Deeper Networks for Image Classification

Sushmitha Mani Gunashekar
s.manigunashekar@se24.qmul.ac.uk

Abstract— **In this work, we construct a hybrid convolutional neural network by combining pretrained ResNet18 and GoogLeNet architectures, to perform image classification. To enable parameter-efficient fine-tuning, selected convolutional layers are augmented with Low-Rank Adaptation (LoRA) modules. The model is evaluated on the MNIST and CIFAR datasets, with ablation studies involving LoRA, projection layer, and their combination to assess the contributions of individual components. Results demonstrate that the LoRA-augmented hybrid model along with projection layer achieves high classification accuracy with minimal additional training overhead, underscoring its potential for scalable and efficient deployment in resource-constrained settings.**

1. Introduction

Image classification has been established as a fundamental task in computer vision, supporting a wide range of applications including handwritten digit recognition, object detection, and medical diagnosis. With the growing complexity of visual data, deep convolutional neural networks (CNNs) have been widely adopted to address the challenges of accurate and efficient classification. In particular, architectures such as ResNet[2] and GoogLeNet [6] have gained prominence due to their ability to learn rich feature representations through deep and structurally diverse layers. These models, pretrained on large-scale datasets such as ImageNet[1], have become standard backbones for transfer learning. By reusing the feature extraction layers of a model trained on a large and diverse dataset, it is possible to drastically reduce training time and computational requirements while achieving competitive accuracy.

In this work, we build on these advances by constructing a Low-Rank Adaptation (LoRA)[4]-augmented hybrid CNN model that combines the feature extraction strengths of both ResNet18 and GoogLeNet . We perform transfer learning by freezing the pretrained backbones and apply parameter-efficient fine-tuning through LoRA modules injected into selected convolutional layers. The combined feature maps are concatenated and passed through a custom classification head, providing a richer representation for improved accuracy. The hybrid model was trained for MNIST dataset and was further evaluated on the CIFAR10 dataset to assess its generalization capabilities.

To investigate the effect of architectural enhancements, ablation studies were performed, to isolate the contribution of each component. Our results show that LoRA is the most effective enhancement, reinforcing its value for scalable deployment. The main contributions of this paper are:

- A CNN model combining ResNet18 and GoogLeNet pretrained backbones, designed to leverage multi-scale and deep feature representations.
- Integration of Low-Rank Adaptation (LoRA) modules into selected convolutional layers to enable parameter-efficient fine-tuning without retraining the full network.
- Comprehensive evaluation on both MNIST and CIFAR datasets, demonstrating generalization across tasks with different complexity levels.
- Ablation studies comparing the proposed hybrid model with LoRA modules to quantify the contribution of each component.

2. Literature Review

The hybrid model designed in this project draws upon two foundational deep learning architectures—ResNet and GoogLeNet—each of which introduced transformative methodologies in convolutional neural network design. These models were selected as the base due to their complementary strengths in feature extraction and representational diversity.

GoogLeNet

GoogLeNet[6], presented a deep yet computationally efficient architecture through its novel Inception modules, which applied convolutions of multiple kernel sizes in parallel to extract multi-scale features. This design allowed for rich spatial representations while keeping parameter counts low, enabled by 1×1 convolutions for dimensionality reduction. Auxiliary classifiers were also introduced mid-network to encourage better gradient flow during training. This model demonstrated how hierarchical features could be aggregated in a scalable manner, a structure not seen in earlier CNNs.

Despite its contributions, GoogLeNet had its drawbacks. The architecture required manual engineering of Inception modules, lacked batch normalization initially, and had limited modularity, making adaptation and expansion challenging. Later improvements like Inception-v2[5], Inception-v3[7] incorporated factorized convolutions and batch normalization, while Inception-v4 and Inception-ResNet[8] merged the inception design with residual learning, improving convergence and depth scalability. These evolutions enhanced both performance and training efficiency, yet the original GoogLeNet remains relevant for its early demonstration of efficient deep architectures.

ResNet (Residual Networks)

ResNet[2] fundamentally changed deep learning by introducing residual connections, which allowed gradients to bypass several layers, mitigating the vanishing gradient problem common in very deep networks. Its identity-based shortcut connections enabled the successful training of architectures exceeding 100 layers, establishing a new benchmark for depth in convolutional networks. ResNet18, a shallower variant, maintains a balance between expressive power and computational efficiency, making it suitable for transfer learning on tasks with smaller datasets.

However, the reliance on identity shortcuts created challenges when input and output dimensions differed, necessitating projection layers. Additionally, performance gains diminished at lower depths, where residual learning offered less benefit. Since its release, numerous extensions have emerged, including Pre-activation ResNet[3], Wide ResNet[10], and ResNeXt[9], each aimed at improving gradient flow, increasing representational capacity, or enhancing efficiency through grouped convolutions. These variants have allowed ResNet to remain competitive even against transformer-based vision models.

