Praktikum Pattern Analysis and Machine Intelligence

Self-supervised learning - Contrastive learning

Investigating the effects of different image augmentations and data sets on pairwise and triplet loss

submitted to:
Julia Ändern
Faculty of Science
Johann Wolfgang Goethe-Universität
Frankfurt am Main

by
Robin Hammer (7224832)
Marc Pavlinec (6172615)
Nathalie Zarbock (6077231)
Anika Zeilmann (7396667)

Table of contents

List of figures

List of abbreviations

BYOL	Bootstrap you own latent
CIFAR10	Canadian Institute for Advanced Research, 10 classes
CNN	Convolutional Neural Network
Fig	Figure
MNIST	Modified National Institute of Standards and Technology database
NLP	natural language processing
n.d.	No date
n.p.	No page
PCA	Principal Component Analysis
ReLu	Rectified Linear Unit
SimCLR	Simple Framework for Contrastive Learning of visual Representations
SNN	Siamese Neural Network
t-SNE	t-distributed stochastic neighbor embedding

1 Motivation & Introduction (Anika)

In the last years, contrastive learning developed to be a topic of interest for many researchers. They constantly develop new architectures and try to improve models. (Yolyan, n.d.) Contrastive learning thereby refers to a technique in machine learning that can learn general features of a data set without needing labels. It is doing so by teaching the model which data points are similar or dissimilar. (Tui, 2021) Instead of using pre-labeling, Contrastive learning as a technique defines “pseudo” labels as standing for the similarity or dissimilarity of inputs. (Yolyan, n.d.) The ability to learn without pre-labels is what attracts many researchers since they often face scarcity of clean and labeled data. Data like that is either expensive or even not available at all. (Yolyan, n.d.) An example for hard to label data is medical imaging. (Tui, 2021)

The objective of this project paper is to elaborate on the effects of changing major parameters in contrastive learning. For that, two data sets will be compared, with three different augmentations each. Additionally, each pairwise and triplet loss will be implemented, generating twelve different scenarios to analyze. Since the focus lays on the comparison, we do not aim for the best accuracy of the model, but for similar conditions of each network. Therefore, we also do not strive for using contrastive learning on data sets that would suggest doing so, but rather on two data sets that provide different complexity levels of images.

This paper is structured as follows. First, we want to explain the basic knowledge needed to understand and implement contrastive learning. Afterwards, we will go into more detail on what contrastive learning is and how it works. The next chapter will introduce the data sets used, followed by an explanation of our implementation. After evaluating our results, a conclusion will be drawn and an outlook on the topic will be given.

2 Basic Knowledge

This chapter is supposed to describe the basics needed for understanding contrastive loss.

2.1 Supervised vs Unsupervised Learning (Robin)

A common distinction for learning methods in machine learning literature is the distinction between supervised and unsupervised learning.

In the supervised setting, the training data consists of input vectors, each with corresponding target vectors.(Bishop, 2006, p. 3) The target vectors can stem from a finite, discrete number of cases – this would be a typical case for classification. In the case of regression, the target vectors form one or more continuous variable. A supervised model estimates the value of the target vector as a (in some cases probabilistic) function of the input vectors. (Provost & Fawcett, 2013, p. 46)

Unsupervised learning problems do not involve target vectors. Use cases may be finding groups of similar data points (clustering), determining the ‘distribution of data with the input space’ (density estimation), or visualizing high-dimensional data into two- or three-dimensional space. (Bishop, 2006, p. 3)

As the upcoming descriptions on contrastive learning will show, contrastive learning can take the form of supervised and the form of unsupervised learning. Newer approaches also use contrastive learning for self-supervised learning: the model automatically tries to extract labels from the data without using label information for training. Usually this approach is not used for directly training a model, but for transfor learning: The model is pre-trained self-supervised, and then fine-tuned for another task or dataset. (Howard & Gugger, 2020, p. 329)

2.2 Classification (Robin)

Classification is a form of supervised learning that tries to predict a categorical variable, therefore *classifying* an observation by assigning it to a predefined category/ class. (James et al., 2013, p. 127) The methods for classification range for example from linear methods like logistic regression, over tree-based models like decision trees and random forests to deep learning methods using neural networks.

A variety of practical, often everyday problems are solved with classification: In medicine, diseases can be inferred based on various symptoms or genetic predisposition.

The recognition of objects in pictures and in videos is done with classification methods. There are numerous use cases in the private sector as well, such as checking the creditworthiness of bank customers or customer segmentation. (Provost & Fawcett, 2013, p. 1f) While all of these methods can be used for structured, tabular data, deep learning methods are “nearly always clearly superior” for the classification of unstructured data like image or audio data. (Howard & Gugger, 2020, p. 282)

2.3 CNN (Anika)

Convolutional neural networks (CNNs) are one of the most used deep neural networks. The name is coming from the “linear operation between matrixes”, which is called convolution. (Albawi et al., 2017, p.1) CNNs are specifically designed for the usage in computer vision, which tries to enable machines to perceive visual input as humans do. It is for instance used for tasks like image recognition or image analysis and classification. The architecture of CNNs imitate the pattern of neurons in the human brain, more specifically the visual cortex. (Saha, 2018)

CNNs consist of multiple layers: A convolutional layer, non-linear layer, pooling layer, and fully-connected layer. (Albawi et al., 2017, p.1) They operate at a pixel level and can learn features and representations of an image in an integrated way. (Hadsell et al., 2005) The CNN algorithm takes an input image and assigns weights and biases to objects on the image, which enables it to differentiate between these objects. (Saha, 2018) In contrast to other algorithms, CNNs reduces the effort for pre-processing, since it can learn the necessary characteristics of filters. (Saha, 2018)

2.4 Visualization Methods for un/ self-supervised Learning (Nathalie)

Principal Component Analysis or short PCA is a way to reduce high-dimensional data into a lower dimension while preserving the information about the data from a higher dimension. An advantage for PCA is that it can visualize dimensions with high variance or large pairwise distance better which leads to large distances in the visualized points of dissimilar images. (Shrivastav, 2019)

T-distributed Stochastic Neighbor Embedding (t-SNE) is method like PCA to transform high-dimensional data into lower-dimensional data. T-SNE differs from PCA that it can keep local similarities between points. Another difference is that PCA often uses a dimension of two to display the data in a lower dimension while t-SNE can also

visualize data in a two-dimensional space but also in three-dimensional space. This makes it possible to compare how related different classes are to each other. Using t-SNE it is possible to adjust a value called perplexity which is a number, usually a value between 5 and 50, to set the effective number of neighbors. (van der Maaten & Hinton, n.d.)

A classifier, as an additional step, is often used along with a visualization, as the output of the initial network can't decide to which class or even potential classes an image belongs to. (Shrivastav, 2019)

Another method for visualization of the training's results would be activation maps. Activation maps can visualize on top of an image with colors indicating the heat or relevance of that specific part of an image which contributes how much to the output of the network and explains it's prediction rather than just viewing the network as a black box. (Choi et al., 2020)

3 Contrastive Learning

3.1 Basic Approach and Intuition (Robin)

The motivation for contrastive learning is to find a good representation of data, that can be used for several tasks. What does 'good representation' mean? While that is not obvious, and depends on the task, there are two attributes that a representation should clearly have: First, the representation should be meaningful in regard to the ground data – that is, some features of the data should be preserved. Second, in order to coherently make use of a representation, similar pairs stay close to each other, while dissimilar ones are far apart. (Hadsell et al., 2006, p. 1) This principle of having similar features close to each other, while keeping dissimilar features apart, is present in all the different contrastive learning approaches.

Contrastive Learning is therefore a form of representation learning, that tries to map a raw data input to a feature vector, trying to extract abstract, 'useful' conceptual features of the data that can be used for a range of downstream tasks. (Le-Khac et al., 2020, p. 1) The nature of the abstract learned features should be invariant to local variations. (Bengio et al., 2014, p. 5) For the use case of object representation in image data, this means that for example rotation, different light influences and different perspectives of an object should ideally not affect the representation of the object.

More concretely, this representation learning is done by comparing among different input samples. The input includes the information if the samples are similar – positives – or dissimilar – negatives. The goal of training contrastive learning is then to learn a representation that maps ‘similar’ examples close together, while mapping dissimilar samples far away. (Khosla et al., 2021, p. 1)

There is a wide array of applications for contrastive learning, in the range of natural language processing (NLP), audio data, graph-structured data and visual data. (Le-Khac et al., 2020)

This report and our project’s use case focusses on the use of contrastive learning for visual data and image representation. The upcoming examples and description should therefore be seen in this setting. While there are for example data augmentation methods for NLP, this paper will only describe augmentation methods for images.

