Machine Learning Project Report

on

Kaggle's

Cdiscount's Image Classification Challenge

Meetika Sharma: mxs173530

Priyank Shah: pks170030

Yash Pradhan: ypp170130

Contents

Chapter 1

Introduction and Problem Description

Image classification is a necessary task these days as most of the data that is available today is in non-textual format. Image classification refers to assigning images to different classes based on certain features that are extracted from images. Complex models of deep learning such as a deep neural network, convolutional neural network are known to perform good for this task. We have used convolutional neural network.

This challenge is from one of the ongoing competitions on kaggle.com. CDiscount is a french e-commerce company which has a lot of images of all the products that it sells and it wants to classify the products in different categories based the image of the product. The current process that is employed by Cdiscount is applying machine learning techniques to the text description of a product to determine its category.

This challenge asks for taking the potential of their method to a whole new level by applying machine learning techniques to images. However, this task is multiclass classification as each image can belong to more than one classes which makes it even more complex.

Chapter 2

Related work

We have referred the following machine learning tasks in order to get an insight about which machine learning techniques to use for classification of images.

- MNIST: This dataset contains images of handwritten digits from 0 – 9. The images are in gray scale. The resolution of images being small, a deep neural net gave good results for the same.
- CIFAR-10: This dataset contains various RGB images belonging to 10 classes. We referred Navin Manaswi's article on how to build an image classifier using TF Learn
- The classification task at hand requires much complex network and hence we referred to Andrew Ng's course on Convolutional Neural Network on Coursera.
- Apart from this, for data preprocessing and analysis we have referred kernels on Kaggle.com in order to get an idea about the same.

Chapter 3

Dataset Description

Dataset Description:

- 9 million products which is almost half of their current catalogue
- Over 15 million images at resolution of 180 x 180
- More than 5000 categories of the products

Description of files:

train.bson : This file contains a list of 7,069,896 dictionaries, one dictionary per product. Each dictionary has a product id (key: _id), the category id of the product (key: category_id) and 1 – 4 images of the product. The images are stored in a list (key: imgs). Each list of image(s) contains a single dictionary per image, format: {'picture': b'...binary string...'}. The binary string is the binary representation of the Image in JPEG format.

train.bson.torrent: Torrent file for downloading the train.bson file. The size of the train.bson file is 58.2 GB.

test.bson: This file contains a list of 1,768,182 products in the same format as the train.bson file. Only difference is that, the category_id is not included in this file. Our objective is the determine the same.

test.bson.torrent: Torrent file for downloading the test.bson file. The size of the test.bson file is 14.5 GB.

train_example.bson: This contains the first 100 records of the train.bson file.

category_names.7z: This shows the hierarchy of classification. The labels are in French. Category_id corresponds to the label in lowest level(level 3) category.

sample_submission.7z: Shows the required format for submission.

Data Distribution

The dictionaries of product have images, and the number of images per product are 1 – 4. The visualization of this distribution is shown below.

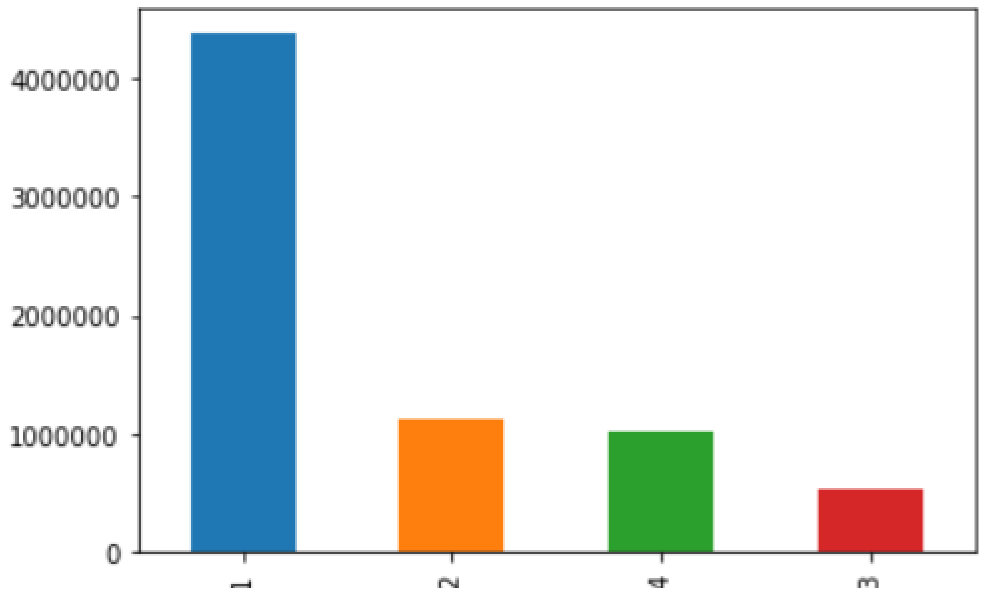


Figure 1: Images per product

For convenience, we have observed the distribution of classes for the training example which is a subset of the train dataset.