Lightweight Fine-Tuning(LoRA)

LoRA[4], originally proposed for NLP tasks, was developed to efficiently fine-tune large pretrained models by injecting low-rank trainable matrices into frozen weights. Instead of updating all parameters, LoRA learns small changes in a reduced-rank space, significantly lowering memory usage and computational cost. This approach is particularly attractive when adapting pretrained models in resource-constrained environments or for tasks where only minor adjustments are needed. For CNNs, LoRA is still an emerging area of study and can be adapted by modifying weight matrices in convolutional filters or linear layers.

One limitation is the lack of standardized integration practices in CNN architectures, which can lead to inconsistent gains. Nonetheless, LoRA provides a compelling mechanism for parameter-efficient transfer learning, and its potential in computer vision continues to be explored. Its lightweight nature makes it suitable for experiments that require quick iteration or deployment on edge devices.

These core components—GoogLeNet and ResNet introduced critical innovations in deep learning, addressing challenges related to depth, efficiency, adaptability, and focus. Their individual advantages and evolving variants make them suitable building blocks for modern image classification models. When combined thoughtfully, as in this project, they allow for the construction of hybrid architectures that are both computationally efficient and performance-rich, even when operating on small or complex datasets like MNIST and CIFAR.

3. Methodology

In this study, a convolutional neural network has been designed by combining two pretrained backbone models - ResNet18 and GoogLeNet, and LoRA is leveraged to selected convolutional layers, to extract their complementary strengths in hierarchical feature extraction and residual learning.

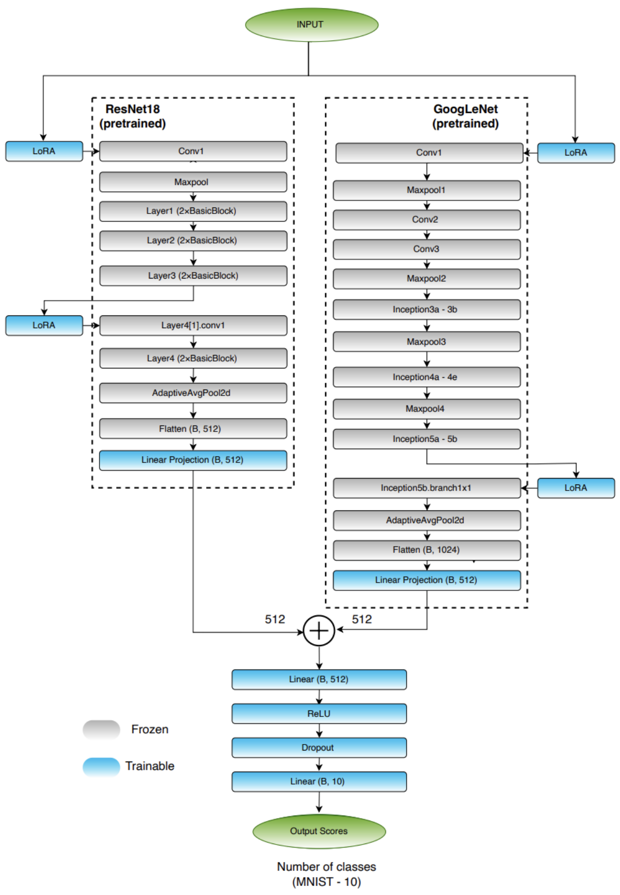


Figure 1. Model Architecture

Dataset Preparation: The MNIST dataset was used as the primary training set, with additional evaluation performed on the CIFAR dataset to assess generalization. The MNIST dataset was preprocessed using a series of augmentations including random rotation, affine transformations, resizing, and color jitter. All grayscale images were converted to three channels to match the input format of pretrained models. All images were resized to 224×224 to conform to the input resolution expected by ResNet and GoogLeNet. Data loaders were configured with a batch size of 128 for training, validation, and testing.

Model Architecture: To enhance feature diversity and improve classification accuracy, we propose a custom hybrid model architecture that integrates ResNet-18 and GoogLeNet, both initialized with pretrained ImageNet weights. As shown in figure 1, the model employs a dual-branch architecture in which both branches process the same input image independently. Each branch is augmented with LoRA modules at selected convolutional layers to allow parameter-efficient fine-tuning.

LoRA module: To enable parameter-efficient fine-tuning, we integrate LoRA modules into selected convolutional layers of the pretrained backbones. Each LoRA module freezes the original convolutional weights and adds two trainable 1×1 convolutions: one projecting the input to a lower-dimensional space and another projecting it back. The resulting low-rank residual is scaled and added to the frozen output, allowing lightweight, task-specific adaptation without updating the full parameter set. During training, only the LoRA parameters and the classification head are updated, ensuring high efficiency and adaptability.

ResNet-18 Branch: The first branch uses ResNet-18, a residual network composed of skip-connected blocks that facilitate gradient flow through deep layers. In our implementation, we retain all layers, freezing their weights to preserve the general feature extraction capabilities. The LoRA modules are inserted into the initial convolutional layer and the second convolutional layer of Layer4. Following feature extraction, an adaptive average pooling layer reduces the spatial dimensions to (1, 1), yielding an output of shape (B, 512, 1, 1), which is then flattened and projected to a 512-dimensional vector.

