HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN I

BỘ MÔN LẬP TRÌNH PYTHON

IMAGE CLASSIFICATION REPORT

Giảng viên hướng dẫn : **KIM NGỌC BÁCH**
Họ và tên sinh viên : **HOÀNG CHÍ HIỂN**
Mã sinh viên : **B23DCCE028**
Lớp : **D23CQCE04-B**

Hà Nội – 2025

Abstract

This report investigates the performance of two neural network architectures for image classification on the CIFAR-10 dataset: a Multilayer Perceptron (MLP) and a Convolutional Neural Network (CNN), with a particular focus on the VGG architecture as a representative CNN model. The primary objective is to evaluate how architectural complexity influences classification accuracy and efficiency, focusing on the role of convolutional layers in extracting spatial features from images. By analyzing the strengths and limitations of these models, the study aims to provide insights into selecting optimal neural network designs for image classification tasks, contributing to a deeper understanding of their applicability in real-world scenarios.

Technology Utilized:

Scikit-learn: a versatile Python library for machine learning, providing tools for data preprocessing, training, and evaluation. Here, it computes metrics like accuracy and confusion matrices, ensuring precise model assessment. Its ease of use makes it ideal for analyzing and evaluating machine learning models.

Pytorch: an open-source deep learning framework by Facebook's AI Research lab, is prized for its dynamic computational graph and efficiency. It supports designing and training models like VGG and MLP, preprocessing data, and loading the CIFAR-10 dataset via torchvision, which includes data augmentation tools like normalization and random cropping. Pytorch's autograd simplifies gradient calculations, and GPU support boosts performance, making it a top choice for rapid prototyping and complex deep learning tasks.

Numpy: Python's core library for numerical computing, enabling efficient array and matrix operations. It handles data manipulation tasks like reshaping image data for model input, ensuring fast and accurate computations throughout the study.

Matplotlib: a flexible Python library for creating visualizations. It generates plots like training loss curves and confusion matrices, offering clear insights into model performance and aiding effective result presentation.

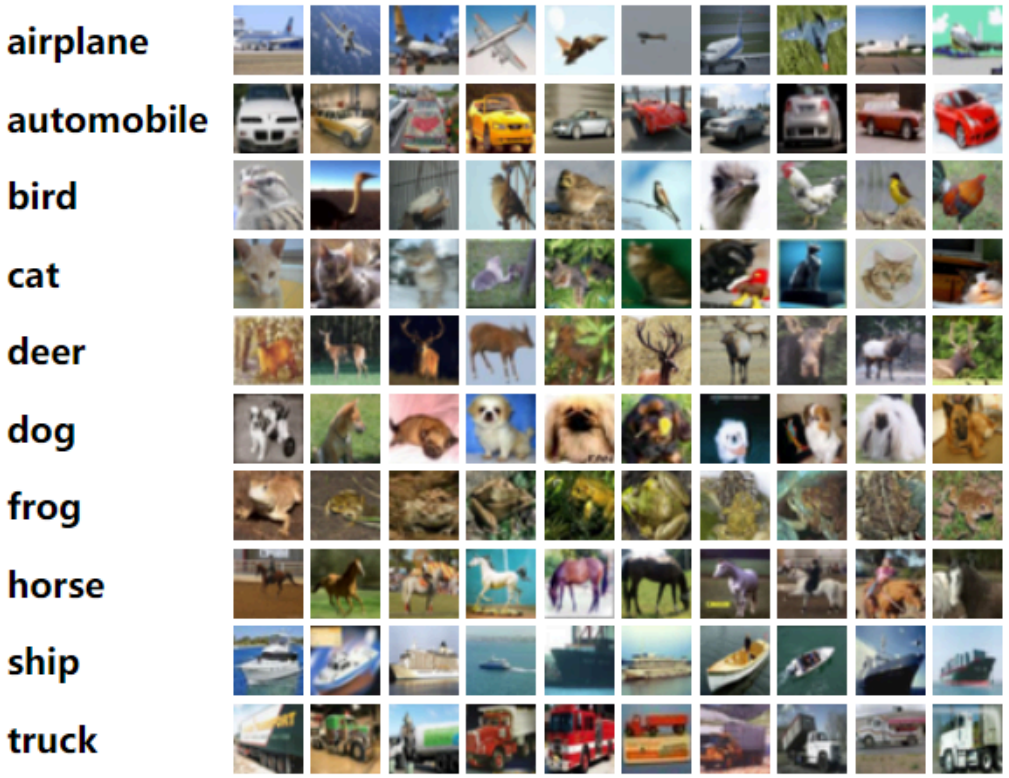
I. Overview

This report presents a comparative analysis of the performance of Multilayer Perceptrons (MLPs) and VGG in the task of image classification, utilizing the widely recognized CIFAR-10 dataset. The objective is to elucidate the inherent architectural differences between these models and their respective

implications for classification accuracy and efficiency when applied to a standard image benchmark.

