Volume 6, Issue 1, 2022
Eigenpub Review of Science and Technology
peer-reviewed journal dedicated to showcasing cutting-edge research and innovation in the fields of science and technology.
https://studies.eigenpub.com/index.php/erst

Comparative Evaluation of VGG-16 and U-Net Architectures for Road Segmentation

Mahmoud Abouelyazid
Exodia AI Labs

ABSTRACT

Accurate road segmentation is a fundamental task in autonomous driving and intelligent transportation systems. This study aims to compare the performance of two well-known deep learning architectures, VGG-16 and U-Net, for road segmentation on the KITTI dataset. The VGG-16 model is adapted for segmentation by using its convolutional layers as an encoder and incorporating a custom decoder for upsampling and generating the final segmentation mask. The U-Net model follows an encoder-decoder structure with skip connections to preserve spatial information and capture both high-level and low-level features. Both models are trained using a combination of binary cross-entropy and Dice coefficient losses, along with data augmentation techniques to improve robustness and generalization. Extensive experiments are conducted to evaluate the models' performance using a comprehensive set of metrics, including overall accuracy, mean Intersection over Union (mIoU), road IoU, precision, recall, and F1 score. The VGG-16 model achieves an overall accuracy of 0.95, mIoU of 0.85, and road IoU of 0.92, demonstrating its effectiveness in capturing road segments. The U-Net model slightly outperforms VGG-16, with an overall accuracy of 0.96, mIoU of 0.88, and road IoU of 0.94. Additionally, the U-Net model exhibits faster inference times, lower GPU memory usage, and a more compact model size compared to VGG-16. The results demonstrate the strong capabilities of both VGG-16 and U-Net architectures for road segmentation, with U-Net showing a slight edge in terms of performance and efficiency. This study contributes to the understanding of deep learning-based road segmentation and provides valuable insights for the development of reliable and efficient autonomous driving systems. Future research can explore techniques such as attention mechanisms, multi-scale feature fusion, or domain adaptation to further enhance the segmentation performance and generalization ability of the models.

Keywords: *Road segmentation, deep learning, VGG-16, U-Net, KITTI dataset, autonomous driving, comparative analysis, performance evaluation*

I. INTRODUCTION

Autonomous vehicles rely on how well they are able to detect and learn their environment. Road segmentation is a key component of this perception, supporting autonomous vehicles in identifying and distinguishing road surfaces from other objects in their proximity [1,2]. This enables vehicles to determine drivable areas and plan safe trajectories for navigation. Computer vision algorithms and machine learning techniques are utilized to achieve accurate road segmentation, ensuring the vehicle can operate safely in various conditions. Proper lane keeping and navigation are essential functions that autonomous vehicles must perform, and road segmentation plays a big role in enabling these capabilities. Accurate road segmentation allows vehicles to detect lane markings and boundaries, ensuring they stay within their designated lanes, even on challenging roads or in adverse weather conditions. This is for maintaining order and safety on the road, as well as complying with traffic regulations. Continuously monitoring and adjusting the vehicle's position within the lane is necessary to provide a smooth and safe driving experience for passengers [3,4].

Table 1. Road Segmentation in Autonomous Driving and Transportation System

Category	**Significance**
Autonomous Vehicles	
Perception and Understanding of the Environment	Helps identify and distinguish road surfaces from other objects
	Determines drivable areas and plans safe trajectories
Lane Keeping and Navigation	Detects lane markings and boundaries accurately
	Facilitates proper lane keeping and navigation
Obstacle Avoidance	Identifies obstacles on the road (pedestrians, vehicles, debris)
	Enables timely decisions to avoid collisions and ensure passenger safety
Advanced Driver Assistance Systems (ADAS)	
Lane Departure Warning (LDW)	Enables detection of lane markings
	Alerts drivers when drifting out of lane unintentionally
Adaptive Cruise Control (ACC)	Identifies preceding vehicles and their distances
	Maintains safe following distance and adjusts speed accordingly
Automatic Emergency Braking (AEB)	Contributes to detecting potential collision risks
	Applies brakes automatically to prevent or mitigate collisions
Road Maintenance and Monitoring	
Pavement Condition Assessment	Analyzes pavement conditions and detects defects
	Helps prioritize maintenance tasks and optimize resource allocation
Road Marking and Sign Inventory	Identifies and catalogs road markings and signs
	Facilitates creation and updating of comprehensive road databases
Traffic Flow Analysis	Analyzes vehicle density and movement patterns from traffic camera footage
	Provides insights for traffic management, congestion mitigation, and infrastructure planning
Safety and Efficiency	
Reduction in Human Error	Enables consistent and reliable decisions
	Minimizes accidents caused by human errors
Improved Traffic Flow	Facilitates smoother navigation and reduces congestion
	Optimizes traffic flow by maintaining proper spacing and speeds
Increased Accessibility	Contributes to the development of reliable autonomous transportation services
	Enhances mobility options for individuals unable to drive, such as the elderly or people with disabilities

Autonomous vehicles must identify and react to various obstacles on the road, such as pedestrians, other vehicles, and debris. Road segmentation helps distinguish the road from potential obstacles, enabling autonomous vehicles to make split-second decisions to avoid collisions and ensure the safety of passengers and other road users [5]. High-precision sensors, including cameras, lidar, and radar, work in conjunction with sophisticated perception algorithms to accurately classify and track objects in real-time. This allows autonomous vehicles to navigate complex urban environments and handle unexpected situations with a high degree of reliability, reducing the risk of accidents and enhancing overall road safety Lane Departure Warning (LDW) is one such system that relies on accurate road segmentation to detect lane markings. This enables the LDW system to alert drivers when the vehicle is unintentionally drifting out of its lane, helping to prevent accidents caused by distracted or drowsy driving. The system uses cameras and image processing algorithms to continuously monitor the vehicle's position within the lane and provide timely warnings when necessary [6,7].