GoogLeNet Branch: The second branch is based on GoogLeNet, which uses Inception modules to perform multi-scale convolutions in parallel. This structure allows for capturing features at multiple resolutions. Similar to the ResNet-18 branch, we freeze all layers and insert LoRAConv2d modules at the initial convolution (Conv1) and in the final inception module. After feature extraction, the (B, 1024, 7, 7) output is pooled to (B, 1024, 1, 1), flattened, and linearly projected to a 512-dimensional vector.

Feature Fusion and Classification: The outputs from both branches are concatenated to form a combined feature vector. This representation is passed through a classification head consisting of a linear layer, ReLU activation, Dropout, and a final linear layer that maps to the output classes corresponding to the digits 0–9 in the MNIST dataset.

4. Implementation of Model Training and Test Settings

To support the experimental results, the model training and testing settings were carefully designed and systematically executed. This section outlines the training process, testing methodology, monitoring techniques, and runtime evidence to validate reproducibility and convergence.

Training Process: The hybrid model was fine-tuned using the **Adam optimizer** with **cross-entropy loss** as the objective function. During training, the pretrained backbone layers of ResNet-18 and GoogLeNet were frozen to reduce computational overhead, while the LoRA modules and the classification head were updated. Mixed-precision training was employed to accelerate computation and reduce memory consumption.

The model was trained separately on the MNIST and CIFAR-10 datasets with specific training configurations suited to each dataset's complexity. For MNIST, the model was trained with a **batch size of 64** and an initial **learning rate of 0.0001**. This configuration led to stable convergence and a high validation accuracy of **98.97%** without the need for further hyperparameter adjustment.

For CIFAR-10, an initial training run used a batch size of 64 and a learning rate of 0.0001, achieving a validation accuracy of approximately 87%. To further optimize performance, a second training run was conducted with an increased batch size of 256 and a higher learning rate of 0.001. This adjustment resulted in an improved validation accuracy of approximately 89.87%. These changes reflect the hybrid model's sensitivity to dataset complexity and optimization settings.

Across both datasets, the model was trained for 40 epochs, and intermediate checkpoints were saved after each epoch to enable model recovery and facilitate detailed performance tracking. Key training metrics, including loss and accuracy, were logged at each epoch using TensorBoard for real-time monitoring and analysis.

Testing Process: After training, the best-performing model checkpoint, selected based on validation accuracy was evaluated on the test splits of MNIST and CIFAR-10.

Performance Evaluation: The model's performance was evaluated based on accuracy and loss on both validation and test sets. A dedicated test loop was implemented to compute the final classification accuracy on the unseen MNIST test data. In addition to overall accuracy, confusion matrices and sample prediction grids were generated to qualitatively analyze model behavior and misclassification patterns.

Result Visualization: Training and validation loss curves, as well as accuracy curves, were plotted for both datasets. These plots, visually confirm smooth convergence trends. For MNIST, rapid early convergence was observed, with gradual improvements toward the later epochs. For CIFAR-10, convergence was slower but steady throughout the training period. TensorBoard logs provided additional insights, allowing real-time tracking of loss and accuracy, and confirming that no major overfitting or instability occurred during training.

Runtime Screenshots and Evidence Screenshots capturing the TensorBoard dashboards, training loss and accuracy curves, and terminal logs were recorded throughout the experiments to ensure transparency and reproducibility. These runtime evidences demonstrate model behavior across different stages of training and testing and have been consolidated into Appendix for reference. They provide a visual record of training, validation, and final model performance. They provide a clear view of training progress, convergence rates, and the accuracy trends over time. These include, TensorBoard loss and accuracy plots after 40 epochs of MNIST training and logs showing the MNIST model achieving a test accuracy of 98.56%.

5. Experimental Results

The proposed LoRA-augmented hybrid model was evaluated on the MNIST test set based on overall classification accuracy, convergence behavior, and class-wise performance. It achieved a final test accuracy of **98.56%**, with training and validation curves showing smooth convergence and minimal overfitting, as illustrated in Figure 2.

To assess generalization beyond the MNIST distribution, the model was also tested on external handwritten digit samples not seen during training. As shown in

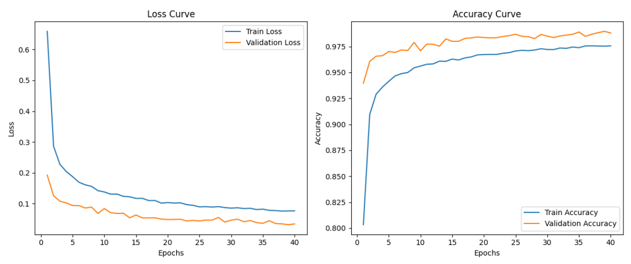


Figure 2. Training and validation accuracy/loss curves on MNIST dataset

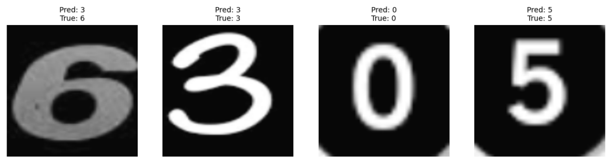


Figure 3. Predictions on external digit samples (not part of MNIST)

Figure 3, the model handled varied shapes and moderate noise well, with most digits classified correctly. This suggests robustness in out-of-distribution settings, although a few misclassifications indicate room for further improvement.

