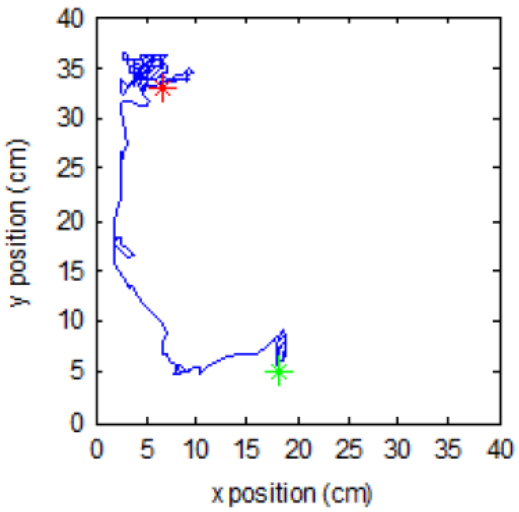
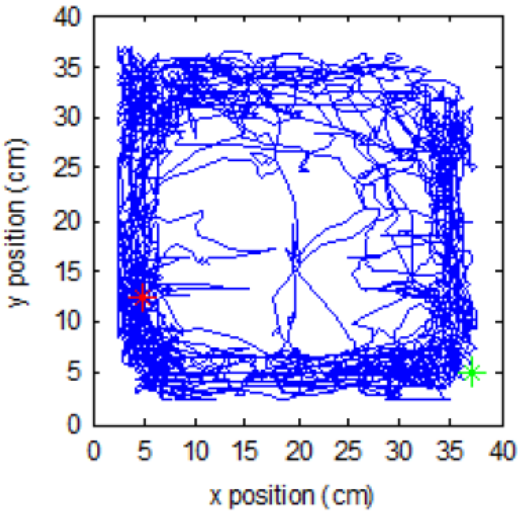




Contrastive Learning for Comparative Behavioural Analysis

Extracting Behavioural Features from Rat Trajectory Data

Master's thesis in Engineering Mathematics and Computational Science

Erik Jansson, Mats Richardson

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Contrastive Learning for Comparative Behavioural Analysis

Extracting Behavioural Features from Rat Trajectory Data

Erik Jansson, Mats Richardson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2023

Contrastive Learning for Comparative Behavioural Analysis
Extracting Behavioural Features from Rat Trajectory Data
Erik Jansson, Mats Richardson

© Erik Jansson, Mats Richardson, 2023.

Supervisor: Lennart Svensson, Department of Electrical Engineering
Advisor: Erik Werner, IRLAB Therapeutics AB
Examiner: Lennart Svensson, Department of Electrical Engineering

Master's Thesis 2023
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Two different example trajectories obtained from individual rats moving freely in a cage.

Typeset in LaTeX
Gothenburg, Sweden 2023

Contrastive Learning for Comparative Behavioural Analysis
Extracting Behavioural Features from Rat Trajectory Data
Erik Jansson, Mats Richardson
Department of Electrical Engineering
Chalmers University of Technology

Abstract

When conducting drug development research, movement pattern analysis is a valuable approach for examining behavioural variations in rats subjected to different substances. In order to reduce the risks of human bias and missed details associated with manually engineered models, machine learning is a viable option to find behavioural features directly from the data. *Contrastive learning* models constitute one such method, which can learn to find relevant behavioural aspects by representing similar substance-induced behaviours similarly. In this thesis, we develop a deep neural network which utilizes contrastive learning to extract behavioral features from rat trajectories induced by different substances. Additionally, various model variations are evaluated and compared against an existing model based on manually engineered features. The results demonstrate similar performance between the proposed and manually engineered models. Surprisingly, the proposed model displays insensitivity to different modifications, and the application of techniques proven successful by other contrastive learning studies does not further enhance performance. These findings suggest a potential underlying issue that may stem from the data, learning approach, or chosen model architecture.

Keywords: Contrastive learning, behavioural analysis, vector embeddings, temporal convolutional networks

Acknowledgements

First and foremost, we would like to thank our corporate supervisor Erik Werner for his generous involvement and sound reasoning. He has provided many valuable thoughts and perspectives that helped to elevate this project. We would also like to thank the other members of the computational team at IRLAB for their encouragement and helpful domain expertise. Lastly, we would like to thank our examiner at Chalmers, Lennart Svensson, for his academic support.

Erik Jansson and Mats Richardson, Gothenburg, 2023-06-21

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

CL	Contrastive Learning
CNN	Convolutional Neural Network
DB	Davies-Bouldin (index)
DR	Dose-Response (score)
FN	False Negative
FP	False Positive
MLP	Multi-Layer Perceptron
NT-Xent	Normalized Temperature Cross Entropy (loss)
PCA	Principal Component Analysis
PLS	Partial Least Squares
PNN	Positive Nearest Neighbour (score)
ResNet	Residual Network
RNN	Recurrent Neural Network
SimCLR	a Simple framework for Contrastive Learning of visual Representations [1]
SimSiam	Simple Siamese (network) [2]
Std	Standard deviation
SupCon	Supervised Contrastive (loss) [3]
TN	True Negative
TP	True Positive
WSS-TSS	quotient of Within Sum of Squares and Total Sum of Squares

Contents

List of Figures

List of Tables

1

Introduction

IRLAB Therapeutics is a biomedical company based in Gothenburg that engages in discovering and developing new pharmaceuticals to treat disorders affecting the central nervous system. The main focus of research lies in treatment drugs for Parkinson's disease which is the second most common primary neurodegenerative disease after Alzheimers, with a current estimate of around 9 million afflicted worldwide. This number is expected to double by 2040, so unless a cure is found, the need for effective means of disease management will only increase [4].

The procedure of developing new medicines is complex and time-consuming, often taking several years of research and clinical trials [5]. Currently, use of animal trials is unavoidable in drug development since several safety studies and screening procedures need to be performed before moving on to humans. The need for pre-human safety studies has a clear ethical reason, and the high cost of human trials necessitates screening for the most promising drug candidates.

However, several challenges arise when using animals instead of humans for drug development. Some desired effects might be harder to observe in animals, and the inability to communicate necessitates other ways of examination. Therefore, behavioural response analysis is used as a way to gauge the effects of research compounds. Observation by domain experts constitutes a qualitative form of behavioural analysis, where fine-grained or obscure behaviours can be detected that would otherwise be hard to quantitatively measure. The problems with qualitative observations are that human bias can influence the readings, and there might not be enough consistency needed for standardized testing. Automated observation such as video tracking eliminates some of the human bias and enables a more efficient way of analyzing behaviour response. One way in which IRLAB performs automated behaviour response analysis is by letting rats move around freely in cages, and recording their position using infrared beams mounted on the sides of the cages. Based on this data, behavioural variations can be quantified using different features that are chosen appropriately.

Choosing these features so that they are informative regarding behaviour is not trivial. There are many different aspects of the recorded data to consider, and the complexity of behaviour makes it difficult to separate relevant patterns from spurious ones. One way to choose such behavioural features is by using domain knowledge and previous research on how to measure behavioral variations in rats. This approach could still suffer from the problem of human bias, based on which

features are selected. There is also a risk of missing aspects that have not yet been considered, or are too complex to be easily measured. Following the increased use of machine learning, several behavioural models have been developed that find informative features autonomously through the use of neural networks [6], [7]. Such a data-driven approach could potentially find features that would otherwise be missed by models using manually produced features, and thereby give more insight into how different compounds affect behaviour.

1.1 Background and Related Work

Currently, IRLAB extracts behavioural information from the recorded movement patterns using a statistical model, consisting of around 300 manually engineered features which are meant to capture different behavioural aspects [8]. For instance, a feature could be the total distance travelled during the experiment, which would be related to the level of activity of the rat. Another feature could be the amount of time spent near the middle of the cage, indicating the level of risk tolerance since open areas are often avoided by rats. The results of one such experiment might not be very informative in itself, but by comparing the features resulting from different substances as well as control experiments, it becomes possible to infer how novel substances affect behaviour in relation to known and well-studied substances. These relative behavioural effects can most easily be visualized by transforming the data from the feature space to two dimensions, using some dimensional reduction algorithm like partial least squares (PLS) regression or principal component analysis (PCA). Plotting the resulting data points in two dimensions would then ideally result in clusters containing substances that yield similar behavioural responses.

The manually selected features for the current model are many and based on previous research, but some informative aspects of the data might still not be picked up. One of the biggest advantages of machine learning is that it can be made to recognize patterns too subtle or complex to be picked up by human perception. Therefore, training a machine learning model to automatically find features to characterize the data might yield a better model than the one currently in use. The manual model will still be useful for our work, however, enabling comparison of the two types of models.

This type of machine learning is called *Feature learning* or *Representation learning*. It revolves around learning informative low-dimensional features from data that, in its raw form, might be highly dimensional, noisy and contain redundant information. Learning such informative features can increase the performance of downstream tasks such as classification, clustering and regression, and it might even be necessary in cases where the raw data is initially too complex, noisy or multi-dimensional [9]. The methods for obtaining features are many and often data- or context-related. Two common examples are PCA, which is used as a dimensionality reduction technique, and Convolutional Neural Networks (CNNs) which are commonly seen in image analysis applications.

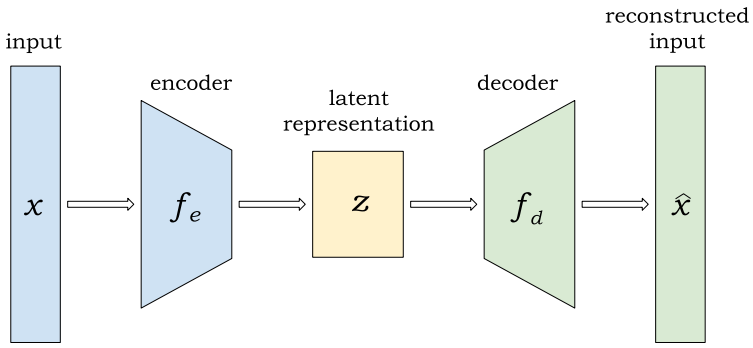


Figure 1.1: Schematic overview of an autoencoder model.

Another dimensionality reduction and feature learning technique is the autoencoder, which has previously been deployed in action segmentation of rats moving freely in cages [7]. The idea of an autoencoder is to project the input x into a lower dimensional space using an encoder network f_e. During training, the latent representation $z = f_e(x)$ is transformed back to the input space using a decoder f_d, in order to reconstruct the input as an approximation from the latent representation, $\hat{x} = f_d(z)$. A well-reconstructed $\hat{x}$ means that little information has been lost during the encoding, so the autoencoder is trained to minimize the reconstruction loss $|x - \hat{x}| = |x - f_d(f_e(x))|$. For a schematic overview of an autoencoder model, see fig. 1.1.

The data used by Luxem, Mocellin, Fuhrmann, *et al.* [7] consists of positional time series of mice moving around in cages, having substantial similarities with the data used in this project. With the goal of action segmentation, each time step is embedded using a recurrent neural network (RNN) encoder, so that these distilled latent embeddings can be used to better classify which action the mouse is performing at any given time. This goal presents a crucial difference compared to our project, since we consider entire time series as single data points to be compared with other time series, instead of individual time steps. As a consequence, the autoencoder would have to embed and reconstruct the entire positional trajectory, becoming nearly impossible given the complexity of animal behaviour and a long enough trajectory. One way around this challenge could be to modify the reconstruction loss with other similarity measures between the original and reconstructed trajectory. This leads us back to square one however, since the goal is for the model itself to learn which aspects of the trajectories are relevant for behavioural change, and which can be disregarded.

One technique used for feature learning where there is no reliance on reconstruction is *contrastive learning*, which instead focuses on the representation of similar and dissimilar samples. This approach has proven to be successful in representation learning, creating latent vector embeddings of image data such as with SimCLR [1]. While a supervised model would use given labels to learn which samples should be represented similarly and which ones should not, a contrastive learning model is generally trained to represent differently transformed versions of the same sample as similarly as possible. In order to avoid collapsing solutions, meaning that the

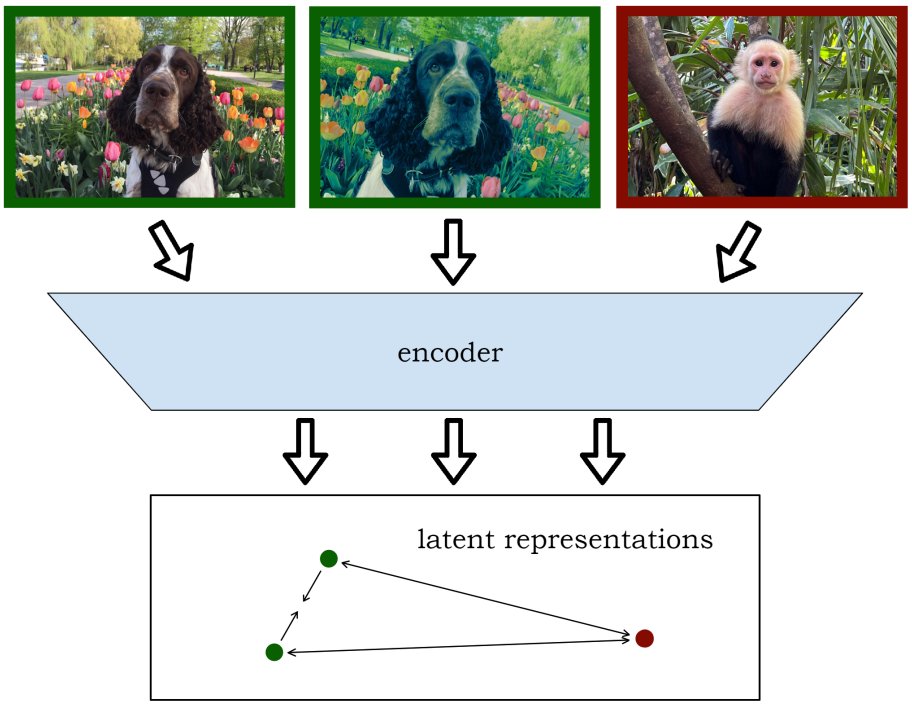


Figure 1.2: Illustration of images being embedded by a contrastive learning model. The two augmented versions of the same image (green) should be represented similarly to each other, but differently from the third image (red).

model encodes all images to the same representation, it is also trained to represent different images as dissimilarly as possible. When the data consists of images, the transformations used are often different types of augmentation techniques such as rotations, crops, or colour adjustments [1], [10]. By indicating that such different versions of the same image should be given similar representations, the model learns to find general salient features, invariant to the augmentations and indicative of relevant information. The concept of contrastive learning is illustrated in fig. 1.2.

While the concept of contrastive learning is general and applicable to different data types, the SimCLR paper [1] is based on image data. A ResNet architecture [11] is used to encode the images into feature vectors, using multiple convolutional layers with residual connections to facilitate training. While the feature learning aspect could make the technique appropriate for our task, the difference in data types would require some changes. Instead of a ResNet CNN used for images, the encoder could be composed of some architecture more suitable for time series data. One such kind of architecture often used to process time series is the RNN [12], which allows output from some nodes to affect subsequent input to the same nodes. The output of the model incorporates multiple time steps of the time series input, where this number of time steps, or the *temporal window*, depends on the recurrent connections. One major drawback with RNNs is that as the attention span increases by making the temporal window larger, the model becomes more difficult to train [13]. Any long-range temporal dependencies would thus be difficult to capture using an RNN.

As a way to remedy this short attention span problem, research has been conducted into using convolutional networks for time series, by convolving the input in the temporal domain and thereby allowing the capture of sequential information and long-term dependencies in data. This kind of network is called Temporal Convolutional Network (TCN) [13], and has proven to perform on par with RNNs on various sequence modelling tasks [12], [14]. As a result of typical areas of application, two common properties of the TCN is causality, which means that the model cannot consider future information, and an output size equal to the input size when model output is desired for each specific time step. These are two properties that are not relevant to our task at hand, since each time series trajectory is supposed to be represented by a single feature vector of appropriate length, without constraints on any chronological structure.

1.2 Objective and Method

The overarching goal of this project is to create a model that can successfully find behavioural differences between rats that have been given different substances. This model could then help to identify the effect of novel research drugs by comparing the behavioural response with those of already known and studied substances. While IRLAB has previously developed such a model using manually chosen features for distinguishing behavioural differences, it would be of interest to examine how an algorithm based on machine learning performs on the same task. A closely related goal will therefore be to examine the performance of this machine learning model in relation to the manual model based on different evaluation metrics. The combination of the two models will also be examined in order to deduce how well they complement each other.

The method used to create such a model is in general that of designing a machine learning algorithm for a new objective and dataset. Suitable evaluation metrics are developed to measure the performance of the model, an encoder network is designed to transform the data into informative vector representations, and a loss function is chosen to facilitate learning. When the model has taken shape, it is analyzed in relation to the manual model to compare their performance and assess how well they complement each other.

1.3 Limitations

This project is faced with the challenge of a vast array of potential architectures and methods to explore. However, we must focus on a limited subset of these techniques, due to the constraints of our project timeline and expertise. The main delimitation of scope lies in only considering *contrastive learning* techniques, training the model to group similar data samples and separate dissimilar ones in the latent space. There are different architectural alternatives for encoding the time series data, and we have chosen to investigate only convolutional architectures. While recurrent networks are often used when processing time series [12], the length of our time

series data would likely make it challenging to successfully train a recurrent network with a large enough context window [13]. The transformer structure has also seen a lot of success with sequential data, but seems to suffer from temporal information loss when applied to time series [15]. Because of these apparent problems and the limited time scope of this project, we have not chosen to investigate such encoder alternatives further.

An additional limitation comes as a result of the available data. We have not found any other data on rats' positional trajectories together with information about administered substances. Therefore, any studies on how well the model generalizes to other datasets have been omitted from the project.

Lastly, the number of model modifications and variations examined makes it unfeasible to test all combinations of these modifications exhaustively. Some combinations of modifications have been tested when there has been reason to believe that the effects of these modifications might be correlated. For efficiency purposes, however, the standard approach is to test modifications separately.

1.4 Contributions and Outline

The main contribution of this thesis work is a thorough examination of how temporal-convolutional contrastive learning models perform on the given task, which is a comparative behavioural analysis of rat trajectories. As an effect of the unique dataset used, current machine learning techniques are incorporated and adapted in a novel way in order to work with the data. New evaluation metrics are also proposed to facilitate a qualified comparison of model performance on the dataset.

As an outline of the rest of this report, chapter 2 provides an explanation of the theoretical foundation on which this thesis is built. The focus lies on contrastive learning and its various components, such as encoder networks, loss functions, regularization techniques, and evaluation methods. In chapter 3, there is first a detailed description of the dataset used in this study. A base model is also presented, including how the techniques described in chapter 2 are utilized and adapted to the current context. The evaluation procedure is also presented here. Chapter 4 begins with the evaluation and comparison of this base model against the manually constructed model developed by IRLAB. Then follow several sections dedicated to describing and evaluating specific modifications made to the base model. This analysis aims to examine how such modifications impact model performance, in order to get a more thorough understanding of the suitability of contrastive learning techniques for the given task. Lastly, chapter 5 presents a general and concluding discussion. This chapter summarizes the previously presented findings, highlights the key results, and offers insights into the implications of the study. Moreover, it suggests potential directions for future research and areas that warrant further exploration regarding machine learning in comparative behavioural analysis.

1.5 Ethical and Sustainability Aspects

The main goal of the research at IRLAB is to discover and develop medicines for disorders affecting the central nervous system. Since achieving this goal would save or improve the lives of many around the world, the cause certainly ought to be considered ethical and aspirational. This is also in agreement with the Sustainable Development Goals set by the United Nations [16]. For instance, the third goal regards the health and well-being of the world's population, with an aim to reduce the mortality caused by non-communicable diseases.

However, in order for a new pharmaceutical to be approved for human usage, it is currently necessary that the compound is first tested on animals (in vivo) to ensure safety. Animal research has different purposes depending on what stage of the drug development process it is applied in [17], but for our purposes it can be categorized as follows:

- **Characterisation of promising candidate medicines**
  After having chosen what disorder to treat and which protein or gene to target, a large amount of different molecules are analysed by simulation and tests on isolated cells or tissue (in vitro) to find the ones that react with the target. These compounds that react in a desirable way are then analyzed further, which includes in vivo testing in order to better characterize the effects using different methods such as neurochemical analysis, gene expression, or subject behavioral analysis.
- **Ensuring the safety of selected candidates**
  The most promising medicine candidates then have to pass toxicological testing, in which each candidate drug is administered in different ways and dosages to a larger number of animal subjects, which produces data about safety and efficacy. The candidates that are deemed safe and effective enough then can move on to clinical trials on humans.

IRLAB aims to minimize the use of animal research, in part by extracting as much useful data as possible from the tests used to characterize promising candidate medicines. This enables a more thorough screening so that fewer inadequate candidates make it to toxicological testing.

The aim of this project is to find methods for characterizing the behavioral effects of different compounds, which perform better than the model currently used. This could improve the screening further, potentially increasing the probability of finding promising candidates while at the same time lowering the risk of unnecessary toxicology tests. One drawback of the project however is that it is still based on in vivo testing, creating an even larger investment in animal research. The United Nations' Sustainable Development Goals does not list any explicit goals regarding animal research, but Keeling, Tunón, Olmos Antillón, *et al.* [18] examined to what extent the goals are compatible with animal welfare, and found that several of the goals imply ethical and responsible treatment of laboratory animals. There will likely be better alternatives to animal research for pharmaceutics development in the future. Given the current circumstances, however, the ethicality of this project becomes

intertwined with the question of whether or not the possibility of finding better treatments for diseases like Parkinson's can justify the use of animal research.

2

Theory

This chapter covers the theoretical basis on which the thesis is built, with focus on contrastive learning and its components. These components include encoder architectures, loss functions, and ways to evaluate performance. Some additional theoretical concepts are also presented, which act as a basis for various model variations that will be presented and analysed in chapter 4.

