Permutation Equivariance of Transformers and Its Applications

Supplementary Material

7. Structure of Transformer

Transformer-based models are the state-of-the-art deep neural networks and have attracted great attention in both areas of computer vision and natural language processing. Models including transformer encoder blocks as their backbone, such as Bert [6], ViT [8], T2T-ViT [35], ViTGAN [12], BEiT [2] and CoCa [34], have been achieving exceeding performance in a great many tasks.

Transformer encoder blocks, as shown in Fig. 9, mainly contain two critical components: Multi-head Scaled-dot-product self-attention and a feed-forward network (MLP). Inputs are fed in the form of patches, which are usually embedding vectors for words in Bert, or for fractions of images in ViT. The relative position of patches are learned by position embeddings [32], which are injected into the model. Fig. 9 shows the main operators in a Transformer where the shortcut and the linear projection in the Attention block are left out for simplicity.

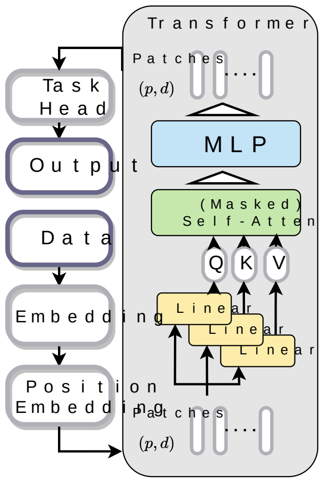

Figure 9. Transformer Encoder Block

The Transformer encoder block is denoted as Enc and the loss is ℓ. The patch embedding of a single input $\boldsymbol{X}$ is expressed as $\boldsymbol{Z}$ of shape (p, d). The first layer in the self-attention contains three parallel linear layers projecting $\boldsymbol{Z}$ to Q, K, V as

$$\boldsymbol{Q} = \boldsymbol{Z}\boldsymbol{W}_Q^\top, \tag{14}$$
$$\boldsymbol{K} = \boldsymbol{Z}\boldsymbol{W}_K^\top, \tag{15}$$
$$\boldsymbol{V} = \boldsymbol{Z}\boldsymbol{W}_V^\top. \tag{16}$$

Q, K, V are fed to the following attention operation

$$\boldsymbol{S} = Softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}}), \tag{17}$$
$$\boldsymbol{A} = \boldsymbol{S}\boldsymbol{V}, \tag{18}$$

where $\boldsymbol{S}$ and $\boldsymbol{A}$ are the softmax output, and the attention output, respectively.

We neglect the attention projection and the residual connection for simplicity. The part following the attention layer is the MLP layer:

$$\boldsymbol{A}_1 = \boldsymbol{A}\boldsymbol{W}_1^\top, \tag{19}$$
$$\boldsymbol{H} = a(\boldsymbol{A}_1), \tag{20}$$
$$\boldsymbol{A}_2 = \boldsymbol{H}\boldsymbol{W}_2^\top \tag{21}$$

where $\boldsymbol{A}_1, \boldsymbol{A}_2$ are the outputs of the linear layers with weights $\boldsymbol{W}_1, \boldsymbol{W}_2$, respectively, and $\boldsymbol{H}$ is the output of the element-wise activation function a which can be ReLu, Tanh, etc.

The backward propagation of Transformer encoder block is as following , we calculate the all the gradients from the final layer back to the first. Gradients are expressed as

$$\begin{aligned}\mathrm{d}l &= \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{A}_2}^\top \mathrm{d}\boldsymbol{A}_2) \\ &= \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{A}_2}^\top (\mathrm{d}\boldsymbol{H})\boldsymbol{W}_2^\top) + \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{A}_2}^\top \boldsymbol{H}\mathrm{d}(\boldsymbol{W}_2^\top)).\end{aligned}$$

The two additive terms are inspected in the following. Let's study $\boldsymbol{H}$ first:

$$\begin{aligned}\mathrm{d}l_1 &\triangleq \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{A}_2}^\top (\mathrm{d}\boldsymbol{H})\boldsymbol{W}_2^\top) \\ &= \mathrm{tr}(\boldsymbol{W}_2^\top \frac{\partial l}{\partial \boldsymbol{A}_2}^\top \mathrm{d}\boldsymbol{H}) \\ &= \mathrm{tr}((\frac{\partial l}{\partial \boldsymbol{A}_2}\boldsymbol{W}_2)^\top \mathrm{d}\boldsymbol{H}),\end{aligned}$$

indicating

$$\frac{\partial l}{\partial \boldsymbol{H}} = \frac{\partial l}{\partial \boldsymbol{A}_2}\boldsymbol{W}_2. \tag{22}$$

For $\boldsymbol{W}_2$,

$$\begin{aligned}\mathrm{d}l_2 &\triangleq \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{A}_2}^\top \boldsymbol{H}\mathrm{d}(\boldsymbol{W}_2^\top)) \\ &= \mathrm{tr}(\mathrm{d}\boldsymbol{W}_2\boldsymbol{H}^\top \frac{\partial l}{\partial \boldsymbol{A}_2}) \\ &= \mathrm{tr}((\frac{\partial l}{\partial \boldsymbol{A}_2}^\top \boldsymbol{H})^\top \mathrm{d}\boldsymbol{W}_2),\end{aligned}$$

and

$$\frac{\partial l}{\partial \boldsymbol{W}_2} = \frac{\partial l}{\partial \boldsymbol{A}_2}^\top \boldsymbol{H}. \tag{23}$$

For $\boldsymbol{A}_1$:

$$\begin{aligned}
\mathrm{d}l_1 &= \mathrm{tr}((\frac{\partial l}{\partial \boldsymbol{H}}^\top \mathrm{d}\boldsymbol{H}) \\
&= \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{H}}^\top \mathrm{d}(a(\boldsymbol{A}_1))) \\
&= \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{H}}^\top a'(\boldsymbol{A}_1) \odot \mathrm{d}\boldsymbol{A}_1)) \\
&= \mathrm{tr}((\frac{\partial l}{\partial \boldsymbol{H}} \odot a'(\boldsymbol{A}_1))^\top \mathrm{d}\boldsymbol{A}_1),
\end{aligned}$$

by Eq. 22, we have

$$\frac{\partial l}{\partial \boldsymbol{A}_1} = \frac{\partial l}{\partial \boldsymbol{A}_2} \boldsymbol{W}_2 \odot a'(\boldsymbol{A}_1). \tag{24}$$

