COMPARATIVE PERFORMANCE ANALYSIS OF NEURAL NETWORK ARCHITECTURES FOR SUPERVISED IMAGE CLASSIFICATION TASKS.

MSc. IN ARTIFICIAL INTELLIGENCE

2024/2025

LEELA PARYHAR AND NIDHI NARESH SONI

ABSTRACT

This project aims to perform Supervised Multi-class image classification on the CIFAR10 dataset, using different Deep Learning algorithms that are known to perform well on Image based tasks, namely, Convolutional Neural Network, Residual Networks, as well as the Densenet Architecture.

Different prepossessing techniques such as Data Augmentation and Normalization have also been applied in order to improve performance and robustness.

The project also includes a further comparison of the performance of the above mentioned algorithms in order to analyse and highlight the trade-offs between lightweight and deeper architectures, and their strengths and limitations on image classification tasks.

For this project, A baseline CNN achieved a good accuracy, while the addition of batch normalization and dropout improved performance of the model. A deeper ResNet-34 further increased accuracy compared to the models before. The models were trained with standard pre-processing and data augmentation, and their comparative analysis highlights the benefits of architectural depth and regularization techniques for small-scale image classification.

TABLE OF CONTENTS

SECTION 1
MOTIVATION AND RATIONALE

1.1: Motivation of the project

Image classification is a central area of research in Deep learning and forms the basis for tasks such as recognition, detection and tracking. Improved classification tasks also results in solving many real world problems, such as, tumor detection, object detection among others.

Using the CIFAR10 dataset is useful for many reasons, especially as it is a standardized, benchmarked, and of a good size, for analyzing the performance of different neural network architectures under controlled conditions.

This project is motivated by the goal of exploring how different neural networks perform on this dataset in order to figure out which model performs with the best performance.

In particular, the project aims to analyze how architectural depth, regularization, and preprocessing techniques affect performance and generalization.

1.2: Significance of the Project

The significance of the project lies in trying to solve the challenge of which is the most appropriate architecture to select for a specific use case while also keeping in mind, performance, and efficiency trade offs. This project also helps to understand and make sense of how different neural network architectures, both lightweight like the CNN, and deeper architectures like the Densenet, Resnet, perform under resource constrained environments and how it would affect feasibility.

This project also addresses the architectural systemic comparison of the different neural network architectures in the following way:

Initially, with the Baseline CNN, the project establishes fundamental convolution performance without enhancement. Afterwards, with the CNN with Regularization, there is quantification of the impact of batch normalization and dropout on basic architectures. Skip connections and architectural depth are examined with the Resnet architecture. Densenet and feature reuse capacity are examined with Densenet. Finally, Efficientnet is also used to represent modern compound scaling.

SECTION 2
STATE OF THE ART

2.1: State of the Art techniques used in the Project

One of the state of the Art used in this project is the Residual Networks (ResNet architecture). Resnets solve the problem of vanishing gradient through the use of skip connections. Some of the techniques exploited by this network include: Skip connections for gradient flow preservation. However, one of the weakness of Resnet lies in the fact that memory consumption scales linearly with depth.

At the moment, Densenet also stands as a State of the Art architecture as it achieved 3.46% top-5 error on ImageNet. It connected each layer to subsequent layer, and also reducing the problem of vanishing gradient. It exploits Dense connectivity patterns for maximum information flow, as well as transition layers for dimensionality reduction.

Furthermore, Regularization techniques have also been exploited in this project to highlight higher learning rate and faster convergence. One of the strong points of Regularization is that it prevents overfitting. However, it can sometimes interfere with Deep Networks.

SECTION 3
OBJECTIVES OF THE PROJECT

3.1 Main Objective of the Project

To perform supervised multi-class image classification on the CIFAR-10 dataset by implementing and evaluating multiple deep learning architectures, namely Convolutional Neural Networks (CNNs) in order to achieve high predictive accuracy, critically examine the influence of architectural depth, feature connectivity, and regularization techniques on performance, efficiency, and generalization in small-scale image classification tasks.

3.2 Specific Objectives of the Project

1. To understand the CIFAR-10 dataset in detail.
2. To apply normalization and augmentation techniques to improve robustness and reduce the risk of overfitting.
3. To design, train and implement multiple neural network architectures for image classification:

- ✓ A **baseline CNN** as the baseline model.
- ✓ A **variant CNN** enhanced with Batch Normalization and Dropout.

- ✓ **ResNet-18** as a lightweight residual network.
- ✓ **ResNet-34** as a deeper residual network for comparative analysis.
- ✓ **DenseNet** to investigate the effect of dense connectivity and feature reuse.

4. To apply optimization and regularization strategies such as adaptive learning rate scheduling, batch normalization, and dropout in order to stabilize training and improve generalization.
5. To evaluate all models using both quantitative metrics (accuracy, precision, recall, F1-score, and confusion matrices) and qualitative analysis of misclassified images.
6. To analyze and compare the performance of the models in terms of predictive accuracy, training stability.

SECTION 4
METHODOLOGY USED FOR THE PROJECT

4.1: Dataset selected for the project

The dataset chosen for this project is the CIFAR-10 benchmark dataset, one of the most widely used datasets for evaluating supervised image classification models.

It contains 60,000 color images of size 32 × 32 pixels, categorized into 10 mutually exclusive classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

The Training set is consist of 50,000 images (5,000 per class) and the Test set consists of 10,000 images (1,000 per class).

Key challenges of the dataset CIFAR-10 are:

Low image resolution (32 × 32 pixels) limits the amount of detail available for feature extraction.

High inter-class similarity, such as between cats and dogs or between airplanes and birds.

Intra-class variation due to changes in background, orientation, and lighting conditions

4.2: Methods implemented

The methodological process followed in this project involves several stages:

4.2.1: Libraries Import Strategy

The following libraries were imported for use in the project:

PyTorch

Provided flexibility for experimenting with complex architectures.

NumPy and Pandas

For numerical operations, matrix handling, and efficient manipulation of dataset batches.

Matplotlib and Seaborn

For generating plots such as training/validation accuracy curves, loss curves, and confusion matrices.

Scikit-learn

For advanced evaluation metrics, and construction of classification reports.

- Weights & Biases (WandB) for hyperparameter logging and experiment management

The following was carried out to manage data and partition it.

4.2.2: Dataset Loading Protocol

The CIFAR-10 dataset loading follows a systematic approach ensuring data integrity and consistency across all experiments:

Training-Validation-Test Split Strategy:

CIFAR10 comes pre-split into:

Training set (train=True) → 50,000 images

Test set (train=False) → 10,000 images

There's a balanced class distribution across all splits.

The random seed is fixed across all experiments for reproducible splits.

4.2.3: Data Preprocessing

Normalization: Pixel values were scaled to the range [0,1] to stabilize gradient descent and speed up convergence.

