Exploring Deep Networks for Image Classification: A Comparative Study of VGG, ResNet, and a Custom Architecture

Manuela Valencia Toro
ec24835@qmul.ac.uk
Student ID: 240427850

Abstract

As the field of computer vision is rapidly evolving, image classification remains one of the most important tasks in deep learning, and architectures like convolutional neural networks (CNNs) have revolutionized the way this task is approached. This report focuses on two of the best known CNN architectures: VGG and ResNet. A comparative analysis of these models is performed, evaluating their performance on image classification tasks using, particularly, the MNIST dataset. The models are evaluated based on accuracy, training dynamics, and their generalization capabilities. There is also an exploration of possible improvements to these models by modifying their training strategies, their architecture, and data modifications. Finally, adaptations to both models are implemented to specifically enhance execution time. This study aims to provide a deeper understanding of the strengths and limitations of VGG and ResNet to ultimately contribute to more efficient and effective strategies for image classification.

1. Introduction

There are a wide range of tasks performed in the field of computer vision, including object detection, which is a technique for locating instances of objects in images or videos; image segmentation, that in general terms is subdividing the image into different objects; image generation, which involves synthesizing images from textual prompts and also transforming existing images. And finally, image classification, that serves as a foundational task that supports many of the more complex applications.

Image classification is about assigning a label or class to an entire image based on its visual content. This can be achieved by two different methods, namely supervised learning, which uses human-generated datasets with annotations, or transfer learning, which uses pre-trained models as feature extractors. There are different types of datasets that are commonly used to train image classification models. Two of the most commonly used are MNIST [4]and CIFAR-10[5]. MNIST consists of grayscale images of handwritten digits ranging from 0 to 9, each with the corresponding label. In contrast, CIFAR-10 contains color images divided into 10 different categories, including animals and objects such as dogs, cats and birds. These datasets provide guidance for evaluating the performance of multiple classification models.

There are several methods to evaluate the performance of image classification models. One of the most common metrics is accuracy, which measures the proportion of correctly classified images out of the total number of samples. It gives an indication of how well the model performs overall. However, accuracy alone may not be sufficient, especially when multiple classes are involved, as the dataset may be unbalanced. In such cases, additional metrics such as precision, recall and the F1 scores are used to provide a more comprehensive evaluation per class. These performance metrics are specially used during the validation and testing phases, as they help evaluate how well the model handles class-specific performance and whether it generalizes effectively across all categories using new data.

This project aims to critically evaluate and compare the performance of VGG16 [1] and ResNet34 [2] for image classification using Deep CNNS. Both models have been important in the development of computer vision, but they use different design approaches. VGG has 16 layers with small 3×3 filters placed one after another, making it simple and easy to understand and implement. ResNet, on the other hand, uses shortcut connections that allow the network to skip some layers. These shortcuts help the model train better, avoiding losing data and key features, especially when the network is very deep. This project first follows the exact design from the original papers [3, 8], and finally performs some adjustments and experiments to improve not only the overall performance but also the execution time.

2. Critical Analysis

In this section, a detailed comparison between the VGG and ResNet architectures is performed by analyzing their design principles, their performance on the MNIST dataset, and possible areas of improvement. As for the hyperparameters, both models were trained and validated over 10 epochs with a mini-batch size of 128. The optimization was performed using the Adam optimizer with a learning rate of 0.001. In addition, the loss function used was cross-entropy, as it is well suited for multi-class classification tasks and finally, the training was performed on a graphics processing unit (GPU) to speed up the process.

Both classifiers were implemented from scratch using pytorch to build, train and test each model. Pytorch provides a flexible and efficient environment for defining deep learning models, managing datasets and tracking performance during training and evaluation. The models were created by inheriting from nn.Module [6], the base class for all neural network modules. The data was extracted from the torchvision library [7] and preprocessed using transforms to change the size from 28x28 to 224x224 and to normalize the data. At this point, it doesn't make much sense to increase the size of the input data that much, but the idea was to implement the model as it was defined the first time.

