Animal ID Tag Recognition with Convolutional and Recurrent Neural Network
Identifying digits from a number sequence with RCNN

Bachelor Degree Project in Information Technology
Basic level 30 ECTS
Spring 2019

Issa Hijazi, Pontus Pettersson

Supervisor: Niclas Ståhl
Examiner: Joe Steinhauer
Abstract

Major advances in machine learning have made image recognition applications, with Artificial Neural Network, blossom over the recent years. The aim of this thesis was to find a solution to recognize digits from a number sequence on an ID tag, used to identify farm animals, with the help of image recognition. A Recurrent Convolutional Neural Network solution called PPNet was proposed and tested on a data set called Animal Identification Tags. A transfer learning method was also used to test if it could help PPNet generalize and better recognize digits. PPNet was then compared against Microsoft Azures own image recognition API, to determine how PPNet compares to a general solution. PPNet, while not performing as good, still managed to achieve competitive results to the Azure API.

Keywords: Machine Learning, Convolutional Neural Network, Recurrent Neural Network, Transfer learning, Microsoft Azure
Acknowledgement

We would like to sincerely thank our supervisor Niclas Ståhl, from the University of Skövde which has supported and helped us throughout the study. We would also like to thank CGI Skara, especially Alexander Kelman, which has supported and provided us the possibility to write this thesis.
## Contents

1 Introduction 1

2 Background 2

2.1 Terminology 2
2.2 Machine learning 3
2.3 Supervised learning 3
2.4 Artificial Neural Network 3
   2.4.1 Fully connected neural network 5
   2.4.2 Convolutional Neural Network 6
   2.4.3 Recurrent Neural Network 7
2.5 Transfer learning 8
   2.5.1 Street View House Numbers Dataset 9
2.6 Data augmentation 9
2.7 Microsoft Azure 10
2.8 Related Work 11

3 Problem 13

3.1 Aim 13
3.2 Motivation 13
3.3 Research Questions 14
3.4 Hypothesis 14
3.5 Objectives 15
3.6 Method 15
   3.6.1 Alternative methods 15
   3.6.2 Data Collection (Objective 2) 16
   3.6.3 Preparations for PPNet (Objective 5) 16
   3.6.4 Preparations for transfer learning (Objective 3, 6) 17
   3.6.5 Training and Testing 18
   3.6.6 How to compare results 18

4 Implementation 20

4.1 The Animal Identification Tag Data set (Objective 2) 20
4.2 Pre-process the data set (Objective 4) 21
4.3 Data augmentation for PPNet (Objective 5) 21
4.4 The PPNet architecture (Objective 5) 22
4.5 Transfer Learning (Objective 6) 22

5 Results 23

5.1 Presentation 23
   5.1.1 Present result of PPNet1 (Objective 6) 23
   5.1.2 Present result of PPNet2 (Objective 7) 23
   5.1.3 Present result of Azure (Objective 8) 23
5.2 Analysis 23
1 | Introduction

Major advances in machine learning have made its applications on different domains, blossom over the recent years. A big factor for this is Artificial Neural Networks (ANN) which uses raw data to learn and discover patterns, which it can use to detect or classify a given input. A lot of tasks that could be automated with artificial intelligence are still being performed by humans. An example is when farm animals are being transported and each animal must be identified. A common industrial practice is to manually retrieve the number sequence from an ID tag used to identify an animal and then document the event that takes place. Due to a large number of animals being transported each day, this becomes time and money consuming for a lot of companies.

The aim of this study was to find a potential ANN solution that can recognize digits from an ID tag with the use of image recognition. To accomplish the aim, a case study was conducted where a model called PPNet, which uses both a Recurrent and Convolutional Neural Network (RCNN), was created and tested with the measurements character level accuracy (CLA) and Levenshtein distance (edit distance). To train and test PPNet a data set called Animal Identification Tags (AIDT) had to be created which contained images of ID tags that have previously been attached to farm animals. The data set contained 350 images which were split up in 280 for training and 70 for testing PPNet. Due to the small size of AIDT, a transfer learning method was tested where a similar data set to AIDT, called Street View House Numbers (SVHN), was used which also contain number sequences. To test this, two different sets of PPNet classifiers were trained and tested. The intent was to determine if this method of transfer learning could help PPNet generalize better and learn to predict unseen data from the AIDT data set. The two sets of PPNet was then compared against Microsoft Azures own image recognition API (Azure), which was done to determine how PPNet compared to a general solution.

Both classifiers of PPNet and Azure managed to achieve over 45.0% CLA. This target was defined together with CGI Skara, where this study was conducted, as this would most likely be enough to identify an animal that is being transported. The results indicate that PPNet, while not performing as good, still managed to achieve competitive results to Azure. The transfer learning method didn’t seem to have any effect on the end performance of PPNet. The authors believe if certain changes were to be made on either the AIDT data set or the PPNet model, better performance than Azure can be achieved. The results received in this study, however, shows that Azure is preferable over PPNet and also require less work due to Azure already being trained.
2 | Background

Many tasks demand visual observations to be performed e.g. waiting for a traffic light switch, finding an empty parking spot, or classifying an object. Many of these tasks are performed manually by humans but have the potential to be automated if the evaluation can be performed with high enough accuracy which varies depending on the task. Image recognition has for a long time been a hot but very complicated topic until recent years, where ANN has received major improvements after the success of [Krizhevsky et al. (2012)]. These have drastically improved accuracy in predicting various objects, or contexts of an image to the point where it is being used to solve real-life problems. This section is used to first describe the terminology used in this thesis. It then describes what machine learning is, various concepts and methods within machine learning that have been used in this thesis, and ends with an explanation of earlier works that have been written in similar areas.

2.1 Terminology

This section is used to define certain terminology used in this thesis.

Model: An abstract representation of an ANN.

Class: A potential predicted outcome.

Classifier: A manifestation of a model that can make predictions and determine which features belong to which class.

Training set: The sample of a data set which is used to fit a classifier.

Test set: The sample of a data set which is used to provide an unbiased evaluation of a classifier. This means that the learning algorithm never has seen this sample.

Epoch: When a classifier has made a forward and a backward pass of all the examples in the training set, i.e it has seen all training examples.

Feed-forward network: An Artificial Neural Network where no cyclic connections exist and the data flow moves only in one direction; from the input to the output.

Label: The correct output to a given input. In this study, the label is a sequence of eleven digits which represent the number sequence from the ID tag.
2.2 Machine learning

Machine learning is a set of data-driven algorithms, which are constructed to enable improvement of performance in a particular task by providing it with previous experience, on that particular task, in the form of data (Domingos, 2012). A commonly recited definition of machine learning is "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" (Mitchell, 1997). Machine learning has, according to Granello and Wheaton (2004), seen a rise in popularity with the rise of big data. LeCun et al. (2015) write that machine learning is also being applied to perform real-life tasks, e.g. filtering information on social media, finding objects in an image, and converting speech to text. A lot of different algorithms are included in the term machine learning, however, according to LeCun et al. (2015), traditional machine learning algorithms, while usefully for certain tasks, are getting outdated due to the limitations in scaling with large amounts of data. The opposite is happening to ANN’s which uses raw data to learn and automatically discover patterns to then detect or classify a given input. Machine learning algorithms are trained with different methods e.g. supervised, unsupervised, or reinforcement learning (Haykin et al., 2009). Which method to use depends on the algorithm and how the data is structured. In this study, an ANN model was built to train with supervised learning due to the availability of a pre-labeled data set.

2.3 Supervised learning

Supervised learning is an input-to-output mapping technique where it uses pairs of values containing the input and the correct output value. The pair of values are used to help the learning algorithm correctly map an input to the correct output (LeCun et al., 2015). Data is often divided into parts to not only train but to also decide how well the learning algorithm can predict unseen data. In this study, the data was split into two parts, the training set, and the test set. Supervised learning is sometimes called Learning with a teacher, which can be due to their similarities where the student is fed with the knowledge to later perform a test to decide the effect of the fed knowledge. An example of this could be showing the learning algorithm images in which it is told that these images contain a cat. When this is done, new images are shown and due to the previous experience, better predictions can now be made when a cat exists in an image (Haykin et al., 2009).