Data Augmentation: Random horizontal flips, rotations, translations, and brightness adjustments were applied to expand the effective dataset size and reduce overfitting.

4.2.4: Hyperparameter Specification

The following hyperparameters were kept fixed for a fair comparison.

Random Seed: 42
Batch Size: 64
Number of Epochs: 10
Base Learning Rate: 0.001
Loss Function: CrossEntropyLoss for multi-class classification
Evaluation Metrics: Accuracy, Precision, Recall, F1-Score
Input Dimensions: 32×32×3 (CIFAR-10 standard)

The following hyperparameters were varied to allow the models to achieve best performance:

- Weight decay
- Learning rate

Batching: Data was divided into mini-batches for efficient GPU training.

4.2.5: Baseline and Model Variants

A baseline CNN was first implemented to establish reference performance.

A variant CNN with Batch Normalization and Dropout was developed to test improvements through regularization.

ResNet-18 and ResNet-34 were implemented to evaluate the role of residual connections and model depth.

A DenseNet model was implemented to study feature reuse and dense connectivity.

4.2.6: Optimization Strategy

Models were trained using the Adam optimizer with learning rate scheduling.

Dropout and Batch Normalization were used to improve generalization.

4.3: Algorithms used in the project

The following architectures were implemented and compared:

Baseline CNN - A simple convolutional network was used as a benchmark.

CNN Variant (with Batch Normalization & Dropout) - Enhanced CNN that improves training stability and reduces overfitting.

ResNet-18 - A residual network with 18 layers and skip connections. Lightweight but more effective than plain CNNs.

ResNet-34 - A deeper residual network (34 layers) designed for stronger hierarchical feature learning at the cost of greater computation.

DenseNet - A dense connectivity model where each layer receives input from all preceding layers, encouraging feature reuse and efficient gradient flow.

4.4: Analytical and computational tools necessary for the project

Programming Environment

The project was developed in **Python**, especially due to the availability of specialized libraries for deep learning project.

Experiments were conducted in **Google Colab**, which provided an interactive environment for iterative model development, visualization, and debugging.

Pytorch

Used for implementing the **baseline CNN**, the **CNN variant (with Batch Normalization and Dropout)**, and **ResNet-18 and 34, densnet121 and efficient net**.

4.5 Analytical Methods

4.5.1 Quantitative Analysis:

Accuracy were computed for each model.

Confusion matrices were generated to analyze class-level performance and error distribution.

SECTION 5
EXPERIMENTS AND RESULTS

5.1: Evaluation Protocol and Conditions

The experiments were conducted to evaluate the performance of convolutional neural network (CNN) architectures for image classification. The setup was as follows as below:

Dataset: we have used CIFAR-10 dataset.

Data Split: The dataset was divided default into training, validation, and test sets. Training and validation were used to tune parameters, while final results were reported on the test set. The ratio is of 50,000 images and 10,000 images.

Implementation Details: Models were implemented in Pytorch. Training was performed for 10 epochs with cross-entropy loss and an adam optimizer was used.

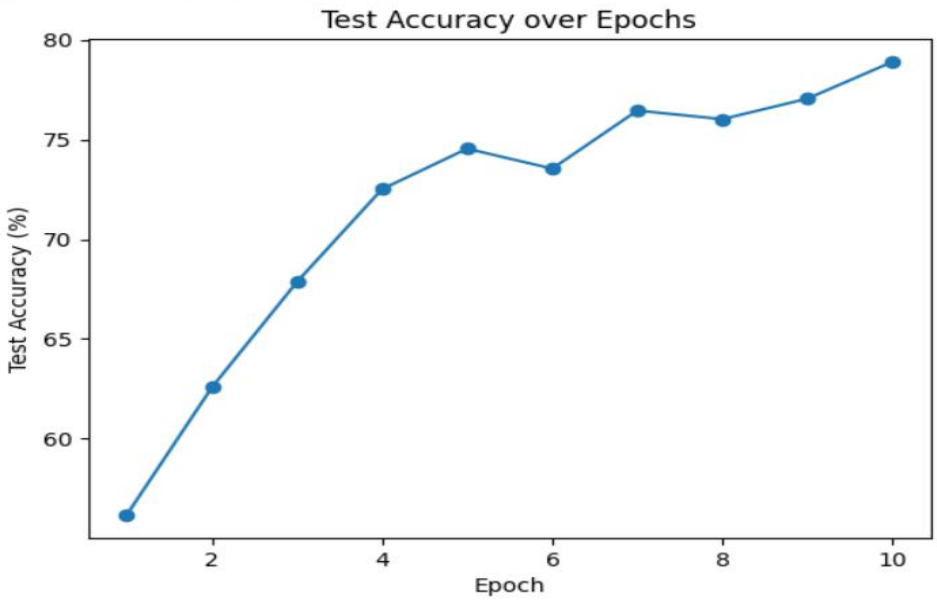
Hardware: The experiments were executed on GPU environment from the runtime.

This consistent protocol ensured fair comparison across baseline and improved models.