Similarly, we calculate the gradients of $\boldsymbol{A}$ and $\boldsymbol{W}_1$:

$$\frac{\partial l}{\partial \boldsymbol{A}} = \frac{\partial l}{\partial \boldsymbol{A}_1} \boldsymbol{W}_1, \tag{25}$$

$$\frac{\partial l}{\partial \boldsymbol{W}_1} = \frac{\partial l}{\partial \boldsymbol{A}_1}^\top \boldsymbol{A}. \tag{26}$$

In the attention operation:

$$\begin{aligned}
\mathrm{d}l_3 &\triangleq \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{A}}^\top \mathrm{d}\boldsymbol{A}) \\
&= \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{A}}^\top (\mathrm{d}\boldsymbol{S})\boldsymbol{V}) + \mathrm{tr}(\frac{\partial l}{\partial A}^\top \boldsymbol{S}\mathrm{d}\boldsymbol{V}) \\
&= \mathrm{tr}((\frac{\partial l}{\partial \boldsymbol{A}} \boldsymbol{V}^\top)^\top \mathrm{d}\boldsymbol{S}) + \mathrm{tr}((\boldsymbol{S}^\top \frac{\partial l}{\partial \boldsymbol{A}})^\top \mathrm{d}\boldsymbol{V}),
\end{aligned}$$

and

$$\frac{\partial l}{\partial \boldsymbol{S}} = \frac{\partial l}{\partial \boldsymbol{A}} \boldsymbol{V}^\top, \tag{27}$$

$$\frac{\partial l}{\partial \boldsymbol{V}} = \boldsymbol{S}^\top \frac{\partial l}{\partial \boldsymbol{A}}. \tag{28}$$

First, for $\boldsymbol{V} = \boldsymbol{Z}\boldsymbol{W}_V^\top$:

$$\begin{aligned}
\mathrm{d}l_4 &\triangleq \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{V}}^\top \mathrm{d}\boldsymbol{V}) \\
&= \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{V}}^\top (\mathrm{d}\boldsymbol{Z})\boldsymbol{W}_V^\top) + \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{V}}^\top \boldsymbol{Z}\mathrm{d}\boldsymbol{W}_V^\top).
\end{aligned}$$

Similarly, the gradients of $\boldsymbol{Z}$ and $\boldsymbol{W}_V$ are:

$$\frac{\partial l}{\partial \boldsymbol{Z}} = \frac{\partial l}{\partial \boldsymbol{V}} \boldsymbol{W}_V, \tag{29}$$

$$\frac{\partial l}{\partial \boldsymbol{W}_V} = \frac{\partial l}{\partial \boldsymbol{V}}^\top \boldsymbol{Z}. \tag{30}$$

Now we focus on $\boldsymbol{S} = Softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}})$:

$$\begin{aligned}
\mathrm{d}l_5 &\triangleq \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{S}}^\top \mathrm{d}\boldsymbol{S}) \\
&= \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{S}}^\top (diag(\boldsymbol{S}) - \boldsymbol{S}^\top \boldsymbol{S})\mathrm{d}(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}})) \\
&= \mathrm{tr}(((diag(\boldsymbol{S}) - \boldsymbol{S}^\top \boldsymbol{S})^\top \frac{\partial l}{\partial \boldsymbol{S}})^\top \mathrm{d}(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}})),
\end{aligned}$$

and thus

$$\frac{\partial l}{\partial \boldsymbol{Q}} = \frac{1}{\sqrt{d}}((diag(\boldsymbol{S}) - \boldsymbol{S}^\top \boldsymbol{S})^\top \frac{\partial l}{\partial \boldsymbol{S}})\boldsymbol{K}, \tag{31}$$

$$\frac{\partial l}{\partial \boldsymbol{K}} = \frac{1}{\sqrt{d}}((diag(\boldsymbol{S}) - \boldsymbol{S}^\top \boldsymbol{S})^\top \frac{\partial l}{\partial \boldsymbol{S}})^\top \boldsymbol{Q}. \tag{32}$$

And similarly the gradients of $\boldsymbol{W}_Q$ and $\boldsymbol{W}_K$ are:

$$\frac{\partial l}{\partial \boldsymbol{W}_Q} = \frac{\partial l}{\partial \boldsymbol{Q}}^\top \boldsymbol{Z}, \tag{33}$$

$$\frac{\partial l}{\partial \boldsymbol{W}_K} = \frac{\partial l}{\partial \boldsymbol{K}}^\top \boldsymbol{Z}. \tag{34}$$

8. Alg. on Permuted Training

Our permuted training is described by pseudo code in Alg. 1. It should be noted that the permutation takes place not on the dimension of 'batches' but on the rest two dimensions. Taking ViT for example, each image is transformed into a (p, d) matrix representing p patches, and each patch denotes a fraction of the image. Each fraction is embedded into a d-dimensional vector.

We further provide a toy example. Let $\boldsymbol{Z}$ of shape $(3, 4)$ and the row shuffle matrix $\boldsymbol{P}_R$ be

$$\boldsymbol{Z} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix} \quad \boldsymbol{P}_R = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

The row permuted feature is

$$\boldsymbol{P}_R\boldsymbol{Z} = \begin{pmatrix} 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 1 & 2 & 3 & 4 \end{pmatrix}.$$

Column permutation is performed in a similar way. Note that permutation matrices are orthogonal, i.e., $\boldsymbol{P}_R^{-1} = \boldsymbol{P}_R^\top$ and $\boldsymbol{P}_C^{-1} = \boldsymbol{P}_C^\top$.

