Machine Learning- II

RVL-CDIP Dataset (Individual Report)

Renu Gopal Reddy Durgampudi

G44607690

Group-5

Instructor: Prof. Amir Jafari

Machine Learning II_DATS_6203_11

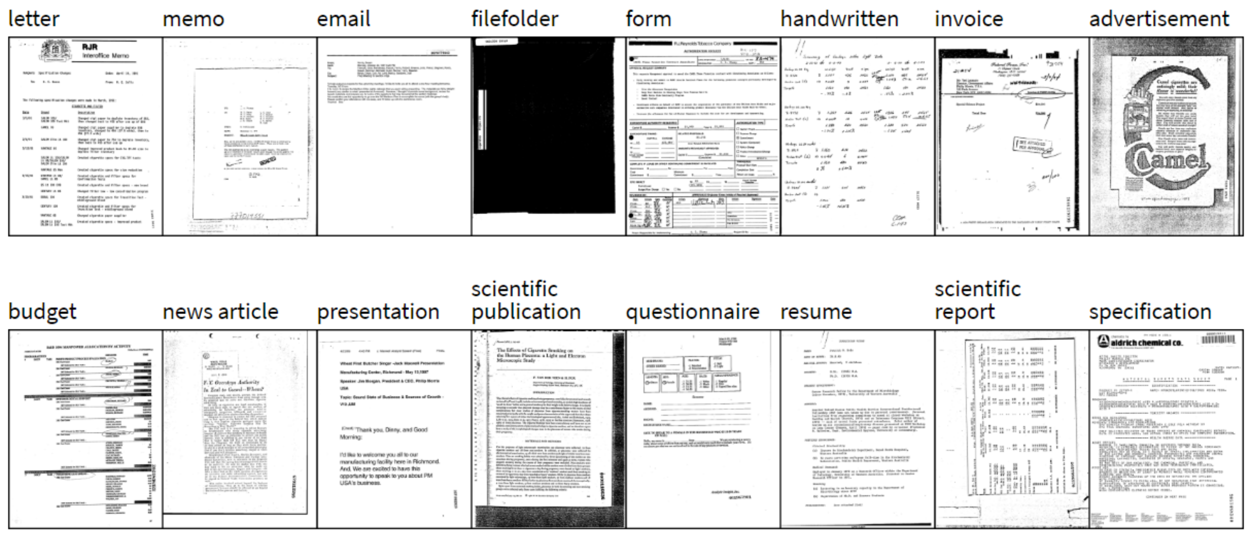
Introduction:

The project is about the document image classification using features learned by convolutional neural networks (CNNs). For this project we considered the RVL-CDIP (Ryerson Vision Lab Complex Document Information Processing) dataset.

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

Data Description:

Documents are classified into various classes based on their text contents and their structural properties. Initially it was observed that a real-life document can be viewed in different ways, in both geometric and logical structure spaces. 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.



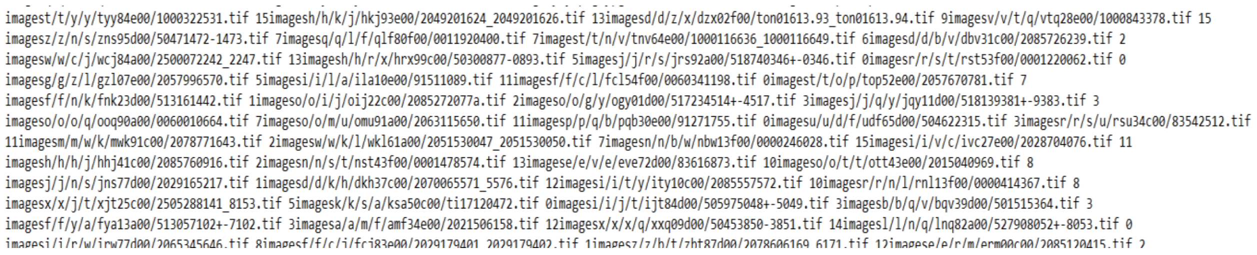
The categories are numbered from 0 to 15, in the following order:

0. letter
1. form
2. email
3. handwritten
4. advertisement
5. scientific report
6. scientific publication
7. specification
8. file folder
9. news article
10. budget

11. invoice
12. presentation
13. questionnaire
14. resume
15. memo

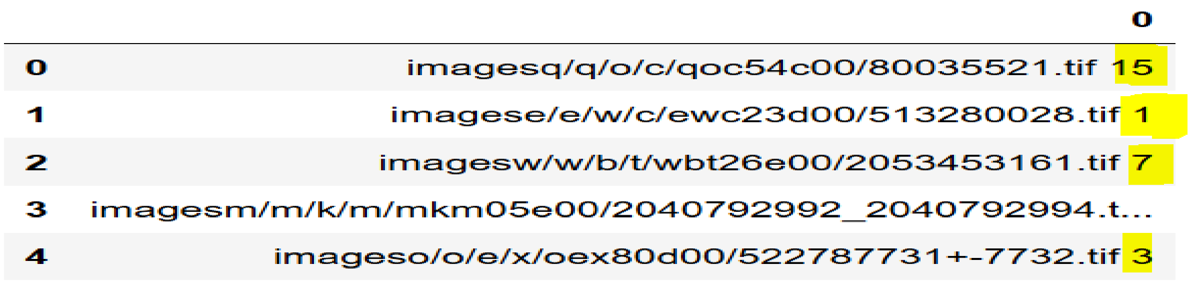
Data Preprocessing:

The downloaded files had two folders images and labels. Images folder contains 25 different folders. Labels had three text documents named train, test and valid. Information in these folders are present in the following way.



```
imagest/t/y/y/tyy84e00/1000322531.tif 15imagesh/h/k/j/hkj93e00/2049201624_2049201626.tif 13imagesd/d/z/x/dzx02f00/ton01613.93_ton01613.94.tif 9imagesv/v/t/q/vtq28e00/1000843378.tif 15
imagesz/z/n/s/zns95d00/50471472-1473.tif 7imagesq/q/l/f/qlf80f00/0011920400.tif 7imagest/t/n/v/tnv64e00/1000116636_1000116649.tif 6imagesd/d/b/v/dbv31c00/2085726239.tif 2
imagesw/w/c/j/wcj84a00/2500072242_2247.tif 13imagesh/h/r/x/hrx99c00/50300877-0893.tif 5imagesj/j/r/s/jrs92a00/518740346+-0346.tif 0imagesr/r/s/t/rst53f00/0001220062.tif 0
imagesg/g/z/l/gzl07e00/2057996570.tif 5imagesi/i/l/a/ila10e00/91511089.tif 11imagesf/f/c/l/fcl54f00/0060341198.tif 0imagest/t/o/p/top52e00/2057670781.tif 7
imagesf/f/n/k/fnk23d00/513161442.tif 1imageso/o/i/j/oij22c00/2085272077a.tif 2imageso/o/g/y/ogy01d00/517234514+-4517.tif 3imagesj/j/q/y/jqy11d00/518139381+-9383.tif 3
imageso/o/o/q/ooq90a00/0060010664.tif 7imageso/o/m/u/omu91a00/2063115650.tif 11imagesp/p/q/b/pqb30e00/91271755.tif 0imagesu/u/d/f/udf65d00/504622315.tif 3imagesr/r/s/u/rsu34c00/83542512.tif
11imagesm/m/w/k/mwk91c00/2078771643.tif 2imagesw/w/k/l/wkl61a00/2051530047_2051530050.tif 7imagesn/n/b/w/nbw13f00/0000246028.tif 15imagesi/i/v/c/ivc27e00/2028704076.tif 11
imagesh/h/h/j/hhj41c00/2085760916.tif 2imagesn/n/s/t/nst43f00/0001478574.tif 13imagese/e/v/e/eve72d00/83616873.tif 10imageso/o/t/t/ott43e00/2015040969.tif 8
imagesj/j/n/s/jns77d00/2029165217.tif 1imagesd/d/k/h/dkh37c00/2070065571_5576.tif 12imagesi/i/t/y/ity10c00/2085557572.tif 10imagesr/r/n/l/rnl13f00/0000414367.tif 8
imagesx/x/j/t/xjt25c00/2505288141_8153.tif 5imagesk/k/s/a/ksa50c00/ti17120472.tif 0imagesi/i/j/t/ijt84d00/505975048+-5049.tif 3imagesb/b/q/v/bqv39d00/501515364.tif 3
imagesf/f/y/a/fya13a00/513057102+-7102.tif 3imagesa/a/m/f/amf34e00/2021506158.tif 12imagesx/x/x/q/xxq09d00/50453850-3851.tif 14imagesl/l/n/q/lnq82a00/527908052+-8053.tif 0
imagesi/i/r/w/irw77d00/2065345646.tif 8imagesf/f/c/j/fcj83e00/2029179401_2029179402.tif 1imagesz/z/h/t/zht87d00/2078606169_6171.tif 12imagese/e/r/m/erm00c00/2085120415.tif 2
```

