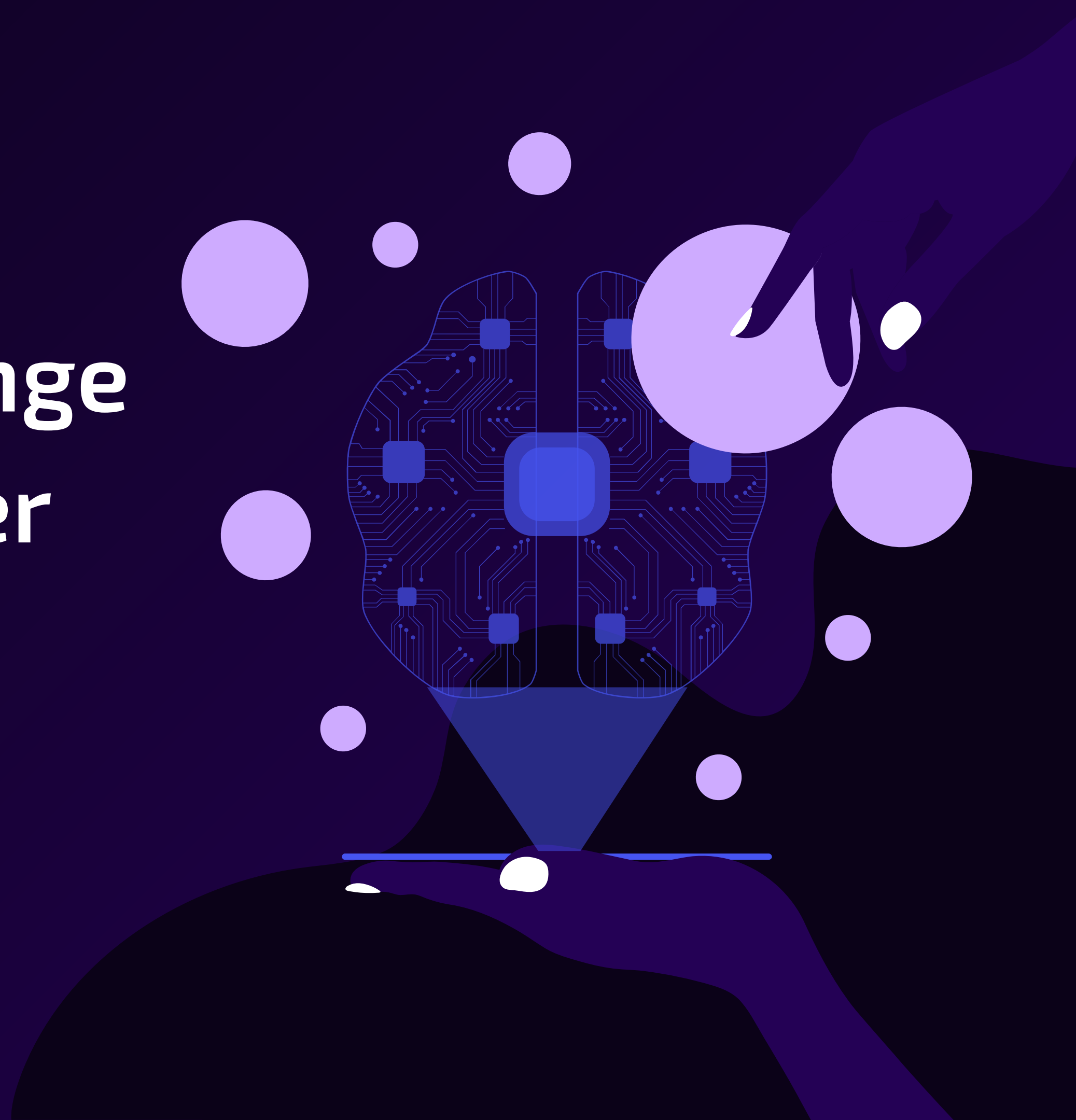


ImageNet Challenge

CNN vs. Transfer Learning

A Comparative Study of Deep Learning
Models for Image Classification



Our team



Ghala



Amal



Ghada



Wafa

Table of contents

01 Introduction

02 Building the
First Model
(CNN)

03 Building the Second
Model (Transfer
Learning)

04 Deployment



01

Introduction




Introduction

This project explores two powerful approaches to image classification:

- CNN from Scratch – A custom-built Convolutional Neural Network (CNN) is designed and trained on the CIFAR-10 dataset to classify images into ten categories. This approach focuses on learning the features directly from the data through a designed architecture, without relying on pre-existing knowledge.
- Transfer Learning – Pretrained models are fine-tuned to adapt to the CIFAR-10 dataset. By leveraging their already learned features, these models can accelerate learning and achieve higher performance with fewer data and computational resources.

The goal of this project is to compare both methods in terms of accuracy and efficiency, identifying the best-performing model for image classification.

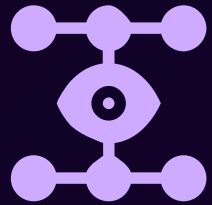


Data Preprocessing



Loading the Data

We loaded the CIFAR-10 dataset, which contains 60,000 images divided into 10 classes



Data Splitting

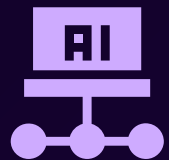
The dataset was automatically split into two parts:

- Training Set: Contains 50,000 images.
- Test Set: Contains 10,000 images



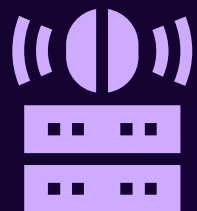
Normalization

The pixel values of the images were scaled to the range $[0, 1]$ by dividing by 255 to facilitate the training process



One-Hot Encoding

We converted the labels into a one-hot encoded format using `to_categorical` to make them suitable for training



Reshaping data dimensions

Adds extra dimensions to the images and labels for compatibility with the model

The background is a dark purple gradient. It features several light purple circles of various sizes scattered across the frame. On the left and right sides, there are stylized hands in a darker purple shade, each pointing its index finger towards a central light purple circle. The overall aesthetic is modern and tech-oriented.

02

Building the First Model (CNN)

Deep CNN Model: Building Architecture

Layer (type)	Output shape	Param #
Conv2d (conv2D)	(None, 32, 32, 32)	2,432
batch_normalization (BatchNormalization)	(None, 32, 32, 32)	128
Conv2d_1 (conv2D)	(None, 32, 32, 32)	9,248
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
dropout (Dropout)	(None, 16, 16, 32)	0
Conv2d_2 (conv2D)	(None, 16, 16, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 16, 16, 64)	256
Conv2d_3 (conv2D)	(None, 16, 16, 64)	36,928
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 64)	0
dropout_1 (Dropout)	(None, 8, 8, 64)	0
Conv2d_4 (conv2D)	(None, 8, 8, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 8, 8, 128)	512
Conv2d_5 (conv2D)	(None, 8, 8, 128)	147,584
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 128)	0
dropout_2 (Dropout)	(None, 4, 4, 128)	0
Conv2d_6 (conv2D)	(None, 4, 4, 256)	295,168
batch_normalization_3 (BatchNormalization)	(None, 4, 4, 256)	1,024
Conv2d_7 (conv2D)	(None, 4, 4, 256)	590,080
max_pooling2d_3 (MaxPooling2D)	(None, 2, 2, 256)	0
dropout_3 (Dropout)	(None, 2, 2, 256)	0
Conv2d_8 (conv2D)	(None, 2, 2, 512)	1,180,16
batch_normalization_4 (BatchNormalization)	(None, 2, 2, 512)	2,048
Conv2d_9 (conv2D)	(None, 2, 2, 512)	2,359,808
max_pooling2d_4 (MaxPooling2D)	(None, 1, 1, 512)	0
dropout_4 (Dropout)	(None, 1, 1, 512)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
dense (Dense)	(None, 1024)	525,312
dropout_5 (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 512)	524,800
dropout_6 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 10)	5,130

