

# **YELP DATASET PHOTO CLASSIFICATION CHALLENGE**

**Presented By:**

Mahesh Narayan Krishnamurthy (mxk166930)

Nitesh Srivatsav Bangalore Naresh(nxb162630)

Vignesh Vijaykumar (vxv160730)

Sreenivas Venkitachalam (sxv163530)

**INDEX**

- 1) Introduction
- 2) Problem Statement
- 3) Dataset Description
- 4) Data Preprocessing
- 5) Work Methodology
- 6) Results
- 7) References

## **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**

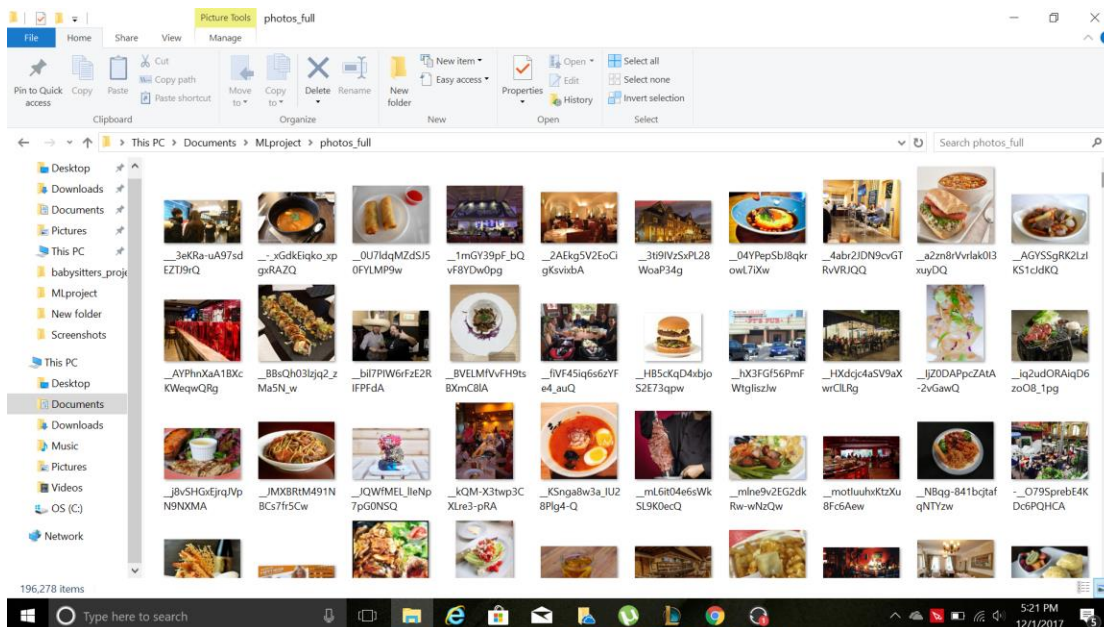
Our primary goal in this project is to classify restaurant based images as:

- '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.

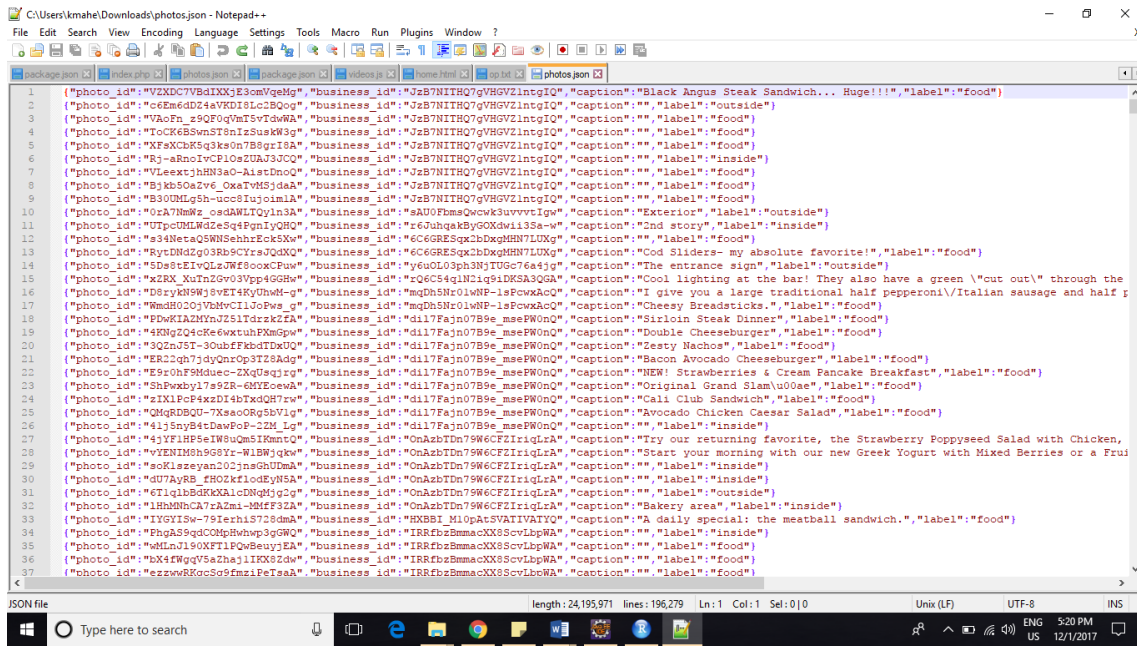


Fig 2:View of the json file

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

- Convert the json file to a list of json files.
- Extract the 'photo\_id' 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
1 package ImageClass
2 import javax.imageio.ImageIO
3 import java.awt.AlphaComposite
4 import java.awt.Color
5 import java.awt.Graphics2D
6 import java.awt.image.BufferedImage
7 import java.awt.image.DataBufferByte
8 import java.awt.image.Raster
9 import java.io.BufferedWriter
10 import java.io.File
11 import java.io.FileWriter
12 import java.io.IOException
13 import java.io.PrintWriter
14 import java.nio.file.FileVisitResult
15 import java.nio.file.Files
16 import java.nio.file.Path
17 import java.nio.file.Paths
18 import java.nio.file.SimpleFileVisitor
19 import java.nio.file.attribute.BasicFileAttributes
20 import java.util.ArrayList
21
22 import org.apache.spark.mllib.classification.{SVMModel, SVMWithSGD}
23 import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics
24 import org.apache.spark.mllib.util.MLUtils
25
26
27
28 object Test extends App {
29
26
27
28 object Test extends App {
29
30 def toGray(image: BufferedImage, label: String): Unit = {
31     val width = image.getWidth()
32     val height = image.getHeight()
33     for(i <- 0 until height){
34     for(j <- 0 until width){
35         val c = new Color(image.getRGB(j, i))
36         val red = (c.getRed() * 0.21).toInt
37         val green = (c.getGreen() * 0.72).toInt
38         val blue = (c.getBlue() * 0.07).toInt
39         val sum = red + green + blue
40         val newColor = new Color(sum, sum, sum)
41         image.setRGB(j, i, newColor.getRGB())
42     }
43     }
44     //ImageIO.write(image, "jpg", new File("C:/Users/VIGNESH/Pictures/Screenshots/test.jpg"))
45     mainTest(image, label)
46 }
47
48 def createResizedCopy(originalImage: BufferedImage, scaledWidth: Integer, scaledHeight: Integer,
49     var imageType = 0
50     if(preserveAlpha == true){
51         imageType = BufferedImage.TYPE_INT_RGB
52     }
53     else{
54         imageType = BufferedImage.TYPE_INT_ARGB
```

