Improving Inference Speed of Perception Systems in Autonomous Unmanned Ground Vehicles

Bradley Selee
bselee@clemson.edu

Improving Inference Speed of Perception Systems in Autonomous Unmanned Ground Vehicles

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Engineering

---

by
Bradley Robert Selee
May 2023

Abstract

Autonomous vehicle (AV) development has become one of the largest research challenges in businesses and research institutions. While much research has been done, autonomous driving still requires extensive amounts of research due to its immense, multi-factorial difficulty. Autonomous vehicles rely on many complex systems to function, make accurate decisions, and, above all, provide maximum safety. One of the most crucial components of autonomous driving is the perception system.

The perception system allows the vehicle to identify its surroundings and make accurate, but safe, decisions through the use of computer vision techniques like object detection, image segmentation, and path planning. Due to the recent advances in deep learning, state-of-the-art computer vision algorithms have made exceptional progress. However, for production-ready autonomous driving, these algorithms must be nearly perfect. Furthermore, even though perception systems are a widely researched area, most research focuses on urban environments and there exists a great need for autonomy in other areas. Specifically, autonomy for unmanned ground vehicles (UGV) needs to be explored. Autonomous UGVs allow for a wide range of applications like military usage, extreme climate exploration, and rescue missions.

This research aims to investigate bottlenecks within a perception system of autonomous UGVs and methods of improving them. The primary investigation focuses on the inference speed of semantic segmentation using deep learning techniques. Unlike object detection, semantic segmentation provides a much better understanding of the environment by providing pixel-wise classification rather than only creating bounding boxes. However, semantic segmentation comes at a much higher computational cost. Secondly, this thesis looks at increasing the image transfer time from a mounted camera to a video processing unit (which serves the deep learning model) using lossy compression. Due to the nature of lossy compression, we must also understand how lossy compression affects the classification accuracy of semantic segmentation. Finally, the challenges faced and preliminary results from future work relating to temporal consistency are discussed.

Dedication

This thesis is dedicated to my parents for their constant support during the physical and mental battles of the COVID-19 pandemic. This thesis is also dedicated to everyone link who endured similar mental battles during the pandemic. Lastly, this thesis is dedicated to Dr. Smith for giving me this opportunity and setting me up for success upon graduation.

Acknowledgments

First, I would like to acknowledge Dr. Melissa C. Smith for establishing the undergraduate Machine Learning and Big Data Creative Inquiry. This truly helped me find my passion, strengthened my motivation for continuous learning, and my decision to pursue a master's degree.

Next, I would like to acknowledge Dr. Jon C. Calhoun who I have known since my junior year of undergrad. He has always offered immense amounts of support in my undergrad and graduate years, even when dealing with many other commitments. Dr. Calhoun is dedicated to ensuring his students learn and will go above what is required while creating a friendly atmosphere.

I would also like to acknowledge Dr. Yongkai Wu. Dr. Wu helped transition my research from TensorFlow to PyTorch from the knowledge and practice gained in his class. Some of the models built and experiments run throughout this research would have been made much harder if I had not switched to PyTorch.

Lastly, I would like to acknowledge Dr. Harlan Russell. He has offered an incredible amount of support in my undergrad years as my advisor and met with me countless times to discuss graduate school and other options. He has also been very flexible and helpful during my transition to graduate school.

This work was supported by Clemson University's Virtual Prototyping of Autonomy Enabled Ground Systems (VIPR-GS), under Cooperative Agreement W56HZV-21-2-0001 with the US Army DEVCOM Ground Vehicle Systems Center (GVSC).

Part of this research was also supported by the U.S. National Science Foundation under Grants SHF-1910197 and SHF-1943114.

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. OPSEC #7385

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Research in self-driving cars has made great strides, primarily due to the rise of deep learning (DL) techniques [28]. With this immense progress in artificial intelligence (AI), state-of-the-art (SOTA) computer vision algorithms have greatly favored DL rather than standard, non-neural network computer vision algorithms. These recent improvements in DL have enabled the advancements made in autonomous vehicles. Current advanced driver assistance systems (ADAS) support low-level autonomy like lane assists and blind spot alerting [33]. These ADAS operate at what is considered level 1 and level 2 autonomy because drivers are still required to perform some actions. However, research is now focused on advancing high levels of autonomy: level 3 (conditional automation), level 4 (high automation), and, ultimately, level 5 (full automation) [53]. To one-day support full automation, several autonomy systems must be improved, most notably the perception systems.

The perception system in autonomous vehicles remains one of the most critical systems. These systems include many components such as a wide range of sensors, a complex collection of vision algorithms, and an intricate hardware setup while simultaneously communicating with other systems in the autonomous vehicle. All of these components could create bottlenecks and unnecessary overhead within the perception system, ultimately delaying the time until the vehicle can make an accurate yet safe decision. Reducing the overhead in certain components can alleviate this delay.

Perception algorithms play a vital role in these autonomous systems because they must quickly understand the vehicle's surroundings in order to make accurate decisions. Two major types of image classification algorithms exist in autonomous vehicle systems: object detection [43] and semantic image segmentation [11]. Object detection can be defined as classifying objects of interest and identifying their locations in an image using bounding boxes [68], while semantic segmentation classifies every pixel in an image into a defined set of class labels [32, 61]. Compared to object detection, semantic segmentation provides a much better understanding of the environment, but at a significant computational cost [23, 22]. Specifically, a trade-off between high segmentation accuracy and high inference speed occurs in deep neural networks (DNN) [22, 33, 18]. This creates an issue because AVs must make decisions exceptionally quickly but be accurate enough to make the correct decision. Most modern semantic segmentation algorithms utilize DNNs and many DNNs adopt the encoder-decoder structure [11, 35]. The encoder consists of a feature extractor, like ResNet [22], which uses convolutional layers [2] to downsample the original image into a feature map. After extracting meaningful features, the decoder upsamples the feature map back to its original image size, creating a semantic map of class probabilities [61].

Typically, more accurate semantic segmentation models require longer inference time. Likewise, models with faster inference speed suffer in accuracy loss [67]. This makes perception a challenge in autonomous vehicles. A vision system must have acceptable accuracy in order to make correct decisions but at the same time needs to make these decisions in real-time [33]. In recent years, research has made great progress in developing DNN architectures that speed up semantic segmentation inference time while maintaining an acceptable level of accuracy [67, 38, 65]. Many of these fast inference models were designed and trained for urban environments, most notably for use in self-driving cars [5]. Additionally, the primary benchmark dataset used for training and evaluation was the Cityscapes dataset [13], which contains thousands of semantically labeled images (2048×1024) from urban environments.

Unfortunately, there is a lack of research evaluating semantic segmentation in the off-road setting; this can be seen from the absence of research investigating this domain and the scarcity of labeled off-road datasets [64, 25]. Autonomous off-road driving covers a wide range of applications like exploration, rescue, and military use. Just like self-driving cars, perception systems are critical for autonomy in unmanned ground vehicles (UGV) and other off-road vehicles [57]. The evaluation of semantic segmentation DNNs in off-road environments is a necessity to integrate higher levels of autonomy in UGVs. Furthermore, there are two key features that can make off-road semantic segmentation more challenging than urban settings [54]. Firstly, the environments are unstructured by nature, which makes identifying traversable areas more difficult. Next, the environments are very noisy due to the type of objects. This can lead to poor generalization of the deep learning model. On the other hand, due to the application of most UGVs, it may be acceptable for semantic segmentation to be less accurate in off-road environments. Because UGVs are typically very large and durable, they can drive over smaller objects. Therefore, detecting obstacles like logs and bushes can be less important, while focusing on faster inference speed could be considered more important. Even though off-road and urban environments contain vastly different terrain, many of the same latency bottlenecks occur in the perception system. Real-time perception systems contain other bottlenecks too and are not only limited by their DNN.

In an autonomous vehicle, a camera must transmit the video feed to an edge device for image processing, as seen in Figure 4.4. Within this transfer, some latency exists [14]. Compressing the images from the camera feed reduces the latency between this data transfer [41, 42]. Autonomous vehicles generate up to 40 TB of data during daily usage [62]. High-resolution cameras and sensors can generate about 4 TB+ of data per day [1]. Without compression, processing and storing this data would require extensive hardware. Compression has been consistently used in solutions to many data problems, therefore, it is logical that compression can improve various overheads in perception systems.

This research focuses on improving the inference speed of perception systems in autonomous UGVs. Therefore, most of the evaluation is performed in off-road terrain; this is also where the novelty lies. First, SwiftNet [38], a semantic segmentation deep learning model designed for high inference speed and accuracy on high-resolution images, is leveraged. The model is then adapted to train and evaluate on the Rellis-3D dataset [25, 18]. Rellis-3D is an extensive off-road dataset designed for semantic segmentation, containing 6234 labeled 1920×1200 images. Next, reducing local image data transfer latency using JPEG compression is looked at. Compression reduces the amount of data over a local network. This size reduction minimizes the latency of local transfer for the images. In a real-time scenario, the system transfers more images and reduces the overall latency. Due to natural data loss from lossy compression, an understanding of the effects of training and evaluating semantic segmentation models on compressed images must be obtained. Finally, the challenges faced during real-life deployment and propose a method to increase temporal consistency are discussed.

While a majority of this work is application based, I believe this work provides the following contributions to the literature [48, 17]:

- Evaluation of a high inference speed semantic segmentation model, SwiftNet for use in off-road unmanned ground vehicles
- Replication and verification of SwiftNet's accuracy and inference speed on the Cityscapes benchmark dataset and Rellis-3D, a new off-road dataset
- Exploration of lossy compression to reduce image transfer latency in real-time perception systems.
- Evaluation of semantic segmentation on compressed images at different compression ratios and a method to increase the mIoU of the compressed images.
- Offering a unique and complex perspective by considering the perception of autonomous vehicles in off-road terrain rather than urban environments.

Chapter 2

Background

For the purpose of this thesis, we assume the audience has a foundational understanding of machine learning (ML) concepts, as well as a basic understanding of deep learning.

The main component of this work is computer vision. Due to the major successes in deep learning, standard non-neural network computer vision algorithms are not covered. Instead, this section discusses foundational components and major vision tasks related to computer vision using deep learning. Next, we discuss major benchmark datasets related to semantic segmentation along with less-known datasets crucial to this research. After, we compare structured terrain with unstructured terrain and explain the difficulties with the latter. Then, we provide a brief background of lossy data compression, which we use in the second part of our research.

2.1 Computer Vision Using Deep Learning

Computer vision allows machines to gain visual knowledge similar to what humans see. While machine learning has existed for a long time now [28], up until recently rule-based algorithms dominated computer vision techniques. For example, assume the goal is to locate colorful balls on a white tile floor in an image. One simple, non-machine learning, approach would be to convert the image to the hue saturation value (HSV) color space, then create and adjust a threshold of the hue values until you have a binary image segmenting the desired ball. This works well because the HSV color space only has one channel, hue, which describes the color feature of the balls, and one threshold is much easier to find compared to three thresholds in the RGB color space [7]. This standard rule-based algorithm works very well for very simple tasks. Now, assume the same task as described above, however, instead of a white tile floor the colorful balls are scattered in the forest. Segmenting these balls from the background using the non-ML approach we just described becomes a lot more difficult. We can extend this problem to a real-world example, segmenting a myriad of obstacles and terrain in an autonomous vehicle driving at high speeds. Many rule-based vision algorithms will fail this task quickly [58], however, using deep learning approaches we can approximate functions using the visual features within images to segment the labels of interest with much greater success. Most deep learning methods for computer vision follow a similar architecture [58]. First, semantic features are extracted from images utilizing convolutional layers, then the architecture is modified for the desired task: image classification, object detection, or image segmentation. While there exist several other vision tasks, this section will only focus on a few. Furthermore, this work specifically focuses on the task of semantic image segmentation so it will be covered much more heavily than image classification and object detection.

2.1.1 Feature Extraction

In computer vision tasks, feature extraction is one of the foundational components of any deep neural network (DNN) with convolutional layers [2]. DNNs learn the important features within an image through a series of convolutional and pooling operations, creating rich feature maps [58]. Convolutional layers use various kernels to extract information from neighboring pixels, capturing spatial context. Similarly, the pooling slides a small window across the image and extracts additional spatial features. The method of extraction depends on the type of pooling. Max pooling locates the maximum pixel value in a $n \times n$ window, while average pooling averages all the pixels in a $n \times n$ window. Often, downsampling of the feature map occurs during pooling; many times, padding is applied during convolution to prevent downsampling [58].

