IT1244 Project Report: Traffic Sign Recognition

Team 4

Amelia Lon Hwee Min, Kuah Si Ying, Muhammad Zikri Bin Zalizan, Tng Kai Yi

Introduction

Traffic sign recognition (TSR) is essential in autonomous driving, enabling vehicles to interpret road signs and ensure road safety. With the rapid advancement of autonomous technologies, accurate TSR is increasingly important. Recent developments in machine learning and deep learning have led to significant improvements in performance, such as:

- Convolutional Neural Networks (CNN), utilizing pretrained models like MobileNet. The final result was an accuracy of 99.66% (Lim et al., 2023).
- A hybrid approach combining Support Vector Machines and Random Forest, leveraging Histograms of Oriented Gradients and the Color Layout Descriptor for feature extraction (Lim et al., 2023), achieving 93.98% accuracy.
- An Efficient CNN (ENet) that integrates two pretrained models, VGG16 and LeNet. LeNet outperformed VGG16 with accuracy levels of 98.6% and 96.7% accuracy, respectively. (Lim et al., 2023).

We aim to explore and evaluate several deep learning models for traffic sign recognition, deepening our understanding of their performance and applicability in the real world.

Dataset

The dataset contains 3,823 training and 2,583 test images of traffic signs across 24 classes (0–23). We observed variation in image dimensions, class imbalance, and inconsistencies in brightness and sign clarity.

Preprocessing of Dataset

Our image augmentation pipeline integrates multiple transformation techniques, each with varying parameters, which were optimized to produce the best accuracies. By applying augmentation, we aim to ultimately lead to better classification performance.

We first set several important parameters, determining how the final images would be after preprocessing. The target size of the images were set to (224,224), which is the standard input for CNN. The transformation probabilities were also carefully controlled and adjusted to mimic real world variations in image data. Due to imbalances in the dataset, rare classes with a smaller number of images to train with, which is determined by a threshold of 150, would undergo stronger augmentations through the higher probabilities and parameters of transformation techniques. This ensures that the model is exposed to more diverse variations of the rare classes, improving model generalization. Images of lower quality, which is determined by lower resolutions, are dropped to prevent misleading features into the model, ensuring that the training data remains representative. The preprocessing pipeline for training images is as followed:

- Auto cropping of irrelevant background
- Rotation of images
- Contrast adjustment
- Saturation adjustment
- Addition of random noise
- Addition of random blur
- Resizing of images
- Normalization of images (excluded for images that are darker than the determined threshold)

Extra copies of images were added into the rare classes to balance the dataset, ensuring fairness while training the models. Class 15 was downsampled to 65% of the original number, due to the high differences between it and the other classes. The class distribution of the training dataset before and after rebalancing is as shown in Fig.17 and Fig.18 of the appendix.

As for the test images, preprocessing was done to only clean the images, and no augmentation was applied. The preprocessing pipeline for test images is as followed:

- Auto cropping of irrelevant background
- Resizing of images
- Normalization of images

Methods

Custom Convolutional Neural Network (CNN)

The first model (Fig. 5) evaluated in this study is a custom-built CNN inspired by LeNet-5 built to classify images into 24 distinct classes. The architecture consists of 3 convolutional layers, 3 batch normalization layers, 3 max-pooling layers, 5 dropout layers for regularization, and 3 fully connected layers to ultimately predict the class labels.

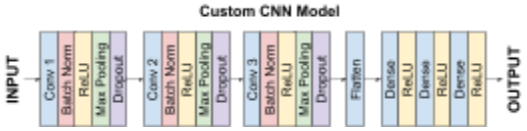

Fig. 5 Custom CNN Model Architecture

Its architecture consists of three convolutional blocks, each followed by batch normalization, ReLU activation, and 2×2 max pooling. The first block uses 6 filters (5×5), reducing the input (224×224×3) to 110×110 feature maps. The second block applies 16 filters (5×5), further downsampling to 53×53, while the third block employs 32 padded filters (3×3), maintaining spatial dimensions before final pooling to 26×26. Dropout regularization (rates: 0.1, 0.2, 0.15) is applied after each block to mitigate overfitting. The flattened features (26×26×32) pass through two fully connected layers (120 and 84 neurons) with increased dropout (0.3, 0.5), culminating in a 24-class output layer.