Every scanned image is saved with '.tif' extension. For understanding the information present in the text document, I Converted the .txt files to .csv files using the python code.



	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

My understanding from this is there is a number after every '.tif'. From my analysis the number is the category of the document. For example, first image has 15 at the end which means the document might be the memo. The highlighted parts in the above figure are the categories of the images which looks like below after splitting the information into two different fields image and class.

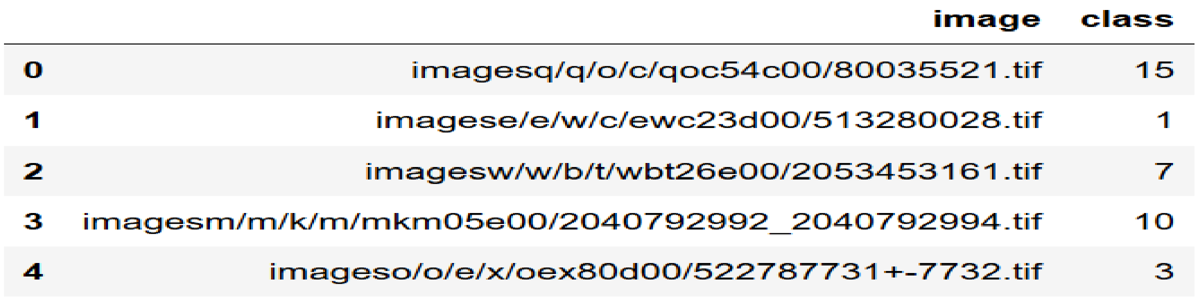


	image	class
0	imagesq/q/o/c/qoc54c00/80035521.tif	15
1	imagese/e/w/c/ewc23d00/513280028.tif	1
2	imagesw/w/b/t/wbt26e00/2053453161.tif	7
3	imagesm/m/k/m/mkm05e00/2040792992_2040792994.tif	10
4	imageso/o/e/x/oex80d00/522787731+-7732.tif	3

Basing on the class number, I used shutil.copy() to sort the images into training set, test set and validation set. As the dataset is so big, manually modified the data into 140 train images per class, 60 test and validation images per class. Then checked the size of the images, most of the images are in 1000*762 pixels, few are above 762 pixels width and below 762 pixels width.

Research about CNN:

Before working on CNN, I understood different concepts in the convolution neural networks like edge detection, padding, stride, maxpooling, Flatten, Dense and softmax layers.

