Unveiling Transformer Perception by Exploring Input Manifolds

Anonymous Author(s)
Affiliation
Address
email

Abstract

This paper introduces a general method for the exploration of equivalence classes in the input space of Transformer models. The proposed approach is based on sound mathematical theory which describes the internal layers of a Transformer architecture as sequential deformations of the input manifold. Using eigendecomposition of the pullback of the distance metric defined on the output space through the Jacobian of the model, we are able to reconstruct equivalence classes in the input space and navigate across them. We illustrate how this method can be used as a powerful tool for investigating how a Transformer sees the input space, facilitating local and task-agnostic explainability in Computer Vision and Natural Language Processing tasks.

1 Introduction

In this paper, we propose a method for exploring the input space of Transformer models by identifying equivalence classes with respect to their predictions. We define an *equivalence class* of a Transformer model as the set of vectors in the embedding space whose outcomes under the Transformer process are the same. The study of the input manifold on which the inverse image of models lies provides insights for both explainability and sensitivity analyses. Existing methods aiming at the exploration of the input space of Deep Neural Networks and Transformers either rely on perturbations of input data using heuristic or gradient-based criteria [16, 22, 17, 14], or they analyze specific properties of the embedding space [5].

Our approach is based on sound mathematical theory which describes the internal layers of a Transformer architecture as sequential deformations of the input manifold. Using eigendecomposition of the pullback of the distance metric defined on the output space through the Jacobian of the model, we are able to reconstruct equivalence classes in the input space and navigate across them. In the XAI scenario, our framework can facilitate local and task-agnostic explainability methods applicable to Computer Vision (CV) and Natural Language Processing (NLP) tasks, among others.

In Section 2, we summarise the preliminaries of the mathematical foundations of our approach. In Section 3, we present our method for the exploration of equivalence classes in the input of the Transformer models. In Section 4, we perform a preliminary investigation of some applicability options of our method on textual and visual data. In Section 5, we discuss the relevant literature about embedding space exploration and feature importance. Finally, in Section 6, we give our concluding remarks[1].

[1]The code to reproduce our experiments can be found in the Supplementary Materials.

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

2 Preliminaries

In this Section, we provide the theoretical foundation of the proposed approach, namely the Geometric Deep Learning framework based on Riemannian Geometry [2].

A neural network is considered as a sequence of maps, the layers of the network, between manifolds, and the latter are the spaces where the input and the outputs of the layers belong to.

Definition 1 (Neural Network)**.** *A neural network is a sequence of $\mathcal{C}^1$ maps Λ_i between manifolds of the form:*

$$M_0 \xrightarrow{\Lambda_1} M_1 \xrightarrow{\Lambda_2} M_2 \xrightarrow{\Lambda_4} \cdots \xrightarrow{\Lambda_{n-1}} M_{n-1} \xrightarrow{\Lambda_n} M_n \tag{1}$$

We call M_0 the input manifold and M_n the output manifold. All the other manifolds of the sequence are called representation manifolds. The maps Λ_i are the layers of the neural network. We denote with $\mathcal{N}_{(i)} = \Lambda_n \circ \cdots \circ \Lambda_i : M_i \to M_n$ the mapping from the i-th representation layer to the output layer.

As an example, consider a shallow network with just one layer, the composition of a linear operator $A \cdot + b$ with a sigmoid function σ, where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$: then, the input manifold M_0 and the output manifold M_1 shall be $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively, and the map $\Lambda_1(\cdot) = \sigma(A \cdot + b)$. We generalize this observation into the following definition.

Definition 2 (Smooth layer)**.** *A map $\Lambda_i : M_{i-1} \to M_i$ is called a smooth layer if it is the restriction to M_{i-1} of a function $\overline{\Lambda}^{(i)}(x) : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ of the form*

$$\overline{\Lambda}^{(i)}_\alpha(x) = F^{(i)}_\alpha \left(\sum_\beta A^{(i)}_{\alpha\beta} x_\beta + b^{(i)}_\alpha \right) \tag{2}$$

for $i = 1, \cdots, n$, $x \in \mathbb{R}^{d_i}$, $b^{(i)} \in \mathbb{R}^{d_i}$ and $A^{(i)} \in \mathbb{R}^{d_i \times d_{i-1}}$, with $F^{(i)} : \mathbb{R}^{d_i} \to \mathbb{R}^{d_i}$ a diffeomorphism.

Remark 1. *Transformers implicitly apply for this framework, since their modules are smooth functions, such as fully connected layers, GeLU and sigmoid activations.*

Our aim is to transport the geometric information on the data lying in the output manifold to the input manifold: this allows us to obtain insight on how the network "sees" the input space, how it manipulates it for reaching its final conclusion. For fulfilling this objective, we need several tools from differential geometry. The first key ingredient is the notion of singular Riemannian metric, which has the intuitive meaning of a degenerate scalar product which changes point to point.

Definition 3 (Singular Riemannian metric)**.** *Let $M = \mathbb{R}^n$ or an open subset of $\mathbb{R}^n$. A singular Riemannian metric g over M is a map $g : M \to Bil(\mathbb{R}^n \times \mathbb{R}^n)$ that associates to each point p a positive semidefinite symmetric bilinear form $g_p : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ in a smooth way.*

Without loss of generality, we can assume the following hypotheses on the sequence (1): *i)* The manifolds M_i are open and path-connected sets of dimension $\dim M_i = d_i$. *ii)* The maps Λ_i are $\mathcal{C}^1$ submersions. *iii)* $\Lambda_i(M_{i-1}) = M_i$ for every $i = 1, \cdots, n$. *iv)* The manifold M_n is equipped with the structure of Riemannian manifold, with metric $g^{(n)}$. Definition 3 naturally leads to the definition of the pseudolength and of energy of a curve.