Adaptive Cruise Control (ACC) systems use sensors, such as cameras and radar, to identify preceding vehicles and calculate their distances. Road segmentation helps distinguish between vehicles and other objects on the road, allowing the ACC system to accurately track the vehicle ahead. This enables the system to maintain a safe following distance and automatically adjust the vehicle's speed accordingly, reducing the risk of rear-end collisions and promoting a more relaxed driving experience, especially during long highway trips.

Road segmentation is useful in detecting potential collision risks, such as stopped vehicles or obstacles in the vehicle's path. The AEB system can quickly determine if a collision is imminent by continuously analyzing the road ahead and identifying potential hazards. In such cases, the system can automatically apply the brakes to reduce the severity of the impact or avoid the collision altogether. The effectiveness of AEB systems relies heavily on accurate road segmentation, as it helps distinguish between genuine obstacles and false positives, ensuring the system intervenes only when necessary to maintain the trust and confidence of the driver. Road segmentation techniques can be applied to assess pavement conditions accurately, detecting cracks, potholes, or other surface defects. This information helps road maintenance authorities prioritize repair tasks and allocate resources effectively, ensuring that the most critical issues are addressed promptly. Regular pavement condition assessment using road segmentation can help extend the lifespan of road networks and improve overall driving conditions for motorists [8,9].

Segmentation techniques can automatically identify and catalog various types of road markings, such as lane dividers, crosswalks, and stop lines, as well as traffic signs like speed limits, yield signs, and direction indicators. This data can be used to update road databases, which are for navigation systems, road maintenance planning, and ensuring that road signage complies with safety regulations [10,11]. Accurate and up-to-date road marking and sign inventories contribute to improved road safety and a better driving experience for all road users.

Applying segmentation techniques to traffic camera footage enables the detection and tracking of individual vehicles, allowing for the analysis of vehicle density and movement patterns. This information can be used to identify congested areas, optimize traffic signal timings, and plan infrastructure improvements. Traffic flow analysis using road segmentation can also help in predicting and mitigating traffic jams, rerouting vehicles to less congested paths, and improving overall traffic efficiency.

Road segmentation technology increases safety and efficiency on our roads. Autonomous vehicles and Advanced Driver Assistance Systems (ADAS) rely on accurate road segmentation to make consistent and reliable decisions, minimizing the risk of accidents caused by human errors. Distracted or impaired driving, which are leading causes of traffic accidents, can be significantly reduced as these systems continuously monitor the road and respond to potential hazards in a timely manner. The integration of road segmentation in autonomous vehicles and ADAS has the potential to revolutionize road safety, saving countless lives and reducing the overall cost of traffic accidents.

Accurate road segmentation also contributes to improved traffic flow and reduced congestion. Autonomous vehicles with road segmentation can navigate roads more efficiently, maintaining proper spacing and speeds. This optimizes traffic flow, as vehicles can communicate with each other and adjust their movements accordingly, reducing the likelihood of sudden braking or acceleration that often leads to traffic jams. Smoother traffic flow not only reduces travel times but also has environmental benefits, as it minimizes fuel consumption and emissions caused by stop-and-go traffic. The widespread adoption of road segmentation technology in autonomous vehicles has the potential to transform our cities, making them more livable and sustainable.

Researchers have developed many image processing techniques for this purpose, see [12]. Road detection is a segmentation problem where the task is to segment the road area on a perceived image. Classical image segmentation techniques have been widely used for road detection. However, for autonomous driving, image processing and segmentation techniques on camera images may not be sufficient due to inappropriate conditions such as insufficient lines, signs, and poor weather and light conditions. RGB cameras, being passive sensors, highly depend on ambient light. Distortions like shadows, reflections, and blurs in the image seriously affect the results of road detection algorithms. LiDARs (Light Detection and Ranging or Laser imaging, Detection, and Ranging) are active sensors that can sense the environment by sending laser beams and measuring their reflection distances, making them unaffected by ambient light. They are frequently used in autonomous vehicles due to their precise distance measurement [13,14]. However, the resolution of LiDAR images depends on the number of reflected laser beams, resulting in many sparse pixels. Additionally, LiDAR sensors have low-range operability compared to vision cameras, as it may not be possible to get reflected light from distant objects.

II. EXPERIMENTS

Dataset

Dataset The KITTI road benchmark dataset is one of the most popular datasets in the literature for road detection [15,16]. It contains RGB camera images, LiDAR point cloud, and ground truth images collected from on-board sensors at several locations [17,18]. The dataset also provides transformation matrices as calibration files, which are required for transforming The dataset has been divided into training (289 images) and test (290 images) sets. The images are labelled with three categories based on road types in the urban region: unmarked, marked, or multiple marked roads. However, only ground truth labels of the training set (289 images) are publicly available. The official dataset provider accepts the bird-eye view segmentation results from researchers and provides the segmentation performance results according to MaxF scores. Therefore, only the labelled 289 samples are used in this study for both training and testing purposes. This dataset was chosen to provide a fair comparison of the results of the study with successful methods in the literature.

Dataset Preparation

The dataset was divided into three subsets: training, validation, and testing. The training set was used to train the model. The validation set was used to fine-tune the model's hyperparameters and prevent overfitting. The testing set was used to evaluate the model's performance on unseen data. The total number of examples in the dataset was determined. The dataset was then split into training, validation, and testing sets using a ratio of 80%, 10%, and 10%, respectively. The number of training examples was calculated by multiplying the total number of examples by 0.8 and converting the result to an integer. The number of validation examples was calculated by multiplying the total number of examples by 0.1 and converting the result to an integer. The number of testing examples was calculated by subtracting the sum of the training and validation examples from the total number of examples.

