Real-time Semantic Segmentation for Autonomous driving

Peeyush Dyavarashetty
Master of Science in Applied Machine Learning
University of Maryland, College Park
College Park, US
peeyush@umd.edu

Srijinesh Alanka
Master of Science in Applied Machine Learning
University of Maryland, College Park
College Park, US
srij18@umd.edu

Rekha Srinivas Chimpiri
Master of Science in Data Science
University of Maryland, College Park
College Park, US
rekhach@umd.edu

Abdul Hannan
Master of Science in Data Science
University of Maryland, College Park
College Park, US
hannn@umd.edu

***Abstract*—In the quest to improve semantic segmentation a technique for understanding and labeling each part of an image—we explored several advanced models to find the most effective approach. Semantic segmentation helps computers understand images by dividing them into meaningful parts, which is crucial for tasks like autonomous driving and medical imaging. We tested various models including BiSeNet, Fast-SCNN, PPLite-Seg, STDC-Seg, and DDRNet, each with different training setups and loss functions. Our experiments revealed that BiSeNet with dataset-specific parameters achieved the best performance in terms of Intersection over Union (mIoU), while other models like PPLite-Seg and DDRNet faced significant computational challenges. The findings underscore the importance of optimizing both model configuration and training strategies to balance accuracy and efficiency. This research not only highlights the strengths and limitations of different approaches but also sets the stage for future advancements in semantic segmentation.**

I. Problem Statement

Real-time semantic segmentation is a cornerstone technology for autonomous driving, enabling vehicles to accurately perceive and interact with their environment. By segmenting visual input into distinct classes such as vehicles, pedestrians, road signs, and lanes, semantic segmentation provides the necessary information for making real-time decisions on the road. This technology is critical for ensuring autonomous vehicles' safety, efficiency, and reliability, as it allows them to detect obstacles, recognize traffic signs, and precisely navigate complex driving scenarios.

Achieving this requires a delicate balance between accuracy and computational efficiency. Autonomous systems must process vast amounts of visual data in milliseconds, which necessitates the development of high-speed, yet accurate, segmentation models. Traditional methods [1]–[6] struggle to achieve real-time semantic segmentation due to limited processing capabilities and lack of robustness, while traditional deep learning models [7]–[10] often face challenges in maintaining real-time performance without sacrificing segmentation quality. Therefore, advancing deep learning techniques to achieve both high accuracy and real-time processing is essential for the continued development and deployment of autonomous driving systems.

II. Deep Learning Models Used

Fast-SCNN is a lightweight and efficient model that leverages an encoder-decoder architecture with a depthwise separable convolution to reduce the computational burden while preserving segmentation accuracy. It focuses on early down-sampling, effectively balancing speed and accuracy, making it well-suited for resource-constrained environments.

BiSeNet adopts a two-branch network structure, where the spatial path captures high-resolution details and the context path focuses on extracting semantic information at a lower resolution. This dual-path design allows BiSeNet to achieve high accuracy while maintaining fast inference speeds, making it ideal for real-time applications in dynamic environments. Together, these models offer a robust solution to the challenges of real-time semantic segmentation in autonomous driving.

YOLO (You Only Look Once) features a single-stage architecture that predicts bounding boxes and class probabilities simultaneously across the entire image. This unique design enables real-time object detection with minimal computational overhead, making it ideal for applications like video surveillance and autonomous driving.

PP-Lite-Seg utilizes a streamlined encoder-decoder architecture with an efficient backbone, employing channel pruning and quantization to minimize computational costs. Its lightweight design allows for real-time segmentation on devices with limited resources.

STDC (Short-Term Dense Connection) adopts a unique dense connection strategy in a lightweight backbone, enhancing feature reuse and reducing redundancy. This architecture strikes a balance between accuracy and speed, making it suitable for real-time tasks.

DDRNet (Dual-Resolution Residual Network) combines a dual-resolution architecture where a high-resolution branch captures detailed spatial information and a low-resolution branch captures global context. This structure allows DDRNet to deliver both high accuracy and efficiency, particularly in urban scene segmentation.

LinkNet employs an encoder-decoder architecture with residual connections, which ensures efficient propagation of spatial information and faster convergence during training. Its unique design allows for quick and accurate segmentation, making it ideal for real-time applications like medical imaging and autonomous navigation.

III. Datasets

The Mapillary Vistas dataset [11] is one of the significant contribution to the field of semantic segmentation for autonomous driving. Unlike Cityscapes, Mapillary Vistas offers a more diverse collection of images sourced from a global community of contributors. It contains 25,000 high-resolution images with fine-grained annotations covering 66 object categories, including traffic signs, vegetation, and various types of vehicles. This extensive and diverse dataset enhances the ability of segmentation models to generalize across different geographic locations, environmental conditions, and traffic scenarios. By providing a wide range of visual contexts, Mapillary Vistas supports the development of more adaptable and reliable perception systems for autonomous vehicles, pushing the boundaries of current segmentation research.

IV. Procedure

A. Loss function

In our study, we employed several loss functions to optimize the performance of our model, including Focal Loss, Cross Entropy Loss, Cross Entropy with class weights, Jaccard Loss, and Dice Loss.

Focal Loss [12] was particularly useful in addressing the class imbalance problem by modulating the loss contribution of hard-to-classify examples through a tuning parameter, gamma, which we experimented using 2 and 3. This adjustment allows the model to focus more on difficult samples, thereby improving performance over minority classes.

