Deeper Networks for Image Classification

Tanisha Hisariya
220929356
ec23691@qmul.ac.uk

Abstract— **Image Classification is a remarkable achievement in the computer vision field with impactful implementation in medical, media, and many other industries. Traditional models were making this task complicated due to a large number of images, this piqued the appearance of deep learning models. Deep learning algorithms have an immense capacity for processing a large number of images giving efficient accuracy. This paper delves into the study of deeper network models like ResNet, VGG, and DenseNet, assessing their performance on image classification tasks. It extends into the modification of these models against their standard configuration, providing a comparative analysis in terms of accuracy and processing speed. Furthermore, it extends into possible future work of models.**

Key Words: Image Classification, Deep Learning, Convolutional Neural Networks.

I. INTRODUCTION

Image Classification is a process of classifying images into respective classes and labels based on their specific features and characteristics. Their classification features are being identified by an algorithm. With the advancement of AI technology, there has been breakthrough research in this area. Earlier machine learning models were being used in this task which involved supervised and unsupervised learning. Though these models were not easy to train they gave very good results for small classes, yet their performance for multi-label classes was limited. With the amount of increase in data in this digitization, these models were giving low effectiveness with high resource optimization which created a research interest in using deep learning algorithms.

Convolutional Neural Networks (CNN)[5] has been emerged marking a shift from conventional networks to Deep Learning. With an excellent feature of autolearning from the closely related features of training data, it achieved great accuracy. However, issues like model overfitting persist. In recent years, there has been much research going on CNN[4][8][9] to improve the model efficiency and complexity time.

This paper will prominently focus on some top-performing CNN models – Resnet34, VGG19, DenseNet121 on MNIST, and CIFAR10 dataset. Each of these models has been chosen for its unique approach to performing this task by assessing their advantages and disadvantages. By Implementing modification against their original code, the paper will offer some insights into deeper network models.

II. RELATED WORKS

The inception of deeper network architectures [7] can be traced back to when in 1989 ConvNet[15] was proposed by LeCun et. al. which was based on supervised learning. Following this in 1998, an upgraded network- LeNet-5[16] was launched, marking a significant advancement in this field. The principles of the network's architecture such as layer stacking and, incorporation of conventional and pooling layers laid the foundation for further research area. The advent of a Graphical Processing Unit (GPU) and a larger training dataset further catalyzed the evolution of deep convolutional neural networks (DCNNs). In the early 2010s, this was then fuelled by the introduction of AlexNet[16] introducing various new features like regularization and multiple layers with unsaturated neurons improving the model performance.

Visual Geometry Group (VGG) came as a breakthrough in 2014 by Simonyan and Zisserman [2] with the implementation of a 3 X 3 convolutional kernel and 2 X 2 pooling kernel, they achieved remarkable performance on ImageNet [13] Dataset. Their architecture increased depth by adding more layers which in turn raised the number of trainable parameters and led to higher computational cost. With the variation of VGG architecture, as the layers have been deepening the gradient explosion becomes prominent leading to a gradient degradation problem.

Resnet [1] developed by He Kaiming et al addresses this degradation problem by introducing an additional residual block in its architecture and utilizing batch normalization. The implementation of shortcut connections helps the network to automatically learn the features and alleviates the disappearance of gradient problems. Even though with the addition of an extra block this contains less number of trainable parameters if compared with the VGG model.

Building upon the architecture of ResNet, DenseNet was introduced by Huang et al. [3]. This network modified the connection between layers by connecting them in a feed-forward manner. In this way, feature maps of the next layers can be directly accessed thus gaining maximum information flow.

Subsequently, in 2010, Glorot and Bengio, [17] addressed the challenges posed by deep models and their difficulty in training and suggested some ways to mitigate those. The major concern for this was with the random weight initialization method and activation function. However, they highlighted the use of Xavier or

He initialization is more effective. Also, non-saturated activation functions such as ReLU can help in the preservance of gradient flow leading a better training of the model.

Additionally, in 2015 Srivastava et.al. [10] emphasized the importance of using the dropouts layer to reduce overfitting in neural networks. They break the coadaptation of the network making a trade-off between efficiency and increased training time. Ioffe and Szegedy [11] emphasized training a neural network in batches and the importance of batch normalization. This leads to attaining more accuracy after training fewer epochs. Recent researchers Yosinski et al. and Razavian et al. talked about transfer learning [6][15][16] which is training a pre-trained model on some other dataset. Moreover, to achieve a better accuracy and speed trade-off, researchers have been working to enhance the deeper network models.

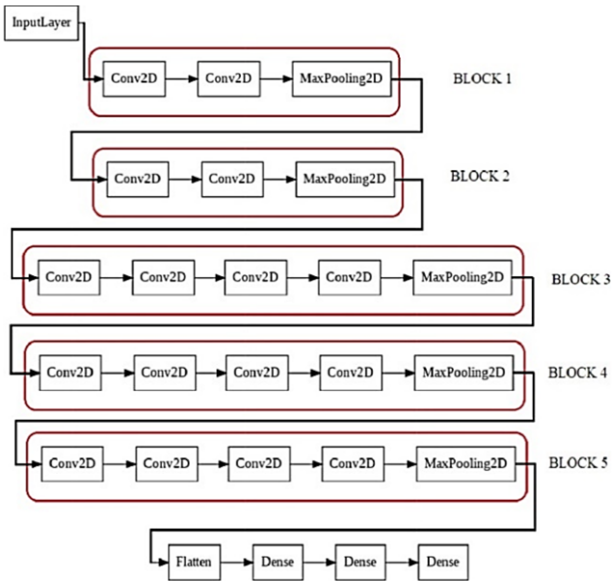
III. METHODOLOGY

A. MODEL SELECTION

VGG - 19

VGG-19 being as the improvement over previous networks featuring a homogeneous topology and a more deeply layered model. The model comprises a of total 19 layers including 16 convolutional layers followed by 3 fully connected layers. Each convolutional layer has a kernel of 3 X 3, this small size of the kernel does not lead to huge growth in parameters as it requires fewer parameters and yet maintains the processing capacity of complex features. Even the effective features of two 3 X 3 kernels are equal to one 5 X 5 kernel but with fewer parameters. This is one of the major advantages of the VGG-19 model.

Fig 1. VGG-19 architecture



Though convolutional layers make no changes in image dimensions so, successive convolutional layers of the model are accompanied by max-pooling layers. The pooling layers have been used to reduce the input's dimension by halving the dimensions of images thus leading to reduced complexity time. Despite the efficient architecture of the model, it is still highly computational as with the increased rate of layers there are in total of 144 million parameters to train.

The model was originally trained on the ILSVRC dataset, giving inputs as batches for faster GPU processing, with random weight initialization, and SGD [14] optimizer to update weights. This all results in intrusive training as regularisation techniques like batch normalization were not included in an earlier phase.

ResNet-34

ResNet-34 is a variation of ResNet architecture with a deeper model. As the name, the model is comprised of 34 layers including a convolutional layer, batch normalization layer, activation layer, and at the end a fully connected layer. The most important feature is a residual layer in Resnet architecture consisting of two 3 X 3 convolutional layers which is followed by a shortcut connection. This connection is very useful in vanishing the gradient problem suffered by previous models as it connects the block's input with the output. Even though its architecture is more depth when compared to the total parameters with the VGG model it still has only 18% of it, thus making it more computationally effective. This computational efficiency is because of dimensionality reduction because of using global average pooling before the final fully connected layer. The introduction of batch normalization is another important feature here which is applied before activation, helping in input data pre-processing of each layer, solving gradient explosion and overfitting problems. To stabilize the learning process, it fully relies on the batch normalization technique and unlike VGG it does not incorporate dropouts.

The model was originally trained on the ImageNet dataset, giving inputs as batches for faster GPU processing, with weight initialization as in [12], SGD optimizer to update weights, and batch regularisation technique.

Fig 2. ResNet-34 architecture

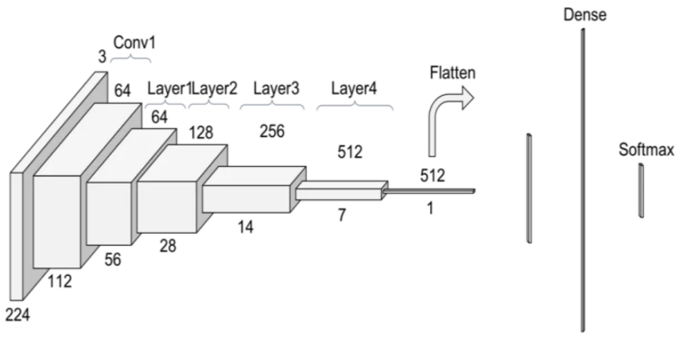
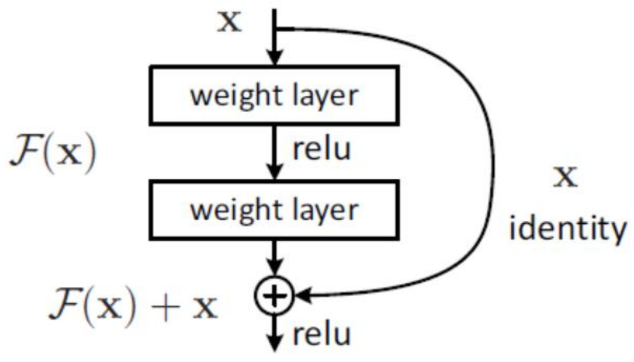


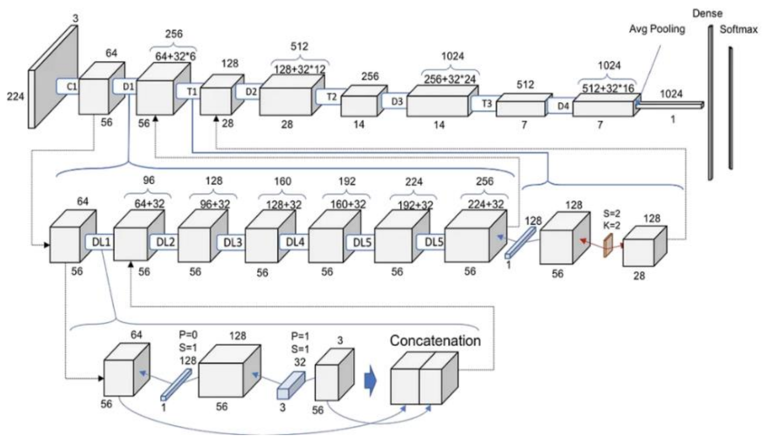
Fig 3. Residual block architecture


x
weight layer
relu
$\mathcal{F}(\mathbf{x})$
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

DenseNet-121

DenseNet-121 is a variant of DenseNet with 121 layers comprising dense blocks and transition blocks. It requires around 8 million parameters to train which is much less than other networks. In a dense block, each layer is connected featuring dense connectivity which ensures maximum information flow across the network. Even this way of connection helps in reducing the redundancy by allowing the access of features of all proceeding layers and concatenating them together and they pass further. This makes the same dimension of the feature map in the dense block but changes in number of filters. Transition blocks consist of either 1 X 1 convolution kernel and 2 X 2 average pooling and are placed between two dense blocks. This placement of the transition block helps in reducing the dimensionality of inputs. There is a new addition of hyperparameters for this model which is termed as growth rate. Since we are adding information to each layer then there is a need to regulate the amount of information, that’s what growth rates do.

