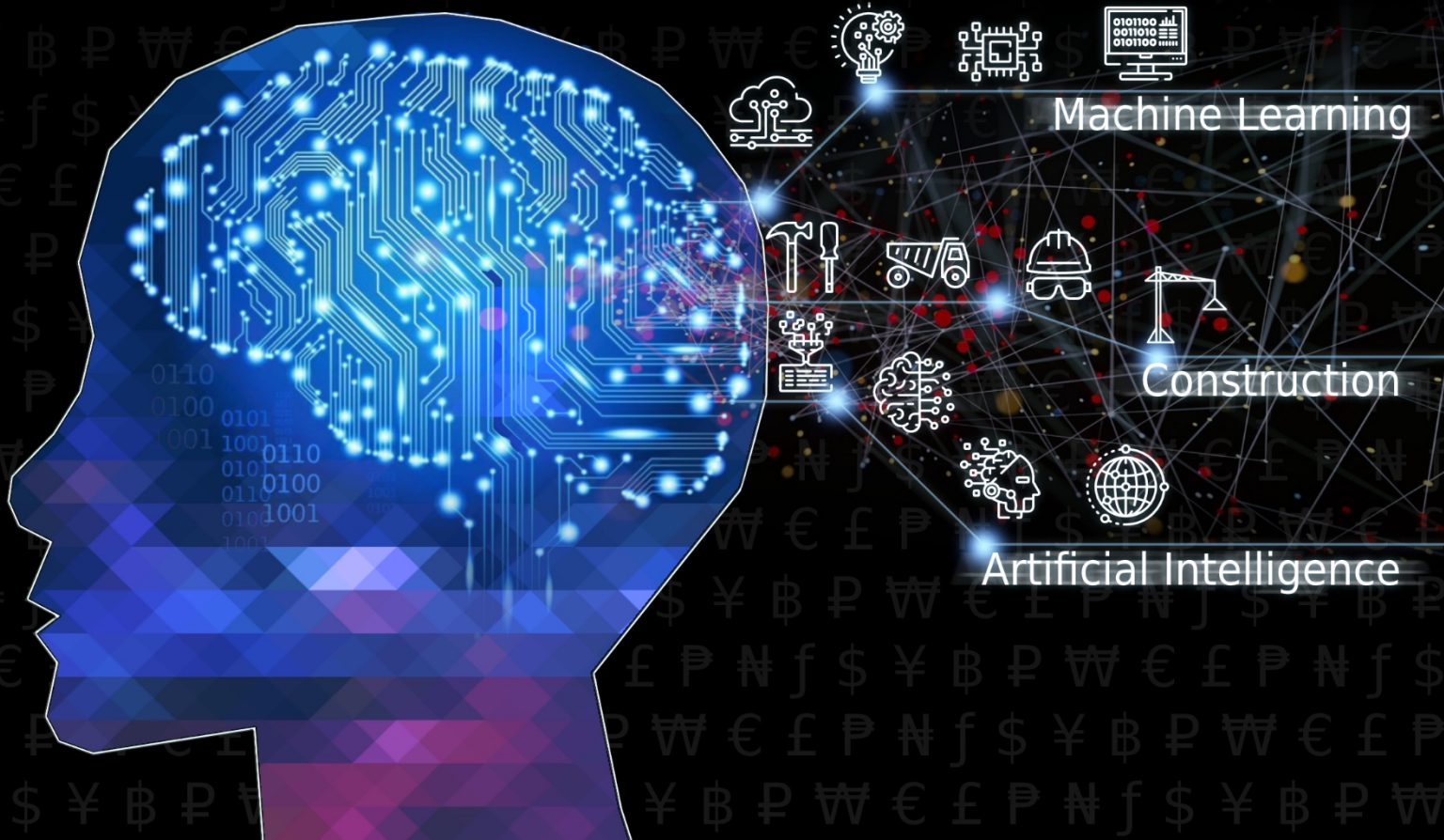
Construction cost estimating with Artificial Neural Networks

Utilising ANNs for conceptual construction cost estimation

Master thesis for Construction Management and Engineering MSc by:

B. (Bas) Pannekoek
CME cohort 2019-2020

Eindhoven University of Technology



Colophon

Title: Construction cost estimating with Artificial Neural Networks

Subtitle: Utilising ANNs for conceptual construction cost estimation

Author:	B. (Bas) Pannekoek
Student ID:	1273175
Email:	pannekoekbas@gmail.com
University:	Eindhoven University of Technology
Graduation Program:	Master Construction Management and Engineering
Track:	

Thesis defence date:	22-06-2020
Report date:	17-06-2020
Version:	1.0
Place:	Eindhoven, the Netherlands

Graduation committee:
Eindhoven University of Technology
Prof. dr. ir. B. D. (Bauke) de Vries
University supervisor | Chairman graduation committee
b.d.vries@tue.nl

Dr. P. (Pieter) Pauwels
University supervisor | 1st supervisor
p.pauwels@tue.nl

Prof. dr. T.A. (Theo) Arentze
University supervisor | 2nd supervisor
t.a.arentze@tue.nl

Graduation company – Stam + De Koning Bouw
Ir. S. (Stijn) van Schaijk
Company supervisor | external advisor
svschaijk@volkerwessels.com

Foreword

I am proud to present this thesis as the final product of my graduation project. Finishing this thesis is my final step towards becoming a Master of Science in the field of Construction Management and Engineering (CME). This graduation thesis is written to complete the MSc program CME at Eindhoven University of Technology (TU/e). The study for this gradation project was done in collaboration with TU/e and Stam + De Koning Bouw. I look back at a graduation period with ups and downs, hard and dedicated work, and especially as a period in which I learned a lot!

Without the advice, support, and collaboration of others, I could not have completed my graduation project. Therefore, I would like to thank the people who helped me obtain my MSc degree. First, I want to thank Pieter Pauwels and Theo Arentze for supervising me during this graduation project. Their guidance, advice, and support helped me in achieving the best results. Secondly, I would like to thank my colleague from Stam + De Koning Bouw, Stijn van Schaijk, who supervised during this project. His experience, vision, support, and guidance really helped me out during this graduation project. Thirdly, I would like to thank Stam + De Koning Bouw for providing me the opportunity, trust, space, and data for successful conducting my graduation research.

Finally, I would like to extend special thanks to my family, girlfriend, and friends, who always supported me during my long period of studying in the construction sector. Their support, advice, and guidance really supported me in achieving an MSc title in the field of CME.

B. (Bas) Pannekoek
Eindhoven, June 2020

Table of contents

Abstract

Keywords: Artificial Neural Networks, Conceptual Cost Estimating, Machine Learning

---

Abstract: Cost estimation in the early phases of a construction project has been difficult for a very long time. Achieving a higher accuracy by using data analytics for early cost estimations would benefit construction firms. In construction management, artificial neural networks (ANNs) are an often used method for cost estimation. This research will study the possibility of using NNs to perform cost estimations for construction projects with the data available within one single firm and whether these estimations deliver a high enough level of accuracy. Theoretically, it has already been proven that cost estimations with NNs appear to be promising for construction cost estimation with highly accurate results. However, most of the studies concerning cost estimation with NNs do not result in practical implementation. Due to the supervised learning, an NN is able to generalise knowledge and learn from examples, which is useful for cost estimations, as they are mostly based on previous cases. For this study a Feedforward backpropagation (FFBP) NN was created. The developed NN was trained on 35 data samples and tested with 15 data samples. An NN can be used for (conceptual) construction cost estimating. However, during this study, it was found that the NN did not yet achieve the desired accuracy level of ≥ 80%. The best performing NN in this study reached an estimation accuracy of 69%. It can be concluded from the results that the developed NN did not achieve the desired accuracy level. However, note that for this study the NN was developed with a limited availability of data. Also, some of the data has been simulated. If the same NN was trained with a higher amount of high-quality actual project cost data, the average accuracy, as well as the accuracy of the conceptual cost estimations made with the NN estimation model, would probably increase. Despite that the desired accuracy level was not reached during the study, the study still contributes to the practical adoption and application of AI and ML in construction. Results of the study were also presented to the management and cost planners of the graduation company involved during the study. They see added value in the use of NNs for conceptual cost estimation in the future.

Summary

Cost estimation in the early phases of a construction project has been difficult for a very long time. During those phases of a construction project, cost estimation typically needs to be based on rules of thumb and the overall knowledge of the project planners. Achieving a higher accuracy by using data analytics for early cost estimations would benefit construction firms, since the accuracy of expert cost estimations varies from -15% to +25% at the conceptual stage of a project. In construction management, artificial neural networks (ANNs), also known as neural networks (NN), are often used to solve numerous problems ranging from tender price prediction, construction cost estimation, project cashflow, risk quantification etc. It can be derived from literature that a predictive data mining technique such as NNs can provide promising results for construction cost estimation. One of the advantages of applying NNs for cost estimating is that a shorter timeframe is required to deliver a conceptual cost estimation. NNs can also be more accurate compared to traditional methods. Accuracy levels of NN-based prediction and forecasting models can get as high as 98%. This research will study the possibility of using NNs to perform conceptual cost estimations for construction projects with the data available within one single Dutch construction firm and whether these estimations deliver a high enough level of accuracy. Theoretically, it has already been proven that cost estimations with NNs appear to be promising for construction cost estimation with highly accurate results. However, most of the studies concerning cost estimation with NNs do not result in practical implementation. That many ML models do not result in practical implementation is not because the ML model was not a success, but because of a lack of capacity in the available data. However, the more data is used, the less the functionality of an machine learning (ML) model. It is therefore important that an AI model is designed based on criteria for its specific application. This study aims at answering the following main research question:

"How can cost estimations with ANNs trained with limited project cost data create more accurate cost estimations for construction projects during the early phases of the construction process?"

NNs are considered one of the main ML categories that can be utilised in combination with a supervised learning technique for regression problems. Due to the supervised learning, an NN is able to generalise knowledge and learn from examples, which is useful for cost estimations, as they are mostly based on previous cases. Before an NN can be utilised for estimations, the NN needs to be constructed. However, constructing an NN is a non-trivial task as decisions need to be made about the NN architecture, activation functions, and loss functions. When the NN is constructed, it can be trained, after which it can be utilised for estimations. The literature showed that NNs can be applied for a wide variety of estimation tasks in the construction sector, ranging from productivity estimations to the prediction of construction costs. The literature concerning construction cost estimation with NNs indicates that, even though they use small data samples, NNs still perform superiorly for construction cost estimation tasks. These findings in the literature support the objective of this study in order to estimate construction costs with NNs, even when data availability is small.

To develop an NN to perform conceptual cost estimations, six basic steps need to be followed and carried out in three phases:

Phase I: Modeling phase

1. Problem definition
2. Data gathering and representing
3. Defining the network

Phase II: Implementation phase

4. Structuring the network
5. Training and testing the network

Phase III: Application phase

6. Applying the network for estimations

By following the phases and steps, an FFBP NN was created. The constructed NN uses leakyReLU as activation function and backpropagation as learning algorithm. In order to perform the cost estimations, the developed NN will be trained on 35 data samples and subsequently, its performance will be tested with 15 data samples. During testing, the NN will be utilised to make construction cost predictions based on unseen data. Estimates made with the NN are considered accurate, as they fall within a range of 20% of the actual construction costs. This means that the developed NN needs to achieve an average accuracy of at least 80% to be considered accurate.

An NN can be used for (conceptual) construction cost estimating, even when only limited data is available. However, during this study, it was found that the NN did not yet achieve the desired accuracy level of ≥ 80%. In this study, the NN has been developed with the PyTorch ML library and was trained with 35 simulated cases. During training, 19 different NN architectures were considered and two different loss criterions were applied to the NN. After training, it was found that an NN with an architecture of 9-19-1, trained with the SL1 Loss criterion, performed best when trained over 1000 epochs. However, during testing this NN reached only an average estimation accuracy of 48%. By retraining the best performing NN with 26 data samples in which outlier cases were removed. In addition, retesting this retrained NN on only those 5 data samples of type of project category 1 (i.e. residential housing projects), its performance has improved to an average estimation accuracy of 69%. This average estimation accuracy level is not high enough to consider the model accurate and useful for conceptual cost estimations, as the average estimation accuracy is ≤ 80%.

It can be concluded from the results of this study that the developed NN did not achieve the desired accuracy level of ≥ 80%. However, note that for this study the NN was developed with a limited availability of data. Also, the data has been simulated for NN training in order to use more data for the study. If the same NN was trained with a higher amount of high-quality actual project cost data, the average accuracy, as well as the accuracy of the conceptual cost estimations made with the NN estimation model, would probably increase. For this study, the following hypothesis was drafted:

“If ANNs trained with a limited amount of project cost data can be used for cost estimations of new projects made in the early phases of the construction process, the cost estimations of construction projects can be made more accurate.”

This hypothesis has not been proven, based on the following arguments:

- The NN that is developed, trained, and tested in this study upon limited data (i.e. 26 data samples for training and 15 data samples for testing) reached an accuracy of 69% which is 11% points less than the desired accuracy level of ≥ 80% required for ‘accurate’ conceptual cost estimations.
- The NN in this study was developed by using simulated data which probably does not represent actual construction project cases. Thus, the results of this study were achieved with simulated data to validate the method and to prove the concept of NNs. Therefore, results of the study would probably differ if only actual project cost data has been used.

Despite that the desired accuracy level was not reached during the study, the study still contributes to the practical adoption and application of AI and ML in construction. Results of the study showed that NNs can also be used when there is only a limited amount of qualitative data available for NN development. Results of the study were also presented to the management and cost planners of the graduation company involved during the study. They received the results enthusiastically, they also see added value in the use of NNs for conceptual cost estimates, especially if the NN can be used to validate whether the calculated costs of a project are in line with the costs of past similar projects.

Samenvatting

De kostenraming in de beginfase van een bouwproject is al heel lang moeilijk. Tijdens deze fase wordt de kostenraming meestal gebaseerd op vuistregels en de algemene kennis van de project planners. Het bereiken van een hogere nauwkeurigheid door het gebruik van gegevensanalyses, die nu varieert van -15% tot +25%, zou bouwbedrijven ten goede komen.

In bouwmanagement worden kunstmatige neurale netwerken (ANN), ook wel bekend als neurale netwerken (NN), gebruikt om problemen op te lossen zoals het voorspellen van de prijs van een aanbesteding, het schatten van de bouwkosten, de cashflow van een project en de kwantificering van risico's. Uit literatuur kan worden afgeleid dat een voorspellende dataminingtechniek zoals NN veelbelovende resultaten kan opleveren: een kortere termijn om een conceptuele kostenraming te maken en NN kunnen ook nauwkeuriger zijn dan traditionele methoden. Nauwkeurigheidsniveaus van op NN gebaseerde voorspellings- en prognosemodellen kunnen oplopen tot 98%.

Dat veel machine learning (ML) modellen niet leidden tot praktische implementatie is niet omdat het ML-model geen succes was, maar vanwege een gebrek aan capaciteit in de beschikbare gegevens. Hoe meer gegevens er worden gebruikt, hoe minder de functionaliteit van een ML-model. Het is daarom belangrijk dat een AI-model wordt ontworpen op basis van criteria voor de specifieke toepassing ervan. Deze studie beoogt een antwoord te geven op de volgende onderzoeksvraag:

"Hoe kunnen kostenramingen met ANN die zijn getraind met beperkte projectkostengegevens nauwkeurige kostenramingen maken voor bouwprojecten tijdens de eerste fasen van het bouwproces?

NN worden beschouwd als een van de belangrijkste ML-categorieën die kunnen worden gebruikt in combinatie met een gecontroleerde leertechniek voor regressieproblemen. Door het gecontroleerde leren is een NN in staat om kennis te veralgemenen en te leren van voorbeelden, wat nuttig is voor kostenramingen, omdat deze meestal gebaseerd zijn op vergelijkbare voorbeelden. Voordat een NN kan worden gebruikt voor schattingen, moet het NN worden samengesteld. Het samenstellen van een NN is echter een belangrijke en moeilijke taak omdat er beslissingen moeten worden genomen over de NN-architectuur, activeringsfuncties en verliesfuncties. Als het NN is samengesteld, kan het worden getraind, waarna het kan worden gebruikt voor schattingen. Uit de literatuur blijkt dat NN kunnen worden toegepast voor een breed scala aan schattingsopdrachten in de bouwsector, variërend van productiviteitsschattingen tot de voorspelling van bouwkosten. De literatuur over de schatting van de bouwkosten met NN geeft aan dat NN, ook al gebruiken ze kleine datamonsters, nog steeds superieur presteren. Deze bevindingen in de literatuur ondersteunen de doelstelling van deze studie om de bouwkosten met NN in te schatten, ook als er weinig gegevens beschikbaar zijn.

Om een NN te ontwikkelen voor het maken van conceptuele kostenramingen moeten zes basisstappen worden gevolgd en uitgevoerd in drie fasen:

Fase I: Modelleringsfase

1. Probleemstelling
2. Gegevens verzamelen
3. Definiëren van het netwerk

Fase II: Uitvoeringsfase

4. Structureren van het netwerk
5. Trainen en testen van het netwerk

Fase III: Toepassingsfase

6. Toepassen van het netwerk voor schattingen

Door het volgen van de fasen en stappen is een feedforward backprogation (FFBP) NN samengesteld. Het geconstrueerde NN gebruikt leakyReLU als activeringsfunctie en backpropagatie als leeralgoritme. Om de kostenramingen uit te voeren wordt het ontwikkelde NN getraind op 35 datamonsters en vervolgens worden de prestaties getest met 15 datamonsters. Tijdens het testen zal het NN worden gebruikt om bouwkosten te voorspellen op basis van ongeziene gegevens. Schattingen die met het NN worden gemaakt, worden als nauwkeurig beschouwd wanneer ze binnen een bandbreedte van 20% van de werkelijke bouwkosten vallen. Dit betekent dat het ontwikkelde NN een gemiddelde nauwkeurigheid van ten minste 80% moet bereiken.

Een NN kan worden gebruikt voor (conceptuele) bouwkostenramingen, ook als er maar beperkte gegevens beschikbaar zijn. Tijdens dit onderzoek is gebleken dat het NN nog niet het gewenste nauwkeurigheidsniveau van ≥ 80% heeft bereikt. Het NN is ontwikkeld met de PyTorch ML bibliotheek en getraind met 35 gesimuleerde datasets. Tijdens de training werden 19 verschillende NN-architecturen overwogen en werden twee verschillende verliescriteria toegepast op het NN. Na de training bleek dat een NN met een architectuur van 9-19-1, getraind met het SL1 verliescriterium, het beste presteerde bij het trainen van het netwerk over 1000 tijdperken (epochs). Tijdens het testen van dit NN werd slechts een gemiddelde schattingsnauwkeurigheid van 48% bereikt. Om de schattingsnauwkeurigheid van het NN te verbeteren, werd besloten het best presterende NN opnieuw te trainen met 26 datamonsters waarin de uitschieters zijn verwijderd. Daaropvolgend zijn de prestaties van dit omgeschoolde NN alleen op die 5 gegevenssteekproeven van het type projectcategorie 1 (d.w.z. woonprojecten) getest, waardoor de nauwkeurigheid verbeterde tot een gemiddelde van 69%. Dit gemiddelde nauwkeurigheidsniveau is niet hoog genoeg (≥80%) om het NN als nauwkeurig en nuttig te beschouwen voor conceptuele kostenramingen. Uit de resultaten van dit onderzoek kan worden geconcludeerd dat het ontwikkelde NN niet het gewenste nauwkeurigheidsniveau van ≥ 80% heeft bereikt. Merk echter op dat het NN voor deze studie is ontwikkeld met een beperkte beschikbaarheid van gegevens. Ook zijn gegevens gesimuleerd voor NN-training, om meer gegevens te kunnen gebruiken voor het onderzoek. Als hetzelfde NN zou worden getraind met een hogere hoeveelheid hoogwaardige feitelijke projectkostengegevens, zou de gemiddelde nauwkeurigheid, evenals de nauwkeurigheid van de conceptuele kostenramingen (die met het NN-ramingsmodel worden gemaakt) waarschijnlijk toenemen. Voor deze studie is de volgende hypothese opgesteld:

"Als ANN, die getraind zijn met een beperkte hoeveelheid projectkostengegevens, worden gebruikt voor kostenramingen van nieuwe projecten, dan worden de kostenramingen van deze projecten nauwkeuriger".

Deze hypothese is niet bewezen op basis van de volgende argumenten:

- Het NN dat in dit onderzoek is ontwikkeld, getraind en getest op beperkte gegevens (d.w.z. 26 datamonsters voor training en 15 datamonsters voor testen) bereikte een nauwkeurigheid van 69%, wat 11% punten minder is dan het gewenste nauwkeurigheidsniveau van ≥ 80% dat vereist is voor 'nauwkeurige' conceptuele kostenramingen.
- Het NN in deze studie is ontwikkeld door gebruik te maken van gesimuleerde gegevens die waarschijnlijk niet representatief zijn voor werkelijke bouwprojecten. De resultaten zullen waarschijnlijk verschillen als alleen feitelijke projectkostengegevens worden gebruikt.

Ondanks dat het gewenste nauwkeurigheidsniveau tijdens het onderzoek niet werd bereikt, draagt het onderzoek toch bij aan de praktische toepassing van AI en ML in de bouw. De resultaten van de studie tonen aan dat NN ook kunnen worden gebruikt wanneer er slechts een beperkte hoeveelheid kwalitatieve gegevens beschikbaar zijn. De resultaten van het onderzoek zijn gepresenteerd aan het management en de calculators van het betrokken afstudeerbedrijf tijdens het onderzoek. Ze hebben de resultaten enthousiast ontvangen en zien toegevoegde waarde in het gebruik van NN voor conceptuele kostenramingen.

List of abbreviations

AEC	Architecture, Engineering and Construction
ANN	Artificial Neural Networks
APE	Absolute Percent Error
BP	Backpropagation
CME	Construction Management and Engineering
FFBP	Feedforward Backpropagation
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MSE	Mean Squared Error
NN	Neural Network

List of figures

List of tables

1. Introduction

In this chapter, the research will be introduced. It will be introduced by first discussing the research motive, background, and context. Subsequently, the research problem and the research design will be described. Finally, at the end of this chapter, a reading guide for the remainder of the thesis will be provided.

1.1 Research introduction

In this paragraph, the presented research is introduced by discussing the research motive, background, and context.

1.1.1 Motive

Nowadays, large volumes of data are generated by the architecture, engineering, and construction (AEC) industry. To gain competitive advantage and increase efficiency, businesses could use data analytics for improved decision-making (Fayyad et al., 1996; Hoogeveen, 2015). Hovnanian et al. (2019) argue that data analytics can uncover critical insights that both speed up and improve the quality of management decisions. Data analysis tools can provide significant benefits to companies, since they allow them to quickly and continuously analyse project data and assess progress. This enables project managers to react faster to potential problems and make better decisions (Ismail et al., 2018).

Cost estimation in the early phases of a construction project has been difficult for a very long time. For those phases of a construction project, cost estimation typically needs to be based on rules of thumb and overall knowledge among the project planners. Achieving a higher accuracy using data analytics for early cost estimations would benefit construction firms, since the accuracy of expert cost estimations varies from -15% to +25% at the conceptual stage of a project (Hoogeveen, 2015; Rekveld, 2017). According to Phaobunjong (2002), the expected accuracy of a definitive estimate for a construction project is within a range of -5% to +15% and the range of a conceptual stage estimate should be within -30% to +50% of the final project cost. AbouRizk et al. (2002), however, state that a conceptual cost estimation for a construction project should be within a range of -15% to +25% of the final project cost. With the advent of emerging data analysis tools, this situation may be amended. Numerous attempts with data analysis tools are already being made to improve the accuracy of (conceptual) construction cost estimation (Zima, 2015).

In the AEC industry, a variety of tools (e.g. Revit, Solibri Office, Trimble Connect, BIMcollab, Synchro etc.) are used to support daily practices. A beneficial by-product of using these tools is that large amounts of data are stored in a structured way. Large amounts of data have the potential to be useful for data analytics. It becomes clear from conversations held with a Dutch construction firm that it recognises the potential of data analytics. They already capture a variety of data in different ways, however, questions such as ‘do we have enough data available for data analytics?’, ‘what can we do with the captured data?’, and ‘how can we learn from it?’, are still unanswered. One of the potential ideas to arise within the company is using (historical) project cost data for cost estimation in the early phases of a construction project.

Knowledge Discovery in Databases (KDD) received major attention over the last years. KDD can assist in data analysis and inference of knowledge. But to derive knowledge from data, data mining algorithms need to be applied to the available data. Hand et al. (2001) defines data mining as “the analysis of large observational datasets to find unsuspected relationships and summarize the data in novel ways so that data owners can fully understand and make use of the data.”

In construction management, artificial neural networks (ANNs) are often used as a data mining technique that solves numerous problems ranging from tender price prediction, construction cost estimation, project cashflow, risk quantification, etc. (Waziri et al., 2017). It can be derived from the

literature that a predictive data mining technique, such as the ANN, can be used for construction cost estimation, showing promising results (Waziri et al., 2017; Elmousalami, 2019). One of the advantages of applying NNs for cost estimating is it requires a shorter timeframe for the delivery of conceptual cost estimation (Juszczyk, 2017). Also, NNs can be more accurate than traditional methods: the accuracy levels of NN-based prediction and forecasting models can get as high as 98% (Waziri et al., 2017). This research will provide insight into whether it is possible to use NNs for conceptual cost estimations of construction projects with the data available within one single Dutch construction firm, and whether these estimations deliver a high enough level of accuracy. Note that the data available at the construction firms is probably insufficient to fully benefit from machine learning abilities such as NNs. However, it will still be interesting to study whether NNs can be used for estimations when limited data is available.

1.1.2 Background

A far larger amount of data, structured and unstructured, is currently available for the built environment. Still, decision-making in the AEC industry relies on previous experience and rules of thumb (Petrova et al., 2018; Petrova et al., 2019) and not on data driven decision-making that contains documented evidence. This current way of decision-making may lead to inaccurate assumptions regarding input parameters used for different decisions and predictions (Petrova et al., 2018).

Artificial Intelligence (AI) is a scientific discipline that studies how machines and algorithms can exhibit intelligent behaviour and it plays an important role in Data Science. AI consists of two main approaches: statistical and symbolic (Minsky, 1991). Symbolic AI resembles human cognitive behaviour and is useful for representing theories or scientific laws in a way that is meaningful to the symbol system and can be meaningful to humans. With statistical AI, intelligence is taken as an emergent property of a system and intelligent behaviour is commonly formulated as an optimization problem (Hoehndorf & Queralt-Rosinach, 2017). Statistical AI approaches are useful in learning patterns or regularities from data. The uptake in the application of statistical AI in recent years is supported by advancements in computational power, data storage, and parallelization in combination to methodological advances in applying machine learning (ML) algorithms and solving optimizations problems (Hoehndorf & Queralt-Rosinach, 2017). These days symbolic AI is less popular and often forgotten, although it is still valid.

As mentioned, there are two types of AI, namely symbolic and statistical. In the construction sector, BIM data can be seen as an example of a part of symbolic AI because it contains a lot of semantics. BIM data expressed in IFCs is rigid and structured, and therefore applying ML techniques to this data makes little to no sense. On the other hand, statistical AI is about data without semantics (e.g. big data, sensor values, Facebook likes, etc.), it is rough unstructured data. ML techniques can then be used to structure this data. Thus, using it directly as input source for NNs cannot be done directly because BIM, with its semantics and structured data, belongs to the symbolic AI. On the other hand, NNs and other ML techniques need unstructured non-semantic data, and thus belong to statistical AI. However, it is possible to use BIM data as the input source when information (i.e. knowledge) is derived from BIM models.

KDD is the overall process of useful knowledge extraction. Fayyad et al. (1996) mention that KDD has evolved and continues to evolve from the intersection between different research fields such as machine learning, databases, statistics, artificial intelligence, knowledge acquisition for expert systems, and high-performance computing. In their article, they state that the KDD process is interactive and iterative, involving numerous steps. One of the steps that needs to be taken in the KDD process is data mining. Petrova et al. (2018) explain the relationship between KDD and data mining as follows: “KDD represents the overall process of knowledge extraction, with knowledge being the end product of the data-driven discovery and data mining being the step in the process which employs

specific algorithms to discover patterns in the given data." This study will only focus on the data mining step of the KDD process in which NNs are applied. Data mining is the essential step in the KDD process that needs to be performed to transform the available raw data into actionable knowledge and insights of immediate value to the end user (Petrova et al., 2019).

1.1.3 Context

According to Fayyad et al. (1996) and Han et al. (2012), examples of widely accepted data mining categories are classification, clustering, association rule mining, regression, summarization and anomaly detection. This targets either predictive (supervised, directed) or descriptive (unsupervised, undirected) analytics. A supervised approach (i.e. predictive) relies upon domain expertise and significant amounts of training data. It describes the qualitative or quantitative relationships between input and output. Due to the predefined inputs and outputs, the discovery of novel knowledge is unlikely. In unsupervised approaches (i.e. descriptive), inputs and outputs are not predefined and it does not rely upon training data. Therefore, unsupervised approaches excel in discovering the intrinsic structure, correlations and associations in data. Due to their predefined target, predictive techniques are backward-oriented while descriptive ones are forward-oriented, making it possible to discover interesting patterns and relationships in the data (Petrova et al. 2018). When people talk about ML, they generally refer to supervised ML. Domingos (2015) defines supervised ML as follows: "every algorithm has an input and output: the data goes into the computer, the algorithm does what it will with it, and out comes the result. […] Machine Learning turns this around: in goes the data and the desired result and out comes the algorithm that turns one into the other."

Hilgers (2018) gives an overview of different traditional and ML algorithms (Figure 1) that could be used for automated valuation models (AVM). Traditional algorithms were mostly used for AVMs. In recent years, however, a shift has occurred in the application of ML algorithms, as they have proven to be hard to beat in terms of prediction accuracy[1]. It is important to realise that AI and ML are not merely connected, but in fact represent different concepts. According to John McCarty (1956), AI involves machines that can perform tasks that are characteristic of human intelligence. ML on the other hand is a simple way of achieving AI. One could obtain AI without ML, although this would require an unbelievable amount of complex code writing (Hilgers, 2018).

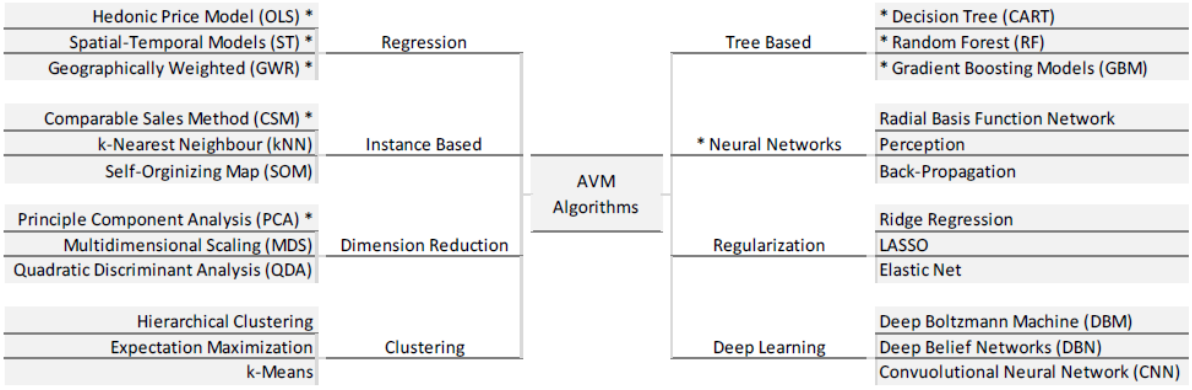


Figure 1: overview of automated valuation model algorithms (Hilgers, 2018)

Neural networks (NNs), or artificial neural networks (ANNs), are a wide class of flexible nonlinear regression and discriminant models, data reduction models, and nonlinear dynamical systems (Sarle, 1994). NNs are extensively used for prediction and research shows that NNs seem suitable for non-parametric cost estimation, especially for early estimates (Kim et al., 2004; Juszcyk, 2017; Elfaham, 2019). Their ability to learn, generalize knowledge, and their predictive capabilities make them a suitable cost estimation tool (Juszczyk, 2017). Also, an NN eliminates the need to find a cost estimating

[1] https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

relationship that describes the cost of a system as a function of the variables that have the most effect on the cost of that system (Kim et al., 2004). However, the performance of NNs strongly depends on the quality and the quantity of examples (Günaydin et al., 2014; Rekveld, 2017). The more examples one has, the less prediction error occurs. Thus, to create a good and accurate prediction model with NNs, there is a need for qualitative, reliable, full-scale (cost) data for projects of various types and conditions. This does not include BIM data in the form of IFCs. Data found in IFCs are way too semantically rigid, therefore it is not possible to directly build NNs from IFCs. However, it is possible to analyse the (BIM) data in IFCs (e.g. with regular code, simple annotation, or information take-offs), and then use the resulting tags with cost data for NNs. An important side note is that this data must be quite similar, otherwise prediction is not possible or not accurate (Rekveld, 2017). The data availability within a single construction firm in the AEC industry is limited and probably not large enough to fulfil the data capacity conditions to utilise ML algorithms (i.e. NNs) to its full potential. However, it would still be interesting to know whether NNs can be utilised when data is available in limited amounts.

NNs consist of “neurons” i.e. simple linear or nonlinear computing elements, interconnected in complex ways and organised in layers (Sarle, 1994; Zhang et al. 1998). The data analysis method is one of the main ways in which an NN is used (Sarle, 1994). Similar to statistical methods, NNs are capable of processing large amounts of data and making predictions that can be surprisingly accurate. However, an NN cannot perform instantly, it needs to be trained to perform the desired task (Zhang et al., 1998; Elfaham, 2019). Training can be seen as the process of determining the arc (link) weights, which are the key characteristics of an NN. The network stores the knowledge it has learned in the arcs. These arcs form the links from input nodes to output nodes, and are constructed in such a way that complex nonlinear mappings can be performed (Zhang et al., 1998). The most applied algorithm used for training is the back propagation (BP) algorithm (Zhang et al., 1998; Kim et al., 2004; Waziri et al., 2017). Werbos (1988) found that NNs trained with backpropagation outperform traditional statistical methods such as regression.

Constructing an NN is a non-trivial task. Decisions need to be made about the number of layers, subsequently the number of nodes in each layer needs to be determined, and finally the number of arcs that interconnects the nodes needs to be determined. The most applied manner of constructing NN forecasters is using the multi-layer perceptron (MLP) model, shown in Figure 2.

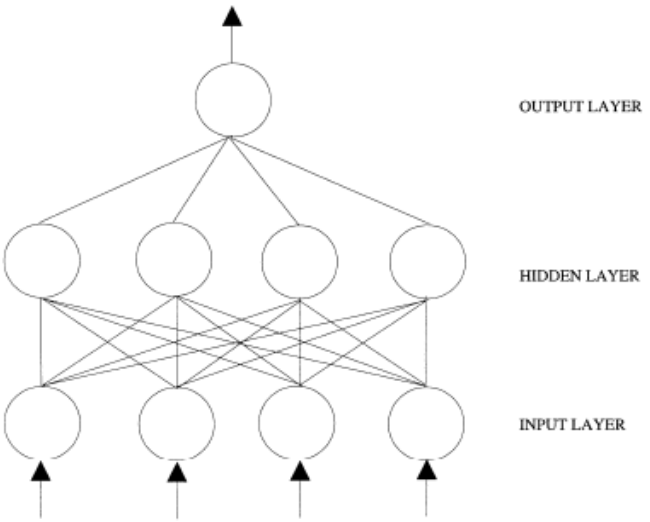


Figure 2: a typical MLP Neural Network (Zhang et al., 1998)

Construction cost estimation with NNs has been subject to a number of studies (Waziri et al., 2017; Elmousalami, 2019). Features such as adaptive learning, self-organisation, real time operation and fault tolerance are advantages of NNs over traditional statistical tools. Despite these advantages, an NN barely offers an explanation on the relationships between the parameters used for modelling. This

makes learning from NNs difficult (Waziri et al., 2017). Neural networks are therefore considered a black box technique. Also, the knowledge acquisition process is very time consuming (Kim et al., 2004; Waziri et al., 2017). The lack of transparency is a major drawback of AI methods. Therefore, the development of methods for visualising, explaining and interpreting deep learning models has recently attracted increased attention. Explainability in AI systems is important for the verification of AI systems and to improve and learn from AI systems, but also to see if AI systems are in compliance with legislations (Samek et al., 2017). Explainable AI systems are known under the name of explainable artificial intelligence (XAI) and fall outside the scope of the present study.

Waziri et al. (2017) reviewed over 100 studies published in refereed journals and conference proceedings, and found that NNs are recognized as more powerful than mathematical and statistical methods in the event of qualitative and quantitative reasoning. The literature shows that the research domain of applying NNs for nonparametric construction cost estimating in construction is mainly focused on conceptual or the early stage estimates of a project (Juszcyk, 2017). Due to the advantages of NNs and insights gained from previous studies, the current study will focus on applying NNs for cost estimation during the early phases of a construction project. Due to concerns regarding the data availability, it might be that a limited dataset will be used for the development, training, and testing of the NN estimation model. However, it will still be interesting to see whether NNs can be utilised for cost estimations with limited datasets, since data availability is limited in the AEC industry especially at a single construction firm.

1.2 Research problem

1.2.1 Problem analysis and research objectives

Severe budget overruns and delays, especially in larger projects, are still common in the construction sector (Klakegg & Lichtenberg, 2016). Flyvbjerg et al. (2002, 2003) studied the reasons behind cost overruns and argues that the main problem can often be found in deliberately underestimating costs and risks. Merrow (2011) also emphasizes that inadequate cost estimations and risk assessments are the reasons behind cost overruns.

Theoretically, it has already been proven that cost estimations with NNs are a promising approach for construction cost estimation with highly accurate results. However, most of the studies concerning cost estimation with NNs do not result in practical implementation (Rekveld, 2017). This is not because the ML model was not a success, but because of a lack of capacity in the available data. However, the more data is used, the less the workability of an ML model. It is therefore important that an AI model is designed based on criteria for its specific application (Computerworld, 2019). Thus, practically performing cost estimations with NNs is a challenging task since the performance of NNs is heavily dependent on the quantity and quality of the dataset (Günaydin et al., 2004). Therefore, the objective of this research is broken down into two parts, a theoretical and a practical target:

The theoretical objective is:
"To improve the accuracy of cost estimations during the early phases of a construction process by using actual project cost data and ANNs."

