Scientific Research Publishing

Journal of Computer and Communications, 2024, 12, 188-207
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Efficient Vision Transformers for Autonomous Off-Road Perception Systems

Max H. Faykus III*, Adam Pickeral*, Ethan Marquez, Melissa C. Smith, Jon C. Calhoun

Holcombe Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, USA
Email: mfaykus@clemson.edu, apicker@clemson.edu, marque6@clemson.edu, smithmc@clemson.edu, joncal@clemson.edu

How to cite this paper: Faykus III, M.H., Pickeral, A., Marquez, E., Smith, M.C. and Calhoun, J.C. (2024) Efficient Vision Transformers for Autonomous Off-Road Perception Systems. *Journal of Computer and Communications*, **12**, 188-207.
https://doi.org/10.4236/jcc.2024.129011

Received: July 23, 2024
Accepted: September 27, 2024
Published: September 30, 2024

Abstract

The development of autonomous vehicles has become one of the greatest research endeavors in recent years. These vehicles rely on many complex systems working in tandem to make decisions. For practical use and safety reasons, these systems must not only be accurate, but also quickly detect changes in the surrounding environment. In autonomous vehicle research, the environment perception system is one of the key components of development. Environment perception systems allow the vehicle to understand its surroundings. This is done by using cameras, light detection and ranging (LiDAR), with other sensor systems and modalities. Deep learning computer vision algorithms have been shown to be the strongest tool for translating camera data into accurate and safe traversability decisions regarding the environment surrounding a vehicle. In order for a vehicle to safely traverse an area in real time, these computer vision algorithms must be accurate and have low latency. While much research has studied autonomous driving for traversing well-structured urban environments, limited research exists evaluating perception system improvements in off-road settings. This research aims to investigate the adaptability of several existing deep-learning architectures for semantic segmentation in off-road environments. Previous studies of two Convolutional Neural Network (CNN) architectures are included for comparison with new evaluation of Vision Transformer (ViT) architectures for semantic segmentation. Our results demonstrate viability of ViT architectures for off-road perception systems, having a strong segmentation accuracy, lower inference speed and memory footprint compared to previous results with CNN architectures.

Keywords

Semantic Segmentation, Off-Road Vision, Transformers, CNNs, Autonomous Driving

*Indicates Equal Contribution.

1. Introduction

Off-road autonomous vehicles, or Unmanned Ground Vehicles (UGVs), are important research efforts in academia and industry. UGVs are typically deployed in challenging terrain and atypical conditions not pre-built for driving. For example, small UGVs are often deployed for terrain exploration or monitoring [1]. Larger UGVs have many uses in military surveillance and defense [2]. These unmanned vehicles use many systems in tandem to make traversal decisions, one of the most crucial being the perception system [3].

Perception systems are classified into two main subsystems: the Position Estimation System and the Environment Perception System [4]. Position Estimation Systems typically use satellite GPS or Inertial Measurement Units (IMUs) to estimate the position of the vehicle [4]. Environment Perception Systems acquire knowledge by scanning the surrounding terrain to detect changes in driving conditions, using various sensors to gather information [4]. Sensor examples include Radio Detection and Ranging (RADAR), Light Detection and Ranging (LiDAR), and various types of cameras [4]. Since RADAR does not support simultaneous detection of multiple objects and LiDAR is used for point-cloud distance data [5]. The cameras used are an important sensor modality that provides fine-grain environmental knowledge for perception systems.

Typically multiple types of camera data—including truecolor red-green-blue (RGB) images, Forward Looking Infrared (FLIR) thermal, multispectral and stereo, are used in an environment perception system. This work focuses on extraction of meaningful insights from RGB image data, also known as Computer Vision [3]. In environment perception, object detection and semantic segmentation are the two most common computer vision tasks with RGB images. Object detection classifies objects in images and marks their location, typically using a bounding box [6]. Semantic segmentation classifies every pixel in an image, providing more fine-grained detail of an image's scene than object detection but at a higher computational cost [7]. Since autonomous vehicles require details of their surroundings for safe and accurate decision-making, semantic segmentation is a crucial element of environment perception systems.

Methods used for semantic segmentation take on numerous forms. In recent years, multi-layered neural networks (deep learning) have shown stronger results in semantic segmentation and other computer vision tasks compared to traditional rule-based algorithms. The use of deep learning has generated much research in the creation of new semantic segmentation neural network designs, or "architectures". The design of an architecture determines how the neural network represents patterns in data or "learns" information. Convolutional neural network (CNN) architectures have traditionally been the state of the art for semantic segmentation [8]-[11]. Vision Transformer (ViT) architectures have recently shown comparable, and in some cases superior, results against CNN architectures in semantic segmentation and other computer vision tasks [12] [13].

Semantic segmentation architectures, in autonomy or otherwise, are typically

evaluated by their ability to segment scenes in well-structured environments; most benchmark datasets contain images from urban areas [14] [15]. For autonomous urban travel, cars, street signs, and roads with distinct lines and intersections give perception systems visual cues for path planning [14]. However, far less research has evaluated semantic segmentation architectures in off-road or unstructured settings. Modeling off-road environments common for UGV deployment, e.g., forests, country roads, deserts, is difficult with meshing between naturally present objects and generally noisy terrain. These environments are rarely studied with recently developed deep learning architectures, observed from: 1) the scarcity of quality, labeled image datasets created in off-road environments [16]; 2) the absence of studies using semantic segmentation architectures with available off-road datasets [3].

In addition, the real-world deployment of deep learning architectures introduces other constraints. Typically UGVs are deployed with limited computational resources. Therefore, semantic segmentation architectures that use large amounts of calculations and time to produce insights about a surrounding environment are undesirable. It is crucial that the architectures deployed in perception systems not only be accurate, but be able to compute segmentation predictions ("inference") quickly on devices with limited computational resources (typically "edge" devices).

This work evaluates the viability of multiple deep learning architectures for use in off-road UGVs, where accuracy must be high and inference speed fast. State-of-the-art architectures are evaluated on the basis of their ability to both accurately segment off-road data and inference quickly. Specifically, results from two different CNN architectures—DeepLabV3+ [17] and Swiftnet [18], are compared with results from two different ViT architectures—EfficientViT [19] and Segformer [20], on off-road data.

This paper contributes the following to the literature:

- Performance evaluation of state-of-the-art ViT architectures in off-road environments;
- Analysis of ViT vs. CNN architectures for determining traversable terrain through semantic segmentation;
- Inference speed analysis of ViT vs. CNN architectures for real-time use.

2. Background and Related Works

When evaluating segmentation architectures, there is a trade-off between accuracy and inference speed [21]-[23]. Typically, more accurate architectures take longer to inference. Conversely, architectures with higher inference speeds normally suffer from accuracy loss.

Finding a balance between accuracy and inference speed is important in off-road autonomous driving for several reasons: 1) Safety: For the vehicle to make decisions that alleviate damage to itself or any cargo, it must have an accurate representation surrounding environments to determine traversable terrain. 2)

Latency Expectations: Deployment of these vehicles for missions requires real-time decision-making. Perception systems must be able to determine changes in an environment multiple times a second. 3) Constrained Resources: UGVs are typically deployed using edge devices for perception system computation. Ensuring the perception system efficiently segments RGB camera data on devices with limited computational resources (small GPUs and low memory) is crucial for deployment.

