Eötvös Loránd University

Faculty of Social Sciences

MASTER'S DEGREE

Distance metric learning using Siamese networks for human pose similarity estimation

Consultant:

Márton Rakovics

Author:

Zsolt Varga

X6R8TK

survey statistics

November 2022

Dedication

In memory of my grandmother, whose love and support got me through so much.

Table of contents

Table of figures

The following is a list of mathematical notations used throughout the thesis.

Set notation

Notation	**Meaning**
$x \in X$	x is an element of the set X
$X \times X$	Cartesian product of the set X
$f: x \mapsto y$	A function f mapping from the domain x to the codomain y
$\mathbb{R}$	The set of real numbers
$\mathbb{Z}_0^+$	The set of positive integers, including 0

Linear algebra notation

Notation	**Meaning**
x	The vector x
x^T	The transpose of the vector x
X	The matrix X
$\|\|x\|\|_1$	The L1 norm of the vector x
$\|\|x\|\|_2$	The L2 norm of the vector x
$\mathbb{R}^n$	The n-dimensional Euclidean space

The following table describes abbreviations used throughout the thesis.

Abbreviation	**Meaning**
AI	Artificial intelligence
ANN	Artificial neural network
BMI	Body mass index
CNN	Convolutional neural network
COVID-19	Coronavirus Disease 2019
DML	Distance metric learning
FFNN	Feedforward neural network
HPE	Human pose estimation
MDS	Multidimensional scaling
ReLu	Rectified linear activation unit
RNN	Recurrent neural network

1. Introduction

1.1. The importance of human motion understanding

The advent of new technologies, specifically Artificial Intelligence (AI), and artificial neural networks (ANN), impacts many fields (e.g., Abiodun et al., 2018). Education and health care (Kreutzer & Sirrenberg, 2020), physiotherapy (Tack, 2019), fitness (Farrokhi et al., 2021), early motor skills development (Vukićević et al., 2019), entertainment (Lachman & Joffe, 2021), and human-computer interaction (Islam et al., 2020) are areas that can benefit significantly from solutions previously requiring full or partial human supervision. All these somewhat disparate domains have one thing in common: human motion is - or can be - involved. What is meant by this is that one can imagine recording a fitness exercise video and getting highly detailed performance feedback. This feedback could mean obtaining a complete analysis of the motions without a fitness or health professional present. Such an analysis could include the visual and semantic correctness of the poses we have taken along with more fine-grained metrics, e.g., the speed of our execution, how deep we have gone during an exercise, and what muscle groups were involved. As another example, we could also imagine being recently discharged patients from a hospital with a musculoskeletal disorder and instructed to follow an exercise routine. How beneficial it would be to have an application to determine whether the execution of said exercises is correct? The data collected during the exercises could be sent to a therapist or physician to adjust the exercises if necessary. A more straightforward example would be navigating our TV or smartphone from a distance or giving a presentation that we can easily control with gestures. The same holds for games or gamified environments where monotonous tasks could be made into pleasant physical activities.

There is another aspect to automatizing such tasks using AI that goes beyond ease of use and comfort. A study conducted in Canada found that the distribution of physiotherapists is uneven (Shah et al., 2019), and waiting lists can be long (Deslauriers et al., 2017). This, in turn, might contribute to lower quality of life and even the development of psychological symptoms

(Deslauriers et al., 2021). However, these findings are mixed, and the authors emphasize the need for high-quality studies. Physiotherapy was reported as often excluded from the statutory benefits package in the European Union, with long waiting times and physical distance between them and the therapist as barriers for many groups (Palm et al., 2021). Apart from issues with availability, physiotherapy practitioners experienced burnout in Poland (Pniak et al., 2021) and Portugal (Jácome et al., 2021) during the COVID-19 pandemic. This is worrying, considering that one study found that the workload of physiotherapists increased dramatically due to breathing complications (Black et al., 2021).

Nevertheless, therapy is a necessity. It can alleviate pain, improve functionality for various orthopedic problems (e.g., Artz et al., 2015; O'Keeffe et al., 2017), and reduce the length of hospital stay (Brusco et al., 2007; Hartley et al., 2016; Henderson et al., 2018). Undoubtedly, the planning and supervision of therapy cannot be replaced by that of a qualified professional; however, a smartphone with a mobile application that can track and evaluate motions could be an invaluable asset for both the patient and the therapist. One could easily imagine that with the removal of barriers such as travel and its costs, lack of qualified workforce, or bulky equipment (Sethi et al., 2020), patients - from the comfort of their homes - would be more willing to adhere to their treatment plan and possibly achieve quicker and better results. This view is further supported by the fact that physiotherapy assessment and telerehabilitation were found to be feasible (Richardson et al., 2017; van Egmond et al., 2018); however, these studies were done on specific patient groups, and the authors emphasize that further research is needed to assess the overall effectiveness. It is worth pointing out that in a survey conducted in Germany, 50% of physiotherapists were in favor of digitalization (Estel et al., 2022). Nonetheless, participation was voluntary, so the results cannot be generalized.

Physiotherapy is just one domain that could greatly benefit from AI. Aiding motor-skills development is another candidate where such technological advances could prove highly useful. Studies revealed that a significant portion of children scored below average on various tests assessing gross motors skills (A. Okely & Booth, 2004; Veldman et al., 2018) or had some kind of motor skill disorder (Nikolić & Ilić-Stošović, 2009). Higher levels of motor skills were linked to lower body mass index (BMI) (A. D. Okely et al., 2004), higher levels of physical activity (Iivonen

et al., 2013), and also self-esteem (Lodal & Bond, 2016). It can be stated that motor skills generally affect the development of domains other than motor behavior, including cognitive and affective abilities, health, and physical growth (Adolph & Hoch, 2020). Considering this, it is not surprising that early interventions are essential and effective in developing fundamental (Brian et al., 2017; Logan et al., 2012) and gross (Veldman et al., 2016) motor skills. Therefore, tools that can improve the assessment and training of early motor skills are valuable.

An innovative example is smart toys (Mironcika et al., 2018) that help both teachers and parents in the early detection of motor skills deficiencies as well as monitoring development while making these activities enjoyable for children. Naturally, children and adolescents love playing games and spend an average 7 hours a day in front of a screen (Rideout et al., 2010), including using smartphones and tablets (Lauricella et al., 2015). A gamified app that entertainingly presents physical exercises could incentivize younger patients to participate in therapy, leading to better results.

AI technology also occupies a natural role in the fitness domain. It is well established that exercising affects both physical and mental functioning. Specific exercises can lower systolic and diastolic blood pressure (Cornelissen & Smart, 2013), especially after training, emphasizing physical activity as a preventive strategy (Carpio-Rivera et al., 2016). It also reduces the risk of cardiovascular diseases, specifically incident coronary heart disease and stroke (J. Li & Siegrist, 2012). Body site-specific exercises were shown to improve or maintain bone mineral density in older men (Kelley et al., 2000), while female children and adolescents benefited from weight-bearing physical activities (Ishikawa et al., 2013). A further health benefit is, of course, weight loss. Regular training reduced the Body Mass Index (BMI) of overweight children, adolescents (Kelley et al., 2014), and adults (Swift et al., 2014) while also increasing muscle volume (Schoenfeld et al., 2017).

Exercising has proven to reduce perceived stress (e.g., Della Valle et al., 2020; Goldstein et al., 2020; Starkweather, 2007) and improve the quality of life in healthy and rehabilitation patients (Gillison et al., 2009). Moderate intensity, weekly aerobic exercises demonstrated a sizeable antidepressant effect, regardless of symptom severity (Morres et al., 2019), reduced psychiatric

symptoms, and enhanced functioning in patients with schizophrenia (Firth et al., 2015). Moreover, it seems viable to use exercise in dealing with the depressive phase of bipolar disorder (Thomson et al., 2015). Training is also associated with improved body image (Hausenblas & Fallon, 2006), self-worth, and self-concept in children and adolescents (M. Liu et al., 2015). Studies show that combined acute and long-term cardiovascular exercises can have a moderate to significant effect on short-term memory (Roig et al., 2013) while also having a positive effect on attention, executive functions, and academic performance in preadolescent children (de Greeff et al., 2018). This list of benefits is non-exhaustive and merely serves examples demonstrating the importance of physical activity.

As with the previous examples regarding physio and motor skills therapy, there are benefits in integrating advances in AI into fitness and sports activities as a means to incentivize and further educate people. Many fitness applications are available (Herrmann & Kim, 2017) and studies demonstrated that primary school children showed higher intrinsic motivation and interest in exercising using such an application (Papastergiou et al., 2021). College students use them to achieve specific goals (Gowin et al., 2015), which include the desire to regain past fitness, improved appearance, improved well-being, stress-relief, mental health benefits, competing with others through self-challenges (Molina & Myrick, 2021). Evidence also suggests that applications help individuals achieve their health and fitness goals while also analyzing and summarizing their data to provide personalized feedback (Higgins, 2016). Considering the high interest and value of these apps it is not surprising that various AI solutions were incorporated into them to enable more sophisticated features. Deep learning architectures such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) were used for human activity (Nithyakani & Ferni Ukrit, 2022), motion (Niu, 2021), and posture recognition (R. Yuan et al., 2021) based on video input. It is also possible to deploy deep learning models on smartphones. Using a CNN architecture, Shrestha & Won (2018) accurately estimated walking speed solely from smartphone sensors while Park et al. (2022) successfully deployed a similar architecture to give enhanced real-time exercise feedback.

These topics are closely tied to human pose estimation (HPE) which can form the basis of motion and action recognition (Difini et al., 2021; Song et al., 2021). Human pose estimation is concerned

with detecting and classifying human body parts. This in practice means returning the 2D or 3D coordinates of the major joints, such as wrists, elbows, shoulders, hips, knees, and ankles, as well as the location of the eyes and ears. The position of the limbs and outlines of the torso can be trivially determined, if needed, by simply creating matching joint pairs. For example, the right lower arm is the part of the body that starts at the right elbow extending to the right wrist (Munea et al., 2020; J. Wang et al., 2021). The number of points one wants to estimate can, in theory, be arbitrarily large; however, the joints mentioned above already give a decent representation of the human body. One must remember that a more granular detection allows for a more detailed representation. An arched back could not be detected without information on the location of the thoracic vertebra, for instance.

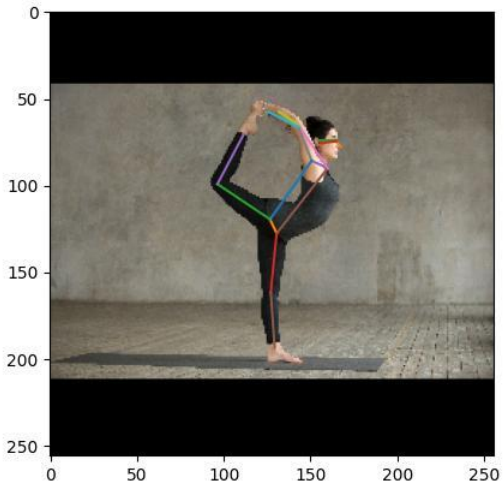

Figure 1. A woman practicing the Natarajasana yoga pose with the detected joints and pose overlay on top. The bent back cannot be detected without additional key points.

Nowadays, many people worldwide have access to smartphones (*Demographics of Mobile Device Ownership and Adoption in the United States | Pew Research Center*, 2021) with high computational capacity and good video capture capabilities. This allows for a practical use case of HPE that uses image data, more specifically video (C. Li et al., 2022), to return an estimate of the human body. Requiring as little as a phone enables an enormous number to use apps based on such technology.

An HPE-based system designed to tackle the previously mentioned goals implicitly requires at least two steps to achieve good results. The first is estimating the human body with high accuracy while the second is making sense of this information. The latter depends upon our goals regarding what to achieve. This is best illustrated with a simple example. Image that one would like feedback on how well a particular motion was executed in a physiotherapy exercise. For the sake of this example, imagine that it is enough that the patient takes three different positions, and the quality of the execution only depends on how well he or she can mimic these positions. One can imagine that a human therapist would require the patient to extend their legs or arms in a certain way. He or she would look at the patient's posture and compare it with a given idealized posture that should be taken. If it is dissimilar from the expected one, then the therapist would give instructions to correct it. Of course, this is a very simplified example and narrows down the very complex topic of rehabilitation to its most essential components, yet it can suffice as a basis for a very simple app that could at least assist the therapist. While comparing two poses is effortless for a human, it is not a trivial task for a machine. What is similarity in this case? How do we define it? This brings us to a critical point that this thesis addresses: the determination if any two poses are similar.

At first, this question may seem of little importance, but one can argue that it is fundamental in any system that operates on human poses. Health and fitness use cases such as physiotherapy and interventions aiding motor skills development as well as fitness exercises are key examples of pose similarity requirement as knowing how similarly the patient executed a specific action compared to a reference is of fundamental importance. It is easy to give examples from other domains as well. If one would like to design an app that operates on various gestures as commands, it is also crucial to correctly detect key poses with high accuracy and match those poses to expected ones. This problem becomes more pronounced when considering that nuanced differences can have entirely different meanings. Considering these problems and the potential use cases, similarity comparisons between human poses is an essential topic.

1.2. A problem with naïve approaches

One could argue that most people could acquire an expert level of knowledge in a field if they invested time and effort to do so. Thus, designing a system to perform similarity matching seems feasible. Nevertheless, implementing an algorithm that measures pose, or motion similarity is not straightforward.

This thesis was inspired by a workplace project, where various methods were developed to tackle the issue of human pose understanding. Unfortunately, the information regarding this research is confidential and therefore cannot be disclosed. Nonetheless, we can illustrate the most common problems with a few simple examples and explain the general issue with simpler approaches.

In some cases, simple metrics and algorithms such as the Euclidean or cosine distance, k-means clustering (Sinaga & Yang, 2020) or k-nearest neighbors (Cunningham & Delany, 2021) can suffice. However, these approaches cannot perform well with more complex tasks or when the data has different meanings. We can imagine using the Euclidean distance to compare two poses. If these poses are visually similar, this metric works well; however, the coordinates describing the joints' locations can change while the underlying semantic meaning stays the same. This can easily happen by simply changing the viewpoint. A downward-facing dog yoga stance is the same whether we observe it from the sides, the top, or from behind, yet the coordinates describing it are completely different. Consequentially, algorithms that rely on the Euclidean distance would also fail. It can also happen that the knowledge of certain joint locations is not necessary. While designing an upper-body exercise, an application could be completely indifferent as to where the legs are. Similarly, we might want to be somewhat forgiving when certain joint positions are off. A squat might be executed correctly, even if the knees are slightly off from their ideal position. Lastly, it can happen that we do not care about which side of the body a certain motion was done to. A leg raise is still a leg raise regardless of whether the subject is using their left or right leg. With this last example it is clear to see why using the Euclidean distance would be impractical as an application would measure a considerable distance between two poses due to the mirrored location of the joints.

To mitigate these issues, we would like to introduce deep similarity learning, which is considered a state-of-the-art method for determining similarity at the time of writing this thesis. This chapter will first define similarity learning and metrics, or similarity functions. This will be followed by an introduction to Siamese neural networks and contrastive loss. Finally, a more advanced version of Siamese networks will be detailed using the so-called triplet loss.

To understand how the similarity comparison problem mentioned above can be solved with state-of-the-art approaches, we must first understand what an Artificial Neural Network (ANN) is and how it operates. However, we must acknowledge that neural networks cover a vast topic, and it would be infeasible to summarize all their aspects herein. For this reason, in the following section, we will briefly introduce the essential mathematical foundations of neural networks - specifically feed-forward neural networks - that are crucial to understanding the upcoming topics. We will also introduce the reader to the terminology used in academia that encapsulates and abstracts complex ideas in statistical and deep learning.

This section begins with a brief introduction to biological neural networks, which inspired ANNs and often motivate new research in the field. Then we will guide the reader through the general setup of statical learning, introducing loss and activation functions and optimization. This is followed by explaining the building blocks and the architecture of a simple Feed-Forward Neural Network (FFNN), such as the neurons, layers, and the connections between them. These foundations are enough to cover other topics, such as metric learning, which will be used for similarity comparisons.

1.3. Biological inspiration

Just like AI and ANNs, neuroscience and neuroanatomy are equally broad fields that are impossible to cover in just a few sentences. However, it is helpful to introduce the most basic terms to bridge the gap between some mathematical constructs and their biological analogues and in doing so clarify the analogies of these terms with respect to their use in the field of AI.

One of the most fundamental building blocks in biological neural networks, e.g., the human brain, is the neuron. The human brain likely contains around 100 billion neurons, although estimates vary about the precise number (von Bartheld et al., 2016).

The basic building blocks of the biological neuron are the cell body (or soma), dendrites, and the axon.

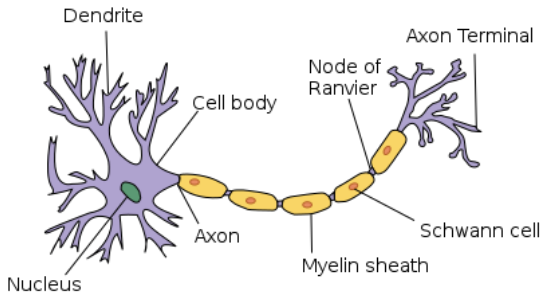

Figure 2. Image of a biological neuron. Adapted from Wikipedia Commons, by User:Dhp1080, 2019, Retrieved June 17, 2022, from https://commons.wikimedia.org/ wiki/File:Neuron.svg. Copyright 2019 by User:Dhp1080. Adapted with permission.

While neurons have many dendritic endings, research indicates that they all have only one axon, meaning they only connect to one other neuron (Andersen & Bi, 2000). The dendrites are connected to other neurons' axons, enabling communication between the cells. Neurons receive signals from other neurons through the dendrites and send signals through the cell's body to the axon. This signal is an electric impulse that releases chemicals in the axon terminals called neurotransmitters into the synaptic gap. The synaptic gap is a space between the end of a neuron's axon and the dendrite of another neuron. The neurotransmitters bind to specific receptors on the dendrites of the receiving neuron. These receptors can be excitatory or inhibitory. An excitatory receptor promotes the generation of electric signals, while an inhibitory

one impedes it. If the excitation reaches a certain level, the neuron will fire, sending an electric impulse along its axon. The firing is a binary event; the neuron either sends a signal or it does not. All the neurons in a system form a complex and interconnected network which is the foundation of many astounding capabilities such as memory, vision, attention, speech, to name just a few.

While neural networks appear straightforward considerable research and development is required to enable ANNs to achieve human-like performance on many of these tasks. CNNs are usually deployed to tackle computer vision tasks such as object detection and image recognition (Ajit et al., 2020), while RNNs are used in tasks involving sequential data such as speech recognition (Amberkar et al., 2018). At present, one of the most important concepts that enabled ANNs to gain better performance was the introduction of the so-called attention mechanism, which in turn was inspired by the attention characteristic in humans (Niu et al., 2021).

Similarity matching in humans also has its roots in neurology. Knowing if two things are similar is seemingly effortless for the human brain (Hahn et al., 2003). We can quickly identify a friend, recognize a face, or know that two people are close relatives based on certain features. Face recognition is especially important. So much so that there is a dedicated region in the brain, the fusiform face area, which plays a crucial role in face perception (Kanwisher & Yovel, 2006). It also appears to be a highly specific and heritable characteristic (Wilmer et al., 2010), the lack of which is considered a cognitive disorder called prosopagnosia (Corrow et al., 2016). It is not only faces that humans can recognize easily, nor is face-recognition the only ability supported by an underlying neural region. It is also the case with human motion and action understanding. Almost anyone can tell if an arm is stretched out or bent at a 90-degree angle, whether it is done to the left or right. It is equally simple to determine if another person is doing jumping jacks or squats. Although the reason is debatable, the biological underpinnings for this mechanism may be the mirror neuron system wherein specific neurons fire when a person is performing a particular action as well as when the same action is only being passively observed (Cook et al., 2014). This system might be the basis of imitation with neuroimaging studies, having identified multiple responsible brain regions such as the dorsal premotor cortex (Molenberghs et al., 2009).