The CIFAR-10 Dataset

The CIFAR-10 dataset, a foundational benchmark in computer vision and machine learning, was introduced by the Canadian Institute for Advanced Research. It comprises 60,000 color images, each size 32×32 pixels, meticulously categorized into 10 distinct classes. Each class contains 6,000 images, encompassing a diverse range of everyday objects such as airplanes, automobiles, and various animals. The dataset is conventionally partitioned into 50,000 images for training and 10,000 images for testing. Its compact dimensions, coupled with its categorical diversity, render CIFAR-10 an ideal platform for evaluating image classification algorithms and deep learning models, striking a balance between computational tractability and representational complexity.



Multilayer Perceptron (MLP)

A Multilayer Perceptron (MLP) represents a fundamental class of feedforward artificial neural networks, extensively employed in supervised learning paradigms such as classification and regression. Architecturally, an MLP is characterized by multiple layers of interconnected computational nodes, or neurons, organized into an input layer, one or more hidden layers, and an output layer. The operational principle of each neuron involves computing a

weighted sum of its inputs, which is subsequently transformed by a non-linear activation function. This multi-layered, non-linear processing capability enables MLPs to approximate complex functions and discern intricate patterns and relationships within the input data.

VGG: A Specialized Convolutional Neural Network

VGG, developed by the Visual Geometry Group at the University of Oxford in 2014, is a seminal deep CNN architecture specifically engineered for large-scale image classification tasks. A defining characteristic of VGG is its homogeneous architecture, which primarily consists of stacked 3×3 convolutional filters followed by max-pooling layers, culminating in fully connected layers. This design facilitates the hierarchical extraction of increasingly abstract features from raw image data. Notably, variants such as VGG16 and VGG19 achieved remarkable success in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014, securing second place in the classification task with a top-5 error rate of 7.3% on the ImageNet dataset, which comprises over 14 million images across 1,000 object categories. This performance underscored the efficacy of deep convolutional architectures with small, repetitive filters for robust image representation.

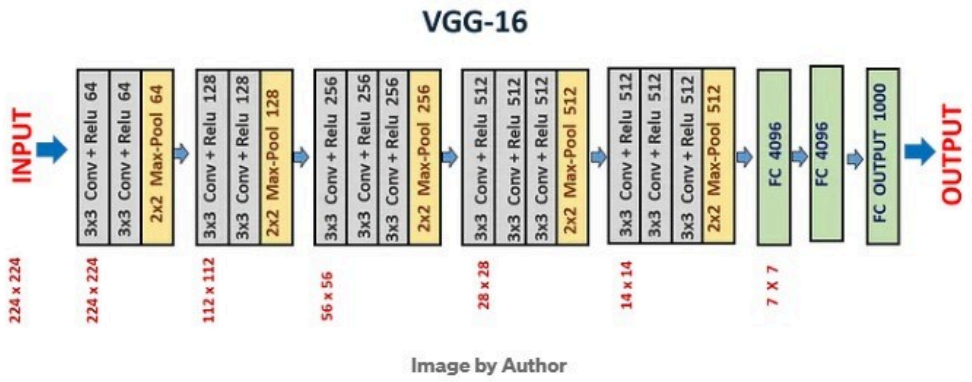


Image by Author

This report will proceed to detail the implementation and experimental results of both MLP and VGG models on the CIFAR-10 dataset, providing a quantitative comparison of their classification capabilities.

II. Preprocessing Data

To prepare the CIFAR-10 dataset for model training and evaluation, the following preprocessing steps were carried out. While many steps are common to both MLP and VGG architectures, there are crucial differences in how the data is prepared for the input of each model type.

