The dogs and cats dataset is composed of 1000 dog and 1000 cat images provided by Microsoft Research, and were originally used in a 2013 Kaggle classification competition. This assignment will evaluate the classification of the images with the use of convolutional neural networks (CNNs) within Python TensorFlow.

The images were provided as 64 x 64 bit grayscale in a .npy file format, and combined into a single array. The array was reshaped to 4096 inputs and the Scikit-learn MinMaxScaler was used to transform the features to be within the range of zero and one. The labels for the images were defined as 0 for cats and 1 for dogs. The data was split so there were 1600 images in the training set and 400 images in the test set.

The data sets were examined in TensorFlow via two different ways. First, the data was classified via plain TensorFlow in a 2x2 completely crossed design benchmark experiment. This was a simple comparison of CNNs which examined the accuracy between two and five convolutional layers. For each layer the kernel size was evaluated at 3x3x1 and 5x5x1 each. Each model configuration was tested three times, as model performance would vary due to different local optima obtained with each assessment. The processing time and accuracy for the test data sets were recorded and compared. Models with two convolutional layers were pooled after each layer. After the second pooling, the data was reshaped and processed by a fully connected (dense) layer with a 50% dropout rate to minimize overfitting. The data was processed by another fully connected (dense) layer for output. In models with five convolutional layers, three convolutional layers were stacked one after another after the second pooling. A third pooling layer followed the convolutional stack and the data was reshaped and processed by a fully connected (dense) layer with a 50% dropout rate. The data was processed by another fully connected (dense) layer for output. The rectifying linear unit (relu) activation function was used in all layers which required an activation function. Keras was used in TensorFlow to create a model, although not in a 2x2 completely crossed design benchmark study.

In the benchmark experiment, the processing time for each model configuration remained fairly consistent and within 20 seconds of each other. A maximum training accuracy of 100% was achieved with all model training runs except for the last test of the model with 5 convolutional layers with a kernel size of five, which only achieved a maximum training accuracy of 60% (data not in summary table). This model obtained the lowest test accuracy (48% in assessment #3). During this assessment, the loss was consistent and did not improve over time, and the accuracy fluctuated throughout the training process. These results indicate that the combination of 5 convolutional layers with a kernel size of 5 could result in a model that no longer allowed the algorithm to learn useful features to make accurate classifications.

Interestingly, the model configuration with 5 convolutional layers and a kernel size of 3 had the lowest processing times. This model configuration also obtained the highest average accuracy (66.9%) with the test set. Increasing the kernel size while holding the number of convolutional layers constant increased the computational time and decreased the average test accuracy.

Future explorations in order to obtain more accurate models should include hyperparameter optimization for both the optimizer and model, and data augmentation given the small size the train/test sets (1600 and 400 images, respectively). Optimization could be pursued utilizing GridSearchCV or Hyperas. These aspects were initially assessed but resulted in no implementation due to: 1) Hyperas appears to not work well in IPython notebooks (https://github.com/maxpumperla/hyperas/issues/24); and 2) coding errors that could not be resolved due to limited time.

The Keras model contained three convolutional layers with a kernel size of three and relu activation function, and each layer was followed by a pooling layer. This architecture was followed by fully connected layer, a dropout rate of 50% to minimize overfitting and a final fully connected layer with a softmax activation function. This model consistently performed well with an average 68.3% test accuracy, although the processing time exceeded seven minutes.

From the benchmark study, it can be concluded that additional layers do not always result in higher test accuracy scores, especially if the kernel size is increased. The type of CNN architecture required will depend on the type and quantity of data used. Models should be tested multiple times to account and assess for different local optima.