2.1 Introduction to Contrastive Learning

Contrastive learning is a technique included in the broader concept of *Representation learning*, with the approach to learning representations by comparing data samples against each other. The goal is to map similar samples close together in the embedding space, and dissimilar samples far apart. The similarity-relation between samples is a key part of contrastive learning. Often, the data is considered in positive and negative pairs of samples. Positive pairs should have a similar representation in the embedding space, and negative pairs a dissimilar representation. With no label information, positive pairs can be generated by pairing one sample from the dataset with an augmented version of the same sample. How augmentations can be created is further discussed in section 2.4.2. All the other pairs in the batch containing that sample are considered dissimilar (negatives). For a batch size of N, this provides $\binom{N}{2}$ total pairs: N positives and the rest being negatives. This type of learning is called *self-supervised* since the model is itself creating a set of pairwise positive and negative labels. because the model is supervising itself on which augmented samples are from the same original samples and those that are not. In supervised setups where the labels are known, positive pairs can be formed from samples belonging to the same class as well as from augmentation.

The idea behind contrastive learning originates from human learning patterns [19]. Humans can learn to recognize objects without noting every small detail, but rather general patterns on a larger scale. In the case of images, it is very easy for a human to know that an image and its flipped counterpart are the same objects. Contrastive learning is a way of trying to make a model reason in the same way. This is normally done by choosing appropriate augmentations so that the *style* of two augmented versions differ, but not the *content*, i.e. what the image represents [20]. A suitable loss function is then formulated, with which the model can be trained to create representations invariant to the augmentations. If the augmentations cover the

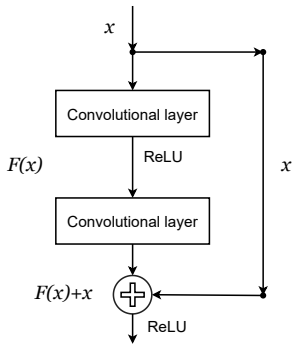


Figure 2.1: Schematic example of a ResNet block [11], characterized by the skip connection performing an identity mapping.

entire style space, meaning that they change all aspects not related to the semantic information, the model should represent images solely based on content.

2.2 Model Architectures

The choice of model architecture determines how the input data x is encoded to the latent representations $z \in \mathbb{R}^n$ in the embedding space, which in general has significantly fewer dimensions. It plays a huge part in how and what features of the data are learned. We will denote the encoder function f_e so that $f_e : I \to \mathbb{R}^N$ where I is a general input domain and $f_e(x) = z$. In this section, we provide an overview of Residual networks (ResNets) as well as Temporal Convolutional Networks (TCNs), as they have been an inspiration for the proposed architecture in this project.

2.2.1 Residual Networks

Residual networks (ResNets) are neural networks that are characterised by residual connections, also known as skip or shortcut connections [11]. The residual connection is a shortcut between the input and the output of a layer in the network. For an input x_I, output x_O, a layer f, and an activation function g, this means that

$$x_O = g(x_I + f(x_I)). \tag{2.1}$$

The motivation for this structure comes from an observed decrease in performance on test *and* training data when adding layers to deep feed-forward networks [11]. This degradation comes from the added complexity of optimizing the deeper network's parameters, which the skip connections are hypothesized to alleviate by improving the statistical properties of the gradients [21].

The general structure of a ResNet is a deep neural network consisting of consecutive blocks, where each block contains one or more layers and a residual connection, as illustrated in fig. 2.1. The layers in the figure are convolutions, but it can differ depending on the specific use case. Moreover, the depth of a ResNet and the types of final layers can also vary. In one of the most famous ResNet papers, He, Zhang, Ren,

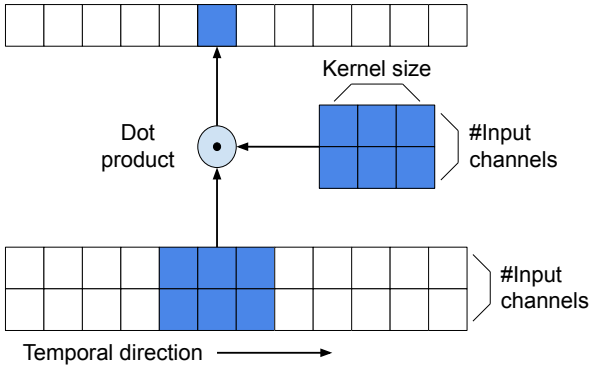


Figure 2.2: Schematic illustration of temporal convolutions. The kernel is slid across the temporal dimension. Additional kernels can be used to increase the output dimension.

et al. [11] presents CNN implementations with depths ranging from 18 layers to 200. These networks were created to learn informative features in images and perform classification, so the final convolutional layer is followed by a fully connected layer with an output equal to the number of classes in the data set. As the image input is processed through the layers, the spatial dimensions are reduced using convolutions. Simultaneously, the number of feature channels is increased.

2.2.2 Temporal Convolutional Networks

The ResNet structure presented by He, Zhang, Ren, *et al.* [11] is designed to process image data and learn informative features by convolving the spatial dimensions. Since images have no temporal information, some alternative approach would be needed for the networks to process time series data and find temporal features. Temporal convolutional networks (TCNs) [12], [13] offer one such possible approach, since they are designed to process sequential data.

TCNs build upon the concept of convolutions, which are widely recognized in image analysis applications where they are typically applied in two dimensions. For TCNs however, convolutions are performed in one dimension, specifically the time domain as visualized in fig. 2.2. The kernels used in the convolutions of a TCN have a shape of (#input channels, kernel size), where the number of input channels is determined by the input data. The kernel size can be adjusted to control the attention span of the kernel, representing the size of the temporal window considered for each output value. The output is computed by taking the dot product between the kernel and the current input, and the kernel is then slid across the temporal domain based on stride. Multiple output channels can be obtained by simply employing additional kernels and performing the computations in parallel. To maintain simplicity, certain aspects of convolutions including stride, padding, and activation function have been omitted in this explanation. For a more comprehensive understanding, we refer to the textbook by Goodfellow, Bengio, and Courville [22].

Through the application of convolutions, TCNs are capable of capturing local features within the data, providing a way to extract valuable information at a fine-

grained level. To encompass global features, TCN architectures typically consist of multiple convolutional layers. This architecture allows for a wider temporal window, enabling the network to learn long-term temporal relationships and potentially complex features from the data. The effectiveness of TCNs in various tasks is supported by numerous promising results as summarized by Bai, Kolter, and Koltun [12]. In their study, TCNs are compared to another commonly used modelling structure for sequential data, namely RNNs. The authors assert that TCNs offer several advantages over RNNs, most notably demonstrating a longer memory capacity compared to typical RNN structures. This characteristic makes TCNs particularly advantageous in scenarios where global features of long time series are essential.

2.3 Contrastive Loss Functions

In this section we will present several contrastive loss functions, starting with simple formulations and moving on to more complex ones. All loss functions depend on similarity or distance metrics, which are presented in the section below.

2.3.1 Similarity and Distance Measures

A similarity measure is used to determine the similarity between the encoded vector representations of different data samples. The similarity should be high if the vectors ought to be considered similar, and low if they are dissimilar. During training in a contrastive learning setting, the similarity between positives should be maximized and minimized between negatives. Similarity measures relating two representations p and q will be denoted $\text{sim}(p, q)$.

As an alternative to similarity, a distance measure d, can be used instead. In this setting, the distance should be minimized between positives and maximized between negatives. Which similarity or distance measure to use depends on how the loss function is formulated, and which aspects of the vector representations that are considered relevant. In the rest of this section, consider two vectors $p, q \in \mathbb{R}^n$.

One example of a distance metric is the Euclidean distance defined as

$$d_e(p, q) = \sqrt{(p_1 - q_1)^2 + ... + (p_n - q_n)^2} \equiv ||p - q||_2. \tag{2.2}$$

The Euclidean distance is often used in different contrastive loss functions. Although very intuitive and predictive in two and three dimensions, it can behave differently in higher dimensions, which is known as the *curse of dimensionality*. As explained by Beyer, Goldstein, Ramakrishnan, *et al.* [23], a consequence of high dimensions is that the Euclidean distances for the nearest and farthest points for a given sample converge with an increasing number of dimensions. In other words, the distances between a sample and its nearest and furthest points are almost equal in high dimension. As this applies to many different data distributions and distance measures, it means that the concept of distance comparison loses meaning in very high dimensions.

Another distance metric is the Manhattan distance, which is defined as

$$d_m(p,q) = |p_1 - q_1| + ... + |p_n - q_n| \equiv ||p-q||_1, \tag{2.3}$$

where $|\cdot|$ is the absolute value. The Manhattan distance is less sensitive to the curse of dimensionality than Euclidean distance and might be preferable in high dimensions [24].

As previously mentioned, a similarity measure serves the same purpose as a distance metric in our problem setting. One commonly used measure is *cosine similarity*, defined as

$$\text{sim}_c(p,q) = \frac{\sum_{i=1}^n p_i \cdot q_i}{\sqrt{\sum_{i=1}^n p_i^2} \cdot \sqrt{\sum_{i=1}^n q_i^2}} \equiv \frac{p \cdot q}{||p||_2 ||q||_2}. \tag{2.4}$$

Since the dot product is normalized by the (Euclidean) length of the vectors, the cosine similarity measures the cosine of the angle between them. As a result, the cosine similarity is close to 1 for very similar vectors, and -1 for opposite ones. Cosine similarity differs from the previously mentioned distance metrics in that it is insensitive to the vector magnitudes.

2.3.2 Loss Functions

In this subsection, we will cover some different loss functions used in contrastive learning, starting with simple examples and then increasing complexity. The loss functions L in this chapter are computed on latent representations denoted $z = f_e(x)$, such that $z \in \mathbb{R}^n$ is an n-dimensional feature vector.

One of the first contrastive loss functions, usually referred to as *pair* loss, was originally formulated by Hadsell, Chopra, and LeCun [25]. For two vectors z_i, z_j, the pair loss is defined as

$$L(z_i, z_j) = (1-Y)\frac{1}{2}||z_i - z_j||_2^2 + Y\frac{1}{2}\max(0, m - ||z_i - z_j||_2^2). \tag{2.5}$$

Here, Y is a binary variable that is equal to 0 if z_i and z_j make a positive pair, and Y is 1 if they make a negative pair. For short, we will refer to a positive pair as just positives, and analogously for a negative pair. When z_i and z_j are positives, the objective is to minimize the squared Euclidean distance between z_i and z_j. However, when z_i and z_j are negatives, the goal is to maximize the distance between them, up to a fixed margin m. The purpose of the margin is to act as an upper bound between negative samples. Increasing the distance more than the margin does not decrease the loss. Lastly, the pair loss $\mathcal{L}$ is computed over all possible pairs (i, j) within in the batch of size N,

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{j=1;j>i}^{N} L(z_i, z_j). \tag{2.6}$$

Instead of using pairs, one can use triplets instead which leads us to the *Triplet margin loss*. This loss function is explained and used by Balntas, Riba, Ponsa, *et al.*

[26], and considers a triplet of feature vectors (z_a, z_p, z_n). Such a triplet consists of an anchor z_a, a positive z_p and a negative z_n. The objective is to map z_a and z_p close together in the embedding space, and z_a and z_n far apart. This can be formulated with the condition

$$||z_a - z_p||_2 + m < ||z_a - z_n||_2, \tag{2.7}$$

where m is a margin parameter comparable to the one in (2.5). With a larger margin, the negative is pushed further away from the anchor compared to the positive. Thus, the loss is defined as

$$\mathcal{L} = \sum_{a=1}^{N} \max\left(0, ||z_a - z_p||_2 - ||z_a - z_n||_2 + m\right), \tag{2.8}$$

where z_p and z_n are chosen specifically for each z_a. The total loss can be computed using all possible positives and negatives. However, the number of triplets scales quadratically with the number of anchors, so it is common to only calculate the loss for a subset of the possible triplets. It is important that the negatives z_n and positives z_p are sampled with care. There are different techniques for triplet sampling, and many include picking z_n and z_p such that $||z_a - z_n||_2 - ||z_a - z_p||_2 < m$ [27]. This results in non-zero loss so that the model can learn something from all chosen triplets. One particular approach is to choose the hardest possible negative that minimizes $||z_a - z_n||_2$. Though, this choice can make the optimization task too difficult leading to a collapsed model [28]. This means that the model maps every input to approximately the same point, which occurs if the smallest possible value attainable for the loss function is the margin value m.

Both the pair loss and triplet margin loss are formulated using the Euclidean distance, but as mentioned in section 2.3.1, it is possible to reformulate the loss functions to use other distance metrics or similarity measures. One type of contrastive loss that comes with another similarity measure is the *NT-Xent* loss, short for *Normalized temperature scaled cross entropy* loss. This loss function is found in SimCLR [1], where the positive embeddings come in pairs (z_i, z_p) that have been created by augmenting the same original sample x in different ways. This is done for each sample within the batch of size N, resulting in $2N$ augmented samples. Each augmented sample is assigned an index from the set $I = \{1, ..., 2N\}$. For $i \in I$, the index of an arbitrary augmented sample and p, the index for the other augmentation from the same sample as i, the loss contribution from z_i is defined as

$$L_i = -\log \frac{\exp\left(\text{sim}(z_i, z_p)/\tau\right)}{\sum_{j \in I \setminus \{i\}} \exp\left(\text{sim}(z_i, z_j)/\tau\right)}. \tag{2.9}$$

where $\tau \in \mathbb{R}^+$ is a temperature parameter, and sim is the cosine similarity, see eq. (2.4). The numerator consists of the similarity between the anchor with index i and the positive p, i.e., one positive pair. This is divided by the similarities between all augmented samples, in total $2N - 1$ pairs. To decrease the loss, the exponent of the positive similarity should be maximized and the exponents of the negative similarities minimized.

In order to obtain the total loss for all samples I in a batch, the individual contributions are simply summed,

$$\mathcal{L} = \sum_{i \in I} L_i. \tag{2.10}$$

A small value of the temperature parameter τ creates a large difference between the exponentials of similar versus dissimilar pairs. By decreasing the value of τ, the encoder is more encouraged to increase similarities between positives rather than decrease similarities between negatives. Tuning τ to an optimal value can lead to improved downstream classification performance [1].

In supervised settings, where the number of positives per sample is known and can be more than one, the *NT-Xent* loss can be generalized. One such implementation is Supervised contrastive (*SupCon*) loss, found in [29], which is defined as

$$\mathcal{L} = \sum_{i \in I} L_i = \sum_{i \in I} -\log \left[\frac{1}{|P(i)|} \sum_{p \in P(i)} \frac{\exp\left(\text{sim}(z_i, z_p)/\tau\right)}{\sum_{j \in I \setminus \{i\}} \exp\left(\text{sim}(z_i, z_j)/\tau\right)} \right]. \tag{2.11}$$

Here, $P(i)$ is the set of all other indices of samples belonging to the same class as z_i. Apart from the averaging over positive loss contributions, this loss follows the same notation as the *NT-Xent* loss. Khosla, Teterwak, Wang, *et al.* [29] have shown that leveraging label information in this way can increase the accuracy in downstream classification of images.

2.4 Preventing Overfitting

Overfitting is a central problem in deep learning, where the model learns features during training that cannot be effectively generalized to unobserved data. To detect overfitting, one common approach is to split the data into training and validation sets. The model is then trained on the training data, while the loss is calculated separately for both the training and validation data during the training process. Ideally, the training and validation loss should follow a similar trend, indicating that the model is learning generalizable features that can describe the validation data. However, if the training and validation loss start to diverge, it is an indication of overfitting. In order to mitigate overfitting, various techniques can be applied, some of which will be discussed in this section.

2.4.1 Reducing Model Capacity

As described by Goodfellow, Bengio, and Courville [22], the capacity of a model refers to its ability to learn a diverse range of functions. It is important to choose an appropriate model capacity that matches the complexity of the task to achieve optimal performance. If the model capacity is too high, the risk of overfitting increases, while too low of a capacity may result in underfitting, meaning that the model cannot adequately address the complexities of the task. The goal is to find the optimal capacity that minimizes the validation loss. A straightforward approach

to reduce overfitting is to decrease the model capacity. In the case of neural networks, the capacity can be controlled by adjusting the number of parameters, such as the number of nodes and layers.

2.4.2 Data Augmentation

One cause of overfitting can stem from not having enough data. The ideal solution to address this issue would be to acquire more data, although this may not always be feasible in practice. An alternative approach to mitigate this problem involves generating new data samples from the existing dataset. This technique, known as *data augmentation*, aims to introduce greater diversity into the dataset. The application of data augmentation has demonstrated successful outcomes across various domains, particularly in image analysis [1], [10] but also in speech recognition [22].

When applying data augmentations, the generated data must be reasonable and meaningful. What reasonable means may vary depending on the data type as well as the task. In the case of image classification, common augmentation techniques include rotations, flips, blurring, and cropping. These augmentations are considered reasonable because, for instance, an image and its rotated counterpart are inherently the same image. It can therefore be said that the style of the image is modified, while the semantic content remains the same. Compared to images, data augmentation techniques for time series data can take on very different forms since the content is represented differently. In an empirical examination of data augmentation techniques for time series classification, Iwana and Uchida [30] explored various common techniques such as jittering, scaling, and window warping with different architectural designs. Their findings indicate that while certain techniques enhance generalizability, making them preferable over others, the effectiveness also depends on the specific architecture being used.

2.4.3 Weight Decay

Another approach to regularize the model is to target the parameters directly. This can be achieved through L_2-regularization, also known as weight decay. L_2-regularization works by adding a term to the loss function that penalizes large squared norms of the weights. The modified objective function thus becomes

$$L + \lambda ||W||_2^2, \tag{2.12}$$

where L represents a suitable loss function, W the vector containing all the weights in the model and $\lambda \in \mathbb{R}^+$ is a scaling parameter. The penalty term $\lambda ||W||_2^2$ encourages smaller weights, effectively restraining the model from overfitting the training data too closely. Increasing λ too much will, however, ultimately lead to every weight being zero, i.e. the model maps every input to 0. By reducing the magnitude of the weights associated with different features, L_2-regularization mitigates the risk of the model overemphasizing noise or irrelevant patterns, which would otherwise deteriorate the generalization performance [31]. This regularization approach thus implicitly encourages the model to learn more robust representations.

2.4.4 Batch Normalization

Batch normalization is a technique proposed by Ioffe and Szegedy [32] that has been shown to significantly improve the training process of some neural networks. It addresses the challenge of internal covariate shift, which refers to the change in the distribution of inputs between layers as the previous layers of a network are updated during training. This issue becomes particularly pronounced in deep neural networks, as the problem of changing distributions becomes amplified by each previous layer.

The key idea behind batch normalization is to normalize the inputs of each layer within the network. This normalization process involves calculating the mean $\mu_{\mathcal{B}}$ and standard deviation $\sigma_{\mathcal{B}}$ within a mini-batch $\mathcal{B}$ of size N. For a d-dimensional input $x_i = (x_i^{(1)}, \ldots, x_i^{(d)})$ in the batch $\mathcal{B}$, the mean and variance are computed by

$$\mu_{\mathcal{B}} = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad \sigma_{\mathcal{B}}^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2. \tag{2.13}$$

To normalize the inputs, each x_i is transformed using

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}. \tag{2.14}$$

Here, ϵ is a small constant added for numerical stability. However, this normalization step alone can limit the expressive power of the layer [22]. To retain the flexibility of the layer, the normalized inputs $\hat{x}_i$ are then scaled by $\gamma \in \mathbb{R}^d$ and shifted by $\beta \in \mathbb{R}^d$ which are learnable parameters. The resulting final output y_i is defined as

$$y_i = \gamma\hat{x}_i + \beta \equiv BN(x_i), \tag{2.15}$$

where BN denotes the batch normalization unit. The γ and β parameters allow the network to learn each feature's optimal scaling and shifting, contributing to faster training [22].

In addition to its impact on training speed, batch normalization has been observed to have a regularizing effect, promoting better generalization in neural networks and sometimes removing the need for other regularization techniques [32], [33]. This regularization property can be attributed to the fact that the mean and standard deviation are computed on a per-batch basis. As batches are randomly created during training, this introduces noise that acts as a regularizer, contributing to improved generalization [34].

2.5 Evaluation

When creating a machine learning model, evaluation is crucial to determine how well the model solves its designed task. Since the definition of a well-performing model often varies depending on the context, the optimal way of evaluation is problem-dependent, based on the task and available data.

In supervised settings, evaluation is simplified since there is a ground truth against which the model's output can be compared. This kind of evaluation can be used for contrastive learning performed on labelled data. To give an example, SimCLR [1] does not consider the image class labels during the training of the actual network. Instead, a small linear classifier is trained on top of the frozen network, in order to gauge how informative the created latent embeddings are. With the image labels as targets, the linear classifier will learn to better classify the embedded images if the contrastive learning model has kept a lot of information and made it more easily accessible.

In order to evaluate the performance of these classifiers and thereby the quality of the embeddings, standard metrics can be used such as accuracy or F_1 score. The F_1 score is the harmonic mean of recall and precision [35],

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}}, \tag{2.16}$$

where

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{2.17}$$

The recall measure can be described as the fraction of positive samples that are correctly classified, whereas the precision is the fraction of all positively classified samples that are indeed true positives. As a generalization to a multi-class setting, the F_1 score can be averaged over all classes. All classes can then be given the same weight (*macro-averaging*) or different weights based on the class frequencies (*micro-averaging*). The latter can be more appropriate when the classes are imbalanced, as each sample is then given the same weight instead of each class.