For training, the model utilised the cross-entropy loss function, optimised using the Adam optimiser with a learning rate of 0.0001 and a weight decay of 0.0005. Adam was chosen for its adaptive learning rate and efficient convergence properties, particularly suited to deep learning classification tasks.

Transfer learning with VGG-16

The VGG-16 is known for its high accuracies in object detection and image classification. The creators enhanced the model by increasing its depth using an architecture with very small (3 × 3) convolution filters, which significantly improved performance over prior configurations.

VGG-16 is a deep learning model known for its high accuracy in object detection and image classification. It features 16 learnable layers, including 13 convolutional layers, five max-pooling layers, and three fully connected layers. The architecture uses small 3×3 convolution filters with stride 1 and same padding, and 2×2 max pooling with stride 2. The number of filters increases from 64 to 512 across the layers, followed by two FC layers with 4,096 units and a final softmax output layer.

In our project, we used a pretrained VGG-16 model trained on ImageNet. We modified the final FC layer to output 24 classes instead of 1,000. The model was implemented using PyTorch's `vgg16`, trained with cross-entropy loss and optimized using Stochastic Gradient Descent.

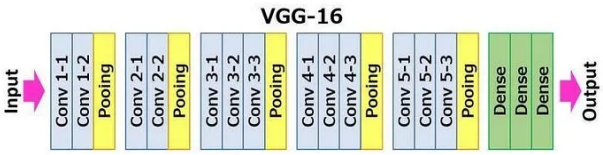

Fig. 6 VGG-16 Model Architecture

Transfer Learning with ResNet-18

The Residual Network (ResNet) architecture, introduced in 2015, was designed to address the vanishing gradient problem, where gradients become too small during backpropagation, slowing down learning in deeper layers. ResNet-18 consists of an initial convolutional layer, multiple residual blocks—each containing a 3×3 convolutional layer, batch normalization, ReLU activation, and skip connections—and a final fully connected (FC) layer.

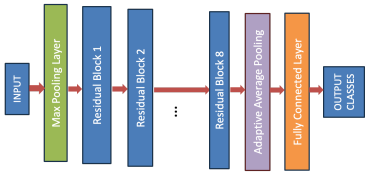

Fig. 7 ResNet-18 Model Architecture

In this project, we utilized a pretrained ResNet-18 model trained on the ImageNet dataset, modifying the final FC layer to output 24 classes instead of the original 1000. ResNet-18 was chosen for its skip connections, which allow outputs from earlier layers to be directly added to deeper layers, improving gradient flow and training stability (Sivaram, 2023). The model was implemented using PyTorch's resnet18 and optimized using Stochastic Gradient Descent (SGD) with Cross-Entropy Loss.

Results and Discussions

Custom Convolutional Neural Network (CNN)

The performance of the custom CNN model was evaluated across 20 training epochs using accuracy, loss curves, confusion matrix and a classification report to assess its effectiveness in traffic sign image classification across 24 categories.

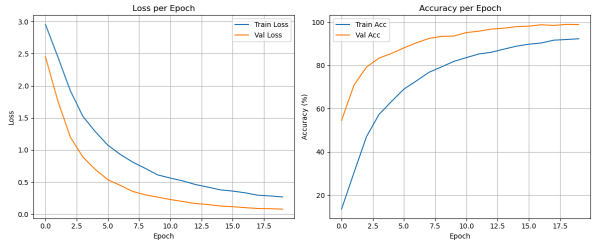

Fig. 8 Training and Validation Loss and Accuracy across 20 epochs

Fig. 8 shows a consistent decline in both training and validation loss, indicating effective learning and generalisation. Notably, validation accuracy exceeded training accuracy across several epochs, likely due to the regularising effect of dropout and improved sampling. By the final epoch, the model achieved approximately 93% training accuracy and 99% validation accuracy.