Common Preprocessing Steps for Both MLP and VGG

1. Tensor Conversion and Normalization

All images in the CIFAR-10 dataset, including both the training and test sets, were converted to PyTorch Tensor format. Subsequently, they were normalized by subtracting the channel-wise mean and dividing by the channel-wise standard deviation, calculated across the entire training dataset. This normalization scales pixel values to a specific range, typically or [0,1], which helps stabilize the training process and accelerate model convergence.

The mean and standard deviation values used are:

Mean: (0.4914,0.4822,0.4465)

Standard Deviation: (0.2023,0.1994,0.2010)

The normalization formula for each pixel x and color channel c is:

$$x'_c = \frac{x_c - \mu_c}{\sigma_c}$$

where μc is the mean and σc is the standard deviation of channel c.

2. Dataset Splitting and Batching

The CIFAR-10 training set, comprising 50,000 images, originally designated for training, was further separated into two distinct groups. A primary Training Set, comprising 40,000 images, was used for the core learning phase of the models. The remaining 10,000 images from this initial set formed the Validation Set. This validation set played a critical role during the model's development, allowing for adjustments to its internal settings (hyperparameters) and for monitoring its performance to prevent it from becoming too specialized to the training data.

The final 10,000 images from the CIFAR-10 test set were set aside as the Test Set. This set was kept completely separate and was not used at any point during training or validation, ensuring an unbiased and definitive assessment of how well the trained models could perform on entirely new, unseen images.

After the datasets were split, the images were organized into manageable groups for efficient processing during model training and evaluation. Instead of feeding individual images to the model one by one, which can be computationally inefficient, the images were processed in mini-batches. For the training data, these mini-batches were created by randomly shuffling the images before each pass through the entire dataset. This randomization prevents the model from learning any specific order of the images. For the validation and test data, the order of images within their batches was kept consistent, as the goal for these sets is consistent evaluation rather than learning. The specific sizes of these mini-batches were:

Training Data: Batches of 128 images are used with shuffling enabled to randomize sample order, mitigating overfitting by varying the sequence of data presented in each epoch.

Validation Data: Batches of 128 images are processed without shuffling to maintain a consistent evaluation order.

Test Data: Batches of 100 images are processed without shuffling, aligning with standard evaluation practices. Additionally, both models utilize two worker processes for parallel data loading, optimizing computational efficiency.

Differences in Preprocessing Steps

To bolster the generalization performance of the VGG model, the training data from the CIFAR-10 dataset is subjected to augmentation techniques that introduce controlled variability, thereby enhancing the model's robustness to diverse image conditions. Two specific augmentation strategies are employed:

Random Cropping: Each 32x32-pixel image is padded by 4 pixels on all sides, after which a random 32x32-pixel region is extracted. This process simulates spatial translations, enabling the model to learn features that remain invariant to minor positional shifts in the image content.

Random Horizontal Flipping: Images are mirrored horizontally with a 40% probability, introducing variations that mimic differing orientations. This technique enhances the model’s ability to generalize across images exhibiting potential left-right symmetry or orientation changes.

Data augmentation is applied exclusively to the VGG model’s training set due to its architecture, which leverages spatial hierarchies and benefits from transformations like random cropping and flipping to learn robust, translation-invariant features. In contrast, the MLP processes flattened image vectors, lacking the ability to exploit spatial relationships, rendering such

augmentations less effective. Additionally, the VGG model's deeper architecture is more prone to overfitting on the CIFAR-10 dataset, necessitating augmentation to increase training data diversity, whereas the simpler MLP is less susceptible and does not require these techniques.

III. Models' Structure

VGG Architecture

The network employs a hierarchical approach, systematically extracting spatial features through convolutional operations and progressively refining them before feeding them into a classification module. Its design emphasizes modularity, with a clear separation between feature extraction and classification stages, ensuring effective learning of visual patterns for the 10-class output.

1. Feature Extraction Section

The feature extraction component of VGG is organized into three distinct convolutional blocks, each designed to capture increasingly complex patterns in the input images. These blocks follow a consistent pattern of convolutional layers, normalization, activation, and spatial reduction, with each block increasing the depth of feature representation to build a rich hierarchy of visual features.

