10-14-2022 10:45 AM

Potential of Vision Transformers for Advanced Driver-Assistance Systems: An Evaluative Approach

Andrew Katoch, *The University of Western Ontario*

Supervisor: Bauer, Michael A., *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Computer Science

Abstract

In this thesis, we examine the performance of Vision Transformers concerning the current state of Advanced Driving Assistance Systems (ADAS). We explore the Vision Transformer model and its variants on the problems of vehicle computer vision. Vision transformers show performance competitive to convolutional neural networks but require much more training data. Vision transformers are also more robust to image permutations than CNNs. Additionally, Vision Transformers have a lower pre-training compute cost but can overfit on smaller datasets more easily than CNNs. Thus we apply this knowledge to tune Vision transformers on ADAS image datasets, including general traffic objects, vehicles, traffic lights, and traffic signs. We compare the performance of Vision Transformers on this problem to existing convolutional neural network approaches to determine the viability of Vision Transformer usage.

Keywords: Vision transformers, image recognition, ADAS

Summary for Lay Audience

One component of Autonomous Driving, Advanced Driving System Assistance Systems (ADAS), are vehicle systems designed to improve driving ability and road safety. These technologies can include Anti-Lock Braking Systems and lane departure warning systems. These systems often have to collect information about the traffic environment, including the presence of traffic objects such as vehicles, pedestrians, traffic lights, and traffic signs. The collection is often done through computer vision, collecting visual information through cameras attached to the vehicle. A common way of parsing this visual information to detect and classify these traffic objects is through machine learning models. Machine learning models differ from traditional computer algorithms as they do not need to be explicitly programmed. Instead, they learn from the data given to them at training time and make decisions based on the information. In this case, machine learning models can learn from traffic image data to make predictions presence and class of traffic objects.

In this thesis, we evaluate a set of pre-trained Vision Transformer models made by Google. Vision Transformers are a new, popular type of machine learning model applying a mechanism called self-attention. Self-attention mechanisms can learn from and form relations between any pair of points in a data sequence. Vision transformers do not compare every pixel but split images into patches for more realistic computer power and memory cost. These patches are arranged into a linear sequence of vectors, and the Vision transformer trains by finding relations in these patches of pixels.

Our research shows that Vision Transformers are competitive with existing Convolutional Neural network models when first pretrained on a large dataset of images and then adjusted to train on a smaller, domain-specific dataset. We apply this concept to an image dataset of vehicles, pedestrians, traffic lights, and traffic signs. We find that classification accuracy when predicting the traffic object in unseen images is higher than classification accuracy from prior research applying Convolutional Neural Networks to the same datasets.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 An Overview

Every year there are 20-50 million people injured in traffic accidents, of which over 1 million are fatal [1], with this number increasing yearly. These injuries are now the global killer of individuals between 5 and 29 years of age, despite there being technology proven to reduce such accidents [2]. This technology is called Advanced Driver Assistant Systems (ADAS), a set of systems designed to enhance driver safety and reduce human error.

Human error accounts for the majority of road accidents [3]; typically because of a loss of alertness due to impaired driving and/or fatigue. Besides impairment, other underlying issues like fatigue are hard to determine in the case of fatalities. Increased smartphone usage has also presented another distraction for drivers. The majority of accidents happen in low-income countries [2], where safety information is often less accessible or laws are not adequately enforced. The global increase in accidents and human error coupled with the effectiveness of ADAS shows a need for the advancement and propagation of such technology.

Autonomous vehicles, vehicles intended to drive without human assistance, are gen-

erally separated into six levels of automation. Level 0 represents no automation, relying wholly on human drivers, and level 5 represents full automation, requiring no driver input [4]. The first three levels, up to level 2, partial driving automation, can be considered the limit of ADAS, with ADAS such as cruise control in level 1 and lane keeping and adaptive cruise control in level 2. Traffic object recognition, the focus of this thesis, is essential for ADAS and could reduce error significantly.

1.2 Problem

Traffic accidents are common, and the issues are generally attributed to human error. Safety is a significant focus of many automobile manufacturers, and ADASs are a common selling point for autonomous vehicle manufacturers [5]. Several tools exist, such as Anti-lock Braking System, backup cameras and state-of-the-art automatic vehicle maneuvering. External sensory networks such as cameras, radar, and LiDAR are enabling many innovations [6]. Examples include Lane Departure Warnings, Rear Cross-Traffic Alerts, and Emergency Brake Assist when approaching obstacles in front.

Artificial Intelligence (AI), specifically Machine Learning (ML), also plays a vital role in ADAS, namely for Computer Vision (CV) ADAS. Traffic object recognition, specifically Traffic Sign recognition, Vehicle recognition and Pedestrian recognition [5], are important CV-based ADAS tasks, generally done via object detection and then classification. In this thesis we will use the words classification and recognition interchangeably. Traffic Sign recognition can process information before the driver can see it, enabling more informed driving. This technology can also be extended to identifying other vehicles, pedestrians, traffic lights and other traffic objects in every direction, allowing for a safer and simplified driving experience. While notable developments exist in this field, there is still room for improvement. Modern techniques reach 96% classification accuracy on their respective datasets [7], though there is still work to be done in production vehicles. In [8], we can see

that in perfect conditions, about 5% of signs are misread, but when signs were artificially altered to test the robustness of these systems, accuracy could drop to as low as 62% from 99%. There is work to be done on the resistance of Traffic Sign recognition to possible real-world image corruption.

1.3 Contribution

This thesis assesses the effectiveness of Google's Vision Transformers (ViT) [9], for the problem of traffic image classification. The ViT is a modification of the Transformer [10] deep-learning architecture for images, originally introduced for Natural Language Processing tasks based on a network mechanism called Self-Attention.

ViTs have quickly become popular in the Computer Vision space [11], and have also been shown to be significantly more robust to image permutations than Convolutional Neural Networks [12]. This thesis examines the potential use of ViTs in the ADAS space through exploration and experimental assessment of the performance of ViTs on classes of images arising in driving scenarios and compares their performance to previous work utilizing the same data [7]. Specifically, when fine-tuned on a pre-cropped traffic image dataset from RoadLAB, ViTs show significant image classification accuracy improvements compared to existing Convolutional Neural Network (CNN) methods used by RoadLAB in [7]. The CNN results were good, but can still be improved upon; a comparison of the original ViT model to these CNN methods will show the possible potential for image transformers for ADAS systems. The results show considerable potential for the application of ViTs to the problem of ADAS traffic detection.

1.4 Thesis Organization

This thesis is organized as follows:

In chapter 2, we examine preliminary and recent work in the field of ADAS TSA and related image classification topics. Chapter 3 explains the related work and the proposed ViT architecture. Chapter 4 presents the image database used, the methods used to choose the ViT models and hyperparameters, the performance evaluation criteria, and the experiment results. Chapter 5 discusses the conclusion, shortcomings, and possible future work.

Chapter 2

Related Work

Advanced Driver Assistance Systems (ADAS) have been a subject of interest for researchers [13] and automobile manufacturers [8] alike. The highest form of ADAS would be a level 5 in autonomy or a fully self-driving vehicle. A level 5 rating requires a multi-faceted perception of the traffic environment, including vision. Computer vision (CV) can be separated into traditional and machine learning algorithms, the latter of which are prevalent in the automotive industry [6]. In visual image recognition, elements of the visual driving environment must be classified to make high-level decisions after the detection stage. To this end, image classification is a pertinent problem.

For the problem of sensing other objects and vehicles, there are four modes of sensing for vehicle detection based on sensing electromagnetic radiation or sound waves. As explained in [14], there exists vehicle hardware to detect millimetre-wave (MM wave) radio signals, LiDAR, a 600-1000 nanometer-wave laser, and vision based on the visible light spectrum. Radar and LiDAR work by propagating signals of their respective frequency and analyzing the subsequently reflected energy. Radar tracks object based on motion, and usage is common in other ADAS applications, but it has a narrow field of view and struggles with wide-open traffic. LiDAR has become increasingly popular and common in detecting traffic in ADAS, segmenting regions of interest and classifying based on size or motion. The fourth

type, ultrasonic sensors, uses high-frequency sound waves to measure nearby, slow-moving or stationary objects, though they are easily susceptible to the environment [15].

Before images can be classified, they must be detected by the hardware. Generally, the visual detection stage is split into non-machine learning and machine learning approaches. Of the ML approaches, Viola and Jones proposed a sliding-window framework with Adaboost classifiers [16] and the use of stochastic Support Vector Machines (SVM). These SVMs use Histograms of Oriented Gradients [17] and Scale-Invariant Feature Transform [18] as features. As CNNs progressed in the 2010s, they became common in the field [19, 20]. The family of Region-based CNNs, R-CNNs, quickly became popular, as well as the regression-based families of the CNNs: You Only Look Once (YOLO) [21] and Single Shot Multibox Detector (SSD) [22].

In this section, we discuss related works in general image classification. Each subdomain of ADAS researched for this thesis, and we discuss the background of Vision Transformers.

2.1 General Image Classification

Machine Learning "is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without being explicitly programmed" [23]. Image classification is a complex topic, and the usage of machine learning techniques has been studied extensively. The "learning" part of Machine Learning can be divided into three categories: supervised learning, where labelled data is used to evaluate model training and prediction; unsupervised learning, where the model finds relations and patterns in data; and reinforcement learning, where the model is an agent trying to solve a problem and is reinforced by its progress toward that reward [23]. Here we will focus on unsupervised learning algorithms, the method most commonly used in CV. Some surveys can be found in [24], [25].

Before advancements in machine learning, image recognition did not always yield good

performance. Systems had to be manually programmed based on prior knowledge [26]. From 2010 to 2017, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) began as an annual visual recognition competition on the ImageNet benchmark and extensive visual object database. It was a notable representation of the image recognition state-of-the-art [27]. Lin et al. won in 2010 using an SVM with two non-linear coding representations via Scale-Invariant Feature Transform and Local Binary Patterns (LBP) [28]. An SVM won in 2011 again [29], but in 2012, a CNN, AlexNet, won [30], kicking off the deep-learning-based popularity. A deep-learning-based model has won each year since then, leading to the domination of deep learning we see to this day (where depth refers to NNs with one or more hidden layers) [25] [31].

CNNs have existed since the 1980s but experienced explosive growth in the 2010s because of more efficient algorithm implementations and Graphics Processing Unit (GPU) hardware advancements. A notable advancement in NN technology is the Visual Geometry Group's VGG model, which implemented a network twice as large as AlexNet and ReLU non-linearity for the nodes [32]. Another notable advancement is Microsoft's ResNet implementing residual learning via skip connection (identity mapping around weight layers), allowing gradients to flow through a network unmitigated [33]. CNNs remain dominant in the scene, but ViTs have recently made strides, [31, 11].

2.2 ADAS Image Classification

In the ADAS spaces, much of the focus of image recognition falls into three domains: traffic light recognition, traffic sign recognition, and vehicle recognition. This work focuses only on classifying traffic lights, traffic signs, and vehicles, so the research dealing with the recognition of these objects will be the focus of this review.

2.2.1 Traffic Light Recognition

Several methods have been studied for Traffic light recognition; a survey can be found in [34]. Lindner et al. notably used CNNs in 2004 [35], achieving 90% accuracy albeit predicted offline at 10Hz; this technique was used in the 2014 Mercedes-Benz S-Class [36]. Shen et al. similar or better results by modelling hue and saturation according to Gaussian distributions [37]. SVMs with Histograms of Oriented Gradients were used by [38], [39]. Notably, CNNs have become the most widely used deep learning architecture [40]; [41] applied YOLO-v1, and Jensen et al. [42] applied the YOLO-9000 to the LISA traffic light dataset. Liu, Yan and Qi [43] applied the recent capsule CNNs [44] on the traffic light dataset, TL_Dataset. Work done in the same lab as this thesis achieved an average of 96.2% accuracy, with up to 99% accuracy on some classes, on a dataset generated in-house [7].

