Permutation Equivariance of Transformers and Its Applications

Hengyuan Xu[†], Liyao Xiang[*†], Hangyu Ye[†], Dixi Yao[‡], Pengzhi Chu[†] and Baochun Li[‡]

[†]Shanghai Jiao Tong University
[‡]University of Toronto

Abstract

Revolutionizing the field of deep learning, Transformer-based models have achieved remarkable performance in many tasks. Recent research has recognized these models are robust to shuffling but are limited to inter-token permutation in the forward propagation. In this work, we propose our definition of permutation equivariance, a broader concept covering both inter- and intra- token permutation in the forward and backward propagation of neural networks. We rigorously proved that such permutation equivariance property can be satisfied on most vanilla Transformer-based models with almost no adaptation. We examine the property over a range of state-of-the-art models including ViT, Bert, GPT, and others, with experimental validations. Further, as a proof-of-concept, we explore how real-world applications including privacy-enhancing split learning, and model authorization, could exploit the permutation equivariance property, which implicates wider, intriguing application scenarios. The code is available at https://github.com/Doby-Xu/ST

1. Introduction

Originating as a tool in natural language processing, Transformer now has permeated various fields such as computer vision, multi-modal tasks, etc. Transformer model, introduced in [32], has revolutionized the way of approaching sequence-based tasks, and further demonstrates versatility and power in a diverse set of tasks with the development of Bert[6], GPT [3, 27], ViT [8], etc. Meanwhile, its intriguing property is discovered, e.g., the outputs being robust or invariant to token shuffling [17, 26]. However, previous works either show the property through empirical observation [26], or by modifying the original model structure [17], typically for input-order-independent tasks.

The shuffling invariance property of Transformer is widely recognized but not clearly understood by the community. ViT shows superior permutation robustness compared to CNN in patch shuffling where the spatial information is totally disrupted [26], indicating that self-attention is invariant to patch ordering. Similarly, [17] proposes Set Transformer, satisfying *permutation equi-/in-variance* — the output of the model should not change under any permutation of the elements in the input set. Their model inherits the network architectures proposed by [36] where each input element is first independently fed into a feed-forward neural network and then aggregated by a pooling operation.

As we found, the previous notion of *permutation equi-/in-variance* is quite limited. In this work, we propose our definition of *permutation equivariance* — meaning that the model trained over any inter- or intra-token permutation is equivalent to the model trained over normal inputs, in contrast to the inter-token permutation in previous works. Our definition essentially pushes one step forward in two-folds: first, the former definition works at a coarser granularity, i.e., tokens are exchanged and permuted while ours incorporates both inter- and intra-token shuffling. Second, our proposed property is stricter in the sense that, it not only requires output equivariance under input permutation in forward propagation (forwarding equivariance) but also demands model weights to be equivalently trained in the backward propagation (backprop equi-/in-variance). We show such backprop equi-/in-variance is closely related to forwarding equivariance by modeling the permutation as row/column shuffling in matrices.

Our findings inspire many potential applications in real-world scenarios. We present a simple, yet effective privacy-enhancing technique for feature offloading, based on the permutation equivariance property. An honest-but-curious cloud merely provides computational service while being blind to the training data, inference data, and the trained model. We show that our method remarkably improves the data utility-privacy tradeoff which is intensively investigated in privacy-preserving split learning [15, 28, 33].

[*]Corresponding author: xiangliyao08@sjtu.edu.cn. The research was supported in part by National Science and Technology Major Project 2021ZD0112801, NSF China (62272306, 62032020, 62136006).

Moreover, the trained model is protected from being effectively fine-tuned by unauthorized parties, yet with almost no impact on the model's performance.

Highlights of our contributions include: first, we reveal the permutation equivariance property in the forward and backward propagation of neural networks. Second, by analyzing the inner workings of the Transformer model, we prove that a wide range of models satisfy permutation equivariance. Third, as a proof-of-concept, we design a privacy-enhancing mechanism and a model authorization scheme to show the promising use of the property. A series of experiments demonstrate the superiority of our design to the state-of-the-art methods in the application scenes.

2. Related Works

2.1. Transformer

Dispensing with recurrence and convolutions, Transformer [32] shows superior performance solely on attention mechanisms in translation tasks, and soon becomes the de-facto standard for natural language processing (NLP). Models such as Bert [6], GPT [3, 27], and a myriad of Transformer-based counterparts [18, 20] have consistently achieved state-of-the-art results across a wide spectrum of tasks. Most recently, with Transformer models as the core, Large Language Models (LLMs) have achieved great success in recognizing, translating, predicting, or generating text or other contents.

Inspired by its success in NLP, Vision Transformer (ViT [8]) was designed and its performance exceeds the state-of-the-art convolutional networks on a variety of image tasks such as classification [31], object detection [19], and semantic segmentation [5]. Substantial efforts have also been devoted into designing powerful pre-trained models [2, 30, 37] with ViT-based backbone, leading to the emergence of a growing family of networks [1, 4, 5, 19].

2.2. Permutation Equi-/In-variance

Permutation invariance property refers to that the output value for a given set is the same regardless of the order of objects in the set [17, 36]. A closely related property permutation equivariance describes that for function f, any permutation $\boldsymbol{P}$ and input $\boldsymbol{X}$, $f(\boldsymbol{P}\boldsymbol{X}) = \boldsymbol{P}f(\boldsymbol{X})$. Commonly, the permutation takes place at the token level and poses additional requirements on models. For example, Deep sets [36] derives the necessary and sufficient conditions for permutation invariance in deep models, i.e., the function can be decomposed in the form $\rho(\sum_{\boldsymbol{X}\in\mathcal{X}}\phi(\boldsymbol{X}))$ for transformations ϕ and ρ. Set transformer [17] further instantiates ϕ and ρ to an adapted encoder and a decoder, respectively. But their permutation invariance limits to the neural network forwarding (or inference), but neglects the invariance in the backward propagation (or training). In contrast, our work does not alter the existing Transformer network structure and illustrates permutation properties in both forward and backward propagations.

Table 1. The notations used.

