Training a robust model from the orbits of neural networks

William Chuang

November 3, 2023

1 Introduction

This is no time for false equivalence: Today, neural networks have a wide range of applications. From unmanned aerial vehicles (UAV) to large language models (LLMs) which could have trillions of parameters. Consider an adversarial attack[32, 49, 2] 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 robust 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 [52, 6, 24], consider the following question: can the neural network be trained to reach a global minimum on its loss surface[29, 18, 33, 14, 17, 15, 19, 22, 13, 27, 36, 38, 10, 37, 28, 41, 53, 31, 8]? In a nutshell, is it possible 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 or landscape).*

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. *[50] The set of equivalent functions of f is an equivalent class.*

To use random matrix theory[30, 40, 44] 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 ϵ be a small positive number ($\epsilon < 1$), and 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) the probability of the output of F is different to the output of f when the input is from X is less than ϵ,

(ii) the probability of the output of F is different to the output of f when the input is from the complement of X is greater than $1 - \epsilon$, and

(iii) the weights of the converged $F(x)$ are not an affine transformations of the weights of the converged $f(x)$

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 so that $F(x)$ can have non-trivial gradients while running gradient descent and back-propagation algorithm on it using a new training set to reach a new local or global minimum.

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 to make the model become robust.

If a model can be replaced by another equivalent model, then the other model might be at the different point on the loss surface[29, 18, 33, 14, 17, 15], and 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 converged 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 if it is a robust solution due to the process to reach the solution is random.

Thus, on the contrary, if its equivalent converged models can be found immediately after a short period of time of training using using a new training set 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.

Applying symmetry groups in machine learning, particularly in training neural networks, is not a new idea. For instance, the idea of applying gauge groups on input data[8]. However, to the best of the author's knowledge, applying group action on the neural network function itself has not appeared in the literature. **An geometric analogy that could help to better build an intuition and understanding of the work introduced in this paper is to consider the**

comparison of two ideas in linear algebra: Gauss elimination and invariant subspace. In Gauss elimination, the coordinate system, i.e. the observer of the input geometric data is fixed, but the input was transformed (using elementary row operations) so that the solutions (the intersections of hyperplanes) could be observed using the fixed coordinate system. In the contrary, in finding the basis of invariant subspace, i.e. the latent vectors or eigenvectors, the procedure is equivalent to rotate the coordinate system to describe the input data–in this case, the input is fixed, but the bases are transformed which is parallel to the idea introduced in this paper as a comparison to the idea of transforming the input data[8]. In this paper, neural network functions have a similar role of bases in invariant subspace and the neural nets were transformed using group actions to their orbits. Furthermore, the method introduced in this paper can be extended so that the input data can also be transformed using group action (by embedding the input to the hyperbolic space using the algorithm introduced in the following sections, then the input can be transformed to its orbits).

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 $\mathbb{H}^2$[3]:

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

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 $M \in \mathrm{PSL}(2, \mathbb{R}) \setminus \{\mathrm{id}\}$, then there exists an isomorphism to map M to a linear fractional (Möbius) map $T_M(z) = \frac{az+b}{cz+d}, a, b, c, d \in \mathbb{R}, z \in \mathbb{C}$. It can be shown that this fractional linear transformation is isomorphic to a (2+1)-dimensional Lorentz boost or (2+1)-dimensional Lorentz transformation in physics.

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. $\square$

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 $z_1, z_2 \in \mathbb{H}^2$, $d_{\mathbb{H}^2}(z_1, z_2) = \rho_{\mathbb{B}^2}(\Psi(z_1), \Psi(z_2))$. $\square$

The 2-dimensional hyperbolic-length product can be generalized to n-dimension[25, 1, 3, 26, 42, 43], 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:*

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

It can be showed that the isometry group of $\mathbb{H}^n$ is the Lorentz group $SO(n, 1; \mathbb{R})$, so the transformation T_M is isomorphic to an (n+1)-dimension Lorentz transformation in physics.