Applications for contrastive learning in the domain of visual data are widespread. The distinction between ‘similar’ and ‘dissimilar’ pictures can for example be used for verification and fine-grained classification problems. (Le-Khac et al., 2020, p. 16) Another use-case is self-supervised representation learning. For example, Sermanet et al. (2018) use contrastive learning for finding common functional instances of roboter movements in video in a self-supervised setting, contrasting them to movement that looks similar, but has different functions. Furthermore, learning a contrastive representation of images can be used to for recommending similar images to users of an image software. (Chechik et al., 2010, p. 1110)

3.2 Creation of similar and dissimilar Information (Marc)

To make contrastive learning even possible, it is necessary for the learner to have at least two datapoints provided to compare so that one could be deemed similar to or different from the other. The types of data can range from images (vision data, see SimCLR) to script (language data, see "word2vec" (Mikolov et al., 2013)), audio data (see "wav2vec" (Schneider et al., 2019)), graph-structured data (see "node2vec" (Grover & Leskovec, 2016)) or even multimodal data (see COALA (Favory et al., 2020)).

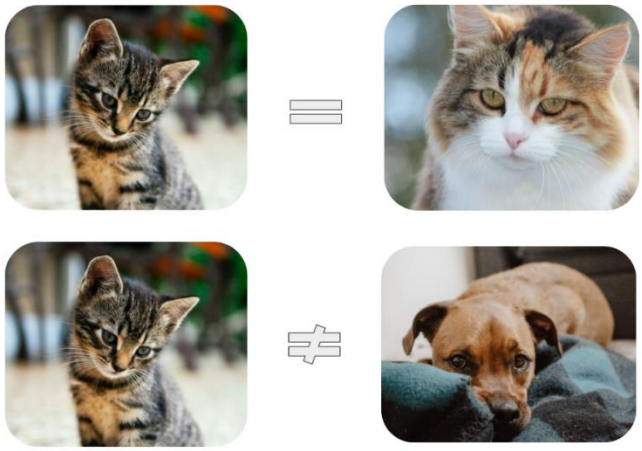

Figure 1 - Two cats looking similar to each other rather than cat and dog (Tiu, 2021)

There are various approaches to create positive or negative comparison material. Both types of pairs make use of an anchor image and have their respective comparison image set based on said anchor. The simplest way of creating such pairs would be to have an image of the same category as the anchor for a positive example and an image of a different category for a negative example, which is also what Chechik et al. (Bishop, 2006) used for research on image similarity learning. This or similar methods presume that the datasets are already clustered in some way (e.g. classified) to filter out images of the same class as or any other class than the anchor image class. Other possibilities to create comparison images would be the usage of multi-sensory prerequisites, e.g. using the same image shot from a different angle or camera (see TCN (Sermanet et al., 2018b)), invariances from context-instance relationships like the use of global and local features (see DIM (Hjelm et al., 2019)), time-related sequential coherence (see VINCE (Gordon et al., 2020)) or data transformation/augmentation (see SimCLR), which will be elaborated on later.

An interesting example for contrastive learning is also presented with BYOL (Bootstrap your own latent, (Grill et al., 2020)), where image representation is learned with no use of negative examples or image pairs but only positives while still wielding similar or better results than other state of the art methods.

It is however not fixed that learning has to specifically work with pairs only. Depending on the loss that is used, learning can take place with triplets as well. Triplets represent, as the name implies, not a pair of images but a convolute of three images containing the anchor image as well as a positive and negative example image.

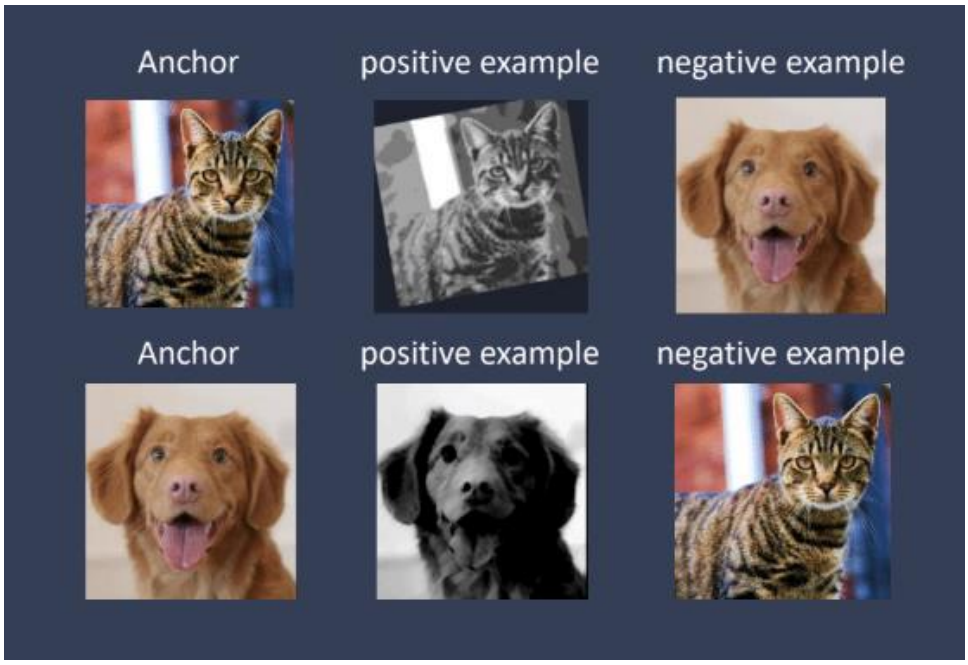

Figure 2 - Visualization of triplets with anchor images as well as positive and negative keys

In a sense, the use of triplets makes the learning of a positive image pair and a negative image pair happen simultaneously. Chechik et al. (Bishop, 2006) make use of triplet loss in their training method.

3.2.1 Data Augmentation (Marc)

Data augmentation is one way to create positive comparison examples of an anchor datapoint. Augmentations include for example:

- mirroring
- color adjustments
- cropping
- rotation
- blurring
- resizing/resolution adjustments

The augmentation is used on the anchor image itself, creating a necessarily similar image to the anchor. One advantage of using augmentation in this step is that it can simply be used on any image dataset without the need for manual research of positive example images, making it well suited for unsupervised learning. Optionally, the anchor image can be augmented as well. Aforementioned BYOL (Grill et al., 2020) as well as SimCLR use the same data augmentation on both their anchor image as well as their positive example image. It is also possible to apply augmentation randomly,

meaning that all augmentations used on a picture can have a certain probability to be applied.

It is important to note that the choice on augmentations depends heavily on the used dataset since some augmentations tend to be more useful for some datasets than others, with some augmentations even rendering to be completely useless for specific datasets. For example, experiments in SimCLR showed that the composition of random cropping and random color distortion is crucial for good performance on learning visual representation of images. (Weng, 2021a) A negative example for image augmentation would be mirroring images of letters or numbers, like the MNIST dataset, since then the information and shape of the supposed number is lost. Our idea is to keep essential/necessary information alive while augmenting and adjust remaining parameters.

3.3 Siamese Neural Network Architecture (Anika)

Neural networks can be used for a variety of tasks, but they rely on the amount of data to perform well. For some problems, it is not easy to obtain more data, for instance in face recognition. To solve this issue, the network architecture *"Siamese Networks"* (SNN) can be used. It is able to use small amounts of images, and still get valuable predictions. The idea of this architecture is to use two or more identical subnetworks. They use the same parameters and weights, and the updating is copied across all subnetworks to determine the similarity of inputs by feature vector comparison. This poses the following advantage: Usual neural networks contain the problem of class addition and removal, since they learn to predict multiple classes. The whole network needs to be updated and retrained when adding or removing a class. Because SNNs learn a similarity function, they can be trained to check if two images are the same. Therefore, SNN can classify new classes of data points without retraining the network. (Benhur, 2020)

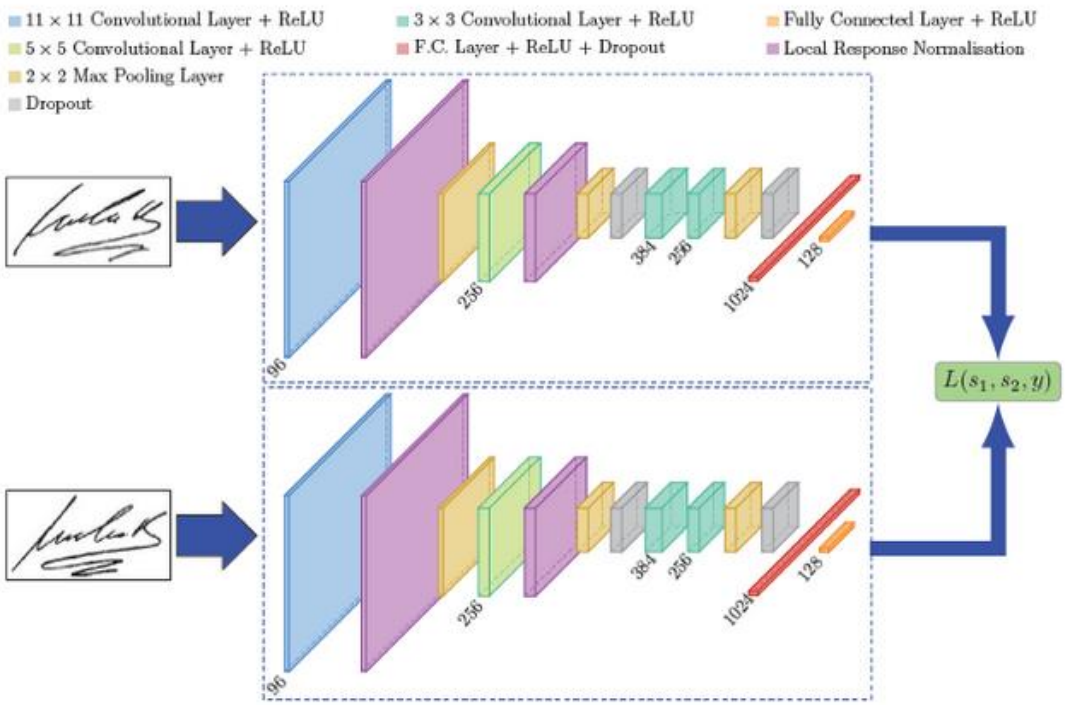

Figure 3 - Siamese Neural Network architecture

3.4 Loss functions

To maximize the similarity between in pairs, the loss between them must be minimized. For that task, loss functions are needed. In the following this paper will explain four different contrastive loss functions and the respective application. We will also argue why we used which of the presented loss functions. (Anika)