In unsupervised settings, the lack of ground truth prevents the use of evaluation methods measuring some form of concrete correctness in the output. Instead, different aspects of the output can be examined, that are presumed to correlate with model performance. Such metrics are often seen in cluster analysis [36], where distinct grouping is desired. One example of a metric used for evaluation of unsupervised group formation is the *Davies-Bouldin index*, which compares the dispersion difference and distance from each group G_i, for $i = 1, ..., k$ to their most similar one G_j [37]. Given two cluster groups G_i and G_j with centroids C_i and C_j, their similarity is defined as

$$R_{ij} = \frac{s_i + s_j}{\|C_i - C_j\|_2}, \tag{2.18}$$

where s_i denotes the average Euclidean distance from each sample in G_i to its centroid C_i. For an illustrated example, see fig. 2.3. A group is only compared to its most similar one, so the Davies-Bouldin index is defined as

$$DB \equiv \frac{1}{N} \sum_{i=1}^{N} \max_{j \neq i} R_{ij}. \tag{2.19}$$

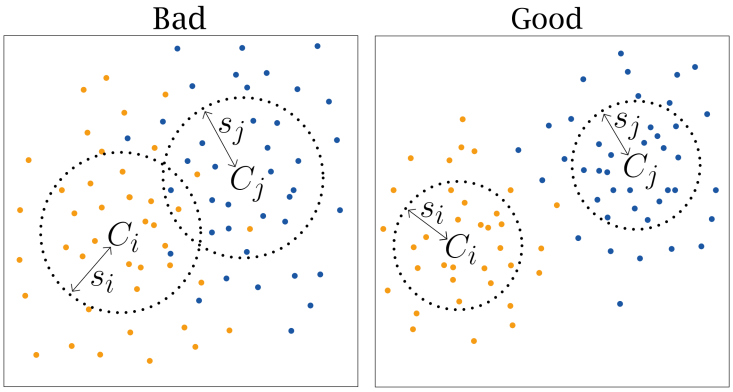


Figure 2.3: Illustration of two different similarity cases with cluster groups G_i (orange) and G_j (blue). Left: The clusters are overlapping, with relatively large average distances s_i, s_j to the cluster centroids C_i, C_j compared to the distance between the centroids. Right: With s_i, s_j being smaller and the centroid distance $\|C_i - C_j\|_2$ being larger, the similarity R_{ij} becomes lower compared to the left case.

Since R_{ij} is non-negative, the lowest possible value of DB is 0. A lower index indicates a larger distance between groups and lower within-group dispersion. This metric can be applied to the latent representations created by a contrastive learning model. Given some kind of label indicating which embeddings belong to the same class, the Davies-Bouldin index could then provide insight into how well the model manages to group the same-class samples.

2.6 Good Practice in Contrastive Learning

When it comes to developing machine learning applications, we are often confronted with a vast number of choices, parameters and combinations to explore. It is therefore helpful to have insight into previously successful approaches that can offer valuable guidance for determining the next suitable step. Regarding contrastive learning, some factors are generally considered particularly important for high performance in previous studies [10].

2.6.1 Projection Head

In the area of contrastive learning, a projection head is a small additional network appended to the encoder. Such a projection head can vary in size and form, but is often a small non-linear multi layer perceptron (MLP) [1], [38]. When using a projection head h, the embeddings $f_e(x) = z \in \mathbb{R}^n$ are projected to a lower dimensional space where the loss is computed. The projected embeddings $h(z)$ are only used for loss computation during training, while the embeddings z are used for evaluation and downstream tasks.

Including a projection head in the model has been shown to increase the generalization performance and suitability for downstream tasks of the latent representations

z [1]. However, the reason for this is not entirely understood [38]. Since the projection head constitutes a nonlinear mapping to a lower-dimensional space, a subset of the feature space is implicitly selected for the loss computation. As a result of this dimensionality reduction, different latent embeddings z get mapped to the same projected embeddings $h(z)$ resulting in a non-trivial null space of h. Gupta, Ajanthan, Hengel, *et al.* [38] examined this null space and found that it contained informative aspects of the representations, that for some datasets could yield better results in downstream tasks than the projected embeddings $h(z)$. This indicates that minimizing the contrastive loss by achieving invariance to augmentations is not equivalent to creating informative and generalizable latent representations. The projection head can thus be seen as a way to allow the model more freedom in forming generalizable latent representations, while still having the projected embeddings that are best suited to minimize the loss.

2.6.2 Other Aspects

One data-related factor considered to be important is the number of negative samples and their quality for each anchor sample [39]. With few negatives, the task to separate the anchor sample from the negatives is not difficult enough for the model to learn meaningful contrastive features. Using a larger batch size increases the diversity of negatives in the batch, thus increasing the contrastive complexity of the task. This has been shown to result in better performance [1], [39]. Another approach that deals with the quality of negatives is through *hard negative mining*. These methods increase the difficulty of the task by searching for negatives that are close to the anchor sample in the embedding space. The hard negatives are then used during training for that particular anchor. Using hard negatives is similar to increasing the batch sizes as both share the effect of making the contrastive task more difficult, which has been shown to advance performance [40]. The difference lies in that increasing batch size just increases the probability of getting hard negatives in the batch, whereas hard negative mining is based on actively searching for these samples.

Another factor that has been advancing performance is the use of extensive data augmentation [1], [41]. In unsupervised settings, this step is even necessary for the model to learn which samples should be interpreted as similar since the augmentations enable the model to create supervisory labels itself. We have covered some data augmentation techniques commonly seen in image analysis and time series analysis in section 2.4.2.

Yet another interesting aspect is the choice of loss function. Chen, Kornblith, Norouzi, *et al.* [1] show results that demonstrate the importance of their chosen loss function, NT-Xent loss. However, Musgrave, Belongie, and Lim [42] claim that the loss function is not as important as previously thought. The authors claim that the increased performance of newly developed constructive loss functions is due to other factors. Additionally, they show that they can comparable performance can be achieved using different losses by tuning hyperparameters properly.

3

Base Model Method

In order to find features in the data indicating behavioural change, the model must learn to create similar representations in the feature space for trajectories made by similarly behaving rats, and dissimilar representations for trajectories that come from rats with different behaviour. Since contrastive learning revolves around placing similar sample points close together, and dissimilar points far apart in the feature space, the technique appears to be appropriate for the problem at hand. The process of finding such a model and representation can roughly be summarized as follows:

1. **Dataset and Preprocessing**
   This first step includes outlier removal and filtering of the data, as well as any coordinate transformations or data augmentations.

2. **Encoder and Data Embedding**
   A neural network is used to convolve the time series, capturing both local and global features into lower-dimensional embeddings.

3. **Loss Computation**
   The loss function aims to minimize the distances between positive samples while maximizing the distances between negative samples, where the positive or negative relation between two samples is determined by their respective substances and doses. The loss gradients are then backpropagated through the network in order to train the model.

4. **Evaluation**
   After training, the model and its embeddings are evaluated based on several metrics, so that comparisons can be made against other variations as well as the manual reference model.

In this chapter, we will describe the dataset and outline the method used to develop our base model. In chapter 4, this model will then be subject to various modifications in order to more thoroughly examine the suitability of contrastive learning for our specific problem and data.

3.1 Dataset

A key to understanding the method choice lies in knowing the dataset that this study is based on. The dataset can broadly be described as consisting of trajectories

from rats that have been administered different substances and doses, and whose movements were then recorded over a period of time. After substance administration, the rat is placed in a cage measuring 40×40 cm. Rows of infrared sensors mounted by the floor on each side of the cage are used to compute the two-dimensional (x, y)-coordinates of the rat. A separate row of sensors located higher up in the cage provides a binary z-coordinate meant to indicate whether the rat is rearing or down on all four paws. The trajectories are recorded at a frame rate of 25 Hz but vary slightly in duration, with the most common length being marginally more than 60 minutes.

3.1.1 Metadata

The trajectories have corresponding metadata, consisting of a set of labels for every trajectory. The labels relevant to this study are:

- **Experiment**
  Each trajectory is part of a so-called *experiment*, with the purpose of examining some specific substance at different dosages. On average, an experiment involves 20 rats, comprising groups of four to five where all rats in each group are given the substance at some dose level. One of the groups is always kept as control, meaning that no active substance is given. Thus, there are almost always at least four or five rats that have been given the same substance and dose.

- **Rat id**
  Each rat in an experiment is given an id-number, between 1 and and the total number of rats involved in that experiment. This number combined with the experiment label creates an id that is unique to that trajectory.

- **Primary substance**
  There are roughly 1000 unique substances tested as primary substance, and some are used in multiple experiments. Some substances tested have well-known effects and act as references against which the experimental substances can be compared.

- **Secondary substance**
  In order to assess the efficacy of some potential symptom-reducing substance, it might sometimes be necessary to first induce those symptoms and then administer the drug of interest to see if the rat returns to normal behaviour. In such experiments, the symptom-inducing substance is administered beforehand as the *secondary substance*. Again, one group of rats is given no active substance in order to remain as control. One other group is only given the secondary substance, also as a form of control. The remaining groups are most commonly given one fixed dose of the secondary substance, and different dosages of the primary substance. There are 19 secondary substances, of which two (hereafter denoted as MK and AMF) are used in 95% of the trajectories with double substances.

- **Primary dose**
  The different dose levels for some primary substance are often chosen to capture the largest possible range of effects. Three to four dose levels are most common for an experiment.
- **Secondary dose**
  For any secondary substance, the dose is generally fixed at some level that induces the symptom-like response which is meant to be treated.
- **Indication**
  This broad label regards the field of use for the specific substance, for example antipsychotic, anaestethic or antiparkinson. Any new substances being tested are labeled as experimental.

In total, the dataset consists of approximately 33,000 trajectories. Since each experiment contains four to five control trajectories, the total control group is considerably larger than other groups, with more than 7000 trajectories. The same applies to the secondary substance control trajectories where MK or AMF is used, but to a lesser extent. The metadata labels that will be used for training the model are primary substance, secondary substance, primary dose and secondary dose. Together, these determine whether or not two trajectories should be considered similar or not, which will be further explained in section 3.3.

3.1.2 Preprocessing

The infrared sensors are configured in such a way that the rat can be detected at 120 discrete positions in the x- and y-direction, respectively. However, unknown errors in either the sensors or the way that the readings are combined can sometimes cause the rat to be assigned an incorrect position. When the position is only slightly off, it becomes hard to detect without a ground truth, so a more thorough analysis is needed to assess the significance of this potential problem. On the other hand, any such small errors might also have a negligible impact on the quality of the data. Easier to detect are the occasions when the rat is assigned a position outside of the cage. In such situations, the rat's position is linearly interpolated using the most adjacent valid positions.

Another aspect of the data that needs to be considered is that the length of different trajectories varies. Many established machine learning libraries assume that data samples share dimensions, in order to more efficiently perform the computations needed. Therefore, the trajectories are set to an equal length of 60 minutes, or 90,000 time steps. Trajectories shorter than 83,000 time steps are excluded from the dataset, and those longer than 83,000 but shorter than 90,000 are extended using the last position of the rat to simulate it being stationary for the remainder of the time. The vast majority (95.5%) which are longer than 90,000 time steps are simply shortened by removing the end of the trajectory.

Using the complete recordings of 60 minutes for each trajectory could be questioned, since the effect of substances decays with time. Thus, many rats might return to "normal behaviour" at the end of the recordings, which could potentially make the

last part of the trajectories non-informative. However, the decay rate for substances and doses varies. Therefore it would difficult to determine when the behavioural effect of each substance becomes insignificant, and there would be a risk of removing useful information. Apart from this potential risk, decreasing the length of the trajectory in this manner would also inject more bias into the model which we wish to avoid. Additionally, a suitable model should be able to handle irrelevant features, so keeping the full length of the trajectory should not be considered a problem.

In order to make it easier for the machine learning model to learn from the data, the (x, y)-coordinates are mapped from the coordinates of the quadratic cage to values between -1 and 1. The dimension of a preprocessed trajectory is $(3; 90,000)$, with (x, y, z)-coordinates where $x, y \in (-1, 1)$, and $z \in \{0, 1\}$.

3.2 Encoder

Following the data preprocessing, our encoder is made to take inputs of shape $(3; 90,000)$ with the goal of transforming the data into an informative vector representation of shape $(C, 1)$, where C is the dimension of the output vector. This is achieved through a neural network consisting of multiple layers that gradually reduce the temporal dimension from 90,000 to 1 while increasing the number of channels, from 3 to C.

The neural net employs strided one-dimensional convolutions in the time domain to capture temporal relationships and increase the number of channels with layer depth. The architecture comprises repeating blocks with three distinct parts: strided convolutions, normal convolutions, and residual connections. The structure is illustrated in fig. 3.1 and is largely inspired by the ResNet architecture proposed by He, Zhang, Ren, *et al.* [11].

In each block, the first layer is a strided convolution with a stride and kernel size of 2, which halves the time dimension. The number of output channels increases exponentially up to a fixed limit set to C, where $C = 100$ for the base model. This convolution is followed by a ReLU activation. After the strided convolution, a normal convolution with kernel size 3 and padding 1 is employed, keeping the data dimension unchanged. Following the normal convolution is an addition with a residual layer, followed by a ReLU activation which concludes the block. However, to perform the addition, the dimensions of the parallel layers must match. Therefore, the residual layer contains a pooling function to ensure that the time dimensions agree, with additional zeros added in the channel dimension to account for the channel increase. Lastly, we also use zero-padding when needed to simplify dimension agreement in the computations within each block. This ensures that each output in the time domain is divisible by two before moving on to the next block. The block structure is repeated 15 times resulting in a 30-layer deep CNN. At this stage, the dimensions of the data are (100, 3). Therefore, the encoder ends with an average pooling in the time dimension to get the final output of shape (100, 1).

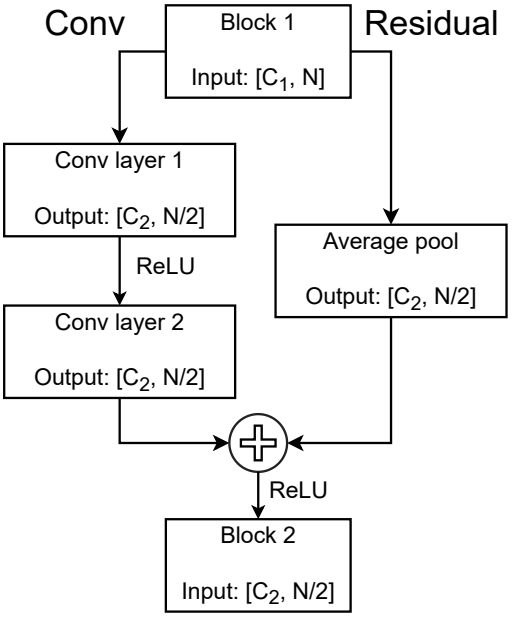


Figure 3.1: Block structure of the encoder. The first convolutional layer has stride 2 and decreases the time dimension by half while increasing channel size. The second convolution keeps input dimensions unchanged, its output is added with that of the residual pooling layer and is then passed to the next block.

The motivation for this encoder structure is derived from Temporal Convolutional Networks (TCNs) and Residual Networks (ResNets). By incorporating temporal convolutions, the encoder has the ability to capture local features within the data, which are crucial for understanding the behavioural aspect of trajectories. At the same time, the deep network structure enabled by residual connections allows the encoder to learn more abstract and global behavioural aspects of the data. For a complete version of the output dimensions and parameter specification of each layer in the encoder, see table A.2 and table A.1 in appendix A.

3.3 Loss

In order to train the encoder, a loss function is required. We use the SupCon loss function [29], explained in section 2.3.2. The loss for all embeddings $z_i \in \mathbb{R}^n$, $i \in I$ of a batch is defined as

$$L = \sum_{i \in I} L_i = \sum_{i \in I} -\log \left[\frac{1}{|P(i)|} \sum_{p \in P(i)} \frac{\exp\left(\text{sim}(z_i, z_p)/\tau\right)}{\sum_{j \in I \setminus i} \exp\left(\text{sim}(z_i, z_j)/\tau\right)} \right]. \tag{3.1}$$

One difference in the base model implementation compared to the original implementation of SupCon can be found in the similarity measure. Instead of cosine similarity, negative Euclidean distance is used such that

$$\text{sim}(z_i, z_j) = -||z_i - z_j||_2. \tag{3.2}$$

The motivation behind this choice lies in the hypothesis that a larger dose should generally yield a greater behavioural response, which would mean that the magnitude of an embedding can contain important information regarding the behaviour. Using cosine similarity, as seen in other studies [1], [29], removes this aspect by normalizing all embeddings before computing the loss. While both Euclidean distance and cosine similarity suffer from the curse of dimensionality, Manhattan distance is said to perform better in high dimensional settings [24]. We will therefore examine the effect of using these different similarity measures in section 4.4.4. Additionally, different choices of temperature τ will be studied in section 4.4.2, but for the base model, τ is set to 1.

3.3.1 Defining Positive and Negative Relations

The positive and negative relations between samples play a key part in how the model is trained. Given a batch with indices $I = \{1, ..., N\}$ and a sample with index i, we define $P(i)$ to be the set of indices of all other samples in I with the same substance/dose combination as i. This means that $P(i)$ is the set of positive samples with respect to i, and the rest, $I \setminus (P(i) \cup \{i\})$, makes up the negatives with respect to i.

This rule for relations between samples is quite restrictive, as it says that two trajectories from rats administered the same substances but different doses make a negative pair. In many cases, this means that a sample only has 3 or 4 total positive pairs in the entire data set, since a substance is often tested in only one experiment. Thus, it is very unlikely that we by random get even one positive pair per sample in a batch using this relation rule. Another effect is that samples with the same substance but slightly different doses will have the same relation (negative) as two samples with completely different doses and substances.

An alternative relation definition could be that two samples are considered positive if they share substances, regardless of dose. However, this would be problematic when the behavioural response of some low substance dose is more similar to the control group than a higher dose of the same substance. Bridging this gap between low-dose trajectories and the control groups would likely require domain expertise and manually decided thresholds for when a substance-specific dose is low enough to be positive with control. Such manually decided relations also come with the risk of introducing bias into the model, which we wish to avoid. Yet another possible definition could incorporate doses in a more nuanced way, by considering trajectories with the same substance but different doses as *semi-positive*. This modification of the base model will be further presented and discussed in section 4.4.

3.3.2 Data Splitting and Sampling

As previously mentioned, a consequence of the strict rule for positive pairs is that random sampling would result in batches with very few positive pairs. This problem is solved by querying positive pairs instead of individual samples from the dataset. More specifically, the unique id of one anchor sample is used for the query, and the

dataset returns that sample together with one random positive sample. This means that duplicates can occur, but these are not considered during loss computation. In order to ensure that each sample is seen by the model, all unique id:s are used for queries during one epoch. A batch of size $2N$ thus contains N anchor samples and N samples positive to at least one of those anchors. This sampling method is similar to the one used in SimCLR [1], but instead of batching two different augmentations from the same sample, we add another sample with a known positive relation, for each anchor sample in the batch. From this perspective, positive trajectories can be viewed as augmented versions of the same underlying behaviour.

The data is split into training (75%) and validation (25%) datasets using the same seed every run for reproducibility. The split is performed on experiment-level, meaning that all trajectories belonging to one experiment are placed in one of the two datasets. This kind of split is motivated by the use case, where a new experiment with a previously unseen substance is embedded by the trained model for behavioural comparison. It would therefore not make sense to evaluate the model using trajectories whose positive counterparts were used for training.

3.4 Evaluation

With the goal of the model being to find features indicative of behavioural change, a well-made evaluation protocol should measure to which extent this goal has been achieved. While the validation loss is correlated to a model's ability to find such features, only considering the loss is not enough since it also changes based on the parts and hyper-parameters of that specific model configuration. Additionally, the lack of ground truth information about which substances should induce which type of behaviour makes evaluation non-trivial. Although being less complete than ground truth labels, the metadata labels can however be used to check some desired properties of the representations, that should be observable in the representations. The properties that we have chosen to take into consideration are as follows:

- **Low Within-group Dispersion, High Global Dispersion**
  Here, a group can be defined differently depending on which metadata label that is used. When using the substance and dose labels, one group is defined as all rats that were given the same substances and doses. As with any clustering problem, the groups should be dense and clearly delimited, which can be measured in relation to the global set of representations.
- **Positive Nearest Neighbours**
  Since the model is supposed to place similar samples close to each other, one intuitive property that should be observed is that the closest neighbouring belongs to the same group. Here we use substances and doses to define the groups.
- **Higher Doses Further Away from Control**
  The model should be able to capture some form of dose-response relationship, which describes how a rat's behaviour changes depending on the dose. Exactly how this dose-response appears will depend on the specific substance, but one

general property that should be fulfilled is that larger doses tend to push the samples further away from the control groups.

- **Between-group Separation**
  While the property of low within-group dispersion, high global dispersion has already been considered, it might also be of interest to measure how well the groups are separated. Some groups should not necessarily be separated, for example, those with substances that give similar behavioural responses. However, under the assumption that a majority of the substances in the dataset give rise to different behaviours, between-group separation is generally desirable.
- **Informative Representations**
  It is important that the model creates the representations in a way that not just minimizes the loss, but also preserves as much information as possible. This can be measured by training a small classifier on the representations against some metadata label. This can indicate if the representations are informative enough for the classifier to learn which representations belong to which class.

Below, we describe the mentioned properties in more detail, as well as the scoring methods used to measure each property. Some additional evaluation methods are also used, that are less objective and concrete, but also offer a more broad and wide-scope perspective of aspects that are difficult to measure. These methods involve visual inspection of the representations, and examination by domain experts at IRLAB.

3.4.1 Low Within-group Dispersion, High Global Dispersion

The within-group dispersion should remain low, while the total dispersion of the embeddings is high. One way to measure this is to consider the quotient of the within-group sum of squares (WSS) over the total sum of squares (TSS). Let z_i denote an embedding with index i in the dataset of size N. With $\bar{z} = \frac{1}{N}\sum_{i=1}^{N} z_i$ being the centroid of the entire dataset, we define the total sum of squares as

$$TSS = \sum_{i=1}^{N} ||z_i - \bar{z}||^2. \tag{3.3}$$

Let $g \in G$ denote a group of samples with the same substances and doses, where G is the set of all unique groups. We define within the sum of squares for a group g as

$$WSS_g = \sum_{i=1}^{|g|} ||z_i^g - \bar{z^g}||^2 \tag{3.4}$$

where z_i^g is an embedding belonging to group g, and $\bar{z^g} = \frac{1}{|g|}\sum_{i=1}^{|g|} z_i^g$ is the centroid for group g. Lastly, we form the WSS-TSS ratio

$$\sum_{g \in G} \frac{WSS_g}{TSS}. \tag{3.5}$$

In this metric, the contributions from all embeddings are weighted equally. However, since the dataset is unbalanced, larger groups will have larger WSS and contribute more. Therefore, we also use another metric where each group contributes equally, called group average WSS-TSS which is defined as

$$\frac{\sum_{g \in G} WSS_g/|g|}{TSS/N}. \tag{3.6}$$

One aspect worth considering with both of these measures is that they only measure each group's dispersion against the centroid of the entire data set, thereby not considering the overlap between groups.