On the test set, the model achieved a test accuracy of 73.33% and a test loss of 0.8966. While lower than validation performance, this still represents relatively strong generalisation across all 24 classes. The confusion matrix (Fig. 9) and classification report (Fig. 10) provides deeper insights.

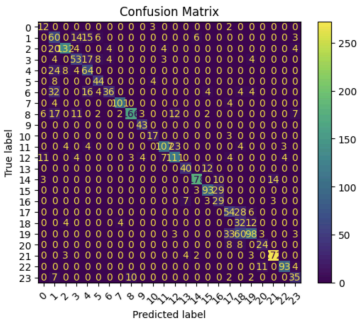

Fig. 9 Confusion Matrix for Test Data

	precision	recall	f1-score	support
0	0.375	0.706	0.490	17
1	0.341	0.577	0.429	104
2	0.874	0.786	0.828	168
3	0.589	0.570	0.579	93
4	0.552	0.640	0.593	100
5	0.688	0.786	0.733	56
6	0.818	0.360	0.500	100
7	0.944	0.927	0.935	109
8	0.927	0.728	0.816	228
9	0.860	1.000	0.925	43
10	0.708	0.850	0.773	20
11	0.884	0.738	0.805	145
12	0.745	0.750	0.747	148
13	0.784	0.769	0.777	52
14	0.942	0.868	0.904	205
15	0.816	0.744	0.778	125
16	0.426	0.690	0.527	42
17	0.491	0.614	0.545	88
18	0.235	0.615	0.340	52
19	0.760	0.490	0.596	200
20	0.632	0.600	0.615	40
21	0.941	0.958	0.949	284
22	0.930	0.861	0.894	108
...				
accuracy			0.733	2583
macro avg	0.704	0.719	0.696	2583
weighted avg	0.779	0.733	0.744	2583

Fig. 10 Classification Report for Custom CNN Model

Best performed classes

Several classes, including class 9, 7, 21 (F1 ≈ 1.00 and 0.9), and class 14, 5, 8, 11, 12, 15, 17, and 20 (F1 > 0.70), demonstrated high precision and recall.

Underperformed classes

Conversely, classes 0, 1, 4, 18, and 23 showed notably lower performance. For example, class 0 achieved high recall (0.706) but low precision (0.375), suggesting it was frequently predicted incorrectly for other classes, potentially due to feature overgeneralization or visual overlap with similar categories.

Overall, the custom CNN effectively learned and generalised across most classes, achieving strong performance on both validation and test data. However, the model's sensitivity to class imbalance and inter-class similarity highlights areas for further improvement through data augmentation, class weighting, or more advanced architectures.

VGG-16

The model was trained for a total of 20 epochs, as the model reached high training accuracies within 20 epochs. We made use of accuracy, loss curves, validation loss, confusion matrix and a classification report to assess the effectiveness of the VGG-16 model over 10 epochs and on the evaluation using test images.

The model shows a steep drop in training loss and a slight decrease in validation loss across epochs. It reaches 100% training accuracy within a few epochs, indicating rapid learning of the training data. However, the validation loss plateaus after epoch 2, suggesting possible overfitting, where the model may have memorized the training data rather than generalizing well to unseen data.

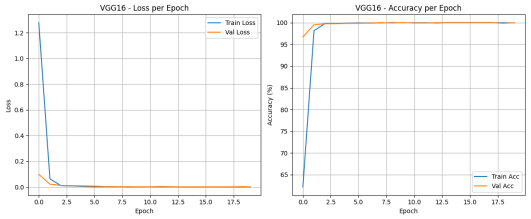

Fig 11 Training and Validation Loss and Accuracy across 10 Epochs

The model achieved a test accuracy of 92.1% and a test loss of 0.24%. The confusion matrix (Fig. 12) depicts the true labels and the predicted labels by the model, which shows a relatively high percentage of labels being predicted accurately.