1.4. Supervised statistical learning

The field of machine learning can be broadly divided into three categories: supervised learning, unsupervised learning, and reinforcement learning. Herein, supervised learning is discussed in detail based and is based on the book *The nature of statistical learning theory* (Vapnik, 1999).

Generally, three interconnected components illustrate statistical learning. Firstly, there is a generator G that produces independent and identically distributed $(i.i.d.)$ n dimensional random vectors from a certain probability distribution $F(X)$. Secondly, a supervisor S evaluates these random vectors and returns some value y. Lastly, there is a learning machine LM that implements some parametrized function $g(x, \theta)$ which aims to approximate the outcome of the supervisor. In other words, the function g produces $\hat{y}$ approximating y.

$$y \approx \hat{y} = g(x, \theta)$$

Our goal is to find g from the possible set of functions G_f that best approximates the supervisor. In short, we are trying to fit a model that best describes our data. As we will see, what is best will depend on our use case and objectives, and certain optimization criteria could lead to different results.

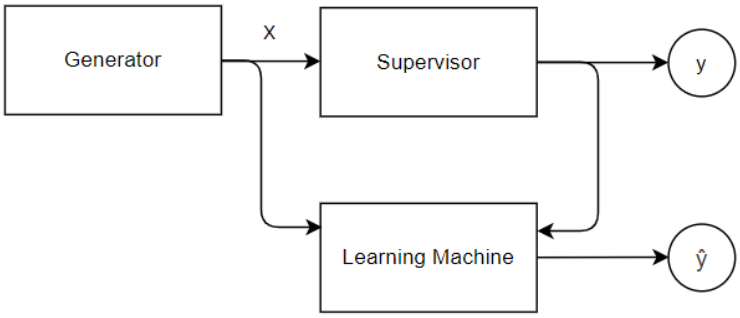

Figure 3. A model of statistical learning adapted from The nature of statistical learning theory (Vapnik, 1999, p. 18).

In a so-called supervised learning setting, we have access to (x, y) tuples where x is the previously mentioned random vector and y is the "correct answer" or ground truth value. In our

case, we can imagine that x is a vector of joint coordinates and y is 1 if this pose is a squat and 0 otherwise. We will call this the training set containing n samples and denote it as

$$T_n = \{(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)\}$$

which in practice is sampled from the joint distribution of X, Y

$$F(X, Y) = F(X)F(Y|\mathrm{X})$$

The ground truth value y is necessary to quantify the goodness of the output of g. This is done by computing the difference between y and $\hat{y}$ according to a loss function L.

$$L(y, \hat{y}) = L(y, g(x, \theta))$$

The number of commonly used loss functions in machine learning and deep learning is high. In their research paper, Q. Wang et al. (2022) described 31 classical loss functions used in classification and regression problems as well as unsupervised settings. The goal of this thesis, however, is not to give an exhaustive description of loss functions; therefore, we only discuss a handful of fundamental ones.

A commonly used loss function is the squared difference. It can be applied when $y, \hat{y} \in \mathbb{R}$ and is defined as

$$L_{squared\ difference} = (y - \hat{y})^2 = \big(y - g(x, \theta)\big)^2$$

Another well-known loss function is the binary cross-entropy

$$L_{binary\ corss-entropy} = -y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$$

for cases when $y \in \{0, 1\}$ and $g: \mathbb{R} \mapsto [0, 1]$, thus $\hat{y} \in [0, 1]$.

The binary cross-entropy is just a special case of the cross-entropy loss used for multiclass prediction; cases where a model needs to predict if a data point belongs to one of K classes. In such instances, we can treat each class as a separate binary classification task and simply sum up the losses.

Generally, we can write this as:

$$L_{cross-entropy} = -\sum_{i=1}^{K} y_i \log(\hat{y}_i)$$

For our purposes, it is necessary also to introduce the hinge loss defined as

$$L_{hinge} = \max(0, 1 - \hat{y} \cdot y)$$

used when $y \in \{-1, 1\}$ and $\hat{y} \in [0, 1]$.

More nuanced loss functions such as the positive pairwise loss and the triplet loss will be introduced in upcoming chapters and are good examples of how important the choice of an adequate loss function is.

Intuitively, we would expect that if the loss was small, the predicted value is close to the ground truth value, whereas if it is big, the prediction is inaccurate and considered incorrect. High and low values for the loss are – again – dependent on the use case and the scaling of the data. For example, if one is trying to estimate someone's height and uses the squared difference loss to determine the accuracy of the estimate, one must not forget whether meters or centimeters is used in the calculation. A value of 1 can be considered excellent if we use the latter, and considerably poor in the case of the former.

It should be noticed that the loss function operates on one ground truth and one predicted value; however, it would be ideal to find an optimal approximator function for the entirety of the training set. We would like to find an approximator function which models previously unseen data, that is x which was not in T_n . For this reason, we introduce the theoretical risk function $R(g)$, defined as the expected loss.

$$R(g) = \int L\big(y, g(x, \theta)\big) dF(X, Y)$$

As with the loss, we expect this function to return a low value if a well-parametrized estimator function has been found. The crucial step to achieve this is minimizing the risk function, but since practically $F(X, Y)$ is unknown, we need to approximate it with the data we have using T_n.

Consequently, we optimize the empirical risk function defined as

$$R_n(g) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, g(x_i, \theta))$$

using an optimization algorithm, such as gradient descent (Ruder, 2017). Minimizing the empirical risk function and finding the optimal θ is called model training or learning. In academia and business setting, the term loss is used interchangeably to describe the risk and empirical risk functions. In the data analysis part of the thesis, we will also use the term loss to describe the empirical risk, due to its brevity.

For our purposes, it is necessary to briefly introduce two types of supervised statistical learning problems. The first is classification, related to the previous example with the squat. Put simply, our goal is to categorize the data into distinct classes. A typical example is predicting if an animal is a dog or a cat if a face matches the phone's owner or deciding if a sequence of human poses was a jumping jack. We can immediately see from these examples that a classification task can be binary or non-binary. For now, it is enough to discuss binary classification problems. In such settings, the return value y of the supervisor is customarily 0 or 1, and g is an indicator function, implying that G_f is the set of indicator functions.

The second type is regression, where the outcome y of the supervisor is a real value, consequently G_f is a set of real-valued functions. A simple and typical example is housing prediction or determining a child's future height based on the height of his or her parents, as well as other environmental factors.

Interestingly, HPE encompasses both approaches. Given an image, we would like to determine the location of each joint. While returning the position of the joints or coordinates in the image is a regression task, identifying the joints is a classification one.

1.5. Neural networks

1.5.1. The perceptron

To further understand how an ANN operates, we must introduce its building block and historically earliest element, the perceptron. The perceptron was first proposed by Rosenblatt (1958) as a binary classification algorithm to determine whether an input belongs to a specific class. It is a function, g such that:

$$g: x \mapsto y, \qquad x \in \mathbb{R}^n, y \in \{-1, 1\}$$

In other words, the input is an n dimensional real-valued vector x, and the output is a scalar. By convention, the target class is assigned the value 1.

The function g is defined as:

$$g(x, w, b) = sign((w^T x) + b)$$

As we can see the perceptron takes the linear combinations of the input values with some weights to which it adds a bias term. This value, in turn, is then fed into the sign function. As we will see shortly, the sign function can be considered a so-called activation function with a unique role in neural networks.

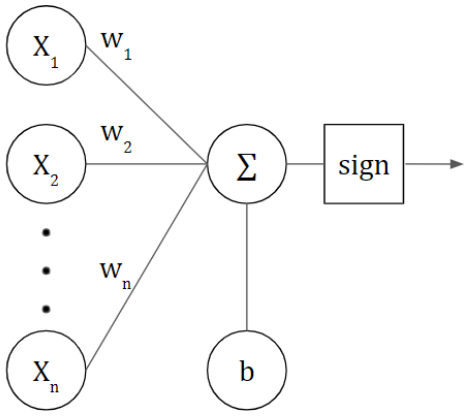

Figure 4. A graphical illustration of the Rosenblatt perceptron for an n-dimensional input.

The above expression carries with it an exciting and critical geometric interpretation, as the formula

$$w^T x + b = 0$$

can be thought of as an equation to a hyperplane. Continuing with our example, let us imagine that we have m poses, each represented by an n dimensional vector in $\mathbb{R}^n$. For convenience, this can be achieved by transforming the x, y coordinate pairs for the joints into a flattened vector. Let us assume that a certain number of these poses are ones that we want to detect, for example, the lotus sitting position. The rest are various other poses. This leaves us with two disjoint sets, one with our target pose and one with all other poses. The perceptron then attempts to find an $n - 1$ dimensional hyperplane that separates these two sets in the n dimensional space spanned by the input vectors. In this scenario, the hyperplane is called the decision boundary. The data points belonging to the target class will have a positive value as they are "above" the decision boundary.

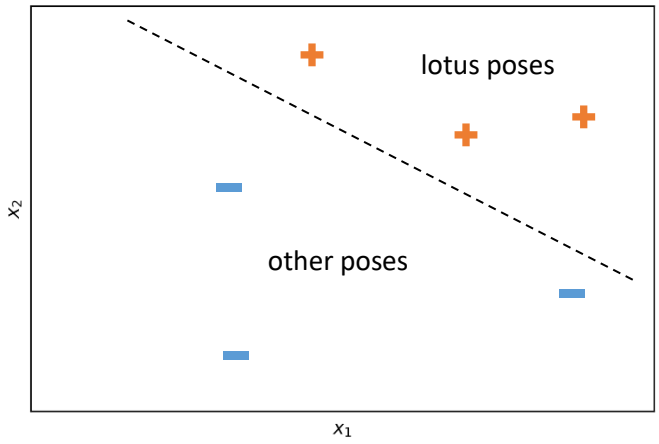

Figure 5. Illustration of the two-dimensional input space X and the hyperplane (dashed line) separating the data points. The data points can be considered n-dimensional points or vectors in the input space.

If the perceptron (or any other classification algorithm) can find a hyperplane that achieves perfect separation, meaning that each element of the respective set lies on either side of the hyperplane, we say that the two sets are linearly separable. However, this is usually not the case

when working with real-world data. The goal is to find a hyperplane that separates most data points well. In our example, this would mean the points belonging to the lotus pose would fall on the positive side of the hyperplane while all other points are negative.

Knowing the decision boundary supposes we already know the parameter vector or weights that determine the hyperplane. As discussed earlier, this is the exact thing we would like to achieve. The specific steps in which the perceptron learns these weights and bias terms do not strictly belong to the topic of this thesis, but it can be proven that they can be found in finite steps by iterating through the data points.

The next fundamental step toward modern deep learning models was the multilayered perceptron, which in modern terminology is a FFNN mentioned previously, and which will be discussed in the following chapter.

1.5.2. General structure

Having introduced the fundamental goal of statistical learning, loss functions, risk minimalization, and the perceptron, we can now turn our attention to neural networks. Before delving into the topic, it is helpful to introduce a few terms that help clarify parts of such a complex system. The basic architecture of an ANN consists of layers of interconnected neurons. A neuron, in simple terms, is a node through which data can flow. It receives an input to which it applies a transformation. This transformation is a linear combination of the inputs with certain weights and a bias term, typically followed by some nonlinear transformation. These nonlinear transformations are called activation functions and are highly important. As we can see from this description, a neuron is simply a mathematical function, or rather a concatenation of two functions.

Let z denote the linear combination with the added bias term:

$$z = w^T x + b, \qquad x, w \in \mathbb{R}^n \; ; \; b, z \in \mathbb{R}$$

and f an activation function such that:

$$a = f(z), \;\; a \in \mathbb{R}$$

then let a denote the activation of a single neuron which is just a more specific term for its output. As one can see, the activation of a single neuron is a scalar.

This fits well with the perceptron model and a neuron's biological bases. Furthermore, the graphical depiction of an artificial neuron is almost identical to that of the Rosenblatt perceptron, the only difference being that the activation function is not necessarily the sign function. We can also see how the artificial neuron resembles a biological one. The incoming data can be considered the dendrites while the transformation functions the cell body. Like its biological counterpart, the artificial neuron also has one output, resembling the axon. A crucial difference, however, is that in ANNs, neurons connect to many other neurons and their output value is not necessarily binary. Using the terminology introduced in the biological example, they do not just "fire or not fire" but produce a more detailed response.

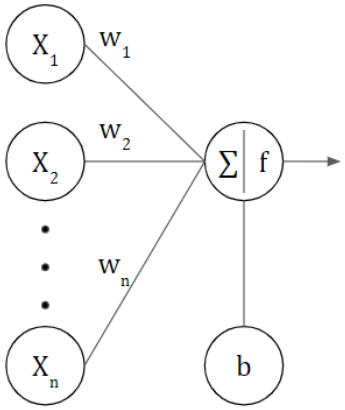

Figure 6. An image of an artificial neuron which, in essence, is a mathematical function.

Many individual neurons form a layer of which there can be multiple in an ANN. Furthermore, each layer can have any number of neurons.

A neural network has three crucial layers. The first one is the input layer which is simply the input data represented as an n-dimensional vector. This is fed into so-called hidden or intermediate layers. We call them hidden because their outputs are not directly observable. Of course, this does not mean we could not obtain the values if we wanted to; it just implies that these values

are usually not of interest. However, we will see in later chapters that it can be highly valuable to access and use the output of these intermediate layers in some cases.

In most cases, one is primarily interested in the output layer. For example, in a binary classification problem, this last layer contains one neuron, which returns a value between 0 and 1. This value is the probability of our input data belonging to the target class. The output layer can also contain multiple neurons if we face a non-binary classification problem. It is important to mention that the output value can also be real-valued, implying that neural networks can be used to perform regression tasks, a feature we will also rely on later in this thesis.

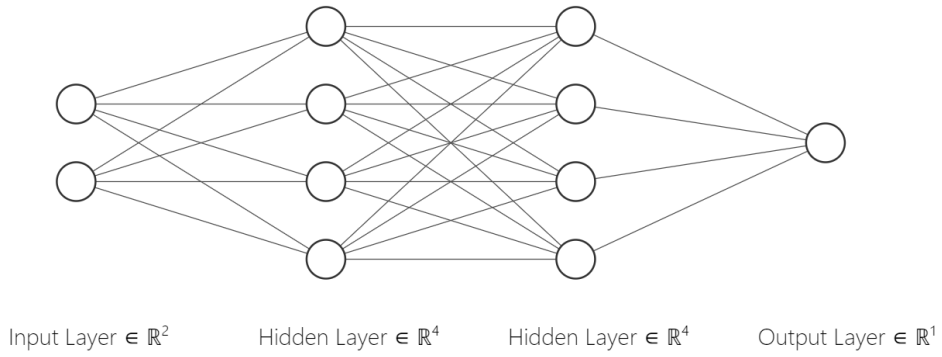

Figure 7. An illustration of a simple feed-forward neural network with two hidden layers.

While it is helpful to imagine the network as a series of interconnected nodes, one must also consider the underlying mathematical equations. As discussed earlier, the nodes are functions, and the connections are weights or scalars. As with a single neuron, it is possible to represent the operations in the layer for all neurons in matrix algebraic form.

Let $l \in \mathbb{Z}_0^+$ denote the index of a layer, then:

$$z_l = W_l^T a_{l-1} + b_l$$

where $z, b \in \mathbb{R}^n$ and W is an $n \times m$ matrix. Furthermore:

$$a_l = f(z_l),\ a_l \in \mathbb{R}^n$$

It is important to emphasize that a_l, z_l and b_l are no longer scalars but vectors. As mentioned before $a_0 = x$, or said otherwise, the input data can be considered the zeroth activation layer.

To reiterate, the output or activation in each layer is an n dimensional vector. This activation is fed into the next laxer. Since the number of neurons in each layer is arbitrary and every neuron is connected to everyone in the previous layer, a weight matrix is needed where n is the number of neurons in the next layer $l+1$ and m is the number of neurons in the current layer l. Consequently $a_{l+1} \in \mathbb{R}^m$. It is worth noting that m can equal n.

Finally, let us note that the whole network can be viewed as a concatenation of many functions from the first to the last layer.

$$\hat{y} = f(W_l^T f(W_{l-1}^T (f(W_{l-2}^T(\dots) + b_{l-2})) + b_{l-1}) + b_l)$$

Using our earlier notation, this can be simply denoted as $\hat{y} = g(x, \theta)$.

The number of layers, the number of neurons in each layer, and the type of connection define the network architecture. The architecture can also vary depending on the task the network is intended to solve. The simplest case is a fully connected feed-forward neural network where every neuron in a layer connects to every other in the previous and following layers. Convolutional neural networks can deal with image data where the input is a tensor rather than a vector, while sequential neural networks can handle temporal dependencies. In some networks, certain layers are connected to their neighboring ones and others down the line. Still other networks still use parallel layers with shared weights, an architectural choice that will be introduced in more detail later.

1.5.3. Activation functions

Apart from the architecture, it is essential to briefly discuss the most important activation functions as they are key to the network's accuracy (Sharma et al., 2020) and influence the learning speed (Kamruzzaman & Aziz, 2002). The latter property will be discussed in more detail later, but for now, it is enough to know that finding the optimal weights for a network involves calculating the derivatives of the activation functions.

As mentioned previously, an activation function f takes the linear combination of inputs z and transforms it into a scalar a. They are sometimes referred to as squashing functions because they typically have a limited range, usually between 0 and 1 or -1 and 1; or sometimes 0 and infinity.

Many activation functions are nonlinear; thus, their role is to add non-linearity into the system. This is needed because, in most real-world cases, the network must adapt to a nonlinear input change (Sharma et al., 2020).

As we saw with the perceptron, one choice can be the sign function defined as

$$sign(z) = \begin{cases} 1 & if\ z > 0 \\ 0 & if\ z = 0 \\ -1 & if\ z < 0 \end{cases}$$

however, this is seldom used in modern neural networks.

A more frequently used activation function is the sigmoid

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

which transforms the input into the open interval $(0, 1)$. We can see this activation function in the output layer of networks, specifically in binary classification tasks.

Another common choice is the hyperbolic tangent

$$\tanh(z) = \frac{e^{2z} - 1}{e^{2z} + 1}$$

which is centered around 0 and has a range of $(-1, 1)$. It has the excellent property that its derivatives are steeper, which makes learning more efficient (Sharma et al., 2020). It is worth noticing that just like the sign function, this can result in a different sign for the output values. The tanh function was also identified as a hardware-friendly solution for CNNs (Abdelouahab et al., 2017).

A more modern activation function is the ReLU (Glorot et al., 2011) which is short for REctified Linear Unit.

$$ReLU(z) = \max(0, z)$$

An advantage of using ReLU is that a neuron will be “deactivated” if the input is equal to or less than 0, resulting in faster learning. This sparsity is more robust against slight changes in the input

data. Nonetheless, this also implies that weights and biases for these neurons are not updated during learning which can create bias. These issues can be mitigated by using variants of ReLU, such as Leaky ReLU (Maas et al., 2013).

Finally, we need to introduce the SoftMax activation which is primarily used in the final layer of multiclass networks:

$$SoftMax(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}} \; for \; i = 1, 2, \ldots, K$$

The SoftMax activation can be understood as assigning a probability that a data point belongs to one of many classes. In other words, if the network predicts K classes, the output of the SoftMax function is a K-dimensional vector, where each value at index i represent the probability that the input belongs to the kth class (Sharma et al., 2020). To give an example, if a network classifies 3 pose classes, e.g., lotus-pose, plank, and push up each input will get a probability value for all the classes. Logically, the index with the highest value is selected as the prediction. In an ideal scenario, a push-up will get a high probability of belonging to the push-up class, and a low probability for the remaining two classes.