Cross Entropy Loss, a widely used loss function in classification tasks, was also utilized in its standard form and with class weighting. The weighted variant of Cross Entropy Loss helped compensate for the class imbalance by assigning higher weights to underrepresented classes, ensuring that the model did not disproportionately favor the majority classes. The weights for the Cross Entropy Loss are computed using the following formula:

$$w_i = \frac{1}{\log(p_i + 1.02)} \tag{1}$$

where p_i represents the proportion of samples in class. This weighting scheme adjusts the loss for each class to account for imbalances, helping the model to better learn from underrepresented classes.

Additionally, we explored Jaccard Loss (also known as the Intersection over Union loss) and Dice Loss, both of which are commonly used in segmentation tasks.

Jaccard Loss [13], also known as Intersection over Union (IoU) loss, is a popular metric used to evaluate the performance of pixel segmentation models. It's calculated by dividing the size of the intersection between a predicted segmentation mask and a ground truth segmentation mask by the size of their union. This ratio quantifies how much the predicted and actual segments overlap.

Dice Loss [14], is a popular loss function used to measure overlap in image segmentation. It's a powerful method for semantic segmentation and well-suited to handling class imbalance in terms of pixel count for foreground and background. It is a measure of the dissimilarity between the predicted segmentation and the true segmentation of an image.

B. Metrics and Hyperparameters

*1) **Intersection over Union**:* Intersection over Union (IoU) is a metric used in computer vision to evaluate the accuracy of image segmentation algorithms. It's also known as Jaccard's Index. IoU measures how well a model can distinguish objects from their backgrounds in an image by quantifying the overlap between the predicted region and the ground truth region. A higher IoU score indicates better segmentation performance, while a lower score indicates poorer performance.

*2) **Polynomial Learning Rate**:* Learning rate decay is a technique used in machine learning models, especially deep neural networks. Throughout the training phase, it entails gradually lowering the learning rate. Learning rate decay is used to gradually adjust the learning rate, usually by lowering it, to facilitate the optimization algorithm's more rapid convergence to a better solution. When a polynomial function, usually a power of the epoch number, is followed, polynomial decay lowers the learning rate. We started with an initial learning rate of 0.5. After every epoch, the learning rate is updated as the product of itself with the given formula:

$$lr = lr * \left(1 - \frac{\text{epoch}}{\text{max\_epoch}}\right)^{\text{power}} \tag{2}$$

C. Segmentation-models-pytorch

Segmentation Models PyTorch [15](SMP) is a popular library for training deep-learning models on image segmentation tasks. It provides a wide range of pre-trained models, loss functions, and utilities that make it easier to implement and experiment with state-of-the-art segmentation techniques. The library includes many popular segmentation architectures such as U-Net, FPN (Feature Pyramid Network), DeepLabV3, PSPNet, and more. These models can be initialized with pre-trained weights from large datasets like ImageNet, which helps with transfer learning and speeds up convergence.

D. *super-gradients 3.5.0*

Super-Gradients is an open-source deep learning training library that simplifies and accelerates the process of training and fine-tuning state-of-the-art deep learning models. Developed by Deci AI [16], it's particularly focused on computer vision tasks like image classification, object detection, and segmentation. The library provides pre-trained models, easy-to-use training pipelines, and efficient data handling, making it a powerful tool for both researchers and practitioners.

E. *Computing Resources - **Zaratan Cluster***

The Zaratan High-Performance Computing (HPC) cluster is the University of Maryland's flagship HPC cluster, maintained by the Division of Information Technology and replacing the Deepthought2 cluster. Coming online in spring 2022, it features 360 compute nodes, each with dual AMD 7763 64-core CPU's. These CPU's are direct liquid cooled to enable all of the approximately 50,000 CPU cores to run at full speed. There are also 20 GPU nodes, each containing four Nvidia A100 GPUs (for a total of 80 GPUs). Theoretical peak performance is 3.5 PFLOPS. The cluster has HDR-100 (100 Gbit) Infiniband interconnects between the nodes, with storage and service nodes connected with full HDR (200 Gbit). The cluster is connected with 200 Gbit Ethernet to various national networks.

F. *Data augmentation*

During the training phase, image data underwent several augmentation techniques to improve the model's robustness and generalization capabilities. One of the key augmentation steps involved resizing the images to a resolution of 1024x2048 pixels. This resizing process ensured that all images had consistent dimensions, which is crucial for maintaining uniformity in model input and optimizing training performance.

In addition to resizing, the images were normalized. Normalization is a process where pixel values are scaled in between -1 and 1. This step helps in stabilizing and accelerating the training process by ensuring that the input data has similar statistical properties. Normalization can also aid in reducing the impact of varying lighting conditions and contrast between images, thus improving the model's ability to generalize across different scenarios.

G. *Why Overfitting?*

Overfitting a model on a sample dataset is a valuable step in the development process as it provides several benefits. First, it serves as a model capability check, confirming that the model is complex enough to learn from the data. It also helps establish a baseline performance, indicating the maximum potential accuracy that the model can achieve on the given data. By deliberately overfitting, we can identify potential issues such as poor feature selection, bugs, or problems with data preprocessing. Once overfitting is achieved, we can employ learning rate scheduling to refine the model, gradually reducing the learning rate to balance between fitting the training data and improving generalization. This iterative improvement process, combined with the right learning rate adjustments, ensures that the model can effectively learn and perform well on unseen data. The principle is that if we can overfit the model over a small subset of data, then it can gradually fit over the whole dataset.

V. Training

A. *YOLO*

We initially adopted the YOLO-seg model using the Ultralytics framework, aiming to overfit the model on a sample dataset as a preliminary step. In the experiments, various hyperparameters were systematically tuned to optimize model performance. The batch sizes were varied between 4, 8, and 16 to observe their effects on convergence and stability. Two optimizers, Adam and SGD, were employed to assess differences in optimization strategies, with initial learning rates (lr_0) of 0.01 and 0.001, and final learning rates (lr_f) following the same values.