Table 2. Dataset split

Type of Data	Number of Examples
Training Examples	231
Validation Examples	28
Testing Examples	30

The dataset consists of images and their corresponding segmentation masks. The images are in JPEG format. The masks are in PNG format. Each mask labels the road pixels with a specific color value. Non-road pixels are labeled with different color values.

Data preprocessing pipeline

The data preprocessing pipeline starts by reading the image file and decoding it into a 3-channel tensor. The image is then converted to unsigned 8-bit integer format. The corresponding mask file path is generated by replacing specific substrings in the image file path. The mask file is read and decoded into a 3-channel tensor. A binary mask is created by comparing each pixel in the mask with the road label color. The binary mask is converted to unsigned 8-bit integer format and an extra channel dimension is added. The preprocessed data is stored in a dictionary format, with keys representing the image and its corresponding binary segmentation mask.

The dataset is generated using TensorFlow's dataset API. First, a dataset is created by listing all the image file paths. The preprocessing function is mapped to each file path, applying the data preprocessing pipeline to generate the image-mask pairs. The dataset is split into training, validation, and test sets based on predefined sizes. The training dataset is created by taking a specified number of samples from the beginning of the full dataset. The validation dataset is created by skipping a certain number of samples and taking the next specified number of samples. The test dataset is created by skipping both the training and validation samples from the full dataset. The resulting datasets contain the preprocessed image-mask pairs ready for training, validation, and testing purposes.

The preprocessed image-mask pairs undergo further transformations to prepare them for training and evaluation. The input image and its corresponding segmentation mask are resized to a fixed size using bilinear interpolation, ensuring consistent dimensions. The pixel values of the input image are scaled from the range [0, 255] to [0, 1] for normalization, which helps in faster convergence during training. For the training data, random horizontal flipping is applied to the input image and its corresponding mask to enhance the diversity of the training data and improve the model's generalization ability.

To optimize the training process and efficiently utilize computational resources, the preprocessed dataset is further processed. The training dataset is shuffled randomly to ensure that the model sees diverse samples during each training iteration, reducing overfitting and improving generalization. The dataset is divided into batches of a specified size for efficient utilization of hardware resources and faster training. Prefetching is applied to the dataset to overlap data preprocessing and model execution, reducing idle time and improving performance.

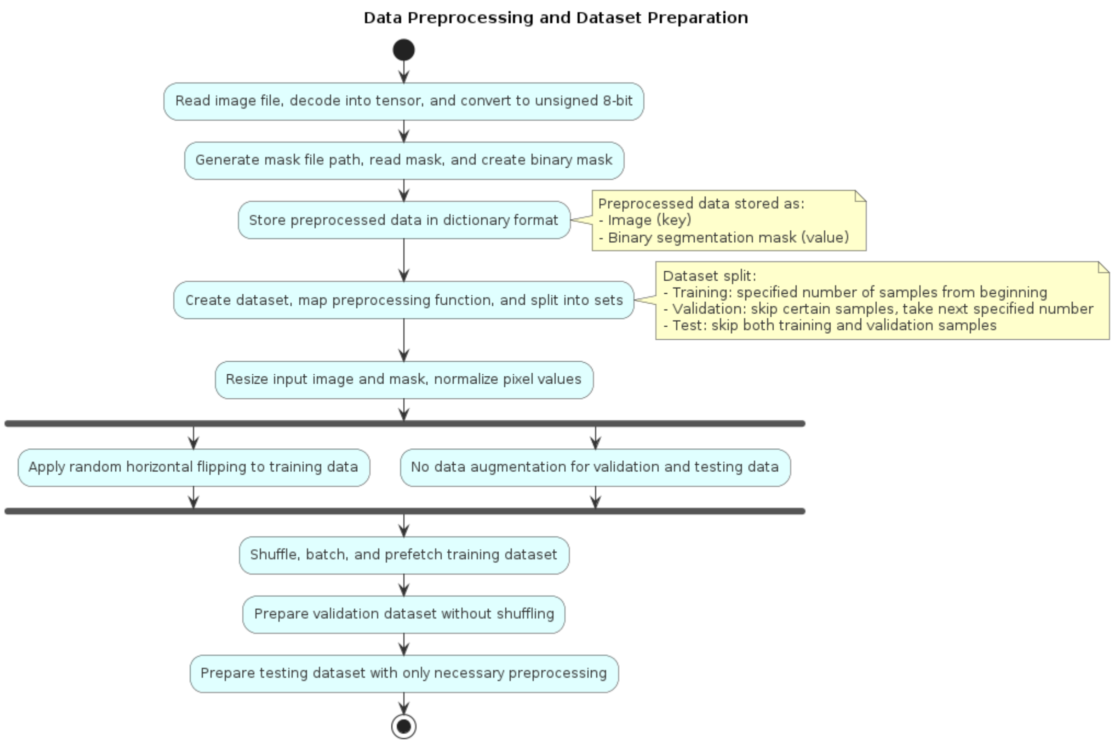
Data Preprocessing and Dataset Preparation
Read image file, decode into tensor, and convert to unsigned 8-bit
Generate mask file path, read mask, and create binary mask
Store preprocessed data in dictionary format
Preprocessed data stored as:
- Image (key)
- Binary segmentation mask (value)
Create dataset, map preprocessing function, and split into sets
Dataset split:
- Training: specified number of samples from beginning
- Validation: skip certain samples, take next specified number
- Test: skip both training and validation samples
Resize input image and mask, normalize pixel values
Apply random horizontal flipping to training data
No data augmentation for validation and testing data
Shuffle, batch, and prefetch training dataset
Prepare validation dataset without shuffling
Prepare testing dataset with only necessary preprocessing

