The first step is import the necessary libraries required.

Then I like to check if my gpu is available before proceeding.

A black screen with white text

Description automatically generated

Next the data directories are specified, image size and depth is as well.

Then we load all image filenames from the dataset and separate them into cats and dogs based on the filenames.

Finally that dataset is split into training and validation sets

A computer screen shot of text

Description automatically generated

Next the unique number of classes is determined and the batch size and training parameters are set.

Then the class weights are computed to handle any imbalance.

A screen shot of a computer program

Description automatically generated

Then the appropriate activation function, class mode and loss function are determined.

A screen shot of a computer program

Description automatically generated

Then using imagedatagenerator to augment the images and create training and validation generators

A screen shot of a computer program

Description automatically generated

Then vgg16 is built with custom layers added on top

**Explanation:**

* **Input Layer:** We define the input shape based on the image dimensions.
* **Base Model:** We load the pre-trained VGG16 model without the top classification layers.
* **Custom Layers:** We add BatchNormalization, Flatten, Dense, and Dropout layers to the base model.
* **Freezing Layers:** We freeze the earlier layers of the base model to retain the pre-trained weights.
* **Output Layer:** We add a final Dense layer with activation based on the number of classes.
* **Model Summary:** We print the summary to see the total parameters and trainable parameters.

A computer screen shot of text

Description automatically generated

Then callbacks are created for tensorboard, and earlystopping

A computer screen shot of text

Description automatically generated

Model is then trained with the model being saved   
A computer screen shot of a program code

Description automatically generated

Then the training and validation loss/accuracy are displayed on a graph as well as a confusion matrix.  
  
Next we implement a custom ImageLoader class to have more control over data loading and augmentation. The cell is to large to screenshot so here is the explanation   
**Explanation:**

* **Inheritance from Sequence:** Ensures compatibility with Keras data generators.
* **Initialization:** Stores parameters like dataframe, directory, target size, batch size, label mapping, etc.
* **\_\_len\_\_ Method:** Returns the total number of batches per epoch.
* **\_\_getitem\_\_ Method:** Generates one batch of data, including loading images, resizing, normalizing, and applying augmentations.
* **on\_epoch\_end Method:** Optionally shuffles the data indices after each epoch.
* **apply\_augmentations Method:** Applies random transformations like rotation, flipping, zooming, and brightness adjustments.

Then the model is retrained using the custom dataloader. Then the model is evaluated the same way as earlier. Achieving the same results.