Algorithm 1 Permuted Training

```
1: Initialization: Initialize the model. Load permutation matrices $\boldsymbol{P}_R, \boldsymbol{P}_C$.
2: Start training
3: repeat
4:     Start a new epoch
5:     repeat
6:         Get a batch of data $\boldsymbol{X}$ from data loader
7:         Get embedding $\boldsymbol{Z}$ of size $(batch\_size, p, d)$.
8:         if using row permutation then
9:             $\boldsymbol{Z}$ = matmul($\boldsymbol{P}_R, \boldsymbol{Z}$)
10:        end if
11:        if using column permutation then
12:            $\boldsymbol{Z}$ = matmul($\boldsymbol{Z}, \boldsymbol{P}_C$)
13:        end if
14:        Send $\boldsymbol{Z}$ to the Transformer Backbone and retrieve the output $\hat{\boldsymbol{Y}}$
15:        if using row permutation then
16:            $\hat{\boldsymbol{Y}}$ = matmul($\boldsymbol{P}_R^{-1}, \hat{\boldsymbol{Y}}$)
17:        end if
18:        if using column permutaton then
19:            $\hat{\boldsymbol{Y}}$ = matmul($\hat{\boldsymbol{Y}}, \boldsymbol{P}_C^{-1}$)
20:        end if
21:        Perform backward propagation
22:    until done all batches
23: until done all epochs
```

9. Permutation-Equivariant Operators

As far as we will show, the following operators are permutation-equivariant:

- Element-wise operators,
- Softmax,
- Linear layer,
- MLP,
- LayerNorm and BatchNorm,
- Attention.

In the following, we will prove the permutation-equivariance of each operator.

Element-wise operators including shortcut, Hadamard product, matrix addition/subtraction and other element-wise functions. We have

Lemma 9.1. *Element-wise operators are permutation-equivariant that*

$$(\boldsymbol{P}_R\boldsymbol{A}\boldsymbol{P}_C) \odot (\boldsymbol{P}_R\boldsymbol{B}\boldsymbol{P}_C) = \boldsymbol{P}_R(\boldsymbol{A} \odot \boldsymbol{B})\boldsymbol{P}_C. \tag{35}$$

where $\odot$ denotes the element-wise operation.

On the left hand-side of the equation, a_{ij} in $\boldsymbol{A}$ and b_{ij} in $\boldsymbol{B}$ are permuted to the same position before being performed the operation. On the right hand-side, a_{ij} and b_{ij} are performed the operation of which the results are permuted. The two are obviously equivariant. Lemma 9.1 also holds for matrix addition and activation function:

$$a(\boldsymbol{P}_R\boldsymbol{A}\boldsymbol{P}_C) = \boldsymbol{P}_R a(\boldsymbol{A})\boldsymbol{P}_C \tag{36}$$

where a is an element-wise activation function, or other element-wise functions like scalar multiplication, division, etc.

Lemma 9.2. *Softmax is permutation-equivariant:*

$$Softmax(\boldsymbol{P}_R\boldsymbol{A}\boldsymbol{P}_C) = \boldsymbol{P}_R Softmax(\boldsymbol{A})\boldsymbol{P}_C. \tag{37}$$

This is because an element is always normalized with the same group of elements, which are not changed in permutations. Thus Softmax is permutation-equivariant.

Lemma 9.3. *Linear layer is permutation-equivariant:*

$$f_{(P)}(\boldsymbol{P}_R\boldsymbol{X}\boldsymbol{P}_C) = \boldsymbol{P}_R f(\boldsymbol{X})\boldsymbol{P}_C \tag{38}$$

where $f(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{W}^\top + b$ and $f_{(P)}(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{W}_{(P)}^\top + b_{(P)}$ and:

$$\boldsymbol{W}_{(P)} = \boldsymbol{P}_C^\top \boldsymbol{W}\boldsymbol{P}_C, \quad b_{(P)} = b\boldsymbol{P}_C.$$

Proof.

$$\begin{aligned} f_{(P)}(\boldsymbol{P}_R\boldsymbol{X}\boldsymbol{P}_C) &= \boldsymbol{P}_R\boldsymbol{X}\boldsymbol{P}_C\boldsymbol{W}_{(P)}^\top + b_{(P)} \\ &= \boldsymbol{P}_R\boldsymbol{X}\boldsymbol{P}_C \cdot \boldsymbol{P}_C^\top\boldsymbol{W}^\top\boldsymbol{P}_C + b\boldsymbol{P}_C \\ &= \boldsymbol{P}_R\boldsymbol{X}\boldsymbol{W}^\top\boldsymbol{P}_C + b\boldsymbol{P}_C \\ &= \boldsymbol{P}_R f(\boldsymbol{X})\boldsymbol{P}_C, \end{aligned}$$

where the bias b is broadcast to each row. Note that if $\boldsymbol{P}_C \neq \boldsymbol{I}$, the identity matrix, this lemma is limited to linear layer with square weight matrix. If $\boldsymbol{P}_C$ is not included, i.e. only row shuffle is used, then all linear layers are row-permutation equivariant. □

Lemma 9.4. *MLP is permutation-equivariant:*

$$f_{(P)}(\boldsymbol{P}_R\boldsymbol{X}\boldsymbol{P}_C) = \boldsymbol{P}_R f(\boldsymbol{X})\boldsymbol{P}_C \tag{39}$$

where

$$\begin{aligned} f(\boldsymbol{X}) &= \sigma(\boldsymbol{X}\boldsymbol{W}_1^\top + b_1)\boldsymbol{W}_2^\top + b_2, \\ f_{(P)}(\boldsymbol{X}) &= \sigma(\boldsymbol{X}\boldsymbol{W}_{1(P)}^\top + b_{1(P)})W_{2(P)}^\top + b_{2(P)}, \end{aligned}$$

and:

$$\begin{aligned} \boldsymbol{W}_{1(P)} &= \boldsymbol{W}_1\boldsymbol{P}_C, \boldsymbol{W}_{2(P)} = \boldsymbol{P}_C^\top\boldsymbol{W}_2, \\ b_{1(P)} &= b_1, b_{2(P)} = b_2\boldsymbol{P}_C, \end{aligned}$$

where σ is the activation function, $\boldsymbol{W}_1 \in \mathbb{R}^{t\times d}$, $\boldsymbol{W}_2 \in \mathbb{R}^{d\times t}$, $b_1 \in \mathbb{R}^t$, $b_2 \in \mathbb{R}^d$, and t is the hidden dimension of MLP.

