Image Based Classification Of Waste Material

A Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Bachelor of Technology
in
Information Technology

by

Registration No.	**Name**
20158070	Sahildeep Singh Raina
20158094	Sarvaigari Govardhini
20158014	Ayush Jain
20158098	Gourav Kumar
20158008	Tejaswini Sundar

to the
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
MOTILAL NEHRU NATIONAL INSTITUTE OF TECHNOLOGY
ALLAHABAD
April, 2018

UNDERTAKING

We declare that the work presented in this report titled "*Image Based Classification Of Waste Material*", submitted to the Computer Science and Engineering Department, Motilal Nehru National Institute of Technology, Allahabad, for the award of the ***Bachelor of Technology*** degree in ***Information Technology***, is our original work. We have not plagiarized or submitted the same work for the award of any other degree. In case this undertaking is found incorrect, we accept that our degree may be unconditionally withdrawn.

April, 2018
Allahabad

Sahildeep Singh Raina,
Sarvaigari Govardhini,
Ayush Jain,
Gourav Kumar,
Tejaswini Sundar

CERTIFICATE

Certified that the work contained in the report titled "*Image Based Classification Of Waste Material*", by *Sahildeep Singh Raina, Sarvaigari Govardhini, Ayush Jain, Gourav Kumar and Tejaswini Sundar*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

(Dr Divya Kumar)
Computer Science and Engineering Dept.
M.N.N.I.T, Allahabad

April, 2018

Preface

A computer vision approach to classifying garbage into usage classes may well be an economical way of dealing with waste. The aim of this project is to process pictures of one piece of garbage and classify it into 5 categories consisting of glass, paper, metal, plastic, and cardboard. We have additionally produced a dataset that contains around 400-500 pictures for every category, that was each hand collected and gathered from online sources [2]. A convolutional neural network is employed during this project, created on Google's open source code library for dataflow programming - Tensorflow[4]. This project additionally uses Transfer Learning to use already existing predicting models of image classification and rewire them for finishing the task at hand. This paper provides an abstract view of our approach for garbage classification using the aforesaid models and algorithms.

Acknowledgements

We would like to take this opportunity to express our deep sense of gratitude to all who helped us directly or indirectly for our thesis work. First, we would like to thank our supervisor, Dr Divya Kumar, for being the best mentor we could ever have. His advise, encouragement and critics are a source of innovative ideas, inspiration and the cause behind the successful completion of this dissertation. The confidence shown on us by him was the biggest source of inspiration for us. His perpetual energy and enthusiasm in research had motivated us. In addition, he was always accessible and willing to help his student with their research. As a result, research life became smooth and rewarding for us. It has been a privilege working with him for last six months. He gave us ample opportunities to explore myself.

We wish to express our sincere gratitude to Prof. Rajeev Tripathi, Director, MNNIT Allahabad, Allahabad and Prof. Neeraj Tyagi, Head, Computer Science and Engineering Department, for providing us all the facilities required for the completion of this thesis work. We would also like thank our friends for their constant motivation, advice and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The ever growing amount of waste in a country with a population of more than 1.3 billion calls for some action. The collection of waste generated by the rapidly expanding cities of developing countries is increasingly beyond the capacity and financial means of the municipal administrations. The current recycling process requires recycling facilities to sort garbage by hand and use a series of large filters to separate out more defined objects. Consumers also can be confused about how to determine the correct way to dispose of a large variety of materials used in packaging. In the era where resources are depleting fast, it is our top priority to improve the existing methods of recycling to ensure sustainable development.

1.1 Motivation

1.1.1 Some Wonderful Minds

The first artificial neural network was invented in 1958 by psychologist Frank Rosenblatt. Called Perceptron, it was intended to model how the human brain processed visual data and learned to recognize objects. Other researchers have since used similar ANNs to study human cognition. ALEXNET[5], the one that started it all (though some may say that Yann LeCuns paper in 1998 was the real pioneering publication). This paper, titled ImageNet Classification with Deep Convolutional Networks, has been cited a total of 6,184 times and is widely regarded as one of the most influential publications in the field. Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton created a large, deep convolutional neural network that was used to win the 2012 ILSVRC (ImageNet Large-Scale Visual Recognition Challenge). The neural network developed by Krizhevsky, Sutskever, and Hinton in 2012 was the coming out party for CNNs in the computer vision community. This was the first time a model performed so well on a historically difficult ImageNet dataset. Utilizing techniques that are still used today, such as data augmentation and dropout, this paper really illustrated the benefits of CNNs and backed them up with record breaking performance in the competition.

1.1.2 Catering the needs of the nation

Another push to take this project was given by the current system of garbage handling in India. Being an overpopulated country, the amount of garbage produced is extraordinary, but the current system fails to collect and dispose/recycle it efficiently.

If recycled properly, India can unlock the hidden potential of garbage as a resource. But that requires efficient garbage sorting. Also, due to poor sense of awareness or lack of education thereof, not everyone is capable of "putting garbage in the right bin". Hence the need of automated garbage classification.

1.1.3 Need of the hour

Waste poses a threat to public health and the environment if it is not stored, collected, and disposed off properly. The perception of waste as an unwanted material with no intrinsic value has dominated attitudes towards disposal. India needs to solve its waste management problem. Be it solid urban waste, sewage or chemical and industrial waste, in India, every waste is mismanaged. So, instead of constructing landfill sites, given existing ones are operating way past their lifespan, experts say that the government should look at innovative methods to recycle waste. Despite the huge strides India is making globally in several spheres, one area that has been completely ignored for long is cleanliness. The need for segregating waste materials according to the desired properties is constantly increasing. As the rate of generation of waste products is high, pure man power cannot do much to counter this process. Therefore, teaching a machine the difference between such objects would not only be a faster method, but also efficient. An amalgamation of technology and motive would give effective results.

1.2 What is classification?

As the name suggests, classification is a general process related to categorization, the process in which ideas and objects are recognized, differentiated, and understood. Classification means differentiating objects, labelling them and assigning similar entities to the same label, and different labels to different ones. In the context of waste material, classification is the ordering of waste products into given set of groups. Object classification is also a part of machine learning and data mining. It's idea is to classify the waste product into a given category based on pre-defined sets used for its training. These techniques can be broadly used by municipal corporations to keep control of the generated waste. Training a machine to classify such objects can help in clearly classifying the object as recyclable or not, and in which further categories they could fall in. From a pile of waste, it soon becomes easier to segregate objects into their appropriate divisions.

1.2.1 Importance of classification

We cannot treat everyone and everything in the same way. While it is okay to surf the internet at your room, the same might not be acceptable inside a classroom. Similarly, drinks are served cold and food hot. So, everywhere we go, we have to classify entities and interact with them accordingly.

Waste is a relative term. What might be an empty, useless bottle for someone might be a useful container for someone else. Similarly, not all kinds of waste material desreve the same kind of treatment. Where biodegradable waste should be used for fertilization of plants, metal should be melted and reused. In general, all recyclable material should be processed and reused. Plastic should not be burned etc. So it becomes necessary to classify waste into different categories so that all of them undergo optimal treatment.

1.3 Overview

In this paper, we attempt garbage classification. Classification is based on transfer learning of given input of images to learn and based on that predicting the category of the waste using the Inception model. The approach used in this paper is transfer learning on convolutional neural networks which is accomplished as follows.

1. Recognising small patterns in image matrix.
2. Assigning values to feature maps based on weighted values of previous neurons.
3. Detecting bigger patterns based on combination of small patterns.
4. Calculating the score of each category based on final layer of feature map values.

Chapter 2

Related Work

Previously, there were several neural network based image classification and analysis projects. However, there were none that pertain specifically to trash classification.