The practical target is:
"To determine to what extent ANNs trained with actual project cost data can be used for cost estimation during the early phases of the construction process, given that data needed for the development of the ANN is available in limited amounts."

Based on the theoretical and practical target, a hypothesis for this study is drafted:
"If ANNs trained with a limited amount of project cost data can be used for cost estimations of new projects made in the early phases of the construction process, the cost estimations of construction projects can be made more accurate."

1.2.2 Problem definitions

Based on the problem analysis and the objectives, two problems are defined. Problem definition 1 is related to the theoretical objective of this study. Problem definition 2 is defined to realise the practical target of the study.

Problem definition 1:
"Can cost estimations be made more accurate by using ANNs trained with limited project cost data?"

Problem definition 2:
"How can ANNs trained with limited project cost data be used for cost estimations during the early phases of the construction process, and how accurate are these ANN-based cost estimations?"

1.2.3 Research questions

Based on the problem definition and the research objectives, the following main research question (MRQ) is drafted:

"How can cost estimations with ANNs trained with limited project cost data create more accurate cost estimations for construction projects during the early phases of the construction process?"

The following sub research questions (SRQ) will contribute in answering the MRQ of this research project. SRQs 1 and 2 belong to problem definition 1, while SRQs 3 and 4 belong to problem definition 2. To get a good overview of which questions need to be answered in order to answer the MRQ in full detail, the SRQs are broken down into separate more detailed questions.

SRQ 1: How can the cost estimation process for construction projects be characterised?

SRQ 2: How can ANNs be utilised for estimation tasks in the construction sector?
- *SRQ 2.1: What are ANNs?*
- *SRQ 2.2: For what estimation tasks can ANNs be applied in the construction sector?*

SRQ 3: How can (conceptual) cost estimations be performed with ANNs?
- *SRQ 3.1: Which data is required for estimations with ANNs?*
- *SRQ 3.2: How can cost estimations be performed with ANNs?*

SRQ 4: How can ANN-based cost estimation be used for construction cost estimating if limited data is available?
- *SRQ 4.1: Can ANNs developed with a limited dataset be utilised for cost estimation?*
- *SRQ 4.2: What is the accuracy of ANNs if they are developed with a limited dataset?*

1.3 Research design

In order to be able to answer the MRQ, this study is divided into two parts: a theoretical and a practical part (Figure 3). The theoretical part addresses problem definition 1 and SRQ 1 and 2, as they are associated with this problem definition. The theoretical part consists only of a literature review. During this literature review, in-depth research is conducted related to the characterisation of the cost estimation process in the construction sector, and how NNs could be utilised for (cost) estimation tasks in the construction sector. The practical part, addresses problem definition 2 and its corresponding SRQs 3 and 4. The practical part of the study can be divided into four stages namely: methodology, development, training and testing, and validation. In the methodology stage, research will be conducted in order to find out how cost estimations can be performed with NNs. Also during the methodology stage, input data derived from project cost calculation data for the NN will be selected, extracted, and prepared. In the development stage, an NN is constructed that can be used for cost estimation. Once the estimation model is constructed, the NN will undergo a training and testing stage. During the validation stage, it will be studied whether the developed NN model is useful enough for conceptual cost estimating. Once the theoretical and practical part are completed, a critical discussion will be provided and subsequently, a conclusion will be drawn in order to answer the MRQ.

.

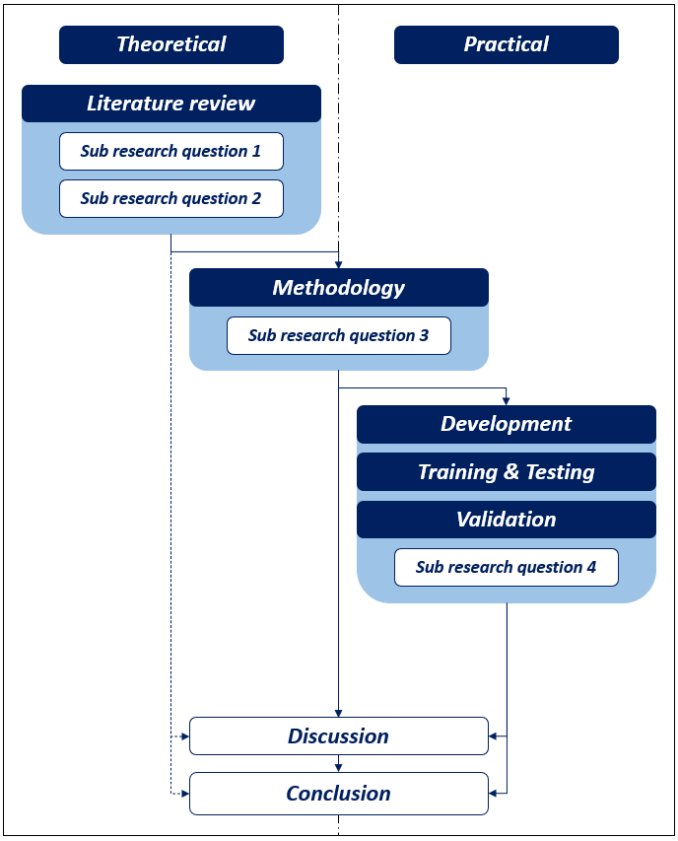


Figure 3: classification of research design

1.4 Research relevance

In this paragraph, the relevance of the presented study will be motivated by first discussing the scientific relevance and secondly the practical relevance.

1.4.1 Scientific relevance

Construction cost estimating is a challenging task in the construction process, especially when they are made during the early phases of the process. This is when limited information is available and many still unknown factors affect the project costs. Since only the most basic and functional decisions about the project have been made and the information for estimating the construction costs is ambiguous and highly subject to change, applying traditional cost estimation methods becomes inaccurate or impossible to implement for accurate cost estimates in these early project phases (Pal et al., 2018). In construction management and engineering (CME), NNs have gained considerable attention in solving complex nonlinear problems such as conceptual cost estimation. According to Waziri et al. (2017), the employment of the NN in construction for cost prediction, schedule estimating, productivity forecasting, predicting dispute occurrences, resolution outcomes, and contract performances, demonstrates its potentials and robustness in addressing problems that have proven difficult for traditional mathematical and statistical approaches to solve.

The application of NNs for (conceptual) cost estimations has been subject to many studies over the recent years. The studies of Jain & Pathak (2014), Kulkarni et al. (2017), and Waziri et al. (2017) all reviewed the application of NNs in CME. These studies all provide comprehensive results for reliable predictions and improved accuracy of NNs in different areas within CME. However, all of these studies lack theoretical explanation and justification about the applied methodology and therefore do probably not result in practical implementation. This study is aimed at closing the gap between scientifically proving the NN concept and practically utilising NNs for conceptual cost estimating. This research is therefore focussed on studying and establishing a clear methodology for utilising NNs. This is done especially for relatively small datasets, since construction firms often do not have large amounts of data stored inhouse. Therefore, the scientific contribution of this study is the eventual knowledge concerning whether NNs can be utilised for conceptual cost estimating when limited data is available, and how they can be used by describing a methodology for NN cost estimation tasks.

1.4.2 Practical relevance

Currently, more and more data is created and stored in a structured way in the construction industry. Having a large amount of data is useful in data analytics. Currently, a lot of construction firms are interested in how to efficiently utilise the data they possess to optimise or support their business processes. By the end of this thesis, the Dutch construction firm that initiated this study will gain more knowledge about applying ML (i.e. NNs) for estimation tasks and whether they can use NNs for their cost estimation objective.

In chapter *6.2 Discussion practical implementation*, the implementation of NNs will be discussed, and in chapter *7.2 Practical recommendations*, recommendations are provided to the Dutch construction firm. Note that the results of this study are generalisable to more firms in the Dutch construction industry. However, one should keep in mind that for this study the data of only one Dutch construction firm has been used. Therefore, the results are not directly translatable to all Dutch construction firms in the industry, as these firms may use different data and techniques.

1.5 Reading guide

The remainder of this thesis is structured as follows (Figure 4):

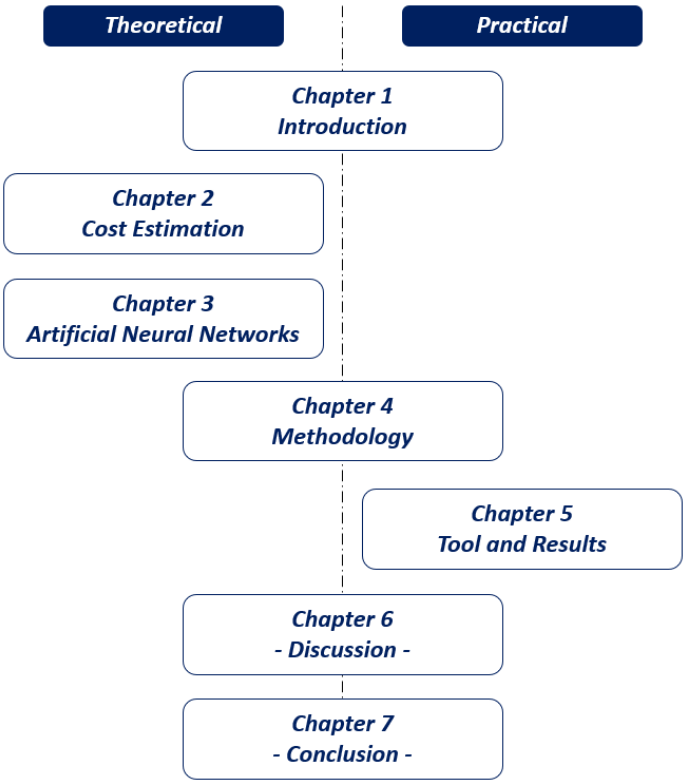


Figure 4: chapter classification

In chapter 2 and 3, the theoretical part of this study is addressed. These chapters provide the literature review for this study. In chapter 2. Cost estimation in the construction sector, the theory of construction cost estimation is discussed and in chapter 3. Artificial Neural Networks, the theory of Artificial Neural Networks is addressed. In chapter 4. Methodology, the methodology derived from literature and practice will be discussed. Chapter 5. Tool development and results, belongs to the practical part of the study and presents the development of the NN estimation tool and the results of training and testing the developed NN. In chapter 6. Discussion, a critical discussion on the results of the study will be presented. In chapter 7. Conclusion, the conclusions drawn after the study are provided and it also contains practical and research recommendations.

2. Cost estimation in the construction sector

This chapter is written to give an in-depth explanation of cost estimating in the construction sector and to answer the following question: *"how can the cost estimation process for construction projects be characterised?"*. This question will be answered in the conclusion part of this chapter by discussing topics related to cost estimating in the construction sector.

2.1 Construction process and cost estimation

In the AEC industry, the construction process can be divided into different stages. For the Dutch construction sector, "De Nieuwe Regeling – Standaardtaakbeschrijving (DNR-STB)" represents the nationally adopted process. According to the DNR-STB (2014), the construction process can be divided into ten phases (BNA & NLingenieurs, 2014). The process starts with the initiative / feasibility stage and ends with the use / exploitation phase. Table 1 shows the phasing of the construction process according to the DNR-STB.

Table 1: phasing of the Dutch construction process according to the DNR-STB 2014

Phase	Dutch	English translation
1	Initiatief / haalbaarheid	Initiative / feasibility
2	Projectdefinitie	Project definition
3	Structuurontwerp	Masterplan design
4	Voorontwerp	Preliminary design
5	Definitief Ontwerp	Definitive design
6	Technisch Ontwerp	Technical design
7	Prijs- en contractvorming	Price and contract formation
8	Uitvoering – Uitvoeringsgereed Ontwerp	Construction – construction design
9	Uitvoering – Directievoering	Construction – supervision
10	Gebruik / exploitatie	Use / exploitation

During the above-mentioned phases, various cost estimations are made. In the AEC industry, cost estimation is a key factor. Project success and quality depends on the accuracy of the estimation. An estimate is the best information source for deciding on a price for a project and it helps organising and planning the construction process (Gilson & Vanreyk, 2016). Cost estimates are used by project initiators to determine the project's scope, financial feasibility, and to allocate budgets, while contractors use estimates to decide whether to bid on a project (Ramos, 2017; Marker, 2017). Good cost estimates are needed because they seal the financial fate of a project. Overestimating the project costs may result in losing the project to a lower bidding competitor. A worse situation occurs when project costs are underestimated, as it can lead to severe financial company losses. However, sometimes it is done intentionally in order to beat the competition in the bidding process. This practice is called a buy-in: the contractor intentionally reduces an initially realistic estimate in order to win the project (Nicholas & Steyn, 2017).

2.2 Cost estimation in general

The U.S. Government Accountability Office (U.S. GAO) defines cost estimation as: "the summation of individual cost elements by using established methods and valid data to predict the future cost of a project based on what is known today" (U.S. Government Accountability Office, 2009). A more concise definition is given by Ramos (2017) who states that: "cost estimating in construction is the process of forecasting the cost of building a physical structure". According to the Association of Advancement of Cost Estimating (AACE): "an estimate is an evaluation of the elements of a project or effort as defined by an agreed-upon scope" (AACE, 2003). However, Dysert (2006) states that a definition of an estimate should more explicitly address the uncertainty involved and defines an estimate as: "a prediction of the probable cost of a project within a given and documented scope and to be completed at a defined

location and point of time in the future". While, Nicholas & Steyn (2017) define an estimate as: "an attempted realistic assessment based upon known facts about the work in terms of required resources, constraints, and the environment". The definitions point out that an estimate is a prediction (of the construction costs) conditioned on known information and involves uncertainty. These definitions show that cost estimations are being defined in different ways. However, it would be useful to have one concise summarising definition of cost estimation. Therefore, cost estimation will be defined in this study as: **the process of forecasting the cost of a project based on a documented scope and known facts by means of the summation of individual cost elements by using established methods and valid data**.

The cost estimation process is a derivative of the process of project cost management (Kharoubi, 2019) and project planning (Nicholas & Steyn, 2017). Project cost management includes four processes (Figure 5): [1] plan cost management, [2] estimate costs, [3] determine budget, and [4] control costs.

Figure 5: the four processes of project cost management

In the planning process, the starting point of the subsequent stages of the cost management process is defined. The approximate values of monetary resources required to complete the project are developed during the estimating process. In the budgeting process, the estimated costs of individual activities are summed to form a cost baseline. Finally, progress is monitored during the controlling process in order to update costs and manage changes to the cost baseline (Project Management Institute, 2017). Cost estimating processes interact and rely on processes from other disciplines and the development of the design. Therefore, cost estimating takes part during different phases (e.g. preparation, design, and construction) of the construction process with different aims (Lu, Lai, & Tse, 2019).

The focus of this research is on the estimating part of the project cost management process, and specifically on conceptual cost estimations made in the design phase of the construction process. Conceptual cost estimations are an essential part of project planning and strategically important. They are needed by the project initiator, contractor, designer, or even the lending organisation (i.e. financer) for purposes such as feasibility studies, financial evaluation of alternatives, or the formulation of an initial budget (Sonmez, 2004).

2.3 Classification of cost estimations

Cost estimates can most simply be classified into just three primary categories: design estimates, bid estimates, and control estimates (Ramos, 2017). *Design estimates* are made during the pre-design and design phase of a project. The most feasible construction method and types are determined with an order of magnitude or with a screening estimate. Subsequently, preliminary or conceptual estimates are made based on the schematic design. Finally, a detailed or definitive estimate is made based on the construction documents. In the tender phase, contractors prepare *bid estimates*, which are used to bid on the construction of a project. Once the contractor has signed an agreement to construct a project and before the actual construction starts, *control estimates* are made. Control estimates are used by contractors to plan ahead and to determine the project's cost to completion (Ramos, 2017). As indicated before, this work focusses on design estimates (i.e. specifically conceptual cost estimates).

The American Society of Professional Estimators (ASPE) classifies estimates according to a five-level system that becomes increasingly more detailed and reliable as the level increases. The classification systems used by ASPE are as follows (Ramos, 2017):

- Level 1: order of magnitude estimate
- Level 2: schematic design estimate
- Level 3: design development estimate
- Level 4: construction document estimate
- Level 5: bid estimate

In the Netherlands, a cost estimation classification system is used that is similar to the classes of the ASPE. The classification of cost estimates in the Netherlands is depicted in the NEN2699:2017 "Investerings- en exploitatiekosten". The NEN2699:2017 is applicable to all projects; it defines when and at what levels work is being done. The norm indicates at various levels which costs can be used in order to calculate and estimate the construction cost. Figure 6 shows that cost estimations evolve from level 1 to level 6, where level 1 is the most global level and level 6 the most detailed level of cost estimating (NEN, 2017).

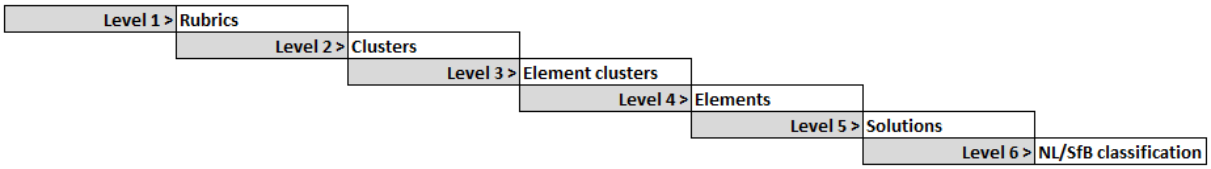


Figure 6: level classifications of cost estimations in the Dutch construction sector

2.4 Ways of cost estimating

Estimating can be done in two ways: top-down estimating and bottom-up estimating. Top-down estimates are made by looking at the project as a whole and are typically based on expert opinions or analogy to similar projects. Bottom-up estimating refers to estimating the costs of a project in detail. Costs of each element are estimated separately and then summed up to form the total project costs. Bottom-up estimates provide more accurate estimates than the top-down method. However, the bottom-up method is more time consuming and requires more data. The estimating approaches can be used in combination; parts of the project that are well defined can be estimated in detail with the bottom-up approach, while parts that are less defined can be estimated top-down (Nicholas & Steyn, 2017).

During the early phases of a project, the cost estimates are mostly made based on the top-down approach. Once the design process evolves over time and the project becomes more detailed, cost estimates will be made by applying the bottom-up approach. The bottom-up method provides the most accurate and reliable estimate as well as the cost figures necessary to establish the project budget and control accounts (Nicholas & Steyn, 2017).

2.5 Cost estimating techniques

The existing cost estimating techniques can be divided into the five categories (Matel, 2019) listed below:

1. Analogous estimating
2. Expert judgement
3. Detailed estimating
4. Parametric estimating
5. Probabilistic estimating

Appendix 1 provides a table that summarises a comparison of the mentioned cost estimating techniques based on their strengths, weaknesses and requirements. This table is based on the current research as well as on the tables provided in the studies of Kharoubi (2019) and Matel (2019).

2.5.1 Analogous estimating

Analogous estimating is known as a comparative estimating method. The cost estimate is developed by reviewing cost from previous similar projects and considering the differences with similar projects (Nicholas & Steyn, 2017; NASA Executive Cost Analysis Steering Group, 2015). Analogous estimates are generally used to study the feasibility of the projects as this method provides information about whether to proceed with the project or not (Burke, 2009). An analogous estimate is a relatively fast estimate (Nicholas & Steyn, 2017; Matel, 2019). If there are minor deviations in the cost data used from previous projects, the estimate can be quite accurate. However, it is very difficult to select the appropriate projects to compare the new project with (Matel, 2019), as this method requires extant information about prior projects. Therefore, companies gather cost data and store it into databases that classify costs according to type of project, work package, element and so on (Nicholas & Steyn, 2017).

2.5.2 Expert judgement

Expert judgement consists of cost estimates made by cost experts (Nicholas & Steyn, 2017; Marker, 2017). In the estimating process, the experts use their cost knowledge of prior similar projects to draw comparisons between past projects and the new project to make and adjust the new cost estimate (Marker, 2017). Usually, expert judgements are only made in the initiative phase and for projects that are poorly defined or if there are no prior similar projects to compare the new project with (Nicholas & Steyn, 2017).

2.5.3 Detailed estimating

Detailed estimating, also known as cost engineering, is estimating the costs of a project from a detailed determination of the costs of each cost category, work package and element of the project, including the duration of the work packages (Ramos, 2017; Nicholas & Steyn, 2017; NASA Executive Cost Analysis Steering Group, 2015). It is a bottom-up estimating method (Nicholas & Steyn, 2017; Matel, 2017) that provides the most accurate estimates of all estimating techniques (Nicholas & Steyn, 2017), but the method is very time consuming (Nicholas & Steyn, 2017; Ramos, 2017; Matel, 2019) and is associated with high costs (Matel, 2019). Detailed estimations are mostly made later on in the design phase because the estimating technique requires detailed project and design information which is often not available in the early phases of a project (Nicholas & Steyn, 2017). Estimation in detail has the advantage that there is the ability to determine exactly what is included in the estimate and check if nothing was overlooked (U.S. Government Accountability Office, 2009). Another advantage is that the detailed estimation method provides insight into the cost drivers of the project. The method has the disadvantage that for every new project a new detailed estimate must be made. This is time consuming and very costly. However, to speed up the estimating process, estimates of certain activities that reoccur in new projects can be taken from previous projects but must be integrated into the context of the new estimate (Matel, 2019).

2.5.4 Parametric estimating

Parametric estimating is a statistical estimating technique that uses multiple regression analysis to relate the construction costs to a model of prediction parameters (Chou et al., 2009; Ramos, 2017). A parametric estimate is derived from an empirical, mathematical and statistical relationship between historical project cost and cost parameters of the project (U.S. Government Accountability Office, 2009; Nicholas & Steyn, 2017). It is important that the cost parameters used in a parametric estimate are the cost drivers of the project (Matel, 2019). Mostly these cost parameters are physical features of a project such as area, volume, weight, or capacity (Nicholas & Steyn, 2017). To obtain a parametric

estimate, cost estimating relationships (CERs) need to be identified, subsequently an algorithm is applied to determine an approximation of the project costs (Kwak & Watson, 2005). To determine cost drivers and relevant CERs, a parametric estimate requires historical data (Ramos, 2017; Matel, 2019). Parametric estimates can be conducted quickly and they can be easily replicated (NASA Executive Cost Analysis Steering Group, 2015), as such they are useful early in the design phase where estimates are needed quickly (Nicholas & Steyn, 2017). Because parametric estimates use data of actual observations, the reliance upon opinion such as expert judgement is eliminated. In order to assure that parametric estimates stay in line with actual cost relationships between project attributes and costs, the CERS must be continually revised. This is a disadvantage of the parametric estimating technique (Matel, 2019).

2.5.5 Probabilistic estimating

In probabilistic estimating, probability distributions are used as input for the cost estimate of one or more cost parameters (Matel, 2019). This estimating technique focusses more on the risks and uncertainties involved in a project and tries to quantify the cost variability of the project. A probabilistic estimate gives insight into the change of exceeding the cost of a particular aspect in the range of possible costs, how much costs could overrun, and how uncertainties drive costs (Matel, 2019).

2.6 Cost estimating in the construction process

Cost estimating is an iterative process. Cost estimates are typically updated and revised as the project evolves over time and the project scope, information, and documents become more detailed (Marker, 2017). The estimating process starts based on a set plan. During the process, the costs and resources needed for project completion are forecasted based on the given level of detail and design development (Halpin, Lucko, & Senior, 2017). In the estimation process, the estimates are based on the information provided, the alternatives analysed, and the identified risks. Therefore, the cost estimation process relies on information from various sources (Project Management Institute, 2017).

Construction cost estimating is done in all phases of the construction process. In the initiative phase, the first estimate is made. Due to the low availability of detailed project information in this early phase, the estimate is the least reliable (Nicholas & Steyn, 2017). Hence, cost estimates in the early phases (i.e. initiative and design phase) are produced to check the affordability of the project and set a realistic cost limit for the client (Kharoubi, 2019). The preliminary estimates are compulsory for high-level decision making, since these estimates influence the feasibility and profitability of the project (Lu, Lai, & Tse, 2019). In the initiative / feasibility phase the estimate is mostly very rough, less reliable and based on cost index numbers per unit, functional unit, an element cluster, or an element. Once the construction process evolves, the estimations become more detailed; and in the construction phase, the estimations are the most detailed. Also, as the project design evolves and more detailed information becomes available, cost estimates become more reliable (Nicholas & Steyn, 2017).

In the Netherlands, the NEN2699:2017 provides an overview of the cost estimates made in the initiative / feasibility, design, and construction phase and their corresponding level of detail. Table 2, provides an overview of the different cost estimations in the construction process and their level of detail (NEN, 2017).

Table 2: construction cost estimations per construction phase

	Phase	Cost estimation	Technique	Level
	Initiative			
1	Initiative / feasibility	Initiatiefbegroting	Analogous/expert	Level 1-2
2	Project definition	Haalbaarheidsbegroting	Analogous/expert	Level 2-3
	Design			
3	Masterplan design	SO begroting	Analogous	Level 3-4
4	Preliminary design	VO begroting	Analogous	Level 3-4
5	Definitive design	DO begroting	Analogous	Level 4-5
6	Technical design	Directiebegroting	Detailed	Level 5-6
7	Price and contract formation	Inschrijfbegroting	Detailed	Level 5-6
	Construction			
8	Construction – construction design	Werkbegroting	Detailed	Level 5-6
9	Construction – supervision	Bewakingsbegroting	Detailed	Level 5-6
	Exploitation			
10	Use / exploitation	Exploitatiekosten	Detailed	Level 5-6

2.7 Accuracy of cost estimations

For (development) projects, accurate estimates are important because projects have budgets and timelines linked to paying back loans and generating revenues (Ramos, 2017). Accurately estimating the project costs is difficult, because the estimation process starts in the initiative phase where little is known about the project (Nicholas & Steyn, 2017). As the project moves on in the design phase and more information becomes known about the project, cost estimates become more certain (Nicholas & Steyn, 2017; Marker, 2017). Cost estimating is an ongoing process; it is normal that estimates are revised during the development of a project in order to ensure accuracy (Marker, 2017). The accuracy of cost estimates is influenced by: the quality of the project plan, the classification level of the estimate, the expertise of the cost expert, the accuracy of cost information, and the quality of the used estimation tools (Ramos, 2017).

The less defined the project, the greater the chance that the actual costs substantially differ from the final detailed estimate. Cost escalation occurs when actual costs exceed estimated costs. Some cost escalation is normal and acceptable, escalation up to 20% is common in the construction sector. In the early phases of a project, the accuracy of cost estimates made by experts varies from -15% to +25% (Hoogeveen, 2015; Rekveld, 2017). According to Phaobunjong (2002), the expected accuracy for a definitive estimate of a construction project is within a range of -5% to +15% and, the range of a conceptual stage estimate is within -30% to +50%. On the other hand, AbouRizk et al. (2002) state that a conceptual cost estimation for a construction project should be within a range of -15% to +25%. The accuracy of estimates can be improved by clearly defining project scope and objectives, and subdividing the project into small standardized work packages that can be estimated in more detail. Also, with a three-point estimate, the accuracy of an estimate can be improved. A three-point estimate combines: (a) optimistic, (b) pessimistic, and (c) most likely cost estimates to determine an expected cost estimate (Nicholas & Steyn, 2017).

Once the final detailed estimate is developed and approved, the cost estimate becomes the project budget and baseline against which project progress and cost performance can be measured (Nicholas & Steyn, 2017). Most often, definitive cost estimates do not remain static through project execution. Since cost estimates are based on numerous assumptions and are contingent upon risks, cost estimates are often updated as soon as these base assumptions change significantly or additional risks are identified (Marker, 2017). However, it is not recommended to revise estimates during the project

execution because this is in conflict with the purpose of establishing a cost baseline to measure progress and control costs (Nicholas & Steyn, 2017).

2.8 Cost estimating innovations in in the construction sector

Construction cost estimating changes and continues to evolve as new technologies, design, and building methods emerge (Ramos, 2017). The emergence and development of Building Information Modeling (BIM) has impacted cost estimating. According to Kharoubi (2019), BIM can support cost estimating in three ways: (1) IFC cost estimation, (2) BIM-based quantity take-off, and (3) integration with cost estimation software. The topic of performing cost estimation with BIM will not be discussed in further detail, since estimation with BIM falls outside the research scope of this study. However, the integration of NNs and BIM (tools) to gather input data for NNs will be discussed briefly in the discussion section of this thesis.

Developments in computer technology and mathematical programming techniques led to a recent development in cost estimating approaches to use complex estimating methods and larger volumes of (historical) cost data (Matel, 2019). Artificial Intelligence (AI) facilitated this change in cost estimating. AI methods allow investigating multi- and non-linear relationships between final costs and design variables (Günaydin & Dogan, 2004). AI methods currently used for cost estimating are mostly related to machine learning (ML), and knowledge-based systems (KBS) (Matel, 2019). An advantage of AI estimating methods is that when limited information is available, it is still possible to estimate costs quickly and fairly accurately (Günaydin & Dogan, 2004). Since cost estimating with AI (and ML) are the main topic of this study, it will be discussed into more detail in the upcoming chapter.

The ultimate goal in cost estimation is to accurately forecast the final cost of a project with no design details available (Roy, 2003). Both in the field of research and practice, diverse approaches and techniques are being refined and used to reach the ultimate goal (Zhou, 2018). However, it is important to keep in mind that estimating the costs of any project with absolute precision is impossible (Ramos, 2017).

2.9 Conclusion

This conclusion is written in order to answer the chapter-related research question *"how can the cost estimation process for construction projects be characterised*?"

Providing a general characterisation of the construction cost estimation process is hard, since the estimation process depends and relies upon a lot of factors (e.g. skills and education of the cost planner, cost estimation method, project information available etc.) and differs at each individual construction firm. Therefore, this conclusion will provide answers to the main characteristics of the cost estimation process required to perform the current study.

Based on the literature, cost estimation can be defined as: the process of forecasting the cost of a project based on a documented scope and known facts, by means of the summation of individual cost elements for using established methods and valid data. The process of cost estimation is a derivative of the processes of project cost management and project planning, because it takes place during different phases of the construction process with different aims. In addition, it interacts with and relies upon processes from other disciplines and the development of the design.

In general, the construction cost estimation process is characterised by making various cost estimations during construction. These become more detailed and accurate as the construction process evolves, the design becomes more detailed, and more is known about processes from other disciplines. According to the NEN2699:2017, ten different cost estimations (Table 2) can be distinguished in the construction process of a project. These different cost estimations can be classified

based on their level of detail and accuracy. The NEN2699:2017 is the Dutch classification system for cost estimations. In the NEN2699:2017, cost estimations are classified into six levels, where level 1 is the most global level and level 6 the most detailed level of cost estimating.

In the early phases of a project, the conceptual cost estimates are very rough and less reliable. They are used to check the affordability of the project and set a realistic cost limit for the client, because these estimates influence the feasibility and profitability of the project. Cost estimates made in the early phases are not the most accurate. As the project moves on during the design phase and more information becomes available about the project, cost estimates become more certain. A detailed estimate, often made just before actual construction on site, is considered the most accurate estimate and it mostly forms the definitive estimate of a construction project. The expected accuracy for a definitive estimate of a construction project is within a range of -5% to +15%, while the range of a conceptual stage estimate is within -15% to +25% of the final project cost.

This study focusses specifically on conceptual cost estimations (level 3-4 estimates), because they are an essential part of project planning and strategically important. They are needed by the project initiator, contractor, designer, or even lending organisation for purposes such as feasibility studies, financial evaluation of alternatives, or the formulation of an initial budget. These conceptual cost estimations are often based on the top-down approach and the analogous estimating technique.

As new technologies, designs, and building methods emerge, construction cost estimating evolves and improves. The rise of BIM in the construction industry has impacted cost estimating in different ways. Developments in computer technology and mathematical programming techniques led to a recent development in cost estimating approaches using complex estimating methods and larger volumes of (historical) cost data. Currently, AI and ML techniques are considered state-of-the-art estimating techniques that improve construction cost estimating in order the reach the ultimate goal: an accurate forecast of the final cost of a project without design details. In the upcoming chapter, a scientifically proven AI / ML technique (i.e. Artificial Neural Networks) will be discussed, including its usability for estimating the conceptual costs of new projects based on historical data.

3. Artificial Neural Networks

In this chapter, Artificial Neural Networks will be explained in more detail and the literature about the application of Neural Networks for (cost) estimation tasks will be reviewed. With this chapter, the following research question will be answered: *"how can ANNs be utilised for estimation tasks in the construction sector?".* The answer to the question above can be found in the conclusion of this chapter. This question will be answered by discussing different topics related to data mining, machine learning, and artificial neural networks.

Data mining is at the core of statistics, machine learning (ML) and artificial intelligence (AI) (Figure 7). Therefore, this chapter first briefly discusses data mining, and machine learning before starting to discuss NNs in detail in 3.3 Artificial Neural Networks.

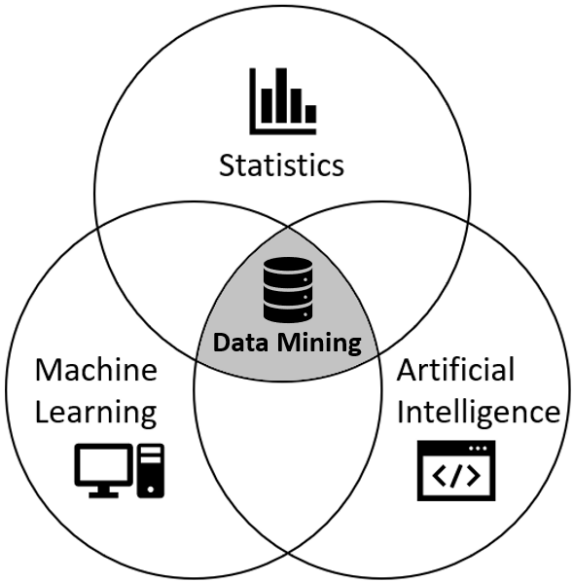


Figure 7: relationship between statistics, ML and AI

3.1 Data mining

Data mining can be seen as an extension of statistics with the advantage and addition of AI (Edelstein, 1999). Although statistics is at the core of data mining, data mining differs significantly from statistics (Friedman, 1997, Hoogeveen, 2015). Statistics are concerned with how to collect the data to answer a specific question, while data mining is concerned with how to discover "secrets" and hidden patterns in data i.e. knowledge discovery (Hand, 1999). It is the term ML that makes data mining different from statistics. ML is about studying computer algorithms to perform specific tasks (e.g. prediction). It is also about learning based on data to perform better in the future (Witten, Frank & Hall, 2011; Schapire, 2008). The goal of ML is to develop algorithms that learn automatically without human intervention, thus the computer improves its knowledge based on new data (Schapire, 2008). ML and data mining are at the core of AI (Schapire, 2008; Hoogeveen, 2015). Without ML it would require an unbelievable amount of complex code writing to achieve AI (Hilgers, 2018). It is important to realise is that data mining does not replace statistics; it must be seen as an extension of conventional statistical techniques (Nassar, 2007). In ML, statistics are still needed, since statistical tests are needed to validate and evaluate ML models and algorithms (Witten, Frank, & Hall, 2011).

There are two primary models in data mining: descriptive data mining models and predictive data mining models. Descriptive models summarise or describe general characteristics or behaviour of the data in the database and focus on finding patterns describing the data (Fayyad, Piatetsky-Shapiro, & Smith, 1996). Predictive models on the other hand, perform inferences on the data in order to make predictions on new data (Doreswamy & Hemanth, 2011; Provost & Fawcett, 2013).

Techniques in data mining can be divided into four categories: (1) association, (2) clustering, (3) classification, and (4) prediction (Nassar, 2007). Descriptive models contain association and clustering, while predictive models contain classification and prediction (Fayyad, Piatetsky-Shapiro, & Smith, 1996, Witten, Frank, & Hall, 2011). Although classification and prediction models both have a predictive nature, there is a difference. Classification techniques predict discrete attributes (i.e. categorical attributes), while prediction techniques try to predict a continuous attribute (i.e. numerical attributes) (Witten, Frank, & Hall, 2011; Provost & Fawcett, 2013).

Data mining has not been actively considered for knowledge discovery during this study, because NNs can be utilised for data mining. NNs can be used for knowledge discovery (i.e. data mining) because they are capable of finding relationships and (hidden) patterns in data (Amarendra, Lakshmi, & Ramani, 2002; Turban, Sharda, Delen, & King, 2010; Juszczyk, 2017).

3.2 Machine learning

In computer science, ML is the field where existing data is utilised to predict, or respond to, future data (Paluszek & Thomas, 2017). ML models are models that can ideally self-learn autonomously from the data that it deals with (i.e. unsupervised learning) (Elfaki et al., 2014). However, most ML models do this by learning from training data (i.e. supervised learning) (Matel, 2019). ML techniques have the advantage that they are able to deal with uncertainty, can work with incomplete data, and that they are able to judge new cases based on experiences acquired from similar cases in the past. Also, ML models can investigate the multi- and nonlinear relationships between parameters.

A drawback of ML is that it is considered to be a black box technique due to the lack of technical justification about the decision the ML model makes (Matel, 2019). In this study the focus will be on supervised learning, since the ML model (i.e. NN) will be provided with training data to learn from. An end user has the choice to either trust the ML model or not, without much of a rational basis. Another drawback of ML is that it requires a very large amount of data in order to be able to build a trustworthy model.

The ML process is illustrated below in Figure 8. The figure shows what happens in a typical ML process. First, a training dataset is exposed to the ML algorithm for training. Once the training is completed, an ML model is established. This ML model can be evaluated using a testing set. Finally, this ML model can be used for implementation in practice. The model can now be provided with data which is unknown to it, in order to give an output based on the patterns and relationships found in the training data (Matel, 2019). High-quality input data is therefore of crucial importance in the use of ML models.

