Animals-10

Deep Learning Project Report



**Group 1**

Luis

Affiong

Nicholai

**Table of Contents**

[**Project Overview 2**](#_yspy8tt3f0xe)

[**Dataset 2**](#_75rf4vta81ax)

[**1. Description of Chosen CNN Architecture 2**](#_lquiyrwpy6ke)

[**2. Explanation of the Preprocessing Steps 3**](#_kn5uvgo00ajj)

[**3. Details of the Training Process 3**](#_l0td0gtsj7z1)

[**4. Results and Analysis of the Model’s Performance 4**](#_ins8uoxsuvl7)

[**5. Best Model Selected 5**](#_1mil7spy9n8e)

[**6. Insights Gained from the Experimentation Process 6**](#_1x9ryqabo6fs)

[**7. Code Quality 6**](#_as65geht0qi1)

[**8. Visualizations and Diagrams 7**](#_qlwgay1qt0vi)

# 

# Project Overview

This project explores image classification using Convolutional Neural Networks (CNNs). The goal is to build and evaluate models capable of accurately classifying images into predefined categories using both custom CNNs and transfer learning.

# Dataset

**Chosen dataset:** Animals10

**Number of classes:** 10 animal categories

**Number of images:** ~28,000

**Image dimensions:** Varies (standardized to 224x224)

# Description of Chosen CNN Architecture

Two models were implemented:

**Custom CNN:**

A Sequential model with 4 Conv2D layers (32–256 filters, kernel size 3×3), each followed by BatchNormalization and MaxPooling. It ends with GlobalAveragePooling, a 128-unit Dense layer with ReLU, Dropout (0.3), and a final softmax output for 10 classes.

**Transfer Learning (ResNet50):**

Used as a base with ImageNet weights. Layers below index 100 were frozen. A GlobalAveragePooling and Dropout (0.5) were added before the final Dense layer. Fine-tuning was applied in a second training phase.

# Explanation of the Preprocessing Steps

* **Image Resizing**: All images resized to 224x224 to fit model input requirements.
* **Data Augmentation**: Applied RandomFlip, RandomRotation, RandomZoom, and RandomContrast using TensorFlow layers to boost data variability and prevent overfitting.
* **Dataset Splitting**: Split into Train (70%), Validation (15%), and Test (15%) sets, using stratification to maintain class balance.
* **Label Encoding**: Converted categorical labels into integer labels using label encoder.
* **Class Imbalance Handling**: Computed class weights to address imbalance, ensuring minority classes contributed equally to training.

# CNN Model Training Process

**Batch Size**: 32

**Optimizer**: Adam

* Base Learning Rate: 1e-3
* Fine-tuning Learning Rate: 1e-5

**Loss Function**: Sparse Categorical Crossentropy

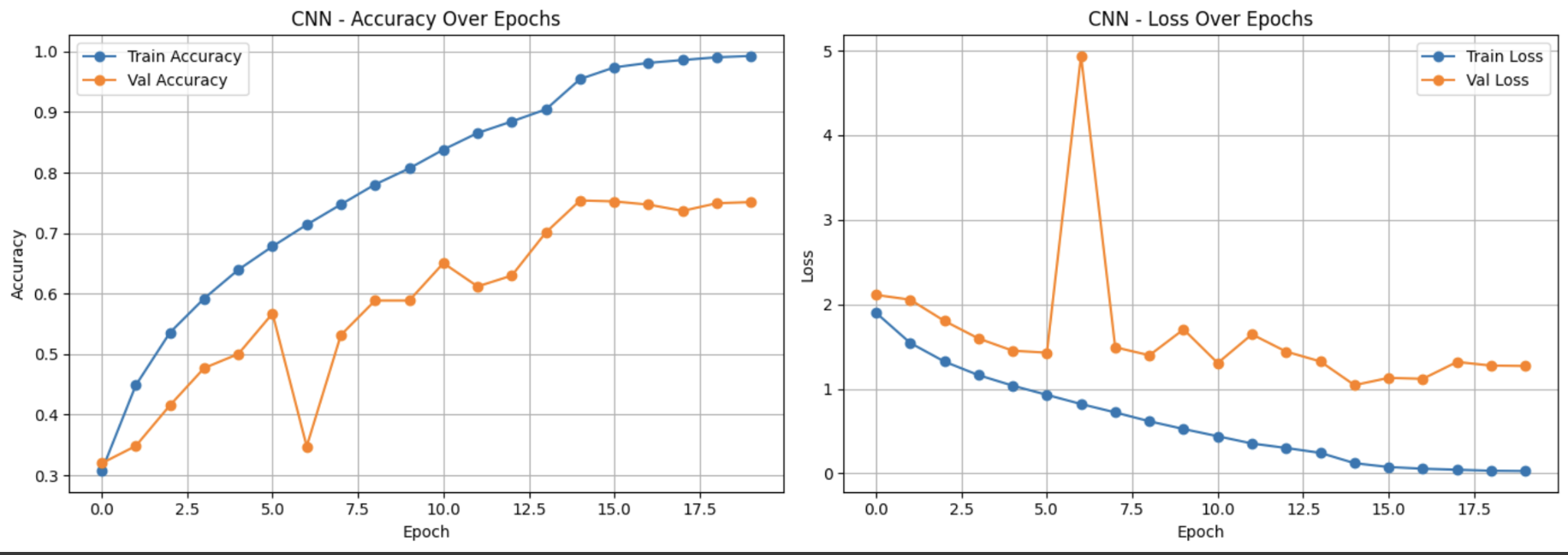
**Callbacks Used**:

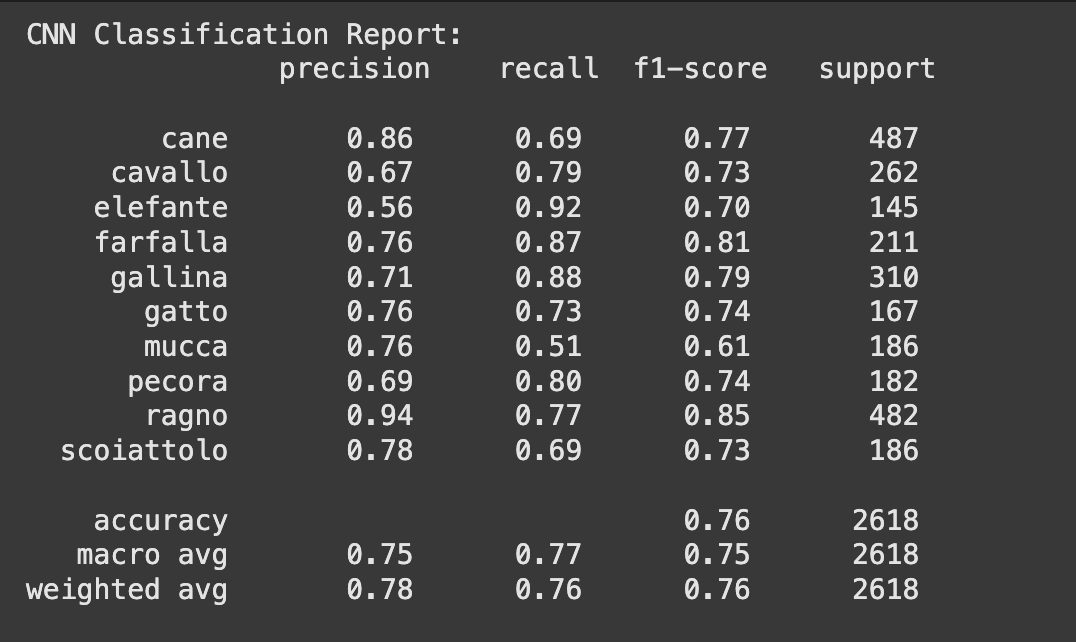
* **EarlyStopping**: Monitored validation loss to prevent overfitting (patience = 5–7 epochs).
* **ReduceLROnPlateau**: Reduced learning rate upon plateau in validation loss (factor=0.2).

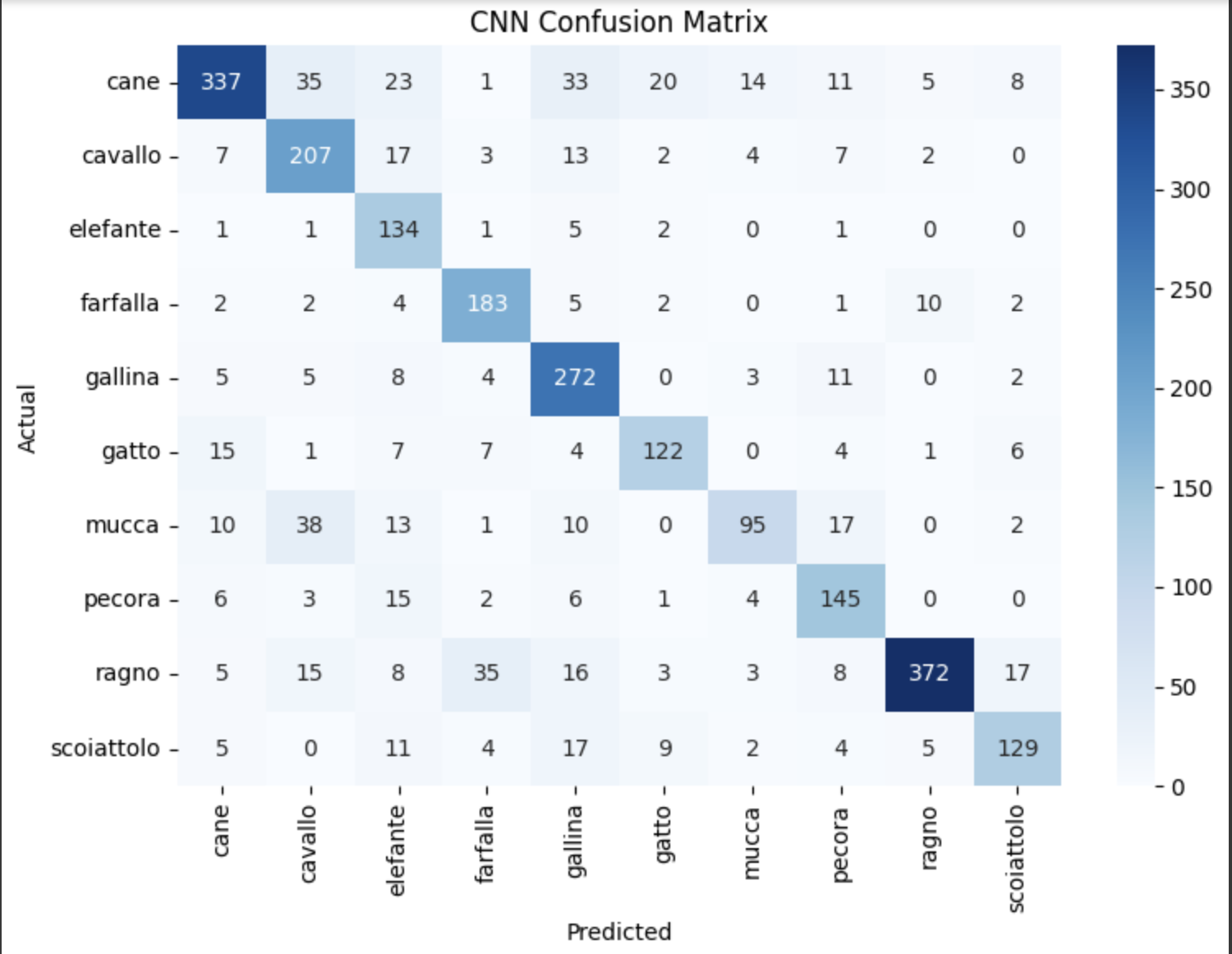
**Training Duration**:

* Custom CNN: Trained for 30 epochs
* ResNet50 Fine-Tuning: Additional 10 epochs after initial freezing phase
* Callbacks: EarlyStopping (patience=5) and ReduceLROnPlateau

# Results and Analysis of the CNN







* Training accuracy improved steadily from 25% to 99%.
* Validation accuracy peaked at ~75%.
* Noticeable overfitting after epoch 15, confirmed by divergence in training/validation curves.
* Early stopping and class weights helped mitigate some overfitting.

**Key Observations**:

* Validation loss initially decreased but later fluctuated despite training accuracy rising, classical overfitting pattern.
* Sharp spikes in validation loss around epoch 6 likely due to unstable learning rate tuning before ReduceLROnPlateau adjustment.

# RESNET50 Transfer Learning Model

To further boost model performance, we leveraged Transfer Learning using a pre-trained ResNet50 model. Instead of training from scratch, we initialized ResNet50 with ImageNet weights and excluded the original fully connected (top) layers. We then added:

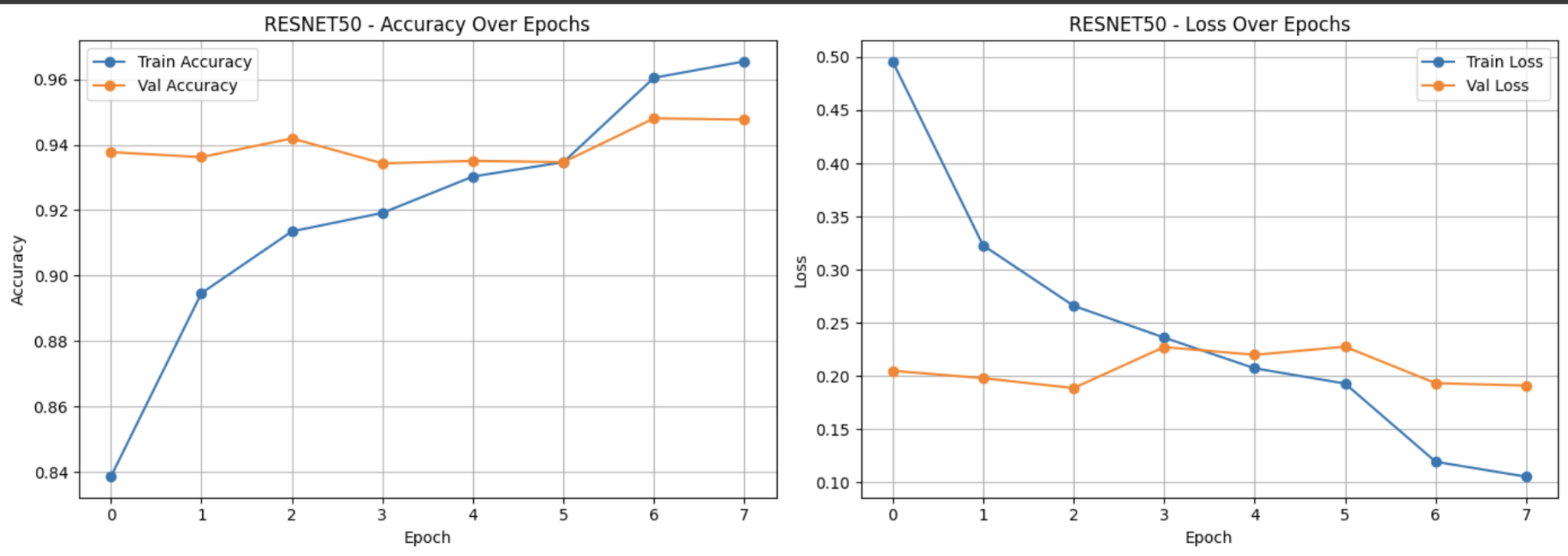
* A GlobalAveragePooling2D layer to reduce parameters and avoid overfitting,
* A Dense layer with 128 units and ReLU activation for non-linearity,
* A Dropout layer with rate 0.3 to improve generalization,
* And a final Dense layer with 10 units for our target classes, using softmax activation.

Initially, the ResNet base was frozen (trainable=False) to act purely as a feature extractor, allowing only the custom layers to learn. After initial training, we unfroze the ResNet base and fine-tuned the model with a very low learning rate (1e-5) to carefully adjust the entire network.

Callbacks like EarlyStopping and ReduceLROnPlateau were again applied to optimize training and prevent overfitting.

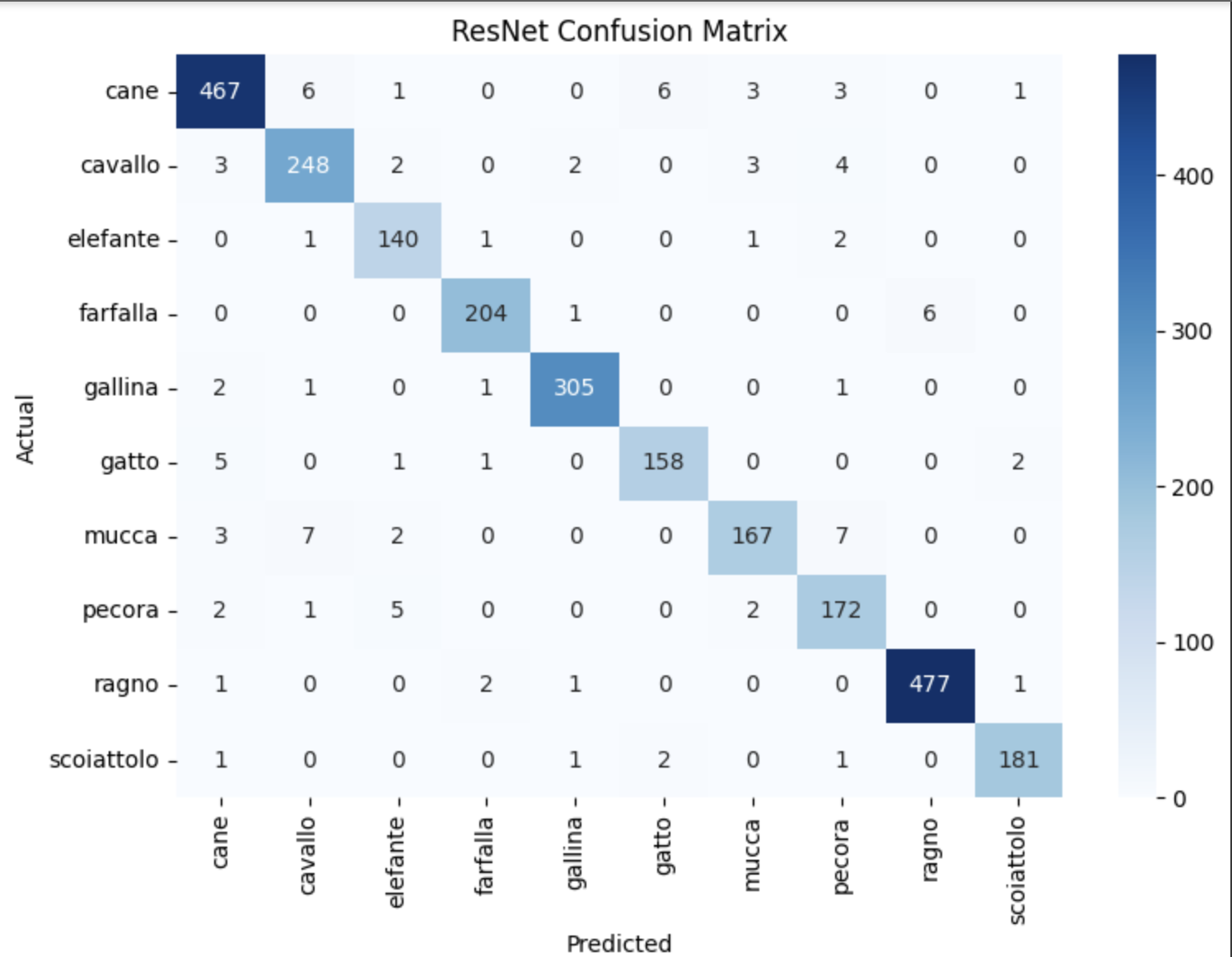
# Results and Analysis of the ResNet50 Model

**Accuracy and loss improved** steadily with **early stopping** and learning rate reduction.



Training was smoother and more stable after finetuning. Validation accuracy rose faster, reaching 96%. A slight discrepancy (or jump) was detected along the accuracy and loss curves during early testing. We believe this was due to the learning rate affecting the validation loss. It was corrected accordingly, as is reflected in the graph above.

The **Confusion Matrix** revealed that some classes were harder to distinguish e.g., 'pecora' and 'mucca' (likely visually similar) than others, as can be seen below.



*Below you can find our* ***Classification Report.*** *It indicates the following:*

* F1 Score: 0.96
* Precision: 0.96
* Recall: 0.96



The classification report for the animal evaluation indicates strong performance across all categories. The model achieved an overall accuracy of 96%, demonstrating its effectiveness in identifying various animals.

The macro and weighted averages of precision (96% and 96% respectively), recall (96% for both), and f1-scores (96% respectively) further underscore the model's robustness across different categories. Overall, the results demonstrate a highly effective classification system for these animals.

# The Best Model Used

The ResNet50 model's application of transfer learning, coupled with the benefits of pretrained weights and fine-tuning, resulted in higher accuracy and better generalization in animal classification tasks. This approach not only improves model performance but also reduces the time and resources required for training, making it a highly effective strategy in the field of deep learning.

# Insights Gained from the Experimentation Process

The experimentation process yielded several key insights:

1. A custom **Convolutional Neural Network (CNN)** served as a baseline but **underperformed** compared to pretrained networks, **emphasizing the benefits** of using **established models** (i.e. ResNet50).
2. Data augmentation proved essential in preventing overfitting by increasing training dataset diversity, which helped the model generalize better to unseen data.
3. Fine-tuning the ResNet model significantly improved performance by adapting the pretrained network to the specific characteristics of the dataset.
4. Addressing Class Imbalance early ensured fair learning across all categories, preventing bias towards majority classes.

# Code Quality

Pylint was used to assess and improve the code quality. The rating improved from 1.85 7.29, indicating a focus on continuous improvement. This shows that the team actively worked on addressing issues and refining the code.

Regularly running Pylint and addressing feedback ensured that the code remained robust and maintainable.

# Models Comparison

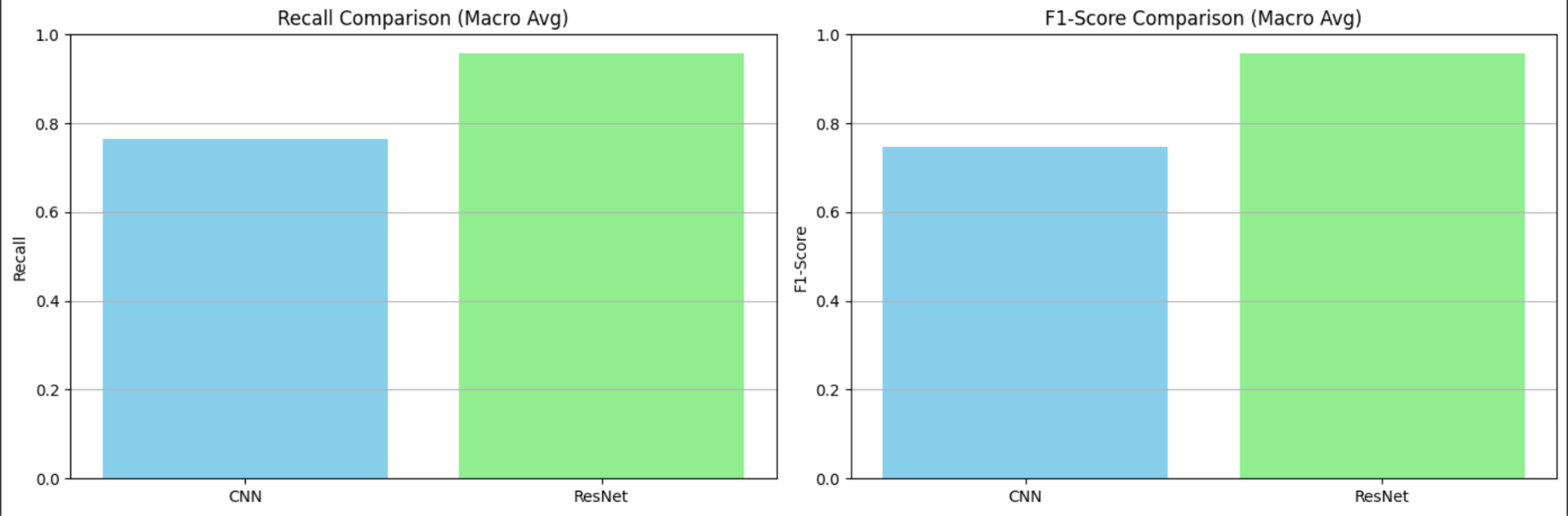
The CNN and ResNet models were evaluated based on their test set performance. Metrics compared include **Loss** and **Precision (Macro Average)**.

