IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 13, No. 4, December 2024, pp. 3987~4002
ISSN: 2252-8938, DOI: 10.11591/ijai.v13.i4.pp3987-4002

Optimisation of semantic segmentation algorithm for autonomous driving using U-NET architecture

Javed Subhedar[1], Mrinal R. Bachute[2], Ketan Kotecha[2]
[1]Department of Electronics and Telecommunication, Symbiosis Institute of Technology, Pune, India
[2]Symbiosis Centre for Applied Artificial Intelligence, Symbiosis International (Deemed University), Pune, India

Article Info

Article history:

Received Dec 13, 2023
Revised Feb 14, 2024
Accepted Apr 18, 2024

Keywords:

Autonomous driving
Car learning to act
Ensemble
Semantic segmentation
U-NET

ABSTRACT

In autonomous driving systems, the semantic segmentation task involves scene partition into numerous expressive portions by classifying and labelling every image pixel for semantics. The algorithm used for semantic segmentation has a vital role in autonomous driving architecture. This paper's main contribution is optimising the semantic segmentation algorithm for autonomous driving by modifying the U-NET architecture. The optimisation techniques involve five different methods, which include; no batch normalisation network, with batch normalisation network, network with reduction in filters, average ensemble network, and weighted average ensemble network. The validation accuracy observed for the five methods were 90.28%, 91.68%, 89.80%, 92.04%, and 92.21% respectively. By reducing the filters in the network, the computation time reduces (Epoch time: 1 s 64 ms/step) as opposed to the typical (Epoch time: 4 s 260 ms/step), but the accuracy reduces. The optimisation techniques were evaluated for metrics like mean intersection over union (IoU), IoU for class, dice-metric, dice_coefficient_loss, validation loss, and accuracy. The dataset of 300 images used for this paper's study was generated using the open-source car learning to act (CARLA) simulator platform.

Corresponding Author:

Mrinal R. Bachute
Symbiosis Centre for Applied Artificial Intelligence, Symbiosis International (Deemed University)
Pune, India
Email: mrinal.bachute@sitpune.edu.in

1. INTRODUCTION

Software simulation can expedite the research process and reduce the cost of research in autonomous urban driving. This paper uses the car learning to act (CARLA) simulator platform to generate a data set of images. CARLA is an open-source simulation software platform with unreal engine 4 for research in autonomous driving [1]. CARLA was established from a bottom-up approach to develop, train, and validate "autonomous driving systems" for the urban area. CARLA simulator works as a server-client system, as shown in Figure 1. The software simulation platform supports flexible sensors and environmental conditions [2]. CARLA permits the flexible configuration of the agent and sensors. It comprises aesthetically premeditated townships with pedestrian road traffic [3]. Town 1 is aimed at training, while Town 2 is utilised solely for testing. CARLA offers additional statistics like distance covered, collisions, and infractions manifestation like gist onto the reverse lane or a footpath [4]. CARLA provides a means for developing and training autonomous systems and evaluating them in controlled situations [5]. In this work, we use the CARLA simulator software platform to spawn the vehicle with sensors in the town to gather semantic segmentation data.

So as to identify the different objects as seen by the autonomous driving system, semantic segmentation is necessary to differentiate various categories or classes of objects like trees, cars, and pedestrians. Semantic segmentation is partitioning an image into many segments called pixels. Semantic segmentation predicts whether each pixel belongs to a particular class. Semantic segmentation is complex because of scene complexity, complicated object boundaries, and small objects.

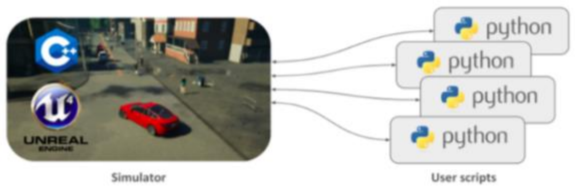


Figure 1. CARLA simulator-client-server [6]

There are many deep-learning models for semantic segmentation which is part of perception task of the autonomous driving system [7]. Some of these models, which include the U-Net Model, PSPNet architecture, DeepLab, PPANet, and SegNet architecture, are discussed. This paper focuses on the UNET model for semantic segmentation for the CARLA image data set and the comparative study of optimisation techniques by building and testing the models in Keras.

2. RELATED WORK

The advanced semantic segmentation system consists of three essential mechanisms: i) a full convolutional network, primarily announced via substituting the end-limited fully connected layers with convolution layers to build effective end-to-end learning with the implication that can receive random input size; ii) conditional random fields, to gather resident, extended-range dependencies in a scene to improve the estimated map; and iii) dilated convolution (or atrous convolution), meant for enhancing the tenacity of in-between feature maps to produce extra precise estimates while up-holding the exact computational rate [8]. Semantic segmentation can be categorised as two-stage and one-stage conduits. In the two-stage conduit, region proposals are initially produced and then tweaked primarily for instance-level segmentation. The one-stage conduit using a fully convolution network is the other standard semantic segmentation method. Image semantic scene segmentation is achieved by depicting single images using supervised and unsupervised techniques. Different approaches are used for semantic segmentation. The region-based semantic segmentation approach first extracts freeform regions belonging to an image and designates them, trailed by region-based classification [9]. In this approach, the whole region features and foreground features improve results when concatenating them collectively as the region of the feature. Drawbacks of this approach include the feature needing to be better suited to the segmentation mission, requiring more spatial information for exact boundary generation, and taking more time, which could significantly affect the final performance. For semantic segmentation, a full convolutional network is another approach that can be used. In this approach, a full convolutional network acquires the mapping from pixel to pixel without removing the region proposals. Since a full convolutional network only has convolution and pooling layers, this approach has the merit of making predictions on arbitrary-sized inputs. The drawback of this approach is that direct predictions of a fully convolutional network are typically in low resolution. The role of semantic segmentation in critical tasks of self-driving technologies is essential. For example, through the path planning phase, it is crucial to understand an object's pose, magnitude, and distance and decide whether to apply brakes or negotiate with the obstacle. There are many semantic segmentation model architectures. Some of them are discussed in the following sections.

2.1. U-NET semantic segmentation architecture

The UNet model was designed for semantic segmentation of bio-medical images [10]. The UNet architecture was also used for dental x-ray image segmentation. The U-Net is a full convolution neural network with elegant architecture. Since it has a U shape, as shown in Figure 2, the architecture is called U-Net. The "U-Net" architecture has two paths: the left and right. The image input is at the topmost left, and the segmentation map of output is at the top right. The input is through the contraction path, down-sampled, and then it goes to the expansion path, where the image is up-sampled. In between, there is a skip connection for dropouts to avoid overfitting. The path on the left is also called the encoder path, and the path on the right is the decoder path. The layers accomplish further implicit data augmentation for dropout at the culmination of the contracting path [11].

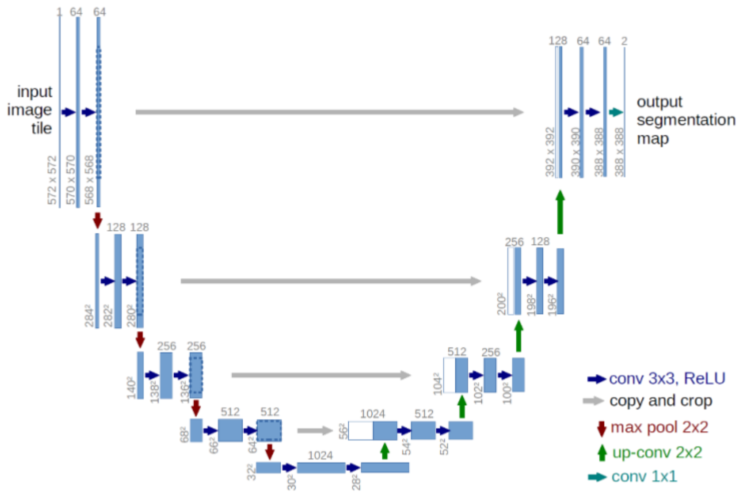


Figure 2. U-Net architecture [12]

The network has no fully connected layers. It applies that the practical portion of every convolution encompasses pixels aimed at the entire framework and is obtainable in the given image. The UNet uses a network and a strategy for training that depends on data augmentation to use the existing annotated samples [10]. The path for contracting uses the distinctive style of a convolution network. It is made of the recurrent use of dual 3 by 3 convolutions, individually succeeded by one rectified linear unit (ReLU) and one 2 by 2 max-pooling of stride 2 meant for down-sampling. At every down-sampling phase, the quantity of feature channels is twice. Each step of the pathway of expansion has an up-sampling of the feature map succeeded through a 2 by 2 convolution, which divides the feature channel, a concatenation with the congruently gathered feature to map out of the path of contraction, and a dual 3 by 3 convolutions, individually succeeded by one ReLU. Cropping is essential as edge pixels are lost in each convolution. A 1×1 convolution map for every 64-component feature vector is the chosen sum of classes at the final layer. Overall, 23 convolutional layers are included in the network. The U-Net network's advantages include higher accuracy given a suitable dataset and training time. U-Net uses a fully convolutional network without contingent on the input size. As a shortcoming, the dimensions of the U-Net must be equivalent to the dimensions of the features. U-Net uses various layers to increase the training time [13].