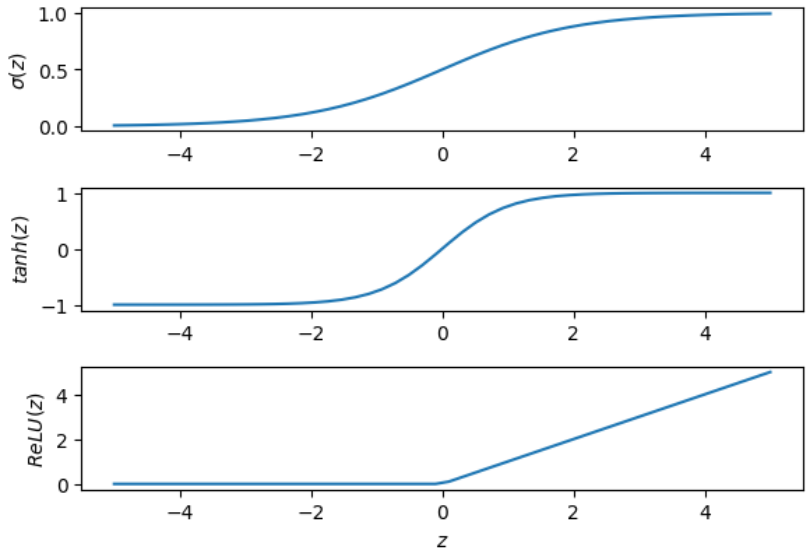

Figure 8. Graphs of the sigmoid, hyperbolic tangent, and ReLU activation functions.

The above-described list of activation functions is by no means exhaustive. In their survey, Apicella et al. (2021) identified multiple dozens of such functions while also providing a taxonomy highlighting the complex approaches developed to build more sophisticated networks. Many modern activation functions are themselves trainable.

1.5.4. Gradient descent

Having described the basic building blocks of neural networks, we can see how we can give a prediction for an incoming data point. Suppose somebody gave us a neural network that accepts pose coordinates as inputs and returns a value between 0 and 1 where numbers closer to one indicate if this pose is a squat. In this scenario, the input data is our feature vector x which also equals a_0. This is fed into the neurons in the next layer, where we take a linear combination of these values with weights and add a bias term vector b producing z which in turn is fed into some activation function f producing a_1. This process continues through the network until the last layer with one neuron using a sigmoid activation function that gives $\hat{y}$.

Upon using the network, we notice that it gives very accurate and reliable predictions. This means it always indicates if a particular pose is a squat and never does so when it is something else. This means that the neural network already knows the optimal weights, bias terms, and best fitting function. The challenge is precisely this; knowing which parameters produce the best results. Of, course, one is not provided with a parameterized network; one must therefore learn these parameters by training the network.

As discussed earlier, learning essentially means optimizing the empirical risk function. In other words, what happens during the training of a neural network is finding parameters for which the risk is minimal. Since, in most cases, it is not possible to have a closed-form analytical solution, various iterative optimization algorithms were developed for finding θ, such as gradient descent.

Let $R_n(\theta)$ be the empirical risk function we would like to optimize. As discussed previously, this involves finding θ^* that minimizes the empirical risk or average loss. In some cases, it may be possible to obtain θ^* through a closed-form analytical solution. A closed-form solution means

that one could obtain the optimal parameters by substituting the input values into an equation derived analytically. An example is a solution to linear regression

$$\theta^* = (X^T X)^{-1} X^T y$$

As we can see here, obtaining θ^* can always be achieved by solving the equation. However, this is not possible in many cases, especially with neural networks. To overcome the problem of not having a closed-form solution, we can opt for an iterative algorithm, such as gradient descent (Ruder, 2017), defined as:

$$\theta_{i+1} = \theta_i - \alpha \frac{\partial}{\partial \theta_i} R_n(\theta_i), \qquad \alpha \in \mathbb{R}, \theta \in \mathbb{R}^n$$

or more concisely

$$\theta_{i+1} = \theta_i - \alpha \nabla R_n(\theta_i).$$

As we can see, a new parameter estimate is produced by subtracting the scaled derivative of the objective function from the value of the previous parameters. This is repeated until convergence is achieved. An intuitive explanation of the algorithm is that we are trying to go down a hill to reach the lowest possible point by taking a step toward the highest point, as shown in the figure below. In essence, we are finding the point in the parameter space for which our loss is minimal. This can be easily illustrated for a one-dimensional parameter space with a convex risk function.

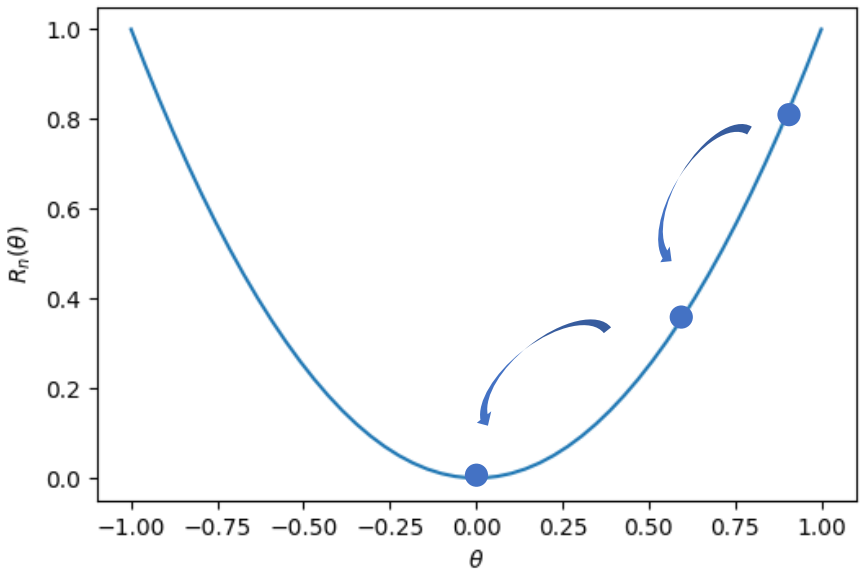

Figure 9. Illustration of the risk as a function of the parameter. The goal is to find the parameter value that yields the lowest risk. The gradient descent algorithm achieves this by calculating the derivative of the risk function and iteratively moving towards the direction of greatest descent.

The step size is heavily influenced by the derivative itself, and the scalar α is commonly referred to as the learning rate. The derivative can sometimes be either too low or too high. In the former case, gradient descent converges very slowly since the value $\frac{\partial}{\partial \theta_i} R_n(\theta_i)$ used for the update is also tiny. This is called the vanishing gradient problem (Tan & Lim, 2019). In the latter case, the estimate may oscillate as the algorithm takes too large steps and "overshoots" the minimum. A one-dimensional parameter space with a convex objective function served only as an illustration. Usually, in complex neural network architectures, the risk function is non-convex. This means there are potentially many local minima where the algorithm can get stuck, and the parameters are not further updated. This, in essence, leaves us with a suboptimal model. A helpful analogy is imagining a ball rolling downhill. We want the ball to reach the lowest point possible; however, the terrain can be configured in such a way that the ball might lose momentum, stopping on a level surface.

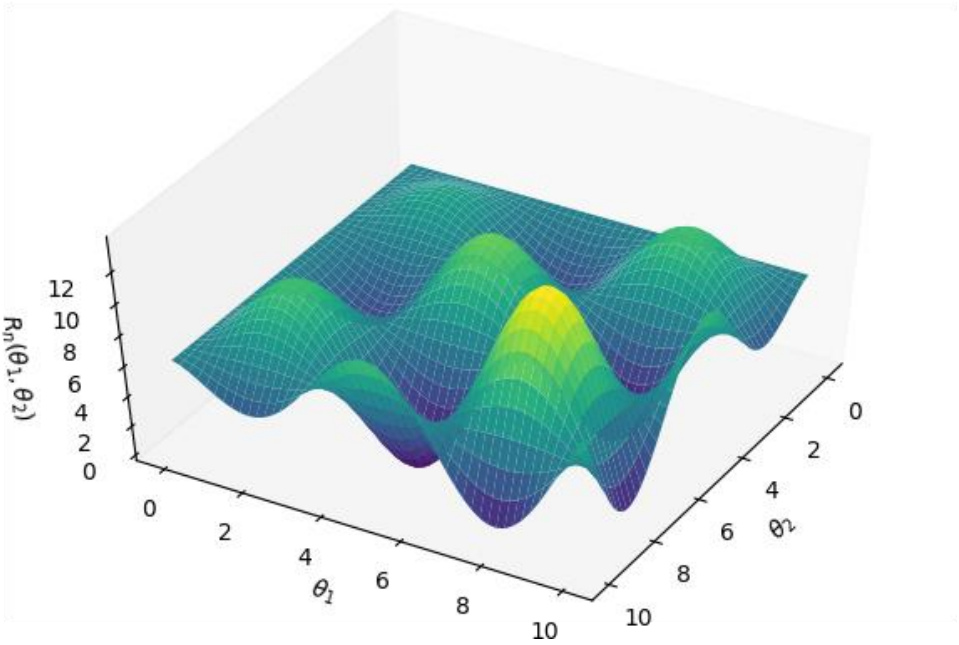

Figure 10. Illustration of a hypothetical non-convex risk function. There are multiple local optima where gradient descent can get stuck, stopping the convergence and yielding suboptimal parameter estimates.

A solution to this problem is using a slightly different optimization algorithm, such as RMSProp (Kurbiel & Khaleghian, 2017), Adam (Kingma & Ba, 2017), Adagrad (Lydia & Francis, 2019) or Nadam (Dozat, 2016). These algorithms can overcome getting stuck in local minima. Continuing with our previous analogy, this is like how a ball would gain momentum while rolling downhill, using that momentum to roll out of a pitch to continue its descent.

Another challenge arises from modern neural networks having many input variables, so they might need to estimate millions of parameters and operate on large amounts of data. Due to this, gradient descent can become computationally challenging, if not entirely unfeasible. Even powerful computers with lots of memory access could take days to train the network. For this reason, an alternative approach, stochastic gradient descent, is used (Bottou, 2010). This does not modify the algorithm itself but rather the input data. Instead of using all the available data points for calculating the gradient for updating the parameter, we group the data into batches and update the parameter using a smaller amount of data in the batch. Processing all batches

and consequently using all data is called an epoch. Training a neural network can involve many epochs.

Updating the weights in a computationally effective way is commonly done with the backpropagation algorithm that computes the loss, layer by layer (Rojas, 1996).

This concludes the most fundamental mathematical underpinnings of artificial neural networks. As mentioned at the beginning of this chapter, the area is vast and active research is undertaken to tackle specific domain-related tasks or simply to improve our theoretical understanding. Nevertheless, the previously covered topics are enough to advance this discussion into more focused parts.

1.6. Few-shot learning

Neural networks, and especially deep neural networks, have gained much popularity in recent years because they can solve many complex problems, a vast majority of them on a near-human level. Some examples can be found in adult and infant pose estimation (e.g., Groos et al., 2022; Kaplan et al., 2021). Sometimes they can even outperform human experts in specific tasks such as medical diagnosis (e.g., Zhou et al., 2021).

This should not come as a surprise since "*standard multilayer feed-forward networks are capable of approximating any measurable function to any desired degree of accuracy, in a very specific and satisfying sense. We have thus established that such "mapping" networks are universal approximators. This implies that any lack of success in applications must arise from inadequate learning, insufficient numbers of hidden units or the lack of a deterministic relationship between input and target.*" (Hornik et al., 1989, p. 363)

This implies that, if needed, one could stack a vast number of layers to solve a challenging machine-learning problem successfully. However, this comes at a cost. Deep learning models sometimes need to estimate millions of parameters to achieve good performance (Y. Yuan et al., 2020); thus, training a supervised deep neural net is very data-intensive. The cost can imply effort as well as actual money. Gathering a task-specific, well-annotated, and high-quality dataset is labor-intensive. Not to mention that some data is difficult to gather due to its scarcity or accessibility. Moreover, there is another problem, namely the number of possible classes. Generally, classifiers do not perform well when there is a large number of classes, especially when there is class imbalance as some classes have only a few examples (Johnson & Khoshgoftaar, 2019).

This is the case with human motion. To achieve our goal to train a network that can generalize to many human poses and tell if two of them are similar we would presumably need hours of video recordings or at least a large number of extracted poses. Once more emphasizing generality: there must be variability in the dataset, so one would need to gather different actors from both sexes and preferably all age groups with a background in different sports/motion activities to

capture a wide range of possible human activities and motions. We would need to create instructions and supervise the data collection process, not to mention paying the people partaking in our recording session. This raw data would need to be preprocessed, quality-checked, and annotated to produce a dataset that can be used for learning. The amount of post-recording time would increase with the data gathered in the recording phase. This could undoubtedly be done, but the resources required would render the data analysis part of the present thesis practically unfeasible.

This appears to be a conflict. As mentioned, training a network that generalizes well on previously unseen data using limited examples is challenging due to the many parameters and the fact that gradient-based optimization methods require many iterative steps to work well (O' Mahony et al., 2019). How can we still achieve good results with a limited dataset?

Researchers in academia and business also realized this problem and developed multiple techniques to mitigate it; thus, the idea of few-shot learning was introduced. Few-shot learning, or more generally K-shot learning, is an umbrella term that refers to a collection of approaches aiming to train machine learning models with fewer examples. One can try to create more (synthetic) data through representation learning and data generation methods, e.g., generative-adversarial networks, variational autoencoders, or simple data augmentation techniques. The second approach is improving the learning process itself. These techniques include meta learning, transfer learning, and metric or similarity learning (Kadam & Vaidya, 2020). In the next section, we will take a closer look at metric learning, expressly distance metric or similarity learning, after which we discuss finding an appropriately parameterized distance function in the few-shot learning framework using Siamese networks.

1.7. Similarity learning and distance metrics

Similarity learning is an area of machine learning / deep learning where the goal is to learn a function that measures distances between data points. In essence, if two objects are similar, they should be "close together." This brings us close to tackling the problem raised in the introduction of the thesis in a mathematical, data-driven way. As we will see, we can express this problem in

terms discussed in the previous chapter; we need to learn a parameterized function that outputs a low distance if two poses are similar and a high distance if the poses are not similar. However, this raises two questions. Firstly: what do we mean by similar? Secondly: what are distance and closeness?

The answer to the first question is not as straightforward as one might think. Is it a problem if, in a specific position, the arms of the person are slightly off? If the torso bent more than expected? The problem can further be complicated. For example, are two squats similar if the legs are in the correct place, but the arms are off? Maybe the position of the arms does not matter for the exercise, and a human fitness instructor would not even focus on it. We could potentially build a system that determines if the pose is more like that of a child or an adult. We can illustrate this problem with examples from other domains as well. Let us imagine that we are working with human speech instead of poses. We can build a system that determines if two speech sequences are similar based on their semantic meaning or something entirely different, for example, the language spoken or even biological differences such as the sex or age of the person talking. The same problem may arise in face recognition or classification. Modern smartphones can be unlocked by feeding in an image of their owner's face. In this case, similarity means being like the owner and not like other people. However, it is entirely possible to define facial similarity differently. For example, we can train a model to determine if the device user is a child or not. Based on this information, certain age-inappropriate content could be excluded. These examples clearly illustrate a fundamental idea: similarity depends on the task at hand and human judgment. It is essential to note that domain-specific knowledge may also be used during training. To illustrate this, let us return to the example of the ballet teacher. He or she most likely has different requirements for two poses - e.g., the demonstrated by him or her, i.e., the ground truth, and one by the student – to be similar. What might look acceptable and aesthetic to a non-specialist can be incorrect based on the instructor's expertise. This knowledge can be leveraged and fed into the system to comply with specific requirements. The technical details of how this can be done will be introduced later.

The second question (distance and closeness) has a more rigorous mathematical definition. For this, we need to introduce the notion of a metric function and metric learning. The following introduction is based on the article by Suárez et al. (2021).

A metric or distance on a non-empty set X is a mapping $d: X \times X \mapsto \mathbb{R}$ such that it satisfies the following properties.

1. Coincidence: $d(x, y) = 0 \Leftrightarrow x = y$, for every $x, y \in X$
2. Symmetry: $d(x, y) = d(y, x)$, for every, $x, y \in X$
3. Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$, for every, $x, y, z \in X$
4. Non-negativity: $d(x, y) \geq 0$, for every $x, y \in X$

As we can see from the above formal definitions, coincidence means that if the distance between x and y is 0 then these two elements must be the same. Symmetry states that the distance x to y must be the same as from y and x. Although distance metrics can be derived for non-numeric attributes will focus on the case with numerical data; that is, we suppose that $X \subset \mathbb{R}^n$.

A simple example of a distance function is the absolute difference or L1 norm

$$\sum_{i=1}^{n} |x_i - y_i| = \|x - y\|_1$$

and the previously mentioned Euclidian distance or the L2 norm, defined as

$$\sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} = \|x - y\|_2$$

We can see how it would make sense to apply this to two poses, e.g., pose A and pose B. If the two poses are perfectly identical, then their coordinates must coincide. Then by definition, two instances of pose A must yield a 0 distance. Now we can alter pose A by a minimal amount by changing the coordinate for the left arm with some small value. Visually, this might look something like moving the arm to the left. This time we must get a number different from 0. We can also introduce pose C, which would look completely different.

In this case, we should see that pose c gets assigned a high distance from both pose A and B while pose A and B have a lower distance; essentially, they are close together. This begs the question of why we would need another metric if there is already the Euclidean distance. As mentioned previously, this metric usually does not handle data with different meanings well. We can imagine that determining the similarity between two human poses for a certain task might be more challenging than just measuring the distances between the joints.

As a next step, we can define metric spaces with the notion of a distance function and a set. A metric space is an ordered pair (X, d) of the set and the metric function defined above.

"Distance Metric Learning (DML) is a machine learning discipline with the purpose of learning distances from a dataset." (Suárez et al., 2021, p. 6). Analogous to the partitioning mentioned before, metric learning can also be considered a supervised or unsupervised learning problem. At present, we focus on the supervised approach. As discussed in previous sections, this supposes a set of ordered tuples:

$$X = \{(x_1, y_1), (x_2, y_2), \dots\ (x_n, y_n)\}$$

where x is considered the input data and y is the ground truth value or label. From this, it is possible to create two disjunct subsets, namely similar and dissimilar defied as

$$S = \{(x_i, x_j) \in X \times X : y_i = y_j\}$$

and

$$D = \{(x_i, x_j) \in X \times X : y_i \neq y_j\}.$$

These are sets containing pairs of same-class and different-class samples. In the set S (similar), each data tuple belongs to the same label. These would be pose-pairs that a human would consider similar. A sitting lotus position from one person must look like a sitting lotus position from another; otherwise, it would be a different pose. The set D (dissimilar) would then contain differently labeled poses. A pair in this set could be a sitting lotus and a downward dog. The above definition is considered the global DML approach. A different approach is the local DML, where the set S is replaced by:

$$S = \{(x_i, x_j) \in X \times X : y_i = y_j \; and \; x_j \in U(x_i)\}$$

where $U(x_i)$ denotes the neighborhood of x_i, which is a set of points that ought to be close to x_j. This must be determined before learning, usually by using some traditional metric, such as the Euclidean distance.

Additionally, we can define a set R with triplets such that:

$$R = \{(x_i, x_j, x_l) \in X \times X \times X : y_i = \; y_j \neq \; y_l\}.$$

meaning we would have one lotus position, then another one perhaps performed by somebody else, and a different pose, e.g., a squat. The importance of this set will be explained further when discussing the so-called triplet loss.

Similarity learning-based approaches have been successfully used in a variety of scenarios, for example, face recognition and identification (Grm et al., 2016), fingerprint matching (Ezeobiejesi & Bhanu, 2018), emotion recognition (Y. Wang et al., 2019), authorship verification (Boenninghoff et al., 2019), time-series clustering (Kim & Moon, 2021) and human gait recognition (D. Liu & Hu, 2021) to name a few.

1.8. Siamese networks

1.8.1. Model architecture

Siamese networks were first introduced by Bromley et al. (1993), and they convey a simple yet powerful idea useful for similarity learning. This introduction to the topic is based on the article from Chicco (2021). Essentially a Siamese network differs from a simple feed-forward neural net in its architecture, as two identical networks are working in tandem, hence the name. Each of these networks reads in an input, processes it, and outputs an n dimensional vector for each of them. These two output vectors are, in turn, fed into some distance metric which can be as simple as the Euclidean or cosine distance. Ideally, this metric should output a low value if the two inputs are similar and a high value if not. As mentioned previously, a neural network can be seen as a concatenation of many parameterized functions. Naturally, this holds for Siamese networks as well. Let us generally denote the network as f. For ease we do not denote θ now. In this context

the output $f(x_i) \in \mathbb{R}^n$ is called the embedding vector or simply embedding. As a distance metric d we can choose the Euclidean distance to measure the similarity between these embeddings. The distance function then can be written as

$$d(x_i, x_j) = \left\| f(x_i) - f(x_j) \right\|_2.$$

Following our previous definition $\mathbb{R}^n$ equipped with the Euclidean distance is a metric space.

