UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

Machine Learning and Data Mining 2

Final Project Report
Vietnamese Traffic Sign Classification

Group Members

Nguyen Gia Bach	22BI13052
Phung Tien Dat	22BI13081
Nguyen Dang Nguyen	22BI13340
Nguyen Quoc Khanh	22BI13211
Giap Do Anh Minh	22BI13282
Nguyen Thi Ha Anh	22BI13029

Table of Contents

Abstract

With the rise of artificial intelligence, many applications have been applied to traffic. Automatic traffic signs classification has become an essential component in technologies such as autonomous vehicles or driver assistance, which helps ensure road safety. This project aims to develop and evaluate models for Vietnamese traffic signs classification using Convolutional Neural Networks (CNNs). We use three approaches, including building a simple CNN, building a deep CNN, and using transfer learning through a pre-trained ResNet18 model. All models are evaluated using metrics such as accuracy, precision, recall, and F1-score. Moreover, three models' performance will be compared and analyzed in both quantitative and qualitative results. The results showed that the ResNet18 model achieved the best performance, with an accuracy of 92.51%, and outperformed both CNNs models built from scratch. This project hopes to create a foundation for future applications in Vietnamese traffic, which would help to enhance the safety for people on the road.

Chapter 1

Introduction

1.1 Context & Motivation

Traffic sign classification plays a crucial role in the development of self-driving cars and driver assistance technologies, which aims to ensure safety to traffic participants. This technology is becoming more important with the development of self-driving cars nowadays. It allows the system to understand the surrounding traffic signs and make or suggest safe driving decisions for users accordingly. In Vietnam, the traffic conditions are complex and the traffic sign designs are regulated according to local standards with different designs, color schemes, or category structures. That requires a localized approach to model development compared to using international traffic signs datasets and applying it to Vietnamese traffic signs classification.

In addition, in recent years, the development of deep learning, especially Convolutional Neural Networks (CNNs) has become the popular method for image classification tasks, including traffic sign classification. However, the performance of CNNs depends greatly on their architecture and depth. The deeper models can capture more image details with many layers, thereby performing better, the computational resources to train them would be more expensive. On the other hand, shallower models are faster to train, but the shallow architecture might prevent them from capturing all the features. Pre-trained models, through transfer learning, are another choice for image classification by reusing learned features from large-scale datasets.

In this project, we are motivated to develop and evaluate three different CNN-based models, and identify an efficient and accurate approach to Vietnamese traffic signs classification. With that, we hope that it could serve as a foundation for future applications such as real-time traffic signs detection for autonomous cars or road assistance technology for drivers which would increase the safety of people on the road.

1.2 Objectives

This project has several objectives. One of them is to construct a simple CNN model with a small number of layers from scratch. This serves as a lightweight baseline model to evaluate other models.

The second objective is to design a deeper CNN model from scratch, this is an improved version with more layers compared to the shallow model.

In addition, to further evaluate the performance of our built models, we will apply transfer learning by fine-tuning a pre-trained CNN model ResNet18 on our working dataset.

Finally, we will evaluate the three models' performance using evaluation metrics such as accuracy, precision, recall, and f1-score, and evaluate their qualitative results across different categories of Vietnamese traffic signs. Moreover, we will also compare and analyze the three models' effectiveness in terms of accuracy, training time, and computational costs.

1.3 Expected Outcomes

By the end of the project, we expect to achieve several outcomes related to Vietnamese traffic signs classification. Firstly, we expect to have strong-performance models in Vietnamese traffic signs classification that are based on three CNN approaches: a shallow simple CNN, a deep CNN, and a pre-trained CNN model. More specifically, the shallow model is expected to have a decent benchmark for comparison with other models and have a fast training time. The deep CNN is expected to improve the classification performance of the shallow model, though its training time may increase. The pre-trained ResNet18 model is likely to achieve the highest accuracy among the three models.

In addition, we will also provide a comparison and comprehensive analysis of the models' performance in different aspects.

Chapter 2

Dataset

2.1 Dataset Description

The dataset that we use in this project is the Vietnamese traffic signs dataset from Roboflow [1]. It contains close-up images of different types of Vietnamese traffic signs, including prohibitory signs, warning signs, and mandatory signs. Each sign has a label representing that sign's name. In total, the dataset contains a total of 4,272 images distributed across 121 classes. All of the images are in 640 x 640 size, as shown in Figure 2.1, however, their resolution and background conditions are varied, reflecting real-world conditions. The samples of the dataset that we are using are shown in Table 2.1.

