Evaluation of Similarity Loss on Out of Distribution generalization of Neural Networks

Johan Bakker[1]

Supervisor: Wendelin Böhmer[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
January 25, 2026

Name of the student: Johan Bakker
Final project course: CSE3000 Research Project
Thesis committee: Wendelin Böhmer, David Tax

Abstract

Image recognition is used in a lot of application nowadays, it is used for example in sign recognition in autonomous cars. Neural networks are well-suited for this task and perform well in them, but the networks themselves are not well understood. Sometimes the model learns something else than intended, for example the forest in the background and not the road sign. This behavior is called spurious correlations and this paper investigates if new methods can reduce this behavior in neural networks. The new method that is used in this paper is called similarity loss. The results do indicate that this method does not significantly reduce spurious correlations.

1 Introduction

Image recognition by neural networks is a useful tool that has many applications [1] and is able to have a similar performance on image classification as humans [2] [3]. Neural networks struggle however when coming across new images it did not train on, we call this the Out-of-Distribution (OOD) generalization.

The reason of this degraded OOD generalization can be explained by two phenomena: overfitting [4] and spurious correlations [5]. Overfitting can occur if the network is sufficiently large compared to the size of the training that it can just make a look-up table for every image in the training data without learning identifying features. Spurious Correlations happen when there is not enough variation in the training data to incentivize the model to learn the features you want it to learn [5]. An example of spurious correlations is when a neural network is trained on the dataset Waterbirds [6] where most of the landbirds have a land background and most of the waterbirds have a water background. Then the model misclassifies a landbird with a water background as a waterbird. Spurious correlations in this paper will mean that the network learns on the background image instead of the foreground, the foreground determines the label of the image. The focus of this paper will be on the reduction of spurious correlations to improve the OOD generalization.

To research spurious correlations a dataset is generated where digits from the MNIST dataset are superimposed on a limited number of CIFAR10 background images per class. This new dataset incentives neural networks to use spurious correlations by having little variation in the background CIFAR10 images. New approaches can be tested to see how the network can be invariant to the background.

To reduce spurious correlations the representation of the images in the network can be changed, this representation is called an embedding of the image. In this paper it will refer to the representation of the image in the penultimate layer.

This paper introduces a new loss function, Similarity Loss: this loss encourages the model to bring the embeddings together when samples have the same label. Specifically for this paper it means that embeddings with the same MNIST number are pushed together. In this paper an evaluation of the effect of a similarity loss on the OOD generalization is conducted.

This paper shows that:

- The addition of a similarity loss based on bringing the same labels together does not significantly change the OOD performance of neural networks with cross-entropy loss.
- Varying the weight of the similarity loss does not significantly change the OOD performance of the neural network unless it is taken to extreme values, then it decreases the OOD performance significantly.
- Varying the amount of epochs does not significantly change the OOD performance, only when it is taken to extreme values, then it decreases the OOD performance.

2 Related works

Spurious correlations or shortcut learning is a well-known phenomena in Neural Networks. The network learns features that generalize well in the training and in-distribution dataset but fail to generalize in the out-of-distribution dataset [7]. These features are also known as spurious features. Spurious features are learned by the model because they are computationally less demanding than recognizing the intended features. The model learns the way of least resistance, therefore the name shortcut learning.

The survey performed by Ye et al. [8] gives an overview of approaches to address spurious correlations. This survey distinguishes three different ways of mitigating spurious correlations: the first one is data centric, this approach changes the input of the model to tackle the spurious correlations, the second approach is focused on the representation learning within the model and the last model is post-hoc, this approach addresses the output. The data centric and representation learning are the approaches that have mainly been used as inspiration for this paper.

One of the earlier research on representation learning was the Invariant Risk Minimization (IRM), [9], IRM tries to find a common predictor over multiple environments to become invariant to those environments. The main disadvantage is the difficulty to construct and set-up the different environments. StableNet [10] uses sample weighting to disentangle relevant and irrelevant features. This works well at the cost of computation.

