YELP DATASET PHOTO CLASSIFICATION CHALLENGE

Presented By:
Mahesh Narayan Krishnamurthy (mxk166930)
Nitesh Srivatsav Bangalore Naresh(nxb162630)
Vignesh Vijaykumar (vxv160730)
Sreenivas Venkitachalam (sxv163530)

INDEX

INTRODUCTION

Photo classification is one of the trending topics in Machine Learning. Photo classification has many applications in different fields such as restaurant industry, remote sensing data applications, study of evolution and other medical fields. Photo classification is of two types -

Unsupervised image classification and ***Supervised*** Image classification. In this project, we are classifying restaurant related items using Supervised learning techniques.

PROBLEM STATEMENT

Our primary goal in this project is to classify restaurant based images as:

- 'Outside'
- 'Inside'
- 'Food'
- 'Drink'

To accomplish this task we are training a powerful classifier on a huge dataset of images.

DATASET DESCRIPTION

In the dataset we are using, there are 196,278 images and a file containing a collection of json files as shown below:

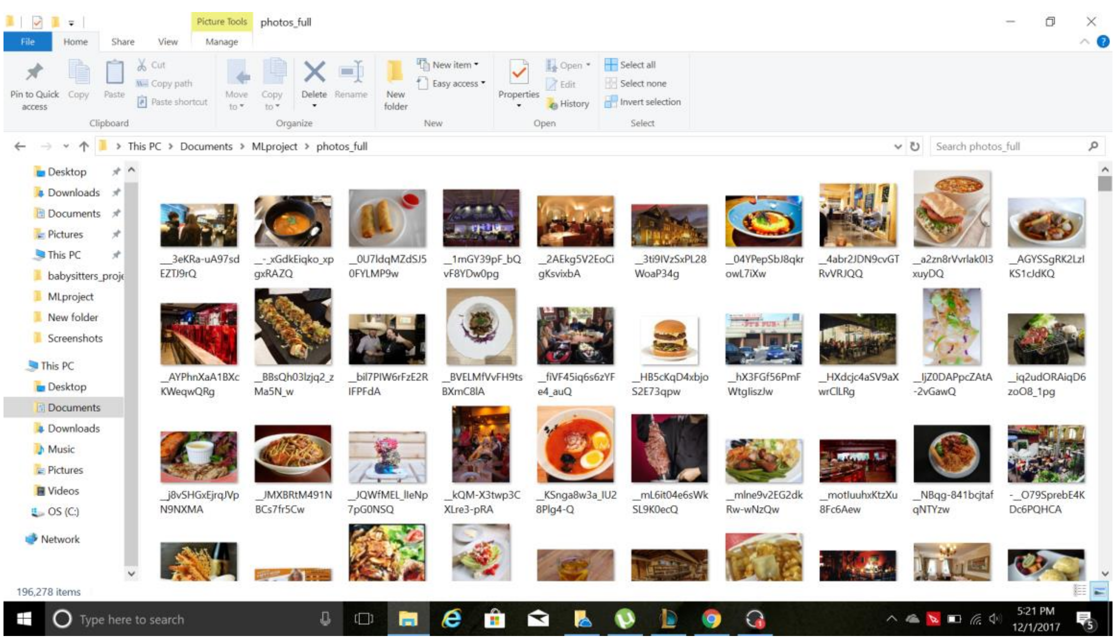


Fig 1:Folder containing all the 196,278 images

DATA PREPROCESSING

The json file is a collection of dictionaries and each dictionary contains information about a particular image in the dataset.

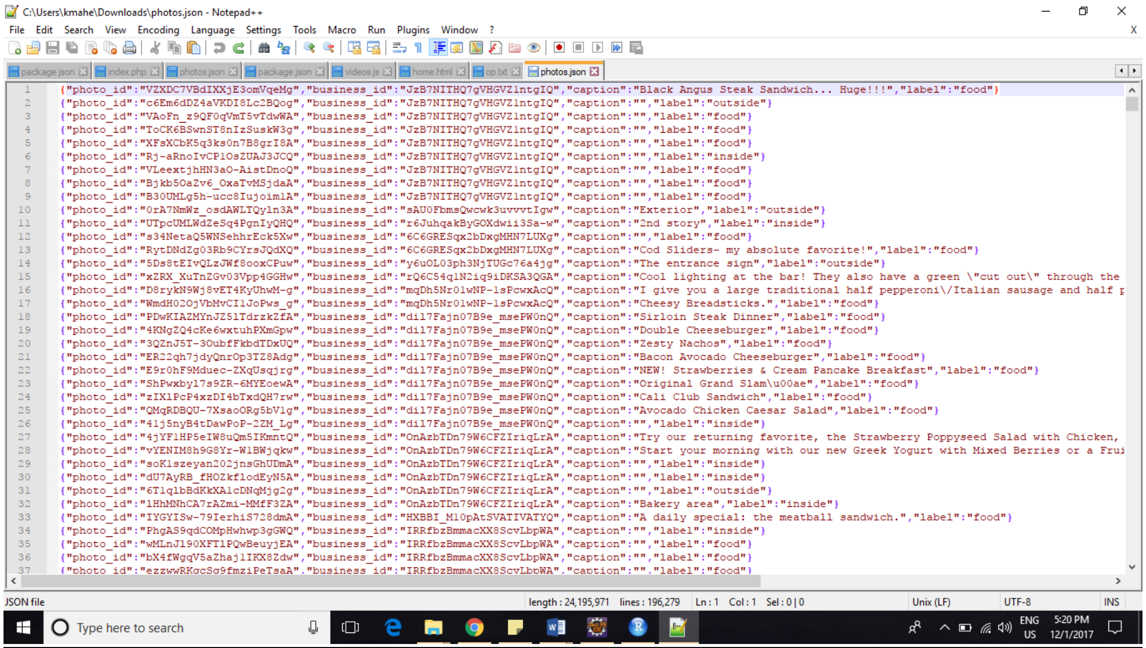


Fig 2:View of the json file

We use **preprocessing.py** to do the following in order

- Convert the json file to a list of json files.
- Extract the 'photoid' and 'label' attribute for each image and appending them into a dataframe.
- Convert the label column to categorical values using LabelEncoder.
- Then we print the count of each label in the dataset. This gives an idea of the dataset distribution.
- Finally we convert the dataframe to a csv file which will be a convenient input to the classifier that we are going to use.