2.1 AlexNet

In the realm of image classification, one acknowledge and extremely capable CNN design is AlexNet[5], that won the 2012 ImageNet LargeScale Visual Recognition Challenge (ILSVRC). The design is comparatively straightforward and not extraordinarily deep, and is, of course, renowned to perform well. AlexNet was powerful as a result of it started a trend of CNN approaches being very talked-about within the ImageNet challenge and turning into the state of the art in image classification.

2.2 AutoTrash

The most similar project we've found was a project from the 2016 TechCrunch Disrupt Hackathon [6] within which the team created AutoTrash, associate degree auto-sorting trashcan that may distinguish between compost and employment employing a Raspberry Pi powered module and camera. Their project was designed making use of Googles TensorFlow and it conjointly includes hardware parts. One thing to notice regarding Auto-Trash is that it solely classifies whether or not something is compostable or recyclable, which may be a less complicated than having 5 or six categories. Another trash connected project was a smartphone application designed to coarsely phase a pile of garbage in a picture [7]. The goal of applying it was to permit citizens to trace and report garbage in their neighborhoods. The dataset used was obtained through Bing Image Search and therefore the authors extracted patches from the pictures to coach their network. The authors utilised a pre-trained AlexNet model and obtained a mean accuracy of 87.69%. The authors did well to require advantage of a pretrained model to enhance generalization. Alternative recycling primarily based classification issues used in physical options to associate degree object. In 1999, a project from Lulea University of Technology [8] worked on recycling metal scraps employing a mechanical form identifier. They used chemical and mechanical strategies like inquiring to spot the chemical contents and current separation. This paper's mechanical approach provides attention-grabbing advancement methods for our project. Another set of image primarily based classification of materials was performed on the Flickr Materials database [10] . The team used options like SIFT, color, microtexture and description form in a very theoretical procedure framework. This project is comparable to ours therein it makes an attempt to classify pictures supported material categories. However, the dataset used is totally different from ours since the pictures are square, of unsullied materials, and with no logos or deformation.

Chapter 3

Proposed Work

The proposed garbage classification process can be divided into three parts:

- Inception Model Usage [11]
- Data Preprocessing
- Training

3.1 Inception Model

Modern image recognition models have millions of parameters. Training them from scratch requires a lot of labelled training data and a lot of computing power (hundreds of GPU-hours or more). Transfer learning is a technique that shortcuts much of this by taking a piece of a model that has already been trained on a related task and reusing it in a new model.

Googles Inception-v3 is trained for the ImageNet [16] Large Visual Recognition Challenge using the data from 2012. This is a standard task in computer vision, where models try to classify entire images into 1000 classes, like "Zebra", "Dalmatian", and "Dishwasher". Inception model is based on deep convolutional neural networking.

The namesake of Inception v3 is the Inception modules it uses, which are basically mini models inside the bigger model. The same Inception architecture was used in

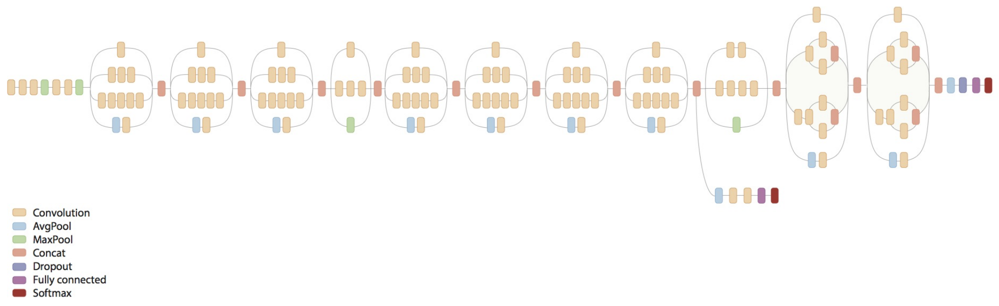


Figure 1: Inception Model Layers

the GoogLeNet model which was a state of the art image recognition net in 2014.

3.1.1 Inception modules

The inspiration comes from the idea that you need to make a decision as to what type of convolution you want to make at each layer: Do you want a 3×3? Or a 5×5? And this can go on for a while.

So why not use all of them and let the model decide? You do this by doing each convolution in parallel and concatenating the resulting feature maps before going to the next layer. Now lets say the next layer is also an Inception module. Then each of the convolutions feature maps will be passes through the mixture of convolutions of the current layer. The idea is that you dont need to know ahead of time if it was better to do, for example, a 3×3 then a 5×5. Instead, just do all the convolutions and let the model pick what's best. Additionally, this architecture allows the model to recover both local feature via smaller convolutions and high abstracted features with larger convolutions.

3.1.2 Architecture

Figure shows the architecture of a single inception module.

Notice that we get the variety of convolutions that we want; specifically, we will be using 1×1, 3×3, and 5×5 convolutions along with a 3×3 max pooling. Max

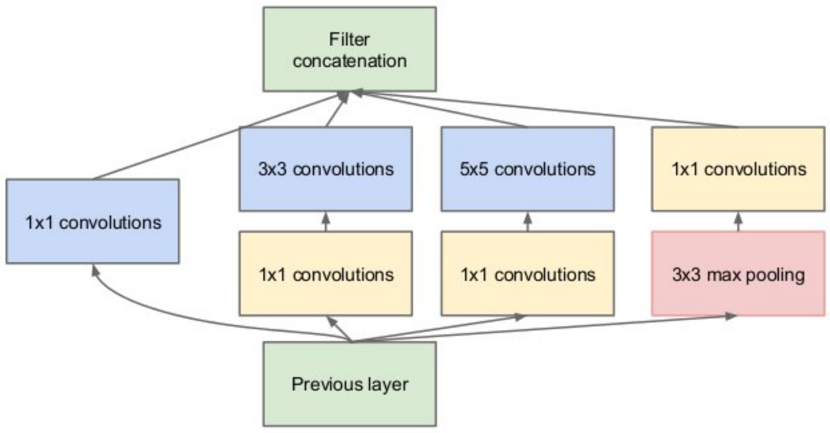


Figure 2: Inception Module Architecture

pooling is added to lower the data we need to deal with. The larger convolutions are more computationally expensive, so the model suggests first doing a 1×1 convolution reducing the dimensionality of its feature map, passing the resulting feature map through a relu (rectified linear unit), and then doing the larger convolution (in this case, 5×5 or 3×3). The 1×1 convolution is key because it will be used to reduce the dimensionality of its feature map.

3.1.3 Dimensionality reduction

The model suggests that you can use 11 convolutions to reduce the dimensionality of your input to large convolutions, thus keeping your computations reasonable. This allows the model to reduce the number of computations by a factor of 10!

3.1.4 Transfer Learning

Transfer learning[17] is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast computation and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

Transfer learning is an optimization that allows rapid progress or improved performance when modeling the second task. Transfer learning is related to problems such as multi-task learning and concept drift and is not exclusively an area of study for deep learning. Nevertheless, transfer learning is popular in deep learning given the enormous resources required to train deep learning models or the large and challenging datasets on which deep learning models are trained.

How to Use Transfer Learning?

You can use transfer learning on your own predictive modeling problems. Two common approaches are as follows:

1. Develop Model Approach
2. Pre-trained Model Approach

Developer Model Approach

- **Select Source Task.** You must select a related predictive modeling problem with an abundance of data where there is some relationship in the input data, output data, and/or concepts learned during the mapping from input to output data.
- **Develop Source Model.** Next, you must develop a skillful model for this first task. The model must be better than a naive model to ensure that some feature learning has been performed.
- **Reuse Model.** The model fit on the source task can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used.
- **Tune Model.** Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

▮ *Pre-trained Model Approach*

- **Select Source Model.** A pre-trained source model is chosen from available models. Many research institutions release models on large and challenging datasets that may be included in the pool of candidate models from which to choose from.
- **Reuse Model.** The model pre-trained model can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used.
- **Tune Model.** Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

3.2 Data Preprocessing

A common way of improving the results of image training is by deforming, cropping, or brightening the training inputs in random ways. This has the advantage of expanding the effective size of the training data thanks to all the possible variations of the same images, and tends to help the network learn to cope with all the distortions that will occur in real-life uses of the classifier. The biggest disadvantage of enabling these distortions in our script is that the bottleneck caching is no longer useful, since input images are never reused exactly. This means the training process takes a lot longer (many hours).