This section details deep learning architectures explored in off-road settings, the image datasets used to evaulate them.

2.1. Deep Learning Architectures for Semantic Segmentation

To effectively segment RGB off-road camera data, a variety of deep learning architectures were utilized. The architectures used in this work fall broadly into two categories: Convolutional Neural Networks and Vision Transformers.

Convolutional neural networks (CNNs) use kernels, also known as filters, to extract spatial features from images [9]. Kernels are typically represented by a small square grid, where each grid element contains a numerical value. This grid transforms images by transforming an input image into a new representation, where each new pixel value is a weighted sum of all pixels in the grid's window. The weights of neighboring pixels that contribute to a new pixel's sum, the kernel's values, are learned through the training process and updated to recognize patterns in images that pertain to specific classes in the input data. The combination of these convolutional layers creates deep neural networks that have proven to work well for computer vision tasks [9] [24] [25].

Transformer architectures [26], originally designed for natural language processing, use self-attention as a means to represent complex patterns in sequences. Self-attention uses attention "heads" to extract information from a sequence of data. Each head transforms individual parts of an input sequence into representations of "Queries", questions about the data, "Keys", answers to these queries, and "Values", which determine how data should be transformed based on matching queries and keys. Each head transforms input sequence data into queries, keys, and values using weights learned through training. Based on the relationship between queries, keys, and values in each head, "attention maps" are created, highlighting complex relationships in the input data, such as semantic meaning or relationships to other data points. Each head creates distinct queries, keys, and values; resulting attention map data from multiple heads are combined to aggregate information. The Vision Transformer architecture (ViT) modified the idea to work with computer vision tasks [13]. New semantic segmentation architectures based on the ideas of the ViT typically have two things in common. First, most multi-scale architectures process input images at multiple scales to combine fine and coarse features [12]. Second, they use attention to create a global receptive field, meaning relationships between patterns everywhere in the image are considered [12].

CNN-based architectures are limited by the spatial window size of their kernels; using attention allows ViT models to find unique relationships that are not limited by such spatial constraints. However, it is well studied that the transformer model is hindered by high computational costs, memory footprint, and a need for large amounts of training data to perform well [27]. Another noted downside is the quadratic computational complexity of typical attention functions, meaning the cost to compute predictions typically grows quadratically with respect to the input data size. This computational cost is incredibly cumbersome when using transformers for real-time applications in autonomous vehicles that use high-resolution images for perception. Several efforts have been made to reduce the memory and computational cost of ViT models. Some of these methods include creating hybrid architectures that combine CNN and ViT architectures which use efficient attention operations [19], or using machine learning to compress data for more efficient processing [20].

2.2. Previous CNN Studies

The DeepLabV3+ CNN architecture shown, in **Figure 1**, was investigated for image segmentation with the Rellis-3D dataset on a previous study [28]. DeeplabV3+ has four primary components: Atrous convolution, Atrous spatial pyramid pooling, and an encoder-decoder structure [17]. An atrous convolution is dilated with holes in the filter weights, allowing for denser feature maps. Atrous spatial pyramid pooling replaces general pooling and introduces global access pooling for a global context. The encoder-decoder structure utilizes a backbone (ResNet in this study) to extract meaningful features in the data.

In [30], the SwiftNet multi-scale architecture [18] was explored with Rellis-3D. Swiftnet also uses an encoder-decoder structure as shown in **Figure 2**. The

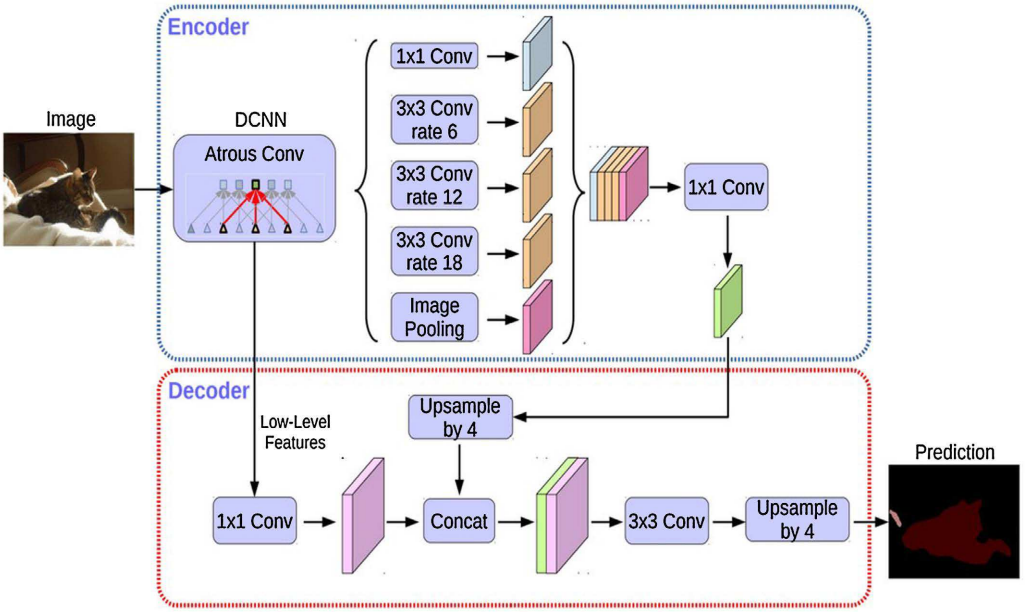

Figure 1. DeeplabV3+ Architecture [29].

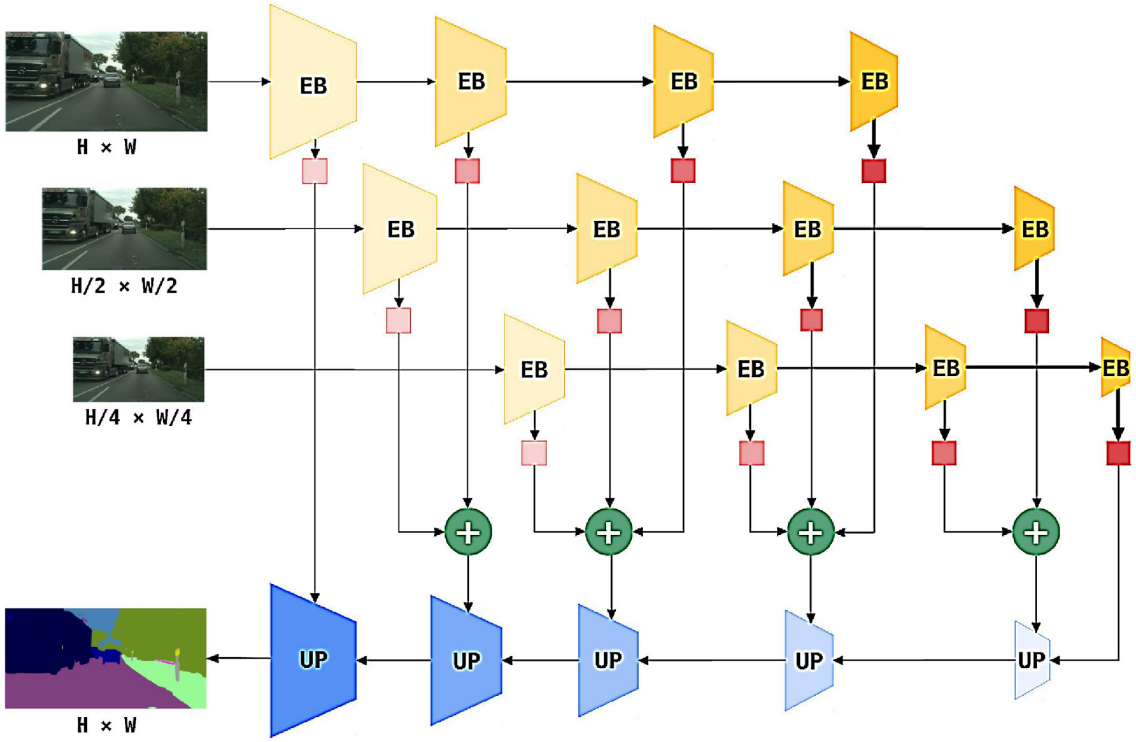

Figure 2. SwiftNet multi-scale architecture [18].