Conv2D (Convolutional Layer): Extracts features from images using filters/kernels to detect patterns like edges and textures

BatchNormalization: Normalizes activations to stabilize training and speed up convergence

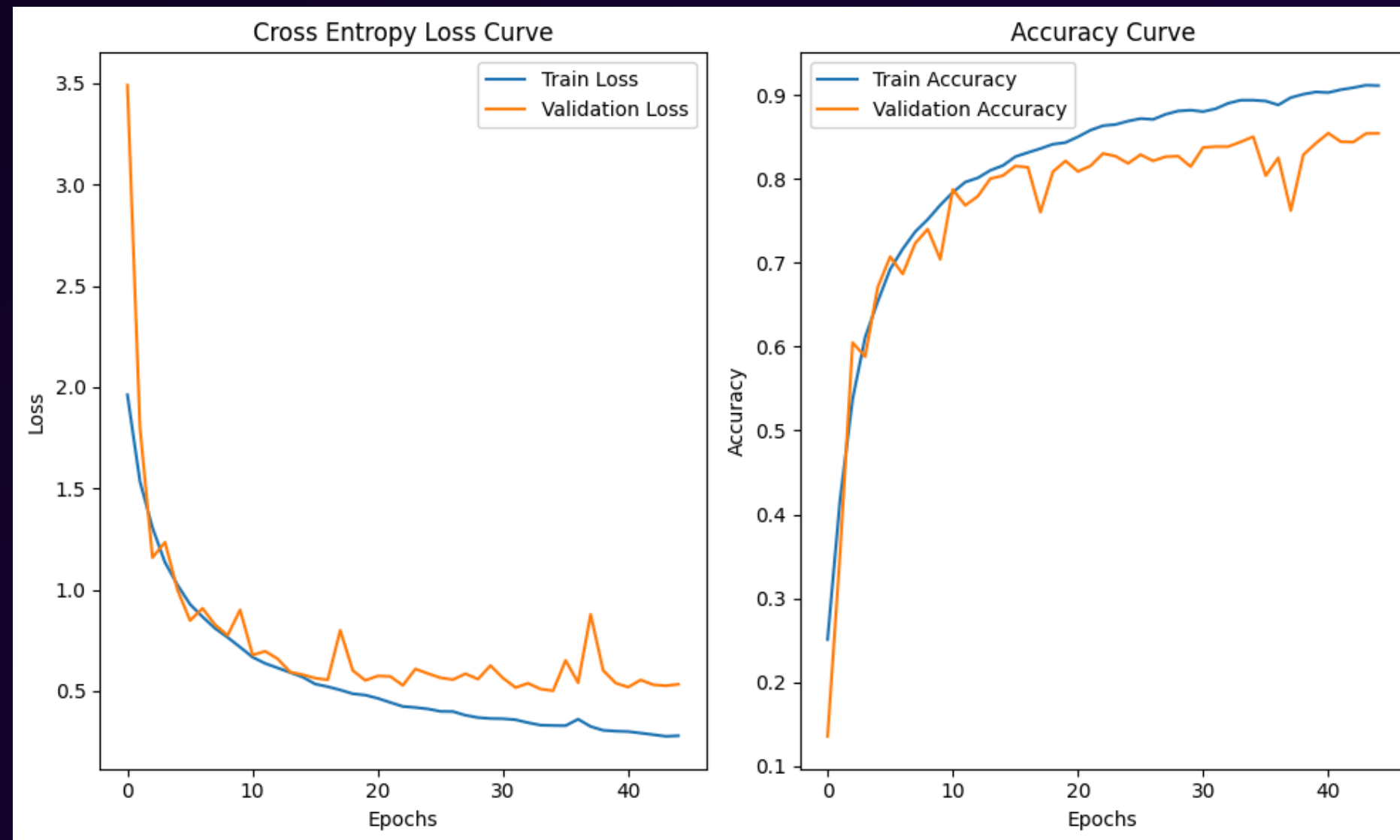
MaxPooling2D : Reduces spatial dimensions, keeping only the most important features to prevent overfitting

Dropout : Randomly drops connections during training to improve generalization and reduce overfitting.

GlobalAveragePooling2D : Reduces feature maps to a single value per channel, minimizing parameters while retaining key information.

Dense (Fully Connected Layer) : Combines extracted features and makes final predictions

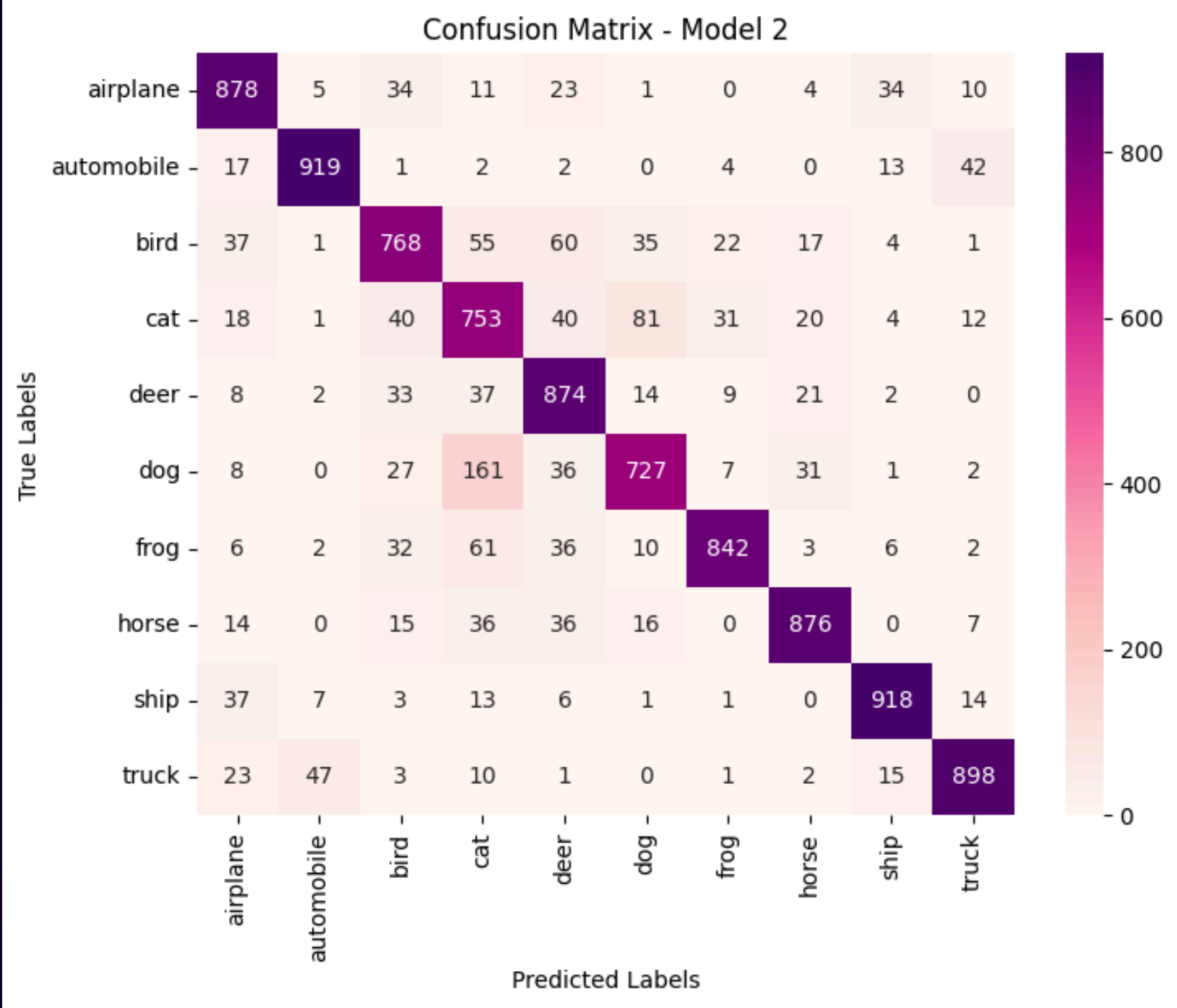
Deep CNN Model: Accuracy and Loss Analysis