```

42     }
43     }
44     //ImageIO.write(image, "jpg", new File("C:/Users/VIGNESH/Pictures/Screenshots/test.jpg"))
45     mainTest(image,label)
46     }
47
48     def createResizedCopy(originalImage: BufferedImage, scaledWidth: Integer, scaledHeight: Integer,
49         var imageType = 0
50         if(preserveAlpha == true){
51             imageType = BufferedImage.TYPE_INT_RGB
52         }
53         else{
54             imageType = BufferedImage.TYPE_INT_ARGB
55         }
56         val scaledBI = new BufferedImage(scaledWidth, scaledHeight, imageType)
57         val g = scaledBI.createGraphics()
58         if (preserveAlpha) {
59             g.setComposite(AlphaComposite.Src)
60         }
61         g.drawImage(originalImage, 0, 0, scaledWidth, scaledHeight, null)
62         g.dispose()
63         return scaledBI
64     }
65
66     def mainTest(img: BufferedImage, label:String){
67         try{
68             /* args[2] is the Class of the image, Class = Male/Female. Vector will be written into a text
69             try{
70

```

```

65
66     def mainTest(img: BufferedImage, label:String){
67         try{
68             /* args[2] is the Class of the image, Class = Male/Female. Vector will be written into a text
69             try{
70
71             val out = new PrintWriter(new BufferedWriter(new FileWriter("C:/Users/VIGNESH/Pictures/Screenshots/test.jpg")))
72             out.println("")
73             //val img = ImageIO.read(new File("C:/Users/VIGNESH/Pictures/Screenshots/test.jpg"))
74             val raster=img.getData()
75             val w=raster.getWidth()
76             val h=raster.getHeight()
77             //out.print(file1.getName());
78             //out.print("test.jpg");
79             //out.print(", "+"test"+",")
80             out.print(label+ " ")
81             var count = 1
82             for (x <- 0 until w)
83             {
84                 for(y <- 0 until h)
85                 {
86                     out.print(count+": "+raster.getSample(x,y,0)+" ")
87                     count = count+1
88                 }
89                 //out.print(" ")
90             }
91             out.println("")
92         }
93         catch{

```



```

105 SparkConf conf = new SparkConf().setAppName("SVM vs Naive Bayes")
106 SparkContext sc = new SparkContext(conf)
107
108 // Load training data in LIBSVM format.
109 val data = MLUtils.loadLibSVMFile(sc, "data/mllib/sample_libsvm_data.txt")
110
111 // Split data into training (60%) and test (40%).
112 val splits = data.randomSplit(Array(0.6, 0.4), seed = 11L)
113 val training = splits(0).cache()
114 val test = splits(1)
115
116 // Run training algorithm to build the model
117 val numIterations = 100
118 val model = SVMWithSGD.train(training, numIterations)
119
120 // Clear the default threshold.
121 model.clearThreshold()
122
123 // Compute raw scores on the test set.
124 val scoreAndLabels = test.map { point =>
125   val score = model.predict(point.features)
126   (score, point.label)
127 }
128
129 // Get evaluation metrics.
130 val metrics = new BinaryClassificationMetrics(scoreAndLabels)
131 val auROC = metrics.areaUnderROC()
132
133 println("Area under ROC = " + auROC)
134
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



```
svm - [-/IdeaProjects/svm] - [-/IdeaProjects/svm/src/main/scala/project/SVM.scala - IntelliJ IDEA 2017.2.5
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

svm - [-/IdeaProjects/svm]
  src
    main
      scala
        project
          SVM.scala
            1 import org.apache.spark.{SparkConf, SparkContext}
            2 import org.apache.spark.mllib.regression.LabeledPoint
            3 import org.apache.spark.mllib.util.MLUtils
            4 import org.apache.spark.SparkContext
            5 import org.apache.spark.SparkConf
            6 object SVM {
            7   def main(args: Array[String]) {
            8     // ...
            9   }
            10 }

Terminal
+ [INFO] Finished at: 2017-12-03T17:40:53-06:00
+ [INFO] Final Memory: 28M/67M
+ [INFO] -----
greenivas@greenivas-VirtualBox:~/IdeaProjects/svm$ spark-submit --class project.SVM --master local /home/greenivas/IdeaProjects/svm/target/svm-1.0-SNAPSHOT.jar /home/greenivas/IdeaProjects/svm/input2.csv
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
17/12/03 17:41:02 INFO SparkContext: Running Spark version 2.2.0
17/12/03 17:41:03 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
17/12/03 17:41:03 WARN Utils: Your hostname, greenivas-VirtualBox resolves to a loopback address: 127.0.1.1; using 10.0.2.15 instead (on interface enp0s3)
17/12/03 17:41:03 INFO SparkContext: Submitted application: KMeansExample
17/12/03 17:41:03 INFO SecurityManager: Changing view acls to: greenivas
17/12/03 17:41:03 INFO SecurityManager: Changing modify acls to: greenivas
17/12/03 17:41:03 INFO SecurityManager: Changing view acls groups to:
17/12/03 17:41:03 INFO SecurityManager: Changing modify acls groups to:
17/12/03 17:41:03 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(greenivas); groups with view permissions: Set(); users with modify permissions: Set(greenivas); groups with modify permissions: Set()
17/12/03 17:41:04 INFO Utils: Successfully started service 'sparkDriver' on port 34215.
17/12/03 17:41:04 INFO SparkEnv: Registering MapOutputTracker
17/12/03 17:41:04 INFO SparkEnv: Registering BlockManagerMaster
17/12/03 17:41:04 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
17/12/03 17:41:04 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
17/12/03 17:41:04 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-2a26d255-edcf-4251-bf13-784a1b5e47e0
17/12/03 17:41:05 INFO MemoryStore: MemoryStore started with capacity 413.9 MB
17/12/03 17:41:05 INFO SparkEnv: Registering OutputCommitCoordinator
17/12/03 17:41:06 INFO Utils: Successfully started service 'SparkUI' on port 4040.
17/12/03 17:41:06 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://10.0.2.15:4040
17/12/03 17:41:06 INFO SparkContext: Added JAR file:/home/greenivas/IdeaProjects/svm/target/svm-1.0-SNAPSHOT.jar at spark://10.0.2.15:34215/jars/svm-1.0-SNAPSHOT.jar with timestamp 151234466399
17/12/03 17:41:06 INFO Executor: Starting executor ID driver on host localhost
^Z
[3]+  Stopped                  spark-submit --class project.SVM --master local /home/greenivas/IdeaProjects/svm/target/svm-1.0-SNAPSHOT.jar /home/greenivas/IdeaProjects/svm/input2.csv
greenivas@greenivas-VirtualBox:~/IdeaProjects/svm$ spark-submit --class project.SVM --master local /home/greenivas/IdeaProjects/svm/target/svm-1.0-SNAPSHOT.jar
```