Another variant of U-Net, termed smoke-UNet, is built on UNet architecture. Smoke-UNet uses an attention mechanism along with the residual block [14]. The residual block improves the feature learning ability of the network. U-Net is successfully applied in many applications for medical image analysis. Two different encoders and one standard decoder are used [15]. Ghost-Net can be combined with U-Net to form a new model architecture called Ghost-UNet. This model combines low-level spatial information and high-level feature maps [16]. Another version of U-Net called the more-residual (MR) U-Net is used for commodity segmentation. This model enhances the model's ability to extract details of goods. This model uses bilinear interpolation instead of deconvolution in the U-Net model to reduce the checker-board effect [17] concatenated residual attention (CRA) U-Net combines residual structure and channel attention mechanism [18]. The convolutional block channel attention (CBCA) module enhances the extraction of deep convolution features and replaces the skip connection in the UNet model. Faster heterogeneous image (FHI)-U-Net uses an autoencoder and decoder for more rapid heterogeneous image segmentation. This structure has fewer layers, resulting in lower parameters and adding high inference speed. FHI-U-Net uses autonomous feature extraction for the images [19]. UNet++ has U-Nets of variable depths. The decoders are connected at the exact resolution by the modified skip connections. The UNet++ has a UNet of varying depth, so it is not susceptible to the choice of the network [20]. The swin transformer boosted U-Net (ST-Unet) uses a swin transformer into a convolutional neural network (CNN) based UNet. It consists of an innovative twin encoder structure of swin transformer and CNN [21]. The Eff-UNet uses the effectiveness of EfficientNet as the encoder for feature extraction and the UNet decoder for reconstructing the segmentation map. This architecture uses high-level and low-level features and spatial info for segmentation [22]. UNet++ can also be used for mapping the cattle rib-eye area. The UNet++ architecture thus can be used to segment cattle rib-eye areas in ultrasound images [23]. The enhancement of ultraviolet based generative adversarial networks (UV-GAN) can be done using U-Net. Boosting of each skip connection is done using attention gates. The irrelevant low-level information is suppressed from the encoders [24].

2.2. Pyramid scene parsing- network

Scene parsing depends on semantic segmentation and knows the class label of every pixel in the image. The pyramid scene parsing-network (PSP-Net) architecture is demonstrated in Figure 3. The PSPNet has 4 parts: input image as shown in Figure 3(a), feature map as shown in Figure 3(b), pyramid pooling module as shown in Figure 3(c), and final prediction as shown in Figure 3(d). In PSPNet, for a given image, the CNN is utilised to get a feature map for the end layer; after that, a module of pyramid parsing is used to get the diverse sub-region illustrations [25] and is trailed by up-sampling and concatenation layers to reach the end feature illustration that contains local and global background info in the pyramid parsing module. The sub-region regular pooling is achieved for every feature map.

The "pyramid pooling module" combines features for four pyramid measures. The red is the bristliest level that performs global average pooling above individual feature maps to produce a one-bin output. The second is the orange, which splits the feature map into 2 by 2 regions and then accomplishes average pooling for every sub-region. The third is blue, which divides the feature map into 3×3 regions and performs average pooling of every sub-region. The fourth is green, the most advanced level; the feature map is split into 6×6 sub-regions, then performs pooling for every sub-region. The bristliest level indicated in red is "global pooling" to make the output of one bin. The subsequent "pyramid level" disjoins the feature map in distinct sub-regions, forming pooled illustrations of other positions. The outcome of diverse classes in the "pyramid pooling module" has a feature map with different magnitudes. Then 1 by 1 convolution is achieved for every pooled feature map to lessen the background illustration to 1/N of the unique one (black) in case the level size of the pyramid is N. Here, N=4 as there are four levels overall (red, orange, blue, and green). Then, bilinear interpolation is achieved to up-sample every small dimension of the feature map to get the equal dimensions as the unique feature map (black). All dissimilar levels of up-sampled feature maps are concatenated by the unique feature map (black). These feature maps are combined as global prior, which is the termination of the pyramid pooling module. It is then given to a convolution layer to get the last per-pixel prediction. Hence, a superior framework for pixel-level prediction is provided by PSPNet.

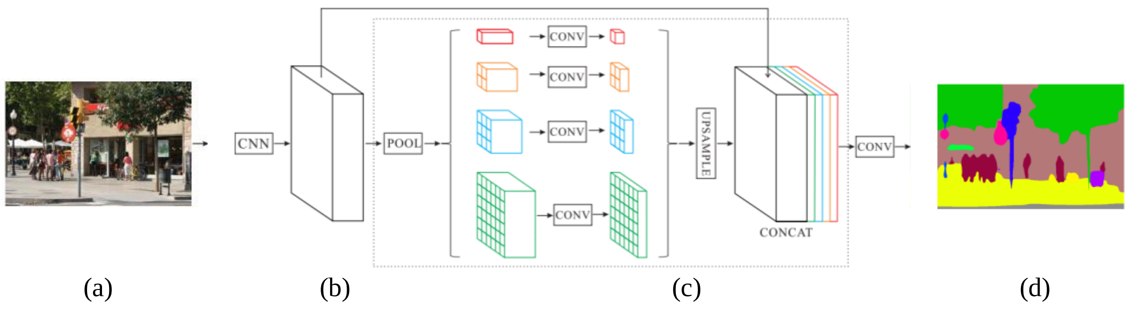


Figure 3. PSP-Net architecture, (a) input image, (b) feature map, (c) pyramid pooling module, and (d) final prediction [25]

2.3. Deeplab semantic segmentation

DeepLab, an advanced model for the semantic segmentation of images, aims to allocate semantic tags (e.g., human being, canine, and cat) to each pixel in an input image. Semantic segmentation DeepLab has the following versions:

- DeepLabv1: this version uses the atrous convolution to regulate the tenacity of feature retorts in the deep convolution neural networks.
- "DeepLabv2": this version uses the atrous spatial pyramid pooling shown in Figure 4 to vigorously segment items at several magnitudes using filters at various sampling rates and actual field of view (FOV).
- "DeepLabv3": the atrous spatial pyramid pooling utilising an image-level feature in this version is used to apprehend long-range info. Likewise, batch normalisation parameters are included to enable the training. In particular, applying atrous convolution at diverse output strides to get output features through training and estimation proficiently permits training at an output stride equal to 16. It accomplishes a high performance at an output stride equal to 8 during the assessment.
- DeepLabv3+: in this version, the DeepLabv3 is extended to comprise a modest but efficient decoder module to improve the segmentation outcomes, specifically with item limitations. Moreover, we can indiscriminately regulate the tenacity of mined encoder features using atrous convolution to compromise

runtime and accuracy in the encoder-decoder [13] edifice. DeepLabv3+ outspreads DeepLabv3 by adding a modest, hitherto practical decoder module to improve segmentation, mainly item borders. DeepLabv3+ has high semantic information from the encoder module, whereas the meek yet active decoder module recovers the precise item borders. The encoder module permits extracting features at a subjective tenacity using atrous convolution [26]. The DeepLab V3+ semantic segmentation provides segmentation frames for model input in the natural environment.

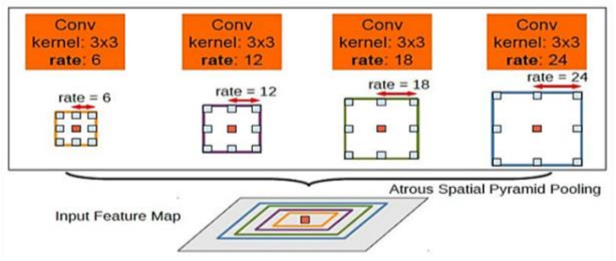

Conv kernel: 3x3 rate: 6
Conv kernel: 3x3 rate: 12
Conv kernel: 3x3 rate: 18
Conv kernel: 3x3 rate: 24
rate = 6
rate = 12
rate = 18
rate = 24
Atrous Spatial Pyramid Pooling
Input Feature Map

Figure 4. Atrous spatial pyramid pooling [27]

The DeepLab model is indicated in Figure 5. Visual geometry group (VGG)-16 or ResNet-101, which is proficient in the mission of image sorting, is used in the study of semantic segmentation through; i) changing completely the fully connected layers to convolutional and ii) growing feature tenacity by atrous convolutional layers, permitting it for calculating feature retorts each of 8 pixels in place of each of 32 pixels in the initial net. After that, a bilinear interpolation is used to up-sample by the score map factor of eight to grasp the unique image resolve, getting the input to a CRF and enhancing the segmentation outcomes. The three critical merits of the DeepLab V1 system are: the speed with atrous convolution, accuracy, and simplicity.