Proof.

$$\begin{aligned} f_{(P)}(\boldsymbol{P}_R\boldsymbol{X}\boldsymbol{P}_C) &= \sigma(\boldsymbol{P}_R\boldsymbol{X}\boldsymbol{P}_C\boldsymbol{W}_{1(P)}^\top + b_{1(P)})\boldsymbol{W}_{2(P)}^\top + b_{2(P)} \\ &= \sigma(\boldsymbol{P}_R\boldsymbol{X}\boldsymbol{W}_1^\top + b_1)\boldsymbol{W}_2^\top\boldsymbol{P}_C + b_2\boldsymbol{P}_C \\ &= \boldsymbol{P}_R(\sigma(\boldsymbol{X}\boldsymbol{W}_1^\top + b_1)\boldsymbol{W}_2^\top + b_2)\boldsymbol{P}_C \\ &= \boldsymbol{P}_R f(\boldsymbol{X})\boldsymbol{P}_C \end{aligned}$$

where the third equation holds due to Lem. 9.1 and the broadcast of bias. □

Lemma 9.5. *Normalization (LayerNorm for example, LN for short) is permutation-equivariant:*

$$\mathrm{LN}_{(P)}(\boldsymbol{P}_R\boldsymbol{X}\boldsymbol{P}_C) = \mathrm{LN}(\boldsymbol{X}) \tag{40}$$

where $\mathrm{LN}(\boldsymbol{X}) = \frac{\boldsymbol{X}-\mathrm{E}(\boldsymbol{X})}{\sqrt{\mathrm{Var}(\boldsymbol{X})-\epsilon}} * \gamma + b$, *and:*

$$\gamma_{(P)} = \gamma\boldsymbol{P}_C, \quad b = b\boldsymbol{P}_C.$$

Since the same $\mathrm{E}(\boldsymbol{X})$ and $\mathrm{Var}(\boldsymbol{X})$ work on each element, permutation dose not affect the normalization operation. And the affine operation is 'column-wise', weight γ and bias b are broadcast to each row.

Lemma 9.6. *Attention* ($\boldsymbol{A} = Softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}})\boldsymbol{V}$) *is permutation-equivariant:*

$$\begin{aligned} &\mathrm{Attetion}(\boldsymbol{P}_R\boldsymbol{Q}\boldsymbol{P}_C, \boldsymbol{P}_R\boldsymbol{K}\boldsymbol{P}_C, \boldsymbol{P}_R\boldsymbol{V}\boldsymbol{P}_C) \\ &\quad = \boldsymbol{P}_R\mathrm{Attetion}(Q, K, V)\boldsymbol{P}_C. \end{aligned} \tag{41}$$

Proof.

$$\begin{aligned} &\mathrm{Attetion}(\boldsymbol{P}_R\boldsymbol{Q}\boldsymbol{P}_C, \boldsymbol{P}_R\boldsymbol{K}\boldsymbol{P}_C, \boldsymbol{P}_R\boldsymbol{V}\boldsymbol{P}_C) \\ &= Softmax(\frac{\boldsymbol{P}_R\boldsymbol{Q}\boldsymbol{P}_C \cdot \boldsymbol{P}_C^\top\boldsymbol{K}^\top\boldsymbol{P}_R^\top}{\sqrt{d}})\boldsymbol{P}_R\boldsymbol{V}\boldsymbol{P}_C \\ &= \boldsymbol{P}_R Softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}})\boldsymbol{P}_R^\top \cdot \boldsymbol{P}_R\boldsymbol{V}\boldsymbol{P}_C \\ &= \boldsymbol{P}_R\mathrm{Attetion}(Q, K, V)\boldsymbol{P}_C \end{aligned}$$

where the second equality holds because of the permutation equivariance of Softmax. □

Multihead attention is a special case. The validity of Thm. 4.4 and Thm. 4.5 is contingent on constraining the permutation $\boldsymbol{P}_C$ to operate within a single head. It means permutation equivariance holds if permutation is performed within each head, or on different heads, but not across heads. In that case, the feasible permutation space shrinks but the space is still considerable. Take a base Transformer for example, the possible permutations is reduced from 768! to 12! × 64!.

We later provide detailed proofs of Thm. 4.1, Thm. 4.2, Thm. 4.4, Thm. 4.5 specifically for Transformer architecture.

10. Proofs on Transformer Encoder Blocks

We show the detailed proof on the Transformer encoder blocks. The notations are shown in Appendix 7, and Transformer Encoder Block is denoted as Enc for short.

10.1. Enc is Forward Permutation-Equivariant

Enc is forward permutation-equivariant. As proven above, all the basic operators in Enc are permutation-equivariant. The following section shows how the combination of the operators still holds in detail. The proofs of Thm. 4.1 and Thm. 4.4 are organized into one where the weight matrices are permuted by $\boldsymbol{P}_R$ and $\boldsymbol{P}_C$ at the same time. The row permutation equivariance can be seen as a special case where $\boldsymbol{P}_C = \boldsymbol{I}$, and the column permutation equivariance is a special case of $\boldsymbol{P}_R = \boldsymbol{I}$.

Proof. First and foremost, we 'encrypt' all the weight matrices by Eq. 6:

$$\boldsymbol{W}_{i(p)} = \boldsymbol{P}_C^\top\boldsymbol{W}_i\boldsymbol{P}_C,$$

where $\boldsymbol{P}_C$ is the column permutation matrix, $\boldsymbol{W}_i$ is the weight of a normal Enc, and $i \in \{Q, K, V\}$. Weights in MLP are 'encrypted' by $\boldsymbol{W}_{1(C)} = \boldsymbol{W}_1\boldsymbol{P}_C$, $\boldsymbol{W}_{2(C)} = \boldsymbol{P}_C^\top\boldsymbol{W}_2$. We denote the Transformer encoder block with such 'encryption' as $\mathrm{Enc}_{(P)}$.