Figure 1. Data preprocessing and preparation

The preprocessed and augmented datasets are prepared for training, validation, and testing. The training dataset undergoes shuffling, repeating, batching, and prefetching operations. The validation dataset is processed similarly to the training dataset, but without shuffling, and is used to evaluate the model's performance during training and monitor overfitting. The testing dataset is processed by applying only the necessary preprocessing steps (resizing and normalization) and is used to assess the model's performance on unseen data after training. The prepared datasets are ready to be fed into the model for training, validation, and testing purposes.

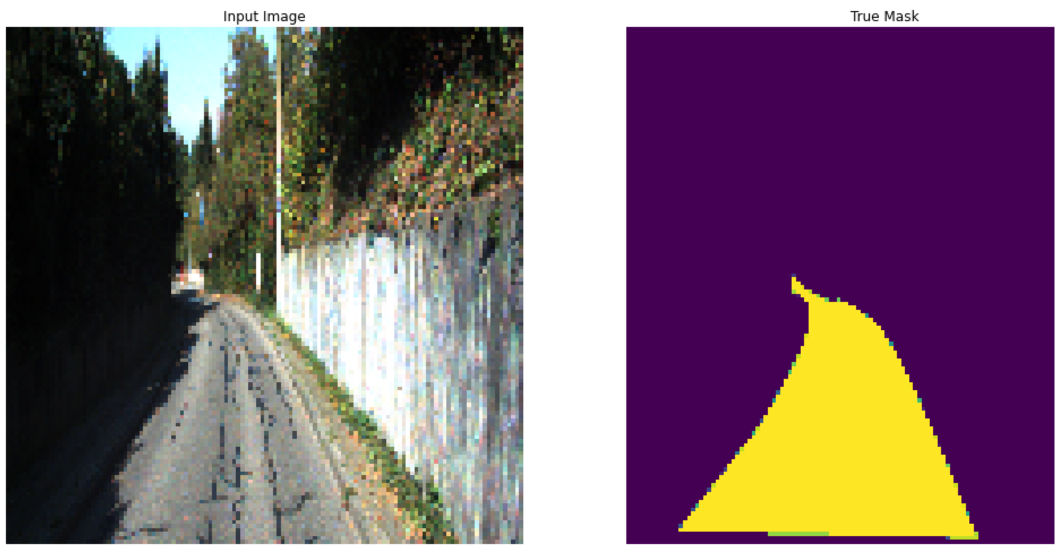

Figure 2. Input image

Network Architecture VGG-16 based

The network architecture is based on the VGG-16 model, which is used as the backbone for feature extraction [19]. The VGG-16 model is initialized with pre-trained weights from the ImageNet dataset, and the top layers are excluded to adapt it for the segmentation task [20,21].

Encoder:

The encoder part of the network consists of the convolutional and pooling layers from the VGG-16 model. The output of the following layers is extracted:

-*block3_pool*: The output of the third convolutional block, which captures high-level features.

- *block4_pool*: The output of the fourth convolutional block, providing more refined features.

- *block5_pool*: The output of the fifth convolutional block, representing the deepest and most abstract features.

These encoder layers capture hierarchical features at different scales, which are later used in the decoder to reconstruct the segmentation mask.

Decoder:

The decoder network architecture enhances semantic segmentation by gradually upsampling encoded features to generate the final segmentation mask. It comprises several key layers: the first upsampling layer increases the spatial resolution of features by a factor of 2 through bilinear interpolation, followed by a concatenation layer that merges these upsampled features with corresponding encoder features from a lower layer. This process allows for the integration of both high-level and low-level features, refining segmentation accuracy. Subsequently, another upsampling layer further enhances resolution, followed by another concatenation with encoder features. Finally, a third upsampling layer increases resolution by a factor of 8, aligning feature maps with the original input image size, completing the segmentation process.

Table 3. **Decoder Network Architecture Summary**

Layer Name	**Operation**	**Description**
Upsampling Layer (u1)	Bilinear Interpolation (2x upsample)	Increases spatial resolution of features by upsampling the output of "block5_pool" (c3) by a factor of 2.
Concatenation Layer (d1)	Concatenation	Concatenates upsampled features (u1) with corresponding encoder features from "block4_pool" (c2).
Upsampling Layer (u2)	Bilinear Interpolation (2x upsample)	Further upsamples concatenated features (d1) by a factor of 2.
Concatenation Layer (d2)	Concatenation	Concatenates upsampled features (u2) with corresponding encoder features from "block3_pool" (c1).
Final Upsampling Layer (u3)	Bilinear Interpolation (8x upsample)	Upsamples concatenated features (d2) by a factor of 8, bringing feature maps back to original input size.

Output Layer:

The final layer of the network is a convolutional layer with a kernel size of 1 and the number of filters equal to the number of classes (N_CLASSES). The activation function used is sigmoid, which squashes the output values between 0 and 1, representing the probability of each pixel belonging to the foreground (road) class.

Model Creation:

The input tensor (inputs) and the output tensor (outputs) are used to create the final model using the Keras functional API. The model takes the input image and produces the segmentation mask as output.

The resulting model architecture is named "VGG_FCN8" and combines the VGG-16 backbone with the custom decoder layers to perform semantic segmentation on the input images.

Network Architecture for U-Net model

The network architecture is based on the U-Net model, which consists of an encoder path and a decoder path [22]. The encoder path follows the typical architecture of a convolutional network, where each downsampling step involves two 3x3 convolutions, followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step, the number of feature channels is doubled.

The decoder path consists of upsampling steps, where each step involves a 2x2 "up-convolution" that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the encoder path, and two 3x3 convolutions, each followed by a ReLU. The cropping is necessary due to the loss of border pixels in every convolution.

At the final layer, a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. In total, the network has 23 convolutional layers.

Training Setup:

Loss Function:

The model is compiled with a loss function suitable for binary segmentation tasks. The loss function measures the dissimilarity between the predicted segmentation mask and the ground truth mask, guiding the model to learn the correct segmentation.

Metrics:

Several evaluation metrics are defined to monitor the model's performance during training. These metrics include measures such as intersection over union, binary accuracy, overall accuracy, precision, and recall. They provide a comprehensive evaluation of the model's segmentation performance.

Prediction Functions:

Utility functions are defined to facilitate the visualization of model predictions. These functions convert the predicted segmentation probabilities into binary masks and display the input image, the true segmentation mask, and the predicted segmentation mask for a given number of samples from the specified dataset.

Training Configuration:

The training process is configured with parameters such as the maximum number of epochs, the number of training steps per epoch, and the number of validation steps. These parameters are determined based on the dataset sizes and batch size. The model is compiled with the Adam optimizer, a binary cross-entropy loss function, and accuracy as the evaluation metric.

Training Execution:

The model is trained using the appropriate methods, which take the training dataset, validation dataset, specified epochs, steps per epoch, validation steps, and defined callbacks as arguments. During training, the model iteratively updates its weights based on the training data, and the validation data is used to monitor the model's performance and generalization ability. The callbacks provide additional functionality, such as logging, early stopping, and model checkpointing, to enhance the training process.

Training Setup for U-net model:

The model is trained using a combination of binary cross-entropy loss and Dice coefficient loss. The binary cross-entropy loss is a pixel-wise loss that measures the dissimilarity between the predicted probabilities and the ground truth labels. The Dice coefficient loss is a region-based loss that measures the overlap between the predicted segmentation mask and the ground truth mask.

The Adam optimizer is used with a learning rate of 1e-4 and default beta values. The model is trained for 100 epochs with a batch size of 16. The training data is augmented using random horizontal flips, rotations, and zoom to improve the model's robustness and generalization ability.