3.4.2 Positive Nearest Neighbours

Given some trajectory, there are on average three to four other trajectories created by rats that were given the same substances and doses. In an ideal scenario, the model would map these trajectories to the same representation, since the induced behavioural changes should be the same. Natural differences in behaviour among the rats prohibit this, but a better model should still group such samples closer together. One way to measure this property is by counting the number of embeddings whose nearest neighbour belongs to the same group. However, the score can be made more robust by considering the three nearest neighbours to an embedding, and whether at least one of those is of the same group as the embedding in question. Since there are often thousands of negatives for every positive, this metric should not be considered easy even if the three closest neighbours are considered instead of only one. For an illustrated example, see fig. 3.2. This group belonging of neighbours is noted for all embedded samples in the validation dataset, and the summed number is then divided by the total number of samples and converted to a percentage. This way, a score of 100 indicates that every embedding constitutes a same-group pair with at least one of its three nearest neighbours in the latent space.

This score is similar to the chosen objective function in that it improves when positive samples are grouped closer together than negative samples. What makes it beneficial as an evaluation metric is that it can still be used to compare models with different loss functions. As an additional upside, it has a clear intuitive interpretation.

3.4.3 Higher Doses Further Away From Control

Under the assumption that higher doses generally change the behaviour to a larger degree than lower doses, a better model would be expected to represent higher dose trajectories as being further away from the control trajectories. Since each substance is administered in around three to four dose levels, this property can be quantified by examining how well the different dose level trajectories are ordered, based on the distance to the corresponding control group.

As an example, say that a single substance s is administered to rats in the three different doses (0.3, 1.0, 3.0). The centroid C_s^i of each dose level is found using the

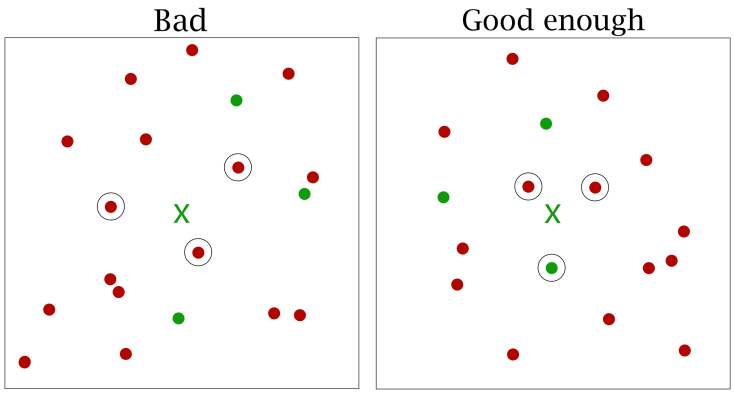


Figure 3.2: Illustration of two different score scenarios with respect to one embedding (green "X"). Left: All three closest neighbours are negative. Right: One of the three closest neighbours is positive, so the embedding contributes to the score.

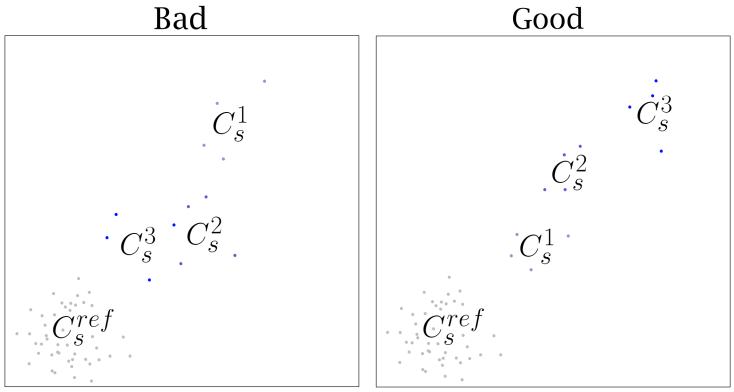


Figure 3.3: Visualisation of two of the dose response examples described below. Left: The dose level centroids C_s^1, C_s^2 and C_s^3 are incorrectly ordered with respect to the control centroid C_s^{ref}. Right: The centroids are correctly ordered, indicating that the model has successfully captured the dose response of substance s.

corresponding latent representations, and the Euclidean distances from these dose level centroids to the control group centroid C_s^{ref} are computed to be [3.7, 5.4, 7.1]. Thus, the model has correctly represented the trajectories so that higher doses get further away from control. Now, let's say that the corresponding centroid distances were instead [4.1, 3.4, 6.7]. The first and second dose levels are now incorrectly ordered, but the representation is still better than if the distances were [6.7, 4.1, 3.4], since it would be completely opposite to the earlier stated assumption of dose-response.

In order to measure the level of sortedness of the dose level centroid distances described above, while also accounting for the different degrees of incorrect orderings, we compute the *inversion number* of each such distance vector. Given a sequence v, an inversion in v is a pair of elements that are out of their natural order. The inversion number $\text{inv}(v)$ is the total number of inversions. This can be interpreted as the number of pairwise transpositions or swaps needed to get the sorted version

of v, while only allowing neighbouring elements to swap places. Using the example above, $\text{inv}([3.7, 5.4, 7.1]) = 0$, $\text{inv}([4.1, 3.4, 6.7]) = 1$, and $\text{inv}([6.7, 4.1, 3.4]) = 3$. Since only dose levels of the same substances can be compared, the inversion number is computed for each substance, or combination of substances in the case of double substance experiments.

For a vector v with n elements, the maximum inversion number is $n(n-1)/2$. This quadratic relation to n needs to be considered, since different substances can have a different number of dose levels. If a substance with two dose levels has an inversion number of 1, it is a bad representation since the dose level centroids are in the wrong order. On the other hand, the same inversion number for a substance with six dose levels means that at least four dose-level centroids are correctly represented, which is quite good. As a way to reduce this imbalance, each inversion number is divided by the number of dose levels tested for that substance.

Let S be the set of tested substance combinations, including single substances. Each $s \in S$ has D_s different dose levels, and a reference centroid C_s^{ref}. For single substance trajectories, the reference centroid will be that of the encoded control trajectories, where no active substance was administered. For double substance trajectories, the reference centroid will be computed using those trajectories where only that secondary substance was administered. The evaluation metric DR, used to measure the degree of dose-response captured by the model, thus becomes

$$\text{DR} = \frac{\bar{D}}{|S|} \sum_{s \in S} \frac{\text{inv}(v_s)}{D_s}, \quad v_s^i = \|C_s^{ref} - C_s^i\|_2, \quad i = 1, ..., D_s \tag{3.7}$$

where v_s is the distance vector for substance(s) s, C_s^i is the centroid for the i-th dose level of s, and $\bar{D}$ is the global average number of dose levels. Since each inversion number is divided by D_s, the total substance-averaged score is multiplied by $\bar{D}$ for interpretability. As a result, DR can be interpreted as the average number of pairwise swaps needed to correctly order a substance's dose level centroid distances.

3.4.4 Between-group Separation

In general cluster analysis, it is often desirable to have clearly defined, well-separated clusters. This would certainly be the case here as well, if there was available metadata regarding which substances yield similar behavioural responses and should thus belong to the same class. Since this information is not available, clear group separation cannot always be expected. For example, two adjacent dose levels of a substance that barely yield any behavioural response will likely be represented similarly, which is natural. The same applies to different substances that do not have any noticeable effect. However, a majority of the substances tested can be assumed to yield some behavioural response, and the doses are selected to cover as wide a range of response as possible. Therefore, a better model will most likely have a larger degree of separation between groups.

The *Davies-Bouldin index* described in section 2.5 is used to measure this property, which compares the dispersion difference and distance from each group G_i, for $i = 1, ..., k$ to their most similar one G_j. Several of the metadata labels included in the data can be used to define group affiliation, but the one we have chosen to use is the combined substance and dose labels. Therefore, all trajectories with the same substance and dose are considered to belong to the same group.

3.4.5 Informative Representations

It is important that the model creates representations in a way that does not just minimize the loss, but also preserves as much information as possible. A common way to measure this is by training a small classifier on the representations against some metadata label. This can indicate if the representations are informative enough for the classifier to learn which representations belong to which class. In our case, we use the label "Indication" described in section 3.1. Trajectories with two substances are labelled by concatenating the indications of each substance. Excluding those labelled "other" or lacking any indication label, there are almost 15,000 trajectories with 26 different valid indication labels. The substance/dose combinations used for training the model are not used as labels, since the average group size of 6.5 would be too challenging for the classifier. The indication labels have some disadvantages too, however. Some of the labels are quite broad, for instance, the "abuse" label that contains all illicit substances tested, regardless of their effect on behaviour. Another very wide label is "experimental", which includes all substances designed by IRLAB. Having such potentially vague labels, high evaluation scores should not be expected. However, we still consider the classifier to be worth including as an evaluation metric, since more informative embeddings should be reflected by a higher classifier score.

To classify the indication of a representation we use an MLP with one hidden layer of size 100 and ReLU activations. The choice to use a non-linear classifier instead of a more conventional linear one [1] is an attempt to compensate for the sub-optimal indication labels, by making the classifier slightly more capable. The input shape is the dimension of the representation, and the output is equal to the number of unique indications. To train the model we use cross entropy loss weighted by class frequency and the optimizer Adam [43] with a learning rate of 0.001. The performance is evaluated using micro-averaged F_1 score, as described in section 2.5. For simplicity, we train the MLP for a fixed number of 200 epochs and evaluate on the validation data after those 200 epochs, regardless of the encoder model.

3.4.6 Visual Inspection and Expert Assessment

In order to get a general understanding of how the embeddings are distributed, they are projected using PCA and visualized in two and three dimensions. A small subset of experiments are excluded from the training and validation datasets, and function as a test dataset for these visualization purposes. The substances used in these experiments have well-known and different behavioural effects. This enables pharmacological domain experts to examine the visualized embeddings, and assess whether the positions of different representations relate to each other in a realistic

way.

This form of visual inspection is also a good way to see how well the aspect of dose-response is captured by the model. Representations with the same substance but different dose levels should ideally form a straight or curved line, where lower doses are closer to the control group. While these kinds of visualizations have been a useful sanity check and occasionally guiding in the model development, we choose to focus more on the quantitative evaluation metrics in this report.

3.4.7 Evaluation Procedure

A subset of the evaluation metrics are computed once every epoch during training, in order to illustrate how the model evolves and changes. Each model is generally trained for 100 epochs, which simplifies comparisons between different models, and is sufficient to show the relevant aspects of how the model evolves. As will be seen in chapter 4, 100 epochs is enough for the models to start overfitting to varying extents, so evaluating the final model instances would not give a fair depiction of relative performance. Therefore, the model instance at the epoch with the lowest validation loss is also saved during training for a more extensive evaluation after training. At the point of lowest validation loss, the evaluated model instances also perform well with regard to the evaluation metrics, which will be seen in section 4.1, making this a simple but fair way to compare models between different runs.

Evaluating the models before they start overfitting also solves a problem related to the MLP classifier. An effect of splitting the data on experiment level is that some uncommon indication labels only occur in either the training or validation dataset. If the same split were to be used for the classifier, this imbalance would pose a problem since some target labels would be absent in one of the datasets. Therefore, the data is reshuffled and then split using the same proportions, but on trajectory level to ensure that both datasets contain all 26 unique labels. Since the evaluation is performed on the model instance with the lowest validation loss, the new split should not be unjustly favouring some models more than others based on the level of overfitting.

The number of features used by the models might differ. The manual model for example has 308 features, while our baseline model is set to 100. Since the number of features is also the number of dimensions in which the evaluation metrics are computed, differently-sized outputs might not give comparable evaluation scores. This is especially precarious since many of the evaluation metrics are based on Euclidean distance in the latent space. As a countermeasure, all representations are projected to a lower number of dimensions when computing these distance-based scores. To perform this projection, we use PCA and retain a fixed number of K principal components to project each representation. We choose the value of K by examining the amount of data variance explained by each principal component. Principal components with low explained variance are likely to lack meaningful information and can therefore be filtered out. This dimensionality reduction is also motivated by the use case of the model, where its embeddings will often be projected to two or three dimensions for visual examination. Since the projected embeddings

will only be used for evaluation and not training, the PCA is made on both the training and validation data. This is motivated by the use case as well, as new experiments will be embedded and then projected onto the same space as the other embeddings. The choice of K will be presented and explained in section 4.1.

4

Experimental Evaluation

In this chapter, we will present and analyze the results of different models. We will first show the performance of the base model with regards to all our evaluation metrics, and compare it to the manual model. As will be seen, the base model has problems with overfitting. Thus, the following sections will show results from attempts to reduce the overfitting behaviour and increase performance. For these sections, we will first introduce a model modification, and then present how the results differ from those of the base model. Each such model modification section will be followed by a discussion about the modification and results. A more extensive discussion of the general results is presented in chapter 5.

4.1 Base Model

The base model consists of the encoder and loss function described in section 3.2 and section 3.3, respectively. It is trained on the preprocessed data which steps are described in section 3.1.2, in batches of size 128. Additionally, the model is trained with the optimizer Adam [43] with $\beta = (0.9, 0.999)$ and the learning rate set to 0.0001. These are the settings that will be used when training all the presented models, except when a modification is clearly stated.

4.1.1 Results

The loss evaluated for training and validation data for the base model is seen in fig. 4.1. During the first 50 epochs, we note that the training and validation loss decrease in a similar manner. This indicates that the model learns features from the training data that can be generalized to the validation data. However, at approximately epoch 60, we see that the validation loss increases as the training loss decreases, i.e., the model is overfitting. This is more clearly visualized in fig. 4.2, where the base model loss also is compared against an untrained base model as well as the manual model. The untrained model produces embeddings using the randomly initiated model weights, which can be interpreted as a form of random feature selection. Therefore, this model should not be considered a strong baseline. Nevertheless, we chose to include it simply to provide a guideline for the metrics that are used. Furthermore, since the untrained base model and manual model are not trained, we only show their respective losses on the validation data. Despite

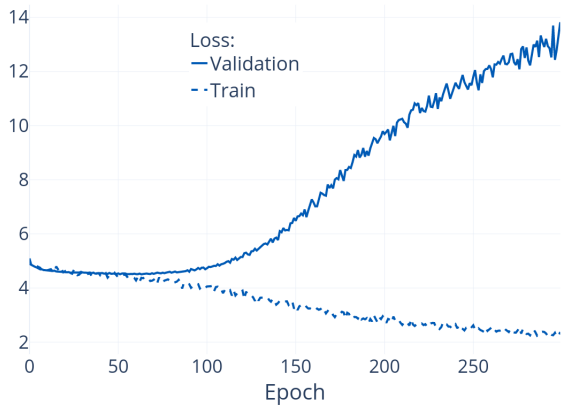


Figure 4.1: Training and validation loss of the base model during 300 training epochs. The different losses diverge and the model starts to overfit after roughly 60 epochs.

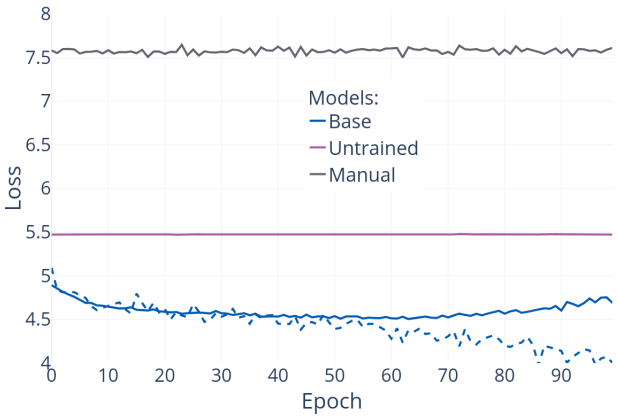


Figure 4.2: Comparison between the loss of different models. The base model reaches its minimum validation loss after 50 – 60 epochs. Surprisingly, the manual model has a higher loss than an untrained base model.

the manual model not being trained, its loss fluctuates across epochs. This is a consequence of different batch compositions between epochs, introducing randomness, which affects the loss value. On the other hand, the untrained model's loss is less variable across epochs since it lacks knowledge of the data. Therefore, differently composed batches between epochs matter less when computing the loss. Remarkably, the manual model has the highest loss, significantly higher than the untrained base model. The specific reason for this is unclear, but it suggests that using only the loss to determine model performance is insufficient.

It may seem like the base model is not learning much before overfitting, since the validation loss looks quite flat in fig. 4.2. However, there is a substantial decrease in the loss during the first epoch that is hidden. As the untrained model has a validation loss of 5.5, it means that the base model achieves roughly the same loss for the first batch during the first epoch. For every batch, the model is updated such that the mean validation loss for the first epoch has decreased to approximately 5, which is reported as the first value in the figure.

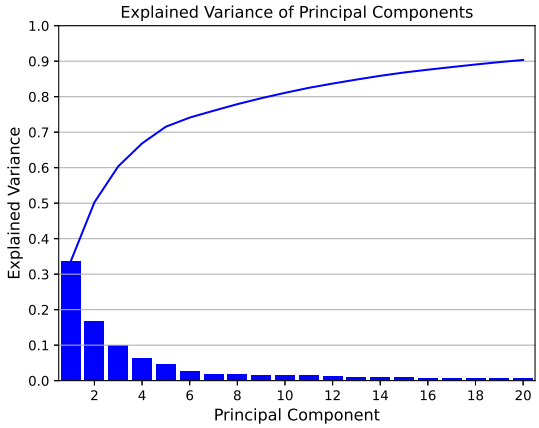


Figure 4.3: Explained variance among the first 20 PCs for the base model. We chose to reduce the dimension to 10 PCs, explaining about 80 % of the variance, before calculating evaluation metrics.

Before calculating the evaluation metrics, we project the embedding onto a lower dimensional space using PCA as explained in section 3.4. Choosing the number of principal components K to keep is done by examining the explained variance of each principal component, visualized in fig. 4.3. The figure shows how the explained variance decreases with each principal component and how the cumulative sum changes. We chose to keep the 10 first principal components, as this number is small enough to heavily reduce the dimension, and yet large enough to keep most of the explained variance (80 %) within the embeddings.

Furthermore, we track how the different evaluation metrics, explained in detail in section 3.4, change during the training. In fig. 4.4, the WSS-TSS ratio and group average WSS-TSS ratio are visualized. For the base model, we see that both scores increase during training. This indicates that within-group dispersion is increasing with training (relative to the total dispersion), which is undesired. The scores for the manual and untrained base model do not change as they are not trained, but instead are used as a reference. Comparing the manual and base model, we note that the base model starts better but gets worse with more training. At 60 epochs, where the validation loss is at its minimum, both WSS-TSS ratios are higher compared to the manual model. In the following results, we will only consider the Group average WSS-TSS ratio, as this metric is more suitable for imbalanced datasets and using both metrics seem redundant. The Group average WSS-TSS ratio is abbreviated ga-WSS-TSS, but for briefness and intuition, we also refer to it as dispersion.

The PNN metric for the manual, base and untrained model is plotted against the number of epochs in fig. 4.5. Recall that this metric gives the percentage of validation samples that constitute a positive pair with at least one of its three closest neighbours. The base model starts fairly low and steadily increases until 50 epochs, where it reaches its approximate peak of 24 %, aligning with its stage for the lowest validation loss. Additionally, the manual model lies at slightly above 23 %, showing a marginally worse performance than the base model at their respective peaks.

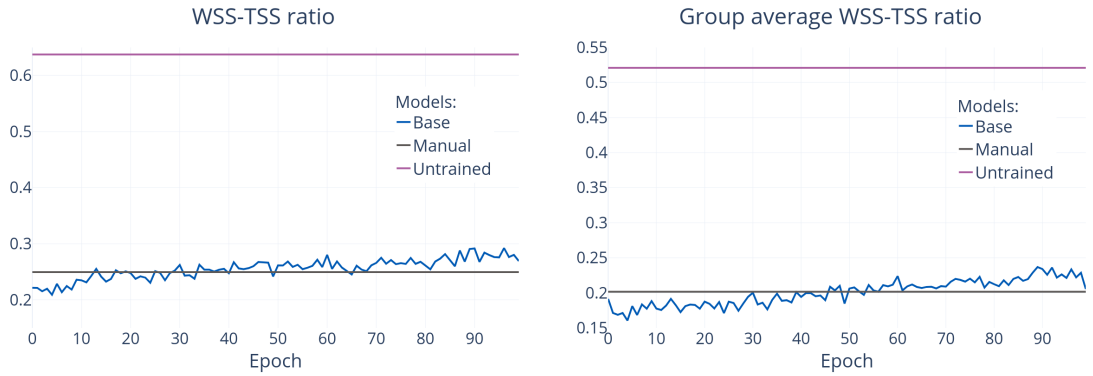


Figure 4.4: WSS-TSS ratios for the base and manual model. Low scores indicate low dispersion within groups, relative to the total dispersion, which is desirable. For the base model, both scores increase during training, suggesting a decline in performance with respect to this metric.

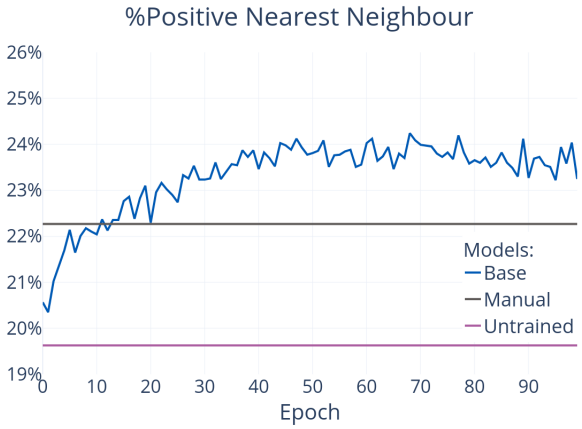


Figure 4.5: Percentage of positive nearest neighbour metric for the base and manual model during training. A high score is better, since it gives the percentage of samples that have a positive among its three closest neighbours in the embedding space.

Additionally, the score measuring dose reponse (DR), where a lower value indicates better captured dose-response relationship. is seen in fig. 4.6. The average number of required pairwise swaps decreases drastically for the base model during the first epochs, reaching a DR below 2. The score continues to decrease slowly with training. Compared to the manual model with a DR of 2.3, this indicates that the base model generally maps higher doses within a substance further away from the control group than the manual model does.

Furthermore, we measure the group separation with the Davies-Bouldin index (DB), plotted in fig. 4.7. As with many of the other metrics, the greatest change is seen early in the training, to then slowly converge in the later stages. Here, the base model reaches a score below 5, whereas the manual model is around 6.3. The takeaway is that the base model has slightly better separation between groups, compared to the manual model.

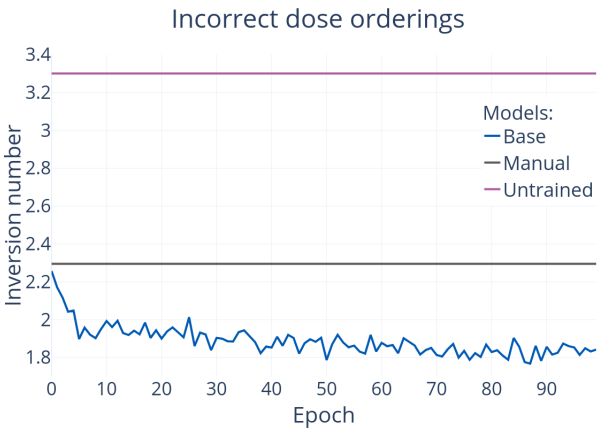


Figure 4.6: Comparison of the ability for different models to order the size of the dose by increasing distance from the control group. A low score is preferable, as it means that few inversions are needed to order doses correctly by level in the embedding space.

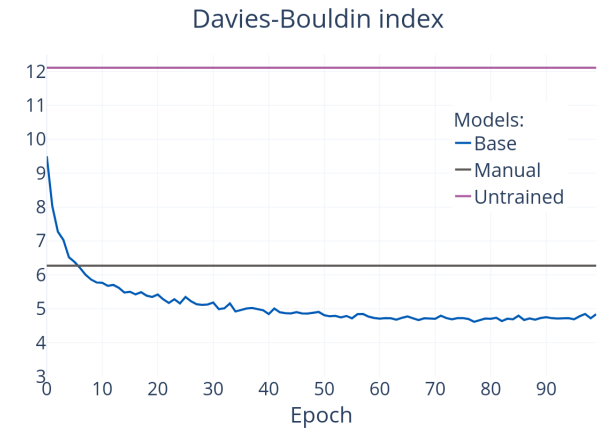


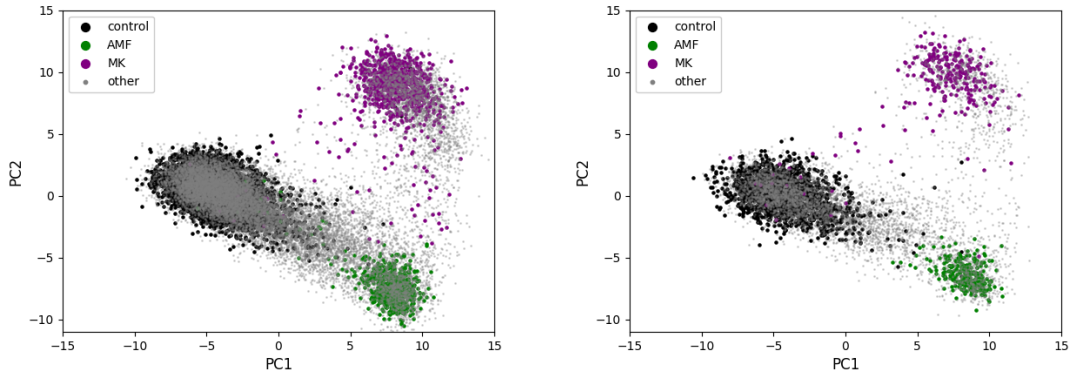
Figure 4.7: Comparison of between-group separation among different models. A low score indicates better separation, which is desirable.

Another evaluation metric is the F_1 score from the classifier trained on top of the model. As a reminder, the classifying task is to predict the correct indication every substance (or substances) is labelled with. These scores for the trained and untrained base model, as well as the manual model, are presented in table 4.1. As can be expected, the untrained model has the lowest F_1 score. More notably, however, the manual model creates representations that get a higher score than those of the base model.

We also visualize the output in two dimensions by projecting the embeddings onto the two first principal components in fig. 4.8, fig. 4.9 and fig. 4.10. Samples are coloured by four types of groups: Control, MK, AMF and ”other”. ”Other” refers to every point that does not belong to either control, MK or AMF. Note that points for ”other”-samples are smaller in the plots as to make the structure of control, MK and AMF more visible.

Table 4.1: Validation F_1 score for classification of substance indication. A high score is advantageous, since it indicates more informative embeddings concerning substance indication.

Model	Val F_1
Base	0.126
Manual	**0.201**
Untrained	0.10



(a) Training data

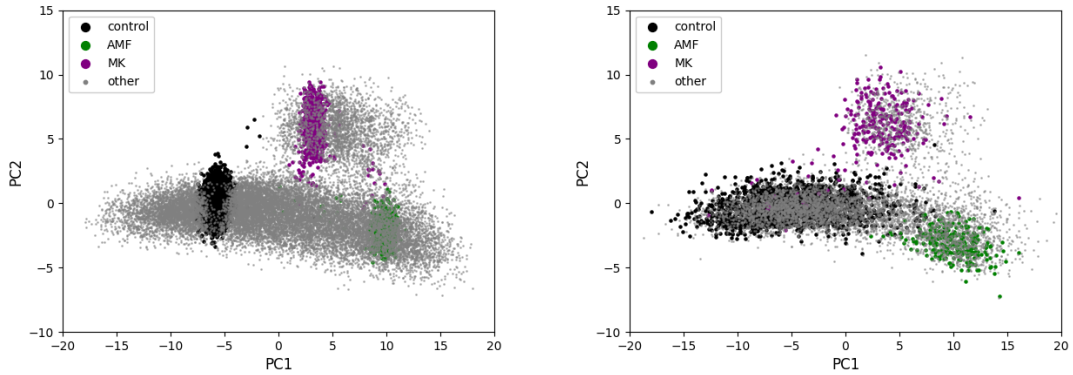
(b) Validation data

Figure 4.8: Embeddings in PC1 and PC2 for the best version base model.

The embeddings from the base model at its lowest validation loss are visualized in fig. 4.8. At this point, the model is not yet overfitted which explains the similar structure between embeddings from training and validation data. Further into training, the model overfits heavily. At this point, after 300 epochs, the embeddings are visualized in fig. 4.9. For the training data, the embeddings for control, MK and AMF lie very compact within each group, and well separated from each other. As expected, this is not the case for the validation data.

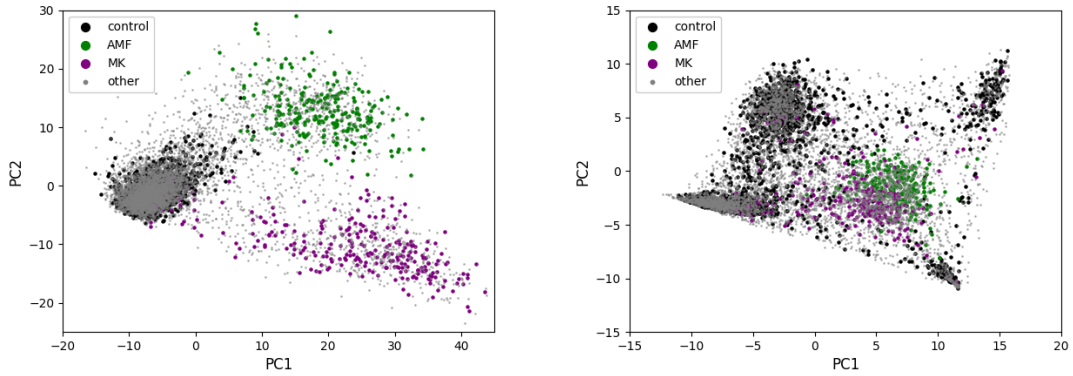
For reference, the validation data embeddings from the manual model and an untrained base model are visualized in fig. 4.10. The manual model manages to divide the three groups control, MK and AMF. Surprisingly, even for an untrained base model, the data seem to contain some structure with respect to these groups. Conversely, even though the manual model seems to find better behavioural features than an untrained base model, this is not represented in their respective loss, as the manual model has a higher loss than the untrained base model, see fig. 4.2.

The overall pattern of the different embeddings is that both the base model and manual model manage to separate the groups of control, MK and AMF from each other. However, accounting for the ”other” group which is spread out among the different groups, would make any visual separation impossible (at least in only two dimensions). Moreover, a desired property that no model accomplishes is to embed the control group in the centre to represent the standard behaviour of a rat, since there should be behavioural changes both increasing and decreasing different aspects



(a) Training data (b) Validation data

Figure 4.9: Embeddings in PC1 and PC2 for an overfitted base model.



(a) Manual model (b) Untrained base model

Figure 4.10: Embeddings in PC1 and PC2

of the normal behaviour. Nonetheless, the use of the SupCon loss during training does not support the achievement of this desired property. The control group is treated equally to the other groups and will be encouraged to have its samples embedded as far away as possible from all other groups.

4.1.2 Variance Analysis

When comparing different models with possibly similar behaviours, it is necessary to know how the evaluation metrics vary between runs with the same model. To investigate this in a simple manner, we trained the base model nine times with identical settings and compared the results. The mean and variance in evaluation metrics from these runs are found in table 4.2. Another model that achieves a metric within the $(\mu \pm 2\sigma)$-range should not be considered as significantly different from the base model, concerning that metric.

Table 4.2: Mean and standard deviation of evaluation metrics for nine runs with the base model.

	Mean (μ)	Standard deviation (σ)	($\mu \pm 2\sigma$)-range
Val loss	4.50	0.005	[4.49, 4.51]
Dispersion	0.213	0.004	[0.205, 0.221]
%PNN	23.40	0.252	[22.89, 23.90]
DR	1.856	0.030	[1.796, 1.915]
DB	4.770	0.068	[4.635, 4.906]
Val F_1	0.123	0.0095	[0.104, 0.142]

Table 4.3: Summary of the evaluation metrics for the base model and the manual model. Arrow specifies whether higher (↑) or lower scores (↓) are indicative of a better model.

Model	Val loss ↓	Dispersion ↓	%PNN ↑	DR ↓	DB ↓	Val F_1 ↑
Base	**4.50**	0.213	**23.4**	**1.86**	**4.77**	0.123
Manual	7.58	**0.202**	22.3	2.29	6.27	**0.201**
Untrained	5.47	0.521	19.7	3.30	12.1	0.10

4.1.3 Analysis

A summary of the evaluation metrics for different models is found in table 4.3, with the best values for each metric written with bold numbers. The arrows after the scores show if a higher or lower score indicates better model performance. Comparing the base model to the manual model, we observe a slight increase in performance in a majority of the evaluation metrics in favour of the base model. However, the model is overfitting after roughly 60 epochs. To prevent overfitting, we will make modifications by different measures, with the goal to increase overall performance. How the modifications are made and how they affect the results will be shown in the following sections.

One major advantage of the manual model is its known features. This allows to interpret the meaning of each axis when plotting the model's embeddings. Consequently, we can precisely identify the differences between different trajectories and gain insights into the specific behavioral effects of new, unknown substances. Unfortunately, such an in-depth analysis is not possible for the base model, as we lack knowledge about its explicit features. However, we can still compare the embeddings of the base model to those of the manual model visually. Such a comparison could offer some indication of the realistic meaning of the base model's features. For example, both models might generate embeddings that exhibit a very similar relationship with respect to a specific feature of the manual model. It is then likely that the corresponding feature of the base model captures a similar behavior. Nevertheless, conducting these types of analyses requires a qualitative approach and a significant amount of domain knowledge, which exceeds the scope of this project and has thus been omitted from this study.

Table 4.4: Evaluation metrics and the number of parameters from varying output size of the base model.

Output size	#Params	Val loss ↓	Dispersion ↓	%PNN ↑	DR ↓	DB ↓	Val F_1 ↑
10	7,356	4.53	**0.183**	21.3	2.00	6.03	0.075
30	56,196	4.51	0.209	23.1	1.89	4.97	0.103
70	259,244	4.51	0.230	23.3	1.94	4.81	0.121
100 (Base)	486,949	**4.50**	0.213	23.4	**1.86**	**4.77**	0.123
150	994,989	4.52	0.216	**23.5**	1.88	4.79	0.131
300	3,348,045	4.51	0.216	23.4	1.88	4.83	**0.157**

4.2 Model Architecture Variations

Since the base model exhibits overfitting behaviour and does not perform consistently better than the manual model, it would be of interest to investigate if modifications can be made to improve the performance. In this section, we present three different types of variations related to the model architecture. First, different model output sizes are examined. Second, a projection head is appended to the encoder, which is also tested with different output sizes. Finally, we also study the effect of adding batch normalization layers to the encoder.

4.2.1 Output Size

In this subsection, the output size of the encoder is varied. As the output size is also a limit on the number of channels in the hidden layers of the encoder, this is a simple way to modify the number of parameters in the model.

Method change
We trained the base model with different output sizes of the values $\{10, 30, 70, 100, 150, 300\}$. A most likely insignificant but necessary note is that some of the smaller models were trained using a higher learning rate to speed up the training, in order to reach its minimum validation loss in 100 epochs.

Results and Analysis
Decreasing the output size did affect the training and validation loss, but only in such a way that larger models reached their minimum validation loss quicker, and after starting to overfit. These graphs are not visualised, but the evaluation scores are found in table 4.4. The best values for different metrics are spread over models with different output sizes. Generally speaking, output size does not have a very large effect on the model and it is difficult to say whether the base model benefits from a large or small output size. These findings indicate that the number of parameters in the model is not the primary reason for overfitting and limitations of performance.

4.2.2 Projection Head

Given the prevalent use of projection heads in contrastive learning algorithms [1], [10], [38], an exploratory performance analysis would not be complete without in-

Table 4.5: The best score in each category from all projection head runs, together with the output dimensions used in that run. Base model scores are included for reference. The most notable performance difference comes from the F_1 classification metric, which reflects the projection head purpose of creating more informative embeddings.

	Val loss ↓	Dispersion ↓	%PNN ↑	DR ↓	DB ↓	Val F_1 ↑
Base score	**4.50**	0.213	**23.4**	1.86	**4.77**	0.123
Best score	4.55	**0.199**	23.3	**1.80**	5.25	**0.189**
$O_{enc.}$	300	1000	300	300	300	300
$O_{proj.}$	100	50	50	100	50	100

cluding them. As described in section 2.6.1, a projection head can be appended to the encoder, which has been shown to increase the latent embeddings' generalization performance. The loss is computed on the projected embeddings, while the latent embeddings are used for evaluation and downstream tasks.

Method Changes

For the choice of projection head, we choose to follow the example of Chen, Kornblith, Norouzi, *et al.* [1] and use an MLP with one hidden layer and a ReLU activation function, so that $h(z_i) = W^{(2)}\sigma(W^{(1)}z_i)$. The sizes of the layers are varied as well as the output size of the encoder, in order to examine if the size of the projection head affects the performance. The encoder and projection head is trained with four and three different output sizes, respectively; $O_{enc.} \in \{1000, 300, 100, 50\}$, and $O_{proj.} \in \{100, 50, 20\}$. Any output size combinations where $O_{proj.} \geq O_{enc.}$ are not tested, since the projection head is meant to reduce the embeddings' dimensions. The size of the first projection head layer is set to $\lceil (O_{enc.} + O_{proj.})/2 \rceil$.

Results and Analysis

As can be seen in table 4.5, including a projection head in the model does not seem to improve the overall performance. It is important to note that the best evaluation scores presented come from different runs, so the performance of a single model with a projection head would be even lower. The improved F_1 score is notable however, being markedly better than the base model score. This performance increase is expected, since projection heads are used in contrastive learning to increase the latent embeddings generalization performance. This increased performance can be observed in downstream tasks such as classification using the embeddings.

However, considering *how* the latent representations are made more informative, a projection head might not be suited for our purposes of grouping similar behaviours. Because of the loss function, the variables on which the loss is computed are trained to be invariant to any augmentations. While we do not use any augmentations in the conventional sense in the base model, different trajectories with the same substance and dose can be thought of as a form of augmentation. This invariance to augmentations arises since positive samples are encouraged to be represented in the same way, even if their style differs as a result of augmentations. In the base model case, the difference in style comes from the unique movement patterns of the

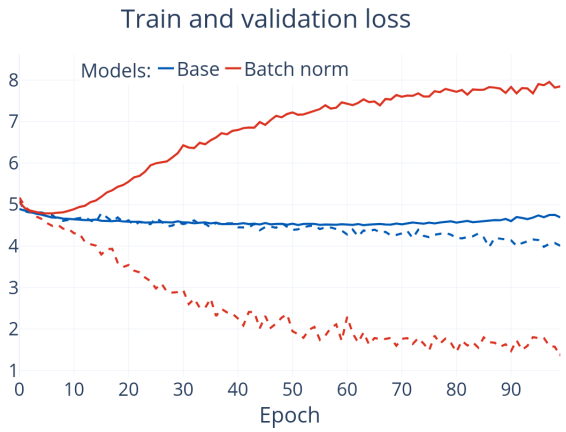


Figure 4.11: Training and validation loss the base model with and without batch norm. Adding batch normalization does not reduce overfitting.

rats. Including a projection head enables the latent representations z to still contain style-specific information coming from the augmentations. This is usually desired for other contrastive learning algorithms, since the embeddings then contain more generalizable information. However, we want trajectories with the same behavioural change to be mapped to the same point, regardless of any augmentations or individual rat "personalities". Therefore, the commonly observed effects of a projection head might not even be desired for this specific problem setting. Since the model exhibits the same amount of overfitting behaviour, there does not seem to be any other reasons for including a projection head either.

4.2.3 Batch Normalization

Batch normalization, discussed in section 2.4.4, is mainly used to speed up training of neural networks, and has also shown a regularizing effect. As batch normalization is also applied in the ResNet architecture proposed by He, Zhang, Ren, *et al.* [11], we examined the effect of adding it to the base model.

Method Change
When batch normalization is used in our encoder, it is applied after every convolutional layer, but before activation, following the standard by He, Zhang, Ren, *et al.* [11]. This results in 30 total batch normalization layers for our encoder.

Results and Analysis
The training and validation loss after training 100 epochs are found in fig. 4.11, where it is compared to the base model. Using batch norm speeds up training as the theory suggests. However, it does not seem to have the regularizing effect as anticipated, as the model overfits even faster. One reason for this could be that the steeper learning curve and thereby faster overfitting out-weights any regularizing effects that the batch normalization layers might have on the model. Moreover, the minimum validation loss as well as the other evaluation metrics are found in table 4.6. Judging from these metrics, it is clear that batch normalization applied in this way does not increase the performance of the model.

Table 4.6: Evaluation metrics with and without batch normalization (BN).

BN	Val loss ↓	Dispersion ↓	%PNN ↑	DR ↓	DB ↓	Val F_1 ↑
No (Base)	**4.50**	**0.213**	**23.4**	**1.86**	**4.77**	**0.123**
Yes	4.79	0.254	21.0	2.01	5.11	0.104

4.3 Data-related Modifications

In this section, we present and analyse the results of data-related modifications. First, we examine the effect of larger batch size, then move on to adding data augmentations. For the last part, the coordinate representation of the data is changed from Cartesian coordinates to egocentric.

4.3.1 Batch Size

A common aspect in contrastive learning applications is the use of large batch sizes in order to increase the number of negatives for each sample. This can make the contrastive modelling task more difficult and can lead to improved performance.

Method Change
To investigate the impact of batch size we trained four different models with different batch sizes $N \in \{32, 64, 128, 256\}$. Because of our sampling technique, where for each data point a positive is sampled, this means that the model is trained on $2N \in \{64, 128, 256, 512\}$ samples per batch.

Results
Changing batch size did not have an effect on the overfitting behaviour, as the training and validation loss for different batch sizes all follow the same tendency. The validation loss is lower when using a lower batch size, but this is a consequence of the loss function increasing with more negatives per batch. Therefore, the validation loss in table 4.7 is omitted. Overall, the changes in the other evaluation metrics are relatively small, indicating that models with different batch sizes learn similar embedding. There is however one exception, which is the validation F_1 score from the classification task. Interestingly, the base model has the worst performance on this task and all other models are outperforming with a score above the $\mu + 2\sigma$-threshold. This result is unexpected, since both lower and higher batch sizes seem to create more informative representations. While there might be some logical explanation, the apparent performance decrease of $N = 128$ could still be spurious, considering the small number of runs and parameter values tested. The $N = 128$ run also performs well in regard to the other evaluation metrics, being somewhat contradictory to the F_1 score. In summary, increasing batch size does not seem to increase the performance as anticipated. These findings suggest that the model does not require additional negatives for each sample in the batch to improve its feature learning. It is likely that the existing negatives pose a significant challenge for the model in differentiating between samples and negatives.

Table 4.7: Summary of the evaluation metrics when changing the batch size. It is not clear whether a large or small batch size improves performance.

Batch size	Dispersion ↓	%PNN ↑	DR ↓	DB ↓	Val F_1 ↑
32	**0.189**	23.1	1.94	4.88	0.152
64	0.203	23.2	1.90	**4.67**	0.159
128 (Base)	0.213	**23.4**	**1.86**	4.77	0.123
256	0.203	22.9	1.88	4.68	**0.162**

Table 4.8: Evaluation metrics of the base model with added augmentations.

