



Convolutional Neural Network for Automatic Trash Classification

Stefán Sturla Stefánsson



**Electrical and Computer Engineering
University of Iceland
2018**

Convolutional Neural Network for Automatic Trash Classification

Stefán Sturla Stefánsson

20 ECTS thesis submitted in partial fulfilment of a
Baccalaureus Scientiarum degree in Mechatronic Engineering

Advisors
Andri Þorláksson
Behnood Rasti

Faculty of Electrical and Computer Engineering
School of Engineering and Natural Sciences
University of Iceland
Reykjavík, Des 2018

Convolutional Neural Network for Automatic Trash Classification
20 ECTS thesis submitted in partial fulfilment of a Baccalaureus Scientiarum degree in
Mechatronics Engineering Technology

Copyright © 2018 Stefán Sturla Stefánsson
All rights reserved

Keilir Institute of Technology
School of Engineering and Natural Sciences
University of Iceland
Skólavegur 3
220 Hafnarfjörður

Phone: 578 4000

Bibliographic Information:

Stefán Sturla Stefánsson, 2018, *Convolutional Neural Network for Automatic Trash Classification*, BSc thesis, School of Engineering and Natural Sciences, University of Iceland, pp. 53.

Printing: Háskólaprent ehf., Fálkagata 2, 107 Reykjavík
Reykjavík, Iceland, Desember 2018

Abstract

This thesis is about adapting a previously built mechanism called ATC that in its present state can place trash into five different categories (plastic, paper, aluminium cans, plastic bottles and general waste) by pressing a button for the selected category. Now by implementing a Convolution Neural Network to the mechanism an attempt will be made to make this process automatic and therefore the push buttons irrelevant.

Datasets for four categories will be made in the camera chamber of the ATC mechanism duplicating the future environment for analysis. Dataset pre-processing with resizing and image augmentation will be discussed as well as splitting the data to training, validation and test sets.

The main features for building, training and testing a Convolution Neural Network are discussed with my decisions and reasoning on selected parameters. Brief history of Neural Networks, Convolution Neural Networks and some successful architecture are reviewed.

The top model accuracy and loss function on the training, validation and test data are displayed visually with confusion matrices and graphs along with comparison with another model. The resulting top model ended up having six convolution layers that achieved 98.65% prediction accuracy on the unseen test images.

Útdráttur

Ritgerð þessi fjallar um betrubætur á áður smíðuðum vélbúnaði sem kallast ATC, sem í núverandi ástandi er fær um að flokka sorp í fimm mismunandi hólf (plast, pappír, áldósir og almennt rusl) á handvirknan máta með því að þrýsta á hnapp fyrir viðkomandi flokk. Með innleiðingu á þyrpingarneti (Convolutional Neural Networks) er ætlunin að gera vélbúnaðinn sjálfvirknan og þ.a.l. gera þrýsihnappa óþarfa.

Búin verða til gagnasett mynda fyrir þá fjóra flokka sem til þarf en myndirnar eru teknar í myndavélahólfi á vélbúnaði til að líkja eftir raunverulegum aðstæðum sem notaðar verða við greiningar í framtíðinni. Skoðuð er forvinna gagnasetta með breytingum á myndum s.s. stærð þeirra, skýrleika og fleira. Einnig er rætt um skiptingu alls gagnasettsins í þjálunar-, staðfestingar- og prufunarsett.

Helstu atriði við hönnun, þjálfun og prufun á þyrpingarnetum eru rædd þar sem gert er grein fyrir minni ákvarðanatöku á mögulegum breytum hvað varðar netið. Farið er yfir sögu þyrpingar- og tauganeta ásamt uppbyggingu nokkurra vel heppnaðra slíkra neta.

Þyrpingarnetið sem varð fyrir valinu náði 98.65% nákvæmni í flokkun á óséðum myndum en niðurstöður eru kynntar og skoðaðar á myndrænan máta fyrir gagnasettin þrjú þ.e. fyrir þjálfunar-, staðfestingar- og prufunarsettin. Samanburður er einnig gerður á módelinu sem varð fyrir valinu og því sem var næst því í nákvæmni.

*To my beloved fiancé Lilja for all her support
throughout my time at the University.*

Table of Contents

List of Figures	ix
List of Tables.....	xi
Abbreviations.....	xiii
Acknowledgements	xv
1 Introduction.....	1
1.1 Project description.....	2
1.2 Objectives.....	3
1.3 Thesis Overview	4
2 Background	5
2.1 Brief History of Neural Networks	5
2.2 Neurons the basic of a Neural Network	6
2.3 Neural Network structure	7
3 Convolutional Neural Networks	9
3.1 Convolutional Layers	9
3.2 Padding.....	11
3.3 Fully connected layer	11
3.4 Architecture of Convolutional Neural Networks	11
4 Important features of the model.....	15
4.1 Activation Functions	15
4.2 Pooling.....	17
4.3 Overfitting and Underfitting.....	17
4.4 Dropout.....	19
4.5 Early stopping.....	19
4.6 Optimization.....	20
4.7 Dataset	20
4.7.1 Dataset Fabrication	20
4.8 Data pre-processing	22
4.8.1 Dataset Augmentation.....	22
4.8.2 One hot encoding	23
4.9 Training, Validation and Test sets.....	23
4.10 Environment	24
4.11 Top model.....	25
5 Results.....	27
5.1 Accuracy results of the top model	27
5.2 Accuracy results of the second-best model	31
5.3 Comparison of top and second model	35

6 Conclusion	39
6.1 Discussion and future work.....	39
7 References.....	41
APPENDIX	45

List of Figures

Figure 1: The figure shows the ATC mechanism designed in Inventor Autodesk.....	2
Figure 2: Flowchart for ATC functionality before (left side) and after (right side) implementing a CNN.....	3
Figure 3: Mark 1 Perseptron machine. (Image taken from the following link: http://www.en.wikipedia.org/wiki/Perceptron)	5
Figure 4: Sketch of a biological neuron. (Image taken from the following link: http://www.commonswikimedia.org/wiki/File:Neuron.svg)	6
Figure 5: Computational neuron. (Image taken from the following link: http://www.researchgate.net/figure/ANN-structure-of-a-nonlinear-model_fig25_316877711)	6
Figure 6: Example of a neural network having an input layer, hidden layer and an output layer. (Image taken from the following link: http://www.commonswikimedia.org/wiki/File:Colored_neural_network.svg).....	7
Figure 7: Shows the steps of a convolutional layer operation.	9
Figure 8: Shows an input image with a convolutional filter of size $3 \times 3 \times 3$ which outputs a 4×4 matrix feature map.....	10
Figure 9: Shows an input image with a convolutional filter of size 3×3 which outputs a $4 \times 4 \times 2$ matrix feature map.	10
Figure 10: Shows the architecture of LeNet-5 developed in 1998. (Image taken from the following link: http://www.yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf)	11
Figure 11: Shows the AlexNet architecture from 2012. (Image taken from the following link: http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf).....	12
Figure 12: Shows a revised architecture of Inception developed by the Google team. (Image taken from the following link: http://www.adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html)	12
Figure 13: Shows the ResNet architecture from Microsoft. (Image taken from the following link: http://www.arxiv.org/pdf/1512.03385.pdf)	13
Figure 14: Plot of a sigmoid activation function $S(x) = e^x/(1 + e^x)$	15
Figure 15: Plot of a ReLU activation function $R(x) = \max(0, x)$	16
Figure 16: Plot of a Tanh activation function $\tanh(x) = 2/(1 + e^{-2x}) - 1$	16

Figure 17: Example of max pool 2×2 filter on a 4×4 matrix. (Image taken from the following link: http://www.en.wikipedia.org/wiki/File:Max_pooling.png)	17
Figure 18: Shows how under and overfitting looks on a graph. (Image taken from the following link: http://www.medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76)	18
Figure 19: Shows how dropout randomly drops neurons connections. (Image taken from the following link: http://www.researchgate.net/figure/Dropout-neural-network-model-a-is-a-standard-neural-network-b-is-the-same-network_fig3_309206911).....	19
Figure 20: Example of images taken above the camera chamber. Images are from the four categorise needed to classify.	21
Figure 21: Image proportion over the entire dataset, containing training, validation and testing data.....	23
Figure 22: Flow chart of the model from TensorFlow Tensorboard revealing step by step information.	25
Figure 23: Showing raw accuracy results on training set.....	27
Figure 24: Results shown for loss function on training set.	27
Figure 25: Validation accuracy results shown.	28
Figure 26: Validation set loss shown in graph.	28
Figure 27: Showing raw accuracy results on training set.....	31
Figure 28: Results shown for loss function on training set.	31
Figure 29: Validation accuracy results shown.	32
Figure 30: Validation set loss.....	32
Figure 31: Showing raw accuracy comparison results on training sets.	35
Figure 32: Results comparison shown for loss function on training sets.....	35
Figure 33: Validation accuracy comparison results shown.....	36
Figure 34: Validation set loss comparison shown.....	36

List of Tables

Table 1: Showing one hot encoding on the four classes.....	23
Table 2: Confusion matrix for test set results plotted with scikit Learn.....	29
Table 3: Confusion matrix for normalized test results plotted with scikit Learn.	30
Table 4: Confusion matrix for test set results plotted with scikit Learn.....	33
Table 5: Confusion matrix for normalized test results plotted with scikit Learn.	34
Table 6: Comparison of top and second model predictions on test set.	37

Abbreviations

API	Application Programming Interface
CNN	Convolutional Neural Network
ConvNet	Convolutional Neural Network
CPU	Central Processing Unit
DL	Deep Learning
GPU	Graphics Processing Unit
GUI	Graphical User Interface
ML	Machine Learning
NN	Neural Network
PC	Personal Computer
ReLU	Rectified Linear Unit
Tanh	Hyperbolic Tangent Function

Acknowledgements

I would like to thank my advisors Andri Þorláksson and Behnood Rasti for their input and guidance throughout the project as well as much appreciated help from them and Þórir Kristinsson in building the ATC mechanism.

I would like to thank all my fellow students that were nothing but helpful and friendly during my time at the University of Iceland.

Lastly, I would like to thank my beloved Lilja, her parents Gunnar and Dista as well as my parents Ingunn and Stefán for all their support and help during my time at the University.

1 Introduction

Over the last couple of decades environmental awareness has increased significantly. This is a complicated issue with many aspects where some are more difficult to solve than others. There are however things that can be done on an individual level. One of them is responsible waste disposal. When done correctly it saves resources, energy and reduces the amount of waste sent to landfills and incinerators.

The focus of this research project is to see if improvements can be made in recycling for normal household and institution, that is improvements in classification of trash to the correct category. Most often the trash is categorised into five compartments (1: Plastic bottles, 2: Aluminium cans, 3: General waste, 4: Paper, 5: Plastic). A part of the problem is that consumers are not sorting it right. They may not be entirely sure about which category the trash belongs to or do not care enough to classify the waste and put it in general waste. Therefore, a mechanical way of sorting could be an effective solution to classify the waste.

According to Sorpa[1], the biggest Icelandic recycling company, it is roughly estimated that up to 70% of garbage that is put in the general waste category is of recyclable matter. Statistics from 2014 also show that the total waste for EU nations was over 2.503 million tonnes the highest recorded, where only 36.2% was recycled [2].

This project will by no means solve these overwhelming issues of poor recycling, but rather show a possible way of implementing a convolutional neural network to a mechanical device to improve recycling for smaller attributes.

1.1 Project description

This thesis project includes improving a previously built and designed prototype of a mechanism that I have been working on in previous courses of my studies.

This mechanism which is called ATC (Automatic Trash Classification) and can be seen in figure 1. It moves trash mechanically to a category chosen by a user. The way the device works is that a user puts trash in to allocated compartment and decides which category the trash belongs to with a selection of five push buttons, that is which of the five categories: general waste, paper, plastic, plastic bottles or aluminium cans. The mechanism then rotates the trash to a camera chamber that takes a picture of the trash before rotating it to its category.

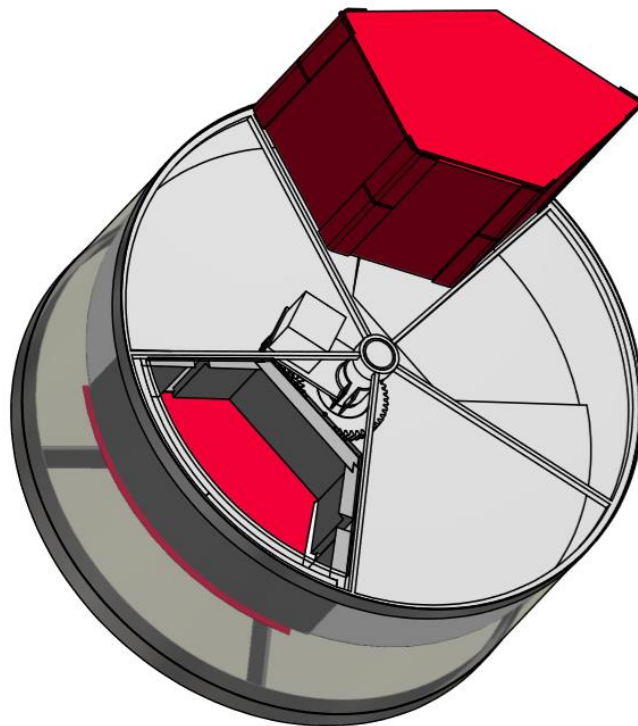


Figure 1: The figure shows the ATC mechanism designed in Inventor Autodesk.

With this thesis an attempt will be made to make this procedure automatic, by implementing a pretrained Convolutional Neural Network (CNN,ConvNet) that can automatically classify which category the trash belongs in. The process will then be that the user puts trash in the device, it rotates to the camera chamber where the picture is taken and analysed by the pretrained CNN, after analysing the picture a prediction will be made to which categories it belongs to. The mechanism will then rotate the trash to its designated category. A flowchart of the ATC functionality before and after the implementation of a CNN can be seen in figure 2.

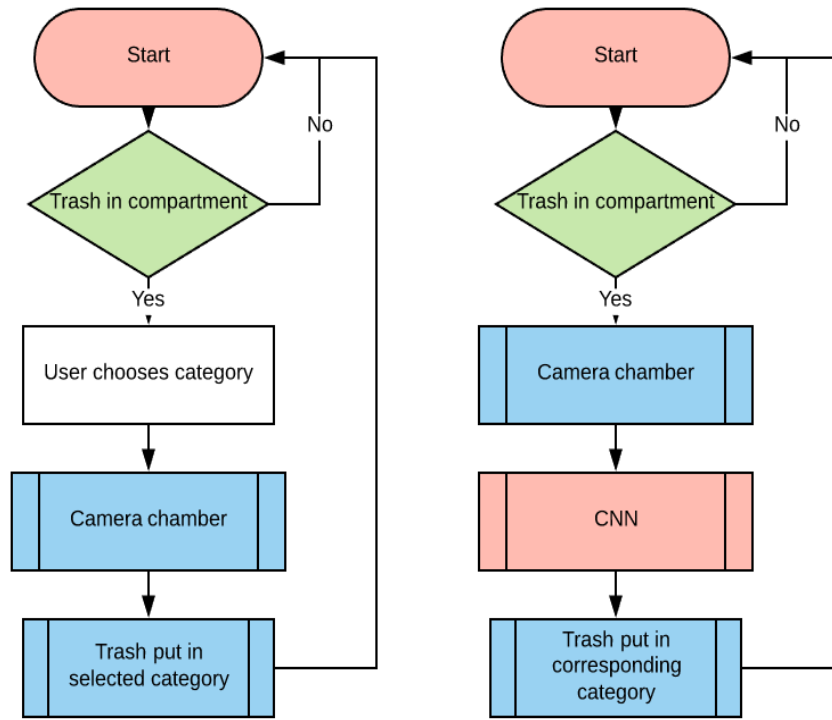


Figure 2: Flowchart for ATC functionality before (left side) and after (right side) implementing a CNN.

