IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 5, 2021, accepted August 19, 2021, date of publication August 24, 2021, date of current version September 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3107353

Estimation of Road Boundary for Intelligent Vehicles Based on DeepLabV3+ Architecture

SUNANDA DAS[1], AWAL AHMED FIME[1], NAZMUL SIDDIQUE[2], (Senior Member, IEEE), AND M. M. A. HASHEM[1]

[1]Department of Computer Science and Engineering, Khulna University of Engineering & Technology, Khulna 9203, Bangladesh
[2]School of Computing, Engineering and Intelligent Systems, Ulster University, Londonderry BT48 7JL, U.K.

Corresponding author: Sunanda Das (sunanda@cse.kuet.ac.bd)

This work was supported in part by Khulna University of Engineering and Technology (KUET), Bangladesh, and in part by Ulster University, U.K.

ABSTRACT Road boundary estimation is an essential task for autonomous vehicles and intelligent driving assistants. It is considerably straightforward to attain the task when roads are marked properly with indicators. However, estimating road boundary reliably without prior knowledge of the road, such as road markings, is extremely difficult. This paper proposes a method to estimate road boundaries in different environments with deep learning-based semantic segmentation, and without any predefined road markings. The proposed method employed an encoder-decoder-based DeepLab architecture for segmentation with different types of backbone networks such as VGG16, VGG19, ResNet-50, and ResNet-101 while handling the class imbalance problem by weighing the loss contribution of the model's different outputs. The performance of the proposed method is verified using the 'ICCV09DATA' dataset. The method outperformed other existing methods and achieved the accuracy, precision, recall, f-measure of 0.9596±0.0097, 0.9453±0.0118, 0.9369±0.0149, and 0.9408±0.0135 respectively while using RestNet-101 as a backbone network and Dice Coefficient as a loss function. The detailed experimental analysis confirms the feasibility of the proposed method for road boundary estimation in different challenging environments.

INDEX TERMS Augmentation, class imbalance, deep learning, DeepLabV3+ architecture, road boundary, semantic segmentation, transfer learning.

I. INTRODUCTION

Nowadays, autonomous vehicles [1] are getting popular worldwide, and are considered to be driver-less, effective, and crash avoiding perfect automobiles of the future. Autonomous vehicles must be able to see their surroundings to comprehend where they can and can't drive, recognize other vehicles on the street, and brake for pedestrians. It is anticipated that there will be around 20.8 million autonomous vehicles in operation alone in the United States by 2030 [2]. Autonomous vehicle in high-level automation (where vehicle drives on its own) requires ideal road conditions, which are not always possible. They are driven by AI and collect data through sensors. Sensors used in autonomous vehicles include LiDAR, Cameras, GPS, etc. enabling the vehicle to see and sense any adverse condition on the road. AI decides on steering, accelerating, and braking based on the data provided by various sensors so that the vehicle can drive safely, detect obstacles or other vehicles on the road, and deal with any unforeseen road conditions. Figure 1 provides a high-level view of the physical system of an autonomous vehicle where the visual road information from the front camera is fed into a neural network architecture for road boundary estimation.

Autonomous vehicles require the perception of the road for safe driving. The most crucial piece of information the autonomous vehicle needs is a complete understanding of the road it is driving on, including knowledge of the lane, lane/road boundary, lane markers, etc. so that it can perform multiple maneuvers such as lane switching, collision avoidance, overtaking, stopping, and so on. Very often road boundaries are broken, not clearly or perfectly defined, or sometimes not marked at all. Furthermore, various conditions like road bending, poor road marking, decay of painted indicators, blurred frames, background clutter, varying lighting conditions, occlusions, etc. make it challenging for the autonomous vehicle to detect the exact road boundaries. All of these factors have motivated us to develop a method that can accurately estimate road boundaries despite non-uniform road shapes, changing illumination

The associate editor coordinating the review of this manuscript and approving it for publication was Khin Wee Lai.

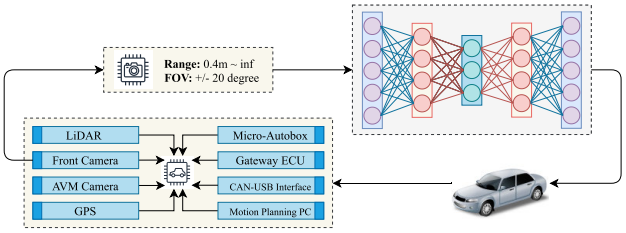


FIGURE 1. High-level view of the physical system.

conditions, and, most importantly, the absence of road markers.

Two main approaches have been used to solve this problem in the past decade. The first approach is the pre-defined feature extraction [3]–[5], where it uses lane markings and road boundaries. The disadvantage of the pre-defined features is that it may not work for all roads due to environmental differences, lighting conditions, and scale mismatch. The second is the model-based techniques [6], [7] where different models are employed to determine curves and straight lines from the edge features.

Our foremost goal is to design an intelligent method to detect road boundaries suitably in different environmental and weather conditions so that the vehicles are safe on roads and secured from accidents. To achieve this objective, we used deep learning-based image segmentation method to detect the drivable region. In digital image processing, image segmentation is an important task. In this work, we have employed semantic segmentation [8] to estimate the road boundary. The aim of semantic segmentation is to classify each pixel and label them according to a predefined class. It is used to classify different objects in the image. It can also be used for various disease and physical injury detection [9], [10] from medical images. Fully Convolutional Network (FCN) [11], ENet [12], U-Net [13], SegNet [14], DeepLab [15] are the most popular algorithms for semantic segmentation that don't need features in advance and can discover relevant features from the training data. In this work, we applied the DeepLabV3+ architecture for the segmentation.

Previous research such as Mana *et al.* [16] did not address the class imbalance problem which led to poor performance on precision, recall, and f-measure. Our proposed method not only handles the class imbalance problem but also considers the size of the dataset to develop a better model. The major contributions of this research are summarized as follows:

- Use of different types of augmentation techniques in the preprocessing phase of the method to deal with data shortage and overfitting problems.
- Employment of customized loss functions such as Dice Coefficient loss, Jaccard Index loss, and weighting the loss contribution of the model's various outputs based on the number of samples of different classes for solving the class imbalance problem.
- Employment of different backbone networks, that take the advantage of transfer learning to extract useful features from images, in the encoder and decoder part of DeepLabV3+ architecture to address the issue of small-sized dataset.
- Improvement of different evaluation parameters such as accuracy, precision, recall, and f-measure over other existing techniques to verify the robustness of the proposed method.

The rest of the paper is organized as follows: Section II describes the related works that have been conducted previously in this field. In Section III, the concept of the DeepLabv3+ architecture and the methodology of this work are demonstrated in detail. Section IV illustrates the experiments and presents the result analysis of the proposed method. The conclusion is drawn in Section V.

II. RELATED WORK

Road boundary estimation has drawn attention recently due to the rapid advancement of autonomous vehicles around the world. Light Detection and Ranging (LiDAR) is one of the most popular methods for road boundary detection. Several papers [17]–[19] used this approach to detect road boundaries.

Satti *et al.* [20] applied a machine learning approach for the detection and tracking of road boundaries. Initially, ConvNet merged with 3×3 Sobel filters are utilized to identify edges. The outputs of this process were fed to a line detection module where the Hough transform is used to recognize lines. Finally, the Kanade-Lucas-Tomasi method tracks the lines. Mana *et al.* [16] employed U-Net architecture for road boundary estimation. They used particle filters to handle the occasional boundary detection failures problem and generated a more consistent prediction. However, their model is only effective for the unbranched roads and strives to detect boundaries for other types of roads such as intersections and acutely curved roads.

Yadav *et al.* [21] proposed a novel approach based on CNN and the color lines model for unmarked road segmentation. The pre-trained SegNet model is used to discover the road texture effectively. In addition, they applied the conditional random field-based graphical model to deal with varying illumination situations. Chiu and Lin [22] suggested a color-based segmentation approach for lane detection in a complex environment. First, they determined the threshold from the region of interest, and then the threshold is used to recognize potential road boundaries. Their approach requires lower computation power and memory. Hernandez *et al.* [23] proposed a combination of color-based and texture line detector approach to detect boundaries for unstructured roads. In the color-based detector, a white balance algorithm is utilized to reduce the impact of illuminations while the texture line detector combines the Canny edge detector with Hough transform to identify parallel boundaries of the road. Their system used the Unscented Kalman Filter (UKF) which filters the noise and is able to recognize the boundaries when the roads are occluded. Chiku and Miura [24] suggested a

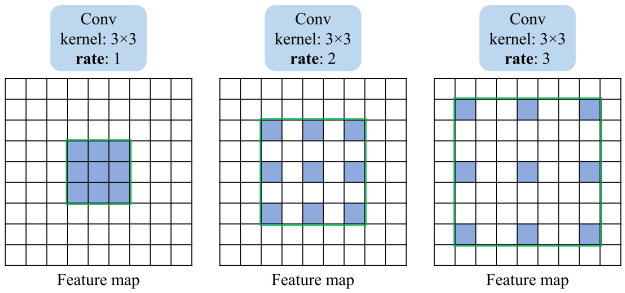


FIGURE 2. Illustration of atrous convolution with kernel size 3 × 3 for different dilation rates.

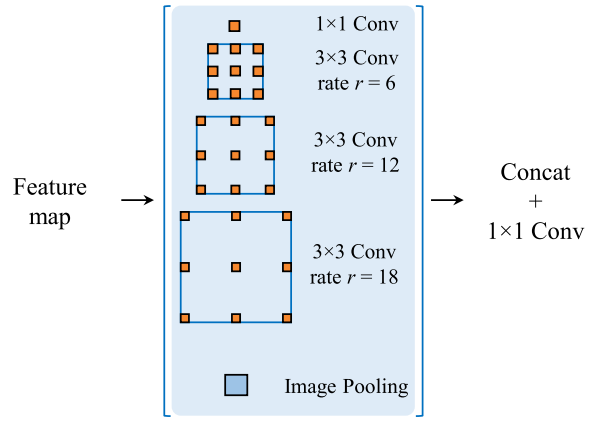


FIGURE 3. Illustration of Atrous Spatial Pyramid Pooling (ASPP). One 1 × 1 convolution, three 3 × 3 atrous convolutions with rate parameters 6, 12, 18, and image pooling are applied to generate the ASPP feature map.

multi-sensory road estimation framework for autonomous navigation which includes piecewise-linear road models, state transition, likelihood evaluation, resampling, and result estimation. Their system considered various shapes of the roads and can switch between different road models so that it can accurately calculate the boundaries.

In [25], Rakotondrajao and Jangsamsi assumed that at the time of image acquisition most of the 2D images contain perspective distortion which makes the lane detection quite difficult. Hence, they try to automatically recognize four points for executing Inverse Perspective Mapping (IPM) to minimize the distortion effects. As in numerous urban and rural areas, there is hardly any road markings, Kühnl and Fritsch [26] suggested a vision-based system for detecting road boundaries where road markings are not expected. Their system includes obtaining SPatial RAY (SPRAY) features, Boundary vicinity classification, Boundary extraction and finally applying partial linear regression to filter out the outliers. Nishida and Muneyasu [27] used both conventional and hyperbolic road models for the detection of road boundaries however, their proposed algorithm is computationally expensive.