For $\boldsymbol{Q}$:

$$\begin{aligned} \boldsymbol{Q}_{(P)} &= \boldsymbol{Z}_{(P)}\boldsymbol{W}_{\boldsymbol{Q}(P)}^\top & (42) \\ &= \boldsymbol{P}_R\boldsymbol{Z}\boldsymbol{P}_C \cdot \boldsymbol{P}_C^\top\boldsymbol{W}_{\boldsymbol{Q}}^\top\boldsymbol{P}_C & (43) \\ &= \boldsymbol{P}_R\boldsymbol{Z}\boldsymbol{W}_Q^\top\boldsymbol{P}_C & (44) \\ &= \boldsymbol{P}_R\boldsymbol{Q}\boldsymbol{P}_C. & (45) \end{aligned}$$

Similarly for $\boldsymbol{K}, \boldsymbol{V}$:

$$\begin{aligned} \boldsymbol{K}_{(P)} &= \boldsymbol{P}_R\boldsymbol{K}\boldsymbol{P}_C, & (46) \\ \boldsymbol{V}_{(P)} &= \boldsymbol{P}_R\boldsymbol{V}\boldsymbol{P}_C. & (47) \end{aligned}$$

For $\boldsymbol{S} = Softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}})$:

$$\begin{aligned} \boldsymbol{S}_{(P)} &= Softmax(\frac{\boldsymbol{Q}_{(P)}\boldsymbol{K}_{(P)}^\top}{\sqrt{d}}) & (48) \\ &= Softmax(\frac{\boldsymbol{P}_R\boldsymbol{Q}\boldsymbol{P}_C \cdot \boldsymbol{P}_C^\top\boldsymbol{K}^\top\boldsymbol{P}_R^\top}{\sqrt{d}}) & (49) \\ &= Softmax(\frac{\boldsymbol{P}_R\boldsymbol{Q}\boldsymbol{K}^\top\boldsymbol{P}_R^\top}{\sqrt{d}}) & (50) \\ &= \boldsymbol{P}_R Softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}})\boldsymbol{P}_R^\top & (51) \\ &= \boldsymbol{P}_R\boldsymbol{S}\boldsymbol{P}_R^\top. & (52) \end{aligned}$$

So for $\boldsymbol{A}$:

$$\begin{aligned}\boldsymbol{A}_{(P)} &= \boldsymbol{S}_{(P)}\boldsymbol{V}_{(P)} && (53)\\ &= \boldsymbol{P}_R\boldsymbol{S}\boldsymbol{P}_R^\top \cdot \boldsymbol{P}_R\boldsymbol{V}\boldsymbol{P}_C && (54)\\ &= \boldsymbol{P}_R\boldsymbol{S}\boldsymbol{V}\boldsymbol{P}_C && (55)\\ &= \boldsymbol{P}_R\boldsymbol{A}\boldsymbol{P}_C. && (56)\end{aligned}$$

Following the attention layer, $\boldsymbol{A}$ is fed to the MLP layer:

$$\begin{aligned}\boldsymbol{A}_{1(P)} &= \boldsymbol{A}_{(P)}\boldsymbol{W}_{1(P)}^\top && (57)\\ &= \boldsymbol{P}_R\boldsymbol{A}\boldsymbol{P}_C \cdot \boldsymbol{P}_C^\top\boldsymbol{W}_1 && (58)\\ &= \boldsymbol{P}_R\boldsymbol{A}\boldsymbol{W}_1 && (59)\\ &= \boldsymbol{P}_R\boldsymbol{A}_1. && (60)\end{aligned}$$

Similarly for $\boldsymbol{A}_2$,

$$\boldsymbol{A}_{2(P)} = \boldsymbol{P}_R\boldsymbol{A}_2\boldsymbol{P}_C. \qquad (61)$$

As for the activation in the middle, the element-wise activation function is permutation-equivariant:

$$\boldsymbol{H}_{(P)} = \boldsymbol{P}_R\boldsymbol{H}. \qquad (62)$$

Overall, we have proved Enc satisfies permutation forward equivariance. □

10.2. Enc is Backward Permutation-Invariant

According to Thm. 4.7, since all the operators in Enc are forward permutation-equivariant, the feature of Enc is backward permutation-equivariant and the weight in Enc is permutation-invariant. The following section shows how the combination of the operators still holds in detail. Similar to the proof of forward permutation equivariance, we prove Thm. 4.2 and Thm. 4.5 altogether in one proof where the weight matrices are permuted by $\boldsymbol{P}_R$ and $\boldsymbol{P}_C$ at the same time. The row permutation equivariance can be seen as a special case where $\boldsymbol{P}_C = \boldsymbol{I}$, and the column permutation equivariance is a special case of $\boldsymbol{P}_R = \boldsymbol{I}$.

Proof. Due to the shuffling and unshuffling procedures of Alg. 1, we have the forward and backward propagation outside of the backbone no different from the normal ones. Hence we only focus on the propagation of the Transformer encoder blocks.

