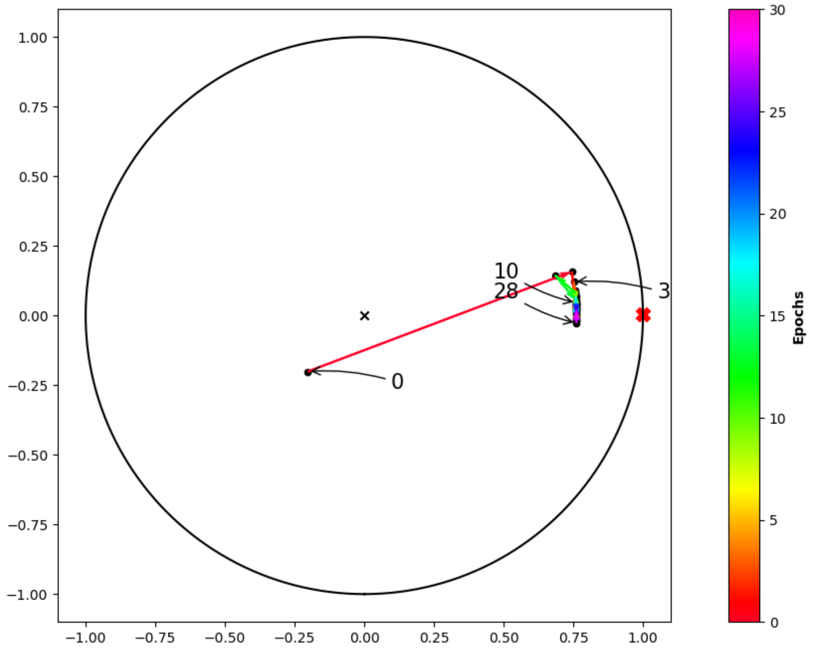
Exploring the Scalability of Hyperbolic Busemann Learning with many Classes in high-dimensional output Spaces

submitted in partial fulfillment for the degree of master of science

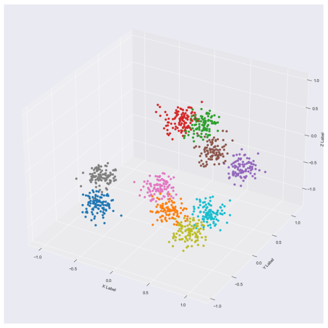
David Jung
13646168

master information studies
data science
faculty of science
university of amsterdam

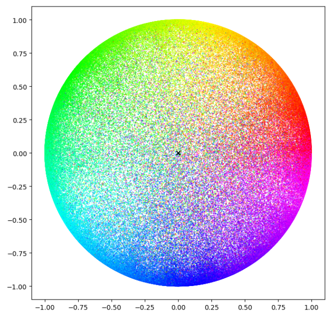
2022-06-30



	Supervisor	Examiner
Title, Name	Mina GhadimiAtigh	Dr. Pascal Mettes
Affiliation	University of Amsterdam	University of Amsterdam
Email	m.ghadimiatigh@uva.nl	p.s.m.mettes@uva.nl



Universiteit van Amsterdam



Exploring the Scalability of Hyperbolic Busemann Learning with many Classes in high-dimensional output Spaces

David Jung
Student ID: 13646168
University of Amsterdam
david.jung@student.uva.nl

ABSTRACT

Several papers have shown that hyperbolic space has become a promising choice of manifold for representation learning of different data types. Using hyperbolic prototypes for classification allows for effective learning in low-dimensional output spaces and the extraction of hierarchical relations amongst classes. However, privileged information about class labels is needed to position the hyperbolic prototypes. Hyperbolic Busemann Learning circumvents this by positioning class prototypes on the ideal boundary of a Poincaré ball which outperforms recent hyperspherical and hyperbolic prototype approaches in low-dimensional output spaces. However, the gap diminishes with many classes using high-dimensional output spaces. Thus, this work investigates the cause of the decreasing gap and successfully scales Hyperbolic Busemann Learning up to 1'000 classes using 1'000-dimensional output spaces. Moreover, this research provides insights about embedding 1'000 classes in two-dimensional output spaces, visualisations of the learning process and loss distributions.

KEYWORDS

Deep Learning, Hyperbolic Busemann Learning, Classification, Poincaré Ball Model, Super-Hyperbolic Classifiers

1 INTRODUCTION

Prototypes have been used for decades for classification tasks within machine learning. A commonly used approach is the Nearest Mean Classifier [32] which represents classes as the mean prototype within a fixed feature space where the mean is computed over all training samples for each class and therewith positioned. Deep learning with prototypes as points in network output spaces has been implemented more and more frequently in recent years [10, 11, 18]. Especially few-shot learning [7, 10, 12, 29] as well as zero-shot learning [15, 29, 33] achieve high accuracy scores by using mean prototypes.

Although high accuracy scores have emerged by deep learning with mean prototypes in Euclidean space, they have a disadvantage, namely the requirement to re-learn prototypes and the inability to use prior label knowledge. This can be solved by using non-Euclidean output spaces for prototypes. Within hyperspherical prototype approaches, prototypes are initialised on the sphere or its higher-dimensional generalisation, which is based on pair-wise angular separation [24, 28], or with word embeddings [3, 30] that use a cosine similarity loss between example outputs and class prototypes. By using hyperspherical prototype methods, there is no need to re-position prototypes continuously. Additionally, including prior semantic knowledge about classes is thereby enabled. Prototype-based learning has lately been extended to the hyperbolic space where prototypes are obtained by embedding label hierarchies [22, 23]. More hierarchically coherent classification results are achieved by using hyperbolic prototype approaches, which presuppose prior hierarchical knowledge for embedding prototypes.

Atigh et al. [1] introduced a hyperbolic prototype network with class prototypes which are positioned as points on the ideal boundary of the Poincaré ball model of hyperbolic geometry. The ideal boundary of the Poincaré ball model is infinitely far away from all other points in hyperbolic space. Additionally, the penalised Busemann loss was proposed, which enables the computation of the proximity between an example output in hyperbolic space and its respective prototype on the ideal boundary. This approach allows knowledge-free positioning from hyperspherical prototypes as well as efficient low-dimensional embeddings from hyperbolic prototypes.

While experiments conducted by Atigh et al. [1] show that hyperbolic Busemann learning (HBL) outperforms both hyperspherical and hyperbolic prototype approaches, it has been observed that when using high-dimensional output spaces for many classes, the performance gap between models using hyperbolic Busemann learning and previous approaches slowly diminishes. Moreover, they conducted experiments with up to 200 classes in a 100-dimensional embedding space. Thus, HBL in a setting with more than 200 classes in a higher dimensional embedding space has not yet been explored. For this reason, this work tries to identify the cause of the diminishing gap and focuses on the scalability of HBL beyond 200 classes where special focus is given to the experiments with 1'000 classes using 1'000-dimensional output spaces. Taking everything into consideration, the following research question can be derived:

Research Question

- How can Hyperbolic Busemann Learning be scaled to 1'000 classes?

Sub-Questions

- To what extent influences the number of classes and the number of dimensions of the output spaces the accuracy scores?
- What are the accuracy scores with 1'000 classes using two-dimensional output spaces?
- How can the learning process be visualised?
- How do the loss values obtained from Hyperbolic Busemann Learning behave in different dimensions with data close to the origin or the ideal boundary of a Poincaré ball?

This paper is structured as follows. In Section 2, the existing literature within the field of prototypes in deep networks is presented. Within Section 3, the data and methodologies utilised in this paper are explained in more detail. The experimental setup is introduced in Section 4. The obtained results of this work are presented in Section 5. The results are discussed in Section 6. Finally, Section 7 draws a conclusion and indicates to future work. The code can be found at: https://github.com/damajojung/HBL_Exploration

2 RELATED LITERATURE

As mentioned in Section 1, this work investigates the scalability of hyperbolic Busemann learning using high-dimensional output spaces. The following sections introduce previous research that has been conducted in the field of hyperbolic networks with a focus on hyperbolic prototype-based networks.

Deep Learning within hyperbolic Space

Hyperbolic space has gained increasing attention over the last years for deep representation learning due to its high capacity for modelling data such as knowledge graphs or synonym hierarchies with hierarchical structures [14]. Such hyperbolic neural networks can yield more compact models than their Euclidean counterparts, resulting in higher efficiency. Numerous papers have been published that investigated different architectures in hyperbolic space, such as hyperbolic embeddings [5, 17, 26], hyperbolic clustering [25], hyperbolic attention networks [8, 16], hyperbolic graph neural networks [2, 21, 27] and hyperbolic prototypical networks [13].

Prototype-based Networks

When it comes to classification within the field of deep learning, most researchers use a softmax classifier with a cross-entropy loss. However, with an increasing number of classes, softmax turns out to be slow within the training process since an output node is needed for each class. Hence, it is desirable to have a constant output size regardless of the number of classes. This can be achieved with prototypical networks where a pre-defined prototype is used, which is generated before the training process [29].

Within machine learning, a prototype is a data value that reflects other values in its class such as a the mean value of a class. This concept can be used in neural networks, where prototypes are points in network output spaces. These neural networks are called prototypical networks [29]. Especially few-shot learning [7, 10, 12, 29] as well as zero-shot learning [15, 29, 33] transpired to be effective when it comes to learning with prototypes from class means. DeepNCM [18] and DeepNNO [11] propose different optimisation schemes in contrast to general supervised learning, where prototypes are fixed to update the network's core and vice versa. However, with a new data sample being added to the dataset, one must constantly update the prototypes, which is sub-optimal. Therefore, several works suggested using the hypersphere as output space instead of the Euclidean space, where prototypes can be positioned a priori based on their angular separation [24, 28] using an orthonormal basis [3], or using word embeddings [3, 24, 30]. Thus, the prototypes remain fixed throughout the learning process and do not require constant updating.

Hyperbolic Prototype-based Networks

As mentioned before, prototypes can be placed a priori in a hyperbolic model. As of now, the Poincaré model and the Lorentz model are the most commonly used hyperbolic models for generalising neural networks in hyperbolic space. However, deep hyperbolic networks are mainly applied in the Poincaré model [14]. Whereas most previous approaches worked within the Poincaré model, Atigh et al. [1] placed class prototypes on the ideal boundary of the Poincaré model, which enables the benefits of the representational power of hyperbolic geometry. Additionally, no prior knowledge is required with this approach. Several works using hyperbolic output spaces have shown to be effective in low dimensions, such as [13, 22, 23]. These low-dimensional embeddings have several notable advantages. Atigh et al. [1] emphasise two important ones, namely *Compactness* and *Interpretability*. The former has potential benefits for data transmission, compression and storage and the latter for interpretation and visualisation of the model performance.

Background

Hyperbolic Busemann Learning. Atigh et al. [1] introduced the idea of positioning class prototypes at ideal points. In hyperbolic geometry, ideal points are points at infinity and form the boundary of the ball of a Poincaré model.

$$\mathbb{I}_d = \{\mathbf{z} \in \mathbb{R}^d : z_1^2 + ... + z_d^2 = 1\} \tag{1}$$

Within hyperbolic space $\mathbb{B}_d$, a set of ideal points is homeomorphic to the hypersphere $\mathbb{S}_d$, which means that there is a continuous function between the ideal points and the hypersphere as well as a continuous inverse function. Consequently, all the methods used to embed prototypes into $\mathbb{S}_d$ can be used to embed prototypes into $\mathbb{I}_d$. When $d = 2$, prototypes can be placed uniformly on the unit sphere $\mathbb{S}_2$. In higher dimensions such as $d \geq 3$, class-agnostic prototype embedding based on separation can be used.

The geodesic distance can not be used directly for prototype-based learning since ideal points are at an infinite geodesic distance from all other points in $\mathbb{B}_d$. For this reason, penalised Busemann loss for hyperbolic learning with ideal prototypes was introduced by Atigh et al. [1].

The Busemann function [4] enables the computation of the distance of a point to infinity and can be defined in any metric space. Let $\mathbf{p}$ be an ideal point and $\gamma_{\mathbf{p}}$ a geodesic ray, parameterised by arc length, tending to $\mathbf{p}$. The Busemann function with respect to $\mathbf{p}$ then is defined for $\mathbf{z} \in \mathbb{B}_d$ as:

$$b_{\mathbf{p}}(\mathbf{z}) = log \frac{\|\mathbf{p} - \mathbf{z}\|^2}{(1 - \|\mathbf{z}\|^2)} \tag{2}$$

In the Poincaré model, the limit can be explicitly calculated and the Busemann function is given as:

$$b_{\mathbf{p}}(\mathbf{z}) = \lim_{t \to \infty} (d_{\mathbb{B}}(\gamma_{\mathbf{p}}(t), \mathbf{z}) - t) \tag{3}$$

The full derivation of Equation 3 can be found in the Appendix in [1]. They proposed a loss which combines the Busemann function with a penalty term which can be seen in Equation 4.

$$\ell(\mathbf{z}, \mathbf{p}) = b_{\mathbf{p}}(\mathbf{z}) - \phi(d) * log(1 - \|\mathbf{z}\|^2) \tag{4}$$

$\phi(d)$ is a scaling factor for the penalty term, which is a function of the dimension of the hyperbolic space and governs the amount of regularisation. The first term in Equation 4 steers a projected input $\mathbf{z}$ towards the class prototype $\mathbf{p}$, whereas the second term penalises overconfidence. Values close to the boundary of $\mathbb{B}_d$ are an example of overconfidence. The prototypes are positioned before training and remain fixed. Therefore, the backpropagation only needs to be done with respect to the inputs. Equation 5 shows the gradient of $\ell(exp_0(\mathbf{x}), \mathbf{p})$ with respect to $\mathbf{x}$. Please note that $\mathbf{1}_d$ is a d-dimensional vector of ones.

$$\nabla_x \ell(exp_0(\mathbf{x}), \mathbf{p}) = (\mathbf{x} - \mathbf{p}) \frac{tanh(\|\mathbf{x}\|)}{\|\mathbf{x}\| - tanh(\|\mathbf{x}\|)\mathbf{p} * x} + ... \tag{5}$$

$$\mathbf{1}_d\mathbf{p} * \mathbf{x} \frac{tanh(\|\mathbf{x}\|)/\|\mathbf{x}\| - 1}{\|\mathbf{x}\| - tanh(\|\mathbf{x}\|)\mathbf{p} * \mathbf{x}} + tanh(\|\mathbf{x}\|/2)$$

3 METHODOLOGY

Data

We generated synthetic datasets that are used for all the experiments conducted in this work. They consist of randomly generated representations sampled from an N-dimensional normal distribution as described by Tulkens [31]. This work created datasets with 10, 100 and 1'000 classes which enables the comparison with CIFAR-10 [19], CIFAR-100 [20] and ImageNet [6]. For each class in each dataset, the centre is defined by uniformly sampling an X-dimensional representation from the interval [-1, 1]. For each centre, samples are drawn from a normal distribution with its centre as mean and a standard derivation (sd) depending on the number of classes. Table 1 shows the various standard deviations.

N Classes	**10**	**100**	**1000**
Standard Deviation	3	2.5	1.25

Table 1: SD for different number of classes

The first step is to define how many classes and dimensions one would like to obtain. In this example, 10 classes with 3 dimensions are used. As mentioned before, a centre is defined for each class by uniformly sampling an X-dimensional representation from the interval [-1, 1]. Table 2 shows the three dimensional centres for 10 classes.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
x	-0.4	0.7	0.8	0.1	-0.2	-0.5	0.6	0.2	0.9	-0.7
y	0.6	0.7	0.2	-0.8	-0.8	-0.9	-0.1	0.3	-0.5	0.6
z	0.7	-0.5	-0.8	0.8	-0.6	0.6	0.3	-0.2	-0.4	-0.5

Table 2: X, Y and Z coordinates of the centres

Finally, one can draw samples from a normal distribution with the centre as its mean and the corresponding standard deviation from Table 1. For example, drawing samples for class 0 can be done with `np.random.normal(-0.4,3)`, `np.random.normal(0.6,3)` and `np.random.normal(0.7,3)` for each dimension respectively. One can see an example for 10 classes with a standard derivation of 3 in Figure 1. All datasets utilised in this work contain 1'000 samples per class in the training set and 200 samples per class for the validation set.

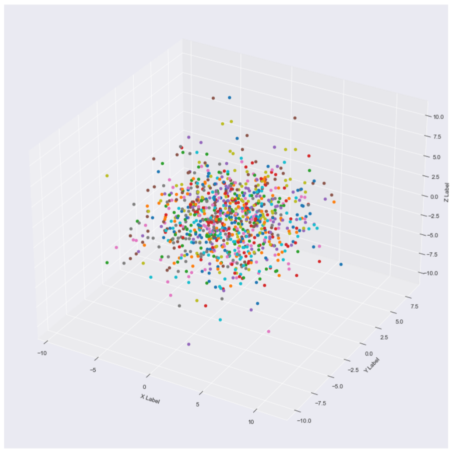


Figure 1: 3D scatterplot of synthetic data with sd 3

Scalability of HBL

The scalability of HBL up to 1'000 classes using 1'000 - dimensional embeddings is the main scope of this research. With the data described in the previous section, one can conduct experiments with 1'000 classes in 1'000-dimensional output spaces and thus investigate HBL's scalability. This research observed vanishing gradients in early experiments and had to find a solution to overcome them.

HBL with Clipping

Guo et al. [9] have indicated that the hybrid architecture connecting Euclidean features to a hyperbolic classifier can cause the vanishing gradient problem. Therefore, they introduced a hard constraint which can be seen in Equation 6 where the Euclidean embeddings are regularised before the exponential map whenever its norm exceeds a given threshold ($r > 0$).

$$C(\mathbf{x}^E; r) = min\{1, \frac{r}{||\mathbf{x}^E||}\} * \mathbf{x}^E \tag{6}$$

Visualisation of learning process

To visualise the learning process over time, one can plot the cumulative predictions after every epoch to identify any irregularities which could emerge during training. Thus, it is possible to detect whether HBL only focuses on a few classes or whether no learning happens at all and it randomly classifies samples.

Loss distributions

Finally, the concept for the experiments regarding the distributions of the loss values for each prototype can be seen in Figure 11 in Appendix A. 1'000 samples with a fixed norm of 0.1 in red and 0.9 in blue are used for calculating the loss values for each prototype which can be used for visualising the distributions. The experimental setup is described in greater detail in the following section.

4 EXPERIMENTS

The main scope of this research is to scale HBL up to 1'000 classes using 1'000-dimensional output spaces. Tulkens [31] investigated hyperspherical alternatives to softmax networks. Thus, they created datasets with 10, 100 and 1'000 classes and utilised up to 1'000-dimensional output spaces as well. These sets allowed them to investigate the effect of output dimensions on performance which is the same objective as the one this research has in mind. Therefore, this work uses the same architecture for the neural network, which is described in the following section. A softmax classifier with a cross-entropy loss is the current state-of-the-art approach within classification tasks. Thus, softmax results are also provided, which can be used for comparison with the hyperbolic results.

Softmax

The following architecture is chosen for the neural network for the synthetic data with softmax:

- Input layer of size d-dimensions
- Hidden fully connected layer number 1 of size 300
- Hidden fully connected layer number 2 of size 300
- Output layer of size N-classes

The chosen hyperparameters for the softmax experiments are as follows:

- 5 training epochs
- Learning rate of 1e-3
- Learning rate decay after epoch 3 & 4 of 10
- Batch size 200
- Log softmax
- ADAM optimiser
- No pre-training
- Dropout rate of 20%
- Rectified Linear Unit activation

Hyperbolic experiments

Regarding the hyperbolic network for the synthetic data, everything stays the same except the size of the output layer, which depends on the dimension of the prototypes. The hyperbolic experiments conducted in this work used the following hyperparameters:

- 30 training epochs
- Learning rate of 1e-3
- Learning rate decay after 20 & 25 epochs of 10
- Batch size of 200
- No pre-training
- Dropout rate of 20%
- Weight decay of 5e-5
- Rectified Linear Unit activation
- $\phi(d) = 0.0$

The softmax and hyperbolic experiments used one NVIDIA GeForce GTX 1080 Ti. No seed was used and every experiment was conducted five times. The obtained averages for the validation accuracy scores and their corresponding 95% confidence intervals are presented in Section 5.

5 RESULTS

High-dimensional experiments

Table 3 shows the obtained validation accuracy scores in percent as described in Section 4. Again, please note that the 10, 100 and 1k in the header for the hyperbolic results represent the output dimensions.

N-classes	Softmax	10	100	1k
10	91.6 ± 0.9	90.5 ± 1.2	90.5 ± 1.0	58.4 ± 4.4
100	85.0 ± 1.1	82.6 ± 0.8	56.7 ± 3.0	45.4 ± 1.6
1000	98.0 ± 0.5	38.2 ± 1.1	10.9 ± 0.7	14.6 ± 0.7

Table 3: Hyperbolic Busemann learning baseline - Accuracy scores over validation set

Clipping - Impact of r-value

Figure 3 shows the impact of different r-values for 1'000 classes. The class prototypes are positioned on the ideal boundary of a two-dimensional Poincaré model and coloured according to their respective class. With an r-value of 0.01, the results of the exponential map outputs are still close to the origin. However, once a bigger r-value is used, the predictions move closer to the class prototype.

How a sample from class 0 is moving through space in the Poincaré model during the learning process can be seen in Figure 2. In this particular experiment, an r-value of 1.0 was used. Thus, the sample keeps some distance from the prototype. It shows how the learning process evolves over time given a specific r-value. Please note that the numbers pointing to the dots represent the corresponding training epoch.

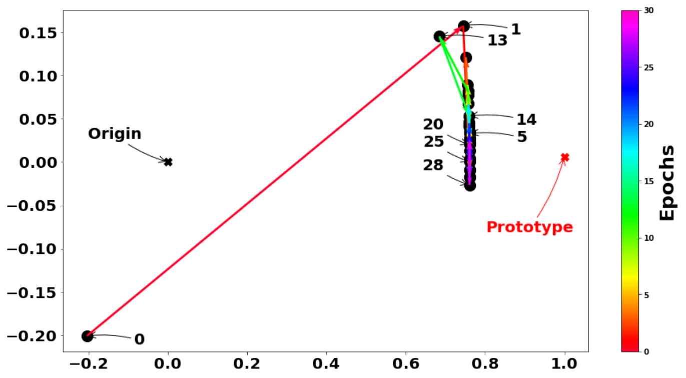


Figure 2: Visualisation of a moving sample image embedding with class label 0 towards the corresponding prototype within 30 epochs

High-dimensional experiments - With Clipping

Table 4 shows the obtained validation accuracy scores as described in Section 4. This time, clipping was used with different r-values for all experiments. Please note that the softmax results remain the same since clipping only affects the hyperbolic results.

r	N Classes	Softmax	10	100	1k
	10	91.6 ± 0.9	89.7 ± 0.6	89.6 ± 0.4	90.1 ± 0.5
0.01	100	85.0 ± 1.1	**83.9 ± 0.5**	84.8 ± 0.5	84.7 ± 0.4
	1000	98.0 ± 0.5	44.0 ± 0.9	32.0 ± 0.2	29.4 ± 0.3
	10	91.6 ± 0.9	90.5 ± 0.5	89.9 ± 0.2	90.1 ± 0.4
0.3	100	85.0 ± 1.1	83.1 ± 0.4	83.6 ± 0.3	83.2 ± 0.9
	1000	98.0 ± 0.5	**68.8 ± 0.9**	61.8 ± 0.2	29.4 ± 0.2
	10	91.6 ± 0.9	90.5 ± 0.8	90.2 ± 0.2	90.2 ± 0.5
0.5	100	85.0 ± 1.1	83.0 ± 0.4	82.7 ± 1.0	78.6 ± 1.7
	1000	98.0 ± 0.5	70.1 ± 0.3	48.4 ± 0.6	22.4 ± 0.9
	10	91.6 ± 0.9	90.7 ± 0.6	90.0 ± 0.4	89.7 ± 0.8
0.7	100	85.0 ± 1.1	83.3 ± 0.2	75.4 ± 2.9	65.6 ± 2.0
	1000	98.0 ± 0.5	60.7 ± 0.9	34.4 ± 1.3	14.9 ± 1.5
	10	91.6 ± 0.9	90.5 ± 0.6	90.1 ± 0.5	90.3 ± 0.4
1.0	100	85.0 ± 1.1	82.5 ± 0.7	57.6 ± 0.9	48.5 ± 3.0
	1000	98.0 ± 0.5	39.3 ± 0.5	21.7 ± 0.7	05.6 ± 3.4

Table 4: Comparison of r-values for different number of classes (10, 100, 1'000) and different output space dimensions - Accuracy scores over validation set

As mentioned by Guo et al. [9], one has to find a sweet spot for a suitable r-value for the data at hand. By taking a closer look at Table 4, the results for 1'000 classes using an r-value of 0.3 achieve the highest accuracy scores compared to the ones from different r-values. Therefore, this paper also investigated different learning rates with an r-value of 0.3, namely 1e-4 and 1e-5. The results from the former one can

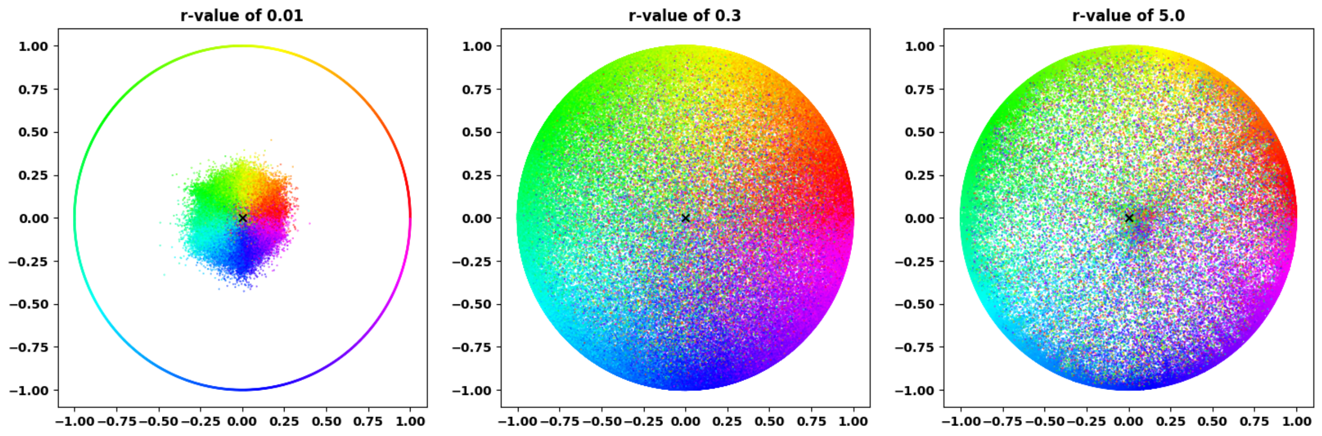


Figure 3: Output of exponential map for different r-values for 1'000 classes in two-dimensional output spaces

be seen in Table 5 and the ones from the latter one in Table 6. Everything else stayed exactly the same. Please note that the best results within the hyperbolic experiments across Tables 4, 5 and 6 are highlighted in bold text.

r	N Classes	Softmax	10	100	1k
0.3	10	91.6 ± 0.9	**91.5 ± 0.6**	**91.6 ± 0.9**	**91.8 ± 0.6**
	100	85.0 ± 1.1	62.8 ± 0.2	**85.3 ± 0.3**	**85.7 ± 0.6**
	1000	98.0 ± 0.5	54.5 ± 0.3	**69.3 ± 0.8**	38.5 ± 1.2

Table 5: Final HBL accuracy scores with r-value = 0.3 & learning rate 1e-4 - Accuracy scores over validation set

r	N Classes	Softmax	10	100	1k
0.3	10	91.6 ± 0.9	62.8 ± 2.4	71.5 ± 1.6	71.1 ± 2.1
	100	85.0 ± 1.1	09.7 ± 0.2	38.3 ± 0.8	46.3 ± 0.8
	1000	98.0 ± 0.5	05.7 ± 0.1	39.2 ± 0.3	**69.2 ± 0.7**

Table 6: Final HBL accuracy scores with r-value = 0.3 & learning rate 1e-5 - Accuracy scores over validation set

In addition to the lower learning rate of 1e-5 in Table 6, this paper conducted the same experiment with an r-value of 0.01 for 1'000 classes with 1'000 dimensions and obtained a validation accuracy of **77.2 ± 1.4%**.

Low-dimensional experiments

As mentioned by Atigh et al. [1], Hyperbolic Busemann Learning enables one to embed high-dimensional data in low-dimensional output spaces, which can be beneficial not only for data compression and thus data storage, but also regarding the interpretability of the behaviour of the model since it is possible to visualise intermediate steps. Therefore, this work also conducted experiments in two-dimensional output spaces with the same hyperparameters. The results can be seen in Table 7.

r	N Classes	2
0.01	10	82.2 ± 1.8
	100	14.5 ± 0.6
	1000	00.8 ± 0.1
0.3	10	85.1 ± 0.7
	100	15.6 ± 0.5
	1000	01.6 ± 0.0
0.5	10	86.6 ± 0.6
	100	24.2 ± 1.0
	1000	02.1 ± 0.1
0.7	10	87.0 ± 0.9
	100	36.3 ± 1.4
	1000	02.7 ± 0.1
1.0	10	87.6 ± 0.7
	100	51.7 ± 3.8
	1000	03.8 ± 0.3
5.0	10	85.5 ± 0.9
	100	66.7 ± 1.4
	1000	16.6 ± 0.8

Table 7: Comparison of r-values for different number of classes (10, 100, 1'000) and two-dimensional output space - Accuracy scores over validation set

Visualisation of learning process over time

By plotting the cumulative predictions for each class during the training process, one can obtain more insights. Figure 4 shows the training process for 1'000 classes using 1'000-dimensional output spaces without clipping for 15 training epochs and a learning rate of 1e-3, which results in a validation accuracy score of 13%. Since there are 1'000 samples per class, the model should predict *Amount of Epochs * 1'000* per class, namely 15'000 in this example, represented by the black horizontal line.

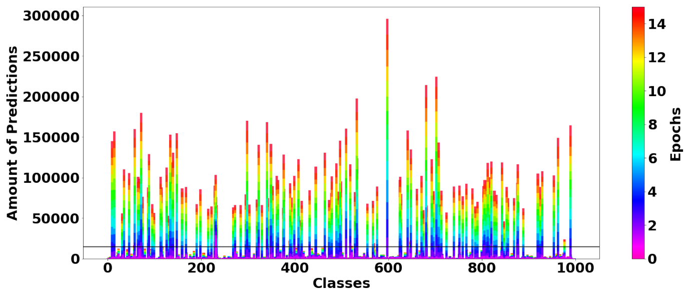


Figure 4: Cumulative predictions during training - No clipping for 1'000 classes using 1'000-dimensional output spaces

Figure 5 shows the cumulative predictions for each class while clipping is used with an r-value of 1e-1, a learning rate of 1e-5 and 15 training epochs which leads to a validation accuracy score of 79%.

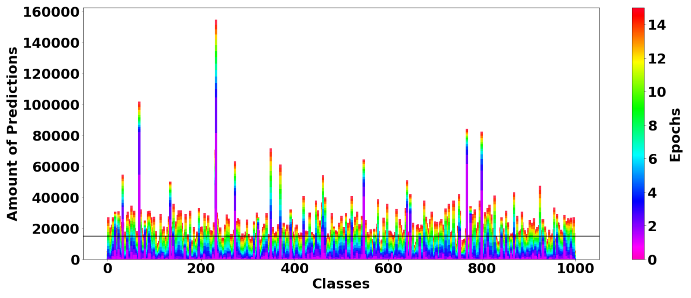


Figure 5: Cumulative predictions during training - Clipping with r = 1e-1 for 1'000 classes using 1'000-dimensional output spaces

Figure 6 shows the cumulative predictions for each class while clipping is used with an r-value of 3e-1 with 1'000 classes using two-dimensional output spaces, which results in a validation accuracy score of 1.6%.

Visualisation of Hyberbolic Busemann Loss values

This work investigated the distributions of the loss values for each prototype with data that has a fixed norm. Figure 7 shows the distributions for the loss values in two-dimensional output spaces with data close to the origin (norm 0.1).
Figure 8 shows the distributions for the loss values in two-dimensional output spaces with data close to the boundary (norm 0.9). In addition, one can also find Figure 12 and Figure

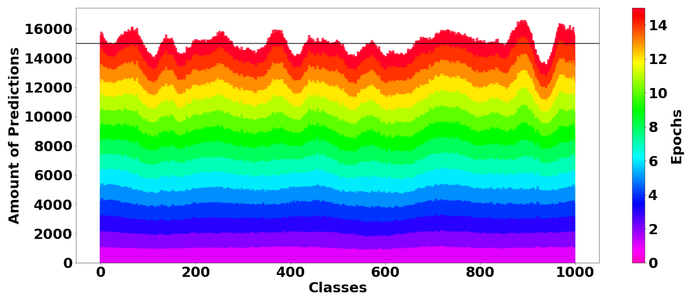


Figure 6: Cumulative predictions during training - Clipping with r = 3e-1 for 1'000 classes using two-dimensional output spaces resulting in a validation accuracy score of 1.6%

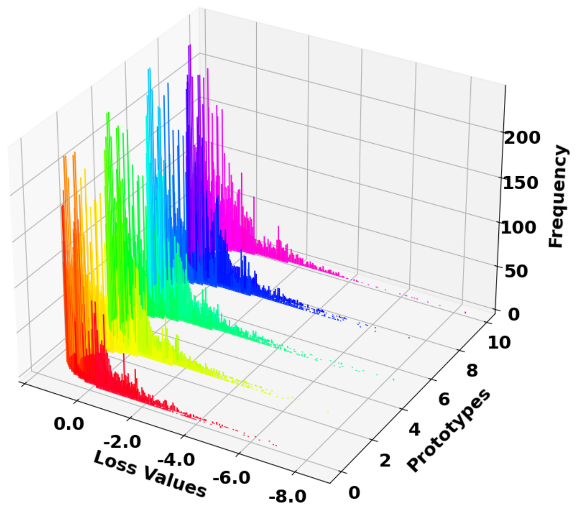


Figure 7: Loss distributions with data that has a fixed norm of 0.1 in two-dimensional output spaces

13 in Appendix A where 1'000-dimensional output spaces are used.

6 DISCUSSION

The obtained results are discussed in the following section. Moreover, the observed influences of different r-values on the results are shown.

Scalability of hyperbolic Busemann learning

Comparing the results to the hyperspherical prototype networks introduced by Mettes et al. [24] and explored by Tulkens [31], the hyperbolic prototype networks using hyperbolic Busemann learning with clipping seem to be a more promising choice in high-dimensional output spaces. Especially when it comes to 1'000 classes using 100- or 1'000-dimensional output spaces. Clipping enables an accuracy boost for 1'000

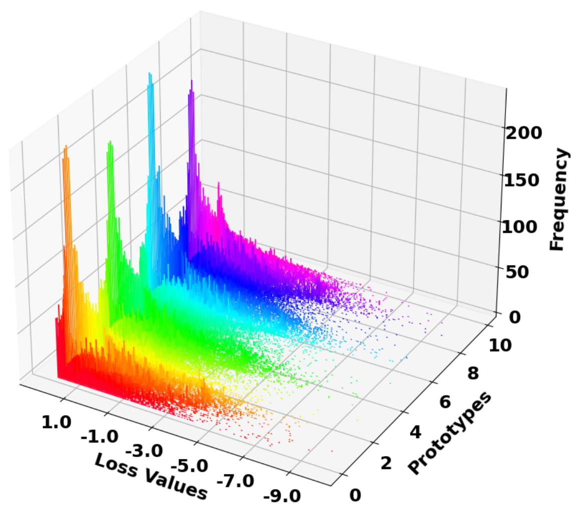


Figure 8: Loss distributions with data that has a fixed norm of 0.9 in two-dimensional output spaces

classes using 1'000-dimensional output spaces from 29.4% to 77.2%.

Clipping

This work started the experiments without the proposed clipping of the Euclidean embeddings by Guo et al. [9]. Therefore, the obtained results were below expectations. By analysing the gradients, it was observed that in high-dimensional output spaces with 100 and 1'000 classes the vanishing gradient problem occurs, which can be solved to a certain degree if clipping is used. This can be seen in Appendix A in Figure 15. Especially the classification task for 1'000 classes benefits from clipping. Moreover, by analysing the results, one can conclude that a lower learning rate for the 1'000 classes should be used. Interestingly, clipping does not prevent the vanishing gradient problem from occurring with 1'000 classes in two-dimensional output spaces.

Relationship between r-values and output dimensions

There seems to be an inverse relationship between the size of the output dimensions of the prototypes and the size of the r-value. The lower the dimension of the prototypes, the bigger the r-value should be chosen and vice versa. This can be seen in Figure 9 where 100 classes are classified. One time with 1'000-dimensional prototypes and one time with two-dimensional prototypes. While using 1'000-dimensional prototypes, the validation accuracy scores are higher with an r-value close to 0.01. With two-dimensional prototypes, the results are better with an r-value of 5.0. Moreover, it was observed that the vanishing gradient problem is still apparent when using 1'000 classes and two-dimensional output spaces which can also be seen in the low accuracy scores in Table 7.

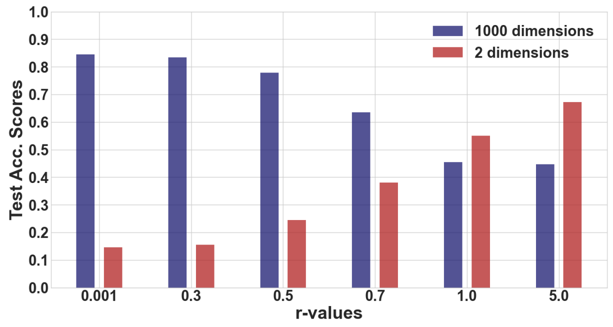


Figure 9: Relationship between r-values and dimensions of prototypes for 100 classes

Visualisation of learning process over time

It is clearly visible that the learning process freezes after the first two epochs in Figure 4. Hence, only the same classes are predicted afterwards. Instead of predicting evenly 15'000 samples per class, represented by the black horizontal line, some of the classes are predicted disproportionately often. Class 598 is predicted 295'791 times. Figure 5 shows the same task with clipping. It is clearly visible that the target of 15'000 predictions per class is closer approximated on average. Nonetheless, class 232 for example, is still 154'684 times predicted.

In contrast to the 1'000-dimensional experiments, within the two-dimensional experiments, it looks as if the classes are selected purely at random, as can be seen in Figure 6. This is caused by the vanishing gradients, which cannot be overcome with clipping in low-dimensional output spaces. However, by looking at Figure 14 in Appendix A, given enough time and using a bigger r-value, it is possible to reach a validation accuracy score of 19%.

Exploration of Hyberbolic Busemann Loss

One expects to find normally distributed loss values for all prototypes, as seen in Figure 13. However, the distributions in Figure 7 and Figure 8 are left-skewed. Moreover, prototypes positioned on and close to the x- and y-axes have remarkable long tails, especially if the data is close to the origin. The distributions get slightly flattened when the data is closer to the ideal boundary, but the tails are still apparent.

Limitations

There are several limitations when it comes to the results of this research. First, the obtained accuracy scores are still below the ones from a softmax network when it comes to 1'000 classes. Therefore, it should be investigated how those ones can be improved. Moreover, the accuracy scores for softmax with 1'000 classes seem to be too high. Even though different datasets were created and the exact same network was used as the one employed for 10 and 100 classes, it resulted in the same scores. Second, the results using two-dimensional output spaces only achieved accuracy scores below 19% for 1'000 classes and show unexpected distributions for the loss values. Last but not least, the synthetic data does not represent real-world data.

7 CONCLUSION

This work demonstrated how Hyperbolic Busemann Learning can be scaled up to 1'000 classes using high-dimensional output spaces by clipping the Euclidean embeddings before the exponential map. This ensures that one can reduce the impact of the vanishing gradient problem during training and thus improve accuracy scores. Whereas r-values of 0.01 or 0.3 should be chosen for 1'000-dimensional output spaces in order to obtain higher accuracy scores, r-values of 1.0 or 5.0 are preferred in two-dimensional output spaces.
Moreover, this research not only sheds light on the performance of using two-dimensional output spaces for 1'000 classes but also for 10 and 100 classes. Accuracy scores below 20% can be reported for 1'000 classes in two-dimensional output spaces since the vanishing gradient problem can not be overcome by clipping the Euclidean embeddings.
In addition, by visualising the number of samples classified after every epoch, it was possible to show how the learning process evolves over time. In high-dimensional output spaces, HBL fails to generalise and predicts some classes disproportionally often. In two-dimensional output spaces, classes are predicted more balanced but also more randomly. Lastly, it was observed that data samples close to the origin have different distributions for their loss values regarding different prototypes. The distributions seem to become more uniform as one moves closer to the edge. Two-dimensional loss values distributions are left-skewed and 1'000-dimensional loss values distributions are normally distributed.

As mentioned before, the synthetic dataset does not represent real-world data. Therefore, a promising way forward from here would be using a real-world dataset, such as ImageNet, which also contains 1'000 classes. Moreover, one should find a way to overcome the vanishing gradient problem with 1'000 classes in two-dimensional output spaces. Finally, it should be investigated what causes the left skewness in the loss distributions for two-dimensional output spaces.

REFERENCES

[1] Mina Ghadimi Atigh, Martin Keller-Ressel, and Pascal Mettes. 2021. Hyperbolic Busemann Learning with Ideal Prototypes. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

[2] Gregor Bachmann, Gary Becigneul, and Octavian Ganea. 2020. Constant Curvature Graph Convolutional Networks. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Hal Daumé III and Aarti Singh (Eds.), Vol. 119. PMLR, 486–496. https://proceedings.mlr.press/v119/bachmann20a.html

[3] Björn Barz and Joachim Denzler. 2019. Deep Learning on Small Datasets without Pre-Training using Cosine Loss. *CoRR* abs/1901.09054 (2019). arXiv:1901.09054 http://arxiv.org/abs/1901.09054

[4] H. Busemann. 1955. *The Geometry of Geodesics*. Number v. 6 in Pure and applied mathematics. Academic Press. https://books.google.nl/books?id=aYSzAAAAIAAJ

[5] Bhuwan Dhingra et al. 2018. Embedding Text in Hyperbolic Spaces. *CoRR* abs/1806.04313 (2018). arXiv:1806.04313 http://arxiv.org/abs/1806.04313

[6] Deng Jia et al. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[7] Frederik Pahde et al. 2020. Multimodal Prototypical Networks for Few-shot Learning. *CoRR* abs/2011.08899 (2020). arXiv:2011.08899 https://arxiv.org/abs/2011.08899

[8] Gulcehre et al. 2018. Hyperbolic Attention Networks. https://doi.org/10.48550/ARXIV.1805.09786

[9] Guo Yunhui et al. 2021. Clipped Hyperbolic Classifiers Are Super-Hyperbolic Classifiers. https://doi.org/10.48550/ARXIV.2107.11472

[10] Kaize Ding et al. 2020. Graph Prototypical Networks for Few-shot Learning on Attributed Networks. *CoRR* abs/2006.12739 (2020). arXiv:2006.12739 https://arxiv.org/abs/2006.12739

[11] Massimiliano Mancini et al. 2019. Knowledge is Never Enough: Towards Web Aided Deep Open World Recognition. *CoRR* abs/1906.01258 (2019). arXiv:1906.01258 http://arxiv.org/abs/1906.01258

[12] Mengye Ren et al. 2018. Meta-Learning for Semi-Supervised Few-Shot Classification. *CoRR* abs/1803.00676 (2018). arXiv:1803.00676 http://arxiv.org/abs/1803.00676

[13] Valentin Khrulkov et al. 2019. Hyperbolic Image Embeddings. *CoRR* abs/1904.02239 (2019). arXiv:1904.02239 http://arxiv.org/abs/1904.02239

[14] Wei Peng et al. 2021. Hyperbolic Deep Neural Networks: A Survey. *CoRR* abs/2101.04562 (2021). arXiv:2101.04562 https://arxiv.org/abs/2101.04562

[15] Wenjia Xu et al. 2020. Attribute Prototype Network for Zero-Shot Learning. *CoRR* abs/2008.08290 (2020). arXiv:2008.08290 https://arxiv.org/abs/2008.08290

[16] Zhang et al. 2020. Hyperbolic graph attention network. In *Proceedings of the AAAI Conference on Ariticial Intelligence*.

[17] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. 2018. Hyperbolic Entailment Cones for Learning Hierarchical Embeddings. *CoRR* abs/1804.01882 (2018). arXiv:1804.01882 http://arxiv.org/abs/1804.01882

[18] Samantha Guerriero, Barbara Caputo, and Thomas Mensink. 2018. DeepNCM: Deep Nearest Class Mean Classifiers. In *International Conference on Learning Representations - Workshop (ICLRw)*.

[19] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. [n.d.]. CIFAR-10 (Canadian Institute for Advanced Research). ([n. d.]). http://www.cs.toronto.edu/~kriz/cifar.html

[20] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. [n.d.]. CIFAR-100 (Canadian Institute for Advanced Research). ([n. d.]). http://www.cs.toronto.edu/~kriz/cifar.html

[21] Qi Liu, Maximilian Nickel, and Douwe Kiela. 2019. Hyperbolic Graph Neural Networks. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/103303dd56a731e377d01f6a37badae3-Paper.pdf

[22] Shaoteng et al. Liu. 2020. Hyperbolic Visual Embedding Learning for Zero-Shot Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[23] Teng et al. Long. 2020. Searching for Actions on the Hyperbole. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1138–1147. https://doi.org/10.1109/CVPR42600.2020.00122

[24] Pascal Mettes, Elise van der Pol, and Cees G M Snoek. 2019. Hyperspherical Prototype Networks. In *Advances in Neural Information Processing Systems*.

[25] Nicholas Monath, Manzil Zaheer, Daniel Silva, Andrew McCallum, and Amr Ahmed. 2019. Gradient-based Hierarchical Clustering Using Continuous Representations of Trees in Hyperbolic Space. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. ACM, New York, NY, USA, 714–722. https://doi.org/10.1145/3292500.3330997

[26] Maximillian Nickel and Douwe Kiela. 2017. Poincaré Embeddings for Learning Hierarchical Representations. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/59dfa2df42d9e3d41f5b02bfc32229dd-Paper.pdf

[27] Wei et al. Peng. 2020. *Mix Dimension in Poincaré Geometry for 3D Skeleton-Based Action Recognition.* Association for Computing Machinery, New York, NY, USA, 1432–1440. https://doi.org/10.1145/3394171.3413910

[28] Jiayi et al. Shen. 2022. Spherical Zero-Shot Learning. *IEEE Transactions on Circuits and Systems for Video Technology* 32, 2 (2022), 634–645. https://doi.org/10.1109/TCSVT.2021.3067067

[29] Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical Networks for Few-shot Learning. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/cb8da6767461f2812ae4290eac7cbc42-Paper.pdf

[30] William Thong, Pascal Mettes, and Cees G. M. Snoek. 2019. Open Cross-Domain Visual Search. *CoRR* abs/1911.08621 (2019). arXiv:1911.08621 http://arxiv.org/abs/1911.08621

[31] Stephan Tulkens. 2021. Hyperspherical Alternatives to Softmax. https://medium.com/towards-data-science/hyperspherical-alternatives-to-softmax-6da03388fe3d.

[32] A.R. Webb. 2002. *Statistical Pattern Recognition.* Wiley. https://books.google.nl/books?id=Gok-AQAAIAAJ

[33] Yunlong Yu, Zhongfei Zhang, and Jungong Han. 2019. Meta-Transfer Networks for Zero-Shot Learning. *CoRR* abs/1909.03360 (2019). arXiv:1909.03360 http://arxiv.org/abs/1909.03360

A ADDITIONAL MATERIAL

Within the additional material, one can find figures which can provide additional insights for the reader. Figure 10 is the same Figure as Figure 4, just using a three-dimensional representation. Figure 11 depicts how this research obtained the loss value distributions. For example, one takes prototype 0, as highlighted in the Figure, and computes all the loss values for each data point with norm 0.1 or 0.9. These loss values can be used for visualising the distributions. Figure 12 shows the loss distributions with data close to the origin (norm 0.1) and Figure 13 shows the same but for data close to the ideal boundary of the Poincaré ball model (norm 0.9) in 1'000-dimensional output spaces. Figure 14 shows the cumulative predictions for 1'000 classes using two-dimensional output spaces with 60 epochs of training which resulted in a validation accuracy score of 19%. For higher accuracy scores, one has to find a way to overcome the vanishing gradient problem for 1'000 classes in two-dimensional output spaces.

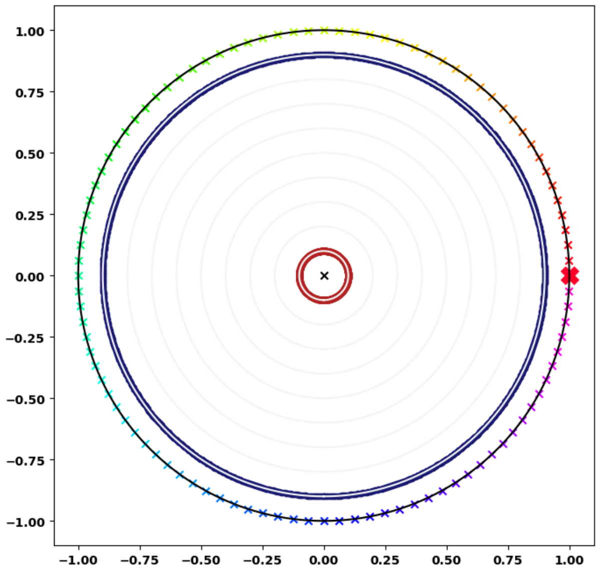


Figure 11: Experimental setup for loss exploration

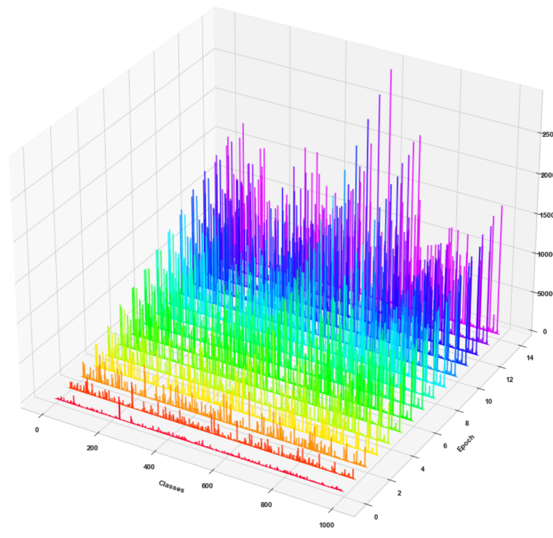


Figure 10: Cumulative predictions during training - No clipping for 1'000 classes using 1'000-dimensional output spaces (Same Figure as Figure 4, but now in 3D)

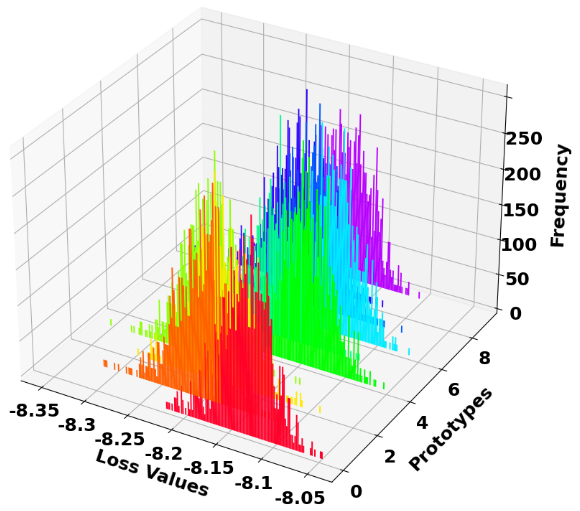


Figure 12: Loss distributions with sample data with norm of 0.1 and 1'000-dimensional output spaces.

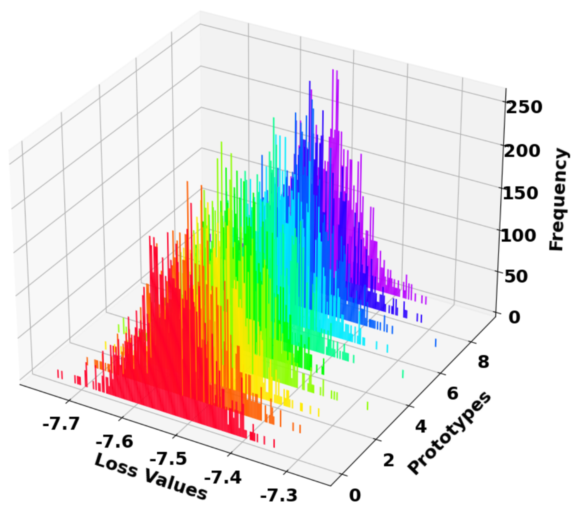


Figure 13: Loss distributions with sample data with norm of 0.9 and 1'000-dimensional output spaces

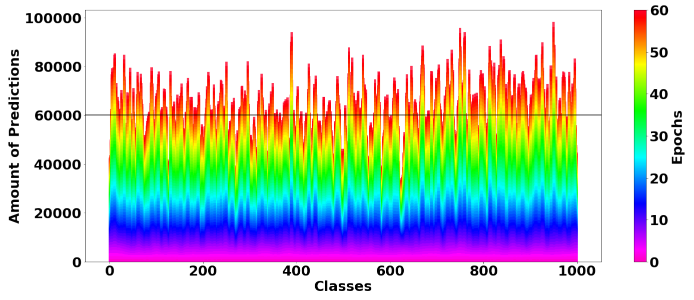


Figure 14: Cumulative predictions during training for 1'000 classes using two-dimensional output spaces with clipping, r = 5.0, 60 Epochs, learning rate of 1e-3, Drop1 at epoch 40, drop 2 at epoch 50. Everything else stayed the same. Resulted in a validation accuracy score of 19%

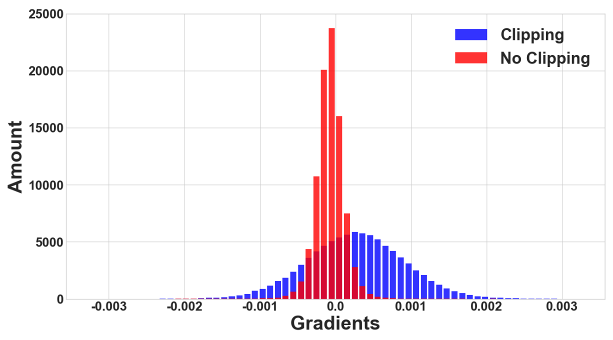


Figure 15: Gradients of the weights of the fully connected layer 1 for 1'000 classes using 1'000-dimensional output spaces

B FIGURES & TABLES

List of Figures

List of Tables