3.4.1 Margin-Loss (Robin)

The term contrastive learning has been introduced by (Hadsell et al., 2006) for the purpose of dimensionality reduction. They describe goal of a contrastive loss function as to learn the parameters of a function ‘in such a way that neighbors are pulled together and non-neighbors are pushed apart’. (Hadsell et al., 2006, p. 2) This can be seen as the general goal of all contrastive loss approaches, therefore finding common features of similar data points. As an advantage over the common dimensionality reduction method of Principal Component analysis (PCA) they mention better generalization and applicability to new data: “It learns a function that is capable of consistently mapping new points unseen during training.”

New about the contrastive loss function was to consider learning elements as pairs of samples X1 and X2, that are either similar (Y = 0), or dissimilar (Y = 1) The loss

function Gw with parameters W for a set of training vectors xi therefore described by Hadsell et al. is as follows:

$$\mathcal{L}(W) = \sum_{i=1}^{P} L(W, (Y, \vec{X}_1, \vec{X}_2)^i) \qquad (2)$$

$$L(W, (Y, \vec{X}_1, \vec{X}_2)^i) = (1 - Y) L_S \left(D_W^i\right) + Y L_D \left(D_W^i\right) \qquad (3)$$

For a number of p training pairs, the loss is calculated by taking the Euclidian distance between the pairs (DW) into account. Ls describes the partial loss function for a pair of similar points, while LD describes the partial loss function for a dissimilar pair. For optimizing the weights (= minimizing the loss function), the loss function should therefore achieve low values in the distance of similar pairs (Ls). LD must be constructed in a way that returns high values for dissimilar pairs:

$$L(W, Y, \vec{X}_1, \vec{X}_2) = (1 - Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\{max(0, m - D_W)\}^2$$

For LD (right part of the equation), a margin is introduced, used to penalize cases in which the distance between dissimilar points is smaller than the margin. This is supposed to create an ‘equilibrium’ that keeps similar points together, while pushing dissimilar points away, ideally ending up in individual clusters for data points with similar attributes. (Hadsell et al., 2006, p. 3f) The concepts behind this loss are present as well in the upcoming described losses.

3.4.2 Cosine Embedding Loss (Marc)

The Cosine Embedding Loss is a contrastive loss that makes use of cosine distance between two feature vectors with different loss calculations based on if the vectors are supposed to be similar or dissimilar to each other. Cosine distance, also called cosine similarity, is a measurement to show how distinctly two vectors point in the same direction and ranges from -1 when completely opposite to 1 when exactly the same.

In Python, this loss function can be imported via “torch.nn.CosineEmbeddingLoss” (PyTorch, 2019) and is defined as follows:

$$\text{loss}(x, y) = \begin{cases} 1 - \cos(x_1, x_2), & \text{if } y = 1 \\ \max(0, \cos(x_1, x_2) - \text{margin}), & \text{if } y = -1 \end{cases}$$

As input, the function accepts two input tensors x1 and x2 as well as an indicator y, which is 1 if both vectors are supposed to be similar and -1 if they should be dissimilar. Additionally, this loss makes use of a margin, which should logically also range between -1 and 1 just like the minimum and maximum values for cosine similarity. The implementation in Python accepts further inputs to specify if the many loss outputs per batch should be returned individually, summed up or (per default) averaged.

3.4.3 Triplet Margin Loss (Robin)

As described in 3.2, one approach of defining similar and dissimilar samples is to create triplets of images. This setup allows to define a loss, that explicitly encourages both similar instances staying together, and dissimilar instances being pushed apart.

An early approach of formulating a loss for triplets has been made by Chechik et al. (2010, p. 1112), who also adapted the concept of using a margin, that defines how far the dissimilar points should at least be from similar points. This is picked up for the usage with deep convolutional networks by (Wang et al., 2014), using the Euclidian distance two measure similarity / distance between images.

Defining a triplet as $t_i = (X_i, X_i^+, X_i^-)$ consisting of i-indexed triplets of anchor image, positive image and negative image, the triplet margin loss is defined as follows:

$$L((W, X_i, X_i^+, X_i^-) = \max(0, m + D(X_{iw}, X_{iw}^+) - D(X_{iw}, X_{iw}^+)$$

Looking at a singular triplet, there will therefore be a loss, if the distance between anchor and positive ($D(X_i, X_i^+)$ plus, the margin is not smaller than the distance between anchor and negative. Training the weights W of a network to minimize this loss, should therefore bring positive samples close together, and push negative samples away at the same time.

3.5 Problems of Unsupervised Contrastive Learning (Robin)

In the completely unsupervised setting of contrastive learning, labels are not known. Negative pairs are created by randomly subsetting observations from the dataset. Consequently, it can happen that the artificially constructed negatives are in fact

similar – they are ‘false negatives’. (Chuang et al., 2020) refer to this phenomenon as sample bias.

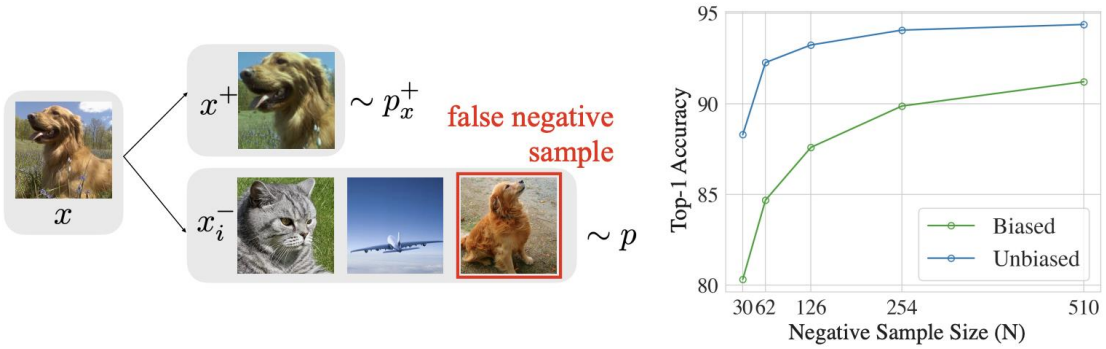

Figure 4 - Sample bias (Chuang et al., 2020)

They show that sample bias reduces the performance of contrastive learning in comparison to a scenario where negative pairs are always ‘true negatives’. They propose a setup of “Debiased Contrastive Loss” that prevents this problem even in the unsupervised setting. (Chuang et al., 2020, p. 3f) The details of this approach are out of scope for this report. Another solution for avoiding sample bias is to take the labels of the data into account. One way of doing this is using the labels for creation of positive and negative pairs, while keeping the training of contrastive loss unsupervised. Another way of avoiding sample bias uses the labels while training calculating the contrastive loss; actively distinguishing between instances from the same and other classes. This adapts the approach of contrastive learning to the supervised setting.

3.6 Supervised Contrastive Learning (Robin)

(Khosla et al., 2021) describe a supervised approach for contrastive learning: The labels of the pictures are taken into account when optimizing for the embedding space.