We denote $\boldsymbol{A}_{3(P)}$ as the reversed intermediate feature that the down-stream head receives:

$$\boldsymbol{A}_{3(P)} = \boldsymbol{P}_R^\top\boldsymbol{A}_{2(P)}\boldsymbol{P}_C^\top. \qquad (63)$$

Since the feature is unshuffled, we have

$$\boldsymbol{A}_{3(P)} = \boldsymbol{A}_3 = \boldsymbol{A}_2. \qquad (64)$$

First, we focus on the MLP layer:

$$\begin{aligned}\mathrm{d}l &= \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{A}_{3(P)}}^\top \boldsymbol{P}_R^\top \mathrm{d}(\boldsymbol{A}_{2(P)})\boldsymbol{P}_C^\top)\\ &= \mathrm{tr}(\boldsymbol{P}_C^\top\frac{\partial l}{\partial \boldsymbol{A}_{3(P)}}^\top \boldsymbol{P}_R^\top \mathrm{d}\boldsymbol{A}_{2(P)})\\ &= \mathrm{tr}((\boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{A}_{3(P)}}\boldsymbol{P}_C)^\top \mathrm{d}\boldsymbol{A}_{2(P)}),\end{aligned}$$

that is:

$$\frac{\partial l}{\partial \boldsymbol{A}_{2(P)}} = \boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{A}_{3(P)}}\boldsymbol{P}_C = \boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{A}_2}\boldsymbol{P}_C \qquad (65)$$

by Eq. 64.

With $\boldsymbol{H}_{(P)} = \boldsymbol{P}_R\boldsymbol{H}\boldsymbol{P}_C^\top$ and Eq. 23, the gradient:

$$\begin{aligned}\frac{\partial l}{\partial \boldsymbol{W}_{2(P)}} &= \frac{\partial l}{\partial \boldsymbol{A}_{2(P)}}^\top \boldsymbol{H}_{(P)}\\ &= \boldsymbol{P}_C^\top\frac{\partial l}{\partial \boldsymbol{A}_2}^\top \boldsymbol{P}_R^\top \cdot \boldsymbol{P}_R\boldsymbol{H}\\ &= \boldsymbol{P}_C^\top\frac{\partial l}{\partial \boldsymbol{A}_2}^\top \boldsymbol{H}\\ &= \boldsymbol{P}_C^\top\frac{\partial l}{\partial \boldsymbol{W}_2},\end{aligned}$$

that is:

$$\frac{\partial l}{\partial \boldsymbol{W}_{2(P)}} = \boldsymbol{P}_C^\top\frac{\partial l}{\partial \boldsymbol{W}_2}. \qquad (66)$$

By Eq. 24 and Eq. 66, we have

$$\begin{aligned}\frac{\partial l}{\partial \boldsymbol{A}_{1(P)}} &= \frac{\partial l}{\partial \boldsymbol{A}_{2(P)}}\boldsymbol{W}_{2(P)} \odot a'(\boldsymbol{A}_{1(P)})\\ &= [\boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{A}_2}\boldsymbol{P}_C \cdot \boldsymbol{P}_C^\top\boldsymbol{W}_2] \odot [\boldsymbol{P}_R a'(\boldsymbol{A}_1)]\\ &= [\boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{A}_2}\boldsymbol{W}_2] \odot [\boldsymbol{P}_R a'(\boldsymbol{A}_1)]\\ &= \boldsymbol{P}_R[\frac{\partial l}{\partial \boldsymbol{A}_2}\boldsymbol{W}_2 \odot a'(\boldsymbol{A}_1)]\\ &= \boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{A}_1},\end{aligned}$$

that is:

$$\frac{\partial l}{\partial \boldsymbol{A}_{1(P)}} = \boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{A}_1}. \qquad (67)$$

The weight $\boldsymbol{W}_{1(P)}$ in the MLP has the following gradient by Eq. 26:

$$\begin{aligned}\frac{\partial l}{\partial \boldsymbol{W}_{1(P)}} &= \frac{\partial l}{\partial \boldsymbol{A}_{1(P)}}^\top \boldsymbol{A}_{(P)}\\ &= \frac{\partial l}{\partial \boldsymbol{A}_1}^\top \boldsymbol{P}_R^\top \cdot \boldsymbol{P}_R\boldsymbol{A}\boldsymbol{P}_C\\ &= \frac{\partial l}{\partial \boldsymbol{W}_1}\boldsymbol{P}_C,\end{aligned}$$

that is:

$$\frac{\partial l}{\partial \boldsymbol{W}_{1(P)}} = \frac{\partial l}{\partial \boldsymbol{W}_1}\boldsymbol{P}_C. \tag{68}$$

And we come to the attention operation, from Eq. 25, we have

$$\begin{aligned}\frac{\partial l}{\partial \boldsymbol{A}_{(P)}} &= \frac{\partial l}{\partial \boldsymbol{A}_{1(P)}}\boldsymbol{W}_{1(P)}\\ &= \boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{A}_1}\boldsymbol{W}_1\boldsymbol{P}_C\\ &= \boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{A}}\boldsymbol{P}_C,\end{aligned}$$

that is:

$$\frac{\partial l}{\partial \boldsymbol{A}_{(P)}} = \boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{A}}\boldsymbol{P}_C. \tag{69}$$

Hence we observe the permutation rules for the gradients of the intermediate-layer outputs vary from the gradients of the weights. As for the gradients of the softmax-layer output, we have

$$\begin{aligned}\frac{\partial l}{\partial \boldsymbol{S}_{(P)}} &= \frac{\partial l}{\partial \boldsymbol{A}_{(P)}}\boldsymbol{V}_{(P)}^\top\\ &= \boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{A}}\boldsymbol{P}_C\cdot\boldsymbol{P}_C^\top\boldsymbol{V}^\top\boldsymbol{P}_R^\top\\ &= \boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{A}}\boldsymbol{V}^\top\boldsymbol{P}_R^\top\\ &= \boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{S}}\boldsymbol{P}_R^\top,\end{aligned}$$

that is:

$$\frac{\partial l}{\partial \boldsymbol{S}_{(P)}} = \boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{S}}\boldsymbol{P}_R^\top. \tag{70}$$