Deep CNN Model Test Performance

- **Test Accuracy: 84.98%**
- **Test Loss: 52.25%**

Deep CNN Model: Classification Report



Model 2 Classification Report:				
	precision	recall	f1-score	support
0	0.84	0.88	0.86	1000
1	0.93	0.92	0.93	1000
2	0.80	0.77	0.79	1000
3	0.66	0.75	0.70	1000
4	0.78	0.87	0.83	1000
5	0.82	0.73	0.77	1000
6	0.92	0.84	0.88	1000
7	0.90	0.88	0.89	1000
8	0.92	0.92	0.92	1000
9	0.91	0.90	0.90	1000
accuracy			0.85	10000
macro avg	0.85	0.85	0.85	10000
weighted avg	0.85	0.85	0.85	10000

VGG Model: Building Architecture

Layer (type)	Output shape	Param #
Conv2d_10 (conv2D)	(None, 32, 32, 64)	1,792
batch_normalization_5 (BatchNormalization)	(None, 32, 32, 64)	256
Conv2d_11 (conv2D)	(None, 32, 32, 64)	36,928
batch_normalization_6 (BatchNormalization)	(None, 32, 32, 64)	256
max_pooling2d_5 (MaxPooling2D)	(None, 16, 16, 64)	0
Conv2d_12 (conv2D)	(None, 16, 16, 128)	73,856
batch_normalization_7 (BatchNormalization)	(None, 16, 16, 128)	512
Conv2d_13 (conv2D)	(None, 16, 16, 128)	147,584
batch_normalization_8 (BatchNormalization)	(None, 16, 16, 128)	512
max_pooling2d_6 (MaxPooling2D)	(None, 8, 8, 128)	0
Conv2d_14 (conv2D)	(None, 8, 8, 256)	295,168
batch_normalization_9 (BatchNormalization)	(None, 8, 8, 256)	1,024
Conv2d_15 (conv2D)	(None, 8, 8, 256)	590,080
batch_normalization_10 (BatchNormalization)	(None, 8, 8, 256)	590,080
max_pooling2d_7 (MaxPooling2D)	(None, 4, 4, 256)	0
Conv2d_16 (conv2D)	(None, 4, 4, 512)	1,180,160
batch_normalization_11 (BatchNormalization)	(None, 4, 4, 512)	2,048
Conv2d_17 (conv2D)	(None, 4, 4, 512)	2,359,808
batch_normalization_12 (BatchNormalization)	(None, 4, 4, 512)	2,048
max_pooling2d_8 (MaxPooling2D)	(None, 2, 2, 512)	0
Conv2d_18 (conv2D)	(None, 2, 2, 512)	2,359,808
batch_normalization_13 (BatchNormalization)	(None, 2, 2, 512)	2,048
Conv2d_19 (conv2D)	(None, 2, 2, 512)	2,359,808
batch_normalization_14 (BatchNormalization)	(None, 2, 2, 512)	2,048
max_pooling2d_9 (MaxPooling2D)	(None, 1, 1, 512)	0
flatten (Flatten)	(None, 512)	0
dense_3 (Dense)	(None, 4096)	2,101,248
dropout_7 (Dropout)	(None, 4096)	0
dense_4 (Dense)	(None, 4096)	16,781,312
dropout_8 (Dropout)	(None, 4096)	0
dense_5 (Dense)	(None, 10)	40,970

BatchNormalization: Normalizes activations to stabilize training and speed up convergence

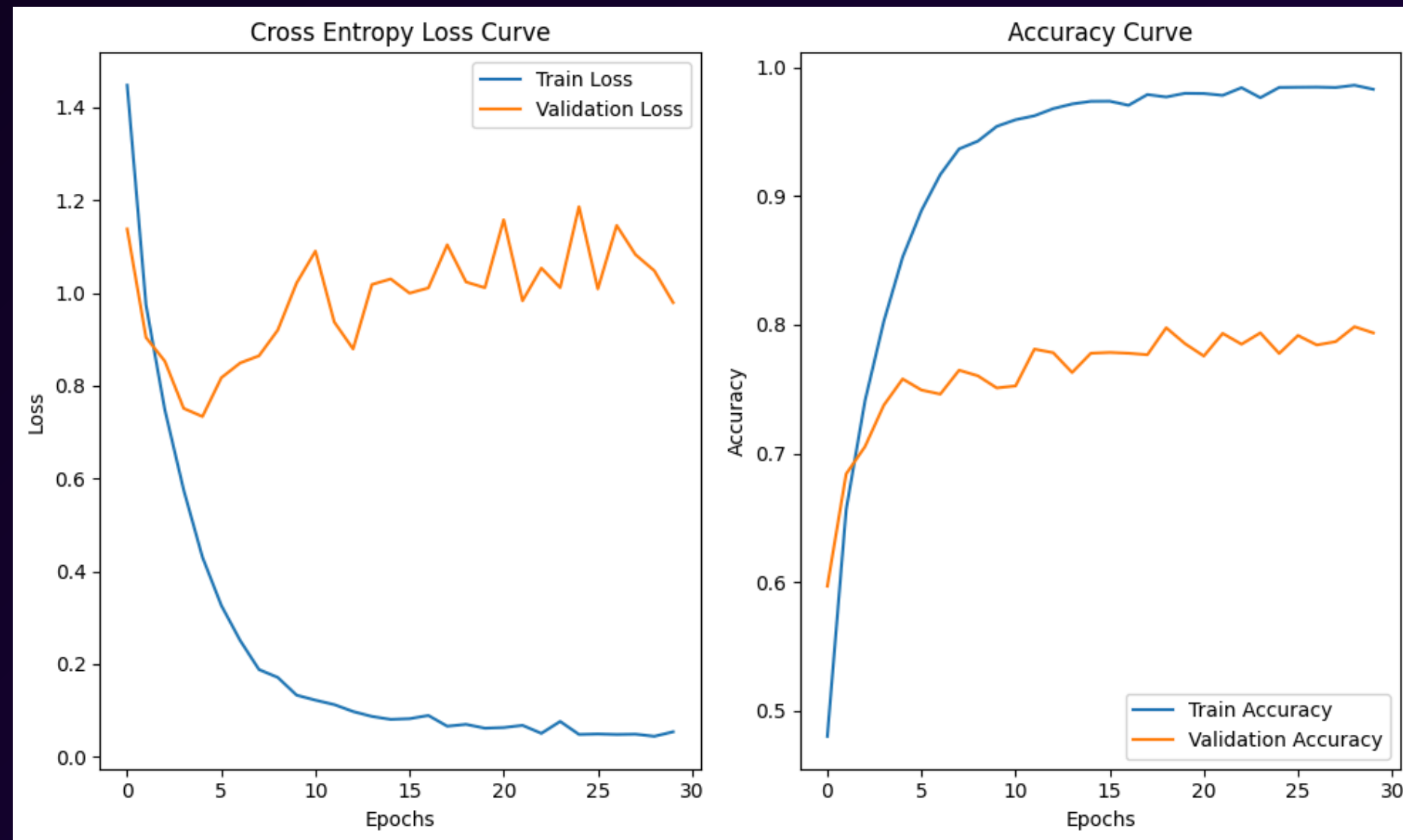
MaxPooling2D : Reduces spatial dimensions, keeping only the most important features to prevent overfitting

Dropout : Randomly drops connections during training to improve generalization and reduce overfitting.