WORK METHODOLOGY

We are implementing the following models to accomplish this classification task:

- SVM - Support Vector Machines
- CNN - Convolutional Neural Networks (A Deep Learning Technique)

We shall present a comparative study of image classification using the above supervised learning techniques. We shall understand and present the merits and de-merits of using the two techniques and finally identify the more accurate methodology for this problem.

SVM APPROACH:

Let us look at the steps in this approach:

- We use scala and LIBSVM - SVM ML library to perform classification.
- After pre-processing, we read the input csv files to obtain the label.
- Images are read from the input directory from the local machine.
- First we convert the images into gray scale format.
- Then we call the resize function to resize the image to 30 x 30 pixels format.
- Then we call the MainTest method to convert the image into vector format.
- We save vector along with label in LIBSVM format as specified by the Apache Spark MLlib.
- Then we model split on 60:40 for training and testing.
- Then, we feed the saved input data to the model and run over 100 iterations.
- Then, we test the model on input data and report the accuracy of the model.
- We also predict the area under the ROC curve and display the same.

Code Screenshots for SVM

Test.scala   Test.scala   FinalProject/pom.xml

```scala
package ImageClass
import javax.imageio.ImageIO
import java.awt.AlphaComposite
import java.awt.Color
import java.awt.Graphics2D
import java.awt.image.BufferedImage
import java.awt.image.DataBufferByte
import java.awt.image.Raster
import java.io.BufferedWriter
import java.io.File
import java.io.FileWriter
import java.io.IOException
import java.io.PrintWriter
import java.nio.file.FileVisitResult
import java.nio.file.Files
import java.nio.file.Path
import java.nio.file.Paths
import java.nio.file.SimpleFileVisitor
import java.nio.file.attribute.BasicFileAttributes
import java.util.ArrayList

import org.apache.spark.mllib.classification.{SVMModel, SVMWithSGD}
import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics
import org.apache.spark.mllib.util.MLUtils



object Test extends App {

```



```scala


object Test extends App {

```



```scala
def toGray(image: BufferedImage, label: String): Unit = {
  val width = image.getWidth()
  val height = image.getHeight()
  for(i <- 0 until height){
  for(j <- 0 until width){
  val c = new Color(image.getRGB(j, i))
  val red = (c.getRed() * 0.21).toInt
  val green = (c.getGreen() * 0.72).toInt
  val blue = (c.getBlue() *0.07).toInt
  val sum = red + green + blue
  val newColor = new Color(sum,sum,sum)
  image.setRGB(j,i,newColor.getRGB())
  }
  }
  //ImageIO.write(image, "jpg", new File("C:/Users/VIGNESH/Pictures/Screenshots/test.jpg"))
  mainTest(image,label)
  }

def createResizedCopy(originalImage: BufferedImage, scaledWidth: Integer, scaledHeight: Integer,
  var imageType = 0
  if(preserveAlpha == true){
  imageType = BufferedImage.TYPE_INT_RGB
  }
  else{
  imageType = BufferedImage.TYPE_INT_ARGB
```

```
  }
  }
  //ImageIO.write(image, "jpg", new File("C:/Users/VIGNESH/Pictures/Screenshots/test.jpg"))
  mainTest(image,label)
  }

def createResizedCopy(originalImage: BufferedImage, scaledWidth: Integer, scaledHeight: Integer,
  var imageType = 0
  if(preserveAlpha == true){
  imageType = BufferedImage.TYPE_INT_RGB
  }
  else{
  imageType = BufferedImage.TYPE_INT_ARGB
  }
  val scaledBI = new BufferedImage(scaledWidth, scaledHeight, imageType)
  val g = scaledBI.createGraphics()
  if (preserveAlpha) {
  g.setComposite(AlphaComposite.Src)
  }
  g.drawImage(originalImage, 0, 0, scaledWidth, scaledHeight, null)
  g.dispose()
  return scaledBI
  }

def mainTest(img: BufferedImage, label:String){
  try{
  /* args[2] is the Class of the image, Class = Male/Female. Vector will be written into a text
  try{

```


```
def mainTest(img: BufferedImage, label:String){
  try{
  /* args[2] is the Class of the image, Class = Male/Female. Vector will be written into a text
  try{
```


```

  val out = new PrintWriter(new BufferedWriter(new FileWriter("C:/Users/VIGNESH/Pictures/Screens
  out.println("")
  //val img = ImageIO.read(new File("C:/Users/VIGNESH/Pictures/Screenshots/test.jpg"))
  val raster=img.getData()
  val w=raster.getWidth()
  val h=raster.getHeight()
  //out.print(file1.getName());
  //out.print("test.jpg");
  //out.print(","+"test"+",")
  out.print(label+ " ")
  var count = 1
  for (x <- 0 until w)
  {
  for(y <- 0 until h)
  {
  out.print(count+":"+raster.getSample(x,y,0)+" ")
  count = count+1
  }
  //out.print(" ")
  }
  out.println("")
  }
  catch{
```

```
SparkConf conf = new SparkConf().setAppName("SVM vs Navie Bayes")
SparkContext sc = new SparkContext(conf)

// Load training data in LIBSVM format.
val data = MLUtils.loadLibSVMFile(sc, "data/mllib/sample_libsvm_data.txt")

// Split data into training (60%) and test (40%).
val splits = data.randomSplit(Array(0.6, 0.4), seed = 11L)
val training = splits(0).cache()
val test = splits(1)

// Run training algorithm to build the model
val numIterations = 100
val model = SVMWithSGD.train(training, numIterations)

// Clear the default threshold.
model.clearThreshold()

// Compute raw scores on the test set.
val scoreAndLabels = test.map { point =>
val score = model.predict(point.features)
(score, point.label)
}

// Get evaluation metrics.
val metrics = new BinaryClassificationMetrics(scoreAndLabels)
val auROC = metrics.areaUnderROC()

println("Area under ROC = " + auROC)
```

```
def test() {
  // read original image, and obtain width and height
  //val path = new File("C:/Users/VIGNESH/Pictures/Screenshots/Test/photo20")
  //val files = path.listFiles()
  val bufferedSource = io.Source.fromFile("C:/Users/VIGNESH/Pictures/Screenshots/Test/inputargs.
    for (line <- bufferedSource.getLines) {
        val cols = line.split(",").map(_.trim)
        //val temp = "C:/Users/VIGNESH/Pictures/Screenshots/Test/"+cols(0)
        //println("C:/Users/VIGNESH/Pictures/Screenshots/Test/"+cols(0)+","+cols(1))
        val photo1 = ImageIO.read(new File("C:/Users/VIGNESH/Pictures/Screenshots/Test/"+cols(0)
        val photo2 = createResizedCopy(photo1, 180, 120, true)
        toGray(photo2,cols(1))
    }
    bufferedSource.close
  /*for(i <- 0  until files.length){
```

