Project Title: Cats vs. Dogs Image Classification Using Deep Learning

Overview

This Deep Learning project, titled "Cats vs. Dogs Image Classification," aims to develop a powerful image classification model capable of distinguishing between images of cats and dogs. The project harnesses the "dogs-vs-cats" dataset, which contains 25,000 labeled images of cats and dogs, distributed across two primary folders: "train" and "test1". This collaborative endeavor involved two contributors, Asolat and Yusuf, and was undertaken as part of a Deep Learning lesson to gain hands-on experience in neural networks, computer vision, and model evaluation.

Purpose of the Project

The primary objectives of this project are as follows:

1. Image Classification: Develop a deep learning model that can accurately classify images as either cats or dogs, showcasing the practical application of deep learning in computer vision tasks.

2. Educational Endeavor: Provide a learning opportunity for both contributors, Asolat and Yusuf, to gain insights into deep learning concepts, data preprocessing, model architecture, and evaluation techniques.

Dataset

The project relies on the "dogs-vs-cats" dataset, a well-established dataset in the field of computer vision. Key dataset details include:

- Dataset Name: dogs-vs-cats

- Image Dimensions: 128x128 pixels

- Image Channels: 3 (RGB)

Contributors and Roles

Asolat and Yusuf played pivotal roles in this project, each contributing their expertise to various facets of the project:

Asolat's Contributions

- Data Preprocessing: Asolat executed data preprocessing tasks, including data augmentation, resizing, and normalization, to prepare the dataset for model training.

- Model Architecture: Asolat designed and implemented the Convolutional Neural Network (CNN) architecture, a vital component for image classification.

- Model Training: Asolat carried out the model training using the processed dataset, optimizing model parameters for accuracy.

- Model Evaluation: Asolat conducted model evaluation, measuring key metrics such as accuracy and making adjustments to enhance performance.

Yusuf's Contributions

- Data Exploration: Yusuf explored the dataset, gaining valuable insights through data visualization and understanding the underlying characteristics of the images.

- Data Splitting: Yusuf divided the dataset into training and validation subsets, a crucial step in training and evaluating the model effectively.

- Hyperparameter Tuning: Yusuf experimented with hyperparameters, fine-tuning the model for optimal performance.

- Documentation: Yusuf documented the project's progression, making sure that the project's results and methods were well-documented for future reference.

Model Architecture

The deep learning model architecture follows the Convolutional Neural Network (CNN) paradigm, consisting of convolutional layers, max-pooling layers, batch normalization, dropout layers, and fully connected layers. The model is designed to capture and extract features from the input images, subsequently making predictions based on these features.

Model Code

Here's a snippet of the code that defines the model:

```python

from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense, Activation, BatchNormalization

model = Sequential()

Convolutional Layers

model.add(Conv2D(32, (3, 3), activation='relu', input\_shape=(Image\_Width, Image\_Height, Image\_Channels)))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

Additional Convolutional Layers

...

Fully Connected Layers

model.add(Flatten())

model.add(Dense(512, activation='relu'))

model.add(BatchNormalization())

model.add(Dropout(0.5))

model.add(Dense(2, activation='softmax'))

Model Compilation

model.compile(loss='categorical\_crossentropy', optimizer='rmsprop', metrics=['accuracy'])

```

Model Training and Evaluation

The model is trained using data generators for preprocessing and augmentation of both training and validation data. Callbacks, including Early Stopping and Learning Rate Reduction, are employed to enhance training efficiency and prevent overfitting.

Model Training Code

Here's a snippet of the code responsible for model training:

```python

epochs = 10

history = model.fit\_generator(

train\_generator,

epochs=epochs,

validation\_data=validation\_generator,

validation\_steps=total\_validate//batch\_size,

steps\_per\_epoch=total\_train//batch\_size,

callbacks=callbacks

)

```

Model Saving and Testing

After successful training, the model is saved for future use. Testing is performed on the "test1" dataset, and predictions are visualized.

Model Saving and Testing Code

Here's a snippet of the code for saving the model and performing testing:

```python

model.save("model1\_catsVSdogs\_10epoch.h5")

...

predict = model.predict\_generator(test\_generator, steps=np.ceil(nb\_samples/batch\_size))

...

```

Custom Image Prediction

The project also includes functionality for making predictions on custom images, allowing users to load their own images and obtain predictions on whether the image contains a cat or a dog.

Conclusion

In conclusion, this Deep Learning project showcases the development of an image classification model for distinguishing between cats and dogs. Through collaborative efforts and extensive documentation, contributors Asolat and Yusuf gained practical experience in various facets of deep learning and computer vision. The code and project files provided can be utilized for implementing and replicating the project.