DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2021

Pose Classification of Horse Behavior in Video

A deep learning approach for classifying equine poses based on 2D key points

MICHAELA SÖDERSTRÖM

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Pose Classification of Horse Behavior in Video

A deep learning approach for classifying equine poses based on 2D keypoints

MICHAELA SÖDERSTRÖM

Master in Systems, Control and Robotics
Date: July 2, 2021
Supervisor: Sofia Broomé
Examiner: Hedvig Kjellström
School of Electrical Engineering and Computer Science
Swedish title: Pose-klassificering av Hästbeteende i Video
Swedish subtitle: En djupinlärningsmetod för klassificering av hästposer baserat på 2D-nyckelpunkter

Abstract

This thesis investigates whether Computer Vision can be a useful tool in interpreting the behaviors of monitored horses. In recent years, research in the field of Computer Vision has primarily focused on people, where pose estimation and action recognition are popular research areas. The thesis presents a pose classification network, where input features are described by estimated 2D keypoints of horse body parts. The network output classifies three poses: 'Head above the wither', 'Head aligned with the wither' and 'Head below the wither'. The 2D reconstructions of keypoints are obtained using DeepLabCut applied to raw video surveillance data of a single horse. The estimated keypoints are then fed into a Multi-layer preceptron, which is trained to classify the mentioned classes. The network shows promising results with good performance. We found label noise when we spot-checked random samples of predicted poses and comparing them to the ground truth, as some of the labeled data consisted of false ground truth samples. Despite this fact, the conclusion is that satisfactory results are achieved with our method. Particularly, the keypoint estimates were sufficient enough for these poses for the model to succeed to classify a hold-out set of poses.

Keywords

Deep Learning, Computer Visison, Horse behavior, Pose estimation, 2D keypoints, Pose classification, DeepLabCut

Sammanfattning

Uppsatsen undersöker främst om datorseende kan vara ett användbart verktyg för att tolka beteendet hos övervakade hästar. Under de senaste åren har forskning inom datorseende främst fokuserat på människor, där pose-estimering och händelseigenkänning är populära forskningsområden. Denna avhandling presenterar ett poseklassificeringsnätverk där indata beskrivs av uppskattade 2D-nyckelpunkter (eller så kallade intressepunkter) för hästkroppsdelar. Nätverket klassificerar tre poser: 'Huvud ovanför manken', 'Huvud i linje med manken' och 'Huvudet nedanför manken'. 2D-rekonstruktioner av nyckelpunkter erhålls med hjälp av DeepLabCut, applicerad på rå videoövervakningsdata för en häst. De uppskattade nyckelpunkterna matas sedan in i ett flerskiktspreceptron, som tränas för att klassificera de nämnda klasserna. Nätverket visar lovande resultat med bra prestanda. Vi hittade brus i etiketterna vid slumpmässiga stickprover av förutspådda poser som jämfördes med sanna etiketter där några etiketter bestod av falska sanna etiketter. Trots detta är slutsatsen att tillfredsställande resultat uppnås med vår metod. Speciellt var de estimerade nyckelpunkterna tillräckliga för dessa poser för att nätverket skulle lyckas med att klassificera ett separat dataset av samma osedda poser.

Nyckelord

Djupinlärning, Datorseende, Hästbeteende, Pose-estimering, Nyckelpunkter, Intressepunkter, Pose-klassificering, DeepLabCut

Acknowledgments

Foremost, I would like to thank my supervisor Sofia Broomé for bringing me calmness at the end of each week during our meetings and for all the support throughout the project. I would also like to thank Hedvig Kjellström for giving me this opportunity and for setting up monthly meetings, enabling qualitative discussions and insights to my own project as well as other students' projects. Lastly, I appreciate all the communication with Pia Haubro Andersen, Linnea Pålsson, Elin Hernlund and Katrina Ask in concern of equine knowledge and data.

Stockholm, June 2021
Michaela Söderström

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

AI Artificial Intelligence

ANN Artificial Neural Network

AU Action Unit

CNN Covolutional Neural Network

CV Computer Vision

DCNN Deep Convolutional Neural Network

DL Deep Learning

DLC DeepLabCut

DNN Deep Neural Network

ERT Ensemble of Regression Trees

FCN Fully Convolutional Network

GCN Graph Convolutional Network

GUI Graphical User Interface

HOG Histogram of Oriented features

HPE Human Pose Estimation

LSTM Long Short Time Memory

ML Machine Learning

MLP Multi-layer perceptron

PPLO Progressive Pseudo-label-based Optimization

RNN Recurrent Neural Network

SPFES Sheep Pain Facial Expression Scale

SVM Support Vector Machine

WS-CDA Weakly- and Semi-supervised Cross-Domain Adaption

Chapter 1

Introduction

Computer vision enables interaction with the physical world in the way it attempts to replicate the human vision. It is developed for computers to understand and describe content in images and videos as a part of the Artificial Intelligence (AI) and the Machine Learning (ML) fields.

Recently, the Computer Vision (CV) field has advanced remarkably, due to the breakthrough of Deep Neural Network (DNN) [1]. In 2012 *Krizhevsky et al.* [2] presented their seminal image classification model with a Deep Convolutional Neural Network (DCNN), which vastly outperformed on the large dataset ImageNet LSVRC-2010. Their network was trained with supervised learning and they particularly showed that the depth of networks are an important matter in achieving good results on a complex dataset.

Face recognition [3] and human action recognition [4][5] are two examples of popular applications in the CV field, where a great amount of research has been performed in the last decade. In contrast to visual recognition applied on humans, corresponding research among animals is not as wide. To our knowledge, *Lu et al.* [6], who estimated sheep pain using facial recognition, are one of the few contributing to this research area for larger animals. *Hummel et al.* [7] recently presented an article for automatic pain detection on horse faces and the results of applying the horse model on donkeys.

This thesis focuses on interpretation of horse behavior based on poses in video. A machine vision system that could recognize animal behavior would have a particularly positive impact on animal welfare. It would facilitate safety for

animals to get the appropriate support and care during different circumstances.

1.1 Purpose

It is of great value to develop research around animal behavior recognition using machine learning. There exists only one previous study, by *Wang et al.* [8], that involves behavior classification on horses based on 2D keypoints, to the best of our knowledge. The aim of this thesis is to make a contribution to a research area, which can improve the welfare of equines and other animals.

1.1.1 Ethical, sustainability and social issues

As the AI field develops, it will become applicable in a wide range of applications. Hence, it is important to consider ethical aspects and issues when performing research within the field. As a matter of fact, it is our responsibility to develop the technology with a conscious mind on how it impacts our society in different ways.

In general, the ethical point of view in animal based projects is a critical acknowledgement to make. Animals do not have the ability to converse with humans through verbal expressions and in many cases, particularly horses, it takes a large amount of knowledge and experience to understand their behavior during different circumstances. Since they are prey animals, they naturally tend to hide signs of weakness, such as when in pain or suffer from illness. Therefore, it is critical to perform research on animals with their best interest in mind and focus on their well-being. If we succeed to do so, there are great opportunities to develop technology which can be used to improve our understanding about animals and hence improve our ability to increase animal welfare as well.

From a sustainability perspective, the development of CV systems can improve the existing environment of monitored animals. Experiments and research performed on horses often require human interaction and equipment put on the animal. These disturbances in the habitual environment can expose the horses to elevated stress levels. By developing CV products we can diminish the need for human interaction and body equipment, which consequently cause less stressful situations for the horses. Further, it will enable more accurate

behavior interpretation while monitoring, since the horses do not need to hide their natural behavior.

Animal health care is important and we can take advantage of CV implementations outside research as well: observations through video surveillance enables capability to follow the condition of horses over time, without the necessity for a caretaker to be physically present. We are also able to develop tools, which can be used to alert in cases of suspicion that the animal is unwell.

Particularly in this thesis, we do not perform any direct experiments on horses, since we use recorded data with purpose from another study. Our expectations are to contribute to enable a more cautious approach for future research methods on horses and other large animals, by examine the potential of monitoring with the use of CV techniques.

1.2 Scientific questions

Pose estimation is a method used to infer the position and orientation of an object in an image or a video. It is performed by building a model that tracks a number of keypoints, which mark significant features of the object. Different estimated keypoints, can be used as input features in a neural network, trained to recognize them against ground truth labels. This is known as pose classification.

In this thesis, pose classification of horse behavior is based on keypoint information from single image frames rather than sequences of image frames. Hence, it is important that the performance on 2D reconstructed keypoints receive a good result, since it is used as the input in the pose classification task.

1.2.1 Engineering problem definition

Humans need a machine vision system to facilitate the automatic recognition of horse behavior to improve medical welfare of equines.

1.2.2 Scientific questions

It is desired to examine weather equine behavior can be recognized using a computer vision based method, where the positions of the horse are inferred from keypoint information received from raw video data. From this the following questions are formulated:

- Is it possible to receive a sufficiently good reconstruction of 2D keypoints from surveillance videos to describe poses of a horse?
- Can poses from horses successfully be classified from single-view data obtained from 2D reconstructed keypoints?
- How well does the trained model perform on unseen data?

1.3 Objective

It is desired that the project contributes to the progress of ability to measure different behaviors of horses given videos. The aim is to present a framework, which is able to classify poses. Future applications of this kind can be used to facilitate interpretation of equine behavior as a tool in ethology studies and implemented in the process of medical care.

To fulfill the objective, this project has been divided into two sub-tasks as follows:

1. **Keypoint reconstruction.** A network will be trained to reconstruct 2D keypoints that describes horse poses from raw video frames.
2. **Pose classification.** Construction of a network that is trained to classify a set of labeled horse poses from keypoint estimates.

1.4 Approach

The construction of a solution to answer the questions formulated in Section 1.2.2, is based on building a method, which follows the sub-tasks in the objective in Section 1.3.

We inspect the available surveillance videos and the corresponding annotated

horse behavior data to understand the data and its limitations. In the reconstruction phase, we use the DeepLabCut toolbox [9] to extract keypoints and evaluate the performance based on weight initialization of two different networks. Here, train and test sets consist of 400 manually labeled frames with keypoints, which represent the positions of different body parts. The data from the best performing network is then used in the pose classification task after necessary pre-processing steps.

Pose classification is performed by an MLP with input features including the coordinates for confident keypoint body parts for single frames. We train the network with the use of a train and validation sets and evaluate the network performance with a test set on unseen data.

1.5 Delimitations

The research follows a quantitative method, where the project is limited to a pre-recorded dataset of a single horse in a box environment. Hence, we are restricted to investigate one specific animal and we do not cover multiple individuals occurring in a frame simultaneously. Furthermore, apart from the different camera views, the environmental circumstances are unvaried and we do not cover surroundings in the wild.

1.6 Contributions

We explore how accurate 2D keypoints can be reconstructed from raw video data and if it is possible to interpret horse poses given the obtained keypoint estimates.

The critical contributions we achieve in this thesis are:

- Examination of how accurate 2D keypoints can be reconstructed on a horse subject from manual labels on 400 frames obtained from raw video surveillance data.
- Evaluation of pose classification performance on unseen data of the estimated 2D keypoints.

- Ideas on how future work can contribute to interpretation of horse behavior.

1.7 Thesis structure

In Chapter 2, we briefly summarize the history of AI development leading up to today's methods. We describe the MLP architecture and continue with related work concerning pose estimation and action recognition for human and animals. Additionally, we mention related work to action recognition using 2D keypoints. Then, we present the available surveillance data and corresponding behavior annotations in Chapter 3. Furthermore, Chapter 4 presents the method used to solve the research questions. Chapter 5 includes the experimental stage of the pose classification network and the results are presented in Chapter 6 along with analysis. Finally, we discuss our findings in Chapter 7 and finish with conclusions and propose future work in Chapter 8.

Chapter 2

Theoretical background

This chapter introduces the evolution of neural networks, which have provided the foundation to build solutions to various problems, such as examination of a CV framework that can be used to support interpretation of horse behavior. We continue to present previous papers related to pose estimation and action recognition, including humans and animals. Then, a summary is included in the end of the chapter to highlight important aspects in the method chapter.