3.2.1 Flipping

Flipping means rotating the image by 180 degrees about vertical axis. It is easy to implement and effective to generate dataset that generally has different pixel positioning than the original one.

3.2.2 Zooming

Image zooming is achieved by pixel replication or by interpolation. Scaling is used to change the visual appearance of an image, to alter the quantity of information stored in a scene representation, or as a low-level preprocessor in multi-stage image processing chain which operates on features of a particular scale. Zooming also achieves the basic objective of generating more data out of existing one.

3.2.3 Bottlenecking

This phase analyzes all the images on disk and calculates and caches the bottleneck values for each of them. 'Bottleneck' is an informal term we often use for the layer just before the final output layer that actually does the classification[21]. (TensorFlow Hub calls this an "image feature vector".) This penultimate layer has been trained to output a set of values that's good enough for the classifier to use to distinguish between all the classes it's been asked to recognize. That means it has to be a meaningful and compact summary of the images, since it has to contain enough information for the classifier to make a good choice in a very small set of values. Generally, images are stored as arrays or grids of numbers (pixel values), and bottleneck files are just text files that store values from these grids/arrays.

Other ways of image morphing include warping, brightening, rotating, duplicating etc.

3.3 Training

Training is one of the chief parts of any machine learning project. This is where the developed model actually trains on the given data, and learns to classify images. The Inception model is trained in this step firstly on the ImageNet dataset and finally, the top layer is trained on the dataset of choice. The following features play an important part in increasing the efficiency of the model while training.

3.3.1 Automatic feature extraction

Probably the most important part of machine learning is to select features on the basis of which a model compares and classifies objects. Although easy for small tasks, this step becomes almost impossible to do manually when dealing with image classification. There are practically tens of thousands of possible features, and manual selection or hit and trial are infeasible. So, we let the model choose interesting values as features from the images. The selected features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data.

3.3.2 Max Pooling

Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned. This is done to in part to help over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation. Max pooling is done by applying a max filter to (usually) non-overlapping subregions of the initial representation Let's say there is a 4x4 matrix representing our initial input. Let's say, as well, that there is a 2x2 filter that run over the input. There is a stride of 2 (meaning the (dx, dy) for stepping over our input will be (2, 2)) and won't overlap regions. For each of the regions represented by the filter, max of that region will be taken and a new, output matrix is created where each element is the max of a region in the original input. This can be understood with the following image -

3.3.3 Flattening

Flattening transforms tridimensional (W-(s-1), H - (s-1), N) tensor into a mono-dimensional tensor (a vector) of size (W-(s-1))x(H - (s-1))xN. 2D convolution layers processing 2D data (for example, images) usually output a tridimensional tensor,

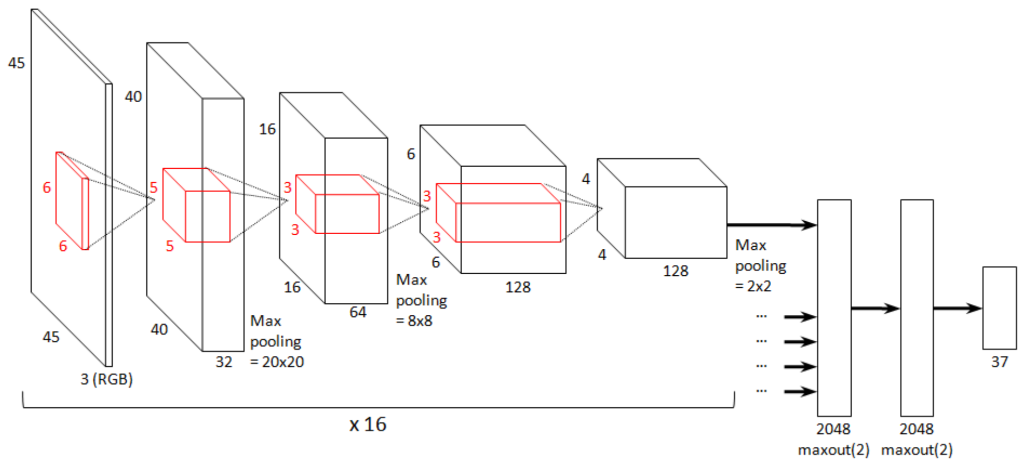


Figure 3: Max Pooling

with the dimensions being the image resolution (minus the filter size -1) and the number of filters. If you use a layer with N filters of size s on WxH images, the result will be of dimension (W-(s-1), H - (s-1), N). You have N images of resolution (W-(s-1), H - (s-1)). This structure is important if you want to chain convolution layers together or with other layers that perform a spatial treatment (pooling, upscaling, etc.). Within classification, usage of fully connected layers that do not take any structure (spatial or otherwise) into account for processing are done in the last steps of the network. The output of the last convolution layers are to be considered as a large piece of unstructured data

Chapter 4

Experimental Setup and Results Analysis

Garbage classification is basically a software based project that takes images as input and predicts classes digitally.

4.1 Requirements

For the project to run, some basic conditions must be met.

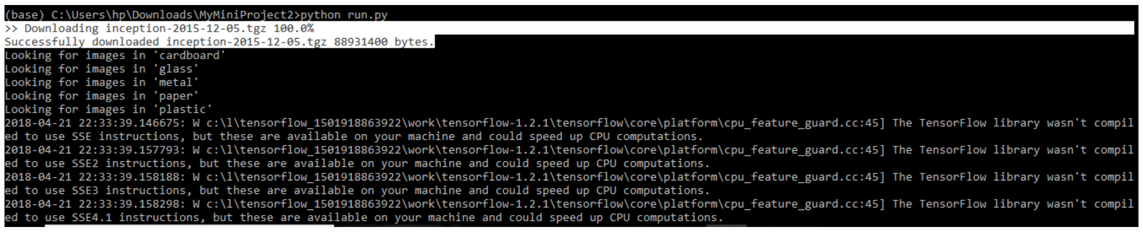
4.1.1 Hardware Requirements

The following hardware requirements must be met:

- Multicore processor
- Atleast 8GB of RAM
- Atleast 1GB disk space
- Preferably a decent graphics card, like GtX Nvidia graphics card

4.1.2 Software Requirements

This project also has some software requirements:



```
(base) C:\Users\hp\Downloads\MyMiniProject2>python run.py
>> Downloading inception-2015-12-05.tgz 100.0%
Successfully downloaded inception-2015-12-05.tgz 88931400 bytes.
Looking for images in 'cardboard'
Looking for images in 'glass'
Looking for images in 'metal'
Looking for images in 'paper'
Looking for images in 'plastic'
2018-04-21 22:33:39.146675: W c:\l\tensorflow_1501918863922\work\tensorflow-1.2.1\tensorflow\core\platform\cpu_feature_guard.cc:45] The TensorFlow library wasn't compil
ed to use SSE instructions, but these are available on your machine and could speed up CPU computations.
2018-04-21 22:33:39.157793: W c:\l\tensorflow_1501918863922\work\tensorflow-1.2.1\tensorflow\core\platform\cpu_feature_guard.cc:45] The TensorFlow library wasn't compil
ed to use SSE2 instructions, but these are available on your machine and could speed up CPU computations.
2018-04-21 22:33:39.158188: W c:\l\tensorflow_1501918863922\work\tensorflow-1.2.1\tensorflow\core\platform\cpu_feature_guard.cc:45] The TensorFlow library wasn't compil
ed to use SSE3 instructions, but these are available on your machine and could speed up CPU computations.
2018-04-21 22:33:39.158298: W c:\l\tensorflow_1501918863922\work\tensorflow-1.2.1\tensorflow\core\platform\cpu_feature_guard.cc:45] The TensorFlow library wasn't compil
ed to use SSE4.1 instructions, but these are available on your machine and could speed up CPU computations.
```

Figure 4: ImageNet Dataset Downloading

- Python, with version greater than 2.6
- An operating system that supports python, like Windows, Mac, or Ubuntu
- Some python libraries like tensorflow, numpy, scipy etc.