2.1.2 Image Classification

A complex combination of many convolutional and pooling layers creates a feature extractor that produces many small-resolution feature maps with rich information about an image. Several feature extractor networks have been designed with great success and now appear as the *backbone* of many other DNNs.

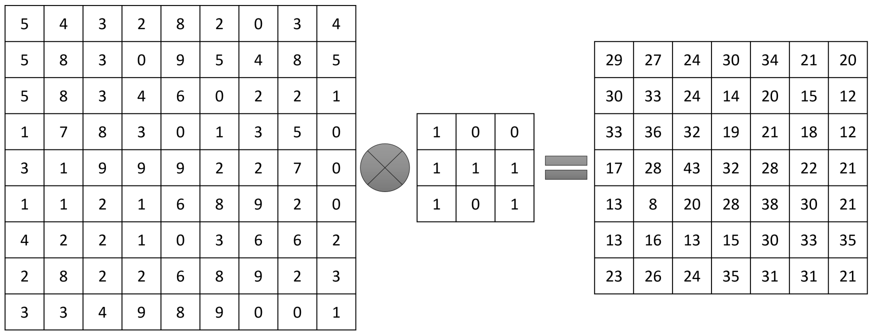


Figure 2.1: Convolution operation.

Image classification aims to classify an image into one of many labels. For this, feature extractors typically conclude with fully connected layers that provide probabilities of an image being assigned that label. In 1998, Yann LeCun created the LeNet-5 architecture designed for image recognition, composed of a series of convolutional neural networks [29]. This was one of the first pioneering efforts in image classification using deep learning. Due to the computational cost of deep learning, a long drought of deep learning research occurred. It was not until 2012 that AlexNet was created. AlexNet is another image classification DNN, but with a much more complex architecture [27] by creating multiple branches, achieving top-1 and top-5 error rates of 37.5% and 17.0%, respectively on the ImageNet dataset [45]. From here, research in deep learning for computer vision exploded and networks continued to become deeper. In 2014, GoogLeNet was created. A 22-layer DNN achieving top-5 error of 15.3% on the ImageNet dataset [52]. Finally, in 2015 the first residual network was created, ResNet. ResNet is a feature extractor with 151 layers, achieving a top-5 error rate of 3.57% on ImageNet. The top-5 error rate is the percentage of times the classifier failed to include the proper class among its top five guesses. ResNet is a massive residual network composed of a series of bottleneck components with identity mapping. Figure 2.2 shows the bottleneck component. In this case, identity mapping is the element-wise summation of previous feature maps before being processed by a series of convolutional layers, with the output of bottleneck convolutions. Depending on the use case, ResNet is a SOTA feature extractor and is still widely used today as the backbone for many different architectures designed for various vision tasks like object detection and image segmentation. ResNet can be made lighter weight using fewer bottleneck components, allowing one to trade off accuracy for speed.

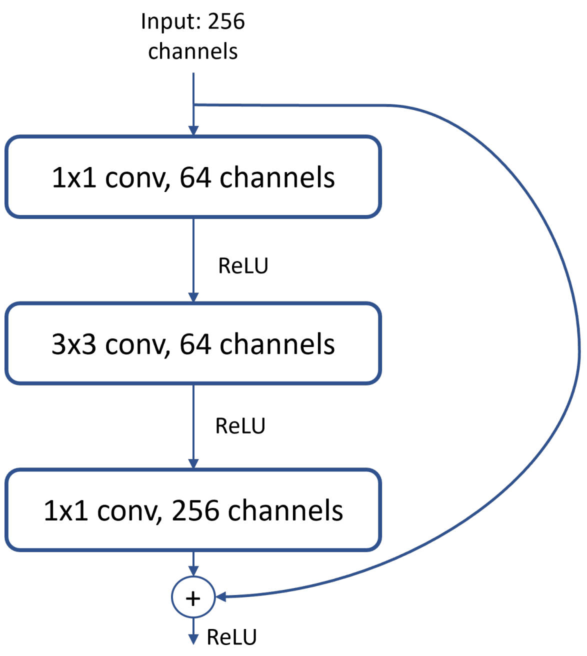


Figure 2.2: ResNet bottleneck component.

2.1.3 Object Detection

Another very common computer vision task is object detection. This task classifies objects of interest and identifies their locations in an image using bounding boxes [69]. All object detectors begin with feature extraction and, therefore, use a backbone similar to ResNet. Object detectors can be split into two types: two-stage detectors and one-stage detectors. Two-stage detectors try to achieve very high accuracy and contain several additional components. Because of this, it allows for a slower yet more accurate feature extractor. Two very popular two-stage detectors Mask R-CNN and Faster R-CNN [21, 19] both use a ResNet-50 as their backbone. Today, arguably, YoloV4 [6] is the most popular one-stage object detector and the most popular object detector in general. Due to its main goal of speed, it uses a variant of ResNet called DarkNet-53 which is slightly more efficient.

Object detection plays a critical role in autonomous driving applications, allowing vehicles to detect pedestrians and road signs, understand their environment, localize themselves relative to other vehicles, and make the final movement decision [33]. While this has demonstrated impressive results, full image segmentation provides a more complete understanding of images. However, providing accurate, full image segmentation has proven very difficult in a real-time setting.

2.1.4 Image Segmentation

Image segmentation can be classified into three different categories: semantic segmentation, instance segmentation, and panoptic segmentation. Semantic segmentation, the focus of this work, classifies every pixel in an image based on a defined set of classes and does not detect different instances of the same class. Instance segmentation classifies every pixel in an object and distinguishes between each instance, but does not label every pixel in the image. Lastly, panoptic segmentation combines semantic and instance segmentation to label every pixel in an image as well as distinguish between each instance of the same label [35]. Figure 2.3 shows a visual of each type of segmentation.

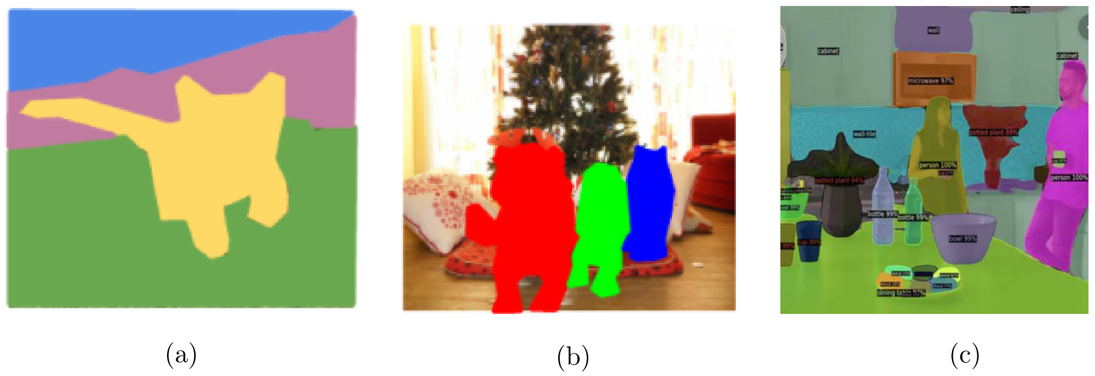


Figure 2.3: (a) Semantic segmentation [51] (b) Instance segmentation [51] (c) Panoptic segmentation [39]

Semantic segmentation provides a broad understanding of a vehicle's surroundings which can be useful to search for traversable areas. Because semantic segmentation networks do not require the localization of objects, their DNN architecture is fairly simple and can be easily compared to image classification. Most semantic segmentation architectures follow an encoder-decoder structure. The encoder is similar to a feature extractor which encodes a raw image into feature maps through the use of convolutional and pooling layers. After obtaining this semantic representation of the image, the resulting feature map is upsampled, with interpolation, back to the original image size except the depth size corresponds to the number of classes being identified. This results in pixel-wise probabilities of each pixel being labeled a specific class. After taking the maximum probability for each pixel along the channel axis, the DNN produces a fully segmented image [35].

Even though semantic segmentation provides a much better understanding of an image compared to object detection, it is much more computationally expensive due to pixel-wise classification. Thus, there exists a large trade-off between segmentation accuracy and inference speed in the

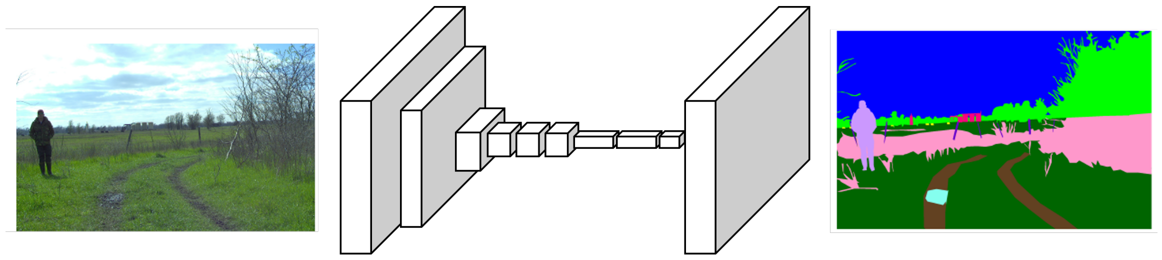


Figure 2.4: Full convolutional network [32].

literature. In 2014, the first semantic segmentation architecture was described using a fully convolutional network (FCN) shown in figure 2.4 [32]. This simple network consists of a backbone to encode the image features and uses interpolation as the decoder to upsample the feature maps back to the original image size. In 2015, U-net [44] revolutionized semantic segmentation by improving FCN. U-net follows a similar structure as FCN but instead of upsampling once, it upsamples in stages. During each upsampling stage, higher-resolution feature maps from previous encoder steps are combined to capture the previous context. From here, the DeepLab architecture series [10, 11, 12] was the next major improvement in semantic segmentation. They designed additional modules to further capture the feature space of images in more depth. Along with other innovations, the main component which came from DeepLab was the atrous convolution. Atrous convolution, or dilated convolution, captures different widths of spatial context by dilating the traditional convolution described by figure 2.5. For example, a 3×3 convolution kernel with a dilation factor of 2 spans a 5×5 region but only captures 9 pixels by skipping every 2 pixels. DeepLabV3+, the final iteration of DeepLab, uses a ResNet-101 backbone but replaces the last few convolutional layers with dilated convolutions. Additionally, the resulting features maps are passed through an atrous spatial pyramid pooling module (ASPP) which performs atrous convolution on the same features maps at different dilation rates. DeepLabV3+ substantially improved the accuracy of semantic segmentation, although, at a high computation cost.

While the previous segmentation models prove effective, they all focus on accuracy rather than inference speed, greatly reducing their functionality in real-time applications. One of the first attempts at creating a real-time semantic segmentation network on high-resolution images was ICNet [67]. In this architecture, three branches are created which each contain a separate feature extractor. The top branch downsamples the original image by a factor of 4, then extracts features through a full ResNet, the middle branch downsamples the image by a factor of 2 and uses a lighter-

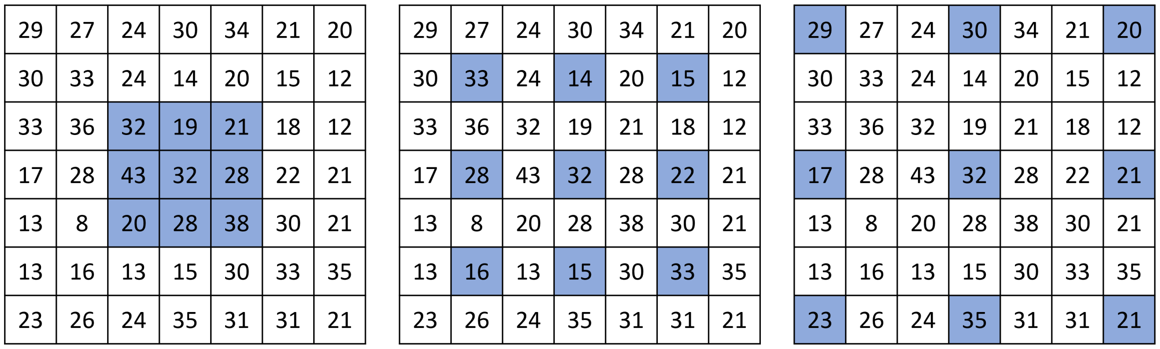


Figure 2.5: Dilated convolution.

weight feature extractor, and the last branch uses a very lightweight feature extractor on the full resolution image. Finally, the corresponding feature maps from each branch are fused together and then upsample back to the original image size to produce a segmented image. This architecture was able to achieve 30 fps on 1024×2048 images. This thesis utilizes a similar architecture, SwiftNet, which will be described in the methodology section. Segmentation accuracy and timing methods will also be described in the methods section.