Flatten : Converts multi-dimensional data into a 1D vector for compatibility with fully connected layers

Dense (Fully Connected Layer) : Combines extracted features and makes final predictions

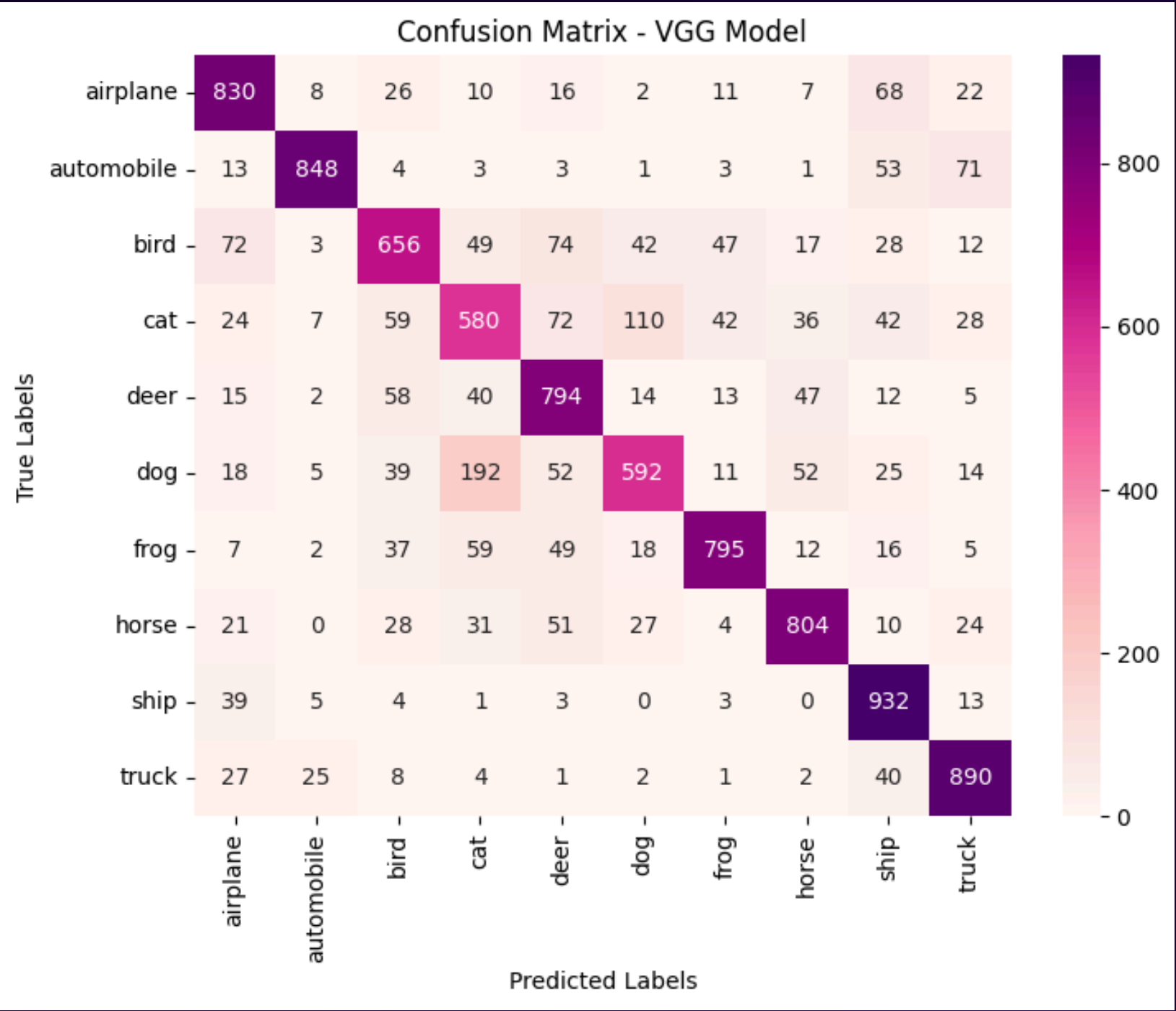
VGG Model: Accuracy and Loss Analysis



VGG Model Test Performance:

- **Test Accuracy: 77.50%**
- **Test Loss: 107.25%**

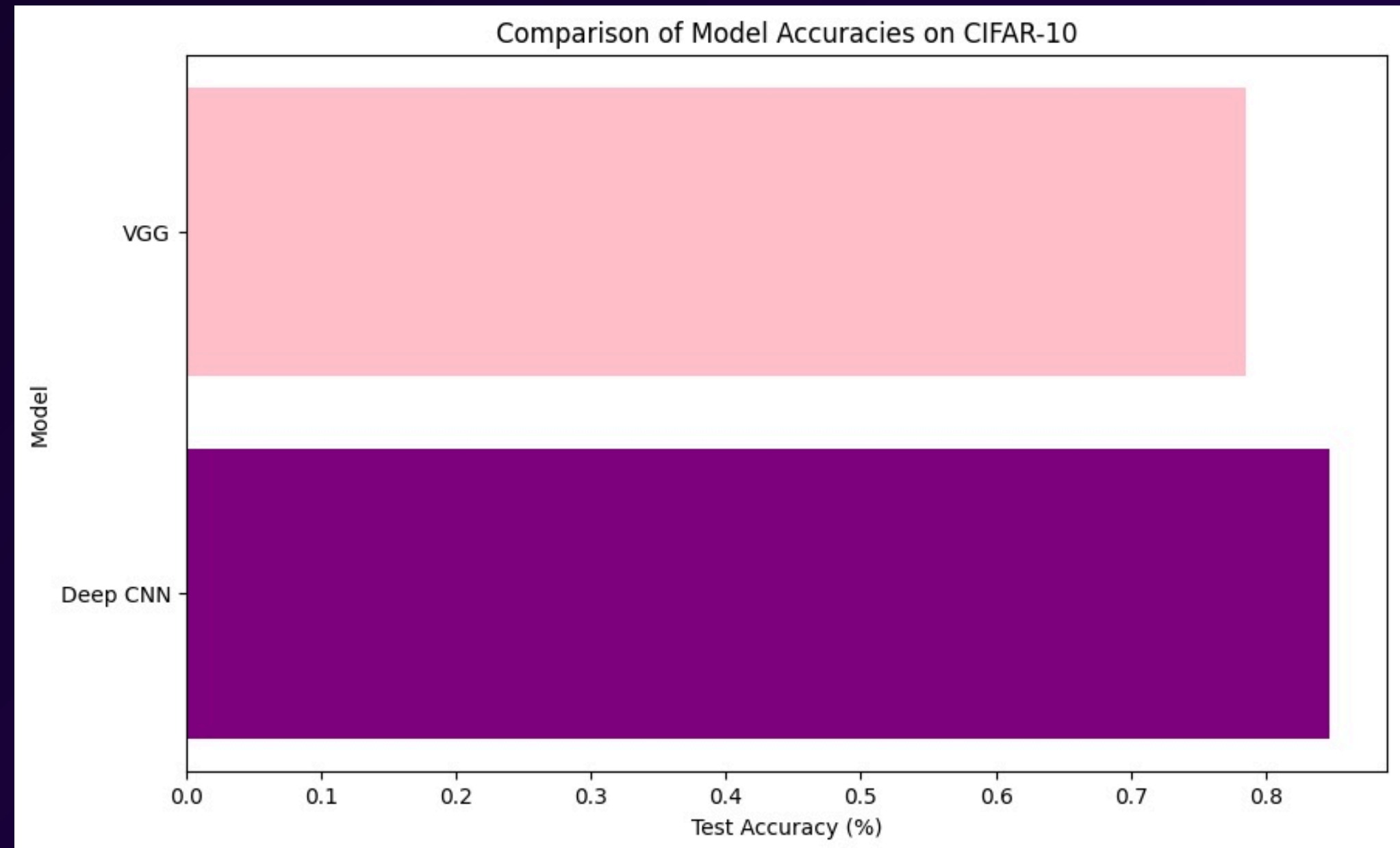
VGG Model: Classification Report



VGG Model Classification Report:				
	precision	recall	f1-score	support
0	0.78	0.83	0.80	1000
1	0.94	0.85	0.89	1000
2	0.71	0.66	0.68	1000
3	0.60	0.58	0.59	1000
4	0.71	0.79	0.75	1000
5	0.73	0.59	0.65	1000
6	0.85	0.80	0.82	1000
7	0.82	0.80	0.81	1000
8	0.76	0.93	0.84	1000
9	0.82	0.89	0.85	1000
accuracy			0.77	10000
macro avg	0.77	0.77	0.77	10000
weighted avg	0.77	0.77	0.77	10000

Compare between the models

We compared the performance of the Deep CNN and VGG models on the CIFAR-10 dataset and found that the Deep CNN model achieved a higher test accuracy. The best-performing model, Deep CNN, has been saved as `deep_cnn_model.h5`



The background is a dark purple gradient. It features several light purple circles of varying sizes. Two hands, rendered in a dark purple silhouette, are positioned on the left and right sides, each pointing towards a central light purple circle. The overall aesthetic is modern and tech-oriented.

