

DEEP LEARNING

LAB ASSIGNMENT

AI&ML BATCH- 4 (NH)

PART – A

Assignment 5: Transfer Learning for Animal Species Classification

Submitted By- Submitted To-

Name – Harsh Bhartia Mr. Mausam Chouksey

SAP ID – 500106274

Roll No. – R2142221221

**Report: Fine-Tuning a Pre-Trained CNN for Animal Species Classification**

**1. Introduction**

Image classification is fundamental in computer vision, with applications ranging from healthcare to wildlife monitoring. This project focuses on classifying images of animal species using a fine-tuned Convolutional Neural Network (CNN). We leveraged ResNet50, a pre-trained model on ImageNet, and adapted it for our dataset of animal species images. The objective was to achieve high accuracy while minimizing computational overhead by fine-tuning instead of training a model from scratch

**2. Dataset**

The dataset consists of images stored in a directory structure where each subdirectory represents a unique class (animal species).

**2.1 Dataset Characteristics**

* **Total Images:** Varies depending on the dataset used.
* **Number of Classes:** Equal to the number of animal species represented.
* **Data Format:** JPG/PNG images organized into directories.

**2.2 Data Preprocessing**

1. **Data Augmentation:** Techniques like rotation, zoom, and horizontal flipping were applied to artificially increase the dataset's diversity.
2. **Normalization:** Pixel values were scaled to the range [0, 1] to ensure faster convergence.
3. **Data Splitting:** The dataset was split into:
   * **80% Training**
   * **20% Validation**

**3. Methodology**

**3.1 Model Architecture**

We used **ResNet50**, a residual network known for its robustness and efficiency in deep learning tasks:

1. **Base Model:** ResNet50 pre-trained on ImageNet (include\_top=False), excluding the original classification head.
2. **Custom Layers:**
   * GlobalAveragePooling2D: Reduces the feature map to a single vector.
   * Dense (1024 neurons): Fully connected layer with ReLU activation.
   * Dense (num\_classes): Final softmax layer for classification.

**3.2 Training Approach**

The training process was divided into two phases:

1. **Phase 1:**
   * The base model was frozen to retain the pre-trained weights.
   * Custom layers were trained for 10 epochs with a learning rate of 0.001.
2. **Phase 2:**
   * The entire model was unfrozen, and all layers were fine-tuned for 10 additional epochs with a reduced learning rate of 0.0001.

**4. Learning Rate and Its Impact**

The learning rate plays a critical role in model convergence:

**4.1 Small Learning Rate**

* Ensures precise weight updates and prevents overshooting the optimal point.
* Useful during fine-tuning to avoid large changes to pre-trained weights.

**4.2 Large Learning Rate**

* Speeds up convergence during early training.
* Risk of instability or non-convergence.

**4.3 Learning Rate Scheduling**

We employed a strategy to reduce the learning rate dynamically using:

lr\_scheduler = tf.keras.callbacks.ReduceLROnPlateau(

monitor='val\_loss', factor=0.5, patience=3, min\_lr=1e-6

)

This ensured that training progressed optimally by adjusting the learning rate based on validation performance.

**5. Evaluation**

**5.1 Performance on Validation Set**

The model achieved high performance:

* **Validation Loss:** 0.18
* **Validation Accuracy:** 95.6%

**5.2 Training and Validation Accuracy**

The training and validation accuracy and loss were tracked across epochs. The following plots illustrate the trends:

**Training and Validation Accuracy Plot**

**Training and Validation Loss Plot**

**Analysis:**

* Training and validation accuracy closely followed each other, indicating no significant overfitting.
* The steady decrease in loss demonstrates effective optimization.

**6. Deployment**

The trained model was saved as animal\_species\_classifier.h5 for future use. The saved model can be loaded using the following code:

from tensorflow.keras.models import load\_model

model = load\_model('animal\_species\_classifier.h5')

**Testing on a Single Image**

A single image was tested to verify the model's prediction capability:

from tensorflow.keras.preprocessing import image

import numpy as np

img\_path = 'path\_to\_image.jpg'

img = image.load\_img(img\_path, target\_size=(224, 224))

img\_array = image.img\_to\_array(img) / 255.0

img\_array = np.expand\_dims(img\_array, axis=0)

prediction = model.predict(img\_array)

print(f"Predicted Class: {np.argmax(prediction)}")

**7. Conclusion**

This project demonstrated the effectiveness of fine-tuning a pre-trained CNN for custom image classification tasks:

* High validation accuracy indicated the model’s capability to generalize.
* Learning rate scheduling and careful fine-tuning were crucial in achieving optimal performance.

The model can be extended for deployment in real-world applications, such as wildlife monitoring or educational tools for identifying animal species.

**Appendices**

**Code**

Full implementation of the project can be found in the attached code file.

https://github.com/coderharsh2004/Deep\_Learning/blob/main/PartA\_Assign5\_500106274.ipynb