2.2.2 Traffic Sign Recognition

Prior approaches for traffic sign recognition included template matching, which achieved good performance on guiding signs (depicting routes and roads) and 46.5% on speed limit sign recognition in 2000 [45]. Similar to traffic light recognition, traffic sign recognition studies have employed SVMs, such as [46], [47] and [48] with the latter two achieving around 95% accuracy. SVMs were compared with Nearest Neighbor Classifiers, Iterative Nearest Neighbor Classifiers, and Sparse Representation-based Classifiers [49], achieving over 95% average accuracy on the German Traffic Sign Recognition Benchmark (GTSRB) [50] and the Belgium Traffic Sign Dataset for Classification (BTSC) [51]. Similar to traffic light recognition, CNNs have been used for traffic sign recognition before the CNN renaissance; in 1997, a CNN was used for the classification of low resolution (30 x 30) photos in [52], with high accuracy (some over 90%) for the time, on simple street signs. Notably, in 2004, a spatiotemporal attention network (although a different implementation to the focus of

this thesis) achieved 85% accuracy on simple signs as well [53]. Neural networks have since grown dramatically in popularity, like with traffic lights. Multiple NNs were used in [54] (one for each sign shape and colour), multi-scale CNNs were used in [55], a CNN trained with Hinge Loss Stochastic Gradient Descent was used in [56], and notably, Generative Adversarial Network in tandem with a CNN in [57]. A CNN achieved 99.7% on the Swedish Traffic Signs Dataset (STSD) [58].

Haque et al. recently proposed DeepThin, a lightweight CNN due to being deep but thin (many layers but few nodes), making the act of training without a GPU feasible due to the small number of operations per layer and image size. DeepThin achieved over 99% accuracy on the GTSRB and 98% accuracy on the BTSC. When pre-training on GTSRB and fine-tuning on BTSC or vice versa, accuracy increased to 98%. Dewi et al. [59] trained Yolov3 [60] and Yolov4 [61] on images synthesized by various modern GANs that were trained on a self-made Taiwan prohibitory sign dataset. Classification accuracy was over 99% depending on the GAN, with the best being YOLOv4 on the Least Squares Generative Adversarial Networks (LSGAN) [62], achieving 100% accuracy on the 3 of the four classes. In 2022, Zheng and Jiang trained and applied five vision transformers on the German Traffic Sign Recognition Benchmark, the Indian Cautionary Traffic sign dataset, and the CSUST Chinese Traffic Sign Detection Benchmark [63] to compare with CNNs. Accuracy was noticeably worse, from around 2% worse up to 12.81% worse. They suggest two possible methods for improvement: pre-training the transformer on a larger dataset, fine-tuning or more effective internal structures, such as convolution. Lastly, from the same lab as this thesis, [7] achieved up to 96.1% accuracy on a dataset generated by the lab.

2.2.3 Vehicle Recognition

Vehicle classification as a field of study was not separate from detection initially. As seen in this survey [64], it was not until the late 1990s to early 2000s that some machine learning

techniques, including neural networks, began to differentiate vehicles. Kalinke et al. fed Hausdorff distances of the vehicles' Local Orientation Coding (LOC) against archetypal models of cars and trucks into a NN [65]. Handmann et al. used a similar method, with a histogram of LOC fed into a NN [66].

As shown in [67], some other techniques included Automatic Traffic Surveillance Systems, which checked for size and linearity of objects; Gupte et al. checked vehicle length and height width to identify trucks and non-trucks, with 90% classification accuracy [68]. Hsieh et al. used size based on lane width and the linearity of vehicles from fixed perspectives to identify cars, minivans, van trucks (including buses), and trucks to around 82-84% accuracy [69]. Avery et al. used lengths based on a fixed perspective for 97.5% accuracy on shorter vehicles and 91.9% accuracy on trucks and longer vehicles [70]. Another technique is using a partial Gabor filter bank, with a recognition rate up to 95% for sedans, vans, hatchbacks, buses and trucks on a database of 1196 images [71]. Lastly, Kafai and Bhanu applied a hybrid dynamic Bayesian network to classify sedans, pickup trucks, SUVs/minivans, and unknown vehicles with a classification rate of 97.63% on a database of 169 videos [72].

Recently, more machine learning methods have become common. Won [73] shows some relatively recent methods. Chen et al. used an SVM for classifying motorcycles, cars, vans, buses and unknown, with a classification accuracy of 94.6% [74]. Karaimer et al. employ an SVM with HOG features for gradient-based classification, combined with the kNN (k Nearest Neighbors) algorithm for classification on a dataset of 124 cars, 104 vans and 48 motorcycles with an accuracy of 96.5% [75]. Mithun et al. used virtual detection lines (MVDLs)-based to make time spatial images fed into a kNN algorithm to classify motorbikes, rickshaws, autorickshaws, cars, jeeps, covered vans, and busses in Dhaka, Bangladesh, and Suwon, South Korea, for an accuracy up to 91% [76]. Deep learning techniques are also prevalent. Dong et al. applied a semisupervised CNN - unsupervised for detection and supervised for classification - on the BIT-Vehicle dataset [77] and one in

[78]. The CNN gave classification accuracies of 88.1%, and 89.4%, respectively. Huttunen et al. fed 6555 vehicle images from Finland into a CNN, classifying cars, buses, trucks, and vans for 97% classification accuracy [79]. Adu-Gyamfi et al. designed a CNN for speed and accuracy, pre-trained on ImageNet [27], and fine-tuned and tested it on the Iowa and Virginia Department of Transportation CCTV data for accuracy of over 89% [80]. Liu et al. trained an ensemble of ResNet models [33] on the MIO-TCD dataset [81] for a classification accuracy of 97.7% [82].

In the state-of-the-art, Ma et al. [83] applied Channel Max Pooling (CMP) to pre-trained CNNs and compared them to non-CMP CNNs on the Cars-196 [84] and Comp-Cars [85], achieving about a 91-95% accuracy on Cars-196 and 97-97% on CompCars - comparable with the existing CNNs. Hedaya et al. applied a super-learner [86] - a data-adaptive ensemble - on the MIO-TCD [81] and BIT-Vehicle datasets [77], with accuracies up to 97.94% and 97.62% respectively [87]. Neupane et al. [88] fine-tuned pre-trained YOLO models, [21], specifically 2 of the YOLOv3 family [60] and YOLOv5 family [89] on a self-proposed dataset of nearly 30,000 images of cars, buses, taxis, bikes, pickup trucks, commercial trucks, and trailers, achieving an accuracy of 95% with the YOLOv5l model. Zhao et al. [90] also apply a YOLO model, by optimizing the YOLOv4 family [61] with an attention mechanism [91] and Feature Pyramid Network (FPN) [92], beating out Faster R-CNN [93] and EfficientDet [94] in several metrics on the BIT-Vehicle [77] and UA-DETRAC [95] datasets. Lastly, [7] achieved 94.8% accuracy with a class minimum of 83.3% for the background null class and less than 3% mislabeling error.

2.2.4 Overall Traffic Object Recognition

The state-of-the-art for multi-class object recognition covers many different technologies, not just computer vision. Ćorović et al. applied YOLOv3 [60] to detect cars, trucks, pedestrians, traffic signs, and traffic lights on the Berkley Deep Drive dataset [96]. Results

were below expectation, with precision and recall scores not reaching above 63% and 55% respectively [97]. Leng et al. applied a Faster-RCNN [98] to U-V disparity maps [99] generated from a stereo vision system. When trained on the KITTI dataset [100], the performance achieved surpasses that of purely using a Faster-RCNN, achieving 70.2% accuracy for detecting cars, 82.1% for pedestrians and 77.9% for cyclists [101]. Li et al. used arrays of ultrasonic sensors along with a conditional likelihood maximization method to detect pedestrians, cyclists and vehicles with class detection rates of 85.7%, 76.7% and 93.1%, respectively, with an overall detection accuracy of 86% [102]. Prakash et al. proposed TransFuser, a novel Multi-Modal Fusion self-attention Transformer integrating image and LiDAR data for an end-to-end autonomous driving model [103]. The transformer can detect other vehicles and traffic lights to navigate, achieving state-of-the-art performance on the CARLA driving simulator [104]. Lin et al. applied a novel CNN technique, Bear-Angle CNN (BA-CNN), to bearing angle images generated from LiDAR point cloud data, to the KITTI dataset, classifying pedestrians, cars and street clutter. Achieved accuracy was competitive with state-of-the-art, achieving 94.7% classification accuracy for pedestrians, 99.3% accuracy for cars, and 94.0% accuracy for street clutter, with an overall accuracy of 96% [105].

2.3 Vision Transformers

Vision transformers [9] are an adaptation of the transformer Natural Language Processing (NLP) deep learning model [10] to image tasks. Transformers quickly become the state-of-the-art [106], surpassing Recurrent Neural Networks (RNN) like Long Short-Term Memory networks with attention, with not just better performance but also using 1/4 of the training time [10]. Transformers rely on attention, similar to mental attention, which allows the model to focus on some parts of the language input. The language input is consumed all at once rather than sequentially. Simultaneous input allows for a more global representation

of the text, thus semantic relations between words, and parallelizability, saving training time compared to the sequential RNNs.

2.3.1 Transformers

The transformer uses an encoder-decoder architecture, as in Figure 2.1. Each transformer layer contains a multi-head self-attention mechanism sub-layer and feed-forward network sub-layer (including the residual normalization layers) for the encoder. The decoder transformer layer includes a second multi-head attention sub-layer, including the output of the encoder. The decoder is not included in ViTs. Vaswani et al. describe their attention mechanism as "mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors" [10]. This query, key and value trio is analogous to an information retrieval system, where the attention unit takes the query and keys to get the values. In this case, the queries (matrix Q) and keys (matrix K) are multiplied with the values (matrix V) to create contextualized embeddings or attention.

2.3.2 Attention

Input embeddings are created by tokenizing the input string X, and for each token i, converting each to a word embedding $x_i \in X$ and a positional encoding; that is to say, there is a vector to represent each word in a given corpus. Q, K and V are formed such that for each word embedding vector x_i, we calculate $q_i = x_i W_Q$, $k_i = x_i W_K$ and $v_i = x_i W_V$, where $q_i \in Q$, $k_i \in K$ and $v_i \in V$. W^Q, W^K, and W^V are weight matrices learned as needed by the transformer [107], of the form $W^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W^K \in \mathbb{R}^{d_{model} \times d_k}$, $W^V \in \mathbb{R}^{d_{model} \times d_v}$ [108]. Attention is thus calculated as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

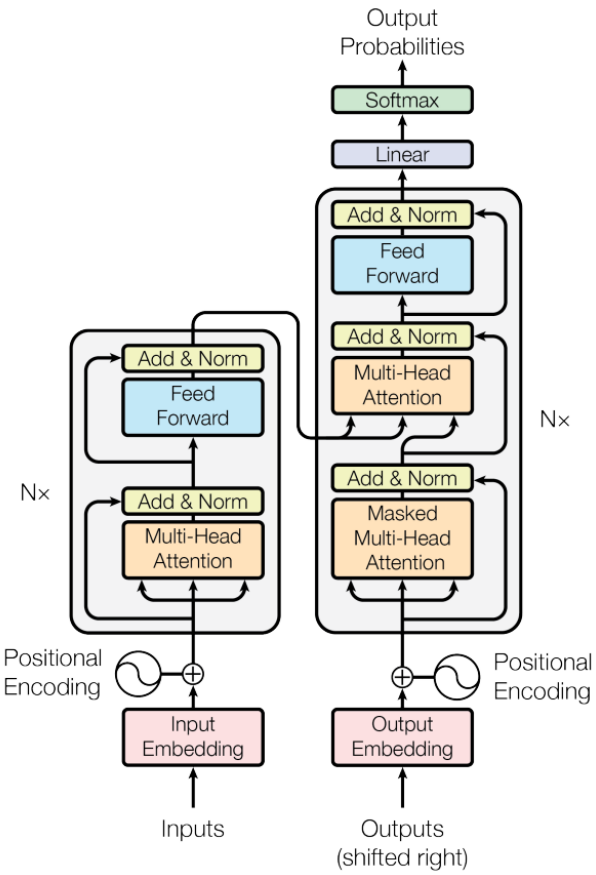

Figure 2.1: *Transformer model architecture. Adapted from* [10, Fig. 1]

This can be seen in Figure 2.2, on the left. Scaling down is done by dividing by $\sqrt{d_k}$, for numerical stability; the dimension of K is $\sqrt{d_k}$. The square root is applied so that the softmax values do not grow too large as the size of K increases [10] (the implementation chooses the value of K). The authors call this form of attention scaled dot-product attention.