5.2: Metrics to measure Performance

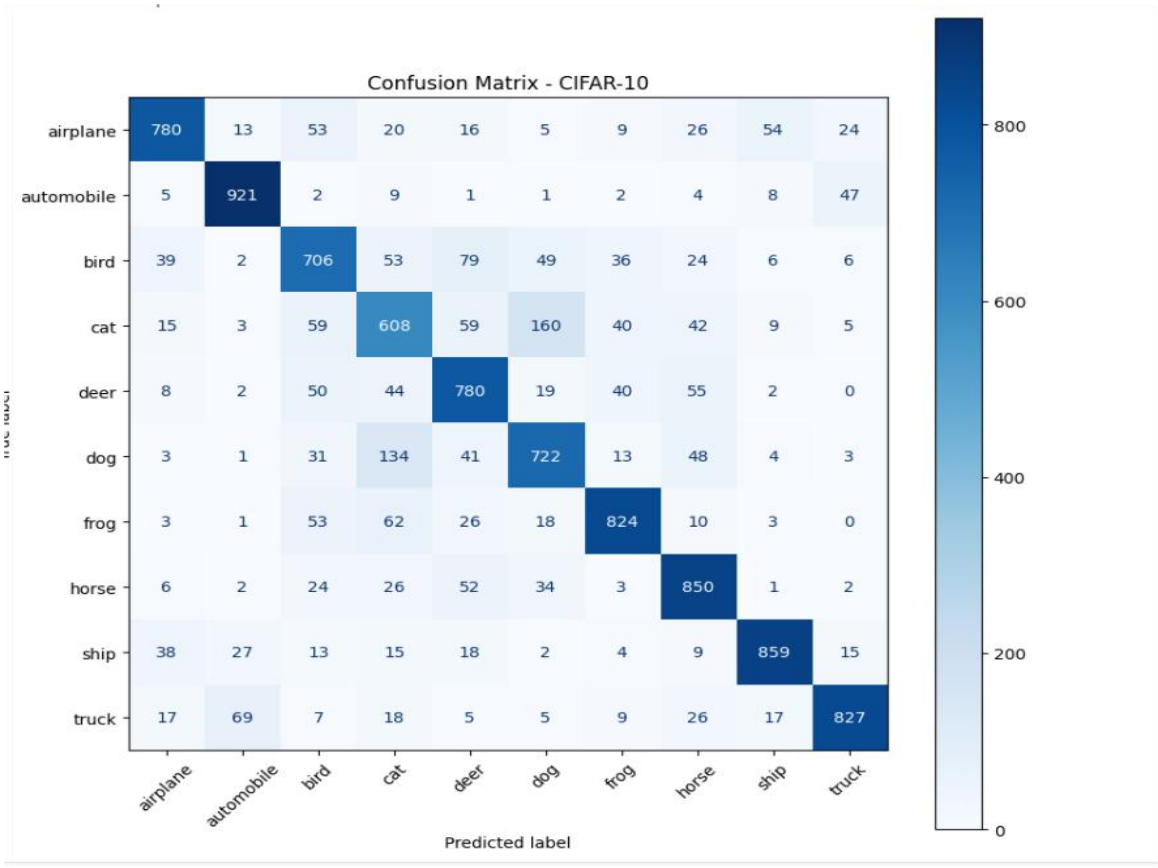
Performance of base link cnn

Test accuracy



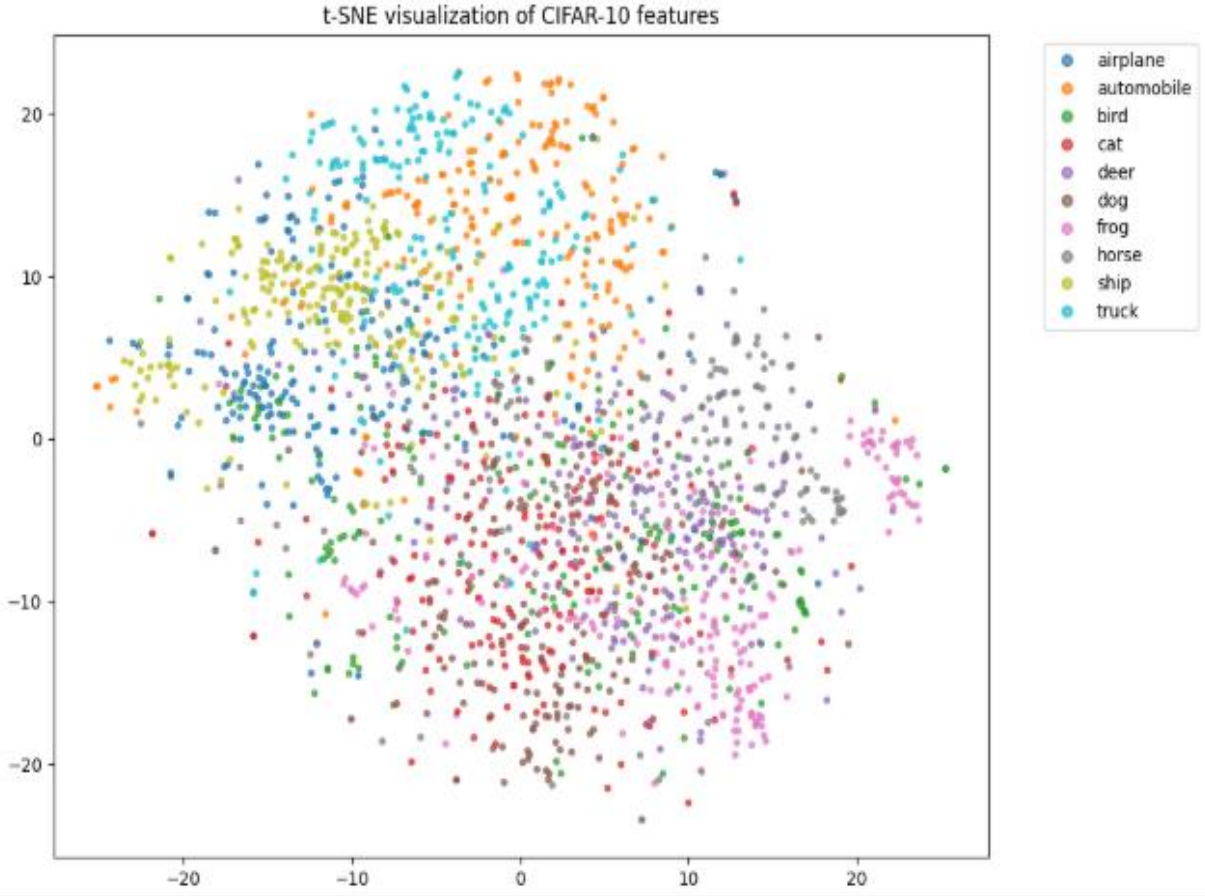

```
Final Test Accuracy: 78.90%
```

Confusion



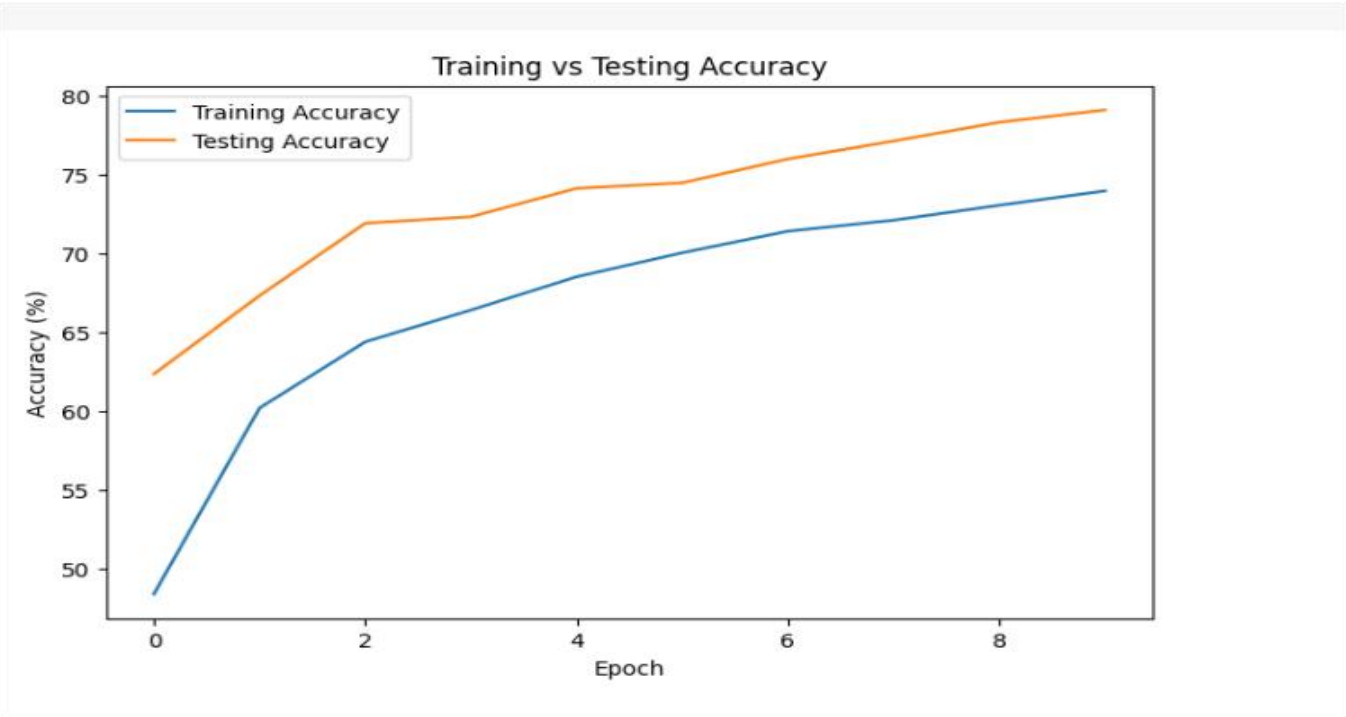
matrix

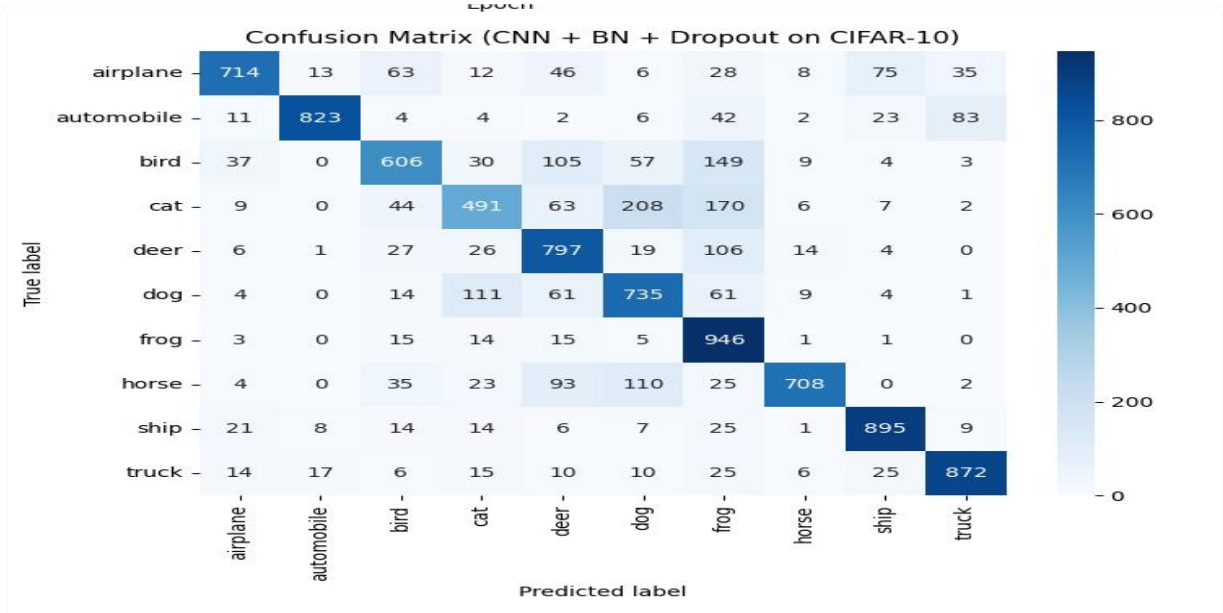
```
              precision    recall  f1-score   support

    airplane       0.83      0.75      0.79      1000
  automobile       0.82      0.94      0.87      1000
        bird       0.69      0.68      0.68      1000
         cat       0.60      0.68      0.64      1000
        deer       0.79      0.75      0.77      1000
         dog       0.75      0.64      0.69      1000
        frog       0.84      0.81      0.83      1000
       horse       0.85      0.81      0.83      1000
        ship       0.79      0.93      0.85      1000
       truck       0.89      0.80      0.85      1000

    accuracy                           0.78     10000
   macro avg       0.78      0.78      0.78     10000
weighted avg       0.78      0.78      0.78     10000

Generating t-SNE visualization...
Running t-SNE... this may take a minute.
```



Cnn with batch normalization and dropout.