These findings confirm that the proposed model generalizes effectively to grayscale digit classification. However, for broader applicability—including real-world images with varied color, style, and background—future work should consider partial backbone fine-tuning and more adaptive fusion strategies.

For CIFAR-10, the model achieved a final test accuracy of **89.36%**, demonstrating good adaptability even when applied to a more visually complex and diverse dataset. The evaluation was performed using a separate test loop, with no gradient updates, to ensure a fair and unbiased assessment of model performance. As shown in table 2, the relative improvements across model variants were less pronounced in CIFAR-10 compared to MNIST under the same number of training epochs, likely due to the richer visual features in CIFAR and the limited fine-tuning depth allowed by freezing the backbone networks.

Quantitative Analysis: To evaluate class-wise performance, precision, recall, and F1-score were computed for the LoRA-augmented hybrid model on the MNIST test set, as summarized in Table 1.

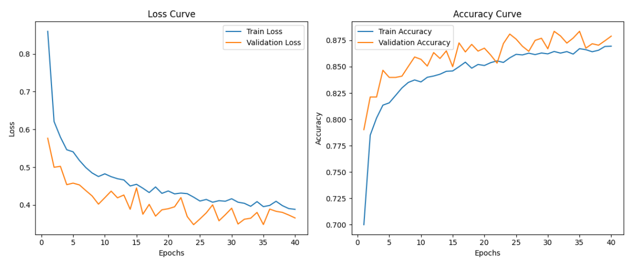


Figure 4. Training and validation accuracy/loss curves on CIFAR-10 dataset

Class	**Precision**	**Recall**	**F1-score**
0	0.9879	0.9959	0.9919
1	0.9843	0.9965	0.9904
2	0.9826	0.9835	0.9831
3	0.9747	0.9911	0.9828
4	0.9837	0.9847	0.9842
5	0.9909	0.9753	0.9831
6	0.9916	0.9864	0.9890
7	0.9835	0.9854	0.9845
8	0.9958	0.9805	0.9881
9	0.9840	0.9752	0.9796
Avg	0.9859	0.9855	0.9857

Table 1. Precision, Recall, and F1-score on MNIST test set (LoRA model)

The model achieved consistently high macro-average scores across all metrics, highlighting its ability to generalize effectively to different digit classes. Digit 0 achieved the highest F1-score, while digit 9 had the lowest, although the overall variations remained small. This indicates that the model maintains strong and balanced classification performance without being overly biased toward specific classes.

The corresponding confusion matrix (Figure 5) provides additional insights into the model's behavior. Most digits were classified with high accuracy, and errors were rare and typically occurred between visually similar digits. For example, digit 5 was occasionally misclassified as 3 or 6, and digit 8 showed minor confusion with 2 and 9. Digits such as 0, 1, and 7 exhibited near-perfect precision and recall, reflecting the model's strong capability in distinguishing simpler and more distinct digit shapes. The prominent diagonal pattern in the confusion matrix further confirms the model's ability to learn discriminative features while minimizing

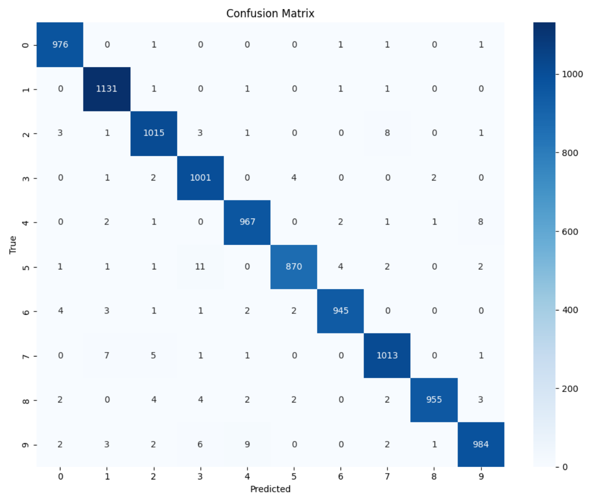


Figure 5. Confusion matrix

false positives and negatives.

Overall, these results demonstrate that the proposed model not only achieves high overall accuracy but also delivers reliable, class-wise consistency, even when subtle variations exist within the handwritten digits.

6. Ablation Study

To evaluate the contribution of individual architectural enhancements, an ablation study was conducted based on the proposed hybrid model with LoRA modules. Four variants were explored relative to this base model, as summarized in 2, all trained on the MNIST dataset using identical preprocessing, optimizer, and learning rate settings.

Model Variant	**Accuracy**
Base model	98.56%
Variant 1	95.09%
Variant 2	98.14%
Variant 3	97.67%
Variant 4	97.89%

Table 2. Ablation study results on MNIST and CIFAR datasets for different model variants.