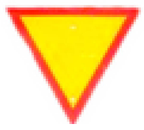
Image	Class Name	Image Size
	103b_CamOtoRePhai	640 x 640
	208_GiaoNhauVoiDuongUuTien	640 x 640
	301a_CacXeChiDuocDiThang	640 x 640
	306_TocDoToiThieuChoPhep	640 x 640

Table 2.1: Examples from the Vietnamese traffic signs dataset that we are using.

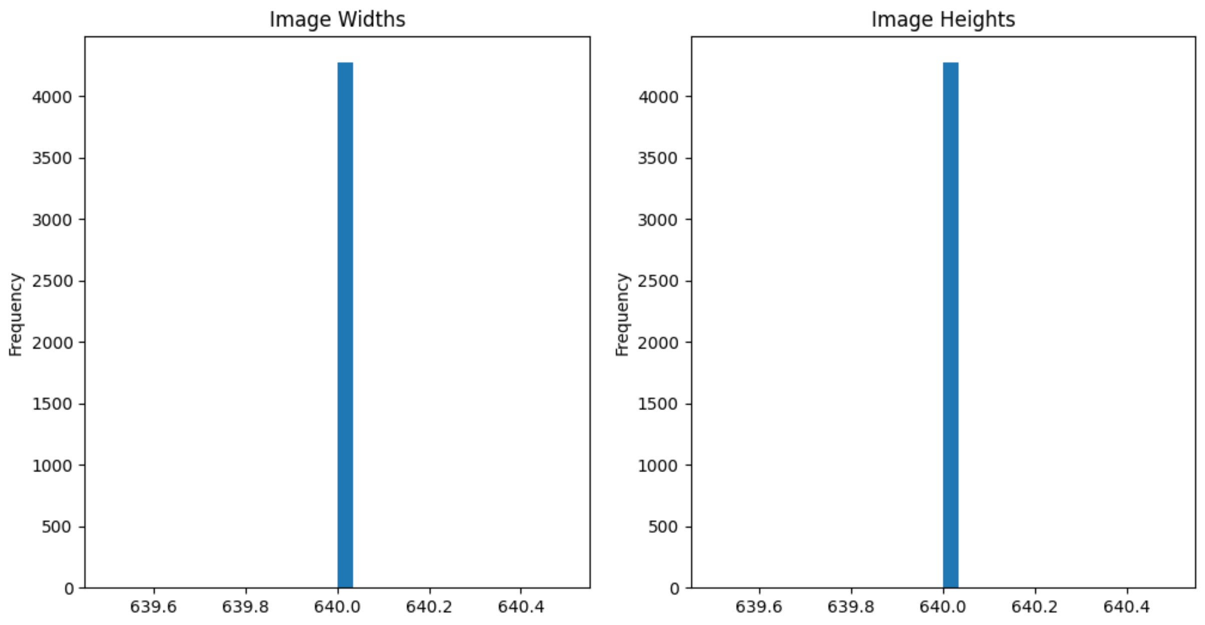


Figure 2.1: Image width and image height for each image in the dataset.

2.2 Data Preprocessing

The dataset is structured into folders of class names; each folder contains the images belonging to that class. Before going into the models, the dataset needs to have some preprocessing steps. Firstly, we check for the number of images per class to verify if the dataset is balanced. The result shows equal distributions between classes, as shown in Figure 2.2. Most classes have a similar number of samples per class, with the mean samples per class being 35.31, the minimum samples in a class being 18, and the maximum samples in a class being 118. Furthermore, we also checked for any empty classes in the dataset. The results in Figure 2.2 confirmed that no empty class. We also checked for invalid images by opening every image file, and found no invalid images in the dataset. Then, the dataset was divided into train/validation/test sets with a ratio of 80/10/10.

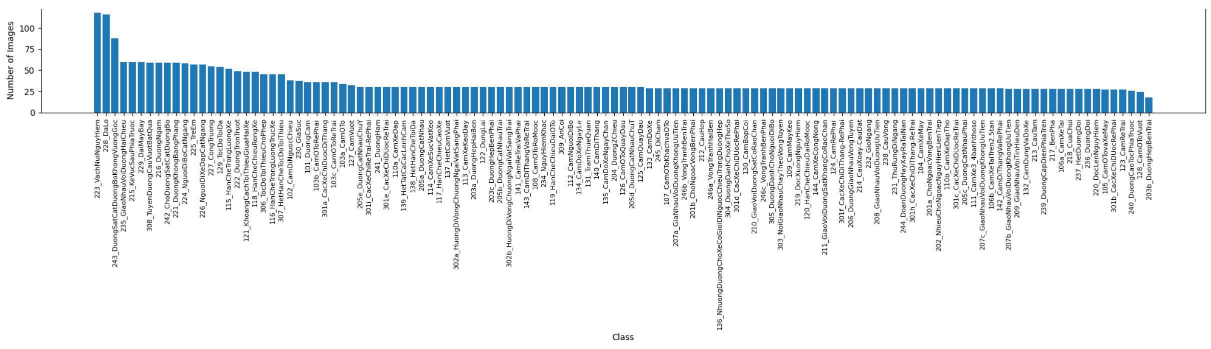


Figure 2.2: The data distribution across all classes in the dataset.