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 a new 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 a new 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 2: Since it is possible that two entries in the matrix AB have the same values. To design an invertible embedding, the following is an example of the embedding function $\phi : \mathbb{R} \to \mathbb{R} \times \mathbb{R} \times \mathbb{H}^{\nvDash}$:

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

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

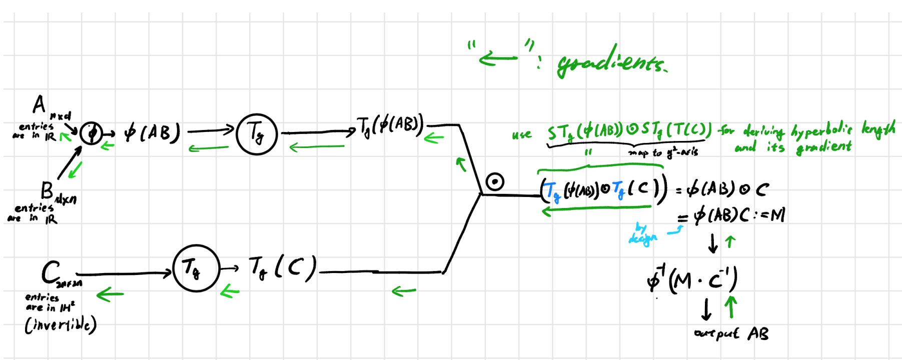

Figure 1: The computational graph of case 1.

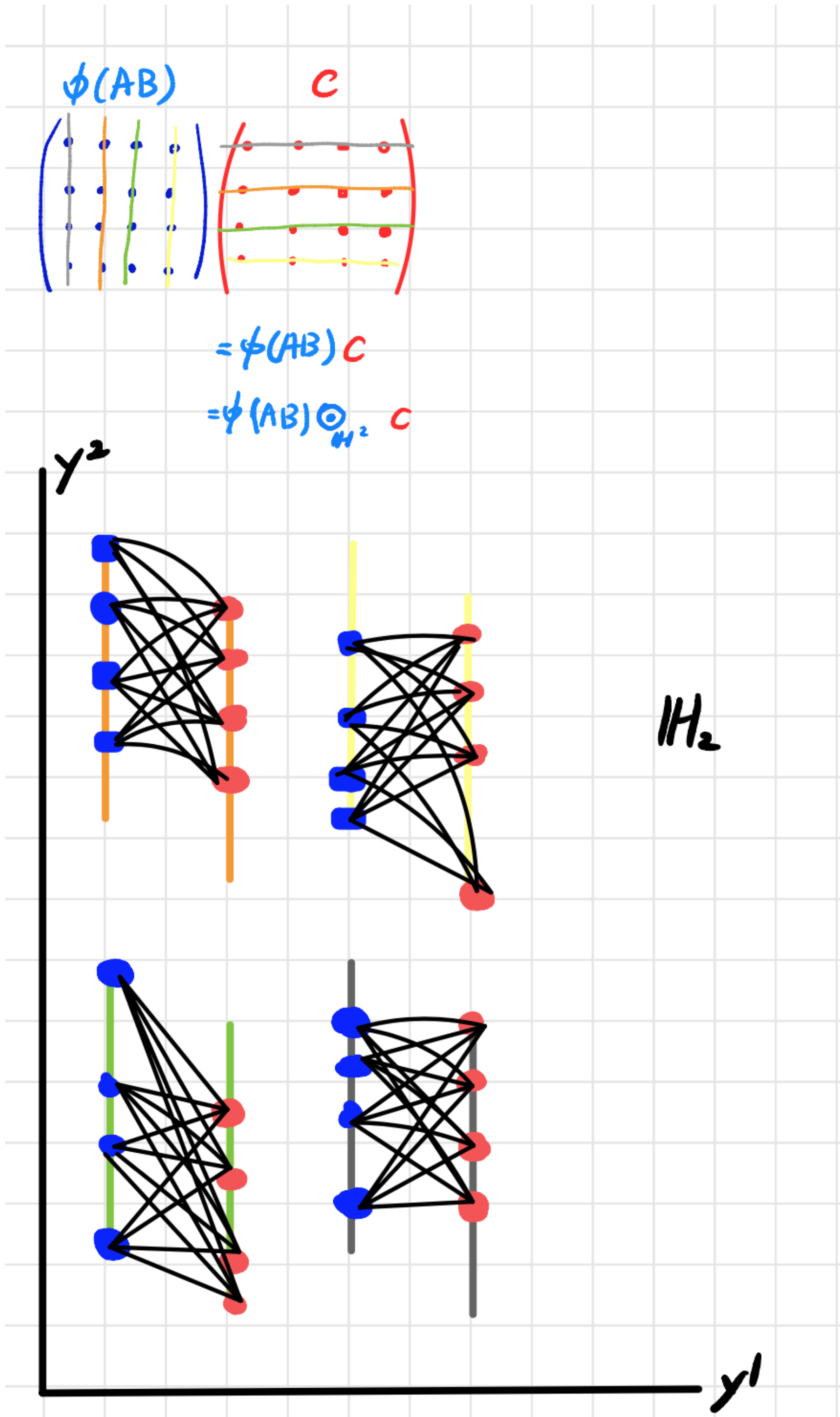

Figure 2: An illustration on how to embed **any** given product of matrices AB in **any neural network models** to hyperbolic space $\mathbb{H}^2$.

