tag{2.4}$$

b. Dice Loss

The Dice loss is based on the Sorensen-Dice coefficient [97]. It was first adapted in image segmentation for boundary delineation of 3D medical images [90]. It gives the ratio of the intersection of the ground truth and predicted segment. The dice loss is given by

$$L_{Dice} = 1 - \frac{2\sum_{n=1}^{N}\sum_{i=1}^{H\times W} gt_i^n y_i^n}{\sum_{n=1}^{N}\sum_{i=1}^{H\times W} gt_i^n + \sum_{n=1}^{N}\sum_{i=1}^{H\times W} y_i^n} \tag{2.5}$$

A simple extension of this dice loss with class weights was given by [98]. In which the weights were inversely proportional to the class frequencies in the ground-truth labels. This introduced class in-variance in the loss function. The class weights are given by

$$w_n = \frac{1}{\sum_{n=1}^{N} gt_i^n} \tag{2.6}$$

2.3.3 Boundary-based loss

It minimizes the distance between the boundaries of the ground truth segment and the predicted segment. We only discuss Hausdorff distance loss in this category.

Hausdorff Distance loss

This loss function is based on the Hausdorff distance (HD). It can be approximated by the distance transform for integration into CNNs [99]. The HD loss is given as

$$L_{HD} = \frac{1}{H\times W}\sum_{n=1}^{N}\sum_{i=1}^{H\times W}[(y_i^n - gt_i^n)(d_{gt_i^n}^2 + d_{y_i^n}^2)] \tag{2.7}$$

here $d_{gt_i^n}$ and $d_{y_i^n}$ are the distance transforms of ground truth and the predicted output, respectively. They give the maximum distance between any pixel in the segmented object and its boundary.

2.4 Proposed Method

As discussed in the previous section, it can be seen that losses from different categories focus on different object regions. Based on this, we model our composite function as a combination of CE loss with the region and boundary-based auxiliary losses. Following [100], an affine combination of the loss functions is used. The proposed composite loss function is of the form:

$$L_{Total} = -[(1-\alpha)\cdot\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{H\times W} gt_i^n \log y_i^n] + [\alpha \cdot L_{Aux\_loss}] \tag{2.8}$$

Here α is the coefficient such that, $\alpha : \mathbb{R} \in (0, 1)$. We tried different values for α (Figure 2.1) and empirically found the value of 0.4 optimal for our case. In the composite function (Equation 2.8), the first term is the CE loss which is responsible for the overall optimization of the network parameters. While the second term, which is the auxiliary loss, aids region-specific learning. Combining more than one auxiliary loss function with CE did not lead to any significant gains; hence we restrict it to one. Unlike CE loss, the region and boundary-based functions require binary class ground truths. In order to generate these, the ground-truth images are convolved with a simple canny edge detection kernel. This is done for every class separately and the resulting losses are added after normalizing. For HD loss, we only consider one object class at a time as foreground and ignore the rest of the classes as background. The losses are then added to obtain the final value from each training sample.

2.5 Experimental Results

Extensive experiments have been performed on the competitive Cityscapes [1] dataset. We select the state-of-the-art method, Efficient Residual Factorized ConvNet model (ERFNet) [47] for the ablation studies.

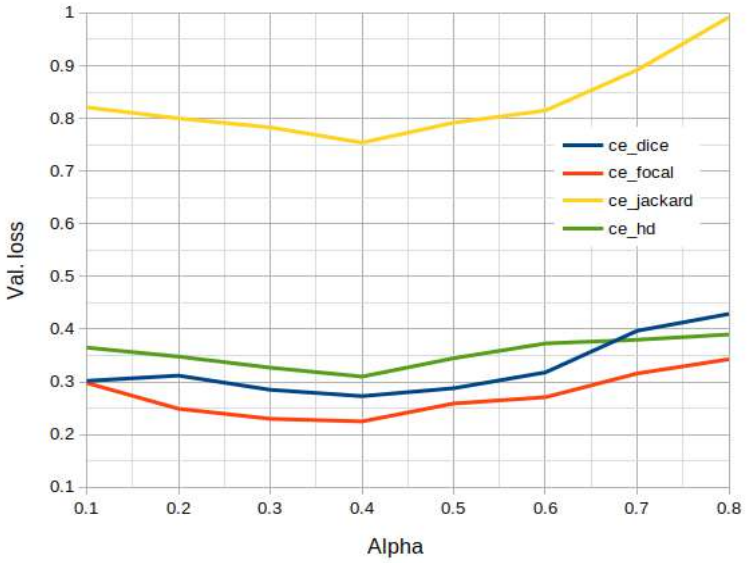


Figure 2.1: Validation loss for different values of Alpha. It can be seen that the composite losses are minimum for alpha = 0.4

2.5.1 Implementation Details

The model has been implemented in PyTorch v1.11 on Nvidia-V100 GPU running with Intel-Xeon-silver 4110 processor. Adam optimizer with an initial learning rate of 5×10^{-4} is utilized in all the experiments. A polynomial learning rate schedule with weight decay of 10^{-4} is used. As data augmentation strategies, random - flips, translations, and crops are selected. We flip the images both horizontally and vertically. For translations, we use a maximum shift of 2 pixels. The crop size is fixed at 512×512. The encoder output is used and bilinearly upsampled to the original resolution to reduce training time and make a fair comparison. All the models have been trained for 100 epochs.

2.5.2 Results and Discussions

The standard intersection-over-union(IoU) metric has been utilized for performance evaluation. The IoU scores for the model trained with different loss functions are given in Table 2.1. We have calculated IoU scores for only 19 classes used in training. It can be seen that using region and boundary-based losses, in addition to distribution-based loss functions, improves the segmentation score on small classes (mentioned in Table 2.2). There is a significant improvement in these classes (fencing, traffic lights, signals, person, and motorcycle). The classes (bus, train, and truck) have fewer samples in the training set, even though the size of each object in the sample is large. The composite loss functions

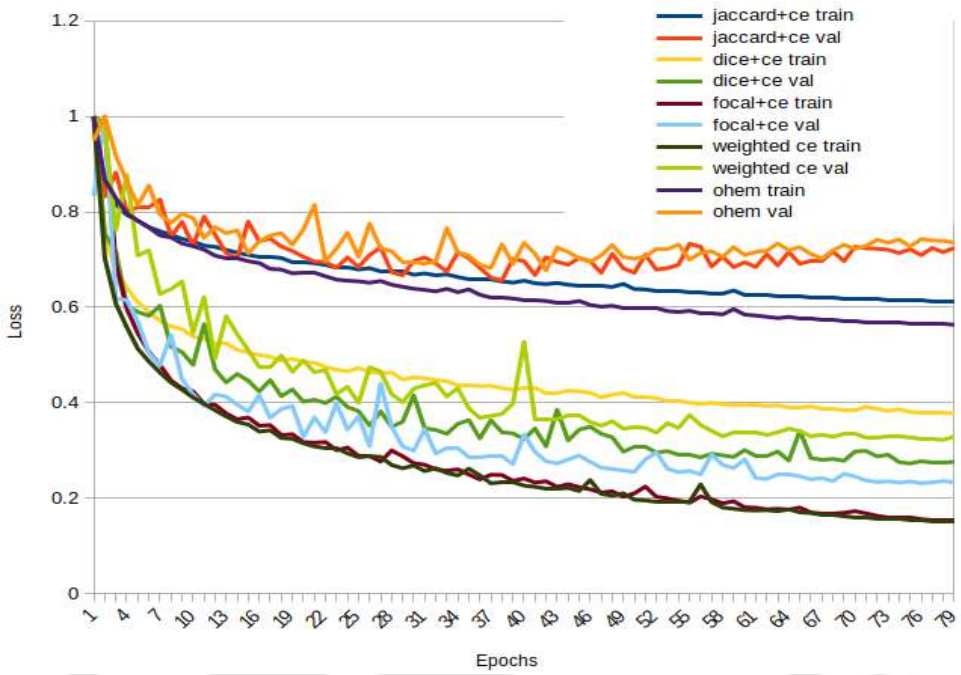


Figure 2.2: Training and Validation Loss (normalised) of first 80 epochs for the loss functions.

have also improved results on these classes (refer Table 2.1). We see that in all cases, using a multi-function loss is beneficial. However, the best score is obtained by using Dice and weighted CE loss functions together. While the Dice loss improves the region-specific performance on the object, weighted CE loss helps improve the segmentation score. To further prove the effectiveness of the proposed loss function, we train some state-of-the-art methods with the composite function. Table 2.2 shows improved results in smaller classes. The maximum gain of 26% is achieved for the motorcycle class in ERFNet. This is followed by a gain of 19.9% for the traffic light in ESPNet [51]. The minimum improvement is observed in CGNet [73] because it is already trained using an auxiliary loss function. However, the gain is still significant.

Figure 2.2 presents the training and validation loss plots for the functions discussed earlier in this section. We have used normalized values of training losses for plotting. The validation loss is calculated only using the weighted CE function for every experiment. The Focal+CE function has the least training and validation loss. Weighted CE follows it closely for training loss but has a higher validation loss. This can be attributed to inferior performance in smaller classes. The validation loss of Dice+CE closely follows the training loss. This implies that the model can better learn the abstract features from the image without undergoing overfitting. As a result, this combination of Dice and CE gives the

Figure 2.3: Qualitative examples of segmented images produced by the model trained with different loss functions; Column 1: Input Image, Column 2: Ground Truth, Column 3: CE loss(Baseline), Column 4: Dice+CE loss. The improvements in some of the small classes are highlighted in black squares.

best mIoU scores and significantly improves the IoU in smaller classes.

Figure 2.3 shows some qualitative examples of the segmented images. It can be seen that the segmentation results of the model trained with the Dice+CE function (Column 3) perform better than CE (Column 4) on smaller classes. The network has improved results on poles, traffic lights, and signals highlighted in black squares in the images. Some improvements can also be seen in large classes (sidewalks-pink color and road segmentation-purple color).

2.6 Conclusion

In this work, we adapted some of the custom image segmentation loss functions and applied them to the road-scene understanding task. The extensive ablation study shows that it is possible to improve the performance of the existing network on smaller classes by using compound loss functions. We have combined different categories of loss functions to enhance the feature learning capability of the network. The improvements have been achieved without structural changes to the network and limited training data.

Table 2.1: Class-wise IoU(%) results for the model trained with different loss functions on the Cityscapes validation subset. Classes(Left to Right): Roa-Road, Sid-SideWalk, Bui-Building, Wal-Wall, Fen-Fence, Pol-Pole, TLi-TrafficLight, TSi-TrafficSignal, Veg-Vegetation, Ter-Terrain, Sky, Per-Person, Rid-Rider, Car, Tru-Truck, Bus, Tra-Train, Mot-Motorcycle, Bic-Bicycle; Clas-Class(mean IoU)

Loss function	**IoU**																			
	Roa	**Sid**	**Bui**	**Wal**	**Fen**	**Pol**	**TLi**	**TSi**	**Veg**	**Ter**	**Sky**	**Per**	**Rid**	**Car**	**Tru**	**Bus**	**Tra**	**Mot**	**Bic**	**Clas**
W CE	96.7	76.6	88.4	**51.2**	44.9	47.1	49.9	60.6	89.3	**59.1**	91.5	65.9	44.5	89.9	57.0	66.7	56.2	33.5	62.1	64.8
Jaccard + CE	96.8	76.5	88.7	50.6	46.9	**48.8**	54.3	60.1	89.5	58.7	91.6	68.1	46.1	90.1	52.6	69.2	**58.2**	39.1	63.3	65.8
Dice + CE	**97.0**	**77.6**	**89.0**	50.3	**47.9**	48.6	**53.1**	**62.1**	**89.7**	**59.1**	91.5	**68.6**	**48.9**	**90.6**	62.2	**70.2**	52.1	**42.2**	**64.2**	**66.6**
Focal + CE	96.8	77.1	88.8	53.9	**47.9**	47.6	52.9	60.7	89.5	57.7	**91.8**	68.4	47.7	90.5	54.2	61.5	52.9	40.0	63.3	65.5
OHEM	96.6	75.7	88.5	50.4	44.4	46.8	52.1	60.4	89.2	59.0	91.7	66.4	46.6	89.8	**63.8**	69.9	54.4	39.2	61.8	65.6
HD + CE	96.1	77.2	87.9	50.7	46.5	48.2	53.0	61.9	89.1	58.3	91.5	67.8	47.2	90.4	56.5	67.4	55.1	**43.2**	62.9	65.9

Maximum IoU of each class are given in bold.

Table 2.2: Class-wise IoU(%) results on smaller classes for some models trained with composite loss function(CE+Dice) on cityscapes validation subset.'Bl' represents the baseline results for original loss function used in the method, 'Cl' represents composite loss function results. 'max. IoU inc.' represents the maximum IoU increase across the classes; the corresponding class is shown in bold.

Method	Fence		Pole		T.Light		T.Signal		Person		Rider		M.Cycle		B.Cycle		mIoU		max. IoU
	Bl	Cl	Bl	Cl	Bl	Cl	Bl	Cl	Bl	Cl	Bl	Cl	Bl	Cl	Bl	Cl	Bl	Cl	inc.(%)
ENet [50]	33.2	35.1	43.4	44.2	34.1	**37.6**	44.0	45.0	65.5	67.5	38.4	40.3	38.8	40.9	55.4	56.0	58.3	60.1	10.3
ERFNet* [47]	44.9	47.9	47.1	48.6	49.9	53.1	60.6	62.1	65.9	68.6	44.5	48.9	33.5	**42.2**	62.1	64.2	64.8	66.6	26.0
CGNet [73]	43.0	43.3	54.1	55.2	59.8	59.6	63.9	62.1	74.9	75.5	54.9	**61.2**	47.3	47.8	60.2	61.8	64.8	65.7	11.4
ESPNet [51]	36.1	38.7	45.0	46.4	35.6	**42.7**	46.3	48.2	67.0	67.9	40.9	43.6	41.8	42.3	57.2	57.1	60.3	61.8	19.9
LEDNet [101]	48.1	50.2	60.9	61.0	60.4	60.7	71.1	72.3	74.3	74.2	51.8	52.4	43.3	**49.7**	70.2	71.9	69.2	70.4	14.8

*Only Encoder output.

Chapter 3

Rethinking DABNet: Light-weight Network for Real-time Semantic Segmentation of Road Scenes

3.1 Introduction

The ever-increasing prosperity of deep learning algorithms has achieved above human-level performance in semantic segmentation applications. Recent model designs with the highest accuracy come with large and complex architectures. The high computational cost associated with such performance makes these algorithms unsuited for resource-constrained devices like smartphones, drones, robots, and intelligent vehicles with limited computational power. Further, the requirement for low latency in such devices makes the scenario more challenging.

The demand for autonomous driving in intelligent transportation systems has motivated researchers to design many lightweight real-time networks for road scene-parsing applications. These methods are primarily based on CNNs, given that modern algorithms like image transformers [63] are still very large for edge device deployment in constrained scenarios. ENet [50] is among the initial algorithms which exclusively target semantic segmentation in mobile devices. It factorizes the convolution kernels in a ResNet [66], an Imagenet backbone structure, to reduce the parameter count at the expense of accuracy loss. Adapting Imagenet backbones to semantic segmentation lacks task-specific design.

Therefore, designing a lightweight backbone from scratch with reduced depth is a common design choice for real-time and lightweight models. However, reducing the depth also decreases the field of view that is essentially required for learning contextual features in the semantic segmentation framework. This issue is resolved by using dilated convolutions in the backbone [47]. DABNet proposes an excellent algorithm using depth-wise factorization and dilated convolutions as functional elements. It proposes a two-branch basic block that jointly learns local and global contextual features. However, the DAB module can only learn global features at a single dilation rate, which may not be very effective in learning multi-scale object features. It uses residual bottleneck connections [76] to reduce the channel width within the module. This ultimately decreases the parameters for thin and deep network design. Nevertheless, the non-linearity in the residual bottleneck can destroy much information when channel width is reduced beyond the manifold of input feature space [42].

In this work, fast inference on resource-constrained devices has been targeted. Due to the availability of limited computational resources and low-latency requirements, these devices restrict the design space to simple network topology. To this end, a novel deep network is designed from scratch without any pre-trained Imagenet backbone. The basic block structure leverages a multi-rate dilation pyramid structure for deep feature aggregation and the use of inverted residual bottlenecks. Due to this inherent structure, it is called an inverse-residual and dilated pyramid (IRDP) block. The primary network has been built for real-time semantic segmentation by repeating the IRDP block without changing the DABNet structure, which can run on resource-constrained devices. The overall network structure of IRDPNet is shown in Figure 3.2. In addition, a lightweight variant, IRDPNet-L, has been proposed along with a full, encoder-decoder-based variant, IRDPNet-D, that utilizes a novel, ultra-lightweight decoder.

The remainder of the Chapter is arranged as follows: Section 3.2 introduces the proposed IRDP module and provides a detailed description of the network architecture. Section 3.3 shows the experimental results and the discussions about the effectiveness of the design choices. Followed by the conclusion in Section 3.5.

3.2 Proposed Method

The existing state-of-the-art methods for semantic scene parsing are primarily based on deep Imagenet backbones or sequence-to-sequence learning models like transformers [64,

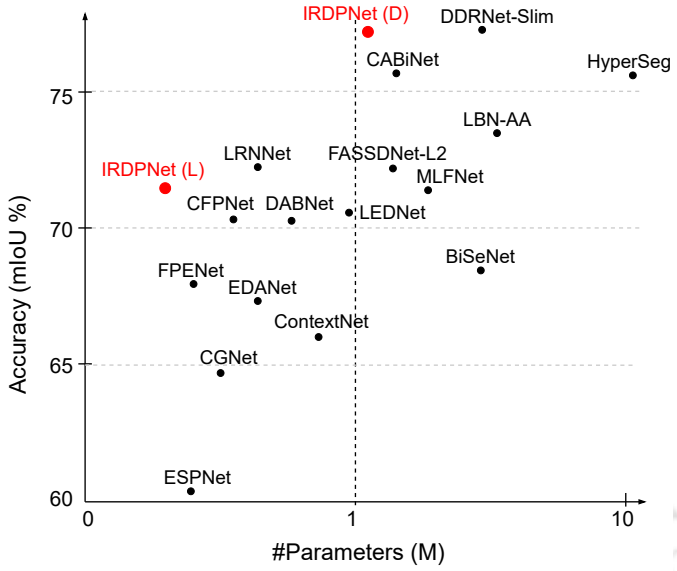


Figure 3.1: Size-accuracy comparison of our proposed method with lightweight and real-time state-of-the-art methods on the Cityscapes test set. The proposed method is highlighted in red. 'L': Lightweight variant, 'D': variant with decoder.

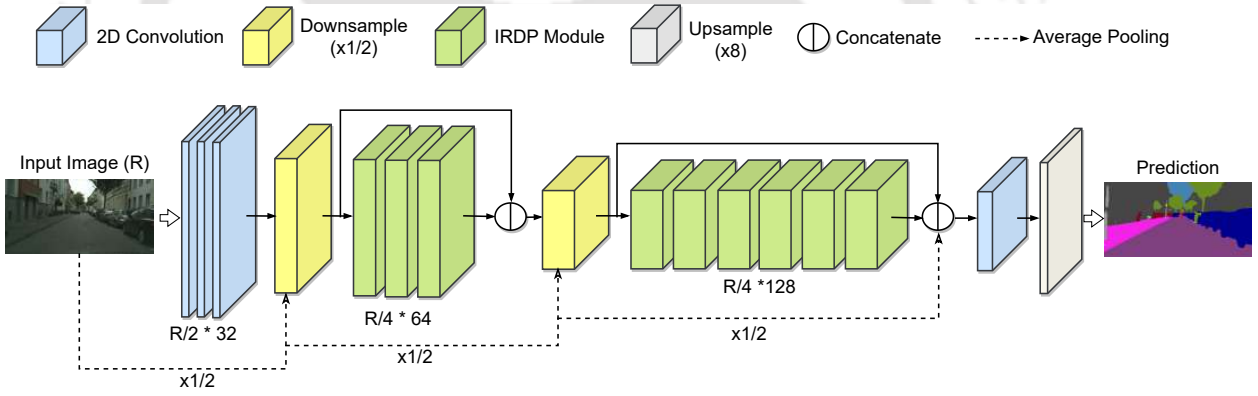
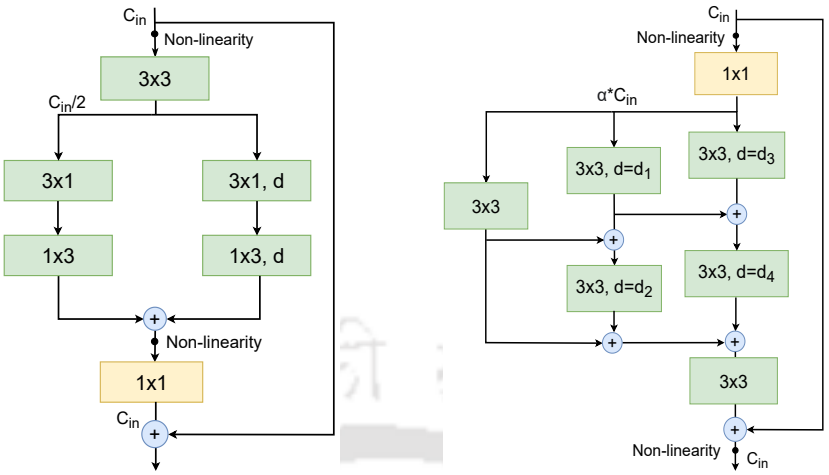


Figure 3.2: Architecture of the proposed Inverse Residual-Dilation Pyramid Network (IRDPNet). 'R' means input image resolution (H x W). 'R * 32/64/128' is the feature map dimension, where 32/64/128 is the number of channels.

63, 102]. The absence of high computing power and small memory on embedded devices limits the deployment of such methods. DABNet proposed a highly efficient network that balanced the size and accuracy trade-off. The proposed network is built on DABNet with carefully redesigned modules. This section details the IRDP module, which forms the fundamental building block of the proposed network, IRDP Net, discussed next. The IRDPNet is extended with a novel fast-lightweight decoder (FLD) to form IRDPNet-D. These modules are discussed subsequently.



(a) DAB Module **(b)** IRDP Module

Figure 3.3: Overview of the basic modules. 'C_{in}': Number of input channels. '3x1' is the convolution kernel size. 'd': Dilation rate. '+' denotes element-wise addition.

3.2.1 Inverse Residual Dilated Pyramid Module

While designing lightweight networks, conventional 3×3 convolution kernels are sparingly used to keep the parameter count low. Asymmetric, depth-wise separable, and point convolutions are extensively used instead [47, 48, 35]. However, these convolutions are approximations of the conventional kernels and marginally drop the network performance. We carefully design our basic block, IRDP module, around these convolutions to restrict the network size, albeit with some loss of accuracy. Inspired by inverted residual bottleneck connections in [42] and pyramid structure for multi-scale feature learning in ESPNet [51], the inverse residual dilated pyramid (IRDP) module is designed. In the following paragraphs, first, the design choices are elaborated. Followed by the overall module structure.

Inverse residual bottleneck structure: Bottleneck structures were proposed in [66] and used in many networks [51, 47, 76] to reduce the feature-space dimensions that ultimately result in parameter savings. However, using ReLU non-linearity and reducing channels in the bottlenecks leads to information loss due to mapping the input manifold of interest to a low-dimensional subspace. Researchers in [42] proved that if the channels are increased in the bottlenecks so that the input manifold lies in a low-dimension subspace of the input space, sufficient information can be preserved. Following this intuition, an inverted residual bottleneck structure is used in the IRDP module. We introduce an expansion factor, $'\alpha' \in \mathbb{Z} > 1$, used as a channel-width multiplier in the proposed module.

Dilation pyramid structure: We reiterate the need for multiple dilations for contextual learning of features at various scales in a semantic segmentation backbone. For this, a pyramid structure is used after the inverted bottleneck in the IRDP module. We have only used a three-branch pyramid structure to restrict the memory usage within the module (Figure 3.3b). The first branch does not use any dilation, thus focusing on learning local information around the centre of the kernel. Two convolution kernels are stacked in the second and third branches, each with different dilation rates. This increases the effective dilation rate of the branch. To further capture multiple-scale features, connections are added between two filter kernels from the previous branch. These branches are responsible for learning surrounding contextual features. The features from all three branches are added at the end.

IRDP module: The overall structure of the proposed module is given in Figure 3.3b alongside the DABModule (Figure 3.3a). The input is filtered with a linear 1×1 convolution kernel, increasing the number of channels by the expansion factor. One of the significant parameter savings in the module design comes from using pointwise convolution instead of conventional convolution for channel expansion. The dilation pyramid follows this, as explained in the previous paragraph. For effective feature fusion, the output of the pyramid is convolved with a 3×3 kernel. It also restores the number of intermediate channels to the input channels. For residual learning, we simply add the input feature maps with the output of the module. The purpose of this skip connection remains the same as in [66], which is better gradient flow during backpropagation. Given the number of input channels C_{in} and output channels C_{out}, the parameters for our IRDP module can be calculated as follows:

$$\begin{aligned} S_{param} &= C_{in} \times 1 \times 1 \times \alpha C_{in} + \alpha C_{in} \times \frac{\alpha C_{in}}{g_1} \times 3 \times 3 \\ &\quad \times 5 + \alpha C_{in} \times \frac{C_{out}}{g_2} \times 3 \times 3 \\ &= \alpha\, C_{in}^2 + 45\, \alpha^2\, \frac{C_{in}^2}{g_1} + 9\, \alpha\, C_{in}\, \frac{C_{out}}{g_2} \end{aligned} \tag{3.1}$$

where g_1 is the number of groups in the convolution kernels inside the pyramid structure and g_2 in the output $3{\times}3$ convolution; for $g_1 = \alpha\, C_{in}$ *and* $g_2 = C_{out}$, we have:

$$\begin{aligned} S_{param} &= \alpha\, C_{in} \left[C_{in} + 45\, \alpha\, \frac{C_{in}}{\alpha C_{in}} + 9\, \frac{C_{out}}{C_{out}} \right] \\ &= \alpha\, C_{in}\, [C_{in} + 54] \end{aligned} \tag{3.2}$$

As it can be seen from Equation 3.2, the number of parameters in the IRDP module depends mainly on the number of input channels

3.2.2 IRDP Net

The proposed network is effectively designed using the IRDP module as the fundamental building block, as presented in Figure 3.2. As can be seen in the figure, the network has four stages. At the same time, the initial stage contains three 3×3 convolution kernels, and the second and third stages are made using multiple IRDP modules. The fourth stage deals with the projection layer and bilinear upsampling. Following a shallow structure makes our network lightweight and fast, which are prime requirements for resource-constrained devices. Table 3.1 gives the detailed network design. The essential design features of the proposed network are mentioned in the following subsections.

Downsampling: Three downsampling operations are performed in the network, each reducing the input feature map resolution (Height $\times$ Width) by half. Some of the recent works [55, 61, 58] use five downsampling operations to reduce the feature map size up to 1/32. However, they are deeper and employ various context aggregation modules like pyramid pooling modules. To keep our architecture fast, we only go up to three stages. The downsampling operation serves two purposes: first, it reduces the number of FLOPs required in the following layers; second, it increases the field of view to learn the contextual information effectively in case of semantic segmentation. Both these functions are essential for the network design. Unlike works such as [47, 51, 61], we do not use a specific downsampler in the first stage. Rather, a 3×3 convolution with a stride of 2 is utilized. Using hardware-optimised standard convolution makes the initial block faster, which operates on the input image resolution.

Following the DABNet, we apply the ENet downsampling module in the second stage. It is a two-parallel branch structure, with maximum pooling kernel and 3×3 convolution with a stride of 2. However, we redesigned the module using asymmetric convolution kernels (Figure 3.4). This further reduces the number of parameters in our downsampler.

Image boundary features reinforcement: Long-range shortcut connections from the input image to downsampling modules are used in DABNet with an intuition of feature reuse. In these connections, the input image is global-average pooled and concatenated with the feature maps of the same resolution. However, it is found that doing so explicitly reinforces the image boundary features lost due to the subsequent low pass filtering by the

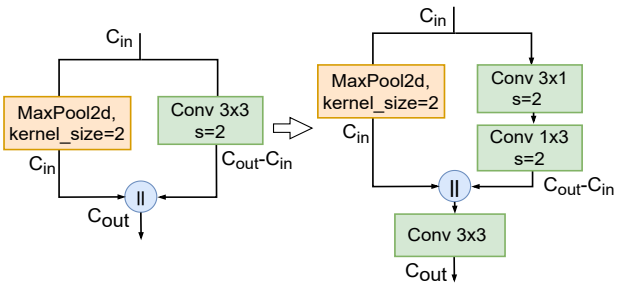


Figure 3.4: Downsampling modules. ENet downsampling module (left) and proposed downsampling module (right). They differ in the convolution kernels.

convolution operations. Input image reinforcement has been used in the second, third and final stages of the network.

Projection layer and Upsampling: In the final stage of the IRDPNet, a linear 1×1 convolution projection layer is used. The function of this layer is to project the higher-dimension manifold to a smaller dimension. This is accomplished by reducing the number of output channels to a value much less than the number of input channels. The feature maps from the third stage are projected to the number of classes in the segmented image, 19 in this case.

Table 3.1: Detailed structure of proposed IRDP Net

Layer	Operator	Mode	Channel	Output size
1	3x3 Conv	stride 2	32	384×768
2	3x3 (G) Conv	stride 1	32	384×768
3	3x3 (G) Conv	stride 1	32	384×768
4	Downsampling	stride 2	64	192×384
5	IRDP module	d=(1,2,1,2)	64	192×384
6-7	2×IRDP module	d=(2,4,8,16)	64	192×384
8	Downsampling	stride 2	128	96×192
9	IRDP module	d=(1,2,1,2)	128	96×192
10-14	5×IRDP module	d=(2,4,8,16)	128	96×192
15	Projection	-	19	96×192
16	Bilinear interpolation	×8	19	768×1536

(G) Conv- Group convolution; d=(d1,d2,d3,d4) is the dilation rate, refer Fig. 3.3b.

