Evaluating Machine Learning Models for Traffic Sign Classification

Project Final Report

Team Members: Nathan Van Utrecht, Patrick Whitehouse

Abstract

This paper evaluates the performance of various machine learning models in classifying traffic signs under challenging conditions, specifically focusing on naturally occurring occlusions that autonomous vehicles may encounter. We use the German Traffic Sign Recognition Benchmark (GTSRB) dataset and introduce random occlusion augmentation to simulate real-world scenarios, ensuring that models must robustly handle partially obstructed signs. Our study compares the performance of five approaches: K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Convolutional Neural Networks (CNN), ResNet-50, and Vision Transformer (ViT). By leveraging data augmentation, transfer learning, and model tuning, we explore the trade-offs between accuracy and inference time, both critical factors in autonomous driving applications. Experimental results show that deep learning models, particularly those leveraging pretrained architectures such as ResNet-50, achieve high accuracy. The CNN model provides an excellent balance of accuracy and speed. Our findings guide future research and practical model selection for real-time, reliable traffic sign recognition systems.

1 Introduction

In recent years, autonomous vehicles have become an increasingly viable technology, offering the potential to transform transportation systems worldwide. However, for these vehicles to operate safely, they must reliably recognize and respond to traffic signs under various conditions, including situations where objects may be occluded due to natural obstructions, such as tree branches across signs or larger vehicles occluding view. Accurate identification of traffic signs despite the natural occlusion that may occur is critical to ensuring that autonomous vehicles can make safe navigation decisions. There is also the added constraint of inference time as decisions need to be made quickly when traveling at a high rate of speed. This project addresses this need by developing a machine learning system to classify traffic signs, emphasizing accounting for occlusion, thereby enhancing the reliability and safety of autonomous driving.

The primary task in this project is traffic sign classification. We aim to implement machine learning algorithms that can detect and classify traffic signs in images, focusing on scenarios where lighting is compromised. To achieve this, we used the German Traffic Sign Recognition Benchmark (GTSRB). This dataset consists of 43 different traffic sign classes, totaling 50 thousand images. To simulate occlusion, we modified the images by randomly introducing black squares to half of the training images. This approach allows us to evaluate the performance of our machine learning models in these conditions, ensuring a robust assessment of each model's ability to handle challenging visual scenarios.

Our project will compare five machine learning models: K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Convolutional Neural Networks (CNN), ResNet-50, and Vision Transformer (ViT). These models represent a range of techniques within the field, from traditional machine learning algorithms to deep learning methods. SVM and KNN offer simplicity and interpretability, while CNNs provide the ability to implicitly learn complex hierarchical features from images. ResNet-50 and ViT were both pre-trained on the ImageNet dataset [1], which means that we were able to leverage low-level feature extractors (edges, lines, shapes, etc.) while fine-tuning the deeper layers on our specific dataset. By evaluating these models, we can determine which approach is best suited for traffic sign classification, given naturally occurring scenarios.

We expect that the deep learning models will outperform both SVM and KNN due to their ability to capture intricate details and patterns within the images. ResNet-50 and ViT are anticipated to perform the best due to their foundational architecture, with CNN being right behind them. We anticipate that SVM will perform better than KNN because of its robust mathematical foundation, but it may still fall short compared to CNNs. Overall, our project aims to highlight the strengths and limitations of each model, providing insights into the most effective strategies for traffic sign classification.

2 Related Work

Image classification is currently a rapidly evolving field. Deep Learning algorithms leverage having many layers to extract features from the images. Rahul Chauhan et al.[2] applied the use of CNNs to MNIST and CIFAR-10, where they saw 99.6% accuracy and 80.10%, respectively. They leveraged dropout to reduce the chance of overfitting the data to achieve these results. They also used max pooling. Max pooling takes the maximum value of a 2x2 or 3x3 grid, maximizing the signal. As a result, more dominant features, such as edges, become more explicit for the network to capture. This applies directly to street signs, as street signs have clean edges.

KNN and SVM are not traditional image classification models. For use in image classification, preprocessing is required to extract features from the images. In Jinho Kim et al. [3] work, the images had features extracted using a bag of words, which in this case consisted of key features in the image. These features were then used to create a histogram that both the KNN and SVM used as a basis for classification. The dataset consisted of 2800 training images and 700 testing images across 4 classes. The KNN saw an accuracy of 78.03%, and the SVM saw an accuracy of 91.9%. While this work had significantly less classes than what is used in this work, it displayed the capability of KNN and SVM when paired with a feature extractor.

Vision transformers are a deep learning model that leverages the attention mechanism initially developed for natural language processing to emphasize key features in the image. In Yuping Zheng et al. [4] the same dataset was used and evaluated across 5 different vision transformers. The results of the vision transformers can be seen in Table 1 Their work shows a clear ability for Vision Transformers in street sign classification.

Table 1: Performance of Vision Transformers on Training, Validation, and Testing Sets

Model	**Training**	**Validation**	**Testing**
ViT	98.27%	98.89%	83.77%
ViT (RealFormer)	98.45%	99.19%	86.03%
ViT (Sinkhorn Transformer)	94.69%	97.04%	82.29%
ViT (Nyströmformer)	79.15%	83.15%	62.41%
TNT	96.83%	97.73%	84.39%

To improve model generalization in image classification, image augmentation approaches have become commonplace. When considering the scope of the task where, a variety of objects, such as branches or other vehicles can often occlude traffic signs. In Zhun Zhong et al. [?], they suggested randomly erasing sections of the image. The benefits of this method include maintaining the overall structure of the object while removing small sections of the object to increase the reliability of the model. In their work using the CIFAR-10 dataset with ResNet18 they were able to decrease error from 5.17% to 4.31%. This idea, applied in a realm with the added benefit of mimicking real-world scenarios, shows significant potential.