Block 1: The first block processes the raw RGB images through two convolutional layers, each applying a set of filters with small 3×3 kernels. These kernels are padded to maintain the input's spatial dimensions, ensuring no information is lost at the edges. After each convolutional layer, batch normalization is applied to stabilize and accelerate training by normalizing the activations, followed by a ReLU activation to introduce non-linearity, allowing the network to learn complex patterns. The block concludes with a 2×2 max-pooling layer, which reduces the spatial dimensions of the feature maps by half, focusing on the most prominent features while reducing computational load.

Block 2: The second block mirrors the structure of the first but doubles the number of filters, enhancing the network's capacity to capture more detailed and abstract features. It consists of two convolutional layers with 3×3 kernels, again padded to preserve spatial dimensions. Each convolution is followed by batch normalization for training stability and ReLU activation to maintain non-linearity. A max-pooling layer at the end of the block further reduces the spatial dimensions, concentrating

the feature representations and preparing them for the next stage of processing.

Block 3: The third block continues the pattern, further increasing the number of filters to capture even more intricate patterns in the data. Like the previous blocks, it includes two convolutional layers with 3×3 kernels, batch normalization, and ReLU activations, followed by a final max-pooling layer. This block significantly deepens the feature representation, enabling the network to model complex visual structures while continuing to reduce the spatial dimensions of the feature maps.

2. Adaptive Pooling and Classification

After the convolutional blocks, the model employs an adaptive average pooling layer to transform the feature maps into a fixed-size representation. This layer reduces each feature map to a single value (1×1 spatial dimension), creating a compact feature vector regardless of the input image size. This step ensures the model can handle varying input dimensions while producing a consistent input for the classification stage.

The classification section consists of a two-layer fully connected network designed to map the extracted features to the 10 CIFAR-10 classes. The first fully connected layer takes the compact feature vector and reduces it to a smaller set of features, applying ReLU activation to maintain non-linearity and a dropout mechanism with a 50% probability to prevent overfitting by randomly disabling half of the connections during training. The second and final layer is a linear transformation that produces 10 output scores, one for each class, suitable for classification (typically paired with a softmax function in the loss calculation to generate probabilities).

Multi-Layer Perceptron (MLP) Architecture

Unlike convolutional networks, the MLP treats the input as a single, flattened vector, processing it through a series of dense layers. This straightforward architecture is less computationally intensive than CNNs but less specialized for image data, making it a baseline model for comparison. Its design focuses on transforming the input through a sequence of linear transformations and non-linear activations to produce class predictions and reducing the input's dimensionality while learning patterns relevant to classification.

Input Processing: The network begins by flattening the input RGB images (32×32 pixels with 3 color channels) into a single, one-dimensional vector. This flattening discards spatial relationships

but creates a uniform input format for the dense layers, treating each pixel value as an independent feature.

First Hidden Layer: The first hidden layer takes the flattened input and transforms it into a smaller set of features using a fully connected layer. A ReLU activation follows, introducing non-linearity to allow the network to model complex relationships between pixel values. This layer serves as the initial stage of feature transformation, capturing patterns in the input data.

Second Hidden Layer: The second hidden layer further refines the features, reducing their dimensionality while applying another ReLU activation. This creates a funnel-like structure, progressively compressing the feature space to focus on the most relevant information for classification. The use of ReLU ensures the network can learn non-linear patterns effectively.

Regularization: After the second hidden layer, a dropout layer is applied with a low probability (10%), randomly deactivating a small fraction of connections during training. This regularization technique helps prevent overfitting, ensuring the model generalizes better to unseen data by avoiding over-reliance on specific neurons.

Output Layer: The final layer is a fully connected linear transformation that maps the reduced feature set to 10 output values, corresponding to the 10 CIFAR-10 classes. No activation is applied at this stage, as the raw scores are typically processed by a softmax function in the loss calculation to produce class probabilities.

Both models are designed for CIFAR-10's 10-class classification task, but the VGG's convolutional design gives it an edge in handling the spatial complexity of images, while the MLP serves as a lightweight baseline for comparison. Together, they illustrate the trade-offs between architectural complexity and computational efficiency in neural network design.

IV. Training Process

The training of neural networks, whether a simpler MLP or a more complex VGG, generally follows a common iterative paradigm. However, specific choices in optimization algorithms, regularization techniques, and training duration often differ significantly due to their distinct architectural properties and the types of data they are designed to process.

Criterion (Loss Function): In both cases, the primary objective of training is to minimize a Cross-Entropy Loss function. This function quantifies the discrepancy between the model's predicted class probabilities and the true

labels, thereby guiding the learning process towards accurate classification. This is a standard and highly effective choice for multi-class classification tasks.