3.2.3 Fast-lightweight decoder (FLD)

In the literature, there are primarily two design possibilities for the decoder, one utilising deconvolution [47] and another using simple bilinear upsampling [55]. Deconvolution enhances representation capability owing to learnable weights; nevertheless, it reduces performance due to the needed kernel memory reorganisation. Bilinear upsampling, on the

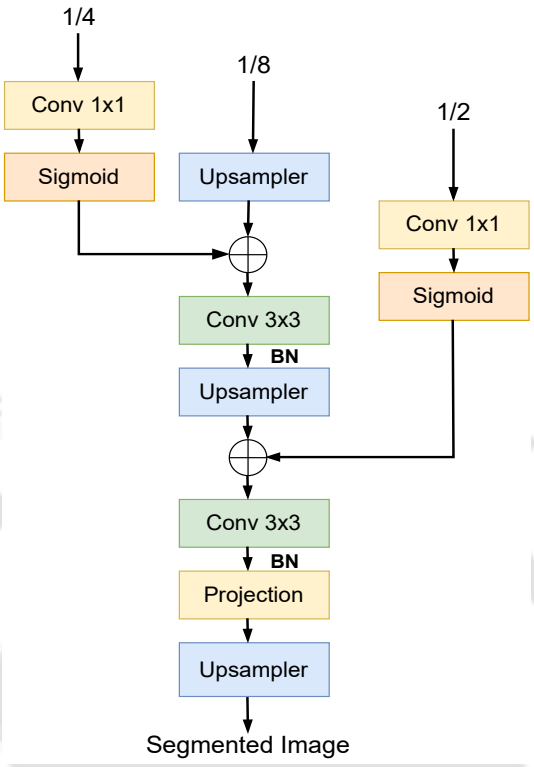


Figure 3.5: The fast and lightweight decoder (FLD). '+' is elementwise sum. 'BN' is batch normalization. '1/2' is the feature map size representing the skip connection from the first stage. Similarly, '1/4' and '1/8' are from the second and third stages.

contrary, is quick and doesn't use weights. To circumvent these constraints, a three-stage decoder (Figure 3.5) is created that leverages bilinear upsampling to boost feature map resolution. Standard 3×3 convolution layers are used rather than deconvolution layers to retain representation capacity.

Encoder skip connections are utilized in the decoder after [22]. The encoder features are merged with the decoded features after each upsampler. Before the addition, pointwise convolution is performed along with a sigmoid non-linearity; it serves two goals. First, it decreases the channel dimensions to match the number of features in the main branch. Second, it appropriately weights the attributes in order to highlight the most significant features. During backpropagation, these are learned automatically.

3.2.4 IRDPNet-D

The proposed network attached to the FLD forms the complete encoder-decoder-based design called IRDPNet-D, where 'D' signifies the decoder. Some of the information is lost

during the downsampling process in the encoder. Decoders are designed to compensate for this lost information while progressively increasing the size of feature maps. However, architectures like [46, 55, 73, 54] forgo the decoder for the sake of high-speed inference. To solve this problem, a fast and lightweight decoder has been designed. This leads to a significant improvement in the accuracy of the network with only minimal speed loss.

3.3 Experimental Results

This section evaluates the proposed network on three widely used datasets, Cityscapes, Camvid, and Berkeley deep drive, for benchmarking road-scene parsing tasks. To begin with, a brief introduction is given to the datasets, followed by the evaluation metrics. Then, an extensive ablation study is conducted to show the efficacy of the proposed network. To conclude, the method is compared with several state-of-the-art methods.

3.3.1 Evaluation Metrics

Accuracy measurement for the semantic image segmentation task is reported using the intersection-over-union (IoU) metric. It is calculated as,

$$IoU = \frac{target \cap prediction}{target \cup prediction} \tag{3.3}$$

Mathematically,

$$IoU = \sum_i \frac{n_{ii}}{t_i + \sum_j [n_{ji} - n_{ii}]} \tag{3.4}$$

where n_{ij} is the number of pixels of class-i predicted to class-j, and $t_i = \sum_j n_{ij}$ is the total number of pixels of class-i. The mean-class IoU (mIoU) is calculated by dividing the IoU score by the total number of classes. The number of parameters and FLOPs are used to measure model size and complexity. A model with very high FLOPs may be too computationally intensive for embedded devices. Speed is measured in terms of frames-per-second(fps).

3.3.2 Training settings

All the ablation studies have been performed on a single Nvidia RTX 3080 Ti graphics card system using PyTorch 1.10 framework and CUDA 11.6. Following [46], we use mini-batch stochastic gradient descent (SGD) and set the momentum and weight decay parameters as 0.9 and 10^{-4}, respectively. 8 for the Cityscapes and 16 for the Camvid dataset are used for batch size. 'poly' learning strategy is used in which the learning rate at each iteration is calculated as follows:

$$new\_lr = initial\_lr \times \left[\frac{current\_iteration}{total\_iteration}\right]^{0.9} \tag{3.5}$$

The initial learning rate is 0.045, and the model is trained for 300 epochs in the case of the Cityscapes and 500 epochs for the CamVid dataset using automatic mixed precision (torch.cuda.amp). We train Cityscapes at an image resolution of 768×1536, while for CamVid, 720×960 has been used. The data augmentation techniques for Cityscapes are random flips, random translation by a maximum of 2 pixels, random crop, and random scale in [0.75, 1.0, 1.25, 1.5, 1.75, 2.0], followed by colour jittering. For training on the Camvid dataset, the random scale is in [0.5, 2.5] on full resolution.

3.3.3 Ablation Study

In this section, extensive ablation studies have been performed to validate the performance of the design choices and modules.

IRDP module: DABNet module uses a two parallel branch structure (Figure 3.3a), we increase the branch to only three. This structure helps to increase the contextual feature learning ability with minimal speed impairment. Pyramidal structure basic blocks in [51, 52] use 4 to 5 parallel branches. However, it is found to increase the number of parameters and inference speed without any significant accuracy gain.

Intra-module skip connections: ESPNet introduced connections among the branches for hierarchical feature fusion (HFF). It solves the gridding problem due to dilated convolution kernels. Along with this solution, we found that skip connections improve the feature mixing at different scales. As a result, the overall accuracy of the model is improved, as seen in Table 3.2.

Position of pointwise convolution and non-linearity: We use pointwise convolution after the input connection for channel expansion in place of 3×3 convolution. This leads

Table 3.2: Ablation study for IRDP module on Cityscapes validation set.

Ablation Study	Type	mIOU (%)	FLOPs* (G)	Params (M)
Intra-module skip	No	70.1	5.12	0.32
	Yes	70.3	5.13	0.32
1×1 Conv position	Top	70.3	5.13	0.32
	End	68.7	17.86	1.35
Nonlinearity position	Before	69.2	5.13	0.32
	After	70.3	5.13	0.32

* FLOPs are reported for an input image of size 1024 × 512; 'M' - Million; 'G' - Giga.

Table 3.3: Ablation study for Downsampling module on Cityscapes validation set for IRDPNet-L.

Ablation Study	Type	mIOU (%)	Speed (fps)	Params (M)
Downsampler	ENet	70.1	81.63	0.37
	Proposed	70.3	81.23	0.32

to massive savings in FLOPs and parameters. In Table 3.2, our design choice reduces the parameter by 4× and FLOPs by 3× with a slight drop in accuracy. The position of non-linearity (PReLU) is shifted after residual addition, compensating for the absence of non-linearity after the final convolution layer as it follows inverse residual linear bottleneck design. It leads to a slight improvement in accuracy.

Downsampling module: As mentioned earlier, we factorize the 3×3 kernel in ENet downsampler into two asymmetric kernels (3×1, 1×3). Table **??** shows that our downsampler design reduces the number of parameters while improving the accuracy of the method. However, the non-optimised hardware kernels slightly reduce the inference speed.

Dilation rates: The proposed module jointly learns the global and local image features. For this, we use five convolution filters in three parallel branches inside the pyramid structure. The first branch has a single filter without any dilation. It learns the local image features. The second and third parallel branches employ dilated convolutions to extract the global context. We use four different dilation rates (d_1, d_2, d_3, d_4), set to (2, 4, 8, 16). To show the effectiveness, the dilation rates are changed as reported in Table 3.4. For the first ablation, all the rates are set to 1, which is effectively no dilation. This scheme performs the worst, showing the importance of dilation in the proposed model. Using coprime dilation rates gave better results in [103]. Following this, the rates are set to (3, 5, 7, 13). However, as reported by DABNet, coprime dilation rates did not benefit the proposed network. To further investigate the effect of larger dilation rates, we set it to (3,

Table 3.4: Ablation study for Dilation rates on Cityscapes validation set.

Dilation rates (d_1, d_2, d_3, d_4)	mIOU (%)
(1,1,1,1)	66.4
(2,4,8,16)	70.3
(3,5,7,13)	69.1
(3,6,12,18)	68.7
(8,10,12,14)	68.2

Table 3.5: Accuracy results for different versions of IRDPNet on Cityscapes test set.

Method	Params. (M)	mIOU (%)
IRDPNet	1.49	75.6
IRDPNet-L	0.32	71.3
IRDPNet-D	1.58	77.2

6, 12, 18) and (8, 10, 12, 14). This further reduces the accuracy, showing that the pyramid structure requires a balanced dilation rate between small and large.

Light version of IRDPNet: In addition to the proposed network, we introduce a light version, IRDPNet-L. This is specifically designed for model size and accuracy trade-offs. The lightweight version differs in two main aspects. First, we depthwise factorize the second and third convolution kernels in the initial block. This majorly reduces the FLOPs as the initial feature maps are large (R/2). Second, the convolutions inside the pyramid structure are also depthwise factorized, reducing the number of parameters. A pointwise convolution follows the depthwise factorized kernels. However, it reduces inference speed due to increased kernel calls, which motivates us to discard the pointwise convolution kernel. With these changes in place, we reduced the parameters from 1.49M to 0.32M. As shown in Table 3.5, in IRDPNet-L, the 4× reduction in size also reduces the accuracy by 5%. However, being an ultra-lightweight design, the achieved accuracy is still above several state-of-the-art lightweights.

Decoder: As mentioned earlier, several works have discarded the decoder for faster inference. However, to improve the accuracy of the backbone network, the FLD is attached to form IRDPNet-D. To further evaluate the effectiveness of the proposed decoder, the baseline network is trained with different decoders. Table 3.6 summarizes these results. RGPNet decoder [104] is designed for dense backbones like ResNet. However, it did not perform well with our backbone. Specifically, it reduced the accuracy by 0.2%. This is due to the backbone being shallower and lighter than the decoder. Consequently, the decoder does not get enough encoded information.

Table 3.6: Evaluation of IRDPNet with different decoders on Cityscapes validation set. Δp is the difference in the number of parameters compared to FLD. Bar represents a negative value.

Decoder	Params. (M)	Δp	mIOU (%)	FPS
RGPNet	2.00	1.91	75.3	34.2
LEDNet	1.30*	1.21	74.5	37.5
LBN-AA	1.2	1.11	75.4	42.4
FASSD-Net	0.95	0.86	78.1	52.0
ESPNet	0.015	$\overline{0.075}$	75.2	65.1
ERFNet	0.26	0.17	74.3	41.2
AGLNet	0.3	0.21	76.5	49.0
FLD (proposed)	0.09	0	77.5	57.6

FLD is the lightest, except for the ESPNet decoder, which uses deconvolution for upsampling. A similar method is used in the ERFNet decoder. Deconvolution is known for reducing the inference speed due to the large number of zero weights in the dilated kernel. As a result, deconvolution-based decoders did not perform well with our encoder. Specifically, the ERFNet reduced the inference speed by 36 fps, while the ESPNet did so by 13.5 fps. The FASSD-Net had the best accuracy of 78.1% but had an additional pyramidal pooling module, drastically reducing the inference speed. LEDNet and AGLNet share a similar decoder structure with an attention pyramid structure; the proposed encoder had a reduced accuracy and speed with both. Specifically, the accuracy was reduced by 2.7% and speed by half. We attribute this performance to a shallow network structure of IRDPNet.

3.3.4 Comparison on the Cityscapes benchmark

This section compares the proposed method with the state-of-the-art semantic segmentation methods on the Cityscapes benchmark. The results are reported in Table 3.7. The table is divided into three sections based on the number of parameters. The first section reports the accuracy-oriented algorithms, which are surprisingly large. In addition to their size, these methods have very high FLOPs and do not perform in real-time (>30 fps). ViT-Adapter-L [105] has the highest reported accuracy on the official bench-marking server.

Networks in the second and third sections directly compare with our three proposed models. While the second section reports algorithms in the range of 1-10 Million (M) parameters, the third section is dedicated to small networks of size <1 M. Our networks, IRDPNet and IRDPNet-D, belong to the second section with 1.49 and 1.58 M parameters, respectively. IRDPNet achieves better accuracy than all the methods of similar size. Specifically,

Table 3.7: Comparison with state-of-the-art on the Cityscapes benchmark. Categorized by decreasing network size from top to bottom.

Method	FLOPs (G)	InputSize	mIoU (%)	FPS	Param (M)
ViT-Adapter-L [105]	-	896 × 896	**84.9**	< 1	347.9
PSP Net [67]	453.6	1024 × 2048	78.4	< 1	65.6
Trans4trans [65]	94.25	768 × 1536	81.5	27.3	49.5
SegFormer-B5 [102]	1447.60	1024 × 1024	84.0	2.5	84.7
Dense ASPP [33]	214.7	1024 × 2048	80.6	< 1	28.6
Hyperseg-M [30]	7.5	512 × 1024	75.8	36.9	10.1
LBN-AA [35]	49.5	448 × 896	73.6	51	6.2
DDRNet-Slim [59]	35.78	1024 × 2048	77.4	108.8	5.7
BiSeNet [55]	14.8	768 × 1536	68.4	105.8	5.8
CABiNet [58]	12	1024 × 2048	75.9	76.5	2.64
FASSD-Net-L2 [61]	45.1	1024 × 2048	72.1	133.1	2.3
ERFNet [47]	21.0	512 × 1024	68.0	41	2.1
ESNet [48]	27.49	512 × 1024	69.1	63	1.66
IRDPNet-D (proposed)	20.31	768 × 1536	77.2	57.6	1.58
MLFNet-MobileV2 [57]	4.67	512 × 1024	71.5	90.8	3.99
AGLNet [75]	13.88	512 × 1024	70.1	52	1.12
ESPNetv2 [52]	2.7	512 × 1024	66.2	67.0	1.25
IRDPNet (proposed)	11.54	768 × 1536	75.6	78.6	1.49
LEDNet [101]	11.44	512 × 1024	70.6	71.0	0.94
ContextNet [54]	6.74	1024 × 2048	66.1	41.9	0.85
DABNet [46]	10.60	1024 × 2048	70.1	27.7	0.76
EDANet [106]	11.34	512 × 1024	67.3	108.7	0.68
LRNNet [107]	8.48	512 × 1024	72.2	71.0	0.68
CFPNet [108]	14.7	1024 × 2048	70.1	30	0.55
CGNet [73]	7.14	512 × 1024	64.8	50	0.5
FPENet [109]	5.7	512 × 1024	68.0	102.0	0.40
ENet [50]	3.8	640 × 360	58.3	52	0.39
ESPNet [51]	4.5	512 × 1024	60.3	112	0.39
IRDPNet-L (proposed)	5.13	768 × 1536	71.3	81.3	0.32

"-" indicates that the method does not report the result.

it has more than 6% mIOU over ESNet [48] while having significantly lower FLOPs. It outperforms ESPNetv2 [52] by more than 10% in accuracy. DDRNet-Slim [59] achieves the highest accuracy in this category. However, the input image resolution is 1.7× larger than ours with 60% more FLOPs. IRDPNet has better accuracy than AGLNet [75] by 5%. Its accuracy and FLOPs are comparable to CABiNet while having 70% fewer parameters. With less than half FLOPs and 60% fewer parameters, the proposed method outperforms FASSD-Net-L2 [61] mIOU by 4%.

The lightweight version, IRDPNet-L, achieves the highest accuracy for the given size among the state-of-the-art. It surpasses DABNet accuracy by 1.2% with less than half of the parameters and FLOPs. Its accuracy is comparable to LEDNet [101] with 1/3rd the parameters and 50% fewer FLOPs. EDANet [106] achieves the highest fps of 108.7 fps, which is 27.5 fps more than ours. FPENet [109] gets an fps of 102. However, both these methods use a 2.25× smaller input resolution.

Figure 3.6: Qualitative results of the proposed networks and DABNet on the Cityscapes validation set. First row: Input image; second row: ground truth; third row: DABNet; fourth row: IRDPNet; fifth row: IRDPNet-L; sixth row: IRDPNet-L + FLD. The critical regions have been highlighted with the box, white being the improved regions and cyan being the failure region.

The qualitative results of the proposed networks and DABNet on the Cityscapes validation set are shown in Figure 3.6. Note that the DABNet has been trained from scratch on 768×1536 image resolution for a fair comparison using open-source code. The critical regions are highlighted in boxes. All of our networks show an improvement over DABNet in these regions. The best output is obtained by IRDPNet, shown in the fourth row. The sixth row presents the output with our FLD, showing significant improvements over only the encoder backbone (IRDPNet-L), shown in the fifth row. Most failure cases are observed for smaller classes, specifically poles and traffic signals.

3.3.5 Comparison on the CamVid benchmark

The proposed method has also been compared with state-of-the-art networks trained on the CamVid dataset in Table 3.8. Our network has 67.8% accuracy with only 0.32M

Table 3.8: Comparison with state-of-the-art on the CamVid benchmark.

Method	InputSize	mIoU (%)	FPS	Params (M)
RTFormer-Slim [110]	960 × 720	81.4	190.7	4.8
EDANet [106]	480 × 360	66.4	40.75	0.7
DFANet [24]	960 × 720	59.3	116	7.8
SwiftNet-MN [68]	960 × 720	65.0	27.7	11.8
ICNet [53]	960 × 720	67.1	46.7	7.8
BiSeNet [55]	960 × 720	65.6	175	5.8
BiSeNetV2-L [56]	960 × 720	73.2	32.7	-
MLFNet-Res34 [57]	960 × 720	69.0	57.2	4.0
LAANet [74]	480 × 360	67.9	112.5	0.7
CGNet [73]	960 × 720	65.6	59.0	0.5
FASSDNet [61]	960 × 720	69.3	80.0	2.85
DABNet [46]	480 × 360	66.4	101.3*	0.76
IRDPNet-L	960 × 720	67.8	95.9	0.32

*Denoted that the specific metric is calculated by us using the open-source code. "-" indicates that the method does not report the result.

parameters while running at 95.9 fps. Following the results of the Cityscapes benchmark, the proposed method also has better accuracy than DABNet on CamVid. Specifically, IRDPNet-L archives 1.4% more accuracy than DABNet with 58% fewer parameters. The accuracy is comparable to ICNet but with 20× fewer parameters. IRDPNet-L outperforms BiSeNet by more than 2%. However, since kernel decomposition has been used to minimise parameters, the network is slower than BiSeNet. With half the parameters, IRDPNet-L has better accuracy than EDANet [106]. FASSDNet outperforms IRDPNet-L in terms of accuracy with 9× more parameters. LAANet [74] is among the fastest but works at half the resolution of the proposed network. These results show that the proposed method can perform on par with state-of-the-art methods while being lightweight.

3.3.6 Evaluation on GPU-Powered Embedded Systems

The embedded system GPUs are designed to work in resource-constrained devices. They are underclocked and have fewer unlocked CUDA cores. Consequently, a lightweight deep network should perform in real-time on these devices. We evaluate the performance of the proposed network on two such devices: Jetson Xavier NX and RTX 3060-Mobile laptop. Table 3.9 presents the technical specifications of these GPUs. We use the official SDK for the Jetson, which has TensorRT for acceleration. However, for a fair comparison on 3060M, we do not use any such optimization. In Table 3.10, the result of IRDPNet-L for two different input resolutions is presented. The proposed model runs faster than most methods on both devices for an input image of size 1024 × 512. The reason can be attributed to the lightweight design and one of the lowest memory usage (1357.5 MB)

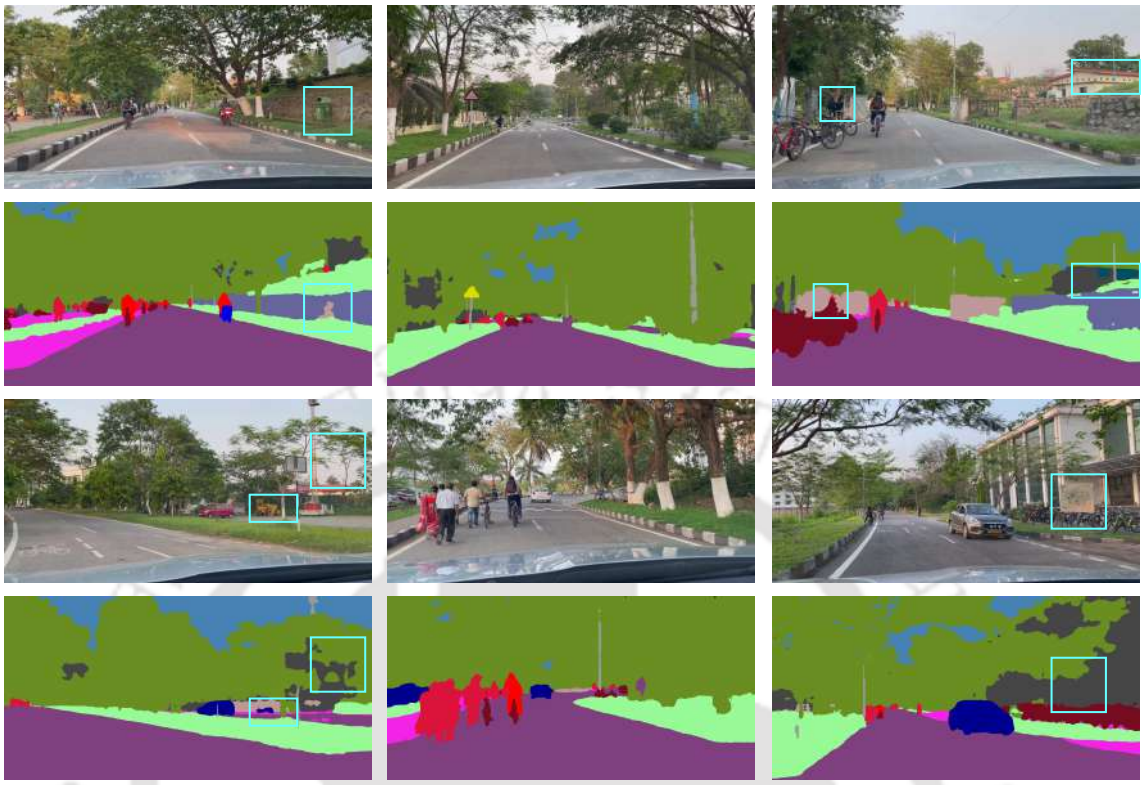


Figure 3.7: Qualitative results of the proposed network (IRDPNet-D) on unseen road images. The critical regions of failure have been highlighted with the box in cyan colour.

Table 3.9: Specifications of the mobile-GPU cards used for inference.

GPU Card	Cores	Speed	RAM	Mem.B/W	Power
Xavier	384	854MHz	8GB	51.2 GBps	15W
3060M	3840	817MHz	6GB	336 GBps	80W

'RAM': Random Access Memory; 'Mem.B/W': Memory Bandwidth.

during one forward inference pass. However, ENet outperforms all the considered methods regarding speed, memory and consumed power because of its shallow and small network structure, albeit having low accuracy. Despite being small, LEDNet is the slowest due to its complex decoder structure. In addition, it uses the maximum memory (3120.8 MB) during inference. Fast-SCNN and ContextNet outperform the proposed method in speed while having a low accuracy score and similar memory(1246.3, 1463.6 MB) and power consumption (5.8, 6.2W). MLFNet has better accuracy and comparable speed but has 12× more parameters and consumes 26% more power (7.6 W).

An important conclusion can be drawn from the results in Table 3.10, that is, all the given networks are unable to run in real-time (> 30 fps) on full-resolution images of the Cityscapes. However, an input resolution of 640×320 is enough to satisfactorily recognize

Table 3.10: Inference speed (fps) of the proposed method for different input image resolutions on mobile GPU-based systems. Full resolution: 2048 × 1024; half resolution: 1024 × 512.

Method	Xavier		3060-M		mIoU	Parm	Mem	Pow
	Full	Half	Full	Half	%	(M)	(MB)	(W)
SwiftNet [68]	2.61	9.9	3.91	14.5	70.2	11.8	1721.3	8.2
ENet [50]	13.8	53.82	20.17	78.68	58.3	0.4	940.1	4.5
ContextNet [54]	10.49	40.91	15.33	55.2	66.1	0.85	1463.6	6.2
LEDNet [101]	0.7	2.73	1.21	4.43	70.6	0.94	3120.8	8.7
BiSeNet [55]	2.42	9.6	3.45	13.77	68.4	5.8	1902.7	7.4
Fast-SCNN [45]	11.49	40.21	14.1	53.6	68.4	1.14	1246.3	5.8
CaBiNet [58]	8.21	30.95	10.45	41.8	76.5	2.64	1279.0	6.5
FASSDNet [61]	7.3	29.2	10.79	38.85	76.0	2.85	2401.7	7.9
MLFNet [57]	8.41	32.79	11.2	43.7	71.5	4.0	2170.4	7.6
IRDPNet-L	9.7	36.86	12.94	49.18	71.3	0.32	1357.5	6.0

'Mem' denotes GPU memory used during one forward inference pass. 'Pow' denotes consumed power during the execution phase on the Xavier module @1100MHz.

any urban street scene [47]. Consequently, for half resolution (1024 × 512), the proposed method achieves above real-time performance on both the GPU cards.

Figure 3.7 shows the visualisation results on unseen road images. The image frames are from video captured in a car with a dash-mounted mobile device inside our institute campus. The video is captured at a resolution of 1920 × 1080, and the input images are resized to 768×1536. It can be seen that our model performs satisfactorily on large objects, even on unseen images. The critical regions of failure have been highlighted in the box. In the first image (left-to-right, raster scan), the dustbin is segmented as fencing. In the third image, a pedestrian is identified as a rider due to its proximity to the cycles. The fourth image has a road roller, which is almost missed. This can be due to the unavailability of the object in the training dataset. In the sixth image, the campus map is identified as a building for similar reasons.

3.4 Contributions

The major contribution of this chapter can be summarised below:

- An inverted residual type dilated pyramid-based basic block has been proposed as shown in Figure 3.3. It is referred to as the IRDP module. It extracts local and global contextual features with a multi-branch pyramid structure. It has a scalable receptive field for deep feature extraction at the block level.

- The proposed network, IRDPNet, is designed based on the IRDP module. It carefully balances the performance-model size trade-off and achieves comparable results with large, state-of-the-art networks.
- In addition, a lightweight decoder has also been designed to improve the network's accuracy further. It can restore certain spatial features lost during the downsampling process.
- Without any Imagenet pre-training, post-refinement, or additional context branch, the proposed method can achieve competitive results on the Cityscapes dataset. Through extensive experiment results, it is shown that the proposed network (IRDP-L) can outperform the DABNet with less than half the number of parameters (**0.32M**). The full-scale version of the network (IRDPNet-D) can achieve **77.2% mIoU** with only 1.58M parameters, beating several state-of-the-art methods of comparable size while working in real-time.

3.5 Summary

The proposed algorithm optimised the classical DABNet model for size and accuracy. The basic block has been redesigned using inverse bottleneck residual and dilation pyramid structures. The inverse residual structure helps in preserving information in the bottleneck. At the same time, the pyramid structure increases the effective receptive field without a significant increase in computation. We designed three variants of our network, which vary in model size. The lightweight version outperforms DABNet in terms of accuracy with less than 50% parameters. This network can run in real-time on resource-constrained embedded devices. To further increase the accuracy of the baseline network, FLD has been designed. This network, IRDPNet-D, reaches the state-of-the-art mIoU score with a fraction of the parameters of similar methods. With the results on a highly competitive benchmark dataset - Cityscapes and CamVid, we show the performance of the networks.

Chapter 4

Block Attention Network

4.1 Introduction

In recent years, spatial attention mechanism has emerged as a powerful tool to enhance the segmentation of images by focusing on relevant spatial features [37, 55, 56]. In context of road-scene, the integration of attention mechanism in the network allows network to give more weight to regions that are likely to contain critical objects such as roads, vehicles, pedestrians, and traffic signs. This selective focus emulates human perception, where attention is directed toward areas of interest, filtering out irrelevant background information. In the previous chapter we have proposed a simple encoder-decoder based lightweight network without any additional mechanism to model pixel relations and capture dependencies.

In this chapter, a new design for a lightweight encoder-decoder network incorporating the attention mechanism is proposed. The network performs on par with the large real-time semantic segmentation networks while keeping the number of parameters small. The encoder is built on a novel attention-guided asymmetric basic block. It has a two-branch structure to simultaneously learn local and global features essential for the semantic segmentation of images. It utilizes group convolution in the main branch to reduce the parameters without compromising representation capability too much. The first parallel branch factorizes the 3×3 convolutions with an asymmetric 3×1 kernel, followed by a 1×3 kernel. We use dilation in these asymmetric convolutions to expand the field of view. The second parallel branch uses dilated kernels, which help learn the local features. Inspired by ResNet [66], a skip connection is added to the basic module. It improves gradient flow during backpropagation. To this skip connection, an attention module is attached, leading to the weighted

Table 4.1: Detailed structure of the proposed Block Attention Network.