After that, data augmentation is applied to the training set images, which includes resizing to 64 x 64 pixels, random rotation, random horizontal flip, color jittering, random affine, normalization with ImageNet's mean and standard deviation, and finally converting to tensors. For the validation and test set images, we only apply resizing to 64 x 64 pixels, normalization, and converting to tensors.

For the labels, we numerically encoded the labels to convert them from class names to numbers ranging from 0 to 120, corresponding to 121 classes. Table 2.2 shows some images after augmentation, their respective class names and class IDs, and image size.

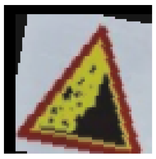
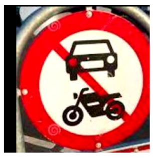
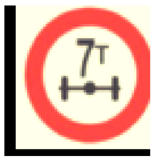
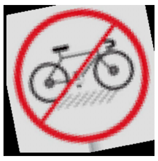
Image	Class Name	Class ID	Image Size
	228_DaLo	84	64 x 64
	105_CamOtoVaXeMay	6	64 x 64
	116_HanCheTrongLuongTrucXe	19	64 x 64
	110a_CamXeDap	12	64 x 64

Table 2.2: Examples from the Vietnamese traffic signs dataset after preprocessing.

Chapter 3

Methodology

3.1 Simple CNN model

The first model developed in this project is a simple CNN designed from scratch. The architecture is relatively shallow with two convolution blocks followed by a fully connected block, as shown in Figure 3.1.