Fig 4. DenseNet-121 architecture

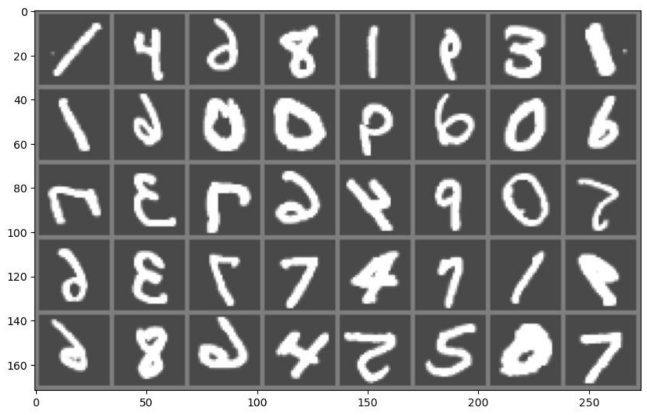


It was initially implemented on Imagenet and a variety of datasets, giving inputs as batches for faster GPU processing. It adopts a similar weight initialization method as used in Resnet architecture. Additionally, it incorporates both batch normalization and dropouts to make stable training and prevent overfitting.

B. DATASET SELECTION

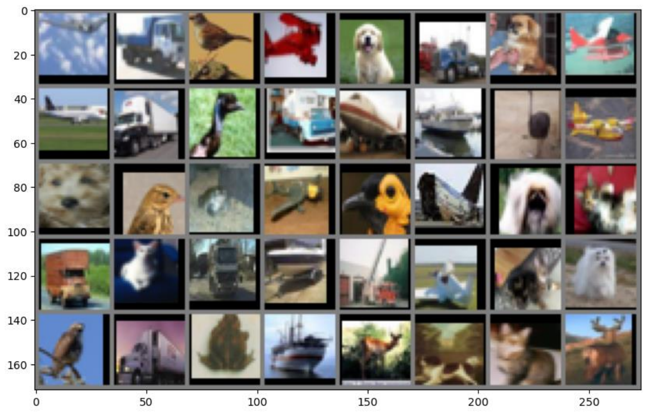
MNIST

MNIST datasets provided by Yann LeCun et. al. [18] are a collection of handwritten digits (0 to 9) and are widely used for neural networks. The dataset has a total of 70000 images comprising 60000 training datasets and 10000 test datasets. Each of the images is greyscale image with resolution of 28 x 28 pixels making it ideal for projects due to its simplicity and manageable size. The availability of these datasets across the Torchvision library makes it easy to import and use for models. The standard format and pre-processing technique applied to the dataset ensure consistency across different model evaluations and implementations.



CIFAR-10

CIFAR-10, a benchmark dataset for image classification tasks, is tiny scale image datasets that belong to 10 classes which include airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The dataset has a total of 60000 images comprising 50000 training and 10000 test datasets. CIFAR-10 introduces the complexities of real-world visual data as the images are 3-channel RGB of 32 X 32 pixels. CIFAR-10 is also available on the Torchvision library making it easy to use for model.



IV. EXPERIMENTS

A. Dataset Preparation

MNIST dataset has been transformed by resizing it to 32 X 32 pixels, flipping it horizontally, and normalizing it to the range [0,1]. Similarly, CIFAR-10 datasets have been pre-processed differently for testing and training. The training dataset has added padding of 4 and a random 32 X 32 pixel from the image, followed by flipping and normalizing pixels. For the test datasets, we just normalized them. Both the datasets has been loaded as batches to facilitate parallel processing.

B. Model Implementation

VGG-19 model was implemented using TorchVision with the same architecture as defined in Fig.1. Though this time the model was implemented with the addition of batch normalization, using a suitable SGD optimizer and categorical cross Entropy loss. In this implementation, the weight initialization technique was also used as Xavier.

The Resnet-34 model is also being implemented with the TorchVision library following the same architecture as defined. The difference with the original implementation of the model is the addition of a dropout layer in its fully connected layer and the Xavier weight initialization method. The optimizer and categorical used are SGD and cross-entropy loss respectively.

For DenseNet-121 no modification to the original code has been made, thus the implementation goes the same as defined in the original implementation.

C. Model Training

All three models have been trained on both MNIST and CIFAR-10 datasets. The validation set is used to monitor the performance of the model and the training set is used to update its parameters. The model has been fine-tuned with the number of epochs and learning rate to avoid overfitting. The learning rate for all three models on both datasets has been set to 0.001. The number of epochs for the MNIST dataset is 20 for VGG-19 and Resnet-34 and 15 for DenseNet-121. For CIFAR-10, VGG-19 and ResNet-34 have been trained using 50 epochs and DenseNet-121 uses 40 epochs. All the models have been trained on JupyterHub and use GPU CUDA for processing. The batch sizes are generally 128 for MNIST and 64 for CIFAR-10.

D. Model Evaluation

All three models have been evaluated based on the accuracy and loss of both datasets. These metrics were used for assessing the practical efficacy of each model providing a quantitative measure of model performance and generalization capability.

E. Comparitive Analysis

A thorough analysis of the performances of models VGG-19, ResNet-34, and DenseNet-121 will be compared. The comparison is done based on the accuracy and loss of both training and test datasets. Furthermore, the models will be also analysed in the terms of total computation time they took.

F. Experimental Result

The results are visually represented through graphs of accuracy and loss for both the datasets across each model. It is further complemented with a table for the accuracy of models. Furthermore, examples of correctly and incorrectly identified images were provided giving its predictive capabilities. Also, the results of a modified model of Resnet-34 and VGG-19 will be analyzed with its original model.

G. Test Result

According to the Table 1. All the models achieve a high accuracy of 98% or higher, with loss of almost zero for MNIST datasets.

Table 1. MNIST Results

Model	Test Accuracy
Original VGG-19	11.35%
Modified VGG-19	99.05 %
Original resNet-34	98.19 %
Modified ResNet-34	99.06 %
DenseNet-121	98.96 %

According to the Table 2. All the models achieve a high accuracy of 87% or higher, with loss of almost zero for CIFAR-10 datasets.

Table 2. CIFAR Results

Model	Test Accuracy
Modified VGG-19	87.54 %
Original ResNet-34	90.01 %
Modified ResNet-34	91.18 %
DenseNet-121	87.59 %

Figure 5. Validation loss of CIFAR dataset

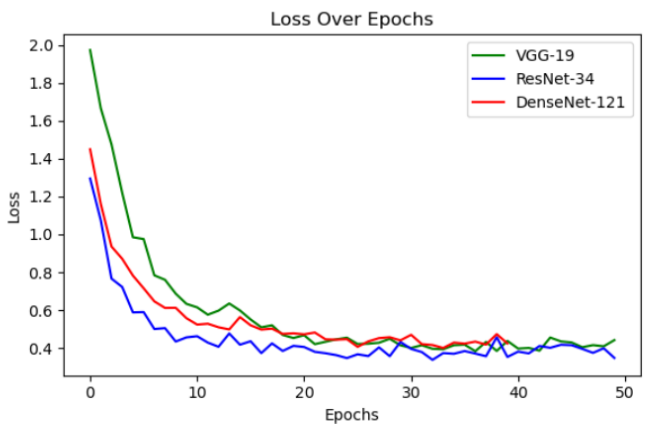


Figure 6. Validation accuracy of CIFAR dataset

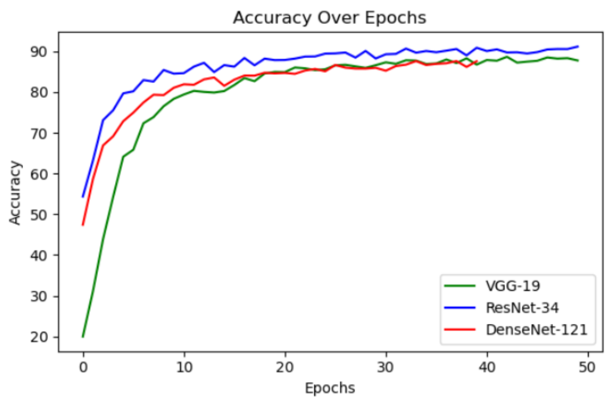


Figure 7. Validation Loss of MNIST dataset

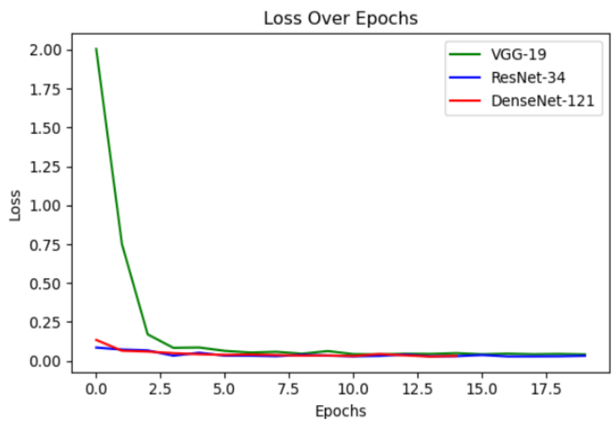
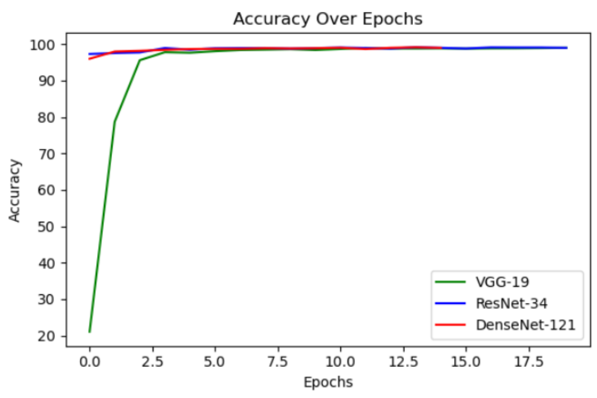