Distribution of products over classes for train_sample

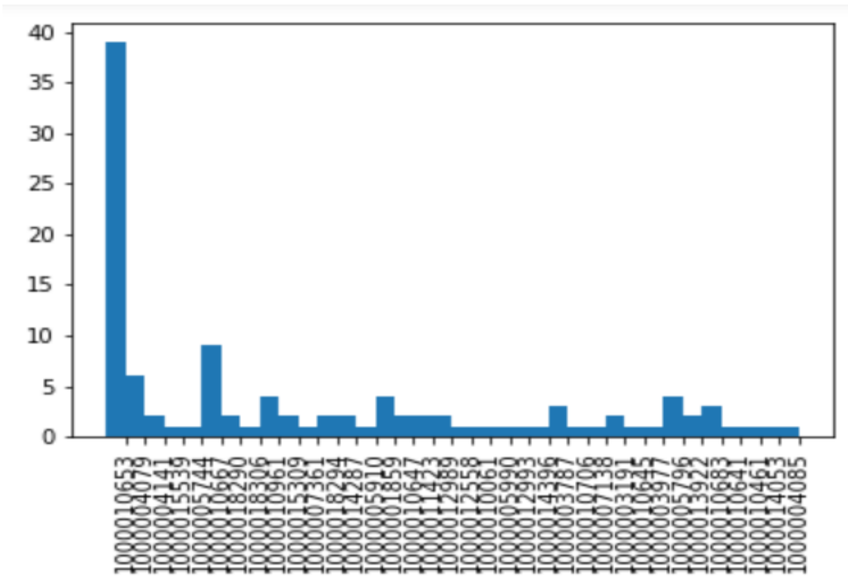


Figure 2: Number of images per category id

The values on the X axis is the category into which the images belong to. The Y axis shows the count of images in each class.

The distribution of all images in the training set is shown in the following plot.

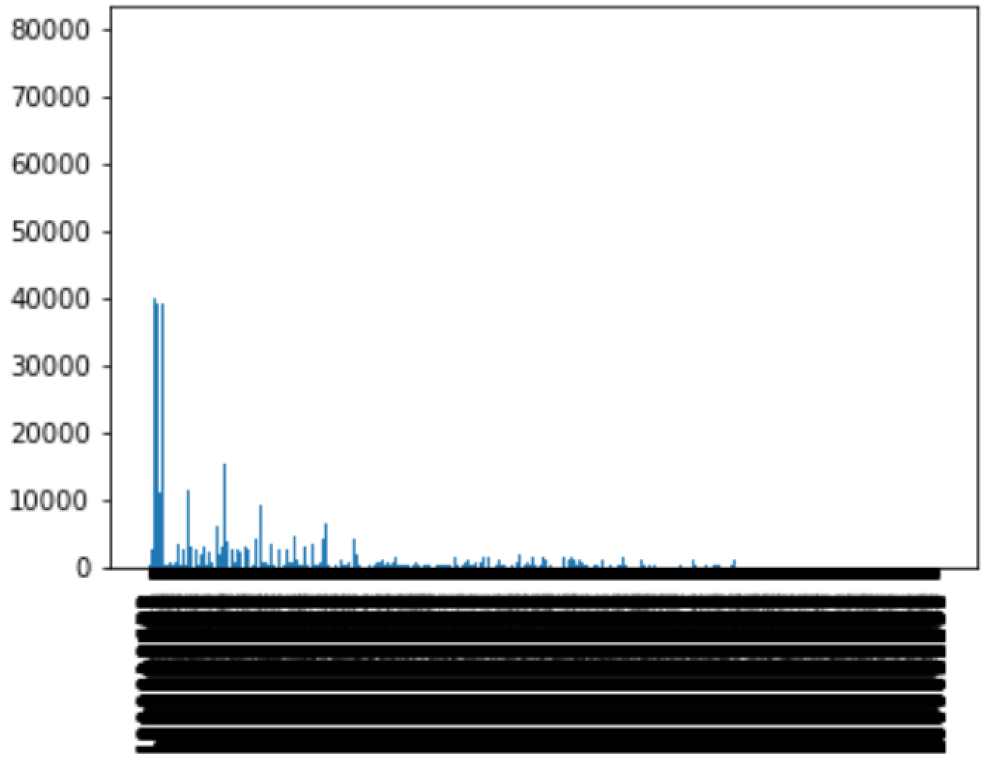


Figure 3: This plot shows that the dataset has class imbalance distribution.