Block	Layer	Module	(S,D)†	Ch.	Output size
Encoder	1	Down	(2,1)	32	512 × 256
	2	BA	(1,2)	32	512 × 256
	3	BA	(1,2)	32	512 × 256
	4	Down1D	(2,1)	64	256 × 128
	5	BA	(1,2)	64	256 × 128
	6	BA	(1,2)	64	256 × 128
	7	BA	(1,2)	64	256 × 128
	8	Down1D	(2,1)	128	128 × 64
	9	BA	(1,2)	128	128 × 64
	10	BA	(1,2)	128	128 × 64
	11	BA	(1,2)	128	128 × 64
	12	BA	(1,2)	128	128 × 64
Decoder	13	Bi-Int*	×2	128	256 × 128
	14	Conv3x3	(1,1)	64	256 × 128
	15	Bi-Int	×2	64	512 × 256
	16	Conv3x3	(1,1)	20	512 × 256
	14	Bi-Int	×2	20	1024 × 512

† S=Stride, D=Dilation; 'Ch.'- Channel
* Bi-Int=Bilinear Interpolation; '×2' denotes the interpolation factor.

addition of the feature maps from the previous block to the present one. It will be shown that adding this attention-skip connection reduces the dilation rates to only 2. This leads to a significant speed-up of the network.

4.2 Proposed Method

BANet follows an encoder-decoder structure, details of which are given in Table 4.1. This section introduces the basic block and other modules, followed by the complete network structure.

4.2.1 Attention Refinement Module

By assigning varying weights to picture pixels, the attention mechanism can selectively emphasise meaningful features while ignoring inconsequential information. Some recent publications [55, 56] have focused on applying attention mechanisms to semantic segmentation to boost the network's feature learning ability and object segmentation accuracy. An improved attention refinement module (ARM) based on [55] is proposed as shown in Figure 4.1.

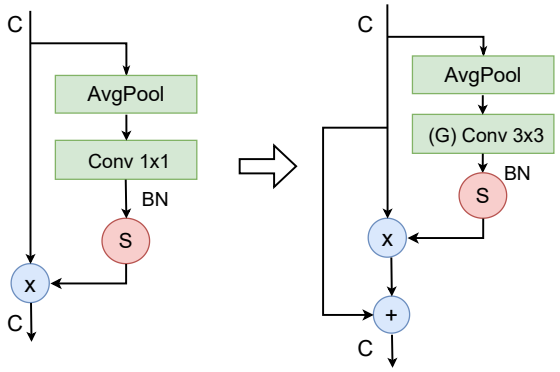


Figure 4.1: Attention Refinement Module. Left: BiSeNet module; right: BANet module. Where 'S': Sigmoid non-linearity; 'AvgPool':global average pooling; 'x':element-wise multiplication; '+': element-wise addition; 'BN': batch-normalization and 'C':number of channels.

The ARM is improved for our application by replacing the 1×1 pointwise convolution with a 3×3 convolution. First, we require a larger receptive field because our attention module operates in initial as well as deeper layers. Second, it must learn contextual features for better semantics because it is used in every block. Our module also differs from the ARM in how it combines weighted features. Directly multiplying the input features with the attention weights leads to the loss of certain features due to zero weights. This problem is solved with an extra connection that adds the input features to the weighted features, as shown in Figure 4.1. In summary, the ARM's weighted features are calculated as,

$$X_{attn} = \sigma\{f_{BN}[W_0(f_{avg}(X_{in}))]\} * X_{in} \tag{4.1}$$

where X_{in} is the set of feature maps at the input of the ARM, f_{avg} is the global average pooling function, W_0 is the convolution kernel weight, f_{BN} is batch normalization and σ is the sigmoid function. In contrast, the weighted features in our improved ARM are calculated as,

$$X_{attn} = \{\sigma\{f_{BN}[W_0(f_{avg}(X_{in}))]\} + 1\} * X_{in} \tag{4.2}$$

4.2.2 Block-attention module

The basic block of our proposed network, referred to as the block-attention module (BA), is shown in Figure 4.2. The efficient bottleneck design is used [66] as it reduces the number of channels before performing convolution and non-linear operations, thus reducing the computations. The BA module jointly learns the local and global contextual features,

quintessential for accurate semantic pixel labelling [[46], [73], [101], [47]] using a two-branch structure. Local features are extracted in the first branch using a 3×3 depth-wise separable convolution without dilation. This is followed by a point-wise fusion operation performed by 1×1 convolution. These design choices drastically reduce the parameter count, reducing the computation and memory requirements. The module uses channel shuffle operation [111] to improve the accuracy.

The second branch is designed to extract global contextual features. Asymmetric convolution is used to minimize the parameter count in which a $n \times n$ 2D convolution kernel is approximated by $n \times 1$ and $1 \times n$ kernels. For a 3×3 convolution, this essentially means a 33% reduction in the number of parameters from 9 to 6 per channel. Dilation is used in the asymmetric kernels to expand the receptive field and gain global context. Batch normalisation and ReLU non-linearity are used after every convolution layer. The feature learning is further enhanced by using a 3×3 convolution in the input branch followed by 1×1 convolutions.

A skip connection is used to prevent the problem of vanishing gradients [66] during back-propagation in the BA module. Nevertheless, unlike other networks, an attention-guided skip connection is introduced. This design technique reduces the dilation rates to only 2, thus improving network speed.

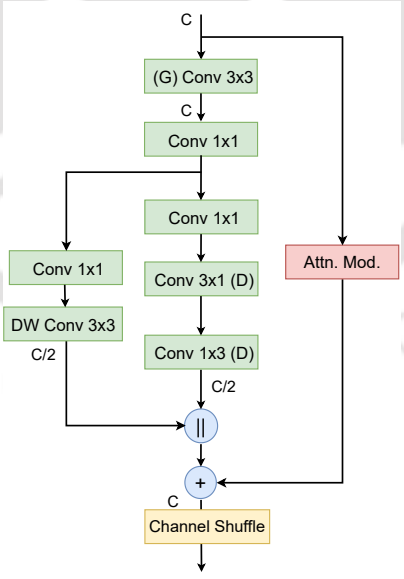


Figure 4.2: An overview of Block-Attention module. 'C': Number of channels. '(G) Conv': Groupwise convolution. 'DW Conv': Depth wise convolution. 'D': Dilated convolution. '||': concatenation; '+': element wise addition. 'Attn. Mod.' is our improved attention refinement module.

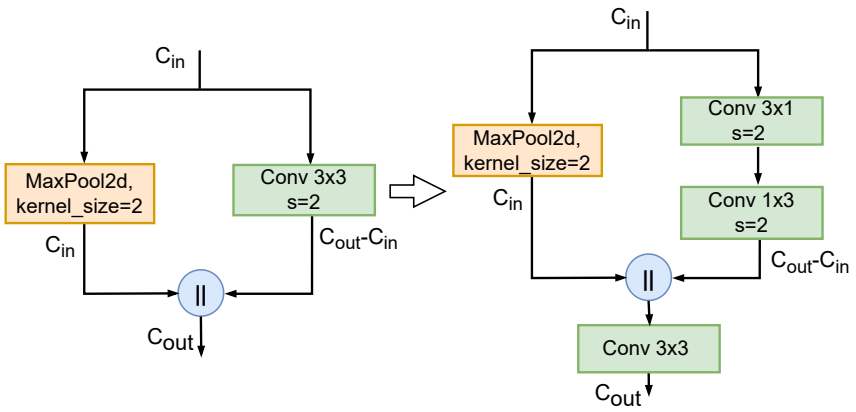


Figure 4.3: Downsampling Module: Left - ENet; right - BANet. 'MaxPool2d' is the 2D maximum pooling kernel along the height and width dimensions; 'C' represents the number of channels; 's' is the convolution stride.

4.2.3 Downsampler

To reduce the spatial resolution of the feature maps and learn scale-invariant features, it is common practice to downsample at each stage in the network. Techniques include max-pooling [112], average pooling [113], and strided convolution. ENet showed that using a pooling operation and strided-convolution in parallel reduces the loss of finer details while downsampling. A similar downsampling module shown in Figure 4.3 is designed with 1D asymmetric convolutions to reduce weights further. This makes our downsampling module extremely lightweight and hence suitable for low-memory devices. A convolution block is added after the concatenation to fuse the features efficiently.

4.2.4 Efficient decoder design

For the decoder, the literature has mainly followed two design choices to increase feature map resolution, deconvolution [47] and simple bilinear upsampling [46]. Deconvolution improves the representation capability due to the learnable weights; however, kernel memory restructuring lowers its speed. By contrast, bilinear upsampling does not require any weights, which makes it fast, but its representation capability is poor. To overcome these limitations, an efficient 3-stage decoder (Figure 4.4) that combines bilinear upsampling and conventional 3×3 convolution layers instead of deconvolution layers to maintain the representation capability is designed.

Following [22], encoder skip connections are used in the decoder. After each upsampler, we fuse the encoder features with the decoded features. However, it is found that simply

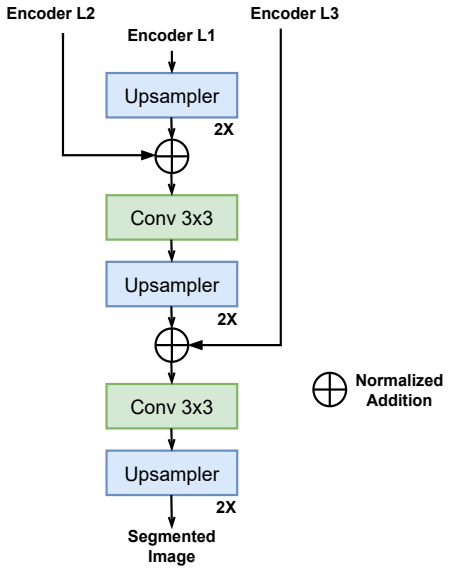


Figure 4.4: Light weight decoder. It is a three-level decoder in which Encoder L1, L2 and L3 activations are used. While L1 features are directly upsampled. L2 and L3 features are added after normalization in the decoder stages. '2X' represents the upsampling factor.

adding features from different layers does not lead to optimal performance because they have activations of varying scales. The features are normalized using the L_2-norm before addition to solve this problem. It is referred to as normalized addition.

4.2.5 Block Attention Network

Figure 4.5 shows the complete network structure. The encoder consists of repetitive block attention modules with downsampling blocks interleaved at appropriate places. The first block of our network is the downsampler which reduces the input resolution by half. The first stage consists of only two BA modules to minimize FLOPs due to a higher-resolution input feature map. We refrain from aggressively downsampling feature maps as in [47] because it lowers the accuracy. The second stage consists of three BA modules after the downsampler. This stage operates at $1/4^{th}$ the input resolution. In the encoder's final stage, we use four BA modules. Thus, we gradually increase the depth of each stage. The final resolution of the encoder output is $1/8^{th}$ the input resolution. The number of channels is restricted to the minimum possible in each stage to reduce layer width. This reduces the memory requirement [73].

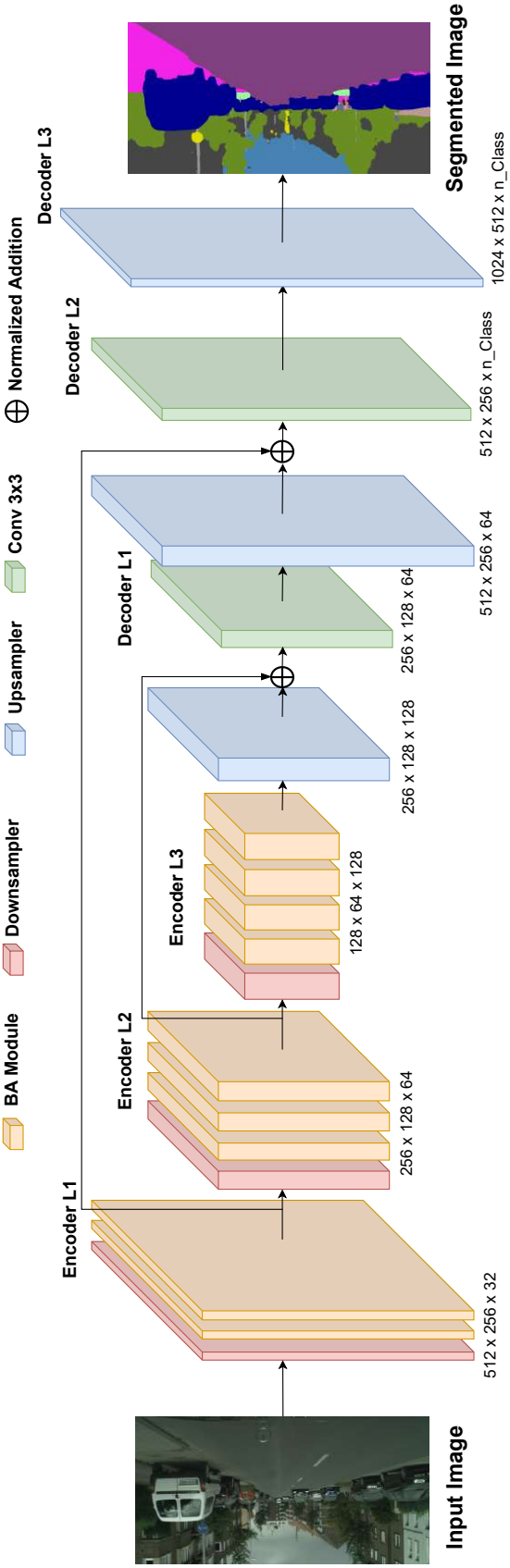


Figure 4.5: Block diagram of the proposed network. The colour of the blocks represents their type, given in the top row. The encoder has three levels, represented as L1, L2 and L3. Each level operates at half the resolution of the previous level. The dimension of the feature maps is given at the bottom of the block as '$Height \times Width \times Channel$'. n_Class is the number of classes in the final segmented image. The network consists of the proposed BA module, downsampler and lightweight decoder, the details of which are discussed in subsequent sections and figures.

4.3 Experiments

In order to validate the proposed BANet, extensive ablations are carried out on the widely used Cityscapes [1] dataset. In this section, the implementation details are discussed. In the ablation studies, the efficacy of our design choices and the network's overall design are investigated. The network's performance is compared with state-of-the-art models on the Cityscapes and CamVid [12] datasets.

4.3.1 Implementation Details

The model is implemented with PyTorch 1.10 [114] and CUDA 10.2. Adam optimizer with an initial learning rate of 5×10^{-3} and weight decay of 1×10^{-4} is used. . For learning rate (lr) adjustments, we use the poly strategy [47]:

$$new\ lr = (initial\ lr) \times \left[\frac{present\ iteration}{total\ iterations}\right]^{0.9} \tag{4.3}$$

Simple yet highly effective cross-entropy loss is used for the error calculation during learning. Data augmentation is essential to increase the number of training samples in order to prevent over-fitting [115]. The following data augmentation techniques are used to increase the number of examples in the Cityscapes training set: random horizontal and vertical flips, random translations of pixels in the range [0, 2], and random crops with a 512 crop size. We train the model with a batch size of 12 and 300 epochs. However, no ImageNet pre-training was employed [54]. The training was performed at an input image resolution of 512×1024, and the segmented output image was upsampled using bilinear interpolation for testing. The inference results have been reported on an Intel Core i7-11700-based desktop with 32GB RAM and a single NVIDIA RTX 3080 GPU unless stated otherwise.

4.3.2 Ablation Studies

This subsection reports experiments performed to support our design choices. For the ablation studies, the networks were trained on the Cityscapes training subset. Results are reported on its validation subset.

BA Module: The detailed evaluation results are presented for different design choices made in the BA module. We first analyze the effectiveness of using two cascaded 1×1 convolution layers in the two paths. The results in Table 4.2 show that using a single

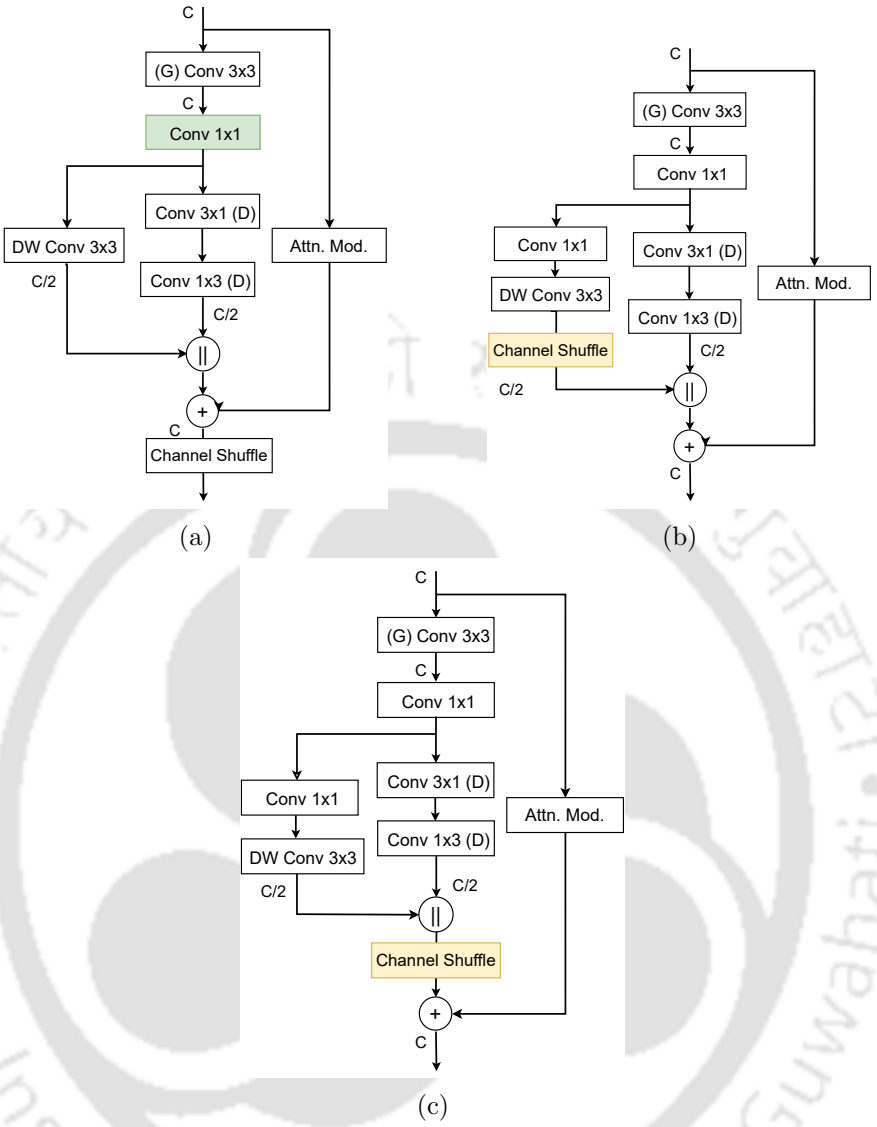


Figure 4.6: Different variants of BA Module. (a) is a lightweight design having a single 1×1 convolution in the main branch. (b) and (c) vary in the position of the channel shuffle operation.

1×1 convolution layer (BA Lite module, Figure 4.6 (a)) reduces the model accuracy with fewer parameters. An added convolution layer improves accuracy by adding learnable parameters along with non-linearity.

Next, the effect of using a channel shuffle operation at three different locations is studied. For the first (BA-1, Figure 4.6 (b), we choose a 3×3 depth-wise separable convolution (DW Conv) branch as used in [111]. The channel shuffle operation after the DW Conv enables the model to learn inter-channel correlation among the feature maps more effectively [111].

Table 4.2: Accuracy and parameter details for using cascaded 1×1 convolution layers in parallel branches of BA Module.

Module	1×1 Conv	Parameters (M)	mIoU (%)
BA Lite	No	0.60	63.9
BA	Yes	0.72	69.8

'M' - Million

Table 4.3: Accuracy comparison for different positions of channel shuffle operation in BA Module.

Module	Shuffle position	Channels	mIoU (%)
BA-1	DW Conv Branch	$C_{in}/2$	64.7
BA-2	After Concat	C_{in}	68.1
BA	After Residual	C_{in}	69.8

'C_{in}' indicates number of channels at input of module.

However, this placement was found to be less effective because the architecture only uses half the number of input channels in each of the two branches.

For the second variant (BA-2, Figure 4.6 (c)) the single branch location after channel concatenation, before adding a residual connection is selected. As the total number of channels is restored in this branch, the channel shuffle operation improved the BA module's feature learning capability. In the third variation of our BA module (Figure 4.2), the channel shuffle layer is put after adding the residual connection. This block design performs the best.

Backbone: As mentioned earlier, the backbone is built using BA and downsampling modules. In Table 4.4, it is seen that having (2, 3, 4) BA modules in corresponding stages gives the best accuracy. A higher number of modules in the initial layer increased inference time because of the large feature map. Increasing the network depth had an inhibitory effect on the network performance. It not only increases the number of parameters but also increases the latency. Given that our basic module could learn the required features with a shallow setting, greater depth showed no improvement.

The Decoder: In this ablation study, BA module-based backbone is fixed and two decoder variants are designed. The first employed bilinear upsampling with 3×3 convolution layers and the second employed 3×3 deconvolution layers directly. As seen in Table 4.5, the first combination has the highest speed and accuracy. For the primary input to the decoder, the final encoder stage feature maps are taken, which are $1/8\times$ the resolution of the input image. For the skip architecture, two auxiliary outputs from the backbone are taken, one after the first stage (1/2 resolution) and another after the second stage (1/4

Table 4.4: Accuracy comparison for different layer combinations in BANet.

Encoder Layers	Channels*	Time† (ms)	Parameters (M)	mIoU (%)
(-,4,8)	(-,64,128)	17.5	1.20	68.8
(2,4,8)	(32,64,128)	18.3	1.30	69.1
(2,3,6)	(32,64,128)	13.9	0.93	68.2
(2,3,4)	(32,64,128)	12.5	0.72	69.8
(2,2,4)	(32,64,128)	11.3	0.63	68.5

† Denotes the forward pass time for a single input image of resolution-1024 × 512, in milliseconds; *denotes channels in (Stage1, Stage2, Stage3).

Table 4.5: Decoder ablation studies. Results of various feature fusion strategies in the decoder.

Decoder	mIoU (%)
Add+Up+Conv	69.8
Add+Deconv	65.1
Mul+Up+Conv	64.5
Mul+Deconv	65.2
Concat+Up+Conv	63.7
Concat+Deconv	64.8

Table 4.6: Decoder ablation studies. Results of the proposed encoder combined with different decoders.

Decoder	Parameters (M)	mIoU (%)
ERFNet [47]	0.15	64.3
ESPNet [51]	0.04	63.5
LEDNet [101]	1.30	65.2
RGPNet [104]	2.00	67.5
FASSD-Net [61]	0.95	66.4
Proposed	0.09	69.8

resolution). This design strategy is adapted from [3]. We have experimented with normalized addition, multiplication, and concatenation of the auxiliary skip connections with the primary input. In the case of addition or multiplication, first stage output is applied with $[1 \times 1, 64]$ convolutions, where 1×1 is the kernel size, and 64 is the number of channels. While, for concatenation $[1 \times 1, 32]$, convolution is used. Similarly, the second stage output is applied with $[1 \times 1, 128]$ and $[1 \times 1, 64]$ convolutions. The channel widths are reduced by half for the concatenation operation while keeping other settings unchanged.

To show the efficiency of our decoder design, we trained the BANet backbone with some of the popularly used decoders from [47, 51, 101, 104, 61]. The results in Table:4.6 shows that the proposed decoder design performs the best. Our network is shallower than other networks, thus requiring a custom-designed decoder.

Table 4.7: Accuracy and inference time comparison for different dilation rates and residual connections.

Dilation rates (Stage1)+(Stage2)+(Stage3)	Residual Connection			
	Attention		Direct	
	mIoU (%)	Time (ms)	mIoU (%)	Time (ms)
(1,2)+(1,2,8)+(1,2,8,16)	63.6	14.38	62.8	13.81
(1,2)+(2,4,6)+(8,10,12)	66.5	13.74	65.2	12.53
2*(1)+3*(4)+(4,6,8,10)	65.1	14.30	63.4	14.07
2*(1)+(1,3,6)+(1,3,6,12)	66.4	13.69	64.7	12.99
2*(4)+3*(8)+4*(10)	67.9	14.47	65.3	13.92
2*(1)+3*(2)+4*(2)	69.8	12.53	67.9	11.77

2*(1) indicates 2 blocks with dilation '1' in a stage.

Dilation rates: To improve contextual information, semantic segmentation networks use dilation of the convolution kernels in the backbone [47, 50, 73]. This increases the pixel's field of view at the kernel's center. Increasing the dilation rate with the depth of the network is common [58, 59, 101]. Table 4.7 shows the ablation study for different dilation rates in the backbone. Dilations only in the asymmetric convolution branch of the BA module are used. It is observed that in the presence of an attention-based skip connection in the basic block, a dilation rate of two throughout gives the best accuracy.

Increasing the dilation rate further has a detrimental effect. However, if we replace the attention skip connection with the standard skip connection, it is seen that similar accuracy is attained at higher dilation rates. This further reinforces our claim of the efficacy of the BA module.

4.3.3 Comparisons on the Cityscapes benchmark

This section compares the overall performance of the proposed architecture with state-of-the-art semantic segmentation networks on the Cityscapes dataset. The best-performing deep network model, in terms of accuracy, is ViT-Adapter-L [105], which is based on the resource-hungry vision transformer [62]. Table 4.8 shows that state-of-the-art models, which require many parameters and FLOPs [67, 33, 105, 102, 65] report the top mIoU scores. Such models are not readily deployable on resource-constrained embedded devices.

Therefore, some of the best-performing lightweight and real-time models are focused upon. ENet [50] is the smallest peer-reviewed semantic segmentation model for road scenes, followed by ESPNet. Our model is substantially more accurate than either of them but requires slightly more parameters. ERFNet outperformed the proposed method by 1.2%.

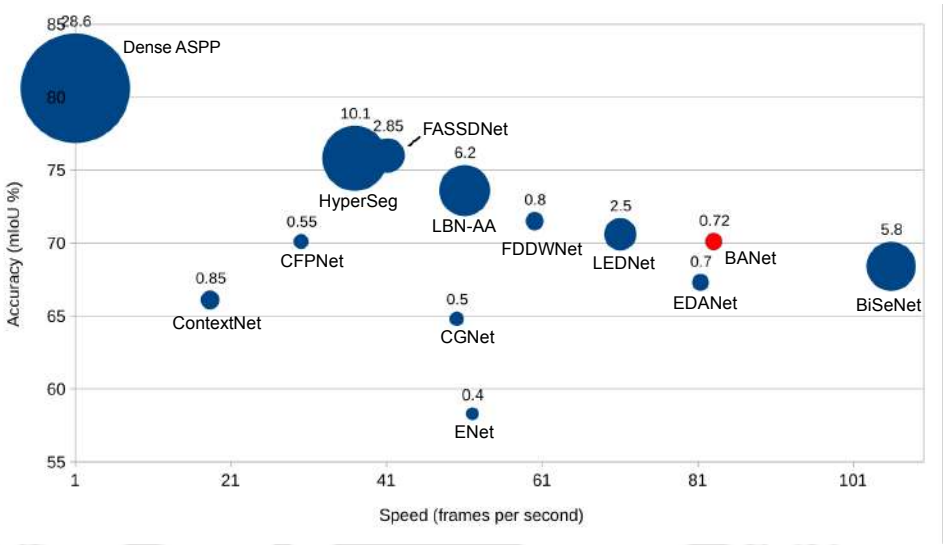


Figure 4.7: Speed, accuracy, and model size comparison on the Cityscapes *test* set. The size of the bubble represents the number of parameters written on top of each. The proposed method is highlighted in red. It balances the accuracy-speed tradeoff while having a small model size. Methods with high accuracy have large sizes and are significantly slower.

Table 4.8: Comparison of evaluation results of BANet with similar networks on Cityscapes *test* benchmark. The first section is for high-accuracy methods, which are usually very large. The second section is for lightweight methods.

Method	FLOPs (G)	InputSize	mIoU (%)	FPS*	Params (M)
Dense ASPP [33]	214.7	2048 × 1024	80.6	< 1	28.6
SegFormer-B5 [102]	1447.60	1024 × 1024	84.0	2.5	84.7
Trans4trans [65]	94.25	1536 × 768	81.5	27.3	49.5
PSP Net [67]	453.6	2048 × 1024	78.4	< 1	65.6
ViT-Adapter-L [105]	-	896 × 896	**84.9**	< 1	347.9
ENet [50]	3.8	640 × 360	58.3	135.4	0.4
ESPNet [51]	4.0	512 × 1024	60.3	112	0.4
EDANet [106]	11.34	512 × 1024	67.3	81.3	0.7
ERFNet [47]	21.0	512 × 1024	68.0	41	2.1
LBN-AA [35]	49.5	448 × 896	73.6	51	6.2
CFPNet[108]	14.7	1024 × 2048	70.1	30.0	0.55
ICNet [53]	28.3	1024 × 2048	69.5	30.3	7.8
Hyperseg-M [30]	7.5	512 × 1024	75.8	36.9	10.1
FDDWNet [116]	10.3	512 × 1024	71.5	60	0.8
BiSeNet [55]	14.8	768 × 1536	68.4	72.3	5.8
MLFNet-MobileV2 [57]	4.67	512 × 1024	71.5	90.8	4.0
ContextNet [54]	6.74	1024 × 2048	66.1	65.5	0.85
FASSDNet [61]	45.1	1024 × 2048	76.0	41.1	2.85
LEDNet [101]	11.44	512 × 1024	70.6	40.0	0.94
CABiNet [58]	12	1024 × 2048	75.9	76.5	2.64
CGNet [73]	7.14	512 × 1024	64.8	50	0.5
BANet	6.96	512 × 1024	70.1	83.2	0.72

*reported on different GPUs.'-' indicates that the method does not report the result.