Iterative Training (Epochs): Training for both models is an iterative process, conducted over multiple epochs. An epoch signifies one complete pass through the entire training dataset. This repetitive exposure to data allows the models to progressively refine their internal parameters.

Optimizer: For both models, the Adam optimizer is utilized. Adam is an adaptive learning rate optimization algorithm that maintains per-parameter learning rates, often leading to efficient convergence. It incorporates principles from both momentum and RMSprop, making it robust across various tasks. Additionally, Adam is used with **weight decay**, a regularization technique that penalizes large weights to prevent overfitting, and potentially **label smoothing** to encourage better generalization.

Learning rate: The initial choice of a larger learning rate for the MLP might be attributed to its simpler architecture and fewer parameters compared to VGG. A less complex model might be able to tolerate and benefit from larger steps in the parameter space without destabilizing the training early on, potentially converging faster to a suitable solution for its inherent complexity. VGG's deeper and more intricate structure generally benefits from more careful learning rate management.

Metrics: During training, **metrics** such as training loss and accuracy are calculated and typically averaged over each epoch to monitor the model's learning progress. Following each epoch, validation loss and accuracy are computed on a separate dataset to assess the model's generalization ability and detect signs of overfitting.

V. Evaluation and Comparison

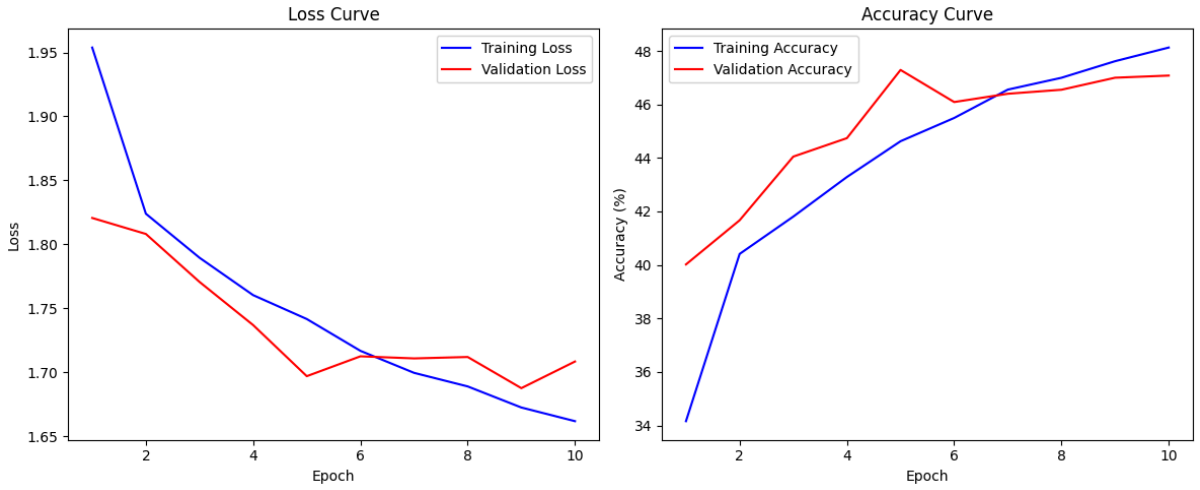


Figure 5.1: learning curve of MLP

Figure 5.1 provides valuable insights into the MLP model's training dynamics. Both the training loss and validation loss exhibit a consistent decreasing trend across all 10 epochs, indicating that the model is successfully learning from the data. Simultaneously, both training accuracy and validation accuracy show a steady increase, signifying an improvement in the model's predictive capability. The validation loss continues to decrease and the validation accuracy continues to rise until the final epoch, suggesting that the model has not yet fully overfit the training data within these 10 epochs, and further training might potentially lead to marginal improvements. However, the curves show a tendency to flatten towards the later epochs, especially around epoch 8-10, implying that the learning rate might be approaching a plateau where significant gains in performance are less likely with continued training under the same hyperparameter settings. This behavior indicates a relatively stable training process for the MLP model.