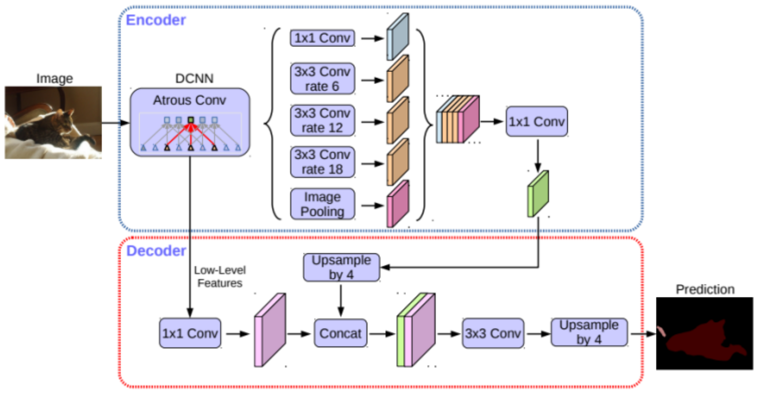


Figure 5. Structure of the Deeplabv3+ network [28]

DeepLabv3 enhances the atrous spatial pyramid pooling module. The post-processing step CRF is not used in DeepLab V3 but was initially used in DeepLabv1 and DeepLabv2. DeepLabv3 uses atrous convolution with up-sampled filters to get dense feature maps and apprehend extensive range context. In DeepLabv3, the connected module progressively duets the atrous rates for encoding multiscale info. In contrast, the atrous spatial pyramid pooling module is improved by image-level features that review the features utilising filters at various sampling rates and active field-of-views. In DeepLabv3, the ground truths are kept intact. DeepLabv3 expressively advances above the earlier DeepLab versions short of CRF post-processing and attains comparable performance.

DeepLabv3+ outspreads DeepLabv3 by accumulating a modest hitherto active decoder module for improving segmentation, particularly the object boundaries. DeepLabv3+ comprises amusing semantic info from the encoder module, whereas the fair, hitherto functional decoder module recovers the precise item borders. The encoder module permits extracting features at a subjective resolution using atrous convolution. Adding a meek, heretofore practical decoder module for retrieving the object borders DeepLabv3+ outspreads

DeepLabv3. At the end of DeepLabv3, the enhanced semantic info is encoded by atrous convolution, permitting unique regulation of the concentration of the encoder features, relying on reasonable calculation means.

Moreover, the decoder consents to complete item boundary retrieval. DeepLabv3+ augments the decoder module over and above the output. For the decoder, three places of diverse design options are deliberated, explicitly, i) the 1×1 convolution utilised to lessen the channels for low-level feature map within the encoder, ii) the 3 by 3 convolution meant to acquire strident segmentation outcomes, and iii) which low-level encoder features ought to be utilised. Deeplabv3+ can semantically segment synthetic aperture radar (SAR) images [28]. The model, Deeplabv3+, with a cross-attention means, can overcome the deficits of accurately extracting the intended features of the image edge. Similarly, it simulates the affiliation amongst the local features of the large-scale target to facilitate a whole singularity in the excellent-scale target segmentation.

The module permits extracting features at a subjective resolution by using atrous convolution. DeepLabv3+ outspreads DeepLabv3 by adding a meek, hitherto effective decoder for recovering the object borders. The enhanced semantic info is encoded at the end of "DeepLabv3" by atrous convolution, permitting one to regulate the density of the encoder features, reliant on a budget of computation; moreover, the decoder module consents to complete item boundary retrieval.

2.4. Point-wise pyramid attention network

The encoder and decoder methods are used in point-wise pyramid attention network (PPANet), aiming at semantic segmentation [26]. The encoder assumes an innovative squeeze non-bottleneck to reference excerpt feature depictions wherever squeeze and expansion are used to get higher segmentation precision. PPANET architecture is shown in Figure 6.

An up-sampling module is premeditated for behaving as a decoder; its resolve is to get missing pixelwise depictions within the encoding block. The point-wise pyramid attention (PPA) and an attention-like module linked in similar depictions form the central portion. Similarly, the decoder is utilised for mending the equivalent illustrations. The squeeze-nbt part shown in Figure 7 is the vital building chunk. The squeeze-nbt (squeeze non-bottleneck) module is built upon a condense-divide-squeeze-combine approach. Point-wise convolution is used initially by the squeeze-nbt module to lessen the feature maps, and then a parallel fire module is applied to study valuable illustrations. The network consists of two additional portions: the PPA and the attention module implanted amid the encoder beside the decoder. The modules in the centre are utilised to augment the amenable field and deliver adequate background info in distinct phases of the encoder portion. Every stage within the encoder consumes dual chunks of the squeeze-nbt component, and the feature map is down-sampled by half at every starting block at every phase utilising stride convolution. The modules in the centre have been used to augment the amenable field and deliver enough background info.

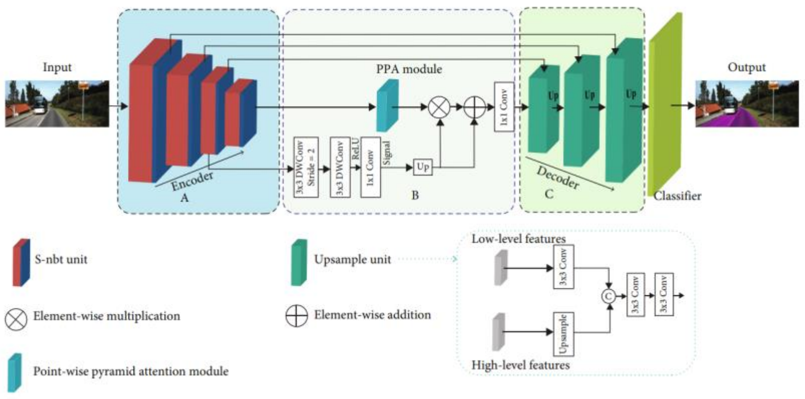


Figure 6. PPANET architecture [26]

The PPA element illustrated in Figure 8 adequately gathers global background info. The PPA comprises dual chunks: the nonlocal portion and vortex pooling. The nonlocal unit produces impenetrable pixel-wise weight and excerpts extensive range reliance. The vortex atrous convolution helps spot an item at various scales. The two modules' recompenses are joined in a single PPA module by analysing the vortex pooling beside the nonlocal dependence. The tri-parallel vortex atrous convolution chunks of distention rates of 3, 6, and 9 and one nonatrous convolution chunk are included in the PPA. The PPA is used to utilise contextual information effectively.

The PPA module enriches the receptive field and provides sufficient context information. However, the limited memory problem has a drawback, which is caused by multiple downsampling and upsampling and storage of parameter values at every step. Also, it is challenging to sustain local details as the whole image is unified into the network.

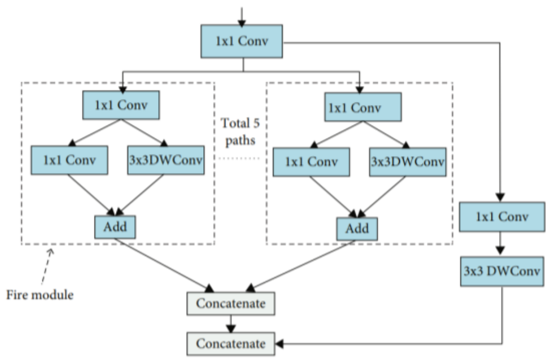


Figure 7. Squeeze-nbt module [26]

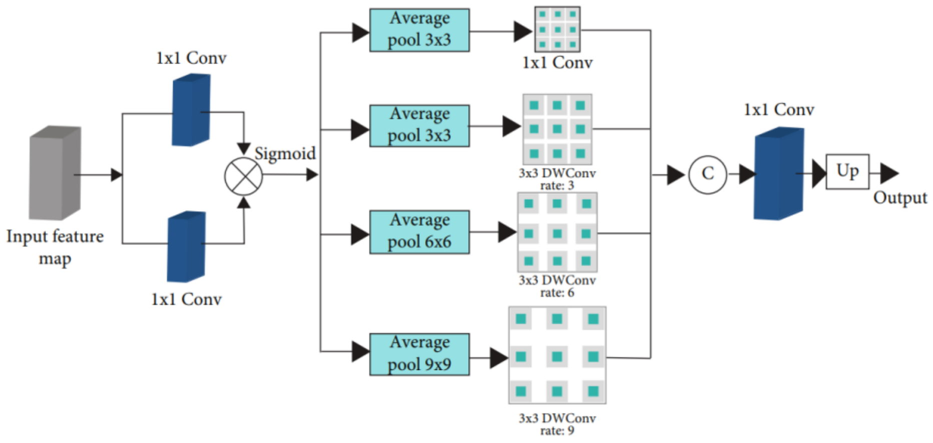


Figure 8. PPA module [26]

2.5. Segnet architecture -semantic segmentation

For an effective architecture of pixel-wise semantic segmentation, the SegNet is premeditated. SegNet shown in Figure 9 is the decoder net that comprises an order of decoders, individuals conforming to every encoder [29]. The decoders utilise the max-pooling metrics established by the equivalent encoder to accomplish nonlinear upsampling of the in-take feature maps. An encoder and connected decoder networks form the SegNet and are trailed by a concluding pixel-wise sorting layer.