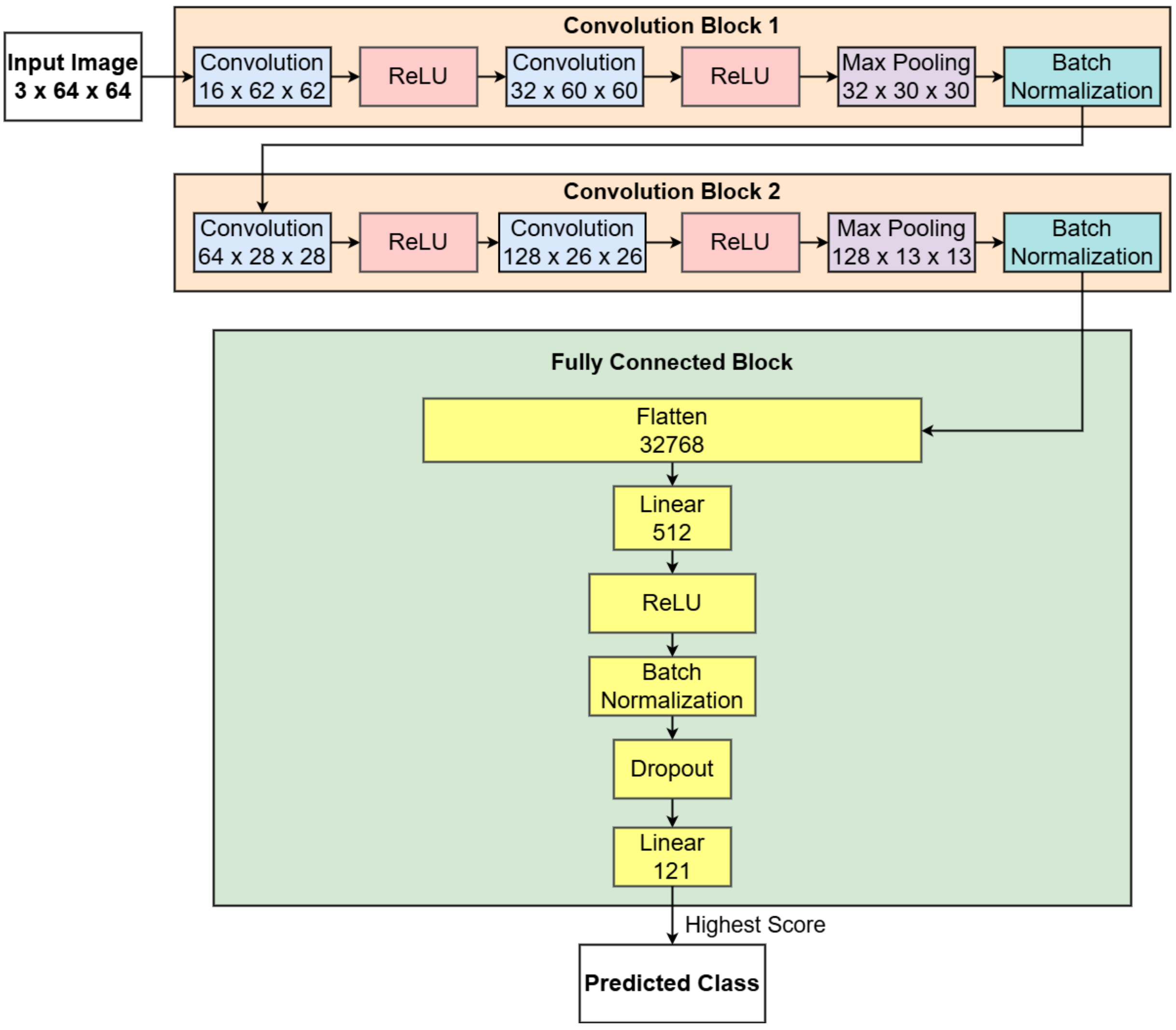


Figure 3.1: Our simple CNN architecture with 2 convolution blocks and a fully connected block.

The input image will go to the first convolution layer of the first convolution block. This layer uses 16 filters with a kernel size of 3 x 3, followed by a ReLU activation function [2]. Then, the second convolutional layer increases the number of filters to 32 and also has a ReLU activation function after it. After that, a Max Pooling with a kernel size of 2 x 2 is followed to reduce the spatial dimensions of the feature maps. Then a batch normalization layer [3] is added to stabilize and accelerate the training.

The second convolution block repeats a similar pattern with two convolutional layers with increasing depths of 64 and 128 filters respectively, each followed by ReLU activation function. This block also includes a max-pooling layer and batch normalization. After the convolution blocks, the feature maps are flattened and passed through a fully connected layer with 512 units, followed by ReLU activation, batch normalization, and a dropout layer with a rate of 0.5 to reduce overfitting. Finally the output layer, which is a fully connected linear layer outputs 121 logits, corresponding to the number of traffic sign classes in the dataset. The highest logit index is selected as the predicted class for the input image.

3.2 Deeper CNN model

To enhance the classification performance and enable the model to learn more complex visual features, we designed a deeper CNN model that improved compared to the previous shallow model. This deep CNN model includes more layers and more filters, which help it extract more visual features, as shown in Figure 3.2.

The architecture consists of four convolution blocks, each containing two convolutional layers with ReLU layers, and followed by a max pooling layer and batch normalization. With the deeper architecture with more convolution blocks, the number of filters increases progressively through each block, from 16 and 32 in the first block to 1024 and 2048 in the final block. This effectively allows this model to capture deeper features of images compared to a maximum of 128 filters in the shallow model.

Another improvement compared to the shallow model is the use of padding. All convolutional layers in the deep CNN use padding to preserve the spatial dimensions before pooling, which allows to retain more spatial information across layers. As a result, the model maintains richer feature maps across the network.

In the fully connected block, the deep CNN model flattens the final feature maps and passes them through a fully connected layer with 512 neurons, followed by a ReLU layer, a batch normalization layer, and a dropout with a rate of 0.28. Then, it outputs the predictions through a linear layer with 121 neurons, which matches the traffic sign classes. Finally, the highest logit index is selected as the predicted class. While both the simple CNN and deep CNN use similar components in the final layers, the large model has a much larger input to the linear layer with 2048 x 4 x 4 due to its larger learned features, compared to 128 x 13 x 13 of the shallow model.

