Deeper Networks for Image Classifcation

Ruthwik Ganesh
230702930

***Abstract*—**This report presents a comprehensive analysis of several well established convolutional neural network (CNN) architectures, namely VGG, ResNet, GoogLeNet, and then taking a deep dive into some of the state-of-the-art CNN models like DenseNet, and EfficientNet-B0, for the task of image classification. Utilizing the MNIST and CIFAR-10 datasets, these models were evaluated to determine their performance in terms of accuracy and efficiency. The experimental setup involved a detailed assessment of each model's training dynamics and classification outcomes. Results indicate that newer architectures like DenseNet and EfficientNet-B0 offer substantial improvements in classification accuracy and computational efficiency compared to older models. Furthermore, variations in performance across the datasets highlighted the impact of architectural choices on model adaptability and scalability. This analysis not only underscores the rapid advancements in CNN design but also provides insights into the practical implications of these models in real-world applications.

I. Introduction

The primary datasets used in this study, MNIST and CIFAR-10, are benchmarks in the machine learning community for evaluating the performance of image classification models. MNIST, a dataset of handwritten digits, and CIFAR-10, which contains images of 10 different classes of objects, provide diverse challenges that test the robustness and adaptability of each CNN architecture.

This analysis aims to:

1. Evaluate and compare the performance of each CNN model in terms of accuracy and computational efficiency.
2. Understand how different architectural features influence model performance on standardized datasets.
3. Identify strengths and limitations of each model, providing a comprehensive overview that can guide future model selection and development in image classification tasks.

The subsequent sections of this report will cover a literature review of the models, detailed methodology of the experiments, presentation of results, discussion of findings, and conclusions drawn from the comparative study.

II. Literature Review

VGG Network: Developed by the Visual Graphics Group at Oxford, the VGG Network [1] simplifies CNN architectures by using stacking 3×3 convolution filters uniformly, which allows it to capture intricate details through a deep network. While VGG achieves high accuracy in image classification tasks, its main disadvantage is the significant computational and memory costs due to its depth and large number of parameters. Despite these limitations, its simplicity and effectiveness make it a popular choice for many computer vision applications.

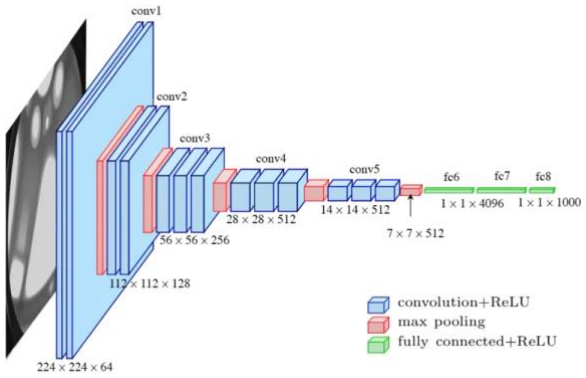


Fig. 1. VGG Layers.

ResNet (Residual Networks): ResNet revolutionized CNN designs with its introduction of skip connections [2], enabling direct gradient flow through the network which helps in training very deep networks without facing the vanishing gradient problem. ResNet models can efficiently handle layers in the hundreds, thus achieving exceptional performance across various tasks. However, these models can be large and slow to train, and their performance may plateau or even degrade with extremely deep configurations.

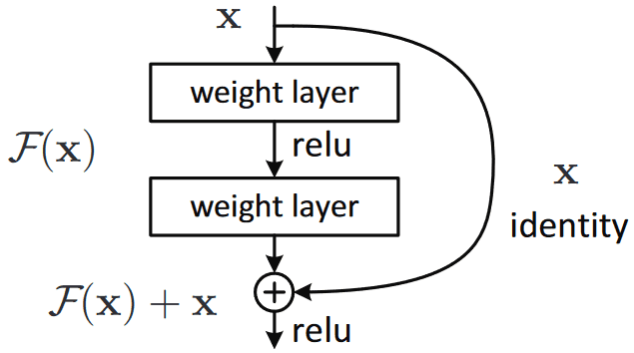


Fig. 2. Skip connection in ResNet.

GoogLeNet (Inception Network): GoogLeNet, or Inception network [3], optimizes the local sparse structure of CNNs with its novel Inception module, which includes parallel convolutional layers and pooling paths. This design allows the network to efficiently capture multi-level features, balancing model depth and computational cost effectively. Despite its efficiency, the complexity of the Inception module design poses challenges in scaling and requires careful tuning to maintain efficiency.

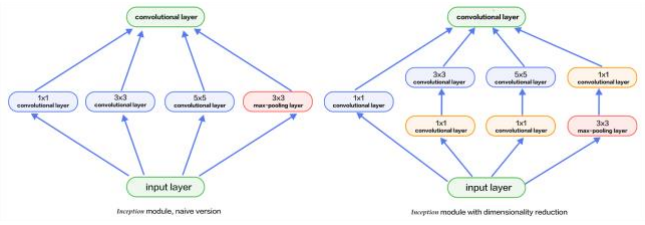


Fig. 3. Inception modules.

DenseNet (Densely Connected Convolutional Networks): DenseNet introduces dense connectivity [4], where each layer connects directly to every other layer, ensuring maximal information flow and significantly improving efficiency. This design leads to reduced model size and fewer parameters without losing accuracy, and effectively mitigates the vanishing-gradient problem. However, DenseNet can suffer from increased memory bandwidth requirements due to the concatenation of feature maps, which can make it computationally inefficient in deeper configurations.

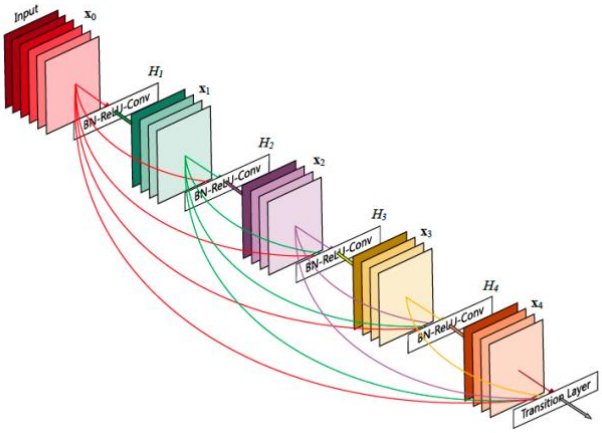


Fig. 4. 5 Layer DenseNet block with a growth rate k=4, each layer take input of preceding feature-map as input.

EfficientNet: EfficientNet employs a systematic approach to scale CNNs by balancing network depth, width, and resolution using a compound coefficient [5]. This method allows for substantial accuracy improvements with fewer parameters and reduced computational costs, making EfficientNet versatile and efficient across various tasks. Nevertheless, the optimization of scaling coefficients requires considerable computational efforts and expertise, which can be a barrier for practical applications.

III. Methodology

Model Selection and Configuration

For this study, five convolutional neural network models were selected: VGG, ResNet, GoogLeNet, DenseNet, and EfficientNet-B0. Each model was configured according to its original specifications with slight modifications to adapt to the MNIST and CIFAR-10 datasets. Models were implemented using PyTorch library.

Datasets

Two benchmark datasets were utilized in the experiments, chosen for their ubiquity and baseline complexity in image classification tasks:

MNIST (Modified National Institute of Standards and Technology database): This dataset is a classic in the field of machine learning, consisting of a large collection of 70,000 grayscale images of handwritten digits, divided into 60,000 training images and 10,000 testing images. Each image is a 28x28 pixel, single-channel (grayscale) image, representing numbers from 0 to 9. The MNIST dataset serves as a fundamental benchmark for image classification models and is particularly useful for evaluating model performance with minimal pre-processing.

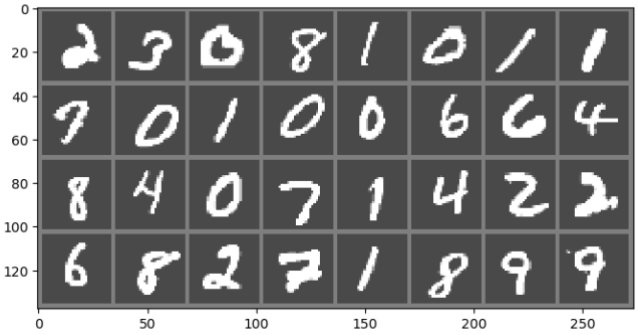


Fig. 5. Sample of MNIST handwritten digits.

CIFAR-10 (Canadian Institute for Advanced Research): Comprised of 60,000 32x32 colour images split into 50,000 training and 10,000 testing images, CIFAR-10 is categorized into 10 classes, including airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Each class is mutually exclusive. This dataset poses a greater challenge than MNIST due to its colour content and the higher complexity of the images. CIFAR-10 tests a model's ability to recognize and classify objects within a natural scene, making it an excellent dataset for advancing beyond the basics of digit recognition offered by MNIST.

Fig. 6. Sample from CIFAR-10 dataset.

Pre-processing

For the MNIST dataset, images were normalized to have a mean of and a standard deviation of (0.1307, 0.3081). Data augmentation techniques such as random horizontal flipping. As for the CIFAR-10 dataset, images were converted to floating-point and normalized to have a mean of and a standard deviation of {(0. 4914, 0. 2023), (0.4822, 0.1994), (0.4465, 0.2010)} and across the RGB channels. Data augmentation techniques such as random cropping, horizontal flipping, cutout and mixup were applied to the training data from CIFAR-10 to simulate real-world variability and to improve the robustness and generalization of the models.

Training Procedure

Each model was trained using a standard stochastic gradient descent (SGD) optimizer with a learning rate of 0.01, momentum of 0.9 and Adam optimizer with a learning rate of 0.001, and a mini-batch size of 64. The learning rate was adjusted according to a reduce on plateau schedule, reducing by a factor of 0.1 every 5 epochs and Cosine Annealing schedule at 200 iterations. Training was conducted over varying number of epochs.

Evaluation Metrics

Model performance was primarily evaluated based on classification accuracy on the test set. Additionally, a classification confusion matrix was also calculated to provide a comprehensive view of model effectiveness. Performance comparisons across models focused on these metrics, as well as computational efficiency measured by training time and the number of parameters.

Hardware and Software

Experiments were carried out on a computing cluster equipped with NVIDIA A100 GPUs. The software stack included Python 3.10.13, and PyTorch 2.2.0+cu118.

IV. Experiments and Results

ResNet, VGG and GoogLeNet models were trained on MNIST and CIFAR-10 datasets with minor changes in the architecture when compared to the ones mentioned in their respective papers.

The table below outlines models' performance trained on MNIST dataset.

TABLE I.

Model	Test Accuracy (%)	Params	Model Size (MB)
VGG-19	97.15	20,564,682	83.08
ResNet34	99.12	11,172,810	52
GoogLeNet	98.86	5,992,090	27.37
DenseNet	99.07	6,957,834	30.06

The table below outlines models' performance trained on CIFAR-10 dataset.

TABLE II.

Model	Test Accuracy (%)	Params	Model Size (MB)
VGG-19	88.87	20,565,834	83.1
ResNet34	90.63	21,282,122	96.82
GoogLeNet	84.42	5,998,362	23.9
DenseNet	90.06	6,956,298	32.42
EfficientNet-B0	90.14	3,598,598	22.13

Optimizers

Below table outlines the test accuracies of the CNN models trained using Adam and SGD optimizers.

TABLE III.

Model	Adam	SGD
VGG-16	88.97	88.75
VGG-19	88.87	87.57
ResNet50	91.30	89.38
ResNet101	91.51	89.66
GoogLeNet	84.42	81.59
EfficientNet-B0	90.14	83.96
DenseNet121	90.06	88.39
DenseNet201	84.30	88.33

Batch size

To test the effect of batch size on model training time and accuracy ResNet18 was trained on various batch sizes.

TABLE IV.

Batch Size	Test Accuracy (%)	Time / epoch
16	89.90	01:29
32	91.56	01:10
64	92.53	01:12
128	92.92	00.43
256	92.11	00:29
512	92.71	00:26

Learning rate

To glean on the impact of learning rate ResNet18 model was trained on various learning rates and model performance was noted.

TABLE V.

Learning rate	Train Accuracy (%)	Test Accuracy (%)
0.01	77.45	55.85
0.001	93.66	87.62
0.0001	97.59	87.96
0.00001	86.56	79.95

Augmentations

1a. Cutout augmentation with 1 square hole of varying pixel sizes of 2, 4, 8 and 16.

TABLE VI.

Cutout size	Train accuracy (%)	Test accuracy (%)
2	95.14	80.80
4	97.41	81.17
8	91.88	80.79
16	85.30	82.41

1b. Cutout augmentation with 4, 8 and 16 square holes of pixel size 2.

TABLE VII.

Cutouts	Train accuracy (%)	Test accuracy (%)
4	97.60	85.38
8	97.77	85.70
16	97.31	85.61

2. Mix-up augmentation with an alpha value 0.2 and 0.4 were tested.

TABLE VIII.

Mix-up alpha	Train accuracy (%)	Test accuracy (%)
0.2	89.89	90.13
0.4	83.79	89.98

Zero-shot Image classifier using CLIP
CLIP embeddings were used to classify images by passing image features to get and class labels.

TABLE IX.

Dataset	Zero-Shot Accuracy (%)
Imagenette	98.70
CIFAR-10	87.24

V. DISCUSSION

In the initial investigation, VGG-19, ResNet34, GoogLeNet, and DenseNet, were trained on the MNIST dataset. Remarkably, these models exhibited a propensity for early convergence (Table 1.), typically within a modest epoch range of 20. This rapid convergence underscored the innate simplicity of the MNIST dataset, wherein the discernible features were readily captured by these models. Given these findings, it is imperative to extend the scope of inquiry by conducting further experiments utilizing the CIFAR-10 dataset, which presents a more complex and diverse set of visual receptive field, thereby potentially eliciting better performance on these models.

Upon training these models on CIFAR-10 we see results as tabulated in Table 2. In-depth analysis of this result is as follows.

ResNet34 tops the list with a test accuracy of 90.63%, demonstrating its effectiveness in utilizing deep residual learning to handle complex image patterns effectively. EfficientNet-B0 closely follows with 90.14% accuracy, illustrating the strength of its balanced scaling strategy across depth, width, and resolution, which optimizes both performance and computational efficiency. DenseNet records a competitive 90.06% accuracy, leveraging its densely connected architecture that ensures maximum information transfer between layers and efficient feature utilization. VGG-19, although slightly behind at 88.87%, showcases robust feature extraction capabilities, though it lacks the advanced learning mechanisms of the newer models. GoogLeNet, with the lowest accuracy at 84.42%, might struggle with CIFAR-10's complexity, indicating that while its inception modules are efficient, they may not capture sufficient high-level features needed for such diverse datasets.

Parameter count directly impacts training and inference times, crucial for model scalability and deployment. GoogLeNet is the most parameter-efficient model with only 5.9M parameters, making it ideal for environments with limited computational resources. EfficientNet-B0 impresses with its minimal parameter use of 3.5M while delivering high accuracy, highlighting its efficient use of model scaling laws. In contrast, ResNet34 and VGG-19 employ 21M and 20M parameters respectively, reflecting older design philosophies that favor depth and width without proportionate attention to parameter efficiency. DenseNet, with 6.9M parameters, strikes a balance between depth and parameter usage, utilizing its connectivity to enhance learning efficiency.

Model size is a critical consideration for deployment, especially in resource-constrained environments. EfficientNet-B0 leads with a minimal footprint of 22.13 MB, supporting its deployment in mobile and embedded systems without sacrificing performance. GoogLeNet also maintains a small model size at 23.9 MB, aligning with its design for computational frugality. DenseNet offers a modest size of 32.42 MB, making it feasible for moderate-resource environments. However, VGG-19 and ResNet34 are on the higher end with model sizes of 83.1 MB and 96.82 MB respectively, which may pose challenges in limited-storage scenarios.

ResNet34, DenseNet, and EfficientNet-B0 emerge as highly effective, balancing accuracy with resource usage, whereas GoogLeNet, despite its lower accuracy, offers benefits in parameter and size efficiency. VGG-19, though less efficient by modern standards, remains competitive in accuracy, making it a viable option where resource constraints are less stringent.

In the next set of experiments, we delve in to tuning hyper-parameters like batch size and learning rate for model improvement and discuss the optimizers that are best suited for classification tasks,

Optimizers adjust neural network parameters to minimize loss functions. Stochastic Gradient Descent (SGD) updates model weights by approximating the gradient using small subsets of data, which can aid in escaping local minima due to the inherent noise. Adam, or Adaptive Moment Estimation [6], adapts learning rates based on first and second moments of gradients, facilitating faster convergence. While Adam is efficient for large datasets and parameters, SGD might achieve better long-term convergence with an appropriate learning rate schedule. Each optimizer's efficacy varies by context, necessitating experiments to identify optimal configurations for specific datasets and models.

Comparing results from Table III, Adam consistently outperformed Stochastic Gradient Descent (SGD) for VGG, ResNet, GoogLeNet, EfficientNet, and DenseNets, due to its adaptive learning rate capabilities that facilitate faster and more effective convergence on complex datasets. The experiments indicate that Adam's mechanism of adjusting learning rates dynamically for each parameter offers significant advantages over SGD, particularly in models like EfficientNet and GoogLeNet where the performance disparity was more pronounced.

Batch size denotes the number of training examples utilized per iteration. Larger batch sizes facilitate efficient computation by leveraging parallel processing but require substantial memory. Conversely, smaller batches contribute to better generalization due to noise-induced regularization, albeit at the risk of unstable training dynamics due to noisier gradient estimates. ResNet18 training, results (Table IV) show an efficiency peak at a batch size of 128, where test accuracy was highest while significantly reducing epoch time. Larger batches processed data more efficiently, leveraging computational resources, but at a cost to model

accuracy beyond this optimal point, suggesting a balance between computational efficiency and gradient estimation noise is crucial for optimal model performance.

Learning rate is a pivotal hyperparameter that determines the magnitude of updates to the model's weights during training, directly influencing the convergence speed and the quality of the final model. A high learning rate may lead to rapid convergence but risks overshooting minimal points, while a low rate might stabilize training yet result in slow convergence and potential entrapment in local minima. The learning rate experiments (Table V) further highlight its critical role in model training dynamics. A moderate learning rate of 0.001 yielded the best balance between training and test accuracy, emphasizing the need for careful tuning to avoid overshooting minima or excessively slow convergence, as seen with rates set too high or too low, respectively.

Finally, we investigate augmentation techniques such as cutout and mix-up which are best suited for preprocessing image data.

Cutout augmentation promotes enhanced generalization by randomly masking parts of training images, compelling the model to learn robust features that are not tied to specific visual cues. This approach significantly boosts the model's performance on unseen data. It also increases model robustness, making it less sensitive to the position and presence of objects, which is crucial in real-world scenarios where objects may be obscured. Additionally, cutout serves as a regularizer, preventing overfitting by discouraging the memorization of intricate, noise-based details within the training data, thus improving general applicability.

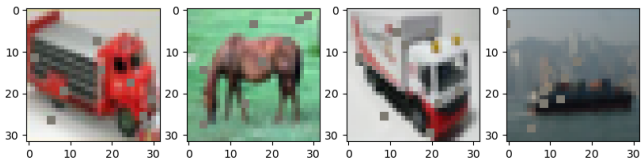


Fig. 7. 8 Cutouts of each 2x2 pixels on CIFAR-10 images.

In implementing cutout augmentation, the regions to be cut out can vary in shape, though squares are commonly used for their simplicity. Key hyperparameters include the number and size of these cutouts, chosen based on image size and dataset complexity—larger or more numerous cutouts may be needed for larger or more complex images.

In two distinct experiments assessing the impact of cutout augmentation on image classification, varying sizes and numbers of cutouts were explored. Experiment 1a (Table VI) manipulated the size of a single square cutout (2, 4, 8, 16 pixels), revealing that training accuracy declines as cutout size increases, indicating larger cutouts obscure critical learning features. However, test accuracy was highest with a moderate-sized cutout (16 pixels), suggesting that smaller cutouts may not adequately regularize, while larger ones enhance generalization on unseen data. Experiment 1b (Table VII) maintained a constant cutout size (2 pixels) but varied the number of cutouts (4, 8, 16). Results showed high training accuracy across all variants, with a peak at 8 cutouts. Yet, test accuracy improved initially but slightly decreased at the highest number of cutouts, indicating potential over-regularization.

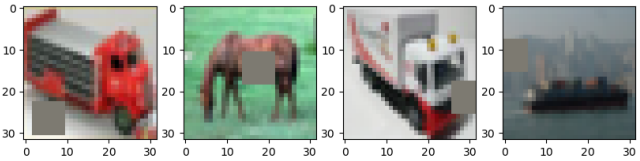


Fig. 8. 1 Cutout of 8x8 pixels on CIFAR-10 images.

The combined findings from these experiments demonstrate that while larger cutouts and increased numbers of cutouts can enforce stronger regularization and simulate more diverse occlusion scenarios, there is a critical balance to be struck. Optimal cutout size and number must challenge the model sufficiently without overwhelming it, ensuring effective learning and robust generalization without excessive over-regularization. This balance is essential for maximizing the efficacy of cutout augmentation in improving model performance on complex image classification tasks.

Mix-up augmentation is a technique for training neural networks, especially beneficial for image classification tasks. This method enhances model generalization and robustness, offering an alternative to traditional Empirical Risk Minimization (ERM). Mix-up [7], [9] operates by creating new training samples through convex combinations of pairs of existing samples and their labels. For two randomly selected training examples (x_i, y_i) and (x_j, y_j), where x represents the input and y is the one-hot encoded label, mix-up calculates a new sample using a mixing parameter λ from a Beta distribution as shown below:

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j,$$
$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j,$$

This approach not only mixes the inputs but also the labels, encouraging the model to adopt linear behaviors between training examples and avoid overfitting. This linearity helps the model generalize more effectively to new, unseen data rather than simply memorizing the training set.