03

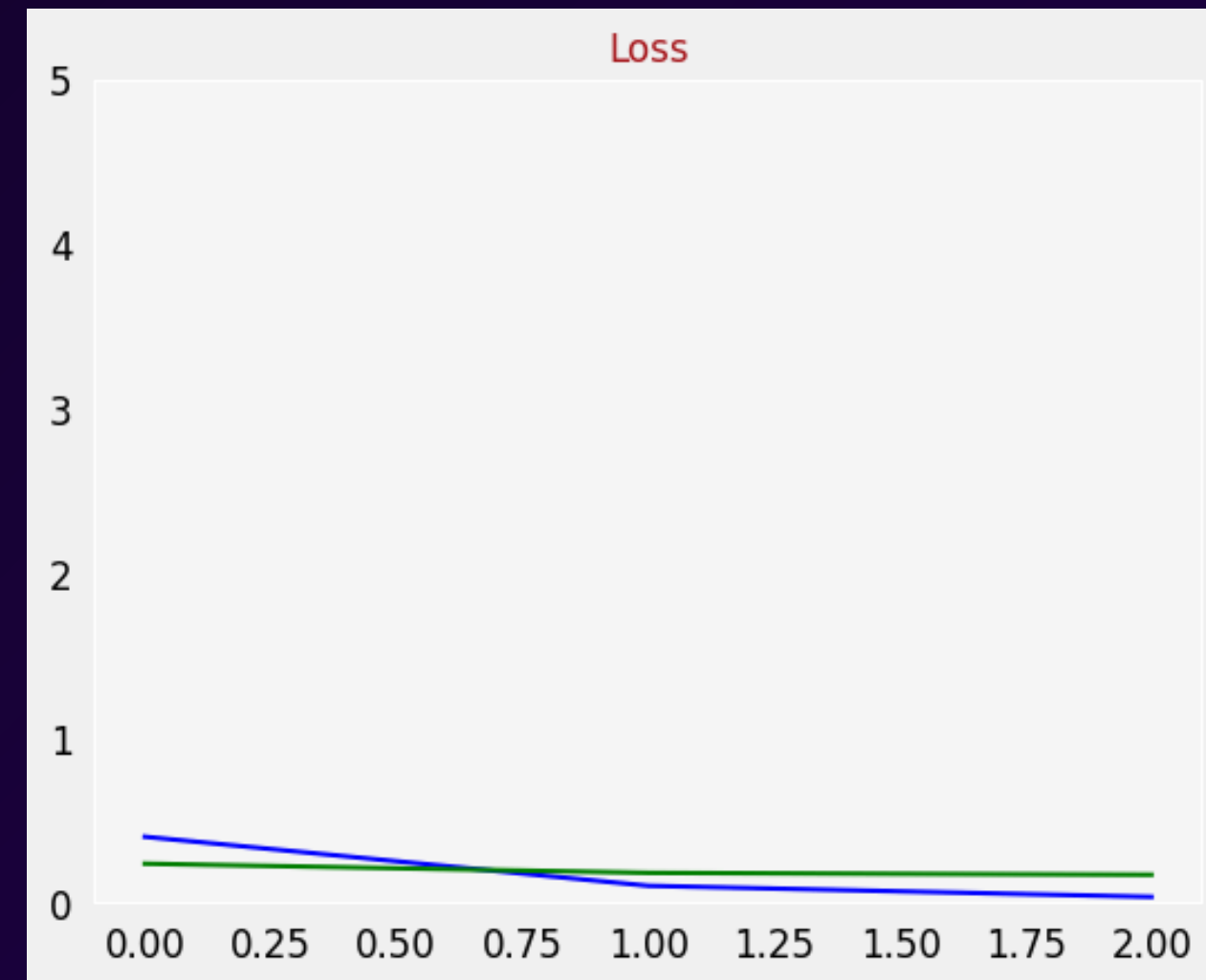
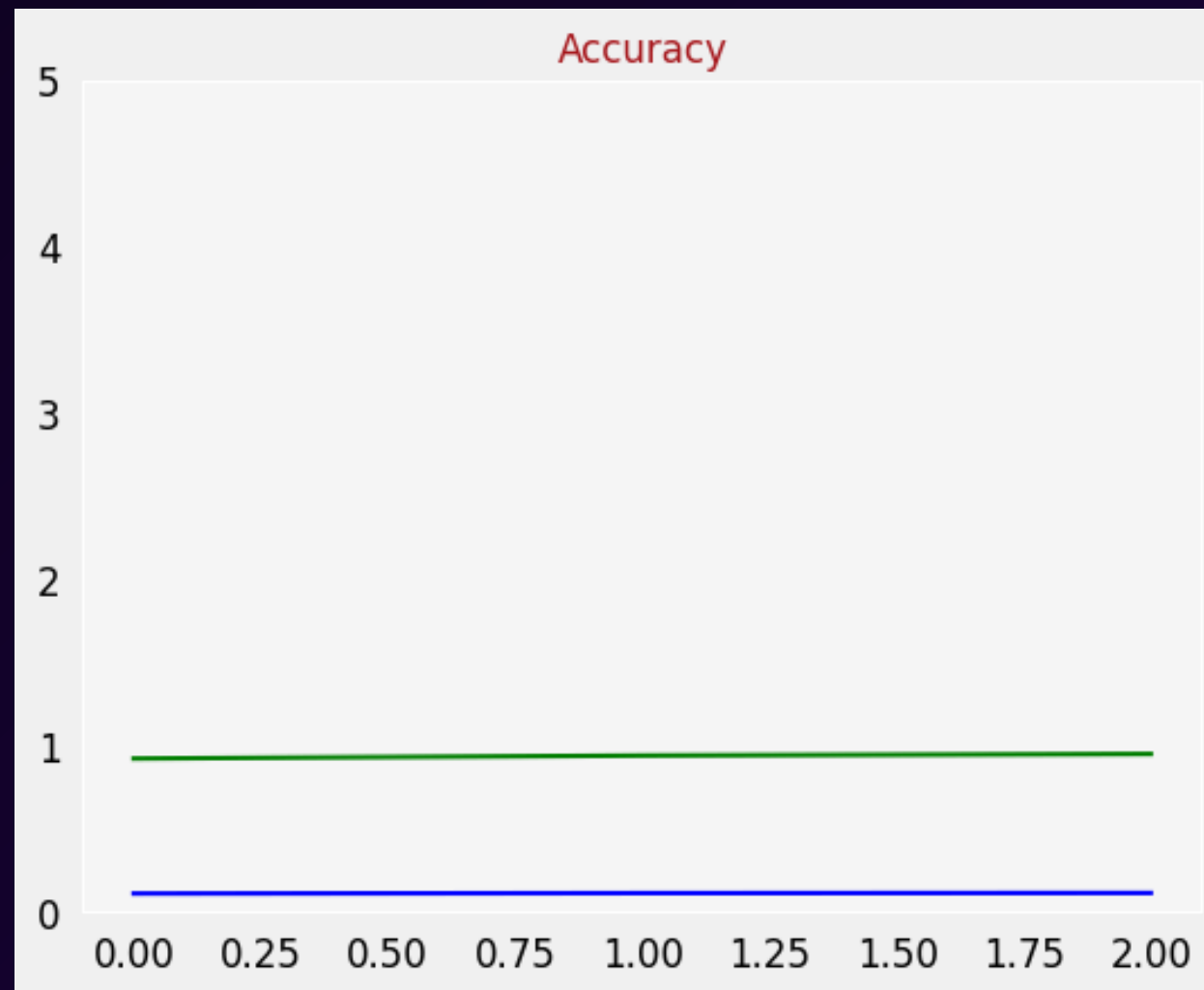
Building the Second Model (Transfer Learning)

