# The orbits of neural networks

William Chuang

October 5, 2023

## 1 Introduction

Inspired by [10, 3, 4], consider the following question: let a training set and a trained neural network be given, can the neural network be trained to reach a global minimum on its loss surface?

Furthermore, today, a large language model (LLM) could have trillions of parameters. If an LLM has a bug, e.g. it may give a false output to a certain set of inputs, is there a way to generate infinitely many equivalent models so that a model that gives false output can be replaced immediately without a further cost to start over to train a new model?

**Definition 1.** *Two models are equivalent if they both give identical outputs on a fixed training set as the input.*

Moreover, to use random matrix theory to study the eigenvalue distribution of the model, or to study the dynamics between the distributions of input and the distributions of the model, it takes time to generate samples of trained models. Not only that, if models are trained parallel, then they could not always be equivalent given there are multiple local minima. Hence, is it correct to collect models as samples to find their distribution or their eigenvalue distribution if they are not equivalent?

Hence, here is the **the essential question** that answered by this paper: is there a way to generate (infinitely many) equivalent models of a given (trained) model and each generated equivalent model can have non-trivial feedback gradients during the training mode using back-propagation?

It is necessary to ask this question so that the trained model can be understood better to a point that the distribution of its weights can be studied using random matrix theory, and if all the other equivalent models can be derived instantaneously, then this can be considered an alternative to apply quantum computing to find all those equivalent solutions in parallel.

If a model can be replaced by another equivalent model, then the other model might be at the different point on the loss surface, then the barrier between the local minimum to the nearest lower local minimum might have a lower height or in some better case with a negative height, meaning the model can be trained to a better model with a smaller error by using a new set of training data to descent to that lower local minimum. Hence, it is also sufficient to ask the question: how to find equivalent models of a given trained model? If the answer to this question is unknown, suppose a trained model cannot distinguish the fake data generated by an adversary model from the true data, or if a model needs to go through a debugging process for some false classified input, then the model not only has to be retrained, but it cannot be justified it is the best solution due to the process to reach the solution is random. Thus, on the contrary, if all of its equivalent models can be found immediately, then chances are, within the set of equivalent models, there might exist a model that will not take the fake input or falsely classified a corner case, meaning a model can be replaced instantly without going through a new training process.

## 2 Hyperbolic-orbit Algorithm

Every entry of the matrices $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{d\times m}$ can be embedded into the upper half-plane $\mathbb{H}^2$. That is, the entries of $A$ and $B$ become complex numbers with positive imaginary part: $a_{ki} \in \mathbb{H}^2$ and $b_{il} \in \mathbb{H}^2$.

The necessary condition to use hyperbolic geometry and Kleinian groups is that since that hyperbolic geometry can also reach the goal that to generate infinitely many equivalent models, and Kleinian groups are isometry group that preserved not only hyperbolic lengths, but angles which can work as the symmetric group of the space, and to act on geometric objects in the space to do rigid transformation.

Furthermore, the sufficient condition to apply hyperbolic geometry and Kleinian groups is

that instead of linear transformations, linear fractional transformations (which is determined by elements in the Kleinian group) can provide non-linear and hence non-trivial feedback in back-propagation. Though it is possible to develop a similar algorithm on other Riemannian manifold that has an automorphism group and if the group elements can also preserve angles and distance, hyperbolic geometry with Kleinian groups might be the easiest example that can be computable, but give non-trivial feedback in back-propagation.

Define a new binary operation between $A$ and $B$ using hyperbolic length on the upper-half plane $\mathbb{H}^2$[2]:

$$\left(A \odot_{\mathbb{H}^2} B^t\right)_{kl} := \sum_{i=1}^{m} d_{\mathbb{H}^2}\left(a_{ki}, b_{il}\right).$$

Then on the upper-half plane, the projective special linear group $\mathrm{PSL}(2,\mathbb{R})$ is its automorphism group. Furthermore, elements of the group are conformal mapping that preserve not only angles but hyperbolic distance. Hence, $\mathrm{PSL}(2,\mathbb{R})$ is also the isometry group of the upper-half plane.