Definition 4 (Pseudolength and energy of a curve)**.** *Let $\gamma : [a, b] \to \mathbb{R}^n$ a curve defined on the interval $[a, b] \subset \mathbb{R}$ and $\|v\|_p = \sqrt{g_p(v, v)}$ the pseudo–norm induced by the pseudo–metric g_p at point p. Then the pseudolength of γ and its energy are defined as*

$$Pl(\gamma) = \int_a^b \|\dot\gamma(s)\|_{\gamma(s)} ds = \int_a^b \sqrt{g_{\gamma(s)}(\dot\gamma(s), \dot\gamma(s))} ds, \qquad E(\gamma) = \int_a^b \|\dot\gamma(s)\|^2_{\gamma(s)} ds \tag{3}$$

The notion of pseudolength leads naturally to define the distance between two points.

Definition 5 (Pseudodistance)**.** *Let $x, y \in M = \mathbb{R}^n$. The pseudodistance between x and y is then*

$$Pd(x, y) = \inf\{Pl(\gamma) \mid \gamma : [0, 1] \to M,\ \gamma \in \mathcal{C}^1([0, 1]),\ \gamma(0) = x,\ \gamma(1) = y\}. \tag{4}$$

One can observe that endowing the space $\mathbb{R}^n$ with a singular Riemannian metric leads to have non trivial curves whose length is zero. A straightforward consequence is that there are distinct points whose pseudodistance is therefore zero: a natural equivalence relation arises, *i.e.* $x \sim y \Leftrightarrow Pd(x, y) = 0$, obtaining thus a metric space $(\mathbb{R}^n/\sim, Pd)$.

The second crucial tool is the notion of pullback of a function. Let f be a function from $\mathbb{R}^p$ to $\mathbb{R}^q$, and fix the coordinate systems $x = (x_1, \dots, x_p)$ and $y = (y_1, \dots, y_q)$ on $\mathbb{R}^p$ and on $\mathbb{R}^q$, respectively. Moreover, we endow $\mathbb{R}^q$ with the standard Euclidean metric g, whose associated matrix is the identity. The space $\mathbb{R}^p$ can be equipped with the pullback metric f^*g whose representation matrix reads as

$$(f^*g)_{ij} = \sum_{h,k=1}^{q} \left(\frac{\partial f_h}{\partial x_i}\right) g_{hk} \left(\frac{\partial f_k}{\partial x_j}\right). \tag{5}$$

The sequence (1) shows that a neural network can be considered simply as a function, a composition of maps: hence, taking $f = \Lambda_n \circ \Lambda_{n-1} \circ \cdots \circ \Lambda_1$ and supposing that $M_0 = \mathbb{R}^p, M_n = \mathbb{R}^q$, the generalization of (5) applied to (1) provides with the pullback of a generic neural network.

Hereafter, we consider in (1) the case $M_n = \mathbb{R}^q$, equipped with the trivial metric $g^{(n)} = I_q$, *i.e.*, the identity. Each manifold M_i of the sequence (1) is equipped with a Riemannian singular metric, denoted with $g^{(i)}$, obtained via the pullback of $\mathcal{N}_{(i)}$. The pseudolength of a curve γ on the i-th manifold, namely $Pl_i(\gamma)$, is computed via the relative metric $g^{(i)}$ via (3).

2.1 General results

We depict hereafter the theoretical bases of our approach. We denote with $\mathcal{N}_i$ the submap $\Lambda_i \circ \cdots \circ \Lambda_n : M_i \to M_n$, and with $\mathcal{N} \equiv \mathcal{N}_0$ the map describing the action of the complete network. The starting point is to consider the pair (M_i, Pd_i): this is a pseudometric space, which can be turned into a full-fledged metric space $M_i/\sim_i$ by the metric identification $x \sim_i y \Leftrightarrow Pd_i(x, y) = 0$. The first result states that the length of a curve on the i-th manifold is preserved among the mapping on the subsequent manifolds.

Proposition 1. *Let $\gamma : [0, 1] \to M_i$ be a piecewise $\mathcal{C}^1$ curve. Let $k \in \{i, i + 1, \cdots, n\}$ and consider the curve $\gamma_k = \Lambda_k \circ \cdots \circ \Lambda_i \circ \gamma$ on M_k. Then $Pl_i(\gamma) = Pl_k(\gamma_k)$.*

In particular this is true when $k = n$, *i.e.*, the length of a curve is preserved in the last manifold. This result leads naturally to claim that if two points are in the same class of equivalence, then they are mapped into the same point under the action of the neural network.

Proposition 2. *If two points $p, q \in M_i$ are in the same class of equivalence, then $\mathcal{N}_i(p) = \mathcal{N}_i(q)$.*

The next step is to prove that the sets $M_i/\sim_i$ are actually smooth manifolds: to this aim, we introduce another equivalence relation: $x \sim_{\mathcal{N}_i} y$ if and only if there exists a piecewise $\gamma : [0, 1] \to M_i$ such that $\gamma(0) = x, \gamma(1) = y$ and $\mathcal{N}_i \circ \gamma(s) = \mathcal{N}_i(x)\ \forall s \in [0, 1]$. The introduction of this equivalence relation allows us to easily state the following proposition.