Fig 3:Output for SVM (I)

```
svm - [-/IdeaProjects/svm] - [-/IdeaProjects/svm/input2.csv - IntelliJ IDEA 2017.2.5
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

svm - [-/IdeaProjects/svm]
  src
    main
      scala
        project
          SVM.scala
            1 import org.apache.spark.{SparkConf, SparkContext}
            2 import org.apache.spark.mllib.regression.LabeledPoint
            3 import org.apache.spark.mllib.util.MLUtils
            4 import org.apache.spark.SparkContext
            5 import org.apache.spark.SparkConf
            6 object SVM {
            7   def main(args: Array[String]) {
            8     // ...
            9   }
            10 }

Terminal
17/12/03 17:42:47 INFO TaskSchedulerImpl: Removed TaskSet 65.0, whose tasks have all completed, from pool
17/12/03 17:42:47 INFO DAGScheduler: ShuffleMapStage 65 (map at MulticlassMetrics.scala:55) finished in 0.067 s
17/12/03 17:42:47 INFO DAGScheduler: looking for newly runnable stages
17/12/03 17:42:47 INFO DAGScheduler: running: Set()
17/12/03 17:42:47 INFO DAGScheduler: waiting: Set(ResultStage 66)
17/12/03 17:42:47 INFO DAGScheduler: failed: Set()
17/12/03 17:42:47 INFO DAGScheduler: Submitting ResultStage 66 (ShuffledRDD[78] at reduceByKey at MulticlassMetrics.scala:57), which has no missing parents
17/12/03 17:42:47 INFO MemoryStore: Block broadcast_123_piece0 stored as bytes in memory (estimated size 1678.0 B, free 413.6 MB)
17/12/03 17:42:47 INFO BlockManagerInfo: Added broadcast_123_piece0 in memory on 10.0.2.15:41229 (size: 1678.0 B, free: 413.9 MB)
17/12/03 17:42:47 INFO SparkContext: Created broadcast_123 from broadcast at DAGScheduler.scala:1006
17/12/03 17:42:47 INFO DAGScheduler: Submitting 1 missing tasks from ResultStage 66 (ShuffledRDD[78] at reduceByKey at MulticlassMetrics.scala:57) (first 15 tasks are for partitions Vector(0))
17/12/03 17:42:47 INFO TaskSchedulerImpl: Adding task set 66.0 with 1 tasks
17/12/03 17:42:47 INFO TaskSetManager: Starting task 0.0 in stage 66.0 (TID 66, localhost, executor driver, partition 0, ANY, 4621 bytes)
17/12/03 17:42:47 INFO Executor: Running task 0.0 in stage 66.0 (TID 66)
17/12/03 17:42:47 INFO ShuffleBlockFetcherIterator: Getting 1 non-empty blocks out of 1 blocks
17/12/03 17:42:47 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 11 ms
17/12/03 17:42:47 INFO Executor: Finished task 0.0 in stage 66.0 (TID 66). 1394 bytes result sent to driver
17/12/03 17:42:47 INFO TaskSchedulerImpl: Removed TaskSet 66.0, whose tasks have all completed, from pool
17/12/03 17:42:47 INFO TaskSchedulerImpl: ResultStage 66 (collectAsMap at MulticlassMetrics.scala:58) finished in 0.030 s
17/12/03 17:42:47 INFO DAGScheduler: Job 63 finished: collectAsMap at MulticlassMetrics.scala:58, took 0.271372 s
Accuracy = 0.5714285714285714
Recall=0.5714285714285714
Precision=0.5714285714285714
Confusion Matrix=3.0 3.0
0.0 1.0
17/12/03 17:42:48 INFO ContextCleaner: Cleaned shuffle 2
17/12/03 17:42:48 INFO BlockManagerInfo: Removed broadcast_122_piece0 on 10.0.2.15:41229 in memory (size: 4.5 KB, free: 413.0 MB)
17/12/03 17:42:48 INFO BlockManagerInfo: Removed broadcast_123_piece0 on 10.0.2.15:41229 in memory (size: 1678.0 B, free: 413.9 MB)
17/12/03 17:42:49 INFO SparkContext: Invoking stop() from shutdown hook
17/12/03 17:42:49 INFO SparkUI: Stopped Spark web UI at http://10.0.2.15:4041
17/12/03 17:42:49 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
17/12/03 17:42:49 INFO MemoryStore: MemoryStore cleared
17/12/03 17:42:49 INFO BlockManager: BlockManager stopped
17/12/03 17:42:49 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
17/12/03 17:42:49 INFO SparkContext: Successfully stopped SparkContext
17/12/03 17:42:49 INFO ShutdownHookManager: Shutdown hook called
17/12/03 17:42:49 INFO ShutdownHookManager: Deleting directory /tmp/spark-02abed49-d32e-4a76-952d-a892ace41908
greenivas@greenivas-VirtualBox:~/IdeaProjects/svm$
```

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\kmahel\Desktop\MS Fall 17\Machine Learning\Project\output20k\convnetlatest.py - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