Test accuracy



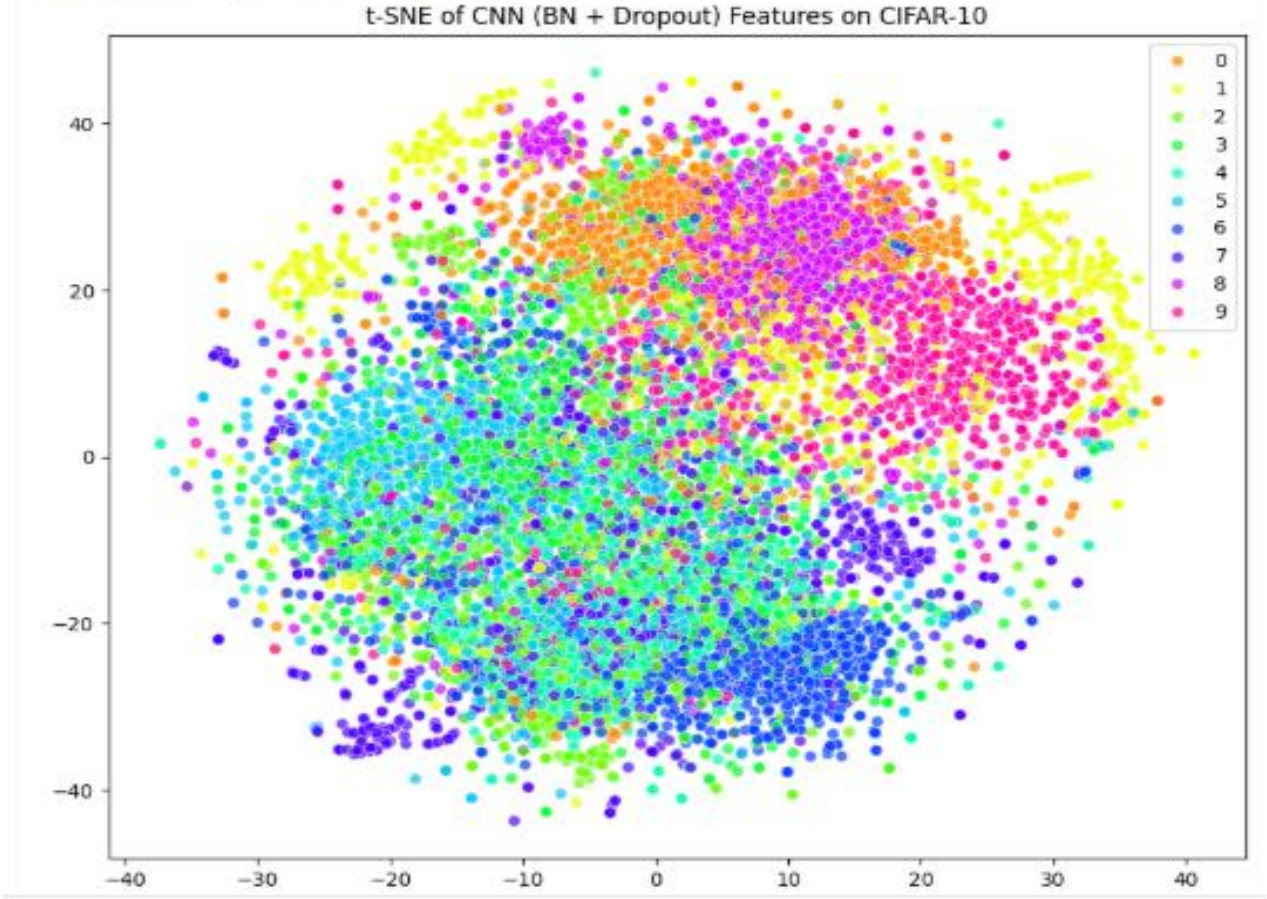


Confusion ma

Classification Report (CIFAR-10):

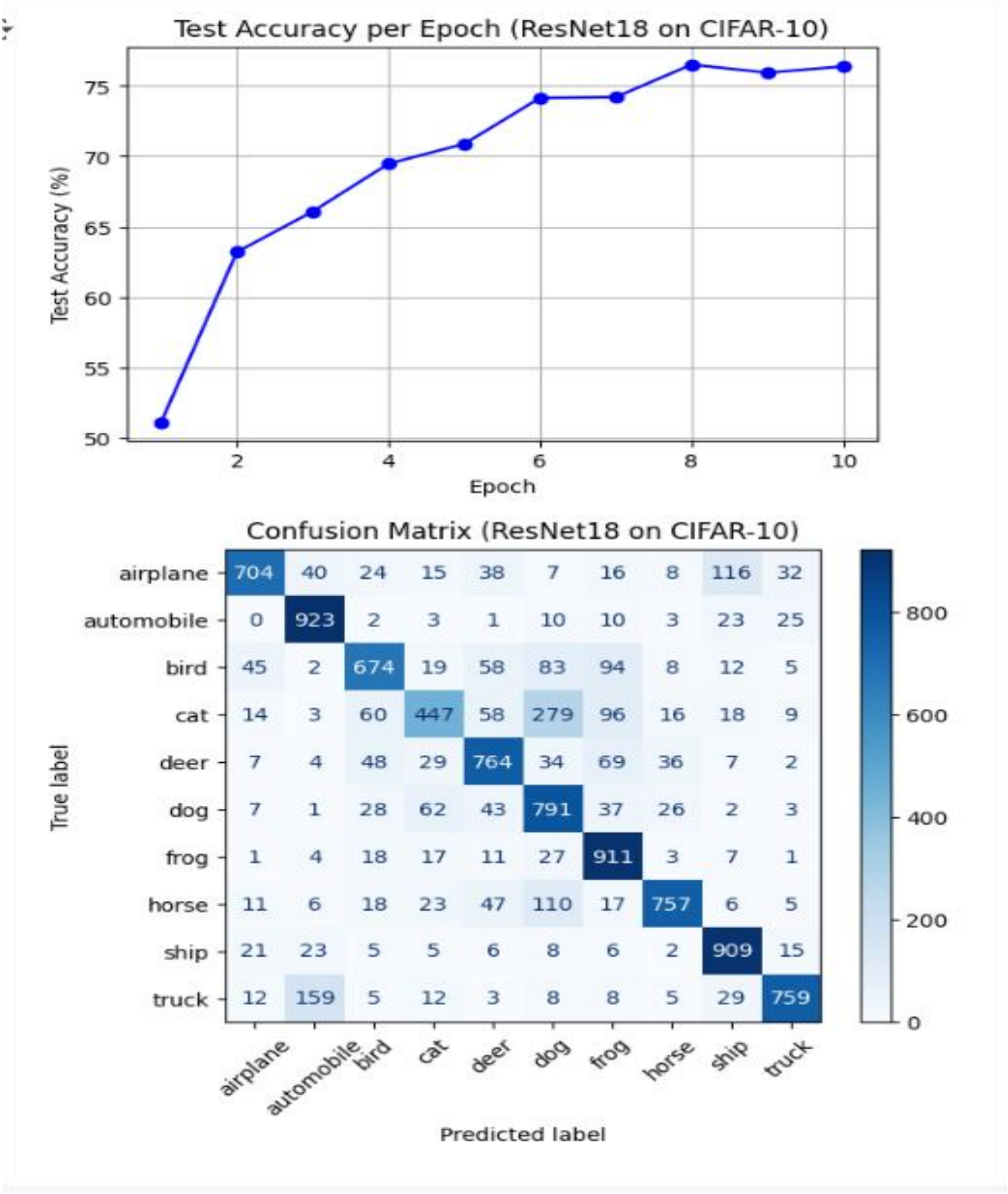
	precision	recall	f1-score	support
airplane	0.84	0.76	0.80	1000
automobile	0.94	0.85	0.89	1000
bird	0.64	0.71	0.67	1000
cat	0.65	0.57	0.61	1000
deer	0.71	0.81	0.76	1000
dog	0.69	0.68	0.68	1000
frog	0.72	0.92	0.81	1000
horse	0.91	0.74	0.82	1000
ship	0.85	0.91	0.87	1000
truck	0.91	0.84	0.87	1000
accuracy			0.78	10000
macro avg	0.79	0.78	0.78	10000
weighted avg	0.79	0.78	0.78	10000

Running t-SNE... (can take several minutes)