Since $\boldsymbol{S}_{(P)}$ follows Eq. 52, we have the gradients for $\boldsymbol{Q}_{(P)}$ combining with Eq. 70:

$$\begin{aligned}\frac{\partial l}{\partial \boldsymbol{Q}_{(P)}} &= \frac{1}{\sqrt{d}}[(diag(\boldsymbol{S}_{(P)}) - \boldsymbol{S}_{(P)}^\top\boldsymbol{S}_{(P)})\frac{\partial l}{\partial \boldsymbol{S}_{(P)}}]\boldsymbol{K}_{(P)}\\ &= \frac{1}{\sqrt{d}}[(\boldsymbol{P}_R diag(\boldsymbol{S})\boldsymbol{P}_R^\top - \boldsymbol{P}_R\boldsymbol{S}^\top\boldsymbol{P}_R^\top\cdot\boldsymbol{P}_R\boldsymbol{S}\boldsymbol{P}_R^\top)\\ &\quad\cdot\boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{S}}\boldsymbol{P}_R^\top]\boldsymbol{P}_R\boldsymbol{K}\boldsymbol{P}_C^\top\\ &= \frac{1}{\sqrt{d}}[(\boldsymbol{P}_R diag(\boldsymbol{S})\boldsymbol{P}_R^\top - \boldsymbol{P}_R\boldsymbol{S}^\top\boldsymbol{S}\boldsymbol{P}_R^\top)\\ &\quad\cdot\boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{S}}\boldsymbol{P}_R^\top]\boldsymbol{P}_R\boldsymbol{K}\boldsymbol{P}_C^\top\\ &= \frac{1}{\sqrt{d}}[\boldsymbol{P}_R(diag(\boldsymbol{S}) - \boldsymbol{S}^\top\boldsymbol{S})\\ &\quad\cdot\boldsymbol{P}_R^\top\cdot\boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{S}}\boldsymbol{P}_R^\top]\boldsymbol{P}_R\boldsymbol{K}\boldsymbol{P}_C\\ &= \frac{1}{\sqrt{d}}[\boldsymbol{P}_R(diag(\boldsymbol{S}) - \boldsymbol{S}^\top\boldsymbol{S})\frac{\partial l}{\partial \boldsymbol{S}}\boldsymbol{P}_R^\top]\boldsymbol{P}_R\boldsymbol{K}\boldsymbol{P}_C\\ &= \frac{1}{\sqrt{d}}\boldsymbol{P}_R(diag(\boldsymbol{S}) - \boldsymbol{S}^\top\boldsymbol{S})\frac{\partial l}{\partial \boldsymbol{S}}\boldsymbol{P}_R^\top\cdot\boldsymbol{P}_R\boldsymbol{K}\boldsymbol{P}_C\\ &= \boldsymbol{P}_R\frac{1}{\sqrt{d}}(diag(\boldsymbol{S}) - \boldsymbol{S}^\top\boldsymbol{S})\frac{\partial l}{\partial \boldsymbol{S}}\boldsymbol{K}\boldsymbol{P}_C\\ &= \boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{Q}}\boldsymbol{P}_C.\end{aligned}$$

By a similar derivation on $\boldsymbol{K}$ we obtain:

$$\frac{\partial l}{\partial \boldsymbol{K}_{(\boldsymbol{P})}} = \boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{K}}\boldsymbol{P}_C. \tag{71}$$

Following a similar proof to the gradients of $\boldsymbol{W}_{1(P)}$ or $\boldsymbol{W}_{2(P)}$, we could easily derive:

$$\frac{\partial l}{\partial \boldsymbol{W}_{Q(P)}} = \boldsymbol{P}_C^\top\frac{\partial l}{\partial \boldsymbol{W}_Q}\boldsymbol{P}_C, \tag{72}$$

$$\frac{\partial l}{\partial \boldsymbol{W}_{K(P)}} = \boldsymbol{P}_C^\top\frac{\partial l}{\partial \boldsymbol{W}_K}\boldsymbol{P}_C. \tag{73}$$

By Eq. 28, the gradient of $\boldsymbol{V}_{(P)}$ is

$$\begin{aligned}\frac{\partial l}{\partial \boldsymbol{V}_{(P)}} &= \boldsymbol{S}_{(P)}^\top\frac{\partial l}{\partial \boldsymbol{A}_{(P)}}\\ &= \boldsymbol{P}_R\boldsymbol{S}\boldsymbol{P}_R^\top\cdot\boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{A}}\boldsymbol{P}_C\\ &= \boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{V}}\boldsymbol{P}_C,\end{aligned}$$

and thus we have

$$\frac{\partial l}{\partial \boldsymbol{V}_{(P)}} = \boldsymbol{P}_R\frac{\partial l}{\partial \boldsymbol{V}}\boldsymbol{P}_C, \tag{74}$$

$$\frac{\partial l}{\partial \boldsymbol{W}_{V(P)}} = \boldsymbol{P}_C^\top\frac{\partial l}{\partial \boldsymbol{W}_V}\boldsymbol{P}_C. \tag{75}$$

So far, we have proved the rule for the gradient of weight matrices:

$$\frac{\partial l}{\partial \boldsymbol{W}_{i(P)}} = \boldsymbol{P}_C^\top \frac{\partial l}{\partial \boldsymbol{W}_i} \boldsymbol{P}_C, \quad i \in \{Q, K, V\}. \tag{76}$$

$$\frac{\partial l}{\partial \boldsymbol{W}_{1(P)}} = \frac{\partial l}{\partial \boldsymbol{W}_1} \boldsymbol{P}_C, \quad \frac{\partial l}{\partial \boldsymbol{W}_{2(P)}} = \boldsymbol{P}_C^\top \frac{\partial l}{\partial \boldsymbol{W}_2}. \tag{77}$$

$\boldsymbol{W}_{i(P)}$ are the weights of $\mathrm{Enc}_{(P)}$ while $\boldsymbol{W}_i$ are the weights of Enc. By induction, we can reach the conclusion that if a Transformer encoder block is randomly initialized and trained with $\boldsymbol{Z}_{(P)}$, it would eventually learn to become $\mathrm{Enc}_{(P)}$, the weights of which are associated with Enc by Eq. 76 and Eq. 77. The proof of backward equivariance on the linear projection in the attention is omitted as its proof is similar and the conclusion is the same with Eq. 76. Hence we have proved backward permutation in-/equi-variance. □

10.3. Proofs on Embeddings

We show in this section that the parameters of the embedding layer F_1, including the position embeddings, are the same despite Alg. 1 is applied or not.

