Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Martin Vold

Skeleton Based Cerebral Palsy Diagnosis using Deep Learning and Attention

Master's thesis in Computer Science (MTDT)
Supervisor: Heri Ramampiaro
August 2020

NTNU
Norwegian University of
Science and Technology

Martin Vold

Skeleton Based Cerebral Palsy Diagnosis using Deep Learning and Attention

Master's thesis in Computer Science (MTDT)
Supervisor: Heri Ramampiaro
August 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Abstract

In recent years, Deep Learning has achieved great results in fields such as Computer Vision and Human Activity Recognition. Within the medical domain, these advances have opened new doors for how problems are addressed and the resulting quality of the new solutions. The use of Computer-based methods for General Movement Assessment has already been proven to assist in early detection of cerebral palsy, enabling earlier treatment that can reduce the effects the disorder has on the affected infant.

We propose a method for predicting an infants risk of developing CP based on movement information extracted from raw video recordings by a Human Pose Estimation model. Taking advantage of attention based techniques such as Multihead Attention we are able to visualize what parts of a recording are deemed most important when generating a prediction.

The proposed base model is able to achieve a F-measure score of 0.7206, suggesting that it is able to learn patterns in the movement data related to CP. Experiments show that the model have issues converging and is unstable, expressing the need for more research before it could be considered an essential part of a computer-based system for early CP detection.

Sammendrag

Dyp læring har i de siste årene oppnådd gode resultater innen forskningsfelt som datasyn og menneskelig aktivitets gjenkjenning. Innen medisin har disse gjennombruddene åpnet nye dører for hvordan problemer blir løst og den resulterende kvaliteten på de nye løsningene. Bruk av datamaskinbaserte metoder for General Movement Assessment har allerede vist seg å legge til rette for tidlig deteksjon av cerebral parese, noe som muliggjør tidligere behandling som kan redusere omfanget av påkjenninger og lidelser det berørte barnet kan oppleve under oppveksten og resten av livet.

Vi foreslår en metode for å predikere risikoen et spebarn har for å utvikle CP basert på bevegelsesinformasjon hentet ut fra videoopptak av en Human Pose Estimation modell. Ved å benytte oss av attention baserte teknikker innen dyp læring som Multi-head Attention, er vi i stand til å visualisere hvilke deler av videooptaket som anses som viktigst når det kommet til genereringen av prediksjonen.

Den foreslåtte basismodellen er i stand til å oppnå en F-score på 0,7206, noe som antyder at den er i stand til å lære mønstre i bevegelsesdata relatert til CP. Eksperimenter viser at modellen har problemer med å konvergere og er ustabil, noe som gir uttrykk for behov for videre arbeid på modellen før den kan betraktes som en vesentlig del av et datamaskinbasert system for tidlig CP-deteksjon.

Preface

This thesis is written by Martin Vold and submitted to the Norwegian University of Science and Technology and concludes a five years Master of Science degree in Computer Science.

During my time working on this Master's project I have addressed a problem faced by the InMotion research group, a group made up of clinicians and researchers from St. Olavs University Hospital and the Norwegian University of Science and Technology. The main supervisor of this project has been Associate Professor Heri Ramampiaro at Department of Computer Science. Associate Professor Espen Alexander F. Ihlen and PhD Research Fellow Daniel Groos, both at Department of Neuromedicine and Movement Science has been supervising as co-supervisors together with Dr. Lars Adde at Department of Clinical and Molecular Medicine.

Acknowledgement

I would like to start of by expressing my gratitude to my supervisor, Associate Professor Heri Ramampiaro, for giving me the opportunity to work on this project and for his support, guidance and encouragement during this whole project.

I'm also grateful to my co-supervisors, Associate Professor Espen Alexander F. Ihlen and PhD Research Fellow Daniel Groos, for their provided expertise and follow up throughout all stages of the project. I would also like to thank Dr. Lars Adde for his insight and feedback on cerebral palsy.

A special thanks to my co-supervisor in Switzerlnad, Professor Rudiger Urbanke at École Polytechnique Fédérale de Lausanne, for the opportunity to work with him and write my thesis from Switzerland, this made my last semester one to remember.

Lastly, I would like to thank my friends and family for supporting me throughout this project and my five years at NTNU.

Contents

Figures

Tables

Code Listings

Chapter 1

Introduction

1.1 Motivation and Background

Cerebral Palsy (CP) is a well recognized neurodevelopmental condition beginning in early childhood and persisting throughout the patients lifespan [1]. The risk of developing CP is particularly high for infants born prematurely. In a study from 2010 it was reported a prevalence of 9% in infants born between 24-32 weeks of gestation, with varying degree of motion impairment [2].

Diagnosing CP is a challenging problem as current techniques are complex and either dependent on expensive equipment, highly trained and specialized clinicians or both and resources like these are not available to everyone. One such technique is the analysis of an infants movement of the arms, legs, head and trunk for specific patterns and characteristics that could indicate a healthy infant. An absence of these movements could indicate that the infant has a neurodevelopmental disorder, such as cerebral palsy [3]. This process of analysing infants movements are performed bedside for up to 15 minutes with infants as young as two months old, postterm, and can also be performed remotely. As the number of clinicians with specialized training and knowledge about this technique are limited, a fully automated system for predicting infants risk of CP based on their movements would open up new possibilities for remote evaluation.

First of all, such a system would aid clinicians in making an early diagnosis which in turn would lead to infants receiving earlier and more efficient treatments. The younger a patient is when receiving treatment the more effective it will be since the brain has the highest plasticity when young, thus being more responsive to the treatment and adapt to the damages easier. Secondly, the system would be accessible to more people as detection of CP would be done in the comfort of ones home rather than at a hospital. Consequently, this would free up a lot of time for clinicians as few people have to visit them for such a procedure, letting their competence and focus go to the patients that needs it the most. Finally, this is a less stressful and non intrusive way of diagnosing CP, both for the parents and the infants, as it does not involve using machines for CT scans, MRI or other forms of tests.

1.2 Problem Statement

Currently there are no fully automatic computer-based methods for early detection of poor development of an infants neurosystem and the traditional methods that are available are either intrusive, costly, time consuming or only accessible for a small part of the population. Efforts that have been made to construct such a system are often not reliable enough, depends on human influence or carefully constructed features by experts with substantial knowledge of the field. Techniques that have been applied to the CP prediction step are traditional statistical methods such as regression models or methods from artificial intelligence and machine learning, e.g. support vector machines [4].

In the last decade, deep learning has been shown to be a powerful tool with performances that compete with and, more often than not, surpasses its traditional counterparts regarding the methods precision and efficiency. It is because of this that we are witnessing a rise in the application of deep learning in different medical domains. However, deep learning is mostly applied when images from x-rays and CT, to name a few, are used for further medical treatment and diagnosis and there exists appropriate amounts of data or the data is easily collected.

The main goal of this project is to make use of the capabilities of deep learning and create a model for the prediction step in a end-to-end deep learning system. This system will be based on video recordings of the subjects and the prediction step will use movement data of the subjects bodyparts extracted from the recordings as a basis for its generated output. This system will let neural networks handle processes such as feature extraction, embedding and everything that is needed to produce an output of the subjects risk of developing CP and should aid clinicians in diagnosing the disorder. In addition, such a method could be of great use and serve as inspiration for the development and research of similar methods to be used in other medical domains dealing with similar conditions or in problems related to the elderly or even athletes.

1.3 Research Questions

The goal of this master thesis is to develop a classification system for an infants risk of developing CP by leveraging from a previous works success in extracting skeletal joint data from raw video recordings. A deep learning model inspired by recent advances in the field was developed and tested on the already generated datasets. To make the model more explainable, different ways of presenting the steps leading up to the prediction was explored as the more understandable information the model can provide to its user the more information could be used in the diagnosis process leading to more accurate diagnoses.

With these goals in mind, the following research questions will be addressed in this thesis:

RQ1: *How can deep learning be used to solve the problem of predicting an infants risk of developing Cerebral Palsy by using human skeleton time series data?*

RQ2: *How can said model aid the users in their evaluating of a patient? Can it recognize the patterns of movement associated with Cerebral Palsy and visualise it to better the users understanding of the prediction, thus making the model usable for medical diagnosis?*

RQ3: *What are the advantages and disadvantages of such a model? Does it compete with today's methods of predicting Cerebral Palsy, such as General Movement Assessment?*

1.4 Research Method

Initially, this project started with a literature study to gather information and knowledge about the state-of-the-art methods related to the problem as well as different deep learning based methods. The data to be used was then analysed and experimented on to define an appropriate representation which could be used for the classification. A model architecture was then proposed, together with several variations of it, based on an analysis of these methods and the data at hand, which was theorised to suit the problem statement. Then a quantitative study was performed where experiments are carried out on the model and its variations such that a comparison based on their performance can be acquired. The project is then concluded from the results of these experiments.

1.5 Context

This project is part of the InMotion project, a larger research project at St. Olav University Hospital in collaboration with the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway. It consists of researchers and clinicians from the Department of Neuromedicine and Movement Science, the Department of Clinical and Molecular Medicine and the Department of Computer Science. The overall goal of the project is to create a fully automated system for predicting an infants risk of cerebral palsy based on its spontaneous movements. This prediction will only be based on a video recording of said infant and be able to be captured by the cellphones of the parents. Hence making this CP indicator available to the masses.

1.6 Contributions

This thesis provides the InMotion team with valuable insight in the task of predicting a infants risk of CP based on skeletal data using a Deep Learning Model. This developed model will make up the second part of a two part system for predicting CP based on raw video recordings. A second contribution is the ability to visualize which parts of the recorded video are the ones the method bases its prediction on. This visualization should aid users of the system in their diagnoses, but should not be considered a diagnosis. In this way the thesis will contribute to the medical field and the field of Deep Learning.

These insights are not only valuable for the InMotion team, but also next years Master students at the Norwegian University of Science and Technology as they will be tasked with continuing the development of this automatic CP prediction system.

1.7 Thesis Outline

The thesis is structured as follows:

- **Chapter 1: Introduction** introduces the motivation and background for the project, as well as the research questions and methods. The projects contributions and relation to the InMotion research initiative it is involved in are also explained.
- **Chapter 2: Theoretical Background** establishes the theoretical aspects that are relevant for the project. An introduction to the medical domain relevant to cerebral palsy is included, as well as concepts in deep learning and human activity recognition.
- **Chapter 3: Previous Work** discusses the previous work within the InMotion project, as well as state-of-the-art approaches and model architectures related to the problem being addressed in this thesis.
- **Chapter 4: Method** describes the proposed method and the methodology for predicting the risk of cerebral palsy in infants.
- **Chapter 5: Results** presents the results obtained from the experiments conducted in th cores of/throughout the project.
- **Chapter 6: Discussion** Contains the final discussion where the results are evaluated and the performance of the model is assessed in light of of its applicability for medical use.
- **Chapter 7: Conclusion and Future Work** concludes this thesis and provides thoughts and suggestions for future work.

Chapter 2

Theoretical Background

This chapter will provide some of the theoretical background needed for this thesis. First, cerebral palsy and a technique for detecting infants with a high risk of this condition is defined. Then an overview of deep learning and some of its concepts is given, before theory relevant for time series classification and human activity recognition is introduced.

2.1 Cerebral Palsy

Cerebral Palsy, or CP for short, is an umbrella term covering a group of non-progressive, but often changing motor impairment syndromes secondary to lesions or anomalies of the brain arising in the early stages of development [5]. The diagnosis is given to around 2 out of every 1000 live born children, but this prevalence rise to 8% and above for infants born 32 week or earlier after gestation [2, 6, 7]. Typical manifestations of CP include various degrees of motor dysfunction, lack of dexterity, musculoskeletal issues, speech difficulties and mental impairments.

Damages suffered by the brain during birth, pregnancy or after birth are usually the reason for the development of CP. Although CP cannot be cured, its outcome can be improved by early detection and the proper treatments, giving the patient fewer difficulties growing up. As a child's brain is constantly changing and developing it is difficult predicting the outcome of any treatment. Given its plasticity is the highest in the earlier years of life [3], the sooner treatment is started, the more susceptible the brain is to change, making early detection crucial.

2.2 General Movements

General Movements refer to the spontaneous movements of fetuses and infants not generated or related to external stimuli. The quality of these movements has been shown to accurately reflect the condition and development of the nervous system of fetuses and infants [8]. These movements are described as gross movements involving the whole body consisting of extension, flexion and rotations lasting from a few seconds to a minute. Though gross in nature, combined they look complex, varied and elegant. General movements shows up as early as when the fetus is 9 to 12 weeks postmenstrual age and last until the infant is around 60 weeks postmenstrual age, 20 week postterm age.

2.2.1 Fidgety Movements

Fidgety movements typically refers to the general movements that appear from 9 weeks postterm age and last until the infant is 20 weeks postterm age. They are described as small movements of moderate speed with variable acceleration of the neck, truck and limbs in all directions [9]. They are only present when the infant is awake and disappear if the infant is sleepy or crying. Fidgety movements have been shown to have a strong link to an infants risk of developing CP [10].

2.2.2 General Movement Assessment

Detection of CP should be done as early as possible in order for the treatment to be initiated as early as possible. Many common methods of detection cannot be conducted early enough as the signs of CP they are looking for are not yet present. But as explained above, general movements, or lack thereof, gives an impression of the state of the infants nervous system and its developments.

Based on this, *general movement assessment* was proposed as a method for assessing whether or not an infant produces these fidgety movements that are generally accompanied with a healthy nervous system in development [11]. An essential part of GMA is the evaluation of the quality and complexity of spontaneous movements by the means of Gestalt perception [12]. Gestalt perception does not focus on the basic and individual movements and pattern, but rather on the intensity and complexity of the whole movement repertoire of the infant. In order to be able to identify the presence or absence of fidgety movements with a high degree of accuracy, clinicians need to train on video recordings of infants to be able to recognize and evaluate the complexity of general movements.

In a systematic literature review Burger et al. [13] examined the predictive validity of general movements. They found that GMA generally shows great promise, especially on infants in the fidgety movement age, with a majority of the studies included getting above 80% sensitivity and specificity. One paper to highlight is Prechtl et al. [10] which achieved sensitivity and specificity of 95% and 96% respectively. Sensitivity is a measure for how many positive cases are correctly identified among all positive cases, given by Equation (2.1). Specificity, on the other hand, is a measure of how many negative cases are correctly identified from all the negative cases, given by Equation (2.2).

$$sensitivity = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{2.1}$$

$$specificity = \frac{True\ Negative}{True\ Negative + False\ Positive} \tag{2.2}$$

The biggest advantage of GMs is that they can be used with great success as a quick, cost-effective and non-invasive assessment method, but it has its limitations. The accuracy of the method relies on the skill of the clinician conducting the assessment. Clinicians with this kind of training might not be availed at every hospital, making them a scarce resource. Also, this way of assessing movements is highly subjective and depends on each clinicians individual interpretation, which can be affected by bias or external factors like long work days or mental presence.

2.3 Deep Learning

As machine learning is a subfield of artificial intelligence so is *Deep learning* a subfield of machine learning. It is concerned with models and algorithms that take inspiration from the structure of the brain and its functions. One could argue about who made the first contributions to this field, but the introduction of the multilayered perceptron (MLP) [14] and backpropagation (BP) [15] may be said to be the foundation that most of the field is built upon.

The constant decreasing in cost of computational power together with its increase in availability and the ever growing accessibility of data has made deep learning as in demand and popular as it is today. The fact that deep learning has significantly improved the state-of-the-art for many problems that the Artificial Intelligence and Machine Learning communities have faced, have also played a part for its increase in popularity. Deep learning is being used to achieve these results includes e.g. in fields such as Computer Vision, Natural Language Processing (NLP), Computer Graphics and Human Activity Recognition (HAR), to mention a few. For this reason we will be giving a brief introduction to neural networks and other relevant concepts in deep learning in this section.

2.3.1 Neural Networks

In literature *neural networks* and MLP are often used interchangeably. This network consist of nodes and edges connected to each other in such a way that they form a computational graph where information flows in only one direction, from the input nodes and to the output nodes. These networks are used to discover and learn patterns and relationships in the data presented to them to produce suitable outputs. This process of generating outputs $\hat{y}$ by processing the input data x in intermediate layers between the input and output layers in the network is called the *forward pass* and it consists of two different stages for each layer in the network. In the first stage each output from the nodes in the previous layer is weighted to determine how much they will contribute to the input values in the subsequent layer by utilizing weights. Then a weighted input is produced by a sum of all the weighted outputs from the prior layer. This operation can be performed and expressed as the matrix multiplication in Equation (2.3) and is often accompanied by adding a bias to the result.

$$z^{(l)} = a^{(l-1)}W^{(l)} + b^{(l)} \tag{2.3}$$

Here $a^{(l-1)}$ is the activated output values from layer $l-1$, $W^{(l)}$ is the weight matrix of layer l and $b^{(l)}$ is the bias vector of layer l.

In the second stage the weighted inputs of a layer, $z^{(l)}$, is passed through a non-linear activation function α. The fact that the activation function is non-linear is a major part of the reason to why neural networks are able to solve difficult tasks other machine learning methods struggle with. The reason the activation function should be non-linear is to make the neural network able to learn more complex non-linear patterns in the data. The sigmoid and tanh activation functions have been widely used in deep learning, but have been slowly replaced with functions such as ReLU (Rectified Linear Unit) and Softplus [16] as these do not suffer from the vanishing gradient problem.

Depending on the complexity of the problem at hand, the architecture of a neural network can be designed in several different ways to suit our needs. The network design is crucial for its performance and typical aspects of the network that needs to be decided are the number of hidden layers, number of nodes in each hidden layer and the activation function used in each node. Few hidden layers and less hidden nodes makes the network faster and easier to train, but it is not complex enough to learn the patterns in the data. Larger networks with more hidden layers and more hidden nodes can solve this problem of underfitting, but will make the model slower and more difficult to train as the process of optimization gets harder with an increasing number of parameters resulting in lower performance. Each design choice comes with its own trade-off. Nevertheless, this flexibility of the design of neural networks makes them suitable to overcome most problems where the data is able to be represented in a way the network can understand. Figure 2.1 shows an example of a neural network which generates two output values by taking three input values and passing them through one hidden layer with five nodes.

For a neural network to be able to learn, a process called *backpropagation* is used.

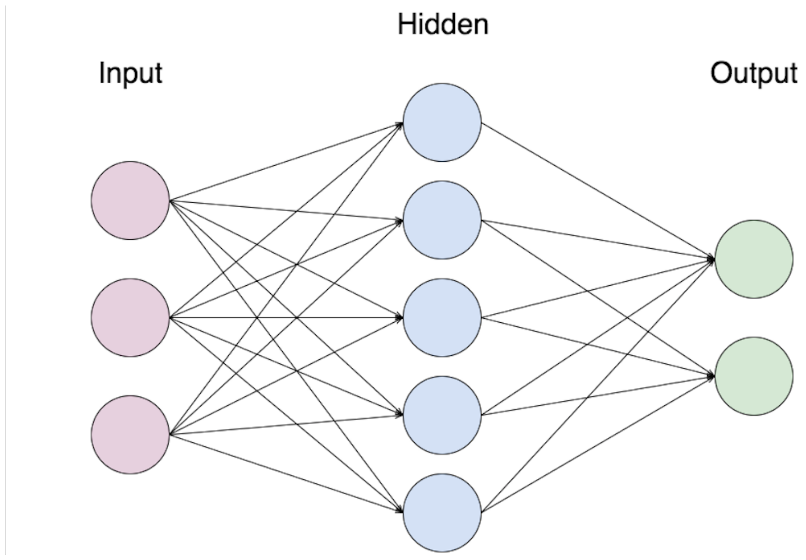


Figure 2.1: A simple fully connected neural network

Given a neural network and a loss function the backpropagation algorithm works by calculating the gradient of the loss function with respect to the weights of the neural network and then updating the weights in the opposite direction of the gradient, thereby reducing the loss of the models future output and optimizing the weights. Since a target y and a predicted output $\hat{y}$ is needed to quantify the loss, backpropagation is most useful when dealing with supervised learning, i.e. we have a target value corresponding to each input vector. During this optimization process the learnt patterns and relationships from the data will be realised in the weights between nodes in the connected layers. As these weights are updated, so are the importance of the different connections between each layer. This embedding of knowledge in the network allows it to able to solve complex tasks, but also makes it difficult to comprehend exactly why it behaves as it does when presented with an input vector. When optimized, a neural network should be able to generalize upon the data it has been presented with, but for this to be possible the data should be of such diversity that the network is able to experience as much of this domain as possible. As such, many examples are required for the data to be able to represent this diversity.

The optimization algorithm described above is most commonly know as Stochastic Gradient Decent (SGD) [17] and is widely used. Another popular optimizer is the Adam optimizer [18]. It computes adaptive learning rates for each parameter that are based on the exponentially decaying average of past gradients and squared gradients. Specifically, the algorithm calculates an exponential moving average of the gradient and the squared gradient. Adam is a competitor to SGD as it is shown to converge effectively and achieves good results fast.

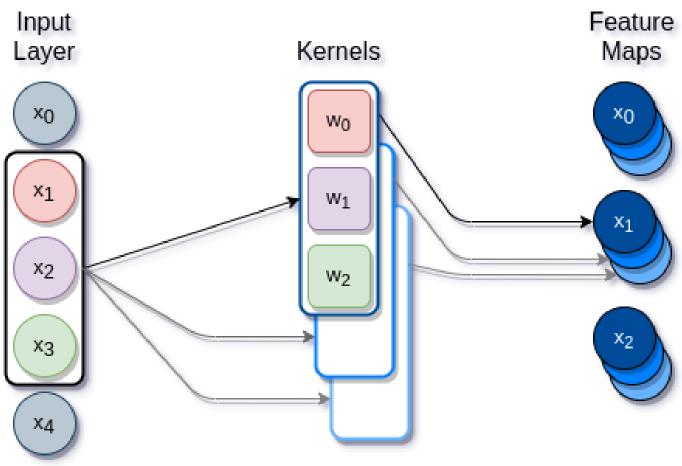


Figure 2.2: A simple 1 dimensions convolutional layer

2.3.2 Convolutional Neural Networks

Convolutional neural networks (CNN) are neural networks that works particularly well when dealing with signal and image data. The idea for these neural networks goes all the way back to 1980 and a proposed computational model for visual pattern recognition called the neocognitron model [19] and gained a surge in popularity in 2012 when the winner of the ImageNet challenge achieved state-of-the-art performance with a CNN called AlexNet [20]. In this section we will be focusing more on the CNNs for signal processing.

CNN differs from regular neural networks in the way that the input values for the next layer is computed during a forward pass. A node is connected to a subset of the nodes in the previous layer and thus only receives information from these nodes. The region from which a node receives its inputs is referred to as its receptive field. This local connection between layers makes an important concept of a convolutional layer possible, namely weight sharing. Weight sharing makes use of the same set of weights to calculate the output of a convolutional layer based on the receptive fields. This output is called a feature map as one set of weights only look for one specific feature or pattern and one set of weights is referred to as a kernel. To be able to detect more than one pattern in a signal it is common to use multiple kernels, this will in turn create multiple feature maps and increase the dimensionality of the signal. This process is illustrated in Figure 2.2 where three kernels of size 3x1 generates three feature maps of size 3x1. CNNs usually comes with a pooling layer following the convolutional layer. This is to reduce the resolution of the feature maps and is effective in the sense that it stores information by using less space. The most common forms of pooling layers are max pooling and average pooling and both highlights different aspects of the feature maps.

CNNs are well suited for deep learning as the weight sharing of the convolutional layers and reduction of information by the pooling layers lets them scales quite good as they require fewer parameters than their fully connected counterparts as fewer parameters makes them easier to optimize.

2.3.3 Transfer Learning

As previously stated, for a neural network to be able to learn properly it has to be presented with a divers set of examples representing the variety of the domain. This makes the performance of deep learning strongly dependent on large dataset, which in turn creates a problem if the training data is insufficient. Collecting the data needed can be complex, expensive and time demanding making it difficult to a large high quality dataset. The idea behind *transfer learning* is to transfer the already learnt knowledge form a deep learning model, trained on a *source domain*, to a new related problem and re-purpose it for the benefits of improvement in learning and generalization. The data domain of the related problem is called *target domain*. This will allow for a significant reduce in training time and the demand for training data by taking advantage of already existing models and datasets.

Reusing a part of or the whole network structure in conjunction with its weights is referred to as network-based transfer learning [21]. A model whose weights and structure is used for transfer learning is know as a pre-trained model. When reusing parts of such a pre-trained model its common to use earlier layers of the model as a starting point for the new model. This can be viewed reusing the pre-trained model as a feature extractor and is useful as deep learning models tends to learn features which can be beneficial across multiple domains. Huang et al. [22] divided their neural network into two parts where the first layer acted as a language independent feature extractor and the last layer as language dependent classifier. This feature extractor was then trained on the whole domain, thus containing knowledge from all the languages, while the classifiers where only presented with the language specific domains.

During training the weights of a pre-trained model can be set to be fixed or just used as initial weights. The decision of whether to tune these weights together with the rest can depended on how similar the domains are or how large the target domain is. Earlier layers of the pre-trained model can aid in the feature extraction from the target domain. If there are few examples from this domain, keeping the earlier layers fixed while tuning the later or new layers might be the best strategy, as the earlier layers of a deep learning model contains more generic features and later layers progressively contain more specific features of the classes contained in the source domain [23]. Fine tuning can still be performed with this approach, but this should be done during a separate tuning step as the parameters of the feature extractor must be adjusted very precisely.

2.3.4 Attention Mechanisms

There exists multiple ways of trying to explain how and why CNNs works and how they perceive images presented to them [24]. These approaches are quite successful when dealing with 2d convolution and images, as images are more often than not trivial for humans to understand and interpret, but not as good for sequential data. When using recurrent neural networks (RNN) for machine translation the problem of capturing long range dependencies arose as information about earlier tokens in the source sentence had to accumulate in a fixed-length hidden state vector, generated by an encoder, which in turn would be used to generate the target sentence by a decoder. As the hidden state vector receives new information about later tokens in the sentence it has to give up some of the information about earlier tokens.

To solve this an *attention mechanism* [25] was proposed to help each predicted token in the target sequence selectively attend to all tokens in the source sequence. In other words, this mechanism allows each new token generated search for a set of position in the source sequence where the most relevant information is concentrated. This freed the model from having to solely base its predictions on a fixed-length hidden state vector regardless of the length of the source sequence, but it also added a computational increase to the model. As an additional benefit to this mechanism, the model was now able to show what parts of the source sequence it directed its attention to and considered more important when generating to target sequence through its global attention weights. Figure 2.3 shows the attention matrix for a machine translation of an English sentence to French. When translating the words *European Economic Zone* its is clear that the model is able to apply its attention to these words in a way that represents the way it is written in French and align its context vector to represent the importance it has given the source sentence. The attention matrix shows that this mechanisms greatly enhances the interpretability of the inner workings of the model and how it produces its predictions, a quality that is very desirable in deep learning.

For the attention mechanism to be able to know which parts of the source sequence to attend to, a context vector needs to be generated. The context vector is generated as given by Equation (2.4). Here the context vector $\mathbf{c}_t$ for token t in the target sequence is computed as a weighted sum of the source sequence, with a length of n tokens.

$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} \overline{\mathbf{h}}_i \tag{2.4}$$

The attention weights of $\boldsymbol{\alpha}_t$ is computed by Equation (2.5) which is a softmax of some predefined alignment score function, score($\mathbf{h}_t$, $\overline{\mathbf{h}}_s$). The alignment score function computes how well a hidden state $\mathbf{h}_t$ and a token from the source sequence $\overline{\mathbf{h}}_s$ match.

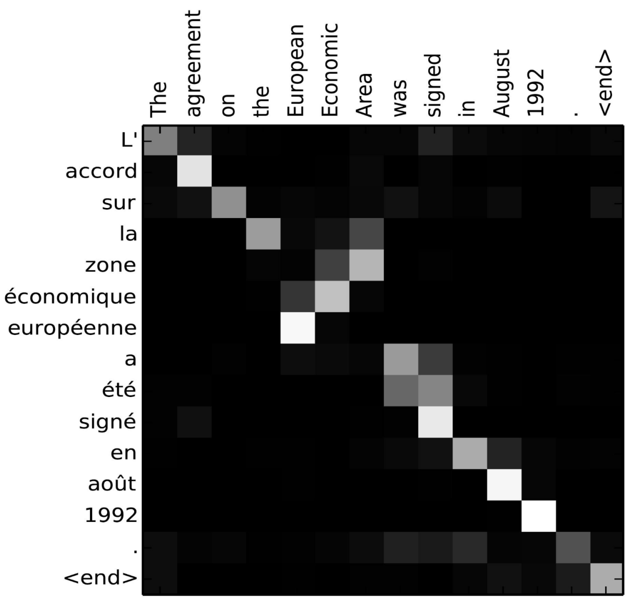


Figure 2.3: Attention matrix for a English to French translation

$$\boldsymbol{\alpha}_t = \text{align}(\mathbf{h}_t,\, \bar{\mathbf{h}}_s) = \frac{\exp(\text{score}(\mathbf{h}_t,\, \bar{\mathbf{h}}_s))}{\sum_{s'} \exp(\text{score}(\mathbf{h}_t,\, \bar{\mathbf{h}}_{s'}))} \tag{2.5}$$

In other words, the attention mechanism attends to the source sequence based on how similar the alignment function finds each of the source tokens to be to the target token ts hidden state. Figure 2.4 illustrates this process.

Bahdanau et. al [25] proposed an alignment function based on a neural network which is jointly trained together with the rest of the models components. This function can be mathematical described as Equation (2.6) where $\mathbf{v}_a^T$ and $\mathbf{W}_a$ are network weights.

$$\text{score}() = \mathbf{v}_a^T\, tanh(\mathbf{W}_a[\mathbf{h}_t; \overline{\mathbf{h}_s}]) \tag{2.6}$$

Other alignment functions have been proposed, such as the dot-product score and general score by Luong et al. [26]

Figure 2.4 shows an attention mechanism which takes the whole source sequence in to consideration when computing the attention weights. This is known as global or soft attention. Two alternatives to this procedure is hard and local attention. The former was proposed by Xu et al. [27] and was used for machine generated captions from images where a patch of the image are selected and attended to at a time. The later was proposed to take advantage of hard attentions less expensive computational cost and be easier to train. This is achieved by selecting an aligned

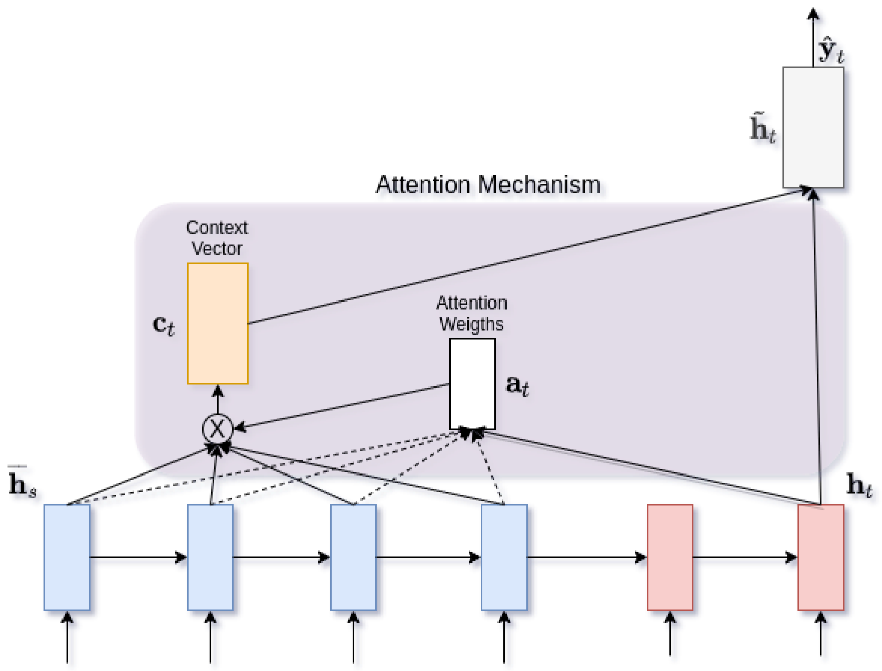


Figure 2.4: Attention mechanism

position p_t and generating the context vector from all source tokens a distance D from this point. The simplest way of choosing the aligned point is by setting it equal to the target tokens position in the sequence, $p_t = t$. Note that a predictive way of choosing p_t exists and is explained in [26].

All attention mechanisms explained so far has been using a source and a target sequence which were inherently different when aligning the source tokens to a target token. Removing the target sequence and making the source sequence attend to itself is the basis behind self attention. A key idea here is to use attention to include the relationships between tokens in the same sequence. This concept was proposed by Cheng et al. [28] for their paper on machine reading.

2.4 The Transformer

Attention mechanisms as described in the section above are limited to being utilized together with recurrent neural networks. While attention aided with the challenge of capturing long range dependencies, the underlying sequential nature of these networks still gave rise to challenges during training and optimization. Due to how RNNs process one element at a time they are slow to train and causes them to struggle with vanishing and exploding gradients [29].

To solve these problems Vaswani et al. proposed the *Transformer* [30], a model that entirely relies on an attention mechanism and which takes the whole sequence as input, removing the sequential aspect RNNs are restrained by. The ability to not have to handle the input in a sequential matter allows for more parallelization which in turn lowers the time needed for the model to converge. A consequence caused by feeding the whole sequence to the model at once is that it looses the notion of each elements relative position in the sequence and the order of the sequence. As the order of elements in sequential data matters, information about this ordering must be added to the sequence in order for the model to be able to make use of it. This is achieved by adding positional encodings to the input sequence before it is presented to the model, Equation (2.7).

$$\begin{aligned} PE_{pos,\,2i} &= sin(pos/10000^{2i/d_{model}}) \\ PE_{pos,\,2i+1} &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{2.7}$$

The idea behind the positional encoding is that by adding these vectors with a specific pattern it will aid the model in determining the sequence order of elements and the spatial difference between elements, i.e. the distance between the two in the sequence. This can be viewed as adding a binary number of the same dimension as the embedded input to the elements and increasing it by one for the next element, but rather than being discrete values the sinusoids in Equation (2.7) produce continuous values which are contained in the interval [-1, 1]. The positional encoding has the same dimension as the embedded sequence so that it can be generated beforehand and added to the input embedding. Keep in mind that this positional encoding is a fixed one and that learned positional encodings exists. Fixed positional encoding have an advantage over learned ones as they allow the model to extrapolate to sequence lengths longer than the ones encountered during training.

Figure 2.5 illustrates the transformer model and give a great overview of its architecture. The transformer is made up of two components, an encoder and a decoder. Both of these components attention mechanisms is the multi-head attention, which consist of multiple scaled dot-product attention units.

2.4.1 Scaled dot-product attention

The input to the scaled dot-product attention, Equation (2.8), consists of queries (Q) and keys (K) of dimension d_k and values (V) of dimension d_v, where the attention weights are given by an inner product of Q and K. Every element in Q is matching every element in K. Since a large sequence would perform many inner products which could grow to be large in magnitude, the resulting weights are scaled by the square root of the number of dimensions in the embedded input. This is to prevent the softmax function to get pushed into regions where the input value is really large or very small and the gradients are extremely small. After applying the softmax an output is produced by performing an inner product

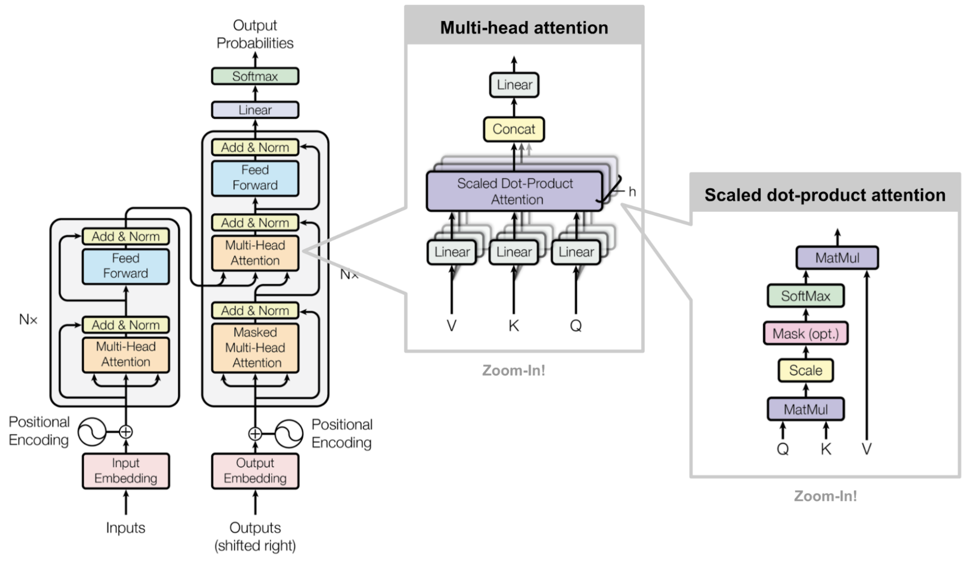


Figure 2.5: Transformer architecture

with V, thereby routing information from QK^T to V of every element attended to. This can be viewed as Q and K constructing the relationships in the sequence and V summarizes all of these relationships to produce a output reflecting the relationship between one element and all others.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK_T}{\sqrt{d_k}})V \tag{2.8}$$

2.4.2 Multi-Head Attention

The relationship between two elements in two different sequences might be very different based on the location of the two in the sequence and the distance between them, here the two elements are thought to be the same in the two sequences. In other words, there might exist multiple different relations between elements and these relations should not be treated equally. Multi-head attention, Equation (2.9), tackles this problem by creating multiple attention matrices in parallel in different heads h.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.9}$$

Each head attend to information from a different representation of the sequence, where $\text{head}_i = \text{Attention}(QW_I^Q, KW_I^K, VW_I^V)$ and $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{d_{model} \times hd_v}$. Since each attention head outputs an attention matrix the result of concatenating all of them needs to be reduced to match the original shape of the input. This is done by a linear projection with weight matrix W^O. The weight matrices to be tuned together with the rest of the model during optimization are W_i^Q, W_i^K, W_i^v and W^O. The memory and time complexity of the multi-head attention computation is $O(n^2 \cdot d)$, where n is the length of the sequence and d is the number of dimensions per element.

The multi-head attention layer does not learn representations based on all other elements in the sequence, it is solely a way to weight each element by its similarities to other elements. To create a new representation of the input embeddings weighted by the similarity score a feed-forward network is used. Its output is the new embeddings which are passed to the subsequent layer.

The multi-head attention component and feed forward component, which is made up from two linear layers increasing the dimensionality of the sequence to d_{ff} for then to reduce it back to d_{model} again, makes up the two sublayers used in the transformer. After each of the sublayers a residual connection is made which adds the input to the sublayer to its output, this in turn makes the training of deep learning models easier [31]. Subsequently layer normalization is performed, which normalizes the inputs on a per datapoint basis over all features [32]. At last dropout is performed for regularization purposes before the residual connection and normalization layer.

All of this combined lets the transformer find three types of relationships in the data presented to it. These are the relationships inside the source sequence, the relationships between the source and target sequence and the relationships inside the target sequence. In comparison, the attention mechanisms explained in Section 2.3.4 are only able to find relationships between the source and target sequence.

While the transformer is made up from an encoder and decoder when used on sequence to sequence problems, the encoder alone could be used for transforming an input sequence and using only the relationships found within itself, e.g. for question answering [33, 34]. It has also been shown that transfer learning is a technique that possesses a lot of potential when used for deep learning models adopting the transformer architecture and fine tuning these pre-trained models can yield very successful results [34, 35].

2.5 Human Activity Recognition

Human Activity Recognition, HAR, can be defined as the task of classifying sequential data from a given set of predefined actions and it is a part of the field of time series classification. The data used for time series classification share the same main characteristic as data used in NLP and speech recognition tasks, namely the sequential aspect, hence resulting in these fields being closely connected. The data is typically collected using sensors captured by video cameras, sensors carried by the subjects such as gyroscopes and accelerometers or both. There are three main problems that makes HAR a challenging problem. The first problem is that the use of these kinds of low-level sensor data poses a lot of challenges as there is no obvious way to extract high-level knowledge about human movements from them [36]. Furthermore, the recorded data also consists of variations as each action performed by a subject may be performed with a lot variations from recording to recording. Lastly the sheer volume of data that needs to be collected and processed poses a problem itself, sensors or cameras capture data with a frequency of around 30Hz or higher so a 30 second recording can easily be composed of 900 or more elements.

If a model is tasked with classifying multiple actions within a single recording we are not only concerned with a recognition problem, i.e. what action is preformed, but also with a localization problem, i.e. where in the recording are the different actions acted out. Human activities can be categorized into six different categories [37]. These categories are (1) gestures, (2) atomic actions, (3) human-to-object or human-to-human interactions, (4) group actions, (5) behaviors and (6) events. Movements corresponding to a high risk of CP would be classified as human behavior as this category refers to physical actions that are associated with emotions, personality and the psychological state of the subject.

As an activity can be composed of more than one movement and last for anywhere from a couple of seconds to multiple minutes, the case of classifying a sequence of movements as indicating a high risk of CP can be considered a HAR problem.

2.5.1 Deep Learning Based Approaches

Conventional machine learning approaches are based on fixed size windows and machine learning which requires hand crafted features. This kind feature engineering is difficult and an expertise in the field is vital for coming up with favorable features. As deep learning requires little to no feature engineering and is able to learn patterns and relationships in data, it is a natural competitor to traditional approaches.

Deep learning based HAR networks are often based on convolutional or recurrent neural networks or a combination of both [38]. Sensor data from body worn sensors or from tracked skeleton joints from video recordings are able to utilize the self-attention mechanism from Section 2.4. Since a goal of this project is using skeleton joint data extracted from video recordings to predict the risk a subject has of developing CP, the HAR approach to time series classification is well suited. Self-attention will allow the network to provide meaningful insights to clinicians, which is a requirement for intelligent computer-based systems being used to aid medical diagnoses.

Chapter 3

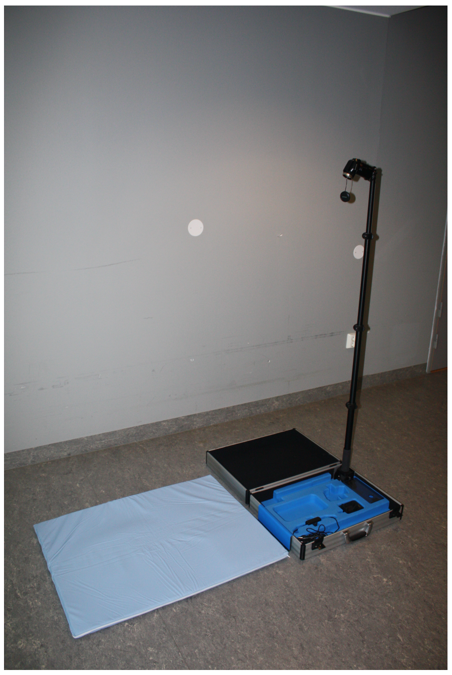
Previous Work

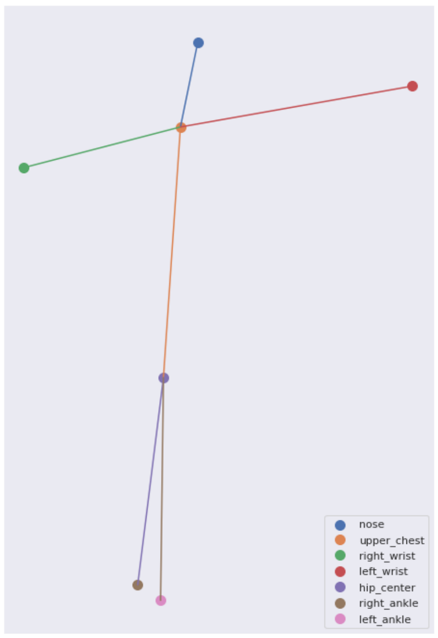
The upcoming chapter will show how ideas from HAR and GMA are used and incorporated in existing computer-based methods designed to solve these two problems. Section 3.1 will give an overview of the progress made by the InMotion team. Following this, Section 3.2 presents existing methods for computer-based GMA and the challenges they have faced. This section is then ended with Section 3.3 which provides an overview of deep learning methods used for HAR.

3.1 InMotion

InMotion is an ongoing research project at St. Olav's University Hospital led by Lars Adde. As described in Section 2.2.2 conventional GMA approaches has its limitations, mainly the need for trained clinicians and the subjective nature of assessment method. InMotion aims to overcome these challenges by utilizing computer-based GMA for CP risk assessment based on video recordings of patients. Adde et al. [39] have already showed that it is feasible for the prediction of CP to be provided by computer-based video analyses and that these video recordings can be used for qualitative and quantitative analysis of FM [40]. Such a method would make GMA available to people in places where there are no trained professionals and it would aid clinicians with their diagnoses, as the system would provide valuable information. Since the prediction of this system will be computer-based, clinicians should not consider it a diagnosis as the model do not provide valid clinical reasons for the predictions, but rather consider it as part of the results that forms a diagnosis together with his own observations and results from other clinical tests.

As this system would be based on video recordings, a set-up for the recording process was created. This set-up, shown in Figure 3.1a, included a camera with a stand, a mattress and a suitcase for storage was then distributed to different hospitals around the world, including hospitals such as University of Chicago Medical


(a) Recording set-up fully deployed



(b) View of the joints tracked in CIMA-19

Figure 3.1: Set-up and view for video recordings of infants

Center (UoC) and Life Care Hospital in Indore, India (LCH), were they have collected recordings of both low- and high-risk infants.

In 2018 Aurlien and Groos [41] developed Human Pose Estimation (HPE) model, called CIMA-Pose, which would track seven points on an infants body as part of their master thesis, visualized in Figure 3.1b. This would be used as a first step in the process of predicting CP by extracting the joint location of the subject to be used further for Computer-based Infant Movement Assessment and CP prediction. The generated skeleton data from this model would be collected into the CIMA-7 dataset consisting of 513 recordings in total. Groos et al. [42] would in 2020 improve upon this work by introducing EfficientPose a model which outperformed OpenPose [43], the most commonly used HPE method used in real-world application, in terms of accuracy and computational efficiency. The data generated by EfficienPose would result in the CIMA-19 dataset, a dataset made up of 377 recordings, each with 19 bodyparts tracked.

3.2 Computer-Based GMA

There have been developed multiple different approaches for performing computer-based GMA designed to overcome the challenges present by common methods, by only considering video recording of the patients. One of these approaches are the previously mentioned method of Adde et al. [39, 40] where information of the infants movements are extracted through a process called frame differencing. With frame differencing movement information is represented as motion images. These motion images are the difference between a frame and the succeeding frame and represents the change of each pixel between two frames. Several quantitative measures are calculated from these motion images, e.g. the mean and standard deviation of the two measures quantity of motion and centroid of motion which are the mean of pixels that changed and the center point of movement respectively. The centroid of movement were found to identify infants with fidgety movements with a sensitivity and specificity of 84% and 71% by comparing its standard deviation with a threshold. Low standard deviation values for the centroid of motion may reflect all the small movements of the whole body, thus supporting this definition of fidgety movements. Although being a simple approach for computer-based GMA it has its drawbacks. Its results are dependent on the quality of the video recordings as noise such as a change in the lighting, movements other than the ones the infants produces or occlusions will give motion images with areas not related to the infants movement highlighted. This was partly solved by the use of filters such as a low pass filter.

A second method proposed by Rahmati et al. [44] utilizing tracked motion trajectories as a foundation for representing the motion data. These trajectories are tracked with a motion segmentation based tracker which computes these trajectories based on the average of all trajectories in one segment of the infants body [45] between consecutive frames of the recording. The trajectories contained in each segment are obtained using optical flow which extracts information about the speed and direction of objects in the recording. The final trajectories are comparable to the trajectories found in the CIMA datasets as they represents the movement of different bodyparts. One downside of this approach is that the final trajectories are generated iteratively, hence the need for manually initializing the point of a trajectory. Following this step, frequency-based features are then extracted by applying a fast Fourier transform (FFT) on the trajectories which results in the magnitude of the frequency components which will be presented to the following CP prediction step, which is performed by partial least square regression (PLSR). This approach of predicting CP gave a sensitivity and specificity of 86% and 92%. In addition to the already mentioned downside of having to manually initialize the start of each of the final trajectory the calculation of the optimal flow introduces a computational expensive cost to the method.

These two approaches handles the movement extraction and CP prediction in very different ways, resulting in different performance and complexity. Yet, common for both of them is that these approaches consist of two different steps to complete the whole process. Firstly, the movement data are extracted from the raw video recordings. This step is then followed by a CP prediction process which bases its predictions on the movement data generated by the first step. This first step is already been created and thoroughly tested by the InMotion team. With the skeleton data generated by CIMA-Pose and EfficientPose a natural next step for reaching InMotions goals would be to develop an automatic CP classifier for the second step of the Computer-based Infant Movement Assessment.

3.3 Deep learning-based HAR

Traditionally, CNNs and RNNs have been the two main neural networks used when it comes to Time Series Classification (TSC). They are also frequently used for Human Activity Recognition and given the strong similarities between the two, it is natural that a majority of HAR methods are based on the same architectures and ideas as TSC.

CNNs are used for capturing temporal signal structures from sequential data within a given window of the sequence with a low memory overhead as the weights between all windows are shared. Zeng et al. [46] developed a model using CNNs to capture the local dependencies in signal data captured with body-worn sensors from datasets such as Opportunity [47] and Skoda [48]. They introduced a weight sharing scheme where the traditional scheme of CNNs are relaxed such as the range of the weight sharing and the window is affected, naming it partial weight sharing. Separate input convolutional and max pooling lasers whose outputs where concatenated before fed to the classification layers was also used, i.e. each axis of the signal data. The combination of these additions to the CNN architecture resulted in their model outperforming the then current state-of-the-art approaches.

Yeng et al. [49] also leveraged from the automatic feature learning capabilities of CNNs to produce state-of-the-art results on the Opportunity dataset. By following alternating convolutional and max pooling layers with a parametric-concatenation which feed its output to a classifier they managed to achieve an accuracy of 87.7%, 83.0% and 86.7% for subject 1 through 3 respectively. The parametric-concatenation layer is a fully connected layer for unifying the feature maps created by the convolutional layers. Both Yeng and Zengs results illustrates CNNs potential to learn feature extraction when dealing with time series data.

One approach using RNNs for HAR was proposed by Murad et al. [50]. By using RNNs, specifically LSTMs, on a window of the sequence they generated an activity prediction for each time step of the data which is then aggregated in order to produce a prediction for the data within the window. This approach yielded a result of 92.0% F1 score on the Opportunity dataset. Despite these promising results the application of pure RNNs for HAR has been more limited than CNNs. It is more common to see the two used in conjunction (? together) than alone benefiting from CNNs potential to learn features and RNNs ability to combine temporal information. The CNN will in this case act like a feature extractor, much like it did for Ordonez et al. [51]. They attached a LSTM network to a CNN and let it interpret the extracted features over several time steps. They also removed the pooling layers that usually accompany every CNN, stating that it interfered with the CNNs ability to extract features as it downsamples the data sequence. With this setup a F1 score of 91.5% was achieved on the Opportunity dataset.

3.3.1 Attention based HAR

Early approaches from deep learning-based HAR based their prediction on the ability of CNNs to perform feature extraction from sequences of movement data. Because of this the temporal context used during feature extraction is fixed. This is not ideal as the activities are of different lengths and some might fit within the context window, while others might not. The publication of Vaswani et al. [30] *Attention Is All You Need* has inspired more recent models to incorporate self-attention, as described in Section 2.4, after the feature extraction step allowing the model to consider and attend to all time steps of a sequence of data [1].

One such model was introduced by Murahari et al. [53]. They constructed an attention model for HAR by adding attention layers to the model proposed by Ordonez et al. [51] which consisted of CNN and LSTM layers. Hence, instead of using the LSTM cells outputs to perform their prediction they would be fed into the attention mechanism, whose output would be the basis for the classification. One key difference between this attention mechanism and the one proposed by Vaswani is that instead of letting every element in the sequence attend to every other element, they restrict the attention span of one element to the 7 elements that came before it in the sequence. This creates a form of masking effect not letting elements in the sequence look into the future nor too far back in the past.

Sun et al. [54] proposed a model with a fully self-attentive layer mapping a query and a set of key-value pair to an output. However, the CNN layer that is found in many of the other deep learning-based HAR methods was replaced by a LSTM network to model the sequential data. This causes the attention layer to stay put until the LSTM layer have finished its computation, this is also the case for previous mentioned model. This inherently sequential nature of the LSTM cells precludes parallelization within training examples, which becomes more critical the longer the sequences are. To combat this they impose a sliding window strategy with an overlap of 50%.

An approach exclusively based on a CNN for the feature extractor was proposed by Zhang et al. [55], overcoming the parallelization issue of the previously mentioned methods. Rather than relying on skeletal data of the subject, this method used raw images as input and the pre-trained VGG-16 network as a feature extractor [56]. as a feature extractor. Using images also allowed them to perform HAR in scenes containing multiple subjects whose action depends and affects

[1]There exists Sequence-to-Sequence models for HAR which makes use of the attention mechanisms proposed by Bahdanau and Luong too, e.g. Tang et al. TSC model [52]. Since we are focusing on self-attention this will not be explained further, but will be mentioned as inspiration for the next master students who will tackle this problem.

each other. A interpretable visualisation of the attention distributions was also developed, but is not too relevant to this project as we do not deal with image data. It is, however, a good display of how the model attends to different parts of the input sequence and objects in the scene.

None of the mentioned methods have relied only on skeletal data and taken advantage of the parallelizable properties of networks such as CNNs over RNNs illustrating the need for such a model. As the attention mechanism gave the Natural Language Processing field an easily understandable visualization for what parts of the input sequence was given the most importance, making the model output more interpretable. This visualization will get increasingly harder to interpret as the sequence length grows, showing the need for a new method of visualization in HAR if we want to gain any insight from the predictions.

Chapter 4

Method

In the upcoming chapter we propose a model for classification of time series data. The model will utilize deep learning in combination with attention to predict if an infant is at high or low risk for developing CP. We will start of by going into detail about the different datasets used to train and evaluate the proposed model. Then the model itself will be described in detail followed by proposed ablations to the model. Subsequently, the training procedure used for optimizing the model as well as data preprocessing and preparation is described. Then, a description of the different evaluation metrics and procedures are presented, wrapping up this chapter.

4.1 Datasets

To test how well the proposed model generalizes to other problems than CP risk prediction it was decided to train it on two different datasets used for Human Activity Recognition. With the model already trained on other similar dataset, it could serve as a pre-trained model to see if transfer learning could give similar performance gains as it has done in fields like computer vision.

In the area of Human Activity Recognition (HAR) there exists multiple datasets that can be used as benchmark datasets [57]. Since HAR is seldom restricted to solely using skeleton data extracted from video recordings, the datasets often contains only the video recordings of the different actions performed by the subjects. This makes most of these datasets not suitable for use in training a model with our restrictions. Luckily for us there exist datasets that have extracted the skeleton data from video recordings with Human Pose Estimation models or specialized hardware, e.g. Xbox Kinect, and presented them together for use of HAR problems like active and assisted living for the elderly [58]. These datasets have the added benefit of containing 3 axis of movement (x, y, z coordinates) compared to the two that the CIMA datasets contain (x, y coordinates), thus containing more

Dataset	Subjects	Actions	Joints	Fps	Avg. seq. Length	Size
CIMA-7	391	2	7	24/29/30	7726	513
CIMA-19	377	2	19	24/29/30	7967	377
UTD-MHAD	8	27	20	30	68	861
KARD	10	18	15	30	118	540

Table 4.1: Comparison of the CIMA, UTD-MHAD and KARD datasets. Average sequence length in frames per second.

information about the subjects poses and movements. A summary of the different datasets listed below can be found in Table 4.1.

4.1.1 CIMA datasets

The CIMA datasets consists of skeleton data containing 7 and 19 different joints, grouped into their own respective datasets that we will refer to as CIMA-7 and CIMA-19. The skeleton data for CIMA-7 has been extracted by CIMA-pose [41] where as for CIMA-19 EfficientPose [42] was used. The different bodyparts tracked by CIMA-pose and EfficientPose can be seen in Figure 4.1. CIMA-pose performance varies quite a bit from video to video. There are in essence two main problems the skeleton data from CIMA-7 suffers from;

- The model mistakenly assigns one joint to the wrong bodypart, e.g. right wrist is assigned to the left wrist
- The model is not able to assign a specific joint to any bodypart and ends up assigning it a location close to the boarder of the image or an object located close to the image boarder.

These errors can be seen in Figure 4.2 together with an example of an in particularly unlucky attempt of tracking a sequence. These tracking errors will not be dealt with when CIMA-7 is used for training, as they are nearly none existent in CIMA-19 thanks to EfficientPose and CIMA-19 will be used as the main CIMA dataset in this thesis.

Both CIMA datasets roughly contain the same subjects, but CIMA-7 has 14 subjects more making the total number of subjects 391 compared to CIMA-17's 377. CIMA-7 also contain skeleton data from multiple video recordings of the same subjects making its total size 513, where as CIMA-17 size is 377, i.e. one recording per subject. This difference of 136 sequences is a result of CIMA-7 containing multiple recordings of the same subjects, either captured during one or multiple visits to the hospital. This is not considered to have a significant impact on the data as a subject will end up with the same diagnosis regardless of when the video recordings where made, hence if a subject is associated with more than one recording all of them will be present in the dataset and not removed to only have one recording per subject.

Target	Fps	Size	Fraction
0	24	67	13.06%
0	29	356	69.4%
0	30	28	5.46%
1	24	25	4.87%
1	29	34	6.63%
1	30	3	0.58%

Target	Fps	Size	Fraction
0	24	38	10.08%
0	29	274	72.68%
0	30	24	6.37%
1	24	13	3.45%
1	29	25	6.63%
1	30	3	0.79%

Table 4.2: Distribution of target and fps values for the CIMA datasets, Non-CP cases have been given the target value 0 and CP cases target value 1. (*Left*) CIMA-7, (*Right*) CIMA-19

	Fraction	CP	non-CP
Train	0.61%	25 (0.11%)	212 (0.89%)
Validation	0.20%	8 (0.10%)	71 (0.90%)
Test	0.19%	8 (0.11%)	67 (0.89%)

Table 4.3: Distribution of subjects into training, validation and test set, CIMA-7

Since mostly the same recording where used, CIMA-7 has a shortest, longest and average sequence length of 1501, 12555 and 7726 frames while these values are 1919, 12556 and 7967 for CIMA-19.

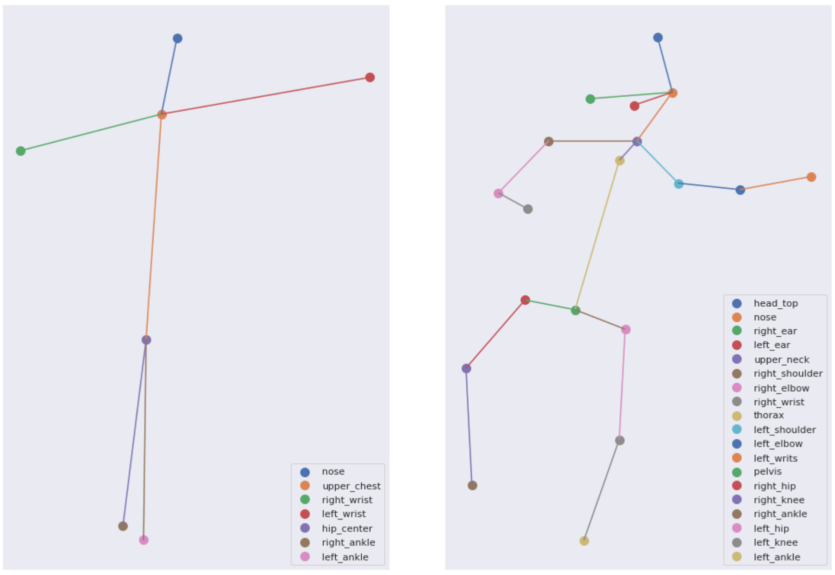
Not all of the video recordings have been captured with the same amount of frames per second. Table 4.2 shows the distribution of target and fps values. From this table we can see that the majority of recordings was captured with a fps of 29, roughly 74% and 79% for CIMA-7 and CIMA-19 respectively. We can also see that the distribution of target values is approximately 90/10 for non-CP and CP making prevalence of CP in both datasets $\tilde{1}$0%. As the data is skewed towards the non-CP case the method of splitting it into a train, validation and test set was chosen with this in mind. The In-Motion team decided on 75 subject ids to be used in the test set resulting in a 80-20 split between the train and test sets with both sets containing 90% non-CP and 10% CP subjects. The training set was then further split into a train and validation set using a stratified sampling approach ensuring an equal distribution of target values in both datasets. Table 4.3 and Table 4.4 gives and overview of the datasets with respect to both CIMA datasets.

4.1.2 HAR datasets

To test how well the proposed model generalizes, it was trained on two different HAR datasets. These datasets where chosen based on the amount of recordings they consisted of and the length of the individual sequences. If the performance on recognizing everyday human actions and interactions are reasonable, then we hope that the recognition of movement pattern specific to healthy and CP affected

	Fraction	CP	non-CP
Train	0.60%	25 (0.11%)	201 (0.89%)
Validation	0.20%	8 (0.11%)	68 (0.89%)
Test	0.20%	8 (0.11%)	67 (0.89%)

Table 4.4: Distribution of subjects into training, validation and test set, CIMA-19



(a) CIMA-7 **(b)** CIMA-19

Figure 4.1: Visualization of the different features contained in the CIMA-7 and CIMA-19 datasets.

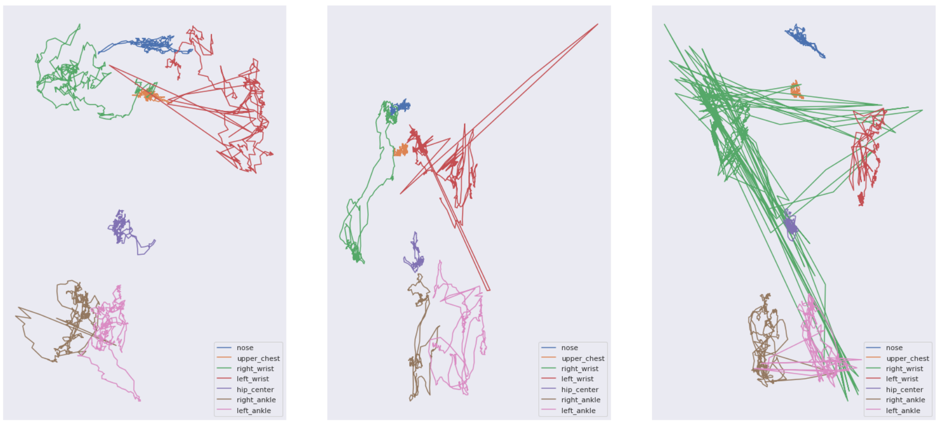


Figure 4.2: Different tracking errors in the CIMA-7 dataset. (*Left*) Left wrist jumps to the position of right wrist and upper chest. (*Middle*) Left wrist jumps away from the subject. (*Right*) Example of a poorly tracked subject.

infants can be recognized and determined.

Another motivating factor for evaluating the model on datasets other than the CIMA datasets is the possibility of leveraging the boost in performance Transfer Learning could provide by using the pre-trained model as a starting point for the final model trained on the CIMA datasets, as described in Section 2.3.3. Since the CIMA datasets contains one less axis of joint position only the x and y positions will be used from these dataset as to make them as similar as possible.

UTD Multimodal Human Action Dataset

The UTD Multimodal Human Action Dataset, *UTD-MHAD* [59], is a dataset that was collected by the Department of Electrical Engineering at the University of Texas, Dallas, for research on human action recognition. UTD-MHAD is a fusion of 4 different datasets consisting of RGB-video data captured at 30 frames per second, depth and skeleton data collected with a Xbox Kinect camera and inertial sensor data collected by two wearable sensors. The skeleton data was extracted with the publicly available software Kinect SDK, which tracks 20 different skeleton joints and their 3D spatial positions. This dataset consists of 27 different actions ranging from simple actions like swipe right and squat to more complex actions like pick up and throw and bowling. In total there are 8 subjects (4 males and 4 females) that are performing each action in a total of 4 times bringing the size of the dataset to 864 data sequences. However, since 3 data sequences are corrupted the true size is 861. During filming each subject were standing upright facing the camera when performing each action.

This dataset does not suffer from the same imbalance between classes as the CIMA datasets does, hence the reason a *leave-one-subject-out* strategy was selected when splitting the data into train, validation and test set. This strategy is similarly to the splitting strategy for the CIMA datasets as it validates and tests the model on sequences of actions performed by subjects not yet seen during training. The main difference is that the validation and test sets only contain one subject each. This leave-one-subject-out strategy resulted in a split of 645-108-108.

The recording lengths vary from 41 to 125 frames. With 30 frames captured per second this results in a shortest sequence just shy of 1.5 seconds and just more than 4 seconds for the longest sequence. The average length of one recording is 68 frames or just less than 2.3 seconds. KARD was chosen as one of two datasets to be used because of these properties. The shorter sequence length means that we are able to perform more experiments on model architecture and hyperparameters setups as the models require less time to train than with the longer sequences of the CIMA datasets. The frequency of which the recordings where captured and the amount of joints present in the data also played a part in the decision as they were similar to the CIMA datasets. Lastly, the small to medium sized pool of different classes present was also important as it reflects more simplistic actions. More complex actions like cooking dinner would require longer recordings, increasing the training time for each model.

Kinect Activity Recognition Dataset

The second HAR dataset used was the Kinect Activity Recognition Dataset, *KARD* [60]. This dataset was created to validate a proposed activity recognition method on a new dataset as well as the well-known publicly available dataset CAD-60 [61]. KARD consists of a dataset for RGB-video data captured at 30 frames per second, depth map and skeleton joint positions in real world and screen coordinates all captured with the use of a kinect. In total there are 15 different 3D joint coordinates extracted from the video recordings. There are a total of 2160 data sequences but given that the CIMA datasets contains screen coordinates, only the screen coordinates will be considered. This results in 540 sequences used for further testing. These 540 sequences are divided over 10 subjects (9 males and 1 female) performing each of the 18 actions 3 times each. One important difference between KARD and UTD-MHAD is that the subject does not always face the camera when performing the actions, e.g. during the walk action the subject starts facing the camera but turns 90 degrees exposing the subjects side before starting to walk.

An alternate version of the leave-one-subject-out was applied when splitting KARD into train, validation and test set. Instead of leaving one subject out for the validation and test set we left two out, still making the model validated and tested on previously not seen subjects as for UTD_MHAD. This resulted in a 324-108-108 split.

KARD contains recordings as short as 42 frames, just shy of 1.5 seconds. This makes the shortest sequences almost the same length as UTD-MHADs shortest, but KARD contain sequences up to 310 frames, 10.33 seconds. The presence of longer sequences makes the average length 118 frames. The same reasoning applies for our reason for choosing KARD as an additional dataset as for UTD-MHAD.

4.2 Transformer Model

The use of deep learning for fidgety movement detection and computer-based GMA has not been explored in depth yet. Neither has the transformer architecture been used with self-attention's promising success in fields dealing with sequenced data such as Natural Language Processing been used to perform HAR. For these reasons we decided to explore such a model utilizing the transformers encoder part, as described in Section 2.4. First the general architecture of the model will be introduced, followed by a description of ablations performed and hyperparameters explored. The same model architecture will be used regardless of what dataset it is trained on. The proposed model will be referred to as CIMA-Attend, as it the model that will follow the CIMA-Pose in the full video-to-CP prediction pipeline.

The general architecture of CIMA-Attend can be seen in Figure 4.3. This model take advantage of the transformers encoder, one out of two parts making up the transformer architecture [30], to extract spatial and temporal information from the embedded sequences without the use of RNNs. This is done iteratively N times after the input has been embedded and positional encoding has been added. Then the final step for producing the classification is to classify the input based on the features extracted.

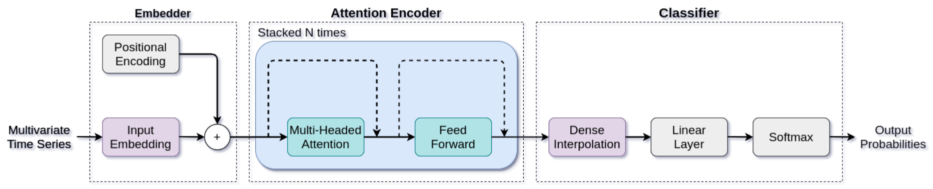


Figure 4.3: Network architecture of the Transformer Model

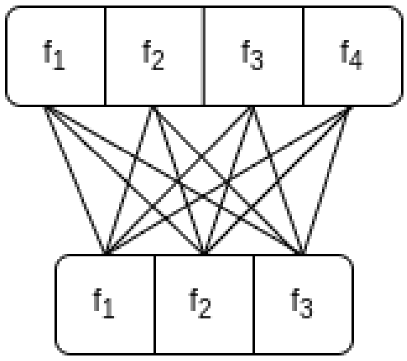
4.2.1 Model Parts

CIMA-Attend consists of three parts. These parts are, in order, the input embedder, attention encoder and classifier. These three parts will be explained in this section together with different variations testes during the experimentation phase of this project. The results of the proposed CIMA-Attend model and its variations will be evaluated in Chapter 5.

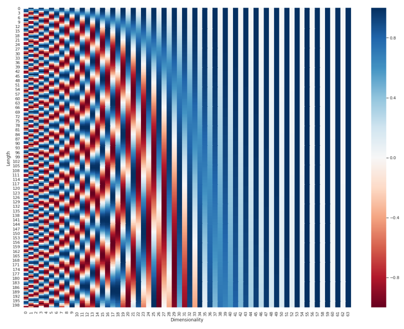
Input Embedder

The first part is concerned with learning a new representation of the input data which the rest of the model can use for generating an output prediction. To increase the dimensionality of the input sequences a learned embedding was used in the first part of CIMA-Attend. This converts the input vectors to vectors of d_{model} dimensions. This projection to a higher dimensional space will allow the embedder to express the data in a latent space with more expressive power than the original data space. As the weights are initialized randomly this embedding has to be optimized together with the rest of the CIMA-Attend's parameters. For the base model, a linear embedder was used with a value of $d_{model} = 128$. Figure 4.4a illustrates a linear embedder, this can be regarded as the same as a convolutional embedder with a kernel size of 1.

As the transformer encoder do not use any recurrence to express the order of which the elements are positioned in the sequence we need to add some information about this relative or absolute position into the sequence for it to be able to make use of the sequential nature of the data. This is done adding a positional encoding to the input embeddings. A fixed positional encoding was chosen over a learned one as it will allow CIMA-Attend to generalize to sequences longer than those it has encountered during training.



(a) Linear input embedding



(b) Positional encoding

Figure 4.4: Changed made to the data at the input layer

The positional encoding used for the proposed model is given by Equation (2.7) and has already been explained in Section 2.4. Figure 4.4b shows an illustration of a sinusoidal positional encoding for data of length 100 and a dimensionality of 64. An implementation of the positional encoding using torch is shown in Code listing 4.1. Here seq_len is the length of the input sequence x on the form $[batchsize, sequence\ length, features]$ and d_{model} is the number of features per feature vector.

Code listing 4.1: Implementaion of the Dense Interpolation

```
import torch

pe = torch.zeros(seq_len, d_model,dtype=torch.double)
position = torch.arange(0, seq_len, dtype=torch.double).unsqueeze(1)
div_term = torch.exp(torch.arange(0, d_model, 2).double() * /
        (-math.log(10000.0) / d_model))
pe[:, 0::2] = torch.sin(position * div_term)
pe[:, 1::2] = torch.cos(position * div_term)
pe = pe.unsqueeze(0)

x += self.pe[:, :x.size(1)].requires_grad_(False)
```

Attention Encoder

The attention encoder in the second part will be using the embeddeder output to allow each elements to attend to all other elements in the given sequence. This is achieved by utilizing multi-head attention as described in Section 2.4.2. In order for the network to encode these relationships into the sequence the attention encoding step is repeated N times. This iterative process is expressed in Figure 4.3. Each feed-forward network are equal in the sense of how they process the data, i.e. they expand the dimensionality from d_{model} to d_{ff} before reducing it back to d_{model}, but they are initialized with their own unique set of weights. For the base model, we used a value of $d_{ff} = 512$ for the feed-forward network and $N = 3$ for the number of attention encoder layers.

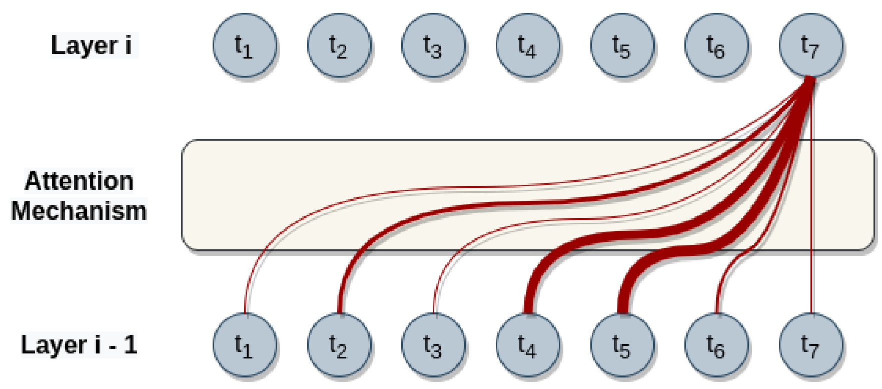


Figure 4.5: Information flow from one attention layer to the next

This encoding of a sequence inner relationships by using attention can be viewed as letting information from one layer flow to an element in the next layer with a rate defined by the attention mechanism. Figure 4.5 illustrates this flow by the thickness of each line. Element t_7 in layer i attends more to element t_2, t_4 and t_5 than the rest from layer $i-1$.

Classifier

The third and final part of CIMA-Attend is the classifier. It consists of three steps for producing an output prediction based on an encoded sequence, a dense interpolation layer, a linear feed-forward layer and a softmax layer to get the probabilities of the low and high risk for developing CP classes.

The simplest approach to represent a sequence as a single vector while maintaining order in the data is to concatenate every time step. This works well for short sequences, but when dealing with longer sequences like data from the CIMA datasets it will lead to a very high-dimensional representation not suitable for further processing. E.g. a sequence of length 870 with 128 features will lead to a vector of length 111.360. Trask et al. [62] proposed a dense interpolation embedding for embedding the structure of a sequence, which proved useful for detecting syntactic features. We hope that this embedding will capture the temporal structure of the movement data required for an accurate prediction of CP. Dense interpolation will reduce a sequence of any length down to a size of $M \times d_{model}$, where M is known as the dense interpolation factor, which is an useful property for classification of sequences with varying lengths. Code listing 4.2 shows an implementation of the dense interpolation algorithm in python by utilizing the torch library and batch matrix multiplication. Here seq_len is the length of the input sequence x on the form $[batchsize, sequence\ length, features]$ and M is the dense interpolation factor. By caching W into a buffer one could save valuable time by not recomputing it for every input, seq_len should be the length of the longest sequence the model will be presented with. The same is true for the positional encoding explained above. For the base model, a factor of $M = 8$ was used.

Code listing 4.2: Implementaion of the Dense Interpolation

```
import torch

W = torch.arange(1, seq_len + 1, dtype=torch.float) * factor / seq_len
W = W.repeat((factor, 1))
tmp = torch.t(torch.arange(1, factor + 1, dtype=torch.float).repeat((seq_len, 1)))
W = torch.pow(1 - torch.abs(W - tmp) / factor, 2)

w = W.repeat(x.shape[0], 1, 1).requires_grad_(False)
x = torch.bmm(w[:, :, :x.shape[1]], x)
```

Dense interpolation has been used for clinical prediction task such as in hospital mortality prediction and phenotyping by Song et al. [63], but it is performance for our classification task needs to be evaluated.

After dense interpolation is performed, each feature vector is concatenated before being fed into a feed-forward network for classification. This network consists of two linear and ReLU activation layers in alternating sequence followed by a final linear layer. The output of this layer is passed through a softmax function to produce the probabilities for each class in the dataset. Each linear layer reduces the dimensionality down to 256, 32 and 2 respectively.

4.2.2 Ablation Studies

As there has not been conducted a lot of research into deep learning-based GMA it is hard to tell if the proposed base model will achieve the wanted results. During this project an ablation study have been performed where parts of the model has either been removed, replaced or repeated to gain insights on their effects on the overall performance.

Input Embedding variations

Data embedding can be performed in multiple different ways. There exists fixed embeddings, e.g. dense interpolation as described above, and learned embeddings. In this project, we focused on learned embeddings for the model's input and the base model used a linear embedder consisting of a feed-forward layer projecting each time step feature vector to a feature vector with 128 dimensions. This can be viewed as a convolutional layer performing the projection with a kernel size of length 1. It is because of this a second input embedder consisting of a convolutional and a transposed convolutional layer are proposed. This embedder will use a kernel size of a length larger then 1, hence shrinking the length of any sequence it is applied to. The transposed convolution will increase the sequence back to it is original length, but will no increase of decrease the feature space. I.e. the convolutional layer will increase the feature dimensionality to d_{model} and the transposed convolution will keep it at d_{model}. This convolutional embedder will be utalizing a kernel size of 15 when used.

Attention Encoder variations

As the base model has 3 stacked attention encoders the amount of stacked encoders is a natural variation of the model to study. As the number of encoders in the network increases so does the depth. As the depth increases, so does the computational path resulting in more processing per sequence, i.e. it takes longer to train the model. Another side effect is that the model takes up more space in memory, limiting the space for use for training data. The goal of this ablation is to study the trade off between precision and time and space complexity. In this project it has been experimented with an encoder stack made up of 1 to 6 stacked encoders.

Classifier variations

The base model uses dense interpolation to reduce the length of a sequence down to a common size, before classification. The reasoning behind this was that the model could handle sequences of varying lengths, but it also depends on the interpolations ability to represent and capture the relationships found in the sequence by the preceding layers. To study the power of dense interpolation a convolutional layer will replace it for reducing the length of the sequence. By using convolutional layers instead the model will not be able to be used on sequences of different lengths as the feature vectors of the output will be concatenated resulting in different lengths. This convolutional classifier will be made out of three convolutional layers all with a kernel size of 15. The second layer will use a dilation with a value of 2 and stride of 2 and the third layer will use a stride of 3 and dilation of 1. In between each layer is a ReLU layer. This output is then fed to a similar linear classifier as in the base model, but with a different input dimension given by the convolutional layers.

4.3 Training

When training a deep learning model it is important to have a clear and well defined training strategy. Not only for a better understanding of the systems pipeline, but also for the purpose of reproducibility. Several aspects of the system should be taken into consideration, like the data that will be presented as input to the model and the optimization process. In this section the training strategy used when running the different experiments is described. The full pipeline of the training and evaluation process is illustrated in Figure 4.6

4.3.1 Data Preprocessing and Preparation

We will draw a distinction between data preprocessing and data preparation as the action of permanently augmenting the data and making non-lasting augmentations to the data respectively. The distinction is necessary as some changes to the data needs to be done in advance as they are only needed to be done once or need

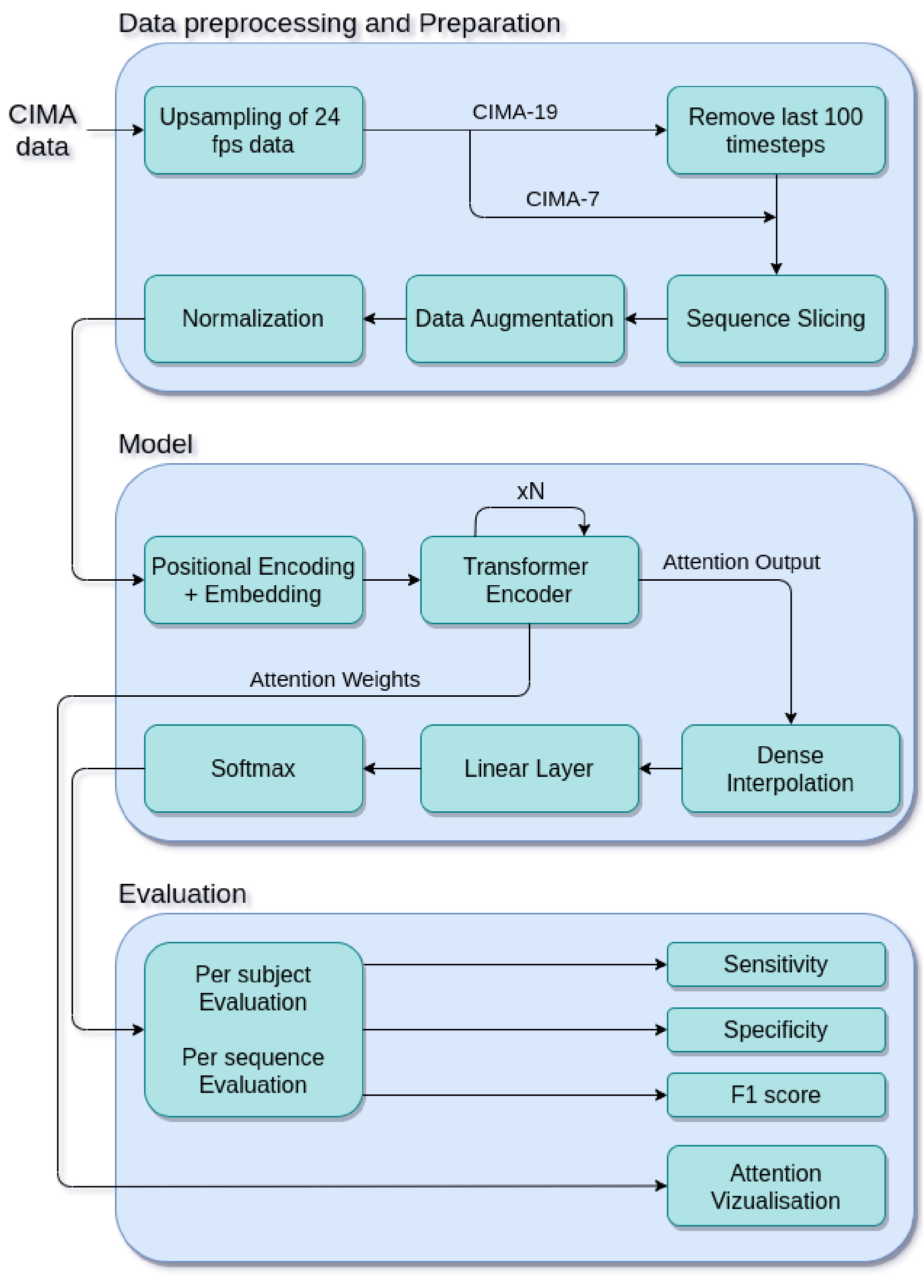


Figure 4.6: Pipeline for the whole process

more time to run to completion, while other can be done on the fly right before the data is presented to the model.

CIMA

Only one preprocessing step was needed for the CIMA datasets. From Table 4.2 we saw that there are few recordings captured at 24 fps compared to 29 fps so these will have to be resampled to make these sequences resemble sequences captured at 29 fps as not to confuse the model with data captured at 4 to 5 frames per second less than the rest. We will not be resampling sequences captured at 29 or 30 fps as given a sequence from four and a half minutes of video will have a length difference of only 274 timesteps, or $\tilde{9}$ seconds. As four and a half minutes is the average sequence length of the recordings used we will argue that the gain of resampling 29 fps recording to 30 fps would do more harm than good as it would alter a substantial amount of the datapoints. We will also argue that resampling the other way around, from 30 to 29 fps, also would not be worth doing as this downsampling could cause loss of information as timesteps are removed and not added to the data sequence. The only difference between the data preprocessing between CIMA-7 and CIMA-19 is that the last 100 frames are removed from each sequence in CIMA-19 due to an issue with the way the batches are created during tracking causing the last batch to suffer from some inconsistencies. These preprocessing steps of this pipeline can be seen in Figure 4.6 together with the data preparation steps, the model architecture and evaluation steps.

The fact that deep learning requires huge amount of data to succeed is well known. This has forced researchers to make an effort to make models trained on less data perform on par with models trained on large amounts of data. E.g Hu et al. [64] managed to achieve state-of-the-art results with their model trained on 10.000 images, compared to models trained on 500.000 images for human face recognition. The fact the CIMA datasets contain 513 and 377 data sequences each makes data preparation a necessity for any model wanting to produce any results worth noting. One way of achieving this is by applying random transformations to the data before presenting it to the model. For the CIMA datasets this process, called data augmentation, includes rotation around the center of the hip of the subject, flipping the data around the y-axis and adding noise from a normal distribution with a mean of 0.0 and a std of 0.001 to each joint position at every timestep. These random transformations are performed on the data with a probability of 30% each and are only applied to the data in the training set. After augmentation we perform one out of two normalization procedures to scale the data to a common scale. The first normalization procedure scales the data from one range to another, commonly known as min-max normalization or rescaling. Equation (4.1) shows the mathematical formulation of this normalization where a and b is the minimum and maximum values of the range the data will be scaled to, $[a, b]$, and X' is the rescaled data. When min-max normalization is used when running experiments the values $a = -1$ and $b = 1$ will be used, scaling the data to the range

$[-1,1]$. This normalization is not done feature-wise but rather on the whole datapoint making the $min(X)$ and $max(X)$ the minimum and maximum value of all the features in one sequence.

$$\mathbf{X}' = a + \frac{(\mathbf{X} - min(\mathbf{X}))(b-a)}{max(\mathbf{X}) - min(\mathbf{X})} \tag{4.1}$$

The second normalization procedure normalises the each feature based on the distance between the pelvis and thorax for CIMA-19 and hip center and upper chest for CIMA-7. If we consider a skeleton to be composed of P joints then each timestep $i = 1, 2, \dots, N$ in a sequence will be on the form $\mathbf{s_i} = [\mathbf{j_0}, \mathbf{j_1}, \dots, \mathbf{j_P}]$. Looking at CIMA-19 and Equation (4.2) [58] describing the normalization, each distance vector between the joints and j_0, the pelvis, will be normalized by the distance between j_0 and j_2, the thorax.

$$\mathbf{d_i} = \frac{\mathbf{j_i} - \mathbf{j_0}}{||\mathbf{j_2} - \mathbf{j_0}||}, \quad i = 1, 2, \dots, P \tag{4.2}$$

Each of the d_i features will be invariant to the position of the skeleton within each frame of the video recording and the now normalized sequence will contain one less feature as shown in Equation (4.3) due to the fact that we can not normalize the distance vector between j_0 and j_0, pelvis and pelvis. For the base mode, min-max normalization was used.

$$\mathbf{f_i} = [\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d_P}] \tag{4.3}$$

Now that the data have been augmented and normalized it could have been presented to the model, but since the sequences vary in length we will make sure to present the model with sequences of the same length during training and evaluation. Even though a fixed sequence length is used for training and evaluation, the nature of the Transformer model ensures that its inputs can be of different lengths if desired. We create sequences of the same length by randomly slicing the input sequence at two timesteps exactly a given sequence length apart. This sequence slicing will, in a way, act like a process to creating more data points that we originally have. This is because a sequence of length 7967 will produce 6467 unique sequences if sliced with a sequence length of 1500 at all possible positions. It might be a little to generous to call all these different slices unique datapoints as the movements in the first 100 frames will differ by only $\tilde{3}$.5 seconds, but given an overlap of only 50% Figure 4.7 shows that this sequence of length 7967 will contain 9 sequences that could be considered to be unique. As 7967 is the average sequence length of CIMA-19, it can be thought of as the dataset contains 9 times more than the original amount datapoints. A result of the sequence slicing is that it will us to batch multiple sequences together, resulting in one optimization step considering more than one sequence as they are all the same length. For the

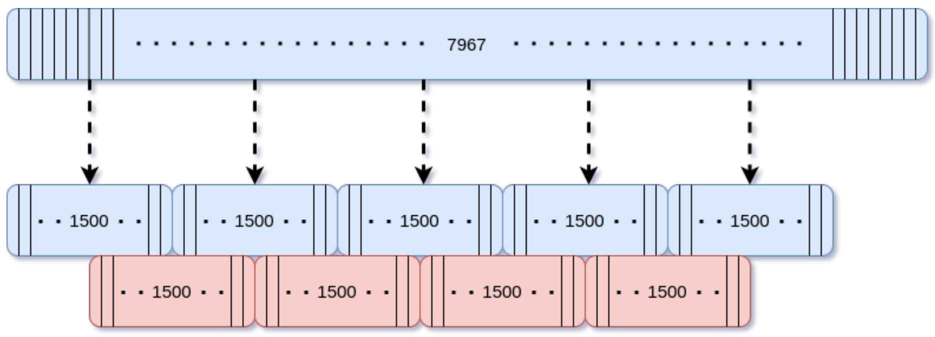


Figure 4.7: Illustration of sequence slicing

base model, a sequence length of 870 was used, this translates to 30 seconds for sequences captured at 29 fps.

KARD and UTD_MHAD

For KARD and UTD_MHAD there is no need of any form of data preprocessing as both datasets have been captured at 30 fps and the sequences does not suffer from any tracking errors like the last 100 frames of CIMA-19. The same data augmentation procedures have been used as with the CIMA datasets except that the rotation have been left out as the subjects are not laying down, but standing and/or walking while performing an action. The same normalization techniques have also been utilized.

Since the data sequences are quite a lot shorter than the ones in the CIMA datasets and the fact that each only contain data about one action, sequence slicing has been replaced with a data padding procedure. This is also due to the fact that one sequence only contains information about that one action it is meant to represent. Removing a part of the sequence will leave us with an incomplete picture of the action, which is inadequate when trying to classify the sequence and might lead to wrongly classified actions. E.g. if the only data in a sequence is of the subject starting to raise its hand, he could either end up just raining the hand or throw something depending on what comes next. This data padding ensures that all sequences presented to the model will have the length of the longest sequence and no information will be left out. The padding is performed by extending the start and the end of a sequence a random amount resulting in equally long sequences where the subjects will look like they are frozen in time during the padded parts of the sequences.

4.3.2 Data Sampling

Table 4.3 and Table 4.4 shows that the CIMA datasets suffer from an overrepresentation of non-CP diagnosed subjects. When trained on imbalanced datasets deep learning models have the tendency to be biased towards the majority classes, which in turn will lead to a higher misclassification rate in the classes that are in a minority. Buolamwini et al. [65] highlights one example of this problem in gender classification. They analyzed the accuracy of commercial gender classification products across light and dark skinned males and females and showed that they performed better on males and light skinned people compared to females and dark skinned people. So, instead of having the model deal with the imbalance, we can attempt to balance the class frequencies. To ensure a prevalence of approximately 50% of subjects diagnosed with CP in the data presented to the model during training we will add each of these sequences to the list of sequences n times such that the amount of CP subject is roughly the same as the amount of non-CP subjects. E.g. we can see from Table 4.4 that CIMA-19 contains 25 subjects diagnosed with CP and adding each of them to the training set 8 times will result in 200 CP datapoints which is close the amount of healthy subjects.

4.3.3 Optimization

Adam was chosen to optimize the model during the optimization process. The parameters used was β_1 and β_2 of 0.9 and 0.98 and ϵ of 1e-9. The learning rate was varied throughout the whole training process by following Equation (4.4).

$$lr = \gamma \cdot d_{model}^{-0.5} \cdot min(step\_num^{-0.5}, \; step\_num \cdot warmup\_steps^{-1.5}) \tag{4.4}$$

Here d_model is the dimensionality of the data after it has passed through the embedder, num_step is the current step number, $warmup\_steps$ is the amount of steps to use for a warm up period and γ is a scalar used for scaling the learning rate. A step in this context is one optimization step, i.e. if our dataset consists of 400 datapoints and a batch size of 20 is used it will take 20 steps to complete one epoch. This results in a learning rate that not only changes after every epoch, but within one epoch too. This learning rate equation will correspond to a linear increase in the learning rate for the first $warmup\_steps$ steps, subsequently decreasing it proportionally to the inverse square root of the step number for the rest of the training procedure.

When using the CIMA datasets the batch size was 15 and the warmup period was set to 100 epochs, resulting in a $warmup\_step$ value of ?? and 2700. For the base model, the factor γ was set to between 0.1 and the number of steps per epoch multiplied by 100 epoch was used for the $warm\_up$ value. Some learning rates are visualized in Figure 4.8 with a variation of d_model, γ and $warmup\_steps$ values. For KARD and UTD_MHAD a warmup period of 600 epochs was used together with a γ of 0.25.

Cross-entropy loss was used as the loss function. It is defined by Equation (4.5) and it is useful when training a classification problem with C classes, such as for HAR with KARD and UTD_MHAD, but it is also useful when C is equal to 2 as with

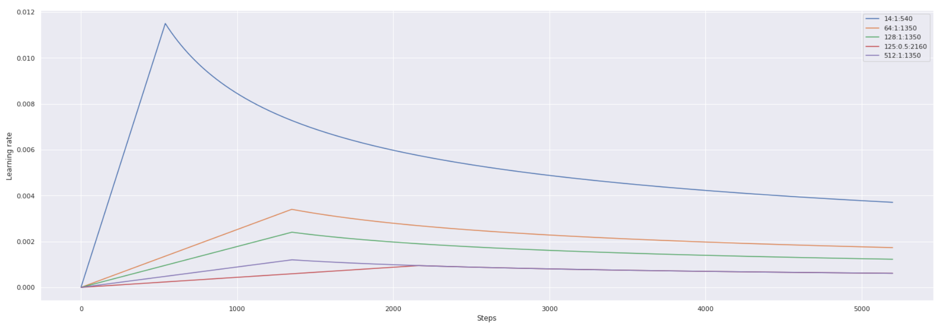


Figure 4.8: Illustration of the learning rate used during training. Legend is on the form *d_model:factor:warmup_step*

the CIMA dataset. In the equation for cross-entropy loss x is a vector containing representing the probability distribution over all possible classes.

$$\text{loss}(x,\ class) = -x[class] + \log(\sum_j exp(x[j])) \tag{4.5}$$

4.3.4 Regularization

For regularization we apply dropout to the sum of the input embeddings and positional encoding, (and) to the output of the multi-head attention and feed-forward network before it is added to the input and normalized and to the output of the convolutional part of the classifier (but this should be mentioned in the model variations). The reason we chose to employ dropout is to prevent overfitting of the training data. For the base model, a dropout rate of $P_{dropout} = 0.1$ is used.

4.3.5 Training the model

All base model, optimizer and dataset parameters has been stated above, but they can also be found in Table 4.5, Table 4.6 and Table 4.7. The number of epochs used for training each model has also been added to Table 4.7 where the optimizer's parameters are listed.

Data	$data\ length$	$batch\ size$	$flip$	$rotation$	$noise$	$normalization$
CIMA-7	870	15	✓	✓	✓	min-max
CIMA-19	870	15	✓	✓	✓	min-max
UTD_MHAD	125	35	✗	✗	✓	min-max
KARD	310	35	✗	✗	✓	min-max

Table 4.5: Summary of data parameters

Model	d_{model}	d_{ff}	N	h	M	$kernel\ size$	$p_{dropout}$
CIMA-7	128	512	3	8	8	1	0.1
CIMA-19	128	512	3	8	8	1	0.1
UTD_MHAD	128	512	6	8	8	1	0.1
KARD	128	512	6	8	8	1	0.1

Table 4.6: Summary of model parameters

Model	γ	$warmup$	$epochs$
CIMA-7	0.1	3600	200
CIMA-19	0.1	2700	200
UTD_MHAD	0.25	11400	1000
KARD	0.25	6000	1000

Table 4.7: Summary of optimizer parameters

In general, the data preparation is performed as described in Section 4.3.1. However, when a pre-trained network is utilized one more step is performed before the data can be presented to a model. The use and performance benefits of transfer learning in computer vision are well tested and documented. There exists models trained on millions of examples for object localization and classification task like VGG [56] and DenseNet [66] which can be used as a starting point or backbone for other computer vision tasks. Such pre-trained models does not exist for Human Activity Recognition. An effort has been made to experiment with transfer learning as it has had such success in other fields. When used in computer vision, all inputs have the same amount of channels, one red, green and blue one for colored images. These channels corresponds to each axis of movement for each joint in out data. I.e. movement data from KARD, which has 15 joints, contains 45 channels, while UTD_MHAD and CIMA19 contains 60 and 38 respectively. Since both HAR datasets contain 3 axis of movements and CIMA only 2, the depth axis will be removed flattening the 3 dimensional data down to 2 dimensions. Furthermore, the dimensionality of UTD_MHAD and CIMA19 is reduced to the amount of dimensions in the flattened KARD data. This is done by removing the joints least similar to the ones tracked in KARD. CIMA7 was chosen to not be included when testing transfer learning as the CIMA19 dataset is the main CIMA dataset used in this project. Additionally, the order of which these channels appear in matter. Joints which as the same or similar should appear in the same channel as it did in the data used for the pre-trained model. This is because the embedder learn the embedding weights and if channels appear in different order from data to data the knowledge embedded in the embedder weights is of no use.

As mentioned VGG and DenseNet are trained on over a million datapoints, thus needing little to no optimization when training the network using them as a backbone or feature extractor. Our models trained on UTD_MHAD and KARD are trained on less than 500 datapoints and will not generalize as well as VGG and DenseNet does to new data. Based on this, when weights from pre-trained models are used they will be updated together with rest of the weights using the same learning rate. The pre-trained model will in this case only act as a starting point for the new model, hopefully increasing its performance and helping with the convergence and stability.

4.3.6 Implementation Details and Hardware

The proposed model have been developed with Pytorch [67], which is a deep learning library for Python. Training was performed on a NVIDIA Tesla V100 PCIE GPU with 32 GB of memory when using the CIMA datasets and on a NVIDIA Tesla P100 PCIE with 16 GB of memory when UTD_MHAD or KARD was used.

4.4 Model Evaluation

A key step in any machine learning pipeline is the training of the model. However, an equally important aspect that should be considered is how well the model generalizes on unseen data. This is important as we need to be able to trust the predictions it makes and make sure that it actually works. Do the model make good classification predictions on future subjects, or subjects that it have not seen before, or is it merely memorizing the subjects it is presented with? So in order to evaluate the precision of the model a set of evaluation metrics will be defined as well as the way these metrics are used during the evaluation.

4.4.1 Evaluation Metric

A way to quantify a models performance is with the use of evaluation metrics and the choice of such metrics depends on the machine learning task at hand. As we are performing binary and multi-class classification we are going to be using evaluation metrics leveraging from the values of the confusion matrix.

Accuracy

Accuracy is the measure of the number of correct predictions out of the total number of predictions, i.e. the ratio of number of correct predictions. It is a simple measure and it works well when there is an equal number of samples belonging to each class, as is the case for the HAR datasets. If the data is skewed towards one or more of the class labels it doe not work as well as if 90% of samples belongs to one class and the last 10% belongs to the other class any model could easily get an accuracy of 90% by labeling all samples to belong to the first class. So, it makes sense to use this measure for models trained and evaluated on the HAR datasets, but not on the CIMA datasets as the prevalence of CP is around 10%.

Sensitivity and Specificity

Sensitivity, also called recall, is a measure of the true positive rate, i.e. the amount of actual positives that are correctly classified as such. On the other hand, *Specificity* is a measure of the true negative rate, i.e. the proportion of actual negative cases that are correctly classified as negative. Sensitivity and specificity is given by Equation (2.1) and Equation (2.2). So, in our case sensitivity measures the percent of subjects that are correctly identified as having a high risk of CP and specificity does the same, but for subjects with a low risk of CP.

F-Measure

Since sensitivity represents the avoidance of false negatives and specificity does the same for false positives there will often be a trade off between the two where in order to gain a higher sensitivity score the specificity will be affected. To take both the sensitivity and specificity into consideration when evaluating the model the harmonic mean of them both will be used. The harmonic mean is also known as the F_1 *measure*, Equation (4.6) and is commonly used in conjunction with precision and sensitivity.

$$F_\beta = (1+\beta^2) \cdot \frac{sensitivity \cdot specificity}{\beta^2 \cdot sensitivity + specificity} \tag{4.6}$$

This F-measure will value both specificity and sensitivity equally as neither false positives nor false negatives can be said to be more important to minimize the presence of than the other. I.e. predicting the an infant has a high risk of developing CP when it does not is as bad as that it does not have a high risk of developing CP when it does. With that being said, it is possible to value one more than the other by setting β to any positive real number. Two other commonly used F-measures are the F_2 *measure*, which weights specificity higher than sensitivity, and the $F_{0.5}$ *measure*, which will put more emphasis on sensitivity than specificity.

Equation (4.6) will be used when evaluating models trained on the CIMA datasets as sensitivity and specificity are terms generally used together with binary classification, hence it is not useful when evaluating multi-class classification models. When evaluating models trained on UTD_MHAD and KARD the F-Measure given by Equation (4.7) will be used instead. In this equation precision has taken the place of specificity. Precision is the measure for the amount of elements retrieved among all the relevant elements, i.e. the amount of true positive predictions among the true positive and false negative. All actions will be associated with its own F-measure and the final score will be the mean of the F-measures from all actions.

$$F_\beta = (1+\beta^2) \cdot \frac{sensitivity \cdot precision}{\beta^2 \cdot sensitivity + precision} \tag{4.7}$$

4.4.2 Classification

During training each datapoint was into a continuous sequence of N consecutive timesteps. Each subject with CP was also represented multiple times in the training set to make it more balanced. When evaluating the model each subject will be represented only once and the data sequences will be split into sequences of N consecutive timesteps with an overlap of 50%, like it is visualised in Figure 4.7. We will present each sequences of length N to the model to get an independent predictions for each of them, instead of just one sequence. This will be referred to as *Classification per Sequence* and will give us multiple predictions for each subject. A consequence of this is that different sequences from a recording of one subject might end up with different predicted classes. To obtain a final prediction for each subject a majority vote is used. This final prediction will be referred to as *Classification per Subject* and is the accumulated prediction of a subjects risk of CP from every sequence the recording was split into. It is also the only classification used for the HAR datasets.

4.4.3 Visualization

To better understand what parts of a sequence is attended to during the prediction process its attention can be visualized. The attention is on the form of a matrix with sides equal length to the input sequence length. The visualisation of this matrix is intuitive and easy to understand when used in fields like NLP, where each timestep represents one word and the sentences aren't too long, but not when dealing with sequences close to 1000 elements long. A better visualization was created for cases dealing with long sequences. The mean of each timestep in the attention matrix is calculated to create a vector containing how much each timestep was attended to. Visualizing this attention vector is more intuitive to understand as each frame of the recording is associated with a attention value, i.e. which frame the model gave the most weight during the generation of a prediction. Such a visualisation could also be paired with a playback of the recording giving the user a better understanding of which frames that has the highest attention value and contain the movements the model deemed most important for its prediction.

Chapter 5

Results

The upcoming chapter will present the results obtained from the training process and proposed model outlined in Chapter 4.

Section 5.1 will present the results from experiments using the HAR datasets UTD_MHAD and KARD. In Section 5.2 and Section 5.3 results obtained from initial testing on the CIMA-7 dataset and more rigorous testing on the CIMA-19 dataset are evaluated. All evaluations are performed on a test set which have been keep apart from the training and validation sets. Evaluation matrices are used as described in Section 4.4.1 This chapter ends by visualizing different attentions matrices and vectors produced during the evaluation phase.

5.1 Human Activity Recognition Results

The upcoming section evaluates the proposed model on the two datasets UTD_MHAD and KARD. The models are compared using the evaluation metrics accuracy and F-measure as described in Section 4.4.1

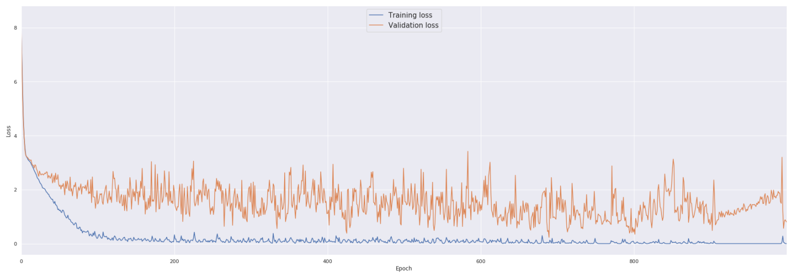
5.1.1 UTD_MHAD

Table 5.1 contains the results of the models trained on the UTD_MHAD dataset. The best accuracy and F-score for each normalization process is highlighted in bold. Classifier, Normalization and Convolutional has been shortened to Clf, Norm and Conv for consistency purposes. The effects of replacing the embedder and classification parts of the model are represented in this table. One can also notice that models trained on UTD_MHAD performs better on data normalized with the minmax technique.

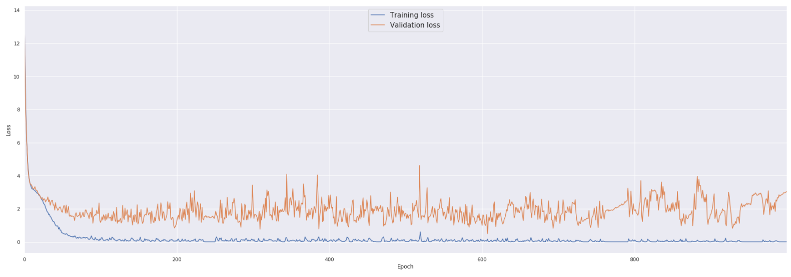
Figure 5.1 show the improvement of the best model for each normalization technique. It is worth noticing that both validation losses starts to oscillate around

Model	Embedder	Clf	Norm	Accuracy	F-score
Base	linear	linear	minmax	0.7963	0.7625
	conv	conv	minmax	0.9259	0.9214
	linear	conv	minmax	0.7685	0.7517
	conv	linear	minmax	**0.9444**	**0.9426**
Base	linear	linear	torso	0.7315	0.6906
	conv	conv	torso	0.8704	0.8621
	linear	conv	torso	0.6759	0.6168
	conv	linear	torso	**0.8981**	**0.8827**

Table 5.1: Results for models trained on UTD_MHAD, Classification per Subject



(a) Loss curve of best UTD_MHAD model, minmax normalization



(b) Loss curve of best UTD_MHAD model, torso normalization

Figure 5.1: Loss curve of the two best best models trained on UTD_MHAD. Loss is measured by the Cross Entropy Loss between the predicted and the true action

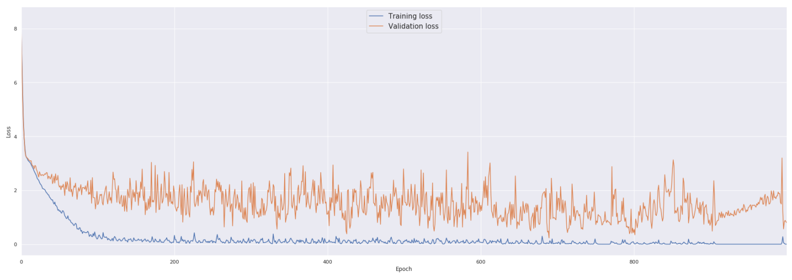
epoch 200, and never seem to steadily improve after this point. These models took on average one and a halt to two hours to train.

5.1.2 KARD

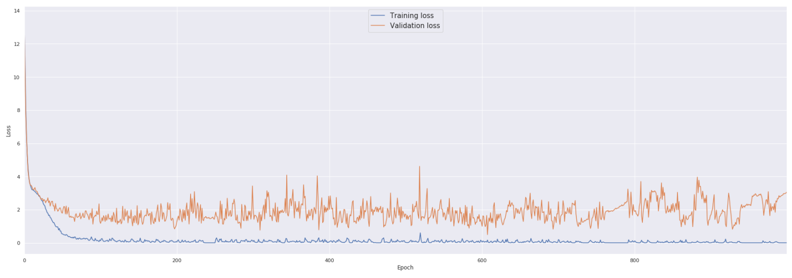
Results for models trained on the KARD dataset is shown in Table 5.2. The best accuracy and F-score for each normalization process is highlighted in bold. Three of the four models trained on minmax normalized KARD data achieves the same accuracy, but they differ in the F-score. Scores on the KARD dataset are very even, with four out of eight models getting the highest accuracy of 0.9537. The loss over all epochs for the models with the highest F-scores are show in Figure 5.2. These models took on average one and a half to two hours to train, much like the UTD_MHAD models.

Model	Embedder	Clf	Normalization	Accuracy	F-score
Base	linear	linear	minmax	**0.9537**	0.949
	conv	conv	minmax	**0.9537**	**0.952**
	linear	conv	minmax	**0.9537**	0.9508
	conv	linear	minmax	0.9352	0.9304
Base	linear	linear	torso	**0.9537**	**0.9523**
	conv	conv	torso	0.9167	0.9063
	linear	conv	torso	0.8889	0.8683
	conv	linear	torso	0.9167	0.8993

Table 5.2: Results for models trained on KARD, Classification per Subject



(a) Loss curve of best KARD model, minmax normalization



(b) Loss curve of best KARD model, torso normalization

Figure 5.2: Loss curve of the two best best models trained on KARD. Loss is measured by the Cross Entropy Loss between the predicted and the true action

5.2 CIMA-7 Results

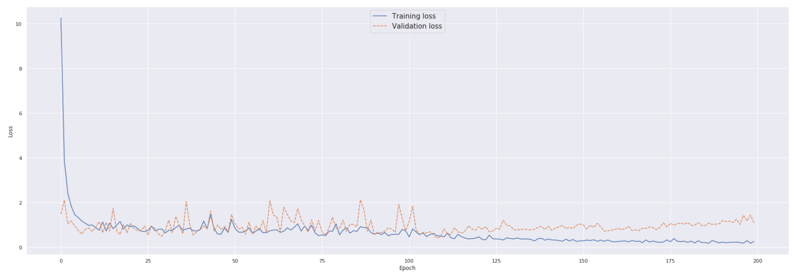
Table Table 5.3 and Table 5.4 shows the results of the experiments trained on the CIMA-7 dataset. The best score for models trained with the different normalization techniques are shown in bold. Figure 5.3a shows the loss improvement for the model with the F-score in bold. Figure 5.3b shows the same models F-score over all epochs.

Model	Embedder	Clf	Normalization	Accuracy	Specificity	F-Score
Base	Linear	Linear	minmax	0.6737	0.6788	0.6762
	Conv	Conv	minmax	0.3368	0.7719	0.469
	Linear	Conv	minmax	0.7053	0.7117	**0.7085**
	Conv	Linear	minmax	0.5632	0.7327	0.6368
Base	Linear	Linear	torso	0.6632	0.6732	**0.6681**
	Conv	Conv	torso	0.4105	0.8768	0.5592
	Linear	Conv	torso	0.4632	0.816	0.5909
	Conv	Linear	torso	0.5316	0.8013	0.6391

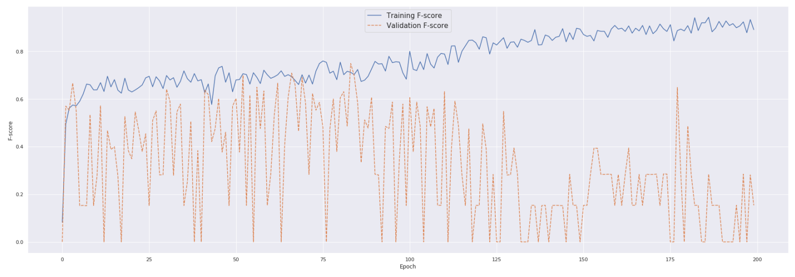
Table 5.3: Results for CIMA-7, Classification per Sequence

Model	Embedder	Clf	Normalization	Sensitivity	Specificity	F-Score
Base	Linear	Linear	minmax	0.6923	0.6889	0.6906
	Conv	Conv	minmax	0.2308	0.8222	0.3604
	Linear	Conv	minmax	0.7692	0.7667	**0.7679**
	Conv	Linear	minmax	0.5385	0.7667	0.6326
Base	Linear	Linear	torso	0.6923	0.7333	0.7122
	Conv	Conv	torso	0.4615	0.9444	0.6201
	Linear	Conv	torso	0.3846	0.8556	0.5307
	Conv	Linear	torso	0.6154	0.9	**0.731**

Table 5.4: Results for CIMA-7, Classification per Subject



(a) Loss curve of best CIMA-7 model



(b) F-score curve of best CIMA-7 model

Figure 5.3: Loss and F-score cure for the best model trained on CIMA-7. Loss is measured by the Cross Entropy Loss between the predicted and the true label

5.3 CIMA-19 Results

Table 5.5 and Table 5.4 show the results for classification per sequence and subject, respectively, for the models trained on the CIMA-19 dataset. The best F-score for each normalization process is highlighted in bold. In addition, the result of the big model is highlighted because__. Transfer learning was only performed with KARD models with the same Embedder, Classifier and Normalization, as described in Section 4.3.5. These tables shows the difference between the base model, models tested in the ablation study, a) and b), models utilizing transfer learning, c) and models trained on data of a different sequence length, d). Models where a part of the base model has been replace are labeled with a) and models where parts of the base model are removed or replicated are labeled b). N denotes the number of attention encoder layers and Seq len the length of the sequences presented to the models. The average training time is 5 hours.

Figure 5.3a shows the loss improvement for the model with the bold F-score. Figure 5.3b shows the same models F-score over all epochs. Not that then validation F-score curve varies a lot and doesn't converge like the training F-score curve does. This is true for the plots of the F-score for all model trained on the CIMA-19 dataset.

Model	*N*	Embedder	Clf	Norm	Transfer	Seq Len	Sensitivity	Specificity	F-Score
Base	3	Linear	Linear	minmax	✗	870	0.5294	0.7817	0.6313
a)	3	Conv	Conv	minmax	✗	870	0.563	0.7319	0.6364
a)	3	Linear	Conv	minmax	✗	870	0.4874	0.8759	0.6263
a)	3	Conv	Linear	minmax	✗	870	0.5294	0.8143	0.6417
b)	1	Linear	Linear	minmax	✗	870	0.5882	0.798	0.6772
b)	2	Linear	Linear	minmax	✗	870	0.5882	0.7672	0.6659
b)	4	Linear	Linear	minmax	✗	870	0.6555	0.74	**0.6952**
b)	5	Linear	Linear	minmax	✗	870	0.6134	0.7998	0.6943
b)	6	Linear	Linear	minmax	✗	870	0.563	0.8098	0.6642
c)	3	Conv	Conv	minmax	✓	870	0.605	0.6504	0.6269
c)	3	Linear	Linear	minmax	✓	870	0.4622	0.8514	0.5991
Base	3	Linear	Linear	torso	✗	870	0.5966	0.8786	0.7107
a)	3	Conv	Conv	torso	✗	870	0.5714	0.837	0.6792
a)	3	Linear	Conv	torso	✗	870	0.479	0.8759	0.6193
a)	3	Conv	Linear	torso	✗	870	0.5294	0.8379	0.6488
b)	1	Linear	Linear	torso	✗	870	0.7143	0.8524	**0.7772**
b)	2	Linear	Linear	torso	✗	870	0.5966	0.8324	0.6951
b)	4	Linear	Linear	torso	✗	870	0.3866	0.9049	0.5417
b)	5	Linear	Linear	torso	✗	870	0.7311	0.692	0.711
b)	6	Linear	Linear	torso	✗	870	0.5798	0.7572	0.6568
c)	3	Conv	Conv	torso	✓	870	0.3866	0.8442	0.5303
c)	3	Linear	Linear	torso	✓	870	0.5294	0.8741	0.6594
d)	3	Linear	Linear	torso	✗	290	0.4948	0.7843	0.6068
d)	3	Linear	Linear	torso	✗	1450	0.3971	0.9088	0.5527

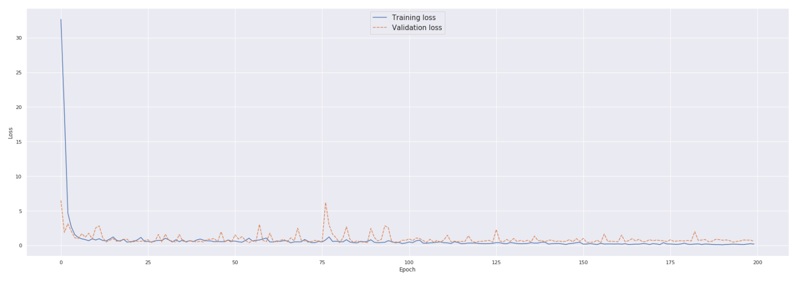
Table 5.5: Results for models trained on CIMA-19, Classification per Sequence

5.4 Attention Visualisation

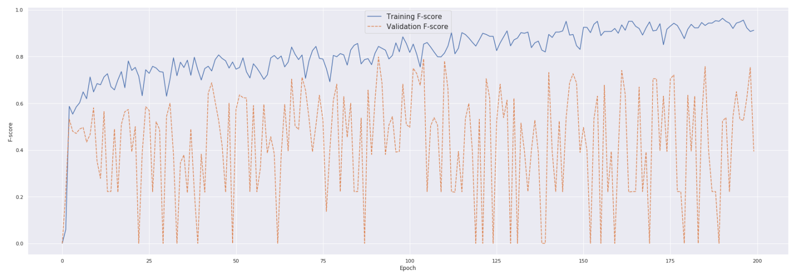
In the following section the attention matrices produced by the models and their corresponding attention vectors are visualized. Figure 5.5 shows the attention from attention encoder layer 1, 3 and 5 from a model trained on the UTD_MHAD dataset. The attention from the same layers from a model trained on the KARD dataset is shown in Figure 5.6. Lastly attention matrices and vectors from attention encoder layer 1 and 3 from a model trained on the CIMA-17 dataset is shown in Figure 5.7.

Model	*N*	Embedder	Clf	Norm	Transfer	Seq Len	Sensitivity	Specificity	F-Score
Base	3	Linear	Linear	minmax	✗	870	0.625	0.8507	**0.7206**
a)	3	Conv	Conv	minmax	✗	870	0.625	0.7463	0.6803
a)	3	Linear	Conv	minmax	✗	870	0.5	0.8955	0.6417
a)	3	Conv	Linear	minmax	✗	870	0.5	0.8955	0.6417
b)	1	Linear	Linear	minmax	✗	870	0.5	0.8657	0.6339
b)	2	Linear	Linear	minmax	✗	870	0.5	0.791	0.6127
b)	4	Linear	Linear	minmax	✗	870	0.625	0.7463	0.6803
b)	5	Linear	Linear	minmax	✗	870	0.5	0.8806	0.6378
b)	6	Linear	Linear	minmax	✗	870	0.5	0.8358	0.6257
c)	3	Conv	Conv	minmax	✓	870	0.625	0.6567	0.6405
c)	3	Linear	Linear	minmax	✓	870	0.5	0.9254	0.6492
Base	3	Linear	Linear	torso	✗	870	0.625	0.9104	0.7412
a)	3	Conv	Conv	torso	✗	870	0.625	0.8806	0.7311
a)	3	Linear	Conv	torso	✗	870	0.5	0.9403	0.6528
a)	3	Conv	Linear	torso	✗	870	0.5	0.8657	0.6339
b)	1	Linear	Linear	torso	✗	870	0.75	0.9403	**0.8344**
b)	2	Linear	Linear	torso	✗	870	0.5	0.8955	0.6417
b)	4	Linear	Linear	torso	✗	870	0.375	0.9851	0.5432
b)	5	Linear	Linear	torso	✗	870	0.75	0.7313	0.7406
b)	6	Linear	Linear	torso	✗	870	0.5	0.8209	0.6215
c)	3	Conv	Conv	torso	✓	870	0.5	0.9254	0.6492
c)	3	Linear	Linear	torso	✓	870	0.625	0.9701	0.7602
d)	3	Linear	Linear	torso	✗	290	0.5	0.8358	0.6257
d)	3	Linear	Linear	torso	✗	1450	0.3750	0.9254	0.5337

Table 5.6: Results for models trained on CIMA-19, Classification per Subject



(a) Loss curve of best CIMA-19 model, minmax normalization



(b) F-score curve of best CIMA-19 model, minmax normalization

Figure 5.4: Loss and F-score cure for the best model trained on CIMA-19 with minmax normalization. Loss is measured by the Cross Entropy Loss between the predicted and the true label

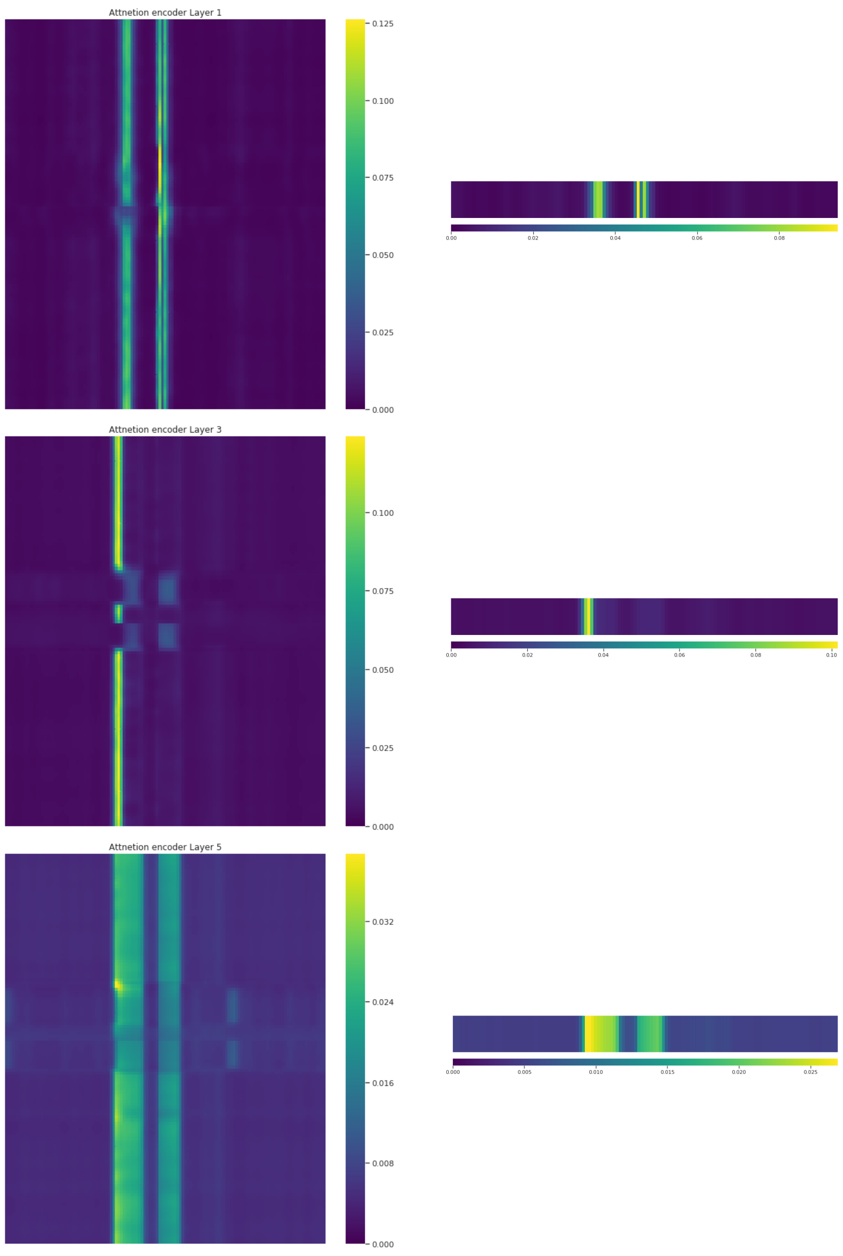


Figure 5.5: Attention matrices and vectors for attention encoder layer 1, 3 and 5 in a UTD model. Action was Draw X

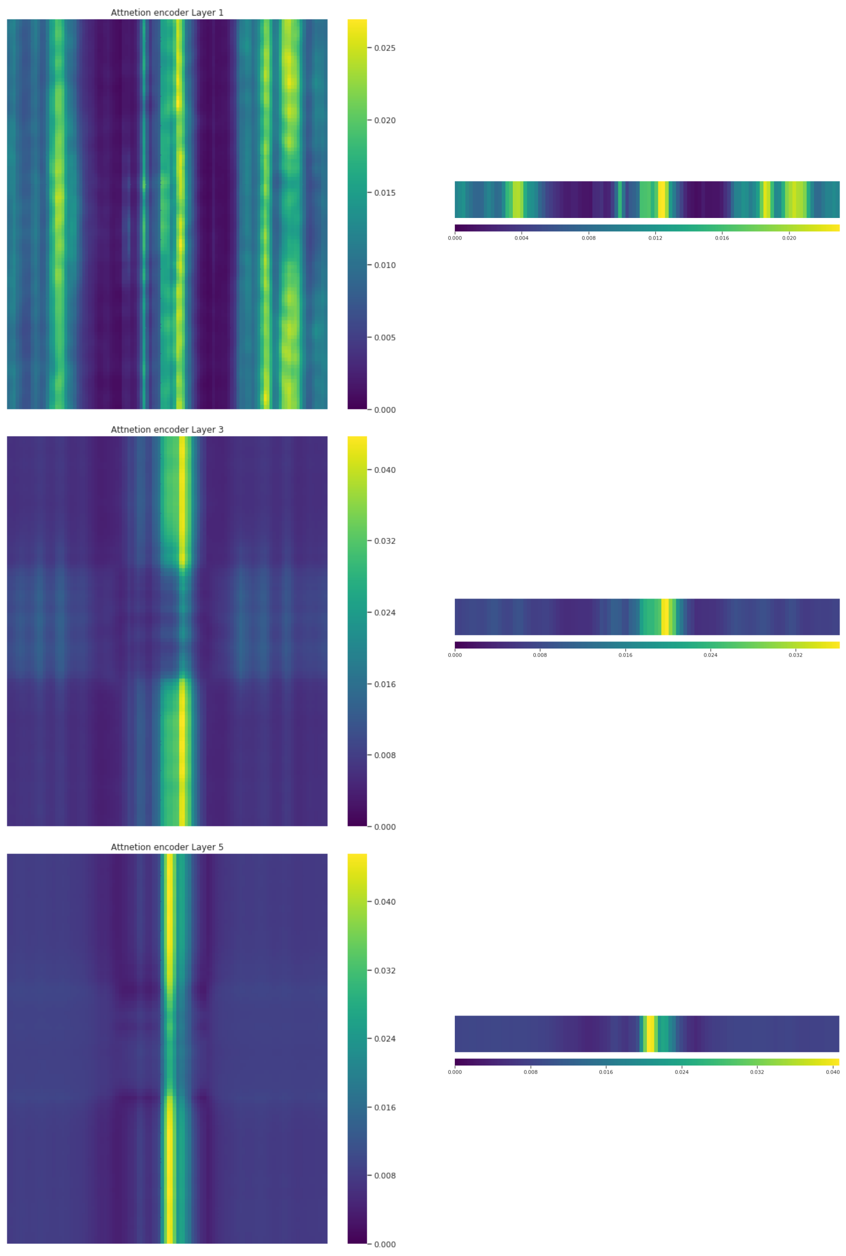


Figure 5.6: Attention matrices and vectors for attention encoder layer 1, 3 and 5 in a KARD model. Action was Bend

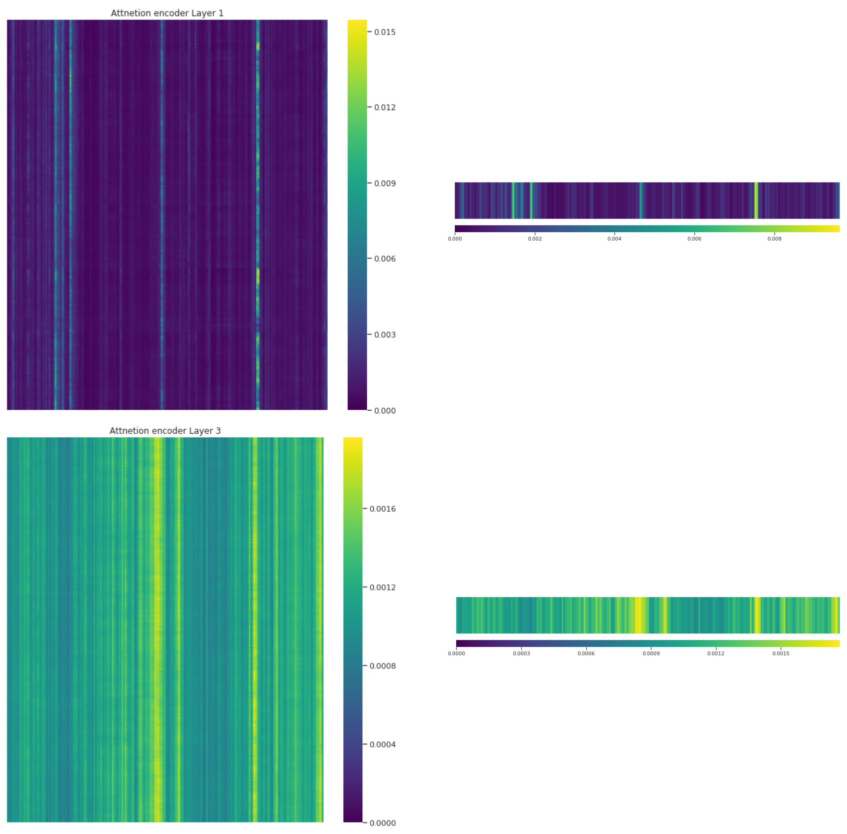


Figure 5.7: Attention matrices and vectors for attention encoder layer 1 and 3 in a CIMA-19 model. Target value was 0, no CP

Chapter 6

Discussion

The upcoming chapter will discuss the results presented in Chapter 5. Section 6.1 will asses the processing performed on the datasets Subsequently, Section 6.2 will discuss the results of the base model, as well as results obtained during the ablation study and effects of transfer learning on the model. Section 6.3, assesses the optimization process and its effect on the models ability to learn movement patterns corresponding to CP. This is followed by a discussion about the attention visualisation in Section 6.4. Finally, in Section 6.5 the research questions are answered based on the results obtained during the project.

6.1 Datasets

During this project the proposed CIMA-Attend model has been tested on 4 different datasets as described in Section 4.1. One reason for this was to see how well the model performs on different datasets. Another reason was to see how well knowledge could be transferred from one problem to the other. Deep learning is known for requiring vast amounts of data to perform well, something we didn't have for this project. The preparation and augmentation process described in Section 4.3.1 increased the amount of unique data sequences we are able to create from these datasets, further increasing the performance of the models.

From the results presented in the previous chapter, it is hard to determine which normalization technique is the dominant one. When torso normalization is used on KARD it doesn't seem to affect the performance of the model. The contrary is true when used in conjunction with UTD_MHAD, the F-score drops by nearly 6% from the best model trained on data normalized with the minmax technique. The same is seen for the CIMA datasets too. Models trained on the CIMA-7 dataset prefers the minmax normalization, while models trained on CIMA-19 seems to favor the torso normalization. This difference in performance can be caused by the tracking errors in CIMA-9, described in Section 4.1.1, as minmax normalization will perform a scaling based on the minimum and maximum values present in the sequence while torso normalization will scale each features based on it's distance to the hip and the distance between the hip and the chest. Sudden jumps and tracking errors of the joints will then be more affected by the torso normalization, making it less preferred by CIMA-7 models. The use of different sequence length didn't seem to have a positive effect on the performance of the model with both a sequence length of 290 and 1450 showed worse results that the base sequence length of 870.

6.2 Model

6.2.1 Base Model

As described in Section 4.2 the base model consist of a linear embedder, attention encoder layers, a dense interpolation layer and a linear classification layer. This model architecture performed well in conjunction with both minmax and torso normalization on CIMA-19. This suggests that considering each times step separately during embedding and using dense interpolation to capture the temporal structure of the movement data works reasonably well for the classification task at hand. It is interesting to notice the rather large differences between the F-score for classification per sequence and per subject. This indicates that when the model makes a wrong prediction it does so by assigning a larger part of the sub-sequences used in evaluation to the same target label causing more false positives or negatives than when a majority vote is used to decide the assigned target label of a sequence.

By looking at the F-score curve in Figure 5.4b of the base model trained on minmax normalized CIMA-19 data, we can see that the validation F-score is very unstable and doesn't seem to converge as the training F-score does. This can be a result of a poor optimization process or data with a rather compacted relationship the target label and will be discussed more in Section 6.3.

6.2.2 Ablation Study

The reason behind exploring the different variations proposed to the model is to search for the best performing model architecture. As described in Section 4.2.2 the variations to the model consists of replacing the input embedder layer, the classification layer or both and increasing or decreasing the number of attention encoder layers used. In total, eight model variation was experimented with, all using both minmax and torso normalization, but only on CIMA-19. In general all ablations to the model regarding any combination of embedder and classifier performed worse than the base model in regards to classification per subject. When minmax normalization were used the F-score dropped by as much as 8% and by more than 10% for models trained on torso normalized data. These results would suggest that dense interpolation is able capture the temporal structure of the movement data better than a series of convolutional layers. This is rather surprising as convolutional layers are know for their ability to capture the spatial and temporal relationships in sequential data. The results also suggests that a linear embedding layer is better suited for embedding the input sequences one time step at a time, than a convolutional embedder taking 15 time steps into account when performing the embedding. This is also quite surprising as the thought behind this embedder was to better the amount of information used when the embedding was performed. Never the less, based on these results the choice of using the base model for further testing during the ablation study was made.

An other ablation made to the model was regarding the number of attention encoder layers used. results from these tests are inconclusive as study doesn't find any relationship between the number of layers and the F-score. A result worth noting is the F-score of 0.8344 when one layer is used together with torso normalization. It's the odd one out, in a positive way, compared to the rest of the results. Comparing it the the F-score of the corresponding model trained on minmax normalized data it is 20% higher, which is an substantial increase and can be explained by a lucky weight initialization, a series of lucky optimization steps or both.

6.2.3 Transfer Learning

Transferring the weight from models trained in UTD_MHAD and KARD had the reverse effect than it was intended to have. Instead of speeding up training and giving better performance, it made the models using a pre-trained model as a starting point perform worse than the base model. It decreased by as much as 4% and 5% for the base models trained on minmax and torso normalization respectively. This could be the cause of using pre-trained model which hadn't been able to generalize well enough. Model used for transfer learning in this project were also trained on a small amount of data, only seeing the movements and specific actions existing within the sequences of the respective dataset. This is far from the quality and amount of data pre-trained models used for transfer learning in computer vision is trained on. Using these models as a starting point for training has shown how powerful and well generalized they. This is not the case for the pre-trained models we trained on different human activity recognition datasets. The fact that the KARD models have seen 18 actions out of all the thinkable actions a human can perform repeated three times by six subjects makes their possibility to generalizes well enough equal to zero.

If their had existed models pre-trained on a HAR dataset equivalent to the ImageNet dataset for computer vision it probably would have had a greater impact on the performance of the models utilizing transfer learning. With that being said, there exists larger dataset made for HAR than UTD_MHAD and KARD, e.g. NTU RGB+D and NTU RGB+D 120, but they were not obtainable during this project.

6.3 Optimization Process

The optimization process used in was based on the Adam optimizer and used a learning rate that would change for each optimization step, described in Section 4.3.3. The learning rate was purposely set really low as to give the models time to warm up and lower the effects of accidentally update the weights too much and jump over a local minimal. The fact that this is a very complicated optimization problem is reflected in Figure 5.3b and Figure 5.4b. Here we can see the F-score jump from 0 to the 0.8 in a matter of a few epochs before going down to the lower scores again. This is a sign of a unstable learning process caused by the complex problem at hand and it will not converge. As mentioned in the start of this chapter, deep learning requires large amount of data to succeed, without it the model will generalize poorly and make errors when presented with previously unseen data. Such is the case with these models. They perform well on UTD_MHAD and KARD as they consists of rather short sequences and of clearly defined actions to be classified. We can see variation in validation F-score for these models too, but to a lesser extent. As the data that makes up the CIMA datasets is everything but this, the optimization fails to converge and we are left with a model that might perform well if we initialize the weights with favorable weights, hit a series of lucky optimization steps or both.

The time each model is let to train compared to the training time of the Transformer Model first proposed, 12 hours to 3.5 days, might also be a reason for why convergence is never achieved. Unfortunately, due to time constraints the training of a model over 1000 epochs was dropped. This model would have taken 24 hours to train and it would have been exciting to see if if would have helped on the convergence problem.

6.4 Attention Visualization

The attention aspect of the model was interesting as it could be used to make more sense out of the predictions of the model. The developed attention vector representation of the attention encoders matrix representation of the attention was created such that it could be paired up with the recording of a subject in such a way that each frame was given an attention score. This way of visualizing the attention was thought to be able to give the user of the model a deeper insight in what parts of the recording that were weighted the most and thus played a bigger role in the generation of the prediction. Unfortunately, we weren't able to test this idea and have the annotated recordings review by an expert in GMA due to time constraints.

6.5 Answering Research Questions

The research questions this project were based on were presented in Section 1.3. From these research questions the project was conducted to evaluate the possibility of prediction CP by the use of a deep learning model in a way that accurate enough to be used for aiding medical diagnoses. During this discussion we have seen that problem comes with a great number of challenges, and as of now we are not confident enough in the proposed model to argue that it is capable of this yet. For the rest of this section we will answer each research question.

RQ1: *How can deep learning be used to solve the problem of predicting an infants risk of developing Cerebral Palsy by using human skeleton time series data?*

Starting with the proposed base model, it seemed that deep learning would be able to predict the risk of Cerebral Palsy by using human skeleton time series data. However, as further testing on different model variation showed this model might not be the best architecture to use on this problem. There were no pattern clear pattern in the model variations and their corresponding evaluation score. By testing the model on different datasets for similar problems, hope was restored for the models ability to accurately predict an infants risk of CP. The amount of data available to train the deep learning model is not enough for assuring stable training and convergence, making the results depend on the available data as well as the model architecture.

Given the results, we can report that the proposed model does not sufficiently address this research question. We do, however, have faith in the proposed model ability to accurately predict the risk of CP if given more training data .

RQ2: *How can said model aid the users in their evaluating of a patient? Can it recognize the patterns of movement associated with Cerebral Palsy and visualise it to better the users understanding of the prediction, thus making the model usable for medical diagnosis?*

A visualization of the attention score for each frame in a recording was created, but due to time constraints we didn't have the opportunity to test it on experts in the field of GMA. Thus, this research question is not sufficiently addressed in this thesis to be able to be answered at the time of writing.

RQ3: *What are the advantages and disadvantages of such a model? Does it compete with today's methods of predicting Cerebral Palsy, such as General Movement Assessment?*

Currently used methods of predicting CP either require access to a trained clinician or relies on carefully crafted features or manual interaction to work. The proposed method only relies on features extracted from raw video recordings by a pose estimator. However, it relies on large amounts of data to be able to learn relationships corresponding to CP in the data. Such a model as the one proposed would be able to be used by everyone without any form of training or knowledge about how the system works, bringing GMA to the masses.

When it comes to the accuracy, the model is beaten by the professionals as well as by other models currently available. It would still need some work before it can overtake the currently used methods of assessment.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis we have explained the importance and effects of a fully automatic Cerebral Palsy risk assessment method. We have presented some theory behind CP and its development in the human brain as well as current methods for detecting infant with a high risk of developing the disorder and the reasons why it is so important to detect them as early as possible. The experts who conduct the tests to determine the risk of CP uses Gestalt Perception and GMA. These techniques require manual work and highly trained professionals and it is with this in mind we present the relevant theoretical background knowledge in Deep Learning and Human Activity Recognition which will be used to create the proposed model. This model is inspired by resent advances in other fields concerned with sequential data such as Natural Language Processing. The proposed model utilizes skeletal data extracted from raw video recordings by an other deep learning model trained for Human Pose Estimation. Models trained on this data were shown to not be as stable as first hoped and showed difficulties in converging and the model was beaten by currently existing techniques in CP detection.

In light of these results, we are not confident in concluding that the model resolves our research questions. We are, however, optimistic in the models abilities as it performs well on other datasets similar to the CIMA datasets and believe that with more data and time it can beat existing methods for computer-based GMA.

7.2 Future Work

Based on the conclusion, we suggest some improvements to the existing parts of the project.

7.2.1 Dataset

To make the model able to generalize better and learn new movements relationships related to CP, a larger and more diverse dataset should be developed. Such a dataset would be beneficial for both the tracker and CP prediction Model.

7.2.2 Tracker Model

By performing tracking in 3 dimensions of movement instead of 2, the increased accuracy of movements could help further improve the CP prediction model by leveraging on movements represented in higher dimensions. Such models already exist and have achieved remarkable results when it comes to predicting depth from a 2 dimensional image. [1]

7.2.3 CP prediction Model

By performing transfer learning with models trained on much larger HAR datasets, we could maybe start seeing some improvements in the performance on the model. Two such larger than normal HAR datasets are NTU RGB+D and NTU RGB+D 120 [69]

The optimization process is crucial for a good result. Experimenting more with the models trained over more epochs could lead to models that will start to converge. One possible alteration is to shrink the learning rate by a factor γ every x epoch to move the model into a fine-tuning stage.

The world of Deep Learning moves incredibly quickly and other transformer architecture like the Reformer [70], Longformer [71] and Linformer [72] have been proposed. Utilizing the advances these models have made could be the next step in the work of the CP detection model.

[1]One such model is proposed by Luo et al. [68]

Bibliography

[1] P. Baxter, C. Morris, P. Rosenbaum, N. Paneth, A. Leviton, M. Goldstein, M. Bax, A. Colver, D. Damiano, H. Graham *et al.*, 'The definition and classification of cerebral palsy', *Dev Med Child Neurol*, vol. 49, no. s109, pp. 1–44, 2007.

[2] G. Beaino, B. Khoshnood, M. Kaminski, V. Pierrat, S. Marret, J. Matis, B. Ledesert, G. Thiriez, J. Fresson, J.-C. ROZÉ *et al.*, 'Predictors of cerebral palsy in very preterm infants: The epipage prospective population-based cohort study', *Developmental Medicine & Child Neurology*, vol. 52, no. 6, e119–e125, 2010.

[3] M. Hadders-Algra, 'General movements: A window for early identification of children at high risk for developmental disorders', *The Journal of pediatrics*, vol. 145, no. 2, S12–S18, 2004.

[4] A. Stahl, C. Schellewald, Ø. Stavdahl, O. M. Aamo, L. Adde and H. Kirkerod, 'An optical flow-based method to predict infantile cerebral palsy', *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 20, no. 4, pp. 605–614, 2012.

[5] P. Rosenbaum, 'Cerebral palsy: What parents and doctors want to know', *Bmj*, vol. 326, no. 7396, pp. 970–974, 2003.

[6] M. Bosanquet, L. Copeland, R. Ware and R. Boyd, 'A systematic review of tests to predict cerebral palsy in young children', *Developmental Medicine & Child Neurology*, vol. 55, no. 5, pp. 418–426, 2013.

[7] T. M. O'Shea, 'Diagnosis, treatment, and prevention of cerebral palsy in near-term/term infants', *Clinical obstetrics and gynecology*, vol. 51, no. 4, p. 816, 2008.

[8] H. F. Prechtl, 'Qualitative changes of spontaneous movements in fetus and preterm infant are a marker of neurological dysfunction.', *Early human development*, 1990.

[9] C. Einspieler, R. Peharz and P. B. Marschik, 'Fidgety movements–tiny in appearance, but huge in impact', *Jornal de Pediatria*, vol. 92, no. 3, S64–S70, 2016.

[10] H. F. Prechtl, C. Einspieler, G. Cioni, A. F. Bos, F. Ferrari and D. Sontheimer, 'An early marker for neurological deficits after perinatal brain lesions', *The Lancet*, vol. 349, no. 9062, pp. 1361–1363, 1997.

[11] C. Einspieler and H. F. Prechtl, 'Prechtl's assessment of general movements: A diagnostic tool for the functional assessment of the young nervous system', *Mental retardation and developmental disabilities research reviews*, vol. 11, no. 1, pp. 61–67, 2005.

[12] M. Hadders-Algra, 'The assessment of general movements is a valuable technique for the detection of brain dysfunction in young infants. a review', *Acta Paediatrica*, vol. 85, pp. 39–43, 1996.

[13] M. Burger and Q. A. Louw, 'The predictive validity of general movements–a systematic review', *European journal of paediatric neurology*, vol. 13, no. 5, pp. 408–420, 2009.

[14] D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning internal representations by error propagation', California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[15] P. J. Werbos, 'Applications of advances in nonlinear sensitivity analysis', in *System modeling and optimization*, Springer, 1982, pp. 762–770.

[16] X. Glorot, A. Bordes and Y. Bengio, 'Deep sparse rectifier neural networks', in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[17] J. Kiefer, J. Wolfowitz *et al.*, 'Stochastic estimation of the maximum of a regression function', *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462–466, 1952.

[18] D. P. Kingma and J. Ba, 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980*, 2014.

[19] K. Fukushima and S. Miyake, 'Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition', in *Competition and cooperation in neural nets*, Springer, 1982, pp. 267–285.

[20] A. Krizhevsky, I. Sutskever and G. E. Hinton, 'Imagenet classification with deep convolutional neural networks', in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[21] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang and C. Liu, 'A survey on deep transfer learning', in *International conference on artificial neural networks*, Springer, 2018, pp. 270–279.

[22] J.-T. Huang, J. Li, D. Yu, L. Deng and Y. Gong, 'Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers', in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2013, pp. 7304–7308.

[23] J. Yosinski, J. Clune, Y. Bengio and H. Lipson, 'How transferable are features in deep neural networks?', in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[24] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, 'Grad-cam: Visual explanations from deep networks via gradient-based localization', in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[25] D. Bahdanau, K. Cho and Y. Bengio, 'Neural machine translation by jointly learning to align and translate', *arXiv preprint arXiv:1409.0473*, 2014.

[26] M.-T. Luong, H. Pham and C. D. Manning, 'Effective approaches to attention-based neural machine translation', *arXiv preprint arXiv:1508.04025*, 2015.

[27] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel and Y. Bengio, 'Show, attend and tell: Neural image caption generation with visual attention', in *International conference on machine learning*, 2015, pp. 2048–2057.

[28] J. Cheng, L. Dong and M. Lapata, 'Long short-term memory-networks for machine reading', *arXiv preprint arXiv:1601.06733*, 2016.

[29] R. Pascanu, T. Mikolov and Y. Bengio, 'On the difficulty of training recurrent neural networks', in *International conference on machine learning*, 2013, pp. 1310–1318.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, 'Attention is all you need', in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[31] K. He, X. Zhang, S. Ren and J. Sun, 'Deep residual learning for image recognition', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[32] J. L. Ba, J. R. Kiros and G. E. Hinton, 'Layer normalization', *arXiv preprint arXiv:1607.06450*, 2016.

[33] A. W. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi and Q. V. Le, 'Qanet: Combining local convolution with global self-attention for reading comprehension', *arXiv preprint arXiv:1804.09541*, 2018.

[34] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, 'Bert: Pre-training of deep bidirectional transformers for language understanding', *arXiv preprint arXiv:1810.04805*, 2018.

[35] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever, 'Language models are unsupervised multitask learners', *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[36] C. Jobanputra, J. Bavishi and N. Doshi, 'Human activity recognition: A survey', *Procedia Computer Science*, vol. 155, pp. 698–703, 2019.

[37] M. Vrigkas, C. Nikou and I. A. Kakadiaris, ‘A review of human activity recognition methods’, *Frontiers in Robotics and AI*, vol. 2, p. 28, 2015.

[38] S. Wan, L. Qi, X. Xu, C. Tong and Z. Gu, ‘Deep learning models for real-time human activity recognition with smartphones’, *Mobile Networks and Applications*, pp. 1–13, 2019.

[39] L. Adde, J. L. Helbostad, A. R. Jensenius, G. Taraldsen, K. H. Grunewaldt and R. Støen, ‘Early prediction of cerebral palsy by computer-based video analysis of general movements: A feasibility study’, *Developmental Medicine & Child Neurology*, vol. 52, no. 8, pp. 773–778, 2010.

[40] L. Adde, J. L. Helbostad, A. R. Jensenius, G. Taraldsen and R. Støen, ‘Using computer-based video analysis in the study of fidgety movements’, *Early human development*, vol. 85, no. 9, pp. 541–547, 2009.

[41] K. Aurlien and D. Groos, ‘Infant body part tracking in videos using deeplearning: Facilitating early detection of cerebral palsy’, Master’s thesis, Norwegian University of Science and Technology, 2018.

[42] D. Groos, H. Ramampiaro and E. Ihlen, ‘Efficientpose: Scalable single-person pose estimation’, *arXiv preprint arXiv:2004.12186*, 2020.

[43] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei and Y. Sheikh, ‘Openpose: Realtime multi-person 2d pose estimation using part affinity fields’, *arXiv preprint arXiv:1812.08008*, 2018.

[44] H. Rahmati, H. Martens, O. M. Aamo, Ø. Stavdahl, R. Støen and L. Adde, ‘Frequency analysis and feature reduction method for prediction of cerebral palsy in young infants’, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 24, no. 11, pp. 1225–1234, 2016.

[45] H. Rahmati, R. Dragon, O. M. Aamo, L. Van Gool and L. Adde, ‘Motion segmentation with weak labeling priors’, in *German Conference on Pattern Recognition*, Springer, 2014, pp. 159–171.

[46] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu and J. Zhang, ‘Convolutional neural networks for human activity recognition using mobile sensors’, in *6th International Conference on Mobile Computing, Applications and Services*, IEEE, 2014, pp. 197–205.

[47] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha *et al.*, ‘Collecting complex activity datasets in highly rich networked sensor environments’, in *2010 Seventh international conference on networked sensing systems (INSS)*, IEEE, 2010, pp. 233–240.

[48] P. Zappi, C. Lombriser, T. Stiefmeier, E. Farella, D. Roggen, L. Benini and G. Tröster, ‘Activity recognition from on-body sensors: Accuracy-power trade-off by dynamic sensor selection’, in *European Conference on Wireless Sensor Networks*, Springer, 2008, pp. 17–33.

[49] J. Yang, M. N. Nguyen, P. P. San, X. L. Li and S. Krishnaswamy, 'Deep convolutional neural networks on multichannel time series for human activity recognition', in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[50] A. Murad and J.-Y. Pyun, 'Deep recurrent neural networks for human activity recognition', *Sensors*, vol. 17, no. 11, p. 2556, 2017.

[51] F. J. Ordóñez and D. Roggen, 'Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition', *Sensors*, vol. 16, no. 1, p. 115, 2016.

[52] Y. Tang, J. Xu, K. Matsumoto and C. Ono, 'Sequence-to-sequence model with attention for time series classification', in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, IEEE, 2016, pp. 503–510.

[53] V. S. Murahari and T. Plötz, 'On attention models for human activity recognition', in *Proceedings of the 2018 ACM International Symposium on Wearable Computers*, 2018, pp. 100–103.

[54] B. Sun, M. Liu, R. Zheng and S. Zhang, 'Attention-based lstm network for wearable human activity recognition', in *2019 Chinese Control Conference (CCC)*, IEEE, 2019, pp. 8677–8682.

[55] Y. Zhang, X. Li, K. Huang, Y. Wang, S. Chen and I. Marsic, 'Tri-axial self-attention for concurrent activity recognition', *arXiv preprint arXiv:1812.02817*, 2018.

[56] K. Simonyan and A. Zisserman, 'Very deep convolutional networks for large-scale image recognition', *arXiv preprint arXiv:1409.1556*, 2014.

[57] H. Liu, R. Feris and M.-T. Sun, 'Benchmarking datasets for human activity recognition', in *Visual Analysis of Humans: Looking at People*, T. B. Moeslund, A. Hilton, V. Krüger and L. Sigal, Eds. London: Springer London, 2011, pp. 411–427, ISBN: 978-0-85729-997-0. DOI: `10.1007/978-0-85729-997-0_20`. [Online]. Available: `https://doi.org/10.1007/978-0-85729-997-0_20`.

[58] E. Cippitelli, S. Gasparrini, E. Gambi and S. Spinsante, 'A human activity recognition system using skeleton data from rgbd sensors', *Computational intelligence and neuroscience*, vol. 2016, 2016.

[59] C. Chen, R. Jafari and N. Kehtarnavaz, 'Utd-mhad: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor', in *2015 IEEE International conference on image processing (ICIP)*, IEEE, 2015, pp. 168–172.

[60] S. Gaglio, G. L. Re and M. Morana, 'Human activity recognition process using 3-d posture data', *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 5, pp. 586–597, 2014.

[61] J. Sung, C. Ponce, B. Selman and A. Saxena, 'Unstructured human activity detection from rgbd images', in *2012 IEEE international conference on robotics and automation*, IEEE, 2012, pp. 842–849.

[62] A. Trask, D. Gilmore and M. Russell, 'Modeling order in neural word embeddings at scale', *arXiv preprint arXiv:1506.02338*, 2015.

[63] H. Song, D. Rajan, J. J. Thiagarajan and A. Spanias, 'Attend and diagnose: Clinical time series analysis using attention models', in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[64] G. Hu, X. Peng, Y. Yang, T. M. Hospedales and J. Verbeek, 'Frankenstein: Learning deep face representations using small data', *IEEE Transactions on Image Processing*, vol. 27, no. 1, pp. 293–303, 2017.

[65] J. Buolamwini and T. Gebru, 'Gender shades: Intersectional accuracy disparities in commercial gender classification', in *Conference on fairness, accountability and transparency*, 2018, pp. 77–91.

[66] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, 'Densely connected convolutional networks', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[67] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, 'Pytorch: An imperative style, high-performance deep learning library', in *Advances in neural information processing systems*, 2019, pp. 8026–8037.

[68] X. Luo, J.-B. Huang, R. Szeliski, K. Matzen and J. Kopf, 'Consistent video depth estimation', *arXiv preprint arXiv:2004.15021*, 2020.

[69] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan and A. C. Kot, 'Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. DOI: 10.1109/TPAMI.2019.2916873.

[70] N. Kitaev, Ł. Kaiser and A. Levskaya, 'Reformer: The efficient transformer', *arXiv preprint arXiv:2001.04451*, 2020.

[71] I. Beltagy, M. E. Peters and A. Cohan, 'Longformer: The long-document transformer', *arXiv preprint arXiv:2004.05150*, 2020.

[72] S. Wang, B. Li, M. Khabsa, H. Fang and H. Ma, 'Linformer: Self-attention with linear complexity', *arXiv preprint arXiv:2006.04768*, 2020.