1.2 Objectives

The objective of this thesis is to obtain the best possible prediction accuracy on a test set for categorization in waste management by designing a CNN that later will be implemented to a mechanism that does the sorting for us.

Emphasis will be on writing the CNN program written in Python programming language with the help of Tensor Flow and TFLearn open source library for the design. Creating or obtaining a dataset with the relevant images for the four classes. Go through data pre-processing, training, validation and test phases to find optimal results with all the features that are required with tuning and testing the model. Final objectives will be to show the results graphically and display the accuracy and loss functions of training sets, validation sets and test sets.

1.3 Thesis Overview

The following summary provides an overview over the main chapters featured in the thesis.

- ***Chapter 1 – Introduction***

Gives a brief introduction, objectives and description of the project.

- ***Chapter 2 – History and structure of Neural Networks***

Brief history and structure of Neural Networks as well as the construction of neurons.

- ***Chapter 3 – Convolutional Neural Networks***

An overview of Convolutional Neural Networks and the main features including the convolution layer, padding and fully connected layer.

- ***Chapter 4 – Important features of the model***

The chapter involves my selections of various functions of the model as well as reasoning for those decisions. Building the dataset is covered with pre-processing and splitting the data in to training, validation and test sets. The programming environment and the model that was chosen is displayed and discussed in detail.

- ***Chapter 5 – Results***

Result on the top model accuracy and loss is presented for the three sets of training, validation and test sets.

- ***Chapter 6 – Conclusion***

The chapter goes briefly over the results of the model along with thoughts and prospects of the project.

2 Background

Neural Network's (NN's) are the first computational processing prediction system, generally used today in artificial intelligence, machine learning, computer vision and pattern recognition. This chapter provides a brief review of Neural Network and its history.

2.1 Brief History of Neural Networks

The idea of Neural Networks is inspired of how neurons in the brain function, when neurophysiologist Warren McCulloch and mathematician Walter Pitts in 1943 made an electrical circuit with threshold logic to mimic the thought process [3].

Five years later Donald Hebb wrote a book on the matter called The Organization of Behaviour [4], where he came to the conclusion that neural pathways are strengthened each time they are used.

In 1957 Frank Rosenblatt, an American psychologist came up with the Perceptron seen in figure 3. It is a two layer network showing an ability to learn certain classifications by adjusting connection weights [5].

In 1959 Bernard Widrow and Marcian Hoff tackled the first real world problem by using adaptive filter that eliminated noise and echoes in phone lines, a tactic still used today [6].

There are many other discoveries that led to gradual improvements in the field, but it came to a halt when Marvin Minsky published the book Perceptrons in 1969 [7]. Minsky argued that Rosenblatt's perceptron had fundamental flaws to it, that the model could not learn how to evaluate exclusive-or logical function [8] and that it would take several if not infinite numbers of iterations to compute an output. This was widely recognised and was the beginning of the "Artificial Intelligence winters" [9] where government funding's were reduced significantly for researches and therefore few discoveries were made for the next 13 years.

Renewed interest in NN materialised when John Hopfield came up with a bidirectional line connection [10] between neurons in his paper in 1982. That same year Japan became competitive proclaiming their interest with a new generation attempted on neural networks which led to more funding's in the field in United States for fear of being left behind with this potentially powerful technology [11].

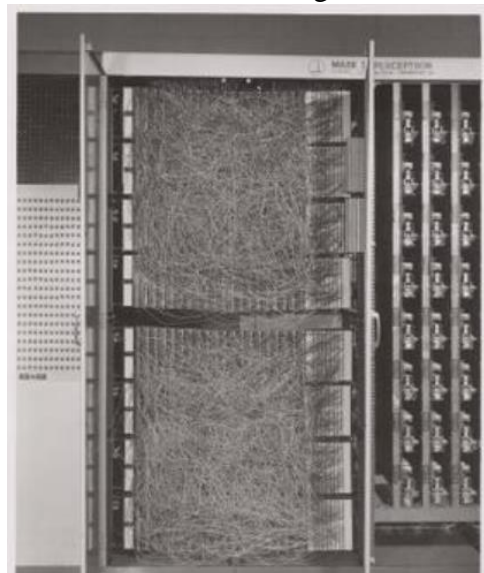


Figure 3: Mark 1 Perseptron machine.
(Image taken from the following link:
[http://www.
en.wikipedia.org/wiki/Perceptron](http://www.en.wikipedia.org/wiki/Perceptron))

Today, with ever more computing power NN and branches of it like CNN, Machine Learning (ML) and Deep Learning (DL) are all around us. From image recognition to self-driving cars, diagnosis's in the medical field or purchase patterns for marketing [12].

2.2 Neurons the basic of a Neural Network

NN's are based on a collection of connected units or nodes called artificial neurons, when connected together they form networks that express some computational tasks depending on the types of neural networks. The system is inspired by the nervous system, a similar manner in which the biological neural networks in the human brain processes information.

In figure 4, a biological neuron is displayed which receives messages through the dendrites from another neuron and passes it along a pathway through branches to the axon terminals where the messages will be passed on again to the next neuron.

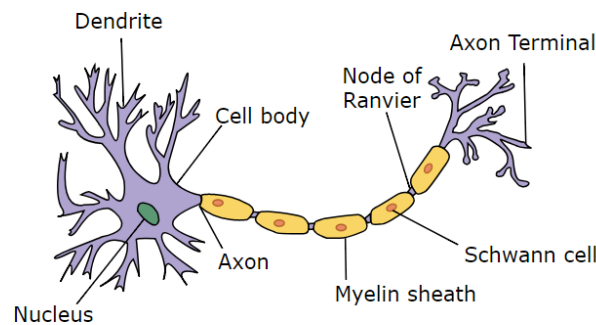


Figure 4: Sketch of a biological neuron. (Image taken from the following link: <http://www.commonswikimedia.org/wiki/File:Neuron.svg>)

The computational neuron that is shown in figure 5 can have a direct input from another neuron or it can receive the input from an external source where the input data can be a binary, integers or floating point values. Along the way to the output, each input is multiplied with a continuous valued weight as well as added biases. The sum of the weighted input then gives a single output depending on the threshold activation function [13] (better discussed in Activation functions chapter 4.1).

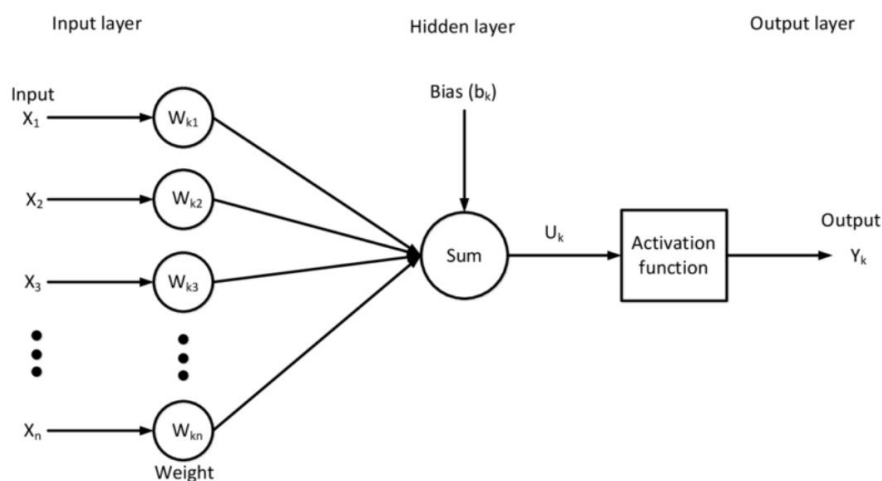


Figure 5: Computational neuron. (Image taken from the following link: http://www.researchgate.net/figure/ANN-structure-of-a-nonlinear-model_fig25_316877711)

2.3 Neural Network structure

A Neural Network is simply multiple neurons connected together in layers. Feedforward NN's are the first and simplest type where we have an input layer, hidden layer and an output layer of neurons as seen in figure 6 [14].

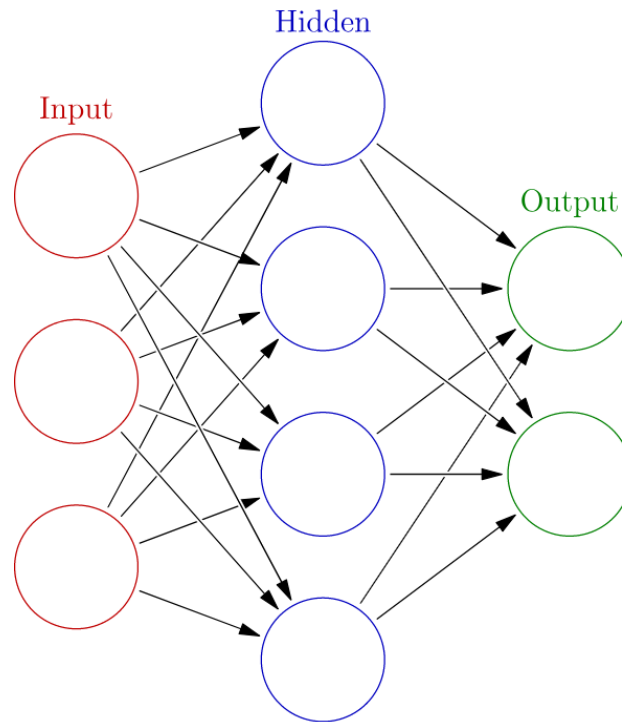


Figure 6: Example of a neural network having an input layer, hidden layer and an output layer.

(Image taken from the following link:

http://www.commonswikimedia.org/wiki/File:Colored_neural_network.svg)

Each layer consists of one or more nodes or neurons, within each layer they do not interact directly with one another. The lines between the nodes indicate the flow of information from one node to the next.

This flow of information from the input layer is copied and sent to the nodes of the hidden layer, the data in the hidden layer is multiplied by weights which are just a set of fixed numbers stored in the model, the information is then summed up and finally passed through activation (see Activation functions chapter 4.1) to make an output [15][13].

3 Convolutional Neural Networks

Convolutional Neural Networks also known as CNN's or ConvNet's are a specific type of neural networks used for supervised and unsupervised learning. The fundamental setup is the same as in a typical NN, having an input layer, output layer and some hidden layers in the middle. With layers made from neurons with the learnable weights and biases.

3.1 Convolutional Layers

What sets the CNN apart from another NN is the way that the architecture of it makes the explicit assumption that the inputs are images. The forward function of data is therefore encoded with certain properties, these properties make it more efficient to implement and reduces the number of parameters in the network significantly[16].

The input for the CNN is taken in as a tensors or matrices representing the image resolution. The convolution layer comes after that which extracts the features from the input. If the image is for example a small coloured 6×6 pixels image, it has a height of 6 pixels, width of 6 pixels and the depth of 3 channels representing each colour scheme (RGB). The convolution operation is done with a filter also known as a kernel in the convolution layer. In this example, there is a $3 \times 3 \times 3$ filter that multiplies its values element-wise with the original input matrix and then sums them up, this is known as a 3×3 convolution because of the filter size. This convolution operation is performed over the input matrix by sliding the kernel one pixel at a time to the right until it cannot go any further, then it begins on the left side again only one pixel lower, this continues until the operation has completed the process for the whole matrix. Part of the convolution is shown in figure 7. The number of steps that the filter takes over the image is called a stride, the stride in the example below is a 1×1 stride.

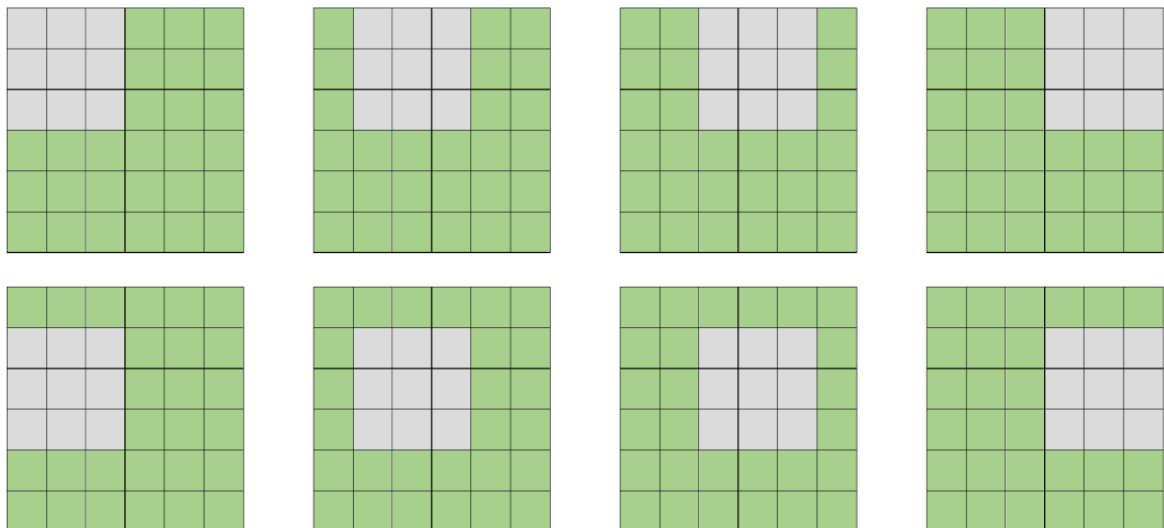


Figure 7: Shows the steps of a convolutional layer operation.

This example of a convolution operation will result in a 4×4 matrix as we can see on figure 8 and is called an activation map or feature map that is used to detect certain features in the image such as horizontal and vertical lines that will help with predicting accurate classification. These filters are not hard-coded but is a learned process over training.

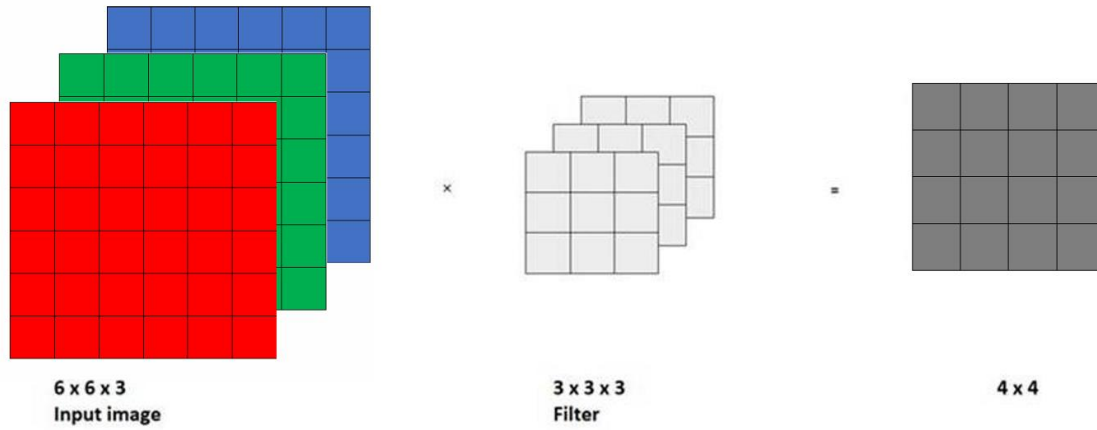


Figure 8: Shows an input image with a convolutional filter of size $3 \times 3 \times 3$ which outputs a 4×4 matrix feature map.