2.2 Datasets

To compare and measure the accuracy of different DNN architectures, many benchmark datasets have been created. This allows one to train their model on the datasets' train set and evaluate its performance on the validation set. In many cases, the test set is private and can only be accessed by submitting a model to the evaluation server; this prevents fine-tuning on the test set to keep model comparisons fair and more generalizable. In this section, major semantic segmentation benchmark datasets will be introduced as well as a common dataset used for pretraining. The main dataset used in this thesis, Rellis-3D [25], will be introduced in the methodology section.

For the task of image classification, most evaluations use the ImageNet dataset [15, 45]. Today, ImageNet contains over 14 million images each labeled 1 of 1000 classes of commonly seen objects. In section 2.1.2, ImageNet was the primary benchmark used to assess the accuracy of image classification and compare different architectures. Furthermore, researchers very commonly use ImageNet to pre-train networks designed for other vision tasks as a form of transfer learning [63]. For network pretraining, researchers typically use a subset of 1.2 million images from the ILSVRC2010 challenge [45]. Pre-training initializes the weights of a DNN with already learned features designed for a similar task. For example, using this method, a network designed for semantic segmentation can be pre-trained on ImageNet to learn basic features which make up objects (e.g., curves, shapes, orientation); then, the network can be fine-tuned on the semantic segmentation dataset. The benefits of pre-training bring some controversy, however, many models throughout the literature pre-train on ImageNet [63].

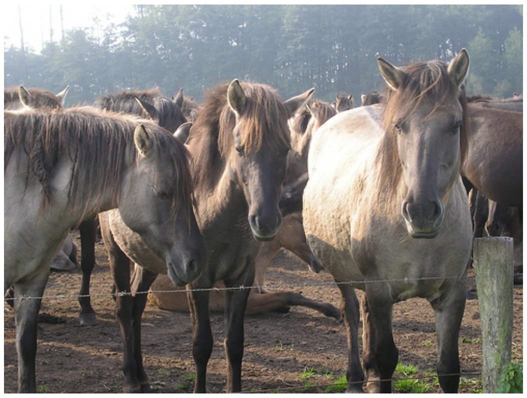
(a) Goldfish (b) Cannon (c) Dogsled

Figure 2.6: Random ImageNet images [15].

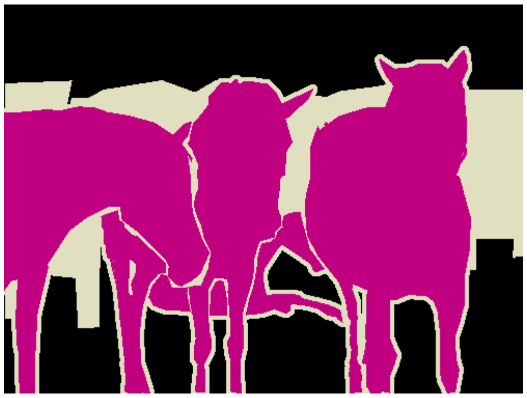
Similar to image classification, many semantic segmentation benchmark datasets exist to compare various architectures. Pascal VOC 2012 was one of the first benchmark datasets for semantic segmentation [16]. It contains just under 7,000 semantically labeled images and 20 classes which include vehicles, animals, and common household objects shown in figure 2.7. Image sizes vary between width and height of 500 pixels. Due to its low resolution, and other factors, the pascal dataset is no longer used as much to assess a semantic segmentation DNN.

A much more relevant dataset, and probably the most benchmarked semantic segmentation dataset, is Cityscapes [13]. The Cityscapes dataset contains thousands of semantically labeled street view images from many cities. 5000 high-resolution images were collected from an ego vehicle for the train and validation set which contains 19 classes. Unwanted classes were labeled as void and should be ignored during training and inference. Cityscapes remains a primary benchmark dataset for several reasons. Firstly, each image is 2048x1024 pixels which evaluates a model's computational efficiency while simultaneously evaluates its segmentation accuracy. Secondly, one of the primary applications for semantic segmentation is autonomous driving therefore Cityscapes helps simulate a real-time scenario. Figure 2.8 shows an example image from the Cityscapes dataset.

Regularly, these benchmark datasets rely on common objects and environments one would come across. For example, because of the myriad research on self-driving cars, many urban environment

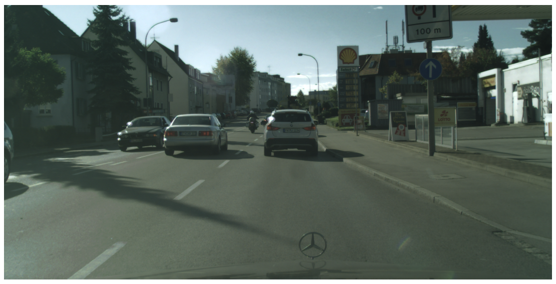


(a) RGB image

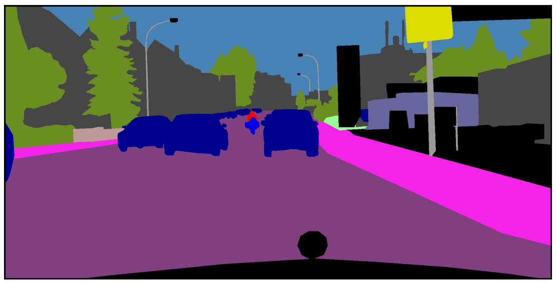


(b) Label image

Figure 2.7: Random Pascal VOC image.



(a) RGB image



(b) Label image

Figure 2.8: Random Cityscapes image.

datasets exist. Unfortunately, little data and research exist in the off-road domain, most notably for autonomous unmanned ground vehicles (UGV). This absence of research can be seen from the lack of research articles and scarcity of labeled off-road datasets [64, 25].

2.3 Segmentation in Unstructured Environments

Due to the increasing demands in today's world, autonomous UGVs have become of large interest. These UGVs cover a wide range of applications like military usage, extreme climate exploration, and search and rescue missions. However, this off-road space provides a completely different rule set than urban environments. Urban areas contain roads, street signs, crosswalks, curbs, and, in general, a much more structured terrain. On the other hand, off-road terrain is very unstructured by nature making it a much more challenging domain [54].

Just like in self-driving cars, perception systems are critical for autonomy in UGVs and other off-road vehicles [57]. The evaluation of semantic segmentation DNNs in off-road environments is a necessity to integrate higher levels of autonomy in UGVs. However, many factors exist which make off-road semantic segmentation more challenging than in urban settings [54]. In off-road terrain, no clear navigation path exists due to its ever-changing landscape, this makes identifying traversable areas a lot more difficult. Additionally, off-road environments are very noisy due to the type of objects. This can lead to poor generalization of a deep learning model, which can be detrimental in a fog-of-war type scenario. More so, images must be segmented in real-time due to the high speeds UGVs need to travel at. This proves difficult, as mentioned before, a trade-off between inference speed and segmentation accuracy occurs in DNNs. On the contrary, due to the application of most UGVs, it may be acceptable for semantic segmentation to be less accurate in off-road environments. Because UGVs are typically very large and durable, they can drive over smaller objects. Therefore, detecting obstacles like logs and bushes can be less important, while focusing on faster inference speed could be considered more important.

In addition to speeding up the inference time of a DNN in a perception system, other bottlenecks exist and can also be investigated. One of these bottlenecks is the image transfer time from a mounted camera on an autonomous vehicle to its video processing unit where the DNN is housed. In this research, we also investigated the use of lossy compression to speed up this image transfer time. Section 2.4 briefly covers a background on lossy compression. The actual experimental setup is described in the methodology and results section.

2.4 Data Compression

The time to transfer data over a network is directly correlated with the size of the data. Reducing the amount of data transferred with data compression should speed up the overall transfer time. Data compression reduces the overall data footprint (size) of files using a variety of algorithms. The type of algorithm depends heavily on the type of data being compressed. The heuristic may involve lossless compression, lossy compression, or a mix of both [47]. Lossless compression reduces the data footprint without any loss to the original data. Unfortunately, the lossless compression ratio is severely limited and, currently, it is near impossible to exceed a compression ratio greater than 2 [20]. Lossy compression also reduces the data footprint, but almost always results in original data loss when decoding. Therefore, one can achieve much higher compression ratios. Lossy data compression heuristics try to balance data reduction and data loss by combining several techniques. Many lossy algorithms compress data in a series of stages using involving data profiling, data manipulation, and data transforms. A very common lossy compression scheme for image data is JPEG compression [59]. JPEG compression works by using a discrete cosine transform (DCT) followed by Huffman encoding. DCT translates the image information from the spatial domain to the frequency domain which represents the data in a more compact form [24]. Huffman encoding is a lossless compression technique that represents symbols that appear most frequently with the least number of bits. JPEG compression is the standard for image files because it greatly reduces its size with little visual data loss. Another, more extensive, lossy compressor that takes advantage of multiple compression stages is SZ3 [30]. SZ3 allows a user to adjust the error bound which balances the amount of compression with the amount of data loss. Furthermore, SZ3 can be tailored to the specific type of data in order to improve its efficiency.

2.5 Summary

Since the explosion of deep learning, many SOTA algorithms have been replaced by DNNs across many disciplines: computer vision, natural language processing, regression, anomaly prediction, and much more. Most computer vision DNNs structure their architecture in a similar way. First, an encoder component extracts semantic features from images using convolutional and pooling layers. Then, a secondary component tailors the DNN to the task-specific goal. For image classification, the secondary component uses fully connected layers to create predicted class probabilities from the feature maps. For object detection, the secondary component uses convolutional layers to form a final prediction feature map such that each *cell* has a specified number of bounding box predictions [6]. Finally, for semantic segmentation, the secondary component typically uses interpolation to upsample the resulting feature maps back to the original input size, ultimately creating a semantic map of pixel-wise probabilities for each class of interest. In the research field of semantic segmentation, some of the most popular architectures include FCN, U-net, and DeepLab. While the architectures have been very successful, they struggle with inference speed which makes real-time deployment nearly impossible. More research needs to study DNN architectures that can perform full image segmentation in real-time. Furthermore, with the rise of autonomous driving, many benchmark datasets and research goals evaluate these segmentation models in city-like environments, however, not much research investigates off-road terrain for use in autonomous UGVs. The off-road domain typically presents harder challenges due to its unstructured nature. Unlike having roads, street signs, and boundaries, off-road typically has no clear path, a mesh of different, overlapping obstacles, and uncertainty. Furthermore, due to the lack of structure, DNNs usually struggle to generalize well on unseen off-road terrain. Other bottlenecks in an autonomous perception system should be explored and experimented with ways to improve the overall perception speed, not just the inference time in a DNN. One bottleneck is the image to video processing unit transfer time. Compressing images from a live video feed will speed up the transfer time, however, then a compressed image will be passed through the DNN. For this method to be viable, the impact of image compression on DNN accuracy must be investigated. The following section discusses related work similar to the experiments run in this thesis.

Chapter 3

Related Work

The two major works in this thesis evaluate methods to improve the overall perception system time, specifically for UGVs, by investigating various bottlenecks. The first work evaluates a highly efficient semantic segmentation DNN on off-road terrain. The second work compresses data from a live camera feed to speed up the transfer time as it approaches the DNN; then, the accuracy of the DNN is evaluated on the compressed data. This section discusses work in the literature similar to the experiments performed in this thesis.

3.1 High Inference Speed Semantic Segmentation DNNs

The trade-off between accuracy and inference speed in semantic segmentation has been realized for quite a while now [22, 33, 18]. Several efforts have been made to speed up semantic segmentation inference while reducing the accuracy as little as possible. Although, not much research evaluates inference speed and accuracy in an off-road setting.