1 1 # -*- coding: utf-8 -*-
2 2 """
3 3 Created on Sun Nov 5 18:36:57 2017
4 4
5 5 @author: Sreenivas, Mahesh, Vignesh, Nitesh
6 6 """
7 7 import torch
8 8 from torch.autograd import Variable
9 9 import torchvision.transforms as transforms
10 10 import torch.nn as nn
11 11 import torch.nn.functional as F
12 12 import torch.optim as optim
13 13 import argparse
14 14 import imageLabel as Image
15 15 from sklearn.model_selection import train_test_split
16 16 from sklearn.metrics import confusion_matrix, f1_score
17 17 import warnings
18 18 #Training Settings
19 19 parser = argparse.ArgumentParser(description='Image classification')
20 20 parser.add_argument('--data', metavar='DIR',
21 21 help='path to image directory')
22 22 parser.add_argument('--label', metavar='FILE', help='path to label file')
23 23 parser.add_argument('--batchsize', type=int, default=64, metavar='N',
24 24 help='input batch size for training (default: 64)')
25 25 parser.add_argument('--testbatchsize', type=int, default=100, metavar='N',
26 26 help='input batch size for testing (default: 1000)')
27 27 parser.add_argument('--validbatchsize', type=int, default=64, metavar='N',
28 28 help='input batch size for validation (default: 64)')
29 29 parser.add_argument('--epochs', type=int, default=100, metavar='N',
30 30 help='number of epochs to train (default: 10)')
31 31 parser.add_argument('--lr', type=float, default=0.01, metavar='LR',
32 32 help='learning rate (default: 0.01)')
33 33 parser.add_argument('--momentum', type=float, default=0.5, metavar='M',
34 34 help='SGD momentum (default: 0.5)')
35 35 #I have made no cuda default True->change when gpu is set up
36 36 parser.add_argument('--no-cuda', action='store_true', default=False,
37 37 help='disables CUDA training')
38 38 parser.add_argument('--seed', type=int, default=1, metavar='S',
39 39 help='random seed (default: 1)')
40 40 parser.add_argument('--log-interval', type=int, default=10, metavar='N',
41 41 help='how many batches to wait before logging training status')
42 42
43 43 #Building convolutional neural network
44 44 class Net(nn.Module):
45 45 def __init__(self):
46 46 super(Net, self).__init__()
47 47 self.conv1=nn.Conv2d(3,5,5)
48 48 self.pool=nn.MaxPool2d(2)
49 49 self.conv2=nn.Conv2d(5,5,5)
50 50 self.fc1=nn.Linear(5*5*5*20)
51 51 self.fc2=nn.Linear(20,25)
52 52 self.fc3=nn.Linear(25,5)
53 53
54 54 def forward(self, x):
55 55 x=self.pool(F.relu(self.conv1(x)))
56 56 x=self.pool(F.relu(self.conv2(x)))
57 57 x=x.view(-1,5*5*5*20)
58 58 x=F.relu(self.fc1(x))
59 59 x=F.dropout(x, training=self.training)
60 60 x=F.relu(self.fc2(x))
61 61 x=F.dropout(x, training=self.training)
62 62 x=self.fc3(x)
63 63 return F.log_softmax(x)
64 64
65 65 #Transformation functions
66 66 def train_transformation(train_dataset, transform):
67 67 train_transform=[]
68 68 for value in train_dataset:
69 69 train_transform.append((transform(value[0]),value[1]))
70 70 return train_transform
71 71
72 72 def test_transformation(test_dataset, transform):
73 73 test_transform=[]
74 74 for value in test_dataset:
75 75 test_transform.append((transform(value[0]),value[1]))
76 76 return test_transform
77 77
78 78
79 79
80 80
81 81
82 82
83 83
84 84
85 85
86 86
87 87
88 88
89 89
90 90
91 91
92 92
93 93
94 94
95 95
96 96
97 97
98 98
99 99
100 100
101 101
102 102
103 103
104 104
105 105
106 106
107 107
108 108
109 109
110 110
111 111
112 112
113 113
114 114
115 115
116 116
117 117
118 118
119 119
120 120
121 121
122 122
123 123
124 124
125 125
126 126
127 127
128 128
129 129
130 130
131 131
132 132
133 133
134 134
135 135
136 136
137 137
138 138
139 139
140 140
141 141
142 142
143 143
144 144
145 145
146 146
147 147
148 148
149 149
150 150
151 151
152 152
153 153
154 154
155 155
156 156
157 157
158 158
159 159
160 160
161 161
162 162
163 163
164 164
165 165
166 166
167 167
168 168
169 169
170 170
171 171
172 172
173 173
174 174
175 175
176 176
177 177
178 178
179 179
180 180
181 181
182 182
183 183
184 184
185 185
186 186
187 187
188 188
189 189
190 190
191 191
192 192
193 193
194 194
195 195
196 196
197 197
198 198
199 199
200 200
201 201
202 202
203 203
204 204
205 205
206 206
207 207
208 208
209 209
210 210
211 211
212 212
213 213
214 214
215 215
216 216
217 217
218 218
219 219
220 220
221 221
222 222
223 223
224 224
225 225
226 226
227 227
228 228
229 229
230 230
231 231
232 232
233 233
234 234
235 235
236 236
237 237
238 238
239 239
240 240
241 241
242 242
243 243
244 244
245 245
246 246
247 247
248 248
249 249
250 250
251 251
252 252
253 253
254 254
255 255
256 256
257 257
258 258
259 259
260 260
261 261
262 262
263 263
264 264
265 265
266 266
267 267
268 268
269 269
270 270
271 271
272 272
273 273
274 274
275 275
276 276
277 277
278 278
279 279
280 280
281 281
282 282
283 283
284 284
285 285
286 286
287 287
288 288
289 289
290 290
291 291
292 292
293 293
294 294
295 295
296 296
297 297
298 298
299 299
300 300
301 301
302 302
303 303
304 304
305 305
306 306
307 307
308 308
309 309
310 310
311 311
312 312
313 313
314 314
315 315
316 316
317 317
318 318
319 319
320 320
321 321
322 322
323 323
324 324
325 325
326 326
327 327
328 328
329 329
330 330
331 331
332 332
333 333
334 334
335 335
336 336
337 337
338 338
339 339
340 340
341 341
342 342
343 343
344 344
345 345
346 346
347 347
348 348
349 349
350 350
351 351
352 352
353 353
354 354
355 355
356 356
357 357
358 358
359 359
360 360
361 361
362 362
363 363
364 364
365 365
366 366
367 367
368 368
369 369
370 370
371 371
372 372
373 373
374 374
375 375
376 376
377 377
378 378
379 379
380 380
381 381
382 382
383 383
384 384
385 385
386 386
387 387
388 388
389 389
390 390
391 391
392 392
393 393
394 394
395 395
396 396
397 397
398 398
399 399
400 400
401 401
402 402
403 403
404 404
405 405
406 406
407 407
408 408
409 409
410 410
411 411
412 412
413 413
414 414
415 415
416 416
417 417
418 418
419 419
420 420
421 421
422 422
423 423
424 424
425 425
426 426
427 427
428 428
429 429
430 430
431 431
432 432
433 433
434 434
435 435
436 436
437 437
438 438
439 439
440 440
441 441
442 442
443 443
444 444
445 445
446 446
447 447
448 448
449 449
450 450
451 451
452 452
453 453
454 454
455 455
456 456
457 457
458 458
459 459
460 460
461 461
462 462
463 463
464 464
465 465
466 466
467 467
468 468
469 469
470 470
471 471
472 472
473 473
474 474
475 475
476 476
477 477
478 478
479 479
480 480
481 481
482 482
483 483
484 484
485 485
486 486
487 487
488 488
489 489
490 490
491 491
492 492
493 493
494 494
495 495
496 496
497 497
498 498
499 499
500 500
501 501
502 502
503 503
504 504
505 505
506 506
507 507
508 508
509 509
510 510
511 511
512 512
513 513
514 514
515 515
516 516
517 517
518 518
519 519
520 520
521 521
522 522
523 523
524 524
525 525
526 526
527 527
528 528
529 529
530 530
531 531
532 532
533 533
534 534
535 535
536 536
537 537
538 538
539 539
540 540
541 541
542 542
543 543
544 544
545 545
546 546
547 547
548 548
549 549
550 550
551 551
552 552
553 553
554 554
555 555
556 556
557 557
558 558
559 559
560 560
561 561
562 562
563 563
564 564
565 565
566 566
567 567
568 568
569 569
570 570
571 571
572 572
573 573
574 574
575 575
576 576
577 577
578 578
579 579
