OULU UNIVERSITY OF
APPLIED SCIENCES

Shalik Ram Sapkota

USING MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE IN PRODUCTION TECHNOLOGY

USING MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE IN PRODUCTION TECHNOLOGY

Shalik Ram Sapkota
Bachelor's thesis
Spring 2019
Degree Programme in Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology

Author: Shalik Ram Sapkota
Title of the bachelor's thesis: Machine Learning and Artificial Intelligence in Production Technology
Supervisor: Jukka Jauhiainen
Term and year of completion: spring 2019 Number of pages: 48

This thesis is the product of collaboration of Centria R&D and Akkuser Oy, with the purpose of developing battery sorting system with machine vision and AI. As the industrial world is entering towards the fourth industrial revolution, AI, machine learning and deep learning are hot topics. In neural networks, a convolutional neural network is widely used to do image recognition, image classification and object detection. The aim of this thesis study was to find out the best solution to classify different kinds of used batteries according to their images to find out their inside chemical composition using the latest technology available now.

To find out the best solution to classify object, we used Google's Tensorflow and OpenCV were used as tools inside the Anaconda virtual environment. In technical words, deep learning CNN models were implemented during training and testing. Series of convolutional layers with filters, a max pooling layer, fully connected layers and other functional requirements were applied to detect whether object is classified or not. The result of this thesis work was good as expected even with limited resources. For implementing this project in real life, many other functions such as databases, a user interface and human-usable application need to be developed.

Keywords: AI, Machine learning, Deep Learning, Max-pooling, Fully connected layers, Softmax function, Tensorflow, OpenCV

PREFACE

Based on the communication between the company and our research team, we came to conclusion that it is time to use the new technology to detect and classify batteries.

I would like to thank Jukka Jauhiainen, for supervising my thesis, and Kaijo Posio, for the language inspection of this thesis. Furthermore, I would like to thank Jorma Hintikka for giving me this opportunity to work with. He is a very good leader. I would like to thank Tero Kerola, Sakari Pieskä and Mamut Kechok for giving a fruitful suggestion to explore my knowledge and to be able to conduct projects regarding AI, machine learning and deep learning. Thanks to Emi Morikawa and Riko Suzuki for being amazing workmates. Also, I would like to thank Tensorflow for amazing documentation.

This project was done for Nivala based Akkuser Oy, which recycles used batteries. The Purpose of the project is to sort batteries by machine vision intelligence to reduce human work force and to increase safety.

Oulu, 30.04.2019

Shalik Ram Sapkota

TABLE OF CONTENTS

VOCABULARY AND ABBREVIATIONS

TERMS	MEANING
Accuracy	Percentage of correct classified images
Class	Category of batteries needed to be sorted
Error rate	Percentage of incorrect classified images
Gradient descent	Slope/partial derivatives
Matrix	Multi-dimensional
OpenCV	Open-Source Computer Vision Library
Optimizer	A function used to compute gradient/partial derivatives of the error
Placeholder	Input training data
Relu	Rectifiled linear unit
Scalar	Single value
Softmax	Activation function
Test data	Data that the network has not seen
Training data	Data used for training
Validation data	Data used for validating algorithm
Vector	One-dimensional

ABBREVIATIONS	MEANING
ANN	Artificial neural network
CNN	Convolutional neural network
DL	Deep learning
ML	Machine Learning
RNN	Recurrent neural network
TF	Tensorflow

1 INTRODUCTION

Artificial intelligence was initially founded in 1956 as an academic discipline. The traditional problems of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and ability to move and manipulate objects. The first practical application was introduced by Alan Turing in 1950s called “machine intelligence” in order to conceptualize that machine can think. This AI-led excitement became nearer when the invention of the computer came true in the 1980s. After that, technology corporations have been doing exploratory research in order to develop technologies closer to the consumer; by making applications and making them affordable. (1)

The production technology has started to rely on computer technology to innovate and improve the productivity and to decrease waste management. In modern time manufacture sectors, the material is not only the form of waste as human time and workforce are also factors that can be considered as waste. Over the last few years, there has been an explosion of technologies such as artificial intelligence, big data, advanced robotics, 3D printing, and IOT. Companies are increasing their research and development in order to get advanced technologies to increase their productivity. (2)

In addition to that, AI has been used for previously manually intensive activities. Every day new technologies are emerging exponentially to overcome the slow production and high-cost structure. There are different kinds of theories and ideas about cost and waste management. If all these theory and ideas are put together, AI is already the ultimate solution to get the best result ever seen before.

As can be seen nowadays, everyone is trying to emphasize how to implement AI on their company to get a high output and to get over the high production cost. The technology is helping them a lot to improve productivity over the years, but they are still not satisfied yet. As the competitiveness of production time and cost is increasing, the cost of goods are reducing in order to be competitive on the market.

Even though there are tons of opportunities, there are threats too. There are many theories (ideas) going around the tech industries and media that AI, robotics and machine learning will take over our jobs. There are a few common repetitive concerns in our society that keep appearing in different forms such as that AI will take over our jobs, the robot will steal my income stream, there will be a massive job-depression, Self-Driving care will replace drivers and robots will rule over us in the future. "*I have exposure to the very cutting-edge AI, and I think people should be really concerned about it*" said Elon musk. Elon Musk, one of the prominent voice that has speak about AI which is of course not a good sign. "*I keep sounding the alarm bell*," Musk continued. "*But until people see robots going down the street killing people, they don't know how to react, because it seems so ethereal*" musk added. He previously labelled AI as our biggest existential threat. The Famous physicist Stephen Hawking told the BBC in 2014 that "*the development of full artificial intelligence could spell the end of the human race*." (3)

As the author is an AI enthusiast, he did some research to state this viewpoint. In this field of research and development, the author has fortunately got the opportunity to talk about and work with many professionals across sectors and methods of Artificial intelligence over the last few months. There are of course challenges and opportunities but there are threats, too. It is so sensitive subject to research and develop that if trying to be evil-minded it could be one of the worst nightmares that humankind has ever seen before. As an AI enthusiastic, the author's concern will always be to develop ethical AI rather than competitive and threat AI. After all, humans are the creators of any kind of AI; humans are able to control what kind of AI to create. It is necessary to emphasize creating narrow AI to accomplish just a specific task instead of general AI that does everything like a human. AI can be used for productive and peaceful purposes, like an atom bomb was invented to stop wars. In author's opinion, there should be certain guidelines of developing AI and robots, as well as the government level policy to overcome certain level of threat. People need to be aware of what we are doing but it is not necessary to be afraid of the technology.

2 ARTIFICIAL INTELLIGENCE IN THE INDUSTRIAL REVOLUTION 4.0

Cloud computing, Internet of Things, real-time sense-and-response technologies, robotics, AI, 3D printing are predicting to revolutionize how goods are produced and services are delivered. We are on the way towards the fourth industrial revolution. This is moving forward in order to introduce IoT in manufacturing making the cyber physical world possible. Manufacturing industries are being converted into smart factories to achieve the highest productivity and to reduce the material and human force waste rate. One central idea is to move from physical industries to cyber-physical industries to improve the production model where materials and machines communicates with each other in real-time without the need of a fixed production plan. (4)

The production technology needs self-learning, self-manufacturing and low cost production model to sustain and grow in very competitive market. As can be seen, AI is growing exponentially everyday, it can compete with human intelligence and even can surpass it in some cases. If the rate of progress in hardware and the raw computing power are considered, they have a capability to surpassing the sensory and mental processing rate of a human being. Even in the present time computers are able to store and retrieve information and solve problems faster than a human. When it comes to using AI and robotics together in production or manufacturing industries, it is certainly faster, cheaper and profitable than a human work force. AI can be used in many fields such as robitics, education, health care, supply chain and logistics as well as cyper security to increase productivity and safety. (4)

2.1 Robotics

AI focuses on how to get machines or computers to perform the things in the same way that a human can do. Robots can already solve problems in limited realms. If the AI robot or computer gathers the fact about a situation through a

sensor or human input, then the robot or computer compares the information to stored data and decides whether information is right or wrong. (5)

There are many examples that any AI computer has defeated a human in many cases, For instance a chess computer can defeat chess master easily, Alpha go defeated 3-times European Champion , Mr Fan Hui, in October 2015(6). Thus, it can be easily predicted that in many cases robots will be more cognitive than human in order to achieve productivity.