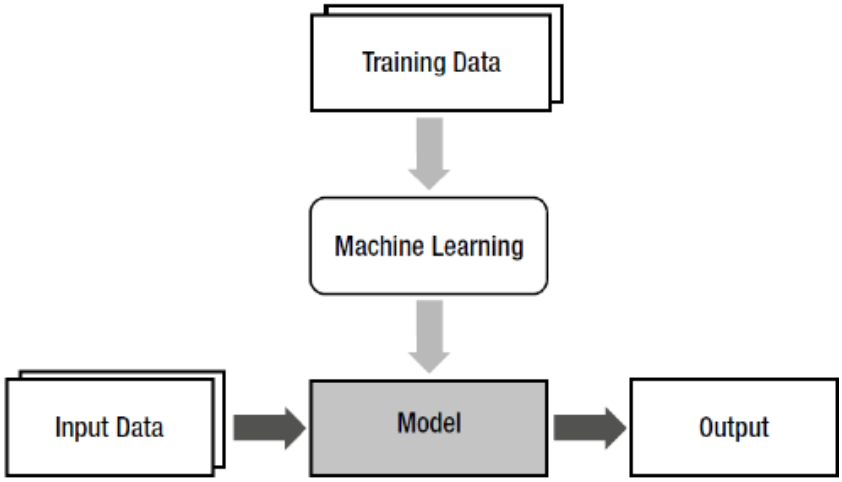


Figure 8: Illustration of machine learning process (Matel, 2019)

3.2.1 Kinds of machine learning

ML methods are data-driven and, just like humans, need to be trained to perform tasks, ML systems need to be trained as well. Based on their training method, ML techniques can be classified into two ML techniques: (1) supervised learning, and (2) unsupervised learning. Predictive data mining models (i.e. classification and prediction) use supervised learning as opposed to descriptive data mining models (i.e. association and clustering), which use unsupervised learning (Aggarwal & Yu, 1999).

Supervised learning

The learning process in supervised learning is based on data that provides input values as well as output values. The process is called supervised because the patterns in the data are identified using the correct corresponding output values of the input values (Matel, 2019). In other words, the data mining algorithm learns by being provided with the actual outcomes of the target of the training data (Provost & Fawcett, 2013). With supervised learning, current knowledge is applied to solve a problem and the answer is compared with the solution provided by the current knowledge. When the answer is wrong, the current knowledge will be modified to solve the next problem better. In supervised learning this is done by revising the ML model several times in order to reduce the error between predicted output and actual output (Matel, 2019).

Reinforcement learning is part of supervised learning. Reinforcement learning is generally used when optimal interaction between algorithm and end user is required (Kim, 2017). It uses sets of input, some output, and grades as training data. Reinforcement learning is mostly concerned with problem-solving in an active environment, in which the specific situation requires specific actions from users or devices. In reinforcement learning the focus is on performance, which involves finding a balance between exploration of the unknown and exploitation of current knowledge (Busoniu, Babuska, de Schutter, & Ernst, 2010; Matel, 2019).

Unsupervised learning

The learning process in unsupervised learning is only based on the input values. The unsupervised learning technique identifies patterns in the data and reacts based on the presence or absence of commonalities in the data. The big advantages of this are that unsupervised learning techniques can learn aspects from data that were not known in advance, and that these techniques can find hidden structures in data. The unsupervised learning technique is mostly used in situations where the “right” answer is not known beforehand (Matel, 2019).

3.2.2 Application of machine learning

Supervised learning concerns classification and regression problems. A situation typically requires regression when the output variable is a continuous value. However, it requires classification when the output variables are categories or discrete classes. In this specific study, the objective is concerned with predicting construction costs and thus a continuous value. Therefore, this research concerns a regression problem. However, this study would concern a classification problem if the objective of the study was classifying projects construction cost on labels such as high cost, medium cost, and low cost.

Regression predictive modelling is the task of approximating a mapping function (f) from input variables (x) to a continuous output variable (y) (Matel, 2019). The algorithm that is used for the prediction of construction costs in this study is an Artificial Neural Network (ANN). According to Matel (2019), this is one of the most commonly used algorithms in regression analysis for cost prediction of construction projects.

3.3 Artificial Neural Networks

Artificial Neural Networks (ANNs) are considered one of the main categories of ML systems (Matel, 2019). Originally, NNs are inspired by the study of processes in the human brain (Günaydin & Dogan, 2004). Just as the human brain stores knowledge it acquired through learning, an NN stores learned knowledge. It stores this knowledge in the inter-neuron connection strength known as synaptic weights (Petroutsatou et al., 2012). A typical NN consists of an interconnected group of neurons (nodes) which process information by mathematical relations between these neurons (Amarendra, Lakshmi, & Ramani, 2002; Petroutsatou et al., 2012).

The NN is a powerful technique for prediction due to the characteristics it possesses (Turban, Sharda, Delen, & King, 2010). Their ability to map input patterns to their associated output patterns, learn from examples (i.e. supervised learning), generalise knowledge, their predictive capabilities, and their robustness and fault tolerance (i.e. they recall full patterns from incomplete, partial or noisy patterns) makes the NN a strong estimation tool (Amarendra, Lakshmi, & Ramani, 2002; Turban, Sharda, Delen, & King, 2010; Juszczyk, 2017).

3.3.1 Neural Network aspects

In this paragraph, the NN architecture (i.e. the NN model), functions, and algorithms needed to utilise NNs will be discussed as they are considered to be the main aspects of NNs (Hoogeveen, 2015).

ANN model architectures

NNs consist of "neurons" i.e. simple linear or nonlinear computing elements, interconnected in complex ways and organised in layers (Sarle, 1994; Zhang et al., 1998). A typical NN is built on three layers: the input layer, the hidden layer, and the output layer (Figure 9).

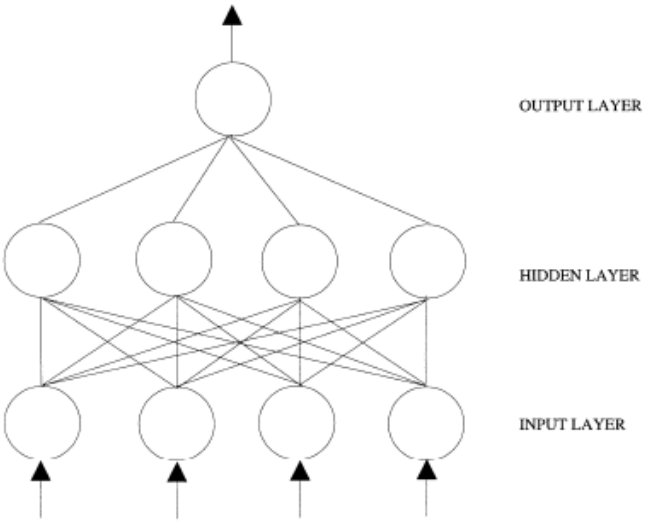


Figure 9: typical NN architecture (Zhang et al., 1998)

NNs can be categorised into two main types: feedforward networks and recurrent networks, also known as feedback networks (Singh & Chauhan, 2009). Within the two main categories, several NN architectures can be distinguished. Single-layer perceptron, multilayer perceptron, and Radial Basis Functions are examples of feedforward networks. Competitive networks, Self-Organising Map (SOM), Adaptive Resonance Theory (ART) and Hopfield Neural Networks are examples of recurrent networks (Jain, Mao, & Mohiuddin, 1996; Amarendra, Lakshmi, & Ramani, 2002; Ni, 2008). The difference between feedforward and recurrent networks lies in the data flows of the models. In a feedforward NN, the data flows only in one direction: forward, from input nodes, through hidden layers, to output nodes. There are no feedback loops present in feedforward NNs since they use the backpropagation

algorithm to learn (Turban, Sharda, Delen, & King, 2010). Recurrent NNs have an two-sided data flow (i.e. forward and backward), thus feedback loops are present in these models (Singh & Chauhan, 2009).

Constructing an NN is a non-trivial task. Decisions need to be made about the number of layers. Subsequently, the number of nodes in each layer needs to be determined and finally, the number of arcs that interconnect the nodes needs to be determined. In this study, a multilayer perceptron feedforward NN will be used as the NNs architecture. NNs with (multiple) hidden layers are called multilayer or deep NNs, which are referred to as multilayer perceptrons. These types of NNs are most commonly used in practical applications (Hoogeveen, 2015; Kim, 2017). A simple example of the multilayer perceptron (MLP) model is given in Figure 9. In an MLP, every node has a connection with all the nodes of the next layer; this is also referred to as a fully connected neural network i.e. each node in one layer is connected to each node in the next layer exactly one time.

Activation functions and loss functions

Neural Networks can compute and learn almost any function. Almost any process can be represented as a functional computation in NNs. Therefore, NNs are considered universal function approximators (Walia, 2017). To learn and make sense of complicated nonlinear complex functional mappings, NNs need activation functions (Walia, 2017; Sharma, 2017; Hoogeveen, 2015; Matel, 2019). An activation function, also known as a transfer function, is a function that is used to convert an input signal of a node in the NN into an output signal. Subsequently, the output signal is used as input in the next layer of the NN (Walia, 2017; Sharma, 2017). Without activation functions, NNs will not be able to learn and model complicated kinds of data and would simply act as a linear regression model (Walia, 2017). Hence, activation functions must be applied to add the ability to the NN to learn something complex and complicated and to make the NN a powerful estimator (Walia, 2017).

Activation functions can be divided into two categories (Sharma, 2017):

1. **Linear activation functions**
   The output with this activation function will not be limited between any range.
2. **Nonlinear activation functions**
   These activation functions are the most used. A nonlinear activation function makes it easy for an NN model to generalise or adapt with variety of data.

Nonlinear activation functions are mainly divided based on their range or curves (Sharma, 2017). The four main nonlinear activation functions are: (1) Sigmoid, (2) tanh, (3) ReLU, and (4) leaky ReLU. Below in Figure 10, the graphs and formulas of these non-linear activation functions are given.

1. **Sigmoid**
   The curve of the Sigmoid or logistic activation function is S-shaped. A reason to use Sigmoid is because the function exists between 0 and 1. Therefore, it is used especially for NN models that predict probability as an output.
2. **Tanh**
   The tanh, or hyperbolic tangent activation function, is similar to Sigmoid. However, the range of the tanh function is from (-1 to 1). An advantage of the tanh function is that negative input values will be mapped strongly negative and zero input values will be mapped near zero. The range of the tanh function makes it very suitable to use for classifications between two classes.
3. **ReLU**
   Nowadays, the Rectified Linear Unit (ReLU) is the most used activation function (Sharma, 2017; Walia, 2017). ReLU is used in almost all the convolutional NNs or deep learning. The range of ReLU is between 0 and infinity. The function has the disadvantage that all the negative values become zero, thereby immediately decreasing the ability of the NN model to fit or train the data properly. Therefore, the ReLU activation function is most often used in cases where the data only concerns positive values.

4. **Leaky ReLU**
   This function is similar to the ReLU function. However, it is modified in order to solve the disadvantage concerning negative values in the ReLU function. Leaky ReLU is most often used in cases where the range of ReLU needs to be increased because the used data contains small negative values.

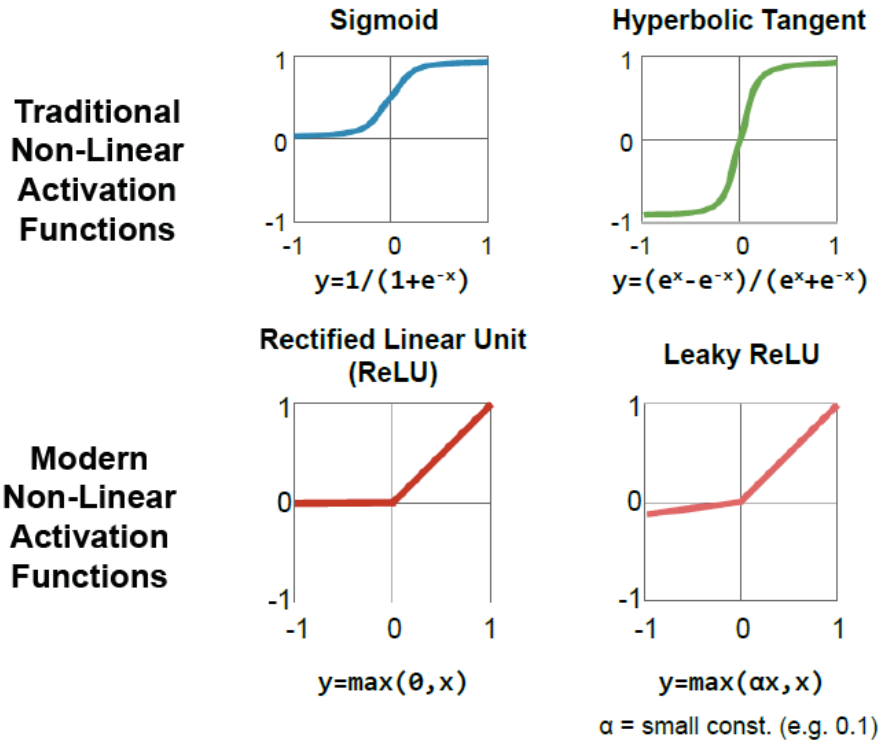


Figure 10: graphs and formulas of nonlinear activation functions

An important feature of activation functions is that they are differentiable in order to perform backpropagation. Backpropagation is an optimisation strategy and is done by propagating backwards in the network to compute gradients of loss (i.e. error) with respect to the weights and subsequently optimising the weights using gradient descent (Walia, 2017). To calculate the loss or error of the NN model, loss functions need to be used. The loss function in NNs can be considered a learning algorithm, since a loss function is used to estimate the loss of the model and accordingly update the weights to reduce the loss on the next evaluation (Brownlee, 2020).

The choice of the loss function must match the specific predictive modelling problem the model is concerned with, such as classification or regression. Also the output layer of the NN model must be configured appropriately to be compatible with the loss function (Brownlee, 2020). According to Brownlee (2020) there are three types of loss functions:

1. Regression loss functions
   a. Mean squared error loss (MSE)
   b. Mean squared logarithmic error loss (MSLE)
   c. Mean absolute error loss (MAE)
2. Binary classification loss functions
   a. Binary cross-entropy
   b. Hinge loss
   c. Squared hinge loss
3. Multi-class classification loss functions
   a. Multi-class cross-entropy loss
   b. Sparse multiclass cross-entropy loss
   c. Kullback Leibler divergence loss

As this study is concerned with a regression problem only the regression loss functions (i.e. MSE, MSLE, and MAE) will be discussed into more detail. Regression predictive modelling problems involve predicting a real-valued quantity. The MSE, MSLE, and MAE loss functions are appropriate for regression predictive modeling problems (Brownlee, 2020).

- **Mean squared error loss (MSE):** is the default loss useful for regression problems. The MSE loss is the preferred function when the target variable is normally distributed. The MSE is calculated by taking the average of the squared differences between predicted and actual values. A perfect MSE value is 0.0. The squaring means that larger mistakes result in more error than smaller mistakes (Brownlee, 2020).
- **Mean squared logarithmic error loss (MSLE):** is an appropriate loss measure when a NN model is predicting unscaled quantities directly. To calculate the MSLE, first the natural logarithm of each of the predicted values must be calculated, and subsequently the MSE. The MSLE has the advantage that large differences in predicted values are punished less hard as with the MSE (Brownlee, 2020).
- **Mean absolute error loss (MAE):** is an appropriate loss function when the distribution of the target variable is mostly normally distributed, but may have outliers. The MAE is calculated as the average of the absolute difference between actual and predicted values (Brownlee, 2020).

3.3.2 Training and learning of Neural Networks

The most commonly used NN architecture is a multilayer perceptron feedforward NN. These NN types learn by applying the backpropagation algorithm (Turban, Sharda, Delen, & King, 2010; Hoogeveen, 2015; Janssen, 2018; Matel, 2019). To predict output values as accurately as possible based on input variables, an NN needs to be configured and trained (i.e. learning by backpropagation). As mentioned earlier in *3.2.1 Kinds of machine learning*, there are two types of learning processes for NNs (supervised, and unsupervised). For this study, the supervised learning process will be used, since the objective is to predict an output value (i.e. construction costs) based on input variables (i.e. cost parameters). A supervised learning process consists of the following steps (Loy, 2018; Matel, 2019):

1. Feedforward propagation (calculating the predicted output)
2. Backpropagation (updating weights and biases)
3. Repeating steps for all training data

The process of feedforward propagation and backpropagation is schematically visualised in Figure 11.

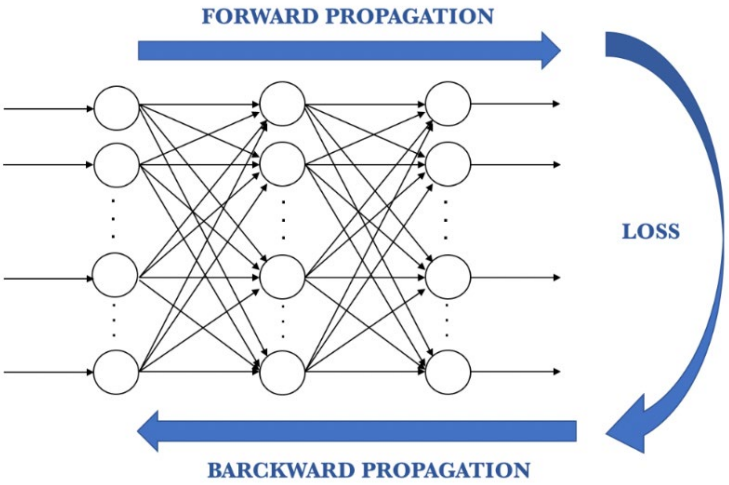


Figure 11: schematic representation of feedforward- and backpropagation

Feedforward propagation

Training an NN starts with feedforward propagation (i.e. calculating the predicted output). In order to do this, the input layers receive the inputs from the data and direct them to the nodes in the hidden layers without calculations. The nodes in the hidden layers and output layer perform the computations of the network and add and adjust weights. Every hidden node takes the weighted input of the previous node and outputs a single value based on a predefined transfer function (Bosscha, 2016). Eventually, the neural network provides an output based on the input and configuration of the weights (Matel, 2019).

Once the NN is established after feedforward propagation, the random configuration of the weights leads to an error between the NN output and the actual value, thus making an NN useless at this point. To configure a useful NN, the configuration of the weights must be optimised to reduce the error between NN output and the actual value. This can be done by applying backpropagation and lots of repetition.

Backpropagation

In order to train an NN model, backpropagation is applied in feedforward NNs. Backpropagation is a type of supervised learning, the backpropagation algorithm calculates the difference between actual and expected output values (i.e. error) (Singh & Chauhan, 2009). According to Poole & Mackworth (2010), backpropagation learning is a gradient descent search through the parameter space to minimise the sum-of-squares error. The backpropagation procedure can be described in four steps (Figure 12); first, a set of training data is provided to the NN. Subsequently, the NN compares the predicted output to the desired output from the training data. The errors in each output neuron are then calculated. The backpropagation algorithm calculates what the output should have been and assigns a different weight to the neuron to match the desired output. The weights need to be adjusted to lower the local error. Backpropagation takes the partial derivative of the squared error of the NN with respect to each weight (Witten, Frank, & Hall, 2011). By doing so, the rate change of the error can be observed, corresponding to the value of the weight increases. If the derivative is positive, the error is increasing when the weight is increasing. The initial weight and biases are chosen randomly. The procedure has to be repeated until the error is minimised to an acceptable value (Abraham, 2005).

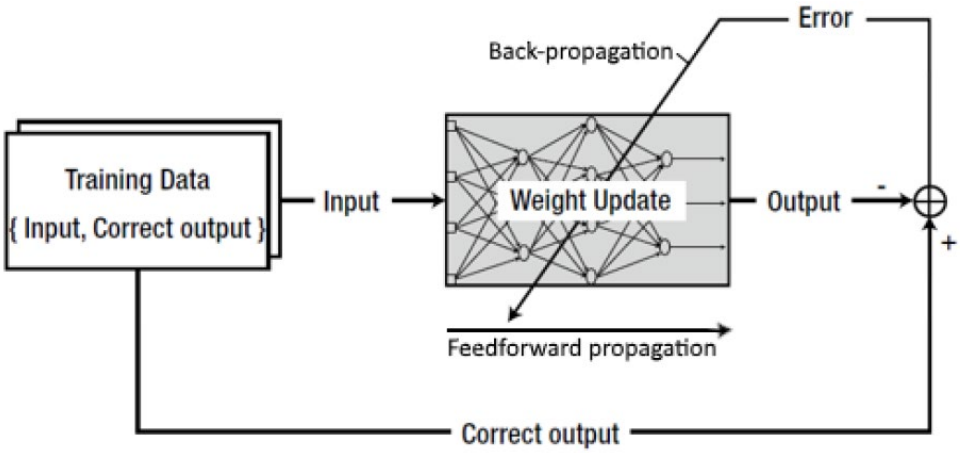


Figure 12: typical backpropagation procedure (Matel, 2019)

Learning rate and momentum are two important parameters for backpropagation. The modification of the weights can be controlled with the learning rate (Witten, Frank, & Hall, 2011). A small learning rate causes small weight changes; on the other hand, a large learning rate causes large weight changes. (Attoh-Okine, 1999). The momentum term determines the effect of past weight changes on the current direction in the weight space (Abraham, 2005). It can be seen as a factor used to speed NN training (Hoogeveen, 2015). The determination of the learning rate and momentum is crucial, but yet a difficult

task. According to a study performed by Attoh-Okine (1999), a learning rate of 0.2 to 0.5 and a momentum of 0.4 to 0.5 provide the best performance.

When to stop training

In the backpropagation algorithm, one way to update the weight is to compute the weight updates for each input sample. The values are stored during one pass through the training set. This is called an epoch (Abraham, 2005). The backpropagation algorithm stops training once the validation error is increasing. This is because the error of the training set can decrease, while the performance of the network decreases (i.e. increasing of the error on unseen data). This is known as early stopping (Prechelt, 2012). Early stopping avoids the overfitting problem in the network's training, which degrades the generalisation ability of the network and therefore affects the network's practicality. The training hence stops at the lowest validation error (Wu & Liu, 2009).

It is important to note that the number of nodes in an NN can influence its generalisation ability. Generalisation indicates how well an NN is able to learn from the data it is trained with and apply what is learned to other incoming data. If it is trained well, it will perform as well in new situations with new data as it will on data it was trained with. If an NN is too complex for the amount of data it is trained with, it will most likely overfit and generalise poorly. Also, if an NN is too simple, it will most likely underfit and generalise poorly. Figure 13 shows the concepts of underfitting and overfitting as well the optimal fit (Matel, 2019).

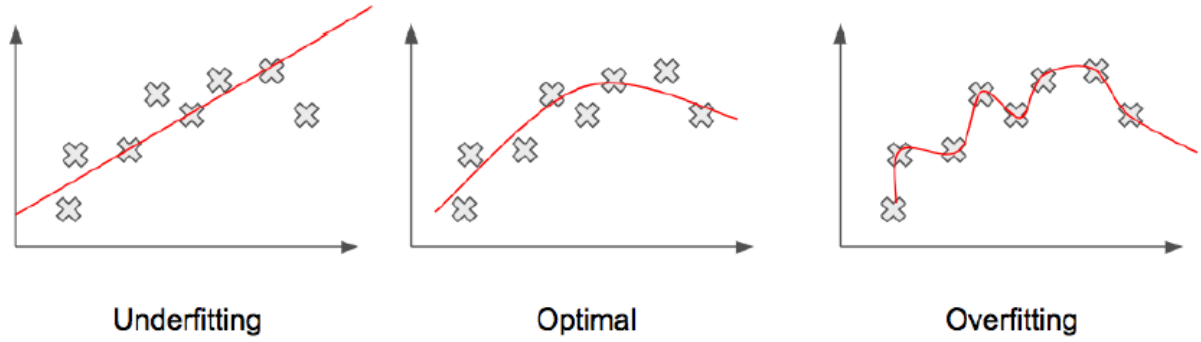


Figure 13: concepts of underfitting and overfitting (Matel, 2019)

Overfitting can be prevented by including features such as early stopping and a dropout layer into the NN code. Early stopping is a regularisation criterium to prevent overfitting. This criterium ensures that NN training is stopped when the loss starts to increase again during a given number of epochs. By applying the early stopping criterium, generalisation error in the NN model can be reduced. Another way to prevent overfitting is by using a dropout layer, as suggested in the study of Hinton et al. (2012). Dropout prevents that the NN model makes complex co-adoptions with the training data. The dropout criterium randomly zeroes some of the elements of the inputs to a node according to a given probability (default value = 0.5). This technique has proven to be effective for regularisation and preventing co-adoption of neurons as described in the paper of Hinton et al. (2012).

3.3.3 Interpretation of Neural Networks

Despite the superior predictive power of the NN over traditional prediction models (e.g. linear regression) and its advantages (i.e. adaptive learning, generalising knowledge, self-organisation, robustness, and fault tolerance), the NN is considered to be a "black box technique" because it barely provides an explanation about its predictions (Intrator & Intrator, 2001; Olden & Jackson, 2002; Kim et al., 2004; Waziri et al., 2017). The enormous amount of weights makes the NN complex in providing explanatory knowledge about the relative influence of variables in the prediction. Therefore, gaining knowledge of the causal relationships in NNs is difficult (Intrator & Intrator, 2001; Olden & Jackson,

2002). The lack of transparency is a major drawback for AI methods. Therefore, the development of methods for visualising, explaining, and interpreting deep learning models has recently attracted increased attention. Explainability in AI systems is important for the verification of AI systems, to improve and learn from AI systems, but also to see if AI systems are in compliance with legislations (Samek et al., 2017). Explainable artificial intelligence (XAI) falls outside the scope of the present study.

Besides XAI, there are other ways to gain knowledge from AI systems such as NNs. A Neural Interpretation Diagram (NID) is the primary method. An NID is basically a visualisation of the trained NN. The colours and line thicknesses make an NID different from traditional NNs, therefore NIDs are also better interpretable (Özesmi & Özesmi, 1999). The line thickness in an NID presents the relative importance of each connection. Thicker lines present greater weights, and therefore a relatively higher contribution of the independent variable. The colours represent the direction of the connection weight in an NID. Negative connection weights are coloured red and positive connection weights are coloured blue (Hoogeveen, 2015). A second method to learn from NNs is by applying the Garson algorithm. This algorithm quantitatively determines the relative importance of each input variable in the NN. But, different from an NID, the algorithm does not provide the direction of the relationship between input and output variables (Garson, 1991; Hoogeveen, 2015).

3.4 ANNs in the construction sector

In this section of the chapter, the literature about the application of NNs for (cost) estimation in the construction sector is reviewed. First, the use of NNs for estimating in the construction sector will be discussed, and subsequently the use of NNs specifically for cost estimations of projects in the construction sector.

3.4.1 Estimating with ANNs

Several studies have reviewed the application of NNs in construction activities and found that NNs have been successfully applied in fields such as: prediction, productivity, risk analysis, safety, duration, decision making, resource optimisation, classification, selection, optimisation and scheduling, and claims and dispute resolution outcomes (Jain & Pathak, 2014; Kulkarni et al., 2017; Waziri et al., 2017). NNs were originally applied in the construction industry of the late 1980's. Adeli & Yeh (1989) were the first who applied NNs for construction-related activities. Since the first application of NNs in construction, NNs have been successfully applied to solve numerous construction industry related problems (Kulkarni et al., 2017; Waziri et al., 2017) and a large number of journal articles about the application of NNs in construction management and engineering (CME) has been published (Jain & Pathak, 2014).

In the field of productivity, labour, and/or equipment, NNs have been utilised to estimate the daily productivity of dozers, the productivity within a developing market for formwork assembly, the duration of steel fixing and concrete pouring activities, the productivity of ceramic wall construction, the labour productivity of marble finishing works, the labour productivity for concreting, and bricklayer (builder) productivity (Kulkarni et al., 2017). NNs have also been applied in the field of risk and safety analysis. In this area, NN-based procedures have been used to predict the likelihood of contractor default, estimate the risk index for an expressway construction stage using the principles of system theory, operability, independence and comparability, and to estimate percentage variation between the forecasted and actual costs of floats at 30, 50, 70, and 100% completion stages (Kulkarni et al., 2017). The most common application of NNs in the field of construction is prediction. NNs have been applied in this field to predict: tender bids, construction budget performance, project cash flow, construction demand, labour productivity, organisational effectiveness, and construction costs (Jain & Pathak, 2014). Predicting construction costs with an NN as a tool is popular, as it has been used to predict the costs of school buildings, residential projects, apartment projects, cost of overall building

projects, cost for highway, tunnels, general overheads, and cost of deviation in reconstruction projects (Kulkarni et al., 2017).

According to Vahdani et al. (2012), NNs have been applied extensively in the last decade to predict the conceptual costs of construction projects. This is not surprising, since Arafa & Alqedra (2011) state that NNs are potentially the most efficient technique for early stage cost estimation (i.e. conceptual cost estimations). According to Arafa & Alqedra (2011) this is because NNs, in comparison with linear regression, are able to model interdependencies between input data, which will inevitably occur when one considers the significant variables on the construction cost. Also, NNs would handle incomplete datasets more effectively. This is a very important advantage of NNs, as complete datasets are not (always) available at the early stage of any project.

3.4.2 Cost estimation with ANNs

In this section, a more in-depth study is made of the research projects that rely on NNs to do their cost estimation. With this review, an early assessment is provided of the extent to which NNs can really be used for this purpose, and what the characteristics of similar studies are. The section ends with an overview table.

In the study performed by Elhag & Boussabaine (1998), two NNs using data of 30 historical projects were developed to predict the tender price of primary and secondary school buildings. The data of the 30 projects was extracted from the Building Cost Information Service (BCIS) and comprised 19 primary school projects and 11 secondary school projects. Thirteen cost factors (type of project, type of contract, market conditions, number of tenderers, site slope, start conditions, ground conditions, excavation conditions, site access, work space in site, number of stories, gross floor area, and duration) were gathered to serve as input variables for the NN models. Both NN models were developed by using the backpropagation algorithm. For the first model, 4 input variables were used and for the second model, 13 input variables were used. The first model had an architecture of 4-3-1 while the architecture of the second model was 13-13-1, where 13, 13, and 1 are the input neurons, hidden neurons, and output neurons respectively. The results of the study revealed that the NN models effectively learned during training and had prediction accuracies of 91.1% and 99.9%. However, in the validation stage, the average accuracy percentages of the models were 79.3% and 82.2% respectively. During training, a root mean squared error (RMSE) of 0.0398 and 0.0001 and mean absolute percentage errors(MAPE) of 8.87% and 0.01% were observed. However, during model validation, the RMSE and MAPE of both models proved to be 0.0669 and 0.0626 in terms of RMSE, and 20.74% and 17.77% in terms of MAPE.

The study of Günaydin & Doğan (2004) investigated the utility of the NN methodology in order to overcome cost estimation problems in the early phases of the building process. During their study, Günaydin & Doğan developed an NN for estimating the square meter cost of a reinforced concrete structural system of 4-8 storey residential buildings. To model the NN, a feedforward NN architecture and the backpropagation learning technique were used. The NN was modelled with NN software (NeuroSolutions) based on a modification of the NN spreadsheet from an earlier study performed in 1994 by Hegazy & Ayed. The developed NN model relied on eight input variables available at the early design stage (total area of the building, ratio of the typical floor area to the total area of the building, number of floors, console direction of the building, foundation system of the building, floor type of the building, and location). It used cost and design data from 30 projects for training and testing. For each training cycle, 20% of the data samples were selected at random for the validation set. This provided a training dataset of 24 projects and a testing dataset of 6 projects. The performance of the proposed NN is measured with cost percentage error (CPE) and mean squared error (MSE). The developed NN model showed an average cost estimation result of 93% and an MSE of 0.038 during model validation.

Best results were obtained by an NN model with a 8-4-1 system architecture. Furthermore, Günaydin & Doğan concluded that the model is useful to make appropriate decisions at early project stages.

Kim et al. (2004) studied the performance of three cost estimation models (Multiple Regression, NNs, and Case-Based Reasoning). The developed models are based on input variables such as year, gross floor area, storeys, total unit, duration, roof types, FND types, usage of basement, and finishing grades, while for output variable the actual cost was used. NeuroShell2 was used for the NN model. In the study, 75 NN models were proposed with varying parameters as the number of neurons in the hidden layer, learning rate and momentum. Based on the cost data of 530 historical projects, the models were examined and the results of the study showed that the NN model was the most accurate. Model performance was measured by the mean absolute error rate (MAER). The system architecture of the best performing NN model was as follows: 12-9(or 25)-1 (0.6-0.6), where 12, 25, 1, 0.6 and 0.6 are the input neurons, hidden neurons, output neurons, learning rate and momentum respectively. The NN model with an architecture of 12-25-1 provided the lowest MAER of 2.97 while the NN model with an 12-9-1 architecture provided an MAER 5.61. On average, the 75 NN models gave an MAER of 5.65, with 8% of the estimates within 2.5% of the actual error rate and 90% within 10%.

Sonmez (2004) compared regression analysis and NNs of conceptual cost estimation for continuing care retirement community projects. For the study, data was gathered from 30 continuing care retirement community (CCRC) projects in the United States. Construction year, location, total building area, proportion car parking area, and area for commons were used as input variables for the NN. Two NN models and one regression model were developed during the study. The results of these models were compared for closeness of fit and prediction performance. Both NN models had a different number of hidden neurons and were trained using the backpropagation algorithm. The architecture of the first NN was 5-6-1, and the architecture of the second NN was 5-3-1. Two error measures were used to asses model performance, namely MSE and MAPE. A prediction performance of $3.6x10^{12}$ and $3.8x10^{12}$ was observed in terms of MSE, and in terms of MAPE, a performance of 12.3% and 11.7% was observed. Results showed that one NN model achieved an accuracy level of 12%, which was considered satisfactory for conceptual cost estimating. Furthermore, Sonmez indicated that, while regression analysis requires a decision about the class of relations to be used in modelling, NNs were able to identify the relationships between variables and project cost. This statement of Sonmez is similar to the findings of Kim et al. (2004), who stated in their study that NNs eliminate the need to find a cost estimating relationship.

Kim et al. (2005) applied hybrid models of NNs and Genetic Algorithms (GA) during a study to estimate the preliminary cost of residential buildings. Data for training and performance evaluation was gathered from 498 residential building projects constructed from 1997 to 2000 in Korea. NeuroSolutions for Excel was used to develop three NN models that had 8 input variables and a varying number of neurons in the hidden layer. After training the NN models, the model performance was assessed by using estimated error rate (ERR) and weighted estimated error rate (WEER) as performance indicators to compare the average weighted error. For model 1 the average WEER was 5.21% at its lowest point when the NN has 16 neurons in the hidden layer, for model 2 the average WEER was found to be the lowest at 4.63%, and for model 3 an average WEER of 5.93% was found to be the lowest when the NN model has 12 neurons in the hidden layer. Furthermore, the results of the study revealed that optimising parameters of backpropagation NNs using GA is most effective in estimating the preliminary cost of residential buildings. According to Kim et al. (2005), GA could help estimators overcome the problem of the lack of adequate rules for determining the parameters of NNs.

Luu & Kim (2009) developed an NN to estimate the total construction cost (TCC) of apartment projects in Vietnam. The seven input variables (rank of project, gross floor area, number of floors, year of construction, petrol price, and steel price) were collected from 91 questionnaires and for training and

testing of the NN 14 datasets of historical apartment projects were used. MATLAB was used to develop and train the model NN toolbox and to assess the model performance, the accuracy measures mean percentage error (MPE) and MAPE were used. The results of the study showed that the NN model is a reasonable predictor for the TCC of apartment projects and that it has the potential to improve the cost estimation model for apartment projects. During validation, an average MPE of -1.5% and average MAPE of 8.5% were observed. As the MAPE value of the NN falls within the acceptable limit of 10%, Luu & Kim conclude that the capability of their NN in predicting TCC of apartment projects has been confirmed during their study.

Cheng et al. (2010) propose an AI approach, the evolutionary fuzzy hybrid neural network (EFHNN), to improve conceptual cost estimate precision. The AI approach proposed by Cheng et al. integrates NN and high order NNs (HONN) into a hybrid NN (HNN). For their study, the data of 28 construction projects was used spanning the years 1997 to 2001. Of the 28 projects, 23 cases were used for training purposes while the remaining 5 cases were used for testing. The estimators developed in the study achieved high levels of precision for construction cost estimation during an early stage of a project. The developed EFHNN model achieved an average overall estimate error of 10.356%, which is considered to be highly accurate, as overall estimation accuracy of within a range of 25% is acceptable and considered accurate by the authors.

The study of ElSawy et al. (2011) presented an NN to predict site overhead cost in Egypt. In order to create the NN, testing data of 52 projects between 2002 and 2009 were used and data from five new projects were used for model validation. Throughout 58 experiments, an adequate NN model was developed with an architecture of 10-13-1. The RMSE technique was used to assess the performance of the NN model. An RMSE of 0.276479 was observed when testing the NN model on five new projects. Results of the study indicate an accuracy level of 80%. During the study, it was observed that the NN model wrongly predicted the site overhead costs for only one project in the validation data sample.

Arafa & Alqedra (2011) aimed at developing an efficient model based on NNs to estimate the costs of building projects at early stages. In their study, they used data of 71 historical building projects from the construction industry of the Gaza Strip and significant parameters obtainable at the pre-design state (e.g. ground floor area, number of storeys etc.) for model development. The NN toolbox from MATLAB was used to develop the NN model. Seven parameters (ground floor area, typical floor area, number of storeys, number of columns, type of footing, number of elevators, and number of rooms) comprised the input layer of the NN. A dataset of 35 data entries was used for the training of the NN model. The remaining data entries (36) were equally divided between validation and testing datasets. An NN of (7-7-7) was found to be the optimal NN architecture. The trained NN reasonably succeeded in predicting the cost estimation of buildings at early stages by just using basic and fundamental information. A coefficient determination of 90% was found between the actual and predicted cost values. Results of the study also revealed that ground floor area, number of storeys, and number of elevators in the building are the most effective variables influencing early stage cost estimates.

Wang et al. (2012) used NNs and support vector machines (SVM) to predict cost and schedule success at the conceptual project stage based on 92 building projects. Of these 92 projects, 67 projects were used for training and the remaining 25 for testing the NN model. The NN models were built with the NeuroSolutions software. First, a single NN was developed as a base model, after which two NN ensemble concepts (bootstrap aggregating NN and adaptive boosting NN) were developed. The developed NNs had overall prediction accuracies of 76% (single NN), 76% (bootstrap aggregating NNs), and 80% (adaptive boosting NNs) for cost success. For schedule success, the overall prediction observed accuracies were 68% (single NN), 72% (bootstrap aggregating NNs), and 80% (adaptive boosting NNs).