Resnet 18

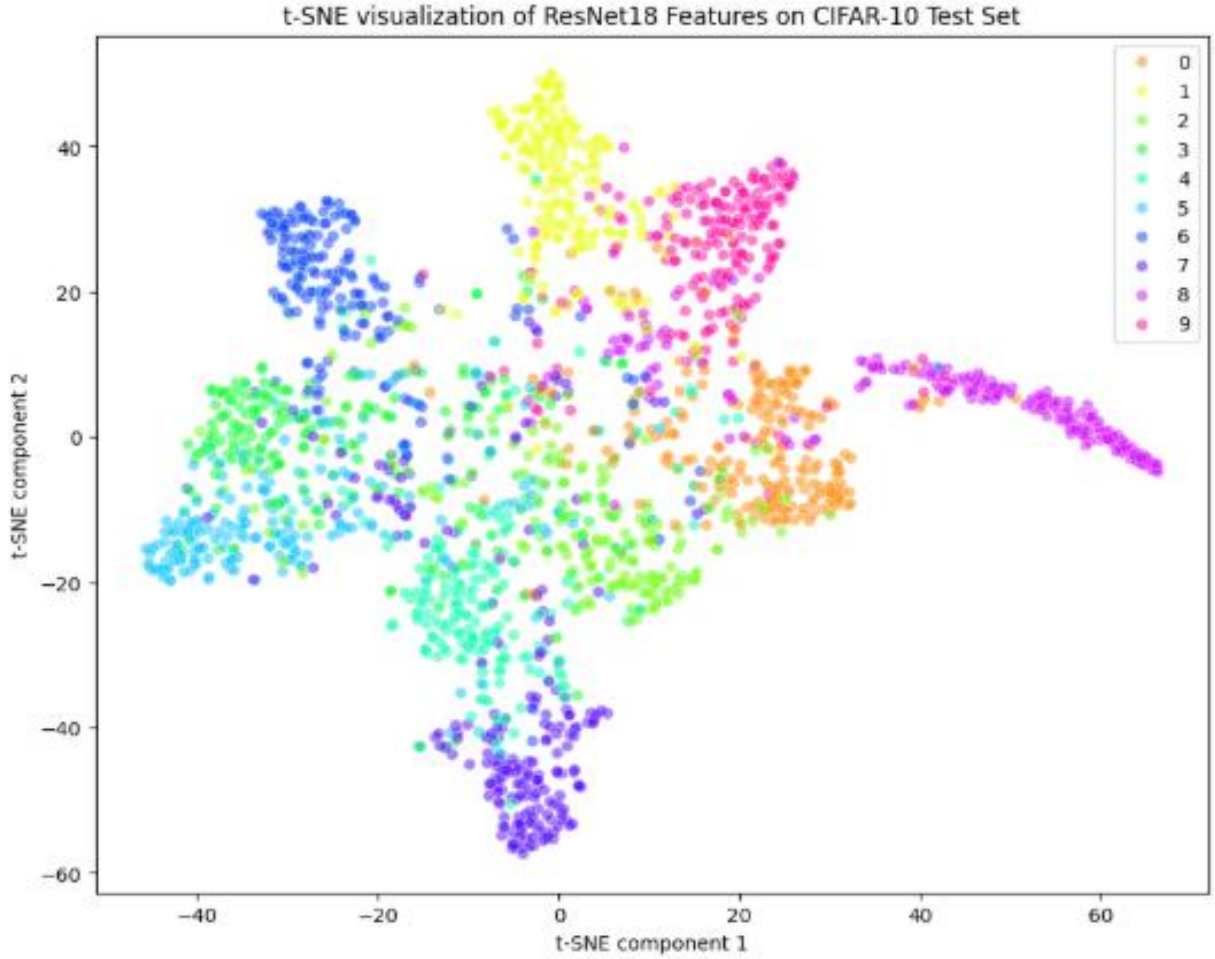
Test accuracy and confusion matrix



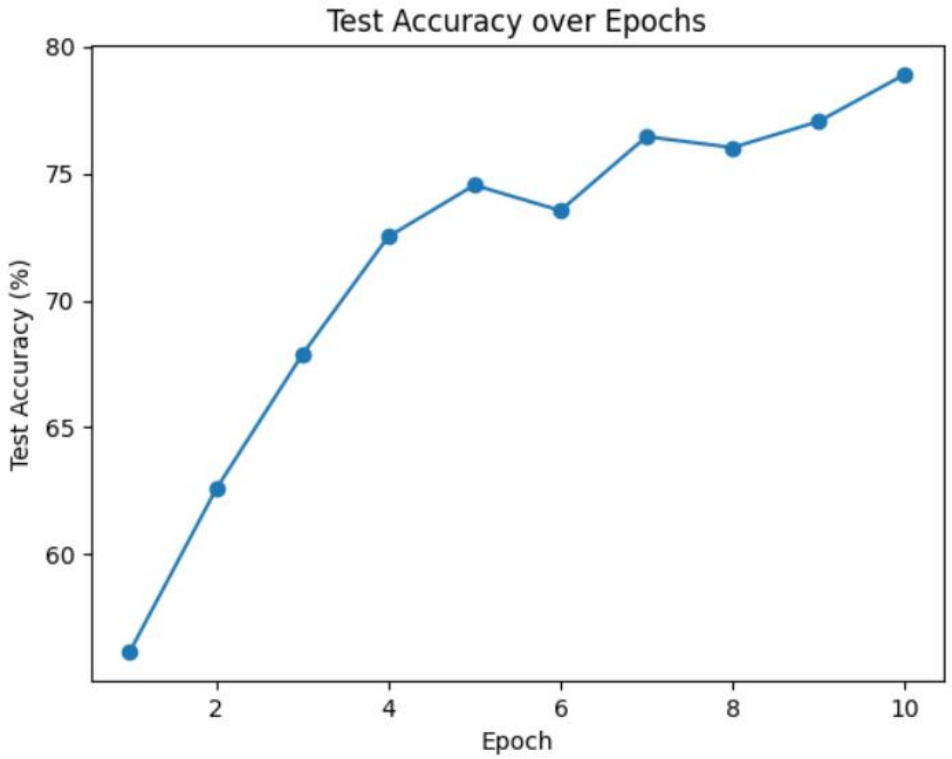
```
plt.show()
```

```
              precision    recall  f1-score   support

    airplane       0.65      0.90      0.76      1000
  automobile       0.86      0.88      0.87      1000
        bird       0.66      0.71      0.69      1000
         cat       0.67      0.55      0.60      1000
        deer       0.86      0.62      0.72      1000
         dog       0.69      0.71      0.70      1000
        frog       0.85      0.85      0.85      1000
       horse       0.82      0.79      0.81      1000
        ship       0.90      0.83      0.86      1000
       truck       0.82      0.88      0.85      1000

    accuracy                           0.77     10000
   macro avg       0.78      0.77      0.77     10000
weighted avg       0.78      0.77      0.77     10000
```

