Training from the orbits of neural networks

William Chuang

October 21, 2023

1 Introduction

This is no time for false equivalence: Today, neural networks have a wide range of applications. From unmanned aerial vehicle (UAV) to large language models (LLMs) which could have trillions of parameters. Consider an adversarial attack on the UAV, or a bug found in an LLM, e.g. it may give a false output to a certain set of inputs, is there a way to instantly recover the system without changing any of its functions? A possible solution is 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.

Furthermore, inspired by [14, 3, 7], 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? In a nutshell, is it feasible to increase a well-trained model's performance?

Defining the following notions to describe the questions more precisely:

Definition 1. *A neural network model (function) is converged if it reaches one of local minima in its moduli space (i.e. loss surface).*

Definition 2. *Two converged neural network models (functions) are equivalent if they both give identical outputs on a fixed training set as the input.*

In other words, if the given model is denoted by f_{model} which was trained by using samples in the training set X, since it is a function, consider a set of all (f_{model}, O), where O is a neighborhood that contains X, then its equivalent model can be defined:

Definition 3. *Two converged neural network models (functions) f and g trained within O_1 and O_2 are denoted by pairs (f, O_1) and (g, O_2). Furthermore, (f, O_1) and (g, O_2) are equivalent, if there exists an open set $O \subset O_1 \cap O_2$ containing x such that $f(x) = g(x)$ when f and g are restricted to the open set O.*

Proposition 1. *[12] The set of equivalent functions of f is an equivalent class.*

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: **For any given neural network model**, 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, and the generated equivalent models can give different output compared the given model when the input is out-of-sample**?

In other words, let f be the given converged model that is trained using samples in the set X. Then, the goal is to find an equivalent converged F in the equivalent class of (f, O_1) such that

(i) $F(x) = f(x), \forall x \in X$,

(ii) there exists a set X' that are not measure zero and $X' \neq X$ such that

$$\mathbb{E}\left[F(X')\right] \neq \mathbb{E}\left[f(X')\right] \text{ and,}$$

(iii) The weights of the converged $F(x)$ are not an affine images of the weights of the converged $f(x)$ so that $F(x)$ can have non-trivial gradients while running gradient descent and back-propagation algorithm on it to reach a new local or global minimum.

With the condition (ii), F cannot be a trivial composition of functions, e.g. $g^{-1} \circ g \circ f$, and with the condition (iii), the weights of the converged F cannot be an affine transformation of the converged f.

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 after a short period of time of training using the same (shuffled) training set without the new training samples such as the input generated by adversarial models, then chances are, **within the set of equivalent converged models, there might exist a converged model** that will not take the fake input or falsely classified a corner case.

2 Hyperbolic-length Product

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 4 (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 5. *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 2. *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 Ψ 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 3. *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[8, 1, 2, 9, 10, 11], for $n \geq 3$. Let A, B be embedded into $\mathbb{H}^n, n \geq 3$.

Definition 6 (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 4. *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.

Let a converged model W and a training set X be given. The goal is to find its converged equivalences.

Definition 7. *Hyperbolic-orbit Algorithm is an algorithm that implements hyperbolic-length products to replace matrix products.*

The following section introduces the hyperbolic-orbit algorithm.

3 Hyperbolic-orbit Algorithm

Let A and B be **any** two matrices in **any** converged neural network model f that was implemented with a matrix multiplication between A and B. **It is prefer to let at least one of A and B to be a weight matrix (tensor) of the model f. In general, A could be any weight matrix (tensor) in any given neural network f, and let B be the identity matrix (tensor).**

Case 1 Let f be **any** neural network model that was trained.

Step 1 Embed entries of AB to $\mathbb{H}^2$ by mapping each column vector of AB to a line parallel to y^2-axis without any overlapping on coordinates so that each pare circles does not have any overlap to each other. Denote the embedding function by ϕ.

Step 2 Constructing an invertible matrix C by taking points on a line that is parallel to y^2-axis without any overlapping in their coordinates and forming edges to its corresponding set of embedded entries from the column vector in AB–the correspondence is determined by matrix multiplication $(AB)C$. (See the following figure for an example when $\phi(AB)$ and C are 4 by 4.)

Step 3 Measuring all the hyperbolic length of edges in each $K_{n,d}$-graph (composed by vertices on the hyperbolic circle and the line that passes through its center), and sum all the lengths in each $K_{n,d}$-graph to make an entry of matrix M.

Step 4 Pick an element $g \in \mathrm{PSL}(2,\mathbb{R})$, derive a linear fractional mapping from g, and denote it by $T_g(z) = \frac{az+b}{cz+d}, z \in \mathbb{H}^2$.