Figure 8. Validation Accuracy of MNIST dataset



V. DISCUSSIONS

A. *Analysis of MNIST results*

From Table 1, it can be seen that the first model to evaluate the performance is VGG-19. The original VGG-19 model yields a very low performance with an accuracy of 11.35%. This can be due to many reasons such as incorrect initialization or overfitting. However, the modified VGG-19 which majorly includes the batch normalization method demonstrated a dramatic improvement in accuracy by getting 99.05%. Adding the batch normalization methods stabilized the training process as the input of each layer tends to be normalized by adding only two extra parameters per activation layer. The trade-off between both the models, demonstrates the [12] paper "…addition of batch-normalization added to state-of-art model performs better results." Additionally, to train around 144 million parameters, custom VGG-19 took around ~ 14 minute.

Resnet-34 gave an outstanding performance on data getting 98.19%. This reflects the skip connection level of the model's robustness and efficiency to maintain the gradient flow across all its 34 layers. However, the introduction of a dropout of 0.2 in the last fully connected layer slightly improved its performance by around 0.04%. This shows a very positive side of using the dropout regularization technique. By only adding it in the last fully connected layer, it fine-tuned the model by removing some features before giving the final output. Despite having less number of parameters to train than VGG-19 this took ~17 min to train which is more than the previous model and shows the depth complexity.

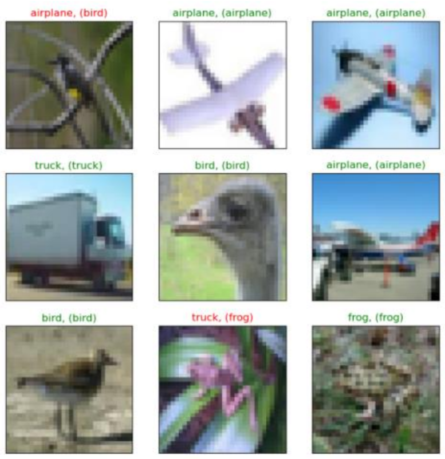
DenseNet-121 achieved an accuracy of 98.96%, which is nearly equal to both the model's performance. The dense architecture of the model mitigates the vanishing gradients and gives a competitive performance. However, the training time is what can be seen as a disadvantage, it took around 48 minutes to run for 15 epochs which is way much greater than the other two.

However, if we take a look at Figures 7 and 8, it can be seen that in the first few epochs VGG-19's accuracy is too low making the model gradually learn and then it improves. This has not been the case with the other two due to its robustness.

B. *Analysis of CIFAR-10 results*

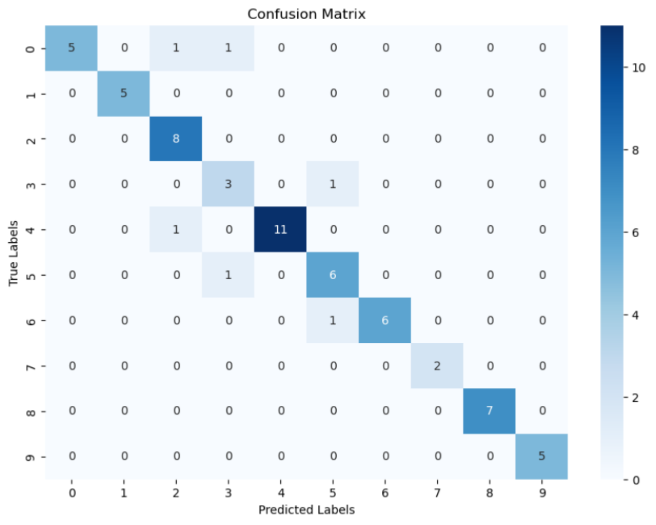
Despite the poor performance of the original VGG-19 on simpler datasets such as MNIST, it was a not good idea to incorporate with CIFAR-10. Though VGG-19 with enhanced features shows a very good result of 87.54% accuracy but less than what we got at MNIST because of the complex dataset. It has a very good architecture of a stack of convolutional layers enhanced with batch normalization, but for the diverse and complex datasets, there is room for improvement to prevent overfitting. With 50 epochs, it took around 37 minutes to train it. Figure 9. Shows some incorrectly predicted output from the model and Appendix shows the confusion matrix after first batch input result.

Figure 9. Output from VGG-19



Resnet-34 with dropout layer achieved an accuracy of 91.18% slightly greater than the original model-90.01%. This shows the model's adaptability even on diverse and complex data. Figure 6 shows that there was a major fluctuation in accuracy over epochs which can be minimized further on by fine tuning the model and incorporating other techniques to improve it. The training time took around 54 minutes given with added complexity of layers and dropout.

Figure 10. Confusion Matrix for one batch



Despite having a dense connection, DenseNet121 performance is somewhat near the VGG-19 model achieving a similar accuracy level. This may be a result of feature redundancy while implementing the complex datasets due to their dense layers. Incorporation of some regularization techniques or fine-tuning the model with variations of growth rate can help it to get better results. The model was trained with 40 epochs running over the highest time of ~90 minutes reflecting its time intensively.

Though all three models perform excellently on both datasets, ResNet-34 with dropout excels every time, telling about the blend of the model's architectural complexity and efficiency.

VI. CONCLUSION

The comparative analysis of the models ResNet34, VGG-19, and DenseNet-121 on the MNIST and CIFAR datasets highlights the performance of each model in image classification tasks. All three models exhibit very good accuracy but ResNet-34 with the enhanced feature of dropouts outperforms both models. These accuracies obtained outline the effectiveness of all models for performing simpler tasks in simple to complex datasets.

The results of this study further give some insights into each model's architectural designs and its strengths. Although DenseNet-121 has a deeper layer, the robustness and adaptability of ResNet-34 models make it easier to train. While VGG-19 can still be beneficial for simpler MNIST datasets.

However, despite notable evolution in the deeper networks model, it still is a topic of research and discussion. The scalability and efficiency of deeper network models, their interpretation, and the tuning of the model still need further refinement. Furthermore, transfer learning can be incorporated while training to make all the models more accurate.

VII. REFRENCES

[1] K.He, X. Zhang, S. Ren, J. Sun, "Deep Residual Learning for Image Recognition"

[2] K. Simonyan, A. Zisserman, "VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION", (ICLR 2015)

[3] G. Huang, Z. Liu, L.van der Maaten, "Densely Connected Convolutional Networks", (Jan-2018)

[4] N. Darapaneni, K. B, A. Reddy Pduri, "Convolutional Neural Networks: A comprehensive Study for Image Classification" (ICIIS,202)

[5] Z. Talal Abbood1 , M. Nasser Hussain Al-Turfi , L. kamil Adday Almajmaie, "An abbreviated review of deep learning-based image classification models" (Dec, 2022)

[6] Jingyuan Bai "Research and application of deep learning based on transfer learning in image classification tasks", (IEEE, 2024)

[7] S.H. Shabbeer Bash , Mohammad Farazuddin, Viswanath Pulabaigari, Shiv Ram Dubey, Snehasis Mukherjee," Deep Model Compression based on the Training History"

[8] Dongliang Li1 · Youyou Li1 · Zhigang Zhang1, "Analysis of convolutional neural networks-based

approaches in fruit disease detection for smart agriculture applications”

[9] Shubhi Shukla, Manaar Alam, Pabitra Mitra & Debdeep Mukhopadhyay, “STEALING THE INVISIBLE: UNVEILING PRE-TRAINED CNN MODELS THROUGH ADVERSARIAL EXAMPLES AND TIMING SIDE-CHANNELS”

[10] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky , Ilya Sutskever, Ruslan Salakhutdinov , “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”

[11] Sergey Ioffe , Christian Szegedy , “Batch Normalization: Accelerating Deep Network Training b y Reducing Internal Covariate Shift” – (March, 2015)

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification” – (Feb, 2015)

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Advances in Neural Information Processing Systems, pp. 1097-1105, 2012.

[14] Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. (1998)., “Gradient based learning applied to document recognition” p*roceedings of the IEEE*. **86** (11)

[15] Jain, Saachi, et al. "A Data-Based Perspective on Transfer Learning." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023.

