Mini Project - Cat Dog Classifier

Aims

This project realizes a Cat Dog Classifier. This classifier will help you quickly identify whether a given image contains a cat or a dog. Using a combination of image recognition algorithms and trained AI models, this classifier can accurately distinguish between a cat or a dog, even when the differences are subtle. Hope you find this application useful and accurate :)

Methods

I mainly employ transfer learning in this project [1]. I define a sequential model using the Keras API from the TensorFlow library. The model is composed of a pre-trained base model, followed by two layers of trainable dense neural networks.

The `base_model` is passed as a parameter to the `Sequential` function, which creates the initial model architecture. The base model can be any pre-trained model that has been trained on a large dataset, such as VGG16 or ResNet. The base model acts as a feature extractor and is used to extract relevant features from the input data. By using a pre-trained model, the training process can be faster and more effective.

After the base model, a `GlobalAvgPool2D` layer is added to the model. This layer computes the average of each feature map in the output of the base model, resulting in a single feature vector per image. This process is known as global average pooling, and it reduces the number of parameters in the model, making it less prone to overfitting.

The next layer added to the model is a `Dense` layer with 128 neurons and a ReLU activation function [2]. This layer is responsible for learning high-level representations of the input data based on the features extracted by the base model. ReLU is a commonly used activation function that has been shown to perform well in deep learning models.

Finally, a `Dense` layer with a single neuron and a sigmoid activation function is added to the model. This layer is the output layer of the model, and it produces a binary classification output, indicating whether the input image belongs to a particular class or not. The sigmoid activation function is commonly used in binary classification problems as it outputs a probability value between 0 and 1.

After defining the model architecture and compiling the model, the next lines of code train the model using the `fit` function.

The `fit` function trains the model for a fixed number of epochs (specified as 10 in this case), using the specified training data, `train_generator`, and validation data, `test_generator`. The `steps_per_epoch` parameter specifies the number of batches of samples to use in each epoch, and the validation data is used to evaluate the model's performance on unseen data.

During training, the model uses the Adam optimizer and binary cross-entropy loss function to optimize the model's weights based on the training data. The `accuracy` metric is used to evaluate the model's performance during training.

Overall, the binary classification model using a pre-trained base model and two dense layers. The model is designed to be efficient and less prone to overfitting by using a pre-trained base model and global average pooling layer. Finally, the base model is set to be non-trainable, and the model is compiled with Adam optimizer and binary cross-entropy loss function. Next, I train the binary classification model using the specified training data and validation data, for a fixed number of epochs, with a fixed number of steps per epoch. The Adam optimizer and binary cross-entropy loss function are used to optimize the model's weights, and accuracy is used to evaluate the model's performance.

Evaluation of Results

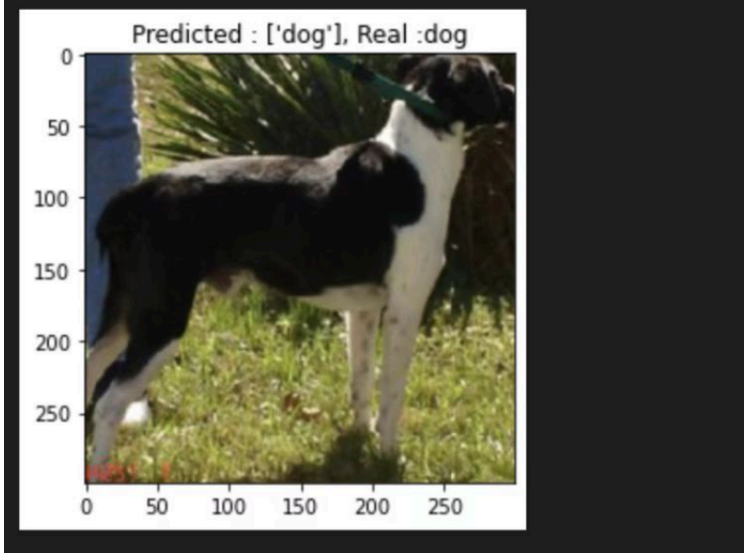
```
Epoch 1/10
25/25 [==============================] - 67s 3s/step - loss: 0.0276 - accuracy: 0.9912 - val_loss: 0.0240 - val_accuracy: 0.9930
Epoch 2/10
25/25 [==============================] - 65s 3s/step - loss: 0.0150 - accuracy: 0.9962 - val_loss: 0.0315 - val_accuracy: 0.9925
Epoch 3/10
25/25 [==============================] - 63s 3s/step - loss: 0.0277 - accuracy: 0.9950 - val_loss: 0.0350 - val_accuracy: 0.9890
Epoch 4/10
25/25 [==============================] - 65s 3s/step - loss: 0.0595 - accuracy: 0.9850 - val_loss: 0.0559 - val_accuracy: 0.9860
Epoch 5/10
25/25 [==============================] - 65s 3s/step - loss: 0.0442 - accuracy: 0.9850 - val_loss: 0.0330 - val_accuracy: 0.9925
Epoch 6/10
```

```
25/25 [==============================] - 65s 3s/step - loss: 0.0270 - accuracy: 0.9912 - val_loss: 0.0260 -
val_accuracy: 0.9930
Epoch 7/10
25/25 [==============================] - 64s 3s/step - loss: 0.0075 - accuracy: 0.9962 - val_loss: 0.0252 -
val_accuracy: 0.9920
Epoch 8/10
25/25 [==============================] - 64s 3s/step - loss: 0.0215 - accuracy: 0.9925 - val_loss: 0.0349 -
val_accuracy: 0.9895
Epoch 9/10
25/25 [==============================] - 63s 3s/step - loss: 0.0254 - accuracy: 0.9950 - val_loss: 0.0269 -
val_accuracy: 0.9940
Epoch 10/10
25/25 [==============================] - 62s 3s/step - loss: 0.0077 - accuracy: 0.9975 - val_loss: 0.0262 -
val_accuracy: 0.9925
```

As the log says, the accuracy is constantly higher than 99%, which is quite ideal.

Prediction Example

```
Text(0.5, 1.0, "Predicted : ['dog'], Real :dog")
```



References

1. F. Zhuang et al., "A Comprehensive Survey on Transfer Learning," in Proceedings of the IEEE, vol. 109, no. 1, pp. 43-76, Jan. 2021, doi: 10.1109/JPROC.2020.3004555.
2. P. Dileep, D. Das and P. K. Bora, "Dense Layer Dropout Based CNN Architecture for Automatic Modulation Classification," 2020 National Conference on Communications (NCC), Kharagpur, India, 2020, pp. 1-5, doi: 10.1109/NCC48643.2020.9055989.