Symbol	Description	Shape
$\boldsymbol{X}, \boldsymbol{Y}$	Data, label	Task dependent
F_1, F_2	Embedding, task head	Task dependent
$\boldsymbol{Z}$	Feature/ output of F_1	$\mathbb{R}^{n\times d}$
$\hat{\boldsymbol{Y}}$	Prediction/ output of F_2	Same shape w/ $\boldsymbol{Y}$
$\mathrm{Enc}(\cdot)$	Transformer encoder	$\mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$
$\mathrm{T}(\cdot)$	Transformer backbone	$\mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$
$\boldsymbol{P}_R$	Row permutation matrix	$\mathbb{R}^{n\times n}$
$\boldsymbol{P}_C$	Column permutation matrix	$\mathbb{R}^{d\times d}$
$\boldsymbol{W}$	Weights of Transformer	$\mathbb{R}^{t\times d}$

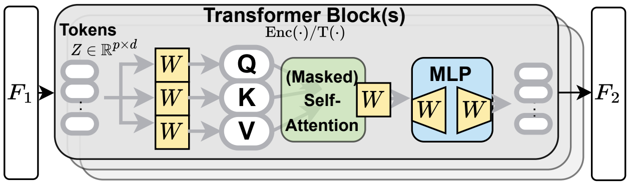
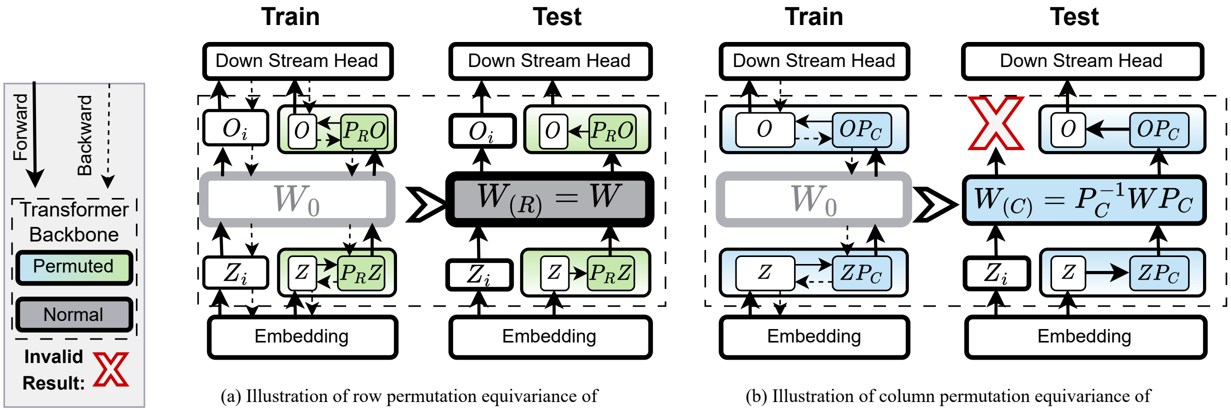


Figure 1. Illustration of Transformer backbone. Learnable weights in permutation are expressed by yellow blocks.

The (forwarding) permutation equivariance property has been widely recognized and applied in many works [9, 22, 26, 29, 33]. It is empirically verified by [26] that ViT is more robust to patch shuffling compared to convolutional networks. Leveraging such a property, Yao *et al.* propose a privacy-preserving split learning framework [33] by patch-shuffling the embedding. The permutation invariance property also sees applications in point cloud processing [9], reinforcement learning [29], neural architecture search [22], etc. However, most target tasks are input-order-insensitive (set-input problem). Different from them, our applications include tasks that rely on the order of the input (e.g., generating text).

3. Notations

We summarize the notations used in this paper in Tab. 1, where n is the number of tokens and d is the dimension of tokens. We use $\boldsymbol{W} \in \mathcal{R}^{t\times d}$ to generally denote the weight matrices of the Transformer backbone including weights in QKV projection, attention projection, and MLP. Details about the location of $\boldsymbol{W}$ in the structure of Transformer backbone are shown in Fig. 1. Transformer backbone usually consists of stacked encoders and decoders. Some components such as bias and LayerNorm do not have matrix-shaped weights, so we leave their discussion to Sec. 4.3.



(a) Illustration of row permutation equivariance of Transformer Encoder. Learned model parameters are the same, with or without permutation in training.

(b) Illustration of column permutation equivariance of Transformer. Model weights are equivalently learned in column permutation.

Figure 2. Illustration of permutation properties. $\boldsymbol{W}$ indicates main parameters in Transformer backbone (stacked Transformer encoders and decoders).

The matrix form of weights allows us to perform row or column permutations. We employ the subscript $_{(R)}$ to refer to row permutation of the corresponding features, weights, derivatives, etc., while using subscript $_{(C)}$ for column permutations. The subscript of $_{(P)}$ generally refers to row, column, and both row and column permutations. Specifically, the row permutation is equivalent to the inter-token permutation in previous works and column permutation is essentially intra-token shuffling. Row/column permutation is expressed by matrix multiplication with the permutation matrix: $\boldsymbol{P}_R\boldsymbol{Z}$, $\boldsymbol{Z}\boldsymbol{P}_C$, or $\boldsymbol{P}_R\boldsymbol{Z}\boldsymbol{P}_C$. The unshuffling is represented by multiplying the inverse of the permutation matrix, i.e., $\boldsymbol{P}_R^{-1}\boldsymbol{Z}_{(P)}$, $\boldsymbol{Z}_{(P)}\boldsymbol{P}_C^{-1}$, or $\boldsymbol{P}_R^{-1}\boldsymbol{Z}_{(P)}\boldsymbol{P}_C^{-1}$.

4. Properties and Proofs

We illustrate the key properties and provide their proofs in this section. Due to space limit, we collect most proofs in Appendix 9 and 10.

4.1. Permutation Equivariance of Transformer

In accordance with the inter-token permutation equivariance of the Transformer encoder [17], the row permutation equivariance can be expressed as

Theorem 4.1 (**Row Permutation Forward Equivariance**). *Transformer encoder is permutation equivariant w.r.t. token permutations, i.e. the row permutation of the input matrix, in forward propagation, i.e.,* $\mathrm{Enc}(\boldsymbol{P}_R\boldsymbol{Z}) = \boldsymbol{P}_R\mathrm{Enc}(\boldsymbol{Z})$ *for any permutation matrix* $\boldsymbol{P}_R \in \mathbb{R}^{n\times n}$.