3 Background

3.1 Dataset

The German Traffic Sign Recognition Benchmark (GTSRB) dataset is widely recognized for its role in research in traffic sign image classification. Consisting of 50 thousand images across 43 classes with the distribution seen in Figure 1. The dataset was introduced in 2011 during the International Joint Conference on Neural Networks, where the dataset was part of a competition for participants at the conference. An example image from each of the 43 classes can be seen in Figure 2, showcasing the complexity of the signs present in the dataset.

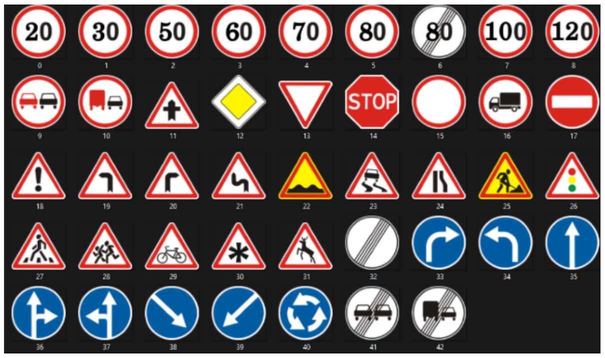

Figure 1: Example of 43 classes

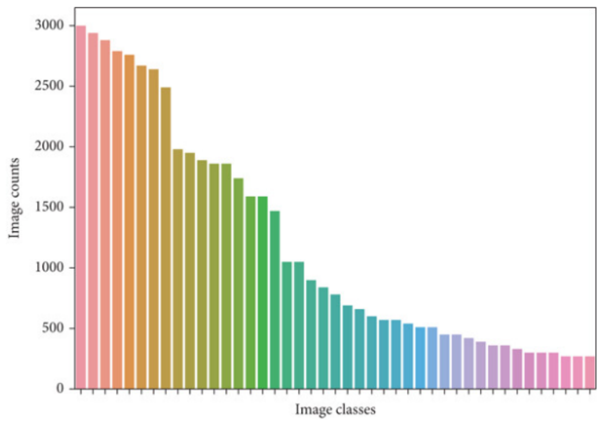

Figure 2: [4] Distribution across classes

3.2 Data Processing

In image classification, the ability to generalize from seen information to unseen information is fundamental in achieving high accuracies, particularly by preventing overfitting, [5] or the memorization of the training data. This is particularly important when applied to real-world scenarios, where, in many cases, there can be infinite depictions of an object based on various circumstances, such as the amount of occlusion, the light levels, and the backdrop of the object. As a result, augmentation of testing images has become a strategy to increase the adaptability of image classification models.

In an effort to identify an augmentation that closely resembles a real-world scenario the model might encounter it was important the method was stochastic. Due to the random nature of occlusion seen in the real world, a similarly random approach seemed adequate. In a work by Zhun Zhong et al. [6], it was proposed to erase portions of an image randomly. By eliminating smaller sections while preserving the overall structure of the object, it was found that the models became more robust.

Maintaining the idea that the augmentation used should mimic real-world scenarios that could be encountered. As a result, the Random Occlusion method was made to occlude various parts of the image with various randomly sized squares. This was done by placing black squares randomly across the image.

The size of the squares was kept relative to the size of the input image, where the size ranged between $\frac{1}{6}$ and $\frac{1}{12}$ of minimum between the height and width of the image. Each augmented image had seven squares randomly placed on the base image, resulting in the occlusion seen in Figure 3.

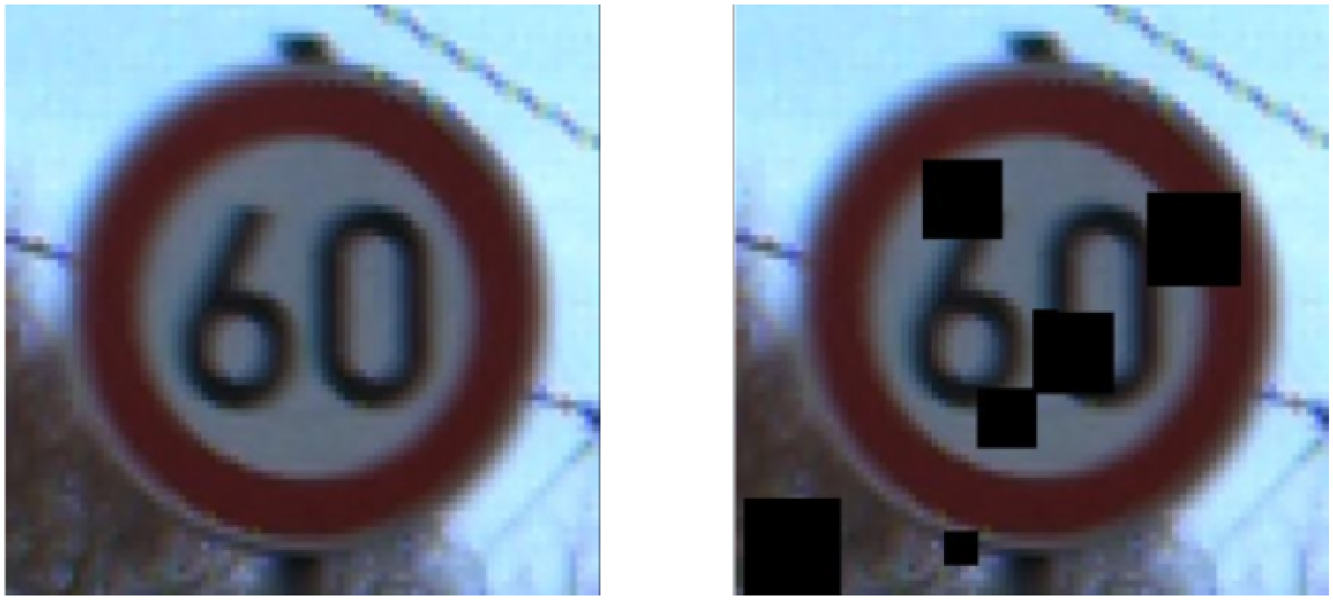

Figure 3: The left image is the base data, and the one on the right is after the random occlusion augmentation.

3.3 K-Nearest Neighbors

The K-Nearest Neighbors model [7] is a classic supervised machine learning algorithm for classification. The premise is that data points that are similar in feature space are likely to be part of the same class. When classifying a new data point, the K most similar data points majority vote for the new point's class where K is as hyperparameter indicating the number of neighbors. In addition to the K number of neighbors, the other two important hyperparameters are how similarity is measured and how voting works. There are different options for similarity computation, such as Euclidean distance, Manhattan distance, and cosine similarity. For voting, there are two main approaches - assign equal weight to all neighbor votes or weight the votes based on similarity to the new data point. These two techniques are known as uniform and distance voting, respectively.

Although KNN does not require training (it is just comparing the similarity to the training dataset), it still requires hyperparameter tuning. We performed a grid search over the hyperparameters in Table 2 and found the optimal hyperparameters to be $k = 5$, Manhattan distance for similarity, and distance weighting for voting. Given that our metric was accuracy, these parameters make sense. The number of neighbors strikes a balance between not enough information to compare and too much information to the point where some of it harming performance. Given the high dimensionality of images, we used PCA reducation on the dataset to capture key features of the images. We tested a model on 50, 100, and 150 component datasets, and we found that the 150 component model worked the best. For our hyperparameters, it makes sense that Manhattan distance performed better since it measures axis-aligned differences, avoiding the issues of the "curse of dimensionality" that can make Euclidean distance less reliable by overemphasizing diagonal relationships. Distance voting also makes sense over uniform voting because it gives higher weight to the effectively more important data points. However, Manhattan distance and distance voting require extra computation time compared to their counterparts, and given that autonomous driving is a time-crucial operation, it would be interesting to further study the trade-off between performance and inference time.

Table 2: K-Nearest Neighbor Grid Search Hyperparameters

Hyperparameter	**Definition**	**Values**
k	Number of neighboring data points to consider	3, 5, 7, 9
similarity	How the similarity between two data points is defined	Manhattan, Euclidean
voting	How the votes are weighted	Uniform, Distance

3.4 Support Vector Machine

Support Vector Machines [8] are another classic supervised machine learning algorithm for classification. The core idea of SVM is to find a hyperplane that best separates data points from different classes in the feature space. This hyperplane is chosen to maximize the margin, which is the distance between the hyperplane and the closest data points from each class, known as support vectors. The larger the margin, the better the generalization to unseen data.

SVM has two key hyperparameters: the regularization parameter (C) and the kernel function. The C parameter is what controls the size of the margin. A small C will allow a wider margin at the expense of some misclassified training samples while a large C has a tighter margin with less misclassifications. The optimal C is one that balances a wide margin for generalization while still accurately classifying data points. One drawback of SVM is that it assumes the data is linearly separable. The kernel function can be used to map the data into a higher-dimensional space to make it linearly separable. Common functions include linear, polynomial, radial basis function (RBF), and sigmoid.

To find the optimal hyperparameters for our implementation, we performed a grid search over the values found in Table 3. We initially tried using different kernel functions, but given the dimensionality/size of our dataset and CPU implementation of scikit, we just used the linear kernel function. Similar to KNN, we fit an SVM model on a 50, 100, and 150 PCA component dataset. The model performed the best on the 150 component dataset which makes sense given it provides more information. There was not a noticeable shift in inference time either, so we will report only the results of the 150 component model. We found that a C value of 1 was most optimal which makes sense given that it achieves the balance discussed earlier.

Table 3: Support Vector Machine Grid Search Hyperparameters

Hyperparameter	**Definition**	**Values**
C	Regularization parameter that controls margin size	0.01, 0.1, 1, 10
kernel	Which kernel function to use to map data to a higher dimensionality	Linear

3.5 Convolutional Neural Network

Convolutional Neural Networks were originally developed by LeCun et al. [2] as a way to perform image recognition on the handwritten MNIST dataset [9] without the use of hand-crafted feature extractors. They work by sliding a learned kernel over an image, summing the products of the input pixel values and the learned kernel values, and mapping this value to a new image. Combined with pooling layers, this effectively allows the model to learn spatial correlations between different pixels and embed the image into a lower dimensional space. This low dimensional representation is then flattened and fed through a traditional multi-layer perceptron [10] to produce the probability of each class.

For our implementation, we stacked five convolutional layers on top of one another to make a deep model that can effectively capture different image features. Each convolutional layer is followed by a batch normalization layer, then a ReLU activation function, then a dropout layer, and finally a max pooling layer. The architecture can be seen in Figure 4. Due to the size of our dataset, we wanted to ensure that the model didn't overfit to the training data. This is why we decided to add dropout layers in addition to the data augmentation discussed in Section 3.2.

The hyperparameters, which were found through manual tuning, can be found in Table 4. One thing we considered while developing the model architecture was the trade-off between model performance and inference time. As we have extensively discussed, autonomous driving is a time-sensitive operation, so the more images we can process per second, the more reliable the car will be. This is why we stopped at adding five layers, although we theoretically could use a model like Visual Geometry Group (VGG) [11] and see better performance.