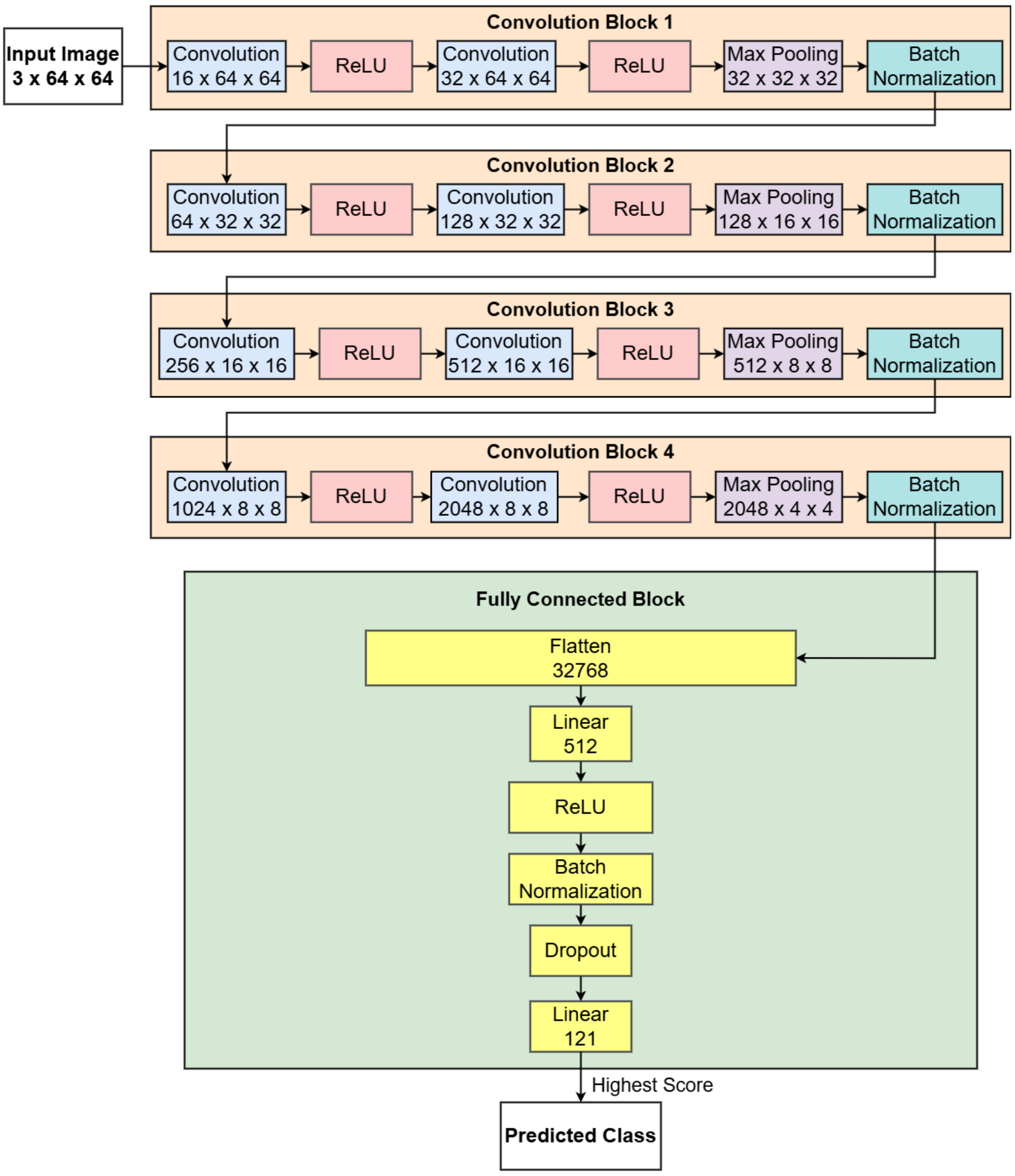


Figure 3.2: Our deeper CNN model has 4 convolution blocks and a fully connected block, which helps to extract more image features.

3.3 Pre-trained ResNet18

To improve classification performance to compare with our built models' performance, we used ResNet18, a deep convolutional neural network with residual connections [4]. ResNet18 is a variant of the ResNet architecture, designed to solve the vanishing gradient problem by using skip connections, which allow gradients to flow more easily through the network [4]. This architecture is significantly deeper than both the simple and deep CNN models described earlier, which offers better performance on complex datasets, as shown in Figure 3.3.