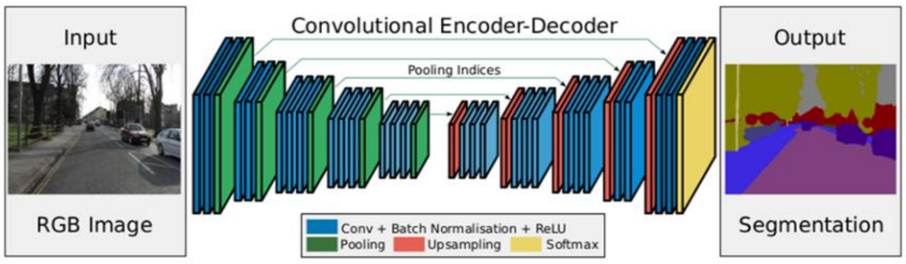


Figure 9. SegNet architecture [29]

The thirteen convolutional layers corresponding to the initial convolutional layers in the VGG16 network intended for entity classification are included in the encoder network. Every encoder part has a conforming decoder layer; thus, the decoder network consumes thirteen layers, and to yield class likelihoods of every pixel independently, the last decoder output is given to a multiclass soft-max classifier. Convolution by a filter group for producing a group of feature maps is achieved for every encoder within the network. Afterwards, batch normalising is done, and an element-wise ReLU max (0; x) is used. Succeeding that, max-pooling by 2×2 size window along with stride of 2 (not overlapping window) is achieved, and the resultant output is sub-sampled with a factor of 2. Max pooling accomplishes transformation invariance above minor spatial modifications within the input image; the sub-sampling ends in a hefty input image framework for every pixel within the map.

Upsampling the input feature map utilising the learned max-pooling measures from the equivalent encoder feature map is done by a suitable decoder in the decoder network. The result of this stage is a sparse feature map. The feature maps are convolved with a decoder filter bank to create dense feature maps. A batch normalisation step is then applied to each map. The decoder equivalent to the opening encoder generates a feature map with many channels, even though its encoder input takes three channels (red, green, and blue). Distinct from the additional decoders in the net, this makes feature maps through sizes and channels similar to the encoder inputs. A soft-max classifier is used for higher-dimensional feature illustration at the end of the ultimate decoder. The soft-max classifier categorises every pixel individually. The K channel image of probabilities k indicates that the total sum of classes is the output from the soft-max classifier. The foretold segmentation resembles the class through extreme probability at every pixel. As only the max-pooling indices of the feature maps are saved, the SegNet is efficient.

3. METHOD

The methodology involves the following steps: the CARLA simulation and the U-Net implementation for different optimisation techniques. First, the client is implemented in Python with the sensor at the appropriate coordinates on the vehicle. The vehicle is spawned once the client runs. During the simulation, the images are captured, stored in the local directory, and act as the dataset for the algorithm implementation. The methodology diagram is shown in Figure 10.

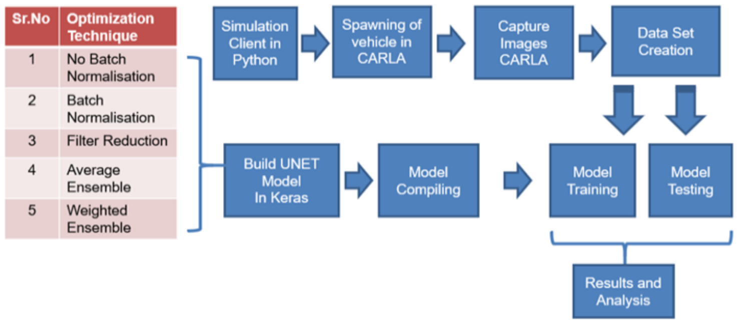
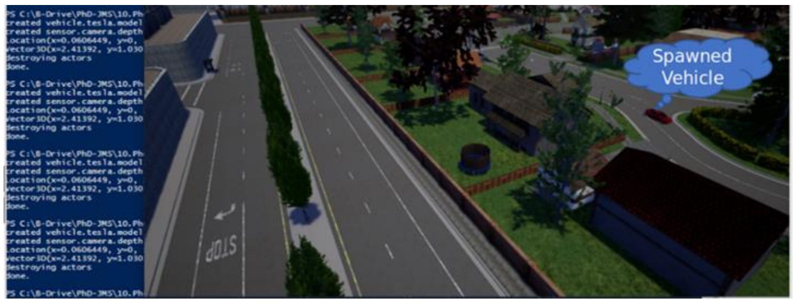


Figure 10. Methodology diagram

The blocks in the methodology diagram are explained:
- Simulation of client in Python: the client side has client modules that control the logic of actors on the scene and set the world (surrounding environment) conditions. The client side is implemented in Python.
- Spawning of the vehicle in CARLA: the Tesla model available in the CARLA simulator is used and is spawned at the defined position, utilising the x, y, and z map coordinates. Spawning requires a blueprint of the selected model. It is the ego vehicle, meaning the user controls the vehicle.
- Images captured CARLA: the images are captured using the RGB module in CARLA. The images are captured from the particular scene in the simulation. The RGB sensor has a listen () method, which is called every time the sensor retrieves data. Different RGB camera attributes are programmable during simulation.
- Dataset of images: images captured during the simulation from the dataset.
- UNET algorithm: the full convolution neural network required for the UNET algorithm is implemented using the Keras platform in Python. Different optimisation techniques are used to modify the architecture.
- Results and analysis: the training, validation, and testing data results are analysed for training and validation accuracy and loss parameters, along with other metrics used for semantic segmentation.

3.1. Car learning to act simulation

The CARLA version 0.95 for Windows 10 is used for simulation. The CARLA client is implemented in the Python 3.7 version. The spawning of the vehicle is shown in Figure 11. The camera view is captured as shown in Figure 12. The data set is created using the RGB camera on the spawned vehicle and saved in the local directory. The size of the images stored is 128×128. The building of the data set is shown in Figure 13.



```
PS C:\B-Drive\PhD-JMS\10.Ph
created vehicle.tesla.model
created sensor.camera.depth
Location(x=0.0606449, y=0,
Vector3D(x=2.41392, y=1.030
destroying actors
done.
PS C:\B-Drive\PhD-JMS\10.Ph
created vehicle.tesla.model
created sensor.camera.depth
Location(x=0.0606449, y=0,
Vector3D(x=2.41392, y=1.030
destroying actors
done.
PS C:\B-Drive\PhD-JMS\10.Ph
created vehicle.tesla.model
created sensor.camera.depth
Location(x=0.0606449, y=0,
Vector3D(x=2.41392, y=1.030
destroying actors
done.
PS C:\B-Drive\PhD-JMS\10.Ph
created vehicle.tesla.model
created sensor.camera.depth
Location(x=0.0606449, y=0,
Vector3D(x=2.41392, y=1.030
destroying actors
done.
PS C:\B-Drive\PhD-JMS\10.Ph
```

Figure 11. Spawning of vehicle in CARLA

Figure 12. CARLA simulation-camera view

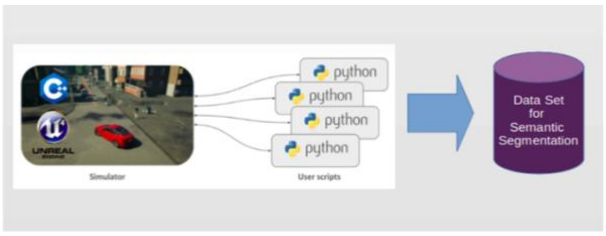


Figure 13. CARLA simulation-dataset creation

3.2. U-NET architecture optimisation

The U-NET implementation is done with the help of the Keras framework [30] in the Google Colab online platform. The number of classes for semantic segmentation is 11. Since it is a multiclass classification, the SoftMax activation function was used. Following optimisation techniques were implemented.

