Name: **Kanaishk Garg**

ID: rishi.gupta90@gmail.com

Title: **Deep Learning with Keras and Tensorflow Project: Pet Classification**

Code Summary:

1. The code begins by importing the necessary libraries for the task, such as TensorFlow and Keras, as well as other utilities like **pydot** and **cv2**. It also sets up the environment and handles any warning suppression.
2. Configuration parameters are defined to control various aspects of the model. These include the size of fully connected layers (**fc_size**), dropout probability (**dropout_prob**), the number of iterations to train the model (**num_iterations**), and the strength of L2 regularization (**l2_regularization**).
3. The file paths for the dataset, model, and graphs are specified using the current working directory (**cwd**) and appropriate subdirectories.
4. An **ImageDataGenerator** object called **train_datagen** is created to apply data augmentation techniques during the training phase. This includes brightness adjustments, shearing, zooming, flipping, rotation, and shifting. The **test_datagen** object is also created, which will be used for testing without augmentation.
5. Two data generators, **train_dataset** and **valid_dataset**, are initialized using the **train_datagen** object. These generators load images from the training directory, resize them to the desired dimensions (225x225 pixels), and create mini-batches of images along with their corresponding labels. The **valid_dataset** is a subset of the training data and is used for validation during training.
6. Another data generator, **test_dataset**, is created using the **test_datagen** object. This generator loads images from the testing directory and prepares them for evaluation.
7. The **model_def** function is defined to construct the architecture of the convolutional neural network (CNN). It comprises several layers, including convolutional layers with ReLU activation, max pooling layers, a flatten layer, fully connected layers with ReLU activation, dropout regularization, and a final softmax output layer. The model is compiled with the Adam optimizer and the categorical cross-entropy loss function.
8. The **model_testing** function takes a model, training dataset, validation dataset, and the number of iterations as inputs. Within this function, a learning rate scheduler using the **ReduceLROnPlateau** callback is created. The model is trained for the specified number of iterations using the training dataset while monitoring the validation accuracy. After training, the model is saved, and the accuracy and loss curves are plotted using Matplotlib.
9. A loop iterates over the **num_iterations** list, creating a new model in each iteration. The **model_testing** function is then called to train and evaluate the model using the

training and validation datasets. Finally, the performance of the model is evaluated on the separate test dataset, and the test loss and accuracy are printed.

In summary, the code implements a pet classification CNN model using TensorFlow and Keras. It includes data augmentation during training, validation using a subset of the training data, and evaluation on a separate test dataset. The model's architecture is defined, trained for multiple iterations, and its performance is assessed using accuracy and loss metrics.

While I was able to minimize the loss quite well the accuracy is not high. Due to being limited in layers we can implement and the small size of the dataset nothing can be done to improve it without major changes to either the structure or the dataset.

During validation model is able to achieve accuracy above 85% but testing shows its only 60% and decrease with increasing number of iterations. This shows model is being overfitted and unable to converge properly. Also softmax activation function does not perform as well as sigmoid activation function. Regularization using l2 helped but it cannot change the reality that dataset is too small. I also tried adding grayscale samples of training images along with normal ones but that also does not improve performance probably due to lack of diversity among the features extracted

Overall Model can be improved a lot more if some restrictions of project are relaxed.