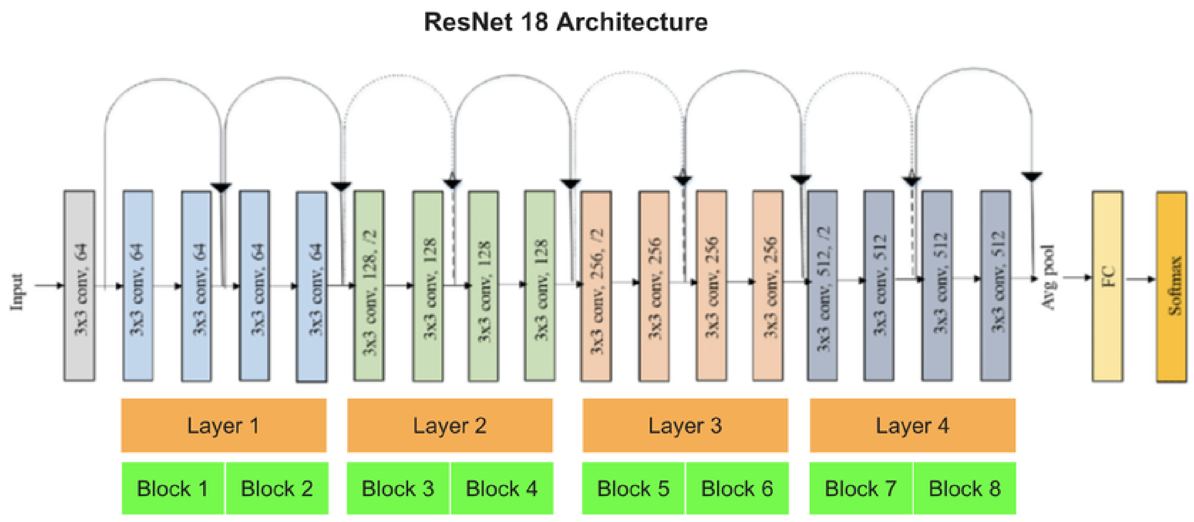


Figure 3.3: The ResNet18 Architecture

The ResNet18 Architecture consists of 18 layers, which are divided into several blocks, each containing convolutional layers, batch normalization, and ReLU activation functions. The core feature of ResNet18 is its use of skip connections, which allow the network to learn the differences between the input and the output of each block.

Regarding Figure 3.3, the architecture begins with an initial convolution layer with 64 filters of size 3 x 3. This is followed by Layer 1, which consists of two blocks, each of which contains convolution layers with 64 filters. After Layer 1, the network will move to Layer 2, which contains two blocks with 128 filters. Similarly, the network continues the progress with Layer 3 and Layer 4, each increasing the filter sizes to 256 and 512, respectively. Each residual block within the layers includes two convolutional layers, followed by batch normalization and ReLU activations. The skip connection in each block ensures that the input is added directly to the output of the convolutional operations. This helps prevent the loss of information during the forward pass, which maintains rich feature maps and enables more efficient training.

After processing through the convolutional layers and residual blocks, the output is passed through a global average pooling layer, which reduces the feature maps' dimensions. Finally, a fully connected (FC) layer is used, followed by a softmax activation that produces the final classification.

Compared to these previous two models, ResNet18 has a significant advantage with the use of pre-trained weights. Those weights have been trained on large-scale datasets like ImageNet, so this pre-training helps the model generalize better and recognize complex patterns with fewer training iterations. Therefore, ResNet18 is particularly suitable for fine-tuning, expecting to improve the classification accuracy on this traffic signs classification dataset.

3.4 Training Process

The hyperparameters used during the training of three models and their training time are summarized in Table 3.1.

Table 3.1: Hyperparameters and the training time of each model: Simple CNN, Deep CNN, ResNet18.

	Simple CNN	**Deep CNN**	**ResNet18**
Optimizer	AdamW	AdamW	AdamW
Learning Rate	1e-3	1e-4	1e-4
Weight Decay	x	0.01	0.01
Loss Function	Cross Entropy	Cross Entropy	Cross Entropy
Scheduler	StepLR	ReduceLROnPlateau	ReduceLROnPlateau
Batch Size	32	32	32
Epochs	20	20	20
Training time	5 minutes 13 seconds	8 minutes 20 seconds	10 minutes 33 seconds

The training process of three models used the PyTorch framework. All three models used the Cross-Entropy loss function [5] and AdamW optimizer [6] to update the parameters. However, the learning rate and weight decay vary for each model. Due to the simple architecture, the CNN model had a learning rate of 1e-3, whereas larger models had a learning rate of 1e-4. The simple CNN model did not have weight decay, but larger models used the weight decay of 0.01 to avoid overfitting.

In addition, different learning rate schedulers were used. The simple CNN used StepLR, which reduced the learning rate at fixed intervals, while Deep CNN and ResNet18 used ReduceLROnPlateau, which adjusts the learning rate dynamically when validation performance plateaus.

Each model was trained for 20 epochs with a batch size of 32 using Google Colab T4-16GB GPU. The training time varies according to the model architecture. The simple CNN completed the training in around 5 minutes, while the deep CNN required more than 8 minutes. The pre-trained ResNet18, which has a larger and deeper architecture, took 10 minutes and a half to train.

3.5 Model Evaluation

To evaluate the performance of our models, we used four metrics: **accuracy**, **precision**, **recall**, and **F1-score**. These metrics help us to understand how well the models can classify Vietnamese traffic signs.

1. **Accuracy**
   Accuracy is a metric that evaluates the overall correctness of the model's predictions.

It is calculated by the sum of True Positives (TP) and True Negatives (TN), divided by the total number of samples. TP are the traffic signs that the model correctly predicts as belonging to a specific class. TN are the traffic signs correctly predicted as not belonging to that class. False Positives (FP) occur when the model incorrectly predicts a traffic sign as belonging to a class it does not belong to. False Negatives (FN) happen when the model fails to recognize a traffic sign as belonging to its correct class.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{True Negatives} + \text{False Positives} + \text{False Negatives}}$$

Overall, higher accuracy means the model is good at making correct predictions.