In Variant 1, the model was trained without any LoRA modules, relying solely on the classification head over frozen ResNet-18 and GoogLeNet features. The notable drop in accuracy highlights the importance of LoRA modules in enhancing feature adaptation and task-specific tuning. Variant 2 introduced LoRA modules only in the first layers, substantially recovering the lost performance and emphasizing the critical role of early-layer adaptation. In Variant 3, applying LoRA modules to the middle layers resulted in a modest improvement, suggesting that mid-level adaptation is beneficial but less impactful than early-layer adjustments. Variant 4 removed the linear projection layer while retaining LoRA modules, leading to a slight performance reduction and indicating that the projection layer aids in better feature-to-class alignment.

Overall, the results show that LoRA modules, particularly when applied to early layers, significantly enhance performance, while the model remains robust across all variant designs with only minor accuracy fluctuations.

7. Conclusion

This study presented a hybrid image classification model combining ResNet-18 and GoogLeNet, enhanced with LoRA modules for parameter-efficient fine-tuning. The experimental results demonstrated that the LoRA-augmented hybrid model achieved highest accuracy on the MNIST dataset. However, when evaluated on the more complex CIFAR-10 dataset, training time increased significantly, and relative performance gains diminished. This outcome highlights the limitations of relying solely on frozen backbone architectures for complex, high-variability datasets.

Future work could explore partial fine-tuning of backbone layers, adaptive feature fusion strategies, or more sophisticated classification heads to boost performance. Additionally, extending the model's capabilities to better generalize beyond handwritten digits—particularly to images with varied styles, textures, and colored backgrounds—would be essential for broader applicability in real-world classification tasks.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[2] 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. 1, 2

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 630–645. Springer, 2016. 2

[4] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 1, 2

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015. 2

[6] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 1, 2

[7] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 2

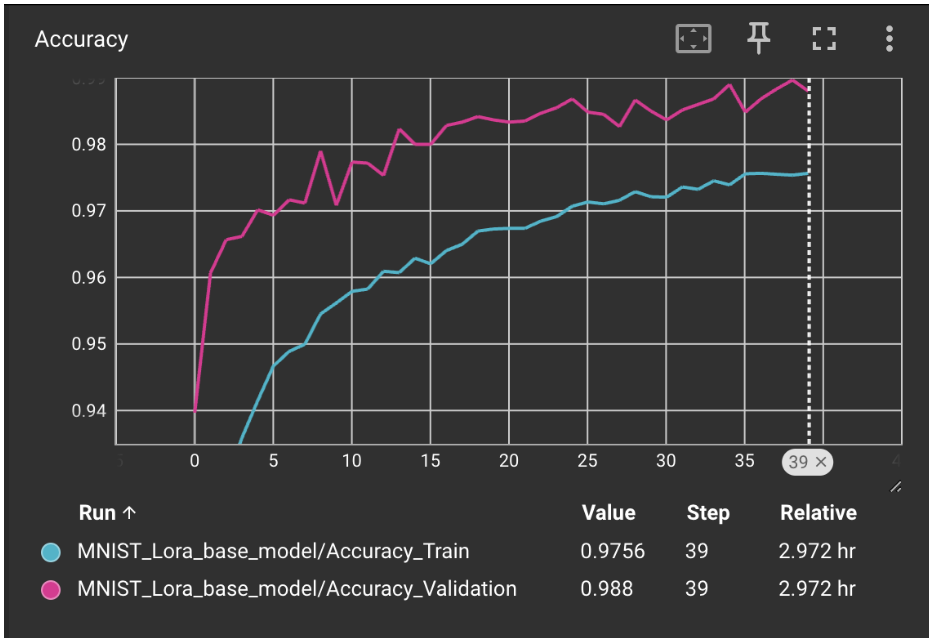
[8] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*, 2017. 2