- No batch normalisation: in this technique, batch normalisation is not used after the convolution block for the four stages of the encoding path in the UNET network.
- Batch normalisation: batch normalisation assists in normalising the outputs of the four convolution layers to get a mean of zero and a standard deviation of 1. Batch normalisation helps the model to regularise and learn faster.
- Filter reduction: in this technique, the UNET architecture is modified to reduce the filter in the encoding path of the architecture, as shown in Figure 14. The number of filters is reduced from 64 to 16 for each size 3×3, 128 to 32 for each size 3×3,256 to 64 for each size 3×3 and 512 to 128 for each size 3×3, respectively. This technique helps reduce computational time. When an image is convolved with filters, the dimensions of the image decrease.
- Average ensemble: the ensemble technique [31] is a method of combining different models. The multiple sub-models contribute similarly to a combined prediction. The averaging ensemble model syndicates the predictions from multiple trained models. It takes the average of all model predictions and gives the final prediction. The limitation of this technique is that each model contributes the same amount to the ensemble prediction, irrespective of how well the model performed. The three models considered for average ensemble are model (M1)-UNET architecture without batch normalisation, model (M2)-UNET architecture with batch normalisation, and model (M3)-UNET architecture with reduced filters. The average ensemble assumes equal weights to all the models. Figure 15 demonstrates the average ensemble modelling technique.
- Weighted ensemble: the technique of weighted ensemble assumes that some models have more prediction skills when compared to others. It means that each model has its advantages and disadvantages. Thus, the models with higher skills are given more weightage. The weighted ensemble is an extension of the average ensemble, which assumes equal weights for all the models. In this paper, the three models considered are M1-UNET architecture without batch normalisation, M2-UNET architecture without batch normalisation,

and M3-UNET architecture with reduced filters. The weights associated with M1, M2, and M3 are W1, W2, and W3 respectively. These fixed weights are multiplied by the prediction made by the model and used in average prediction calculation. Figure 16 represents the weighted average ensemble modelling technique.

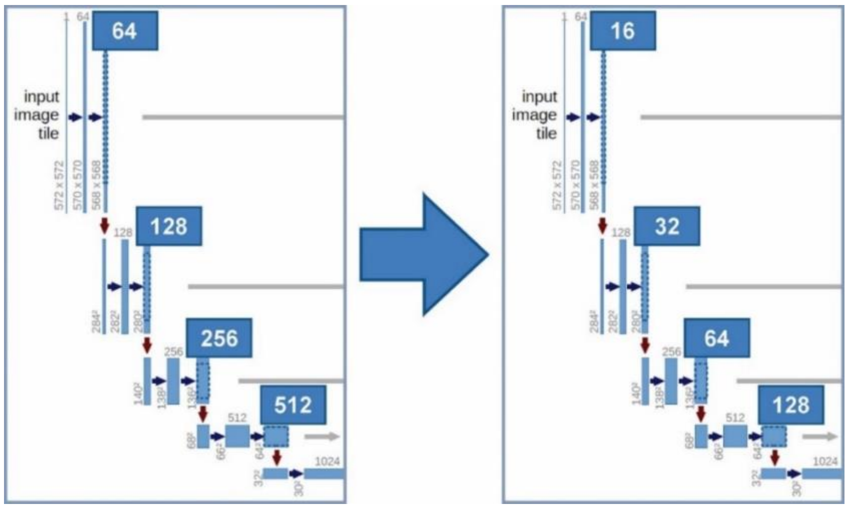


Figure 14. Filter reduction technique [12]

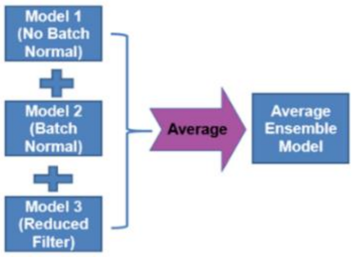


Figure 15. Average ensemble modelling technique

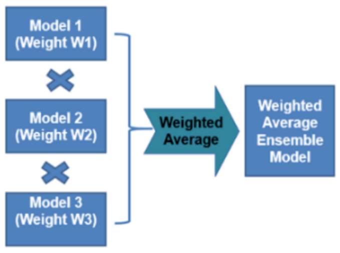


Figure 16. Weighted average ensemble modelling technique

4. RESULTS AND DISCUSSION

The image dataset obtained by CARLA simulation is used for the UNET Model optimisation techniques. Total images of 300 along with their masks are generated. The simulation details and results observed after implementing the UNET Model optimisation techniques are demonstrated in following sections.

4.1. Car learning to act simulation results

Using the CARLA simulation platform images are generated. The CARLA simulation is carried out to generate 300 images. Along with these images their masks are also generated. Once the dataset is generated, the training (240 images) and validation (60 images) distribution is done as shown in Table 1.

Table 1. Dataset distribution

Parameter	Numbers
Total CARLA images	600
Original image data size	300
Mask image data size	300
Training data size	240
Validation data size	60

4.2. UNET optimisation results

The five techniques for optimising the semantic segmentation using UNET model implementation are carried out for 100 epochs. The max pixel value in the image was observed to be 231. A batch size of 30 was selected.

– No batch normalisation: the training loss reduces from 3.8962 to 0.1151 after 100 epochs. The validation loss reduces from 2.1547 to 0.4084 after 100 epochs. The graphs for training and validation losses for 100 epochs are shown in Figure 17. The training accuracy increased from 36.27% to 95.71% after 100 epochs. The validation accuracy rose from 39.52% to 90.28% after 100 epochs. The graphs for training and validation accuracy increase are shown in Figure 18. Figure 19(a) shows the test image, Figure 19(b) shows the ground truth label, and Figure 19(c) shows the prediction on the test image after the model is trained for 100 epochs. The mean IoU observed was 0.46047738 after 100 epochs. The dice metric was 0.8832081, and the dice_coefficient_loss was 0.116791904.

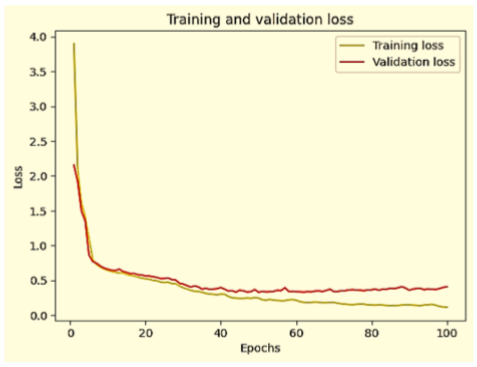


Figure 17. Training and validation loss after 100 epochs without batch normalization technique

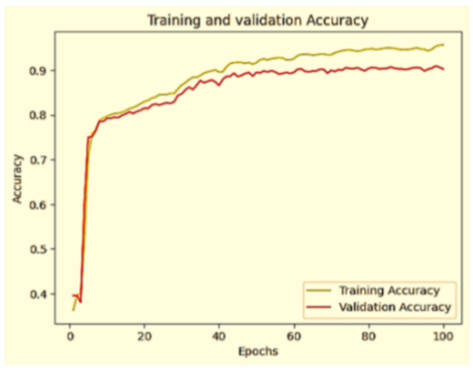


Figure 18. Training and validation accuracy after 100 epochs without batch normalization technique

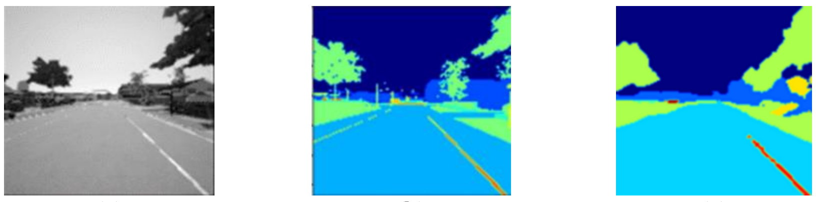


Figure 19. Comparison of (a) test image, (b) ground truth, and (c) predicted image for without batch normalization technique

– With batch normalisation, the training loss was reduced from 1.2546 to 0.0807 after 100 epochs. The validation loss reduces from 5.974 to 0.3011 after 100 epochs. The graphs for training and validation losses are shown in Figure 20. The training accuracy increased from 65.33% to 96.89% after 100 epochs. The validation accuracy rose from 39.52% to 91.61% after 100 epochs. The graphs for training and validation accuracy increase are shown in Figure 21. Figure 22(a) shows the test image, Figure 22(b) shows the ground truth label, and Figure 22(c) shows the prediction on the test image after the model is trained for 100 epochs.

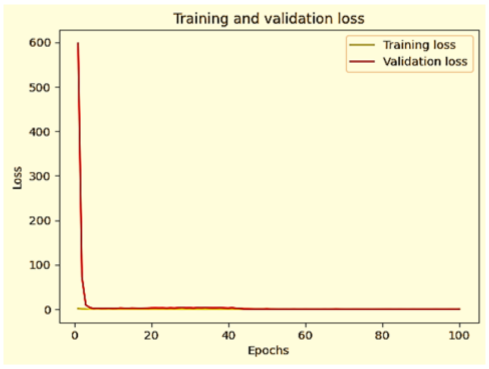


Figure 20. Training and validation loss after 100 epochs with batch normalization technique

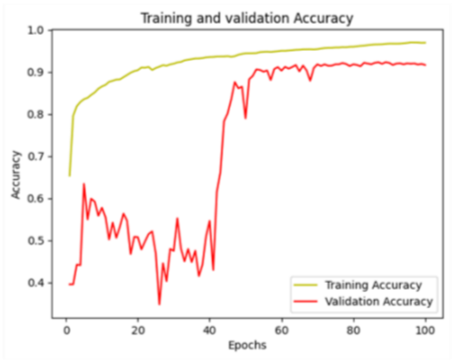


Figure 21. Training and validation accuracy after 100 epochs with batch normalization technique

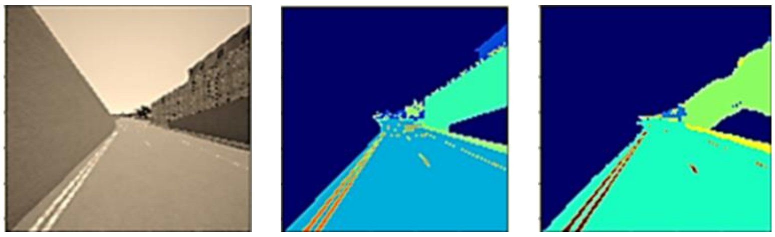

(a) (b) (c)

Figure 22. Comparison of (a) test image, (b) ground truth, and (c) predicted image with batch normalization technique

The mean IoU observed was 0.48857138 after 100 epochs. The dice metric was 0.9041302, and the dice_coefficient_loss was 0.09586978.

- Filter reduction: the training loss reduced from 2.1707 to 0.1299 after 100 epochs. The validation loss reduced from 1.7368 to 0.3923 after 100 epochs. The graphs for training and validation losses are shown in Figure 23. The training accuracy increased from 30.22% to 95.26% after 100 epochs. The validation accuracy rose from 37.32% to 89.80% after 100 epochs. The graphs for training and validation accuracy increase are shown in Figure 24. Figure 25(a) shows the test image, Figure 25(b) shows the ground truth label, and Figure 25(c) shows the prediction on the test image after the model is trained for 100 epochs. The mean IoU observed was 0.4433159 after 100 epochs. The dice metric was 0.88062567, and the dice coefficient loss was 0.119374335.
- Average ensemble: the training loss reduced from 0.1145 to 0.0339 after 100 epochs. The validation loss increases from 0.2686 to 0.4325 after 100 epochs. The graphs for training and validation losses are shown in Figure 26. The training accuracy rose from 95.74% to 98.44% after 100 epochs. The validation accuracy increased from 91.69% to 92.10% after 100 epochs. The graphs for training and validation accuracy increase are shown in Figure 27.

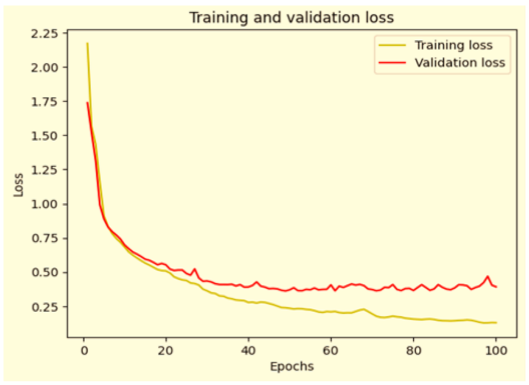


Figure 23. Training and validation loss after 100 epochs (filter reduction technique)

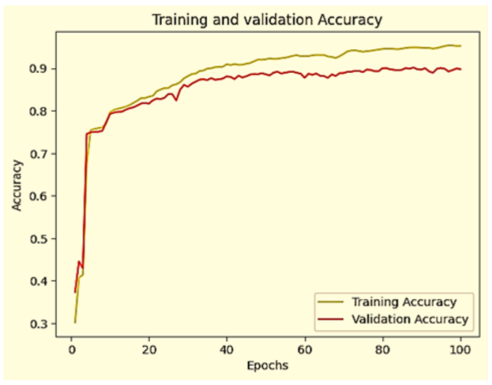


Figure 24. Training and validation accuracy after 100 epochs (filter reduction technique)

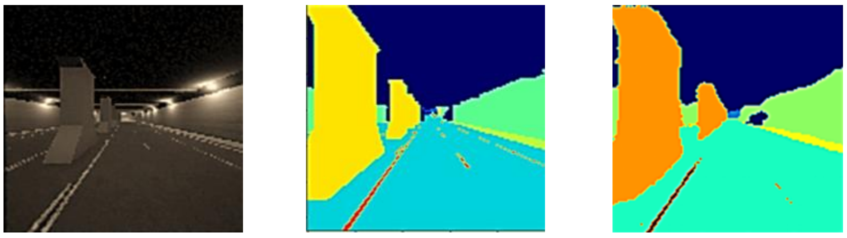

(a) (b) (c)

Figure 25. Comparison of (a) test image, (b) ground truth, and (c) predicted image for filter reduction technique

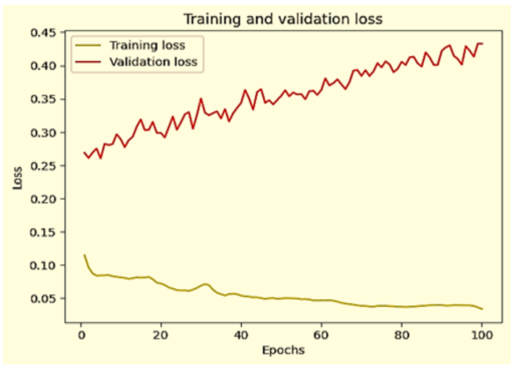


Figure 26. Training and validation loss after 100 epochs (average ensemble technique)

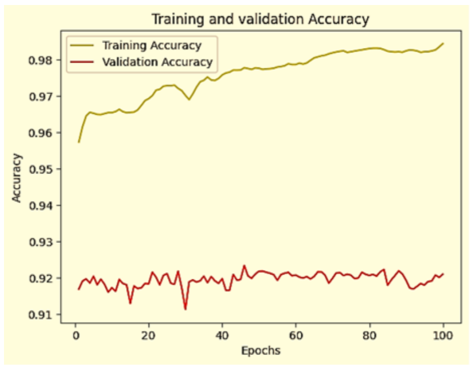
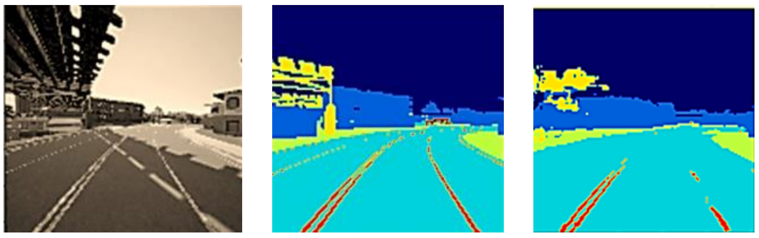


Figure 27. Training and validation accuracy after 100 epochs (average ensemble technique)

Figure 28(a) shows the test image, Figure 28(b) shows the ground truth label, and Figure 28(c) shows the prediction on the test image after the model is trained for 100 epochs. The mean IoU observed was 0.5070416 after 100 epochs. The dice metric was 0.9050604, and the dice_coefficient_loss was 0.09493959.



(a) (b) (c)

Figure 28. Comparison of (a) test image, (b) ground truth, and (c) predicted image for average ensemble technique

– Weighted average ensemble: since model 2 has the highest accuracy, the weight assigned is 0.6; the next highest accuracy is for model 3, so the weight given is 0.3, and for model 2, the weight shown is 0.1. For this combination of weights, the weighted average ensemble model [32] training loss reduced from 0.1038 to 0.0219 after 100 epochs. The validation loss increases from 0.2557 to 0.4315 after 100 epochs. The graphs for training and validation losses are shown in Figure 29. The training accuracy rose from 96.32% to 99.39% after 100 epochs. The validation accuracy increased from 91.39% to 91.90% after 100 epochs. The graphs for training and validation accuracy increase are shown in Figure 30.

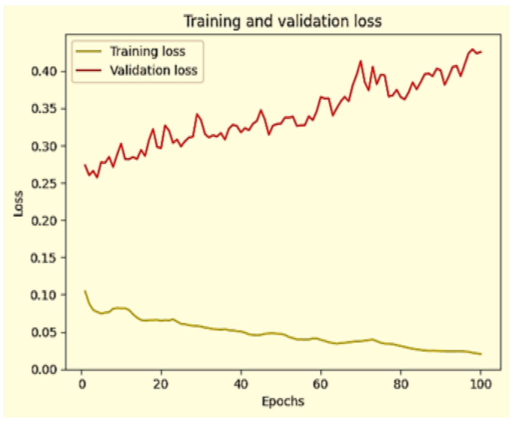


Figure 29. Training and validation loss after 100 epochs (weighted average ensemble technique)

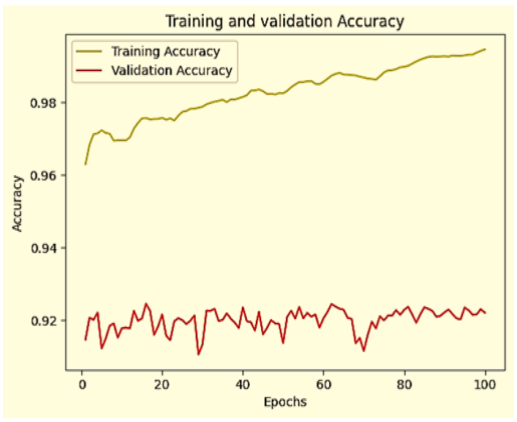
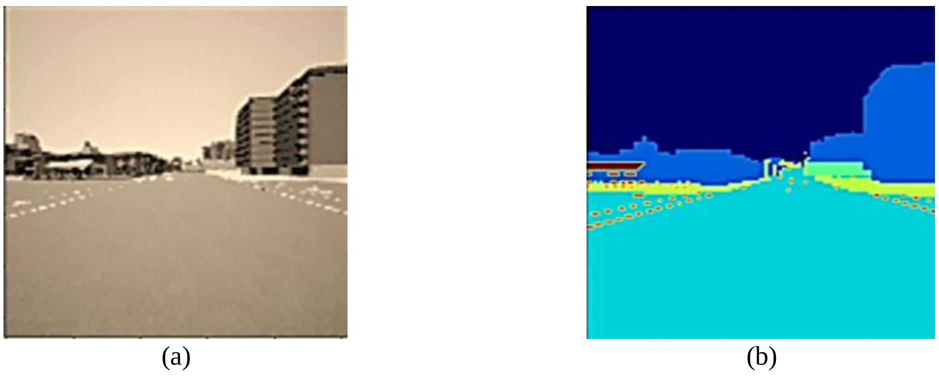


Figure 30. Training and validation accuracy after 100 epochs (weighted average ensemble technique)

Figure 31(a) shows the test image, Figure 31(b) shows the ground truth label, and Figure 31(c) shows the prediction on the test image after the model is trained for 100 epochs. The mean IoU observed was 0.5254637 after 100 epochs. The dice metric was 0.91319996, and the dice_coefficient_loss was 0.08680004. The comparison of results for the optimisation techniques for models with i) without batch normalisation, ii) with batch normalisation, iii) with reduced filter, iv) with average ensemble, and v) weighted average ensemble for the CARLA datasets after 100 epochs is shown in Table 2.


(a) (b)

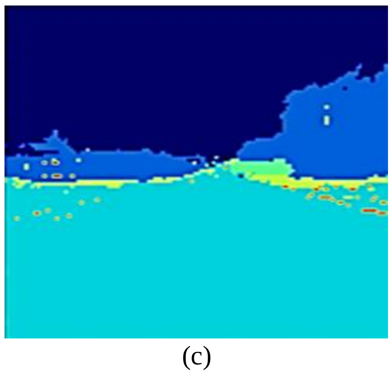

(c)

Figure 31. Comparison of (a) test image, (b) ground truth, and (c) predicted image for weighted average ensemble technique

Table 2. Summary of results

Sr. No	Parameters	Optimisation techniques				
		1	2	3	4	5
1	Training loss	0.1151	0.0807	0.1299	0.0339	0.0219
2	Train accuracy	0.9571	0.9689	0.9526	0.9844	0.9939
3	Valid loss	0.4084	0.3011	0.3923	0.4325	0.4315
4	Val. accuracy	0.9028	0.9168	0.8980	0.9204	0.9221
5	IoU Class 1	0.9624968	0.9610608	0.95677716	0.96463263	0.9635136
6	IoU Class 2	0.02	0.01	0.03	0.027586207	0.13846155
7	IoU Class 3	0.851734	0.8472761	0.8345403	0.85810524	0.85981905
8	IoU Class 4	0.9970636	0.9966345	0.9969176	0.99746764	0.99668646
9	IoU Class 5	0.7406022	0.7919159	0.7254831	0.7987827	0.8260491
10	Mean IoU	0.46047738	0.48857138	0.4433159	0.5070416	0.5254637
11	Dice metrics	0.8832081	0.9041302	0.88062567	0.9050604	0.91319996
12	DiceCoeff.Loss	0.1167919	0.0958697	0.11937433	0.09493959	0.08680004
13	Epoch time	3 s 222 ms/step	4 s 260 ms/step	1 s 64 ms/step	8 s 532 ms/step	8 s 545 ms/step

5. CONCLUSION

The study evaluates the different optimisation techniques for semantic segmentation intended for autonomous driving using the UNET Model. The validation accuracy was lowest for the model in Sr. No. 3 and highest for the model in Sr. No. 5. The computation time was lowest for the model in Sr. No. 3 and highest for the model in Sr. No. 5. There are minor variations in class IoU for the different optimisation techniques. The dice metrics is highest for the model in Sr. No. 5 and lowest for the model in Sr. No. 3. From the study, it can be concluded that the ensemble techniques help improve the model's accuracy at the cost of computational time. We hope these findings in semantic segmentation using UNET architecture will benefit other researchers in the autonomous driving ecosystem. Future works would consider additional experimentation and simulations to optimise other semantic segmentation algorithms in the autonomous driving system. A similar experiment, for instance segmentation for an autonomous driving application study, can be planned using different algorithms to extend this research. Also, other data sets can used for experimentation.

ACKNOWLEDGEMENTS

The authors would like to thank the staff of Symbiosis Institute of Technology Pune.

REFERENCES

[1] CARLA, "CARLA documentation," *CARLA Simulator*. Accessed: Jul. 04, 2021. [Online]. Available: https://carla.readthedocs.io/en/latest/#getting-started

[2] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: an open urban driving simulator," in *1st Annual Conference on Robot Learning*, pp. 1-16, 2017.

[3] F. Codevilla, M. Miiller, A. Lopez, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, Australia, pp. 4693–4700, 2018, doi: 10.1109/ICRA.2018.8460487.

[4] L. G. Cuenca, E. Puertas, J. F. Andrés, and N. Aliane, "Autonomous driving in roundabout maneuvers using reinforcement learning with q-learning," *Electronics*, vol. 8, no. 12, 2019, doi: 10.3390/electronics8121536.

[5] T. Buhet, E. Wirbel, and X. Perrotton, "Conditional vehicle trajectories prediction in carla urban environment," *2019 International Conference on Computer Vision Workshop, ICCVW 2019*, pp. 2310–2319, 2019, doi: 10.1109/ICCVW.2019.00284.

[6] CARLA, "CARLA: open-source simulator for autonomous driving research.," *CARLA*, 2022. [Online]. Available: https://carla.org//

[7] M. R. Bachute and J. M. Subhedar, "Autonomous driving architectures: insights of machine learning and deep learning algorithms," *Machine Learning with Applications*, vol. 6, 2021, doi: 10.1016/j.mlwa.2021.100164.

[8] X. Wang, Y. Qian, C. Wang, and M. Yang, "Map-enhanced ego-lane detection in the missing feature scenarios," *IEEE Access*, vol. 8, pp. 107958–107968, 2020, doi: 10.1109/ACCESS.2020.3000777.

[9] Y. Guo, Y. Liu, T. Georgiou, and M. S. Lew, "A review of semantic segmentation using deep neural networks," *International Journal of Multimedia Information Retrieval*, vol. 7, no. 2, pp. 87–93, 2018, doi: 10.1007/s13735-017-0141-z.

[10] W. Weng and X. Zhu, "UNet: convolutional networks for biomedical image segmentation," *IEEE Access*, vol. 9, pp. 16591–16603, 2021, doi: 10.1109/ACCESS.2021.3053408.

[11] O. Ronneberger, P. Fischer, and T. Brox, "Dental X-ray image segmentation using a U-shaped deep convolutional network," *International Symposium on Biomedical Imaging*, pp. 1–13, 2015.

[12] O Ronneberger, "U-Net: convolutional networks for biomedical image segmentation," *Vision: Pattern Recognition and Image Processing*, 2021. Accessed: Aug. 02, 2021. [Online]. Available: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

[13] A. Gurita and I. G. Mocanu, "Image segmentation using encoder-decoder with deformable convolutions," *Sensors*, vol. 21, no. 5, pp. 1–27, 2021, doi: 10.3390/s21051570.

[14] G. Men, G. He, and G. Wang, "Concatenated residual attention unet for semantic segmentation of urban green space," *Forests*, vol. 12, no. 11, 2021, doi: 10.3390/f12111441.

[15] J. L. Arrastia *et al.*, "Deeply supervised unet for semantic segmentation to assist dermatopathological assessment of basal cell carcinoma," *Journal of Imaging*, vol. 7, no. 4, 2021, doi: 10.3390/jimaging7040071.

[16] I. A. Kazerouni, G. Dooly, and D. Toal, "Ghost-UNet: an asymmetric encoder-decoder architecture for semantic segmentation from scratch," *IEEE Access*, vol. 9, pp. 97457–97465, 2021, doi: 10.1109/ACCESS.2021.3094925.

[17] Z. Wu, L. Zhao, and H. Zhang, "MR-UNet commodity semantic segmentation based on transfer learning," *IEEE Access*, vol. 9, pp. 159447–159456, 2021, doi: 10.1109/ACCESS.2021.3130578.

[18] Q. Yang, T. Ku, and K. Hu, "Efficient attention pyramid network for semantic segmentation," *IEEE Access*, vol. 9, pp. 18867–18875, 2021, doi: 10.1109/ACCESS.2021.3053316.

[19] M. H. Sheu, S. M. S. Morsalin, S. H. Wang, L. K. Wei, S. C. Hsia, and C. Y. Chang, "FHI-Unet: faster heterogeneous images semantic segmentation design and edge AI implementation for visible and thermal images processing," *IEEE Access*, vol. 10, pp. 18596–18607, 2022, doi: 10.1109/ACCESS.2022.3151375.

[20] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "UNet++: redesigning skip connections to exploit multiscale features in image segmentation," *IEEE Transactions on Medical Imaging*, vol. 39, no. 6, pp. 1856–1867, 2020, doi: 10.1109/TMI.2019.2959609.

[21] X. He, Y. Zhou, J. Zhao, D. Zhang, R. Yao, and Y. Xue, "Swin transformer embedding UNet for remote sensing image semantic segmentation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, 2022, doi: 10.1109/TGRS.2022.3144165.

[22] B. Baheti, S. Innani, S. Gajre, and S. Talbar, "Eff-UNet: A novel architecture for semantic segmentation in unstructured environment," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2020-June, pp. 1473–1481, 2020, doi: 10.1109/CVPRW50498.2020.00187.

[23] M. J. D. Melo *et al.*, "Automatic segmentation of cattle rib-eye area in ultrasound images using the UNet++ deep neural network," *Computers and Electronics in Agriculture*, vol. 195, 2022, doi: 10.1016/j.compag.2022.106818.

[24] I. S. Na, C. Tran, D. Nguyen, and S. Dinh, "Facial UV map completion for pose-invariant face recognition: a novel adversarial approach based on coupled attention residual UNets," *Human-centric Computing and Information Sciences*, vol. 10, no. 1, 2020, doi: 10.1186/s13673-020-00250-w.

[25] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," *30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017, pp. 6230–6239, 2017, doi: 10.1109/CVPR.2017.660.

[26] M. A. M. Elhassan *et al.*, "PPANet: point-wise pyramid attention network for semantic segmentation," *Wireless Communications and Mobile Computing*, vol. 2021, no. 1, 2021, doi: 10.1155/2021/5563875.

[27] Y. Kong, Y. Liu, B. Yan, H. Leung, and X. Peng, "A novel deeplabv3+ network for sar imagery semantic segmentation based on the potential energy loss function of gibbs distribution," *Remote Sensing*, vol. 13, no. 3, pp. 1–13, 2021, doi: 10.3390/rs13030454.

[28] H. Zeng, S. Peng, and D. Li, "Deeplabv3+ semantic segmentation model based on feature cross attention mechanism," *Journal of Physics: Conference Series*, vol. 1678, no. 1, 2020, doi: 10.1088/1742-6596/1678/1/012106.

[29] 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, doi: 10.1109/TPAMI.2016.2644615.

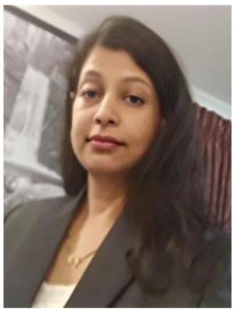
[30] S. Bhattiprolu, "Python for microscopists," *GitHub,* 2021. Accessed: Nov. 13, 2021. [Online]. Available: https://github.com/bnsreenu/python_for_microscopists/blob/master/219-unet_model_with_functions_of_blocks.py

[31] A. Abdollahi, B. Pradhan, and A. M. Alamri, "An ensemble architecture of deep convolutional Segnet and Unet networks for building semantic segmentation from high-resolution aerial images," *Geocarto International*, vol. 37, no. 12, pp. 3355–3370, 2022, doi: 10.1080/10106049.2020.1856199.

[32] M. Aslam, T. M. Khan, S. S. Naqvi, G. Holmes, and R. Naffa, "Ensemble convolutional neural networks with knowledge transfer for leather defect classification in industrial settings," *IEEE Access*, vol. 8, pp. 198600–198614, 2020, doi: 10.1109/ACCESS.2020.3034731.

BIOGRAPHIES OF AUTHORS

Javed Subhedar was born in India. He received his degree in B.E. [Electronics] from Walchand College of Engineering Sangli, Maharashtra State, India, and M.S. [Automotive Electronics] from Coventry University, and he is pursuing a Ph.D. from Symbiosis Institute of Technology, Pune Symbiosis International (Deemed University). He has rich experience in the automotive industry, and his current research area includes autonomous driving. He can be contacted at email: javed.subhedar.phd2020@sitpune.edu.in.



Mrinal R. Bachute is received Ph.D. (Electronics) and M.E. (Digital Electronics), currently an Associate Professor and Industry Liaison Officer at the Department of Electronics & Telecommunication Engineering in Symbiosis Institute of Technology, Pune Symbiosis International (Deemed University) Pune, Maharashtra, India. She has 20 years of teaching experience and has guided many postgraduate graduate students with their projects. Her areas of specialisation are digital image processing, machine learning, artificial intelligence, and adaptive signal processing. She has received research funding from the University of Pune and AICTE QIP grants. She has presented her work at many international workshops and conferences and also worked as session chair. Her research papers have been published in reputed journals and conferences at the national and international levels. She has delivered invited talks and expert sessions at various national and international levels, including at Langara University, Vancouver, Canada, organised by IET Canada, at ZE Power Engineering, Vancouver, Canada and IET, Trinidad, and Tabago. She has worked as a reviewer for conferences and reputed journals like Springer Nature and Elsevier. She can be contacted at email: mrinal.bachute@sitpune.edu.in.

Ketan Kotecha has Ph.D. and M.Tech. from (IIT Bombay) and is currently holding the positions of Head Symbiosis Centre for Applied A.I. (SCAAI), Director, Symbiosis Institute of Technology, CEO, Symbiosis Centre for Entrepreneurship and Innovation (SCEI), Dean, Faculty of Engineering, Symbiosis International (Deemed University). He has expertise and experience in cutting-edge research and projects in AI and deep learning for the last 25 + years. He has published 100+ widely in a number of excellent peer-reviewed journals on various topics ranging from cutting-edge AI, education policies, teaching-learning practices and AI for all. He is a recipient of the two SPARC projects worth INR 166 lacs from the MHRD govt of India in AI in collaboration with Arizona State University, USA and the University of Queensland Australia, and also the recipient of numerous prestigious awards like Erasmus+ faculty mobility grant to Poland, DUO-India professors fellowship for research in Responsible AI in collaboration with Brunel University, UK, LEAP grant at Cambridge University UK, UKIERI grant with Aston University UK, and a grant from Royal Academy of Engineering, the UK under Newton Bhabha Fund. He has published 3 patents and delivered keynote speeches at various national and international forums, including at Machine Intelligence Lab, USA, at IIT Bombay under the World Bank project, at the International Indian Science Festival organised by the Department of Science Technology, Govt of India and many more. He is also an academic editor of the Peerj Computer Science journal and Associate Editor of IEEE Access journal. He can be contacted at email: head@scaai.siu.edu.in.