2.2 Education

Education has been like robotics in a way that it only concentrates on one specific field rather than gives value to a wide range of knowledge to support the growth of a wise, civil and futuristic human being. Some of the teens and youth are also getting depressed about not getting as much success and progress as they are supposed to. Around the world children are out of the schools and they are going towards the dark future. To tackle this problem, artificial intelligence can be used to teach and analyse the overall problems of the human psychology in order to provide them better a education. (7)

2.3 Health care

The AI assisted robotics surgery, which allows doctors to perform complex procedures with a greater control of the conventional approach, has already been practised. The AI assisted robotic surgery, administrative tasks, pattern analysis, virtual nursing assistants and image analysis for disease recognition are going to be revolutionary in the health care sector and can be achieved with the help of artificial intellegence. Billions of euros can be saved annually as virtual nurses and service can be available 24 hours a day and 7 days a week. AI can be used to provide answers for patients' questions, to monitors patients and provide a quick answer as well as review and learn from past data. Nowadays, an image analysis to find out different kinds of diseases, is also possible. Doctors and scientists agree that sometimes in some cases AI with a more precise method can do better than a human. (8)

2.4 Supply chain and logistics

A supply chain and logistics are the network-based nature of the industries. The main task of their business is implementing and scaling the supply industrial goods from one place to a targeted place with human components inside some certain framework. This is the time to revolutionize the supply chain management. Researchers at IBM stated that in the supply chain and logistics estimately only 10% of current systems, data, and interactions include elements of an AI analysis. AI investments are already improving and generating a larger return than investers have never seen before.

Chatbot, tracking goods, analyzing supply chain agent behaviours, tracking the different supply chain system and finding out cost difference between two of them can be beneficial for logistics sector in order to get a greater delivery speed and lower cost. Data is growing exponentially in supply chain on daily basis in both structured and unstructured form, which both can manipulate the traditional working ways in logistics. Logistics companies are largely depending on the physical and cyber network, which should work with a high volume, low margin and time sensitive deadlines with a lean assist allocation. AI can help to optimize the network of companies with a higher efficiency in order to redefine today's behaviours and practices, take operations proactive to reactive, forcast and predict processes, change a system from manual to autonomous and change services from standarised to personalized. (9)

2.5 Cyber security

As people are moving fast to cyber-physical technologies, cyber security is one of the crucial factors in order to maintain the system secure. It is a famous saying in the cyber security field that the prevention achieves everything that the detection cannot. Conventional security systems are quite slow and insufficient to face the sophisticated cyber threats. Artificial intelligence techniques can improve the overall security and performance and also find out high level cyber threat related risks and concern. Banking, aviation, government, defences are the

main target for cyber attackers these days as well as those sectors that have crucial information, extremely important and effective at the policy making level. Cyber security is not only a technological issue, it is one of the regulatory subjects where there can be various ways to deal with its risks and threats.

AI can help to find the pattern of cyber-attack, figuring it out within very small fraction of time, and taking an action or notifying the concerning authorities to stop the attack. To train the AI to make a proper decision for future attack preventions, it is necessary to see the past of the attack and recognise the patterns of attack.

Google has used AI for classification of emails whether it is spam or not. Phishing, malware, Zeroday, and APT detection can be classified and prevented with the help of trained deep neural networks. Identification of inside threat, advanced intrusion detection and network anomaly detections can also be found out, they can guide the system and experts to make appropriate predictions and decisions. (**Error! Reference source not found.**)

3 MACHINE LEARNING

Machine learning is the science of programming computers where they can learn from data. Arther Samuel tells: "*machine learning is the field of study that gives computer the ability to learn without being explicitly programmed*". There used to be a dream that a computer would do their task without a human interference but it is not a dream anymore as it is possible now. Machine learning is already implemented in many fields, such as categories emails (spam or not), face recognition, object detection, voice /sound recognition and grammar checking. (**Error! Reference source not found.**)

In the Figure 1 it is described how machine learning works in real life. First of all, the particular problem that needs to be solved is studied. After that, enough data to train the machine learning algorithm is collected or constructed. The collected data is fed into the model so that it will evaluate the solution. The program will learn from own experiences so that the more data comes inside the training model the easier it is to evaluate the solution. (**Error! Reference source not found.**)

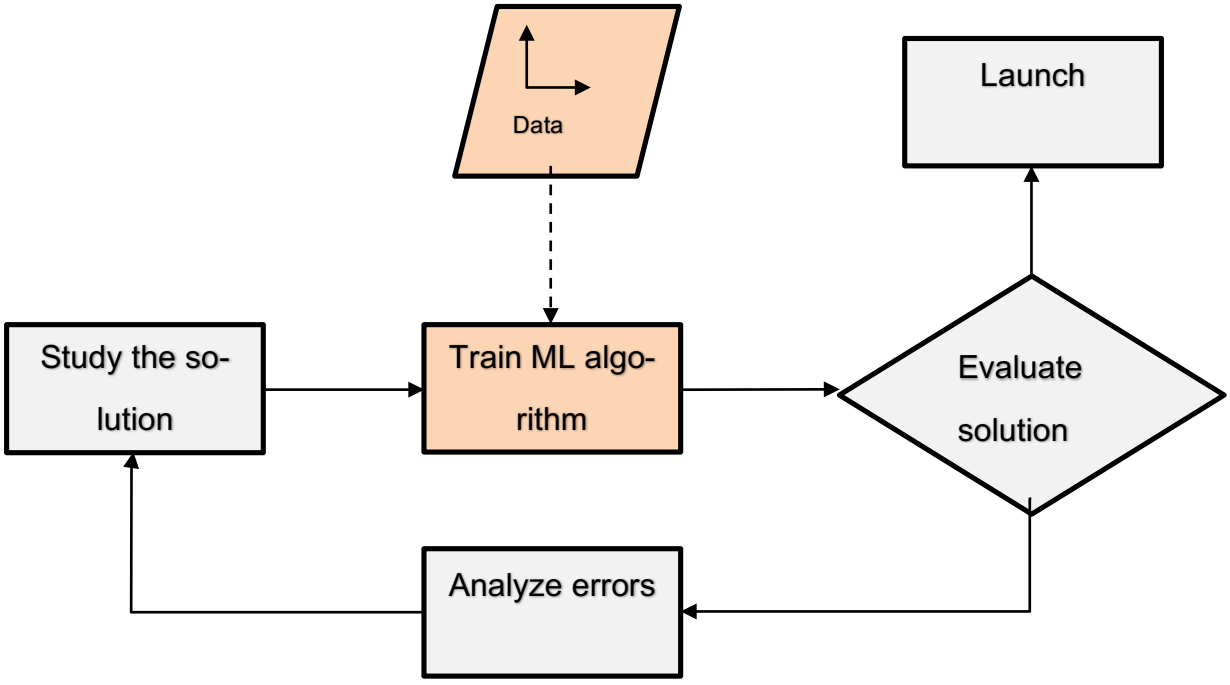

FIGURE 1. Machine learning approach

3.1 Supervised machine learning

Supervised machine learning is mainly used and focused on building the model that makes the prediction based on the known dataset. As adaptive algorithm observes and identifies patterns in data, a computer learns from the observation. When more observation comes to the model, it improves the predictive performance. While preparing the supervised machine learning process, first of all, data needs to be prepared, then the algorithm needs to be chosen. After that, the next steps are: fitting a model, choosing a validation method, examining the fitting model and updating until the results are satisfying. All these steps should be done to get a better supervised machine learning model. (**Error! Reference source not found.**) An overview of a supervised machine learning approach is shown in the figure 2.

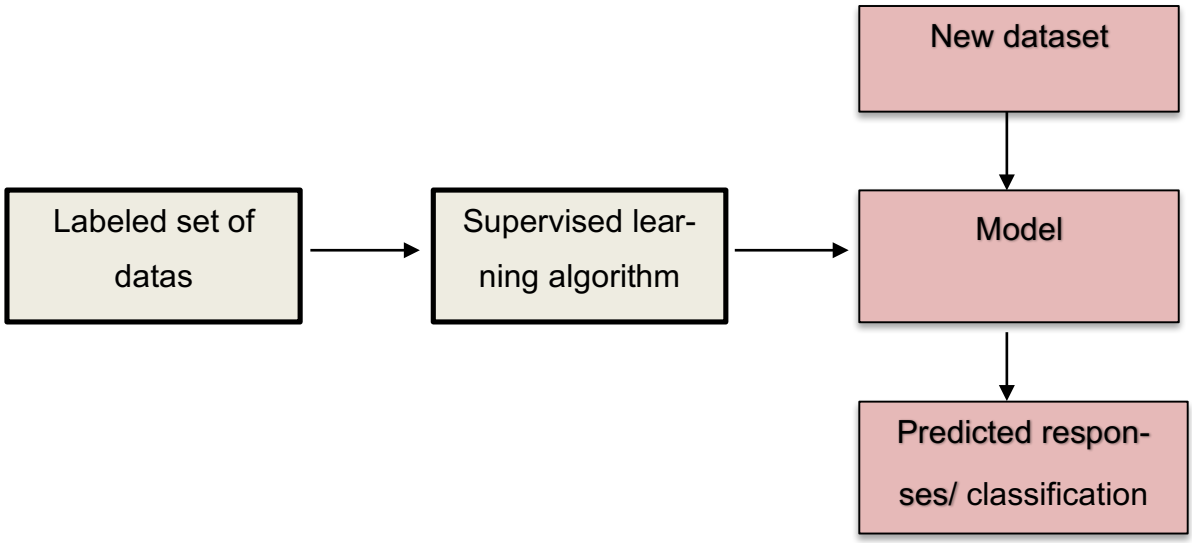

FIGURE 2. Supervised machine learning

3.2 Unsupervised machine learning

Unsupervised machine learning is the process of clustering the result out of unlabelled data inputs. The given data to an unsupervised machine learning algorithm is not labelled, which means that input variables are given with no corresponding output variables. In the process, the algorithms are left to themselves to find interesting structure in the data. An unsupervised machine learning process just gather the clusters of information from data. (**Error! Reference source**

not found.). An overview of an unsupervised machine learning approach is shown in the figure 3.

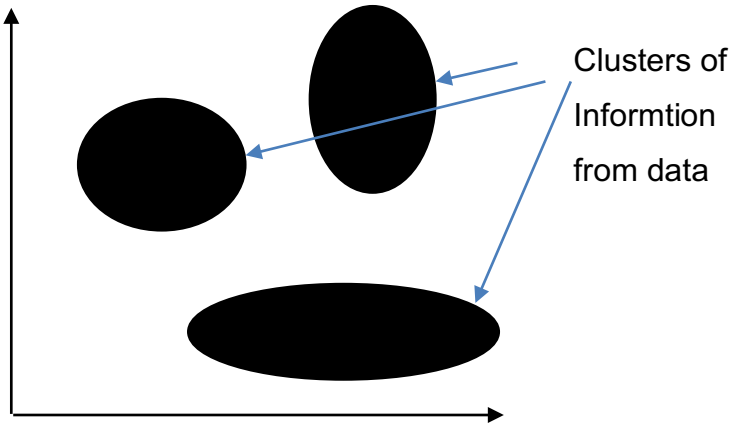

FIGURE 3. Unsupervised machine learning approach

3.3 Reinforcement learning

Reinforcement learning(RL) is a method of machine learning, where an agent can learn to interact with an environment in order to maximize the result. Reinforcement learning emphasizes more on the learning problem than the learning method. In the learning method learning agents are more guided by the problems they have faced in the past experienced and try to avoid them. The reinforcement learning becomes more efficient when the learner becomes aware of which action has the high probability of achieving reward without being guided. (**Error! Reference source not found.**). The figure 4 illustrates the basic idea and element of the reinforcement learning model.

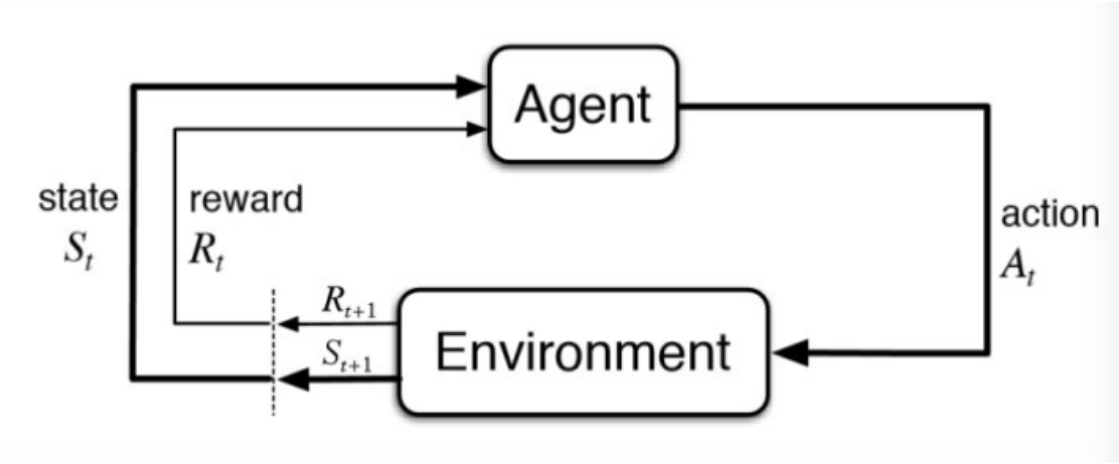

*FIGURE 4. Reinforcement learning model (**Error! Reference source not found.**)*

4 CONVOLUTIONAL NEURAL NETWORK(CNN)

Convolutional neural network(CNN) is a class of deep and feed-forward artificial neural networks that are applied to analyzing a visual imagery, such as image classification, object detection etc. Feedforward neural networks are known as a multi-layered network of neurons. CNNs use a variation of multilayer perceptron designed to require minimal pre-processing while doing machine learning processes. (**Error! Reference source not found.**)

Convolution neural networks were designed or inspired by biological processes. Neurons are electrically excitable cells in living organisms that receive processes and transmit information. Neurons are interconnected to each other to form a neural pathway and neural circuits. Similar to biological neurons, all technical neural networks are interconnected with each other to find the best observation around the given data. (**Error! Reference source not found.**)

CNN consists of an input and an output layer, as well as multiple hidden layers in order to make CNN faster and more efficient. In CNN there are many layers including convolutional layers, locally connected layers, fully-connected layers. The figure 5 below gives the overall architecture of CNN and how it is constructed and how it processes.

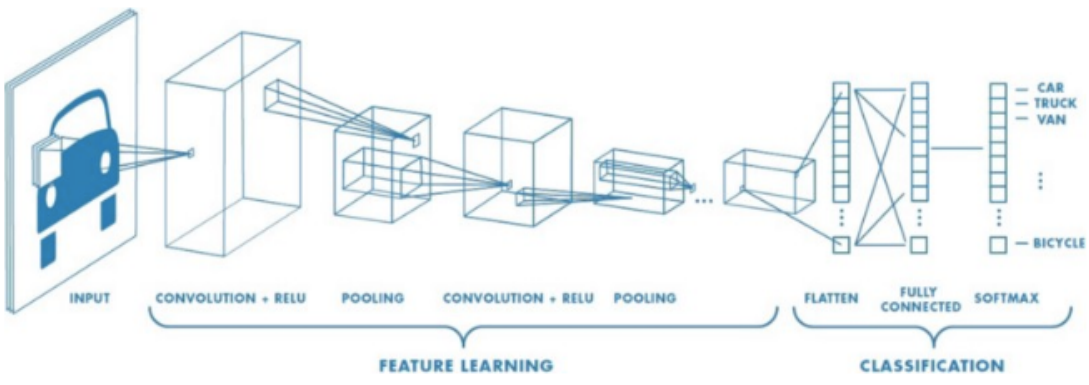

*FIGURE 5. CNN architecture (**Error! Reference source not found.**)*

4.1 Convolutional layers

Convolutional layers apply a convolutional operation to the input layer and pass the result to the next layer. The convolution layer emulates the response of an individual neuron to visual stimuli. Usually, each convolutional neuron processes data only for its receptive field. A fully connected feedforward neural network can be used to learn a feature, as well as to classify data. When the convolutional layers have more than one output channel, it is not practical to apply this architecture to image related tasks. In that case, the layer has a different multi-channel filter to calculate each output. For example, if there is a layer with five input channels (RGB) and seven output channels, the layer would have 7 filters and 3 channels per filter. There are some other factors needed to be known, such as a number of parameters, while preparing convolutional layers. In the figure 5, each unit is connected to a constant number of units in a local region of the previous layers. (**Error! Reference source not found.**)

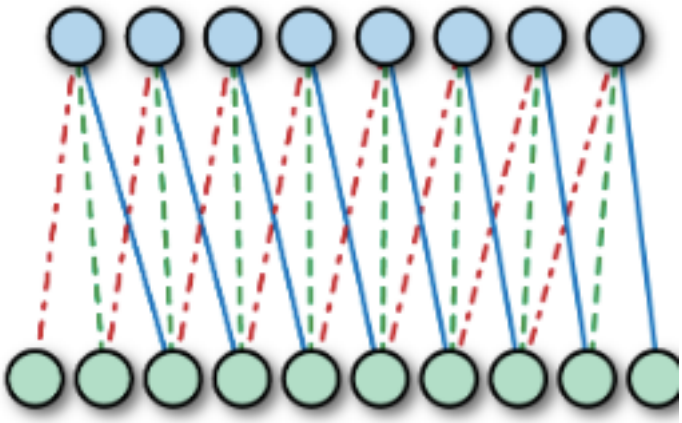

*Figure 5. convolution layer figure illustration (**Error! Reference source not found.**)*

4.2 Number of parameters

While designing a neural network, the number of trainable parameters matters significantly. In the convolutional layers, filters and biases are the parameters that are trained. It is important to know the number of parameters that the convolutional layer adds up to the network. (**Error! Reference source not found.**). The

number of parameters in the neural network can be calculated using the following equation 1.

$$number\ of\ parameters = (F_w * F_h *\ d_i + 1) * d_0$$

In the equation 1, d_i and d_o are the depth of input channel and depth of output channel, respectively. F_w and F_h are the filter width and filter height, respectively. (**Error! Reference source not found.**)

4.3 Locally connected layers

Technically, convolutional layers are locally connected layers. They are locally connected with shared weights. In the same filter all the positions are run in the image, where the same pixel positions "share" the same filters. It allows the network to modify the filter weights until the desired performance has been reached. While this process is great at image classification related work, it still tends to miss some subtle nuances of dimensional arrangements. (**Error! Reference source not found.**)

4.4 Convolution layers with 1x1 filter size

Using a 1x1 filter does not make any sense from the image processing point of view but it can be helpful by adding nonlinearity to your network. Factually, a 1x1 filter calculates the linear combination of corresponding pixels of the input channel and output result by an activation function which adds up the nonlinearity. (5)

4.5 Max Pooling layers

Pooling layers reduce the size of the dimension of the output by changing the kernel by a function of the output values. For instance, max pooling is done by applying a max filter to non-overlapping sub regions of the initial representation. If there is a 4*4 matrix that represents the initial input, there is a 2*2 filter that will run over the output. There is a stride of 2 meanings the (dx, dy) for stepping over the input will be (2, 2) and overlap the region. To make it clearer the figure 6 gives an overall representation.

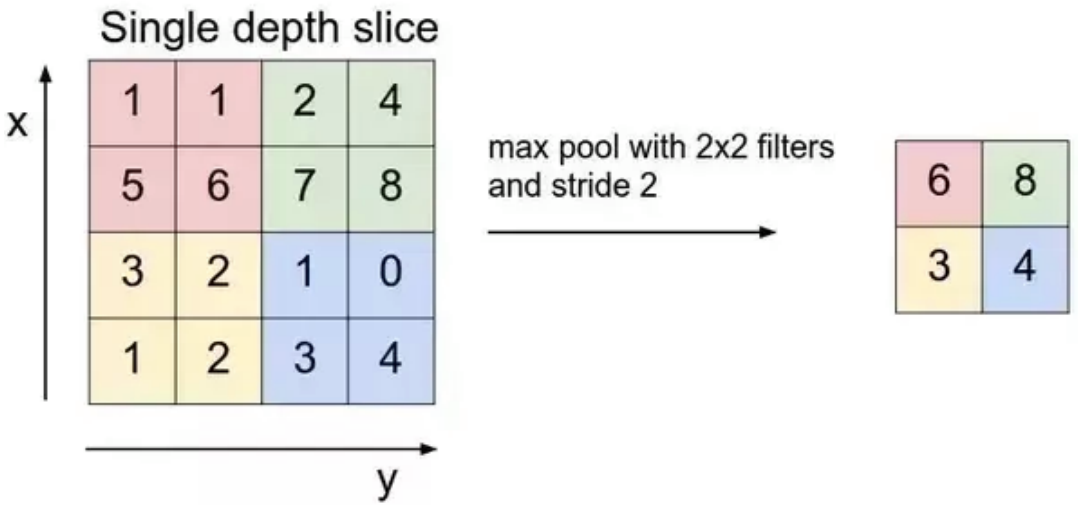

*FIGURE 6. Max-pooling layer representation (**Error! Reference source not found.**)*

4.6 Fully connected layers

After several convolutional and max pooling layers, fully connected layers at the output end are needed to be used to detect high level features. A fully connected layer is a decision layer which decides if the result of classification is either true or false. As represented in the figure 7, all layers are connected with each other.

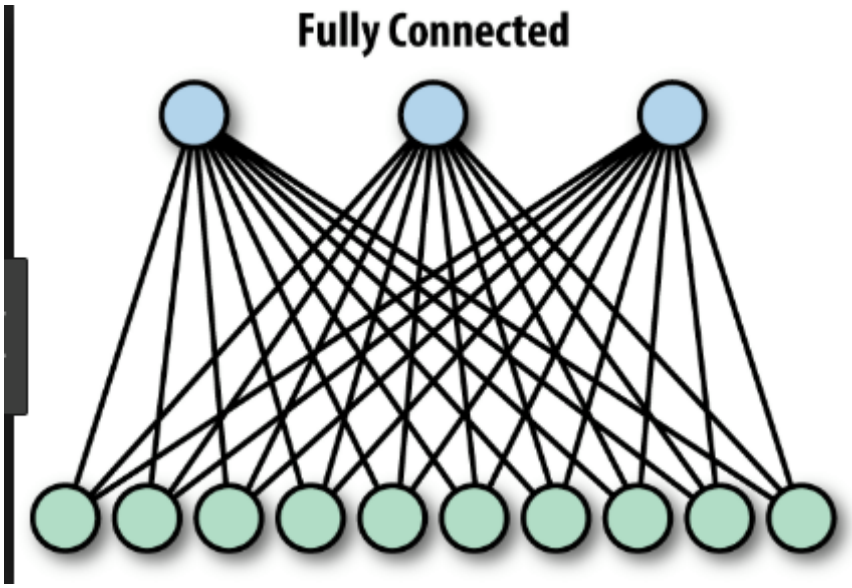

*FIGURE 7. Fully connected layer (**Error! Reference source not found.**)*

5 TECHNOLOGIES AND TOOLS

5.1 Python programming language

Python is an interpreted, object-oriented, high-level programming language with a dynamic semantics which has high-level built in data structures. Python can be combined with dynamic typing and dynamic binding. Python's syntax is very simple to learn and is close to English language itself. Python emphasizes its readability, which leads to reducing the cost of program maintenance. The Python version 3.7 was used during the project implementation. (**Error! Reference source not found.**)

In Python shell, it is possible to do a basic level of Python scripting from a command prompt. The figure 8 shows the basic coding in Python.

```
a = 5
b = 6

print(a + b)
print(a * b)
print(a - b)
print(a / b)
print(a % b)
```

FIGURE 8. Basic Python code.

In the figure 9 the result of Python code based on the figure 8 is represented.

FIGURE 9. Result of python code from figure 8.

The figure 10 demonstrates how Python works for a command prompt.

```
shaliks-MacBook-Air:~ shaliksapkota$ python3
Python 3.7.0 (default, Aug  7 2018, 13:01:09)
[Clang 9.0.0 (clang-900.0.39.2)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

FIGURE 10. Python in command prompt.

5.2 TensorFlow

TensorFlow is an Open-Source software library developed by the Google's Brain team for an internal Google use for dataflow programming across a range of tasks. It is a symbolic math library, which can also be used for machine learning applications such as deep learning. Tensorflow is one of the primary tools for deep learning. It is an open source AI library, using dataflow graphs to build models for various applications and systems. It is not limited to a neural network or even machine learning, but it can run a complex quantum physics stimulation if needed. (**Error! Reference source not found.**) . The Figure 11 illustrates an example how simple Tensorflow code works.

```
In [1]: import tensorflow as tf

/anaconda2/envs/tfdeeplearning/lib/python3.5/importlib/_boots
trap.py:222: RuntimeWarning: compiletime version 3.6 of modul
e 'tensorflow.python.framework.fast_tensor_util' does not mat
ch runtime version 3.5
  return f(*args, **kwds)

In [2]: x1 = tf.constant([1,2,3,4])

In [3]: x2 = tf.constant([5,6,7,8])

In [4]: result = tf.multiply(x1, x2)

In [5]: sess = tf.Session()

In [6]: print(sess.run(result))
[ 5 12 21 32]

In [7]: sess.close()
```

FIGURE 11. Basic Tensorflow code

5.2.1 Tensors

In TensorFlow all data is passed between operations in computational graphs, and these are passed in the form of tensors, hence the name TensorFlow. The word tensor from new Latin means “that what stretches”.

The Tensor structure helps by giving the freedom to shape the dataset in the way that it is wanted. When it comes to dealing with image related problems, it is very helpful due to the nature of how information in images are encoded. When it comes to images, every image contains a certain height and weight, so it would make sense to represent the information contained in it with a two-dimensional structure. (**Error! Reference source not found.**)

Images are encoded into colour channels and the image data is represented into each colour intensity in a colour channel at a given point. The most common colour channel is RGB (Red, Blue, Green). The Information contained in an image is the intensity of each channel colour of the image as shown in the figure 12.

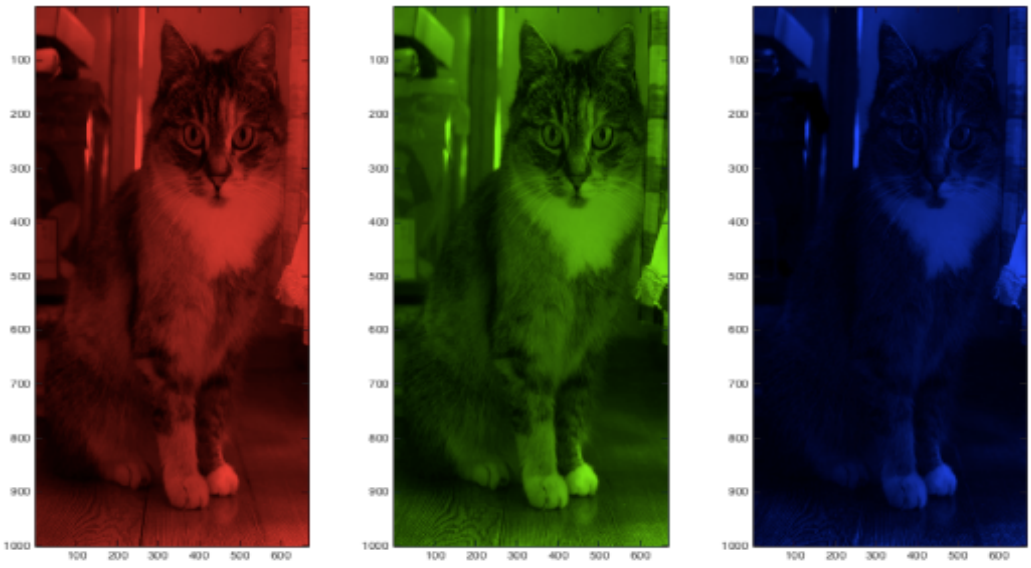

FIGURE 12. changes of colour with same image

5.2.2 TensorBoard

TensorBoard gives the visual picture of all parameters that have been input and output. The example of TensorBoard is shown in the figure 13, where the visualization of the parameters includes accuracy, train, cross-entropy, soft-max, pool.

To open the TensorBoard, tensorboard --logdir=./logs need to be logged into the local, and then it can be accessed to the http:/localhost:6006.

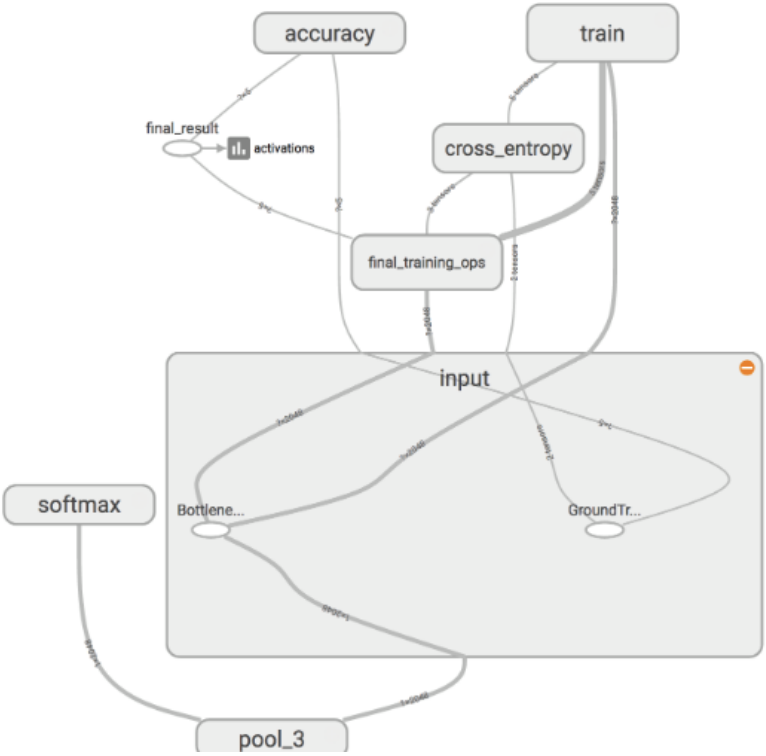

*FIGURE 13. Tensorboard visualization (**Error! Reference source not found.**)*

5.3 Keras

Keras is a high-level neural network API, which is written in Python and is capable of running on the top of Tensorflow, Microsoft Cognitive Toolkit (CNTK) or theano. Keras allows easy and fast prototyping of neural networks and supports both a convolutional network and a recurrent network or both together. Keras can run

seamlessly on both CPU and GPU. Keras is famous for its user friendliness, modularity, easy extensibility and compatibility with Python. (**Error! Reference source not found.**)

5.4 Jupyter notebook

Jupyter notebook is a flexible and popular tool which allows to put code, The result of the code and any kind of plot or visualization in the same document or file. Jupyter notebook is a web based interface that allows fast creating, prototyping and sharing of data-related projects. It is friendly for data cleaning and transformation, statistical modeling, machine learning, numerical stimulation and much more. The name Jupyter is an acronym of three languages that it was designed for: JUlia, PYThon, and R. (**Error! Reference source not found.**)

5.5 OpenCV

OpenCV (open source computer vision) is a library of programming functions mainly focused on real-time computer vision originally developed by Intel. OpenCV is a cross-platform and free to use under the open-source BSD license. It supports the deep learning frameworks TensorFlow, Torch or PyTorch and Caffe. OpenCV can run on the (Windows, Linux, MacOS, FreeBSD, NetBSD, OpenBSD and desktop) operating systems and (Android, iOS, Maemo and Blackberry 10) mobile operating systems. (**Error! Reference source not found.**)

The OpenCV area includes 2D and 3d features toolkits, egomotion estimation, facial recognition system, gesture recognition, human computer interaction (HCI), mobile robotics, motion understanding, object identification, segmentation and recognition, stereopsis stereo: deep perception from 2 cameras, structure from motion and augmented reality.

To support some of the OpenCV areas, OpenCV includes a statistical machine learning library that contains boosting, decision tree learning, gradient boosting trees, expectation-maximization algorithm, k-nearest neighbour algorithm, naive bayses classifiers, artifical neural networks, random forest, support vector(SVM) and deep neural networks(DNN). (**Error! Reference source not found.**)

5.6 Atom

Atom is a free and open-source text and code editor. Atom provides various important features such as autocomplete, find and replaces, atom packages, version control in atom, basic customization. The Atom Environment has its environment modularity when it comes to add more features, e.g. installing packages. (**Error! Reference source not found.**)

5.7 Conda Virtual Environment

Conda is an open source package and environment management system that runs on Windows, MacOS and Linux operating systems. Conda has functionality to install, run and updates packages and their dependencies. It makes easier to work while creating, saving, loading and switching between various environments in the local computers. Initially, Conda was created for Python programs but it can package and distribute software for any other languages. (**Error! Reference source not found.**)

5.8 Panda

Panda is an open source, BSD-licensed library providing a high performance, handy data structure and data analysis tools for python programming language. Panda is a Python package providing a quick, flexible and expressive data structure to make programming more functional with relational and labelled data. Panda is very useful while importing data, exporting data, creating test objects, viewing or inspecting Data, selecting and data cleaning. (**Error! Reference source not found.**). To get started with Panda the imports shown in The figure 14 need to be performed.

```
import pandas as pd
import numpy as np
```

*FIGURE 14. Panda starting imports. (**Error! Reference source not found.**)*

5.9 Numpy

Numpy is the essential package for scientific computing with Python. Numpy is an important library for Python programming that provides a multidimensional array object, many derived objects, routing for fast operations on arrays, selection, shape manipulation, sorting, mathematical operation and many more. (**Error! Reference source not found.**). To get started with Numpy the imports shown in the figure 15 need to be performed.

```
import numpy as np
```

*FIGURE 15. Numpy starting import. (**Error! Reference source not found.**)*

5.10 Matplotlib

Matplotlib is a Python 2D plotting library which produces the variety of quality figures with the best graphic and visual interfaces. Matplotlib makes difficult things easier and hard things possible with its amazingly working functionality and interactive environments across platforms.It can be used in Python scripts, web application servers, the Jupyter notebook, four graphical user interface toolkits and many other platforms. Matplotlib is used to generate plots, bar charts, error/charts, histograms, power spectra. with just a few lines of codes. (**Error! Reference source not found.**). The Figure 16 gives an idea what kind of figures can be generated with matplotlib.

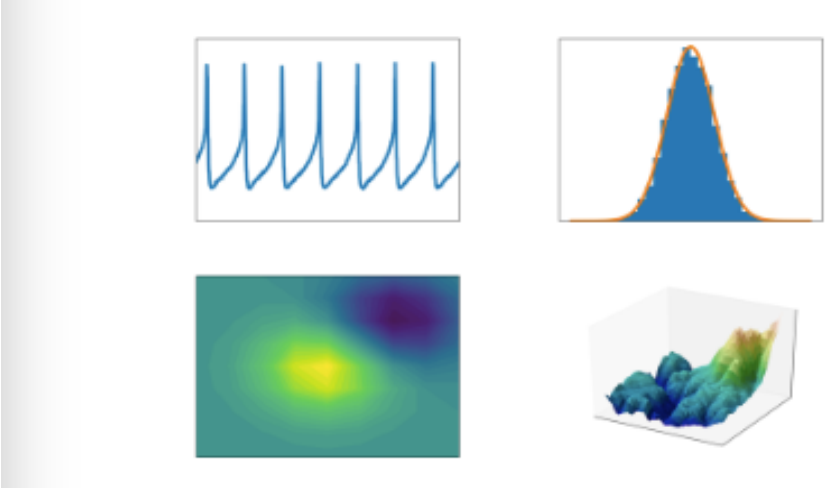

*FIGURE 16. Matplotlib visualization. (**Error! Reference source not found.**)*

6 IMPLEMENTATION

6.1 Design of the project

The aim of the project was to develop a program to identify objects based on the category it belongs to. In this project the objects to be identified were used batteries that belong to categories: Alkaline, NIMH, NICD, LIION and unknown battery. Two different options for battery recognition were suggested: object recognition or text detection. The project team contained many members whose approach to the project was different. The author's implementation part for the project only contained object recognition with CNN.

Before the actual implementation, all of the technologies were downloaded. The technologies used in the project are described in the chapter 4 Technologies and tools. After that, data to be categorized with neural networks was gathered; a video of the batteries was recorded with a phone camera and it was split into image frames. Images were used to train, test and validate the CNN model that predicts into which category the battery belongs to (Alkaline, NIMH, NICD, LIION or unknown battery). An overview of the project is shown in the figure 17.

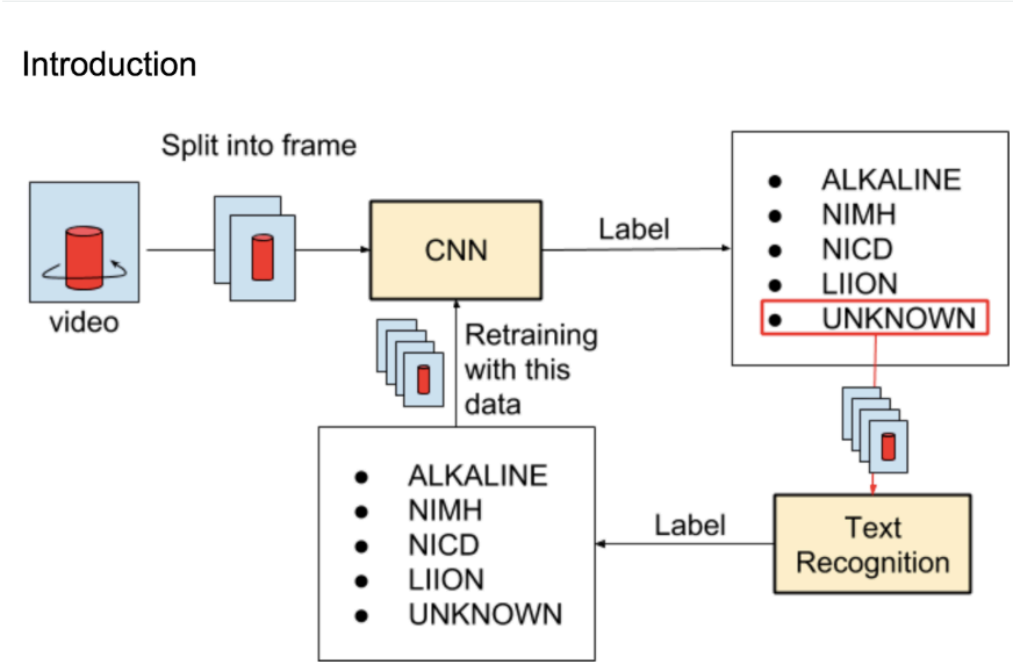

FIGURE 17. Overview of the project.

6.2 CNN models

In the project different models and processes were used to find out the best possible solution for battery recognition. In addition to own model based on TensorFlow, pre-trained models, such as inceptionv3, ResNet50, and VGG models with Keras were used to find out if the results and accuracy of own model is relevant. Inside these models many parameters and methods were included and many hours were spent to train all these different models. For testing purposes Jupyter notebook was also used.

In own model, the image size of 128 and the batch size of 32 was used and AdamOptimizer was used as an optimizer. The Optimizer, image size and batch size of own model as well as pre-trained models are represented in the table I.

TABLE 1. Models and optimizer, image size and batch size used in each model.

Models	**Own model**	**Inceptionv3**	**ResNet50**	**VGG**
Optimizer	Adam Optimizer	SDG	SDG	RMSproper
Image size	128	224	224	150
Batch size	32	32	32	32

In the implementation part of this thesis the own model is described but in the result part also the results of other models are compared with the results of the own model.

6.3 Project structure

The project structure contains the directories of the own model. The project directories are described below and shown in the figure 18.

- **Testing_data** directory contains the images of the batteries that are different from the training images.
- **Training_data** directory contains the images of batteries for training that are different from testing images.

- **Checkpoint** directory captures the exact value of all parameters.
- **Dataset.py** directory contains code for the manipulation and resizing of images to make it easy to load the images during training.
- **Predict.py** directory mainly works on loading images and reading images using OpenCV as well as restoring the trained model and finally predicting the new images.
- **Train.py** directory is the implementation of convolutional neural network and contains all the parameters, convolutional layers, max-pooling layer, fattened layer, fully connected layer, weight, biases and others required components for the convolutional neural network.
- **.meta** directory stores the graph structure.
- **.data** directory stores the values of each variable in the graph.
- **.index** identifies the checkpoint.

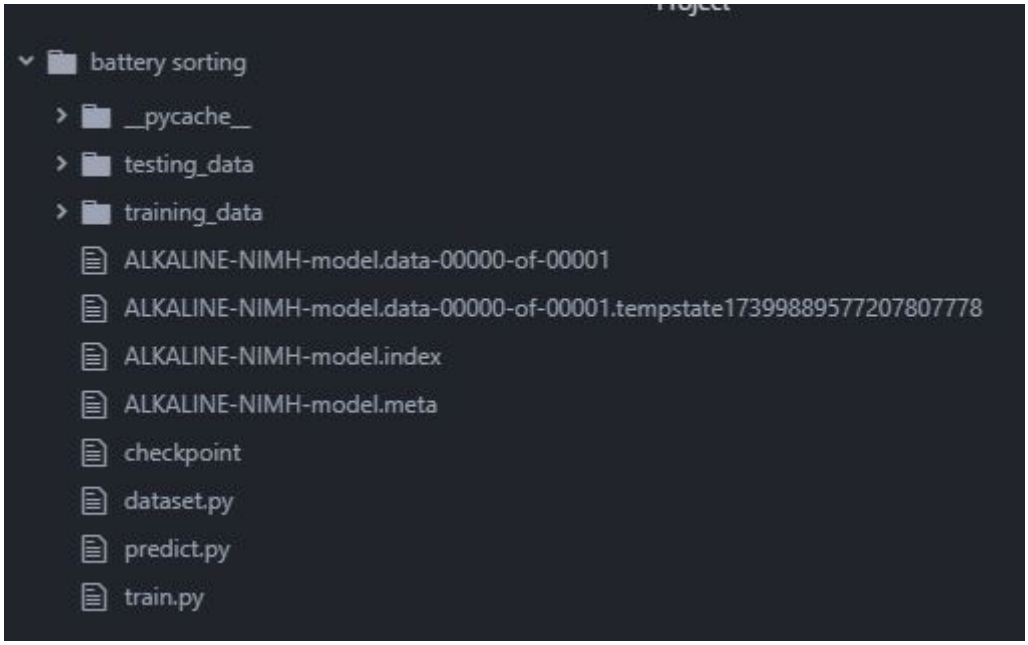

FIGURE 18. Project structure.

6.4 Preparing data by image splitting

All the data used in this project was collected during this project. After recording the video of different kinds of batteries, many frames were extracted. The video was recorded at home and working place by rotating table so the image could be taken from every angle. The quality and clarity of the image frames were an issue

due to lack of better tools because of funding and lack of experience with tools. Many thousand images were extracted from the series of recorded video using OpenCV.

The image splitting process is listed and explained briefly below.

i) **Importing packages:** Packages like OpenCV as cv2, numpy as np and OS module were imported. These packages deal with their own functions whenever they need them inside the program. All imported packages are shown in the figure 19.

ii) **Feeding video:** The video was fed to extract the frames in the same folder and the directory of the video was given in the program.A Video named example.mp4 has been linked with the function shown in the figure 19.

iii) **Running program:** After feeding the video the program which extracts the frames was run. The virtual environment and OpenCV were activated before running the program (shown in the figure 20). The program that was run created jpeg images inside the data folder shown in the figure 20. A video with 30 images per minute was extracted. Some of the extracted images are show in figure 21.

```python
import cv2
import numpy as np
import os

# Playing video from file:
cap = cv2.VideoCapture('example.mp4')

try:
    if not os.path.exists('data'):
        os.makedirs('data')
except OSError:
    print ('Error: Creating directory of data')

currentFrame = 0

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Saves image of the current frame in jpeg file
    name = './data/frame' + str(currentFrame) + '.jpeg'
    print ('Creating...' + name)
    cv2.imwrite(name, frame)

    # To stop duplicate images
    currentFrame += 1

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

FIGURE 19. Screenshot of Image splitting.

```
[(cv) (base) shaliks-Air:desktop shaliksapkota$ cd frame
[(cv) (base) shaliks-Air:frame shaliksapkota$ ls
example.mp4     splitframe.py
[(cv) (base) shaliks-Air:frame shaliksapkota$ python splitframe.py
Creating..../data/frame0.jpeg
Creating..../data/frame1.jpeg
Creating..../data/frame2.jpeg
Creating..../data/frame3.jpeg
Creating..../data/frame4.jpeg
Creating..../data/frame5.jpeg
Creating..../data/frame6.jpeg
Creating..../data/frame7.jpeg
Creating..../data/frame8.jpeg
Creating..../data/frame9.jpeg
```

FIGURE 20. Extracting image out of video.

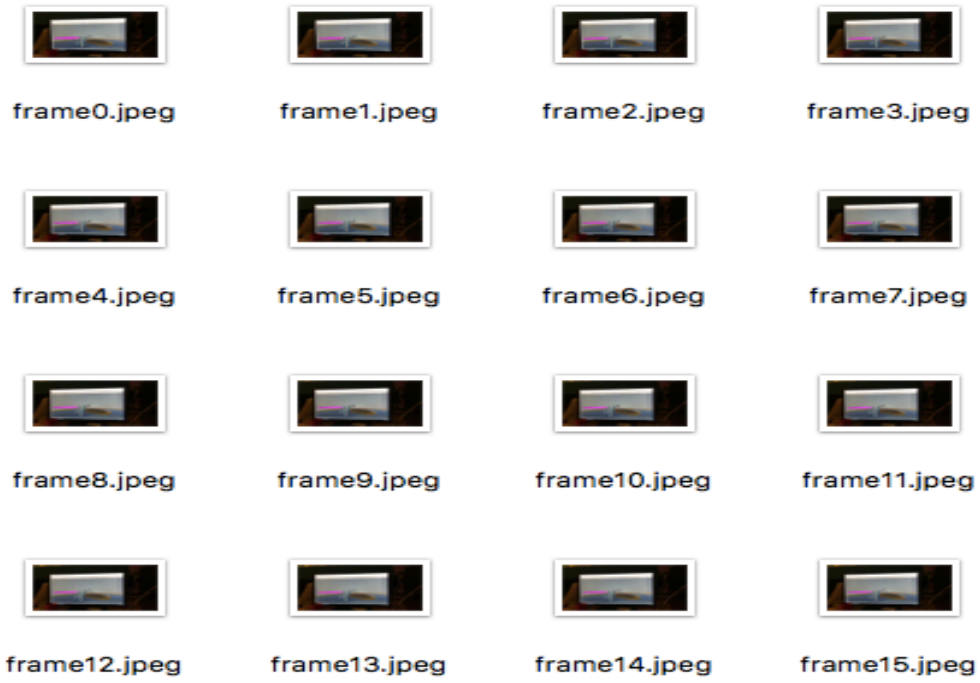

FIGURE 21. Extracted images from video.

6.5 Training and test datasets

There were two kinds of datasets inside the classification process: training and testing datasets. The training set was a subset to train a model, whereas the test set was a subset to test the trained model. The ratio of the training set and the testing set is approximately 80:20 as shown in the figure 22.

While preparing the training dataset, a different training dataset was produced with new batteries, noise batteries (labelled slightly unclear) and new frames (both new and noise batteries and also from the different angles, light, distance and quality). All datasets were labelled and scaled properly. For the testing dataset all three kinds of categories were mixed.

The datasets need to meet three conditions in order to get good results while training and testing.

1) Both training and testing data must be sufficient to yield statistically meaningful results.

2) The same data cannot be used in both training and test dataset.

3) Training set data cannot be from a different category than being trained.

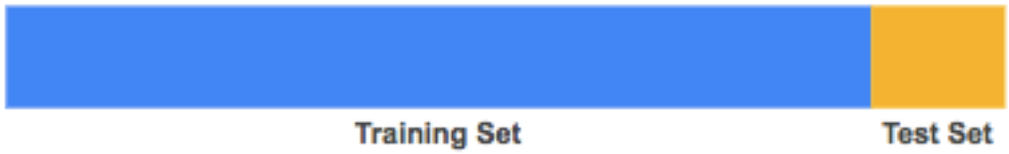

FIGURE 22. Ratio of training and testing dataset. (Medium, cited 15th October 2018)

6.6 Importing libraries and datasets

First of all, the libraries required during the implementation process, such as Tensorflow, Numpy, Panda, time, scikit-learn were imported. All imported libraries are shown in the figure 23. Also, datasets were imported using import method (figure 23).

```
import dataset
import tensorflow as tf
import time
from datetime import timedelta
import math
import random
import numpy as np
import os
import matplotlib.pyplot as plt
import pandas as pd
#%matplotlib as inline
from sklearn import preprocessing
import cv2

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
##disable this if it doesn't work
#Adding Seed so that random initialization is consistent
from numpy.random import seed
seed(1)
#edited from hash
from tensorflow import set_random_seed
set_random_seed(2)
#edited from hash
```

FIGURE 23. Import libraries and dataset.

6.7 Setting algorithm constant parameters

While creating and running algorithms, constant parameters that do not change during the process, are needed. Constant parameters, batch size, learning rate and maximum training steps were set to 32, 0.0005 and 1000, respectively (figure 24).

```
batch_size = 32
learning_rate = 0.0005
max_steps = 1000
```

FIGURE 24. Setting algorithm constant parameters.

The validation size was set to 0.2, image size to 128 and number of channels to 3, and the training data was set to the training path (figure 25).

```
validation_size = 0.2
image_size = 128
num_channels = 3
train_path='training_data'
```

FIGURE 25. Setting the parameters.

6.8 Creating CNN parameters

Many different parameters were created to use them later in the convolution neural network process. Filter sizes were initially set to 3, but during the process the sizes were changed when needed. Similarly, the number of filters and size of the fully connected layer were created and modified later on when needed (figure 26).

```
##Network graph params
filter_size_conv1 = 3
num_filters_conv1 = 32

filter_size_conv2 = 3
num_filters_conv2 = 32

filter_size_conv3 = 3
num_filters_conv3 = 64

fc_layer_size = 128
```

FIGURE 26. Creating CNN parameters.

The weight was generated from a truncated normal distribution with a standard deviation value of 0.05 (figure 27).

```
def create_weights(shape):
    return tf.Variable(tf.truncated_normal(shape, stddev=0.05))
```

FIGURE 27. Creating weight.

Biases, which help to control the value at which the activation function will trigger, were created (figure 28).

```
def create_biases(size):
    return tf.Variable(tf.constant(0.05, shape=[size]))
```

FIGURE 28. Creating biases.

A convolutional layer model was created (figure 29). Input, number of channels, convolutional filter size and number of filters are the parameters inside the convolutional layer model.

```
def create_convolutional_layer(input,
               num_input_channels,
               conv_filter_size,
               num_filters):
```

Figure 29. Creating convolutional layer model.

A max pooling layer was created with a value, k-size, strides and padding. The-output of pooling layer was fed to Relu, which is an activation function. (figure 30)

```
    ## We shall be using max-pooling.
    layer = tf.nn.max_pool(value=layer,
                           ksize=[1, 2, 2, 1],
                           strides=[1, 2, 2, 1],
                           padding='SAME')
    ## Output of pooling is fed to Relu which is the activation function for us.
    layer = tf.nn.relu(layer)

    return layer
```

Figure 30. Max pooling layer.

A flattened layer was created to shape the layer that was received from the previous layer (figure 31).

```
def create_flatten_layer(layer):

    layer_shape = layer.get_shape()

    num_features = layer_shape[1:4].num_elements()

    layer = tf.reshape(layer, [-1, num_features])

    return layer
```

Figure 31. Flattened layer.

A multilayer convolutional model, which can use given parameters, was created (figure 32).

```
layer_conv1 = create_convolutional_layer(input=x,
               num_input_channels=num_channels,
               conv_filter_size=filter_size_conv1,
               num_filters=num_filters_conv1)
layer_conv2 = create_convolutional_layer(input=layer_conv1,
               num_input_channels=num_filters_conv1,
               conv_filter_size=filter_size_conv2,
               num_filters=num_filters_conv2)

layer_conv3= create_convolutional_layer(input=layer_conv2,
               num_input_channels=num_filters_conv2,
               conv_filter_size=filter_size_conv3,
               num_filters=num_filters_conv3)
```

FIGURE 32. Multi-Layer convolutional model.

A Layer was defined which is a matmul (matrix multiplication) function in Tensorflow. The matmul function takes x as an input and produces $wx + b$. The relu activation function was also used after the layer had been processed. (figure 33)

```
137     layer = tf.matmul(input, weights) + biases
138     if use_relu:
139         layer = tf.nn.relu(layer)
140
141     return layer
142
```

Figure 33. Layer as matmul multiplication.

6.9 Parameters for loading and saving

A saving model was created by the saver.save function giving the folder name ALKALINE-NIMH-model. (figure 34)

```
217             saver.save(session, './ALKALINE-NIMH-model')
```

Figure 34. Saving model by saver session.

Global variables were initialized to allow TensorFlow to run all existing variables in the program (figure 35).

```
182 session.run(tf.global_variables_initializer())
```

Figure 35. initializing global variables.

AdamOptimizer was initialized as an optimizer and used for improving the speed and performance for a model that has to be trained (figure 36).

```
177   optimizer = tf.train.AdamOptimizer(learning_rate=5e-4)
```

Figure 36. Initialize AdamOptimizer as Optimizer.

A cost function called cross entropy was set and used to calculate the total loss of model. A reduce mean method was used. (figure 37)

```
176 cost = tf.reduce_mean(cross_entropy)
```

Figure 37. Cost function.

The get_default_graph function was set to access the default graph. (figure 39)

```
64  graph = tf.get_default_graph()
```

Figure 39. Accessing the default graph.

A Function named tf.session was set to restore the saved models. (figure 40)

```
57  sess = tf.Session()
```

Figure 40. Restoring the saved models.

A Function named saver.restore was used to load the weights saved the earlier ones. (figure 40)

```
60  # Step-2: Now let's load the weights saved using the restore method.
61  saver.restore(sess, tf.train.latest_checkpoint('./alkaline-nimh data'))
```

Figure 41. Loading the weights saved using the restore method.

7 RESULTS

During the training and testing, according to the plan, the first data was gathered with new batteries, then with noise batteries and at last with new frames. Initially new batteries images were introduced with models and later labelled batteries were introduced which were slightly taped or scratched. The idea behind the noise images was to find out the more realistic data that fits for real life images of used batteries. The images set of new frames images contains both new batteries and noise batteries from the different angle, light, distance and quality. All images were scaled and labelled. Noise data and new frame data were bigger than new batteries data. Testing datasets were from used batteries. During the testing already learned models were also used. New batteries trained models were used while trained and test with noise batteries datasets.

First the own model was tested with new batteries in the training dataset resulting to the accuracy of 59.4%. With noise batteries, which were taped to make the label unclear, the accuracy was 84.4%. When new batteries and noise batteries were combined in a new dataset (new frame), the accuracy reached to 100%. 100% accuracy was achieved only with testing data, not with real life testing.

In other models the same datasets were used but different results were obtained. The results of own model as well as other models are represented in table 2.

TABLE 2. Accuracy of different battery datasets achieved by different models.

Models		**Own model**	**Incep-tionv3**	**ResNet50**	**VGG**
New batter-ies		59.4 %	79.18 %	58.03 %	21.91 %
Noise batter-ies	*Alkaline*	84.4 %	18.97 %	44.61 %	32.14 %
	LiIon	84.4 %	82.27 %	100 %	100 %
	NiMH	84.4 %	93.22 %	89.26 %	96.61 %
	Nicd	84.4 %	-	-	-
New frame	*Alkaline*	100 %	99.73 %	-	32.1 %
	LiIon	100 %	100 %	-	100 %
	NiMH	100 %	100 %	-	96.61 %
	Nicd	100 %	100 %	-	-

8 CHALLENGES

In our project system there was no problem while testing the system inside the computer with command a line interface. The problem of our system was testing in real life environment as only low accuracy, averagely below 50%, was obtained. The project system was tested in a real life environment, where low resolution web cameras with OpenCV and tesseract were used.

The probable causes of low accuracy in real life environment testing includes light angle, motion of battery, camera quality and low processing power. The low resolution web cameras were not able to take clear and bright images of batteries, which decreases the efficiency of object detection by our system. The processing power of computer was also inefficient for the system to detect an object.

Also, most of the tools used in our project are targeted for specialists and researchers, so testing and installing tools before the actual implementation was challenging and time-consuming.

9 FUTURE WORK

The Main focus of this thesis work was to see and experiment how object detection and image classification work in real life, although it is hard to get a good result without a huge amount of data.

The Future work of this project includes producing and using better data, usage of better tools and methods and implementation of better models. To make our project system usable, the user interface and human usable system need to be developed. The user must be able to add more data, where models and result can be seen. Also, the statistics should be saved and re-used when needed to make the system effective. The mechanical part of the system is another factor to be improved. It needs to be built with convey belt, high-resolution cameras, sensors and CPUs/GPUs.

10 CONCLUSIONS

Increasing the productivity using machine learning algorithms is becoming a trend in major industrial sectors. For a better industrial progress, highly successful machine learning models are needed to reduce the cost of production. Object and image classification can be a break-through when generating new records in accuracy and reducing human errors. In the field of AI, accuracy of the specific task decides the customer's satisfaction. Higher accuracies are especially important when it comes to case sensitive tasks such as batteries, medicines and disease diagnosis as a wrong classification can lead to an accident or even cost a human life.

My thesis project could be one of the methods in AI to improve waste management, not only with batteries but also with other waste. The object and image detection system developed can be used for various other purposes as well. That is why my research work is not only valuable for battery recycling company but also for all the manufacturing companies that use some form of sorting and arrangement of object or classification of images. Furthermore, this project can be improved in the future with a better model, better data and additional training. I would say that this bachelor's thesis was successful and productive in terms of its usefulness and futuristic approach.

REFERENCES

1. The History of Artificial Intelligence, cited 06th of January 2019. https://en.wikipedia.org/wiki/History_of_artificial_intelligence
2. Artificial intelligence general study, cited 06th of January. https://en.wikipedia.org/wiki/Artificial_intelligence
3. Elon musk and Stephen hawking on AI, Cited. 22nd of April 2019 https://www.theverge.com/
4. Schwab K.;(2017). The fourth industrial revolution. p.100-300
5. Understanding of CNN – Deep Learning cited 15th October 2018 https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148
6. How the computer beat the go master cited 15th October 2018 https://www.scientificamerican.com/article/how-the-computer-beat-the-go-master/
7. 5 ways AI is changing education. Cited 06th of January 2019 https://elearningindustry.com/ai-is-changing-the-education-industry-5-ways
8. Artificial intelligence in Healthcare. Cited 06th of January 2019 https://en.wikipedia.org/wiki/Artificial_intelligence_in_healthcare
9. Artificial intelligence in logistic, cited 06th of January https://www.logistics.dhl/content/dam/dhl/global/core/documents/pdf/glo-ai-in.logistics-white-paper.pdf
10. The role of Artificial intelligence in Cybersecurity, cited 06th of January https://biztechmagazine.com/article/2018/06/role-artificial-intelligence-cyber-security

11. What is machine learning? Cited 06th of January 2019 http://theconversation.com/what-is-machine-learning-76759
12. Google developer. Introduction to machine learning, cited15.10.2018. https://developers.google.com/machine-learning/crash-course/ml-intro
13. Russell, Stuart J.; Norvig, Peter (2010). Artificial Intelligence: A Modern Approach (Third ed.). Prentice Hall. ISBN 9780136042594. P.695-697
14. Hinton, Jeffrey; Sejnowski, Terrence (1999). Unsupervised Learning: Foundations of Neural Computation. MIT Press. ISBN 978-0262581684. P.1-19
15. Kaelbling, Leslie P.; Littman, Michael L.; Moore, Andrew W. (1996). "Reinforcement Learning: A Survey". Journal of Artificial Intelligence Research. P.237-285
16. Reinforce learning. 5 things you need to know about Reinforcement learning. Cited 06th of January. https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html
17. Convolutional neural network with Tensorflow, cited 15th October 2018 https://www.datacamp.com/community/tutorials/cnn-tensorflow-python
18. Understanding of CNN – Deep Learning cited 15th October 2018 https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148
19. Train a convolutional neural network as a classifier, cited01.01.2019 https://machinelearningmindset.com/blog/
20. A summary of neural network layers. Cited 15th October 2018. https://medium.com/machine-learning-for-li/different-convolutional-layers-43dc146f4d0e
21. Understanding locally connected layers in convolutional neural networks, cited 06th January 2019 https://prateekvjoshi.com/2016/04/12/understanding-locally-connected-layers-in-convolutional-neural-networks/
22. TensorFlow, cited 15th October 2018. https://www.tensorflow.org/

23. Python. Python documentation, cited 15th October 2018. https://www.python.org/
24. Keras. The Python Deep learning library cited 06th of january2019. https://keras.io/
25. Jupyter Notebook open source web application, cited 15th of October. https://jupyter.org/
26. Opencv library, cited 15th October 2018 https://opencv.org/
27. Atom. one of the finest and hackable test editor for the 21st century, cited 06th of January. https://blog.atom.io/2014/02/26/introducing-atom.html
28. Conda. Open source package management system, cited 06th of January. https://conda.io/docs/index.html
29. Pandas, Pandas documentation, cited 06th of January. http://pandas.pydata.org/pandas-docs/stable/overview.html
30. Numpy. Numpy documentation, cited 06th of january. http://www.numpy.org/
31. Matplotlib. Matplotlib documentation, cited 06th of January. https://matplotlib.org/
32. Linux. Operating system, cited 15.10.2018. https://www.linux.com/
33. Git. Free and open source distributed version control system cited 06.01.2019. https://git-scm.com/