encoder blocks (EB) are comprised of ResNet-18 layers [11]. The features are extracted at three scales (full, 1/2, and 1/4 resolution) using different branches. The decoder consists of a ladder-style structure with two inputs: the low-resolution feature maps from the preceding upsampling module (UP), and the high-resolution feature maps from the encoder blocks. The feature maps are combined with summation before passage to the decoder. All upsampling is done with bilinear interpolation.

2.3. Efficient Vision Transformer Architectures

As previously discussed when comparing CNNs and ViTs, the biggest hindrance of the ViT is the memory consumption and computational cost self-attention [27]. Efforts to use hierarchical pyramidal fusions, convolutional layers, and self-supervised Vision Transformers have been made to reduce computational complexity and memory footprint [12]. In this study, two recently developed architectures, Segformer and EfficientViT, are investigated because of their cited efficient use for semantic segmentation.

2.3.1. Segformer

The Segformer architecture was created for semantic segmentation with a lightweight multi-layer perceptron (MLP) decoder and multi-scale attention [20]. The architecture uses attention at large and small scales of the input image data to capture fine-grained and coarse feature maps. Segformer uses projection to shorten input sequences into a smaller representation, making attention slightly more

efficient, although it is still quadratically complex. As shown in Figure 3, attention feature maps are created at the 1/4, 1/8, 1/16, and 1/32 scale of the original input image. These feature maps are merged and upsampled using nearest-neighbor interpolation and then passed to the decoder. The decoder uses an MLP to output a 1/4 scale prediction segmentation. For this study, we used bicubic interpolation to upsample the final prediction segmentation back to full scale.

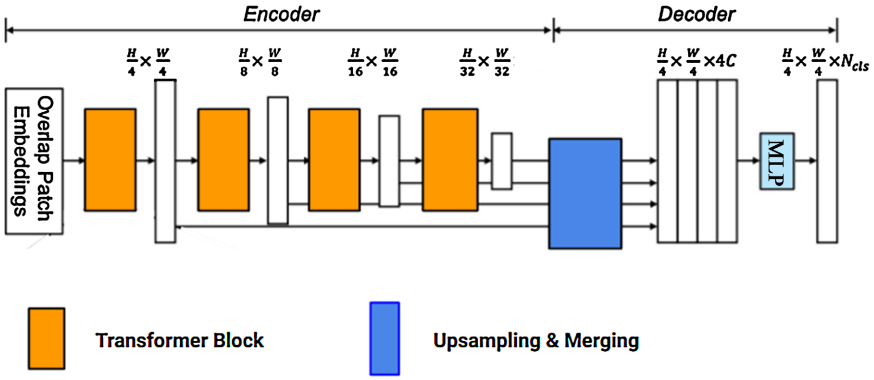

Figure 3. Segformer architecture [20].

2.3.2. EfficientViT

EfficientViT is a hybrid CNN-ViT architecture for low-latency computer vision in real-world systems. EfficientViT follows the typical encoder-decoder structure for segmentation neural networks. The encoder is pre-trained on the ImageNet dataset [31] for classification. The encoder backbone comprises an input stem and four stages that contribute to the produced feature maps. The full architecture is shown in Figure 4. The input stem is a simple convolutional layer followed by a depth-wise separable convolution layer. The first two stages consist of multiple mobile inverted bottleneck convolutional layers. Stages 3 and 4 consist of the same convolutional layers as Stages 1 and 2, followed by the EffcientViT module: a ReLU Linear Attention module with convolutions to aggregate nearby tokens. Using ReLU as a function in attention calculation, in place of the traditional softmax function [13], allows for hardware efficiency, but is weaker for discovering patterns [19]. When input data is passed through the backbone, the outputs of stages 2, 3, and 4 are saved, forming a pyramid of feature maps. Bicubic upsampling is then used to match their spatial and channel size, followed by a fusion of

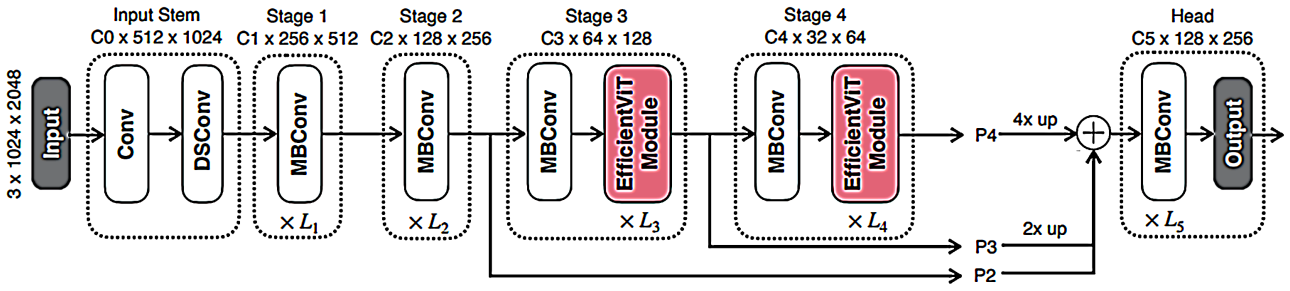

Figure 4. EfficientViT Architecture [19].

the data by addition. The head design is simple, using a few MBConv layers to decode the feature maps.

The linear attention backbone, the EfficientViT module, is shown in Figure 5. The learned query (Q), key (K), and value (V) matrices are fed into three channels before concatenation. The first layer only uses the ReLU linear attention function. The second and third layers additionally pass the resulting tokens through depth-wise separable convolution layers, with kernel sizes 3 × 3 and 5 × 5, respectively, to aggregate nearby information. The tokens are then passed through a 1 × 1 group convolution, aggregating channels into groups for efficient computation. This hierarchy creates three different scales of representation in the tokens. After passing through the Multi-Scale Linear Attention module, the data is passed through a simple feed-forward network with a depth-wise separable convolution layer to project the data further. This addition helps compensate for the weakness of ReLU as an attention function.