2. **Precision**
   Precision shows how many of the model's predictions are correct. The precision formula is the number of true positive predictions divided by the total number of positive predictions.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

3. **Recall**
   Recall measures how well the model identifies actual traffic signs from the dataset. To calculate that, the number of true positives is divided by the sum of true positives and false negatives.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

4. **F1-score**
   F1-score combines precision and recall to show the balance between them. It is calculated by dividing the product of precision and recall by the sum of precision and recall. Then it is multiplied by 2.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Chapter 4

Results and Discussion

4.1 Experimental Results

4.1.1 Quantitative Results

The three CNN models, Simple CNN, Deep CNN, and Resnet18, were evaluated on the test set using metrics discussed in Section 3.5. The results of the models are shown in Table 4.1.

Table 4.1: Quantitative results of our three CNN approaches.

	Accuracy	**Precision**	**Recall**	**F1-score**
Simple CNN	90.60	89.67	90.60	89.40
Deep CNN	91.55	90.33	91.55	90.63
ResNet18	**92.51**	**90.72**	**92.51**	**90.91**

Based on the results, the simple CNN model achieved an accuracy of 90.6%, a precision of 89.67%, a recall of 90.60%, and an F1-score of 89.40%. For a simple, shallow architecture such as the simple CNN model, these results are very decent and strong. Due to the large number of labels, we could not draw the confusion matrix; instead, we summarized the top five classes misclassified by each model, as shown in Tabel 4.2. The model has some problems in classifying class 48, class 49, class 103, class 110, and class 4, with each class having 3 to 4 misclassifications. This might suggest that these classes might have visual similarities, which makes it harder for the model to distinguish.

Table 4.2: Top five classes misclassified by three models (in descending order from left to right).

	Top 5 Classes misclassified (misclassifications)
Simple CNN	Class 48 (4) - Class 49 (4) - Class 103 (4) - Class 110 (4) - Class 4 (3)
Deep CNN	Class 45 (4) - Class 62 (4) - Class 105 (4) - Class 109 (4) - Class 4 (3)
ResNet18	Class 4 (4) - Class 107 (4) - Class 110 (4) - Class 26 (3) - Class 52 (3)

On the other hand, the deep CNN model, which has a deeper network, achieved increased results with an accuracy of 91.55%, precision of 90.33%, recall of 91.55%, and F1-score of 90.63%. In terms of class misclassification, the model struggled to classify classes 45, 62, 105, 109, and 4. Moreover, class 4 appeared in both the Simple CNN and Deep CNN's top 5 misclassified classes, indicating that this class might have specific characteristics that make it difficult for the models to classify accurately.

In addition, the Resnet18, which leveraged a pre-trained architecture, achieved higher results with an accuracy of 92.51%, precision of 90.72%, recall of 92.51%, and an F1-score of 90.91%. The top 5 misclassified classes for ResNet18 were class 4, class 107, class 110, class 26, and class 52. Notably, class 4 appeared in all three models' top 5 misclassified classes, which might indicate that this class is relatively challenging for the three models to classify.

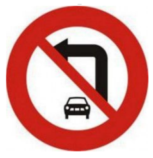
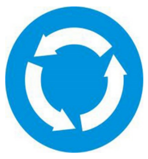
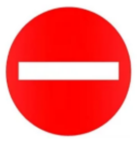
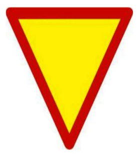
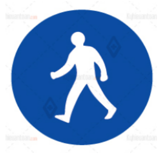
Overall, it is clear that the ResNet18 outperformed both the simple CNN and Deep CNN across all metrics, due to its learned features on larger datasets. However, the simple CNN model achieved comparable results, given its relatively shallow architecture, and faster training time compared to larger models. On the other hand, the deep CNN showed an improvement over the simple CNN, which confirms that adding depth to the network can enhance the model's ability to correctly classify more traffic signs.

Moreover, it is worth noting that a simple CNN model has the fastest training time, which makes it ideal in cases where low computational resources are prioritized. In contrast, when the model's accuracy is preferred, deeper models like deep CNN and ResNet18 would be a better choice, though they would require more training time.

4.1.2 Qualitative Results

In addition to the quantitative results, we also assessed the models' performance on unseen images. These images were manually labeled with ground truth values to compare with the predictions of the three models, as shown in Table 4.3.