ResNet50 Model: Building Architecture

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 32, 32, 3)	0
up_sampling2d (UpSampling2D)	(None, 224, 224, 3)	0
resnet50 (Functional)	(None, 7, 7, 2048)	23,587,712
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 1024)	2,098,176
dense_1 (Dense)	(None, 512)	524,800
classification (Dense)	(None, 10)	5,130

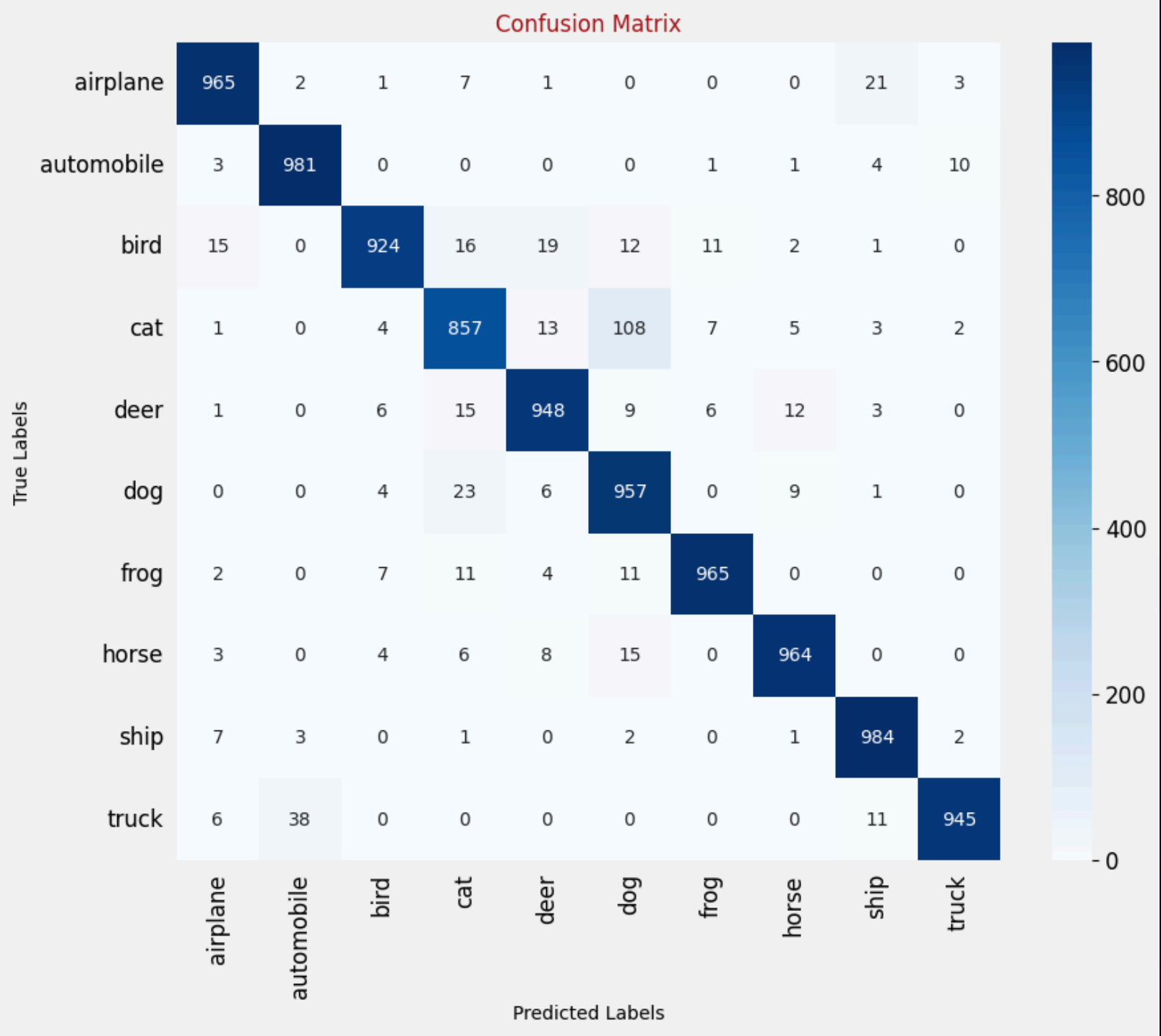
The model is designed to leverage pre-trained weights from ResNet50, with most layers frozen to retain learned features, while the dense layers are trainable for fine-tuning on the CIFAR-10 dataset.

ResNet50 Model: Accuracy and Loss Analysis



The model achieves a high accuracy of 94.73% and a low loss of 0.1717, indicating strong performance on the CIFAR-10 dataset. The plot demonstrates consistent convergence, with both accuracy and loss stabilizing over training epochs.