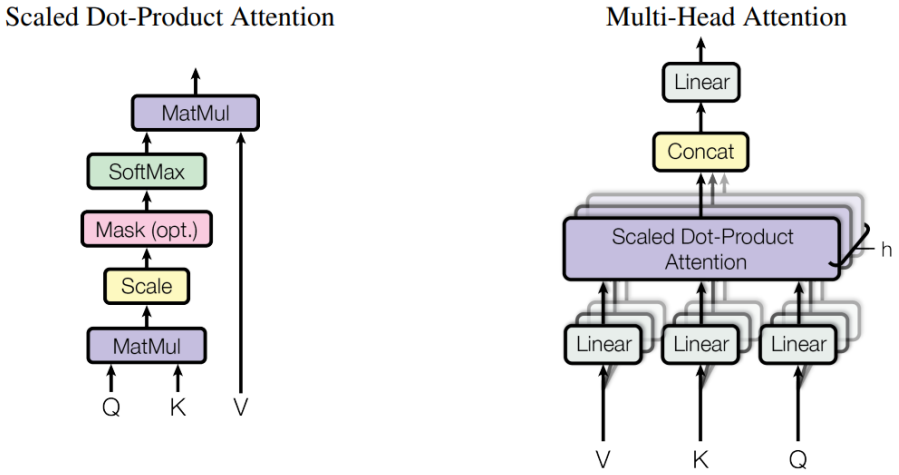

Figure 2.2: *Scaled Dot-Product Attention (left). Multiple attention layers in parallel for Multi-head Attention (right). Adapted from* [10, Fig. 2]

Multi-Head attention applies the same process h times in parallel, as in Figure 2.2 on the right. Each scaled-dot product attention 'head' has its own set of weights and embeddings, allowing each head to capture different information about the data. Each scaled dot-product attention head is concatenated and projected again:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, vW_i^V)$$

Where $W^O \in \mathbb{R}^{hd_v \times d_{model}}$, and h varies by implementation.

2.3.3 Vision Transformer

Motivated by the successes in NLP scaling the size of transformers up, Dosovitskiy et al. experimented with images, seeking to apply transformers to vision as directly as possible [9]. ViTs are structured with only the transformer encoder, connected to an MLP classification head. Each transformer layer is slightly different, with normalization prior to the multi-head self-attention and feed-forward layers. To scale realistically to the size of image data, they reshape each "image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ into a sequence of flattened 2D patches $\mathbf{x} \in \mathbb{R}^{N \times (P^2 \cdot C)}$, where (H, W) is the resolution of the original image, C is the number of channels, (P, P) is the resolution of each image patch, and $N = HW/P^2$ is the resulting number of patches, which also serves as the effective input sequence length for the Transformer". Figure 2.3 shows the image reshaping process.

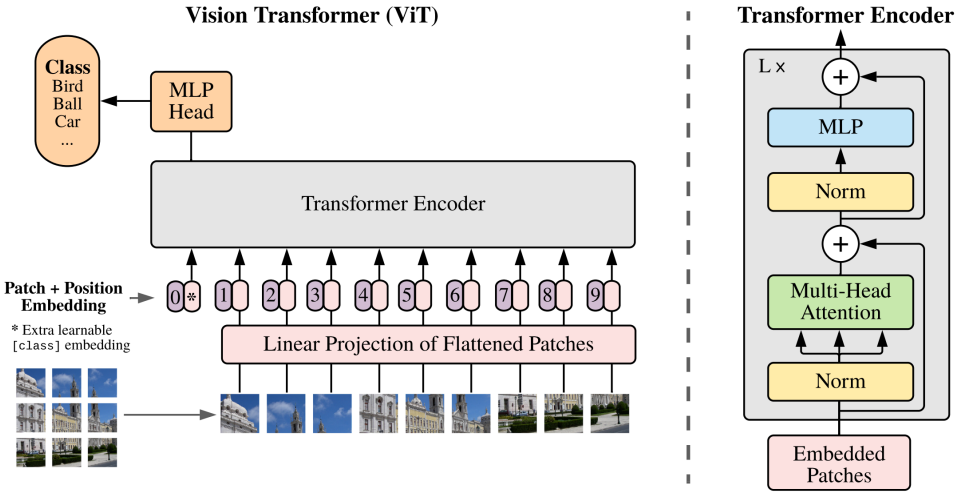

Figure 2.3: *Overview of the Vision Transformer model, illustrating the splitting and linear embedding of images(left). Single transformer layer (right). Adapted from* [9, Fig. 1].

Each patch is coupled with a learnable 1D position embedding for positional awareness. Like BERT, the popular NLP transformer [109], another embedding is added to each sequence of embedded patches to represent the class at the output, leading into the MLP classification head. The patches are flattened and mapped to D dimensions with a trainable linear projection for the patch embeddings, as D is the latent vector used throughout

the transformer. The authors also experimented with Hybrid models formed by CNNs providing the input sequences. Model size variants are based on BERT, with the model sizes ViT-Base, ViT-Large and ViT-Huge as seen in Table 2.1 (although Steiner et al. have since proposed ViT-Tiny and ViT-Small, as well as ResNet hybrid variants [110]). Model variants allow two possible sizes for P, 16 or 32, written as ViT-L/16 or ViT-L/32. This thesis will only look at the ViT-Base and ViT-Large variants.

Model	Layers	Hidden size D	MLP size	Heads	Params
Vit-Base	12	768	3072	12	86M
Vit-Large	24	1024	4096	16	307M
Vit-Huge	32	1280	5120	16	632M

Table 2.1: *Vision Transformer model variant information. Adapted from* [9, Table 1].

Compared with state-of-the-art pre-trained ResNet and EfficientNet models, ViTs pre-trained on the same dataset outperform or match classification accuracy in comparisons while requiring much fewer computational resources, when trained on sufficiently large datasets. ViTs train significantly faster than CNNs: the largest ViT model, ViT-H, took about one-third of the time to pre-train on JFT-300M as their ResNet152 model, and about one-fifth of the time as their EfficientNet-L2 model. When pre-trained on smaller datasets, ViTs are slightly worse or comparable to CNN performance. More can be seen in Table 2 and Figures 2-5 in [9]. ViTs generalize better on these large datasets, on the scale of 14M to 300M images, with datasets such as ImageNet-21k and JFT-300M. When CNNs and ViTs are pre-trained on the same, smaller datasets, ViTs underperform partly due to their lack of image-specific inductive biases. This is because, unlike CNNs and RNNs, ViTs do not have an inherent notion of neighbourhoods of pixels from convolutions or a memory/latent belief held over time. This lack of biases makes ViTs (and transformers in general), much more general NN structures than CNNs or RNNs. While they can quickly compute large amounts of data, they lack specific inductive biases to make sense of data without external help. ViTs require large amounts of augmentation and regularization in addition to the large amounts of data to learn useful biases that are inherent in other

models. These ViTs form very unbiased estimators, so while they do not have many biases to guide their understanding, with enough data they can avoid being held back by biases present in other NN architectures. Thus ViTs can be considered much more general than CNNs and RNNs.

Newer image transformers have been proposed since the original ViT paper in 2020. Data-Efficient image Transformers, or DEiT, introduced by Touvron et al. of Facebook AI and Sorbonne University in December 2020 proposed a transformer-specific teacher-student distillation strategy [111]. They propose DeiT-B based on ViT-B, as well as two smaller models denoted as DeiT-Ti and DeiT-S (for tiny and small, respectively), whose sizes are used for ViT-Ti and ViT-S in the future. Notably, the DeiT transformers learned better from CNN teachers rather than transformer teachers. DeiT-B results with standard distillation are competitive with ViT-B results and when trained along with the proposed distillation strategy, even DeiT-Ti results outperform the much larger ViT-B and ViT-L models, with all results competitive with the state-of-the-art. DeiT-B models when fined-tuned at the larger image size of 384px x 384px outperform ResNet, RegNet [112] and EfficientNet [113] models on ImageNet.

The Shifted Windows (Swin) transformer proposed by Liu et al. of Microsoft Research Asia in March of 2021 is an image transformer for classification, detection and segmentation [114]. They propose sizes Swin-T, Swin-S, Swin-B, and Swin-L based on the above-mentioned ViT and DeiT models. Swins compute self-attention across larger windows of patches, and then self-attention across a set of shifted windows, accounting for the boundaries of the windows in the first self-attention block. There are four stages in each Swin, with the window size growing each stage and at least a pair of self-attention blocks. Swin outperforms ViTs, similarly sized DEiTs, EffNets and RegNets on ImageNet with better throughput than all compared models except DEiTs. Swins achieve state-of-the-art performance on the COCO object detection and segmentation dataset [115] and the ADE20K image dataset [116].

In April of 2021, Caron et al. of Facebook AI, Grenoble Alpes University Inria Research Center and Sorbonne University introduced the DINO method for self-supervision (self-**di**stillation with **no** labels, thus DINO) [117]. Applying DINO to ViTs, they find two properties of self-supervised ViTs not present with supervised ViTs or CNNs; self-supervised ViTs contain features encoding the scene layout and object boundaries, as well as performing well with just a k-NN classifier (without fine-tuning, linear classifier or data augmentation). DINO performs well on several tasks, including image classification and video segmentation, among others. It outperforms state-of-the-art CNN and ViT methods when all are using a linear classifier, achieving 80.1% accuracy on ImageNet.

The last image transformer we will cover, BEiT, was introduced in June of 2021, by Bao et al. of the Harbin Institute of Technology and Microsoft Research [118]. Inspired by BERT [118], BEiT stands for Bidirectional Encoder and uses a masked image modelling task to pre-train. They tokenize an image and it into patches, and then randomly mask some proportion of the image patches. Using a ViT backbone, BEiT attempts to predict the original image tokens based on the encoding of the corrupted image patches. BEiT achieves competitive results for semantic segmentation on ADE20K among ViTs and state-of-the-art image classification results on ImageNet when compared with other ViT-B and ViT-L sized models, with a classification accuracy of 86.3%.

Notably, image transformers are being used in industry, with Generative Pre-trained Transformers (GPTs) [119], possibly iGPT [120], being used by American electric and autonomous vehicle manufacturer Tesla in their AutoPilot program. In June of 2022, Tesla CEO Elon Musk, confirmed they are using multiple GPT models on the social media platform Twitter [121], stating they "are replacing C heuristics for post-processing of the vision NN's 'giant bag of points"'. GPT would be apt for taking vision data and predicting possible paths of traffic objects, given its generative role. CNN architectures also are/were being used at Tesla. On Tesla AI Day in 2021, Andrej Karpathy, a past Director of Artificial Intelligence at Tesla, explained they used RegNet CNNs for object detection

for the architecture's latency-accuracy trade-off. In their "HydraNet" multi-task learning system for tasks downstream of object detection, they use RegNet models for Object Detection, Traffic Lights and Lane Prediction. Some or all of these RegNets may have been replaced with GPT.

Chapter 3

Dataset and Methods

We discuss the dataset used, how it was collected, and its organization. We then discuss the hardware and software used for the testing environment. Then, we discuss the ViT model variants used. We then present the hyperparameters used in experimentation. After that, we discuss the performance metrics we monitored and the results of trained ViT models on the datasets.

3.1 Data Collection and Organization

Dosovitskiy et al. found that ViTs have a much lower pre-training cost than CNNs. However, they may require a much larger amount of data to be competitive with CNNs, at least when pre-training, as they are more prone to overfitting [9]. Thus, the performance on a limited training set compared to CNNs is of interest in both model performance and training time compared to CNNs. In this paper, we will compare model performance but can only present model training time as it is not provided. To facilitate this, we use the RoadLAB [122] image dataset from [7]. The RoadLAB project designed a sensory vehicle system for on-road vehicle experimentation, enabling field study of intelligent ADAS systems. RoadLAB outfitted a vehicle with calibrated stereo cameras on the roof, fine-

grained internal vehicle metric collection, and driver behavioural data collection. The vehicle is shown in Figure 3.1.