2.1 The development of neural networks in Artificial Intelligence

The early development of AI, known as the Cybernetics wave, proceeded during the 1940s-1960s. In 1943, Warren McCulloch and Warren Pitts published a paper regarding applications of neural networks to AI. Donald Hebb, introduced the Hebbian learning theory in 1949 and Frank Rosenblatt developed his work about the perceptron neuron during the end of 1950s to the early 1960s. This was presumed to be the beginning of a revolution of neural networks, but resulted in the AI Winter. The progress of neural networks was disrupted for 15 years. It was a result of a mathematical proof in the book Perceptrons, published by Marvin Minsky and Seymour Papert in 1969, conducting the non-linear limitations of a two-layer perceptron [10], which in fact was a trivial problem to solve.

The second wave, known as the Connectionism wave, progressed during the

1980s-1990s. The expansion of backpropagation, introduced by David Rumelhart in 1986, enabled training with multi-layer neural networks [10].

Yann LeCun and Geoffrey Hinton [11] are probably two of the most known people with their work related to the current era of the Deep learning wave, which emerged from 2006 [10]. Their work on Deep Learning architectures have built the ground of today's research in a wide range of research within the AI field.

2.2 Multilayer perceptrons

For multi-class pose classification we use the MLP architecture, presented in Figure 2.1, explained here at a high-level. For more detailed information about MLP and other neural network architectures we recommend this great machine learning book [12].

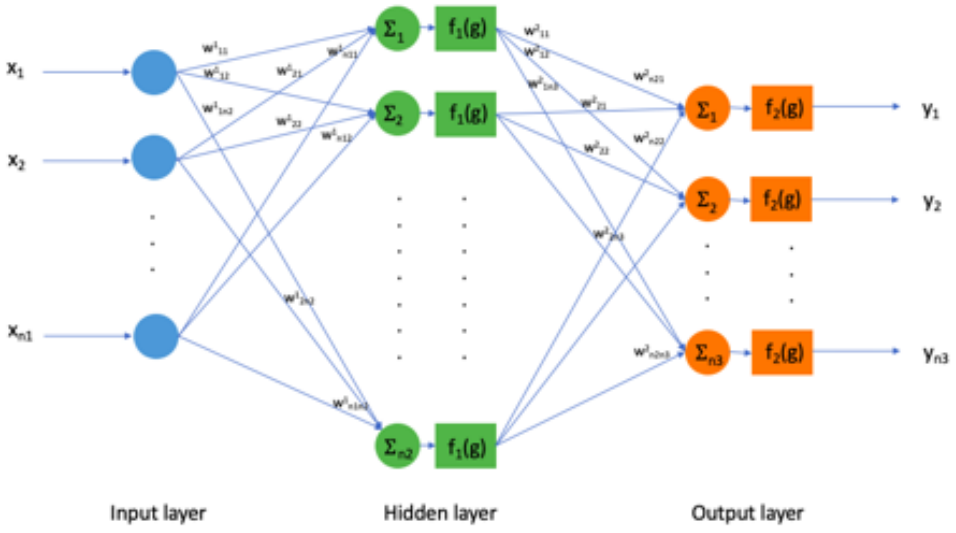


Figure 2.1 – An MLP with three layers. Figure inspired by [13].

The MLP consists of nodes constructed with an input layer, at least one hidden layer and an output layer. Typically, it utilizes a supervised learning approach with the backpropagation learning algorithm for training [13]. Supervised learning means that the network has access to labeled output data, corresponding to each input. Backpropagation is simply computation of the gradient errors between the predicted output and actual output, propagated back to the

input layer, one layer at a time. The nodes are fully connected with each other through different weights. These weights are updated in the learning process based on the gradient errors calculated in the backpropagation step. Each time information is passed though a hidden layer, we transform the output of the nodes with an activation function. The choice of activation function defines how the weighted sum is transformed in each layer and we typically use different activation functions for the hidden and output layers depending on the purpose of the network.
Figure 2.1 shows the basic architecture of an MLP with three layers. The network can mathematically be described as follows [13]:

$$y_r = f_2\left(\Sigma_{q=0}^{n_2} w_{qr}^2 \cdot f_1\left(\Sigma_{p=0}^{n_1} w_{pq}^1 \cdot x_p\right)\right), \quad r \in [1, n_3]. \tag{2.1}$$

Here, the indices of the input, hidden and output nodes are annotated as p, q and r respectively. x_p denotes the input node of the input layer, w_1 the weight set connecting the input and hidden layers and w_2 the weight set connecting the hidden and output layers. $f_1(g)$ and $f_2(g)$ are the activity functions of the hidden layer and output layer respectively and finally, y_r denotes the outputs of the network [13].

2.3 Related work

First we introduce related work which concern different approaches of human and animal pose estimation. Then, corresponding approaches for general action recognition are illustrated. At last, we mention other work, related to animal action recognition.

2.3.1 Human pose estimation

Human Pose Estimation (HPE), is a research field that has been developed for decades and advanced during the revolution of deep learning. One of the first papers that adopted the deep learning approach in HPE, with results on state-of-the-art level at that point or higher, was presented by *Toshev et al.* [14] in 2014. The architecture is based on the DCNN model proposed for image classification [2] and formed as a regression problem. The images are cropped around the predicted joints throughout the layers so the network can learn more fine features. They found that DNNs are advantageous in holistic reasoning.

Chen et al. [15] present a collection of deep learning-based approaches of 2D and 3D human pose estimation in a recent survey. The work focuses on monocular images and video data. The authors underline the advantages of deep neural structures compared to the older, more shallow structures such as [14], but also the importance of efficient networks and adequate training data.

However, pose estimation have yet many challenges, which include occlusion, low resolution, incomplete data caused by single-view, annotation difficulties and real-time performance. To annotate a large dataset manually is also costly and hence not effective [16]. Consequently, deep learning solutions are developed to handle these issues in different ways. For instance, [17] propose a Covolutional Neural Network (CNN) framework for occlusion-awareness.

2.3.2 Animal pose estimation

To label all animal species is an impossible task and today only a few datasets with labeled animal poses are publicly available. Consequently, it causes a major challenge in performance of animal pose estimation since there exists a great variation between animal species as well. *Cao et al.* [18] examine this problem to enable larger annotated datasets for animal species to facilitate training and testing on animal pose estimation tasks. The proposed method focuses on four-legged mammals and exploits prior information from labeled human and animal poses and transfer in action classification on unlabeled animal species. A similar approach is performed by *Rashid et al.* [19]. They feed warped images of horse faces to a pre-trained facial human keypoint detector. The purpose of warped images is to facilitate adaption to the animal faces, since they are visually different from human faces.

Automated machine vision solutions for animals are indeed desired in different industries. With a focus on the farming industry, *Khan et al.* [20] propose a DNN model for animal skeleton extraction based on RGB data inspired by [21]. *Haggag et al.* [22] use RGB-D data, focusing on semantic segmentation on quadruples for a markerless pose estimation approach of the body parts, which can be used for pose estimation.

Mathis et al. [23] investigate the robustness and generalization performance of within-domain and out-of-domain image horse datasets, using three ImageNet

architectures: MobileNetV2s, ResNets, and EfficientNets. Their results show the advantage of using transfer learning over training a network from scratch, especially for out-of-domain data.

2.3.3 Human action recognition

One of the many applications in pose estimation is action recognition. In action recognition we are required to model the spatial as well as the temporal information, which means that it is a more high-level problem as compared to pose estimation. Different approaches of solving this problem are based on RGB videos, infrared videos, depth map sequentials and skeleton data [24].

Wang et al. [25], introduced dense trajectories based on optical flow as an approach to action recognition in videos. In 2014 *Simonyan and Zisserman* proposed a two-stream network for action recognition in videos [26]. Prior work in video action often include Histogram of Oriented features (HOG) and Support Vector Machine (SVM) approaches. They describe shallow high-dimensional encoding of local spatio-temporal features. The authors built two separate recognition streams based on ConvNets combined with late fusion. The temporal stream is based on optical flow and boosts the performance.

The human body can be represented with joints and bones, which form a topological map commonly named skeleton data. These maps are more robust against scaling, occlusion, motion speeds and viewpoints compared to former methods such as RBG videos and depth map sequences. However, skeleton data is represented by some key features, which include spatial correlations between bones and neighbour joints, strong temporal correlation and a co-occurrence relationship between the spatial and temporal domains [27]

Three common methods in skeleton action recognition based on deep learning are sequence-based methods, image-based methods and graph-based methods. These are often modelled by Recurrent Neural Network (RNN), CNN and Graph Convolutional Network (GCN) respectively [24].

Traditional hand-crafted features make action recognition tasks difficult to generalize over different datasets. In comparison, deep learning methods learn features automatically and because of their remarkable performance they have become popular to use in action recognition [27].

In a recent survey [27], *Ren et al.* discuss and highlight some of the state-of-the-art 3D skeleton-based action recognition methods based on deep learning with a focus on RNN, CNN and GCN methods. Figure 2.2 illustrates the general pipeline of 3D skeleton action recognition for deep learning models.

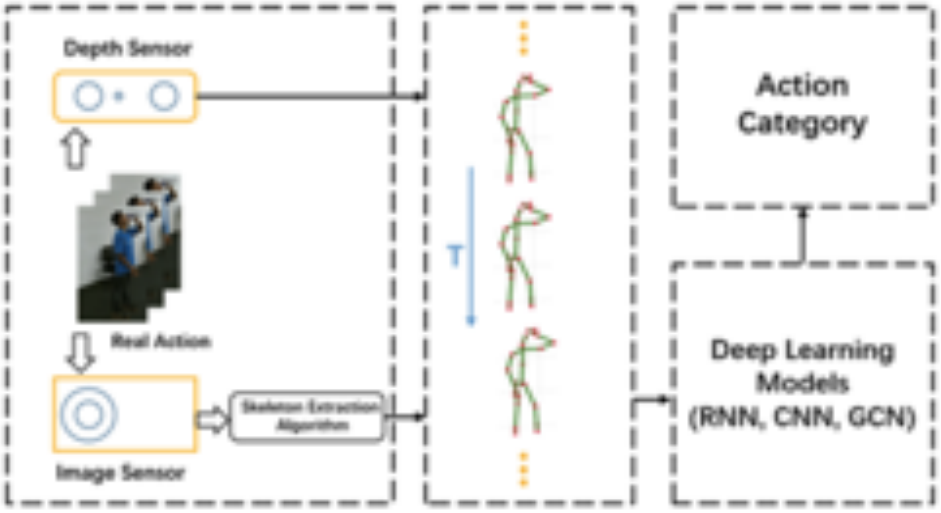


Figure 2.2 – The general pipeline used for skeleton action recognition shows that the skeleton data is obtained either from pose estimation algorithms or from depth sensors. They are in turn sent into a deep learning model that classifies the action. The general aspects of the these deep architectures are discussed in the next few paragraphs. Figure source: [27].

Skeleton Action Recognition with RNN. RNNs have strong temporal modeling ability, but weak spatial modeling ability and a common problem with RNNs are that they suffer from vanishing and sometimes exploding gradients. Long Short Time Memory (LSTM) solves the latter problem to some extent, but can also suffer from gradient decay due to its tangent and sigmoid activation function [27].

Skeleton Action Recognition with CNN. CNN methods are stronger for vision applications than methods based on RNN. Nonetheless, challenges include architecture design, size and speed of the model, changes in viewpoint and information loss in occlusions [27].

Li et al. [28] propose a translation-scale invariant mapping from video to images, where they let RGB components represent the joint coordinates as pixels for a sequence of images. Further, they introduce a multi-scale CNN to solve the action recognition task, where the multi-scale property is a consequence of a fixed convolution kernel with varied input size.

Another approach to solve the skeleton action recognition task is presented by [29]. The authors propose a two-stream CNN, inspired by [26], but in comparison they use raw skeleton coordinates of the joints instead of mapping the skeleton joints to RGB images. Input of the second stream is the skeleton motion, which is computed as the difference of two consecutive joint sets.

Skeleton Action Recognition with GCN. *Shi et al.* saw a gap in pre-existing graph methods, that use un-directed graphs. They motivate that un-directed graphs limit the dependencies that exist between joints and bones. Therefore they propose a directed graph to further improve the skeleton action recognition problem, which reaches state-of-the-art performance on the NTU-RGBD and Skeleton-Kinetics datasets.

2.3.4 Animal action recognition

Most related papers found on animal action recognition do not use deep learning techniques to exploit the specific tasks. However, we think it is important to highlight previous work within the field as well.

In a recent work [30], *Broomé et al.* perform an automatic Deep Learning (DL) approach on recognizing pain vs. no pain in equines based on raw video data. They investigate three architectures and their results show that sequential information is important in the task of recognizing equine pain. Transfer learning is applied on two of the architectures using ImageNet weights. They accomplish the conclusion from comparison between their architectures as one-stream spatial networks and two-stream spatial-temporal networks. Additionally, they perform interesting baseline comparisons between model performance and interpretation from veterinarian experts in classifying equine pain.

Lu et al. [6] propose a solution for an automatic pain level estimate in sheep. The multi-layer approach is able to generalize within the domain of different facial datasets. They also created a sheep facial Action Unit (AU) taxonomy for

frontal sheep faces. Facial features of sheep are described by HOG descriptors and the classification procedure is performed by training a SVM from the labeled AU and the HOG extracted features.

An automatic pain face evaluation model is proposed by *Pessanha et al.* [31] to facilitate evaluation of disease progress in sheep. The model is suited for varied sheep head rotations. Similarly to [6], they extract facial features from HOG and train a SVM to estimate pain.

Hummel et al. [7] perform and evaluate methods to study automatic facial landmarking and pain estimation in horses and donkey faces by classification and fusion. They contribute with a manually annotated horse and donkey dataset that includes landmarks and pain level scores. The authors also emphasized the difficulties of transferring models between the equine family with the purpose of automatic facial landmarking and pain estimation. Similarly to [6][31], they use a SVM classifier, but also experiment with , and CNN beyond the HOG descriptors. The pipeline with HOG and CNN respectively achieve the highest pain estimation performance.

Another CV based approach applied on animal behavior is presented by *Liu et al.* [32]. They propose a method to detect harmful tail-biting among group-housed pigs through action recognition in video. As [30], they utilize spatial-temporal DL methods to detect the behavior. The pigs are tracked individually and the spatial features are handled by a CNN, while the temporal features are interpreted by an LSTM.

2.3.5 Animal behavior studies through 2D keypoints

We have found two research papers closely correlated to the method we examine in this thesis. *Wang et al.* [8] use reconstructed 2D keypoints obtained from DeepLabCut [33] to classify horse lameness in videos. They compare two classification networks; Naïve Bayes and the decision tree algorithm, J48. The networks are trained to classify lameness on three different views separately; side-view, front-view and rear-view. In contrast, we examine pose classification on various views simultaneously in this thesis.

In the second work [34], *Kil et al.* examine the possibility to use automated tracking of equine body parts in videos to measure horse behavior. They label three body parts; wither, tail and nose and investigate the potential to track

changes in horse behavior two days respectively eight days after colic surgery. From their results they conclude that CV is a promising tool for automatic classification of horse behavior.

2.4 Summary

The Related work Section 2.3 presents a glimpse of the performed research in pose estimation and action classification. As the CV field and ML field develop, these research areas have a great potential of being implemented to a wide range of applications.

Major research have been applied on human data, but we can declare the necessity of focusing on animal purposes as well. For instance, Section 2.3.4 presents a batch of work focus on using CV to evaluate pain on large animals. That is one important application towards improving animal welfare.

In this thesis, we focus on classifying animal poses based on frame per frame data rather than sequential data, but it is still closely related to the action classification tasks since the methodology could be developed to more high-level approaches. Previous work in Section 2.3.5 illustrates interesting applications on how automated 2D keypoints tracking can contribute to animal behavior research.

Chapter 3

Method, part I: Horse dataset

This project is restricted to using pre-recorded surveillance videos of single horse in a box-environment. The data was collected prior to this study under approval by the Swedish Ethics Committee in accordance with the Swedish legislation on animal experiments by [35]. An ethogram with annotated behavioral activities, associated with the surveillance videos, is also available [36]. These include basal behaviors such as 'eating', 'grooming', 'standing' and 'kicking; as well as spatial behaviors such as 'placement of the head', 'placement in the box' and 'direction in the box'.

3.1 Horse subject

The dataset we use in this project consists of surveillance videos of one horse subject. It is a standardbred trotter gelding of 13 years when the videos were recorded. The color of the gelding is brown. The subject did not indicate any signs of diseases [35].

3.2 Surveillance videos

All videos include a single horse subject, staying in a box-environment without human surveillance. The box has a setup of four cameras, which enables corresponding view variations of a subject during the same recorded sequence. The horse subject is recorded twice with a frame rate of 20 fps and the original

resolution is 2688x1520. As a result, eight video sequences are available in total, all approximately 45 minutes each.

3.2.1 Camera setup

The videos were recorded with four IR network surveillance cameras from Hikvision Digital Technology Co., Hangzhou China. The placement of the cameras were in each corner of the box, approximately 0.180 m above the ground. Thereafter, the recorded videos were cut out to fit 45 minutes in VideoPad Professional (version 7.11, NCH Software, Inc., Greenwood Village, USA). More details in Ask [35].

3.3 Behavior annotation of the surveillance videos

The behaviors of the horse subject were manually annotated by the author of [36] for the video sequences in the dataset. The author used the open source software BORIS [37] as support in the annotation process.

The four videos from each set of recordings are retouched and concatenated into one single video with four quadrants and behavior annotation is performed from these. The concatenated videos are also cut to perfect the synchronisation between the videos in the best way possible. The reason behind this was that the BORIS software lacked capacity to run four high resolution videos. Hence, the concatenated videos could run with the original resolution. Since the videos are not always perfectly synchronized, as mentioned in Section 3.3.1, the drawback for us at this point is that it is impossible to tell exactly which video the labeled time for an action is annotated on. We hold the data for the separate video recordings, which differ slightly in playback speed, as displayed in Table 3.1 and obtained from measures in OpenCV.

Recorded view	Recorded fps
1_view00	19.97
1_view01	19.93
1_view02	20.0
1_view03	20.0
2_view00	19.96
2_view01	20.0
2_view02	20.0
2_view03	20.0

Table 3.1 – Video recordings of each camera view with corresponding fps.

3.3.1 Synchronization issues

There exists sequences, where there is an issue with lagging. The consequences of the synchronization issues are that [36] could not always cover annotation over the complete video sequences. Hence, the videos of the horse subject do not cover a complete annotation regarding behavior, see Table 3.2.

Subject	Start time video hh:mm:ss	Annotation start mm:ss	Annotation stop mm:ss
Horse ID	09:43:00	00.00	45.00
Horse ID	13:13:00	06.27	45.00

Table 3.2 – An overview of the total annotation time for the two video recordings of the horse subject.

Based on the annotated time in Table 3.2, only the sequences which include behavior annotation will be considered to be used as input data for the supervised pose classification task.

3.3.2 Discussion about the available data

The data used in this project was not recorded with the application of this particular thesis in mind. Therefore, the surveillance video recordings might not be optimal for this task. It exists challenges when configuring the camera setup for the video recording sessions. Firstly, it is important to place the cameras in a way so the view is practical. Secondly, one must decrease the risk

for horses to chew on the cameras to prevent the equipment being destroyed as a consequence.

To visualize the data we are working with, Figure 3.1 presents a pick of example frames occurring in the videos. The view of the horse subject in the videos vary substantially: depending on the position and distance from the cameras, a horse subject is viewed in different scales which consequently results in frames with frequently occluded body parts. As previously stated in this Chapter, the data also includes noise, at times so severe that the horse subject is not visible in the frame. Besides this, we underline the good lightning conditions our data persist.

Figure 3.1 – Example of frame captures showing the variability of data of the horse subject.

Chapter 4

Method, part II: Recognition of horse behavior through video surveillance

In this chapter, the key methods and ideas behind the implementations are presented. An overview of the workflow is presented in Figure 4.1. First, necessary information about the data is presented. Then, we dedicate a section explaining the methods and reasoning behind the 2D keypoint reconstruction and pose classification network. Finally, we conclude the chapter with a few implementation details.

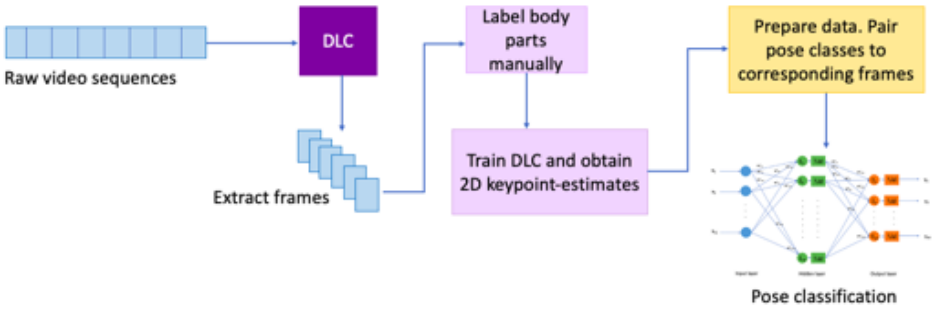


Figure 4.1 – Raw video sequences are input in a pose estimation toolbox. Frames are extracted from the videos and body parts are manually labeled. From the DLC toolbox [33] we obtain reconstructed 2D keypoints of the labeled body parts. We pair these frames with their corresponding pose class and then the data is fed into a pose classification network.

The overall structure we carry out for the investigation is: a number of frames are manually annotated with a set of keypoints on the horse's body obtained from each video. The reconstruction of keypoints is performed from training via the DeepLabCut toolbox [9]. Analysis of the performance is performed based on evaluation by DeepLabCut and by manual inspection.

Given the output from DeepLabcut, a classification network is built and trained on the poses and their pose labels. The architecture is constructed as a MLP classification network. The network performance is evaluated on a hold-out set and we spot-check random samples of true and false predicted poses and compare them to the corresponding visual frames with confident keypoints.

4.1 Pose labels

Inspection of the annotated horse subject in the Excel files indicated three interesting poses to choose as pose labels for the pose classification task: 'Head above the wither', 'Head aligned with the wither' and 'Head below the wither'. Each pose label were given a corresponding class label used in classification training as presented in Table 4.1. These labels are chosen primarily for their considered simplicity. We believe it is easier for the network to classify spatial behaviors, such as the head position compared to basal behaviors such as eating. In future applications, it would be of interest to include the basal behaviors, such as 'eating' and 'kicking', as subclasses of the head positions. However, we assume that it is beneficial to add temporal features and perform the classification on sequences for these events.

Pose label	Class label
Head above the wither	0
Head aligned with the wither	1
Head below the wither	2

Table 4.1 – Each pose label is paired with a corresponding class label, which are used in the pose classification network.

We choose to focus on a small set of poses to establish a valid evaluation of the performance of the trained network. The advantage of the chosen pose labels is that they exist in all videos and the poses cover each video fully. However, it is possible to extend labels within these pose labels in future experiments, since they include behavioral actions such as standing and eating.

4.2 2D keypoint reconstruction

The goal of the keypoints reconstruction task is to obtain 2D pose estimations from single horse data of the eight videos. We use the popular DeepLabCut toolbox [33] for reconstruction purposes.

4.2.1 DeepLabCut – a toolbox for animal pose estimation

In this project we use the DeepLabCut toolbox [33], but other toolboxes for pose estimation also exist. These include OpenPose [38] and DeeperCut [39], developed for human pose estimation. Additionally, LEAP [40] and DeepPoseKit [41] are deep learning tools that have been applied on animals.

DeepLabCut is an OpenSource toolbox developed by Mathis Group and Mathis Lab [33], which focuses on animal pose estimation. It is user friendly and does not require advanced programming skills [9]. The method exploits transfer learning for domain adaption and deep learning to reconstruct keypoints on animals' bodies. Transfer learning typically allows better generalization and robustness as well as shorter training time and smaller datasets. In the article [23], *Mathis et al.* present advantages of transfer learning based on their study, which compare performance on pose estimation with and without transfer learning.

We summarize the general pipeline: in short, the user creates a project and uploads the video sequences, which DeepLabCut can extract frames from. The extracted frames allows the user to manually annotate points of interest. Further, reconstruction training is performed on the raw frames with initialization weights from a pre-trained network, typically ResNet. After training, the performance is evaluated. If satisfied, the user can analyze the videos and visualize the confident keypoints over whole video sequences. An overview of a 2D reconstruction pipeline can be reviewed in paper [9].

The extracted frames are labeled using the DeepLabCut Graphical User Interface (GUI). The creators behind DeepLabCut [33] encourage to label the keypoint positions as consistently as possible and the user can choose either if to label occluded body parts or not. If occluded body parts are labeled,

the network learns to guess the keypoint positions in these cases. When the analysis of the trained videos is performed, the user gets access to the predicted keypoint pixel coordinates and their corresponding likelihood for each frame.

4.2.2 Keypoint positions

The initial approach when deciding body parts to annotate with keypoints was to choose them with horse anatomy in mind with some modification. At first, we considered to annotate 20 keypoints focusing on the whole body of the horse subject, but later we reconsidered the amount of keypoints. Instead, the annotation of keypoints focus on the head and neck area. One reason for this was to increase the likelihood to label keypoints consistently since the annotation is not performed by an expert. The second reason is that we assume that these are beneficial body parts to include as input features in the pose classification network. As acknowledged in Section 3.3.2, the data include a wide variation of the captured horse positions in the different views, hence the decrease of keypoint labels is a reasonable choice to make. Table 4.2 presents the final decision of eight body parts to label and Figure 4.2 the color that represents each body part. The colors are arbitrarily chosen and only matters in visualization purposes.

Figure 4.2 – Each body part corresponds to a color used to visualize labels in DeepLabCut.

Number	Body part
1	Croup
2	Highest point back
3	Nostril Left
4	Eye Left
5	Ear Left
6	Ear Right
7	Eye Right
8	Nostril Right

Table 4.2 – The final body parts labeled in DeepLabCut.

The keypoints in Table 4.2 are chosen, since they are reasonably easy for a non-expert to annotate with sufficient accuracy. 'Highest point back' is the wither of the horse and we label the rear, 'Croup', to give some information about the orientation of the horse in relation to the head.

4.2.3 2D keypoint reconstruction approach

We create a DeepLabCut project for the horse subject with the eight videos. Next, 50 frames that are used in the label process are extracted from each video. K-means clustering is applied to extract the frames to create a diverse training set of the horse. The k-means method down-samples the videos and clusters the frames. Each frame is treated like a vector and frames from different clusters are selected to label [9]. Table 4.3 shows the results from the frame extraction step.

Recorded view	# frames	Recorded fps	Total length ss
1_view00	53940	19.97	2700.47
1_view01	53813	19.93	2699.54
1_view02	54053	20.0	2702.87
1_view03	54004	20.0	2700.47
2_view00	53978	19.96	2704.38
2_view01	53990	20.0	2699.93
2_view02	53999	20.0	2700.19
2_view03	54005	20.0	2700.53

Table 4.3 – The Table presents the number of extracted frames in DeepLabCut from the recorded fps and length of the corresponding recorded sessions and camera view points.

The body parts are manually labeled as stated in Table 4.2. As recommended by the DeepLabCut developers [33], invisible body parts were not labeled and occluded parts correspondingly. However, body parts considered to be sufficient visible to label were labeled. To give the network the opportunity to learn to skip labeling noisy frames that do not include any body parts we keep very noisy frames. In total, we label 400 frames. Before training, the individual labeled frames are manually checked to confirm the labeling does not contain any possible mistakes. When checking the labels, the natural variation of the manual labeling is obvious, due to human error. This indicates that label fewer keypoints is a good start when not involving experts.

According to the article [42], ResNet50 is often a good choice as the pre-trained network, but for more challenging problems it is possible to consider ResNet101 or even ResNet152. For our keypoint reconstruction we demonstrate a training comparison of ResNet50 and ResNet101 in Subsection 4.2.4.

It is interesting to compare the performances since we suspect that it is possible that our data is challenging.

Further, we analyze the keypoint detection performance of the network with the best performance. From the analysis, we also create videos which contain the confidence of keypoint predictions for visualisation purposes. The output data obtained from DeepLabCut, which consists of the predicted keypoint coordinates and corresponding likelihood values, is further used in our pose classification network after data preparation.

4.2.4 Details on training and evaluation

The training process is performed accordingly: we run the training with pre-trained ResNet50 and ResNet101 networks respectively, to initialize the weights. The networks are trained over 300 000 iterations with batch size 1. The number of iterations are chosen to confidently verify that the loss reaches a plateau and the batch size is chosen to fit program ware limitations with the large images. Figure 4.3 shows the loss over time for the two networks. The confidence parameter, p-cutoff is the default value 0.6. This parameter sets the boundary of which confidence level the predicted keypoints are kept for classification. As recommended by the developers [33], we use image augmentation, where random cropping is used and the scaling interval was set to 10-50% of the original size. Consequently, the size pends approximately between sizes 286x152-1344x760. Usually in CV, we work with smaller frame sizes, but according to the developers the network performs slightly better on larger frame sizes [43]. Their recommended scaling parameters are set to 50-125%, but since our original sized frames are large, we lower that scale.

To clarify, the training is performed with a train set of 380 frames and a test set of 20 frames. These frames are composed of all eight videos. The networks are trained with a learning rate of 0.005 for the first 10 000 iterations and then increased to 0.02 for the remaining iterations, as default in DeepLabCut. The results for ResNet50 and ResNet101 is presented in Figure 4.3.

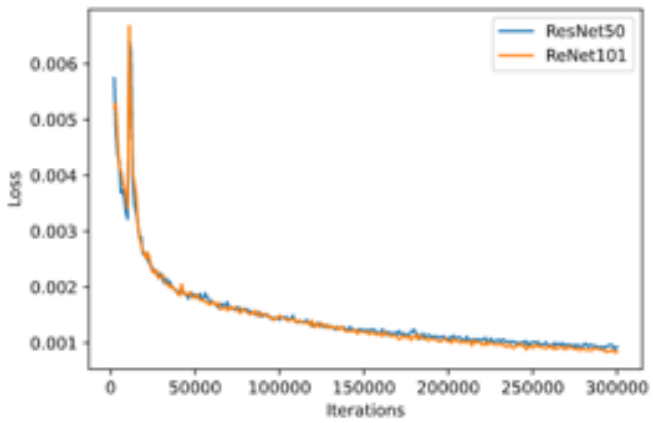


Figure 4.3 – Comparison of the training loss with ResNet50 and ResNet101 for 300 000 iterations.

The loss displayed in Figure 4.3 is similar for both networks. If the networks would have been trained for further iterations the result might have slightly improved, but we think it is sufficient for evaluation. We observe a spike after approximately 10 000 epochs before the network stabilizes again. An explanation to this could be that the change of learning rate causes the network weights get into a bad local minima, which it later recovers from. Since the loss for ResNet101 in this case is slightly better than ResNet50 after training, we choose to proceed with the results from ResNet101. Table 4.4 presents samples from the generated DeepLabCut output of the keypoint estimates.

Body parts	**X**	**Y**	**Likelihood**
Nostril R	470.45	710.85	0.07380
Eye R	470.42	711.94	0.05061
Ear R	445.42	397.06	0.7161
Ear L	445.40	397.25	0.8392
Eye L	470.15	713.48	0.7682
Nostril L	428.14	1007.50	0.3484
Highest point back	966.64	499.79	0.9641
Croup	1250.78	459.81	0.9412

Table 4.4 – Example content of DLC output data for a random frame with rounded values. X and Y denotes the estimated keypoint coordinates in an image and the likelihood is the result from a softmax, $\in [0, 1]$.

4.3 Preparation of input data

The aim of this section is to present the method used to generate the input data used in the pose classification network.

4.3.1 Curating the behavior annotation data

The Excel files with annotated horse behavior in Section 3.3 were cleaned up, sorted and converted to csv-file format. It was performed by removing unwanted columns, excess pose labels, renaming the Swedish activity labels from Swedish to corresponding English labels and adding a column with the class labels. An example of the raw unedited Swedish data is presented in Figure 4.4 and an example row from the sorted data is presented in Table 4.5.

Subject	Time	Total length	FPS	Behavior	Behavioral category	Modifier 1	Comment	Status
Horse A	1065.602	2698.530	30	Huvud placering	huvudplacering	c nedanför manken		START
Horse A	1068.555	2698.530	30	ben flexion/ext	Ben	Vänster framben		POINT
Horse A	1069.103	2698.530	30	Huvud placering	huvudplacering	c nedanför manken		STOP

Figure 4.4 – Samples of the unedited Swedish behavior annotation data. The Time column denotes the time for the events in seconds, Total length is the duration of the recorded video in seconds, and FPS the annotated recorded playback speed. Further, there exist columns for Behavior, Behavioral category and Modifier 1, where the latter is the column we collect our chosen poses from. Start and stop for the behavioral events are denoted by Status.

In Figure 4.4 we observe that the annotated fps is 30, and not around 20 as presented in Tables 3.1 and 4.3. Since we have measured the same playback speeds in DeepLabCut and manually controlled in OpenCV, we do not consider to use 30 fps for further data processing.

Subject	**Start time ss**	**Stop time ss**	**Pose label**	**Class**
Horse A	0.604	1065.601	Head above the wither	0

Table 4.5 – The curated data includes columns of the horse subject, start and stop time for annotation of the pose label and its corresponding class. This Table presents a sample from the curated data.

4.3.2 Pair DLC output frames with corresponding class

The start and stop time, declared in Table 4.5, reveal that some frames do not have an annotated class. These frames are removed after calculating which frames belong to which class, based on the obtained output of reconstructed keypoints from DeepLabCut, accordingly:

$$frame_i = fps_{view} \cdot t_i, \quad i \in [start, stop], view \in [00, 01, 02, 03], \tag{4.1}$$

where $frame_i$ and is the frame number, fps_{view} the playback speed for the specific camera view point and t_i is the annotated time for an occurring class. The output structure from predicted pixel coordinates and corresponding likelihood from DeepLabCut is constructed as in the small example of two body part columns in Figure 4.5.

	Nostril_R			Eye_R		
	x	y	likelihood	x	y	likelihood
13	493.612671	642.214844	0.026177	491.182373	644.970215	0.160674
14	493.512115	636.915710	0.035404	492.361877	639.365967	0.153143
15	493.751160	634.252808	0.035825	492.766937	635.737366	0.160967
16	1116.506714	1322.956909	0.038153	492.274963	626.422058	0.100630
17	1116.194092	1323.429321	0.038004	493.735779	620.931396	0.131515

Figure 4.5 – Sample of DeepLabCut output of two body parts after performing the analysis step on a video. Each body part column includes three subcolumns with predicted x and y coordinates and their corresponding likelihood. The index row corresponds to a specific video frame, $frame_i$, calculated from Equation 4.1.

Each video frame is paired with its corresponding class and subsequently we choose to only use the most confident keypoints. This means that we set a value -1 on all pair of x and y coordinates, which have a likelihood below the confident value, p-cutoff=0.6. All likelihood columns are then removed, since we only mean for every frame row to include x and y values and its corresponding class label.

To enable an approach to qualitatively analyze our test results, we assure to add the video ID and frame number as the third and second last columns; the class labels are contained as the last column. Lastly, we concatenate the following datasets used in our pose classification network accordingly: two

separate datasets, which contains the data from the four corresponding video recordings from the different views and one final dataset which contains all the data from the eight videos.

4.3.3 Class distribution

In total, the final dataset includes 378 167 rows of pose data for all labeled body parts. An examination of the class distributions in Figure 4.6 reveals that the classes are imbalanced. Class 0, 'Head above the wither', contains the majority of data samples and Class 2, 'Head below the wither', is clearly underrepresented. This is a case of severe imbalance, but the cause of imbalance in our case is from natural occurrence in the horse behavior.

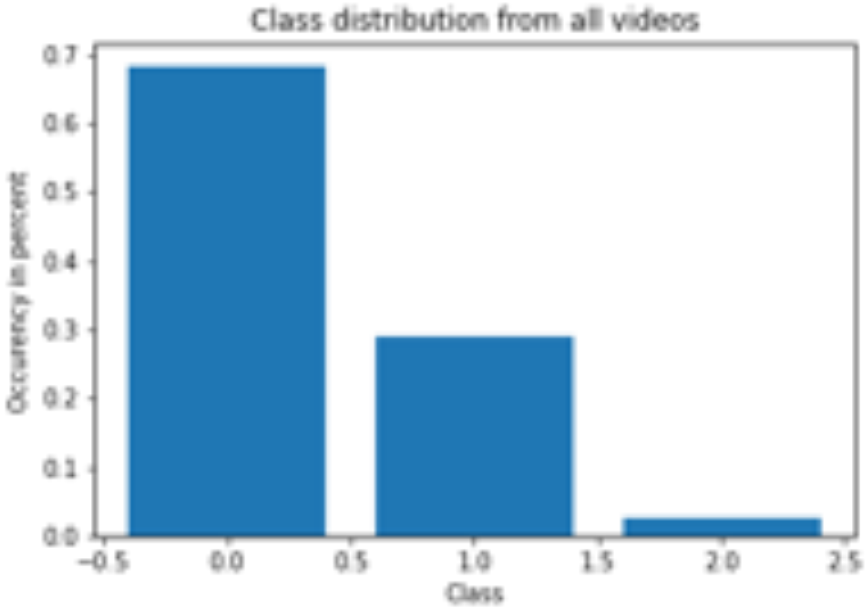
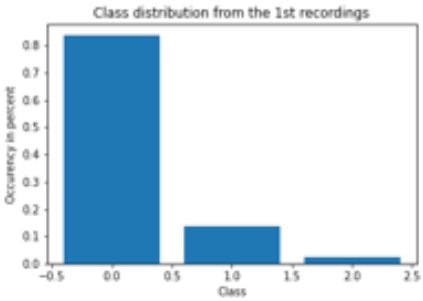


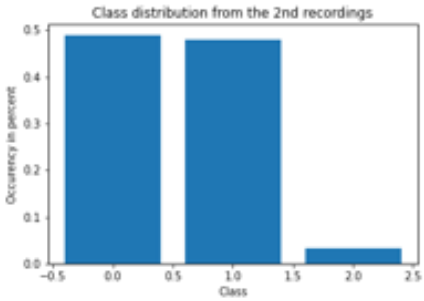
Figure 4.6 – Percentage class distribution of the final dataset, which consists of the eight concatenated videos. Class 0, 'Head above the wither', consists of 258 387 data samples and represents about 68 % of the total data distribution. Class 1, 'Head aligned with the wither', which consists of 109 568 samples represents about 29 % and Class 2, 'Head below the wither', with 10 212 samples corresponds to 2.7 % of the distribution.

It is interesting to visualize the class distribution separately for the two recordings as well. They are presented in Figure 4.7 and we can observe that Class

1, 'Head aligned with the wither', has low representation in (a) and high representation in (b), while Class 0 is underrepresented in both figures.



(a) Class 0 has 18 0570 data samples, Class 1; 29 908 samples and Class 2; 4 971 samples.



(b) Class 0 has 79 660 data samples, Class 1; 77 817 samples and Class 2; 5 241 samples.

Figure 4.7 – Figures (a) and (b) present the percentage class distribution of the dataset that belongs to the first and second round of recorded videos. We declare the exact amount of samples in each figure.

We attempt to solve the imbalance issue by oversample the minority classes with a random oversampling technique. This is performed by computing the class weights and pass them to the `WeightedRandomSampler` method in PyTorch. This method is passed as a sampler in PyTorch `DataLoader`, which provides an iterable over a dataset. Thus, minority classes have a higher probability to be included in each batch. This means that the minority class examples will be exposed to the network multiple times. An overview of how the class weights are computed can be reviewed in pseudo code, Algorithm 1.

Algorithm 1: Computation of the class weights.

computeClassWeights$(y_train, train_set)$:

i. `class_count` ← Count the number of occurrences in each class

ii. `target_list` ← Create a list with all the outputs obtained from `train_set`

iii. Shuffle `target_list`

iv. Compute class weights

`class_weights` ← $1/$`class_count`

v. Get weights for each sample

`class_weights_all` ← `class_weights`$[$`target_list`$]$

return `class_weights`

4.4 Pose classification

The pose classification uses the results from the 2D reconstruction phase generated from the preparation of input data and classifies the given pose labels. In this section, we discuss the approach behind building our classification network. For reproducibility purposes, a seed is used when creating the train, validation and test datasets. The seed is specified throughout the experiments in Chapter 5.

4.4.1 Prepare datasets

To prevent data leakage, that is to prevent statistical information to be shared between the separate datasets, the dataset is split into a training, validation and test set *before* any further processing. Then, we normalize it accordingly:

$$X_n = \frac{(X_n - min(X_n))}{(max(X_n) - min(X_n))}, \quad n \in [train, validation, test], \tag{4.2}$$

where X_n is the original sample value, $min(X_n)$ the minimum column value and $max(X_n)$ the maximum column value in each dataset. Next, we resample the dataset used for training with our resampling method. The final datasets are created by allowing shuffling the frame indices. Since each frame contains a single pose, we are not dependent on the order of frames.

4.4.2 Model architecture

For our MLP classification network, the number of input features are presented by the x and y coordinates of the predictions of the body parts. As mentioned in Section 4.3, less confident body parts are given value -1 to demonstrate that they are not included, while confident body parts preserve their estimated values. The input dimension is hence, 1x16 for each frame fed to the network. The output layer consists of three nodes, which represent the class labels.

We choose the ReLU activation function in the hidden layers since real life data is non-linear and ReLU is a popular choice that handles non-linearity [44] and the Softmax activation function in the output layer to classify the poses.

To limit issues with vanishing or exploding gradients, we use He-initialization with normal distribution to initialize the weights [45]. This is typically a good choice since the chosen activation function is ReLU and has no derivative at zero. For the use of logarithmic Softmax in the output layer we initialize the weights with Xavier initialization [46].

Cross-entropy loss is applied to our network since it is a multi-classification problem. In our use case, the loss is the weighted average across observations for the mini-batches we set to 100 samples for both train and test sets. The total loss for an iteration is computed by multiplying the size of each batch. The epoch loss is then calculated by dividing the sum of the total loss with the length of the specific dataset.

The optimization algorithm used during training is Adam, simply because it is often a good choice when it is desired that the network does not get stuck in local optimum and it is desirable for it to converge fast [47].

4.4.3 Validation of the MLP network

To verify that the MLP network works properly, we use the Iris dataset [48] and feed the network with a small set of data and try to overfit the network performance. It is performed to ensure that the network can overfit to the training set when it is trained on only a few examples. We use random seed 400 for train and test splits and no regularization techniques are applied. The loss of the verification run is presented in Figure 4.8.

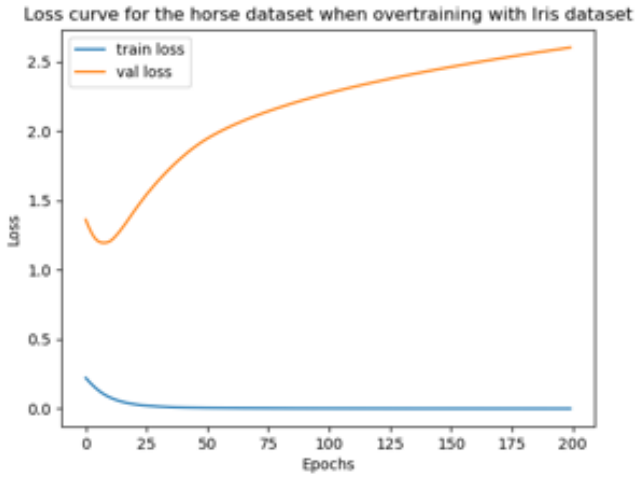


Figure 4.8 – The result of overtraining the Iris dataset with a batch size of 5 samples for each train and test sets over 200 epochs with learning rate 1e-3. The MLP network has 100 nodes in the input layer and 50 nodes in the hidden layer.

After verification on the Iris dataset, we perform the same procedure for one of the horse videos. The result of the loss is presented in Figure 4.9.

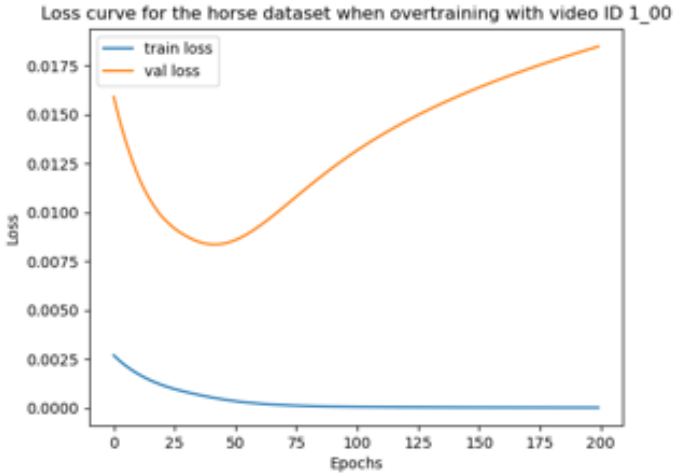


Figure 4.9 – The result of overtraining the Horse dataset for video ID 1_00 with a batch size of 8 samples for each train and test sets over 200 epochs with learning rate 1e-3. The MLP network has 80 nodes in the input layer and 40 nodes in the hidden layer.

The loss in Figure 4.9, which is trained with data from one of our videos, follows the same trend as Figure 4.8 trained with the Iris dataset. They show that it is possible to overtrain our network on a small batch of data. The results give an indication on that our network behaves as expected.

4.4.4 Implementation details

2D reconstruction: Frame extraction and labeling of the horse subjects are performed on a local computer, while the remaining steps; training, evaluation and creation of labeled videos are executed on GeForce GTX 1080 Ti, GPU. Dependencies used are Python 3.7, Tensorflow 2.2, Cuda 10.1, Deeplabcut 2.1.8 and Deeplabcut-core. When running on GPU we set DLCLight to True.

Data preparation and pose classification: PyTorch 1.8.1, Pandas 1.1.3, Numpy 1.19.2, Matplotlib 3.3.2, OpenCV 4.5.1 and Scikit-learn 0.23.2.

Chapter 5

Experiments

This chapter presents the experimental stage of the pose classification network. We aim to construct experiments that generate answers to our primary scientific questions formulated in Section 1.2.2, related to pose classification. The experiments are motivated throughout the chapter.

5.1 Model fitting

The model is fitted with the use of train and validation datasets. We begin the process with one hidden layer and without regularization techniques to examine the network performance trained with three different random seeds used in the train test split. Different seeds are used to observe if the trend corresponds to various dataset splits. There are thousands of alternatives for choosing and testing random seeds. We progress with three different randomly chosen seeds; 400, 0 and 102. The loss and accuracy results from training a three layer MLP with strucure 80 nodes in the input layer and 40 nodes in the hidden layer for 50 epochs with these random seeds are presented in Appendix B. From these figures we observe that they all follow the same trend. Without further justifications, we use random seed 400 for continuous network optimization.

We train the same network over 200 epochs with the chosen seed and the results are presented in Figure 5.1 and 5.2.

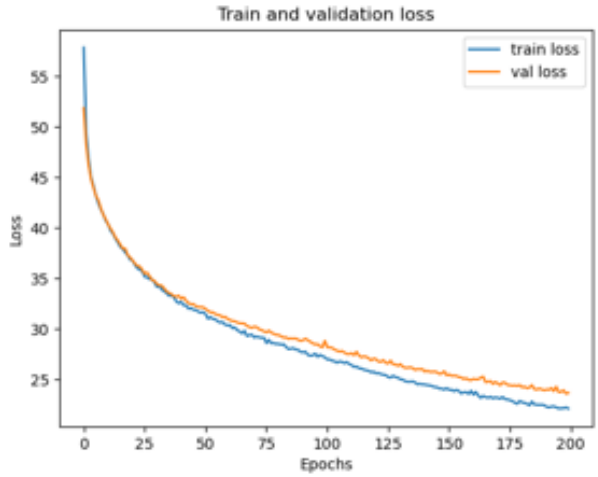


Figure 5.1 – Loss curve for random seed 400, with learning rate 1e-4 and batch sizes 100 for train and validation sets. The MLP layer structure is 80-40 neurons for input and hidden layer. No regularization.

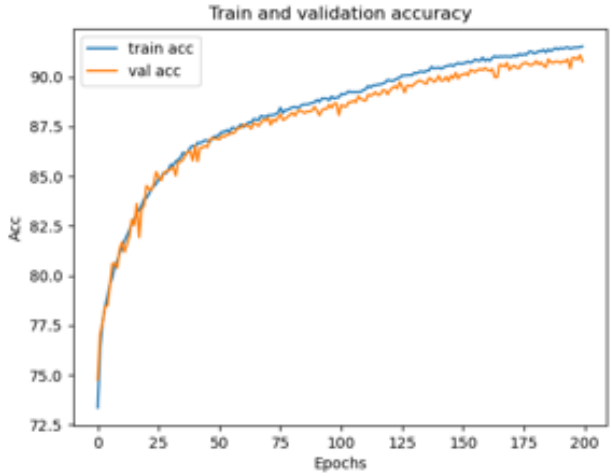


Figure 5.2 – Accuracy curve for random seed 400, with learning rate 1e-4 and batch sizes 100 for train and validation sets. The MLP layer structure is 80-40 neurons for input and hidden layer. No regularization.

From the Figures 5.1 and 5.2 above, we observe that the validation loss decently follows the train loss curve and that the performance follows the expected behavior; the validation loss is relatively higher than the train loss, but

its corresponding accuracy is slightly below the train accuracy. Although, the loss is somewhat high and we choose to expand the network with one more hidden layer of 40 neurons as a strategy to lower the curve. The loss and accuracy curve for the four layer MLP, trained over 500 epochs, is presented in Figure 5.3 and Figure 5.4.

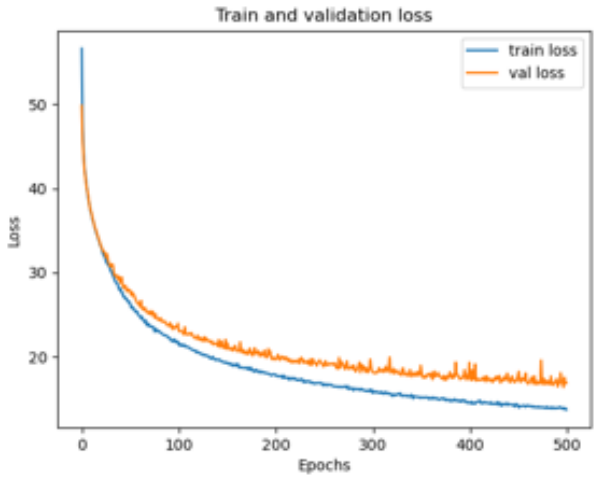


Figure 5.3 – Loss curve for random seed 400, with learning rate 1e-4 and batch sizes 100 for train and validation sets. The MLP layer structure is 80-40-40 neurons for input and hidden layers. No regularization.

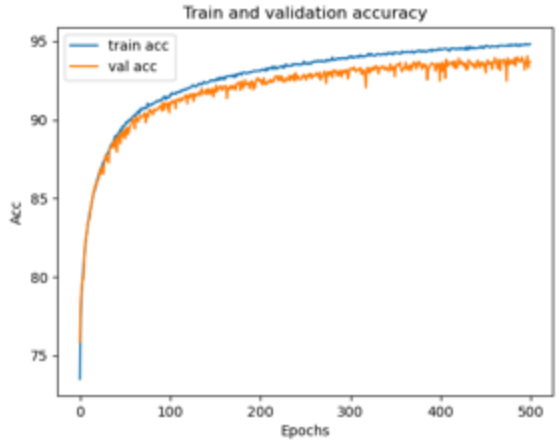


Figure 5.4 – Accuracy curve for random seed 400, with learning rate 1e-4 and batch sizes 100 for train and validation sets. The MLP layer structure is 80-40-40 neurons for input and hidden layers. No regularization.

The loss and accuracy curves in Figures 5.3 and 5.4 are improved with a faster decreasing loss and slightly better accuracy compared to the corresponding epochs for the three-layer network. However, the validation curves fluctuate more and we observe that the network slightly overfits. To resolve this issue, we introduce dropout regularization after the first layer.

5.2 Final hyperparameter settings

We present our final hyperparameters as follow: layer structure: (16, 80)-(80, 40)-(40, 40)-(40, 3) with dropout regularization of 0.1 in the first hidden layer. We train over 500 epochs with a learning rate of 1e-4 with batch sizes 100 for the train and test sets. The loss and accuracy curves for the final model, trained for 500 epochs, is presented in Figure 5.5 and Figure 5.6.

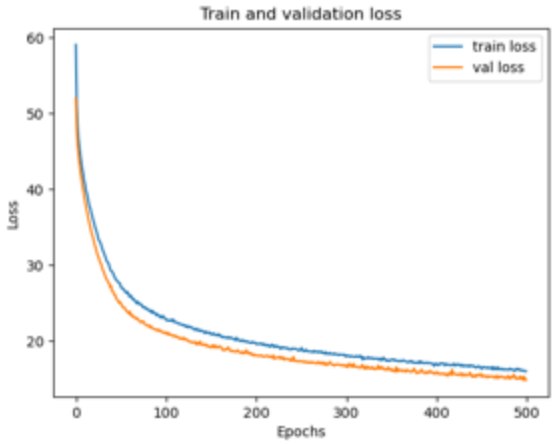


Figure 5.5 – Loss curve for random seed 400, with learning rate 1e-4 and batch sizes 100 for train and validation sets. The MLP layer structure is 80-40-40 neurons for input and hidden layers, with dropout regularization 0.1 after the first hidden layer.

Figure 5.5 demonstrates that the dropout stabilized the network with improved loss performance. We observe that the validation loss is somewhat lower than the train loss, which is expected since dropout regularization is added. The regularization only affects the training process, meaning that the validation curve is more robust.

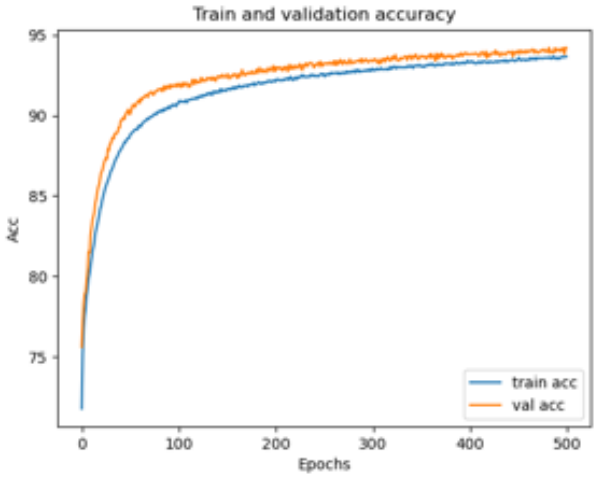


Figure 5.6 – Accuracy curve for random seed 400, with learning rate 1e-4 and batch sizes 100 for train and validation sets. The MLP layer structure is 80-40-40 neurons for input and hidden layers, with dropout regularization 0.1 after the first hidden layer.

Figure 5.6 also implies that the model performance is more stable and for the same reason as for the loss curve, the validation accuracy is higher because it is more robust due to the dropout regularization.

According to the loss and accuracy results, the final network imply that our model should perform at least sufficiently well on the test data. Since there are no tendencies for the loss curve to overfit at the point of 500 epochs and the performance has started to plateau, we are satisfied with the model and do not introduce early stopping to find the best weights. Hence, this is the model we evaluate the hold-out set on.

Chapter 6

Results and analysis

First, we introduce the results obtained from the 2D keypoint classification by providing visual examples of when the network predicts confident keypoints successfully and unsuccessfully. Then the results from the pose classification network is presented. All results follow with analysis.

6.1 Results from 2D keypoint reconstruction

Evaluation of training is performed by computing the mean average Euclidean pixel error between the manual labels and the predicted labels by DeepLabCut [9]. The results from training with ResNet101 for 300 000 iterations are presented in Table 6.1.

Variable	Value
Training iterations	300000
Training dataset (prc)	95
Train error (px)	105.4
Test error (px)	94.12
p-cutoff used	0.6
Train error with p-cutoff	54.57
Test error with p-cutoff	53.53

Table 6.1 – Training results with ResNet101. "prc" is an abbreviation for percent (here referring to the size of the training dataset) and "px" is an abbrevation for pixels.

From Table 6.1 we observe that the train and test error for confident predictions are around 54 pixels (recall that the original frame size is 2688x1520). With a better consistency in the manual labels, the performance could improve. Another approach to improve the results is to extract outlier frames, i.e. frames that have faulty predictions, relabel them correctly, merge them to the training set and train the network again. This step is not performed here due to the time limit. We regard the confident predictions as good enough at this time.

In Appendix B we present a few results of the predicted keypoints compared to the manual labels. From these frames, we can observe that the network would benefit from refinement, but the most confident keypoints is overall fairly well.

6.2 Results from pose classification

The final model is evaluated on the unseen test set. We plot the result from a calculated confusion matrix and construct it as a heatmap, presented in Figure 6.1. From the heatmap we observe that the network perform well on all classes, yet Class 2, 'Head below the wither', has few samples compared to Class 0, 'Head above the wither', and Class 1 , 'Head aligned with the wither'.

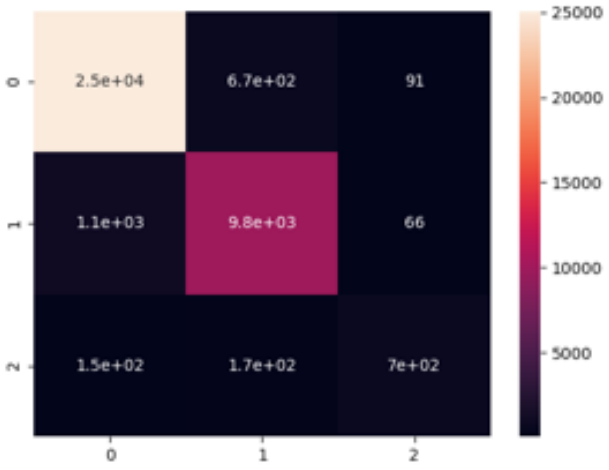


Figure 6.1 – The confusion matrix presented in a heatmap. The vertical axis represent the ground truth labels and the horizontal axis the predicted labels. The diagonal demonstrates the results of true positive predictions.

A presentation of the performance on the test data can further be reviewed in Table 6.2. Inspection of the macro averages, the high precision indicates that the rate of false positive predictions is low and the high recall indicates that we have a high rate of true positive predictions. Since we have an imbalanced class distribution, the F1-score gives the most relevant feedback on the network performance. The F1-score measures the balance between the precision and recall. We have achieved an F1-score of 87 %. For the respective classes, Class 2 has the lowest F1-score, which is expected due to its low representation in the distribution. Although, the score is yet significantly above the random performance of 33 % and we can also observe the very high precision of 82 %.

Classification report	**Precision**	**Recall**	**F1-score**	**Support**
0	0.95	0.97	0.96	25839
1	0.92	0.90	0.91	10957
2	0.82	0.68	0.74	1021
Accuracy			0.94	37817
Macro average	0.90	0.85	0.87	37817
Weighted average	0.94	0.94	0.94	37817

Table 6.2 – Classification report. The Table presents the precision, recall, F1-score and number of data samples, i.e. support separately for each class as well as averaged. The accuracy is the F1-score for all classes. For imbalanced datasets, the macro average presents the most representable results.

We evaluate the performance further by spot-checking a few true and false predicted labels from video 1_00, recorded and visually inspect the confident predicted keypoint estimates for corresponding frames. Figures 6.2-6.7 demonstrates the spot-check samples with information about the ground truth labels and their predicted labels.

Figure 6.2 – Frame number 3306. Ground truth: 0; Prediction: 0

Figure 6.3 – Frame number 35192. Ground truth: 1; Prediction: 1

Figure 6.4 – Frame number 51249. Ground truth: 2; Prediction: 2

Figure 6.5 – Frame number 50165. Ground truth: 0; Prediction: 2

Figure 6.6 – Frame number 30109. Ground truth: 1; Prediction: 0

Figure 6.7 – Frame number 50630. Ground truth: 2; Prediction: 0

Inspection of Figures 6.2-6.4 with true positive predictions clearly demonstrates good examples of when the network performs well. They all have accurate confident keypoint estimates even though, there exists a faulty one in Figure 6.4.

False predictions, such as in Figure 6.6 demonstrates an example of when the network does not succeed in classifying the poses. However, in this case we can make sense of why: the ground truth pose in Figure 6.6 is ambiguous, even for the visual eye.

The undoubtedly interesting images here are presented by Figures 6.5 and 6.7. The ground truth label in Figure 6.5 is Class 0, 'Head above the wither', while the real-life position clearly is displayed by Class 2, 'Head below the wither', which also is the prediction by the network. Further investigation is performed by calculating the time, which corresponds to frame number 50165; time elapse 2512 s. Then we go back to check the class label of the frame at the annotated time in the original Excel file. We detect that the horse moves from Class 0 to Class 2 at annotated time 2510.603-2510.604 s. The reasonable explanation to this misleading ground truth label is tracked back to the synchronization performed between the videos, mentioned in Chapter 3. In the other case, the ground truth label in Figure 6.7 is Class 2, while the predicted label is Class 0. This is also a case where the network has correctly classified the true pose, while the ground truth label is wrong. The same procedure is repeated, to compare the annotated time labels in the original file with the given ground truth label, where frame number 50630; time elapse 2353.303 s. Between 2533.854-3533.855 s, we observe the same case with a pose transition from Class 0 to Class 2. Hence, we draw the same conclusion, leading back to the synchronization issue.

Chapter 7

Discussion

In this chapter we discuss the final results from the 2D keypoint reconstruction and the pose classification performance. We highlight positive and negative aspects of our method.

We have demonstrated that it is possible to classify a selection of equine poses from estimated 2D keypoints with good results on held-out data with a reasonably shallow neural network, MLP. From Table 6.1 in Section 6.1 we have observed that DeepLabCut was able to predict confident body parts with a train and test error around 54 pixels. As stated in the analysis, performance could improve further by refining the labels from outlier frames. Thus, it is also possible that the amount of confident keypoints could improve as well. Inspection of the visualization of predicted keypoints in Appendix A display that the most effortless keypoints to predict are the ears, wither and the croup, while the other seems more difficult for the network to predict confidently without refinements. Although, we also observe that the network fails to predict even the more effortless keypoints accurately at times. From this, we confirm that the input data in the pose classification network is imperfect, but also includes accurately predicted confident keypoint estimates. Hence, this indicates that it was a favorable decision to remove less confident keypoint estimates in our data.

The final MLP network used for pose classification demonstrates a good performance in classifying the selected poses. The macro average F1-score presented in the classification report in Table 6.2 of 87 % is impressive, significantly above the random score, with the knowledge that our data is noisy and

it consists of misleading ground truth labels as was detected in Figures 6.5 and 6.7. The overall score might have been even better for this reason. Our network performs remarkably well also on the smallest minority class, Class 2, which represents about 3 % of the class distribution. Remember that our final dataset consists of over 350 000 samples, which probably have contributed positively to the great effect. These show promising results for future applications.

The results from the spot-check of the predicted sample data verify some gaps in our method. That the input data consists of false ground truth labels is unsatisfactory, but on the other hand it would take effort to manually double check that each frame is paired with their right class. Therefore, we consider it as acceptable extra noise. To avoid this in other applications, it is desirable that the specific information match all surveillance data properly. On the other hand, we are satisfied with the final results, since we have managed to generate interesting results and we underline that the network performance is better than our labeled classes in some cases. Nevertheless, it is possible to improve future work in many ways, which we discuss this briefly in Chapter 8.

Chapter 8

Conclusions and future work

In this final chapter we conclude our findings by answering the scientific questions stated in Section 1.2.2 and present ideas for future development of horse behavior studies using CV techniques.

8.1 Conclusions

Our results demonstrate promising prospects to develop CV techniques for interpretation purposes of horse behavior on a more profound level in the future. By estimating sufficient 2D keypoints, which represent body parts, from raw video data we have found that it is possible to classify three poses from single frames. Evaluation of the test results show that the trained model performed well given confident keypoint predictions even though our data was noisy. It was also successful in interpreting the minority classes, especially Class 2 that only represented about 3 % of the class distribution. Consider that we labeled eight keypoints and about half of them had accurate confident predictions many times, we infer that it is not necessary to label more body parts for these particular poses. Despite this, for other use cases with more challenging poses, it could mean that it is necessary to label more keypoints. Lastly, we point out that the evaluation results are not completely reliable due to the existence of false ground truth labels in the data. However, from manual observation we can conclude that the results were more than acceptable despite this issue.

8.2 Future work

We have many ideas on how to progress with work, which concern interpretation of horse behavior: it is desirable to expand the number of poses and examine classification performance on more advanced poses and to refine the 2D reconstruction to achieve even better results. Further, contrary to use single-view inputs, one can add dimension to the inputs by constructing a multi-view approach and concatenate frames from the same instant as input. Additionally, it is interesting to add the temporal aspect by, for instance, building a two-stream CNN and perform classification on action sequences rather than poses. A two-stream network would contain one spatial stream and on temporal stream. Another expansion is to reconstruct a 3D skeleton and classify poses or activities based on the 3D reconstructions of poses.

Bibliography

[1] Waseem Rawat and Zenghui Wang. “Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review.” In: *Neural Computation* Volume 29, Issue 9 p. 2352-2449 (2017).

[2] Ilya Sutskever Alex Krizhevsky and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks.” In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105. URL: https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[3] Park U. and Jain A.K. “3D Model-Based Face Recognition in Video.” In: *Advances in Biometrics. ICB 2007. Lecture Notes in Computer Science* 4642 (2007). DOI: https://doi.org/10.1007/978-3-540-74549-5_113.

[4] Karen Simonyan and Andrew Zisserman. *Two-Stream Convolutional Networks for Action Recognition in Videos*. 2014. arXiv: 1406.2199 [cs.CV].

[5] Limin Wang, Yu Qiao, and Xiaoou Tang. “Action Recognition With Trajectory-Pooled Deep-Convolutional Descriptors.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[6] Y. Lu, M. Mahmoud, and P. Robinson. “Estimating Sheep Pain Level Using Facial Action Unit Detection.” In: *2017 12th IEEE International Conference on Automatic Face Gesture Recognition (FG 2017)*. 2017, pp. 394–399. DOI: 10.1109/FG.2017.56. URL: http://ieeexplore.ieee.org/document/7961768/.

[7] H. I. Hummel et al. “Automatic Pain Detection on Horse and Donkey Faces.” In: *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*. 2020, pp. 793–800. DOI: 10.1109/FG47880.2020.00114.

[8] Yiqi Wang et al. “Identifying Lameness in Horses through Deep Learning.” In: *Proceedings of the 36th Annual ACM Symposium on Applied Computing*. SAC ’21. Virtual Event, Republic of Korea: Association for Computing Machinery, 2021, pp. 976–985. ISBN: 9781450381048. DOI: 10.1145/3412841.3441973. URL: https://doi.org/10.1145/3412841.3441973.

[9] Tanmay Nath* et al. “Using DeepLabCut for 3D markerless pose estimation across species and behaviors.” In: *Nature Protocols* (2019). DOI: https://doi.org/10.1038/s41596-019-0176-0.

[10] Charles C. Tappert. “Who Is the Father of Deep Learning?” In: *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*. 2019, pp. 343–348. DOI: 10.1109/CSCI49370.2019.00067.

[11] Yann LeCun, Y. Bengio, and Geoffrey Hinton. “Deep Learning.” In: *Nature* 521 (May 2015), pp. 436–44. DOI: 10.1038/nature14539.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[13] D. Lo, Chih-Chiang Wei, and En-Ping Tsai. “Parameter Automatic Calibration Approach for Neural-Network-Based Cyclonic Precipitation Forecast Models.” In: *Water* 7 (July 2015), pp. 3963–3977. DOI: 10.3390/w7073963.

[14] Alexander Toshev and Christian Szegedy. “DeepPose: Human Pose Estimation via Deep Neural Networks.” In: *2014 IEEE Conference on Computer Vision and Pattern Recognition* (June 2014). DOI: 10.1109/cvpr.2014.214. URL: http://dx.doi.org/10.1109/CVPR.2014.214.

[15] Yingli Tian Yucheng Chen and Mingyi Hea. *Monocular human pose estimation: A survey of deep learning-based methods*. 2019. DOI: https://doi.org/10.1016/j.cviu.2019.102897.

[16] Zhenyu Liu Rui Li and Jianrong Tan. “A survey on 3D hand pose estimation: Cameras, methods, and datasets.” In: 93 (2019), pp. 251–272. DOI: https://doi.org/10.1016/j.patcog.2019.04.026.

[17] Y. Cheng et al. “Occlusion-Aware Networks for 3D Human Pose Estimation in Video.” In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 723–732. DOI: 10.1109/ICCV.2019.00081.

[18] J. Cao et al. “Cross-Domain Adaptation for Animal Pose Estimation.” In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 9497–9506. DOI: 10.1109/ICCV.2019.00959. URL: https://ieeexplore.ieee.org/document/9009505.

[19] Maheen Rashid, Xiuye Gu, and Yong Jae Lee. “Interspecies Knowledge Transfer for Facial Keypoint Detection.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[20] Ullah M. Quddus Khan A. Khan S. and Cheikh F.A. “A Bottom-Up Approach for Pig Skeleton Extraction Using RGB Data.” In: *Image and Signal Processing. ICISP 2020. Lecture Notes in Computer Science*. Vol. 12119. Springer, Cham., 2020. URL: https://doi.org/10.1007/978-3-030-51935-3_6.

[21] Eric T. Psota et al. “Multi-Pig Part Detection and Association with a Fully-Convolutional Network.” In: *Sensors* 19.4 (2019). ISSN: 1424-8220. DOI: 10.3390/s19040852. URL: https://www.mdpi.com/1424-8220/19/4/852.

[22] Hussein Haggag et al. “Semantic body parts segmentation for quadrupedal animals.” In: *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2016, pp. 000855–000860. DOI: 10.1109/SMC.2016.7844347.

[23] Alexander Mathis et al. “Pretraining Boosts Out-of-Domain Robustness for Pose Estimation.” In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2021, pp. 1859–1868.

[24] Lei Shi et al. “Skeleton-Based Action Recognition With Directed Graph Neural Networks.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019, pp. 7912–7921. URL: https://openaccess.thecvf.com/content_CVPR_2019/papers/Shi_Skeleton-Based_Action_Recognition_With_Directed_Graph_Neural_Networks_CVPR_2019_paper.pdf.

[25] H. Wang et al. “Action recognition by dense trajectories.” In: *CVPR 2011*. 2011, pp. 3169–3176. DOI: 10.1109/CVPR.2011.5995407.

[26] Karen Simonyan and Andrew Zisserman. *Two-Stream Convolutional Networks for Action Recognition in Videos*. 2014. arXiv: 1406.2199 [cs.CV].

[27] Bin Ren et al. *A Survey on 3D Skeleton-Based Action Recognition Using Learning Method*. 2020. arXiv: 2002.05907 [cs.CV]. URL: http://arxiv.org/abs/2002.05907.

[28] Bo Li et al. “Skeleton based action recognition using translation-scale invariant image mapping and multi-scale deep CNN.” In: *2017 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*. 2017, pp. 601–604. DOI: 10.1109/ICMEW.2017.8026282.

[29] Chao Li et al. “Skeleton-based action recognition with convolutional neural networks.” In: *2017 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*. 2017, pp. 597–600. DOI: 10.1109/ICMEW.2017.8026285.

[30] Sofia Broomé et al. “Dynamics are Important for the Recognition of Equine Pain in Video.” In: *CoRR* abs/1901.02106 (2019). arXiv: 1901.02106. URL: http://arxiv.org/abs/1901.02106.

[31] F. Pessanha, K. McLennan, and M. Mahmoud. “Towards automatic monitoring of disease progression in sheep: A hierarchical model for sheep facial expressions analysis from video.” In: *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*. 2020, pp. 387–393. DOI: 10.1109/FG47880.2020.00107. URL: https://ieeexplore.ieee.org/document/9320292.

[32] Dong Liu et al. “A computer vision-based method for spatial- temporal action recognition of tail-biting behaviour in group-housed pigs.” In: *Biosystems Engineering* 195 (July 2020), p. 27. DOI: 10.1016/j.biosystemseng.2020.04.007.

[33] Alexander Mathis et al. “DeepLabCut: markerless pose estimation of user-defined body parts with deep learning.” In: *Nature Neuroscience* (2018). URL: https://www.nature.com/articles/s41593-018-0209-y.

[34] Nuray Kil, Katrin Ertelt, and Ulrike Auer. “Development and Validation of an Automated Video Tracking Model for Stabled Horses.” In: vol. 10. Animals, 2020. DOI: https://doi.org/10.3390/ani10122258.

[35] Katrina Ask et al. “Identification of Body Behaviors and Facial Expressions Associated with Induced Orthopedic Pain in Four Equine Pain Scales.” In: *Animals* 10.11 (Nov. 2020), p. 2155. ISSN: 2076-2615. DOI: 10.3390/ani10112155. URL: http://dx.doi.org/10.3390/ani10112155.

[36] Pålsson Linnea. “Activity budget and pain behavior in horses with induced othopedic pain.” In: (2020).

[37] Olivier Friard and Marco Gamba. “BORIS: a free, versatile open-source event-logging software for video/audio coding and live observations.” In: *Methods in Ecology and Evolution* 7.11 (2016), pp. 1325–1330. DOI: https://doi.org/10.1111/2041-210X.12584. eprint: https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210X.12584. URL: https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.12584.

[38] Z. Cao et al. “OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

[39] Eldar Insafutdinov et al. *DeeperCut: A Deeper, Stronger, and Faster Multi-Person Pose Estimation Model*. 2016. arXiv: 1605.03170 [cs.CV].

[40] Willmore Pereira T.D. Aldarondo D.E. and L. et al. “Fast animal pose estimation using deep neural networks.” In: 16 (2019), pp. 117–125. DOI: https://doi.org/10.1038/s41592-018-0234-5.

[41] Jacob M Graving et al. “DeepPoseKit, a software toolkit for fast and robust animal pose estimation using deep learning.” In: *eLife* 8 (2019), e47994. DOI: https://doi.org/10.7554/eLife.47994.

[42] In: (). URL: https://github.com/DeepLabCut/DeepLabCut/wiki/What-neural-network-should-I-use%5C%3F-(Trade-offs,-speed-performance,-and-considerations)?fbclid=IwAR3Os62LY3wWFmFfC4MfdIrmummDWG8DFC6r68_aTkBJpTuveGSwA4mn8ME.

[43] Gary A Kane et al. “Real-time, low-latency closed-loop feedback using markerless posture tracking.” In: *eLife* 9 (Dec. 2020). Ed. by Gordon J Berman et al., e61909. ISSN: 2050-084X. DOI: 10.7554/eLife.61909. URL: https://doi.org/10.7554/eLife.61909.

[44] Theodoridis Sergios. *Machine Learning (Second Edition), Chapter 18 - Neural Networks and Deep Learning*. Second Edition. Academic Press, 2020, pp. 901–1038. ISBN: 978-0-12-818803-3. DOI: DOI:10.1016/B978-0-12-818803-3.00030-1. URL: https://www.sciencedirect.com/science/article/pii/B9780128188033000301.

[45] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification." In: *CoRR* abs/1502.01852 (2015). arXiv: 1502.01852. URL: http://arxiv.org/abs/1502.01852.

[46] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html.

[47] Jing Feng et al. "Flower Recognition Based on Transfer Learning and Adam Deep Learning Optimization Algorithm." In: New York, NY, USA: Association for Computing Machinery, 2019. ISBN: 9781450372985. DOI: 10.1145/3366194.3366301. URL: https://doi.org/10.1145/3366194.3366301.

[48] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

Appendix A

2D keypoint-estimates versus manual labels

Here is an extraction of predicted keypoints from the test set results in DeepLabCut, presented in figures A.1-A.13. Manual labels are denoted with "+", confident predictions (p-cutoff > 0.6) with "." and less confident predictions (p-cutoff =< 0.6) with "x" [9]. The keypoint colors correspond to a predicted body part corresponding to what is presented in Figure 4.2.

Figure A.1 – Example of test result from DeepLabCut for predicted vs. manual labels.

Figure A.2 – Example of test result from DeepLabCut for predicted vs. manual labels.

Figure A.3 – Example of test result from DeepLabCut for predicted vs. manual labels.

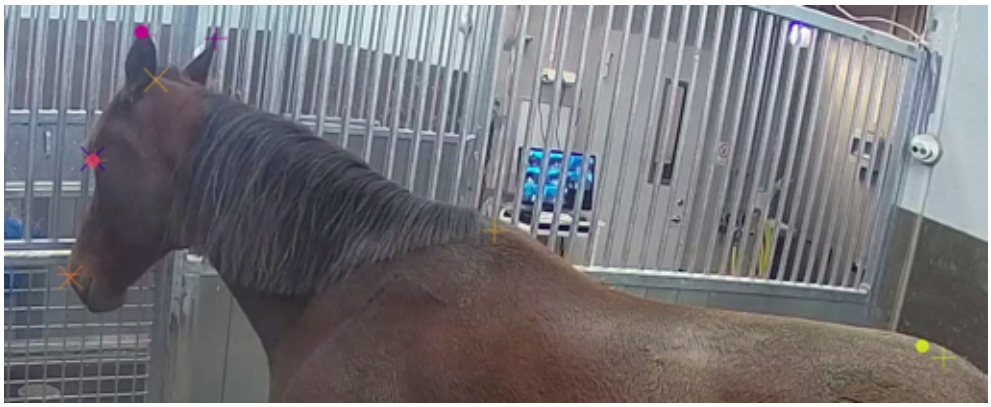


Figure A.4 – Example of test result from DeepLabCut for predicted vs. manual labels.

Figure A.5 – Example of test result from DeepLabCut for predicted vs. manual labels.

Figure A.6 – Example of test result from DeepLabCut for predicted vs. manual labels.

Figure A.7 – Example of test result from DeepLabCut for predicted vs. manual labels.

Figure A.8 – Test result from DeepLabCut for predicted vs. manual labels.

Figure A.9 – Example of test result from DeepLabCut for predicted vs. manual labels.

Figure A.10 – Example of test result from DeepLabCut for predicted vs. manual labels.

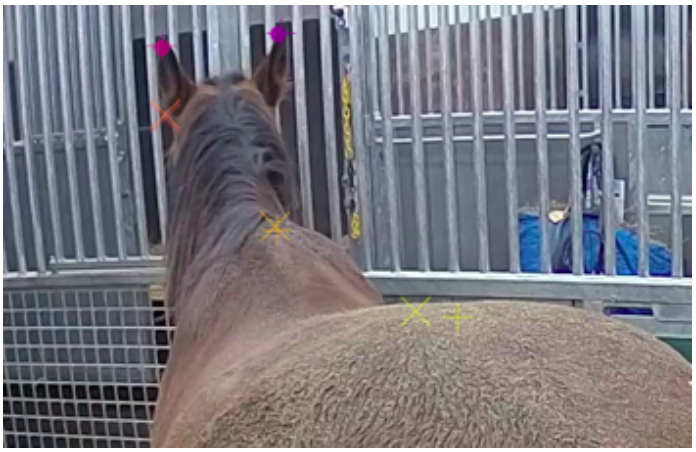


Figure A.11 – Example of test result from DeepLabCut for predicted vs. manual labels.

Figure A.12 – Example of test result from DeepLabCut for predicted vs. manual labels.

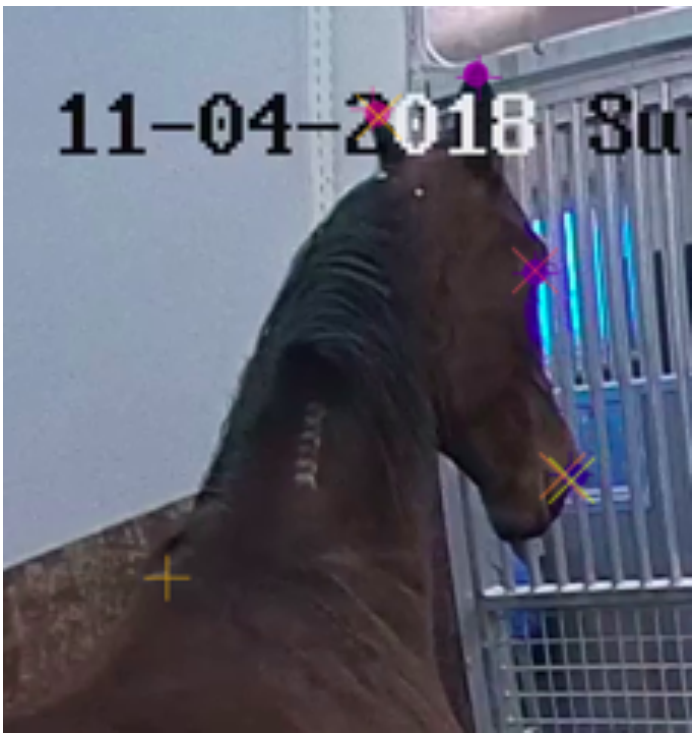
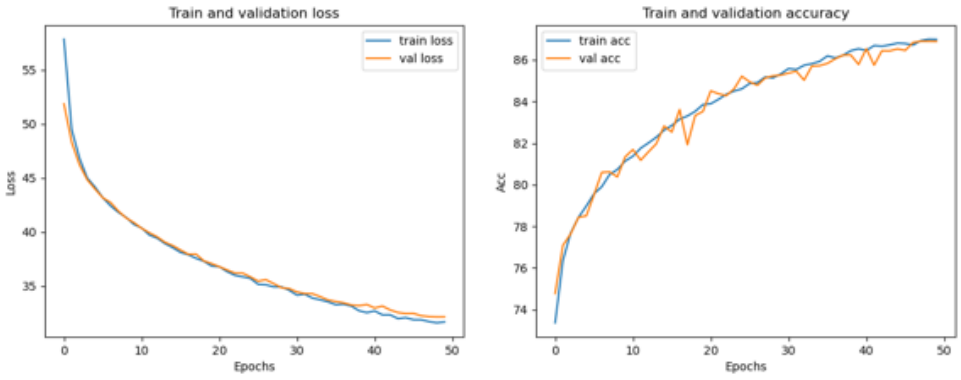


Figure A.13 – Example of test result from DeepLabCut for predicted vs. manual labels.

Appendix B

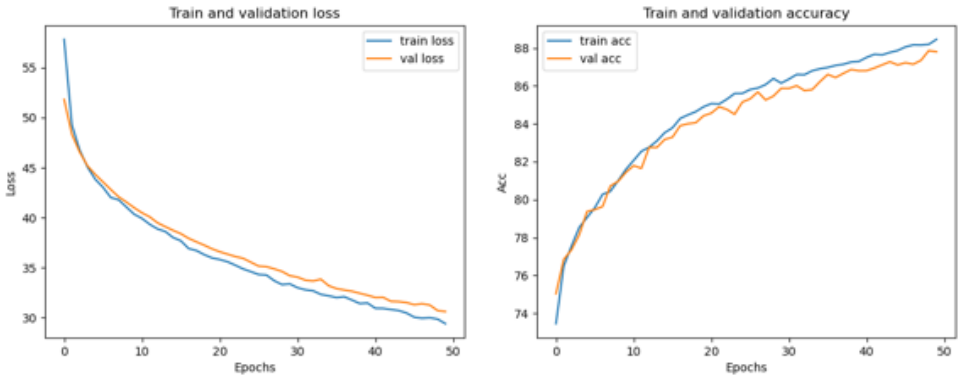
Loss and accuracy for random seeds

Figures B.1-B.3 presents the results from Chapter 5, where three different seeds were used to split the train, validation and test sets and examine the the trends of loss and accuracy over 50 epochs.



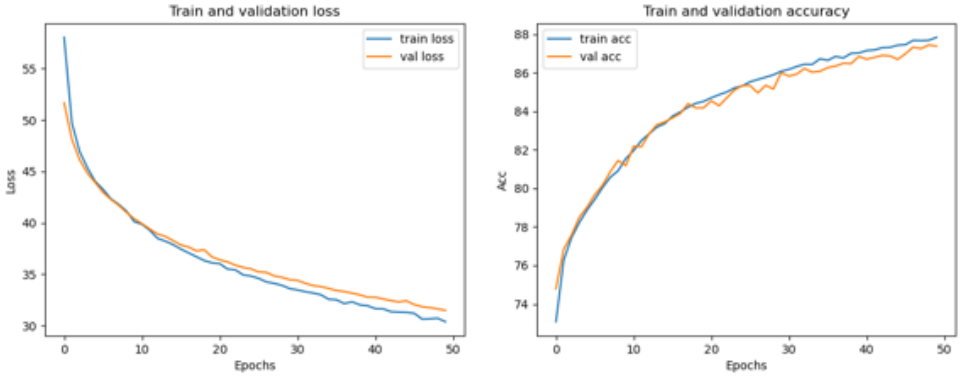
(a) Loss with random seed 400. **(b)** Accuracy with random seed 400.

Figure B.1 – Loss and accuracy curves for random seed 400, with learning rate 1e-4 and batch sizes 100 for train and validation sets. The MLP layer structure is 80-40 neurons for input and hidden layer. No regularization.



(a) Loss with random seed 0. **(b)** Accuracy with random seed 0.

Figure B.2 – Loss and accuracy curves for random seed 0, with learning rate 1e-4 and batch sizes 100 for train and validation sets. The MLP layer structure is 80-40 neurons for input and hidden layer. No regularization.



(a) Loss random seed 102. **(b)** Accuracy random seed 102.

Figure B.3 – Loss and accuracy curves for random seed 102, with learning rate 1e-4 and batch sizes 100 for train and validation sets. The MLP layer structure is 80-40 neurons for input and hidden layer. No regularization.

TRITA -EECS-EX-2021:639