580 580
581 581
582 582
583 583
584 584
585 585
586 586
587 587
588 588
589 589
590 590
591 591
592 592
593 593
594 594
595 595
596 596
597 597
598 598
599 599
600 600
601 601
602 602
603 603
604 604
605 605
606 606
607 607
608 608
609 609
610 610
611 611
612 612
613 613
614 614
615 615
616 616
617 617
618 618
619 619
620 620
621 621
622 622
623 623
624 624
625 625
626 626
627 627
628 628
629 629
630 630
631 631
632 632
633 633
634 634
635 635
636 636
637 637
638 638
639 639
640 640
641 641
642 642
643 643
644 644
645 645
646 646
647 647
648 648
649 649
650 650
651 651
652 652
653 653
654 654
655 655
656 656
657 657
658 658
659 659
660 660
661 661
662 662
663 663
664 664
665 665
666 666
667 667
668 668
669 669
670 670
671 671
672 672
673 673
674 674
675 675
676 676
677 677
678 678
679 679
680 680
681 681
682 682
683 683
684 684
685 685
686 686
687 687
688 688
689 689
690 690
691 691
692 692
693 693
694 694
695 695
696 696
697 697
698 698
699 699
700 700
701 701
702 702
703 703
704 704
705 705
706 706
707 707
708 708
709 709
710 710
711 711
712 712
713 713
714 714
715 715
716 716
717 717
718 718
719 719
720 720
721 721
722 722
723 723
724 724
725 725
726 726
727 727
728 728
729 729
730 730
731 731
732 732
733 733
734 734
735 735
736 736
737 737
738 738
739 739
740 740
741 741
742 742
743 743
744 744
745 745
746 746
747 747
748 748
749 749
750 750
751 751
752 752
753 753
754 754
755 755
756 756
757 757
758 758
759 759
760 760
761 761
762 762
763 763
764 764
765 765
766 766
767 767
768 768
769 769
770 770
771 771
772 772
773 773
774 774
775 775
776 776
777 777
778 778
779 779
780 780
781 781
782 782
783 783
784 784
785 785
786 786
787 787
788 788
789 789
790 790
791 791
792 792
793 793
794 794
795 795
796 796
797 797
798 798
799 799
800 800
801 801
802 802
803 803
804 804
805 805
806 806
807 807
808 808
809 809
810 810
811 811
812 812
813 813
814 814
815 815
816 816
817 817
818 818
819 819
820 820
821 821
822 822
823 823
824 824
825 825
826 826
827 827
828 828
829 829
830 830
831 831
832 832
833 833
834 834
835 835
836 836
837 837
838 838
839 839
840 840
841 841
842 842
843 843
844 844
845 845
846 846
847 847
848 848
849 849
850 850
851 851
852 852
853 853
854 854
855 855
856 856
857 857
858 858
859 859
860 860
861 861
862 862
863 863
864 864
865 865
866 866
867 867
868 868
869 869
870 870
871 871
872 872
873 873
874 874
875 875
876 876
877 877
878 878
879 879
880 880
881 881
882 882
883 883
884 884
885 885
886 886
887 887
888 888
889 889
890 890
891 891
892 892
893 893
894 894
895 895
896 896
897 897
898 898
899 899
900 900
901 901
902 902
903 903
904 904
905 905
906 906
907 907
908 908
909 909
910 910
911 911
912 912
913 913
914 914
915 915
916 916
917 917
918 918
919 919
920 920
921 921
922 922
923 923
924 924
925 925
926 926
927 927
928 928
929 929
930 930
931 931
932 932
933 933
934 934
935 935
936 936
937 937
938 938
939 939
940 940
941 941
942 942
943 943
944 944
945 945
946 946
947 947
948 948
949 949
950 950
951 951
952 952
953 953
954 954
955 955
956 956
957 957
958 958
959 959
960 960
961 961
962 962
963 963
964 964
965 965
966 966
967 967
968 968
969 969
970 970
971 971
972 972
973 973
974 974
975 975
976 976
977 977
978 978
979 979
980 980
981 981
982 982
983 983
984 984
985 985
986 986
987 987
988 988
989 989
990 990
991 991
992 992
993 993
994 994
995 995
996 996
997 997
998 998
999 999
1000 1000
1001 1001
1002 1002
1003 1003
1004 1004
1005 1005
1006 1006
1007 1007
1008 1008
1009 1009
1010 1010
1011 1011
1012 1012
1013 1013
1014 1014
1015 1015
1016 1016
1017 1017
1018 1018
1019 1019
1020 1020
1021 1021
1022 1022
1023 1023
1024 1024
1025 1025
1026 1026
1027 1027
1028 1028
1029 1029
1030 1030
1031 1031
1032 1032
1033 1033
1034 1034
1035 1035
1036 1036
1037 1037
1038 1038
1039 1039
1040 1040
1041 1041
1042 1042
1043 1043
1044 1044
1045 1045
1046 1046
1047 1047
1048 1048
1049 1049
1050 1050
1051 1051
1052 1052
1053 1053
1054 1054
1055 1055
1056 1056
1057 1057
1058 1058
1059 1059
1060 1060
1061 1061
1062 1062
1063 1063
1064 1064
1065 1065
1066 1066
1067 1067
1068 1068
1069 1069
1070 1070
1071 1071
1072 1072
1073 1073
1074 1074
1075 1075
1076 1076
1077 1077
1078 1078
1079 1079
1080 1080
1081 1081
1082 1082
1083 1083
1084 1084
1085 1085
1086 1086
1087 1087
1088 1088
1089 1089
1090 1090
1091 1091
1092 1092
1093 1093
1094 1094
1095 1095
1096 1096
1097 1097
1098 1098
1099 1099
1100 1100
1101 1101
1102 1102
1103 1103
1104 1104
1105 1105
1106 1106
1107 1107
1108 1108
1109 1109
1110 1110
1111 1111
1112 1112
1113 1113
1114 1114
1115 1115
1116 1116
1117 1117
1118 1118
1119 1119
1120 1120
1121 1121
1122 1122
1123 1123
1124 1124
1125 1125
1126 1126
1127 1127
1128 1128
1129 1129
1130 1130
1131 1131
1132 1132
1133 1133
1134 1134
1135 1135
1136 1136
1137 1137
1138 1138
1139 1139
1140 1140
1141 1141
1142 1142
1143 1143
1144 1144
1145 1145
1146 1146
1147 1147
1148 1148
1149 1149
1150 1150
1151 1151
1152 1152
1153 1153
1154 1154
1155 1155
1156 1156
1157 1157
1158 1158
1159 1159
1160 1160
1161 1161
1162 1162
1163 1163
1164 1164
1165 1165
1166 1166
1167 1167
1168 1168
1169 1169
1170 1170
1171 1171
1172 1172
1173 1173
1174 1174
1175 1175
1176 1176
1177 1177
1178 1178
1179 1179
1180 1180
1181 1181
1182 1182
1183 1183
1184 1184
1185 1185
1186 1186
1187 1187
1188 1188
1189 1189
1190 1190
1191 1191
1192 1192
1193 1193
1194 1194
1195 1195
1196 1196
1197 1197
1198 1198
1199 1199
1200 1200
1201 1201
1202 1202
1203 1203
1204 1204
1205 1205
1206 1206
1207 1207
1208 1208
1209 1209
1210 1210
1211 1211
1212 1212
1213 1213
1214 1214
1215 1215
1216 1216
1217 1217
1218 1218
1219 1219
1220 1220
1221 1221
1222 1222
1223 1223
1224 1224
1225 1225
1226 1226
1227 1227
1228 1228
1229 1229
1230 1230
1231 1231
1232 1232
1233 1233
1234 1234
1235 1235
1236 1236
1237 1237
1238 1238
1239 1239
1240 1240
1241 1241
1242 1242
1243 1243
1244 1244
1245 1245
1246 1246
1247 1247
1248 1248
1249 1249
1250 1250
1251 1251
1252 1252
1253 1253
1254 1254
1255 1255
1256 1256
1257 1257
1258 1258
1259 1259
1260 1260
1261 1261
1262 1262
1263 1263
1264 1264
1265 1265
1266 1266
1267 1267
1268 1268
1269 1269
1270 1270
1271 1271
1272 1272
1273 1273
1274 1274
1275 1275
1276 1276
1277 1277
1278 1278
1279 1279
1280 1280
1281 1281
1282 1282
1283 1283
1284 1284
1285 1285
1286 1286
1287 1287
1288 1288
1289 1289
1290 1290
1291 1291
1292 1292
1293 1293
1294 1294
1295 1295
1296 1296
1297 1297
1298 1298
1299 1299
1300 1300
1301 1301
1302 1302
1303 1303
1304 1304
1305 1305
1306 1306
1307 1307
1308 1308
1309 1309
1310 1310
1311 1311
1312 1312
1313 1313
1314 1314
1315 1315
1316 1316
1317 1317
1318 1318
1319 1319
1320 1320
1321 1321
1322 1322
1323 1323
1324 1324
1325 1325
1326 1326
1327 1327
1328 1328
1329 1329
1330 1330
1331 1331
1332 1332
1333 1333
1334 1334
1335 1335
1336 1336
1337 1337
1338 1338
1339 1339
1340 1340
1341 1341
1342 1342
1343 1343
1344 1344
1345 1345
1346 1346
1347 1347
1348 1348
1349 1349
1350 1350
1351 1351
1352 1352
1353 1353
1354 1354
1355 1355
1356 1356
1357 1357
1358 1358
1359 1359
1360 1360
1361 1361
1362 1362
1363 1363
1364 1364
1365 1365
1366 1366
1367 1367
1368 1368
1369 1369
1370 1370
1371 1371
1372 1372
1373 1373
1374 1374
1375 1375
1376 1
```

```
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 ?

61     x=self.fc3(x)
62     return F.log_softmax(x)
63 #Transformation functions
64 def train_transformation(train_dataset,transform):
65     train_transform=[]
66     for value in train_dataset:
67         train_transform.append((transform(value[0]),value[1]))
68     return train_transform
69 def test_transformation(test_dataset,transform):
70     test_transform=[]
71     for value in test_dataset:
72         test_transform.append((transform(value[0]),value[1]))
73     return test_transform
74 def validation_transformation(test_dataset,transform):
75     valid_transform=[]
76     for value in test_dataset:
77         valid_transform.append((transform(value[0]),value[1]))
78     return valid_transform
79 #Train function
80 def train(model,train_loader,optimizer,epoch):
81     model.train()
82     for batch_idx,(data,target) in enumerate(train_loader):
83         if args.cuda:
84             data,target=data.cuda(),target.cuda()
85             data,target=Variable(data),Variable(target)
86             optimizer.zero_grad()
87             output=model(data)
88             loss=F.nll_loss(output,target.long())
89             loss.backward()
90             optimizer.step()
91         if batch_idx % args.log_interval == 0:
92             print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
93                 epoch, batch_idx * len(data), len(train_loader.dataset),
94                 100. * batch_idx / len(train_loader), loss.data[0]))
95 #Test function
96 def test(model,test_loader):
```

```
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 ?

93     epoch, batch_idx * len(data), len(train_loader.dataset),
94     100. * batch_idx / len(train_loader), loss.data[0]))
95 #Test function
96 def test(model,test_loader):
97     model.eval()
98     correct=0
99     counter=0
100     for batch_idx,(data,target) in enumerate(test_loader):
101         if args.cuda:
102             data,target=data.cuda(),target.cuda()
103             output=model(Variable(data))
104             _,predicted=torch.max(output.data,1)
105             if counter==0:
106                 trueList=target.long()
107                 predictedList=predicted.view(len(predicted))
108             else:
109                 trueList=torch.cat((trueList,target.long()),0)
110                 predictedList=torch.cat((predictedList,predicted.view(len(predicted))),0)
111                 correct+=(predicted==target.long().view(-1,1)).cpu().sum()
112                 counter+=1
113     accuracy=100*(correct/len(test_loader.dataset))
114     return accuracy,trueList,predictedList
115 #Converting categorical variables to original labels
116 def val_labels(valueList,labels):
117     convList=[]
118     for value in valueList:
119         convList.append(labels[value])
120     return convList
121 #Main function
122 if __name__=="__main__":
123     args = parser.parse_args()
124     args.cuda = not args.no_cuda and torch.cuda.is_available()
125     torch.manual_seed(args.seed)
126     if args.cuda:
127         torch.cuda.manual_seed(args.seed)
128     model=Net()
```

```
C:\Users\kmahel\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.bt photos.json op.bt convnetlatest.py

113 accuracy=100*(correct/len(test_loader.dataset))
114 return accuracy,trueList,predictedList
115 #Converting categorical variables to original labels
116 def valLabels(valueList,labels):
117     convList=[]
118     for value in valueList:
119         convList.append(labels[value])
120     return convList
121 #Main function
122 if __name__=="__main__":
123     args = parser.parse_args()
124     args.cuda = not args.no_cuda and torch.cuda.is_available()
125     torch.manual_seed(args.seed)
126     if args.cuda:
127         torch.cuda.manual_seed(args.seed)
128     model=Net()
129     if args.cuda:
130         model.cuda()
131     #optimizer and criterion for neural network
132     optimizer=optim.SGD(model.parameters(),lr=args.lr,momentum=args.momentum)
133     #normalizing data
134     normalize=transforms.Normalize(mean=[0.485, 0.456, 0.406],
135                                   std=[0.229, 0.224, 0.225])
136     train_transform=transforms.Compose([transforms.RandomSizedCrop(224),
137                                       transforms.RandomHorizontalFlip(),
138                                       transforms.ToTensor(),
139                                       normalize])
140     test_transform=transforms.Compose([transforms.RandomSizedCrop(224),
141                                       transforms.ToTensor(),
142                                       normalize])
143     validation_transform=transforms.Compose([transforms.RandomSizedCrop(224),
144                                             transforms.ToTensor(),
145                                             normalize])
146     #taking directory with image folder from command line
147     imagefile=args.data
148     labelfile=args.label

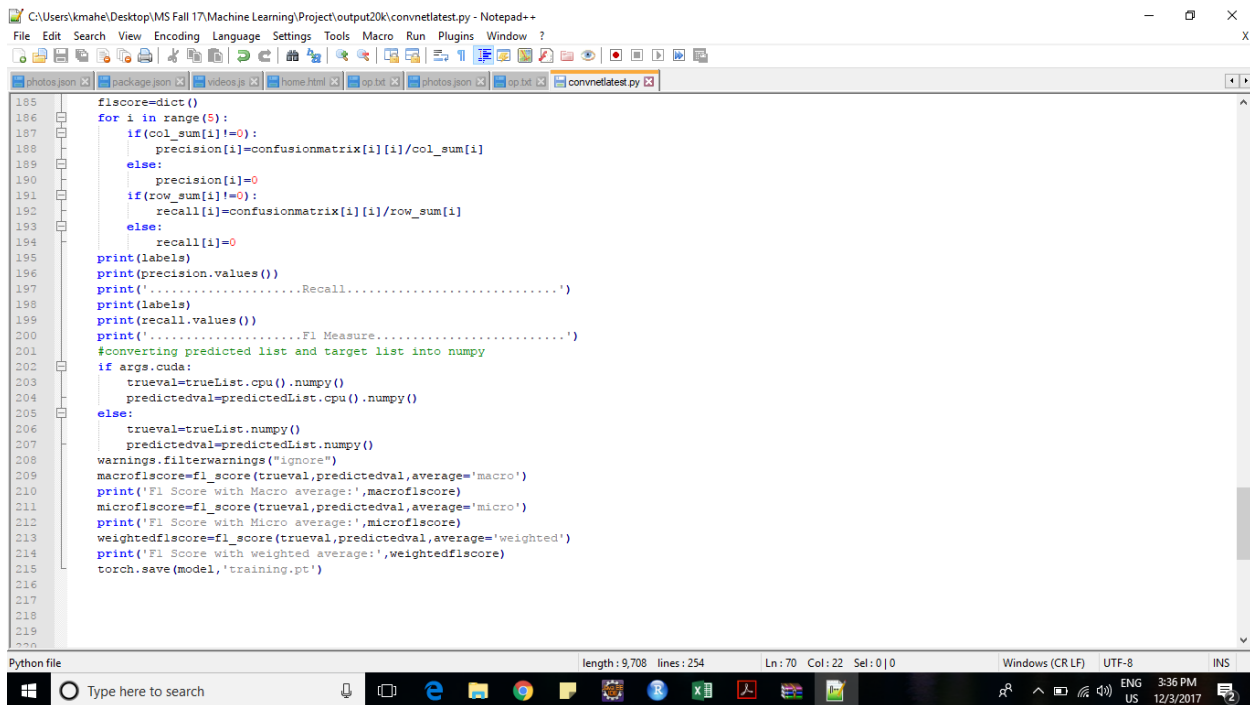
Python file length: 9,708 lines: 254 Ln: 70 Col: 22 Sel: 0 | 0 Windows (CR LF) UTF-8 INS
```

```
C:\Users\kmahel\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.bt photos.json op.bt convnetlatest.py

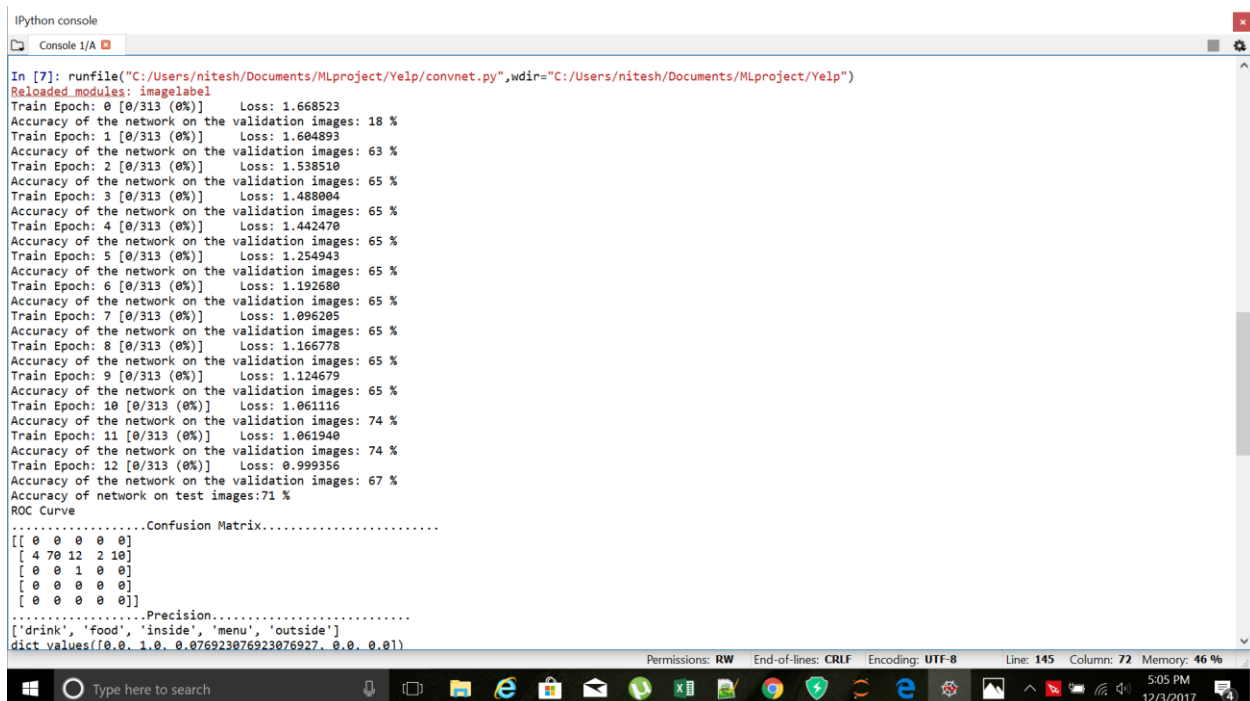
137     transforms.RandomHorizontalFlip(),
138     transforms.ToTensor(),
139     normalize])
140     test_transform=transforms.Compose([transforms.RandomSizedCrop(224),
141                                       transforms.ToTensor(),
142                                       normalize])
143     validation_transform=transforms.Compose([transforms.RandomSizedCrop(224),
144                                             transforms.ToTensor(),
145                                             normalize])
146     #taking directory with image folder from command line
147     imagefile=args.data
148     labelfile=args.label
149     #calling class from imageloading
150     imagedataset=Image.ImageDataset(imagefile,labelfile)
151     #splitting datasets into train,test and validation datasets
152     trainvaliddataset,testdataset=train_test_split(imagedataset,test_size=0.2,random_state=0)
153     traindataset,validdataset=train_test_split(trainvaliddataset,test_size=0.2,random_state=0)
154     #Transformation of image into tensor
155     traindataset=train_transformation(traindataset,train_transform)
156     testdataset=test_transformation(testdataset,test_transform)
157     validdataset=validation_transformation(validdataset,validation_transform)
158     #dataloader for train test and validation
159     train_loader=torch.utils.data.DataLoader(traindataset,batch_size=args.batchsize)
160     test_loader=torch.utils.data.DataLoader(testdataset,batch_size=args.testbatchsize)
161     validation_loader=torch.utils.data.DataLoader(validdataset,batch_size=args.validbatchsize)
162     #Training phase
163     prev_accuracy=0
164     for epoch in range(0,args.epochs):
165         train(model,train_loader,optimizer,epoch)
166         accuracy,predictedList,trueList=test(model,validation_loader)
167         print('Accuracy of the network on the validation images: %d %%' % (accuracy))
168         if (prev_accuracy>accuracy):break
169         #prev_accuracy=accuracy
170     accuracy,predictedList,trueList=test(model,test_loader)
171     print('Accuracy of network on test images:%d %%' % (accuracy))
172     print('BCC Curve')

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



```
185 f1score=dict()
186 for i in range(5):
187     if(col_sum[i]!=0):
188         precision[i]=confusionmatrix[i][i]/col_sum[i]
189     else:
190         precision[i]=0
191     if(row_sum[i]!=0):
192         recall[i]=confusionmatrix[i][i]/row_sum[i]
193     else:
194         recall[i]=0
195 print(labels)
196 print(precision.values())
197 print('.....Recall.....')
198 print(labels)
199 print(recall.values())
200 print('.....F1 Measure.....')
201 #converting predicted list and target list into numpy
202 if args.cuda:
203     trueval=trueList.cpu().numpy()
204     predictedval=predictedList.cpu().numpy()
205 else:
206     trueval=trueList.numpy()
207     predictedval=predictedList.numpy()
208 warnings.filterwarnings("ignore")
209 macrof1score=f1_score(trueval,predictedval,average='macro')
210 print('F1 Score with Macro average:',macrof1score)
211 microf1score=f1_score(trueval,predictedval,average='micro')
212 print('F1 Score with Micro average:',microf1score)
213 weightedf1score=f1_score(trueval,predictedval,average='weighted')
214 print('F1 Score with Weighted average:',weightedf1score)
215 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, 0.0, 0.0])
```

Fig 5:Output for 500 images (I)



```
IPython console
Console 1/A
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]
 [ 4 70 12  2]
 [ 0  0  1  0]
 [ 0  0  0  0]]
.....Precision.....
['drink', 'food', 'inside', 'menu', 'outside']
dict_values([0.0, 1.0, 0.07692307692307692, 0.0, 0.0])
.....Recall.....
['drink', 'food', 'inside', 'menu', 'outside']
dict_values([0.0, 0.7142857142857143, 1.0, 0.0, 0.0])
.....F1 Measure.....
F1 Score with Macro average: 0.195238095238
F1 Score with Micro average: 0.717171717172
F1 Score with weighted average: 0.826358826359
In [8]: |
```

Fig 6: Output for 500 images (II)

```
ubuntu@ip-172-31-20-54: ~/src
Train Epoch: 96 [0/6301 (0%)] Loss: 0.290199
Train Epoch: 96 [640/6301 (10%)] Loss: 0.175536
Train Epoch: 96 [1280/6301 (20%)] Loss: 0.111908
Train Epoch: 96 [1920/6301 (30%)] Loss: 0.228598
Train Epoch: 96 [2560/6301 (40%)] Loss: 0.115675
Train Epoch: 96 [3200/6301 (51%)] Loss: 0.109480
Train Epoch: 96 [3840/6301 (61%)] Loss: 0.170252
Train Epoch: 96 [4480/6301 (71%)] Loss: 0.133502
Train Epoch: 96 [5120/6301 (81%)] Loss: 0.059193
Train Epoch: 96 [5760/6301 (91%)] Loss: 0.156924
Accuracy of the network on the validation images: 77 %
Train Epoch: 97 [0/6301 (0%)] Loss: 0.165999
Train Epoch: 97 [640/6301 (10%)] Loss: 0.093215
Train Epoch: 97 [1280/6301 (20%)] Loss: 0.104764
Train Epoch: 97 [1920/6301 (30%)] Loss: 0.108464
Train Epoch: 97 [2560/6301 (40%)] Loss: 0.104767
Train Epoch: 97 [3200/6301 (51%)] Loss: 0.172959
Train Epoch: 97 [3840/6301 (61%)] Loss: 0.150254
Train Epoch: 97 [4480/6301 (71%)] Loss: 0.103374
Train Epoch: 97 [5120/6301 (81%)] Loss: 0.126681
Train Epoch: 97 [5760/6301 (91%)] Loss: 0.092013
Accuracy of the network on the validation images: 77 %
Train Epoch: 98 [0/6301 (0%)] Loss: 0.279906
Train Epoch: 98 [640/6301 (10%)] Loss: 0.123843
Train Epoch: 98 [1280/6301 (20%)] Loss: 0.113763
Train Epoch: 98 [1920/6301 (30%)] Loss: 0.100034
Train Epoch: 98 [2560/6301 (40%)] Loss: 0.137873
Train Epoch: 98 [3200/6301 (51%)] Loss: 0.224755
Train Epoch: 98 [3840/6301 (61%)] Loss: 0.109272
Train Epoch: 98 [4480/6301 (71%)] Loss: 0.154779
Train Epoch: 98 [5120/6301 (81%)] Loss: 0.163631
Train Epoch: 98 [5760/6301 (91%)] Loss: 0.215262
Accuracy of the network on the validation images: 77 %
Train Epoch: 99 [0/6301 (0%)] Loss: 0.154882
Train Epoch: 99 [640/6301 (10%)] Loss: 0.133999
Train Epoch: 99 [1280/6301 (20%)] Loss: 0.078321
Train Epoch: 99 [1920/6301 (30%)] Loss: 0.203860
Train Epoch: 99 [2560/6301 (40%)] Loss: 0.181202
Train Epoch: 99 [3200/6301 (51%)] Loss: 0.240138
Train Epoch: 99 [3840/6301 (61%)] Loss: 0.088844
Train Epoch: 99 [4480/6301 (71%)] Loss: 0.157750
Train Epoch: 99 [5120/6301 (81%)] Loss: 0.091502
Train Epoch: 99 [5760/6301 (91%)] Loss: 0.186038
Accuracy of the network on the validation images: 76 %
ROC Curve
.....Confusion Matrix.....
[[ 1  0  0  0]
 [ 46 1122  85  3]
 [ 29  67 298  5]
 [ 0  0  0  0]]
.....Precision.....
['drink', 'food', 'inside', 'menu', 'outside']
dict_values([0.012048192771084338, 0.93577981651376152, 0.61954261954261958, 0.0, 0.36224489795918369])
.....Recall.....
['drink', 'food', 'inside', 'menu', 'outside']
dict_values([0.5, 0.88, 0.59009900990099007, 0.0, 0.37765957446808512])
.....F1 Measure.....
F1 Score with Macro average: 0.38096339556
F1 Score with Micro average: 0.757360406091
F1 Score with weighted average: 0.777393909315
```

Fig 7: Output for 20k images (I)

```

ubuntu@ip-172-31-20-54: ~/src
Train Epoch: 2 [3200/6301 (51%)] Loss: 0.952698
Train Epoch: 2 [3600/6301 (57%)] Loss: 0.644115
Train Epoch: 2 [4000/6301 (63%)] Loss: 0.882285
Train Epoch: 2 [4400/6301 (70%)] Loss: 0.873279
Train Epoch: 2 [5120/6301 (81%)] Loss: 0.878769
Accuracy of the network on the validation images: 64 %
Accuracy of network on test images:64 %
ROC Curve
.....Confusion Matrix.....
[[ 0 0 0 0 0]
 [78 1192 487 10 129]
 [ 5 7 74 1 67]
 [ 0 0 0 0 0]
 [ 0 0 0 0 0]]
.....Precision.....
[ 'drink', 'food', 'inside', 'menu', 'outside']
dict_values([0.0, 0.994161801501251, 0.15384615384615385, 0.0, 0.0])
.....Recall.....
[ 'drink', 'food', 'inside', 'menu', 'outside']
dict_values([0.0, 0.65638766519823788, 0.48051948051948051, 0.0, 0.0])
.....F1 Measure.....
F1 score with Macro average: 0.204756793401
F1 score with Micro average: 0.642639593999
F1 score with weighted average: 0.747126764009
(pytorch_p36) ubuntu@ip-172-31-20-54:~/src$ python convnetlatest.py --data photo20k --label out20k.csv
Train Epoch: 0 [0/6301 (0%)] Loss: 1.651749
Train Epoch: 0 [640/6301 (10%)] Loss: 1.523007
Train Epoch: 0 [1280/6301 (20%)] Loss: 1.403224
Train Epoch: 0 [1920/6301 (30%)] Loss: 1.204563
Train Epoch: 0 [2560/6301 (40%)] Loss: 0.989767
Train Epoch: 0 [3200/6301 (51%)] Loss: 1.266196
Train Epoch: 0 [3840/6301 (61%)] Loss: 0.924181
Train Epoch: 0 [4480/6301 (71%)] Loss: 1.083312
Train Epoch: 0 [5120/6301 (81%)] Loss: 1.143484
Train Epoch: 0 [5760/6301 (91%)] Loss: 1.084876
Accuracy of the network on the validation images: 63 %
Train Epoch: 1 [0/6301 (0%)] Loss: 0.968926
Train Epoch: 1 [640/6301 (10%)] Loss: 0.801624
Train Epoch: 1 [1280/6301 (20%)] Loss: 0.993395
Train Epoch: 1 [1920/6301 (30%)] Loss: 1.054983
Train Epoch: 1 [2560/6301 (40%)] Loss: 0.836995
Train Epoch: 1 [3200/6301 (51%)] Loss: 1.151276
Train Epoch: 1 [3840/6301 (61%)] Loss: 0.796361
Train Epoch: 1 [4480/6301 (71%)] Loss: 0.884788
Train Epoch: 1 [5120/6301 (81%)] Loss: 0.967362
Train Epoch: 1 [5760/6301 (91%)] Loss: 1.007859
Accuracy of the network on the validation images: 60 %
Train Epoch: 2 [0/6301 (0%)] Loss: 1.044227
Train Epoch: 2 [640/6301 (10%)] Loss: 0.783783
Train Epoch: 2 [1280/6301 (20%)] Loss: 1.038384
Train Epoch: 2 [1920/6301 (30%)] Loss: 1.044339
Train Epoch: 2 [2560/6301 (40%)] Loss: 0.881968
Train Epoch: 2 [3200/6301 (51%)] Loss: 0.904651
Train Epoch: 2 [3840/6301 (61%)] Loss: 0.793937
Train Epoch: 2 [4480/6301 (71%)] Loss: 0.861324
Train Epoch: 2 [5120/6301 (81%)] Loss: 0.983313
Train Epoch: 2 [5760/6301 (91%)] Loss: 0.801706
Accuracy of the network on the validation images: 65 %
Train Epoch: 3 [0/6301 (0%)] Loss: 1.004654
Train Epoch: 3 [640/6301 (10%)] Loss: 0.735761
Train Epoch: 3 [1280/6301 (20%)] Loss: 0.838205
Train Epoch: 3 [1920/6301 (30%)] Loss: 0.921688

```

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

	<b>46,1112,85,3,19</b> <b>29,67,298,5,67</b> <b>0,0,0,0,0</b> <b>7,98,3,71,7)</b>
--	--

### **SVM Classifier (Tested for 50 images)**

<b><u>Metrics</u></b>	<b><u>Value</u></b>
<b>Accuracy</b>	<b>57.14%</b>
<b>Precision</b>	<b>0.571</b>
<b>Recall</b>	<b>0.571</b>
<b>Confusion Matrix</b>	<b>(3.0,3.0 0.0,1.0)</b>

- 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/>