The data was collected from sixteen driving sequences over a fixed 28.5-kilometre driving route in London, ON, Canada. In particular, video taken from the stereoscopic cameras during these driving sequences was used to form the RoadLAB image dataset. Video frames were separated and used to make an offline dataset, with which Shirpour [7] developed four independent multi-scale HOG-SVM models to detect traffic objects and create bounding boxes for the image dataset to identify five classes of images: background, pedestrians, signs, traffic lights, and vehicles. A Faster R-CNN model was also used to detect vehicles in parallel with the HOG-SVM. The images were integrated, and redundant detections were eliminated. These detected images were used as the dataset for classification by Shirpour and in this thesis. Due to the nature of front-facing cameras, most images of vehicles were from the front or back, rather than the sides. Traffic lights were also detected from multiple angles.

Figure 3.1: RoadLAB instrumented vehicle (left). Forward stereoscopic vision system (right).

Shirpour trained the ResNet-101 models independently on five classes of traffic objects mentioned, as well as three more specific models to identify three classes of vehicles, five classes of traffic lights, and 20 classes of traffic signs. The five classes of traffic objects include the null background class, vehicles, traffic lights, traffic signs, and pedestrians. The three classes of vehicles include the null background class, cars, buses, and semi-trucks. The five classes of traffic lights are images of the null background class, obscured

traffic lights, and green, red and yellow traffic lights. The traffic sign dataset includes typical Canadian signs, such as arrow signs, construction signs, bicycle lane signs, parking signs, and yield signs, in addition to the null background class. Some examples are shown in Figures 3.2, 3.3, 3.4 and 3.5. The classification results of these four ResNet-101 models will form the comparison for evaluating the ViTs.

Figure 3.2: Examples of some pedestrian images.

3.2 Testing Environment

Classification experimentation and testing were performed upon the general purpose cluster, Cedar, of the Digital Research Alliance of Canada. Cedar is located at Simon Fraser University. Hardware available included six cores of Intel Silver 4216 Cascade Lake @ 2.1GHz CPUs, up to 187GB of memory, and an NVIDIA V100 Volta GPU unit, with

Figure 3.3: Examples from the vehicle class. Shown are two car class examples (left), a background example (top right), and a truck example (bottom right).

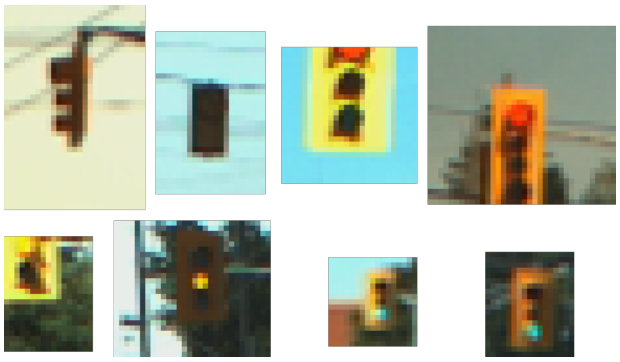

Figure 3.4: Examples from the traffic light class. Shown are two obscured traffic light examples (top left), two red traffic light examples (top right), two yellow traffic light examples (bottom left), and two green light examples (bottom right).

32GB of HBM2 memory, with an SSD allocated at each compute node. More information can be found at [123].

The Python programming language was used for the implementation, and the Vision Transformer was programmed using the Tensorflow machine learning framework [124]. Additional tools included the Project Jupyter notebook format and editor for experimenting and reproducing work and generating Python scripts.

Figure 3.5: Examples from the vehicle class. The signs presented clockwise from the top left are "Yield," "Maximum Speed Limit," "No Entry," traffic arrows, "Pedestrian Crossing," "No trucks," "Not a Through Street", and "Exit Only."

3.3 Model Evaluation

We considered two Python packages for supplying the Vision Transformer models. The official ViT repository by Google does not offer the model in the Tensorflow framework, so two alternatives were assessed. The first is the Github repository in [125], available as the Python package vit-keras, and the popular machine learning repository HuggingFace. The latter was cumbersome to integrate and slower to train, so experimentation proceeded with the first package.

Of the ViT model variants mentioned in [9], vit-keras offers the ViT-Base (ViT-B) and ViT-Large (ViT-L) sizes, with patch sizes 16 and 32. Initial testing showed higher classification accuracies with the ViT-L models while applying the higher resolutions mentioned in [110] (384px x 384px as opposed to the pre-training resolution of 224px x 224px) did not yield significant performance improvements while using around 3x the memory. Large-size ViTs take at least 2x the training time of Base size ViTs, which follows as ViT-L models have 24 transformer layers and 16 self-attention heads per layer as opposed to the 12 layers and 12 self-attention heads of ViT-B models. We can also see that the larger image size took around 2x the training time at a minimum, with smaller patch sizes taking longer than larger patch sizes. Note that for ViTs, smaller patch sizes are computationally more costly, as sequence length is inversely proportional to the square of the patch size. Larger

ViTs are also more accurate than smaller ones given a sufficiently sized dataset. Here, our dataset seems to be large enough for larger-sized ViTs to outperform smaller ones, even without augmentation or regularization. So these performance metrics are expected [9]. A comparison is shown in Table 3.1.

Model Variant	Image Size	Average Validation Accuracy	Average Training Time
ViT-B/16	224px x 224px	92.98%	26.4 mins
	384px x 384px	91.57%	78 mins
ViT-B/32	224px x 224px	91.33%	12.17 mins
	384px x 384px	92.23%	22.14 mins
ViT-L/16	224px x 224px	93.83%	84 mins
	384px x 384px	93.4%	246 mins
ViT-L/32	224px x 224px	93.26%	33.97 mins
	384px x 384px	92.93%	65.33 mins

Table 3.1: *Testing the effects of higher resolution input images on each model variant. Validation accuracy is given for each as an average of three runs.*

Before training, each dataset (overall, signs, traffic lights, and vehicles) was split into a training and test set with a 90%-10% split. During the search, a random 20% of the remaining training data would be used for validation, giving training during search 72% of the dataset and validation 18%. Overall dataset and split sizes are shown in Table 3.2. The same ViT model and hyperparameters were trained for each dataset, resulting in four models. The latter three datasets are somewhat limited in size and present a decent comparison for ViTs and CNNS, given that ViTs depend on large dataset sizes.

Dataset	Total	Training	Validation	Test
Traffic Objects	13,245	9,535	2,383	1,327
Vehicles	3,037	2180	543	304
Traffic Lights	2,636	1,898	472	266
Traffic Signs	2,581	1,859	457	265

Table 3.2: *Number of total dataset images and numbers of images in splits. Training, validation and test sets are random subsets of 72%, 18% and 10% of each dataset.*

3.4 Hyperparameter Tuning

Fine-tuning was assisted by the KerasTuner hyperparameter optimization framework [126], and model and parameter tracking were facilitated by MLflow [127]. KerasTuner was run mainly with the RandomSearch tuner, shown in [128] to be more efficient than grid search. KerasTuner facilitated a hyperparameter sweep over parameters such as model variants, image size, learning rates, optimizers and data augmentation. These were not run all at once. Instead, model variants, optimizers, and data augmentation had separate searches to keep the final search size realistic. Some of the parameters we tuned and their values are presented in Table 3.3. Chosen values are in bold. Note that the model for traffic object classification was an outlier, achieving the best results with a ViT-B/16 and learning rate 3e-05.

Model	Image Size	Optimizers	Learning Rate	Data Augmentation
- ViT-B/16	- **224px x 224px**	- SGD	- 1e-3	- RandAugment
- ViT-B/32	- 384px x 384px	- SGDW	- 1e-4	- mixup [129]
- **ViT-L/16**		- **Adam**	- 3e-5	- CutMix [130]
- ViT-L/32		- AdamW	- **1e-5**	- **Manually**
		- AdaMax	- 3e-6	
			- 1e-6	

Table 3.3: *Some of the hyperparameter search space.*

Among optimizers, we began with the commonly used Stochastic Gradient Descent (SGDW) [131] and Adam optimizers [132]. SGD calculates the gradient of the loss function for each training sample and multiplies it by a given learning rate to determine the amount to adjust parameters in a machine learning model. This adjustment allows the network to learn parameter weights and perform predictions. The formula for SGD is generally given as $w := w - \eta \nabla Q_i(w)$, where w is a weight in a model, η is the learning rate, and Q_i is the value of the loss function for any i-th training example. Adam is a variant that adaptively estimates first and second-order moments. It continuously updates the gradient's averages and the gradient's squares and uses these to calculate how much to change a weight. Adam

also reduces the impact of these averages with each update via a shrinking hyperparameter. Adamax is another variant suggested in [132] to use the infinity norm L^P, a norm inversely proportional to a value approaching infinity. It is sometimes superior to Adam for models with embeddings. SGDW and AdamW are modifications by Loschilov and Hutter in [133] to allow L_2 weight decay to regularize variables with larger gradients better, and proposes this leads to better generalization error.

Learning rate refers to the amount that the gradient of a loss function can affect a parameter's weight. It is typically in the range (0,1] and affects the speed and possibility of convergence of an optimization function. In practical terms, a high learning rate allows faster training and an increase in accuracy but can cause the function to move past minimum values for the function by moving too quickly. Conversely, a low learning rate slows the down training but can allow the optimization function to move into minimal values and converge to an optimal accuracy. Thus it is common practice to start with a larger learning rate and lower it over time, such as multiplying by 0.1, to allow the function to learn quickly and adapt to the gradient and converge. Some optimizers like Adam handle this automatically.

As shown in 3.1, results were impressive even without augmentation. This is interesting as the nature of ViTs should lead them to be worse than CNNs given the same amount of data, especially without significant augmentation or regularization. This shows that during pre-training, a significant understanding of images and the ViT's own inductive biases were formed, leading to decent fine-tuning results out-of-the-box. Image augmentation was tested manually, as well as with some modern techniques. RandAugment, is an automatic augmentation policy which randomly applies augmentations based on two parameters, N and M, controlling the number and intensity of augmentation procedures to use, respectively [134]. The mixup policy combines data features and labels so that the network does not get too overconfident about the relationship between features and labels using a parameter λ that controls the degree of interpolation. Here, $\lambda \sim \text{Beta}(\alpha, \alpha), \text{for } \alpha \in (0, \infty)$,

such that $\hat{x} = \lambda x_i + (1 - \lambda)x_j$ and $\hat{y} = \lambda y_i + (1 - \lambda)y_j$, for a given training example. As α approaches 0, the images become fully converged. CutMix is similar to mixup but replaces mixing features with cutting regions of images and replacing them with a region of another image. The value α for CutMix - based on the same notion as mixup - is typically only set to 1 [130]. Parameters for RandAugment and the values examined are shown in Table 3.4.

RandAugment		mixup	CutMix
N	M	α	α
- 0	- 0	- 0.0001	- 1
- 2	- 2	- 0.1	
- 5	- 7	- 0.2	
- 10	- 5	- 0.5	
	- 15	- 0.8	
	- 30	- 1.0	

Table 3.4: *Parameters for RandAugment, mixup and CutMix.*

The manual augmentation parameter search space is presented in Table 3.5. Each value v given for shear, zoom, or translation represents a range $(-v, v)$ randomly sampled to pick a value to shift in terms of rotational degrees, the fraction of image size, or the fraction of image width and height, respectively. Brightness values represent a range from [lower, upper] to pick a brightness value to offset image brightness. Some parameters, such as brightness, zoom and flip were tested in isolation prior to performing broader hyperparameter searches. The motivation was to narrow the search space to more feasible sizes given limited time and compute resources. Augmentations were chosen based on real-world intuition. Slight shearing and zooming based on the angle of cameras seem plausible, as were translations to simulate signs or vehicles not fully captured. Horizontal flips were considered as vehicles were typically symmetrical, however, vertical flips were not as none of the including traffic objects are likely to be seen vertically flipped. Brightness range can be seen as similar to real-world light conditions; random contrast shifting is also applicable but was not thoroughly tested. The augmentation was ultimately chosen from this search space and is marked in bold. Some additional tests for brightness and colour channel

shifting are presented in 3.6. Shown in 3.7, the inclusion of shearing showed inconsistent effects on performance and did not improve overall validation accuracy, so it was omitted.