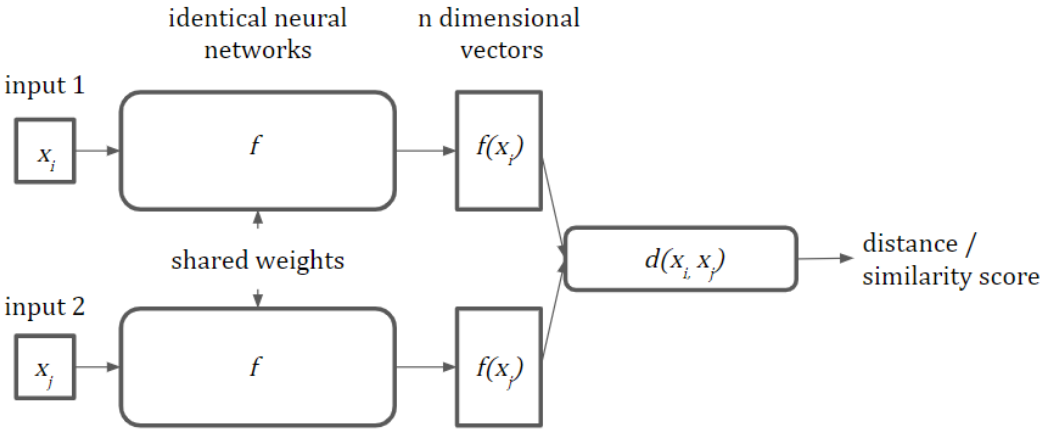

Figure 11. The architecture of a Siamese network. It takes two data points and processes them identically, producing two n-dimensional vectors as output. These vectors are fed into a standard metric function such as the Euclidean distance to infer the similarity.

During the learning process, our goal is to find optimal weights for f such that d is small for similar objects. As we discussed earlier, similarity depends on our definition; consequentially, a low distance in the embedding space can mean semantic similarity. It is important to emphasize that the two networks share and learn the same weights; the difference is only in the input data. These weights are usually updated in unison by some gradient-based algorithm optimizing an objective function during learning.

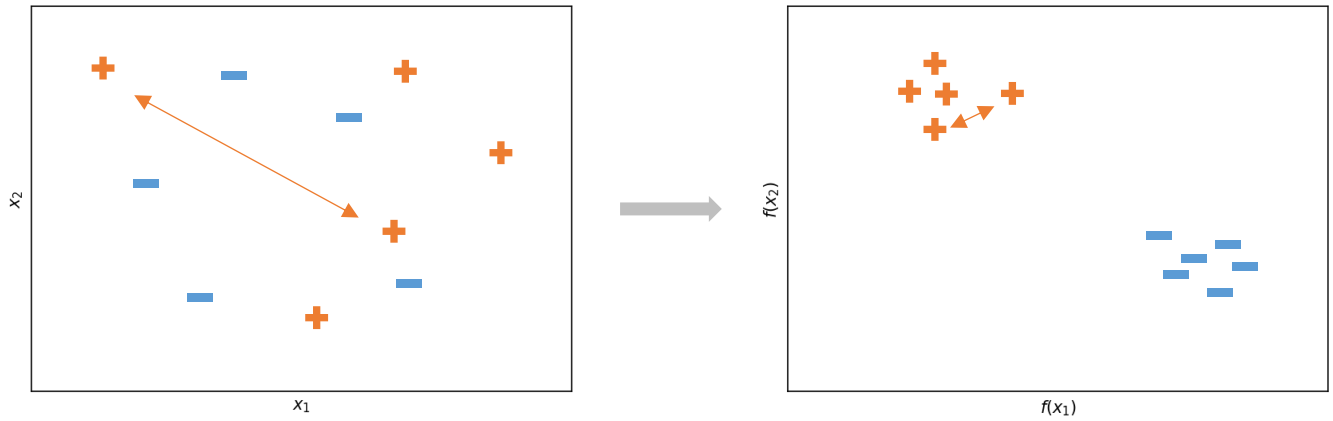

Figure 12. A simplified example illustrating similarity learning in two dimensions, where shape and color indicate similarity. On the left, we can see the data vectors as two-dimensional points in the original feature space, while on the right, in the embedding space. The distances between similar objects are smaller after applying the transformation f.

Though it is not strictly related it is worth mentioning that the input data does not necessarily have to be a vector. It can be a tensor such as images (e.g., Melekhov et al., 2016), or sequential (e.g., Varior et al., 2016). One can use two convolutional neural networks in parallel with a fully connected layer as an output in this setting. The metric then can be applied to these vectors. This is precisely how face recognition and identification can be done. Siamese networks were successfully used to solve many different tasks. In an overview, Chicco (2021) reported 12 different fields where Siamese networks have been applied, ranging from audio and speech signal processing through video analysis to medicine and health.

1.8.2. Contrastive loss

While the architecture of the Siamese network is quite simple, training one that performs well requires some additional concepts. Earlier, we introduced the notion of a loss function where we stated that choosing one that best suits our needs is pivotal to training a network that behaves according to our expectations. By tailoring the loss function, we can alter the training such that it can solve the similarity problem.

A standard loss function that is well suited for these problems is the so-called contrastive loss, which was introduced by Chopra et al. in 2005. It was designed to solve face verification using metric learning. The description below contains a slightly modified version of the original loss so that we are consistent with the earlier notations.

They propose $d(x_i, x_j, \theta) = \|f(x_i, \theta) - f(x_j, \theta)\|_1 \; and \; m > 0$ such that

$$d(x_1, x_2, \theta\,) + m < d(x_1, x_2', \theta)$$

where x_1, x_2 are a similar pair and $x_1, x_2{}'$ are a dissimilar pair. They are, in fact, elements of the S and D introduced in the metric learning section. They argue that the L1 norm between semantically similar data points should be less than the distance between dissimilar data points. The value m can be thought of as a margin which implies that there is also a minimum required distance between similar and dissimilar points.

From this, the loss function for the ith sample can be defined as

$$L(x_1, x_2, y, \theta)^i = y \cdot L_S(d(x_1, x_2, \theta))^i + (1 - y) \cdot L_D\ (d(x_1, x_2', \theta))^i.$$

L_S and L_D are the partial loss functions for the similar and dissimilar pairs respectively, and they should be designed in such a way that minimizing L will increase $d(x_1, x_2, \theta)$ for genuine pairs and decrease it for impostor pairs. This means that L_S should decrease monotonically and L_D should be monotonically increasing.

The value y is the label we get from manual annotation. It is 1 if the input data are considered similar and 0 otherwise. In the case of a genuine pair, the second part of the equation cancels out, and the loss function becomes $L_S\ (d(x_1, x_2, \theta))^i$ so, in essence, the goal becomes minimizing the distance between the embedding vectors. The opposite is true when inputting a mismatching pair because the first term of the equation gets dropped since y is 1.

A common way to use the contrastive loss is in the following form.

$$L(x_1, x_2, y, \theta)^i = y \cdot d(x_1, x_2, \theta)^i + (1 - y) \cdot \max\,(0, m - \ d(x_1, x_2', \theta)^i)$$

From this, we see that L_D is similar to the hinge loss and the role of the max function is optimization. If the distance between dissimilar points is higher than the margin, we do not wish

to spend time and computational capacity pushing the data further apart. As mentioned before, the constant m denotes the margin, and in the strict sense, it is a hyperparameter that was chosen in advance. For example, if we set $m = 1$ we require our dissimilar data points to be at least this far apart by whichever distance metric we use.

The risk or objective function for the whole data is defined as:

$$R_n = \sum_{i=1}^{n} L(x_1, x_2, y, \theta)^i$$

In layman's terms, the Siamese network with the contrastive loss function tries to learn a parametrized function that pulls similar objects together while pushing dissimilar objects apart. It is important to emphasize that we do not feed in the whole dataset during training but rather iterate over data pairs with matching or mismatching labels. This raises the question of how these pairs are created, a topic we will discuss shortly.

1.8.3. Triplet loss

Having understood the contrastive loss and its role in the metric learning scenario, we can introduce the triplet loss (Schroff et al., 2015), which extends the core ideas of the contrastive loss naturally by further utilizing the previously discussed properties of a metric function. Like the contrastive loss, it was proposed to solve face recognition and clustering tasks.

As the name suggests, it always takes a triplet of data points, precisely an anchor x_a^i , a positive input x_p^i and a negative input x_n^i . As one might notice $\forall\left(x_a^i, x_p^i, x_p^i\right) \in R$ as mentioned during the introduction to metric learning. The anchor input is our target; an example could be a specific human pose, such as a downward dog. The positive input is a similar input or an input with the same class, such as another person performing a downward dog, maybe even from a different camera angle. The negative class is, as in the case of the contrastive loss, a datapoint with a different label, for example, a lotus pose.

Let $d(x_i, x_j, \theta) = \|f(x_i, \theta) - f(x_j, \theta)\|_2$ $and\ m > 0$, furthermore $\|f(x_i, \theta)\|_2 = 1$, meaning that the embeddings are located on an n dimensional hypersphere. Then we want to satisfy the following constraint:

$$\|f(x_a^i, \theta) - f(x_p^i, \theta)\|_2^2 + m < \|f(x_a^i, \theta) - f(x_n^i, \theta)\|_2^2$$

$$= d(x_a^i, x_p^i, \theta)^2 + m < d(x_a^i, x_n^i, \theta)^2 .$$

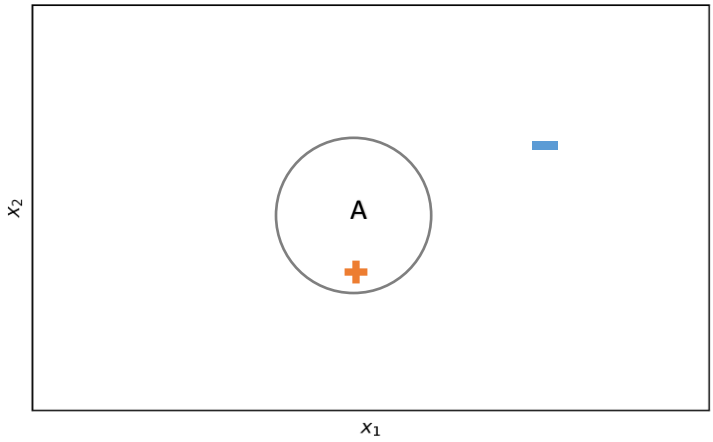

Figure 13. For the triplet loss, the distance metric is the squared L2 norm. The positive data point should lie within a margin m while the negative is outside.

The loss for the ith triplet is defined as

$$L(x_a^i, x_p^i, x_n^i, \theta) = \max\left(d(x_a^i, x_p^i, \theta)^2 - d(x_a^i, x_n^i, \theta)^2 + m, 0\right)$$

This formulation serves computational efficiency just like we have seen with the contrastive loss. A loss value less than zero would imply that the constraint is already satisfied since the positive example is closer to the anchor than the negative, with the latter being outside the margin; thus, there is no reason for further optimization.

The risk function which we would like to minimize for the complete training set is

$$R_n = \sum_{i=1}^{n} L\left(x_a^i, x_p^i, x_n^i, \theta\right)$$

As we can see from the above formulation, "*to learn a discriminative feature embedding, triplet loss maximizes the margin between the intra-class distance and the inter-class distance*" (Yu et al., 2018, p. 2), which is exactly the outcome to be achieved.

When introducing the Siamese network architecture, we specified two parallel networks whose job is to embed the input data. This architecture worked naturally with the contrastive loss where we only had input pairs. How can we use the Siamese network model with the recently introduced triplet loss? The solution comes rather naturally and is quite simple. We extend the network to have three parallel feed-forward networks instead of two. The first branch takes the anchor, the second branch the positive input, and the third branch the negative example.

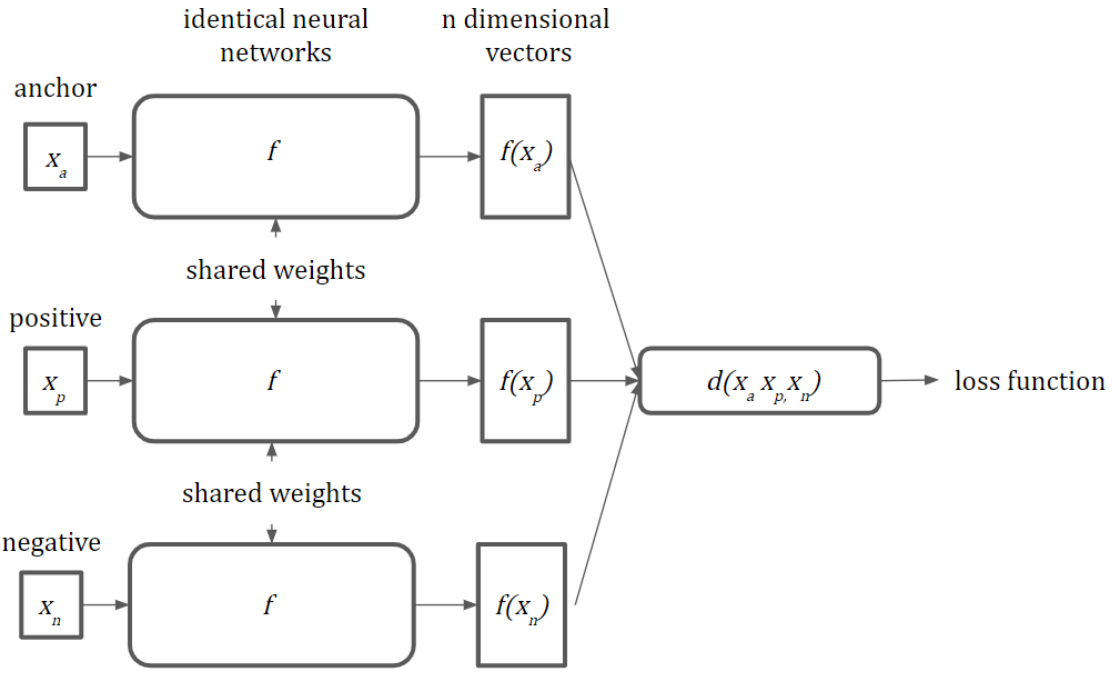

Figure 14. A Siamese network with three branches. An anchor, a positive, and a negative example are fed into the network, creating three embeddings. These embedding vectors are then fed into the loss function. The loss is then minimized using gradient descent.

1.8.4. Semi-hard triplet mining

One of the challenges Siamese networks pose lies in the training setup, namely the triplets introduced in the previous section. Firstly, as the dataset grows, the number of possible triplets grows cubically, making the training on all the triplets unrealistic (Yu et al., 2018). Secondly, while many triplets can be created from the dataset, not all are equally informative; in fact, most may not contribute much to the training at all. Some of these cases are trivial, for example, creating a triplet where the anchor and the positive are identical. Other cases might be too easy, meaning that the negative element is too different, and the network has no problem learning it since the distance to the anchor is already high.

On the other hand, selecting triplets that are too hard would impede the optimization process and result in slow convergence, potentially resulting in poor model performance. This means that selecting triplets at random leads to almost no convergence while selecting too hard triplets leads to getting trapped in a local minimum point (Schroff et al., 2015). For this reason, finding triplets that are hard enough but not too hard is crucial. A rigorous definition of "hardness" and an algorithm that can produce optimal triplets to achieve good training results are needed. A common method for finding these triplets is the so-called semi-hard triplet mining.

When applying semi-hard triplet mining, triplets are categorized into three groups: easy, hard, and semi-hard. We say that a triplet is easy if $d(x_a^i, x_p^i, \theta) + m < d(x_a^i, x_n^i, \theta)$. This intuitively makes sense, considering the previous explanation of the triplet loss. If the anchor and its positive pair are already closer to each other and farther away from the dissimilar negative object by some margin, then no further optimization is needed since the imposed constraint is already satisfied and the loss is 0.

On the other hand, we say that a triplet is hard when $d\left(x_a^i, x_p^i, \theta\right) > d(x_a^i, x_n^i, \theta)$. This case is "hard" because it is precisely the opposite of what we would like to achieve; the negative sample is closer to the anchor than the positive case. Continuing our previous example with human poses, this would imply that a warrior yoga position would be closer to a lotus position than another warrior pose. It is trivial to state that this can hardly be the basis of any system willing to inform us about similar poses.

The third case is the semi-hard triplets. We say that a triplet is semi-hard when $d(x_a^i, x_p^i, \theta) < d(x_a^i, x_n^i, \theta) < d(x_a^i, x_p^i, \theta) + m$. This means that the positive example is closer to the anchor than the negative one, but the said negative is still within the margin. Our objective with the learning is to "move" the negative example outside the margin.

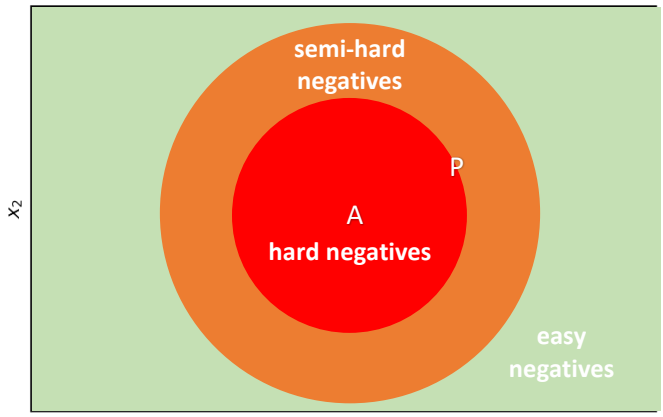

Figure 15. An illustration of the regions of the embedding space with respect to the negative sample.

As we can see, understanding what semi-hard triplets do not answer the question of how to produce or mine them. Numerous papers have been published concerning optimal strategies (e.g., Bhosale et al., 2021; Harwood et al., 2017; Xuan et al., 2020; Yu et al., 2018). These techniques can be broadly categorized into offline and online mining approaches. In the former case, the triplets are identified before training on the whole dataset, while in the latter, triplets are created during the learning phase in each mini-batch (Sikaroudi et al., 2020). The number of triplet mining techniques is large, and a detailed discussion of all of them would again be beyond the scope of this thesis. Deep learning libraries used by Python implement the semi-hard triplet loss and online mining out-of-the-box. Many other loss and mining techniques are also available.

This concludes the introduction to metric learning, Siamese networks, and triplet mining. In the following section, a simple proof of concept of how this knowledge can be applied to tackle the problem of human-pose similarity is presented.

2. Analyses

2.1. Dataset description

Having checked human pose-related benchmark datasets, we have concluded that many of them are unfit for a hands-on demonstration in this thesis as they lack multiview joint coordinates data or the same instances of human actions. Instead, we created a new dataset using typical fitness and yoga pose images. We have collected a total of 140 different images in jpg format for 14 categories from the website, Freepik (*Freepik*, n.d.).

The categories are:

- butterfly sitting
- downward facing dog
- forward lunges
- lotus pose
- natarajasana pose
- plank
- plow pose
- side lunge
- side plank
- squat
- tree pose
- upward facing dog
- warrior I
- warrior II

Figure 16. Examples for each of the 14 categories.

The images were downloaded between 01.05.2022 and 31.05.2022 and named after their respective categories with a suffix from 1 to 10. The keywords used for the search were the name of the action category itself or its plural form. We selected pictures that depicted people of different sex and ethnicity, in different clothing and under varied lighting and background conditions. It was essential to include variations in the pose and the viewing angle. As a criterium, the pose must be recognized as belonging to the respective action category.