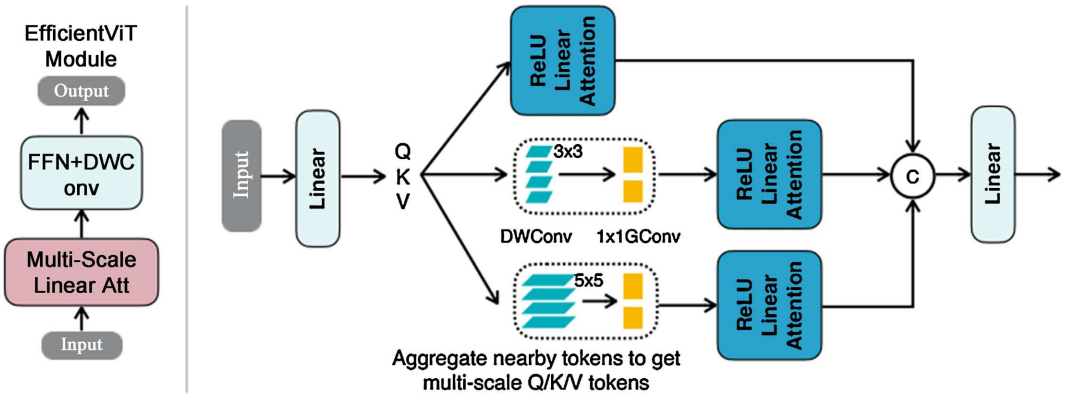

Figure 5. EfficientViT Module [19].

2.4. Datasets

To evaluate the segmentation ability of a deep learning architecture in an off-road setting, it must be trained on a large-scale dataset with labeled images of off-road environments. The datasets used in this study are the Rellis-3D dataset [32] and the CAVS Traversability (CaT) dataset [33]. These specific datasets were chosen because they contain 1) thousands of labeled images in various off-road environments; 2) images in the datasets are high-resolution. A large number of images gives architectures wider range of scenes to learn from and be evaluated on. The high resolution of the images allows architectures to make detailed predictions about the surrounding scenes, a necessary trait for real-world use. These two factors are rare to find in off-road datasets [16], making them the best choices for study in an off-road setting.

2.4.1. Rellis-3D Dataset

Rellis-3D is an off-road dataset created to fill the lack of multi-modal datasets for off-road environments. This off-road dataset challenges state-of-the-art deep learning architectures designed to segment urban data. It provides a full sensor

stack that includes RGB camera images, LiDAR point clouds, stereo images, high-precision GPS measurements, and IMU data. This multimodal data aims to enhance autonomous off-road navigation with a comprehensive ontology of object and terrain classes.

The Rellis-3D image collection contains 6234 labeled RGB images of size 1200 × 1920 [32]. **Figure 6** shows the ontology of the Rellis-3D dataset. Twenty class labels consist of two main subgroups: 1) traversable areas such as dirt, grass, asphalt; 2) obstacles—bushes, trees, objects, and poles. Since there are very few dirt labels, as seen in the dataset's label distribution in **Figure 7**, this label is excluded from the study.

Figure 6. Rellis-3D image example and ontology [32].

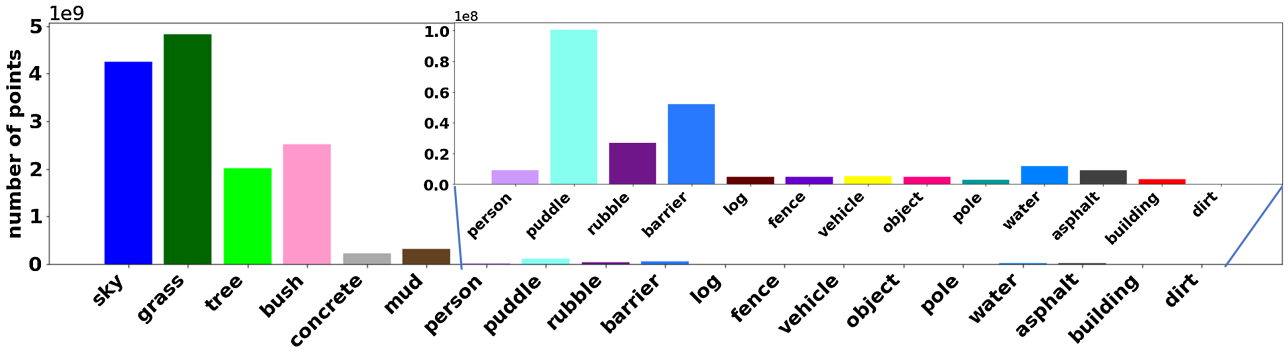

Figure 7. Rellis-3D data distribution.

2.4.2. CaT: CAVS Traversability Dataset

The Center for Advanced Vehicular Systems (CAVS) Traversability dataset (CaT) was created to explore off-road terrain in environments containing obstacles, ditches, and hidden objects [33]. The dataset includes 3624 labeled RGB images of varying high-definition sizes. The terrain in the images is segmented to show the traversing ability of three different-sized vehicles: a sedan, a pickup, and a sizeable off-road vehicle. A sedan is considered the vehicle with the least traversability and

the off-road vehicle the most. Figure 8 shows example images and annotations from the dataset. As shown in Figure 9, the CaT dataset has a class distribution with 25.29% of the pixels representing the driving capabilities of a sedan, 14.69% for a pickup, and 15.17% for an off-road vehicle. The last 44.86% are background pixels or untraversable terrain.

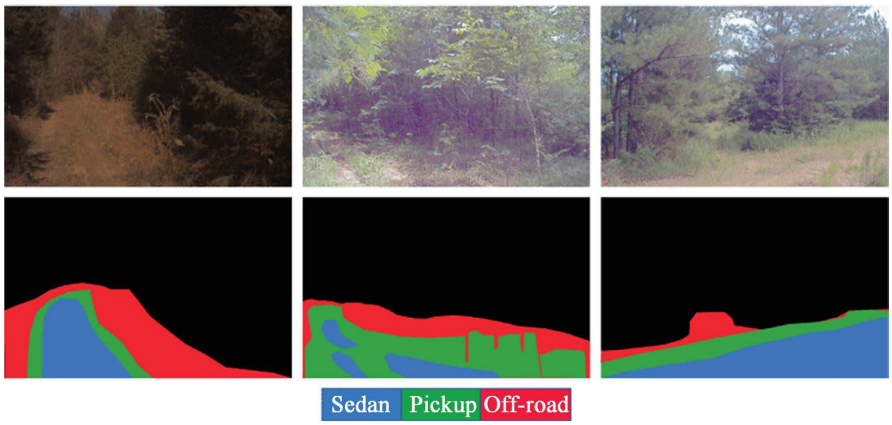

Figure 8. CaT Image Examples and Corresponding Traversability Labels [33].