For developing an occlusion-free road segmentation system, Yan *et al.* [28] proposed a LiDAR-based LMRoadNet network where the network is trained using a weighted loss function. Using a 1/4 scale feature map, they can perform road ground height prediction and road topology detection simultaneously with reduced complexity, and their fusion strategy can expand the field of view of the autonomous driving system. Perng *et al.* [29] demonstrated a system for detecting road boundaries, that takes into account a variety of road types (structured, unstructured) and conditions (daytime, nighttime, rainy day). A convolution autoencoder first removes unwanted objects from the image, leaving just lane markers to be fed to a hyperbolic model, and then the lanes are tracked by a particle filter. However, their system struggles to distinguish boundary features from the background especially at night time.

For off-road scene understanding, Gao *et al.* [30] proposed a contrastive learning technique that employs contrasting positive and negative samples in a self-supervised pipeline to gain discriminative representations. Their system can produce fine-grained semantic results that deliver rich information for robots traversing difficult off-road environments. Their system, on the other hand, sometimes fails to differentiate against unseen category samples, which might lead to erroneous predictions. Abdollahi *et al.* [31] applied MUNet which is a modified version of U-Net in the generative section of the Generative Adversarial Network (GAN) to generate high-resolution segmentation output. Before feeding the image to the network, Laplacian filtering is used as a pre-processing step. The accuracy of their proposed system is slightly lower compared to the other evaluation metrics such as precision, recall, f1-score, etc. For the estimate of road boundaries, Dewangan and Sahu [32] suggested an encoder-decoder-based system using U-Net, Seg-Net, and Fully Convolutional Network (FCN). They concluded that U-Net architecture with dice coefficient can predict better results compared to other models. A brief summary of the related work is presented in Table 1.

III. METHODS AND MATERIALS

A. DeepLab FOR IMAGE SEGMENTATION

DeepLabv3+ [41] is an encoder-decoder based network for semantic segmentation. This network is based on the concept of Atrous Convolution and Atrous Spatial Pyramid Pooling (ASPP).

1) ATROUS CONVOLUTION

In Atrous convolution, the active field of view of the convolution is managed by a rate parameter. The generalized form of atrous convolution can be presented as follows:

$$y[i] = \sum_{k} x[i + r \cdot k]w[k] \tag{1}$$

where w is the filter, i represents each location of output y, x is the input feature map and r represents the atrous rate which corresponds to the stride. In a regular convolution layer, the kernel size restricts the area on which the operation

TABLE 1. Brief summary of the related work.

Authors	Techniques	Dataset	Weakness/Remarks
Mano et al. [16]	U-Net + Particle filter	ICCV09DATA [33]	Only works for unbranched roads and failed to handle class imbalance problem.
Han et al. [17]	LiDAR + Integrated probabilistic data association filter (IPDAF)	Self-build	The estimation of roll and pitch angles needs to be improved.
Yalcin et al. [18]	LiDAR + Adaptive breakpoints detection algorithm	Self-build	Extraction of the breakpoints and estimation of roll and pitch need to be improved.
Homm et al. [19]	LiDAR + histogram-based approach	Self-build	Required to identify all relevant road boundaries in the occupancy grid.
Satti et al. [20]	CNN + Hough transform + Kanade-Lucas-Tomasi (KLT)	Self-build	The feature extraction process can be improved.
Yadav et al. [21]	pre trained SegNet + Color Lines Model	KITTI [34] & CamVid [35]	Did not address class imbalance and evaluation metrics can be improved.
Chiu and Lin [22]	Color-based segmentation	Self-build	When dealing with traffic signs drawn on the road and light problems caused by reflections, imperfect lane recognition can still be observed.
Hernandez et al. [23]	Canny edge detector with Hough transform + Unscented Kalman Filter (UKF)	Self-build	Did not address different shapes of roads such as intersections and acutely curved roads etc.
Chiku and Miura [24]	Piecewise linear Road models + particle filter	Self-build	Needs to switch between different road models to correctly detect road boundaries.
Rakotondrajao and Jangsamsi [25]	Inverse Perspective Mapping (IPM)	Self-build	Lane marking required.
Kühnl and Fritsch [26]	SPRAY Features + Boundary Vicinity Classification	KITTI-ROAD [36]	Doesn't require road markings, however the vicinity classifier can be improved to obtain a better recognition rate.
Nishida and Muneyasu [27]	Hyperbolic road model + Hough transform	Self-build	Computationally expensive.
Yan et al. [28]	LMRoadNet	Self-build Multi Road dataset based on SemanticKITTI [37]	Able to recognize the integrated road area even when it is occluded.
Perng et al. [29]	Convolutional auto-encoder + hyperbolic model + particle filter	SELab lane dataset & Caltech dataset [38]	Required lane markings, only applicable for single lane.
Gao et al. [30]	Contrastive Learning	Self-build off road dataset	The prediction process requires a considerable amount of computational cost. To improve computational efficiency, temporal and spatial consistency needs to be investigated.
Abdollahi et al. [31]	Generative Adversarial Network (GAN)	Massachusetts road dataset [39]	Cannot estimate continuous road parts and hardly identify roads in complex areas.
Dewangan and Sahu [32]	U-Net, Seg-Net, FCN-32	Camvid dataset [40]	The system needs to be investigated in challenging environments.

is to be performed. Hence, a larger area demands a large kernel. However, working with a large kernel is operationally expensive and time-consuming. This problem can be solved by using atrous convolution [42]. It decides the working area with the size of the kernel as well as the dilation factor. Standard convolution is a particular case of atrous convolution where dilation rate $r = 1$. Depending on the dilation rate, the kernel expands and then the rest of the positions are filled with zeros. As a result, an atrous convolution with kernel size 3×3 and dilation rate $r = 2$ works as a regular convolution with a kernel size of 5×5. A convolution with kernel size 3×3 and dilation rate $r = 3$ works as a convolution with kernel 7×7. Figure 2 illustrates this concept. In general, a $k\times k$ kernel with an atrous dilation rate r will produce a kernel $k_e = k + (k-1)(r-1)$ and introduces $r-1$ zeros within consecutive filter values. This process helps to secure denser features without the cost of learning any extra parameters.

2) ATROUS SPATIAL PYRAMID POOLING (ASPP)

To retrieve information on a different scale, several atrous convolution layers can be applied to the image. The ASPP can capture multi-scale data efficiently with several atrous rates. The ASPP includes (a) one convolution with kernel size $= 1\times1$ and three convolutions with kernel size $= 3\times3$ and dilation rates (6, 12, 18) respectively (b) the image-level features with image pooling. As shown in Fig. 3, the resulting features from the five branches are concatenated together and passed through a 1×1 convolution.

DeepLabv3 works as an encoder and extracts useful features at arbitrary resolution. Besides, the ASPP can explore the convolutional features at various scales with different dilation rates. Hence, the rich semantic information can be found from the output feature map of the encoder networks which generally contains 256 channels and is 32 times smaller compared to the resolution of the input image.

The rich encoded features from the encoder networks are upsampled bilinearly with a factor of 4 and then concatenated with the lower level features that come from a backbone network with the same shape. As the lower-level features include a substantial number of channels, they are reduced using 1×1 convolution before the concatenation so that they cannot outweigh the encoded features. After the concatenation,

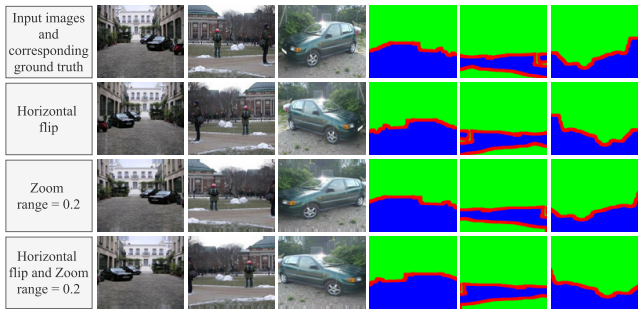


FIGURE 4. Different types of augmentation techniques applied in the method during training phase.

the features are passed through a few 3×3 convolutions and finally experienced another bilinear upsample by a factor of 4.

B. IMAGE AUGMENTATION AND SCALING

Semantic segmentation is one kind of classification. To perfectly classify each pixel of the test set, the model should learn to classify on a different scale. So we randomly applied 20% zoom on the training dataset. Horizontal flip is applied randomly to rearrange the position of the road in the image. Different types of augmentation techniques are illustrated in Fig. 4. All the images are resized to 256×256. Images in the dataset are in the RGB color model so, it contains values of the pixels between (0−255) in three different channels. We divide it by 255 to maintain the range between (0−1). Then the train set of the dataset is divided into two parts: training and validation set.

C. BackBone NETWORKS

It's common to train a ConvNet on a dataset that is considerably large, and then use the ConvNet in initialization or as a fixed feature extractor for the task at hand with the help of transfer learning.

Transfer learning [43] concentrates on storing experiences while solving one problem and applies that experience in resolving other related problems. In deep neural networks, initial layers extract general features. Hence, it is beneficial to apply the weights of a pre-trained model, trained on a larger dataset, for training other models with a small dataset to get better performance. Some popular datasets for pre-trained models are ImageNet [44], MNIST [45], CIFAR [46]. All of those datasets contain a large number of images. Backbone networks are used for feature extraction from images. Deep Convolutional Neural Network (DCNN) based backbone networks can extract high-level features and also downsample images from the input. Generally, a DCNN based backbone network includes Convolution, pooling, activation function, etc. We tried with 4 different state-of-the-art DCNN based backbone network architectures i.e. VGG16, VGG19, ResNet-50 and ResNet-101 for our method.

1) VGG

VGG16 and VGG19 architectures were introduced by Simonyan and Zisserman [47] for large scale image recognition back in 2014. VGG16 is based on VGG architecture with 16 layers. It consists of 5 convolution blocks and some fully-connected layers. The First 2 convolution blocks have 2 convolution layers each and the other 3 convolution blocks have 3 convolution layers each. The convolution operation uses a kernel of size 3×3. Those convolution layers automatically extract features from images. After each convolution layer, a rectified linear unit (ReLU) is used as an activation function. There is a max-pooling layer after each convolution block, so a total of 5 such layers. Each pooling operation is done by a kernel of size 2×2 with a stride of 2 and no padding. At the end, there are three fully-connected layers. First, two of them have 4096 channels each and the last layer has 1000 channels.

VGG19 is another variant of VGG architecture that contains 19 weight layers. It also has 5 convolution blocks and the same fully-connected layers. However, the last three convolution blocks contain an extra convolution layer each compared to VGG16. Details are presented in Table 2.

TABLE 2. Architecture of VGG16 and VGG19. The parameters of the convolutional layer are shown as "conv⟨receptive field size⟩-⟨number of channels⟩."