Additionally, a cosine learning rate schedule was considered to evaluate its impact on model performance. The Intersection over Union (IoU) thresholds were set to 0.5, 0.7, and 0.9 to determine the trade-off between precision and recall in object detection. Warmup strategies, including epochs and momentum, were also explored with different configurations, specifically warmup epochs of 0, 3, and 5, and warmup momentum values ranging from 0.8 to 1.0.

The loss function components, such as classification loss (cls), Distribution Focal Loss (DFL), and scale, were adjusted within specified ranges to balance the model's accuracy and robustness. Techniques like random erasing and varying crop fractions were applied as data augmentation methods to enhance model generalization. Finally, the experiments included trials with and without a predefined configuration file (yolov8n-seg.cfg) to assess the effect of custom model architecture settings.

Despite these extensive efforts and adjustments, the YOLO-seg model failed to overfit the sample dataset. The model's inability to adapt, even under varied and specific conditions, indicated inherent limitations in its architecture or compatibility with the task at hand. This persistent underperformance, despite a well-considered configuration, ultimately led us to conclude that the YOLO-seg model was not suitable for our project's requirements, prompting its discontinuation.

B. *Fast-SCNN*

In this study, we explored the training and evaluation of the Fast-SCNN model using several configurations. The models were trained using different loss functions, learning rate schedules, and optimizers to assess their impact on performance. Specifically, we used Cross Entropy Loss with and without class weights, Focal Loss, and the original parameters specified in the Fast-SCNN research paper.

Initially, the Fast-SCNN model was trained using Cross Entropy Loss with Adam optimizer. The learning rate was set at $1e-3$ for the first 30 epochs, followed by a reduction to

Loss function	Optimizer	Epochs	mIoU (1024x2048)	avg_time (1024x2048)
Cross Entropy Loss	Adam	60	**0.219134911**	0.006267594
Cross Entropy Loss with weights	Adam	60	0.213734561	0.00672004
Focal Loss (γ = 3)	Adam	25	0.196436506	0.006197016
Cross Entropy with polynomial lr	SGD + Momentum = 0.9	52	0.216329643	0.006472178

TABLE I: *Results after experimenting with Fast-SCNN*

Loss function	Optimizer	Epochs	mIoU (1024x2048)	avg_time (1024x2048)
Cross Entropy Loss with weights	Adam	60	0.226107562	0.010200358
Focal Loss (γ = 2)	Adam	18	0.224503599	0.005276626
Cross Entropy with polynomial lr	SGD + Momentum = 0.9	52	**0.247017986**	0.005859865

TABLE II: *Results after experimenting with BiSeNet*

$1e-4$ for epochs 31 to 60. The training loss demonstrated a consistent decrease, as depicted in Figure 1.This method required approximately 3.5 hours per epoch. However, a slight increase in the validation loss was observed after 31 epochs, which could be attributed to the learning rate adjustment.

Class weights were then incorporated into the Cross-Entropy Loss to address the class imbalance. The learning rate schedule remained the same as in the previous configuration, and this method required around 1.5 hours per epoch. As shown in Figure 2, the validation loss exhibited a similar trend, with a minor increase after the learning rate was reduced. This suggests that the application of class weights did not significantly alter the model's response to the learning rate change.

Next, we explored the use of Focal Loss with a gamma value of 3 to address class imbalance more effectively. The learning rate was initially set at $1e-3$ for the first 10 epochs, reduced to $1e-4$ for epochs 11 to 20, and further decreased to $1e-5$ for epochs 21 to 25. This method required approximately 5 hours per epoch. The results, illustrated in Figure 3, showed a more pronounced decrease in training loss, reflecting the effectiveness of Focal Loss in this scenario.

Following the original Fast-SCNN implementation [17] , we employed the SGD optimizer with a momentum of 0.9. A stepwise learning rate reduction, as recommended in the original study, was applied. As illustrated in Figure 4, both the training and validation losses exhibited a gradual decrease. This method required approximately 3.5 hours per epoch. However, a slight increase in the slope of the validation loss was observed after 30 epochs, likely due to the learning rate adjustment. This experiment demonstrated that while SGD can effectively optimize the Fast-SCNN model, careful tuning of the learning rate is crucial to maintaining validation performance.

Overall, the Fast-SCNN model performed better with the Cross-Entropy Loss and class weights at higher learning rates, as shown in Figure 1 and Figure 2. The Focal Loss configuration showed promise in handling class imbalance, but the model was sensitive to learning rate adjustments (Figure 3). The use of SGD in place of Adam, while effective, required careful management of the learning rate schedule to maintain consistent validation performance, as seen in Figure 4. The results are summarized in Table I. Notably, the highest mIoU was achieved by the Cross Entropy Loss without weights.

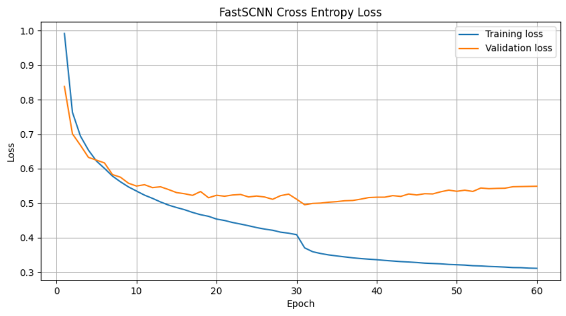

Fig. 1: *Fast-SCNN Cross Entropy Loss*

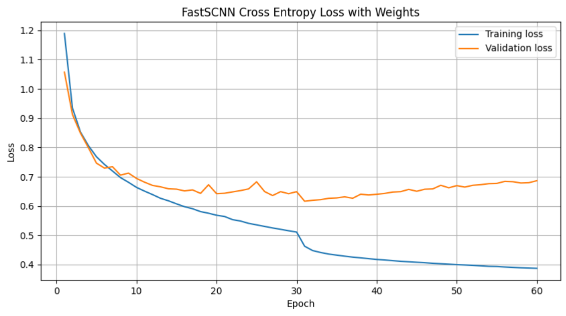

Fig. 2: *Fast-SCNN Cross Entropy Loss with Weights*

C. BiSeNet

we assessed the performance of the BiSeNet model under various configurations to determine the optimal settings for segmentation tasks.