Step 5 Apply T_g to every entry z of $\phi(AB)$ and C to map every entry (could be trillions) of both matrices to their images $T_g(z)$ to derive the new weights of the given model, i.e. to derive an equivalent model.

Step 6 Apply gradient descent back-propagation algorithm on the whole model to reach a new local or global minimum with the same in-sample (but could be shuffled) training set.

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. Denote the embedding images by $\phi(A)$ and $\phi(B)$.

Step 2 The dimension of hyperbolic space could start with two dimensions to optimize the memory usage.

Step 3 Pick an element $g \in \mathrm{PSL}(2,\mathbb{R})$, derive a linear fractional mapping from g, and denote it by $T_g(z) = \frac{az+b}{cz+d}, z \in \mathbb{H}^2$.

Step 4 Apply T_g to every entry z of $\phi(A)$ and $\phi(B)$ to map every entry (could be trillions) of both matrices to their images $T_g(z)$ to derive the new weights of the given model, i.e. to derive an equivalent model.

Step 5 Apply gradient descent back-propagation algorithm on the whole model to reach a new local or global minimum with the same in-sample (but could be shuffled)

training set.

Remark 1: The algorithm not only preserves the hyperbolic lengths, but the angle between its angle to the y^2 axis–this is the reason to use conformal mapping T_g. Hence, in Case 1 and Case 2, the hyperbolic-product could actually preserve the information of negative values, and one more layer of operation, i.e. subtraction, could be easily implement with the algorithm using the projection of the hyperbolic length to the y^2-axis. That is, it can have a zero, positive, or negative projection. Furthermore, the amount and sign of each of these projections (could be trillions) are preserved when $T_g \in \mathrm{PSL}(2,\mathbb{R})$ is applied to these (trillions) of weights, and the choice of T_g is infinitely many–images of all of the choices of T_g form the orbit of the model, and each image on the orbit gives an equivalent model.

Remark 1: Since it is possible that two entries in the matrix AB have same values. To design an invertible embedding, the following is an example of the embedding function $\phi : \mathbb{R} \to \mathbb{R} \times \mathbb{R} \times \mathbb{H}^{\not=}$:

$$\phi(x_{ij}) = (i, j, (ax + b) + ci)$$

where x_{ij} is the i-th row and j-th column from AB, and a, b, c are some real numbers such that $(ax + b) + ci \in \mathbb{H}^2$.

Figure 1: The computational graph of case 1.

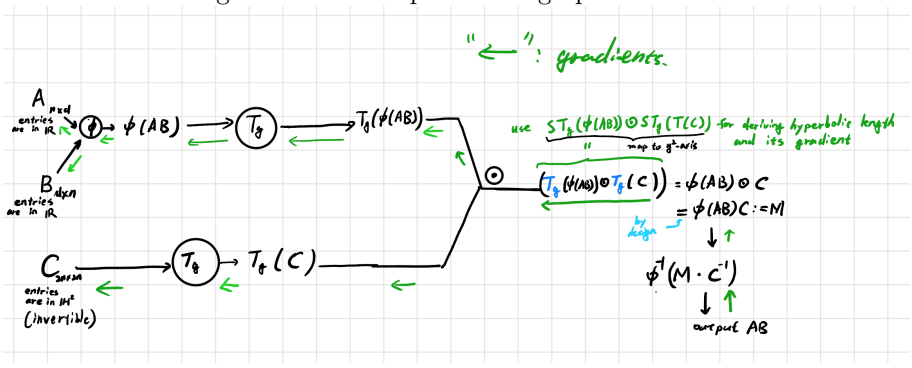


Figure 2: An illustration on how to embed **any** given product of matrices AB in **any neural network models** to hyperbolic space $\mathbb{H}^2$.

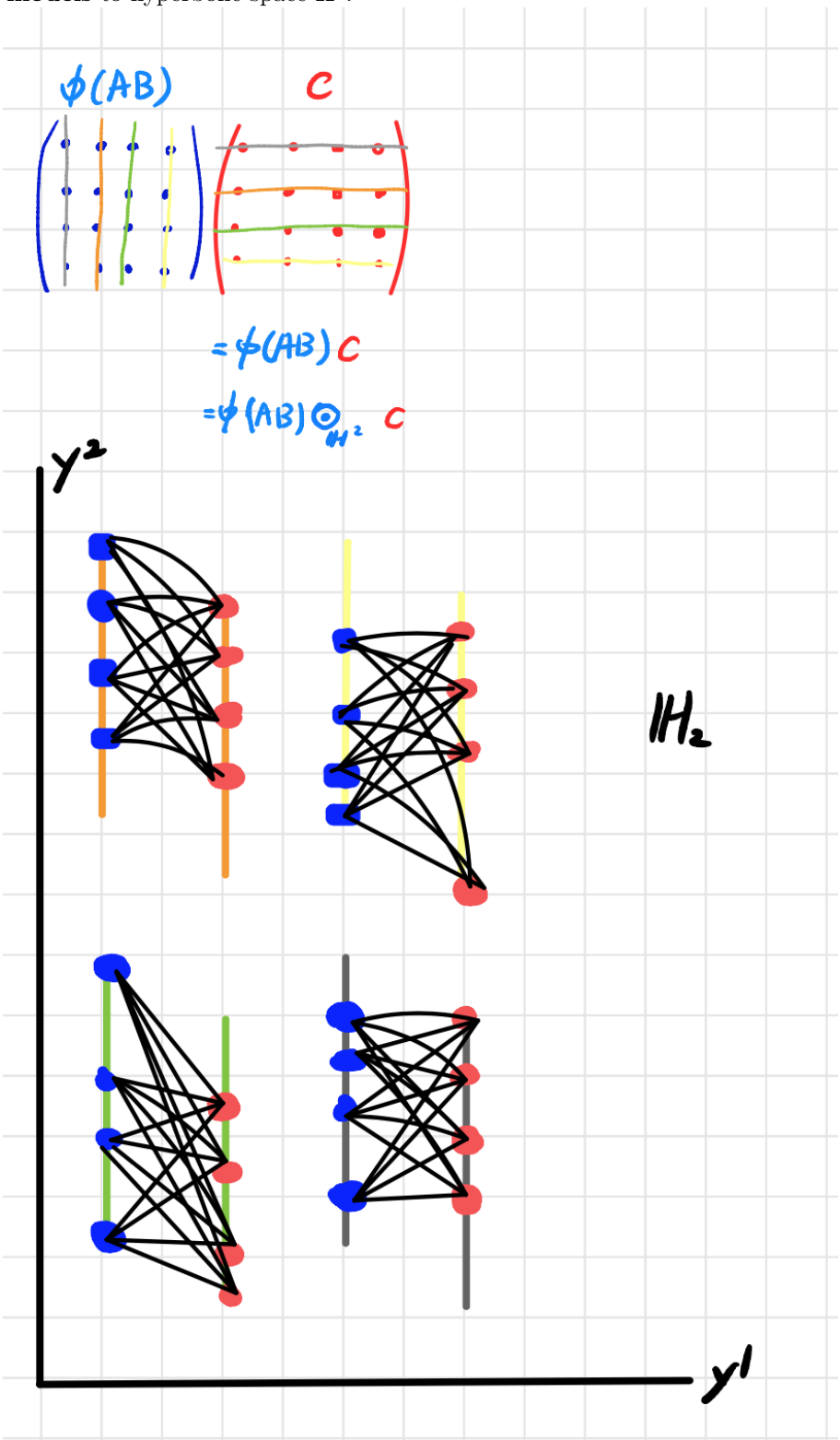


Figure 3: Gradients of the hyperbolic-length product.

$x \in T_m(M_I')$ $\quad x, y \in \mathbb{H}^2$ and ~~they~~ are entries of $T_m(M_I')$

$y \in T_m(C')$ $\quad T_m(C')$.

$$d_H(x,y) = d_H(S(x), S(y)) = \ln\left(\frac{S(x)}{S(y)}\right)$$

on y_2-axis

$$\frac{\partial d_H}{\partial y} = \frac{\partial}{\partial y} \ln\left(\frac{Sx}{Sy}\right)$$

$$= \frac{Sy\,(-Sx)\left(\frac{\partial S}{\partial y}\right)}{Sx\,(Sy)^2}$$

$$= \frac{-1}{Sy}\frac{\partial S}{\partial y} = \frac{-1}{Sy} \cdot \frac{1}{(cy+d)^2} = \boxed{\frac{-1}{(ay+b)(cy+d)}}$$

$$\frac{\partial d_H}{\partial x} = \frac{Sy}{Sx}\,\frac{\frac{\partial S}{\partial x} - 0}{(Sy)^2}$$

$$= \frac{1}{Sx\,Sy} \cdot \frac{\partial S}{\partial x} = \frac{1}{Sx\,Sy}\frac{1}{(cx+d)^2} = \frac{1}{\left(\frac{ax+b}{cx+d}\right)\left(\frac{ay+b}{cy+d}\right)(cx+d)^2}$$