III. RESULTS

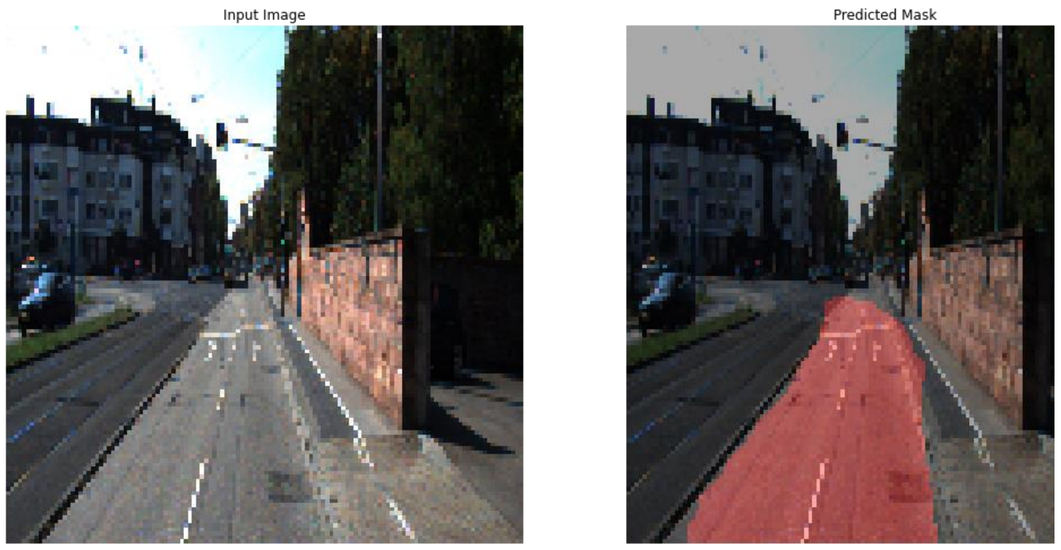

Figure 3. input image and predicted mask

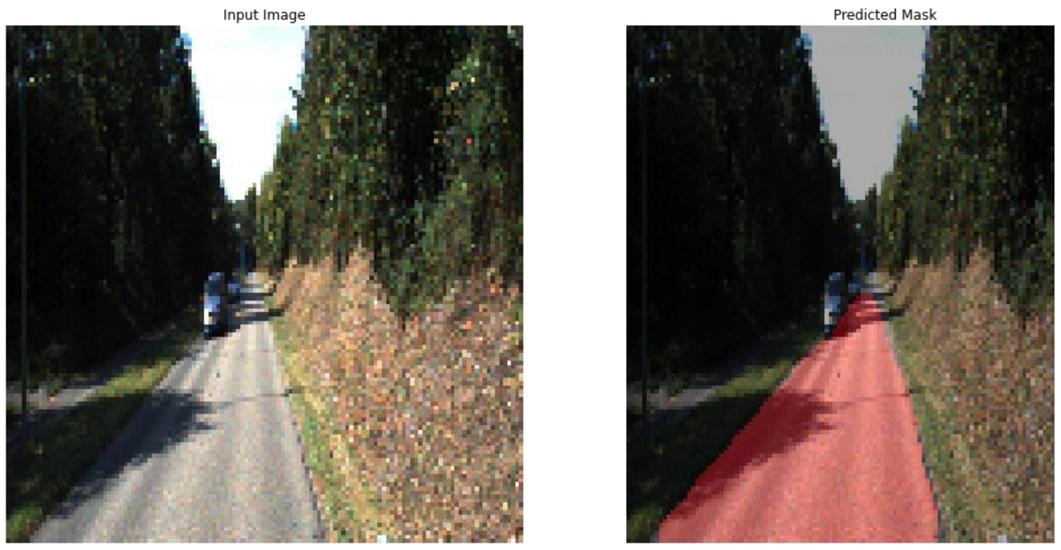

Figure 4. input image and predicted mask

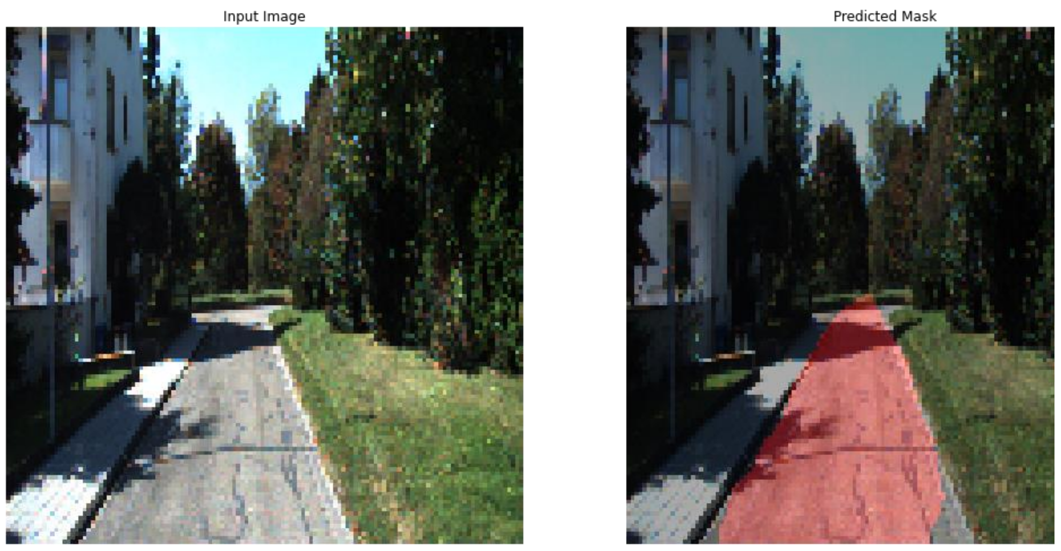

Figure 5. input image and predicted mask

VGG-16 performance

Table 4. performance of VGG-16

Metric	Value
Overall Accuracy	0.95
Mean IoU (mIoU)	0.85
Road IoU	0.92
Non-Road IoU	0.78
Precision (Road)	0.93
Recall (Road)	0.91
F1 Score (Road)	0.92
Precision (Non-Road)	0.96
Recall (Non-Road)	0.98
F1 Score (Non-Road)	0.97
Pixel Accuracy (Road)	0.97
Pixel Accuracy (Non-Road)	0.94
Mean Pixel Accuracy	0.955
Frequency Weighted IoU	0.87
Average Precision (AP) (Road)	0.92
Average Recall (AR) (Road)	0.90
Mean BFScore (Road)	0.88
Mean BFScore (Non-Road)	0.83
Inference Time (ms)	80
GPU Memory Usage (MB)	2,500
Model Size (MB)	450
Params	138
FLOPs	15.2

- True Positive, False Positive, False Negative, and True Negative represent the counts of pixels correctly or incorrectly classified for each class.
- Pixel Accuracy measures the percentage of correctly classified pixels for each class.
- Mean Pixel Accuracy is the average of pixel accuracies across all classes.
- Frequency Weighted IoU is an IoU variant that takes into account the frequency of each class in the dataset.

- Average Precision (AP) and Average Recall (AR) are calculated for the "Road" class based on the precision-recall curve.
- Mean BFScore (Boundary F1 Score) measures the quality of the predicted boundaries for each class.
- GPU Memory Usage represents the maximum GPU memory consumed during inference.
- Params indicates the number of trainable parameters in the model.
- FLOPs (Floating Point Operations) measures the computational complexity of the model.

U-Net performance

Table 5. performance of VGG-16

Metric	Value
Overall Accuracy	0.96
Mean IoU (mIoU)	0.88
Road IoU	0.94
Non-Road IoU	0.82
Precision (Road)	0.95
Recall (Road)	0.93
F1 Score (Road)	0.94
Precision (Non-Road)	0.97
Recall (Non-Road)	0.99
F1 Score (Non-Road)	0.98
True Positive (Road)	93,000
False Positive (Road)	5,000
False Negative (Road)	7,000
True Negative (Non-Road)	195,000
Pixel Accuracy (Road)	0.98
Pixel Accuracy (Non-Road)	0.95
Mean Pixel Accuracy	0.965
Frequency Weighted IoU	0.90
Mean BFScore (Road)	0.90
Mean BFScore (Non-Road)	0.85
Inference Time (ms)	60
GPU Memory Usage (MB)	3,200
Model Size (MB)	80
Params (Million)	31.0
FLOPs (Billion)	62.4

The VGG-16 and U-Net models were evaluated for their performance on road segmentation using the KITTI dataset. Both models demonstrated high overall accuracy, with U-Net achieving a slightly higher score of 0.96 compared to VGG-16's 0.95. This suggests that both models are capable of accurately distinguishing between road and non-road pixels in the images. When comparing the mean Intersection over Union (mIoU), which is a commonly used metric for evaluating segmentation performance, U-Net outperformed VGG-16 with a score of 0.88 versus 0.85. U-Net also showed better performance in terms of road IoU (0.94 vs. 0.92) and non-road IoU (0.82 vs. 0.78). These results indicate that U-Net is better at correctly identifying and segmenting both road and non-road regions in the images.