Proposition 3. *Let $x, y \in M_i$, then $x \sim_i y$ if and only if $x \sim_{\mathcal{N}_i} y$.*

The following corollary contains the natural consequences of the previous result; the second point of the claim below is the counterpart of Proposition 2.

Corollary 1. *Under the hypothesis of Proposition 3, one has that $M_i/\sim_i = M_i/\sim_{\mathcal{N}_{i+1}}$. Moreover, if two points $p, q \in M_i$ are connected by a $\mathcal{C}^1$ curve $\gamma : [0, 1] \to M_i$ satisfying $\mathcal{N}_i(p) = \mathcal{N}_i \circ \gamma(s)$ for every $s \in [0, 1]$, then they lie in the same class of equivalence.*

Making use of the Godement's criterion, we are now able to prove that the set $M_i/\sim_i$ is a smooth manifold, together with its dimension.

Proposition 4. $\dfrac{M_i}{\sim_i}$ *is a smooth manifold of dimension $dim(\mathcal{N}(M_0))$.*

This last achievement provides practical insights about the projection π_i on the quotient space, that consists the building block of the algorithms used for recovering and exploring the equivalence classes of a neural network.

Proposition 5. *$\pi_i : M_i \to M_i/\sim_i$ is a smooth fiber bundle, with $Ker(d\pi_i) = \mathcal{V}M_i$, which is therefore an integrable distribution. $\mathcal{V}M_i$ is the vertical bundle of M_i. Every class of equivalence $[p]$ is a path-connected submanifold of M_i and coincide with the fiber of the bundle over the point $p \in M_i$.*

3 Methodology

The results depicted in Section 2.1 provide powerful tools for investigating how a neural network sees the input space starting from a point x. In particular we point out the following remarks: i) If two points x, y belonging to the input manifold M_0 are are such that $x \sim_0 y$, then $\mathcal{N}(x) = \mathcal{N}(y)$; ii) given a point $p \in M_n$, the counterimage $\mathcal{N}^{-1}(p)$ is a smooth manifold, whose connected components are classes of equivalences in M_0 with respect to $\sim_0$. A necessary condition for two points $x, y \in M_0$ to be in the same class of equivalence is that $\mathcal{N}(x) = \mathcal{N}(y)$; iii) any class of equivalence $[x]$, $x \in M_0$, is a maximal integral submanifold of $\mathcal{V}M_0$. The above observations directly provide with a strategy to build up the equivalence class of an input point $x \in M_0$. Proposition 5 tells us that $\mathcal{V}M_0$ is an integrable distribution, with dimension equal to the dimension of the kernel of $g^{(0)}$: we can hence find $dim(Ker(g^{(0)}))$ vector fields which are a base for the tangent space of M_0. This means that we can compute the eigenvalue decomposition of $g^{(0)}_x$ and consider the L linearly independent eigenvectors, namely $\{v_l\}_{l=1,\dots,L}$, associated to the null eigenvalue: these eigenvectors depend *smoothly* on the point, a fact that is not trivial when the matrix associated to the metric depends on several parameters [15]. We can build then all the null curves by randomly selecting one eigenvector $\tilde{v} \in \{v_l\}$ and then reconstruct the curve along the direction $\tilde{v}$ from the starting point x. From a practical point of view, one is led to solve the Cauchy problem, a first order differential equation, with $\dot{\gamma} = \tilde{v}$ and initial condition $\gamma(0) = x$.

3.1 Input Space Exploration

This whole procedure is coded in the Singular Metric Equivalence Class (SiMEC) and the Singular Metric Exploration (SiMExp) algorithms, whose general schemes are depicted in Algorithms 1 and 2. SiMEC reconstructs the class of equivalence of the input via the exploration of the input space by randomly selecting one of the eigenvectors related to the zero eigenvalue. On the opposite, in SiMExp, in order to move from a class of equivalence to another we consider the eigenvectors relative to the nonzero eigenvalues. This requires the slight difference in lines 5 to 7 between Algorithm 1 and Algorithm 2.

Algorithm 1 The Singular Metric Equivalence Class (SiMEC) algorithm.

1: Set the network $\mathcal{N}$; choose the maximum number of iterations. Choose the input p_0.
2: **for** $k = 0, 1, \dots, K-1$ **do**
3: Compute $g^n_{\mathcal{N}(p_k)}$
4: Compute the pullback metric $g^0_{p_k}$
5: Diagonalize $g^0_{p_k}$ and find the eigenvectors $\{v_l\}_l$ associated to the zero eigenvalue
6: Randomly select $\tilde{v} \in \{v_l\}_l$
7: $\delta = 1/\sqrt{\max(\text{eigenvalues of } g^0_{p_k})}$
8: $p_{k+1} \leftarrow p_k + \delta\tilde{v}$
9: **end for**
10: Optionally: store $\{p_k\}_{k=0,\dots,K}$ for optimizing future computations
11: Project p_k to the nearest feasible region

Algorithm 2 The Singular Metric Exploration (SiMExp) algorithm.

1: Set the network $\mathcal{N}$; choose the maximum number of iterations. Choose the input p_0.
2: **for** $k = 0, 1, \dots, K-1$ **do**
3: Compute $g^n_{\mathcal{N}(p_k)}$
4: Compute the pullback metric $g^0_{p_k}$
5: Diagonalize $g^0_{p_k}$ and find the eigenvectors $\{w_l\}_l$ associated to the non-zero eigenvalue
6: Randomly select $\tilde{w} \in \{w_l\}_l$
7: $\delta = 2/\sqrt{\max(\text{eigenvalues of } g^0_{p_k})}$
8: $p_{k+1} \leftarrow p_k + \delta\tilde{w}$
9: **end for**
10: Optionally: store $\{p_k\}_{k=0,\dots,K}$ for optimizing future computations
11: Project p_k to the nearest feasible region

There are some remarks to point out. From a numerical point of view, the diagonalization of the pullback may lead to have even negative eigenvalues: hence one may use the notion of energy of a curve, related to the pseudolength. The update rule for the new point (line 8) amounts to solve the differential problem via the Euler method: for a reliable solution, we suggest to choose a small step-length δ. On the other hand, if the value of δ is too small more iterations are needed to move

away from the starting point sensibly. Therefore there is a trade-off between the reliability of the solution and the exploration pace. The proof of the well-posedness theorem for Cauchy problems, cf. [18, Theorem 2.1], yields some insights, suggesting to set δ equal to the inverse of the Lipschitz constant of the map $\mathcal{N}$ – which in practice we can estimate with the inverse of the square root of the largest eigenvalue λ_M of the pullback metric $g^0_{p_k}$. This is our default choice for Algorithm 1. We also note that Algorithm 1 is more sensitive to the choice of the parameter δ compared to Algorithm 2. To build points in the same equivalence class Algorithm 1 needs to follow a null curve closely with as little approximations as possible, namely with a small δ. In contrast Algorithm 2, whose goal is to change the equivalence class from one iteration to the next, does not have the same problem and larger δ are allowed. Out default choice is therefore to set $\delta = 2\lambda_M^{-1/2}$ for Algorithm 2. As for the computational complexity of the two algorithms, the most demanding step is the computation of the eigenvalues and eigenvectors, which is $O(n^3)$, with n the dimension of the square matrix $g^0_{p_k}$ [20]. Since all the other operations are either $O(n)$ or $O(n^2)$, we conclude that the complexity of both Algorithms 1 and 2 is $O(n^3)$.

3.2 Interpretability

Algorithms 1 and 2 allow for the exploration of the equivalence classes in the input space of a Transformer model. However, the points explored by these algorithms may not be directly interpretable by a human perspective. For instance, an image or a piece of text may need to be decoded to be "readable" by a human observer. Furthermore, we present an interpretation of the eigenvalues of the pullback metric which allows us to define a feature importance metric. We present two interpretability methods for Transformers based on input space exploration. Both methods are then demonstrated on a Vision Transformer (ViT) trained for digit classification [8], and two BERT models, one trained for hate speech classification and the other trained for MLM [7, 19].

Algorithm 3 Feature Importance Analysis Using Pullback Metric $g^0_{x^e}$

Inputs:
    Transformer model T with: Tokenizer t_T, Embedding layer e_T, Intermediate layers l_T
    Input data x
Tokenize input x to obtain tokens $x^t = t_T(x)$
Compute embeddings $x^e = e_T(x^t)$
Compute intermediate representations $g^n_{l_T(x^e)}$
Calculate the pullback metric $g^0_{x^e}$
Diagonalize $g^0_{x^e}$ to extract eigenvalues
Identify the maximum eigenvalue for each embedding, indicating its importance
Output: Heatmap of embedding importance based on the eigenvalues

Algorithm 4 Exploration of Embedding Space in Transformers

Inputs:
    Transformer model T with: Tokenizer t_T, Embedding layer e_T, Intermediate layers l_T
    Input data x (image or text)
Retrieve segments $x^t = t_T(x)$.
Choose segments $P = \{p | p \in x^t\}$ for updates; keep others unchanged.
Compute embeddings $x^e = e_T(x^t)$.
Apply SiMEC or SiMExp on x^e, updating embeddings for segments in P.
Outputs: Modified input embedding, one for each SiMEC/SiMExp iteration.

Feature importance. Consider a Transformer model T whose architecture includes a tokenizer t_T (or patcher for images) that segments the input so that each segment can be converted into a continuous representation by an embedding layer e_T. This results in a matrix of dimensions $n_s \times h$, where n_s represents the number of segments, and h denotes the hidden size of the model's embeddings. The eigenvalues of the pullback metric can be used to deduce the importance of each embedding and, by extension, the significance of the segments they represent, with respect to the final prediction. The process for determining the importance of textual tokens or image patches is outlined in Algorithm 3. The appearance of the resulting heatmaps varies according to the type of input used. An example of experiments with ViT on the MNIST dataset [12] is shown in Figure 1 that depicts heatmaps for two MNIST instances. Figure 2, on the left, illustrates two experiment using Algorithm 3 on both a BERT model for hate speech detection and a BERT model for MLM.

Interpretation of input space exploration. Using SiMEC and SiMExp to explore the embedding space reveals how Transformer models perceive equivalence among different data points. Specifically, these methodologies facilitate the sequential acquisition of embedding matrices $p_0 \dots p_K$ at each iteration, as detailed in Algorithms 1 and 2. Algorithm 4 implements a practical application of the

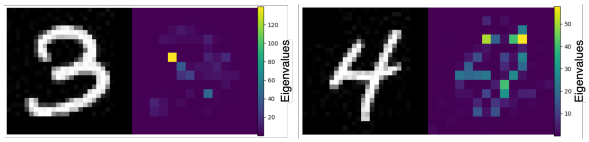

Figure 1: Example output from Algorithm 3 applied to digit classification. These two instances are predicted as 3 (left) and 4 (right). The brightness of the color indicates the eigenvalue's magnitude. The brighter the color, the more sensitive the patch. This indicates that changes in the values of these sensitive patches are likely to have a greater impact on the prediction probabilities. Each patch in the heatmap corresponds to a 2×2 square pixel.

Figure 2: Example outputs from Algorithm 3. The darker the color, the higher the token's eigenvalue. Left: The sentence analysed is classified as "offensive" by the BERT for hate speech detection, with significant contributions from tokens `[CLS]`, `politicians`, `corrupt`, and `##eit` (part of the word *deceitful*). Right: Example instance processed by a BERT model for masked language modeling. `[MASK]` is predicted as "ham", with the most influential tokens being `pizza` and `cheese`.

SiMEC/SiMExp approach with Transformer models. A key feature of this method is its ability to selectively update specific tokens (for text inputs) or patches (for image inputs) during each iteration. This selective updating allows us to explore targeted modifications that prompt the model to either categorize different inputs as the same class or recognize them as distinct. Unlike traditional approaches where modifications are predetermined, this method lets the model itself guide us to understand which data points belong to specific equivalence classes. To interpret embeddings resulted from the exploration process, they must be mapped back into a human-understandable form, such as text or images. The interpretation of an embedding vector depends on the operations performed by the Transformer's embedding module e_T. If e_T consists only of invertible operations, it is feasible to construct a layer that performs the inverse operation relative to e_T. The output can then be visualized and directly interpreted by humans, allowing for a comparison with the original input to discern how differences in embeddings reflect differences in their representations (e.g., text, images). If the operations in e_T are non-invertible, a trained decoder is required to reconstruct an interpretable output from each embedding matrix $p_0 \dots p_K$. When using a BERT model, it is feasible to utilize layers that are specialized for the masked language modeling (MLM) task to map input embeddings back to tokens. This approach is effective whether the BERT model in question is specifically designed for MLM or for sentence classification. In the case of sentence classification models, it is necessary to select a corresponding MLM BERT model that shares the same internal architecture, including the number of layers and embedding size.

Algorithm 5 depicts the process of interpreting Algorithm 4 outputs for both ViT and BERT experiments. After initializing the decoder according to the model type, the embeddings $p_0 \dots p_K$ need to be constrained to a feasible region. This region is defined by the distribution of embeddings derived from the original input instances. Next, the embeddings are decoded, and the selected segments for exploration are extracted. These segments are then used to replace the corresponding parts of the original input instance. Figure 3 depicts an example outcome of Algorithm 5 applied on a ViT exploration experiment. Given that the interpretation process includes both a capping step and a decoding step (lines 10 and 11 of Algorithm 5), it's important to note that there isn't a direct 1:1 correspondence between each iteration's update and the interpretation outcomes. Our primary focus is on exploring the input embedding space, rather than the input image or input sentence spaces. For further investigation, we provide a detailed discussion on considering interpretation outputs as alternative prompts in Section 4.

Algorithm 5 Interpretation for Exploration results for ViT and BERT models.

Inputs:
    Transformer model T with: Tokenizer t_T, Embedding layer e_T, Intermediate layers l_T
    Modified embeddings $p_0 \dots p_K$ resulted from Algorithm 4 applied on an input x
    $P = \{p | p \in x^t\}$ indices of updated segments
If T **is** ViT:
    Initialize decoder d with weights from e_T.
If T **is** BERT:
    Initialize decoder with intermediate and final layers of a BERT for MLM task.
Compute embeddings distributions for original input data
Use the original embeddings distributions to cap $p_0 \dots p_K$
Decode modified embeddings $p_0 \dots p_K$ using d to generate the corresponding images/sentences $X' = x'_0 \dots x'_K$.
For each $x' \in X'$: replace segments relative to indices P in x with those in x'.
Outputs:
Modified input images/sentences, one for each SiMEC/SiMExp iteration.

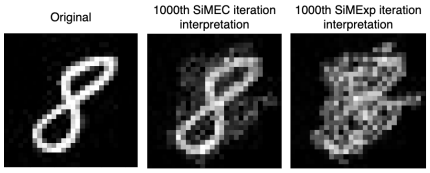

Figure 3: Example of SiMEC and SiMExp output interpretation for ViT digit classification. Left: Original MNIST image of an "8". Center: Interpretation of a p_{1000} from a SiMEC experiment, where p_{1000} is predicted as "8". Right: Interpretation of a p_{1000} from a SiMExp experiment, where p_{1000} is predicted as "4". All patches are subject to SiMEC and SiMExp updates.

4 Experiments

Experiments are conducted on textual and visual data. We aim to perform a preliminary investigation of 3 features of our approach: (i) how the class probability changes on the decoded output of SiMEC/SiMExp, (ii) what is the trade-off between the quantity and the quality of the output, and (iii) how our method can be used to extract feature importance-based explanations.

In the textual case, we experiment with hate speech classification datasets: we use HateXplain[2] [13], which provides a ground truth for feature importance, plus a sample of 100 hate speech sentences generated by prompting ChatGPT[3], which serve purposes (i) and (ii). In the visual case, we perform experiments on MNIST [12] dataset.

Using interpretation outputs as alternative prompts An interesting investigation is to determine if our interpretation algorithm (Algorithm 5) can generate alternative prompts that stay in the same equivalence class as the original input data or move to a different one, based on SiMEC and SiMExp explorations. We test how the probability assigned to the original equivalence class by the Transformer model changes as the SiMEC and SiMExp algorithms explore the input embedding manifold.

For BERT experiments we generate prompts to inspect the probability distribution over the vocabulary for tokens updated by Algorithms 1 and 2. We decode the updated $p_0 \dots p_K$ using Algorithm 5, focusing on tokens updated through the iterations. For each of these decoded tokens, we extract the top-5 scores to obtain 5 alternative tokens to replace the original ones, creating 5 alternate prompts. We then extract the prediction $i^* = \arg\max_i y_i$ for the original sentence, which represents the output whose equivalence class we aim to explore. Finally, we classify the new prompts, obtaining the corresponding predictions $Y = \mathbf{y}^{(0)} \dots \mathbf{y}^{(K)}$, where each $\mathbf{y}^{(k)} \in \mathbb{R}^N$, N being the number of prediction classes. We visualize the prediction trend for the i^*th value in every $\mathbf{y}^{(0)} \dots \mathbf{y}^{(K)}$ categorizing the images into two subsets: those that lead to a change in prediction $Y_c = \{\mathbf{y}^{(k)} \in Y \mid \arg\max_i y_i^{(k)} \neq i^*\}$ and those that don't $Y_s = \{\mathbf{y}_i \in Y \mid \arg\max_i y_i^{(k)} = i^*\}$.

[2]MIT License

[3]Used prompts are included in the Supplementary Materials.

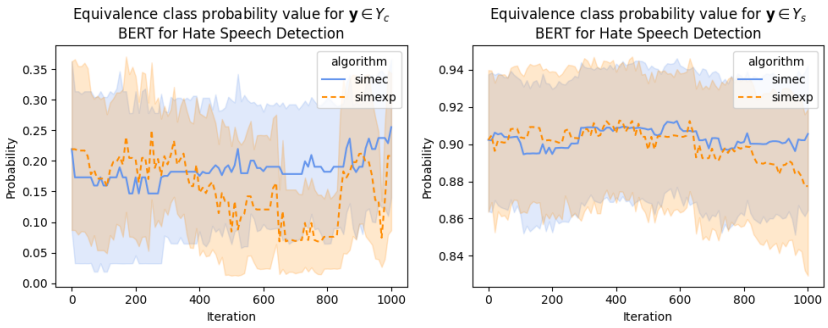

Figure 4: Analysis involving results SiMEC and SiMExp applied to BERT for hate speech detection. Left: Prediction values for i^* for each $\mathbf{y} \in Y_c$. Right: Prediction values for $\mathbf{y} \in Y_s$.

Sentence classification experiments[4] involved 1000 iterations from both SiMEC and SiMExp, applied to a subset of 8 sentences from the ChatGPT hate speech dataset. The plot on the left side of Figure 4 illustrates that, as the original embeddings are increasingly modified, SiMExp tends to produce alternatives with lower prediction values for i^* compared to SiMEC. Thus, even if predictions change in SiMEC experiments, the equivalence class prediction value remains approximately constant and higher than in SiMExp. Considering the plot on the right side of Figure 4, SiMExp identifies prompts that lower the prediction value for i^*. ViT and MLM experiments are detailed in the Supplementary Materials.

Input space exploration We measure the time required to explore the input space of a ViT with the SiMEC algorithm and compare it with a perturbation-based method. The perturbation-based method mimics a trial-and-error approach as it takes an input image and, at each iteration, perturbs it by a semi-random vector $\mathbf{v}_{t+1} = a_t \mathbf{v}_t + \eta\epsilon$, where $a_t = 1$ if $y_t = y_{t-1}$, $a_t = -1$ otherwise, ϵ is an orthogonal random vector from a standard normal distribution and η is the step length. With the perturbation, we obtain a new image, then check whether the model yields the same label for the new image. The perturbation vector is re-initialized at random from a normal distribution 20% of the times to allow for exploration. We construct this method to have a direct comparison with ours in the absence of a consolidated literature about the task.

We train a ViT model having 4 layers and 4 heads per layer on the MNIST dataset[5]. The SiMEC algorithm is run for 1000 iterations, so that it can generate 1000 examples starting from a single image. In a sample of 100 images, the average time is approximately 339 seconds.[6] In the same time, the perturbation-based algorithm can produce up to 36000 images. However, we notice that the perturbation-based algorithm ends up producing monochrome (pixel color has zero variance) or totally noisy images, which provide little information about the behavior of the model. Excluding only the images with low color variance (< 0.01), we are left, on average, with 19 images (standard deviation 13.9). SiMEC, in contrast, doesn't present this behavior, as all 1000 images have high enough intensity variance and are thus useful for explainability purposes.

As BERT has many more parameters with respect to our ViT model, processing textual data takes longer. Specifically, in a sample of 16 sentences, the average time needed to run 1000 iterations on a sentence is 7089 seconds, taking into account both MLM and classification experiments.

Feature importance-based explanations We compare our method against Attention Rollout (AR) [1] and the Relevancy method proposed by Chefer et al. [6]. In the textual case, we provide a quantitative evaluation using the HateXplain dataset, which contains 20147 sentences (of which 1924 in the test set) annotated with *normal*, *offensive* and *hate speech* labels as well as the positions of words that support the label decision. We then measure the cosine similarity between the importance assigned by each method to each word in a sentence and the ground truth. Notice that, since the

[4]Model used: `huggingface.co/ctoraman/hate-speech-bert`

[5]Using Adam optimizer, the model achieved the highest validation accuracy (96.25%) in 20 epochs.

[6]All experiments are based on the current PyTorch implementation of the algorithms and run on a Ubuntu 20.04 machine endowed with one NVIDIA A100 GPU and CUDA 12.4.

dataset contains multiple annotations, the ground truth y for each word w is obtained as the average of the binary labels assigned by each annotator, and therefore $y(w) \in [0; 1]$. We also normalize all scores in $[0; 1]$ so to have them on the same scale. The average similarity achieved by our method is **0.707** (standard deviation $\sigma = 0.302$), against **0.7** ($\sigma = 0.315$) for Relevancy and **0.583** ($\sigma = 0.318$) for AR. This proves our method to be more effective in finding the most sensitive tokens for classification. We provide an example on image classification in the Supplementary Materials.

5 Related work

Our work relates to embedding space exploration literature, and has at least one collateral applications in the XAI domain, namely producing feature importance-based explanations.

Embedding space exploration. Works dealing with embedding space exploration mostly focus on the study of specific properties of the embedding space of Transformers, especially in NLP. For instance, Cai et al. [5] challenge the idea that the embedding space is inherently anisotropic [10] discovering local isotropy, and find low-dimensional manifold structures in the embedding space of GPT and BERT. Biś et al. [3] argue that the anisotropy of the embedding space derives from embeddings shifting in common directions during training. In the field of CV, Vilas et al. [21] map internal representations of a ViT onto the output class manifold, enabling the early identification of class-related patches and the computation of saliency maps on the input image for each layer and head. Applying Singular Value Decomposition to the Jacobian matrix of a ViT, Salman et al. [17] treat the input space as the union of two subspaces: one in which image embedding doesn't change, and another one for which it changes. Except for the last one, all the aforementioned approaches rely on data samples. By studying the inverse image of the model, instead, we can do away with data samples.

Feature importance-based explanations. Feature importance is a measure of the contribution of each data feature to a model prediction. In the context of Computer Vision and Natural Language Processing, it amounts to giving a weight to pixels (or patches of pixels) in an image and tokens in a piece of text, respectively. In recent years, much research has focused on Transformers in both CV and NLP. Most approaches are based on the attention mechanism of the Transformer architecture. Abnar and Zuidema [1] quantify the overall attention of the output on the input by computing a linear combination of layer attentions (Attention Rollout) or applying a maximum flow algorithm (Attention Flow). To overcome the limitations [4] of attention-based methods, Hao et al. [11] use the concept of *attribution*, which is obtained by multiplying attention matrices by the integrated gradient of the model with respect to them. Chefer et al. [6] propose the Relevancy metric to generalize attribution to bi-modal and encoder-decoder architectures. Other methods are perturbation-based, where perturbations of input data are used to record any change in the output and draw a saliency map on the input. In order to overcome the main issue with such methods, i.e. the generation of outlier inputs, Englebert et al. [9] apply perturbations after the position encoding of the patches. In contrast with these methods, ours does not need arbitrary perturbations of inputs, and considers all parameters of the model, not only the attention query and key matrices.

6 Conclusions

Our exploration of the Transformer architecture through a theoretical framework grounded in Riemannian Geometry led to the application of our two algorithms, SiMEC and SiMExp, for examining equivalence classes in the Transformers' input space. We demonstrated how the results of these exploration methods can be interpreted in a human-readable form and conducted preliminary investigations into their potential applications. Notably, our methods show promise for ranking feature importance and generating alternative prompts within the same or different equivalence classes.

Future research directions include expanding our experimental results and delving deeper into the potential of our framework for controlled input generation within an equivalence class. This application holds significant promise for enhancing the explainability of Transformer models' decisions and for addressing issues related to bias and hallucinations.

References

[1] S. Abnar and W. Zuidema. Quantifying attention flow in transformers. 2020. arXiv:2005.00928.

[2] A. Benfenati and A. Marta. A singular Riemannian geometry approach to Deep Neural Networks I. Theoretical foundations. *Neural Networks*, 158:331–343, 2023.

[3] D. Biś, M. Podkorytov, and X. Liu. Too much in common: Shifting of embeddings in transformer language models and its implications. In *Proceedings of the 2021 conference of the North American chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5117–5130, 2021.

[4] G. Brunner, Y. Liu, D. Pascual, O. Richter, M. Ciaramita, and R. Wattenhofer. On identifiability in transformers. *International Conference on Learning Representations*, 2019.

[5] X. Cai, J. Huang, Y. Bian, and K. Church. Isotropy in the contextual embedding space: Clusters and manifolds. In *International conference on learning representations*, 2020.

[6] H. Chefer, S. Gur, and L. Wolf. Generic attention-model explainability for interpreting bi-modal and encoder-decoder transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 397–406, 2021.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423`.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. arXiv:2010.11929.

[9] A. Englebert, S. Stassin, G. Nanfack, S. A. Mahmoudi, X. Siebert, O. Cornu, and C. De Vleeschouwer. Explaining through transformer input sampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 806–815, 2023.

[10] J. Gao, D. He, X. Tan, T. Qin, L. Wang, and T.-Y. Liu. Representation degeneration problem in training natural language generation models. In *International conference on learning representations*, volume abs/1907.12009, 2019. URL `https://api.semanticscholar.org/CorpusID:59317065`.

[11] Y. Hao, L. Dong, F. Wei, and K. Xu. Self-attention attribution: Interpreting information interactions inside transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12963–12971, 2021.

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[13] B. Mathew, P. Saha, S. M. Yimam, C. Biemann, P. Goyal, and A. Mukherjee. Hatexplain: A benchmark dataset for explainable hate speech detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 14867–14875, 2021.

[14] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The Limitations of Deep Learning in Adversarial Settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387, Mar. 2016. doi: 10.1109/EuroSP.2016.36.

[15] F. Rellich and J. Berkowitz. *Perturbation Theory of Eigenvalue Problems*. New York University. Institute of Mathematical Sciences. Gordon and Breach, 1969. ISBN 9780677006802.

[16] M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[17] S. Salman, M. M. B. Shams, and X. Liu. Intriguing equivalence structures of the embedding space of vision transformers, 2024. arXiv:2401.15568.

[18] M. E. Taylor. *Partial differential equations. I: Basic theory*, volume 115 of *Appl. Math. Sci.* Cham: Springer, 3rd corrected and expanded edition edition, 2023. ISBN 978-3-031-33858-8; 978-3-031-33861-8; 978-3-031-33859-5. doi: 10.1007/978-3-031-33859-5.

[19] C. Toraman, F. Şahinuç, and E. H. Yilmaz. Large-scale hate speech detection with cross-domain transfer. In *Proceedings of the Language Resources and Evaluation Conference*, pages 2215–2225, Marseille, France, June 2022. European Language Resources Association. URL `https://aclanthology.org/2022.lrec-1.238`.

[20] L. N. Trefethen and D. I. Bau. *Numerical linear algebra. Twenty-fifth anniversary edition*, volume 181 of *Other Titles Appl. Math.* Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2022. ISBN 978-1-61197-715-8.

[21] M. G. Vilas, T. Schaumlöffel, and G. Roig. Analyzing vision transformers for image classification in class embedding space. *Advances in Neural Information Processing Systems*, 36, 2024.

[22] M. Wu, H. Wu, and C. Barrett. Verix: Towards verified explainability of deep neural networks. *Advances in neural information processing systems*, 36, 2024.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In the abstract and introduction we claim that we present a method for the exploration of equivalence classes in the input space of Transformer models, which is analyzed in depth in Section 3. The mathematical theory we refer to is deepened in Section 2.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations and tradeoff given by numeric integration in Subsection 3.1 and theoretical assumptions are enumerated in Section 2. Computational efficiency of our algorithms is discussed in Subsection 3.1. We conducted experiments on 3 datasets only, one of which of small dimensions since our main focus is on the mathematical theory grounding the application of the method to Transformers, as stated at the beginning of Section 4. Other limitations are mentioned throughout Section 4, including the fact that our investigations in the human-readable scenario are at a preliminary stage.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The full proofs are part of two previously published papers which we cannot disclose for anonymity requirements. We replicate the relevant proofs in the supplementary material, part of which will be removed from the final version of the paper, referencing to the other papers.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Pseudo-code of the proposed algorithms is reported in Subsections 3.1 and 3.2 so to make the algorithms reproducible, plus our implementation is made available in the supplementary material. Experiments, including the complete setting, and the respective baselines are described in Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All experiments are made reproducible through scripts provided as supplementary material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All details are provided in Section 4: details of the analyzed architectures, number of iterations of the SiMEC/SiMExp algorithms, technical infrastructure on which the experiments were performed, amount of data the experiments were performed on.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The standard deviation is reported for the experiment that supports the claims of the paper, i.e. the one on feature importance-based explanations, in Section 4. Standard deviation is also reported for the number of uninformative images produced by the perturbation-based baseline method in the Input space exploration experiment.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: All the experiments were performed on the same infrastructure, which is reported in a footnote in Section 4. Time of execution is one of the key indicators reported for the Input space exploration experiments. More computing power would be required for experiments on bigger Transformer models.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: We comply with the terms of use of the datasets employed in the experiments, and we deem our work has no potentially harmful effect on people safety, security, discrimination, surveillance, harassment, nor on human rights. Our proposal does not contribute to spread bias and unfairness towards certain groups of people nor to harm the environment.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [No]

Justification: Although the impacts of XAI on society is broad and deep, in this paper we focus only on the technical problem of exploring the equivalence classes in the input space of Transformers, which doesn't add any specific impact to the discussion about XAI in general.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The datasets used in the paper are explicitely mentioned in the references, as required by the terms of use. Where applicable, the license is also reported.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our work doesn't include crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our work does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.