**Tools/Frameworks used:**

1. TensorFlow: An open souce library which specializes in Machine Learning Applications, especially Deep Learning.
2. Docker Toolbox: It is a minimal Virtual Machine with all the dependences needed to build TensorFlow Serving. The following command was used through docker to install TensorFlow:

***docker run –it gcr.io/tensorflow/tensorflow:latest-devel***

1. Python 3.5: Supports tensorflow library.

**Solution Description:**

The provided images were first divided into two folders of categories true and false. I used the Inception V3 Network that had been trained originally for Large Visual Recognition and can differentiate between around 1000 different classes. This helped in the pre-training process. Now we have the data and the trainer. The following script was run in Docker to remove the old final layer and train on the data set provided so that we are able to classify into true and false categories.

***python tensorflow/tensorflow/examples/image\_retraining/retrain.py \***

***--bottleneck\_dir=/tf\_files/bottlenecks \***

***--how\_many\_training\_steps 500 \***

***--model\_dir=/tf\_files/inception \***

***--output\_graph=/tf\_files/retrained\_graph.pb \***

***--output\_labels=/tf\_files/retrained\_labels.txt \***

***--image\_dir /tf\_files/train\_img\_dir***

We train the final layer of the Neural Network only because the starting and the hidden layers have already been trained in the Inception Network. It used convolutional neural networks with cross validation to train on the training set. This script runs for 500 steps and took around 20 minutes to complete. Here bottleneck is used to refer to the layer just before the final layer. Finding the hidden layers before the bottleneck is why the process is time consuming. I’ve attached images of the training process with the following data:

Final test accuracy: 89.2%

When two images were used as test data to test our model, I got the following accuracy scores:

1. For a false image:

False (score) = 0.82690

True (score) = 0.17310

1. For a true image:

False (score) = 0.68950

True (score) = 0.31050

The script used for classifying a test image has been attached with the mail with explanation. The code was run in Docker with the following command:

***python /tf\_files/label\_image.py /tf\_files/train\_img\_dir/false/FALSE\_\_201.jpg***

***python /tf\_files/label\_image.py /tf\_files/train\_img\_dir/true/TRUE\_\_1020.jpg***

Help was taken for this project from Google Codelabs.