Varun Venkitachalam

CS4990

**CS4990 Project 2 Second Try Updates**

**UPDATES:**

Put training, fine-tuning, and printing into one file. Used transfer learning with 15 layers of VGG-16 and added 4 classification layers to properly flatten, add dropout, and dense data. Increased dropout in range 0.6 – 0.8 to prevent overfitting. Added data augmentation for training set with sheer rate / augmentation value around 0.65 – 0.75. Fine tuned the model by freezing all but the last 5 layers of the model. Broke the data into 3 equal datasets of male/female and put each data set into one final male/female folder. Biggest issue was using predict generator and loading weights with by\_names = True, which caused problems predicting the new images. Updated version loads each image in the directory and uses model.predict to accurately classify the image. Stores final result into csv file. **Currently 4th** on Leaderboard with **Log Loss** of **0.15932.**

Overview:

The data for this project was multiple jpg images consisting of either males or females. The purpose of the project was to use classification to develop a deep learning model to properly solve the problem. The data initially read through the csv file, allocating the targets for each train\_jpg image, and was split into two categories – Male and Female – for both the training\_set and validation\_set. The model was then trained, validated, fine-tuned, and tested with a different dataset and the results were stored into a csv file using pandas.

Training/ Validation:

The methods for training our weights model incorporated VGG16 which is a pre-trained model that was available. The directories containing the training and validation datasets were read in and converted into a numpy file. This numpy file was adjusted to match the proper shape for the data and was saved to represent the images in the directory. The weights from the VGG16 model were also loaded and incorporated with the numpy file. Our created weights model is then trained using train/target data alongside the VGG16 model.

Fine Tuning:

The pre-trained VGG16 model was used as a convolutional base for our own classifier. We then add the weights model created in the training section on the top, and freeze the model until the last block. VGG16 incorporated 15 layers, so those 15 layers were set to false under trainable. The learning rate as well as the number of epochs were adjusted in order to yield the best results for our weights model. One learning rate that was generally successful was “0.0001” as the number is relatively small.

Testing:

After the model was fine-tuned, it was tested based off of the testing images in the directory. These images were extracted in a similar manner to the training and validation sets, and the weight model we created is loaded in. The results done by the model.predict function were stored into a numpy array, to a pandas data frame, and then finally a csv file. The results generated a low log loss as well.

Files:

UltimateGenderSplit.py splits the directories into two. CrazyAlteredGenderCombo.py trains our weights model, fine tunes the classification layers of our model, and predicts the test images. Results are stored in a csv/ best csv is CrazyPredictions32FineTuned.csv