To derive an equivalent model, take any $M \in \mathrm{PSL}(2,\mathbb{R})\backslash\{\mathrm{id}\}$, then there exists an isomorphism to map $M$ to a linear fractional (Möbius) map $T_M(z) = \frac{az+b}{cz+d}, a, b, c, d \in \mathbb{R}, z \in \mathbb{C}$.

**Definition 2** (Hyperbolic-length product)**.** *Define the hyperbolic-length product is the following new operation:*

$$\left(T_M(A) \odot_{\mathbb{H}^2} T_M(B^t)\right)_{kl} := \sum_{i=1}^{m} d_{\mathbb{H}^2}\left(T_M\left(a_{ki}\right), T_M\left(b_{il}\right)\right).$$

**Definition 3.** *Let $W$ be a neural network model and assume $W$ converged already with the hyperbolic-length product implemented. Then with the above notations, the orbit of $W$ of the given weights $\{a_{ki}, b_{il}\}$ (points in the hyperbolic space) is the set:*

$$\{T_M(a_{ki})\} \cup \{T_M(b_{il})\}$$

*for all $M \in PSL(2,\mathbb{R})$, and for all $k \in \{1,...,n\}$, $l \in \{1,...,d\}$, and $i \in \{1,...,m\}$.*

**Proposition 1.** *With the above notations, then the newly defined product is an invariant:*

$$\left(T_M(A) \odot_{\mathbb{H}^2} T_M(B^t)\right)_{kl} = \sum_{i=1}^{m} d_{\mathbb{H}^2}\left(T_M\left(a_{ki}\right), T_M\left(b_{il}\right)\right) = \sum_{i=1}^{m} d_{\mathbb{H}^2}\left(a_{ki}, b_{il}\right) = \left(A \odot_{\mathbb{H}^2} B^t\right)_{kl}.$$

*Proof.* This result follows immediately from the property of the Möbius mapping $T_M$ when it is corresponding to a matrix $M$ in the projective special linear group $\mathrm{PSL}(2,\mathbb{R})$, then $T_M$ is an isometry under the hyperbolic metric. □

Let $\mathbb{B}^2 := \{z \in \mathbb{C} : |z| < 1\}$ be the Poincare's disc and $\Psi$ be the Cayley's transformation $\Psi : \mathbb{H}^2 \to \mathbb{B}^2$. Then, $\Psi\left(T_M(z)\right) = \frac{a'z+b'}{c'z+d'}$ where $a', b', c', d' \in \mathbb{C}, z \in \mathbb{C}$.

**Proposition 2.** *With the above definition and proposition, then the value of the product has the same value when the points $a_{ki}$ and $b_{il}$ are sent to Poincare's disc:*

$$\left(T_M(A) \odot_{\mathbb{H}^2} T_M(B^t)\right)_{kl} = \sum_{i=1}^{m} d_{\mathbb{H}^2}\left(T_M\left(a_{ki}\right), T_M\left(b_{il}\right)\right) = \sum_{i=1}^{m} \rho_{\mathbb{B}^2}\left(\Psi\left(T_M\left(a_{ki}\right)\right), \Psi\left(T_M\left(b_{il}\right)\right)\right).$$

*Proof.* Since for each $z_1, z_2 \in \mathbb{H}^2$, $d_{\mathbb{H}^2}(z_1, z_2) = \rho_{\mathbb{B}^2}(\Psi(z_1), \Psi(z_2))$. □

The 2-dimensional hyperbolic-length product can be generalized to $n$-dimension[5, 1, 2, 6, 7, 8], for $n \geq 3$. Let $A, B$ be embedded into $\mathbb{H}^n, n \geq 3$.

**Definition 4** (Hyperbolic-length product)**.** *Define n-dime the hyperbolic-length product is the following new operation:*

$$\left(T_M(A) \odot_{\mathbb{H}^n} T_M(B^t)\right)_{kl} := \sum_{i=1}^{m} d_{\mathbb{H}^n}\left(T_M\left(a_{ki}\right), T_M\left(b_{il}\right)\right).$$

This $n$-dimension result could be useful for applying the hyperbolic-orbit algorithm to any trained models that were trained without implementing the hyperbolic-length product during the training. Likewise, for higher dimensions, the hyperbolic-length product is also an invariant:

**Proposition 3.** *With the above notations, then the newly defined product is an invariant:*

$$\left(T_M(A) \odot_{\mathbb{H}^n} T_M(B^t)\right)_{kl} = \sum_{i=1}^{m} d_{\mathbb{H}^n}\left(T_M\left(a_{ki}\right), T_M\left(b_{il}\right)\right) = \sum_{i=1}^{m} d_{\mathbb{H}^n}\left(a_{ki}, b_{il}\right) = \left(A \odot_{\mathbb{H}^n} B^t\right)_{kl}.$$

The question that aimed to solve by using this algorithm was mentioned in the first section. The goal is to replace matrix multiplication with a caveat that the situation mostly does not include finding inverses.

In summary, there are two cases to apply the hyperbolic-orbit algorithm. Let a model $W$ and a training set $X$ be given.

**Definition 5.** *Hyperbolic-orbit Algorithm is an algorithm that implements hyperbolic-length product to replace matrix products.*

The following is the hyperbolic-orbit algorithm introduced by the paper. There could be more than one way to implement hyperbolic-length product.

Case 1 $W$ was trained.

Step 2 Embed entries of $AB$ to $\mathbb{H}^2$. Denote the embedding function by $\phi$.

Step 2 If $AB$ is not invertible, then extend it to a $2n \times 2n$ matrix which is invertible: $\begin{pmatrix} AB & I \\ I & 0 \end{pmatrix}$.

If $AB$ is invertible, then denote it by $M := AB$, if not then denote $M := \begin{pmatrix} AB & I \\ I & 0 \end{pmatrix}$.

Step 3 Randomly generate an invertible real positive matrix $M''$ such that $C := M'^{-1}M''$ and entries of $C$ is in $\mathbb{H}^2$.

Step 4 (i) If $C$ is not invertible, then extend $C$ to $C' := \begin{pmatrix} C & I \\ I & 0 \end{pmatrix}$, $M'$ to $M'_I := \begin{pmatrix} M' & I \\ I & 0 \end{pmatrix}$, $M''$ to $M''_I := \begin{pmatrix} M'' & I \\ I & 0 \end{pmatrix}$.

(ii) If $C$ is invertible, rename $C$ to $C'$, $M'$ to $M'_I$, and $M''$ to $M''_I$.

Case 2 $W$ was not trained.

Method 1 Treat $W$ as trained and reduce it to either case 1.

Method 2 Train $W$ without using hyperbolic-length product till it converges, then treat it as trained, i.e. again, apply methods of case 1.

Method 3 Implement the hyperbolic-length product into $W$ directly.

Step 1 The selected matrices $A$ and $B$ can be embedded freely to hyperbolic space as long as the hyperbolic-length product can work properly in the usual training process.

Step 2 The dimension of hyperbolic space could start with two dimension to optimize the memory usage. If there is no restriction on the use of memory, then the easiest way is starting with dimension $n \times d$, where $\times$ is the scalar product in $\mathbb{N}$.

Figure 1: Gradients of the hyperbolic-length product.

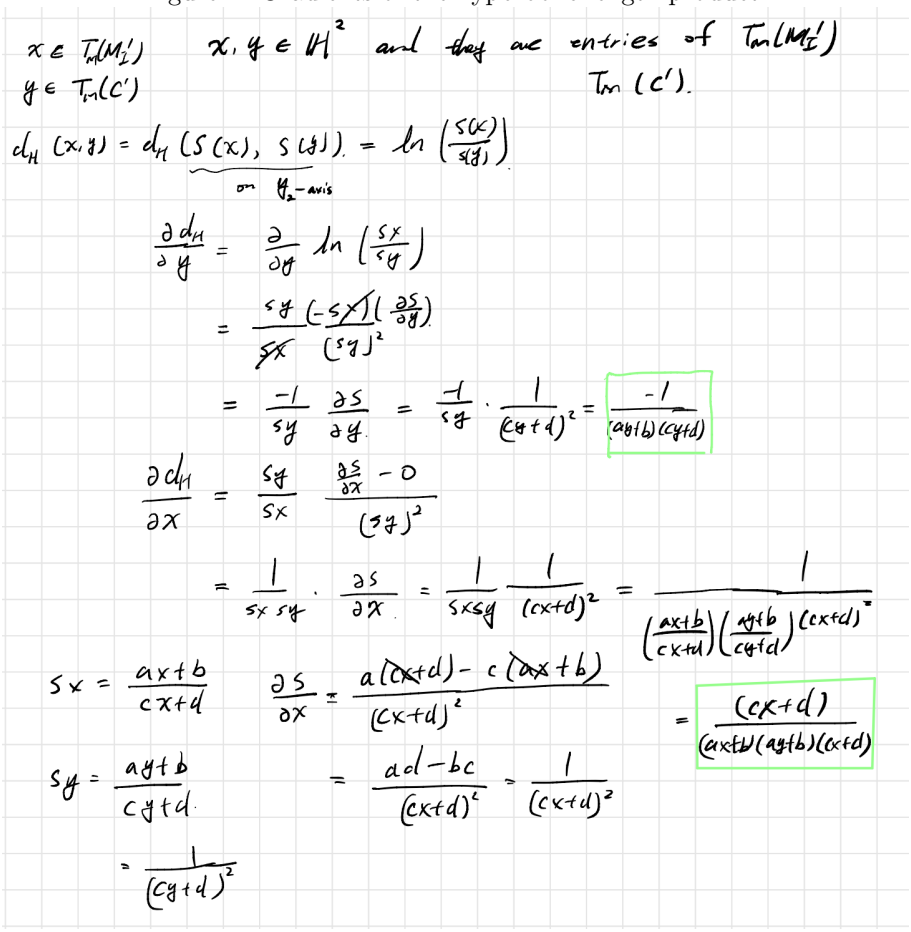


Figure 2: The computational graph of Case 2 when both $AB$ and $C$ are invertible.

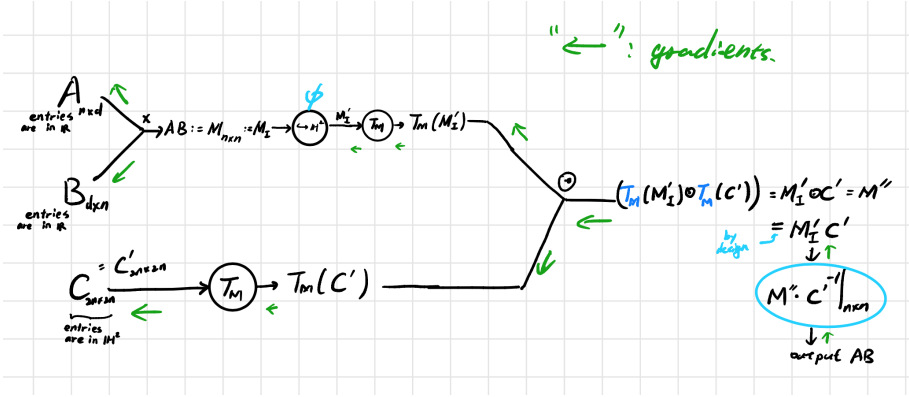



Figure 3: The computational graph of Case 2 when $C$ is invertible, and $AB$ is not invertible.

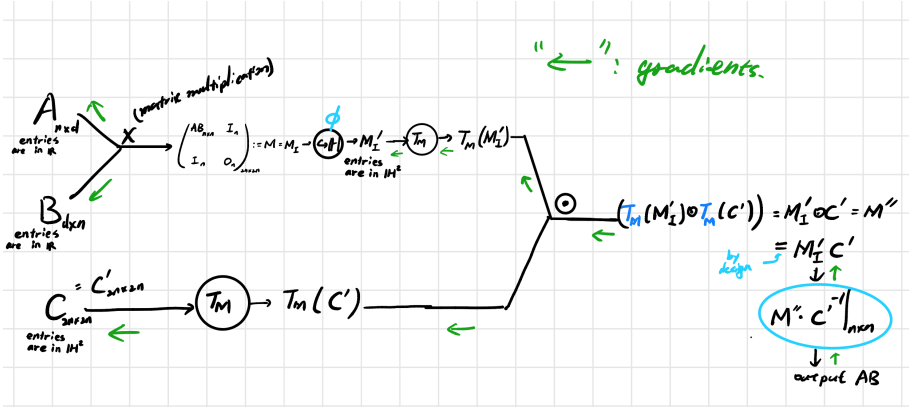



Figure 4: The computational graph of Case 2 when both $AB$ and $C$ are not invertible.

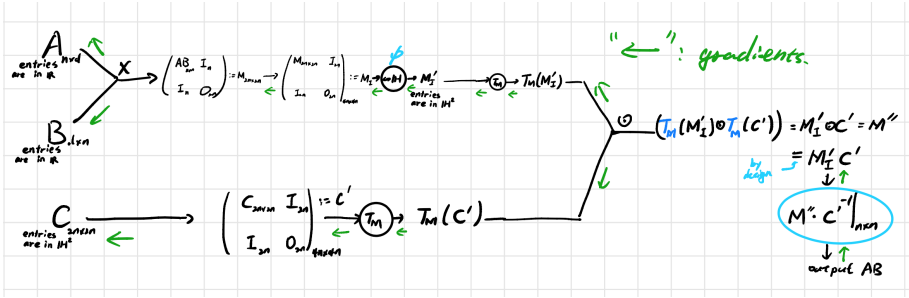
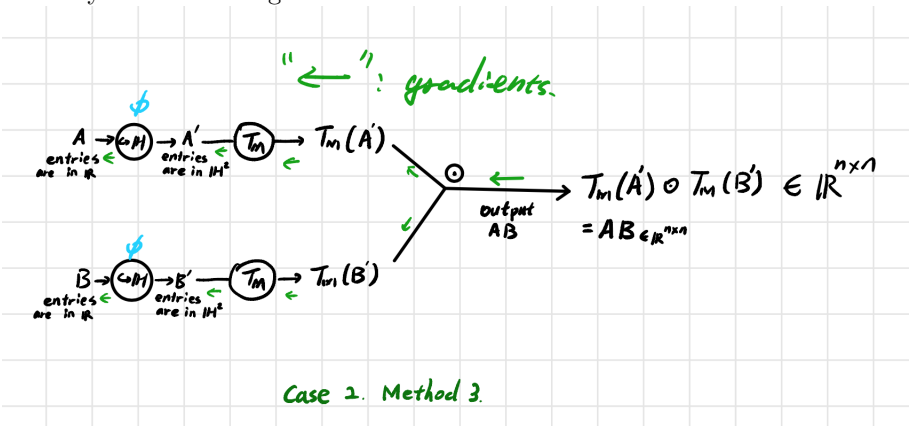


Figure 5: The computational graph of Case 2 when hyperbolic-length product is implemented to the model directly before training.




# 3 Examples

## 3.1 Self-attention mechanism in any transformer-based models with the hyperbolic-length product implemented before training

All layers of a given neural network model can be implemented with the above hyperbolic-orbit algorithm. The following is an example when the hyperbolic-orbit algorithm is applied to the self-attention mechanism, i.e. transformer-based models. Let $A = Q \in \mathbb{C}^{n\times d_q}$ and $B = K^t \in \mathbb{C}^{d_k\times n}$ in self-attention mechanism[9] using dynamic scaling factor $\beta$ (defined in our another work in progress), and by convention let $d_q = d_k$.

# 4 Discussion

In Case 1, since each step from step 1 to step 3 is invertible, after these operations, hyperbolic-length product can be applied to $M_I'$ and $C'$, then apply all the inverses and restrictions to back to $AB$, so the output is also unchanged, but the orbit of the neural network can be found using the above hyperbolic-orbit algorithm.

Furthermore, the isometry group of $\mathbb{H}^n$ is a group of Möbius mapping, hence its group elements preserve not only hyperbolic length but angles, thus the information of the sign of each $a_{ki}b_{il}$ could

be preserved.

With the hyperbolic-orbit algorithm, it is possible to find an appropriate Möbius mapping that is corresponding to $M \in \mathrm{PSL}(2,\mathbb{R})$ such that training a 3-node neural network become not necessarily NP-hard in [3]. Further, if every layer of a given neural network is implemented with the hyperbolic-orbit algorithm introduced in this paper, by combining with perturbations and parallel computing, the steepest descent path toward the global minimum could be designed. The algorithm introduced by the paper can help to save training time for parallel computing in the ensemble method, that is, instead of starting over from the beginning to train $N'$ models in parallel, $N' \in \mathbb{N}$, the training can start with $N'$ models that are equivalent to a trained model that already has an error that cannot be reduced by using the old training set–but on the orbits of the trained model, each equivalent model on the orbit can be kept training with a new training set by updating the weight matrices using back-propagation with the gradient descent algorithm. Then, in each epoch, the best model could be selected and instantaneously its $N'$ equivalent models could be obtained using the hyperbolic-orbit algorithm.

Furthermore, this algorithm could also be applied for non-linear encryption. For instance, the cardinality of the orbit of $A$ and $B$ is infinite, thus there are infinitely many ways to encrypt messages in A or $B$. Let message be embedded in $(T_1(A) \odot_{\mathbb{H}^n} T_1(B))^N$ by using $T_1$. Then the message can only be deciphered if the receiver knows the two correct isometries $T_3$ and $T_2$ and an integer $N$ as keys to decode two encrypted matrices: $T_2(B)$ and $T_3(A)$ to reconstruct the message in $(T_1(A) \odot_{\mathbb{H}^n} T_1(B))^N$ using $(T_1(A) \odot_{\mathbb{H}^n} T_1(B))^N = (A \odot_{\mathbb{H}^n} B)^N$. The meaning of the matrix $B$ is not only a lock, but it can obfuscate the frequency of characters used in the message in a random method.

## 5 Limitation

The hyperbolic-orbit algorithm has a limitation when it is applied to case 2 if the model was trained. To show the limitation, consider an example where there are two points $a_1$, and $a_2$ from $A$, and two points $b_1$ and $b_2$ in $B$. In $\mathbb{H}^n, n \geq 3$, suppose the hyperbolic length between $b_1$ to $a_1$ and $b_2$ to $a_1$ are both zero, and the hyperbolic length between $b_1$ and $a_2$ is $L$. But, for $b_2$, the

hyperbolic length between $b_2$ and $a_2$ is $2L$. If this happens, it might be possible to find an auxiliary invertible matrix $C$ and instead of applying hyperbolic-orbit algorithm on $A$ times $B$, applying it to $AB$ times $C$ and multiply the final result with $C$ inverse such that the matrix product $ABC$ is positive and does not have any case like the one mentioned in this section. This is the starting point of designing case 1 in the algorithm

# References


[1] Lars V. Ahlfors, *Möbius transformations in several dimensions*, Ordway Professorship Lectures in Mathematics, University of Minnesota, School of Mathematics, Minneapolis, Minn., 1981. MR 725161

[2] Alan F. Beardon, *The geometry of discrete groups*, Graduate Texts in Mathematics, vol. 91, Springer-Verlag, New York, 1995, Corrected reprint of the 1983 original. MR 1393195

[3] Avrim Blum and Ronald Rivest, *Training a 3-node neural network is np-complete*, Advances in neural information processing systems **1** (1988).

[4] J Stephen Judd, *Neural network design and the complexity of learning*, MIT press, 1990.

[5] M. È. Kapovich and L. D. Potyagaĭlo, *On the absence of finiteness theorems of Ahlfors and Sullivan for Kleinian groups in higher dimensions*, Sibirsk. Mat. Zh. **32** (1991), no. 2, 61–73, 212. MR 1138441

[6] Michael Kapovich, *Kleinian groups in higher dimensions*, Geometry and dynamics of groups and spaces, Progr. Math., vol. 265, Birkhäuser, Basel, 2008, pp. 487–564. MR 2402415

[7] John G. Ratcliffe, *Foundations of hyperbolic manifolds*, Graduate Texts in Mathematics, vol. 149, Springer, Cham, [2019] ©2019, Third edition [of 1299730]. MR 4221225

[8] José Seade and Alberto Verjovsky, *Higher dimensional complex Kleinian groups*, Math. Ann. **322** (2002), no. 2, 279–300. MR 1893917

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, *Attention is all you need*, Advances in neural information processing systems **30** (2017).

[10] Chulhee Yun, Suvrit Sra, and Ali Jadbabaie, *Global optimality conditions for deep neural networks*, arXiv preprint arXiv:1707.02444 (2017).