However, it has higher FLOPs (21Giga) and model size (2.1M). This is due to many encoder layers with asymmetric convolutions and deconvolution kernels in the decoder. Despite having 10× more parameters than the average lightweight model, Image-Cascade Network [53] has comparable accuracy. This shows that there are several redundancies present in the deep networks. Hyperseg [30], though has comparable FLOPs with our model, it has 15× more parameters than the proposed model. With 16.5% fewer parameters, BANet can significantly improve over ContextNet [54]. LEDNet [101] achieves impressive accuracy, working in real-time but has 3.5× more parameters and FLOPS than the proposed method. LBN-AA [35], FASSADNet [61], and CABiNet [58] are more accurate than our model but have many parameters and FLOPs. These results show that BANet is comparable with the SOTA methods trained on Cityscapes. In addition, it can efficiently balance size, speed, and accuracy, which is the prime requirement of model design for embedded devices. Therefore, some of the best-performing lightweight and real-time models are focused upon. ENet [50] is the smallest peer-reviewed semantic segmentation model for road scenes, followed by ESPNet. Our model is substantially more accurate than either of them but requires slightly more parameters. ERFNet outperformed the proposed method by 1.2%. However, it has higher FLOPs (21Giga) and model size (2.1M). This is due to many encoder layers with asymmetric convolutions and deconvolution kernels in the decoder. Despite having 10× more parameters than the average lightweight model, Image-Cascade Network [53] has comparable accuracy. This shows that there are several redundancies present in the deep networks. Hyperseg [30], though has comparable FLOPs with our model, it has 15× more parameters than the proposed model. With 16.5% fewer parameters, BANet can significantly improve over ContextNet [54]. LEDNet [101] achieves impressive accuracy, working in real-time but has 3.5× more parameters and FLOPS than the proposed method. LBN-AA [35], FASSADNet [61], and CABiNet [58] are more accurate than our model but have many parameters and FLOPs. These results show that BANet is comparable with the SOTA methods trained on Cityscapes. In addition, it can efficiently balance size, speed, and accuracy, which is the prime requirement of model design for embedded devices.

4.3.4 Comparisons on the CamVid benchmark

To further demonstrate the generalisation ability of the proposed model, it is compared with some state-of-the-art networks trained on the CamVid dataset in Table 4.9. Our network has decent accuracy with a small memory footprint while running in real-time. Specifically, the accuracy of the proposed method is comparable to BiSeNet but with 8× fewer parameters. However, due to more usage of 1×1 convolution kernels, our network is

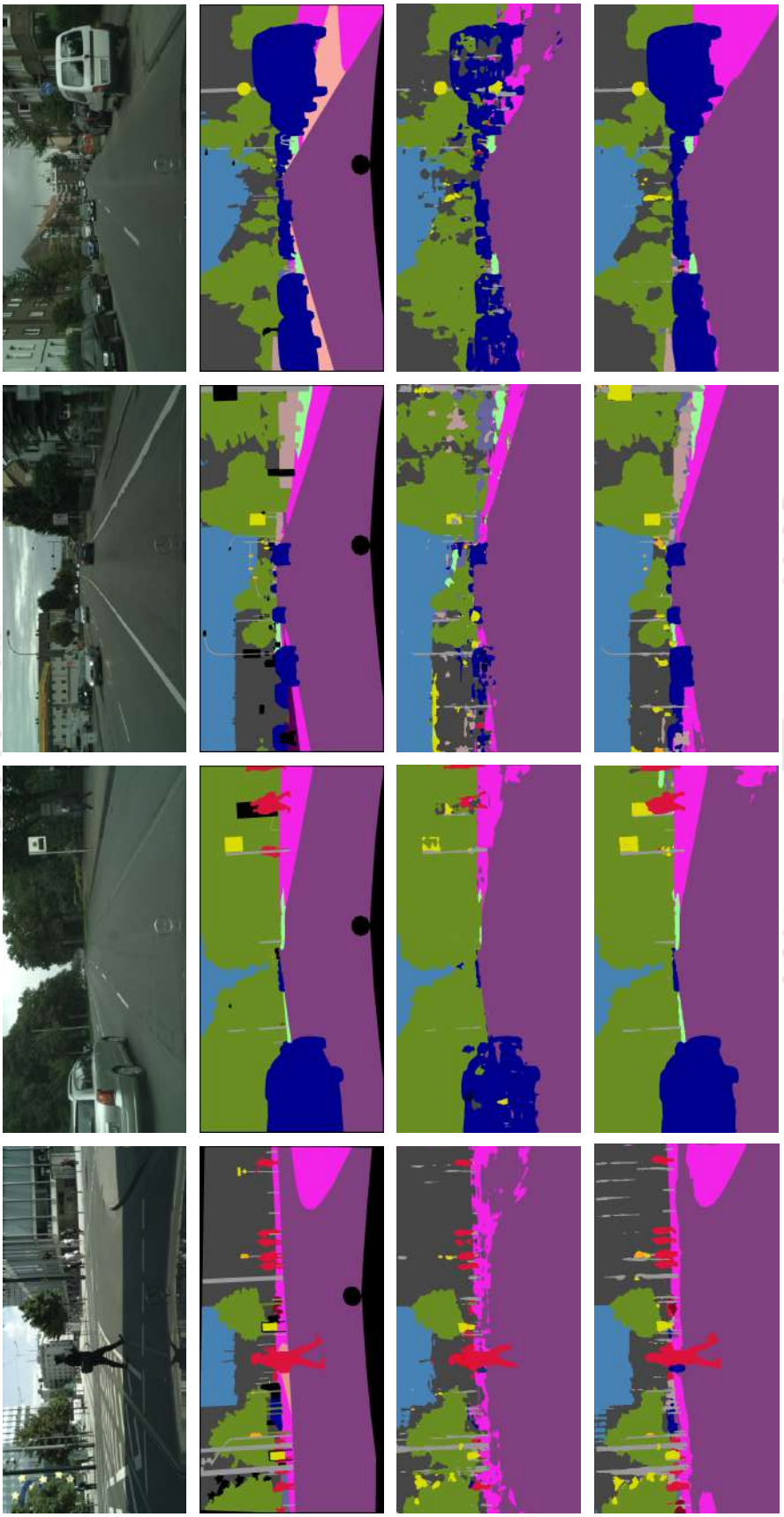


Figure 4.8: Qualitative results on Cityscape validation set. First row: Input image; second row: ground truth; third row: BANet output without normalized addition and decoder; fourth row: BANet output with normalized addition and decoder. It can be seen that the method performs the best with the decoder.

slower than BiSeNet. EDANet [106] performs closely to our method but with significantly less speed. FASSDNet outperforms in terms of speed and accuracy but with 4× more parameters. LAANet [74] was among the fastest but worked at a quarter of the resolution of the proposed network. These results show that our method can perform on par with state-of-the-art methods while being lightweight.

Table 4.9: Comparison of evaluation results of BANet with similar networks on Camvid *test* benchmark.

Method	InputSize	mIoU (%)	FPS*	Params (M)
RTFormer-Slim [110]	960 × 720	81.4	190.7	4.8
SegNet [22]	480 × 360	46.4	49.4	15.2
EDANet [106]	480 × 360	66.4	40.75	0.7
DFANet [24]	960 × 720	59.3	116	7.8
SwiftNet-MN [68]	960 × 720	65.0	27.7	11.8
ICNet [53]	960 × 720	67.1	46.7	7.8
BiSeNet [55]	960 × 720	65.6	175	5.8
BiSeNetV2-L [56]	960 × 720	73.2	32.7	-
MLFNet-Res34 [57]	960 × 720	69.0	57.2	4.0
LAANet [74]	480 × 360	67.9	112.5	0.7
CGNet [73]	960 × 720	65.6	59.0	0.5
FASSDNet [61]	960 × 720	69.3	80.0	2.85
BANet	960 × 720	66.3	68.1	0.72

*As reported in method on different GPUs. '-' indicates that the method does not report the result.

4.3.5 Qualitative results

The segmented images from the Cityscapes *validation* set and CamVid *test* set are shown in Figure 4.8 and Figure 4.9, respectively. Figure 4.8 shows that the lowest dilation rates achieve the best qualitative results in the backbone. The first row in the figure shows only the encoder output. Without a decoder, the model performs poorly, as seen in these segmented images. The CamVid visualization results show that, while our method can accurately segment large objects like roads, cars, vegetation, buildings, and sky, it misses smaller objects like poles and traffic signs.

The segmented images from an unseen Berkeley deep drive (BDD) [11] dataset are also visualized in Figure 4.10. This dataset was chosen for its class compatibility with Cityscapes. It is observed that the method performs well for large objects, even on unseen road scene datasets. In addition, to further verify the generalizability of the proposed network, we obtain the segmentation results on images captured from the institute campus shown in

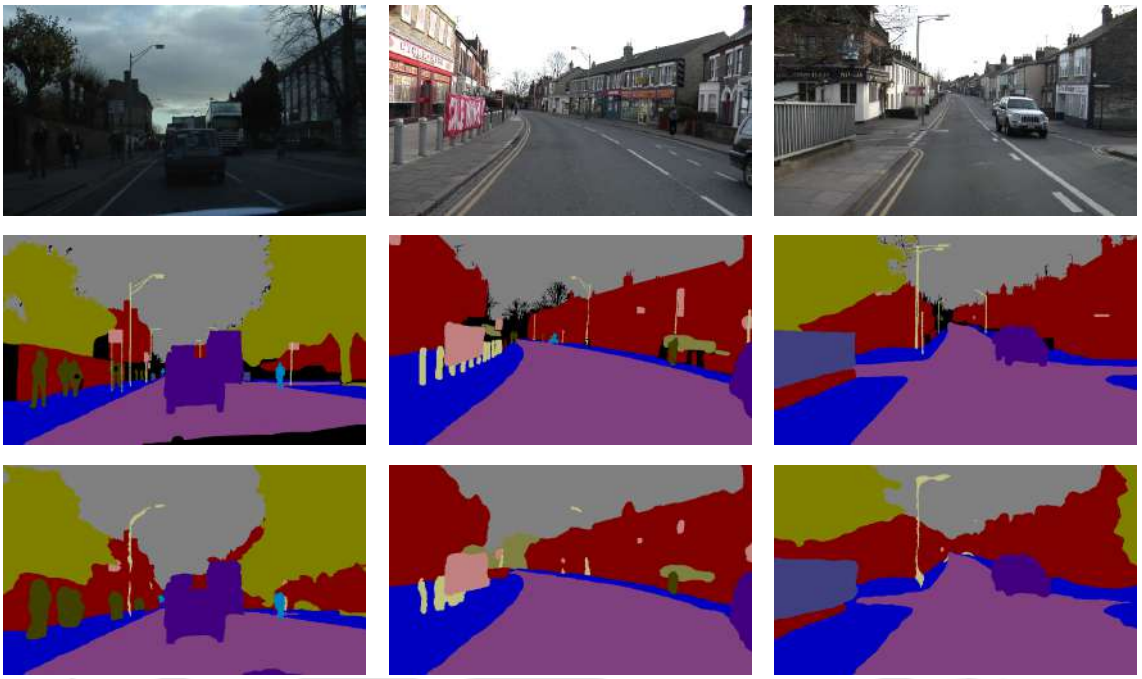


Figure 4.9: Qualitative results: Validation images on the CamVid dataset. First row: Input image; second row: ground truth; third row: BANet output.

Table 4.10: Inference speed(fps) of the proposed method for different input image resolutions on mobile GPU-based systems.

Method	Jetson Xavier		940MX		mIoU	Params
	2048 × 1024	1024 × 512	2048 × 1024	1024 × 512	%	(M)
SwiftNet [68]	2.61	9.9	2.4	8.64	70.2	11.8
ENet [50]	13.8	53.82	13.4	45.56	58.3	0.4
ContextNet [54]	10.49	40.91	10.2	35.7	66.1	0.85
Fast-SCNN [45]	11.49	40.21	11.34	37.42	68.4	1.14
LEDNet [101]	0.7	2.73	0.5	1.75	70.6	0.94
BiSeNet [55]	2.42	9.6	2.1	7.35	68.4	5.8
CaBiNet [58]	8.21	30.95	7.9	27.65	76.5	2.64
FASSDNet [61]	7.3	29.2	7.1	26.27	76.0	2.85
MLFNet [57]	8.41	32.79	8.15	30.15	71.5	4.0
BANet	9.02	34.23	8.2	30.12	70.1	0.72

Figure 4.11. The images are randomly selected from videos recorded through a car-dash-mounted camera at 1080p and 30fps. The network used is pre-trained on the Cityscapes dataset. The visualization shows that the performance of the network, even on unseen images, is at-par with the datasets used for training.

4.3.6 Evaluation on mobile GPU systems

Mobile GPUs are manufactured for resource-constrained devices like laptops and embedded systems. They are designed with fewer CUDA cores and lower clock frequency. As a result,

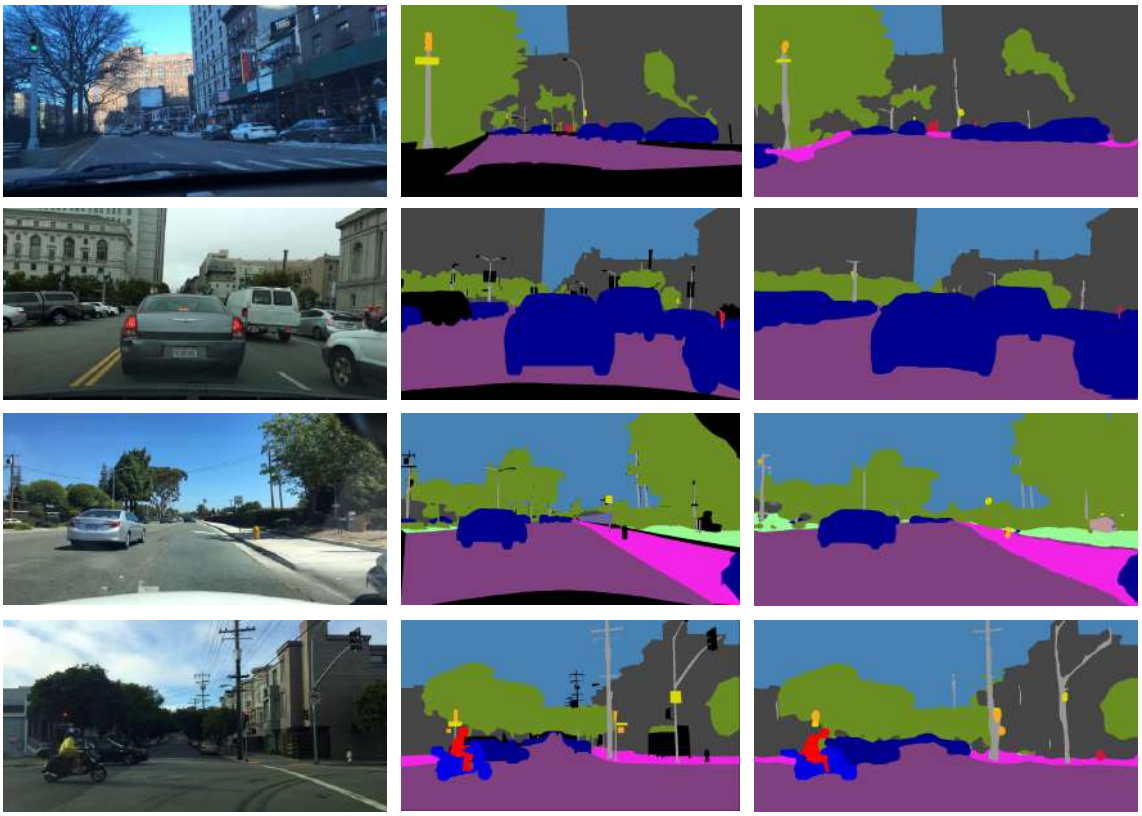


Figure 4.10: Qualitative results on the unseen dataset: Validation results on Berkeley Deep Drive (BDD) dataset. First column: Input image; second column: ground truth; third column: BANet output.

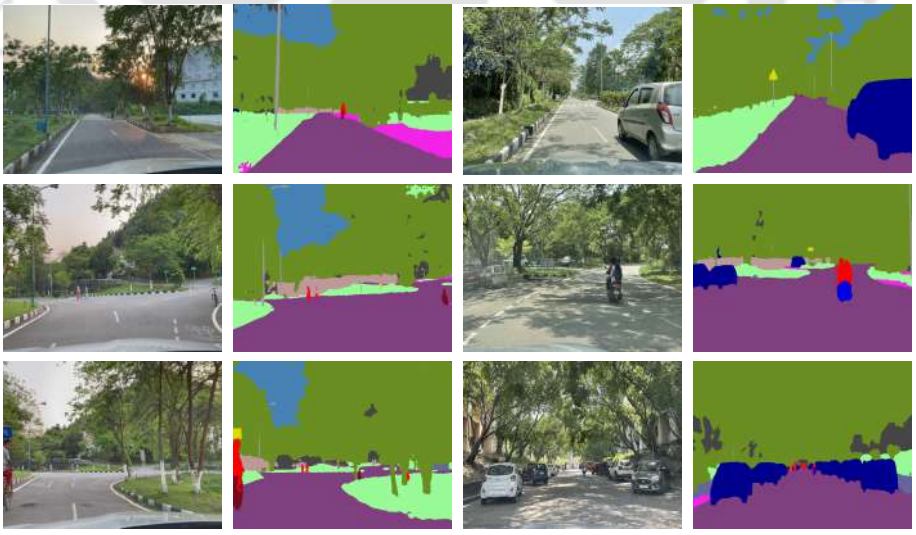


Figure 4.11: Qualitative results on unseen images: Results on our custom captured images in the institute campus. First/third column: Input images; second/fourth column: BANet outputs.

they have low power consumption on the order of $20W$. The proposed method is evaluated on two mobile GPUs, NVIDIA Jetson Xavier and NVIDIA 940MX. Table 4.11 presents their technical specifications. Both the GPUs are comparable, apart from their speed and memory bandwidth. In Table 4.10, we present the performance analysis of the proposed model for two different input resolutions. Our model runs faster than SwiftNet, LEDNet, BiSeNet, and CaBiNet on both devices for a 1024×512 input image. This can be attributed to the lightweight design. However, ENet is faster than the other methods because of its shallow and extremely small network structure, but it has low accuracy. Despite having a very small size, LEDNet shows an abysmal speed. This could be due to its extremely complex decoder design resulting in many FLOPs (ref. Table 4.8). Fast-SCNN is faster than the proposed model in speed with a low accuracy score.

Table 4.11: Specifications of the mobile-GPU cards.

GPU Card	Cores	Speed	RAM	Mem.B/W	Power
Xavier	384	854MHz	8GB	51.2 GBps	15W
940MX	384	1.2GHz	4GB	40.1 GBps	20W

'RAM': Random Access Memory; 'Mem.B/W': Memory Bandwidth.

4.4 Contribution

The main contributions of this chapter are summarized below:

- A basic bottle-neck block is proposed, incorporating an attention-based skip connection, referred to as the block-attention module. It extracts local and contextual features with a parallel branch structure comprising dilated asymmetric and depth-wise separable convolutions.
- Our network, BANet, builds on the repetition of the BA module with downsampling modules placed at appropriate locations. The network balances model size and accuracy while improving inference speed.
- A novel, lightweight decoder is designed using efficient 3×3 convolutions to include in our network. With only simple skip connections from the encoder stages, it can recover some spatial details lost during the downsampling process.
- Without any pre-training, post-refinement, or pyramid pooling modules (PPMs), our lightweight architecture achieves competitive results on the Cityscapes and CamVid

datasets. Specifically, with only 0.72M parameters, BANet achieves a mIoU of 70.1% and 66.3% on the Cityscapes and CamVid benchmarks, respectively.

4.5 Summary

This paper proposed a lightweight real-time semantic segmentation model for road-scene understanding, targeting resource-constrained devices. We have developed a novel block-attention module that uses simple attention-based skip connections for enhanced feature learning capabilities. The network based on the block-attention module has reduced latency due to lower dilation rates in the encoder backbone. We have also introduced a lightweight and efficient decoder design in our network, which performs better than popular decoders. Through extensive experiments on highly competitive benchmark datasets - Cityscapes and CamVid, we have presented the efficacy of our design choices. Appropriately balancing size, accuracy, and speed, BANet outperforms some of the attention based state-of-the-art lightweight models while achieving 70.1% and 66.3% mIoU on Cityscapes and CamVid datasets in real time. The inference speed of 34.23 fps on mobile devices further shows that the proposed model is suitable for resource-constrained devices.

Chapter 5

Context Guided Multiscale Attention for Realtime Semantic Segmentation

5.1 Introduction

The effective handling of multi-scale objects in the road-scene images poses a significant challenge in semantic segmentation networks. A simple CNN architecture with conventional convolution kernels can learn the local correlations among the pixels, however, they struggle to efficiently capture large objects due to lack of sufficient contextual information. In the previous chapters, our design strategy mostly revolved around using dilated convolutions in building blocks to enlarge receptive fields and capture broader context. However, they may lead to dominant representations of large objects, neglecting details of small or isolated objects. A simple attention mechanism was incorporated in the block level in Chapter 4 to model the context at local level.

In this Chapter we further explore advance techniques such as multi-scale feature encoding and context aggregation along with attention mechanism to efficiently handle the multi-scale objects and learn global context for accurate segmentation. Based upon these techniques, we propose the CGMA-Net (Context Guided Multiscale Attention Network), a convolutional neural network for the semantic segmentation of road scenes in a resource-constrained environment. In this work, two modules have been designed: the Fast Hybrid Module (FHM) and the Multi-Scale Attention Module (MSAM). The tailored encoder

backbone is built using the FHMs. At the same time, the decoder is designed around the MSAMs. The FHM is designed to jointly capture local semantics and contextual features at multiple scales through a three-branched structure. The first branch learns the local feature through conventional convolution. The second branch captures the contextual features using stacked dilated convolution kernels. This setup increases the kernel receptive field. The computational complexity is reduced by carefully applying depth-wise separation to the dilated kernels. In line with the requirement of capturing the image features around smaller objects and boundaries, the MSAM is designed to generate an attention vector guided by low-level context features. The MSAM can be configured to extract features at different scales, thus improving the feature discrimination ability. The weights are efficiently learned through pointwise convolutions, further reducing the computations.

By leveraging the fast hybrid and multi-scale attention modules, the proposed CGMA-Net achieves state-of-the-art segmentation accuracy among lightweight real-time networks on the extremely challenging Cityscapes and CamVid datasets. The proposed method achieves remarkable inference speed with a low computational cost of 5.86 Giga FLOPs. Through comprehensive experiments conducted on the Cityscapes dataset, the efficacy of the design choices is shown. Due to its inherent fast and lightweight design, the proposed method can run in real-time on low-power embedded devices, such as Jetson Xavier and 3060-Mobile GPU.

The remaining chapter is organized as follows: Section 2 introduces the related works; the functional modules, along with the network architecture, are detailed in Section 3; the implementation details, along with the ablation studies and comparison with state-of-the-art and qualitative results, are discussed in Section 4; Section 5 concludes the paper with the scope of future works.

5.2 Proposed Method

For accurate semantic segmentation, global context and local semantics features are essential. While global context is required to understand the scene and placement of objects, local semantics is required to label individual object pixels. Multi-scale feature learning is essential to segment objects of varying sizes in a typical road scene image. This section discusses the proposed Fast Hybrid Module (FHM), which essentially learns the context and semantics at local level. To further enhance the global feature learning capability of the network, a Multi-Scale Attention Module (MSAM) is proposed, which is discussed subsequently. Finally, the complete structure of the CGMA-Net is described.

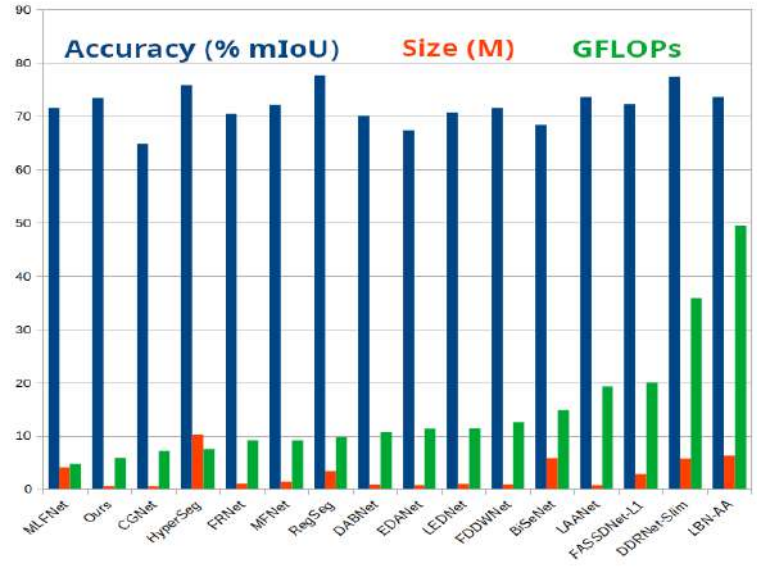


Figure 5.1: Comparison with some state-of-the-art lightweight networks on the Cityscapes dataset [1]. The proposed network achieves comparable accuracy with one of the minimum sizes and FLOPs.

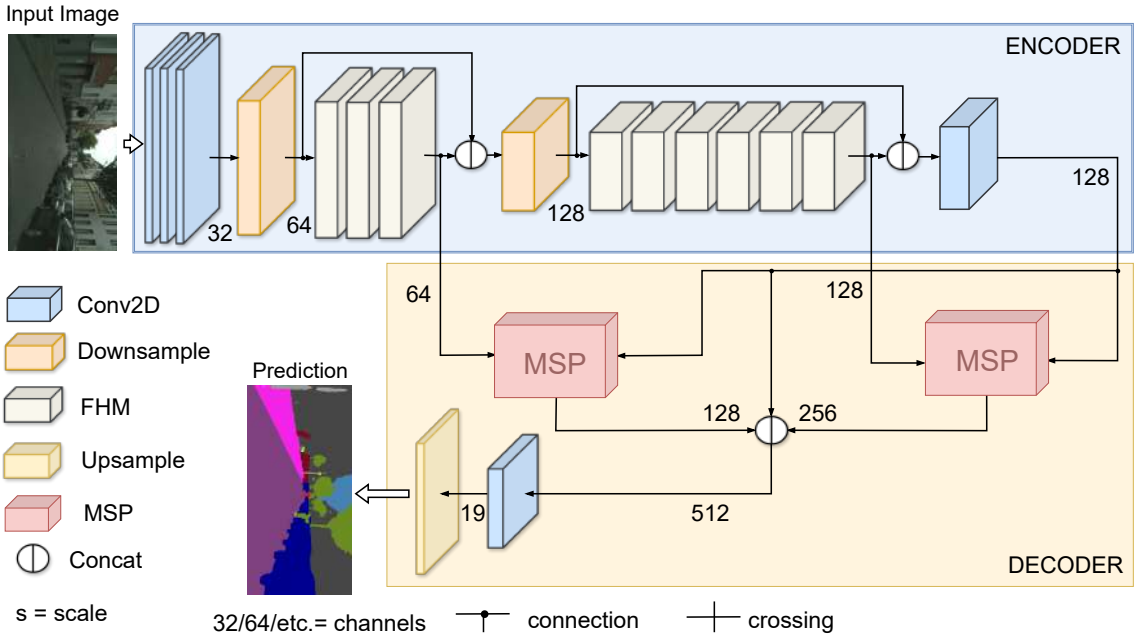
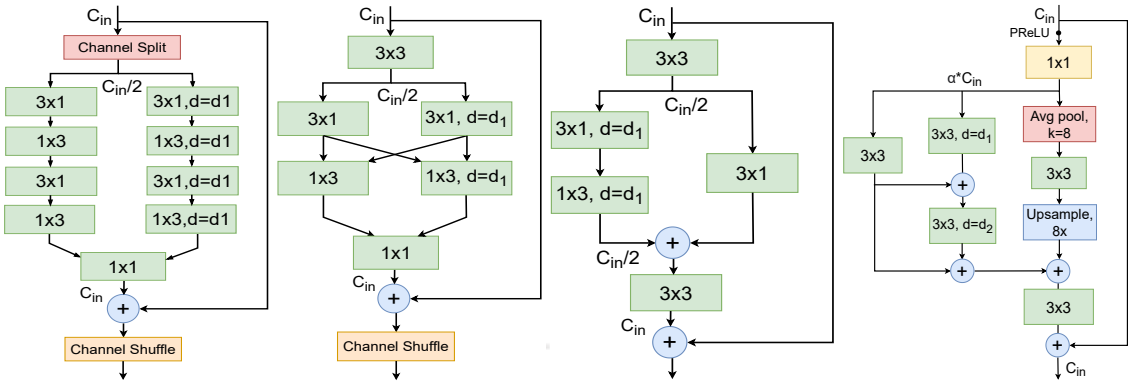


Figure 5.2: Architecture of the proposed Context Guided Multiscale Attention Network (CGMA-Net).