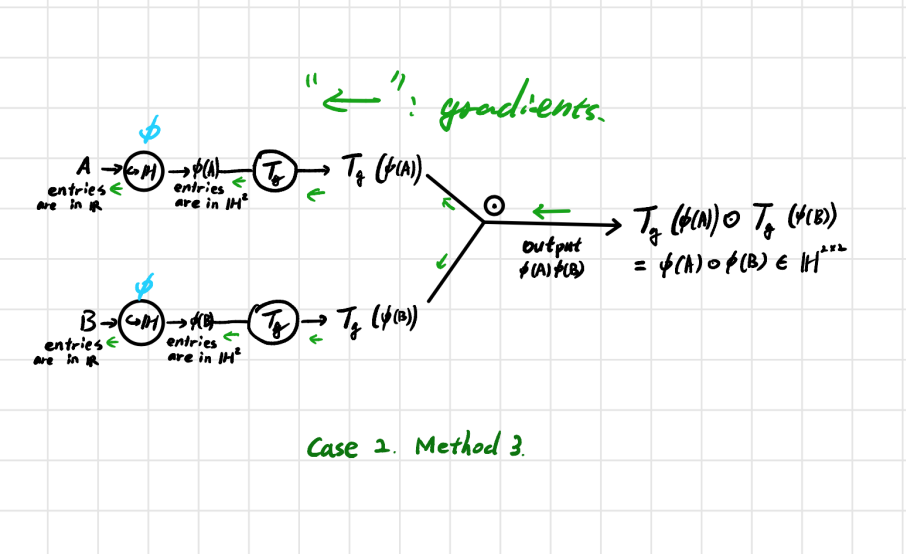
$$Sx = \frac{ax+b}{cx+d} \qquad \frac{\partial S}{\partial x} = \frac{a(cx+d) - c(ax+b)}{(cx+d)^2}$$

$$= \boxed{\frac{(cx+d)}{(ax+b)(ay+b)(cx+d)}}$$

$$Sy = \frac{ay+b}{cy+d} \qquad = \frac{ad-bc}{(cx+d)^2} = \frac{1}{(cx+d)^2}$$

$$= \frac{1}{(cy+d)^2}$$

Figure 4: The computational graph of Case 2 when a hyperbolic-length product is implemented to the model directly before training.



4 Examples

4.1 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[13] using dynamic scaling factor β (defined in our another work in progress), and by convention let $d_q = d_k$. Then, for each entry of the matrix QK^T, its denominator is a finite Poincaré series[5], and its dynamic scaling factor (work in progress) is corresponding to the entropy of the system, e.g. a system could be words in an input string.

5 Empirical Results

Work in progress. The off-line code could be finished after the quals.

6 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 converged 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' converged models that are equivalent to the given converged model with the same in-sample (but could be shuffled) training set and are converged to different minima in the moduli space.

Since the moduli space of any given converged neural network is determined by the range of images of $T_g \in \mathrm{PSL}(2,\mathbb{R})$, i.e. its orbits. By choosing a different set of isometries to form T_g could determine a different fundamental group and a corresponding hyperbolic surface M. The set of equivalence class of the metrics on the the surface M could be investigated using its corresponding the moduli space of hyperbolic surfaces (which is also known as the Teichmüller space). There is a unique Kähler-Einstein metric [4], and the geometry and topology of loss surface of any given converged neural network are determined by the weights of that neural net, the geometry and topology of the family loss surfaces could be investigated using recent results of Kähler-Einstein metric.

The range of $T_g \in \mathrm{PSL}(2,\mathbb{R})$ determines the range of orbits of a given converged neural network. The number of possible ranges are determined by the hyperbolic surface that is determined by the group generated by all T_g that were used to map the orbits. The number of these hyperbolic surfaces could also be enumerated using the method introduced in[6] which gives not only a systematic way to enumerate these hyperbolic surfaces, but also can be used to enumerate the number of ways to transform any given neural net on its orbit by machines in a systematically.

Since each T_g is a continuous open mapping, there exists an open set O to cover all of images of the weight tensor of a given model f. By studying the topology and geometry of the moduli space that this open set O can move over, the topology and geometry of loss surface could be investigated. In other words, this paper also demonstrates a correspondence between the moduli space of T_g, i.e. the Teichmüller space, and the loss surface (landscape) of any given model.

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 a large 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. Since T_2 and T_3 are linear fractional, i.e. they have the form $\frac{az+b}{cz+d}$, thus the keys only need eight real numbers plus a large integer.

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] Shiu-Yuen Cheng and Shing-Tung Yau, *On the existence of a complete kähler metric on non-compact complex manifolds and the regularity of fefferman's equation*, Communications on Pure and Applied Mathematics **33** (1980), no. 4, 507–544.

[5] William Huanshan Chuang, *The hausdorff dimension of limit sets of well-distributed schottky groups*, Ph.D. thesis, San Francisco State University, 2022.

[6] ———, *An invariant between hyperbolic surfaces and lattice spin models*, arXiv preprint arXiv:1511.02291 (2023).

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

[8] 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

[9] 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

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

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

[12] Loring W Tu, *Manifolds*, An Introduction to Manifolds, Springer, 2011.

[13] 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).

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