Table 4: Convolutional Neural Network Hyperparameters

Hyperparameter	**Definition**	**Values**
n	The number of training epochs	10
loss function	The function used to compute the model loss	Cross entropy
optimizer	The optimization method used for back-propagation	Adam
η	The learning rate for the model	0.001
ρ	Probability of dropout	0.2

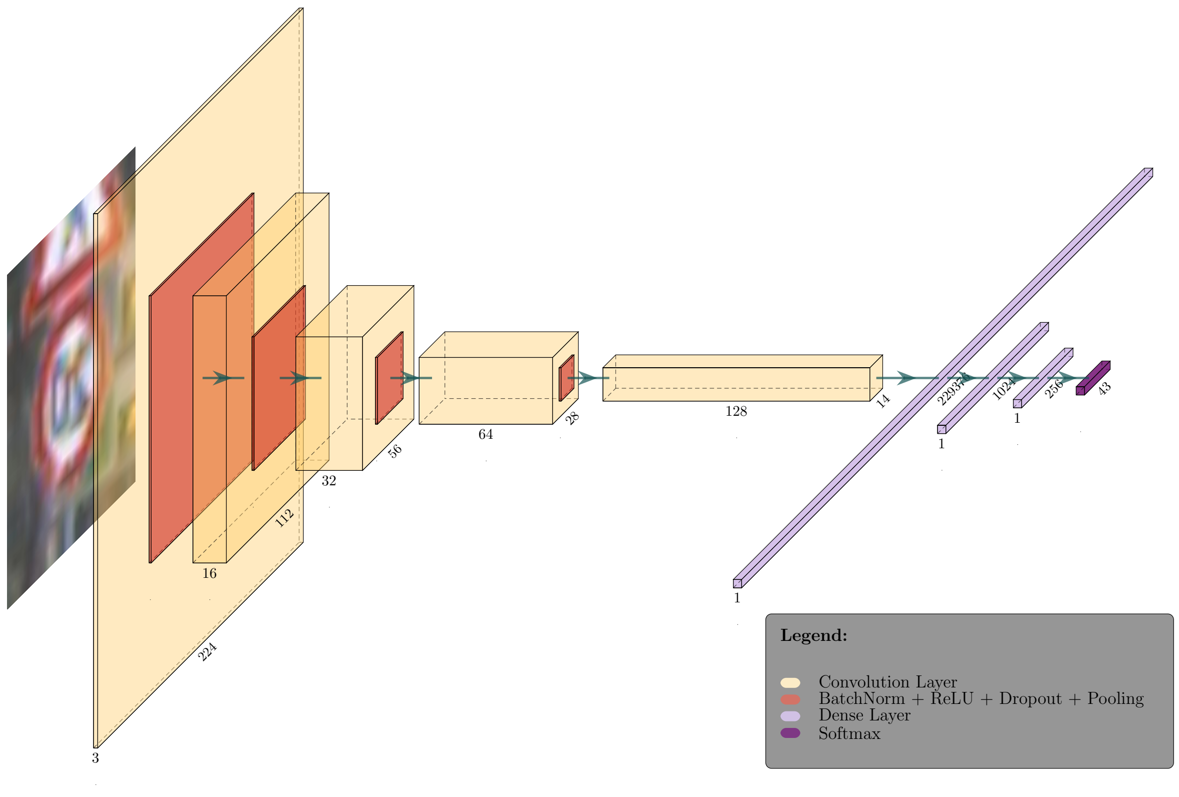

Figure 4: Architecture for CNN model

3.6 ResNet-50

The main issue with traditional CNNs is the vanishing gradient problem. Since each layer requires matrix multiplication for back-propagation, the gradient will either tend toward 0 or ∞ in deep CNNs depending on whether the loss is below or above one. Residual Networks (ResNet) [12] get around this by having skip connections like the one in Figure 5.

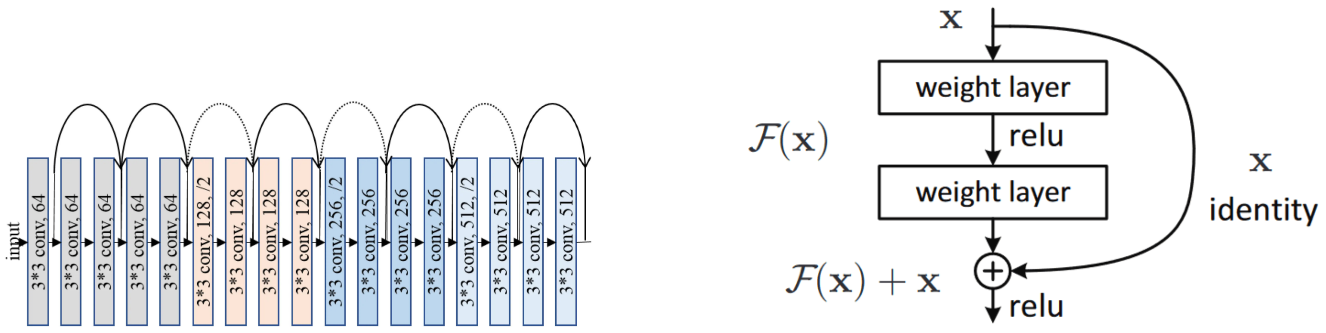

Figure 5: The ResNet architecture is on the left and the residual skip connection diagram is on the right

The connection allows layers that the model deems useless to behave like an identity mapping ($\mathcal{F}(x) \approx 0$). The skip connections also allow improve gradient flow which preserves gradient magnitude and enables effective training of very deep networks. As previously mentioned, we did not train a ResNet model from scratch. There are different families of pretrained ResNet models that have been trained on the ImageNet [1] dataset. We chose the largest model, ResNet-50. The 50 indicates the total number of layers in the model (49 convolutional and 1 fully connected). Using a pre-trained model allows us to leverage low-level feature extractors that are shared across images (edges, lines, shapes, etc.) while fine-tuning the deeper layers on our specific traffic sign dataset.