5.2.1 Fast hybrid module

The fast hybrid module (FHM) serves as the primary building block of the encoder, embodying an inverse residual bottleneck design [34]. This module features a carefully designed three-branch structure, as shown in Figure 5.3d. A multi-branch design has proven effective in efficiently encoding spatial relationships and forming a feature pyramid to learn representations. While [51] used five branches in the basic block, [52] redesigned it with four branches to improve the performance by further reducing the operations. However, we found optimal performance using only three branches like [117, 118]. A common choice for the basic block is the bottleneck structure [66, 47, 46, 59, 76, 116], which reduces the number of input channels for processing within the block. This is essential to reduce the computations in deep networks. However, [42] showed that if the input manifold of interest (or simply, the number of valuable features) is a subset of the input feature space (represented as the total number of input channels), only then the bottleneck structure followed by the non-linear transformation can preserve all the essential features. This is because non-linearity can map many features to zero values outside the output feature space. For example, a line is mapped to a ray in 2D space by a ReLU. As a remedy, the inverse bottlenecks essentially expand the number of output channels so that the manifold of interest can be retained even after collapsing the feature space by the non-linearity. A more formal proof can be found in [42]. The expansion ratio of the channels is controlled by a hyperparameter 'α'. As already determined in our previous work [119], a value of $\alpha = 2$ is sufficient to balance computational efficiency and retained feature richness. A 1×1 convolution layer is used to increase the channel dimension. The feature maps are simultaneously processed in three parallel branches, each having a strategic importance. The first branch leverages a 3×3 standard convolution to capture local semantic details efficiently, which is essential for understanding fine-grained image features. The second branch employs a sequence of dilated convolutions with progressively increasing dilation rates, enabling contextual information extraction over varying receptive fields. The skip connection between the first and second branches plays a crucial role in enhancing the network's functionality: it facilitates improved gradient flow during backpropagation and increases the range of effective dilation rates, thereby enabling robust multiscale feature learning. In feature pyramids, dilations are progressively increased with each parallel branch. However, large dilation rates can significantly slow down inference due to sparse kernel operations and zero padding, necessitating kernel restructuring [61]. Moreover, they introduce the gridding effect in the reconstructed image [120]. To solve this, the third branch strategically reduces the input feature map dimensions using average pooling



(a) LEANet Module **(b)** LAANet Module **(c)** MFNet Module **(d)** FHM:Fast-Hybrid Module

Figure 5.3: Overview of some basic modules. 'C_{in}': Number of input channels. '3x1' is the convolution kernel size. 'd': Dilation rate. '+' denotes element-wise addition. 'α' is the width multiplier.

by a factor of 8, followed by a 3×3 convolution. This configuration emulates the effect of a dilated convolution with a rate of 8, capturing wide context but with notably reduced computational expense. The integration of these three branches allows FHM to concurrently process input feature maps across multiple scales with different convolutional strategies, ensuring efficient multiscale feature aggregation and reduced inference time. This hybrid approach seamlessly enhances dense feature extraction architectures, optimizing real-time semantic segmentation tasks by preserving both local detail and global context without compromising on speed or computational load.

5.2.2 Multi-scale attention module

First, the problem formulation is discussed in the following lines. Suppose we have an encoded feature vector cube X, of dimension $h \times w \times C$. There is a need to label each pixel location in the original image using the vectors x_i at position i in X. However, not all features in X are equally crucial for label assignment. Hence, an attention-guided context map is generated for the weighted addition of the features. The context information is extracted from the low-level feature maps, $\overline{X}$. For a given scale, s, X is divided into $s \times s$ size patches following [121]. If the weighted feature at position i is denoted as $z^s_{i,j}$, it can be generated using X and $\overline{X}$ as follows:

$$z^s_{i,j} = \sum_{j=1}^{s \times s} \beta_{i,j} y_j \tag{5.1}$$

where y_j is the reduced feature map obtained from X by max-pooling operation and convolution and $\beta_{i,j}$ is the attention weight for position i generated using $\overline{X}$ at j. It can be calculated as follows:

$$\beta_{i,j} = f_{Attention}(\overline{X_i}, j) \tag{5.2}$$

The attention function, $f_{Attention}$ is defined as follows:

$$f_{Attention}(\overline{X_i}, j) = \sigma(W_{att}(W_{ch}(\overline{X_i}) + \overline{X_i})) \tag{5.3}$$

where σ is the sigmoid function; W_{att} is the attention weight generated by a pointwise convolution kernel, and W_{ch} is the channel weight generated using average pooling operation along the channel axis of the low-level feature maps.

The main difference in this work is how $z_{i,j}$ is calculated. Methods discussed in [121] generate $z_{i,j}$ by only utilizing information in encoded feature vector cube X. At the same time, the context information from a low-level feature map $\overline{X}$ is used to generate weights. The primary intuition behind this is that as we move along the depth of the deep network, high-frequency information, like image boundaries, is lost [72]. These high-frequency details are exploited in the low-level feature maps to weigh the encoded feature maps suitably. The method to calculate MSAM features can be described by Algorithm 1.

Implementation: The MSAM architecture is shown in Figure 5.4. It has two inputs: the first is for low-level feature maps ($\overline{X}$), and the second is for high-level feature maps (X). In the first branch, $\overline{X}$ is initially convolved with a 1×1 filter to match the channel dimension. Then, channel attention is applied to these feature maps. The weights, W_{ch}, are calculated by passing through global average pooling followed by a 1×1 convolution. A context-guided attention vector, β, is obtained by generating weights, W_{att}, through a 1×1 convolution followed by a sigmoid function to the weighted low-level feature maps.

In the second branch, the high-level feature maps are first reduced by average pooling with a kernel of size 's'. After passing through a 1×1 convolution kernel, the reduced feature map, Y, is obtained. Y is multiplied with the attention-vector β to obtain the weighted output feature map, Z.

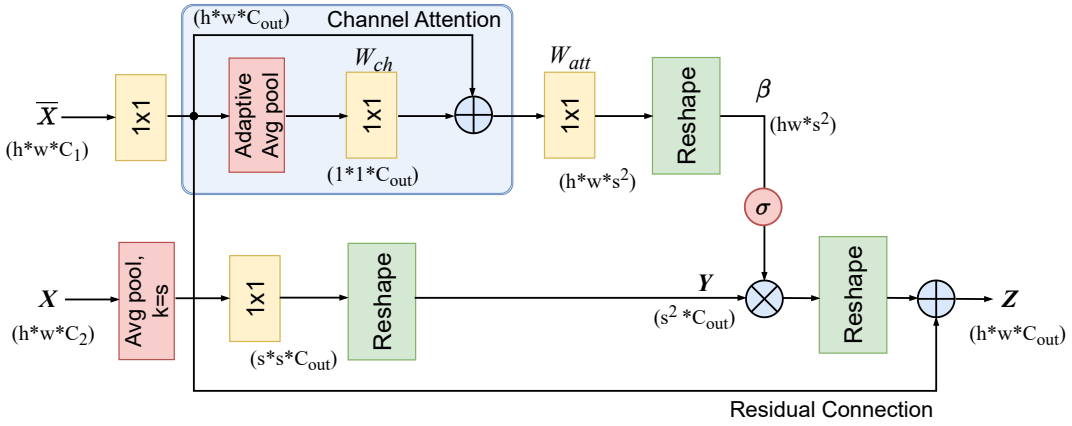


Figure 5.4: Architecture of the Multi-Scale Attention Module (MSAM). '(h*w*C)' is the feature map cube dimension. 's' is the scale. 'X' is high-level feature input, while 'Y' is the size-reduced feature cube. '$\overline{X}$' is low-level feature input. 'β' is the attention weight matrix.

Algorithm 1: MSAM calculation process

Input: $\overline{X} \in \mathbb{R}^{[B,C_1,h,w]}$: low-level SFM $X \in \mathbb{R}^{[B,C_2,h,w]}$: high-level SFM

`//SFM: sub-feature-map`

Output: $Z \in \mathbb{R}^{[B,C_{out},h,w]}$: output tensor

`/* Functions are in Pytorch style */`

`//Attention vector generation`

$\overline{X} \leftarrow \text{Conv2d}(\overline{X}, kernel = 1, groups = 1)$;
$\overline{F}_{avg} \leftarrow \text{AdaptiveAvgPool2d}(\overline{X}, \text{size=1})$;
$W_{ch} \leftarrow \text{Conv2d}(\overline{F}_{avg}, kernel = 1, groups = 1)$;
$\overline{F}_{avg} = W_{ch} + \overline{F}_{avg}$;
$W_{att} \leftarrow \text{Conv2d}(\overline{F}_{avg}, kernel = 1, groups = 1)$;
$\beta \leftarrow \text{Softmax}(\text{Reshape}(W_{att}, shape = hw \times s^2))$;

`//Context guided SFM generation`

$F_{avg} \leftarrow \text{AdaptiveAvgPool2d}(X, \text{size=s})$;
$F \leftarrow \text{Conv2d}(F_{avg}, kernel = 1, group = 1)$;
$Y \leftarrow \text{Reshape}(F, shape = s^2 \times C_{out})$;
$Z \leftarrow \beta * Y$;
$Z \leftarrow Z + \overline{X}$;

5.2.3 CGMA Net

The block-level architecture and detailed structure of CGMA-Net are shown in Figure 5.2 and Table 5.1, respectively. It follows an encoder-decoder network design with a three-stage encoder using FHM blocks. Stage 1 comprises three 3×3 convolution kernels, the first being conventional with stride=2 and the others depthwise separable. Stages 2 and 3 have intra-stage skip connections and contain three and six FHMs starting with downsamplers, respectively. The output of each stage is concatenated with its input through the skip connections to improve feature learning and gradient flow. The encoder output is 1/8th the size of the input, with 128 channels achieved through a 1×1 convolution operation before passing it to the decoder.

The decoder has two stages, each having a multi-scale pyramid (MSP) made of three MSAMs (scales 6, 3, and 1) (ref. Figure 5.5). The first stage receives inputs from the last encoder block before concatenation and channel reduction, resulting in 256 channels. The second stage takes low-level feature input from the second encoder block and high-level feature input from the final encoder block, yielding 128-channel feature maps. All MSAM outputs and final encoder outputs are concatenated and passed through a 1×1 convolution filter, reducing 512 channels to 19 classes. These stages operate at 1/8th of the input resolution, reducing inference time and computational load. The final decoder output is bilinearly upsampled to the original input resolution.

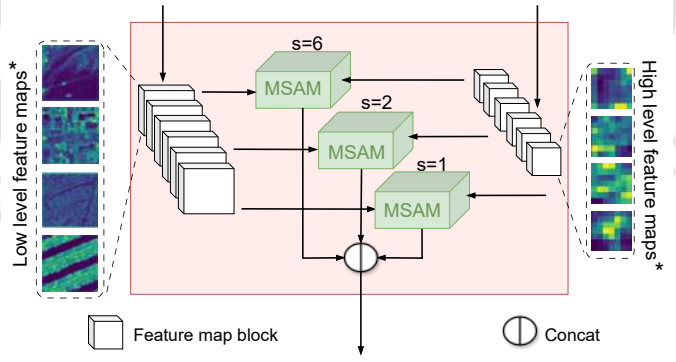


Figure 5.5: Structure of Multi-Scale Pyramid (MSP). *Feature map images taken from [2] for representation purpose.

5.3 Experimental Results

In this section, the results of the proposed network are presented on three publically available datasets: Cityscapes [1], CamVid [12], and Berkeley Deep Drive [11]. First,

Table 5.1: Detailed structure of proposed CGMA Net

	Layer	Operation	Mode	Channel	Output size
Encoder	1	3x3 Conv	stride=2	32	384x768
	2	3x3 (G) Conv	stride=1	32	384x768
	3	3x3 (G) Conv	stride=1	32	384x768
	4	Downsampling	stride=2	64	192x384
	5-7	FHM	d=(2,4)	64	192x384
	8	Downsampling	stride=2	128	96x192
	9-14	FHM	d=(2,4)	128	96x192
	15	1x1 Conv	stride=1	128	96x192
Decoder	16	MSAM	scale=6	128	96x192
	17	MSAM	scale=2	64	96x192
	18	MSAM	scale=1	64	96x192
	19	MSAM	scale=6	64	96x192
	20	MSAM	scale=2	32	96x192
	21	MSAM	scale=1	32	96x192
	22	1x1 Conv	stride=1	19	96x192
	23	Upsampling	factor=8	19	768x1536

(G) Conv- Group convolution; d=(d1,d2) is the dilation rate, refer Figure 5.3d.

the evaluation metrices are discussed, followed by the training settings. Then, the ablation studies are performed to show the importance of the design choices, followed by the comparison results with state-of-the-art methods.

Accuracy is measured in terms of mIoU. The computational complexity of the model is reported in terms of the number of floating point operations (FLOPs). It essentially measures the computations that involve multiplication and addition. As embedded devices have low computational power, a model with small FLOPs is preferred. The number of segmented frames-per-second (fps) has been used to establish the latency of the deep network. Following the convention, a fps of 30 and above is considered real-time.

Since the end goal of the work is to design a network for resource-constrained devices, the best possible accuracy-speed trade-off is needed with the constraints of model size and computation complexity. Previous works [122, 74] have considered the first three parameters, neglecting the FLOPs. A new scaler index has been formulated with all four parameters and termed as *Efficiency Index,* (EI), calculated as follows:

$$EI_i = \log \left[\frac{a_i \times m_{\mathrm{i}}^* + b_i \times f_{\mathrm{i}}^*}{c_i \times p_{\mathrm{i}}^* + d_i \times g_{\mathrm{i}}^*} \right] \tag{5.4}$$

Where m_i, f_i, p_i, g_i are mIoU, fps, size, and FLOPs of model i; the corresponding 'max' and 'min' values denote their maximum and minimum values selected from all the methods being compared; a_i, b_i, c_i, d_i are coefficients such that $a_i + b_i = 1$ and $c_i + d_i = 1$. The

parameters in Equation 5.4 are calculated as follows:

$$
\begin{aligned}
a_i &= \frac{f^*_{\max} - f^*_{\min}}{m^*_{\max} - m^*_{\min}}\, b_i \\
m^*_{\mathrm{i}} &= \frac{m_i}{m_{max}} \\
f^*_{\mathrm{i}} &= \frac{f_i}{f_{max}} \\
c_i &= \frac{p^*_{\max} - p^*_{\min}}{g^*_{\max} - g^*_{\min}}\, d_i \\
p^*_{\mathrm{i}} &= \frac{p_i}{p_{max}} \\
g^*_{\mathrm{i}} &= \frac{g_i}{g_{max}}
\end{aligned}
\tag{5.5}
$$

Here, $f^*_{\max}$ and $f^*_{\min}$ represent the maximum and minimum values of f^*_{i}. Similarly, the values of m^*_{i}, p^*_{i}, and g^*_{i} are represented. $f_{\max}$ and $f_{\min}$ represent the maximum and minimum values of FLOPs among all the methods being compared. Similarly, the corresponding values of m, p, and g are calculated. A 'log' scale is introduced to calculate EI to accommodate the huge parameter variations. It is noted that EI is suitable for ranking lightweight and real-time models because of its inverse relationship with the model size and FLOPs.

5.3.1 Training settings

All the ablation studies have been performed on a single Nvidia RTX 3090 graphics card system using PyTorch 1.10 framework and CUDA 11.6. A mini-batch stochastic gradient descent (SGD) optimizer is utilized for Cityscapes and BDD with a momentum of 0.9. Weight decay is set to 10^{-4}. Adam optimizer is applied for Camvid, owing to its small dataset size. The experiments involving Cityscapes and BDD are trained with a batch size of 8, while for CamVid, it is 12. 'poly' learning strategy is used with the rate at each iteration given as:

$$
new\_lr = initial\_lr \times \left[\frac{current\_iteration}{total\_iteration}\right]^{0.9} \tag{5.6}
$$

The initial learning rate is 0.045, and the training is done for 350 epochs for Cityscapes. Meanwhile, for BDD and CamVid, the epochs increased to 500 and 1000. For training, Cityscapes is used at a resolution of 768×1536. At the same time, CamVid and BDD

Table 5.2: Experimental results of CGMA-Net with different basic blocks on the Cityscapes validation set.

Method	mIOU (%)	FLOPs * (G)	Param (M)
LEANet	71.1	10.99	1.07
LAANet	70.5	7.94	0.8
MFNet	69.7	6.43	0.66
FHM	72.3	5.86	0.54

* FLOPs are reported for an input image of size 1024×512; 'M' - Million; 'G' - Giga.

are trained at 720×960. Following [46], online hard example mining (OHEM) loss is used for Cityscapes and cross-entropy (CE) loss is used for CamVid. The data augmentation techniques are random flips, random translation by a maximum of 2 pixels, random crop, and random scale in [0.5, 2.0], followed by colour jittering.

5.3.2 Ablation Study

In this section, a series of ablation studies have been conducted to validate the design choices in the modules and the overall proposed architecture. The performance is measured on the validation set of the Cityscapes.

Fast Hybrid Module: Methods [72, 70, 71, 46] proposed two branched basic blocks where one branch learns local features and other, the global context through dilated kernels. However, a single dilation rate was used. To verify the effectiveness of the FHM, it is replaced with the modules shown in Figure 5.3 and train the network on the same settings. While LEANet [122] stacked four kernels in each branch, LAANet [74] introduced criss-cross skip connections between the two branches. MFNet [72] followed an asymmetric design. Table 5.2 shows the result for different basic blocks. LEANet module achieves similar accuracy, albeit with double FLOPs and model size. Among the modules, FHM achieves the best accuracy with a minimum FLOP of 5.86 Giga and a model size of 0.54 M.

Pooling branch: To show the advantage of using the pooling operation for downsampling in place of large dilations in convolution kernels, the pooling-convolution-upsampling operation is replaced in the third branch. Instead, a depth-wise dilated kernel is introduced with different dilation rates (d). First, the kernel is dilated with a factor of 8. This essentially has a similar field of view (FOV) as the convolution kernel after the downsampling process by pooling. Table 5.3 summarizes the results. It is observed that using a

dilated convolution increases the inference time with increased FLOPs. Specifically, for a dilation rate of 8, the inference time is increased by 5%, and FLOPs are increased by 60 Million. Similarly, for a dilatation rate of 16, the inference time is increased by more than 20%. The skip connection between the two semantic branches of FHM further improves the accuracy by 0.8%. We further conduct ablation with different numbers of branches in FHM. Different structures with 2, 3, and 4 branches have been analyzed in Table 5.4. It can be seen that 3 branches with average-pool downsampling in the third branch give the optimal performance, balancing the accuracy, inference time, and memory requirement. A 4-branch structure with dilated convolutions gives the best accuracy, but it requires 50% more GPU memory and has a lower inference speed by 18%.

Table 5.3: Ablation study for pooling branch and skip connection in FHM on Cityscapes validation set.

Ablation Study	Type	mIoU (%)	FLOPs (G)	fpt (ms)
Pooling branch	Avg.Pool	72.3	5.86	7.82
	DConv-8	69.7	5.92	8.23
	DConv-16	70.8	5.92	8.75
Semantic branch-skip connection	No	71.5	5.85	7.61
	Yes	72.3	5.86	7.82

'DConv-8': Dilated convolution with rate = 8; 'fpt':forward pass time; 'ms':millisecond.

Table 5.4: Ablation study for the different number of branches in FHM on Cityscapes validation set.

Nos.	3rd/4th branch	Param (K)	Mem (MB)	mIoU (%)	FLOPs (G)	fpt (ms)
2	-/-	537.123	1450	69.5	5.71	7.30
3	APC/-	544.397	1554	72.3	5.92	7.82
3	DC8DC16/-	564.771	1864	72.6	6.23	8.94
4	DC8/DC16	578.595	1928	71.7	6.45	8.20
4	DC4DC8/DC8DC16	592.419	2304	74.1	6.76	9.63

'DC8': Dilated Conv with dilation = 8; 'APC': Average Pool+Conv 'fpt':forward pass time; 'ms':millisecond.

Context guidance: The benefit of context guidance in the attention module is analyzed by removing the encoder skip connections. Instead, the high-level feature input (X) is used in both the input branches of MSAM (refer Figure 5.4). The corresponding change required in MSAM is to replace C_1 in the 1×1 kernel of $\overline{X}$ branch to C_2. It reduces the context guidance branch to only channel attention. The module generates the attention weights mainly using the semantic information present in the high-level features. It is believed that low-level features are equally crucial for the model to learn finer details, such as boundaries and smaller classes while generating attention weights. Table 5.5 summarizes

Table 5.5: Ablation study for context guidance in MSAM on Cityscapes validation set.

Context-input	mIOU (%)	FLOPs (G)	Param (M)	Mem. (MB)	fpt (ms)
low-level	72.3	5.86	0.544	1554	7.82
high-level	70.5	5.90	0.576	1405	8.35

Mem.: GPU reserved memory in MegaByte (MB).

the result. Using only the high-level feature input reduces the accuracy by 1.8%. It also increases the number of parameters and FLOPs. Specifically, 3.2×10^4 parameters and 40 Million FLOPs. This is because high-level feature maps have more channels than low-level ones. However, a 9.5% reduction in memory is observed as the model does not require storing some low-level feature maps.

Figure 5.6 shows the gradient-weighted channel activation maps [123] of some of the classes for the two types of context input in MSAM. The channel activations of the last layer are used to generate the heat maps. The second column is for high-level context input, whereas the third is for low-level context input. The bounding box highlights that the proposed method emphasizes the important region critical to the target class. In the first row, MSAM improves the body region of the person class. For the car class, shown in the second row, the high-intensity region on the road is minimized in addition to improving car boundaries. Similarly, for bicycles, the large areas of the road are reduced in the third row. Traffic signals and poles are used as target classes in the fourth and last rows, respectively. In both of these sets of images, MSAM increases the focus on the target object rather than the surroundings.

Dilation rates in FHM: Dilation of the convolution kernel is quintessential in semantic segmentation to increase the FOV to capture the context information around the centre pixel. The FHM has two dilated kernels in the second branch 5.3 (d) with dilation d_1 and d_2. Six combinations of dilation rates are used for the ablation study. Table 5.3 reports the result. To begin with, no dilation is used in the FHMs. This achieves the lowest accuracy of 70.4%, reinforcing the importance of context features for accurate segmentation. Even (2,4,6,8,10,12) and odd (3,5,7,9,11) dilation rates are used for the next set of experiments. Following [46], dilation rates were gradually increased with the depth. In another experiment, only d_2 is varied. However, keeping a constant dilation rate of (2,4) gave the best accuracy of 72.3%. This is because FHM has an additional branch that extracts features by downsampling, along with two dilated and one undilated kernel. This scheme learns rich features at multiple scales, resulting in accurate segmentation.

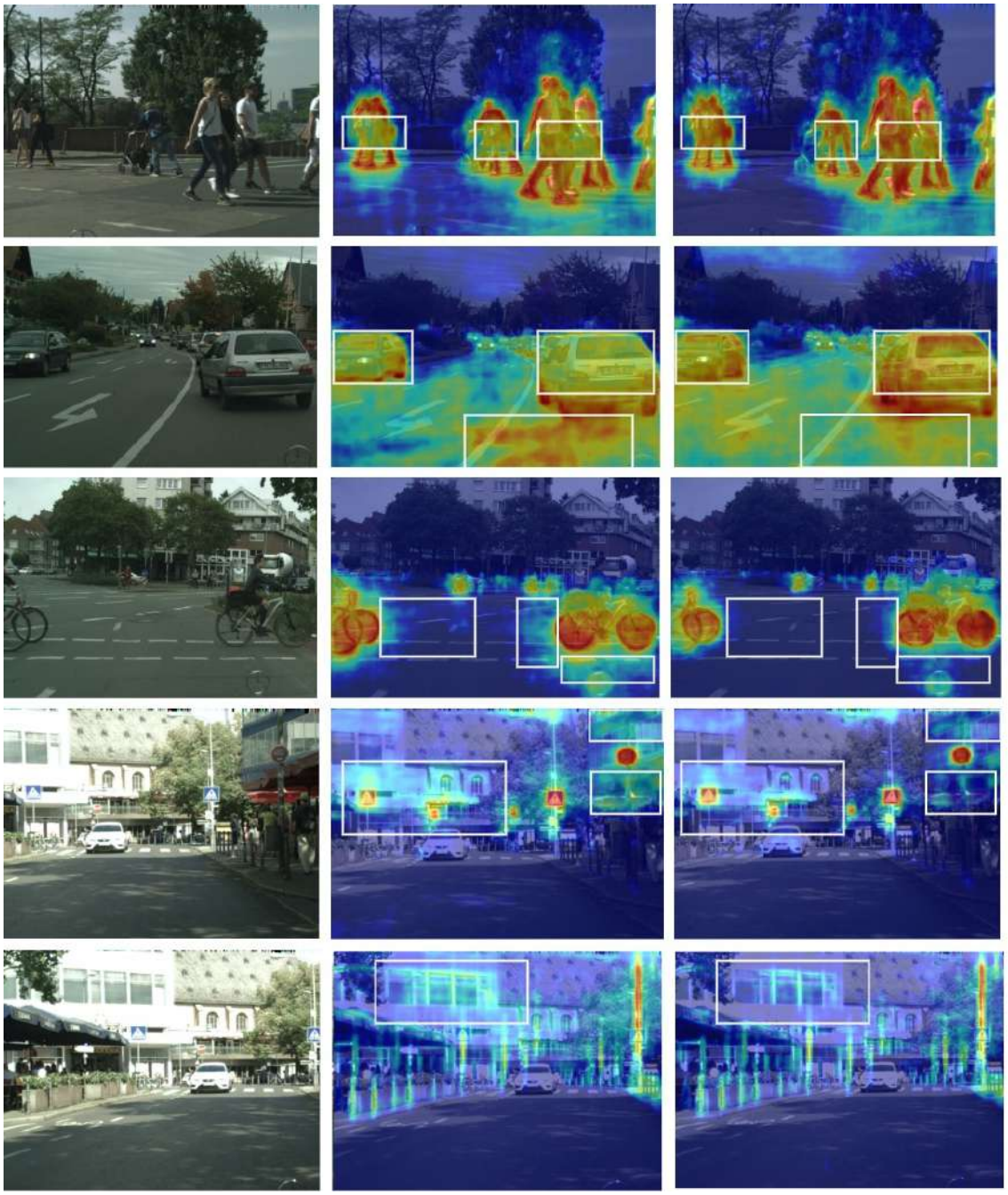


Figure 5.6: Gradient-weighted class activation maps (Grad-CAMs) of the last network layer for the person, car, bicycle, traffic signal, and pole classes (top to bottom). First column: input image; second column: CAM for high-level context input in MSAM; third column: CAM for low-level context input in MSAM. The critical regions are highlighted in the bounding box. Red and blue colors signify the highest and lowest intensities, respectively.

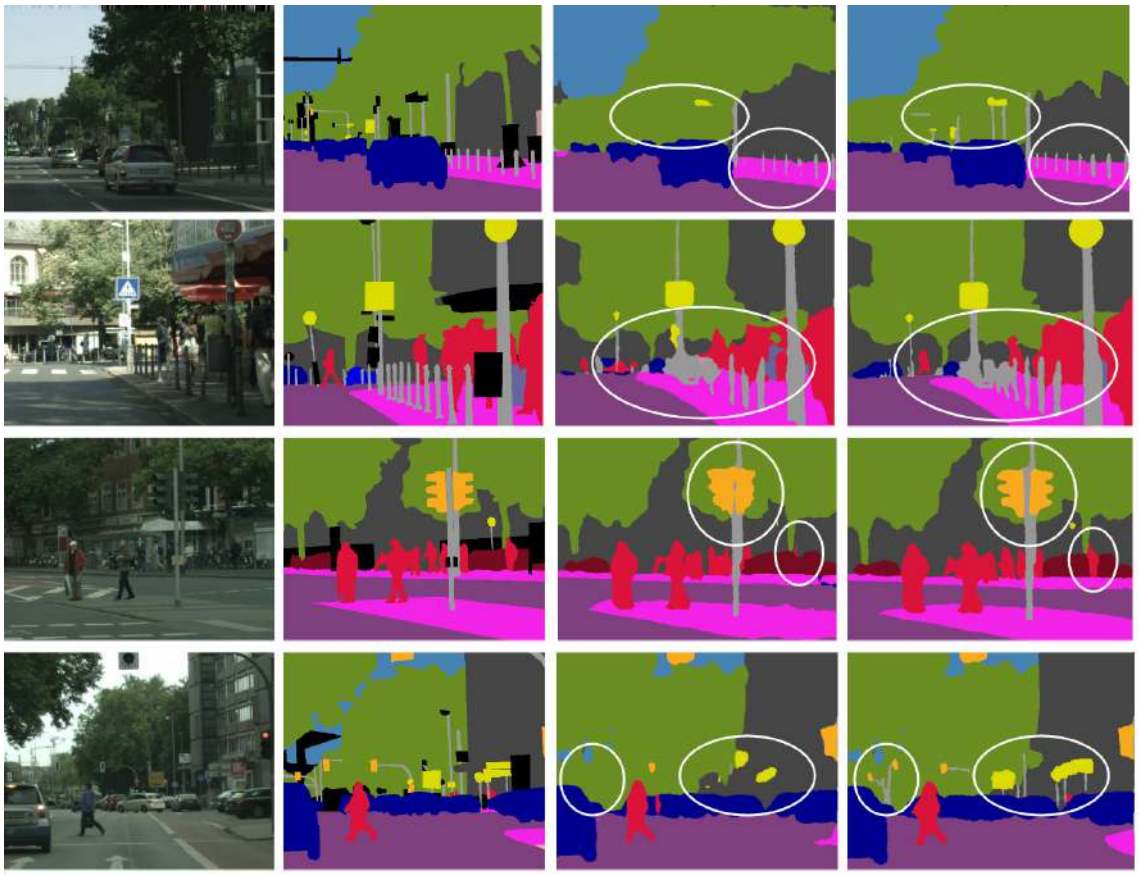


Figure 5.7: Qualitative results on the Cityscapes showing the effectiveness of adding the Multi-Scale Attention Modules in the CGMA-Net. Column 1: Input images; Column 2: Ground truth; Column 3: Output without MSAM; Column 4: Output with MSAM (1,3,6). The segmentation improvement over smaller objects is highlighted.

Encoder depth: To see the effect of encoder depth on the performance of the network, we change the numbers of FHMs in various stages. It can be seen that increasing the number of FHMs in the second stage has a less prominent effect on speed and FLOPs than in the first stage. This is because the feature maps in the first stage are double the size of the second stage. Using only two FHMs in Stage 1 leads to inferior performance compared to using three modules, which shows the importance of effective low-level feature representation learning. The combination of three and six FHMs in Stages 1 and 2, respectively, achieves the optimum performance in terms of accuracy while balancing the speed and computation requirement.

Decoder: Two experiments were performed as a part of the decoder ablations. In the first, the decoder is removed, and only the backbone is evaluated against the Cityscapes dataset. In the second, a full encoder-decoder network is trained with different MSAM scales. Table 5.8 shows that the inference speed gets an 18% boost without the decoder.

Table 5.6: Ablation study for Dilation rates in FHM on the Cityscapes validation set.