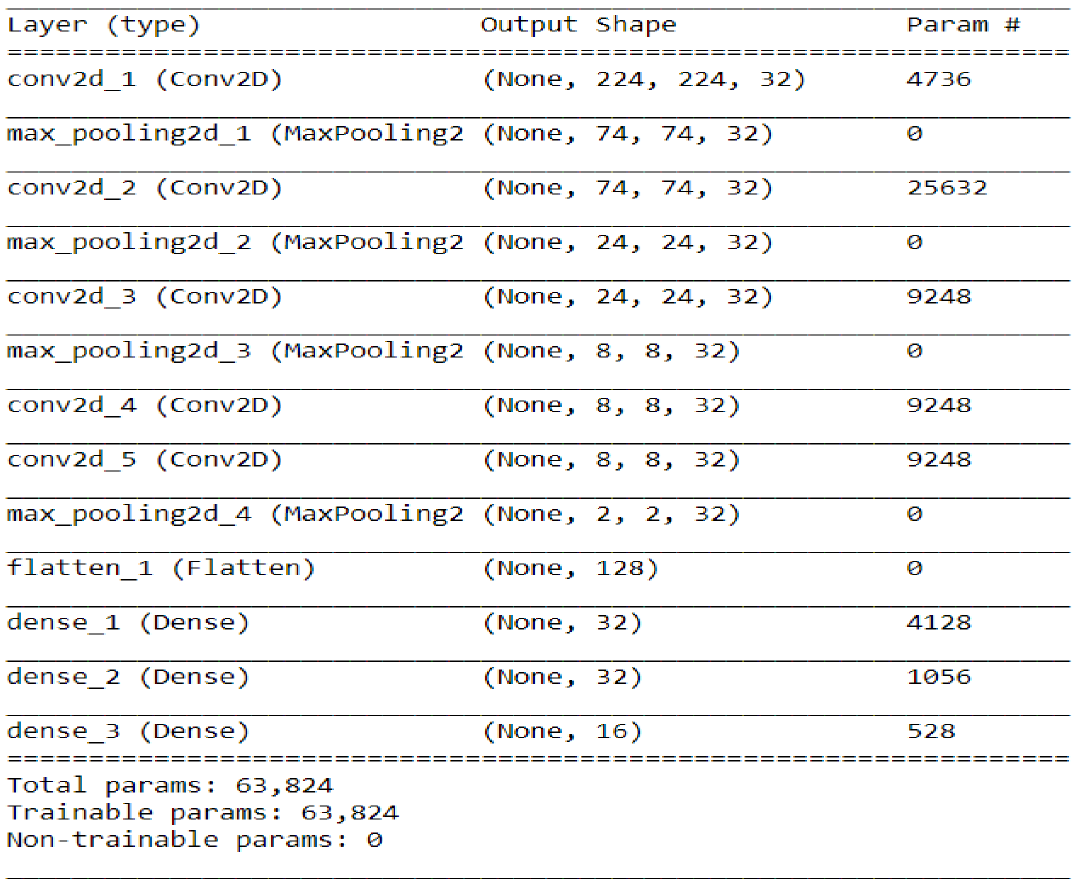
Rescaled the images to 224*224 as the model is taking long time to show the results when trained with the original size. Tried to rescale the images to ½, ¼, 1/8, 1/16th parts. As we are reducing the size of the images, the information in the image is becoming a line or dot.

Model Building:

Built a basic CNN in Keras with 5 convolution layers, 4 Max pooling layers, 2 flatten layers and one prediction layer (softmax).

In the first layer of the convolution the kernel size is 7*7 and used 32 such kernels

The summary of the model is



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 224, 224, 32)      4736
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 74, 74, 32)        25632
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 24, 24, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 24, 24, 32)        9248
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 32)          9248
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 32)          9248
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 2, 2, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                4128
_________________________________________________________________
dense_2 (Dense)              (None, 32)                1056
_________________________________________________________________
dense_3 (Dense)              (None, 16)                528
=================================================================
Total params: 63,824
Trainable params: 63,824
Non-trainable params: 0
_________________________________________________________________
```

Number of epochs used 10

Optimizer used is Adam

Loss: Mean Squared error

Metrics: Accuracy

Batch_size is 64

Verbose: 2

Results: Test accuracy of 8.125% which is very low.

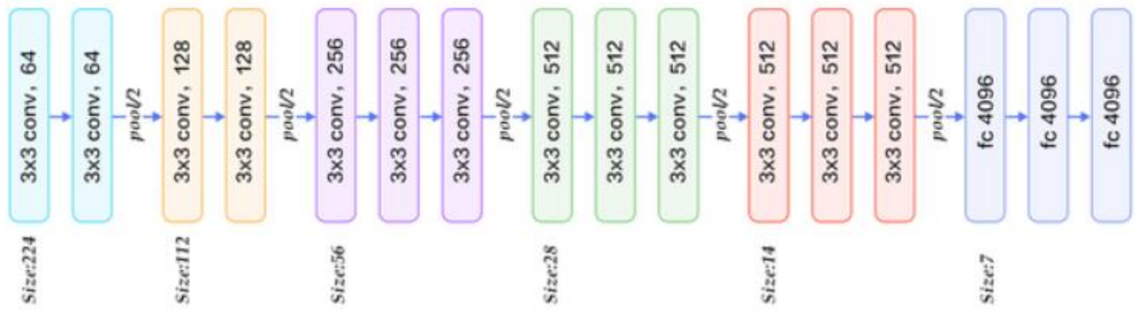
Researched on the concepts like VGG-16, ImageNet, Resnet, etc..

VGG-16 layer:

VGG-16 layer is also called as Oxfordnet. Also called as Visual Geometry Group of Oxford named after who developed it. It is a simplified version of AlexNet and yields better results than AlexNet.

This is built using convolution layers of 3*3 kernel size, stride = 1 and padding as 'same'. All max pooling performed will be of size 2*2, stride = 2. Fully connected layers at end and total of 16 layers. The transfer function used in convolution layers and fully converted layers is ReLU.

Architecture:

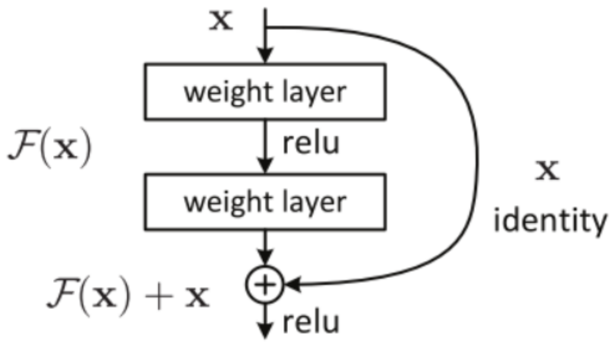


In the source code there is a code called applications from where VGG-16 is imported. This is pre-implemented. VGG-16 is trained on imagenet. They have taken imagenet dataset onto VGG network and the model is trained. So we actually get a model with all weights filled in which means we can reuse this model which is readily available in Keras. So we get VGG-16 trained and built in Keras. No need to choose the kernel sizes and strides.

ResNet:

This is called as Residual Network. ResNets were published in late 2015.In plain networks like VGG-16 with the increase in number of layers both training error and test error even after many iterations were worse even after using dropout and relu from where the residual network came into picture. The core idea of ResNet is identity shortcut connection that skips one or more layers which is called skip connection which can be seen in the below image of architecture of ResNet. Similar idea can be seen in Long Short Term Memory Netowrk (LSTM).

Architecture of ResNet:



a residual block

Imagenet:

One of major reasons for the popularity for deep learning is:

1. lots of data
2. lots to compute
3. lots of algorithms

it is a database which contributed most to the deep learning. ImageNet contains 14 million images. In each image there are objects belonging to 20000 classes. Some of these classes can be ambiguous. For example, there is an image of dog, so dog is a class, and the breed of a dog is also a class. So, the image belongs to both the class dog and class dog breed which is meant as ambiguity.

Code from google: 15%

References:

https://arxiv.org/pdf/1801.09321.pdf

https://arxiv.org/pdf/1409.1556.pdf

https://arxiv.org/pdf/1512.03385.pdf