Output for SVM

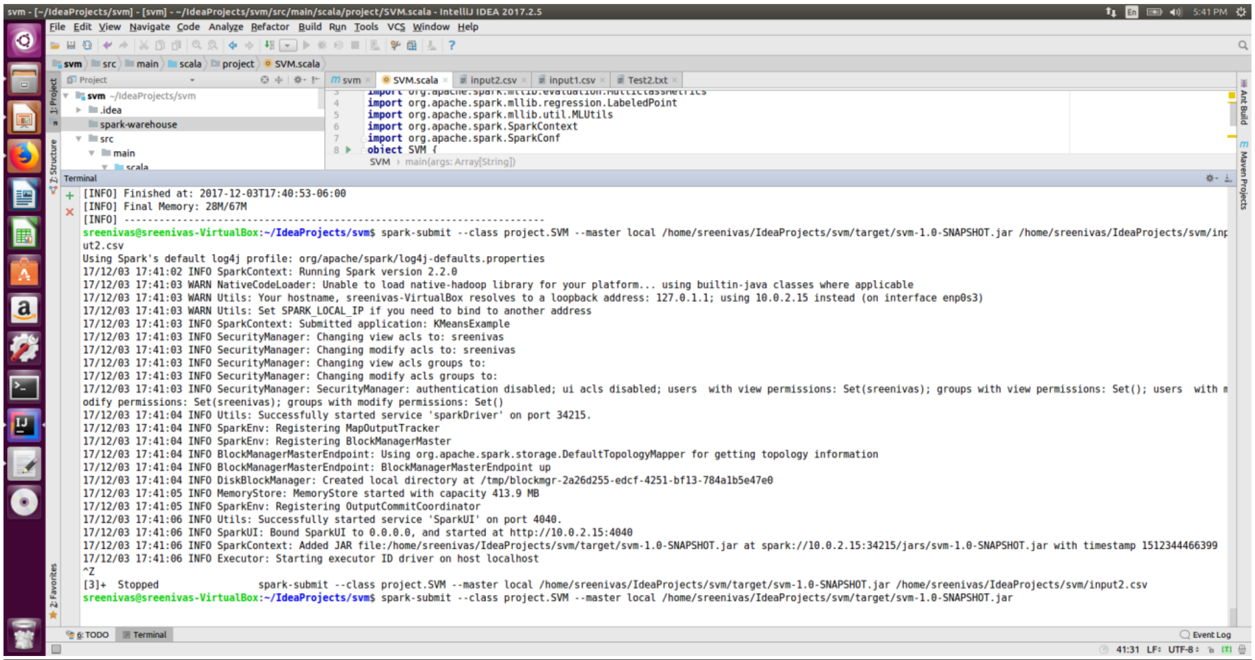


Fig 3:Output for SVM (I)

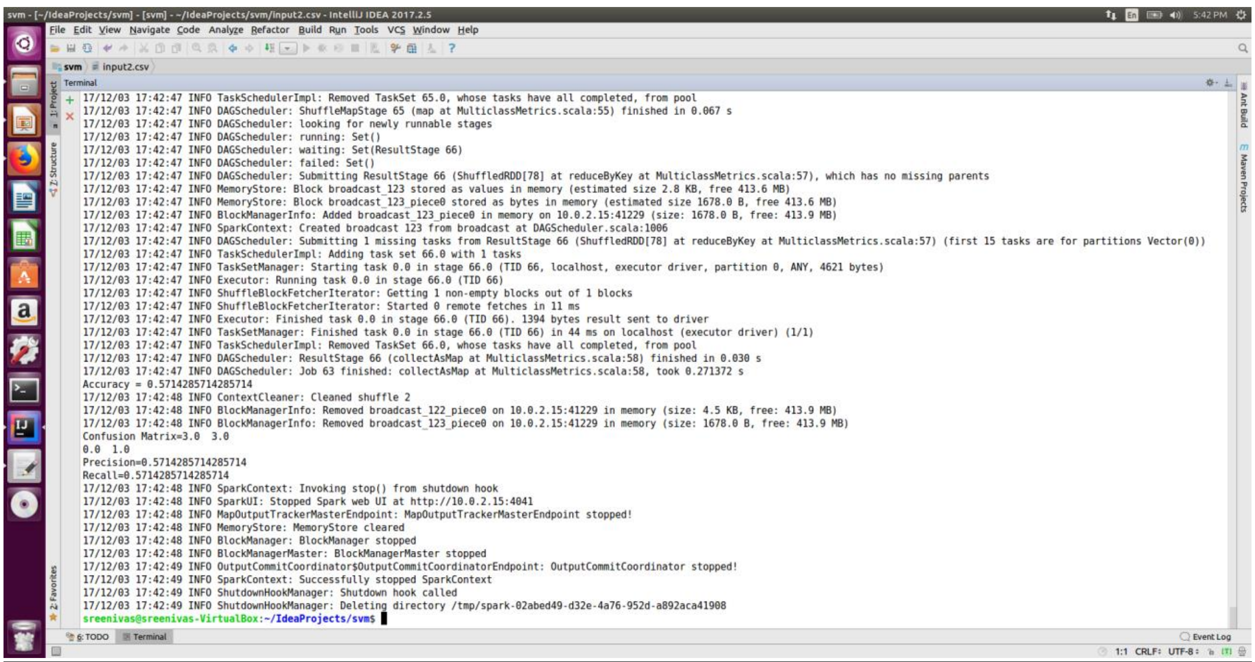


Fig 4:Output for SVM (II)