Dilation rates ($\mathbf{L_5d_1}$,$\mathbf{L_5d_2}$) + ... + ($\mathbf{L_{14}d_1}$, $\mathbf{L_{14}d_2}$)*	**mIOU** (%)
No dilation	70.4
(2,4)+(2,4)+(2,4)+(2,4)+(2,4)+(2,4)+(2,4)+(2,4)	72.3
(1,1)+(1,1)+(2,4)+(1,1)+(1,1)+(2,4)+(4,6)+(6,8)+(8,10)	71.1
(1,1)+(1,1)+(1,1)+(1,1)+(1,2)+(1,4)+(1,6)+(1,8)+(1,10)	71.7
(1,1)+(1,3)+(1,5)+(1,1)+(1,1)+(1,3)+(1,5)+(1,7)+(1,9)	71.2
(1,1)+(2,3)+(4,5)+(1,1)+(1,1)+(2,3)+(4,5)+(6,7)+(8,11)	72.0

* $L_i d_j$ represents FHM in layer-i with dilation-j. FHMs are located in layers 5-7 and 9-14.

Table 5.7: Ablation study for encoder depth by changing the number of FHMs in various stages (S1/S2) on Cityscapes validation set.

S1/S2	**FLOPs (G)**	**Param (M)**	**mIoU %**	**fpt** (ms)
2/5	5.16	0.510	69.7	6.87
2/6	5.37	0.535	70.5	6.99
2/8	5.79	0.583	70.6	7.25
3/5	5.71	0.526	71.9	7.76
3/6	5.86	0.549	72.3	7.82
3/8	6.34	0.599	72.2	8.03
3/10	6.76	0.648	73.1	8.25

However, it also leads to a 3.3% decrease in accuracy. The qualitative results in Figure 5.7 show the effectiveness of MSAM. The network output is improved around small objects and boundaries, highlighted in the images. Three combinations have been selected for the MSAM scales. A scale of 1 implies no scaling. Combining (1,3,6) gives the best accuracy with intermediate inference speed among the three sets. (1,6,9) is the fastest because of the smaller number of patches (a larger patch scale size means a smaller number of total patches). However, it has slightly lower accuracy. (1,2,3) performed poorly in accuracy because of less scale variation, leading to similar feature learning instead of multi-scale features. Reducing the number of scales from three to two led to a significant drop in accuracy, specifically by 2.1%. We also experimented with four scales (1,3,6,9), but it achieved similar accuracy to three scales; rather, it increased the memory and FLOPs, thus showing that a choice of three scales is sufficient for this context.

5.3.3 Comparison on the Cityscapes benchmark

The proposed method is compared with the state-of-the-art methods on the Cityscapes benchmark in Table 5.9. For a fair comparison, the fps are calculated on a common

Table 5.8: Ablation study for decoder on Cityscapes validation set.

Ablation Study	Type	mIOU (%)	fpt (ms)	Mem. (MB)	FLOPs (G)
Decoder	No	69.0	6.62	1410	4.77
	Yes	72.3	7.82	1554	5.86
MSAM scales	(1,2)	70.2	7.13	1480	5.40
	(1,2,3)	70.9	8.20	1546	5.95
	(1,3,6)	72.3	7.82	1554	5.86
	(1,6,9)	72.1	7.67	1500	5.81
	(1,3,6,9)	71.9	8.45	1631	6.18

platform. The table is divided into two sections. The first section reports the accuracy-oriented method based on transformers [62]. These methods have large sizes and require a large number of FLOPs. Consequently, they require large memory and power. The highest accuracy of 84.9% is reported by ViT-Adapter-L [105]. However, with 347.9 Million (M) parameters, it becomes sufficiently large for resource-constrained devices such as autonomous vehicles and robotics. The other two transformer-based methods, Trans4trans [65] and SegFormer [102], have the lowest Efficiency Index (EI) value among the compared methods due to the size and FLOPs, despite having high accuracy. The second section compares smaller methods targeting real-time deployment. These methods are based on highly efficient CNNs for vision tasks. DDRNet [59] has the highest accuracy of 77.4% but has 7× more FLOPs and 10× more size than the proposed method. Moreover, its accuracy can be attributed to the ability to process full-resolution (1024×2048) images. HyperSeg [30] has comparable FLOPs and is more accurate by 2%. However, it has 20× more parameters, thus having a low EI of 1.397. MLFNet [57] has lower FLOPs and comparable accuracy and speed but has more parameters. Specifically, it has 7.5× more parameters. CGNet [73] has a comparable size but has 50% lower fps and 9% less accuracy despite having fewer parameters. MFNet [72] has similar accuracy to the proposed method and is 17 fps faster; nevertheless, it has 2.5× more parameters. The proposed method achieves the highest EI score of 2.596, efficiently balancing the size, accuracy, and speed trade-offs. It also has one of the lowest FLOP requirements of 5.86 G, making it suitable for low computation power devices.

5.3.4 Comparison on the CamVid benchmark

The proposed method is compared with state-of-the-art networks on the CamVid dataset in Table 5.10 to show the effectiveness and generality. The most accurate method is RTFormer [110], which is based on the transformers. Among the CNN-based methods,

Table 5.9: Comparison with state-of-the-art on the Cityscapes benchmark. Categorized by increasing efficiency index from top to bottom.

Method	Input Size	FLOPs (G)	FLOPs$^{\$}$ (G)	mIoU (%)	FPS$^{\#}$	FPS*	Param (M)	EI
ViT-Adapter-L [105]	896 × 896	-	-	**84.9**	< 1	-	347.9	-
SegFormer [102]	512 × 1024	17.7	17.7	84.0	2.5	47.6	84.7	0.827
Trans4trans [65]	768 × 1536	94.25	41.88	81.5	27.3	27.3	49.5	0.087
SDPT-Tiny [124]	512 × 1024	63.4	63.4	77.3	63.8	63.8	3.6	0.438
NDNet [125]	512 × 1024	26.8	26.8	76.5	161	161	18.7	0.483
ESS-SASD [80]	512 × 1024	117.26	117.26	71.79	41	70	11.38	0.109
Hyperseg-M [30]	512 × 1024	7.5	7.5	75.8	36.9	52	10.1	0.775
LCFNet [126]	512 × 1024	7.28	7.28	76.58	171	183	7.28	0.991
PIDNet-S [8]	512 × 1024	23.79	23.79	75.5	116.5	116.5	7.6	0.689
SegTransConv [127]	512 × 1024	10.16	10.16	72.8	57.3	57.3	7	0.837
LBN-AA [35]	448 × 896	49.5	64.65	73.6	51	81.6	6.2	0.834
DDRNet-Slim [59]	1024 × 2048	35.78	8.94	77.4	108.8	155.8	5.7	0.985
BiSeNet [55]	768 × 1536	14.8	6.57	68.4	105.8	148	5.8	0.980
LAANet [74]	512 × 1024	19.2	19.2	73.6	95.8	134	0.67	0.997
MLFNet [57]	512 × 1024	4.67	4.67	71.5	90.8	118	3.99	1.137
FASSD-Net-L1 [61]	1024 × 2048	20.0	5	72.1	78.0	114	2.8	1.271
LMFFNet [7]	512 × 1024	16.7	16.7	75.1	118.9	118.9	1.4	1.011
IRDPNet [119]	768 × 1536	11.54	5.13	75.6	78.6	110	1.49	1.370
MFNet [72]	512 × 1024	9.1	9.1	72.1	116	145	1.34	1.221
LEDNet [101]	512 × 1024	11.44	11.44	70.6	71.0	86.5	0.94	1.137
FRNet [71]	512 × 1024	9.1	9.1	70.4	127	151	1.01	1.245
FDDWNet [116]	512 × 1024	12.5	12.5	71.5	60	72	0.80	1.110
DABNet [46]	1024 × 2048	10.60	10.60	70.1	27.7	60.6	0.76	1.160
RegSeg [70]	512 × 1024	9.78	9.78	77.6	86.7	86.7	3.34	1.057
EDANet [106]	512 × 1024	11.34	11.34	67.3	108.7	161	0.68	1.184
LCNet [118]	512 × 1024	15.9	15.9	73.3	185.2	185.2	0.51	1.106
CGNet [73]	512 × 1024	7.14	7.14	64.8	50	53	0.5	1.297
CGMA-Net (Proposed)	512 × 1024	5.86	5.86	73.4	128	128	0.54	**1.476**

'-' method does not report the result. '$' Normalised FLOPs for 512 × 1024 input size. '#' FPS reported in the literature. '*' FPS calculated on RTX 3090 GPU.

ours is among the top accuracy lightweight networks. Specifically, it is 1.2% less accurate than FASSDNet [61] but has 5× less parameters. LAANet [74] has comparable accuracy and size while being faster by 27 fps. However, it processes images at half resolution of 480 × 360. CGNet [73] has the smallest size with 0.5 M parameters. Nevertheless, it is slower by 26 fps and is 2.5% less accurate. These results prove that the proposed method has comparable performance with the state-of-the-art methods on CamVid, which is a comparatively small dataset.

5.3.5 Comparison on the BDD100K benchmark

To further validate the performance of the proposed method, we also provide the quantitative results on the challenging BDD100K [11] dataset in Table 5.11. The large size of DeepLabv3 [32] and a ResNet backbone allow it to perform better in terms of accuracy.

Table 5.10: Comparison with state-of-the-art on the CamVid benchmark.

Method	InputSize	mIoU (%)	FPS	Param (M)
RTFormer-Slim [110]	960 × 720	81.4	190.7	4.8
SwiftNet-MN [68]	960 × 720	65.0	27.7	11.8
DFANet [24]	960 × 720	59.3	116	4.8
ICNet [53]	960 × 720	67.1	46.7	7.8
BiSeNet [55]	960 × 720	65.6	175	5.8
BiSeNetV2-L [56]	960 × 720	73.2	32.7	-
MLFNet-Res34 [57]	960 × 720	69.0	57.2	4.0
LMFFNet [7]	960 × 720	72.0	116.4	1.4
LAANet [74]	480 × 360	67.9	112.5	0.7
EDANet [106]	480 × 360	66.4	40.75	0.7
CGNet [73]	960 × 720	65.6	59.0	0.5
FASSDNet [61]	960 × 720	69.3	80.0	2.85
DABNet [46]	480 × 360	66.4	101.3*	0.76
CGMA-Net (Proposed)	960 × 720	68.1	85.5	0.54

Table 5.11: Comparison with state-of-the-art on the BDD100K benchmark.

Method	Input Size	Backbone	mIoU (%)	Param (M)
DeepLabv3 [32]	1280 × 720	ResNet101	64.8	11.8
ContextNet [54]	1280 × 720	Scratch	44.5	0.85
MLFNet [57]	1024 × 512	MobileNetV2	53.5	3.99
LCNet [118]	1280 × 720	Scratch	58.5	0.74
SFNet-Lite [128]	1280 × 720	STDC	59.4	13.7
STDC [56]	1280 × 720	Scratch	52.1	12.0
CGMA-Net	1280 × 720	Scratch	54.2	0.54

The CGMA-Net achieves comparable accuracy with fewer parameters. Specifically, with only 0.54 M parameters, it surpasses the accuracy of both MLFNet [57] and STDC [56]. LCNet [118] achieves a better accuracy but with a noticeably larger size.

5.3.6 Comparison on the KITTI benchmark

Following [129] and [7], we validate the generalizability of the CGMA-Net by evaluating the predictions on the KITTI dataset using the model trained on Cityscapes. The results are shown in Table 5.12. CGMA-Net can achieve accuracy comparable to state-of-the-art methods with a fraction of FLOPs. Specifically, it is 2.2% more accurate than LETNet [117] with less than 50% FLOPs and model size. It outperforms methods such as [73, 46, 101, 47] by a significant margin. LMFFNet [7] achieves better accuracy by 3.6%. However, it requires 3× more FLOPs and has thrice the number of parameters than CGMA-Net.

Table 5.12: Comparison with state-of-the-art on the KITTI benchmark.

Method	Input Size	FLOPs (G)	mIoU (%)	Param (M)
CGNet [73]	1024 × 512	7.14	36.9	0.5
DABNet [46]	1024 × 512	10.6	37.2	0.8
LEDNet [101]	1024 × 512	11.44	40.3	1.0
ERFNet [47]	1024 × 512	26.9	42.9	2.1
LETNet [117]	1024 × 512	13.6	43.5	1.0
LMFFNet [7]	1024 × 512	16.7	49.3	1.4
CGMA-Net	1024 × 512	5.86	45.7	0.54

Table 5.13: Specifications of the mobile-GPU cards used for inference.

GPU Card	Cores	Speed	RAM	Mem.B/W	Power
Xavier	384	854MHz	8GB	51.2 GBps	15W
3060M	3840	817MHz	6GB	336 GBps	80W

'RAM': Random Access Memory; 'Mem.B/W': Memory Bandwidth.

5.3.7 Evaluation on GPU-Powered Embedded Systems

High-end GPU accelerators are unsuitable for embedded and edge devices due to resource limitations. Nvidia's Xavier and Jetson family GPUs, designed for such devices, have reduced resources. To ensure real-time performance, we assess the proposed network on the Jetson Xavier NX and RTX 3060-Mobile laptop. The technical details of these GPUs are given in Table 5.13. The official SDK for the Jetson with TensorRT acceleration is used. However, no such acceleration is used on 3060M for a fair comparison. Table 5.14 compares the method against the state-of-the-art for inference speed.

The proposed lightweight design excels in speed on both devices for input resolutions of 1024 × 512, outperforming most methods with comparable power consumption and using only 1554 MB GPU reserved memory. However, ENet is the fastest and most power efficient due to its shallow structure despite having low accuracy. Moreover, it consumes the least GPU memory of 940 MB. LEDNet is the slowest due to its complex decoder structure despite having only 0.94 M parameters. Fast-SCNN and ContextNet surpass the proposed method in speed but with lower accuracy. Specifically, Fast-SCNN is 2 fps faster, and ContextNet is 2.5 fps faster. MLFNet offers good accuracy but with 8× more parameters and at a slower fps.

The results in Table 5.14 show that most of the networks are unable to run in real-time (> 30 fps) on full-resolution images of the Cityscapes. However, an input resolution of 640 × 320 is enough to satisfactorily recognize any urban street scene [47]. The proposed

Table 5.14: Inference speed(fps) of the proposed method for different input image resolutions on mobile GPU-based systems. Full resolution: 2048 × 1024; half resolution: 1024 × 512.

Method	Jetson-Xavier		3060-Mobile		mIoU	Param	Mem	Pow
	Full	Half	Full	Half	%	(M)	(MB)	(W)
SwiftNet [68]	2.61	9.9	3.91	14.5	70.2	11.8	1721	8.2
ENet [50]	13.8	53.82	20.17	78.68	58.3	0.4	940	4.5
ContextNet [54]	10.49	40.91	15.33	55.2	66.1	0.85	1464	6.2
LEDNet [101]	0.7	2.73	1.21	4.43	70.6	0.94	3121	8.7
BiSeNet [55]	2.42	9.6	3.45	13.77	68.4	5.8	1903	7.4
Fast-SCNN [45]	11.49	40.21	14.1	53.6	68.4	1.14	1246	5.8
CaBiNet [58]	8.21	30.95	10.45	41.8	76.5	2.64	1279	6.5
FASSDNet [61]	7.3	29.2	10.79	38.85	76.0	2.85	2402	7.9
MLFNet [57]	8.41	32.79	11.2	43.7	71.5	4.0	2170	7.6
CGMA-Net (Proposed)	10.1	38.6	14.5	52.3	72.3	0.54	1554	7.8

network can work at a resolution of 1024 × 512 in real-time, with 38.6 fps and 52.3 fps on Jetson Xavier and 3060M. Thus, it is suitable for edge device deployment.

5.3.8 Qualitative Analysis of the Segmented Output Images

The proposed method has also been tested on real driving sequences captured at 1080p resolution in our city, as shown in Figure 5.10. While the network encounters challenges with certain objects in the scene, it demonstrates the importance of semantic and context information for accurate classification. An interesting example is image 'e', where a small board on a tree is misidentified as a traffic symbol on a pole. Despite minor misclassifications on small objects, the network excels in accurately identifying major classes like road, car, pedestrian, sidewalk, building, and sky, affirming its effectiveness.

The segmented images from Cityscapes and BDD datasets, as well as real-life road scenes, are analysed. Figure 5.8 compares outputs from the proposed network on the Cityscapes with CFPNet, chosen for its similar accuracy and size. Three key improvements have been highlighted in the segmented images using white bounding boxes. In the first image, the proposed method excels in delineating pedestrians (box 1), recognizing traffic signals (box 2), and enhancing the visibility of small poles (box 3). These improvements are consistent across various images.

Figure 5.9 displays the results of the method on the BDD dataset using a network trained on Cityscapes. The model performs well on large object classes such as cars, roads, buildings, sidewalks, trees, and vegetation, among others, but faces challenges with smaller classes due to the BDD dataset's diversity. One notable failure is marked with a red

bounding box, where a tower is misidentified as a building because it was absent in the Cityscapes training samples.

To show the performance of the proposed method in real life, we capture the driving sequence from a dashcam at 1080p resolution in and around our city. Figure 5.10 shows the results. For the same reason given in the previous para, the method fails for certain objects present in the scene. An interesting result can be seen in image 'e', highlighted in the left bounding box; a small board fixed to a tree is identified as a traffic symbol mounted on a pole. This practical example shows how a deep neural network requires semantic and context information for accurate classification. Nevertheless, despite some misclassification on tiny objects, the proposed network performs accurately on the major classes such as road, car, pedestrian, sidewalk, building and sky. Thus proving its efficacy.

Critical cases: The performance of the network on adverse weather conditions and night scenes is shown in Figure 5.11. Low visibility severely affects the scene segmentation capability of the network trained on vanilla datasets. To improve the performance, several image de-hazing and fog removal algorithms have been proposed using synthetic datasets such as Foggy Cityscapes [130] and Cityscapes rain [131]. For night-time enhancement, few authors have proposed domain knowledge distillation [132]. The integration of LiDAR data has also seen increasing usage recently to improve performance in adverse weather conditions [133].

5.4 Contributions

The major contribution of this paper can be summarised below:

- A fast hybrid module (FHM) is proposed with an inverse-residual bottleneck structure to capture local and context information. It is designed with depth-wise separable and point-wise convolution filters to minimise the number of parameters.
- A multi-scale attention module (MSAM) is proposed to enhance the feature learning ability of the network. It uses the low-level features to generate attention weights for the high-level features at various scales. Thus, the attention is 'context guided' and 'multi-scale'.
- A lightweight real-time network, CGMA-Net, is designed with FHMs and MSAMs, which obtains state-of-the-art accuracy on the challenging Cityscapes and CamVid datasets for semantic segmentation. Moreover, CGMA-Net obtains the trade-off

between accuracy and model size with one of the lowest computational requirements of 5.86 FLOPs.

- In addition, a new metric, *Efficiency Index* (EI), is proposed to measure the performance of lightweight and real-time networks in terms of accuracy, speed, size and FLOPs. It considers all the important factors that dictate the deployment of networks on resource-constrained edge devices.

5.5 Summary

This paper proposes a lightweight network, CGMA-Net, for real-time semantic segmentation of road scenes in resource-constrained devices. An optimal trade-off between size, accuracy and speed is obtained at a very low computational requirement. For this purpose, the FHM and MSAM blocks have been proposed. The FHM acquires local and context information at the module level by leveraging dilated convolutions and downsampling operations in a multi-branch structure. The MSAM generates context-guided attention weights at multiple scales for high-level feature learning. Through an elaborate set of experiments on highly competitive Cityscapes, CamVid, BDD100K, and real-life scenes, the performance of CGMA-Net has been shown. Furthermore, the experiments on low-power embedded devices/GPUs such as Jetson Xavier NX and 3060M show that the proposed network can achieve real-time performance.

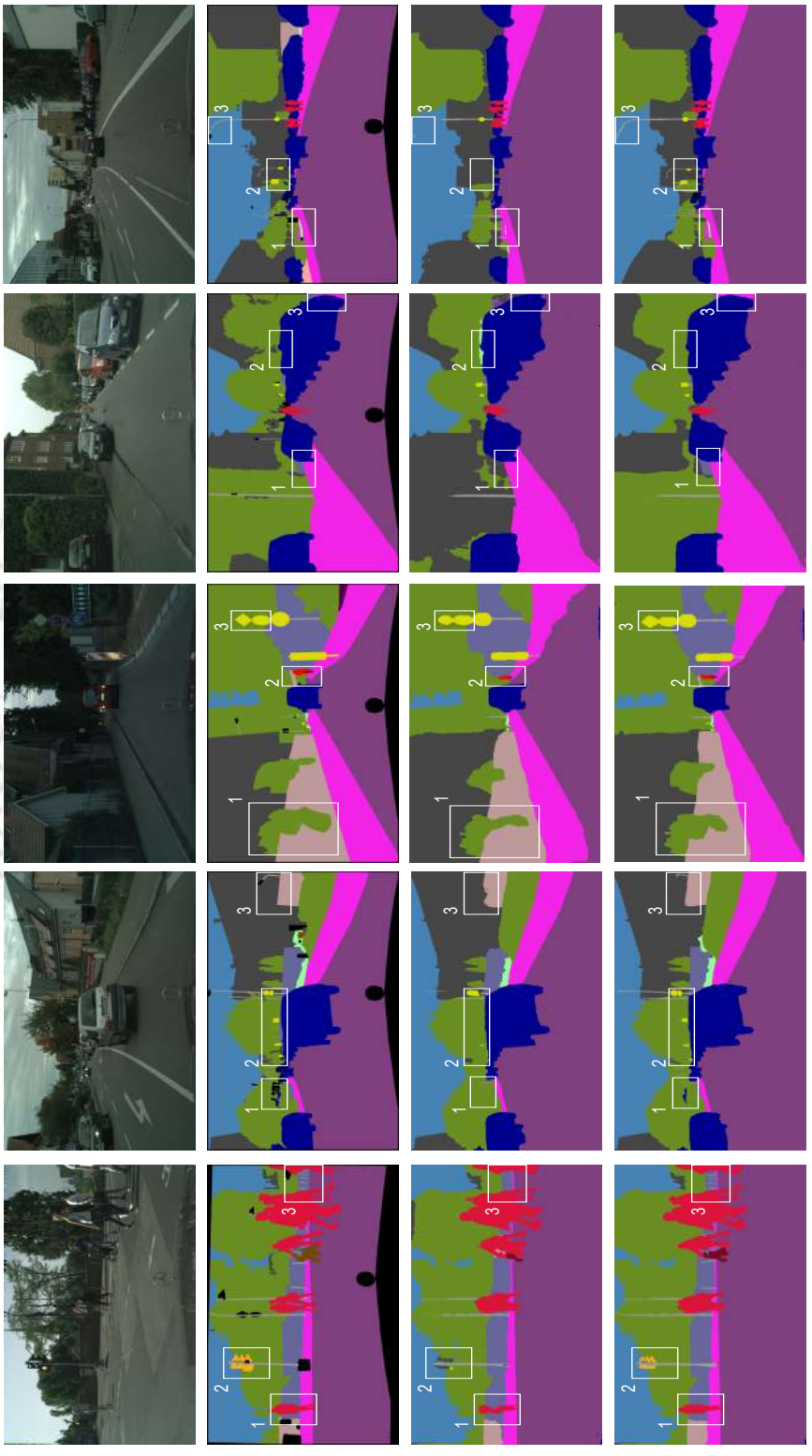


Figure 5.8: Qualitative results on the Cityscapes validation set. Row 1: Input image; row 2: Ground truth; row 3: CFPNet output; row 4: CGMA-Net output. The critical regions have been highlighted in the bounding box.

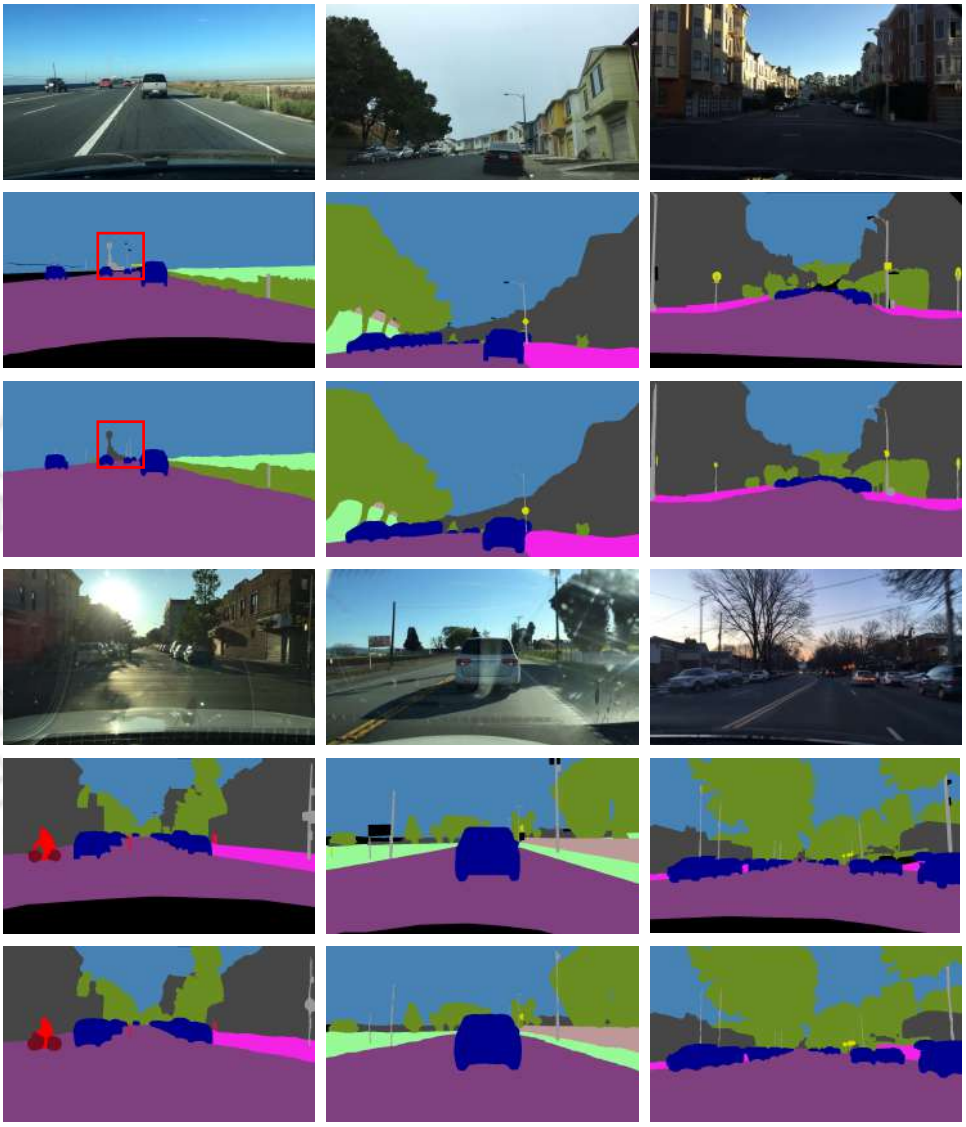


Figure 5.9: Qualitative results of the proposed network on BDD dataset. Row 1 & 4: Input images; row 2 & 5: Ground truths; row 3 & 6: Segmented outputs. Input images: 'a' to 'f' (clockwise from left-top). The critical failure region is highlighted in the red bounding box.

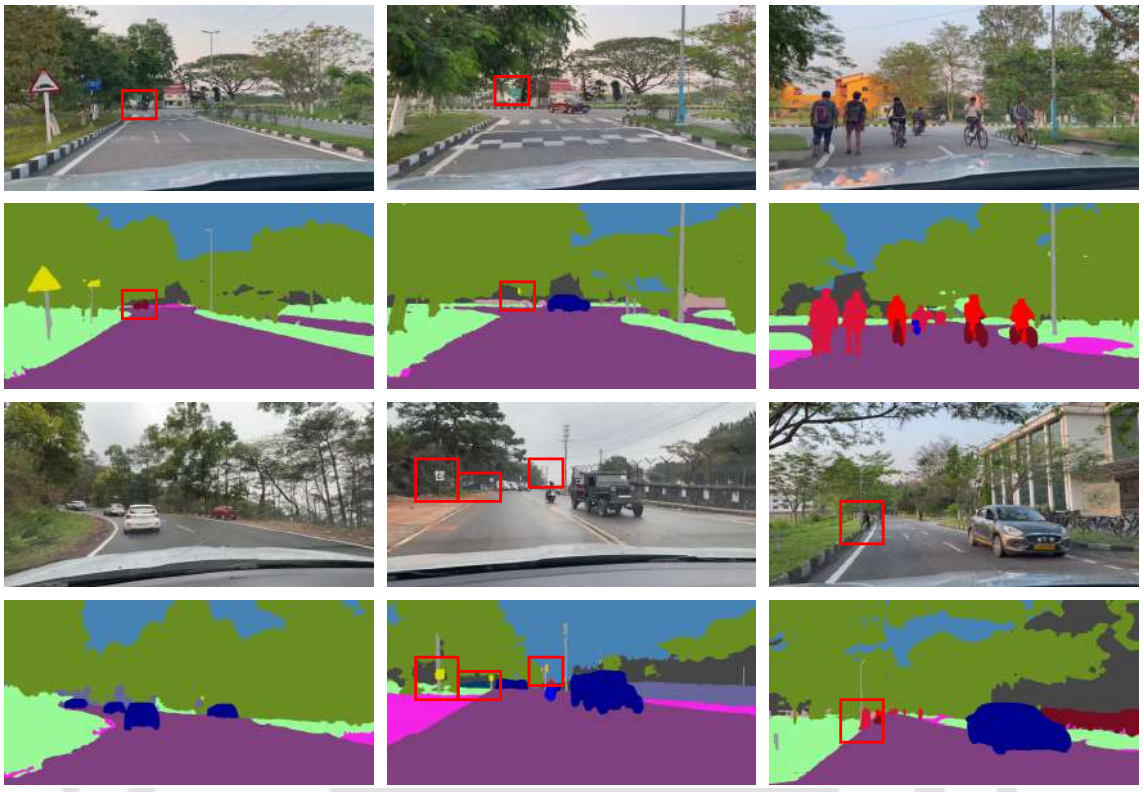


Figure 5.10: Qualitative results of the proposed network on real-life captured road scenes. Row 1 & 3: Input images; row 2 & 4: Segmented outputs. Input images: 'a' to 'f' (clockwise from left-top). The critical failure region is highlighted in the red bounding box.