4.2 Working

The project uses Transfer Learning to re-train Inception v3 model based on convolutional neural networks on Google's Tensorflow architecture. The steps involved in the project are explained in sequential manner.

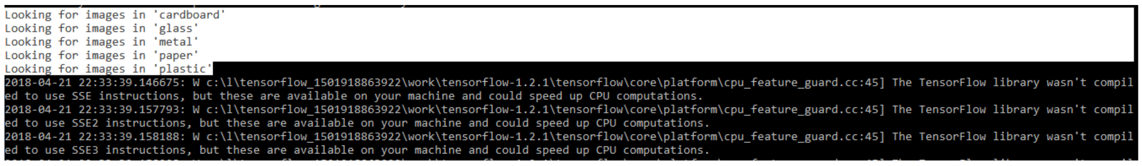
4.2.1 ImageNet Data Download

The first step is to train the Inception model on the ImageNet dataset. For that, the dataset is needed. In case the ImageNet dataset is not present in the specified path, it is downloaded from the link http://download.tensorflow.org/models/image/imagenet/inception-2015-12-05.tgz. Internet connectivity is required for this step.

4.2.2 Image Loading

The images on which the model needs to train, in this case, the images of garbage, are loaded in this step.

In order to start the transfer learning process, a folder named dataset needs to be created in the root of the project folder. This folder will contain the image data sets for all the subjects, for whom the classification is to be performed. The name of the folder in which an image is present is taken as the correct label for that image.



```
Looking for images in 'cardboard'
Looking for images in 'glass'
Looking for images in 'metal'
Looking for images in 'paper'
Looking for images in 'plastic'
2018-04-21 22:33:39.146675: W c:\l\tensorflow_1501918863922\work\tensorflow-1.2.1\tensorflow\core\platform\cpu_feature_guard.cc:45] The TensorFlow library wasn't compil
ed to use SSE instructions, but these are available on your machine and could speed up CPU computations.
2018-04-21 22:33:39.157793: W c:\l\tensorflow_1501918863922\work\tensorflow-1.2.1\tensorflow\core\platform\cpu_feature_guard.cc:45] The TensorFlow library wasn't compil
ed to use SSE2 instructions, but these are available on your machine and could speed up CPU computations.
2018-04-21 22:33:39.158188: W c:\l\tensorflow_1501918863922\work\tensorflow-1.2.1\tensorflow\core\platform\cpu_feature_guard.cc:45] The TensorFlow library wasn't compil
ed to use SSE3 instructions, but these are available on your machine and could speed up CPU computations.
```

Figure 5: Searching for data in dataset directory

For example, create the dataset folder and add the images for all the data sets in the following manner:

```
|
—- /dataset
|   |
|   |
|   − − /A
|   |        A1.jpg
|   |        A2.jpg
|   | ...
|   |
|   |
|   − − /B
|           B1.jpg
|           B2.jpg
|   ...
|
```

This enables classification of images between the A and B data sets.

4.2.3 Image Morphing

The loaded images are morphed, if needed to increase the size of dataset.

For example, Figure 6 is image of a glass bottle.

Figure 7 is an image of the same bottle, but flipped.

And Figure 8 is an image of the same bottle, but zoomed.

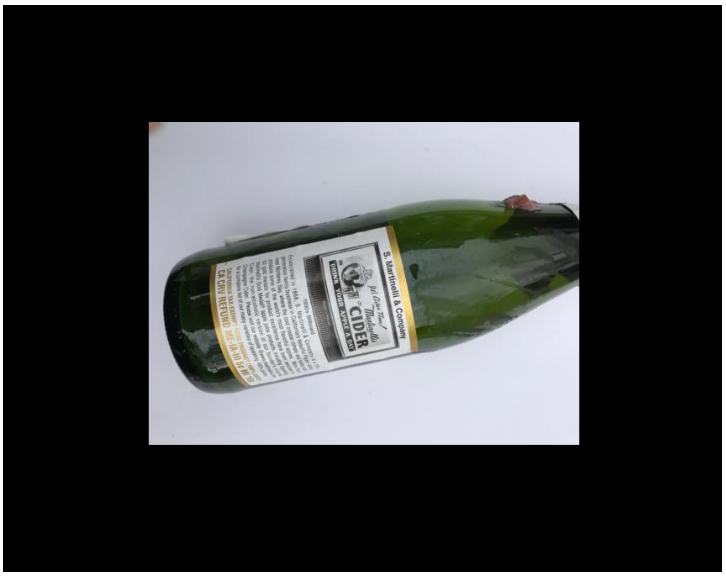


Figure 6: Glass bottle

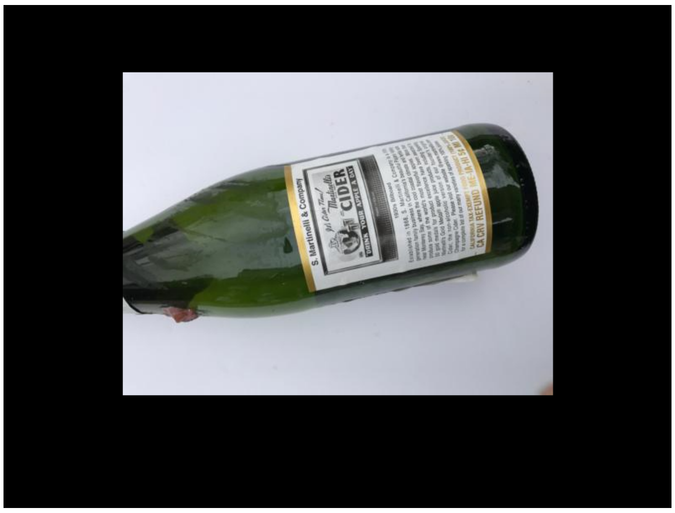


Figure 7: Rotated image of glass bottle

Figure 8: Zoomed image of glass bottle

We get three images from one image. Doing this for all dataset will increase its size by a factor of 3!