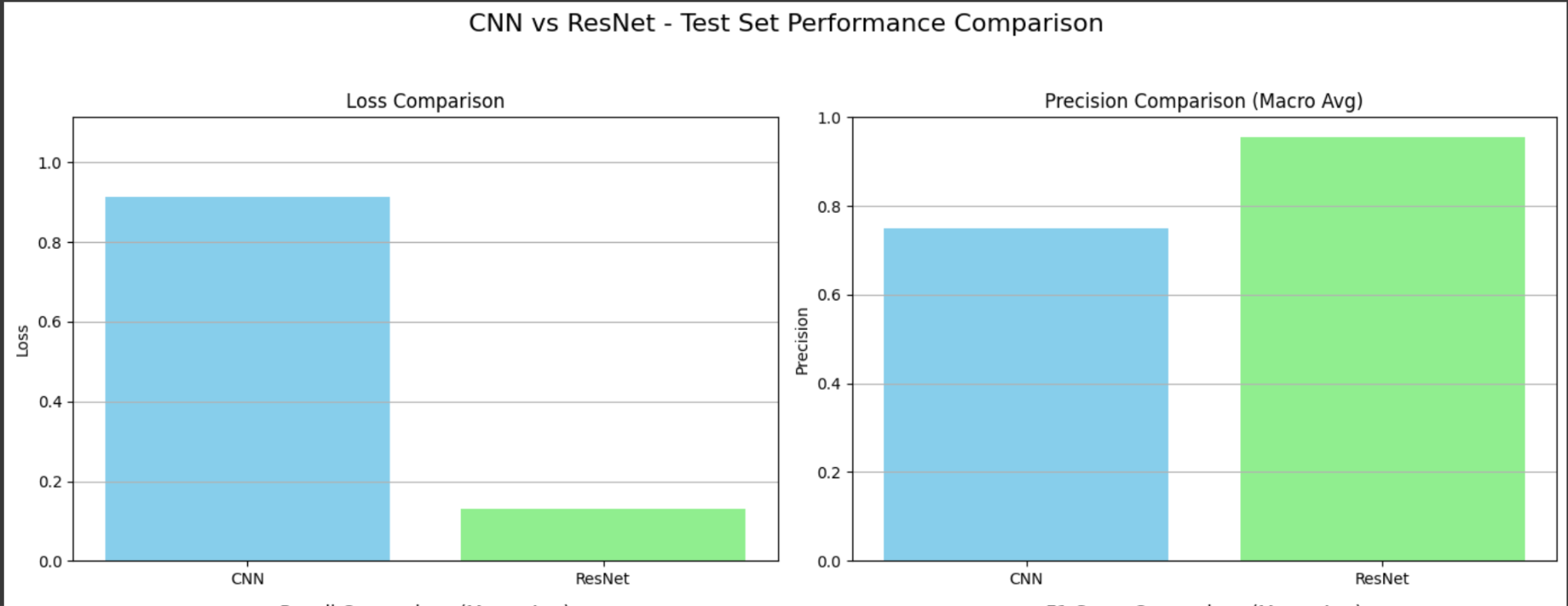
From the comparison:

* **ResNet outperformed the custom CNN** significantly in both **lower loss** and **higher precision**.
* **ResNet achieved a much lower loss** (~0.1) compared to CNN (~0.9).
* **ResNet’s macro-averaged precision** was **around 95%**, while CNN's precision plateaued at **around 75%**.

This demonstrated the power of transfer learning — **ResNet leveraged pre-trained ImageNet knowledge**, leading to faster convergence and much stronger generalization, even with fewer epochs.

Visually, the **loss and precision bar charts** clearly highlight ResNet's superior performance over the custom CNN.





# Summary

The addition of a pre-trained ResNet50 significantly improved classification performance compared to a custom-built CNN. Fine-tuning the ResNet layers after feature extraction further enhanced accuracy without overfitting, confirming the effectiveness of transfer learning for the Animals10 dataset.