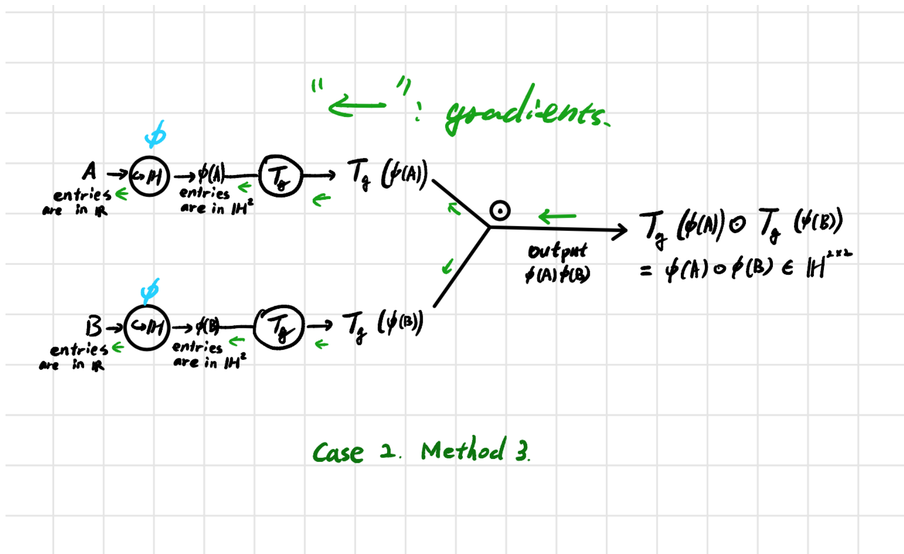

Figure 3: The computational graph of Case 2 when a hyperbolic-length product is implemented to the model directly before training.

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[51] 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[11], 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

It is crucial to know that when applying back-propagation, the gradients of the hyperbolic product, i.e. the derivative of a hyperbolic length is not derived by taking the derivative by using general formulas of hyperbolic length. Instead, the hyperbolic curve (a geodesic) is mapped to the y^2-axis first, then taking a derivative on the transformed function that describes this vertical line (which is also a geodesic). In this way, the model can converge with a stable gradient (that is not either a constant function, an identity function, zero function, or diverging to infinity) derived from a composition of fraction linear functions. For instance, assume there are two weights (e.g. entries of $T_g(C)$ and $T_g(\phi(AB))$) that are represented as two distinct points in the hyperbolic space $x, y \in \mathbb{H}^2$, and x, y are not on the positive y^2-axis. Then the hyperbolic length between x and y is the length of a geodesic that connects x and y measured by hyperbolic metric.

Let S be a fractional linear map that is also constructed by an element in the isometry group $\mathrm{PSL}(2, \mathbb{R})$, i.e. S is not an arbitrary fractional linear mapping that maps x, y to the positive y^2-axis, it must satisfy the condition that its determinant of four coefficients is equal to one. Then

$$d_{\mathbb{H}^2}(x, y) = d_{\mathbb{H}^2}(Sx, Sy) = \ln\left(\frac{Sx}{Sy}\right),$$

where $Sx = \frac{ax+b}{cx+d}$ and $Sy = \frac{ay+b}{cy+d}$ are on the positive y^2-axis.

Then gradients of the hyperbolic products or the gradients provided by the hyperbolic-orbit algorithm are the following:

$$\frac{\partial d_{\mathbb{H}^2}}{\partial y} = \frac{\partial}{\partial y} \ln \frac{Sx}{Sy} = \frac{-1}{(ay+b)(cy+d)},$$

$$\frac{\partial d_{\mathbb{H}^2}}{\partial y} = \frac{\partial}{\partial x} \ln \frac{Sx}{Sy} = \frac{cx+d}{(ax+b)(ay+b)(cx+d)}.$$

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 [6]. 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 are converged to different minima in the moduli space using a new training set.