We trained the BiSeNet model using Cross Entropy Loss with class weights. For this setup, the learning rate was set to $1e-3$ for the first 30 epochs. This was followed by a reduction to $1e-4$ for epochs 31 through 60 and further decreased to $1e-5$ from epochs 61 to 90. As illustrated in Figure 5, the training loss shows a notable plateau after the 61st epoch, indicating that the model's performance does not improve significantly beyond this point.

we used the BiSeNet model with Focal Loss. The learning rate schedule for this setup was stepwise: $1e-3$ for epochs 1 to 5, $1e-4$ for epochs 6 to 9, and $1e-5$ for epochs 10 to 18.

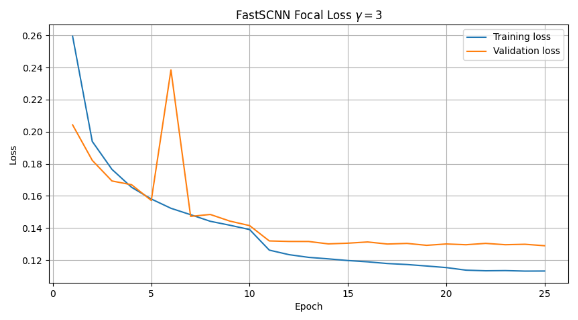

Fig. 3: *Fast-SCNN Focal Loss with* $\gamma = 3$

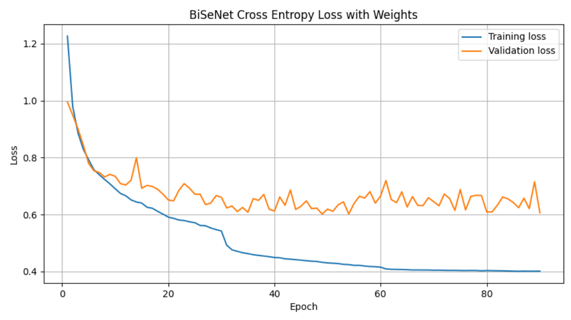

Fig. 5: *BiSeNet cross Entropy Loss with weights*

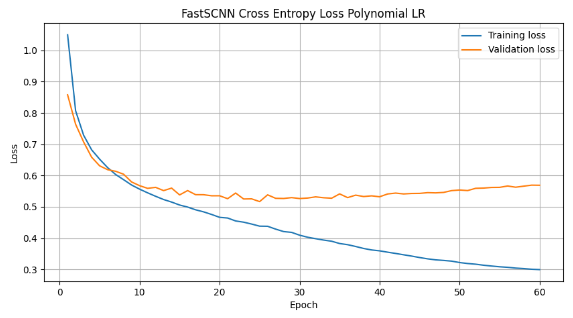

Fig. 4: *Fast-SCNN trained with the parameters used for training Cityscapes as shown in the paper*

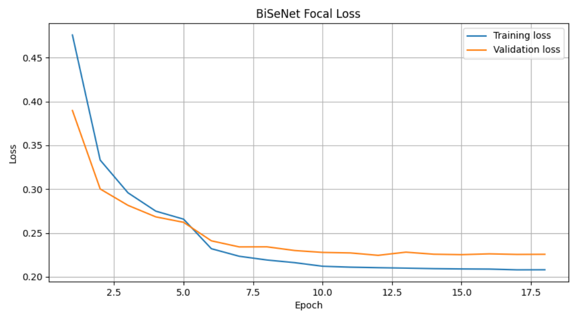

Fig. 6: *BiSeNet Focal Loss*

Figure 6 demonstrates that the training loss stabilizes after the initial epochs, with no substantial decrease observed beyond the 10th epoch. This indicates that while Focal Loss helps in handling class imbalance, it does not lead to continuous improvement in training loss after a certain point.

We employed the BiSeNet model with training parameters specifically tuned for the Cityscapes dataset, as described in the original research [18]. This configuration, depicted in Figure 7, achieved the highest mean Intersection over Union (mIoU) compared to the previous two models, as shown in Table II. Notably, although the training loss was high at the 60th epoch, the mIoU was at its peak at the 52nd epoch. This peak performance in mIoU led us to consider this model configuration as the most effective up to the 52nd epoch.

In summary, each configuration of the BiSeNet model offered unique insights into performance optimization. It is noteworthy that for Focal Loss, each epoch took approximately 4.5 hours to complete, while for Cross Entropy Loss with weights, each epoch took around 1 hour. The BiSeNet model with training parameters specifically tuned for the Cityscapes dataset took around 1.5 hours per epoch. This highlights the significant computational demands of these experiments and the careful consideration needed for each model's configuration and training duration. The Cross Entropy Loss with class weights showed a plateau in training loss, while Focal Loss helped stabilize the loss but did not continue to improve significantly. The Cityscapes training parameters provided the best overall performance, particularly in terms of mIoU, demonstrating the importance of using dataset-specific parameters for achieving optimal results.