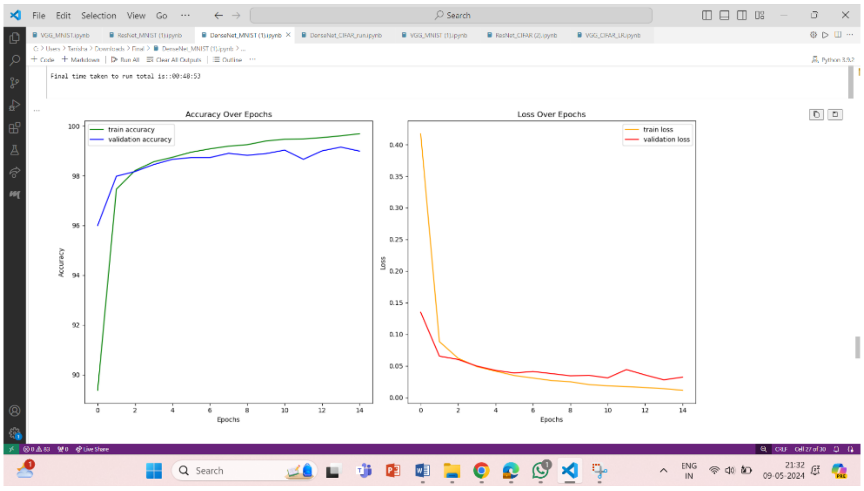
[16] Gim Jinsu, Huaguang Yang, and Lih-Sheng Turng. "Transfer learning of machine learning models for multi-objective process optimization of a transferred mold to ensure efficient and robust injection molding of high surface quality parts." Journal of Manufacturing Processes, 87, (2023): 11-24

[17] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks

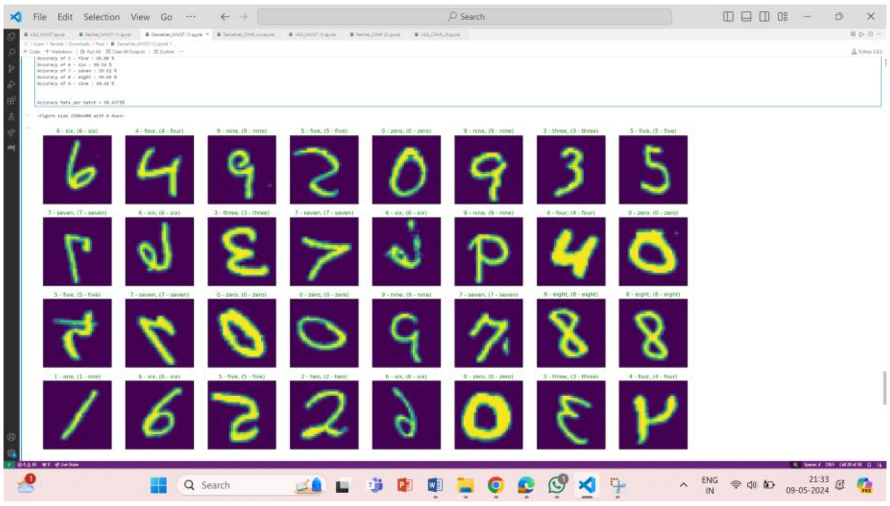
[18] Y. LeCun, http://yann. lecun. com/exdb/mnist/ (1998)

Appendix

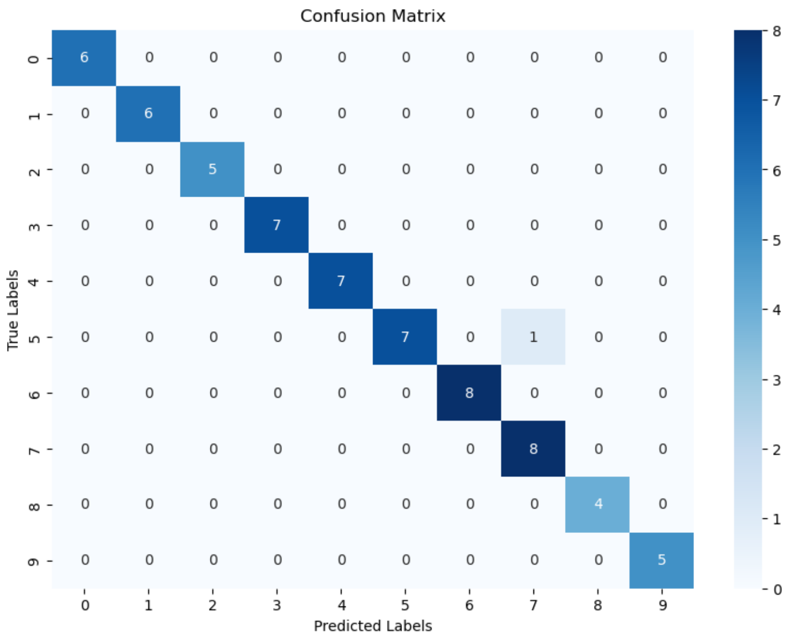
1. DenseNet-121 MNIST - Accuracy and Loss



```
Final time taken to run total is::00:48:53
```



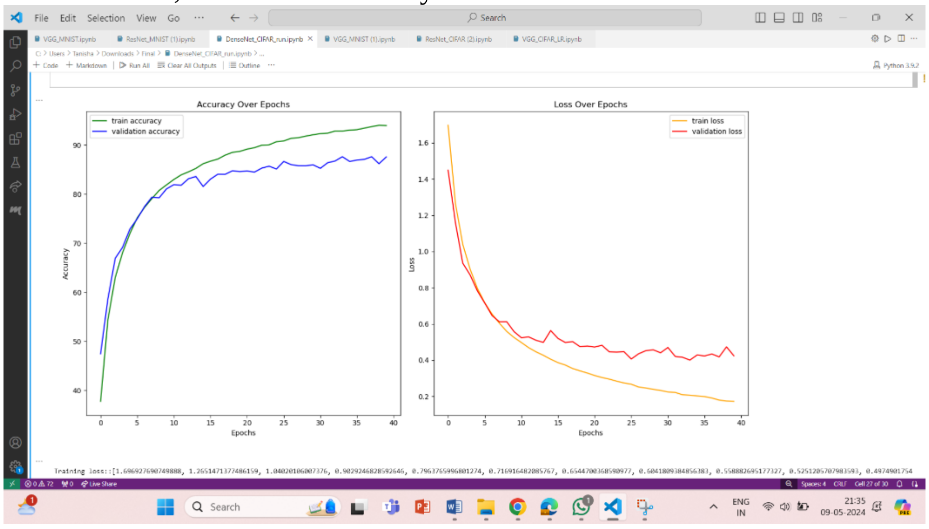
```
Accuracy of 5 - five : 98.88 %
Accuracy of 6 - six : 98.54 %
Accuracy of 7 - seven : 99.42 %
Accuracy of 8 - eight : 98.49 %
Accuracy of 9 - nine : 98.41 %


Accuracy Rate per batch = 98.4375%
<Figure size 2000x400 with 0 Axes>
```



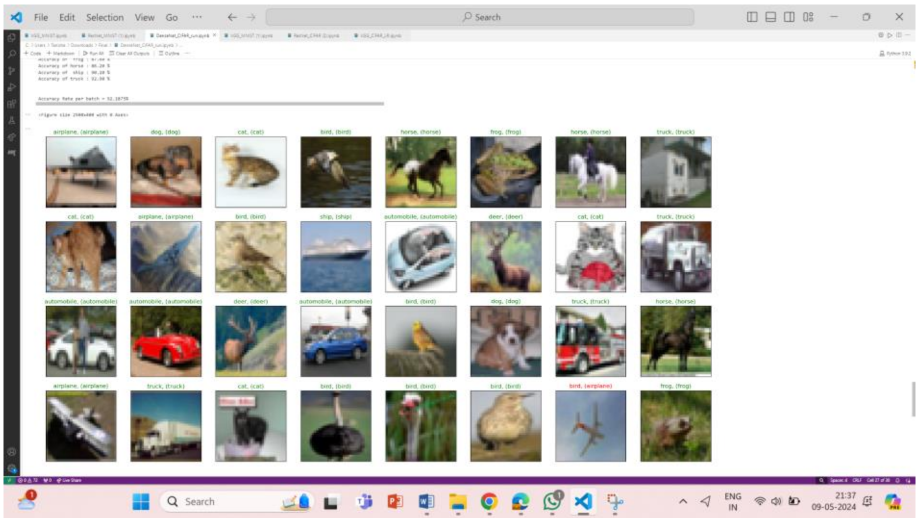
Confusion Matrix

True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	6	0	0	0	0	0	0	0	0	0
1	0	6	0	0	0	0	0	0	0	0
2	0	0	5	0	0	0	0	0	0	0
3	0	0	0	7	0	0	0	0	0	0
4	0	0	0	0	7	0	0	0	0	0
5	0	0	0	0	0	7	0	1	0	0
6	0	0	0	0	0	0	8	0	0	0
7	0	0	0	0	0	0	0	8	0	0
8	0	0	0	0	0	0	0	0	4	0
9	0	0	0	0	0	0	0	0	0	5

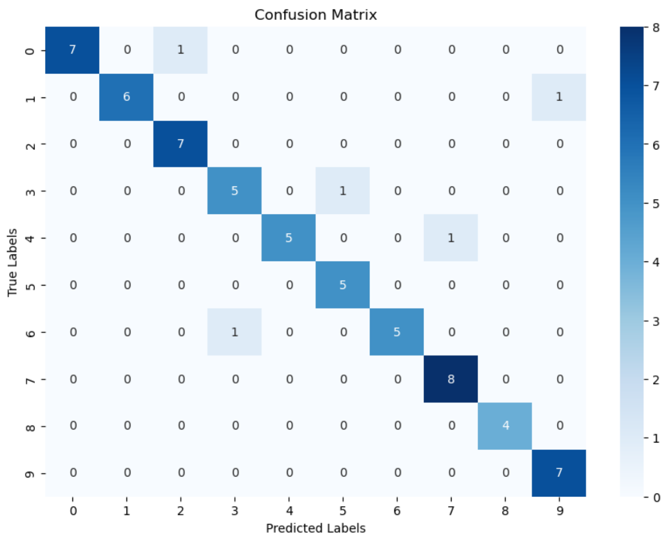
2. DenseNet-121, CIFAR-10 Accuracy vs Loss



```
Training loss::[1.696927690749888, 1.2651471377486159, 1.04020106007376, 0.9029246828592646, 0.7963765996801274, 0.716916482085767, 0.6544700368590977, 0.6041809384856383, 0.558882695177327, 0.5251205707983593, 0.4974901754
```