Chapter 4

Preprocessing Techniques

- The image data is given is the bson (binary json) format

train_example.bson


```
431b 0000 105f 6964 0000 0000 0004 696d
6773 001e 1b00 0003 3000 161b 0000 0570
6963 7475 7265 0003 1b00 0000 ffd8 ffe0
0010 4a46 4946 0001 0100 0001 0001 0000
ffdb 0043 0008 0606 0706 0508 0707 0709
0908 0a0c 140d 0c0b 0b0c 1912 130f 141d
1a1f 1e1d 1a1c 1c20 242e 2720 222c 231c
1c28 3729 2c30 3134 3434 1f27 393d 3832
3c2e 3334 32ff db00 4301 0909 090c 0b0c
180d 0d18 3221 1c21 3232 3232 3232 3232
3232 3232 3232 3232 3232 3232 3232 3232
3232 3232 3232 3232 3232 3232 3232 3232
3232 3232 3232 3232 3232 ffc0 0011 0800
b400 b403 0122 0002 1101 0311 01ff c400
1f00 0001 0501 0101 0101 0100 0000 0000
0000 0001 0203 0405 0607 0809 0a0b ffc4
00b5 1000 0201 0303 0204 0305 0504 0400
0001 7d01 0203 0004 1105 1221 3141 0613
5161 0722 7114 3281 91a1 0823 42b1 c115
52d1 f024 3362 7282 090a 1617 1819 1a25
2627 2829 2a34 3536 3738 393a 4344 4546
4748 494a 5354 5556 5758 595a 6364 6566
6768 696a 7374 7576 7778 797a 8384 8586
8788 898a 9293 9495 9697 9899 9aa2 a3a4
a5a6 a7a8 a9aa b2b3 b4b5 b6b7 b8b9 bac2
c3c4 c5c6 c7c8 c9ca d2d3 d4d5 d6d7 d8d9
dae1 e2e3 e4e5 e6e7 e8e9 eaf1 f2f3 f4f5
```


Figure 4: Image data in bson format

- We converted the bson data files into json

```
{
    '_id': 0,
    'imgs': [
                {
                    'picture: b'BINARY STRING FOR IMAGE 1'
                },
                {
                    'picture: b'BINARY STRING FOR IMAGE 2'
                }
            ],
    'category_id': 1000010653
}
```

Figure 5 Image data in json format

- We have one dictionary per product. As seen the keys in dictionary for each product are '_id' – product id, 'category_id' – Category ID of the product, 'imgs' – a dictionary containing all the images of that product.

- The image data is in hex format (e.g \xff), we have converted the images in the encoded format to RGB values, and these RGB values of the pixels of image serves as raw features for the learner.
- RGB values range from 0-255, the size of each image is 180*180*3 where 180*180 is the dimension of the image and 3 is the number of channels.
- We plotted these using the matplotlib to view the images, here are a few of them

Figure 6: Sample training images.