2.1. VGG architecture:

VGG (Visual Geometry Group) is a well-known convolutional neural network that became popular due to its simplicity and straightforward design. The architecture consists of multiple convolutional layers with small 3×3 filters and pooling layers [5]. This design, combined with a very understandable implementation, allows the network to learn highly complex features. However, VGG networks require a very large number of parameters, which makes them computationally expensive and prone to overfitting on smaller datasets.

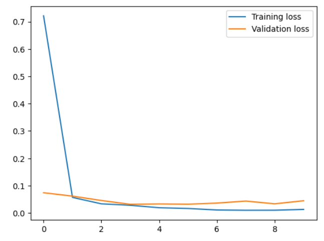

Figure 1. VGG Training and Validation Loss

The figure 1 shows how the VGG16 loss changed during training and validation. In the first few epochs, there is a sharp decrease in training loss, indicating that the model quickly learned useful features from the data. In the next epochs the loss values begin to stabilize, showing slower improvements and smaller fluctuations in validation, which is a common pattern as the model converges. The similarity between training and validation loss, along with their consistent downward trend, suggests that the model is not overfitting and is generalizing effectively.

After the 10 epochs, the model achieved a validation accuracy of 0.988 and an F1 score of 0.987 with an upward trend over the course of the epochs, as shown in figure 2. These results indicate that the model was correclty implemented and performs very well on unseen data. The high F1 score also suggests a good balance between precision and recall across all classes, confirming that the model not only predicts accurately but also handles class distribution effectively. This supports the conclusion that the model generalizes well and is suitable for image classification tasks on this dataset.

Regarding the testing stage, the model had a really good performance with an accuracy of 0.986 and an F1 score of 0.985. These results are consistent with the validation metrics, confirming the model's ability to generalize to completely unseen data. The small difference between validation and test results shows that the model works well not only on known data but also on new data, also confirming that it is not overfitting. This means it learned useful patterns and is not just memorizing.

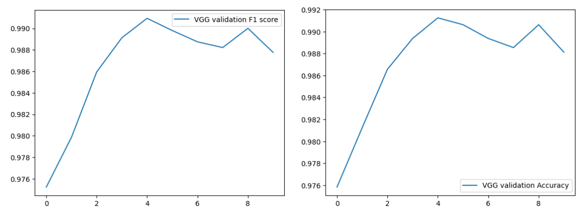

Figure 2. VGG Validation Accuracy and F1 Score

Finally, the confusion matrix, as shown in 3, has a clear pattern where all the diagonal elements are highlighted in blue. This indicates that the model is correctly classifying most of the instances, with the true positives corresponding to each class being the dominant values. The rest of the matrix is white, which suggests that the number of misclassifications is relatively low. This pattern implies that the model performs well in distinguishing between the different classes, with few errors in its predictions.

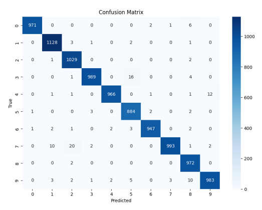

Figure 3. VGG MNIST Test Confusion Matrix

Although VGG achieved good accuracy, it took approximately 57 minutes (3460 seconds) to complete 10 epochs, using 375 batches for training and 95 for validation. This training time is notably longer than more modern architectures, especially considering the simplicity of the dataset. Additionally, the training process becomes slower as the network depth increases, which is one of the known limitations of the VGG architecture. Based on this, even though this is an accurate design, it is not a recommendable architecture to classify the MNIST dataset. There are multiple improvements to do in order to decrease the execution time and to adapt the VGG design to the input data used, including the reduction of the number of layers.

2.2. ResNet Architecture:

One of the main challenges when training deep neural networks is the vanishing gradient problem. As the network gets deeper, the gradients used to update the weights during backpropagation can become very small, making it difficult for the model to learn, leading to worse performance, even when adding more layers. ResNet (Residual Network), as mentioned before, solves this problem by introducing residual connections that skip one or more layers and directly add the input of a block to its output.

In this application, ResNet showed superior performance compared to VGG, especially in terms of training speed and convergence. The model's ability to skip layers means that it can learn more efficiently and generally performs better. In this case, it took about 46 minutes (2815 seconds) to complete the training iterations for 10 epochs, using the same 375 batches for training and 95 for validation, which is significantly less time.