Augmentations	Val loss ↓	Dispersion ↓	%PNN ↑	DR ↓	DB ↓	Val F_1 ↑
None	**4.50**	0.213	**23.4**	1.86	**4.77**	0.123
Flip+rotate	4.52	**0.210**	23.0	**1.85**	4.85	**0.177**

4.3.2 Data Augmentation

As described in section 2.4.2, data augmentation techniques are used to reduce overfitting. These techniques increase the diversity and variability of the training data, allowing the model to generalize better to unseen examples.

Method Change
The augmentation techniques we have considered are inspired by image analysis, as they are applied in the (x, y)-coordinates of the trajectories. We augment the data by transformations consisting of rotations and flips, in total eight different transformations. Four of those are rotations by 0, 90, 180 or 270 degrees in the (x, y)-domain, and the other four consist of flips in four different axes: x, y, $y = x$ and $y = -x$.

The transformations are applied exclusively and randomly, meaning that each trajectory has an equal probability of being transformed in one of the eight ways before passing it onto the encoder during training. Therefore, using these augmentations during training will effectively increase the training dataset by a factor of eight (assuming that the model is trained for sufficiently many epochs). For validation and evaluation, the augmentations are turned off.

Results and Analysis
The training and validation loss for the base model, with and without augmentations are visualized in fig. 4.12. We note that the losses of the two models behave very similarly, reaching the same minimum validation loss at 4.5 at around 60 epochs. Afterwards, the model with augmentations seems to overfit at a slightly slower rate compared to the base model.

Moreover, the evaluation metrics are presented in table 4.8. The different models have approximately the same scores, except for validation F_1 score in the classification of substance indication. Augmentations raise the F_1 score significantly, implying that these representations are more informative with respect to substance indication rather than the representations obtained from the base model.

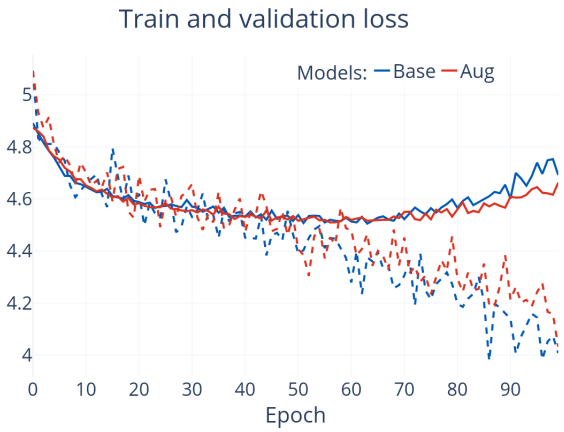


Figure 4.12: Training and validation loss for the base model with and without augmentations. The curves follow a very similar trend and augmentation does not seem to reduce overfitting.

Despite augmenting with flips and rotations, effectively increasing the amount of training data by a factor of eight, this technique had minimal impact on training and overfitting, resulting in similar performance. These results suggest that the model can already discern the similarity between a sample and its rotated or flipped counterpart, indicating a solid understanding of this concept. While this demonstrates the model's proficiency, it also implies that these techniques are relatively simplistic and do not challenge the model enough to enhance performance.

4.3.3 Coordinate Representation

As explained in section 3.1, the dataset contains Cartesian (x, y, z)-coordinates representing position over time for each trajectory. This kind of coordinate representation is useful for behavioural aspects related to where the rat is located in the cage. For example, more time spent in the middle of the cage is a sign of higher risk tolerance, which can be used to compare behavioural changes of different substances. Other kinds of behaviour can be observed by instead examining relative movement, such as running in circles or often switching directions. These kinds of features can of course also be extracted when using Cartesian coordinates, but could be worth emphasising by a change of coordinate representation. The modification presented below concerns *egocentric coordinates*, representing the data from the rat's perspective.

Method Change

instead of (x_i, y_i)-coordinates for each time frame i, the data is transformed to include d_i, the distance traveled since the previous time frame, and θ_i, representing how much the rat has switched direction. Let

$$\nu_i = \begin{cases} 0 \quad \text{if } y_i - y_{i-1} = x_i - x_{i-1} = 0, \\ \arctan2(y_i - y_{i-1}, x_i - x_{i-1}) \quad \text{otherwise.} \end{cases} \tag{4.1}$$

Similar to arctan but unambiguous, $\arctan2(y, x)$ gives the angle α (in radians, with $-\pi < \alpha \leq \pi$) between the positive x-axis and the vector represented by

(x, y). Therefore, ν_i gives a non-relative angle between the positional change and the positive x-axis. The new egocentric coordinates are defined as

$$d_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}, \tag{4.2}$$

$$\theta_i = \begin{cases} \nu_i - \nu_j + 2\pi, & \text{if } \nu_i - \nu_j \leq -\pi \\ \nu_i - \nu_j, & \text{if } -\pi < \nu_i - \nu_j \leq \pi \\ \nu_i - \nu_j - 2\pi, & \text{if } \pi < \nu_i - \nu_j, \end{cases} \tag{4.3}$$

where $j < i$ is the most recent index where $d \neq 0$. Thus, $\theta_i \in (-\pi, \pi]$ is the relative angle between the last two position changes. The case-dependent addition or subtraction of 2π is made to avoid ambiguity coming from the periodicity of angles. d_0 is set to zero, and ν_{-1} is set to π as an arbitrary non-zero starting value. The binary z-coordinate, which indicates whether the rat is rearing or down on all four paws, remains unchanged. In order to avoid large angle changes caused by sensor inaccuracies and the granularity of the data, a running average of the (x, y)-coordinates are used for computing d and θ, where three consecutive positions are averaged.

The use of egocentric coordinates also changes the possibility for augmentations. For example, the rotation transformation described in section 4.3.2 becomes irrelevant since the relative movements of the rat remain the same. New types of augmentations also become viable, like modifying the distances d with some multiplicative Gaussian noise factor. The second type of modification presented in this subsection regards such new augmentations. Distances d_i are multiplied with a noise term $\eta_i \sim \mathcal{N}(1, \sqrt{0.25})$. The direction changes θ_i are also multiplied with a random variable $\tau_i \in \{-1, 1\}$, $p(\tau_i = 1) = 0.5$, so that there is a 50% change that the rat turns in the opposite direction compared to the original movement.

Results

The training and validation loss is visualized in fig. 4.13. Both of the egocentric models reach a slightly lower minimum validation loss in fewer training epochs compared to the base model. The lowest validation loss is achieved with the egocentric coordinate representation together with augmentations.

Analysis

As seen in table 4.9, using egocentric coordinates seems to increase the overall performance of the base model. This could indicate that coordinate representations placing more emphasis on relative movement make it easier for the model to find generalizable features. However, the positive change is marginal, and the overall problem with overfitting persists.

When including the augmentations described earlier, the model exhibits further improvement. Slightly varying the distances travelled between time steps and randomly mirroring turns therefore seem to better isolate some of the relevant and generalizable movement patterns. However, these generalizable patterns made more accessible by the augmentations might come at the cost of other useful information. If an augmented trajectory would be transformed back to Cartesian coordinates, the rat would be moving outside of the cage as a result of the modified distances

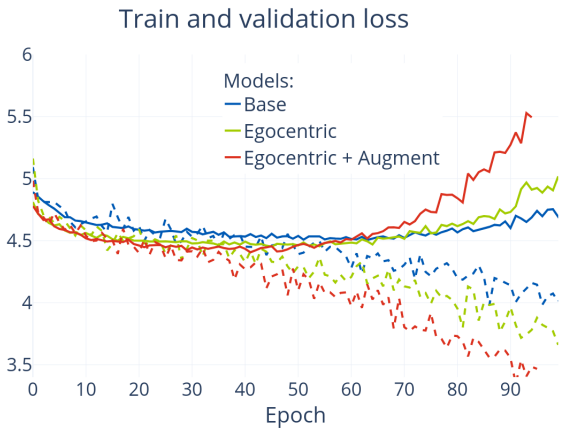


Figure 4.13: Training and validation loss for the base model, egocentric model, and augmented egocentric model. The egocentric model has a slightly lower validation loss minimum than the base model, and starts to overfit after roughly 40 epochs.

Table 4.9: Comparison of evaluation metrics for Cartesian and egocentric coordinate representations. Results from using the egocentric augmentations are also included.

Coordinate repr.	Val loss ↓	Dispersion ↓	%PNN ↑	DR ↓	DB ↓	Val F_1 ↑
Cartesian (Base)	4.50	0.213	23.4	1.86	4.77	0.123
Egocentric	4.45	0.223	24.0	**1.71**	4.56	0.154
Egocentric + Aug	**4.41**	**0.206**	**24.7**	1.76	**4.51**	**0.179**

and mirrored direction changes. Any information regarding the rat's position in the cage is thereby lost, and behaviours such as running in circles are also obfuscated. This information trade-off and the fact that the observed improvements are small highlight the difficulty of designing augmentations that cover a lot of the style space, while keeping the content unchanged.

4.4 Loss Modifications

Yet another way to modify the base model is with variations in the loss functions. We will examine the effect of changing the current loss in different ways, and lastly change the entire function to Triplet margin loss.

4.4.1 Weight Decay

A common regularization technique is to apply L_2-regularization during training, also known as weight decay, see section 2.4.3. This technique imposes lower values of the squared norm of the model weights, which can help to restrain the model from overfitting.

Method Change

An attempt to reduce overfitting was performed by training the base model with five different weight decay settings: $\lambda \in \{0, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$, where $\lambda = 0$ is

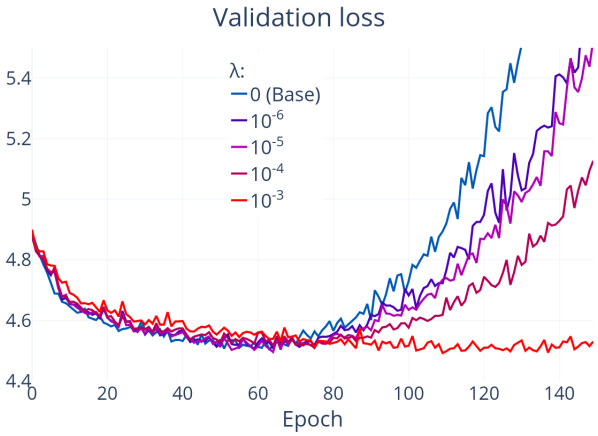


Figure 4.14: Validation loss from training 150 epochs with different weight decay parameters λ. Increasing weight decay slows down the decrease in validation loss but does not seem to reduce minimum validation loss.

Table 4.10: Evaluation metrics from applying weight decay to different extents.

λ	Val loss ↓	Dispersion ↓	%PNN ↑	DR ↓	DB ↓	Val F_1 ↑
0 (Base)	4.50	0.213	**23.4**	1.86	4.77	0.123
10^{-6}	4.51	0.212	22.9	1.95	4.57	0.148
10^{-5}	4.50	0.213	23.3	**1.85**	**4.53**	0.176
10^{-4}	4.51	0.220	23.2	1.87	4.56	0.157
10^{-3}	**4.49**	**0.201**	23.2	**1.85**	4.54	**0.178**

equivalent to using the base model. The models were trained with the same learning rate of 0.0001 but for 150 epochs as to reach their minimum validation loss during the training.

Results and Analysis

The validation loss of the different runs are found in fig. 4.14. Increasing the value of λ expectedly slows down the learning such that validation loss decreases more slowly. However, regardless of λ, all the models achieve approximately the same minimum validation loss.

Based on the evaluation metrics presented in Table 4.10, weight decay demonstrates a potentially positive effect on performance. When comparing the scores obtained using $\lambda = 10^{-3}$ against table 4.2, we see that three of the scores (Dispersion, DB, F_1) are outside of the $(\mu \pm 2\sigma)$-ranges of the base model. This indicates an actual improvement rather than random fluctuations. However, the improvement is quite marginal and the model still exhibits a similar overfitting behaviour. The lack of a clear trend when increasing λ also makes it harder to draw any firm conclusions. Therefore, a more extensive investigation with more runs and parameter values would be needed to determine the full potential of weight decay.

Table 4.11: Evaluation metrics from changing the temperature parameter τ in the SupCon loss.

Temperature τ	Val loss ↓	Dispersion ↓	%PNN ↑	DR ↓	DB ↓	Val F_1 ↑
0.01	4.70	0.257	22.5	2.02	5.10	0.092
0.1	4.51	0.238	23.4	1.92	4.76	0.107
1 (Base)	4.50	**0.213**	23.4	1.86	4.77	0.123
10	**4.47**	0.223	**23.7**	1.84	**4.63**	0.114
100	4.49	0.234	23.5	**1.81**	4.81	**0.159**

4.4.2 Temperature

The temperature parameter τ is part of the SupCon loss, which scales the similarities between samples. By adjusting the value of τ, one can effectively control the magnitude of the differences between similar and dissimilar pairs. A higher value of τ reduces the disparity between these pairs, encouraging the model to focus on capturing subtle and intricate similarities However, a lower value of τ amplify the distinction between similar and dissimilar pairs, promoting the recognition of broader patterns and relationships within the data.

Method Change
In order to study the effect of τ, we trained five models with different values for τ. We let τ vary on a logarithmic range from 0.01 to 100, including 1 which is used in the base model.

Results and Analysis
The evaluation metrics obtained from training with different values of τ are presented in table 4.11. The results indicate that using low values of τ leads to a decrease in performance across multiple scores. Conversely, for larger values of τ, the optimal values for each score are distributed among different choices of τ. However, these changes in performance are relatively small, and increasing τ above 1 does not appear to have a substantial impact on performance.

By adjusting the value of τ, the model's attention can be tailored to focus on either local or global patterns within the data. Despite the minor variations in performance, it is unlikely that the model's performance can be greatly increased by adjusting the attention to either small or large similarities. The consistent performance across different values of τ suggests that the model's ability to capture relevant similarities is not significantly affected by the choice of temperature parameter.

4.4.3 Semi-positives

Every substance examined has been tested in different doses. For the base model, only trajectories with the same substance/dose combination are considered positive, but it is reasonable to believe that the same substance in slightly different doses should also yield a somewhat similar behavioural response. Below, we propose another relation type called *semi-positives*, in order to view some of the relations as more of a spectrum rather than binary.

Table 4.12: Evaluation scores from the base model with and without the semi-positive relations in the loss function.

Loss variation	Val loss ↓	Dispersion ↓	%PNN ↑	DR ↓	DB ↓	Val F_1 ↑
SupCon (Base)	**4.50**	**0.213**	23.4	1.86	**4.77**	0.123
Semi-positives	4.54	0.222	**23.7**	**1.81**	4.80	**0.173**

Method Change

For a sample z_i, let $P(i)$ be the set of indices of all other samples with the same substance as z_i. If a secondary substance is also involved, all samples with indices in $P(i)$ must have the same secondary substance and dose as z_i. This is not a very limiting requirement, since the vast majority of secondary substances are only administered in one common dose, as described in section 3.1. The only difference compared to how $P(i)$ was previously defined is that the primary dose is allowed to differ between samples. The SupCon loss can then be modified with a weight term w_{ip} for the positive pairs:

$$L = \sum_{i \in I} L_i = \sum_{i \in I} -\log \left[\frac{1}{|P(i)|} \sum_{p \in P(i)} \frac{w_{ip} \cdot \exp\left(\text{sim}(z_i, z_p)/\tau\right)}{\sum_{j \in I \setminus \{i\}} \exp\left(\text{sim}(z_i, z_j)/\tau\right)} \right]. \tag{4.4}$$

where

$$w_{ip} = \begin{cases} 1, & \text{if } d_i = d_p = 0 \\ 1 - \frac{|d_i - d_p|}{d_i + d_p} & \text{otherwise.} \end{cases} \tag{4.5}$$

The weight $w_{ip} \in [0, 1]$ depends on the primary doses d_i and d_p for samples with indices $i \in I$ and $p \in P(i)$. More similar primary doses will result in w_{ip} being closer to 1. With this modification, the relation for samples with identical substances and doses remains unchanged as $w_{ip} = 1$ when $d_i = d_p$.

Results and Analysis

The evaluation metrics presented in table 4.12 show little difference in performance, with an exception for the F_1 score. The inclusion of semi-positive relations appears to enhance the information related to the indication label, resulting in an increased F_1 score. This outcome could be attributed to the fact that the semi-positive modification encourages the model to consider larger groups of trajectories as partly similar, which is more in line with the indication label since the clinical field of use is dose-independent. The dispersion (ga-WSS-TSS) and Davies-Bouldin metrics are not modified, and thus use the old relation definition. It is therefore not surprising that the modified model performs slightly worse according to these metrics, as the semi-positive relations likely cause more dispersion between same-substance groups. While the loss is slightly higher for the modified model, a comparison might not be viable since two different loss functions are technically used.

Overall, the semi-positive inclusion in the SupCon loss does not significantly reduce overfitting or distinctly improve general model performance. Merely changing the positive same-substance relations from binary to continuous seems to be insufficient for the model to clearly learn better embeddings. However, this lack of improvement

does not necessarily mean that the problem lies elsewhere. Since the sampling method described in section 3.3.2 is unchanged, the positive pairs used to form the batches are still strictly positive. Any semi-positive pairs must therefore occur by chance, which is not very likely considering the large number of unique substances in the dataset. The absence of notable change in performance could thus be an effect of the small actual change made to the model. Furthermore, considering that all of the evaluation scores that still allow for fair comparison show a marginal yet positive performance increase, the relation definition could deserve a more thorough examination. For example, changing the negative relations would likely have a much larger impact on the model. While no such negative-relation modifications are made in this thesis, they are further discussed in chapter 5.

4.4.4 Similarity Measures

In previous applications using NT-Xent and SupCon loss, cosine similarity has been used as opposed to negative Euclidean in our base model. With this in mind, as well as considering the curse of high dimensions, we explored how different similarity measures affected model performance.

Method Change
The base model was run with three different similarity measures in the SupCon loss, namely negative Euclidean distance, negative Manhattan distance, and cosine similarity. In order to avoid numerical instability, the cosine similarity was changed to

$$\frac{z_i \bullet z_j}{||z_i||_2||z_j||_2 + \epsilon}. \tag{4.6}$$

where $z_i, z_j \in \mathbb{R}^n$ as usual and $\epsilon = 10^{-9}$.

Results and Analysis
When training the model using Manhattan distance, the loss followed a very similar trend to the base model. However, when employing cosine similarity, the training loss showed minimal improvement, and the validation loss reached a minimum of 5.16 before increasing. These findings suggest that the model utilizing cosine similarity did not learn highly informative representations, which is supported by the evaluation metrics presented in table 4.13.

In contrast, Manhattan distance performed on par with the base model and exhibited better validation F_1 score for substance classification, indicating that the embeddings derived from this metric are more informative in terms of substance indication. Though, since the other metrics for Euclidean and Manhattan distances are highly similar, it is unlikely that the embeddings differ significantly.

To summarize, cosine similarity does not seem to be suitable for this task. The magnitude of the embeddings, which may contain information about the strength of the behavioural response, is likely important in creating good representations. The comparable performance of the other distance-based similarity metrics suggests that the model's performance is not significantly hindered by the choice of similarity metric. Given the minimal performance difference between Manhattan and Euclidean

Table 4.13: Evaluation metrics from using different similarity measures.

Similarity	Val loss ↓	Dispersion ↓	%PNN ↑	DR ↓	DB ↓	Val F_1 ↑
Neg Euclidean (Base)	**4.50**	**0.213**	23.4	**1.86**	**4.77**	0.123
Neg Manhattan	**4.50**	**0.213**	**23.7**	**1.86**	4.81	**0.182**
Cosine similarity	5.16	0.490	21.8	2.45	105	0.065

distances, it is unlikely that using a high dimension of 100 for the latent space poses a problem during model training. This conclusion is based on Manhattan distance being less sensitive than Euclidean to the curse of dimensionality.

4.4.5 Triplet Margin Loss

Instead of modifying the current SupCon loss, it is also interesting to see how the model reacts to a different loss function. The number of different contrastive loss functions that could be applied is many, and we chose to explore the triplet margin loss.

Method Change
Triplet margin loss is introduced in section 3.3 and is defined as

$$\mathcal{L} = \sum_{a \in I} \max\left(0, ||z_a - z_p||_2 - ||z_a - z_n||_2 + m\right). \tag{4.7}$$

The parameter $m \in \mathbb{R}^+$ is the margin for the difference in distances $||z_a - z_p||_2$ and $||z_a - z_n||_2$, and the model was trained with three different margins $m \in \{0.1, 1, 10\}$.

Given an anchor z_a, we chose the hardest possible positive z_p within the batch, meaning the index $p = \arg\max_{i \in P(a)} ||z_a - z_i||$, giving the positive sample furthest away from the anchor. For choosing the negative z_n, we first define the set of negatives $N(a) = I \setminus (P(a) \cup \{a\})$. Further, we tried three different sampling rules for z_n: Random negative, random hard negative, and hardest negative. The random negative rule is simply picking $n \in N(a)$ randomly. Choosing the hardest negative is performed by picking n such that $n = \arg\min_{i \in N(a)} ||z_a - z_i||$. Analogously to how the positive sample was chosen, this gives the negative sample that is the closest to the anchor. Lastly, picking a random hard negative is done by choosing a random n such that $n \in \{i \in N(a) : ||z_a - z_i|| - ||z_a - z_p|| < m\}$. Intuitively, this gives a negative which distance from the anchor is at most a margin value greater than the distance from the anchor to the positive. Choosing z_n in this way provides a z_n that is hard enough for it to contribute to a nonzero loss.