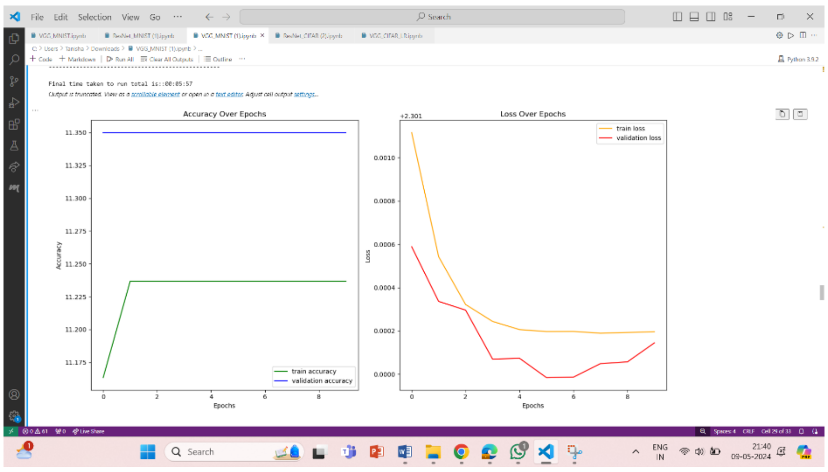


```
Accuracy of  frog : 87.00 %
Accuracy of horse : 86.20 %
Accuracy of  ship : 90.20 %
Accuracy of truck : 92.90 %


Accuracy Rate per batch = 32.1875%

<Figure size 2500x400 with 0 Axes>
```



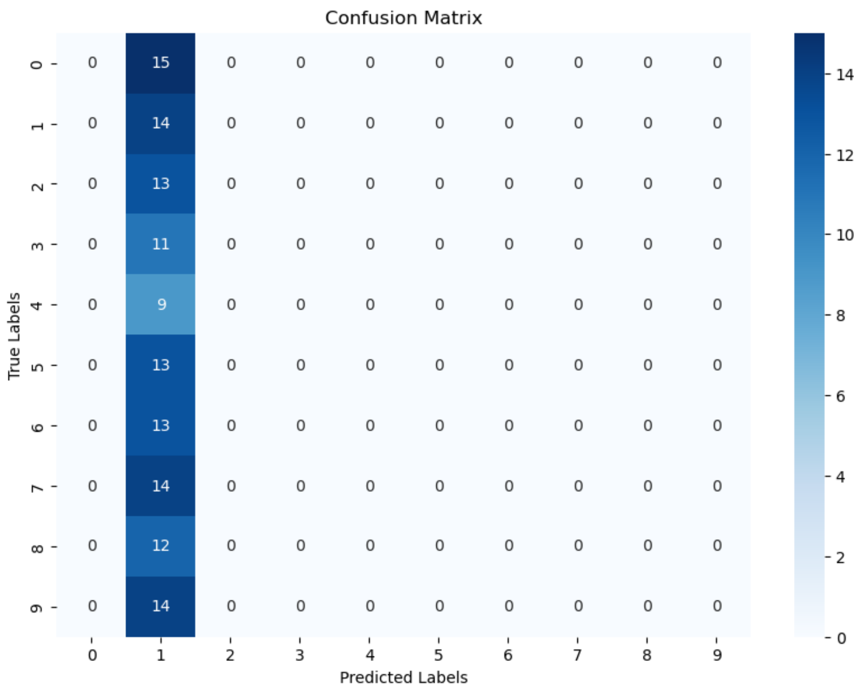
	0	1	2	3	4	5	6	7	8	9
0	7	0	1	0	0	0	0	0	0	0
1	0	6	0	0	0	0	0	0	0	1
2	0	0	7	0	0	0	0	0	0	0
3	0	0	0	5	0	1	0	0	0	0
4	0	0	0	0	5	0	0	1	0	0
5	0	0	0	0	0	5	0	0	0	0
6	0	0	0	1	0	0	5	0	0	0
7	0	0	0	0	0	0	0	8	0	0
8	0	0	0	0	0	0	0	0	4	0
9	0	0	0	0	0	0	0	0	0	7

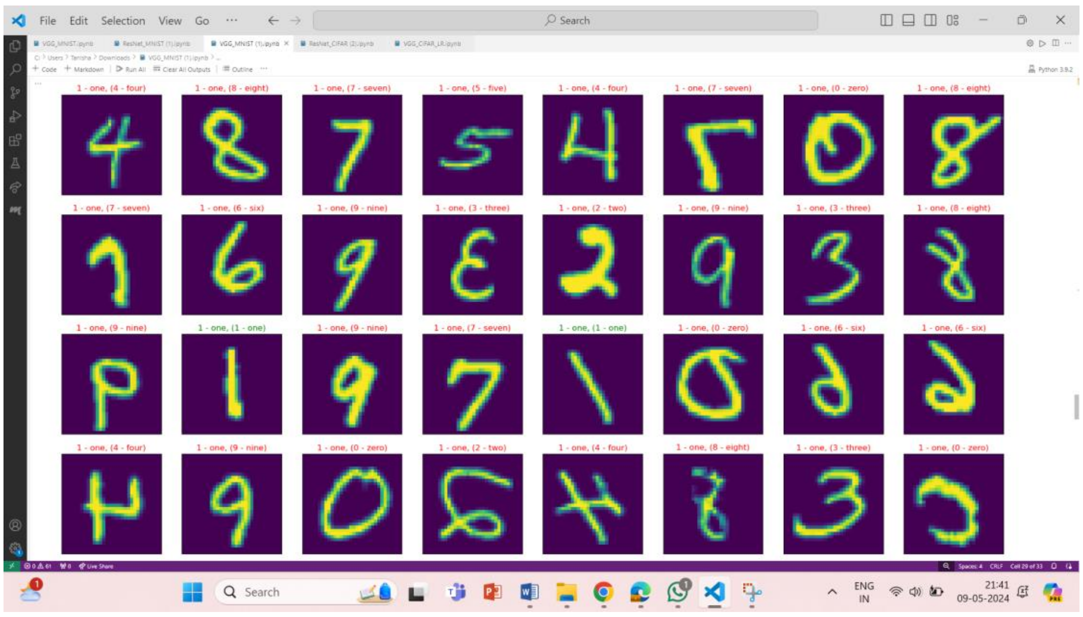
3. VGG-19 MNIST – Original



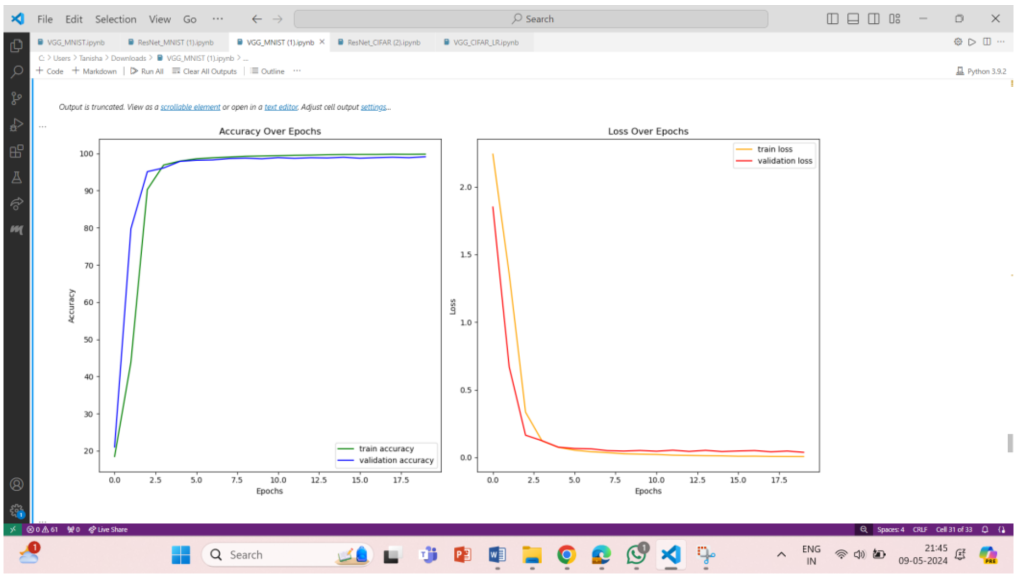
```
---------------------------------------------------

Final time taken to run total is::00:05:57
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

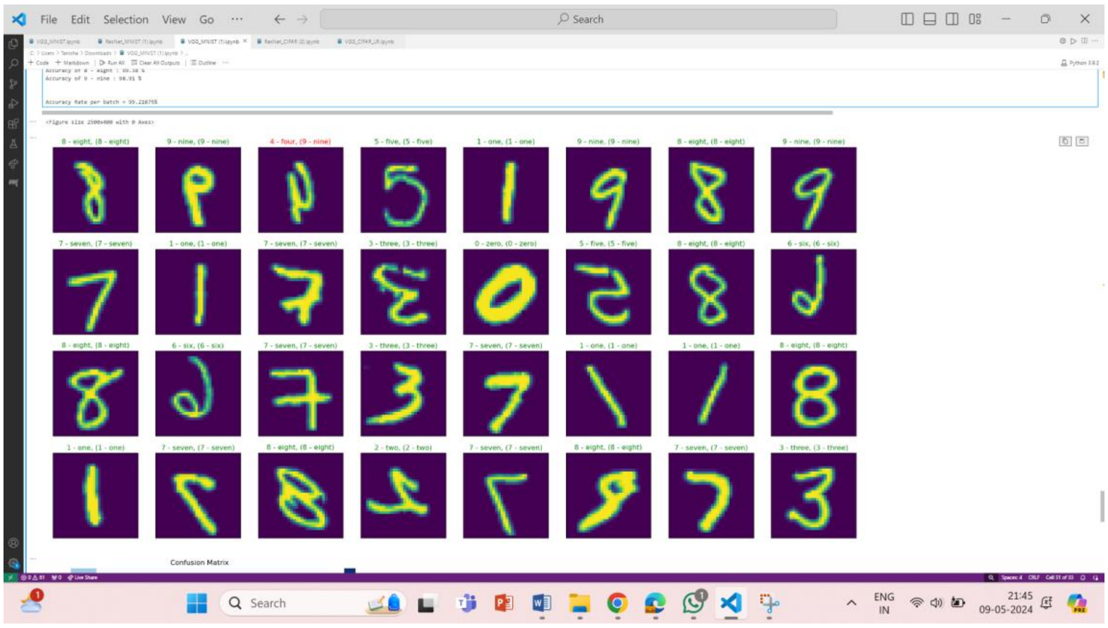




4. VGG-19 MNIST – Modified

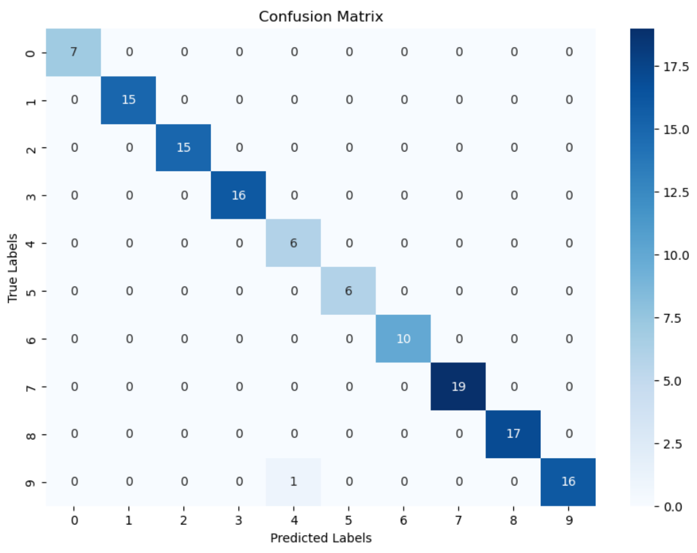


Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...



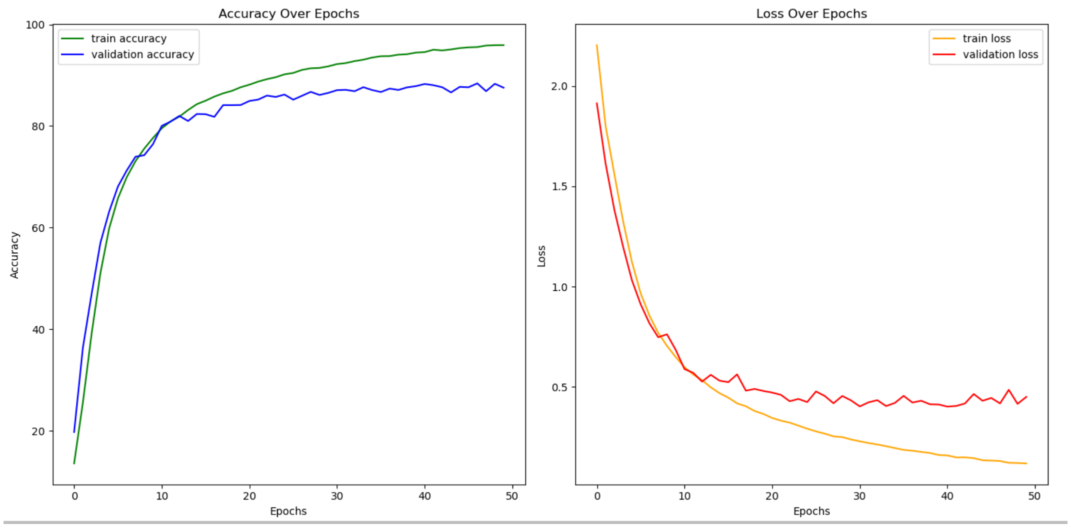
```
Accuracy of 8 - eight : 99.38 %
Accuracy of 9 - nine : 98.91 %