Augmentation	Value
Shear	20, 45, 60, 75, 90, 120
Zoom	0.0, **0.1**, 0.25, 0.5, 0.75, 0.9
Translation (x)	**0.1**, 0.25, 0.5, 0.75, 0.9
Translation (y)	**0.1**, 0.25, 0.5, 0.75, 0.9
Horizontal Flip	False, **True**
Brightness Range (lower)	0.0, 0.1, 0.25, 0.5, 0.75, 0.9
Brightness Range (upper)	0.0, 0.1, 0.25, 0.5, 0.75, 0.9
Channel Shift	0.0, 0.1, 0.25, 0.5, 0.75, 0.9

Table 3.5: *Search space of the manual data augmentation parameters.*

Validation Accuracy	Brightness Range (lower)	Brightness Range (upper)	Channel Shift
79.1%	0.25	0.75	0.0
77.4%	0.25	0.25	0.75
77.2%	0.5	0.25	.75
77.0%	0.9	0.75	0.5

Table 3.6: *Best four results for search over brightness ranges and channel shift ranges.*

Validation Accuracy	Shear Setting
93.7%	60
93.4%	120
92.4%	90
90.9%	75

Table 3.7: *Examples of minimal effect of shear settings on performance*

To perform a more analogous comparison, we searched over some of the same parameters as our baseline CNN method in [7]. Manual augmentation settings used for the CNNs are shown in 3.8. Their method used cross-validation and explored various ranges for the tuning parameters, then retrained on the complete training set after choosing parameters. Additional details were not given. It should be noted that cross-validation was not used for this thesis. Otherwise, we attempted to make training as similar as possible, with both studies using the same datasets for training and testing.

Method	Description	Range
Translate	Each image is translated in the horizontal and vertical direction by a distance, in pixels	(-10, 10)
Rotate	Each image is rotated by an amount, in degrees	(-15, 15)
Scale	Each image is scaled in the horizontal and vertical direction by a factor	(0.5, 1.5)
Shear	Each image is sheared along the horizontal or vertical axis by a factor	(-30, 30)

Table 3.8: *Data augmentation parameters and descriptions used in [7]. Scale is analogous to Zooming in the horizontal and vertical directions independently.*

During each trial for the model search facilitated by KerasTuner RandomSearch, training was performed over ten epochs, with a reduced learning rate policy with a patience setting of 2 epochs and an early stopping policy with a patience setting of 4 epochs. Both policies evaluated validation accuracy with a delta of 1e-4, applying the policy when validation accuracy had not increased by the delta in the number of specified patience epochs. Model search reported training loss and accuracy and validation loss and accuracy, with validation accuracy being the guiding metric. For manual augmentation, we used a batch size of 16 for ViT-L model variants and a batch size of 32 for ViT-B model variants. For ViT-B/16 models, training time per epoch was typically 150-160 seconds, or 250-270 milliseconds per step, while for ViT-B/32 models, training time per epoch was typically 60-90 seconds, or 100-110 ms/step. For ViT-L/16 models, training time per epoch was typically 450-470 seconds or 770-790 ms/step, while for ViT-L/32 variants, training time per epoch was around 150-170 seconds or 250-270 ms/step. Models usually used all ten epochs during the search. This is summarized in Table 3.9.

For RandAugment, we used a batch size of 16 for both model variants. Training time per epoch was around 200 seconds per epoch or 650-675 ms/step for ViT-B/16 models and around 100-110 seconds per epoch or 330-350 ms/step for ViT-B/32 models. For ViT-L/16 models, training time per epoch was typically 450-470 seconds or 770-790 ms/step, while for ViT-L/32 variants, training time per epoch was around 150-160 seconds or 250-

270 ms/step. It seems as though, as the model size increases, the disparities between augmentation schemes minimize. RandAugment-trained models usually converged in 6-8 epochs before early stopping. This is summarized in Table 3.10.

Model Variant	Batch Size	s/Epoch	ms/step
ViT-B/16	32	150-160	250-270
ViT-B/32	32	60-90	100-110
ViT-L/16	16	450-470	770-790
ViT-L/32	16	150-170	250-270

Table 3.9: *Batch size and approximate time taken for training models with manual augmentation during a search.*

Model Variant	Batch Size	s/Epoch	ms/step
ViT-B/16	16	200	650-675
ViT-B/32	16	100-110	330-350
ViT-L/16	16	450-470	770-790
ViT-L/32	16	150-160	250-270

Table 3.10: *Batch size and approximate time taken for training models with random augmentation during a search. Tune given in seconds per epoch and milliseconds per step.*

Recall that smaller patch sizes are more computationally costly than larger patch sizes.

Chapter 4

Experimental Results

4.1 Performance Evaluation

At test time, models were automatically selected using the best results of the search stage with manual augmentations, as detailed above. The model variant selected was Vit-L/16 with the Adam optimizer and a learning rate of 1e-5 for each model except the traffic object classification model, for which the model variant ViT-B/16 was used with a slightly larger learning rate of 3e-5. The training parameters were adjusted to have a batch size of 32 and 50 training epochs, except for the traffic object classification model, for which we allocated 100 training epochs to allow its larger dataset size to converge. We set the reduced learning rate policy with a patience setting of 5 epochs and the early stopping policy with a patience setting of 10. After training was complete, we evaluated the model on the test set, and now we report performance.

We used four criteria to assess the value of the ViTs' performance. These are test accuracy, precision, recall, and F1 score. They are defined as:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \tag{4.1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.3}$$

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

where TP is the number of True Positive predictions, and FN is the number of False Positive predictions.

4.2 Results

The final training run for the traffic objects model ran for 34 epochs, converging at a validation accuracy of 94.16%, where each epoch took about 150-160 seconds per epoch or 255-270 ms/step. The vehicle classification model ran for 14 epochs, converging at a validation accuracy of 96.13%, and each epoch took about 30-35 seconds, or 440-460 seconds/step. The traffic light model ran for 15 epochs, converging at a validation accuracy of 98.52%, and each epoch took about 85-90 seconds or 785-800 ms/step. Lastly, the traffic sign model ran for 36 epochs, converging at a validation accuracy of 96.50%, with each epoch taking about 90-95 seconds, or 785-795 ms/step. This is shown summarized in Table 4.1. The time taken for the traffic objects model seems much larger, though this may be because the dataset is much larger than the others.

Dataset	Validation Accuracy	Epochs Taken	s/Epoch	ms/step
Traffic Objects	94.16%	34/100	150-160	255-270
Vehicles	96.13%	14/50	30-35	440-460
Traffic Lights	98.52%	15/50	80-95	785-800
Traffic Signs	96.50%	36/50	90-95	785-795

Table 4.1: *Approximate time to train the final model for each dataset. Time is measured in seconds per epoch and milliseconds per step. The batch size was 32, and each training step took 1 second.*

After training and testing the models for general traffic objects, vehicles, traffic lights

and traffic signs, we compare the performance metrics to those of the ResNet-101 CNN models used in [7] in Table 4.2. Additionally, we compute confusion matrices for each model to show the accuracy per class, shown in Figures 4.1, 4.2, 4.3, and 4.4.

Dataset	CNN Accuracy	ViT Accuracy	Precision	Recall	F1
Traffic Objects	94.1%	97.7%	97.7%	97.9%	97.8%
Vehicles	94.8%	96.7%	96.8%	96.6%	96.7%
Traffic Lights	96.2%	99.2%	99.3%	99.4%	99.4%
Traffic Signs	96.1%	97.2%	97.6%	96.9%	97.2%

Table 4.2: *Performance metrics comparison of ViT models and Res-Net101 from [7]. Precision, recall and F1 metrics are from the ViT models.*

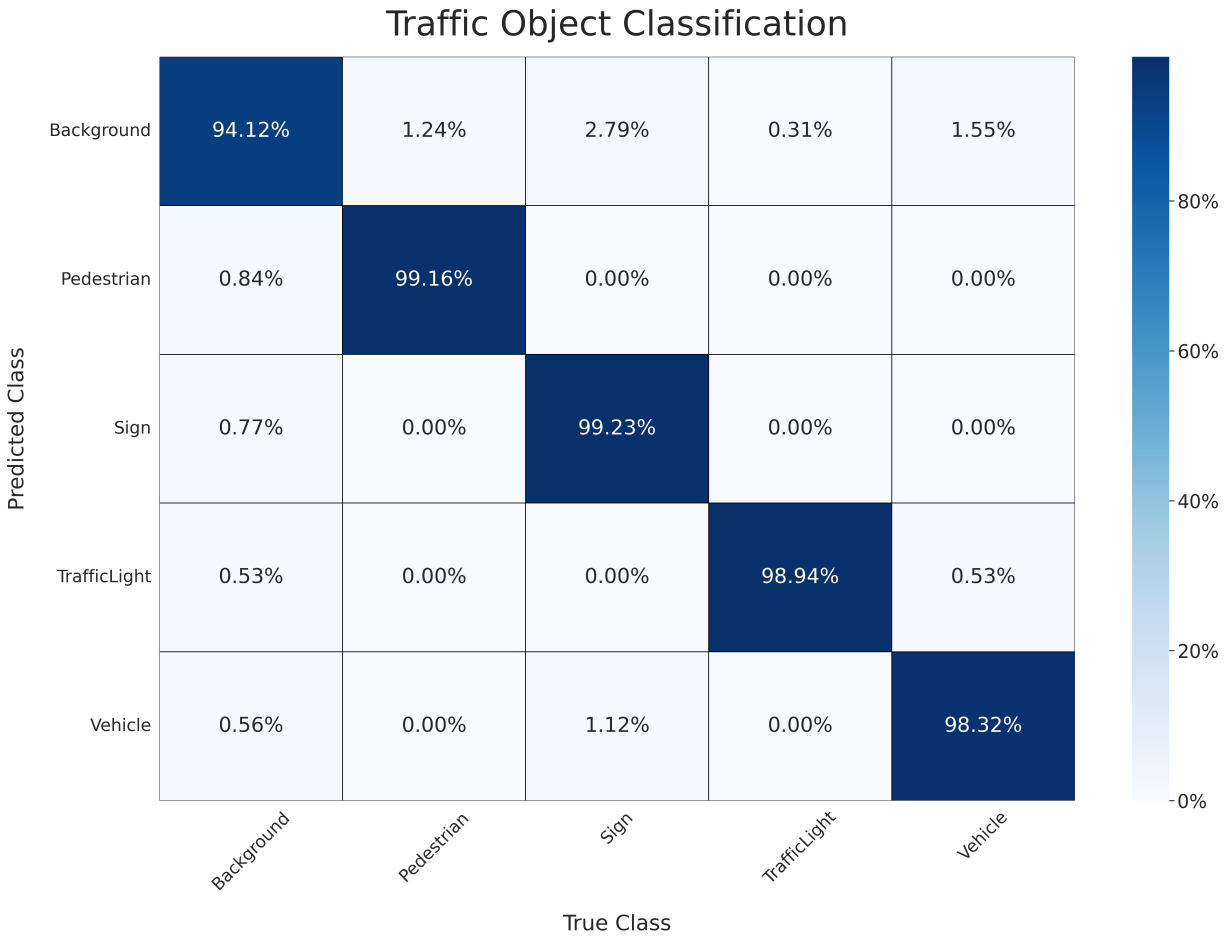

Figure 4.1: *Confusion matrix for traffic object classification results.*

The performance metrics show promising results, with the accuracy increasing at least 1.1% up to 3% from the Res-Net-101 models. The most significant improvement was for

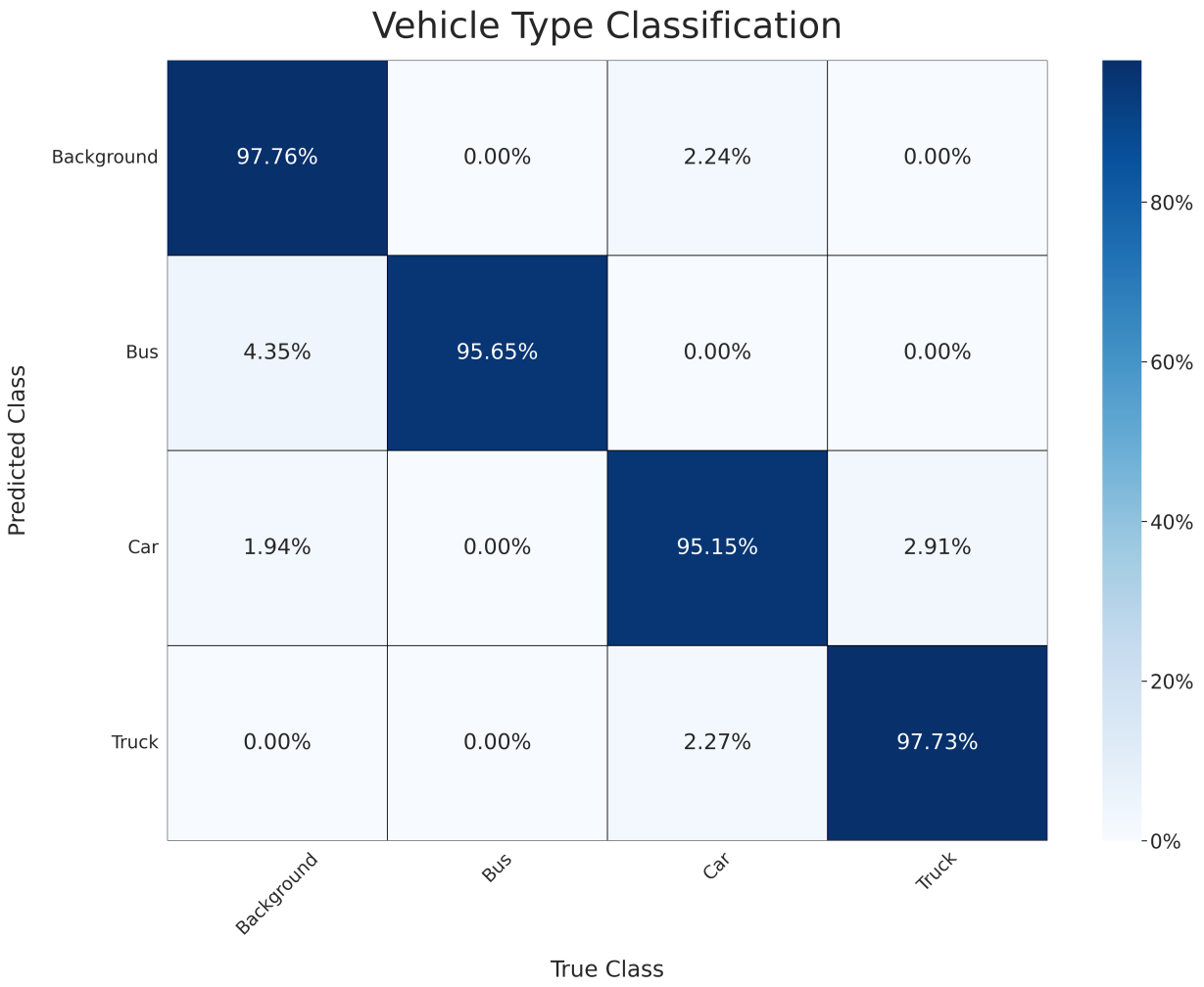

Figure 4.2: *Confusion matrix for vehicle classification results.*

the traffic light dataset. The results are otherwise impressive, as the accuracy improves for every class, with traffic light classification accuracy over 99%. Precision, recall and F1 results are also promising, showing little deviance from the classification accuracy and high relevance of results.

Traffic object classification results are presented in Figure 4.1 show an accuracy of 97.9%, an increase of 3.8%, with the pedestrian and sign classes achieving the highest accuracies, at 99.16% and 99.23% each. Comparing traffic object classification shows an increase in accuracy for every class compared to [7, Fig. 3.10]. The most significant increase was for the vehicle class, at 11.82%.

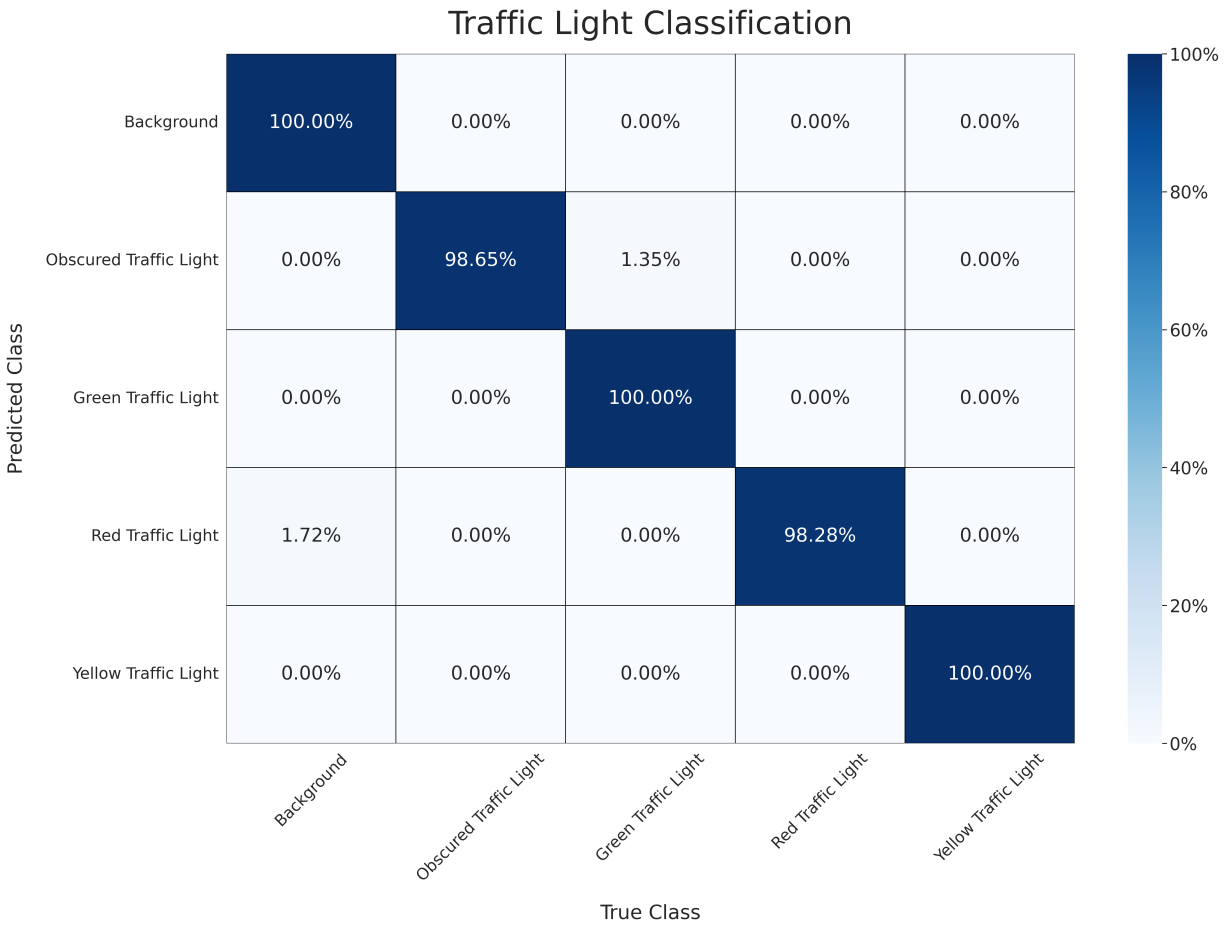

Figure 4.3: *Confusion matrix for traffic light classification results.*

Vehicle classification results are presented in Figure 4.2 show an accuracy of 96.7%, an increase of 1.9%, showing mixed results, as the bus and truck classes did not improve, dropping 0.75% and 1.07% respectively. However, every other per class accuracy improved, the most for the null background class, going from 87.3% to 97.76% [7, Fig. 3.13], the most significant increase for the class.

Traffic light classification results are presented in Figure 4.3 show an accuracy of 99.2%, up 3%, the most significant improvement in the comparison. Notably, there were three perfect accuracy scores for the background, green traffic light and yellow traffic light classes, with the lowest accuracy still being an impressive 98.28%. The greatest increase was for the green traffic light class, up 10% from the ResNet-101 model [7, Fig. 3.12], and notably,

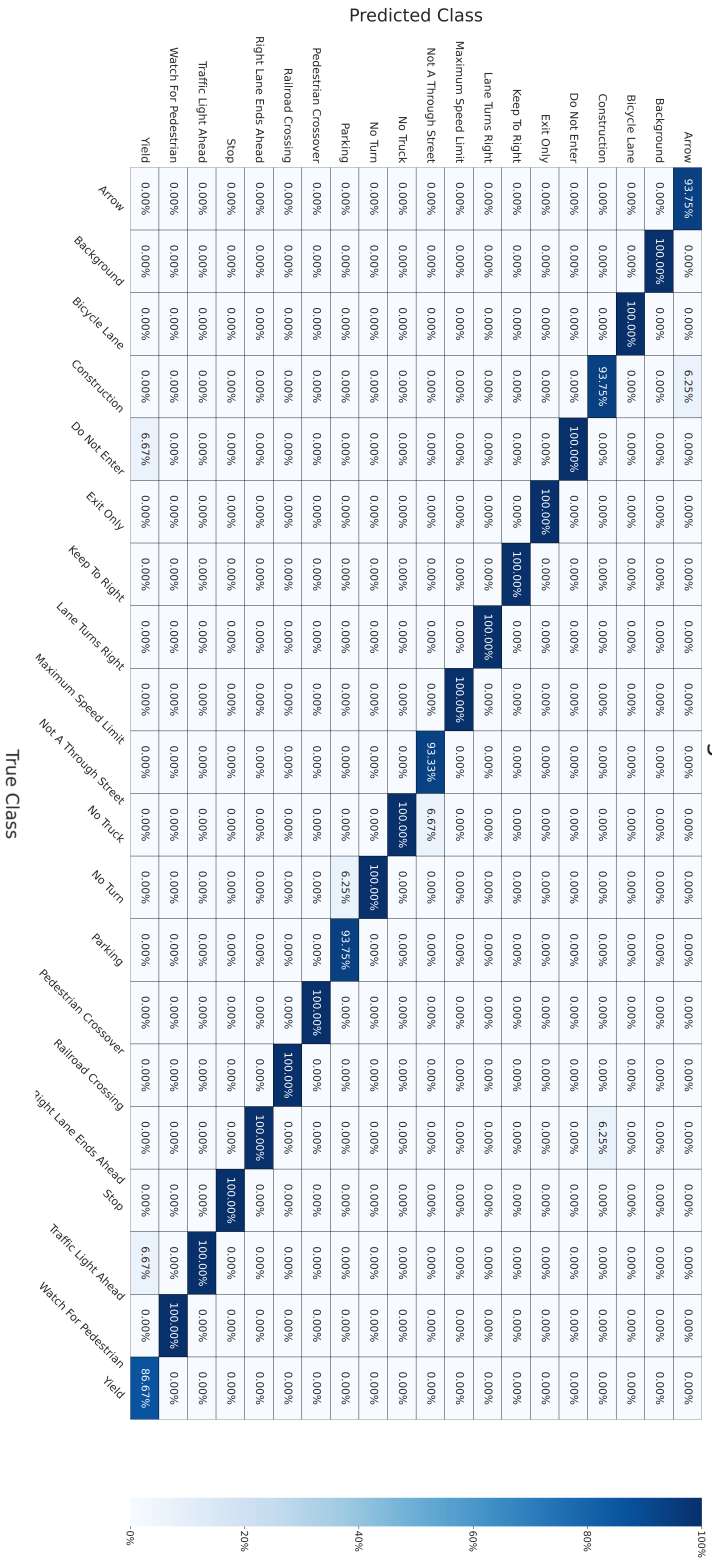

Figure 4.4: *Confusion matrix for traffic sign classification results.*

the obscured traffic light, and red traffic light classes performed worse, down from 98.8% and 99.2% respectively, marking a decrease of almost 1% for the latter.

As shown in Figure 4.4, the traffic sign classification accuracy showed a modest improvement, with accuracy increasing 1.1% from 96.1% to 97.1%. For this class, however, 15 of the 20 classes achieved perfect accuracy, up from 11 for the Res-Net101 model [7, Fig. 3.11]. The remaining five, arrow signs, construction signs, no truck signs, parking signs, and yield signs, showed mixed results, with only the arrow class increasing by 0.05%. The minimum accuracy, however, did increase, going from 84.4% for the "No Turn" sign class, to 86.67% for the "Yield" class, an improvement of 2.27%.

4.3 Discussion

We see that the ViT improves upon the CNN results in nearly every situation for accuracy. As mentioned prior, the lack of inductive biases for transformers means they need much more data as well as augmentation and regularization than CNN models. Transformers however seem to not saturate with data as soon as CNNs do, so these results may improve in the future with more training data or more advanced augmentation. Larger ViT models generalize better than smaller models but require more data to realize their potential. This held in our situation, as well as smaller patch sizes being more computationally expensive, with 16 x 16 patches being at least 2x slower to train than 32 x 32 patch sizes. One caveat is inference speed, which is where the ResNet-101 model significantly outperforms ViTs, as the inference throughput (images/s) on ImageNet is significantly faster among ResNet models. On an NVIDIA V100 Volta GPU with 16GB of memory, ResNet-101 model is almost 9x faster than a ViT-B/16 model and 27x faster than a ViT-L/16 model [111, Table 5]. The DEiT model in the mentioned paper could also be the solution, as all compared DEiT models have better performance than the ViT models, and all have much higher throughput. The DeiT-Ti notably is almost 3x faster than the ResNet-101 model, and the

DeiT-S model is slightly faster but with better results on ImageNet.

In this chapter, we presented the final model variants, optimizers and learning rates used for the ViT models. We trained them one final time and training times, and report the test metrics. When comparing to the models [7], we can see a net increase in performance accuracy, improving by at least 1% per dataset. There were, however, still some mixed results. Among the vehicle classification dataset, the bus class classification accuracy decreased slightly, and five classes decreased in accuracy for the traffic sign dataset. However, this is a promising increase in classification accuracy, given the somewhat limited dataset sizes and the ViT models' need for large datasets.

Chapter 5

Conclusion

In this thesis, we have explored the promising Vision Transformer architecture and assessed performance in vision-related ADAS scenarios. we have examined four ViT models on the problems of general traffic object classification, vehicle classification, traffic light classification and traffic sign classification. We discussed related work, including general image classification, ADAS-specific classification, and Vision Transformers. We fine-tuned four variants of Google's ViT model to the pre-detected image datasets provided by Shirpour [7] and RoadLAB [122]. We experimented with ViTs, explored different hyperparameter options and evaluated the performance of ViTs on these datasets, and found impressive results, improving from 1% to 3% per dataset. We also demonstrate results when approaching the model size variants ViT-B\16, ViT-B\32, ViT-L\16, and ViT-L\32 as well as results when fine-tuning at higher resolutions.

The first objective of this study is to determine the utility of ViTs in ADAS applications, especially compared to existing CNN methods, such as the prior method used by RoadLAB in [7]. The prior CNN method worked well to classify traffic objects, so we have met our objective and can see that the use of ViTs is viable for further ADAS research. We presented an assessment showing competitive ViT performance, achieving a classification accuracy of 94.1% when classifying vehicles, traffic lights, traffic signs and pedestrians on

a dataset, and testing accuracies of 94.8%, 96.2%, and 96.1% on vehicles, traffic light and traffic signs datasets. While working on the same datasets, the ViT outperformed prior CNN methods in [7]. The second objective of this study

5.1 Future Work

As stated earlier, ViTs are significantly more robust to image permutations than CNNs. This robustness was not assessed and stands as potential future work in preparing a test dataset of corrupted images, perhaps in different driving weather conditions, or identifying a set of synthetic image augmentations to serve as a simulation for corrupted images. After choosing a method of image permutation/corruption, the robustness of CNNs and image transformers could be compared by augmenting data at test time. Testing the performance of these models on the augmented dataset could serve as a comparison for assessing robustness. Other work would include testing the ViT architecture's performance on different dataset sizes and assessing the impact of dataset size on ViT classification accuracy. We are also interested in applying ViT to other driving datasets and examining the performance of driving conditions in different parts of the world, perhaps on traffic signs in other countries. Lastly, several different transformer models for image classification have been released since Google's ViT [135]. Comparing the state-of-the-art image classification transformers could be helpful in applications in ADAS in terms of efficiency or performance. Performance or efficiency could be compared by training on a selection of driving datasets and comparing performance metrics, as well as inference speed, and the size of models relative to their performance.

Bibliography

[1] U. Z. Abdul Hamid, F. R. Ahmad Zakuan, K. Zulkepli, M. Z. Azmi, H. Zamzuri, M. A. Abdul Rahman, and M. Zakaria, "Autonomous emergency braking system with potential field risk assessment for frontal collision mitigation," in *2017 IEEE Conference on Systems, Process and Control (ICSPC)*, pp. 71–76, 12 2017.

[2] S. D. of Health, "Global status report on road safety 2018." `https://www.who.int/publications/i/item/9789241565684`, Jun. 2018. Accessed: 2022-08-20.

[3] K. A. Brookhuis, D. de Waard, and W. H. Janssen, "Behavioural impacts of advanced driver assistance systems – an overview," *European Journal of Transport and Infrastructure Research*, vol. 1, Jun. 2001.

[4] M. Capallera, Q. Meteier, E. de Salis, L. Angelini, S. Carrino, O. A. Khaled, and E. Mugellini, "Owner manuals review and taxonomy of adas limitations in partially automated vehicles," in *Proceedings of the 11th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, AutomotiveUI '19, (New York, NY, USA), p. 156–164, Association for Computing Machinery, 2019.

[5] A. Moujahid, M. ElAraki Tantaoui, M. D. Hina, A. Soukane, A. Ortalda, A. ElKhadimi, and A. Ramdane-Cherif, "Machine learning techniques in adas: A review," in *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, pp. 235–242, 2018.

[6] H. J. Vishnukumar, B. Butting, C. Müller, and E. Sax, "Machine learning and deep neural network — artificial intelligence core for lab and real-world test and validation for adas and autonomous vehicles: Ai for efficient and quality test and validation," in *2017 Intelligent Systems Conference (IntelliSys)*, pp. 714–721, 2017.

[7] M. Shirpour, N. Khairdoost, M. Bauer, and S. Beauchemin, "Traffic object detection and recognition based on the attentional visual field of drivers," *IEEE Transactions on Intelligent Vehicles*, pp. 1–1, 2021.

[8] D. Babić, D. Babić, M. Fiolić, and Z. Šarić, "Analysis of market-ready traffic sign recognition systems in cars: A test field study," *Energies*, vol. 14, no. 12, 2021.

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *CoRR*, vol. abs/2010.11929, 2020.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[11] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, and D. Tao, "A survey on vision transformer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2022.

[12] M. Naseer, K. Ranasinghe, S. H. Khan, M. Hayat, F. S. Khan, and M. Yang, "Intriguing properties of vision transformers," *CoRR*, vol. abs/2105.10497, 2021.

[13] A. Shaout, D. Colella, and S. Awad, "Advanced driver assistance systems - past, present and future," in *2011 Seventh International Computer Engineering Conference (ICENCO'2011)*, pp. 72–82, 2011.

[14] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773–1795, 2013.

[15] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Gläser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1341–1360, 2021.

[16] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, pp. 137–154, May 2004.

[17] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, 2005.

[18] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[19] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp. 237–242, 2020.

[20] W. Zhiqiang and L. Jun, "A review of object detection based on convolutional neural network," in *2017 36th Chinese Control Conference (CCC)*, pp. 11104–11109, 2017.

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[23] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, pp. 381–386, 2020.

[24] P. N. Druzhkov and V. D. Kustikova, "A survey of deep learning methods and software tools for image classification and object detection," *Pattern Recognition and Image Analysis*, vol. 26, pp. 9–15, Jan 2016.

[25] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[26] H. Fujiyoshi, T. Hirakawa, and T. Yamashita, "Deep learning-based image recognition for autonomous driving," *IATSS Research*, vol. 43, no. 4, pp. 244–252, 2019.

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, pp. 211–252, Dec 2015.

[28] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang, "Large-scale image classification: fast feature extraction and svm training," in *CVPR 2011*, pp. 1689–1696, IEEE, 2011.

[29] J. Sánchez and F. Perronnin, "High-dimensional signature compression for large-scale image classification," in *CVPR 2011*, pp. 1665–1672, IEEE, 2011.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Sys-*

tems (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[31] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM Comput. Surv.*, dec 2021. Just Accepted.

[32] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, 09 2014.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[34] M. B. Jensen, M. P. Philipsen, A. Møgelmose, T. B. Moeslund, and M. M. Trivedi, "Vision for looking at traffic lights: Issues, survey, and perspectives," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 1800–1815, 2016.

[35] F. Lindner, U. Kressel, and S. Kaelberer, "Robust recognition of traffic signals," in *IEEE Intelligent Vehicles Symposium, 2004*, pp. 49–53, 2004.

[36] U. Franke, D. Pfeiffer, C. Rabe, C. Knoeppel, M. Enzweiler, F. Stein, and R. G. Herrtwich, "Making bertha see," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, June 2013.

[37] Y. Shen, U. Ozguner, K. Redmill, and J. Liu, "A robust video based traffic light detection algorithm for intelligent vehicles," in *2009 IEEE Intelligent Vehicles Symposium*, pp. 521–526, 2009.

[38] C. Wang, T. Jin, M. Yang, and B. Wang, "Robust and real-time traffic lights recognition in complex urban environments," *International Journal of Computational Intelligence Systems*, vol. 4, no. 6, pp. 1383–1390, 2011.

[39] Z. Shi, Z. Zou, and C. Zhang, "Real-time traffic light detection with adaptive background suppression filter," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 3, pp. 690–700, 2016.

[40] R. Qian, B. Zhang, Y. Yue, Z. Wang, and F. Coenen, "Robust chinese traffic sign detection and recognition with deep convolutional neural network," in *2015 11th International Conference on Natural Computation (ICNC)*, pp. 791–796, IEEE, 2015.

[41] K. Behrendt, L. Novak, and R. Botros, "A deep learning approach to traffic lights: Detection, tracking, and classification," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1370–1377, 2017.

[42] M. B. Jensen, K. Nasrollahi, and T. B. Moeslund, "Evaluating state-of-the-art object detector on challenging traffic light data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[43] X. Liu and W. Q. Yan, "Traffic-light sign recognition using capsule network," *Multimedia Tools and Applications*, vol. 80, pp. 15161–15171, Apr 2021.

[44] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *Advances in neural information processing systems*, vol. 30, 2017.

[45] J. Miura, T. Kanda, and Y. Shirai, "An active vision system for real-time traffic sign recognition," in *ITSC2000. 2000 IEEE Intelligent Transportation Systems. Proceedings (Cat. No.00TH8493)*, pp. 52–57, 2000.

[46] S. Lafuente-Arroyo, P. Gil-Jimenez, R. Maldonado-Bascon, F. Lopez-Ferreras, and S. Maldonado-Bascon, "Traffic sign shape classification evaluation i: Svm using distance to borders," in *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, pp. 557–562, 2005.

[47] S. Maldonado Bascón, J. Acevedo Rodríguez, S. Lafuente Arroyo, A. Fernndez Caballero, and F. López-Ferreras, "An optimization on pictogram identification for the road-sign recognition task using svms," *Computer Vision and Image Understanding*, vol. 114, no. 3, pp. 373–383, 2010.

[48] F. Zaklouta and B. Stanciulescu, "Real-time traffic sign recognition in three stages," *Robotics and Autonomous Systems*, vol. 62, no. 1, pp. 16–24, 2014. New Boundaries of Robotics.

[49] M. Mathias, R. Timofte, R. Benenson, and L. Van Gool, "Traffic sign recognition — how far are we from the solution?," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2013.

[50] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, vol. 32, pp. 323–332, 2012. Selected Papers from IJCNN 2011.

[51] R. Timofte and L. Van Gool, "Sparse representation based projections," in *Proceedings of the 22nd British machine vision conference-BMVC 2011*, pp. 61–61, BMVA Press, 2011.

[52] A. de la Escalera, L. Moreno, M. Salichs, and J. Armingol, "Road traffic sign detection and classification," *IEEE Transactions on Industrial Electronics*, vol. 44, no. 6, pp. 848–859, 1997.

[53] C. Fang, C. Fuh, P. Yen, S. Cherng, and S. Chen, "An automatic road sign recognition system based on a computational model of human recognition processing," *Computer Vision and Image Understanding*, vol. 96, no. 2, pp. 237–268, 2004. Special Issue on Event Detection in Video.