Training Results
The training and validation losses from running triplet margin loss with m set to 1 with varying negative sampling techniques are illustrated in fig. 4.15. Using the hardest negative for each anchor seems to make the loss stuck at the margin value. This indicates a collapsed model, as the lowest possible loss is achieved by mapping the positive and negative to the same point as the anchor. Relaxing the choice of z_n to samples of all hard negatives, or samples of all possible negatives, makes the model learn from the data during training. It looks from the figure that random hard

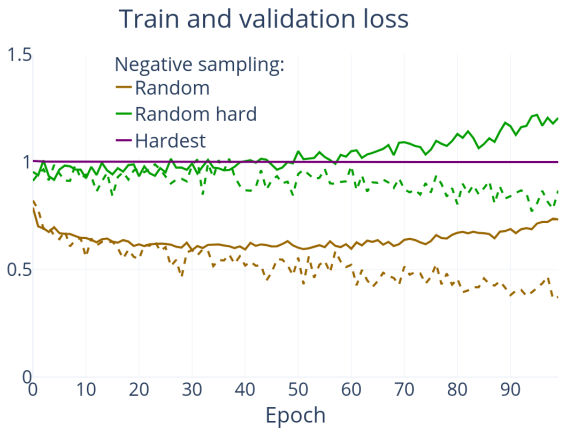


Figure 4.15: Triplet margin loss for different sampling techniques for choosing z_n.

negative sampling overfits very fast during training. However, it is difficult to make this conclusion based on the validation loss, as the triplets used in the calculation of the loss vary between epochs. Additionally, the number of triplets used is also a very small fraction of the total possible triplets within each batch. This further explains that the validation loss is not representative of the whole dataset.

Another consequence of the non-representative validation loss is the difficult task to in decide when each model is at its best stage during training. The best version of each model does not necessarily align with the epoch for its lowest validation loss. Therefore, stopping the model at its lowest validation is not a valid rule for picking the best model.

Evaluation Metrics

As it is difficult to choose the best possible model version for evaluation when using triplet loss, we also show how the evaluation metrics change during training in fig. 4.16. Here, three different models with margin $m = 1$ are trained with different rules for selecting the negative z_n used in the triplet loss. These are compared against each other, as well as against the base model. Remember that the base model uses the SupCon loss, and does not have a sampling rule or a margin. The figures demonstrate that the triplet loss models sampling negatives by random or random hard negative increases their performance with more training. This applies to every metric except dispersion (group average WSS-TSS). Once again, we demonstrate that stopping models using triplet loss at their lowest validation loss is not a good rule for picking the best model. Another remark is that the model using only the hardest negative perform very poorly. However, the overall takeaway from these figures is that none of the models using triplet loss seem to perform better than the base model, using SupCon loss.

Lastly, we do not show any results from the classification evaluation because it is unclear when the best version of each model occurs during training. If the best version can not be chosen concisely, then obviously classifying indications with the best version can not be performed.

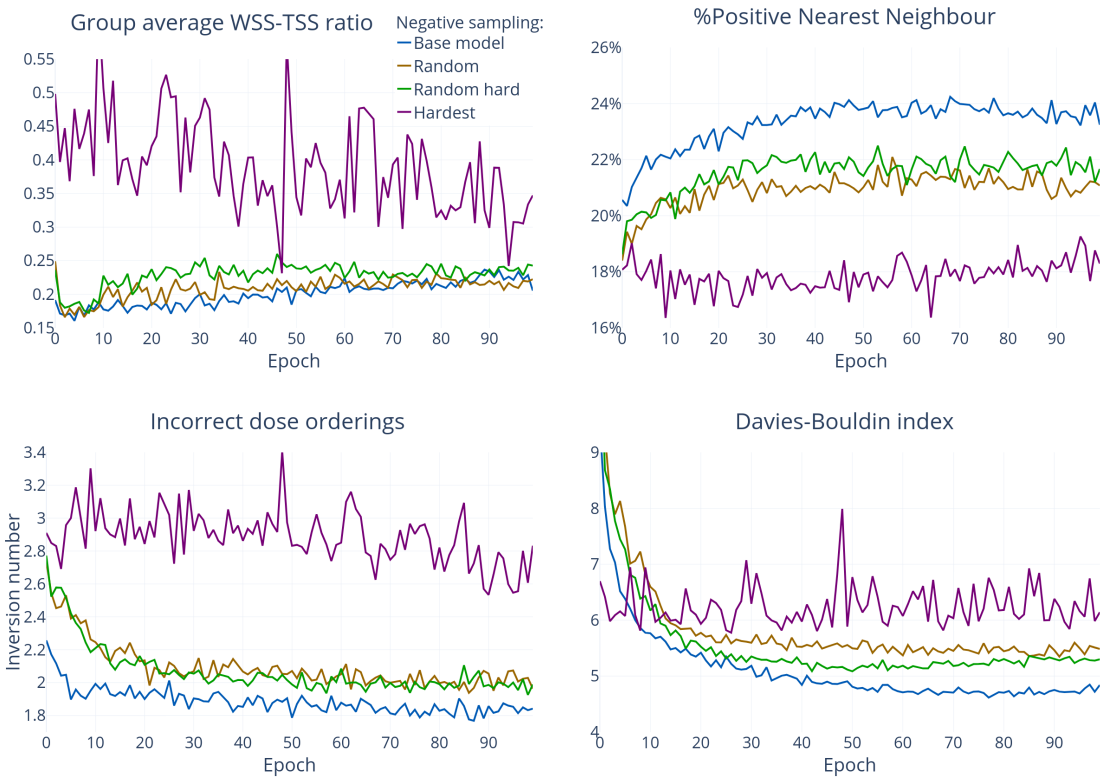


Figure 4.16: Evaluation metrics for models using triplet loss with different negative sampling techniques, compared to the base model using SupCon loss. The legend involves all four plots. Using the hardest possible negative yields the worst performance on every metric. The other sampling techniques perform better but still worse overall than the base model.

Analysis

Swapping SupCon loss for triplet margin loss did not result in any performance improvement. While this does not provide a clear indication of whether SupCon loss is suitable for the task, it does suggest that triplet margin loss is not a good choice, at least based on the options explored in this study.

Although the triplet loss did not enhance the model's performance, it did shed light on an interesting aspect related to negative samples. Triplet loss is commonly trained using negative mining techniques to increase the difficulty of the contrastive task and promote better learning. However, in this specific problem, using the hardest possible negatives led to significantly poor model performance, and there was no noticeable performance difference between using randomly selected negatives and randomly selected hard negatives.

These findings suggest that the negatives naturally present in the batch are already challenging enough for the model to learn from, and selecting the hardest negatives makes the task overly difficult. Instead of increasing the difficulty of the problem, these observations point toward the need to explore approaches that make the problem easier in some way.

Table 4.14: Evaluation metrics for base, manual and combined model

Model	Dispersion ↓	%PNN ↑	DR ↓	DB ↓	Val F_1 ↑
Base	0.213	23.4	**1.86**	**4.77**	0.123
Combined	**0.177**	**23.6**	2.07	5.44	0.171
Manual	0.202	22.3	2.29	6.27	**0.201**

4.5 Combined Model

Previously, in section 4.1, we focused on comparing the performance of the base model embeddings with the manual embeddings. However, we did not investigate whether these embeddings possess distinct information or capture different features. If the embeddings from the two models exhibit notable differences, leveraging both sets of embeddings could potentially yield better results. Moreover, if the combined embeddings contain more information than their individual counterparts, it suggests that the constituent parts indeed capture distinct features. This background motivates the analysis of a combined model, which this section will be dedicated to.

Method Change
The combined model involves integrating the embeddings from both the base and manual models. To accomplish this, the base model is trained in the usual manner, and after each epoch, the embeddings of each sample are saved. These saved embeddings are then normalized and concatenated with the normalized embeddings from the manual model, where the normalization ensures that each feature is on the same scale. The evaluation process proceeds as normal, with the concatenated embeddings being projected to 10 dimensions using PCA and assessed with our different evaluation metrics.

Results and Analysis
The evaluation metrics over time for the base, manual, and combined models are visualized in fig. 4.17. Remarkably, the combined model demonstrates a significant improvement in the dispersion (group average WSS-TSS) score compared to the individual models. This suggests that the combination of embeddings yields a more favourable outcome than considering them separately. However, for the remaining scores, the combined model achieves scores that lie between those of its constituent parts. This implies that the different embeddings do not complement each other effectively according to these metrics.

Additionally, we extract the embeddings from the best-performing version of the base model obtained during training. These embeddings are then incorporated into the creation of the combined model embeddings, which are subsequently employed for classifying substance indication. The validation F_1 score from the classification and other evaluation metrics are presented in table 4.14. Consistent with our earlier findings, these scores reinforce the notion that the combined model's performance falls between that of the individual models, indicating a lack of complementary features in the different embeddings. Notably, it is worth mentioning the dispersion score, where the combined model showcases an exception to this pattern.

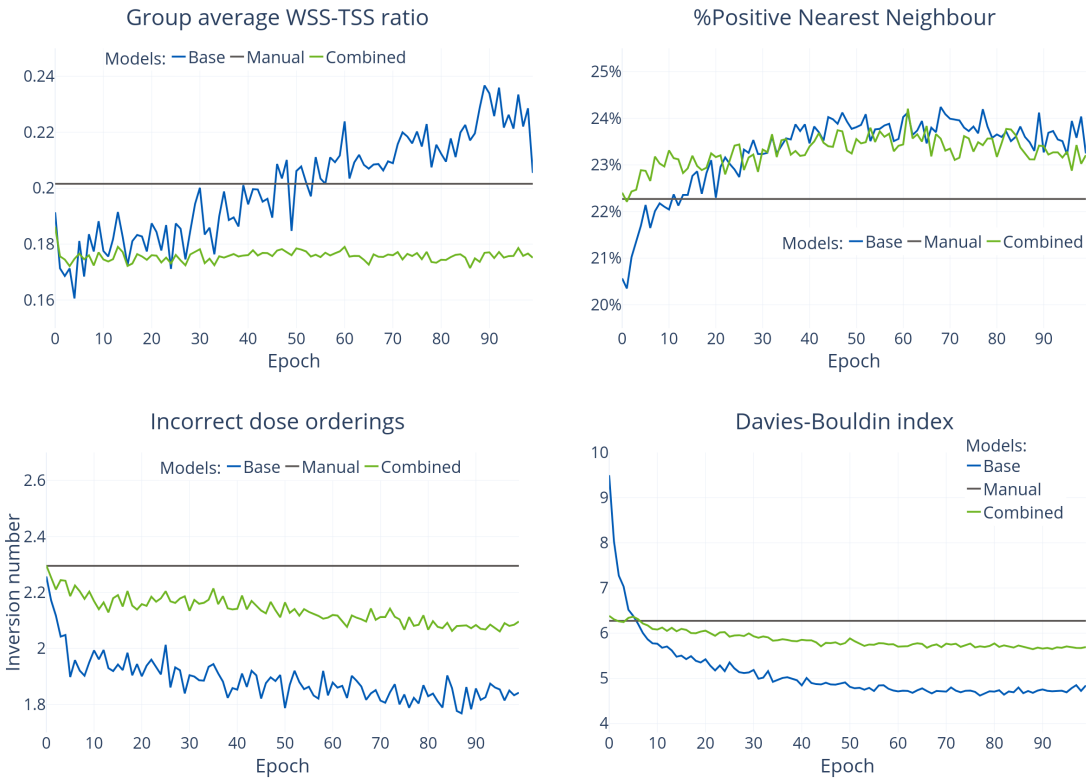


Figure 4.17: Evaluation metrics for the base and manual model, as well as the combination of the two. For a majority of metrics, the combined model takes values between the base and manual model implying that the different types of embeddings do not complement each other particularly well.

5

Discussion

The results presented and analyzed in chapter 4 consistently show that while the contrastive learning models perform on par with the manual model, they are generally insensitive to any modifications intended to increase performance. This lack of improvement is especially noteworthy since many of the examined modifications have previously been associated with performance increase for CL models, as described in section 2.6. For example, the augmentations presented in section 4.3.2 practically makes the dataset eight times larger, but as the results show, there is no significant change in performance. The modifications examining different hyperparameter values, such as output size in section 4.2.1, batch size in section 4.3.1, or temperature in section 4.4.2 do not only further demonstrate a lack of improvement, but also indicate that model performance does not decrease easily either. Based on these findings, we can conclude that the model is quite robust to various changes in the data as well as the model components. There are several possible reasons for this behaviour, which will be further explored in the following sections.

5.1 Problematic Fundamentals of the Dataset

As with all machine learning problems, the results are limited by the quality of the data. The dataset used in this project has grown in size over the span of more than 20 years, now being an extensive and unique source of behavioural data for drug development. There are however different aspects of the data that might be problematic from a machine learning standpoint. One such aspect is the long time period during which the data has been collected. Even though the standardized experiments are conducted using the same procedure as in the beginning, different people have handled the experiments, and different strains of laboratory rats have been used. A statistical analysis would need to be conducted in order to assess whether the data varies based on when it was collected, which has been outside the scope of this thesis.

Another aspect of the data that should be taken into consideration is that different substances will not always yield different behavioural responses. Two derivatives of the same substance might yield the same behavioural response, and there are likely multiple substances that have no discernible effect at all. The base model is constructed in such a way that those similar substances are considered to be just as different as sedatives compared with central stimulants, to give an example. The

same goes for different dose levels, which might be problematic for substances whose effects are relatively independent of dose. Disregarding the fact that substance and dose similarity should be a spectrum rather than binary might be a reason for the model's inability to improve further, since it could then receive contradictory or inaccurate label information. The attempted solution in section 4.4.3, using semi-positive relations for different dose levels of the same substance, made no significant difference regarding performance. This means that if the problem arises from inaccurately defined dissimilarities, only considering dose level is not enough. Having a corresponding way to specify substance dissimilarities in a more nuanced manner might therefore be helpful. However, this would require knowledge about how the different substances relate to each other, which is not possible since many of the substances tested do not have clearly profiled effects.

Lastly, it is important not to forget that the data is based on living rats, whose behaviour depends on many factors other than just administered substances. The time of day, impressions made by the handler and earlier experiences are just a few examples of such consequential factors. If the substance-induced behaviours of the rats are drowned out by their natural behaviour variations, it will not be possible for the model to represent those trajectories differently from control trajectories without overfitting. This problem might be amplified by the fact that only the position of the rats is recorded. Using video data, the model would in theory be able to detect more subtle behavioural patterns, such as different stationary activities. However, there would also be considerably more noise from natural behaviour variations, so it is not sure that the task would become easier.

5.2 Differences to other CL Datasets

Since CL methods have been proven to be successful in the field of representation learning [10], a comparison of the different datasets used could be helpful to understand the lack of improvement observed in this project. The SimCLR [1] encoder is trained on the ILSVRC2012 dataset [44], which is a subset of the ImageNet dataset. It contains 1000 object classes with a total of 1,431,167 annotated images, of which approximately 1,200,000 are used as training data. ILSVRC2012 thus contains around 1200 times as many training images as it contains classes, whereas the corresponding proportion for our dataset is 6.5 if we consider trajectories with the same substances and doses to comprise one class. This large difference could be a reason for why the contrastive learning approach is considered to perform well in other situations, but not for us.

One study examining SimCLR performance in different settings [45] found that an increased number of training samples notably improves the amount of relevant information captured by the embeddings, up to a certain point. Without changing the number of classes, different amounts of ImageNet images were used to train a SimCLR model as described by Chen, Kornblith, Norouzi, *et al.* [1]. Compared to 50,000 training images, the top-1 accuracy achieved by using the embeddings for classification increased almost 20 percentage points when including 500,000 images. Improvements diminished when increasing the amount of training data further, but

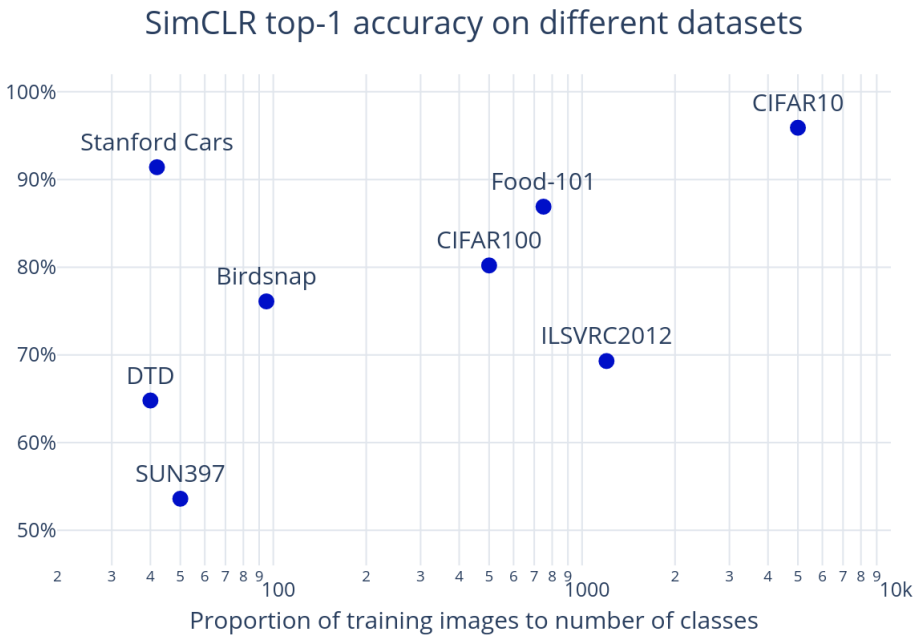


Figure 5.1: Reported top-1 accuracy of SimCLR trained on different image datasets. The datasets shown are of differing sizes, and have different proportions of training images to the number of object classes. A positive trend can be observed between proportion and classification accuracy.

the results highlight that more data certainly helps in contrastive learning.

This pattern can also be observed when investigating the contrastive learning performance presented by Chen, Kornblith, Norouzi, *et al.* [1] on different datasets. A SimCLR model was trained and evaluated on eight different image datasets, where the evaluation was performed using the standard linear classifier protocol and top-1 accuracy.

While the reported results presented in fig. 5.1 indicate that a higher proportion of training samples to classes improve the quality of the latent representations, the correlation is not obvious. This is of course expected, since many aspects of a dataset determine how well a model is able to learn the data. One such aspect related to the evaluation method used is the amount of test data. Since the linear classifier is trained using the embedded test samples, a bigger test dataset tends to improve the classifier and thereby achieve higher accuracy [45]. The Stanford Cars dataset [46] for example has a test set that is as big as the training dataset, which might help explain the high top-1 accuracy using that dataset. Another important aspect is the quality of the data. This includes image quality when training the model, but also the labels used for testing, such as the percentage of incorrectly labelled test images. The image quality is also related to the difficulty inherent in what the images depict. Clear and easily learned differences between object classes obviously improve the chance of a model performing well.

All these aspects are worth considering when comparing our dataset to the ones normally used for contrastive learning. The most notable difference of course is that

the rat trajectories are time series, and not images. However, since the trajectories are still processed using convolutions in the temporal domain, they could be seen as very long one-dimensional images with (x, y, z)-channels. Seen from this perspective, our ResNet-inspired encoder should be able to embed them in a satisfactory manner given the appropriate gradients. This view is of course very simplified, since other differences could still have a large impact. One such potentially significant difference is the more aggressive downsampling we perform to go from 90,000 temporal dimensions to one. Moreover, the dataset we use has only 6.5 times as many trajectories as there are substance-dose groups, a proportion that is significantly lower than the image datasets shown in fig. 5.1. Since there is also an overlap arising from different substances that yield the same behavioural changes or none at all, the data could be considered challenging for contrastive learning purposes.

This relative difficulty compared to other datasets becomes more apparent when considering augmentations. There are many different augmentations for images that change style, but not content given some problem setting. Examples of these are colour distortion, cropping and resizing, mirroring, or Gaussian noise [1]. The problem with rat movement trajectories is that many of these augmentations also change content, i.e. how the rat appears to behave. Cropping and resizing an image of a dog would still result in an image depicting a dog, unless the crop is too small. The corresponding augmentation performed on a trajectory would make the rat appear as if it was moving slower, as a result of the resized crop. Colour distortions are not applicable, and Gaussian noise applied to the Cartesian positions would make the rat appear jittery, which would also be interpreted as abnormal behaviour. The mirroring and rotation augmentations are viable, as described in section 4.3.2. Other augmentations, for example Gaussian noise can be configured to change style but not content when using egocentric coordinates, which can be seen in section 4.3.3. However, as the corresponding results indicate, the encoder does not seem to interpret these augmented trajectories very differently than the original ones. The augmentations are thus not enough to isolate all actual substance-induced behavioural changes. While other augmentation could be conceived that remove more of the "behavioural background", these are not as trivial to find as when using images.

5.3 Suitability of the Chosen Approach

Even though the examined contrastive learning models perform on par with the currently used manual model, we believe that there is more to be learned from the extensive dataset used in this study. Therefore, the proposed base model and various modifications should not, without further alteration, be considered sufficient for the task of comparative behavioural analysis on this dataset. This leads to the question of whether a completely different approach should be pursued, or if the one described in this thesis warrants further study.

While we have not explored other encoder architectures than TCNs, we deem it possible, but not likely, that the lack of high performance is primarily caused by the encoder. The main reason for this rationale is that the training loss of the base

model is able to become significantly lower than at the start of training, as can be seen in fig. 4.1. Furthermore, the overfitting behaviour shows that the model is able to learn spurious features in the data, which also implies that the encoder is complex enough. It could be possible that the actually relevant features are much harder to extract than the spurious ones, and therefore require a more sophisticated encoder. However, given the restrictions posed by the long time series data, it is not trivial to come up with other architectures with higher capacity than our encoder inspired by the widely substantiated ResNet architecture [11]. Therefore, there are likely other aspects of the problem worth examining before turning to alternative encoder architectures.

Regarding the method used to train the encoder, it is difficult to determine if contrastive learning is a viable approach for our task. As described in section 5.2, there are some notable differences between the dataset we use and datasets used in other contrastive learning contexts. Fewer positive samples and augmentations are two examples, and could very well be causing the problem of overfitting. This does not mean that contrastive learning should be ruled out, but more sophisticated techniques for augmentations or increasing group size would likely be needed. Other machine learning techniques would probably also suffer from the low number of positive samples, but one problem amplified by contrastive learning is the similarity between some of the negative samples. If the behavioural response is very similar, two trajectories should not be separated even when different substances have been administered. Formulating the model objective in this way might force the model to find spurious patterns that cannot be generalized, in order to separate such similar trajectories. Considering this problem, approaches that do not rely so heavily on negative samples could be more suitable. However, there are also potential solutions compatible with contrastive learning, for example revising what constitutes a negative relation and thereby mitigating contradictory labelling.