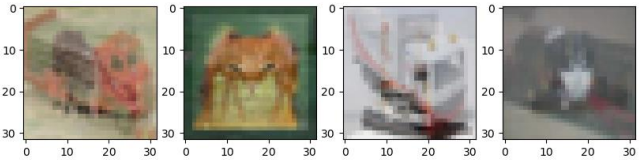


Fig. 9. Mix-up augmentation overlays image from one class onto the other.

The table (Table VII) illustrates the impact of the mix-up parameter alpha on training and test accuracy for a neural network. With alpha = 0.2, both training (89.89%) and test accuracy (90.13%) are high, indicating effective generalization with moderate data interpolation. Increasing alpha to 0.4 lowers training accuracy to 83.79% but maintains high test accuracy at 89.98%, suggesting that stronger interpolation helps prevent overfitting, as the model remains robust on unseen data despite reduced training performance.

This trend demonstrates that higher alpha values enhance generalization at the cost of slight decreases in training accuracy, confirming mix-up's utility in improving model robustness against overfitting.

Significant benefits of mix-up include improved model robustness against adversarial attacks and input perturbations, and a reduction in the likelihood of memorizing corrupted labels. Despite these advantages, mix-up is straightforward to implement and adds minimal computational overhead, making it an efficient and potent augmentation method.

VI. Novel approach to zero-shot image classification

Zero-shot learning (ZSL) represents a significant frontier in machine learning, particularly within the realm of image classification. Traditional image classification techniques require extensive labeled datasets for each class; however, zero-shot learning aims to classify images into categories that have never been explicitly labeled during the training phase. This capability is especially critical in scenarios where obtaining labeled examples is impractical.

Zero-shot image classification using Contrastive Language–Image Pre-training (CLIP) [8] embeddings is a method where a model classifies images into categories it has never explicitly been trained on. CLIP is pre-trained on a diverse dataset of image-text pairs, can generate embeddings for both images and textual descriptions. For classification, an image is first passed through CLIP's image encoder, producing a vector representation. Simultaneously, potential class labels are formulated as descriptive text phrases and encoded into vectors using the CLIP text encoder. The core of zero-shot classification lies in measuring the similarity (typically cosine similarity) between the image's vector and each label's vector. The class whose text vector is closest to the image vector is predicted as the most appropriate category for the image. This technique allows CLIP to apply its learned visual-textual associations to categorize new images into classes it was not explicitly trained to recognize, demonstrating a profound understanding of both visual and linguistic content.

CLIP extends to numerous applications including image captioning, content-based image retrieval, and automated tagging systems, making it a pivotal tool in AI for bridging multimodal information and adapting to a broad spectrum of visual understanding tasks. CLIP is a versatile and powerful model for bridging visual and textual data, but like all models, it has its limitations. The model's reliance on internet-sourced image-text pairs introduce biases from the training data, often resulting in skewed outcomes. Although CLIP has improved robustness compared to traditional models, it remains susceptible to adversarial attacks where minor, typically undetectable alterations to images can mislead the model. The model's dual-encoder architecture demands substantial computational resources, limiting its accessibility due to the extensive data processing required. Additionally, CLIP struggles with contextual subtleties and abstract concepts not well-represented in its training set, affecting its performance on specialized tasks or unfamiliar datasets. The model's dependence on the precision of textual descriptions also influences its accuracy, as poorly articulated or vague labels can lead to imprecise classifications.

VII. Conclusion

This study has presented a comprehensive analysis of five major convolutional neural network architectures: VGG, ResNet, GoogLeNet, DenseNet, and EfficientNet-B0, utilizing MNIST and CIFAR-10 datasets to assess their performance in image classification tasks. The results confirm that newer architectures like DenseNet and EfficientNet-B0 outperform older models in terms of classification accuracy and computational efficiency, demonstrating the importance of architectural innovation in handling complex image patterns. The use of advanced augmentation techniques and optimization strategies further highlighted the adaptability and scalability of these models to various data complexities.

While the current study has made substantial contributions to understanding the comparative advantages of these models, several avenues remain open for further exploration. Future work could focus on the integration of hybrid models that combine the strengths of these architectures to enhance performance. Additionally, exploring the application of these models in real-world scenarios across different domains could provide deeper insights into their practical viability and limitations. Lastly, continued refinement of training strategies and learning rates, especially in the context of emerging datasets and evolving computational platforms, will be crucial in maintaining the relevance and effectiveness of these deep learning models in advancing the field of computer vision.

References

[1] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv:1409.1556, 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," arXiv:1512.03385, 2015.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," arXiv:1409.4842, 2014.

[4] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," arXiv:1608.06993, 2016.

[5] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," arXiv:1905.11946, 2019.

[6] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv:1412.6980v9, 2017.

[7] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond Empirical Risk Minimization," arXiv:1710.09412, 2017.

[8] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning Transferable Visual Models From Natural Language Supervision," arXiv:2103.00020, 2021.

[9] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, M. Li, "Bag of Tricks for Image Classification with Convolutional Neural Networks," arXiv:1812.01187, 2018.

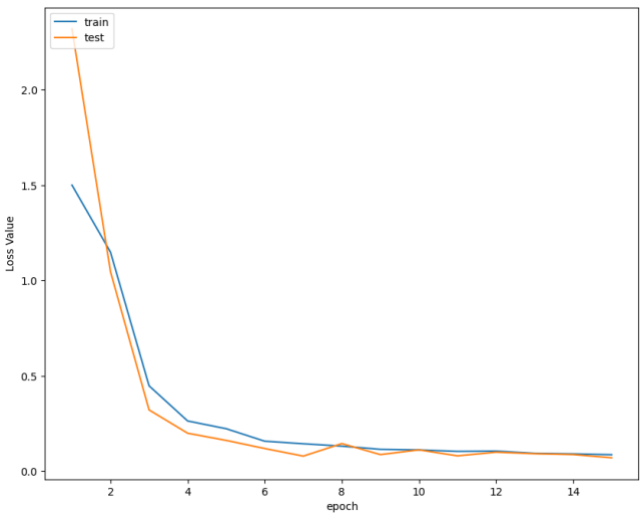
Appendix

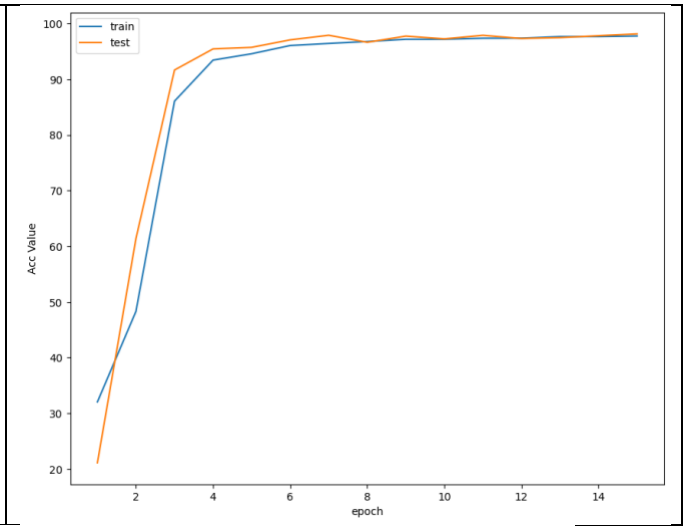
Failure cases –

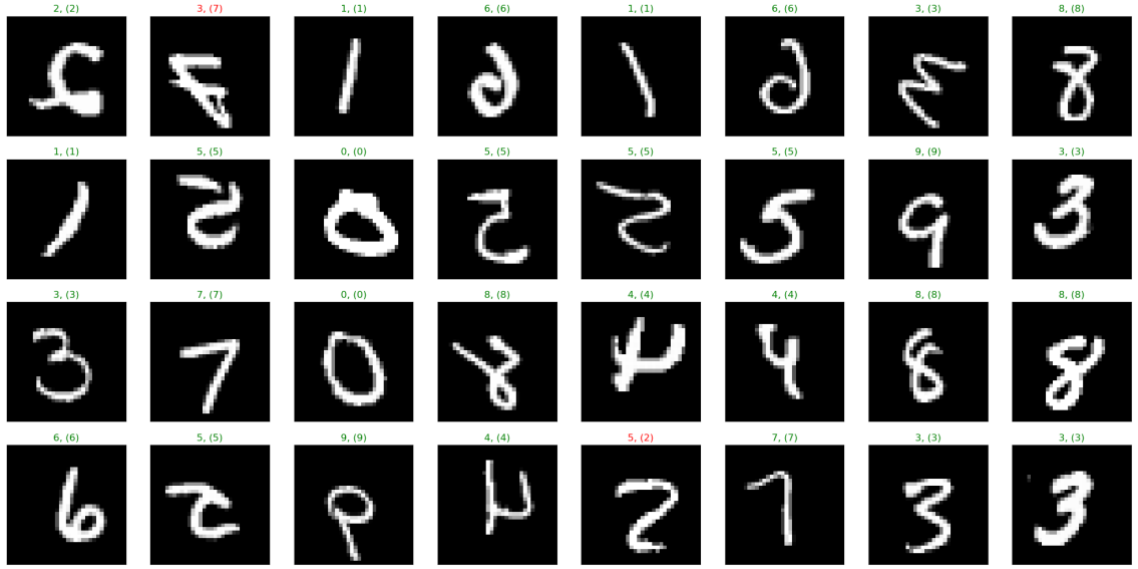
1. Mixed Precision Training of ResNet, DenseNet on CIFAR10 dataset – Did not show any performance improvement.
2. Poor zero shot performance of The MNIST and Fashion-MNIST datasets on image classifier using CLIP.

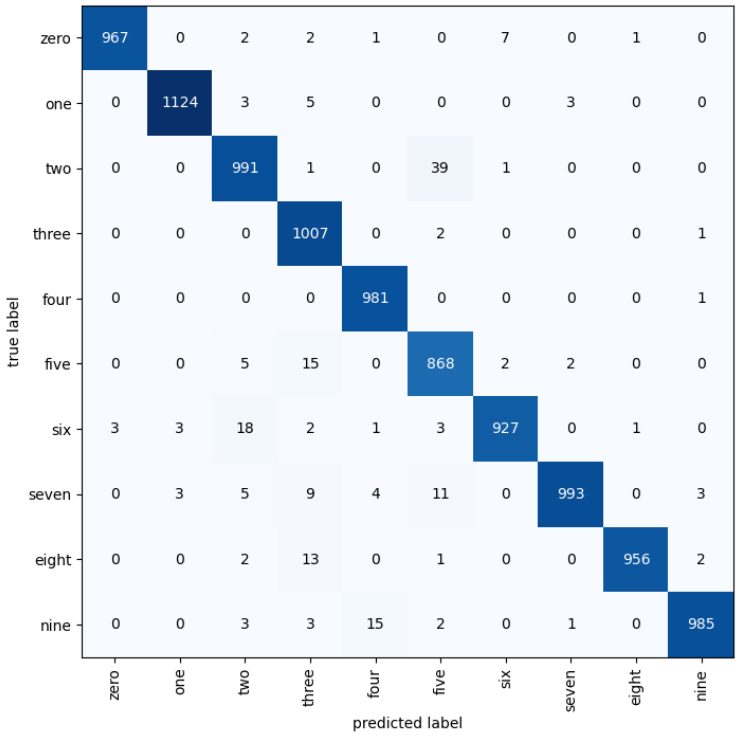
MNIST training

VGG-19

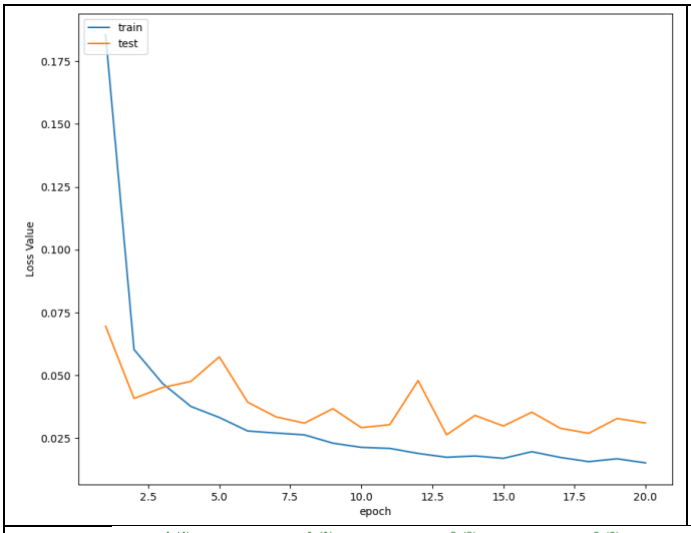


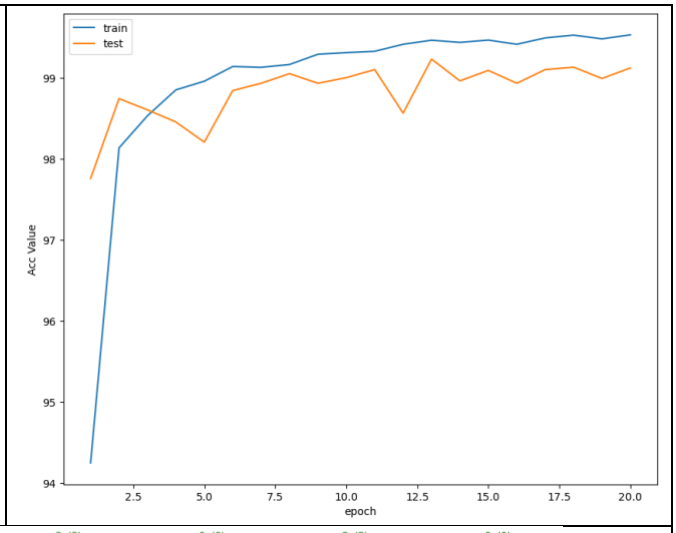


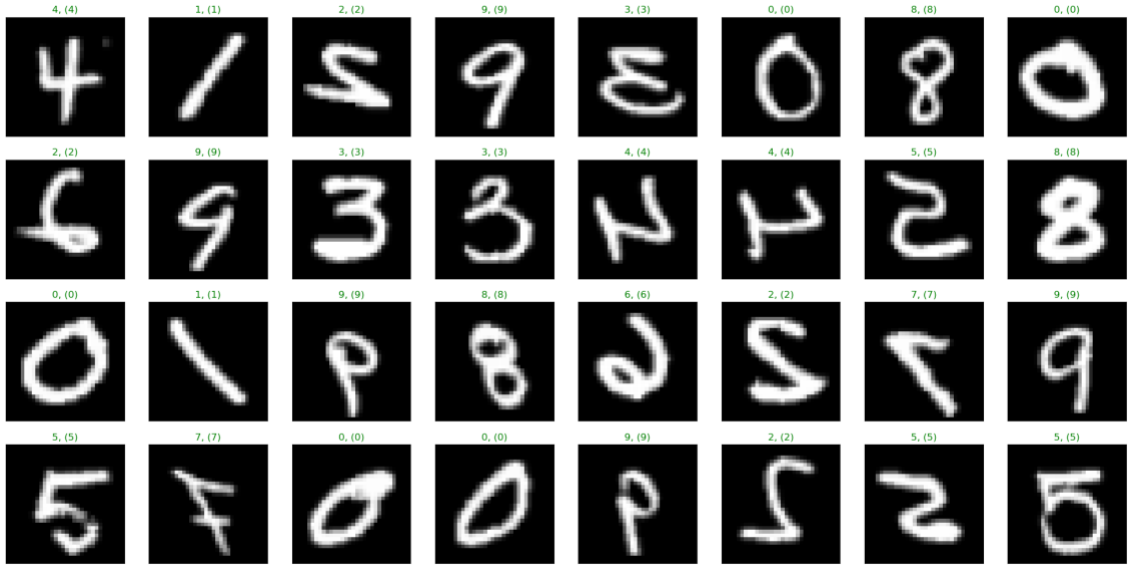


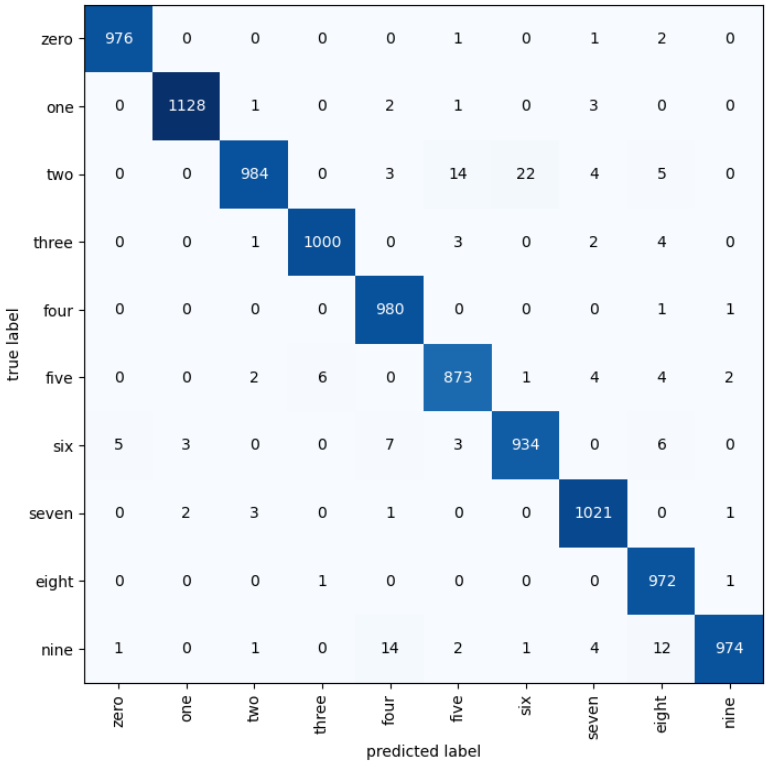


ResNet34

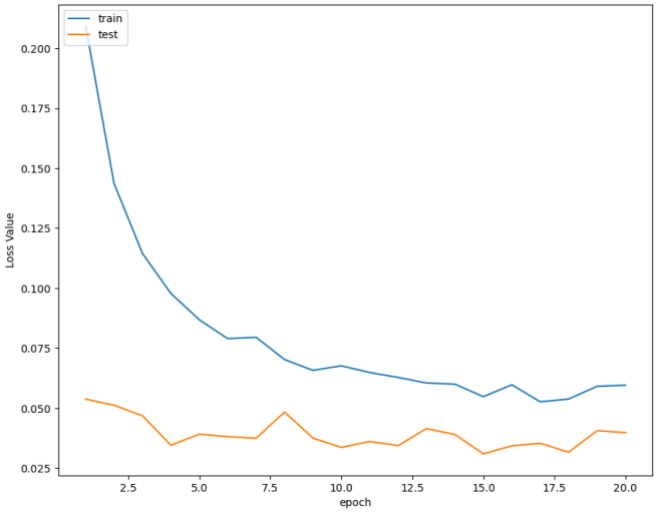

train
test
Loss Value
epoch

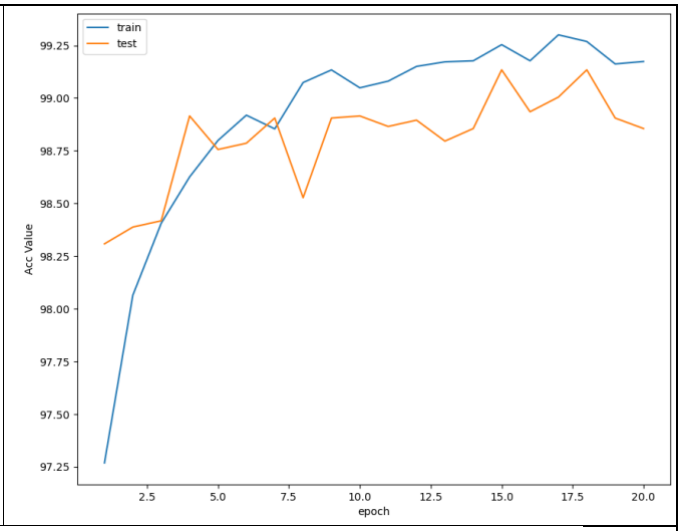

train
test
Acc Value
epoch

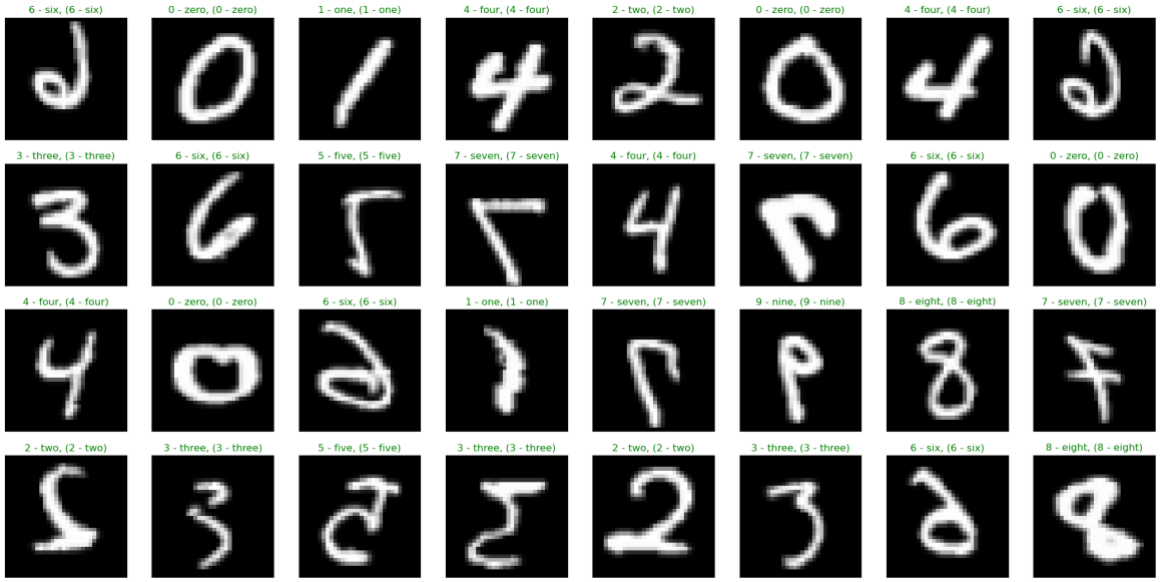

4, (4)
1, (1)
2, (2)
9, (9)
3, (3)
0, (0)
8, (8)
0, (0)
2, (2)
9, (9)
3, (3)
3, (3)
4, (4)
4, (4)
5, (5)
8, (8)
0, (0)
1, (1)
9, (9)
8, (8)
6, (6)
2, (2)
7, (7)
9, (9)
5, (5)
7, (7)
0, (0)
0, (0)
9, (9)
2, (2)
5, (5)
5, (5)

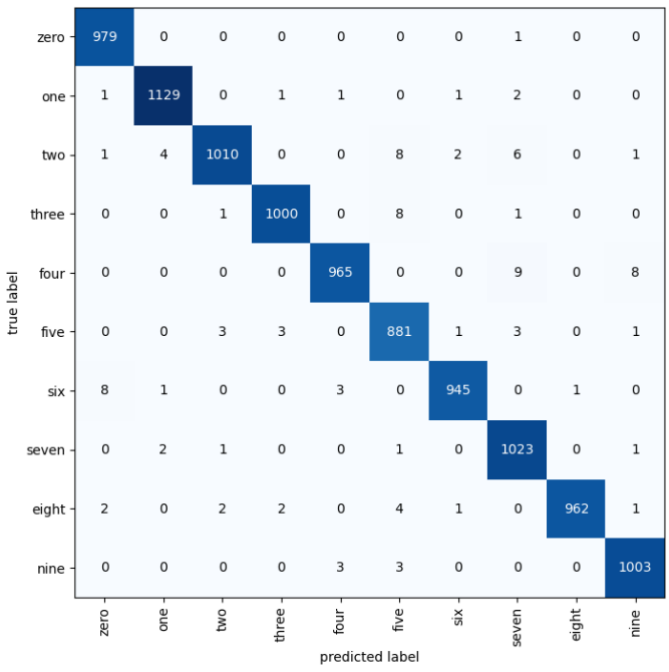

zero 976 0 0 0 0 1 0 1 2 0
one 0 1128 1 0 2 1 0 3 0 0
two 0 0 984 0 3 14 22 4 5 0
three 0 0 1 1000 0 3 0 2 4 0
four 0 0 0 0 980 0 0 0 1 1
five 0 0 2 6 0 873 1 4 4 2
six 5 3 0 0 7 3 934 0 6 0
seven 0 2 3 0 1 0 0 1021 0 1
eight 0 0 0 1 0 0 0 0 972 1
nine 1 0 1 0 14 2 1 4 12 974
true label
zero one two three four five six seven eight nine
predicted label

GoogLeNet


train
test
Loss Value
epoch

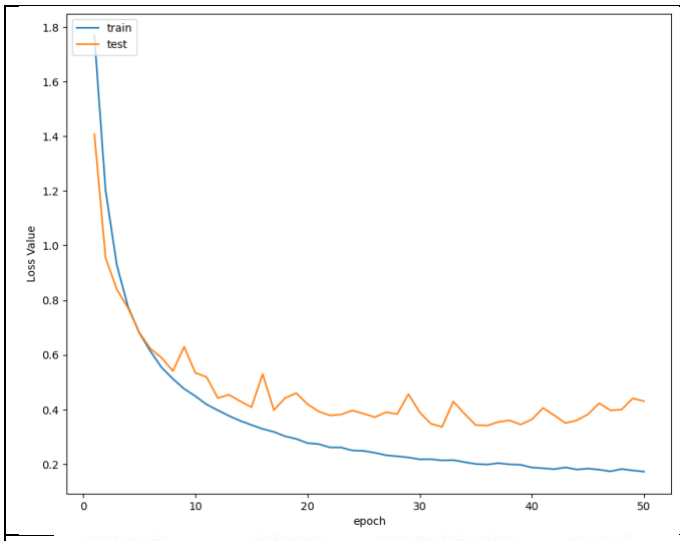

train
test
Acc Value
epoch

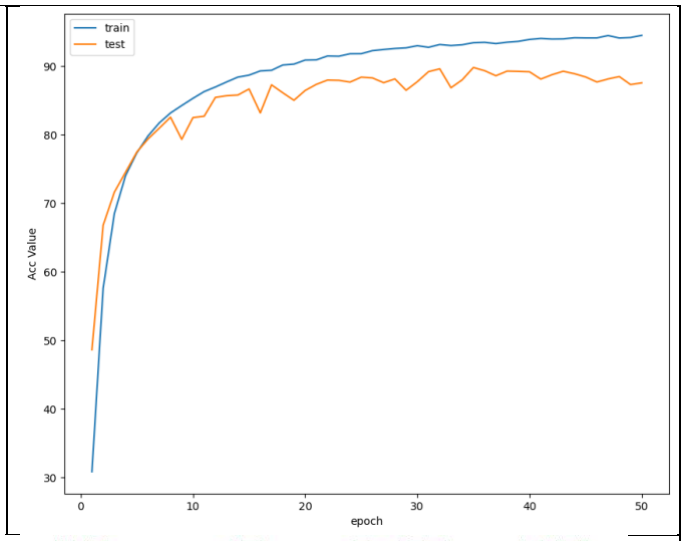

6 - six, (6 - six)
0 - zero, (0 - zero)
1 - one, (1 - one)
4 - four, (4 - four)
2 - two, (2 - two)
0 - zero, (0 - zero)
4 - four, (4 - four)
6 - six, (6 - six)
3 - three, (3 - three)
6 - six, (6 - six)
5 - five, (5 - five)
7 - seven, (7 - seven)
4 - four, (4 - four)
7 - seven, (7 - seven)
6 - six, (6 - six)
0 - zero, (0 - zero)
4 - four, (4 - four)
0 - zero, (0 - zero)
6 - six, (6 - six)
1 - one, (1 - one)
7 - seven, (7 - seven)
9 - nine, (9 - nine)
8 - eight, (8 - eight)
7 - seven, (7 - seven)
2 - two, (2 - two)
3 - three, (3 - three)
5 - five, (5 - five)
3 - three, (3 - three)
2 - two, (2 - two)
3 - three, (3 - three)
6 - six, (6 - six)
8 - eight, (8 - eight)

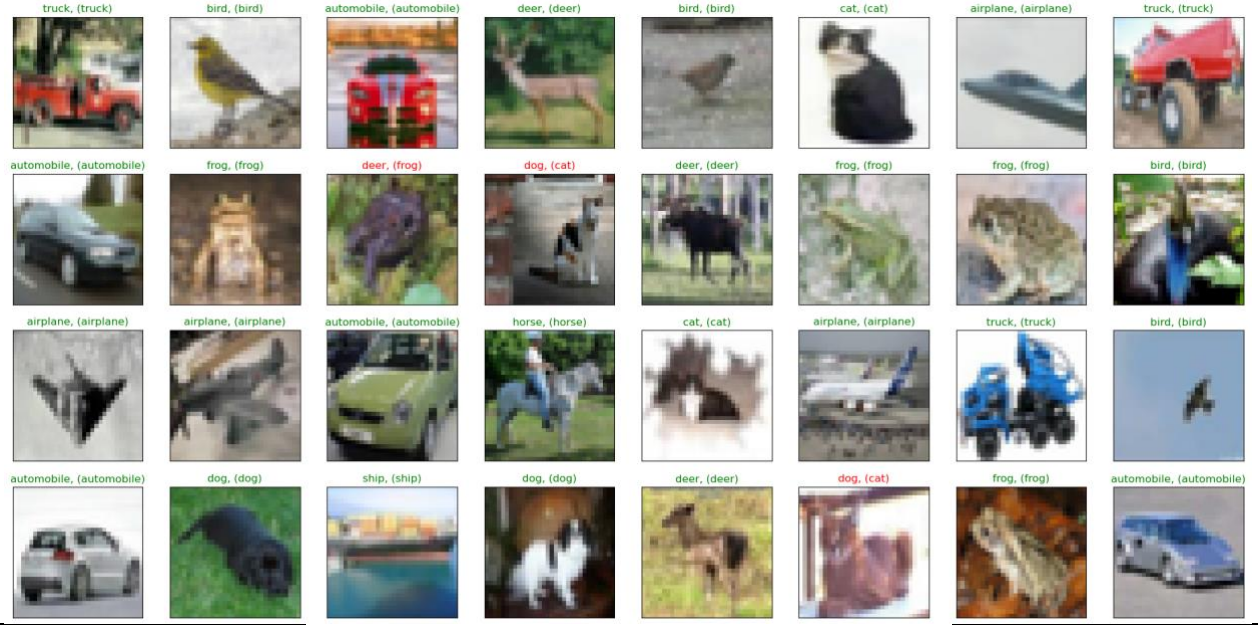

zero 979 0 0 0 0 0 0 1 0 0
one 1 1129 0 1 1 0 1 2 0 0
two 1 4 1010 0 0 8 2 6 0 1
three 0 0 1 1000 0 8 0 1 0 0
four 0 0 0 0 965 0 0 9 0 8
five 0 0 3 3 0 881 1 3 0 1
six 8 1 0 0 3 0 945 0 1 0
seven 0 2 1 0 0 1 0 1023 0 1
eight 2 0 2 2 0 4 1 0 962 1
nine 0 0 0 0 3 3 0 0 0 1003
true label
zero one two three four five six seven eight nine
predicted label

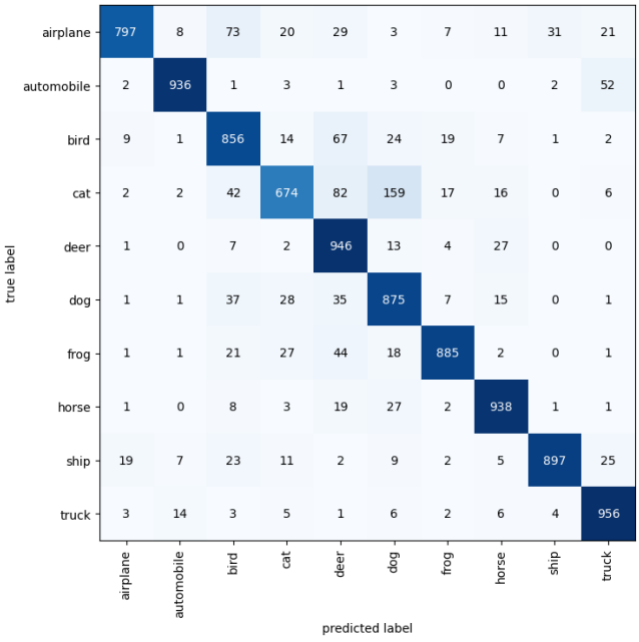
CIFAR-10 Training

VGG-19

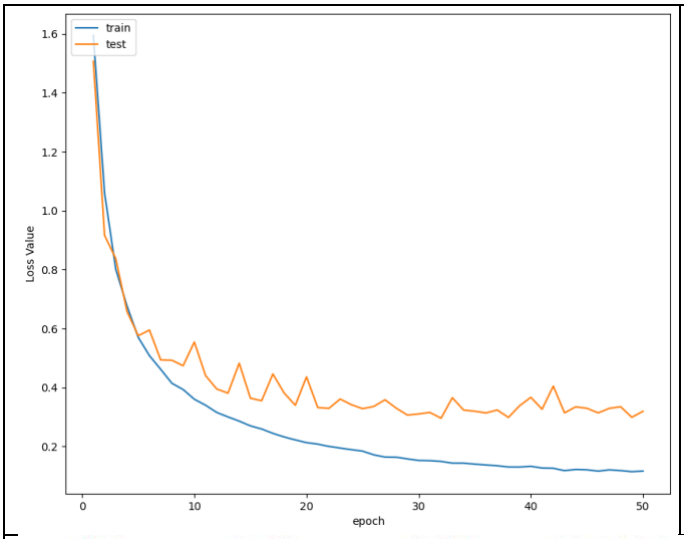


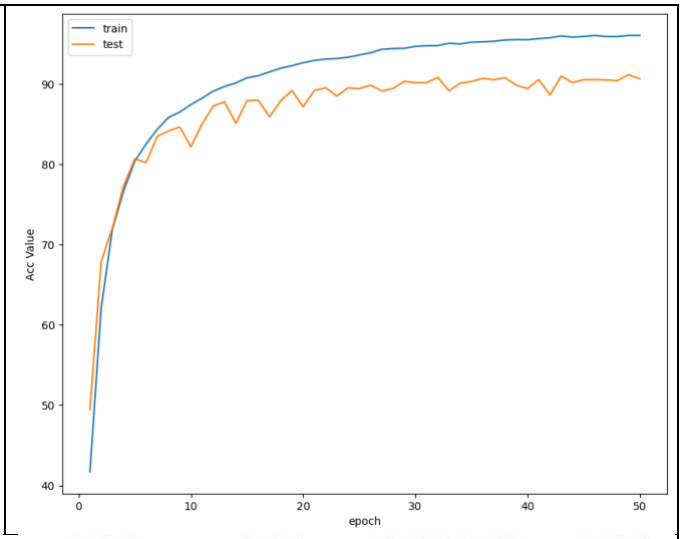


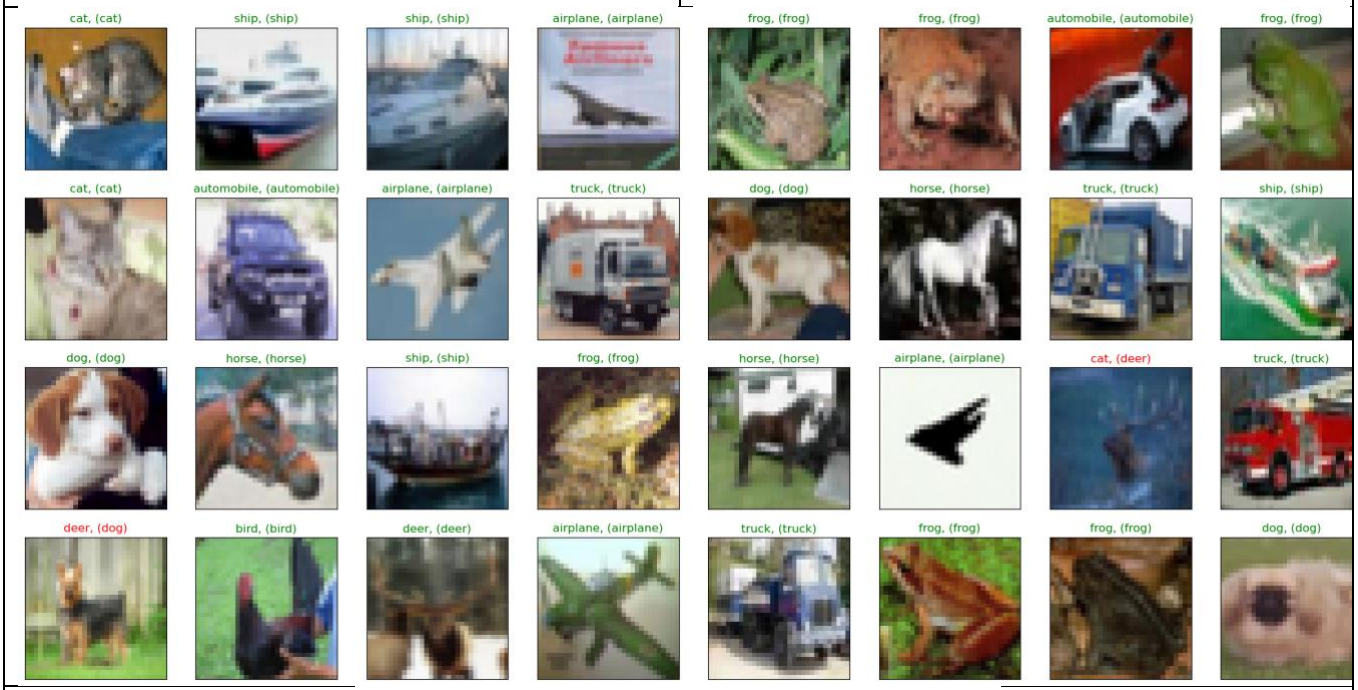


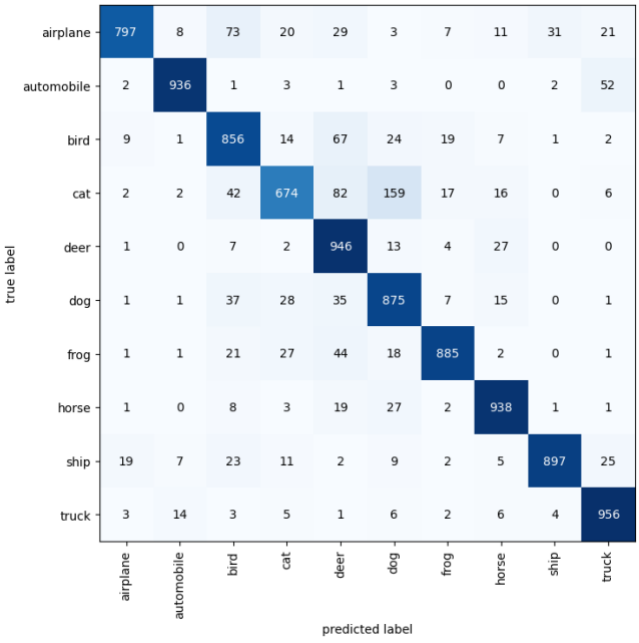


ResNet34

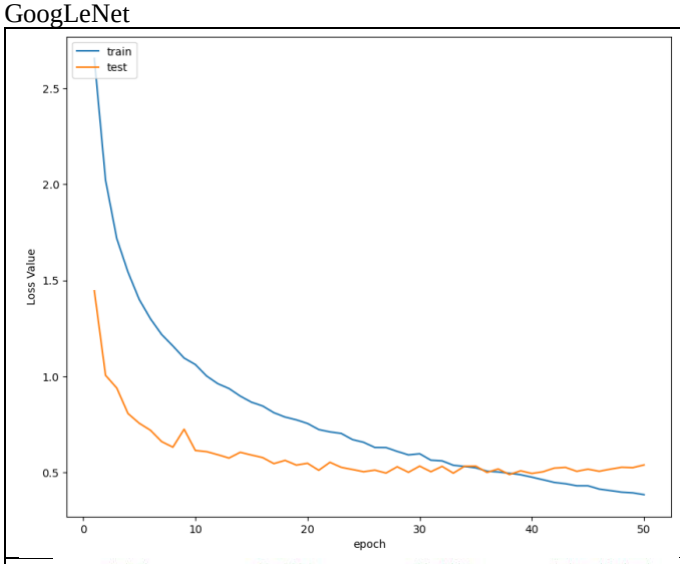

train
test
Loss Value
epoch

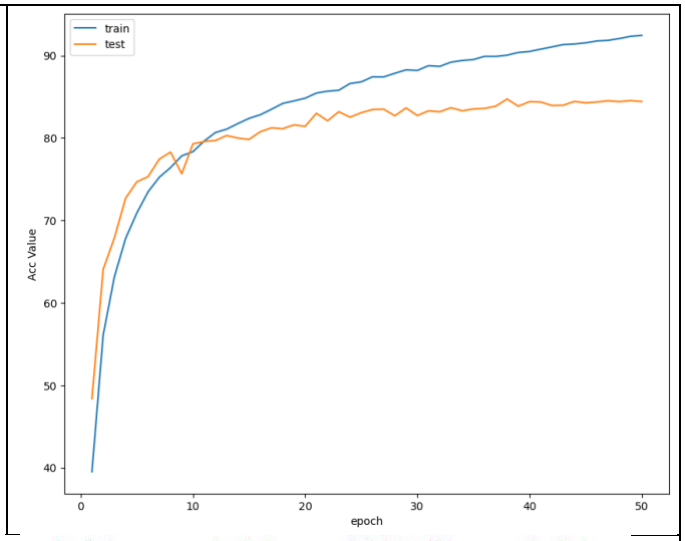

train
test
Acc Value
epoch


cat, (cat)
ship, (ship)
ship, (ship)
airplane, (airplane)
frog, (frog)
frog, (frog)
automobile, (automobile)
frog, (frog)
cat, (cat)
automobile, (automobile)
airplane, (airplane)
truck, (truck)
dog, (dog)
horse, (horse)
truck, (truck)
ship, (ship)
dog, (dog)
horse, (horse)
ship, (ship)
frog, (frog)
horse, (horse)
airplane, (airplane)
cat, (deer)
truck, (truck)
deer, (dog)
bird, (bird)
deer, (deer)
airplane, (airplane)
truck, (truck)
frog, (frog)
frog, (frog)
dog, (dog)

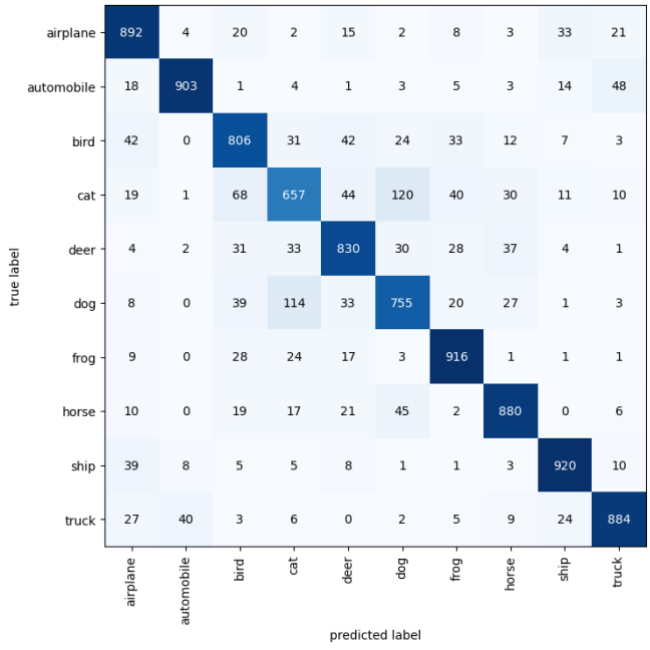

airplane 797 8 73 20 29 3 7 11 31 21
automobile 2 936 1 3 1 3 0 0 2 52
bird 9 1 856 14 67 24 19 7 1 2
cat 2 2 42 674 82 159 17 16 0 6
deer 1 0 7 2 946 13 4 27 0 0
dog 1 1 37 28 35 875 7 15 0 1
frog 1 1 21 27 44 18 885 2 0 1
horse 1 0 8 3 19 27 2 938 1 1
ship 19 7 23 11 2 9 2 5 897 25
truck 3 14 3 5 1 6 2 6 4 956
true label
airplane automobile bird cat deer dog frog horse ship truck
predicted label

GoogLeNet

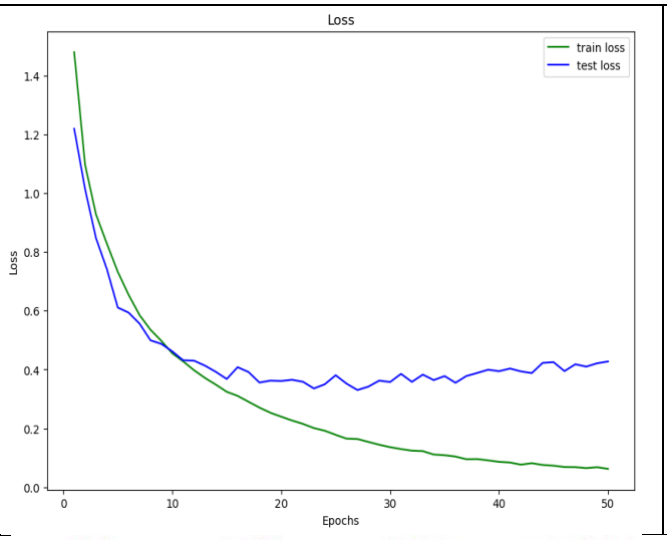

train
test
Loss Value
epoch

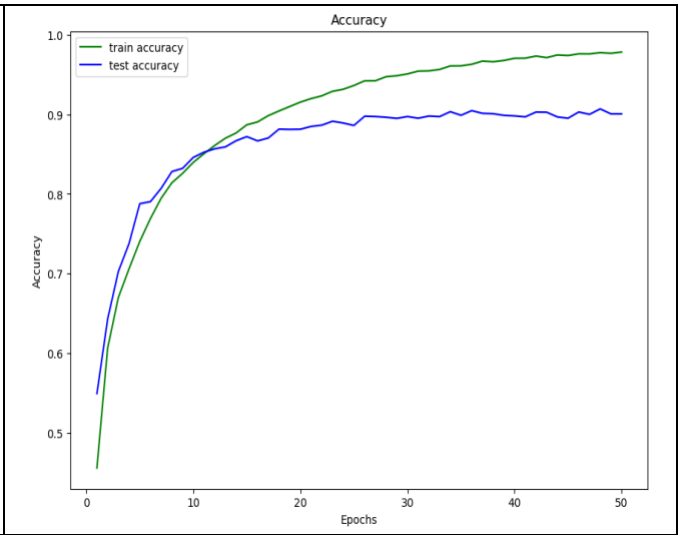

train
test
Acc Value
epoch

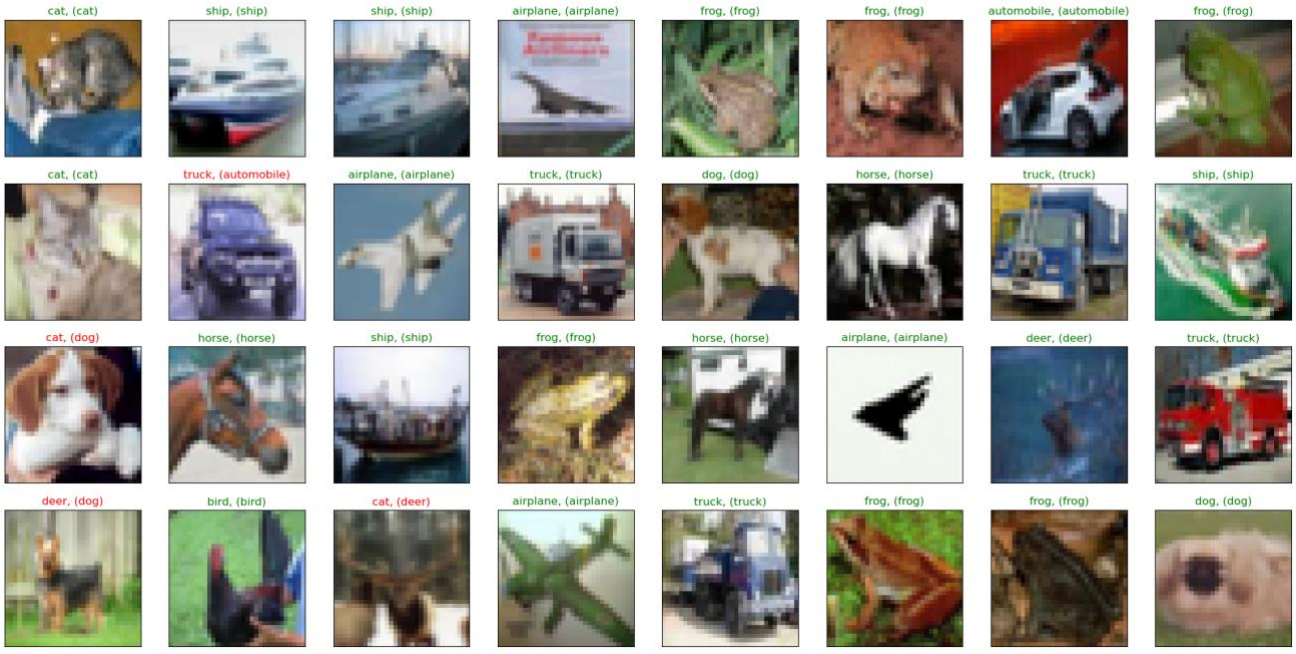

cat, (cat)
ship, (ship)
ship, (ship)
airplane, (airplane)
frog, (frog)
frog, (frog)
truck, (automobile)
frog, (frog)
cat, (cat)
automobile, (automobile)
airplane, (airplane)
truck, (truck)
dog, (dog)
horse, (horse)
truck, (truck)
ship, (ship)
dog, (dog)
horse, (horse)
ship, (ship)
frog, (frog)
horse, (horse)
airplane, (airplane)
deer, (deer)
truck, (truck)
dog, (dog)
bird, (bird)
deer, (deer)
airplane, (airplane)
truck, (truck)
frog, (frog)
frog, (frog)
dog, (dog)

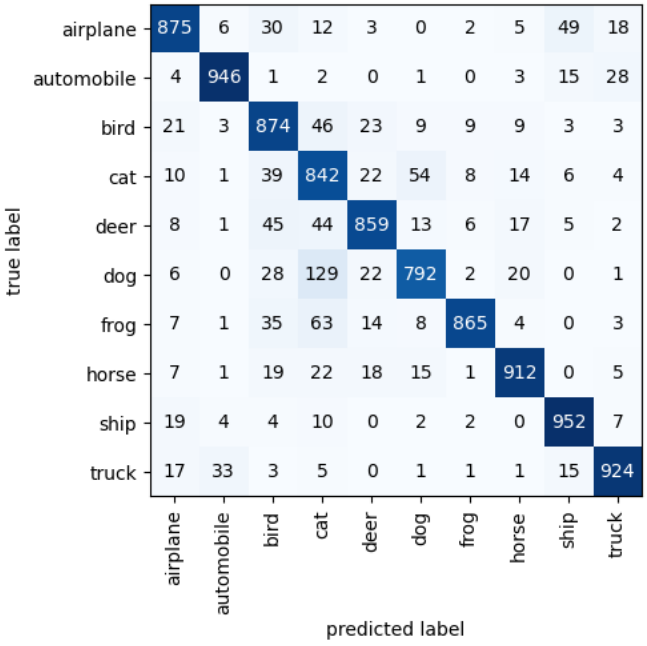

airplane 892 4 20 2 15 2 8 3 33 21
automobile 18 903 1 4 1 3 5 3 14 48
bird 42 0 806 31 42 24 33 12 7 3
cat 19 1 68 657 44 120 40 30 11 10
deer 4 2 31 33 830 30 28 37 4 1
dog 8 0 39 114 33 755 20 27 1 3
frog 9 0 28 24 17 3 916 1 1 1
horse 10 0 19 17 21 45 2 880 0 6
ship 39 8 5 5 8 1 1 3 920 10
truck 27 40 3 6 0 2 5 9 24 884
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
true label
predicted label

DenseNet

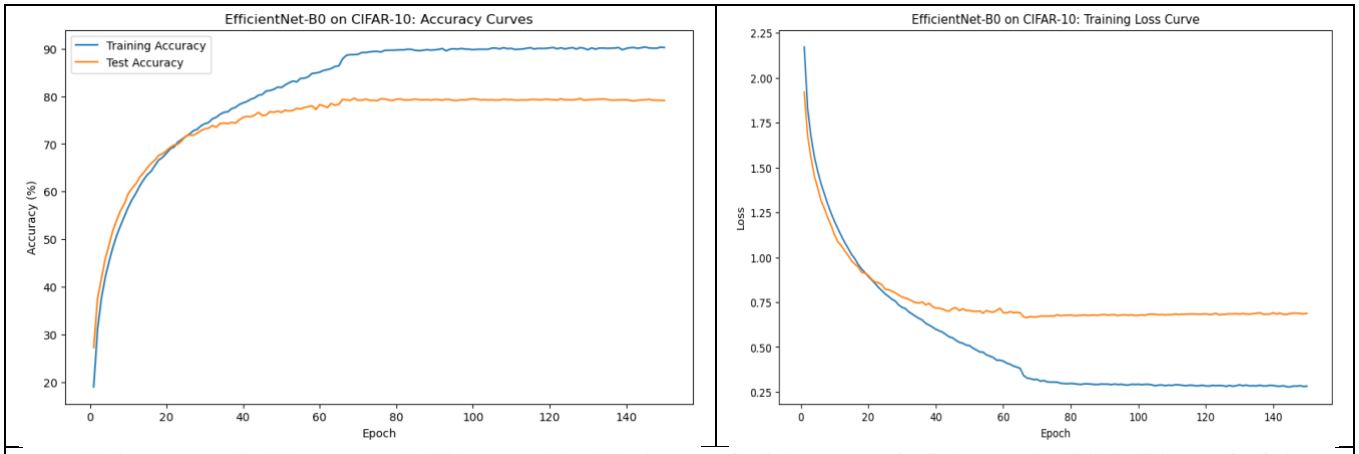

Loss
train loss
test loss
Loss
Epochs

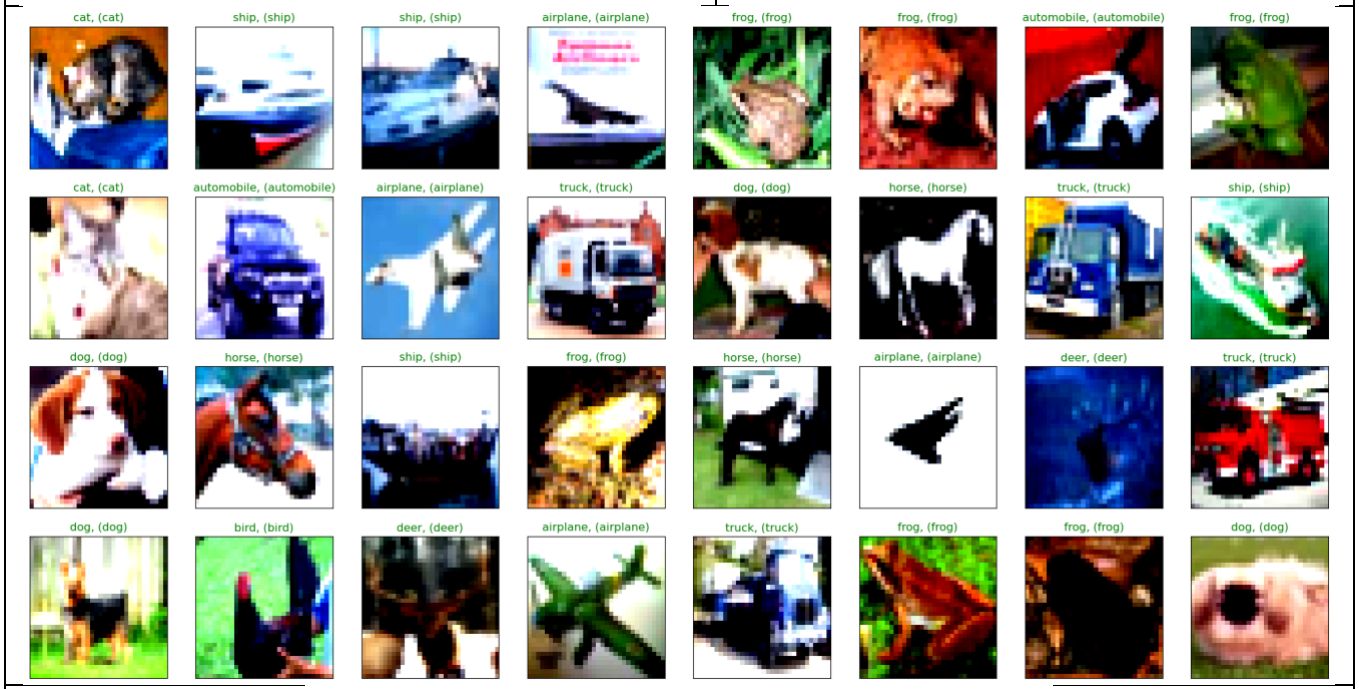

Accuracy
train accuracy
test accuracy
Accuracy
Epochs

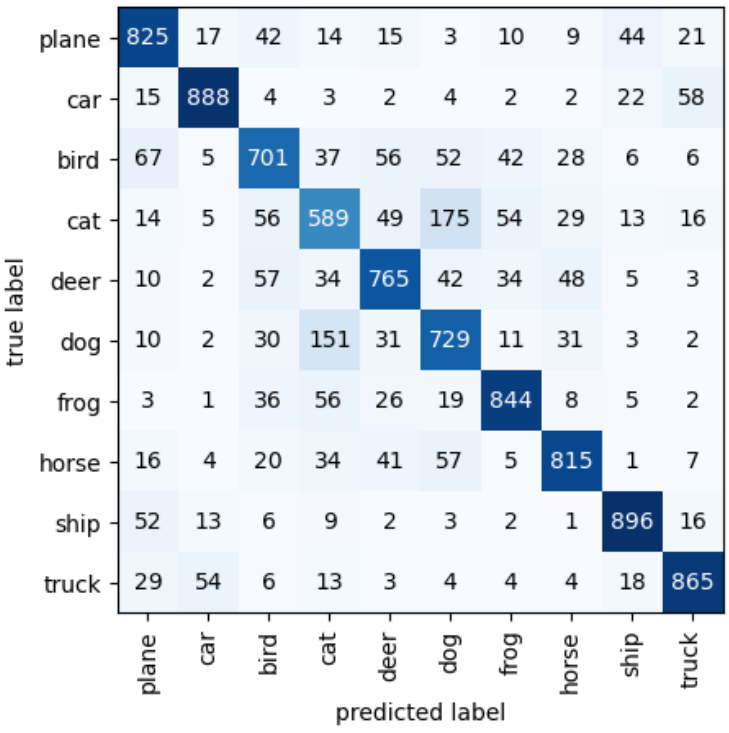

cat, (cat)
ship, (ship)
ship, (ship)
airplane, (airplane)
frog, (frog)
frog, (frog)
automobile, (automobile)
frog, (frog)
cat, (cat)
truck, (automobile)
airplane, (airplane)
truck, (truck)
dog, (dog)
horse, (horse)
truck, (truck)
ship, (ship)
cat, (dog)
horse, (horse)
ship, (ship)
frog, (frog)
horse, (horse)
airplane, (airplane)
deer, (deer)
truck, (truck)
deer, (dog)
bird, (bird)
cat, (deer)
airplane, (airplane)
truck, (truck)
frog, (frog)
frog, (frog)
dog, (dog)


airplane 875 6 30 12 3 0 2 5 49 18
automobile 4 946 1 2 0 1 0 3 15 28
bird 21 3 874 46 23 9 9 9 3 3
cat 10 1 39 842 22 54 8 14 6 4
deer 8 1 45 44 859 13 6 17 5 2
dog 6 0 28 129 22 792 2 20 0 1
frog 7 1 35 63 14 8 865 4 0 3
horse 7 1 19 22 18 15 1 912 0 5
ship 19 4 4 10 0 2 2 0 952 7
truck 17 33 3 5 0 1 1 1 15 924
true label
airplane automobile bird cat deer dog frog horse ship truck
predicted label

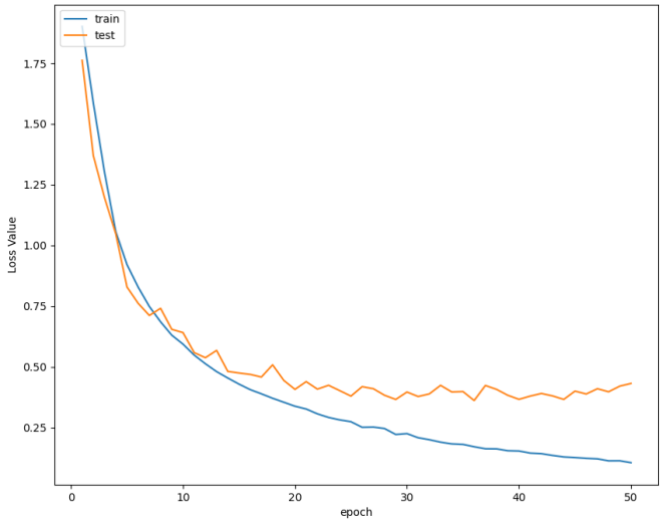
EfficientNet – B0

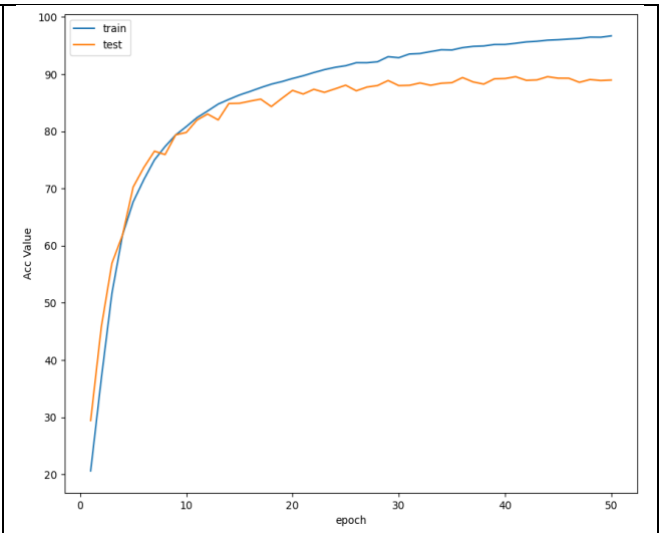


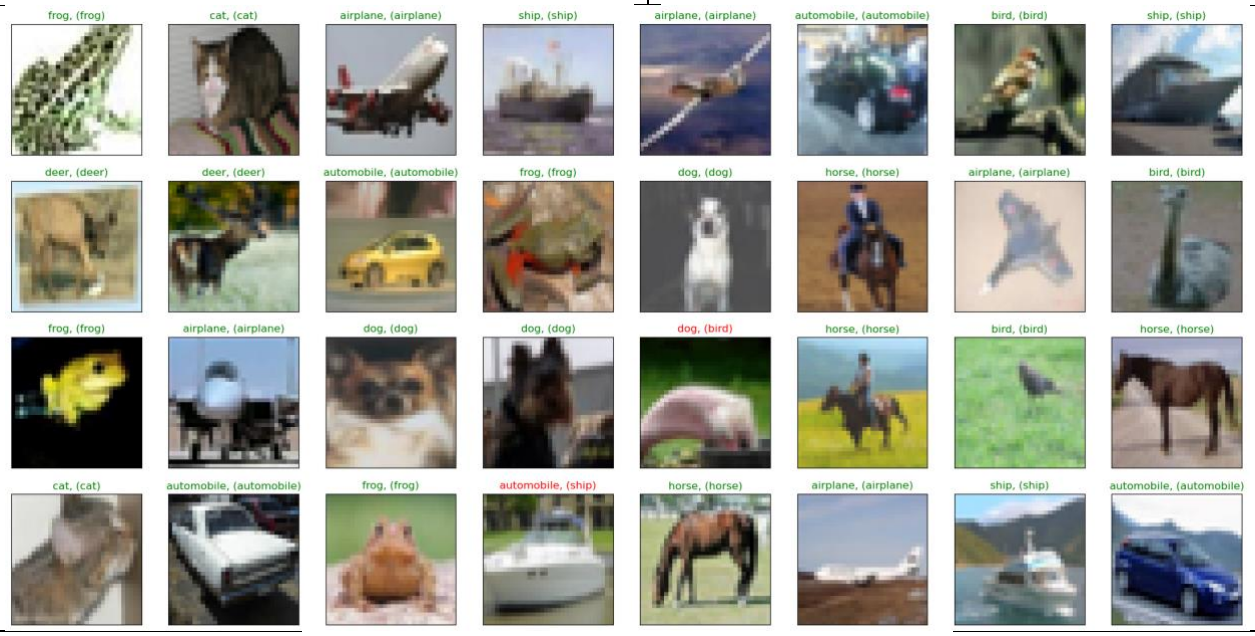


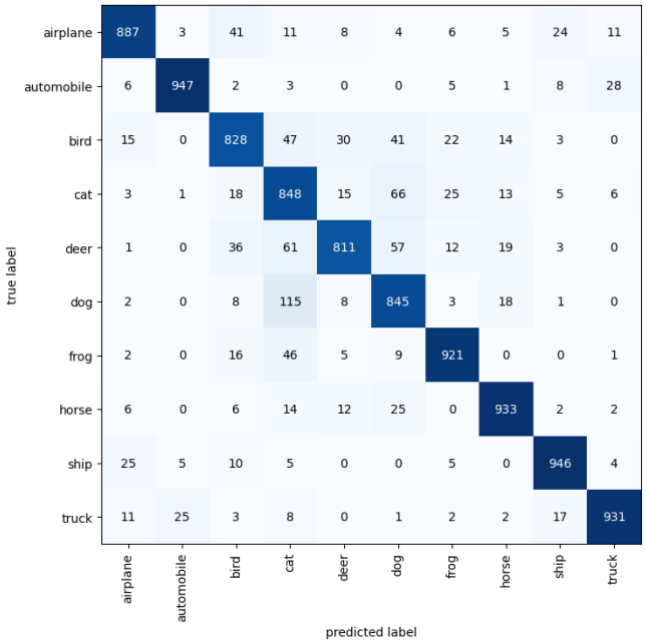


VGG-16

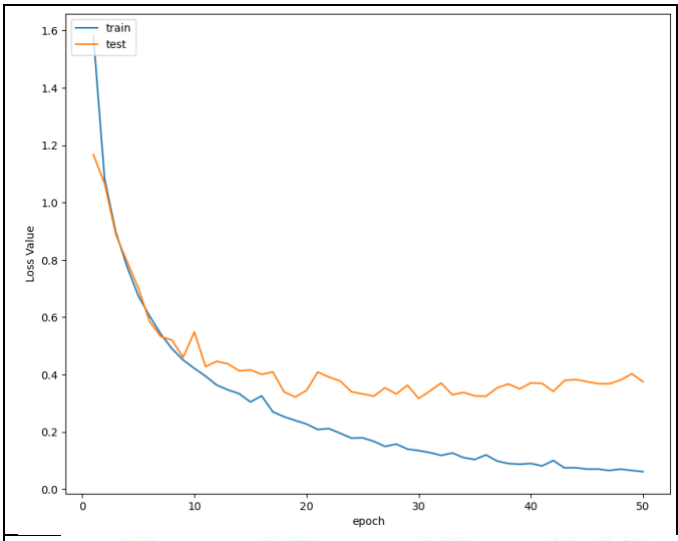

train
test
Loss Value
epoch

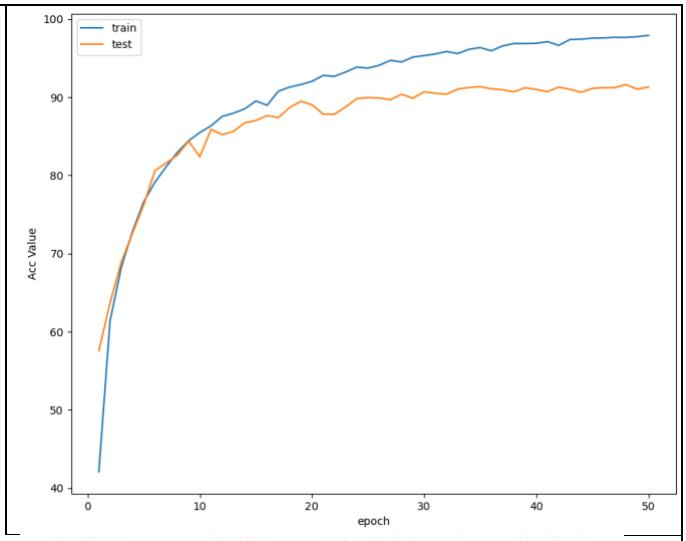

train
test
Acc Value
epoch

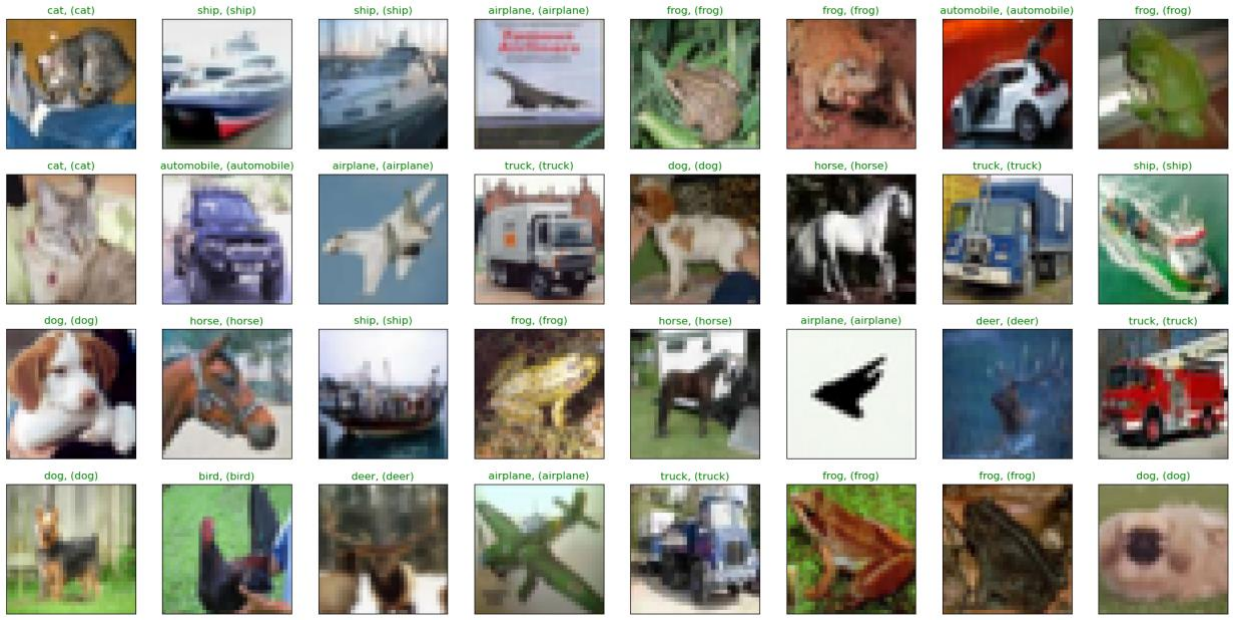

frog, (frog)
cat, (cat)
airplane, (airplane)
ship, (ship)
airplane, (airplane)
automobile, (automobile)
bird, (bird)
ship, (ship)
deer, (deer)
deer, (deer)
automobile, (automobile)
frog, (frog)
dog, (dog)
horse, (horse)
airplane, (airplane)
bird, (bird)
frog, (frog)
airplane, (airplane)
dog, (dog)
dog, (dog)
dog, (bird)
horse, (horse)
bird, (bird)
horse, (horse)
cat, (cat)
automobile, (automobile)
frog, (frog)
automobile, (ship)
horse, (horse)
airplane, (airplane)
ship, (ship)
automobile, (automobile)

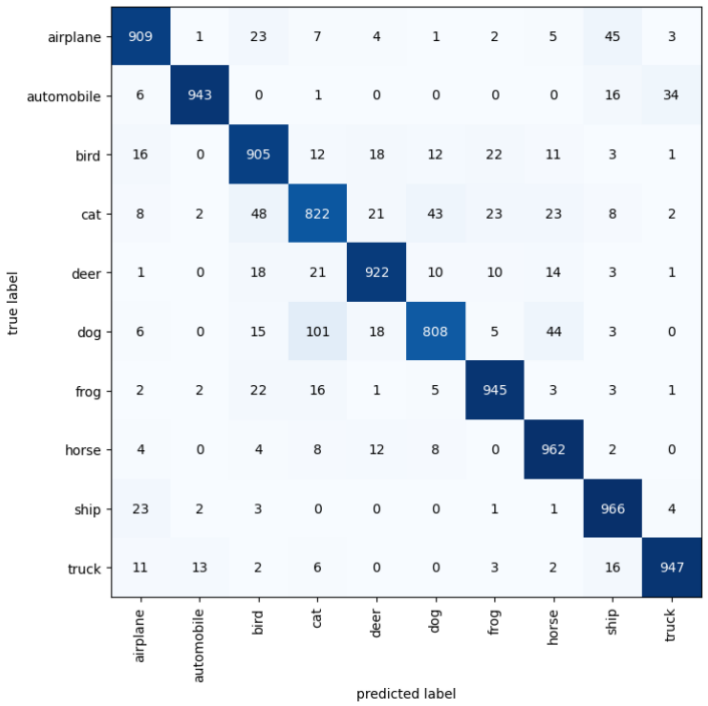

airplane 887 3 41 11 8 4 6 5 24 11
automobile 6 947 2 3 0 0 5 1 8 28
bird 15 0 828 47 30 41 22 14 3 0
cat 3 1 18 848 15 66 25 13 5 6
deer 1 0 36 61 811 57 12 19 3 0
dog 2 0 8 115 8 845 3 18 1 0
frog 2 0 16 46 5 9 921 0 0 1
horse 6 0 6 14 12 25 0 933 2 2
ship 25 5 10 5 0 0 5 0 946 4
truck 11 25 3 8 0 1 2 2 17 931
true label
airplane automobile bird cat deer dog frog horse ship truck
predicted label

ResNet50

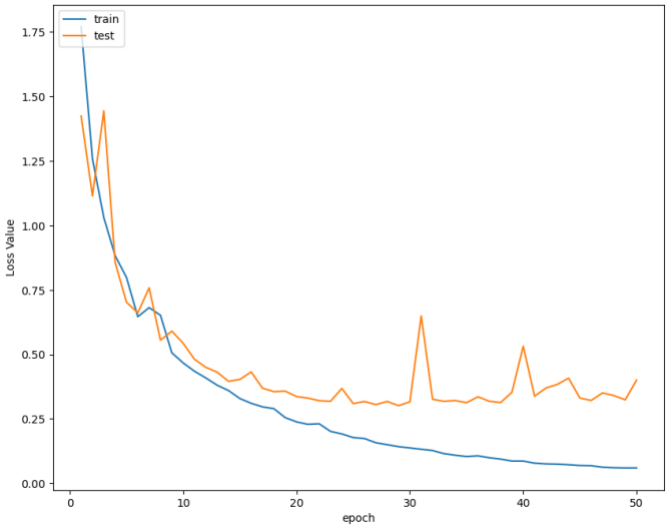

train
test
Loss Value
epoch

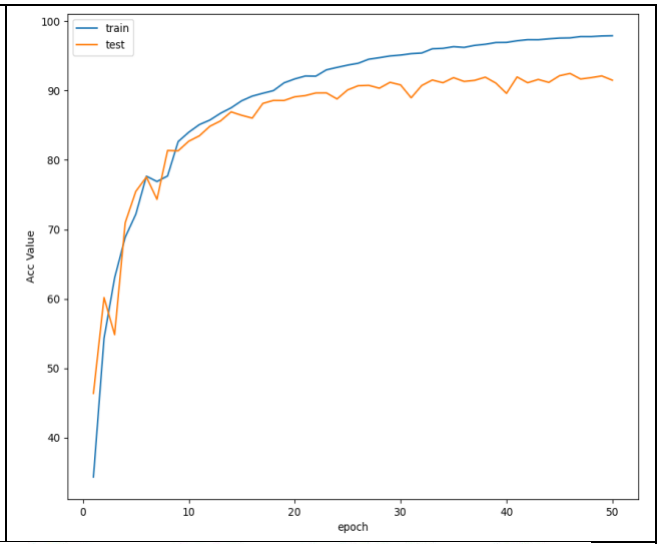

train
test
Acc Value
epoch

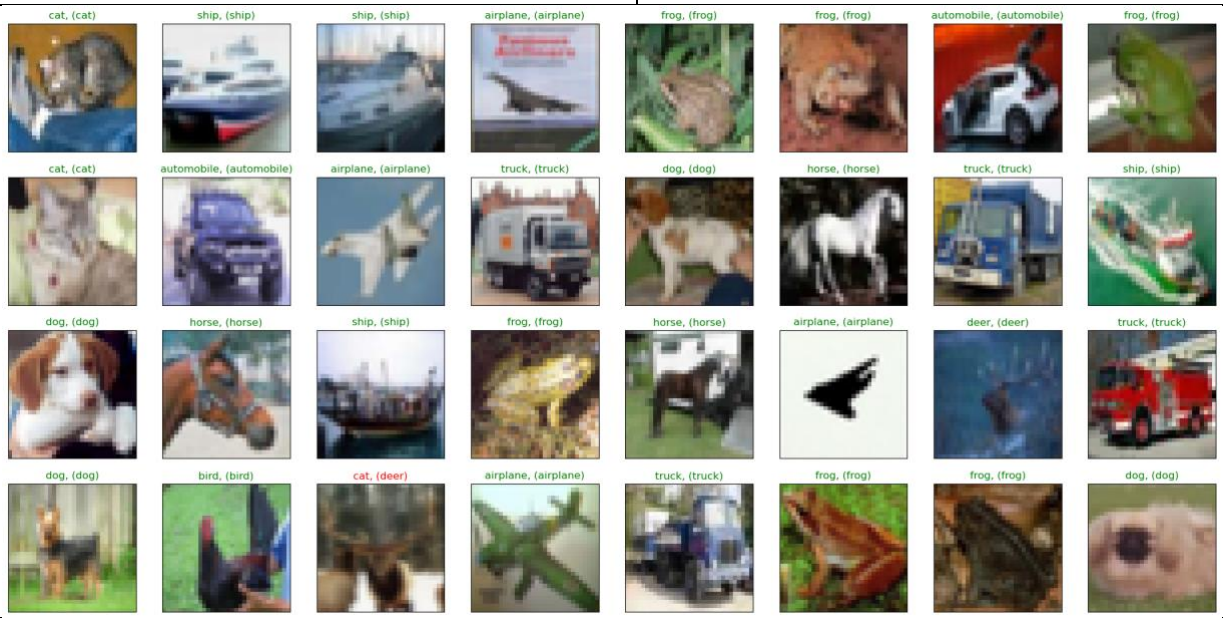

cat, (cat)
ship, (ship)
ship, (ship)
airplane, (airplane)
frog, (frog)
frog, (frog)
automobile, (automobile)
frog, (frog)
cat, (cat)
automobile, (automobile)
airplane, (airplane)
truck, (truck)
dog, (dog)
horse, (horse)
truck, (truck)
ship, (ship)
dog, (dog)
horse, (horse)
ship, (ship)
frog, (frog)
horse, (horse)
airplane, (airplane)
deer, (deer)
truck, (truck)
dog, (dog)
bird, (bird)
deer, (deer)
airplane, (airplane)
truck, (truck)
frog, (frog)
frog, (frog)
dog, (dog)

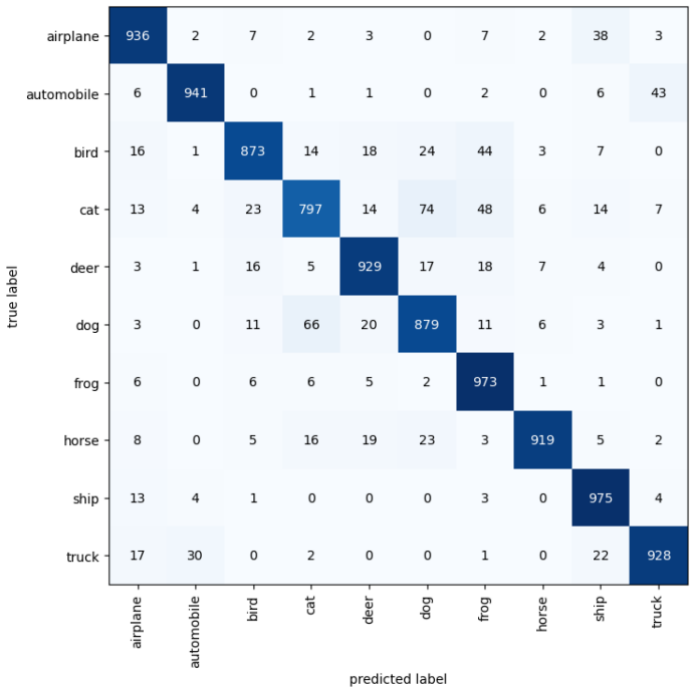

airplane 909 1 23 7 4 1 2 5 45 3
automobile 6 943 0 1 0 0 0 0 16 34
bird 16 0 905 12 18 12 22 11 3 1
cat 8 2 48 822 21 43 23 23 8 2
deer 1 0 18 21 922 10 10 14 3 1
dog 6 0 15 101 18 808 5 44 3 0
frog 2 2 22 16 1 5 945 3 3 1
horse 4 0 4 8 12 8 0 962 2 0
ship 23 2 3 0 0 0 1 1 966 4
truck 11 13 2 6 0 0 3 2 16 947
true label
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
predicted label

ResNet101

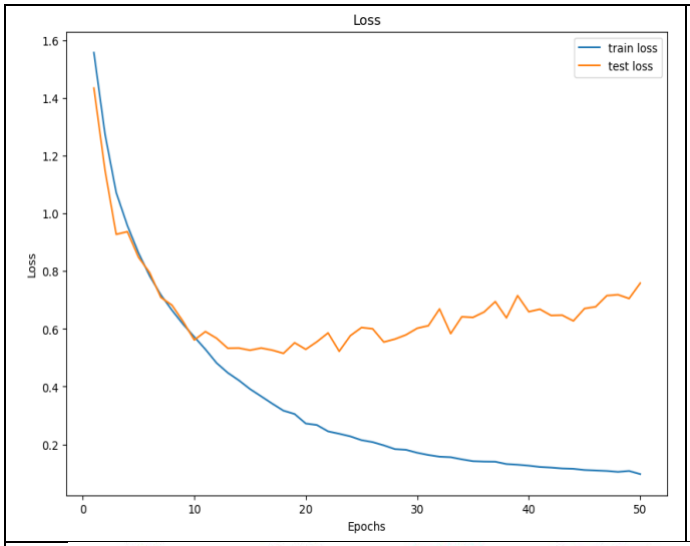

train
test
Loss Value
epoch

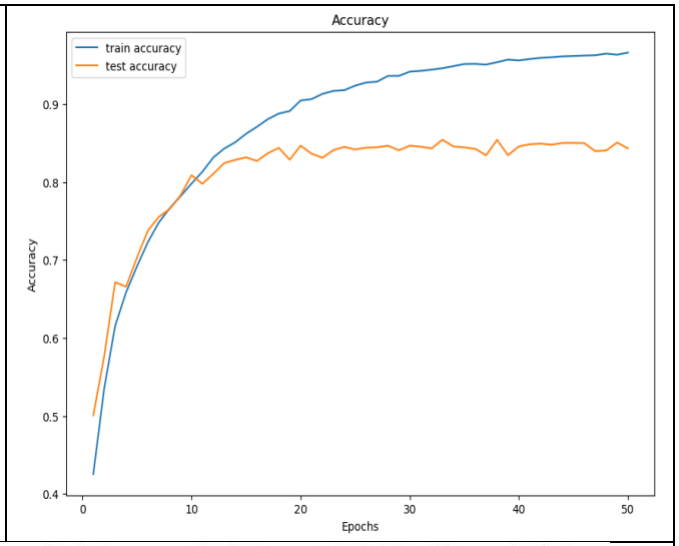

train
test
Acc Value
epoch

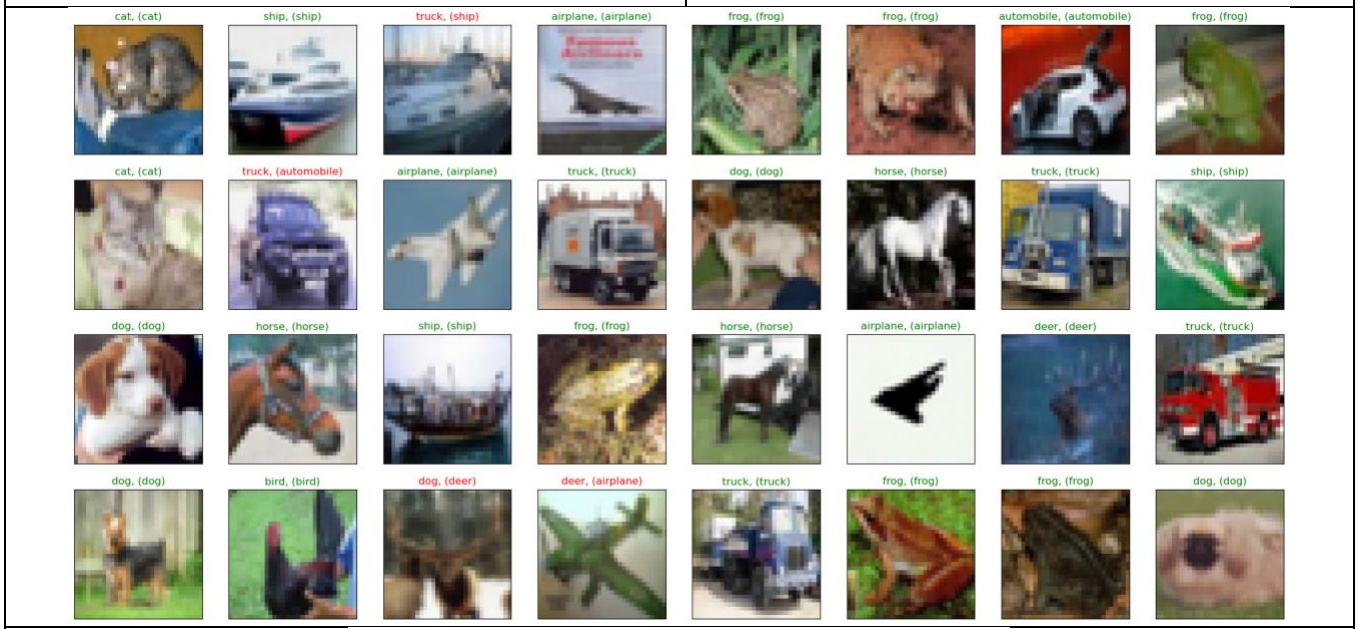

cat, (cat)
ship, (ship)
ship, (ship)
airplane, (airplane)
frog, (frog)
frog, (frog)
automobile, (automobile)
frog, (frog)
cat, (cat)
automobile, (automobile)
airplane, (airplane)
truck, (truck)
dog, (dog)
horse, (horse)
truck, (truck)
ship, (ship)
dog, (dog)
horse, (horse)
ship, (ship)
frog, (frog)
horse, (horse)
airplane, (airplane)
deer, (deer)
truck, (truck)
dog, (dog)
bird, (bird)
cat, (deer)
airplane, (airplane)
truck, (truck)
frog, (frog)
frog, (frog)
dog, (dog)

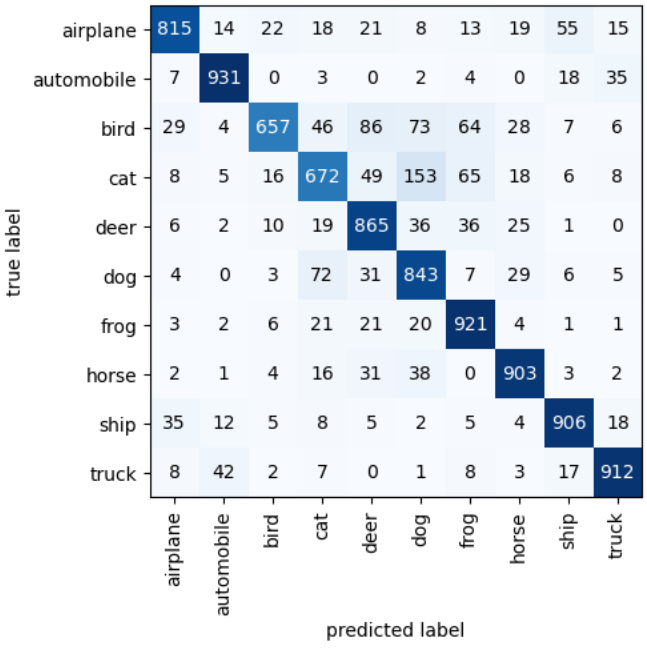

airplane 936 2 7 2 3 0 7 2 38 3
automobile 6 941 0 1 1 0 2 0 6 43
bird 16 1 873 14 18 24 44 3 7 0
cat 13 4 23 797 14 74 48 6 14 7
deer 3 1 16 5 929 17 18 7 4 0
dog 3 0 11 66 20 879 11 6 3 1
frog 6 0 6 6 5 2 973 1 1 0
horse 8 0 5 16 19 23 3 919 5 2
ship 13 4 1 0 0 0 3 0 975 4
truck 17 30 0 2 0 0 1 0 22 928
true label
airplane automobile bird cat deer dog frog horse ship truck
predicted label