[9] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 2

[10] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 2

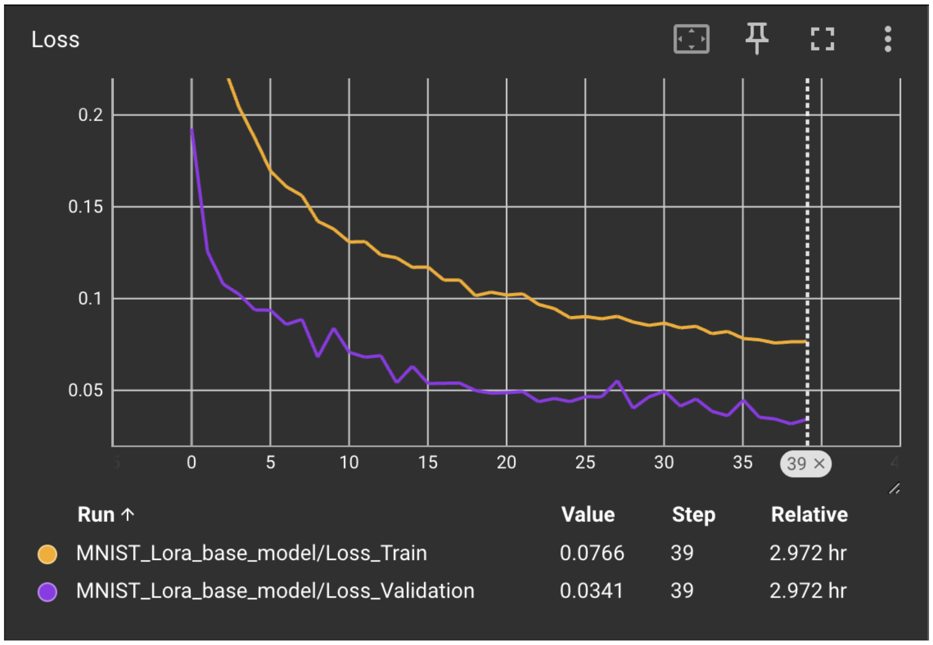
Appendix

Proof of Results



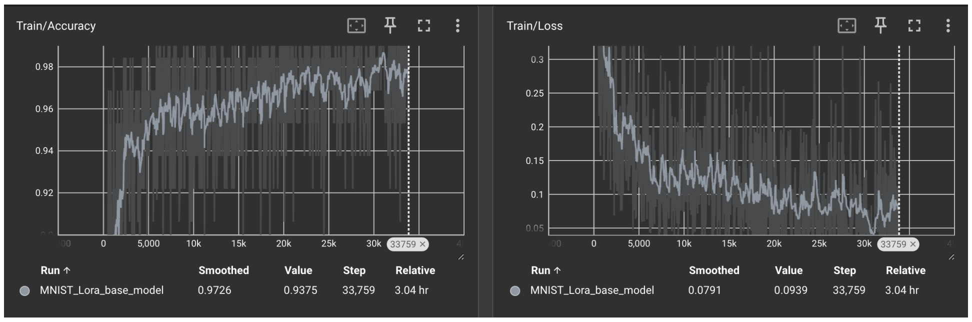
	Run ↑	Value	Step	Relative
●	MNIST_Lora_base_model/Accuracy_Train	0.9756	39	2.972 hr
●	MNIST_Lora_base_model/Accuracy_Validation	0.988	39	2.972 hr

Figure 6. Training and validation accuracy for every epoch



	Run ↑	Value	Step	Relative
●	MNIST_Lora_base_model/Loss_Train	0.0766	39	2.972 hr
●	MNIST_Lora_base_model/Loss_Validation	0.0341	39	2.972 hr

Figure 7. Training and validation loss for every epoch

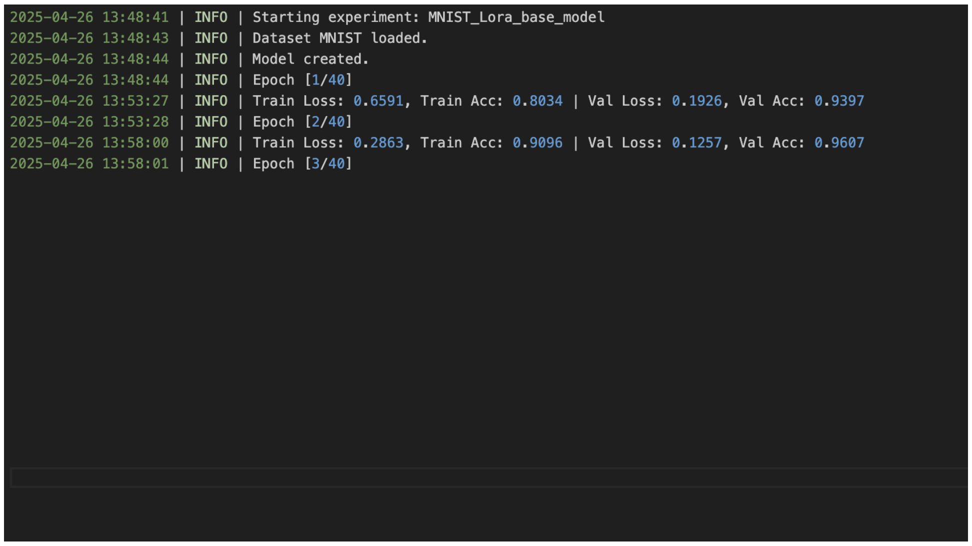


Run ↑	Smoothed	Value	Step	Relative
● MNIST_Lora_base_model	0.9726	0.9375	33,759	3.04 hr

Run ↑	Smoothed	Value	Step	Relative
● MNIST_Lora_base_model	0.0791	0.0939	33,759	3.04 hr

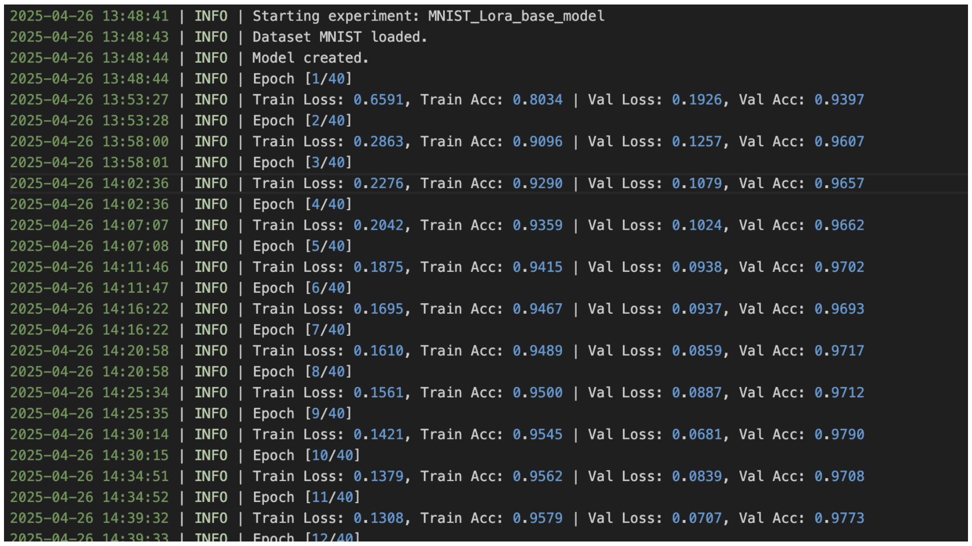
Figure 8. Tensorboard screenshots showing (right) loss and (left) accuracy of every iteration during training.