The raw pixels of images act as features

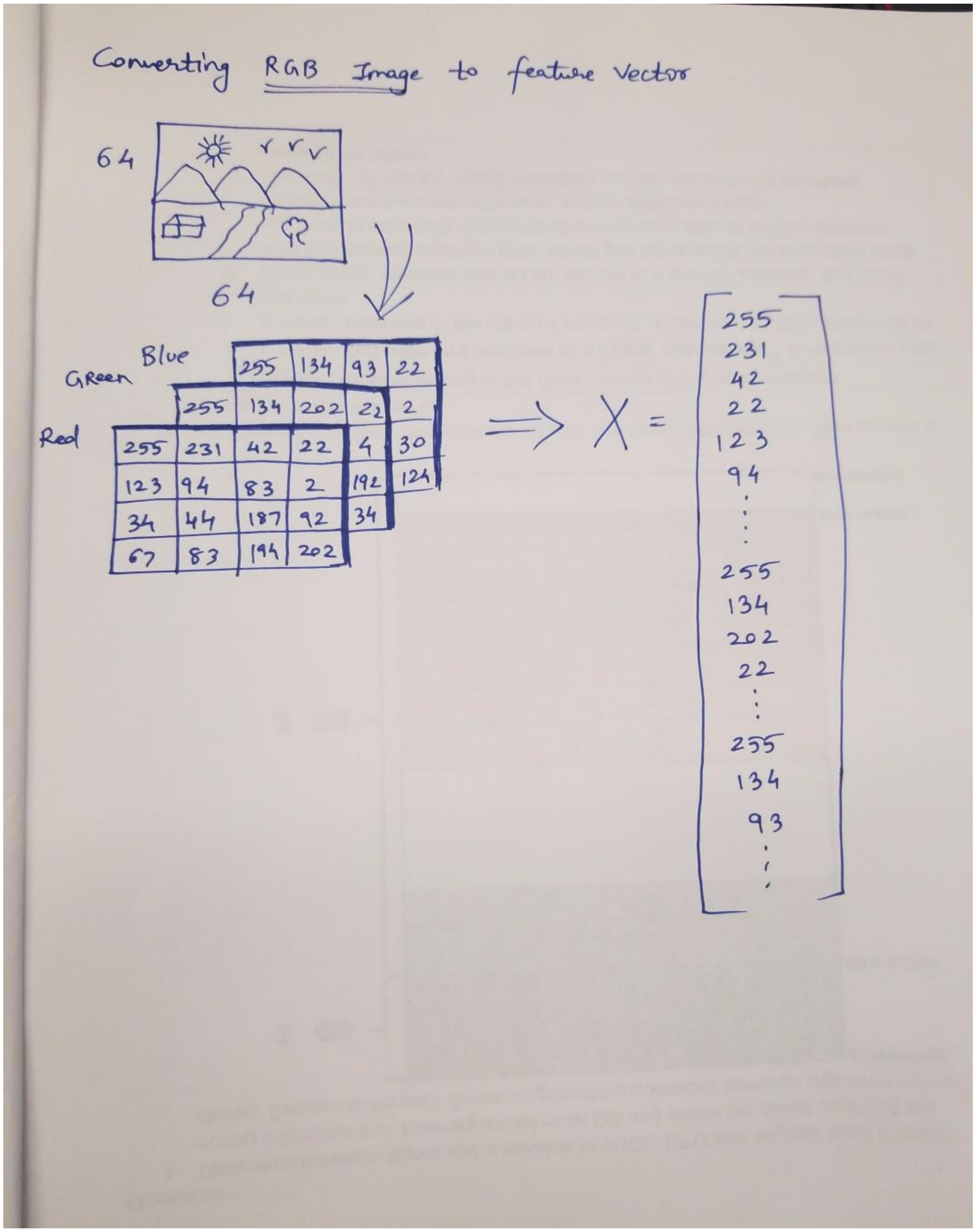


Figure 7: Converting pixel values to single feature vector.

Chapter 5

Our proposed solution, and methods:

We have used convolutional neural network for the purpose of image classification.

Why CNN and not other classifier studied in this course?

We have used the image in their raw format as features for the classifier to learn on. As CNN has the special functionality of using convolution layers, it can learn from the RGB pixels of Image.

Also unlike other datasets, in images it is not necessary that all the images are similar say centered in same place, orientation, lightening etc. The CNN model works efficiently because it learns from the convolutions of image which contains only the important features in an image.

Consider the following classifiers:

- Perceptron
- Decision Tree
- Linear SVM
- Naïve Bayes
- Random Forests

Neural Networks were used for the character recognition, but in that case the images were gray scale.

In case of more complex Image Classification tasks like this, CNN is known to have performed the best, also these classifiers don't perform well with the RGB values of the images as features.

