Professional MBA
Entrepreneurship & Innovation

Interchangeability of neural networks in different industry settings: An approach of deriving a general framework for the implementation of predictive machine actions.

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

supervised by
Prof. Dr. Marc Gruber

Alexander Antonic, BSc (WU)

00851746

Vienna, 30.06.2019

CONTINUING EDUCATION CENTER

EXECUTIVE ACADEMY

Affidavit

I, **ALEXANDER ANTONIC, BSC (WU)**, hereby declare

1. that I am the sole author of the present Master's Thesis, "INTERCHANGEABILITY OF NEURAL NETWORKS IN DIFFERENT INDUSTRY SETTINGS: AN APPROACH OF DERIVING A GENERAL FRAMEWORK FOR THE IMPLEMENTATION OF PREDICTIVE MACHINE ACTIONS.", 77 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, 30.06.2019

[illegible]

Signature

Abstract

The aim of this master thesis is to propose a general framework that facilitates the implementation of neural networks in the manufacturing industry and beyond. The process steps described in this framework provide a structured approach and serve as a guide if neural networks are considered to be implemented in an industrial setting. Furthermore, having and following this approach not only saves time as there is a risk of overlooking certain steps, which then need to be reconsidered, but can also increase the accuracy of the model itself.

Moreover, a key point of this thesis is the question, when and for which problems state-of-the-art artificial neural networks can be applied in manufacturing. To answer this, first an extensive review of recent academic literature was conducted. Secondly, an analysis was performed, which includes the qualitative assessment of two recent use-cases of neural networks used in a manufacturing context. Based on the findings of the literature review as well as the qualitative assessment a general framework was derived. In order to prove the usability of this framework was used to self-develop a neural network application solving an existent problem in supply chain management and procurement, specifically the prediction of whether a delivery of goods will be on time or late. Subsequently, the performance of the network and its prediction accuracy were validated.

Indeed, it was observed that neural networks are most suitable for solving regression and classification problems, whereas for manufacturing the focus is on classification problems. In addition, a general framework was derived which was also used to self-develop a neural network. This neural network was not only able to solve the underlying problem of predicting and classifying on time or late deliveries of goods, but also achieved a performance measured by binary accuracy of 0.82, a validation accuracy for on time deliveries of 0.82 and a validation accuracy for late deliveries of 0.72.

Keywords: machine learning, neural networks, deep learning, artificial intelligence, classification, on time deliveries, manufacturing, prediction, general framework, software, implementation

Acknowledgement

I would first like to thank my thesis advisor Prof. Marc Gruber at the College of Management of Technology at École Polytechnique Fédérale de Lausanne. He immediately responded whenever I ran into trouble or had a question about my research topic or the writing process. Moreover, he gave me his trust and allowed this paper to be my own work. His guidance steered me in the right direction and made me focus on what was most important.

Moreover, I would like to thank my work colleague and friend Emir who was involved in the programming of the neural network for this thesis. Without his passionate participation and input, on-time delivery prediction could not have been successfully developed.

Finally, I would also like to thank my dear companion and best friend Petra. Nobody has been more important to me in the pursuit of this project. Her love and understanding are with me in whatever I pursue. Furthermore, I would like to thank my parents for their great support in all aspects which helped me not only to finish my studies but also to accelerate in life. They are my ultimate role models.

[illegible]

Alexander Antonic

Table of Contents

Table of Figures

List of Tables

1. Introduction

The rapid development in the field of information and communication technology has led to many revolutionary disciplines and improvements in artificial intelligence (AI). Recently, AI is penetrating the manufacturing industry, where it contributes to better quality, lower costs and reduced time to market (Rittinghouse & Ransome, 2009). In addition, artificial intelligence enables manufacturers to decrease their conversion costs by up to 20 percent. Up to 70 percent of this cost reduction is caused by higher workforce productivity as it can reduce manual labor and repetitive activities in manufacturing processes (Küpper, et al., 2018). Intelligent computer models can predict delivery times, calculate the optimal routes under the assumption of minimal costs and anticipate failure of machine components. As a result, expensive product recalls or disruptions in production can be prevented. This is especially important in industries with low yields or small margins. In addition, artificial intelligence will make it easier for companies to share production lines as well as technologies across different industries. The use of AI might enable companies to anticipate potential damages and quality defects of goods produced by manufacturers (Kusiak, 2017). Therefore, the use of AI is becoming a key factor for increasing productivity of manufacturing operations. Manufacturers will play an essential role in driving the success of AI by merging it into their machinery and equipment (Küpper, et al., 2018).

Although, the vast amount of data and the increase of processing power contributed to dramatic improvements of AI, current technological implementations of artificial intelligent applications are still mostly limited to specific areas like dialogue response as well as image and speech recognition. Therefore, AI is suitable for specific types of problems with a limited set of possibilities where certain results can be expected. To be more specific, machine learning algorithms perform well at extracting particular patterns, but in the real world there can be an infinite number of possibilities causing infinite data extraction and calculation times due to overloads of database systems. However, deep learning is dependent on large-scale numerical data inputs (Lu, Li, Chen, Kim, & Serikawa, 2017). Nevertheless, the data needs to be normalized, gathered with appropriate speeds and frequencies, stored, shared and processed. At the same time, data privacy and security issues have to be addressed as well (Kusiak, 2017). Furthermore, artificial intelligent systems currently cannot associate data like the human brain, resulting in outputs that can be easily misinterpreted and incorrectly used (Lu, Li, Chen, Kim, & Serikawa, 2017).

Currently, leading industries in energy, aircraft, computing and semiconductor manufacturing face difficult challenges when attempting to implement AI. This is because, many companies do not know what to do with their data or how to use machine learning algorithms properly to improve their processes in general. Furthermore, most of the companies think that their databases are too big and complex for analysis. Companies which failed to implement AI in an integrated manner must react fast by adopting a more structured approach (Küpper, et al., 2018).

Apart from this, machinery and automation producers have to identify shortcomings that AI might be able to address by determining enablers and providing solutions, which have to be tested and scaled up (Küpper, et al., 2018). Hence, one of the gaps which needs to be filled by academic research is the practical application of artificial intelligence (Kusiak, 2017). Only about one in five companies has implemented AI in one way or another, which shows that there is a large gap between aspiration and successful execution (Ransbotham, Kiron, Gerbert, & Reeves, 2017).

1.1 Problem Statement

Ultimately, this presents manufacturing companies not only with the issue of misinterpreting or using artificial intelligence incorrectly, but also with not knowing how and for which processes AI can and should be implemented. Moreover, manufacturers face difficulties in dealing with data. This is the reason why producers need to have a structured approach or framework which will enable them to understand artificial intelligence in general and to identify suitable applications.

Therefore, I formulated the following questions I would like to answer in this thesis:

(1) When and for which problems can state-of-the-art artificial neural networks be applied in manufacturing?

(2) Whether, and to what extent can a general framework be derived by analyzing existing applications of artificial neural networks in different industry settings?

(3) Whether and to what extent an artificial neural network can be developed to solve the problem of predicting and classifying on time deliveries (OTD) for supply chain management with this framework?

To be specific, point 1 was formulated with the intention to analyze state-of-the-art neural networks in the manufacturing industry. In detail, I aim to assess current applications of NN and their contribution to problem solving in different manufacturing settings. In the course of this process I aim to examine both the theoretical approaches and the practical applications to derive a potential standard implementation framework for neural networks, which ideally can be used in different industry settings. This is why I formulated the second question. Furthermore, this set of standard circumstances is supposed to be used to develop a neural network, which helps to solve a real-world problem in supply chain management, namely the prediction of on time deliveries. In fact, the data needed for the development process of the neural network will be provided by a renowned German manufacturing company. Ultimately, the network will solve the underlying problem of the manufacturer and thereby will be integrated into our company's current product portfolio, where it will serve as a new microservice software solution. Hence, the formulation of the third problem.

1.2 Objective of the Thesis

Thus, the aim of the thesis is to derive a general framework to be suitable for a practical implementation of artificial neural networks by analyzing academic papers as well as real world use cases across different industry settings. To be specific, a set of standard circumstances will be recommended for the implementation of neuronal networks to solve regression and classification problems.

Accordingly, the thesis will attempt to achieve the following objectives:

(1) To identify, analyze and propose recommendations for the use of neural networks in the manufacturing industry.

(2) To identify, analyze and assess the prediction performance of a self-developed neural network using binary accuracy (BACC) as measurement.

(3) To identify, analyze and indicate a general framework for practical implementation of deep neural networks in the manufacturing industry.

1.3 Methodological Approach

My research process started by studying subject-matter academic literature. First, historical academic papers were reviewed to build up an understanding of how artificial intelligence evolved from different scientific disciplines over time. Second, for machine learning and its subdiscipline, deep learning, various resources were taken into consideration. On the one hand, reports and specialist publications were used. On the other hand, due to the fact that deep learning started to become prominent between 2006 and 2012 most of my attention was focused on academic papers, journals, books and dissertations issued during that period and thereafter. Moreover, the increasing attention neural networks have received throughout recent years generated a highly dynamic scientific field where breakthroughs and records are occurring frequently. This is why my research is heavily based on up-to-date literature, such as the qualitative assessment of the practical applications of neural networks in manufacturing, which was based on two recent academic use-cases. This was mainly due to the fact that none out of the three contacted manufacturing companies in Austria applied neural networks in their manufacturing process. The core of the research was to derive a framework for the implementation of neural networks to solve regression and classification problems based on the conducted research and findings. Furthermore, the usability of the framework was tested by developing a real neural network application. The output and accuracy of the neural network was then validated quantitatively by the binary accuracy statistical measurement.

1.4 Structure of the thesis

Chapter 1 states the importance that manufacturers need to adapt a structured approach for the implementation of artificial intelligence to stay competitive. Moreover, the chapter underlines the necessity that companies have to identify shortcomings that AI might be able to address. Further, it is mentioned that manufacturers need to determine enablers, test solutions and scale them. The chapter also includes the research questions and describes the structure of the thesis. **Chapter 2, 3 and 4** review the history, introduce existing concepts and elaborate on the methods related to machine and deep learning. Thus, state-of-the art theoretical and practical concepts were considered and summarized. **Chapter 5** assesses state-of-the-art applications of neural networks in the manufacturing industry. For this purpose, practical use-cases were analyzed qualitatively. **Chapter 6** derives a general framework for the implementation of neural networks based on the insights gained from chapter 2 – 4 as well as from the qualitative analysis of the practical use-cases assessed in chapter 5. In **Chapter 7** a neural network is developed with the use of the general framework derived in chapter 6 to solve the classification problem of predicting and classifying on time deliveries. **Chapter 8** provides a summary and discusses the thesis limitations. Finally, research recommendations are proposed for implementing neural networks in the manufacturing industry.

2. Introduction to Artificial Intelligence

2.1 Definition of AI

Artificial Intelligence as we know it today has evolved from many diversified disciplines like philosophy, mathematics, economics, neuroscience, psychology, computer engineering, control theory/cybernetics and linguistics. Moreover, AI can be categorized generally, such as in learning and perception or by performing a specific task, like playing chess, driving cars on public streets, diagnosing diseases, proving mathematical theorems, composing music, and translating languages. Therefore, many different definitions and typologies of what constitutes an AI exist today (Russel & Norvig, 2010). Nevertheless, in the most cases AI is referred to as a non-human intelligence or a computer system that is capable of performing human mental tasks such as perception, conversation, adaptive learning from experience, strategizing, reasoning about others and decision making (Department of Defense, Defense Science Board, 2016).

Additionally, Nils J. Nilsson (2010) has established in his book *"The Quest for Artificial Intelligence: A History of Ideas and Achievements"* the following definition: *"Artificial Intelligence is that activity devoted to making machines intelligent, and intelligence is that quality that enables an entity to function appropriately and with foresight in its environment."* (Nilsson, 2010, p. 13). In other words, artificial intelligence is the ability of a machine or software to autonomously compose, select and decide on appropriate actions based on knowledge acquired from an uncertain environment, whereby applied actions ought to maximize the probability of success (Naval Postgraduate School, 2015).

However, the first person to define the term artificial intelligence as *"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."* was John McCarthy (1995) (McCarthy, http://www-formal.stanford.edu, 2007, p. 2).

2.2 The history of AI

2.2.1 The First AI Spring 1956 - 1975

In 1956 ten scientists sharing a common interest in neural nets, automation and information theory met at Dartmouth College. There during a six-week workshop Professor J. McCarthy at Stanford University, Professor M. L. Minsky at the Massachusetts Institute of Technology, Professor H. Simon and Professor A. Newell at Carnegie Mellon University, all awarded with the Turing Award, along with C.E. Shannon "the father of information theory" and other fellow colleagues, established the concept of "artificial intelligence" (Pan, Heading toward Artificial Intelligence 2.0, 2016). McCarthy, Minsky, Simon and Shannon (1955) targeted in their study proposal which they submitted to the Rockefeller Foundation, the following proceedings: *"The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves."* (McCarthy, Minsky, Rochester, & Shannon, 1955, p. 2). Although, all major thinkers in the field of artificial intelligence at that time participated in the workshop, it did not lead to any significant scientific breakthroughs. Nevertheless, this shaped the starting point of that field for decades to come (Bostrom, 2014).

Though, many of the technical ideas which would be utilized for artificial intelligence existed much earlier. Thomas Bayes (1763) provided a reasoning framework for describing probabilities of events based on prior knowledge or relations of conditions (Bayes, 1763). Furthermore, Charles Babbage (1835) planned the first calculating engines which were designed to tabulate polynomial functions (Babbage, 2010). Moreover, George Boole (1853) showed that logical reasoning could be performed in a systematic way (Boole, 1853). Progress in statistics during the twentieth century showed that reasoning or conclusions could be drawn from data. However, Alan Turing (1950) wrote in his essay *Computing Machinery and Intelligence* and anticipated that software would be a limiting factor in the development of artificial intelligence. Therefore, he proposed machine learning as a possibility to program an AI (Turing, 1950).

During the 1960s and 70s scientific disciplines in the area of heuristic search, used to construct proofs to validate mathematical theorems and computer vision or character recognition, which set the basis for face recognition used today, emerged. Moreover, first attempts on natural language processing, mobile robotics and machine learning were made (Stanford University, 2016). Arthur Samuel (1959) investigated in *Some Studies in Machine Learning Using the Game of Checkers* two machine-learning procedures. He verified that a computer program can learn to play a better game of checkers compared to its creator. This is done through improvements it made while playing against itself. Remarkably, the machine did so in 8-10 hours of machine-playing. Nevertheless, the rules of the game as well as a sense of direction and a list of parameters had to be given. These parameters could have been related to the game or not and their correct signs and relative weights didn't have to be specified. The performance of the machine was sufficient to greatly outperform average players and it was stated that the devised learning schemes could be applied to real-life problems (Samuel, 1959). Moreover, one year before F. Rosenblatt (1958) published *The Perceptron: A probabilistic model for information storage and organization in the brain* where he developed the concept of the Rosenblatt perceptron, an early form of a neural network compound of a single neuron (Rosenblatt, 1958). This model became the basis for today's artificial neuronal networks and was until the mid-eighties state-of-the-art. Although, promising advances were made in different disciplines till then, the field could still not produce significant practical successes by the 80s (Stanford University, 2016).

2.2.2 The First AI Winter 1974 - 1980

After the rapid progress of the past years, the capabilities of AI hit its first limitations. On the one hand, this was due to the problem of handling combinatorial explosion of possibilities. On the other hand, shifting away to real-world problems would have required high processing power which was out of reach at that time. Moreover, technical limitations regarding processor speed and memory exacerbated further proceedings (Bostrom, 2014). In addition, several other setbacks in the field of natural language processing and neural networks occurred which led to significant funding cutbacks (Spiegeleire, Maas, & Sweijs, 2017). Moreover, Marvin L. Minsky and Seymour A. Papert (1969) were finding more drawbacks due to performance limitations of F. Rosenblatt's early perceptrons (Minsky & Papert, 1969).

2.2.3 The Second AI Spring 1980 - 1987

However, after these major setbacks in the 70s, expert systems became popular. These were hard-coded and rule-based programs which are able to imitate decision-making processes of a human-expert. Thus, expert systems had been used as supporting tools for managers. Moreover, improvements in natural and computer sciences led to the development of backpropagation algorithms which enabled neural networks to learn a wider range of functions. Thereupon, after almost a ten-year standstill, scientists were able to continue on the development of neural networks applications (Spiegeleire, Maas, & Sweijs, 2017).

2.2.4 Another AI Winter 1987 – 1993

Subsequent to the second AI spring another AI winter occurred. According to Lenat and Guha (as cited in De Spiegeleire, 2017) this was largely caused by the flooding of desktop computers by Apple and IBM, which outperformed hardware based on artificial intelligence. Consequently, AI hardware companies collapsed and expert systems had been considered as generally ineffective and unreliable as they tended to crash if they were fed with unusual inputs (Lenat and Guha, as cited in De Spiegeleire, 2017, pp. 33-34).

2.2.5 The AI Upswing 1993 - 2005

Generally, it can be said that the lessons learned from the 80s until the end of the 90s, caused by AI incompatibility with variations in the information environment, were that knowledge should be gained from the environment automatically instead of learning vast amounts of knowledge from human experts (Pan, Heading toward Artificial Intelligence 2.0, 2016). Moreover, research focused more on solving domain specific problems and in 1997 Garry Kasparov, the world's chess champion at that time, was defeated by Deep Blue (McCorduck, 2004). Thus, technological advancements both in hardware and software as well as the increasing world-wide connectivity made the task of building systems driven by real-world data much more feasible. Further, more reliable and cheaper sensor technology reduced the complexity to build robots. Additionally, the spread of the internet, its capacity for gathering, store and process large amounts of data, enabled statistical techniques to derive patterns, predictions and solutions from data (Stanford University, 2016).

2.3 Modern AI

The starting point for neural networks was in 2006. During that year a paper published by Geoffrey E. Hinton and Vinod Nair (2006) showed how to train a deep neuronal network for recognizing handwritten digits. The neuronal network was capable of achieving state-of-the-art precision of greater than 98 percent (Hinton & Nair, Inferring Motor Programs from Images of Handwritten Digits, 2006). This was the beginning of "Deep Learning" and at that time no other machine learning technique could have achieved similar results. More than 10 years later machine learning can be found in many high-tech products. Applications range from ranking web search results, speech recognition, recommender systems to autonomous driving and object recognition (Géron, 2017). Consequently, all these developments have allowed AI to evolve as an in-depth influence on the lives we have today (Stanford University, 2016). Since then, AI has expanded to fields of research including mechanical theorem proving, machine translation, expert systems, game theory, pattern recognition, machine learning, robotics and intelligent control (Pan, Heading toward Artificial Intelligence 2.0, 2016). Additionally, intelligent algorithms profited also from supporting industry related developments, such as cheap availability of computing power, faster networks, connectivity and cloud infrastructures as well as open-source availability of very large datasets coming from various sources like social media (Russel & Norvig, 2010).

Recap

Chapter 2 showed how the general term "Artificial Intelligence" evolved from different scientific disciplines like mathematics, statistics, engineering, psychology and computer science over time. Moreover, it was stated that many concepts existed much earlier and were fostering research. However, due to unsatisfying performance of hardware and software and the lack of data, machine learning algorithms could not be applied to solve practical problems. It was through the mass availability of the internet in the mid-nineties and the related generation of data as well as the further development in the field of F. Rosenblatt's perceptrons by Hinton's and Vinod's development of a deep neural network among other things that artificially intelligent programs were able to achieve state-of-the-art performance in current industries. Therefore, the next chapter introduces machine learning and gives a brief overview of existing deep learning concepts and methods. Thus, state-of-the art theoretical and practical concepts will be explained and summarized so that a basic understanding of artificial neural networks can be developed.

3. Machine Learning

The term "machine learning" (ML) refers to an algorithms ability to automatically detect meaningful patterns in data. They are also able to learn from experience. Moreover, ML is primarily suitable for tasks which require knowledge extraction from large and complex sets of data like astronomical, genomic or medical data, e-commerce, search engine and weather data. Because of the high complexity of patterns, human programming is unable to cover such a granular specification of how such tasks would have to be executed (Shwartz & David, 2014). In general, a machine learning algorithm collects vast amounts of data and automatically figures out a good behavior out of it (Grosse, CSC321: Intro to Neural Networks and Machine Learning, 2018).

Alan Turing (1950) described machine learning as a computer having the ability to adapt to new conditions and to determine and project patterns (Turing, 1950). Thus, insights can be derived to generate and train predictive models. These algorithms are essentially different compared to static, preset-rules or instructions using "manually" coded "if" and "else" decision rules to process data or to modify user input. Although, manually coded decision rules are feasible in applications where humans have a high understanding of the process to model, manually written rules have major disadvantages. This is because, the logical decision output is customized to a single domain and task which would require rewriting the whole code if the environment of the task would change even slightly. Furthermore, in order to design effective rules, it is necessary to have an understanding of how a human expert would have made the decision. In fact, where a manually coded approach would not be effective compared to machine learning is for example face recognition. This is because, pixels (the smallest unit of a digital image on a computer) are perceived differently by humans and computers. In fact, the sheer difference in representation makes it for computers much easier to come up with a constructive set of rules derived from different sets of variables or features required for identification of faces in digital images. However, a large collection of face images is required to train the learning algorithm in order to increase the likelihood of a successful face detection (Müller, 2016). Because, traditional programming is rigid or static, meaning that once a program has been written it stays unchanged. However, tasks change dynamically over time or from user to user. In contrast, machine learning algorithms adapt themselves to the environment (Shwartz & David, 2014).

To be exact, a machine learning algorithm is given a large dataset specifically related to a given task. Hence, this enables the algorithm to extract statistical patterns from which task related rules can be derived (Khumoyun, Cui, & Hanku, 2016, pp. 2-4). Nevertheless, the embedding of prior knowledge is crucial for the success of machine learning algorithms. Therefore, a central issue of the theory of machine learning deals with the development of tools for expressing domain knowledge and translating it into a learning bias, so that the effect of the bias on the success of learning can be measured. Although strong prior knowledge increases the learning of further examples, strong prior domain expertise reduces the flexibility of the learning (Shwartz & David, 2014). Moreover, if a system needs to adjust to a changing environment, machine learning is a suitable application. Further, ML algorithms behave autonomously which can contribute to privacy and fairness (Grosse, CSC321: Intro to Neural Networks and Machine Learning, 2018). However, Shalev-Schwartz and Ben-David (2014) stated in their book “Understanding Machine Learning” that integration of prior or existing knowledge biasing the learning algorithm is a driving factor for a successful application of machine learning. The process is referred to as the “No-Free-Lunch theorem” (Shwartz & David, 2014). What is more, a fundamental problem of machine learning is to discover patterns that generalize (Zhang, Lipton, Li, & Smola, 2019).

3.1 The Four Core Components

There are four core components machine learning comprises, independently from its application. First, data the machine can learn from. Second, a model which transforms the data. Third, a loss function that acts as a performance measure to quantify the quality of the underlying model. Fourth, an algorithm to set the parameters of the model, so that the loss function is minimized (Zhang, Lipton, Li, & Smola, 2019).

3.2 Data

Machine learning can’t be applied without having data or a collection of *examples* also referred to as data input points, instances or samples. Apart from this, a sample has to contain a suitable numerical representation. Further, an example or *numerical representation* comprises labeled data or inputs, or, in other words, a collection of *numerical attributes*, also called *features* or *covariates* (Zhang, Lipton, Li, & Smola, 2019). This is because the performance of machine learning algorithms relies heavily on the numerical representations of the given data. Moreover, there is a significant dependence on representations. Additionally, intelligent structuring and indexing of data makes search operations exponentially faster. What is more, the choice of features or covariates has a huge effect on the ML performance. Thus, the right adjustment of features contributes to solving tasks related to artificial intelligence (Goodfellow, Bengio, & Courville, 2016). This is why constant or fixed-length vectors, a sample having an equal number of features, can reduce complexity. The length of the vectors is known as *dimensionality*. Also, the *quantity of data* available to train the machine learning algorithm is crucial, because less pre-conceived assumptions have to be made (Zhang, Lipton, Li, & Smola, 2019). However, *data integrity* and *consistency* are important for learning and predictive performance. Also, increased attention has to be paid to samples where groups of people are *unrepresented* or discriminated in the training set. This can have severe negative impacts. For example if a skin cancer recognition system goes into production without ever seeing dark skin before. Moreover, data containing societal prejudices or past *biased decisions* could unintentionally reflect and automate historical injustices (Zhang, Lipton, Li, & Smola, 2019).

Indeed, machine learning can be used not only to discover the mapping from representation to output but also for discovering the representation itself, which is called *representation* or *feature learning*. Furthermore, self-learned representations perform better compared to hand-engineered or hand-labeled features. As a result, artificially intelligent systems can adapt more rapidly to new tasks and with much less human intervention. Depending on the complexity of the task, a feature learning algorithm can find good sets of features for simple tasks in minutes, or hours to months for more demanding tasks. In contrast, manually labeled data can take decades including huge human time and effort. In fact, the autoencoder is one example of a representation learning algorithm which utilizes an encoder and a decoder function. On the one hand, the encoder function converts the input data into a different representation, while the decoder function is reversing it into the original format. In addition, factors of variation explaining the data have to be partitioned. To be exact, Ian Goodfellow, Yoshua Bengio and Aaron Courville (2016) describe “factors” as separate sources of influence, unobserved objects which can affect observable sets. For example, factors of variations when analyzing speech records can be the age, sex, accent and spoken words of the speaker (Goodfellow, Bengio, & Courville, 2016).

For this reason, the unobserved factors of variations have an influence on the observed data, which represents a challenge for real-world AI applications. This is why *disentangling* and *discarding* of unwanted factors is needed. However, this is a very difficult undertaking, considering the case of speech recognition, when for example the accent of the speaker has to be removed. Because, in order to discard an accent, near human based understanding of the data is needed. Consequently, one has to think of a *tradeoff* between the effort needed for obtaining representations and solving the original task. This is also described as the central problem of representation learning. In order to overcome this, deep learning is used. To be exact, it solves this problem by introducing other and less complex representations

(Goodfellow, Bengio, & Courville, 2016). Although data is in many applications plentiful, computation time is the main bottleneck (Shwartz & David, 2014).

Thus, a successful application of machine learning is heavily dependent on whether meaningful datasets including the desired output pairs can be created or not. Moreover, collecting the data is a critical process and can be vastly different from use case to use case (Müller, 2016). Further, different models are used for processing sequences like strings of text or time series data and for fixed-length vector forms (Zhang, Lipton, Li, & Smola, 2019).

3.3 Datasets

In the table below, large databases containing data to pre-train neural networks are listed. These provide a very useful starting point, because once trained on already existing databases, the features learned are useful to solve new tasks (Goodfellow, Bengio, & Courville, 2016).

Name	Task
MNIST	Image Classification
ImageNet	Object Recognition
ILSVRC	Object Recognition
Microsoft Coco	Image Recognition, Segmentation, Captioning
Cityscapes	Autonomous driving dataset
PASCAL VOC	Object recognition dataset
KITTI	Autonomous driving dataset
NYUv2	Indoor RGB-D dataset
LSUN	Large-scale Scene Understanding challenge
VQA	Visual question answering
Madlibs	Visual Madlibs (question answering)
Flickr30K	Image captioning
MovieDescription	Automatic movie clip description
MPI Sintel Dataset	Optical flow dataset
BookCorpus	Corpus of books

Table 1: "Popular datasets" (University of Toronto, 2018)

Recap

To summarize, in this chapter it was explained why machine learning is needed as a substitute to traditional programming. Moreover, the four core components needed to make machine learning artificially intelligent were described. These comprise the data, the handling of the data, a learning model which transforms the data, a loss function and an algorithm to tune the hyperparameters. Furthermore, the autoencoder function as a way for representation learning was mentioned and why disentangling is needed. This is why, in the next chapter the focus will be on the model, namely neural networks and its functionality.

4. Deep Learning: A machine learning model for representation learning

Generally, deep learning is a subdiscipline of machine learning based on a neural network and it becomes deep as the number of its layers is increasing. Basically, it is inspired by the idea of learning from example. In contrast to traditional computation where arithmetic calculations and lists of explicit instructions are performed very fast, deep learning advances by evaluating examples and a set of instructions to solve a task on its own. Furthermore, when it makes a mistake, a deep neural network for example is able to modify the underlying model autonomously and without external input. While machine learning algorithms still need guidance from engineers if inaccurate predictions were made, a deep learning model can determine and adjust on its own if a prediction was inaccurate (Buduma, 2016). To be specific, a model is a computational process that ingests data of one type and outputs predictions of a different type (Zhang, Lipton, Li, & Smola, 2019). Goodfellow, Bengio & Corville (2016) stated that the elementary example of a deep learning model can be described as feedforward deep network or multilayer perceptron (Goodfellow, Bengio, & Courville, 2016). This is because neural networks can handle complex problems where data is extremely high dimensional, but the relationships to be measured are highly nonlinear (Buduma, 2016).

Thus, deep learning (DL) enables mapping of input to output values via utilizing a neural network. Moreover, it can produce additional layers of more abstract features in addition to the initial input it receives. In contrast to traditional machine learning techniques, deep learning simplifies complicated mapping into a series of interlaced simplistic mappings, each represented by different layers in the neural network. In other words, it uses mathematical functions under the principle of a neural network composed to overcome the problem of representation learning (Goodfellow, Bengio, & Courville, 2016). Therefore, deep learning handles *varying-length* data more effectively compared to traditional methods (Zhang, Lipton, Li, & Smola, 2019). The following subchapter will therefore explain the concept of a neuronal network.

4.1 The biological neuron

Almost every area of artificial intelligence attempt to mimic cognitive processes like logic or probabilistic reasoning (Ertel, 2017). Thus, artificial neural networks are biologically-inspired neuronal systems (Shwartz & David, 2014). In biology, a neuron or nucleus is considered as the basic unit of processing in the brain (Grosse, CSC321: Intro to Neural Networks and Machine Learning, 2018). Moreover, it is optimized to accept information from other neurons. Further, the results of the processed information are sent to other cells. Inputs are received through dendrites which branch off from the neuron in a tree-like structure. The connection of the dendrites is strengthening or weakening depending on how often they are used. As a result, the strength of each connection determines the contribution of the input affecting the output. Thereupon, all incoming weighted inputs are summed up in the cell body. Subsequently, the information is transformed into a new signal. Then, the electrical signal is transferred along the axon to other neurons (Buduma, 2016).

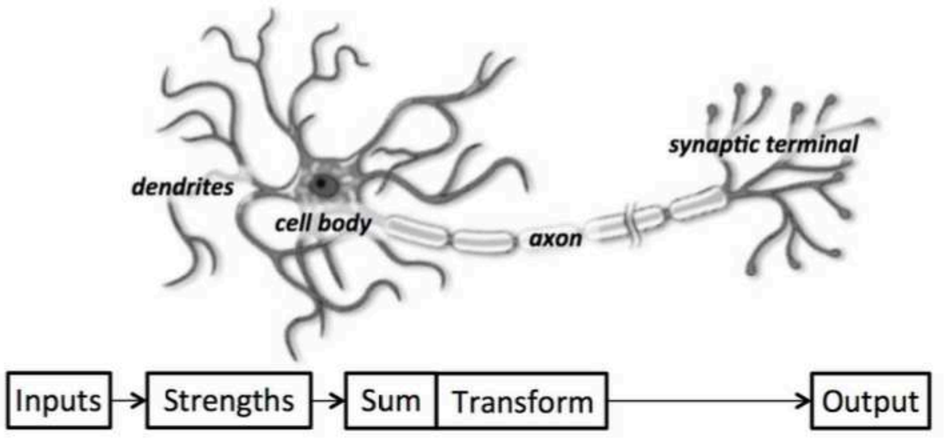


Figure 1: "A functional description of a biological neuron's structure." (Buduma, 2016)

4.2 The artificial neuron

For artificial neural networks the process described above is much more simplified. In this case, a linear neuron has a set of connections to receive weighted inputs from other neurons plus a bias, also called a sum of inputs. If the sum of inputs is between a given accepted range, they will be passed on to the next layer of neurons. Thus, affecting the calculations of the next layer or the predicted outputs of the output layer. This process is also called forward propagation as illustrated in figure 2. Whether a sum will be forwarded or put back is checked by an activation function (Grosse, CSC321: Intro to Neural Networks and Machine Learning, 2018).

$$z = \sum_{j} w_j x_j + b.$$

Where the inputs xj are multiplied by weights w, determining the strength of the connections between input and output, plus a bias b. This bias is moving the function so that it is not restricted to the origin. Additionally, the value b is a determinant for a neuron's activation when there is a lack of inputs (Grosse, CSC321: Intro to Neural Networks and Machine Learning, 2018).

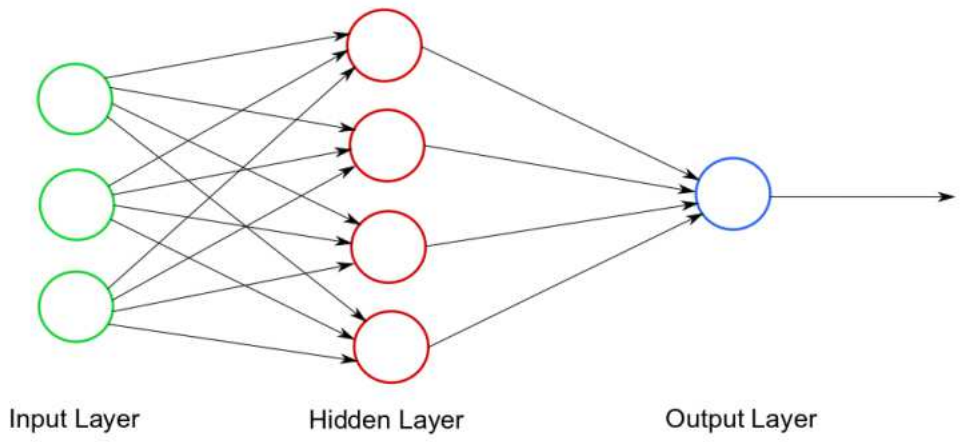


Figure 2: "A simple neural network"; (Mayo, 2017)

The figure above shows a simple neural network where each neuron has a connection to various other neurons, allowing signals to flow in one direction from left (input layer) to right (output layers) including through any number of hidden layers in between (Mayo, 2017).

4.3 Feed-forward networks

Feedforward neural networks or linear multilayer perceptron's are the fundamental deep learning models. To be specific, their goal is to estimate a function (model class) which validates numerical mapping parameters in order to map an input vector x into an output y. The goal therefore is to predict an output (target value) which is equal to the actual output value. For this, the algorithm has to learn the values of the hyperparameters which lead to the best function estimation. The name feedforward network originates from the lack of feedback connections in which outputs are fed again into the model. Typically, the first layer is called input and the last one is called output layer. However, in between it can have several hidden layers. If there is more than one hidden layer, the network is considered as deep feed-forward network. Further, each layer receives a new representation of the input. Because, the manner of the layers in between is not directly specified by the training data, the algorithm must decide on how to use the respective layers in order to estimate the best output. In contrast, in a recurrent network the information can flow in cycles. Additionally, these networks can remember information for a long time but are more difficult to train (Goodfellow, Bengio, & Courville, 2016). In training modern neural networks, a technique called backpropagation is used. Thus, produced error signals are sent backwards through incoming connections. (Grosse, CSC321: Intro to Neural Networks and Machine Learning, 2018).

4.4 Activation function

As the neuron receives all values from connected neurons multiplied by their weights plus a bias, the value is checked by an activation function. It allows a neuron, depending on the values it receives, to be activated or not and to pass the signal to other neurons. Its purpose is to transform or to smooth the Z value which can range from -inf to +inf, because once multiplied by weights and summed, values can quickly get beyond their original scale. Therefore, an activation function converts its input to a value between 0 and 1. As a result, better adjustment for the distance of the error can be made. There are several different activation functions. For instance, the equation below shows a sigmoid activation function (Bishop, 2006).

$$y = \frac{1}{1 + e^{-z}}$$

In the figure below, it can be seen that only hidden layers and output layers possess activation functions, whereas input layer neurons do not. Moreover, to be useful these functions have to be nonlinear and continuously differentiable. This is because nonlinearity allows the neural network to be a general approximation, while a continuously differentiable function allows a more efficient back propagation of errors throughout the network (Mayo, 2017).

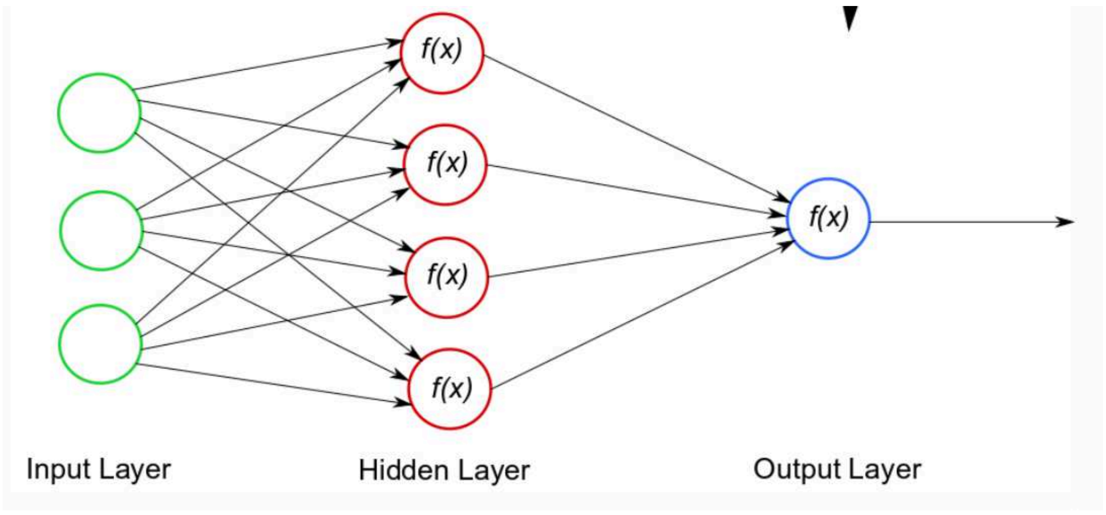


Figure 3: "Activation function of neural network", (Mayo, 2017)

4.5 Gradient descent

Because a neuron can have connections to multiple other neurons forming a network, weights for each layer need to be estimated. This is done through stochastic gradient descent (SGD) optimization algorithms. In general, the error is minimized resulting from the difference of the estimated output and the actual output. Thus, for each weight a gradient descent is calculated. Moreover, SGD is an optimization algorithm, based on a convex function which iteratively adjust its parameters to minimize an underlying function to its local minimum (Shwartz & David, 2014).

4.6 Convolutional Neural Networks

Convolutional Neural Network (CNN's) are networks developed for having images as inputs. In contrast of looking for similarities between input features, convolutional neural networks are looking into patterns in features which are near to each other. However, a multi-layer perceptron receiving images with three color channels would need a huge amount of weights that must be trained. Additionally, these networks are not translation invariant to receiving shifted versions of images. Convolutional neural networks try all x- and y-positions since objects in images can appear at any x- and y-positions. To be exact, a convolutional neural network looks at specific smaller parts of images at the time determining whether an object is there or not (Bishop, 2006).

4.7 Confusion Matrix

A confusion matrix can be used as a way to measure how well an algorithm performs. This is explained in more detail in chapter 5 (Fawcett, 2005).

4.8 Other factors

In addition, computational resources are a driving force for a successful application of neural networks (Goodfellow, Bengio, & Courville, 2016). This is because larger models can be run, meaning that a great collection of neurons is more useful compared to small collections. Thus, larger networks achieve higher accuracy on more complex tasks. Also, faster CPUs, GPUs for AI inference processing, faster network connectivity and better software infrastructure play an important role for the application of neural networks (Goodfellow, Bengio, & Courville, 2016).

4.9 Frameworks

The table below lists powerful neural net frameworks. These frameworks allow to implement easy-to-use and quickly to implement advanced neural net models (Grosse, CSC321: Intro to Neural Networks and Machine Learning, 2018).

Name	Topic
Caffee	Deep learning for image classification
TensorFlow	Open source software library for machine intelligence (deep learning)
Theano	Deep learning library
Mxnet Gluon	Deep learning library
PyTorch	Scientific computing framework with wide support for machine learning algorithms
LIBSVM	A library for support vector machines (matlab, python)
Scikit	Machine learning in Python
Autograd	A lightweight automatic differentiation library

Table 2: "Powerful neural net frameworks." (Grosse, CSC321: Intro to Neural Networks and Machine Learning, 2018)

4.10 Types of learning

4.10.1 Supervised Learning

In general, there are several types of a neural networks ability to learn. This is dependent on the kind of data collected. To be exact, it can be distinguished between three different types of machine learning (Grosse, CSC321: Intro to Neural Networks and Machine Learning, 2018). An algorithm which is able to generalize from individual known examples and as a result can automate decision-making processes is called supervised learning. Hence, it is applicable if the user can provide the algorithm with large input and desired output data pairs (Müller, 2016). For example, if a neural net is trained to distinguish between two different objects like cats and dogs, images containing these objects have to be collected and labeled each as cat or dog (Grosse, CSC321: Intro to Neural Networks and Machine Learning, 2018). In other words, a set of training cases consisting input vectors and desired output vectors is provided (Hinton, Supervised learning in multilayer neural networks, 1999). To be exact, the learner receives a training set containing labeled data like true/ not-true. Based on the experience acquired, the machine learning algorithm can figure out a rule to label or predict the output of newly arriving input data. This newly arriving and therefore unlabeled input data is termed test data. Therefore, it can be said that the input represents training data which constitutes experience while the output represents expertise (Shwartz & David, 2014). In such cases, the algorithm not only learns by finding patterns in input/output pairs but also by being supervised with correct results if incorrect predictions were made. This is why incorrect data labels can affect the accuracy of the output and decrease the effectiveness of the supervised learning model. Because the algorithm learns a function, given a sample of input and output pairs, that estimates the best relationship between input and output pairs observed in the sample (Müller, 2016). Generally, the approximation or prediction of the outputs is made either as classification or regression (Khan, Jan, & Farman, 2019).

The most used type of learning in practice is supervised. This is partly because, a great number of tasks can be described as a probability problem. In detail, the probability of an unknown target can be estimated given some available evidence or labeled examples. However, depending on the type, size and the number of labels and instances, supervised machine learning can take many forms. As a result, vast modeling decisions are required (Zhang, Lipton, Li, & Smola, 2019).

To summarize, a task has to be defined first, where the algorithms main purpose or goal is to train itself based on the input it receives to predict an actual output or target (Grosse, CSC321: Intro to Neural Networks and Machine Learning, 2018). First, input data points (x) or examples have to be selected randomly. Second, the labels (y) or outputs have to be acquired. Subsequently, the input data points are mapped to the corresponding labels. In this way, a training set emerges which is fed into a supervised machine learning algorithm. At this point the supervised learning algorithm, a function, receives the training set and outputs another function called the learned model. Ideally, the supervised machine learning algorithm can then be given a new unseen input or example whereas the corresponding labels can be predicted. An abstraction of this process is shown in figure 3 below (Zhang, Lipton, Li, & Smola, 2019).

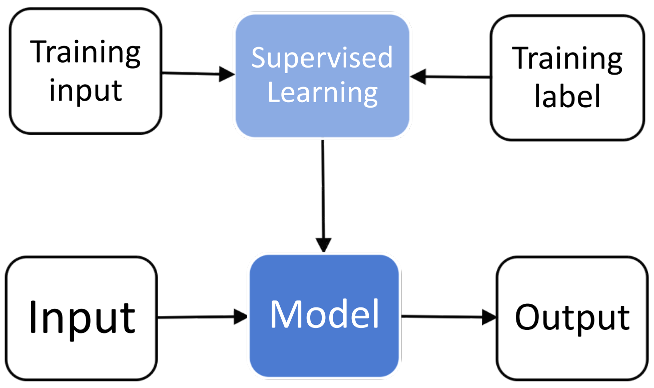


Figure 4: "Supervised Learning Process" (Zhang, Lipton, Li, & Smola, 2019)

4.10.2 Evaluation

However, good predictions of outputs on the training data do not imply that the model generalizes well. This is why, the available data has to be split into two sets. On the one hand, the partition consists of the training set, where the data is used for fitting the model parameters. On the other hand, unseen data or a test set is used for evaluating the models or algorithm's performance (Grosse, CSC321: Intro to Neural Networks and Machine Learning, 2018). Nevertheless, the performance of both the training and test set can deviate significantly from each other. Hence, the failure of generalizing learning is described as overfitting problem. In general, the underlying functions objective is to minimize the error between the predicted and the true instances. Because a lower error is better, these objective functions are called loss functions or cost functions. Moreover, the definition of a loss function is dependent on the model's parameters and dataset. Examples for evaluation functions are accuracy/error rate, precision and recall, squared error, likelihood, posterior probability, information gain, K-L divergence, cost/utility function and margin (Zhang, Lipton, Li, & Smola, 2019).

4.10.3 Overfitting

The term overfitting describes a situation when the model is more closely fit to the training data instead to the test data or the underlying distribution and therefore fails to generalize. To be precise, the training error is the error of a model resulting from the training set. In addition, the generalization error is the expected error produced by a model when it fails to make a prediction from an infinite stream of unseen data points which have the same underlying data distribution as the original sample. Since the generalization error cannot be calculated exactly, the performance of the model has to be estimated against a test set. In fact, the test set is a random selection of inputs from the training set (Zhang, Lipton, Li, & Smola, 2019).

4.10.4 Optimization

In order to find the right parameters for minimizing the loss function, optimization functions are used. One of the most used for neural networks is gradient descent. The model's training efficiency is directly affected by the performance of the optimization algorithm. Although, optimization algorithms are used to reduce the training error, deep learning aims to reduce the generalization error. This is why attention has to be paid to both aspects. According to Zhang, Lipton, Li & Smola (2019) factors which influence generalization are the number of tunable parameters or degrees of freedom. If they are large the model tends to be more prone to overfitting. Moreover, if there is a high variation in the range of values the weights can take, the model tends not to generalize. Moreover, the number of training examples influences whether a model tends to overfit or not (Zhang, Lipton, Li, & Smola, 2019).

As stated by Goodfellow, Bengio & Courville (2016) in their book, supervised algorithms achieve an acceptable performance with approximately 5,000 labeled examples per category. Moreover, if trained with datasets containing more than 10 million labeled examples human performance can be matched or exceeded (Goodfellow, Bengio, & Courville, 2016).

4.10.5 Validation

In addition to splitting the dataset into a training and test set, also a validation set is needed. This is because, the hyper-parameters of the model need to be adjusted manually. This is also referred to as tuning of the hyper-parameters. The term hyperparameter consists the numbers of samples (batch size) and the learning rate. Consequently, a low learning rate, requires the model to take more iterations or epochs, passes through the data, to get to a better solution. Hence, correct adjustment of the mentioned factors can increase the accuracy of the model. Usually, this is done through repeated trial and error until the suited parameters are found. Ideally, these adjustments can be learned as parts of model training (Zhang, Lipton, Li, & Smola, 2019).

4.10.6 Unbalanced data

Predicting classes which are represented extremely low in the dataset can be very difficult. This is also referred to as the problem of unbalanced classes in the dataset. Hence, either more data has to be collected to even the classes out, or examples from larger classes have to be deleted. Therefore, unbalanced classes lead to suboptimal results, as the model never gets enough insides at the class in question. Thus, the risk of not having sufficient representations increases the problem of creating a validation or test sample. Currently, there are few possibilities to approach the problem of unbalanced data. First, the class which has enough information can be deleted randomly, so that a sufficient relationship between two classes is maintained in the dataset. This approach is called undersampling. However, there is a high possibility that important information about the predictive class is erased as well. Second, copies can be created for the underrepresented class to increase the number of observations, also called oversampling. Though, this can lead to overfitting to the training data. Third, a Synthetic Minority Over-sampling (SMOTE) can be applied. To be specific, observations of unbalanced classes which are similar to the existing one are synthetically created using k-nearest neighbors classification (Ali, Shamsuddin, & Ralescu, 2015).

4.10.7 Types of problems supervised learning aims to solve

There are two kinds of prediction problems in machine learning. These can be distinguished between regression and classification problems. On the one hand, predictions of continuous numerical values are considered as regression problems. On the other hand, predictions of discrete categories are associated to classification problems (Zhang, Lipton, Li, & Smola, 2019). The data for the figure below comes from a study by Stamey et al. (1989) that investigated the correlation between the level of PSA (prostate specific antigen) and other clinical measurements (Stamey, et al., 1989). Although the scatterplot matrix of the variables shows that two of the predictors “svi” and “gleason” are categorical and therefore of discrete nature, the other predictors are continuous (Hastie, Tibshirani, & Friedman, 2017).

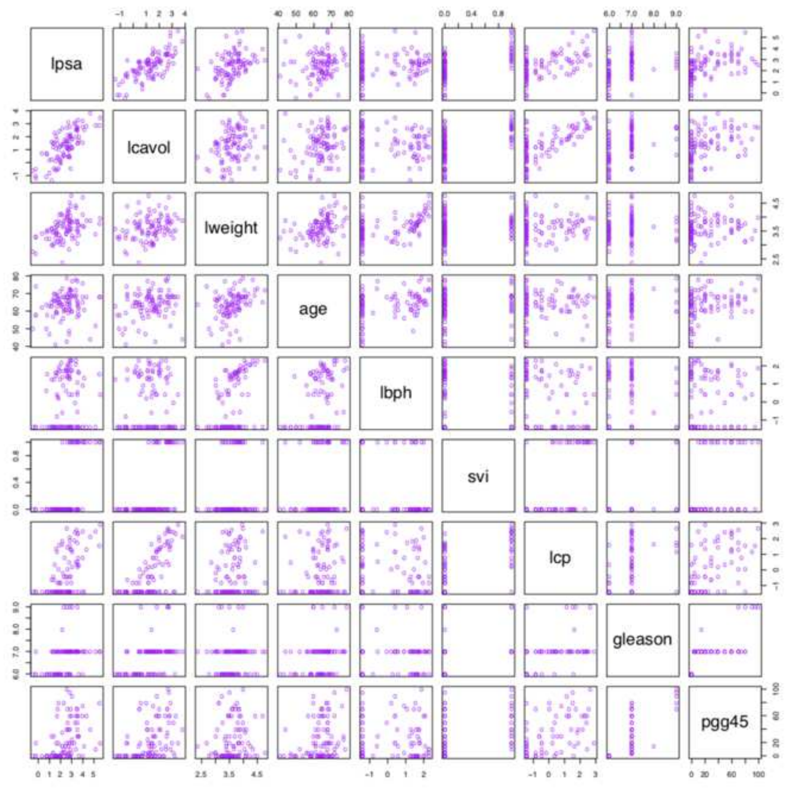


Figure 5: "Scatterplot matrix of the prostate cancer data." (Hastie, Tibshirani, & Friedman, 2017)

4.10.8 Regression Problems

Linear regression models are useful for predicting continuous values like the price at which a car will be sold, or the number of wins of a basketball team, or the number of days a delivery will take, or a patient will remain hospitalized before being discharged. Generally, the prediction is expressed as a linear algebra notation. However, in practice it is highly unlikely that the prediction lines up exactly as a linear function of the features. Therefore, errors have to be expected which need to be minimized. This is why a noise term has to be incorporated. Hence, the goal of linear regression is to find the weight vector and bias term (offset or intercept) that maps each feature to a convergence of the predictor of the corresponding label (Zhang, Lipton, Li, & Smola, 2019).

4.10.9 Classification Problems

In deep learning, classification is used for describing two kinds of problems. On the one hand, assessing hard assignments which is true when a datapoint belongs to only one class. On the other hand, when a datapoint does not belong specifically to one class but can have different degrees of membership in several classes. In other words, the probability that each category applies. If there are more than two possible classes to predict, the classification problem is called multi-label class classification. Furthermore, while for regression problems the loss function is minimized, for classification problems it is cross-entropy. Furthermore, classification problems usually do not have a natural order among the classes. In order to overcome the problem of how to represent categorical data, one hot encoding is used. Therefore, the labels are represented as a multi-dimensional vector. For this, the component corresponding to its category has to be set to one for a given instance while all other components are set to zero. Hence, one-hot encoding can be treated as a probability distribution for each input. Thereby, cross-entropy compares the labels true probability distribution with the model's prediction of the class. Consequently, the closer the cross-entropy approaches zero, the better the prediction by the model (Zhang, Lipton, Li, & Smola, 2019).

So, to predict multiple classes, a model with several outputs per category is needed. In other words, each category classification output needs its own linear function. This means, assumed there are three output categories and four features, 12 scalars would be needed to map the weights and three scalars to represent the biases. If every input can be assigned to exactly one class, the outputs need to be converted into discrete predictions. This is done through treating the output values as the relative confidence levels that an item belongs to a specific category (Zhang, Lipton, Li, & Smola, 2019). The table below shows the softmax activation, so that each sample (row) sums up to 1. The first sample shown in table 3 has a 29.45% probability to belong to class 0.

Class	Sample 1	Sample 2	Sample 3
0	0.29450637	0.34216758	0.36332605
1	0.21290077	0.32728332	0.45981591

Table 3: "Samples and softmax activation"; (Raschka, 2016)

Nevertheless, taking the confidence levels directly from the output layer can lead not only to misinterpretations but also to the model's inability to learn. Especially, when it comes to choosing the label with the largest output value as valid prediction (Zhang, Lipton, Li, & Smola, 2019). To turn these probabilities back into class labels the argmax-index position of each row can be taken, which outputs the values illustrated in the table 4. Consequently, it can be seen that although the first sample has a 29.45% probability to belong to class 0 the algorithm predicted class 2 which can be seen in the table below (Raschka, 2016).

Predicted Class label vs actual label	
Predicted Class Label	2
Actual Class Label	0

Table 4: "Predicte vs actual class labels"; (Raschka, 2016)

However, to minimize the error a cost function has to be defined. Otherwise, the argmax can match the label, which means that there is no error and the model can't be trained which is equal to cross-entropy of zero. Furthermore, if the argmax is unequal to the label, gradient descent-based optimization can't be applied. Consequently, in order to encourage the model to use computed probabilities instead of taking the values from the output layer directly, *softmax logistic regression* has to be utilized. That makes sure that on unseen data the probabilities are nonnegative and summed up to 1 (Zhang, Lipton, Li, & Smola, 2019).

4.10.10 Reinforcement Learning

If there are no examples of the desired behavior but a method to score whether the produced behavior was good or not, referred to as *reward signal*, *reinforcement learning* can be applied (Zhang, Lipton, Li, & Smola, 2019). In contrast to supervised learning the learning algorithm is not given optimal output examples. Instead, the algorithms have to discover suitable actions which lead to the most reward. This is done through trial and error interactions with the environment. Hence, the goal of the learning algorithm is to receive a maximum numerical reward signal. If it managed to find suitable actions for a given situation, a reward is given afterwards. Often, the applied action not only affects the immediate reward, but also the next steps and through that all successive reward signals. A general characteristic of reinforcement learning is to find the right balance between exploration and exploitation. In other words, reinforcement learning is a trade-off between the systems use of new actions and actions that are known to yield a high reward. Although, a desired outcome achieved consists of a sum of successful and unsuccessful actions, the reward must be attributed appropriately to all of the actions that led to it. Sutton & Barto (2018) in their book described this as a *credit assignment* problem (Sutton & Barto, 2018). The figure below shows that the agent interacts with an environment over a series of time steps. At each of the steps the agent receives an observation from the environment and has to decide on an action. This action then is transmitted back to the environment. If the action was appropriate, the agent receives a numerical reward signal. Subsequently, an additional observation is received by the agent on which a new action has to be chosen. Zhang et al. (2019) described this process as *policy*. Consequently, the goal of the reinforcement algorithm is to develop a good policy (Zhang, Lipton, Li, & Smola, 2019).

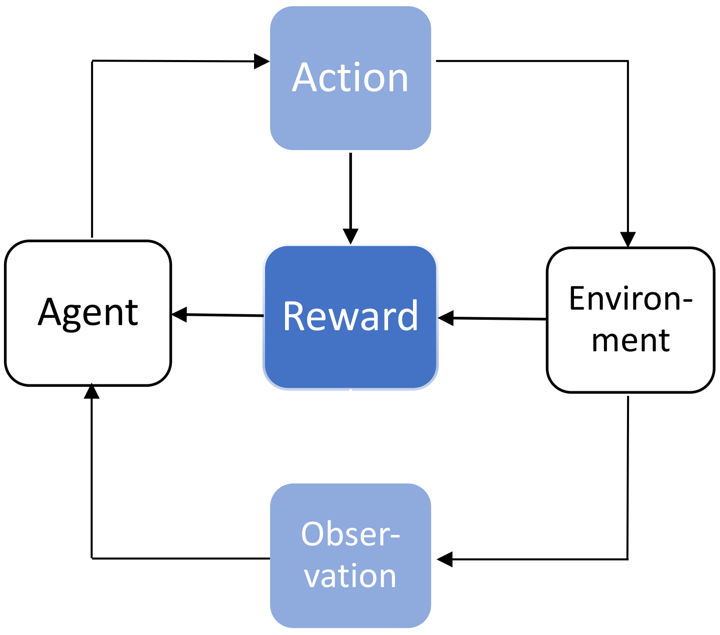


Figure 6: "Reinforcement Learning Process", (Zhang, Lipton, Li, & Smola, 2019)

4.10.11 Unsupervised Learning

Working successfully with vast quantities of unlabeled examples or semi-supervised data is an important research area (Goodfellow, Bengio, & Courville, 2016). Therefore, in unsupervised learning, neither examples of behaviors nor rewarding signals are available, only data where hidden patterns want to be derived (Grosse, CSC321: Intro to Neural Networks and Machine Learning, 2018). In this case, no such distinction between test and training data exists. Therefore, the learner receives large unlabeled data sets and has to come up with a compressed version or summary of the data. Thus, a typical task of an unsupervised learning algorithm is clustering data sets into subsets of similar objects (Shwartz & David, 2014). In other words, the goal of unsupervised learning is to learn a probability distribution, which matches an unlabeled dataset as close as possible. This is referred to as distribution modeling and unlabeled dataset can be a collection of pictures or sentences. Moreover, unsupervised learning can draw high-level explanations yielding to an unbiased insight into the structure of the respective dataset compared to supervised learning (Grosse, CSC321: Intro to Neural Networks and Machine Learning, 2018). To be specific, unsupervised learning learns to draw examples or denoise data from a distribution. Further, it finds a representation of x which describes x as exact as possible by collecting and preserving as much information about it as possible. Additionally, the algorithms aim is to keep the representation simpler or easier to access than x itself. Otherwise, a penalty is executed. In practice, the three most common representations used are lower-dimensional representations, sparse representations and independent representations. As stated by Goodfellow, Bengio, & Courville (2016) the concept of representation learning and the related algorithms is one of the central themes of deep learning (Goodfellow, Bengio, & Courville, 2016).

Recap

To summarize, in chapter 4 the basic concepts and methods used for artificial neural networks were explained. Beginning with the structure of a neural network and its layers, it was also described how input neurons send their calculated inputs to neurons of the hidden layers. Based on the value received an activation function determines whether a neuron should fire and distribute the signals to other neurons or not, which leads to the prediction of a label. Additionally, using gradient descent as a way to optimize the error between the predicted and the actual label was explained. Moreover, types of learning including supervised, unsupervised and reinforcement learning were described. It was also explained how models can be evaluated as well as validated, what overfitting is, how to train and optimize a model. Furthermore, the differences between feedforward-, recurrent- and convolutional neural networks were presented. Finally, problems which neural networks aim to solve were described.

5. Qualitative analysis of artificial neural networks in the manufacturing industry

In this chapter practical use cases of deep neural network applications in the context of different industrial settings and their results are analyzed. Consequently, the gathered qualitative data will not only unveil application patterns, but will help to derive a general framework for the implementation of neural networks in the manufacturing industry. Afterwards, the derived framework will then be used to develop a deep neural network and its results as well as the model accuracy will be validated quantitatively.

5.1 Image-based manufacturing analytics: Improving the accuracy of an industrial pellet classification system using deep neural networks

The first use case deals with image-based production classification for predicting pellet shapes. This is considered a supervised learning problem. Moreover, the inputs are given in form of an image and a classifier is developed that predicts a class label for each picture from a set of possible class labels. Rendall et al. (2018) trained a deep neural net capable of predicting the shape of pellets in an industrial setting (Rendall, et al., 2018).

5.1.1 Business Case

Shapes of pellets are an important indicator of quality in the pellet manufacturing industry. Hence, deviations from round shapes, which are considered as a sign of good quality, can negatively impact customer satisfaction, pellet storage as well as transportation and can cause cross-contamination plus other undesirable outcomes (Rendall, et al., 2018).

5.1.2 Dataset and collection

All the 5923 images used as input by Rendall et al. (2018) were RGB images with a resolution of 96 x 96 pixels. Thus, having three-color channels red, green and blue. Moreover, a high-speed camera took individual pictures of the pellets. Therefore, the inputs which the network receives are pixels from pellet images (Rendall, et al., 2018).

5.1.3 Task considered

A supervised learning task requires the existence of labeled data. For that reason, Rendall et al. (2018) let process experts manually label 5923 pellet images. In this way, they identified two types of labels per image. The first one was dedicated to the shape of the pellet, where a “good pellet” was characterized by having a round shape, and pellets with other shapes being considered as a “bad pellet”, having a different chemical characteristic. The second label type was associated with the existence of tails. To be specific, two types of classification tasks were defined separately, namely pellet shape classification and tails detection. This was due to the fact that, the number of samples were limited and the lack of data prevented the development of an individual classifier able to simultaneously predict the quality and the existence of tails (Rendall, et al., 2018).

5.1.4 Methods used

For the image classification a deep convolutional network was used (Rendall, et al., 2018). Thereby, max-pooling layers were used. Because of this, the model is invariant to small local changes in the input transformations (LeCun, Haffner, Bottou, & Bengio, 2001). The main purpose of this approach is to determine, whether specific features occur in certain regions, rather than to find their exact location which is useful when solving image classification problems (Rendall, et al., 2018). Furthermore, the model allows for parameters to be shared while the total number of weights is restricted. As a result, the hyperparameters can be adjusted more easily. This is why only local features of sub-regions of the whole image are inferred. Therefore, these extracted features are then merged into subsequent phases of processing which enables the convolutional network to derive higher-order features. Consequently, this leads to insights and information about the image as a whole (Bishop, 2006).

The figure below represents a part of a convolutional neural network and its layers and illustrates this process. These layers have entities which are organized into tiers, called feature maps. Further, these entities take inputs only from small regions of the image. At the same time, all entities are obliged to share the same weight values (Bishop, 2006).

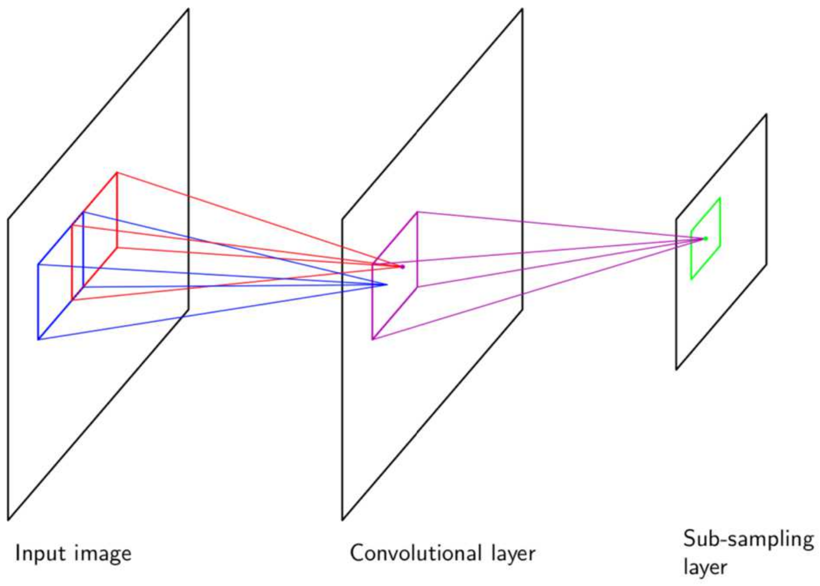


Figure 7: "A part of a convolutional network", (Bishop, 2006)

As activation function, Rendal et al. (2018) used the rectifying linear unit (ReLU) function. Moreover, Rendall et al. (2018) adapted the deep neuronal network initially utilized in the MNIST dataset (Rendall, et al., 2018).

5.1.5 Network and Layers

In their study Rendall et al. (2018) used a network containing four million weights and fully connected layers or dense layers. Thereby, each input neuron is connected to every output neuron. Because, there were two classes identified by Rendall et al. (2018) as classification problems, the last layer consisted two output neurons. This is because, the number of classes determines the number of output layers in the last dense layer. Furthermore, when it comes developing a neural net the number of neurons as well as the number of layers and how they are configured is essential. To work around this problem, Rendall et al. (2018) used an already existing network structure and adjusted it according to their classification tasks (Rendall, et al., 2018).

As illustrated in the figure below, this is considered as transfer learning. To be specific, a deep learning model or neural net is trained on one set of data. Subsequently, the derived insights are generalized to be suitable for other classification problems. Generally, the method of transfer learning is suitable if there is not much training data available. Additionally, with this method useful classifiers can be leveraged (Pan & Yang, A Survey on Transfer Learning, 2010). For transfer learning Rendall et al. (2018) used the VGG-16 network which originally was developed by Simonyan & Zisserman (2015) during the ImageNet Challenge 2014 (Rendall, et al., 2018). The study showed that significant prediction improvements can be made by increasing the depth of the network to more layers. In addition, they indicate that the representations derived generalize well to other datasets. Also, they made their best-performing convolutional nets publicly available (Simonyan & Zisserman, 2015). Rendall et al. modified the VGG-16 network and renamed it to DNN_{VGG16} which receives RBG images as inputs compared to grayscale images the DNN_{4M} receives (Rendall, et al., 2018).

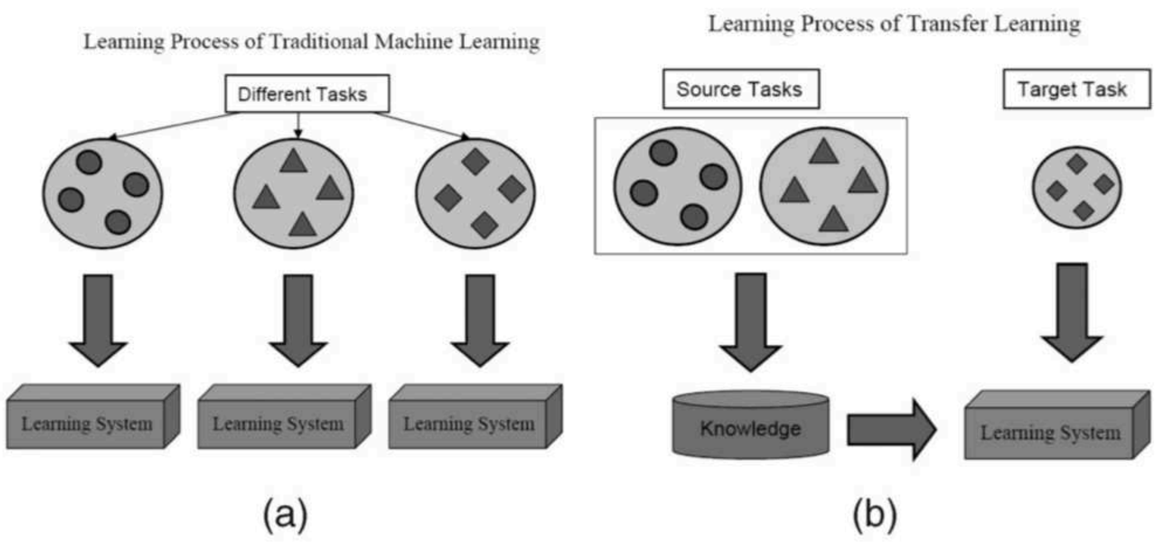


Figure 8: "Different learning processes between (a) traditional machine learning and (b) transfer learning.", (Pan & Yang, 2010)

However, a study by Huh, Agrawal, & Efros (2016) showed that finetuning the last layer from a convolutional neural net which is pre-trained on a full dataset in order to predict classes of randomly split data performs better compared to a CNN trained specifically on the random split of data as shown in figure 9 below. Nevertheless, for the prediction of classes of minimal splits, adding more data hurts the performance. This is because, adding training data contributes only to the model's class prediction accuracy if the data is correlated to the target task (Huh, Agrawal, & Efros, 2016).

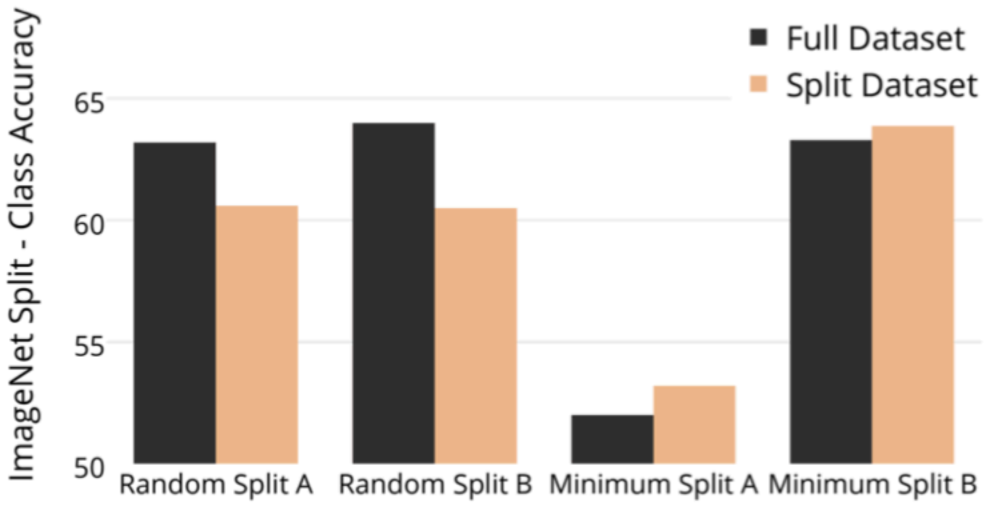


Figure 9: "Does adding arbitrary classes to pre-training data always improve transfer performance?", (Huh, Agrawal, & Efros, 2016)

Moreover, Rendall et al. (2018) added one dense layer which had to be trained after removing the last layer from the pre-trained network. On the one hand, this provides the model with degrees of freedom. On the other hand, this approach allows the model to adapt to the new target domain. Consequently, the number of samples for training the model can be reduced while allowing the network structure to be specified (Rendall, et al., 2018).

5.1.6 Model training

In order to solve the two classification problems both networks the DNN_{VGG16} and the DNN_{4M} were trained using transfer learning. Moreover, for hyperparameter tuning the gradient-descent algorithm was utilized in combination with the backpropagation method which makes the network a recurrent neural network (Rendall, et al., 2018).

To reduce the risk of overfitting the model to the training set and to minimize the generalization error, dropout was utilized (Rendall, et al., 2018). To be specific, a single model mimics having a large number of diversified network architectures by dropping out nodes and their connections randomly during training. This is especially suitable for large neural nets handling small datasets. Because large networks tend to be slow, it is difficult for them to deal with overfitting. Hence, dropping units and layers randomly significantly reduces overfitting and improves the performance of neural networks on supervised learning tasks, compared to other regularization methods (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). For the first classification task of predicting whether the pellet has a good or bad shape, the dataset was split as follows:

Number of images per dataset: Identifying pellet shapes	
Training	2961
Validation	1777
Test	1185

Table 5: " Distribution of images per dataset for the classification of pellet shapes.", (Rendall, et al., 2018)

The table below shows the distribution of the classes for the first classification task in percent:

Percentage of images from each class	
Good pellets	59%
Bad pellets	41%

Table 6: "Percentage of images of each class for good or bad pellets.", (Rendall, et al., 2018)

Regarding the task of whether pellets have a tail or not, the dataset was unbalanced. This was due to the fact that only 5% of all pellets showed a tail. Therefore, Rendall et al. (2018) down-sampled the main class. As a result, only 729 images were considered in total. Accordingly, the distribution can be seen in the table below (Rendall, et al., 2018).

Number of images per dataset: Identifying pellet tails	
Training	327
Validation	202
Test	200

Table 7: "Distribution of images per dataset for the classification of pellets with or without tails.", (Rendall, et al., 2018)

Thus, the distribution of pellets having tails, or no tails looks as follows:

Percentage of images from each class	
Without tail	59%
With tail	41%

Table 8: "Percentage of images of each class for pellets with and without tails.", (Rendall, et al., 2018)

As mentioned the previous chapters, the reason for splitting the dataset into a training set is because the weights of the neural network have to be updated. Furthermore, the validation set is used to validate the prediction accuracy of the model while the test set measures the generalization ability of the underlying DNN model (Zhang, Lipton, Li, & Smola, 2019). Additionally, there is an optimum in the model's ability to extract useful features to solve the underlying classification problem and by that is contributing to validation accuracy. In fact, if the optimal point is surpassed the model starts to overfit the training set. In this case, no additional learning can be achieved, because the validation error approaches zero. Therefore, Rendall et al. (2018) limited the number of epochs to 20. To be specific, 20 checkpoints are deployed and the training is stopped if the model rapidly decreases in validation accuracy at one of these points (Rendall, et al., 2018).

On the one hand, for the DNN_{4M} the weights were started at random values initially. On the other hand, for the DNN_{VGG16} the VGG-16 weights were utilized except for the last modified layer, where also random values were initialized. As stated by Rendall et al (2018) this approach allows the model to adapt to the new classification problems more easily (Rendall, et al., 2018).

The training was performed in two stages. First, the 500,000 parameters of the last layer were adjusted. Subsequently, all 16 million weights were adjusted in the network though at a lower learning rate which can be derived from the table below (Rendall, et al., 2018):

DNN_{VGG16} Network	Number of weights	Layer(s)
First Training Stage:	500,000	Last
Second Training Stage:	16.000.000	All
Learning rate:	0.0001	All

Table 9: "Training stages and training weights.", (Rendall et al. 2018)

What is more, Rendall et al. (2018) used image augmentation techniques to increase the size of the training set. In doing so, the images were rotated indiscriminately between 0 and 180° on both horizontally and/ or vertically. Because, the rotation does not impinge with the figure of the pellet the technique is prone to increase classifiers' performance (Rendall, et al., 2018).

5.1.7 Validation

For measuring the effectiveness of the networks ability to assign the right class to the images used in the training set, the loss function (cross-entropy) is presented in the equation below:

$$Loss = -\frac{1}{N}\sum_{j=1}^{N}\left[y_j \, ln\left(o_j\right) + (1-y)\, ln\left(1 - o_j\right)\right]$$

N can be interpreted as the number of training examples, whereby y_j is a binary class label for the jth training example and o_j is the output of a neuron in the softmax layer for the jth training example. After the loss is minimized through the use of a training set, the network can be regarded as fully trained. Therefore, it is ready for predicting new samples. Moreover, the networks ability to generalize has to be tested in a test set. Importantly, it should be independent of the training set, so that an unbiased estimation of a prediction error can be assessed (Rendall, et al., 2018).

Furthermore, each method was assessed for its prediction accuracy by measuring the number of samples correctly labelled for the quality as well as the tail classifier (Rendall, et al., 2018). The equation listed below calculates how often a classifier is correct (Fawcett, 2005):

$$Accuracy = \frac{TP + TN}{N}$$

Therefore, the accuracy is equal to the sum of correctly classified images related to the quality class and number of images correctly classified belonging to the tail class (True Positive and True Negatives) divided by the total number of samples. Hence, the numerator represents the sum of all correctly classified samples. Additionally, the equation shown below answers the question of how often a classifier is wrong (Fawcett, 2005):

$$Misclassification\ Rate = 1 - \frac{TP + TN}{N}$$

or

$$Misclassification\ Rate = \frac{FP + FN}{N}$$

Another robust prediction accuracy used not only by Rendall et al. (2018) but is a standard performance measurement machine learning in general, is the area under the receiver operating curve (AUC) used for multiclass problems (Fawcett, 2005). In detail, the receiver operating characteristics (ROC), a probability curve and the area under the receiver operating curve (AUC), representing the degree of separability, are based on the confusion matrix. To be specific, it is a performance measurement for binary or multiclass classification problems. Additionally, it is a table consisting of four quadrants with four different combinations of predicted and actual values. The table below illustrates the principle of a confusion matrix considering classification problems using only two classes (Fawcett, 2005):

Number of predictions (N)	Positive class	Negative class
Yes (actual class)	True Positives	False Positives (type 1 error)
No (predicted class)	False Negatives (type 2 error)	True Negatives

Table 10: "Confusion matrix and common performance metrics calculated from it.", (Fawcett, 2005)

If given a binary classifier and an instance, there are four possible outcomes. Abstracted to the pellet classification problem the table would mean that, If the quality of the pellet was predicted as good (true) and it actually is good (positive), it is counted as true positive. Further, if the quality of the pellet was predicted as bad (true) and it actually is bad (negative), it is a true negative. Additionally, if the quality of the pellet was predicted as bad (false) but it has good quality (positive), it is considered as a type 1 error or false positive. Likewise, if the quality of the shape was predicted as good (false) but it is bad (negative), it is a type 2 error or false negative (Fawcett, 2005).

However, since the predicted outputs of the models were continuous a threshold was needed to transform the output into a discrete prediction class. Since binary variables were used to encode the classes, Rendall et al. (2018) set the threshold value to 0.5. Thus, to tell how much the model is capable of separating classes the AUC – ROC curve concept was used (Rendall, et al., 2018). Generally, the model determines a model's capability of distinguishing between classes. Hence, the higher the AUC, the better the neural network is at predicting good pellets as good and bad pellets as bad. This also refers to the other classification problem of identifying tails. To be specific, the AUC determines the probability that randomly chosen positive instances are ranked higher than randomly chosen negative instances. Therefore, a models AUC of 0.5 means that it has no class separation ability, because both distribution curves overlap. In contrast, an AUC of 1 represents an ideal measure of separability (Fawcett, 2005).

The figures below show various AUC levels:

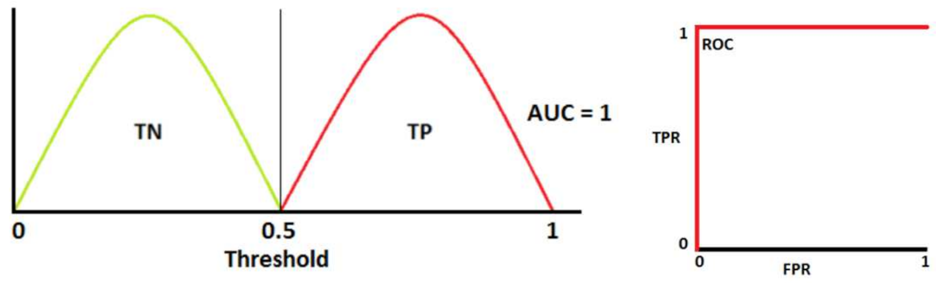


Figure 10: "Ideal measure of separability.", (Narkhede, 2018)

The figure above shows a model which perfectly distinguishes between true positive and true negative classes (Narkhede, 2018).

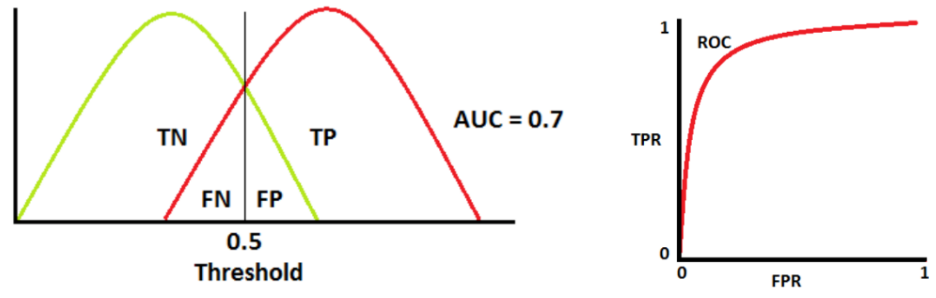


Figure 11: "70 percent chance of separability, introducing type 1 and type 2 error.", (Narkhede, 2018)

Above, a model having an AUC of 0.7 can be seen. In detail, this can be interpreted as a model's ability of separating positive from negative classes successfully by 70 percent chance (Narkhede, 2018).

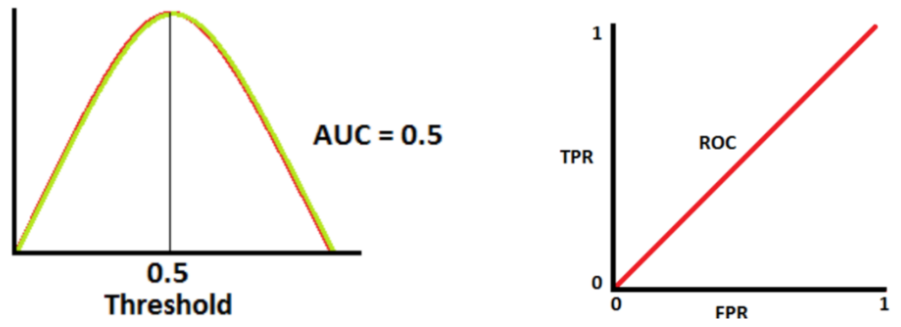


Figure 12: "Model can't distinguish between positive and negative classes.", (Narkhede, 2018)

The figure illustrates a model with AUC = 0.5. Basically, the two ROC curves are overlapping which means that the model is not able to distinguish classes not at all (Narkhede, 2018).

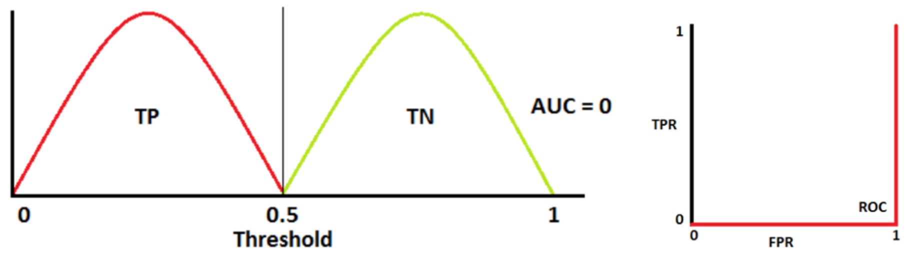


Figure 13: "Inverse predicting classes.", (Narkhede, 2018)

A model confusing positive for negative classes is shown in the figure above (Narkhede, 2018).

5.1.8 Results

Following results for good/ bad quality classification with the DNN_{VGG16} using pre-defined features as predictors had been achieved by Rendall et al. (2018):

Good/ bad quality classification	
Network	DNN_{VGG16}
Weight Initialization	ImageNet, transfer learning Last layer, random
Training Set	0.971
Validation Set	0.966
Test Set	0.967

Table 11: "Performance of pre-defined features as predictors." (Rendall, et al., 2018)

In the study of Rendall et al. (2018) two major advantages of deep neural networks compared to state-of-the-art methods were described. First, DNN's abilities to automatically abstract features from images. Although, predefined features are easier to interpret, adding new meaningful features is a complex task which can be overcome with the use of DNN. Furthermore, the algorithm is able to extract much more granulate and relevant morphometric pellet features compared to state-of-the-art approaches. Second, a 3% increase in accuracy was achieved, which has a significant effect on the performed product quality control and process economic outcome. What is more, the outputs of the layers can be illustrated and investigated which can lead to the observation of meaningful patterns (Rendall, et al., 2018).

Following results for tail classification with the DNN_{4M} using pre-defined features as predictors and DNN_{VGG16} utilizing transfer learning had been achieved by Rendall et al. (2018):

Tail/ no tail classification		
Network	DNN_{4M}	DNN_{VGG16}
Weight Initialization	random	ImageNet, transfer learning Last layer, random
Training Set	0.813	0.985
Validation Set	0.832	0.985
Test Set	0.825	0.975

Table 12: "Performance of classifiers for tail and no tail detection". (Rendall, et al., 2018)

The table showed below indicates that the shallow network DNN_{4M} is not an effective classifier. In their study Rendall et al. (2018) argued that this is a consequence of the small dataset and the random initialization of the weights. However, the DNN_{VGG16} performed remarkably well. This was mainly due to the fact that transfer learning contributed to better hyperparameter tuning. Notably, the DNN_{VGG16} achieved the highest classifying accuracy of 96% as illustrated in the AUC table below and showed that the silhouettes are relevant for identifying pellet shape and assessing tails (Rendall, et al., 2018).

The area under the receiver operating curve (AUC) for good and bad pellets classifier		
Network	DNN_{4M}	DNN_{VGG16}
Training Set	0.994	0.996
Validation Set	0.992	0.995
Test Set	0.988	0.993

Table 13: "AUC for good and bad pellets classifier.", (Rendall, et al., 2018)

The area under the receiver operating curve (AUC) for detecting pellets with tails		
Network	DNN_{4M}	DNN_{VGG16}
Training Set	0.994	0.996
Validation Set	0.992	0.995
Test Set	0.988	0.993

Table 14: "AUC for pellets with tails classifier.", (Rendall, et al., 2018)

Recap

To solve the underlying problem Rendall et al. (2018) compared two classifiers with each other. On the one hand, a deep neural net with four million randomly set weights and pre-defined features. On the other hand, a VGG-16 network with transfer learning (DNN_{VGG16}). However, the DNN_{VGG16} was superior over the DNN_{4M}. This was because the algorithm was able to extract much more features which led to a better hyperparameter adjustment (Rendall, et al., 2018).

5.2 An end-to-end neural architecture for optical character verification and recognition in retail food packaging

The second use case deals with ensuring food safety, which is of special interest for food manufacturers and is considered as an important requirement across all food supply chains. This is because incorrect labelling of pre-packaged food leads to product recalls or food safety incidents like food poisoning. In order to solve the task Ribeiro et al. (2018) combined and leveraged a global level convolutional neural network for high level food package image quality assessment with a local level fully convolutional network for "use by" date region of interest (ROI) identification. This is considered as a supervised learning problem for multi category classifications (Ribeiro, et al., 2018).

5.2.1 Business Case

Besides the high financial costs associated to product recalls, food manufacturers also endanger their reputation and the health of consumers. As a result, lawsuits and legal fines could emerge. The main reasons for incorrect labelling stem from human error as well as varying types of equipment faults. Currently, labelling checks are done manually and have no statistically significant correctness because quality checks are not performed on a continuous basis. Therefore, it is of highest interest for the respective industry to have an automatic and robust system, which is capable of recognizing expiry dates (Ribeiro, et al., 2018).

5.2.2 Datasets and collections

Similar to the first use case, three channeled images were used. Two datasets consisting 1404 and 6379 images were set up (Ribeiro, et al., 2018).

5.2.3 Tasks considered

Like in the first use case, Ribeiro et al. (2018) labeled the images. This was done in order to assign the images to categories. Pictures which had missing days, months, or both were categorized as "incomplete". Moreover, images were assigned to a category named “unreadable” if the images were distorted, blurred and/or showed illuminations. Additionally, food package label images,which had no date were assigned to this category as well (Ribeiro, et al., 2018).

5.2.4 Methods used

In their study Ribeiro et al. (2018) built two convolutional networks. In the first step, a global and fully connected CNN was introduced to perform the pre-processing and selection of suitable images for date recognition. To be exact, the high-level image quality information was assessed by training the network on the image categories readable/unreadable, complete/partial or no date. Moreover, the training set was used to train the network on classifying unreadable or incomplete labelled packing, so they could be discarded and separated from images which were complete. This is because images with missing values could have triggered false positives in the image recognition processes of the subsequent convolutional network. Furthermore, a second validation dataset was then used to evaluate the performance of the respective network which can be observed in Table 15 (Ribeiro, et al., 2018).

In the second step, a fully convolutional network (FCN) for image processing and detecting the actual use by dates in the images was utilized. To be exact, features are extracted on multiple levels to localize the “use by” date region of interest. Therefore, 70 percent of the collected images were used for fine-tuning the fully convolutional network whereas 30% were used for testing purposes. The underlying accuracies can be seen in Table 16 (Ribeiro, et al., 2018).

5.2.5 Network and layers

In order to identify and recognize the “use by” dates, a fully connected convolutional network was devised (Ribeiro, et al., 2018) with a size of two 2048 hidden unit layers. The basis for the FCN used by Ribeiro et al. (2018) was a scene text detector originally developed by Zhou et al. (2017) that directly produces word or line level predictions from complete images with the use of a single neural network (Zhou, et al., 2017). Hence, this network was fine-tuned on the food package dataset. Additionally, the architecture of the FCN consists of a feature extractor part, a feature-merging branch and an output layer. For the stem part of the FCN, PVANet was utilized. This was originally developed by Kim et al. (2016) and is explained in more detail in the underlying study (Kim, Hong, Roh, Cheon, & Park, 2016). This was done to reduce large variations in image size, which make it hard to choose the suitable kernel size for the convolutional network. To be specific, a larger kernel is needed for globally distributed information while a smaller kernel is needed for locally distributed information. Moreover, deep neural networks tend to overfit (Ribeiro, et al., 2018).

To overcome this difficulty, Ribeiro et al. (2018) made the network wider by utilizing filters with multiple sizes operating on the same level. In the feature-merging branch and in every merging stage the feature map from the previous stage is put into an unpooling layer first to double its size. Furthermore, it is then bundled with the current feature map (Ribeiro, et al., 2018). Subsequently, a convolutional bottleneck cuts down the number of prediction channels, because a large number of channels on large feature maps is likely to increase the amount of processing in later stages. Then, after the final feature map of the merging branch was produced by a 3x3 convolutional network, it is fed to the output layer. Moreover, the final output layer contained several 1x1 convolutional network operations to extrapolate 32 channels of feature maps into one channel of score map and a multi-channel geometry map. Further information can be found in the study of Zhou et al. (2017) (Zhou, et al., 2017).

5.2.6 Model training

Like in the first use case also Ribeiro et al. (2018) used transfer learning, pre-trained CNN weights and applied it to their food package image dataset utilizing the Rectifier Linear Unit (ReLU) as activation function in a fully-connected network consisting of two 2048 hidden unit layers. This approach was followed because of the limited amount of training data. Moreover, they modified the existing architecture of the Inception-V3 network. This was done by adding new fully connected layers and final softmax layer. Thus, to further decrease the risk of overfitting and to improve the performance of the network, the dropout technique was executed. To retain more information in the input layer, the dropout probability was set to 0.8, whereas for each hidden fully-connected unit it was set to 0.5 (Ribeiro, et al., 2018). To be specific, the probability of retaining a unit where p = 1, means no dropout, whereas for low values of p it is the other way around. Srivastava et al. (2014) mentioned in their study that typical values of p for hidden units are between 0.5 – 0.8 and for input layers with image patches as inputs a value of 0.8 is used. Moreover, the choice of p is connected to hidden units (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). To further reduce the risk of overfitting Ribeiro et al. (2018) used data augmentation and transfer learning as well (Ribeiro, et al., 2018).

5.2.7 Validation

As optimization or cost function cross-entropy was used. In addition to stochastic gradient-descent the Adam (adaptive moment estimation) optimization algorithm was taken into consideration as to include adaptive learning rate. In detail, it can handle very well sparse gradients and non-stationary data. Moreover, the method takes little memory, is robust and can be applied to problems that are large in terms of data and/or parameters. Furthermore, it is suitable to a broad range of non-convex optimization problems in machine learning (Kingma & Ba, 2015). Additionally, the model utilized loss functions for score map and geometry map. Therefore, the loss function can be defined as:

$$L = L_s + \lambda_g L_g$$

where L_s and L_g represent the losses for the score and geometry map, λ_g weights the influence between these two losses (Zhou, et al., 2017). Thus, Ribeiro et al. (2018) set the weighting to 1 (Ribeiro, et al., 2018). Henceforth, class-balanced cross entropy was used to reduce the number of hyperparameters to be improved and to reduce the complexity of the pipeline. Additionally, to decrease the variation of the sizes in images a loss for geometries was defined (Ribeiro, et al., 2018). This is described in more detail in the study of Zhou et al. (2017) (Zhou, et al., 2017). The above defined loss function was optimized by the Adam optimizer until the performance stopped improving (Ribeiro, et al., 2018).

5.2.8 Results

The filtering ability of the first network led to a reduction in processing power or computational costs for the second network. Thus, a useful side effect was that statistical information of image properties were derived which can be used as an indicator of triggering potential equipment faults. The table below shows that despite the small training set and high data variance, high classification accuracies for all tasks were achieved. To be exact each classification tasks had big leaps from training to the test set (Ribeiro, et al., 2018).

Complete vs.	Dataset	Images	Accuracy %
Unreadable	1	645 vs 645	90.1
	2	2847 vs 2847	96.8
Partial/No Date	1	645 vs 444	89.3
	2	2954 vs 2954	95.9

Table 15: "Global based experiment results with network 1.", (Ribeiro, et al., 2018)

The table below shows the detection accuracies for identifying the region of interest for clear images. In order to extract the date from the ROI the Maximally Stable Extremal Regions (MSER) algorithm was applied (Ribeiro, et al., 2018). To be specific, the MSER based scene detection method enables to detect most of the characters in images even if the quality is low (Yin, Yin, Huang, & Hao, 2013). Moreover, Ribeiro et al. (2018) concluded that the advantage of using the cascade of two networks is the filtering out of blurry or uncomplete images which can lead to false positive recognition. As a result, false positive recognition of “use by” dates can be inhibited (Ribeiro, et al., 2018).

	Tested Clear Images	Accuracy %
Dataset 1	240	98
Dataset 2	482	97.10

Table 16: "Local based experiment results with network 2.", (Ribeiro, et al., 2018)

Recap

To conclude, the architecture is based on two networks. In the first network optical character verification is performed in which suitable candidate images are verified. After, these images are then fed into the subsequent fully convolutional network for identification of the region of interests. Finally, through the utilization of the MSER algorithm optical character recognition is performed (Ribeiro, et al., 2018).

6. Deriving a general framework

6.1 Explanation

According to my conducted research I set out to derive a general framework/ workflow for the implementation of deep learning which can be seen in the event driven process chain (EPC), as shown in figure 12 below. To be specific, it describes the basic steps needed to solve a deep learning classification or regression problem. Furthermore, two already existing approaches by Kotsiantis (2007) and Wang, Cui, Wang, Xiao & Jiang (2018) were also considered.

6.2 Evaluation of the problem

The first step towards engaging a machine learning algorithm is to formulate the problem precisely. This includes defining the inputs and outputs as well as to choose an appropriate algorithm family. Although neural networks or deep learning algorithms accelerate in learning from raw sensor signals, images and mapping between sentences, they may not be the best solution for other tasks. Generally, deep learning can solve regression and classification problems. To be specific, a regression problem is about the prediction of a continuous value, while a classification problem is about prediction of discrete values (Zhang, Lipton, Li, & Smola, 2019)

6.3 Data collection

Starting with no or little data, its acquisition is required. On the one hand, this can be done through crawling data on the Web, accessing a company's data lake or retrieving data directly from machines and their sensors. On the other hand, new data can be generated by installing hardware like cameras to machines which take photos of goods or products (Roh, Heo, & Whang, 2018). Further, it is crucial to understand which data is required to solve the underlying regression or classification problem. Additionally, it has to be clarified who the owner of the data is and to have permission for further processing. Moreover, in industrial settings it is often hard to access and find the actual location of the data in question. Additionally, retrieving data from different resources like machines and systems can be a complex and time-consuming task. For example, in some cases industrial machines are self-contained systems and do not share data across networks (Roh, Heo, & Whang, 2018).

6.4 Data checking & pre-processing (cleaning)

Once the data has been acquired successfully, the next step is to tag samples with one or more labels. To be specific, assuming enough images were produced by a camera system mounted to a machine, experts can start annotating them. Thereby, a distinction has to be made between the use of existing labels to let a semi-supervised learning algorithm predict the rest of the labels, or crowd-based labelling of individual examples can be performed. However, the latter process is often very expensive. This is why utilizing "weak labelling" can be an alternative labelling technique. To be exact, the aim of this approach is to generate a large quantity of low-quality labels which compensate for the overall lower quality. Furthermore, most of the research in data labelling for machine learning has been focused on classification problems rather than regression problems. Additionally, depending on the data type labelling techniques differ significantly. In fact, extracting information from text is vastly different from object detection and classification of images. Nevertheless, if acquiring new data is too difficult, the existing dataset can be re-labelled and model training can be improved. On the contrary, re-labeling and cleaning the existing dataset may be a faster approach to increase the accuracy of the classification or prediction algorithm compared to acquisition of new data. Thus, for improving data, cleaning and re-labeling is recommended and can be improved by transfer learning or making the model more robust against noise and bias (Roh, Heo, & Whang, 2018).

Another task which has to be performed before feeding data into a deep network is data-preprocessing. In this phase missing continuous values are often replaced by the mean. However, there are several statistical techniques to normalize datasets. As a result, after normalization had been applied all features are brought to the same scale. Second, for classification tasks discrete values have to be pre-processed using one-hot encoding. Hence, the data is transformed into vectors (Zhang, Lipton, Li, & Smola, 2019). Further, if the data is biased, noisy or labelled incorrectly, production machine learning platforms like TensorFlow Extended can be used. These have individual components which aim to reduce data errors. In addition, there are several state-of-the-art data cleaning systems available on the market (Roh, Heo, & Whang, 2018).

6.6 Model selection

After the data has been checked and cleaned, an algorithm has to be selected. Usually, several candidate models have to be evaluated. These can be linear neural networks, deep neural networks, recurrent neural networks and convolutional neural networks. Often the models in comparison are fundamentally different from each other. This is why one should have knowledge about the specific model related architectures as well as their advantages and disadvantages. One the one hand, models of the same class can be compared with divergent parameter settings. On the other hand, models can be compared by their varying numbers of hidden layers, hidden units and different activation functions applied to each layer. Hence, to be able to determine the best performance and to choose the best hyperparameters among potential and varying candidate models, a validation set needs to be employed (Zhang, Lipton, Li, & Smola, 2019).

6.7 Training and validation

In this process the data has to be split into a training set which comprises a collection of randomly selected and classified samples. Consequently, the training set is used to tune the loss functions, so that the labels can be predicted more accurately and to reduce overfitting. However, the generalization error cannot be measured on training data solely. Therefore, a validation set has to be set up as well. Nevertheless, using only one specific part of the data for the validation set and separating it is suboptimal and leads to loss of data. For that reason, K-Fold Cross-Validation technique is used. For example, the data is split into non overlapping subsets or four blocks where the first 75 percent (three blocks) are kept for training purposes and the last 25 percent for validation. Since, there is a risk of leaving relevant data aside by defining only the last 25 percent as the testing data, K-Fold Cross-Validation utilizes a new block for every iteration and summarizes the results at the end (Zhang, Lipton, Li, & Smola, 2019). Especially, this process is time-consuming and requires computational resources. Therefore, if the network is large, training can be performed in parallel on multiple servers (Goodfellow, Bengio, & Courville, 2016).

What is more, training and validation errors have to be compared with each other to determine if the model is under- or overfitting. This is the case when the gap between the training and validation error is substantial. For example, if the model lacks the ability to reduce the generalization gap between training and validation errors one could assume that the model is too simple to capture the pattern. This assumption is often referred to as underfitting. In contrast, a training error substantially lower than the validation error can indicate overfitting. However, in practical applications the best predictive models are known to perform much better on training data compared to validation data. To conclude, Zhang et al. (2019) in their book recommend focusing more on the validation error than comparing the gap between the training and validation errors. To further decrease the risk of overfitting dropout technique can be used (Zhang, Lipton, Li, & Smola, 2019).

When solving classification problems, it has to be checked if the validation and test set include representations of the target distribution. Meaning, assumed one wants to solve the problem of classifying images of good or bad quality glass the total number of images received from the manufacturer available is crucial. Especially, if there is not sufficient data available for training a neural network, then additional data from different sources (the web or other companies) has to be gathered. Hence, each the training and testing set for example have to include 25 percent of the glass images source initially from the manufacturer while the other 50 percent flow into the training set including also the total number of glass images collected from other sources. (Zhang, Lipton, Li, & Smola, 2019)

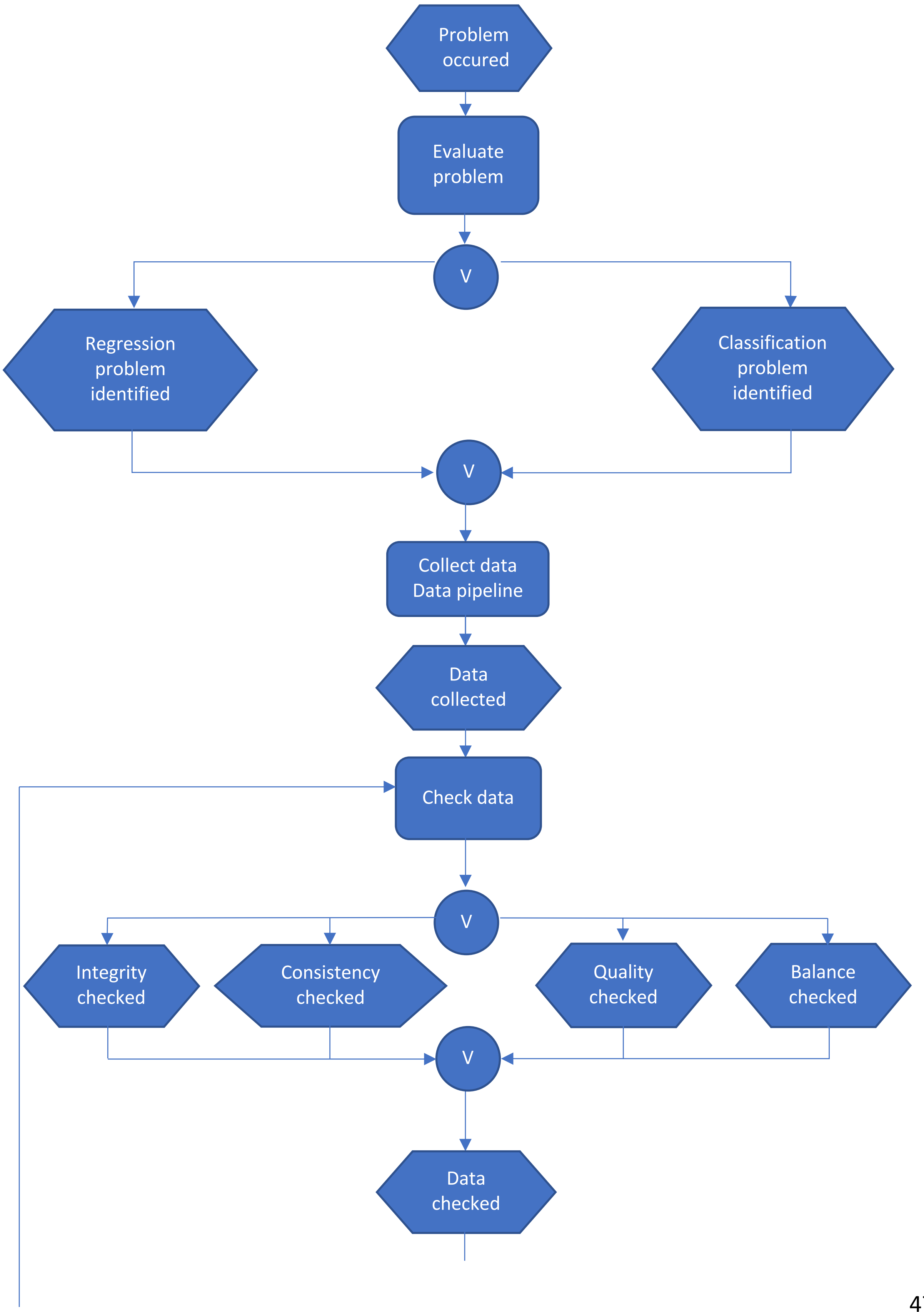
In the process of training a neural network has to perform multiple iterations over the dataset. Thus, from the total number of samples mini-batches are passed over to train the algorithm, which is usually about 10 percent of the total representation. Each pass updates the model and aims to increase the validation accuracy. This process is fundamental to training a machine learning algorithm. Moreover, it is important that the batch sizes are reasonable to make use of the available GPU processing power at parallelizing operations. Because, reading data is often associated with generating performance bottlenecks for training, a data loader is utilized (Zhang, Lipton, Li, & Smola, 2019).

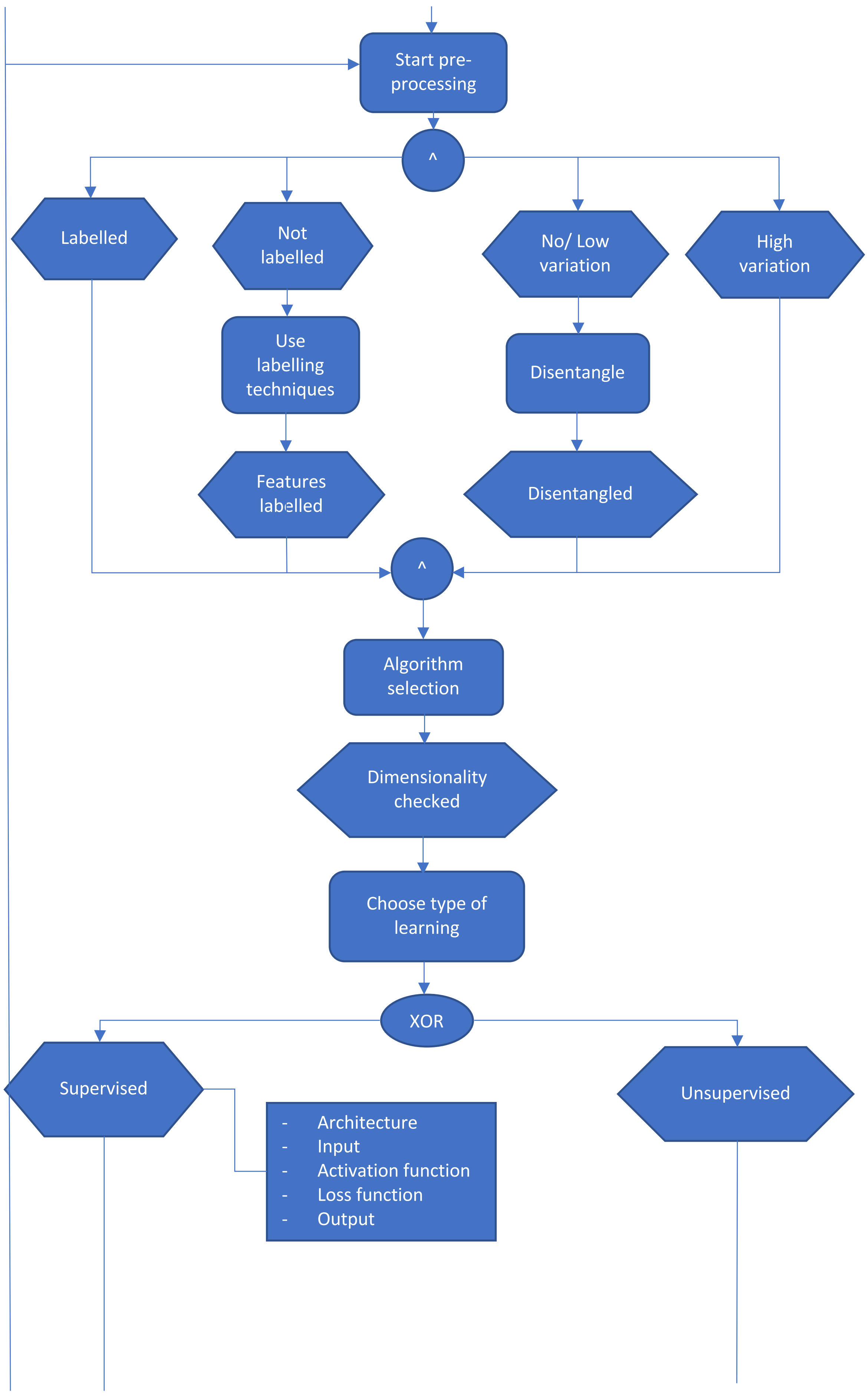
6.8 Generalization error

Once, training is finished the algorithm has to be run against a testing set, so that the generalization error can be estimated. In other words, this data is used for testing the final model to confirm the actual predictive ability of the network. Depending on whether the generalization error is acceptable or not, different steps have to be repeated. This also includes readjustment of the hyperparameters (Zhang, Lipton, Li, & Smola, 2019).

6.9 The Framework

Below the derived framework and its processes which were described throughout chapter 6 can be found.




Start pre-processing
^
Labelled
Not labelled
No/ Low variation
High variation
Use labelling techniques
Disentangle
Features labelled
Disentangled
^
Algorithm selection
Dimensionality checked
Choose type of learning
XOR
Supervised
Unsupervised
- Architecture
- Input
- Activation function
- Loss function
- Output

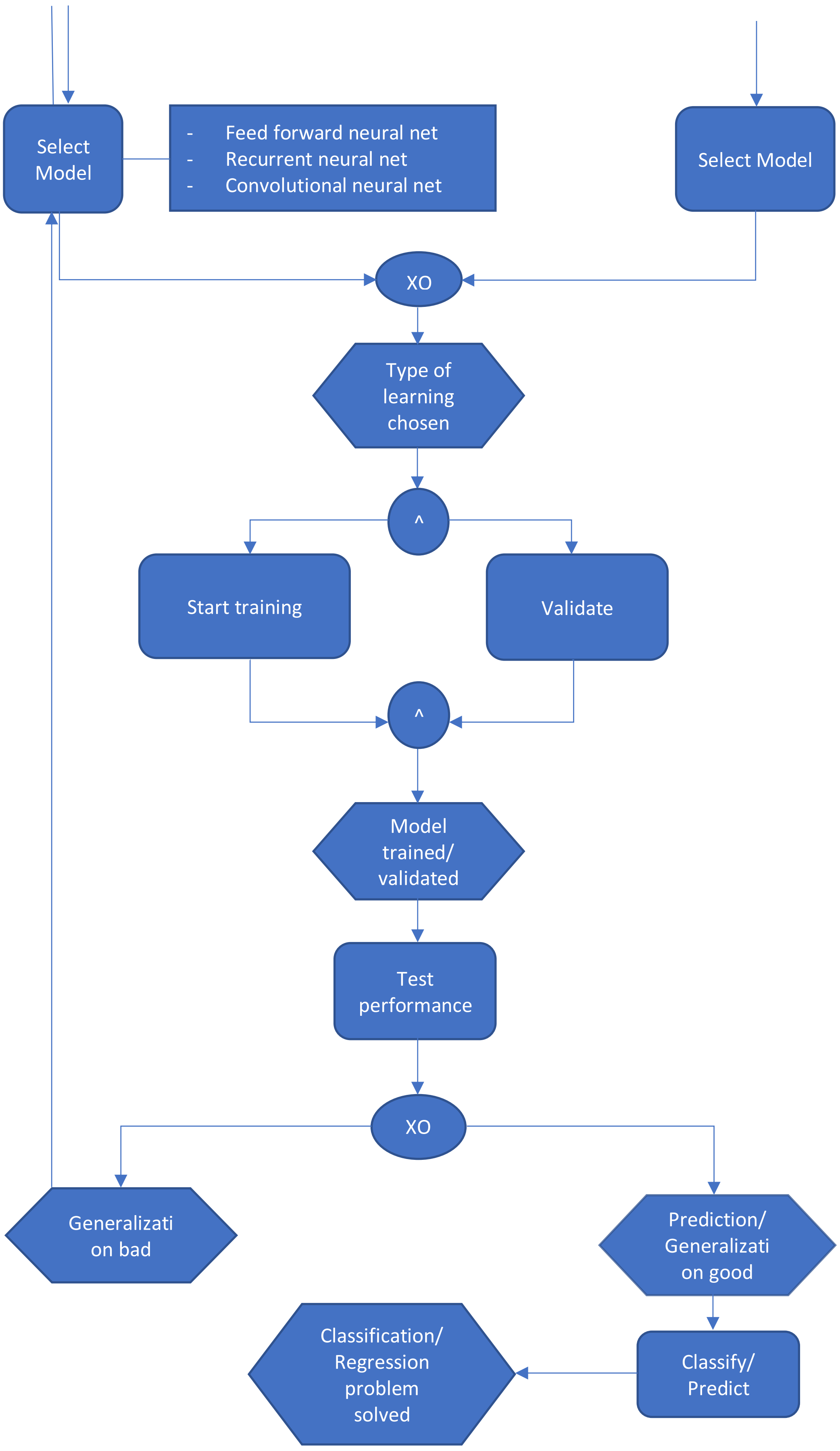


Figure 14: Basic deep learning framework/ workflow

7. Development and quantitative validation of a deep neural network for predicting on time deliveries (OTDs)

7.1 The problem

On time delivery is a performance measure of process and supply chain efficiency. It is the actual delivery time of goods or products which were sent by a supplier to its customers. Ideally, the promised delivery time is equal to the actual delivery time. However, in the real world this is often not the case. Thus, a supplier's inability to comply with the agreed delivery period or deadline could indicate bottlenecks along the supply chain. In procurement, it is not sufficient to measure the delivery performance after the circumstance became known, but also to anticipate potential late deliveries. This is because, lead times are important to ensure an orderly and timely operation in production or manufacturing systems. Especially, when operating on a (JIT) just in time basis. Additionally, delivery performance has become a foundational metric of success (Pai, Hebbar, & Rodrigues, 2015).

An accurate prediction of the actual delivery time can contribute to a continuous and accurate flow of materials at the right time and in the right quantity with optimum cost while ensuring better adaption to dynamic environments. Moreover, it helps the receiver of the goods or materials to gain or hold a competitive edge over its competitors in the market and for a longer time (Pai, Hebbar, & Rodrigues, 2015). Additionally, the study of Green, Whitten, & Inman (2008) concluded that there is a positive relationship between the performance of logistics and organizational performance within the sector of manufacturing (Green, Whitten, & Inman, 2008). Moreover, low delivery speed or late delivery is significantly related to the complexity of a supply chain (Milgate, 2001).

7.2 The idea

Our company specializes in developing source to pay software solutions. However, not a single AI based solutions has been developed so far. In order to solve this classification problem of whether a delivery will be late or not and to enhance our existing software product portfolio towards artificial intelligence, a neural network will be developed with the use of the general framework derived in the previous chapter. Besides, by solving this problem for our industrial customers huge value would not only be generated for our customers but also for our company.

The idea is that the system can predict already when an order is placed if it will be delivered on time or not compared to the agreed delivery date. This will be shown in our solution immediately after an order has been placed (i.e. transmitted to the solution). It could even be displayed before an order is placed if the data is available at that point. This is considered a binary classification problem and a neural network will be used to perform the classification.

7.3 Data

The raw data is fetched via SQL from the database and written into a CSV file. Subsequently, it is processed by the Pandas Library which converts it to an array of rows. To be specific, it consisted 1.517.533 rows and 70 columns. In the next step, the raw data was pre-processed by removing outliers and using diverse labelling techniques. The figure below shows a structure of the features as well as the targets (labels) which need to be predicted.

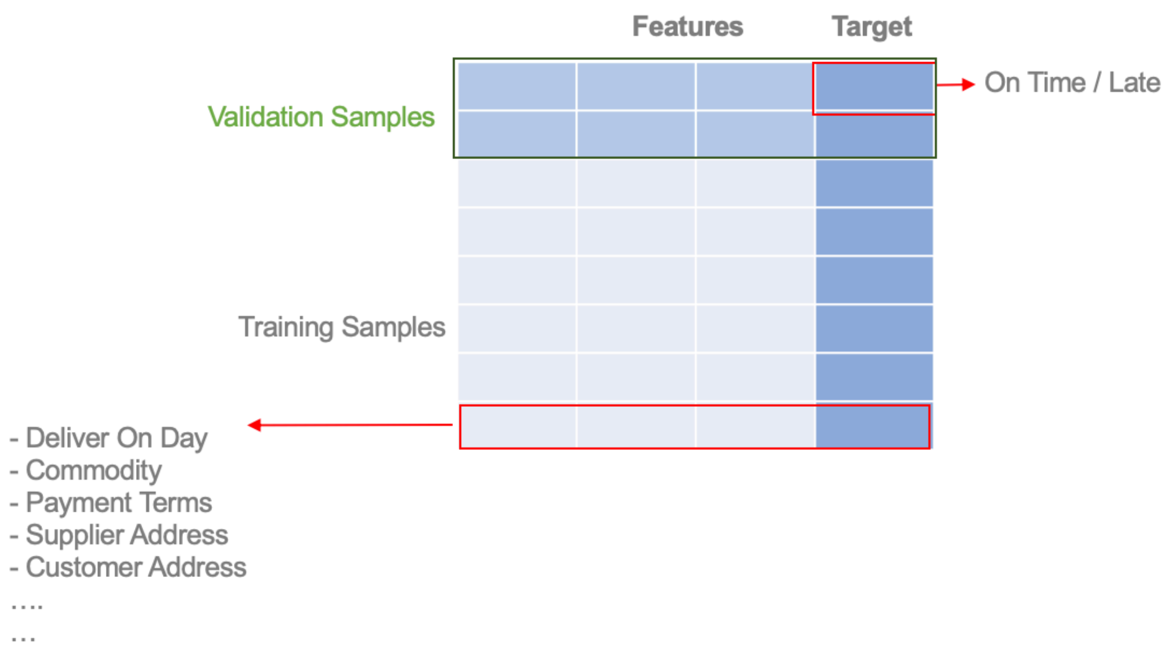
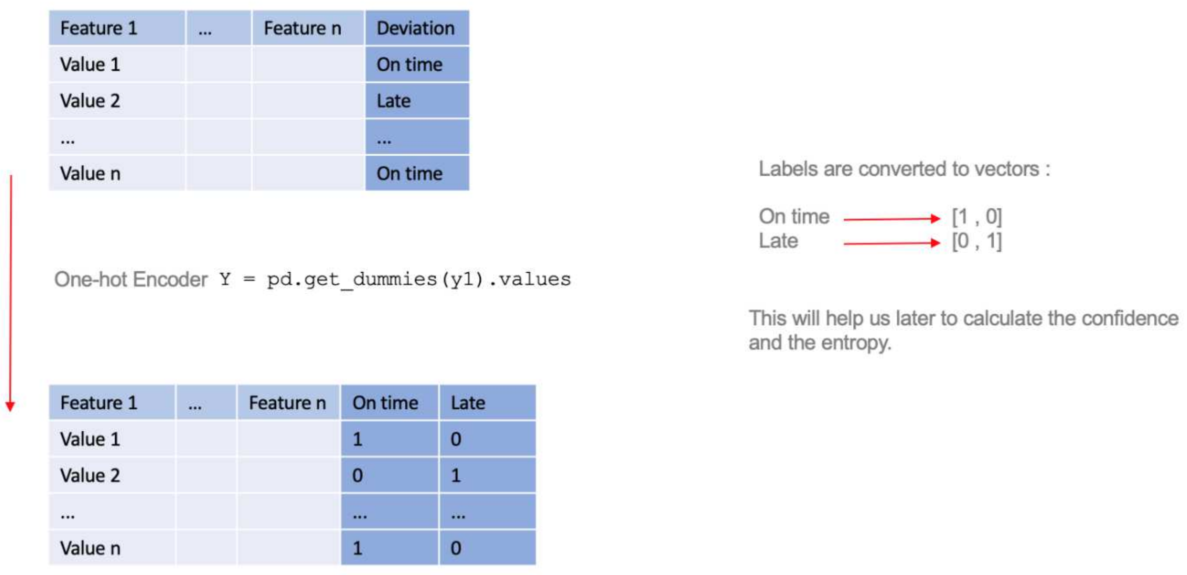


Figure 15: Data

Moreover, to overcome the problem of how to represent categorical data or discrete values one hot encoding was used. After, the processed data consisted 1.474.869 rows which is roughly about 97 percent of the raw data and 30 – 500 columns. However, the dataset contained imbalanced classes. This was due to the fact that the class “On Time” represented with 1.417.245 rows about 96 percent of the dataset compared to class “Late” which represented about 4 percent.



Feature 1	...	Feature n	Deviation
Value 1			On time
Value 2			Late
...			...
Value n			On time

One-hot Encoder `Y = pd.get_dummies(y1).values`

Feature 1	...	Feature n	On time	Late
Value 1			1	0
Value 2			0	1
...			...	...
Value n			1	0

Labels are converted to vectors :

On time	[1 , 0]
Late	[0 , 1]

This will help us later to calculate the confidence and the entropy.

Figure 16: One-hot encoding

7.4 Model selection

The figure below shows the network principle used for solving the binary classification problem. Thereby, the neural network receives the features as input and compares its predictions with the actual labels. It then tries to minimize the cross-entropy loss (the difference between predicted and actual class). During the training phase the algorithm learns to adjust the values of the hyperparameters by utilizing Adam as optimization algorithm.

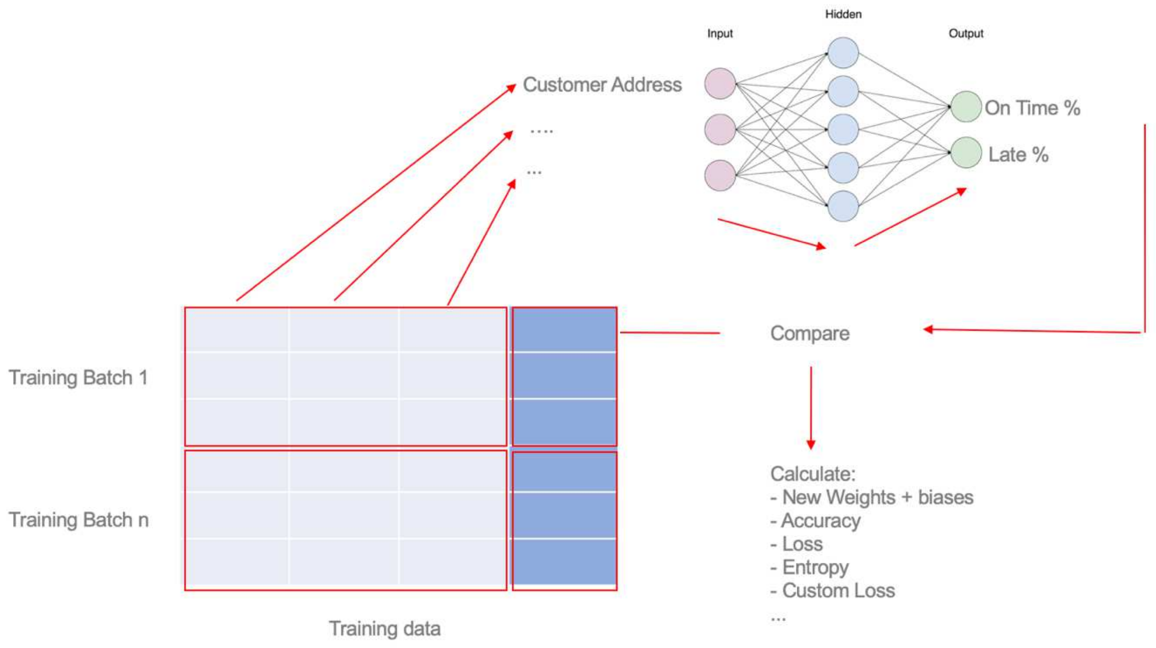
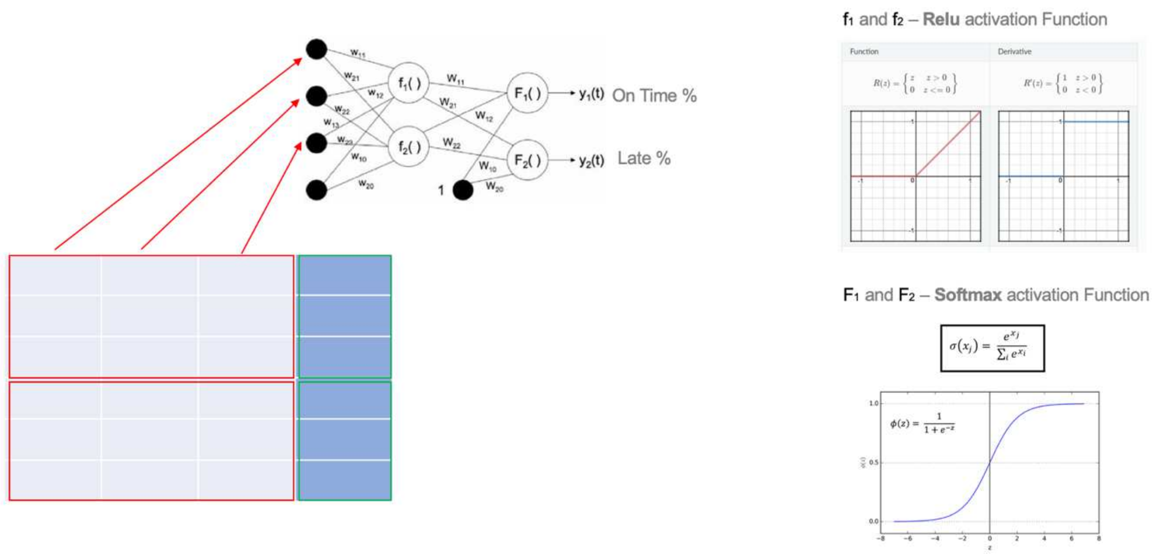


Figure 17: Network principle

The hidden layers passing of the values to other neurons was decided by the ReLU function. Therefore, only positive elements are retained and negative elements are discarded by setting the nodes to 0. This makes optimization during training better while it reduces the issue of the vanishing gradient problem. In the subsequent layer softmax regression was utilized forcing the model to use probabilities instead of taking the values from the output layer directly ensuring that on unseen data the probabilities are nonnegative and summed up to 1 (Zhang, Lipton, Li, & Smola, 2019).

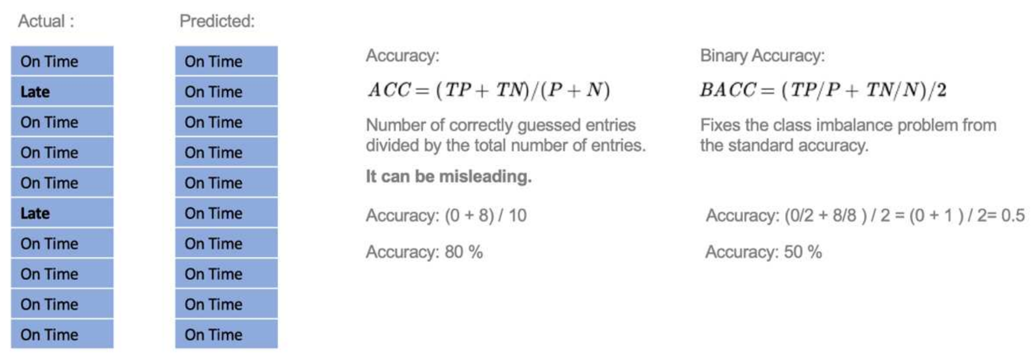


$f_1()$
$f_2()$
$F_1()$
$F_2()$
$y_1(t)$ On Time %
$y_2(t)$ Late %
f_1 and f_2 – Relu activation Function
Function
Derivative
$R(z) = \begin{cases} z & z > 0 \\ 0 & z <= 0 \end{cases}$
$R'(z) = \begin{cases} 1 & z > 0 \\ 0 & z < 0 \end{cases}$
F_1 and F_2 – Softmax activation Function
$\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}}$
$\phi(z) = \frac{1}{1+e^{-z}}$

Figure 18: Activation functions

7.5 Training and quantitative validation

The figure below illustrates the accuracy of the model. Although, the model predicted 8 out of 10 cases correctly it can be misleading. Because, it shows the result of an unbalanced training set. Thus, to measure the model's prediction accuracy more precisely the binary accuracy was considered as performance measure. This fixes the class imbalance problem from the standard accuracy.



Actual :	Predicted:
On Time	On Time
Late	On Time
On Time	On Time
On Time	On Time
On Time	On Time
Late	On Time
On Time	On Time
On Time	On Time
On Time	On Time
On Time	On Time

Accuracy:

$ACC = (TP + TN)/(P + N)$

Number of correctly guessed entries divided by the total number of entries.

It can be misleading.

Accuracy: (0 + 8) / 10

Accuracy: 80 %

Binary Accuracy:

$BACC = (TP/P + TN/N)/2$

Fixes the class imbalance problem from the standard accuracy.

Accuracy: (0/2 + 8/8) / 2 = (0 + 1) / 2= 0.5

Accuracy: 50 %

Figure 19: Accuracy and binary accuracy

On top of the figure below the binary accuracy as well as the loss of the training set is illustrated for the late deliveries. The lower graph shows the same but for the validation set. Hence the training set had a binary accuracy of 0.86 and a loss of 1.85. However, binary accuracy of the validation is 0.76 and the loss 5.10.

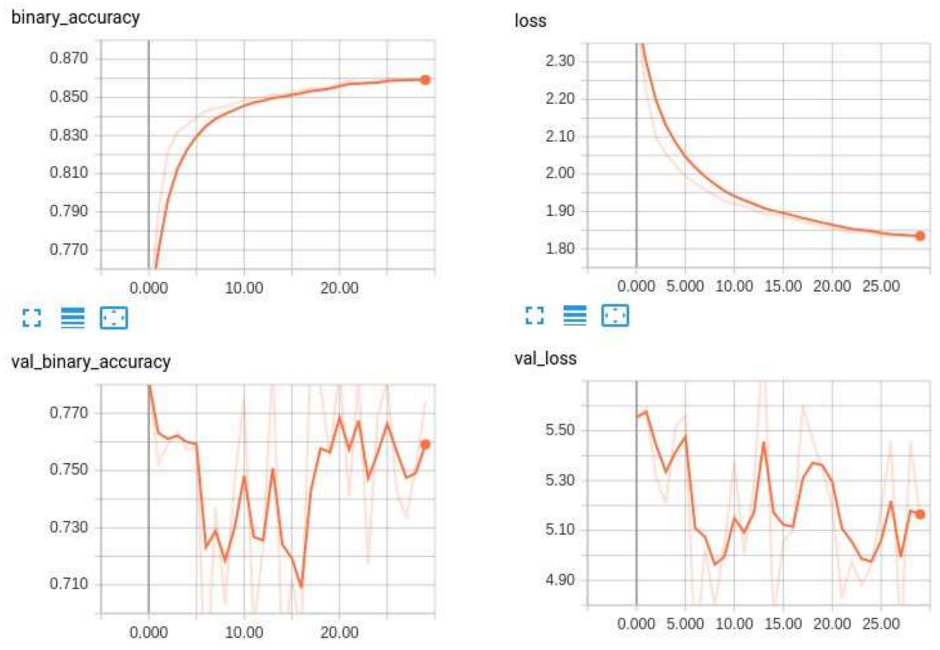


Figure 20: Training vs validation

Additionally, for the training set the model predicted in about 92 percent of the cases the class correctly. In contrast, the generalization ability of the model is bad which can be seen in the validation accuracy of 70 percent. Zhang, Lipton, Li & Smola (2019) mentioned in their paper that a scenario like this may indicate that the model is overfitting (Zhang, Lipton, Li, & Smola, 2019).

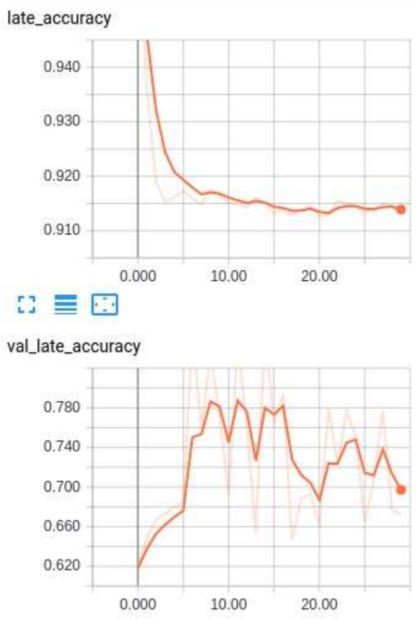


Figure 21: Late deliveries training vs validation

Next, a new feature was added to see how it affects the performance. The figure below shows the accuracies for the old (orange color) and the feature enhanced model (blue color). According to the metrics, the performing epoch of the model is iteration 15, because subsequent iterations only increase the validation loss. This can be observed in the chart at the bottom right. In addition, this can be an indication that the model has overfit after step 15.

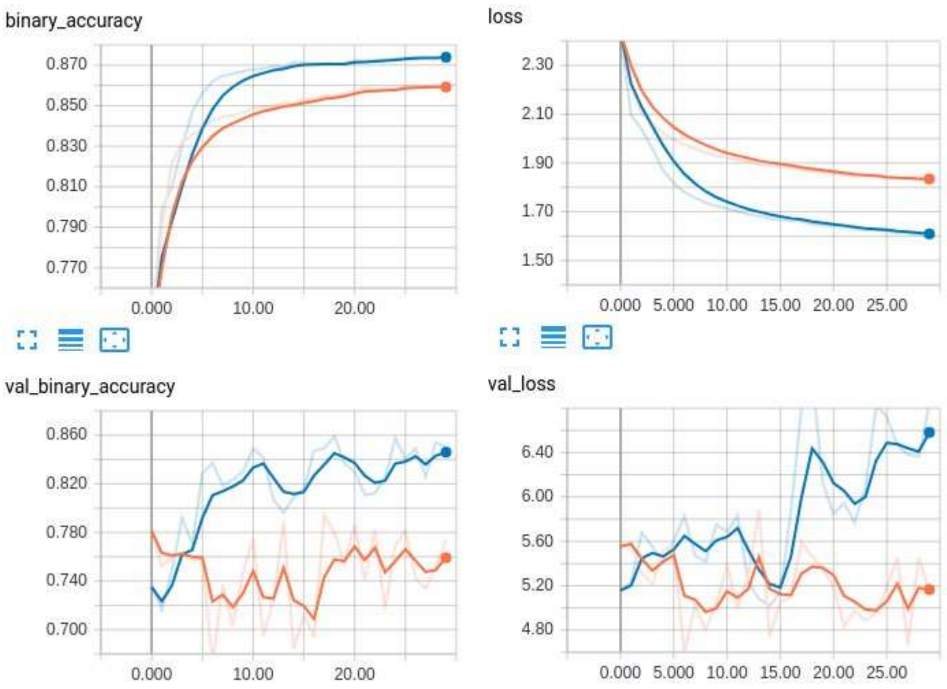


Figure 22: Old vs new feature

When looking at binary cross-entropy, we can conclude that after adding this new feature, the model gained more confidence. Thus, the distances between the actual value vector and predicted value vectors were getting smaller.

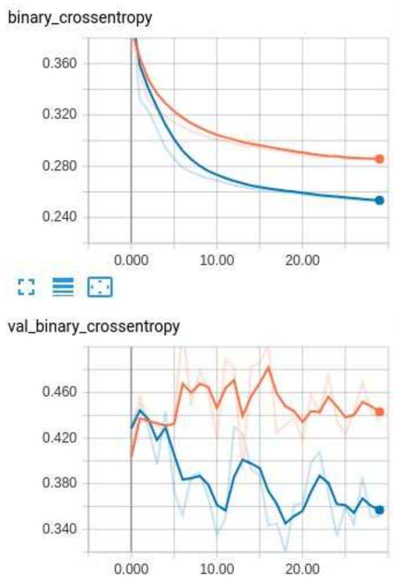


Figure 23: Old vs new feature cross entropy

Furthermore, it can be seen in the line chart below that the model performed much better on the validation set for on time deliveries compared to the first model. In contrast, the validation of the late deliveries decreased in accuracy compared to the initial model. Therefore, adding a new feature decreased the accuracy of the “late deliveries” from 0.76 to 0.72 whereas for on time deliveries it increased from 0.71 to 0.82 percent.

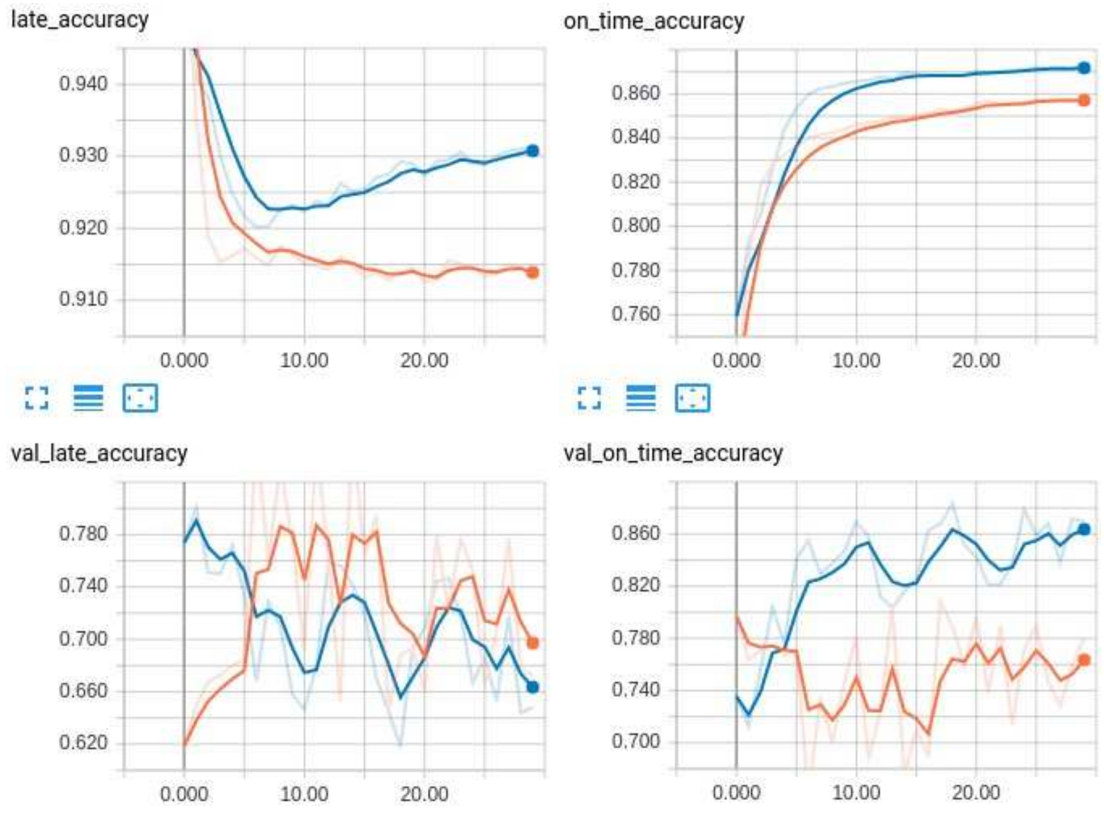


Figure 24: Training vs validation of one time and late deliveries

7.6 Development

As it is illustrated in figure 18, Python was used for programming. After the problem had been evaluated, the collected data was fed for data checking and analysis into Pandas, an open source tool for data analysis. Next, TensorFlow an open source deep learning library and Keras, a high-level API built on TensorFlow was utilized. Compared to other systems it has better support for distributed systems and has development funded by Google. It provides primitives for defining functions on tensors and automatically computing their derivatives. For visualization purposes TensorBoard which has built-in visualization tools were used (Google, 2019).

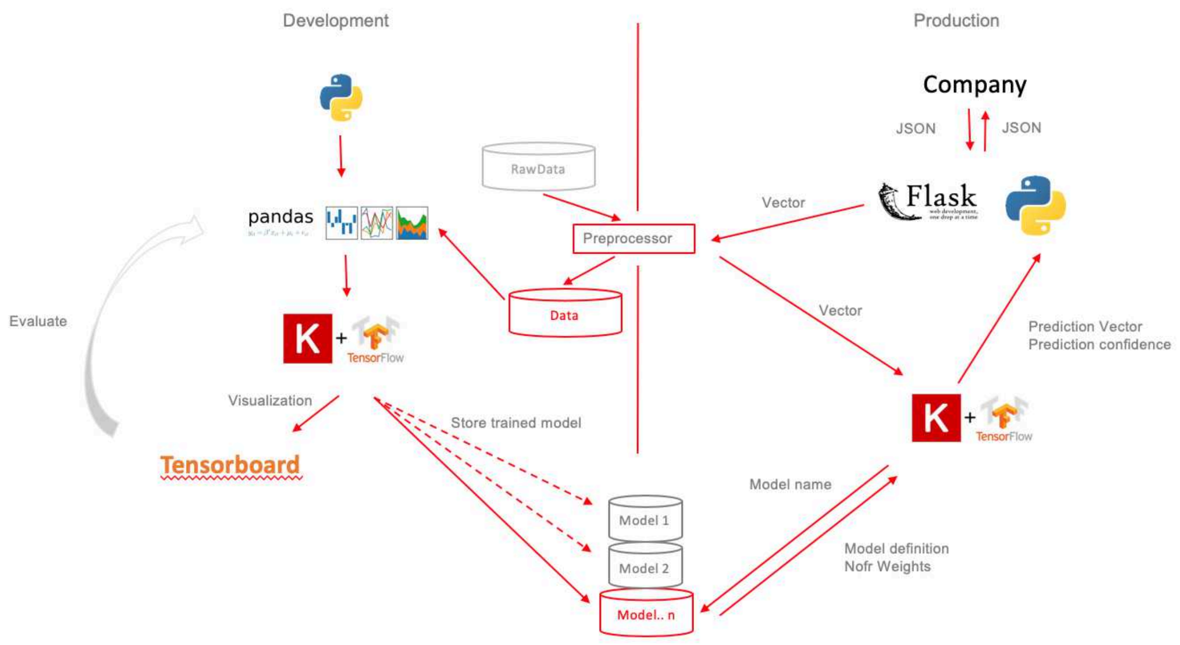


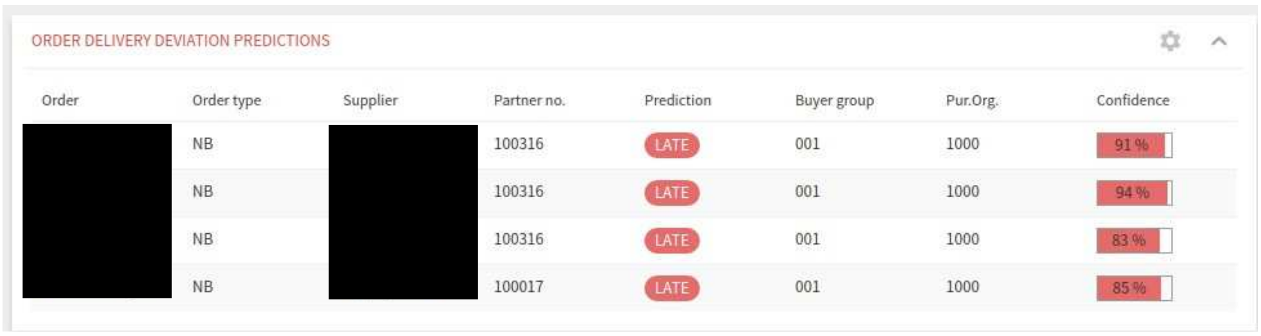
Figure 25: Development and Production

7.7 Production and outcome

A pre-trained deep learning model was developed which is able to predict if an order will be late as soon as the data is received from an external ERP system. Moreover, it is an independent solution which can be trained with historical data and then be used through an API from any of our solutions. The model will be integrated into our Workflow-Engine where it can be configured on a per customer basis. Further, the orders which are predicted as late are displayed in a dashboard box where the customer has an overview of the late orders and can take measures in order to react to the anticipated situation. The vectorization of the data enables us to see how confident the predictions are which is useful for the model evaluation during the development process. Also, in production we can serve the confidence of the prediction alongside the prediction itself, so that the customer can take different types of actions based on the confidence of the prediction. In addition, the models can be adjusted quickly using the Keras API which is perfect for prototyping a neural network. Furthermore, new features can be easily fed into the network using the Pandas library. Thus, multiple models can be pretrained for a customer and the results can be served in milliseconds. What is more, any solution of our company can integrate this which makes it scalable.

7.8 End-user experience and next steps

The results of the prediction are represented to the End-user in form of a dashboard box. Moreover, the box indicates the prediction if the on-time delivery will be late or not. Additionally, the prediction confidence will be displayed as well. As a result, the customer will be able to set up a specific chain of events based on the prediction and confidence using a workflow engine. In the next steps and in addition to the late or not late classification, a regression of how much early or late an order might be in days will be developed.



ORDER DELIVERY DEVIATION PREDICTIONS

Order	Order type	Supplier	Partner no.	Prediction	Buyer group	Pur.Org.	Confidence
[redacted]	NB	[redacted]	100316	LATE	001	1000	91 %
[redacted]	NB	[redacted]	100316	LATE	001	1000	94 %
[redacted]	NB	[redacted]	100316	LATE	001	1000	83 %
[redacted]	NB	[redacted]	100017	LATE	001	1000	85 %

Figure 26: OTD box in the front end

8. Discussion and Conclusion

The final chapter of the thesis summarizes the previously presented contend and discusses the thesis limitations. Finally, a set of standard circumstances to be suitable for the implementation of neural networks in manufacturing will is presented.

8.1 Summary

Machine learning and the basics of deep learning as well as their practical implementation in manufacturing is the main focus of the thesis. To make these rather abstract terms tangible for manufacturing companies, so that they can adapt to a structured approach for the implementation of neural networks more easily and therefore stay competitive, this thesis proposes a general framework of the processes needed to implement neural networks. The general framework describes on a high-level process steps which serve as a helping tool which should be taken into consideration when implementing neural networks in practice. The thesis provides a holistic view by reviewing the history of artificial intelligence in general and by introducing existing methods of neural networks as well as their practical application in manufacturing on a theoretical basis. In addition, the utility of the framework was tested by taking its provided process steps into consideration during self-development of a neural network which aim is to classify deliveries based on whether they are on time or late.

As a base for the development of the general framework, chapters 2, 3 and 4 review the history of artificial intelligence, introduce the discipline of machine learning and neural networks. Moreover, in chapter 4 the concept of neural networks/ deep learning is explained and the related methods are elaborated. Thus, state-of-the art theoretical concepts were considered and summarized on which the general framework builds up. Subsequently, chapter 5 assesses state-of-the-art applications of neural networks in the manufacturing industry. For this purpose, two practical use-cases were analyzed qualitatively. In both cases similarities can be seen in terms of underlying data, chosen neural network model, transfer learning, hyperparameter tuning activation functions, training, testing and validation of the model.

Chapter 6 derives a general framework for the implementation of neural networks on behalf of the insights gained from chapter 2 – 4 as well as from the qualitative analysis of the practical use-cases assessed in chapter 5. The framework combines methods and tools from all aforementioned domains and starts with (1) evaluation of the problem, (2) checking the data, (3) pre-processing the data, (4) selecting the algorithm, (5) the model (6) type of learning, (7) training, (8) validating, (9) testing and (10) optimizations.

Consequently, in Chapter 7 a neural network is developed with the use of the general framework derived in chapter 6 to solve the classification problem of classifying on time deliveries. To be specific the framework provided a helping tool and showed that a neural network can be successfully developed to solve the underlying problem. The model's ability to generalize was validated and tested by the statistical measurement of binary accuracy and validation accuracy.

8.2 Limitations and problems

The framework presented is very general and reflects only on a high level the basic steps needed to implement machine learning utilizing neural networks. Moreover, the wide selection of algorithms and statistical tools needed in different steps of the process chain made it difficult to find a "one framework fits all" solution. Initially, the qualitative analysis was supposed to be made by visiting three Austrian manufacturing companies. However, none of the asked companies had implemented a neural network application within their manufacturing processes so far. Therefore, I had to search for practical applications of machine learning in manufacturing described in academy by browsing through various university databases. After I found 35 recent use cases, only two of them were based on neural networks. Although it may seem that both practical use cases relied on images as input as well as on convolutional networks and transfer learning, the techniques used were vastly different from each other and couldn't be compared fully. On the one hand, this was due to the fact that there was no data and, on the other hand no source code available to be compared. Though, this would be beyond the scope of the thesis. Regarding the development of our neural network to solve the problem of classifying whether a delivery will be on time or late improvement of the current classification accuracy is needed. Although no industry accuracy benchmark could be found in the course of this research, the customer in question expects a model classification accuracy at least above 90 percent. Still we showed that the problem could be solved and that a customer is willing to pay for the solution. However, I could not compare our developed model to another existing model solving the same underlying problem. This was due to the fact, that existing approaches were superficial and lacking the data as well as source code.

8.3 Recommendations for Implementation

When considering the implementation of a neural network in the manufacturing industry, the following recommendations synthesize the qualitative as well as quantitative findings and propose concrete suggestions.

The first question which should be answered is whether a pattern can be detected or not. Furthermore, machine learning to be implemented in manufacturing is favorable if a machine or process operates in a dynamic environment and therefore needs to adapt itself. Otherwise, cheaper and more sufficient solutions may be applicable instead. Moreover, it has to be clarified if the problem which needs to be solved is either a classification or regression problem or both. Next, the availability of the data has to be checked. Subsequently, data has to be gathered and organized. After, a very time-consuming transformation of the data follows. The process starts with an analysis of the data to check the consistency which may requires normalization, labelling and balancing. This ensures that no group is unrepresented or discriminated in the training set and to minimize the risk of decreasing the effectiveness of the underlying model. However, if the data is biased the main class can be down-sampled or outliers can be deleted. Furthermore, if no experts are available to label the samples, crowd or weak labelling can be applied. If there is no possibility to label the samples at all, representation learning is suggested.

After the data transformation and pre-processing steps have been completed and depending on the data available a neural network has to be chosen. While for image classifications convolutional neural networks are the preferred option, for other tasks no specific model recommendation can be given. Though, research proposes to start with one model and then to test it against other models. The one with the best performance measured by validation accuracy, loss or cross entropy is preferable. Within the model the activation function has to be chosen. According to my research and the assessment of theory and practical applications the ReLu activation function was used the most. Furthermore, as optimization algorithm adaptive moment estimation is recommended, because it takes little memory, is robust and can be applied to models having many hyperparameters and samples. In addition, to increase the performance of a classifier, image augmentation techniques can be used and the number of performing epochs can be limited to save time and processing power. If data is scarce practical use-cases indicated that transfer learning can be used to leverage useful classifiers and enhance the performance of the model.

What is more, when the model is prone to overfitting, dropout can be used. Thus, units and layers are randomly dropped which significantly reduces overfitting and improves the performance of neural networks on supervised learning tasks. Then, the model needs to be validated. Preferable k-fold cross validation should be performed. Although this process is time consuming it spares the test set and therefore avoids withdrawing the test set, because usually it can only be used once. After the model had been validated and tested and depending on the validation as well as test accuracy, the classification or regression task either could be solved or the complete process has to be started again. Below a high-level instruction is presented.

1. Evaluate problem and check alternatives
2. Gather and organize data
3. Transform, pre-process, clean, visualize the data
4. Choose model dependent on data, optimization algorithm, activation function
5. Perform training and validate, optimize using dropout, transfer learning etc. and validate again, go back to step 3 if no success had been achieved
6. Hyperparameter tuning
7. Test against other models performance and choose the best one or go back to step 3 or 5

8.4 Conclusion

Neural networks are applicable for solving both regression and classification problems. Though, my theoretical research as well as the qualitative analysis of use-cases where neural networks in the manufacturing industry had been implemented indicate that the focus is more on classification rather than regression. Moreover, while for the use-cases in manufacturing images served as input for the model, our network received discrete data. Therefore, model choice or whether to use a convolutional neural network like it was the case for the manufacturing use-cases or a recurrent neural network to predict the on-time deliveries, is dependent on the type of data available. In other words, the thesis showed that neural networks can be applied to solve regression as well as classification problems in general and that this is not dependent on a specific industry but more on the data available.

Further, the success of implementing a neural network and therefore solving the underlying classification or regression problem is heavily dependent on various variables. Starting with the acquisition of data academic literature indicates that the performance of a neural network can be positively influenced by the quantity of labelled data available. Moreover, existing research states that no neural network can be trained without data. It should be stressed here that the development of our neural network would have not been possible without data. Another model performance influencing factor is the balance of the dataset. Not only this was an issue in one of the use-cases but also during the development of our neural network. Thus, we removed the outliers and down sampled the main class as proposed by theory. As a result, overfitting of our model in subsequent epochs was reduced. Moreover, current research has shown that the choice and the adjustment of the features can have positive effects on the performance of the model. This is also underpinned by the quantitative analysis of our developed neural network in which this effect was measured by comparing the initial performance of the model with the model’s performance after a feature was added. The result showed that adding a new feature increased the confidence level of the model for on-time deliveries from 71 to 82 percent.

Furthermore, the framework in chapter 6 originally derived by analyzing academic as well as real world use cases was used to solve a classification problem of predicting on-time deliveries in supply chain management. Although the framework was utilized during the development process, it served just as a guideline and can’t be considered as a specific handling instruction suitable for all neural network implementations. However, in chapter 7 it was shown that a neural network can be used to predict whether a delivery will be on-time or late with a cross-entropy of 0.36, a validation accuracy for on-time deliveries of 82 percent and for late deliveries of 71 percent. Nevertheless, in the near future the model will be optimized by reducing factors of variations. Further, other models solving the same tasks will be evaluated and compared. In addition, dropout will be utilized to reduce the risk of overfitting and to minimize the generalization error.

Bibliography

Ali, Shamsuddin, & Ralescu. (2015). Classification with class imbalance problem: A Review. *Int. J. Advance Soft Compu. Appl, Vol.7*, pp. 176 - 195.

Babbage. (2010). *History of Mr. Babbage's Calculating Engines. In H.P. Babbage (Ed.), Babbage's Calculating Engines: Being a Collection of Papers Relating to Them; Their History, and Construction (pp. 1-20).* New York: Cambridge University Press.

Bayes, ". E. (1763, January 1). *www.royalsocietypublishing.org.* Retrieved from https://royalsocietypublishing.org/doi/pdf/10.1098/rstl.1763.0053

Bishop. (2006). *Pattern Recognition and Machine Learning.* Cambridge: Springer.

Boole. (1853, November 30). *An investigation of the laws of thought on which are founded the mathematical theories of logic and probabilities.* Retrieved from http://www.gutenberg.org: http://www.gutenberg.org/files/15114/15114-pdf.pdf?session_id=8dfba09ab07e9b71661781e96eeafdf190e7d5aa

Bostrom. (2014). *Superintelligence.* Oxford, United Kingdom: Oxford University Press.

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

Chollet. (2017). *Deep Learning with Python.* USA: Manning Publications.

Cordts, Enzweiler, Omran, Benenson, Ramos, Franke, . . . Schiele. (2016). *The Cityscapes Dataset for Semantic Urban Scene Understanding.* Retrieved from https://www.cityscapes-dataset.com: https://www.cityscapes-dataset.com/wordpress/wp-content/papercite-data/pdf/cordts2016cityscapes.pdf

Department of Defense, Defense Science Board. (2016). *Report of the Defense Science Board Summer Study on Autonomy.* Washington, D.C.: Office of the Under Secretary of Defense for Acquistion, Technology and Logistics.

Ertel. (2017). *Introduction to Artificial Intelligence (2nd ed.).* London: Springer International Publishing AG.

Fawcett. (2005). An introduction to ROC analysis. *Pattern Recognition Letters 27 (2006)*, pp. 861 - 874.

Géron. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow.* Sebastopol, CA 95472: O'Reilly Media, Inc.

Goodfellow, Bengio, & Courville. (2016). *Deep Learning.* Cambridge, Massachusetts: The MIT Press.

Green, Whitten, & Inman. (2008). The impact of logistics performance on organizational performance in a supply chain context. *Supply Chain Management: An International Journal*, pp. 317 - 327.

Grosse. (2018). *CSC321: Intro to Neural Networks and Machine Learning.* Retrieved from University of Toronto Computer Science CSC321 Blockboard website:

http://www.cs.toronto.edu: http://www.cs.toronto.edu/~rgrosse/courses/csc321_2018/readings/L01%20Introduction.pdf

Grosse, Farahmand, & Carrasquilla. (2018, September). *CSC411: Intro to Neural Networks and Machine Learning.* Retrieved from University of Toronto Computer Science CSC411 Blockboard website: http://www.cs.toronto.edu: http://www.cs.toronto.edu/~rgrosse/courses/csc411_f18/

Hastie, Tibshirani, & Friedman. (2017). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Stanford: Springer.

Hinton. (1999). Supervised learning in multilayer neural networks. *The MIT Encyclopedia of the Cognitive Sciences. In Robert A. Wilson & Frank C. Keil (Eds.)*, pp. 814-815.

Hinton, & Nair. (2006). *Inferring Motor Programs from Images of Handwritten Digits.* Toronto, M5S 3G5 Canada: Department of Computer Science, University of Toronto.

Huh, Agrawal, & Efros. (2016). *What makes ImageNet good for transfer learning?* Berkeley: Berkeley Artificial Intelligence Research (BAIR) Laboratory UC Berkeley.

Küpper, Lorenz, Kuhlmann, Bouffault, Heng, Wyck, . . . Schlageter. (2018). *AI in the Factory of the Future, The Ghost in the Machine.* The Boston Consulting Group.

Khan, Jan, & Farman. (2019). *Deep Learning: Convergence to Big Data Analytics.* Peshawar: Springer Nature Singapore Pte Ltd.

Khumoyun, Cui, & Hanku. (2016, pp. 2-4). *Spark based distributed Deep Learning framework for Big Data applications. Paper presented at international conference on information science and communications Technologies (ICISCT), Tashkkent, Uzbekistan.* Seoul: Department of Internet&Mulitmedia Engineering, Konkuk University.

Kim, Hong, Roh, Cheon, & Park. (2016). *PVANET: Depp but Lightweight Neural Networks for Real-time Object Detection.* Seoul: Intel Imaging and Camera Technology.

Kingma, & Ba. (2015). Adam: A Method for Stochastic Optimization. *ICLR*, pp. 1 - 15.

Kotsiantis. (2007). Supervised Machine Learning: A Review of Classification Techniques. *Informatica 31*, pp. 249 - 268.

Kusiak. (2017, April 6). Smart manufacturing must embrace big data. *COMMENT*, pp. pp. 23-25.

LeCun, Haffner, Bottou, & Bengio. (2001). Gradient-Based Learning Applied to Document Recognition. *Intelligent Signal Processing*, pp. 306 - 351.

Lu, Li, Chen, Kim, & Serikawa. (2017). *Brain Intelligence: Go beyond Artificial Intelligence.* Springer Science+Business Media, LLC 2017.

Müller. (2016). *Introduction to Machine Learning with Python.* Sebastopol, CA 95472: O'Reilly Media, Inc.

McCarthy. (2007, p. 2, 11 12). Retrieved from http://www-formal.stanford.edu: http://www-formal.stanford.edu/jmc/whatisai.pdf, Accessed April 2019

McCarthy, Minsky, Rochester, & Shannon. (1955, p. 2, 8 31). *A Proposal for the Darthmouth Summer Research Project on Artificial Intelligence, Accessed April 2019.* Retrieved from http://jmc.stanford.edu: http://jmc.stanford.edu/articles/dartmouth/dartmouth.pdf

McCorduck. (2004). *Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence.* Natick, Massachusetts: A K Peters, Ltd.

Milgate. (2001). Supply chain complexity and delivery performance: an international exploratory study. *Supply Chain Management: An International Journal*, pp. 106 - 118.

Minsky, & Papert. (1969). *Perceptrons: An Introduction to Computational Geometry.* Massachusetts: Massachusetts Institute of Technology.

Narkhede. (2018, June 26). *Towards Data Science*. Retrieved from https://towardsdatascience.com: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Naval Postgraduate School. (2015, 3 11). *https://human-factors.arc.nasa.gov.* Retrieved from https://human-factors.arc.nasa.gov/workshop/autonomy/download/presentations/Shaddock%20.pdf, Accessed April 2019

Nilsson. (2010, p. 13). *The Quest for Artificial Intelligence: A History of Ideas and Achievements.* Cambridge: Cambridge University Press.

Pai, Hebbar, & Rodrigues. (2015). Impact of Delivery Delay on the Manufacturing Firm Inventories: A System Dynamics Approach. *Proceedings of the Business Management International Conference 2015*, pp. 38 - 45.

Pan. (2016, December 2016). Heading toward Artificial Intelligence 2.0. *Engineering*, pp. 409-412.

Pan, & Yang. (2010). A Survey on Transfer Learning. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 22, NO. 10*, pp. 1345 - 1356.

Ransbotham, Kiron, Gerbert, & Reeves. (2017). Reshaping Business With Artificial Intelligence. *MITSloan Management Review and The Boston Consulting Group*, p. 24.

Rendall, Castillo, Lu, Colegrove, Broadway, Chiang, & Reis. (2018). Image-based manufacturing analytics: Improving the accuracy of an industrial pellet classification system using deep neural networks. *Chemometrics and Intelligent Laboratory Systems*, pp. 26 - 35.

Ribeiro, Gong, Calivá, Swainson, Gudmundsson, Yu, . . . Kollias. (2018). *An End-To-End Deep Neural Architecture for Optical Character Verfication and Recognition in Retail*

Food Packaging. Lincoln: Machine Learning Group (MLearn), School of Computer Science, University of Lincoln.

Rittinghouse, & Ransome. (2009). *Cloud Computing: Implementation, Management and Security.* Boca Raton, FL 33487-2742: CRC Press, Taylor & Francis Group.

Roh, Heo, & Whang. (2018). A Survey on Data Collection for Machine Larning: a Big Data - AI Integration Perspective. *IEEE*, pp. 2 - 16.

Rosenblatt. (1958). The Perceptron: A Probabilistic Model dor Information Storage and Organization in the Brain. *Psychological Revie*, 386 - 407.

Russel, & Norvig. (2010). *Artificial Intelligence: A Modern Approach, Third Edition.* Upper Saddle River, New Jersey: Pearson Eductation Inc.

Samuel. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal*, pp. 211 - 229.

Shwartz, & David. (2014). *Understanding Machine Learning: From Theory to Algorithms.* New York: Cambridge University Press.

Simonyan, & Zisserman. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations, 2015* (pp. pp. 1 - 14). Oxford: Department of Engineering Science, University of Oxford.

Spiegeleire, D., Maas, & Sweijs. (2017). *Artificial Intelligence and the Future of Defense: Strategic Implications for Small- and Medium-Sized Force Providers.* The Hague: The Hague Centre for Strategic Studies (HCSS).

Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. In Y. Bengio (Eds.). *Journal of Machine Learning Research 15*, pp. 1929 - 1958.

Stamey, Kabalin, Mcneal, Joshstone, Freiha, Redwine, & Yang. (1989). Prostate Specific Antigen in the Diagnosis and Treatment of Adenocarcinoma of the Prostate. II. Radical Prostatectomy Treaded Patients. *The Journal of Urology*, pp. 1485 - 1486.

Stanford University. (2016). *Artificial Intelligence and Life in 2030, One Hundred Year Study on Artificial Intelligence.* Stanford: Stanford University.

Sutton, & Barto. (2018). *Reinforcement Learning.* Cambridge: MIT Press.

Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna. (2015). *Rethinking the Inception Architecture for Computer Vision.* London: University College London.

Turing. (1950, October). *Computing Machinery and Intelligence.* Retrieved from http://phil415.pbworks.com: http://phil415.pbworks.com/f/TuringComputing.pdf

University of Toronto. (2018). *Machine Learning in Computer Vision*. Retrieved from http://www.cs.utoronto.ca/~fidler/teaching/2018/CSC2548.html#information

Wang, Cui, Wang, Xiao, & Jiang. (2018). Machine Learning for Networking: Workflow, Advances and Opportunities. *IEEE Network*, pp. 92 - 99.

Yin, Yin, Huang, & Hao. (2013). Robst Text Detection in Natural Scene Images. *IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 36, no. 5*, pp. 970 - 983.

Zhang, Lipton, Li, & Smola. (2019). *Dive into Deep Learning.* Retrieved from UC Berkeley STAT 157 Blackboard website: https://courses.d2l.ai: https://en.d2l.ai/d2l-en.pdf

Zhou, Yao, Wen, Wang, Zhou, He, & Liang. (2017). *EAST: An Efficient and Accurate Scene Text Detector.* Beijing, China: Megvii Technology Inc.