D. PPLite-Seg

We utilized the PPLite B50 model [19], constructed using the supergradients module, for image segmentation tasks. The initial phase involved training the model over 50 epochs on a small subset of the training dataset to deliberately induce overfitting. This process allowed us to fine-tune and determine the optimal learning rate parameter, which was set at $1e-3$, which can be observed from Figure 8. However, when transitioning to training on the entire dataset, we encountered significant computational challenges. Utilizing four GPUs within the Zaratan cluster, each epoch took approximately 9-10 hours to complete. Due to the substantial computational demand,

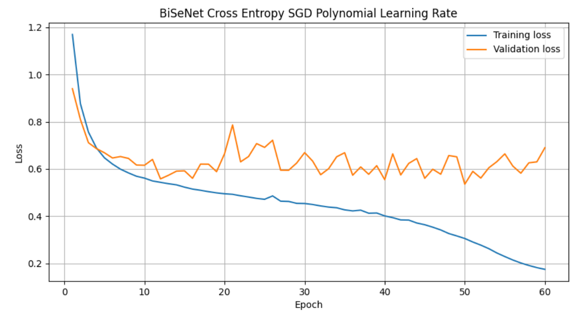

Fig. 7: *BiSeNet trained with the parameters used for training Cityscapes as in the paper*

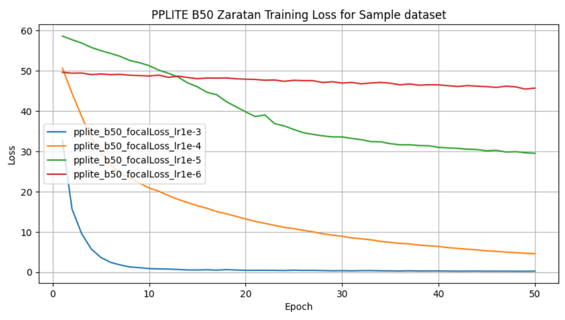

Fig. 8: *Resultant training error on the training dataset after training the PPLite B50 model on the sample dataset. This shows that the learning rate $1e-3$ will be best to run the model.*

we were only able to train for 2.5 epochs before halting the process.

The model's performance was evaluated using Focal Loss as the primary loss function, with the Intersection over Union (IoU) and Dice coefficient serving as the key evaluation metrics. Initial results from the first two epochs provided valuable insights. During the first epoch, the model demonstrated a focal loss of 1.4301, IoU of 0.039, and Dice coefficient of 0.0543 on the training set, while on the validation set, these values were 0.3632, 0.0687, and 0.0932, respectively. By the second epoch, significant improvements were observed, with the training focal loss decreasing sharply to 0.3343, and corresponding IoU and Dice coefficients improving to 0.0792 and 0.1084, respectively. These metrics reflect the model's ability to better capture and segment features as training progressed, albeit over a limited number of epochs due to computational constraints.

In our study, we experimented with the PPLite T50 model, initially utilizing a combined loss function of Dice, Cross-Entropy, and Edge Loss but failed and now selected the Cross-Entropy loss function with Adam optimizer, having a learning rate of $1e-3$ and weight decay of $1e-4$. The training was conducted with a batch size of 8, an image resolution of 512x1024, and a multi-GPU configuration spanning 4 GPUs. We allocated a runtime of 40 hours on the computing cluster; however, the model managed to complete only 3 epochs within this timeframe.

The results of the initial training attempts are summarized as follows. For Epoch 1, the training focal loss was 1.8783, with an IoU of 0.0325 and a Dice score of 0.0436. The validation metrics showed a focal loss of 0.3912, an IoU of 0.0514, and a Dice score of 0.0665. In Epoch 2, the training focal loss improved to 0.3506, with the IoU increasing to 0.0712 and the Dice score to 0.097. Similarly, validation metrics showed a reduction in focal loss to 0.3096, an IoU of 0.0883, and a Dice score of 0.1195. By Epoch 3, further improvements were observed, with the training focal loss decreasing to 0.2969, the IoU increasing to 0.0969, and the Dice score reaching 0.1316. The validation focal loss also decreased to 0.2794, with the IoU rising to 0.1058 and the Dice score to 0.142.

Despite these improvements, the training was halted due to the excessive computational demands, as subsequent attempts indicated that a single epoch would require approximately 28 hours to complete, a duration unsustainable given the limited GPU resources available on the Zaratan cluster.

E. STDC

In our study, we implemented the STDC-Seg model [20] using the supergradients module on the Zaratan cluster over 24 hours, utilizing a distributed data-parallel mode across 4 GPUs. The training process was conducted on a dataset consisting of 18,000 images, with a batch size of 4 per GPU, resulting in an effective batch size of 16. The model used was the STDC1Seg, comprising 9.35 million parameters, and the training was configured with a learning rate of 0.001 and weight decay adjustments of 0.0001 for certain parameter groups.

Throughout the training process, three epochs were completed. In the first epoch, the model's training metrics indicated a DiceCEEdgeLoss/main_loss of 2.1123, with auxiliary losses recorded as 2.4689 and 2.4226. The overall loss was calculated to be 7.8379, with an IoU of 0.1028. Validation metrics in the first epoch yielded a main loss of 1.8353, with an IoU of 0.1402. Progressively, the second epoch showed improvement in the training loss, with the main loss decreasing to 1.8364 and the IoU increasing to 0.1468. Similarly, validation losses reduced to 1.7195, with the IoU rising to 0.1769. The third epoch further reduced the main loss to 1.779, with a corresponding validation IoU of 0.1907.

Although the model exhibited consistent improvements in both training and validation metrics over the epochs, the computational demands of the STDC-Seg model were significant, requiring nearly the entire allocated 24-hour period to complete just three epochs. This indicates a need for further optimization in both the model and the training process to ensure more efficient utilization of computational resources in future experiments.