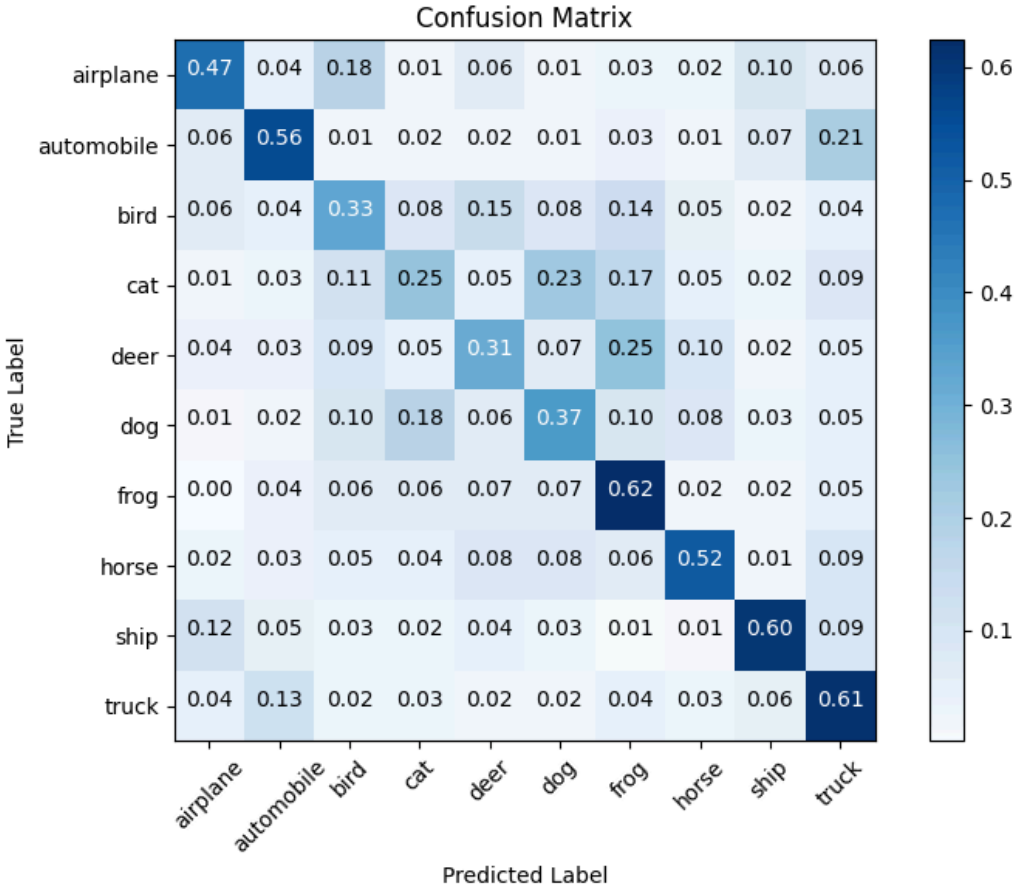


Figure 5.2: confusion matrix of MLP

The normalized confusion matrix (Figure 5.2) visually represents the classification performance of the MLP model across the 10 CIFAR-10 classes.

The diagonal elements, which represent the proportion of correctly classified instances for each class, highlight the model's strengths and weaknesses.

The model demonstrates strong performance in classifying "automobile" , "ship" , and "truck", indicating it accurately distinguishes these categories from others. Conversely, "cat" is often confused with "dog" and "bird" , while "bird" is commonly misclassified as "airplane" , "cat" , and "deer" . This common confusion among visually similar animal classes (e.g., cat-dog, bird-airplane/deer) is typical for simpler models like MLPs when dealing with complex image datasets like CIFAR-10, which contain fine-grained visual distinctions. The performance on "deer" and "horse" also indicates moderate challenges. Overall, the confusion matrix underscores the limitations of an MLP in capturing intricate spatial features necessary for high accuracy in image classification, leading to noticeable misclassifications, particularly between visually similar animal categories.