[54] A. Broggi, P. Cerri, P. Medici, P. P. Porta, and G. Ghisio, "Real time road signs recognition," in *2007 IEEE Intelligent Vehicles Symposium*, pp. 981–986, 2007.

[55] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale convolutional networks," in *The 2011 International Joint Conference on Neural Networks*, pp. 2809–2813, 2011.

[56] J. Jin, K. Fu, and C. Zhang, "Traffic sign recognition with hinge loss trained convolutional neural networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 1991–2000, 2014.

[57] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, and S. Yan, "Perceptual generative adversarial networks for small object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[58] Y. Zhu, C. Zhang, D. Zhou, X. Wang, X. Bai, and W. Liu, "Traffic sign detection and recognition using fully convolutional network guided proposals," *Neurocomputing*, vol. 214, pp. 758–766, 2016.

[59] C. Dewi, R.-C. Chen, Y.-T. Liu, X. Jiang, and K. D. Hartomo, "Yolo v4 for advanced traffic sign recognition with synthetic training data generated by various gan," *IEEE Access*, vol. 9, pp. 97228–97242, 2021.

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

[61] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[62] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2794–2802, 2017.

[63] J. Zhang, W. Wang, C. Lu, J. Wang, and A. K. Sangaiah, "Lightweight deep network for traffic sign classification," *Annals of Telecommunications*, vol. 75, no. 7, pp. 369–379, 2020.

[64] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection using optical sensors: a review," in *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No.04TH8749)*, pp. 585–590, 2004.

[65] T. Kalinke, C. Tzomakas, and W. von Seelen, "A texture-based object detection and an adaptive model-based classification," in *Procs. IEEE Intelligent Vehicles Symposium*, vol. 98, pp. 341–346, Citeseer, 1998.

[66] U. Handmann, T. Kalinke, C. Tzomakas, M. Werner, and W. Seelen, "An image processing system for driver assistance," *Image and Vision Computing*, vol. 18, no. 5, pp. 367–376, 2000.

[67] K. Yousaf, A. Iftikhar, and A. Javed, "Comparative analysis of automatic vehicle classification techniques: a survey," *International Journal of Image, Graphics and Signal Processing*, vol. 4, no. 9, p. 52, 2012.

[68] S. Gupte, O. Masoud, and P. Papanikolopoulos, "Vision-based vehicle classification," in *ITSC2000. 2000 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 00TH8493)*, pp. 46–51, IEEE, 2000.

[69] J.-W. Hsieh, S.-H. Yu, Y.-S. Chen, and W.-F. Hu, "Automatic traffic surveillance system for vehicle tracking and classification," *IEEE Transactions on intelligent transportation systems*, vol. 7, no. 2, pp. 175–187, 2006.

[70] R. P. Avery, Y. Wang, and G. S. Rutherford, "Length-based vehicle classification using images from uncalibrated video cameras," in *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No. 04TH8749)*, pp. 737–742, IEEE, 2004.

[71] P. Ji, L. Jin, and X. Li, "Vision-based vehicle type classification using partial gabor filter bank," in *2007 IEEE International Conference on Automation and Logistics*, pp. 1037–1040, IEEE, 2007.

[72] M. Kafai and B. Bhanu, "Dynamic bayesian networks for vehicle classification in video," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 1, pp. 100–109, 2011.

[73] M. Won, "Intelligent traffic monitoring systems for vehicle classification: A survey," *IEEE Access*, vol. 8, pp. 73340–73358, 2020.

[74] Z. Chen, T. Ellis, and S. A. Velastin, "Vehicle detection, tracking and classification in urban traffic," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pp. 951–956, IEEE, 2012.

[75] H. C. Karaimer, I. Cinaroglu, and Y. Bastanlar, "Combining shape-based and gradient-based classifiers for vehicle classification," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pp. 800–805, 2015.

[76] N. C. Mithun, N. U. Rashid, and S. M. Rahman, "Detection and classification of vehicles from video using multiple time-spatial images," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1215–1225, 2012.

[77] Z. Dong, Y. Wu, M. Pei, and Y. Jia, "Vehicle type classification using a semisupervised convolutional neural network," *IEEE transactions on intelligent transportation systems*, vol. 16, no. 4, pp. 2247–2256, 2015.

[78] Y. Peng, J. S. Jin, S. Luo, M. Xu, and Y. Cui, "Vehicle type classification using pca with self-clustering," in *2012 IEEE International Conference on Multimedia and Expo Workshops*, pp. 384–389, IEEE, 2012.

[79] H. Huttunen, F. S. Yancheshmeh, and K. Chen, "Car type recognition with deep neural networks," in *2016 IEEE intelligent vehicles symposium (IV)*, pp. 1115–1120, IEEE, 2016.

[80] Y. O. Adu-Gyamfi, S. K. Asare, A. Sharma, and T. Titus, "Automated vehicle recognition with deep convolutional neural networks," *Transportation Research Record*, vol. 2645, no. 1, pp. 113–122, 2017.

[81] Z. Luo, F. Branchaud-Charron, C. Lemaire, J. Konrad, S. Li, A. Mishra, A. Achkar, J. Eichel, and P.-M. Jodoin, "Mio-tcd: A new benchmark dataset for vehicle classification and localization," *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 5129–5141, 2018.

[82] W. Liu, M. Zhang, Z. Luo, and Y. Cai, "An ensemble deep learning method for vehicle type classification on visual traffic surveillance sensors," *IEEE Access*, vol. 5, pp. 24417–24425, 2017.

[83] Z. Ma, D. Chang, J. Xie, Y. Ding, S. Wen, X. Li, Z. Si, and J. Guo, "Fine-grained vehicle classification with channel max pooling modified cnns," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3224–3233, 2019.

[84] L. Yang, P. Luo, C. Change Loy, and X. Tang, "3d object representations for fine-grained categorization," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, pp. 554–561, 2013.

[85] L. Yang, P. Luo, C. Change Loy, and X. Tang, "A large-scale car dataset for fine-grained categorization and verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3973–3981, 2015.

[86] C. Ju, A. Bibaut, and M. van der Laan, "The relative performance of ensemble methods with deep convolutional neural networks for image classification," *Journal of Applied Statistics*, vol. 45, no. 15, pp. 2800–2818, 2018.

[87] M. A. Hedeya, A. H. Eid, and R. F. Abdel-Kader, "A super-learner ensemble of deep networks for vehicle-type classification," *IEEE Access*, vol. 8, pp. 98266–98280, 2020.

[88] B. Neupane, T. Horanont, and J. Aryal, "Real-time vehicle classification and tracking using a transfer learning-improved deep learning network," *Sensors*, vol. 22, no. 10, 2022.

[89] G. Jocher, K. Nishimura, T. Mineeva, and R. Vilari no, "Yolov5," 2020. Available online: https://github.com/ultralytics/yolov5 (accessed on 26 August 2021).

[90] J. Zhao, S. Hao, C. Dai, H. Zhang, L. Zhao, Z. Ji, and I. Ganchev, "Improved vision-based vehicle detection and classification by optimized yolov4," *IEEE Access*, vol. 10, pp. 8590–8603, 2022.

[91] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

[92] B.-Y. Sun, X.-M. Zhang, J. Li, and X.-M. Mao, "Feature fusion using locally linear embedding for classification," *IEEE transactions on neural networks*, vol. 21, no. 1, pp. 163–168, 2009.

[93] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[94] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10781–10790, 2020.

[95] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, and S. Lyu, "Ua-detrac: A new benchmark and protocol for multi-object detection and tracking," *Computer Vision and Image Understanding*, vol. 193, p. 102907, 2020.

[96] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving video database with scalable annotation tooling," *arXiv preprint arXiv:1805.04687*, vol. 2, no. 5, p. 6, 2018.

[97] A. Ćorović, V. Ilić, S. Durić, M. Marijan, and B. Pavković, "The real-time detection of traffic participants using yolo algorithm," in *2018 26th Telecommunications Forum (TELFOR)*, pp. 1–4, 2018.

[98] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[99] Z. Hu and K. Uchimura, "Uv-disparity: an efficient algorithm for stereovision based scene analysis," in *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, pp. 48–54, IEEE, 2005.

[100] X. Xu, J. Amaro, S. Caulfield, G. Falcao, and D. Moloney, "Classify 3d voxel based point-cloud using convolutional neural network on a neural compute stick," in *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pp. 37–43, IEEE, 2017.

[101] J. Leng, Y. Liu, D. Du, T. Zhang, and P. Quan, "Robust obstacle detection and recognition for driver assistance systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 4, pp. 1560–1571, 2020.

[102] G. Li, S. E. Li, R. Zou, Y. Liao, and B. Cheng, "Detection of road traffic participants using cost-effective arrayed ultrasonic sensors in low-speed traffic situations," *Mechanical Systems and Signal Processing*, vol. 132, pp. 535–545, 2019.

[103] A. Prakash, K. Chitta, and A. Geiger, "Multi-modal fusion transformer for end-to-end autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7077–7087, June 2021.

[104] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*, pp. 1–16, PMLR, 2017.

[105] C.-C. Lin, C.-H. Kuo, and H.-T. Chiang, "Cnn-based classification for point cloud object with bearing angle image," *IEEE Sensors Journal*, vol. 22, no. 1, pp. 1003–1011, 2022.

[106] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

[107] R. Futrzynski, "How to get meaning from text with language model bert — ai explained." Accessed 26 August 2022.

[108] W. contributors, "Transformer (machine learning model)." `https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)`, 2019. [Online; accessed 28-August-2022].

[109] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[110] A. Steiner, A. Kolesnikov, X. Zhai, R. Wightman, J. Uszkoreit, and L. Beyer, "How to train your vit? data, augmentation, and regularization in vision transformers," *CoRR*, 2021.

[111] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," 2020.

[112] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollar, "Designing network design spaces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[113] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *CoRR*, vol. abs/1905.11946, 2019.

[114] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *CoRR*, vol. abs/2103.14030, 2021.

[115] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[116] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ade20k dataset," *International Journal of Computer Vision*, vol. 127, no. 3, pp. 302–321, 2019.

[117] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," *CoRR*, vol. abs/2104.14294, 2021.

[118] H. Bao, L. Dong, and F. Wei, "Beit: BERT pre-training of image transformers," *CoRR*, vol. abs/2106.08254, 2021.

[119] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *CoRR*, vol. abs/2005.14165, 2020.

[120] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever, "Generative pretraining from pixels," in *International conference on machine learning*, pp. 1691–1703, PMLR, 2020.

[121] B. Crider, "Tesla full self driving is using gpt for vision - dr. know it all explains what this means," Jun 2022.

[122] S. S. Beauchemin, M. A. Bauer, T. Kowsari, and J. Cho, "Portable and scalable vision-based vehicular instrumentation for the analysis of driver intentionality," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 2, pp. 391–401, 2011.

[123] D. R. A. of Canada, "Cedar." `https://docs.alliancecan.ca/wiki/Cedar`, November 2016. Accessed: 2022-08-28.

[124] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[125] F. Morales, "vit-keras." `https://github.com/faustomorales/vit-keras`, November 2020. Accessed: 2022-08-28.

[126] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi, *et al.*, "Keras-tuner." `https://github.com/keras-team/keras-tuner`, 2019.

[127] M. A. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S. A. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, F. Xie, and C. Zumar, "Accelerating the machine learning lifecycle with mlflow," *IEEE Data Eng. Bull.*, vol. 41, pp. 39–45, 2018.

[128] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," *Journal of machine learning research*, vol. 13, no. 2, 2012.

[129] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *CoRR*, vol. abs/1710.09412, 2017.

[130] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[131] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010* (Y. Lechevallier and G. Saporta, eds.), (Heidelberg), pp. 177–186, Physica-Verlag HD, 2010.

[132] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[133] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[134] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical data augmentation with no separate search," *CoRR*, vol. abs/1909.13719, 2019.

[135] HuggingFace and contributors, "Models." https://huggingface.co/models?pipeline_tag=image-classification&sort=downloads, 2021. [Online; accessed 31-August-2022].

Curriculum Vitae

Name:	Andrew Katoch
Post-Secondary Education and Degrees:	The University of Western Ontario London, Ontario, Canada 2015-2020 B.Sc.
Honours and Awards:	None
Related Work Experience:	Teaching Assistant The University of Western Ontario 2020-2022