Lossy Compression to Reduce Latency of Local Image Transfer for Autonomous Off-Road Perception Systems

Max H. Faykus, Bradley Selee, Jon C. Calhoun and Melissa C. Smith
Holcombe Department of Electrical and Computer Engineering - Clemson University, Clemson, SC, USA
Email {mfaykus, bselee, jonccal, smithmc}@clemson.edu

Abstract*—Autonomous vehicles greatly rely on their perception system for navigation. Semantic segmentation provides a much better understanding of a vehicle's surroundings than object detection. Unfortunately, complete image segmentation comes at a higher computational cost than object detection, which complicates developing a real-time perception system using semantic segmentation. Perception systems contain other bottlenecks too, and are not only limited by their deep learning model. An inherent amount of latency exists in data transfer, specifically through Ethernet. A vehicle's camera feed must be transferred to an edge device for image processing as part of the autonomous driving decision-making process. This study investigates decreasing image transfer time by using various levels of JPEG compression as well as further understanding how compression affects the accuracy of semantic segmentation. Additionally, as most autonomous driving research focuses on urban environments, we look to explore autonomous unmanned ground vehicles (UGVs) in the off-road space by using the Rellis-3D dataset. We train and evaluate SwiftNet, a state-of-the-art semantic segmentation model, at different JPEG compression ratios and identify the accuracy. The transfer time of these different compression ratios is tested on three images. Results show a continual decrease in accuracy occurs as the compression ratios increase. When training SwiftNet on the train set with no compression, the highest compression ratio of 16.96 achieves a mean intersection over union (mIoU) score of 67.9% compared to the baseline achieving 78.9% mIoU. There is an increase in the accuracy of the higher compression ratios by training SwiftNet on the corresponding compression ratios; the highest compression ratio reaches 74.9% mIoU. Lastly, we notice a positive transfer speedup of these higher compression ratios when inducing JPEG compression in all transfer scenarios: (a) 1870 images (b) 10 images, (c) 1 image. Each scenario has a speedup of $1.18\times$, $1.14\times$, and $1.06\times$, respectively.

***Index Terms*—Semantic Segmentation, Lossy Compression, Off-Road Vision, Data Transfer, Autonomous Driving**

I. Introduction

Research in self-driving cars has made great strides, primarily due to the rise of deep learning (DL) techniques [1]. The perception system in autonomous vehicles remains one of the most critical components so the vehicles can understand their surroundings. Vision algorithms like object detection [2], and image segmentation [3] assist in the driving and decision-making of autonomous vehicles. Compared to object detection, semantic segmentation provides a much better understanding of the environment, but at a significant computational cost, [4], [5]. Specifically, a trade-off between high segmentation accuracy and high inference speed occurs in deep neural networks (DNN) [5]–[7]. This creates an issue because autonomous vehicles must make decisions exceptionally quick but be accurate enough to make the correct decision. 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 1. Within this transfer, some latency exists [8]. Compressing the images from the camera feed can reduce the latency between this data transfer [9], [10]. Autonomous vehicles generate up to 40 TB of data during daily usage [11]. High-resolution cameras and sensors can generate ≈4TB+ of data per day [12]. Without compression, processing and storing this data would require extensive hardware.

Most autonomy research explores well-structured environments, such as urban areas with roads and street signs [13]. Unfortunately, little research has investigated autonomy in an off-road setting; this can be seen from the absence of research investigating this domain and the scarcity of labeled, off-road datasets [14], [15]. Similar to self-driving cars, perception systems are critical for autonomy in unmanned ground vehicles (UGVs), and other off-road vehicles [16]. Similarly, the entire process must be accomplished in real time so the vehicle can make correct decisions while traveling at high speeds. Even though off-road and urban environments contain vastly different terrain, many of the same latency bottlenecks occur in the perception system.

This research aims to accomplish two objectives. First, to reduce local image data transfer latency using JPEG compression. Compression reduces the amount of data sent 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. Second, investigate the semantic segmentation accuracy when training and evaluating the compressed images. This explores the effects of lossy data reduction on training and evaluating a Deep Neural Network (DNN) model.

Our contributions include the following:

- 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.