PYTORCH APPROACH - CNN

Let us look at the steps in this approach:

- We perform preprocessing as the first step where we extract out the necessary columns(photo id and label) for the classification process.
- The images and their corresponding labels are in two different files(image folder and the json file respectively).We create an image dataset with each tuple contain image and its corresponding label.We then feed this as input to the CNN.
- The images are loaded in and transformed into tensors of normalized range[0,1].
- We then define a CNN to take in 3-channel images(RGB) as input.
- We then define the loss function(negative log likelihood) and train the images on the network we defined before.
- We use 60% of the images for training, 20% for validation and 20% for testing.
- Once we train,validate and test the network, save the model and its parameters.
- We then use the model to predict the images and output the most probable class for an image.
- We construct a confusion matrix and F-score of the parameters to depict the accuracy and precision of the model.
- Since there are about 200,000 images in the dataset we are running the CNN on a GPU.We are using the Amazon EC2 instance to run the network.

Code Screenshots for CNN:

C:\Users\kmahe\Desktop\MS Fall 17\Machine Learning\Project\output20k\convnetlatest.py - Notepad++

```python
# -*- coding: utf-8 -*-
"""
Created on Sun Nov  5 18:36:57 2017

@author: Sreenivas, Mahesh, Vignesh, Nitesh
"""
import torch
from torch.autograd import Variable
import torchvision.transforms as transforms
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import argparse
import imagelabel as Image
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,f1_score
import warnings
#Training Settings
parser = argparse.ArgumentParser(description='Image classification')
parser.add_argument('--data', metavar='DIR',
                    help='path to image directory')
parser.add_argument('--label',metavar='FILE',help='path to label file')
parser.add_argument('--batchsize', type=int, default=64, metavar='N',
                    help='input batch size for training (default: 64)')
parser.add_argument('--testbatchsize', type=int, default=100, metavar='N',
                    help='input batch size for testing (default: 1000)')
parser.add_argument('--validbatchsize', type=int, default=64, metavar='N',
                    help='input batch size for validation (default: 64)')
parser.add_argument('--epochs', type=int, default=100, metavar='N',
                    help='number of epochs to train (default: 10)')
parser.add_argument('--lr', type=float, default=0.01, metavar='LR',
                    help='learning rate (default: 0.01)')
parser.add_argument('--momentum', type=float, default=0.5, metavar='M',
                    help='SGD momentum (default: 0.5)')
#I ahve made no cuda default True->change when gpu is set up
```

C:\Users\kmahe\Desktop\MS Fall 17\Machine Learning\Project\output20k\convnetlatest.py - Notepad++

```python
                    help='disables CUDA training')
parser.add_argument('--seed', type=int, default=1, metavar='S',
                    help='random seed (default: 1)')
parser.add_argument('--log-interval', type=int, default=10, metavar='N',
                    help='how many batches to wait before logging training status')

#Buiding convolutional neural network
class Net(nn.Module):
    def __init__(self):
        super(Net,self).__init__()
        self.conv1=nn.Conv2d(3,5,5)
        self.pool=nn.MaxPool2d(2)
        self.conv2=nn.Conv2d(5,5,5)
        self.fc1=nn.Linear(5*53*53,20)
        self.fc2=nn.Linear(20,25)
        self.fc3=nn.Linear(25,5)
    def forward(self,x):
        x=self.pool(F.relu(self.conv1(x)))
        x=self.pool(F.relu(self.conv2(x)))
        x=x.view(-1,5*53*53)
        x=F.relu(self.fc1(x))
        x=F.dropout(x,training=self.training)
        x=F.relu(self.fc2(x))
        x=F.dropout(x,training=self.training)
        x=self.fc3(x)
        return F.log_softmax(x)
#Transformation functions
def train_transformation(train_dataset,transform):
    train_transform=[]
    for value in train_dataset:
        train_transform.append((transform(value[0]),value[1]))
    return train_transform
def test_transformation(test_dataset,transform):
    test_transform=[]
    for value in test_dataset:
```

C:\Users\kmahe\Desktop\MS Fall 17\Machine Learning\Project\output20k\convnetlatest.py - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

photos.json | package.json | videos.js | home.html | op.txt | photos.json | op.txt | convnetlatest.py

```python
            x=self.fc3(x)
            return F.log_softmax(x)
#Transformation functions
def train_transformation(train_dataset,transform):
    train_transform=[]
    for value in train_dataset:
        train_transform.append((transform(value[0]),value[1]))
    return train_transform
def test_transformation(test_dataset,transform):
    test_transform=[]
    for value in test_dataset:
        test_transform.append((transform(value[0]),value[1]))
    return test_transform
def validation_transformation(test_dataset,transform):
    valid_transform=[]
    for value in test_dataset:
        valid_transform.append((transform(value[0]),value[1]))
    return valid_transform
#Train function
def train(model,train_loader,optimizer,epoch):
    model.train()
    for batch_idx,(data,target) in enumerate(train_loader):
        if args.cuda:
            data,target=data.cuda(),target.cuda()
        data,target=Variable(data),Variable(target)
        optimizer.zero_grad()
        output=model(data)
        loss=F.nll_loss(output,target.long())
        loss.backward()
        optimizer.step()
        if batch_idx % args.log_interval == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(train_loader.dataset),
                100. * batch_idx / len(train_loader), loss.data[0]))
#Test function
```

Python file | length : 9,708 lines : 254 | Ln : 70 Col : 22 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS

Type here to search | ENG US | 3:35 PM 12/3/2017

C:\Users\kmahe\Desktop\MS Fall 17\Machine Learning\Project\output20k\convnetlatest.py - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

photos.json | package.json | videos.js | home.html | op.txt | photos.json | op.txt | convnetlatest.py

```python
                epoch, batch_idx * len(data), len(train_loader.dataset),
                100. * batch_idx / len(train_loader), loss.data[0]))
#Test function
def test(model,test_loader):
    model.eval()
    correct=0
    counter=0
    for batch_idx,(data,target) in enumerate(test_loader):
        if args.cuda:
            data,target=data.cuda(),target.cuda()
        output=model(Variable(data))
        _,predicted=torch.max(output.data,1)
        if counter==0:
            trueList=target.long()
            predictedList=predicted.view(len(predicted))
        else:
            trueList=torch.cat((trueList,target.long()),0)
            predictedList=torch.cat((predictedList,predicted.view(len(predicted))),0)
        correct+=(predicted==target.long().view(-1,1)).cpu().sum()
        counter=counter+1
    accuracy=100*(correct/len(test_loader.dataset))
    return accuracy,trueList,predictedList
#Converting categorical variables to original labels
def vallabels(valueList,labels):
    convList=[]
    for value in valueList:
        convList.append(labels[value])
    return convList
#Main function
if __name__=="__main__":
    args = parser.parse_args()
    args.cuda = not args.no_cuda and torch.cuda.is_available()
    torch.manual_seed(args.seed)
    if args.cuda:
        torch.cuda.manual_seed(args.seed)
    model=Net()
```

Python file | length : 9,708 lines : 254 | Ln : 70 Col : 22 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS

Type here to search | ENG US | 3:35 PM 12/3/2017

C:\Users\kmahe\Desktop\MS Fall 17\Machine Learning\Project\output20k\convnetlatest.py - Notepad++

```python
    accuracy=100*(correct/len(test_loader.dataset))
    return accuracy,trueList,predictedList
#Converting categorical variables to original labels
def vallabels(valueList,labels):
    convList=[]
    for value in valueList:
        convList.append(labels[value])
    return convList
#Main function
if __name__=="__main__":
    args = parser.parse_args()
    args.cuda = not args.no_cuda and torch.cuda.is_available()
    torch.manual_seed(args.seed)
    if args.cuda:
        torch.cuda.manual_seed(args.seed)
    model=Net()
    if args.cuda:
        model.cuda()
    #optimizer and criterion for neural network
    optimizer=optim.SGD(model.parameters(),lr=args.lr,momentum=args.momentum)
    #normalizing data
    normalize=transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                    std=[0.229, 0.224, 0.225])
    train_transform=transforms.Compose([transforms.RandomSizedCrop(224),
            transforms.RandomHorizontalFlip(),
            transforms.ToTensor(),
            normalize])
    test_transform=transforms.Compose([transforms.RandomSizedCrop(224),
            transforms.ToTensor(),
            normalize])
    validation_transform=transforms.Compose([transforms.RandomSizedCrop(224),
            transforms.ToTensor(),
            normalize])
    #taking directory with image folder from command line
    imagefile=args.data
```

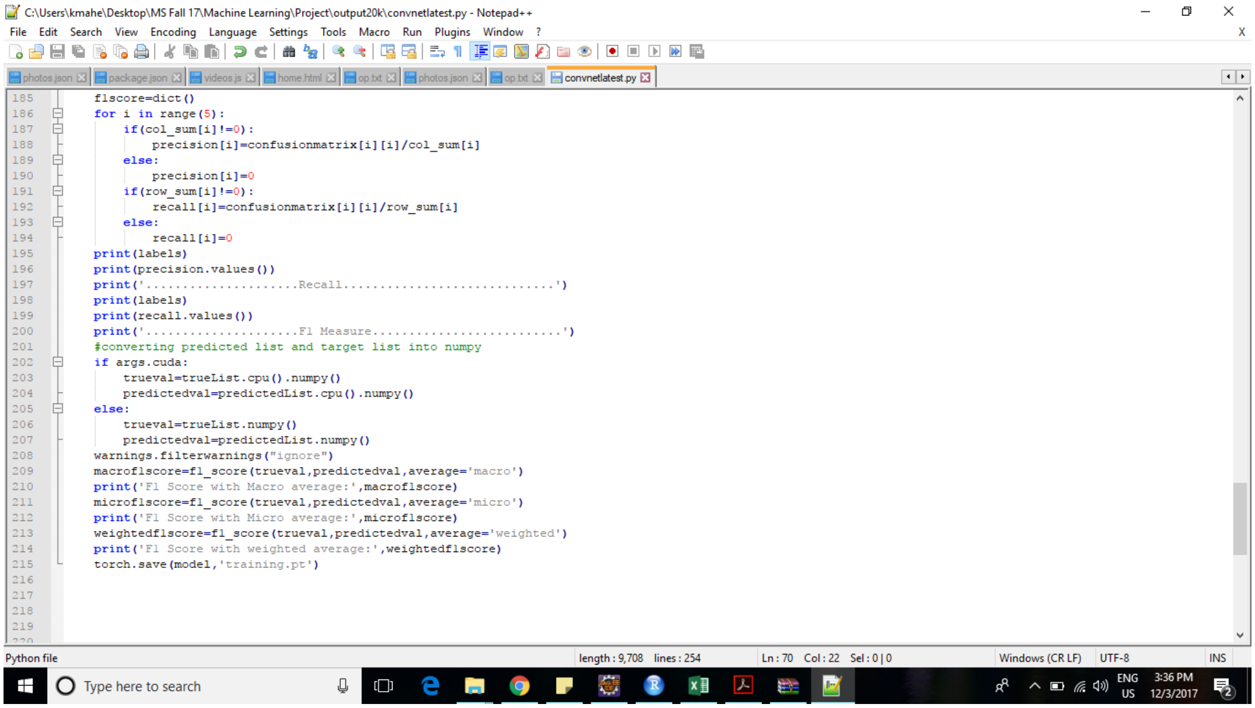
Python file length : 9,708 lines : 254 Ln : 70 Col : 22 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

C:\Users\kmahe\Desktop\MS Fall 17\Machine Learning\Project\output20k\convnetlatest.py - Notepad++

```python
            transforms.RandomHorizontalFlip(),
            transforms.ToTensor(),
            normalize])
    test_transform=transforms.Compose([transforms.RandomSizedCrop(224),
            transforms.ToTensor(),
            normalize])
    validation_transform=transforms.Compose([transforms.RandomSizedCrop(224),
            transforms.ToTensor(),
            normalize])
    #taking directory with image folder from command line
    imagefile=args.data
```