The 2-dimensional, 17 key point pose coordinates were extracted using the single pose version of MoveNet Thunder V4 (*TensorFlow Hub*, 2022), an accurate (Jo & Kim, 2022; Washabaugh et al., 2022) open-source network designed for HPE. The model is available under the Apache-2.0 license.

2.2. Data cleaning and processing

The network takes 256x256x3 tensors where the depth dimension corresponds to RGB channels with values between 0 and 255. Consequently, before pose extraction, we processed all images to have a standard 256 by 256-pixel width and height. The images were padded to prevent change in the original image's aspect ratio, thus creating disproportionate human bodies.

Following this step, the network predicted the y and x coordinates along with the confidence scores image-by-image for the nose, left eye, right eye, left ear, right ear, left shoulder, right shoulder, left elbow, right elbow, left wrist, right wrist, left hip, right hip, left knee, right knee, left ankle, and right ankle. The coordinate values are normalized to the image frame between 0 and 1. Similarly, the confidence score is also in this range.

The predicted key points were graphed on the processed images in the next step. We manually checked each image and discarded ones where the estimation was wrong, meaning that one or more predicted key points diverged significantly from their actual position.

The following data were discarded:

- butterfly sitting 6
- downward facing dog 6
- forward lunge 7
- lotus 8
- natarajasana pose 2
- natarajasana pose 4
- natarajasana pose 8
- natarajasana pose 9
- plow pose 8
- side lunge 3
- side plank 9
- squat 2
- upward facing dog 4
- warrior I 4
- warrior II 5

Due to the relatively large number of missing data for the natarajasana pose, we decided to remove all its instances from the dataset. The final analysis involved 116 poses.

We partially followed the method proposed by Xiaohan Nie et al. (2015) to process the poses prior to analyses. This is required for two for two main reasons. First, the actor in the image can appear in different positions performing the pose; however, this should not affect our results. It is indifferent whether, e.g., a lotus pose appears in the middle of the image or the lower left corner. For this reason, we translated each pose such that the origin is at the center of the two hip joints. Secondly, relative size poses a further issue. Since the actor can appear closer or further away in the image, we scaled the poses so that the torso has a unit length. This implies that the length of each body part is expressed in torso units.

The 17 x and y coordinates were concatenated as a final step into one 34-dimensional vector according to the previously described joint order. This serves as the input to the Siamese network.

2.3. Network description

As the size of the input data and the cardinality of the different categories is fairly small compared to what other neural networks are trained on a model with a few layers and nodes can suffice. It is, of course, not only the layers and the nodes that affect the accuracy of the network. The activation function, the optimizer and its step size, the number of epochs, and the margin for the triplet loss are all hyperparameters that cannot be learned with training. Typically, one could do a grid or random search to optimize these hyperparameters (Liashchynskyi & Liashchynskyi, 2019). This would require creating all combinations of the possible hyperparameters and iterating through them. The combinations that produce the best results on some metric would then be selected as the best and used further on. Considering that, presumably, a simple network can perform well, we manually tried a few combinations and chose the one that appeared to work sufficiently well.

The final architecture is a fully connected feed-forward neural network with three parallel branches and 5 layers. The first layer has 32 nodes, while the rest has 20. The first and second layers have a tanh activation function, while the third and fourth have ReLu. The fifth layer has no activation because we do not want to restrict the output range. This last layer is normalized to have unit length following Schroff et al. (2015). As a reminder: this implies that our embeddings will live on the 20-dimensional unit-sphere. We used the triplet loss as the risk function to minimize. For ease, we will simply refer to it as the loss.

The number of epochs is set to 50, and the margin for the triplet loss to 0.5. For the optimizer, we selected the Nadam algorithm (Dozat, 2016) with a learning rate of 0.001. Considering the low amount of data, no batches were created.

To be consistent across runs, the model training was always initiated with a fixed seed of 42 to mitigate the stochastic nature of the training.

Before proceeding, we need to elaborate on training and test sets. As its name suggests, the training set is part of our data for training purposes only. Meanwhile, the test set is data that is only used for evaluation and has never been introduced in the training phase. It is easy to justify the need to create these two sets. During training, it is easy to overfit, or in other words, it is likely to find a function that performs too well on our data and, in turn, poorly on previously unseen data.

Nevertheless, our primary goal is to predict new cases or use our model on previously unseen data. This means that it would be ideal to immediately gather new data in real-world settings and evaluate our model performance on that, but this is not feasible, and one must work with the previously gathered data. This splitting is typically done at random. Due to the scope of the thesis, we must exclude more complex topics regarding training and, test sets, overfitting.

The input data was split into test and training sets in an 80:20 ratio using the same fixed seed and stratified sampling; therefore, the relative frequency within each category is maintained. The triplets were created using the Keras API, providing an online semi-hard triplet mining method that creates optimal triplets for both the test and training data on the fly.

2.4. Results

Thus far, we have gained a deep understanding of statistical learning, neural networks, and, more specifically, Siamese networks for similarity learning. We understand learning means and how a learned model can be used to evaluate input data. However, we have not discussed how the goodness of a neural network can be evaluated, yet it is of fundamental interest. As a reminder, our goal is to give a high similarity score to poses that human annotators deem similar. We could use state-of-the-art network architectures, rely on solid research, and best practices and still produce a poorly performing neural network. We would never be able to confidently deploy our model and use it if we had no information about how it performs. The work herein will present a quantitative and qualitative assessment of the training outcome.

2.4.1. Training performance

Regarding evaluation, it is vital to assess our model's quality during training and afterward. While training, this can be done by tracking the loss at each epoch. As we train for more epochs, we expect the loss to decrease until it eventually flattens out once no further optimization can be achieved. If we see no decrease in the loss, no learning is happening. Customarily the loss on both the training and test sets are plotted on two different lines, which gives further insight into potential problems.

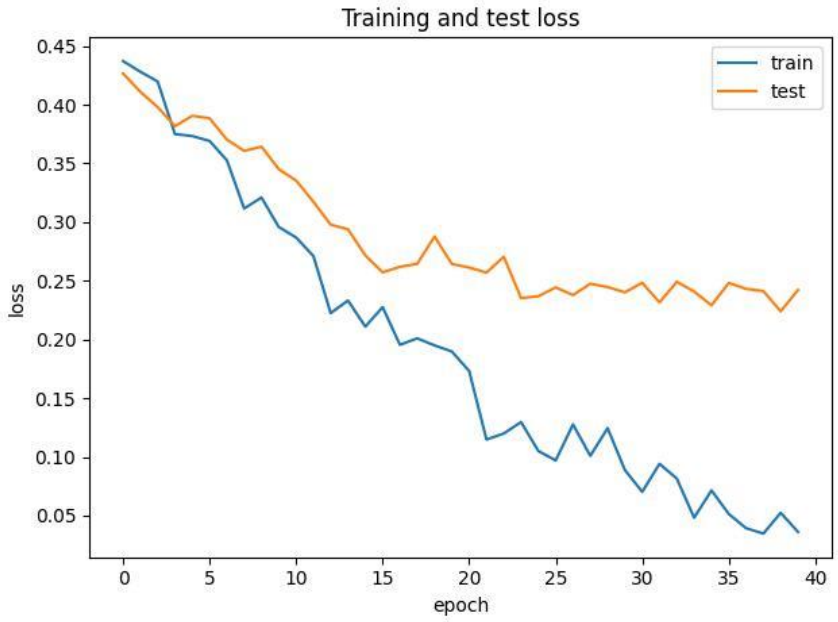

Figure 17. Training and test loss or risk at each epoch.

As demonstrated in Figure 17, the training and test loss decrease with each epoch, meaning that the training is successful. The decrease, however, is not monotonical as the optimization algorithm descends towards the global minimum through local minima. As expected, the training loss is lower than the test loss since the model learns to optimize for "already-seen" data better. We can observe that the training loss keeps decreasing even after 40 epochs, while the test loss plateaus after 25 iterations. We could argue that the test loss started slightly increasing, which implies that further training would only overfit the model that could not perform well on new data.

2.4.2. Model performance

Having trained the network, we need to evaluate its performance. Ideally, we would do this on the test set, but the low number of data points hinders the creation of meaningful plots and metrics; thus, we run analyses on the entire dataset. It is important to emphasize that this gives biased results because the model will always perform better on the data it has been trained on. Nonetheless, it can prove the point that metric learning can be successful.

Several, now industry and academia standard, evaluation metrics have been developed and used to evaluate regression and classification performance. However, evaluating the performance of Siamese networks is somewhat different and is dependent on our goals. This is because, in practice, we would use the embedding vector that is later fed into the distance metric and triplet loss. One way of evaluating the performance of these networks is to use evaluation metrics adapted for classification tasks with an additional change.

It is a good idea to first imagine having a binary classifier as it can serve as the basis for understanding the essential classification metrics. In our case, this could be a network that predicts true if a particular input pose is a lotus pose and false if it is not. In this setup, there can be four outcomes. The network can correctly classify an input as a lotus pose when it is genuinely a lotus pose; this would be a true positive (TP). It could also falsely predict a lotus pose when it is something else, for example, a downward dog. This would be a false positive (FP). Another possibility is that it correctly rejects a different pose. For example, it classifies any other pose other than the target lotus pose as a negative example. This would be a true negative (TN). The last case is the false negative when the network would classify a lotus pose as something else (TN).

The number of true positives, false positives, true negatives and false negatives can be used to calculate many further metrics. We will focus on three: precision, recall, and specificity.

Precision, or positive predictive value, is the percentage of the true positive cases compared to all the cases that were labeled positive:

$$precision = \frac{TP}{TP + FP} = \frac{TP}{total\ number\ of\ predicted\ positives}$$

Recall also called True Positive Rate (TPR), hit rate, or sensitivity measures the number of correctly classified positive or true cases concerning the total number of positive cases.

$$recall = \frac{TP}{TP + FN} = \frac{TP}{total\ number\ of\ positives}$$

Specificity, sometimes called True Negative Rate (TNR), is the number of correctly classified negative cases or false labels concerning the total number of negatives.

$$specificity = \frac{TN}{TN + FP} = \frac{TN}{total\ number\ of\ negatives}$$

These metrics are simple and logical to use; however, there is an issue in our case. The above-described metrics can assess the performance of a binary classifier, and with some extensions, they would be able to assess the quality of a multiclass classifier; however, the Siamese network introduced earlier does not perform any classification. It merely embeds the original data into a lower-dimensional space. Consequently, we need to take additional steps to be able to use precision, recall, and specificity for embeddings. In a way, we need to dichotomize the problem.

We will base our method on Schroff et al. (2015) and modify it to accommodate our analysis.

Let $X_e = f(X)$ denote the embedded data matrix we obtained using the trained model.

Then let us define $D(x_i, x_j) = \|x_i - x_j\|_2^2$ furthermore, let

$$P_S = \{(x_i, x_j) \in X_e \times X_e : y_i = y_j\ and\ i < j\}, \text{ while}$$

$$P_D = \{(x_i, x_j) \in X_e \times X_e : y_i \neq y_j\ and\ i < j\}.$$

In other words, P_S is the set of all pairs with the same label, excluding the case when both items in the pair are identical. P_D on the other hand, is the set of all pairs with a different label. This means that there are $\binom{n}{2}$ items in each set.

Following this, let us define true positives as:

$$TP(d) = \{(i,j) \in P_S : D(x_i, x_j) < d\}$$

This implies that if the squared Euclidean norm between the embedding of the two poses from the same class is smaller than the threshold d we consider it a true positive.

We can similarly define true negatives as:

$$TN(d) = \{(i,j) \in P_D : D(x_i, x_j) > d\}$$

meaning that if the distance for a dissimilar pair is higher than the threshold, it is considered a true negative.

Accordingly, false positives are instances of dissimilar data pairs that are under the threshold:

$$FP(d) = \{(i,j) \in P_D : D(x_i, x_j) < d\}$$

We can see how these definitions make sense considering the distance metric learning paradigm, specifically in the case of Siamese networks. The triplet loss function aims to pull together data points of the same class while pushing away instances of others. If a pair belongs to the same category, for example, two lotus poses, we want the distance between them to be small; if the pair is of different classes, the distance should be high.

The way to calculate precision, recall, and specificity are identical as described above; however, the exact number will depend on our choice of d. A very large d means that our model would label everything as similar. Meanwhile, a small d would result in a high number of rejections. Ultimately, selecting the threshold depends on our needs. If we would rather have a high recall at the expense of false detections, we can opt to make it large. This can be the case for controlling games; it would most likely not matter if the user did an extra swing with a sword but missing the action could mean game over.

There can be cases when the opposite is true. An example could be physiotherapy, where the patient must follow motions very closely. Therefore, choosing a smaller threshold and thus having high precision is better, even if a few detections are missing. Of course, there is no need to dichotomize similarity in such a way since, if needed, a system on top could use the distance itself. If required, the distance could even be normalized by, e.g., using the maximal possible distance on the n-dimensional hypersphere.

Choosing the ideal d is thus not trivial; however, there are some algorithms, e.g., Youden's J statistic, that can select one based on some criteria (Bewick et al., 2004). For our purposes, we used a naïve approach and selected $d = 0.5$ since that was the margin parameter in the triplet loss.

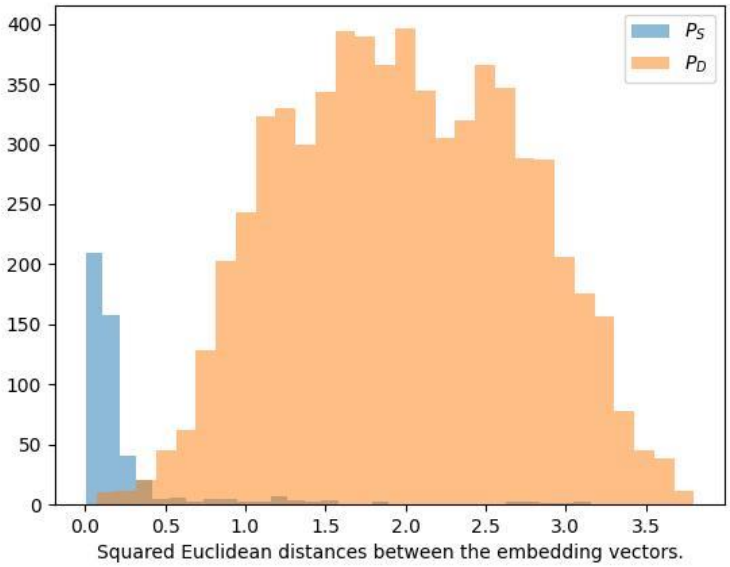

Figure 18. Histograms depicting the distances in P_s and P_D. The distances in P_s are smaller than in P_D and the overlap is minimal, indicating a good separation between the two sets.

Using this threshold, we obtained the following values.

Metric	Result	Metric	Result
TP(0.5)	434	recall	0.89
FP(0.5)	61	precision	0.87
TN(0.5)	6474	specificity	0.99

This table shows that the model achieves good performance on all metrics. With a threshold of 0.5, a similar pair is detected almost 90% of the time.

2.4.3. Qualitative analysis

We used multidimensional scaling (MDS) to visually represent the different poses before and after applying the transformation. *"MDS refers to a set of statistical techniques that are used to reduce the complexity of a data set, permitting visual appreciation of the underlying relational structures contained therein."* (Hout et al., 2013, p. 1). Like other models, this technique also provides a function that maps similar items close. A crucial difference to metric learning is that MDS is an unsupervised learning method that does not need labels and can use multiple distance metrics. Its goal is simply to allow visualizing data points in a lower dimensional space preserving information about the distances. For the current visualization, we chose the Euclidean distance as it was used in the metric learning process. Further details about the method are omitted due to brevity.

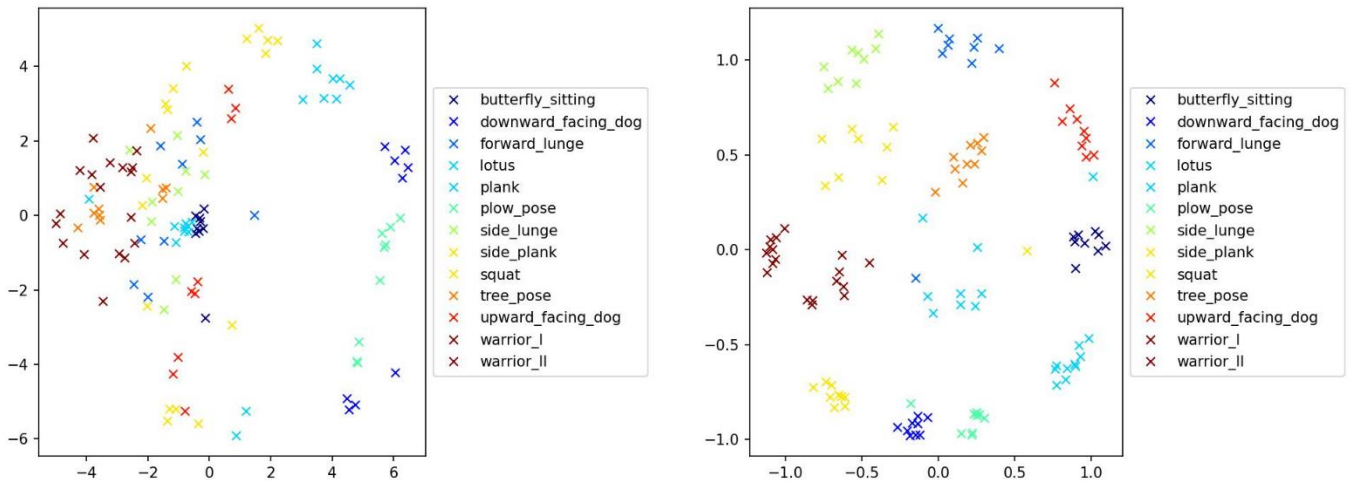

Figure 19. Two-dimensional representation of the poses and the embedded poses using MDS. On the left side, we can see that certain classes are closer to each other while others are spread out. On the right, the classes are much more separated and well-organized. Interestingly we can also observe a circle-like shape that can be the lower dimensional hypersphere.

One can specify how many dimensions to reduce the original data to. Intuitively, the more dimensions we allow the better we can map the data. Presently we choose 2 dimensions as it is the easiest to visualize.

2.4.4. Comparative analysis

We saw how distance metric learning helped cluster the poses in a lower dimensional embedding space; however, we do not know if this method is better than other solutions. In the introduction, we argued that this technique is a better choice when dealing with a high number of classes and a low amount of data in these classes. Due to this, we conducted a comparative analysis by training a multiclass-classifier network using either the poses or the embeddings as input. A drawback of such a classifier is that we could not use the distances between the outputs to measure semantic similarity, even when accessing hidden layers since we never optimized for it. Nevertheless, this way, we could assert two things. Firstly, if a classifier could perform well separating the different pose classes, and secondly whether using embeddings can boost the classifier's performance.

We used a four-layer, fully connected, feed-forward neural network. The first layers have 20 nodes with tanh activation, except for the third one, which uses ReLu. The last layer has 13 nodes,

corresponding to the number of classes. This layer uses the SoftMax activation function. The output for a given input is a 13-dimensional vector that gives predictions for each of the 13 classes. The loss function to optimize for is the categorical cross-entropy.

We trained for 50 epochs using the Nadam optimizer (Dozat, 2016) with a learning rate of 0.01. As in the previous case, no batches were created.

The data was processed and split into test and training cases described in section 2.3. Both for training and selecting the splits, fixed seeds were used to maintain consistency across multiple executions.