Contrastive learning is an approach that brings embeddings together when they share certain features and pushes them apart when they do not share those features. SimCLR [11] is an application of contrastive loss which yield good performance even when learning unsupervised. Correct-n-Contrast (CnC) [12] uses an Empirical Risk Minimization (ERM) model to predict class labels which are then used with contrastive learning to bring samples together if they have the same class but different predictions and separate samples apart if they have a different class but the same prediction.

Examples of data centric approaches are SOft Data Augmentation (SODA) [13], where the representation of an augmentation of input is compared with the representation of an non-augmented input and the mutual information between

those representations is maximized. MaDi [14] is an approach where distractions are masked such that model generalizes better to new distractions.

Similarity loss is different than before mentioned approaches, it changes the embedding itself based on the labels. While being compatible with cross-entropy loss it tries to decrease the dependence on spurious correlations by bringing embeddings together that share the same label.

3 Methodology

The methodology section is split in three different parts: data generation, model and similarity loss. Data generation will describe how the data sets used in the research are generated. The next part (the model) describes the neural network, its layers and baseline loss function. And in the last part (the similarity loss) an additional loss function on top of the existing loss function will be discussed and how it is implemented.

3.1 Data generation

As noted in the Introduction, the first data set that is used for this research is the MNIST dataset which has 70,000 images of the numbers 0 through 9. The second dataset that is used is the CIFAR-10 dataset. This dataset contains 60,000 images divided over 10 classes with 6,000 images each. The classes are airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The MNIST dataset contains 28 by 28 grayscale images and the CIFAR-10 contains 32 by 32 color images. To combine images from both data sets into a combined image the MNIST images needed to be padded into 32 by 32 pixel images. After these padding the images from the two datasets are multiplied with each other to create a combined image, see figure 1. Every MNIST number has its own unique (disjunct) set of background CIFAR-10 images, there is a parameter which can allow some overlap between the different background image sets but the default will create disjunct sets.

Figure 1: Examples of the generated data set used in the experiments, from left to right: a zeven with a boat as background, a nine with a cat as background, a nine with a deer as background and a different nine with the same deer as background

3.2 The model

The model for the neural network that is used in this research is the 1989 LeNet neural network made by Yann LeCun [15]. The MNIST database was used to benchmark this network so it is well suited to recognize MNIST numbers. It contains 4 convolutional layers with activation and pooling layers in between, then it has 3 fully connected layers with the last layer having 10 neurons which are used to determine the number. Convolutional layers have a filter/kernel which returns a feature vector for every pixel and its surroundings. The activation function (in this case ReLu) maps this value to zero if it is below a threshold of zero or otherwise it does nothing with the feature vector and the pooling layer combines multiple feature vectors (in this case 4) into 1 feature vector. The network has a relatively small number of parameters (62,006) such that it is not likely to overfit on the training data set. The neural network uses an optimizer (in this case it uses the Adam optimizer [16] and the loss function is cross-entropy loss on the last layer of the network and the label of the image. The cross-entropy is defined as:

$$\mathcal{L}_{CE} = l(x, y) = \frac{\sum_{n=1}^{N} l_n}{N}$$

Where $\mathbf{x}$ is the input, the output of the network, $\mathbf{y}$ is target, in our case the label of the image and $\mathbf{N}$ is the size of the minibatch. So the cross-entropy loss is computed on every sample in the minibatch after which the mean is taken over all the samples in that minibatch. The cross-entropy on a single sample is computed as follows:

$$l_n = -\sum_{c=1}^{C} \log \frac{\exp(x_{n,c})}{\sum_{i=1}^{C} \exp(x_{n,i})} y_{n,c}$$

Where $\mathbf{n}$ is the n-th sample in the minibatch, $\mathbf{C}$ is the amount of classes, in our case 10, the numbers 0 to 9, $\mathbf{c}$ and $\mathbf{i}$ are respectively the c-th and i-th class, $\mathbf{x_{n,c}}$ and $\mathbf{x_{n,i}}$ are the probability of respectively class $\mathbf{c}$ and $\mathbf{i}$ for sample $\mathbf{n}$, and $\mathbf{y_{n,c}}$ is the target for class c for sample n.

3.3 Similarity loss

This novel loss function encourages the model to bring the embeddings of closely related images together such that its reliance on spurious correlations decreases. This loss uses the embedding of the penultimate layer from the i-th sample and is denoted as $\mathbf{z}_i$. The set $\mathcal{Q}$ contains all the labels. The set $\mathcal{T}_q$, where $q \in \mathcal{Q}$, is defined as all the samples i where the $y_i = q$.

This subsection is divided in two different approaches to calculate a similarity loss on the penultimate layer. The first approach has been used in the project and it will be shown that it is equivalent to the second approach. The reason that the first (less intuitive) approach is used, is because its time complexity is $O(|\mathcal{T}|)$ while the time complexity of the second approach is $O(|\mathcal{T}|^2)$. This is because the mean for average embedding with a certain label in the first approach only needs to be computed once and then it is compared $|\mathcal{T}|$ times with the embedding with same label while the second approach has $|\mathcal{T}|$ times $(|\mathcal{T}|-1)$ comparisons between all the embeddings with the same label.

Embedding to average Embedding comparison
First the average embedding is calculated for every label.

$$\mu_q = \frac{1}{|\mathcal{T}_q|} \sum_{i \in \mathcal{T}_q} \mathbf{z}_i \tag{1}$$

Then the Mean Squared Error (MSE) is calculated between every embedding $\mathbf{z}_i$ and the average embedding with y_i.

$$S_i = \|\mathbf{z}_i - \mu_q\|_2^2 \tag{2}$$

Then all the MSE's are summed into a loss function for the label q.

$$\mathcal{L}_{avg,\ q} = \sum_{i \in \mathcal{T}_q} S_i \tag{3}$$

If we rewrite equation 3 we get:

$$\mathcal{L}_{avg,\ q} = \sum_{i \in \mathcal{T}_q} \|\mathbf{z}_i - \frac{1}{|\mathcal{T}_q|} \sum_{j \in \mathcal{T}_q} \mathbf{z}_j\|_2^2 \tag{4}$$

Which becomes, see for intermediate steps equation 13 until equation 30 in appendix A.1:

$$\mathcal{L}_{avg,\ q} = \sum_{i \in \mathcal{T}_q} \|\mathbf{z}_i\|_2^2 - \frac{1}{|\mathcal{T}_q|} \sum_{i \in \mathcal{T}_q} \sum_{j \in \mathcal{T}_q} \mathbf{z}_i^\top \mathbf{z}_j \tag{5}$$

Embedding to Embedding comparison

S_{ij} gives the Mean Squared Error (MSE) between embedding $\mathbf{z}_i$ and $\mathbf{z}_j$:

$$S_{ij} = \|\mathbf{z}_i - \mathbf{z}_j\|_2^2 \tag{6}$$

The Loss for this label then becomes:

$$\mathcal{L}_{pair,\ q} = \sum_{i,j \in \mathcal{T}_q \times \mathcal{T}_q} S_{ij} \tag{7}$$

$$\mathcal{L}_{pair,\ q} = \sum_{i,j \in \mathcal{T}_q \times \mathcal{T}_q} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2 \tag{8}$$

This can be rewritten to, see for intermediate steps equations 32 until 41 in appendix A.2.

$$\mathcal{L}_{pair,\ q} = 2|\mathcal{T}_q|(\sum_{i \in \mathcal{T}_q} \|\mathbf{z}_i\|_2^2 - \frac{1}{|\mathcal{T}_q|} \sum_{i \in \mathcal{T}_q} \sum_{j \in \mathcal{T}_q} \mathbf{z}_i^\top \mathbf{z}_j) \tag{9}$$

Which means that:

$$\mathcal{L}_{pair,\ q} = 2|\mathcal{T}_q|\mathcal{L}_{avg,\ q} \tag{10}$$

Because both losses are proportional to each other and the loss is multiplied by the parameter λ, both losses can be used for calculating the Similarity loss. As explained in the beginning of this section the first approach has been used for this project because of it lower time complexity.

$$\mathcal{L}_{sim} = \sum_{q \in \mathcal{Q}} \mathcal{L}_{avg,\ q} \tag{11}$$

4 Experimental Evaluation

The influence of the similarity loss on the total loss was varied with the λ parameter, this gives the total loss function:

$$\mathcal{L}_{total} = \mathcal{L}_{crossentropy} + \lambda * \mathcal{L}_{sim} \tag{12}$$

To gain more information about the influence of the similarity loss on the performance the experiment was performed with $\lambda = 0, 0.25, 0.5, 0.75$, this gave the results that can be seen in figure 2.

As is visible in figure 2 the different standard deviations of the runs overlap either partially or completely with each other, so the runs did not significantly differ from each other. You could then wonder if the similarity loss could not make an impact due to the amount of epochs. Therefore the amount of

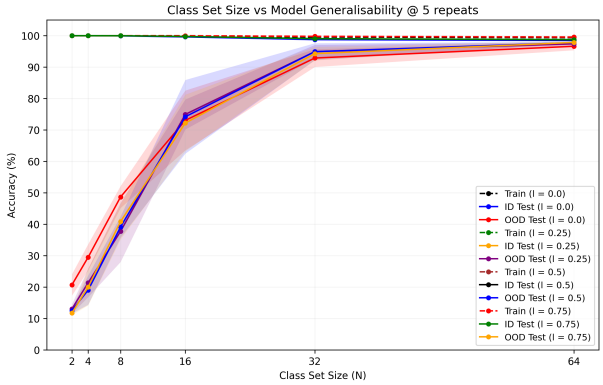

Figure 2: An graph of the performance of fours experiments where the experiments had respectively a λ of 0, 0.25, 0.5 & 0.75, the line is the mean of every experiment, the shaded area the 95% confidence interval and the amount of different seeds is 5

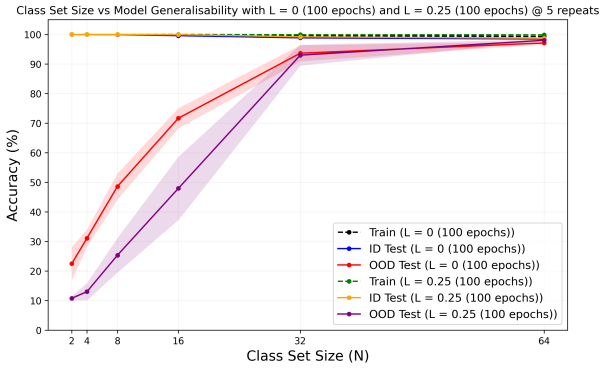

Figure 3: An graph of the performance of two experiments where one experiment had a λ of 0 and the other 0.25, the line is the mean of every experiment, the shaded area the 95% confidence interval and the amount of different seeds is 5

epochs was changed from 10 to 100, and the experiment was repeated with a $\lambda = 0, 0.25$, this gave the results of figure 3.

Because the overlap was still present, the question arose if the similarity loss had an actual effect. To test this, two runs were started where $\lambda = 10, 100$ to look if the training would proceed differently when the similarity loss was the largest contributor to the loss. Figure 4 shows these results.

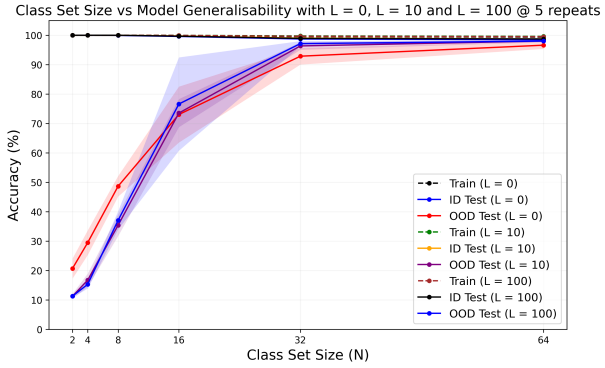

Figure 4: An graph of the performance of three experiments where the λ respectively is 0, 10 & 100, the line is the mean of every experiment, the shaded area the 95% confidence interval and the amount of different seeds is 5

Because the results are still mostly overlapping with each other the question could be asked if the cross-entropy loss and the similarity loss are correlated. To test this out a run was started where the cross-entropy loss was set to 0 and the λ was 1, figure 5 shows the result of that run.

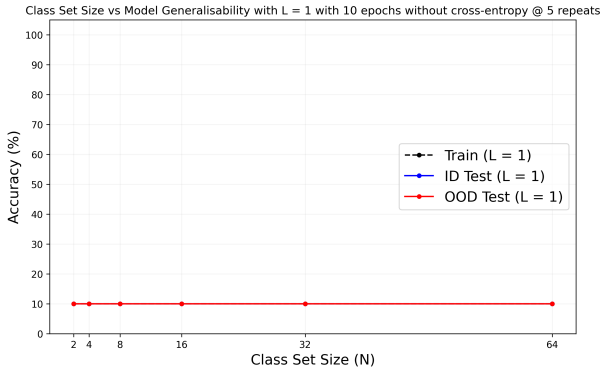

Figure 5: An graph of the performance of one run with a λ of 1 and a cross-entropy loss of 0, the line is the mean of every experiment, the shaded area the 95% confidence interval and the amount of different seeds is 5

As is visible in figure 5 the accuracy stays at 10%. This means that the network has the same performance as guessing the number. When looking at the intermediate losses for this experiment and the previous experiments with cross-entropy loss it is interesting to note that the similarity loss starts from 10^{-5}. This indicate that the starting state of the network is already close together as the similarity loss calculates differences between embeddings. Because there was no cross-entropy loss in the experiment of figure 5 and the network starts close together, the optimizer does not change the network enough to come out of this local minima. These results incentivizes to change the initialization of the network to make sure that the embeddings are further from each other at the start of the run.

5 Responsible Research

5.1 Ethics

Neural networks are increasingly used in society, for example in predicting crime rates but also in image recognition and speech-to-text applications. Because this can cause a dependency on neural networks it is important to think about the consequences of neural networks when researching them. Energy consumption is a good example of this, energy consumption has been drastically increased by the use of generative AI. This research tries to improve the OOD generalization without using more and more diverse training data, which can make neural networks more efficient and save energy when running them. Because this research has been conducted on a small model with a low-resolution data set the energy consumption was brought to a minimum.

5.2 Reproducibility

Every experiment was repeated at least 5 times to filter out outliers. Furthermore different amount of epochs and λ's have been used to be certain that the results are caused by the new loss function and not other parameters.

6 Conclusions, Discussions and Future Work

The main research question is "How does similarity loss affect the out-of-distribution generalization ability of neural networks?". The results from this paper show that the Similarity loss in its current form does not significantly affect the out-of-distribution generalization ability. Because the current form of Similarity loss also brings embeddings together that share the same background, it brings embeddings together were the background is more similar than the MNIST number itself. So this may result in worse OOD generalization because the model becomes more dependent on spurious correlations. That may also explain why the OOD generalization is worse with the Similarity loss on small class set sizes, because the backgrounds are more similar than the foregrounds. What could be a topic of further research is looking into hybrid loss functions, where the similarity loss start later. Furthermore a research into a similarity loss that only brings embeddings together with the same label but with different background would be informative to determine if clustering embeddings without clustering the spurious features together gives a better OOD generalization. A special case of this research would be to only cluster embeddings that have the exact same foreground but different background. In this way the model would presumably become invariant to the background if there are labels available for both the foreground and background.

References

[1] Uzair Aslam Bhatti, Jingbing Li, Saqib Ali Nawaz, Mengxing Huang, Raza Muhammad Ahmad, and Yazeed Yasin Ghadi. *Remote-Sensing Image Classification*, book section 2, pages 18–47. CRC Press, Boca Raton, 1st edition, 2024.

[2] Manuel Caldeira, Pedro Martins, Jose Cecília, and Pedro Furtado. Comparison study on convolution neural networks (cnns) vs. human visual system (hvs), 2019.

[3] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012.

[4] Panissara Thanapol, Kittichai Lavangnananda, Pascal Bouvry, Frédéric Pinel, and Franck Leprévost. Reducing overfitting and improving generalization in training convolutional neural network (cnn) under limited sample sizes in image recognition, 2020.

[5] Théo Sourget, Michelle Hestbek-Møller, Amelia Jiménez-Sánchez, Jack Junchi Xu, and Veronika Cheplygina. Mask of truth: Model sensitivity to unexpected regions of medical images. *Journal of Imaging Informatics in Medicine*, 2025.

[6] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization, 2019.

[7] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks, 2020.

[8] Wenqian Ye, Luyang Jiang, Eric Xie, Guangtao Zheng, Yunsheng Ma, Xu Cao, Dongliang Guo, Daiqing Qi, Zeyu He, Yijun Tian, Megan Coffee, Zhe Zeng, Sheng Li, Ting hao (Kenneth)Huang, Ziran Wang, James M. Rehg, Henry Kautz, and Aidong Zhang. The clever hans mirage: A comprehensive survey on spurious correlations in machine learning, 2025.

[9] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization, 2019.

[10] Xingxuan Zhang, Peng Cui, Renzhe Xu, Linjun Zhou, Yue He, and Zheyan Shen. Deep stable learning for out-of-distribution generalization, 2021.

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

[12] Michael Zhang, Nimit S. Sohoni, Hongyang R. Zhang, Chelsea Finn, and Christopher Ré. Correct-n-contrast: A contrastive approach for improving robustness to spurious correlations, 2022.

[13] Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation, 2021.

[14] Bram Grooten, Tristan Tomilin, Gautham Vasan, Matthew E. Taylor, A. Rupam Mahmood, Meng Fang, Mykola Pechenizkiy, and Decebal Constantin Mocanu. Madi: Learning to mask distractions for generalization in visual deep reinforcement learning, 2023.

[15] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

A Detailed equations

A.1 Embedding to Average Embedding comparison

If we rewrite equation 4 we get:

$$\mathcal{L}_{avg,\,q} = \sum_{i\in\mathcal{T}_q}(\mathbf{z}_i - \frac{1}{|\mathcal{T}_q|}\sum_{j\in\mathcal{T}_q}\mathbf{z}_j)^\top \tag{13}$$

$$(\mathbf{z}_i - \frac{1}{|\mathcal{T}_q|}\sum_{j\in\mathcal{T}_q}\mathbf{z}_j) \tag{14}$$

$$= \sum_{i\in\mathcal{T}_q}(\mathbf{z}_i^\top\mathbf{z}_i - \mathbf{z}_i^\top(\frac{1}{|\mathcal{T}_q|}\sum_{j\in\mathcal{T}_q}\mathbf{z}_j) - \tag{15}$$

$$(\frac{1}{|\mathcal{T}_q|}\sum_{j\in\mathcal{T}_q}\mathbf{z}_j)^\top\mathbf{z}_i + \tag{16}$$

$$(\frac{1}{|\mathcal{T}_q|}\sum_{j\in\mathcal{T}_q}\mathbf{z}_j)^\top(\frac{1}{|\mathcal{T}_q|}\sum_{j\in\mathcal{T}_q}\mathbf{z}_j)) \tag{17}$$

$$= \sum_{i\in\mathcal{T}_q}(\|\mathbf{z}_i\|_2^2 - \tag{18}$$

$$\frac{1}{|\mathcal{T}_q|}\sum_{j\in\mathcal{T}_q}\mathbf{z}_i^\top\mathbf{z}_j - \frac{1}{|\mathcal{T}_q|}\sum_{j\in\mathcal{T}_q}\mathbf{z}_j^\top\mathbf{z}_i + \tag{19}$$

$$\frac{1}{|\mathcal{T}_q|^2}\sum_{j\in\mathcal{T}_q}\mathbf{z}_j^\top(\sum_{j\in\mathcal{T}_q}\mathbf{z}_j)) \tag{20}$$

Because these two sums are equal:

$$\frac{1}{|\mathcal{T}_q|}\sum_{j\in\mathcal{T}_q}\mathbf{z}_i^\top\mathbf{z}_j = \frac{1}{|\mathcal{T}_q|}\sum_{j\in\mathcal{T}_q}\mathbf{z}_j^\top\mathbf{z}_i \tag{21}$$

And to make the multiplication of two sums less confusing the index of the second sum is changed to k:

$$\frac{1}{|\mathcal{T}_q|^2}\sum_{j\in\mathcal{T}_q}\mathbf{z}_j^\top(\sum_{j\in\mathcal{T}_q}\mathbf{z}_j) \to \frac{1}{|\mathcal{T}_q|^2}\sum_{j\in\mathcal{T}_q}\mathbf{z}_j^\top(\sum_{k\in\mathcal{T}_q}\mathbf{z}_k) \tag{22}$$

Both of these give the following:

$$\begin{aligned}\mathcal{L}_{avg,\,q} &= \sum_{i\in\mathcal{T}_{\text{Ц}}}(\|\mathbf{z}_i\|_2^2 - \frac{2}{|\mathcal{T}_q|}\sum_{j\in\mathcal{T}_q}\mathbf{z}_i^\top\mathbf{z}_j + && (23)\\ &\quad \frac{1}{|\mathcal{T}_q|^2}\sum_{j\in\mathcal{T}_q}\sum_{k\in\mathcal{T}_q}\mathbf{z}_j^\top\mathbf{z}_k) && (24)\\ &= \sum_{i\in\mathcal{T}_q}\|\mathbf{z}_i\|_2^2 - \frac{2}{|\mathcal{T}_q|}\sum_{i\in\mathcal{T}_q}\sum_{j\in\mathcal{T}_q}\mathbf{z}_i^\top\mathbf{z}_j + && (25)\\ &\quad \frac{1}{|\mathcal{T}_q|^2}\sum_{i\in\mathcal{T}_q}\sum_{j\in\mathcal{T}_q}\sum_{k\in\mathcal{T}_q}\mathbf{z}_j^\top\mathbf{z}_k && (26)\end{aligned}$$

Because the last part is not dependent on i the sum over i can be replaced by the multiplication over the length of $\mathcal{T}_q$.

$$\frac{1}{|\mathcal{T}_q|^2}\sum_{i\in\mathcal{T}_q}\sum_{j\in\mathcal{T}_q}\sum_{k\in\mathcal{T}_q}\mathbf{z}_j^\top\mathbf{z}_k = \frac{|\mathcal{T}_q|}{|\mathcal{T}_q|^2}\sum_{j\in\mathcal{T}_q}\sum_{k\in\mathcal{T}_q}\mathbf{z}_j^\top\mathbf{z}_k \quad (27)$$

Because the $\mathbf{z}_i$ embeddings are the same as the $\mathbf{z}_j$ and $\mathbf{z}_k$ we can change the index from k to i.

$$\begin{aligned}\frac{|\mathcal{T}|_q}{|\mathcal{T}_q|^2}\sum_{j\in\mathcal{T}_q}\sum_{k\in\mathcal{T}_q}\mathbf{z}_j^\top\mathbf{z}_k &= \frac{1}{|\mathcal{T}_q|}\sum_{j\in\mathcal{T}_q}\sum_{i\in\mathcal{T}_q}\mathbf{z}_j^\top\mathbf{z}_i && (28)\\ &= \frac{1}{|\mathcal{T}_q|}\sum_{i\in\mathcal{T}_q}\sum_{j\in\mathcal{T}_q}\mathbf{z}_i^\top\mathbf{z}_j && (29)\end{aligned}$$

This gives us:

$$\begin{aligned}\mathcal{L}_{avg,\,q} &= \sum_{i\in\mathcal{T}_q}\|\mathbf{z}_i\|_2^2 - \frac{2}{|\mathcal{T}_q|}\sum_{i\in\mathcal{T}_q}\sum_{j\in\mathcal{T}_q}\mathbf{z}_i^\top\mathbf{z}_j + && (30)\\ &\quad \frac{1}{|\mathcal{T}_q|}\sum_{i\in\mathcal{T}_q}\sum_{j\in\mathcal{T}_q}\mathbf{z}_i^\top\mathbf{z}_j && (31)\end{aligned}$$

And that gives equation 5

A.2 Embedding to Embedding comparison

Rewriting equation 8 gives:

$$\begin{aligned}\mathcal{L}_{pair,\,q} &= \sum_{i\in\mathcal{T}_q}\sum_{j\in\mathcal{T}_q}\|\mathbf{z}_i - \mathbf{z}_j\|_2^2 && (32)\\ &= \sum_{i\in\mathcal{T}_q}\sum_{j\in\mathcal{T}_q}(\mathbf{z}_i - \mathbf{z}_j)^\top(\mathbf{z}_i - \mathbf{z}_j) && (33)\\ &= \sum_{i\in\mathcal{T}_q}\sum_{j\in\mathcal{T}_q}(\mathbf{z}_i^\top\mathbf{z}_i - \mathbf{z}_j^\top\mathbf{z}_i - && (34)\\ &\quad \mathbf{z}_i^\top\mathbf{z}_j + \mathbf{z}_j^\top\mathbf{z}_j) && (35)\\ &= \sum_{i\in\mathcal{T}_q}\sum_{j\in\mathcal{T}_q}(\|\mathbf{z}_i\|_2^2 - 2\mathbf{z}_i^\top\mathbf{z}_j + \|\mathbf{z}_j\|_2^2) && (36)\\ &= \sum_{i\in\mathcal{T}_q}\sum_{j\in\mathcal{T}_q}\|\mathbf{z}_i\|_2^2 - 2\sum_{i\in\mathcal{T}_q}\sum_{j\in\mathcal{T}_q}\mathbf{z}_i^\top\mathbf{z}_j + && (37)\\ &\quad \sum_{i\in\mathcal{T}_q}\sum_{j\in\mathcal{T}_q}\|\mathbf{z}_j\|_2^2 && (38)\\ &= |\mathcal{T}_q|\sum_{i\in\mathcal{T}_q}\|\mathbf{z}_i\|_2^2 + |\mathcal{T}_q|\sum_{j\in\mathcal{T}_q}\|\mathbf{z}_j\|_2^2 - && (39)\\ &\quad 2\sum_{i\in\mathcal{T}_q}\sum_{j\in\mathcal{T}_q}\mathbf{z}_i^\top\mathbf{z}_j && (40)\end{aligned}$$

Because the first two sums are the same, the loss becomes:

$$\mathcal{L}_{pair,\,q} = 2|\mathcal{T}_q|\sum_{i\in\mathcal{T}_q}\|\mathbf{z}_i\|_2^2 - 2\sum_{i\in\mathcal{T}_q}\sum_{j\in\mathcal{T}_q}\mathbf{z}_i^\top\mathbf{z}_j \quad (41)$$

Which becomes equation 9