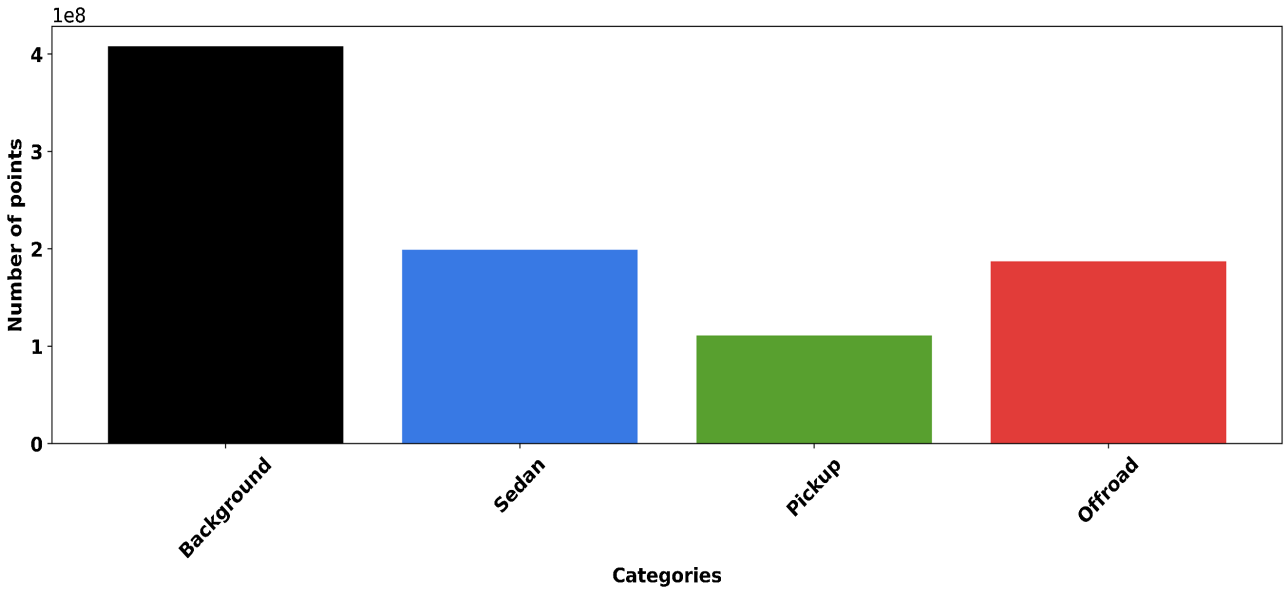

Figure 9. CaT data distribution.

3. Methods

To determine if ViTs improve a UGV perception system, two different ViT architectures for semantic segmentation are evaluated on the Rellis-3D off-road dataset. ViT architectures are evaluated based on their accuracy, ability to identify traversable terrain and inference speed. Additionally, the inference memory usage and architecture memory size for each architecture are compared. The model must be suited for real-time use and is further explored on the CaT dataset. Results for the ViT architectures are compared to previous studies of CNN-based architectures

on the Rellis-3D dataset [28] [30]. All hardware and software used for testing are listed in the Appendix.

3.1. Model Training

The Segformer and EfficientViT architectures are implemented in Python using PyTorch. To update the weights of the neural networks in both architectures, the AdamW optimization algorithm is used with default parameter values for the weight decay, epsilon, and beta parameters [34]. Both models were trained until convergence. Other specific hyperparameters for each architecture are documented in their respective papers and detailed below [19] [20].

3.1.1. Segformer Training Parameters

To train the Segformer architecture, an initial learning rate of 0.00006 is used with a polynomial learning rate scheduler, as documented in the original paper [20]. Random flipping and random cropping were used for pre-processing the images as documented in the original paper [20].

3.1.2. EfficientViT Training Parameters

For the EfficientViT architecture, training began with an initial 20 epochs of warm-up training. In the warm-up epochs, the learning rate gradually increased from 0.0 to the base learning rate of 0.001. The learning rate was adjusted throughout the training based on a cosine learning rate scheduler [35]. Random flipping, random cropping, hue changing, and random erasing of image data were used for pre-processing [36].

3.2. Evaluation Metrics

The CNN and ViT architectures were evaluated based on their ability to recognize and generalize patterns in an off-road setting (segmentation accuracy) and their ability to do so efficiently (inference speed and memory usage).

Segmentation Accuracy

The primary accuracy measurement in segmentation is intersection over union (IoU), shown in Equation (1). The intersection and union are based on the true positive (TP), false positive (FP), and false negative (FN) predictions of each class. The mean IoU (mIoU), is an average of all the individual class IoU scores (see Equation (2)). For exploring Rellis-3D, the architectures are trained on 70% of the dataset (4364 images) and evaluated on 30% of the image data (1870 images), which is the same as previous studies. For exploring CaT, 70% of the dataset (2356 images) was used for training, and 30% (1088 images) was used for testing.

$$\mathrm{IoU}_{\mathrm{class}} = \frac{\mathrm{Prediction}_{\mathrm{class}} \bigcap \mathrm{GroundTruth}_{\mathrm{class}}}{\mathrm{Prediction}_{\mathrm{class}} \bigcup \mathrm{GroundTruth}_{\mathrm{class}}} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}+\mathrm{FN}} \tag{1}$$

$$\mathrm{IoU} = \frac{\sum \mathrm{IoU}_{\mathrm{class}}}{n_{\mathrm{classes}}} \tag{2}$$

3.3. Inference Speed and Memory Usage

The timing approach for evaluating inference speeds and memory consumption is detailed in Listing 1. The architectures were timed on 200 iterations of predicting segmented images on Rellis-3D resolution data (1200 × 1920), and the average results were reported. Since perception systems must transfer knowledge to the CPU for decision-making, the inference speed calculations included the time to transfer the predictions back to the CPU.

```
import torch
from torch.nn import functional as F

n_trials = 200
start_event = torch.cuda.Event(enable_timing=True)
end_event = torch.cuda.Event(enable_timing=True)
model.eval().cuda()
torch.cuda.synchronize()
image_data.shape # [1, 3, 1200, 1920] Rellis3D resolution

durations = []
memory_stats = []
with torch.inference_mode():
    for _ in range(n_trials):

        start_event.record() # start timing

        outputs = model(image_data)            # inference
        _, predictions = torch.max(outputs, 1) # condense class predictions
        predictions = predictions.byte().cpu() # CPU transfer

        end_event.record()   # end timing

        durations.append(start_event.elapsed_time(end_event))

        peak_memory = torch.cuda.max_memory_allocated() # measuring inference memory
        memory_stats.append(peak_memory)


avg_inf_time = sum(durations) / len(durations)
average_memory = sum(memory_stats) / len(memory_stats)
```

Listing 1. Inference Speed and Memory Data Collection in PyTorch.

4. Results and Discussion

This section presents new results for inference speed and memory consumption, as well as mIoU on the Rellis-3D dataset. Additionally, the number of parameters and size in memory of each architecture are detailed. We then measure the inference time of the most accurate architectures on a Jetson Xavier AGX edge device to ensure real-time viability. Further, using the architecture deemed most suited for accurate and fast inference, the CaT dataset was explored. We compare CaT results against the benchmark IoU scores outlined in the CaT dataset paper [33].

4.1. Rellis-3D Accuracy

First, the CNN and ViT architectures are evaluated on the Rellis-3D dataset and compared for accuracy in the off-road setting. The class and mIoU results are shown in Table 1.

Table 1. Class and Mean IoU Accuracy (%) on Rellis-3D.