Following the property, we perform row permutation to the output of the embedding layer F_1 and invert the permutation at the input to the downstream task head F_2, i.e., for

$$\boldsymbol{Z} = F_1(\boldsymbol{X}), \hat{\boldsymbol{Y}} = F_2(\mathrm{Enc}(\boldsymbol{Z})), \tag{1}$$

we perform shuffling and unshuffling as

$$\boldsymbol{Z}_{(R)} = \boldsymbol{P}_R F_1(\boldsymbol{X}), \hat{\boldsymbol{Y}}_{(R)} = F_2(\boldsymbol{P}_R^{-1}\mathrm{Enc}(\boldsymbol{Z}_{(R)})). \tag{2}$$

According to Thm. 4.1, the permuted input to F_2 is $\boldsymbol{P}_R^{-1}\mathrm{Enc}(\boldsymbol{P}_R\boldsymbol{Z}) = \mathrm{Enc}(\boldsymbol{Z})$. Given the shuffling and unshuffling operations, we can derive the following key property:

Theorem 4.2 (**Row Permutation Backward Invariance**). *In backward propagation, gradients of the Transformer encoder in the natural setting (Eq. 1) and the permuted setting (Eq. 2) are the same:*

$$\frac{\partial l}{\partial \boldsymbol{W}} = \frac{\partial l}{\partial \boldsymbol{W}_{(R)}}, \tag{3}$$

given loss function l.

With simple induction, we have:

Corollary 4.3. *The weights of the Transformer encoder learned in the natural setting (Eq. 1) and that learned in the permuted setting (Eq. 2) are the same:*

$$\boldsymbol{W} = \boldsymbol{W}_{(R)}. \tag{4}$$

The shuffling and inverse shuffling procedures and the properties are displayed in Fig. 2(a). In training, token embeddings can be randomly shuffled without affecting the learning results, provided that the input to the task head is inversely permuted. 'Without affecting' here means that the trained weights are exactly the same as the learned weights without any permutation.

Similar to row permutation, we could also perform column shuffling to the input, i.e., intra-token permutation:

$$\boldsymbol{Z}_{(C)} = F_1(\boldsymbol{X})\boldsymbol{P}_C, \hat{\boldsymbol{Y}}_{(C)} = F_2(\mathrm{T}_{(C)}(\boldsymbol{Z}_{(C)})\boldsymbol{P}_C^{-1}). \tag{5}$$

Unfortunately, permutation invariance does not hold column-wise, i.e., $\mathrm{T}(\boldsymbol{Z}\boldsymbol{P}_C) \neq \mathrm{T}(\boldsymbol{Z})\boldsymbol{P}_C$; rather, we could achieve $\mathrm{T}_{(C)}(\boldsymbol{Z}\boldsymbol{P}_C) = \mathrm{T}(\boldsymbol{Z})\boldsymbol{P}_C$ by adjusting the weights in T according to

$$\boldsymbol{W}_{(C)} = \boldsymbol{P}_C^{-1}\boldsymbol{W}\boldsymbol{P}_C. \tag{6}$$

Formally, the property holds as

Theorem 4.4 (Column Permutation Forward Equivariance). *Stacked Transformer encoder and decoder is permutation equivariant w.r.t. column permutations, in forward propagation, i.e., $\mathrm{T}_{(C)}(\boldsymbol{Z}\boldsymbol{P}_C) = \mathrm{T}(\boldsymbol{Z})\boldsymbol{P}_C$ for any permutation matrix $\boldsymbol{P}_C \in \mathbb{R}^{d\times d}$, where weights in $\mathrm{T}_{(C)}$ are weights in T permuted by Eq. 6.*

With Thm. 4.4, we can easily derive that $\hat{\boldsymbol{Y}}_{(C)}$ in Eq. 5 is equal to $\hat{\boldsymbol{Y}}$ in Eq. 1. This property is depicted in Fig. 2(b). The case bears resemblance to homomorphic encryption, wherein the input $\boldsymbol{Z}$ and backbone weights $\boldsymbol{W}$ are 'encrypted' by $\boldsymbol{P}_C$, and the output is 'decrypted' using $\boldsymbol{P}_C^{-1}$. Although shuffling does not offer the same level of security compared to homomorphic encryption, it disguises the original inputs and models to some extent. Only those who possess the key $\boldsymbol{P}_C$ can obtain the correct output and effectively use the model. Without the key, however, it is hard to guess the correct output, or take advantage of the model (see Sec. 5).

Likewise, in the corresponding backprop of column-permuted forwarding (Eq. 5), the property holds as:

Theorem 4.5 (Column Permutation Backward Equivariance). *In backward propagation, gradients of Transformer parameters in the natural setting (Eq. 1) and the permuted setting (Eq. 5) are correlated by the following equation:*

$$\frac{\partial l}{\partial \boldsymbol{W}_{(C)}} = \boldsymbol{P}_C^{-1}\frac{\partial l}{\partial \boldsymbol{W}}\boldsymbol{P}_C \tag{7}$$

where l is the loss function.

With simple induction, we have:

Corollary 4.6. *The weights of Transformer learned in the natural setting (Eq. 1), T, and that learned in the permuted setting (Eq. 5), $\mathrm{T}_{(C)}$, are correlated by $\boldsymbol{W}_{(C)} = \boldsymbol{P}_C^{-1}\boldsymbol{W}\boldsymbol{P}_C$, given the same randomly initialized weights.*

4.2. General Permutation Equivariant Networks

In this section, we show that the row and column permutation equivariance in the forward and backward propagations can be generalized to a broader class of permutation equivariant networks, apart from Transformers. Here permutation generally refers to row and column permutation:

$$\boldsymbol{Z}_{(P)} = \boldsymbol{P}_R F_1(\boldsymbol{X})\boldsymbol{P}_C,\ \ \hat{\boldsymbol{Y}}_{(P)} = F_2(\boldsymbol{P}_R^{-1} f_{(P)}(\boldsymbol{Z}_{(P)})\boldsymbol{P}_C^{-1}), \tag{8}$$

where $f_{(P)}$ is the Transformer backbone f has its weight permuted by Eq. 6.