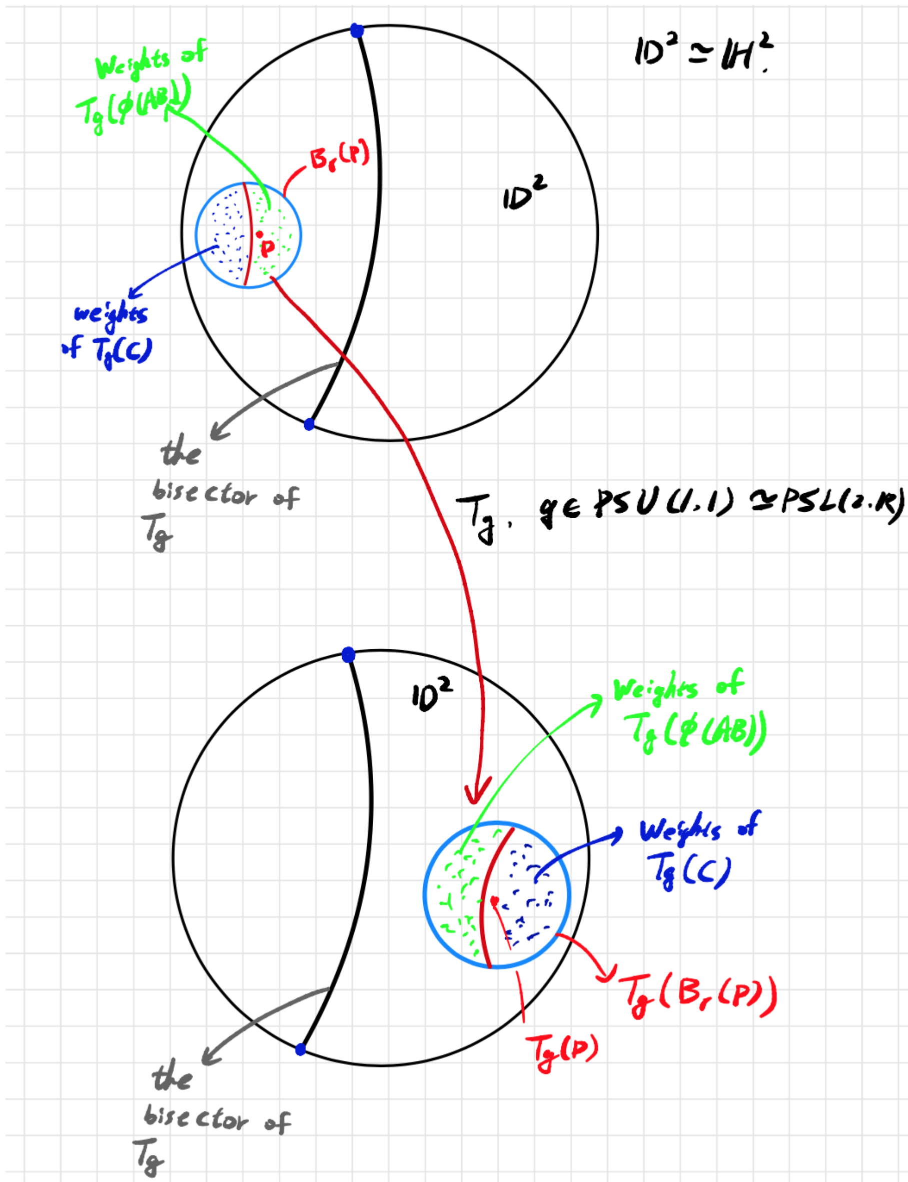

Figure 4: Using an open set $B_r(p)$ centered at $p \in \mathbb{H}^2$ with radius $r > 0$ that covers all embedded weights of the given converged model to study the correspondence between the location of $B_r(p)$ in $\mathbb{H}^2$ and the location of the model in the loss surface.

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 classes of the metrics on the surface M could be investigated using its corresponding moduli space of hyperbolic surfaces (which is also known as the Teichmüller space[45, 5]). There is a unique Kähler-Einstein metric [9, 21], 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.

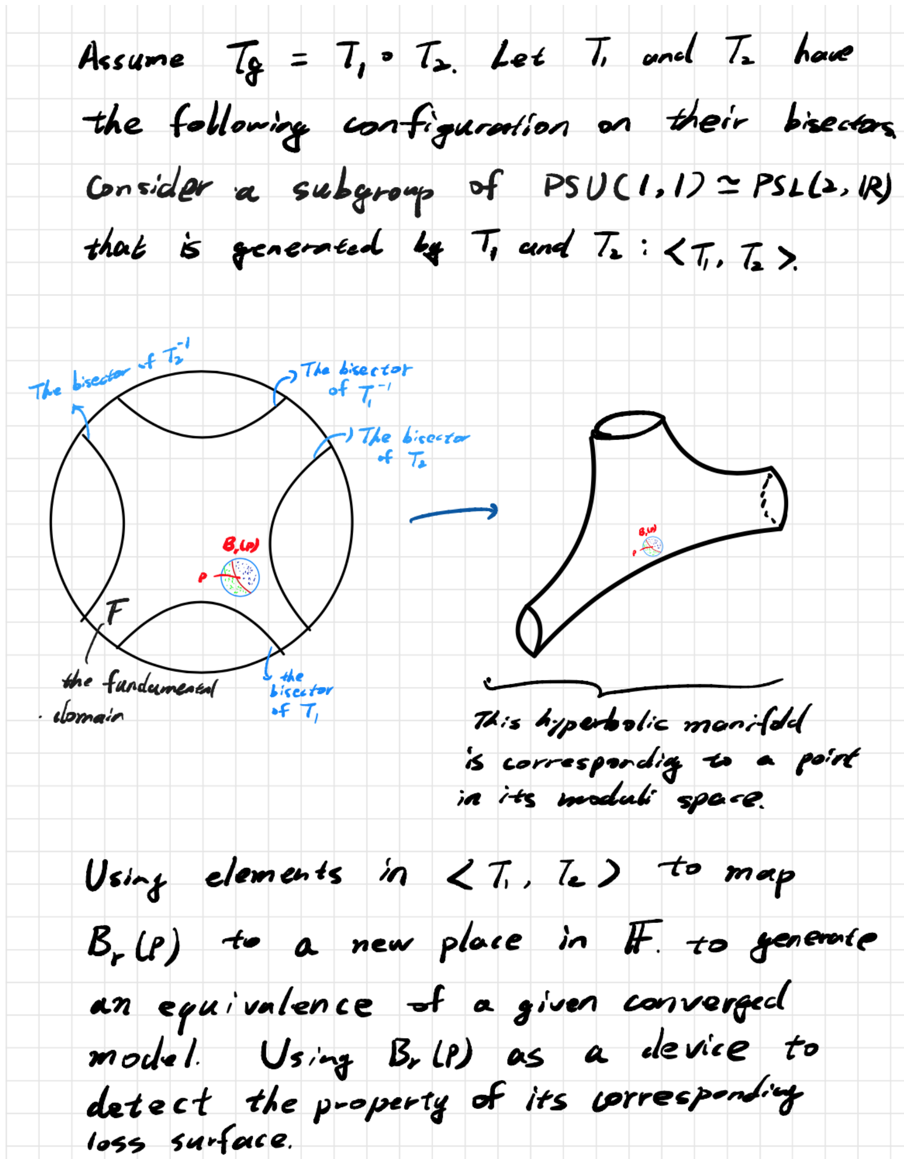

Figure 5: Focusing on a two-generator subgroup of $\mathrm{PSL}(2, \mathbb{R})$ (it can be called a Schottky group, Fuchsian group, or Kleinian group) and its fundamental domain in $\mathbb{H}^2$. **The geometry, topology, and the group generators of the hyperbolic manifold[48, 45, 20, 47, 34, 4] determine the gradients that are provided by the hyperbolic-orbit algorithm.**

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[12] 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.

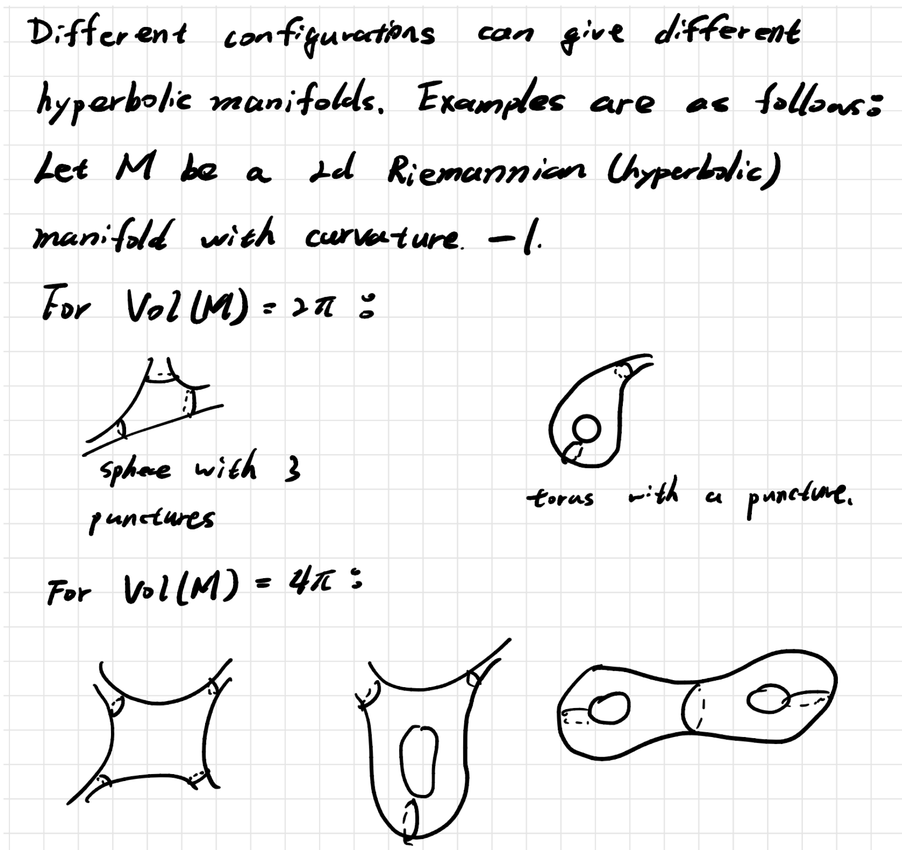

Figure 6: Different configurations can provide different oriented Riemannian manifolds.

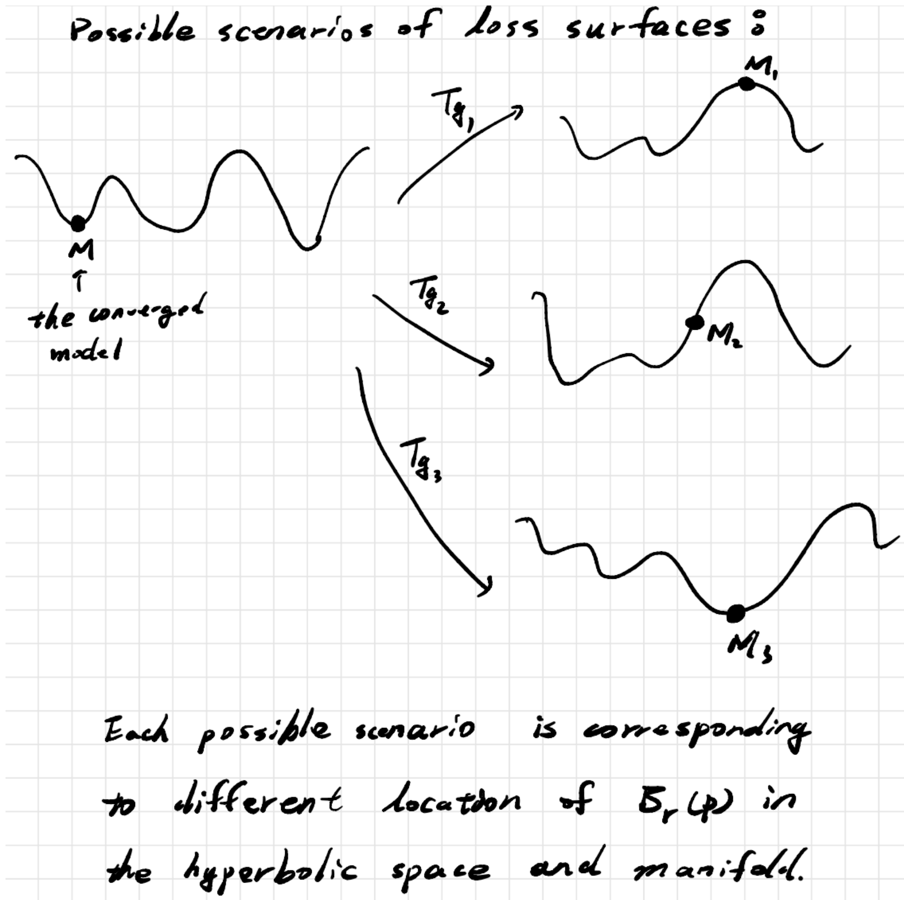

Figure 7: Possible scenarios of loss surface. Each possible scenario of the new loss surface after applying fractional linear mapping (which is corresponding to an element to the two-generator group) to the weights of the given converged model. Notice that it is possible that the model was mapped to the deleted neighborhood of a new local minimum or the global minimum.

Since each T_g is a continuous open mapping, there exists an open set O to cover all 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 the loss surface could be investigated. In other words, this paper also demonstrates a correspondence between the moduli space of T_g[45, 46], and the loss surface (landscape)[29, 18, 33, 14, 17, 15] of any given model.

There is an intuitive understanding from the physics of classical gravity: Since each fractional linear mapping T_g can be parameterized[45, 16], the gradients from back-propagation can also be set to modify the parameters of T_g. If it is without letting gradients to go through the parameter set of T_g, then the group action of neural networks could be considered a pure transformation on the (spacetime) Riemannian manifold without affecting the platform, i.e. the (spacetime) Riemannian manifold. However, if the gradients are allowed to go through the parameter set of T_g, then the the (spacetime) Riemannian manifold is affected (interacted), then in addition to use moduli space of Riemannian manifold, the back-propagation procedure could also have a control over the geometry and topology of the Riemannian manifold. In brief, if the fractional linear mapping T_g is allowed to interact with the platform of particles (embedded weights are the locations that label where particles are created or excited in spacetime and the platform is the Riemannian manifold), then it is a general relativity perspective[35]; if not, then it is a perspective from special relativity[35].

There is also a further intuitive understanding from the physics of quantum physics for the Riemannian manifold (algebraic curve[21]): it is possible to quantize the platform of particles and derive its eigenvalues, Laplacian, Selberg Zeta function, and Selberg trace formula[7, 39, 23]. A sketch of proof of Selberg trace formula and one of its nice applications can be found in Jacquet and Langlands' *Automorphic Forms on GL(2)*[23].

Please mind that the above two intuitive understandings do not imply whether the Riemannian manifold (i.e. the hyperbolic surface, algebraic curve) has any connection to the spacetime of our universe or not. The geometry of the spacetime is similar to the latent space of a usual neural network. Furthermore, it is similar to the development of the theory of differential equations: the original use of differential equations is for describing the motion of a moving particle in kinetics. However, this does not mean that differential equations can only be used to describe motions of moving particles. Likewise, the math of gravity could not only be used to describe gravity, but also possibly many other phenomena that share the same mathematical structure or could be described using the same framework.

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.

Compared to starting over to train a new model, the advantage of this hyperbolic-orbit algorithm is that it can preserve most of the learned features in the model while reaching a new local or global minimum. Since the loss surface is changed by the hyperbolic-orbit algorithm (since the weights of the model were mapped to new images in the hyperbolic space by a fractional linear mapping), the old local or global minimum might be mapped to a hill or halfway up a mountain.

Without using the hyperbolic-orbit algorithm, to train a converged model, it might take a long time with a large data set to move the model out of a local or global minimum and then move it to another minimum that can take care of not only the old training set but also the new training set. Further, a usual technique is to slice the model[54] which could be costly when the model is large (e.g. with trillions of parameters).

Using the hyperbolic-orbit algorithm, it is very likely that after using a small set of new data, the model can converge to a new minimum without losing features that it already learned, i.e. the model could give the same output if the input is from the old training set. Furthermore, **it is possible that the model was mapped to a deleted neighborhood of a new local minimum or the global minimum**. If it is mapped to any place that is not a local minimum, then when a new training is applied, the model would start to converge to a local minimum that has the steepest gradient near the starting point which means the weights of the new model would be modified immediately.

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] Anish Athalye, Nicholas Carlini, and David Wagner, *Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples*, International conference on machine

learning, PMLR, 2018, pp. 274–283.

[3] 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

[4] Riccardo Benedetti and Francesco Bonsante, *(2+1)-Einstein spacetimes of finite type*, arXiv preprint arXiv:0704.2152 (2007).

[5] Lipman Bers, *Finite dimensional teichmüller spaces and generalizations*, Bulletin of the American Mathematical Society **5** (1981), no. 2, 131–172.

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

[7] David Borthwick, *Spectral theory of infinite-area hyperbolic surfaces*, second ed., Progress in Mathematics, vol. 318, Birkhäuser/Springer, [Cham], 2016. MR 3497464

[8] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković, *Geometric deep learning: Grids, groups, graphs, geodesics, and gauges*, arXiv preprint arXiv:2104.13478 (2021).

[9] 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.

[10] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun, *The loss surfaces of multilayer networks*, Artificial intelligence and statistics, PMLR, 2015, pp. 192–204.

[11] William Huanshan Chuang, *The hausdorff dimension of limit sets of well-distributed schottky groups*, 2022.

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

[13] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio, *Sharp minima can generalize for deep nets*, International Conference on Machine Learning, PMLR, 2017, pp. 1019–1028.

[14] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht, *Essentially no barriers in neural network energy landscape*, International conference on machine learning, PMLR, 2018, pp. 1309–1318.

[15] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai, *Gradient descent finds global minima of deep neural networks*, International conference on machine learning, PMLR, 2019, pp. 1675–1685.

[16] Lester R Ford, *Automorphic functions*, vol. 85, American Mathematical Soc., 2004.

[17] C Daniel Freeman and Joan Bruna, *Topology and geometry of half-rectified network optimization*, arXiv preprint arXiv:1611.01540 (2016).

[18] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson, *Loss surfaces, mode connectivity, and fast ensembling of dnns*, Advances in neural information processing systems **31** (2018).

[19] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe, *Qualitatively characterizing neural network optimization problems*, arXiv preprint arXiv:1412.6544 (2014).

[20] Mikhael Gromov, *Hyperbolic manifolds, groups and actions*, Riemann surfaces and related topics: Proceedings of the 1978 Stony Brook Conference (State Univ. New York, Stony Brook, NY, 1978), vol. 97, 1981, pp. 183–213.

[21] Kentaro Hori, Sheldon Katz, Albrecht Klemm, Rahul Pandharipande, Richard Thomas, Cumrun Vafa, Ravi Vakil, and Eric Zaslow, *Mirror symmetry*, vol. 1, American Mathematical Soc., 2003.

[22] Daniel Jiwoong Im, Michael Tao, and Kristin Branson, *An empirical analysis of deep network loss surfaces*, (2016).

[23] Hervé Jacquet and Robert P Langlands, *Automorphic forms on gl (2): Part 1*, vol. 114, Springer, 2006.

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

[25] 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

[26] 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

[27] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio, *Generalization in deep learning*, arXiv preprint arXiv:1710.05468 **1** (2017), no. 8.

[28] Pranav Kumar, Taniya Mandal, and Swapnamay Mondal, *Black holes and the loss landscape in machine learning*, arXiv preprint arXiv:2306.14817 (2023).

[29] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein, *Visualizing the loss landscape of neural nets*, Advances in neural information processing systems **31** (2018).

[30] Cosme Louart, Zhenyu Liao, and Romain Couillet, *A random matrix approach to neural networks*, The Annals of Applied Probability **28** (2018), no. 2, 1190–1248.

[31] James Lucas, *Optimization and loss landscape geometry of deep learning*, Ph.D. thesis, University of Toronto (Canada), 2022.

[32] Shiqing Ma, Yingqi Liu, Wen-Chuan Lee, Xiangyu Zhang, and Ananth Grama, *Mode: automated neural network model debugging via state differential analysis and input selection*, Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2018, pp. 175–186.

[33] Katherine Mary Malan, *A survey of advances in landscape analysis for optimisation*, Algorithms **14** (2021), no. 2, 40.

[34] Curt McMullen, *Riemann surfaces and the geometrization of 3-manifolds*, Bulletin of the American mathematical society **27** (1992), no. 2, 207–216.

[35] Charles W Misner, Kip S Thorne, and John Archibald Wheeler, *Gravitation*, Macmillan, 1973.

[36] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro, *Exploring generalization in deep learning*, Advances in neural information processing systems **30** (2017).

[37] Quynh Nguyen and Matthias Hein, *The loss surface of deep and wide neural networks*, International conference on machine learning, PMLR, 2017, pp. 2603–2612.

[38] Quynh Nguyen, Mahesh Chandra Mukkamala, and Matthias Hein, *On the loss landscape of a class of deep neural networks with no bad local valleys*, arXiv preprint arXiv:1809.10749 (2018).

[39] nLab authors, *Selberg zeta function*, `https://ncatlab.org/nlab/show/Selberg+zeta+function`, November 2023, Revision 14.

[40] Jeffrey Pennington and Yasaman Bahri, *Geometry of neural network loss surfaces via random matrix theory*, International conference on machine learning, PMLR, 2017, pp. 2798–2806.

[41] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli, *Exponential expressivity in deep neural networks through transient chaos*, Advances in neural information processing systems **29** (2016).

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

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

[44] Mohamed El Amine Seddik, Cosme Louart, Mohamed Tamaazousti, and Romain Couillet, *Random matrix theory proves that deep learning representations of gan-data behave as gaussian mixtures*, International Conference on Machine Learning, PMLR, 2020, pp. 8573–8582.

[45] Mika Seppälä and Tuomas Sorvali, *Geometry of riemann surfaces and teichmüller spaces*, Elsevier, 2011.

[46] Dennis Sullivan, *Hyperbolic geometry and homeomorphisms*, Geometric topology, Elsevier, 1979, pp. 543–555.

[47] William P Thurston, *Three dimensional manifolds, kleinian groups and hyperbolic geometry*, (1982).

[48] ———, *The geometry and topology of three-manifolds: With a preface by steven p. kerckhoff*, vol. 27, American Mathematical Society, 2022.

[49] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel, *Ensemble adversarial training: Attacks and defenses*, arXiv preprint arXiv:1705.07204 (2017).

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

[51] 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).

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

[53] Yaoyu Zhang, Zhongwang Zhang, Tao Luo, and Zhiqin J Xu, *Embedding principle of loss landscape of deep neural networks*, Advances in Neural Information Processing Systems **34** (2021), 14848–14859.

[54] Yingyi Zhang, Zan Wang, Jiajun Jiang, Hanmo You, and Junjie Chen, *Toward improving the robustness of deep learning models via model transformation*, Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering, 2022, pp. 1–13.