General Architecture of a Convolutional Neural Net

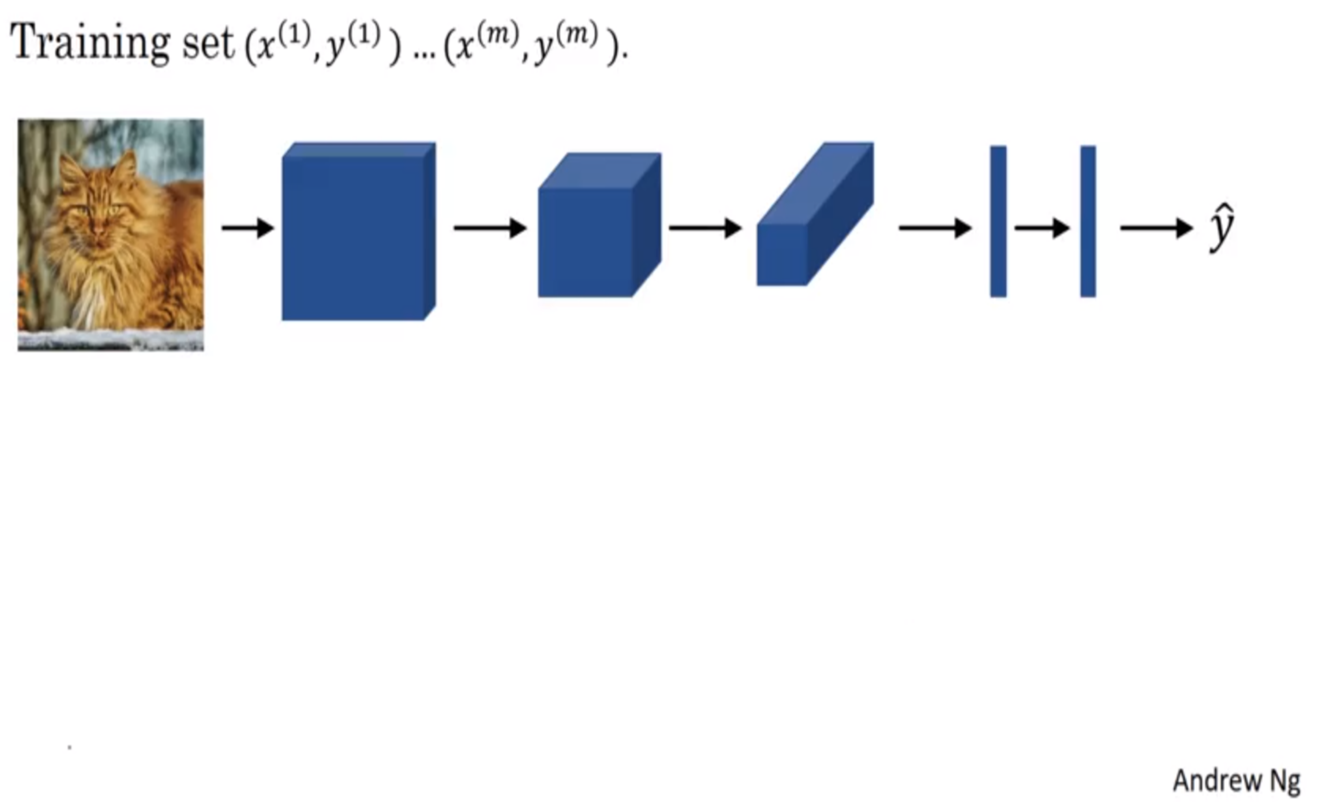


Figure 8: CNN Example

Learning Procedure

1) Firstly, the images from the BSON format are converted to the RGB which is the standard format for images.
2) The image is fed into the network, its dimension is H*W*3, H is the height of image, W is the width of image
3) Next the image undergoes the convolution and pooling layers.

