Name-Mukesh Avudaiappan

Student Id - 22024161

KERAS ASSIGNMENT

INTRODUCTION:

In the age of artificial intelligence and computer vision, image classification is still an important job with applications ranging from wildlife monitoring to pet recognition. A convolutional neural network (CNN) for image categorization is created using TensorFlow and Keras in the code that is provided. The model is trained using a dataset of animal pictures, primarily of cats, dogs, and wild animals. This study analyzes the code critically in addition to the architecture and training process of the model.

The Model's Architecture:

- → Convolutional layers (Conv2D) and max-pooling layers (MaxPooling2D) make up the CNN architecture, which is used for feature extraction.
- → The classification head is formed by dense layers (Dense), while the global average pooling layer (GlobalAveragePooling2D) minimizes spatial dimensions. There are a total of 362,378 trainable parameters listed in the model summary.
- → The convolutional and pooling layers extract hierarchical characteristics from input images while capturing spatial relationships and patterns.

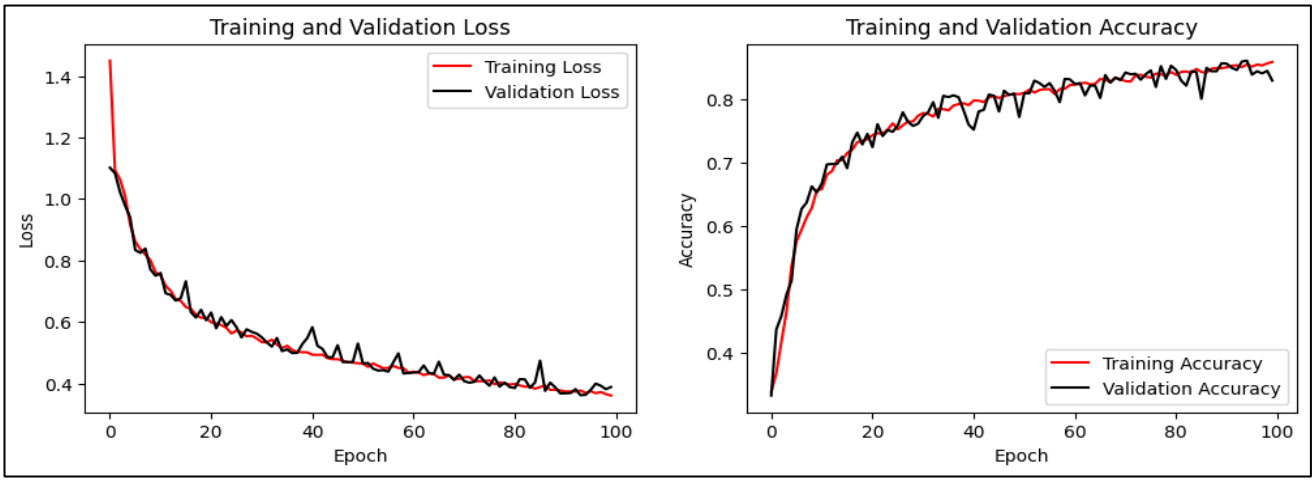
Model: "model"

Layer (type)	Output Shape	Param #
Input (InputLayer)	[(None, 64, 64, 3)]	0
Conv0 (Conv2D)	(None, 64, 64, 16)	448
Conv1 (Conv2D)	(None, 64, 64, 32)	4640
Conv2 (Conv2D)	(None, 64, 64, 32)	9248
Pool1 (MaxPooling2D)	(None, 32, 32, 32)	0
Conv3 (Conv2D)	(None, 32, 32, 64)	18496
Conv4 (Conv2D)	(None, 32, 32, 64)	36928
Pool2 (MaxPooling2D)	(None, 16, 16, 64)	0
Conv5 (Conv2D)	(None, 16, 16, 128)	73856
Conv6 (Conv2D)	(None, 16, 16, 128)	147584
GlobalPool (GlobalAveragePooling2D)	(None, 128)	0
FC1 (Dense)	(None, 512)	66048
Output (Dense)	(None, 10)	5130