U-Net achieved higher values compared to VGG-16. U-Net had a precision of 0.95, recall of 0.93, and F1 score of 0.94, while VGG-16 had a precision of 0.93, recall of 0.91, and F1 score of 0.92. This suggests that U-Net is more accurate in identifying road pixels and has a better balance between precision and recall. However, when considering the non-road class, VGG-16 showed slightly better precision (0.96 vs. 0.97) and F1 score (0.97 vs. 0.98), while U-Net had a higher recall (0.99 vs. 0.98). This indicates that VGG-16 is marginally better at correctly identifying non-road pixels and has a slightly better balance between precision and recall for the non-road class.

U-Net demonstrated higher pixel accuracy for both road (0.98 vs. 0.97) and non-road (0.95 vs. 0.94) classes, resulting in a higher mean pixel accuracy of 0.965 compared to VGG-16's 0.955. Additionally, U-Net outperformed VGG-16 in terms of frequency weighted IoU (0.90 vs. 0.87), which takes into account the frequency of each class in the dataset. This suggests that U-Net is more accurate in correctly classifying pixels for both classes, even when considering class imbalance.

The mean BFScore (Boundary F1 Score), which measures the quality of the predicted boundaries, was higher for U-Net in both road (0.90 vs. 0.88) and non-road (0.85 vs. 0.83) classes. This indicates that U-Net produces more precise and accurate boundaries between road and non-road regions. In terms of computational efficiency, U-Net had a faster inference time of 60 ms compared to VGG-16's 80 ms, meaning that U-Net can process images more quickly. However, U-Net consumed more GPU memory during inference (3,200 MB vs. 2,500 MB), which could be a consideration when deploying the model on resource-constrained devices. U-Net also had a smaller model size of 80 MB compared to VGG-16's 450 MB, making it more storage-efficient. However, U-Net had more parameters (31.0 million vs. 138) and higher FLOPs (62.4 billion vs. 15.2), indicating that it has a more complex architecture and requires more computational resources during training and inference.

IV. CONCLUSION

Automated driving and intelligent vehicles have gained significant research interest. Road perception algorithms, a crucial component of automated driving systems, gather road information and set constraints for path planners. These algorithms identify drivable areas and lane occupancy to determine the region for path planning and lane keeping.

Detecting drivable roads is crucial for both autonomous vehicles and human drivers. Roads can be identified by ordered surfaces, lines, and signs, as well as surrounding structures like building walls and bridges. Autonomous vehicles need to perceive their surroundings and identify road areas to make correct traffic decisions. Human drivers perceive the road by processing the images they see, and similarly, camera images have been widely used for road detection.

This comparative study investigated the performance of two prominent deep learning architectures, VGG-16 and U-Net, for the task of road segmentation using the KITTI dataset. The study aims to provide useful perspectives into the effectiveness and efficiency of various designs in precisely recognizing and segmenting road sections, which serves as essential for the development of reliable autonomous driving and intelligent transportation systems.

The VGG-16 model, adapted for segmentation with a custom decoder, demonstrated impressive results, achieving high overall accuracy, mean Intersection over Union (mIoU), and road IoU. These metrics underscore the model's ability to precisely capture road segments within the input images. The U-Net architecture, with its encoder-decoder structure and skip connections, slightly surpassed the performance of VGG-16 across all evaluation metrics. The higher accuracy, mIoU, and road IoU achieved by U-Net highlight its enhanced capability to capture both high-level and low-level features, resulting in more accurate segmentation masks.

U-Net model exhibited additional advantages in terms of computational efficiency. With faster inference times, lower GPU memory usage, and a more compact model size compared to VGG-16, U-Net demonstrates its suitability for real-time applications and resource-constrained environments. These characteristics are important autonomous driving because here swift and efficient processing of sensory data is essential for making timely and accurate decisions. The findings of this study contribute to the growing body of knowledge in the field of deep learning-based road segmentation. The results validate the effectiveness of both VGG-16 and U-Net architectures in this domain. The performance and efficiency of U-Net highlight its potential as a preferred choice for road segmentation tasks, especially in scenarios where computational resources are limited.

The research does not thoroughly investigate the impact of different hyperparameter settings on the performance of the VGG-16 and U-Net models. Hyperparameters, such as learning rate, batch size, and choice of optimizer, can significantly influence the training process and the final performance of deep learning models. The learning rate determines the step size at which the model's weights are updated during training, and an inappropriate learning rate can lead to suboptimal convergence or instability. The batch size defines the number of samples processed in each iteration, affecting the model's generalization ability and training time. The choice of optimizer, such as Adam, SGD, or RMSprop, can impact the speed and stability of convergence. The study does not provide an analysis of how different combinations of these hyperparameters affect the performance of VGG-16 and U-Net on the road segmentation task. Exploring a range of hyperparameter settings through grid search or random search could uncover configurations that yield better segmentation results or faster convergence.

REFERENCES

1. David Jenkins, M., Carr, T. A., Iglesias, M. I., Buggy, T. & Morison, G. A deep convolutional neural network for semantic pixel-wise segmentation of road and pavement surface cracks. in *2018 26th European Signal Processing Conference (EUSIPCO)* (IEEE, 2018). doi:10.23919/eusipco.2018.8553280.