Class	DeeplabV3+ [28]	Swiftnet [30]	EfficientViT	Segformer
grass	72.70	91.83	92.07	85.65
tree	83.45	90.04	90.06	82.05
pole	7.57	42.15	40.53	14.48
water	53.35	81.48	79.22	50.63
sky	95.84	97.54	97.57	96.28
vehicle	26.96	67.30	65.34	31.02
object	24.89	72.73	68.44	13.32
asphalt	60.95	86.08	85.34	58.40
building	10.49	65.08	59.46	9.49
log	25.97	61.79	56.90	36.12
person	66.46	92.52	90.78	71.23
fence	15.79	65.61	58.31	18.88
bush	70.95	85.09	85.55	73.18
concrete	80.23	91.24	90.96	84.83
barrier	65.57	87.63	86.19	68.37
puddle	59.27	80.96	80.69	67.08
mud	29.51	65.46	66.09	45.37
rubble	36.43	77.87	74.96	49.88
mIoU	49.24	77.9	76.03	53.13

EfficientViT and Swiftnet were the strongest performing architectures with 76.03% and 77.9% mIoU, respectively. The class IoUs for these architectures show that each is well generalized to large terrain patterns and small obstacles/objects.

Results from the previous study show that DeepLabV3+ could generalize to the large terrain patterns—e.g., tree, grass, sky, bush and concrete, while it struggled to generalize to the smaller objects/obstacles. Similarly, while struggling with the smaller objects, Segformer generalizes well for significant patterns in the dataset—e.g., grass, bush, concrete, sky, and trees. Both architectures seem to be affected by the class imbalance challenge common in off-road datasets, with sky, grass, tree, and bush being the most over-represented classes in the Rellis-3D dataset, as previously shown in **Figure 7**. Prediction segmentation results with the ViT architectures compared to the ground truth segmentation are shown in **Figure 10**. The traversable tracks of **Figure 10(d)** show a mixture of classes, highlighting Segformer's inaccuracy on small patterns. As seen in **Figure 10(c)**, EfficientViT smoothly identifies traversability patterns in the off-road environment, barely deviating from the ground truth segmentation in **Figure 10(b)**.

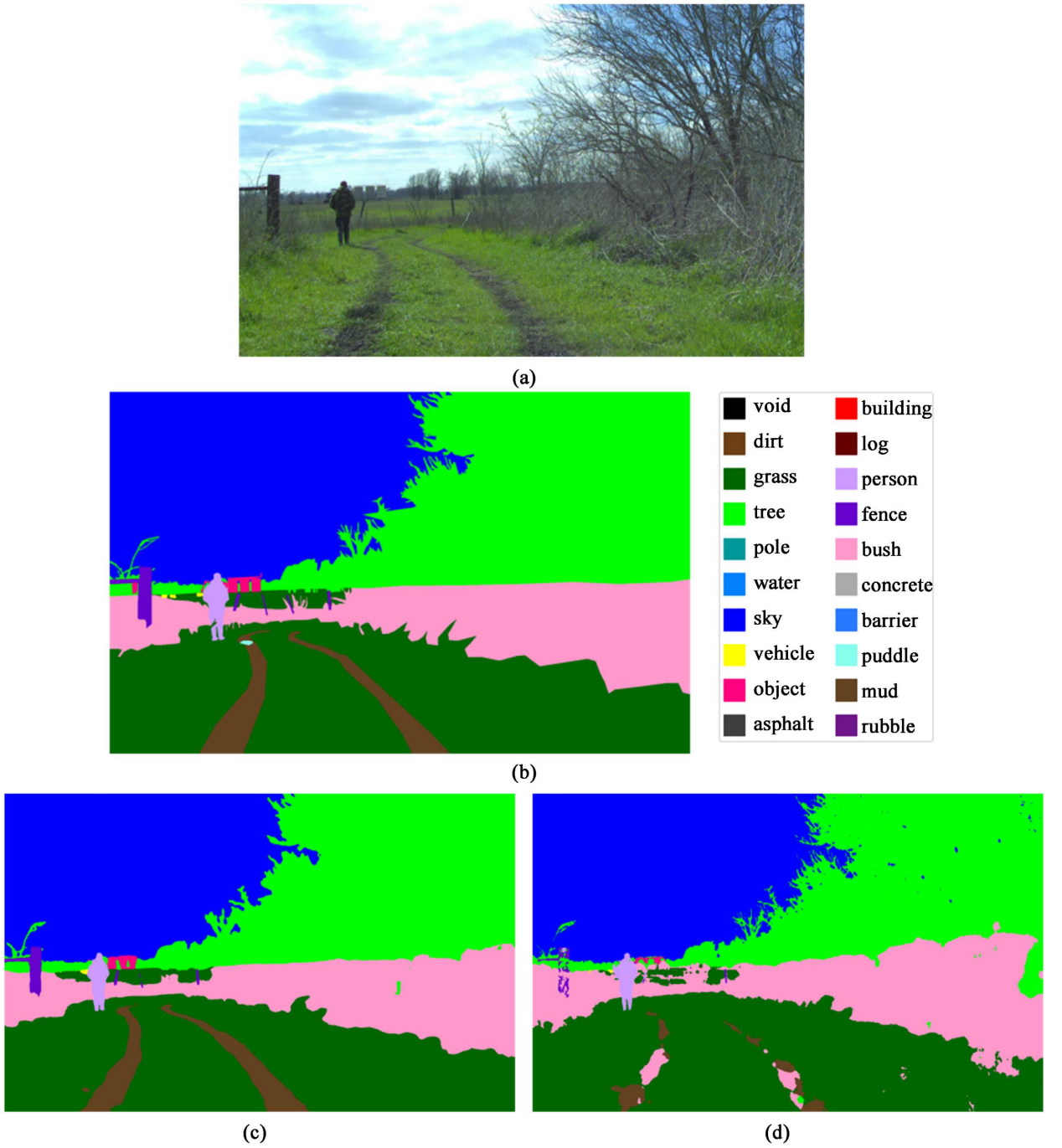

Figure 10. Rellis3D Example Image and ViT Predictions. (a) RGB Image; (b) Ground Truth Segmentation; (c) EfficientViT Predicted Segmentation; (d) Segformer Predicted Segmentation.

4.2. Rellis-3D Inference Speed and Memory Usage

Based on the accuracy results, Swiftnet and the ViT architectures have results promising for real-world use. To determine the viability of each architecture for implementation on an edge device, a baseline comparison of inference speed, inference memory usage, parameters, and architecture size, these architectures are studied using a large GPU.

4.2.1. Large GPU Study

All inference results presented in Table 2 were measured using an NVIDIA V100 GPU (specifications shown in Appendix Table A1).

Table 2. Architecture Inference Speed and Memory Usage on Rellis-3D.

Architecture	Inference Speed	Parameters	Architecture Size	Inference Memory Usage
EfficientViT	11.53 ms	0.7 M	2.76 MB	392 MB
Swiftnet	23.32 ms	12 M	46.14 MB	746 MB
Segformer	87.86 ms	3.7 M	14.22 MB	2571 MB

EfficientViT outperformed the other state-of-the-art architectures in terms of inference speed. It is about twice as fast as the CNN-based Swiftnet on a V100 while using fewer parameters and half as much memory for inference as Swiftnet. Segformer suffers from a slower inference speed and a significant increase in memory consumption for inference, likely due to the inefficient attention functionality, a notable downside of self-attention with high-resolution images.