Layer Name	Output Shape	16 Weight Layers	19 Weight Layers
$block1\_convx$	224 × 224	conv3-64	conv3-64
		conv3-64	conv3-64
$block1\_pool$	112 × 112	maxpool	
$block2\_convx$	112 × 112	conv3-128	conv3-128
		conv3-128	conv3-128
$block2\_pool$	56 × 56	maxpool	
$block3\_convx$	56 × 56	conv3-256	conv3-256
		conv3-256	conv3-256
		conv3-256	conv3-256
		—	conv3-256
$block3\_pool$	28 × 28	maxpool	
$block4\_convx$	28 × 28	conv3-512	conv3-512
		conv3-512	conv3-512
		conv3-512	conv3-512
		—	conv3-512
$block4\_pool$	14 × 14	maxpool	
$block5\_convx$	14 × 14	conv3-512	conv3-512
		conv3-512	conv3-512
		conv3-512	conv3-512
		—	conv3-512
$block5\_pool$	7 × 7	maxpool	
		FC-4096	
		FC-4096	
		FC-1000	
		soft-max	

2) ResNet

The residual network [48] is quite appropriate for the training of deeper networks. The ResNet introduced the idea of 'identity shortcut connection' as shown in Fig. 5. Deep Neural Networks suffer from vanishing gradient problem as the gradient can easily shrink to zero. In ResNets, gradients can flow through skip connections and backpropagate to the earlier layers. A residual building block can be defined as

$$y = F(x, \{W_i\}) + x \tag{2}$$

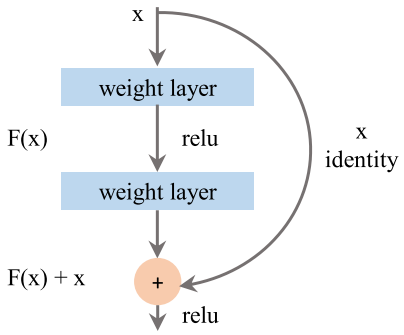


FIGURE 5. Building block of residual learning with skip connection.

TABLE 3. Architecture of ResNet-50 and ResNet-101. The building blocks for these architectures are shown in brackets, with the numbers of blocks stacked.

Layer Name	Output Shape	50-Layers	101-Layers
$conv1$	112×112	7×7, 64, stride 2	7×7, 64, stride 2
$conv2\_x$	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
$conv3\_x$	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$
$conv4\_x$	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$
$conv5\_x$	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	3×3 average pool, 1000-d fc	3×3 average pool, 1000-d fc

where the x, y represent the input and output vectors and $F(x, \{W_i\})$ depicts the residual mapping. For Fig. 5, $F = W_2\sigma(W_1x)$ where σ is the ReLU activation function. For changing dimensions, a linear projection W_s needs to be performed to adjust with the dimension.

$$y = F(x, \{W_i\}) + W_sx \tag{3}$$

Therefore, we can inject shortcuts to produce a residual version of a deep network. When the input and output have the same dimension, the identity shortcuts can be inserted using Eq. (2) and for matching the dimensions, a projection shortcut can be employed using Eq. (3). Residual networks with 50 layers are called ResNet-50. ResNet-101 also works with the same idea as ResNet-50. ResNet-101 has 101 layers so it can go deeper than ResNet-50 and extract more advanced features. Table 3 illustrates the detailed architecture for the ResNet-50 and ResNet-101.

In this work, we employed VGG16, VGG19, ResNet-50, and ResNet-101 as pre-trained models and used '*Imagenet*' weight in those models. First, a deep neural network-based model is pre-trained where some generalized features are extracted from different layers of that network. These features are fed to the encoder and decoder of DeepLab architecture based on their depth to enhance the performance of the method. Eventually, we fine-tuned our proposed model on the segmentation dataset while using augmentation techniques to reduce the overfitting.

D. CLASS IMBALANCE

Multi-class classification expects all classes to be equally distributed. When members of some class (minority class) are much lower than the members of other class or the number of elements of some class (majority class) is strictly higher than the members of other class or classes, then a class imbalance problem occurs. With this problem, the model is biased towards the majority class. Sometimes it may neglect some classes if the amount of training data is too low compared to the majority class. Semantic Segmentation also expects that pixels of all classes should be equal in quantity.

For solving the class imbalance problem, we applied different customized loss functions such as the Dice coefficient and the Jaccard index in our method and analyze their effect on the performance of the system. The Dice coefficient D and Jaccard Index J between two volumes p, q can be written as

$$D(p, q) = \frac{2\sum_i^N p_iq_i + \epsilon}{\sum_i^N p_i^2 + \sum_i^N q_i^2 + \epsilon} \tag{4}$$

$$J(p, q) = \frac{\sum_i^N p_iq_i + \epsilon}{\sum_i^N p_i^2 + \sum_i^N q_i^2 - \sum_i^N p_iq_i + \epsilon} \tag{5}$$

where ϵ is a small positive number which is added to avoid divide by zero error. To formulate the loss function, we employed $1 - D(p, q)$ and $\{1 - J(p, q)\} \times \epsilon$ in the method as Dice Coefficient Loss and Jaccard Index Loss respectively.

Secondly, we try to weigh the loss contributions of different model outputs using Eq. (6). The final loss used by the model is the weighted sum of all the individual losses.

$$w_i = \frac{samples_T}{class_T \times samples_i} \tag{6}$$

where w_i is the weight for i class, $samples_T$ represents the total number of samples in the dataset, $class_T$ is the total number of unique classes and $samples_i$ denotes the total number of samples in class i.

E. MODEL ARCHITECTURE

In this work, we applied an encoder-decoder based network which resembles the DeepLabV3+ [41] architecture for segmentation. According to Fig. 6, we extract two different layers L_e and L_d from the backbone network for the encoder and decoder respectively. The detailed information about different layers and their respective shapes are illustrated in Table 4. The method utilized VGG16, VGG19, ResNet-50, and ResNet-101 as backbone networks. The ASPP of the encoder applied different dilation rates $r = (6, 12, 18)$ for capturing multi-scale information. After concatenation,

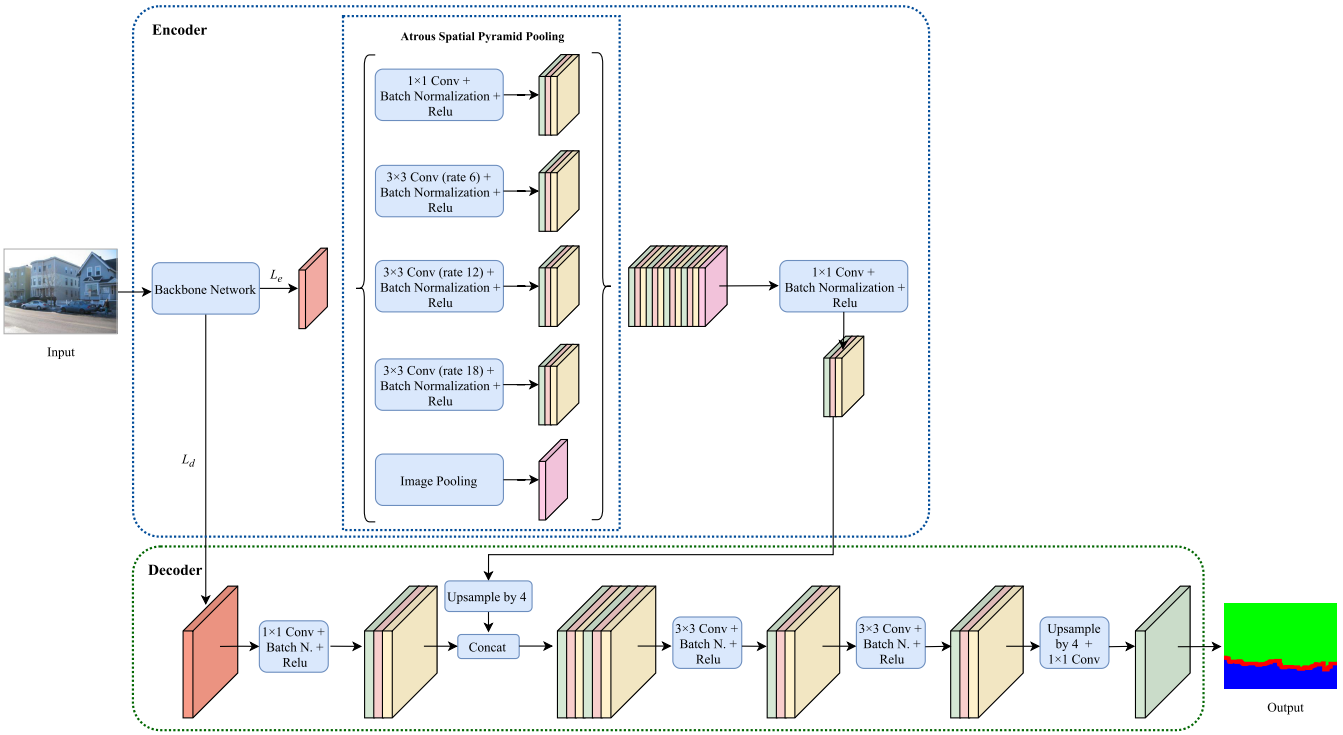


FIGURE 6. Architecture of DeepLabV3+ with backbone network.

TABLE 4. Detailed information about various layers extracted from different backbone networks and employed in the encoder and decoder of DeepLabV3+ architecture.

Backbone Networks	Layer Extracted For Encoder (L_e)	L_e Layer's Shape	Layer Extracted For Decoder (L_d)	L_d Layer's Shape	Pre-trained	Total Trainable Parameters
VGG16	'$block5\_conv3$'	$16 \times 16 \times 512$	'$block3\_conv3$'	$64 \times 64 \times 256$	'$Imagenet$'	20,150,947
VGG19	'$block5\_conv4$'	$16 \times 16 \times 512$	'$block3\_conv4$'	$64 \times 64 \times 256$	'$Imagenet$'	25,460,643
ResNet-50	'$conv4\_block6\_2\_relu$'	$16 \times 16 \times 256$	'$conv2\_block3\_2\_relu$'	$64 \times 64 \times 64$	'$Imagenet$'	11,819,875
ResNet-101	'$conv4\_block23\_2\_relu$'	$16 \times 16 \times 256$	'$conv2\_block3\_2\_relu$'	$64 \times 64 \times 64$	'$Imagenet$'	30,838,115

the features are fed through an 1×1 convolution and upsampled by a factor of 4 so that it can again be concatenated with the features of the decoder. In the last state of the decoder, the features are again upsampled by a factor of 4 and fed through an 1×1 convolution before giving the final output.

As semantic segmentation requires substantial spatial information, feature maps with the smaller resolution are not very effective for this kind of task. The atrous convolution of the encoder of our method can maintain spatial resolution by using different dilation rate at different blocks of the ASPP. Therefore, the size of the feature map is not reduced and the architecture provides better result at segmentation.