Accuracy Rate per batch = 99.21875%
```

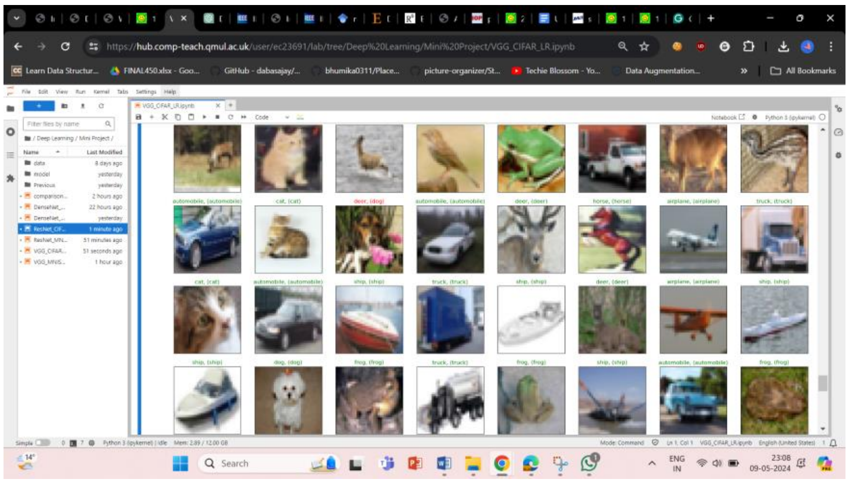
```
<Figure size 2500x400 with 0 Axes>
```

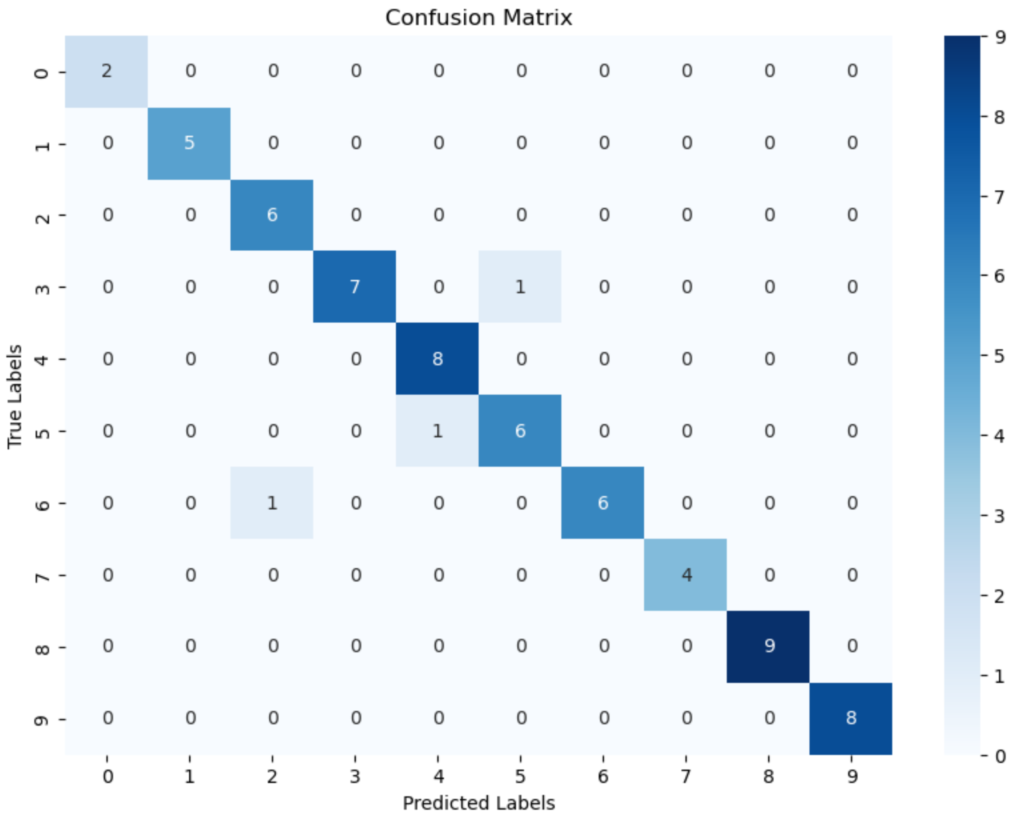


	0	1	2	3	4	5	6	7	8	9
0	7	0	0	0	0	0	0	0	0	0
1	0	15	0	0	0	0	0	0	0	0
2	0	0	15	0	0	0	0	0	0	0
3	0	0	0	16	0	0	0	0	0	0
4	0	0	0	0	6	0	0	0	0	0
5	0	0	0	0	0	6	0	0	0	0
6	0	0	0	0	0	0	10	0	0	0
7	0	0	0	0	0	0	0	19	0	0
8	0	0	0	0	0	0	0	0	17	0
9	0	0	0	0	1	0	0	0	0	16

5. VGG -19 CIFAR

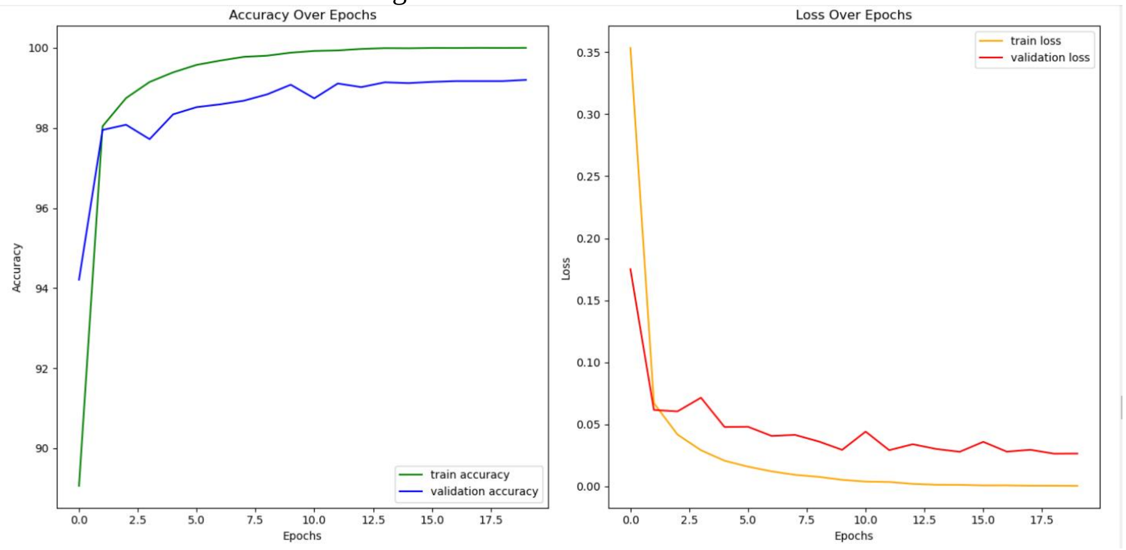


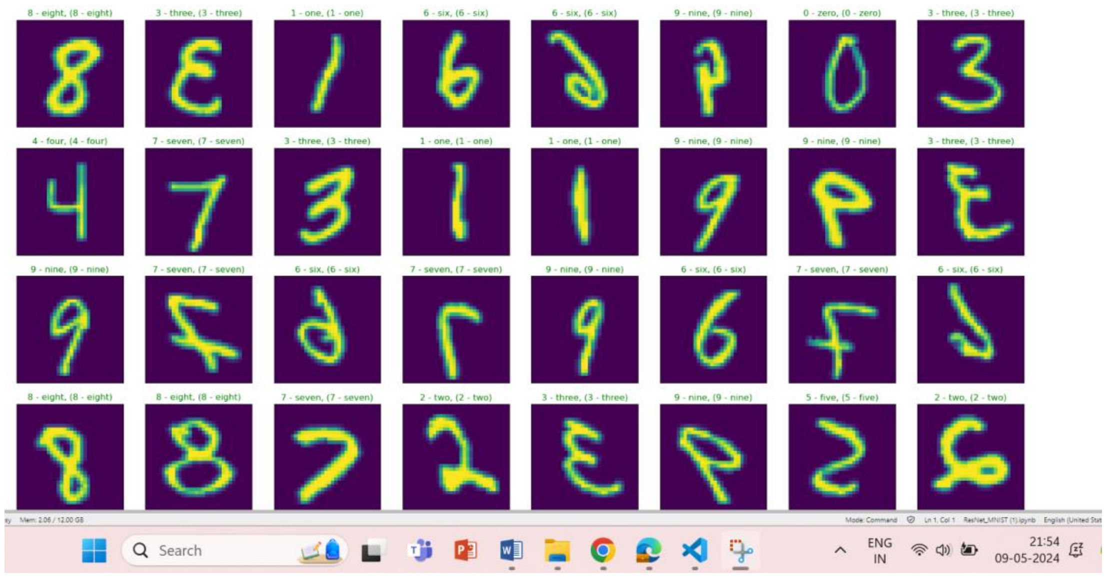


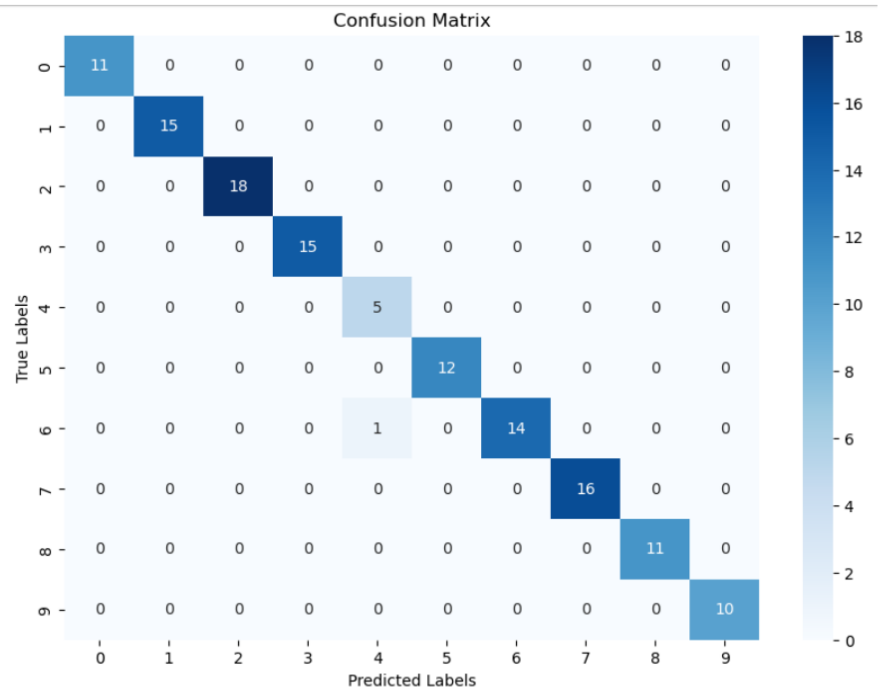


	0	1	2	3	4	5	6	7	8	9
0	2	0	0	0	0	0	0	0	0	0