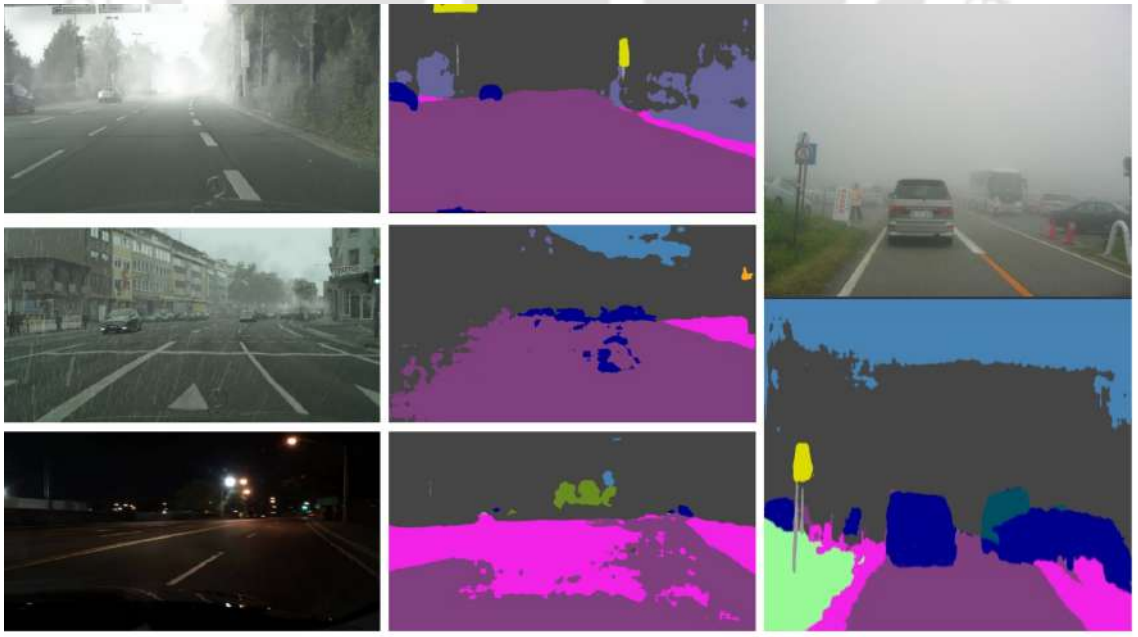


Figure 5.11: Qualitative results of the proposed network on some adverse scenes such as foggy, rainy and night.

Chapter 6

HCINet: A Hierarchical Approach of Context Integration in Real-time Semantic Segmentation

6.1 Introduction

The previous chapters have followed a single-path network architecture. Starting from a simple design with no bells and whistles in IRDPNet, BANet exploited the attention mechanism, followed by the CGMANet, which extensively relied on multiscale attention in the decoder to capture the long-range dependencies and global context. Apart from single-path architecture, the other prevalent design is the multi-path network, in which different branches operate at different feature map sizes to effectively learn local semantics and contextual features at multiple scales. In DDRNet [59], BiSeNet [55], STDC [134] and ContextNet [54], the input image and the resulting extracted feature tensors are processed parallelly in two different branches of different resolutions and depth. The deep branch results in the final feature tensor with resolution 1/32 or 1/64 to that of the input image. This deep branch increases the resultant receptive-field and is responsible for high-level semantic information extraction. The shallow branch, on the other hand, retains the final resolution to 1/8 and is responsible for extracting low-level spatial and local details. The output features from the deep and the shallow branches are then fused to preserve both local spatial and global contextual details in the final segmentation map. Apart from two-branched approach, there also exist methods (such as HRNet [60]) that use more branches for simultaneous extraction of multi-resolution features. While this strategy significantly

improves the accuracy, it increases the network size and memory requirement as the feature tensors in different branches are retained till the final prediction stage. This causes memory bottleneck during the practical implementation on a resource-constrained platform. To solve the above mentioned issues, our proposed network, Hierarchical Context Integration Network (HCINet), is specifically designed to handle multiple branches efficiently. More specifically, the feature tensors produced in different branches stay for a very short duration of time until they are merged with the features of the next branches in a hierarchical fashion as shown in Figure 6.1. This makes only two branches functional at a time and thus significantly eases up the memory requirement. In addition, the repetition rates of the basic blocks in different branches are kept small to limit the number of parameters; allowing us to achieve high accuracy in a parameter efficient way.

6.2 Proposed Method

In this section, we first briefly introduce our network architecture, followed by its functional blocks. In the end, we give complete detail of the proposed model. The overview of our network can be seen in Figure 6.1. It has an asymmetric encoder-decoder design, with a shallow-lightweight decoder to maintain the inference speed. The encoder is four-branched following HRNet, each branch made by the repetition of the basic block. However, it does not retain the branches throughout to minimise the memory requirements. Each branch process feature maps at a different resolution, thus leading to multi-scale learning. The learned features from each branch are downsampled through our Efficient Downsampling Block (EDB) before fusion with the subsequent branch. The encoder outputs from two branches are upsampled through the decoder before the final projection. We discuss the functional blocks in the following sub-section.

6.2.1 Basic Block

Our basic block is shown in Figure 6.2 (a). It has four asymmetric convolution kernels of size 3 to reduce the number of parameters. While the first two are non-dilated, the next two are dilated with a rate of '2'. The dilation is required to increase the field-of-view of the kernel for context aggregation. Our basic block design is based on ERFNet; however, we do not use bottlenecks. This gives better accuracy, albeit with a slight increase in channels. We also add a residual connection facilitating the gradient flow.

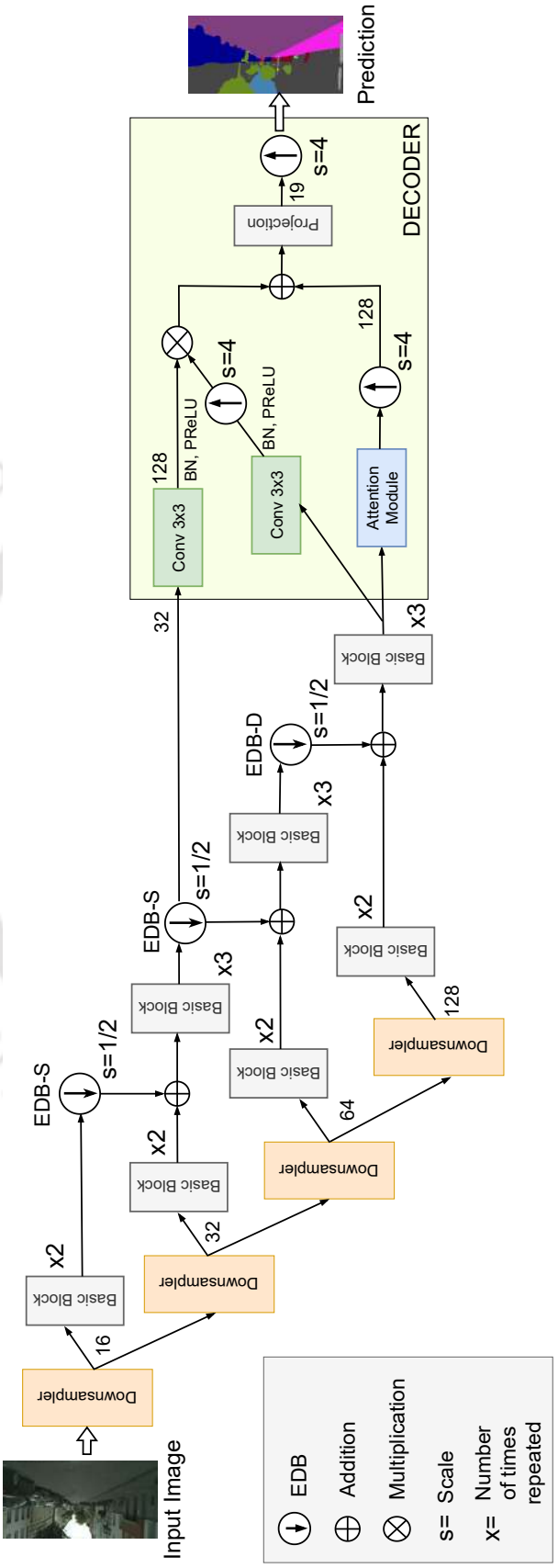


Figure 6.1: The network architecture of the proposed HCINet. Context extracted at two different scales in two different branches are merged hierarchically to serve as the context computation of the next branch. The application of context guidance and attention mechanism in the decoder result in preservation of finer details in the final segmentation map.

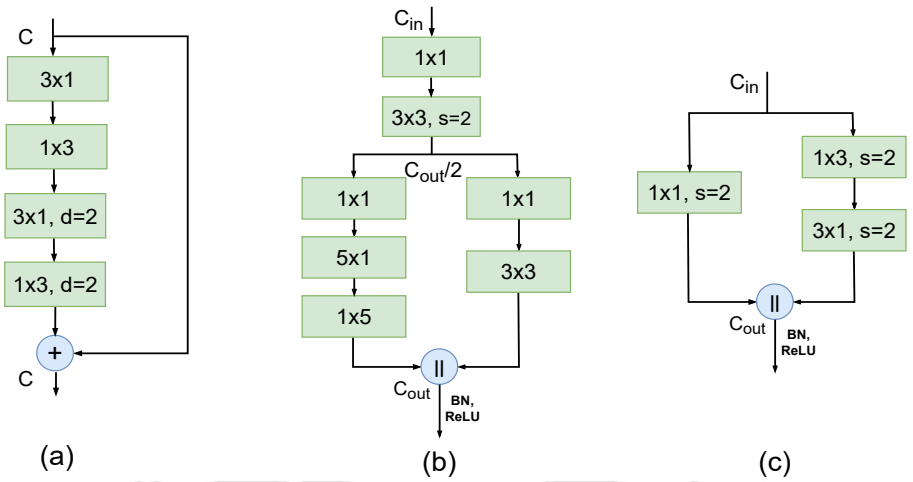


Figure 6.2: Illustration of the functional blocks: (a) Basic block; (b) Efficient Downsample Block-Shallow (EDB-S); (c) Efficient Downsample Block-Deep (EDB-D).'+': Elementwise addition; '||': Concatenation; 'C': number of channels; '3×1':kernel size.

6.2.2 Efficient Downsample Block

The EDB is shown in Fig. 6.2. We designed two variants of the EDB, shallow(S) and deep(D). While the shallow block is used in the early stage, the deep block is used in the later stage as the increase in channel dimension will increase the computation requirements. In EDB-S, the input undergoes channel reduction through a pointwise convolution operation, followed by a 3×3 convolution with a stride of '2'. This kernel is responsible for reducing the feature map dimension by half. The features are further processed in two parallel branches with kernels of varying sizes. While the context branch has a kernel of '5', the semantic branch has '3' as its size. This essentially helps in better feature learning capability. The outputs from both branches are concatenated, followed by batch normalization and ReLU non-linearity. The design of EDB-D is lightweight and has only three convolution layers, as shown in Figure 6.2(c).

6.2.3 Other Blocks

Downsampler: Our downsampler is shown in Figure 6.3(a). It is based on ENet; however, we have factorized the dense 3×3 kernel into asymmetric ones to reduce the parameters. The max-pooling operation is also used in the downsampler to retain low-level information. The outputs are finally concatenated.

Attention Block: We design a simplified channel attention block as represented in Figure 6.3(b). The attention block weighs the channels to suppress redundant information.

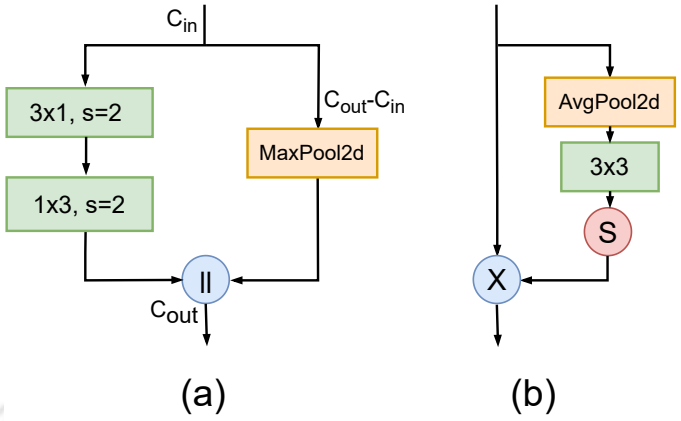


Figure 6.3: Illustration of functional blocks. (a) Downsampler; (b) Simplified Attention Module. 'S': Sigmoid; 's': stride.

To calculate the weights, we first average pool the feature maps, followed by a 3×3 convolution. These weights are learned automatically during backpropagation. A sigmoid layer is used to remove the negative weights. As a final operation, the channels are multiplied by the weights.

6.2.4 Network Architecture

The main motivation of our work is to design a network with the best possible accuracy and size trade-off. The network details are given in Table 6.1. As stated in the previous sections, our network follows a multi-pathway design with four branches. Each branch leverages the efficiency of the basic block and EDB, starting with a downsampler. Feature maps from a branch are added to the next branch through the EDB; the fused features are then jointly learned through subsequent basic blocks. For accurate segmentation, both global context and local semantic information is required. To fulfil this need, we take a long-range skip connection from a higher resolution branch which contains the context information. The semantic details are retained in the low-resolution deep branch. In the decoder, we use an attention module in the semantic branch and guided upsampling of the context features through a cross-connection. This technique improves the overall accuracy of the network. The features from both branches are finally added before projection. We use a 1×1 convolution to project from higher dimension feature space to the number of classes, which in our case is 19. The resolution is finally restored through bilinear upsampling.

Table 6.1: Detailed structure of the proposed method

	Layer	Operation	Mode	Channel	Output size
Encoder	1	Downsampling	stride=2	16	256x512
	2-3	Basic Block×2	d=2	16	256x512
	4	EDB-S	stride=2	32	128x256
	5	Downsampling	stride=2	32	128x256
	6-7	Basic Block×2	d=2	32	128x256
	8-10	Basic Block×3	d=2	32	128x256
	11	EDB-S	stride=2	64	64x128
	12	Downsampling	stride=2	64	64x128
	13-14	Basic Block×2	d=2	64	64x128
	15-17	Basic Block×3	d=2	64	64x128
	18	EDB-D	stride=2	128	32x64
	19	Downsampling	stride=2	128	32x64
	20-21	Basic Block×2	d=2	128	32x64
	22-24	Basic Block×3	d=2	128	32x64
Decoder	25	3×3 DWConv	stride=1	128	128x256
	26	3×3 Conv	stride=1	128	32x64
	27	Upsampling	scale=4	128	128x256
	28	Attention Module	-	128	128x256
	29	Upsampling	scale=4	128	128x256
	30	1x1 Conv	stride=1	19	128x256
	31	Upsampling	factor=4	19	512x1024

DWConv: Depthwise convolution; d:dilation rate.

6.3 Experimental Results

We test the proposed network on the Cityscapes [1] and CamVid [12] datasets. We briefly introduce these datasets and the implementation details and evaluation metrics. Finally, we compare the proposed network with the state-of-the-art methods.

6.3.1 Implementation Details

We implement the network using PyTorch v1.11 deep learning framework on a system with Nvidia-V100 GPU. Adam optimizer with an initial learning rate of 5×10^{-4} and a polynomial schedule with weight decay of 10^{-4} is used. For data augmentation, random crops, flips and translations are used. The crop size is fixed at 512×1024 for Cityscapes, while the original resolution is used for CamVid. A scale is randomly selected from [0.75, 2.0]. Training is done for 400 epochs with a batch size of 16.

Table 6.2: Ablation study for MCAP module on Cityscapes validation set.

Ablation Study	Value	mIOU) (%)	FPS	Params (M)
Basic Module	(2,3)	74.8	82.4	1.65
	(3,5)	73.2	63	2.1
	(5,7)	74.1	54	2.83
Dilation rates	2	74.8	82.4	1.65
	5	74.1	80	1.65
	7	73.7	79.2	1.65
	9	73.8	78.6	1.65

6.3.2 Evaluation Metrics

To calculate the segmentation accuracy, we use the mean intersection-over-union score. The inference speed is measured in terms of frames per second (fps). The model size is reported in the number of parameters (Params) in Million (M).

6.3.3 Ablation Study

We perform ablation studies on the Cityscapes dataset to verify our network architecture and report the results on its validation subset. We use different combinations of basic blocks in each layer. Table 6.2 shows a combination of (2,3) where 2 is the number of basic blocks in the first layer and 3 in the last layer. Increasing the blocks to (3,5) and (5,7) did not improve the accuracy. Specifically, the accuracy and fps were decreased by 1.6% and 19. We also experimented with the dilation rate of the asymmetric convolution in the basic block. A dilation of 2 returned the best accuracy. Increasing the rate to 5, 7, and 9 reduced the accuracy and slowed the speed due to the large effective kernel size.

6.3.4 Comparison on the Cityscapes benchmark

In this sub-section, we report the performance of our method on the Cityscapes benchmark along with a comparison with other state-of-the-art lightweight real-time methods. Table 6.3 shows the comparison result. The accuracy-oriented methods are mostly based on the transformers [105, 62, 65]. They are large sizes and require high FLOPs, thus unsuitable for resource-constrained devices. Lightweight models are based on CNNs and are given in the second part of the table. Our method is more accurate than most of the works except Hyperseg [30] and DDRNet [59]. However, DDRNet needs 3× more FLOPs and 4 million more parameters than ours. Hyperseg is 6× larger than the proposed model. Both

Table 6.3: Comparison with state-of-the-art on the Cityscapes benchmark. Categorized by decreasing network size from top to bottom.

Method	FLOPs (G)	InputSize	mIoU (%)	FPS	Params (M)
ViT-Adapter-L [105]	-	896 × 896	**84.9**	< 1	347.9
PSP Net [67]	453.6	1024 × 2048	78.4	< 1	65.6
Trans4trans [65]	94.25	768 × 1536	81.5	27.3	49.5
SegFormer-B5 [102]	1447.60	1024 × 1024	84.0	2.5	84.7
Hyperseg-M [30]	7.5	512 × 1024	75.8	36.9	10.1
LBN-AA [35]	49.5	448 × 896	73.6	51	6.2
DDRNet-Slim [59]	35.78	1024 × 2048	77.4	108.8	5.7
BiSeNet [55]	14.8	768 × 1536	68.4	105.8	5.8
FASSD-Net-L2 [61]	45.1	1024 × 2048	72.1	133.1	2.3
ERFNet [47]	21.0	512 × 1024	68.0	41	2.1
ESNet [48]	27.49	512 × 1024	69.1	63	1.66
MLFNet-MobileV2 [57]	4.67	512 × 1024	71.5	90.8	3.99
AGLNet [75]	13.88	512 × 1024	70.1	52	1.12
LEDNet [101]	11.44	512 × 1024	70.6	71.0	0.94
DABNet [46]	10.60	1024 × 2048	70.1	27.7	0.76
HCINet (Ours)	11.15	512 × 1024	74.5	82.4	1.65

"-" indicates that the method does not report the result.

BiSeNet [55] and FASSDNet [61] are faster but are less accurate and larger; this is due to the more use of conventional 3×3 convolutions which are hardware optimised for faster inference.

6.3.5 Comparison on the CamVid benchmark

Table 6.4 shows the result of state-of-the-art methods on the CamVid dataset. Our method achieves 67.9% mIoU with 70 fps. FASSDNet achieves better accuracy and fps, however, with a larger size. Specifically, it is 1.5% more accurate and processes ten more fps. BiSeNetV2 [56] is highly accurate but runs at just 32.7 fps. Our method is more accurate than SwiftNet [68], MLFNet [57], DABNet [46] and DFANet [24]. This can be due to the multipath way architecture which leads to joint-feature learning at different scales in our method.

6.4 Contributions

The main contribution of this work can be summarized as follows:

- We propose a novel multi-path network to capture contextual information of different scales in different branches. More importantly, the hierarchical fashion in which the

Table 6.4: Comparison with state-of-the-art on the CamVid benchmark.

Method	InputSize	mIoU (%)	FPS	Params (M)
RTFormer-Slim [110]	960 × 720	81.4	190.7	4.8
DFANet [24]	960 × 720	59.3	116	7.8
SwiftNet-MN [68]	960 × 720	65.0	27.7	11.8
ICNet [53]	960 × 720	67.1	46.7	7.8
BiSeNet [55]	960 × 720	65.6	175	5.8
BiSeNetV2-L [56]	960 × 720	73.2	32.7	-
MLFNet-Res34 [57]	960 × 720	69	57.2	4.0
FASSDNet [61]	960 × 720	69.3	80	2.85
DABNet [46]	480 × 360	66.4	101.3*	0.76
HCINet (Ours)	960 × 720	67.9	70	1.65

*Denoted that the specific metric is calculated by us using the open-source code. "-" indicates that the method does not report the result.

context is extracted and integrated enable only two branches to be functional at a time; making the network memory efficient.

- A guided upsampling of the deep contextual information is proposed in the decoder to boost the final segmentation accuracy.
- We propose a novel Efficient Downsampling Block (EDB) which uses a combination of asymmetric and symmetric convolutions (in its deep variant) at different receptive fields. This enables extraction of multi-scale context during the downsampling operations.

6.5 Summary

In this work, we proposed a novel HCINet which effectively balances the accuracy-size trade-off while maintaining real-time speeds. Our network leverages the multi-scale and joint feature learning capability of the backbone structure. Our experiments demonstrate that the proposed method can achieve state-of-the-art performance on both Cityscapes and CamVid datasets in realtime.

Chapter 7

Conclusions and future work

This thesis investigated the challenges in designing and application of lightweight deep learning models for semantic segmentation in a resource-constrained environment. The key challenges are (1) low inference time and optimum memory utilization on edge devices (2) multiscale feature learning for effective segmentation (3) aggregation of local semantic and contextual information to mitigate the issue of feature loss due to repeated downsampling (4) effective utilization of attention mechanism in context of lightweight semantic segmentation. The work carried out in this thesis focused on designing various architectures and modules to solve the above challenges. In this chapter, the key observations and discussions are put forward, along with the possible directions for future work.

7.1 Conclusions

Chapter 1 serves as an introduction to the thesis, providing an overview of the key concepts such semantic segmentation in the context of autonomous driving, and the datasets utilized throughout the study. In addition to outlining the fundamentals, the chapter delves into the existing methods in the literature. The literature survey introduces the classical methods of semantic segmentation, followed by the prior deep learning based methods. Furthermore, it explores the underlying motivation driving the thesis and outlines the specific problem definitions that the thesis adresses. Through this comprehensive overview, the introduction sets stage for the subsequent chapters, providing context and framing the research objectives within the broader landscape of semantic segmentation.

The presence of many object classes and class imbalance in the road-scene images plague the accuracy of the deep-learning models. The second chapter addresses this issue by proposing a composite loss function inspired by bio-medical image segmentation. As accurate delineation of organs and their boundaries are vital for bio-medical images, researchers have proposed multiple custom loss functions to achieve this. We apply various region-based and boundary-based loss functions developed therein and combine them with the cross-entropy loss to develop a composite loss function, thereby increasing the accuracy. The empirical results demonstrate the significant improvement of segmentation over smaller classes by using the proposed composite loss function without any change in the existing network architecture.

Chapter 3 proposes a simple encoder-decoder method, inverse-residual dilation pyramid network (IRDPNet). It represents a significant advancement in lightweight semantic segmentation models. By adopting a redesigned basic block inspired by DABNet, IRDPNet improves segmentation accuracy while significantly reducing network size. The use of groups of depth-wise dilated convolutions with varying rates creates a spatial pyramid, enhancing computational efficiency and enabling the network to learn contextual features at multiple dilations. Additionally, the incorporation of an inverted residual structure with an expansion layer preserves crucial information during feature space reduction. Overall, IRDPNet achieves real-time performance with improved accuracy and less than half the parameters of DABNet. It addresses the challenge of reducing computational complexity without compromising performance with its simple architecture.

Accurate modelling of pixel relations and capturing long-range dependencies are pivotal for accurate segmentations of the objects present in the image. The block-attention network (BANet) proposed in Chapter 4 solves this problem by effectively integrating a self-attention mechanism in the basic block design. Unlike traditional self-attention mechanisms with high computational requirements, a modified attention module is proposed by without using the fully-connected layers. Moreover, the overall basic block is designed with multiple branches by stacking dilated convolutions to learn rich feature representations at every block level. In addition, a fast and lightweight decoder is proposed to fuse features from several encoder stages to further boost the segmentation accuracy. The study of the attention mechanism undertaken in this chapter and the methods used in Chapter 3 underlay the foundations of the algorithm proposed in the next chapter.

Context awareness enhances the model's ability to distinguish between similar objects in a scene and reduce confusion. High-level semantic features have often been exploited in non-local attention mechanisms to incorporate this. However, high-level semantic features

often lack spatial details crucial for precise boundary segmentation. Chapter 5 solves this limitation by proposing a context-guided multi-scale attention (CGMA) mechanism in which low-level spatially rich features guide the high-level semantic attention weight generation. The multi-scale design ensures capturing both fine details (like edges and small objects) and broader context (like background and larger objects), guiding the network to focus on contextually relevant areas across different resolutions. In addition a fast-hybrid module (FHM) is also proposed as a basic building block based on inverse-residual bottleneck design studied in Chapter 3. It effectively learns the local semantics and contextual features using variably dilated kernels. By integrating the proposed blocks into a deep network, state-of-the-art results were achieved on benchmark datasets.

These contributions represent a significant step forward in solving the critical challenges of real-time semantic segmentation for autonomous driving. The combination of advanced loss functions, lightweight architectures, and efficient attention mechanisms enables models to achieve high segmentation accuracy without exceeding computational limits. By addressing the inherent challenges of class imbalance, boundary precision, and computational constraints, these innovations pave the way for more reliable and intelligent perception systems in autonomous vehicles. Consequently, they contribute to the advancement of autonomous driving technology, allowing for safer and more efficient navigation in complex real-world environments.

7.2 Scope for future work

- Techniques such as model distillation in which a large teacher model is used to train a lightweight student model, can be further exploited to improve the performance of the proposed lightweight models.
- The temporal aspect of the driving sequence can be explored to incorporate details like optical flow to improve the model predictions further.
- Semantic segmentation backbone can be integrated into multi-task networks to solve multiple problems such as drivable area segmentation, trajectory planning, collision avoidance, and traffic sign recognition. Thus, a practical multi-task network can be developed in addition to the proposed methods.
- The qualitative results in the Chapters show that the misclassification of pixels majorly occurs at object boundaries and small classes. This issue, though mitigated to an extent through custom loss functions, still occurs as a major deterrent. Thus,

further techniques, such as filtering out the easily identifiable pixels, can be explored to improve the classification of these hard pixels.

- The thesis predominantly proposes lightweight CNN-based architectures for accurate semantic segmentation. However, the recent development of low-complexity and real-time segmentation transformer-based methods provides new insights. The capability of transformers to model the long-range pixel dependencies can be exploited in parallel to the CNNs to develop an efficient architecture.

Appendix A

Dataset Class Definitions

All major datasets for autonomous driving follow the semantic class definitions introduced by the Cityscapes. These definitions are carefully structured to represent the urban environments comprehensively. The semantic classes are organized hierarchically and grouped into broader categories to help distinguish between different types of objects or regions in the images. There are a total of 30 classes grouped into eight top-level categories as mentioned in Table A.1. To ensure consistency, the datasets use unique class IDs, RGB colour codes (refer Figure A.1), and semantic names for annotations. Classes within the void category are labelled as ignored during training. While comprehensive, any road scene dataset poses a class imbalance challenge because common classes dominate the labels, such as roads, buildings, sky, vegetation, and cars. Several classes, such as riding or training, are underrepresented.

Table A.1: Cityscapes object classes and categories.

Categories	**Classes**
flat	road, sidewalk, parking†, rail track†
human	person, rider
vehicle	car, truck, bus, on rails, motorcycle, bicycle, caravan†, trailer†
construction	building, wall, fence, guard rail†, bridge†, tunnel†
object	pole, pole group†, traffic sign, traffic light
nature	vegetation, terrain
sky	sky
void	ground†, dynamic†, static†

'†' indicates that the class is not used for model training and evaluation.

Figure A.1: RGB color ids for the segmentation classes.

Appendix B

Lightweight Convolutions

Asymmetric, depthwise separable, group, and pointwise convolution reduce a deep network's size and computational costs. For group convolution, the $k \times k$ filter kernel is divided into groups along the channel axis. This reduces kernel weights and multiply-accumulate operations in the hyperparameter order. Figure B.1 shows an example of group convolution for an input feature map size of $Height \times Width \times Channels(= 6)$. A value of 3 represents a three-fold reduction in kernel weights.

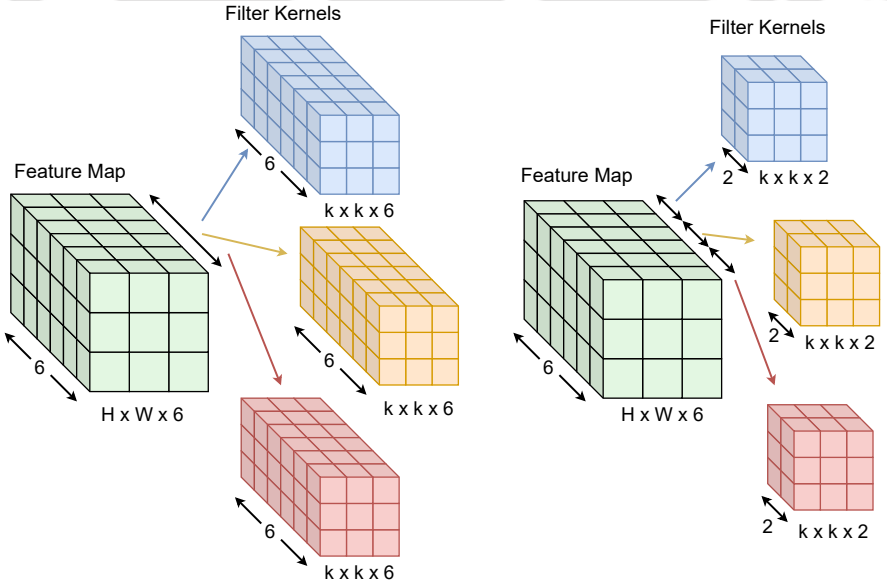


Figure B.1: Illustration of group convolution. The feature map size is $H \times W \times 6$, where 6 is the number of input channels. An output channel size of 3 requires three $k \times k$ filter kernels of channel depth 6, each, in the case of conventional convolution (left). Whereas in group convolution, for 'g'=3, the filter kernel depth reduces to 2 each (right).

Table B.1: Filter parameter count for various convolutions (Conv.)

Conv. type	Kernel size	Parameters
standard	$k \times k$	$k^2 C_{in} C_{out}$
groupwise	$k \times k$	$k^2 C_{in} C_{out}/g$
pointwise	1×1	$C_{in} C_{out}$
asymmetric	$k \times 1$	$k C_{in} C_{out}$

Depthwise separable convolution decomposes a Conv2d kernel into a group convolution kernel followed by a pointwise convolution. The number of groups, g, is equal to the number of input channels. So, each channel of $H \times W$ feature map is multiplied by a single filter kernel of size $k \times k$. For pointwise convolution, a 1×1 filter kernel is used instead of the standard $k \times k$ kernel. Table B.1 presents the total parameter count of these factorized kernels. It shows that pointwise convolution is the lightest for a given input and output channel value. However, due to its sparsity, the hardware kernel calls for pointwise convolution are not as optimized as other convolutions.

Bibliography

[1] M. Cordts, M. Omran, S. Ramos, T. Scharwächter, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset," in *CVPR Workshop on The Future of Datasets in Vision*, 2015.

[2] S. Mei, K. Yan, M. Ma, X. Chen, S. Zhang, and Q. Du, "Remote Sensing Scene Classification Using Sparse Representation-Based Framework With Deep Feature Fusion," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 5867–5878, 2021.

[3] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *IEEE Transaction on Pattern Analysis and Machine Intelligence(PAMI)*, vol. 39, (USA), p. 640–651, Apr. 2017.

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

[5] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object Detection With Deep Learning: A Review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[6] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," in *3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 730–734, 2015.

[7] M. Shi, J. Shen, Q. Yi, J. Weng, Z. Huang, A. Luo, and Y. Zhou, "LMFFNet: A Well-Balanced Lightweight Network for Fast and Accurate Semantic Segmentation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 6, pp. 3205–3219, 2023.

[8] J. Xu, Z. Xiong, and S. P. Bhattacharyya, "PIDNet: A Real-time Semantic Segmentation Network Inspired by PID Controllers," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Los Alamitos, CA, USA), pp. 19529–19539, IEEE Computer Society, jun 2023.

[9] S. Li, Q. Yan, W. Shi, L. Wang, C. Liu, and Q. Chen, "HoloParser: Holistic Visual Parsing for Real-Time Semantic Segmentation in Autonomous Driving," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–15, 2023.

[10] OpenAI, "ChatGPT [Large language model]," 2024.

[11] F. Yu, H. Chen, X. Wang, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition(CVPR)*, pp. 2636–2645, 2020.

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

[13] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[14] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[15] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 679–698, 1986.

[16] R. Nock and F. Nielsen, "Statistical region merging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1452–1458, 2004.

[17] L. Grady and E. Schwartz, "Isoperimetric graph partitioning for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 3, pp. 469–475, 2006.

[18] G. Coleman and H. Andrews, "Image segmentation by clustering," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 773–785, 1979.

[19] X. Zheng, Q. Lei, R. Yao, Y. Gong, and Q. Yin, "Image segmentation based on adaptive K-means algorithm," *EURASIP Journal on Image and Video Processing*, vol. 68, no. 1, 2018.

[20] A. Bleau and L. Leon, "Watershed-Based Segmentation and Region Merging," *Computer Vision and Image Understanding*, vol. 77, no. 3, pp. 317–370, 2000.

[21] H. Noh, S. Hong, and B. Han, "Learning Deconvolution Network for Semantic Segmentation," in *2015 IEEE International Conference on Computer Vision (ICCV)*, (Los Alamitos, CA, USA), pp. 1520–1528, IEEE Computer Society, dec 2015.

[22] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[23] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds.), (Cham), pp. 234–241, Springer International Publishing, 2015.

[24] H. Li, P. Xiong, H. Fan, and J. Sun, "DFANet: Deep Feature Aggregation for Real-Time Semantic Segmentation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Los Alamitos, CA, USA), pp. 9514–9523, IEEE Computer Society, jun 2019.

[25] P. Chao, C.-Y. Kao, Y. Ruan, C.-H. Huang, and Y.-L. Lin, "HarDNet: A Low Memory Traffic Network," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3551–3560, 2019.

[26] M. Orsic, I. Kreso, P. Bevandic, and S. Segvic, "In Defense of Pre-Trained ImageNet Architectures for Real-Time Semantic Segmentation of Road-Driving Images," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Los Alamitos, CA, USA), pp. 12599–12608, IEEE Computer Society, jun 2019.

[27] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1175–1183, 2017.

[28] A. Chaurasia and E. Culurciello, "LinkNet: Exploiting encoder representations for efficient semantic segmentation," in *2017 IEEE visual communications and image processing (VCIP)*, pp. 1–4, IEEE, 2017.

[29] G. Lin, A. Milan, C. Shen, and I. Reid, "RefineNet: Multi-path Refinement Networks for High-Resolution Semantic Segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5168–5177, 2017.

[30] Y. Nirkin, L. Wolf, and T. Hassner, "HyperSeg: Patch-wise Hypernetwork for Real-time Semantic Segmentation," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4060–4069, 2021.

[31] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Object Detectors Emerge in Deep Scene CNNs," in *International Conference on Learning Representations (ICLR)*, 2015.

[32] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, p. 834—848, April 2018.

[33] M. Yang, K. Yu, C. Zhang, Z. Li, and K. Yang, "DenseASPP for Semantic Segmentation in Street Scenes," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3684–3692, 2018.

[34] W. Liu, A. Rabinovich, and A. Berg, "ParseNet: Looking Wider to See Better," *arXiv preprint arXiv:1506.04579*, 2015.

[35] G. Dong, Y. Yan, C. Shen, and H. Wang, "Real-Time High-Performance Semantic Image Segmentation of Urban Street Scenes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3258–3274, 2021.

[36] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local Neural Networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7794–7803, 2018.

[37] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3146–3154, 2019.

[38] Y. Yuan, L. Huang, J. Guo, C. Zhang, X. Chen, and J. Wang, "OCNet: Object Context for Semantic Segmentation," *International Journal of Computer Vision*, vol. 129, no. 8, p. 2375–2398, 2021.

[39] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "CCNet: Criss-Cross Attention for Semantic Segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[40] Z. Zhu, M. Xu, S. Bai, T. Huang, and X. Bai, "Asymmetric Non-Local Neural Networks for Semantic Segmentation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 593–602, 2019.

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[42] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018.

[43] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[44] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Los Alamitos, CA, USA), pp. 1800–1807, IEEE Computer Society, jul 2017.

[45] R. P. K. Poudel, S. Liwicki, and R. Cipolla, "Fast-SCNN: Fast Semantic Segmentation Network," in *30th British Machine Vision Conference,BMVC*, p. 289, BMVA Press, 2019.

[46] G. Li, I. Y. Yun, J. Kim, and J. Kim, "DABNet: Depth-wise Asymmetric Bottleneck for Real-time Semantic Segmentation," in *BMVC*, 2019.

[47] E. Romera, J. M. Álvarez, L. M. Bergasa, and R. Arroyo, "ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 2018.

[48] Y. Wang, Q. Zhou, J. Xiong, X. Wu, and X. Jin, "ESNet: An Efficient Symmetric Network for Real-Time Semantic Segmentation," in *Pattern Recognition and Computer Vision* (Z. Lin, L. Wang, J. Yang, G. Shi, T. Tan, N. Zheng, X. Chen, and Y. Zhang, eds.), (Cham), pp. 41–52, Springer International Publishing, 2019.

[49] "FasterSeg: Searching for Faster Real-time Semantic Segmentation, author=Chen, Wuyang and Gong, Xinyu and Liu, Xianming and Zhang, Qian and Li, Yuan and Wang, Zhangyang," in *International Conference on Learning Representations*, 2020.

[50] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation," in *4th International Conference on Learning Representations, ICLR*, 2018.

[51] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "ESPNet: Efficient Spatial Pyramid of Dilated Convolutions for Semantic Segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[52] S. Mehta, M. Rastegari, L. Shapiro, and H. Hajishirzi, "ESPNetv2: A Light-Weight, Power Efficient, and General Purpose Convolutional Neural Network," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Los Alamitos, CA, USA), pp. 9182–9192, IEEE Computer Society, jun 2019.

[53] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for Real-Time Semantic Segmentation on High-Resolution Images," in *Computer Vision – ECCV 2018* (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), pp. 418–434, Springer International Publishing, 2018.

[54] R. P. K. Poudel, U. D. Bonde, S. Liwicki, and C. Zach, "ContextNet: Exploring Context and Detail for Semantic Segmentation in Real-time," in *BMVC*, 2018.

[55] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiSeNet: Bilateral Segmentation Network for Real-Time Semantic Segmentation," in *Computer Vision – ECCV 2018* (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), Springer International Publishing, 2018.

[56] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, "BiSeNet V2: Bilateral Network with Guided Aggregation for Real-Time Semantic Segmentation," *International Journal of Computer Vision*, vol. 129, no. 11, pp. 3051–3068, 2021.

[57] J. Fan, F. Wang, H. Chu, X. Hu, Y. Cheng, and B. Gao, "MLFNet: Multi-Level Fusion Network for Real-Time Semantic Segmentation of Autonomous Driving," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 756–767, 2023.

[58] M. Y. Yang, S. Kumaar, Y. Lyu, and F. Nex, "Real-time Semantic Segmentation with Context Aggregation Network," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 178, pp. 124–134, 2021.

[59] H. Pan, Y. Hong, W. Sun, and Y. Jia, "Deep Dual-Resolution Networks for Real-Time and Accurate Semantic Segmentation of Traffic Scenes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 3, pp. 3448–3460, 2023.

[60] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep High-Resolution Representation Learning for Visual Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3349–3364, 2021.

[61] L. Rosas-Arias, G. Benitez-Garcia, J. Portillo-Portillo, J. Olivares-Mercado, G. Sanchez-Perez, and K. Yanai, "FASSD-Net: Fast and Accurate Real-Time Semantic Segmentation for Embedded Systems," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2021.

[62] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[63] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, "Segmenter: Transformer for Semantic Segmentation," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7242–7252, IEEE Computer Society, oct 2021.

[64] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. S. Torr, and L. Zhang, "Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6877–6886, IEEE Computer Society, jun 2021.

[65] J. Zhang, K. Yang, A. Constantinescu, K. Peng, K. Müller, and R. Stiefelhagen, "Trans4Trans: Efficient Transformer for Transparent Object and Semantic Scene Segmentation in Real-World Navigation Assistance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 19173–19186, 2022.

[66] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[67] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid Scene Parsing Network," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6230–6239, 2017.

[68] M. Oršić and S. Šegvić, "Efficient semantic segmentation with pyramidal fusion," *Pattern Recognition*, vol. 110, p. 107611, 2021.

[69] F. Yu and V. Koltun, "Multi-Scale Context Aggregation by Dilated Convolutions," *4th International Conference on Learning Representations, ICLR*, 2016.

[70] R. Gao, "Rethinking Dilated Convolution for Real-time Semantic Segmentation," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 4675–4684, 2023.

[71] M. Lu, Z. Chen, Q. M. J. Wu, N. Wang, X. Rong, and X. Yan, "FRNet: Factorized and Regular Blocks Network for Semantic Segmentation in Road Scene," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 4, pp. 3522–3530, 2022.

[72] M. Lu, Z. Chen, C. Liu, S. Ma, L. Cai, and H. Qin, "MFNet: Multi-Feature Fusion Network for Real-Time Semantic Segmentation in Road Scenes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 20991–21003, 2022.

[73] T. Wu, S. Tang, R. Zhang, J. Cao, and Y. Zhang, "CGNet: A Light-Weight Context Guided Network for Semantic Segmentation," *IEEE Transactions on Image Processing*, vol. 30, pp. 1169–1179, 2021.

[74] X. Zhang, B. Du, Z. Wu, and T. Wan, "LAANet: lightweight attention-guided asymmetric network for real-time semantic segmentation," *Neural Computing and Applications*, vol. 34, no. 5, pp. 3573–3587, 2022.

[75] Q. Zhou, Y. Wang, Y. Fan, X. Wu, S. Zhang, B. Kang, and L. J. Latecki, "AGLNet: Towards real-time semantic segmentation of self-driving images via attention-guided lightweight network," *Applied Soft Computing*, vol. 96, p. 106682, 2020.

[76] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *CoRR*, vol. abs/1704.04861, 2017.

[77] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, pp. 6105–6114, PMLR, 2019.

[78] W. He, M. Wu, M. Liang, and S.-K. Lam, "CAP: Context-Aware Pruning for Semantic Segmentation," in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 959–968, 2021.

[79] X. Xu, Q. Lu, L. Yang, S. Hu, D. Chen, Y. Hu, and Y. Shi, "Quantization of Fully Convolutional Networks for Accurate Biomedical Image Segmentation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8300–8308, 2018.

[80] S. An, Q. Liao, Z. Lu, and J.-H. Xue, "Efficient Semantic Segmentation via Self-Attention and Self-Distillation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 15256–15266, 2022.

[81] Y. Liu, K. Chen, C. Liu, Z. Qin, Z. Luo, and J. Wang, "Structured Knowledge Distillation for Semantic Segmentation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2599–2608, 2019.

[82] Y. Liu, Y. Wang, L. Chang, and J. Zhou, "A Fast and Efficient FPGA-based Level Set Hardware Accelerator for Image Segmentation," in *2020 IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA)*, pp. 61–62, 2020.

[83] P. Morì, M.-R. Vemparala, N. Fasfous, S. Mitra, S. Sarkar, A. Frickenstein, L. Frickenstein, D. Helms, N. S. Nagaraja, W. Stechele, and C. Passerone, "Accelerating and pruning CNNs for semantic segmentation on FPGA," DAC '22, (New York, NY, USA), p. 145–150, Association for Computing Machinery, 2022.

[84] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, (Los Alamitos, CA, USA), pp. 2999–3007, IEEE Computer Society, oct 2017.

[85] Z. Ding, X. Han, P. Liu, and M. Niethammer, "Local temperature scaling for probability calibration," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6889–6899, 2021.

[86] N. Abraham and N. M. Khan, "A Novel Focal Tversky Loss Function With Improved Attention U-Net for Lesion Segmentation," *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pp. 683–687, 2019.

[87] M. Berman, A. Triki, and M. B. Blaschko, "The Lovasz-Softmax Loss: A Tractable Surrogate for the Optimization of the Intersection-Over-Union Measure in Neural Networks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4413–4421, 2018.

[88] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," *Advances in neural information processing systems*, vol. 31, 2018.

[89] Y. S. Aurelio, G. M. de Almeida, C. L. de Castro, and A. P. Braga, "Learning from imbalanced data sets with weighted cross-entropy function," *Neural processing letters*, vol. 50, no. 2, pp. 1937–1949, 2019.

[90] T. Eelbode, J. Bertels, M. Berman, D. Vandermeulen, F. Maes, R. Bisschops, and M. B. Blaschko, "Optimization for Medical Image Segmentation: Theory and Practice When Evaluating With Dice Score or Jaccard Index," *IEEE Transactions on Medical Imaging*, vol. 39, no. 11, pp. 3679–3690, 2020.

[91] L. Fidon, W. Li, L. C. Garcia-Peraza-Herrera, J. Ekanayake, N. Kitchen, S. Ourselin, and T. Vercauteren, "Generalised Wasserstein Dice Score for Imbalanced Multi-class Segmentation Using Holistic Convolutional Networks," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, (Cham), pp. 64–76, Springer International Publishing, 2018.

[92] M. Fan, S. Lai, J. Huang, X. Wei, Z. Chai, J. Luo, and X. Wei, "Rethinking BiSeNet for Real-Time Semantic Segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9716–9725, June 2021.

[93] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 761–769, 2016.

[94] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training Deep Neural Networks on Noisy Labels with Bootstrapping," 2014.

[95] I. Radosavovic, P. Dollár, R. Girshick, G. Gkioxari, and K. He, "Data Distillation: Towards Omni-Supervised Learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[96] M. A. Rahman and Y. Wang, "Optimizing intersection-over-union in deep neural networks for image segmentation," in *International symposium on visual computing*, pp. 234–244, Springer, 2016.

[97] S. B. Dalirsefat, A. da Silva Meyer, and S. Z. Mirhoseini, "Comparison of similarity coefficients used for cluster analysis with amplified fragment length polymorphism markers in the silkworm, Bombyx mori," *Journal of Insect Science*, vol. 9, no. 1, 2009.

[98] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. Jorge Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," in *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pp. 240–248, Springer, 2017.

[99] D. Karimi and S. E. Salcudean, "Reducing the hausdorff distance in medical image segmentation with convolutional neural networks," *IEEE Transactions on medical imaging*, vol. 39, no. 2, pp. 499–513, 2019.

[100] O. Ecabert, J. Peters, H. Schramm, C. Lorenz, J. von Berg, M. J. Walker, M. Vembar, M. E. Olszewski, K. Subramanyan, G. Lavi, *et al.*, "Automatic model-based segmentation of the heart in ct images," *IEEE transactions on medical imaging*, vol. 27, no. 9, pp. 1189–1201, 2008.

[101] Y. Wang, Q. Zhou, J. Liu, J. Xiong, G. Gao, X. Wu, and L. J. Latecki, "LEDNet: A Lightweight Encoder-Decoder Network for Real-Time Semantic Segmentation," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 1860–1864, 2019.

[102] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers," in *Advances in Neural Information Processing Systems*, 2021.

[103] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell, "Understanding Convolution for Semantic Segmentation," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1451–1460, 2018.

[104] E. Arani, S. Marzban, A. Pata, and B. Zonooz, "RGPNet: A Real-Time General Purpose Semantic Segmentation," in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 3008–3017, 2021.

[105] Z. Chen, Y. Duan, W. Wang, J. He, T. Lu, J. Dai, and Y. Qiao, "Vision Transformer Adapter for Dense Predictions," 2022.

[106] S. Y. Lo, H. M. Hang, S. W. Chan, and J. J. Lin, "Efficient dense modules of asymmetric convolution for real-time semantic segmentation," in *Proceedings of the ACM Multimedia Asia*, pp. 1–6, 2019.

[107] W. Jiang, Z. Xie, Y. Li, C. Liu, and H. Lu, "LRNNET: A Light-Weighted Network with Efficient Reduced Non-Local Operation for Real-Time Semantic Segmentation," in *2020 IEEE International Conference on Multimedia and Expo Workshop (ICMEW)*, (Los Alamitos, CA, USA), pp. 1–6, IEEE Computer Society, jul 2020.

[108] A. Lou and M. Loew, "CFPNET: Channel-Wise Feature Pyramid For Real-Time Semantic Segmentation," in *2021 IEEE International Conference on Image Processing (ICIP)*, pp. 1894–1898, 2021.

[109] M. Liu and H. Yin, "Feature Pyramid Encoding Network for Real-time Semantic Segmentation," in *Proceedings of the British Machine Vision Conference (BMVC)* (K. Sidorov and Y. Hicks, eds.), pp. 203.1–203.13, BMVA Press, September 2019.

[110] J. Wang, C. Gou, Q. Wu, H. Feng, J. Han, E. Ding, and J. Wang, "RTFormer: Efficient Design for Real-Time Semantic Segmentation with Transformer," in *Advances in Neural Information Processing Systems* (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds.), vol. 35, pp. 7423–7436, Curran Associates, Inc., 2022.

[111] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," *arXiv*, 2017.

[112] A. Giusti, D. C. Cireşan, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," in *2013 IEEE International Conference on Image Processing*, pp. 4034–4038, 2013.

[113] M. Lin, Q. Chen, and S. Yan, "Network In Network," *arXiv*, 2013.

[114] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada* (H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, eds.), pp. 8024–8035, 2019.

[115] A. Kaur, A. P. S. Chauhan, and A. K. Aggarwal, "Prediction of Enhancers in DNA Sequence Data using a Hybrid CNN-DLSTM Model," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 20, no. 2, pp. 1327–1336, 2023.

[116] J. Liu, Q. Zhou, Y. Qiang, B. Kang, X. Wu, and B. Zheng, "FDDWNet: A Lightweight Convolutional Neural Network for Real-Time Semantic Segmentation," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2373–2377, 2020.

[117] G. Xu, J. Li, G. Gao, H. Lu, J. Yang, and D. Yue, "Lightweight Real-Time Semantic Segmentation Network With Efficient Transformer and CNN," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 12, pp. 15897–15906, 2023.

[118] M. Shi, S. Lin, Q. Yi, J. Weng, A. Luo, and Y. Zhou, "Lightweight Context-Aware Network Using Partial-Channel Transformation for Real-Time Semantic Segmentation," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–16, 2024.

[119] S. Mazhar, N. Atif, M. K. Bhuyan, and S. R. Ahamed, "Rethinking dabnet: Light-weight network for real-time semantic segmentation of road scenes," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 6, pp. 3098–3108, 2024.

[120] G. Yang, Y. Wang, and D. Shi, "Reparameterizable Dual-Resolution Network for Real-time Semantic Segmentation," 2024.

[121] J. He, Z. Deng, L. Zhou, Y. Wang, and Y. Qiao, "Adaptive Pyramid Context Network for Semantic Segmentation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7511–7520, 2019.

[122] X.-L. Zhang, B.-C. Du, Z.-C. Luo, and K. Ma, "Lightweight and efficient asymmetric network design for real-time semantic segmentation," *Applied Intelligence*, vol. 52, no. 1, pp. 564–579, 2022.

[123] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 618–626, 2017.

[124] H. Cao, G. Chen, H. Zhao, D. Jiang, X. Zhang, Q. Tian, and A. Knoll, "SDPT: Semantic-Aware Dimension-Pooling Transformer for Image Segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 11, pp. 15934–15946, 2024.

[125] Z. Yang, H. Yu, Q. Fu, W. Sun, W. Jia, M. Sun, and Z.-H. Mao, "NDNet: Narrow While Deep Network for Real-Time Semantic Segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 9, pp. 5508–5519, 2021.

[126] L. Yang, Y. Bai, F. Ren, C. Bi, and R. Zhang, "LCFNets: Compensation Strategy for Real-Time Semantic Segmentation of Autonomous Driving," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 4, pp. 4715–4729, 2024.

[127] J. Fan, B. Gao, Q. Ge, Y. Ran, J. Zhang, and H. Chu, "SegTransConv: Transformer and CNN Hybrid Method for Real-Time Semantic Segmentation of Autonomous Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 2, pp. 1586–1601, 2024.

[128] X. Li, J. Zhang, Y. Yang, G. Cheng, K. Yang, Y. Tong, and D. Tao, "SFNet: Faster and Accurate Domain Agnostic Semantic Segmentation via Semantic Flow," *International Journal of Computer Vision*, vol. 132, no. 2, pp. 1573–1405, 2023.

[129] Q. Zhou, L. Wang, G. Gao, B. Kang, W. Ou, and H. Lu, "Boundary-Guided Lightweight Semantic Segmentation With Multi-Scale Semantic Context," *IEEE Transactions on Multimedia*, vol. 26, pp. 7887–7900, 2024.

[130] C. Sakaridis, D. Dai, and L. Van Gool, "Semantic Foggy Scene Understanding with Synthetic Data," *International Journal of Computer Vision*, vol. 126, pp. 973–992, Sep 2018.

[131] X. Hu, C.-W. Fu, and L. Zhu, "Depth-Attentional Features for Single-Image Rain Removal," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[132] Y. Liu, S. Wang, C. Wang, M. Lu, and Y. Sang, "Latent domain knowledge distillation for nighttime semantic segmentation," *Engineering Applications of Artificial Intelligence*, vol. 132, p. 107940, 2024.

[133] J. Sanchez, J.-E. Deschaud, and F. Goulette, "Cola: Coarse-label multisource lidar semantic segmentation for autonomous driving," *IEEE Transactions on Robotics*, vol. 41, pp. 1742–1754, 2025.

[134] M. Fan, S. Lai, J. Huang, X. Wei, Z. Chai, J. Luo, and X. Wei, "Rethinking BiSeNet for real-time semantic segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9716–9725, 2021.

[135] A. C. C. Bovik, M. Clark, and W. S. Geisler, "Multichannel texture analysis using localized spatial filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, 1990.

[136] Xuming He, R. S. Zemel, and M. A. Carreira-Perpinan, "Multiscale conditional random fields for image labeling," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition,CVPR*, vol. 2, pp. II–II, 2004.

[137] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using Gabor filters," in *1990 IEEE International Conference on Systems, Man, and Cybernetics Conference Proceedings*, pp. 14–19, 1990.

[138] M. Yang, K. Yu, C. Zhang, Z. Li, and K. Yang, "DenseASPP for Semantic Segmentation in Street Scenes," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3684–3692, 2018.

[139] M. Saha and C. Chakraborty, "Her2Net: A Deep Framework for Semantic Segmentation and Classification of Cell Membranes and Nuclei in Breast Cancer Evaluation," *IEEE Transactions on Image Processing*, vol. 27, no. 5, pp. 2189–2200, 2018.

[140] G. Coleman and H. Andrews, "Image segmentation by clustering," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 773–785, 1979.

[141] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015.

[142] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size," in *5th International Conference on Learning Representations, ICLR,Toulon, France*, 2017.

[143] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.," in *IEEE International Conference on Computer Vision(ICCV)*, pp. 1026–1034, IEEE Computer Society, 2015.

[144] V. Doan, D. Nguyen, Q. Tran, D. Nguyen, and T. Le, "Real-Time Image Semantic Segmentation Networks with Residual Depth-Wise Separable Blocks," in *Joint 10th IEEE International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th IEEE International Symposium on Advanced Intelligent Systems (ISIS)*, pp. 174–179, 2018.

[145] X. Li, A. You, Z. Zhu, H. Zhao, M. Yang, K. Yang, S. Tan, and Y. Tong, "Semantic Flow for Fast and Accurate Scene Parsing," in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), (Cham), pp. 775–793, Springer International Publishing, 2020.

[146] M. Orsic, I. Kreso, P. Bevandic, and S. Segvic, "In Defense of Pre-Trained ImageNet Architectures for Real-Time Semantic Segmentation of Road-Driving Images," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12599–12608, 2019.

[147] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for MobileNetV3," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1314–1324, 2019.

[148] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6, 2017.

[149] Y. Sun, W. Zuo, P. Yun, H. Wang, and M. Liu, "FuseSeg: Semantic Segmentation of Urban Scenes Based on RGB and Thermal Data Fusion," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1000–1011, 2021.

[150] W. Zhou, Y. Xiao, W. Yan, and L. Yu, "CMPFFNet: Cross-Modal and Progressive Feature Fusion Network for RGB-D Indoor Scene Semantic Segmentation," *IEEE Transactions on Automation Science and Engineering*, pp. 1–11, 2023.

[151] S. Mazhar, N. Atif, M. K. Bhuyan, and S. R. Ahamed, "S-net: A lightweight real-time semantic segmentation network for autonomous driving," in *Computer Vision and Image Processing - 8th International Conference, CVIP 2023, Jammu, India, November 3-5, 2023, Revised Selected Papers, Part II*, vol. 2010 of *Communications in Computer and Information Science*, pp. 147–159, Springer, 2023.

[152] A. Kendall and R. Cipolla, "Geometric Loss Functions for Camera Pose Regression With Deep Learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[153] H. Jung, H.-S. Choi, and M. Kang, "Boundary Enhancement Semantic Segmentation for Building Extraction From Remote Sensed Image," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–12, 2022.

[154] H. Kervadec, J. Bouchtiba, C. Desrosiers, E. Granger, J. Dolz, and I. Ben Ayed, "Boundary loss for highly unbalanced segmentation," in *Proceedings of The 2nd International Conference on Medical Imaging with Deep Learning* (M. J. Cardoso, A. Feragen, B. Glocker, E. Konukoglu, I. Oguz, G. Unal, and T. Vercauteren, eds.), vol. 102 of *Proceedings of Machine Learning Research*, pp. 285–296, PMLR, 08–10 Jul 2019.

[155] W. Jiang, Y. Wu, L. Guan, and J. Zhao, "DFNet: Semantic Segmentation on Panoramic Images with Dynamic Loss Weights and Residual Fusion Block," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 5887–5892, 2019.

[156] A. Balagopal, H. Morgan, M. Dohopolski, R. Timmerman, J. Shan, D. F. Heitjan, W. Liu, D. Nguyen, R. Hannan, A. Garant, N. Desai, and S. Jiang, "PSA-Net: Deep learning–based physician style–aware segmentation network for postoperative prostate cancer clinical target volumes," *Artificial Intelligence in Medicine*, vol. 121, p. 102195, 2021.

[157] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu, "Dynamic Convolution: Attention Over Convolution Kernels," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.