F. DDRNet

We employed the DDRNet23Slim model [21], a lightweight variant of the Dual-Resolution Network (DDRNet) architecture, using the supergradients module in python to conduct image segmentation tasks on a small sample dataset. The model, consisting of 5.70 million parameters, was trained and evaluated using a distributed data-parallel approach across four GPUs. The dataset used for this experiment consisted of 40 samples in the training set and the time is set for 5 hours in the Zaratan cluster to complete 50 epochs. The images were resized and normalized before being fed into the network. The training was conducted with the Distributed Data-Parallel setup, having 4 fully utilized GPUs, with batch size 2 per GPU making the effective batch size 8. Four different learning rates ($1e-6$, $1e-5$, $1e-4$, and $1e-3$) were used across multiple training sessions to observe the effect on model performance.

At a learning rate of $1e-6$, the model's performance over 50 epochs was suboptimal. The Focal Loss during training plateaued at approximately 23.511, with a minimal reduction from the previous epoch's loss (23.5263). The Intersection over Union (IoU) and Dice coefficients remained stagnant at 0.0007 and 0.0013, respectively. Validation metrics mirrored the training results, showing minimal improvement with a Focal Loss of 23.5994, an IoU of 0.0012, and a Dice coefficient of 0.0023.

Increasing the learning rate to $1e-5$ led to slight improvements in loss and accuracy metrics by epoch 45. The training Focal Loss decreased to 19.6169, while the validation Focal Loss improved to 19.5352. However, IoU and Dice metrics showed a decrease from their previous best, with training values of 0.0003 and 0.0005, and validation values of 0.0003 and 0.0007, respectively.

At a learning rate of $1e-4$, the model demonstrated more significant improvements by epoch 31. The training Focal Loss dropped to 10.8912, while the validation Focal Loss followed suit at 10.914. IoU and Dice coefficients increased to 0.0099 and 0.0121 for training, and 0.0093 and 0.0113 for validation, respectively.

The most substantial improvements were observed at a learning rate of $1e-3$ by epoch 23. The training Focal Loss was reduced to 0.8429, and the validation Focal Loss to 0.9567. Correspondingly, the IoU and Dice coefficients reached 0.023 and 0.0288 for training, and 0.0231 and 0.029 for validation.

The results indicate that the DDRNet23Slim model performs better with higher learning rates on this particular dataset as shown in Figure 9. Lower learning rates, such as $1e-6$ and $1e-5$, did not significantly improve model performance, leading to stagnation in loss reduction and negligible improvements in IoU and Dice metrics. Conversely, the learning rate of $1e-3$ enabled the model to optimize faster and achieve higher accuracy metrics within fewer epochs.

Overall, the DDRNet23Slim model, when trained with an appropriate learning rate, demonstrates the promising potential for segmentation tasks, particularly in scenarios with constrained computational resources but the only limitation is that it took a whole 5 hours just to complete 23 epochs for a sample dataset. This states it takes a long time to complete 18000 images from the Mapillary dataset. So, we avoided DDRNet for now.

G. LinkNet

For the development of our semantic segmentation model, we employed LinkNet [22], constructed using the segmentation-models-pytorch library. The images were transformed to a resolution of 1024x2048 pixels to suit the input requirements of the network. In terms of optimization, we utilized both Jaccard Loss and Dice Loss as our primary loss functions, allowing for the assessment of performance from multiple perspectives. The Intersection over Union (IoU) was used as our evaluation metric, with a threshold set at 0.5, to rigorously measure the overlap between predicted and ground truth segmentations.

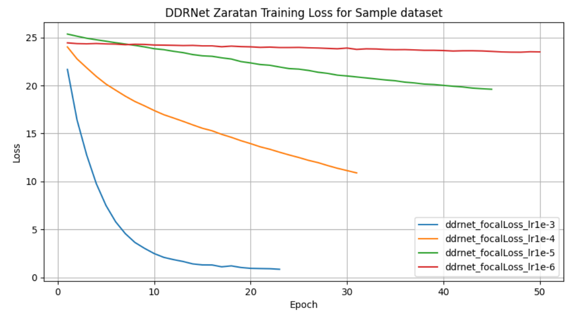

Fig. 9: *Resultant training error on the training dataset after training the DDRNet model for 5 hours on the sample dataset in the Zaratan cluster. This shows that the learning rate* $1e-3$ *will be best to run the model. This also shows that it took only 23 epochs for 5 hours for a sample dataset stating that the model will train very slowly if the whole Mapillary dataset is used.*

The model training was set to run for 25 epochs with a learning rate of 0.0001. We explored two encoder architectures, resnet18 and resnet34, with the latter being selected for final deployment due to its superior performance during preliminary tests. Unfortunately, due to time constraints, we were unable to complete the training and evaluation phases of the model, leaving its full potential untested.

