Centre for Innovation (CFI) – IIT Madras

Team ARIHANT

Software Module Member – Mini-Project Technical Review

Name: Vishnu Vinod	**Ph No: 8157022999**
Roll No: CS19B048	**Mail:** vishnuvinod2001@gmail.com

Question 1

The question asks us to process the given image to identify the edges clearly from the given picture. The following processing has been applied on the image to obtain the best quality results.

- Image is read in in **grayscale** format to allow efficient gradient calculation
- A **filter** is applied on the image. The kernel filter chosen for this purpose is a **150 x 1 blurring filter**. This filter averages the values along a vertical line of 150 pixel length. This results in the effect of the shiny reflection from the image being nullified as much as possible. The output image is more or less uniformly coloured in the vertical direction. Horizontal direction intensity distribution remains unaffected.
- **Adaptive Thresholding** is now applied on the image to remove unnecessary colour variations and obtain the edges more or less sharply.
- Finally a simple **Sobel filter** is applied to give the gradient (x-gradient) of the image.
- From this image it is very easy to count the number of sheets in the given image

The values and specifics of the processing program such as kernel sizes, scale factors, thresholding parameters, type of adaptive thresholding to be used have all been optimized by running a very large number of trials with various combinations of values/ methods.

Generally this sort of brute force method is unnecessarily time consuming. However over time one develops a sort of intuition for these, which can be really helpful in quick optimization of the processing module.

__Question 2__

This question involves a classic image classification problem.

1. **Create_Train_Test.py :**
   - The file ‘Create_Train_Test.py’ contains the code to split the set of all images into a train set and a test set. This is done by using the **random.sample()** function from the python random library.
   - The **shutil.move()** function in the python shutil library is used to move the selected images from the source folder to the target folder.

2. **Image_Classifier.ipynb :**
   - The image classifier starts off by making a large number of potentially useful imports, some of which haven’t been used in the program itself. (Most of such imports were used in previous versions of the program while testing etc.)
   - An important thing to note here is that the image data format is set to “channels_last” mode.
   - Google drive I then mounted on the disk to allow faster and more efficient data reading and processing. **google.colab.drive.mount()** function is used here.
   - We now begin defining the Deep Learning Model. I have decided here to build my own model from scratch. Though the process of tuning hyperparameters to get a decent test accuracy of 50% was painstakingly lengthy it has helped me gain an insight into how various hyperparameters influence performance of a network.
   - **keras.model.Sequential()** is used to group a number of sequentially defined layers into a **keras.model.Model** object.
   - The **add()** method of the keras.model.Model object is used to add new layers onto the model.
   - The layers are defined using the common **keras.layers.Dense()**, **keras.layers.Conv2D()**, and the **keras.layers.MaxPooling2D()** functions.
   - **keras.layers.Dropout()** is used to introduce dropout regularization into the model. This reduces overfitting and increases the validation set (here test set) accuracy by a considerable degree.
   - The overall structure of the Deep Learning model I have implemented is based heavily on the VGG16 architecture. However the model is simpler and much smaller than VGG16 itself.
   - **Compile()** method of the keras.model.Model is used to compile the model with required optimizers (here Adam is chosen)
   - From **keras.preprocessing.image** we import the **ImageDataGenerator** class. This is used to create a data generator that directly generates labelled data generator from a directory if needed. It also allows for automatic data augmentation, though I have not used this feature here.

- **ImageDataGenerator.flow_from_directory()** is used to create a datagenerator from the directory path alone. (All .jpg, .png, .jpeg, .tiff, .bmp files will be read in in appropriate form). Pixel data is also rescaled by a factor of 1/255 to fit all pixel values within the range of 0 and 1.
- **Model.fit()** is used to now train the assembled model on the generated training data. Validation data used here is that of the test set itself and is used used only to show variation of test set accuracy and loss per epoch in an efficient manner.
- **Keras.utils.plot_model()** is used to plot the overall structure of the Deep Lerning Model used.
- Standard **matplotlib.pyplot** functions are used to plot accuracy and loss per epoch for both training set and test set data.
- **Model.predict()** is used to carry out prediction for all the test set data now using the trained model. The list returned is just a list of probabilities. This is converted into a list of predictions by using the **np.argmax()** function.
- **sklearn.metrics.confusion_matrix()** is used to calculate the confusion matrix based on the actualled labelled data and the predicted data. This confusion matrix is then plotted (magnified plot used)

Though it was possible to use pre-trained weights and models for this project, since it was my first unsupervised project (outside of the DL courses I have been taking) I wanted to build a neural network from scratch. The training accuracy after 20 epochs peaks at 88% at the end. The validation set accuracy peaks at 53% but reduces to 48% towards the end of training. The validation set accuracy obtained though acceptable as a first try is not a great model. Further improvement can be done primarily by increasing the volume of training data and also slightly by carrying out data augmentation.

Overall it was a fun learning experience and I look forward to working with Team ARIHANT in the future as member of the CVI & DL team.