Runtime screenshots



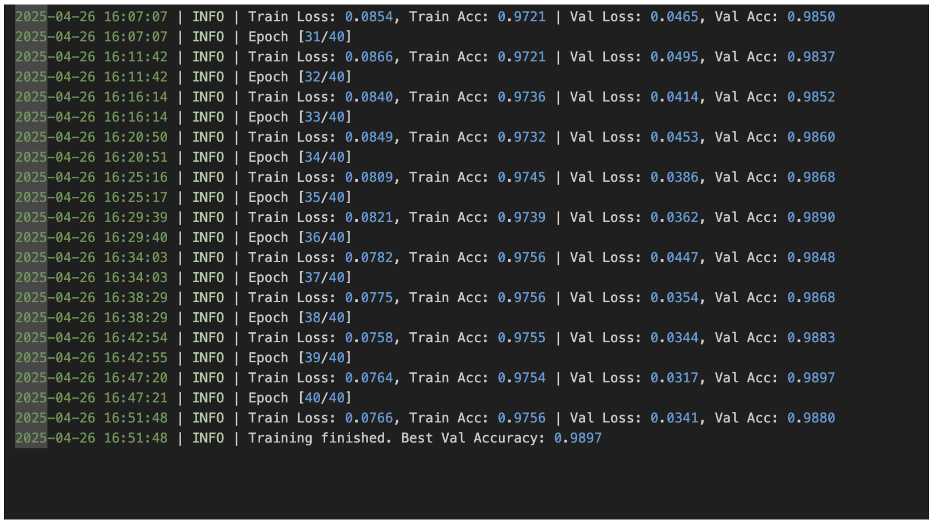
```
2025-04-26 13:48:41 | INFO | Starting experiment: MNIST_Lora_base_model
2025-04-26 13:48:43 | INFO | Dataset MNIST loaded.
2025-04-26 13:48:44 | INFO | Model created.
2025-04-26 13:48:44 | INFO | Epoch [1/40]
2025-04-26 13:53:27 | INFO | Train Loss: 0.6591, Train Acc: 0.8034 | Val Loss: 0.1926, Val Acc: 0.9397
2025-04-26 13:53:28 | INFO | Epoch [2/40]
2025-04-26 13:58:00 | INFO | Train Loss: 0.2863, Train Acc: 0.9096 | Val Loss: 0.1257, Val Acc: 0.9607
2025-04-26 13:58:01 | INFO | Epoch [3/40]
```

Figure 9. Runtime Screenshot 1



```
2025-04-26 13:48:41 | INFO | Starting experiment: MNIST_Lora_base_model
2025-04-26 13:48:43 | INFO | Dataset MNIST loaded.
2025-04-26 13:48:44 | INFO | Model created.
2025-04-26 13:48:44 | INFO | Epoch [1/40]
2025-04-26 13:53:27 | INFO | Train Loss: 0.6591, Train Acc: 0.8034 | Val Loss: 0.1926, Val Acc: 0.9397
2025-04-26 13:53:28 | INFO | Epoch [2/40]
2025-04-26 13:58:00 | INFO | Train Loss: 0.2863, Train Acc: 0.9096 | Val Loss: 0.1257, Val Acc: 0.9607
2025-04-26 13:58:01 | INFO | Epoch [3/40]
2025-04-26 14:02:36 | INFO | Train Loss: 0.2276, Train Acc: 0.9290 | Val Loss: 0.1079, Val Acc: 0.9657
2025-04-26 14:02:36 | INFO | Epoch [4/40]
2025-04-26 14:07:07 | INFO | Train Loss: 0.2042, Train Acc: 0.9359 | Val Loss: 0.1024, Val Acc: 0.9662
2025-04-26 14:07:08 | INFO | Epoch [5/40]
2025-04-26 14:11:46 | INFO | Train Loss: 0.1875, Train Acc: 0.9415 | Val Loss: 0.0938, Val Acc: 0.9702
2025-04-26 14:11:47 | INFO | Epoch [6/40]
2025-04-26 14:16:22 | INFO | Train Loss: 0.1695, Train Acc: 0.9467 | Val Loss: 0.0937, Val Acc: 0.9693
2025-04-26 14:16:22 | INFO | Epoch [7/40]
2025-04-26 14:20:58 | INFO | Train Loss: 0.1610, Train Acc: 0.9489 | Val Loss: 0.0859, Val Acc: 0.9717
2025-04-26 14:20:58 | INFO | Epoch [8/40]
2025-04-26 14:25:34 | INFO | Train Loss: 0.1561, Train Acc: 0.9500 | Val Loss: 0.0887, Val Acc: 0.9712
2025-04-26 14:25:35 | INFO | Epoch [9/40]
2025-04-26 14:30:14 | INFO | Train Loss: 0.1421, Train Acc: 0.9545 | Val Loss: 0.0681, Val Acc: 0.9790
2025-04-26 14:30:15 | INFO | Epoch [10/40]
2025-04-26 14:34:51 | INFO | Train Loss: 0.1379, Train Acc: 0.9562 | Val Loss: 0.0839, Val Acc: 0.9708
2025-04-26 14:34:52 | INFO | Epoch [11/40]
2025-04-26 14:39:32 | INFO | Train Loss: 0.1308, Train Acc: 0.9579 | Val Loss: 0.0707, Val Acc: 0.9773
2025-04-26 14:39:33 | INFO | Epoch [12/40]
```