Example for figure 8 calculations for the output feature map $= (n - f + 1) \times (n \times n \times nch) \times$ numbers of filters. Input matrix is represented by a $(n \times n \times nch)$ where n is the height and width of the input matrix and nch signifies the depth of the image (RGB) [17]. The kernel is represented by $(f \times f \times nch)$ where f is the height and width of the kernel and nch the depth. The output of the feature map results $(6 - 3 + 1) \times (6 - 3 + 1) \times 1 = 4 \times 4 \times 1$.

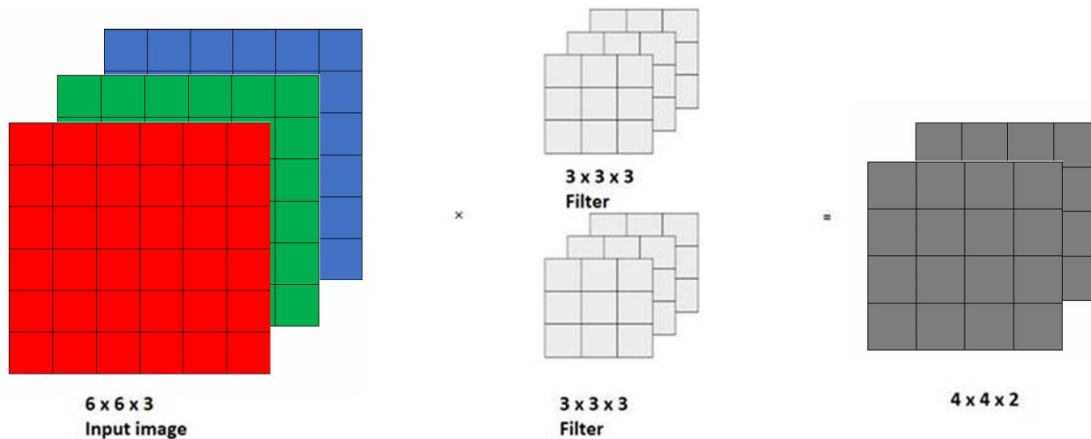


Figure 9: Shows an input image with a convolutional filter of size 3×3 which outputs a $4 \times 4 \times 2$ matrix feature map.

Normally there are multiple filters to detect many features in an image, when two are used as in figure 9 the output will be a $4 \times 4 \times 2$ stacked layer $(6 - 3 + 1) \times (6 - 3 + 1) \times 2 = 4 \times 4 \times 2$. Important to note that last number 2 in output represents number of filters and not number of channels as is in the filter and input.

3.2 Padding

As we can see in the previous example of convolution, the input of a $6 \times 6 \times 3$ matrix is reduced to a $4 \times 4 \times 2$ matrix feature map which means reduction of spatial dimension (height and width). Padding or zero padding is a way to keep or control the spatial dimensions by padding the borders with zeros [18]. If zero padding is 1 the input image would be represented as an $8 \times 8 \times 3$ matrix, then the feature map would result in a $6 \times 6 \times 2$ matrix, preserving the spatial size for input and output of the convolution.

3.3 Fully connected layer

The fully connected layer is different than the convolutional layer where the spatial structure is not preserved [18]. All the information from the convolution is flattened out to a one-dimensional vector where all the neurons in the fully connected layer are connected to all the activations in previous layers of the model. Together with an activation function such as softmax or sigmoid (Activation Functions chapter 3.1) the fully connected layer uses the previously obtained features from the convolution to determine the correct classification of an image.

3.4 Architecture of Convolutional Neural Networks

Over the years, architecture of CNN has changed with better and deeper designs, from constantly improving CPU power to vast collection of GPU's connected together via internet for considerable impact on image classification and object detection. Below are shown a few successful architectures of CNN over the last 20 years most of them being winners from the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [19], a competition of classifying 1000 different classes with a dataset of over one million images.

LeNet-5 (1998)

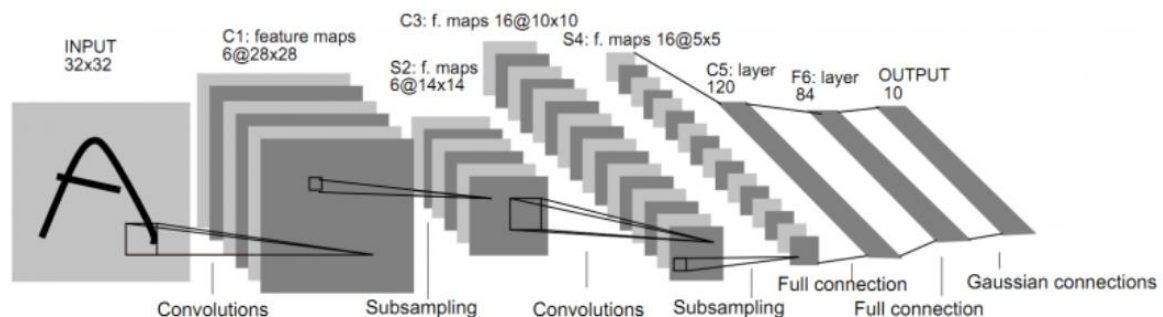


Figure 10: Shows the architecture of LeNet-5 developed in 1998. (Image taken from the following link: <http://www.yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf>)

LeNet-5 is a 7-level Convolution Network shown in figure 10 designed for handwritten and machine-printed character recognition. The model's input is a 32×32 pixel image, the model has the three fundamental layer types: Two Convolutional Layers, Two Pooling Layers and a Fully-Connected layers with Tanh and Sigmoid activations [20].

AlexNet (2012)

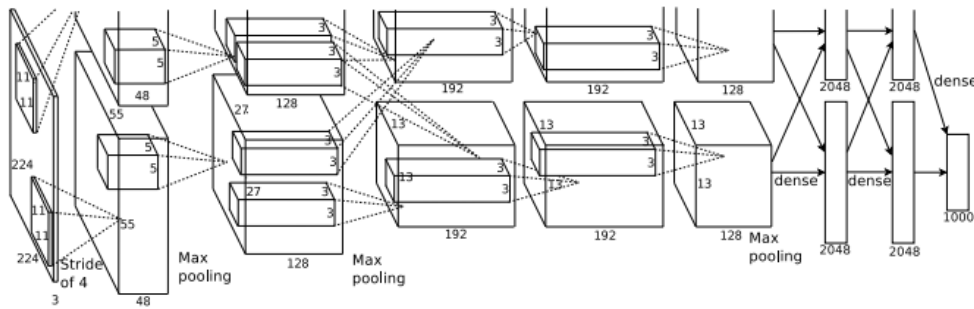


Figure 11: Shows the AlexNet architecture from 2012. (Image taken from the following link: <http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf>)

The ground-breaking AlexNet shown in figure 11 that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 is a much deeper network than LeNet-5 consisting of 7 hidden layers, 650,000 neurons and 60 million parameters with max pooling, dropout, data augmentation and ReLU activations [21].

Inception (2014)

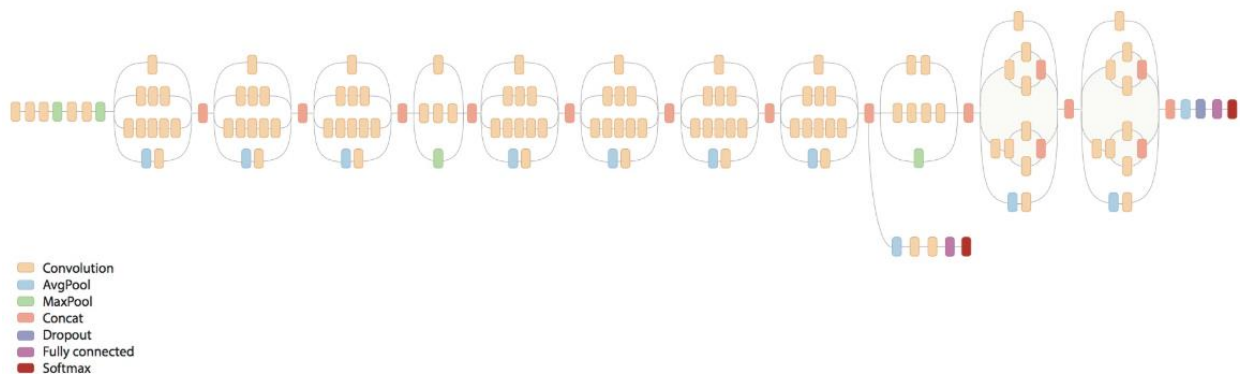


Figure 12: Shows a revised architecture of Inception developed by the Google team. (Image taken from the following link: <http://www.adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html>)

A ConvNet called Inception shown in figure 12 from Google won the ILSVRC 2014 competition is inspired by LeNet-5. It is a model with much depth and width while trying to keep computational cost at a minimum. The input consists of a 224 x 224 RGB images and all convolutions are activated by rectified linear activations. In the last inception module global average pooling is used instead of Fully Connected layer [22].

ResNet (2015)

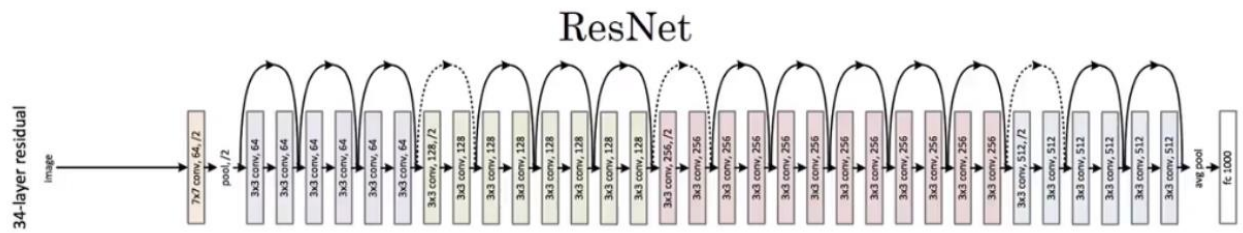


Figure 13: Shows the ResNet architecture from Microsoft. (Image taken from the following link: <http://www.arxiv.org/pdf/1512.03385.pdf>)

Residual Neural Networks (ResNet) is a ConvNet from Microsoft that won the ILSVRC 2015 competition. The model uses 224 x 224 randomly cropped images, augmentation and batch normalisation along with SGD activations on minibatches. The model introduces a shortcut connection that skip one or more layer it is a way of identity mapping where outputs are added to outputs of the stacked layer as shown in figure 13 [23][24].

4 Important features of the model

The chapter involves the selections of important functions of the model such as activations, pooling, overfitting and ways of minimizing chances of over and underfitting with dropout and early stopping. The construction of the dataset is as well covered along with data augmentation, pre-processing and splitting the data to test, training and validation sets. The architecture of the model that achieved the best results is displayed and covered in detail along with the programming environment.

4.1 Activation Functions

Activation functions or transfer functions play a vital role in Neural Networks by converting an input into an output [13]. The function decides which computational neuron fires/activates and which one does not and with that carries on useful information that will ultimately decide which category an image belongs to.

Many activation functions are available, linear and non linear, the most common one's being: Linear function, Sigmoid, Tanh, ReLU and Leaky ReLU.

The sigmoid function has a range $[0, 1]$ as shown in figure 14. It was a popular activation function for CNN's but has been replaced in modern models because of a vanishing gradient problem and a slow convergence.

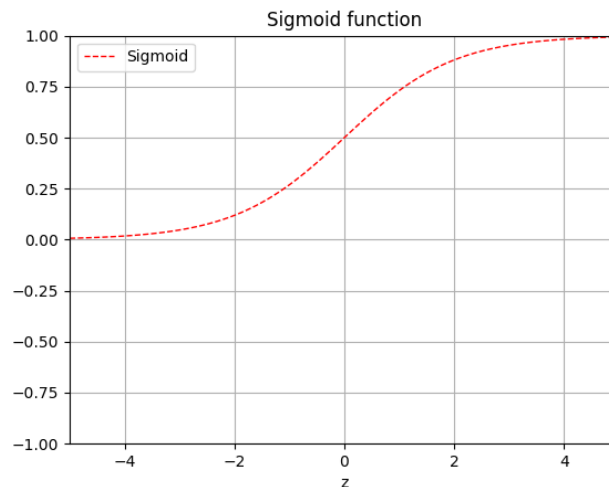


Figure 14: Plot of a sigmoid activation function $S(x) = \frac{e^x}{1 + e^x}$)

For the top model a rectify linear unit known as ReLU activation is used on all the hidden convolutional layers. ReLU is a widely used non linear activation function for CNN's. There are advantages in using ReLU which has a range of $[0, \infty]$ as seen in figure 15.

Firstly, ReLU is a computationally easy and effective activation function compared to sigmoid activation making the training phase much faster. It also partially tackles the problem with vanishing gradient. As well as increasing sparsity resulting in better representation [25].

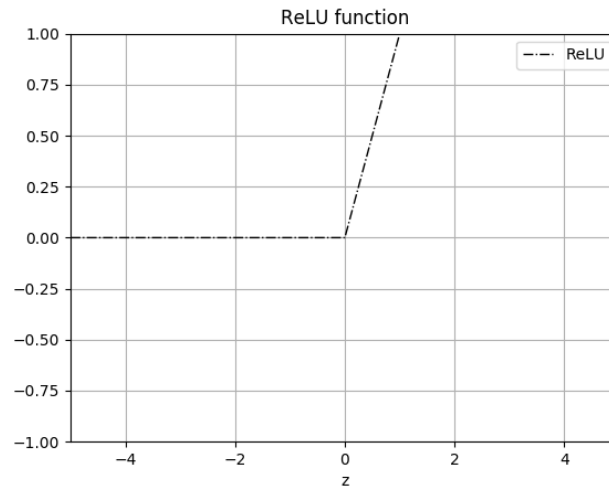


Figure 15: Plot of a ReLU activation function $R(x) = \max(0, x)$

For the top model there are two fully connected layers used, the first one having an ReLU activation function and the second one having a Tanh activation function which is a type of hyperbolic tangent similar to the sigmoid function but varies in the range $[-1, +1]$ and steepness of gradient as shown in figure 16.

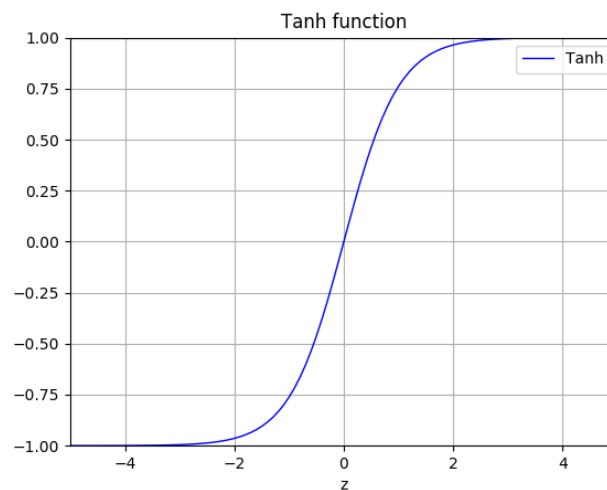


Figure 16: Plot of a Tanh activation function $\tanh(x) = \frac{2}{(1+ e^{(-2x)})} -1$

Softmax is a type of sigmoid function that is used on the output in fully connected layer. It is a way of forcing the outputs of a neural network to sum the probability's up to one to represent a probability distribution of an image into a single labelled class.

4.2 Pooling

Pooling operations make up another important building block in CNNs, there are a few types like average pooling, L2-norm pooling and max pooling [26]. For the top model max pooling is used that directly follows some of the convolution layers. As seen in figure 17 a demonstration of a 4×4 matrix is reduced to a 2×2 matrix by a 2×2 max pooling filter with a stride of 2, where the highest number in each colour is passed on forward with the most relevant information.