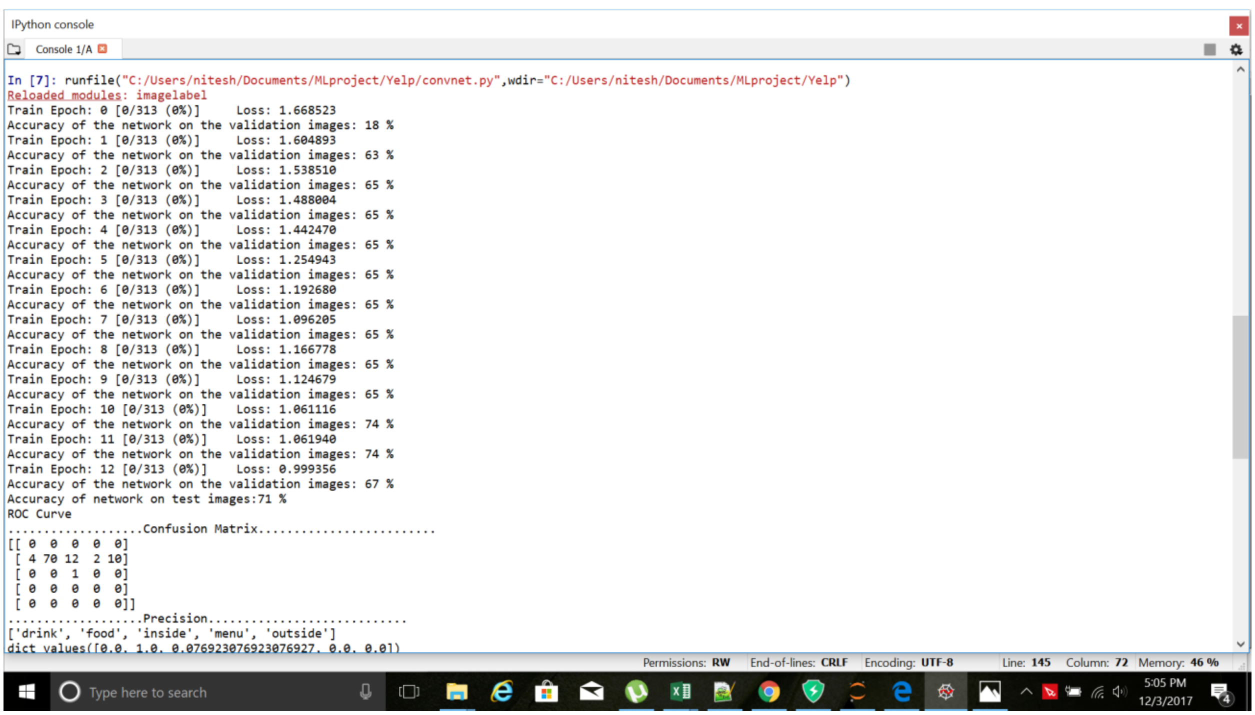

```python
    labelfile=args.label
    #calling class from imageloading
    imagedataset=Image.Imagedataset(imagefile,labelfile)
    #splitting datasets into train,test and validation datasets
    trainvaliddataset,testdataset=train_test_split(imagedataset,test_size=0.2,random_state=0)
    traindataset,validdataset=train_test_split(trainvaliddataset,test_size=0.2,random_state=0)
    #Transformation of image into tensor
    traindataset=train_transformation(traindataset,train_transform)
    testdataset=test_transformation(testdataset,test_transform)
    validdataset=validation_transformation(validdataset,validation_transform)
    #dataloader for train test and validation
    train_loader=torch.utils.data.DataLoader(traindataset,batch_size=args.batchsize)
    test_loader=torch.utils.data.DataLoader(testdataset,batch_size=args.testbatchsize)
    validation_loader=torch.utils.data.DataLoader(validdataset,batch_size=args.validbatchsize)
    #Training phase
    prev_accuracy=0
    for epoch in range(0,args.epochs):
        train(model,train_loader,optimizer,epoch)
        accuracy,predictedList,trueList=test(model,validation_loader)
        print('Accuracy of the network on the validation images: %d %%' % (accuracy))
        #if(prev_accuracy>accuracy):break
        #prev_accuracy=accuracy
    accuracy,predictedList,trueList=test(model,test_loader)
    print('Accuracy of network on test images:%d %%' %(accuracy))
```


Python file length : 9,708 lines : 254 Ln : 70 Col : 22 Sel : 0 | 0 Windows (CR LF) UTF-8 INS



```
    f1score=dict()
    for i in range(5):
        if(col_sum[i]!=0):
            precision[i]=confusionmatrix[i][i]/col_sum[i]
        else:
            precision[i]=0
        if(row_sum[i]!=0):
            recall[i]=confusionmatrix[i][i]/row_sum[i]
        else:
            recall[i]=0
    print(labels)
    print(precision.values())
    print('.....................Recall.............................')
    print(labels)
    print(recall.values())
    print('.....................F1 Measure.........................')
    #converting predicted list and target list into numpy
    if args.cuda:
        trueval=trueList.cpu().numpy()
        predictedval=predictedList.cpu().numpy()
    else:
        trueval=trueList.numpy()
        predictedval=predictedList.numpy()
    warnings.filterwarnings("ignore")
    macroflscore=f1_score(trueval,predictedval,average='macro')
    print('F1 Score with Macro average:',macroflscore)
    microflscore=f1_score(trueval,predictedval,average='micro')
    print('F1 Score with Micro average:',microflscore)
    weightedflscore=f1_score(trueval,predictedval,average='weighted')
    print('F1 Score with weighted average:',weightedflscore)
    torch.save(model,'training.pt')
```