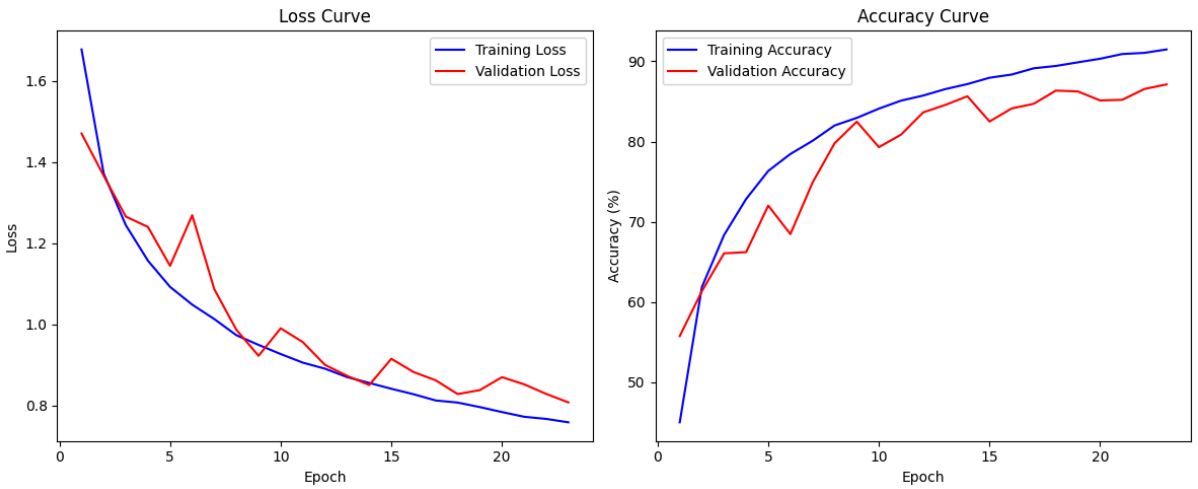


Figure 5.3: Learning curve of VGG

The learning curve for the VGG model, presented in Figure 3.3, offers a compelling demonstration of effective deep learning training over an extended period of more than 20 epochs. Both the training loss and validation loss exhibit a steep initial decrease, indicating rapid learning in the early stages, followed by a more gradual, yet consistent, decline throughout the subsequent epochs. This sustained reduction in both loss metrics signifies that the model is continuously refining its understanding of the data, effectively minimizing prediction errors on both seen and unseen examples. Concurrently, the training accuracy and validation accuracy show a robust and significant upward trend, progressively reaching high levels. This parallel improvement in accuracy across both datasets is a strong indicator of the model's capacity to learn complex features and generalize well.

While a slight gap may exist (indicating that the model performs marginally better on data it has seen during training), this gap does not significantly widen, suggesting that the VGG model, despite its depth and complexity, is not severely overfitting the training data. This controlled generalization ability is likely attributable to the combined effects of regularization techniques (e.g., Batch Normalization, Dropout as well as Weight Decay), the robust data augmentation applied, and the effectiveness of the learning rate schedule. The extended training duration of epochs, facilitated by these techniques, allows the VGG model to fully leverage its deep architecture to extract intricate hierarchical features from the image data, leading to its superior performance.