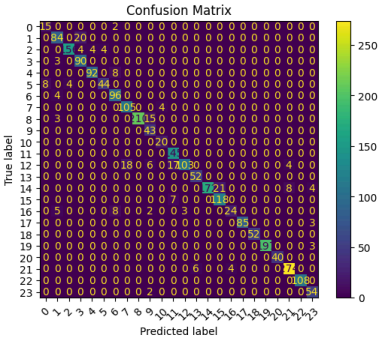

Fig. 12 Confusion matrix for VGG-16

Best performed classes

Several classes demonstrated exceptionally high performance with near-perfect scores. Classes 18, 20, and 22 achieved perfect F1-scores (1.000), reflecting flawless precision and recall. Other high-performing classes included 19 (F1 = 0.992), 17 (F1 = 0.983), 13 (F1 = 0.945), 21 (F1 = 0.961), and 8 (F1 = 0.959). Notably, class 2 also attained strong scores with a precision of 0.975 and an F1-score of 0.951, showcasing consistent predictions.

	precision	recall	f1-score	support
0	0.652	0.882	0.750	17
1	0.848	0.808	0.828	104
2	0.975	0.929	0.951	168
3	0.789	0.968	0.870	93
4	0.958	0.920	0.939	100
5	0.917	0.786	0.846	56
6	0.842	0.960	0.897	100
7	0.854	0.963	0.905	109
8	1.000	0.921	0.959	228
9	0.632	1.000	0.775	43
10	0.833	1.000	0.909	20
11	0.858	1.000	0.924	145
12	0.972	0.696	0.811	148
13	0.897	1.000	0.945	52
14	1.000	0.839	0.912	205
15	0.849	0.944	0.894	125
16	0.857	0.571	0.686	42
17	1.000	0.966	0.983	88
18	1.000	1.000	1.000	52
19	1.000	0.985	0.992	200
20	1.000	1.000	1.000	40
21	0.958	0.965	0.961	284
22	1.000	1.000	1.000	108
23	0.844	0.964	0.900	56
accuracy			0.921	2583
macro avg	0.897	0.919	0.902	2583
weighted avg	0.930	0.921	0.921	2583

Fig. 13 Classification report for VGG-16 model

Underperformed classes

Conversely, class 16 had the lowest performance, with an F1-score of 0.686 due to a significantly lower recall (0.571), suggesting that many true instances of class 16 were misclassified. Class 0 also had a lower F1-score (0.750) caused by reduced precision (0.652), high false positive rate. Class 9 showed perfect recall (1.000) but low precision (0.632), implying it was frequently predicted when it shouldn't have been. This might be due to feature overlap or class imbalance.

The VGG16-based model achieved an overall accuracy of 92.1%, with a macro F1-score of 0.902, highlighting its robust ability to generalize across a diverse set of 23 classes. While the model performs strongly overall, addressing lower-performing classes through data augmentation, class rebalancing, or architectural refinements could further improve classification outcomes and reduce confusion among visually similar categories.

ResNet-18

We trained the model for 20 epochs to prevent underfitting and overfitting, monitoring training and validation loss and accuracy throughout. Training accuracy improved from 14.23% to 99.13%, while validation accuracy increased from 30.02% to 99.20%. The small 0.04% gap between them suggests good generalization with no signs of overfitting.

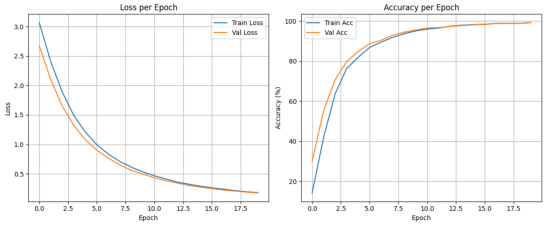

Fig. 14 Loss per Epoch, Accuracy per Epoch for ResNet18.

On the test data, the model achieved an accuracy of 82.19% and a loss of 0.8109. A confusion matrix was used to visualize classification performance, revealing that class 4 (60 km/h) was often misclassified as class 1 (30 km/h) 44 times—indicating difficulty in distinguishing similar speed limit signs.