1	0	5	0	0	0	0	0	0	0	0
2	0	0	6	0	0	0	0	0	0	0
3	0	0	0	7	0	1	0	0	0	0
4	0	0	0	0	8	0	0	0	0	0
5	0	0	0	0	1	6	0	0	0	0
6	0	0	1	0	0	0	6	0	0	0
7	0	0	0	0	0	0	0	4	0	0
8	0	0	0	0	0	0	0	0	9	0
9	0	0	0	0	0	0	0	0	0	8

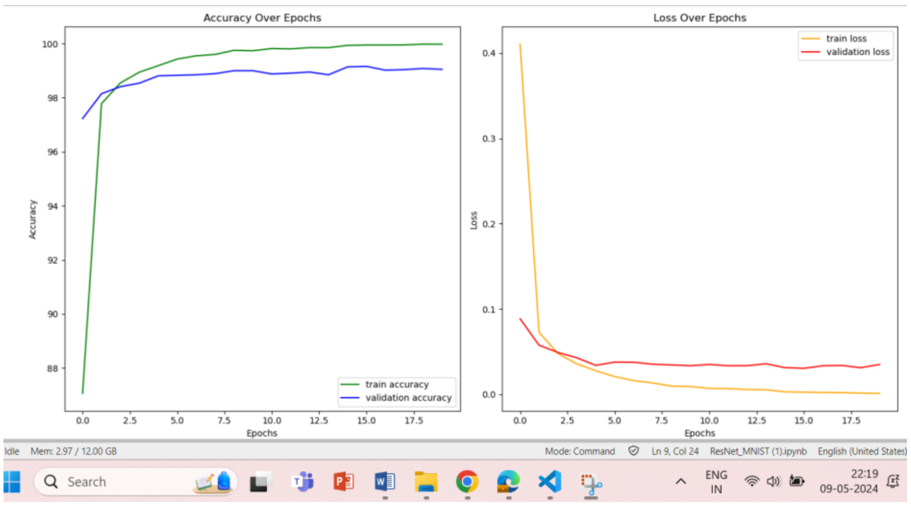
6. ResNet -34 MNIST – Original

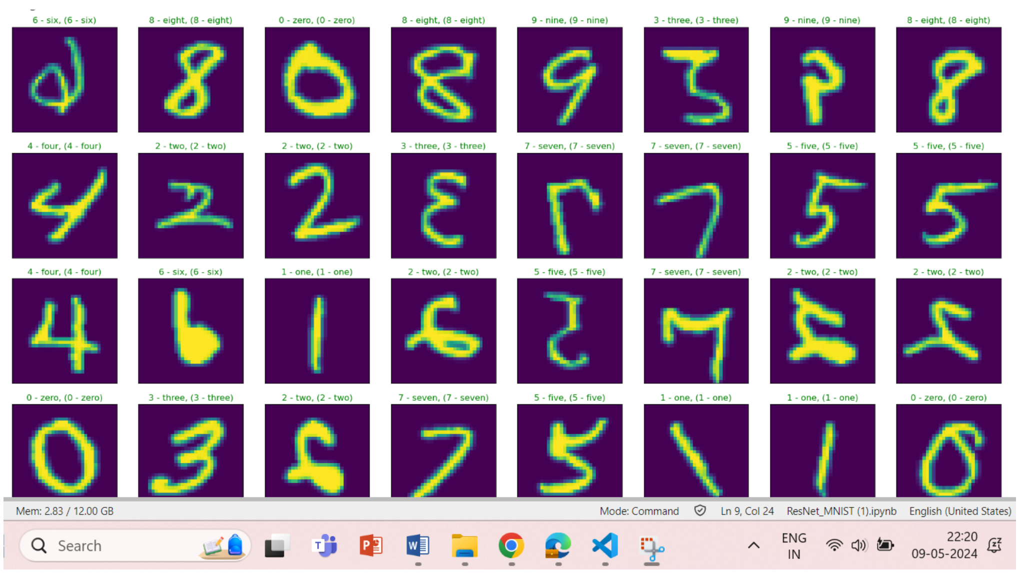


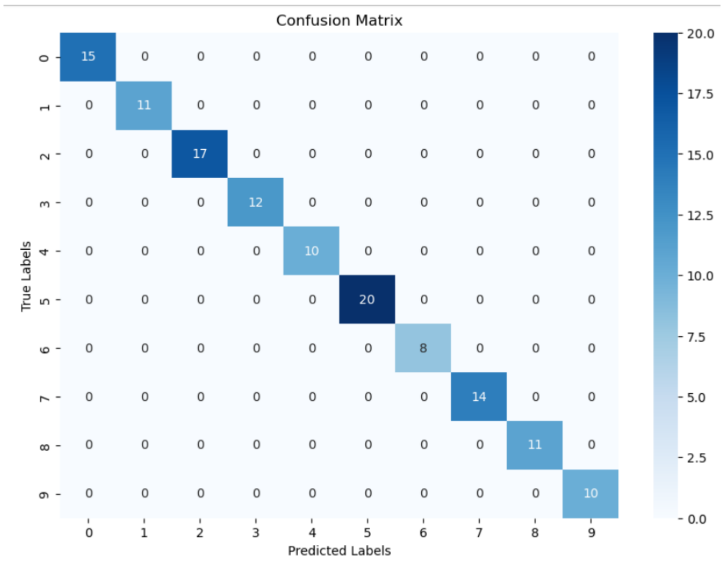




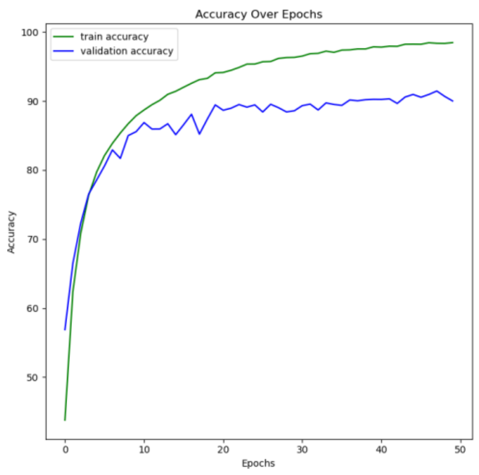
7. ResNet34- MNIST – Modified

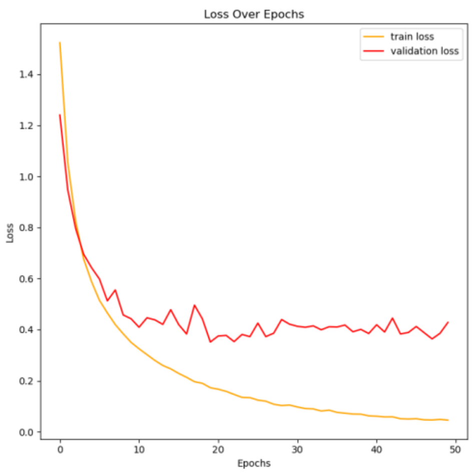


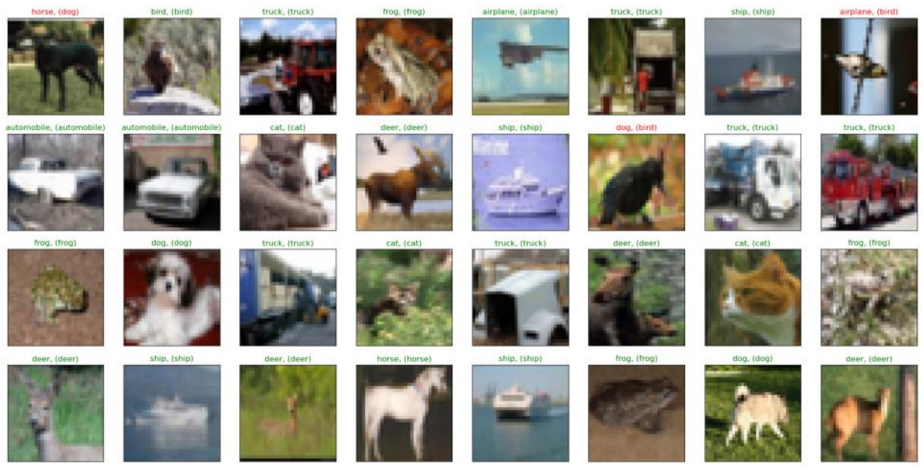


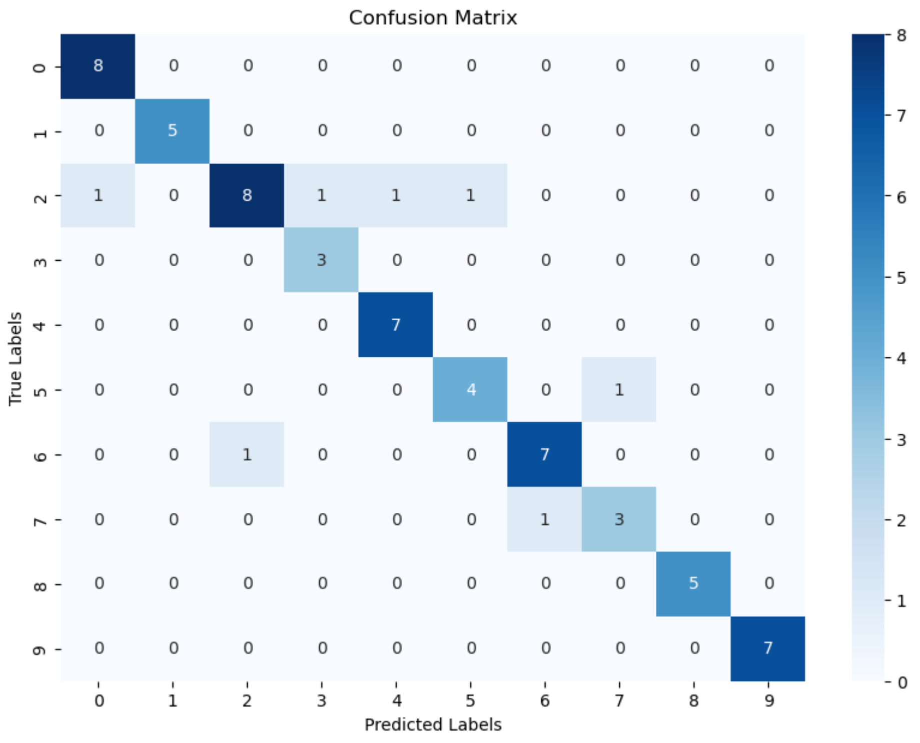


8. ResNet CIFAR Original

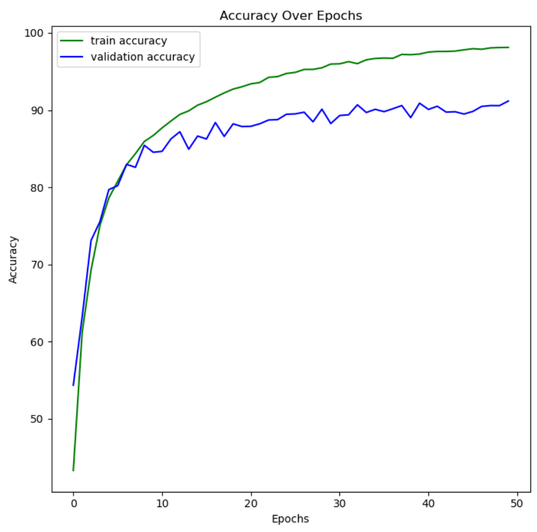


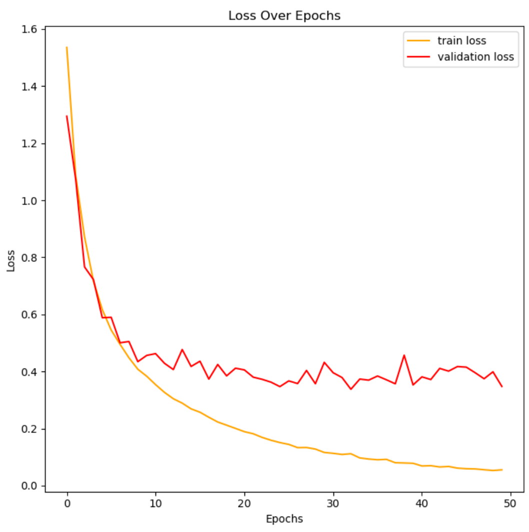


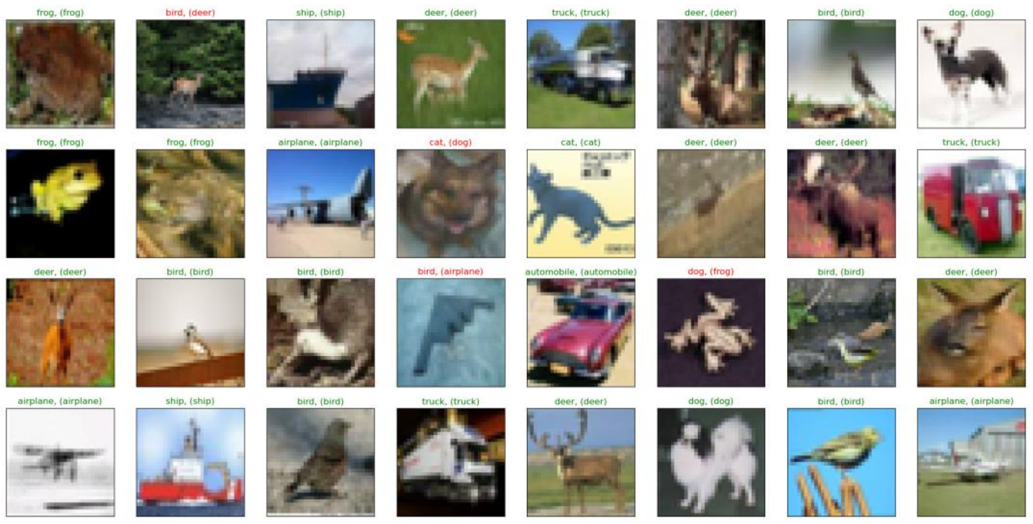


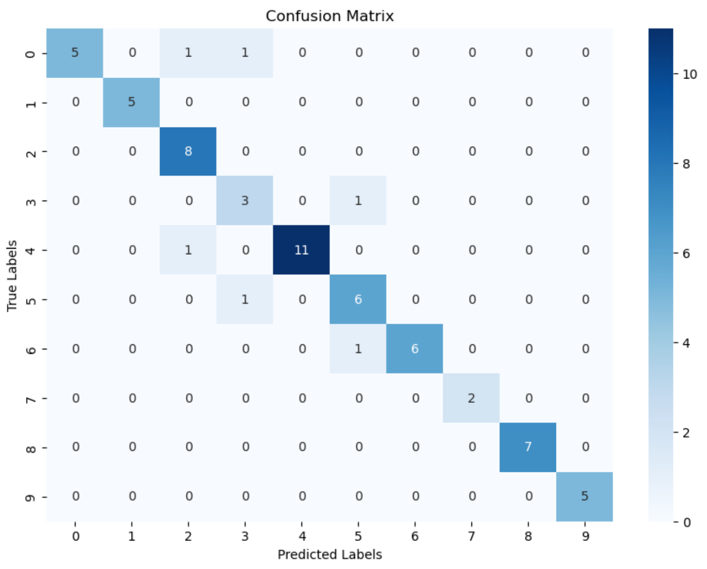


9. ResNet CIFAR -10 Modified


Accuracy Over Epochs
train accuracy
validation accuracy
Accuracy
Epochs


Loss Over Epochs
train loss
validation loss
Loss
Epochs




Confusion Matrix
True Labels
Predicted Labels
0: 5 0 1 1 0 0 0 0 0 0
1: 0 5 0 0 0 0 0 0 0 0
2: 0 0 8 0 0 0 0 0 0 0
3: 0 0 0 3 0 1 0 0 0 0
4: 0 0 1 0 11 0 0 0 0 0
5: 0 0 0 1 0 6 0 0 0 0
6: 0 0 0 0 0 1 6 0 0 0
7: 0 0 0 0 0 0 0 2 0 0
8: 0 0 0 0 0 0 0 0 7 0
9: 0 0 0 0 0 0 0 0 0 5