2.4 Artificial Neural Network

Marblestone et al. (2016) writes that ANN’s were inspired by the functionality of how the brain handles and interprets information. An ANN is made up of a set of neurons in different layers. Every neuron, except for those in the last layer, is connected to a set of neurons in the forward adjacent layer with connection strength called weight. An ANN can be made up of three kinds of layers, shown in Figure 2.7, the input layer which is the first layer, the output layer which is the last layer and the hidden layers which exist between the input and output layer. To train the network to predict the correct output, a forward propagation is first done in which an input is received and propagated from the input layer to the output layer. The neurons inside each layer receive a set of inputs,
perform a computation on them, and outputs a new value, through the weights. The output is sent to the connected neurons in the next layer which repeats the process. This is repeated until the output layer is reached. The output layer outputs a final prediction of the input, given to the input layer. To enable the network to recognize non-linear patterns as well as limiting the amplitude of the output of a neuron an activation function, which in this study used Leaky Relu \( \text{(Maas et al., 2013)} \), is added after each neuron \( \text{(Glorot and Bengio, 2010)} \). The Softmax activation function \( \text{(Gold et al., 1996)} \) is also used to calculate the probability distribution of every class in an ANN when a prediction has been made in the output layer. A more in-depth explanation of how a neural operate can be seen in Figure 2.1. When a forward propagation of the input is completed, an evaluation is done with a loss function, e.g. cross entropy \( \text{(Rubinstein and Kroese, 2013)} \), to measure how much the prediction diverges from the label. \( \text{Goodfellow et al., (2016)} \) writes that "The training examples specify directly what the output layer must do at each point \( x \); it must produce a value that is close to \( y \). The behavior of the other layers is not directly specified by the training data. The learning algorithm must decide how to use those layers to produce the desired output, but the training data don’t say what each layer should do." The value from the loss function is used to make small changes on the network with the help of an optimization algorithm like gradient descent \( \text{(LeCun et al., 1998)} \) to reduce the value from the loss function and make the network produce the correct result. This process is known as back-propagation \( \text{(Haykin et al., 2009)} \text{, Hecht-Nielsen, 1992, LeCun et al., 2012)} \). Different architectures of ANN’s have been tested since first introduced due to the many different layouts and functions that can be applied to a network. Other parameters to also consider are the connection between the layers as well as the number of layers that should be added, depending on what kind and amount of data that is being used as well as time limitations. Larger networks usually need more data or else they would need to do more epochs to learn. If the data set size is limited the network can easily overfit due to it being trained on the same data, too many times. They also need more computational power, due to more parameters on the network, to be trained to sufficient accuracy, but research also suggests that a larger network enables the neural network to achieve better results \( \text{(Szegedy et al., 2015)} \).
\[ v = \sum_{n=1}^{m} w_n \cdot x_n \quad y = \varphi(v + b) \]

**Figure 2.1:** A visual representation of the internal structure of a neuron. The neuron receives a set of inputs \( x_{1-m} \), through their respective weight \( w \) that are connected to the neurons in the previous layer. The sum of the product for each \( x \) and \( w \) are calculated with the formula for \( v \). \( v \) is forwarded to the activation function \( \varphi(\cdot) \) with a bias \( b \) which is used to increase or decrease the activation threshold. The result of the neuron is outputted through \( y \), calculated with the formula for \( y \), which can be different depending on which activation function that is used. Why this structure is used is out of the scope for this thesis, but can be read in [Haykin et al. (2009)](#).

### 2.4.1 Fully connected neural network

A fully connected neural network is a network where each neuron in every layer is connected to every neuron in the forward adjacent layer. Figure 2.2 represents a fully connected neural network with ten neurons. For these ten neurons there exist 21 weights and by adding another neuron to the network at least three new weights will have to be included. High-resolution images can be larger than 800 in both height and width while also having three layers in-depth for red, green, and blue color. Each neuron after the input layer for these images would need to have \( 800 \cdot 800 \cdot 3 = 1,920,000 \) weights connected to it. This deteriorates scalability due to a large number of connections between the neurons as more calculations need to be done. Due to a neuron being connected to all neurons in the forward adjacent layer, make it hard for the fully connected neural network to learn features in a particular place on the image. Both these problems can be solved by using a Convolution Neural Network (CNN) ([Haykin et al. 2009](#)).
2.4.2 Convolutional Neural Network

The main difference between a fully connected neural network and a CNN is that, for a CNN, a neuron in the hidden or output layer only connects to a subset of neurons of the previous layer, called the receptive field, instead of complete connection between the layers. Fewer connections enable faster learning and better accuracy with fewer data and enable larger networks to be built. The theoretical foundation of CNN was first documented by [Hubel and Wiesel (1959, 1962)], who showed that certain neurons in a cat’s brain responded to a particular behavior e.g. exposure to spots of light which seemed to stimulate certain neurons. This behavior has afterward been mimicked in ANN’s by having a neuron connected to a subset of neurons, called the receptive field [LeCun et al., 1998]. This enables the extraction of low-level features of an input e.g. image or time series. These low-level features are combined over multiple layers to recognize high-level features and at the end decide an output e.g. an object in an image. To enable different patterns to occur in the same part of the input, multiple feature maps, which are matrices of values connected to the previous layer to find local features, are used in each convolutional layer. The number of feature maps in a convolutional layer is referred to as depth. Another component used for CNN is pooling layers. They result in smaller feature maps shown in Figure 2.3 where the first and second layer remains the same depth, but the size of the feature maps are reduced. The new feature maps are used to preserve the features but not as an accurate description of their position as the earlier feature maps. This is however not always the case as valuable information about the features can be deleted. The benefits of pooling layers are a more compact representation of the earlier feature maps which results in fewer calculations, which enables a larger network to be built. This also reduces the effect of noise as well as making the network less likely to overfit as the exact position of the features is less relevant. Different types of pooling layers like max, average, and sum pooling exist where all have similar benefits [Boureau et al., 2010; Haykin et al., 2009; LeCun et al., 1999; Scherer et al., 2010].
2.4.3 Recurrent Neural Network

A Recurrent Neural Network (RNN) is a neural network that makes use of a recurrent layer. A recurrent layer, shown in Figure 2.4b, is similar to a fully connected layer but also makes use of a cyclic path to connect each neuron to itself. This is used to enable a neuron to send its output, which is a prediction of its received input, to itself. This enables it to produce multiple outputs or sequential data from the same original input and then learn to consider context and relations in the input. RNN has therefore successfully been used in the context of natural language processes, video sequences, and object recognition in an image. Examples, where RNN’s have been used, are translation between languages where the RNN receives a text sequence and produces another text sequence, as well as the creation of captions for images by first identifying the objects and with a recurrent layer interpreting the relation between the objects. RNN’s changes the general data-flow which is seen in a feed-forward network due to it being able to send the output to itself instead of just the next layer. This does however not change how RNN’s learn when back-propagating through the network as according to Goodfellow et al. (2016), the recurrent layer will then behave as a set of layers. This does, however, mean that the weights that connect the recurrent layer to itself will be updated multiple times. This often leads to the weight either vanishing or exploding when using back-propagation. Different solutions have been tried to solve these problems with one of these, suggested by Hochreiter and Schmidhuber (1997), is Long-Short-Term-Memory (LSTM). LSTM is being used to solve a lot of different problems (Graves and Schmidhuber 2005, Greff et al., 2017). LSTM is an extension of a regular RNN but with added components such as memory blocks that allow it to process an arbitrary number of states in memory cells which solves the issues of regular RNN’s when using back-propagation and also enables it to predict long-term dependencies (Goodfellow et al., 2016, Haykin et al., 2009, LeCun et al., 2015). Over time LSTM has changed, and today often also makes use of a component called forget gates to enable longer sequences of input over a longer period (Gers et al., 1999).
2.5 Transfer learning

Transfer learning refers to training on a task and then using that experience to apply a head start on a new but similar task called target task. This is today being used to solve two core problems that are limiting the possibilities of applying artificial intelligence into everyday life. As mentioned before, ANN scale with more data but the reality is that for many problems a large amount of data is hard or even impossible to acquire and even if it was possible it requires a lot of work to have the data ready for use. This is where transfer learning can be used by using similar existing data set to train and then use either the knowledge base or just a part of it, to enable better predictions on the data set for which the network was intended. The other core problem is that training with a large amount of data takes a long time, sometimes weeks which make it unfeasible for certain projects to use. A lot of tasks have a lot in common e.g. identifying a cat or a dog on an image. These tasks can, therefore, make use of the same classifier. Classifiers can, therefore, be trained with a large amount of data for a long time and then be reused to perform similar tasks. The layers in the classifier can be modified to fit the task e.g. include or remove potential outcomes from the output layer. The classifier is then trained on the target data set to fit the data that its meant to predict. According to Pan and Yang (2010), three main questions need to be answered; what to transfer, how to transfer, and when to transfer. What to transfer refers in this case to what part to use of a trained neural network. As previously explained for CNN’s, the first layers are used to recognize low-level features which in later layers are combined to high-level features, and at the end an object. When using transfer learning it’s important, to get a positive effect, to consider the similarities of the features, between the two data sets. This is done to decide which feature maps that will likely be similar enough to help the predictions on the targeted data set (Pan and Yang 2010). How to transfer refers to how the method of transfer learning and how it should be performed. When to transfer refers to when transfer learning should be used by analyzing if it will yield better results when performing a certain task. Rosenstein et al. (2005) writes that, if data sets differ too much then transfer learning can have the opposite effect and learning to predict correct outcomes becomes worse as it will be harder to train and therefore need even more data. Yosinski et al. (2014) wanted to test where the
general features of an ANN exist. He showed that the more the data sets differ, the worse performance boost was given for the target task when not replacing layers that contain high-level features. It could also lead to negative results. He, however, also showed that pre-training with a non-similar data set to the target data set still can be used to produce better results, by keeping layers with low-level features, than using random weights.

2.5.1 Street View House Numbers Dataset

The Street View House Numbers data set (SVHN) \cite{Goodfellow:2013} is a data set with images of house numbers where each image contains between one to six digits each, with examples shown in Figure 2.5. The data set contain ten classes, one for each digit. SVHN was created with the Google Street View tool and contains over 600 000 images. These images exist in two different formats where the first format contain the original images with the images varying in size and the second format where the images have been re-sized to 32x32 pixels and cropped to have one digit of the sequence as the number to predict and the other numbers used as noise. In this thesis, the first format was used due to the similarities of ID tags used to identify a farm animals, where both make use of the sequence of numbers to identify an object.

(a) Image number 2 with house number 23  
(b) Image number 79 with house number 1922  

Figure 2.5: Examples from the SVHN data set. The images differ a lot in the number of digits but also how they are presented.

2.6 Data augmentation

A major limitation to train a neural network is not having enough data, which limits the potential size of an ANN that can be used and how much a network can be trained as small data sets easily result in overfitting \cite{Srivastava:2014}. \cite{Krizhevsky:2012} writes "The easiest and most common method to reduce overfitting on image data is to artificially enlarge the data set using label-preserving transformations.". Data augmentation is when using the original data set to create a temporary larger data set, which originated from the original data set. Multiple copies of an image are made, which include a modification e.g. rotations, shifts, change in colors. An example of this can be seen in Figure 2.6. This enables an ANN to train on different variations of the same image which enables it to learn instead of memorizing. \cite{Mikolajczyk:2018}. 
Figure 2.6: Examples of an image used to create multiple augmented images. Data augmentation was not used on the SVHN data set when training the ANN.

2.7 Microsoft Azure

Microsoft (2019) provides Software as a Service (SaaS). SaaS provides programs to run on a cloud, which means that they are accessible through the internet with the use of e.g. a web browser, application, or email. There exist many reasons to why this is needed, e.g. the consumer cannot control the underlying infrastructure or operating system, need more storage, or just want to outsource hardware or software (Mell et al., 2011). SaaS enables effective use of hardware due to multiple people making use of the same hardware which would otherwise be in the idle majority of the time. This also makes it possible for dynamic power on the hardware depending on what task that is needed to be executed e.g. training a neural network or just playing an old game that would need different GPU power. Microsoft Azure (Azure) also provides an easy to use machine learning, computer vision implementation which can predict objects and sequence in an image. The computer vision Application Programming Interface (API) has been trained to give sufficient results for a lot of different images and tasks like detecting objects, describe an image and generate a thumbnail and many more. The classifier used in this study is referred to as Read text in images by Microsoft (2019).
2.8 Related Work

The method of combining RNN and CNN (RCNN), example model is shown in Figure 2.8, have been used with different architectures to identify different kinds of character sequences on an image. Naz et al. [2017] made use of an RCNN, where the recurrent layer was a multidimensional LSTM, to recognize Urdu characters, which are used in India, on an image. They made use of 44 different classes and received a 98.12% complete sequence accuracy. Rawls et al. [2017] used an RCNN, with an LSTM as a recurrent layer, to recognize sequences in both English and Arabic, written by humans and a machine print. They used two different measurements which were Word Error Rate, which compares predicted sequence to actual sequence, and Character Error Rate which compares predicted words to the actual words. Both are derived with edit distance (Levenshtein, 1966). Edit distance is used to determine the number of operations needed for a sequence to equal another sequence, with examples shown in Figure 2.7. Dutta et al. [2018]; Dai Nguyen et al. [2016] both used an RCNN, with a bidirectional LSTM as a recurrent layer, to predict hand-written words and math symbols. Quang and Xie [2016] even used an RCNN, with an LSTM as a recurrent layer, to predict DNA-sequences.

Figure 2.7: Demonstrate how edit distance is calculated. Three operations exist which are insert, substitute, and delete. For ISSA to equal the string ASSAY, a substitution, and an insert is needed. For 02315 to equal the number sequence 0281, a substitution and delete have to be applied. For both cases, two different operations are needed which means that edit distance is equal to two for both cases.

Cheang et al. [2017] examined if it was possible to combine CNN and RNN to recognize license plate characters. They tested two different models, with and without a recurrent layer as well as with and without data augmentation. The data set contained 2713 labeled images which contained between four to eight characters each. A modified VGG-16 (Simonyan and Zisserman, 2014), which is a 16 layered network, was used as a model with a recurrent layer before the last layer. The image was first passed through the CNN part to extract features from the image and then passed through the recurrent layer to predict the sequence of characters. To measure the performance of the different models, three different measurements were used which were perfect prediction, edit distance, and an average ratio which was used to compare sequence length of the predicted and actual sequence. According to the study, a recurrent layer vastly increased the performance of the model by achieving 76.53% perfect predictions, 0.74 average edit distance, and 0.94 average ratio. According to their conclusion, this method could be compared to solutions that are today being used in the industry while being easier to implement. They could also see that many of the wrong predictions came from characters that looked similar like e.g. M and N, and C and G. The network could, however, distinguish between D and
0, which are very similar, due to the context where D according to the format could not be a valid character. In all these studies the CNN was used as a feature extractor while the RNN was used for sequential predictions. According to these studies, CNN and RNN complement each other for tasks that involve sequence predictions from images.

No study could be found where image recognition has been used to recognize digits from ID tags that are used to identify farm animals. The most similar problem was from Cheang et al. (2017). When comparing the used ID tags in this study to the example images shown in their study, major differences exist e.g. the sequence lengths and the noise on the images. The context between the two problems differs as they are applied in environments with different conditions.

Figure 2.8: Demonstrates the combination of RNN and CNN, often called Recurrent Convolutional Neural Network. The CNN part is used to recognize important features. Before the output of a CNN is used, the dimension of the output needs to be reduced to fit a RNN. RNN uses this input to create multiple predictions. The size of the CNN or RNN can differ in layers and size of the layers.
3 | Problem

This section of the thesis explains the aim of this study, the motivation to why this is important, and the questions that need to be answered for the aim to be accomplished. Previous studies, described in section 2.8, were used to hypothesize which results that could be expected. How this study is conducted and how the different methods were implemented is also explained.

3.1 Aim

The aim of this study was to find a potential ANN solution that can recognize multiple digits of a number sequence from an ID tag, used by the farming industry to identify an animal, on an image.

A RCNN model, called PPNet which refers to the author of this thesis Pontus Pettersson, was created to test a potential solution. To test PPNet, a small data set of 350 images, called Animal Identification Tag (AIDT) was created due to there being no available data set. Two different sets of classifiers were created with PPNet. This was done to test if this method of transfer learning could be used to get a feasible result without the need of creating a bigger data set, which takes a lot of time. To provide CGI Skara with information if this solution is worth investing in, a comparison of results was done between the two training approaches of PPNet and Azure.

3.2 Motivation

A lot of different jobs include a process where a physical object needs to be read to identify something or someone. An example of this is when an animal is being transported and this event needs to be documented. This process is done manually by first retrieving the ID number from the ID tag. The information about which animal and what events that have occurred, then needs to be inserted into a database. Due to the magnitude of the total of animals transportation’s every day, this process becomes a time-consuming task. The motivation to this study is that, by automating and expediting this process with the use of image recognition, possibly reduce cost, time, and menial tasks. Another benefit is reducing human errors that could occur. Alternative methods that could be used to expedite the identification of an animal can be e.g. bar code which is used in stores to identify a product or an ID chip which often is used to identify pets. For these methods to be used in farms, the presentation and retrieval of information would have to be changed for the industry. The solution presented in this study still makes use of common practices and only gives an alternative method to the retrieval of information and in return expedite the process. The change of behavior is however not forced as the currently used methods are still available.
3.3 Research Questions

To accomplish the aim of this study, three different questions had to be answered.

RQ1: To what extent can PPNet identify a number sequence on an ID tag from the AIDT data set?

RQ2: How does PPNet compare, in identifying a number sequence on an ID tag from the AIDT data set, to Azure?

RQ3: To what extent can pre-training PPNet with the SVHN data set help to identify a number sequence on an ID tag from the AIDT data set?

3.4 Hypothesis

- **H1**: PPNet will achieve a minimum of 45.0% character level accuracy on average from the AIDT data set.

According to CGI Skara where this study was conducted, identifying five out of eleven digits from the number sequence, regardless of the correct position, would with high probability be enough to uniquely identify an animal that has been prepared for transportation in a particular farm. This corresponds to 45.0% CLA which previous studies have shown to be achievable in similar areas by using a RCNN. PPNet and the AIDT data set were both smaller in size, then the ones used in related work, however, they managed to achieve far better than 45.0% CLA. This indicates that 45.0% CLA could be achievable by PPNet.

- **H2**: PPNet will achieve higher performance on the AIDT data set than Azure.

Azure is a general machine vision API. It has been trained on a lot of different data sets with different sequences and characters. The PPNet, however, will be trained to specifically predict digits from ID tags, almost identical to the environment of the AIDT test set, but with different number sequences. PPNet should, therefore, be able to achieve higher performance.

- **H3**: Pre-training PPNet with SVHN will make it receive higher character level accuracy on the AIDT dataset.

A lot of similarities between SVHN and AIDT exists e.g. sequences of digits and only digits in the sequences. It also exists a big difference which is the sequence length. SVHN should, however, be able to be used in a transfer learning process to successfully help the CNN better identify digit features due to both data sets having sequences of digits. No previous study could, however, be found that presents a viable method for using transfer learning for a RCNN model. The belief is that if CNN can better recognize features, then it will also help the RNN make better predictions.
3.5 Objectives

<table>
<thead>
<tr>
<th>Nr</th>
<th>Description</th>
<th>Contributors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Research the problem in the given area through literature papers</td>
<td>Issa, Pontus</td>
</tr>
<tr>
<td>2</td>
<td>Create a data set which includes images with identification tags</td>
<td>Issa</td>
</tr>
<tr>
<td>3</td>
<td>Find similar data set to AIDT to gather more training data</td>
<td>Issa</td>
</tr>
<tr>
<td>4</td>
<td>Pre-process images from the created data set</td>
<td>Pontus</td>
</tr>
<tr>
<td>5</td>
<td>Build model PPNet with RCNN</td>
<td>Pontus</td>
</tr>
<tr>
<td>6</td>
<td>Create three classifiers of PPNet which are first trained on SVHN, then trained on AIDT and then tested on the AIDT test set (PPNet1)</td>
<td>Issa</td>
</tr>
<tr>
<td>7</td>
<td>Create three additional classifiers of PPNet which are directly trained on the AIDT training set and then tested on the AIDT test set (PPNet2)</td>
<td>Pontus</td>
</tr>
<tr>
<td>8</td>
<td>Test Azure on the AIDT test set</td>
<td>Issa, Pontus</td>
</tr>
<tr>
<td>9</td>
<td>Compare the results from PPNet1, PPNet2, and Azure</td>
<td>Issa, Pontus</td>
</tr>
</tbody>
</table>

3.6 Method

This section is used to describe the different approaches taken for the study to be completed. It includes a description of what a case study is, why it was chosen as the main method and its limitations. It also describes how the AIDT data set was created, how PPNet was trained and tested, and how the classifiers were compared.

Case study

To test to what extent, an ANN could be used to predict digits in a number sequence a case study was conducted. A case study can be used to analyze how a certain method or algorithm would handle a particular case (Wohlin et al., 2012). In this study, the PPNet model was the algorithm while the ID tags with an actual ID number sequence were the case which in real-life is used to identify animals in farms. To test the PPNet solution, real animal ID tags were first needed to be collected and labeled, to create the AIDT data set. This was then used for training and testing. The results from the tests were then used to determine if this particular algorithm could be used to solve this problem in this specific context. Case studies can therefore often be hard to generalize due to the specific case that is being tested.

3.6.1 Alternative methods

According to the definitions of case study and experiment written by Wohlin et al. (2012), both have a lot of similarities, where he also writes:

> If we, for example, would like to compare two methods, the study may be defined as a case study or an experiment, depending on the scale of the evaluation, the ability to isolate factors and feasibility for randomization.

(Wohlin et al., 2012, p. 14)
An experiment could be a viable option. However, the comparison between PPNet1, PPNet2, and Azure was not sufficiently controlled in such a way that an experiment was possible. To create an experiment, a higher level of control is required, e.g. change a specific variable and measure how this change affects the results. The aim of this thesis was to try to find an ANN solution for a real-life situation and according to Wohlin et al. (2012), a case study is often done in a typical situation.

A literature study could have been used to find a potential ANN solution by making an in-depth examination of the different ANN implementations to similar problems. This would however only result in a discussion for solutions that might be used. By doing a case study, evidence could be provided that a particular method can be used at least in this context which is more concrete.

3.6.2 Data Collection (Objective 2)

Data for the case study was created with the help of CGI Skara who provided 350 physical identification tags that have previously been used to identify farm animals. The retrieval of the number sequence from the ID tag is done differently on different farms where some first remove the tag from the animal and others who leave it on. For this study, the ID tags were removed from the animal and placed in front of a transparent table. This method was chosen due to the complications of accessing someone else’s property and the time that was going to be invested to create the data set images in a real-life setting where the ID tag is still on an animal. Images were taken with the camera LEICA VARIO-SUMMILUX-H1:1.6-2.4/27-80 ASPH, from a Huawei P20 Pro, right above the ID tags. A resolution of [18:9] 7MP, which corresponds to a width of 3648 and length of 1824 pixels, was used to take each image. This was the lowest resolution on the phone which was used to reduce the size of each image, as the size was already too big to fit PPNet as it would need an input size of over six million parameters.

3.6.3 Preparations for PPNet (Objective 5)

The reason for choosing a RCNN for this study was due to its frequent usage in similar areas. It has for different data sets achieved state-of-the-art performance for recognizing a sequence in an image, which is described in section 2.8. For this study, RCNN was used to output a number sequence, trying to predict the actual number sequence of an ID tag from an image of the AIDT data set. The CNN, which was the first part of PPNet, receives an image as an input. The purpose of the CNN was to create a new representation of the input image, which recognizes relevant digit-features. This would be used as an input for the RNN which consisted of LSTM layers. The RNN was used to produce eleven predictions of the input where the RNN has learned to specify which features that are relevant for each prediction.

To build PPNet, the programming language Python was used together with the library Keras. Keras is a high-level API that is used to implement deep learning models. It can run on top of either Tensorflow (Abadi et al., 2016), CNTK (Seide and Agarwal, 2016), or Theano (Bergstra et al., 2010), which are low level deep learning libraries. PPNet used Tensorflow to build the network which is the default setting. Keras was selected due to its simplicity for building, training, and evaluating an ANN model.
When training an ANN with Keras, an average loss and accuracy are calculated throughout each epoch (Chollet et al., 2015). The accuracy is used to determine how many of the predicted digits that were correctly predicted both positions in the sequence and class. This can be used to determine if the ANN is learning to more accurately predict the training data, which it has been shown multiple times. To speed up the matrix multiplications to train an ANN, the GPU was used as it can do calculations in parallel (Nguyen et al., 2019).

### 3.6.4 Preparations for transfer learning (Objective 3, 6)

The SVHN data set was found and chosen as the data set to use for transfer learning. The reason why it was chosen was due to the similar features that exist in each image of both SVHN and AIDT, where both contain a sequence of digits. There exist, however, also differences between them both which caused problems and had to be solved for SVHN to potentially be used successfully. The first problem with using the SVHN data set was the dynamic sequence length as some had a sequence length of five while others only had one. This was solved by deciding a fixed length for all labels. The class ‘END’ was added to the end of each label to fit the fixed length.

The second problem was the difference in sequence length between the data sets AIDT and SVHN. Two potential solutions existed for this problem. The first was to use a fixed length of the label length to match the sequence length of the AIDT data set which is eleven digits. Figure 3.1a would then have 1922 and seven consecutive 'END’s as its label and Figure 3.1b would have 3 with ten consecutive 'END’s. This meant that for a lot of images their labels would for the majority contain the class ‘END’. There was, therefore, a possibility that this would prevent other classes from being taught and the only thing that was getting predicted was ‘END’s. The second solution was to create a modification of PPNet better suited to train with the SVHN data set. Instead of PPNet predicting eleven digits, it now only made five predictions, which is the max number sequence length for the images used on SVHN. The fixed max sequence length of five would result in the examples shown in Figure 3.1. The problem with the second solution was that the modified PPNet no longer was suited to train on AIDT. This could, however, be solved by removing and initialize new LSTM layers to instead predict eleven digits again. This however corresponded to how transfer learning was meant to be used in this study, which was for the CNN to learn to recognize features. As only the LSTM layers needed to be removed the CNN could remain the same.
3.6.5 Training and Testing

The training was done on two different sets of classifiers of the PPNet model. The first was PPNet1 which is a set of three classifiers. PPNet1 was first trained on the SVHN data set and then trained on the AIDT training set. The second set was PPNet2 which also is a set of three classifiers of the PPNet model. PPNet2 was directly trained on the AIDT training set without any pre-training. The classifiers for both PPNet1 and PPNet2 were both initialized with random weight using seed numbers one, two, and three, for convenience. When PPNet1 and PPNet2 were finished training, they and Azure were tested on the AIDT test set to decide their performance. A flow of which data sets that were used for PPNet1, PPNet2, and Azure can be shown in Figure 3.2.

![Figure 3.2](image)

3.6.6 How to compare results

To determine the performance of the classifiers, two different measurements were taken into consideration. The first one, character level accuracy, was used to decide how big percentage of the predicted digits that also existed in the actual number sequence, regardless of the position in the sequence. This measurement alone could, however, lead to misleading conclusions e.g. thousand of random numbers could result in 100% CLA while not predicting the actual number sequence. The edit distance was, therefore, also used to determine the average number of changes that would be needed for the predicted sequence, to be identical to the actual sequence for each image. Examples of how edit distance is calculated can be shown in Figure 2.7. These two measurements have com-
monly been used when predicting different sequences which are shown in section 2.8.

Four different outcomes were possible to occur where first is the worst while the fourth is the best:

- High CLA and high edit distance: A lot of predictions that are correct but in the incorrect position.
- Low CLA and high edit distance: Very few predictions or same predicted digit over and over again which are mostly wrong.
- Low CLA and low edit distance: Not many predictions were made, but those who were, were probably correct.
- High CLA and low edit distance: Many correct predictions and the classifier has learned. This is the best outcome and the one which is aimed to achieve.
4 Implementation

The section describes how the data set was created, pre-processed and augmented. It also describes the architecture of PPNet, how it was pre-trained, and how transfer learning was done.

4.1 The Animal Identification Tag Data set (Objective 2)

The identification tags on the images, shown in Figure 4.1, contains two sections. The top section is used to describe the origin of an animal. It includes the country code which uses two letters to describe the country and the section ID which has six digits that are used to identify the farm in which the animal was born. The ID tags that were used were from Sweden, which meant that they all had the same country code SE. It was therefore decided that this wouldn’t be included in the predictions. The bottom section is used to identify the specific animal with an animal ID, which has four digits, and a control digit. The data set has ten classes one for each digit, with each image containing a total of eleven digits. The ID tags are not all the same as some are shorter in length than others, some were dirtier, and some contain different irrelevant characters, which are not used for identifying the animal. The data were labeled in a JSON-file where a reference to an image was made, and the number sequence for that image was stated. The creation of the data set was done manually by the authors of this thesis. Due to the time-span of this thesis, the different sections were not separated when creating the data set. The neural network would therefore not learn that different sections exist. The aim can still be satisfied by analyzing how a certain method can predict sequences in this context.

![Image of ID tags from the AIDT data set.](image)

**Figure 4.1:** Examples of ID tags from the AIDT data set. Due to confidentiality, some digits are masked.
4.2 Pre-process the data set (Objective 4)

To prepare the AIDT data set for training and testing it had to be divided and pre-processed to facilitate for the network to train adequately. The data set, which contained 350 images, was split into two parts. One part contained 80% of the images and was used as a training set, and the remaining 20% of the images were used as a test set. The Python library Pillow (Lundh et al., 1995-2016) was used to preserve the features of the image when re-sizing each image from 3648 x 1824 to 150 x 75 pixels. Each image was then normalized by dividing each pixel with 255 which would give it a value between zero and one. This would according to LeCun (1993), help expedite back-propagation.

4.3 Data augmentation for PPNet (Objective 5)

Keras offers a data augmentation function where the user specifies which augmentation settings to use. Each time an image is used for training PPNet, the augmentation function receives the original image and creates an augmented copy. The network will then train on the augmented image which becomes an image that PPNet has never seen. The augmentation settings that are used for AIDT have according to Mikołajczyk and Grote (2018) shown to be effective to help an ANN generalize. The augmentation used for the AIDT training set was:

- rotation_range = 45. Randomly rotate the image between 0 - 45 degrees.
- width_shift_range = 0.2. Randomly shift the image with a fraction of total width.
- height_shift_range = 0.2. Randomly shift the image with a fraction of total height.

These settings also match the variations that could occur when taking a picture of an ID tag in a real-life scenario. Some may take a picture with a slight rotation, shown in Figure 4.2b or shift the ID tags in different directions.

![Figure 4.2: Examples of ID tags from the AIDT data set which display the original image and the augmented image. Due to confidentiality, some digits are masked.](image)
4.4 The PPNet architecture (Objective 5)

The PPNet model, shown in Appendix A – Figure A.1, was created using a RCNN architecture to find digits in a number sequence from an image. The CNN layers were used to extract features from the images. PPNet first contained three convolutional layers, with the first two layers also using max pooling. The outputs from these three layers were normalized with the Keras function `keras.layers.BatchNormalization`. PPNet then continued with two fully connected layers that were used to reduce the input size for the two LSTM layers. This was done to reduce the number of neurons used in the proceeded two LSTM layers as they demanded a lot of memory. The LSTM layers were created to make twelve predictions, used to predict each digit in the image, and an 'END' prediction, which could look like 03123141231END. The 'END' was used to indicate that the ID sequence reached its end. These twelve predictions were sent to two proceeding fully connected layers which determined the class for each prediction. Every layer in the network used Leaky Relu as an activation function except for the last layer which used Softmax to calculate the probability distribution of potential outcomes. The twelve outputs from the LSTM were then used to make a prediction for each output and produce a sequence of digits corresponding to the number sequence on the image. To calculate the loss, the `keras.losses.categorical_crossentropy` was used. This architecture was not created or tested to perform the best possible result. It was created to get results that show if RCNN can be used to predict digits on the AIDT data set and still be tested within the given time-frame.

4.5 Transfer Learning (Objective 6)

When pre-training the three classifiers for PPNet1 the first 5000 images of SVHN were used. They were pre-processed with the same settings as the AIDT data set aside from the label length. PPNet1 was trained for 500 epochs without any data augmentation. Why few images and no data augmentation were used was due to time limitation as more images lead to longer epochs and data augmentation makes training slower. When finishing the training of SVHN the two LSTM and the last two fully connected layers were replaced with two new LSTM and two fully connected layers. The new layers were initialized with random values with random weights, using the same seed that was used to create them. The new layers were of the same types as the original PPNet to enable training on the AIDT data set. PPNet1 was then retrained on the AIDT training set for 100 epochs with the same data augmentation settings as PPNet2.
5 | Results

This chapter will present all the acquired results of the study. This chapter is divided into two parts where the first part presents all the results from the different objectives, and the later part analyzes all the results.

5.1 Presentation

The presentation of the result is divided into three different parts, where every part is connected to an objective. Every part will present its given result from the corresponding objective.

5.1.1 Present result of PPNet1 (Objective 6)

The aim of objective 6 was to train three classifiers, first on the SVHN data set and then on the AIDT data set. This was done to test a transfer learning method and see if it can be used to help the PPNet model generalize better. The three classifiers of PPNet1 managed to achieve an average of 61.8% CLA which corresponds to 6.8 digits per image that were classified correctly. The average edit distance achieved by PPNet1 was 6.5 which means that less than five digits for each number sequence, were on the correct position.

5.1.2 Present result of PPNet2 (Objective 7)

The aim of objective 7 was to train three additional classifiers of PPNet; trained and tested on the AIDT data set. The result of PPNet2 which is presented in this section and Table 5.1 shows the average of PPNet2. PPNet2 achieved 60.8% CLA which corresponds to an average of 6.7 out of eleven digits being correctly classified. For the edit distance, PPNet2 achieved an averaged of 6.3 numbers of instructions per image, needed for the predicted sequence to be identical to the actual sequence.

5.1.3 Present result of Azure (Objective 8)

The aim of objective 8 was to test how Azure performed on the AIDT test set. The result from Azure, shown in Table 5.1 displays the values for the different measurements. Azure achieved a CLA of 63.6%. This corresponds to a 7.0 correctly predicted digits per image out of eleven numbers. The edit distance for Azure was 4.6 instructions per image.

5.2 Analysis

This chapter is used to analyze the results gathered from PPNet1, PPNet2, and Azure. They will first be analyzed separately to its corresponding objective before they are compared against each other.
5.2.1 Evaluate result of PPNet1 (Objective 6)

After the training had been done on the SVHN training set, each classifier of PPNet1 had, at 500 epochs, managed to achieve close to 100% accuracy on the training set. This means that each prediction was correctly predicted. A test was done on a SVHN test set which contained 100 unseen images from the SVHN data set. This was done to test if PPNet1 had learned to recognize and predict number sequences from the SVHN data set. For these 100 images, the classifiers of PPNet1 which only had been trained on SVHN managed to achieve a 48.0% CLA and an edit distance of 1.4, for each image. For classifier with seed one, 20 of these images, no correct predictions were made while 21 out of 100 images were perfectly predicted. This indicates that PPNet1 had learned to better predict unseen images from the SVHN data set but would need to train on more data or use data augmentation to avoid overfitting.

After one training epoch on the AIDT training set, a loss of 2.4 and 14% accuracy was achieved. This is slightly better than randomly predicting digits, which would result in 9% accuracy, calculated with $\frac{1}{11}$, due to eleven classes. The learning process can be seen in Figure 5.1 where accuracy goes from 14.0% to 60.0% and loss starts at 2.4 and ends at 1.2. It can also be seen in the graphs that the accuracy was still improving and loss still decreasing. More training could be used to get a better result on the AIDT training set but this does not, however, assure that it would generalize more and better predict unseen data.

Concerning hypothesis H1, the minimum expected and accepted CLA was 45.0% and is shown to be supported due to the achieved CLA of 61.8% for PPNet1. All classifiers for PPNet1, however, seemed to have learned to always predict eleven digits. This made some predictions seem as if they were random due to the images being very dirty and the predicted number sequence not once being in the correct position. Four out of the 70 images used for testing, started with a zero. Only once for all three classifiers used for PPNet1, did a prediction not start with a zero. In that case, the actual sequences did start with zero while the first classifier of PPNet1 predicted a one. Both first and second classifiers for PPNet1 always predicted a zero, three, or four as the second prediction. The third classifier had more variations on the second digit but still, the majority predictions for the second digit were a zero, three, or four. Due to a lot of images of the AIDT data set coming from the same farm the first five digits of the sequences are similar.
5.2.2 Evaluate result of PPNet2 (Objective 7)

The PPNet2 model achieved a better result than expected. According to the hypothesis H1, the accepted level of CLA was 45.0%. The result for PPNet2, shown in Table 5.1, displayed a CLA of 60.8% which was above the accepted level of 45.0%. This result indicates that the PPNet2 could achieve more than 45% and therefore has support for H1. After one epoch of training on the AIDT training set with PPNet2, it had reached an accuracy of 24.0% and a loss of 2.3, which is shown in Figure 5.2. After 100 epochs it reached an accuracy of 70.0% and a loss of 0.9. During training, the accuracy increased and the loss decreased for every epoch. It is, however, difficult to determine how many epochs that are needed to help PPNet2 generalize better. Figure 5.2 indicates that with more epochs it would probably increase its accuracy and decrease its loss on the training set.

All the images of the AIDT data set had a sequence which was eleven digits long. The majority of the images in the AIDT data set also began the number sequence with a zero. PPNet2 seems to have learned this pattern as it would always predict eleven digits and always start with a zero. Some ID tags were barely human-readable and the digits for these predictions were rarely correct. This indicated that these predictions were made based on previously learned patterns. Not all the number sequences started with a zero even though PPNet2 always would start the predictions with a zero. PPNet2 also had some digits that often got confused with each other like three and eight as well as zero and eight. Other digits that also often got confused was five and seven. The reason for this could be that these features in combination with dirt from the ID tags, easily can look to have similar features e.g. a zero but with dirt in the middle of the number can make it look like an eight.

Regarding research question RQ1, which asks to what extent PPNet can identify a number sequence on an ID tag, is considered answered. This is due to the displayed result with an average of 6.7 correctly predicted digits in a sequence, out of eleven.
5.2.3 Evaluate result of Azure (Objective 8)

Azures result, shown in Table 5.1, got a CLA of 63.6% which corresponds to 7.0 characters per image. A 4.6 edit distance means that more than 50.0% of the predicted digits were also in the correct place of the sequence. Azure makes use of dynamic prediction which means that the length of the predictions can vary. In this test, the prediction lengths varied from zero to fifteen. When no predictions were made, the numbers on the image were visible but included a lot of dirt which could have affected Azure when determining features. When too many predictions were made by Azure, the ID tags had digits in other spaces aside from the ID number sequence, which were not a part of the identification sequence. A possible cause of the varying length can be seen in Figure 5.3. Azure confused the numbers five and seven in the AIDT test set multiple times.

**Figure 5.2:** Demonstrate a graph of accuracy and loss during training which ran for 100 epochs on PPNet2 with seed 1, on the AIDT training set.

**Figure 5.3:** Demonstrates an example of a tag that includes other numbers unrelated to the identification sequence. Above the "SJV" sequence, the sequence "2000" is included vertically.
5.2.4 Compare results from PPNet1, PPNet2 and Azure (Objective 9)

Table 5.1: Results for the different classifiers

<table>
<thead>
<tr>
<th>Name</th>
<th>CLA (Avg %)</th>
<th>Digits per image (Avg)</th>
<th>Edit Distance (Avg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPNet1</td>
<td>61.8</td>
<td>6.8</td>
<td>6.5</td>
</tr>
<tr>
<td>PPNet2</td>
<td>60.8</td>
<td>6.7</td>
<td>6.3</td>
</tr>
<tr>
<td>Azure</td>
<td>63.6</td>
<td>7.0</td>
<td>4.6</td>
</tr>
</tbody>
</table>

The results, shown in Table 5.1, for PPNet1 and PPNet2 showed similar results with minor differences. PPNet1 managed to achieve 1.0% better CLA than PPNet2, which corresponds to identifying 0.1 more digits per image independent of order. For edit distance, PPNet2 managed to achieve 0.2 fewer instructions than PPNet1, which approximately corresponds to one more correctly predicted digit, for every four images. In short, PPNet1 achieved better CLA while PPNet2 achieved better edit distance. Due to the small sample size for each set of classifiers and the restrictions which this induces to perform a conclusive statistical analysis, a possible difference between PPNet1 and PPNet2 cannot be determined. Regarding research question RQ3, which asks to what extent pre-training PPNet with SVHN can help it identify digits from the AIDT data set, is considered answered. Even if a difference existed, according to these results would be minor but not enough for the used method of transfer learning, to be sufficient considering the additional time that was needed for pre-training. The results could, therefore, not show any support for hypothesis H3.

Azure managed to achieve 1.8% and 2.8% better CLA than PPNet1 and PPNet2 respectively. These results correspond to at least 0.2 more digits per image which approximately corresponds to identifying one more digit for every five images independent of the order of that digit. The major result difference is edit distance where Azure managed to achieve 1.9 for PPNet1 and 1.7 for PPNet2, fewer instructions for each image. One major factor to the edit distance is that Azure seems to only predict based on the features which it can recognize while both PPNet1 and PPNet2 learn to always predict eleven digits. A reason for this is the common number sequence length of eleven digits for every image on the AIDT data set. This means that even if some features are not recognizable it will still try to predict eleven digits based on previously recognized pattern and not features, as this has higher chance to predict more correctly, than not predicting at all. Regarding research question RQ2, which asks how PPNet compares in identifying a number sequence on an ID to Azure, is considered answered. The results indicate that PPNet, while achieving competitive results, still didn’t manage to perform as well as Azure. Support for hypothesis H2 could, therefore, not be shown.
6 | Discussion

This chapter begins with a summary of this thesis and the validity threats that have existed throughout the study. A discussion is done on the created data set, PPNet model, and the gathered results. These are compared to each other and to previous works, which are then used to draw different conclusions. The last sections are used to discuss the relevant social aspects that need to be considered and end with future work that can be done within this field.

6.1 Summary

The aim of this study was to find a potential ANN solution that recognizes digits from a number sequence on an ID tag, used to identify animals in farms. This was done with image recognition which today is being used to perform a lot of different tasks e.g. self-driving cars and identifying objects in an image. The motivation for this study was to find a method to expedite the currently used process for identifying an animal, where the number sequence is retrieved and documented manually which adds up to a lot of time due to the number of times this occurs. Similar problems have been solved with the use of an RCNN where a high complete sequence accuracy and CLA have been achieved. A CLA of 45.0% was together with CGI Skara decided to be the minimum and acceptable result for the solution as this is probably enough for a particular farm animal to be identified.

In this thesis, a case study was used to accomplish the aim where a model called PPNet, and data set called AIDT was created. The AIDT data set contained 350 images one for each ID tag received by CGI Skara. The data set was split into two sets, training set with 280 images and test set with 70 images. For the PPNet model, six different classifiers were created and divided into two sets PPNet1 and PPNet2 which contained three classifiers each. PPNet2 was directly trained on the AIDT data set while PPNet1 first trained on a data set called SVHN and then trained on the AIDT data set. The transfer learning method was meant to be used to counteract the small number of images on the AIDT data set which have shown to limit the learning of an ANN. PPNet1 and PPNet2 were then compared to the general image recognition API from Microsoft Azure. Another method used to counteract a small data set was data augmentation which was used when all classifiers of the PPNet model were trained on AIDT.

When PPNet1 and PPNet2 were finished training, they and Azure, were tested on the AIDT test set. For testing, two different measurements were used. The first one was CLA which refers to character level accuracy and the second one was edit distance. Edit distance was used to compare how much a predicted sequence, predicted by a classifier, differed to the actual sequence from the ID tag on an image.

The acquired results showed that PPNet1 managed to achieve an average CLA of 61.8% and averaged 6.5 in edit distance. PPNet2 achieved a CLA of 60.8% and averaged 6.3 in edit distance. This indicates that PPNet can achieve well over 45.0% CLA which was hypothesized. The performance of both PPNet1 and PPNet2 were very similar which indi-
cates that pre-training the network had no positive effect which was not expected. Both PPNet1 and PPNet2 did receive worse results than Azure in CLA, where it achieved 63.6%, and in edit distance, where it achieved 4.6 instructions.

6.2 Validity threats

To contribute with a study, a degree of validity must be satisfied to be accepted in the research field and CGI Skara where this thesis was written. According to Wohlin et al. (2012) four types of threats exist which are Conclusion, Internal, Construct, and External. These four threats are described and explained how they were counteracted.

6.2.1 Conclusion Validity

Conclusion validity refers to the possibility to draw the correct conclusion, and if the conclusion can be made with the method of choice.

Fishing and the error rate

This threat is divided into two parts fishing for a result and error rate which refers to a type 1 error that has been made. Fishing refers to when the analyzer is looking for a specific result or outcome due to bias or personal gain which creates false-negatives. This is a threat because the analyzer is the person doing the research which may influence the result. Examples of when fishing often occurs are when the research questions are formulated after the results, another would be when picking outliers on tests to present better results. Studies, where fishing has occurred, are harder to replicate as removing fishing will probably lead to a different result (Parker and Szymanski, 1992).

This threat has been counteracted with three different methods. First, the research questions and hypotheses were written before the test began, and all classifiers, which were trained, were also evaluated. The second was that when creating classifiers, three arbitrary seeds were chosen, without knowing how the seeds would affect the performance. The third was that the measurements that have been used are independent and not affected by external factors other than the performance of PPNet.

Reliability of measures

This threat refers to how reliable the measurements by considering their objectivity. Objective measures can often also be repeated and return the same value. Due to them being replaceable also makes them easier to generalize as the result always remains the same.

The measurements used in this study, to measure the performance, was CLA and edit distance. Both of these measurements are objective and will always return the same result e.g. if an identical classifier to PPNet1 seed 1, receive an AIDT image then they both will also produce the same output.
6.2.2 Internal Validity

This refers to internal variables that could change over time or different circumstances without the researches knowledge.

**Testing**

Wohlin et al. (2012) writes "If the test is repeated, the subjects may respond differently at different times since they know how the test is conducted.". Due to the limited knowledge of how often Azure is updated and how it trains there is a possibility that if the test is done again with Azure in the future, it may result in a different result. There is a possibility that the imported images from the tests can be used to train the network.

This was counteracted by addressing when the tests were executed. There is, however, no guarantee that previous versions of Azure can be retrieved in the future.

6.2.3 Construct Validity

The validity for construct refers to the relationship between the received result and the theory that has been explained and was meant to be tested. This can be affected by both the design of the test as well as social factors.

**Inadequate preoperational explication of constructs**

This threat refers to how well defined the different constructs or components of a test are. How the test is executed and which measures must be sufficiently clear. When stating that a model has better accuracy it should be clear what better refers to. If results are being stated without being defined, it could lead to misinterpretations for the readers. One word can mean different things in different circumstances e.g. accuracy of what is being measured.

This threat is handled by defining, in section 3.6, what methods are being used to gather the results. Section 3.6.6 is used to state why each measurement is used, what they are, and how they are constructed.

**Mono-method bias**

Using a single type of measure may include a measurement bias due to the lack of context. This could be counteracted by involving different measurements that complement each other and clarifies the misinterpretations that could occur with only one measurement. An example of this in this study would be if a classifier predicted all digits but in the wrong order and then only present that all digits were predicted and the classifier had 100% accuracy.

To counteract this threat, two different measurements were used which were CLA and edit distance which is described in section 3.6.6.
6.2.4 External Validity

External validity refers to the different conditions that could restrict the conclusions of the study to generalize and be applied to its industrial usage area. According to Wohlin et al. (2012) "We can only generalize the result to environments that are similar to the experimental setting."

**Interaction of setting and treatment**

This threat refers to the tested settings and treatments of the experiment not reflecting how it works in the industry. This can be done if using old tools are out-dated or the environment where the experiment is taking place does not resemble where it will or can be used.

This threat was handled by using real ID tags to create data, which have previously been used in farms to identify an animal.

6.3 Conclusion

6.3.1 Conclusions taken from the AIDT data set

The AIDT data set was created by ID tags received by CGI Skara. These ID tags were further received by a farmer. This resulted in many of the ID tags having similar numbers sequences e.g. the first digit of the sequence most often is a zero, the second digit most often is a three, four or eight. This minimizes the variance on the training set and limits generalization. Instead of the classifier learning to recognize features, it may instead learn patterns in the training set. This also affects the differences between the training set and the test set. If the number sequences on the test set are similar to the training set then this would give a result that indicates that the classifier is good at predicting the digits. The classifier has instead learned the different patterns and is instead making predictions based on this, instead of the features. An example of this is if a lot of images on the training set starting with the sequence 031325. Instead of predicting the digits, it would instead learn to always guess the first digits as 031325. A k-fold cross-validation (Kohavi et al. 1995) could be used, where k classifiers are created. Each classifier is then trained and tested on different parts of the data set, compared to the other classifiers. This can be used to minimize bias that occurs if a training set is too similar to the test set, which is the case when a classifier learns patterns and gets a result that indicates good performance.

Another lack of variance in the data set is the sequence lengths which always are eleven digits. This lead to the classifier learning to always predict eleven digits even though many ID tags having digits that were hard to recognize. This was seen in some cases, where Azure didn’t recognize any digits while all PPNet classifiers made eleven predictions. For some images, these predictions seemed to be random. Applying these improvements to the AIDT data set would most likely make significant improvements with helping PPNet to better predict digits from the ID tags.
6.3.2 Conclusions taken from PPNet

The authors interpretation of the result received by the test of PPNet is that the RCNN method can be used to at least partly automate this process and expedite the identification of animals in farms. There are, however, obstacles and further development needed with the used model in this study, which needs to be handled before this specific model can be used. The interpretation of the results is that the received results indicate that PPNet can predict digits from an ID tag. It does however always predict eleven digits which make it hard to determine the correctly predicted digits. This can be due to low variance in the training set which was previously described. Different methods can, however, also be applied to the model which counteract this behavior. One of these is to use a threshold that makes PPNet only output if a certainty for a prediction is met.

Small differences have previously shown to have a big impact on the performance of a classifier. This can be seen with AlexNet [Krizhevsky et al., 2012] and ZFNet [Zeiler and Fergus, 2014], which is a modified version of AlexNet, where small changes had a big impact on the performance. In this study, the majority of the time went to learn the functionality of ANN and then RCNN to create a model that can learn to predict sequential data. Due to the limited time and knowledge on how to design an architecture that can perform this task, made it difficult to experiment with different architectures. This most likely affected the performance of PPNet, negatively.

The comparison to Azure showed that both PPNet1 and PPNet2 performed worse in both CLA and edit distance. There exist, however, big differences between them. One big difference is the amount of training and training data that have been used. The AIDT data set was very small and therefore limited how much the network could generalize. Both graphs in Figure 5.1 for PPNet1, and Figure 5.2 for PPNet2 showed that classifiers were still learning to perform better on the training data. It can not be dismissed that more training wouldn’t have a positive effect on the test set as well. Even though Azure itself didn’t train on the AIDT data set it has most likely been trained on a lot of different but similar data sets. The ANN architecture used for Azure is also probably a lot bigger than the one used for PPNet. The authors of this study had, however, no insight into how the Azure network was built or trained. The results received in this study show that Azure is preferable over PPNet and also requires less work due to Azure already being trained.

6.3.3 Conclusions taken from the transfer learning

An attempt to solve the small data set issue was done, which was pre-training three classifiers on the SVHN data set before training on the AIDT data set. The result indicated no difference in the end. The idea is to get a head start on the task you actually want to perform. This was, however, not what was seen in this case as when transferring from SVHN to AIDT as the starting point was worse than starting with random weights. One possible explanation would be the overfitting that occurred to the classifiers when training on the SVHN data set. If this is the case than applying more images to the SVHN data set can help PPNet achieve a better result without adding more images to the AIDT data set. Another reason why the transfer learning method didn’t work was due to the few layers that were reset. The difference between the data set could be larger than anticipated and the layers with high-level features didn’t, therefore, fit both data sets. This could be solved by removing more layers. The last considered reason was that the limiting factor
was the model itself. Due to the used hardware, the size of the LSTM had to be reduced as they caused out of memory issues while training. Both PPNet1 and PPNet2 achieved similar results. A possible reason for this is that the pre-trained CNN of PPNet1 didn’t help the LSTM layers and therefore was the limiting factor to why better results weren’t achieved.

### 6.3.4 Comparison to previous work

Results found in this thesis followed the same pattern as the other theses mentioned in section 2.8, where a RCNN was used to recognize different kinds of sequential data, from an image. The result, however, didn’t reach the same performance level as they demonstrated. There exist some differences between this study and those mentioned in section 2.8. The environmental differences are that these ID tags are used in dirtier conditions. This will include noise when trying to predict the different digits as some will be less visible. Some digits may even appear as a different digit when including dirt in different parts. The main difference, however, was the size of the data sets. The AIDT data set makes up 12% of the size of the data set used by Cheang et al. (2017), and less than 1% of most of the studies referred to in section 2.8. Another explanation, to why the performance differed from the other studies is the use of architecture. Cheang et al. (2017) used a VGG-16 CNN to which they added a RNN. VGG-16 is a commonly used architecture after its performance in ILSVRC 2014, which is a competition where different architectures compete to achieve the best results. The PPNet architecture had to be downsized due to the limited time and performance of the hardware. It is hard to make a valid comparison between the performance to these studies, due to the big differences that exist in data sets, different methods that were used, and architecture. They do, however, indicate that better results could be achieved by applying further improvements which are described in section 6.3.1 and 6.3.2.

Yosinski et al. (2014) showed and argued, that the more the data sets differ the less positive effect will occur when using transfer learning to learn to recognize features. This could be a reason why the results from the transfer learning method, used in this study, didn’t show any positive effects. They, however, also showed that the first layers don’t seem to be specific to a data set and can almost be applied to any task. Due to time constraints, there was not enough time to investigate this further. However, Pan and Yang (2010), show different conclusions about how similar the data sets need to be when applying transfer learning. The interpretation of the current state of transfer learning is that researchers agree that transfer learning is a method that can most definitely help with problems that exist within ANN. There still exist split opinions on how and when it should be applied.
6.3.5 Social Aspects

The social benefits that arise with the given results are that this study further emphasizes the use of RCNN when recognizing multiple characters from an image. Even though this study tested for digits on ID tags, the problem itself exists in a lot of similar areas, which can draw the benefit of the results.

An ethical problem that exists within machine learning is that it is being applied to perform tasks that have previously been performed by humans e.g. self-driving trucks, and manufacturing. This solution further tests the possibility to automate a task that is today being performed by humans. [Relihan (2018)] however, writes that machine learning shouldn’t be discussed as it replacing humans for different tasks but instead redesign which tasks that are being handled by humans. This argument can be applied in this task as this wouldn’t remove a job but instead make it more efficient.

Another ethical problem is that this study can be seen as strengthening the slaughter industry which today itself has a lot of ethical problems as mass production, contribution to global warming and more. While this study itself is not enough to automate any processes, it can still encourage future development in this area. The authors believe that the importance of these questions wouldn’t change even if this task were to be fully automated. These questions that exist today, need to be discussed and solved independently of this study.

6.4 Future Work

There exist intervention that can be applied to the PPNet model as well as the training of a classifier which would likely result in better performance. The most important factor to achieve a better result is believed to be, adding trainable data which can help the network recognize digits. This can be done by either finding a transfer learning method that can help PPNet generalize or add more images to the AIDT data set. If the choice is to add more images to AIDT, it’s important to take into consideration which images are being added to the data set, to increase the variance. The most important behavior for an ANN to be applied and in any way automate the task described in this thesis, is for it to only predict digits when features are present and not always predict eleven digits. This is believed to be an effect of the similarities in the data set and can be solved by increasing the size and variance of the AIDT. This can, as described in section 6.3.2 potentially be solved by applying a threshold before an output is done. This would most likely lower the current result but would make the method more viable for industrial usage. The importance of network size has for a long time been emphasized [Simonyan and Zisserman (2014)]. They, however, demand more data and time to train. By adding more data to AIDT, it enables the use of a larger network that can generalize better.

The authors of this study had minimal prior knowledge about machine learning and its applications, which demanded a lot of time and work before an actual implementation could be developed. This led to a small time-frame to actually change and test the architecture of the model and its different variables e.g. optimizer, learning rate, or data augmentation settings. One method that has shown to prevent the network from overfitting, which was a problem due to the small data set, is dropout [Srivastava et al. (2014)].
The authors believe if changes were to be made on either the AIDT data set or PPNet, better performance than Azure can be achieved.

In this study only, CLA and edit distance were used as a measurement. For future work, it would be a good idea to also include a measure like perfectly predicted sequence accuracy. Even though a low edit distance and high CLA indicate that a lot of predictions were correct, it still clarifies how a classifier performed on a data set.
Bibliography


Figure A.1: PPNet-architecture