FIGURE 7. Detailed visualization of the ICCV09DATA [33] and the 3 class segmentation used in the experiments.

IV. EXPERIMENTS

A. DATASET

We used the 'ICCV09DATA' dataset [33] for evaluating our method. This dataset, also known as 'The Stanford Background Dataset' introduced in Gould *et al.* [33] for evaluating models for semantic scene understanding is very suitable for our research as we applied semantic segmentation-based deep learning to estimate the road boundary. It contains images with approximately 320-by-240 pixels. The pixels are divided into 8 different classes: sky, tree, road, grass, water, building, mountain, and foreground object. Figure 7. shows those

TABLE 5. Performance comparison of different methods with respect to various loss functions. Values of the metrics are based on the mean ± the margin of error at 95% confidence interval.

Methods	Input Shape	Loss Function	Loss Weight	Evaluation Metrics			
				Accuracy	*Precision*	*Recall*	*F-measure*
Mano et al. [16]	256×256×3	—	—	0.955±0.0047	0.609±0.0393	0.552±0.0399	0.571±0.0392
U-Net	256×256×3	Categorical Cross Entropy	'*no*'	0.9157±0.0108	0.3782±0.0721	0.3748±0.0772	0.3917±0.0756
SegNet	256×256×3	Categorical Cross Entropy	'*no*'	0.9131±0.0109	0.3572±0.0719	0.3716±0.0778	0.3453±0.0737
FCN8s	256×256×3	Categorical Cross Entropy	'*no*'	0.9055±0.0117	0.3735±0.0712	0.4098±0.0758	0.3955±0.0742
ENet	256×256×3	Categorical Cross Entropy	'*no*'	0.9087±0.0117	0.3937±0.0729	0.3778±0.0772	0.3302±0.0720
DeepLabV3+ (VGG16)	256×256×3	Categorical Cross Entropy	'*no*'	0.9528±0.0105	0.9371±0.0154	0.9219±0.0162	0.9294±0.0157
			'*yes*'	0.9545±0.0107	0.9397±0.0156	0.9242±0.0165	0.9319±0.0160
		Jaccard Index	'*no*'	0.9553±0.0098	0.9385±0.0144	0.9286±0.0149	0.9335±0.0146
			'*yes*'	0.9547±0.0095	0.9376±0.0139	0.9277±0.0144	0.9326±0.0142
		Dice Coefficient	'*no*'	0.9542±0.0107	0.9384±0.0156	0.9251±0.0165	0.9317±0.0160
			'*yes*'	0.9543±0.0100	0.9377±0.0147	0.9262±0.0152	0.9319±0.0149
DeepLabV3+ (VGG19)	256×256×3	Categorical Cross Entropy	'*no*'	0.9523±0.0107	0.9366±0.0156	0.9208±0.0166	0.9286±0.0161
			'*yes*'	0.9548±0.0110	0.9388±0.0162	0.9265±0.0168	0.9326±0.0165
		Jaccard Index	'*no*'	0.9530±0.0112	0.9350±0.0165	0.9251±0.0169	0.9300±0.0167
			'*yes*'	0.9550±0.0106	0.9381±0.0156	0.9283±0.0161	0.9332±0.0158
		Dice Coefficient	'*no*'	0.9551±0.0097	0.9400±0.0140	0.9259±0.0151	0.9329±0.0145
			'*yes*'	0.9552±0.0100	0.9393±0.0146	0.9271±0.0155	0.9333±0.0150
DeepLabV3+ (ResNet-50)	256×256×3	Categorical Cross Entropy	'*no*'	0.9574±0.0099	0.9438±0.0145	0.9292±0.0153	0.9364±0.0149
			'*yes*'	0.9575±0.0099	0.9448±0.0143	0.9280±0.0155	0.9363±0.0149
		Jaccard Index	'*no*'	0.9563±0.0095	0.9399±0.0140	0.9300±0.0145	0.9349±0.0142
			'*yes*'	0.9580±0.0085	0.9425±0.0123	0.9326±0.0130	0.9375±0.0126
		Dice Coefficient	'*no*'	0.9578±0.0085	0.9434±0.0123	0.9306±0.0133	0.9370±0.0128
			'*yes*'	0.9581±0.0088	0.9432±0.0126	0.9319±0.0136	0.9375±0.0131
DeepLabV3+ (ResNet-101)	256×256×3	Categorical Cross Entropy	'*no*'	0.9581±0.0103	0.9448±0.0149	0.9289±0.0161	0.9372±0.0155
			'*yes*'	0.9592±0.0103	0.9451±0.0149	0.9316±0.0158	0.9391±0.0154
		Jaccard Index	'*no*'	0.9576±0.0107	0.9419±0.0157	0.9320±0.0162	0.9369±0.0159
			'*yes*'	0.9579±0.0100	0.9423±0.0146	0.9324±0.0152	0.9373±0.0149
		Dice Coefficient	'*no*'	0.9582±0.0086	0.9446±0.0124	0.9307±0.0134	0.9375±0.0129
			'*yes*'	**0.9596±0.0097**	**0.9453±0.0118**	**0.9369±0.0149**	**0.9408±0.0135**

TABLE 6. Performance comparison of different backbone networks in the DeepLabV3+ architecture over Dice Score, Dice Score Loss, Jaccard Index, and Jaccard Index Loss.

Methods	Dice Score		Jaccard Index		Dice Score Loss		Jaccard Index Loss	
	unweighted	*weighted*	*unweighted*	*weighted*	*unweighted*	*weighted*	*unweighted*	*weighted*
DeepLabV3+ (VGG16)	0.9547±0.0107	0.9551±0.0099	0.9556±0.0098	0.9550±0.0094	0.0452±0.0107	0.0449±0.0099	0.0444±0.0098	0.0450±0.0094
DeepLabV3+ (VGG19)	0.9554±0.0097	0.9558±0.0100	0.9533±0.0111	0.9553±0.0106	0.0445±0.0097	0.0442±0.0100	0.0467±0.0111	0.0446±0.0106
DeepLabV3+ (ResNet-50)	0.9584±0.0085	0.9589±0.0087	0.9565±0.0095	0.9583±0.0084	0.0416±0.0085	0.0411±0.0087	0.0435±0.0095	0.0417±0.0084
DeepLabV3+ (ResNet-101)	**0.9586±0.0086**	**0.9601±0.0097**	**0.9579±0.0106**	**0.9584±0.0099**	**0.0414±0.0086**	**0.0399±0.0097**	**0.0421±0.0106**	**0.0416±0.0099**

classes [8 colors with their corresponding class]. We keep the road portion of the images and make others as background. Then we create a border between road and background which creates 3 classes: background, road, and road boundary. This dataset contains 715 images. We kept 100 images randomly as the test set and used the other 615 images for training purposes.

We have also tested 14 images containing different shapes of roads. These images are collected from 'The Geograph® Britain and Ireland Project' [49].

TABLE 7. McNemar's test on different methods with respect to DeepLabV3+(ResNet-101, pre-trained on ***imagenet***, Dice Coefficient with Weight Loss).

Methods	Loss Functions	Loss Weight	McNemar's Test	
			'p-value' with no continuity correction	*'p-value' with continuity correction*
U-Net	Categorical Cross Entropy	'no'	$5.123513\times e^{-79}$	$5.887469\times e^{-79}$
SegNet	Categorical Cross Entropy	'no'	$5.448216\times e^{-105}$	$6.402622\times e^{-105}$
FCN8s	Categorical Cross Entropy	'no'	$1.935747\times e^{-51}$	$2.808303\times e^{-51}$
ENet	Categorical Cross Entropy	'no'	$4.053112\times e^{-87}$	$5.820169\times e^{-87}$
DeepLabV3+ (VGG16)	Categorical Cross Entropy	'no'	$1.174144\times e^{-210}$	$2.260219\times e^{-210}$
		'yes'	$9.466543\times e^{-230}$	$1.807002\times e^{-229}$
	Jaccard Index	'no'	$1.148780\times e^{-228}$	$2.112613\times e^{-228}$
		'yes'	$2.827925\times e^{-226}$	$5.634394\times e^{-226}$
	Dice Coefficient	'no'	$3.437014\times e^{-133}$	$6.056248\times e^{-133}$
		'yes'	$8.584517\times e^{-76}$	$1.262509\times e^{-75}$
DeepLabV3+ (VGG19)	Categorical Cross Entropy	'no'	$6.305704\times e^{-72}$	$9.092791\times e^{-72}$
		'yes'	$2.495180\times e^{-130}$	$4.342783\times e^{-130}$
	Jaccard Index	'no'	$5.352554\times e^{-182}$	$9.696584\times e^{-182}$
		'yes'	$2.179004\times e^{-80}$	$3.319981\times e^{-80}$
	Dice Coefficient	'no'	$3.188415\times e^{-145}$	$5.625502\times e^{-145}$
		'yes'	$2.693779\times e^{-38}$	$3.643865\times e^{-38}$
DeepLabV3+ (ResNet-50)	Categorical Cross Entropy	'no'	$1.765894\times e^{-16}$	$2.239808\times e^{-16}$
		'yes'	$1.481467\times e^{-94}$	$2.393157\times e^{-94}$
	Jaccard Index	'no'	$1.997798\times e^{-131}$	$3.442275\times e^{-131}$
		'yes'	$4.876897\times e^{-29}$	$6.350494\times e^{-29}$
	Dice Coefficient	'no'	$2.243259\times e^{-92}$	$3.825357\times e^{-92}$
		'yes'	$7.292309\times e^{-253}$	$1.502996\times e^{-252}$
DeepLabV3+ (ResNet-101)	Categorical Cross Entropy	'no'	$7.786661\times e^{-73}$	$1.229495\times e^{-72}$
		'yes'	$1.717040\times e^{-65}$	$2.465138\times e^{-65}$
	Jaccard Index	'no'	$9.159958\times e^{-299}$	$1.946905\times e^{-298}$
		'yes'	$1.02527\times e^{-129}$	$1.781583\times e^{-129}$
	Dice Coefficient	'no'	$1.133959\times e^{-196}$	$2.209380\times e^{-196}$

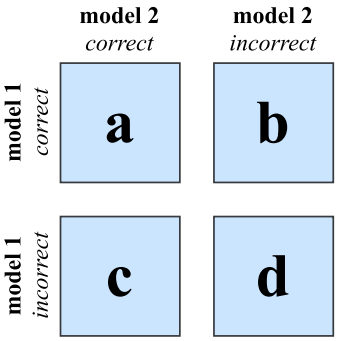


FIGURE 8. Illustration of contingency table for McNemar's test.

B. EXPERIMENTAL SETTING

The method is implemented on a Windows 10 PC which has a RAM of 16GB, an Intel Core i7 processor, a CPU speed of 3.60 GHz, and an Nvidia GeForce GTX 1050Ti graphical processing unit (GPU) (768 Nvidia Cuda cores). We used Python 3.6 and 'Keras' API to build the proposed model.

C. EVALUATION METRICS

Accuracy, precision, recall, and f-measure are utilized for the quantitative evaluation of the segmentation task. The metrics are computed according to Eqs. (7)−(10) where T_p, T_n, F_p, F_n are the number of true positives, true negatives, false positives, and false negatives at the pixel level, respectively.

$$Accuracy = \frac{(T_p + T_n)}{(T_p + F_p + T_n + F_n)} \tag{7}$$

$$Precision = \frac{T_p}{(T_p + F_p)} \tag{8}$$

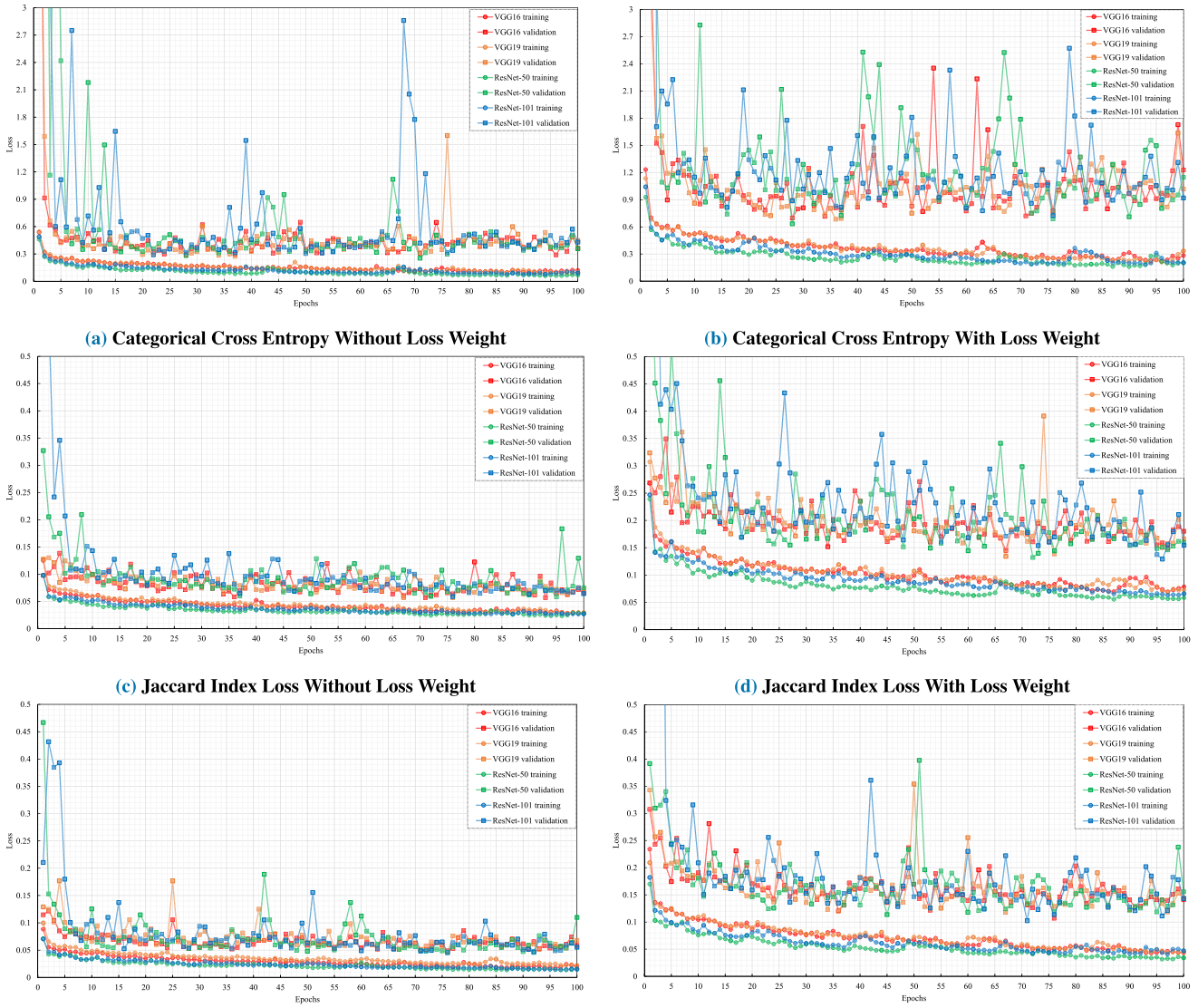
$$Recall = \frac{T_p}{(T_p + F_n)} \tag{9}$$

$$F\text{-}measure = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} \tag{10}$$

McNemar's Test or within-subjects chi-squared test is a non-parametric statistical test that signifies the predictability of models. The McNemar's test uses a contingency table that tabulates the outcomes of two different models as described in Fig. 8. The McNemar test statistic (sometimes known as the ''chi-squared'') is calculated as follows:

$$\chi^2 = \frac{(b-c)^2}{(b+c)} \tag{11}$$

When the total of the b and c cells is considerably large, χ^2 follows a one-degree-of-freedom chi-squared distribution, and we can estimate the *p-value* assuming that the null hypothesis is true after setting a significant threshold, such as $\alpha = 0.05$. Another often used variety of the McNemar



(a) Categorical Cross Entropy Without Loss Weight

(b) Categorical Cross Entropy With Loss Weight

(c) Jaccard Index Loss Without Loss Weight

(d) Jaccard Index Loss With Loss Weight

(e) Dice Coefficient Loss Without Loss Weight

(f) Dice Coefficient Loss With Loss Weight

FIGURE 9. Loss versus Epochs graphs during training and validation phases with respect to different loss functions.

statistic, the continuity corrected version, can be calculated as follows:

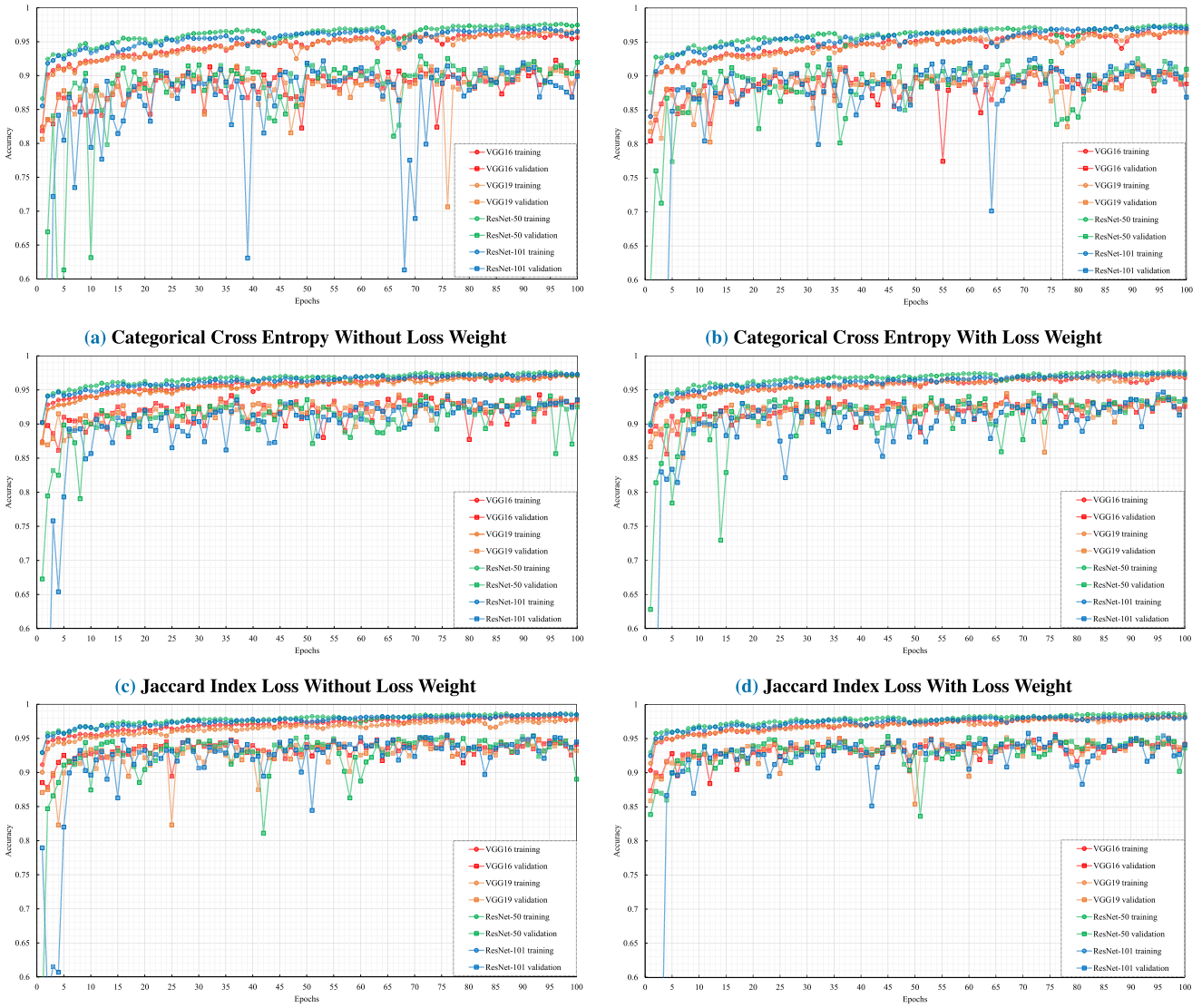
$$\chi^2 = \frac{(|b-c|-1)^2}{(b+c)} \tag{12}$$

D. PROCEDURE

In the proposed method, we applied different types of pre-trained models such as VGG16, VGG19, ResNet-50, and ResNet-101 with the 'imagenet' weight for the initial feature extraction. We have taken the output from two different layers (L_e, L_d) with two different depths from each of those backbone networks for the encoder and decoder respectively. For the encoder, we applied '*block*5_*conv*3', '*block*5_*conv*4', '*conv*4_*block*6_2_*relu*', '*conv*4_*block*23_2_*relu*' as L_e and for decoder, '*block*3_*conv*3', '*block*3_*conv*4', '*conv*2_*block*3_2_*relu*', '*conv*2_*block*3_2_*relu*' layers are extracted as L_d from VGG16, VGG19, ResNet-50 and ResNet-101 respectively. The details of these layers can be found in Table 4.

The last column of Table 4 represents the total number of trainable parameters for each of the backbone networks. In the training phase, these parameters are adjusted by the optimizer after backpropagation was employed for gradient computation which means the ResNet-101 with a total trainable parameter of 30,838,115 will need a substantial amount of computation during the training process.

In the model, we have employed two types of activation functions. ReLU [50] activation function for internal layers and Softmax [51] activation function in final layer for the multi-class classification. For upsampling, we utilized Bi-linear Interpolation to estimate the necessary values. We used Dice Coefficient Loss, Jaccard Index Loss and



(a) Categorical Cross Entropy Without Loss Weight

(b) Categorical Cross Entropy With Loss Weight

(c) Jaccard Index Loss Without Loss Weight

(d) Jaccard Index Loss With Loss Weight

(e) Dice Coefficient Loss Without Loss Weight

(f) Dice Coefficient Loss With Loss Weight

FIGURE 10. Accuracy versus Epochs graphs during training and validation phases with respect to different loss functions.

Categorical Cross-Entropy separately as loss functions and the 'adam' optimizer while compiling the model. As most of the pixels of our dataset are in the background, it creates a class imbalance problem. That's why we applied the Dice Coefficient, Jaccard Index as loss functions to deal with the problem and weighted the model's loss contribution according to Eq. (6).

E. RESULT ANALYSIS

The trained models are tested on the 100 images which are different from the training dataset. We also tested the proposed model on some of the other images from 'The Geograph® Britain and Ireland Project' [49] to verify the robustness of the model.

To understand the improvement of the proposed method with ResNet-101 and dice coefficient loss over other methods, we compared the performance of different methods in Table 5 in terms of accuracy, precision, recall, and f-measure. All of the evaluation metrics are presented with a margin of error at a 95% confidence interval. Although Mana *et al.* [16] achieved a notable accuracy of 0.955±0.0047, it performs inadequately as it only managed to produce 0.609±0.0393, 0.552±0.0399, and 0.571±0.0392 for precision, recall, and f-measure respectively. This is also true for other popular segmentation methods such as U-Net, SegNet, FCN8s, and ENet where we haven't applied any pre-trained backbone during the training process. We applied four different backbone networks (VGG16, VGG19, ResNet-50, ResNet-101), three different loss functions (Categorical Cross Entropy, Jaccard Index Loss, Dice Coefficient Loss) and combine them with and without 'loss weight' so that we

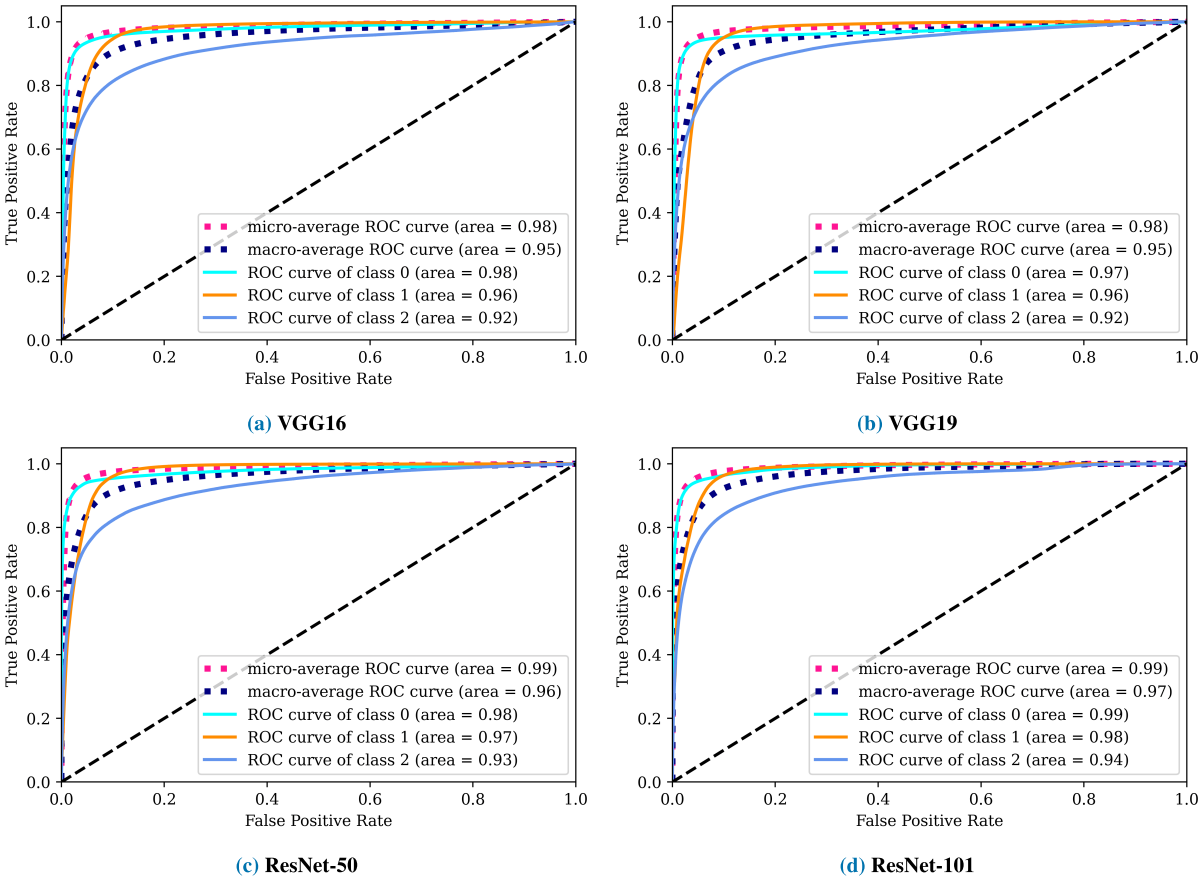


FIGURE 11. Comparison of different backbone networks used in DeepLabV3+ architecture with respect to multi-class ROC AUC on ICCV09DATA.

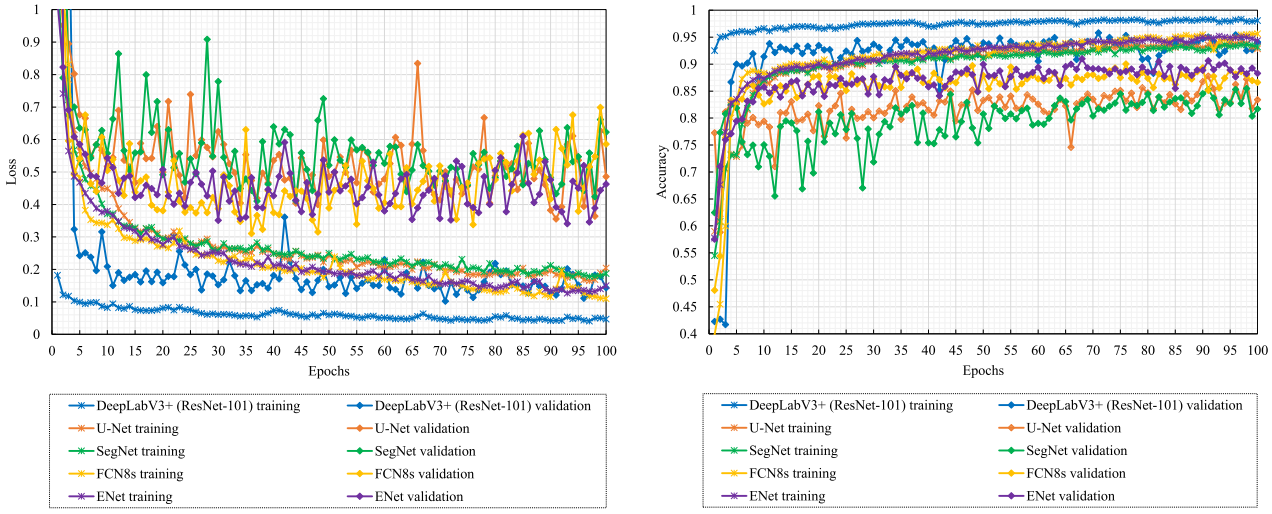
can analyze their effect on the method. In most cases, the weighted version performs better than the unweighted version of the loss functions. Every version of the models of the proposed method provides satisfactory results with respect to the four evaluation factors i.e. accuracy, precision, recall, and f-measure. Our method provides the best result for ResNet-101 while using the Dice Coefficient Loss with 'loss weight' and achieves accuracy, precision, recall, and f-measure of 0.9596±0.0097, 0.9453±0.0118, 0.9369±0.0149, and 0.9408±0.0135 respectively which is very promising compared to Mana *et al.* [16].

In Table 6, we attempted to evaluate the impact of Dice Score, Jaccard Index, Dice Score Loss, and Jaccard Index Loss for the test set. We applied Dice Score and Jaccard Index in the model to quantify the performance of the segmentation method. Again, the weighted version produces better outcomes than the unweighted version i.e. $Dice\_Score_{unweighted} < Dice\_Score_{weighted}$, $Jaccard\_Index_{unweighted} < Jaccard\_Index_{weighted}$, $DS\_Loss_{unweighted} > DS\_Loss_{weighted}$, and likewise $JI\_Loss_{unweighted} > JI\_Loss_{weighted}$.

Specifically, the ResNet-101 outperforms other models as it achieves a Dice score of 0.9586±0.0086, 0.9601±0.0097, a Jaccard Index of 0.9579±0.0106, 0.9584±0.0099, a Dice Score Loss of 0.0414±0.0086, 0.0399±0.0097, and a Jaccard Index Loss of 0.0421±0.0106, 0.0416±0.0099 for the unweighted and weighted version respectively.

A non-parametric statistical test such as McNemar's test has been carried out on different methods to demonstrate the statistical significance of the proposed ResNet-101 model. The details of the experiment are presented in Table 7. Table 7 shows that the *p-value* of the test is considerably less than the threshold i.e. 0.05 in both cases, regardless of whether the continuity correction is used or not which indicates that we can reject the null hypothesis for both cases and conclude that the performance of the proposed model is significantly different from the other methods.

For determining the best loss function for our method, the effects of different loss functions on the models are analyzed in Fig 9. Every loss function is expressed with and without 'loss weight'. For categorical cross-entropy, the loss is presented on the scale of $(0 \sim 3)$ and for other loss functions, a scale of $(0 \sim 0.5)$ is used. Every subfigure exhibits the loss vs epochs curve for the training and validation phase and for each of the backbone networks. According to Fig. 9a, and Fig. 9b, in categorical cross-entropy, for both versions,



(a) Loss versus Epochs

(b) Accuracy versus Epochs

FIGURE 12. Loss and accuracy plots of the proposed DeepLabV3+(ResNet-101, pre-trained on *imagenet*) with other popular segmentation networks (without pre-trained backbones).

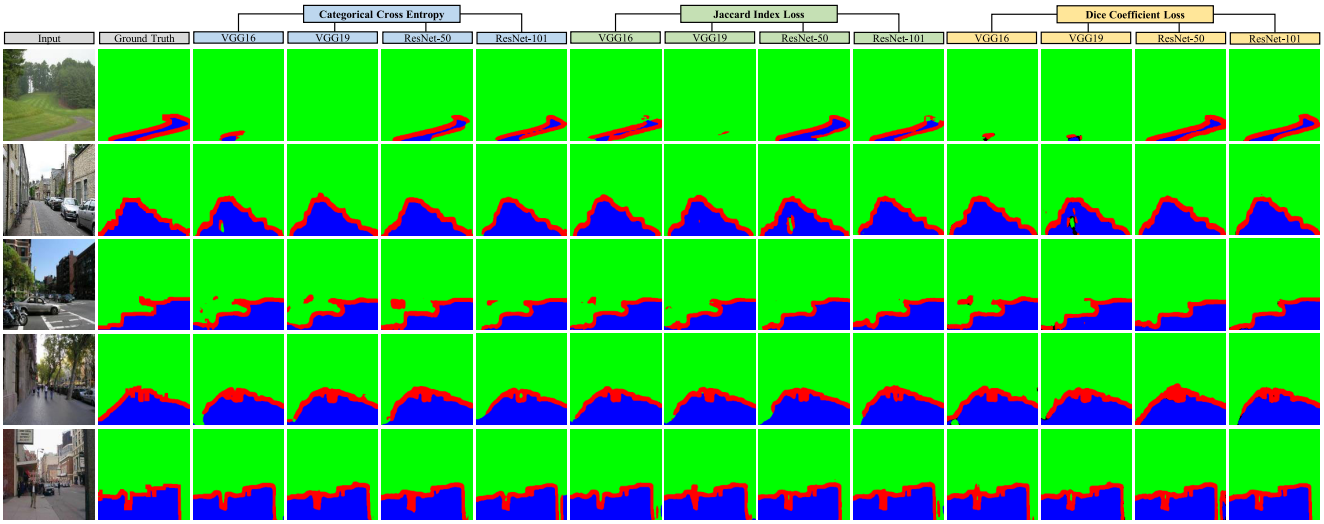


FIGURE 13. Segmentation outputs from the DeepLabV3+ architecture based on different backbone networks with various loss functions.

the training loss decreases after each epoch and the validation loss remain approximately constant after decreasing at a certain point. For other loss functions i.e. Jaccard Index Loss and Dice Coefficient Loss, both the training and validation loss decrease after each epoch. While categorical cross-entropy is maintaining the approximately same loss for the validation set, Jaccard Index and Dice Coefficient Loss are assisting the models to learn significantly on the training and validation data by decreasing the train and validation loss in each epoch. Moreover, the loss generated by models with Dice Coefficient Loss for the training and validation phase is much lower compared to models with other loss functions. Therefore, from the graphical representation of loss functions, it is evident that the models with dice coefficient loss perform better than the models compiled with other loss functions. Similarly, we inspected the accuracy vs epochs curves for every model, as shown in Fig.10 which provides sufficient evidence that the proposed DeepLabV3+ architecture with ResNet-101 model compiled with Dice Coefficient loss with Loss weight is free of overfitting. Table 8 shows the average time of each epoch during the training process for different methods, where averages are calculated over 100 epochs. The DeepLabV3+ with ResNet-101 and dice coefficient loss takes relatively a little bit more time for a single epoch than other methods.

In Fig. 11, we present the ROC plot for the different backbone networks with Dice Coefficient Loss, where class 0, class1, and class 2 indicate road, background, and road

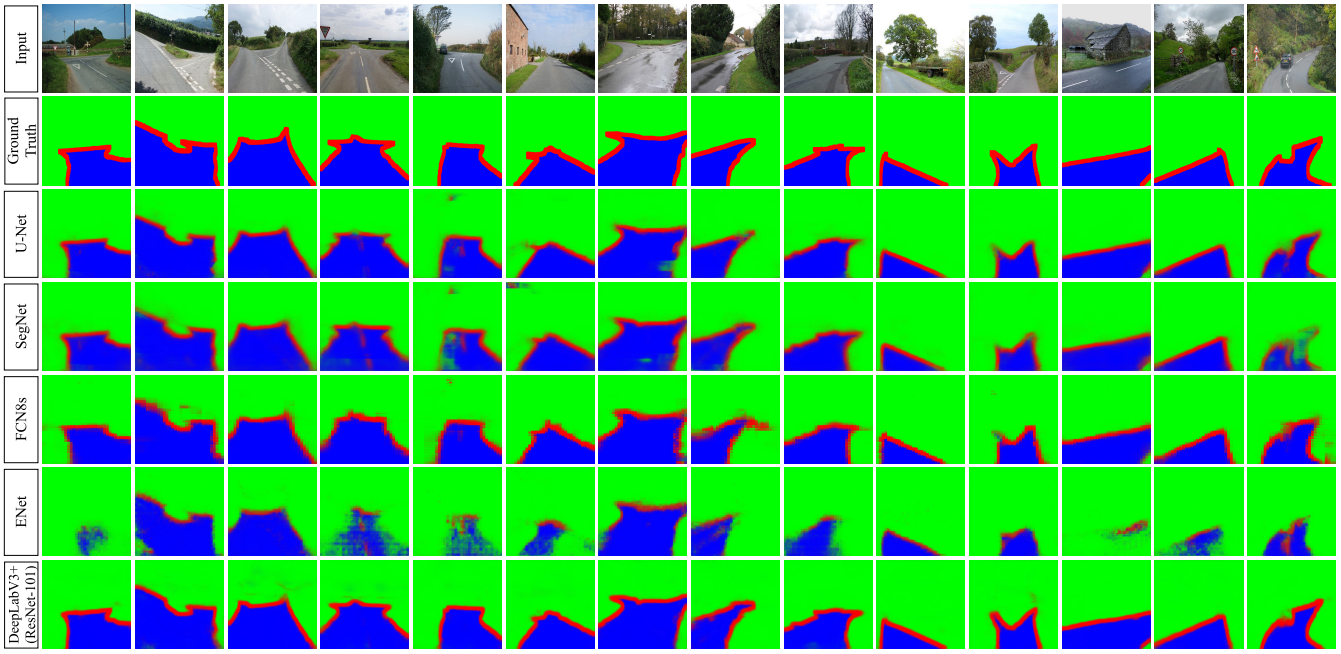


FIGURE 14. Segmentation result comparison between the proposed DeepLabV3+(ResNet-101, pre-trained on ***imagenet***) with other popular segmentation networks (without pre-trained backbones). These input images are collected from 'The Geograph® Britain and Ireland Project' [49].

TABLE 8. Average time per epoch for different methods during training phase along with different loss functions such as Categorical Cross Entropy (C_Cross), Jaccard Index (Jaccard), and Dice Coefficient (Dice).

Methods	Input Shape	Loss Function	Loss Weight	Time (Seconds)
U-Net	256×256×3	C_Cross	'*no*'	25.3714
SegNet	256×256×3	C_Cross	'*no*'	27.6504
FCN8s	256×256×3	C_Cross	'*no*'	31.0459
ENet	256×256×3	C_Cross	'*no*'	28.7024
DeepLabV3+ (VGG16)	256×256×3	C_Cross	'*no*'	30.5015
			'*yes*'	30.0419
		Jaccard	'*no*'	29.5967
			'*yes*'	30.6730
		Dice	'*no*'	31.5774
			'*yes*'	29.7571
DeepLabV3+ (VGG19)	256×256×3	C_Cross	'*no*'	29.3566
			'*yes*'	29.0868
		Jaccard	'*no*'	30.7171
			'*yes*'	31.8643
		Dice	'*no*'	30.7830
			'*yes*'	30.2579
DeepLabV3+ (ResNet-50)	256×256×3	C_Cross	'*no*'	30.2095
			'*yes*'	28.3215
		Jaccard	'*no*'	28.4802
			'*yes*'	30.4335
		Dice	'*no*'	29.2432
			'*yes*'	28.5952
DeepLabV3+ (ResNet-101)	256×256×3	C_Cross	'*no*'	30.9885
			'*yes*'	31.4643
		Jaccard	'*no*'	31.4099
			'*yes*'	31.2117
		Dice	'*no*'	33.4448
			'*yes*'	34.0404

boundary respectively. The higher the area, the better the model is at predicting different classes. Also, a model with a higher area presents that the model has high-class separation capability. According to the plot, ResNet-101 performs more reliably as it maintains a significant area under curve for every class. Fig.12 illustrates the loss and accuracy graphs of the proposed DeepLabV3+(ResNet-101, pre-trained on *imagenet*) with respect to other popular segmentation networks (without any pre-trained backbones). The proposed method has a lower training and validation loss than other methods, as demonstrated in Fig.12. Similarly, the training and validation accuracy is higher than the respective training and validation accuracy of other methods. Furthermore, for both subgraphs, the proposed method converges in less number of epochs compared to other methods (without any pre-trained backbone networks).

To estimate the best model with respect to the loss function, the segmentation outputs of different models for various loss functions are illustrated in Fig. 13. The first and second column present different shapes of roads and their corresponding ground truth values. Later, we offer the results of each of the input images for different backbone networks with three different loss functions. After carefully analyzing the segmentation results, it is apparent that the output generated by ResNet-101 with dice coefficient loss approximately resembles the corresponding ground truth values.

Figure 14 compares the segmentation results of the proposed DeepLabV3+(ResNet-101, pre-trained on *imagenet*) with other popular segmentation methods where no pre-trained network is applied during the training process. To verify the robustness of different methods, Fig. 14 uses different shapes of roads as inputs, and all the models are trained for 100 epochs. Although the results generated by U-Net, SegNet, FCN8s are almost good, these models clearly

need much more number of epochs during training to reach convergence and the segmentation results of the ENet are deeply inadequate compared to other methods. On the other hand, the proposed method accurately identifies the road, road boundaries, and background for each of the images. These results clearly show that the proposed method is quite effective in estimating the boundaries of various challenging shaped roads.

V. CONCLUSION

The revolution introduced by autonomous vehicles will transform the traditional transportation system for good. Therefore, accurate detection of the road boundary can help to develop intelligent vehicles for future road transportation.

In this paper, we proposed a method with DeepLabV3+ Architecture and by adopting different pre-trained deep neural network based models for road boundary estimation. One of the advantages of the work compared to traditional feature-based methods is that our method does not require any existing road markings for the estimation of road boundaries. The atrous convolution of DeepLab preserves the spatial resolution of feature maps which is quite beneficial for segmentation. The robustness of the model is further accelerated by transfer learning as it allows our method to use pre-trained networks, trained on the Imagenet dataset, for the segmentation task. Besides, as the number of background pixels is considerably high compared to road and road boundary, we applied customized loss functions i.e. Dice Coefficient loss and Jaccard Index loss as they consider overlap while estimating loss function and weighted the loss contribution of the model's different output. Analyzing the outputs of the method, we can conclude that the proposed method with ResNet-101 and Dice Coefficient Loss is quite adequate for challenging environments such as the different shapes of roads and various brightness situations as the method outperformed other networks regarding the accuracy, precision, recall, and f-measure.

For uniform and straight roads, our method is capable of determining the road boundary. Although the overall boundary estimation for crossroads and roads with bending is comparatively good, it requires further improvement. Very often potholes and damaged roads are the cause of road accidents. In the future, we also plan to develop models that can detect potholes and road damages along with road boundaries so that undesired accidents and mishaps can be prevented.

REFERENCES

[1] M. V. Rajasekhar and A. K. Jaswal, "Autonomous vehicles: The future of automobiles," in *Proc. IEEE Int. Transp. Electrific. Conf. (ITEC)*, Aug. 2015, pp. 1–6.

[2] *Projected Number of Autonomous Vehicles in Operation in the United States in 2025 and 2030*. Accessed: Jan. 6, 2021. [Online]. Available: https://www.statista.com/statistics/750149/us-autonomous-vehicles-in-operation-forecast/

[3] L.-W. Tsai, J.-W. Hsieh, C.-H. Chuang, and K.-C. Fan, "Lane detection using directional random walks," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2008, pp. 303–306.

[4] Y. He, H. Wang, and B. Zhang, "Color-based road detection in urban traffic scenes," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 4, pp. 309–318, Dec. 2004.

[5] Y. U. Yim and S.-Y. Oh, "Three-feature based automatic lane detection algorithm (TFALDA) for autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 4, no. 4, pp. 219–225, Dec. 2003.

[6] Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using B-snake," *Image Vis. Comput.*, vol. 22, no. 4, pp. 269–280, Apr. 2004.

[7] W. Yue, D. Shen, and E. K. Teoh, "Lane detection using spline model," *Pattern Recognit. Lett.*, vol. 21, no. 8, pp. 677–689, 2000.

[8] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A review on deep learning techniques applied to semantic segmentation," 2017, *arXiv:1704.06857*. [Online]. Available: https://arxiv.org/abs/1704.06857

[9] M.-H. Laves, J. Bicker, L. A. Kahrs, and T. Ortmaier, "A dataset of laryngeal endoscopic images with comparative study on convolution neural network-based semantic segmentation," *Int. J. Comput. Assist. Radiol. Surgery*, vol. 14, no. 3, pp. 483–492, Mar. 2019.

[10] L. Ding, K. Zhao, X. Zhang, X. Wang, and J. Zhang, "A lightweight U-Net architecture multi-scale convolutional network for pediatric hand bone segmentation in X-ray image," *IEEE Access*, vol. 7, pp. 68436–68445, 2019.

[11] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3431–3440.

[12] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A deep neural network architecture for real-time semantic segmentation," 2016, *arXiv:1606.02147*. [Online]. Available: https://arxiv.org/abs/1606.02147

[13] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput.-Assisted Intervent*. Cham, Switzerland: Springer, 2015, pp. 234–241.

[14] V. Badrinarayanan, A. Handa, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling," 2015, *arXiv:1505.07293*. [Online]. Available: https://arxiv.org/abs/1505.07293

[15] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2017.

[16] K. Mana, H. Masuzawa, J. Miura, and I. Ardiyanto, "Road boundary estimation for mobile robot using deep learning and particle filter," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2018, pp. 1545–1550.

[17] J. Han, D. Kim, M. Lee, and M. Sunwoo, "Enhanced road boundary and obstacle detection using a downward-looking LIDAR sensor," *IEEE Trans. Veh. Technol.*, vol. 61, no. 3, pp. 971–985, Mar. 2012.

[18] O. Yalcin, A. Sayar, O. F. Arar, S. Apinar, and S. Kosunalp, "Detection of road boundaries and obstacles using LIDAR," in *Proc. 6th Comput. Sci. Electron. Eng. Conf. (CEEC)*, Sep. 2014, pp. 6–10.

[19] F. Homm, N. Kaempchen, J. Ota, and D. Burschka, "Efficient occupancy grid computation on the GPU with lidar and radar for road boundary detection," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2010, pp. 1006–1013.

[20] S. K. Satti, K. S. Devi, P. Dhar, and P. Srinivasan, "A machine learning approach for detecting and tracking road boundary lanes," *ICT Exp.*, vol. 7, no. 1, pp. 99–103, Mar. 2021.

[21] S. Yadav, S. Patra, C. Arora, and S. Banerjee, "Deep CNN with color lines model for unmarked road segmentation," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 585–589.

[22] K.-Y. Chiu and S.-F. Lin, "Lane detection using color-based segmentation," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2005, pp. 706–711.

[23] D. E. Hernandez, S. Blumenthal, E. Prassler, S. Bo, and Z. Haojie, "Vision-based road boundary tracking system for unstructured roads," in *Proc. IEEE Int. Conf. Unmanned Syst. (ICUS)*, Oct. 2017, pp. 66–71.

[24] T. Chiku and J. Miura, "On-line road boundary estimation by switching multiple road models using visual features from a stereo camera," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 4939–4944.

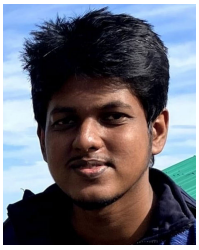
[25] F. Rakotondrajao and K. Jangsamsi, "Road boundary detection for straight lane lines using automatic inverse perspective mapping," in *Proc. Int. Symp. Intell. Signal Process. Commun. Syst. (ISPACS)*, Dec. 2019, pp. 1–2.

[26] T. Kühnl and J. Fritsch, "Visio-spatial road boundary detection for unmarked urban and rural roads," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2014, pp. 1251–1256.

[27] M. Nishida and M. Muneyasu, "Detection of road boundaries using hyperbolic road model," in *Proc. Int. Symp. Intell. Signal Process. Commun. Syst.*, Nov. 2012, pp. 521–526.

[28] F. Yan, K. Wang, B. Zou, L. Tang, W. Li, and C. Lv, "LiDAR-based multi-task road perception network for autonomous vehicles," *IEEE Access*, vol. 8, pp. 86753–86764, 2020.
[29] J. W. Perng, Y. W. Hsu, Y. Z. Yang, C. Y. Chen, and T. K. Yin, "Development of an embedded road boundary detection system based on deep learning," *Image Vis. Comput.*, vol. 100, Aug. 2020, Art. no. 103935.
[30] B. Gao, S. Hu, X. Zhao, and H. Zhao, "Fine-grained off-road semantic segmentation and mapping via contrastive learning," 2021, *arXiv:2103.03651*. [Online]. Available: https://arxiv.org/abs/2103.03651
[31] A. Abdollahi, B. Pradhan, G. Sharma, K. N. A. Maulud, and A. Alamri, "Improving road semantic segmentation using generative adversarial network," *IEEE Access*, vol. 9, pp. 64381–64392, 2021.
[32] D. K. Dewangan and S. P. Sahu, "Road detection using semantic segmentation-based convolutional neural network for intelligent vehicle system," in *Data Engineering and Communication Technology*. Singapore: Springer, 2021, pp. 629–637.
[33] S. Gould, R. Fulton, and D. Koller, "Decomposing a scene into geometric and semantically consistent regions," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep. 2009, pp. 1–8.
[34] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.
[35] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and recognition using structure from motion point clouds," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2008, pp. 44–57.
[36] J. Fritsch, T. Kuehnl, and A. Geiger, "A new performance measure and evaluation benchmark for road detection algorithms," in *Proc. 16th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2013, pp. 1693–1700.
[37] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "Semantickitti: A dataset for semantic scene understanding of LiDAR sequences," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2019, pp. 9297–9307.
[38] M. Aly, "Real time detection of lane markers in urban streets," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2008, pp. 7–12.
[39] V. Mnih, *Machine Learning for Aerial Image Labeling*. Toronto, ON, Canada: Univ. Toronto, 2013.
[40] J. Fauqueur, G. Brostow, and R. Cipolla, "Assisted video object labeling by joint tracking of regions and keypoints," in *Proc. IEEE 11th Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–7.
[41] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 801–818.
[42] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017, *arXiv:1706.05587*. [Online]. Available: https://arxiv.org/abs/1706.05587
[43] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. Hershey, PA, USA: IGI Global, 2010, pp. 242–264.
[44] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[45] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.
[46] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on CIFAR-10," *Unpublished Manuscript*, vol. 40, no. 7, pp. 1–9, 2010.
[47] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556
[48] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.
[49] *The Geograph Britain and Ireland Project*. Accessed: Jul. 8, 2021. [Online]. Available: https://www.geograph.org.uk/
[50] Y. Li and Y. Yuan, "Convergence analysis of two-layer neural networks with relu activation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 597–607.
[51] R. A. Dunne and N. A. Campbell, "On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function," in *Proc. 8th Aust. Conf. Neural Netw.*, Melbourne, VIC, Australia, vol. 181, Jun. 1997, p. 185.

SUNANDA DAS received the B.Sc. degree (engineering) in computer science and engineering (CSE) from Khulna University of Engineering & Technology (KUET), Khulna, Bangladesh, in 2018. He is currently working as a full-time Lecturer at the CSE Department, KUET. His research interests include machine learning, deep learning, artificial intelligence, natural language processing, and computer vision. He has published a couple of articles on these topics and is actively working on various research projects with his students, colleagues, and collaborators.



AWAL AHMED FIME was born in Jashore, Bangladesh. He is currently pursuing the B.Sc. degree in computer science and engineering (CSE) with Khulna University of Engineering & Technology (KUET), Khulna, Bangladesh. His research interests include computer vision, artificial intelligence, signal processing, machine learning, and deep learning.

NAZMUL SIDDIQUE (Senior Member, IEEE) received the Dipl.-Ing. degree in cybernetics from TU Dresden, Germany, the M.Sc. degree in computer science from BUET, Bangladesh, and the Ph.D. degree in intelligent control from the Department of Automatic Control and Systems Engineering, The University of Sheffield, U.K. He is currently with the School of Computing, Engineering and Intelligent Systems, Ulster University. He has published more than 170 research articles in the broad area of computational intelligence, vehicular communication, robotics, and cybernetics. He authored or coauthored five books published by John Wiley, Springer, and Taylor & Francis. His research interests include robotics, cybernetics, computational intelligence, nature-inspired computing, stochastic systems, and vehicular communication. He is a fellow of the Higher Education Academy and a member of different committees of IEEE SMCS. He guest edited eight special issues of reputed journals on *Cybernetic*, *Intelligence*, *Computational Intelligence*, *Neural Networks*, and *Robotics*. He has been involved in organizing many national and international conferences and co-edited seven conference proceedings. He is on the Editorial Board of the *Scientific Research* (Nature), *Journal of Behavioral Robotics*, *Engineering Letters*, *International Journal of Machine Learning and Cybernetics*, *International Journal of Applied Pattern Recognition*, and *Advances in Robotics Research*. He is also on the Editorial Advisory Board of the *International Journal of Neural Systems*.

M. M. A. HASHEM received the bachelor's degree in electrical and electronic engineering from Khulna University of Engineering & Technology (KUET), Khulna, Bangladesh, in 1988, the master's degree in computer science from the Asian Institute of Technology (AIT), Bangkok, Thailand, in 1993, and the Ph.D. degree in artificial intelligence systems from Saga University, Japan, in 1999. He is currently a Professor with the Department of Computer Science and Engineering, KUET. He has published more than 100 refereed articles in international journals/conferences. He is a coauthor of a book titled *Evolutionary Computations: New Algorithms and their Applications to Evolutionary Robots* and Series: *Studies in Fuzziness and Soft Computing* (Berlin, NY, USA: Springer-Verlag, 2004; Vol. 147, ISBN: 3-540-20901-8). His research interests include artificial intelligence, machine learning, intelligent transportation systems, bioinformatics, biomedical engineering, health informatics, biomedical instrumentation, evolutionary computations, and soft computing.