For the transfer learning process, we changed the dimensions of the last fully connected layer so that it outputs 43 classes, and we froze every layer besides the last two. The hyperparameters can be found in 5. We chose a lower learning rate for the convolutional layers because we didn't want to lose all of the information gained from training on ImageNet. The fully connected layer learning rate is higher though because the weights from ImageNet are useless for our problem. One direction for future research would be investigating the trade-off between performance and inference time with the smaller ResNet models.

Table 5: ResNet-50 Hyperparameters

Hyperparameter	**Definition**	**Values**
n	The number of training epochs	5
loss function	The function used to compute the model loss	Cross entropy
optimizer	The optimization method used for back-propagation	Adam
η_{fc}	The learning rate for the fully connected layer	0.001
η_{res}	The learning rate for the unfrozen residual layers	0.0001

3.7 Vision Transformer

While convolutional layers have consistently proven to be great feature extractors, they are fundamentally limited by their design to only be local in scope. Convolutional filters are applied to small receptive fields which means they struggle to capture global features without the use of very deep models. The architecture of attention layers solves this since they are able to model relationships between any pair of elements in an input sequence regardless of their spatial distance. This concept led to the transformer [13] architecture which has fueled numerous breakthroughs in the realms of NLP and sequence modeling. Since pixels often have spatial correlation (i.e. pixels near each other typically represent the same object), Dosovitskiy et al. [14] adapted the transformer architecture for images.

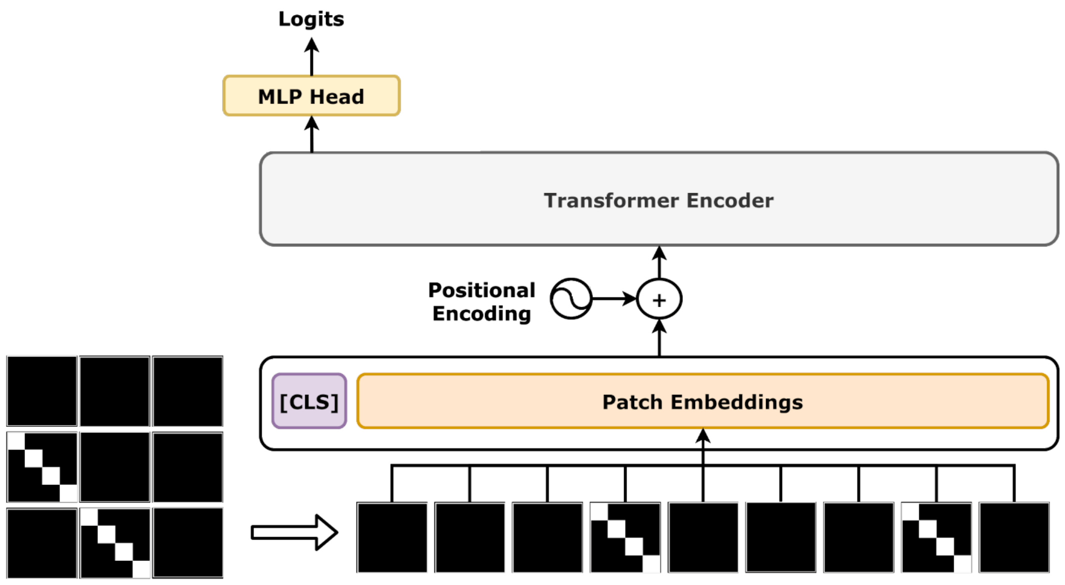

Figure 6: Simplified vision transformer architecture

The ViT architecture can be seen in Figure 6. The image is split up into patches whose size depends on the model being used. Multiple models were trained on the ImageNet dataset, and we used the base model with a patch size of 16x16. These patches are then fed through learned embeddings to reduce their dimensionality. Afterwards they go through positional embedding, a technique introduced in [13], which helps the model understand the spatial relationships between the patches. These features are then fed into the transformer encoder which is finished off by an MLP head that predicts the classes.

For the transfer learning process, we changed the dimension of the MLP head so that it outputs 43 classes, and we froze every encoder block besides the last four. Similar to our ResNet-50 training process, we chose a lower learning rate for the encoder blocks than we did for the MLP head. This ensured that the encoder blocks didn't forget everything from their ImageNet pretraining while making sure that the MLP head learns new relationships. The hyperparameters can be found in Table 6

Table 6: ViT Hyperparameters

Hyperparameter	**Definition**	**Values**
n	The number of training epochs	10
loss function	The function used to compute the model loss	Cross entropy
optimizer	The optimization method used for back-propagation	Adam
η_{fc}	The learning rate for the MLP head	0.001
η_{res}	The learning rate for the unfrozen encoder blocks	0.0001

4 Experimental Results

When evaluating models for this problem, there were two key metrics to keep track of. Primarily, there was the accuracy of the model as the correct decision can only be made when an image is classified correctly. Secondarily, there was the inference time. As seen in Figure **??** when traveling at 80 km/h, the braking distance is staggering. This dataset includes a 120 km/h speed limit class and originates from Germany, where some stretches of the Autobahn have no enforced speed limit and vehicles commonly travel at over 200 km/h. In light of these high speeds, achieving fast inference times is especially crucial. With traffic signs such as speed limits enforced by photo, proper classification and timely recognition are vital. In the following sections, we will present both of these metrics for each model, offer some intuition as to why they got these results, and give some potential future research directions.

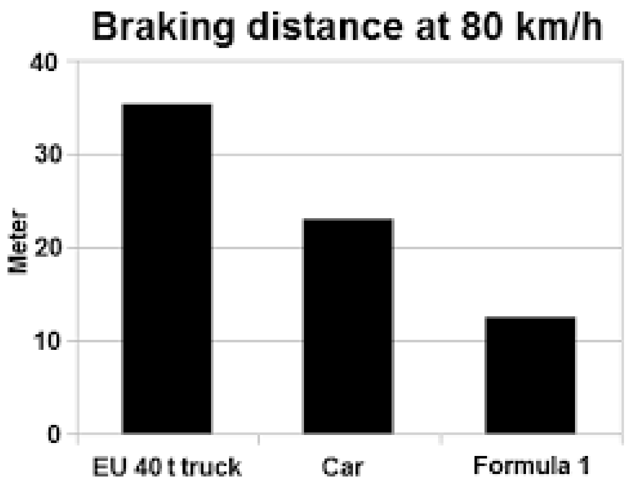

Figure 7: The amount of distance needed to brake for different vehicles at 80 km/h

4.1 K-Nearest Neighbor

The KNN model achieved a test accuracy of 50.51%, which is alarmingly low, given the importance of accuracy in this domain. This result is likely due to the similarity between speed signs where the only difference is the number in the middle, resulting in shared features across several classes. Due to the large number of distance computations, the model had an inference time of 1.3251 ms.

One area of future research would be pairing KNN with an autoencoder. Since KNN is so heavily reliant on the quality of the dataset, it is likely that PCA reduction did not adequately capture the key features in the images. Using an autoencoder, like the one described in [15], could allow better feature representation which would improve model performance. Given that KNN had a relatively quick inference time, the added overhead of an autoencoder likely would not make the new inference time unviable.

4.2 Support Vector Machine

When looking into the viability of SVM it was found that the model was able to achieve an accuracy of 73.23%. While this is not particularly stunning, the actual upside of this model comes in the form of the inference time, where the model was able to make decisions in 0.0005 ms. This makes the model valuable for situations where speed is of the utmost importance. We believe that the subpar accuracy comes from the choice of kernel. As discussed in Section 3.4, we had to use a linear kernel given the dimensionality and size of the dataset. This means that the model likely struggled to learn non-linear patterns like textures and shapes. The quick inference time comes from only needing to compute dot products with the few support vectors.

Given these outcomes, there are a few research directions that we would like to explore in the future. Scaling the dataset down to only a few hundred images per class would allow use of different kernel mappings that can capture non-linear relationships. Although SVMs are typically considered strong individual learners, we could also try to ensemble multiple models together to improve reduce variance in the predictions. Since the inference time is so quick, there is room to explore increasing the model complexity while still being able to predict new images in a timely manner.

4.3 Convolutional Neural Network

The custom CNN model demonstrated an impressive testing accuracy of 95.93%. This showed a significant improvement over the SVM and KNN models, highlighting why CNN's have become so popular in the image classification space, with their ability to extract special features from images. The custom CNN was able to achieve a fast inference time of 0.9763 ms making the model well-suited for real-time traffic sign recognition due to the combination of high accuracy and speed, which are essential for this domain.

The training metrics are displayed in Figure 8. Besides the fundamental ability of CNNs to perform image classification, we believe that our model especially excelled because of two things: the dataset and the architecture. The dataset is large, clean, and relatively unbiased. Coupled with our augmentation, this gave the model an abundance of high-quality data to learn from. With this much data comes the risk of overfitting. However, we avoided this by using batch normalization and dropout to regulate the model.

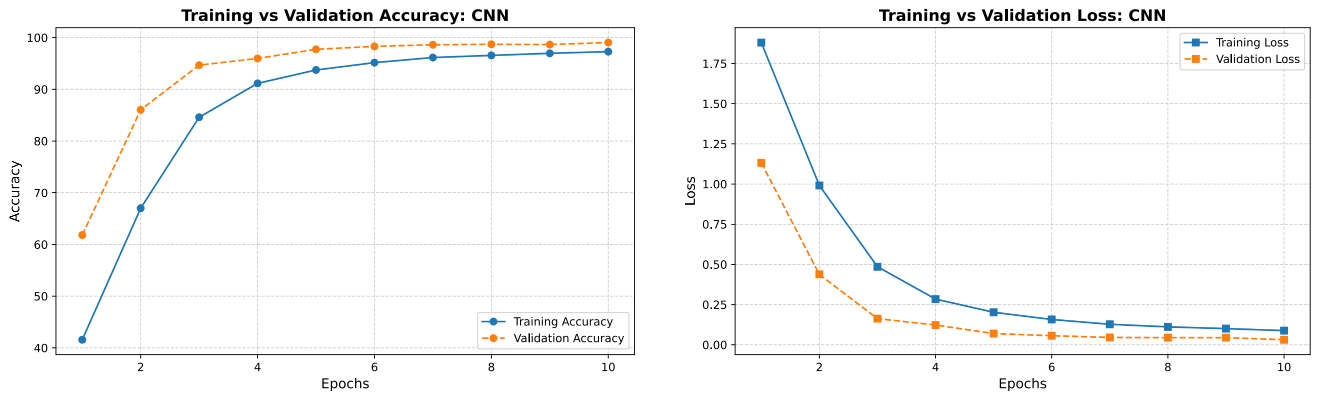

Figure 8: CNN training and validation accuracies and losses.

Although the model performed well, we believe that a human would perform better than 95.93%. In order to boost performance, we could add a few more convolutional layers. This would allow the model to learn the tricky images that it is currently failing on. Although it would increase inference time, we don't anticipate that being an issue given how fast it already is.

4.4 ResNet-50

ResNet-50, an advanced CNN model, was able to achieve an impressive testing accuracy of 98.18%. This near-perfect result came at the cost of a relatively high inference time with respect to the other models at 4.5004 ms per prediction. The high accuracy makes ResNet-50 a strong contender for any situation where the highest accuracy is important. However, the higher computational cost of many layers could present a problem in situations where speed is essential.

The ResNet-50 training metrics are displayed in Figure 9. Although certainly not the first to come to this conclusion, we believe that ResNet-50 performed so well because of its ability to leverage low-level feature extractors. All images share common characteristics like edges and lines. Since ResNet-50 was trained on such a large corpus of images, it has really accurate low-level feature extractors. This makes transfer learning especially effective.

Although the inference time seems relatively fast (it could still process $\tilde{2}50$ images per second), this test was done an A100 GPU. Autonomous cars likely will not have access to such powerful hardware, so the inference time could be much slower. At high speeds, this is potentially problematic. As mentioned in Section 3.6, we used the largest model of the ResNet family. One interesting future research direction would be testing the performance of the ResNet-18 and ResNet-34 models. Since they are smaller, they would have better inference times, and we anticipate that they would still perform well.

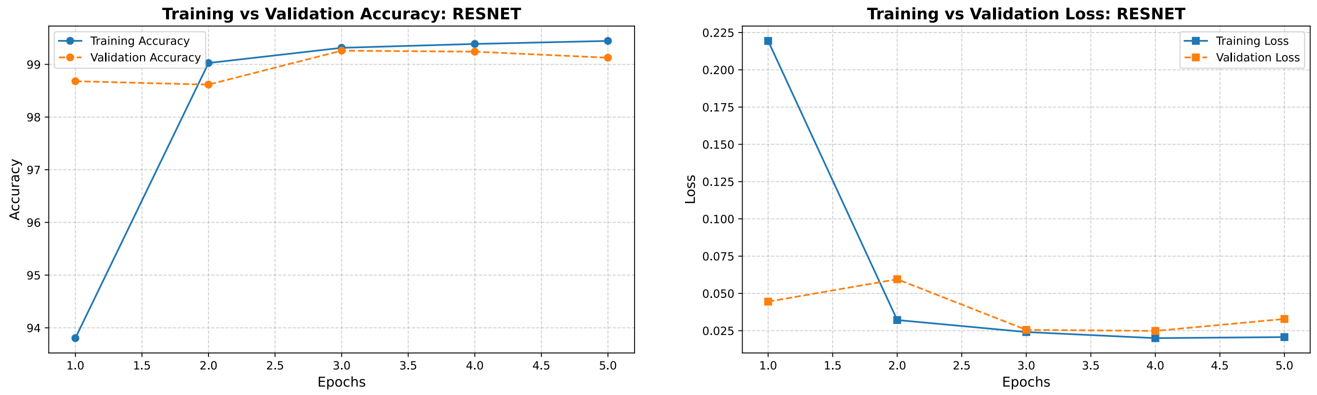

Figure 9: ResNet-50 training and validation accuracies and losses.

4.5 Vision Transformer

The vision transformer model achieved a test accuracy of 91.77%. While slightly underperforming CNN models, the transformer was still able to outperform other vision transformers that had been implemented on the same dataset 1. With this said, the model was also on the upper end in terms of inference time at 3.9087 ms. This model shows promise in the vision transformer space, however, in comparison to CNNs, there is still room for improvement.

The testing metrics for ViT are shown in 10. We believe that the slight underperformance is due to two factors: fundamental limitations of attention layers and dataset size. Convolution layers have an implicit inductive bias because they inherently encode spatial relationships between pixels. On the other hand, attention layers only see patches of pixels which do not have the same structural assumptions as convolution layers. The other reason we believe ViT did not perform as well is because of the dataset size. Transformers were designed to be trained on large corpuses of NLP data like BookCorpus and English

Wikipedia. Although a total of 80,000 images is not objectively small, it is relative to these large NLP datasets.

One interesting future research direction would be trying different transfer learning strategies. We arbitrarily unfroze the last four encoder blocks, so there could be some performance gains that come with fine-tuning this. Doing more augmentation like flipping and random cropping could also artificially inflate our dataset and give the transformer enough data to learn on.

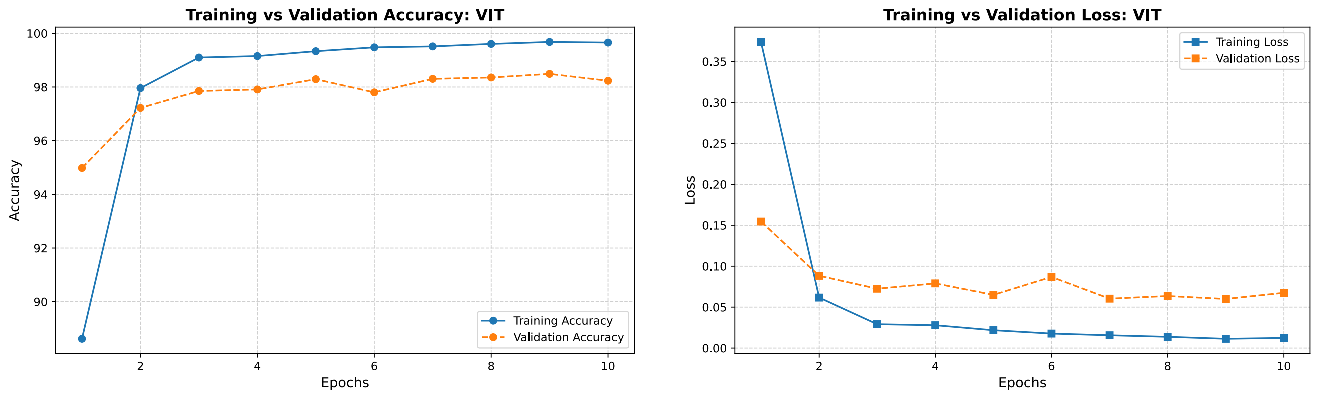

Figure 10: ViT training and validation accuracies and losses.

4.6 Results Summary

The inference time and testing accuracy results are summarized in Figure 11. In terms of accuracy, the traditional machine learning models (KNN and SVM) performed the worst while the advanced deep learning models (CNN, ResNet, and ViT) performed the best. In terms of inference time, SVM was in a league of its own. KNN and CNN were both quick, and unsurprisingly, the large ResNet and ViT models were the slowest. One final interesting research direction would be doing majority voting betweeen CNN, ResNet, and ViT. Although this would likely make the inference time too large, it would be an interesting way to see if the models are learning different features.

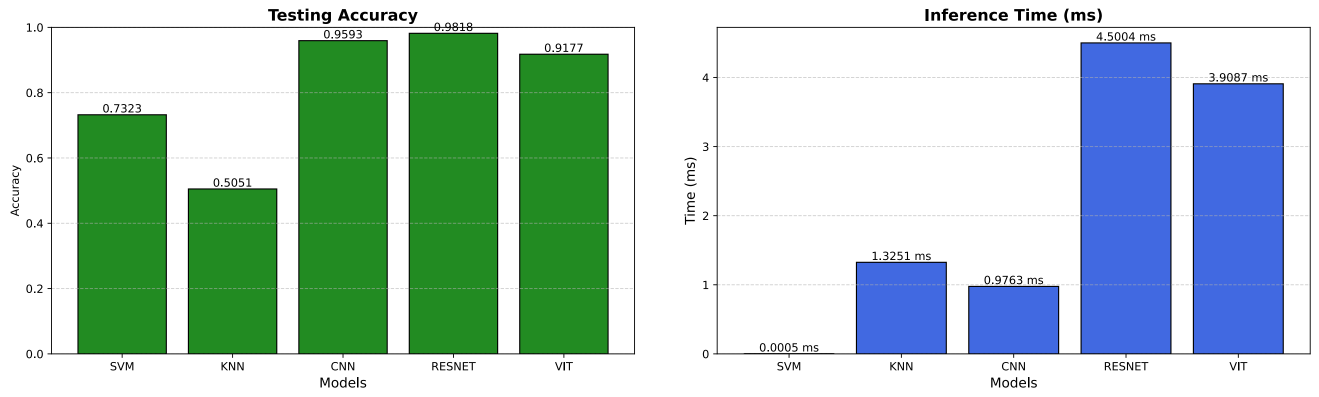

Figure 11: The left image summarizes the testing accuracies for each model, and the right image shows inference times for each model.

5 Conclusion

Overall, while ResNet-50 offers the highest testing accuracy, the higher inference time might prohibit implantation on real-time applications requiring high-speed decision making. The custom CNN struck a balance between having a high accuracy and low inference time, making it the most suitable model when it comes to high-speed traffic sign recognition. The vision transformer, while not able to achieve as high of an accuracy as the CNN models, was still impressive in the vision transformer domain. The SVM, despite the lower accuracy, saw a remarkably low inference time, which could be useful in situations where speed is prioritized over accuracy.

6 References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[2] Y. Lecun, L. Jackel, L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik, *Learning algorithms for classification: A comparison on handwritten digit recognition.* World Scientific, 1995, pp. 261–276.

[3] J. Kim, B.-S. Kim, and S. Savarese, “Comparing image classification methods: K-nearest-neighbor and support-vector-machines,” in *Proceedings of the 6th WSEAS International Conference on Computer Engineering and Applications, and Proceedings of the 2012 American Conference on Applied Mathematics*, ser. AMERICAN-MATH’12/CEA’12. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2012, p. 133–138.

[4] Y. Zheng and W. Jiang, “Evaluation of vision transformers for traffic sign classification,” *Wireless Communications and Mobile Computing*, vol. 2022, no. 1, p. 3041117, 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/2022/3041117

[5] M. Xu, S. Yoon, A. Fuentes, and D. S. Park, “A comprehensive survey of image augmentation techniques for deep learning,” *Pattern Recognition*, vol. 137, p. 109347, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320323000481

[6] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, “Random erasing data augmentation,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 08 2017.

[7] K. Taunk, S. De, S. Verma, and A. Swetapadma, “A brief review of nearest neighbor algorithm for learning and classification,” in *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, 2019, pp. 1255–1260.

[8] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf, “Support vector machines,” *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[9] L. Deng, “The mnist database of handwritten digit images for machine learning research [best of the web],” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[10] F. Rosenblatt, “The perceptron: a probabilistic model for information storage and organization in the brain.” *Psychological review*, vol. 65 6, pp. 386–408, 1958. [Online]. Available: https://api.semanticscholar.org/CorpusID:12781225

[11] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2015. [Online]. Available: https://arxiv.org/abs/1409.1556

[12] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2023. [Online]. Available: https://arxiv.org/abs/1706.03762

[14] 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,” 2021. [Online]. Available: https://arxiv.org/abs/2010.11929

[15] D. Bank, N. Koenigstein, and R. Giryes, “Autoencoders,” 2021. [Online]. Available: https://arxiv.org/abs/2003.05991