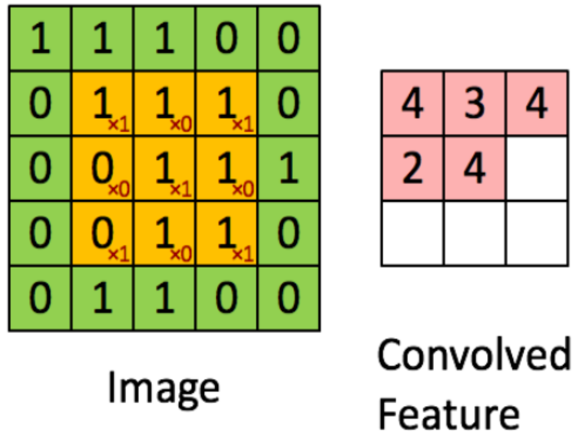


Figure 9: Example of Convolution

4) The convolution for RGB image happens as shown

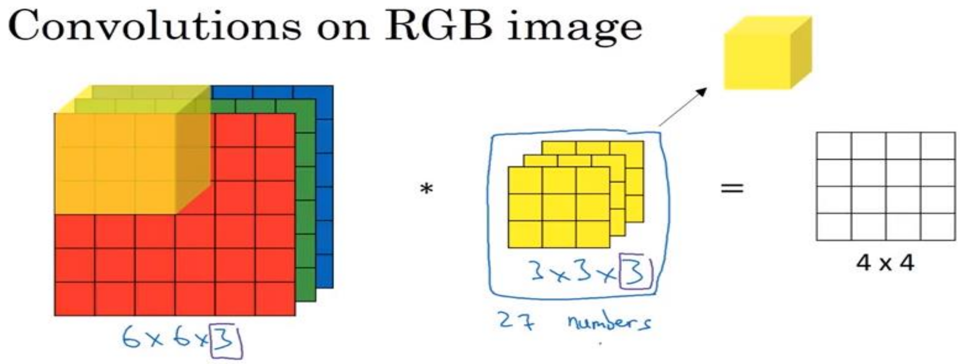


Figure 10: Example of Convolution for RGB

5) There are different kinds of pooling, for examples: Average Pooling, Max Pooling etc.

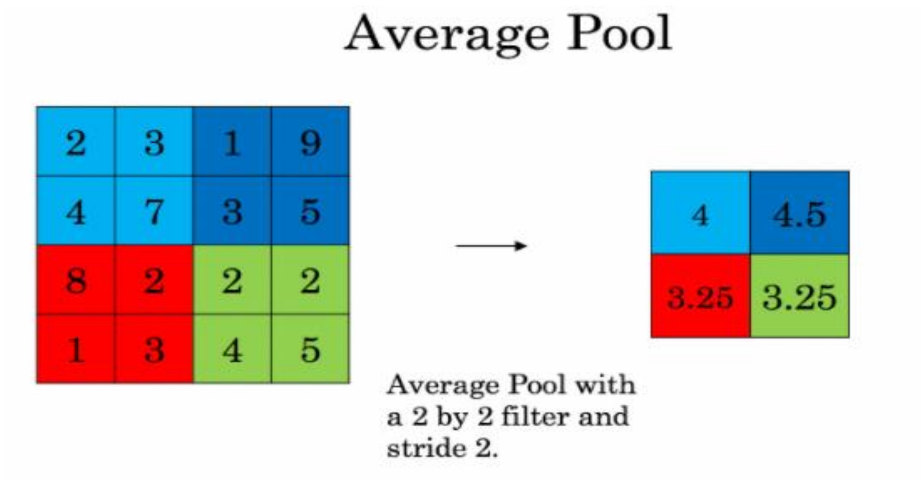


Figure 11: Example of average pooling

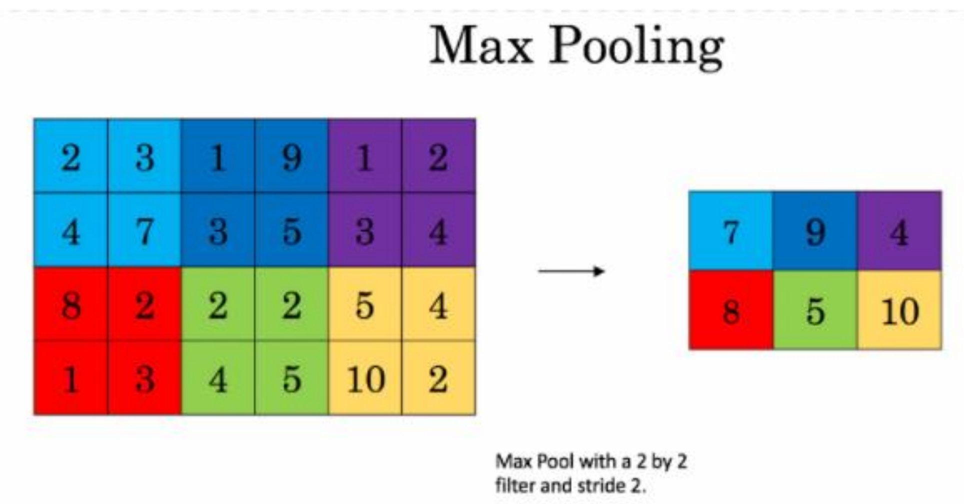


Figure 12: Example of max pooling.

6) The activation function used is ReLU function, filters act as weights of Network.
7) The last layer in the ConvNet is the fully connected layer, and uses the softmax function which gives final output as a probability distribution over classes, the one with the highest confidence is the predicted category for that example.
8) The Loss is measured by cross-entropy i.e. log loss function.