4.2.4 Dataset Splitting

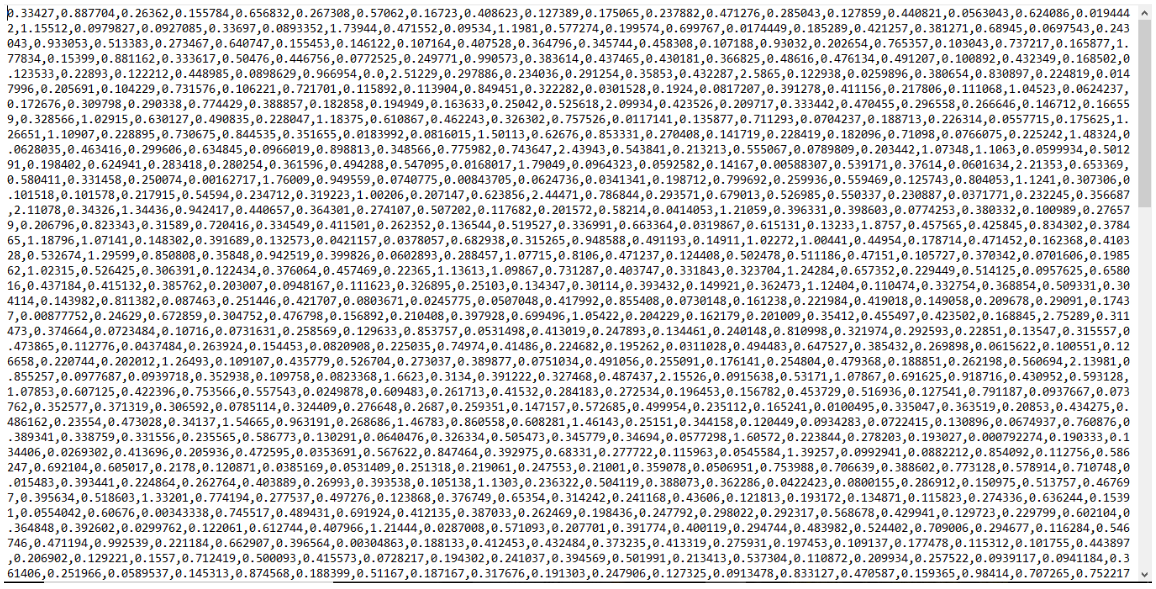
The dataset provided is of roughly 3000 images. In this part, the dataset provided is splitted into 3 parts - training, cross-validation and testing. Consider the following problem - Assume a machine that predicts if it will rain in a desert, according to atmospheric inputs. If the machine does nothing and just predict "No Rain" all the time, it will be correct in most cases, which means high accuracy. But in reality, the machine is not at all a good predictor.

To avoid such situations, additional cross checking is done using cross validation sets.

By default, 10% of the dataset is selected as cross-validation set and another 10% is selected as test set. The model trains on the remaining 80% of the provided dataset. These values can be modified.

4.2.5 Bottlenecking

As explained, the bottleneck values of images are stored (if training for the first time). Because every image is reused multiple times during training and calculating each bottleneck takes a significant amount of time, it speeds things up to cache



```
0.33427,0.887704,0.26362,0.155784,0.656832,0.267308,0.57062,0.16723,0.408623,0.127389,0.175065,0.237882,0.471276,0.285043,0.127859,0.440821,0.0563043,0.624086,0.019444
2,1.15512,0.0979827,0.0927085,0.33697,0.0893352,1.73944,0.471552,0.09534,1.1981,0.577274,0.199574,0.699767,0.0174449,0.185289,0.421257,0.381271,0.68945,0.0697543,0.243
043,0.933053,0.513383,0.273467,0.640747,0.155453,0.146122,0.107164,0.407528,0.364796,0.345744,0.458308,0.107188,0.93032,0.202654,0.765357,0.103043,0.737217,0.165877,1.
77834,0.15399,0.881162,0.333617,0.50476,0.446756,0.0772525,0.249771,0.990573,0.383614,0.437465,0.430181,0.366825,0.48616,0.476134,0.491207,0.100892,0.432349,0.168502,0
.123533,0.22893,0.122212,0.448985,0.0898629,0.966954,0.0,2.51229,0.297886,0.234036,0.291254,0.35853,0.432287,2.5865,0.122938,0.0259896,0.380654,0.830897,0.224819,0.014
7996,0.205691,0.104229,0.731576,0.106221,0.721701,0.115892,0.113904,0.849451,0.322282,0.0301528,0.1924,0.0817207,0.391278,0.411156,0.217806,0.111068,1.04523,0.0624237,
0.172676,0.309798,0.290338,0.774429,0.388857,0.182858,0.194949,0.163633,0.25042,0.525618,2.09934,0.423526,0.209717,0.333442,0.470455,0.296558,0.266646,0.146712,0.16655
9,0.328566,1.02915,0.630127,0.490835,0.228047,1.18375,0.610867,0.462243,0.326302,0.757526,0.0117141,0.135877,0.711293,0.0704237,0.188713,0.226314,0.0557715,0.175625,1.
26651,1.10907,0.228895,0.730675,0.844535,0.351655,0.0183992,0.0816015,1.50113,0.62676,0.853331,0.270408,0.141719,0.228419,0.182096,0.71098,0.0766075,0.225242,1.48324,0
.0628035,0.463416,0.299606,0.634845,0.0966019,0.898813,0.348566,0.775982,0.743647,2.43943,0.543841,0.213213,0.555067,0.0789809,0.203442,1.07348,1.1063,0.0599934,0.5012
91,0.198402,0.624941,0.283418,0.280254,0.361596,0.494288,0.547095,0.0168017,1.79049,0.0964323,0.0592582,0.14167,0.00588307,0.539171,0.37614,0.0601634,2.21353,0.653369,
0.580411,0.331458,0.250074,0.00162717,1.76009,0.949559,0.0740775,0.00843705,0.0624736,0.0341341,0.198712,0.799692,0.259936,0.559469,0.125743,0.804053,1.1241,0.307306,0
.101518,0.101578,0.217915,0.54594,0.234712,0.319223,1.00206,0.207147,0.623856,2.44471,0.786844,0.293571,0.679013,0.526985,0.550337,0.230887,0.0371771,0.232245,0.356687
,2.11078,0.34326,1.34436,0.942417,0.440657,0.364301,0.274107,0.507202,0.117682,0.201572,0.58214,0.0414053,1.21059,0.396331,0.398603,0.0774253,0.380332,0.100989,0.27657
9,0.206796,0.823343,0.31589,0.720416,0.334549,0.411501,0.262352,0.136544,0.519527,0.336991,0.663364,0.0319867,0.615131,0.13233,1.8757,0.457565,0.425845,0.834302,0.3784
65,1.18796,1.07141,0.148302,0.391689,0.132573,0.0421157,0.0378057,0.682938,0.315265,0.948588,0.491193,0.14911,1.02272,1.00441,0.44954,0.178714,0.471452,0.162368,0.4103
28,0.532674,1.29599,0.850808,0.35848,0.942519,0.399826,0.0602893,0.288457,1.07715,0.8106,0.471237,0.124408,0.502478,0.511186,0.47151,0.105727,0.370342,0.0701606,0.1985
62,1.02315,0.526425,0.306391,0.122434,0.376064,0.457469,0.22365,1.13613,1.09867,0.731287,0.403747,0.331843,0.323704,1.24284,0.657352,0.229449,0.514125,0.0957625,0.6580
16,0.437184,0.415132,0.385762,0.203007,0.0948167,0.111623,0.326895,0.25103,0.134347,0.30114,0.393432,0.149921,0.362473,1.12404,0.110474,0.332754,0.368854,0.509331,0.30
4114,0.143982,0.811382,0.087463,0.251446,0.421707,0.0803671,0.0245775,0.0507048,0.417992,0.855408,0.0730148,0.161238,0.221984,0.419018,0.149058,0.209678,0.29091,0.1743
7,0.00877752,0.24629,0.672859,0.304752,0.476798,0.156892,0.210408,0.397928,0.699496,1.05422,0.204229,0.162179,0.201009,0.35412,0.455497,0.423502,0.168845,2.75289,0.311
473,0.374664,0.0723484,0.10716,0.0731631,0.258569,0.129633,0.853757,0.0531498,0.413019,0.247893,0.134461,0.240148,0.810998,0.321974,0.292593,0.22851,0.13547,0.315557,0
.473865,0.112776,0.0437484,0.263924,0.154453,0.0820908,0.225035,0.74974,0.41486,0.224682,0.195262,0.0311028,0.494483,0.647527,0.385432,0.269898,0.0615622,0.100551,0.12
6658,0.220744,0.202012,1.26493,0.109107,0.435779,0.526704,0.273037,0.389877,0.0751034,0.491056,0.255091,0.176141,0.254804,0.479368,0.188851,0.262198,0.560694,2.13981,0
.855257,0.0977687,0.0939718,0.352938,0.109758,0.0823368,1.6623,0.3134,0.391222,0.327468,0.487437,2.15526,0.0915638,0.53171,1.07867,0.691625,0.918716,0.430952,0.593128,
1.07853,0.607125,0.422396,0.753566,0.557543,0.0249878,0.609483,0.261713,0.41532,0.284183,0.272534,0.196453,0.156782,0.453729,0.516936,0.127541,0.791187,0.0937667,0.073
762,0.352577,0.371319,0.306592,0.0785114,0.324409,0.276648,0.2687,0.259351,0.147157,0.572685,0.499954,0.235112,0.165241,0.0100495,0.335047,0.363519,0.20853,0.434275,0.
486162,0.23554,0.473028,0.34137,1.54665,0.963191,0.268686,1.46783,0.860558,0.608281,1.46143,0.25151,0.344158,0.120449,0.0934283,0.0722415,0.130896,0.0674937,0.760876,0
.389341,0.338759,0.331556,0.235565,0.586773,0.130291,0.0640476,0.326334,0.505473,0.345779,0.34694,0.0577298,1.60572,0.223844,0.278203,0.193027,0.000792274,0.190333,0.1
34406,0.0269302,0.413696,0.205936,0.472595,0.0353691,0.567622,0.847464,0.392975,0.68331,0.277722,0.115963,0.0545584,1.39257,0.0992941,0.0882212,0.854092,0.112756,0.586
247,0.692104,0.605017,0.2178,0.120871,0.0385169,0.0531409,0.251318,0.219061,0.247553,0.21001,0.359078,0.0506951,0.753988,0.706639,0.388602,0.773128,0.578914,0.710748,0
.015483,0.393441,0.224864,0.262764,0.403889,0.26993,0.393538,0.105138,1.1303,0.236322,0.504119,0.388073,0.362286,0.0422423,0.0800155,0.286912,0.150975,0.513757,0.46769
7,0.395634,0.518603,1.33201,0.774194,0.277537,0.497276,0.123868,0.376749,0.65354,0.314242,0.241168,0.43606,0.121813,0.193172,0.134871,0.115823,0.274336,0.636244,0.1539
1,0.0554042,0.60676,0.00343338,0.745517,0.489431,0.691924,0.412135,0.387033,0.262469,0.198436,0.247792,0.298022,0.292317,0.568678,0.429941,0.129723,0.229799,0.602104,0
.364848,0.392602,0.0299762,0.122061,0.612744,0.407966,1.21444,0.0287008,0.571093,0.207701,0.391774,0.400119,0.294744,0.483982,0.524402,0.709006,0.294677,0.116284,0.546
746,0.471194,0.992539,0.221184,0.662907,0.396564,0.00304863,0.188133,0.412453,0.432484,0.373235,0.413319,0.275931,0.197453,0.109137,0.177478,0.115312,0.101755,0.443897
,0.206902,0.129221,0.1557,0.712419,0.500093,0.415573,0.0728217,0.194302,0.241037,0.394569,0.501991,0.213413,0.537304,0.110872,0.209934,0.257522,0.0939117,0.0941184,0.3
61406,0.251966,0.0589537,0.145313,0.874568,0.188399,0.51167,0.187167,0.317676,0.191303,0.247906,0.127325,0.0913478,0.833127,0.470587,0.159365,0.98414,0.707265,0.752217
```