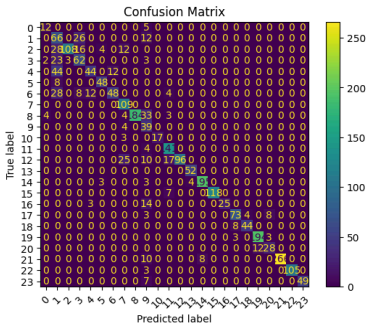

Fig. 15 Confusion matrix for test data.

Best performed class

From the classification report, class 22 was the best performed class, with precision of 1 and recall of 0.972. This is an indication that the model correctly identifies most of actual class 22 and did not misclassify any other classes as class 22. Other classes with high F1 scores include class 10, 13, 14, 19, 21, 23.

Underperformed class

From the classification report, class 9 was the most underperformed class. The model shows a low precision of 0.267, where many of the predictions of the class are incorrect. However, it has a high recall of 0.907, indicating the model correctly identifies most of the actual class 9 signs. The F1-score of 0.413 tells us that the model is fairly good at identifying most of the actual class 9 signs, but still struggles with making correct predictions when it labels something as class 9.

	precision	recall	f1-score	support
0	0.667	0.706	0.686	17
1	0.335	0.635	0.439	104
2	0.973	0.643	0.774	168
3	0.554	0.667	0.605	93
4	0.746	0.440	0.553	100
5	0.873	0.857	0.865	56
6	0.800	0.480	0.600	100
7	0.694	1.000	0.820	109
8	1.000	0.807	0.893	228
9	0.267	0.907	0.413	43
10	1.000	0.850	0.919	20
11	0.820	0.972	0.890	145
12	1.000	0.649	0.787	148
13	0.929	1.000	0.963	52
14	0.961	0.951	0.956	205
15	1.000	0.944	0.971	125
16	1.000	0.595	0.746	42
17	0.869	0.830	0.849	88
18	0.917	0.846	0.880	52
19	0.942	0.970	0.956	200
20	0.718	0.700	0.709	40
21	1.000	0.937	0.967	284
22	1.000	0.972	0.986	108
23	1.000	0.875	0.933	56

Fig. 16 Classification report for ResNet-18 model.

This model outperformed our custom CNN but didn't match the performance of VGG16. However, each model demonstrated their strengths in classifying different classes accurately.

Overall, our 3 models do not outperform human recognition of traffic signs with respective test accuracies respectively, especially when the traffic sign images are not as sharp or in poor lighting conditions for the model. However, they still performed with relative high reliability and consistency.

References

Lim, X. R., Lee, C. P., Lim, K. M., Ong, T. S., Alqahtani, A., & Ali, M. (2023, May 11). Recent advances in traffic sign recognition: Approaches and datasets. Sensors (Basel, Switzerland).https://pmc.ncbi.nlm.nih.gov/articles/PMC10223536/#sec3-sensors-23-04674

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for Large-Scale image recognition. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1409.1556

Sivaram. (2023, August 14). What are skip connections in deep learning?. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/08/all-you-need-to-know-about-skip-connections/

Appendix

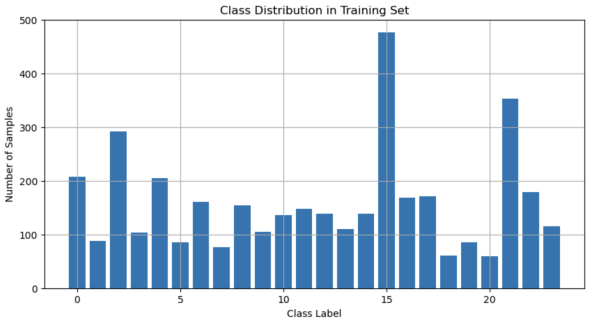

Fig. 17 Class Distribution in Training Set

Fig. 18 Class Distribution in Testing Set

A small sample of 10, depicting the training images which are proprocessed and denormalized for visualisation is provided as below.

Fig. 19 Preprocessed Training Images

A small sample of 10, depicting the test images which are preprocessed and denormalized for visualisation is provided as below.

Fig. 20 Preprocessed Testing Images