One of the first successful attempts at a high inference speed and maintaining an acceptable level of accuracy was ICNet, accomplished by creating a multi-branch architecture [67]. First, the original image is downsampled by a factor of 2 and 4, then each image size is passed to a branch of the network with the smallest image going through the most amount of convolutional layers, and the larger images going through the least amount of convolutional layers. Finally, each branch is fused together by a cascade module. BiSeNet [65] is another multi-branch attempt to utilize larger and smaller image sizes on different branches. BiSeNet has two paths. The first path uses a lightweight model and is designed to grab context information and the second path has only three convolutional layers which extract spatial information. A feature fusion module is used to merge the extracted features from both paths. Several other attempts exist that experiment with similar, but different, architectures to retain representative image features with less expensive convolutional operations [35]. While many of these models are effective, they are only evaluated on the primary benchmark dataset Cityscapes, and secondary datasets like CamVid (another city-like environment), and COCO-stuff (common objects in context) [8, 9]. Most models were not evaluated in an unstructured, off-road environment.

3.2 Off-Road Environment Semantic Segmentation

Even though many semantic segmentation models do not evaluate performance on off-road datasets, there still exists research which investigates semantic segmentation in the off-road setting with datasets like RUGD and Rellis-3D [64, 25]. Even with the existence of off-road datasets, there is still the issue of class imbalance and irregular objects. To overcome these problems, OFFSEG [57] strives to create a two-stage framework for rich semantic segmentation results. The first stage condenses the ontology into four classes (sky, obstacles, traversable, and non-traversable regions), then trains a BiSeNet model on this reduced ontology. In the second stage, the regions of interest are extracted from the segmented image, and then further sub-classes (grass, mud, puddle, etc) are created using k-means clustering [3] from the regions of interest. Finally, the segmented sub-classes are appended to the first segmented image. This results in a detailed semantic image of class labels. Another approach explores transfer learning to alleviate the lack of off-road datasets [63, 49]. In this approach, a lightweight model based on VGG-16 [36] is trained using synthetic data as the intermediate domain. Then, the synthetically pre-trained model is trained with a real off-road dataset called the Freiburg Forest dataset [56]. Unfortunately, the accuracy of the model pre-trained with synthetic data achieved about the same accuracy as the model without pre-trained synthetic data.

This thesis evaluates a SOTA semantic segmentation DNN, SwiftNet, on a large off-road dataset, Rellis-3D. The primary goal of this study is to investigate the inference of SwiftNet on off-road images while maintaining an acceptable level of accuracy.

3.3 Data Compression to Reduce Latency

Several studies exist which demonstrate the use of data compression to reduce latency for a variety of applications. In [31], the authors target data compression on mobile-edge computing for energy internet through a local area network (LAN). In their work, they design a framework that uses several lossy compression algorithms to improve random access potential and reduce transmission latency. With their framework, network latency is reduced when measured from 200 to 300 random accesses compared to the traditional architecture. Another study [42] tries to minimize the network bandwidth requirements using H.264 video compression [34] for a roadside pedestrian safety application. The researchers develop an error-bound lossy compression strategy to dynamically adjust the compression levels of video frames in different weather conditions. Video transmission bandwidth requirements are reduced by up to 14$\times$ other SOTA strategies. This thesis expands upon this research by instead of utilizing object detection with YOLOv3 [43] in an urban environment; we investigate semantic segmentation using SwiftNet [38] in an off-road setting. Furthermore, this thesis measures latency in a local Ethernet-connected system instead of bandwidth through a wireless network.

Because we use JPEG-compressed images for deep learning, the effects of lossy compression on DNNs must be understood. In [4], the authors compress time series data using a compression method called Discrete Wavelet Transform and study the effect of classification accuracy from a fully convolutional network (FCN). They found that the FCN model could still achieve 80-90% accuracy on compressed data. Our experiments investigate a similar problem but with semantic segmentation accuracy on image data.

The remainder of this thesis covers the methodology for our work, experimental results, a discussion of the results, and future work to be considered.

Chapter 4

Research Design and Methods

In this thesis, two bottlenecks within an autonomous perception system of UGVs were investigated. First, the primary investigation evaluates a SOTA deep learning model, SwiftNet, designed for real-time inference on high-resolution, off-road images. Much of literature focuses on segmentation accuracy rather than inference speed. This work seeks to add to the literature by evaluating inference speed as the primary metric. After, we discuss the secondary investigation. This considers the image transfer time through an Ethernet cable from a mounted camera to a processing unit containing the DNN. We compress the images from the mounted camera, transfer the data to the processing unit, and measure the overall speedup. We aim to improve the Ethernet transfer time through image compression without a substantial decrease in accuracy. For both investigations, the PyTorch deep learning framework is used[40].

4.1 Evaluation of Semantic Segmentation

The following section discusses the methods to design, implement, train, and evaluate SwiftNet. Specifically, this includes the DNN architecture, the off-road dataset, the training protocol, the evaluation protocol, and the metrics used.

4.1.1 SwiftNet Architecture

SwiftNet follows two common methodologies in architecture design discussed in section 2.1. Firstly, SwiftNet follows an encoder-decoder structure. Secondly, SwiftNet pre-trains the image encoder on the ImageNet dataset [50] in order to benefit from the knowledge learned. The encoder is built from lightweight feature extractors, ResNet-18 and Mobile Net V2 [46]. The decoder, which upsamples the rich feature maps to the original input size, consists of a ladder-style structure with two inputs: the low-resolution feature maps from the preceding upsampling module, and the high-resolution feature maps from the corresponding encoder block. Finally, the upsampled input features and adjacent encoder features combine with elementwise summation and fused with a 3x3 convolution. The SwiftNet implementation has two architectures, a single-scale architecture, and a multi-scale architecture with pyramidal fusion [38]. The multi-scale architecture, shown in Figure 4.1, extracts features at three different image sizes using three different branches. Each of the four encoder blocks shares the same parameters between branches. SwiftNet also uses a unique loss function called the boundary-aware loss [69]. This emphasizes the importance of pixels near semantic boundaries in hopes of avoiding overfitting in small objects [37]. In this thesis, the multi-scale architecture based on the ResNet-18 backbone is opted for due to the quality of their results.

4.1.2 Rellis-3D Dataset

Rellis-3D is an extensive off-road dataset designed for semantic segmentation, containing 6234 labeled 1920×1200 images [25] with 20 class labels. Figure 4.3 shows random images, colored ground truth labels, and normal ground truth labels from the Rellis-3D dataset. For this research, 70% of the data was used for the train set and the remaining 30% was used for the test set. Figure 4.2 shows the Rellis-3D ontology. Rellis-3D contains 20 class labels which consist of traversable areas like dirt, grass, asphalt, rubble, and obstacles like bushes, trees, objects, and poles. Because the dataset was recorded after a rainy day, there are very few dirt labels so this label was excluded from the study.

4.1.3 Implementation

After establishing SwiftNet for the Cityscapes dataset, the model was adjusted to run with the Rellis-3D dataset. Most notably the number of output channels were changed to match the number of classes in Rellis-3D, a created a custom dataset class for Rellis-3D was created, and the mean and standard deviation of Rellis-3D were calculated. The Cityscapes and Rellis-3D dataset

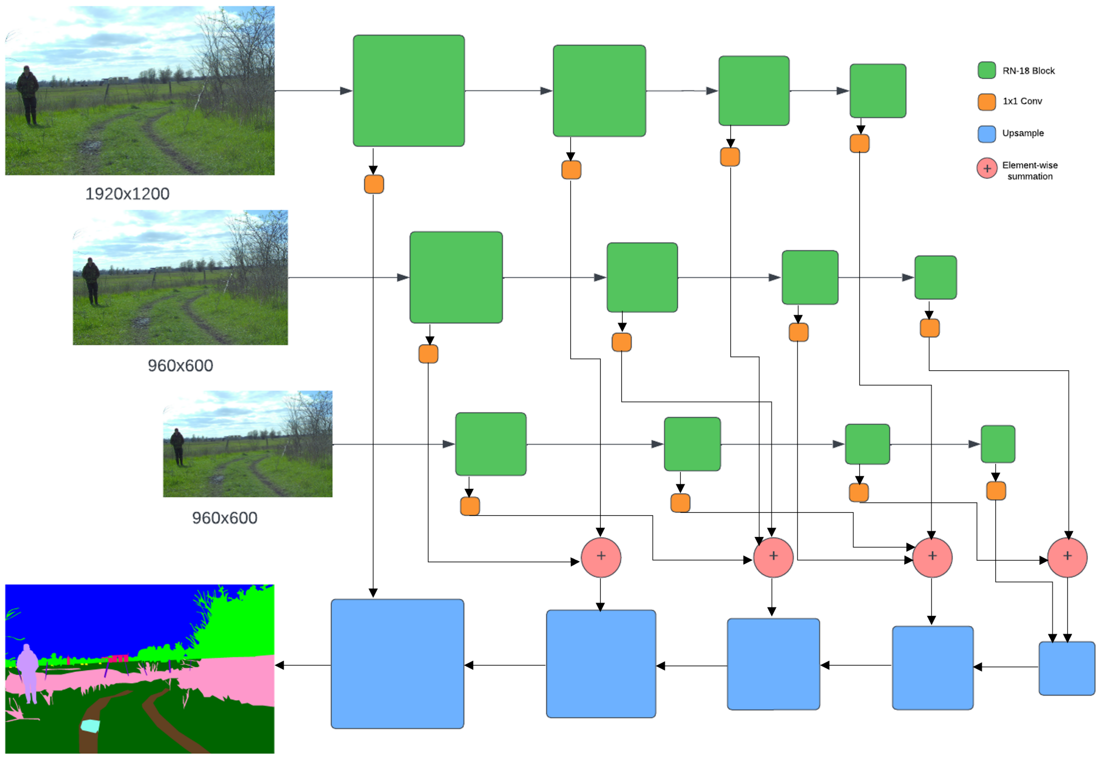


Figure 4.1: SwiftNet Architecture. The green blocks represent ResNet-18 blocks. The orange blocks represent 1×1 convolutions to establish dimensionality. The red circles represent element-wise summation. The blue blocks represent upsampling using bilinear interpolation.

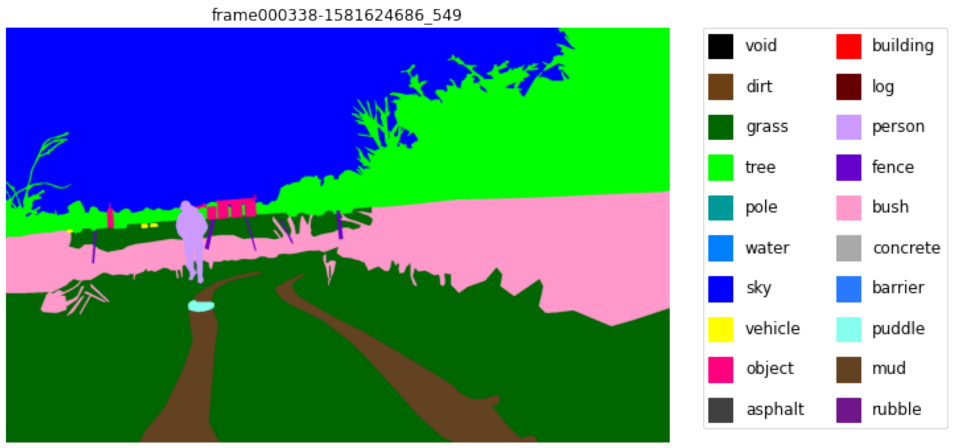


Figure 4.2: Rellis-3D ontology.