Figure 10. Runtime Screenshot 2



```
2025-04-26 16:07:07 | INFO | Train Loss: 0.0854, Train Acc: 0.9721 | Val Loss: 0.0465, Val Acc: 0.9850
2025-04-26 16:07:07 | INFO | Epoch [31/40]
2025-04-26 16:11:42 | INFO | Train Loss: 0.0866, Train Acc: 0.9721 | Val Loss: 0.0495, Val Acc: 0.9837
2025-04-26 16:11:42 | INFO | Epoch [32/40]
2025-04-26 16:16:14 | INFO | Train Loss: 0.0840, Train Acc: 0.9736 | Val Loss: 0.0414, Val Acc: 0.9852
2025-04-26 16:16:14 | INFO | Epoch [33/40]
2025-04-26 16:20:50 | INFO | Train Loss: 0.0849, Train Acc: 0.9732 | Val Loss: 0.0453, Val Acc: 0.9860
2025-04-26 16:20:51 | INFO | Epoch [34/40]
2025-04-26 16:25:16 | INFO | Train Loss: 0.0809, Train Acc: 0.9745 | Val Loss: 0.0386, Val Acc: 0.9868
2025-04-26 16:25:17 | INFO | Epoch [35/40]
2025-04-26 16:29:39 | INFO | Train Loss: 0.0821, Train Acc: 0.9739 | Val Loss: 0.0362, Val Acc: 0.9890
2025-04-26 16:29:40 | INFO | Epoch [36/40]
2025-04-26 16:34:03 | INFO | Train Loss: 0.0782, Train Acc: 0.9756 | Val Loss: 0.0447, Val Acc: 0.9848
2025-04-26 16:34:03 | INFO | Epoch [37/40]
2025-04-26 16:38:29 | INFO | Train Loss: 0.0775, Train Acc: 0.9756 | Val Loss: 0.0354, Val Acc: 0.9868
2025-04-26 16:38:29 | INFO | Epoch [38/40]
2025-04-26 16:42:54 | INFO | Train Loss: 0.0758, Train Acc: 0.9755 | Val Loss: 0.0344, Val Acc: 0.9883
2025-04-26 16:42:55 | INFO | Epoch [39/40]
2025-04-26 16:47:20 | INFO | Train Loss: 0.0764, Train Acc: 0.9754 | Val Loss: 0.0317, Val Acc: 0.9897
2025-04-26 16:47:21 | INFO | Epoch [40/40]
2025-04-26 16:51:48 | INFO | Train Loss: 0.0766, Train Acc: 0.9756 | Val Loss: 0.0341, Val Acc: 0.9880
2025-04-26 16:51:48 | INFO | Training finished. Best Val Accuracy: 0.9897
```

Figure 11. Runtime Screenshot 3

Few other cases of success and failure

As observed in Figure 12, the model exhibits confusion when confronted with unseen samples that share structural similarities, leading to misclassifications. Addressing this challenge remains an important direction for future work.

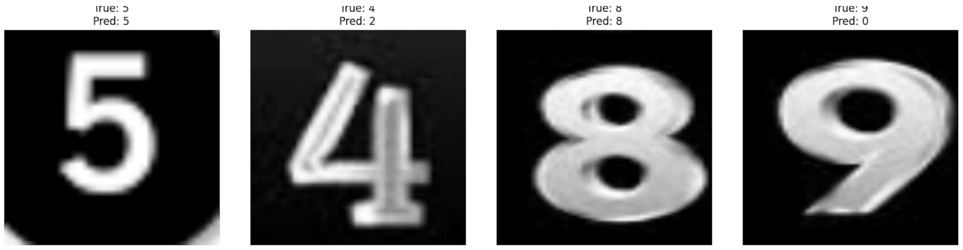


Figure 12. Few more predictions on external digit samples (not part of MNIST)

Potential improvements include the integration of attention-based mechanisms to refine feature discrimination, partial fine-tuning of backbone layers to capture more task-specific representations, and augmentation of the training dataset with synthetically varied samples to improve robustness against distributional shifts.