Figure 9: Bottlenecked file for an image

these bottleneck values on disk so they don't have to be repeatedly recalculated. By default, they're stored in the /tmp/bottleneck directory, and if we rerun the script they'll be reused so we don't have to wait for this part again. Typical bottleneck file looks like Figure 9:

The output in Figure 10 tells exactly how many bottlenecks were stored.

4.2.6 Initialization

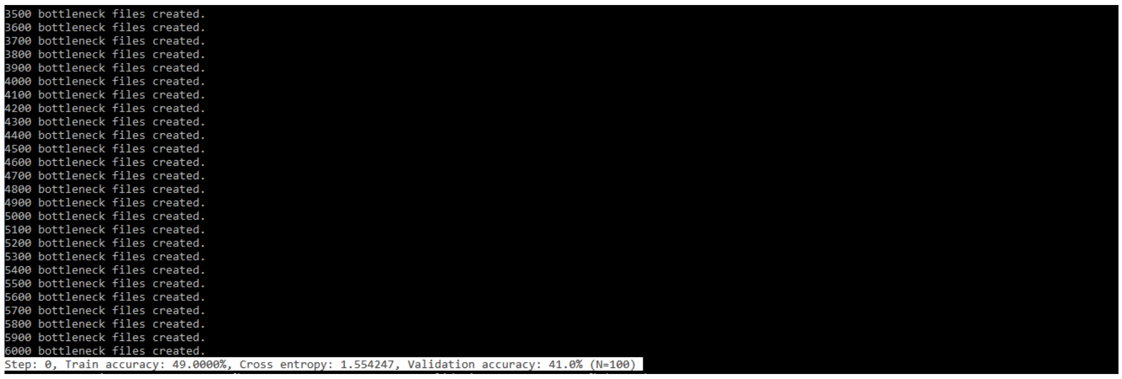
Now, the graph object for modelling is initialized with the command line values like train-validate-test split percentage, morphing, destination paths, number of epochs, evaluation metric etc. on tensorflow.

4.2.7 Training

At this point, the model is ready to be trained firstly on ImageNet (if it is being trained for the first time) and then on the images provided. With each passing epoch, the accuracy increases as the model learns from previous mistakes and learns to judge better. In the end, a decent accuracy is obtained.

```
2900 bottleneck files created.
3000 bottleneck files created.
3100 bottleneck files created.
3200 bottleneck files created.
3300 bottleneck files created.
3400 bottleneck files created.
3500 bottleneck files created.
3600 bottleneck files created.
3700 bottleneck files created.
3800 bottleneck files created.
3900 bottleneck files created.
4000 bottleneck files created.
4100 bottleneck files created.
4200 bottleneck files created.
4300 bottleneck files created.
4400 bottleneck files created.
4500 bottleneck files created.
4600 bottleneck files created.
4700 bottleneck files created.
4800 bottleneck files created.
4900 bottleneck files created.
5000 bottleneck files created.
5100 bottleneck files created.
5200 bottleneck files created.
5300 bottleneck files created.
5400 bottleneck files created.
5500 bottleneck files created.
5600 bottleneck files created.
5700 bottleneck files created.
5800 bottleneck files created.
5900 bottleneck files created.
6000 bottleneck files created.
```

Figure 10: Bottlenecking during training



```
3500 bottleneck files created.
3600 bottleneck files created.
3700 bottleneck files created.
3800 bottleneck files created.
3900 bottleneck files created.
4000 bottleneck files created.
4100 bottleneck files created.
4200 bottleneck files created.
4300 bottleneck files created.
4400 bottleneck files created.
4500 bottleneck files created.
4600 bottleneck files created.
4700 bottleneck files created.
4800 bottleneck files created.
4900 bottleneck files created.
5000 bottleneck files created.
5100 bottleneck files created.
5200 bottleneck files created.
5300 bottleneck files created.
5400 bottleneck files created.
5500 bottleneck files created.
5600 bottleneck files created.
5700 bottleneck files created.
5800 bottleneck files created.
5900 bottleneck files created.
6000 bottleneck files created.
Step: 0, Train accuracy: 49.0000%, Cross entropy: 1.554247, Validation accuracy: 41.0% (N=100)
```

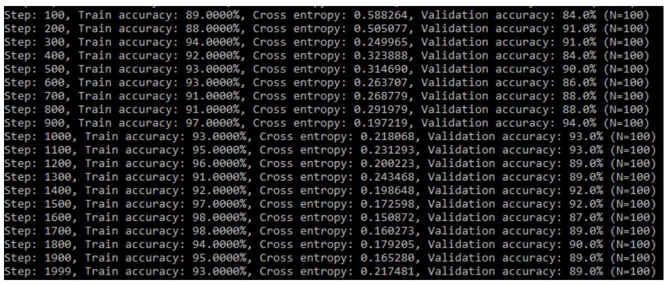
Figure 11: Low Accuracy in initial epoch

In the early iterations, accuracy is very low because the model has not trained so well, as shown in Figure 11.