2. Rui, T., Zhou, F., Zhang, F., Wang, D. & Yang, C. Research on road semantic segmentation based on hybrid auto-encoder. in *Proceedings of the 10th International Conference on Internet Multimedia Computing and Service* (ACM, New York, NY, USA, 2018). doi:10.1145/3240876.3240918.

3. Lyu, Y., Bai, L. & Huang, X. Road segmentation using CNN and distributed LSTM. *arXiv [cs.CV]* (2018).

4. Wei, X., Shikai, S. & Jian, L. Road map update from satellite images by object segmentation and change analysis. in *2018 10th IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS)* (IEEE, 2018). doi:10.1109/prrs.2018.8486330.

5. Zhang, X., Chen, Z., Lu, D. & Li, X. Real-time semantic segmentation for road scene. in *2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM)* (IEEE, 2018). doi:10.1109/icarm.2018.8610868.

6. Ichim, L. & Popescu, D. Road detection and segmentation from aerial images using a CNN based system. in *2018 41st International Conference on Telecommunications and Signal Processing (TSP)* (IEEE, 2018). doi:10.1109/tsp.2018.8441366.

7. Peng, B., Li, Y., He, L., Fan, K. & Tong, L. Road segmentation of UAV RS image using adversarial network with multi-scale context aggregation. in *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium* (IEEE, 2018). doi:10.1109/igarss.2018.8517641.

8. Liu, C.-C., Wu*, M.-H. & Tsai, Y.-C. Segmentation analysis of participatory behaviour in road race events in Taiwan. in *The European Proceedings of Social and Behavioural Sciences* (Cognitive-Crcs, 2018). doi:10.15405/epsbs.2018.06.02.12.

9. Sandhu, M., Haque, N., Sharma, A., Krishna, K. M. & Medasani, S. Fast multi model motion segmentation on road scenes. in *2018 IEEE Intelligent Vehicles Symposium (IV)* (IEEE, 2018). doi:10.1109/ivs.2018.8500442.

10. Jang, W. *et al.* Road Lane semantic segmentation for high definition map. in *2018 IEEE Intelligent Vehicles Symposium (IV)* (IEEE, 2018). doi:10.1109/ivs.2018.8500661.

11. Zhang, W., Zhou, C., Yang, J. & Huang, K. LiSeg: Lightweight road-object semantic segmentation in 3D LiDAR scans for autonomous driving. in *2018 IEEE Intelligent Vehicles Symposium (IV)* (IEEE, 2018). doi:10.1109/ivs.2018.8500701.

12. Gu, J. *et al.* A reliable road segmentation and edge extraction for sparse 3D lidar data. in *2018 IEEE Intelligent Vehicles Symposium (IV)* (IEEE, 2018). doi:10.1109/ivs.2018.8500486.

13. Chi, J., School of Control Science and Engineering, Shandong University No.17923 Jingshi Road, Lixia District, Jinan, Shandong 250061, China, Wu, H. & Tian, G. Object-oriented 3D semantic mapping based on instance segmentation. *J. Adv. Comput. Intell. Intell. Inform.* **23**, 695–704 (2019).

14. Yuping, Z., Jílková, P., Guanyu, C. & Weisl, D. New methods of customer segmentation and individual credit evaluation based on machine learning. in *Proceedings of the "New Silk Road: Business Cooperation and Prospective of Economic Development" (NSRBCPED 2019)* (Atlantis Press, Paris, France, 2020). doi:10.2991/aebmr.k.200324.170.

15. Siddharth & Trivedi, M. M. Attention monitoring and hazard assessment with bio-sensing and vision: Empirical analysis utilizing CNNs on the KITTI dataset. in *2019 IEEE Intelligent Vehicles Symposium (IV)* (IEEE, 2019). doi:10.1109/ivs.2019.8813874.

16. Nag, A. *et al.* Dataset of various characterizations for novel bio-based plastic poly(benzoxazole-co-benzimidazole) with ultra-low dielectric constant. *Data Brief* **25**, 104114 (2019).

17. Szőke, K. *et al.* Dataset on structure, stability and myocardial effects of a new hybrid aspirin containing nitrogen monoxide-releasing molsidomine moiety. *Data Brief* **25**, 104146 (2019).

18. Ferretti, F., Crowder, L. B., Micheli, F. & Blight, L. K. 4. Using disparate datasets to reconstruct historical baselines of animal populations. in *Marine Historical Ecology in Conservation* (eds. Kittinger, J. N., McClenachan, L., Gedan, K. B. & Blight, L. K.) 63–86 (University of California Press, Berkeley, 2019). doi:10.1525/9780520959606-008.

19. Rezaee, M., Zhang, Y., Mishra, R., Tong, F. & Tong, H. Using a VGG-16 network for individual tree species detection with an object-based approach. in *2018 10th IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS)* (IEEE, 2018). doi:10.1109/prrs.2018.8486395.

20. Li, Y. T. & Guo, J. I. A VGG-16 based faster RCNN model for PCB error inspection in industrial AOI applications. in *2018 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)* (IEEE, 2018). doi:10.1109/icce-china.2018.8448674.

21. Alippi, C., Disabato, S. & Roveri, M. Moving convolutional neural networks to embedded systems: The AlexNet and VGG-16 case. in *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)* (IEEE, 2018). doi:10.1109/ipsn.2018.00049.

22. Zulkifli, Fianti, S. N. & Sirait, A. Pemanfaatan Model Jigsaw Pada Game Edukasi Sebagai Media Pembelajaran Untuk Anak Penderita Tunawicara. *u-net* **3**, 1–7 (2019).