Table 4.3: Qualitative results of three models: (1): Simple CNN, (2): Deep CNN, and (3): ResNet18, on random unseen images. Red texts represent the incorrect predictions, while green texts represent the correct ones.

Image	Predicted Class Name	Ground Truth
	(1): 103b_CamOToRePhai (2): 110a_CamXeDap (3): 103c_CamOToReTrai	103c_CamOtoReTrai
	(1): 110a_CamXeDap (2): 301d_CacXeChiDUocRePhai (3): 303_NoiGiaoNhau ChayTheoVongTuyen	303_NoiGiaoNhau ChayTheoVongTuyen
	(1): 221_DuongKhongBangPhang (2): 102_CamDiNguocChieu (3): 102_CamDiNguocChieu	102_CamDiNguocChieu
	(1): 208_GiaoNhauVoiDuongUuTien (2): 208_GiaoNhauVoiDuongUuTien (3): 208_GiaoNhauVoiDuongUuTien	208_GiaoNhauVoiDuongUuTien
	(1): 305_DuongDanhChoNguoiDiBo (2): 305_DuongDanhChoNguoiDiBo (3): 305_DuongDanhChoNguoiDiBo	305_DuongDanhChoNguoiDiBo

In general, the ResNet18 model, with its deep architecture, outperformed the remaining models with all five correct predictions. The deep CNN showed poorer performance compared to ResNet18, with 2 wrong predictions out of five images. The simple CNN only predicted 2 correct images. These results once again confirmed the importance of model depth in capturing image features and reducing misclassification. Overall, the ResNet18 model, due to its deep architecture, showed the best performance in terms of both quantitative metrics and qualitative results. It was able to correctly predict more traffic signs that were difficult for the other models.

4.2 Models Limitations

Despite achieving good metrics, the models still have some limitations that need to be addressed. Firstly, the dataset has limited labels and does not cover the full range of Vietnamese traffic signs. This limits the model's ability to generalize and accurately classify diverse signs that may appear in real-world scenarios.

Secondly, the model might struggle with signs that have similar designs. A clear example is Class 4, which consistently appeared among the top misclassified classes across all three models. This suggests a limitation in the models' inability to capture fine-grained visual differences between similar-looking traffic signs.

Additionally, the models are only designed to classify traffic signs from close-up, cropped images. Therefore, their performance drops significantly when classifying some real-world scenario images, such as traffic signs in the street or road images. To handle such scenarios effectively, the classification models would need to be integrated with an object detection model to correctly localize the traffic signs in the images and classify them.

Chapter 5

Conclusion

5.1 Conclusion

In this project, we created three different CNN models to classify Vietnamese traffic signs: a simple CNN, a deep CNN, and a pre-trained ResNet18. The ResNet18 model gave the best performance with the highest accuracy, precision, recall, and F1-score. The deep CNN model also showed good results, which were better than the simple CNN in all metrics. Although the architecture of the simple CNN model is simpler, it produced comparable results and was much faster to train.

However, the models still face some challenges, such as limited support for a wide range of traffic signs, and misclassification of some traffic signs of similar design. Furthermore, the models were only trained on close-up images, and their performance could be degraded in real-world traffic sign images that are not cropped.

Overall, while deeper models like ResNet18 provided significantly higher performance, lighter models like simple CNN can still provide useful results. Furthermore, this project could serve as a foundation for future improvements in traffic sign classification systems, which could be applied to real-world scenarios like self-driving cars or driver assistance.

5.2 Future Work

We plan to continue improving the model to address these challenges and enhance the traffic sign classification performance. Firstly, we would like to expand the dataset to cover more Vietnamese traffic signs, which helps the model recognize a wider range of signs. We will collect and annotate more traffic signs in Vietnam and add to the dataset.

In addition, we would continue to develop or combine deep models, which might help the model learn fine-grained details and correctly classify similar traffic signs.

Finally, we plan to integrate the model in an object detection model that can be applied to real-world scenarios where the traffic signs in images are not cropped. This model would allow it to correctly localize the traffic signs in images and classify them effectively.

References

[1] Truong, “bien-bao-giao-thong-viet-nam zalo dataset,” feb 2023. [Online]. Available: https://universe.roboflow.com/truong-a6rzc/bien-bao-giao-thong-viet-nam-zalo

[2] A. F. Agarap, “Deep learning using rectified linear units (relu),” *arXiv preprint arXiv:1803.08375*, 2018.

[3] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International conference on machine learning*. pmlr, 2015, pp. 448–456.

[4] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] Z. Zhang and M. Sabuncu, “Generalized cross entropy loss for training deep neural networks with noisy labels,” *Advances in neural information processing systems*, vol. 31, 2018.

[6] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” *arXiv preprint arXiv:1711.05101*, 2017.