Animal10 Dataset Image Classification Using PyTorch

Vansh Datta
1303819

Abstract

This project aims to develop a PyTorch image classification model to classify different animals based on images. The Animals-10 dataset, which is one of the most commonly used computer vision image datasets, is utilized as the foundation. Transfer learning via the application of a pre-trained ResNet18 architecture is utilized alongside extensive data preprocessing and augmentation. Training is conducted for several epochs, with the performance of the model being measured through validation accuracy and confusion matrices. The last model exhibits excellent classification performance, with high accuracy and the capability to generalize well to unseen data.

Introduction

Image classification is the basic issue in computer vision and forms the foundation for more complicated systems such as object detection and scene understanding. In this project, we want to categorize animal images into one of ten classes using deep learning. We chose PyTorch because it is flexible and efficient when building and training neural networks. The Kaggle Animals-10 dataset presents a realistic and challenging classification task, given the variety of the animals and visual similarity between certain classes. Employing transfer learning and proper evaluation techniques, we aim to train an optimal and accurate classifier.

Dataset Description

The dataset used is Animals-10, obtained from Kaggle. It consists of over 26,000 images of 10 different animal categories, including:

- Cat
- Dog
- Horse
- Elephant
- Butterfly
- Chicken
- Cow
- Sheep
- Spider
- Squirrel

The dataset is organized into folders, with each subfolder representing a category. Images vary in size, resolution, and background complexity, providing a robust testbed for classification.

To prepare the dataset:

- All images are resized to a uniform shape (224x224 pixels) to match the input size expected by ResNet models.
- Images are augmented with random transformations like horizontal flipping, rotation, and brightness adjustment to improve model generalization.

The dataset is split into:

- Training set: 80% of the data.
- Validation set: 20% of the data.

This split ensures the model is trained and validated on different subsets, helping assess real-world performance.

Data Preprocessing

Good performance by the model depends on good data preprocessing. The following transformations are applied:

• Resizing: Scales all images to 224x224 pixels.

• Random Horizontal Flip: Included to account for left-right variations.

• Random Rotation: Helps the model handle differently rotated inputs.

• Color Jitter: Modifies brightness and contrast to account for lighting situations.

• Normalization: Converts pixel values to a standard range, helping with quicker convergence during training.

These enhancements encourage dataset diversity and dissuade overfitting. Data is then organized with PyTorch's DataLoader for handling batching and shuffling during training and validation.

Model Architecture

This project uses transfer learning with ResNet18, an ImageNet pre-trained convolutional neural network. Some of the most critical reasons behind this choice are:

- Proven performance on image classification.
- Reduced training time due to pre-trained weights.
- Balanced model complexity and speed.

ResNet18's last layer is replaced so that it outputs 10 class probabilities instead of the default 1000. The earlier layers are frozen or fine-tuned depending upon training performance.

The model is trained with:

- Cross-Entropy Loss: Suitable for multi-class classification.
- Adam Optimizer: Renowned for adaptive learning rate and quick convergence.

The model trains using a GPU (if available), speeding up training significantly.

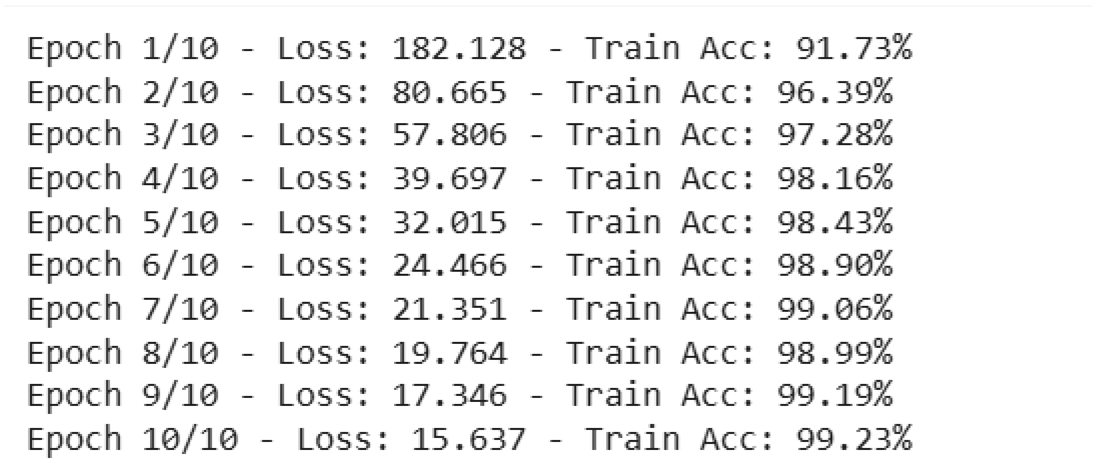
Training Process

Training occurs over a number of epochs (e.g., 10–20), with each epoch including:

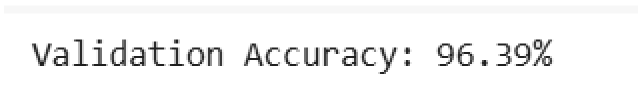
- Forward pass: Model predicts output probabilities.
- Backward pass: Update weights and compute gradients.
- Evaluation on the validation set after each epoch.

Following each epoch:

- Record training accuracy and loss.
- Compute validation accuracy to monitor generalization



```
Epoch 1/10 - Loss: 182.128 - Train Acc: 91.73%
Epoch 2/10 - Loss: 80.665 - Train Acc: 96.39%
Epoch 3/10 - Loss: 57.806 - Train Acc: 97.28%
Epoch 4/10 - Loss: 39.697 - Train Acc: 98.16%
Epoch 5/10 - Loss: 32.015 - Train Acc: 98.43%
Epoch 6/10 - Loss: 24.466 - Train Acc: 98.90%
Epoch 7/10 - Loss: 21.351 - Train Acc: 99.06%
Epoch 8/10 - Loss: 19.764 - Train Acc: 98.99%
Epoch 9/10 - Loss: 17.346 - Train Acc: 99.19%
Epoch 10/10 - Loss: 15.637 - Train Acc: 99.23%
```



```
Validation Accuracy: 96.39%
```

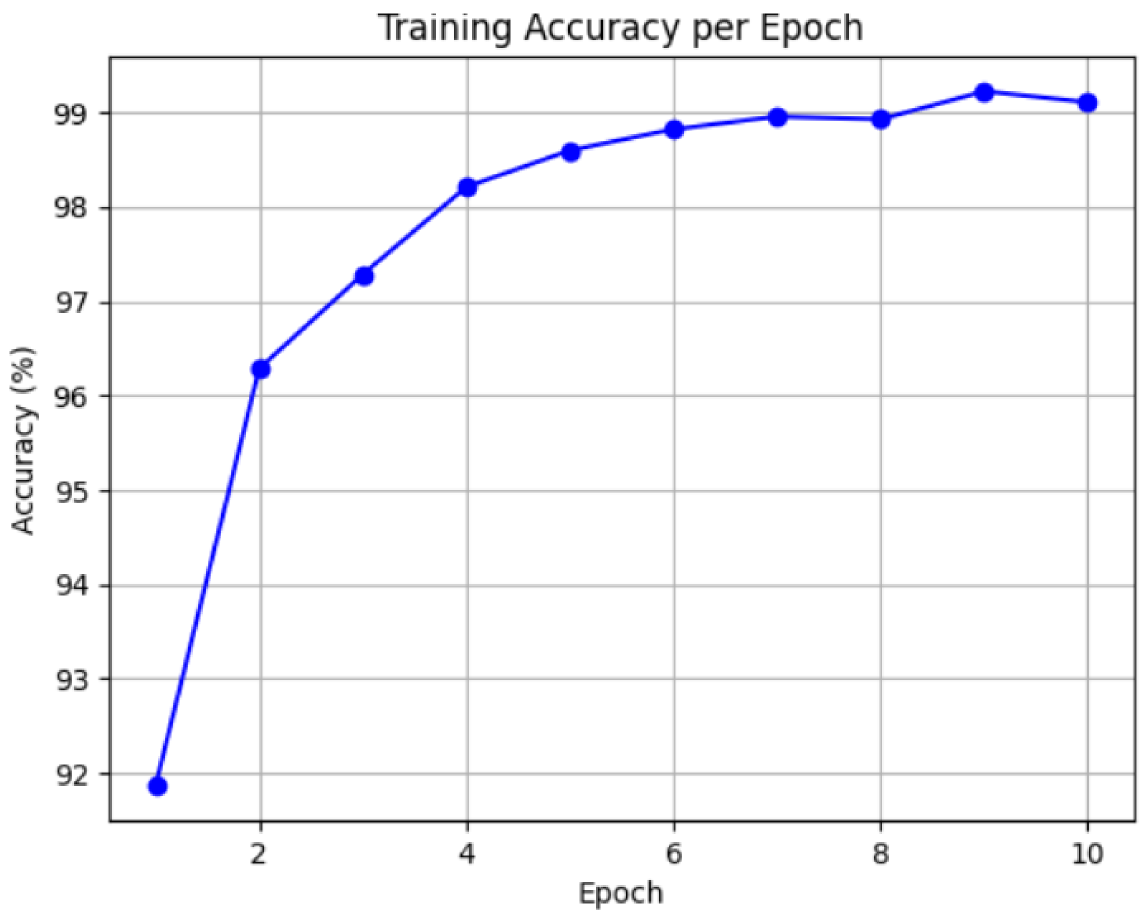
Evaluation and Results

After training, the model's performance is evaluated on the validation set against two main measures:

- Overall Accuracy: Measures how many predictions are correct.
- Confusion Matrix: Illustrates the distribution of correct and incorrect predictions across all classes.

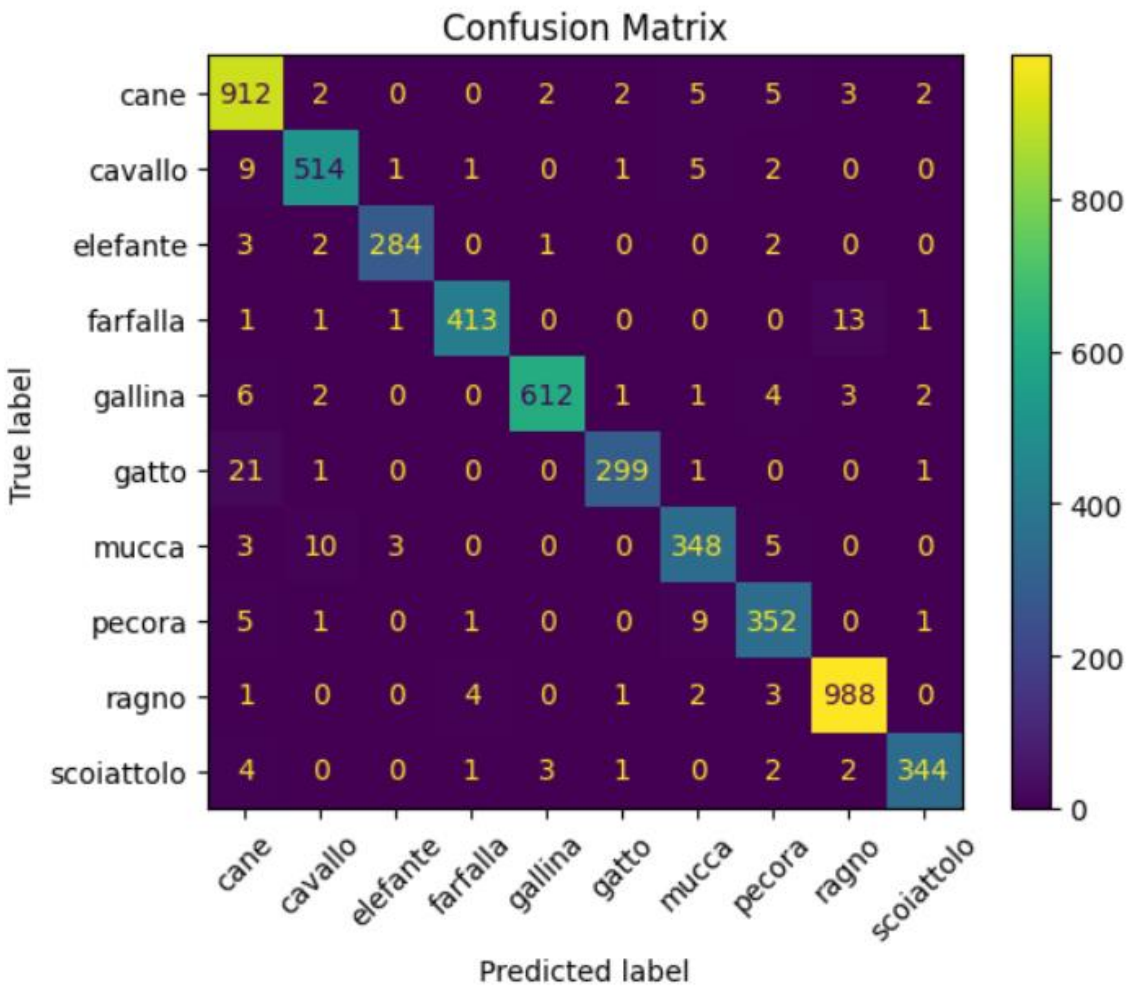
Results Summary :

- Final Validation Accuracy: 96.39%
- Best Epoch: Epoch 10



Most Confused Class Pairs:

- `cane` (dog) misclassified as `gatto` (cat) — 21 times.
- `gatto` (cat) misclassified as `cane` (dog) — 6 times.
- `farfalla` (butterfly) misclassified as `ragno` (spider) — 13 times.
- `mucca` (cow) confused with `pecora` (sheep), and vice versa.
- `scoiattolo` (squirrel) misclassified as `farfalla` and `gallina`.

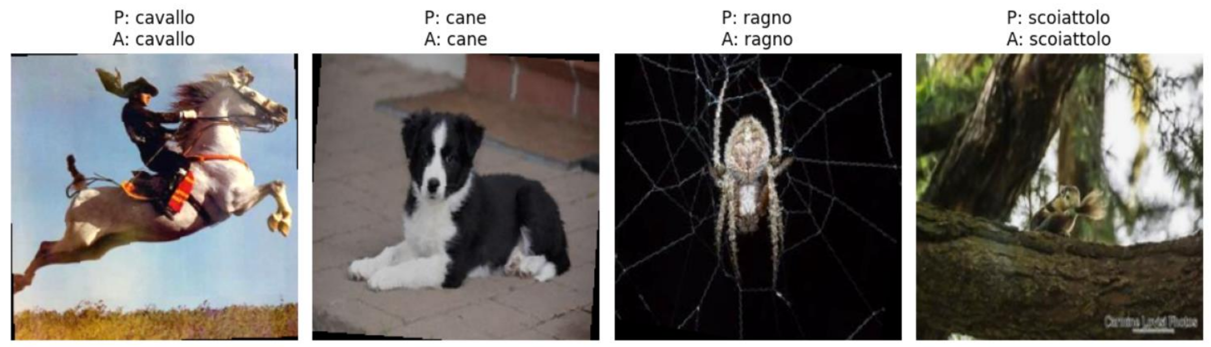


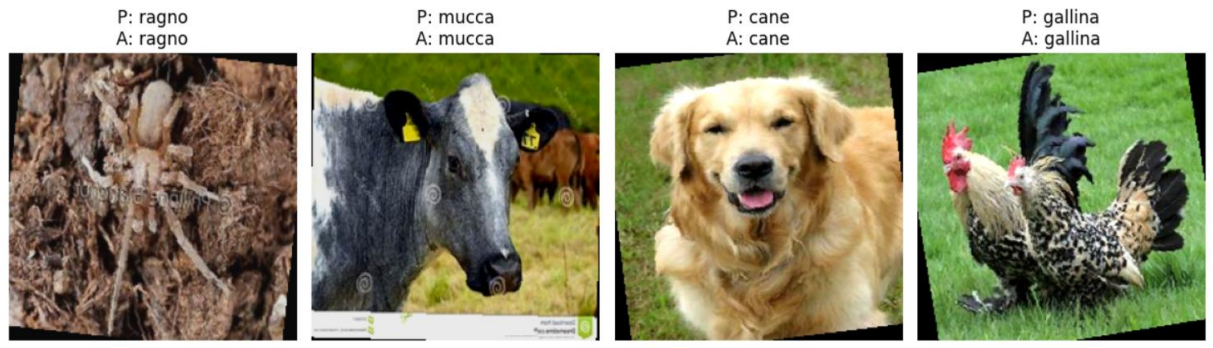
model classifies very well on every class except one, and ragno (spider) and cane (dog) are the top true positives.

Dog vs Cat (cane vs gatto) and Cow vs Horse (mucca vs cavallo) are the most incorrectly classified pairs, which is consistent with usual misclassification patterns in animal data sets.

Accuracy can be adjusted to some degree for gatto, farfalla, and mucca.

Visualization





Saving and Deployment

To utilize the trained model again for future use:

- Weights of the model are saved to a file.
- The model can be reloadable during inference without retraining.

It enables:

- Testing new images in the future.
- Utilization in real-world applications such as animal recognition programs or educational games.

For deployment, a simple interface can be designed for the purpose of uploading pictures and showing the predicted labels. This enables the model to be used outside the notebook.

Conclusion and Future Work

This project demonstrates the power of transfer learning for image classification using PyTorch. With very little training time and an aggressive architecture like ResNet18, high accuracy can be achieved on complex datasets like Animals-10. Proper data augmentation and verification are crucial to high performance.

Key Takeaways:

- Deep models generalize well with data augmentation.
- Pre-trained models reduce training requirements drastically.
- Accuracy can vary between classes due to similarity in images.

Future Improvements:

- Use early stopping and learning rate scheduling.
- Validate on new data to test generalization.