Machine Learning II

Final Project

Individual Report – Geetha Satya Mounika Ganji (G40652882)

Name of the Dataset: RVL-CDIP data

Link to download the data: http://www.cs.cmu.edu/~aharley/rvl-cdip/

Introduction:

The RVL-CDIP dataset consists of 400,000 images in 16 classes, with 25,000 images per class. There are 320,000 training images, 40,000 validation images, and 40,000 test images. The images are sized so their largest dimension does not exceed 1000 pixels.

The label files list the images and their categories in the following format:

path/to/the/image.tif category

where the categories are numbered 0 to 15, in the following order:

1. letter
2. form
3. email
4. handwritten
5. advertisement
6. scientific report
7. scientific publication
8. specification
9. file folder
10. news article
11. budget
12. invoice
13. presentation
14. questionnaire
15. resume
16. memo

Data Preprocessing:

1. **Converting .txt files to .csv files:**

The data provided:

1. train.txt

2. test.txt
3. val.txt which are converted into .csv files to facilitate the analysis.

When converted into .csv file the data looked as below:

	0
0	imagesq/q/o/c/qoc54c00/80035521.tif 15
1	imagese/e/w/c/ewc23d00/513280028.tif 1
2	imagesw/w/b/t/wbt26e00/2053453161.tif 7
3	imagesm/m/k/m/mkm05e00/2040792992_2040792994.t...
4	imageso/o/e/x/oex80d00/522787731+-7732.tif 3

Figure 1. Data after converting from .txt to .csv format

The highlighted parts in the figure 1 are the classes of the images which looks like below after splitting.

	image	class
0	imagesq/q/o/c/qoc54c00/80035521.tif	15
1	imagese/e/w/c/ewc23d00/513280028.tif	1
2	imagesw/w/b/t/wbt26e00/2053453161.tif	7
3	imagesm/m/k/m/mkm05e00/2040792992_2040792994.tif	10
4	imageso/o/e/x/oex80d00/522787731+-7732.tif	3

Figure 2. Image and its class

Basing on the class, the images were pushed into respective file locations using shutil.copy() which is done for all three .txt files namely training set, test set, validation set.

2. **Checking shape of image:**

Using opencv the shape of image is checked.

Shape of image is accessed by img.shape which returned a tuple of number of rows, columns and channels. As the tuple contained number of rows, columns and channels, came to a conclusion that it is not a grayscale image (if image is grayscale the tuple would contain only number of rows and columns).

```
img = cv2.imread('C:/Users/geeth/Documents/Google_cloud/Project/Regional_whole/valid/email/0000945975.tif')
print(img.shape)

```

```
(1000, 762, 3)
```

3. Image selection for model building:

After discussion, each image is divided into 5 parts:

1. right_body: right part of the image
2. left_body: left part of the image
3. whole: total image without partitions
4. header: header of the image
5. footer: lower part of the image

As the dataset considered is too large to build a model, image selection is done as below:

1. right_body:
    a. train: 144 images per class
    b. test: 60 images per class
    c. validation: 60 images per class