Following is the architecture of our Convolutional Neural Network.

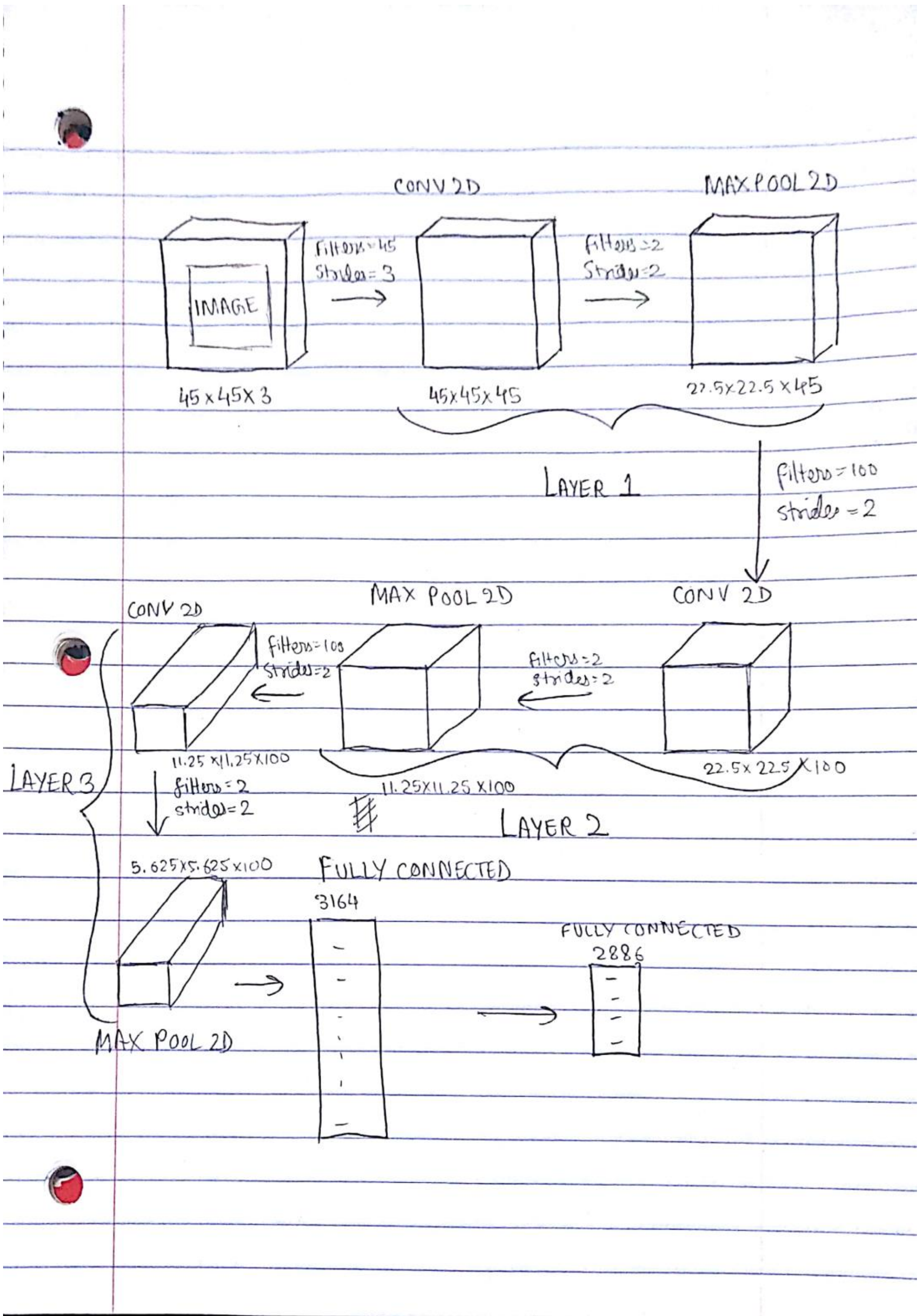


Figure 13: Network architecture of our model

About this ConvNet:

- The Image is rescaled from 180*180 to 45*45
- Layer 1 consists of convolutional layer and max pool layer
- After passing through this network, the network starts learning and essential features are retained, and passed to the next layer
- The second layer is also convolution and max pooling where more learning happens.The formula for the dimensions of image that goes into next layer is as follows:

Layer l is convolution layer

$f^{[l]}$ = filter size

$p^{[l]}$ = padding

$s^{[l]}$ = stride

$n_c^{[l]}$ = # of filters

Dimension of filter

$f^{[l]} \times f^{[l]} \times n_c^{[l]}$

Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_C^{[l-1]}$

Output: $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

$$n_H^{[l]} = \left\lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

Similarly, for $n_W^{[l]}$

Figure 14: Formulas for calculating dimensions

- After layer three, next is the fully connected layer, the activation function used in the last layer is softmax, which converts the predictions to a probability distribution because we have categories of images.