Another factor that could potentially have limited the performance is the independent testing of model modifications. In general, each modification was tested individually, and since no significant improvements were achieved, the modification was reverted and another change was tested. This approach was adopted to maintain a manageable number of runs for the project. However, it assumes that the impact of each model adjustment on performance is independent of others, disregarding the possibility that certain modifications may not yield improvements individually but could potentially do so when combined. Nevertheless, it is unlikely that two changes causing minimal performance impact would collectively contribute to significant improvement. This assumption, though, can only be confirmed through empirical evidence. An exception was made to the regular testing procedure when varying the output size of the encoder and projection head (see section 4.2.2), as the model's performance is likely heavily dependent on their combination. For instance, it is probable that a very high output size of the encoder would necessitate a relatively large output size from the projection head to achieve satisfactory performance.

The evaluation methods used in this project merit discussion, since it is not trivial how to evaluate the behavioural representations. The reason for having multiple different metrics is the lack of ground truth information relating the behavioural

effects of different substances. The proposed metrics are therefore meant to complement each other. For example, the dose response metric does not consider whether different trajectories are grouped or not, as long as they are correctly ordered with respect to dose level and the relevant control cluster. This is complemented by the WSS-TSS scores, since they measure the dispersion of same-group trajectories in relation with total dispersion. Using multiple scores could however introduce difficulties when comparing different models that perform well with respect to different scores. Having a way to rank the metrics according to importance will probably be desirable when the models start performing sufficiently well, and one final model is to be chosen for deployment. If better models are created which require reliable and precise evaluation metrics, the design of the MLP-classifier would likely need to be revisited. More specifically, the use of the indication labels as targets are sub-optimal, since some of the labels are vague and might contain substances with completely different behavioural effects.

A fundamental motivation behind this project has been to develop a model that relies more heavily on the dataset itself rather than prior knowledge, as the manual model is. One reason for this approach is to mitigate human bias that can arise when manually engineering features. However, it is crucial to acknowledge that machine learning models are not inherently free from bias. Every choice made during the model's development involves certain assumptions. For instance, considering all pairs of substances as negative implies that the embeddings from these trajectories should be distinct, which may not necessarily be accurate. Given our belief that the base model can benefit from incorporating additional domain knowledge, it becomes even more important to be aware of what assumptions are being made and to critically evaluate the potential biases that may arise.

Lastly, there are some challenges that would most likely remain, even if another approach than contrastive learning is chosen. Two of these problems described in section 5.1 are the long time frame of data collection, and the individual behaviour of the rats. Both of these aspects affect the noise-to-signal ratio of the data, which limits how much relevant information can actually be obtained from the trajectories. There is a risk that the trajectories contain so much noise that considerably higher performance is not attainable, which would be problematic regardless of approach. However, a thorough data analysis would be needed in order to gauge the severity of this problem. Considering the potential value of a well-performing behavioural model, we believe that this task is worth further attention, either with the approach described in this thesis or some alternative one.

5.4 Conclusion and Future Work

The main objective of this project has been to develop a machine learning model that can successfully find behavioural differences between rats that have been given different substances. In order to measure performance, we have developed several metrics that measure different desired properties of the model. Since the proposed model performs on par with or better than the previously made manual model, we would say that our general objective has been accomplished. We have also made

modifications to the model with the intent of improving performance, mainly by decreasing overfitting. These changes have been regarding the encoder, loss function and the data, but the model has been surprisingly invariant to most modifications. Our findings suggest that problems might lie within unconsidered aspects of the dataset, or that our CL approach to this problem is not optimal. However, by developing the evaluation metrics and examining different aspects of the model, we have created a strong foundation from which future studies can proceed. Below, we present some ideas that might be of interest to explore in further studies.

Centering the Control Cluster
One small change that could lead to improvements is to treat the control group differently from the other groups. Given that the control group represents the "standard behaviour" for a rat, it would be reasonable to expect these embeddings to be positioned closer to the centre in relation to all other embeddings. Though, as seen in fig. 4.8, section 4.1, the control is pushed away from all the other groups. Training the model with the SupCon loss encourages this property since the control group is large and the loss decreases when its similarities with other groups are minimized. From the model's perspective, it therefore makes sense to put the control group at the edge instead of in the middle, since this allows for further distances and thereby minimized similarities. To address this, the SupCon loss could be modified by adding a term encouraging each control group embedding to be close to the centre of all embeddings $\bar{z}$. One possible suggestion is by the term $\lambda Y(i)||z_i - \bar{z}||$, where $\lambda > 0$ is a constant and $Y(i)$ is a binary variable that is 1 if z_i belongs to the control group and 0 else. By incorporating such a modification, the model may potentially improve its ability to differentiate between other substances, rather than solely focusing on separating the control group from all other substances. This adjustment could lead to more informative features of the different substance-induced behaviours.

Further Data Augmentations
One suggestion supported by other studies in contrastive learning is to implement additional and more aggressive data augmentations. The augmentations examined in this thesis did not affect the model performance in a notable way, which could indicate that they are not extensive enough. As suggested in section 4.3.3, the egocentric augmentations could also be removing too much content information regarding a rat's position in the cage. This highlights the earlier mentioned challenge of creating augmentations that extensively change the trajectories' style without actually changing the underlying behaviour. Below, we suggest a type of augmentation based on substituting parts of different positive samples, in order to train the network on new and previously unseen combined trajectories.

The behavioural effects of some substances will differ over time as it is metabolized. However, assuming that the drug administration to all positive rats is synchronized with respect to the trajectory recordings, positive rats should exhibit similar behavioural changes at any given time of the trajectories. Therefore, if one section of a trajectory was to be replaced with the corresponding section from another positive sample, the resulting trajectory should still depict the same behavioural change. A problem could arise at the transition between two trajectory segments, where there

would be some undesired apparent teleportation in the cage. This could however be avoided by using egocentric coordinates, so that the position would appear to remain the same. By substituting random sections of a trajectory with the corresponding sections of other positive ones, it would thereby be possible to create new patchwork trajectories with the same behavioural changes as their constituent parts.

Apart from increasing the number of different positive samples seen by the model, this augmentation could potentially have another important effect. Since the rats are living animals with complex behaviours, each rat will to some extent have a unique "personality", or base behaviour. This poses a challenge, since the model must learn to separate this unrelated behaviour from the change induced by the substance. By combining different trajectories in the way described above, there is a chance that the individual behavioural differences could average each other out, and thereby isolate the consistent substance-induced behaviour. Alternatively, this kind of substitution could also be performed on the latent embeddings. Switching the corresponding elements of two positive embeddings could then create new valid combinations, since positive embeddings should represent the same behaviour.

Semi-negative Relations

A common issue in other contrastive learning studies is that few negatives actually contribute to the learning process, which motivates using a large batch size and hard negative mining, as discussed in section 2.6. Contrarily to those findings, our results display that this lack of hard negatives is not an issue for this project. Increasing the batch size and using harder negatives (in combination with triplet margin loss) did not improve performance. These results suggest that the problem is not primarily caused by the negatives being too easy. It could even be possible that the current negatives are excessively challenging, which may be problematic. For instance, consider a scenario where the model aims to distinguish trajectories from rats administered different substances that induce similar behavioural changes. To address this, one approach could involve transforming the negative relation from binary to categorical, similar to the introduction of semi-positives. The magnitude of a negative relation could for example depend on other neurological metadata, such as similarities in which receptors the different substances interact with. Thereby, samples with different substances but similar behavioural changes would hopefully not be considered as negative as samples from substances causing substantially different behaviours. This adaption would lessen the constraint on the model to separate all different substances regardless of behavioural effect, potentially leading to more informative behavioural features. However, it is not clear what kind of additional information would be best suited for determining the more nuanced negative relations, or how that information should be incorporated. If this approach were to be further explored, some domain expertise would likely be needed.

SimSiam: an Alternative to Contrastive Learning

In order to reduce the potential problems that come with negative relations, it could be worth considering techniques that do not rely on negatives at all. One such example is *SimSiam*, presented by Chen and He [2]. The overall structure is similar to SimCLR, in that two augmented versions of the same data sample are encoded and represented as similarly as possible. Recall that SimCLR also

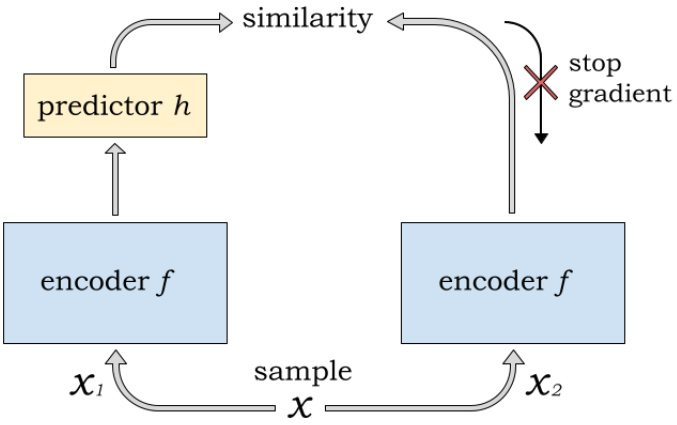


Figure 5.2: A visualisation of the structure used in SimSiam.

separates dissimilar samples, in order to avoid a trivial solution where all samples are represented in the same way. SimSiam avoids such trivial solutions by including a MLP h that, given one of the positive embedded samples, predicts the representation of the other sample. This prediction is then compared with the real representation of the other sample using some similarity measure, which functions as the model loss. The gradients from the sample that does not pass through h are stopped, which is shown to be crucial in order to avoid model collapse. For a schematic visualisation of the structure, see fig. 5.2.

Considering the possibility that questionable negative relations might be a cause of the overfitting behaviour observed in this project, a technique such as the one mentioned above could be a step in the right direction. However, the lack of negatives means that the model must rely solely on the positive relations to create its embedding space. As previously observed in section 5.2, the dataset we use has a lot fewer same-group samples and not as many apparent augmentation options as the image datasets that SimSiam was developed for [2]. Therefore, more extensive augmentation techniques might be required before SimSiam can be made to work well for this task and dataset.

Bibliography

[1] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning*, PMLR, Nov. 21, 2020, pp. 1597–1607.

[2] X. Chen and K. He, "Exploring simple siamese representation learning," presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 15 750–15 758.

[3] R. Jiang, T. Nguyen, P. Ishwar, and S. Aeron, "Supervised contrastive learning with hard negative samples," Aug. 31, 2022. DOI: 10.48550/arXiv.2209.00078. (visited on 02/20/2023).

[4] "IRLAB website," About IRLAB, [Online]. Available: https://irlab.se/about-irlab/ (visited on 06/09/2023).

[5] R. S. Feldman, J. S. Meyer, and L. F. Quenzer, *Principles of neuropsychopharmacology*. Sunderland, Mass: Sinauer Associates, 1997.

[6] S. Schneider, J. H. Lee, and M. W. Mathis, "Learnable latent embeddings for joint behavioural and neural analysis," *Nature*, vol. 617, no. 7960, pp. 360–368, May 2023. DOI: 10.1038/s41586-023-06031-6.

[7] K. Luxem, P. Mocellin, F. Fuhrmann, *et al.*, "Identifying behavioral structure from deep variational embeddings of animal motion," *Communications Biology*, vol. 5, no. 1, p. 1267, Nov. 18, 2022. DOI: 10.1038/s42003-022-04080-7.

[8] S. Waters, P. Svensson, J. Kullingsjö, *et al.*, "In vivo systems response profiling and multivariate classification of CNS active compounds: A structured tool for CNS drug discovery," *ACS Chemical Neuroscience*, vol. 8, no. 4, pp. 785–797, Apr. 19, 2017. DOI: 10.1021/acschemneuro.6b00371.

[9] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013. DOI: 10.1109/TPAMI.2013.50.

[10] S. Appalaraju, Y. Zhu, Y. Xie, and I. Fehérvári, "Towards good practices in self-supervised representation learning," 2020. DOI: 10.48550/ARXIV.2012.00868. (visited on 05/17/2023).

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[12] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," Apr. 19, 2018. DOI: 10.48550/arXiv.1803.01271. (visited on 02/02/2023).

[13] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 156–165.

[14] R. Wan, S. Mei, J. Wang, M. Liu, and F. Yang, "Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting," *Electronics*, vol. 8, no. 8, p. 876, Aug. 7, 2019. DOI: 10.3390/electronics8080876.

[15] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?," 2022. DOI: 10.48550/ARXIV.2205.13504. (visited on 06/14/2023).

[16] "Sustainable development goals | united nations development programme," UNDP, [Online]. Available: https://www.undp.org/sustainable-development-goals (visited on 12/14/2022).

[17] Nuffield Council on Bioethics, Ed., *The ethics of research involving animals*, London: Nuffield Council on Bioethics, 2005, ISBN: 9781904384106.

[18] L. Keeling, H. Tunón, G. Olmos Antillón, *et al.*, "Animal welfare and the united nations sustainable development goals," *Frontiers in Veterinary Science*, vol. 6, p. 336, Oct. 10, 2019. DOI: 10.3389/fvets.2019.00336.

[19] S. Albelwi, "Survey on self-supervised learning: Auxiliary pretext tasks and contrastive learning methods in imaging," *Entropy*, vol. 24, no. 4, Apr. 14, 2022. DOI: 10.3390/e24040551.

[20] J. von Kügelgen, Y. Sharma, L. Gresele, *et al.*, "Self-supervised learning with data augmentations provably isolates content from style," in *Advances in Neural Information Processing Systems*, vol. 34, Curran Associates, Inc., 2021, pp. 16 451–16 467.

[21] R. P. Monti, S. Tootoonian, and R. Cao, "Avoiding degradation in deep feed-forward networks by phasing out skip-connections," in *Artificial Neural Networks and Machine Learning ICANN 2018*, vol. 11141, Cham: Springer International Publishing, 2018, pp. 447–456. DOI: 10.1007/978-3-030-01424-7_44.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[23] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is nearest neighbor meaningful?" In *Database Theory ICDT99*, vol. 1540, Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 217–235. DOI: 10.1007/3-540-49257-7_15.

[24] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *Database Theory ICDT 2001*, vol. 1973, Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 420–434. DOI: 10.1007/3-540-44503-X_27.

[25] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Com-*

puter Vision and Pattern Recognition - Volume 2 (CVPR'06), vol. 2, New York, NY, USA: IEEE, 2006, pp. 1735–1742. DOI: 10.1109/CVPR.2006.100.

[26] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks," in *Procedings of the British Machine Vision Conference 2016*, York, UK: British Machine Vision Association, 2016, pp. 119.1–119.11. DOI: 10.5244/C.30.119.

[27] H. Xuan, A. Stylianou, and R. Pless, "Improved embeddings with easy positive triplet mining," presented at the Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2020, pp. 2474–2482.

[28] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, "Sampling matters in deep embedding learning," presented at the Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2840–2848.

[29] P. Khosla, P. Teterwak, C. Wang, *et al.*, "Supervised contrastive learning," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 18 661–18 673.

[30] B. K. Iwana and S. Uchida, "An empirical survey of data augmentation for time series classification with neural networks," *PLOS ONE*, vol. 16, no. 7, e0254841, Jul. 15, 2021. DOI: 10.1371/journal.pone.0254841.

[31] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Proceedings of the 4th International Conference on Neural Information Processing Systems*, ser. NIPS'91, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Dec. 2, 1991, pp. 950–957.

[32] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, PMLR, Jun. 1, 2015, pp. 448–456.

[33] P. Luo, X. Wang, W. Shao, and Z. Peng, "Towards understanding regularization in batch normalization," 2018. DOI: 10.48550/ARXIV.1809.00846. (visited on 06/05/2023).

[34] C. Garbin, X. Zhu, and O. Marques, "Dropout vs. batch normalization: An empirical study of their impact to deep learning," *Multimedia Tools and Applications*, vol. 79, no. 19, pp. 12 777–12 815, May 2020. DOI: 10.1007/s11042-019-08453-9.

[35] D. M. W. Powers, "Evaluation: From precision, recall and f-measure to ROC, informedness, markedness and correlation," 2020. DOI: 10.48550/ARXIV.2010.16061. (visited on 05/28/2023).

[36] J.-O. Palacio-Niño and F. Berzal, "Evaluation metrics for unsupervised learning algorithms," 2019. DOI: 10.48550/ARXIV.1905.05667. (visited on 05/30/2023).

[37] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979. DOI: 10.1109/TPAMI.1979.4766909.

[38] K. Gupta, T. Ajanthan, A. v. d. Hengel, and S. Gould, "Understanding and improving the role of projection head in self-supervised learning," 2022. DOI: 10.48550/ARXIV.2212.11491. (visited on 04/13/2023).

[39] C.-H. Yeh, C.-Y. Hong, Y.-C. Hsu, T.-L. Liu, Y. Chen, and Y. LeCun, "Decoupled contrastive learning," 2021. DOI: 10.48550/ARXIV.2110.06848. (visited on 05/23/2023).

[40] Y. Kalantidis, M. B. Sariyildiz, N. Pion, P. Weinzaepfel, and D. Larlus, "Hard negative mixing for contrastive learning," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 21 798–21 809.

[41] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola, "What makes for good views for contrastive learning?" In *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 6827–6839.

[42] K. Musgrave, S. Belongie, and S.-N. Lim, "A metric learning reality check," in *Computer Vision ECCV 2020*, ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2020, pp. 681–699. DOI: 10.1007/978-3-030-58595-2_41.

[43] "Adam PyTorch 2.0 documentation," [Online]. Available: https://pytorch.org/docs/stable/generated/torch.optim.Adam.html (visited on 05/15/2023).

[44] O. Russakovsky, J. Deng, H. Su, *et al.*, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec. 1, 2015. DOI: 10.1007/s11263-015-0816-y.

[45] E. Cole, X. Yang, K. Wilber, O. Mac Aodha, and S. Belongie, "When does contrastive visual representation learning work?," presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 14 755–14 764.

[46] L. Yang, P. Luo, C. C. Loy, and X. Tang, "A large-scale car dataset for fine-grained categorization and verification," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 3973–3981. DOI: 10.1109/CVPR.2015.7299023.

A

Appendix 1

Table A.1: Summary of each block of the encoder in the base model. The table is divided into convolutional (main) and residual connections and specifies the output dimensions of each layer, where the first dimension is arbitrary and equals the batch size. The "Zero pad"-layer acts between blocks to make the input divisible by two, which simplifies calculations regarding dimension agreement.

Block	**Main connection**	**Output Shape**	**#Params**	**Residual connection**	**Output Shape**
1	StridedConv1d	[-1, 6, 45000]	42	AveragePool	[-1, 3, 45000]
1	NormalConv1d	[-1, 6, 45000]	114	AddZeros	[-1, 6, 45000]
2	StridedConv1d	[-1, 12, 22500]	156	AveragePool	[-1, 6, 22500]
2	NormalConv1d	[-1, 12, 22500]	444	AddZeros	[-1, 12, 22500]
3	StridedConv1d	[-1, 18, 11250]	450	AveragePool	[-1, 12, 11250]
3	NormalConv1d	[-1, 18, 11250]	990	AddZeros	[-1, 18, 11250]
4	StridedConv1d	[-1, 30, 5625]	1,110	AveragePool	[-1, 18, 5625]
4	NormalConv1d	[-1, 30, 5625]	2,730	AddZeros	[-1, 30, 5625]
4.5	Zero pad	[-1, 30, 5626]	—	—	—
5	StridedConv1d	[-1, 48, 2813]	2,928	AveragePool	[-1, 30, 2813]
5	NormalConv1d	[-1, 48, 2813]	6,960	AddZeros	[-1, 48, 2813]
5.5	Zero pad	[-1, 48, 2814]	—	—	—
6	StridedConv1d	[-1, 75, 1407]	7,275	AveragePool	[-1, 48, 1407]
6	NormalConv1d	[-1, 75, 1407]	16,950	AddZeros	[-1, 75, 1407]
6.5	Zero pad	[-1, 75, 1408]	—	—	—
7	StridedConv1d	[-1, 100, 704]	15,100	AveragePool	[-1, 75, 704]
7	NormalConv1d	[-1, 100, 704]	30,100	AddZeros	[-1, 100, 704]
8	StridedConv1d	[-1, 100, 352]	20,100	AveragePool	[-1, 100, 352]
8	NormalConv1d	[-1, 100, 352]	30,100	—	—
9	StridedConv1d	[-1, 100, 176]	20,100	AveragePool	[-1, 100, 176]
9	NormalConv1d	[-1, 100, 176]	30,100	—	—
10	StridedConv1d	[-1, 100, 88]	20,100	AveragePool	[-1, 100, 88]
10	NormalConv1d	[-1, 100, 88]	30,100	—	—
11	StridedConv1d	[-1, 100, 44]	20,100	AveragePool	[-1, 100, 44]
11	NormalConv1d	[-1, 100, 44]	30,100	—	—
12	StridedConv1d	[-1, 100, 22]	20,100	AveragePool	[-1, 100, 22]
12	NormalConv1d	[-1, 100, 22]	30,100	—	—
13	StridedConv1d	[-1, 100, 11]	20,100	AveragePool	[-1, 100, 11]
13	NormalConv1d	[-1, 100, 11]	30,100	—	—
13.5	Zero pad	[-1, 100, 12]	—	—	—
14	StridedConv1d	[-1, 100, 6]	20,100	AveragePool	[-1, 100, 6]
14	NormalConv1d	[-1, 100, 6]	30,100	—	—
15	StridedConv1d	[-1, 100, 3]	20,100	AveragePool	[-1, 100, 3]
15	NormalConv1d	[-1, 100, 3]	30,100	—	—
Final	FinalPool	[-1, 100, 1]	—	—	—

Table A.2: Parameter specification of layers inside the encoder of the base model.

	Kernel Size	**Stride**	**Padding**
StridedConv1d	2	2	0
NormalConv1d	3	1	1
AveragePool	2	2	0
FinalPool	3	1	0

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY