Attention Networks in Hyperbolic Space

Dhruvesh Patel **Praful Johari** **Prashant Ranjan** **Rishabh Gupta**
{dhruveshpate pjohari pranjan rishabhgupta}@umass.edu

1 Problem statement

While a lot of recent research in NLP is focused on using larger and deeper neural networks to achieve better performance on complex tasks, there is another novel and orthogonal approach which promises to give more parsimonious representations by utilizing the geometry of the hyperbolic space instead of Euclidean space.

It has been observed that naturally occurring complex data like natural language, tend to contain highly non-euclidean latent structures. In these situations, the euclidean space isn't expressive enough to provide meaningful and compact representations (Michael M Bronstein and Vandergheynst, 2017).

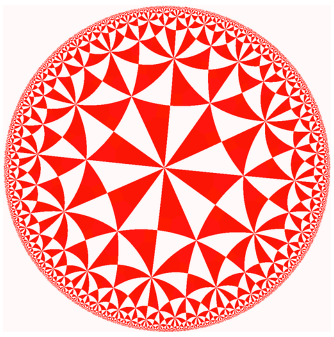
Figure 1: Triangular Hyperbolic tiling in Poincaré disk model [https://en.wikipedia.org/wiki/Poincar%C3%A9_disk_model

In order to see how hyperbolic space might be a better alternative for embedding these naturally occurring hierarchies, we can consider one of the 2-D models for the hyperbolic space, for instance the Poincaré disk model. Figure 1 depicts the triangular tiling in a Poincaré disk. If one interprets this picture in Euclidean space, one could say that the triangles get smaller (have lesser area) as one moves towards the boundaries of the circle. However, in the hyperbolic space, all these triangles have equal area. Which shows that, there is more (infinite) space as one approaches the boundaries of the disk. Such a space can also be seen as the continuous analog of a tree like graph where the number of nodes grow exponentially with depth. Hence, hyperbolic space should allow one to capture hierarchy as well as similarity with lesser number of parameters compared to Euclidean space. Introduced by " Nickel and Kiela (2017)" as a way to incorporate correct modeling bias for representations with inherent hierarchy, this approach has gained popularity and has been extended to a full-fledged neural learning paradigm in " Ganea et al. (2018a)".

In natural language, hierarchies manifest in several ways: relations like hyponymy/hypernymy in nouns and verbs (Figure 2), semantic hierarchy as one goes from a word (more general in meaning) to a phrase to a sentence (more specific in meaning). For the latter case, as shown in Figure 3, if right compositional transformations are used on word representations, one can utilize the hierarchical structure of the hyperbolic space to capture semantic generality/specificity without increasing the dimensionality of the representation space. As pointed out by "Nickel and Kiela (2017)", hyperbolic space is naturally suited for the representation of such a tree-like hierarchy. Taking motivation from these observations, in this work we propose to train an end-to-end hyperbolic neural network to learn and use representations in hyperbolic space to perform a common NLP task like natural language inference. In particular, we wish to propose and analyze attention mechanisms in hyperbolic neural network for this task. We will do this by understanding the geometry of the space and by using the right transformations to utilize its hierarchical nature.

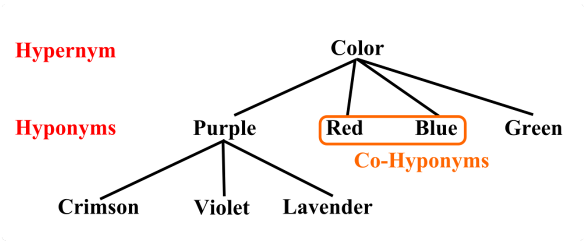

Figure 2: An example of the hierarchy between hyponyms and hypernym [https://en.wikipedia.org/wiki/Hyponymy_and_hypernymy]

2 Related work

In a short time, there has been new and exciting work which aims to use the geometry of hyperbolic space to learn representations of natural language. "Nickel and Kiela (2017)" train hyperbolic embeddings for WordNet (Miller, 1995) and test them for transitive closure and lexical entailment task. "Dhingra et al. (2018)" focus on learning sentence embedding in an unsupervised manner using Skip-Thoughts (Kiros et al., 2015) like framework. "Tay et al. (2018)" use hyperbolic representations of questions and answers to perform Question-Answering task on several different datasets. However, all these approaches use either a projection or reparameterization to convert representations in Euclidean space to representations in hyperbolic space. Moreover, they use neural networks which are partially in Euclidean space and partially in hyperbolic space. "Tifrea et al. (2018)" successfully train hyperbolic word representations from scratch using a GloVe (Pennington et al., 2014) like approach. Going a step further, "Ganea et al. (2018a)" train neural network with states as well as parameters in hyperbolic space to perform tasks like textual entailment and noisy prefix recognition. Continuing on the lines of these last two works, starting from training unsupervised hyperbolic embeddings and using hyperbolic neural network layers, we wish to propose and use hyperbolic attention mechanisms to perform the task of natural language inference. *Note*: Since this is nascent field, we are not able to cite more papers.

3 What you proposed vs. what you accomplished

- ~~Prepare euclidean baseline models for the natural language inference task~~
- *Prepare hyperbolic word embeddings*: We decided to initialize the embeddings randomly within the Poincaré ball and made them tunable. Hyperbolic operations ensure that the embedding vectors remain within the ball.
- ~~Implement basic operations and neural network layers in hyperbolic space~~
- ~~Debug issues in training hyperbolic neural network~~
- ~~Implement and train an attention based model in hyperbolic space for the natural language inference task~~
- ~~Analyze the performance of hyperbolic attention, comparison with the baseline attention model in Euclidean space, experimenting with different initializations and hyperparameters~~
- ~~Analyze the outputs of hyperbolic attention and perform extensive error analysis~~

4 Your dataset

We are evaluating our models by analyzing their performance on the task of natural language inference using the Multi-Genre Natural Language Inference (MultiNLI) (Williams et al., 2017)[1] dataset. This dataset is designed to evaluate a model's ability to understand and reason about natural language sentences. Each sample has a premise which is chosen from different genres (10 in total like fiction,report etc.) and a label and hypothesis is proposed by a crowd worker in response to a premise. The task is to determine whether a hypothesis is true (entailment), false (contradiction), or undetermined (neutral). See Table 11 for an example from various genres. The dataset is comprehensive with 433K samples of hypothesis/premise pairs and exhaustive in its coverage as it contains data from ten different genres of English language (written and spoken) which ensures that it covers the language in almost all its complexity.

For model evaluation we are using held out data in the MultiNLI competition hosted on Kaggle [2].

[1]https://www.nyu.edu/projects/bowman/multinli/

[2]https://www.kaggle.com/c/multinli-matched-open-evaluation/data

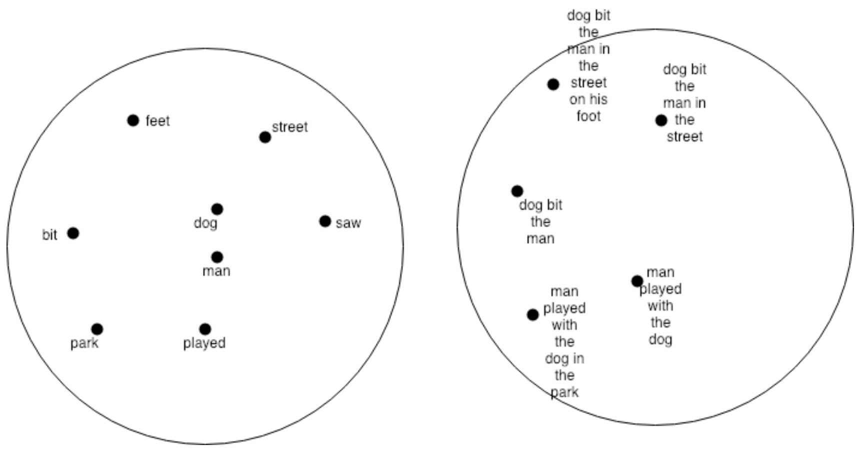
feet
street
bit
dog
saw
man
park
played
dog bit the man in the street on his foot
dog bit the man in the street
dog bit the man
man played with the dog in the park
man played with the dog

Figure 3: Representations for words in hyperbolic space (left) which can be composed to form the representations of phrases and sentences (right). As the specificity increases, the representations move closer to the periphery utilizing the hierarchical nature of the space

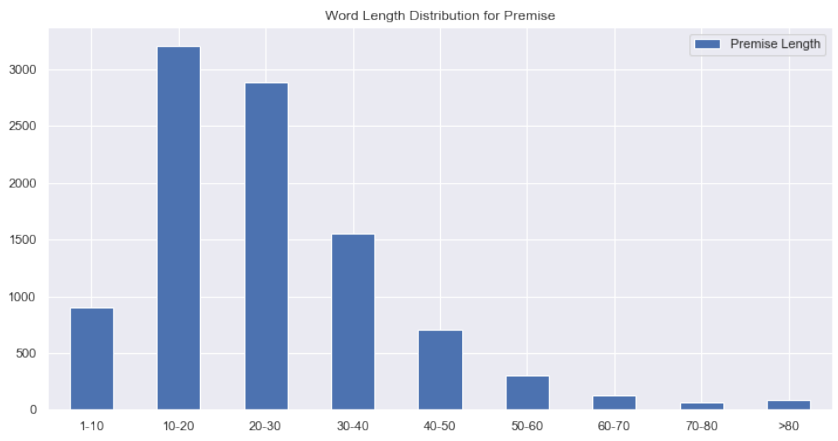

Figure 4: Premise length distribution

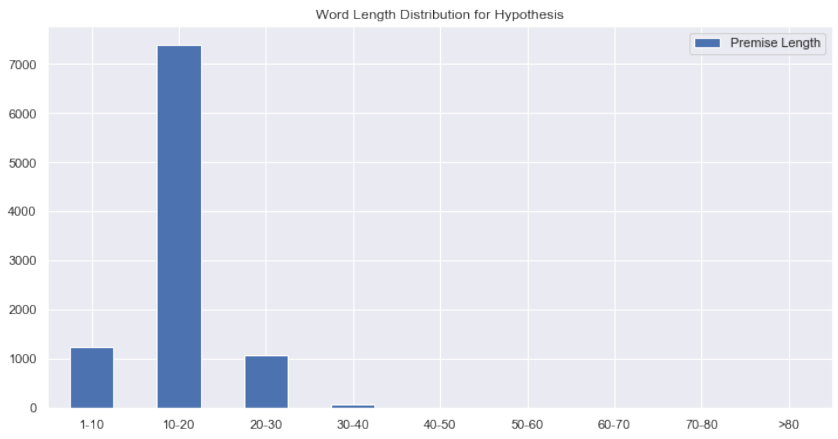

Figure 5: Hypothesis length distribution

Premise	Label	Hypothesis
Your gift is appreciated by each and every student who will benefit from your generosity.	neutral	Hundreds of students will benefit from your generosity.
yes now you know if everybody like in August when everybody's on vacation or something we can dress a little more casual or	contradicts	August is a black out month for vacations in the company.
At the other end of Pennsylvania Avenue, people began to line up for a White House tour.	entails	People formed a line at the end of Pennsylvania Avenue.

Figure 6: Example instances from the MultiNLI dataset

The sentence pairs in the data is already annotated with textual entailment information so no further text annotation is required.

The dataset is challenging because not only does it have a high average sentence length, it also has diverse linguistic phenomena due to the fact that it encompasses various genres consequently making the inference task difficult. Also, the examples in the dataset requires worldly knowledge about entities to infer the sentences which adds up to the complexity of the Natural Language Inference task. Also, according to (Gururangan et al., 2018), the annotation process of MultiNLI dataset introduces certain artifacts which enable the model to make correct classification just by looking at the hypothesis alone. So, the current state-of-the-art models on this dataset are mostly learning such artifacts and not actually learning the entailment relation between premise and hypothesis which still remains a challenging task. We use torchtext library with spacy for data preprocessing.

5 Baselines

We implemented and trained a baseline model in Euclidean space for comparison with its hyperbolic counterpart. We kept 10% of the data aside for testing/analysis and used the rest for training. For a given sample, we obtain a concatenated representation of hypothesis and premise using pretrained GloVe embeddings. We then pass this to a basic network architecture (Figure 7) that consists of two RNN layers, followed by a feed-forward layer and a softmax layer with cross entropy loss which performs 3-way classification. The data is fed in mini-batches of size 128. The weights are updated using Adam optimizer, with a learning rate of 0.01.

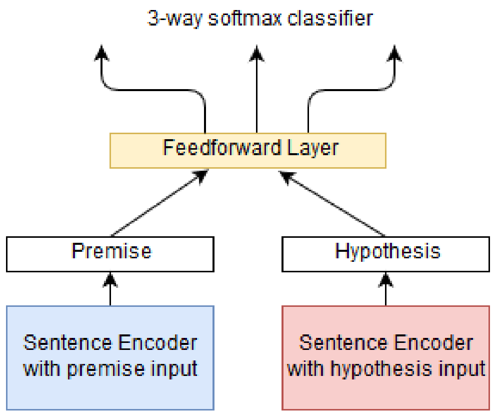
3-way softmax classifier
Feedforward Layer
Premise
Hypothesis
Sentence Encoder with premise input
Sentence Encoder with hypothesis input

Figure 7: Network Architecture

To build more complex encoders, we implemented attention mechanism as shown in the figure(Figure 8). The query vector is the last hidden state of the hypothesis encoder and the key vectors are all hidden states of the premise encoder. We get the weighted representation of the premise hidden states using dot product between query and key vectors as the scoring function. We append this representation to our query vector and feed it to the feedforward layer as before.

The accuracy attained on the baseline models is listed in the Table 9.

6 Your approach

After performing the error analysis of the baseline models (reported in Section 8), we concluded that incorporating inductive bias to capture hierarchical relationships between words and phrases can result in an improved accuracy on the task of natural language inference. Representations in hyperbolic space are naturally suited for this. Continu-

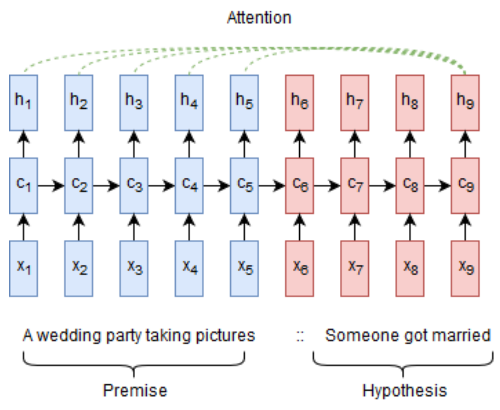

Figure 8: Attention Architecture

ing on the work of "Ganea et al. (2018a)", which proposes hyperbolic analogs of linear (matrix multiplication in Euclidean vector space) and non-linear transformations (pointwise non-linearity), we use the operations in gyrovector space (Ungar, 2005) to come up with a formulation for attention mechanism in the hyperbolic space. We then compare the performance of this attention mechanism with the baseline Euclidean attention analogs, hyperbolic RNNs, and euclidean RNNs, on the task of natural language inference.

We performed the following tasks in order to successfully complete our implementation:

1. Implemented basic algebraic operations in the Poincaré ball as described in Section 7.2
2. Used these operations as building blocks to implement the hyperbolic analogs of euclidean RNN and GRU
3. Proposed and implemented a simple attention mechanism in hyperbolic space as shown in Section 7.4
4. Trained hyperbolic RNN, GRU, Deep Averaging and Attention Model and their corresponding euclidean counterparts on MultiNLI dataset

The details of our approach and implementation are mentioned in Section 7.

All the code used in this project was implemented from scratch by us, and we didn't rely on any external implementation for the same.

As expected, due to the appropriate inductive bias in hyperbolic space to model entailment, our hyperbolic models perform better (in most cases) than their euclidean counterparts, even if we use more number of parameters in the euclidean models. Section 8 explains our results and observations in detail.

7 Implementation

7.1 Tools and compute resources

We use PyTorch for implementing our neural networks and torchtext with spaCy to preprocess the data and word embeddings. The initial baseline models have been trained a on dual core CPU with 8 GB RAM. The hyperbolic RNN is trained on a Tesla M60 GPU on a G3xLarge AWS instance.

7.2 Basic Operations

As defined in "Ganea et al. (2018b)", we implemented the following hyperbolic equivalents of basic euclidean operations in PyTorch.

Addition: The *Möbius addition* of x and y in $\mathbb{D}_c^n$ is defined as

$$x \oplus_c y := \frac{(1 + 2c\langle x, y\rangle + c\|y\|^2)x + (1 - c\|x\|^2)y}{1 + 2c\langle x, y\rangle + c^2\|x\|^2\|y\|^2}.$$

We can see that, when $c = 0$, one recovers the Euclidean addition of two vectors in $\mathbb{R}^n$.

Scalar Multiplication: For $c > 0$, the *Möbius scalar multiplication* of $x \in \mathbb{D}_c^n \setminus \{\mathbf{0}\}$ by $r \in \mathbb{R}$ is defined as

$$r \otimes_c x := (1/\sqrt{c}) \tanh(r \tanh^{-1}(\sqrt{c}\|x\|)) \frac{x}{\|x\|},$$

and $r \otimes_c \mathbf{0} := \mathbf{0}$. Note that, one recovers the Euclidean counterpart when c goes to zero: $\lim_{c \to 0} r \otimes_c x = rx$.

Distance: If we define the generalized hyperbolic metric tensor g^c as the metric conformal to the Euclidean one, with conformal factor $\lambda_x^c := 2/(1 - c\|x\|^2)$, then the induced distance function on $(\mathbb{D}_c^n, g^c)$ is given by

$$d_c(x, y) = (2/\sqrt{c}) \tanh^{-1}\left(\sqrt{c}\|-x \oplus_c y\|\right).$$

Again, observe that $\lim_{c \to 0} d_c(x, y) = 2\|x - y\|$, *i.e.* we recover Euclidean geometry in the limit.

Exponential and Logarithmic Maps: For any point $x \in \mathbb{D}_c^n$, the exponential map $\exp_x^c : T_x\mathbb{D}_c^n \to \mathbb{D}_c^n$ and the logarithmic map $\log_x^c$:

$\mathbb{D}_c^n \to T_x\mathbb{D}_c^n$ are given for $v \neq \mathbf{0}$ and $y \neq x$ by:

$$\exp_x^c(v) = x \oplus_c \left(\tanh\left(\sqrt{c}\frac{\lambda_x^c\|v\|}{2}\right)\frac{v}{\sqrt{c}\|v\|}\right),$$
$$\log_x^c(y) = \frac{2}{\sqrt{c}\lambda_x^c}\tanh^{-1}(\sqrt{c}\|-x\oplus_c y\|)\frac{-x\oplus_c y}{\|-x\oplus_c y\|}.$$

Softmax: Given K classes and $k \in \{1,\dots,K\}$, $p_k \in \mathbb{D}_c^n$, $a_k \in T_{p_k}\mathbb{D}_c^n \setminus \{\mathbf{0}\}$:

$$p(y=k|x) \propto \exp\left(\frac{\lambda_{p_k}^c\|a_k\|}{\sqrt{c}}\sinh^{-1}\left(\frac{2\sqrt{c}\langle -p_k\oplus_c x, a_k\rangle}{(1-c\|-p_k\oplus_c x\|^2)\|a_k\|}\right)\right),$$
$$\forall x \in \mathbb{D}_c^n.$$

Notice that when c goes to zero, this goes to $p(y=k|x) \propto \exp(4\langle -p_k + x, a_k\rangle) = \exp((\lambda_{p_k}^0)^2\langle -p_k+x, a_k\rangle) = \exp(\langle -p_k+x, a_k\rangle_0)$, recovering the usual Euclidean softmax.

Matrix-Vector Multiplication If $M : \mathbb{R}^n \to \mathbb{R}^m$ is a linear map, which we identify with its matrix representation, then $\forall x \in \mathbb{D}_c^n$, if $Mx \neq \mathbf{0}$ we have

$$M^{\otimes_c}(x) = (1/\sqrt{c})\tanh\left(\frac{\|Mx\|}{\|x\|}\tanh^{-1}(\sqrt{c}\|x\|)\right)\frac{Mx}{\|Mx\|},$$

and $M^{\otimes_c}(x) = \mathbf{0}$ if $Mx = \mathbf{0}$.

Möbius version of a function: In general, for any $f : \mathbb{R}^n \to \mathbb{R}^m$, the *Möbius version of* f is given by the map from $\mathbb{D}_c^n$ to $\mathbb{D}_c^m$ by:

$$f^{\otimes_c}(x) := \exp_{\mathbf{0}}^c(f(\log_{\mathbf{0}}^c(x))),$$

where $\exp_{\mathbf{0}}^c : T_{\mathbf{0}_m}\mathbb{D}_c^m \to \mathbb{D}_c^m$ and $\log_{\mathbf{0}}^c : \mathbb{D}_c^n \to T_{\mathbf{0}_n}\mathbb{D}_c^n$. Note that similarly as for other Möbius operations, we recover the Euclidean mapping in the limit $c \to 0$ if f is continuous, as $\lim_{c\to 0} f^{\otimes_c}(x) = f(x)$.

Concatenation of multiple input vectors: If we are given $x_1 \in \mathbb{D}_c^n$, $x_2 \in \mathbb{D}_c^p$, $x = (x_1\ x_2)^T \in \mathbb{D}_c^n \times \mathbb{D}_c^p$, and M, M_1, M_2, then we define $M \otimes_c x := M_1 \otimes_c x_1 \oplus_c M_2 \otimes_c x_2$. Note that when c goes to zero, we recover the Euclidean formulation, as $\lim_{c\to 0} M \otimes_c x = \lim_{c\to 0} M_1 \otimes_c x_1 \oplus_c M_2 \otimes_c x_2 = M_1x_1 + M_2x_2 = Mx$.

7.3 Hyperbolic RNNs

We used the above operations to implement and train hyperbolic equivalents of common RNN architectures from scratch in PyTorch.

Vanilla RNN: For parameters $W \in \mathcal{M}_{m,n}(\mathbb{R})$, $U \in \mathcal{M}_{m,d}(\mathbb{R})$, $b \in \mathbb{D}_c^m$, a vanilla RNN can be defined as:

$$h_{t+1} = \varphi^{\otimes_c}(W \otimes_c h_t \oplus_c U \otimes_c x_t \oplus_c b),$$

$$h_t \in \mathbb{D}_c^n,\ x_t \in \mathbb{D}_c^d.$$

where $\varphi^{\otimes_c}$ is the Möbius version of a pointwise non-linearity like ReLU, tanh etc.

GRU: Similarly, we also adapted the GRU architecture. The reset gate r_t is given by

$$r_t = \sigma \log_{\mathbf{0}}^c(W^r \otimes_c h_{t-1} \oplus_c U^r \otimes_c x_t \oplus_c b^r),$$

and similarly for the update gate z_t. The intermediate hidden state becomes:

$$\tilde{h}_t = \varphi^{\otimes_c}((W\text{diag}(r_t)) \otimes_c h_{t-1} \oplus_c U \otimes_c x_t \oplus b),$$

Finally, we propose to adapt the update-gate equation as

$$h_t = h_{t-1} \oplus_c \text{diag}(z_t) \otimes_c (-h_{t-1} \oplus_c \tilde{h}_t).$$

Note that when c goes to zero, one recovers the usual GRU.

7.4 Hyperbolic Attention

The main task while defining any soft attention mechanism is designing the scoring function which takes in a query and a set of keys and returns a score for every key. The output, attended representation, is the weighted sum of value vectors where the weights are the scores computed in the previous step. Intuitively, the scoring function measures a relationship, be it non-directional (similarity) or directional (entailment), between a query and a key. In our case, the query, key and value vectors are hyperbolic representations of words, phrases and sentences and we wish to measure similarity using the attention. Once we have the scores which measure similarity, the weighted mobius addition of values will compose the values in a manner that composed representation will move towards the ball boundary if it is more specific, in meaning, than its constituents or towards the center if it is less specific than its constituents.

If we use the attention defined as above, the different directions (in the tangent space at zero $T_{\mathbf{0}_n}\mathbb{D}^n_c$) in the poincare ball will represent different semantic meaning while the radial distance from the center, will represent generality (closer to center) and specificity (closer to ball boundary).

Extending the definition of mobius version for a function described in (7.2), we define, for $q, k \in \mathbb{D}^n_c$, the scoring function, $s^{\otimes_c} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$ as shown in (1),

$$s^{\otimes_c}(q, k) = \frac{< \log^c_{\mathbf{0}}(q), \log^c_{\mathbf{0}}(k) >}{\| \log^c_{\mathbf{0}}(k) \|} \tag{1}$$

where, $< .,. >$: $\mathbb{R}^n \to \mathbb{R}^+$ is the usual euclidean inner product.

Once the scores $(s_1, \ldots, s_m)$ are computed for keys $(k_1, \ldots, k_m)$, we scale and mask these appropriately so that they represent valid probability weights $(p_1, \ldots, p_m)$. The attended representation obtained is $\sum_{i=1}^m p_i \otimes_c v_i$, where $v_i \in \mathbb{D}^n_c$ are the value vectors. In our models, the value vectors and the key vectors are the same.

7.5 Optimization

Adam for euclidean parameters As shown before, the parameters for all the linear transformations are in euclidean space. We use Pytorch's Adam (Kingma and Ba, 2014) optimizer to optimize these parameters.

RSGD for hyperbolic parameters The biases and word embeddings are in hyperbolic space. To optimize these, we implement our own Riemannian Stochastic Gradient Descent, by using the fact that gradients in the hyperbolic space can be obtained by scaling the euclidean gradients (Ganea et al., 2018b) as shown in (2).

$$\nabla^{\mathbb{D}^n_c}_x L = \frac{1}{\lambda^c_x} \nabla^{\mathbb{R}^n}_x L \tag{2}$$

7.6 Issues and challenges

Slow training - As shown in (2), we need to transform the euclidean gradients of hyperbolic parameters, including the word embeddings, before updating the parameters using them. Since, the updates to the embeddings are sparse, one should ideally be using a sparse tensor to compute the updates efficiently, however, pytorch does not support operations involving both sparse and dense tensors. Hence, we had to use dense tensor for the gradients of the embeddings. This causes, about 5x increase in training time.

Precision - The presence of transcendental functions and their inverses in the operations in the neural network cause training issues – particularly, undefined gradients. These issues were resolved by making sure that the hyperbolic vectors do not lie on the ball boundary or become exactly zero and by using double precision numbers throughout the network. But this also had the unwanted consequence of even slower training.

Vanishing gradients - Another issue while training is that of vanishing gradients. We are observing very slow (almost stalled) training. We hypothesize that this could be due to negligible gradients.

Order of operands - Since the code is very complex, the interactions between operations are hard to imagine and even harder to debug. We wrote the code of GRU and RNN from scratch, converting the euclidean operations to the hyperbolic operations from the ground up. However, since the operations are not commutative, even a small detail like reversing the order of operands will impact the training in an unforeseen way which otherwise won't be an issue with the euclidean counterpart. The basic strategy of debugging these networks by setting c = 0 didn't help in this particular case since the operations would be incorrect only in hyperbolic space. This prohibited us from going all out and experimenting with extremely complex architectures from the start.

8 Results

The accuracies of our hyperbolic models and their corresponding euclidean counterparts on the task of NLI using dataset MultiNLI are reported in Table 9. **It has to be noted that the embedding dimension and the hidden dimensions used in the euclidean models is 100 while those used in the hyperbolic models is 5.** We had to increase the size of the euclidean models to get strong baselines because we are not using complex architectures.

As seen in Table 9, the hyperbolic networks consistently do better than their euclidean counterparts even with more parameters. However, we observed that adding hyperbolic attention to our hyperbolic models does not improve its performance much. This could be due to the simple nature of our scoring function (Eq. 1). Another thing to note is that hyperbolic Deep Averaging Net-

Architecture/Accuracy	Euclidean(baseline)	Hyperbolic
DAN	53.21	**59.36**
GRU	56.36	**58.2**
GRU+Attention	60.54	**61.92**

Figure 9: Accuracy of Euclidean(Baseline) and Hyperbolic models on MNLI test set

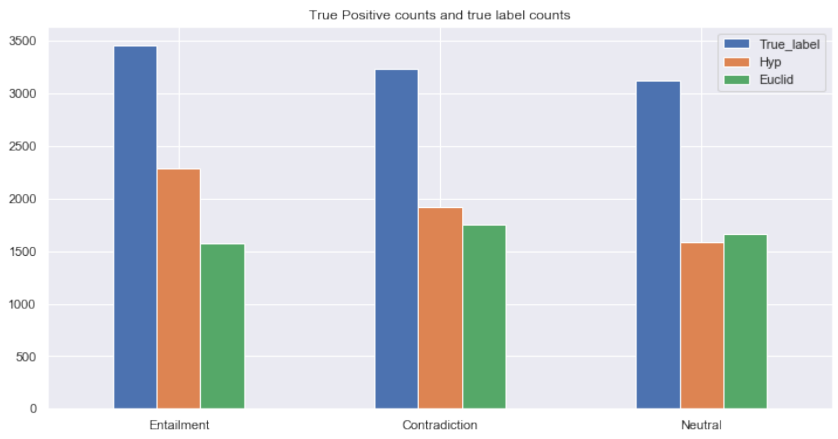

Figure 10: Actual class distribution vs true positive instances in DAN (Euclidean) and DAN (Hyperbolic) networks

work (Hyp DAN) performs much better than the euclidean DAN. The reason for this is that hyperbolic DAN is not a bag of words model. The mobius addition $\oplus$ is not commutative hence while computing the average of word representations in hyperbolic space, the result is dependent on the order of words.

9 Error Analysis

9.1 Errors in baselines

Inferring entailment has been one of the long standing issues in the field of NLP and this fact was underscored by analyzing the misclassified examples. We observed that our baseline model generally fails for entailment class because it gets confused between the entailment and neutral classes. For e.g. *although future medicare costs are expected to consume a growing share of the federal budget and the economy pressure is mounting to expand medicare s benefit package to cover prescription drugs which will add billions to medicare program costs/most people surveyed have said that they do want prescription drugs to be covered.* For this, the true label is **neutral** but the model prediction is **entailment**. Furthermore, *the corrida always starts on time even if the clock occasionally has to be stopped/the corrida always begins when scheduled'* has true label as **entailment**, and the model prediction as **neutral**.

Also, the model fails at examples which require worldly knowledge about entities. For instance, when it sees the *broker-dealer*, it assumes it is related to car sales, whereas it can also be related to trading desks. Additionally, longer sentence length for inference task results in bad performance. For instance, when the sentence length is higher than 25, the model loses track of context and makes random predictions. e.g: *coverage of federal employees 590 employer entity contributions to pension and other retirement benefit plans for federal employees employer entity contributions to social insurance programs employer entity payments for unemployment benefits and workers compensation 590 federal employee contributions to health benefits plan for current coverage of federal employees federal employee contributions to pension and other retirement benefit plans fees on post 1991 direct loans and loan guarantees fines and penalties /individual income taxes corporation income taxes social insurance taxes and contributions excise employee contributions can be a significant figure in this case'* For the above example, The true label is *neutral* which is not correct. After studying the dataset, we also observed that despite leveraging the distributional information from GloVe embeddings, the model fails at analyzing the lexical relationships, like hypernymy.

Premise	**Hypothesis**	**Label**	**Euclidean**	**Hyperbolic**
Well , my brother had a , **cow** , tied out , and a **goat** , the **cow 's offspring** .	My brother owned a few **animals** .	entailment	neutral	entailment
Lush , green fields that stretched for miles , wisps of white clouds scattered across a clear , blue sky , tranquil **silence broken only by cheerful bird songs** , the fresh , sweet scent of **morning** air .	**The birds were making noise early in the day** .	entailment	neutral	entailment
Controllers would notify their supervisors , who in turn would inform management all the way up to FAA headquarters in **Washington** .	The FAA managers then relayed the information to the **white house** .	neutral	entailment	neutral
When they 're **running** with the wind in their faces and the sun on their backs , or stepping barefoot onto cool , green grass .	They are having a barefoot **race** .	neutral	entailment	neutral
Like **in most schools** in Charlotte , if you go on other sides , **it 's like everybody 's just white** , but **in my school , there was every kind of person you could ever meet** .	**My school** was filled with people **who were all extremely alike** .	contradiction	neutral	contradiction
He had a pillow , one of those big pillows , and it looked **red , all of it with blood and swollen** like this . So he said , from the running down from several places , but the one that , this one [indicates eyebrow] , was , was too much .	His pillow was **immaculately white and clean** despite the multiple injuries he sustained .	contradiction	neutral	contradiction

Figure 11: Example instances from the MultiNLI dataset where hyperbolic model performs better than euclidean model

9.2 Error Analysis of Hyperbolic Models

In the table, we have handpicked various examples which capture generality/specificity relationship of premise and hypothesis. It can be seen that the hyperbolic models is able to better learn such directional relationship because of its geometry and able to correct mistakes made by its euclidean counterparts.

10 Contributions of group members

- Dhruv: hyperbolic implementation, debugging and resolving training issues
- Praful: hyperbolic implementation, debugging and resolving training issues
- Prashant: euclidean baseline, evaluation scripts, data and error analysis
- Rishabh: euclidean baselines implementation, incorporating attention mechanism, error analysis

11 Conclusion

We observed the following improvements while working with Hyperbolic Neural Networks:

- **Expressive power:** A Hyperbolic Neural Network with hidden dimension size of 5 has similar expressive power of equivalent euclidean networks with hidden dimension size of 100 when the data has partial ordering like in the case of NLI.
- **Order preservance:** Since mobius addition is neither commutative nor associative, the hyperbolic neural networks defined in terms of mobius operations are able to learn the order of words in a sentence even with networks like Hyperbolic Deep Averaging Networks. We observed that Hyperbolic Deep Averaging Networks perform almost as good on the task of NLI as Hyperbolic GRU and Euclidean GRU.
- **Inherent non-linearities:** Transformations in hyperbolic space are inherently non linear, which enables us to skip the additional non linearities required when operating in euclidean space.

Future work: This is a field in its nascent stages and a lot of work can be done. If we could continue working on this project, we would like to do the following things:

- **Adam optimizer:** We will first create a Hyperbolic version of Adam optimizer, which will greatly reduce our training time.
- **Better pretrained embeddings:** We would like to first create pretrained embeddings in this model of Hyperbolic space, and then create a corresponding version of ELMo like embeddings which should result in comparable performance to the SoTA on most NLP tasks.
- **More complex architectures:** We would also like to implement more complex architectures like Transformers in this space.
- **Better/Unique architectures:** Finally, we would like to find out ways to use hyperbolic embeddings in ways that don't work as well with Euclidean embeddings. For example, we hypothesize that because of the inherent order preservance nature and non linearities, a simple DAN can actually be converted to a version of Recurrent Neural Network with minimal changes, which might even be faster/better than a one to one conversion of euclidean RNN to hyperbolic space.

References

Dhingra, B., Shallue, C. J., Norouzi, M., Dai, A. M., and Dahl, G. E. (2018). Embedding text in hyperbolic spaces. *arXiv preprint arXiv:1806.04313.*

Ganea, O., Becigneul, G., and Hofmann, T. (2018a). Hyperbolic neural networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 5345–5355. Curran Associates, Inc.

Ganea, O.-E., Bcigneul, G., and Hofmann, T. (2018b). Hyperbolic entailment cones for learning hierarchical embeddings.

Gururangan, S., Swayamdipta, S., Levy, O., Schwartz, R., Bowman, S., and Smith, N. A. (2018). Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Michael M Bronstein, Joan Bruna, Y. L. A. S. and Vandergheynst, P. (2017). Geometric deep learning: going beyond euclidean data. In *IEEE Signal Processing Magazine*.

Miller, G. A. (1995). Wordnet: A lexical database for english. *COMMUNICATIONS OF THE ACM*, 38:39–41.

Nickel, M. and Kiela, D. (2017). Poincaré embeddings for learning hierarchical representations. In *NIPS*.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Tay, Y., Tuan, L. A., and Hui, S. C. (2018). Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 583–591. ACM.

Tifrea, A., Bécigneul, G., and Ganea, O. (2018). Poincaré glove: Hyperbolic word embeddings. *CoRR*, abs/1810.06546.

Ungar, A. A. (2005). *Analytic hyperbolic geometry: Mathematical foundations and applications*. World Scientific.

Williams, A., Nangia, N., and Bowman, S. R. (2017). A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.