4.2.2. Edge Device Study

Based on the results from the large GPU study, Swiftnet and EfficientViT are viable for edge device use given their fast inference speed and low memory usage. Only the Swiftnet and EfficientViT architectures were translated to run on the smaller edge device since the results of Table 2 show that the Segformer inference time was significantly slower than the other two architectures, even with a powerful GPU like a V100. To verify that Swiftnet and EfficientViT maintain their inference speed in a real-time setting, the architectures were tested on a NVIDIA Jetson Xavier AGX edge device (specifications shown in Appendix Table A2). After testing these two architectures on the Xavier with the same method from Algorithm 1, the NVIDIA TensorRT engine [37] was used to optimize the architectures for inference on the Xavier. EfficientViT strongly outperforms Swiftnet regarding inference speed on the edge device as shown in Table 3. Without TensorRT, it is more than 3× faster; with TensorRT, it is about 4× faster. Based on these results, EfficientViT has the traits most desirable for real-world performance: strong segmentation accuracy substainally faster inference speed than the other architectures studied.

4.3. CaT Dataset Results

Since the results form Rellis-3D show EfficientViT is the most viable architecture

Table 3. Inference speed on jetson xavier AGX edge device.

Architecture	No Optimization	TensorRT Optimized
EfficientViT	114 ms	83 ms
Swiftnet	388 ms	321 ms

Table 4. IoU (%) results on CaT compared to state of the art benchmark [33].

Classes / Architectures	Sedan	Pickup	Off-Road	mIoU
PSPNet w/ResNet-18	90.44	66.62	79.71	78.92
PSPNet w/ResNet-34	91.21	68.64	80.52	80.12
PSPNet w/ResNet-50	90.70	67.40	80.00	79.36
PSPNet w/ResNet-101	91.64	69.08	81.00	80.57
EfficientViT	**98.22**	**92.01**	**93.09**	**94.44**

for real-world use, with high inference speed and accuracy, we compare it to current results with the CaT dataset to further test the ability of the architecture to determine traversable terrain in a different off-road setting. The results for training EfficientViT on the CaT dataset are shown in Table 4. When comparing these results to the state-of-the-art CaT Benchmark, EfficientViT detects the three types of traversable terrain in the off-road environment more accurately. Comparing our results to the state-of-the-art benchmark from the CaT dataset [33], these results achieved a mIoU score of 94.44% a significant increase in mIoU 13.87% over the CaT benchmark of 80.57%. Individually, these results show an improved IoU score for sedan traversability by 6.57%, pickup by 22.93% and off-road by 12.09%. Traversability accuracy with both CaT and Rellis-3D coupled with high inference speed results prove EfficientViT is extremely viable for real-world use in determining traversable terrain in a perception system.

5. Conclusion

Using a state-of-the-art ViT architecture, EfficientViT, we were able to demonstrate the viability of a ViT architecture for us in an off-road perception system. Compared to previous results with a CNN architecture, Swiftnet, EfficientViT maintained a strong accuracy in off-road environments while having a much faster inference speed. EfficientViT has 1.9% mIoU reduction on the Rellis3D dataset compared to Swiftnet, while being 2× as fast as the Swiftnet for inference on a large GPU, and up to 4× as fast on an edge device with TensorRT optimization. Additionally, EfficientViT uses half as much memory for inference as Swiftnet and has a 20× smaller model size—two traits extremely desirable in real-world systems with limited memory capacity. The use of hardware efficient attention and efficient convolution operations makes this architecture extremely fast, while maintaining a strong accuracy with few parameters. These results make EfficientViT a viable option for real-time use in UGV perception systems.

EfficientViT also demonstrated new state-of-the-art results on the CaT dataset with 94.44% mIoU on traversable terrain. These results further demonstrate the ability of the EfficientViT architecture to determine traversable terrain for a UGV, maintaining high accuracy, fast inference, and low memory usage.

To add to current developments toward integrating higher levels of autonomy

into UGVs, this research provides insights into new methods for improving off-road perception systems. Use of new semantic segmentation architectures that maintain accuracy, with a lower memory footprint and higher inference speed, will alleviate latency and memory bottlenecks within the perception system, allowing vehicles to make safe decisions in real-time.

Future Work

Perception systems may deploy a variety of sensors including RADAR, LiDAR, FLIR, multispectral and stereo images. Combinations and fusions of these sensor modalities can lead to a richer understanding of the surrounding environment, for example providing depth/distances for contextual information. In future work, the use and adaptation of ViT architectures with these additional sensor modalities for enriched perception will be explored.

With power and physical space restrictions common on autonomous vehicles, data transfer can be utilized to send perception data to external devices for increased computation demands. Offloading data for processing can introduce new challenges where restricted bandwidth of the transfer requires data manipulation to maintain high processing speeds and reduce latency.

Acknowledgements

DISTRIBUTION STATEMENT A. approved for public release; distribution is unlimited. OPSEC#8920.

This work was supported by the Virtual Prototyping of Autonomy Enabled Ground Systems (VIPR-GS), a US Army Center of Excellence for modeling and simulation of ground vehicles, under Cooperative Agreement W56HZV-21-2-0001 with the US Army DEVCOM Ground Vehicle Systems Center (GVSC).

This research was also supported by the U.S. National Science Foundation under Grants SHF-1910197, SHF-1943114 and CCF-2312616.

Clemson University is acknowledged for their generous allotment of compute time on the Palmetto Cluster.

Clemson Future Computing Technologies Laboratory summer research students Michael Ellis, Adam Niemczura, Precious Eyabi and Ryan Chen are acknowledged for their contributions to this study.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Akhil, K.P., Manikutty, G., Ravindran, R. and Rao, R.B. (2019) Autonomous Navigation of an Unmanned Ground Vehicle for Soil Pollution Monitoring. 2019 2*nd International Conference on Intelligent Computing, Instrumentation and Control Technologies*, Kannur, 5-6 July 2019, 1563-1567. https://doi.org/10.1109/icicict46008.2019.8993292

[2] Long, L.N., Hanford, S.D., Janrathitikarn, O., Sinsley, G.L. and Miller, J.A. (2007) A

Review of Intelligent Systems Software for Autonomous Vehicles. 2007 *IEEE Symposium on Computational Intelligence in Security and Defense Applications*, Honolulu, 1-5 April 2007, 69-76. https://doi.org/10.1109/cisda.2007.368137

[3] Islam, F., Nabi, M.M. and Ball, J.E. (2022) Off-Road Detection Analysis for Autonomous Ground Vehicles: A Review. *Sensors*, **22**, Article 8463. https://doi.org/10.3390/s22218463

[4] Rosique, F., Navarro, P.J., Fernández, C. and Padilla, A. (2019) A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research. *Sensors*, **19**, Article 648. https://doi.org/10.3390/s19030648

[5] Pavitha, P.P., Rekha, K.B. and Safinaz, S. (2021) Perception System in Autonomous Vehicle: A Study on Contemporary and Forthcoming Technologies for Object Detection in Autonomous Vehicles. 2021 *International Conference on Forensics*, *Analytics*, *Big Data*, *Security*, Bengaluru, 21-22 December 2021, 1-6. https://doi.org/10.1109/fabs52071.2021.9702569

[6] Redmon, J. and Farhadi, A. (2018) Yolov3: An Incremental Improvement.

[7] Chen, L.-C., Papandreou, G., Schroff, F. and Adam, H. (2017) Rethinking Atrous Convolution for Semantic Image Segmentation.