4. Preprocessing for Keras:

1. Data is loaded as files and labels separately

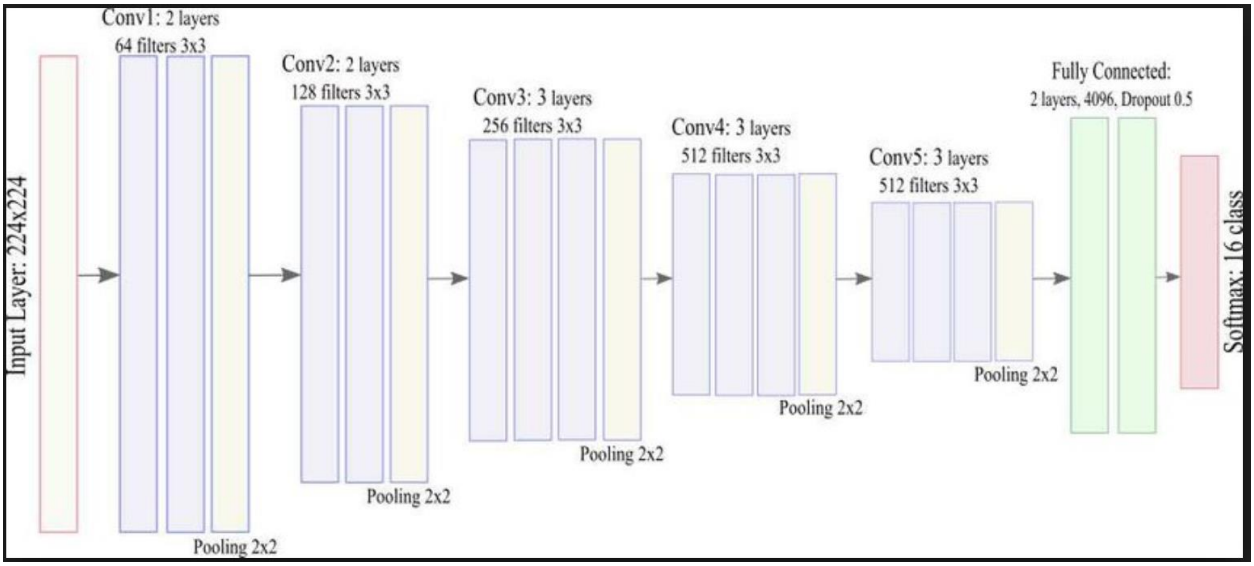
```
train_files, train_targets = load_dataset('/home/ubuntu/Deep-Learning/Project/Regional/train/right_body')
valid_files, valid_targets = load_dataset('/home/ubuntu/Deep-Learning/Project/Regional/valid/right_body')
test_files, test_targets = load_dataset('/home/ubuntu/Deep-Learning/Project/Regional/test/right_body')
```

2. When using Keras CNN, 4D array is required as input with shape.
3. The convert_4darray function takes file path to image as input returns a required 4D array suitable for CNN. The function first loads image and resizes it to an image that 224*224.
4. The convert_4darrays function takes numpy array of image paths as input and returns a 4D array with shape.
5. Images are rescaled by dividing every pixel in every image by 255.

Model Building:

1. **VGG16:**
   VGG-16 is a convolutional neural network trained on more than a million images from Imagenet database. The network is 16 layers deep and can classify images into 1000 categories. The default input shape for VGG-16 is 224*224.

Architecture of VGG-16:



VGG-16 is imported from keras.applications.vgg16. Keras Applications are deep learning models that are based on pre-trained weights.

Format:

VGG16(include_top = True, weights = 'None', input_tensor = None, input_shape = None, pooling = None, classes=1000)

Arguments:

1. **include_top:** decided whether to include 3-fully connected layers at the top of the network
2. **weights:**
    a. **None:** the weights are randomly initialized.
    b. **imagenet:** uses weights form pre-trained Imagenet.
3. **input _tensor:** optional. This is output of layers.Input()
4. **input_shape:** optional. Only specified if include_top is specified as False. Else the input_shape has to be 224*224 which channels 3.
5. **pooling:** optional. Used for feature extraction when include_top is specified as False.
    a. **None:** output of model will be a 4Darray output of the last convolution layer
    b. **avg:** global average pooling will be applied to the output of last convolution layer.
    c. **max:** global max pooling will be applied to the output of last convolution layer.
6. **classes:** optional. Only to be specified if include_top is False, and weights is specified as 'None'. Specifies number of classes to classify images into.

In constructing VGG-16 n RVL-CDIP dataset, weights are specified as ‘imagenet’ and input_shape as (224,224,3).

VGG16 was built on both header images and whole images.

Summary of model:

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808

```
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None, 16)                401424
=================================================================
Total params: 15,116,112
Trainable params: 15,116,112
Non-trainable params: 0
```

Using pretrained VGG16 from keras.applications.vgg16 and pretrained imagenet weights, another model was built with below configuration:

Number of epochs: 10

Optimizer: Adam

Loss: Mean Squared error

Metrics: Accuracy

Batch_size: 64

Verbose: 2

Loss: categorical_crossentropy

Result: Test accuracy of 67%

Code used from google:

((20-10)/(20+52)) * 100 =13.88 %

References:

1. https://keras.io/layers/pooling/
2. https://keras.io/applications/#vgg16
3. https://machinelearningmastery.com/use-pre-trained-vgg-model-classify-objects-photographs/
4. https://keras.io/optimizers/
5. http://agnesmustar.com/2017/04/19/build-vgg16-scratch-keras-part/