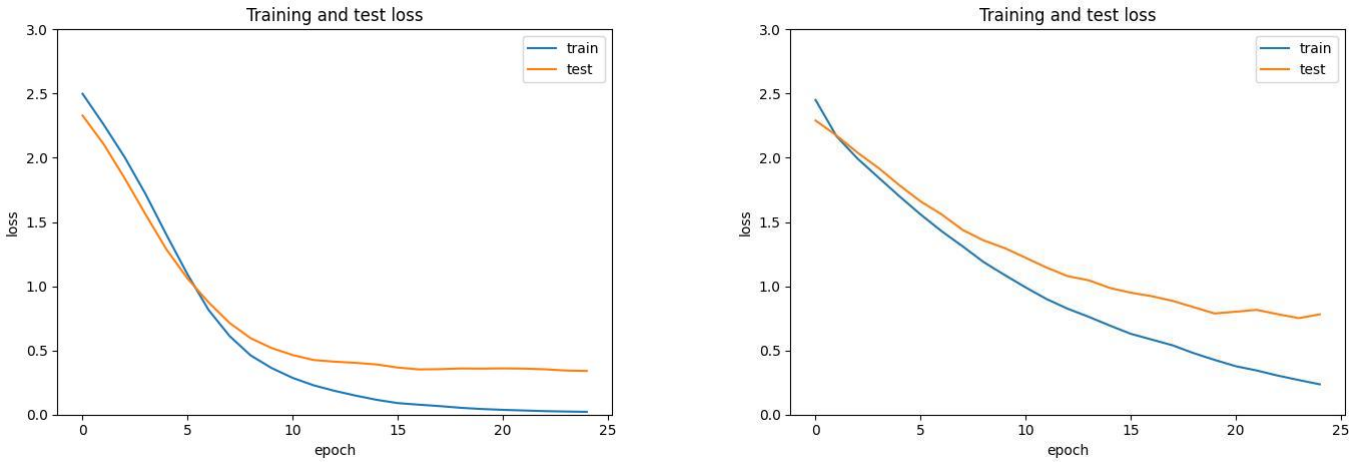

Figure 20. Training and test loss for the multiclass-classifier network. On the left side we use the learned embedding as input while on the right side the pose. The loss decreases faster and reaches a lower value when training with the embeddings.

Both models converged as the test and training losses decreased and eventually plateaued. We can see that training was faster when using the pose embeddings (left image) instead of the original poses (right image). In fact, the training loss seems to be decreasing for the standard pose input even after 25 epochs, with the test loss indicating a slight increase that could eventually result in overfitting.

As the model can output a class label, we can directly use the classification metrics without creating the previous combinations with a distance-based cutoff point. However, we face a different issue; namely, we used metrics to measure the performance of binary classifiers. Since

the current model can predict 13 different classes, we need to alter how we calculate them. Fortunately, the extension comes naturally. To calculate the recall, precision, and specificity for the whole data, we only need to calculate the values for each class separately and then average them. This is defined as macro-averaging. The values for the separate classes are calculated by dichotomizing the data using the one vs. all method. Using this approach, we iterate over the different classes and treat the class under inspection as the target or positive class, while all the data belonging to different classes as the negative class.

Formally, let c be the total number of different classes, and let $metric_i$ be some metric for class i using one vs. all classification. Then we say that $metric_M = \frac{1}{c}\sum_{i=1}^{c} metric_i$ is the macro average of that metric. Specifically for the metrics of interest, these are the following:

$$precision_M = \frac{1}{c}\sum_{i=1}^{c} precision_i$$

$$recall_M = \frac{1}{c}\sum_{i=1}^{c} recall_i$$

$$specificity_M = \frac{1}{c}\sum_{i=1}^{c} specificity_i$$

The evaluation was only done on the test dataset.

Pose		Embedding	
Metric	Result	Metric	Result
$recall_M$	0.77	$recall_M$	0.92
$precision_M$	0.73	$precision_M$	0.92
$specificity_M$	0.76	$specificity_M$	0.92

As we can see from the above table, using the embeddings resulted in significantly better recall (+15%), precision (+19%), and specificity (+16 %) compared to using the raw poses as the input. This means that our metric-learned embeddings can even serve as input features to other neural networks tackling different tasks of human motion understanding.

3. Discussion

3.1. Summary

In the present thesis, we investigated the use of distance metric learning with a Siamese neural network architecture to embed human poses into a lower dimensional space. Our goal was to create a map between the original input and the embedding such that the Euclidean distance is small for similar data points and large for dissimilar data points in the embedding space. This pushing and pulling of data can be seen as minimizing the intra-cluster variance and maximizing inter-cluster variance. The distance between pairs of embeddings corresponds to a certain similarity depending on the selection of the data and human annotation. This analysis focused on different viewpoints and variances in execution style.

The results demonstrate that using metric learning to create a semantic similarity-based human pose embedding is possible. Superior results are seen for using these embeddings when using a cutoff-point-based and multiclass modeling analysis. Moreover, we have demonstrated that using the embeddings results in a better-performing classifier and faster convergence during training. This has multiple implications. Firstly, it is possible to create systems that require some sense of non-trivial similarity measure which could not be solved by a simple metric, such as the Euclidean distance itself. This can be a change in the input pose due to the camera view or more specific requirements such as invariance to sidedness and the position of body parts. The embedded distances could be used for other analytics, e.g., matching and clustering. Secondly, it can serve as input to further models, as we saw with the multiclass classification. These combined methods can unlock easy-to-set-up systems for healthcare professionals, teachers, fitness professionals, application developers, and entertainers.

3.2. Limitations

The results are promising and have interestingimplications for different domains. However, we must admit certain caveats as particular sources of bias make the present findings difficult to extrapolate.

First, the amount of data we collected is low and restricted. We do not have a comprehensive list of all possible human actions with enough variance, and one could argue that such a “population” does not exist. This is problematic since we opened the introduction by stating that supervised learning supposes an i.i.d. sample. A model trained on the wrong data would most likely not generalize well and underperform compared to the somewhat artificial scenarios presented in the analysis. This issue is further exaggerated by removing several data points due to incorrect joint detections. While clean data is pivotal, ensuring that the resulting dataset has no systematic biases is also essential. Removing one action made the learning process easier, but this comes at the cost of decreased generality. The data cleaning process would also require standardization since only one person checked the quality of the pose overlays without a clear set of rules. A better approach would be for two or more annotators to decide to remove images based on objective criteria. The correspondence between their responses should then be analyzed to check the inter-rater reliability (Gisev et al., 2013) and enable better decisions on what data points to remove.

A further issue is the splitting of the training and testing data. As the triplet mining algorithm requires an even number of positive cases to avoid creating duplicates, we had to include more data for training and less for testing. Considering the already low sample size, sometimes only two poses were in the test set. Further restricting the training set would have impeded the learning; thus, we opted in favor of a more significant number of input data there. Consequently, the embedding model was analyzed on the whole dataset. This is discouraged for understandable reasons, and we must acknowledge that the results are likely better than if we only used the test set.

We must also point out that the network architectures used for metric learning and the classification were chosen after some trial and error and not through a rigorous process. Optimally we would have used a grid or random search to find an ideal architecture concerning the number of layers, nodes, activation functions, optimizer algorithms, and the step size used. The models presented herein are also small compared to the ones presented in research articles; however, we must not forget that this is only due to the small amount of data. As stated above, a model we would deploy in actual use would require more diverse training instances and result in a more complex network architecture.

Specific metrics were also adapted to suit the analysis better. Usually, this would not be a problem, but one needs to be aware that the metrics became a function of a cutoff point which would require further tuning. It would also be necessary to analyze the combination of data pairs using the standard pose input. Due to the different scaling of the data, we would need to normalize the pose vectors to have unit length and find the cutoff by using, e.g., Youden's J statistic. With both the embeddings and the poses decision boundary optimized, we could compare the effect of embedding on the distances.

Lastly, the quality of the embeddings depends on the quality of the pose estimator model. While Movenet provides accurate estimates of the body's joints, it can undoubtedly make errors and subsequently affect the quality of similarity learning. Although we removed problematic data, our goal is not to learn invariance to the mistakes of the pose estimator. For this reason, it would be a better choice to separate the pose extraction and further analyses and collect more accurate ground truth data by, e.g., using sensors on the body or having people annotate the exact key points of the body. This data would be a better candidate for our metric learning model.

Regardless of these limitations, the present study serves well as a proof-of-concept and can be a basis for further research.

3.3. Future research

We believe future studies should explore a few options to achieve better results. Addressing the limitations presented above would be the first step. One should obtain a more extensive and varied dataset, preferably with ground truth annotations on the pose. This would potentially enable training a more generalizable and reliable model for end products such as health and fitness applications. A larger test and validation set would result in less biased evaluation metrics. Moreover, the effect of the input data along with the model's variance could be checked using k-fold cross-validation (e.g., Jung, 2018). An example of such a dataset is the Human3.6M (Ionescu et al., 2014), containing 3.6 million 3D human poses from 4 different viewpoints. It is possible to tackle more specific tasks or problematic cases with a specialized dataset, e.g., the Yoga-82 (Verma et al., 2020), hierarchically containing 82 different pose classes. An advantage of this dataset is having some difficult features such as occluded joints, additional objects, and unusual viewpoints; challenges that the model can face in everyday use.

Apart from addressing these limitations, future research could examine the temporal aspects of similarity comparison. While we investigated embedding single poses, it is also possible to embed sequences using recurrent neural networks and triplet loss. This approach was successfully used for speech emotion recognition (Amberkar et al., 2018), intention detection (Ren & Xue, 2020), and user identification (Vamosi et al., 2022). Considering pose sequences instead of single pose instances can carry additional benefits. We could argue that specific poses in time can correspond to an entirely different motion when observed for a more extended period. As a simple illustration, we can imagine somebody doing a backward and a forward arm circle. If we stop at any moment, we cannot be sure what motion we observe. We would only see arms stretched forward, upward, or next to the body. We could not do better than guessing. The motion may be stretching arms, standing upright, or something else; we cannot be sure. Even if someone told us that it is an arm circle, we could not tell if it is performed forward or backward. Access to multiple poses would result in richer information and more accurate detections, unlocking similarity comparisons between motion sequences. A sequence-based system is a natural extension of single-pose-based comparisons as human motion has a time component. Consequently, we

would also be able to detect errors in motion. If a person does something unexpected during the exercise, such as bending the arms during the arm circle, the embeddings would be different.

The temporal aspect could be combined with so-called probabilistic embeddings. A probabilistic embedding represents data as a probability distribution instead of an n-dimensional point. It is usually a Gaussian distribution parameterized by its mean and variance. Such representations can be beneficial when dealing with uncertainty or ambiguity (e.g., Shi & Jain, 2019), which can undoubtedly be the case with, e.g., occluded joints. While the probabilistic approach has been successfully used to create view-independent (Sun et al., 2020) and occlusion robust (T. Liu et al., 2022) embeddings for single pose instances, to our knowledge, no paper has investigated using probabilistic embeddings with pose sequences. Such an approach could provide the benefits of leveraging temporal information while providing a more robust representation of the data. Nonetheless, these solutions are more complex, require a larger dataset, and result in a bigger network that could run slower on less computationally powerful devices, e.g., smartphones.

3.4. Conclusion

In the present thesis, we addressed the problem of human pose similarity comparison. We argued that while specific tasks could be solved with traditional metrics such as the Euclidean distance, others cannot. Viewpoint and execution-style are examples where traditional distance-based techniques would not perform well. As a solution, we introduced supervised statistical learning and artificial neural networks, specifically distance metric learning using Siamese networks and semi-hard triplet mining that aim to embed data points according to a pre-defined similarity definition. We showed that this approach works well on a small selection of custom-gathered fitness and yoga poses. The embeddings provided good recall, precision, and specificity both on combinations of similar and dissimilar pairs and when used as input to a classifier network. A qualitative analysis using MDS showed that the embeddings cluster better than the raw poses. Future research should verify these findings using a larger, openly available dataset and experiment with temporal embeddings of pose sequences potentially using probabilistic embeddings.

References

Abdelouahab, K., Pelcat, M., & Berry, F. (2017). Why TanH is a Hardware Friendly Activation Function for CNNs. *Proceedings of the 11th International Conference on Distributed Smart Cameras*, 199–201. https://doi.org/10.1145/3131885.3131937

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, *4*(11), e00938. https://doi.org/10.1016/j.heliyon.2018.e00938

Adolph, K. E., & Hoch, J. E. (2020). The Importance of Motor Skills for Development. *Building Future Health and Well-Being of Thriving Toddlers and Young Children*, *95*, 136–144. https://doi.org/10.1159/000511511

Ajit, A., Acharya, K., & Samanta, A. (2020). A Review of Convolutional Neural Networks. *2020 International Conference on Emerging Trends in Information Technology and Engineering (Ic-ETITE)*, 1–5. https://doi.org/10.1109/ic-ETITE47903.2020.049

Amberkar, A., Awasarmol, P., Deshmukh, G., & Dave, P. (2018). Speech Recognition using Recurrent Neural Networks. *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*, 1–4. https://doi.org/10.1109/ICCTCT.2018.8551185

Andersen, S. S. L., & Bi, G. (2000). Axon formation: A molecular model for the generation of neuronal polarity. *BioEssays*, *22*(2), 172–179. https://doi.org/10.1002/(SICI)1521-1878(200002)22:2<172::AID-BIES8>3.0.CO;2-Q

Artz, N., Elvers, K. T., Lowe, C. M., Sackley, C., Jepson, P., & Beswick, A. D. (2015). Effectiveness of physiotherapy exercise following total knee replacement: Systematic review and meta-analysis. *BMC Musculoskeletal Disorders*, *16*(1), 15. https://doi.org/10.1186/s12891-015-0469-6

Bewick, V., Cheek, L., & Ball, J. (2004). Statistics review 13: Receiver operating characteristic curves. *Critical Care*, *8*(6), 508. https://doi.org/10.1186/cc3000

Bhosale, S., Chakraborty, R., & Kopparapu, S. K. (2021). *Semi Supervised Learning For Few-shot Audio Classification By Episodic Triplet Mining* (arXiv:2102.08074). arXiv. https://doi.org/10.48550/arXiv.2102.08074

Black, C., Klapaukh, R., Gordon, A., Scott, F., & Holden, N. (2021). Unanticipated demand of Physiotherapist-Deployed Airway Clearance during the COVID-19 Surge 2020 a single centre report. *Physiotherapy*, *113*, 138–140. https://doi.org/10.1016/j.physio.2021.03.010

Boenninghoff, B., Hessler, S., Kolossa, D., & Nickel, R. M. (2019). Explainable Authorship Verification in Social Media via Attention-based Similarity Learning. *2019 IEEE International Conference on Big Data (Big Data)*, 36–45. https://doi.org/10.1109/BigData47090.2019.9005650

Bottou, L. (2010). Large-Scale Machine Learning with Stochastic Gradient Descent. In Y. Lechevallier & G. Saporta (Eds.), *Proceedings of COMPSTAT'2010* (pp. 177–186). Physica-Verlag HD. https://doi.org/10.1007/978-3-7908-2604-3_16

Brian, A., Goodway, J. D., Logan, J. A., & Sutherland, S. (2017). SKIPing with teachers: An early years motor skill intervention. *Physical Education and Sport Pedagogy*, *22*(3), 270–282. https://doi.org/10.1080/17408989.2016.1176133

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1993). Signature Verification using a "Siamese" Time Delay Neural Network. *Advances in Neural Information Processing Systems*, *6*. https://proceedings.neurips.cc/paper/1993/hash/288cc0ff022877bd3df94bc9360b9c5d-Abstract.html

Brusco, N. K., Shields, N., Taylor, N. F., & Paratz, J. (2007). A Saturday physiotherapy service may decrease length of stay in patients undergoing rehabilitation in hospital: A randomised controlled trial. *Australian Journal of Physiotherapy*, *53*(2), 75–81. https://doi.org/10.1016/S0004-9514(07)70039-9

Carpio-Rivera, E., Moncada-Jiménez, J., Salazar-Rojas, W., & Solera-Herrera, A. (2016). Acute Effects of Exercise on Blood Pressure: A Meta-Analytic Investigation. *Arquivos Brasileiros de Cardiologia*, *106*, 422–433. https://doi.org/10.5935/abc.20160064

Chicco, D. (2021). Siamese Neural Networks: An Overview. In H. Cartwright (Ed.), *Artificial Neural Networks* (pp. 73–94). Springer US. https://doi.org/10.1007/978-1-0716-0826-5_3

Chopra, S., Hadsell, R., & LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, *1*, 539–546 vol. 1. https://doi.org/10.1109/CVPR.2005.202

Cook, R., Bird, G., Catmur, C., Press, C., & Heyes, C. (2014). Mirror neurons: From origin to function. *Behavioral and Brain Sciences*, *37*(2), Article 2. https://doi.org/10.1017/S0140525X13000903

Cornelissen, V. A., & Smart, N. A. (2013). Exercise Training for Blood Pressure: A Systematic Review and Meta-analysis. *Journal of the American Heart Association*, *2*(1), e004473. https://doi.org/10.1161/JAHA.112.004473

Corrow, S. L., Dalrymple, K. A., & Barton, J. J. (2016). Prosopagnosia: Current perspectives. *Eye and Brain*, *8*, 165–175. https://doi.org/10.2147/EB.S92838

Cunningham, P., & Delany, S. J. (2021). k-Nearest Neighbour Classifiers—A Tutorial. *ACM Computing Surveys*, *54*(6), 128:1-128:25. https://doi.org/10.1145/3459665

de Greeff, J. W., Bosker, R. J., Oosterlaan, J., Visscher, C., & Hartman, E. (2018). Effects of physical activity on executive functions, attention and academic performance in preadolescent children: A meta-analysis. *Journal of Science and Medicine in Sport*, *21*(5), 501–507. https://doi.org/10.1016/j.jsams.2017.09.595

Della Valle, E., Palermi, S., Aloe, I., Marcantonio, R., Spera, R., Montagnani, S., & Sirico, F. (2020). Effectiveness of Workplace Yoga Interventions to Reduce Perceived Stress in Employees: A Systematic Review and Meta-Analysis. *Journal of Functional Morphology and Kinesiology*, *5*(2), Article 2. https://doi.org/10.3390/jfmk5020033

Demographics of Mobile Device Ownership and Adoption in the United States | Pew Research Center. (2021). https://www.pewresearch.org/internet/fact-sheet/mobile/

Deslauriers, S., Déry, J., Proulx, K., Laliberté, M., Desmeules, F., Feldman, D. E., & Perreault, K. (2021). Effects of waiting for outpatient physiotherapy services in persons with musculoskeletal disorders: A systematic review. *Disability and Rehabilitation*, *43*(5), 611–620. https://doi.org/10.1080/09638288.2019.1639222

Deslauriers, S., Raymond, M.-H., Laliberté, M., Lavoie, A., Desmeules, F., Feldman, D. E., & Perreault, K. (2017). Access to publicly funded outpatient physiotherapy services in Quebec: Waiting lists and management strategies. *Disability and Rehabilitation*, *39*(26), 2648–2656. https://doi.org/10.1080/09638288.2016.1238967

Difini, G. M., Martins, M. G., & Barbosa, J. L. V. (2021). Human Pose Estimation for Training Assistance: A Systematic Literature Review. *Proceedings of the Brazilian Symposium on Multimedia and the Web*, 189–196. https://doi.org/10.1145/3470482.3479633

Dozat, T. (2016). *INCORPORATING NESTEROV MOMENTUM INTO ADAM*. 4.

Estel, K., Scherer, J., Dahl, H., Wolber, E., Forsat, N. D., & Back, D. A. (2022). Potential of digitalization within physiotherapy: A comparative survey. *BMC Health Services Research*, *22*(1), 496. https://doi.org/10.1186/s12913-022-07931-5

Ezeobiejesi, J., & Bhanu, B. (2018). Patch Based Latent Fingerprint Matching Using Deep Learning. *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2017–2021. https://doi.org/10.1109/ICIP.2018.8451567

Farrokhi, A., Farahbakhsh, R., Rezazadeh, J., & Minerva, R. (2021). Application of Internet of Things and artificial intelligence for smart fitness: A survey. *Computer Networks*, *189*, 107859. https://doi.org/10.1016/j.comnet.2021.107859

Firth, J., Cotter, J., Elliott, R., French, P., & Yung, A. R. (2015). A systematic review and meta-analysis of exercise interventions in schizophrenia patients. *Psychological Medicine*, *45*(7), 1343–1361. https://doi.org/10.1017/S0033291714003110

Freepik. (n.d.). Freepik. Retrieved November 6, 2022, from https://www.freepik.com

Gillison, F. B., Skevington, S. M., Sato, A., Standage, M., & Evangelidou, S. (2009). The effects of exercise interventions on quality of life in clinical and healthy populations; a meta-analysis. *Social Science & Medicine*, *68*(9), 1700–1710. https://doi.org/10.1016/j.socscimed.2009.02.028

Gisev, N., Bell, J. S., & Chen, T. F. (2013). Interrater agreement and interrater reliability: Key concepts, approaches, and applications. *Research in Social and Administrative Pharmacy*, *9*(3), 330–338. https://doi.org/10.1016/j.sapharm.2012.04.004

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep Sparse Rectifier Neural Networks. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 315–323. https://proceedings.mlr.press/v15/glorot11a.html

Goldstein, E., Topitzes, J., Brown, R. L., & Barrett, B. (2020). Mediational pathways of meditation and exercise on mental health and perceived stress: A randomized controlled trial. *Journal of Health Psychology*, *25*(12), 1816–1830. https://doi.org/10.1177/1359105318772608

Gowin, M., Cheney, M., Gwin, S., & Franklin Wann, T. (2015). Health and Fitness App Use in College Students: A Qualitative Study. *American Journal of Health Education*, *46*(4), 223–230. https://doi.org/10.1080/19325037.2015.1044140

Grm, K., Dobrisek, S., & Struc, V. (2016). Deep pair-wise similarity learning for face recognition. *2016 4th International Conference on Biometrics and Forensics (IWBF)*, 1–6. https://doi.org/10.1109/IWBF.2016.7449690

Groos, D., Adde, L., Støen, R., Ramampiaro, H., & Ihlen, E. A. F. (2022). Towards human-level performance on automatic pose estimation of infant spontaneous movements. *Computerized Medical Imaging and Graphics*, *95*, 102012. https://doi.org/10.1016/j.compmedimag.2021.102012

Hahn, U., Chater, N., & Richardson, L. B. (2003). Similarity as transformation. *Cognition*, *87*(1), 1–32. https://doi.org/10.1016/S0010-0277(02)00184-1

Hartley, P., Adamson, J., Cunningham, C., Embleton, G., & Romero-Ortuno, R. (2016). Higher Physiotherapy Frequency Is Associated with Shorter Length of Stay and Greater Functional Recovery in Hospitalized Frail Older Adults: A Retrospective Observational Study. *The Journal of Frailty & Aging*, *5*(2), 121–125. https://doi.org/10.14283/jfa.2016.95

Harwood, B., Kumar B G, V., Carneiro, G., Reid, I., & Drummond, T. (2017). *Smart Mining for Deep Metric Learning*. 2821–2829. https://openaccess.thecvf.com/content_iccv_2017/html/Harwood_Smart_Mining_for_ICCV_2017_paper.html

Hausenblas, H. A., & Fallon, E. A. (2006). Exercise and body image: A meta-analysis. *Psychology & Health*, *21*(1), 33–47. https://doi.org/10.1080/14768320500105270

Henderson, K. G., Wallis, J. A., & Snowdon, D. A. (2018). Active physiotherapy interventions following total knee arthroplasty in the hospital and inpatient rehabilitation settings: A systematic review and meta-analysis. *Physiotherapy*, *104*(1), 25–35. https://doi.org/10.1016/j.physio.2017.01.002

Higgins, J. P. (2016). Smartphone Applications for Patients' Health and Fitness. *The American Journal of Medicine*, *129*(1), 11–19. https://doi.org/10.1016/j.amjmed.2015.05.038

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8

Hout, M. C., Papesh, M. H., & Goldinger, S. D. (2013). Multidimensional scaling. *WIREs Cognitive Science*, *4*(1), 93–103. https://doi.org/10.1002/wcs.1203

Iivonen, K. S., Sääkslahti, A. K., Mehtälä, A., Villberg, J. J., Tammelin, T. H., Kulmala, J. S., & Poskiparta, M. (2013). Relationship between Fundamental Motor Skills and Physical Activity in 4-Year-Old Preschool Children. *Perceptual and Motor Skills*, *117*(2), 627–646. https://doi.org/10.2466/10.06.PMS.117x22z7

Ionescu, C., Papava, D., Olaru, V., & Sminchisescu, C. (2014). Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(7), 1325–1339. https://doi.org/10.1109/TPAMI.2013.248

Ishikawa, S., Kim, Y., Kang, M., & Morgan, D. W. (2013). Effects of Weight-Bearing Exercise on Bone Health in Girls: A Meta-Analysis. *Sports Medicine*, *43*(9), 875–892. https://doi.org/10.1007/s40279-013-0060-y

Islam, Md. M., Islam, Md. R., & Islam, Md. S. (2020). An Efficient Human Computer Interaction through Hand Gesture Using Deep Convolutional Neural Network. *SN Computer Science*, *1*(4), 211. https://doi.org/10.1007/s42979-020-00223-x

Jácome, C., Seixas, A., Serrão, C., Teixeira, A., Castro, L., & Duarte, I. (2021). Burnout in Portuguese physiotherapists during COVID-19 pandemic. *Physiotherapy Research International*, *26*(3), e1915. https://doi.org/10.1002/pri.1915

Jo, B., & Kim, S. (2022). Comparative Analysis of OpenPose, PoseNet, and MoveNet Models for Pose Estimation in Mobile Devices. *Traitement Du Signal*, *39*, 119–124. https://doi.org/10.18280/ts.390111

Johnson, J. M., & Khoshgoftaar, T. M. (2019). Survey on deep learning with class imbalance. *Journal of Big Data*, *6*(1), 27. https://doi.org/10.1186/s40537-019-0192-5

Jung, Y. (2018). Multiple predicting K-fold cross-validation for model selection. *Journal of Nonparametric Statistics*, *30*(1), 197–215. https://doi.org/10.1080/10485252.2017.1404598

Kadam, S., & Vaidya, V. (2020). Review and Analysis of Zero, One and Few Shot Learning Approaches. In A. Abraham, A. K. Cherukuri, P. Melin, & N. Gandhi (Eds.), *Intelligent Systems Design and Applications* (pp. 100–112). Springer International Publishing. https://doi.org/10.1007/978-3-030-16657-1_10

Kamruzzaman, J., & Aziz, S. M. (2002). A note on activation function in multilayer feedforward learning. *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, *1*, 519–523 vol.1. https://doi.org/10.1109/IJCNN.2002.1005526

Kanwisher, N., & Yovel, G. (2006). The fusiform face area: A cortical region specialized for the perception of faces. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *361*(1476), 2109–2128. https://doi.org/10.1098/rstb.2006.1934

Kaplan, S., Juvonen, J., & Lensu, L. (2021). A Practical Hybrid Active Learning Approach for Human Pose Estimation. In A. Torsello, L. Rossi, M. Pelillo, B. Biggio, & A. Robles-Kelly (Eds.), *Structural, Syntactic, and Statistical Pattern Recognition* (pp. 334–343). Springer International Publishing. https://doi.org/10.1007/978-3-030-73973-7_32

Kelley, G. A., Kelley, K. S., & Pate, R. R. (2014). Effects of exercise on BMI z-score in overweight and obese children and adolescents: A systematic review with meta-analysis. *BMC Pediatrics*, *14*(1), 225. https://doi.org/10.1186/1471-2431-14-225

Kelley, G. A., Kelley, K. S., & Tran, Z. V. (2000). Exercise and bone mineral density in men: A meta-analysis. *Journal of Applied Physiology*, *88*(5), 1730–1736. https://doi.org/10.1152/jappl.2000.88.5.1730

Kim, J., & Moon, N. (2021). A deep bidirectional similarity learning model using dimensional reduction for multivariate time series clustering. *Multimedia Tools and Applications*, *80*(26), 34269–34281. https://doi.org/10.1007/s11042-020-10476-6

Kingma, D. P., & Ba, J. (2017). *Adam: A Method for Stochastic Optimization* (arXiv:1412.6980). arXiv. https://doi.org/10.48550/arXiv.1412.6980

Kreutzer, R. T., & Sirrenberg, M. (2020). Fields of Application of Artificial Intelligence—Health Care, Education and Human Resource Management. In R. T. Kreutzer & M. Sirrenberg (Eds.), *Understanding Artificial Intelligence: Fundamentals, Use Cases and Methods for a Corporate AI Journey* (pp. 167–193). Springer International Publishing. https://doi.org/10.1007/978-3-030-25271-7_6

Kurbiel, T., & Khaleghian, S. (2017). *Training of Deep Neural Networks based on Distance Measures using RMSProp* (arXiv:1708.01911). arXiv. https://doi.org/10.48550/arXiv.1708.01911

Lachman, R., & Joffe, M. (2021). *Applications of Artificial Intelligence in Media and Entertainment* [Chapter]. Analyzing Future Applications of AI, Sensors, and Robotics in Society; IGI Global. https://doi.org/10.4018/978-1-7998-3499-1.ch012

Lauricella, A. R., Wartella, E., & Rideout, V. J. (2015). Young children's screen time: The complex role of parent and child factors. *Journal of Applied Developmental Psychology*, *36*, 11–17. https://doi.org/10.1016/j.appdev.2014.12.001

Li, C., Liu, S., Yao, L., & Zou, S. (2022). Video-based body geometric aware network for 3D human pose estimation. *Optoelectronics Letters*, *18*(5), 313–320. https://doi.org/10.1007/s11801-022-2015-8

Li, J., & Siegrist, J. (2012). Physical Activity and Risk of Cardiovascular Disease—A Meta-Analysis of Prospective Cohort Studies. *International Journal of Environmental Research and Public Health*, *9*(2), Article 2. https://doi.org/10.3390/ijerph9020391

Liashchynskyi, P., & Liashchynskyi, P. (2019). *Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS* (arXiv:1912.06059). arXiv. https://doi.org/10.48550/arXiv.1912.06059

Liu, D., & Hu, J. (2021). L1-Norm Distance Metric Learning for Gait Recognition. *2021 13th International Conference on Wireless Communications and Signal Processing (WCSP)*, 1–4. https://doi.org/10.1109/WCSP52459.2021.9613587

Liu, M., Wu, L., & Ming, Q. (2015). How Does Physical Activity Intervention Improve Self-Esteem and Self-Concept in Children and Adolescents? Evidence from a Meta-Analysis. *PLOS ONE*, *10*(8), e0134804. https://doi.org/10.1371/journal.pone.0134804

Liu, T., Sun, J. J., Zhao, L., Zhao, J., Yuan, L., Wang, Y., Chen, L.-C., Schroff, F., & Adam, H. (2022). View-Invariant, Occlusion-Robust Probabilistic Embedding for Human Pose. *International Journal of Computer Vision*, *130*(1), 111–135. https://doi.org/10.1007/s11263-021-01529-w

Lodal, K., & Bond, C. (2016). The relationship between motor skills difficulties and self-esteem in children and adolescents: A systematic literature review. *Educational Psychology in Practice*, *32*(4), 410–423. https://doi.org/10.1080/02667363.2016.1206847

Logan, S. W., Robinson, L. E., Wilson, A. E., & Lucas, W. A. (2012). Getting the fundamentals of movement: A meta-analysis of the effectiveness of motor skill interventions in children. *Child: Care, Health and Development*, *38*(3), 305–315. https://doi.org/10.1111/j.1365-2214.2011.01307.x

Lydia, A. A., & Francis, F. S. (2019). *Adagrad—An Optimizer for Stochastic Gradient Descent*. *6*(0972), 3.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. *In ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.

Melekhov, I., Kannala, J., & Rahtu, E. (2016). Siamese network features for image matching. *2016 23rd International Conference on Pattern Recognition (ICPR)*, 378–383. https://doi.org/10.1109/ICPR.2016.7899663

Mironcika, S., de Schipper, A., Brons, A., Toussaint, H., Kröse, B., & Schouten, B. (2018). Smart Toys Design Opportunities for Measuring Children's Fine Motor Skills Development. *Proceedings of the Twelfth International Conference on Tangible, Embedded, and Embodied Interaction*, 349–356. https://doi.org/10.1145/3173225.3173256

Molenberghs, P., Cunnington, R., & Mattingley, J. B. (2009). Is the mirror neuron system involved in imitation? A short review and meta-analysis. *Neuroscience & Biobehavioral Reviews*, *33*(7), 975–980. https://doi.org/10.1016/j.neubiorev.2009.03.010

Molina, M. D., & Myrick, J. G. (2021). The 'how' and 'why' of fitness app use: Investigating user motivations to gain insights into the nexus of technology and fitness. *Sport in Society*, *24*(7), 1233–1248. https://doi.org/10.1080/17430437.2020.1744570

Morres, I. D., Hatzigeorgiadis, A., Stathi, A., Comoutos, N., Arpin-Cribbie, C., Krommidas, C., & Theodorakis, Y. (2019). Aerobic exercise for adult patients with major depressive disorder in mental health services: A systematic review and meta-analysis. *Depression and Anxiety*, *36*(1), 39–53. https://doi.org/10.1002/da.22842

Munea, T. L., Jembre, Y. Z., Weldegebriel, H. T., Chen, L., Huang, C., & Yang, C. (2020). The Progress of Human Pose Estimation: A Survey and Taxonomy of Models Applied in 2D Human Pose Estimation. *IEEE Access*, *8*, 133330–133348. https://doi.org/10.1109/ACCESS.2020.3010248

Nikolić, S. J., & Ilić-Stošović, D. D. (2009). Detection and prevalence of motor skill disorders. *Research in Developmental Disabilities*, *30*(6), 1281–1287. https://doi.org/10.1016/j.ridd.2009.05.003

Nithyakani, P., & Ferni Ukrit, M. (2022). Video Based Human Gait Activity Recognition Using Fusion of Deep Learning Architectures. In G. Manogaran, A. Shanthini, & G. Vadivu (Eds.), *Proceedings of International Conference on Deep Learning, Computing and Intelligence* (pp. 571–579). Springer Nature. https://doi.org/10.1007/978-981-16-5652-1_51

Niu, Z. (2021). A Lightweight Two-stream Fusion Deep Neural Network Based on ResNet Model for Sports Motion Image Recognition. *Sensing and Imaging*, *22*(1), 26. https://doi.org/10.1007/s11220-021-00350-6

Niu, Z., Zhong, G., & Yu, H. (2021). A review on the attention mechanism of deep learning. *Neurocomputing*, *452*, 48–62. https://doi.org/10.1016/j.neucom.2021.03.091

O' Mahony, N., Campbell, S., Carvalho, A., Krpalkova, L., Hernandez, G. V., Harapanahalli, S., Riordan, D., & Walsh, J. (2019). One-Shot Learning for Custom Identification Tasks; A

Review. *Procedia Manufacturing*, *38*, 186–193. https://doi.org/10.1016/j.promfg.2020.01.025

O'Keeffe, M., Hayes, A., McCreesh, K., Purtill, H., & O'Sullivan, K. (2017). Are group-based and individual physiotherapy exercise programmes equally effective for musculoskeletal conditions? A systematic review and meta-analysis. *British Journal of Sports Medicine*, *51*(2), 126–132. https://doi.org/10.1136/bjsports-2015-095410

Okely, A., & Booth, M. (2004). Mastery of fundamental movement skills among children in New South Wales: Prevalence and sociodemographic distribution. *Journal of Science and Medicine in Sport*, *7*(3), 358–372. https://doi.org/10.1016/S1440-2440(04)80031-8

Okely, A. D., Booth, M. L., & Chey, T. (2004). Relationships between Body Composition and Fundamental Movement Skills among Children and Adolescents. *Research Quarterly for Exercise and Sport*, *75*(3), 238–247. https://doi.org/10.1080/02701367.2004.10609157

Palm, W., Webb, E., Hernández-Quevedo, C., Scarpetti, G., Lessof, S., Siciliani, L., & van Ginneken, E. (2021). Gaps in coverage and access in the European Union. *Health Policy*, *125*(3), 341–350. https://doi.org/10.1016/j.healthpol.2020.12.011

Papastergiou, M., Natsis, P., Vernadakis, N., & Antoniou, P. (2021). Introducing tablets and a mobile fitness application into primary school physical education. *Education and Information Technologies*, *26*(1), 799–816. https://doi.org/10.1007/s10639-020-10289-y

Park, J., Chung, S. Y., & Park, J. H. (2022). Real-Time Exercise Feedback through a Convolutional Neural Network: A Machine Learning-Based Motion-Detecting Mobile Exercise Coaching Application. *Yonsei Medical Journal*, *63*(Suppl), S34–S42. https://doi.org/10.3349/ymj.2022.63.S34

Pniak, B., Leszczak, J., Adamczyk, M., Rusek, W., Matłosz, P., & Guzik, A. (2021). Occupational burnout among active physiotherapists working in clinical hospitals during the COVID-19 pandemic in south-eastern Poland. *Work*, *68*(2), 285–295. https://doi.org/10.3233/WOR-203375

Ren, F., & Xue, S. (2020). Intention Detection Based on Siamese Neural Network With Triplet Loss. *IEEE Access*, *8*, 82242–82254. https://doi.org/10.1109/ACCESS.2020.2991484

Richardson, B. R., Truter, P., Blumke, R., & Russell, T. G. (2017). Physiotherapy assessment and diagnosis of musculoskeletal disorders of the knee via telerehabilitation. *Journal of Telemedicine and Telecare*, *23*(1), 88–95. https://doi.org/10.1177/1357633X15627237

Rideout, V. J., Foehr, U. G., & Roberts, D. F. (2010). Generation M[superscript 2]: Media in the Lives of 8- to 18-Year-Olds. In *Henry J. Kaiser Family Foundation*. Henry J. https://eric.ed.gov/?id=ED527859

Roig, M., Nordbrandt, S., Geertsen, S. S., & Nielsen, J. B. (2013). The effects of cardiovascular exercise on human memory: A review with meta-analysis. *Neuroscience & Biobehavioral Reviews*, *37*(8), 1645–1666. https://doi.org/10.1016/j.neubiorev.2013.06.012

Rojas, R. (1996). The Backpropagation Algorithm. In R. Rojas (Ed.), *Neural Networks: A Systematic Introduction* (pp. 149–182). Springer. https://doi.org/10.1007/978-3-642-61068-4_7

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. https://doi.org/10.1037/h0042519

Ruder, S. (2017). *An overview of gradient descent optimization algorithms* (arXiv:1609.04747). arXiv. https://doi.org/10.48550/arXiv.1609.04747

Schoenfeld, B. J., Ogborn, D., & Krieger, J. W. (2017). Dose-response relationship between weekly resistance training volume and increases in muscle mass: A systematic review and meta-analysis. *Journal of Sports Sciences*, *35*(11), 1073–1082. https://doi.org/10.1080/02640414.2016.1210197

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 815–823. https://doi.org/10.1109/CVPR.2015.7298682

Sethi, A., Ting, J., Allen, M., Clark, W., & Weber, D. (2020). Advances in motion and electromyography based wearable technology for upper extremity function rehabilitation: A review. *Journal of Hand Therapy*, *33*(2), 180–187. https://doi.org/10.1016/j.jht.2019.12.021

Shah, T. I., Milosavljevic, S., Trask, C., & Bath, B. (2019). Mapping Physiotherapy Use in Canada in Relation to Physiotherapist Distribution. *Physiotherapy Canada*, *71*(3), 213–219. https://doi.org/10.3138/ptc-2018-0023

Sharma, S., Sharma, S., & Athaiya, A. (2020). ACTIVATION FUNCTIONS IN NEURAL NETWORKS. *International Journal of Engineering Applied Sciences and Technology*, *04*(12), 310–316. https://doi.org/10.33564/IJEAST.2020.v04i12.054

Shi, Y., & Jain, A. K. (2019). *Probabilistic Face Embeddings*. 6902–6911. https://openaccess.thecvf.com/content_ICCV_2019/html/Shi_Probabilistic_Face_Embeddings_ICCV_2019_paper.html

Sikaroudi, M., Ghojogh, B., Safarpoor, A., Karray, F., Crowley, M., & Tizhoosh, H. R. (2020). Offline Versus Online Triplet Mining Based on Extreme Distances of Histopathology Patches. In G. Bebis, Z. Yin, E. Kim, J. Bender, K. Subr, B. C. Kwon, J. Zhao, D. Kalkofen, & G. Baciu (Eds.), *Advances in Visual Computing* (pp. 333–345). Springer International Publishing. https://doi.org/10.1007/978-3-030-64556-4_26