But as iterations increase, so does the accuracy, as depicted in Figure 12.

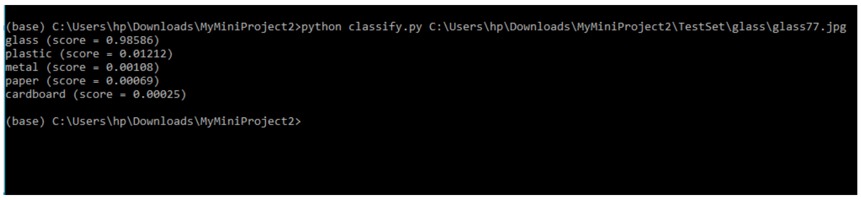
4.2.8 Saving model for future use

After training, the model is saved for future use as trained_graph.pb. This enables us to use our model for predictions without having to retrain it each time.



```
Step: 100, Train accuracy: 89.0000%, Cross entropy: 0.588264, Validation accuracy: 84.0% (N=100)
Step: 200, Train accuracy: 88.0000%, Cross entropy: 0.505077, Validation accuracy: 91.0% (N=100)
Step: 300, Train accuracy: 94.0000%, Cross entropy: 0.249965, Validation accuracy: 91.0% (N=100)
Step: 400, Train accuracy: 92.0000%, Cross entropy: 0.323888, Validation accuracy: 84.0% (N=100)
Step: 500, Train accuracy: 93.0000%, Cross entropy: 0.314690, Validation accuracy: 90.0% (N=100)
Step: 600, Train accuracy: 93.0000%, Cross entropy: 0.263707, Validation accuracy: 86.0% (N=100)
Step: 700, Train accuracy: 91.0000%, Cross entropy: 0.268779, Validation accuracy: 88.0% (N=100)
Step: 800, Train accuracy: 91.0000%, Cross entropy: 0.291979, Validation accuracy: 88.0% (N=100)
Step: 900, Train accuracy: 97.0000%, Cross entropy: 0.197219, Validation accuracy: 94.0% (N=100)
Step: 1000, Train accuracy: 93.0000%, Cross entropy: 0.218068, Validation accuracy: 93.0% (N=100)
Step: 1100, Train accuracy: 95.0000%, Cross entropy: 0.231293, Validation accuracy: 93.0% (N=100)
Step: 1200, Train accuracy: 96.0000%, Cross entropy: 0.200223, Validation accuracy: 89.0% (N=100)
Step: 1300, Train accuracy: 91.0000%, Cross entropy: 0.243468, Validation accuracy: 89.0% (N=100)
Step: 1400, Train accuracy: 92.0000%, Cross entropy: 0.198648, Validation accuracy: 92.0% (N=100)
Step: 1500, Train accuracy: 97.0000%, Cross entropy: 0.172598, Validation accuracy: 92.0% (N=100)
Step: 1600, Train accuracy: 98.0000%, Cross entropy: 0.150872, Validation accuracy: 87.0% (N=100)
Step: 1700, Train accuracy: 98.0000%, Cross entropy: 0.160273, Validation accuracy: 89.0% (N=100)
Step: 1800, Train accuracy: 94.0000%, Cross entropy: 0.179205, Validation accuracy: 90.0% (N=100)
Step: 1900, Train accuracy: 95.0000%, Cross entropy: 0.165280, Validation accuracy: 89.0% (N=100)
Step: 1999, Train accuracy: 93.0000%, Cross entropy: 0.217481, Validation accuracy: 89.0% (N=100)
```

Figure 12: Accuracy increases with epoches



```
(base) C:\Users\hp\Downloads\MyMiniProject2>python classify.py C:\Users\hp\Downloads\MyMiniProject2\TestSet\glass\glass77.jpg
glass (score = 0.98586)
plastic (score = 0.01212)
metal (score = 0.00108)
paper (score = 0.00069)
cardboard (score = 0.00025)

(base) C:\Users\hp\Downloads\MyMiniProject2>
```

Figure 13: Test result for a single image

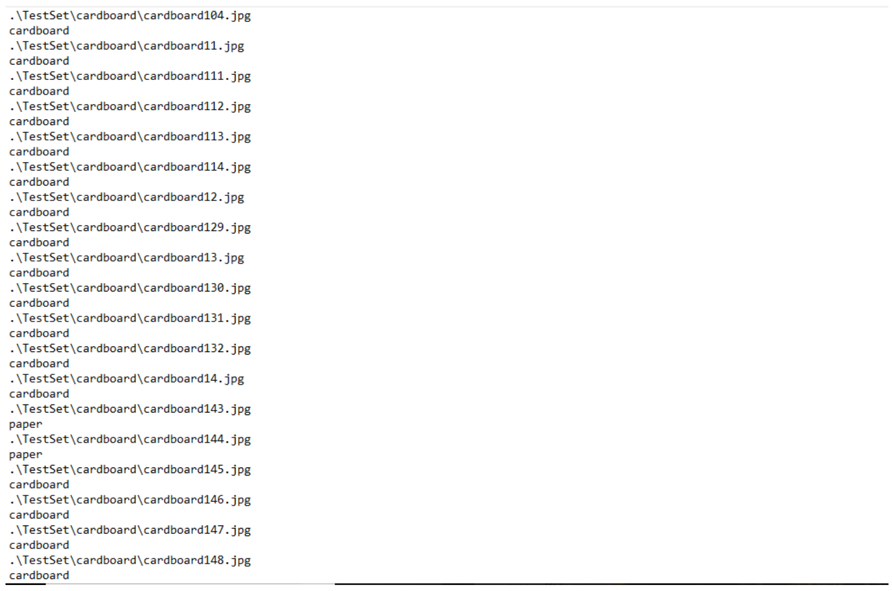
4.2.9 Predicting

The final step of the project is predicting the categories of new images. 601 separate images for testing purposes. The model can classify multiple images at a time too. If the path to a directory is given, all images inside that directory and the sub-directories are classified one by one. The prediction can be a single label or all labels with the likeness of the image belonging to that label. If a single file is given as input, we get the output as in Figure 13.

If path to a directory is given, we get the output as in Figure 14.

4.3 Result

The following results were obtained after extensive testing of the model obtained after training. It should be noted that although training takes time, it is a one time process only and subsequent predictions take only a few seconds.



```
.\TestSet\cardboard\cardboard104.jpg
cardboard
.\TestSet\cardboard\cardboard11.jpg
cardboard
.\TestSet\cardboard\cardboard111.jpg
cardboard
.\TestSet\cardboard\cardboard112.jpg
cardboard
.\TestSet\cardboard\cardboard113.jpg
cardboard
.\TestSet\cardboard\cardboard114.jpg
cardboard
.\TestSet\cardboard\cardboard12.jpg
cardboard
.\TestSet\cardboard\cardboard129.jpg
cardboard
.\TestSet\cardboard\cardboard13.jpg
cardboard
.\TestSet\cardboard\cardboard130.jpg
cardboard
.\TestSet\cardboard\cardboard131.jpg
cardboard
.\TestSet\cardboard\cardboard132.jpg
cardboard
.\TestSet\cardboard\cardboard14.jpg
cardboard
.\TestSet\cardboard\cardboard143.jpg
paper
.\TestSet\cardboard\cardboard144.jpg
paper
.\TestSet\cardboard\cardboard145.jpg
cardboard
.\TestSet\cardboard\cardboard146.jpg
cardboard
.\TestSet\cardboard\cardboard147.jpg
cardboard
.\TestSet\cardboard\cardboard148.jpg
cardboard
```

Figure 14: Test result for a multiple images in a directory

Table 1: Accuracy Table

Data type	**Correctly Classified Images**	**Total Images**	**Accuracy**
Dataset	2885	3039	94.93%
Independent Test Set	559	601	93.01%

4.4 Result Analysis

Looking at the table, the accuracy appears to be high in training set. The problem of overfitting is also avoided since test accuracy is high too. This means that the model is doing well and is able to distinguish between different images with appreciable accuracy.