DenseNet201



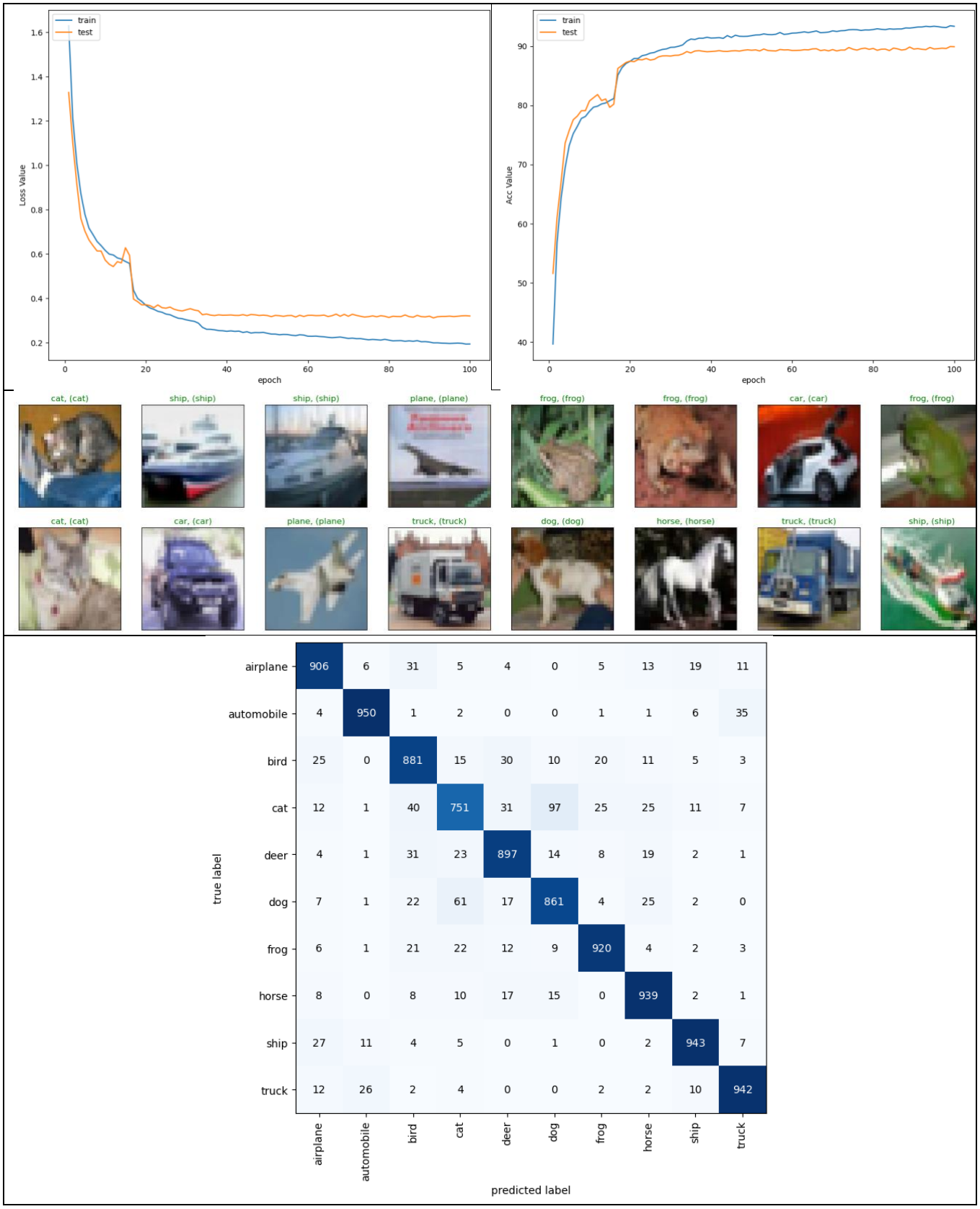




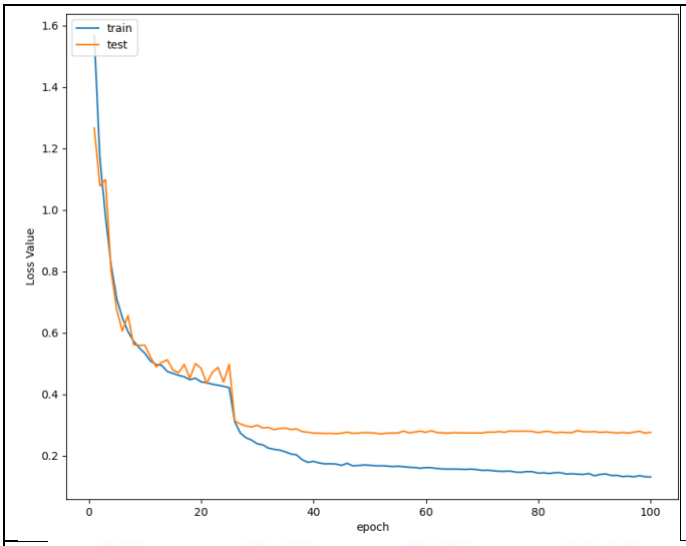


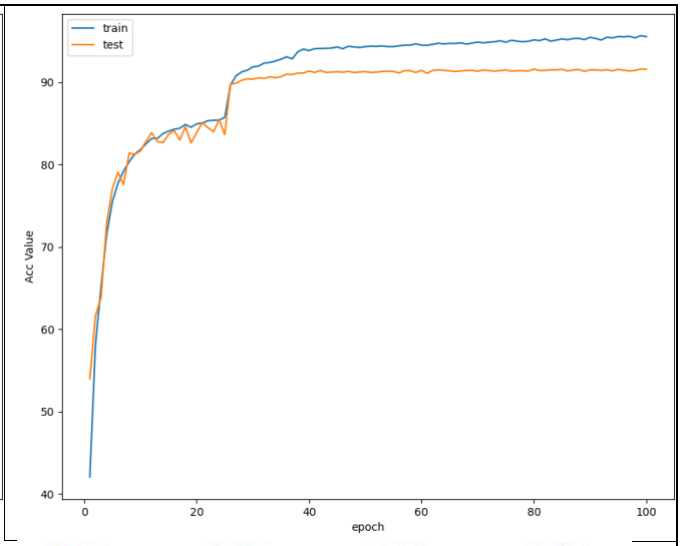
Batch Size

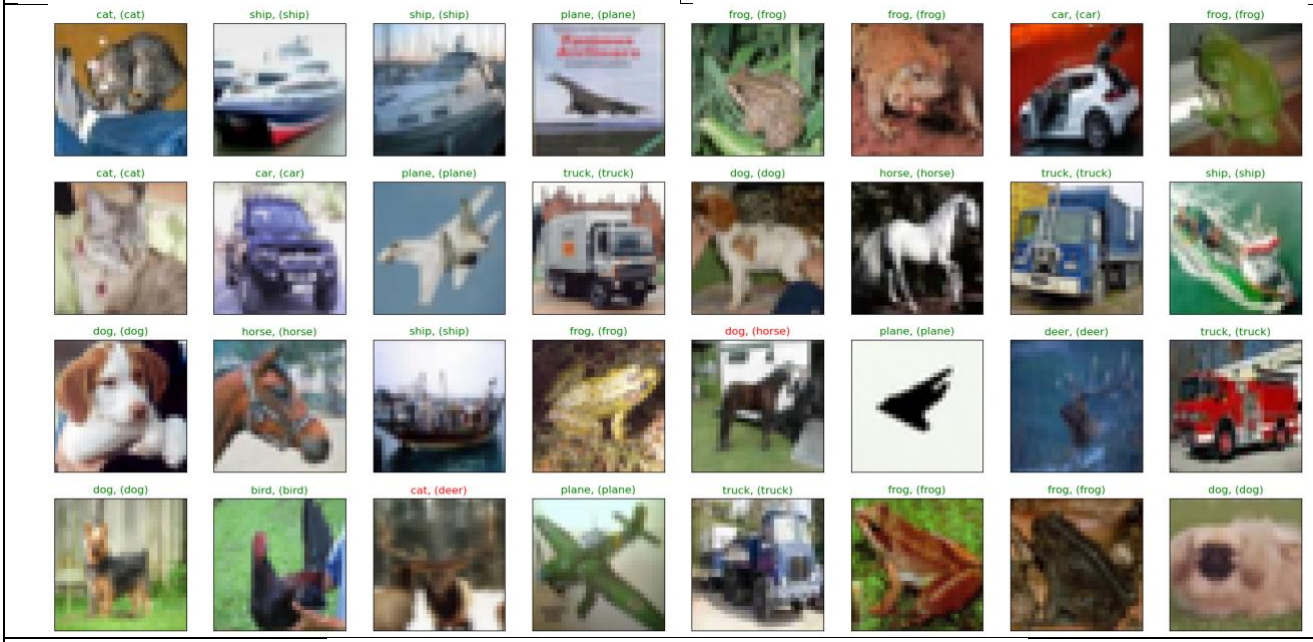
16

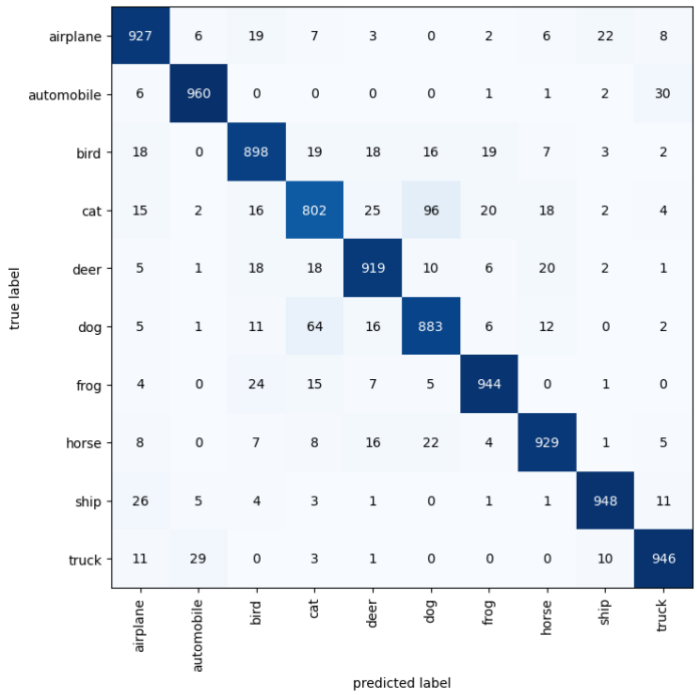


Batch size 32

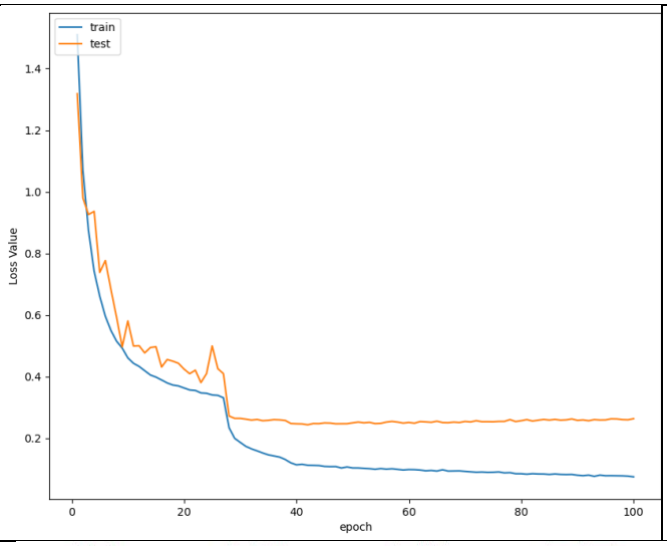


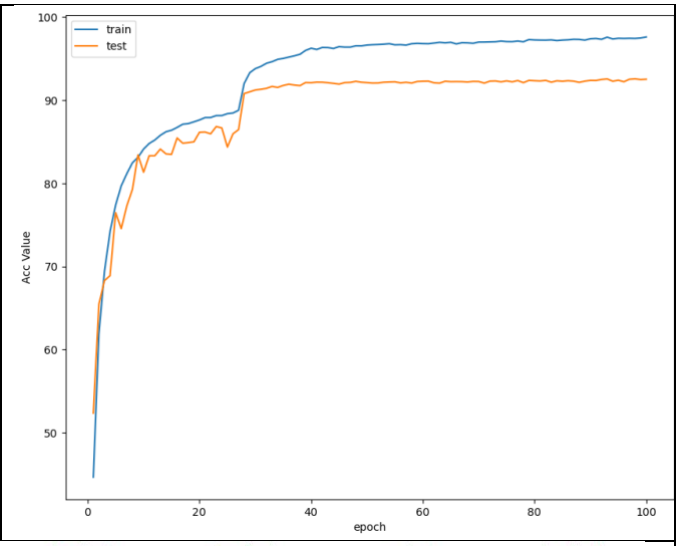


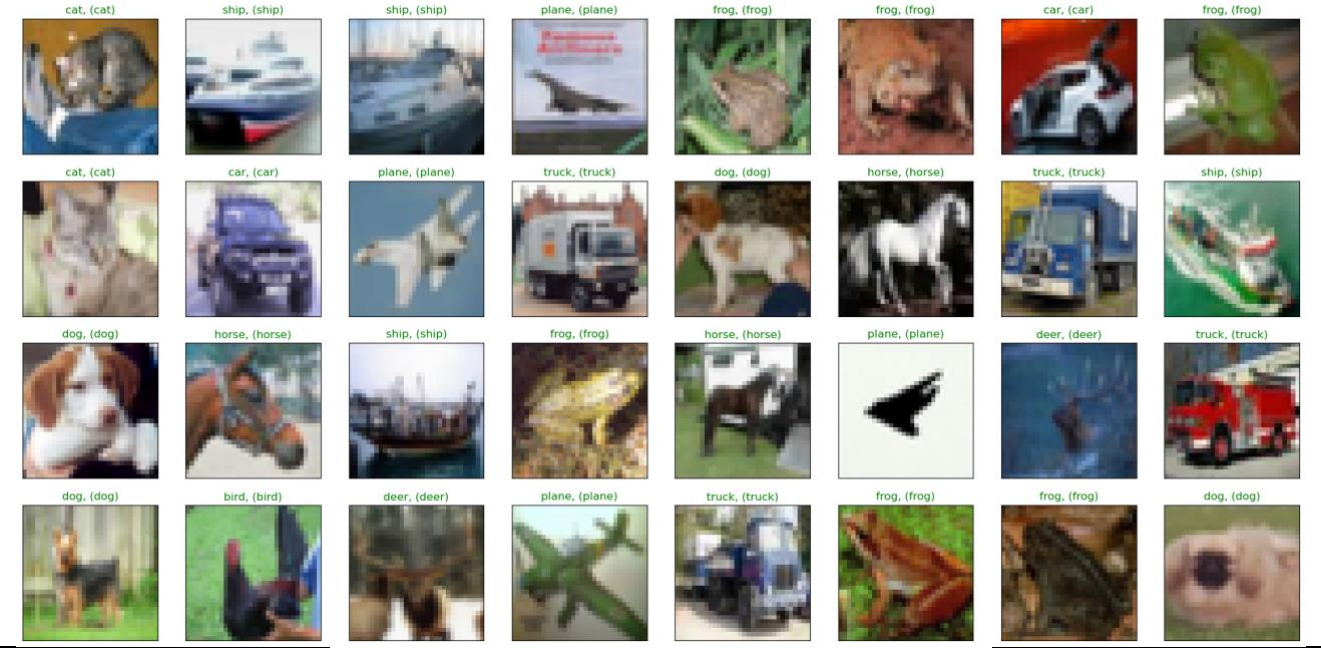


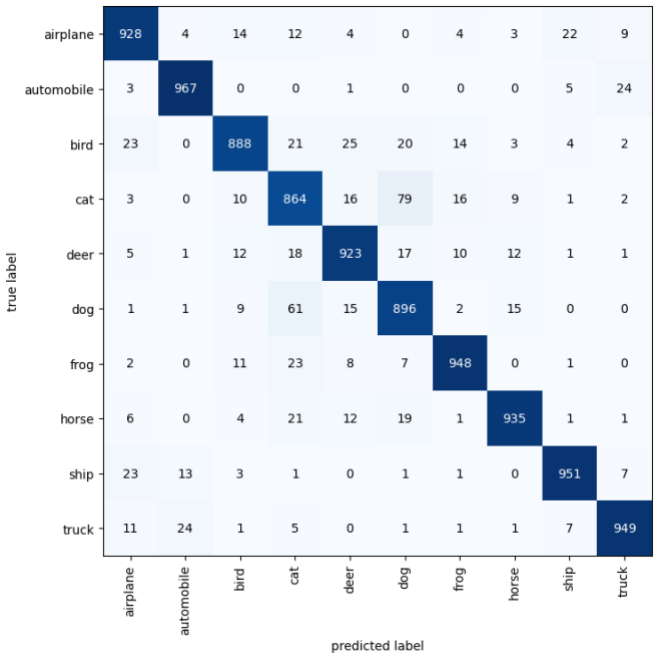


Batch size 64

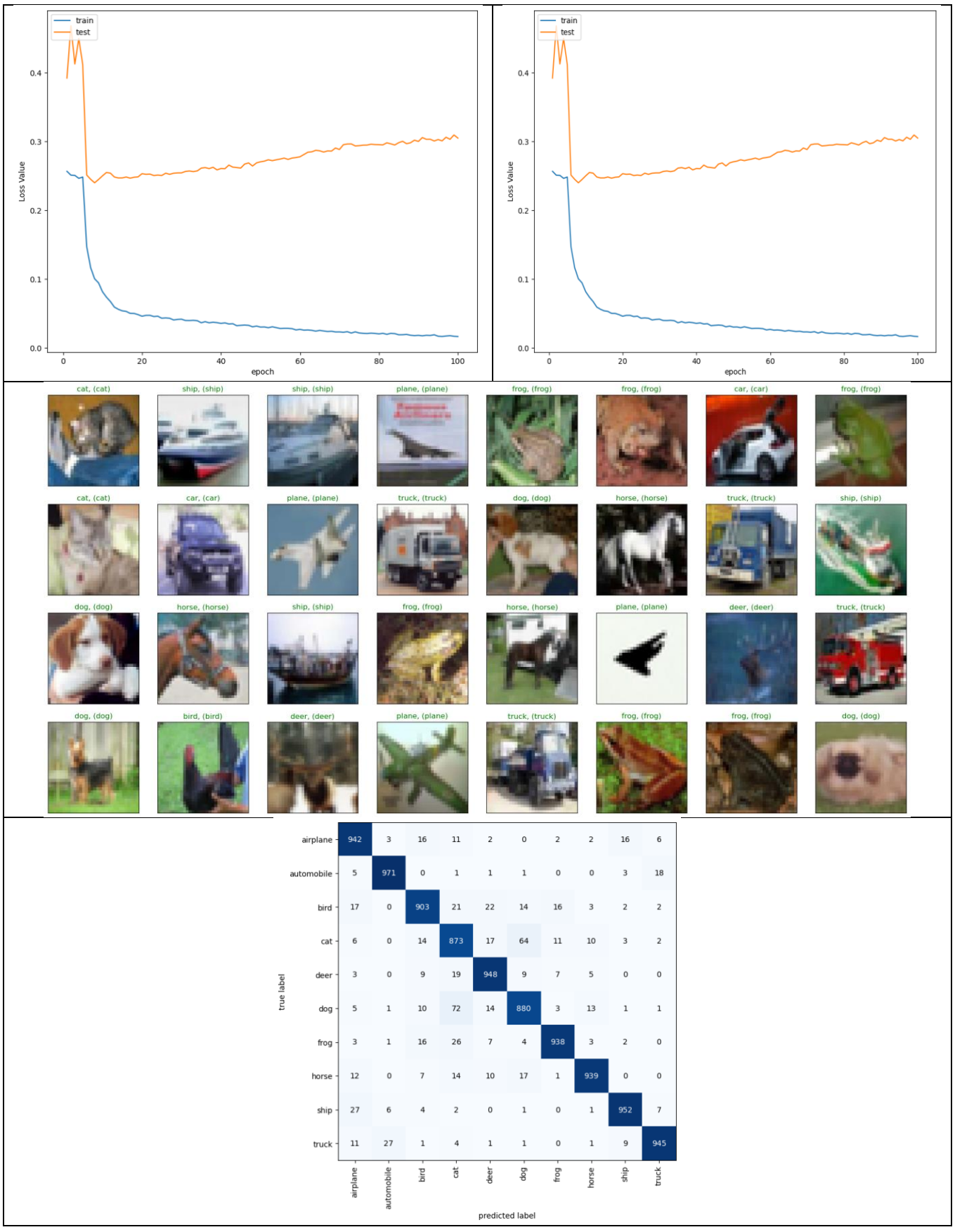




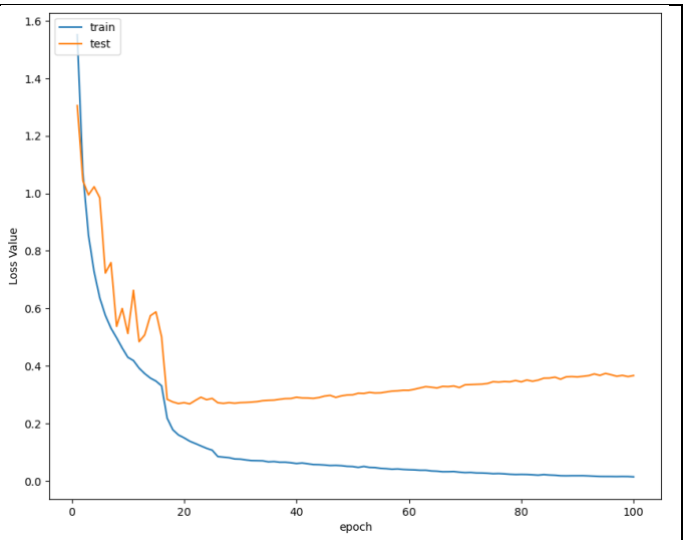


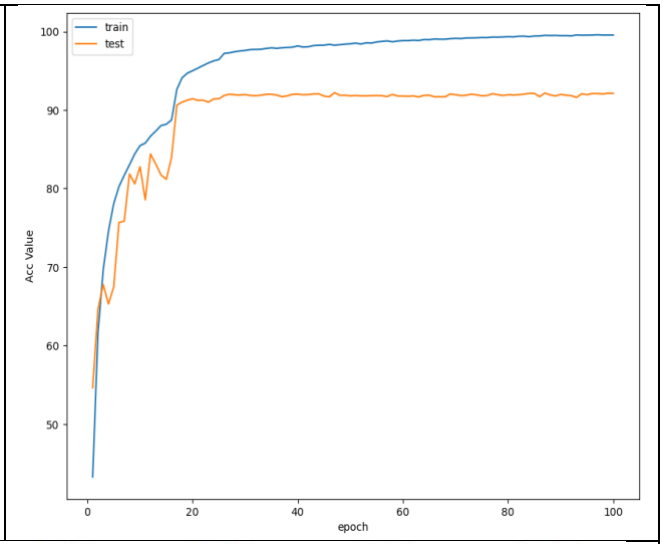


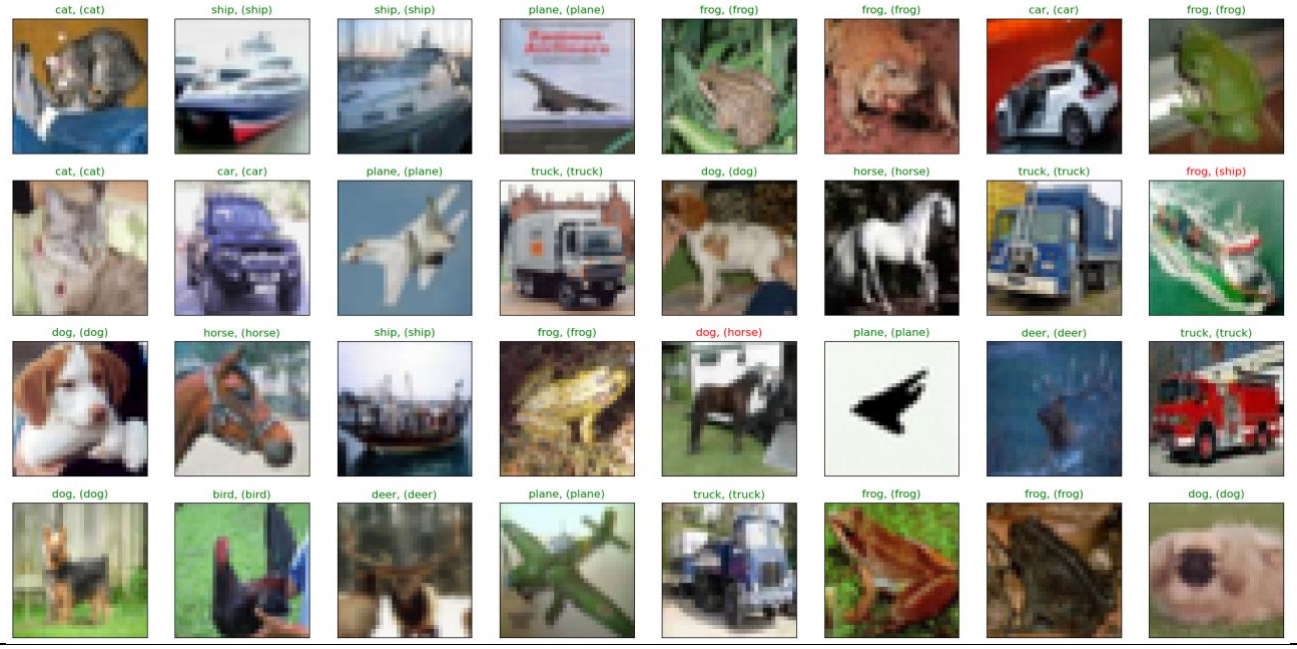
Batch size 128

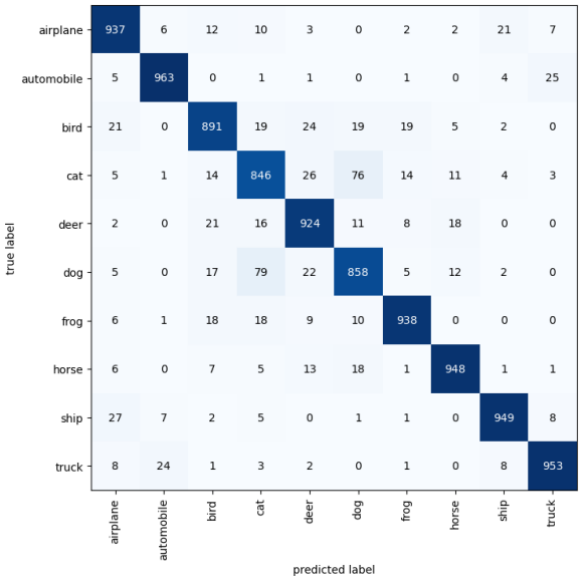


Batch size 256

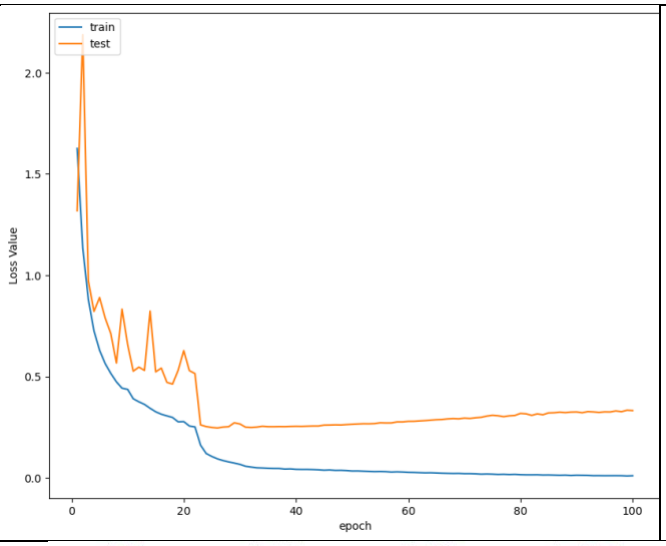


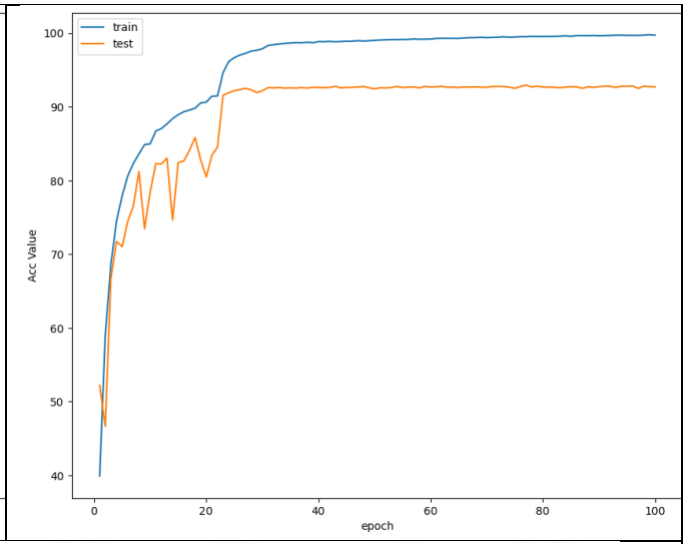


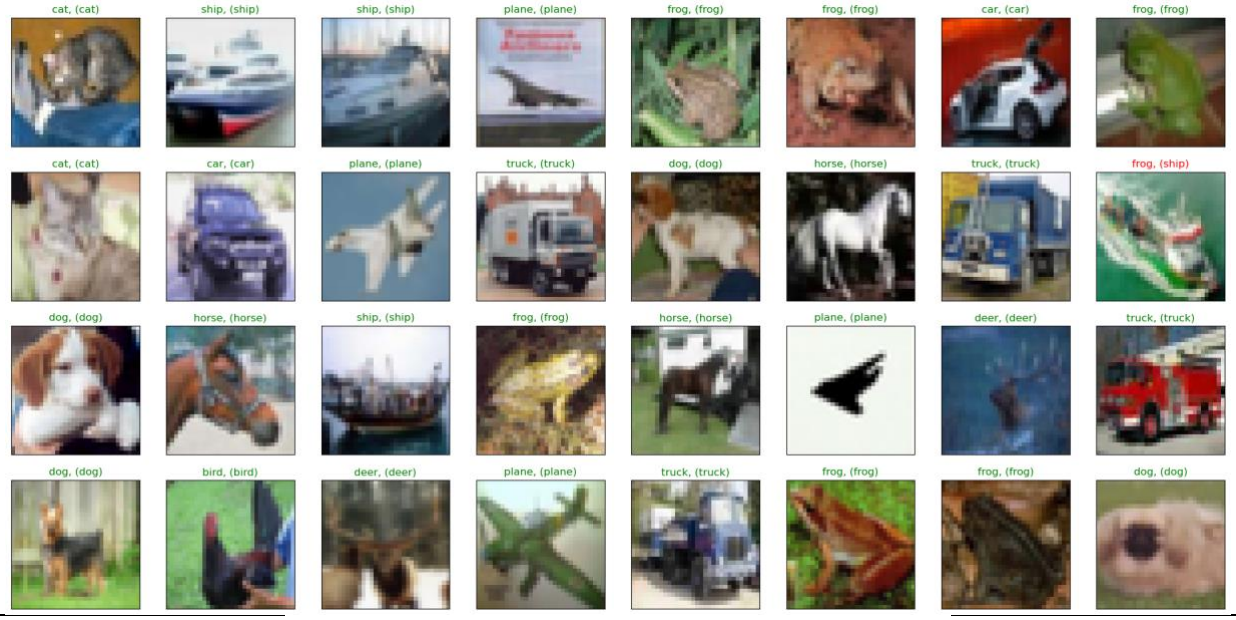


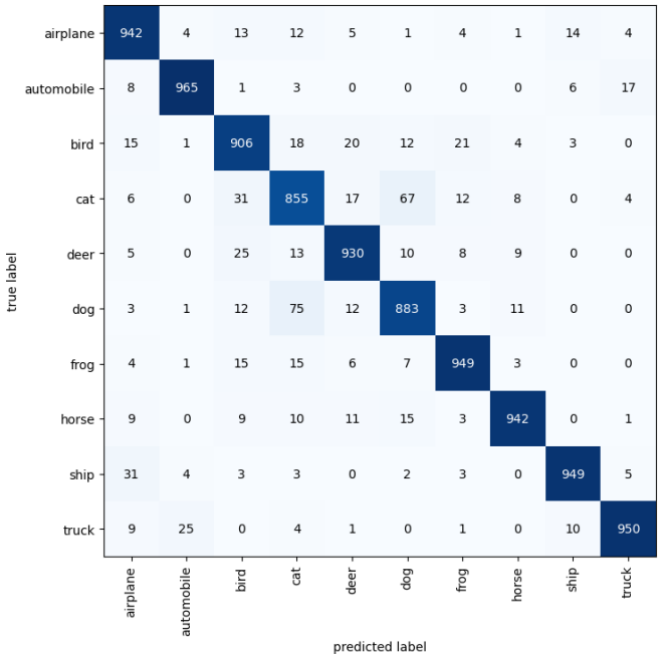


Batch size 512



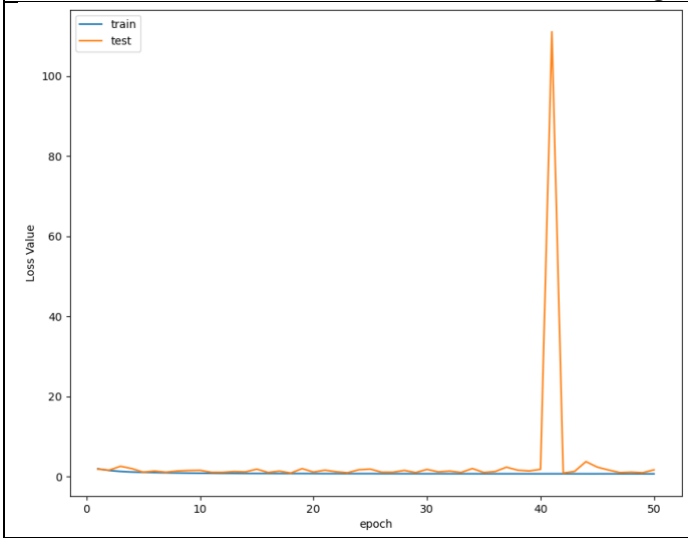


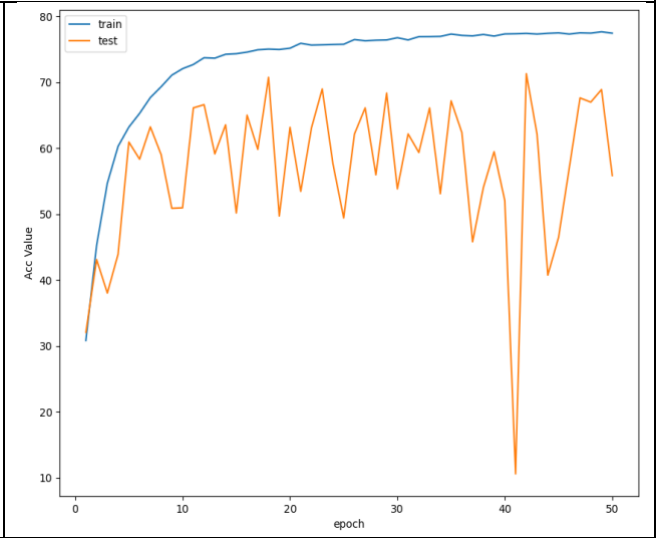




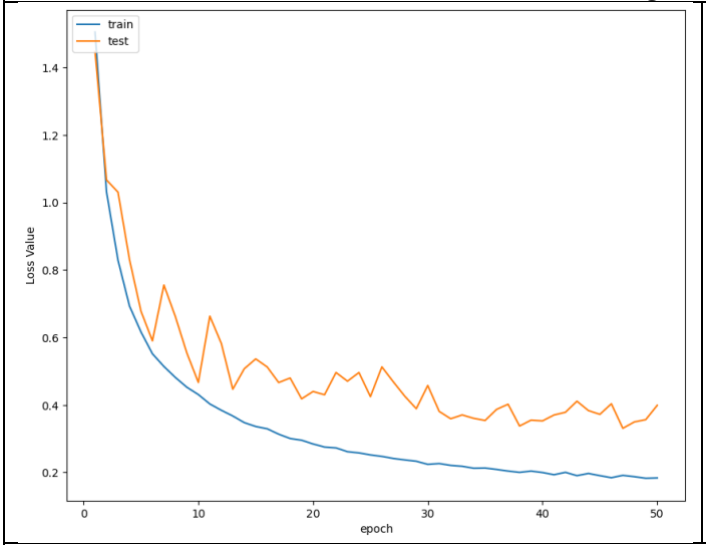
Learning Rate

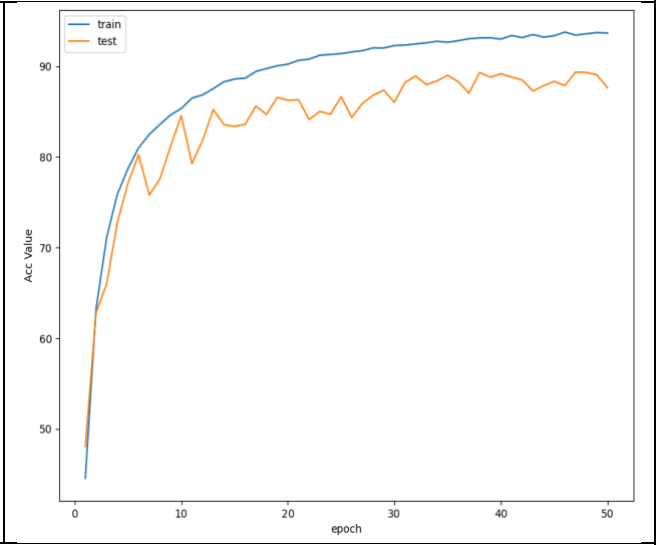
Learning rate – 0.01



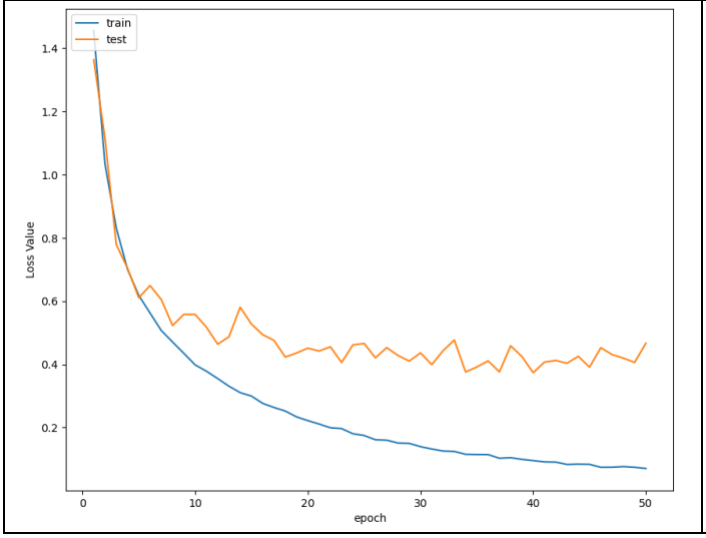


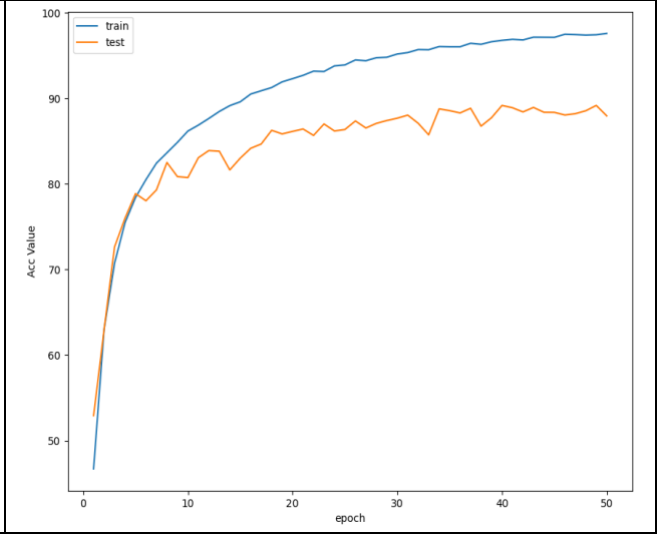
Learning rate – 0.001

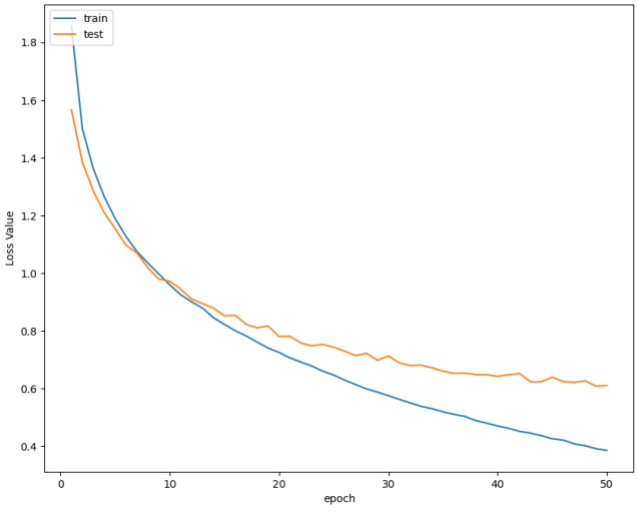
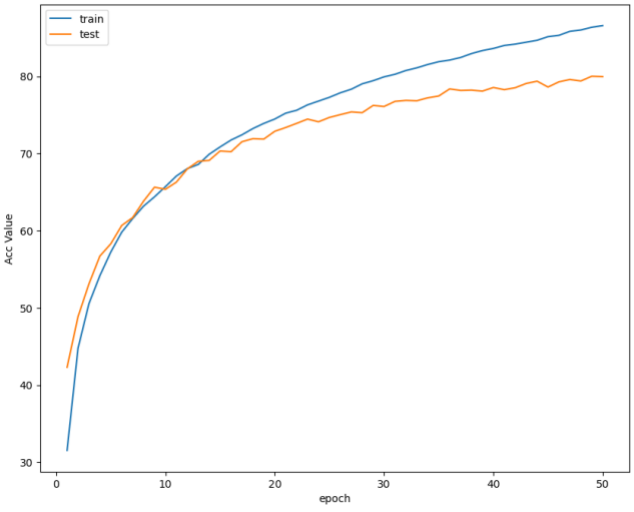




Learning rate – 0.0001

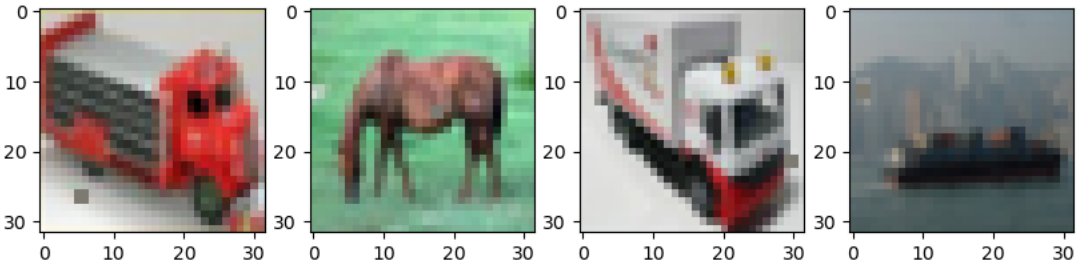
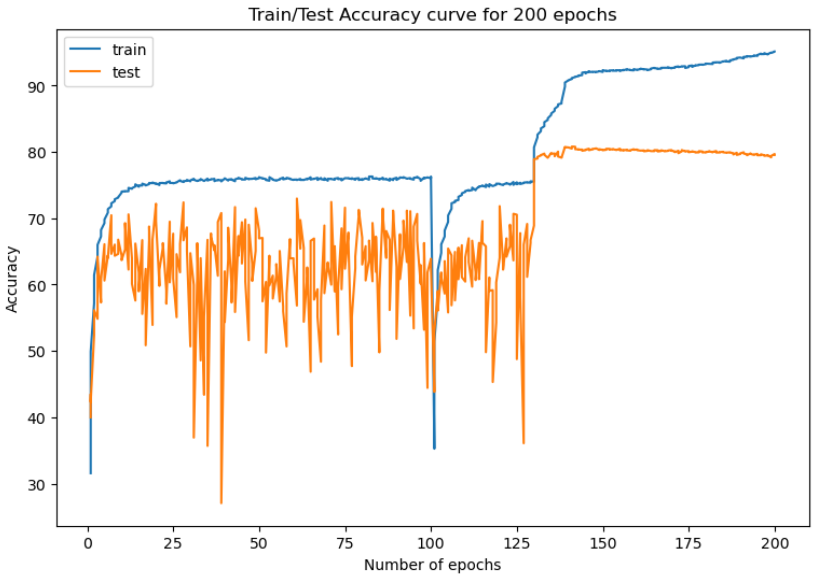
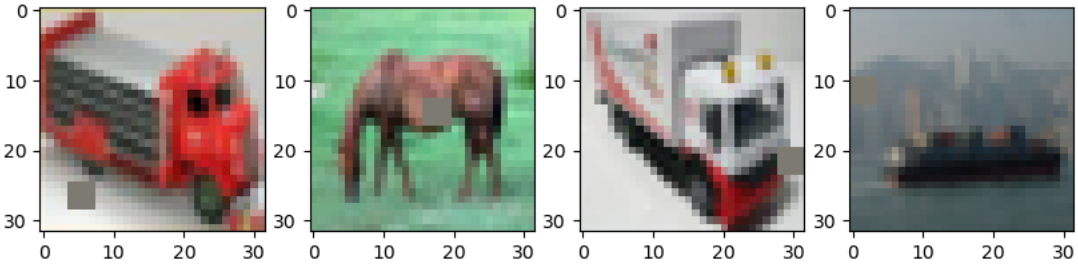
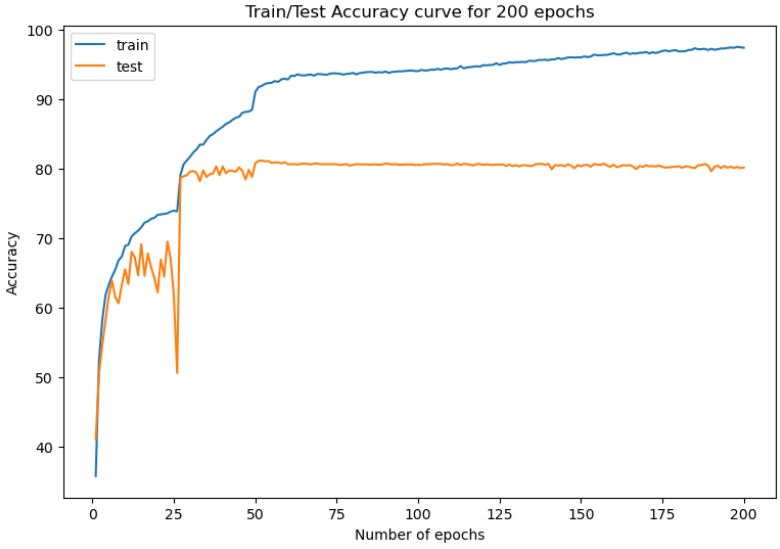




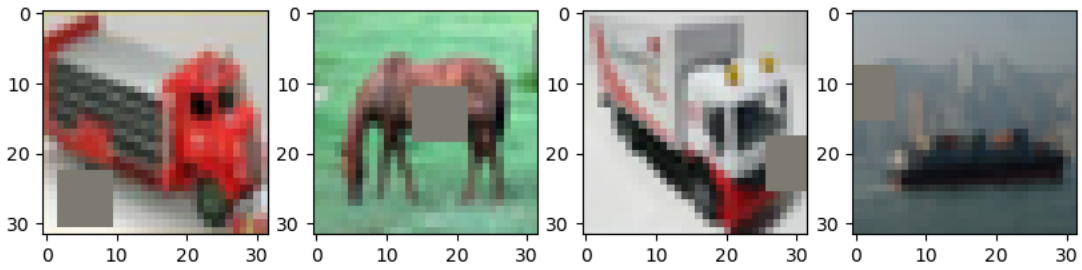
0.00001	
 train test Loss Value epoch	 train test Acc Value epoch

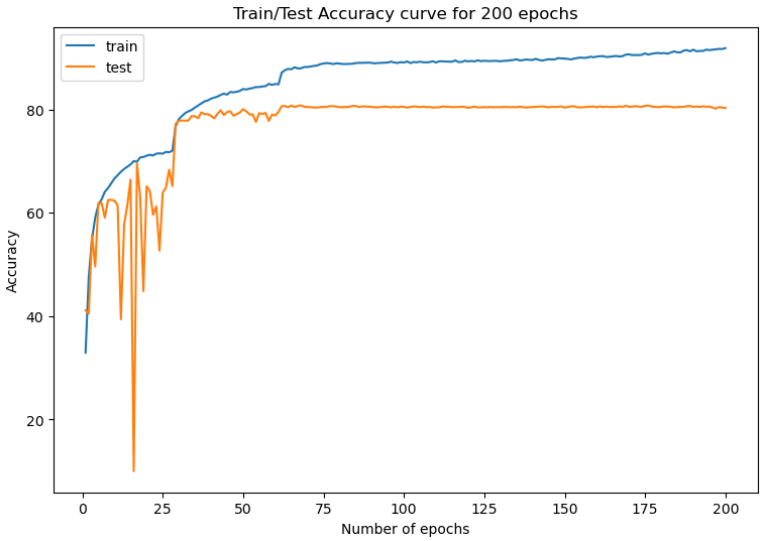
Augmentations

Cutout Size

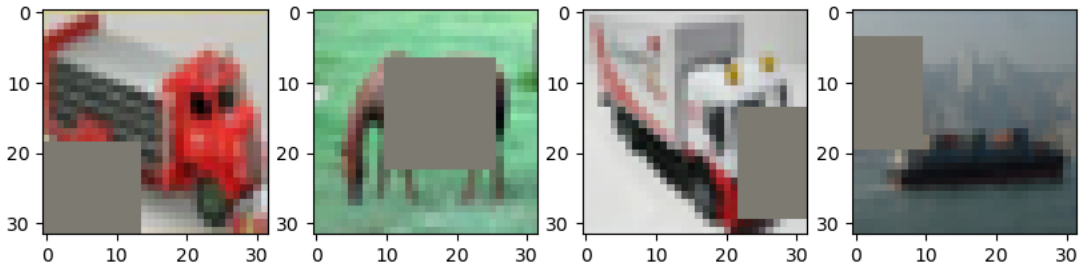
Cutout size – 2x2 pixels


Cutout size – 4x4 pixels



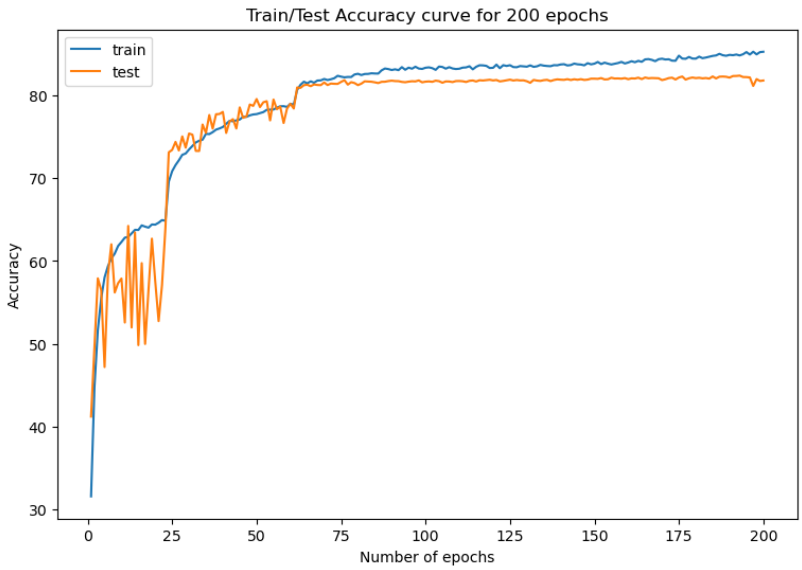
Cutout size – 8x8 pixels





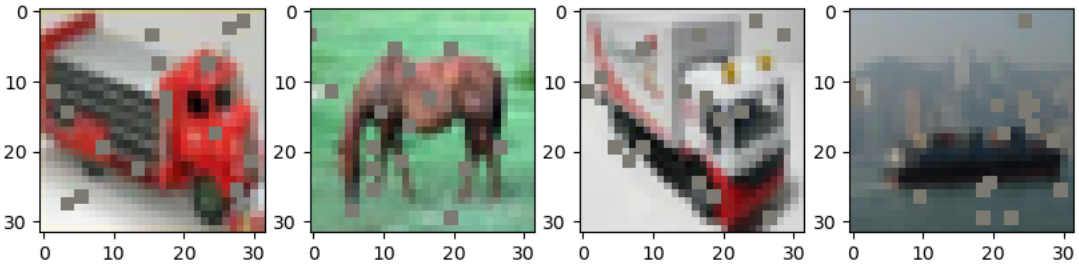
Cutout size – 16x16 pixels

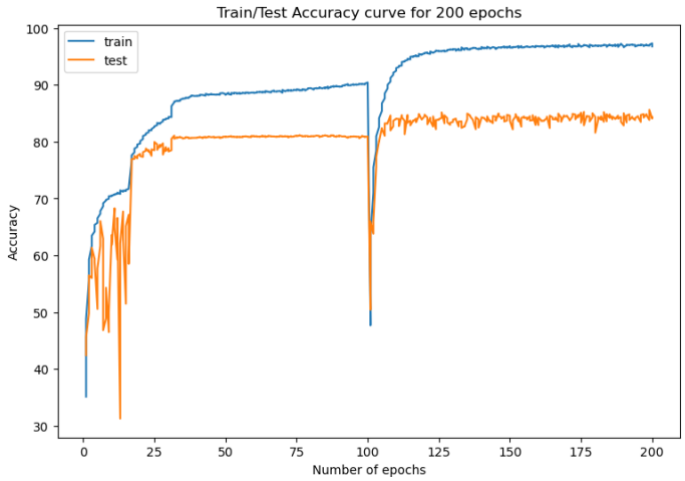




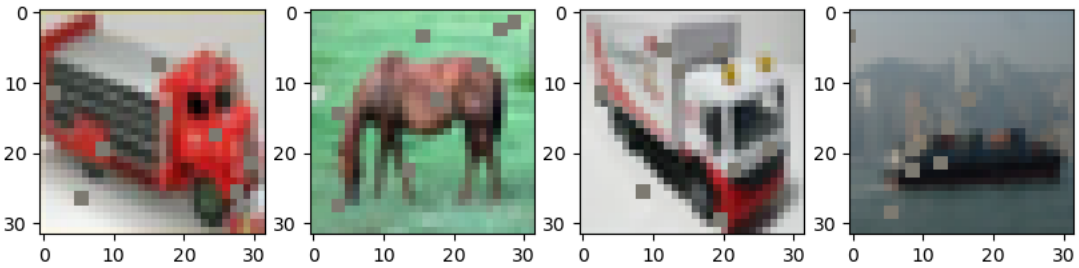
Number of Cutouts

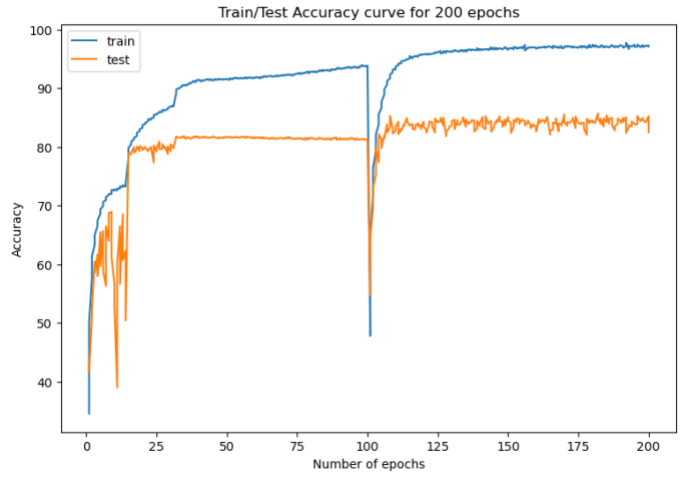
16 Cutouts



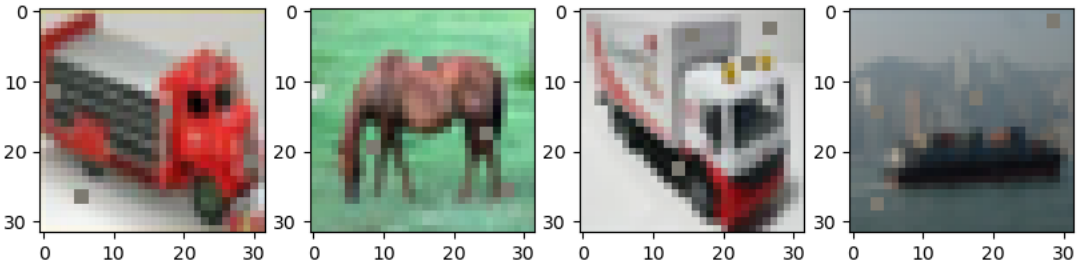


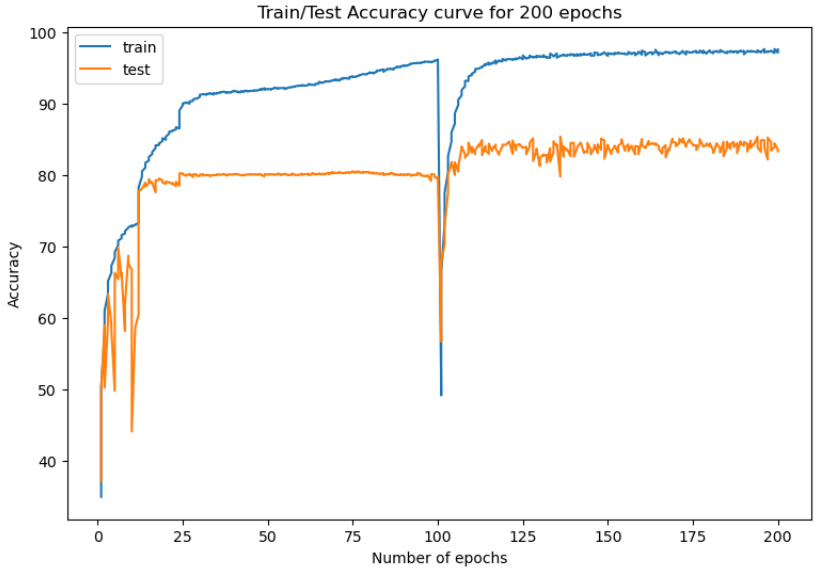
8 Cutouts





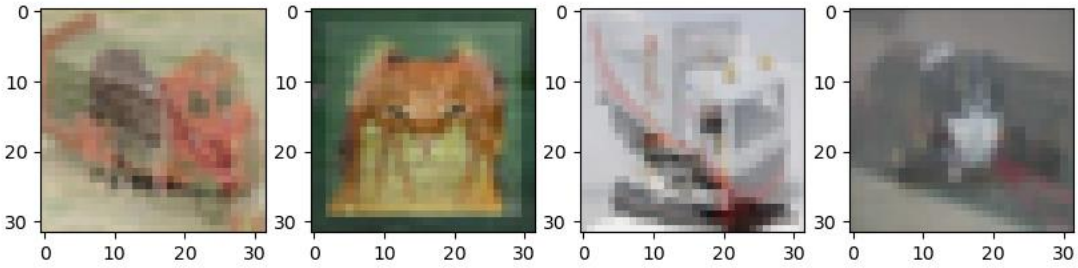
4 Cutouts

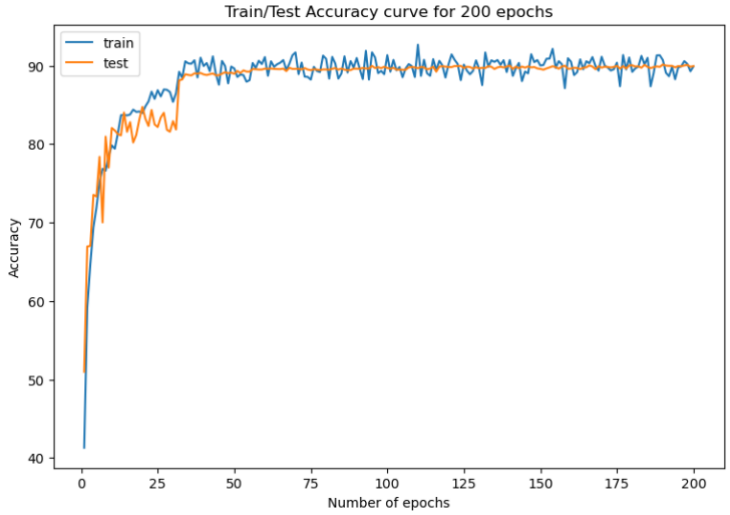



Train/Test Accuracy curve for 200 epochs
train
test
Accuracy
Number of epochs
100
90
80
70
60
50
40
0
25
50
75
100
125
150
175
200

Mix-up

Alpha = 0.2





Alpha = 0.4