Sinaga, K. P., & Yang, M.-S. (2020). Unsupervised K-Means Clustering Algorithm. *IEEE Access*, *8*, 80716–80727. https://doi.org/10.1109/ACCESS.2020.2988796

Song, L., Yu, G., Yuan, J., & Liu, Z. (2021). Human pose estimation and its application to action recognition: A survey. *Journal of Visual Communication and Image Representation*, *76*, 103055. https://doi.org/10.1016/j.jvcir.2021.103055

Starkweather, A. R. (2007). The Effects of Exercise on Perceived Stress and IL-6 Levels Among Older Adults. *Biological Research For Nursing*, *8*(3), 186–194. https://doi.org/10.1177/1099800406295990

Strasburger, V. C. & Council on Communications and Media. (2011). Children, Adolescents, Obesity, and the Media. *Pediatrics*, *128*(1), 201–208. https://doi.org/10.1542/peds.2011-1066

Suárez, J. L., García, S., & Herrera, F. (2021). A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges. *Neurocomputing*, *425*, 300–322. https://doi.org/10.1016/j.neucom.2020.08.017

Sun, J. J., Zhao, J., Chen, L.-C., Schroff, F., Adam, H., & Liu, T. (2020). View-Invariant Probabilistic Embedding for Human Pose. In A. Vedaldi, H. Bischof, T. Brox, & J.-M. Frahm (Eds.), *Computer Vision – ECCV 2020* (pp. 53–70). Springer International Publishing. https://doi.org/10.1007/978-3-030-58558-7_4

Swift, D. L., Johannsen, N. M., Lavie, C. J., Earnest, C. P., & Church, T. S. (2014). The Role of Exercise and Physical Activity in Weight Loss and Maintenance. *Progress in Cardiovascular Diseases*, *56*(4), 441–447. https://doi.org/10.1016/j.pcad.2013.09.012

Tack, C. (2019). Artificial intelligence and machine learning | applications in musculoskeletal physiotherapy. *Musculoskeletal Science and Practice*, *39*, 164–169. https://doi.org/10.1016/j.msksp.2018.11.012

Tan, H. H., & Lim, K. H. (2019). Vanishing Gradient Mitigation with Deep Learning Neural Network Optimization. *2019 7th International Conference on Smart Computing & Communications (ICSCC)*, 1–4. https://doi.org/10.1109/ICSCC.2019.8843652

TensorFlow Hub. (2022, September 10). https://tfhub.dev/google/movenet/singlepose/thunder/4

Thomson, D., Turner, A., Lauder, S., Gigler, M. E., Berk, L., Singh, A. B., Pasco, J. A., Berk, M., & Sylvia, L. (2015). A brief review of exercise, bipolar disorder, and mechanistic pathways. *Frontiers in Psychology*, *6*. https://www.frontiersin.org/article/10.3389/fpsyg.2015.00147

Vamosi, S., Reutterer, T., & Platzer, M. (2022). A deep recurrent neural network approach to learn sequence similarities for user-identification. *Decision Support Systems*, *155*, 113718. https://doi.org/10.1016/j.dss.2021.113718

van Egmond, M. A., van der Schaaf, M., Vredeveld, T., Vollenbroek-Hutten, M. M. R., van Berge Henegouwen, M. I., Klinkenbijl, J. H. G., & Engelbert, R. H. H. (2018). Effectiveness of physiotherapy with telerehabilitation in surgical patients: A systematic review and meta-analysis. *Physiotherapy*, *104*(3), 277–298. https://doi.org/10.1016/j.physio.2018.04.004

Vapnik, V. (1999). *The Nature of Statistical Learning Theory*. Springer Science & Business Media.

Varior, R. R., Shuai, B., Lu, J., Xu, D., & Wang, G. (2016). A Siamese Long Short-Term Memory Architecture for Human Re-identification. In B. Leibe, J. Matas, N. Sebe, & M. Welling

(Eds.), *Computer Vision – ECCV 2016* (pp. 135–153). Springer International Publishing. https://doi.org/10.1007/978-3-319-46478-7_9

Veldman, S. L. C., Jones, R. A., & Okely, A. D. (2016). Efficacy of gross motor skill interventions in young children: An updated systematic review. *BMJ Open Sport & Exercise Medicine*, *2*(1), e000067. https://doi.org/10.1136/bmjsem-2015-000067

Veldman, S. L. C., Jones, R. A., Santos, R., Sousa-Sá, E., & Okely, A. D. (2018). Gross motor skills in toddlers: Prevalence and socio-demographic differences. *Journal of Science and Medicine in Sport*, *21*(12), 1226–1231. https://doi.org/10.1016/j.jsams.2018.05.001

Verma, M., Kumawat, S., Nakashima, Y., & Raman, S. (2020). *Yoga-82: A New Dataset for Fine-grained Classification of Human Poses* (arXiv:2004.10362; Version 1). arXiv. http://arxiv.org/abs/2004.10362

von Bartheld, C. S., Bahney, J., & Herculano-Houzel, S. (2016). The search for true numbers of neurons and glial cells in the human brain: A review of 150 years of cell counting. *Journal of Comparative Neurology*, *524*(18), 3865–3895. https://doi.org/10.1002/cne.24040

Vukićević, S., Đorđević, M., Glumbić, N., Bogdanović, Z., & Đurić Jovičić, M. (2019). A Demonstration Project for the Utility of Kinect-Based Educational Games to Benefit Motor Skills of Children with ASD. *Perceptual and Motor Skills*, *126*(6), 1117–1144. https://doi.org/10.1177/0031512519867521

Wang, J., Tan, S., Zhen, X., Xu, S., Zheng, F., He, Z., & Shao, L. (2021). Deep 3D human pose estimation: A review. *Computer Vision and Image Understanding*, *210*, 103225. https://doi.org/10.1016/j.cviu.2021.103225

Wang, Y., Qiu, S., Li, J., Ma, X., Liang, Z., Li, H., & He, H. (2019). EEG-Based Emotion Recognition with Similarity Learning Network. *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 1209–1212. https://doi.org/10.1109/EMBC.2019.8857499

Washabaugh, E. P., Shanmugam, T. A., Ranganathan, R., & Krishnan, C. (2022). Comparing the accuracy of open-source pose estimation methods for measuring gait kinematics. *Gait & Posture*, *97*, 188–195. https://doi.org/10.1016/j.gaitpost.2022.08.008

Wilmer, J. B., Germine, L., Chabris, C. F., Chatterjee, G., Williams, M., Loken, E., Nakayama, K., & Duchaine, B. (2010). Human face recognition ability is specific and highly heritable. *Proceedings of the National Academy of Sciences*, *107*(11), 5238–5241. https://doi.org/10.1073/pnas.0913053107

Xiaohan Nie, B., Xiong, C., & Zhu, S.-C. (2015). *Joint Action Recognition and Pose Estimation From Video*. 1293–1301. https://openaccess.thecvf.com/content_cvpr_2015/html/Nie_Joint_Action_Recognition_2015_CVPR_paper.html

Xuan, H., Stylianou, A., & Pless, R. (2020). *Improved Embeddings with Easy Positive Triplet Mining*. 2474–2482. https://openaccess.thecvf.com/content_WACV_2020/html/Xuan_Improved_Embeddings_with_Easy_Positive_Triplet_Mining_WACV_2020_paper.html

Yu, B., Liu, T., Gong, M., Ding, C., & Tao, D. (2018). *Correcting the Triplet Selection Bias for Triplet Loss*. 71–87. https://openaccess.thecvf.com/content_ECCV_2018/html/Baosheng_Yu_Correcting_the_Triplet_ECCV_2018_paper.html

Yuan, R., Zhang, Z., Le, Y., & Chen, E. (2021). Adaptive Recognition of Motion Posture in Sports Video Based on Evolution Equation. *Advances in Mathematical Physics*, *2021*, e2148062. https://doi.org/10.1155/2021/2148062

Yuan, Y., Deng, Y., Zhang, Y., & Qu, A. (2020). Deep learning from a statistical perspective. *Stat*, *9*(1), e294. https://doi.org/10.1002/sta4.294

Zhou, W., Yang, Y., Yu, C., Liu, J., Duan, X., Weng, Z., Chen, D., Liang, Q., Fang, Q., Zhou, J., Ju, H., Luo, Z., Guo, W., Ma, X., Xie, X., Wang, R., & Zhou, L. (2021). Ensembled deep learning model outperforms human experts in diagnosing biliary atresia from sonographic gallbladder images. *Nature Communications*, *12*(1), Article 1. https://doi.org/10.1038/s41467-021-21466-z

Appendices

These appendices contain the most important parts of the code we used for the analysis.

A. Python script to obtain poses from images

```
import json

from tqdm import tqdm

from pose_extraction.image_to_pose import get_coordinates_vector_from_image
from pose_transformations.transform import process_pose
from repository_config.respository_config import RepoConfig


def run():
    path_root_images = RepoConfig().path_data / "images"

    l_labels = []
    l_coordinates = []

    for image in tqdm(path_root_images.rglob("*.jpg")):
        l_labels.append(str(image.stem))

        coordinates =
process_pose(get_coordinates_vector_from_image(path_image=image))
        l_coordinates.append(coordinates.tolist())

    d_data = dict(
        zip(l_labels, l_coordinates)
    )

    path_save = RepoConfig(output_sub_folder="poses").path_save /
"poses.json"

    with open(path_save, "w") as f:
        json.dump(d_data, f, indent=2)


if __name__ == '__main__':
    run()
```

B. Class adapting the MoveNet model for further use

```
import tensorflow_hub as hub

from singleton_decorator import singleton


@singleton
class MoveNet:

    def __init__(self):
        self.model =
hub.load("https://tfhub.dev/google/movenet/singlepose/thunder/4")

        self.movenet = self.model.signatures['serving_default']

    def detect(self, input_image):
        outputs = self.movenet(input_image)
        keypoints_with_scores = outputs['output_0'].numpy()
        return keypoints_with_scores
```

C. Functions to center and torso normalize the poses

```
import numpy as np

from data_types.bodypart import BodyPart
from data_types.bodyparts import BodyParts
from data_types.pose import Pose


def center_pose_to_hip(coordinates: np.ndarray) -> np.ndarray:
    hip_center = get_hip_center(coordinates)
    window = np.lib.stride_tricks.sliding_window_view(coordinates, 2)[::2]
    return np.hstack(window - hip_center).flatten()


def get_hip_center(coordinates) -> np.ndarray:
    limb = BodyPart.from_coordinates(coordinates=coordinates,
body_part=BodyParts.LOWER_TORSO)
    return limb.get_center_point().as_array()


def normalize_pose(coordinates: np.ndarray) -> np.ndarray:
    y_normalized = coordinates[0::2] / coordinates[0::2].max()
    x_normalized = coordinates[1::2] / coordinates[1::2].max()
    return np.vstack((y_normalized, x_normalized)).T.flatten(order="C")


def torso_normalize_pose(coordinates: np.ndarray) -> np.ndarray:
    pose = Pose(coordinates=coordinates)

    lower_torso_center = pose[BodyParts.LOWER_TORSO].get_center_point()
    upper_torso_center = pose[BodyParts.UPPER_TORSO].get_center_point()

    torso = BodyPart(
        key_point_1=lower_torso_center,
        key_point_2=upper_torso_center,
        name="TORSO"
    )
    torso_length = torso.get_length()

    l_pose = []
    for key_point in pose.d_key_points.values():
        normalized_coordinates = (key_point / torso_length).as_array()
        l_pose.append(normalized_coordinates)

    return np.stack(l_pose, axis=0).flatten()


def process_pose(coordinates: np.ndarray) -> np.ndarray:
    normalized_pose = torso_normalize_pose(coordinates=coordinates)
    return center_pose_to_hip(coordinates=normalized_pose)
```

D. Functions to extract and normalize poses

```
from pathlib import Path
from typing import Tuple

import numpy as np
import tensorflow as tf
from tqdm import tqdm

from data_types.data_types import X, y
from movenet.movenet import MoveNet
from pose_extraction.utils import get_label_from_image_name


def preprocess_image(path_image: Path) -> np.ndarray:
    image = tf.io.read_file(str(path_image))
    image = tf.compat.v1.image.decode_jpeg(image)
    image = tf.expand_dims(image, axis=0)
    return tf.cast(tf.image.resize_with_pad(image, 256, 256), dtype=tf.int32)


def get_yx_coordinates_and_score_from_image(path_image: Path) -> np.ndarray:
    image = preprocess_image(path_image)
    return MoveNet().detect(image)


def get_coordinates_vector_from_detection(detection: np.ndarray) ->
np.ndarray:
    return detection[0][0][:, 0:2].flatten(order="C")


def get_coordinates_vector_from_image(path_image: Path) -> np.ndarray:
    detection =
get_yx_coordinates_and_score_from_image(path_image=path_image)
    return get_coordinates_vector_from_detection(detection=detection)


def get_pose_and_labels(path_root_images: Path) -> Tuple[X, y]:
    l_labels = []
    l_coordinates = []

    for image in tqdm(path_root_images.rglob("*.jpg")):
        l_labels.append(get_label_from_image_name(image.name))

        coordinates = get_coordinates_vector_from_image(path_image=image)
        l_coordinates.append(coordinates)

    return np.array(l_coordinates), np.array(l_labels)
```

E. Python script to run metric learning

```
import tensorflow as tf
import tensorflow_addons as tfa

from analysis.data_handling import get_train_test_set, get_x_y_data
from repository_config.respository_config import RepoConfig
from visualization.plot_model_results import plot_training_history, plot_mds


def run():
    tf.random.set_seed(42)

    model = tf.keras.Sequential([
        tf.keras.layers.Dense(32, activation="tanh"),
        tf.keras.layers.Dense(20, activation="tanh"),
        tf.keras.layers.Dense(20, activation="relu"),
        tf.keras.layers.Dense(20, activation="relu"),
        tf.keras.layers.Dense(20, activation=None),
        tf.keras.layers.Lambda(lambda x: tf.math.l2_normalize(x, axis=1))
    ])

    model.compile(
        optimizer=tf.keras.optimizers.Nadam(0.001),
        loss=tfa.losses.TripletSemiHardLoss(margin=0.5),
    )

    X_train, X_test, y_train, y_test = get_train_test_set()

    history = model.fit(
        x=X_train,
        y=y_train,
        epochs=40,
        shuffle=True,
        validation_data=(X_test, y_test),
    )

    path_save = RepoConfig(output_sub_folder="model_results/model").path_save
    model.save(path_save)
    path_save = RepoConfig(output_sub_folder="model_results/plots").path_save
    plot_training_history(history=history, path_save=path_save /
"training_history.jpg")
    plot_mds(x=X_test, y=y_test, path_save=path_save / "mds_original.jpg")
    plot_mds(x=model.get_embedding(X_test), y=y_test, path_save=path_save /
"mds_metric_learned.jpg")
    X, y = get_x_y_data()
    plot_mds(x=X, y=y, path_save=path_save / "all_data_mds_original.jpg")
    plot_mds(x=model.get_embedding(X), y=y, path_save=path_save /
"all_data_mds_metric_learned.jpg")


if __name__ == '__main__':
    run()
```

F. Script to run the comparative analysis

```
import numpy as np
import tensorflow as tf
from sklearn.metrics import classification_report, recall_score

from analysis.data_handling import get_one_hot_encoded_training_test_set
from models.pose_embedder import PoseEmbedder
from repository_config.respository_config import RepoConfig
from visualization.plot_model_results import plot_training_history


def run():
    use_embedding = True

    tf.random.set_seed(42)

    X_train, X_test, y_train, y_test =
get_one_hot_encoded_training_test_set()

    labels = {index: column for index, column in enumerate(y_train.columns)}

    if use_embedding:
        X_train = PoseEmbedder().get_embedding(X_train)
        X_test = PoseEmbedder().get_embedding(X_test)

    model = tf.keras.Sequential([
        tf.keras.layers.Dense(20, activation="tanh"),
        tf.keras.layers.Dense(20, activation="tanh"),
        tf.keras.layers.Dense(20, activation="relu"),
        tf.keras.layers.Dense(13, activation="softmax"),
    ])

    model.compile(
        optimizer=tf.keras.optimizers.Adam(0.01),
        loss="categorical_crossentropy",
        metrics=[tf.keras.metrics.Recall(), tf.keras.metrics.Precision()]
    )

    history = model.fit(
        x=X_train,
        y=y_train,
        epochs=25,
        shuffle=True,
        validation_data=(X_test, y_test),
    )

    model.summary()

    y_true = [labels[prediction] for prediction in
np.argmax(np.array(y_test), axis=1)]
    y_pred = [labels[prediction] for prediction in
np.argmax(model.predict(X_test), axis=1)]

    print(classification_report(y_true=y_true, y_pred=y_pred,
```

```
zero_division=0))

    print(recall_score(y_true, y_pred, pos_label=0, average="macro"))

    path_save =
RepoConfig(output_sub_folder="classifier_model_results/plots").path_save
    plot_training_history(history=history, path_save=path_save /
"training_history.jpg")

    print()


if __name__ == '__main__':
    run()
```

G. Script to analyze the embeddings as a classification problem

```
import itertools

import numpy as np

from analysis.data_handling import get_train_test_set, get_x_y_data
from models.pose_embedder import PoseEmbedder
from repository_config.respository_config import RepoConfig
from visualization.plot_evaluation import plot_distances


def get_distance(
        pose_1: np.ndarray,
        pose_2: np.ndarray,
) -> float:
    return np.linalg.norm(pose_1 - pose_2) ** 2


def is_distance_under_threshold(
        distance: float,
        threshold: float = 0.5,
) -> bool:
    return distance <= threshold


def is_accept(
        pose_1: np.ndarray,
        pose_2: np.ndarray,
):
    distance = get_distance(pose_1=pose_1, pose_2=pose_2)
    return is_distance_under_threshold(distance=distance)


if __name__ == '__main__':
    X_train, X_test, y_train, y_test = get_train_test_set()

    X = get_x_y_data()

    X = PoseEmbedder().get_embedding(X)

    together = np.hstack((X, np.atleast_2d(y_test).T)).tolist()
    combinations = list(itertools.combinations(together, 2))

    same = [
        (
            np.atleast_2d(np.array(comb[0])[0:-1]).astype(np.float32),
            np.atleast_2d(np.array(comb[1])[0:-1]).astype(np.float32)
        )
        for comb in combinations if comb[0][-1] == comb[1][-1]
    ]


    different = [
        (
```

```
            np.atleast_2d(np.array(comb[0])[0:-1]).astype(np.float32),
            np.atleast_2d(np.array(comb[1])[0:-1]).astype(np.float32)
        )
        for comb in combinations if comb[0][-1] != comb[1][-1]
    ]

    true_positives = [is_accept(
            pose_1=comb[0],
            pose_2=comb[1],
        ) for comb in same
    ]

    true_negatives = [ not is_accept(
            pose_1=comb[0],
            pose_2=comb[1],
        ) for comb in different
    ]

    false_positives = [
        is_accept(
            pose_1=comb[0],
            pose_2=comb[1],
        ) for comb in different
    ]

    recall = sum(true_positives) / len(same)
    precision = sum(true_positives) / (sum(false_positives) +
sum(true_positives))
    specificity = sum(true_negatives) / len(different)

    true_distance = [get_distance(
        pose_1=comb[0],
        pose_2=comb[1],
    ) for comb in same
    ]

    false_distance = [get_distance(
        pose_1=comb[0],
        pose_2=comb[1],
    ) for comb in different
    ]

    path_save = RepoConfig(output_sub_folder="plots/evaluation").path_save
    plot_distances(ture_distances=true_distance,
false_distances=false_distance, path_save=path_save / "histogram.jpg")

    y_true = np.hstack((np.full(
        shape=len(true_distance),
        fill_value=1,
        dtype=np.int
    ), np.full(
        shape=len(false_distance),
        fill_value=0,
        dtype=np.int
    )))
```