In 4, the left image shows that the model was probably overfitted in the first epochs, as the validation loss is significantly higher than the training loss. After the third epoch, both losses begin to stabilize and gradually approach each other until they reach a minimum of 0.018 and 0.02 respectively. On the right side of the figure, the accuracy and F1 score display an upward trend across the epochs in the validation set, with both metrics showing almost identical values throughout all epochs. This consistent improvement in both accuracy and F1 score indicates that the model is learning effectively over time.

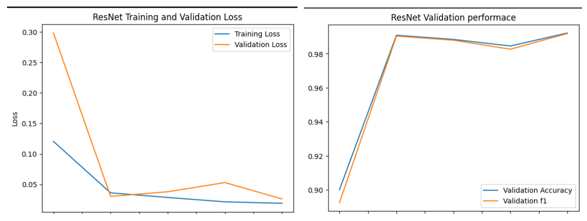

Figure 4. Left: ResNet Training and Validation Loss. Right: Validation Accuracy and F1 Score

In terms of its performance on the test set, the accuracy and F1 score were 0.992 and 0.991, respectively. Combining this with the performance during training described before, it can be concluded that ResNet, like the VGG model, generalizes well to new and unseen data, suggesting that the model has successfully learned the underlying patterns in the dataset without overfitting. Compared to VGG, that performed well but required more training time, the ResNet model provides faster convergence with consistently high performance on both the training and test datasets.Residual connections in ResNet enable a more efficient training process by facilitating the flow of gradients through deeper layers. This not only accelerates training but also improves accuracy. Finally, the figure 5 shows the confusion Matrix for the test set, confirming its performance is similar to that of VGG, as both models exhibit high accuracy in classifying the digits and minor misclassifications.

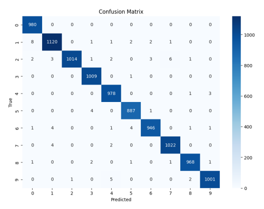

Figure 5. ResNet MNIST Test Confusion Matrix

Although both models demonstrated good overall performance, ResNet showed slightly better results, not only in terms of accuracy and F1 score but also regarding execution time. This suggests that ResNet is more efficient and better suited, offering both improved performance and faster training compared to VGG. However, the complexity introduced by residual connections makes ResNet more challenging to implement, and for a relatively simple dataset like MNIST, the added complexity may not be justified. Despite the better execution time compared to VGG, it is still not significantly faster as it should be.

3. Improvements and Modifications:

Since the performance of VGG and ResNet was generally good, my main focus was to improve the execution time for both and make them fit the specifications of the MNIST dataset by avoiding additional data, which not only creates delays but also noise, by default. First, I updated the transformation applied to the dataset to obtain normalized 28x28 images in grayscale, and to be able to feed the models with this new data I had to make some changes.

3.1. Modified VGG:

The figure 6 shows the final structure of the modified VGG model reflecting the adjustments made to adapt it to the MNIST dataset. It visually outlines the reduction in convolutional blocks, the updated number of input and output channels, and the downsized fully connected layer. The first convolution layer accepts 1 input channel instead of 3 so that it can work with grayscale input. I also reduced the number of output channels from 64 to 32, reduced the depth of the model and the computational cost because as it was shown before, there are extra complexity that is not longer needed. With this changes I ensure the basic structure is preserved while the execution time and complexity are reduced without affecting the performance. Since the input data has a smaller size, I also reduced the first fully connected layer from 512 to 256 making the network even lighter.