Chapter 6

Experimental Results and Analysis

```
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
```

```
(84138, 3, 45, 45) (84138, 2886) (21035, 3, 45, 45) (21035, 2886)
```

Figure 15: Shapes of training and testing dataset

```
model.evaluate(X_test, y_test)
```

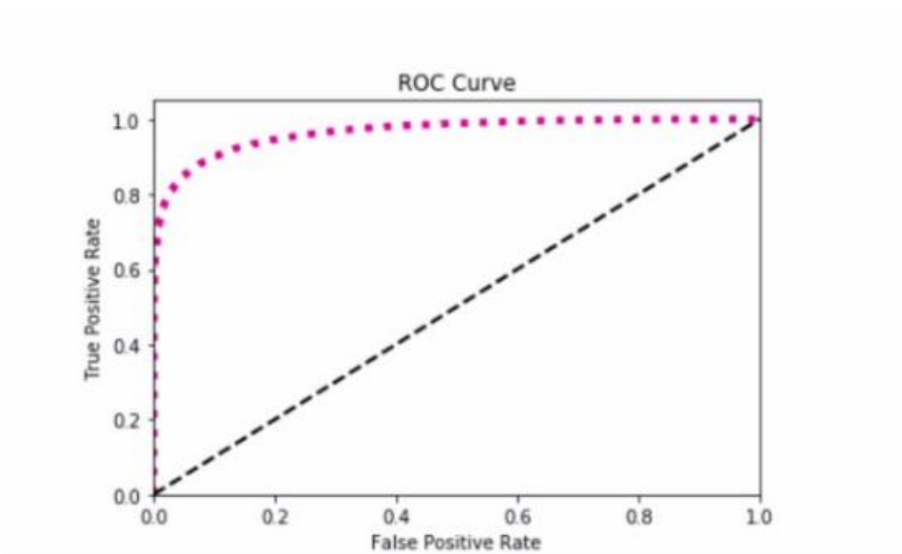
```
[0.36272878533931874]
```

Figure 16: Accuracy of the model

```
from sklearn import metrics
metrics.auc(fpr["micro"],tpr["micro"])
```

```
0.96593397735438435
```

Figure 17: Area under the curve of the model.



figu

Figure 18: ROC Curve

Chapter 7

Conclusion

We have implemented CNN for image classification. The accuracy of our model is around 36% which can be improved by training it on whole dataset. Due to limited resources available, we couldn't train our model on whole dataset, instead we took 50,000 samples with 2886 classes. We have evaluated the model on three metrics i.e. ROC, AUC and log loss. The AUC was approximately 0.96 and log loss was 2.75 up to two decimal points.

We have used raw pixel values as the input to the network. We can get higher accuracy by extracting features such as corners, edges from the images and passing them to the network as this would make the model less complex and it would run faster.

Chapter 8

Contribution of team members

Understanding the Project: Meetika, Priyank, Yash

Defining the scope of the competition: Meetika, Priyank, Yash

Developing project plan and timeline: Priyank

Conducting extensive research: Meetika, Priyank, Yash

Analysis and Data Preprocessing: Meetika, Yash

Writing Project status report: Meetika, Priyank, Yash

Writing the code: Priyank, Meetika, Yash

Final report: Meetika, Yash

Chapter 9

References

- https://www.coursera.org/learn/convolutional-neural-networks/home/info
- https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_table_of_contents_feature2d/py_table_of_contents_feature2d.html
- https://www.kaggle.com/humananalog/keras-generator-for-reading-directly-from-bson
- https://www.linkedin.com/pulse/image-classifier-using-tflearn-navin-manaswi
- https://www.youtube.com/watch?v=3BXfw_1_TF4
- http://parneetk.github.io/blog/cnn-cifar10/
- https://cs231n.github.io/convolutional-networks/
- http://cv-tricks.com/tensorflow-tutorial/training-convolutional-neural-network-for-image-classification/
- https://www.tensorflow.org/tutorials/layers