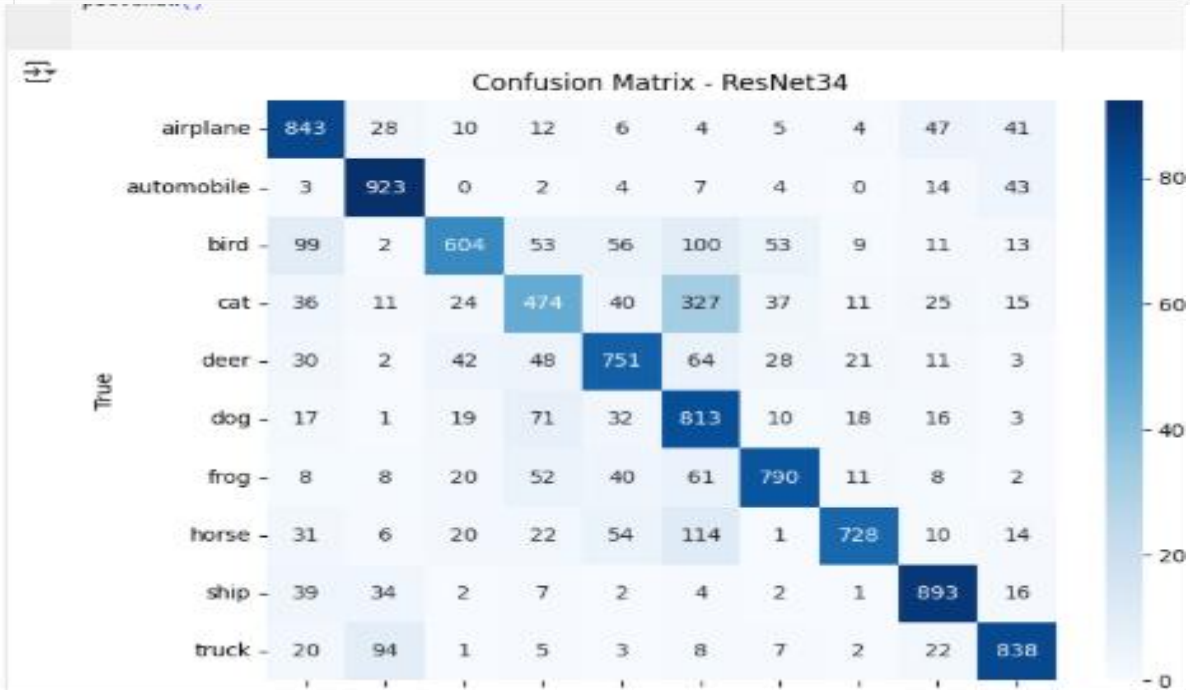
```
Running t-SNE... this may take a few minutes
```



Resnet34




```
Final Test Accuracy: 78.90%
```

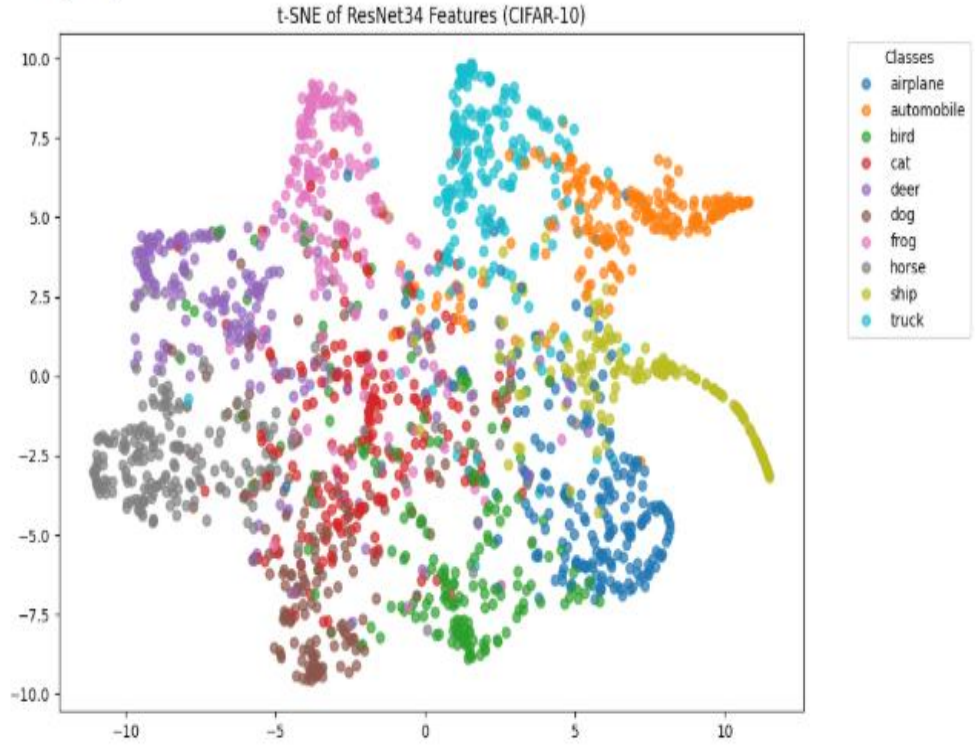


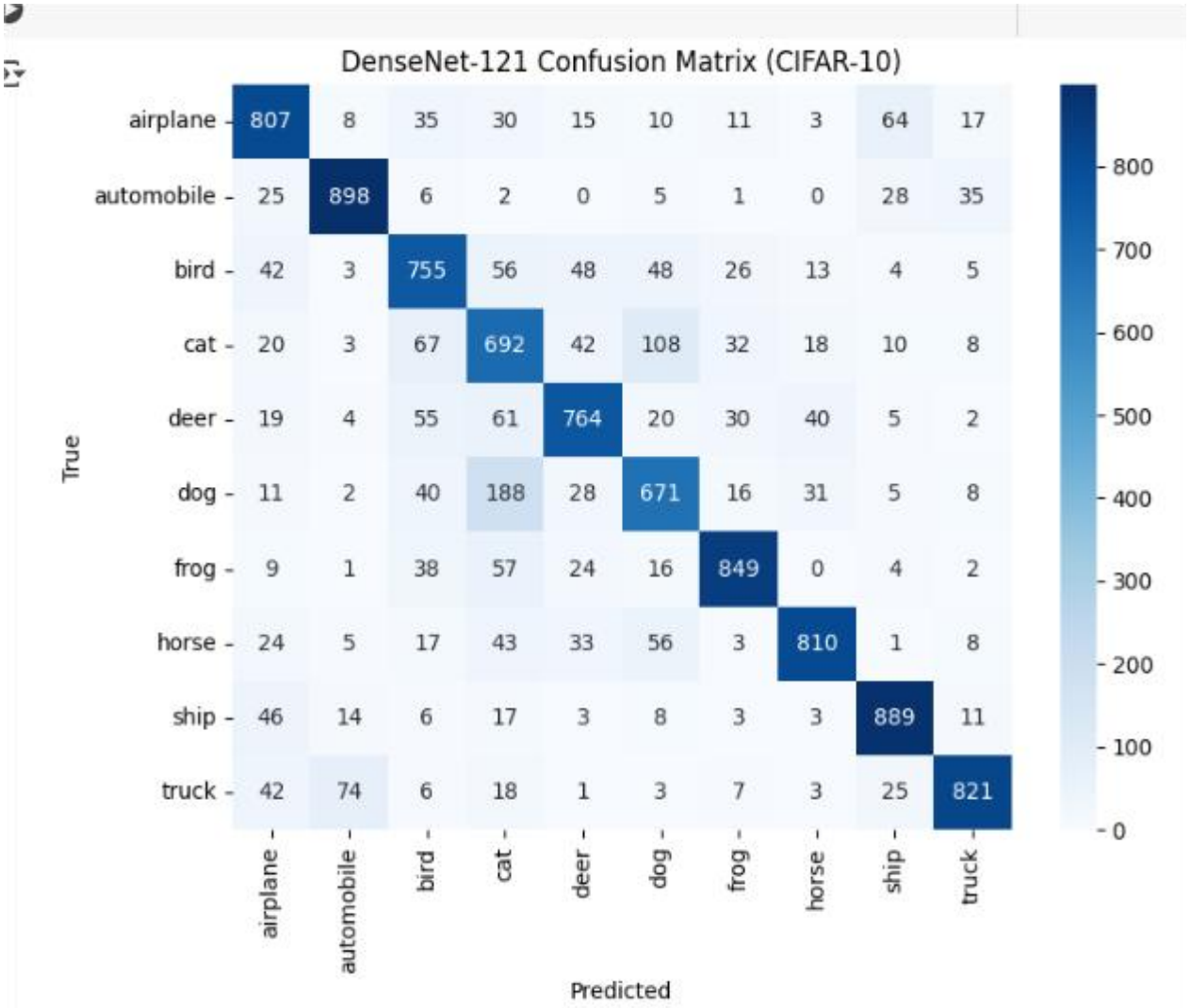
Predicted

Classification Report:

	precision	recall	f1-score	support
airplane	0.75	0.84	0.79	1000
automobile	0.83	0.92	0.88	1000
bird	0.81	0.60	0.69	1000
cat	0.64	0.47	0.54	1000
deer	0.76	0.75	0.76	1000
dog	0.54	0.81	0.65	1000
frog	0.84	0.79	0.82	1000
horse	0.90	0.73	0.81	1000
ship	0.84	0.89	0.87	1000
truck	0.85	0.84	0.84	1000
accuracy			0.77	10000
macro avg	0.78	0.77	0.76	10000
weighted avg	0.78	0.77	0.76	10000

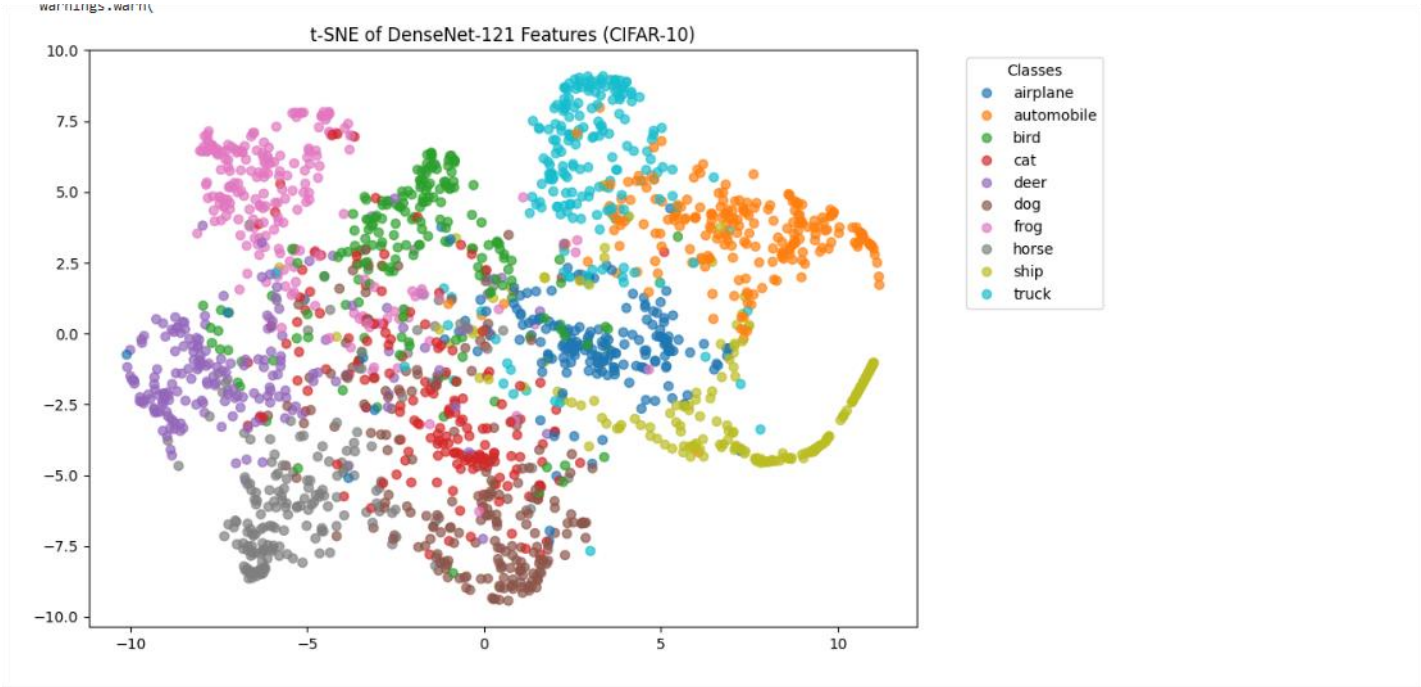
```
/usr/local/lib/python3.12/dist-packages/sklearn/manifold/_t_sne.py:1164: FutureWarning: 'n_iter' was renamed to 'max_iter' in version 1.5 and will be removed in 1.7.
  warnings.warn(
```





Classification Report:

	precision	recall	f1-score	support
airplane	0.77	0.81	0.79	1000
automobile	0.89	0.90	0.89	1000
bird	0.74	0.76	0.75	1000
cat	0.59	0.69	0.64	1000
deer	0.80	0.76	0.78	1000
dog	0.71	0.67	0.69	1000
frog	0.87	0.85	0.86	1000
horse	0.88	0.81	0.84	1000
ship	0.86	0.89	0.87	1000
truck	0.90	0.82	0.86	1000
accuracy			0.80	10000
macro avg	0.80	0.80	0.80	10000
weighted avg	0.80	0.80	0.80	10000



SECTION 6
CONCLUSION

6.1 Conclusion

In conclusion, the project successfully evaluated five distinct CNN architectures, evaluated on the CIFAR 10 Dataset providing insights into the effectiveness of both light weight and heavier Deep Learning Architectures for image classification task. This project revealed a class performance hierarchy in which DenseNet perform best with 80.41% followed by ResNet34 which had an accuracy of 76.57% . ResNet18 achieved 76% while CNN with Batch Normalization and Dropout had an accuracy of 75% and finally CNN Baseline achieved a fairly good accuracy as well.

6.2 Recommendation for Future Work

Scale the project to include CIFAR 100 for class complexity.

Implement ImagesNet to showcase the true potential of deeper architectures likeDenseNet and ResNet.

Integrate the project with selfsupervised learning to expand and implement algorithms such as contrastive learning.