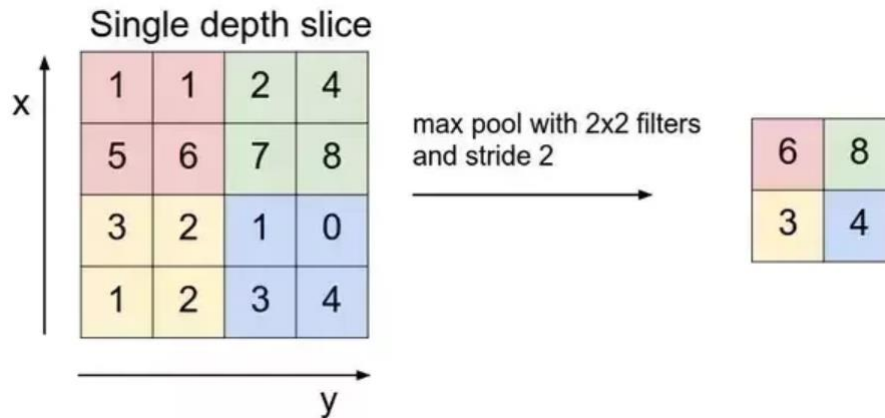


Figure 17: Example of max pool 2×2 filter on a 4×4 matrix. (Image taken from the following link: http://www.en.wikipedia.org/wiki/File:Max_pooling.png)

This max pooling operation decreases the computation cost in the model's training phase by dropping the number of parameters to learn. It also makes the features it detects more robust by passing on the most relevant information.

4.3 Overfitting and Underfitting

Overfitting and underfitting are the two most common errors that materialise in neural networks during the training phase of the data. What is desired is a thin line between the two, so the network model can predict or generalise with respectable certainty which category an unseen image belongs in.

Underfitting occurs when a model performs poorly on the training set, that is it has a poor percentage of accuracy that will lead to bad generalisation on unseen images in the test phase. Underfitting is not a problem that is hard to overcome, for it is noticeable problem that transpires in the training phase. Fixing it requires the model to be tested with different algorithm to get the test/validation accuracy of the network up, running the training phase for a longer time and/or tweak some parameters of the convolution layers [27].

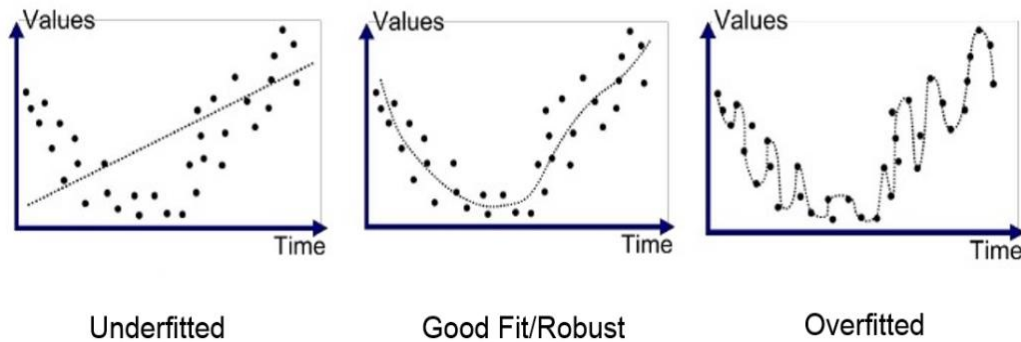


Figure 18: Shows how under and overfitting looks on a graph. (Image taken from the following link: <http://www.medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76>)

Overfitting is a much more common problem than underfitting and can be a difficult one to correct. Overfitting occurs when a model performs well on the training/validation set but poorly on the test set of unseen images. The reason can be that the dataset is small compared to the training time, it can transpire to some features of trained images to be learned, but that are not really a factor for prediction of images that are unseen. It is basically taking in the noise or random fluctuations and fitting it with the data instead of the underlying relationship between the main features in each category [28]. The difference between over and underfitting is shown in figure 18, underfitting is shown in the first image where the line does not fit the points in the graph. The image on the right side on the other hand fits all the points but is too specific to the data it is training on and can therefore not generalize well on unseen images, the middle image is what is desired where the line is balanced with the points.

4.4 Dropout

Dropout is one of the most effective way to control overfitting. The dropout function randomly disables or drops out units as well as their connections from the network as shown in figure 19. By doing that, some neurons will not contribute to the forward pass as well as the backpropagation of the model. Consequently, the model gets smaller which is computationally easier but the main benefit for implementing dropout is that by not having the presence of neurons all around, it prevents them from co-adapting to one another. In other words, dropout forces the network to learn more robust features that will lead to better generalisation [29].

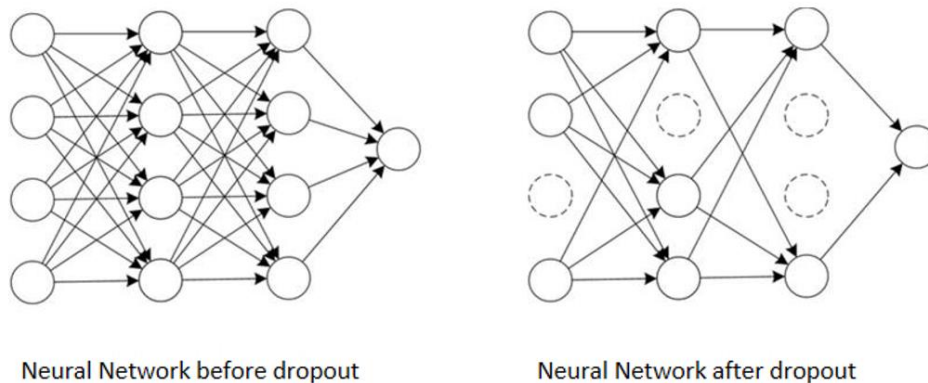


Figure 19: Shows how dropout randomly drops neurons connections. (Image taken from the following link: http://www.researchgate.net/figure/Dropout-neural-network-model-a-is-a-standard-neural-network-b-is-the-same-network_fig3_309206911)

Dropout is used in the range of 0-1 where the neurons are multiplied with the output in that range. It is a process of trial and error to find out what works for each model.

4.5 Early stopping

Early stopping is another tactic of regularization used to prevent overfitting and unnecessary computation, it is used in the training phase and stops the model from continuing training a model when it has reached its “best” accuracy on a validation and/or training set [30]. The way it works is that after each epoch (one forward and backward pass of the entire training data) the validation accuracy and/or training accuracy is checked and compared to the validation accuracy and/or training accuracy in the previous epoch and if there has not been any increase between them, training will be stopped. This can be programmed in, or this can be estimated by running the program and monitoring the progress.

Usually the validation accuracy goes up and down, so it is important to set “patience”, a feature used to wait a certain time before stopping training, that is to make the model stop training if it has not increased its accuracy over the last 10 epochs for example.

4.6 Optimization

Optimization serve the purpose of minimizing the loss function by updating the model's weights and biases in the training phase. Today there are many optimization algorithms available like GradientDescentOptimizer, AdadeltaOptimizer, AdagradOptimizer, AdagradDAOptimizer, MomentumOptimizer, AdamOptimizer, FtrlOptimizer, ProximalGradientDescentOptimizer, ProximalAdagradOptimizer and RMSPropOptimizer.

This process of finding optimal (minimal) loss function is an iterative process meaning that one pass of the limited dataset is not enough. The dataset needs to be run through over and over again with the optimizers that update the weights and biases to gradually achieve the minimum loss function. Over the years gradient decent optimization algorithms have been the most popular optimization functions for NN.

An Adaptive Moment Estimation (Adam) optimization algorithm was chosen for the top model which has proven itself suitable for most problems. Furthermore, it is recommended as the default optimization method for deep learning applications in a Stanford CS231 course [30]. The algorithm is computationally efficient, has little memory requirements and is straightforward to implement.

4.7 Dataset

The foundation of a good image classifier is the quality of data in the dataset used to train the Network. In order to get decent accuracy on image predictions we need good consistent images. Today we do not have any trouble getting the images, but it is a tedious job to find the relevant ones. Even when the relevant images have been found there can be an issue if they are not labelled or structured as intended, which means most of the images are unusable or take a lot of work to get in the right format.

The dataset needed for the CNN consist of four categories of images to classify the five classes. They are aluminium cans, paper, plastic and plastic bottles.

An attempt will be made to categorise the fifth class, the general waste class. It will be attempted without images being collected. The images for the general waste class are very diversified and therefore the size and time it would take to make that dataset for the project is not enough. For that reason, a decision was made to leave it out of the image dataset and rather activate that classification by using the prediction accuracy for the other four classes. Then if a certain percentage of accuracy is not met for an image it would be sorted to the general waste category.

4.7.1 Dataset Fabrication

As previously mentioned, it is difficult to find the appropriate dataset that represent the challenges that are being confronted. In order to get the most relevant representation of real time functionality of the ATC mechanism a decision was made to build a new dataset for the categories.

The ATC mechanism has a camera chamber where pictures of the trash is analysed before it is moved mechanically to its category. The mechanism takes in a single piece of trash and therefore I need a dataset of images representing the analysis that occurs in ATC. All the images are captured with a Raspberry pi v2 camera that is positioned inside the camera chamber, where the distance from an object is 20 cm. Example of images from all categories can be seen in figure 20.



Figure 20: Example of images taken above the camera chamber. Images are from the four categorise needed to classify.

A setup was made where a simple Python program with four push button inputs to a Raspberry Pi 3 computer was constructed. Each button input was programmed to take a picture with a Raspberry V2 camera where a LED ribbon light output would be turned on while images are captured for clearer images. For each button the picture was saved to a specified folder and labelled appropriately that is as aluminium cans, paper, plastic or plastic bottles.

The images collected are saved in HD resolution (1280×720 pixels) although the images may be resized when it comes to training. Total images collected for the dataset were 4755, divided as follows:

- Aluminium cans 1000
- Paper 1582
- Plastic 1173
- Plastic bottles 1000

The aluminium cans class includes images of 330 ml and 500 ml cans from Pepsi, Coca Cola, Monster and many more in crumpled and whole condition.

The paper class includes images of magazines, newspapers, white papers, paper cups, napkins and soft paper in any condition (shredded, wrinkled and whole).

The plastic class includes images of see through bags, coloured bags, candy wrappers and plastic cups along with hard see through plastic in any condition (shredded, wrinkled and whole).

Plastic bottles class includes images of 500 ml and 1L plastic bottles from Pepsi, Coca Cola and many more, in crumpled and whole condition.

The dataset for plastic and paper categories were made bigger than the two other categories because of the similarities between them, which could cause problems distinguishing them apart with the CNN.

4.8 Data pre-processing

Popular datasets for image classification like MNIST dataset [31] have 70000 images of handwritten digits and CIFAR-10 dataset [32] has around 60000 images for ten classes of classification. My dataset of roughly 5000 images for four classes is not a large dataset, but it can be a good foundation to build upon. The differences between my dataset and previously mentioned datasets besides the size is that I have fewer classes of classification, larger images for training and a controlled environment of image capturing. The images for the dataset are resized to numerous sizes for later evaluations purposes. From a 450×450 RGB images to 50×50 grayscale images compared to 32×32 RGB images in the CIFAR-10 dataset and 28×28 grayscale images in the MNIST dataset.

4.8.1 Dataset Augmentation

By image augmentation the dataset can be enlarged. There are many options to do this [33], in my process I focused on three methods. By using rotation, flip and blur, where rotation takes a copy of every picture and rotates it by a set angle or random angle. Flip or mirror where a copy is taken of every image and flipped from right to left or upside down and blur where a copy is taken of every image and Gaussian Blur filter is implemented on the copy. Each of these methods by itself doubles the dataset size. Image augmentation is done in real time before convolution and training starts and is not done on the test set.

4.8.2 One hot encoding

When working with labelled data it is necessary to get the images in the right format for algorithms cannot work with categorical data directly. One hot array is a way to deal with that by converting categorical data into vectors of 0's and 1's [34]. The vectors length is equal to the number of categories, as for my case I have four categories and therefore the vector's length is four where each element in the vector corresponds to one of the four labels.

Table 1: Showing one hot encoding on the four classes.

Aluminium cans	1	0	0	0
Paper	0	1	0	0
Plastic	0	0	1	0
Plastic bottles	0	0	0	1

Labelled images are now represented as one hot encode vectors as we can see in table 1, aluminium cans as [1,0,0,0], paper as [0,1,0,0], plastic as [0,0,1,0] and plastic bottles as [0,0,0,1].

4.9 Training, Validation and Test sets

Training set is part of the labelled data that the model runs over and over to learn algorithms by constantly updating the weights of the classifier with backpropagation.

Like the training set the validation set consist of labelled images that are used to validate the model during the training phase and to tune the hyper parameter of the classifier.

The test set is entirely separated from the training set and has nothing to do with the construction of the model itself. It is the final assessment of efficiency on the model by running unlabelled and previously unseen images through the trained network.

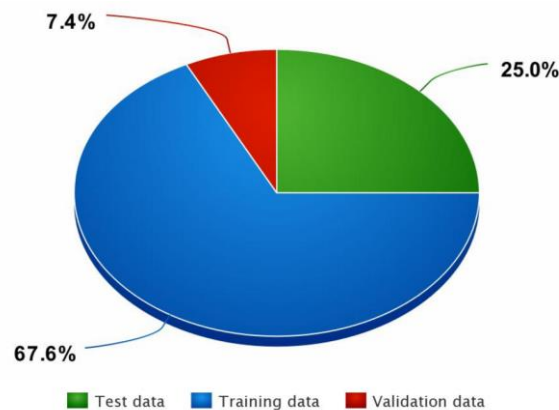


Figure 21: Image proportion over the entire dataset, containing training, validation and testing data.

Splitting the dataset to training set and test set is essential for evaluation purposes. I decided to split my dataset of 4755 images into 75% training and 25% testing. There are 3566 labelled images in training set and 1189 unlabelled images in the testing set. A portion of the training set is reserved for validation, that is 350 images of the training set as is shown in figure 21.

4.10 Environment

The CNN is programmed in Python programming language in a TensorFlow environment an open-source software library from Google. TensorFlow was originally developed by Google Brain Team to conduct machine learning research, it is used for numerical computation using data flow graphs. That is the main feature, expressing numeric computation as a computation graph. This framework allows you to build complex models with small operations, make gradient calculations and make each operation function evaluations at a given point. Another feature of TensorFlow that helps with visualisation, understanding, debugging and optimizing TensorFlow programs is TensorBoard [35].

Along with the TensorFlow environment I used TFLearn [36] which is a high-level API Python module for distributed machine learning. It makes some processes more accessible to use as well as having a Scikit-learn interface. A list of hardware and software is as follows:

Computer

- I5 intel 4 cores
- 8GB RAM
- GTX 1050 GPU
 - 6.1 computing capability
 - 2GB memory
 - Nvidia driver 384.111

Programming environment

- Ubuntu desktop version 16.04
- Python3
- Sublime Text3
- TensorFlow 1.2.0 / TFLearn
- Cuda Toolkit 9.1
 - cuDNN 4.0
 - Anaconda for Python

4.11 Top model

After trying several different models of variously deep CNN with different input images scaling from large RGB images to small grayscale images, a decision was made to have the input contain grayscale 400×400 pixel images. This variation ended up being the top model with the best accuracy on the test set consisting of the typical CNN architecture that has an input layer, hidden layers in the middle and fully connected layers at the end. Real time image augmentation is performed during the training phase with random blur and random rotation on images.

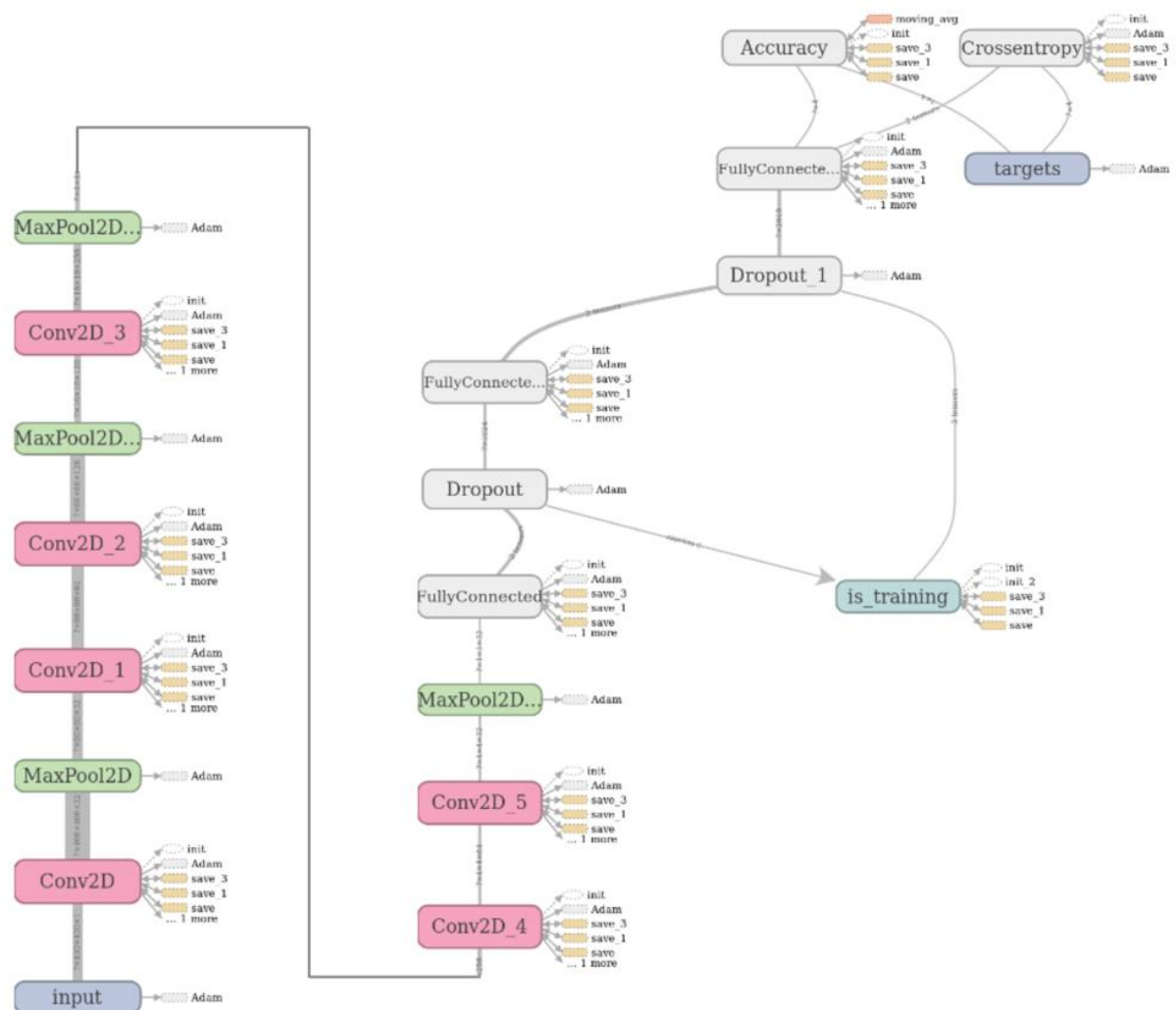


Figure 22: Flow chart of the model from TensorFlow Tensorboard revealing step by step information.

Figure 22 shows a flow chart of the model's functionality. The input takes in the images in a $400 \times 400 \times 1$ matrix which goes to the first convolution layer that has 32 filters where the filter size is 5×5 . The strides are set to $[1 \ 1 \ 1 \ 1]$ meaning that there is a thorough overlay for spatial recognition, as well as having no need for zero padding. Activation of all the convolution layers are conducted with the rectified linear unit (ReLU) function.

After the first convolution layer a max pooling function is conducted with a 5×5 filter that reduces the input to the next convolution to an $80 \times 80 \times 32$ matrix.

The second convolution consist of 64 filters also having 5×5 matrix size with a stride of $[1 \ 1 \ 1 \ 1]$. This action produces an $80 \times 80 \times 64$ matrix output to the next convolution layer that has 128 filter with the same activation and same filter sizes as before. The second max pooling layer follows that reduces the image most relevant information to an output of $16 \times 16 \times 128$ matrix. The fourth convolution has 256 filters with 5×5 kernel size, that outputs a $16 \times 16 \times 256$ matrix that goes to the third max pooling function that decreases the information further to a $4 \times 4 \times 256$ matrix.

The fifth and sixth convolution layers with 64 and 32 filters come after that before the last max pooling operation that gives an output of $1 \times 1 \times 32$ matrix to a the fully connected layers.

There are two fully connected layers with different activation functions. The first fully connected layer is activated by ReLU with 1024 nodes and a dropout rate of 0.6 and the second fully connected layer is activated by a Tanh function which flattens the information to 2048 nodes and has a dropout rate of 0.7 to prevent the model from overfitting. Finally, we have the softmax activation for the classification for the 4 classes needed. The regression is performed by Adam optimizer with a learning rate of $1e-4$.

The training set contains 3216 images so the validation for the training phase was set around 11% or 350 images. Batch size is 12 which is low but is needed for computational reasons, this is a compromise between learning rate and batch size. The batch size is the number of images that are taken through training at a given time, because the whole dataset cannot be passed through the network all at once. The number of batches needed to go through the dataset ones is $3216 \text{ images} / \text{batch size } 12 = 268 \text{ batches} = 1 \text{ epoch}$. The number of epochs is set at 40 a number found after some testing with early stopping.

5 Results

In this section, the results of the top model are displayed, which is the accuracy and loss functions of the model in training phase for the training set and validation set. The test sets were analysed with confusion matrices for good visualisation representation on results and comparisons are done on the second-best model to the top one. Smoothing on the graphs are applied that need to be taken into consideration.

5.1 Accuracy results of the top model

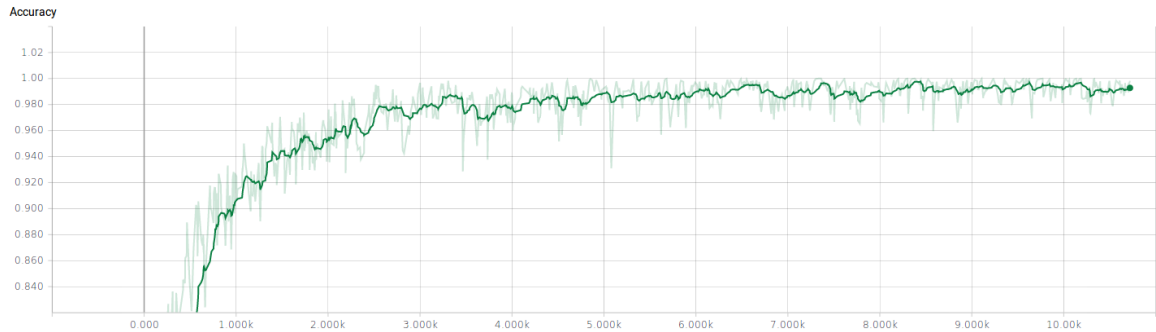


Figure 23: Showing raw accuracy results on training set.

Results on the training set accuracy can be observed in figure 23 where the Y axis represents the percentage of accuracy and X axis represents the steps. By looking at the graph it is visible that the accuracy increases significantly for the first 3000 steps and then gradually increases and ending with a 0.9982 (99.82%) accuracy.

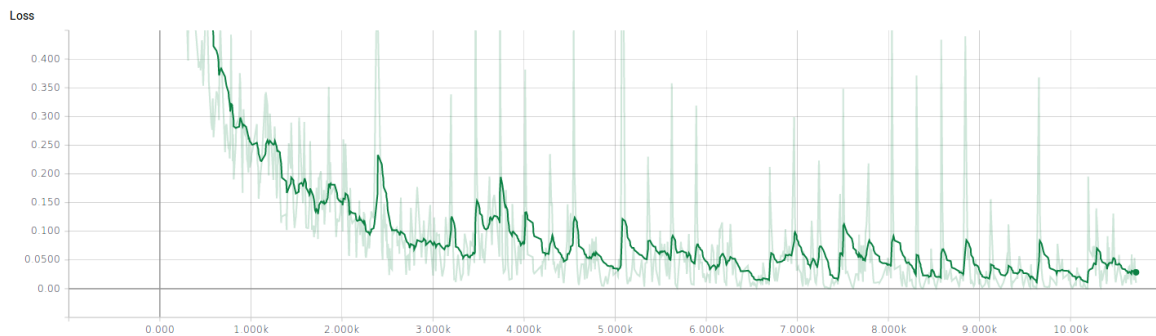


Figure 24: Results shown for loss function on training set.

Results on the training set loss function can be observed in figure 24 where the Y axis represents the percentage of loss and X axis represents the steps. By looking at the graph it is visible that the loss function decreases significantly for the first 3000 steps and then gradually approaches zero, ending with a 0.01027 (1.027%) loss.

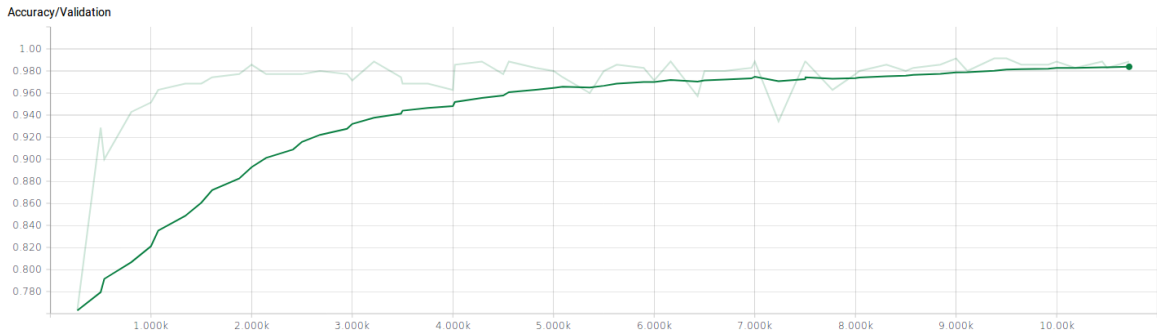


Figure 25: Validation accuracy results shown.

Results for the validation set accuracy is shown in figure 25, the Y axis represents the accuracy and X axis represents the steps. By observing the graph, it is clear that the main improvements occur over the first 2000 steps ending with a 0.9886 or a 98.86% accuracy on the validation set.

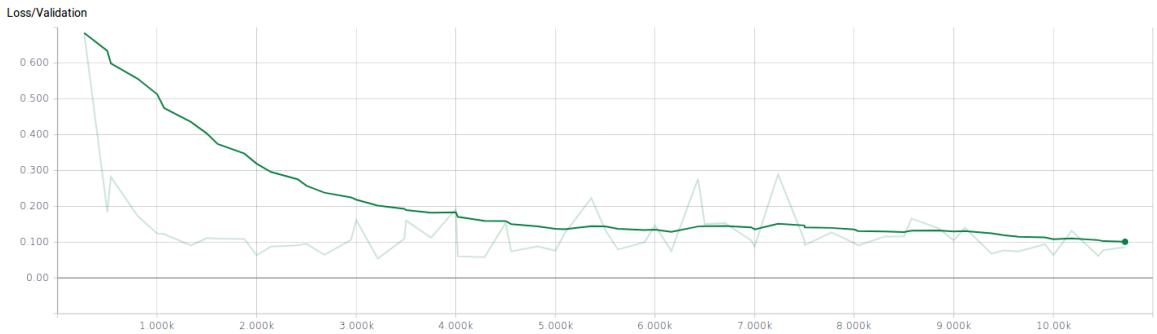
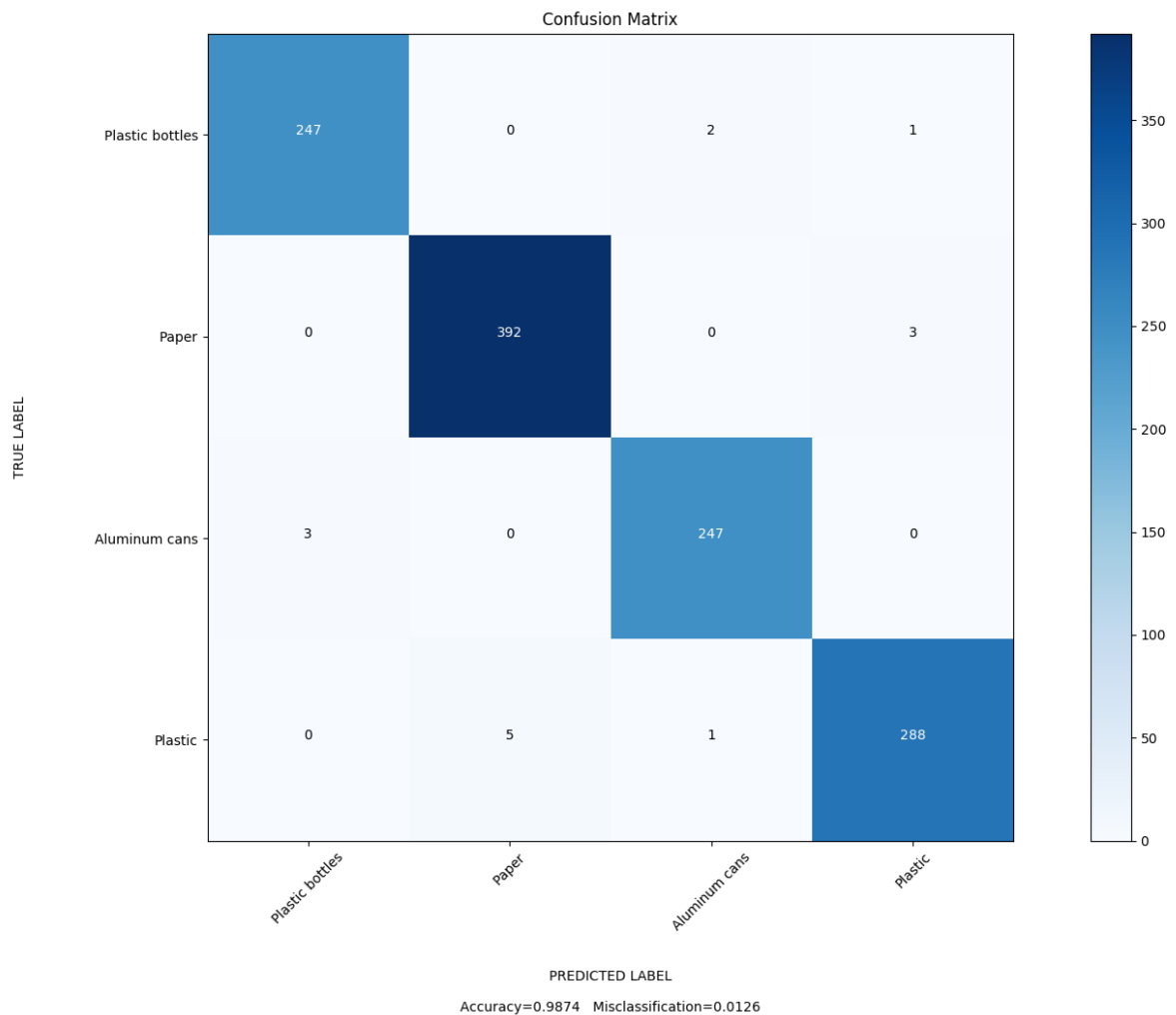


Figure 26: Validation set loss shown in graph.

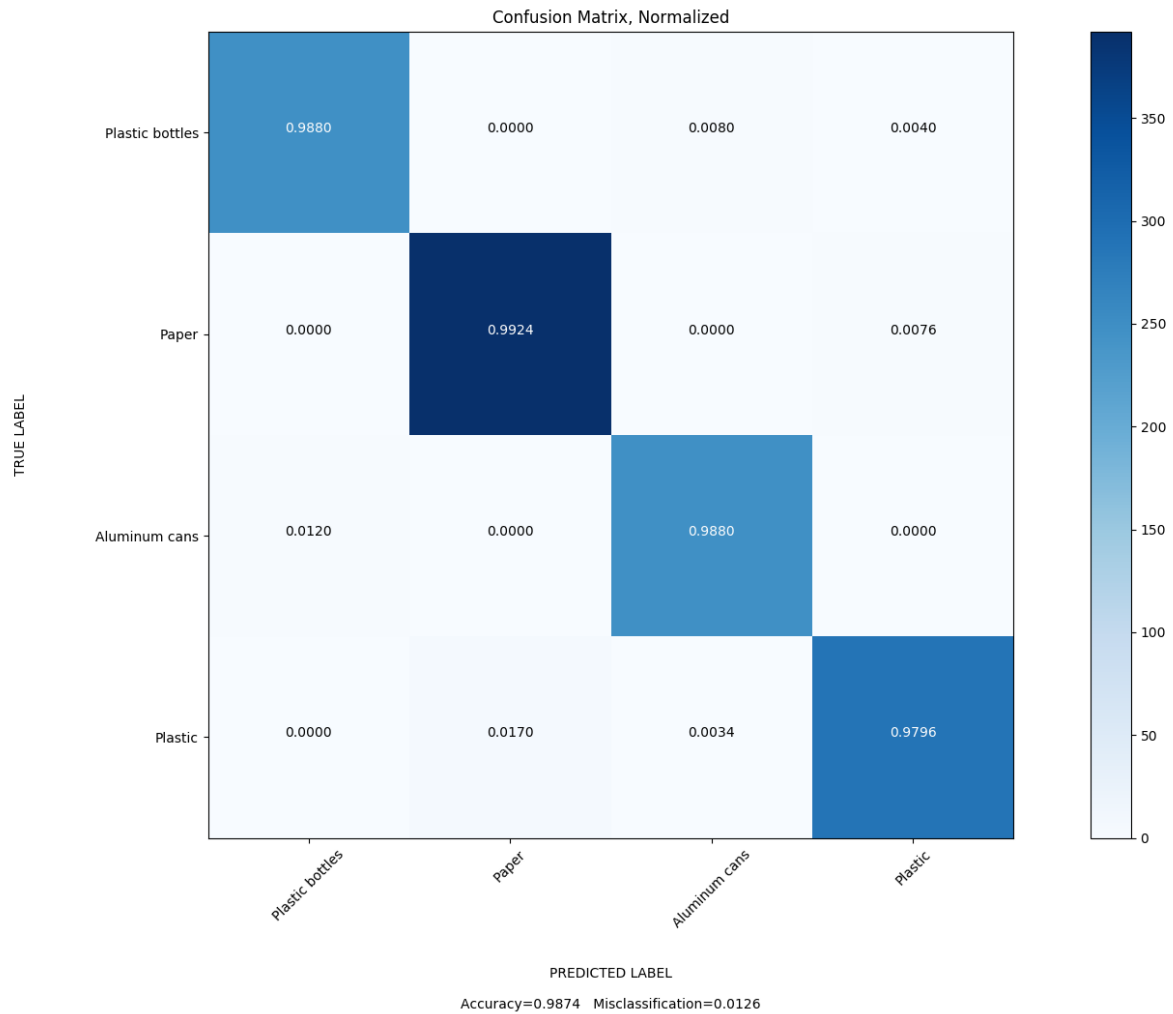
The process of the loss function on the validation set can be observed in figure 26, the Y axis represents the accuracy and X axis represents the steps. On the graph it shows how drastically the loss reduces in the beginning and ends up being 0.08558 or 8.558%.

Table 2: Confusion matrix for test set results plotted with scikit Learn.



Results from the test set were correctly predicted 1174 times out of the 1189 images giving a 98.74% accuracy. Most of the classification errors are between the plastic and paper categories as we can see in the confusion matrix in table 2 where the Y axis represents the true labels and the X axis represents the predicted labels. Out of 395 images in the paper category three of them were misclassified to be plastic and in the plastic category five out of the six misclassification errors were thought to be paper. There are many resemblances between the two categories so in advance it was expected that most prediction errors would be in the paper and plastic categories. Three errors were made in the plastic bottle category, two of the images were predicted to be aluminium cans and one to be plastic. The aluminium can category also had three predictions wrong, all of them predicted to be plastic bottles.

Table 3: Confusion matrix for normalized test results plotted with scikit Learn.



Looking at the normalised confusion matrix in table 3, where Y axis represents the true labels and the X axis represents the predicted labels, we can see the prediction accuracy on each category. The lowest accuracy of 97.96% is in the plastic classification where 1.7% were misdiagnosed as paper. The best classification accuracy was achieved in the paper category with 99.24% accuracy. In the aluminium cans category 98.8% classification accuracy was achieved and plastic bottle category ended up with the same classification accuracy of 98.8%.

5.2 Accuracy results of the second-best model

The second-best model is similar to the top model in having six convolution layers and containing two fully connected layers with Tanh and ReLU activations. However, the convolution layers in this model have either 32 or 64 filters and max pooling is applied directly after each convolution layer. Image augmentation is done somewhat different and the dropout rate of the fully connected layers are both set to 0.7. Below are the results for the accuracy and loss function on the training and validation sets, the test set results are presented with a confusion matrix. Smoothing on the graphs are applied that need to be taken into consideration.

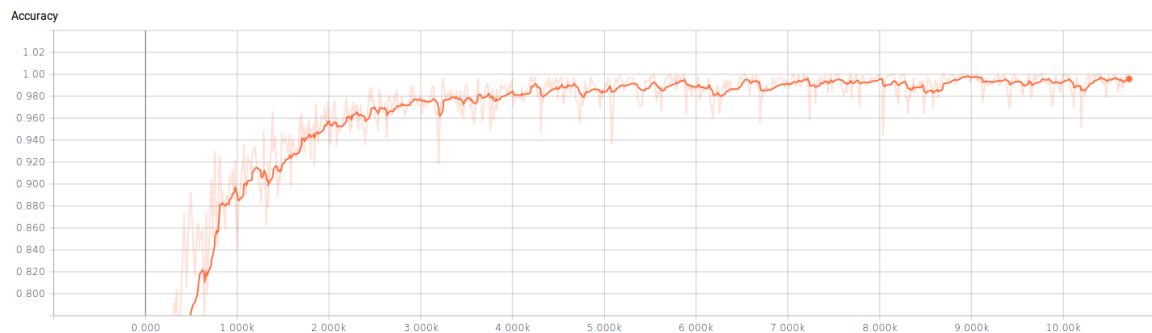


Figure 27: Showing raw accuracy results on training set.

Results from the training set accuracy can be observed in figure 27 where the Y axis represents the percentage of accuracy and the X axis represents the steps. By looking at the graph it is visible that the accuracy increases significantly for the first 3000 steps, then gradually increases and ending with a 0.9917 (99.17%) accuracy.

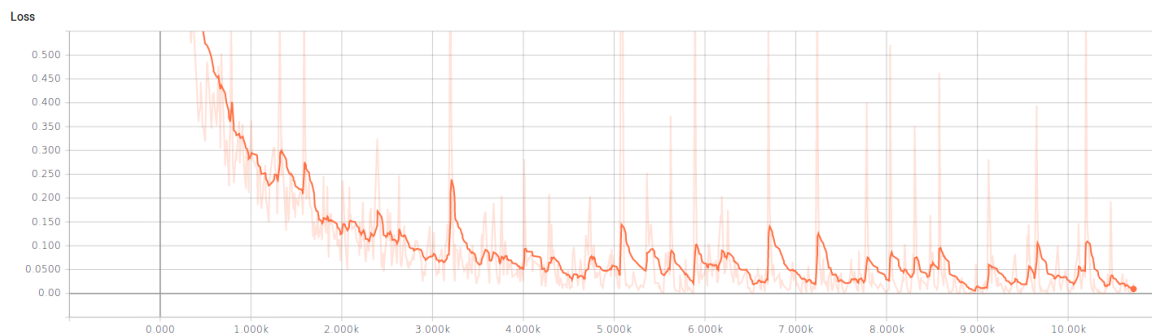


Figure 28: Results shown for loss function on training set.

Results on the training set loss function can be observed in figure 28 where the Y axis represents the percentage of loss and the X axis represents the steps. By looking at the graph it is visible that the loss function decreases significantly for the first 3000 steps and then gradually approaches zero, ending with a 0.0214 (2.14%) loss.

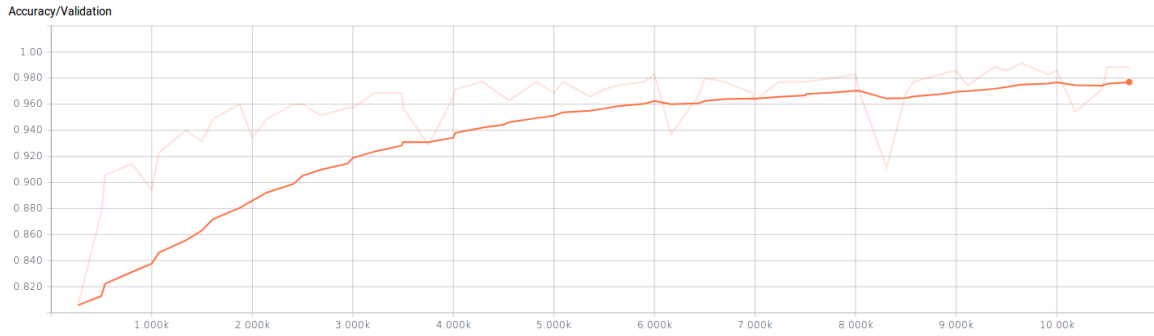


Figure 29: Validation accuracy results shown.

Results from the validation set accuracy is shown in figure 29, the Y axis represents the accuracy and the X axis represents the steps. It shows a gradual increase in prediction precision ending with 0.9886 or 98.86% accuracy.

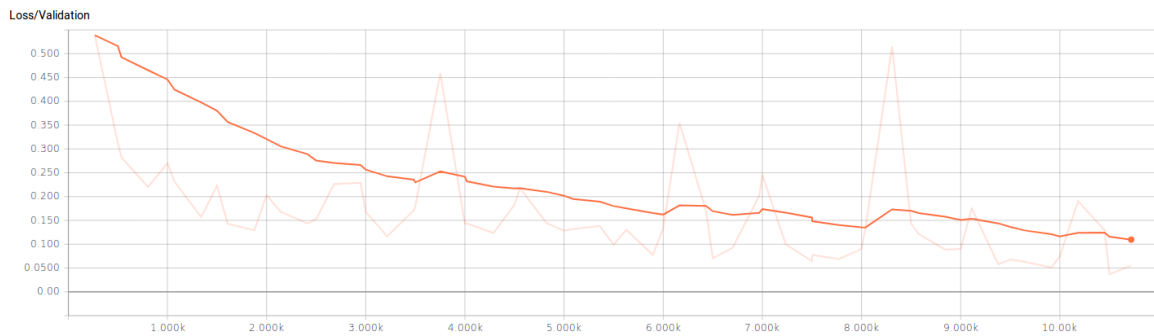
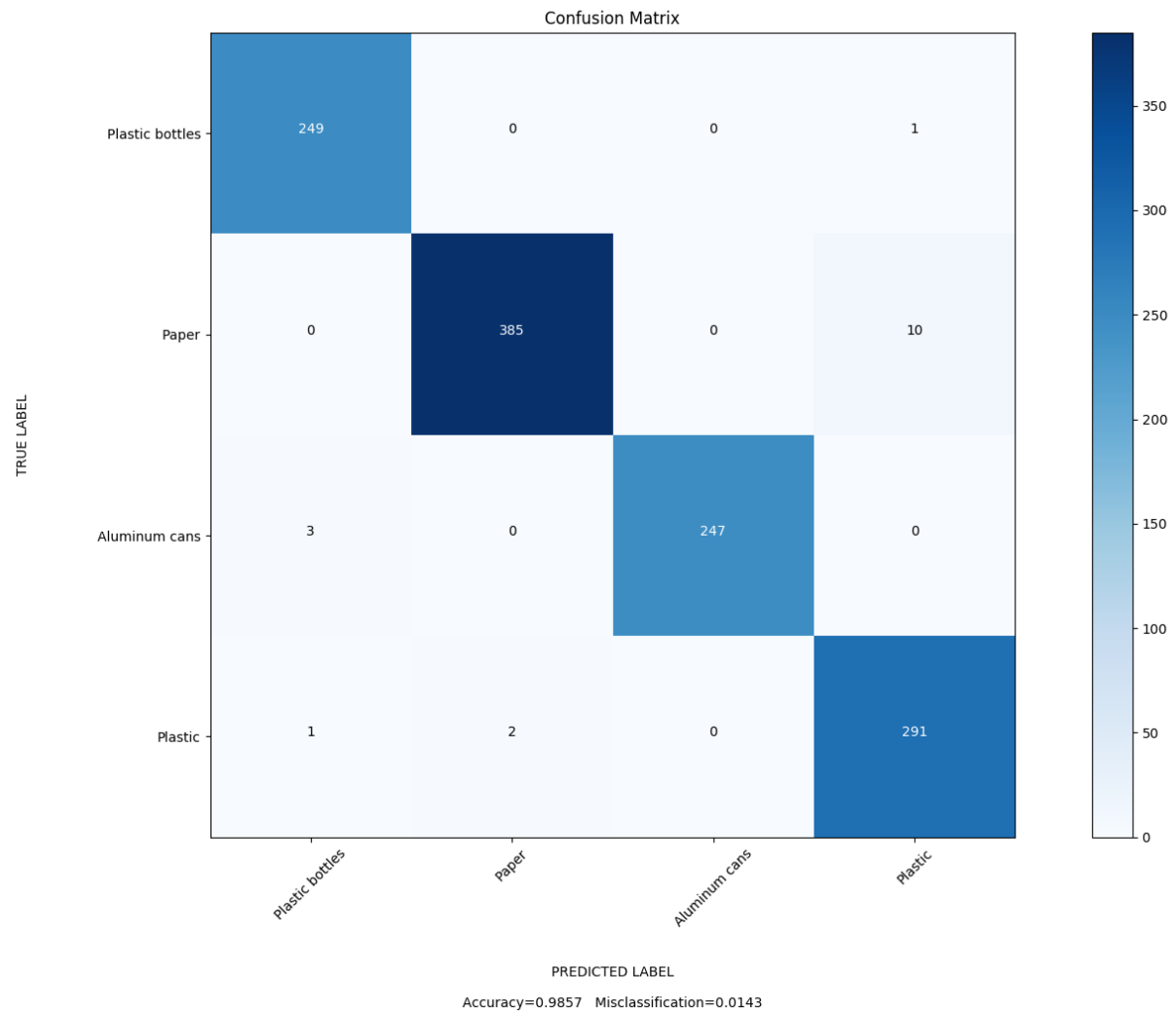


Figure 30: Validation set loss.

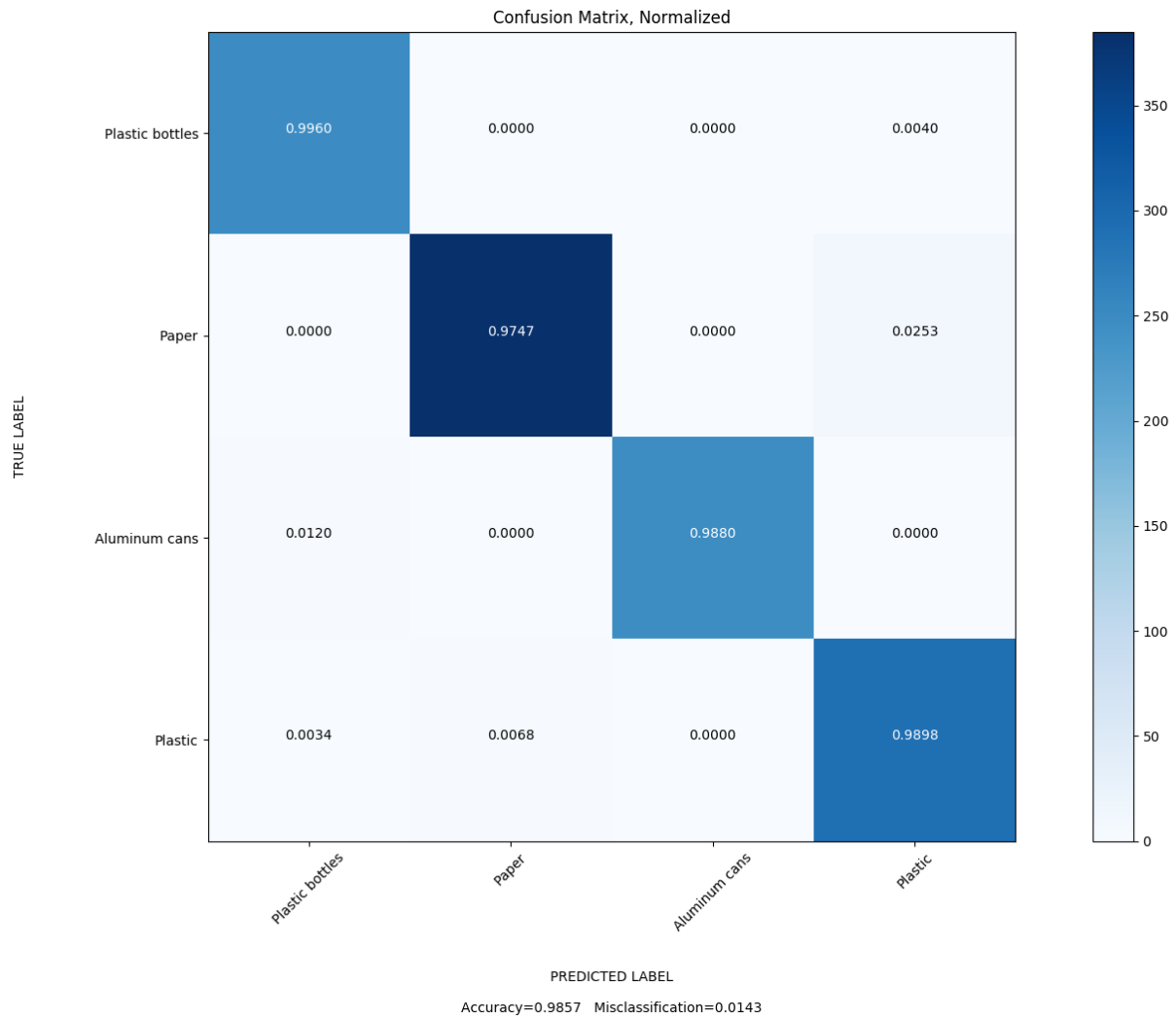
The process of the loss function on the validation set can be observed in figure 30, the Y axis represents the accuracy and X axis represents the steps. The graph shows how the loss reduces considerably in the beginning and ends up being 0.05484 or 5.48%.

Table 4: Confusion matrix for test set results plotted with scikit Learn.



Results from the test set were correctly predicted 1172 times out of the 1189 images producing a 98.57% accuracy. Most of the classification errors were between the plastic and paper categories as we can see in the confusion matrix in table 4 where the Y axis represents the true labels and the X axis represents the predicted labels. Out of 395 images in the paper category, ten of the images were predicted to be plastic. There are many similarities between the two categories so in advance it was expected that most prediction errors would be in the paper and plastic categories. Out of the 294 images of plastic two of them were thought to be paper and one predicted to be a plastic bottle. The accuracy of plastic bottles was very good with only one of the predicted as plastic. Out of the 250 images of aluminium cans, only three of them were misdiagnosed as plastic bottles.

Table 5: Confusion matrix for normalized test results plotted with scikit Learn.



Looking at the normalised confusion matrix table 5 where the Y axis represents the true labels and the X axis represents the predicted labels, we can see the prediction accuracy on each category. The lowest accuracy of 97.47% was in the paper classification where 2.53% were misdiagnosed as plastic. The best classification accuracy was achieved in the plastic bottle category with 99.6% accuracy. In the aluminium cans category 98.8% classification accuracy was achieved and plastic category ended up with a classification accuracy of 98.98%.

5.3 Comparison of top and second model

Both models have the same runtime in the training phase (40 epochs) and have very similar results on the accuracy of both validation and training sets. The main different between the two of them are in the test set results and loss function. Below is the comparison on the two models for test, train and validation sets. Smoothing on the graphs are applied that need to be taken into consideration.

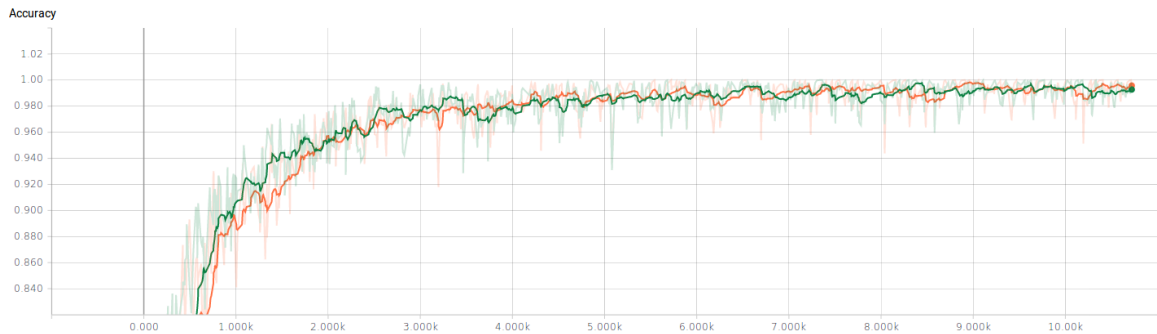


Figure 31: Showing raw accuracy comparison results on training sets.

Comparison can be seen in figure 31 where accuracy on the training set of the top model (green line) and second-best model (red line) are displayed. The Y axis on graph represents the accuracy and the X axis represents the steps. The models perform similarly where the top model ends with a 99.82% accuracy compared to a slightly less 99.17% accuracy. It is visible that the top model has a steeper learning curve in the beginning but otherwise they are similar.

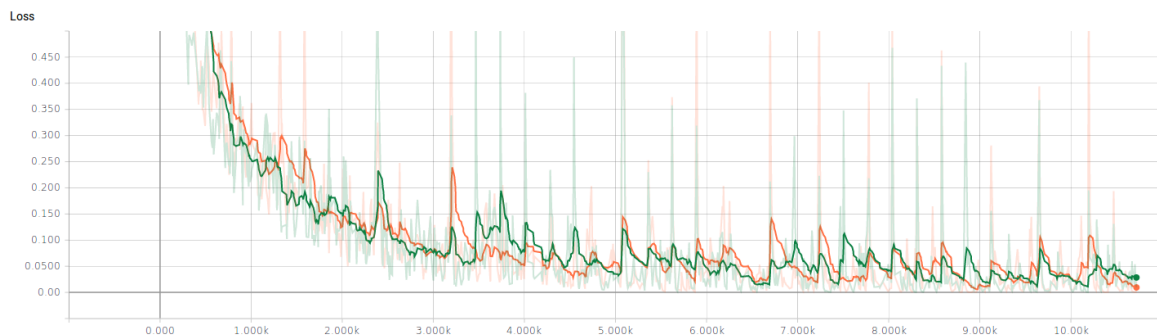


Figure 32: Results comparison shown for loss function on training sets.

Comparison on the loss function on training sets for the two models are shown in figure 31 where the Y axis represents the accuracy and the X axis represents the steps. The top model (green line) has a steeper loss in the beginning and has a better final result in a 1.027% loss function compared to 2.14% on the second-best model (red line).

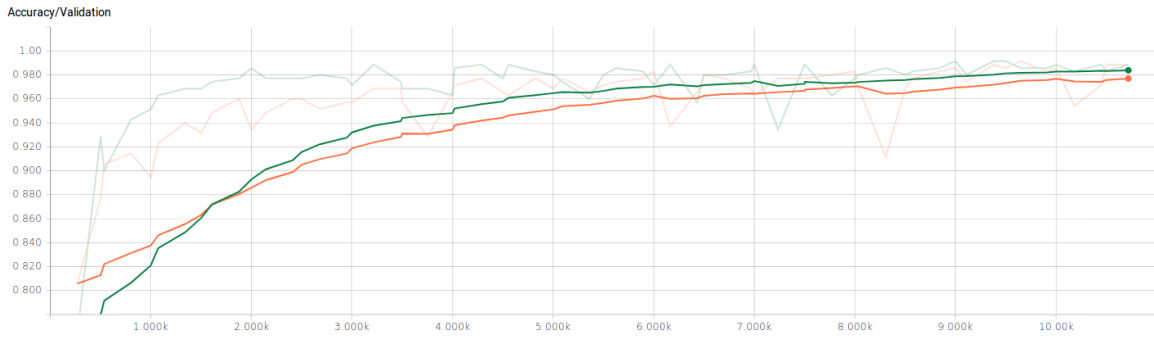


Figure 33: Validation accuracy comparison results shown.

Accuracy on the validation set for the top model (green line) and second-best model (red line) is displayed in figure 33 where the Y axis represents the accuracy and the X axis represents the steps. Both models achieve the same result with a 98.86% accuracy on the validation set. The difference is that the top model has a greater increase in the start compared to the second-best model.

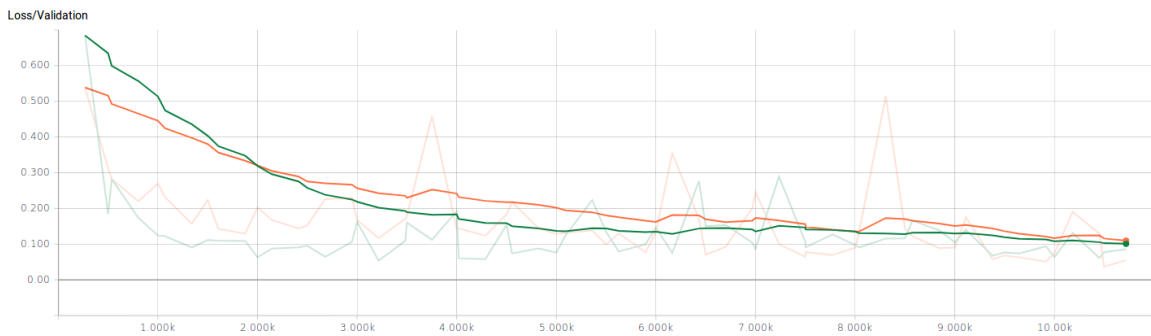


Figure 34: Validation set loss comparison shown.

Loss on validation comparison for the top model (green line) and second-best model (red line) is shown in figure 34 where the Y axis represents the accuracy and the X axis represents the steps. Better result is achieved by the second-best model with 5.48% loss compared to the top model with 8.558% loss.

Table 6: Comparison of top and second model predictions on test set.

True label	Plastic bottles	247 249	0 0	2 0	1 1
	Paper	0 0	392 385	0 0	3 10
	Aluminium cans	3 3	0 0	247 247	0 0
	Plastic	0 1	5 2	1 0	288 291
		Plastic bottles	Paper	Aluminium cans	Plastic
		Predicted label			

The difference between the two models are interesting and can be seen in table 6 where the green text represents the top model predictions and red text represents the second-best model predictions. If the paper category is observed the difference are three prediction errors from the top model compered to ten from the second-best model where all of them are predicted to be plastic. The difference can be explained with the number of filters present in the top model. The highest number of filters in one of the convolutional layers are 256, compared to 64 filters in the second-best model. With these many filters, the top model is able to detect other patterns that the second-best model is not able to. In other categories the difference is minimally better for the second-best model.

6 Conclusion

This thesis shows that a respectable classification accuracy can be achieved by implementing a Convolutional Neural Network on the four classes of the project. By creating a relatively small dataset of less than 5000 images taken in a controlled environment and using image augmentation an accuracy of 98.74% was achieved on the test set.

With several different architecture experiments the best model ended up having an input containing 400×400 grayscale pixel images, data augmentation of rotating and blurring, six convolution layers and two fully connected layers. ReLU activation was applied on the six convolution layers that had 32, 64, 128 and 256 filters. In between the convolution layers, four max pooling operations were used to down sample the matrix that sped up computation and made the detected features more robust. The fully connected layers used ReLU and Tanh activation and with the help of dropout and early stopping experiments, overfitting was limited. The results on the training and validation set resembled the outcome of the test set where the most categorical errors turned out to be in the paper/plastic category as was expected.

There is room for improvement that could involve a different network or some changes of the current model. But the groundwork for implementing a CNN model on the ATC mechanism has been achieved by the project.

6.1 Discussion and future work

The current CNN model together with the ATC mechanism can serve a role to make a better algorithm in the future. When the ATC is up and running digital copies will be made of the trash as it is sorted to the corresponding categories. These copies will be labelled images for each class that can be used for a new larger dataset. The model is limited by the quality and size of the dataset it is trained with.

The model is categorising four classes with good accuracy for plastic bottles, aluminium cans, plastic and paper. There are however additional classes that need to be categorised, for example general waste and organic waste. Large datasets need to be made for those classes because of the diversity of items in them and preferably be created in a controlled environment like the other datasets. The attempted to categories the general waste category by using the “low” percentage of accuracy on the other categories to activate the general waste category did not return the desired results. The accuracy for the image’s predictions are high, even when they are incorrectly classified. The percentage fluctuates too much to find the fine line or percentage that is suitable to classify the general waste class.

The input selection of grayscale images is something that should be reconsidered. When the images are grayscale the model is relying on pattern recognition instead of relying on colour as well as pattern recognition with RGB images. For my experiments the results on accuracy were better with grayscale images that also meant it took less time for training the model which is a plus for real time analysing for trash in the ATC mechanism.

Some concerns involve the chances of the model overfitting the data in training because of the controlled environment that the images are taken from. That is that all the images in all

the categories have the same background. In retrospect I think it is not likely to have an impact on generalising/predicting since all future unseen images will have the same background, and other regularization techniques are used to control overfitting.

The future prospect of an automatic sorting mechanism such as the ATC is great. There are many reasons for companies, institutions and governments to invest in this technology. The environmental reasons for investing are that better recycling corresponds to less landfills and reduction of the carbohydrate footprint. The financial advantages can be seen for example in the general waste compartment which is the largest category where a rough estimate of 50% - 70% is misclassified. It is the most expensive part in terms of disposal but if the general waste is categorized correctly that part of the bin will decrease significantly and therefore save money, even turning that cost into revenue since recycling centres pay for categorised plastic and paper waste at their facilities.

Environmental and social responsibility is a large factor in today's organizational cultures and by implementing an automatic way of recycling into the company it will be in its benefit. It would show the determination that the company is taking, to do their part for better recycling. The government has even a stronger reason to take part in better recycling for they have to lead with an example for the rest of us to follow.

7 References

- [1]"SORPA - Árangur í umhverfismálum", Sorpa.is, 2018. [Online]. Available: <http://www.sorpa.is/>. [Accessed: 05- Oct- 2018].
- [2]ec.europa.eu, "Waste statistics," 15 2017. [Online]. Available: http://ec.europa.eu/eurostat/statisticsexplained/index.php/Waste_statistics#Main_statistical_findings. [Accessed: 12- Nov- 2017].
- [3]Walter H. Pitt and Warren S. McCulloch, A logical calculus of the ideas immanent in nervous activity. 1943, p. 101.
- [4]D. Hebb, *The organization of behavior*. New York, London, 1949, p. 73.
- [5]"Frank Rosenblatt biography, list of Frank Rosenblatt inventions | edubilla.com", *Edubilla.com*, 2018. [Online]. Available: <http://www.edubilla.com/inventor/frank-rosenblatt/>. [Accessed: 25- Jan- 2018].
- [6]"Neural Networks - History", *Cs.stanford.edu*, 2018. [Online]. Available: <https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html>. [Accessed: 3- Mar- 2018].
- [7]M. Minsky and S. Papert, *Perceptrons*. 1969.
- [8] M. Minsky and S. Papert, *Perceptrons*. 1969, p. 77-81.
- [9]C. Smith and B. McGuire, "The History of Artificial Intelligence", *Courses.cs.washington.edu*, 2006. [Online]. Available: <https://courses.cs.washington.edu/courses/csep590/06au/projects/history-ai.pdf>. [Accessed: 12- Apr- 2018].
- [10]"The Hopfield Model", *Page.mi.fu-berlin.de*, 1996. [Online]. Available: <https://page.mi.fu-berlin.de/rojas/neural/chapter/K13.pdf>. [Accessed: 12- Nov- 2018].
- [11] McCorduck, Pamela (2004), *Machines Who Think* (2nd ed.) p.441
- [12]"Neural Networks - Applications", *Cs.stanford.edu*, 2018. [Online]. Available: <https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Applications/index.html>. [Accessed: 14- Mar- 2018].
- [13]"CS231n Convolutional Neural Networks for Visual Recognition", *Cs231n.github.io*, 2018. [Online]. Available: <http://cs231n.github.io/neural-networks-1/>. [Accessed: 21- Aug- 2018].
- [14]"Perceptrons - the most basic form of a neural network - Applied Go", *Applied Go*, 2016. [Online]. Available: <https://appliedgo.net/perceptron/>. [Accessed: 28- Aug- 2018].
- [15]S. Smith, *Digital Signal Processing*. Saint Louis: Elsevier Science & Technology, 2013, pp. 458-461.
- [16]J. Ahire, *Artificial Neural Networks The brain behind AI*. Independently published, 2018, p. 118.

- [17]A. Ng, "Convolutions Over Volume - Foundations of Convolutional Neural Networks | Coursera", *Coursera*, 2018. [Online]. Available: <https://www.coursera.org/lecture/convolutional-neural-networks/convolutions-over-volume-ctQZz>. [Accessed: 10- Okt- 2018].
- [18]"CS231n Convolutional Neural Networks for Visual Recognition", *Cs231n.github.io*, 2018. [Online]. Available: <http://cs231n.github.io/convolutional-networks/>. [Accessed: 12- Okt- 2018].
- [19]"ImageNet Large Scale Visual Recognition Competition (ILSVRC)", *Image-net.org*, 2015. [Online]. Available: <http://image-net.org/challenges/LSVRC/>. [Accessed: 13- Nov- 2018].
- [20]Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, *Gradient-Based Learning Applied to Document Recognition*. 1998, p. 7.
- [21]A. Krizhevsky, I. Sutskever and G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*. 2012, pp. 1-5.
- [22]C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, *Going deeper with convolutions*. 2014, pp. 1-10.
- [23]J. Jordan, "Common architectures in convolutional neural networks.", *Jeremy Jordan*, 2018. [Online]. Available: <https://www.jeremyjordan.me/convnet-architectures/>. [Accessed: 15- Okt- 2018].
- [24]K. He, X. Zhang, S. Ren and J. Sun, *Deep Residual Learning for Image Recognition*. 2015, pp. 1-6.
- [25] D. Pedamonti, Comparison of non-linear activation functions for deep neural networks on MNIST classification task. 2018. Available: <https://arxiv.org/pdf/1804.02763.pdf> [Accessed: 14- Okt- 2018].
- [26]"CS231n Convolutional Neural Networks for Visual Recognition", *Cs231n.github.io*, 2018. [Online]. Available: <http://cs231n.github.io/convolutional-networks/#pool>. [Accessed: 3- Okt- 2018].
- [27]J. Brownlee, "Overfitting and Underfitting With Machine Learning Algorithms", *Machine Learning Mastery*, 2016. [Online]. Available: <https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/>. [Accessed: 4- Nov- 2018].
- [28]W. Koehrsen, "Overfitting vs. Underfitting: A Conceptual Explanation", *Towards Data Science*, 2017. [Online]. Available: <https://towardsdatascience.com/overfitting-vs-underfitting-a-conceptual-explanation-d94ee20ca7f9>. [Accessed: 24- Okt- 2018].
- [29]A. Budhiraja, "Learning Less to Learn Better—Dropout in (Deep) Machine learning", *Medium*, 2016. [Online]. Available: <https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5>. [Accessed: 4- Jun- 2018].
- [30]"CS231n Convolutional Neural Networks for Visual Recognition", *Cs231n.github.io*, 2018. [Online]. Available: <http://cs231n.github.io/neural-networks-3/>. [Accessed: 4- Jun- 2018].

- [31]Y. LeCun, C. Cortes and C. J.C.Burges, "MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges", *Yann.lecun.com*, 2018. [Online]. Available: <http://yann.lecun.com/exdb/mnist/>. [Accessed: 22- Aug- 2018].
- [32]A. Krizhevsky, V. Nair and G. Hinton, "CIFAR-10 and CIFAR-100 datasets", *Cs.toronto.edu*, 2018. [Online]. Available: <https://www.cs.toronto.edu/~kriz/cifar.html>. [Accessed: 14- Nov- 2018].
- [33]A. Damien, "Data Augmentation - TFLearn", *Tflearn.org*, 2018. [Online]. Available: http://tflearn.org/data_augmentation/. [Accessed: 14- Nov- 2018].
- [34]J. Brownlee, "Why One-Hot Encode Data in Machine Learning?", *Machine Learning Mastery*, 2017. [Online]. Available: <https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/>. [Accessed: 10- Okt- 2018].
- [35]"TensorFlow", *TensorFlow*, 2018. [Online]. Available: <https://www.tensorflow.org/>. [Accessed: 1- Nov- 2018].
- [36]A. Damien, "TFLearn | TensorFlow Deep Learning Library", *Tflearn.org*, 2018. [Online]. Available: <http://tflearn.org/>. [Accessed: 10- Okt- 2018]

APPENDIX

```
'''
CNN_ATC_PREP.py
Image preparation, resizing images to a 400 x 400 grayscale pixel images.
Splitting them into training and test with one hot labels.
Partly taken from Sentdex https://pythonprogramming.net tutorial and
TFlearn tutorials tflearn.org edited Stefán Sturla Stefánsson
'''

import cv2
import numpy as np
import os
from random import shuffle
from tqdm import tqdm
import tensorflow as tf
import matplotlib.pyplot as plt
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
from tflearn.data_augmentation import ImageAugmentation
from sklearn import metrics

#Original image path
TRAIN_DIR = '/home/stefan/Desktop/rusl/Train_label'
TEST_DIR = '/home/stefan/Desktop/rusl/Test_Label'
IMG_SIZE = 400

#Creating one hot labeling from dataset labels
def label_img(img):
    word_label = img.split('.')[0]
    if word_label == 'floskur': return [1,0,0,0] #plastic bottles
    elif word_label == 'pappir': return [0,1,0,0] #paper
    elif word_label == 'dosir': return [0,0,1,0] #aluminum cans
    elif word_label == 'plast': return [0,0,0,1] #plastic

#Training data fabrication with 3566 400 x 400 grayscale images
def create_train_data():
    training_data = []
    for img in tqdm(os.listdir(TRAIN_DIR)):
        label = label_img(img)
        path = os.path.join(TRAIN_DIR, img)
        img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        training_data.append([np.array(img), np.array(label)])
    shuffle(training_data)
    np.save('training_data400.npy', training_data)
    return training_data

#Test set fabrication with 1189 400 x 400 grayscale images
def create_test_data():
    testing_data = []
    for img in tqdm(os.listdir(TEST_DIR)):
        label = label_img(img)
```

```

        path = os.path.join(TEST_DIR,img)
        img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        testing_data.append([np.array(img),np.array(label)])
    shuffle(testing_data)
    np.save('testing_data400.npy', testing_data)
    return testing_data

train_data = create_train_data()
test_data = create_test_data()

'''
Stefán Sturla Stefánsson
CNN_ATC_TRAIN.py
Training CNN phase
Input 400 x 400 grayscale
Learning rate 1e-4
Image augmentation, rotation and blur
6 convolution layers, 32, 64, 128, 256
4 max pooling layers, 5 x 5
2 fully connected layers ReLU, Tanh
'''

import cv2
import numpy as np
import os
from random import shuffle
from tqdm import tqdm
import tensorflow as tf
import matplotlib.pyplot as plt
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
from tflearn.data_augmentation import ImageAugmentation
from sklearn import metrics

#image size and learning rate
IMG_SIZE = 400
LR = 1e-4

#Model name for Tensorbord visualization
MODEL_NAME = 'prufal'

#Loading Train set images
train_data = np.load('training_data400.npy')

#Splitting train sett into validation set containing 350 images
train = train_data[:-350]
test = train_data[-350:]

#Seperating label and image for the training
X_train = np.array([i[0] for i in train]).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
y_train = [i[1] for i in train]
#Seperating label and image for the validation set

```

```

X_val = np.array([i[0] for i in test]).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
y_val = [i[1] for i in test]

# Create extra synthetic training data rotating, flipping, blurring.
img_aug = ImageAugmentation()
#img_aug.add_random_flip_leftright()
img_aug.add_random_rotation(max_angle=6)
img_aug.add_random_blur(sigma_max=2.0)

# Building the convolutional neural network
tf.reset_default_graph()
#Input of the network along with real time image augmentation
network = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1],
data_augmentation=img_aug, name='input')

#First convolution layer with 32 filters 5 x 5 in size
network = conv_2d(network, 32, 5, activation='relu')
#Max pooling function 5 x 5
network = max_pool_2d(network, 5)

#Second convolution layer with 64 filters 5 x 5 in size
network = conv_2d(network, 64, 5, activation='relu')

#Third convolution layer with 128 filters 5 x 5 in size
network = conv_2d(network, 128, 5, activation='relu')
#Max pooling function 5 x 5
network = max_pool_2d(network, 5)

#Forth convolution layer with 256 filters 5 x 5 in size
network = conv_2d(network, 256, 5, activation='relu')
#Max pooling function 5 x 5
network = max_pool_2d(network, 5)

#Fifth convolution layer with 64 filters 5 x 5 in size
network = conv_2d(network, 64, 5, activation='relu')

#Sixth convolution layer with 32 filters 5 x 5 in size
network = conv_2d(network, 32, 5, activation='relu')
#Max pooling function 5 x 5
network = max_pool_2d(network, 5)

#Fully-connected 1024 node neural network, and ReLU activation function
network = fully_connected(network, 1024, activation='relu')
#Dropout set at 0.6
network = dropout(network, 0.6)

#Fully-connected 2048 node neural network, and Tanh activation function
network = fully_connected(network, 2048, activation='tanh')
#Dropout set at 0.7
network = dropout(network, 0.7)

#Regression and output for categories set
network = fully_connected(network, 4, activation='softmax')
network = regression(network, optimizer='adam', learning_rate=LR,
loss='categorical_crossentropy', name='targets')

# Training the network

```

```

model = tflearn.DNN(network, tensorboard_dir='ATC6',
tensorboard_verbose=0)
    #input image            input labels
model.fit({'input': X_train}, {'targets': y_train}, n_epoch=40,
        validation_set=({'input': X_val}, {'targets': y_val}),
        snapshot_step=500, show_metric=True, batch_size=12,
run_id=MODEL_NAME)

#saved model name
model.save('ATC6.tflearn')

'''
Stefán Sturla Stefánsson
CNN_ATC_TEST.py
Predictions on test set containing 1189 images
98.7384% prediction accuracy
'''

import cv2
import numpy as np
import os
from random import shuffle
from tqdm import tqdm
import tensorflow as tf
import matplotlib.pyplot as plt
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
from tflearn.data_augmentation import ImageAugmentation
from sklearn import metrics

IMG_SIZE = 400
LR = 1e-4

#Loading Test set images
test_data = np.load('testing_data400.npy')

#Separating label and image for the test set
X = np.array([i[0] for i in test_data]).reshape(-1, IMG_SIZE, IMG_SIZE,
1)
Y = [i[1] for i in test_data]

#The model without image augmentation
tf.reset_default_graph()

network = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

#First convolution layer with 32 filters 5 x 5 in size
network = conv_2d(network, 32, 5, activation='relu')
#Max pooling function 5 x 5
network = max_pool_2d(network, 5)

```

```

#Second convolution layer with 64 filters 5 x 5 in size
network = conv_2d(network, 64, 5, activation='relu')

#Third convolution layer with 128 filters 5 x 5 in size
network = conv_2d(network, 128, 5, activation='relu')
#Max pooling function 5 x 5
network = max_pool_2d(network, 5)

#Forth convolution layer with 256 filters 5 x 5 in size
network = conv_2d(network, 256, 5, activation='relu')
#Max pooling function 5 x 5
network = max_pool_2d(network, 5)

#Fifth convolution layer with 64 filters 5 x 5 in size
network = conv_2d(network, 64, 5, activation='relu')

#Sixth convolution layer with 32 filters 5 x 5 in size
network = conv_2d(network, 32, 5, activation='relu')
#Max pooling function 5 x 5
network = max_pool_2d(network, 5)

#Fully-connected 1024 node neural network, and ReLU activation function
network = fully_connected(network, 1024, activation='relu')
network = dropout(network, 0.6)

#Fully-connected 2048 node neural network, and Tanh activation function
network = fully_connected(network, 2048, activation='tanh')
network = dropout(network, 0.7)

#Regression and output for categories set
network = fully_connected(network, 4, activation='softmax')
network = regression(network, optimizer='adam', learning_rate=LR,
loss='categorical_crossentropy', name='targets')

model = tflearn.DNN(network)

#load model
model.load('ATC6.tflearn')

#Running the images through the model
i = 0
while i < 1189:
    pre = model.predict([X[i]])
    print (pre)
    pre = np.around([pre])
    print (pre)
    print (Y[i])
    print ('*' * 70)
    i = i + 1

```