ResNet50 Model: Classification Report



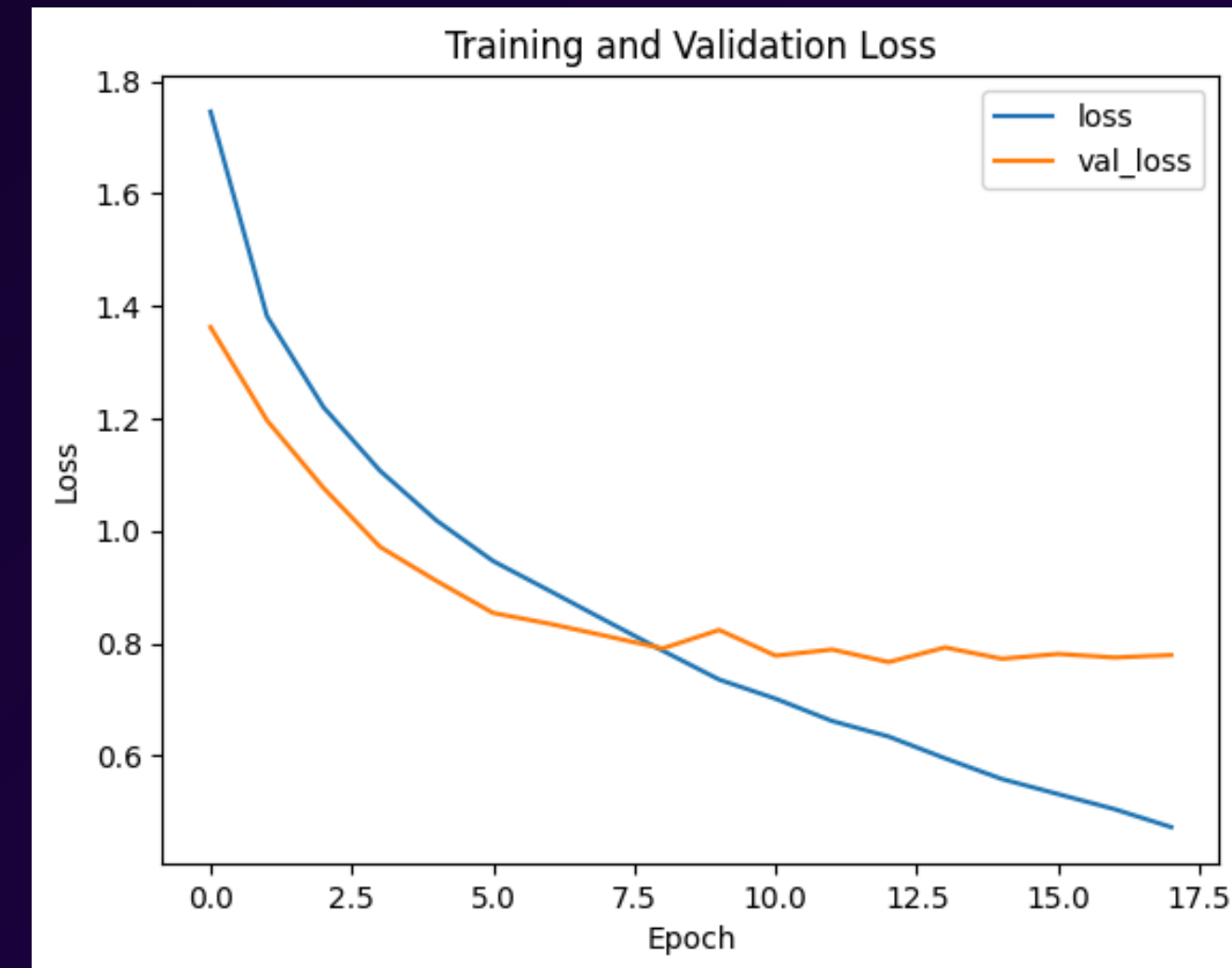
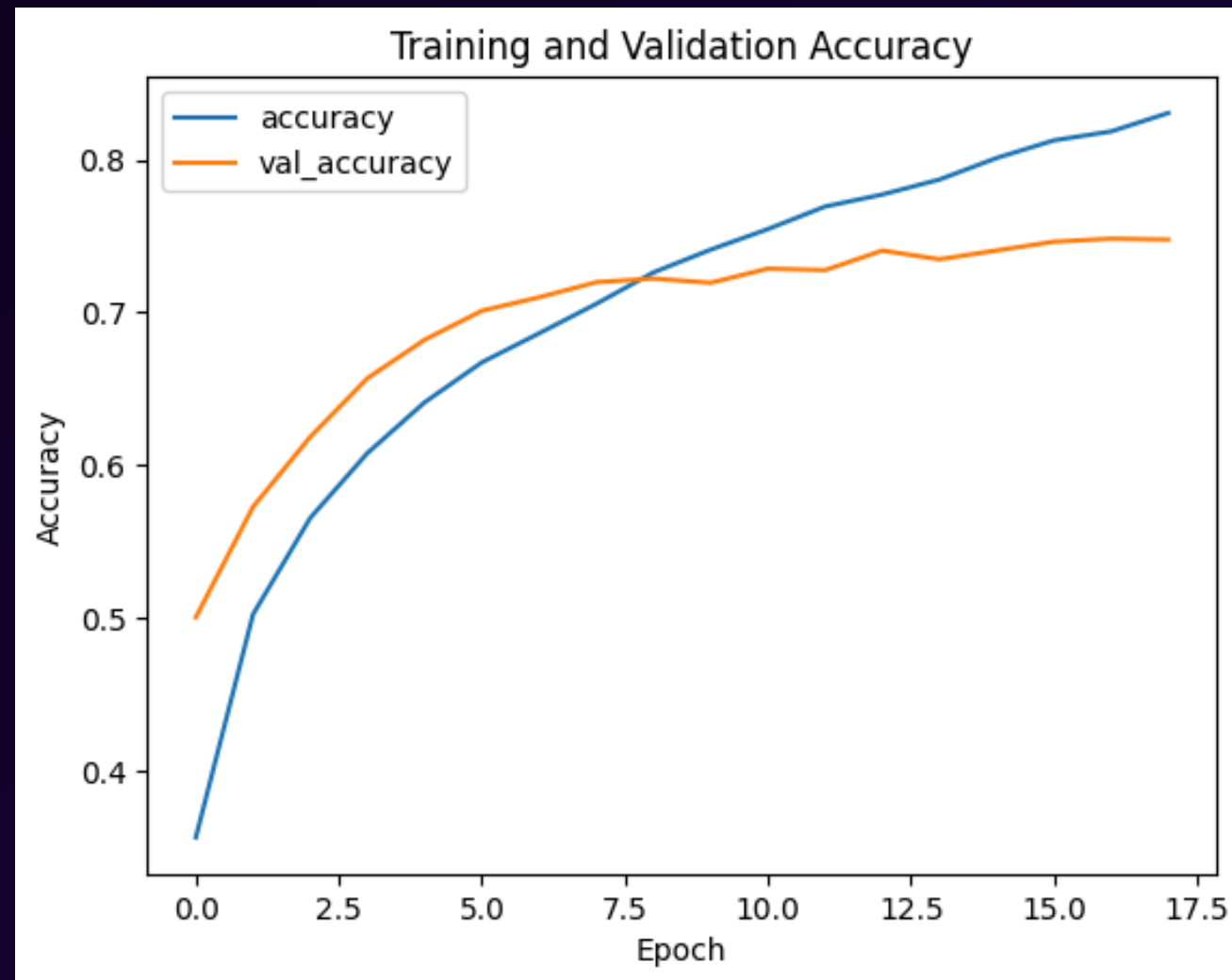
The confusion matrix highlights the model's strong performance, with high diagonal values indicating accurate predictions across most classes. Misclassifications are minimal, with the highest confusion observed between similar classes like cats and dogs.

VGG16 Model: Building Architecture

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 32)	896
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_1 (Conv2D)	(None, 16, 16, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_2 (Conv2D)	(None, 8, 8, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 128)	262,272
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1,290

The VGG16 model It consists of convolutional layers (Conv2D) for feature extraction, max pooling layers for dimensionality reduction, and fully connected (Dense) layers for classification. A dropout layer is included to prevent overfitting, and the final output layer has 10 classes, indicating a multi-class classification problem