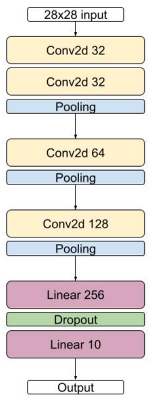

Figure 6. VGG Modified

In terms of performance, the execution time was significantly reduced from 57 minutes to just 3 minutes (199 seconds), representing a reduction of approximately 96.5%, while still maintaining high accuracy and F1 score in both the training and testing phases. Figure 7 illustrates the similar behavior between the original VGG16 and the proposed modified VGG model. Regarding the loss function, although the modified model began with a higher value and took longer to stabilize, both eventually converged to similar minimum values. On the right, it is possible to observe that the accuracy is comparable between both models. VGG16 remains consistently stable, oscillating between 0.98 and 0.99, while the modified VGG starts at 0.94 and steadily increases until reaching 0.989. Although the final accuracy is almost the same than that of the original VGG16 during validation, on the test set for the modified model it is even higher (the green dot), confirming that this adaptation is well-suited for the MNIST dataset. However, for larger and more complex input images this simplification will impact the final performance.

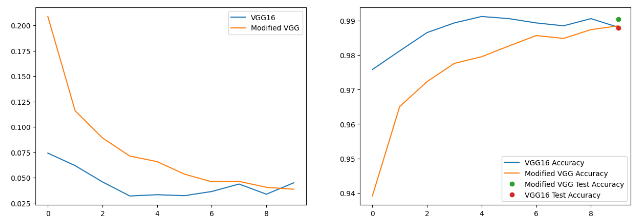

Figure 7. VGG16 vs VGG modified validation Loss and Accuracy

I also used CIFAR-10 to determine if the changes made are only effective for simpler datasets such as MNIST. CIFAR exhibits greater diversity and complexity and often requires deeper networks such as VGG16 to effectively capture the patterns in the data. Figure 8 shows the performance of the model on both datasets in the test stage and illustrates the results. It was necessary to adapt the CIFAR dataset to feed the model and to update the first layer of the model. The first step was to downsize the data to 28x28, normalize it and convert it to tensors in order to use the GPU. The input channels of the first layer were changed from 1 to 3 to allow the model to receive RGB images. I decided to do this because otherwise important data would be lost.

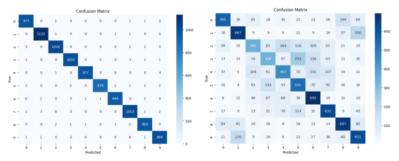

Figure 8. Test Confusion Matrix - Modified VGG with MNIST(left) vs CIFAR10(right)

The confusion matrices for the modified VGG model trained on MNIST and CIFAR datasets show significant differences in classification performance. When applied to MNIST (left), the matrix is almost completely concentrated along the diagonal, indicating near perfect classification with minimal wrong classifications between classes. However, when the same model is trained on CIFAR (right), the matrix exhibits greater variance, with prominent off-diagonal elements indicating increased misclassification and lower class-specific accuracy as the model struggles more with distinguishing between similar classes due to the higher variability and color information. This contrast shows the ability of the model to work with simpler datasets such as MNIST, while struggling with the increased complexity and diversity in CIFAR. In this case, the final accuracy for the test set was 0.55 while the F1 score was 0.547262, compared to MNIST, the performance was reduced by around 40%. In cases like this, one possible strategy to improve performance is to apply data augmentation. To implement it, classical methods like rotation, scaling and cropping can be used, as well as more advanced techniques such as elastic deformations, which can help the model to generalize better by introducing variations into the training data.

3.2. Modified ResNet:

ResNet model was also modified to align it with the MNIST dataset. The most notable changes are reducing the number of residual blocks per layer to only 1 and updating the initial convolution to use smaller kernel (from 7x7 to 3x3) and stride. I also removed the initial MaxPool2d, because since the input image is smaller, the spatial dimensions no longer need to be reduced, transforming the initial ResNet into a lighter and faster architecture. The figure 9 shows these changes in a simplified way.

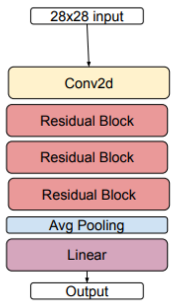

Figure 9. Modified ResNet

This new version of the ResNet architecture showed notable improvements in terms of execution time. It took only around 4 minutes (235 seconds) to complete all the training and validation epochs, 93% faster than the original implementation. Compared with the modified version of VGG it took a little bit more and one possible reason for this is the model complexity, which, in the case of very deep networks, is very useful but in this specific case, it does not provide significant benefits compared to the simpler structure of VGG, which allows it to train faster despite having fewer optimizations.