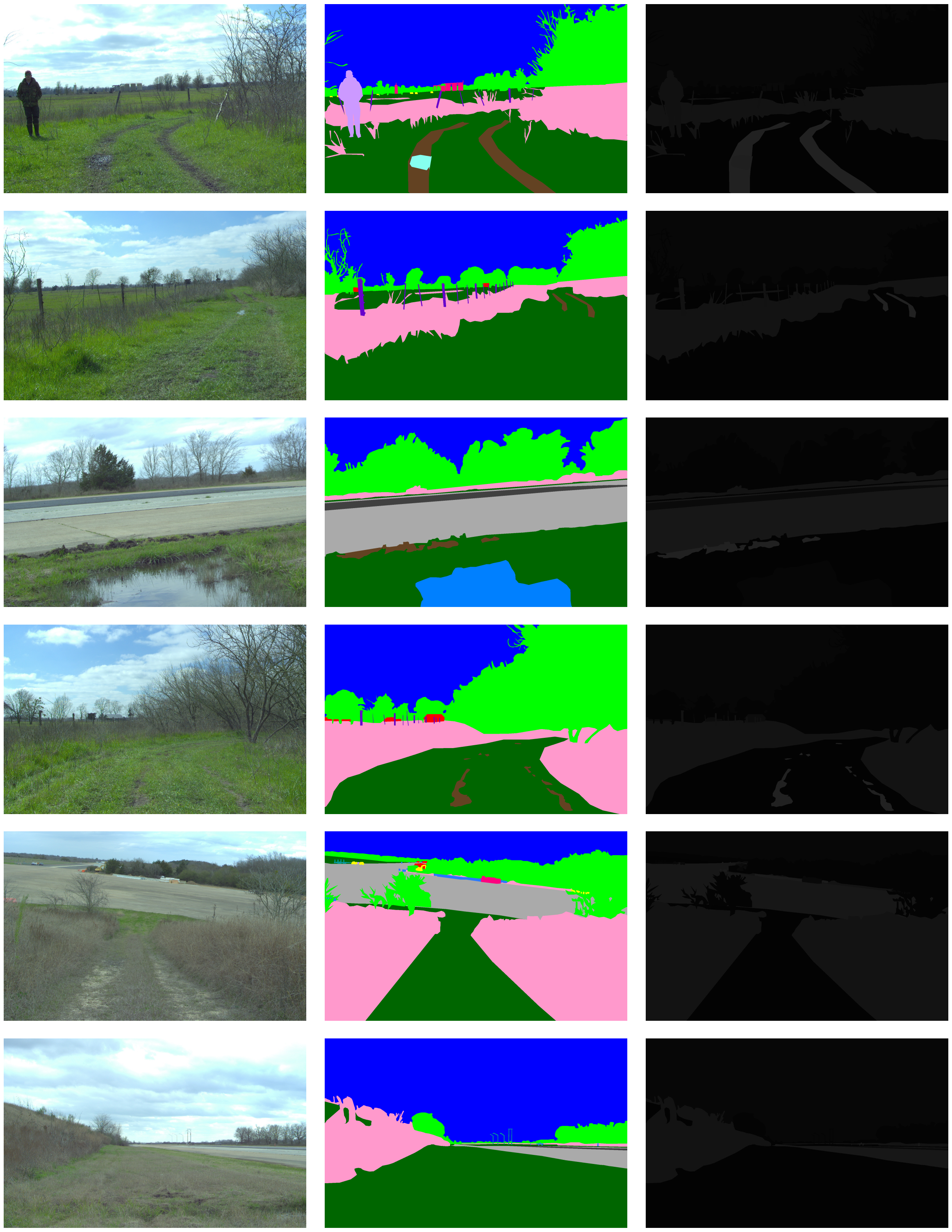
follow the same training and evaluation protocol on a single Titan RTX GPU.

4.1.4 Training

This research follows the same training protocol as in [38]. In detail, SwiftNet uses the Adam optimizer [26], the boundary-aware loss function, and an initial learning rate of $4 \cdot 10^{-4}$ which decays with cosine annealing at a decay rate of 10^{-4}. Each image in the train set was augmented with random horizontal flipping [50], random scaling between $0.5 - 2.0$, and random cropping of size 768×768. Finally, each image was normalized by using the mean and standard deviation of the dataset. The batch size is set to 14 and the model was trained for 250 epochs. During training, the model saves the weights from the epoch with the highest accuracy. The weights with the highest accuracy were used for the evaluation phase.

4.1.5 Evaluation

For model evaluation, the Rellis-3D test set was used which contains 1,870 images. The primary metric was inference speed, frames per second (fps), and the model's accuracy was the secondary metric. When describing the accuracy of a semantic segmentation model, the primary metric used is the Jaccard Index [16], also known as the intersection over union (IoU), shown in equation 4.1. The intersection and union rely on the true positive (TP), false positive (FP), and false negative (FN) values. Pixels labeled as void do not contribute to the score. To describe the overall IoU, the mean IoU is used (mIoU), an average of all the class IoU scores shown in equation



(a) (b) (c)

Figure 4.3: Random images from Rellis-3D (a) Raw image (b) Ground truth label image with the color map applied (c) Ground truth label image of class IDs

4.2.

$$IoU_{class} = \frac{TP}{TP + FP + FN} \tag{4.1}$$

$$mIoU = \frac{\sum IoU_{class}}{n\_classes} \tag{4.2}$$

For timing the inference speed in PyTorch, a similar approach as in [38] was followed. Algorithm 1 shows the timing method assuming a batch size of 1 in frames per second. It is important

Algorithm 1 Proposed timing approach in PyTorch

```
device = torch.device('cuda')
n = len(dataset_loader)
model.eval()
model.to(device)
torch.cuda.synchronize()
start_t = perf_counter()
with torch.inference_mode():
    for data, target in dataset_loader:
        data, target = data.to(device)
        outputs = model(data)
        _, pred = torch.max(outputs, 1)
        pred = pred.byte().cpu()
    torch.cuda.synchronize()
    end_t = perf_counter()
    FPS = n / (end_t - start_t)
```

to note that the current literature is very ambiguous in describing inference speed. Inference speed depends on many factors such as hardware used and the exact timing interval in code. Many papers describe the GPU used for inference, however, not many papers describe the exact timing interval. For example, if the GPU to CPU transfer time is left out, this reduces a lot of overhead and could artificially inflate the FPS. The next section focuses on our secondary bottleneck investigation using lossy compression.

4.2 Semantic Segmentation with Lossy Compression

A real-time perception system in autonomous vehicles can be described similarly to figure 4.4. Reducing the latency from the camera feed to the video processing unit speeds up the overall system time. For this work, the setup first takes in camera data (to simulate this pipeline, the Rellis-3D images are used) and performs lossy JPEG compression with FFMPEG. SCP transfers

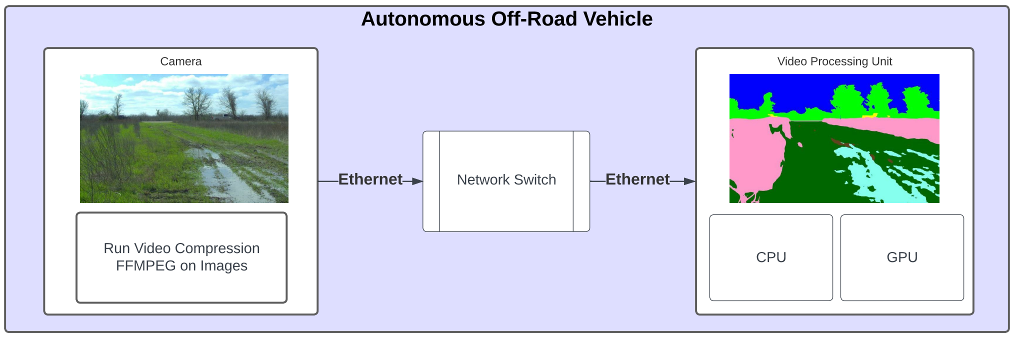

Autonomous Off-Road Vehicle
Camera
Run Video Compression FFMPEG on Images
Ethernet
Network Switch
Ethernet
Video Processing Unit
CPU
GPU

Figure 4.4: Local data transmission setup to simulate an autonomous off-road vehicle. Our local computer represents the camera, and the Jetson Xavier represents the Video Processing Unit. Both devices are connected through two Ethernet cables and a Network Switch (Linksys SE1500).

the image data over Ethernet to a local network switch and then sent to the local Jetson Xavier, which serves as the video processing unit. The Jetson Xavier then performs semantic segmentation using SwiftNet to classify the terrain from the compressed image data. We also seek to understand the effects of image compression on segmentation accuracy

To benefit from image compression, the time to send compressed images, T_{comp_send}, must be less than the time to send uncompressed images, T_{send}. Where T_{send} is defined as the data size over the Ethernet bandwidth (BW), T_{Comp} is defined as the amount of time to compress the image, and T_{comp_send} is defined as the sum of the time to compress the data and the time to send the compressed data.

$$T_{send} = \frac{Size}{BW} \tag{4.3}$$

$$T_{comp\_send} = \frac{Size}{T_{Comp}} + \frac{Comp\_Size}{BW} \tag{4.4}$$

$$T_{comp\_send} < T_{send} \tag{4.5}$$

The following section describes the approach to implement compression in a real-time vision system. First, the compression method, dataset, and DNN for this experiment are discussed. Then, the data transfer setup is described.

4.2.1 Image Compression

With the FFMPEG tool [55], the Rellis-3D train and test set were compressed using JPEG compression [59] into 31 different compression levels. This research uses compression level to describe the image quality tuner value on FFMPEG. Compression levels correspond to an average compression ratio shown in figure 4.5, so a higher compression level equates to a higher compression ratio. The compression ratio is the relative reduction in data size from the JPEG compression algorithm.

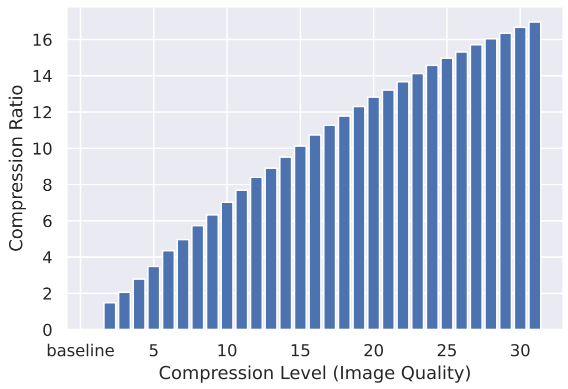


Figure 4.5: FFMPEG image quality (compression level) to compression ratio. The baseline represents level 1, with no compression.

4.2.2 Semantic Segmentation

Like the primary investigation, SwiftNet is used for semantic segmentation. SwiftNet is utilized because it favors high inference speed, which will more likely be found in a real-time perception system compared to other DNNs like DeepLabV3+ [11] which favors higher accuracy. Two experiments regarding semantic segmentation accuracy are performed on compressed images. First, a baseline model was trained on the Rellis-3D train set without compression. Then, the mIoU was evaluated at each compression level using the test set to observe how compression affects the model's accuracy. Secondly, a model is trained at six different compression levels (5, 10, 15, 20, 25, and 30) and the mIoU was evaluated at the corresponding compression level. For example, if the model was trained on the train set with level 15 compression, this model was evaluated at the level 15 compressed test set.

4.2.3 Data Transmission

To simulate image transfer in an autonomous vehicle, a local Ethernet system described in figure 4.4 was set up. To verify the communication and find the theoretical limits of our system, the bandwidth from our local computer to the Jetson Xavier using iPerf3 was measured. Secure Copy Protocol (SCP) was used to send three sets of compressed images through the system for the image transfer experiments. The transfer between the two local hosts for each image set at different compression levels was timed. The next section presents the results of both investigations.

Chapter 5

Experimental Results and Discussion

In this chapter, the results of both investigations will be discussed. Preliminary results from future work topics will be included in the future work section.

5.1 Evaluation of Semantic Segmentation

First, this thesis presents unique results for the inference speed and mIoU on the Rellis-3D dataset. Although this research focuses on off-road environments, Cityscapes results will be shown and compared to the original SwiftNet paper [38]. This can serve as replication and verification of results.

5.1.1 Rellis-3D

Using SwiftNet's multi-scale architecture on the Rellis-3D dataset, an average inference speed of 24 FPS with an mIoU of 77.9% was achieved. These results are encouraging due to the difficulty of segmenting unstructured terrain. Similarly, in the IoU class breakdown, table 5.1, traversable and non-traversable regions obtain higher accuracy than smaller objects. Autonomous UGVs are interested in traversing off-road terrain and are more likely to drive over small objects like logs and poles rather than avoid them. In this scenario, higher inference speed is favored over higher accuracy. Figure 5.1 shows various segmentation results from the Rellis-3D dataset. The figure is split into two rows. For each row, the top image represents the raw camera image, the middle image represents the SwiftNet prediction, and the bottom image represents the ground truth labels.

Class	IoU (%)
grass	91.83
tree	90.04
pole	42.15
water	81.48
sky	97.54
vehicle	67.30
object	72.73
asphalt	86.08
building	65.08
log	61.79
person	92.52
fence	65.61
bush	85.09
concrete	91.24
barrier	87.63
puddle	80.96
mud	65.46
rubble	77.87

Table 5.1: Individual class IoU from the Rellis-3D dataset

The inference speed achieved in this work on Rellis-3D is lower by about 10 FPS than what was achieved in the original SwitfNet study on the Cityscapes dataset, which inferences at 34 FPS on a GTX 1080Ti GPU with the multi-scale architecture. Several reasons exist as to why the study presented here might have achieved lower than expected inference speed compared to the original study. First, the Rellis-3D images are 1920×1200 which is a total of $1920 * 1200 = 2304000$ pixels. The Cityscapes images are $2048 * 1024 = 2097152$ pixels, making the Rellis-3D images more computationally expensive. However, in this study, when timing SwiftNet on the Cityscapes dataset there is minimal difference in inference speed between datasets. Second, the inference speed is timed at the start of the forward pass after the model and tensor are transferred to the GPU, and the timing ends after the predicted classes for each pixel is transferred back to the CPU. If the timing ends after the predicted classes but before the GPU to CPU transfer, the inference speed increases to 30 FPS. Even though this study obtained a lower inference speed than the original SwiftNet work, it still obtains high inference speed and can be a very viable DNN in perception systems for autonomous UGVs. In terms of accuracy, this study achieved a higher mIoU than the original

Class	IoU (%)
road	95.10
sidewalk	79.40
building	91.30
wall	50.10
fence	49.11
pole	61.84
taffic light	67.91
traff sign	74.96
vegetation	91.98
terrain	58.64
sky	94.10
person	80.76
rider	61.50
car	94.17
truck	67.58
bus	82.33
train	65.41
motorcycle	54.81
bicycle	75.93

Table 5.2: Individual class IoU from the Cityscapes dataset

SwiftNet work on the Cityscapes dataset, which is summarized in table 5.3. Two major reasons for this could be that Cityscapes uses 19 labels while Rellis-3D only uses 18, and Rellis-3D has more images in its training set than Cityscapes.

5.1.2 Cityscapes

In this research, the Cityscapes data set was used to implement SwiftNet and attempt to replicate the original research. Because this thesis focuses on off-road environments, this section is only briefly covered. The Cityscapes dataset contains 5000 images for semantic scene understanding in urban environments [13]. It contains a train set of 2975 images, a validation set of 500 images, and a test set of 1525 images. The test set labels are private and an mIoU score on the test can only be obtained by submitting a model to the evaluation server. For this reason, the current study evaluates the trained SwiftNet model on the validation set. In this study, SwiftNet attains 26.9 FPS with an mIoU of 73.5%. Like Rellis-3D, Cityscapes produces slightly lower FPS than the original study. Additionally, the mIoU appears slightly lower than the original. Additionally, the mIoU appears slightly lower than the original, which may be attributed to the stochastic nature of the neural network training process. Table 5.2 shows the individual IoU score for each label and table

5.3 summarizes the metrics between the current and original study.

Study	Dataset	FPS	mIoU (%)
Current study	Rellis-3D	24.5	77.9
Current study	Cityscapes	26.9	73.5
Original study	Cityscapes	34.0	75.9

Table 5.3: Inference speed and mIoU comparison of this work and the original work [38]

5.2 Semantic Segmentation with Lossy Compression

This section covers the results of the experiments pertaining to lossy compression and data transfer. First, the semantic segmentation accuracy of the SwiftNet model at various compression levels was explored. This includes the baseline model evaluated at multiple compression levels and the model trained at six different compression levels. Second, the latency of Ethernet transfer, the overhead of compression, and the overall end-to-end segmentation were timed.

5.2.1 Segmentation Accuracy of Various Compression Levels

When compressing images in autonomous vehicles, the compressed images must achieve an acceptable level of accuracy in the environment to navigate safely. Like before, because this research focuses on autonomy in UGVs, the Rellis-3D dataset was used. The same training and evaluating protocol as in section 4.1.4 was followed. Figure 5.2 illustrates the mIoU for the uncompressed model evaluated with test data compressed to a given level.

The baseline evaluation with no compression achieves a relatively high mIoU of 78.92%. From here, the accuracy steadily decreases in small increments when evaluated at each compression level. The most significant drop in accuracy occurs from compression level 27-31, the lowest mIoU being 67.90%. The most significant end-to-end speedup in later experiments occurs towards the higher compression ratios. Therefore, maintaining higher accuracy at these compression ratios becomes crucial. The following experiment obtains higher accuracy at the end compression range.

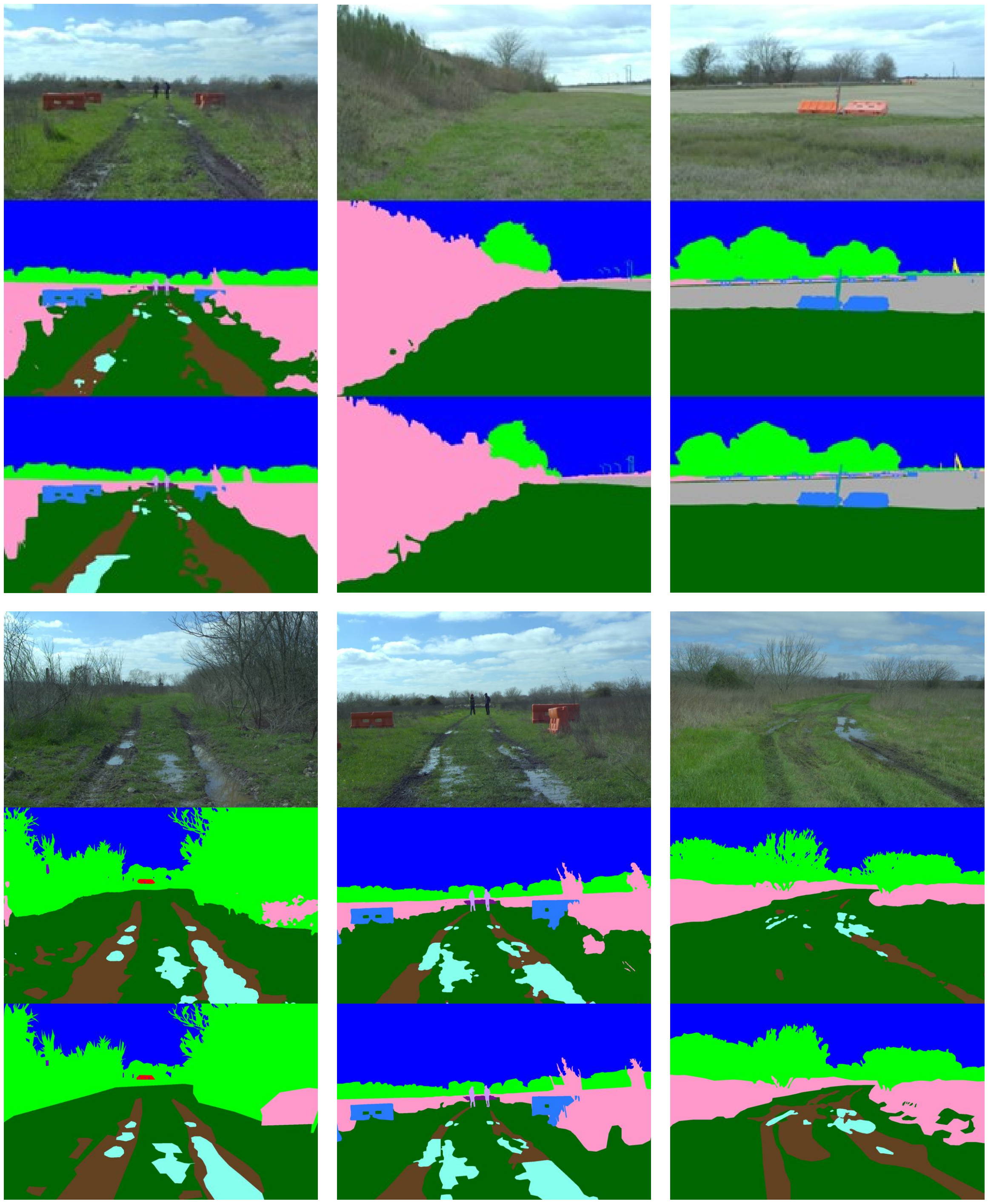


Figure 5.1: Segmented images with SwiftNet from the Rellis-3D test set: original image (top), predicted segmented image (middle), and ground truth segmented label (bottom)

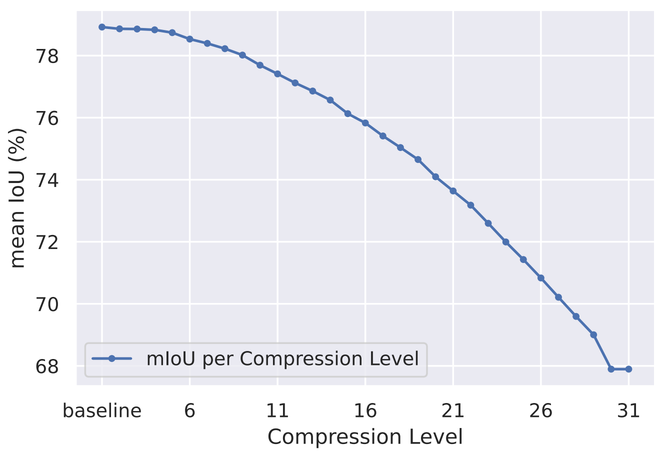


Figure 5.2: Evaluation of mIoU at each compression level using a model trained only on the baseline dataset (no compression).

5.2.2 Training with Various Compression Levels

Due to the data loss from the JPEG lossy compression, a deep learning model can misclassify object boundaries if the pixel has high enough distortion. From this study, training a segmentation model on these distorted images with a given compression level and then evaluating using the same level achieves higher mIoU at higher compression ratios. The highest compression level reaches a mIoU of 74.9%, a 7% mIoU increase from the baseline trained model at the same compression ratio. Due to time constraints, only six different compression levels were trained for 100 epochs. Figure 5.3 shows a significant drop off in accuracy compared to figure 5.2 from compression levels 2-6, but from level 6-30, only a 2% mIoU drop exists. Furthermore, compression levels 20-30 in figure 5.3 show a significant improvement in mIoU compared to figure 5.2. This most likely occurs because lossy compression slightly distorts the features of an object, so training a model on the distorted features will allow it to learn the object better. These results show a strong semantic segmentation accuracy when training and evaluating compressed images; further experiments now focus on image compression for increased end-to-end system speed.

Looking at figure 5.3 and 5.4 we can see that a compression-trained model performs best at compression level 12 or above. Looking at the accuracy depreciation, the range 12-25 stays within 0.4% mIoU difference on either end. When the compression level goes above this range into the 25-31 levels, the mIoU takes another 0.5% decrease.

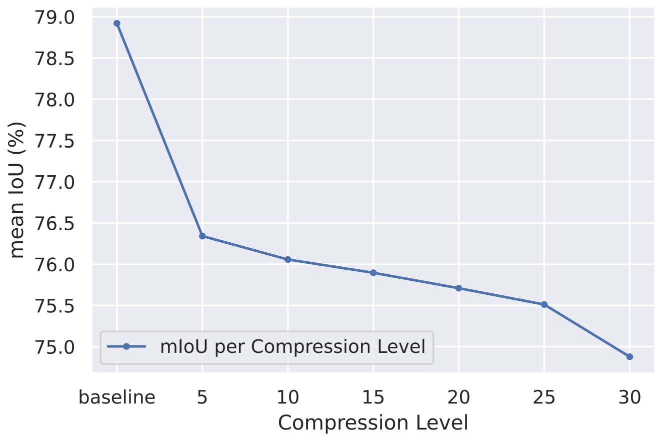


Figure 5.3: Evaluation of mIoU on six different image quality levels, trained on its respective image quality level.

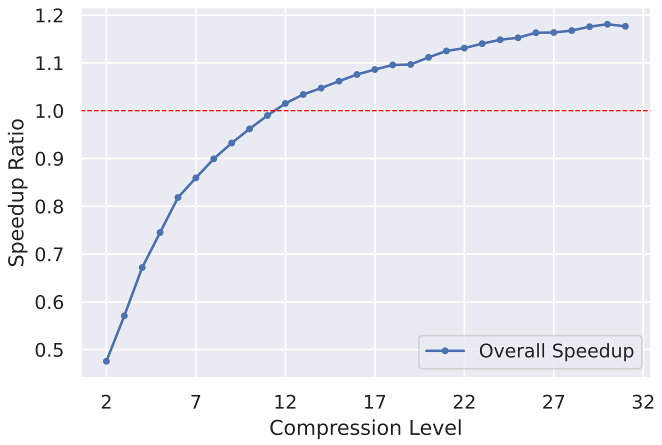
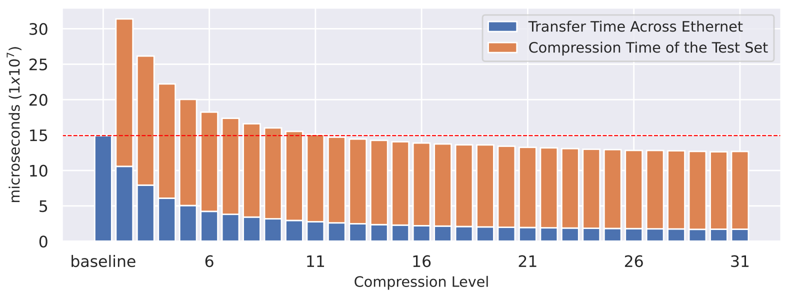


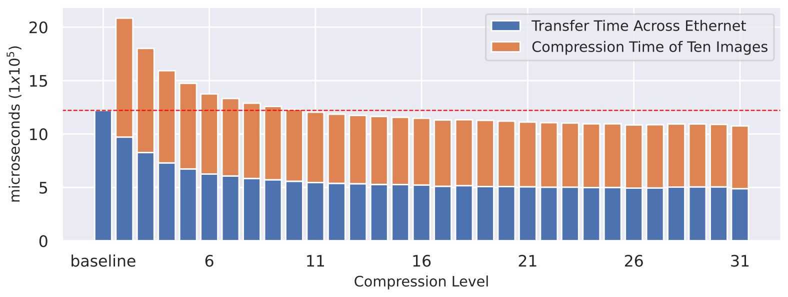
Figure 5.4: Overall transmission speedup of the entire compressed Rellis-3D test set. Values above 1.0 represent a positive speedup, and values below 1.0 did not benefit from compression.

5.2.3 Transfer, Compression, and End-to-End Timing

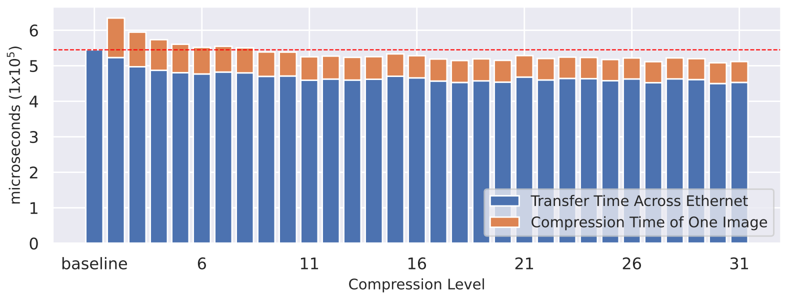
Using SCP, the transfer and compression time of the Rellis-3D test set (1870 images), ten images, and one image were measured. The compression time contains a summation of per-image compression. Figure 5.5 provides the transfer and compression times for three sets of images measured, corresponding to the blue and orange bars, respectively. The baseline level does not include an orange bar because the baseline has no compression. The total summation of orange and blue bars provides the entire end-to-end system time. Compression levels below the red dashed line demonstrate a positive overall speedup when inducing compression to reduce the image transfer



(a) Transfer batch size: full test set of 1870 images.



(b) Transfer batch size of 10 images.



(c) Transfer batch size of 1 image.

Figure 5.5: Transfer, compression, and overall system time for the entire Rellis-3D test set when using a different number of images. Compression levels below the dashed red line show an end-to-end speedup compared to the baseline.

time.

In figure 5.5(a), the first positive system speedup occurs beginning at compression level 12 compared to baseline. The speedup continues to increase in small increments to compression level 31. Compression level 31 shows the greatest speedup of 1.18×. While figure 5.5(a) results look promising, in a real-time perception system, it may be unreasonable to compress and transmit over 1000 images before segmenting occurs. Figure 5.5(b) shows a positive speedup beginning at compression level 11 and continuing to gradually increase to compression level 31 by 1.14×. Compressing and transmitting ten images at a time shows a very noticeable speedup in overall system time. This could be very viable in a real-time perception system due to the high frame rate of modern cameras. More so, figure 5.3 shows a high mIoU score for the compression levels at which the speedup occurs. Finally, while figure 5.5(c) does show a speedup beginning at compression level 9, no consistent speedup trend exists compared to the other figures. The highest speedup occurs at level 30 by 1.06×. The compression and transfer times stay very similar across compression levels, most likely due to the very small compression sample and the maximum bandwidth capabilities of our real-time system setup. The speedup gained through sending a different amount of images is due to the overhead of setting up the communication protocol. The 5.5(c) had this overhead on every image in the dataset, while 5.5(b) was able to send 10 images before introducing more over head. 5.5(a) only required this setup once before transferring the dataset.

Chapter 6

Conclusion

With the continued effort to integrate higher levels of autonomy in self-driving cars, research in autonomous UGVs falls behind. The evaluation of state-of-the-art DNNs in off-road terrain helps alleviate these gaps. In the context of perception, semantic segmentation provides a better understanding of the surroundings than object detection but at a higher computational complexity. SwiftNet demonstrates strong semantic segmentation results in terms of both inference speed and accuracy in less structured environments. Because components of perception systems in autonomous UGVs must operate in real-time, other bottlenecks exist other than the DNN inference speed. Another bottleneck is the communication latency within the perception system, specifically through Ethernet. This thesis finds that compressing groups of images before transferring them to the processing unit reduces the latency with slight accuracy loss. Much of this accuracy loss can be recovered by training the DNN on compressed images beforehand.

The primary investigation was evaluated in real-life scenarios through a live video feed on a Husky robot. Upon deployment, two major challenges were faced: fog of war and temporal consistency. Naturally, increasing inference speed through a DNNs architecture can lead to both of these challenges, but so can highly accurate models. Fog of war relates to the uncertainty of the environment in military scenarios, such as a dust storm in the desert. In general, fog of war can be referred to as the generalization of a deep learning model. The second challenge we faced was temporal consistency. This refers to the uncertainty of a DNN prediction, causing the prediction to vary between two or more classes of each frame in a video, resulting in a flickering effect. Because temporal consistency has a major effect on motion planning and the development of a cost map, this challenge was chosen to be investigated first. The following section provides very brief preliminary results in an effort to increase temporal consistency on a live video feed.

6.1 Future Work and Preliminary Results

To help increase temporal consistency, a method similar to AuxAdapt [66] shown in figure 6.1 was used. The main idea behind AuxAdapt is to make the DNN predictions more confident. This works by using a powerful, but efficient, main segmentation network and a lightweight auxiliary

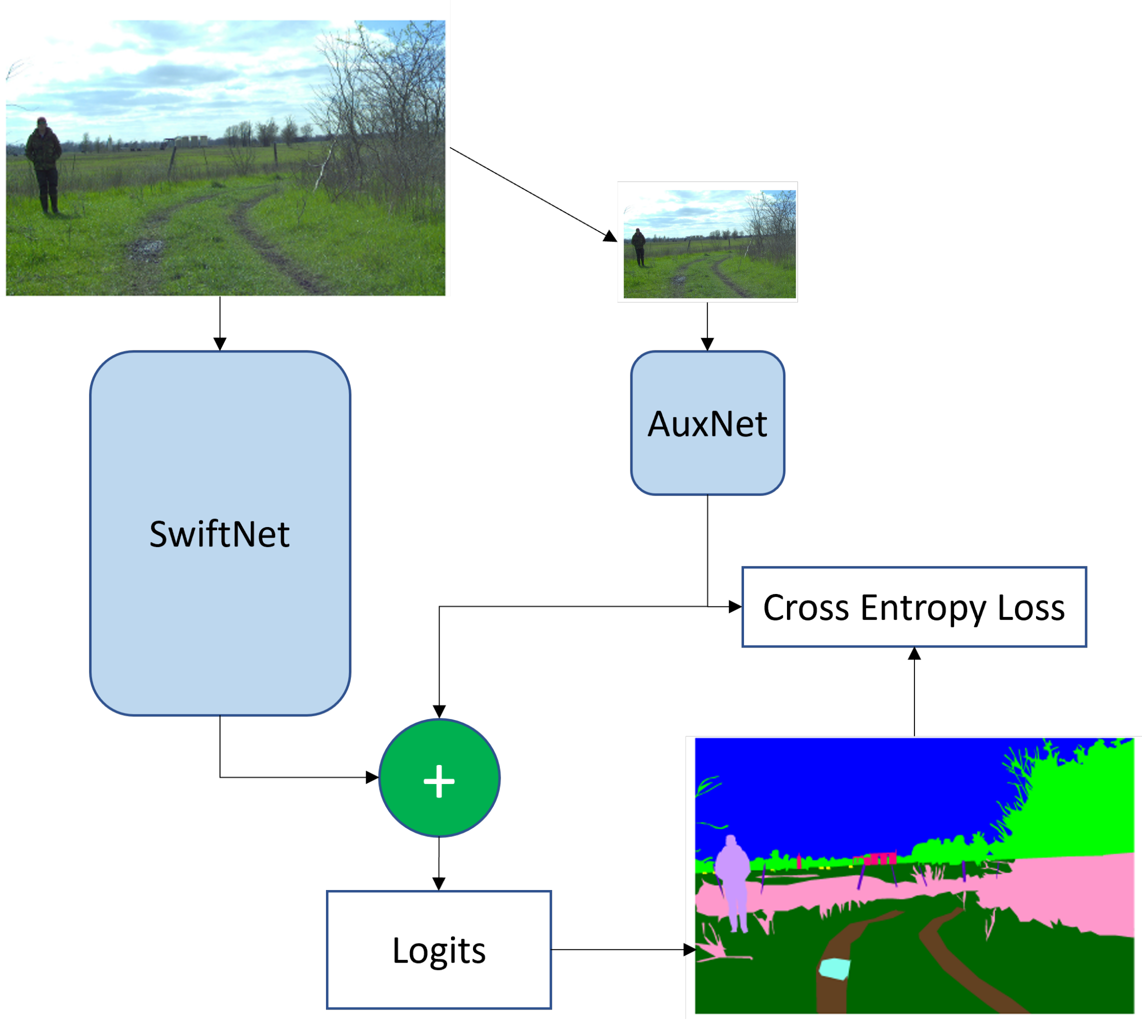
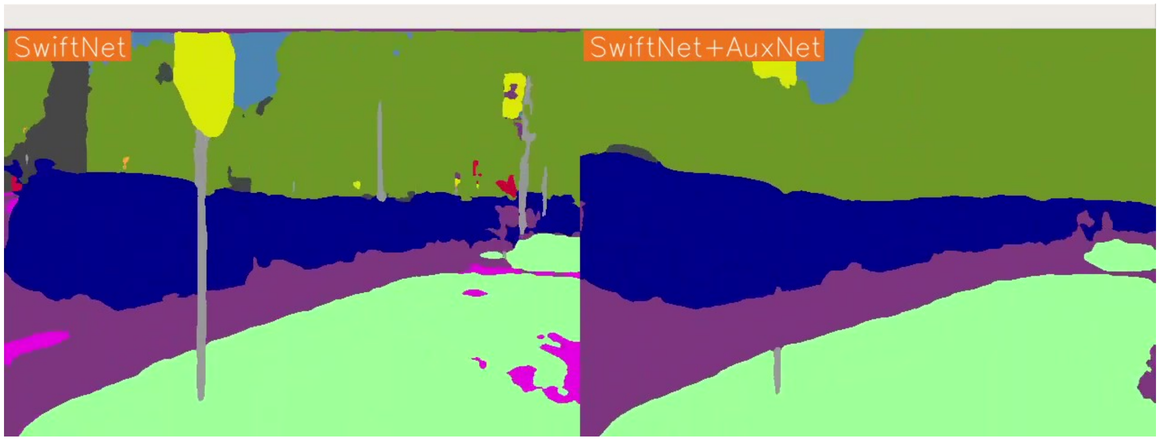


Figure 6.1: AuxAdapt method using SwiftNet as the main network and HRNet as the lightweight auxiliary network.

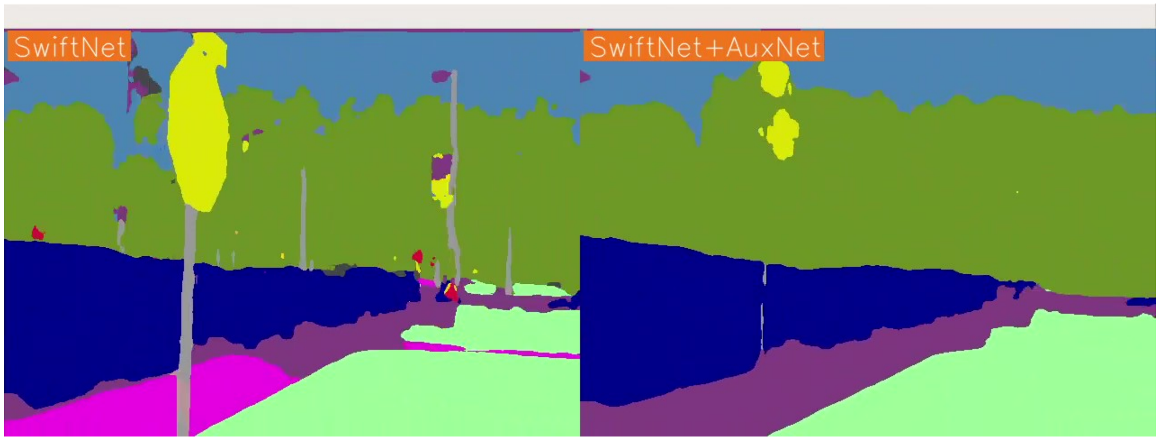
network. For this work, SwiftNet is used as the main network and a lightweight version of HRNet as the auxiliary network [60]. The main network was fully trained on Rellis-3D and the AuxNet was only pre-trained on the ImageNet. The main network inputs the full-resolution image and the auxiliary network inputs a downsampled image, in this case by a factor of 3. The logits from both networks are combined with element-wise summation and a final segmentation map is created by taking the maximum logits for each class channel. The cross-entropy loss is found by using the AuxNet predictions and the final segmentation map as the ground truth labels. Finally, we perform backpropagation only on the lightweight AuxNet because the main network is kept frozen. This allows the overall system to become more confident with the main network's predictions. Similarly, this method does not add much overhead because we use a small input image on a lightweight network and only backpropagate on this lightweight network. This allows us to keep the overall system in real-time.

Figure 6.2 tries to demonstrate the increase in temporal consistency. The results become more noticeable when seeing the video in real-time. The left image shows the regular SwiftNet predictions and the right image shows the AuxAdapt predictions. Frame 1 on the left shows good but inconsistent predictions when a color should be fully solid, while the right image shows a more solid segmentation result. On the other hand, the right image is too confident with its predictions and does not pick up the stop sign well. Looking at frames 2 and 3, the left images show more spotty predictions, some of which appear in frames 1 and 3 but not frame 2, creating this flickering effect. The right image shows a more solid and fluent prediction result throughout all 3 frames.

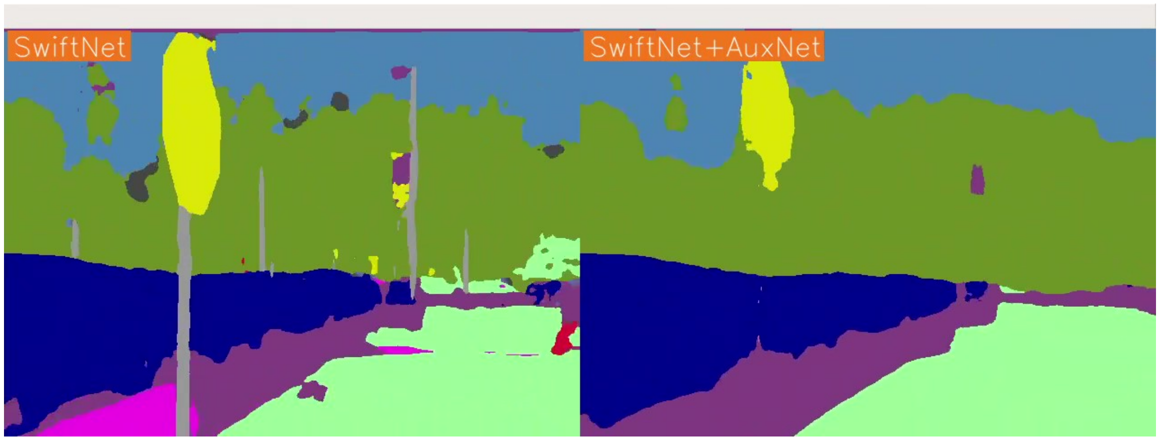
This thesis demonstrated two techniques that can improve inference speed in autonomous perception systems. First, the viability of a robust DNN to perform real-time off-road terrain segmentation was evaluated. Second, a method to improve the image to DNN transfer time using lossy compression was proposed and tested. Due to the focus on faster inference speed, several challenges mentioned above were intensified through trade-offs in architecture and image quality. Future work now focuses on increasing the generalization of off-road terrain and continuing to improve temporal consistency without additional overhead.



(a) Frame 1.



(b) Frame 2.



(c) Frame 3.

Figure 6.2: AuxAdapt results (right) compared with original SwiftNet predictions (left) over a period of 3 frames.

Bibliography

[1] Autonomous vehicle data storage. https://premioinc.com/pages/autonomous-vehicle-data-storage (Date last accessed 10-Nov-2022).

[2] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

[3] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

[4] Joseph Azar, Abdallah Makhoul, Raphaël Couturier, and Jacques Demerjian. Robust iot time series classification with data compression and deep learning. *Neurocomputing*, 398:222–234, 2020.

[5] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius B Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago M Paixao, Filipe Mutz, et al. Self-driving cars: A survey. *Expert Systems with Applications*, 165:113816, 2021.

[6] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[7] Gary Bradski. The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 25(11):120–123, 2000.

[8] Gabriel J Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *European conference on computer vision*, pages 44–57. Springer, 2008.

[9] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1209–1218, 2018.

[10] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[11] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[12] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[13] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[14] J-D Decotignie. Ethernet-based real-time and industrial communications. *Proceedings of the IEEE*, 93(6):1102–1117, 2005.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[16] Mark Everingham, SM Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

[17] Max H Faykus, Bradley Selee, Jon C Calhoun, and Melissa C Smith. Lossy compression to reduce latency of local image transfer for autonomous off-road perception systems. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 3146–3152. IEEE, 2022.

[18] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, and Jose Garcia-Rodriguez. A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing*, 70:41–65, 2018.

[19] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[20] Apoorv Gupta, Aman Bansal, and Vidhi Khanduja. Modern lossless compression techniques: Review, comparison and analysis. In *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pages 1–8. IEEE, 2017.

[21] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[23] Markus Hofmarcher, Thomas Unterthiner, José Arjona-Medina, Günter Klambauer, Sepp Hochreiter, and Bernhard Nessler. Visual scene understanding for autonomous driving using semantic segmentation. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 285–296. Springer, 2019.

[24] Abir Jaafar Hussain, Ali Al-Fayadh, and Naeem Radi. Image compression techniques: A survey in lossless and lossy algorithms. *Neurocomputing*, 300:44–69, 2018.

[25] Peng Jiang, Philip Osteen, Maggie Wigness, and Srikanth Saripalli. Rellis-3d dataset: Data, benchmarks and analysis. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 1110–1116. IEEE, 2021.

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[28] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[29] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[30] Xin Liang, Kai Zhao, Sheng Di, Sihuan Li, Robert Underwood, Ali M Gok, Jiannan Tian, Junjing Deng, Jon C Calhoun, Dingwen Tao, et al. Sz3: A modular framework for composing prediction-based error-bounded lossy compressors. *IEEE Transactions on Big Data*, 2022.

[31] Luning Liu, Xin Chen, Zhaoming Lu, Luhan Wang, and Xiangming Wen. Mobile-edge computing framework with data compression for wireless network in energy internet. *Tsinghua Science and Technology*, 24(3):271–280, 2019.

[32] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[33] Yifang Ma, Zhenyu Wang, Hong Yang, and Lin Yang. Artificial intelligence applications in the development of autonomous vehicles: a survey. *IEEE/CAA Journal of Automatica Sinica*, 7(2):315–329, 2020.

[34] Detlev Marpe, Thomas Wiegand, and Gary J Sullivan. The h. 264/mpeg4 advanced video coding standard and its applications. *IEEE communications magazine*, 44(8):134–143, 2006.

[35] Yujian Mo, Yan Wu, Xinneng Yang, Feilin Liu, and Yujun Liao. Review the state-of-the-art technologies of semantic segmentation based on deep learning. *Neurocomputing*, 493:626–646, 2022.

[36] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[37] Marin Orsic, Ivan Kreso, Petra Bevandic, and Sinisa Segvic. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12607–12616, 2019.

[38] Marin Oršić and Siniša Šegvić. Efficient semantic segmentation with pyramidal fusion. *Pattern Recognition*, 110:107611, 2021.

[39] Papers With Code. Panoptic segmentation], 2023. [Online; accessed March 27, 2023].

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[41] Mizanur Rahman, Mhafuzul Islam, Jon Calhoun, and Mashrur Chowdhury. Real-time pedestrian detection approach with an efficient data communication bandwidth strategy. *Transportation research record*, 2673(6):129–139, 2019.

[42] Mizanur Rahman, Mhafuzul Islam, Cavender Holt, Jon Calhoun, and Mashrur Chowdhury. Dynamic error-bounded lossy compression to reduce the bandwidth requirement for real-time vision-based pedestrian safety applications. *Journal of Real-Time Image Processing*, 19(1):117–131, 2022.

[43] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[44] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[45] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[46] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[47] Khalid Sayood. *Introduction to data compression.* Morgan Kaufmann, 2017.

[48] Bradley Selee, Max Faykus, and Melissa Smith. Semantic segmentation with high inference speed in off-road environments. Technical report, SAE Technical Paper, 2023.

[49] Suvash Sharma, John E Ball, Bo Tang, Daniel W Carruth, Matthew Doude, and Muhammad Aminul Islam. Semantic segmentation with transfer learning for off-road autonomous driving. *Sensors*, 19(11):2577, 2019.

[50] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[51] superannotate. Complete guide to semantic segmentation [updated 2023], 2023. [Online; accessed March 27, 2023].

[52] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[53] Araz Taeihagh and Hazel Si Min Lim. Governing autonomous vehicles: emerging responses for safety, liability, privacy, cybersecurity, and industry risks. *Transport reviews*, 39(1):103–128, 2019.

[54] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. In *The 2005 DARPA grand challenge*, pages 1–43. Springer, 2007.

[55] Suramya Tomar. Converting video formats with ffmpeg. *Linux Journal*, 2006(146):10, 2006.

[56] Abhinav Valada, Gabriel L Oliveira, Thomas Brox, and Wolfram Burgard. Deep multispectral semantic scene understanding of forested environments using multimodal fusion. In *International symposium on experimental robotics*, pages 465–477. Springer, 2016.

[57] Kasi Viswanath, Kartikeya Singh, Peng Jiang, PB Sujit, and Srikanth Saripalli. Offseg: A semantic segmentation framework for off-road driving. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 354–359. IEEE, 2021.

[58] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, Eftychios Protopapadakis, et al. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[59] G.K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992.

[60] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364, 2020.

[61] Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell. Understanding convolution for semantic segmentation. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 1451–1460. Ieee, 2018.

[62] Ruijun Wang, Liangkai Liu, and Weisong Shi. Hydraspace: Computational data storage for autonomous vehicles. In *2020 IEEE 6th International Conference on Collaboration and Internet Computing (CIC)*, pages 70–77. IEEE, 2020.

[63] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[64] Maggie Wigness, Sungmin Eum, John G Rogers, David Han, and Heesung Kwon. A rugd dataset for autonomous navigation and visual perception in unstructured outdoor environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5000–5007. IEEE, 2019.

[65] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341, 2018.

[66] Yizhe Zhang, Shubhankar Borse, Hong Cai, and Fatih Porikli. Auxadapt: Stable and efficient test-time adaptation for temporally consistent video semantic segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2339–2348, 2022.

[67] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[68] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.

[69] Mingmin Zhen, Jinglu Wang, Lei Zhou, Tian Fang, and Long Quan. Learning fully dense neural networks for image semantic segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9283–9290, 2019.