VI. Results

We selected BiSeNet and Fast-SCNN due to their promising performance in real-time semantic segmentation tasks. To evaluate their performance, we developed a code to test these models on a video with a frame rate of 30fps and a duration of 20 seconds. Although the models themselves are fast, the overall processing time slightly exceeded the video duration. This discrepancy is primarily attributed to the image transformation and output writing processes (i/o). Specifically, the video was not initially in the required resolution of 1024x2048, necessitating a resizing step followed by transformation into a tensor format for model input. The processing was segmented into several key operations: reading time, which denotes the time taken to read a frame from the video; transform time, indicating the duration for image transformation; model time, reflecting the time taken by the model to make predictions; and writing time, which involves converting the model output into colored pixels based on the Mapillary dataset for human interpretation. We compiled the results as shown in the Table III

VII. Future Works

While our study has demonstrated promising results in real-time semantic segmentation for autonomous driving using Fast-SCNN and BiSeNet, there are several areas where future research can extend and improve upon our work:

1) ***Exploration of Advanced Architectures:*** Future research could explore the integration of more advanced architectures such as Transformers or hybrid models that combine CNNs with attention mechanisms. These models have shown

Model	Loss Function	Epochs	Reading time	Transform time	Model time	Writing time	Total time
Fast-SCNN	Cross Entropy Loss	60	0.220215797	14.29831839	3.444334507	11.85758781	29.8204565
Fast-SCNN	Cross Entropy Loss with weights	60	0.196428537	14.94667745	3.252543211	16.98035002	35.37599921
Fast-SCNN	Focal loss with $\gamma = 3$	25	0.219221592	15.7236073	3.057817459	15.45706391	34.45771027
Fast-SCNN	Polynomial based Learning Rate	60	0.243033171	14.79372787	3.122980595	13.7604754	31.92021704
BiSeNet	Cross Entropy Loss with weights	60	0.221566677	18.44306016	2.357757807	60.91800642	81.94039106
BiSeNet	Focal loss with $\gamma = 2$	19	0.169178486	18.62406111	3.474766493	59.74437118	82.01237726
BiSeNet	Polynomial based learning rate	52	0.223995447	19.60550523	1.835825682	60.2686708	81.93399715

TABLE III: *Time took the models with our code. The model took the least time whereas the i/o took longer time for the video of 20s with Nvidia RTX 4070 mobile 8GB*

potential in improving segmentation accuracy and could be optimized for real-time applications.

*2) **Transfer Learning Across Diverse Datasets**:* To improve generalization across different driving environments, future research could explore transfer learning techniques that leverage pre-trained models on diverse datasets. Investigating domain adaptation methods to handle variations in geographic locations, weather conditions, and traffic scenarios could further enhance model performance.

*3) **PyTorch Multiprocessing**:* Exploring PyTorch's multiprocessing features could enhance the efficiency of real-time semantic segmentation models. By managing data loading and inference tasks across multiple CPU cores and GPUs, it's possible to reduce latency, a critical factor in autonomous driving. Future research could focus on optimizing workload distribution and minimizing overhead to achieve faster inference times while maintaining accuracy.

*4) **Long-Term Evaluation/Powerful Computing Resources**:* Finally, there is a need for extensive long-term evaluation and testing of these segmentation models in real-world driving scenarios. Future work could involve deploying these models in actual autonomous vehicles and assessing their performance over extended periods, providing valuable insights for further refinement and optimization.

VIII. Conclusion

In conclusion, the findings from our experiments underscore the critical role that model architecture, loss function selection, and learning rate scheduling play in the overall performance and efficiency of image segmentation tasks. The BiSeNet model, known for its robust architecture, was paired with Cross Entropy Loss and a polynomial learning rate, demonstrating the effectiveness of this combination in achieving a high level of accuracy. However, the Fast-SCNN model stood out for its exceptional efficiency, delivering higher processing speeds and maintaining a commendable performance level.

The influence of the loss function on training dynamics cannot be overstated. While more complex loss functions like Focal Loss offer nuanced advantages in handling class imbalances, they also introduce additional complexity into the training process. In contrast, Cross Entropy Loss, with its straightforward implementation and well-understood behavior, emerged as a more practical choice, balancing ease of use with consistent results. This simplicity not only facilitates more predictable training outcomes but also aligns well with optimizing for both speed and efficiency.

Furthermore, the learning rate schedule proved to be a pivotal factor in determining the convergence rate and overall training duration. The polynomial learning rate used with BiSeNet contributed to its accuracy, highlighting how such schedules can be fine-tuned to enhance model performance. Meanwhile, Fast-SCNN's efficiency was partly attributed to its simpler approach, reinforcing the idea that a more streamlined training process can yield significant benefits in contexts where speed is crucial.

Overall, the interplay between these components—model architecture, loss function, and learning rate schedule—illustrates the importance of a holistic approach to model development. By carefully considering each of these elements, it is possible to strike a balance between accuracy and efficiency, ultimately leading to more effective and optimized image segmentation solutions.

IX. Contribution

In the course of this project, **Peeyush** played a pivotal role in ensuring the feasibility and execution of key components. Initially, he evaluated the practicality of his chosen topic, "Automatic Music Generation," assessing its potential for completion within the allocated timeline. His research efforts extended to an in-depth review of various architectures and strategies behind both Real-time and Normal Image Segmentation, which informed the subsequent development of Fast-SCNN and BiSeNet models. Peeyush also played a crucial role in integrating the Mapillary Vistas dataset into the project, converting images to a 1024x2048 format and testing the overfitted codes on local machines. His collaborative efforts with Srijinesh led to the successful utilization of Zaratan clusters, where they developed code for efficient training on multiple GPUs using supergradients, ultimately training various models such as BiSeNet, Fast-SCNN, DDRNet, and PPLiteB50. In addition to the model training and evaluation, Peeyush contributed significantly by developing custom code for calculating the mean Intersection over Union (mIoU), a critical metric for assessing the performance of semantic segmentation models. He also wrote validation scripts to rigorously test the models. To demonstrate the practical applications of the trained models, Peeyush implemented additional code to apply Fast-SCNN and BiSeNet on video data, enabling real-time semantic segmentation. This approach not only validated the models' performance in a controlled environment but also showcased their effectiveness in real-world scenarios, further highlighting their potential for practical deployment. He also contributed significantly to the creation of the presentation and write-up, covering sections on the Problem Statement, Background, Plan of Execution, Metrics, and Results.