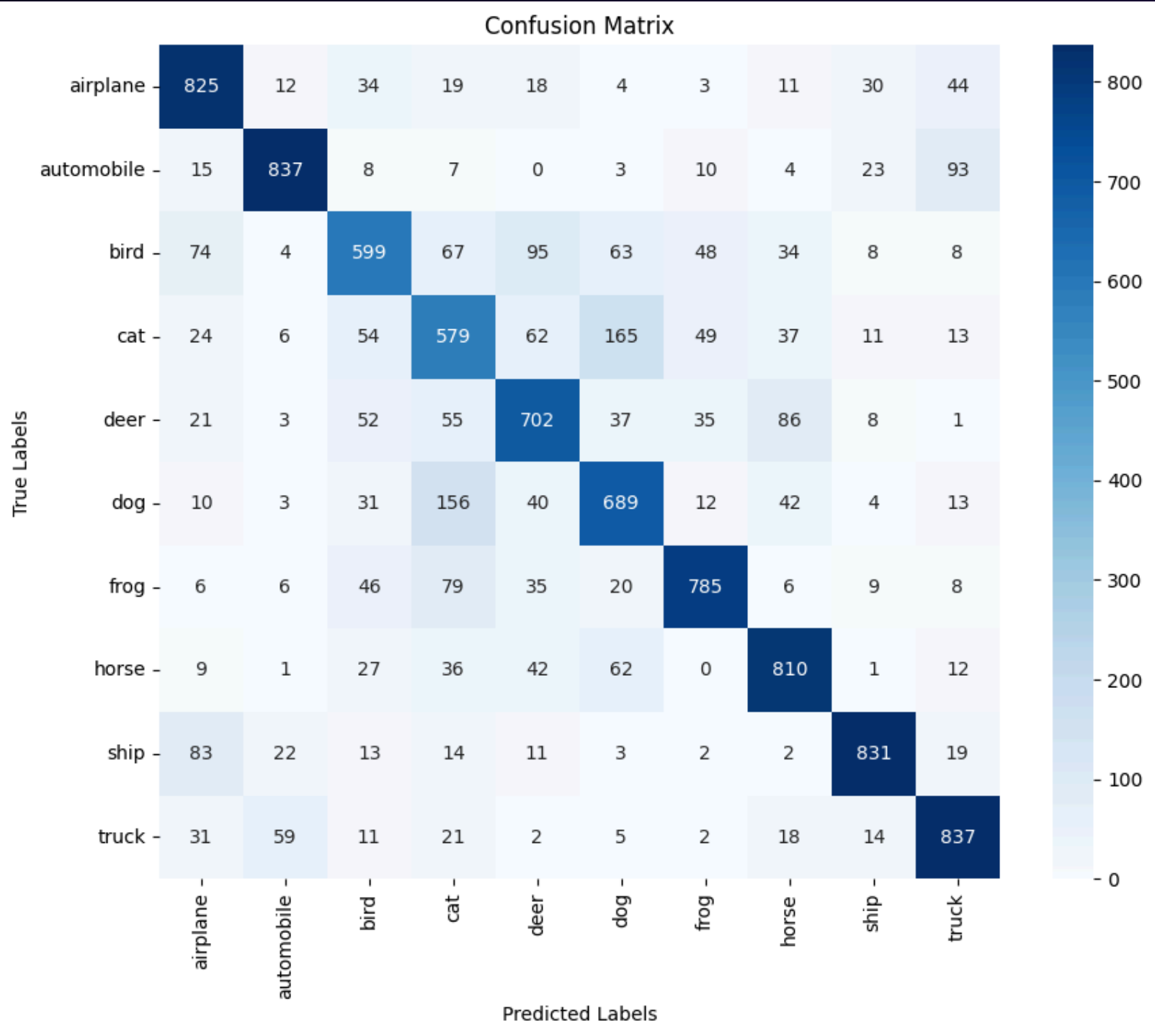
VGG16 Model: Accuracy and Loss Analysis



Accuracy Graph (Left): Shows the training and validation accuracy over epochs. The model improves steadily, but a gap between training and validation suggests potential overfitting.

Loss Graph (Right): Displays the training and validation loss reduction. While training loss decreases consistently, validation loss stabilizes with slight fluctuations, indicating the need for further fine-tuning.

VGG16 Model: Classification Report



The confusion matrix provides insights into the model's classification performance across different categories. The diagonal values represent correct predictions, while off-diagonal values indicate misclassifications. The model performs well on certain classes but struggles with others, suggesting potential areas for improvement such as data augmentation or fine-tuning.



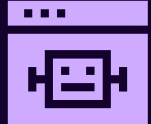
04

Deployment

Deployment: Cat Picture


cat

99.64%




deer

0.03%



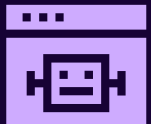
dog

0.27%




ship

0.00%




truck

0.00%








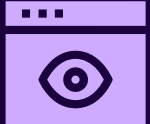
airplane

0.00%




automobile

0.00%



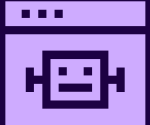
bird

0.01%



frog

0.04%



horse

0.00%

Deployment: Cat Picture

Image Classifier

Choose File

Cat_November_2010-1a.jpg

Upload and Classify

Predicted Class: cat

Probabilities:

airplane: 0.00%
automobile: 0.00%
bird: 0.01%
cat: 99.64%
deer: 0.03%
dog: 0.27%
frog: 0.04%
horse: 0.00%
ship: 0.00%
truck: 0.00%

Deployment: Truck Picture

cat

0.02%

deer

0.00%

dog

0.00%

ship

0.01%

truck

99.73%



airplane

0.06%

automobile

0.17%

bird

0.01%

frog

0.00%

horse

0.01%

Deployment: Truck Picture

Image Classifier

Choose File

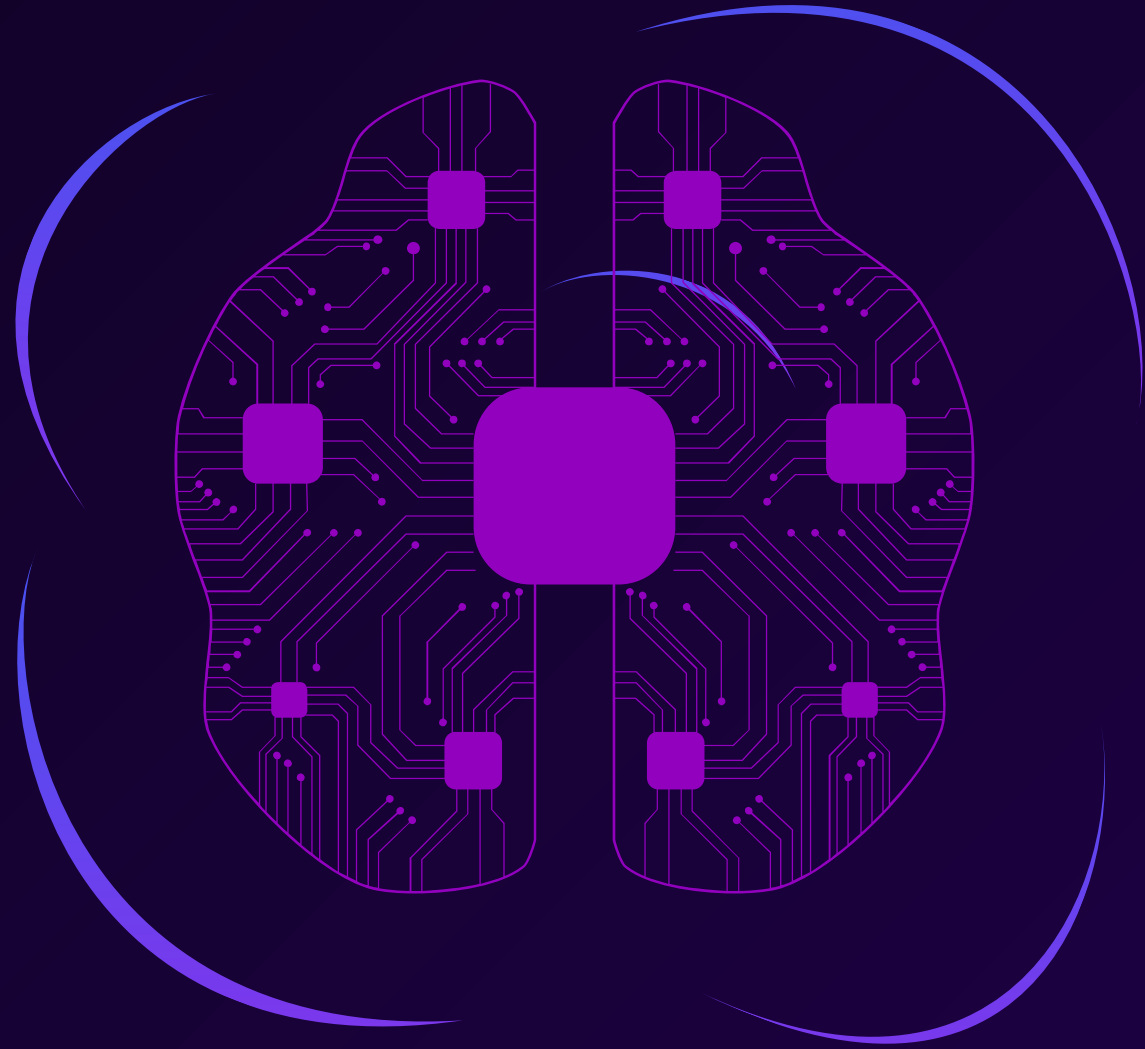
MeetCustomer_FullWidth_2x-18.jpeg.webp

Upload and Classify

Predicted Class: truck

Probabilities:

airplane: 0.06%
automobile: 0.17%
bird: 0.00%
cat: 0.02%
deer: 0.00%
dog: 0.00%
frog: 0.00%
horse: 0.01%
ship: 0.01%
truck: 99.73%



Thank you