We show the conclusion in three steps: 1) a majority of common neural network operators satisfy forward permutation equivariance; 2) if an operator is permutation-equivariant in forwarding, it must be permutation-equivariant in the backward propagation; 3) a network composed by the aforementioned operators are both forward and backward permutation-equivariant.

We provide proofs for 1) in Appendix 9 covering a wide variety of operators, such as linear projection, attention, norms, element-wise operators (shortcut skip, Hadamard product, activation, etc.), and softmax. The exception is the type of operators working on sliding windows, e.g., convolutional. For 2) and 3), we will prove the following theorem:

Theorem 4.7 (General Permutation Equivalent Networks)**.** *If f is composed by $f_N \circ f_{N-1} \circ \cdots \circ f_1$ where $f_1, \ldots, f_N$ that are permutation-equivariant in the forward propagation, and each contains weights (if any) as linear arguments, i.e., $f = f_N(\cdots f_2(f_1(\boldsymbol{Z}\boldsymbol{W}_1^\top)\boldsymbol{W}_2^\top)\cdots \boldsymbol{W}_N^\top)$, f is permutation-equivariant in the backward propagation, i.e., the weights $\boldsymbol{W}$ in f are associated with those of $f_{(P)}$ by $\boldsymbol{W}_{(P)} = \boldsymbol{P}_C^{-1}\boldsymbol{W}\boldsymbol{P}_C$.*

To prove Thm. 4.7, we first prove the following lemma:

Lemma 4.8. *If f is composed by permutation-equivariant operators as in Thm. 4.7, the derivative of f with respect to feature $\boldsymbol{Z}$ in the natural (Eq. 1) and in the permuted (Eq. 8) settings are correlated by*

$$\frac{\partial l}{\partial \boldsymbol{Z}_{(P)}} = \boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{Z}}\boldsymbol{P}_C. \tag{9}$$

Proof. It is obvious that in forwarding, the combination of permutation-equivariant operators remains to be permutation-equivariant since one can perform induction on the simple case of $f_{1(P)}(f_{2(P)}(\boldsymbol{P}_R\boldsymbol{Z}\boldsymbol{P}_C)) = f_{1(P)}(\boldsymbol{P}_R f_2(\boldsymbol{Z})\boldsymbol{P}_C) = \boldsymbol{P}_R f_1(f_2(\boldsymbol{Z}))\boldsymbol{P}_C$. Hence through all layers in f, it holds that

$$f_{(P)}(\boldsymbol{Z}_{(P)}) = \boldsymbol{P}_R f(\boldsymbol{Z})\boldsymbol{P}_C. \tag{10}$$

By Eq. 8 and Eq. 10, the forward propagation feeds $f(\boldsymbol{Z})$ into F_2, which is equivalent to the vanilla forwarding, and hence the losses are the same.

Now consider the backward propagation. As the loss l is invariant by permutation, we differentiate l through $\boldsymbol{Z}_{i(P)}$ which is the intermediate output of layer i of $f_{(P)}$. By the forward permutation equivariance, we know that it is associated with $\boldsymbol{Z}_i$, the output of layer i at f, as $\boldsymbol{Z}_{i(P)} = \boldsymbol{P}_R\boldsymbol{Z}_i\boldsymbol{P}_C$. Hence we have $\mathrm{d}\boldsymbol{Z}_{i(P)} = \boldsymbol{P}_R\mathrm{d}\boldsymbol{Z}_i\boldsymbol{P}_C$. Therefore, the derivative of l with respect to $\boldsymbol{Z}_{i(P)}$ for any layer i

in the permuted setting is

$$\begin{aligned} \mathrm{d}l &\triangleq \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{Z}_{i(P)}}^{\top} \mathrm{d}\boldsymbol{Z}_{i(P)}) = \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{Z}_{i(P)}}^{\top} \boldsymbol{P}_R \mathrm{d}\boldsymbol{Z}_i \boldsymbol{P}_C) \\ &= \mathrm{tr}(\boldsymbol{P}_C \frac{\partial l}{\partial \boldsymbol{Z}_{i(P)}}^{\top} \boldsymbol{P}_R \mathrm{d}\boldsymbol{Z}_i) = \mathrm{tr}((\boldsymbol{P}_R^{\top} \frac{\partial l}{\partial \boldsymbol{Z}_{i(P)}} \boldsymbol{P}_C^{\top})^{\top} \mathrm{d}\boldsymbol{Z}_i). \end{aligned}$$

The last equality suggests $\frac{\partial l}{\partial \boldsymbol{Z}_i} = \boldsymbol{P}_R^{\top} \frac{\partial l}{\partial \boldsymbol{Z}_{i(P)}} \boldsymbol{P}_C^{\top}$ according to Thm. 6 of [14]. Since $\boldsymbol{P}_R^{\top} = \boldsymbol{P}_R^{-1}$ and $\boldsymbol{P}_C^{\top} = \boldsymbol{P}_C^{-1}$, Eq. 9 holds completing the proof. □

Then we prove Thm. 4.7.

Proof. Without causing any confusion, let $\boldsymbol{Z}_i$ denote the output of $f_i(f_{i-1}(\cdots f_1(\boldsymbol{Z}\boldsymbol{W}_1^{\top})\cdots \boldsymbol{W}_{i-1}^{\top})\boldsymbol{W}_i^{\top})$. We differentiate the loss through $\boldsymbol{Z}_i$ by

$$\begin{aligned} \mathrm{d}l &\triangleq \mathrm{tr}((\frac{\partial l}{\partial \boldsymbol{Z}_i})^{\top} \mathrm{d}\boldsymbol{Z}_i) = \mathrm{tr}((\frac{\partial l}{\partial \boldsymbol{Z}_i})^{\top} \mathrm{d}(\boldsymbol{Z}_{i-1} \cdot \boldsymbol{W}_i^{\top})) \\ &= \mathrm{tr}((\frac{\partial l}{\partial \boldsymbol{Z}_i})^{\top} \boldsymbol{Z}_{i-1} \mathrm{d}\boldsymbol{W}_i^{\top}) = \mathrm{tr}((\frac{\partial l}{\partial \boldsymbol{Z}_i}^{\top} \boldsymbol{Z}_{i-1})^{\top} \mathrm{d}\boldsymbol{W}_i). \end{aligned}$$

Hence the gradient of $\boldsymbol{W}_i$ is $\frac{\partial l}{\partial \boldsymbol{W}_i} = \frac{\partial l}{\partial \boldsymbol{Z}_i}^{\top} \boldsymbol{Z}_{i-1}$. Similarly, the gradient of $\boldsymbol{W}_{i(P)}$ is as follows:

$$\begin{aligned} \frac{\partial l}{\partial \boldsymbol{W}_{i(P)}} &= \frac{\partial l}{\partial \boldsymbol{Z}_{i(P)}}^{\top} \boldsymbol{Z}_{i-1(P)} = \boldsymbol{P}_C^{\top} \frac{\partial l}{\partial \boldsymbol{Z}_i}^{\top} \boldsymbol{P}_R^{\top} \boldsymbol{P}_R \boldsymbol{Z}_{i-1} \boldsymbol{P}_C \\ &= \boldsymbol{P}_C^{\top} \frac{\partial l}{\partial \boldsymbol{Z}_i}^{\top} \boldsymbol{Z}_{i-1} \boldsymbol{P}_C = \boldsymbol{P}_C^{-1} \frac{\partial l}{\partial \boldsymbol{W}_i} \boldsymbol{P}_C. \end{aligned} \tag{11}$$

The second equality holds due to permutation equivalence of $\boldsymbol{Z}_{i-1}$ and Lem. 4.8. Following a similar argument to Corollary 4.6, we have $\boldsymbol{W}_{i(P)} = \boldsymbol{P}_C^{-1} \boldsymbol{W}_i \boldsymbol{P}_C$ for any layer i, given the same randomly initialized weights in the natural and permuted settings. Thereby the backward permutation equivariance holds. Proof completes. □

It should be noted that the pair of arguments — permutation forward and backward equivariance — seem to be circular, i.e., the backward equivariance holds on the condition of the forward equivariance while the latter holds at $\boldsymbol{W}_{(P)} = \boldsymbol{P}_C^{-1} \boldsymbol{W} \boldsymbol{P}_C$. But one can break such a circle by permuting (the inputs and) the initial weights by Eq. 6 in the first place. Then the forward equivariance holds and one can train the model with Eq. 5. Eventually, the backward equivariance can be derived. In the case of training from scratch, it does not matter if the initial weights are permuted since they are all random.

This shows why the permutation-invariance in Set Transformer [17] is a pseudo-invariance. The proposed permutation-invariance head, PMA [17], only achieves forward permutation invariance. However, input of PMA does not follow Lem. 4.8, but rather $\frac{\partial l}{\partial \boldsymbol{Z}_{(P)}} = \frac{\partial l}{\partial \boldsymbol{Z}}$ which destroys the permutation equivariant structure of Eq. 11: instead of getting $\frac{\partial l}{\partial \boldsymbol{W}_{i(P)}} = \frac{\partial l}{\partial \boldsymbol{Z}_i}^{\top} \boldsymbol{P}_R^{\top} \boldsymbol{P}_R \boldsymbol{Z}_{i-1}$, they obtain $\frac{\partial l}{\partial \boldsymbol{W}_{i(P)}} = \frac{\partial l}{\partial \boldsymbol{Z}_i}^{\top} \boldsymbol{P}_R \boldsymbol{Z}_{i-1}$ that the gradients are not aligned anymore.

4.3. Other Components

Notably, the weights in Layer Norm (γ), bias (b) and MLP defined in the form of $\mathrm{MLP}(\boldsymbol{X}) = \sigma(\boldsymbol{X}\boldsymbol{W}_1^{\top})\boldsymbol{W}_2^{\top}$ do not follow Eq. 6 for permutation equivariance. But the intuition is the same: pure token shuffling (row permutation) would not affect these operators; column permutation should be corrected by parameters permutation:

$$\gamma_{(C)} = \gamma \boldsymbol{P}_C, \; b_{(C)} = b \boldsymbol{P}_C, \tag{12}$$

$$\boldsymbol{W}_{1(C)} = \boldsymbol{W}_1 \boldsymbol{P}_C, \; \boldsymbol{W}_{2(C)} = \boldsymbol{P}_C^{-1} \boldsymbol{W}_2. \tag{13}$$

The proof can be found in Appendix 9 and 10. Parameters in F_1, including position embedding, are also not influenced. Proofs can be found in Appendix 10.3.

It is worth noting that the position embedding in $F_1(\cdot)$ is not affected by permutation in both forward and backpropagation since it is added before permutation. The proof in Appendix 10.3 shows that parameters outside of the permutation-inversion scheme are not affected.

5. Experiments

In this section, we provide empirical evidence to support our theoretical findings. As a proof-of-concept, we conduct a series of experiments to demonstrate the potential applications of these properties in real-world scenarios.

5.1. Setup

Our implementation is built on `Pytorch` and `Torchvision`. We validate our theorems by a range of models and tasks including: the `timm` [1] version of pre-trained ViT-Base for image classification, the official version of ViT-Adapter [5] for semantic segmentation (pre-trained with DeiT, using UperNet), huggingface's pre-trained Bert and GPT2 for text classification, and GPT2 for text generation. The image tasks are chosen as 10-label classification on Cifar10 [16] consisting of 60,000 natural images, semantic segmentation on ADE20k [38, 39] containing 25,210 images with pixel-level annotations. For text classification, we use IMDB [23] dataset with 50,000 movie reviews for fine-tuning a sentimental classifier, and a natural language inference dataset SNLI[2] with over 500k sentences. For text generation, we use the GPT2 trained by huggingface on WebText dataset [27] for zero-shot generation on WikiText2 dataset [25].

[1] https://github.com/rwightman/pytorch-image-models

[2] https://nlp.stanford.edu/projects/snli/

Table 2. Test results (%) of ViT-Base for image classification and ViT-Adapter for segmentation. 'Col.' means column and 'P.' stands for permutation.

	ViT-Base	ViT-Adapter		
	Acc	aAc	mIoU	mAcc
Test w/o P.	97.65	83.12	48.75	60.09
Test w/ Row P.	97.65	83.12	48.75	60.09
Test w/ Col. P.	97.65	83.12	48.75	60.09

Table 3. Test accuracy (%) of ViT-Base for image classification and Bert for text classification. Models are trained by Eq. 1 and Eq. 2.'Col.' means column and 'P.' stands for permutation.

	ViT-Base	Bert (Small)
Train w/o P.	97.75	76.4
Train w/ Row P.	97.73	76.5
Train w/ Col. P.	97.94	-

Table 4. Test accuracy (%) of Bert and GPT2 for text classification, and perplexity of GPT2(G) for text generation.

	Bert	GPT2	GPT2(G)
Test w/o P.	94.00	94.03	48.55
Test w/ Col. P.	94.00	94.03	48.55

All Transformer backbones are of base size, characterized by a token dimension of 768, 12 heads, and 12 layers. The evaluation metrics are test accuracy for classification tasks, mIoU, aAcc, and mAcc for semantic segmentation, and perplexity for text generation. More detailed setup can be found in Appendix 11.

5.2. Properties Validation

Row Permutation Forward Equivariance. We validate Thm. 4.1 by performing inference on 'vit_base_patch16_224' pre-trained timm model, the officially released ViT-Adapter, and a small Bert with 2 layers and input of size $(128, 256)$, since the padding mask in huggingface implemented Bert do not follow the properties. The ViT-Base model and the small Bert are fine-tuned on Cifar10 and SNLI respectively before inference. We feed into the same trained model the permuted (Eq. 2) and normal (Eq. 1) features of test data, respectively.

The results for image tasks are displayed in Tab. 2. It is evident that the test results for ViT-Base and ViT-Adapter, both with and without row permutation, are identical up to two decimal places. Similarly, the output accuracies of Bert are closely shown in Tab. 3.

Row Permutation Backward Equivariance. To validate Thm. 4.2, we train the pre-trained ViT-Base and the small Bert by Eq. 1 and Eq. 2, respectively. As shown in Tab. 3, the test results for ViT-Base and Bert with and without row permutation are almost identical, which is consistent with our findings.

Table 5. Test accuracy (%) of Bert and GPT2 for text classification with models trained by Eq. 1 and Eq. 5.

	Bert	GPT2
Train w/o P.	94.00	94.03
Train w/ Col. P.	93.72	93.66

Col. Permutation Forward Equivariance. To validate Thm. 4.4, we test ViT-Base, ViT-Adapter, Bert, and GPT2 on corresponding test datasets. The language models are fine-tuned on the IMDB training set before testing. We first test the model trained by Eq. 1 to get the results without permutation, and then permute the trained model by Eq. 6 to get $\mathrm{T}_{(C)}$ in Eq. 5. The model is reported under the 'Test w/ Col. P.' category in tables.

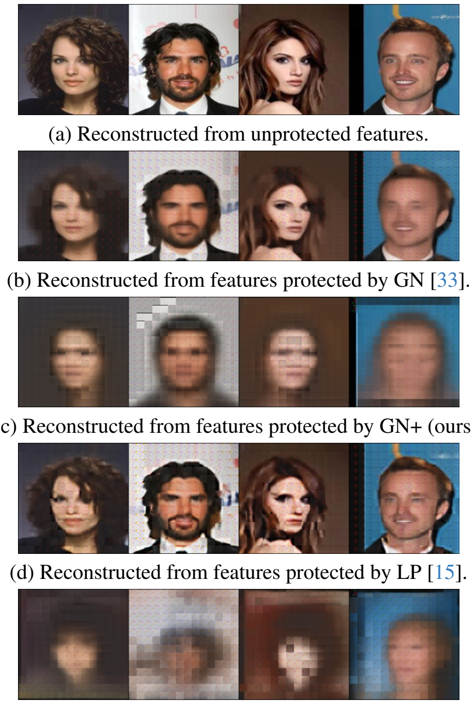
The results are displayed in Tab. 2 and Tab. 4. It is evident that the test results for ViT, Bert, and GPT2, both with and without column permutation, are almost identical. Additionally, if we treat the permuted model $\mathrm{T}_{(C)}$ as a normal model and feed normal inputs, the test accuracy falls to that of random guess, i.e., about 10% on Cifar10 and 50% on IMDB, indicating power similar to 'encryption.' On the text generation task, with or without column permutation, the resulting perplexity remains at 48.55. If we feed the permuted model $\mathrm{T}_{(C)}$ with normal inputs, the perplexity of the output rises to the order of 10^7, akin to a randomly initialized language model.

Col. Permutation Backward Equivariance. To validate Thm. 4.5, we fine-tune pre-trained ViT-Base, Bert, and GPT2 by Eq. 5 with their initial weights permuted by Eq. 6. Note that if the model is trained from scratch, it is unnecessary to permute its initial weights as they are random. As shown in Tab. 3 and Tab. 5, the test results for the three models, trained with and without column permutation, are very close with minor gaps due to randomness. Thus all properties are validated.

5.3. Applications

In this section, we showcase the potential applications of permutation equivariance property in a real-world context as a proof-of-concept. First, we apply the property to enhancing privacy-preserving split learning in [15] and [33]. Following that, we demonstrate how the column permutation of weights can serve as a model 'encryption' or 'authorization' tool, which restricts parties without permutation 'key' from utilizing the model for inference or fine-tuning.

Privacy-Preserving Split Learning. We implement the split learning framework following [15, 33] on a multi-label classification task on CelebA dataset [21], which comprises 2,022,599 images from 10,177 celebrities. In the split learning setting, the server holds the Transformer backbone T



(a) Reconstructed from unprotected features.

(b) Reconstructed from features protected by GN [33].

(c) Reconstructed from features protected by GN+ (ours).

(d) Reconstructed from features protected by LP [15].

(e) Reconstructed from features protected by LP+ (ours).

Figure 3. Reconstruction results of model inversion attacks to features. '+' means the privacy-preserving technique is enhanced by our row permutation.

while the client holds private data, label, the embedding layer F_1, and the task head F_2. F_1 is computed locally on the client as the input data should be kept private from the untrusted server. F_2 is also with the client due to the labels are unknown to the server. The client sends the embeddings to the server and retrieves outputs of T for downstream tasks. The semi-honest server honestly follows the protocol but tries to reconstruct the private data from the embedding by model inversion attack.

We assess the utility of downstream tasks by classification accuracy and how private the client's embeddings are in protecting the inputs. The privacy is expressed by reconstruction metrics including Structural Similarity (SSIM), Peak Signal to Noise Ratio (PSNR) [13], and F-SIM of an MAE [11] inversion model [7, 10, 24]. The worse the reconstruction, the better the privacy protection. For more details of the setup, please see Appendix 11, which aligns with the threat model in [15, 33].

We perform row permutation to enhance the protection level of Gaussian Noise (GN), loss-pass filter (LP) with a radius of 0.05, and Batch Shuffle (BS) techniques. All baselines follow their original implementation. Two row permutation matrices of shape 197 × 197 are applied. The results are presented in Tab. 6. As shown, for GN and LP, the row permutation significantly enhances privacy while maintaining the same level of utility as the baselines. For BS, both the utility and privacy are enhanced by our permutation way. The reconstruction visualization effect under model inversion attack is provided in Fig. 3 which clearly shows considerable privacy improvement.

Table 6. Utility and privacy on CelebA. ↓ means desirable direction. '+' means an enhanced version by our row permutation. Our methods are marked in light gray.

	Utility	Privacy		
	Acc /% ↑	SSIM ↓	PSNR ↓	F-SIM ↓
SL	91.9	0.645	16.25	0.933
GN [15]	88.8	0.457	15.47	0.663
GN+	88.8	0.167	8.068	0.187
LP [15]	90.1	0.450	14.71	0.614
LP+	90.2	0.110	5.687	0.154
BS [33]	90.8	0.148	8.180	0.176
BS+	91.1	0.143	7.613	0.167

To see how permutation affects the trade-off between accuracy and privacy, we report the trade-off curve of LP in Fig. 7 by varying the radius r from 0.01 to 0.05. Typically, with the decrease of the radius, the privacy is enhanced with the cost of utility: as the radius is reduced to 0.01, the SSIM, PSNR, and F-SIM scores decrease to 0.190, 7.950, and 0.196, respectively, while the accuracy drops to 85.6%. With shuffling, one does not need to reduce r to reach a high privacy level. For example, $r = 0.05$ achieves an accuracy of 90.2% under shuffling while reaching privacy no worse than $r = 0.01$ without shuffling. Overall, with row permutation, LP is able to obtain a better trade-off curve for all ranges of radiuses, as shown in Fig. 7, with high privacy levels at all times.

Model Encryption. Model parameter leakage, especially for proprietary large language models, could harm the intellectual property of the model owner. We consider an application that 'encrypts' the trained model weights by a permutation key, i.e., only the party who knows the permutation matrix can correctly use the model. As previously reported in Sec. 5.2, inference on the 'encrypted' model without the column-permutation key is close to random guesses. Further, we show that without the key, fine-tuning the 'encrypted' model is also ineffective.

We compare the training curves of three fine-tuning approaches. As a benchmark, the 'normal' approach fine-tunes the pre-trained models with normal procedures. Second, the 'authorized' approach refers to that the pre-trained model is permuted by Eq. 6 ($\boldsymbol{P}_C$ of shape 768 × 768) to obtain $\mathrm{T}_{(C)}$ before being fine-tuned with permutation (following Eq. 5). The permutation matrix $\boldsymbol{P}_C$ acts as an authorization key in the training process. Third, assume $\mathrm{T}_{(C)}$ is stolen by an unauthorized party who does not own $\boldsymbol{P}_C$. The unauthorized party then tries to fine-tune $\mathrm{T}_{(C)}$ with normal

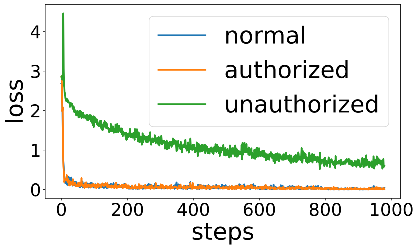


Figure 4. Training curves of fine-tuning ViT. The authorized has a performance close to normal while the unauthorized has a high loss.

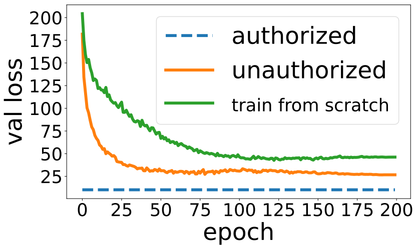


Figure 5. Validation loss curves of ViT trained to convergence. The unauthorized is far worse than the authorized but better than train-from-scratch.

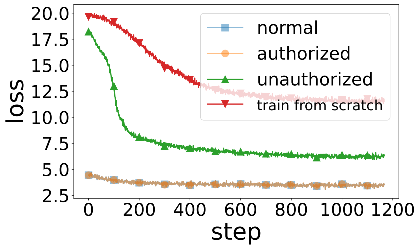


Figure 6. Training curves of fine-tuning GPT2. The unauthorized is far worse than the authorized but better than train-from-scratch.

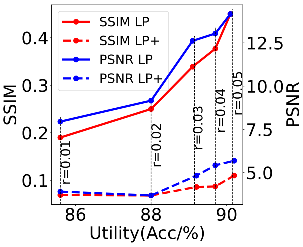


Figure 7. The utility-privacy trade-off curve by varying the radius of LP. The radius is denoted by r in the figure.

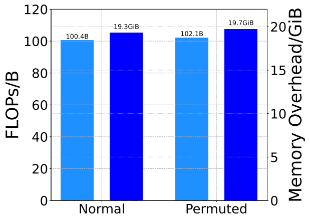


Figure 8. Comparison of computational and memory overhead.

procedures.

The testing accuracy of the fine-tuned models is displayed in Tab. 7 with their training curves until 1000 steps are provided in Fig. 4 for ViT-Base. The figure shows that the unauthorized can hardly learn as much as the authorized. To see if the unauthorized would further improve with a better training strategy, we train both the authorized and unauthorized until full convergence (of the authorized) in Fig. 5. The train-from-scratch model is trained with the same strategy, suggesting the lower bound of the unauthorized: in the worst case, the unauthorized party has to train random weights from the start, indicating no use of the 'encrypted' model.

It is observed that the performance of the unauthorized sits in between the authorized and that of training from scratch. A similar trend is observed in fine-tuning GPT2 on WikiText2 for text generation. The convergence curves are given in Fig. 6 where the authorized behaves almost identically to the normal. The unauthorized has a curve in between the normal and the train-from-scratch. Hence one can conclude that permutation indeed prevents anyone who is unaware of the permutation matrices from taking advantage of a trained model for inference or from releasing the full power of the pre-trained model by fine-tuning.

Table 7. Test accuracy (%) of the normal models fine-tuned in normal procedure, 'encrypted' models fine-tuned in authorized and unauthorized manners.

Model-FineTuning	ViT	Bert	GPT2
Normal-Normal	97.74	94.00	94.03
Encrypted-Authorized	97.94	93.72	93.66
Encrypted-Unauthorized	78.03	77.25	86.16

5.4. Efficiency

To see how much additional overhead our method incurs to vanilla learning, we evaluate the efficiency of our method by floating point operations (FLOPs) of the Transformer-Base backbone, and the memory consumption of training ViT-Base on Cifar10 in the normal and the permuted settings, respectively. For a more precise comparison, exactly four more additional matrix multiplications are applied to the permuted setting: two row permutation by $\boldsymbol{P}_R \in \mathbb{R}^{197\times197}$ and two column permutation by $\boldsymbol{P}_C \in \mathbb{R}^{768\times768}$. $\boldsymbol{P}_R$ is sampled per batch whereas $\boldsymbol{P}_C$ is fixed for the model. Results of Fig. 8 show that our method are almost as efficient as normal learning, incurring negligible overhead in computation and memory consumption.

6. Conclusion

We revise the concept of permutation equivariance and prove it as a property for Transformer-based models. The permutation equivariance of prior works is only a subset of ours: we show inter- and intra- token permutation equivariance in both forward and backward propagation. We not only theoretically analyze the property but also validate it through experiments. Finally, we propose two applications of this property: privacy-enhancing split learning and model authorization. Our research contributes to a deeper understanding of Transformer-based models and their wide potential applications.

References

[1] Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *CVPR*, 2023. 2

[2] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT pre-training of image transformers. In *International Conference on Learning Representations*, 2022. 2, 1

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1, 2, 7

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 2

[5] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*, 2022. 2, 5

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1, 2

[7] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4829–4837, 2016. 7, 8

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 1, 2

[9] Nico Engel, Vasileios Belagiannis, and Klaus Dietmayer. Point transformer. *IEEE access*, 9:134826–134840, 2021. 2

[10] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015. 7, 8

[11] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022. 7

[12] Shota Hirose, Naoki Wada, Jiro Katto, and Heming Sun. Vitgan: Using vision transformer as discriminator with adaptive data augmentation. In *2021 3rd International Conference on Computer Communication and the Internet (ICCCI)*, pages 185–189. IEEE, 2021. 1

[13] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition (ICPR)*, pages 2366–2369. IEEE, 2010. 7

[14] Pili Hu. Matrix calculus: Derivation and simple application. *City University of Hong Kong, Tech. Rep*, 2012. 5

[15] Jonghu Jeong, Minyong Cho, Philipp Benz, Jinwoo Hwang, Jeewook Kim, Seungkwan Lee, and Tae-hoon Kim. Privacy safe representation learning via frequency filtering encoder. In *IJCAI-ECAI Workshop on Artificial Intelligence Safety (AISafety 2022)*, 2022. 1, 6, 7, 8

[16] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5

[17] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019. 1, 2, 3, 5

[18] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019. 2

[19] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *European Conference on Computer Vision*, pages 280–296. Springer, 2022. 2

[20] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 2

[21] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015. 6

[22] Shun Lu, Yu Hu, Peihao Wang, Yan Han, Jianchao Tan, Jixiang Li, Sen Yang, and Ji Liu. Pinat: a permutation invariance augmented transformer for nas predictor. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8957–8965, 2023. 2

[23] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, 2011. Association for Computational Linguistics. 5

[24] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706. IEEE, 2019. 7, 8

[25] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016. 5

[26] Muhammad Muzammal Naseer, Kanchana Ranasinghe, Salman H Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Intriguing properties of vision transformers. *Advances in Neural Information Processing Systems*, 34: 23296–23308, 2021. 1, 2

[27] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 1, 2, 5, 7

[28] Abhishek Singh, Ayush Chopra, Ethan Garza, Emily Zhang, Praneeth Vepakomma, Vivek Sharma, and Ramesh Raskar. Disco: Dynamic and invariant sensitive channel obfuscation for deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12125–12135, 2021. 1

[29] Yujin Tang and David Ha. The sensory neuron as a transformer: Permutation-invariant neural networks for reinforcement learning. *Advances in Neural Information Processing Systems*, 34:22574–22587, 2021. 2

[30] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021. 2

[31] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. In *European Conference on Computer Vision*, pages 516–533. Springer, 2022. 2

[32] 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. 1, 2

[33] Dixi Yao, Liyao Xiang, Hengyuan Xu, Hangyu Ye, and Yingqi Chen. Privacy-preserving split learning via patch shuffling over transformers. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 638–647, 2022. 1, 2, 6, 7, 8

[34] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models, 2022. 1

[35] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 558–567, 2021. 1

[36] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017. 1, 2

[37] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M. Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection, 2022. 2

[38] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017. 5

[39] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127:302–321, 2019. 5