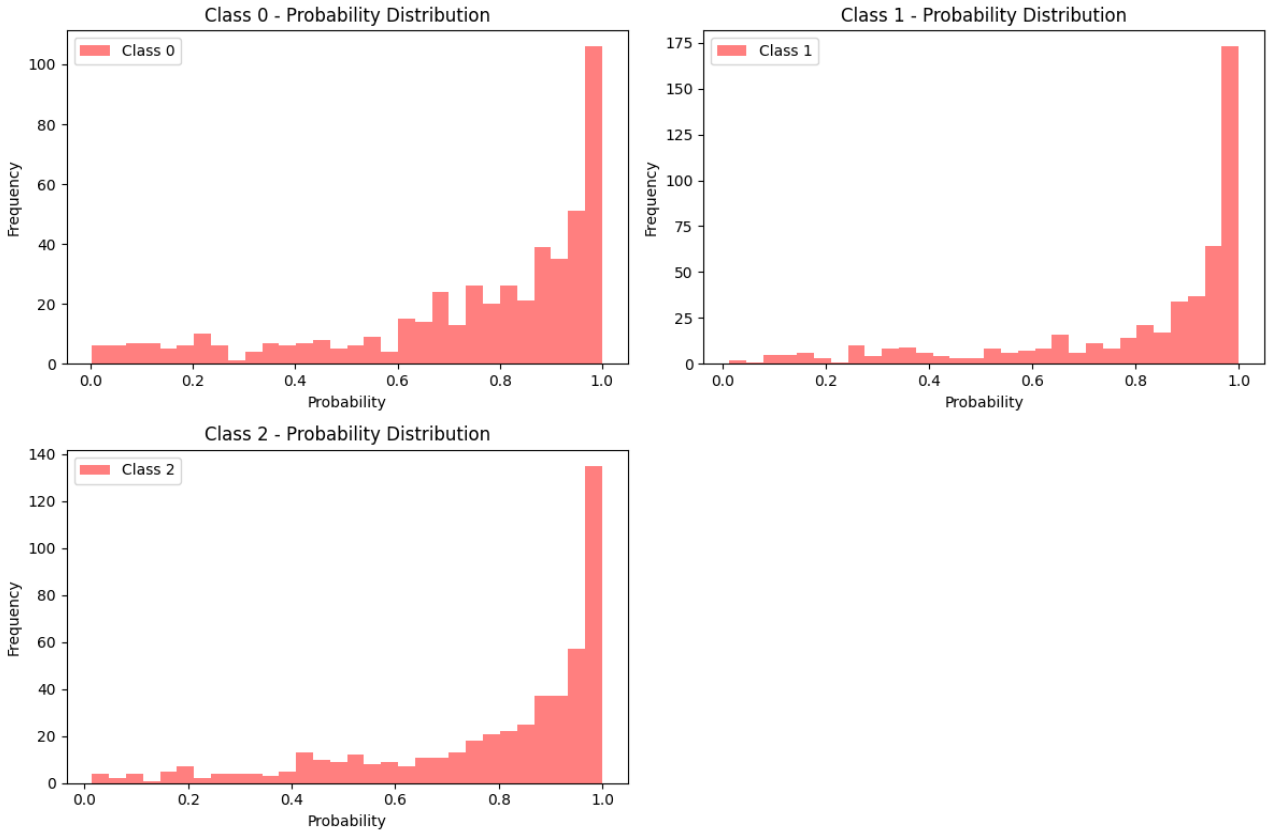
Data Processing: The dataset is loaded and preprocessed effectively by the **tf.data** API. While the **img_process** method resizes images to 64x64 pixels and converts them to float32, the **parse_image** function reads and decodes images. The loading of datasets for training and validation sets is optimized using ***caching, shuffling, batching, and prefetching.***

Training Process: The training procedure is thorough, including model checkpointing mechanisms, dynamic learning rates, and instructive visualizations to create a well-performing and resilient animal categorization model.. The implementation of **ModelCheckpoint** and **LearningRateScheduler** callbacks allows for the saving of optimal weights determined by validation accuracy and training rate adjustment.

Evaluation and Results: Plotting of the accuracy and loss during training and validation is done over epochs. The best weights are loaded for the final evaluation after the model has been assessed on the validation set. The evaluation's findings—including accuracy and loss—are shown. Visualizing the probability distributions for each class offers insights into the confidence levels of the model for various classes.



Analysis: The model's decreased training and validation loss indicates that it is learning effectively. The model exhibits good generalization to the validation set, as seen by the accuracy curves. The probability distribution plots show how confident the model is in its ability to forecast each class.



Conclusion:

As a result, the code serves as an excellent example of how to construct, train, and assess CNNs for image classification application. It offers a strong platform for more experimentation and performance enhancement. The model is more stable and more broadly applicable to the **LearningRateScheduler** and **ModelCheckpoin**t callbacks. The probability istribution plots shed light on the model's confidence in its predictions, providing valuable insights into its decision-making process.