In addition to the improvement in execution time, the figure 10 shows that performance was not affected. This suggests that the optimizations made to the ResNet architecture, which resulted in faster training times, did not affect the model's ability to learn and generalize effectively, demonstrating the efficiency of the updated version of ResNet.

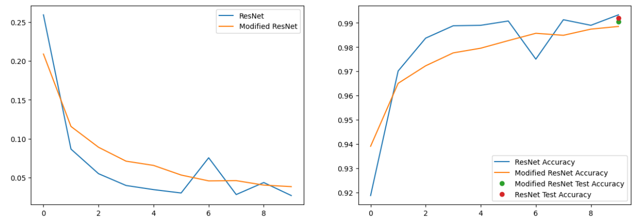

Figure 10. ResNet vs ResNet modified validation Loss (left) and Accuracy (right)

Regarding the loss function, the left subplot shows the training loss over epochs for ResNet (blue line) and the modified ResNet (orange line). Initially, both models start with a relatively high loss, but the loss of the ResNet converges faster in the first epochs. The modified ResNet maintains a slightly higher loss over the epochs, but shows a more stable decrease compared to ResNet, which shows some fluctuations in later epochs. The right subplot plots the training and testing accuracy across epochs for both models and shows an upward trend across epochs, with ResNet achieving slightly higher accuracy, while the modified ResNet appears more stable. The last points, marked with red and green dots, represent the test accuracy for ResNet and Modified ResNet respectively. Both models achieve similar test accuracy, confirming the general performance was not affected with the modifications.

4. Conclusion:

In conclusion, both VGG16 and ResNet34 demonstrated strong performance on the MNIST dataset, showing results with high accuracy and F1 scores. However, ResNet outperformed VGG in terms of training time and generalization capabilities, making it more efficient for this task. Regarding VGG16, while its architecture was simple and effective, it was slower to train and, by modifying it to fit the MNIST dataset, the training time was significantly reduced while maintaining performance. Finally, despite superior results of the ResNet architecture, the added complexity of residual connections may not always justify its use for simpler datasets like MNIST. In general, this analysis highlights the trade-off between accuracy, efficiency, and complexity when choosing between CNN architectures.

Model	Training Time	Accuracy	F1 Score
VGG16	3460s	98.6%	0.985
ResNet34	2815s	99.2%	0.991
Modified VGG	199s	99%	0.99
Modified ResNet	235s	98.9%	0.989

Table 1. Comparison of VGG and ResNet on MNIST Dataset

Table 1 summarizes the performance of the evaluated models. ResNet34 outperforms VGG16 in accuracy (99.2% vs. 98.6%) and F1-score (0.991 vs. 0.985), while also reducing training time (2815s vs. 3460s) due to its residual connections that facilitate gradient flow and faster convergence. The modified versions of them demonstrate the impact of architectural adjustments, achieving comparable accuracy and F1-score with significantly reduced training times (199s for Modified VGG and 235s for Modified ResNet).These findings show the balance and relationship between model complexity, training time, and performance, emphasizing the need to optimize network structures for faster training without losing accuracy and any performance metric used.

5. Additional Documents

To support all the information provided in this report, two additional documents have been included. These documents outline the complete process and execution times for each model, along with the metrics per epoch and the final results. The first document provides detailed execution times, while the second includes the metrics gathered during the training phase and the final evaluation of the models.

References

[1] DigitalOcean. Writing resnet from scratch in pytorch, 2024. 1

[2] DigitalOcean. Vgg from scratch in pytorch, 2024. 1

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[4] PyTorch. Cifar10 — torchvision 0.13.1 documentation, 2025. 1

[5] PyTorch. Mnist — torchvision 0.13.1 documentation, 2025. 1, 2

[6] PyTorch Team. torch.nn.module — pytorch documentation, 2024. 2

[7] PyTorch Team. torchvision.datasets.mnist — torchvision documentation, 2024. 2

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2