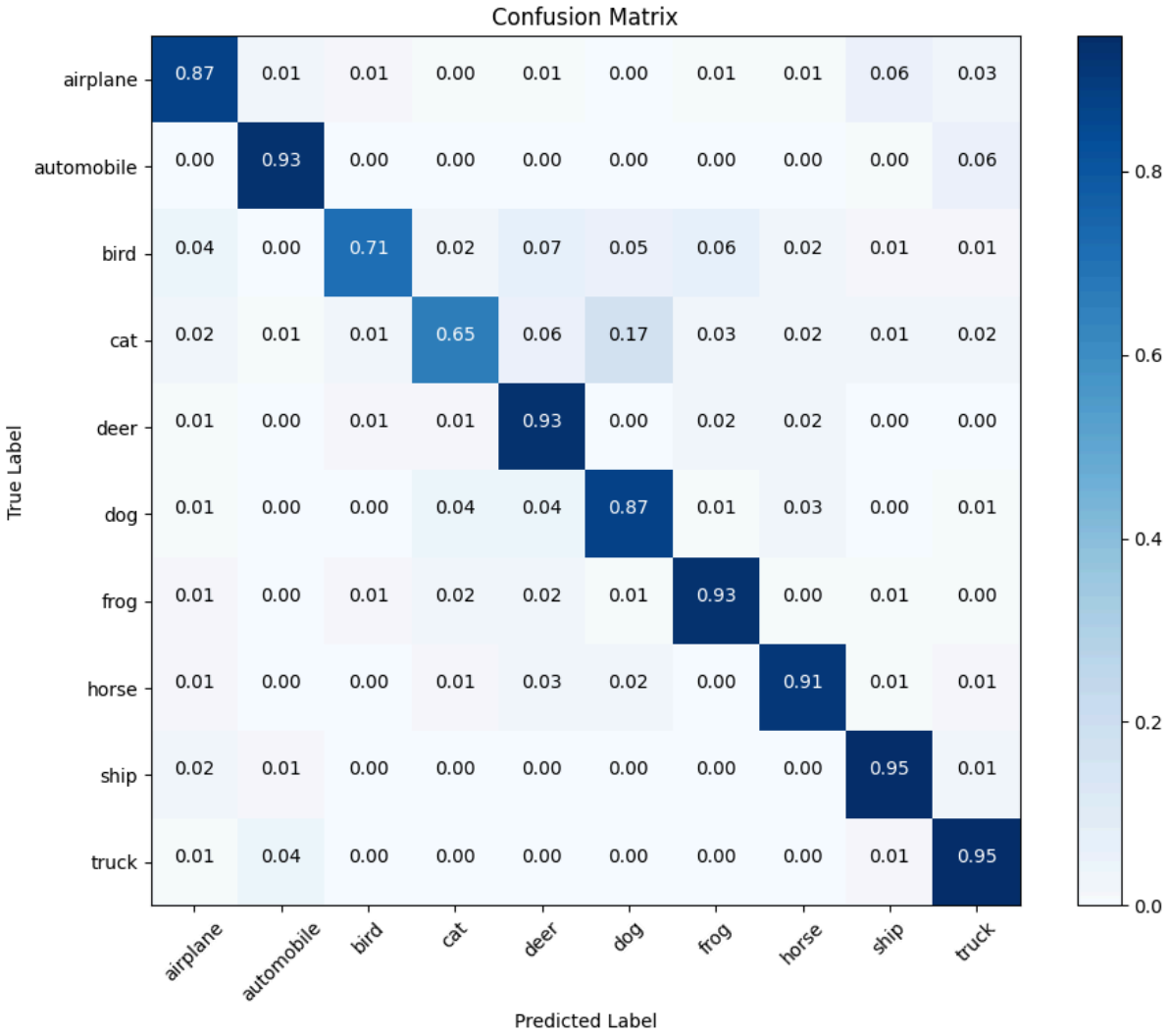


Figure 5.4: Confusion matrix of VGG

The normalized confusion matrix, displayed in Figure 3.4, provides an in-depth quantitative and qualitative assessment of the VGG model's classification efficacy on the challenging CIFAR-10 test set. A prominent observation is the significantly higher values across the main diagonal

compared to the MLP model's confusion matrix, unequivocally demonstrating the VGG model's superior discriminative power and overall higher classification accuracy.

The model exhibits excellent performance across several categories, achieving very high accuracy rates for "airplane" (**0.87**), "automobile" (**0.93**), "ship" (**0.95**), and "truck" (**0.95**). These high diagonal values indicate that the VGG model, with its convolutional layers, is highly effective at capturing and distinguishing the unique visual features that define these objects, leading to robust and accurate predictions for these classes.

Crucially, the VGG model also demonstrates substantial improvements in classifying categories that proved particularly challenging for the simpler MLP architecture, notably "cat" and "dog." The accuracy for "cat" has increased significantly to **0.65**, and for "dog" to **0.87**. While these values are not as high as for the vehicle categories, they represent a considerable leap in performance, indicating that the VGG's hierarchical feature extraction capabilities are much better equipped to differentiate between visually similar animal species. Even with this improvement, some confusion persists between "cat" and "dog," as is common in image classification tasks due to their inherent visual similarities; however, the proportion of these misclassifications is notably reduced compared to the MLP.

The off-diagonal elements across the entire matrix are generally much lower than those observed in the MLP's confusion matrix, signifying a substantial reduction in misclassifications across all pairs of classes. This indicates that the VGG model has learned more distinct and robust feature representations for each category, minimizing overlap in the feature space. For instance, common confusions observed in the MLP, such as "bird" being misclassified as "airplane" or "deer" becoming "horse," are significantly reduced or almost eliminated. This comprehensive reduction in misclassifications across the board underscores the profound advantage of using a deep convolutional architecture like VGG for complex image classification tasks, showcasing its ability to effectively capture intricate visual patterns and achieve high generalization accuracy.