[8] Ketkar, N. and Moolayil, J. (2021) Convolutional Neural Networks. In: Ketkar, N. and Moolayil, J., Eds., *Deep Learning with Python*, Apress, 197-242. https://doi.org/10.1007/978-1-4842-5364-9_6

[9] Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2017) Imagenet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*, **60**, 84-90. https://doi.org/10.1145/3065386

[10] Simonyan, K. and Zisserman, A. (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition.

[11] He, K., Zhang, X., Ren, S. and Sun, J. (2016) Deep Residual Learning for Image Recognition. 2016 *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, 27-30 June 2016, 770-778. https://doi.org/10.1109/cvpr.2016.90

[12] Khan, S.H., Naseer, M., Hayat, M., Zamir, S.W., Khan, F.S. and Shah, M. (2021) Transformers in Vision: A Survey.

[13] Dosovitskiy, A., Beyer, L., Kolesnikov, A., *et al.* (2020) An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

[14] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., *et al.* (2016) The Cityscapes Dataset for Semantic Urban Scene Understanding. 2016 *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, 27-30 June 2016, 3213-3223. https://doi.org/10.1109/cvpr.2016.350

[15] Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A. and Torralba, A. (2017) Scene Parsing through ADE20K Dataset. 2017 *IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, 21-26 July 2017, 5122-5130. https://doi.org/10.1109/cvpr.2017.544

[16] Szabó, L. and Weltsch, Z. (2024) A Comprehensive Review of Existing Datasets for Off-Road Autonomous Vehicles. 2024 *IEEE 22nd World Symposium on Applied Machine Intelligence and Informatics*, Stará Lesná, 25-27 January 2024, 403-410. https://doi.org/10.1109/sami60510.2024.10432820

[17] Chen, L., Zhu, Y., Papandreou, G., Schroff, F. and Adam, H. (2018) Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In: Ferrari, V., Hebert, M., Sminchisescu, C. and Weiss, Y., Eds., *Computer Vision—ECCV* 2018, Springer, 833-851. https://doi.org/10.1007/978-3-030-01234-2_49

[18] Oršić, M. and Šegvić, S. (2021) Efficient Semantic Segmentation with Pyramidal Fusion. *Pattern Recognition*, **110**, Article 107611. https://doi.org/10.1016/j.patcog.2020.107611

[19] Cai, H., Li, J., Hu, M., Gan, C. and Han, S. (2023) Efficientvit: Lightweight Multi-Scale Attention for High-Resolution Dense Prediction. 2023 *IEEE/CVF International Conference on Computer Vision*, Paris, 1-6 October 2023, 17256-17267. https://doi.org/10.1109/iccv51070.2023.01587

[20] Xie, E., Wang, W., Yu, Z., Anandkumar, A., *et al.* (2021) Segformer: Simple and Efficient Design for Semantic Segmentation with Transformers.

[21] Hofmarcher, M., Unterthiner, T., Arjona-Medina, J., Klambauer, G., Hochreiter, S. and Nessler, B. (2019) Visual Scene Understanding for Autonomous Driving Using Semantic Segmentation. In: Samek, W., Montavon, G., Vedaldi, A., Hansen, L.K. and Müller, K.R., Eds., *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, Springer, 285-296. https://doi.org/10.1007/978-3-030-28954-6_15

[22] Ma, Y., Wang, Z., Yang, H. and Yang, L. (2020) Artificial Intelligence Applications in the Development of Autonomous Vehicles: A Survey. *IEEE/CAA Journal of Automatica Sinica*, **7**, 315-329. https://doi.org/10.1109/jas.2020.1003021

[23] Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., Martinez-Gonzalez, P. and Garcia-Rodriguez, J. (2018) A Survey on Deep Learning Techniques for Image and Video Semantic Segmentation. *Applied Soft Computing*, **70**, 41-65. https://doi.org/10.1016/j.asoc.2018.05.018

[24] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016) You Only Look Once: Unified, Real-Time Object Detection. 2016 *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, 27-30 June 2016, 779-788. https://doi.org/10.1109/cvpr.2016.91

[25] Howard, A.G., Zhu, M., Chen, B., *et al.* (2017) MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.

[26] Vaswani, A., Shazeer, N., Parmar, N., *et al.* (2017) Attention Is All You Need.

[27] Tay, Y., Dehghani, M., Bahri, D. and Metzler, D. (2020) Efficient Transformers: A Survey.

[28] Faykus, M.H., Selee, B. and Smith, M. (2023) Utilizing Neural Networks for Semantic Segmentation on RGB/LIDAR Fused Data for Off-Road Autonomous Military Vehicle Perception. https://doi.org/10.4271/2023-01-0740

[29] Chen, L., Zhu, Y., Papandreou, G., Schroff, F. and Adam, H. (2018) Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In: Ferrari, V., Hebert, M., Sminchisescu, C. and Weiss, Y., Eds., *Computer Vision—ECCV* 2018, Springer, 833-851. https://doi.org/10.1007/978-3-030-01234-2_49

[30] Selee, B., Faykus, M. and Smith, M. (2023) Semantic Segmentation with High Inference Speed in Off-Road Environments. https://doi.org/10.4271/2023-01-0868

[31] Russakovsky, O., Deng, J., Su, H., Krause, J., *et al.* (2014) Imagenet Large Scale Visual Recognition Challenge.

[32] Jiang, P., Osteen, P., Wigness, M. and Saripalli, S. (2021) RELLIS-3D Dataset: Data, Benchmarks and Analysis. 2021 *IEEE International Conference on Robotics and Automation*, Xi'an, 30 May-5 June 2021, 1110-1116. https://doi.org/10.1109/icra48506.2021.9561251

[33] Sharma, S., Dabbiru, L., Hannis, T., Mason, G., Carruth, D.W., Doude, M., *et al.* (2022) Cat: CAVS Traversability Dataset for Off-Road Autonomous Driving. *IEEE Access*, **10**, 24759-24768. https://doi.org/10.1109/access.2022.3154419

[34] Loshchilov, I. and Hutter, F. (2017) Fixing Weight Decay Regularization in Adam.

[35] Loshchilov, I. and Hutter, F. (2017) SGDR: Stochastic Gradient Descent with Warm Restarts. *International Conference on Learning Representations*, Toulon, 24-26 April 2017.

[36] Yang, S., Xiao, W., Zhang, M., Guo, S., Zhao, J. and Shen, F. (2023) Image Data Augmentation for Deep Learning: A Survey.

[37] Zhou, Y. and Yang, K. (2022) Exploring Tensorrt to Improve Real-Time Inference for Deep Learning. 2022 *IEEE* 24*th Int Conf on High Performance Computing & Communications*; 8*th Int Conf on Data Science & Systems*; 20*th Int Conf on Smart City*; 8*th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application* (*HPCC/DSS/SmartCity/DependSys*), Hainan, 18-20 December 2022, 2011-2018.

Appendix

Table A1. V100 inference testing specifications.

GPU Name	NVIDIA Tesla V100
Power Cap	250 W
CUDA Cores	5120
GPU Memory	16 GB (GPU dedicated)
CUDA Version	12.4
Python Version	3.11.4
PyTorch Version	2.1.0
Torchvision Version	0.16.0

Table A2. Jetson Xavier testing specifications.

Device Name	Jetson Xavier AGX
Power Cap	15 W
CUDA Cores	512
GPU Memory	32 GB (shared)
CUDA Version	11.8
Python Version	3.8.0
PyTorch Version	2.0.0
Torchvision Version	0.15.0