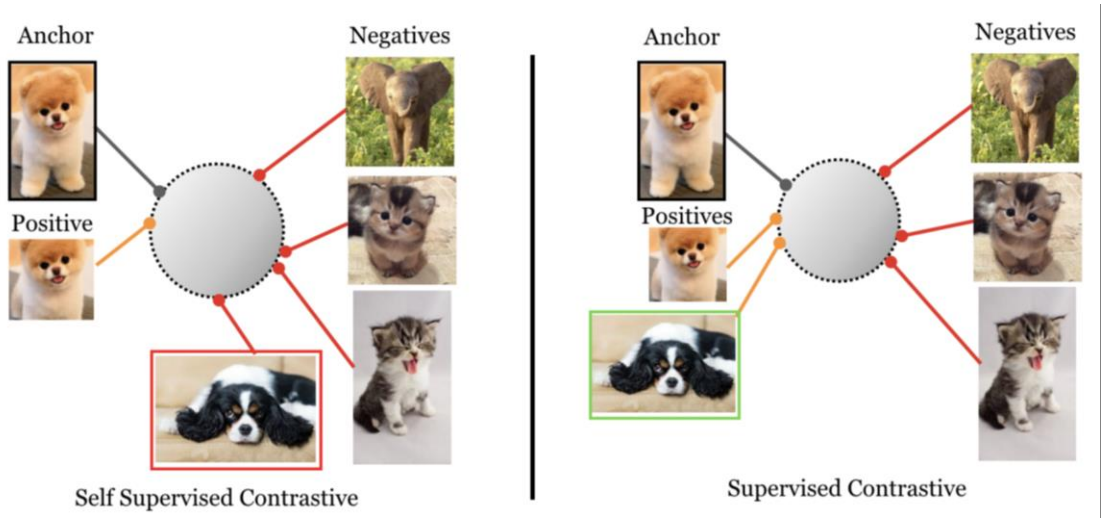

Figure 5 - Supervised vs. self-supervised contrastive losses: In the supervised setting, all samples from the corresponding class as the positive pair are contrasted against the remaining negatives of a batch. (Khosla et al., 2021, p. 2)

This is realized by using a loss function ‘SubCon’, that can take many positives and many negatives into account. Unlike in the previous approaches, positives are therefore not exclusively variations of the same picture, but also pictures with identical labels, coming from the same class. It can therefore be seen as a generalization of the triplet loss.

Another difference to the previously described losses is that all pictures of a batch are taken into account together to calculate a loss, whereas for the margin loss approaches, only two or three images are taken into account at a time. This potentially allows for a much higher degree of generalization and “contrastive power”. (Khosla et al., 2021, p. 5)

4 The Data Sets

Before discussing the actual implementation, I closer look at the data used is needed, since it conducts the base for the outcomes. For the implementation, we chose the CIFAR-10 and MNIST data set. Although the advantage of contrastive learning is the usage on unlabeled data, we decided to use already labeled data to be able to clarify at the end of the training which data augmentations and loss functions worked better. So, the focus does not lay in the actual use case of contrastive learning, but in the comparison of the different implementations. (Anika)

4.1 CIFAR-10 (Anika)

The CIFAR-10 dataset is a subset of the 80 million tiny images dataset[1]. It was compiled by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. It consists of 10 classes with each 6000 images, each 32x32. In Fig. 5, the classes of the dataset and ten random pictures are shown.

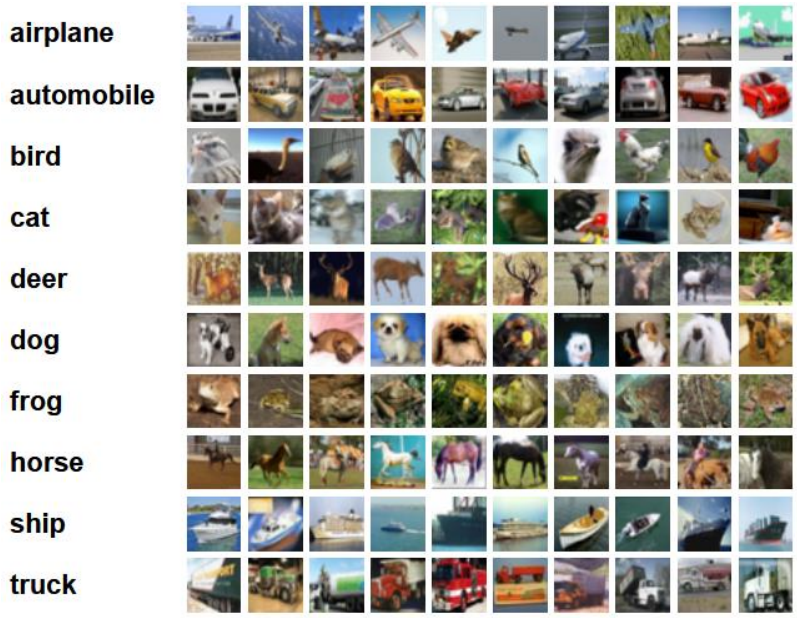

Figure 6 - Classes from CIFAR-10 with each 10 examples

The classes are mutually exclusive, meaning that there is for instance no overlap for automobile and trucks. (Krizhevsky, n.d., n.p.)

[1] https://groups.csail.mit.edu/vision/TinyImages/

When looking at the individual classes, they can be sectioned again in animals and means of transport. To us, that posed the advantage of seeing if our model would be able to recognize the differences between the two apart from the actual classes.

From the 60,000 images, 50,000 are for training and 10,000 for testing. 10,000 pictures build a batch so that the dataset consists of five training batches and one test batch. For the test batch, 1000 random pictures from each class were selected. The remaining pictures are allocated to the training batches in random order, which results in some training batches containing more images from one class than another. (Krizhevsky, n.d., n.p.)

4.2 MNIST (Marc)

The second and widely popular MNIST dataset is a collection of handwritten digits 0 to 9 with 70,000 images in total, 60,000 images for training and 10,000 for testing. It was created by Yann LeCun, Corinna Cortes and Christopher J.C. Burges as a subset of the NIST handwriting dataset by normalizing the single black and white characters into a greyscale 20x20 pixel box and finally center them into a 28x28 image.

Figure 7 - MNIST data set examples

As the image attributes imply, this dataset is not as complex as CIFAR10 and does not contain as much information in one image as in an image of the latter, making preprocessing easier as well. Same as for CIFAR10, all classes are mutually exclusive as there are no multiple classifications for one number image.

5 Implementation (Nathalie)

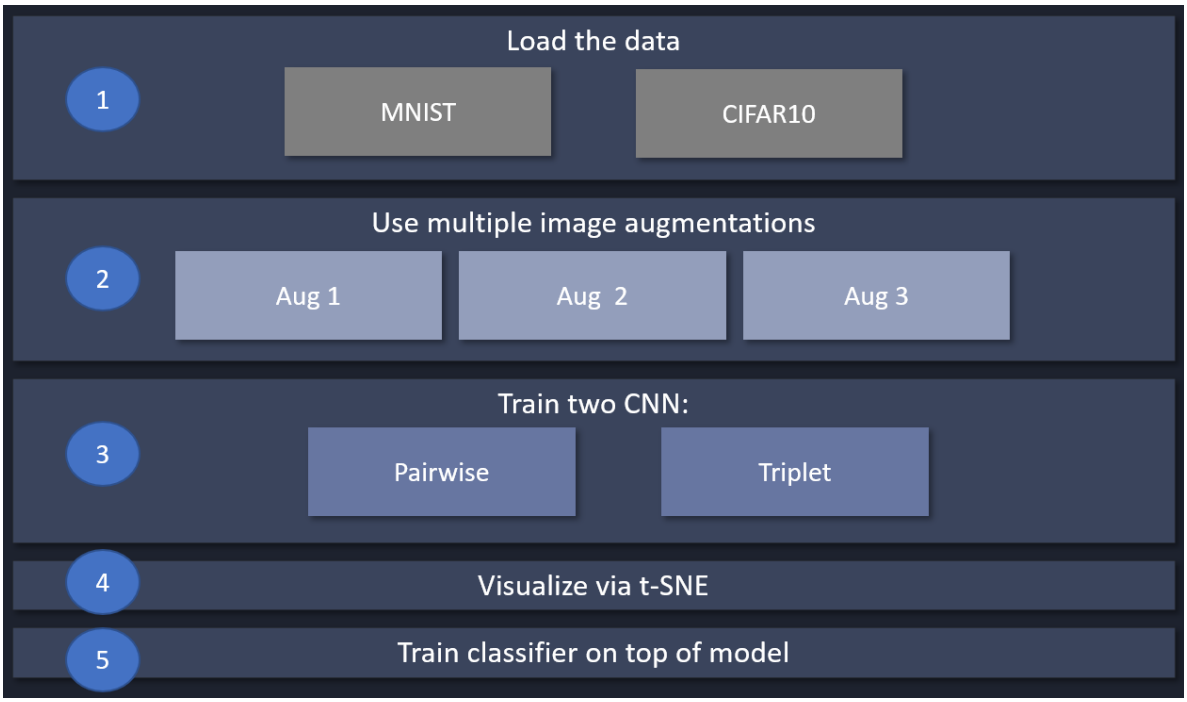

Figure 8 - Research Design

In our exercise, we want to make a comparison of performance between pairwise and triplet training using the same prerequisites. The experimental setup consists of five steps which will be explained in the following paragraphs. In the following, the steps 1-5 marked on Fig. 8 will be described on a conceptual level, to further support an intuition for how contrastive learning works.

5.1 Data Loading

We will focus on contrastive learning for images in this exercise. First the data needs to be chosen and loaded. We decided to use the datasets MNIST and CIFAR-10, because they are freely available and both integrated datasets of PyTorch, which can be found in the collection of datasets in the package torchvision. Another strong argument for those two datasets was that we wanted to focus on the comparison of the effects from different augmentations and to compare different loss functions in the pairwise and triplet training and didn't want to put our focus on preparing and acquiring data. Another factor is also that both datasets contain ten classes, so we could reuse our code.

5.2 Image Augmentation

Data transformation is what we decided to use for our exercise as a positive example generator. In Python, the augmentation can be done by using “transforms” in the “torchvision” package. We decided to use three different augmentations and want to see the effect of gradually increasing the strength of augmentation on our training results. The first augmentation should serve as a baseline for the other augmentations and does only apply normalization to the images and converts it into a tensor. The second augmentation is taken from the paper *“Deep Neural Networks with Relativity Learning for Facial Expression Recognition”* and applies to the image rescaling, padding and random rotation and scales the image to a uniform size (Yanan Guo et al., 2016). This augmentation was interesting for us because it is a simple augmentation which is useful for different datasets and the augmentation isn’t too strong either. The third augmentation is used in the SimCLR paper *“A Simple Framework for Contrastive Learning of Visual Representations”,* and applies certain stochastic augmentations which are random cropping and resizing back to original image size, random color distortion and random gaussian blur as well as a random horizontal flip (Chen et al., 2020). We found this augmentation useful because it is a stronger augmentation and useful for different types of datasets.

Due to the nature of Contrastive Learning, the data needs to be restructured for each type of training. For pairwise training, we need to form tuples for anchor-positive and anchor-negative pairs respectively. For triplet training we need to form triplets instead containing an anchor image, a positive example, and a negative example. In both trainings we decided for the positive example to use the augmented version of the anchor image because it would make the loss calculation would be beneficial for separating from negative examples and can assign the class to different versions of the same picture.

Because both of our datasets include labels with images, we used this type of clustering for creating negative examples to the anchor image with simply using an image of a different class. (Vijayrania, 2020). We generate the pairs using lists for each class of the dataset for the anchor, positive and negative images making use of the labels. It is also important to mention that we avoid the problem of sample bias (see chapter 3.6) by creating the negative pairs regarding the class information; so that negative samples always come from a different class than the positives. The fact that images

in the lists are in the same order as the classes, we need to shuffle the images for the negative examples so that the training is more diverse.

The data for testing is not augmented and doesn't need to get processed further for using it for testing the classifier later.

5.3 Network Structure

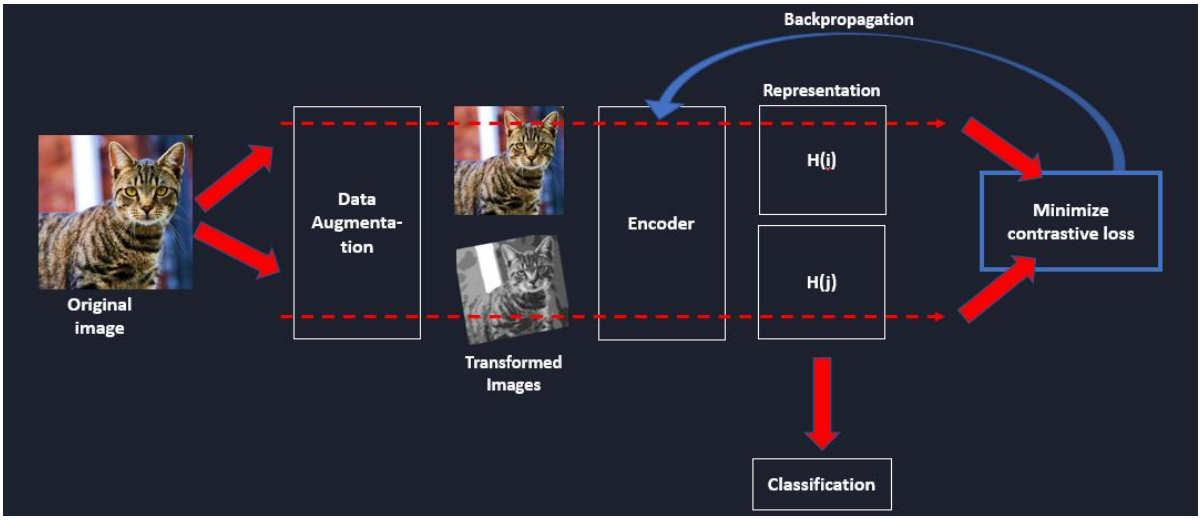

Figure 9 - Network Structure

This graphic shows the different steps of contrastive learning framework that we have used in our network structure. While it is inspired by the Illustrated SimCLR Framework (Chaudhary, 2020), its main components can be found in most of the approaches for supervised learning used for images.

To perform contrastive learning, we decided to use a sequential network with layers of several two-dimensional convolution layers with batch normalization and leaky ReLu before the input gets passed through a sigmoid function and gets flattened for the calculation of the losses. We decided for a batch size of 1024 because of the good performance SimCLR had with high batch sizes and for 10 epochs. We decided specifically for this batch size, because if we trained for more epochs, the loss didn't improve much. For pairwise training, we chose a learning rate of 0.05 and for triplet training a learning rate of 0.15. The reason behind this is that we saw the loss functions performed drastically different. For loss functions we decided to use Cosine Embedding Loss for pairwise training and Triplet Margin Loss for triplet training because they are part of PyTorch and are sufficient for our use case. Another reason was that we didn't need to implement a custom defined loss function which would need further

tweaking and might have compatibility issues if we changed the way the training is done.

The structure of the network is shown in the table below.

Layer (type)	Output Shape	Number of Parameters
Conv2d-1	[-1, 32, 15, 15]	128
LeakyReLU-2	[-1, 32, 15, 15]	0
Conv2d-3	[-1, 64, 8, 8]	8,192
BatchNorm2d-4	[-1, 64, 8, 8]	128
LeakyReLU-5	[-1, 64, 8, 8]	0
Conv2d-6	[-1, 128, 5, 5]	32,768
BatchNorm2d-7	[-1, 128, 5, 5]	256
LeakyReLU-8	[-1, 128, 5, 5]	0
Conv2d-9	[-1, 256, 3, 3]	131,072
BatchNorm2d-10	[-1, 256, 3, 3]	512
LeakyReLU-11	[-1, 256, 3, 3]	0
Conv2d-12	[-1, 8, 2, 2]	8,192
Sigmoid-13	[-1, 8, 2, 2]	0
Flatten-14	[-1, 32]	0

In the beginning there are multiple convolutional layers paired with leaky ReLu functions. In after the second up to the second-to-last convolutional layer we decided to put batch normalization layers. After the last convolutional layer follows a Sigmoid layer and we flatten the output for further processing. We decided to use Leaky ReLu with a negative slope of 0.2 because of the small size of the images which are present in both datasets used. The structure is oriented on different papers such as (Yanan Guo et al., 2016) VGG19, ResNet50 (He et al., 2015) and SegNet (Badrinarayanan et al., 2017.), but our model is a lot more toned down and doesn’t go that much in depth because we were trying different models. However, the accuracy didn’t improve much and had to deal with longer computation times which gave us more negative than positive effects.

5.4 Visualization

After training is finished, the results of the contrastive training can be visualized. We decided that for our project t-SNE would be well suited, as for it is the goal of contrastive learning to separate classes by similarity. For pairwise training and triplet training we display the output from the network given the input of the test dataset. To avoid complexity and longer calculation time we limited t-SNE to display the first 1000 points.

5.5 Classifier

With the results received by training the network we can't measure any accuracy yet and therefore need to implement a classifier. For the classification task to work, we train a Logistic Regression Classifier with the library Scikit-learn, so that it is possible to calculate the accuracy of prediction.

As Bengio et al. (2014) describe, "in good high-level representations, the factors are related to each other through simple, typically linear dependencies. This can be seen in many laws of physics, and is assumed when plugging a linear predictor on top of a learned representation" Therefore, to quantitatively evaluate the quality of the different contrastive representations, we are able to train a linear classifier. We decided to use the classification algorithm 'Logistic Regression'.

In addition to traditional measurement of the achieved results from the classifier e.g., accuracy, we also calculated a top-3/top-n accuracy and visualized the accuracy for each class through a confusion matrix.

6 Evaluation

6.1 Results (Anika)

As discussed above, t-SNE can help to evaluate the results of unsupervised learning techniques. For our experiment, we generated t-SNE for each scenario, so six per data set, with three for each pairwise and triplet loss, representing the different augmentations implemented. On Fig. 10, the results for MNIST and pairwise are displayed. The used Cosine Embedding Loss led to line-like structures in the clustering. It can be observed that the pad, rotate and scale augmentation seems to have the best result, against the common assumption that heavy augmentation leads to better results. (Weng, 2021) Interestingly, the number 8 is completely scattered when using the SimCLR augmentation. A possible explanation for the better performance of Augmentation (B) in this scenario could be the simplicity of the data set, as well as the different variants of each number already given. The filter might “overdo” the altering, leading to the loss of recognizable features needed for clustering. Augmentation (C) does contain a horizontal flip, which likely led to the loss of essential features to recognize the numbers.

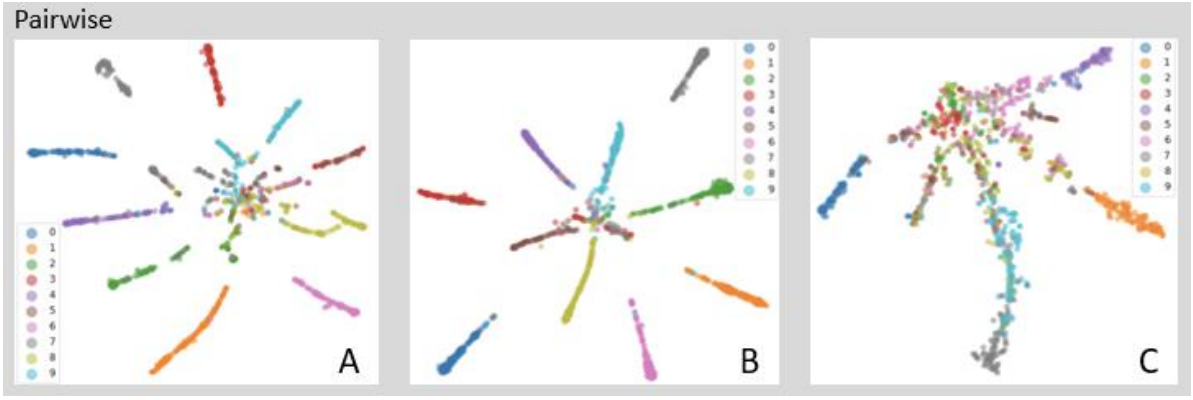

Figure 10 - t-SNE for Pairwise and MNIST: (A)No augmentation, (B)Pad, rotate and scale, (C)SimCLR Augmentation

Fig. 11 shows the results for MNIST and triplet loss. Here a clear improvement in relation to the amount of augmentation can be observed – It seems like heavier augmentation helped the quality of the model. In (C), an almost perfect class separation is displayed. This is interesting, since Filter (C) does contain the horizontal flip, which led to poor results in the scenario above. It seems like the inclusion of a negative example (triplet loss) does compensate for that obstacle.

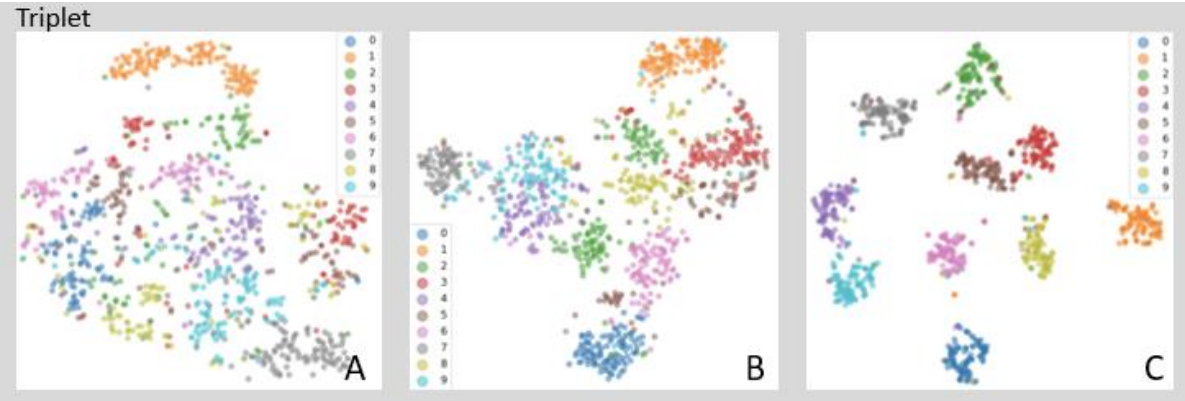

Figure 11 - t-SNE for Triplet and MNIST: (A)No augmentation, (B)Pad, rotate and scale, (C)SimCLR Augmentation

On Fig. 12 below, the t-SNE for the CIFAR10 data set and Pairwise Loss are shown. A clear difference between the results on MNIST can be observed, the clustering here is not as clear, which indicates a worse accuracy. Nevertheless, the SimCLR Augmentation led to the clearest picture: The classes blue (airplane), orange (automobile) and violet (deer) are clustered relatively well.

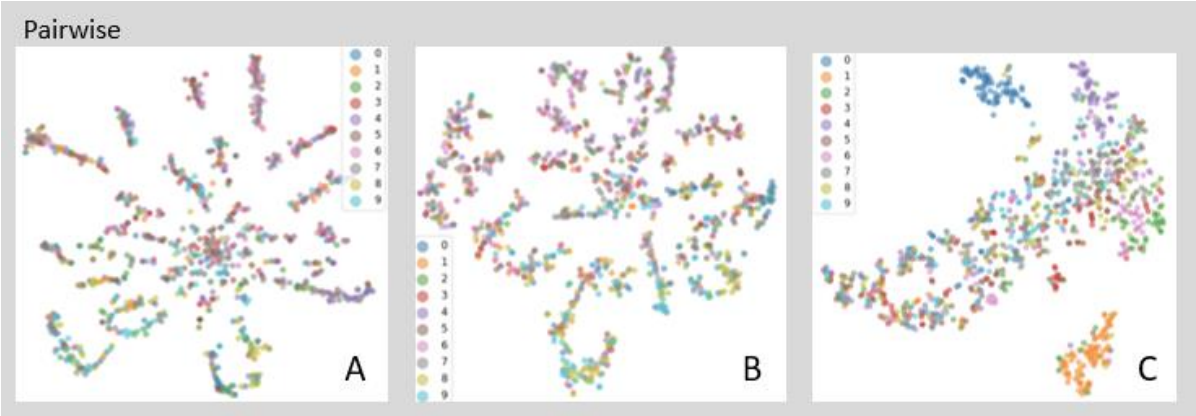

Figure 12 - t-SNE for Pairwise and CIFAR10: (A)No augmentation, (B)Pad, rotate and scale, (C)SimCLR Augmentation

Fig. 13 is the last t-SNE Chart. Training with triplet loss on CIFAR10 unfortunately did lead to quite poor results; at least no clear clustering can be observed. Although, it seems like the clustering is getting slightly better with increasing augmentation level.

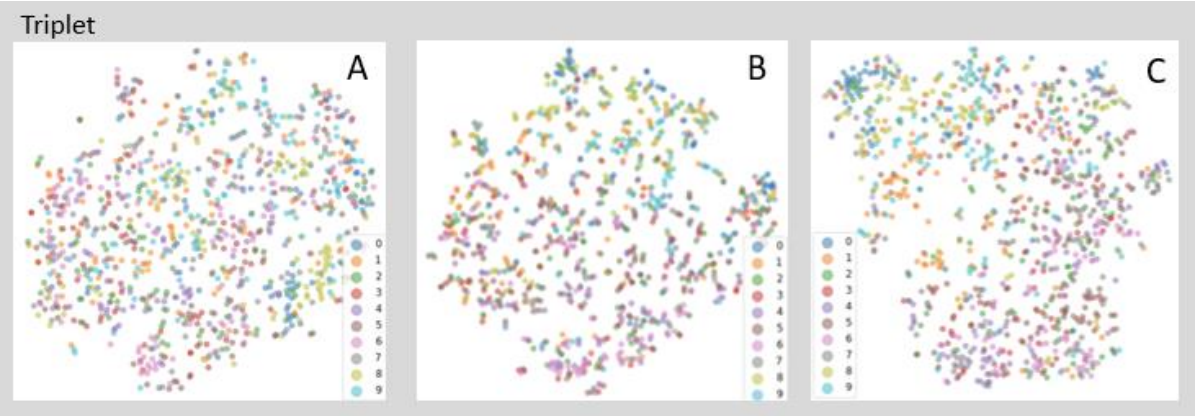

Figure 13 - t-SNE for Triplet and CIFAR10: (A)No augmentation, (B)Pad, rotate and scale, (C)SimCLR Augmentation

As explained in Chapter 5.3, t-SNE can indicate on how the well the network did. To quantify results, we implemented a classifier on top of the model, which uses the labels of the data sets. Even though this is practicably against the original thought of contrastive learning, we wanted to be able to draw a numerical comparison between all variants of our experiment.

On Fig. 14, a matrix displays the relationship of true and predicted labels when training with SimCLR Augmentation and Triplet loss on MNIST. To explain the meaning of the numbers: The original 10,000 training images of the data set had to be split again into training and testing for the implementation of the classifier. Therefore, the test set of the classifier contained 2,000 images in total, so 200 per class. So the sums of each row (and also column) sum up to 200, which is also the best result achievable. It can be observed that very little misclassifications happened, which explains the accuracy of 96%.

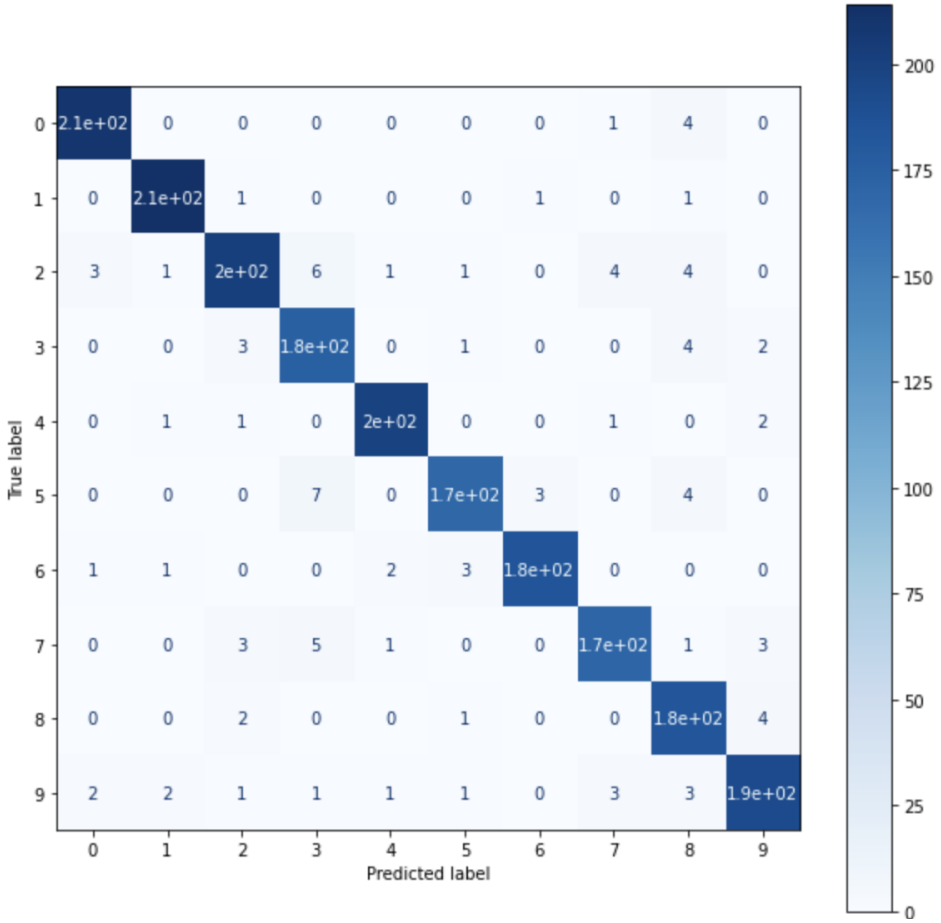

Figure 14 - Results of training with SimCLR Augmentation and Triplet Loss on MNIST

As the unprecise clustering on the t-SNE indicated, the results when training on CIFAR10 are not as well as when training on MNIST. Looking at Fig. 15, the model classified many test pictures wrong. The best results were achieved for airplane, automobile, frog, ship, and truck. A possible explanation for that could be that the model struggled classifying animals from each other, except for frog. A reason for that could be the clearly different shape and color frogs have, and that they are probably also photographed in different surroundings (water, mud, leaves). Birds, cats, deer, dogs, and horses do have somehow similar shapes and are probably photographed with related backgrounds (forest, house, town).

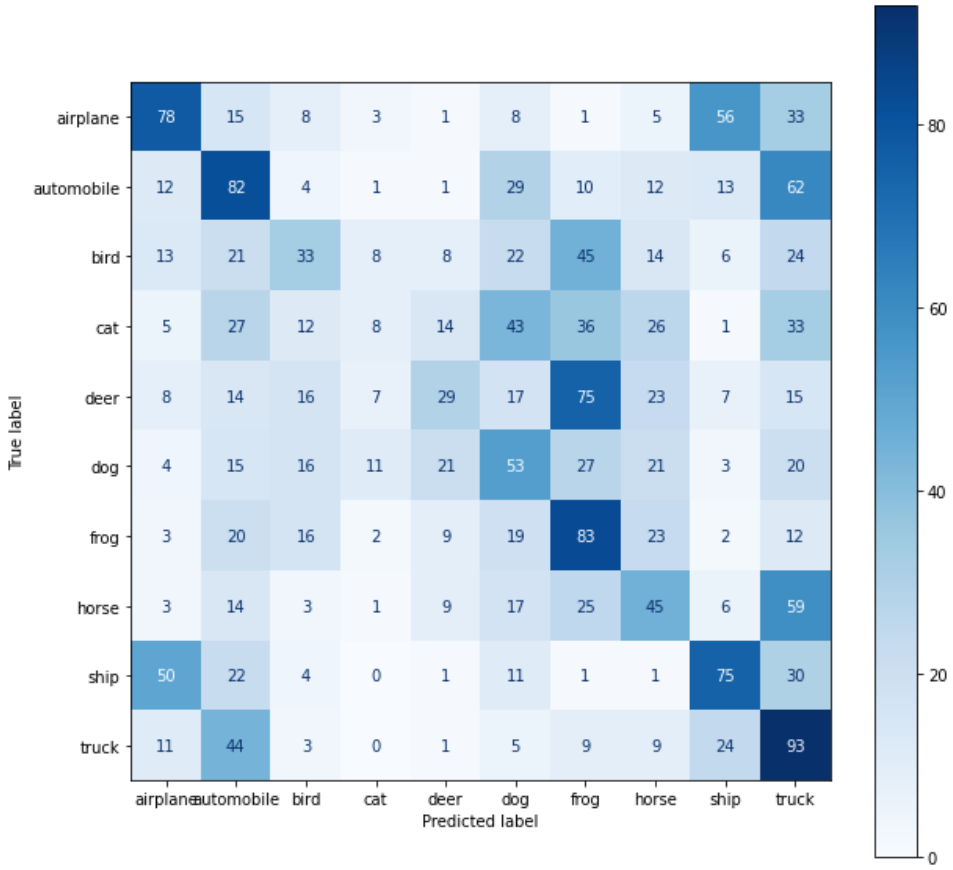

Figure 15 - Results of training with SimCLR Augmentation and Triplet Loss on CIFAR10

6.2 Limitations (Anika & Robin)

In the following, we want to discuss the limitations of our project.

This first and most obvious limitation is that our approach does only work for datasets with labels: That goes for the triplet and negative pair creation, but also for the classifier. Since the negative examples would have to be chosen randomly, doing contrastive learning without labels would lead to false pairs and therefore sample bias as described in 3.5. Since we have used datasets with only 10 classes, the sample bias would be quite high and probably diminish the contrastive learning.

Secondly, we used datasets that were processable via google collab notebooks. Therefore, our current approach is implemented on a rather small scope. A comparison with more losses and bigger datasets would be possible but would need more computational resources to realize. Our findings may therefore not be completely transferrable to datasets that differ for example in domain, size and number of classes.

The performance of models may have been beeinträchtigt bei the somewhat arbitrary and random selections of positive / negative pairs and triplets. While more recent approaches take a lot of effort into carefully selecting / mining for triplets that are particularly hard to train (e.g. Quelle einfügen); we are training our models on the same set of randomy selected samples from the same / other classes. This may have lead to an overly quick reduction of the loss without learning meaningful features.

Another limitation of our experiment are the fixed augmentations used. Although this approach made sense when it comes to the comparison of the results, it practically led to partly poor outcomes. The chosen augmentation should always be adapted to the nature of data so that features to distinguish the classes remain.[2] For instance, the zoom on pictures can lead to losing the actual object that should be recognized. Another example is the color alterations on animals, which for example would harden the classification of different bird breeds.

7 Conclusion and Outlook (All)

Our project was supposed to show the impact of different image augmentations and data sets on contrastive learning. As we saw in chapter 6.1; heavier augmentation did not lead to better results. The importance of a fit between data set and augmentation became clear, since relevant features necessary for classification must not get lost during the image altering. When comparing the results of MNIST and CIFAR10 scenarios, it became clear that the complexity of the data set does impact the outcome as well: MNIST led to overall better accuracy (no matter the augmentation) than CIFAR10. (Anika)

Furthermore, differences between pairwise and triplet training illustrate the impact of augmentation for contrastive learning. Especially triplet training benefits from stronger augmentation on the positive key image, which supports verification of the SimCLR research results for the choice of image transformation. Our exercise could be extended with additional augmentation on the anchor images and compare results. Even though the network training itself was unsupervised with no usage of labels, we optimized our training data by creating supervised negative key images

[2] If the goal is a high accuracy!

since we know that all of these images are not of the same class as the anchor image. Another research extension could be to assign negative keys completely at random from the dataset as a fully unsupervised approach to examine the risk of having an image of the same class as a negative key and the consequential impact on learning. (Marc)

Note Robin: Few sentences about learning experience; trying a lot and failing at first;

8 Appendix

The code and files used for this paper can be found on a publicly available GitHub:

https://github.com/lunoon/PAMI_Contrastive_Learning/

Literature

Albawi, Saad, Tareq Abed Mohammed, und Saad Al-Zawi. 2017. „Understanding of a convolutional neural network“. In *2017 International Conference on Engineering and Technology (ICET)*, 1–6. Antalya: IEEE. https://doi.org/10.1109/ICEngTechnol.2017.8308186.

Badrinarayanan, Vijay, Alex Kendall, und Roberto Cipolla. 2017. „SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation“. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* 39 (12).

Bengio, Yoshua, Aaron Courville, und Pascal Vincent. 2014. „Representation Learning: A Review and New Perspectives“. *arXiv:1206.5538 [cs]*, April. http://arxiv.org/abs/1206.5538.

Benhur, Sean. 2020. „A friendly introduction to Siamese Networks“. https://towardsdatascience.com/a-friendly-introduction-to-siamese-networks-85ab17522942.

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. Information Science and Statistics. New York: Springer.

Chaudhary, Amit. 2020. „The Illustrated SimCLR Framework“. Amit Chaudhary. 4. März 2020. https://amitness.com/2020/03/illustrated-simclr/.

Chechik, Gal, Varun Sharma, Uri Shalit, und Samy Bengio. 2010. „Large Scale Online Learning of Image Similarity Through Ranking“. *J. Mach. Learn. Res.* 11 (März): 1109–35.

Chen, Ting, Simon Kornblith, Mohammad Norouzi, und Geoffrey Hinton. 2020. „A Simple Framework for Contrastive Learning of Visual Representations“. *arXiv:2002.05709 [cs, stat]*, Juni. http://arxiv.org/abs/2002.05709.

Chi, Zewen, Li Dong, Furu Wei, Nan Yang, Saksham Singhal, Wenhui Wang, Xia Song, Xian-Ling Mao, Heyan Huang, und Ming Zhou. 2021. „InfoXLM: An Information-Theoretic Framework for Cross-Lingual Language Model Pre-Training“. *arXiv:2007.07834 [cs]*, April. http://arxiv.org/abs/2007.07834.

Choi, Hongjun, Anirudh Som, und Pavan Turaga. 2020. „AMC-Loss: Angular Margin Contrastive Loss for Improved Explainability in Image Classification“. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 3659–66. Seattle, WA, USA: IEEE. https://doi.org/10.1109/CVPRW50498.2020.00427.

Chuang, Ching-Yao, Joshua Robinson, Lin Yen-Chen, Antonio Torralba, und Stefanie Jegelka. 2020. „Debiased Contrastive Learning“. *arXiv:2007.00224 [cs, stat]*, Oktober. http://arxiv.org/abs/2007.00224.

Erdem, Kemal. 2020. „t-SNE clearly explained“. *Towards Data Science*, 13. April 2020. https://towardsdatascience.com/t-sne-clearly-explained-d84c537f53a.

Favory, Xavier, Konstantinos Drossos, Tuomas Virtanen, und Xavier Serra. 2020. „COALA: Co-Aligned Autoencoders for Learning Semantically Enriched Audio Representations“. *arXiv:2006.08386 [cs, eess, stat]*, Juli. http://arxiv.org/abs/2006.08386.

Freeman, Bill, Rob Fergus, und Antonio Torralba. 2020. o. J. „80 million tiny images dataset“. https://groups.csail.mit.edu/vision/TinyImages/.

Gordon, Daniel, Kiana Ehsani, Dieter Fox, und Ali Farhadi. 2020. „Watching the World Go By: Representation Learning from Unlabeled Videos“. *arXiv:2003.07990 [cs]*, Mai. http://arxiv.org/abs/2003.07990.

Grill, Jean-Bastien, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, u. a. 2020. „Bootstrap your own latent: A new approach to self-supervised Learning“. *arXiv:2006.07733 [cs, stat]*, September.

http://arxiv.org/abs/2006.07733.
Grover, Aditya, und Jure Leskovec. 2016. „node2vec: Scalable Feature Learning for Networks“. *arXiv:1607.00653 [cs, stat]*, Juli. http://arxiv.org/abs/1607.00653.
Hadsell, R., S. Chopra, und Y. LeCun. 2006. „Dimensionality Reduction by Learning an Invariant Mapping“. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)*, 2:1735–42. New York, NY, USA: IEEE. https://doi.org/10.1109/CVPR.2006.100.
He, Kaiming, Xiangyu Zhang, Shaoqing Ren, und Jian Sun. 2015. „Deep Residual Learning for Image Recognition“. *arXiv:1512.03385 [cs]*, Dezember. http://arxiv.org/abs/1512.03385.
Hjelm, R. Devon, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, und Yoshua Bengio. 2019. „Learning deep representations by mutual information estimation and maximization“. *arXiv:1808.06670 [cs, stat]*, Februar. http://arxiv.org/abs/1808.06670.
Howard, Jeremy, und Sylvain Gugger. 2020. *Deep Learning for Coders with Fastai and PyTorch: AI Applications without a PhD*. Beijing Boston Farnham Sebastopol Tokyo: O'Reilly.
IbPRIA 2009, und IbPRIA 2009, Hrsg. 2009. *Pattern Recognition and Image Analysis: 4th Iberian Conference, IbPRIA 2009 Póvoa de Varzim, Portugal, June 10-12, 2009 Proceedings*. Lecture Notes in Computer Science (Internet) 5524. Berlin Heidelberg: Springer Berlin Heidelberg Springer e-books.
James, Gareth, Daniela Witten, Trevor Hastie, und Robert Tibshirani, Hrsg. 2013. *An introduction to statistical learning: with applications in R*. Springer texts in statistics 103. New York: Springer.
Kaggle. „Challenges in Representation Learning: Facial Expression Recognition Challenge Learn facial expressions from an image“. 2013. o. J. https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data?select=fer2013.tar.gz.
Khosla, Prannay, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, und Dilip Krishnan. 2021. „Supervised Contrastive Learning“. *arXiv:2004.11362 [cs, stat]*, März. http://arxiv.org/abs/2004.11362.
Krizhevsky, Alex. 2009. o. J. „The CIFAR-10 dataset“. https://www.cs.toronto.edu/~kriz/cifar.html.
Le-Khac, Phuc H., Graham Healy, und Alan F. Smeaton. 2020. „Contrastive Representation Learning: A Framework and Review“. *IEEE Access* 8: 193907–34. https://doi.org/10.1109/ACCESS.2020.3031549.
Maaten, Laurens van der, und Geoffrey Hinton. o. J. „Visualizing Data using t-SNE“. Journal of Machine Learning Research 9 (2008). https://jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf.
Mikolov, Tomas, Kai Chen, Greg Corrado, und Jeffrey Dean. 2013. „Efficient Estimation of Word Representations in Vector Space“. *arXiv:1301.3781 [cs]*, September. http://arxiv.org/abs/1301.3781.
Provost, Foster, und Tom Fawcett. 2013. *Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking*. 1. ed., 2. release. Beijing: O'Reilly.
PyTorch. 2019. *CosineEmbeddingLoss PyTorch 1.9.1 documentation*. Python. https://pytorch.org/docs/stable/generated/torch.nn.CosineEmbeddingLoss.html.
Saha, Sumit. 2018. „A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way“. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.
Schneider, Steffen, Alexei Baevski, Ronan Collobert, und Michael Auli. 2019. „wav2vec:

Unsupervised Pre-training for Speech Recognition“. *arXiv:1904.05862 [cs]*, September. http://arxiv.org/abs/1904.05862.

Sermanet, Pierre, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, und Sergey Levine. 2018. „Time-Contrastive Networks: Self-Supervised Learning from Video“. *arXiv:1704.06888 [cs]*, März. http://arxiv.org/abs/1704.06888.

Shrivastav, Namratesh. 2019. „PCA vs t-SNE: which one should you use for visualization“. *Analytics Vidhya*, 28. September 2019. https://medium.com/analytics-vidhya/pca-vs-t-sne-17bcd882bf3d.

Tiu, Ekin. 2021. „Understanding Contrastive Learning“. *Towards Data Science*, 7. Januar 2021. https://towardsdatascience.com/understanding-contrastive-learning-d5b19fd96607.

Vijayrania, Nilesh. 2020. „Self-Supervised Learning Methods for Computer Vision“. *Towards Data Science*, 15. Dezember 2020. https://towardsdatascience.com/self-supervised-learning-methods-for-computer-vision-c25ec10a91bd.

Wang, Jiang, Yang Song, Thomas Leung, Chuck Rosenberg, Jinbin Wang, James Philbin, Bo Chen, und Ying Wu. 2014. „Learning Fine-grained Image Similarity with Deep Ranking“. *arXiv:1404.4661 [cs]*, April. http://arxiv.org/abs/1404.4661.

Weng, Lilian. 2021. „Contrastive Representation Learning“. *lilianweng.github.io/lil-log*. https://lilianweng.github.io/lil-log/2021/05/31/contrastive-representation-learning.html.

Yanan Guo, Dapeng Tao, Jun Yu, Hao Xiong, Yaotang Li, und Dacheng Tao. 2016. „Deep Neural Networks with Relativity Learning for facial expression recognition“. In *2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, 1–6. Seattle, WA, USA: IEEE. https://doi.org/10.1109/ICMEW.2016.7574736.

Yolyan, Lilit. 2021. o. J. „Review on Self-Supervised Contrastive Learning“. https://towardsdatascience.com/review-on-self-supervised-contrastive-learning-93171f695140.