Lyne & Maximinio (2014) developed an NN model to predict the total structural costs of building projects in the Philippines. They used six input variables (number of basements, floor area, number of storeys, volume of concrete, area of formwork, and weight of reinforcement steel) and historical data of 30 projects. The data was randomly divided into training data (60%), validation data (20%), and network generalisation data (20%). The NN toolbox in MATLAB was used to construct the NN, and the MSE accuracy measure was used to asses model performance. Favourable training and testing phase outcomes were observed during model development. The study resulted in an NN model that reasonably predicted the total structural costs of building projects. A after several trials, an NN with an architecture of 6-7-1 was found to be the best predicting NN model with an MSE equal to $2.98x10^{15}$.

Magdum & Adamuthe (2017) developed several NNs and MLPs with varying architectures for construction cost prediction to identify the best NN and MLP model and their belonging number of neurons, activation functions, and hidden layers. Feedforward NNs and the backpropagation algorithm were used for the NNs. To model the NN, six variables (cement, sand, steel, aggregate, mason, and skilled labour) were used as input for the NN. Four NN models and twelve MLP models were compared by evaluating the RMSE of the NN models. The best scoring NN was found to have an architecture of 6-8-1 with an RMSE equal to 41.69. Overall results showed that MLP and NN models provided better results than regression methods, however, the performance of NNs on training and testing data varies significantly.

Arabzadeh et al. (2018) developed NNs, regression, and hybrid models to estimate the costs of spherical storage tanks. The number of input neurons for the NN is equal to the number of variables as thickness, tank diameter, and length of the weld. Two NN models with two different learning algorithms, namely the Levenberg–Marquardt and the Bayesian regularized learning algorithm, were developed in MATLAB. To assess model performance, the MSE performance indicator was used. During training, MSE values of $2.53e^{-4}$ for the LMNN and $5,07e^{-4}$ for the BRNN were observed, while during testing MSE values of 0,291 for LMNN and 0,333 for the BRNN were observed. Furthermore, the results indicated that NNs are more accurate than regression, however, results also showed that hybrid NNs outperform single NNs.

In Table 3 below, an overview of the studies is provided related to the data availability, NN models, best performing NN architecture, and results obtained during the studies. The review of the studies and Table 3 reveal that the data availability of most studies was limited. The data availability of the reviewed studies ranges from 14 to 530 projects. Even though most of the studies used small data samples during the study for NN training, testing, and validations, the results of these studies show good results regarding to model performance. These findings support the objective of this research to perform cost estimates with NNs and limited data availability.

However, it may be that in the studies where limited data has been used, the data was comparable and strongly related, and therefore the NNs of the studies yield good results. All reviewed studies used a feedforward backpropagation NN architecture with only one hidden layer. All NN architectures in the reviewed studies differ, because the creation of an NN is task and problem specific. For the development of the NNs, most studies used NN software packages (e.g. Matlab or NeuroSolutions) and project variables that are easily obtainable at early project stages. Results of the studies reviewed indicate that NN performance after training is higher in most cases than NN performance after testing or validation. This is probably due to feeding the NN unknown data that has not been used to train the NN.

Table 3: overview of ANN cost estimating studies

Study	Data availability	NN models	NN architecture	Results
Elhag & Boussabaine (1998)	30 projects	2	4-3-1 13-13-1	RMSE 0,0398 ; 0,001 {TR} 0,0669 ; 0,0626 {VAL} MSE 8,87% ; 0,01% {TR} 20,74% ; 17,77% {VAL} Prediction accuracy 91,13% ; 99,99% {TR} 79,3% ; 82,2% {VAL}
Günaydin & Doğan (2004)	30 projects	1	8-4-1	MSE 0,038 {VAL} Prediction accuracy 93% {VAL}
Kim et al. (2004)	530 projects	75	12-9-1 12-25-1	MAER 2,97; 5,61 Average MAER 5,65 Prediction accuracy 8% of the estimates within 2.5% of the actual error rate, and 90% within 10%
Sonmez (2004)	30 projects	2	5-6-1 5-3-1	MSE $3,6x10^{12}$ and $3,8x10^{12}$ MAPE 12,3 and 11,7
Kim et al. (2005)	498 projects	3	8-16-1 8-?-1 8-12-1	WEER 5,21% ; 4,63%, and 5,93%
Luu & Kim (2009)	14 projects	1	n.d.	MPE -1,5% MAPE 8,5%
Cheng et al. (2010)	28 projects	1	HNN*	Average overall estimate error 10,356%
ElSawy et al. (2011)	52 projects	58	10-13-1	RMS 0.276479 Prediction accuracy 80%
Arafa & Alqedra (2011)	71 projects	1	7-7-7	Prediction accuracy 90%
Wang et al. (2012)	92 projects	3	n.d.	Prediction accuracy 76% cost success 68% schedule success
Lyne & Maximinio (2014)	30 projects	1	6-7-1	MSE $2,98x10^{15}$
Magdum & Adamuthe (2017)	n.d.	4	6-8-1	RMSE 41,69

Arabzadeh et al. (2018)	n.d.	2	3-11-1 (LMNN) 3-10-1 (BRNN)	MSE $2.53e^{-4}$ LMNN {TR} $5,07e^{-4}$ BRNN {TR} 0,291 LMNN {VAL} 0,333 BRNN {VAL} Prediction accuracy 90%

n.d. = not defined in study
HNN* = Study concerns application HNNs which architecture is different from basic NNs

3.5 Conclusion

This conclusion answers the chapter-related research question: *"how can ANNs be utilised for estimation tasks in the construction industry?"*.

To utilise NNs as a statistical AI method, a prediction data mining technique and supervised machine learning approach need to be used, since data mining and machine learning are at the core of AI. NNs are considered to be one of the main ML categories that can be utilised for cost estimations in combination with a supervised learning technique for regression problems. Due to the supervised learning, an NN is able to generalise knowledge and learn from examples, which is useful for cost estimations as they are mostly based on previous cases.

Before an NN can be utilised for estimations, the NN needs to be constructed. However, this is a non-trivial task as decisions need to be made about the NN architecture, activation functions, and loss functions. In most estimation tasks performed with NNs, a single-layer feedforward backpropagation NN was used. The architecture of these NNs consist of an input layer, a hidden layer, and an output layer.

The nodes in all layers are typically fully connected (i.e. each node in a layer is connected to a node in the next layer exactly one time). NNs with this architecture are called feedforward backpropagation NNs, since information flows only one way (i.e. forward) and NN weights and biases are updated through a backpropagation algorithm. Learning rate and momentum are two important parameters for backpropagation: they control the weight modification as well as the effect of past weight changes. When the NN is constructed, it can be trained and utilised for estimations after training. The performance of estimations done with NNs needs to be assessed in order to measure whether the NN estimations are accurate. Different performance indicators exist to assess the NN estimation performance, however, most used is the MSE technique.

The literature showed that NNs can be applied for a wide variety of estimation tasks in the construction sector, ranging from productivity estimations to the prediction of construction costs. The literature concerning construction cost estimation with NNs indicates that, even though NNs use small data samples, they still are superior for construction cost estimation tasks. These findings support the objective of this study to estimate construction costs with NNs, even when data availability is small. They also indicate that NNs are most suitable for conceptual cost estimations. Therefore, during the remainder of this study, an NN will be constructed in order to perform conceptual cost estimations based on historical of past construction projects.

4. Methodology

In the previous chapters, the theory of (conceptual) cost estimation and Artificial Neural Networks were discussed in order to form a theoretical framework. In this chapter, the theory of the previous chapters will be translated into a methodology for the system development of an NN which can be applied to a practical task (i.e. conceptual cost estimation of construction projects). First, the theory and methodologies used for the system development is discussed and why they are relevant for the specific research. At the end of the chapter, the discussion of the methodology of NN development and the specific NN development process will lead to the following research question to be answered: "how can (conceptual) cost estimations be performed with NNs?".

4.1 Introduction

The objective of this study is to perform cost estimations with NNs. The literature indicated that NNs are an appropriate technique for early stage cost estimations (i.e. conceptual cost estimations). Conceptual cost estimations are level 3-4 estimations often made in early design stages (i.e. the stage of masterplan design and preliminary design) for feasibility studies, and to set a realistic cost limit for the client. These conceptual cost estimates are mostly performed according to the top-down approach and the analogous estimating technique. The latter technique matches perfectly with the supervised learning reasoning of NNs because the analogous estimating technique is known as a comparative estimating method. With the analogous estimating technique, the cost estimate is developed by reviewing costs from similar projects in the past and by considering the differences with similar projects. Before conceptual cost estimations can performed with an NN however, it needs to be constructed first. The remainder of this chapter is devoted to describing the methodology used to construct an NN that can be utilised to perform conceptual cost estimates.

For the development of the NN model, the programming language Python will be used. Python is open source, friendly, and easy to learn as programming language. It is easily interpretable , interactive, and object-oriented. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes (Python, n.d.). For the code writing of the NN model, the open source machine learning library of TensorFlow (TF) as well a library of PyTorch will be used.

TF is an end-to-end open source platform for ML. It has a comprehensive, flexible ecosystem of tools, libraries and community resources, this enables researchers to push the state-of-the-art in ML and developers easily build and deploy ML-powered applications (TensorFlow, n.d.). In TF, tf.keras can be used for easy model building and the training of deep learning models (i.e. NNs). The high-level application programming interface (API) tf.keras is often used for fast prototyping, state-of-the-art research, and productions. It has the following advantages: user-friendliness, modular and composable, easily extendable (TensorFlow, n.d.). Besides TF, the open source ML framework PyTorch will be used as well. This framework provides a rich ecosystem of tools and libraries that can be used to develop ML-powered applications. In PyTorch, torch.nn is a module which can be used for the creation and training of NNs (PyTorch, n.d.).

Both ML libraries will be used to develop an NN in Python suitable to perform conceptual cost estimations. In chapter 5. Tool development and results, the coding of the NN with the ML libraries of TensorFlow and PyTorch will be discussed in more detail.

Artificial Neural Network development

In the remainder of the chapter, the development process of the Neural Network, which will be used during the study to predict conceptual construction costs of construction projects, will be discussed. Generally, NNs are developed through three phases and six basic steps (Hegazy et al., 1994; Günaydin & Doğan, 2004; Luu & Kim, 2009; ElSawy, 2011;):

Phase I: Modeling phase

1. Problem definition
2. Data gathering and representing
3. Defining the network

Phase II: Implementation phase

4. Structuring the network
5. Training and testing the network

Phase III: Application phase

6. Applying the network for estimations

The model development phases and their required steps to develop an NN suitable for the objective of this study will be discussed in the upcoming paragraphs.

4.2 Modeling phase

The modeling phase involves the steps of problem definition, data gathering and representing, and defining the neural network. In this paragraph these steps will be discussed in detail.

4.2.1 Problem definition

In the problem definition step, the objective is to decide on, the problem that will be addressed with the NN, which information needs to be used, and what the NN will do. This is already clearly expressed in previous chapters, but for clarity and simplicity it will be provided below again.

The objective of this study is to perform (conceptual) construction cost estimations with an NN based on actual project cost data of past projects. By the end of this study the developed NN must be able to predict the (conceptual) construction costs within a limit that is seen as accurate for conceptual construction cost estimates. From the conclusion of chapter 3 *(3.5 Conclusion)*, we can conclude that we want to reach an accuracy of about 80% in order to be within range of state-of-the-art solutions and proposals.

4.2.2 Data gathering and representing

Once the problem and objective of the NN is established, the data gathering and representation can start. The focus in this step is on gathering as much data as possible within the practical limits of the available resources (e.g. time, money etc.) (Hegazy et al., 1994). As NNs are data dependent, gathering enough data is important to create a reliable and effective dataset (Hegazy et al., 1994; Waziri et al., 2017; Arabzadeh et al., 2018). A comprehensive dataset is thus needed, as unavailability of certain data can be a limitation for the developed NN model in providing meaningful output (Kulkarni et al., 2017). However, it is not the data quantity that is most import, the quality and representativeness of the data is more important. A good measure to validate data representativeness is the range covered by the data (Hegazy et al., 1994).

Kim et al. (2012) state in their study that estimation models often require large amounts of fully documented data without anything missing in the data to achieve high accuracy. Also, Günaydin & Doğan (2004) mention in their study that the performance of NNs strongly depends on the quantity and quality of data because NNs learn from examples. In the construction industry, organisations often do not possess such large amounts of data. However, the data can be incomplete, because NNs are capable of dealing with incomplete and noisy data (Kim et al., 2012; Arafa & Alqedra, 2011). An important aspect to realise before using it is that NNs can only deal with data on a numerical scale. Qualitative data can still be used, but it needs to be converted into a numeric form before it can serve as input data (Günaydin & Doğan, 2004; Kulkarni et al., 2017).

For this study, actual project cost data is gathered from a construction firm in the Netherlands. In total, project cost data of 15 historical projects executed between 2017 and present (2020) are gathered. Ten of these 15 projects comprise multi-storey housing projects (i.e. apartments), while the remaining 5 comprise residential housing projects. The construction costs of these 15 projects range from €991.331,- to €37.710.000,-. Table 4 provides a first representation of the gathered data samples in terms of project type, number of units, square meter gross floor area, and construction costs. Later on, in chapter 5 (5.1 Data analysis) 5. Tool development and results, a more detailed data analysis of the gathered data samples will be provided.

Table 4: representation of gathered data samples

Project	Price index	Type	# of units	M2 GFA	Construction cost
1	July 2017	Houses	62	12.566	€ 8.985.000
2	July 2017	Houses	33	4.169	€ 4.077.000
3	October 2017	Apartments	24	2.877	€ 2.929.000
4	February 2018	Apartments	199	41.186,39	€ 23.432.580
5	February 2018	Apartments	105	13.201,04	€ 17.956.183
6	March 2018	Apartments	19	4.438,50	€ 4.676.895
7	June 2018	Apartments	125	14.155,30	€ 20.781.766
8	June 2018	Apartments	26	1.983,75	€ 2.517.868
9	December 2018	Houses	31	5.903.32	€ 5.746.624
10	March 2019	Houses	31	3.878,03	€ 4.381.000
11	May 2019	Apartments	233	31.717,50	€ 37.710.000
12	June 2019	Apartments	185	15.897	€ 23.670.773
13	July 2019	Houses	124	18.077,20	€ 15.928.802
14	October 2019	Apartments	158	20.702,05	€ 31.389.253
15	November 2019	Apartments	9	657,90	€ 991.331

Previous research already proved that, even with limited data availability, results of NNs for cost estimations can still be encouraging. Therefore, despite the data availability for this study being limited, it was decided to still use the limited available data to develop an NN suitable to perform cost estimations. The boxplot in Figure 14 shows the distribution of the available data based on their project type (i.e. 1 = houses and 2 = apartments) and related to their associated construction costs. Although the distribution of the data on the ranges is not uniform, the data can still be used, since NNs possess the capability of dealing with incomplete and noisy data (Arafa & Alqedra, 2011).

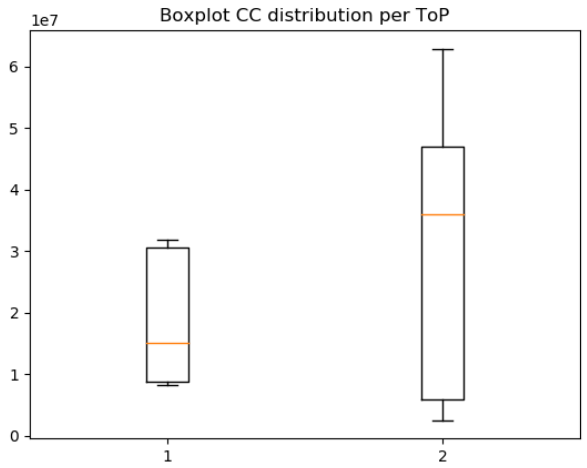


Figure 14: boxplot of CC distribution of original gathered data

The data availability for this study is limited, as only 15 datasets were gathered from a Dutch construction firm. For NNs this does not have to be a problem, because previous research already proved that, even with limited data availability, results of NNs for cost estimations can still be encouraging. However, it is desirable to use more data for NNs as they normally rely upon thousands of cases. Therefore, it was decided to simulate data based on the 15 gathered data samples. This is done to ensure that the NN will be trained with 'enough' data.

In Python, a code for data simulation was developed (Python code is provided in Appendix 3). By using the probabilistic distribution of housing and apartment projects of the 15 gathered data samples and then using these probabilities, decisions were generated by making use of np.random_choice which has the ability to randomise based on given probabilities. By using the probabilities of the original data samples and by using randomly chosen multiplication factors new data was simulated by the data simulation code. For this, the data simulation code uses the principle of linear interpolation, as new data points are derived within the range of a collection of known discrete data points assuming a certain relationship between these points. Also, the code assumes a continuous uniform distribution. This is a distribution on an interval with a constant probability density, meaning that there is no preference for any value from that interval. To ensure that the simulated data is in line with the original data and falls within a certain range, simple checks were added to the code to make sure that unrealistic data samples will be excluded. Checks that were added to the code are for example limits on the number of storeys, number of units, and building height. When the simulated data exceeded the set limit in the code, the simulated data sample was declared invalid and was excluded from the simulated data. By using this data simulation code, a dataset of 35 new data samples was simulated. Therefore, a total of 50 data samples can now be used for this study for NN training and testing. The complete dataset for this dataset is provided in Appendix 2.

The boxplot in Figure 15 shows the distribution of the simulated data based on their project type (i.e. 1 = houses and 2 = apartments) related to their associated construction costs. The distribution of the simulated data on the ranges is also not uniform, however, as indicated earlier this does not have to be a big concern.

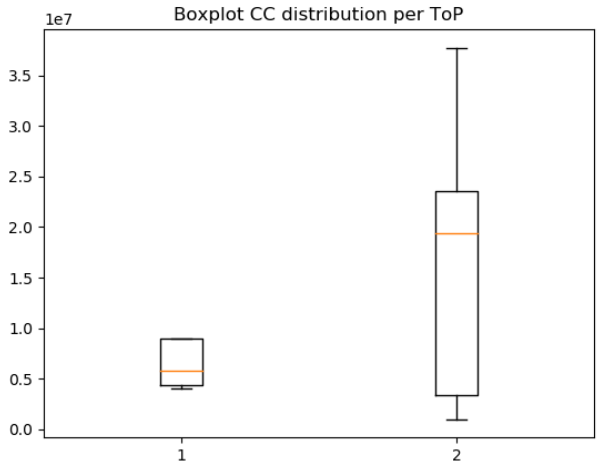


Figure 15: boxplot of CC distribution of simulated data

The objective of the study is to perform conceptual cost estimations with NNs. As indicated earlier, conceptual cost estimates are made during the early stages of the construction process when little is known about the project. Therefore, the estimation attributes that will be used as input variables for the development of the NN for this study are attributes that can easily be identified in early design stages. These attributes are similar to attributes used by cost planners who calculate the conceptual

construction cost according to the analogous cost estimating method. Besides the similarity to variables used for conceptual cost estimates by cost planners, the attributes are also similar to input variables used in similar studies which used NNs for cost estimation.

The nine attributes that will be used as input variables are as follows: type of project, gross floor area, usable floor area, gross building volume, number of storeys, roof area, façade area, number of units, and building height. As the objective of the study is to estimate the (conceptual) construction cost, this attribute must be considered the output variable. Table 5 provides an overview of the cost estimation attributes considered in this research in terms of unit, measurement scale, and range. That the simulated data covers a greater range than the original data is not a concern, as this is done intentionally. This ensures that the range of the testing data (i.e. original data) is completely covered through the training data (i.e. simulated data). Therefore the NN model will probably make more accurate estimations.

Table 5: overview of cost estimation attributes considered in this study

ID	Attribute	Unit	Measurement scale	Range
1	Type of project	-	Nominal	1 = houses {19} 2 = apartments {31}
2	Gross floor area	M^2	Numeric	{658 ; 41404}
3	Usable floor area	M^2	Numeric	{526 ; 30156}
4	Gross building volume	M^2	Numeric	{1657 ; 112815}
5	Number of storeys	floors	Numeric	{3 ; 48}
6	Roof area	M^2	Numeric	{244 ; 19887}
7	Façade area	M^2	Numeric	{714 ; 24904}
8	Number of units	pcs	Numeric	{9 ; 398}
9	Building height	M^1	Numeric	{10 ; 154}
10	Construction costs	€	Numeric	{991331 ; 62778506}

In Figure 16, the correlation coefficients of the 9 estimation variables that will be used as input variables for the NN are presented for the original data as well the simulated data. The graph of the correlation coefficients reveals that the correlation pattern of the simulated data is almost similar to that of the original data. This supports the usefulness and reliability of the simulated data for this study.

In *5.1 Data analysis* the correlation coefficients of the estimations variables of the original data as well the simulated data will be discussed into more detail.

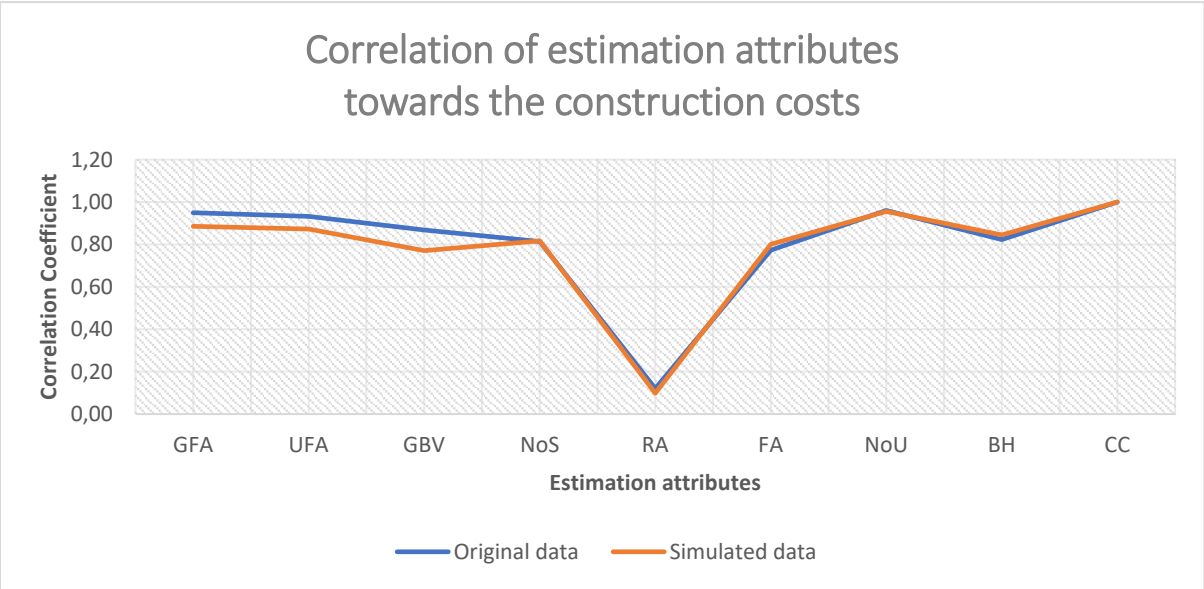


Figure 16: correlation of estimation attributes towards the construction costs

Note that in this study data was simulated based upon the 15 original cases gathered from a Dutch construction firm. Due to the limited availability, this data simulation was initialised in order to ensure that more than 15 data samples can be used for NN training and development. Due to the fact that simulated data is used in this study, results of the study should be carefully interpreted as the simulated data probably does not represent the reality. Still, the simulated data can be used for this study to prove the concept of NNs and to validate the method followed. Therefore, practical implementations and recommendations based on this study should not be directly implemented into the daily practice of construction cost estimators.

Defining the network

One of the most complex tasks during the modeling phase includes defining the architecture of the NN, because designing an NN is a complex and dynamic process that requires the determination of the NN structure and internal rules (i.e. activation functions and learning algorithms) (Günaydin & Doğan, 2004; Elbeltagi et al., 2014). A good design of the architecture for an NN is important, since the performance of an NN depends on the architecture of the model (Hegazy et al., 1994; Magdum & Adamuthe, 2017). In determining the architecture of an NN, the designer has many degrees of freedom because decisions need to be made about a lot of design parameters (i.e. number of input nodes, number of hidden layers, number of nodes in the hidden layer, activation functions, learning algorithms etc.). This flexibility represents an important problem: no defined methodology for defining the architecture of an NN (Hegazy et al., 1994). According to Günaydin & Doğan (2004), an NN is empirically rather than theoretically derived. This is in line with the statement of Hegazy et al. (1994) who state that designing an NN is much more an art than it is science. Also, different NNs can produce similar results with the same set of data (Elbeltagi et al., 2014). Therefore, different NN architectures, using different configurations for the design parameters, need to be considered to find the optimal NN architecture (Hegazy et al., 1994).

The majority of studies that applied NNs for (cost) estimation tasks developed their NNs with 3 layers (i.e. input layer, hidden layer, and output layer) in accordance with the feedforward backpropagation (FFBP) NN architecture (Elbeltagi et al., 2014; Kulkarni et al., 2017). According to Hegazy et al. (1994), one hidden layer is sufficient enough to generate an arbitrary mapping between input and output. Also, one hidden layer is suggested to avoid overtraining (Wang et al., 2012). In this study, the NN will also be designed according to the FFBP architecture and will use one single hidden layer as suggested by Hegazy et al. (1994).

For the design of the NN, the number of input nodes, hidden layers and nodes, and output nodes needs to be determined first. The selection of these parameters is problem dependent and does not follow a specific method. Often the selection of input parameters relies on prior knowledge of the problem, past experience, characteristics of the data, and data availability (Kulkarni et al., 2017). A simple and straightforward NN architecture suggested by Hegazy et al. (1994) is an NN with all inputs presented in the input layer and all outputs present in the output layer. Therefore, in this study the input parameters will consists of the 9 cost estimation attributes mentioned before. As the objective of the study is to estimate conceptual construction costs, the output layer in this study consists of one layer comprising one node.

Determining the appropriate number of hidden layers and the number of nodes in these hidden layers is important in feedforward NNs (Elbeltagi et al., 2014). This is important because the hidden layer extracts and remembers the features from the input patterns in order to predict the outcome of the NN (Günaydin & Doğan, 2004). Usually the number of hidden layers and number of hidden nodes is determined by trial and error (Hegazy et al., 1994; Günaydin & Doğan, 2004; Luu & Kim, 2009; ElSawy, 2011; Wang et al., 2012; Elbeltagi et al., 2014; Kulkarni et al., 2017). As the NN for this study will comprise only one hidden layer, only the number of nodes in this single hidden layer needs to be determined. The studies of Hegazy et al. (1994) and Magdum & Adamuthe (2017) provide rules-of-thumb to determine these nodes. According to Hegazy et al. (1994), the number of nodes in the hidden layer is: (1) 0.75m, (2) m, or (3) 2m+1, where m = number of nodes in the input layer. On the other hand, Magdum & Adamuthe (2017) provide the following rules-of-thumb in their study: (1) the number of hidden nodes should be in the range between the size of the input layer and the size of the output layer, (2) the number of hidden nodes should be 2/3 of the input layer size plus the size of the output layer, (3) the number of hidden nodes should be less than double the input layer size. For the determination of the hidden nodes of the NN architecture of this study, the mentioned rules-of-thumb will be used. This means that the nodes in the hidden layer can vary from 1 to 19, were 1 is the minimum and 19 the maximum.

As mentioned in *3.3.2 Training and learning of Neural Networks*, overfitting can be prevented by including features such as early stopping and a dropout layer into the NN code. Both will be implemented during training in the NN developed for this study.

As there is no specific rule in determining the NN architecture, this study will for conveniently start with an NN architecture of 9-9-1, were 9, 9, and 1 are the input nodes, hidden nodes, and output nodes respectively. The basic configuration of the NN architecture used during this study is given in Figure 17.

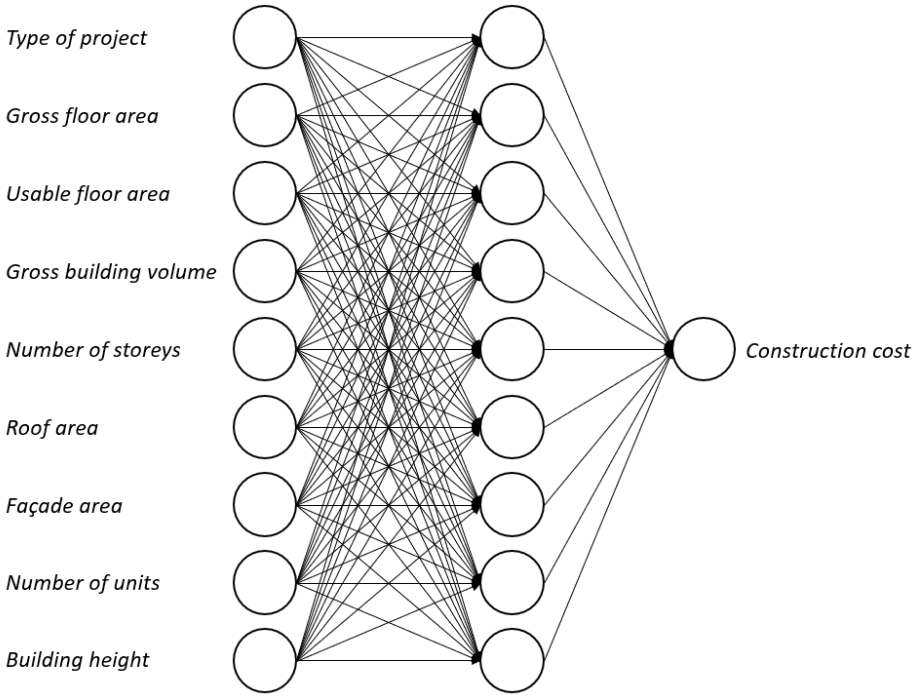


Figure 17: basic configuration of the NN architecture used during this study

It is recommended to experiment with the NN architecture, because different NN architectures can produce similar results. Therefore, this study will experiment with the NN architecture based on trial and error, in order to find the best performing NN architecture. This will be done by varying with the configuration of the number of nodes in the hidden layer.

The (leaky) ReLU activation function will be used as the activation function for the NN. Nowadays, the ReLU activation function is the most used activation function in ML models. However, ReLU has a disadvantage with regard to negative values. Therefore, it is decided to use leaky ReLU instead of the regular ReLU. This ensures that the NN model is also able to process (small) negative values. As stated, a feedforward backpropagation NN architecture will be used in this study. These FFBP NNs generally use a backpropagation learning algorithm. In the next paragraph, a more detailed explanation of this learning algorithm will be given.

4.3 Implementation phase

During the implementation phase, the NN paradigm selected for the problem will be coded and subsequently, the NN training and testing (validating) takes place (Hegazy et al., 1994). Coding the NN is one of the process steps during the implementation phase of NN modelling. During the implementation phase, the selected NN paradigm, in this study a feedforward backpropagation NN, will be coded. A summarisation of the coding process will be provided later on in the beginning of chapter 5 *(5.2 Model development)*.

Before constructing and utilising the NN during training and testing, it will be helpful to provide some background information about the mathematical representation of an NN. According to Magdum & Adamuthe (2017) the equation of an NN can be formulated as given in the equation below:

$$y = f(net) = f(\sum_{i=1}^{n}(X_i W_i + b)$$

In the equation, N is the number of inputs, X_i the input given to the processing element, W_i are the weights and b is a bias term. f is the activation function, which performs a mathematical operation on the signal output (Magdum & Adamuthe, 2017).

4.3.1 Structure the network

A supervised learning backpropagation process for NNs is illustrated in Figure 18 below. Before the learning (i.e. training) process can begin, a dataset with data about the input variables of the NN and their corresponding output values needs to be established. With this established dataset, the NN can be trained. A typical learning process for NNs comprises the following three steps carried out in an iterative sequence:

1. **Feedforward propagation**
   The input and correct output from the data are entered into the NN. Subsequently, the NN output is obtained and the error is calculated with the correct output.
2. **Backpropagation**
   The error contribution in each node is calculated. Subsequently, the weights are adjusted to reduce the network error.
3. **Repeating steps 2-3 for all training data**

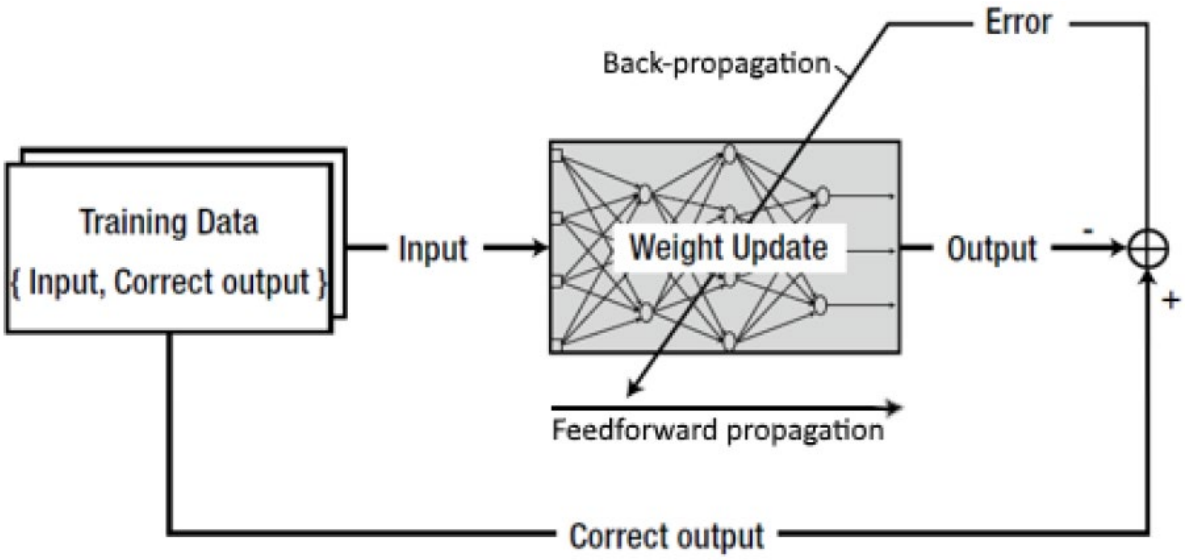


Figure 18: illustration of backpropagation process (Matel, 2019)

The training process of an NN begins with the principle of feedforward propagation. Before this can start, however, the weights of the NN need to be initialised. When first training an NN, weight initialisation is done randomly. Usually, the weights are set to small random values between -0.5 and 0.5. Afterwards, the feedforward propagation process can start. During this process, the input layers receive inputs from a datapoint and direct them to the hidden layers without calculations. The nodes in the hidden layers and output layers perform the computations of the NN, and add and adjust the weights. Every hidden node takes the weighted input of the previous node and outputs a single value based on a predefined activation function (Bosscha, 2016; Matel, 2019), which in the case of this study is the leaky ReLU activation function.

Once an NN has been established, the random weight configuration initially leads to an error between the output of the NN and the desired output. At this moment, the NN is useless for any estimation application. To make the NN useful, the weight configuration needs to be optimised in a way that the error between NN output and desired output is minimised. This can be done by applying backpropagation.

In most developed NNs in Construction Engineering and Management, the backpropagation (BP) learning algorithm is used. The BP algorithm, which belongs to the realm of supervised learning, is the most popular typology and learning algorithm for NNs (Hegazy et al., 1994; Hegazy & Ayed, 1998; Elhag & Boussabaine, 1998; Günaydin & Doğan, 2004; Kim et al., 2004; Arafa & Alqedra, 2011; Elbeltagi et al., 2014; Jain & Pathak, 2014; Waziri et al., 2017) due to its simplicity, easiness to code, generalisation capability, and nonlinear modeling performance (Elhag & Boussabaine, 1998; Günaydin & Doğan, 2004; Jain & Pathak, 2014; Waziri et al., 2017). As indicated, the BP algorithm is a supervised learning approach because the weights in the NN are adjusted using a number of training inputs and the corresponding target values (Arafa & Alqedra, 2011). BP incorporates a learning algorithm called the 'generalized delta rule'. This algorithm is responsible for training the NN usually over thousands of iterations (called epochs) (Rumelhart et al., 1986; Kim et al., 2004). The generalized delta rule algorithm uses a gradient descent approach to minimise the error between the estimated value and the desired target value (Rumelhart et al., 1986; Hegazy et al., 1994; Elhag & Boussabaine, 1998; Kim et al., 2004; Luu & Kim, 2009; Jain & Pathak, 2014). After each cycle, the network error (often observed with the Mean Squared Error) is backpropagated from the output layer to update the NN weights (Hegazy & Ayed, 1998; Luu & Kim, 2009; Arafa & Alqedra, 2011). This adjusting mechanism is repeated until the network reaches a certain level of accuracy (Arafa & Alqedra, 2011). This is done in order to establish a unique set of NN weights which enables the NN to produce outputs that are very close to the desired outputs (Rumelhart et al., 1986; Hegazy et al., 1994). When this is achieved, the NN is considered to be knowledgeable (Hegazy et al., 1994).

Learning rate and momentum are two important training parameters in BP. These terms are specified at the start of the training cycle and control the training speed and stability of the NN. With the learning rate, the weight modification during each training iteration can be controlled. The momentum term prevents oscillations to allow faster learning (Hegazy et al., 1994; Günaydin & Doğan, 2004). Hegazy et al. (1994) suggest that the learning rate and momentum can be set to 0.7 and 0.9 respectively. However, according to a study performed by Attoh-Okine (1999), the learning rate must be within 0.2 to 0.5 and the momentum within 0.4 to 0.5 for best NN performance. Günaydin and Doğan (2004) found that large learning rates often result in oscillations in weight changes and divergence of algorithm. Therefore, during this study, the learning rate and momentum will be set to the values as suggested by the study of Attoh-Okine (i.e. learning rate within 0.2 to 0.5 and momentum within 0.4 to 0.5).

4.3.2 Training and testing the network

During the training and testing phase, the NN will be trained and tested. Training requires the preparation of the data and NN training to establish an NN estimation model. Subsequently, the performance of the developed NN model in terms of accuracy is evaluated during testing.

Before the NN model can be trained, appropriate data splitting is required to ensure that the model will be trained with all data patterns present in the available data in order to predict meaningful results (Kulkarni et al., 2017). According to Günaydin & Doğan (2004), the training data should cover all spectrums of the available data. However, this is not a rule because there are no acceptable generalised rules established to determine the suitable size of the training data. Also, the study of Arafa & Alqedra (2011) found that training data evenly distributed over the entire range should allow NNs to achieve the target behaviour. Generally, 60-80% of the available data is used for training and the remaining for testing (Kulkarni et al., 2017; Magdum & Adamuthe, 2017). Training data will be used during the training process to find the relationships between input variables and output variables, while testing data will be used as validation data during testing to verify the NN performance (Kim et al., 2005).

To have a good training and testing mNNer, it is important to define the training and testing tolerance of the NN upfront. This tolerance value specifies how accurate the NN output must be considered correct during training and testing. Meaningful tolerances are usually specified as a percentage of the output range. A tolerance value of 0.1, for example, means that the NNs output value must be within 10% of the range of the actual output to be considered correct (ELSawy, 2011). In this study, the tolerance value will be set to 0.2 meaning that the NNs output value must be within 20% of the range of the actual output to be considered correct. In the following paragraph, it will be explained why a tolerance value of 20% is chosen.

The ability of NNs to be trained on historical data and learn from this data, is a major property that makes NNs superior to algorithmic and other AI-based systems (Hegazy & Ayed, 1998). NNs gain their problem-solving capability by learning from input and their corresponding output patterns. A training dataset contains training examples that consist out of a pair of input and corresponding output patterns (Hegazy et al., 1994). During a typical NN backpropagation training process, an NN will be trained over a number of training examples. At the start of the training process, a training dataset is presented to the NN with which the network calculates the output. The differences between the calculated outputs and the actual outputs are then evaluated and used to adjust the NN weights in order to reduce the difference. Usually, the difference between input and output mapping is reduced by the NN by minimising the mean squared error (MSE) of the NN (ElSawy, 2011). Thus, training an NN is a continuous process of adjusting the connection weights with a mathematical optimisation algorithm (i.e. backpropagation), until the NN weights reach a value at which the network error is minimal (Hegazy et al., 1994; Hegazy & Ayed, 1998; Günaydin & Doğan, 2004; ElSawy, 2011; Kulkarni et al., 2017). At this point, the NN generates its most accurate outputs. Adjusting the NN weights is important, because the accuracy of the NN depends on these weights (Hegazy & Ayed, 1998).

All NN models considered in this study will be trained in a supervised mode according to the backpropagation algorithm. A typical NN training procedure can be summarised by the following steps (Hegazy et al., 1994; Vahdani et al., 2012):

1. A training example is selected from the training dataset, applying the input vector to the network input buffer.
2. Network forward calculations are performed (i.e. feedforward propagation) on a layer-by-layer basis. The outputs of the nodes in a layer are used as inputs in the succeeding layer. Finally, the output layer produces the network outputs.
3. Network backward calculations (i.e. backpropagation) are performed at the output layer. Subsequently, the weight connections feeding the output layer are adjusted.
4. Network backward calculations (i.e. backpropagation) are performed for the hidden layers, starting at the last hidden layers (the layer just before the output layer). Subsequently, the weight connections feeding the hidden layers are adjusted. Backward calculations are continued until the weights feeding the first hidden layer are adjusted.
5. Steps 1 through 4 are repeated for all training examples and the errors with respect to all outputs and all examples are added. The average MSE of the NN is then calculated. By the end of this step, one training iteration is completed.
6. If the calculated MSE in step 5 does not satisfy the set target error, a new training iteration can be started starting from step 1.

Training must be stopped when the MSE remains unchanged for a given number of epochs, or when the MSE starts to increase (Günaydin & Doğan, 2004; Arafa & Alqedra, 2011). Training must be stopped at this point to avoid overtraining. The NN memorises the training values and will be unable to make meaningful predictions when presented with new unknown data. During training, NNs tend to overfit. Another challenging task during training is to obtain an NN that ensures generalization (Arafa & Alqedra, 2011). By applying validation, i.e. enabling the NN to explain unseen data during testing, overfitting can be avoided and generalization can be achieved (Kim et al., 2005; Kulkarni et al., 2017).

In this study, validation will be done by providing a test dataset to the NN. By manually separating the data into a training and test dataset, this test dataset for validation will be created. As mentioned in *4.2.2 Data gathering and representing*, a total of 50 data samples can be used in this study for NN training and testing. Of these 50 samples, the 35 simulated data samples (i.e. 70% of the available data) will be used for NN training, while the original 15 data samples (i.e. 30% of the available data) will be used for NN testing.

Overfitting and generalization issues in NNs can also be improved by using a simple NN architecture and by using more data for training and testing (Elbeltagi et al., 2014; Kulkarni et al., 2017). Elbeltagi et al. (2014) state that the number of patterns present in the training and test data significantly influences the generalization ability of an NN, because using a higher number of training patterns probably increases the ability of the NN to learn and achieve more accurate results.

Once training is completed, the NN needs to be tested with unseen data to evaluate the accuracy of the developed NN model (Kulkarni et al., 2017). In essence, testing an NN is similar to training it. However, during testing the NN is presented with data it has never seen before and no weight adjustments are made. Testing the NN is of importance, because testing ensures that the developed NN was successfully trained and an adequate generalisation is achieved (Arafa & Alqedra, 2011). According to Hegazy et al. (1994), NN generalization is the most important performance measure considered during NN testing. Satisfactory generalization is achieved when the NN responds well to data outside the training set. If the NN testing results are satisfactory, the NN will be ready to use. If not, the NN needs more or better data, or in the worst case the NN architecture needs to redesigned (ElSawy, 2011).

4.4 Application phase

During the application phase, the developed NN model will be applied in order to perform its required task, which for this study is the estimation of construction costs. The estimation performance of the NN model is evaluated by comparing the estimated construction cost with the actual construction costs and calculating the error with the help of error functions.

Applying the network for estimations

Once the NN model is constructed, trained, and tested, the model can be applied for cost estimations. The results of the estimations provided by the NN models need to evaluated in order to measure the accuracy of the NNs and to find the best performing NN architecture (Elfahham, 2019). Estimation results are often evaluated and measured in terms of accuracy (Kim et al., 2004; Waziri et al., 2017), because prediction accuracy is one key advantage of NNs over traditional methods. This adds to its popularity and usage (Waziri et al., 2017). According to Waziri et al. (2017), prediction accuracies of NNs can get as high as 98% as observed by Waziri et al. in the studies of Geiger et al. (1998), Alqahtani & Whyte (2013), and Bala et al. (2014).

The objective of this study is to estimate the conceptual construction costs as accurately as possible. According to Phaobunjong (2002), the expected accuracy for a definitive estimate of a construction project is within a range of -5% to +15%, and the range of a conceptual stage estimate should be within -30% to +50%. While AbouRizk et al. (2002) state that a conceptual cost estimation for a construction project should be within a range of -15% to +25%. According to Elbeltagi et al. (2014), 20% is a typically expected error percentage for conceptual cost estimates. Therefore, in this study, the estimates made with the NN are considered accurate when they fall within a range of 20% of the actual construction costs.

In most of the works that used NNs for estimation, the performance of NNs was evaluated through mean squared error (MSE), mean absolute error (MAE), and mean absolute percentage error (MAPE)

(Günaydin & Doğan, 2004; Kim et al., 2004; Arafa & Alqedra, 2011; Vahdani et al., 2012; Kulkarni et al., 2017; Waziri et al., 2017; Elfahham, 2019). Note that smaller values of the MSE and MAE indicate better accuracy (Elfahham, 2019). In this study, the NN model's performance in terms of accuracy will be evaluated by using the following four performance indicators:

1. Mean Squared Error (MSE)
2. Mean Absolute Error (MAE)
3. Mean Absolute Percentage Error (MAPE)
4. Average accuracy

The equations of these performance indicators are presented below. In this equation, *N* is the total number of training or test datasets, *i* is the project number, A_i the actual construction cost, and P_i the predicted construction costs.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(A_i - P_i)^2$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}(A_i - P_i)$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{(A_i - P_i)}{A_i} \times 100\%$$

$$Average\ accuracy = 100\% - MAPE$$

4.5 Conclusion

In this conclusion, the answer to the chapter-related research question: "how can (conceptual) cost estimations be performed with ANNs?" will be provided.

In order to develop an NN that performs cost estimations, three phases and six basic steps need to be followed and carried out (Figure 19):

Phase I: Modeling phase

1. Problem definition
2. Data gathering and representing
3. Defining the network

Phase II: Implementation phase

4. Structuring the network
5. Training and testing the network

Phase III: Application phase

6. Applying the network for estimations

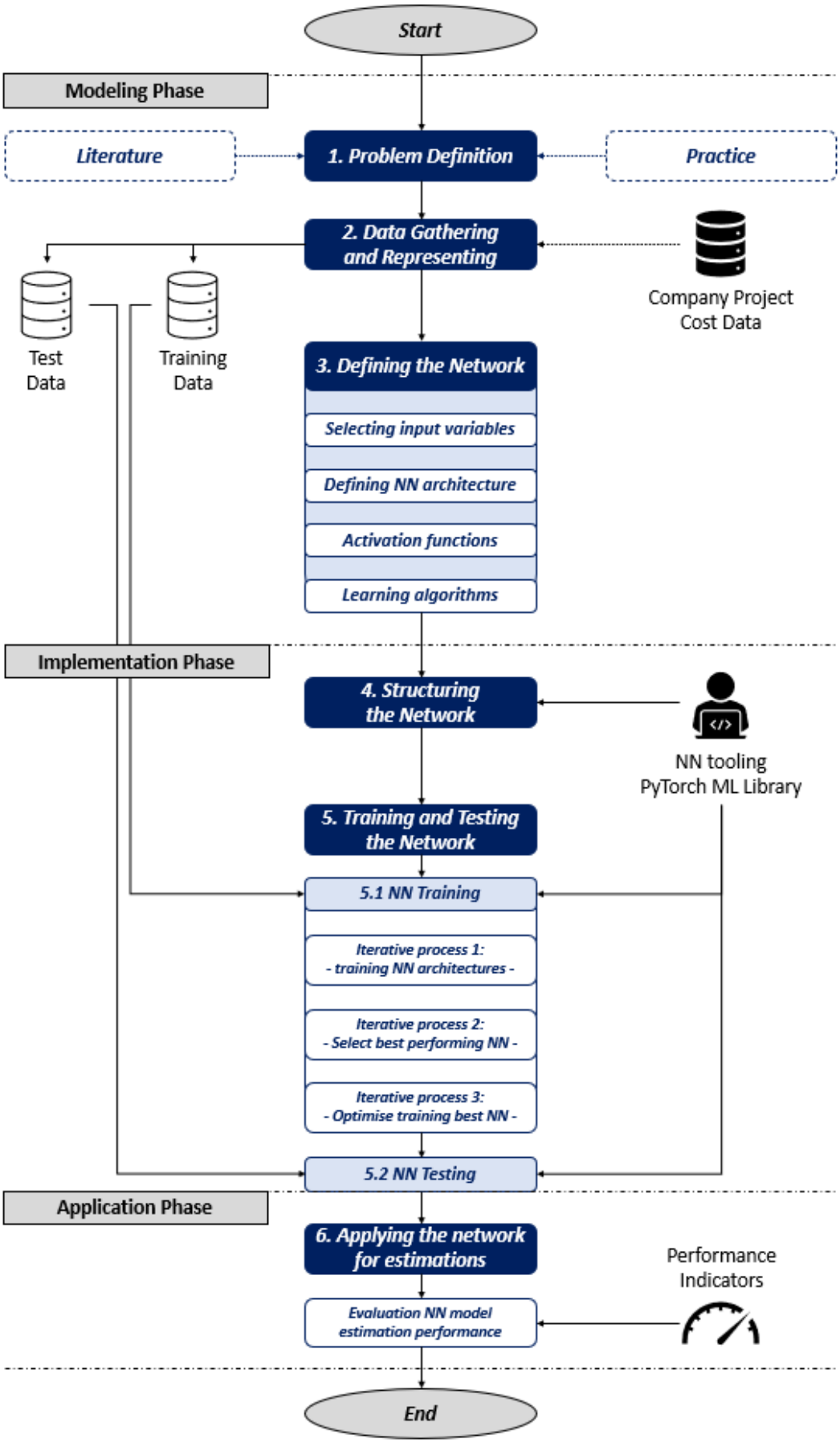


Figure 19: visualisation of method to be followed for NN development

By following the phases and steps illustrated in Figure 19, an FFBP NN was created with the following architecture: 9-9-1, where 9, 9, and 1 are the input nodes, hidden nodes, and output nodes respectively. Since different NN architectures can produce similar results, experiments will be conducted with the configuration of the NN architecture based on trial and error in order to find the best performing NN architecture. The constructed NN uses leakyReLU as the activation function and backpropagation as learning algorithm. In order to perform the cost estimations, the developed NN will be trained with 35 data samples (i.e. training data) and subsequently, the NN performance will be tested with 15 data samples (i.e. testing data). During testing, the NN will be utilised to make construction cost predictions based on unseen data. The prediction performance of the NN will be evaluated during testing with the help of performance indicators such as: MSE, MAE, MAPE, and average accuracy. Estimates made with the NN are considered accurate, as they fall within a range of 20% of the actual construction costs. This means that the developed NN needs to reach at least an average accuracy of 80% to be considered accurate.

Note that data is of high importance for NNs to provide meaningful output. The performance of NNs strongly depends on the quality and representativeness of the data. However, NNs are capable of dealing with incomplete and noisy data, thus datasets do not have to be complete. It is important to realise that NNs can only deal with numerical data. On the other hand, qualitative data can be used as well but it needs to be converted into a numeric form first. For this study, data of 15 construction projects were gathered from a construction firm in the Netherlands. The gathered data comprises numeric project cost data of residential housing projects and multi-storey housing projects. With the help of a data simulation tool constructed in Python, new data was created based on the data of the 15 projects. This resulted in 35 new data samples that will be used for NN training. The original 15 data samples will be used for NN testing. From the project cost data, 9 cost attributes were derived; these will serve as input variables for the NN.

Now that the NN is constructed, it can be trained, tested, and the estimation performance can be evaluated. Results of the NN training and testing process will be described in the upcoming chapter. Also, in the next chapter the coding process of the NN is summarised.

5. Tool development and results

In this chapter, the proposed NN as described in the methodology will be trained and tested. The results of the NN training and testing will be presented in this chapter, as well as a summary of the NN coding process in Python. In the conclusion of this chapter, the answer to the following research question will be answered: "how can ANN-based cost estimation be used for construction cost estimating if limited data is available?"

First, a data analysis of the training and test data will be provided. Subsequently, the NN coding process in Python will be discussed, after wich the results of NN training and testing will be provided. Finally, a conclusion will be drawn on the found results.

5.1 Data analysis

This paragraph provides an analysis of the data that will be used in this study for NN training and testing. The total dataset for training and testing contains data of 50 projects in total. Of the 50 data samples, 15 samples were gathered from a Dutch construction firm and 35 samples were simulated by using a data simulation code developed in Python. According to what is described in *4.3.2 Training and testing the network*, 70% of the data in the total dataset will be used for NN training and the remaining 30% will be used for NN testing. For this study, the data splitting results in 35 data samples for training (i.e. the simulated data samples) and 15 data samples for testing (i.e. original gathered data samples). Below, a detailed analysis of the originally gathered test data and the simulated training data will be provided. First, the analysis of the originally gathered test data will be provided and subsequently, the analysis of the simulated training data will be provided. The data analysis is provided in this way in order to be able to compare the originally gathered data with the simulated data, in such a way that it can be verified that the simulated data largely corresponds to the actual data.

5.1.1 Analysis of test data

The data that will be used in this study for NN testing consists of 15 data samples. These 15 data samples are similar to the data gathered from the Dutch construction firm and are discussed in *4.2.2 Data gathering and representing*. Of these data samples, 5 samples are related to residential housing projects and the remaining 10 are related to multi-storey housing projects. In Table 6, statistics such as the number of samples, minimum, maximum, mean and standard deviation are given. In Table 7, the correlation coefficients of these data samples are provided. Subsequently, scatterplots are provided to analyse the data patterns (i.e. linearity and relativity) in the testing data.

Table 6: statistics of test data (i.e. original data)

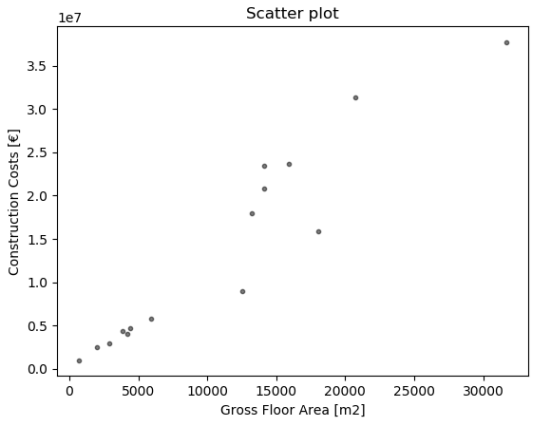
Variable	N	Min	Max	Mean	Std. Dev.
Type of Project	15	-	-	-	-
Gross Floor Area	15	658	31718	10957	8640,048
Usable Floor Area	15	526	25374	8563	6815,245
Gross Building Volume	15	1657	71047	26106	19603,39
Number of Storeys	15	3	24	10	7,936
Roof Area	15	244	8194	2225	2276,803
Façade Area	15	714	12452	5412	3829,725
Number of Units	15	9	233	91	75,413
Building Height	15	10	77	41	25,4148
Construction Costs	15	991331	37710000	13678272	11667019

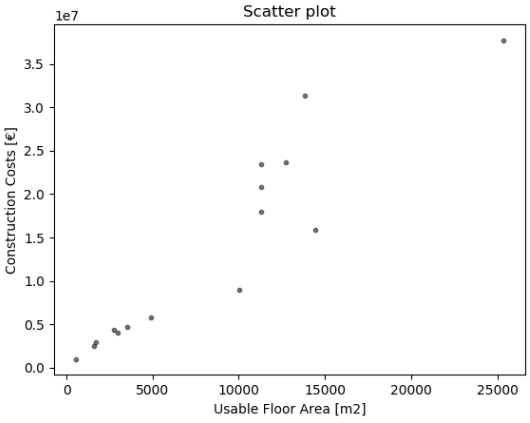
Table 7: correlation coefficients of test data

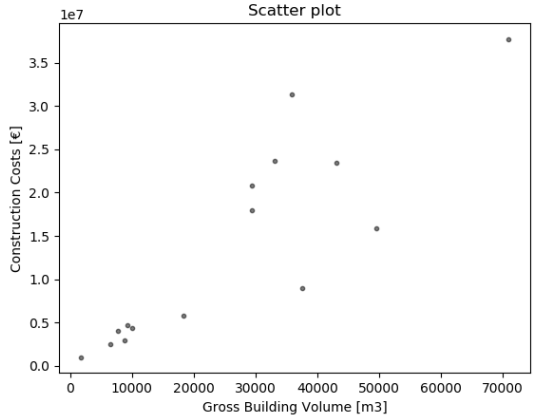
	GFA	UFA	GBV	NoS	RA	FA	NoU	BH	CC
GFA	-	0,994	0,960	0,615	0,391	0,798	0,910	0,707	0,950
UFA	0,994	-	0,971	0,587	0,409	0,772	0,908	0,679	0,932
GBV	0,960	0,971	-	0,485	0,529	0,776	0,883	0,654	0,868
NoS	0,615	0,587	0,485	-	-0,316	0,599	0,770	1,000	0,814
RA	0,391	0,409	0,529	-0,316	-	0,481	0,173	0,306	0,121
FA	0,798	0,772	0,776	0,599	0,481	-	0,761	0,919	0,773
NoU	0,910	0,908	0,883	0,770	0,173	0,761	-	0,791	0,960
BH	0,707	0,679	0,654	1,000	0,306	0,919	0,791	-	0,823
CC	0,950	0,932	0,868	0,814	0,121	0,773	0,960	0,823	-

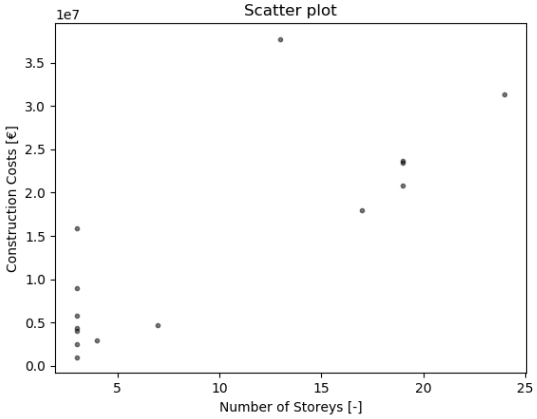
It can be derived from the table with the correlations coefficients that there are three variables that have a correlation coefficient of ≥ 0.9, one variable with a correlation coefficient of ≥ 0.85, two variables with a correlation coefficient of ≥ 0.80, and two variables with a correlation coefficient < 0.8. As expected, the variables with a linear relationship to the construction cost have a higher correlation coefficient compared to variables that are expected to have a nonlinear relationship to the construction costs.

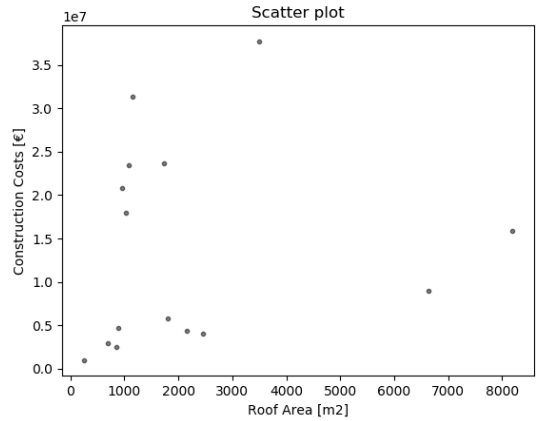
The scatterplots of the test data shown below reveal that there is some linearity and relativity in the data. The linearity of the data is mainly present in the data of projects with construction costs ranging up to 10 million. However, there is also some nonlinearity present in the data. This is especially evident in the scatterplots of the variables roof area and façade area. Also, the scatterplots indicate that the data range of the individual variables is not entirely covered, as discovered earlier. The test data can still be used, since NNs can deal with nonlinear, incomplete, and noisy datasets as indicated in *4.2.2 Data gathering and representing*.

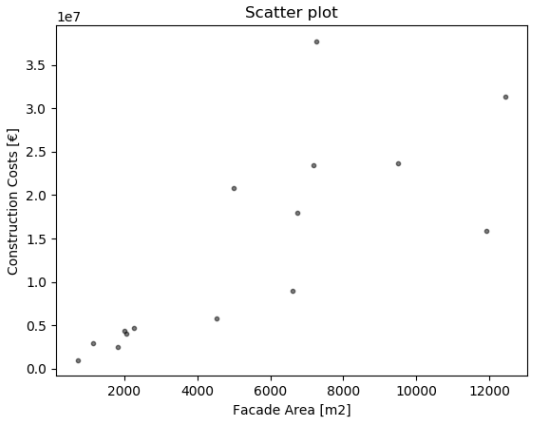


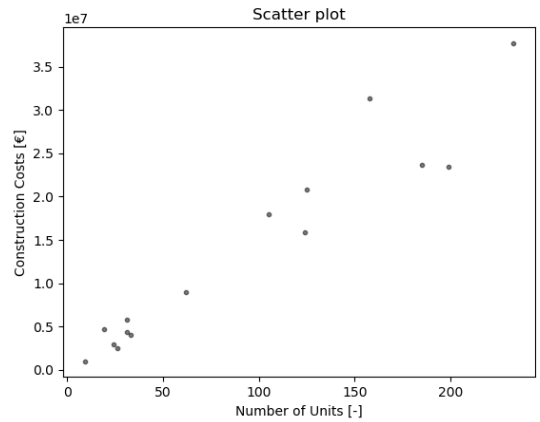


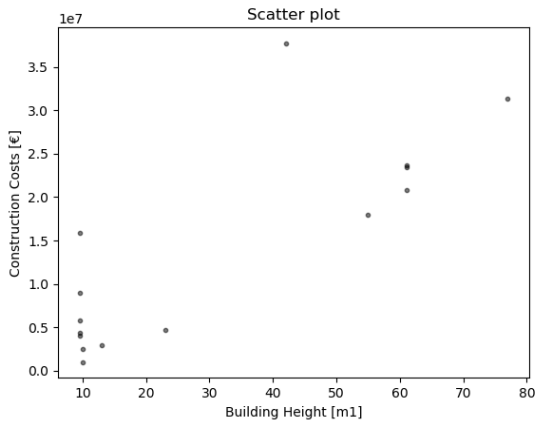












5.1.2 Analysis of simulated training data

The data that will be used in this study for NN training consists of 35 data samples which were simulated based upon the 15 gathered data samples from a Dutch construction firm. Of these data samples, 14 samples are related to residential housing projects and the remaining 21 are related to multi-storey housing projects. In Table 8, statistics such as number of samples, minimum, maximum, mean, and standard deviation are given and in Table 9 the correlation coefficients of these data samples are provided. Subsequently, scatterplots are provided to analyse the data patterns (i.e. linearity and relativity) in the training data.

Table 8: statistics of training data (i.e. simulated data)

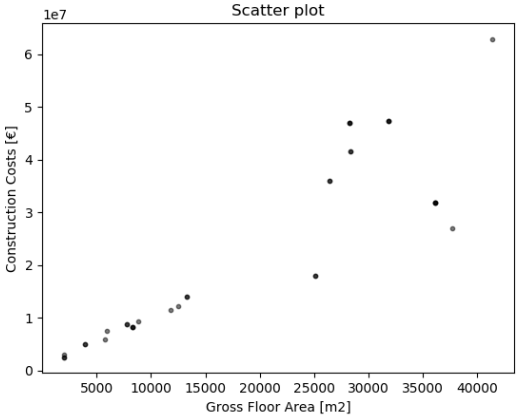
Variable	N	Min	Max	Mean	Std. Dev.
Type of Project	35	-	-	-	-
Gross Floor Area	35	1974	41404	19700,77	12879,2
Usable Floor Area	35	1578	30156	15540,91	10270,3
Gross Building Volume	35	4971	112815	49308,80	33926,8
Number of Storeys	35	3	48	16,06	16
Roof Area	35	732	19887	5436,86	5605
Façade Area	35	1819	24904	11088,26	7646,3
Number of Units	35	26	398	168,86	129,5
Building Height	35	10	154	56,14	47,8
Construction Costs	35	2517868	62778506	23579829,97	17606452,3

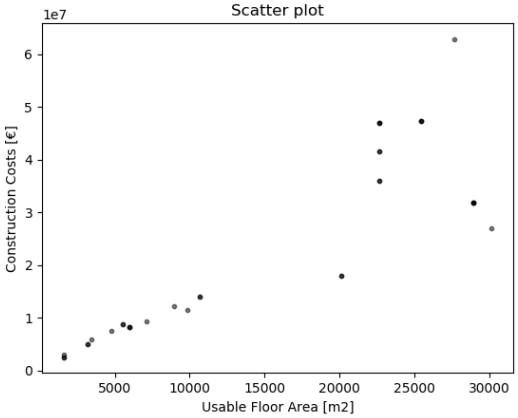
Table 9: correlation coefficients of training data

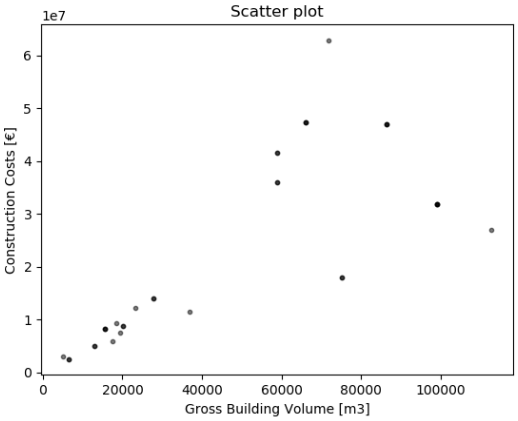
	GFA	UFA	GBV	NoS	RA	FA	NoU	BH	CC
GFA	-	0,995	0,948	0,495	0,536	0,959	0,843	0,541	0,885
UFA	0,995	-	0,958	0,487	0,543	0,951	0,842	0,531	0,872
GBV	0,948	0,958	-	0,326	0,657	0,924	0,790	0,370	0,771
NoS	0,495	0,487	0,326	-	-0,453	0,352	0,738	0,996	0,817
RA	0,536	0,543	0,657	-0,453	-	0,613	0,139	-0,400	0,098
FA	0,959	0,951	0,924	0,352	0,613	-	0,782	0,393	0,801
NoU	0,843	0,842	0,790	0,738	0,139	0,782	-	0,763	0,956
BH	0,541	0,531	0,370	0,996	-0,400	0,393	0,763	-	0,845
CC	0,885	0,872	0,771	0,817	0,098	0,801	0,956	0,845	-

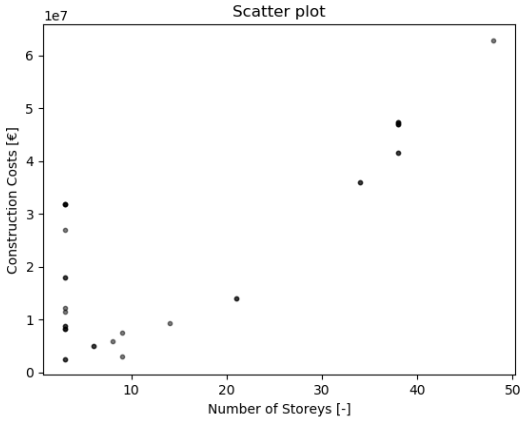
It can be derived from the table with the correlations coefficients that there is one variable that has a correlation coefficient of ≥ 0.95, two variables with a correlation coefficient of ≥ 0.85, three variables with a correlation coefficient of ≥ 0.80, and two variables with a correlation coefficient < 0.8. Again as expected, the variables with a linear relationship to the construction cost have a higher correlation coefficient when compared to the variables that are expected to have a nonlinear relationship to the construction costs.

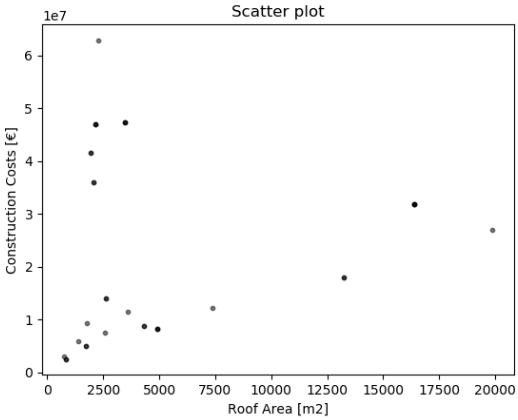
The scatterplots of the test data shown below reveal that there is some linearity and relativity in the data. The linearity in the data is mainly present in the data of projects with construction costs ranging up to 15 million. However, there is also some nonlinearity present in the data. This is especially present in the scatterplots of the variables roof area and façade area. Also, the scatterplots indicate that the data range of the individual variables is not entirely covered, as discovered earlier. The training data can still be used, since NNs can deal with nonlinear, incomplete, and noisy datasets as indicated in *4.2.2 Data gathering and representing*.

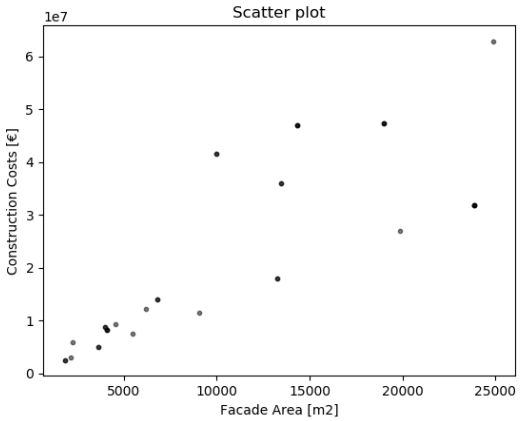

Scatter plot
1e7
Construction Costs [€]
Gross Floor Area [m2]

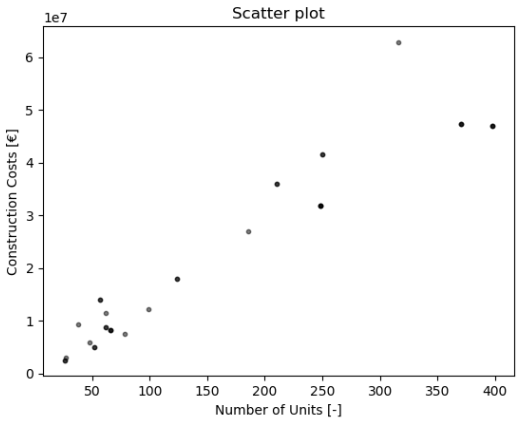

Scatter plot
1e7
Construction Costs [€]
Usable Floor Area [m2]

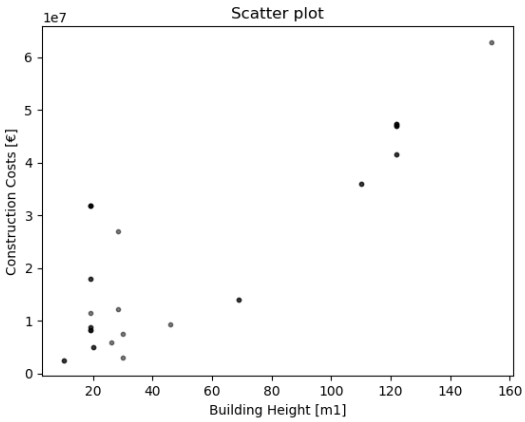

Scatter plot
1e7
Construction Costs [€]
Gross Building Volume [m3]


Scatter plot
1e7
Construction Costs [€]
Number of Storeys [-]


Scatter plot
1e7
Construction Costs [€]
Roof Area [m2]


Scatter plot
1e7
Construction Costs [€]
Facade Area [m2]


Scatter plot
1e7
Construction Costs [€]
Number of Units [-]


Scatter plot
1e7
Construction Costs [€]
Building Height [m1]

When the simulated training data is compared to the testing data that was gathered from a Dutch construction firm, it can be concluded that the simulated data is almost similar in terms of data patterns and correlation coefficients. Despite of the correlation coefficients of the simulated data being somewhat lower than the original data, they still show the same pattern as the original data. Also, from the scatterplots can be derived that the simulated data follows a pattern that is similar to the data pattern of the originally gathered data. Therefore, it can be stated that the simulated data somewhat represents the reality. However, it still remains simulated data. Thus, one should be careful with interpreting the results of this study. The results could differ if actual project cost data was used. Still, the simulated training data can be used for this study to prove the concept of NNs and to validate the established method.

5.2 Model development

In this paragraph, the NN development coding process will be described. Coding the NN is one of the process steps during the implementation phase of NN modeling. During the implementation phase, the selected NN paradigm (in this study a feedforward backpropagation NN) will be coded. As mentioned earlier, the NN will be coded using two different ML libraries for NN model building and training: 1. TensorFlow Keras, and 2. PyTorch. Using these two ML libraries results in the development of two individual different NNs. However, only one of the two developed NNs will be used for the remainder of this study. In the next paragraph, the choice for one of the two NNs will be motivated. The full code of both developed NNs can be found in Appendix 4.

5.2.1 Choice of NN ML library

Two ML libraries (i.e. TensorFlow and PyTorch) have been used in this study for the development and coding of the NN. Two NN models were coded using the two different ML libraries, however, only one NN model will be used. Based on the first experimentations in terms of training, testing, and predicting with both NN models, the most appropriate NN model will be selected. Subsequently, the chosen NN model will be used for this study. The chosen NN model will then be trained with the training data and with the test data, the NN will be utilised to make predictions. In order to better evaluate the prediction performance of the selected NN model, the predictions of the NN model will be assessed by using performance indicators such as MSE, MAE, MAPE and average accuracy. With the help of these performance indicators, the best performing NN model will be selected by the end of this study.

TensorFlow NN model

It is relatively easy to code an NN in Python with the TF Keras ML library for deep learning models. The coding process can be completed relatively quickly and the code is easy to understand. Also, coding the NN does not require a high amount of lines of code; only preparing the model in order to make predictions on unseen data (i.e. test data) requires a bit more coding. These are all arguments in favour of using the TF NN model. However, during the first experiments with the TF NN model it was found that the model is not stable during training and in making predictions. The results of the model fluctuate and sometimes, for unclearable reasons, it is not able to train itself and make predictions. Therefore, due to the limitations found during initial experiments, this model is considered unsuited for further use during this study.

PyTorch NN model

It requires a bit more lines of code and coding skills to code an NN in Python with the PyTorch ML library and the torch.nn module of PyTorch, when compared to coding with tf.keras. Still, the NN is relatively fast and easy to code. In order to simplify, two codes were written in Python for the NN. One for the training of the NN and the other for NN model evaluation. Because of these two codes, training the NN is done separately from NN predicting. However, both models use the same NN architecture, as the trained model is saved and is used for NN evaluation and predicting in the evaluation model. The initial experiments with the PyTorch NN model showed that the developed model remains stable

during training and prediction. The results of the model fluctuate way less when compared to the TF NN model. However, using a learning rate between 0.2 and 0.5, as suggested in the methodology, resulted in an NN model that did not learn. This problem was solved by adjusting the learning rate to 0.001 and adding a weight decay parameter. After these adjustments, it was found that the PyTorch model was able to train itself and make predictions without showing any noticeable errors. Based on the good experiences with the NN model during initial experimenting, it can be stated that this NN model is suitable for further use during the study. Therefore, the NN model developed with the PyTorch ML library will be selected as NN model, which will be used for the remainder of this study.

5.2.2 Model development with PyTorch

The NN model that will be used in this study is developed with the PyTorch ML library and the associated torch.nn ML module for NN models from PyTorch. Below, the coding process of this NN model will be explained with a summary of the coding process. To simplify, there a two codes written in Python: one for the training of the NN and the other for NN model evaluation.

Training model

First, the functions and classes needed to develop the NN model need to be imported into the working environment. For the development of the NN model, the following libraries will be used: NumPy, Pandas to load the data, PyTorch to define the NN model, and Matplotlib to plot graphs.

```
1  # Importing libraries
2  import torch
3  from torch.autograd import Variable
4  import torch.nn.functional as F
5  import torch.utils.data as Data
6
7  import matplotlib.pyplot as plt
8
9  import pandas as pd
10 import numpy as np
```

Secondly, the NN model can be defined. In this study, the NN model will be defined by using the torch.nn module from the PyTorch library. In PyTorch, NN models are usually defined as a sequence of layers. A sequential model will be created for this study as well. The first thing to do is to ensure that the input layer has the right number of input variables. By using the argument torch.nn.Linear(9, 9), a module for linear transformation consisting of 9 inputs and 9 outputs is created. Subsequently, the argument torch.nn.LeakyReLU specifies that this layer in the NN uses the leakyReLU activation function. By using these arguments, an NN can be constructed. In this study, the NN consists of 1 input layer, 1 hidden layer, and 1 output layer. The input layer consists of 9 nodes, equal to the amount of input variables used in this study. The hidden layer in the NN consists of 9 nodes. The output layer consists of 1 node, equal to the output variable considered in this study.

```
1  # Defining the NN model
2  def create_model():
3      return torch.nn.Sequential(
4              torch.nn.Linear(9, 9),
5              torch.nn.LeakyReLU(),
6              torch.nn.Dropout(0.5),
7
8              torch.nn.Linear(9, 1),
9  )
```

Now that the NN model architecture is defined, the model needs to be prepared for training. Preparing the model for training means loading the training data, calculating the means and standard deviations in order to normalize the training data, splitting the data into input and labels (i.e. output variables), creating tensors to pass the data through the network, and defining an NN optimiser in order to optimise the weight adjustment during backpropagation.

```
1  # Preparing the model for training
2  def train(model, csv_name, num_epochs=100, batch_size=5):
3
4  # Read training data from CSV
5  dataframe = pd.read_csv (csv_name, delimiter=",", header=None)
6  dataset = dataframe.values
7
8  # Calculate Means and Standard Deviations
9  means = np.average(dataset, axis=0)[None, :]
10 std = np.std(dataset, axis=0)[None, :]
11
12 # Normalize the dataset
13 dataset = (dataset – means) / std
14
15 # Split into input and labels
16 x = dataset[:,0:9]
17 y = dataset[:,9]
18
19 # Create Tensors for usage with PyTorch
20 x, y = torch.Tensor(x), torch.Tensor(y)
21
22 optim = torch.optim.Adam(model.parameters(), lr=0.35, m=0.45, weight_decay=0.001)
23 criterion = torch.nn.MSELoss()
24
25 dataloader = Data.DataLoader(
26       dataset=Data.TensorDataset(x, y),
27       batch_size=20,
28       shuffle=True
29 )
```

Now that the model is prepared for training, it can be trained. During NN training, the network is provided with training data, based on which, the NN calculates its output and based on the difference between the NN output and the actual output, the error (i.e. loss) is calculated. Based on the loss, the NN weights are adjusted after a given number of epochs in order to train a NN model with minimal loss.

```
1  # Training the model
2  model.train()
3
4  epoch.losses = []
5  for epoch in range(num_epochs):
6      losses=[]
7      for iter, (b_x, b_y) in enumerate(dataloader):
8          prediction = model(b_x)
```

```
9
10          loss = criterion(prediction, b_y.unsqueeze(-))
11
12          losses.append.(loss.item())
13          optim.zero_grad()
14          loss.backward()
15          optim.step()
16
17      epoch_losses.append(np.average(losses))
18      if epoch %  == 0:
19        print("Epoch #:", epoch, "Loss=", np.average(epoch_losses[-5:]))
20
21  return means, std
```

After training, the NN model will be saved. Subsequently, the saved trained NN model will be used during evaluation in order to validate its performance on unseen data.

Evaluation model

After the NN model is trained, it needs to be assessed to validate its performance in terms of accuracy. In this study, model evaluation is done by using a testing dataset including data that is unknown for the trained NN. The model evaluation will be performed by running a different code in Python (i.e. evaluation code). This code will be discussed below.

Since the evaluation code uses the same NN architecture as the trained NN model, this NN architecture needs to be defined in the code of the evaluation model as well. The coding process of this NN is similar to the coding of the NN in the training model.

```
1  # Defining the NN model
2  def create_model():
3       return torch.nn.Sequential(
4                torch.nn.Linear(9, 9),
5                torch.nn.LeakyReLU(),
6                torch.nn.Dropout(0.5),
7
8                torch.nn.Linear(9, 1),
9  )
```

The trained NN model needs to be prepared for testing before it can be tested with help of the evaluation code. Preparing the model for evaluation is basically the same as preparing the model for training. However, no optimizer is defined in the evaluation model, because during testing the weights of the NN model will not be adjusted to reach a better performance. Model preparation during evaluation now means loading the test data, calculating the means and standard deviations in order to normalize the test data, splitting the data into input and labels (i.e. output variables), and creating tensors to pass the data through the network.

```
1  # Preparing the model for evaluation
2  def evaluate(model, csv_name, means, std)
3
4  # Read evaluation date from csv
5  dataframe = pd.read_csv(csv_name, delimiter=",", header=None)
```

```
dataset = dataframe.values

# Normalize the dataset
dataset = (dataset – means) / std

# Split into inputs and labels
x = dataset[:,0:9]
y = dataset[:,9]

# Create Tensors for usage wit PyTorch
x, y = torch.Tensor(x), torch.Tensor(y)

dataloader = Data.DataLoader(
        dataset=Data.TensorDataset(x, y),
        batch_size=1
)
```

Now that the model is prepared for testing, it can be tested. During testing, the NN model will be provided with data it has not seen before. Based on this data, the NN will make predictions. These predictions will be compared to the actual values in the testing dataset. Based on the difference between the actual value and the predicted value the mean error of the NN model can be determined.

Predictions with the evaluation model are made by lines of code aimed at making predictions. In order to make predictions, these lines of code use the input variables (i.e. x variable) specified in the test data. Subsequently, the predictions are compared to the actual values of the construction cost in the test data (in the code specified as ground_truths) to calculate the error (i.e. difference between actual and predicted value). In the code, the error is given by the term 'mean error'. The mean error is calculated in accordance with the principle of calculating the performance indicator MAE.

```
# Model evaluation
model.eval()

print("All values are in millions.")

ground_truths = []
predictions = []
for iter, (b_x, b_y) in enumerate(dataloader):
    prediction = model(b_x)
    b_y          = (b_y[0].item()          * std[0, -1]) + means[0, -1]
    prediction = (prediction[0].item() * std[0, -1]) + means[0, -1]

    b_y          *= 1e-6
    prediction *= 1e-6

    ground_truths.append(b_y)
    predictions.append(prediction)

    print("Ground Truth=" + str(b_y), "  \tPrediction=" + str(prediction))

print("Mean Error = ", np.mean(np.abs(np.array(ground_truths) – np.array(predictions))))
```

Now that the NN model is developed in Python, this network (i.e. PyTorch NN model) can be trained with the training data first. Subsequently, the performance of the trained NN will be tested. In the next paragraph, the results of the NN training will be provided and discussed. In the paragraph thereafter, the results of NN testing will be provided and discussed.

5.3 Results of NN training

In this paragraph, the results of the NN training will be discussed. During training the NN will be presented to a training dataset. Note that this training dataset consists of the 35 simulated data samples and therefore one should interpret the training results carefully, since the simulated data does not represent the reality. Based on the training data, the NN will learn the patterns in the data eventually in order to be able to make predictions with limited error. During training, error is observed in terms of loss, the lower the loss the better the NN is trained. Since the results of NN training can vary over different NN architectures, it was decided to perform NN training for different NN architectures in order to find the best performing NN architecture. In total, 19 different NN architectures will be trained.

A first analysis of a typical training run revealed that the training (i.e. learning) pattern of the NN is noisy, as can be seen in Figure 20. Normally, the learning process is smoother and the loss decreases according to the gradient descent principle as indicated by the red line in Figure 20. The noisy learning pattern of the NN is probably due to the limitations in the (training) dataset, as the data is limited in amounts, does not cover the entire range, and has outliers. Due to this discovery, applying an early stopping function to the code of the NN does not work, as the training process will then probably be stopped whilst the model is still learning. Therefore, determining the best performing NN during training will be done based on the average loss.

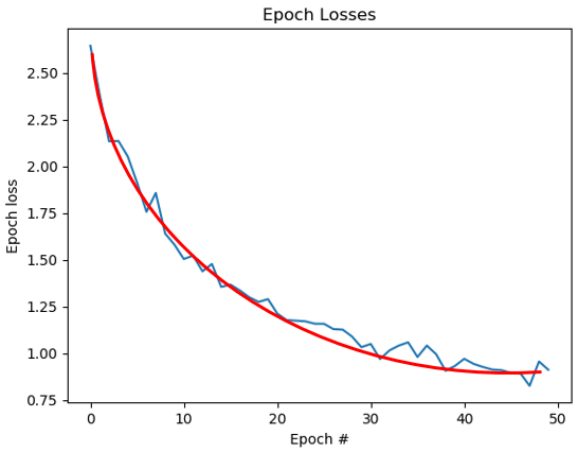


Figure 20: first analysis of learning pattern developed NN

The first NNs will be trained using the MSE Loss criterion. For training these NNs, the learning rate, momentum, and weight decay are set to values of 0.001, 0.45, and 0.001 respectively. The NNs will be trained over a different number of epochs, it is arbitrarily chosen to train the NNs over 50, 100, and 150 epochs respectively. The results of the 19 NNs trained with the MSE Loss criterion can be found in Table 10.

Table 10: training results of NNs trained with MSE Loss criterion

Model #	NN Arch	Loss criterion	LR	M	WD	Epoch = 50	Epoch = 100	Epoch = 150
						Avg. Loss	Avg. Loss	Avg. Loss
1	9-1-1	MSE Loss	0,001	0,45	0,001	1,291	0,872	0,747
2	9-2-1	MSE Loss	0,001	0,45	0,001	1,009	0,781	0,816
3	9-3-1	MSE Loss	0,001	0,45	0,001	0,619	0,646	0,664
4	9-4-1	MSE Loss	0,001	0,45	0,001	0,980	0,657	0,564
5	9-5-1	MSE Loss	0,001	0,45	0,001	0,689	0,546	0,481
6	9-6-1	MSE Loss	0,001	0,45	0,001	0,688	0,425	0,544
7	9-7-1	MSE Loss	0,001	0,45	0,001	0,512	0,442	0,381
8	9-8-1	MSE Loss	0,001	0,45	0,001	0,718	0,407	0,384
9	9-9-1	MSE Loss	0,001	0,45	0,001	0,552	0,536	0,362
10	9-10-1	MSE Loss	0,001	0,45	0,001	0,394	0,374	0,480
11	9-11-1	MSE Loss	0,001	0,45	0,001	0,478	0,384	0,352
12	9-12-1	MSE Loss	0,001	0,45	0,001	0,482	0,313	0,441
13	9-13-1	MSE Loss	0,001	0,45	0,001	0,498	0,333	0,352
14	9-14-1	MSE Loss	0,001	0,45	0,001	0,443	0,325	0,326
15	9-15-1	MSE Loss	0,001	0,45	0,001	0,409	0,315	0,280
16	9-16-1	MSE Loss	0,001	0,45	0,001	0,521	0,384	0,302
17	9-17-1	MSE Loss	0,001	0,45	0,001	0,445	0,275	0,262
18	9-18-1	MSE Loss	0,001	0,45	0,001	0,299	0,304	0,265
19	9-19-1	MSE Loss	0,001	0,45	0,001	0,473	0,314	0,324

The results of the NNs trained with the MSE Loss criterion reveal that when the number of epochs increases the loss of the NN decreases. Also, when the number of nodes in the hidden layer increases the loss decreases. However, this was found to hold only until an NN architecture of 9-17-1 was reached. Increasing the number of nodes in the hidden layer to more than 17 did not result in a better performing NN in terms of loss. Based on Table 10, can be concluded that an NN with an architecture of 9-17-1 was the best performing NN during training, with an average loss of 0.262.

The MSE Loss criterion penalises errors severely as it squares the error. Therefore, it was decided to apply a different loss criterion as well. It was chosen to use the SmoothL1Loss criterion (SL1 Loss), since this loss criterion is less sensitive to outliers in the data, which are present in the training data. The SL1 Loss uses a squared term if the absolute element-wise error falls below 1 and an L1 term otherwise, and is less sensitive to outliers than the MSE Loss (PyTorch, n.d.). The mathematical expression of the SL1 Loss is given by:

$$\mathrm{loss}(\mathrm{x},\mathrm{y}) = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}} \mathrm{Z_i}$$

Where Z_i is given by:

$$Zi = \begin{cases} 0.5(X_i - Y_i)^2, & if\ |X_i - Y_i| < 1 \\ |X_i - Y_i| - 0.5, & otherwise \end{cases}$$

X and *Y* are arbitrary shapes with a total of n elements each the sum operation still operates over all the elements, and divides by *N*.

The same 19 NN architectures are trained again, using the same settings as trained with the MSE loss criterion, however now the SL1 Loss criterion is used during training. The results of the 19 NNs trained with the SL1 Loss criterion can be found in Table 11.

Table 11: training results of NNs trained with SL1 Loss criterion

Model #	NN Arch	Loss criterion	LR	M	WD	Epoch = 50	Epoch = 100	Epoch = 150
						Avg. Loss	Avg. Loss	Avg. Loss
1	9-1-1	SL1 Loss	0,001	0,45	0,001	0,575	0,619	0,509
2	9-2-1	SL1 Loss	0,001	0,45	0,001	0,533	0,451	0,357
3	9-3-1	SL1 Loss	0,001	0,45	0,001	0,419	0,392	0,471
4	9-4-1	SL1 Loss	0,001	0,45	0,001	0,308	0,255	0,418
5	9-5-1	SL1 Loss	0,001	0,45	0,001	0,538	0,372	0,348
6	9-6-1	SL1 Loss	0,001	0,45	0,001	0,359	0,301	0,258
7	9-7-1	SL1 Loss	0,001	0,45	0,001	0,344	0,356	0,250
8	9-8-1	SL1 Loss	0,001	0,45	0,001	0,318	0,343	0,314
9	9-9-1	SL1 Loss	0,001	0,45	0,001	0,334	0,225	0,241
10	9-10-1	SL1 Loss	0,001	0,45	0,001	0,360	0,267	0,254
11	9-11-1	SL1 Loss	0,001	0,45	0,001	0,304	0,258	0,200
12	9-12-1	SL1 Loss	0,001	0,45	0,001	0,263	0,356	0,293
13	9-13-1	SL1 Loss	0,001	0,45	0,001	0,200	0,215	0,199
14	9-14-1	SL1 Loss	0,001	0,45	0,001	0,316	0,312	0,220
15	9-15-1	SL1 Loss	0,001	0,45	0,001	0,328	0,218	0,159
16	9-1-16	SL1 Loss	0,001	0,45	0,001	0,285	0,265	0,241
17	9-1-17	SL1 Loss	0,001	0,45	0,001	0,295	0,197	0,212
18	9-1-18	SL1 Loss	0,001	0,45	0,001	0,225	0,226	0,204
19	9-1-19	SL1 Loss	0,001	0,45	0,001	0,285	0,267	0,155

The results of the NNs trained with the SL1 Loss criterion again reveal that when the number of epochs increase and the number of nodes in the hidden layer the loss of the trained NN decreases. Compared to the results of the trained NNs with the MSE Loss criterion, the NNs trained with the SL1 Loss criterion perform better. They probably perform better because the SL1 Loss is less sensitive to outliers, and therefore the SL1 Loss is probably more suitable as loss function for the NN developed in this study. It can be concluded from Table 11 that an NN with an architecture of 9-19-1 was the best performing NN during training with an average loss of 0.155.

It was found that an NN with an architecture of 9-17-1 is the best performing NN trained with the MSE Loss criterion and an average MSE loss of 0.262. Figure 21 shows the graph in which the learning pattern of this NN is plotted. On the other hand, an NN with an architecture of 9-19-1 trained with the SL1 Loss criterion was found to be the best performing NN, with an average loss of 0.155. Figure 22 shows the graph in which the learning pattern of this NN is shown.

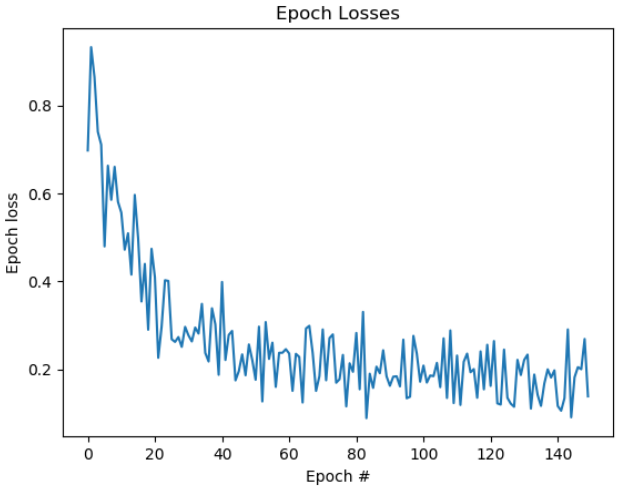


Figure 21: learning pattern of NN 9-17-1 MSE Loss

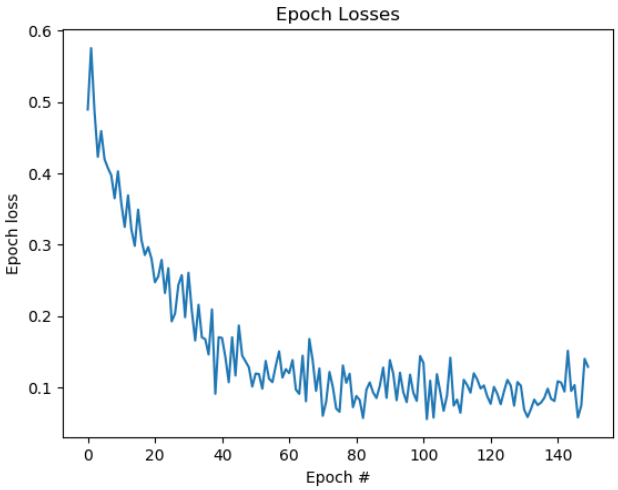


Figure 22: learning pattern of NN 9-19-1 SL1 Loss

Because it was found that the loss decreases when the number of epochs increases, it was decided to train the two best performing NNs over an even higher number of epochs. Arbitrarily, it was chosen to train these NNs over 175, 200, 225, 250, 500 and 1000 epochs to find out whether the loss of the NNs decreases even more. Results of these training runs can be found in Table 12.

Table 12: training results of best performing NNs on more epochs

Mod-el #	NN Arch	Loss criterion	Epoch = 175	Epoch = 200	Epoch = 225	Epoch = 250	Epoch = 500	Epoch = 1000
			Avg. Loss	Avg. Loss	Avg. Loss	Avg. Loss	Avg. Loss	Avg. Loss
17*	9-17-1	MSE Loss	0,250	0,262	0,261	0,225	0,209	0,165
19*	9-19-1	SL1 Loss	0,165	0,129	0,152	0,140	0,133	0,077

It can be concluded from Table 12 that the average loss of the NNs has decreased again when the number of epochs is increased. The best performing NN models are trained on 1000 epochs. They have an average loss of 0.165 for the NN model with an architecture 9-17-1 and trained with the MSE Loss criterion, and 0.077 for the NN model with architecture of 9-19-1 and trained with the SL1 Loss criterion. Below, in Figure 23 and 24, the graphs in which the learning pattern of these NNs is plotted are presented.

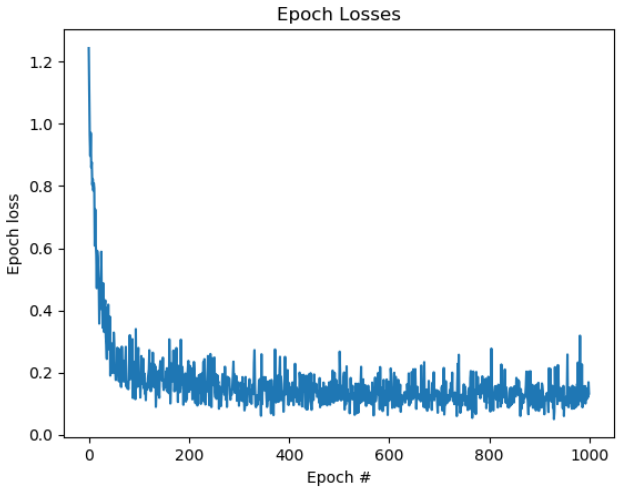


Figure 23: learning pattern of NN 9-17-1 MSE Loss after 1000 epochs

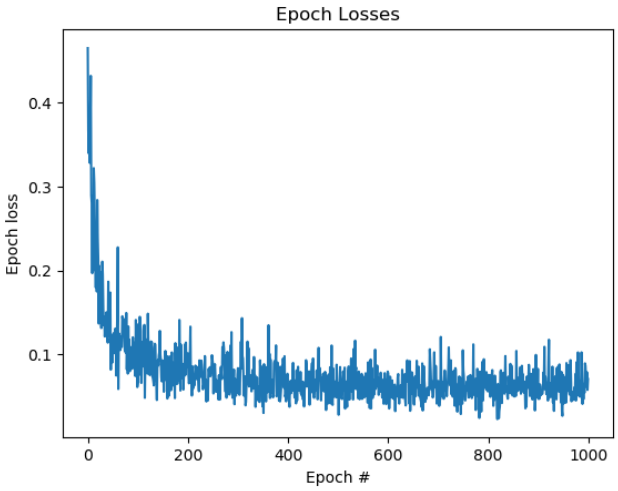


Figure 24: learning pattern of NN 9-19-1 SL1 Loss after 1000 epochs

5.4 Results of NN testing

The two best performing NNs will now be tested in order to evaluate their performance on unseen data. During NN testing, the developed and trained NNs will make predictions based on data on which the model has not been trained. After the predictions are made, the prediction performance of the NN can be evaluated. The predictions of the NN model will be assessed using performance indicators such as: MSE, MAE, MAPE and average accuracy (formulas provided in *4.4 Application phase*). With the help of these performance indicators, the best performing NN model will be selected. When one of the NNs reaches an average accuracy of ≥ 80% the NN model can be considered as accurate to be useful for conceptual construction cost estimating. However, keep in mind that the NN model is trained on simulated data and therefore testing results could differ if actual project cost data was used for NN training.

Test results NN 9-17-1 MSE Loss

Table 13 provides the test results of the NN with an architecture of 9-17-1, trained with the MSE loss criterion. This table indicates that this NN model is able to predict the construction costs despite that it was trained upon a little amount of training data. However, it must be concluded that the NN model mostly overestimates the construction costs when compared to the actual value of the construction costs. In the graph in Figure 25, the absolute percent error (APE) of the NN is plotted. In this graph an APE of 0% is seen as a 100% accurate prediction. The graph reveals that some NN predictions have an APE of ≥ 50%. Based on the graph and the table below, it can be concluded that the NN predicts worst on project data of which the actual constructions costs are ≤ 9 million and ≥ 25 million.

The performance of the model is evaluated by using the described performance indicators; this resulted in: an MAE of € 3.873.021, an MSE of 2,23E+13, an MAPE of 59%, and an average accuracy of 41%. Based on the performance indicators, it can be concluded that the model is not accurate enough to be useful for predicting conceptual construction costs.

Table 13: test results of NN 9-17-1 MSE Loss criterion

Actual	Predicted	Error	Squared Error	Absolute Percent Error
€ 8.985.000	€ 12.763.136	€ -3.778.136	1,43E+13	42,05
€ 4.077.001	€ 7.246.036	€ -3.169.035	1,00E+13	77,73
€ 2.929.001	€ 5.336.351	€ -2.407.350	5,80E+12	82,19
€ 23.432.580	€ 20.368.248	€ 3.064.332	9,39E+12	13,08
€ 17.956.183	€ 14.697.928	€ 3.258.255	1,06E+13	18,15
€ 4.676.895	€ 6.362.112	€ -1.685.217	2,84E+12	36,03
€ 20.781.766	€ 15.643.046	€ 5.138.720	2,64E+13	24,73
€ 2.517.868	€ 5.108.462	€ -2.590.593	6,71E+12	102,89
€ 5.746.624	€ 9.011.128	€ -3.264.504	1,07E+13	56,81
€ 4.381.000	€ 7.357.920	€ -2.976.920	8,86E+12	67,95
€ 37.710.000	€ 24.820.434	€ 12.889.566	1,66E+14	34,18
€ 23.670.773	€ 20.400.970	€ 3.269.803	1,07E+13	13,81
€ 15.928.802	€ 17.238.403	€ -1.309.601	1,72E+12	8,22
€ 31.389.253	€ 24.940.485	€ 6.448.768	4,16E+13	20,54
€ 991.332	€ 3.835.842	€ -2.844.510	8,09E+12	286,94
		MAE	**MSE**	**MAPE**
		€ 3.873.021	2,23E+13	59,02
				Average Accuracy
				40,98

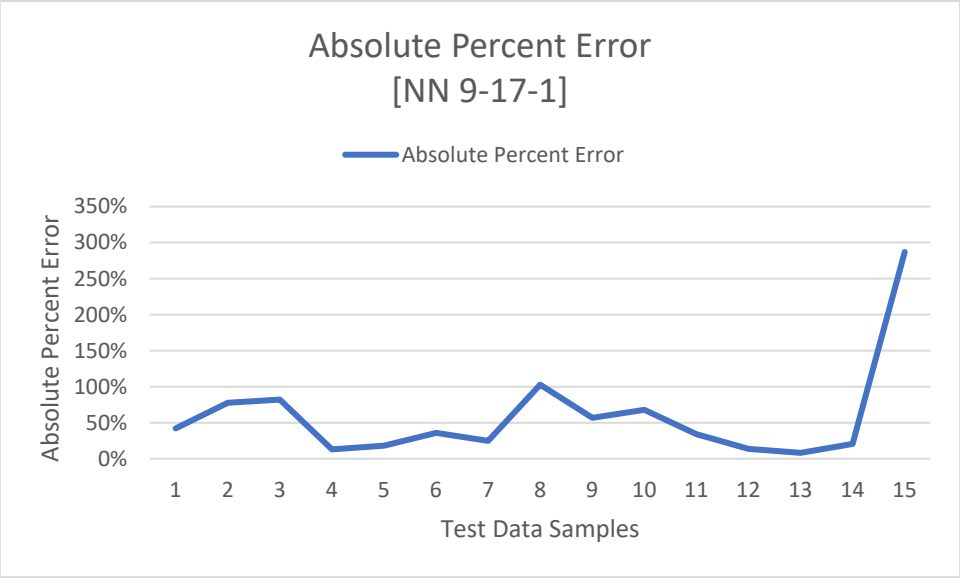


Figure 25: absolute percent error of NN 9-17-1 after testing

Test results NN 9-19-1 SL1 Loss

Table 14 provides the test results of the NN with an architecture of 9-19-1, trained with the SL1 loss criterion. It can be derived from the table that this NN model is able to predict the construction costs despite that it was trained on a little amount of training data. However, it must be concluded that the NN model mostly overestimates the construction costs when compared to the actual value of the construction costs. However, in comparison with the MSE NN model, the estimates and performance is better. Based on the graph in Figure 26 in which the APE is plotted and based on the table below, it can be concluded that the NN still predicts worse on project data of which the actual construction costs are ≤ 9 million and ≥ 25 million as the APE on this estimate is still ≥ 50%.

The performance of the model is evaluated by using the described performance indicators; this resulted in: an MAE of € 3.328.655, an MSE of 1,98E+13, an MAPE of 52%, and an average accuracy of 48%. Based on the performance indicators, it can be concluded that the model is not accurate enough to be useful for predicting conceptual construction costs.

Table 14: test results of NN 9-19-1 SL1 Loss criterion

Actual	Predicted	Error	Squared Error	Absolute Percent Error
€ 8.985.000	€ 11.388.300	€ -2.403.300	5,78E+12	26,75
€ 4.077.001	€ 6.723.341	€ -2.646.340	7,00E+12	64,91
€ 2.929.001	€ 4.961.852	€ -2.032.852	4,13E+12	69,40
€ 23.432.580	€ 21.770.981	€ 1.661.599	2,76E+12	7,09
€ 17.956.183	€ 15.726.982	€ 2.229.201	4,97E+12	12,41
€ 4.676.895	€ 6.619.075	€ -1.942.180	3,77E+12	41,53
€ 20.781.766	€ 16.283.311	€ 4.498.455	2,02E+13	21,65
€ 2.517.868	€ 5.012.325	€ -2.494.457	6,22E+12	99,07
€ 5.746.624	€ 8.109.284	€ -2.362.660	5,58E+12	41,11
€ 4.381.000	€ 6.633.533	€ -2.252.533	5,07E+12	51,42
€ 37.710.000	€ 24.458.910	€ 13.251.090	1,76E+14	35,14
€ 23.670.773	€ 21.685.607	€ 1.985.166	3,94E+12	8,39

€ 15.928.802	€ 16.690.679	€ -761.877	5,80E+11	4,78
€ 31.389.253	€ 24.767.091	€ 6.622.162	4,39E+13	21,10
€ 991.332	€ 3.777.280	€ -2.785.949	7,76E+12	281,03
		MAE	**MSE**	**MAPE**
		€ 3.328.655	1,98E+13	52,39
				Average Accuracy
				47,61

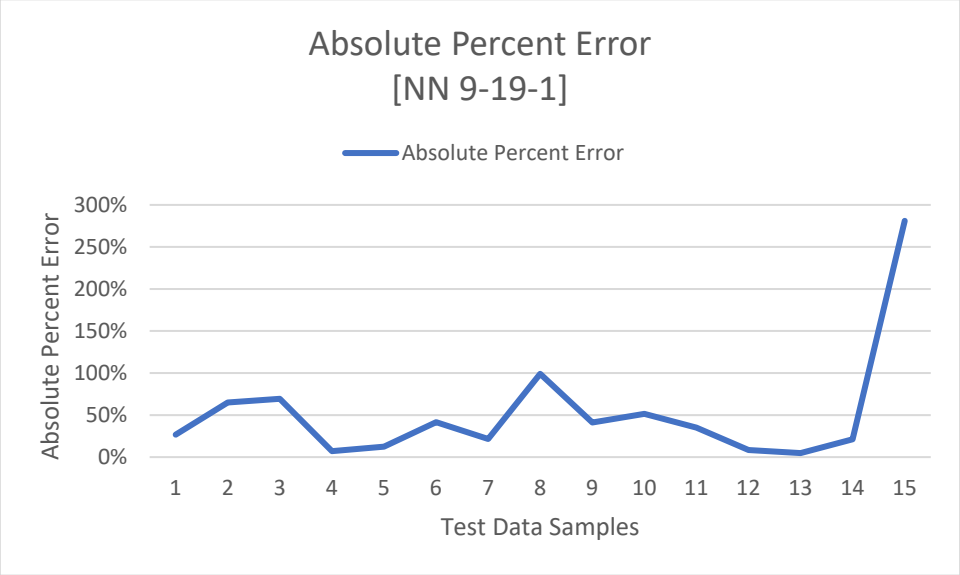


Figure 26: absolute percent error of NN 9-19-1 after testing

Improving estimation performance

It can be concluded from the first tests with the NNs that the desired average accuracy level of ≥ 80% is not reached. By using a loss criterion that is less sensitive to outliers, the estimation performance of the NN model slightly increased with 7% from 41% to 48%.

The best performing NN achieved an average accuracy of 48%. It was found that this NN on certain data predicts with an absolute percent error of ≥ 50%. This is probably caused by the outliers present in the training data. Therefore, it is decided to remove the outliers from the training data and to retrain the best performing NN architecture (i.e. NN 9-19-1 SL1 Loss criterion) again upon an optimized training dataset in which outliers are excluded. Subsequently, the performance of this NN will be evaluated by testing it again upon the test dataset, as well as testing it upon a test dataset in which only one type of project is considered.

Based on the boxplot provided in Figure 15 in chapter *4.2.2 Data gathering and representing*, it can be concluded that data samples in the training dataset with a construction cost of ≤ 1 million and ≥ 45 million belong to the outliers in the training data. Therefore, these data samples were removed from the training dataset. This resulted in a training dataset comprising 26 data samples, of which 14 data samples belong to type of project category 1 (i.e. residential housing projects) and 12 data samples belong to type of project category 2 (i.e. multi-storey housing projects). With this optimised dataset, the NN with an architecture of 9-19-1 and the SL1 Loss criterion will be trained again. In Figure 27, a graph is presented in which the learning pattern of the NN on this optimised data is plotted. After training over 1000 epochs, this NN achieved an average loss of 0.10. In comparison with the same NN that was trained upon 35 data samples, including outliers cases, the average loss is now (i.e. when outliers cases are removed) slightly higher. This is probably caused by the fact that only 26 data

samples could have been used for NN training, which is a very limited amount of data samples for NN training.

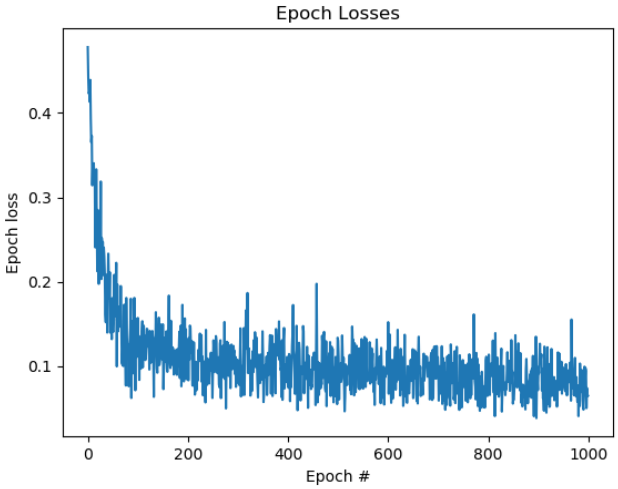


Figure 27: learning pattern of NN 9-19-1 SL1 Loss criterion on optimised data

To see whether the estimation performance of the NN has improved now that it has been trained with a training dataset in which outliers are excluded, it will be tested again to evaluate its estimation performance again. First, the estimation performance of the NN will be tested based on the 15 testing data samples. Secondly, the testing dataset will be split into two categories, namely the type of project 1 (i.e. residential housing projects) and the type of project 2 (i.e. multi-storey housing projects), in order to evaluate with which category of projects the NN performs best in terms of estimation.

Table 15 provides the test results of the NN 9-19-1 SL1 Loss criterion that was trained on 26 data samples in which outlier cases were excluded. It can be derived from the table that the NN is still able to predict the construction cost despite that the model was trained on only 26 training data samples. The table indicates that the NN still overestimates the construction costs, however far less than before. However, based on the graph in Figure 28 in which the APE is plotted and the table below, it can be concluded that the NN still predicts worse on project data of which the actual construction costs are ≤ 9 million, as the APE of these estimates falls above 35%. In comparison with the NN trained on 35 data samples including outlier cases, this NN performed way better. The MAE, MSE, and MAPE of the NN decreased to values of: € 2.407.363, 8,11E+12, and 34% respectively. On the other hand, the average accuracy increased to 66% where it was 48% before.

Based on the performance indicators (MAE, MSE, MAPE, and average accuracy), it can be concluded that although the average accuracy increased to a level of 66%, the model is still not considered accurate enough to be useful for predicting conceptual construction costs.

Table 15: test results of optimised NN 9-19-1 SL1 Loss criterion

Actual	Predicted	Error	Squared Error	Absolute Percent Error
€ 8.985.000	€ 12.622.712	€ -3.637.712	1,32E+13	40,49
€ 4.077.001	€ 5.908.329	€ -1.831.328	3,35E+12	44,92
€ 2.929.001	€ 4.069.378	€ -1.140.377	1,30E+12	38,93
€ 23.432.580	€ 20.431.696	€ 3.000.884	9,01E+12	12,81
€ 17.956.183	€ 16.827.318	€ 1.128.865	1,27E+12	6,29
€ 4.676.895	€ 6.613.912	€ -1.937.016	3,75E+12	41,42
€ 20.781.766	€ 18.047.276	€ 2.734.490	7,48E+12	13,16

€ 2.517.868	€ 3.709.031	€ -1.191.163	1,42E+12	47,31
€ 5.746.624	€ 7.489.122	€ -1.742.498	3,04E+12	30,32
€ 4.381.000	€ 5.695.069	€ -1.314.070	1,73E+12	29,99
€ 37.710.000	€ 31.432.836	€ 6.277.165	3,94E+13	16,65
€ 23.670.773	€ 22.045.141	€ 1.625.632	2,64E+12	6,87
€ 15.928.802	€ 17.497.508	€ -1.568.706	2,46E+12	9,85
€ 31.389.253	€ 25.999.143	€ 5.390.110	2,91E+13	17,17
€ 991.332	€ 2.581.755	€ -1.590.423	2,53E+12	160,43
		MAE	**MSE**	**MAPE**
		€ 2.407.363	8,11E+12	34,44
				Average Accuracy
				65,56

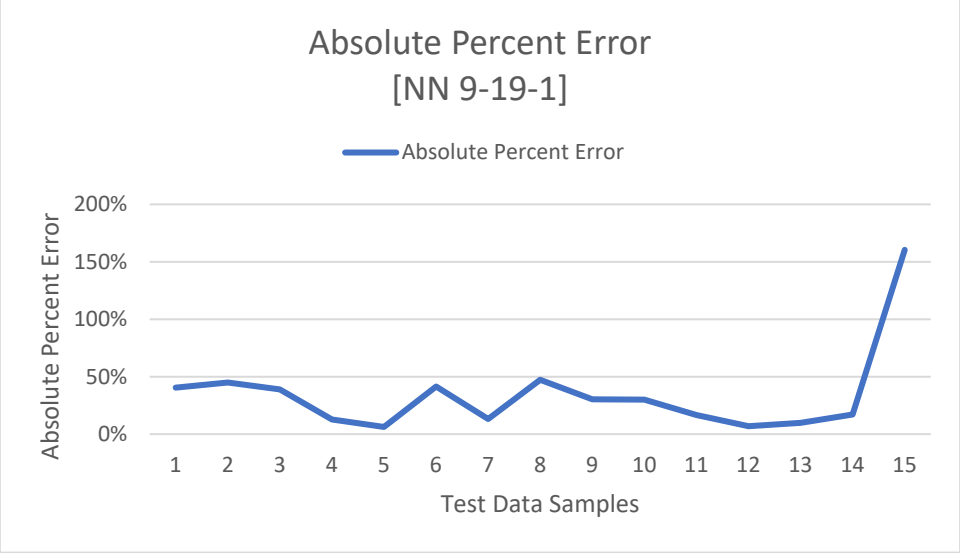


Figure 28: absolute percent error of optimised NN 9-19-1 after testing

Estimating with one type of project category

To see whether the estimation performance of the NN increases as only one category of the type of projects is used during testing, it was decided to evaluate it based on the test data of only one type of project. First, the NN model will be tested on the 5 testing data samples that belong to the type of project category 1 (i.e. residential housing projects). Secondly, it will be tested upon the 10 remaining data samples that belong to the type of project category 2 (i.e. multi-storey housing projects).

Table 16 provides the test results for when the NN is tested on the 5 data samples that belong to residential housing projects (i.e. ToP = 1). It can be concluded from the table that the NN model overestimates the construction costs for residential housing projects with an MAE of € 2.018.863. Based on the graph in Figure 29, can be concluded that the absolute percentage error of all estimates falls below 50%, resulting in an MAPE of 31%. Also, the MSE (4,67E+12) decreased. On the other hand, the average accuracy (69%) of the NN model increased. Nevertheless, the NN model cannot be considered accurate enough to be useful for conceptual cost estimations, as these estimations require an average accuracy of ≥ 80%.

Table 16: test results of optimised NN 9-19-1 SL1 Loss criterion on TOP = 1 projects

Actual	Predicted	Error	Squared Error	Absolute Percent Error
€ 8.985.000	€ 12.622.712	€ -3.637.712	1,32E+13	40,49
€ 4.077.001	€ 5.908.329	€ -1.831.328	3,35E+12	44,92
€ 5.746.624	€ 7.489.122	€ -1.742.498	3,04E+12	30,32
€ 4.381.000	€ 5.695.069	€ -1.314.070	1,73E+12	29,99
€ 15.928.802	€ 17.497.508	€ -1.568.706	2,46E+12	9,85
		MAE	**MSE**	**MAPE**
		€ 2.018.863	4,76E+12	31,11
				Average Accuracy
				68,89

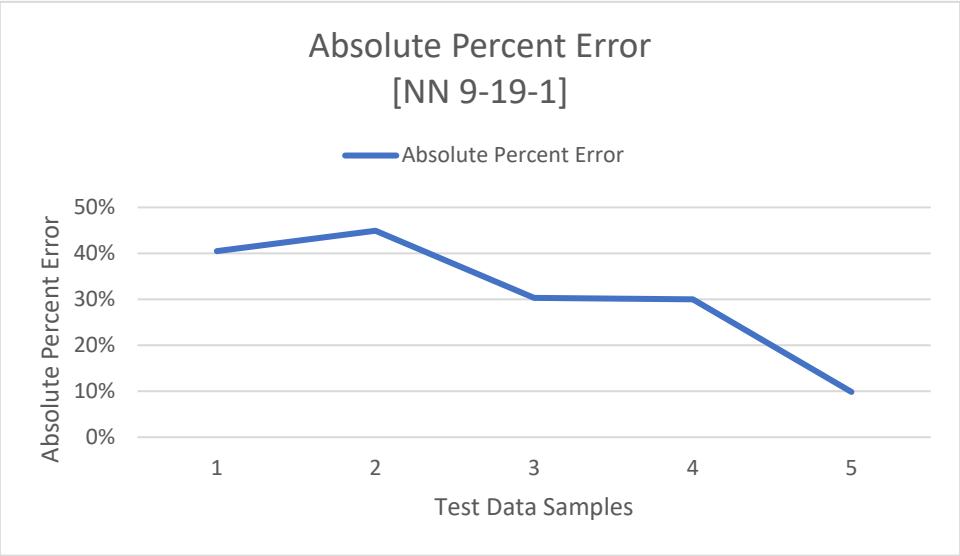


Figure 29: absolute percent error of optimised NN 9-19-1 after testing on TOP = 1 projects

Table 17 provides the test results for when the NN is tested on the 10 data samples that belong to multi-storey housing projects (i.e. ToP = 2). It can be concluded from the table that in most cases the NN model underestimates the construction costs for multi-storey housing projects with an MAE of € 2.601.613. Based on the graph in Figure 30, can be concluded that the absolute percentage error of almost all estimates falls below 50%, resulting in an MAPE of 36%. The MSE (9,79E+12) also increased. The average accuracy (64%) of the NN model, however, decreased in comparison with the estimation results achieved on the entire test dataset and the test data associated with residential housing projects. Therefore, the NN model cannot be considered accurate enough to be useful for conceptual cost estimations.

Table 17: test results of optimised NN 9-19-1 SL1 Loss criterion on TOP = 2 projects

Actual	Predicted	Error	Squared Error	Absolute Percent Error
€ 2.929.001	€ 4.069.378	€ -1.140.377	1,30E+12	38,93
€ 23.432.580	€ 20.431.696	€ 3.000.884	9,01E+12	12,81
€ 17.956.183	€ 16.827.318	€ 1.128.865	1,27E+12	6,29
€ 4.676.895	€ 6.613.912	€ -1.937.016	3,75E+12	41,42
€ 20.781.766	€ 18.047.276	€ 2.734.490	7,48E+12	13,16
€ 2.517.868	€ 3.709.031	€ -1.191.163	1,42E+12	47,31
€ 37.710.000	€ 31.432.836	€ 6.277.165	3,94E+13	16,65
€ 23.670.773	€ 22.045.141	€ 1.625.632	2,64E+12	6,87
€ 31.389.253	€ 25.999.143	€ 5.390.110	2,91E+13	17,17
€ 991.332	€ 2.581.755	€ -1.590.423	2,53E+12	160,43
		MAE	**MSE**	**MAPE**
		€ 2.601.613	9,79E+12	36,10
				Average Accuracy
				63,90

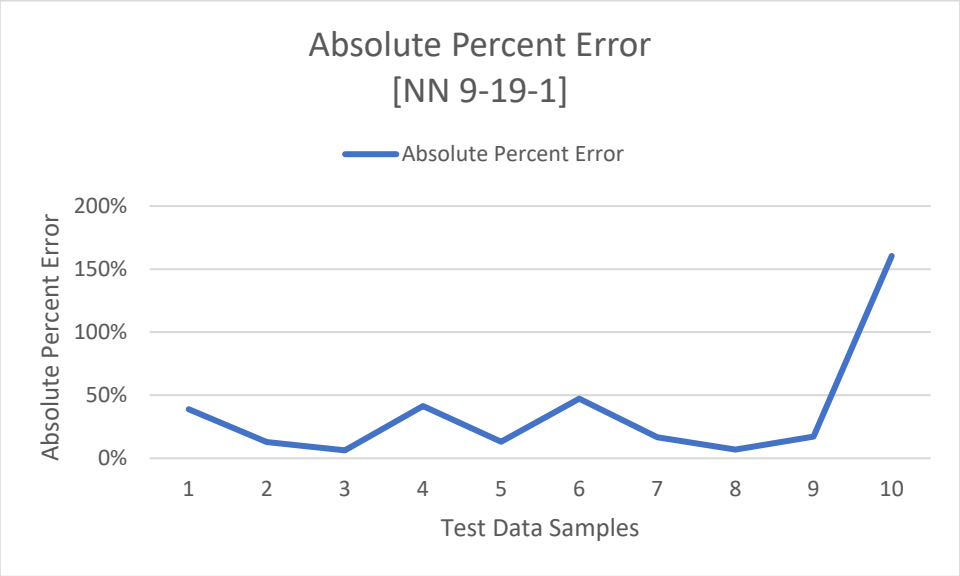


Figure 30: absolute percent error of optimised NN 9-19-1 after testing on TOP = 2 projects

In the test dataset of multi-storey housing projects, one data sample (i.e. data sample with an actual construction cost of € 999.332) is clearly visible that can be considered as an outlier data sample. If this data sample would not have been included in the test dataset, the estimation performance of the NN model on the test dataset with the 10 data samples of ToP = 2 would have been different. This would have resulted in an MAE of € 2.713.967, an MSE of 1,06E+13, an MAPE of 22%, and an average accuracy of 78%. Still, the desired level of average accuracy of ≥ 80% would not have been reached. However, this would then have been the best performance of the developed NN model that approaches the desired average accuracy level of ≥ 80%. Therefore, it can be stated that the developed NN model would be able to estimate construction costs with an average accuracy of 78%, if the expected actual construction cost falls within the range of ≥ 1 million and ≤ 38 million. However, note that this accuracy level was achieved with the help of simulated data that was used for NN training. The results would probably be different, as actual project cost data was used for NN training.

5.5 Comparison of results

In this paragraph, the results of the best performing NN will be compared to results of similar studies found in literature. The results of the NN developed for this study will be compared with results of similar studies that also used ± 50 data samples for NN training and testing. Based on the literature review, the results of this study can be compared with studies performed by Elhag & Boussabaine (1998), Günaydin & Doğan (2004), Sonmez (2004), Luu & Kim (2009), ElSawy et al. (2011), and Lyne & Maximinio (2014). The results after performance evaluation during testing of the NNs in these studies are summarised in Table 18.

Table 18: summarisation of results from comparable ANN related studies

Study	Data availability	Results
Elhag & Boussabaine (1998)	30 projects	MSE 20,74% ; 17,77% Prediction accuracy 79,3% ; 82,2%
Günaydin & Doğan (2004)	30 projects	MSE 0,038 Prediction accuracy 93%
Sonmez (2004)	30 projects	MSE 3,6E+12 and 3,8E+12 MAPE 12,3 and 11,7
Luu & Kim (2009)	14 projects	MPE -1,5% MAPE 8,5%
ElSawy et al. (2011)	52 projects	Prediction accuracy 80%
Lyne & Maximinio (2014)	30 projects	MSE 2,98E+15

After evaluating the performance during testing, the best performing NN developed in this study had an MSE of 4,76E+12, an MAPE of 31%, and an average accuracy of 69%. Based on these results, it can be concluded that the NN achieves almost similar results as the results in similar studies. In terms of MSE, the NN developed in this study performed better than the NNs developed in Lyne & Maximinio (2014), while it performed slightly worse in comparison to the NN developed in Sonmez (2004). However, in terms of the MAPE and the average prediction accuracy, the developed NN performed less than the NNs developed in the studies of Günaydin & Doğan (2004), Sonmez (2004), and Luu & Kim (2009).

However, as the NN model is tested on the test data samples that belong to multi-storey housing projects in which the outlier cases were excluded, it would have achieved the following results: an MAE of € 2.713.967, an MSE of 1,06E+13, an MAPE of 22%, and an average accuracy of 78%. These results are better in terms of MSE than the studies of Sonmez (2004) and Lyne & Maximinio (2014). In terms of average prediction accuracy, the results are comparable to the studies of Elhag & Boussabaine (1998) and ElSawy et al. (2011).

Based on the comparison of the results, it can be concluded that the results of this study show resemblances to the results of similar studies found in the literature. These findings support the significance and relevance of this study, and support the results of the NN model developed in this study. However, one should keep in mind that for this study, simulated data with limited data availability has been used for NN training. Despite the fact that the simulated data shows similarities with the actual project cost data, it still remains data that probably does not represent the reality. Therefore, the achieved results of the developed NN in this study should be interpreted carefully. The results of this study prove that an NN can be used for conceptual cost estimations with desirable results. However, if the actual project cost data was used for NN training, these results could have been different.

5.6 Conclusion

In this conclusion, the following research question will be answered: "*how can ANN-based cost estimation be used for construction cost estimating if limited data is available?*"

An NN can be used for (conceptual) construction cost estimating when limited data is available. However, in this study it was found that the NN that was developed according to the methodology described in chapter *4. Methodology* did not yet achieve the desired accuracy level of ≥ 80%, since the best performing NN in this study obtained an average estimation accuracy of 69%. If this NN will be improved by using more qualitatively real project cost data, it can be used for construction cost estimating. It will then probably achieve an average estimation accuracy of ≥ 80% or higher, as was found during the literature study in chapter *3. Artificial Neural Networks*.

In this study, the NN has been developed with the PyTorch ML library and was trained with 35 simulated cases. During training, 19 different NN architectures were considered and two different loss criterions were applied to the NN. After training it was found that an NN with an architecture of 9-17-1 trained with the MSE Loss criterion and an NN with an architecture of 9-19-1 trained with the SL1 Loss criterion performed best when trained over 1000 epochs, with an average epoch loss of 0.165 (NN 9-17-1 MSE) and 0.077 (NN 9-19-1 SL1 Loss) respectively. During testing, both best performing NNs during training were tested on 15 new data samples in order to evaluate their prediction performance in terms of MSE, MAE, MAPE, and average accuracy. By the end of testing, it was found that the NN with an architecture of 9-19-1 trained with the SL1 Loss criterion performed best, with an average accuracy of 48%.

By retraining the best performing NN (i.e. NN 9-19-1 SL1 Loss) on 26 data samples in which the outlier cases were removed and only testing it on data samples of type of project category 1 (i.e. residential housing projects), the performance of the NN improved to an average estimation accuracy of 69%. This average estimation accuracy level, of the best performing NN in this study, is not high enough to consider the NN model accurate and useful for conceptual cost estimations, as the average estimation accuracy is ≤ 80%.

Note that for this study, simulated data has been used for NN training and that due to the use of the simulated data, the results could differ as the simulated data mostly does not represent the reality. Therefore, the results of this study should be interpreted carefully because if real project cost data had been used for NN training, the results of the study would probably have been different. Therefore, it is recommended to first validate the results of this study by applying real project cost data to NN training before the results of this study can lead to practically applying NNs for conceptual construction cost estimating. In the next chapter, a critical discussion on this study will be provided, along with a discussion of the practical application of NNs and lessons learned from this study.

6. Discussion

In this chapter, a critical discussion on the study will be provided. First, the results of the study will be discussed. Subsequently, the practical application of the developed NN will be discussed and finally, the limitations belonging to this research will be discussed.

6.1 Discussion of results

In this study, the objective was to find out whether NNs could be used for conceptual cost estimations when limited (training) data is available. During the literature study, it was already found that NNs trained with limited amounts of data can be applied for construction cost estimations and can reach a prediction accuracy of ≥ 90%, as was found in the study of Günaydin & Doğan (2004). In this study, if the developed NN would achieve an average prediction accuracy of ≥ 80%, it would be considered accurate and useful for conceptual construction cost estimating. According to the described methodology in chapter *4. Methodology*, an NN was developed with an architecture of 9-19-1 where 9 is the amount of input nodes, 19 the amount of nodes in the hidden layer, and 1 is the node in the output layer respectively. This developed NN was trained over 1000 epochs on 26 simulated training data samples and tested on 15 actual project data samples. After testing, it was concluded that this best performing NN was able to estimate the (conceptual) construction cost based on 9 input variables: type of project, gross floor area, usable floor area, gross building volume, number of storeys, roof area, façade area, number of units, and building height. These were estimated with an average accuracy level of 69%. The desired accuracy level was thus not reached in this study. Therefore, the developed NN can be considered not accurate enough and thus not useful for conceptual cost estimating. However, note that for this study simulated training data has been used, which has probably impacted the results. If for example the actual project cost data had been used for NN training, the results of the study would probably have been different.

Normally, ML models rely upon the data of thousands of cases. However, the data availability for this study was limited. Therefore, a limited amount of data (i.e. 50 data samples in total) has been used in this study in order to establish the NN (ML) model. Since an NN is comparable to linear regression, the data condition rule of linear regression would also be generalisable to NNs. This rule is as follows: sample results are generalisable to populations if # observations / # predictors = 15:1. This means that for this study 9*15 = 135 data samples should have been used to establish the NN (ML) model. Now, only 50 data samples have been used and the NN model could therefore probably be overfit on the data. This results in a worse fit on the data and therefore a lower average prediction accuracy. It would thus be a good starting point for later studies to use the data condition rule of linear regression for NNs as well.

Section *5.5 Comparison of results* mentions that the results of this study show similarities to the results of previous NN related studies in terms of performance indicators (i.e. MSE, MAPE, prediction accuracy etc.). But, most of the developed NNs in other studies do perform better than the NN of this study. Why these NNs perform better than the presented NN is unclear. The studies used similar (input) variables as the presented study. However, how the data used in these studies looks like related to the variables used in the studies in terms of data conditions as; range covered, correlations etc. is not made clear in those studies. These studies may have obtained good NN results with good qualitative datasets. Instead of using noisy datasets from construction practice as done in this study, the reviewed studies could maybe have used only datasets that were selected beforehand. This way of data selecting could have affected the results of these studies, as in construction practice only good qualitative datasets are not always available for cost estimating. Therefore, one should be careful with interpreting the results of these studies as they may not represent the real practice of construction cost estimating.

6.2 Discussion practical implementation

Nearly none of the studies regarding the use of NNs for construction cost estimating result in practical implementation. Not because of the results achieved with the NNs in the studies, but most often due to a lack of capacity in data. Another possible concern, is probably that an NN is seen as a black box technique and before they can be used by cost planners, they must have at least some prior knowledge of NNs. Therefore, they are probably not applied in the construction sector yet as a practical tool for construction cost estimating. In the literature, it was found that NNs could specifically be used for conceptual construction cost estimating, which is done according to the analogous estimating technique. The reasoning of NNs is similar to this estimation method, as NNs use data to learn from. This makes the NN a suitable tool for conceptual construction cost estimating, especially when the NN can achieve an average prediction accuracy level of ≥ 80%. Unfortunately, the NN developed in this study did not reached this desired accuracy level. It reached an average accuracy of 69% and can thus not yet be used for conceptual construction cost estimating.

Despite that the desired accuracy level was not reached during the study, the study still contributes to the practical adoption and application of AI and ML in construction. Results of the study showed that NNs can also be used when there is only a limited amount of qualitative data available for NN development. Besides, the study also showed that NNs can be applied for a lot of construction related activities. These findings prove that NNs can be seen as useful technique for applying AI in construction cost estimating. However, the fact that NNs are considered to be a black-box technique should be kept in mind. But, still this study has shown that when NNs are developed with the right amount of qualitative project cost data, they can reach accuracy levels that are desirable for conceptual construction cost estimating. Therefore, the NN seems to be a promising technique for construction cost estimation in the future.

If the NN could be applied for conceptual construction cost estimating by the construction firm, the firm would be able to make fast and accurate conceptual cost estimations within in a limited time frame and with little effort. To make these estimations, the data belonging to the estimation variables considered by the model need to be entered into the model and subsequently, the cost estimate is made by running the model. However, applying the developed NN for cost estimations requires the cost planner to have prior knowledge of NNs and ML. Also, to apply the NN, the cost planner needs to have at least some coding skills. Therefore, in the next chapter *(7.2 Practical recommendations)*, practical recommendations will be provided regarding the implementation of NNs within the construction firm. Also, a recommendation regarding the further development of the NN into a powerful, state-of-the-art cost estimation will be provided in the upcoming chapter. Results of the study were also presented to the management and cost planners of the graduation company involved during the study. They received the results enthusiastically, they also see added value in the use of NNs for conceptual cost estimates, especially if the NN can be used to validate whether the calculated costs of a project are in line with the costs of past similar projects.

When AI in construction becomes more mature in the construction sector, and when NNs are studied and utilised more and more they could maybe be linked with BIM data repositories (i.e. BIM models) or data repositories of AEC firms. Linking NNs and BIM models is interesting as BIM models nowadays begin to serve more and more as data repositories in the AEC-industry. A lot of data is stored in BIM models, not only geometric data but also for example cost data. However, the data in BIM models is either available in proprietary data formats or the difficult to use neutral data format IFC. Therefore, coding or software is needed to access the stored data. Once the data is accessible, however, the data can probably be used for a lot of tasks for which NNs can be constructed as well. This makes BIM a very interesting data source, especially when it is also linked with data repositories of AEC firms. In this way, more and more data can be gathered and probably be used for the development of useful NNs in the AEC industry.

6.3 Research limitations

During this study, an attempt was made in developing an NN that can be used for conceptual construction cost estimating. The results prove that the developed NN cannot be used for conceptual construction cost estimating yet. As with every study, this study has its limitations. In this paragraph, these limitations will be discussed. During the study the following limitations were encountered:

- The NN methodology is a methodology that relies more on trial and error than on using and following a sound scientifically proven method. Therefore, in determining the architecture of an NN, the designer has a lot of freedom in making decisions about a lot of design parameters (i.e. number of input nodes, number of hidden layers, number of nodes in the hidden layer, activation functions, learning algorithms etc.). This flexibility implies an important problem: there is no defined methodology for defining the architecture of an NN. Thus, the NN for this study is empirically, rather than theoretically, derived. If a theoretically proven method could have been applied, the architecture of the NN as well as the results would probably have been different. In this study, a beginning was made in establishing the required theoretical methodology for developing NNs. However, this method still relies upon assumptions made in accordance with the literature and trial and error. Therefore, the method described in this study should be validated before it is proven to be correct and useful for NN development.
- Predictive data mining models (e.g. NNs) are heavily dependent on the size and quality of the data. Generally, using a high number of high-quality data samples results in a more accurate prediction model. In this research, a limited amount of data (i.e. 15 data samples) were gathered from a construction firm in the Netherlands. To be able to use more data for NN development, specifically NN training, a data simulation tool was developed in Python. With the help of this tool, 35 new data samples were simulated based on the existing data. These simulated data samples have then been used for NN training. Due to the usage of simulated data instead of actual data, the results of this study can be different as the simulated data probably does not represent reality. Therefore, one should be careful with interpreting the results of this study. Further research is therefore recommended in order to first validate the results of this study with actual project cost data.
- In total, 50 data samples have been used for the development of the NN in this study. This is a relatively small number of cases, as ML models normally rely on thousands of cases. Therefore, if more real project cost data samples had been gathered and used, the results of the study would probably have been different. A good indication of the required data would be using the data condition rule of linear regression. This would imply that for every input variable of the NN, at least 15 data samples should be gathered. This is because the rule is that sample results are generalisable to populations when # observations / # predictors = 15:1.
- During this study, data of two project typologies were used, namely the data of residential housing projects and multi-storey housing projects. It was assumed that the NN model would be able to generalise and thus, different project typologies could have been used. Due to time constraints and data limitations, however, no NN model was made for a singular project typology. If only one project typology had been chosen for the NN model, the results could have been different.
- The NN has one big limitation, which is that it is considered to be a black box technique, as they do not provide any explanation about their reasoning in making predictions. Therefore, learning from an NN is difficult and validating whether the NN applied the right reasoning for predicting is near impossible due to the lack of explanation. Due to these limitations, the objective of an NN must never be to learn from it. Still, when the NN has been used for predictions, this needs to be done carefully, since it can also use the wrong assumptions. Therefore, it is probably better to utilise an NN as a prediction evaluation tool instead of utilising it as a standalone estimation tool.

7. Conclusion

In this chapter, a conclusion will be drawn for the study by answering the research questions that were formulated at the beginning of this thesis. When the research questions have been answered, recommendations for practical application and further research will be provided.

7.1 Conclusion

This paragraph will answer the research questions that belong to this thesis in order to formulate a final conclusion for the study. Based on the problem definitions and research objectives mentioned in chapter 1. Introduction the research questions were formulated.

First, the conclusions to the sub research questions will be provided and subsequently, the conclusion to the main research question will be provided.

7.1.1 Conclusion to sub research questions

In order to be able to answer the MRQ of this research project, four SRQs were formulated. SRQs 1 and 2 belong to problem definition 1, and SRQs 3 and 4 belong to problem definition 2. The problem definitions as well conclusions to these four SRQs will be provided below.

Problem definition 1:

"Can cost estimations be made more accurate by using ANNs trained with limited project cost data?"

SRQ 1: How can the cost estimation process for construction projects be characterised?

Cost estimation can be defined as: the process of forecasting the cost of a project based on a documented scope and known facts, by means of the summation of individual cost elements by using established methods and valid data. Making various cost estimations that become more accurate as the construction process evolves, characterises the construction cost estimation process. Based on their level of detail and accuracy, cost estimations can be classified in the Netherlands according to the NEN2699:2017, where level 1 is the most global level and level 6 the most detailed level of cost estimating. In this study, the focus is specifically on conceptual cost estimations, since they are made during the early phases of the construction process. These conceptual cost estimations are often based on the top-down approach and the analogous estimating technique. The expected accuracy level of a conceptual cost estimate is within -15% to +25% of the final project cost.

SRQ 2: How can ANNs be utilised for estimation tasks in the construction sector?

To utilise NNs for estimation tasks, a prediction data mining technique and supervised ML approach need to be used. With the aid of supervised learning, an NN is able to generalise knowledge and learn from examples. Decisions about the NN architecture, activation functions, and loss functions need to be made before an NN can be utilised for estimations. In most estimation tasks performed with NNs, a single-layer feedforward backpropagation (FFBP) NN was used. The architecture of these NNs consist of an input layer, a hidden layer, and an output layer. In this study, a single-layer FFBP was used as well.

Problem definition 2:

"How can ANNs trained with limited project cost data be used for cost estimations during the early phases of the construction process, and how accurate are these ANN-based cost estimations?"

SRQ 3: How can (conceptual) cost estimations be performed with ANNs?

To perform conceptual cost estimations with an ANN, three phases and six steps need to be completed:

Phase I: Modeling phase

1. Problem definition
2. Data gathering and representing

3. Defining the network

Phase II: Implementation phase

4. Structuring the network
5. Training and testing the network

Phase III: Application phase

6. Applying the network for estimations

By following these phases and steps, an FFBP NN was created. If the developed NN reaches an average accuracy of ≥ 80%, the NN is considered accurate enough to be used for conceptual cost estimations.

SRQ 4: How can ANN-based cost estimation be used for construction cost estimating if limited data is available?

NN-based cost estimation can be used for conceptual construction cost estimation. However, the NN developed in this study cannot be used for this objective yet, as this NN did not achieve the desired average estimation accuracy level yet. In this study, the NN was constructed based on the process steps mentioned above and described in the methodology. The NN was coded with the PyTorch ML library in Python and subsequently trained on 35 simulated cases. The NN was tested on 15 new actual cases in order to evaluate its prediction performance. By the end of NN testing, it was found that the NN achieved an average accuracy of 69%. Therefore, it must be concluded that the developed NN cannot be used for conceptual construction cost estimating yet, as the desired accuracy level of ≥ 80% has not been achieved.

7.1.2 Conclusion to main research questions

In this paragraph, a conclusion will be drawn in order to answer the MRQ of this study. The MRQ for this study was formulated as follows:

"How can cost estimations with ANNs, trained with limited project cost data, create more accurate cost estimations for construction projects during the early phases of the construction process?"

The objective of this research was to study whether NN-based cost estimations would improve the accuracy of conceptual cost estimations. In order to complete this objective, a state-of-the-art technology was studied and applied for conceptual construction cost estimations. Conceptual cost estimations are often made according to the top-down approach and the analogous estimating technique that is known to be a comparative estimating method, which is where the cost estimate is developed by reviewing cost from previous similar projects and considering the differences with similar projects. In chapter *3. Artificial Neural Networks* is described that NNs due to their supervised learning approach are able to generalise knowledge and are able to learn from examples. This is useful for conceptual cost estimations as they are often based on previous cases, and by considering the differences between cases. Since these estimates are made according to the analogous estimating technique by following a top-down approach. It can be derived from chapter *4. Methodology* that when the NN reaches an accuracy level of ≥ 80% (i.e. a prediction error of ± 20%), it is considered to be accurate enough for conceptual cost estimating. Based on the described methodology, an NN was developed. After training the NN with 26 simulated cases and evaluating its prediction performance on 5 new actual project cases of residential housing projects, on which the NN had not been trained yet, it was found that an NN trained with an architecture of 9-19-1, trained with the SL1 Loss criterion over 1000 epochs, achieved an average prediction accuracy of 69%.

It can therefore be concluded from the results of the study that the developed NN did not achieve the desired accuracy level of ≥ 80%. Note that for this study, the NN was developed with a limited availability of data. Also, the data has been simulated for NN training in order to enable more use to the data for the study. If the same NN was trained with a higher amount of actual, high-quality project

cost data, the average accuracy of the developed NN would probably increase, as well as the accuracy of the conceptual cost estimations made with the NN estimation model.

Now that we know the results of the study, the hypothesis formulated at the start of this thesis can be answered. Based on the theoretical and practical target, the following hypothesis for this study was drafted:

"If ANNs trained with a limited amount of project cost data can be used for cost estimations of new projects made in the early phases of the construction process, the cost estimations of construction projects can be made more accurate."

This hypothesis has not been proven, based on the following arguments:

- The NN that was developed, trained, and tested upon limited data (i.e. 26 data samples for training, and 15 data samples for testing) in this study reached an accuracy of 69% which is 11% points less than the desired accuracy level of ≥ 80% required for 'accurate' conceptual cost estimations.
- The NN in this study was developed using simulated data that probably does not represent actual construction project cases. Thus, the results of this study were achieved upon simulated data in order to validate the method and to prove the concept of NNs. Therefore, the results of the study would probably differ if only actual project cost data had been used.

7.2 Practical recommendations

This study was done in collaboration with a Dutch construction firm, therefore the practical application of the developed NN tool is of high importance. In this paragraph, a practical recommendation regarding the implementation of the NN estimation tool will be provided.

Note that the developed NN tool in this study has not achieved the desired accuracy level yet. Therefore, the NN tool should first be revised, retrained, and tested on a higher number of qualitative actual project cost data samples. Once the NN has reached the desired accuracy level so that it can be useful for conceptual cost estimations, it can be applied for this task. Therefore, the practical recommendation provided below could not directly be made with the developed NN in this study. However, it could be used to improve the developed NNs and to support future practical applications of NNs in the construction industry.

Data is of high importance for the prediction performance of NNs. In this study, the NN was developed, trained, and tested on a relatively small amount of data. To improve the performance of the created NN, more data should be used to train it. Therefore, it is recommended that the construction firm starts collecting and storing project cost data in order to create a project cost database that can be used as (training) data source for the NN. Data of the used project input variables in this study, and their associated construction costs, should be gathered by the construction firm. This can be done simply by creating an Excel file in which these parameters are stored. As a starting point, the firm could aim at collecting at least 135 data cases, since this is the amount of data needed to fulfil the linear regression data condition which probably is also generalisable to NNs. Another important thing is that the NN is kept up to date in order to ensure its reliability. Therefore, it is important that the project cost database is updated frequently and consequently, and also, that the NN will be retrained frequently and consequently as well. Otherwise, the NN would just be a static tool that loses its usefulness.

Using the NN requires an understanding of the theory behind it. In addition, the NN is also considered to be a black-box technique. Therefore, understanding the reasoning of the NNs and the estimations made by them is difficult. As such, utilising an NN for conceptual cost estimations should be done

carefully. It is recommended to deploy the developed NN tool as an evaluation or validation tool (indicative) rather than using the NN solely for estimating the conceptual construction costs. It is recommended to use the NN estimation tool during the early stages of the construction process e.g. to validate whether the cost estimates made by the cost planner are in line with the cost calculations of past similar projects. By using the NN estimation tool, the cost planner is able to validate the made cost calculation fast and easily. If the estimation is proven to be in line with the construction cost of past similar project the cost planner can use this knowledge gained by using the NN for supporting its made cost calculation as it has been validated by an estimation tool that uses documented evidence.

Due to time and resource constraints, the developed NN tool now only consists of rough code in Python. Thus, only the back-end code (i.e. the NN code) for a cost estimation tool is available. However, for the easy application of a cost estimation tool, it is helpful to have an easy-to-use frontend user interface that is easy to understand by the end user. Therefore, it is a recommendation to develop a frontend interface for the NN cost estimation tool. In Figure 31, a concept of this frontend interface for the NN estimation tool is provided. In this concept, the cost planner can enter the values of the prediction parameters under the input header, run cost predictions under the cost estimating header, and evaluate the executed cost estimations under the output header. Besides providing a frontend interface, the estimation tool should also provide a function that updates the NN in the backend code, in order to be able to update the estimation tool and to improve its accuracy.

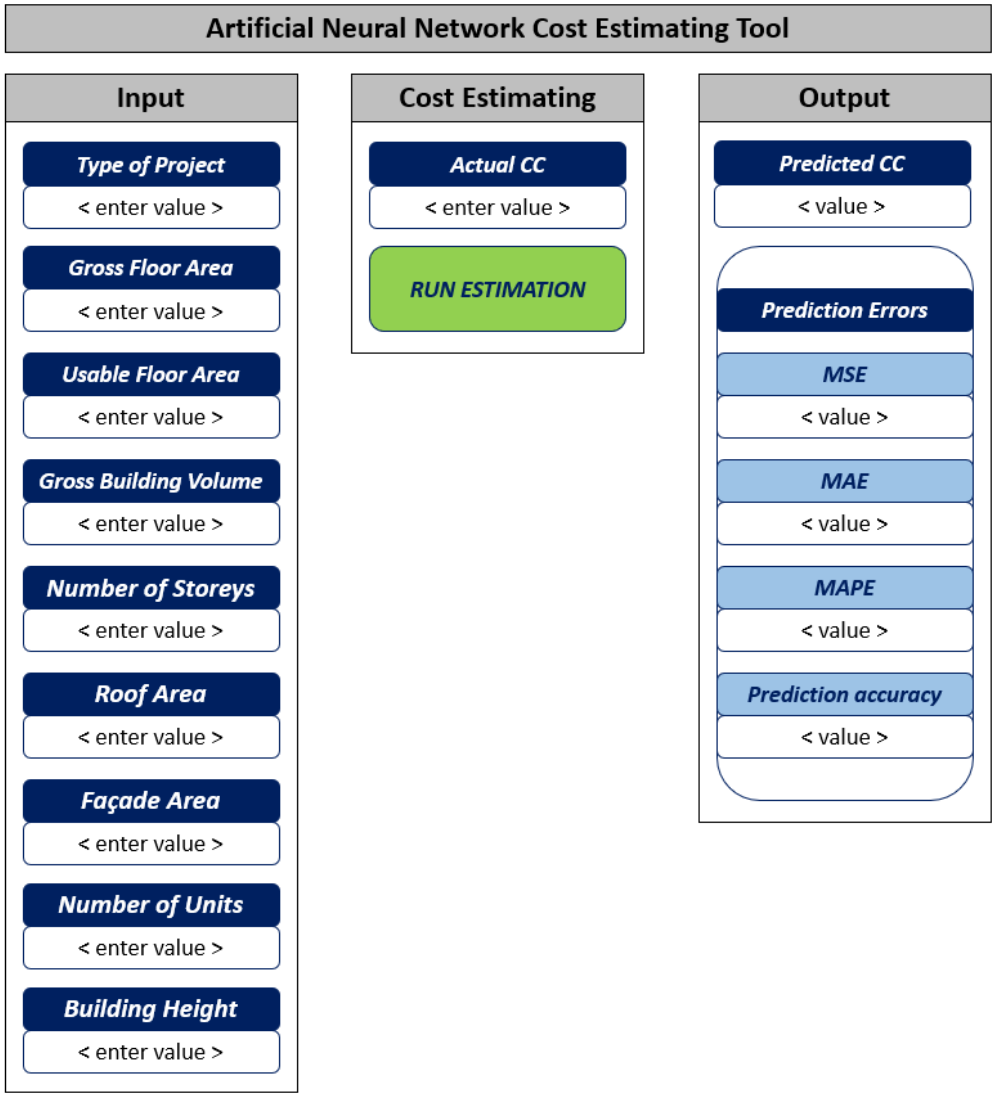


Figure 31: concept of front-end interface for developed NN estimation tool

7.3 Research recommendations

In this paragraph recommendations for further research will be discussed:

- Due to the limited data availability, the data used in this study has been simulated. The usage of this simulated data could have affected the results of the study, as simulated data does not represent reality. Therefore, it is recommended to validate the results of this study with further research by using actual project cost data for NN training, to see whether this ends up with different results compared to the presented study.
- In this study, the NN was more empirically than theoretically derived due to a missing, clear, scientifically proven methodology on NN development for estimations tasks. Therefore, a method has been constructed in this study based on literature and trial and error, in order to solve the issue of a missing methodology for NN development. However, whether the described method used for this study is correct and is applicable to future studies regarding NN studies for estimation tasks is unclear and should be validated in future research. Therefore, future research could focus on establishing a clear methodology for NN development for estimation tasks, with the methodology established in this study as a starting point. In future studies, researchers should focus more on the data conditions (i.e. amount of data required, the type of variables, data patterns of the variables, etc.) that apply to NNs as this is clearly missing at the moment. A good starting point here could be studying whether the data conditions that hold for linear regression can also be applied to NNs as well.
- The explicability of NNs could be a limitation for practically applying NNs for estimations tasks, as it does not reveal its reasoning. Despite the power of the data mining tool (i.e. NN), it still remains a garbage-in-garbage-out technique. When the NN algorithm is applied to low-quality data, it will fail to discover high-quality knowledge. Thus, one should never trust the NN model blindly. To improve the explicability of NNs, future research could perhaps focus on linking explainable AI (XAI) to statistical AI (i.e. e.g. predictive data mining algorithms as NNs).
- It was found in the literature that some studies used variations to traditional NNs or hybrid NNs, showing that hybrid NNs possibly outperform single NNs. If this holds true for this study as well was not studied due to time constraints and the set objective of solely applying single NNs for construction cost estimations. For future research, it is recommended to study whether hybrid NNs outperform single NNs on estimation accuracy when they are applied to the same construction cost estimation task.
- Nowadays, BIM forms an important data repository in the construction industry. The BIM model stores more than merely the geometric data of the project to be realised. Therefore, BIM models could potentially serve as an important (input) data source for NNs. However, retrieving the right amount of data from the right quality to be useful for NNs from BIM models remains a challenging task. Therefore, it would be interesting to study whether BIM and NNs can be linked and how BIM can serve as data source for NNs.
- An interesting application of NNs towards the integration of BIM software and NNs would be to study whether NNs have potential to be useful for the classification of clashes during a clash detection process. NNs are proven to be strong classifiers when it comes down to the classification of images. However, to what extent they can classify images of clashes in BIM models accurately is an interesting research question for new research regarding the application and utilisation of AI in the AEC industry.

References

AACE. (2003). *Cost Estimation Classification System.*

AbouRizk, S., Babey, G., & Karumanasseri, G. (2002). Estimating the cost of capital projects: an emperical study of accuracy levels for municipal government projects. *Canadian Journal of Civil Engineering, 29*, 653-661.

Abraham, A. (2005). Artificial Neural Networks. In P. Sydenham, & R. Thorn, *Handbook of Measuring System Design* (pp. 901-908). Oklahoma: John Wiley & Sons, Ltd.

Adeli, H., & Yeh, C. (1989). Perceptron Learning in Engineering Design. *Computer-Aided Civil and Infrastructure Engineering, 4*(4), 247-256.

Aggarwal, C. C., & Yu, P. S. (1999). Data Mining Techniques for Associations, Clustering and Classification. *PAKDD '99: Proceedings of the Third Pacific-Asia Conference on Methodologies for Knowledge Discovery and Data Mining*, 13-23.

Amarendra, K., Lakshmi, K. V., & Ramani, K. V. (2002). Research on Data Mining Using Neural Networks. *International Journal of Computer Science & Informatics, 2*(1).

Arabzadeh, V., Niaki, S., & Arabzadeh, V. (2018). Construction cost estimation of spherical storage tanks: artificial neural networks and hybrid regression - GA algorithms. *Journal of Industrial Engineering International, 14*, 747-756.

Arafa, M., & Alqedra, M. (2011). Early Stage Cost Estimation of Buildings Construction Projects using Artificial Neural Networks. *Journal of Artificial Intelligence, 4*(1), 63-75.

Attoh-Okine, N. (1999). Analysis of learning rate and momentum term in backpropagation neural network algorithm trained to predict pavement performance. *Advances in Engineering Software, 30*, 291-302.

BNA & NLingenieurs. (2014). *Standaardtaakbeschrijving DNR-STB 2014: Toelichting.* Retrieved from https://www.nlingenieurs.nl/assets/e4a75bb165/Standaardtaakbeschrijving-20141.pdf

Bosscha, E. (2016). *Big data in railway operations: Using artificial neural networks topredict train delay propagation.* Retrieved from https://essay.utwente.nl/71006/1/Bosscha%2C%20E.%200192341.pdf

Brownlee, J. (2020). *Deep Learning Performance*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

Burke, R. (2009). *Project Management: Planning and Control Techniques.* John Wiley & Sons, Inc.

Busoniu, L., Babuska, R., de Schutter, B., & Ernst, D. (2010). *Reinforcement Learning and Dynamic Programming Using Function Approximators.* CRC Press.

Cheng, M.-Y., Tsai, H.-C., & Sudjono, E. (2010). Conceptual cost estimates using evolutionary fuzzy hybrid neural network for projects in construction industry. *Expert Systems with Applications, 37*, 4224–4231.

Chou, J.-S., Yang, I.-T., & Chong, W. K. (2009). Probabilistic simulation for developing likelihood distribution of engineering project cost. *Automation in Construction, 18*(5), 570-577.

Computerworld. (2019, April 24). *Topics*. Retrieved from Computerworld: https://computerworld.nl/big-data/110247-ai-vereist-vaak-veel-meer-capaciteit-dan-beschikbaar-is

Domingos, P. (2015). *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World.* New York: Basic Books.

Doreswamy, H., & Hemanth, K. S. (2011). Hybrid Data Mining Technique for Knowledge Discovery from Engineering Materials Data Sets. *International Journal of Database Management Systems (IJDMS), 3*, 1-12.

Dysert, L. R. (2006). Is "estimate uncertainty" an oxymoron? *AACE International Transactions*.

Edelstein , H. (1999). *Introduction to Data Mining and Knowledge Discovery, Third Edition.* Potomac: Two Crows Corporation.

Elbeltagi, E., Hosny, O., Abdel-Razek, R., & El-Fitory, A. (2014). Conceptual Cost Estimate of Libyan Highway Projects Using Artificial Neural Network. *International Journal of Engineering Research and Applications, 4*(8), 56-66.

Elfahham, Y. (2019). Estimation and prediction of construction cost index using neural networks, time series, and regression. *Alexandria Engineering Journal*.

Elfaki, A. O., Alatawi, S., & Abushandi, E. (2014). Using Intelligent Techniques in Construction Project Cost Estimation: 10-Year Survey. *Advances in Civil Engineering*.

Elhag, T. M., & Boussabaine, A. H. (1998). An Artificial Neural System for Cost Estimation of Construction Projects. *14th Annual ARCOM Conference*.

Elmousalami, H. H. (2019). Intelligent methdology for project conceptual cost prediction. *Heliyon, 5*(5).

ElSawy, Hosny, H., & Abdel Razek, M. (2011). A Neural Network Model for Construction Projects Site Overhead Cost Estimating in Egypt. *International Journal of Computer Science Issues, 8*(3), 273-283.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. *AI Magazine, 17*(3), 37-54.

Flyvbjerg , B., Holm, M. S., & Buhl, S. (2002). Underestimating Costs in Public Works Projects: Error or Lie? *Journal of the American Planning Association, 68*, 279-295.

Flyvbjerg, B., Holm, M. S., & Buhl, S. (2003). How common and how large are cost overruns in transport infrastructure projects? *Transport Reviews, 23*, 71-88.

Friedman, J. H. (1997). *Data Mining and Statistics: What's the Connection?* Stanford: Stanford University.

Garson, G. (1991). Interpreting neural-network connection weights. *Artificial Intelligence Expert Volume 6*, 47-51.

Gilson, N. K., & Vanreyk, A. J. (2016). Review of Cost Estimation Models. *International Journal of Scientific Engineering and Research, 4*(3), 42-44.

Günaydın, H. M., & Doğan, S. Z. (2004). A neural network approach for early cost estimation of structural systems of buildings. *International Journal of Project Management*, 595-602.

Halpin, D. W., Senior, B. A., & Gunnar, L. (2017). *Construction Management.* John Wiley & Sons, Inc.

Han, J. W., Kamber, M., & Pei, J. (2012). *Data Mining Concepts and Techniques* (3rd ed.). Waltham, US: Morgan Kaufmann.

Hand, D. J. (1999). *Statistics and data mining: intersecting disciplines.* ACM SIGKDD Explorations Newsletter.

Hand, D., Mannila, H., & Smyth, P. (2001). *Principles of Data Mining.* Cambridge: MIT Press.

Hegazy, T., & Ayed, A. (1998). Neural Network Model for Parametric Cost Estimation of Highway Projects. *Journal of Construction Engineering Management, 124*(3), 210-218.

Hegazy, T., & Moselhi, O. (1994). Analogy-Based Solution to Markup Estimation Problem. *Journal of Computing in Civil Engineering, 8*(1), 72-87.

Hegazy, T., Fazio, P., & Mosheli, O. (1994). Developing Practical Neural Network Applications Using Back-Propagation. *Microcomputers in Civil Engineering, 9*, 145-159.

Hilgers, B. A. (2018). *Automated valuation models for commercial real estate in the Netherlands traditional regression versus machine learning techniques.* Retrieved from https://research.tue.nl/en/studentTheses/automated-valuation-models-for-commercial-real-estate-in-the-neth

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. Retrieved from arXiv: https://arxiv.org/abs/1207.0580

Hoehndorf, R., & Queralt-Rosinach, N. (2017). Data Science and symbolic AI: Synergies, challenges and opportunities. *Data Science, 1*(1-2), 27-38.

Hoogeveen, N. (2015). *Building Knowledge Modelling: From BIM Data to Artificial Intelligent Knowledge Systems.* Retrieved from https://repository.tudelft.nl/islandora/object/uuid:3bd80168-2326-40ed-92ca-9705dea4733f/?collection=research

Hovnanian, G., Kroll, K., & Sjödin, E. (2019). *Our insights.* Retrieved from Website of McKinsey&Company: https://www.mckinsey.com/industries/capital-projects-and-infrastructure/our-insights/how-analytics-can-drive-smarter-engineering-and-construction-decisions

Intrator, O., & Intrator, N. (2001). Interpreting neural-network results: a simulation study. *Computational Statistics & Data Analysis*, 373-393.

Ismail, S., Bandi, S., & Maaz, Z. (2018). An Appraisal into the Potential Application of Big Data in the Construction Industry. *Internation Journal of Built Environment and Sustainability, 5*(2), pp. 145-154.

Jain, A. K., Mao, J., & Mohiuddin, K. M. (1996). Artificial Neural Networks: A Tutorial. *Computer*, 31-44.

Jain, M., & Pathak, K. K. (2014). Applications of Artificial Neural Network in Construction Engineering and Management - A Review. *Inte rnational Journal of Engineering Technology, Management and Applied Sciences, 2*(3), 134-142.

Janssen, J. (2018). *Developing a model for estimating road capacity values for weaving sections.* Retrieved from https://www.utwente.nl/en/et/tem/education/Master/finished_graduation_projects/afstudeerders_per_jaar_2/pdf/2018-janssen-j.c.-1504002-openbaar.pdf

Juszczyk, M. (2017). The Challenges of Nonparametric Cost Estimation of Construction Works with the use of Artificial Intelligence Tools. *Procedia Engineering*, 415-422.

Kharoubi, Y. (2019). *Towards 5D BIM: A Process Map for Effective Design and Cost Estimation Integration.* Retrieved from http://resolver.tudelft.nl/uuid:b5539dcb-358b-410a-a3cb-21043a9228ce

Kim, G. H., An, S. H., & Kang, K. I. (2004). Comparison of construction cost estimating models based on regression analysis, neural networks, and case-based reasoning. *Building and Environment, 39*(10), 1235-1242.

Kim, G. H., Seo, D. S., & Kang, K. I. (2005). Hybrid Models of Neural Networks and Genetic Algorithms for Predicting Preliminary Cost Estimates. *Journal of Computing in Civil Engineering, 19*(2), 208-211.

Kim, H.-J., Seo, Y.-C., & Hyun, C.-T. (2012). A hybrid conceptual cost estimating model for large building projects. *Automation in Construction, 25*, 72-81.

Kim, P. (2017). *MATLAB Deep Learning.* Apress.

Klakegg, O. J., & Lichtenberg, S. (2016). Successive Cost Estimation – Successful Budgeting of Major Projects. *Procedia - Social and Behavioral Sciences*, 176-183.

Kulkarni, P. S., Londhe, S. N., & Deo, M. C. (2017). Artificial Neural Networks for Construction Management: A Review. *Journal of Soft Computing in Civil Engineering, 1*(2), 70-88.

Kwak, Y. H., & Watson, R. J. (2005). Conceptual estimating tool for technology-driven projects: exploring parametric estimating technique. *Technovation, 25*(12), 1430-1436.

Loy, J. (2018). *How to build your own Neural Network from scratch in Python*. Retrieved from Towards Data Science: https://towardsdatascience.com/how-to-build-your-own-neural-network-from-scratch-in-python-68998a08e4f6

Lu, W., Lai, C. C., & Tse, T. (2019). *BIM and Big Data for Construction Cost Management.* Routledge.

Luu, V. T., & Kim, S.-Y. (2009). Neural Network Model for Construction Cost Prediction of Apartment Projects in Vietnam. *Korean Journal of Construction Engineering and Management, 10*(3), 139-147.

Lyne, C., & Maximino, J. (2014). An Artificial Neural Network Approach to Structural Cost Estimation of Building Projects in the Philippines. *DLSU Research Congress 2014.* Manila.

Magdum, S. K., & Adamuthe, A. C. (2017). Construction Cost Prediction Using Neural Networks. *Journal of Soft Computing, 8*(1), 1549-1556.

Marker, A. (2017). *Smartsheet: The Ultimate Guide to Project Cost Estimating*. Retrieved from Smartsheet: https://www.smartsheet.com/ultimate-guide-project-cost-estimating

Matel, E. (2019). *An Artificial Neural Network Approach For Cost Estimation of Engineering Services.* Retrieved from https://essay.utwente.nl/78107/1/Matel%2C%20E.%201867482%20_openbaar.pdf

Merrow, E. W. (2011). *Industrial megaprojects. Concepts, Strategies and Practices for Success.* Hoboken, New Jersey: John Wiley and sons.

Minsky, M. (1991). Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *AI Magazine, 12*(2), 34-51.

NASA Executive Cost Analysis Steering Group. (2015). *NASA Cost Estimating Handbook.* NASA.

Nassar, K. (2006). Application of Data-Mining to tate Transportation Agencies' Project Databases. *ITcon, 12*, 139-149.

NEN. (2017). *NEN2699:2017 Investerings- en exploitatiekosten van onroerende zaken.* Retrieved from https://www.briswarenhuis.nl/docs/norm/nen2699-2017

Ni, X. (2008). Research of Data Mining Based on Neural Networks. *Journal of Engineering and Technology, 39*, 381-384.

Nicholas, J. M., & Steyn, H. (2017). *Project Management for Engineering, Business and Technology.* Routledge.

Olden, J., & Jackson, D. (2002). Illuminating the "black box": a randomization approach for understanding variable contribution in artificial neural networks. *Ecological Modelling 154*, 135-150.

Özesmi, S., & Özesmi, U. (1999). An artificial neural network approach to spatial habitat modeling with interspecific interaction. *Ecological Modeling 116*, 15-31.

Pal, B., Mhashilkar, A., Pandey, A., Nagphase, B., & Chandanshive, V. (2018). Cost Estimation Model (CEM) of Buildings by ANN (Artificial Neural Networks) - A Review. *International Advanced Research Journal in Science, Engineering and Technology, 5*(2), 26-28.

Paluszek, M., & Thomas, S. (2017). *MATLAB Machine Learning.* Apress.

Petroutsatou, K., Georgopoulos, E., Lambropoulos, S., & Pantouvakis, J. P. (2012). Early Cost Estimating of Road Tunnel Construction Using Neural Networks. *Journal of Construction Engineering and Management, 138*(6).

Petrova, E., Pauwels, P., Svidt, K., & Jensen, R. L. (2018). In Search of Sustainable Design Patterns: Combining Data Mining and Semantic Data Modelling on Disparate Building Data. In I. Mutis, & T. Hartmann, *Advances in Informatics and Computing in Civil and Construction Engineering - Proceedings of the 35th CIB W78 2018 Conference: IT in Design, Construction and Management.* Springer.

Petrova, E., Pauwels, P., Svidt, K., & Jensen, R. L. (2019). Towards Data-Driven Sustainable Design: Decision Support based on Knowledge Discovery in Disparate Building Data. *Architectural Engineering and Design Management, 15*(5), 334-356.

Phaobunjong, K. (2002). *Parametric cost estimating model for conceptual cost estimating of building construction projects.* Austin USA: the University of Texas.

Poole, D., & Mackworth, A. (2010). *Artificial Intelligence: Foundations of Computational Agents.* Cambridge: Cambridge University Press.

Prechelt, L. (2012). Early Stopping - But When? In G. Montavon, G. Orr, & K. Müller, *Neural Networks Tricks of the Trade* (pp. 53-67). Springer Berlin Heidelberg.

Project Management Institute. (2017). *A guide to the project management body of knowledge (PMBOK guide).* Pennsylvania: Project Management Institute, Inc.

Provost, F., & Fawcett, T. (2013). *Data Science for Business.* Sebastopol: O'Reilly.

Python. (n.d.). *About Python*. Retrieved from Python: https://www.python.org/

PyTorch. (n.d.). *PyTorch* . Retrieved from PyTorch: https://pytorch.org/

Ramos, D. (2017). *Smartsheet: Construction Cost Estimating*. Retrieved from Smartsheet: https://www.smartsheet.com/construction-cost-estimating

Rekveld, Y. (2017). *BIM Based Cost Estimation Knowledge Management.* Retrieved from https://www.ofcoursecme.nl/mdocs-posts/bim-based-cost-estimation-knowledge-management-the-application-of-a-knowledge-management-system-by-combining-bim-models-and-the-data-mining-process-to-estimate-the-costs-of-residential-buildings/

Roy, R. (2003). *Cost Engineering: Why, what and how?, Decision Engineering Report Series.* Cranfield University.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart, & J. L. McClelland, *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations* (pp. 318-362). Cambridge: MIT Press.

Samek, W., Wiegand, T., & Müller, K. R. (2017). Explainable Artificial Intelligence: Understanding, Visualizing and Interpretating Deep Learning Models. Retrieved from https://arxiv.org/abs/1708.08296

Sarle, W. S. (1994). Neural Networks and Statistical Models. *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, 1-13.

Schapire, R. (2008). *Theoretical Machine Learning.* Princeton: Princeton University Press.

Sharma, S. (2017). *Activation Functions in Neural Networks*. Retrieved from Towards Data Science: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Singh, Y., & Chauhan, A. S. (2009). Neural Networks in Data Mining. *Journal of Theoretical and Applied Information Technology*, 37-42.

Sonmez, R. (2004). Conceptual cost estimation of building projects with regression analysis and neural networks. *Canadian Journal of Civil Engineering, 31*(4), 677-683.

TensorFlow. (n.d.). *About TensorFlow*. Retrieved from TensorFlow: https://www.tensorflow.org/about/

TensorFlow. (n.d.). *Learn TensorFlow*. Retrieved from TensorFlow: https://www.tensorflow.org/guide/keras

Turban , E., Sharda, R., Delen, D., & King, D. (2010). Neural Networks for Data Mining. In E. Turban, R. Sharda, D. Delen, & D. King, *Business Intelligence: A Managerial Approach.* Prentice Hall.

U.S. Government Accountability Office. (2009). *GAO Cost Estimating and Assessment Guide.* Retrieved from https://www.gao.gov/new.items/d093sp.pdf

Vahdani, B., Mousavi, S. M., Mousakhani, M., Sharifi, M., & Hashemi, H. (2012). A Neural Network Model Based on Support Vector Machine for Conceptual Cost Estimation in Construction Projects. *Journal of Optimization in Industrial Engineering, 10*, 11-18.

Visual Studio Code. (n.d.). *Docs*. Retrieved from Visual Studio Code: https://code.visualstudio.com/

Walia, A. S. (2017). *Activation functions and it's types - Which is better?* Retrieved from Towards Data Science: https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f

Wang, Y.-R., Yu, C.-Y., & Chan, H.-H. (2012). Predicting construction cost and schedule success using artificial neural networks ensemble and support vector machines classification models. *International Journal of Project Management, 30*, 470-478.

Waziri, B. S., Bala, K., & Bustani, S. A. (2017). Artificial Neural Networks in Construction Engineering and Management. *International Journal of Architecture, Engineering and Construction, 6*(1), 50-60.

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks, 1*, 339-356.

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques Third Edition.* Burlington: Morgan Kaufmann Publishers.

Wu, X., & Liu, J. (2009). A New Early Stopping Algorithm for Improving Neural Network Generalization. *Second International Conference on Intelligent Computation Technology and Automation (IEEE Computer Society)*, 15-18.

Zhang, G., Hu, M. Y., & Patuwo, B. E. (1998). Forecasting with artificial neural networks: the state of the art. *International Journal of Forecasting, 14*(1), 35-62.

Zhou, L. (2018). *A Machine Learning Approach for Conceptual Cost Prediction of Road Pavement Project.* Retrieved from http://resolver.tudelft.nl/uuid:43053254-dee7-4262-9cdc-913549a18c4e

Zima, K. (2015). The Case-based Reasoning Model of Cost Estimation at the Preliminary Stage of a Construction Project. *Procedia Engineering*, 57-64.

Appendices

Appendix 1: Comparison table of cost estimating techniques

Appendix 2: Project dataset

Appendix 3: Python code of data generation tool

Appendix 4: Python code of Neural Network

Appendix 1

Comparison table of cost estimating techniques

This appendix provides a table that summarises a comparison of the five cost estimating techniques discussed in chapter 2, based on their strengths, weaknesses, and requirements. The table is based on the current research as well as on tables provided in the studies of Kharoubi (2019) and Matel (2019).

Estimating technique	Method	Strengths	Weaknesses	Requirements
Analogous estimating	Top-down	- Quick estimating method - Reasoning of estimating is easily understood - Useful in early phases with limited level of detail	- Normalisation of cost data required - Depends on data of similar projects - Hard to identify similar projects	- Knowledge and data of similar projects
Expert judgement	Top-down	- Draws comparisons between past and new projects	- Relies on information from individuals - Only done in initiative phase	- Experts for estimating - Knowledge and data of similar projects
Detailed estimating	Bottom-up	- Reliable, structured, and accurate estimate - Provides insight into major cost drivers of the project - Ability to determine what is included and excluded in the estimate	- Detailed project information must be available - Time consuming - High costs - New estimates for every project	- Detailed information about the project - Experts for estimating - Work breakdown structure - Man-hour estimates
Parametric estimating	Top-down	- Quick and accurate estimate - Estimates can be replicated easily - Eliminates the reliance on opinion through use of actual observations	- Identifying (CERs) is difficult - Improper use of CERs can lead to estimating errors - CERs must be revised continuously	- Historical data of prior projects - Statistical software - Statistical knowledge
Probabilistic estimating	Top-down	- Provides insight in the probability of cost overruns	- Cost distributions need to be identified for each cost component - Cost components correlations must be identified for every cost component - Probability distributions require continuous revisiting	- Historical data of prior projects - Statistical software - Statistical knowledge

Appendix 2

Project dataset

This appendix contains a table in which the complete dataset used in this study is listed. The data samples in bold represent the original 15 data samples gathered from a Dutch construction firm.

ToP	GFA	UFA	GBV	NoS	RA	FA	NoU	BH	CC
2	28310	22648	58884	38	1922	10000	250	122,0	41563532
2	3968	3174	12966	6	1710	3638	52	20,0	5035736
2	1974	1578	4971	9	732	2142	27	30,0	2973993
1	7756	5556	20202	3	4322	3998	62	19,0	8762000
2	1984	1587	6483	3	855	1819	26	10,0	2517868
2	8878	7102	18466	14	1760	4532	38	46,0	9353790
1	37698	30156	112815	3	19887	19866	186	28,5	26955000
1	7756	5556	20202	3	4322	3998	62	19,0	8762000
1	8338	5958	15490	3	4908	4116	66	19,0	8154000
2	5754	3466	17508	8	1394	2276	48	26,0	5858000
1	11806	9828	36798	3	3608	9058	62	19,0	11493248
2	13317	10653	27699	21	2640	6798	57	69,0	14030685
1	12507	8937	23235	3	7362	6174	99	28,5	12231000
2	28268	22628	86372	38	2154	14352	398	122,0	46865160
1	36154	28922	99034	3	16388	23850	248	19,0	31857604
1	8338	5958	15490	3	4908	4116	66	19,0	8154000
1	36154	28922	99034	3	16388	23850	248	19,0	31857604
1	25132	20104	75210	3	13258	13244	124	19,0	17970000
1	36154	28922	99034	3	16388	23850	248	19,0	31857604
2	5952	4761	19449	9	2565	5457	78	30,0	7553604
2	3968	3174	12966	6	1710	3638	52	20,0	5035736
2	28268	22628	86372	38	2154	14352	398	122,0	46865160
2	26402	22634	58848	34	2070	13474	210	110,0	35912366
2	28310	22648	58884	38	1922	10000	250	122,0	41563532
2	1984	1587	6483	3	855	1819	26	10,0	2517868
2	26402	22634	58848	34	2070	13474	210	110,0	35912366
2	31794	25436	66134	38	3466	18978	370	122,0	47341546
2	28268	22628	86372	38	2154	14352	398	122,0	46865160
2	41404	27638	71858	48	2292	24904	316	154,0	62778506
2	31794	25436	66134	38	3466	18978	370	122,0	47341546
1	36154	28922	99034	3	16388	23850	248	19,0	31857604
1	8338	5958	15490	3	4908	4116	66	19,0	8154000
2	31794	25436	66134	38	3466	18978	370	122,0	47341546
1	25132	20104	75210	3	13258	13244	124	19,0	17970000
2	13317	10653	27699	21	2640	6798	57	69,0	14030685
1	**12566**	**10052**	**37605**	**3**	**6629**	**6622**	**62**	**9,5**	**8985000**
1	**4169**	**2979**	**7745**	**3**	**2454**	**2058**	**33**	**9,5**	**4077000**
2	**2877**	**1733**	**8754**	**4**	**697**	**1138**	**24**	**13,0**	**2929000**
2	**14134**	**11314**	**43186**	**19**	**1077**	**7176**	**199**	**61,0**	**23432580**

2	**13201**	**11317**	**29424**	**17**	**1035**	**6737**	**105**	**55,0**	**17956183**
2	**4439**	**3551**	**9233**	**7**	**880**	**2266**	**19**	**23,0**	**4676895**
2	**14155**	**11324**	**29442**	**19**	**961**	**5000**	**125**	**61,0**	**20781766**
2	**1984**	**1587**	**6483**	**3**	**855**	**1819**	**26**	**10,0**	**2517868**
1	**5903**	**4914**	**18399**	**3**	**1804**	**4529**	**31**	**9,5**	**5746624**
1	**3878**	**2778**	**10101**	**3**	**2161**	**1999**	**31**	**9,5**	**4381000**
2	**31718**	**25374**	**71047**	**13**	**3503**	**7260**	**233**	**42,0**	**37710000**
2	**15897**	**12718**	**33067**	**19**	**1733**	**9489**	**185**	**61,0**	**23670773**
1	**18077**	**14461**	**49517**	**3**	**8194**	**11925**	**124**	**9,5**	**15928802**
2	**20702**	**13819**	**35929**	**24**	**1146**	**12452**	**158**	**77,0**	**31389253**
2	**658**	**526**	**1657**	**3**	**244**	**714**	**9**	**10,0**	**991331**

Appendix 3

Python code of data generation tool

This appendix contains the code of the data simulation tool that was developed in Python in order to simulate new data samples based on the originally gathered 15 data samples from a Dutch construction firm.

```
import utility as ut
import numpy as np
import tqdm
import csv
import argparse
from termcolor import cprint


def check_validity(data_item, mult_fact):
    if data_item[0] == 1:
        return "house"
    elif data_item[0] == 2:
        if data_item[4] * mult_fact > 100:
            return "invalid"
        else:
            return "appartment"
    elif data_item[0] == 2:
        if data_item[7] * mult_fact > 500:
            return "invalid"
        else:
            return "apartment"
    elif data_item[0] == 2:
        if data_item[8] * mult_fact > 120:
            return "invalid"
        else:
            return "apartment"

def gen_type_prob(_data_list):
    type_prob = np.zeros((1, 2))
    for item in _data_list:
        if item[0] == 1:
            type_prob[0, 0] += 1
        elif item[0] == 2:
            type_prob[0, 1] += 1
        else:
            cprint("Type Error: ", "red", end="", attrs=["bold"])
            cprint("Unknown", "red", attrs=["blink", "bold"])
    sum_val = np.sum(type_prob)
    type_prob[0, 0] = type_prob[0, 0] / sum_val
    type_prob[0, 1] = type_prob[0, 1] / sum_val
    return type_prob
```

```
def generate_house(item, mult_fact):
    item_tag = 1
    gross_floor_area = item[1] * mult_fact
    usable_floor_area = item[2] * mult_fact
    gross_building_volume = item[3] * mult_fact
    number_of_stories = item[4]
    roof_area = item[5] * mult_fact
    facade_area = item[6] * mult_fact
    number_of_units = item[7] * mult_fact
    building_height = item[8] * mult_fact
    construction_cost = item[9] * mult_fact
    new_item = [
        item_tag,
        gross_floor_area,
        usable_floor_area,
        gross_building_volume,
        number_of_stories,
        roof_area,
        facade_area,
        number_of_units,
        building_height,
        construction_cost,
    ]
    return new_item

def generate_appartment(item, mult_fact):
    item_tag = 2
    gross_floor_area = item[1] * mult_fact
    usable_floor_area = item[2] * mult_fact
    gross_building_volume = item[3] * mult_fact
    number_of_stories = item[4] * mult_fact
    roof_area = item[5] * mult_fact
    facade_area = item[6] * mult_fact
    number_of_units = item[7] * mult_fact
    building_height = item[8] * mult_fact
    construction_cost = item[9] * mult_fact
    new_item = [
        item_tag,
        gross_floor_area,
        usable_floor_area,
        gross_building_volume,
        number_of_stories,
        roof_area,
        facade_area,
        number_of_units,
        building_height,
        construction_cost,
    ]
```

```
    return new_item
def data_generator(_data_list, num_samples):
    type_prob = gen_type_prob(_data_list)
    generated_data = []
    while len(generated_data) != num_samples:
        building_type = np.random.choice([1, 2], p=type_prob[0])
        locations = np.where(_data_list[:, 0] == building_type)
        random_item = _data_list[np.random.choice(locations[0])]
        multiplication_factor = np.random.choice([1, 2, 3], p=[0.6, 0.3, 0.1])
        validity_status = check_validity(random_item, multiplication_factor)

        if validity_status == "invalid":
            cprint("Error: Stories out of bounds", "red", attrs=["bold"])
            continue
        elif validity_status == "house":
            new_item = generate_house(random_item, multiplication_factor)
            cprint("x---Successfully Generated House---x", "green", attrs=["bold"])
        elif validity_status == "appartment":
            new_item = generate_appartment(random_item, multiplication_factor)
            cprint("x---Successfully Generated Appartment---x", "green", attrs=["bold"])
        else:
            cprint("Status Unknows", "red", attrs=["bold"])
            exit(0)
        generated_data.append(new_item)
    return generated_data

def main(save_path, num_samples):
    dl = ut.read_csv("data/Dataset_SDK2.1.csv", panda=False)
    #Change Path here to the currently existing csv
    dl_np = np.array([np.array(x, dtype=float) for x in dl])
    generated_data = data_generator(dl_np, num_samples)
    with open(save_path, "w") as writefile:
        wr = csv.writer(writefile)
        wr.writerows(generated_data)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "-s", "--save_at", required=True, help="path for saving the file")
    parser.add_argument(
        "-n",
        "--num_samples",
        type=int,
        required=True,
        help="number of samples to be generated", )

    args = parser.parse_args()
    save_path = args.save_at
    num_samples = args.num_samples
```

```
main(save_path, num_samples)
```

Appendix 4

Python code of Neural Network

This appendix provides the codes of the Neural Networks that were developed in Python by using the PyTorch (i.e. torch.nn) ML library, as well as the Tensorflow Keras ML library. First, the NN code used during this study made with the PyTorch ML library will be presented. Subsequently, for those interested, the code of the NN developed with the TensorFlow Keras will be provided as well.

PYTORCH NN MODEL

Code of NN training model

```
import torch
from torch.autograd import Variable
import torch.nn.functional as F
import torch.utils.data as Data

import matplotlib.pyplot as plt

import pandas as pd
import numpy as np
import argparse

def create_model():
   return torch.nn.Sequential(
         torch.nn.Linear(9, 19),
         torch.nn.LeakyReLU(),
         torch.nn.Dropout(0.5),

         torch.nn.Linear(19, 1),
      )

f = open("output_train_NN19_E1000.txt", "w+")
for x in range(1):

   f.write('\n\nrun ' + str(x) + '\n--------------------------------\n\n')

   def train(model, csv_name, num_epochs=1000, batch_size=5):
      # Read training data from CSV.
      dataframe = pd.read_csv(csv_name, delimiter=",", header=None)
      dataset = dataframe.values

      # Calcualate Means and Standard Deviations
      means = np.average(dataset, axis=0)[None, :]
      std = np.std(dataset, axis=0)[None, :]

      # Normalize the dataset
      dataset = (dataset - means) / std
```

```
    # Split into inputs and labels.
    x = dataset[:,0:9]
    y = dataset[:,9]

    # Create Tensors for usage with pytorch
    x, y = torch.Tensor(x), torch.Tensor(y)

    optim = torch.optim.SGD(model.parameters(), lr=0.001, momentum=0.45, weight_decay=0.001)
    criterion = torch.nn.SmoothL1Loss()

    dataloader = Data.DataLoader(
        dataset=Data.TensorDataset(x, y),
        batch_size=5,
        shuffle=True
    )

    model.train()

    epoch_losses = []
    for epoch in range(num_epochs):
        losses=[]
        for iter, (b_x, b_y) in enumerate(dataloader):
            prediction = model(b_x)

            loss = criterion(prediction, b_y.unsqueeze(-1))

            losses.append(loss.item())
            optim.zero_grad()
            loss.backward()
            optim.step()

        epoch_losses.append(np.average(losses))
        if epoch % 5 == 0:
            print("Epoch #:", epoch, "Loss=", np.average(epoch_losses[-5:]))
            losses = []
    f.write('Epoch: ' + str(epoch) + '\n')
    f.write('Loss: ' + str(epoch_losses) + '\n')

    plt.plot(epoch_losses)
    plt.title('Epoch Losses')
    plt.ylabel('Epoch loss')
    plt.xlabel('Epoch #')
    plt.show()

    return means, std
```

```
if __name__ == "__main__":
    parser = argparse.ArgumentParser()

    parser.add_argument(
      "-csv", type=str, required=True, help="path to csv file containing training data.")

    parser.add_argument(
      "-model", type=str, required=True, help="file path for saving trained model")

    args = parser.parse_args()

    model = create_model()
    means, std = train(model, "training_data.csv")
    torch.save({
      "model_state_dict": model.state_dict(),
      "means": means,
      "std": std
      }, args.model)

f.close()
```

Code of NN evaluation model

```
import torch
from torch.autograd import Variable
import torch.nn.functional as F
import torch.utils.data as Data

import matplotlib.pyplot as plt

import pandas as pd
import numpy as np
import argparse

def create_model():
  return torch.nn.Sequential(
      torch.nn.Linear(9, 19),
      torch.nn.LeakyReLU(),
      torch.nn.Dropout(0.5),

      torch.nn.Linear(19, 1),
    )

f = open("output_eval_NN19_SL1_E1000_opt_top2.txt", "w+")
for x in range(10):

  f.write('\n\nrun ' + str(x) + '\n--------------------------------\n\n')
```

```
    def evaluate(model, csv_name, means, std):
    # Read training data from CSV.
    dataframe = pd.read_csv(csv_name, delimiter=",", header=None)
    dataset = dataframe.values

    # Normalize the dataset
    dataset = (dataset - means) / std

    # Split into inputs and labels.
    x = dataset[:,0:9]
    y = dataset[:,9]

    # Create Tensors for usage with pytorch
    x, y = torch.Tensor(x), torch.Tensor(y)

    dataloader = Data.DataLoader(
        dataset=Data.TensorDataset(x, y),
        batch_size=1
    )

    model.eval()

    print("All values are in millions.")

    ground_truths = []
    predictions = []
    for iter, (b_x, b_y) in enumerate(dataloader):
        prediction = model(b_x)
        b_y        = (b_y[0].item()       * std[0, -1]) + means[0, -1]
        prediction  = (prediction[0].item() * std[0, -1]) + means[0, -1]

        b_y        *= 1e-6
        prediction  *= 1e-6

        ground_truths.append(b_y)
        predictions.append(prediction)

        print("Ground Truth=" + str(b_y), "  \tPrediction=" + str(prediction))
        f.write('Ground Truth=' + str(b_y) + '\n')
        f.write('Prediction=' + str(prediction) + '\n')
    print("Mean Error = ", np.mean(np.abs(np.array(ground_truths) - np.array(predictions))))
    f.write('Mean Error=' + str(np.mean(np.abs(np.array(ground_truths) - np.array(predictions)))) + '
\n')
```

```
    if __name__ == "__main__":
    parser = argparse.ArgumentParser()

    parser.add_argument(
        "-csv", type=str, required=True, help="path to csv file containing test data.")

    parser.add_argument(
        "-model", type=str, required=True, help="file path of trained model")

    args = parser.parse_args()

    model = create_model()

    checkpoint = torch.load(args.model)
    model.load_state_dict(checkpoint['model_state_dict'])

    evaluate(model, "test_data_top2.csv", checkpoint["means"], checkpoint["std"])

f.close()
```

TENSORFLOW NN MODEL

```
# Importing libraries
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import SGD

# Importing data
df = pd.read_csv(r'C:\Users\baspa\OneDrive\Graduation TUe\02. Thesis\Data collection\Dataset\ANN SDK development\Dataset SDK 2.1\training_data.csv', delimiter=',',
            names = ['ToP', 'GFA', 'UFA', 'GBV', 'NoS', 'RA', 'FA', 'NoU', 'BH', 'CC'])

#print("Dataframe: ")
#print(df)

# Command for file writing
f = open("output_TF1.txt", "w+")

# Splitting data into input (X) and output (Y) variables
x_train = df.iloc[:, 0:9]
y_train = df.iloc[:, 9]

for x in range(1):

   f.write('\n\nrun ' + str(x) + '\n--------------------------------\n\n')

   # Definining the Keras model
   model = Sequential()
   model.add(Dense(9, input_dim=9, kernel_initializer='normal', activation='relu'))
   model.add(Dense(1, activation='relu'))

   # Compiling the Keras model
   opt = SGD(learning_rate=0.35, momentum=0.45)
   model.compile(loss='msle', optimizer=opt, metrics=['msle'])

   # Fit the Keras model on the data
   history = model.fit(x_train, y_train, epochs=100, batch_size=5, verbose=1)
   tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)

   # Evaluate the Keras model
   _, accuracy = model.evaluate(x_train, y_train)
   a = '%.2f' % (accuracy * 100)
   #print("x: " + str(a))
   #print("y: " + str(abs(float(a) - 26301.29)))
```

```
    if abs(float(a) - 27680.65) < 0.00001:
        print('breaking')
        continue

    print('Accuracy: %.2f' % (accuracy * 100))
    f.write('Accuracy: %.2f' % (accuracy * 100) + '\n')

    # Predicting with the Keras model
    test_data = np.array([1, 12566, 10052, 37605, 3, 6629, 6622, 62, 9.5]) #Actual CC = 8985000
    x = model.predict(test_data.reshape(1,9), batch_size=1)

    print("Predicted CC: ")
    print(x[0][0])
    f.write(str(x[0][0]) + '\n')

    # Code lines to write data to .txt file
    # list all data in history
    # print(history.history.keys())
    # print(history.history.values())
    #f.write('val_loss: ' + str(history.history['val_loss']) + '\n')
    #f.write('val_msle: ' + str(history.history['val_msle']) + '\n')
    f.write('loss: ' + str(history.history['loss']) + '\n')
    f.write('msle: ' + str(history.history['msle']) + '\n')

f.close()

plt.plot(history.history['loss'])
#plt.plot(history.history['val_loss'])
plt.title('Epoch loss')
plt.ylabel('Epoch loss')
plt.xlabel('Epoch #')
#plt.legend(['train', 'test'], loc='upper left')
plt.show()
```