Srinivas made significant contributions by exploring advanced project ideas such as transfer learning and identifying relevant models and datasets for image segmentation. After adapting to Zaratan's clusters, he implemented various models, including Fast-SCNN, PPLite T50, STDC, and ENet [23], with assistance from Peeyush. Srinivas experimented with a range of loss functions, learning rates, and learning rate schedulers. He successfully trained various models, including BiSeNet, Fast-SCNN, and PPLite, and played a key role in validating every model's performance, ensuring the robustness and accuracy of the results. These results were communicated and analyzed carefully so that the rest of the team could make crucial decisions in finalizing a suitable model with the remaining resources and time constraints at hand. Additionally, Srinivas conducted an in-depth study of SuperGradients and reviewed various research papers on real-time semantic segmentation. He created a comprehensive write-up on these techniques, which provided valuable insights for the team. He also managed job submissions for the models on the Zaratan cluster and significantly contributed to creating the slides and the write-up for the project.

Hannan was instrumental in the conceptualization and validation of the project's central idea—using semantic segmentation for autonomous driving. He thoroughly reviewed recent research and surveys to stay abreast of advancements in semantic segmentation, with a focus on cutting-edge architectures like YOLO and BiSeNet. His work involved fine-tuning these models, experimenting with hyperparameters to enhance their performance. Hannan collaborated closely with the team to set up and optimize the use of Zaratan clusters for GPU-intensive tasks, ensuring that the training processes were executed efficiently. His technical contributions included the successful integration and format conversion of the Mapillary Vistas dataset, as well as the training of YOLO and BiSeNet models, where he critically analyzed their performance in the context of the project's objectives. Beyond technical work, Hannan also took the lead in drafting significant sections of the project presentation and report, including the Problem Statement, Background, and Deep Learning Models Used, while also ensuring the final deliverables were clear and coherent.

Srijinesh's efforts were crucial in the exploration and selection of the project's final focus area—image segmentation. He began by evaluating various ideas, such as Black and White Image to Color Conversion, and investigated potential datasets like the Image Colorization dataset. After finalizing image segmentation as the project's focus, he explored different datasets, ultimately confirming the Mapillary Vistas dataset for training the models. His collaboration with Peeyush and Srinivas led to the identification of super_gradients as an essential tool for leveraging pretrained image segmentation models. Srijinesh also worked on implementing LinkNet with a pretrained ResNet-34 encoder over the Cityscapes dataset and played a key role in integrating the PPLiteB50 model within the Zaratan cluster, adapting it to process images at 1024x2048 dimensions. His efforts ensured that the project was equipped with robust, scalable solutions for model training and deployment.

REFERENCES

[1] 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.

[2] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[3] M.-S. Park and W.-J. Song, "A new design method of 2-d linear-phase fir filters with finite-precision coefficients," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 41, no. 7, pp. 478–482, 1994.

[4] R. O. Duda, P. E. Hart *et al.*, *Pattern classification*. John Wiley & Sons, 2006.

[5] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 1, 2001, pp. 105–112 vol.1.

[6] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, pp. 721–741, 1984.

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

[8] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015. [Online]. Available: https://arxiv.org/abs/1505.04597

[9] K. A. Kiani and T. Drummond, "Solving robust regularization problems using iteratively re-weighted least squares," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017, pp. 483–492.

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

[11] G. Neuhold, T. Ollmann, S. R. Bulò, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5000–5009.

[12] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2018. [Online]. Available: https://arxiv.org/abs/1708.02002

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

[14] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 fourth international conference on 3D vision (3DV)*. Ieee, 2016, pp. 565–571.

[15] P. Iakubovskii, "Segmentation models pytorch," https://github.com/qubvel/segmentation_models.pytorch, 2019.

[16] S. Aharon, Louis-Dupont, Ofri Masad, K. Yurkova, Lotem Fridman, Lkdci, E. Khvedchenya, R. Rubin, N. Bagrov, B. Tymchenko, T. Keren, A. Zhilko, and Eran-Deci, "Super-gradients," 2021. [Online]. Available: https://zenodo.org/record/7789328

[17] R. P. K. Poudel, S. Liwicki, and R. Cipolla, "Fast-scnn: Fast semantic segmentation network," 2019. [Online]. Available: https://arxiv.org/abs/1902.04502

[18] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," 2018. [Online]. Available: https://arxiv.org/abs/1808.00897

[19] J. Peng, Y. Liu, S. Tang, Y. Hao, L. Chu, G. Chen, Z. Wu, Z. Chen, Z. Yu, Y. Du, Q. Dang, B. Lai, Q. Liu, X. Hu, D. Yu, and Y. Ma, "Pp-liteseg: A superior real-time semantic segmentation model," 2022. [Online]. Available: https://arxiv.org/abs/2204.02681

[20] M. Fan, S. Lai, J. Huang, X. Wei, Z. Chai, J. Luo, and X. Wei, "Rethinking bisenet for real-time semantic segmentation," 2021. [Online]. Available: https://arxiv.org/abs/2104.13188

[21] Y. Hong, H. Pan, W. Sun, and Y. Jia, “Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes,” 2021. [Online]. Available: https://arxiv.org/abs/2101.06085

[22] A. Chaurasia and E. Culurciello, “Linknet: Exploiting encoder representations for efficient semantic segmentation,” in *2017 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, Dec. 2017. [Online]. Available: http://dx.doi.org/10.1109/VCIP.2017.8305148

[23] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, “Enet: A deep neural network architecture for real-time semantic segmentation,” 2016. [Online]. Available: https://arxiv.org/abs/1606.02147