Output for CNN



```
In [7]: runfile("C:/Users/nitesh/Documents/MLproject/Yelp/convnet.py",wdir="C:/Users/nitesh/Documents/MLproject/Yelp")
Reloaded modules: imagelabel
Train Epoch: 0 [0/313 (0%)]     Loss: 1.668523
Accuracy of the network on the validation images: 18 %
Train Epoch: 1 [0/313 (0%)]     Loss: 1.604893
Accuracy of the network on the validation images: 63 %
Train Epoch: 2 [0/313 (0%)]     Loss: 1.538510
Accuracy of the network on the validation images: 65 %
Train Epoch: 3 [0/313 (0%)]     Loss: 1.488004
Accuracy of the network on the validation images: 65 %
Train Epoch: 4 [0/313 (0%)]     Loss: 1.442470
Accuracy of the network on the validation images: 65 %
Train Epoch: 5 [0/313 (0%)]     Loss: 1.254943
Accuracy of the network on the validation images: 65 %
Train Epoch: 6 [0/313 (0%)]     Loss: 1.192680
Accuracy of the network on the validation images: 65 %
Train Epoch: 7 [0/313 (0%)]     Loss: 1.096205
Accuracy of the network on the validation images: 65 %
Train Epoch: 8 [0/313 (0%)]     Loss: 1.166778
Accuracy of the network on the validation images: 65 %
Train Epoch: 9 [0/313 (0%)]     Loss: 1.124679
Accuracy of the network on the validation images: 65 %
Train Epoch: 10 [0/313 (0%)]    Loss: 1.061116
Accuracy of the network on the validation images: 74 %
Train Epoch: 11 [0/313 (0%)]    Loss: 1.061940
Accuracy of the network on the validation images: 74 %
Train Epoch: 12 [0/313 (0%)]    Loss: 0.999356
Accuracy of the network on the validation images: 67 %
Accuracy of network on test images:71 %
ROC Curve
...................Confusion Matrix.........................
[[ 0  0  0  0  0]
 [ 4 70 12  2 10]
 [ 0  0  1  0  0]
 [ 0  0  0  0  0]
 [ 0  0  0  0  0]]
...................Precision............................
['drink', 'food', 'inside', 'menu', 'outside']
dict_values([0.0, 1.0, 0.07692307692307692 7, 0.0, 0.0])
```

Fig 5:Output for 500 images (I)

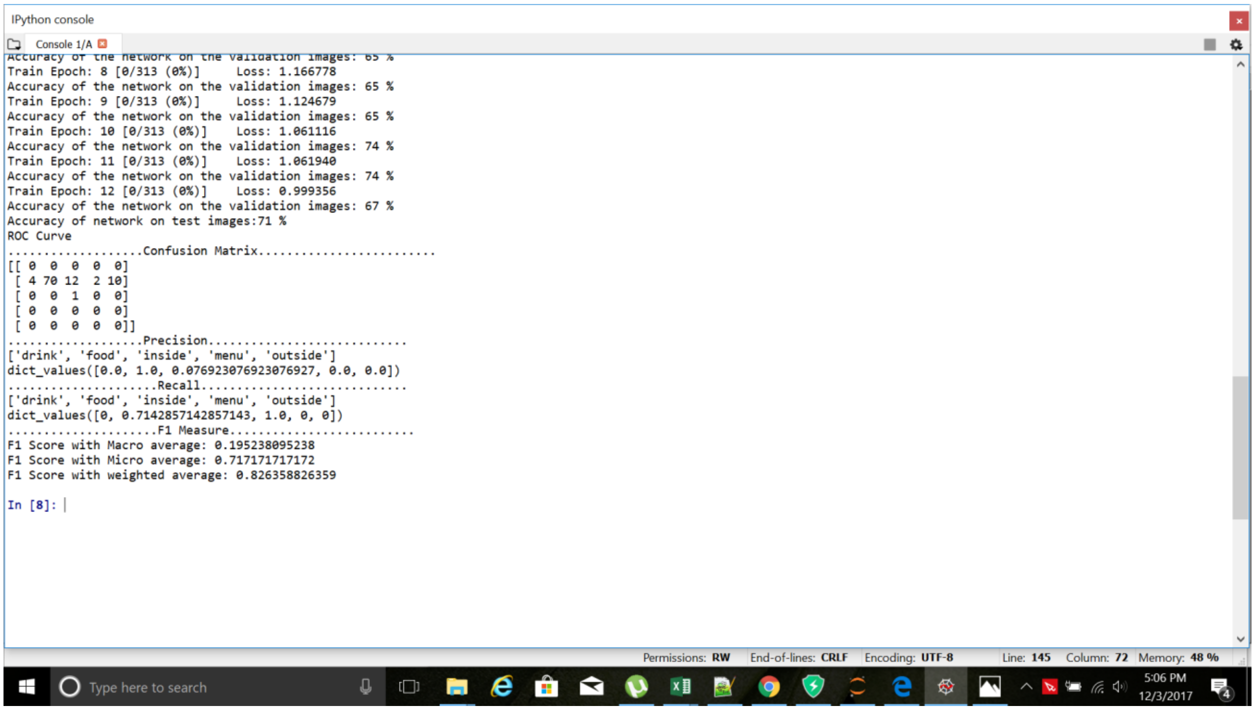


Fig 6:Output for 500 images (II)

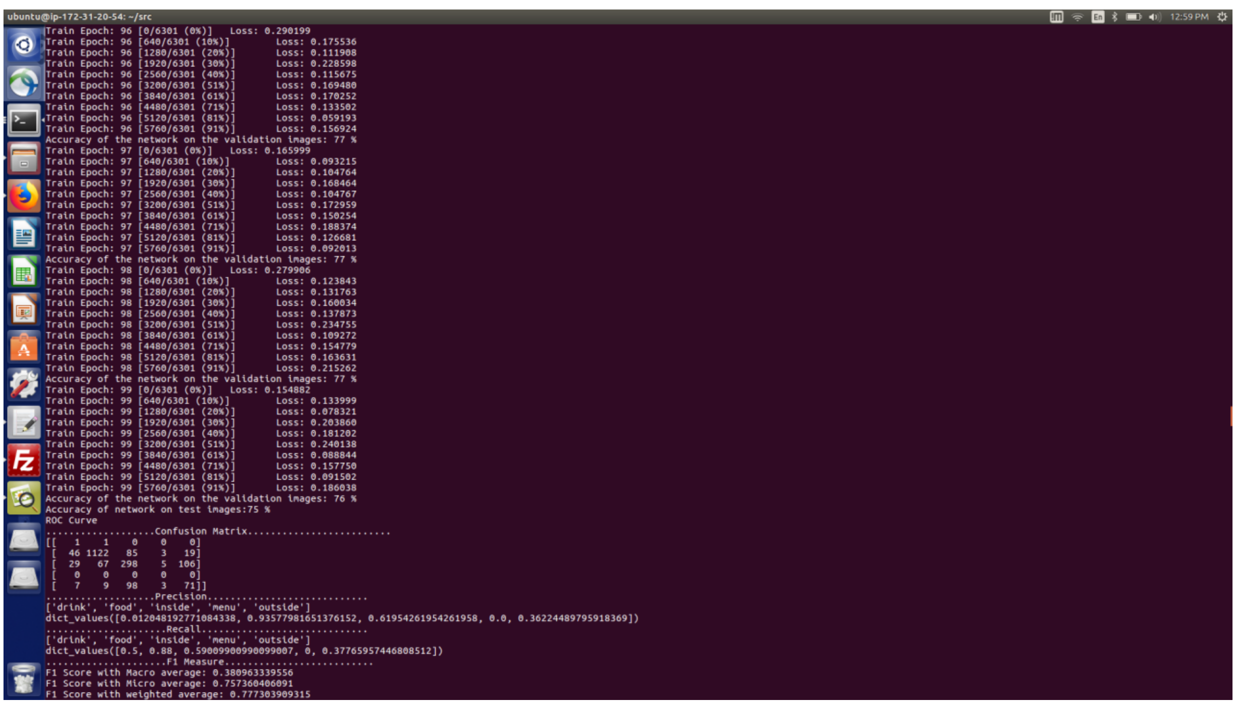


Fig 7:Output for 20k images (I)

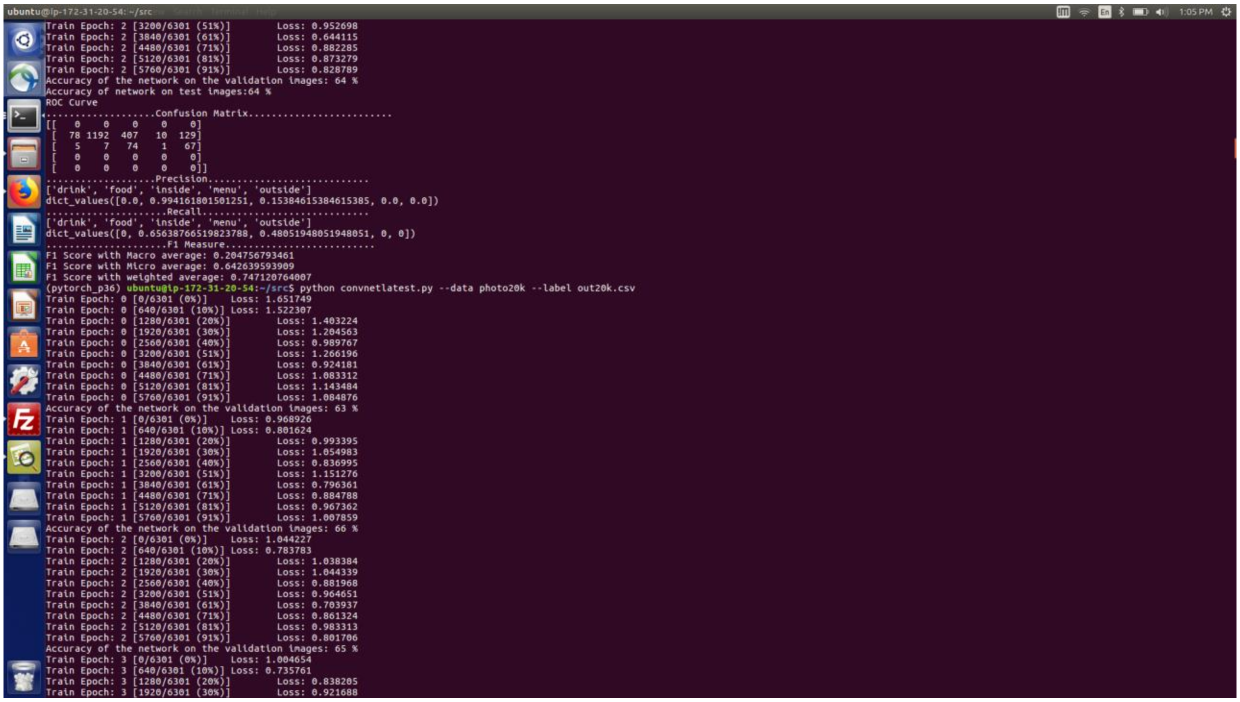


Fig 8:Output for 20k images (II)

RESULTS

CNN Classifier

Metrics	Value
Accuracy for 500 images	**71%**
Accuracy for 20k images	**75%**
F score for 500 images	**0.8263**
F score for 20K images	**0.777**
Confusion Matrix for 500 images	**(0,0,0,0,0 4,70,12,2,10 0,0,1,0,0 0,0,0,0,0 0,0,0,0,0)**
Confusion Matrix for 20K images	**(1,1,0,0,0**

	46,1112,85,3,19 29,67,298,5,67 0,0,0,0,0 7,98,3,71,7)

SVM Classifier (Tested for 50 images)

Metrics	**Value**
Accuracy	**57.14%**
Precision	**0.571**
Recall	**0.571**
Confusion Matrix	**(3.0,3.0 0.0,1.0)**

- On observing the output from the classifiers we can see that SVM is mainly dependent on identifying the core set of values from the vector output which it perceives to be the support vectors
- Once these are established, all the other points or values become redundant which leads to problems in classification leading to lowering precision
- However, when there are genuinely distinct images with lesser number of images to train and test, we can see that the classifier works better
- On the other hand, CNN being a deep learning algorithm uses several hidden layers and conv-nets to accurately classify the image
- It is able to work even better with images which are close or resemble one another with respect to the classes
- The CNN when run on the GPU is much faster and far more powerful than the SVM classifier.

REFERENCES

1. http://blogs.quovantis.com/image-classification-using-apache-spark-with-linear-svm/
2. https://spark.apache.org/docs/2.2.0/mllib-linear-methods.html
3. http://otfried.org/scala/image.html
4. http://pytorch.org/tutorials/
5. https://aws.amazon.com/documentation/ec2/