However, there are some mis-classifications. For example, the cardboard image shown in Figure 15 was incorrectly classified as paper.

This seems like a forgivable mistake, but it also means that there is scope for improvement. There are some features which although work for majority of cases, fail in some tricky ones.

The following could be done to increase the accuracy of the model even further:

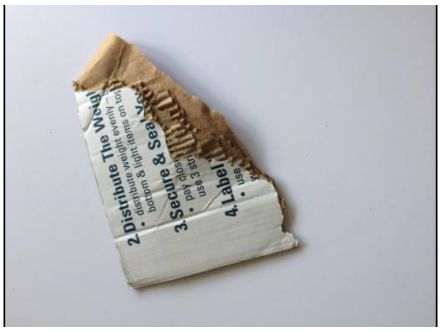


Figure 15: Cardboard image incorrectly classified as paper

- Collect more data
- Use deeper model with more layers
- Change the underlying model for something better

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Convolutional Neural Networks showcase high levels of control over performance that can be achieved by making effective use of theoretical and mathematical insights. Convolutional Neural Networks are the most effective models when it comes to image classification. Many real world problems are being efficiently tackled using Convolutional Neural Networks, and MNIST [9] represents a simple, Hello World-type use-case of this technique. More complex problems such as object and image recognition require the use of deep neural networks with millions of parameters to obtain state-of-the-art results.

Thus a Machine Learning researcher or engineer in todays world can rejoice at the technological melange of techniques at her disposal, among which an in-depth understanding of Convolutional Neural Networks is both indispensable and empowering.

5.2 Future work

The framework used in this project for the application of object count can be further extended and improvised through various means. Firstly, we can make simple modifications like increasing the training size, more epochs in training, and bigger size images for improving the accuracies.

Many technologies are used for waste collection as well as for waste management. The Information gathering is cumbersome. So a robot can be developed using our software module that will be capable of detecting the object in the random movement and after detecting the object the robot senses by webcam and followed by image processing, after the segmentation process the robot classifies the waste and dump it into different classes.

Among the promising areas of neural networks research are recurrent neural networks (RNNs) using long shortterm memory (LSTM). These areas are delivering the current state of the art in time-series recognition tasks like speech recognition and handwriting recognition. RNN/autoencoders are also capable of generating handwriting/ speech/images with some known distribution. Deep belief networks, another promising type of network utilizing restricted Boltzman machines (RMBs)/autoencoders, are capable of being trained greedily, one layer at a time, and hence are more easily trainable for very deep networks

Appendix A

Some Complex Proofs and simple Results

A.1 Building a Complex Neural Network

Convolutional Neural Networks produce the state of the art results for many computer vision tasks. A lot of work and thought has been put into the CNN model and its specific details to make it work so well. This success has prompted many attempts at expanding and improving this model. Inspired by the motivations presented in chapter 1, we consider complex valued Convolutional Neural Networks where both inputs and weights are complex. We build our model as a generalization to the real model, with the hope of applying the known practices and shared beliefs about Convolutional Neural Networks to its complex valued variation.

Given the aforementioned building blocks, the last detail before implementing a CNN is to specify its design end to end, and to decide on the layer dimensions of the Convolutional layers. A quick and dirty empirical formula[15] for calculating the spatial dimensions of the Convolutional Layer as a function of the input volume size and the hyperparameters we discussed before can be written as follows: For each (ith) dimension of the input volume, pick:

$W_{out}(i) = 1 + \frac{W_{In}(i) - R + 2P}{S}$

where is the (i^{th}) input dimension, R is the receptive field value, P is the padding value, and S is the value of the stride. Note that the formula does not rely on the depth of the input.

To better understand better how it works, lets consider the following example:

1. Let the dimensions of the input volume be 288x288x3, the stride value be 2 (both along horizontal and vertical directions).

2. Now, since W_{In} = 288 and S = 2, (2.P R) must be an even integer for the calculated value to be an integer. If we set the padding to 0 and R = 4, we get W_{out}=(288-4+2.0)/2+1 =284/2 + 1 = 143. As the spatial dimensions are symmetrical (same value for width and height), the output dimensions are going to be: 143 x 143 x K, where K is the depth of the layer. K can be set to any value, with increasing values for every Convolutional layer added. For larger networks values of 512 are common.

3. The output volume from a Convolutional layer either has the same dimensions as that of the Convolutional layer (143x143x2 for the example considered above), or the same as that of the input volume (288x288x3 for the example above).

Input $\longrightarrow$
[([Conv $\rightarrow ReLu] \times N) - - > Pool?] \times M \longrightarrow$
[FullyConnected $\rightarrow ReLu] \times K \longrightarrow FullyConnected$

The generic arrangement of layers can thus be summarized as follows:

Where N usually takes values between 0 and 3, $M \geq 0$ and $K[0, 3)$. The expression indicates multiple layers, with or without per layer-Pooling. The final layer is the fully-connected output layer. See [8] for more case-studies of CNN architectures, as well as a detailed discussion of layers and hyper-parameters.

References

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.

[2] Trashnet and Related Files, https://github.com/garythung/trashnet

[3] How to Retrain an Image Classifier for New Categories, https://www.tensorflow.org/tutorials/image_retraining

[4] Tensorflow,An open source machine learning framework for everyone. https://www.tensorflow.org/

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks, in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 10971105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenetclassification-with-deep-convolutional-neural-networks.pdf

[6] J. Donovan, Auto-trash sorts garbage automatically at the techcrunch disrupt hackathon. [Online]. Available: https://techcrunch.com/2016/09/13/auto-trash-sortsgarbage-automatically-at-the-techcrunch-disrupt-hackathon/

[7] G. Mittal, K. B. Yagnik, M. Garg, and N. C. Krishnan, Spotgarbage: Smartphone app to detect garbagTe using deep learning, in Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, ser. UbiComp 16. New York, NY, USA: ACM, 2016, pp. 940945. [Online]. Available: http://doi.acm.org/10.1145/2971648.2971731

[8] Advanced Image Classification With Lulea, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.896.554&rep=rep1type=pdf

[9] MNIST Dataset, yann.lecun.com/exdb/mnist

[10] Training models using Flickr, https://www.researchgate.net/publication/274710144_Flickr_Image_C

[11] Google's Inception, https://www.kaggle.com/google-brain/inception-v3

[12] Inception modules: explained and implemented, https://hacktilldawn.com/2016/09/25/inception-modules-explained-and-implemented/

[13] Inception in TensorFlow, https://github.com/tensorflow/models/tree/master/research/inception

[14] Transfer Learning: retraining Inception V3 for custom image classification with Wisdom d'Almeida, https://becominghuman.ai/transfer-learning-retraining-inception-v3-for-custom-image-classification-2820f653c557

[15] Introduction to Tensorflow, https://codelabs.developers.google.com/codelabs/cpb102-txf-learning/index.html?index=..%2F..%2Findex#1

[16] ImageNet, www.image-net.org/

[17] A Gentle Introduction to Transfer Learning for Deep Learning by Jason Brownlee, https://machinelearningmastery.com/transfer-learning-for-deep-learning/

[18] Image Recognition With Tensorflow, https://www.tensorflow.org/tutorials/image_recognition

[19] Classication of Trash for Recyclability Status, http://cs229.stanford.edu/proj2016/report/ThungYang-ClassificationOfTrashForRecyclabilityStatus-report.pdf

[20] CNN Made Easy, https://xrds.acm.org/blog/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/

[21] BottleNecking, cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf