Professional MBA
Automotive Industry

Enabling automation with deep learning at optical X-ray inspection station succeeding various soldering processes in electronic control units manufacturing

A Master's Thesis submitted for the degree of
“Master of Business Administration”

supervised by
Univ.Prof. Dr.Ing. Prof.eh. Dr.h.c. DI Wilfried Sihn

Lic. Dorian-Mihai Toparcean

11742793

Vienna, 11.03.2020

Affidavit

I, **LIC. DORIAN-MIHAI TOPARCEAN**, hereby declare

1. that I am the sole author of the present Master's Thesis, "ENABLING AUTOMATION WITH DEEP LEARNING AT OPTICAL X-RAY INSPECTION STATION SUCCEEDING VARIOUS SOLDERING PROCESSES IN ELECTRONIC CONTROL UNITS MANUFACTURING", 84 pages, bound, and that I have not used any source or tool other than those referenced or any other illicit aid or tool, and
2. that I have not prior to this date submitted the topic of this Master's Thesis or parts of it in any form for assessment as an examination paper, either in Austria or abroad.

Vienna, 11.03.2020

Signature

Abstract

This master thesis describes a general framework, a development environment and a step by step process based on case study, to facilitate implementation of deep learning in electronic control units manufacturing industry focusing on optical failure detection at automated optical inspections processes after soldering processes. The approach comes here from the desire to encourage and support any electronic manufacturing company to benefit from applying latest advances in artificial intelligence, focusing on deep learning with convolutional neural networks, because access to environment and high level libraries over powerful frameworks are more than ever simplified, easy accessible, open source, based on cloud computing where also the needed environments is configured and ready to use at very attractive cost. Furthermore, this thesis will demonstrate the benefit of how automated decision making can be enabled by trained neuronal networks and prove that the knowledge bar is now inclined towards process experts with hands on business and extended analytical skills more then on classical practitioners having programming skills and mathematics and statistics knowledge only.

This master thesis sets a special focus on image recognition making use of convolutional neural network. As a starting point Google and Facebook open source machine learning libraries, TensorFlow and PyTorch, have been studied, developed by artificial intelligence research groups of corresponding companies. These are offering incredibly powerful models for computer vision applications.

Indeed, it was observed that convolutional neural networks are the most suitable for analysing visual images. The case study will demonstrate that the ResNet50 trained convolutional neural network is able, with an accuracy of 0.93, to classify the optical images acquired by the X-ray system, enabling sorting them according to defined classes and in the end enabling automated decision according to pass fail criteria.

Keywords: deep learning in manufacturing electronics, neural networks, convolutional neural networks, machine learning, artificial intelligence, multiclass classification.

Acknowledgment

I would like firstly to thank my thesis advisor, Univ.Prof. Dr.Ing. Prof.eh. Dr.h.c. DI Wilfried Sihn, Professor of Industrial Engineering and System Planning at the Institute for Management Sciences at Vienna University of Technology, in the same time in the Board of Directors of the mentioned Institute and Managing Director of Fraunhofer Austria Research GmbH. He rightly focused my initial broad subject. I appreciated all along the coaching experience, the way he encouraged me and intrusted me with freedom to act.

Moreover, I would like to thank to my supervisor within Continental, Dr. Sami Krimi, heading the Automotive Manufacturing within Continental Automotive, one of the initiators of the start-up Data Analytics within Continental Electronic Plants, who inspired me and gained me as an active contributor since very beginning.

I would also like to acknowledge Mr. Oswald Kolb, General Manager of Continental Automotive Sibiu, who immediately recognize the technological advantages of deep learning and supported with its Focus Factory Transmission team my proposed case study. It was great to have the opportunity to work together with Sibiu colleagues, Head of Industrial Engineering and Information Technology, Mr. Cosmin Sideras, Smart Factory Engineer, Mr. Dragos Toma, Test Engineer, Mr. Adrian Lascu and many other wonderful former colleagues of mine, at the implementation and roll out of the image model classification described in the case study.

I am also grateful to my management at Continental Vienna and Continental Regensburg, Mr. Alem Helias, General Manager of Continental Automotive Austria GmbH and Mr. Ronald Rossbacher, Head of Operation Transformation and Head of Life Cycle Management within Business Unit Connected Car Networking, for their continuous support and my active enrolment to this MBA program.

And finally, last but by no means least, I must express my gratitude to my wife Alexandra and my two children, Alex and Raul, and of course my dear mother, Meluzina-Maria, and my younger brother Daniel. This is my dear family which I was depriving off care, affection and quality moments during my studies. This accomplishment would not have been possible without them. Thank you.

LIST OF ABBREVIATIONS

Acronym	**Definition**
AGI	Artificial General Intelligence
AI	Artificial Intelligence
AlexNet	Alex (Krizhevsky) Network
ANI	Artificial Narrow Intelligence
ANN	Artificial Neural Network
ASI	Artificial Superintelligence
CNN	Convolutional Neural Network
CP /CPK	Process Capability
CPU	Central Processing Unit – Main Processor
DL	Deep Learning
DPMO	Defects per Million Opportunities
ECU	Electronic Control Unit
GD	Gradient Descent
GoogLeNet	Google Network Architecture based on LeNet
GPU	Graphics Processing Unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IP	Intellectual Property
IT	Information Technology
LeNet	LeCun Network Architecture
LR	Learning Rate
ML	Machine learning
MLP	Multilayer Perceptron
MSE	Mean Squared Error
MSRT	Microsoft Research Team
PPM	Parts per million
ReLU	Rectified Linear Unit

ResNet	Residual Network Architecture MSRT
RGB	Red Green Blue
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SVMs	Support Vector Machines
USP	Unique Selling Proposition
VGGNet	Visual Geometry Group Network
X-Ray	Acquisition equipment of electromagnetic radiation in form of Images
ZFNet	Zeiler and Fergus Network

Notation

Numbers:

i. a : A scalar

ii. x: A vector

iii. X: A matrix

iv. x_i : The *i*-th element of vector x

v. x_{ij}: The element of matrix X at row *i* and column *j*

Set Theory:

i. N: The set of natural numbers

ii. Z: The set of integers

iii. R: The set of real numbers

Functions and Operators:

i. $f(\_)$: A function

ii. $log(\_)$: The natural logarithm

iii. e^x: The natural exponential function

iv. *T*: Transpose of a vector or a matrix

v. $*$ convolution operator

vi. $\sum$: Series addition

Calculus:

i. $\frac{dy}{dx}$: Derivative of y with respect to x

ii. $\frac{\partial y}{\partial x}$: Partial derivative of y with respect to x

iii. $\nabla\, y_x$: Gradient of y with respect to x

iv. $\int_a^b f(x)dx$: Definite integral of f from a to b with respect to x

v. $\int\, f(x)\, dx$: Indefinite integral of f with respect to x

Probability and Information Theory:

i. $P(\_)$: Probability distribution

ii. $p(x)$: Probability density function

iii. $H(X)$: Entropy of random variable X

iv. $H(y, p)$: Cross-entropy discrete probability distribution

Table of Content

Chapter 1. Introduction

Automotive Industry has recently entered in an era of transformation since the requirements around the automobile are more demanding than ever. Megatrends like connectivity and digitalization, electrification, internet of things are paving the way towards fulfilment of most demanding customer requirements in terms of comfort, safety, autonomous driving, connectivity, life style and probably the most urgently addressed: environment. Automotive manufacturing is as well revolutionized by Industry 4.0 and artificial intelligence beginning to make its way in various application. As part of the supply chain, electronic manufacturers are stringently asked to contribute to increase functionality demand and real time communication. Thus, the electronic units in the automobile are more and more complex with high computing power and intelligent antennas integrated so that they can connect to outer environment, being other automobiles, cities infrastructure and trough internet to every source of information broadcasted. With this extended and constantly increase of automobile capability, the autonomous driving is being set as an immediate target for OEMs. The benefits are immense if we think only at safety, comfort and in the end the time we can gain! Of course, they are many more. In order to achieve autonomous driving, the OEM's must see the environment in real time and take instant decisions on how to respond to surrounding and not only. Therefore, we need to see the environment, analyse it and decide in consequence how the automobile should behave. To achieve that the humans turned to its own creation, AI, and its special domain of deep learning. Through it, the automobile, equipped with different sensors, cameras and radars, is sensing the environment and sending this information to a brain, simulated trough deep neural networks, which can identify the traffic situation as a human does. Thus, seeing the environment, it is then to decide how to act in consequence. On a very abstract way, this is how autonomous driving application is behaving.
The thesis focuses on deep learning introduction in electronic manufacturing's and tries to add value through production chain, offering alternative methods to classical problem-solving approach, where socio-systems have the human as main decision maker, which by visualizing and analysing specific situation, is acting to what he 'sees' or senses.

1.1. Problem description

In the company's manufacturing electronics control units, there are mainly two separated focused production areas. First one is the so-called front end, where printed circuit board assembly takes place, meaning that electronic components are mounted by surface mounted devices on solder paste printed circuit boards and reflow soldered than inspected with optical inspections systems by cameras or X-ray. The second one is the final assembly area where product is being mechanically assembled, tested and finally packed for delivery. Here we find as well soldering of various connectors, sensors, cables or selective coating processes followed usually by optical inspections stations. Highly skilled and experienced operators are making real time decision after analysing the solder joints pictures in order to avoid wrong, not enough filled, missing solder joints on solder bridges or other defect types. Due highly concentration demand and 24 hours continuous production this inspection position is prone to slip through defects which can cause severe damages by wrong classification.

On this specific optical quality gates there are complex visual catalogues which serve the purpose of categorisation of pass-fail criteria's and comprise from different defect classes or border line acceptance criteria. This one is updated continuously as new defect possibilities are identified or as a consequence off a costumer claim and retraining sessions are initiated with involved personal.

For each unit tested, data in form of pictures are captured then send for analyse to technician which in the end decides if unit is pass or fail. All data are stored but not used, but rather kept for possible post claims purposes. Manufacturers have difficulties in associating data with the referred catalogue or acceptance criteria, in general data is not used at its potential or even misused. Many companies do not think or do not know that AI can play an decisive role in this type of image processing classification. They have no knowledge nor resources to understand AI so we cannot talk yet about a framework or a general structural approach in identifying applicability and benefits.

1.2. Master Thesis main research Questions

Therefore, the thesis focuses in bridging academic research in AI image recognition with a practical application in the industry. The driving questions will be:

1) When and for which category of problems can we successful apply deep learning in electronic manufacturing production
2) A trained convolutional neuronal network in X-Ray image recognition can replace human driven classification at the optical inspection stations, with other words, is the solution based on deep learning qualitatively more efficient, quicker and reliable then socio-technical system? When should a clear cut occur?
3) What are the generic steps to be taken as a successive approach to enable such technical self-deciding systems?

The focus will be to research one of the most powerful open source deep learning frameworks like PyTorch, which is providing pre-trained neural networks in image recognition, and look for possible reuse and applicability of solving in production real time classification problem of failed units, rather than programming a neural network from scratch. Having these guiding questions as driving elements, thesis will focus in describing a general framework based on the step-to-step process flow approach to solve the sorting and classification problem. With the other words the algorithm and corresponding functions used to solve the case study will be generalized and use a standardized approach to solve classification on optical inspection station.

1.3. Scope of the Thesis

The thesis will focus in connecting academic research in AI image recognition with a practical application in the electronic production and set the general approach for development and integration in a productive environment of such deep learning application.

In this effort, the Fastai and PyTorch libraries will be used to do deep learning under the interactive development tool Jupyter Notebook. The focus will be set in gathering enough and relevant data and use it to (re)train a convolutional neural network with 50 layers, ResNet50, inherited from torch vision, on the so-called training set. This data containing a relevant sample size will be grouped on classes representing also resulting evaluated vector. After a successful validation on a validation data set, this trained algorithm will be activated in production initially as a support for decision pass or fail.

The thesis will focus on step-by-step problem solving based on case study and draw a framework to approach similar problems. Neural network functionality will be briefly touch as a prerequisite of understanding the deep learning with a focus on convolutional neural networks.

1.3.1. Out of Scope

The thesis will not cover Python programming language and code explanation, nor detailed functions explanation. The same statement will apply to Fastai library function. No software installation nor explicitly used GPU capabilities will not be revealed as this are dependent on each studied case. There will be as well no qualitative measurement of automated program against the current socio-system cost even though the calculation can be easily deducted. To reproduce and develop own application certain programming and business skills are required, therefore not anyone reading this thesis can simply formulate and solve similar problem. Those having this initiative are encouraged to seek for relevant curses, like Fastai online course so that they can scan and figure out the capabilities they have. More details are provided in last chapter, Chapter 6, 6.2. Recommendation, but also in Chapter 5, 5.1. General process development.

1.4. Methodological Approach

Studying the literature in the domain of deep learning, reveals that the discipline it started to become relevant beginning with 2006. It has been started in academic environment and heavily invested in lately because of proven benefits in different fields like: autonomous driving and object recognition, voice control or speech recognition in smart phone, image recognition and medical diagnostic and many others.

A significant performer and enabler to deep learning success is the gaming industry. As deep learning requires huge computing power, companies like NVIDIA and AMD are supporting here with GPU computing capabilities. Without this computing power, it is hardly imaginable that deep learning popularity would have boomed like in our days.

This study focuses particularly on the Convolutional Neural Network as this are the models used to solve image classification in case study, particularly the residual neural networks.

The fundamental of my research was to gather the existing knowledge and find solved problems and applicability of deep learning image recognition in IT, social networking media, academic environment , medical diagnostics and connect it with automotive manufacturing by finding a practical application to a group of processes which will show relevance and not at last the benefit of being applied. Since I visited during my study quite a few suppliers in electronic manufacturing, I realised the industry is lacking in implementation of machine learning despite the fact there are many opportunities within industry where problem solving using deep learning could prove far more beneficial then robot process automatization or by suppliers developed customs system. I observed that the main reason why deep learning should be preferred in optical inspection application is that the data generated is very product specific and represents the company know how in certain problem-solving including design USP. Therefore, in-house development of such applications will also further protect IP, as data is not externalized, company can react on internal changes easily and extremely fast and most beneficial is that you are independent of a supplied solution which should be specified, defined and redefined as updates are occurring or adaption are needed due company updating or modifying its processes, changing its products or image acquisitions systems.

1.5. Thesis structure

In the Chapter 1. the motivation to introduce deep learning in automotive electronic production is described because of the benefits I foresee it brings with it, even though it is not a trivial task and it is quite a challenging path even for innovative companies. Herewith, the special application in image recognition with deep learning will be introduced with a quite challenging case study, trying to answer the driving research questions, making use of methodological approach of literature study, analysing current state of the art advances in image recognition in academic environment, industry, medicine.

Hence, powerful frameworks will be explored which are easing the way towards industrial application for image recognition using ready configured environments to run in the browser.

After a brief introduction in Chapter 2. of artificial intelligence, machine learning and deep learning and their relation, neural networks are detailed and gradual introduced in Chapter 3. From perceptron, a deep dive into neural network model will be made, then trough machine learning specific learning types, setting the base for multiclass classification problem solving. Training a model will be in detailed described, before introducing the powerful deep neural networks model architecture, the convolutional neural network.

The case study in Chapter 4. is being approached using the ResNet50 residual neural network model, which will be learned to identify soldering failures in the images acquired by an X-ray system, succeeding four separate process steps of soldering.

Chapter 5. will be a generalization in form of a derived framework and a process flow. It will pave the way to a standard approach with a defined environment to give a generic approach to problem solve similar cases in electronic manufacturing and other connected industries.

In the last chapter, Chapter 6. the research questions will be answered. Trough experience, observation or lesson learned, some limitations and recommendations will conclude the research study.

Chapter 2. Introduction to AI

To dive and understand deep learning, there is the need to describe and understand the artificial intelligence and machine learning. Hence, after introduction of AI, ML and DL, a summary how they are related is shown.

2.1. Artificial Intelligence

Artificial intelligence is defined as a field of study in September 1955 by a young assistant mathematics professor, John McCarthy, at Dartmouth College. He was also the organizer of the very first workshop where a study on artificial intelligence was carried out on following fields: computers, natural language processing, neural networks, theory of computation, abstraction and creativity. That workshop took place in summer of 1956 at Dartmouth College with an initial nominated group of 11 scientist.

The very first definition of AI belongs to him as well: '' It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable.'' (McCarthy 2007).

However, many inventors theorized on machines that are thinking long before computers even existed (Lovelace 1843).

In early days, AI tackled and solved various formal problems which are difficult to be solved by human beings. If those problems could have been described by a list of formal mathematical rules or algorithms then computers could solve them relatively easy, independent of complexity. An example of how artificial intelligence can over perform human intelligence was proved by IBM's Deep Blue chess-playing system which defeated the world champion Garry Kasparov in 1997 (Hsu 2002). The innovation factor in Deep Blue's design was that it was able to select the best possible movements by measuring their outcome rather than compute all possibilities. Without having this potential, Deep Blue would have needed to be an even more powerful computer. Similar, Google's Alpha Go, has a more sophisticated analyses method of deciding which potential moves to pursue and calculate. AlphaGo was able to equalize human expertise in 2015 (Silver 2016) using Deep Learning combined with the method Monte Carlo tree sampling. This concept will be widely introduced in Chapter 3.

This type of AI, on which systems or machines perceive a complex situation and choses among the most optimal opportunity from among possibilities is called ''reactive machines''. As the name implies, they are reactive to a perception rather than storing memories with the purpose to use them.

The mainstream tools of AI having the most potential ability to handle huge amount of data and perform complex calculation are machine learning and deep learning.

The second type of AI, 'limited memory machines', are able to memorize data for short period of time. They are limited in memorizing it temporarily and are not able to add it to a library of their experience. This type of AI is widely use in self-driving cars. Storing temporary data is the outmost need to create a life surrounding map of active objects and traffic indicators which are observed for a certain period of time in navigating through roads.

The third type of AI is being referred as 'theory of mind'. Researchers hope someday to build computers that simulate human relation and perceive human intelligence and also emotional impact on human by event or environments. As I could not find any reference or evidence of existence, I will state here that a minded computer does not exist yet.

'Self-aware machines' is the fourth type of artificial intelligence. This is rather the ultimate goal of AI development. The concept of a conscious machine, becoming truly self-aware is not likely to be realized soon.

'Artificial narrow intelligence' (ANI) represents the all existing AI created to date and represents the fifth type of artificial intelligence. This includes the most complex AI that uses machine learning and deep learning.

The sixth type of AI is the so called 'artificial general intelligence' (AGI). This represent the ability to of an AI agent to learn, perceive, understand and completely function like a human being.

The AI last classification type is the 'artificial superintelligence' (ASI). This will occur as AGI will become the most capable form of intelligence on earth. ASI will outperform the humans in whatever they do because of overwhelming greater memory, faster data processing and analysis, and decision – making capabilities. At this point in time, these machines may also threaten our very own existence (Joshi 2019).

2.2. Machine learning

As pointed out already, machine learning is a subset of AI. It is a method of training algorithms such that they can learn from the provided data and make accurate predictions finally being able to make decision.

No matter what kind of machine learning problem we are trying to solve or looking at, we must tackle its four core components (Zhang A. 2019):

(1) A *dataset*, representing the input for a model to learn from
(2) The *model* itself, which will transform and model the data

(3) A model evaluation function telling us how good our model actually is, called *loss function*
(4) An *algorithm* to minimize the loss function, by that updating the parameter set of the model, allowing it to learn

Therefore, a mandatory precondition for successful learning is the data set used for learning. Here, the bias effect must eliminate or reduce to minimum as this is for sure one of the driving factors for a successful application of machine learning (Shalev-Shwartz 2014).
To learn with Machine learning requires a model, which is going to make use of the data to make some prediction. In machine learning, a few models are able to act as described. The most relevant ones are:

(1) ANN's, or artificial neural network, my main research area in this thesis with a particular case of ANN, CNN
(2) Support Vector machines or SVMs, which is a binary and linear classifier
(3) Decision trees, or classification trees which is a predictive model
(4) Regression analysis, statistical method that relates input variables and their features. An example is later discussed in the thesis, as linear regression with MSE it is being exemplified, in order to find the line fitting best to a data distribution
(5) Bayesian networks, a graphical probabilistically method interrelating dependencies on variable set

Primarily, a machine learning algorithm has to have the ability to recognize meaningful patterns in the learning set of data. These algorithms are not to be limited to narrow set of rules as this will limit to static decision models and will not allow input data variation.
The first algorithms where programmers wrote hard coded rules that defined the behaviour of AI ware very popular in video games industry. Here it is desired that system delivers a predictable user experience.
However, this approach cannot be applied widely in voice recognition nor in image recognition as we cannot define logical rules to precisely describe every aspect of learning and behaviour in ways that can be transformed in computer rules (like if-then statements).
A very good example of problem probably not solvable with static programming approaches rules, would be the recognition of handwritten digits. Making use of a relatively simple neural network, a model to identify handwritten digits can be developed, which can learn from a given dataset, for example the well-known MNIST database (LeCun, Cortes and Burges, THE MNIST DATABASE 1998).

This specific neural network will develop its own behaviour of identifying handwritten digits and provide with a certain accuracy its own interpretation (Nielsen 2015).

2.3. Deep Learning

Deep learning is a subset of machine learning and bases its learning on deep neural networks. Typical learning on deep neural networks is learning by example. ‘’Most tasks consisting in mapping an input in a form of vector to an output vector, and that are easy for a person to do rapidly, can be accomplished via deep learning, given sufficiently large models and sufficiently large datasets of labelled training examples.’’ (Goodfellow, Bengio and Courville 2016).
A deep learning model can determine and adjust on its own if a prediction would be judged as being not accurate enough. (Buduma 2016).
One of the most important deep learning models is represented by the deep feedforward networks also called feedforward neural networks or multilayer perceptrons (MLP). This naming is related to how the information flows through the deep neural network only in one direction, forward, from input vector through network layers to the output vector. The convolutional neural networks are feedforward neural networks.
The recurrent neural networks, which are closer in functioning to our brain neurons network, are that feedforward neural network which include feedback connections between the network layers.

2.4. Summary on the relation between AI, ML and DL.

‘’The true challenge of artificial intelligence proved to be solving tasks that are easy for people to perform but hard for people to describe formally – problems that we solve intuitively, that feel automatic, like recognizing spoken words or faces in images.’’ (Goodfellow, Bengio and Courville 2016).
As described above, machine learning would have restricted capability to solve this complex handwritten recognition problem as its rather difficult to structure data so that an algorithm could clearly classify it. As a subset of Machine learning, deep learning has a better approach to tackle this challenge as it is using deep neural networks which are able to process huge amount of data.
To summarise, Machine learning is a subset of AI, and deep learning is a subset of Machine learning, as shown in Figure 1.

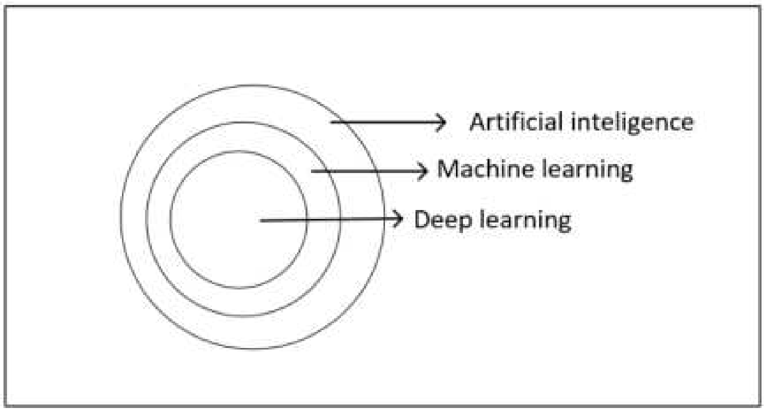


Figure 1 : Relation between AI, ML, DL

Using deep learning, the images to be classified will be send as input trough neural network layers and they are broken down to features which will be learned by the neural network. Those will be later recognized in new images so that the neural network can classify or identify them. A deep learning model is able to determine and adjust of its own if its prediction is not enough accurate. A typical model can be described as a feedforward deep network or multilayer perceptron (Goodfellow, Bengio and Courville 2016).

To summarize the main difference between machine learning and deep learning would be within feature extraction and classification, completely performed by deep neural network in deep learning but assisted in case of machine learning as it depends on algorithms which is parsing and learning from given data. The Figure 2 illustrates the different approach in image car recognition with machine learning compared with deep learning.

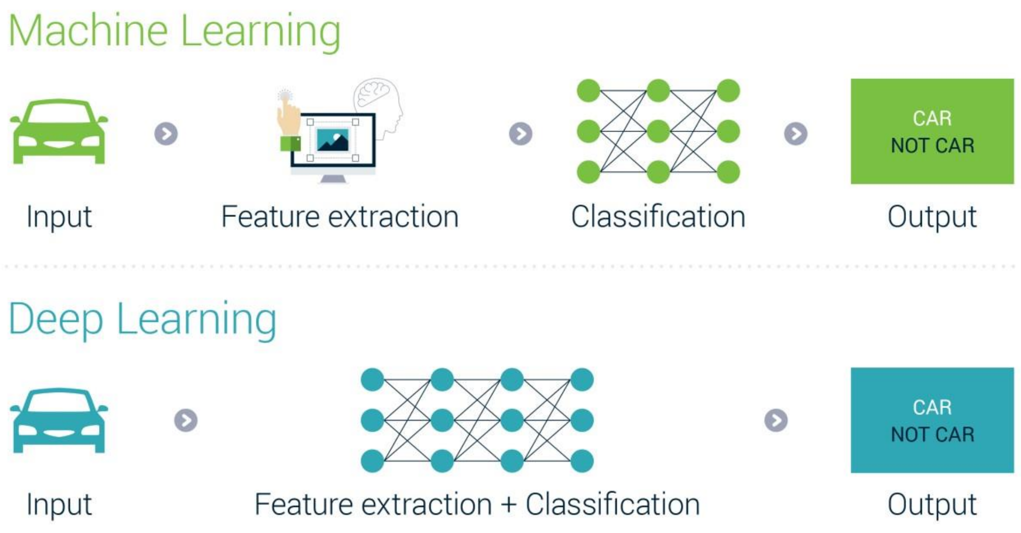


Figure 2 : Difference between Machine Learning Vs. Deep Learning (Inteliment 2019)

Chapter 3. Deep Learning with Convolutional Neural Network

3.1. Introduction to Artificial Neural Networks

In order to understand Neural Networks, also denoted Artificial Neural Networks (ANN), the artificial neuron concept will be introduced, which is based on the human neuron concept (Figure 3: Right) and the way it functions. In the human neuronal system, a neuron receives information's from connected neuron though dendrites connections. All these incoming signals are weighted summed up and then transformed into a new signal which will be transferred along axon to other neurons. (Buduma 2016).

3.1.1. Perceptron

Perceptron, developed by Frank Rosenblatt in 1950's, is simulating the human neuron function but restrict its inputs and output values to a binary 0 or 1.

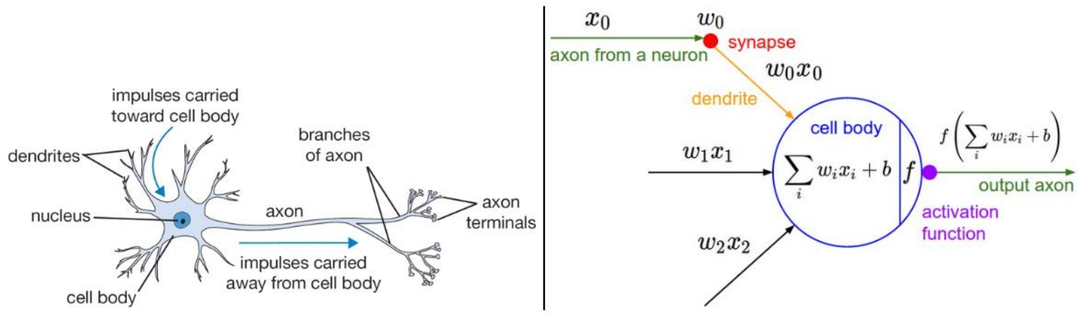


Figure 3: Biological neuron (left) and its mathematical model (right) (Karpathy 2019)

Considering a positive integer number of binary *inputs* $X_{1,}$, ..., X_n, Rosenblatt introduced real number called *weights* for each of them, w_1 , …, w_n , representing the importance of respective input to the output. The perceptron's output which can only be 0 or 1, as in Equation 1, it is being determined by whether the weighted sum is greater or lower than a real number called *threshold value*. This is also a neuron parameter. We can write the precise algebraic term in form of a function, called *activation function* (see Figure 3.):

$$output = \begin{cases} 0, & if \quad \sum_i w_i x_i \le threshold \\ 1, & if \quad \sum_i w_i x_i > threshold \end{cases}$$

Equation 1: Perceptron activation function

Further on, the term *threshold* will be replaced by *bias b*, where b ≡ - threshold. (see Figure 3). By varying the weights and the bias, the perceptron will change its output. In this way, different model of decisions can be achieved. Trying to imagine what a perceptron is, a correlation with being a device that makes decisions by weighing up evidence may help.

Another way we can use networks of perceptrons is to compute logical functions like AND, OR, NAND, XOR. This computational universality of perceptrons suggests that a network of perceptrons can be as powerful as any other computing device. Considering these properties, ‘’It turns out that we can devise *learning algorithms* which can automatically tune the weights and biases of a network of artificial neurons. This tuning happens in response to external stimuli, without direct intervention by a programmer. These learning algorithms enable us to use artificial neurons in a way which is radically different to conventional logic gates. Instead of explicitly laying out a circuit of NAND and other gates, our neural networks can simply learn to solve problems, sometimes problems where it would be extremely difficult to directly design a conventional circuit” (Nielsen 2015).

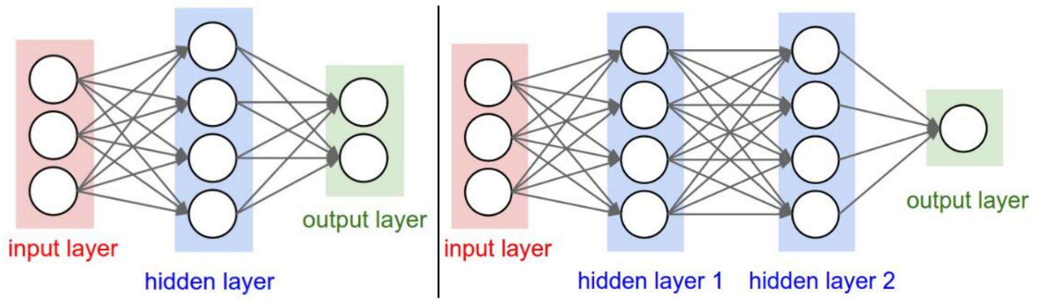


Figure 4: Left: A 2-layer Neural Network; Right: A 3-layer Neural Network (Karpathy 2019)

In the figure above, 2 different neural networks are shown. In the left side, there is a two-layer neural network with one hidden layer of 4 neurons and one output layer with 2 neurons. In the right side, a three-layer neural network with three inputs, two hidden layers of 4 neurons each and one output layer.

3.1.2. Sigmoid neuron

So far, we learned that activation function of a perceptron output is binary 0 or 1. Supposing we want to solve the handwritten digit classification using a network of perceptrons. In this case we will pass a vector of pixels containing the raw data of a picture containing a written digit and we would like that the network will correctly learn to classify that digit. And supposing the network misclassifying a picture of handwritten picture of '7' with '1'. Consequently, we will figure out how to make a small change in weights and biases (by changing the activation function), so the network improves its output. In this way the network will learn. This learning will affect also classification of the other digits, so we might solve the correct classification of one number, but because due new weights and biases the output of a perceptron might flip and cause an uncontrolled change in a different way.

Therefore, we want that the learning occurs at a lower rate, so the change in input only creates a small change in output instead of an unexpected result.

This can be achieved with the introduction of the sigmoid neuron. This is similar with the perceptron, but its output function called the sigmoid function, as in Equation 2, can be any value between 0 and 1:

$$\sigma(z) = \frac{1}{1+e^{-z}} \quad \text{where } z = \sum_i w_i x_i + b$$

Equation 2: Sigmoid neuron activation function: sigmoid function

For efficient learning and modelling of complex data, modern neural networks are using non-linear activation function like sigmoid function or logistic cost function.

Another example of activation function, which is widely use for convolutional neural network, the rectified linear (ReLU) as in Equation 3, (Jarrett, et al. 2009):

$$R(x) = max(0, x), where\ x \in R$$

Equation 3: ReLU activation function

3.1.3. Neural networks architecture

Until now we discussed about neural networks which have a layer called input layer holding respective input neurons and output layer holding respective output neurons. The layers in between are called hidden layers. It can be one or many more. (Figure 4). The multiple layer networks, like the one in Figure 4- Right, are called multilayer perceptrons or MLPs even though are made up of perceptrons or not. Furthermore, the output from one layer is used as input to the next layer, and there is no connection between neurons in the same layer. Such networks are called feedforward neural networks. (Figure 4).

However, feedback loops in between hidden layers are possible. Those specific neural networks using temporary feedback loops are called recurrent neural networks. These models have neurons which are temporarily activated, for limited time before becoming quiescent. The general idea is to fire more neurons in cascade. Loops do not cause problems in this model because a neuron output, which is input for the next layer neurons, affects its inputs sometimes later. (Chung, et al. 2015).

Even not in the scope of the thesis, as being part of RNNs, the bidirectional recurrent neural networks are mentioned here as they are important in speech recognition, handwriting recognition and bio informatics and another generalization of recurrent networks, recursive neural networks with application in computer vision. (Graves 2012).

3.2. Basics of Machine Learning

3.2.1. Types of learning

Before starting to introduce different leaning types in artificial neural network it will be probably better to explain the term learning in this context.

For humans, during the learn process it is supposed that neural structure of the brain is altered, increasing or decreasing its synaptic connections depending on their activity. In this regards, relevant information is easier to recall than older information. Relevant information has stronger synaptic connection. For the less relevant information, synaptic connection will timely weaken.

This model is simulated at a much lower scale by the artificial neural networks. Therefore, by adjusting the weighted connections found between neurons in the network this imitates the strengthening and weakening of this synaptic connection giving the network the ability to learn.

This ability to learn is based on learning algorithms. A machine learning algorithm has the ability to learn from data (Goodfellow, Bengio and Courville 2016).

Their objective is to find a set of weight matrices which applied to network will assign an input to a correct output.

There are three major machine learning types dependent on the data collected: supervised learning, unsupervised learning and reinforced learning (Goodfellow, Bengio and Courville 2016).

3.2.1.1. Supervised learning, regression and classification problems

In supervised learning, the goal is to learn a mapping from input to outputs on a given training set of data, consisting of *paired* labeled inputs – outputs. While processing an input, the actual output of the network will be compared with the desired output. The difference or the error calculated between desired output and actual output it can then be used to make corrections to the network by updating its weights and biases.

An algorithm which is able to extrapolate from an individual data set, containing a significant input and desired output pairs, and as a result can automate decision making process is a supervised learning algorithm. (Müller und Guido 2016).

The learner receives a training data set of paired inputs – outputs. After completing the training, a new set of inputs is provided, this time unlabeled, called test data set. This data are not specifically being labeled but as we provide them, we know what their corresponded label or output should be predicted. If this are correctly identified, we are having a certain confidence that our model is capable of correct classification. Very important for supervised learning is the data veracity: incorrect labeled data will weaken the ability of the model to generalize well.

For a correct prediction of the model in supervised learning, it is important that the available data is split over proportionally and aleatory in two data sets, one representing the training data and the other one the test data.

The supervised learning aims to solve two problems:

(1) regression problems

(2) classification problems

Regression models are useful in prediction of continuous values like the price of a house depending on size, distance to downtown, number of bathrooms or how many ice-creams will be sold while temperature is rising. In general, regression techniques are predicting continuous responses and address quantitative problems. A linear regression attempts to model the relationship between two variables by fitting linear equation to observed data. Returning to the example of how many ice-creams will be sold on a hotter day, a data will be collected about how many ice-creams are sold at different outdoor temperature. Temperature and ice-cream numbers are in this case the variables of this dataset. By regression analysis, one can relate them and start making predictions. In practice, it's highly unlikely to have a linear dependence between variables. In this case we will try to learn models to minimize the loss between prediction and actual or observed values, by introducing an error function. This can be found in literature as loss function or objective function or cost function. One example of loss function is the mean squared error (MSE), as in Equation 4 below:

$$MSE = \frac{1}{n}\sum_{i}(y_i - y'_i)^2$$

Equation 4: Mean Squared Error loss function

Classification models are predicting a qualitative response by analysing data in order to recognize a pattern. Usually, the data is categorized in classes. If we refer to handwritten recognition problem, then our model will analyse a feature vector, in this case the pixel values of the hand-written digit, and predict to which class, from '0' to '9', the given input belongs to. The simplest form of classification is when there are only two classes, a problem which is called binary classification. In binary classification, the prediction is closer to probability calculation. If we flip a coin and have a probability of 50% that it comes up a head, then the remaining probability for tail will be 1- probability that we obtain a head.

In case of handwritten digits, there are 10 possible classes an input image has to be assigned to. This problem is called multiclass classification. While on regression problems typically the focus will be to minimize the MSE loss function, the common loss function to be minimized for the classification problems is called cross-entropy (Zhang A. 2019). Cross-entropy loss is measuring the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. A perfect model would have a log loss of 0.

For example, an output layer z of a CNN classifying the handwritten digits, will be forced with SoftMax function, as in Equation 5, to be a discrete probability distribution over the k number of classes[1], where k in the digit classification problem is 10, corresponding to numbers from 0 to 9.

$$softmax(z) = \frac{e^z}{\sum_{i=1}^{k} e^{z_i}}, \; z = xW + b.$$

Equation 5: SoftMax activation function

For example, the expected result of a true labelled '3' corresponding to a handwritten 3 picture given to the model, will be formatted as a one - hot vector, which means that the value in the vector corresponding to the correct label will be set to 1 and all other positions will be set to 0:

$$Y(3) = (0,0,0,1,0,0,0,0,0,0)^T, where\ T\ i\ transposed\ vector\ Y,$$

Equation 6: Hot vector for digit '3'

[1] each element of output layer vector will have values between 0 and 1, and the sum of all elements is 1.

In classification model, the aim is to minimize the loss between what the model believes the output distribution should be and what the original distribution really is. Cross entropy, as in Equation 7, is defined as:

$$H(y,p) = -\sum_i y_i \log(p_i), \text{ where}$$
$$p \text{ the predicted probability}, y \text{ the binary indicator of hotvector}, i \text{ number of classes}$$

Equation 7: Cross entropy loss function

This function is used as a loss function in neural networks which have SoftMax activation, though having the output a probability distribution. (Bishop 2006).

An example of multiclass classification problem resolved using deep learning recently (2019) with Fastai library over PyTorch is the one of Cats and Dogs. This special classification is called fine-grained classification, as some of the categories are very similar. (Howard 2019). The academic data set used, Oxford-IIITPet Dataset, depicts 12 cat breeds and 25 dogs, each of category or breeds has 200 images labeled data. (Parkhi, et al. 2012).

At that time, in 2012, the model used to differentiate between the 37 distinct categories was 59,21% accurate using a complex model specific to pet detection with separate models for head, body and image (Parkhi, et al. 2012).

Using a pre-trained model of a convolutional neural network, resnet34, Jeremy Howard obtained an accuracy of 94% (Howard 2019).

A pre-trained model uses transfer learning technique. In the transfer learning, the idea is to overcome the isolated learning paradigm and utilizing knowledge acquired for one task to solve related ones. Therefore, knowledge gained while recognizes for example dogs and cats can be used to recognize horse breads.

This model of residual convolutional neural network is further pursued in case study problem solving to respond the master thesis research questions.

Some of the classification's problems cannot be solved by binary classification or multiclass classification. The typical problem requiring a different approach is tagging. One example is satellite images classification problem, on dataset *planet-understanding-the-amazon-from-space* from Kaggle[2], a competition website for deep learning problem solving. The aim is to identify in each image from Planet Amazon dataset, the type of features contained from 17 defined classes like forest, agriculture, roads, water, weather and so on. So instead of identifying the highest probability from the output vector corresponding to one single class, which is

[2] https://www.kaggle.com/

typically done in multiclassification problem, we have to find and compare each probability of those classes with a determined threshold (Howard 2019). So that the model will assume the picture contains those features or classes. The problem of learning to predict classes that are not mutually exclusive is called multi-label classification (Zhang A. 2019).

3.2.1.2. Unsupervised learning

In supervised learning, data sets contain features and corresponding target values are given, so that we know exactly what a model should learn and is expected to do.
In unsupervised learning the model is provided with a large data set that is unlabelled. Moreover, there is no clear distinction between training dataset or test dataset. This model is expected for example to find some patterns or important features or to cluster data by different attributes (Goodfellow, Bengio and Courville 2016).
When dealing with high dimensional data, it is often useful to reduce its dimensionality by projecting data to lower dimensional subspace by that capturing only what is essential for our model to discover. Most common approach to dimensional reduction is called principal component analysis, which can be interpreted as an unsupervised version of linear regression algorithm that learns a representation of data (Goodfellow, Bengio and Courville 2016).

3.2.1.3. Reinforced Learning

Reinforced learning is a type of machine learning algorithm that enables an agent which interacts with an environment to take those suitable actions that are leading to a reward. Reinforced learning is using rewards to signal a positive behaviour and punishment to signal a negative one.
The agent learns from its own actions and experiences. Its goal would be to find a suitable model that would maximize the total cumulative reward. For that it is in constant interaction with the environment over a series of timesteps. The agent receives an observation from the environment at a timestep and must choose an action. In a repetitive loop, this action is sent to environment and this is responding back with a reward.

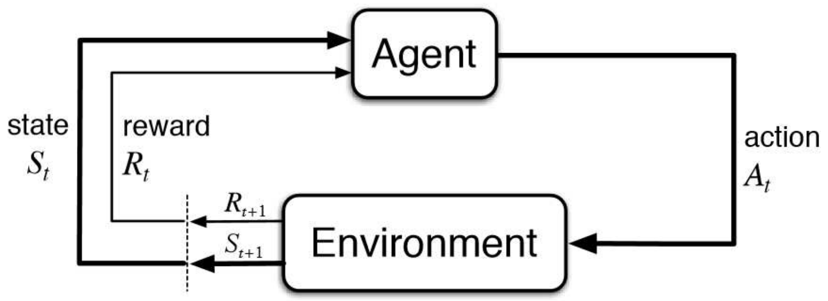


Figure 5: Action-reward feedback loop (Bhatt 2018)

This function of mapping from observation to actions is called a policy. Hence, the very purpose of the agent is to produce a policy which is optimal (Zhang A. 2019).
In order to build an optimal policy, the agent faces the dilemma of exploring new states, as shown in Figure 5, while maximizing its overall reward at the same time. This is called the exploration versus exploitation trade-off: to obtain a lot of rewards, a reinforced learning agent must prefer actions that it has tried in the past and proved effective but must discover such actions by trying actions that it has not selected before. The agent has to exploit what it has already experienced to obtain reward, but it also has to explore in order to make better action selection in the future. (Sutton und Barto 2018).
Reinforced learning applicability is within gaming industry and most important for our industrial purpose in robotics and industrial automation.

3.2.2. Optimization algorithms

3.2.2.1. Gradient descent

For the linear regression problem above, where number of sold ice-cream have a temperature dependency, the mean squared error or MSE was defined as a loss function and was anticipated that we would like to optimize it, in our effort to find the best fitting linear regression. Optimization refers to the task of minimizing the function. For that, there is a need to calculate the derivative or the gradient of our loss function. The derivative of the function $y = f(x)$, where both x, y are real numbers is denoted as $f'(x)$ or $\frac{dx}{dy}$. The derivative $f'(x)$ gives the slope of $f(x)$ at the point . Therefore, it specifies how to scale a small change in input to obtain a corresponding change in the output: thereof following approximation (considering a Taylor expansion [3]):

$$f(x + \epsilon) \approx f(x) + \epsilon f'(x).$$

For small enough $\epsilon > 0$, it is known that

$$f\left(x - \epsilon * sign(f'(x))\right) < f(x)$$

Equation 8: Gradient descent

Therefore, $f(x)$ can be minimize or reduced by moving x in small steps , with opposite sign of the derivative. This technique is called gradient descent *(here exemplified in one dimension)* as in Equation 8, where ϵ is the learning rate.

When $f'(x) = 0$, the derivate is also 0 so there is no indication to which direction to move towards minimum. This point is known as *stationary point* or *critical point*.

A *local minimum* and a *local* maximum are points where $f(x)$ is lower respective higher then all neighbouring points.

The point of absolute lowest value of $f(x)$ is the *global minimum*.

To minimize MSE, as defined in Equation 4, where

$$y'_i = mx_i + c, \text{ and } MSE = \frac{1}{n}\sum_i (y_i - y'_i)^2 ,$$

Equation 9: y' as linear function of slope m and intercept c

it is necessary to compute the derivative in m and c defined in Equation 9.

Then the gradient, as in Equation 10, is the vector derivative:

$$\nabla MSE_{(m,c)} = \left[\frac{\partial MSE}{\partial m}, \frac{\partial MSE}{\partial c}\right]^T.$$

Equation 10: Gradient of MSE in respect to slope m and intercept c

For a small learning rate ϵ then the new values approximated for m and c, as in Equation 11, are:

[3] Taylor polynomial: $T_n(x) = \sum_{i=1}^{n} \frac{f^{(i)}(a)}{i!} (x - a)^i$

$$m = m - \epsilon \frac{\partial MSE}{\partial m} \; and \; c = c - \epsilon \frac{\partial MSE}{\partial c};$$

where

$$\frac{\partial MSE}{\partial m} = \frac{-2}{n} \sum_{i=0}^{n} x_i (y_i - y'_i) \; and \; \frac{\partial MSE}{\partial c} = \frac{-2}{n} \sum_{i=0}^{n} (y_i - y'_i),$$

Equation 11: Calculation of new slope and intercept after one step of gradient descend

By repeating the algorithm, for a certain number of times, called *epochs*, the m and c will be optimized so that the distance to the input points is constantly minimized, like this the line $Y = mX + c$ will best fit the linear regression.

For function with multiple inputs, the concept of partial derivatives it is being used, generalizing the above MSE.

Hence, the gradient off f, where $f: R^n \to R$, is a vector $\nabla f(x)$, as in Equation 12. The element i of the gradient is the partial derivate of f with respect to x_i.

$$\nabla f(x) = \left[\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \ldots, \frac{\partial f(x)}{\partial x_n} \right]^T$$

Equation 12: Gradient off f in respect to all variable

(Goodfellow, Bengio and Courville 2016).

3.2.2.2. Stochastic Gradient Descent

In case of training of deep neural networks in solving classification problems, training on large datasets where computational power is bottleneck even on high performance GPU, a speed up would be of a substantial value.

The commonly used procedure to optimize $f: R^n \to R$ is to adjust $x_t \in R^n$, the parameter vector at a time step t, using gradient information $\nabla f_t(x_t)$ obtained on a small $t -$ th batch, called *minibatch* of the b *datapoints*. The Stochastic Gradient Descent, as in Equation 13, becomes an extension of gradient descent to stochastic optimization of f:

$$x_{t+1} = x_t - \epsilon_t \nabla f_t(x_t), \qquad where \; \epsilon_t \; is \; the \; learning \; rate.$$

Equation 13: Stochastic gradient descent

(Loshchilov and Hutter 2017).

A simplified way to look at the SGD is GD using minibatches. An epoch of training is however finished when all training data is seen by the network, and not only after applying GD to one minibatch.

3.2.3. Training a model

Before introducing the forward propagation and backword propagation, a recap of the core components of machine learning problems, which were introduced before is necessary. D*ata set* that we can learn from, *a model* to transform the data like a *neural network*, *a loss function* that quantifies how good is a model , and an algorithm to train our model, *SGD*, adjusting the model's parameters : the weights and biases in our neural network case.

3.2.3.1. Forward propagation

The forward propagation refers to the calculation required for the model to generate an output corresponding to the input.

In the same time, forward propagation refers to the storage of intermediate variables, including the output.

To review what it would be needed to store, lets reconsider the neural network with one hidden layer from Figure 4.Left, and let's assume the input vector is $x(3) = (x_1, x_2, x_3)^T$, the weights or parameters of the $W^{(1)} = (w_1, w_2, w_3)$, then $z = W^{(1)} x$, is the intermediate variable vector. The hidden layer vector $h(4) = (h_1, h_2, h_3, h_4)^T$ will be obtained by applying an activation function to z. In the same way the hidden variable *h* is an intermediate variable. Assuming the parameters or weights $W^{(2)}$ of the hidden layer, the output layer variable will be a vector of length 2, $o = W^{(2)} h$. With loss function l and for example the label *y* also a vector of length of 2 the loss of a single data input can be calculated, $L = l(o, y)$. For simplicity the *bias* term was ignored.

3.2.3.2. Backward propagation

Backpropagation algorithm refers to the method of calculating the gradient of neural network parameters. As in the forward propagation, back propagation calculates and stores the intermediate variable of an objective function related to each layer of the neural network and the gradient of the parameters (weights and biases) in the order of output layer to the input layer (Zhang A. 2019).

Back propagation is the algorithm for determining how a single training example will modify the neural network parameters, its weights and biases, for those changes to cause the most rapid decrease to the cost function, while a gradient descent algorithm will be doing so looking at all the training examples and averaging the changes.

3.3. Convolutional Neural Networks

As this master thesis focuses on approaching a special type of problems in automotive manufacturing, classification problems to be more specific, in this chapter a special attention will dedicate to the CNN model with focus on ResNet architecture.

3.3.1. Introduction

Convolutional neural networks are neural networks which are specialized in image processing. Since its introduction by (LeCun, Bottou, et al. 1998), LeNet-5 proved to be very efficient at classification task as the hand-written digit classification and also in face detection applications. Lately, convolutional neural networks are used in large scale in image and video classification, speech and natural language processing and other grid - like topology data (Gu, et al. 2018).

3.3.1.1. CNNs Various Architectures Evolutions

There are various architectures of CNN available: LeNet, AlexNet, VGGNet, GoogleLeNet, ResNet, ZFNet.

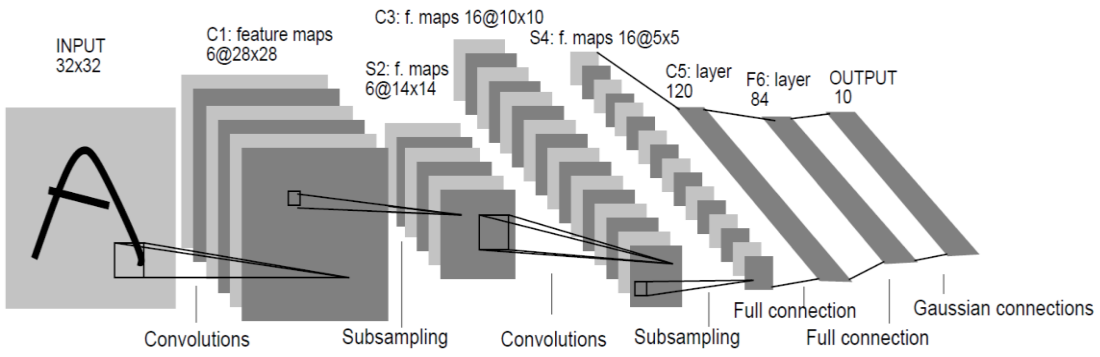


Figure 6: Original architecture of LeNet-5 (LeCun, Bottou, et al. 1998).

LeNet-5, as in Figure 6, improved by (LeCun, Bottou, et al. 1998), has 7 layers and can be trained with backpropagation algorithm. It could have recognized visual patterns directly from raw pixels with practically little pre-processing (Gu, et al. 2018).

AlexNet, proposed by Alex Krizhevsky (Krizhevsky et. al., 2012) shows a classical CNN architecture, very similar with LeNet-5 but with a deeper structure and proved to overcome the difficulties in training deep CNN's. It also made use of powerful GPU implementation and regularization like dropouts. It won the ImageNet[4] Large scale Visual Recognition Challenge, ILSVRC, competition in 2012, a data set with more than a 14 million images grouped in more than 21,000 categories[5].

ZFNet (Zeiler and Fergus 2014) outperformed AlexNet in ImageNet classification in 2013.They used a deconvolutional network model, trained on Nvidia GPU (Zeiler, Taylor and Fergus 2011).

VGGNet model belongs to University of Oxford and stands for Visual Geometry Group. Their model differs from ZFNet above by main differences in processing fixed image size of 224 x 224 RGB and having small receptive filter 3x3 on convolutional layers (Simonyan and Zisserman 2015).

VGGNet model was outperformed in 2014 ILSVRC competition by GoogLeNet architecture, from Google Inc. Group. Here the task was to classify images from 1000 categories, with provided 1,2 million images for training, 50, 000 for validation and 100,000 for testing. Even though their architecture model was using 12 times fewer parameters then in AlexNet winner of 2012, success was guaranteed by the synergy between CNN deep architecture and classical computer vision, regionlets for generic object detection (Szegedy, et al. 2015).

ResNet stands for residual neural networks. This model was used by Microsoft Research Team, MSRA, in the ILSVRC competition 2015. On the ImageNet classification task from 1000 categories described above, a ResNet convolutional neural network model achieved 3,57% error scoring best in ILSVRC 2015 contest. ResNet152, having 152 layers, is 20 times deeper then AlexNet and 8 times deeper then VGGNet but having lower complexity then VGGNet. (Gu, et al. 2018).

The core idea of ResNet is the introduction of identity shortcut connection, that skips one or more layers as shown in the below Figure 7:

[4] http://www.image-net.org/challenges/LSVRC
[5] http://www.image-net.org/download-imageurls

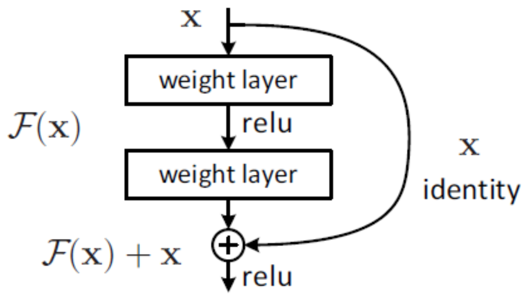

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 7: Residual building block (He, et al. 2015)

They hypothesize that it is easier to optimize the residual mapping than optimize the original unreferenced mapping (He, et al. 2015).

The below Figure 8 contrasts the networks architecture for ImageNet: a VGG- 19, a 34 Layer plain network and a 34 Layer ResNet.

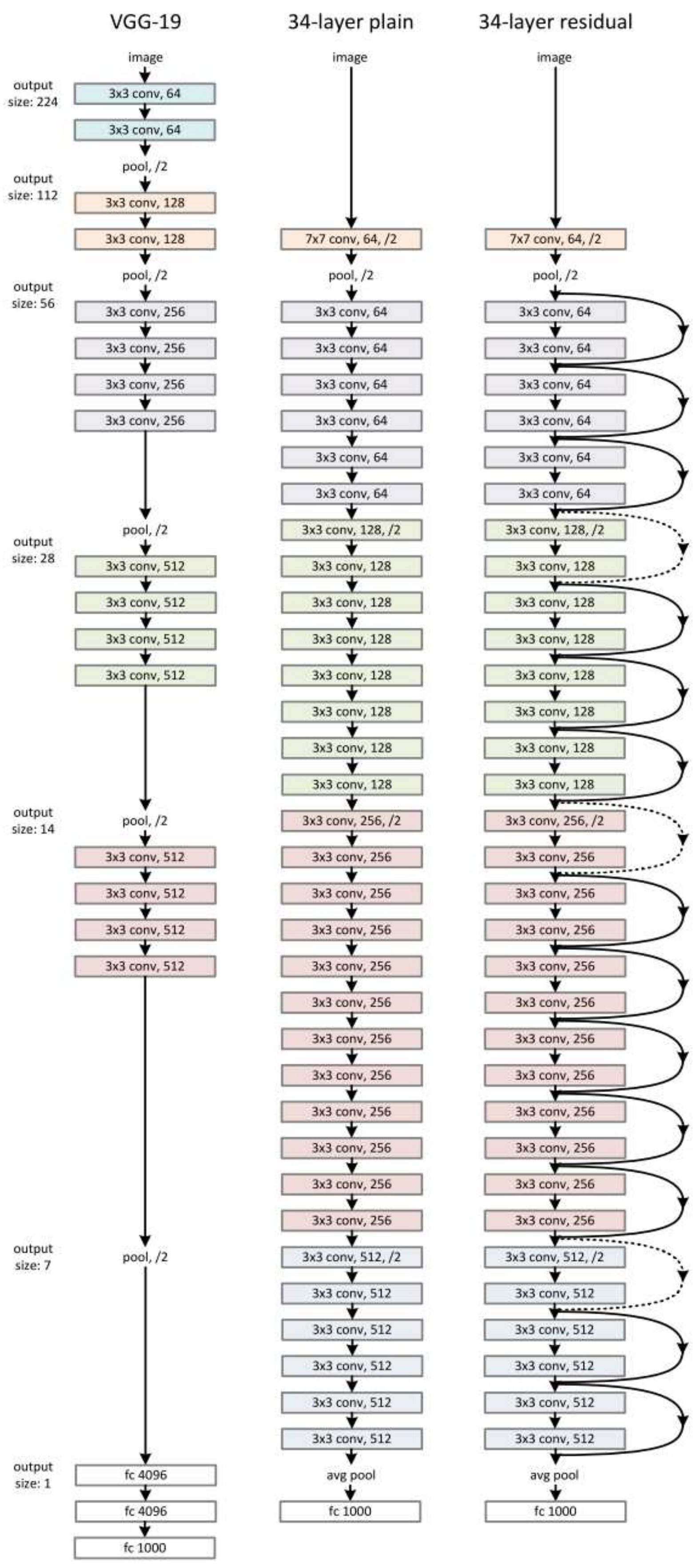


Figure 8: Example network architecture for ImageNet (He, et al. 2015)

3.3.2. CNN's Architecture

In their basic structure, the CNN's are taking an image input, 1D, 2D or 3D, and pass it to the feature learning part and then trough classification part. In the feature learning part, as can be seen in Figure 6, the image is broken down to features, by repetitive convolutions and pooling mechanisms, then flattened to a vector of features giving a prediction of the classification in the classification part.

Before introducing convolution, let us have a quick recap of a matrix multiplication, shown in Equation 14, to have in mind the process used by the traditional neural networks, and to contrast with later introduced convolution concept Figure 10.

$$\begin{pmatrix} 35 & 19 & 25 \\ 11 & 19 & 12 \\ 4 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} =$$

$$\begin{pmatrix} 35\cdot(-1)+19\cdot0+25\cdot1 & 35\cdot(-2)+19\cdot0+25\cdot2 & 35\cdot(-1)+19\cdot0+25\cdot1 \\ 11\cdot(-1)+19\cdot0+12\cdot1 & 11\cdot(-2)+19\cdot0+12\cdot2 & 11\cdot(-1)+19\cdot0+12\cdot1 \\ 4\cdot(-1)+1\cdot0+0\cdot1 & 4\cdot(-2)+1\cdot0+0\cdot2 & 4\cdot(-1)+1\cdot0+0\cdot1 \end{pmatrix} =$$

$$\begin{pmatrix} 9 & -1 & 9 \\ 20 & 21 & 20 \\ -4 & -8 & -4 \end{pmatrix}$$

Equation 14: 3x3 Matrix multiplication. (45 computations)

3.3.2.1. Convolution, stride and padding

Convolutional neural networks or CNN's are neural networks using convolution in place of general matrix multiplication in at least one of their layers (Goodfellow, Bengio and Courville 2016).

However, a discrete convolution ca be written also as a matrix multiplication. The reason it's not being done in practical, is that operation is too slow in comparison with convolution calculation, has weights untrainable of value 0, and weight which are equal and have to stay equal trough training process. This are called *shared weights*. (Goodfellow, Bengio and Courville 2016). (Kleinsmith 2017).

Convolution is a mathematical operation on two integral functions of real value argument denoted with an Asterix, '$*$':

having two functions,

$f(x), g(x): R \to R,$ a function $z(x): R \to R\,, where\ x, a \in R,\ then$

$$z(x) = (f * g)(x) = \int_{-\infty}^{\infty} f(y)g(x-a)da$$

Equation 15: Convolution off f trough g

is called the convolution of f trough g, Equation 15.

In convolutional network terminology, f, which is a multidimensional array of data, is referred as *input*, g, which is a multidimensional array of parameters, is referred as a *kernel* or *filter*, the right term in the matrix multiplication from Equation 14, and the output z, which is also a multidimensional , as a *feature map or a channel*.

If the functions $f\ and\ g$ application is restricted to domain to Z, and $x, a \in Z$ then we can write the discrete convolution, Equation 16, as:

$$z(x) = (f * g)(x) = \sum_{a=-\infty}^{\infty} f(a)g(x-a)$$

Equation 16: Discrete convolution off f trough g

The filter or kernel is used in machine learning as *feature extraction.* (Goodfellow, Bengio and Courville 2016). Filters are finding features, trough the convolutional layers, from simple groups of pixels oriented in a diagonal or vertical or gradient in the first layer, to curves, or circles, or top left corners or bottom left corners as they go deeper in layers can combine things and will find patterns that are repeating as geometric patterns or even text. In very deep layers they will combine the already found features and put them together to find first eyes, then faces, then legs and in the end layers animals or humans or complex object like bicycles, cars. Here an example of two filters, which can perform edge detection. This are called *sobel* filter:

$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$ for horizontal edge detection, and

$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$ for vertical edge detection.

Equation 17: Sobel kernels examples for edge detection

In machine learning the filter are predefined, design by programmers, as shown in Figure 2.They have a purpose of discovering features known to human. (Equation 17).

The beauty of CNNs, and deep learning in general, is that these filters are not defined in their architecture, but they are learned during the training process. Like that CNNs can detect abstract objects already in the first layers, that predefined human filters might not be able to find out. (Figure 2).

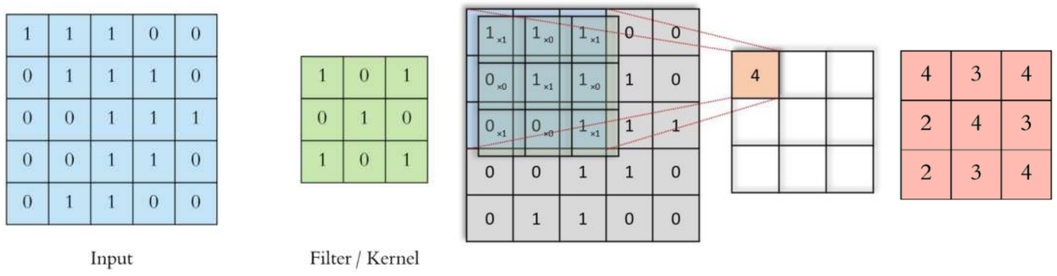
Let us look now to a practical example in which we have as an input a matrix of 5x5 pixels, and a 3x3 filter or kernel like in Figure 9. The output after convolution will be 3x3 matrix. That is because, each of 3x3 block in the input 5x5 input is then multiplied elementwise with the kernel then all nine values are being added up to form one output value. This is a convolution[6], as shown below in Equation 18:

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} =$$

$$= (1*1+1*0+1*1)+(0*0+1*1+1*0)+(0*1+0*0+1*1) = 4$$

Equation 18: A convolution with a 3x3 kernel

Then the kernel will step with one element to the right, to calculate the output 3, corresponding to row 1 and column 2, then make another step to right to calculate the output 3, corresponding to row 1 and column 3, and so on continuing this movement by step down, left and right till outputs the last value. (Figure 9. Right).



1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

1	0	1
0	1	0
1	0	1

4	3	4
2	4	3
2	3	4

Figure 9: Input 5x5 matrix with 3x3 kernel. After convolution the output or convoluted feature or feature map is the block Right of size 3x3. (Balsys 2019)

To reduce the output rapidly through layers, there are two processes steps which can be applied, both specific for CNNs. This are stride and padding.

In general, kernels have bigger dimensions then 1x1. By applying many successive convolutions, the size of the output it will decrease. To prevent this, padding can be applied, which consequently will increase the input size.

[6]A convolution with a 3x3 kernel consists of 17 computations.

In case a reduction of input resolution is desired or even a higher reduction of the output is requested than by applying kernels in convolutions, stride can be applied.
The step with which the kernel shifts over the input matrix is called Stride and is denoted by an S. If we move to one pixel at a time, then S is 1. If we move the filter to 2 pixels at a time, then we have a Stride 2. Hence, we can say we are sliding by two pixels usually on the right side of input.
As seen in Figure 8, the usual kernel dimension used in CNN is 3x3, but can also be of 7x7 as in the first layer.
Here there was considered an input with a single channel, of form 5x5x1. In case of multichannel images (eq. RGB has 3 channels) 5x5x3, then the filter or kernel will have the same depth as the that of input image, 3x3x3, as in Figure 10. A convolution will mean that we will add up the 3x3x3 = 27 values and the output will still be a one depth channel feature:

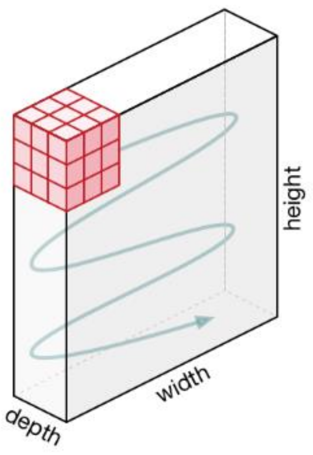


Figure 10: A 3x3x3 Kernel movement trough an input RGB image. (Saha 2018)

Each convolutional layer in CNN has usually more than one filter. As seen in Figure 9, a convolution filter of size 3x3, with a stride 1 will reduce the input size from 5x5 to an output value of 3x3.
Looking into the architecture of LeNet-5, as shown in Figure 6, on the first convolutional layer, it has 6 filters of 5x5x1 each and a stride of 1. Therefore, inputting a 32x32x1 image, with unmodified size, the convolutional layer one will output an image of 28x28x6. To review, the total weights and biases, also described as trainable parameters are: (LeCun, Bottou, et al. 1998).

$$Weights + Bias = 5 * 5 * 1 * 6 + 6 = 156,$$

and in total 122304 connections:

$$28 * 28 * 156 = 122304.$$

Filters and convolutions task are to extract the features from an input. After a convolution, as in Figure 9, the output dimension is reduced compared to input image. To prevent this dimension reduction, a process named *padding* could be applied. To keep the output size of the same dimension 5x5 as the input size, see Figure 9, when convolved with a 3x3 filter, we would need to add to input an outer layer so that now the input size would be of dimension 6x6. In this case we denominate this as being a padding of one, $= 1$.

3.3.2.2. Pooling

Some layers in the CNNs are not convolutional layers. As we go deeper in the network the connections and parameters number drastically increase. To condense the mapped features often a pooling layer is inserted immediately after a convolutional layer. (Nielsen, 2016).
If the first reason of pooling layer is to reduce the amount of computations, by down sampling the output after activation, with ReLU for ResNet, the second would be to select and transmit to next convolution layer only the augmented data. However, there are critical voices among scientists against pooling since its usage tends to lose positional information.
There are many types of pooling but most widely used are L2 pooling, max pooling, average pooling.
The example below, Figure 11, shows average pooling with 2x2 filters and a stride 2:

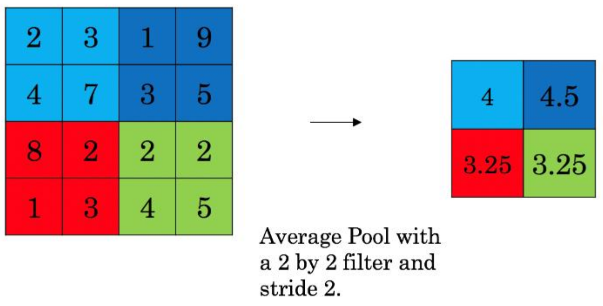


Figure 11: Average pooling 2x2 filter and stride 2. (Balsys 2019)

Hence, having now all these notions introduced, we can summarize the formula for calculating the output size of a convolutional layer, Equation 19, where O is the output size, K is the kernel size, S is the stride and P denotes pooling, where Input has equal height and length, same for kernel:

$$O = \frac{(W - K + 2P)}{S} + 1$$

Equation 19: Formula to calculate the output size of a convolutional layer

3.3.2.3. Fully connected layer

To summarize from a simple perspective the ResNet34 model used in ImageNet: there was a constant enrichment of complex features trough convolutional layers with alternative strides, each convolution followed by activation with ReLU, and occasionally applied pooling. The output of the convolutional part of the model is now a multidimensional array or tensor of 11x11x512 in case of ResNet34. The network ends with a global average pooling layer and a 1000-way fully-connected layer with SoftMax function, that is because of the target vector in ImageNet is 1000, as the problem is to classify images from 1000 categories (Gu, et al. 2018). Then we need the cross-entropy cost function to assess how far we are from the hot-vector as described in the classification problem chapter. For example in Fastai library, when we will create a model of ResNet34, the weight matrix having 1000 columns will to be split in 2 weight matrixes with a ReLU activation in between, the second one having dynamically encoded the data categories according to the data bunch object as column number with raw number corresponding to column number of previous layer (Howard 2019).

3.4. Summary

During this chapter the basic concept and methods used in neural network ware introduced. It was the intention to deep dive from basic perceptron neuron, trough machine learning problem solving approach, with neural network model, then finally focusing on deep learning with convolutional neural networks. A detailed analyze of ResNet architecture was covered with the intention to set the fundaments for the qualitative analyze of the raised problem in the case study. Finally, all mechanic needed to understand functionality and usage of this powerful model was explained to have the understanding, the tools and means of solving a classification problem. Other specific connecting concepts will be covered during problem solving in the next chapter.

Chapter 4. Case study: multiclass classification of X-ray acquired images with ResNet50

In this chapter the focus will be to solve a very delicate problem for manufacturing automotive industry based on the theory and concepts introduced in previous chapter. The optical inspection stations, or better said processes, in all automotive electronic manufacturing companies are quality walls towards customer satisfaction and a quick feedback on pre-processes reliability and quality. You probably ask yourself why these stations are needed after all. Probably you ask yourself how reliable these pre-processes are or if they are reliable, capable enough. In the automotive electronic, the defects per million opportunity targets are very challenging, and particularly more challenging when the product fulfils and increase safety functionality. The DPMO targets, combined with first pass yield targets and other key performance indicators like retouch and repair acceptance criteria, are creating a challenging environment for industrial engineers and their process development. The development department is the first contributor to the quality of manufactured electronic control units. They are constantly improving design using six sigma approach with direct impact in process capability where minimum process capability, cp / cpk values, are set to a minimum of 1,67. A crucial role is the quality of the supplied parts and subassemblies which need to conform tough quality norms. Still, there are a group of processes which are still far for perfect reliability and due design limitation unavoidable in a real business environment considering economics factors and restrictions by design, weight, reliability. Part of this process's groups are the soldering processes. There are certain technologies available for soldering like reflow soldering, iron soldering, selective mini wave soldering, wave soldering or even manual soldering where no other design solution could be affordable. These processes are key processes in electronics printed circuit board manufacturing. For each control unit the output of a soldering process is controlled against industry standard IPC-A-610 Revision G dated 2017 where automotive specific acceptance conditions are defined. On top of this standard there are specific customer requirements for example in terms of void acceptance within solder joints, meant to highlight that product is expected to have a certain reliability over life time.

The post inspection of soldering processes being optical inspection or X-ray systems are holding the key in sorting out the failures generated by soldering and are also important information on the reliability, adjustments and process improvements to be taken as a close loop system. Therefore, for each electronic control unit we acquire multiple pictures sometimes preselected by optical system or just passed all to an operator deciding whether the solder joint is within de limits or not. In the electronic plants manufacturing, the acquisition rates for post reflow AOI inspection are in the range of 50 to 100 pictures per second within one single equipment, as observed during visits performed in electronic manufacturing productions. The situation is not different in the back-end area for high resolution and high-speed camera systems but much slowly for X-ray systems where we are in the range of 5-10 picture per second. The pictures acquired by X-ray are grey level (black and white) pictures and are the most acquisition time demanding, especially if the solder joints are disposed around the PCB or assembly and not focused within coverage area.

There are no libraries or catalogues for this type of X-ray system so that a preselection or an assisted system could ease up classification on operator side. Therefore, the focus was set on system where all acquired pictures are sent for evaluation to operator. For each unit in my case study, there are 20 pictures taken to be evaluated according to pass-fail acceptance criteria. If there is one nonconformity, the hole product will be evaluated as failing the step and will be quarantined, in order for it to be retouched or scrapped.

The general purpose is to show that this visual inspection classification problem can fully be entrusted to a trained convolutional neural network in productive environment.

The primarily goal is to secure the 100 % success of quality wall defined by 100% reliable evaluation of pass-fail decision, meaning a trustful classification of the acquired images.

The immediate economic benefit is easily to be deducted, as currently this station is 24 hours and 7 days per week humanly operated station. The loading and therefore the distributive attention of the operator is huge since the line takt time is 20 sec / unit inspection time having with one X-Ray equipment. The possibility of line redesign and re-layouting including different approach of automation would be an additional benefit, which should be evaluated case by case.

The cost of development and implementation it is not evaluated here as depend to much on every company capability and strategy equal make or buy decision.

4.1. Problem description

The Optical inspection station consist of an X-Ray equipment, in my application Viscom X8068. This is a manual operated station, that means charged and discharged by an operator. It consists of a round table with 4 working positions, which is completed loaded before closed and exposed to X-Ray inspection. The system acquires 80 pictures corresponding to the 4 units and send them to a terminal where operator makes a classification of each based on predefined classes. As each unit is uniquely identified by a barcode, so that we have the associated pictures classified on part number. A classified failure is then automatically recognized by the traceability system which forces the unit to enter a diagnostic loop so it cannot be further processed in the routing: will be rejected by the next station.

All possible errors are maintained in the failure catalogs, which are used for training purposes. In the catalog, all failures, marked in red and by NOK, are paired to a good evaluation, marked in green and **OK,** and refer to the same area as a negative and positive example.

In the Figure 12, it can be observed in the left side a good example on all 7 solder joints after soldering a metallic foil with pins terminal, followed by the same picture with an example where solder joints are missing, caused due bad soldering process.

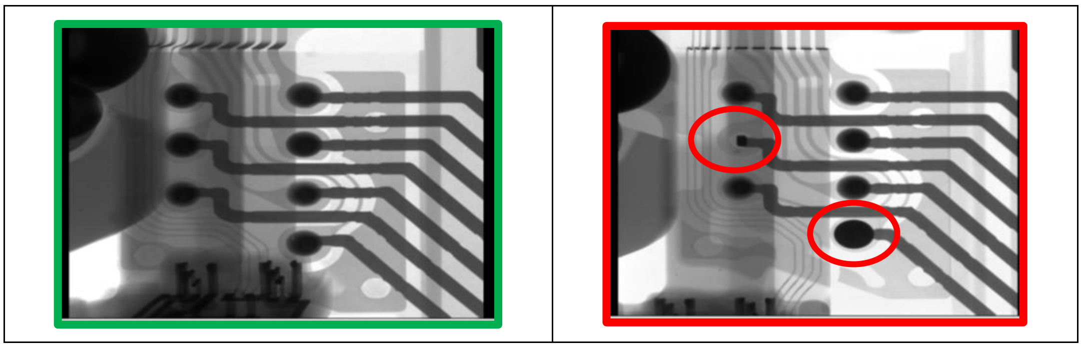


Figure 12: Foil Pin soldering: Left OK in green –Correct shape of soldering, without solder balls or excess material, no short, wires not deteriorated; Right NOK in red, missing soldering shown by the red circles

In the Figure 13, a second example contrasting the good and bad evaluation is shown, this time on a different area.

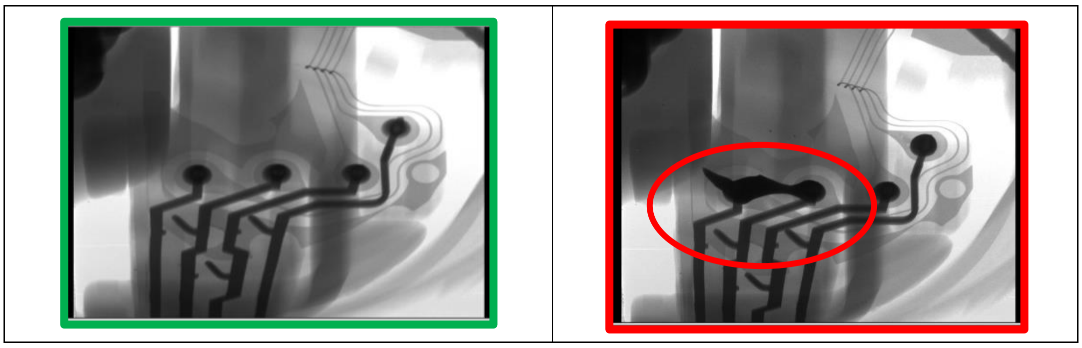


Figure 13: Foil Pin soldering: Left OK in green –Correct shape of soldering, without solder balls or excess material, no short, wires not deteriorated; Right NOK in red, short circuit trough solder excess shown by the red circle

In the Figure 14, an example on metal metal soldering can be observed and in Figure 15, another good and bad example are contrasted, this time for foil-foil soldering.

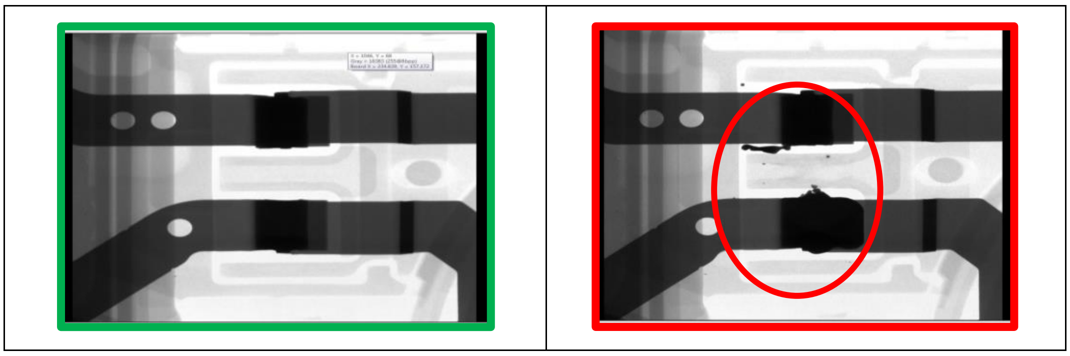


Figure 14: Metal-metal soldering: Left OK in green –Correct shape of soldering, without solder balls or excess material, no short; Right NOK in red, solder excess shown by the red circle

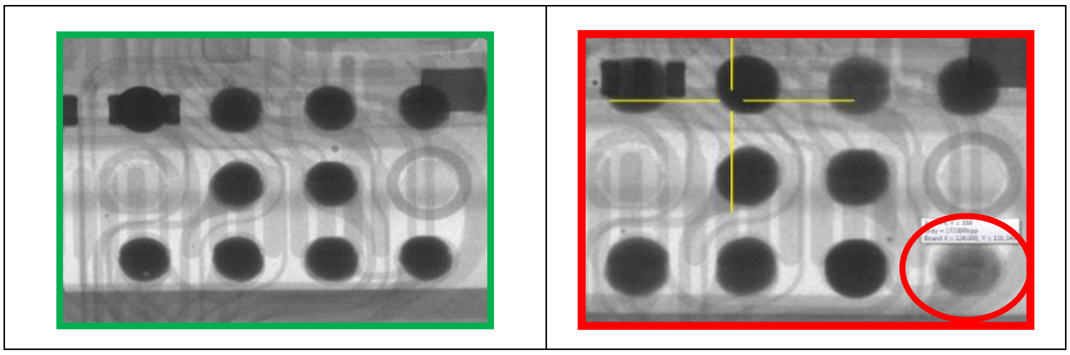


Figure 15: Foil-foil soldering: Left OK in green –Correct shape of soldering, without solder balls or excess material, no short; Right NOK in red, missing soldering shown by the red circle

In reality, there is a more complex situation which will be described next: there are 3 possible defects for a solder joint, with the 4th one denoted '*pseudo_error*' which means 'a good classified solder joint'[7]. The 3 defects would be:

(1) *'short-circuit'* or '*solder bridge'* as shown in Figure 13,
(2) *'no_solder_or_bad_solder'* with different defects like voids, missing soldering, defect with adherence of solder on pads or other defects
(3) '*limit_error'* which is basically a filling characteristic for a trough hole soldering where, filling < 75% is a failure, as defined in IPC-610-A[8] class 3. Hence, the 'errors' classes would be: ('limit_error', 'pseudo_error', 'no_solder_or_bad_solder', 'solder bridge').

These 4 possibilities are to be combined with 10 pin types: 'NK1', 'PLCD', 'nMot', 'THT', 'sG6R', 'SGVC', 'NK2', 'Ball', 'MM', 'Sens_press'. Thus, we should have 40 target classes, but we train

[7] Acceptance criteria's according to IPC-610-A class 3
[8] http://www.ipc.org

the neural network only on first 18, ordered by their count, as defined in Table 1. Because we strip the pin type and we keep just the error, thus reducing to 4 target classes, we can *'guess'* the correct answer for any of the theoretical 40 classes defined from combining the 4 errors and the 10 pin types. If the dataset representation would have had enough images for each class, we could have also defined 40 classes.

There are 3 consecutive soldering processes, followed by the X-ray inspection, to solder sensors to assembly, with two selective soldering processes and a hot air soldering process. A fourth process, an iron soldering process, will be considered as well in the inspection as it is part of a sub-assembly supplied from a different production line, also to be soldered on the final product.

The selective soldering process is used for foil – pin soldering and metal-metal soldering. The hot air soldering is used to solder two foils together. The iron soldering process is joining a metallic wire terminal to a metallic terminal, with a similar output as in Foil-Pin soldering from Figure 16: Center.

In general, on this assembly we have to inspect by X-ray minimum 8 areas with a different combination of soldering processes and specific component. These are encoded in the classes defined in Table 1 and combined with specific defects for a solder joint. The soldering experts are responsible with labeling and maintenance of defect types. In general, the classes are encoded on the soldered component name and combined with operator classification based on possible defect for a solder joint that could apply to this classes and are part of the failure catalogues. This image sorting and storage is performed automatically by the X-Ray software after the operator classification. Therefore, a correct classification is mandatory to have a qualitative dataset. For each product there are 20 images acquired by the X-ray. The product is considered as pass after all X-ray displayed images are classsified as ' pass inspection '.

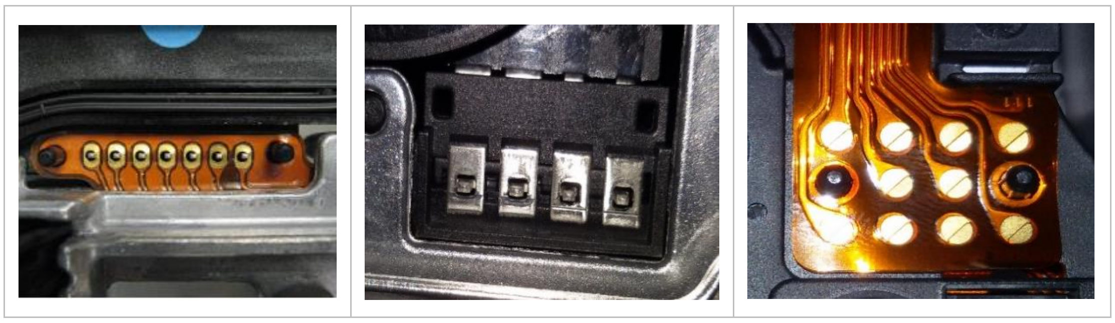


Figure 16: Types of soldering: Left – foil&pin; Center – metal&metal; Right - foil&foil

All this acquired images by X-Ray, pass and fail, are kept tracked off, for each ECU produced, for minimum 10 years as required by OEM's and or automotive norms. Picture classification is performed 100% by operators, according to failure catalogues, containing pass / fail and border line acceptance criteria examples for each inspected position and possible solder defect. To be more exact, operator is able to classify single failures or group failures, as this can offer information on process or components reliability. If he depicts an error, he will be pop upped with a defect list with possible error types, which are corresponding with the classes from Table 1. This products group have variants with different design and population but using the above described technics.

4.2. Setting Environment

To approach this classification problem, following environment has been used:

(1) Jupyter Notebook[9].
    - A web application for data science incorporation documents creation and sharing, visualization like graphics and text, and most important a programming environment which is incorporating by default the IPython kernel and is open source
(2) The deep learning libraries used are PyTorch[10] and Fastai[11] build on PyTorch, both open sources.
(3) Cloud computing based on NVIDIA GPU
(4) Cloud storage services

4.3. Dataset and data modelling

As this product is already in production since couple of years, there are relevant data available to start with data collection. However, the classification should match the defects in the latest failure catalogue. The dynamic changes of supply base influencing process parameters do not constitute an issue for the classification problem, as long as design remains unchanged, but they are to influence the appearance of solder joints and decision making, due they influence product quality.

Data collection is done for all classes to be predicted, as in the production environment. Those pictures are then stored correspondingly to their classes in S3 Amazon cloud.

There were initially 18 different classes identified which are relevant for our learning process : 'Ball-pseudo-error', 'MM-pseudo-error', 'NK1-no-solder-or-bad-solder', 'NK1-pseudo-error',

[9] https://jupyter.org/
[10] https://pytorch.org/
[11] https://docs.fast.ai/

'NK2-no-solder-or-bad-solder', 'NK2-pseudo-error', 'PLCD-no-solder-or-bad-solder', 'PLCD-pseudo-error', 'SGVC-no-solder-or-bad-solder', 'SGVC-pseudo-error', 'THT-no-solder-or-bad-solder', 'THT-pseudo-error', 'THT-solderbridge', 'limit-error', 'nMot-no-solder-or-bad-solder', 'nMot-pseudo-error', 'sG6R-no-solder-or-bad-solder', 'sG6R-pseudo-error'.

4.3.1. Image processing

Sometimes the images acquired are not suitable for learning in their raw form. As in my complex case, because of the high variation in geometry of components to be soldered and different soldering processes applied, combined with distance and deepness variation to X-ray objective, the images must be transformed. The process of image transformation is called augmentation.

4.3.1.1. Image Augmentation

The image augmentation technology expands the scale of training datasets by randomly changing the training images to create similar but different training examples. Hence, augmentation is used in order to reduce a model dependency on properties like brightness, colour, positioning.

Most common method used in the image augmentation is flipping. Flipping images is not changing the size of the image nor the category of the object. In deep learning
both horizontal and vertical flipping are used.

Resizing an image and cropping are also widely used. Cropping is mostly used in order to reduce the dependency of the model on the object position. Basically, we can use it to compensate the pooling layers effect, which we have seen that it reduces the model sensitivity of the convolutional layer to target location. Resizing is useful for example in upscaling an image which was cropped to maintain initial size (Zhang A. 2019).

The changing color methods is left out intentionally since here the gray scale pictures are processed. From the method of overlaying multiple augmentation methods only a flipping combination vertical and horizontal was used.

No matter how we use augmentation, we need to assure that all objects are having the same shape and size when passed to a GPU, since it needs to apply the same set of instructions to all in order to be fast in processing them.

For data set of the case study, the procedure we used is the image reduction to 224 x 224, which is the typical size used for classification problems. More, in almost all cases we have square images and very less cases are using rectangular images.

The results where then 50% flipped on vertical axis, chosen in an arbitrary way, and then we have flipped each result plus initial 50% of the images on horizontal axis.

4.3.1.2. Data Normalization

Before creating a data object to be passed to the model the data should also be normalized. To train deep neural network the most effective algorithm is stochastic gradient descent, which was introduced in Chapter 3. , making use of working with minibatches rather the hole batch, speeding up the training.
Normalization makes use of a concept of internal covariate shift, which is a change in input distribution to a learning system. Making use of ReLU activation function and small learning rates, the distribution of inputs, in this case nonlinear inputs, will stay stable and the effect would be an accelerated training. Normalization on batches in deep networks practically reduces the variances on layer inputs. Again, this focuses on reducing significant the training steps, between 7 to 14 times, maintaining the same model accuracy (Ioffe and Szegedy 2015).

4.3.2. Dataset classes; training and validation set

In the final form, after the augmentation and normalization, the dataset passed to the learner grouped on classes and with corresponding number of images is defined as follows (Table 1):

Classes	Number of images per class
THT-solderbridge	2500
NK2-pseudo-error	2500
NK1-no-solder-or-bad-solder	2500
Ball-pseudo-error	2500
PLCD-pseudo-error	2500
THT-no-solder-or-bad-solder	2500
NK1-pseudo-error	2500
MM-pseudo-error	2500
SGVC-pseudo-error	2500
THT-pseudo-error	2500
sG6R-pseudo-error	2500
SGVC-no-solder-or-bad-solder	1750
PLCD-no-solder-or-bad-solder	1750
nMot-pseudo-error	1250
limit-error	1250
sG6R-no-solder-or-bad-solder	1250
NK2-no-solder-or-bad-solder	1250
nMot-no-solder-or-bad-solder	1250

Table 1: Classes and corresponding number of images

The list with the split between training and validation data is shown below in Table 2. The split is arbitrarily performed, Training set containing 80% of the images from the Dataset, and the Validation set the remaining 20% of the images from the Dataset:

Dataset	Size in number of images
Training	29800
Validation	7450
Dataset total	37250

Table 2: Split of Dataset in Training set and Validation set

4.4. Model training

As there I could not find supporting evidence on related to different models performance on similar datasets, there were two architectures considered in the first attempt: ResNet34 of PyTorch (with Fastai as high-level framework) and MobileNet of TensorFlow (Keras).
Training a complete model of ResNet34 and a mobilenet_v2_035_160 didn't reveal significant difference in performance even though the Dataset approach was different.
There was a resource base decision to continue with ResNet architecture and perform a small comparison between a 34-layer ResNet versus a 50-layer ResNet.

4.4.1. Quantitative analyse: comparing ResNet34 and ResNet50

To start with, the two ResNet34 and ResNet50 were considered, pretrained models on ImageNet dataset. Hence, there will not be arbitrary initial weights and biases. This was presented in chapter Classification problems when the transfer learning was touched. The main advantages were highlighted as being a very efficient method to shorten training time, especially visible on large datasets.

4.4.1.1. Training the residual models

An *ImageDataBunch* object $data$ based on dataset with characteristic and contend described in Table 1 and Table2 was passed to both learners. We have trained last layers of the pre-trained models of both ResNet34 and ResNet50 for 4 epochs and a learning rate of 3e-3 or 0,003, as in Figure 17.

$$\begin{aligned}
&learn34\ = cnn\_learner(data, models.resnet34, metrics \\
&\qquad\qquad = [accuracy, error\_rate]) \\
&learn34.fit\_one\_cycle(4, max\_lr = 3e-3) \\
&learn50\ = cnn\_learner(data, models.resnet50, metrics \\
&\qquad\qquad = [accuracy, error\_rate]) \\
&learn50.fit\_one\_cycle(4, max\_lr = 3e-3)
\end{aligned}$$

Figure 17: Learner for ResNet34 and ResNet50; trained for 4 epochs at leaning rate 0,003

The results are listed in the below tables, Table 3 for ResNet34 and Table 4 for ResNet50:

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.811925	0.609456	0.746443	0.253557	03:00
1	0.604283	0.451854	0.817987	0.182013	02:57
2	0.439521	0.344463	0.865101	0.134899	02:57
3	0.369390	0.318131	0.875302	0.124698	02:57

Table 3: ResNet34 classification results @default learning rate 3e-3

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.743820	0.711489	0.723893	0.276107	06:02
1	0.527735	0.422609	0.835436	0.164564	05:52
2	0.372991	0.314853	0.880671	0.119329	05:52
3	0.288309	0.260871	0.895973	0.104027	05:52

Table 4: ResNet50 classification results @default learning rate 3e-3

It was expected that ResNet50 will be more accurate than ResNet34, as they performed in a similar way with the ImageNet dataset. (He, K., et. al., 2015).

The difference of 2% in error rate is quite significant, despite the higher training time required, which is practically double for ResNet50 in comparison with training time of ResNet34. The $accuracy$ here is to be interpreted as the number of correct classified image from validation set divided by total number of images from the validation set and the error_rate is 1- $accuracy$.

The $train\_loss$ and $valid\_loss$ are representing the cross-entropy loss on training respective on validation datasets.

However, this result can be improved. To improve accuracy, we will need to train the hole model, not only the last layers, but with different learning rates. To allow model to be trained on every layer, the hole layers model should be open for training, by calling the $unfreeze()$ function. As a prerequisite, to find out on which learning rates a complete training should occur, an additional function call, learning rate finder, $lr\_find()$ will help with visualizing and selecting the appropriate rates. Ideal is to search to pass to model a learning rate interval which keeps a decreasing trend of the loss as in the graph from Figure 18 (Howard 2019).

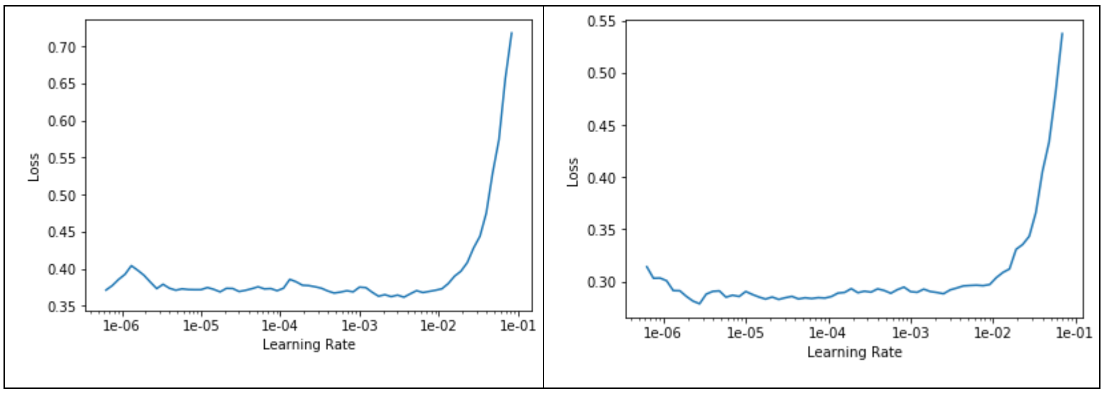


Figure 18: Learning rate finding: Left - ResNet34; Right - ResNet50

4.4.1.2. Train the hole models

With models unfreeze, a retrain will be restarted for 4 epochs, on the same dataset. Figure 21. In case of learning rate, there was a new approach applied, by using a learning rate interval and instead of single value: (1e-5, 1e-3) for ResNet34 and (1e-6, 1e-3) for ResNet50. Passing the function $slice(lr\_min, lr\_max)$, will have the effect that the very first layers will be trained with the lowest interval value and the very last ones with the highest value of the interval, all other layer with a distribution in between.

$$learn34.unfreeze()$$
$$learn50.unfreeze()$$
$$learn34.fit\_one\_cycle(4, \max\_lr = slice(1e-5,1e-3))$$
$$learn50.fit\_one\_cycle(4, \max\_lr = slice(1e-6,1e-3))$$

Figure 19: Unfreeze and retrain ResNet34 and ResNet50 for 4 epochs at leaning rate interval

The retraining results are visible in Table 5 for ResNet34 and Table 6 for ResNet50.

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.426164	0.391862	0.848322	0.151678	03:52
1	0.375032	0.312802	0.878792	0.121208	03:52
2	0.260104	0.228288	0.909933	0.090067	03:52
3	0.198114	0.212135	0.917718	0.082282	03:52

Table 5: ResNet34 classification results @ LR (1e-5,1e-3)

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.383733	0.324190	0.871678	0.128322	07:36
1	0.335155	0.314032	0.871946	0.128054	07:35
2	0.240270	0.204141	0.922819	0.077181	07:35
3	0.178364	0.185198	0.930201	0.069799	07:35

Table 6: ResNet50 classification results @ LR (1e-6, 1e-3)

4.4.2. Summary of quantitative analyse

By training the complete models at determined learning rates, a significant reduction of the failure rate could be seen: ResNet50 *error_rate* decrease from 10,4 % (Table 4) to 6,9 % (Table 6) and ResNet34 *error_rate* decreased from 12,4% (Table 3) to 8,2% (Table 5). However, training time per epoch increased with 29% if we compare ResNet50 pre-trained against ResNet50 fully-trained model (Table 4, Table 6) and with 31% if we compare ResNet34 pre-trained against ResNet34 fully-trained model (Table 3, Table 5).

In the same time, train_loss improved by 11% and valid_loss improved by 7,5% for ResNet50 fully trained against the ResNet50 pre-trained (Table 4, Table 6). The same positive trend is visible for fully trained ResNet34 if we compare it to pre-trained ResNet34 model: train_loss improved by whooping 17,1% and valid_loss improved by 10,6% (Table 3, Table 5).

As the dataset is of a negligible size for the CPU and GPU considered in my case, and as well the proven superior performance of ResNet50 fully trained model, it was easily to conclude, that in this small quantitative study this model, ResNet50, is the right choice for further use in production environment.

4.5. ResNet50 model evaluation

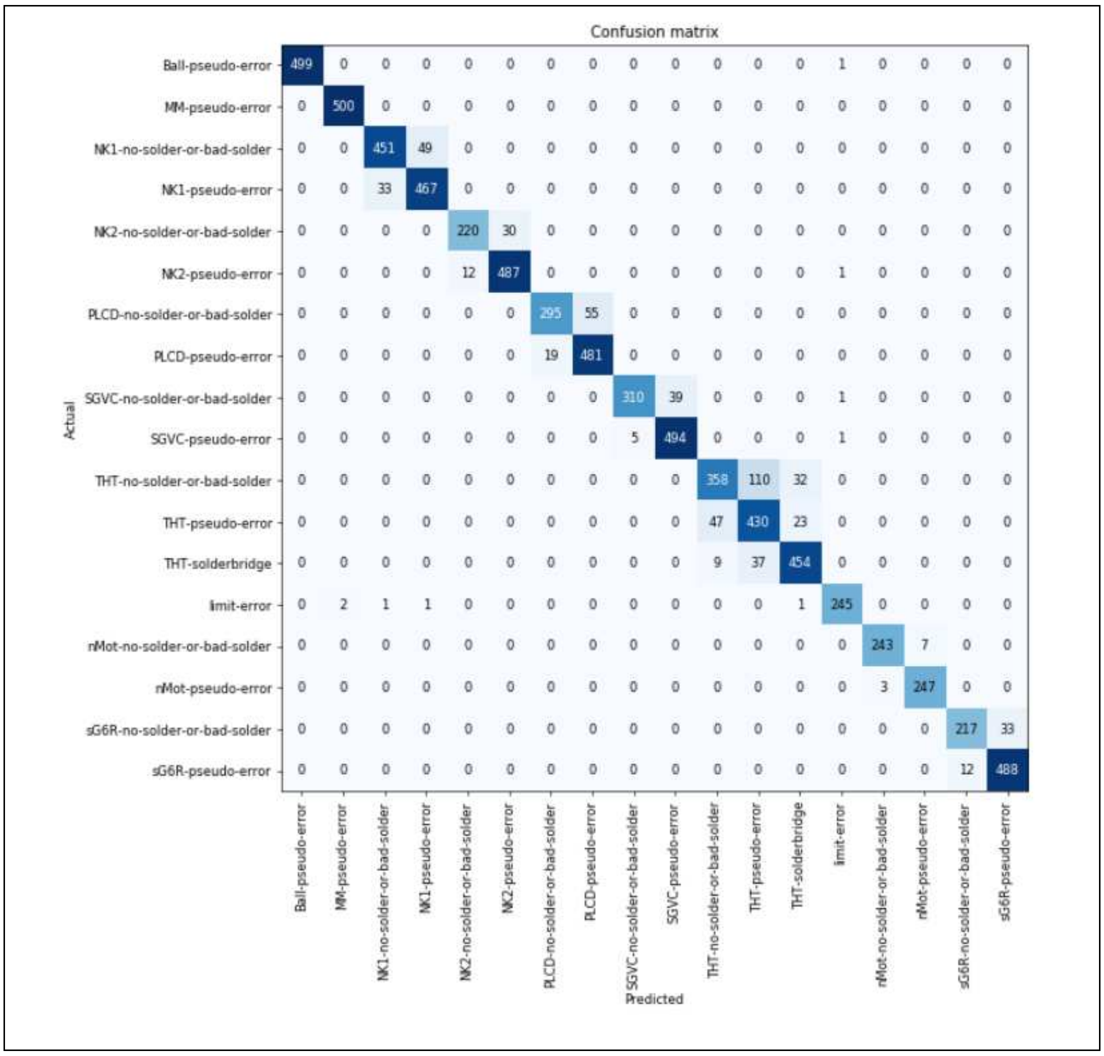
At this point in time a fully capable classifier exists. Further on, we can make use of it and ask it to classify images as trained. But just before that, there are two important steps:

4.5.1. Save model for retraining

Before going further, the model will be saved, preferably giving it a relevant name, *'xray_resnet50_18c_20b.pth'*. It might be needed to retrain it later, directly after evaluating the model if the data classification pointed out by learner is miss labelled or as it will be exposed to new data, probably it will identify unknown errors or new classes might occur or it will simply not be confident enough on some images and will need to be retrained after a power user classification of respective images.

4.5.2. Analyse the top loses

A set of function helps in model evaluation. A simple function call will create an evaluation matrix, contrasting predicted classes by the model on horizontal line with actual image classification from *Validation dataset*, as shown in Figure 20.



	Ball-pseudo-error	MM-pseudo-error	NK1-no-solder-or-bad-solder	NK1-pseudo-error	NK2-no-solder-or-bad-solder	NK2-pseudo-error	PLCD-no-solder-or-bad-solder	PLCD-pseudo-error	SGVC-no-solder-or-bad-solder	SGVC-pseudo-error	THT-no-solder-or-bad-solder	THT-pseudo-error	THT-solderbridge	limit-error	nMot-no-solder-or-bad-solder	nMot-pseudo-error	sG6R-no-solder-or-bad-solder	sG6R-pseudo-error
Ball-pseudo-error	499	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
MM-pseudo-error	0	500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NK1-no-solder-or-bad-solder	0	0	451	49	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NK1-pseudo-error	0	0	33	467	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NK2-no-solder-or-bad-solder	0	0	0	0	220	30	0	0	0	0	0	0	0	0	0	0	0	0
NK2-pseudo-error	0	0	0	0	12	487	0	0	0	0	0	0	0	1	0	0	0	0
PLCD-no-solder-or-bad-solder	0	0	0	0	0	0	295	55	0	0	0	0	0	0	0	0	0	0
PLCD-pseudo-error	0	0	0	0	0	0	19	481	0	0	0	0	0	0	0	0	0	0
SGVC-no-solder-or-bad-solder	0	0	0	0	0	0	0	0	310	39	0	0	0	1	0	0	0	0
SGVC-pseudo-error	0	0	0	0	0	0	0	0	5	494	0	0	0	1	0	0	0	0
THT-no-solder-or-bad-solder	0	0	0	0	0	0	0	0	0	0	358	110	32	0	0	0	0	0
THT-pseudo-error	0	0	0	0	0	0	0	0	0	0	47	430	23	0	0	0	0	0
THT-solderbridge	0	0	0	0	0	0	0	0	0	0	9	37	454	0	0	0	0	0
limit-error	0	2	1	1	0	0	0	0	0	0	0	0	1	245	0	0	0	0
nMot-no-solder-or-bad-solder	0	0	0	0	0	0	0	0	0	0	0	0	0	0	243	7	0	0
nMot-pseudo-error	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	247	0	0
sG6R-no-solder-or-bad-solder	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	217	33
sG6R-pseudo-error	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	488

Figure 20: Confusion matrix ResNet50 fully trained

Ideally, at the class intersection following the main diagonal we will have to have the images number corresponding to the class from validation set. To be more specific, if the class SGVC-no-solder-or-bad-solder from the validation set has 350 images and the model would have predict all 350 images as corresponding to the same class, at the intersection we would have had also 350 and 0 in rest.

The validation loss it is already known, $valid\_loss = 18{,}5\%$. Thereof, was expected to have misinterpreted images. An overview of the images that ware 'most confusing' for the model could also be displayed as complementary data for confusion matrix, as in Table 7.

Actual class like in Validation set	Predicted class by model	Number of images from actual class predicted as a different class
THT-no-solder-or-bad-solder	THT-pseudo-error	110
PLCD-no-solder-or-bad-solder	PLCD-pseudo-error	55
NK1-no-solder-or-bad-solder	NK1-pseudo-error	49
THT-pseudo-error	THT-no-solder-or-bad-solder	47
SGVC-no-solder-or-bad-solder	SGVC-pseudo-error	39
THT-solderbridge	THT-pseudo-error	37
MM-pseudo-error	NK1-no-solder-or-bad-solder	33
SGVC-no-solder-or-bad-solder	sG6R-pseudo-error	33
THT-no-solder-or-bad-solder	THT-solderbridge	32
NK2-no-solder-or-bad-solder	NK2-pseudo-error	30
THT-pseudo-error	THT-solderbridge	23
PLCD-pseudo-error	PLCD-no-solder-or-bad-solder	19
NK2-pseudo-error	NK2-no-solder-or-bad-solder	12
sG6R-pseudo-error	sG6R-no-solder-or-bad-solder	19
NK2-pseudo-error	NK2-no-solder-or-bad-solder	12
sG6R-pseudo-error	sG6R-no-solder-or-bad-solder	12
THT-solderbridge	THT-no-solder-or-bad-solder	9
nMot-no-solder-or-bad-solder	nMot-pseudo-error	7
SGVC-pseudo-error	SGVC-no-solder-or-bad-solder	5
nMot-pseudo-error	nMot-no-solder-or-bad-solder	3

Table 7: Detailed list for confusion matrix with threshold value minim 3

In this list we have on the left side, in actual, the classes corresponding to validation set from which the model classified 110 images as being of the ‘THT-pseudo-error’ class, scoring the highest confusion among classification. The list contains all the images from actual classes interpreted as being from a different class, predicted classes, with a number of images greater than a chosen number.

However, this is a detail which doesn’t points to those exact images which were misinterpreted. *Top_ losses* function is helping out here: it will display in the notebook those losses which were predicted with a high probability to be in a class but actually are included in the different class, like in Figure 21. Exactly those images should be reanalysed by responsible process engineers since can be that they were either wrong labelled or they represent border line acceptance criteria’s or other. In this case, taking image 2 from upper raw and image one from raw number 2, it could be easily observed that they are mirrored on vertical, hence representing the same confusing image for the model. The network identified them as failures, THT-no-solder-or-bad-solder, but they are part of good examples in validation set, classified as good parts under the THT-pseudo-error.

This analyse, visualization gives the possibility of the power users to review through network classifier the predefined classes, and their critical contend. At this time a data cleaning and or a retraining could be triggered following the losses analysis.

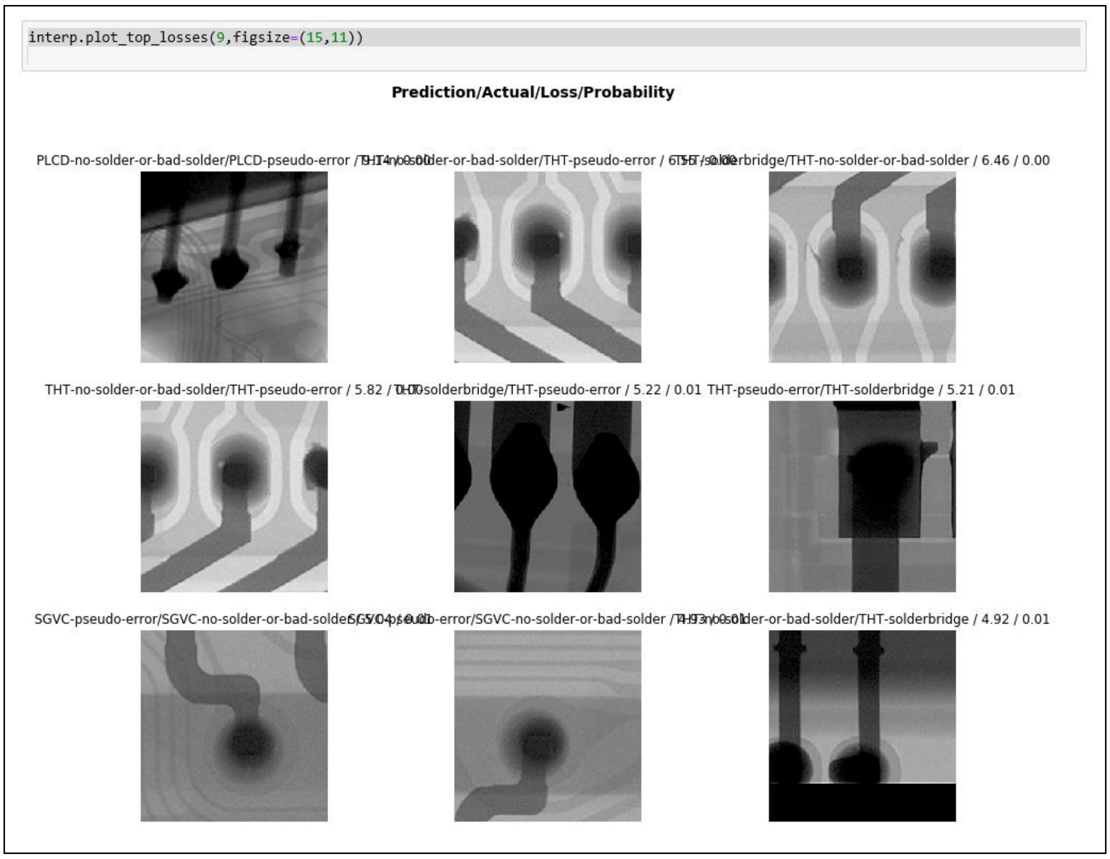

```
interp.plot_top_losses(9,figsize=(15,11))
```
Prediction/Actual/Loss/Probability
PLCD-no-solder-or-bad-solder/PLCD-pseudo-error / 9.14 / 0.00
THT-no-solder-or-bad-solder/THT-pseudo-error / 6.56 / 0.00
THT-solderbridge/THT-no-solder-or-bad-solder / 6.46 / 0.00
THT-no-solder-or-bad-solder/THT-pseudo-error / 5.82 / 0.00
THT-solderbridge/THT-pseudo-error / 5.22 / 0.01
THT-pseudo-error/THT-solderbridge / 5.21 / 0.01
SGVC-pseudo-error/SGVC-no-solder-or-bad-solder / 5.04 / 0.01
SGVC-pseudo-error/SGVC-no-solder-or-bad-solder / 4.93 / 0.01
THT-no-solder-or-bad-solder/THT-solderbridge / 4.92 / 0.01

Figure 21: Displaying the top 9 losses for the ResNet50 fully trained model

4.6. Setting a productive environment and testing in production

As we have a fully productive model, we would like to see it working. Therefore, before running in production it should be defined how it should behave when interpreting a result. A threshold representing the level of confidence while making decision should be determined. To detail threshold - accuracy concept, we calculate the threshold for each error, corresponding to target class, using the following method:

For a given class, we filter the corresponding images from the validation dataset, then pass them to the model for classification. Some of the resulted probabilities will be true positive, meaning a true predicted error or a real failure made by model during classification, some will be false positive. Hence, we have a data-model with one feature, the probabilities resulted from validation set classification, denoted ‘nn’, and a target class, correct / fail, where correct is class 1 and fail class 0, as in Figure 23. We can use this data model to train a decision tree that will help us to exclude the results from the neural network that are very probable to be false alarms. We must do this for each class so we will have 18 decision trees.

For example, for 'NK1-pseudo-error' class we have the following probabilities that resulted from the model displayed in a box plot model, as shown in Figure 22, where red values are true negatives, blue values are true positives on a scale of probability values from 0,1 to 1:

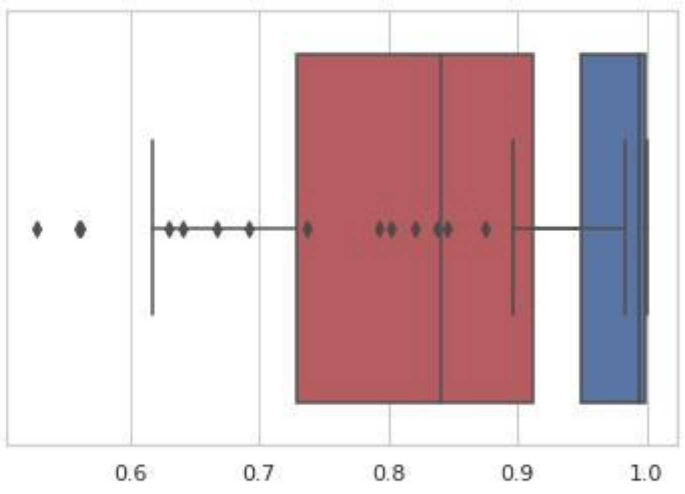


Figure 22 : NK1-pseudo-error class probabilities distribution.

and a decision tree with depth 3 would look like:

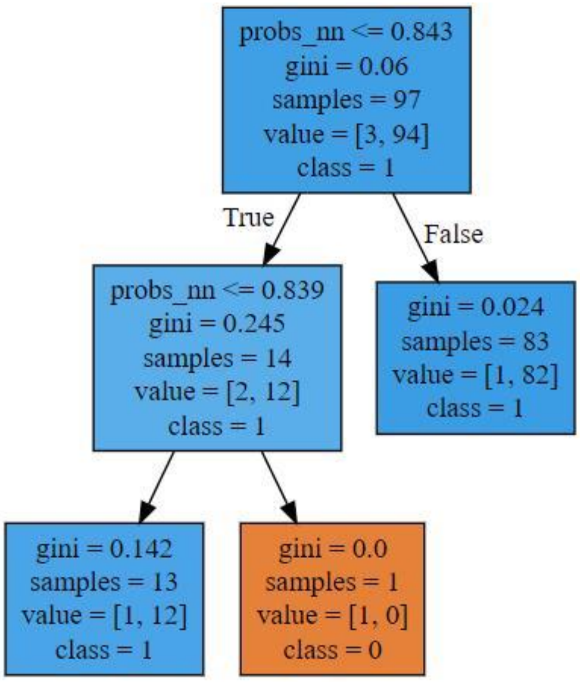


Figure 23: Decision tree to determine the probability of incorrect predictions to the corresponding class

Hence, it can be said that all NK1-pseudo-error with probability below 0.839 or equal with it are probably according with the validation set, incorrect or true negative, as shown in Figure 23.

4.6.1. Production phase, retraining

At this phase we create an application software, Rest_api, with our network loaded. A process, Apache NIFI[12] workflow, is watching the directory where images are saved by the X-Ray equipment.

Each saved image on directory, will trigger automatically a Rest_api request. The Rest_api will take the image, will classify it, and will save it to a database, Post_error, as shown in Figure 24.

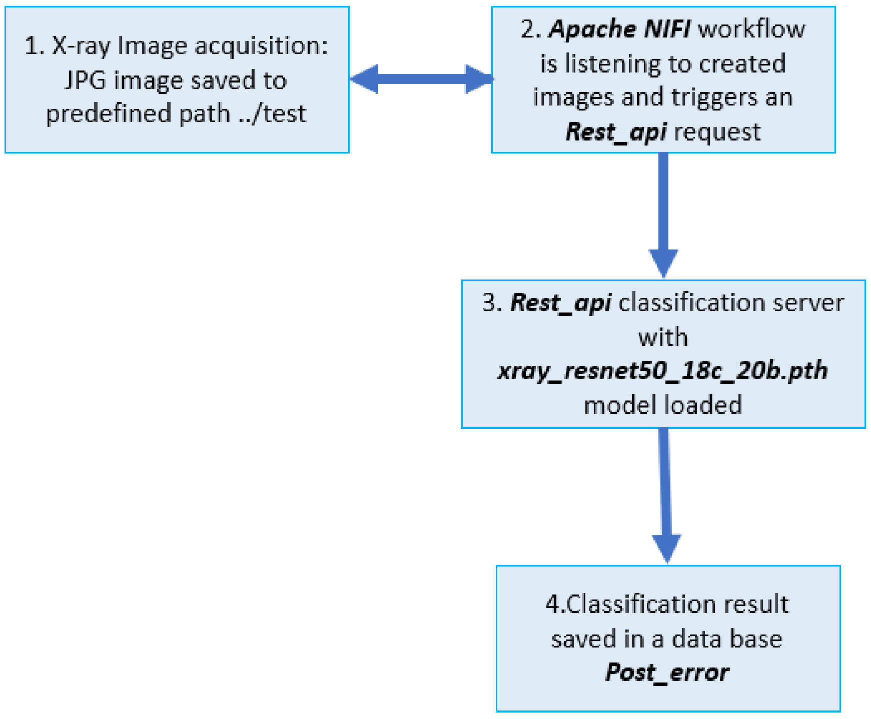

1. X-ray Image acquisition: JPG image saved to predefined path ../test
2. ***Apache NIFI*** workflow is listening to created images and triggers an ***Rest_api*** request
3. ***Rest_api*** classification server with ***xray_resnet50_18c_20b.pth*** model loaded
4.Classification result saved in a data base ***Post_error***

Figure 24: Model xray_resnet50_18c_20b.pth in production

As model is able now to do a parallel classification to the operator, evaluations loops will be performed, to periodically check its classification correctness and its confidence level, it has to have in respective classification. In case the network accuracy is low or if it decreases over time, an expert must manually correct the saved classifications results, and start again with training the model, practically going back to phase one.

[12] https://nifi.apache.org/

4.6.1.1. Limited automated decision based on confidence level

Now that the model is able to classify, it should be actually let to take decision. It is its very purpose. Therefore, it would be preferred to define the decision responsibility of the model based of a confidence level. To start with, threshold value was calculated on the test data and set to 90%.

If for an error or NOK part the confidence level is higher than the defined 90% threshold, model should take the pass-fail decision and a user notification should parallel be created. For a classified error below, the confidence level of 90%, the operator should be asked to interpret the image and classify it, so that now the decision is taken by operator. For the model there will be no immediate specific action but saving the images in the database and an expert review should follow the classification. Case by case a retraining should be planned.

This is a set-up for a model – operator tandem classification and model is still in evaluation mode.

The image below, Figure 25, describes level of decision based on 90% confidence level, deducted on the validation set, with a summary of the defined responsibility split:

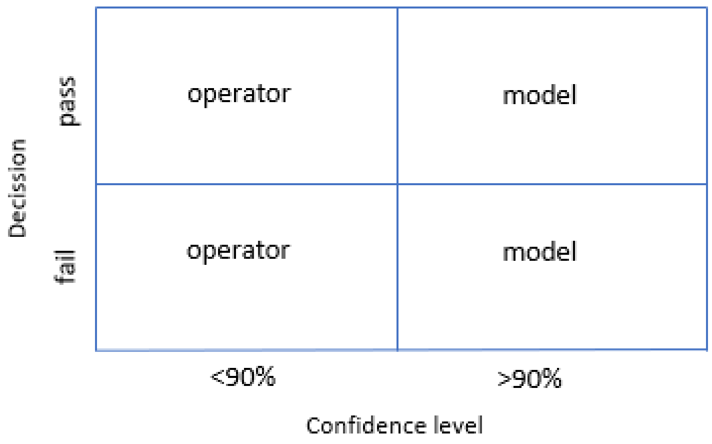


Figure 25: Split decision on classification with confidence level 90%

4.6.1.2. Model decision automation

But the thesis aims, and my goal is, to have the model autonomous and self-deciding and self-learning.

Therefore, following steps are technically to be planned after assessment of feasibility:

(1) Set automatically retraining loops with dynamic triggers based on real time classification results signalizing decreasing performance.

(2) Trigger expert opinion in case of new error types or error types combination.

(3) Install signal notification as errors tend to generalize: stop production line as in Jidoka[13] stop, on 3 consecutive failures in half an hour interval or maximum number of allowed failures in a defined time period. Line stop means do not produce waste and immediate actions are required: failure must be analysed and corrective actions to be taken before restart. Here a retraining doesn't need to be triggered.

4.7. Model classification results on a test set

To test model on a test set, there were 2,382,420 X-ray images collected. This images to be classified were unseen for my model and were unlabelled. The model used is the one loaded by the Rest_api classification application, and the confidence level set was 90%. All images classified with over 90% confidence were not re-checked by experts.

The Table 8 data shows only model capability in terms of image classification and is not to be confused or mistaken with a pareto or failure rate per inspected unit.

For each unit 20 images were acquired. Therefore, is possible that one unit has more than one error. It is as well possible to have on one unit a combination of multi pass and multi error.

Category	Confidence level	Number of images in % from test set	Number of images from test set
Errors	CL >=90%	21%	50686
Errors	CL<90%	79%	191854
Pass	CL>=90%	69%	1457500
Pass	CL <90%	31%	682380

Table 8: Test set single image evaluation with Rest_api on 2,382,420 images

This Table 8 summarizes that in a potential '*limited automatically decision mode*', limited by the minimum confidence level of 90%, the model could sole decide, on this test set, the classification result in proportion of 63% of the cases, in the end taking the pass-fail decision:

$$(50686 + 1457500)/(50686 + 191854 + 1457500 + 682380) = 0{,}63$$

All images, being errors and passes, classified with a confidence level below 90%, are being inspected and re-validated by the soldering experts.

[13] https://www.lean.org/lexicon/jidoka

After a calculation of the confidence level, succeeding an evaluation mode survey by engineering, the results could change. Supposing that the confidence level of 87 % will suffice, the model will be able to sole decide in 88 % of the cases. In contrast, if confidence level should be set higher, for example 95 %, then the model will be able to sole decide in 44% of the cases. As shown in beginning of the subchapter 4.6., Setting a productive environment and testing in production, where I've introduced class specific threshold calculation method, model has selective ability to identify the real failures corresponding to each class. Therefore, more accurate would be that we do not define here a maximum confidence level to fit all classes, but rather allow model to classify based of the specific threshold calculated for each class. In this way the model will be allowed to maximize its classification ability.

4.8. Summary and current status

At this moment, the model is a multiclass image classifier, which is being productive and capable of automated decision with a defined degree of responsibility. As the results converging at the optical inspection create a dynamic situation with a lot of variability, experts are constantly reviewing the dataset and its classes, and the classification results they consider critical or are highlighted by the model as being critical. The retraining is triggered by expert. User notifications are installed, and border line interpretations are requalified by human, followed by retraining of the model if necessary.

Chapter 5. Generalization of enabling automated image classification using CNN's in electronic manufacturing

The results obtained till now with the ResNet50 model on the complex dataset are excellent since operators are significantly de-loaded and entering a consulting mode, taking care of non-trivial classifications and driving process improvements.
In the same time experts are preoccupied with retraining, documentation, fine-tuning and extension of solution towards AOI post reflow. In addition, most important, the possibility to have a faulty ECU delivered to customer is drastically reduced.

5.1. General process development

Following process steps are derived generalizing the case study of an image classification solution based on CNN's in electronic manufacturing based:

1. Problem description
    a. Define expert team (including external resources)
    b. Define goal and success criteria
    c. Identify classification problem[14]
    d. Tools and environment assessment
    e. Create a project plan and enable operation

2. Data understanding
    a. Collect initial data
    b. Verify data quality
    c. Explore data
    d. Classify data

3. Data pre- processing
    a. Label data
    b. Quantitative representation
    c. Qualitative representation

[14] Multiclass or multi-label classification

    d. Experts cross check

4. Setting environment
    a. Create Jupiter notebook
    b. Import libraries
    c. Import frameworks
    d. Set cloud GPU computing
    e. Set Storage

5. Data pre-process
    a. Import data
    b. Data augmentation
    c. Data normalization
    d. Create Training set
    e. Create Validation set
    f. Create Test set (optional)
    g. Create data object

6. Training of the model
    a. Select appropriate (pretrained) model
    b. Learn with default learning rate @ data object
    c. Save model
    d. Run learning rate finder
    e. Unfreeze model
    f. Train hole model with appropriate learning rates
    g. Save model

7. Start Classification interpretation
    a. Analyse confusion matrix and most confused
    b. Analyse top losses
    c. Clean data

8. Retraining the model
    a. Actualize data
    b. Train the model: go to Training

9. Create classification application
    a. Load model
    b. Connect the application to image source
    c. Determine threshold
    d. Set the minimum accepted confidence level for decision

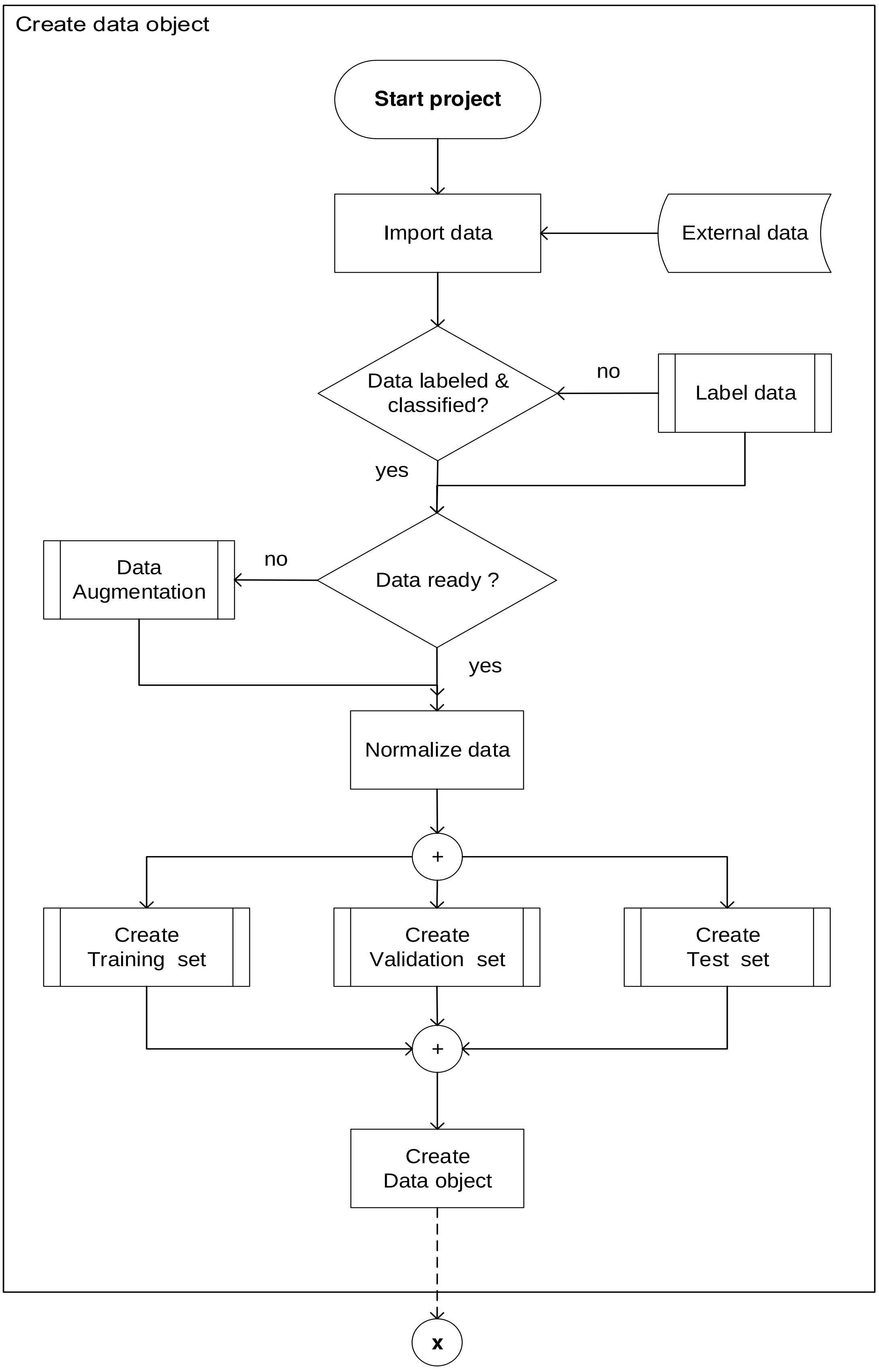
10. Classification and testing model
    a. Classify new data
    b. Save classification in a database
    c. Monitor accuracy
    d. Trigger retraining on accuracy losses

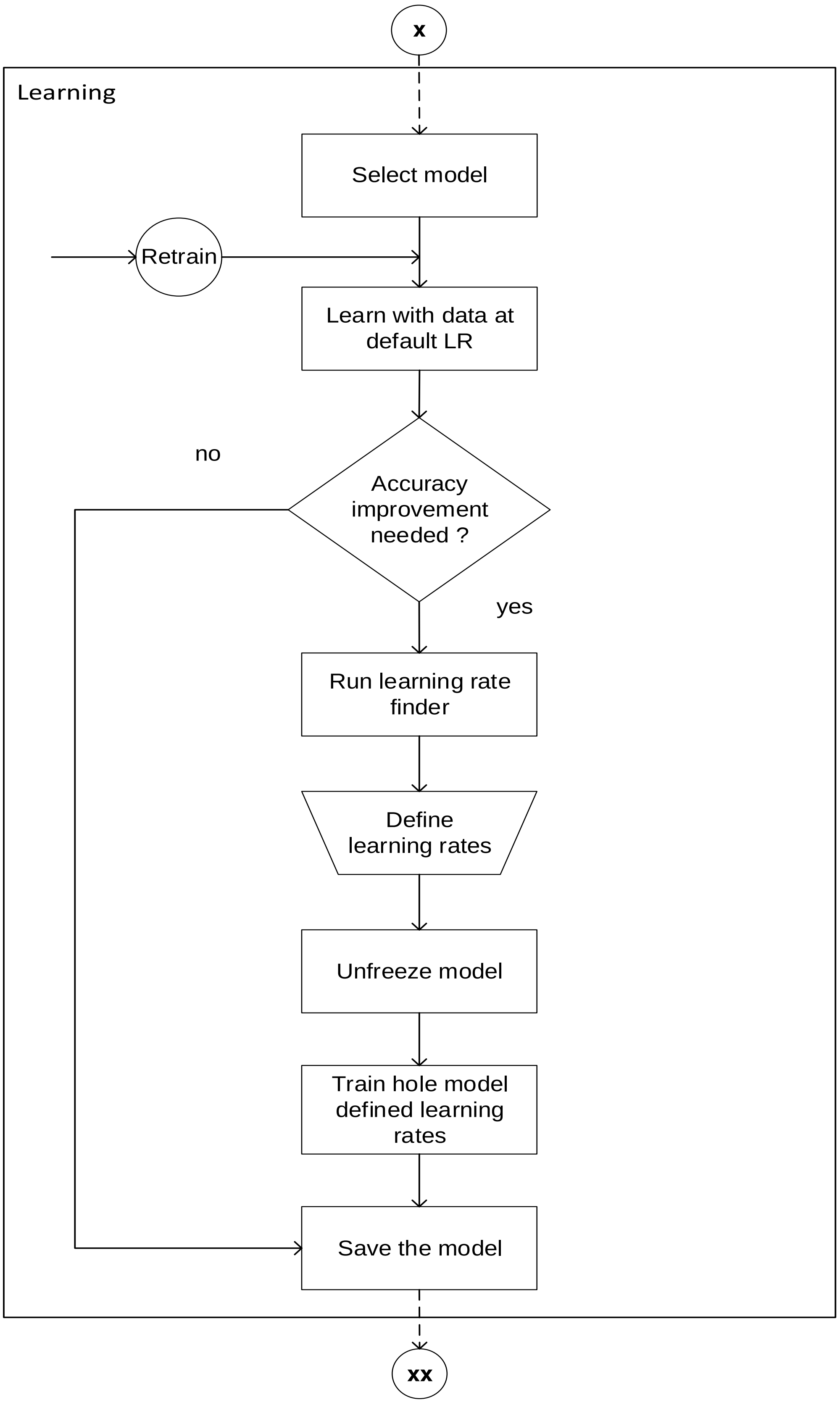
11. Decision degrees of implementation within production
    a. Learn and monitor (passive mode for production)
    b. Support decision (qualification by model, decision 100% by operator)
    c. Limited automated decision based on confidence level
    d. Fully automated decision

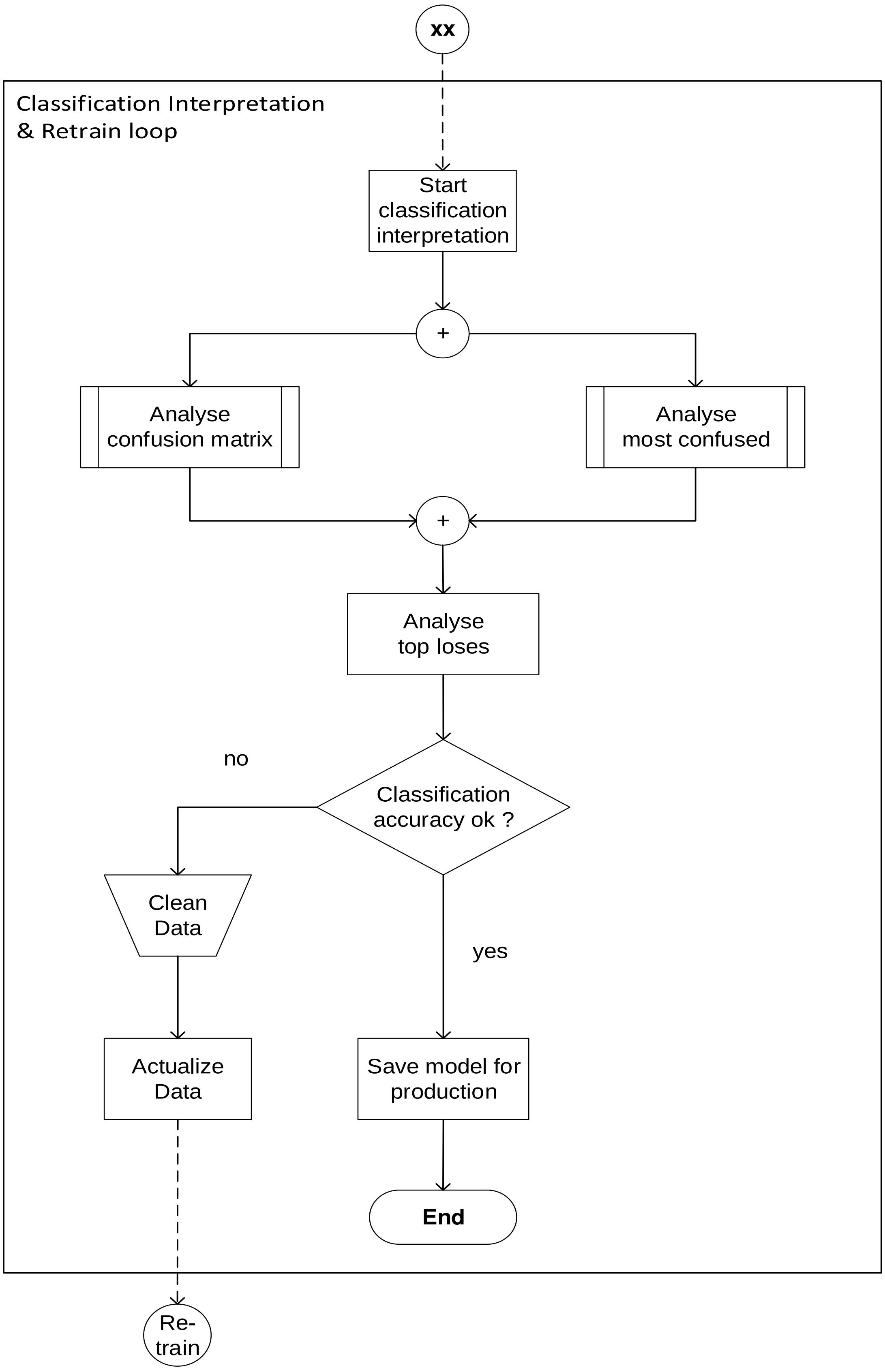
The steps 1 to 3, will be commented next, and will also be detailed in Chapter 6. , 6.1. Limitation, correlated with organizational capabilities and its set-up based on strategy.
Before solving a problem, an organization needs to be able to identify it, define it, classify it and formulate organizational and business impact. Now problem solving has to be handled in terms of resources, capabilities and infrastructure. Solving a problem is not always generalizing a solution nor assuring that current solution will suffice in the future or changing environment. Therefore, a maintenance loop should be designed together with a constant inquiry if solution or setup is economically and qualitative competitive for the organization.
Setting up the environment from Step 4 was already described for the case study. More detailed information's will be provided in Chapter 6, 6.2. Recommendation and will be combined with organizational set-up and capability. It is as well related with company strategy.
Step 11 was already described during case study and will be commented in subchapter 6.3. It is related with the result of case study and by generalization is answering the main research questions of the master thesis.
Next, the process flow diagram schematic will be presented for the steps beginning with 5th step to 10th step, to help visualize how this process steps are connected as they represent the pure implementation.

5.2. Process flow generalization diagram

The process flow diagram for the process steps 5 to 10, Figure 26, is illustrated below:


Create data object
Start project
Import data
External data
Data labeled & classified?
no
Label data
yes
Data Augmentation
no
Data ready ?
yes
Normalize data
+
Create Training set
Create Validation set
Create Test set
+
Create Data object
x


x
Learning
Select model
Retrain
Learn with data at default LR
no
Accuracy improvement needed ?
yes
Run learning rate finder
Define learning rates
Unfreeze model
Train hole model defined learning rates
Save the model
xx


xx
Classification Interpretation & Retrain loop
Start classification interpretation
+
Analyse confusion matrix
Analyse most confused
+
Analyse top loses
Classification accuracy ok ?
no
yes
Clean Data
Actualize Data
Save model for production
End
Re-train

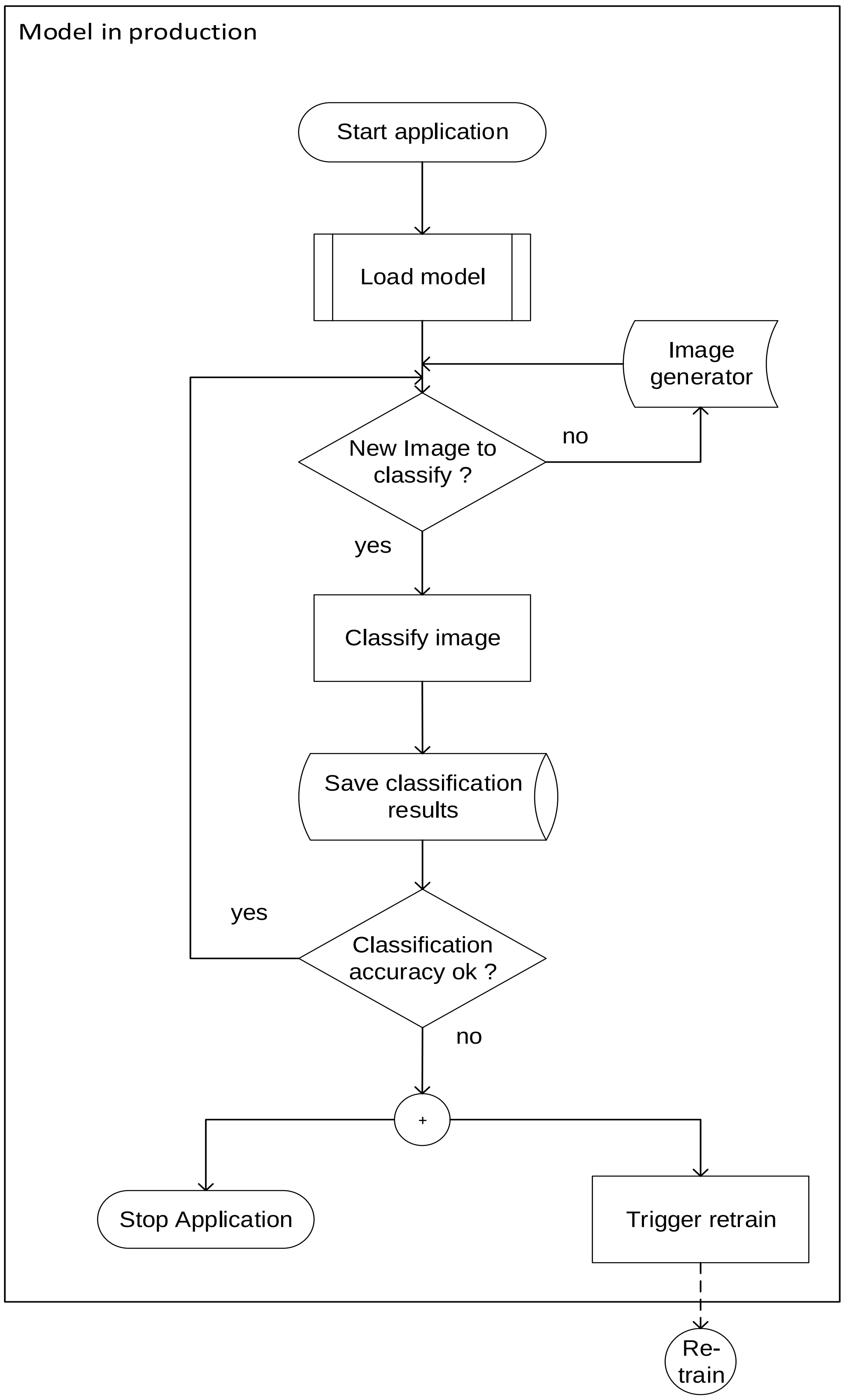


Figure 26: Process flow generalization step 5-10

Chapter 6. Conclusions

The thesis last chapter is summarizing experience through deep learning and practical application with the case study, and of course it will answer the driving research questions. This was a quite long journey for me even though my knowledge in both study fields I personally consider that being quite strong.

Therefore, for each manufacturing company who would like to benefit using deep learning and machine learning as technology boosters both in data management and using data to create a competitive advantage through innovative problem solving, with the purpose of being faster and more accurate in decision making, a fundamental question is arising: Ready for innovative changes?

6.1. Limitation

The journey through machine learning and deep learning is rather difficult and not trivial view nor easy understanding is to be expected.

Before even starting, a company should asses its internal knowledge based on current setup. Understanding its capability and background, its mandatory to be able to form an objective conclusion answering the question: What are our capabilities?

And this question should scan the company setup, going through quality management system and organizational structure. The biggest limitation I've encounter during case study definition, solving and deployment were:

(1) Data understanding at all disciplines level
(2) Structural access, limitation by need to know principle
(3) Rules and regulation, ownership of data
(4) Quality rules and change management process
(5) Industrial engineering and supplier base
(6) IT and infrastructure
(7) Data scientist business understanding
(8) Programming skills and experience

A first approach for a company with not in-house capabilities to develop such application is to turn to an external provider and build from there on as the projects are developing or progressing. However, to be able to formulate such problems and possible solution, certain know how in programming, mathematics and machine learning must exist. Those basic skills with business understanding should be a good starting point in exploring alternative innovative solutions.

6.2. Recommendation

Each electronic manufacturing organization should have an assessment method on how to start using deep learning on image classification. This could come after evaluating internal capabilities or consulting some of the service providers.

Turning to external providers might not be wrong at all. However, be ready and expect that they are not at all familiar with your business, so they lack in business understanding which you have to offer. Besides that, you will offer access to all your information and know how. Of course, therefore the Non-Disclosures Agreement are there.

Starting with in house know how grows building might be time, resource consuming even slightly demotivating if progress is not timewise obviously. This is to be combined with expert acquisition and upgrading of IT infrastructure as their recommendation.

Here we do not have a classical make – buy decision process to follow.

I believe in a combination of both: seeking consulting if you start from scratch and build up your own capability, even partnering with universities or other companies would help.

Next, I will describe an alternative based on a hybrid set-up, when programming capability is in house, and your company has a developed IT infrastructure:

If programming skills are available, in C++, or Java, then turning to Python under Jupyter notebook, would not be a problem. From there on, skills can easily be built on since high level frameworks and computing power are helping in our days more than ever.

Further on, I recommend visiting the Fast.ai online course, super easily accessible (Howard 2019). There are crystal clear explanation and examples on how to think, program and create deep learning application in image classification. The Fastai library provides explanations, code and repositories and gives access to communities and forums where you can gain from experienced practitioners and programmers.

In my case study, to detail on infrastructure, Jupiter notebook, Version 1.0.0. was used. It can connect with many kernels to allow programming in many languages. Is incorporating by default the IPython kernel, in here with Python version 3.7.6., an open source environment.

The deep learning libraries used are PyTorch[15] Version 1.1.0 by Facebook AI research laboratories and Fastai[16] Version 1.0.55 by Fast.ai build on PyTorch, both are open source.

[15] https://pytorch.org/

[16] https://docs.fast.ai/

These projects are requiring high computational power, therefore, needing that, an online cloud computing can help, in my case Amazon EC2[17] was used, with GPU based on NVIDIA CUDA library 10.1, Tesla K80 GPU 0/11441MiB.

Since everything should be made available for computing, in the presented case study the large number of images, a compatible storage system will be needed. Amazon services were considered as well and used S3[18] storage system.

As alternative, ready to use environments to start with deep learning from the browser are described and guidance is offered. Probably the easiest way is to use a cloud service, accessible through web browser from Crestle[19] or Paperspace[20] or many others. Both service providers offer an already set up cloud service with preinstalled and needed frameworks, which are already configured to run on a GPU in the cloud. This kind of information is also part of Fastai course (Thomas 2017).

Thesis focused was on image classification, multilabel and multiclass classification, and the special application on optical inspection systems image classification, as they are a mandatory step in quality assurance in electronic manufacturing. Having this problem improved using deep learning, in my case residual neural networks, a lot more connected application could be developed, which are targeting to improve time wise, and cost wise your operation like process quality monitoring, maintenance and equipment performance, supplier quality monitoring and classification. This information can be shared in a network if operation has more than one production facility in one or more countries. And huge opportunity can be derived. With consolidated information's there are more objective assessments or possible, of technology providers, suppliers but also own manufacturing capabilities and performance.

6.3. Answering the main research questions of the master thesis

The driving research questions were over and over attacked during the thesis chapter. Thus, I will only concatenate the answer together:

1) When and for which category of problems can we successful use deep learning in electronic manufacturing production

[17] https://aws.amazon.com/de/ec2/
[18] https://aws.amazon.com/de/s3/
[19] https://crestle.ai/
[20] https://www.paperspace.com/

The thesis focused on introducing an autonomous and automated solution for the electronic manufacturing production at the optical inspection stations. This station represents a validation process of previous steps. It can succeed a soldering process or any kind of optical inspection like contact or connector pins of an electronic control unit, labels, seals or gasket inspections. Equipped with an acquisition system, this station requires an evaluation in most of the cases by operator. This is a classification problem, therefore part to supervised learning, which can be entrusted to a convolutional neural network. Thus, deep learning can be applied for image classification, being a multi label or multiclass classification.

2) A trained convolutional neuronal network in X-Ray image recognition can replace human driven classification at the optical inspection stations, with other words, is the solution based on deep learning qualitatively more efficient, quicker and reliable then socio-technical system? When should a clear cut occur?

The case study demonstrated that a convolutional neural network is able to classify with an accuracy of 93% the images acquired by the X-ray system succeeding 4 soldering steps of different components on subassembly. The final solution of the productive model is however still to be finetuned and concluded with responsible engineers. Four sub- steps were defined during implementation of the model in production:

(1) Learn and monitor (passive mode for production)
(2) Support decision (qualification by model, decision 100% by operator)
(3) Limited automated decision based on confidence level
(4) Fully automated decision

The first two sub-steps ware trivially solved. With confidence level calculated on the *validation set* of 90%, the ResNet50 model was asked to classify on a test set of 2,382,420 images. In 63% of the cases it would have been able to take the pass-fail decision because the rest of 37 % of the images where classified with a confidence level under the set accuracy of 90%. After a recalculation of the confidence level, succeeding an evaluation mode survey by engineering, the results could change. An improvement can only be anticipate at this time, however supposition should be proven further on during production, analyses and eventual retraining loops. My personal expectation is that the model autonomic classification should go up to 85% from images he is intrusted to classify. With this, the limited automated decision is to be configured, calculated and rolled out after the needed remaining steps.

Coming to last step, of autonomous and fully automated decision, I am only able to comment it. There is in progress a process development through which the model could react on it's on classification performance, triggering automated retraining loops so that it could learn to increase its accuracy. This concept has been introduced in subchapter Model decision automation, 4.6.1.2. However, this would have to trigger a series of processes, maybe including classification requests to engineering which would probably create a delay in running production. Therefore, a procedure should deal with all possible aspects, even designing unconventional solutions. When dealing with 24 h production; like for example installing a worldwide network of power users to be reached out every time, which in case of an image qualification / requalification request from the model dealing with a lower accuracy could be notified by sending the image or images and asking for feedback within a certain time period. It will then use the feedback for retraining purpose, after data classification like in retraining loop from Figure 26.

The model is therefore not yet able to be fully autonomous in this complex case, being however able to significantly de-load the operator activities.

There are many other opportunities as described where autonomous model can sole perform classification tasks.

3) What are the generic steps to be taken as a successive approach to enable such technical self-deciding systems?

This last question was answered trough detailed process description and process flow diagram in Chapter 5. and Figure 26. Infrastructure and needed knowhow, as well a recommendation how to approach a know how ramp up within your company, were described in the previous subchapter Recommendation.

6.4. Outlook

The scope of the master thesis to install a fully automated and autonomous system could not have been completed as this is rather a complex problem, which require not only a classification model development. Beginning with the organization, data collection and ending with a product and its qualitative requirement measured in form of customer satisfaction, this problem solving requires a validation of successive changes that only can add value never deteriorating an established quality stand.

6.5. Closing statement

Due open source access and supportive solution with high-level frameworks and web based fully configured environments, is more than easy to use the power of machine learning and especially deep learning within almost any electronic manufacturing organization and not only. It is the organization capability and its people innovative thinking to search and explore problem solving with these powerful models.

List of figures

List of Equation

List of tables

Bibliography

Balsys, Rokas. 2019. "Convolutional Neural Networks(CNN) explained." *pylessons.com.* May 08. Accessed Jan 2020. https://pylessons.com/CNN-tutorial-introduction/.

Bhatt , Shweta. 2018. "Reinforcement Learning 101." *Towards data science.* March 19. Accessed January 2020. https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292?gi=d2c476b26315.

Bishop, C. M. 2006. *Pattern recognition and machine learning.* New York: Springer.

Buduma, N. 2016. *Fundamentals of Deep Learning: Designing Next Generation Artificial Intelligence Algorithms.* Sebastopol, CA 95472: O'Reilly Media Inc.

Chung, J, C Gulcehre, K Cho, und Y Bengio. 2015. „Gated Feedback Recurrent Neural Networks." *ICML.*

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning.* MIT Press.

Graves, A. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks.* Springer.

Gu, J., Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, et al. 2018. "Recent Advances in Convolutional Neural Networks." *Pattern Recognition*, May. doi:10.1016/j.patcog.2017.10.013.

He, K., X. Zhang, S. Ren, und J. Sun. 2015. *Deep Residual Learning for Image Recognition.* Microsoft Research, IEEE, Boston: Computer Vision and Pattern Recognition. doi:arXiv:1512.03385v1.

Howard, Jeremy. 2019. "Practical Deep Learning for Coders, v3." *Fast.ai.* Accessed January 2020. https://www.fast.ai/.

Hsu, F.H. 2002. *Behind Deep Blue: Building the Computer That Defeated the World Chess Champion.* Priceton, NJ,USA: Princeton University Press.

Inteliment. 2019. *Let's Understand the Difference between Machine Learning Vs. Deep Learning.* September 18. Accessed December 2019. https://www.inteliment.com/blog/our-thinking/lets-understand-the-difference-between-machine-learning-vs-deep-learning.

Ioffe, S., and C. Szegedy. 2015. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* arXiv e-prints. doi:arXiv:1502.03167.

Jarrett, K., Kavukcuoglu, Ranzato, M. K, and Y., LeCun. 2009. "What is the best multi-stage architecture for object recognition?" *ICCV.* Kyoto: IEEE. doi:10.1109/ICCV.2009.5459469 .

Joshi, Naveen. 2019. "Cognitive World." *Forbes.* June 19. Accessed December 2019. https://www.forbes.com/sites/cognitiveworld/2019/06/19/7-types-of-artificial-intelligence/#37b7da233ee7.

Karpathy, A. 2019. "Convolutional Neural Networks for Visual Recognition." *GitHub.* Accessed December 2019. http://cs231n.github.io/neural-networks-1/.

Kleinsmith, M. 2017. "CNN's from different viewpoints." *Medium.com.* Accessed December 2020. https://medium.com/impactai/cnns-from-different-viewpoints-fab7f52d159c.

LeCun, Y., C. Cortes, and C. J. C. Burges. 1998. *THE MNIST DATABASE.* Accessed December 2019. http://yann.lecun.com/exdb/mnist/.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner. 1998. "Gradient-Based Learning Applied to Document Recognition." *Proceedings of the IEEE .* IEEE. doi:10.1109/5.726791.

Loshchilov, Ilya, and Frank Hutter. 2017. "SGDR: Stochastic Gradient Descent with Warm Restarts." *ICLR.* Toulon.

Lovelace, Ada. 1843. „Notes upon L.F. Menabrea's " Sketch of the Analytical Engine invented by Charles Babbage"." *Scientific Memoirs Selected from the Transactions of Foreign Academies of Science and Learned Societies*, 691-731.

McCarthy, John. 2007. *What is AI?* November 12. Accessed November 2019. http://www-formal.stanford.edu/jmc/whatisai.pdf.

Müller, A. C., und S. Guido. 2016. *Introduction to Machine Learning with Python: A Guide for Data Scientists.* Sebastopol: O'Reilly Madia Inc.

Nielsen, M. A. 2015. *Neural networks and Deep Learning.* Determination Press.

Parkhi, O.M., A. Vedaldi, A. Zisserman, und C. V. Jawahar. 2012. „Cats and dogs." *Conference on Computer Vision and Pattern Recognition.* Providence: IEEE. Zugriff am Jan 2020. doi:10.1109/CVPR.2012.6248092 .

Parki, O.M., A. Vedaldi, und A. Zissermann. 2015. „Deep Face Recognition." Paper, Visual Geometry Group , Department of Engineering Science, University of Oxford, Oxford. Zugriff am January 2020. http://www.robots.ox.ac.uk/~vgg/publications/2015/Parkhi15/parkhi15.pdf.

Saha, Sumit. 2018. "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way." *Towards data science.* Dec 15. Accessed Jan 2020. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

Shalev-Shwartz, S. and Ben-David, S. 2014. *Understanding Machine Learning: From Theory to Algorithms.* New York: Cambridge University Press.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., et al.,. 2016. „Mastering the game of go with deep neural networks and tree search.“ *Nature 529* (Nature) 484-489.

Simonyan, Karen, and A Zisserman. 2015. "Very deep convolutional networks for large-scale image recognition." *International Conference on Learning Representations.* San Diego: ICLR. doi:1409.1556v6.

Sutton, R. S., und A. G. Barto. 2018. *Reinforcement Learning: An Introduction.* Second Edition. Cambridge , MA: The MIT Press.

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, D. Anguelov, S. Reed, D Erhan, V. Vanhoucke, and A. Rabinovich. 2015. "Going deeper with convolutions." *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* Boston: IEEE - Institut of Electrical and Electronis Engineers. doi:10.1109/CVPR.2015.7298594.

Thomas, R. 2017. "What you need to do deep learning." *Fast.ai.* Nov 16. Accessed January 2020. https://www.fast.ai/2017/11/16/what-you-need/.

Zeiler, M. D., G. W. Taylor, and R. Fergus. 2011. "Adaptive deconvolutional networks for mid and high level feature learning." *ICCV.* Barcelona: IEEE. doi:10.1109/ICCV.2011.6126474 .

Zeiler, Matthew D., and Rob Fergus. 2014. "Visualizing and Understanding Convolutional Networks." *European Conference on Computer Vision.* Zurich: Springer.

Zhang A., Lipton Z. C., Li M., and Smola A. J. 2019. *Dive into Deep Learning. Retrieve from UC Berkeley.* Berkeley, CA: Spring.