- A unique and complex perspective by considering the perception of autonomous vehicles in off-road terrain rather than urban environments.

II. Background

Autonomous vehicles generate a large amount of data, usually between 20 TB and 40 TB during daily usage [11]. The camera alone generates data at a rate of 20 to 60 Mbps [11], although this can vary highly based on the image quality. Scene understanding and obstacle sensing require large amounts of camera data to process. Scene understanding needs to perform feature extraction, and template matching must be able to identify signs and signals during driving. At the same time, obstacle sensing requires processing camera data for edge detection, thresholding, object detection, and segmentation. [17].

When working with camera data, JPEG [18] is a standard in lossy image compression. JPEG uses a Discrete Cosine Transform (DCT) based method, in 8×8 pixel blocks running through the image [18]. DCT compresses by separating the image into parts that differ by frequencies, with less important frequencies being discarded and only critical frequencies being used in decompression [19]. Due to its wide use, it serves as a good compression scheme for reducing image sizes and reducing bandwidth transmission requirements.

III. Related Work

Several studies exist which demonstrate the use of data compression to reduce latency for a variety of applications. In [20], 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 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 [9], [10] tries to minimize the network bandwidth requirements using H.264 video compression [21] 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× other state-of-the-art strategy. The current study expands upon this research by instead of utilizing object detection with YOLOv3 [2] in an urban environment; we investigate semantic segmentation using SwiftNet [22] in an off-road setting. Furthermore, this study 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 [23], 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) [24]. 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.

IV. Methods

A real-time perception system in autonomous vehicles can be described similarly to Figure 1. Reducing the latency from the camera feed to the video processing unit speeds up the overall system time. Our setup first takes in camera data (to simulate this pipeline, the Rellis-3D dataset images are used) and performs lossy JPEG compression with FFMPEG. SCP transfers 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 a deep neural network (DNN) to classify the terrain from the compressed image data.

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.

$$T_{send} = \frac{Size}{BW} \tag{1}$$

$$T_{comp\_send} = \frac{Size}{T_{Comp}} + \frac{Comp\_Size}{BW} \tag{2}$$

$$T_{comp\_send} < T_{send} \tag{3}$$

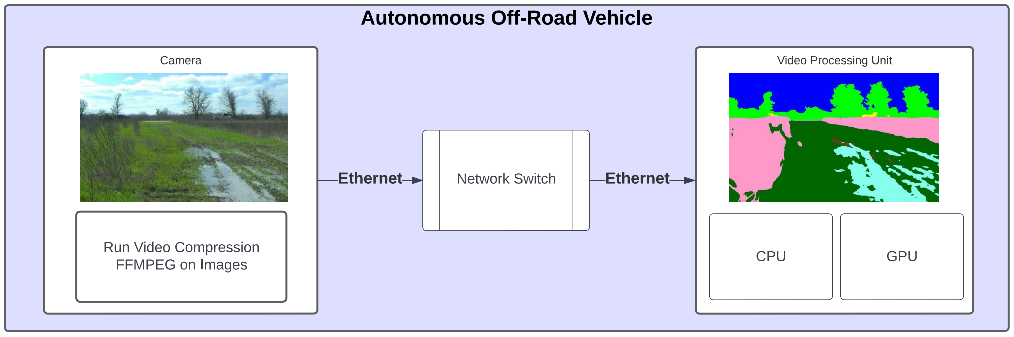
The following section describes our approach to implementing compression in a real-time vision system. First, we discuss the compression method and dataset for this experiment. Next, we briefly explain semantic segmentation, the DNN used, and the segmentation experiments performed. Finally, we demonstrate our data transfer setup.

A. Image Compression

With the FFMPEG tool [25], we use JPEG compression [18] to compress the train and test set into 31 different compression levels. We use compression level to describe the image quality tuner value on FFMPEG. Compression levels correspond to an average compression ratio shown in Figure 2, 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.

B. Semantic Segmentation

For semantic segmentation, we use SwiftNet [22], [26], a state-of-the-art deep learning model designed for high inference speed and accuracy on high-resolution images. We utilize SwiftNet because it favors high inference speed, which will more likely be found in a real-time perception system compared to other DNNs like Deeplabv3+ [3] which favor higher accuracy. We implemented, trained, and evaluated SwiftNet using the PyTorch [27] deep learning framework. When describing the accuracy of a semantic segmentation



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

Fig. 1: 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).

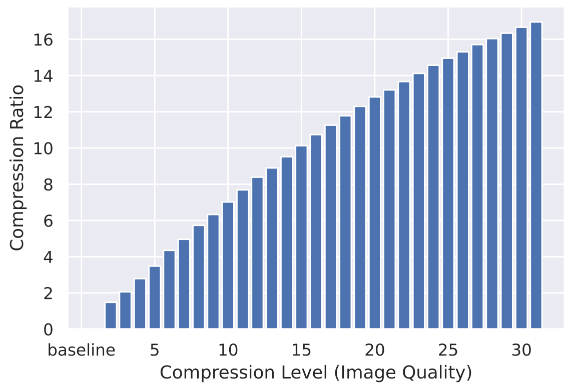


Fig. 2: FFMPEG image quality (compression level) to compression ratio. The baseline represents level 1, with no compression.

model, the primary metric used is the Jaccard Index [28], also known as the intersection over union (IoU), shown in equation 4. 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 5.

$$IoU_{class} = \frac{TP}{TP + FP + FN} \tag{4}$$

$$mIoU = \frac{\sum IoU_{class}}{n\_classes} \tag{5}$$

Two experiments regarding semantic segmentation accuracy are performed on compressed images. First, we train a baseline model on the Rellis-3D train set without compression. Then, we evaluate the mIoU at each compression level using the test set to observe how compression affects the model's accuracy. Secondly, we train a model at six different compression levels (5, 10, 15, 20, 25, and 30) and evaluate the mIoU at the corresponding compression level. For example, if the model is trained on the train set with level 15 compression, we evaluate this model on the level 15 compressed test set.

C. Data Transmission

To simulate image transfer in an autonomous vehicle, we set up a local Ethernet system described in Figure 1. To verify the communication and find the theoretical limits of our system, we measure the bandwidth from our local computer to the Jetson Xavier using iPerf3. We use Secure Copy Protocol (SCP) to send three sets of compressed images through our system for our image transfer experiments. We time the transfer between the two local hosts for each image set at different compression levels.

V. Experimental Results and Discussion

This section covers the results from the experiments performed in this research and an interpretation of the outcomes. First, we explore the semantic segmentation accuracy of the SwiftNet model at various compression levels. This includes the baseline model evaluated at multiple compression levels and the model trained at six different compression levels. Second, we time the latency of Ethernet transfer, the overhead of compression, and overall end-to-end segmentation.

A. 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. This research considers autonomy in unmanned ground vehicles (UGVs); thus, an off-road dataset, Rellis-3D [29], was utilized. Rellis-3D is an extensive off-road dataset designed for semantic segmentation, containing 6234 labeled 1920×1200 images. We create a train and test set by randomly splitting 70% of the data into the train set and using the remaining 30% for the test set. For this experiment, we train the SwiftNet model using the train set with no compression. We follow the same training protocol as [22]; most notably, we train for 250 epochs and evaluate the epoch which achieved the best validation

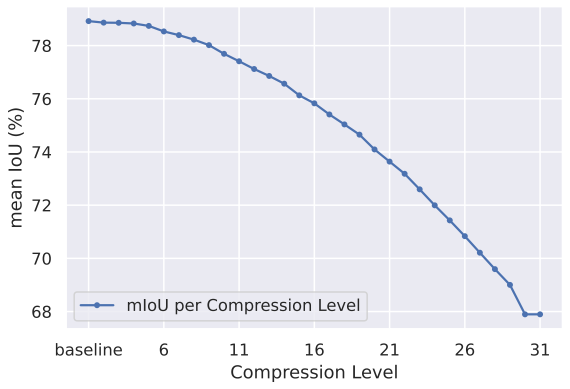


Fig. 3: Evaluation of mIoU at each compression level using a model trained only on the baseline dataset (no compression).

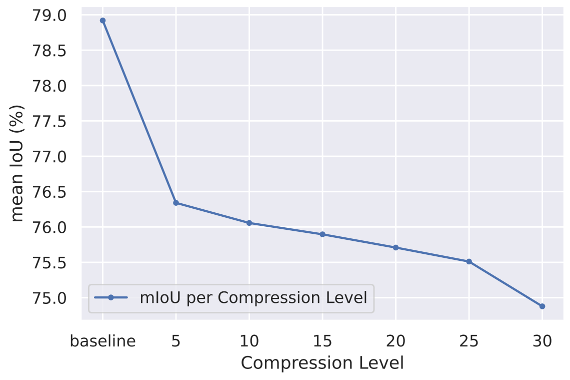


Fig. 4: Evaluation mIoU on six different image quality levels, trained on its respective image quality level.

accuracy. Figure 3 illustrates the mIoU for our 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.

B. Training with Various Compression Levels

Due to data loss from the JPEG lossy compression, a deep learning model can misclassify object boundaries if the pixel has high enough distortion. From our 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, we only trained six different compression levels for 100 epochs. Figure 4 shows a significant drop off in accuracy compared to Figure 3 from compression levels 2–6, but from level 6–30, only a 2% mIoU drop exists. Furthermore, compression levels 20–30 in figure 4 show a significant improvement in mIoU compared to Figure3. 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 increasing end-to-end system speed.

Looking at figure 4 and 5 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% mIoU decrease.

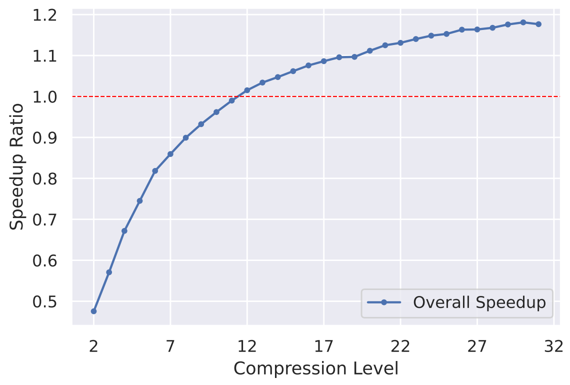


Fig. 5: Overall transmission speedup of the entire compressed Rellis-3D test set. Values above 1.0 represent a positive speedup, and values below 1.0 did not benefit from compression.

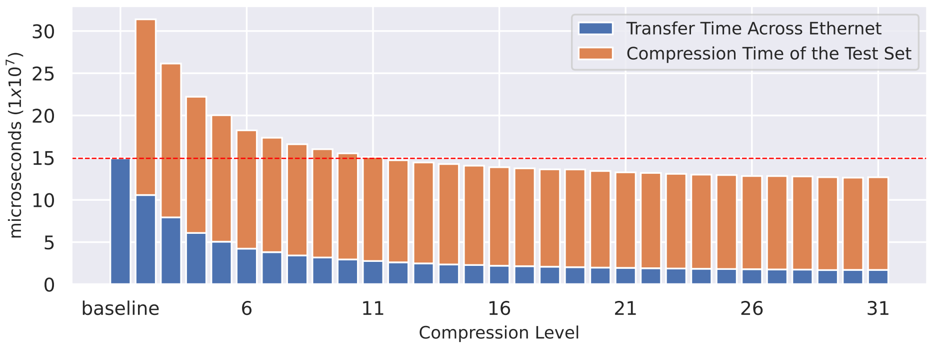
C. Theoretical Bandwidth Testing

We performed a ping-pong test to measure the bandwidth between the local machine and a Jetson Xavier connected with an Ethernet cable going through a switch. This test was implemented using iPerf3 in a bidirectional test. The iPerf3 server is set up on the Jetson Xavier, and the client is connected through an Ethernet.

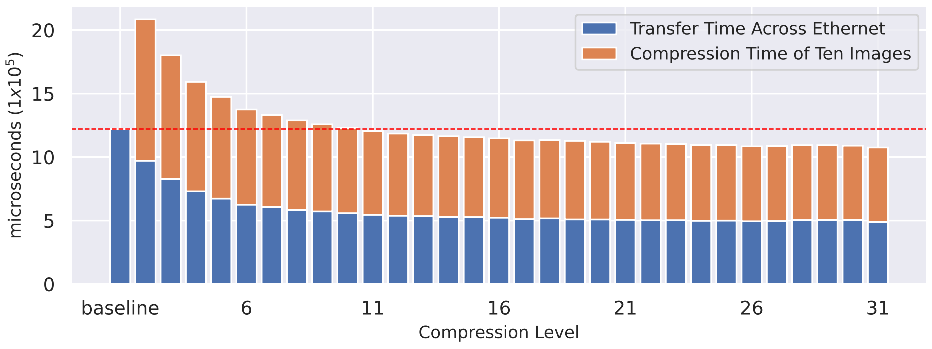
Running this test, we were able to achieve a Bandwidth of 0.09 Gbits/second, and this test was run for 120 seconds, reporting the rate every 30 seconds. This test was performed over the TCP port 5001. For this test, the total amount of data transferred over each 30-second interval is 1.29 GB.

D. Transfer, Compression, and End-to-End Timing

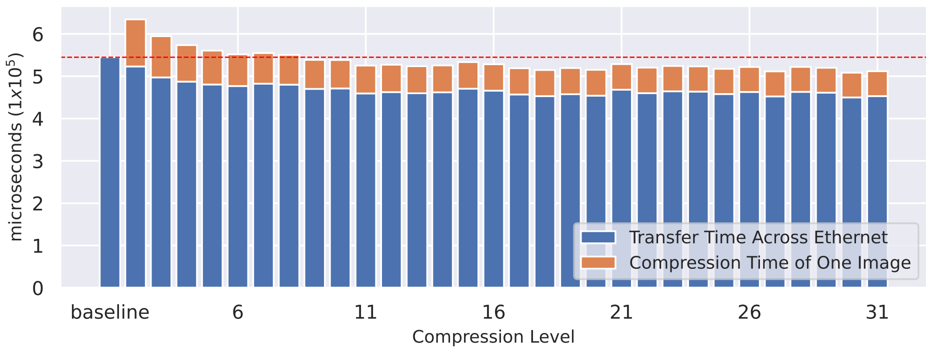
Using SCP, we measure the transfer and compression time of the Rellis-3D test set (1870 images), ten images, and one image at every compression level. The compression time contains a summation of per-image compression. Figure 6 provides the transfer and compression times for the three sets



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



(b) Transfer batch size of 10 images.



(c) Transfer batch size of 1 image.

Fig. 6: 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.

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 time.

In Figure 6(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 $1.18\times$. While Figure 6(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 6(b) shows a positive speedup beginning at compression level 11 and continuing to gradually increase to compression level 31. The most significant speedup, once again, occurs at level 31 by $1.14\times$. 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 4 shows a high mIoU score for the compression levels at which the speedup occurs. Finally, while Figure 6(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 6(c) had this overhead on every image in the dataset, while 6 was able to send 10 images before introducing more overhead. 6(a) only required this setup once before transferring the dataset.

VI. Conclusion

This study compresses camera data at various compression ratios to reduce image transfer latency over Ethernet. As a result, the image transfer time to the perception system improves while the mIoU of semantic segmentation on compressed images remains high but steadily decreases; this decrease can be mitigated by training SwiftNet on the corresponding compression levels to retain higher levels of mIoU. These results suggest that the more images compressed and transferred, the higher the overall speedup. Out of the three timing experiments performed, compressing and sending 10 images appears the most logical for real-time systems. Our approach achieves a speedup in transfer time for all three scenarios, represented by Figure 6. Scenario (a) achieved the highest speedup of 1.18 at level 31, scenario (b) achieved the highest speedup of 1.14 at level 31, and scenario (c) achieved the highest speedup of 1.06 at level 30.

This work can be continued by expanding to various datasets and NN models. This can also be tested on different lossy compression algorithms, showing the effects of varying levels of compression ratios and image quality. This could be extended for self-driving cars in urban environments for future work since this research primarily focuses on autonomous UGVs in the off-road setting. Additionally, other real-time semantic segmentation models and compression algorithms can be explored. We focus on improving the camera-to-perception time, but other bottlenecks exist in autonomous vehicles which can further be approached.

Acknowledgments

DISTRIBUTION A. Approved for public release; distribution unlimited. OPSEC#7078

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 and SHF-1943114.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, 2015.

[2] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[3] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[4] M. Hofmarcher, T. Unterthiner, J. Arjona-Medina, G. Klambauer, S. Hochreiter, and B. Nessler, "Visual scene understanding for autonomous driving using semantic segmentation," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pp. 285–296, Springer, 2019.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[6] Y. Ma, Z. Wang, H. Yang, and L. Yang, "Artificial intelligence applications in the development of autonomous vehicles: a survey," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 2, pp. 315–329, 2020.

[7] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez, "A survey on deep learning techniques for image and video semantic segmentation," *Applied Soft Computing*, vol. 70, pp. 41–65, 2018.

[8] J.-D. Decotignie, "Ethernet-based real-time and industrial communications," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1102–1117, 2005.

[9] M. Rahman, M. Islam, J. Calhoun, and M. Chowdhury, "Real-time pedestrian detection approach with an efficient data communication bandwidth strategy," *Transportation research record*, vol. 2673, no. 6, pp. 129–139, 2019.

[10] M. Rahman, M. Islam, C. Holt, J. Calhoun, and M. 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*, vol. 19, no. 1, pp. 117–131, 2022.

[11] R. Wang, L. Liu, and W. Shi, "Hydraspace: Computational data storage for autonomous vehicles," in *2020 IEEE 6th International Conference on Collaboration and Internet Computing (CIC)*, pp. 70–77, IEEE, 2020.

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

[13] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

[14] M. Wigness, S. Eum, J. G. Rogers, D. Han, and H. 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)*, pp. 5000–5007, IEEE, 2019.

[15] P. Jiang, P. Osteen, M. Wigness, and S. Saripalli, "Rellis-3d dataset: Data, benchmarks and analysis," in *2021 IEEE international conference on robotics and automation (ICRA)*, pp. 1110–1116, IEEE, 2021.

[16] K. Viswanath, K. Singh, P. Jiang, P. Sujit, and S. Saripalli, "Offseg: A semantic segmentation framework for off-road driving," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pp. 354–359, IEEE, 2021.

[17] M. Puškár, M. Fabian, J. Kadarova, P. Blišt'an, and M. Kopas, "Autonomous vehicle with internal combustion drive based on the homogeneous charge compression ignition technology," *International Journal of Advanced Robotic Systems*, vol. 14, no. 5, p. 1729881417736896, 2017.

[18] G. Wallace, "The jpeg still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[19] R. T. Nageswara and K. D. Srinivasa, "Image compression using discrete cosine transform," *Computer Sciences and Telecommunications*, no. 3, pp. 35–44, 2008.

[20] L. Liu, X. Chen, Z. Lu, L. Wang, and X. Wen, "Mobile-edge computing framework with data compression for wireless network in energy internet," *Tsinghua Science and Technology*, vol. 24, no. 3, pp. 271–280, 2019.

[21] D. Marpe, T. Wiegand, and G. J. Sullivan, "The h. 264/mpeg4 advanced video coding standard and its applications," *IEEE communications magazine*, vol. 44, no. 8, pp. 134–143, 2006.

[22] M. Oršić and S. Šegvić, "Efficient semantic segmentation with pyramidal fusion," *Pattern Recognition*, vol. 110, p. 107611, 2021.

[23] J. Azar, A. Makhoul, R. Couturier, and J. Demerjian, "Robust iot time series classification with data compression and deep learning," *Neurocomputing*, vol. 398, pp. 222–234, 2020.

[24] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[25] S. Tomar, "Converting video formats with ffmpeg," *Linux Journal*, vol. 2006, no. 146, p. 10, 2006.

[26] M. Orsic, I. Kreso, P. Bevandic, and S. 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*, pp. 12607–12616, 2019.

[27] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” *Advances in neural information processing systems*, vol. 32, 2019.

[28] M. Everingham, S. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge: A retrospective,” *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015.

[29] P. Jiang, P. Osteen, M. Wigness, and S. Saripalli, “Rellis-3d dataset: Data, benchmarks and analysis,” 2020.