Theorem 10.1. *The parameters of F_1 trained with or without permutation are the same.*

Proof. We denote the output of F_1 as $\boldsymbol{Z}_0$, and the input of the Transformer backbone as $\boldsymbol{Z}$. In normal setting (Eq. 1), $\boldsymbol{Z}$ equals to $\boldsymbol{Z}_0$, and so do their gradients. In the permuted setting where we use subscript $_{(P)}$ to denote all the varibles, the input to the backbone is the permuted output of $F_{1(P)}$:

$$\boldsymbol{Z}_{(P)} = \boldsymbol{P}_R \boldsymbol{Z}_{0(P)} \boldsymbol{P}_C. \tag{78}$$

To prove the weights of $F_{1(P)}$ is equivalent to those of F_1, we need to prove:

$$\frac{\partial l}{\partial \boldsymbol{Z}_{0(P)}} = \frac{\partial l}{\partial \boldsymbol{Z}_0}. \tag{79}$$

It is clear that

$$\begin{aligned} \mathrm{d}l &= \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{Z}_{(P)}}^\top \mathrm{d}\boldsymbol{Z}_{(P)}) \\ &= \mathrm{tr}(\boldsymbol{P}_C^\top \frac{\partial l}{\partial \boldsymbol{Z}}^\top \boldsymbol{P}_R^\top \boldsymbol{P}_R \mathrm{d}\boldsymbol{Z}_{0(P)} \boldsymbol{P}_C) \\ &= \mathrm{tr}(\boldsymbol{P}_C \boldsymbol{P}_C^\top \frac{\partial l}{\partial \boldsymbol{Z}}^\top \mathrm{d}\boldsymbol{Z}_{0(P)}) \\ &= \mathrm{tr}(\frac{\partial l}{\partial \boldsymbol{Z}}^\top \mathrm{d}\boldsymbol{Z}_{0(P)}), \end{aligned}$$

where the second equality holds by Lemma 4.8. Hence,

$$\frac{\partial l}{\partial \boldsymbol{Z}_{0(P)}} = \frac{\partial l}{\partial \boldsymbol{Z}} = \frac{\partial l}{\partial \boldsymbol{Z}_0}. \tag{80}$$

The invariance of the weights of F_1 can be derived from Eq. 80. □

It is worth noting that the position embeddings are added before the permutation, so Transformer gets the right position information, since Transformer modeling the position by the position embeddings instead of the order of the input.

10.4. Proofs on Masked Attention

Masked attention is an essential component in the Transformer Decoder Block which is the backbone of the generetive language model [3, 27]. The token permutation can hardly pass through the nonlinear effect of the masked attention, but the column permutation cancels out before the mask takes effect. Thus for the masked attention, we apply column permutations only.

Theorem 10.2. *The results of Masked Softmax with or without column permutation are the same:*

$$MS_{(P)} = MS. \tag{81}$$

Proof. For $MS = Softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}})$:

$$\begin{aligned} MS_{(P)} &= Masked - Softmax(\frac{\boldsymbol{Q}_{(P)}\boldsymbol{K}_{(P)}^\top}{\sqrt{d}}) && (82) \\ &= Masked - Softmax(\frac{\boldsymbol{Q}\boldsymbol{P}_C \cdot \boldsymbol{P}_C^\top \boldsymbol{K}^\top}{\sqrt{d}}) && (83) \\ &= Masked - Softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}}) && (84) \\ &= Masked - Softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}}) && (85) \\ &= MS. && (86) \end{aligned}$$

□

11. Detailed Experimental Setup

11.1. Training Setup

In fine-tuning ViT-Base on Cifar10, an Adam optimizer with a fixed learning rate 10^{-4} is used. The model is trained for 5 epochs with a cross-entropy loss. In fine-tuning Bert and GPT2 on IMDB classification, an Adam optimizer with a fixed learning rate 10^{-5} is used. The models are trained for 2 epochs.

For the 'unauthorized' and train-from-scratch setting on Cifar10 of ViT, the default training setting of ViT is used. Random data augmentation and cosine scheduler are added and the model is trained for 200 epochs till convergence.

The text generation in the properties validation experiments is zero-shot learning on huggingface pre-trained GPT2. The model is fine-tuned with Adam optimizer and learning rate 10^{-5} for 1 epoch.

For the CelebA attribute classification task in the 'privacy-preserving split learning' experiments, we adopt `timm` pre-trained model vit_base_patch16_224 (ViT-Base) to transfer to a 40-binary-attribute classification task. A SGD optimizer is used with a cosine scheduler, for which the (initial, final) learning rate are set to $(0.05, 2 \times 10^{-4})$ and $(5 \times 10^{-4}, 2 \times 10^{-6})$ for the classification head and the encoder blocks, respectively.

11.2. Threat Model of Privacy-Preserving Split Learning

The threat model is consist with [15, 33]. We assume the edge holds a private training data set $\mathbb{D}_{train} = \{X, Y\}$, where X are the private data and Y are the private labels. The edge aims at training a model with the assistance of the cloud, yet without exposing the private data. The cloud possesses powerful computing power and is honest-but-curious, meaning it obeys the protocol and performs the learning task accordingly, but is curious about the private data. The edge selects a model and splits it into three parts: F_1, Enc, F_2. F_1, F_2 are parts close to the input layer and the output layer, respectively, which are sufficiently lightweight to deploy on the edge, whereas Enc is the major part run in the cloud.

Referring to the loss function as L_{task} and the local privacy-preserving method as M, the ultimate goal is to jointly train F_1, Enc, F_2 to

$$\underset{F_1, \mathrm{Enc}, F_2}{\text{minimize}} \; L_{task}(F_2(\mathrm{Enc}(F_1(X))), Y), \tag{87}$$

without the edge revealing X, Y to the cloud. Accessing $F_1(X)$ should not permit the cloud to infer about X. The cloud could launch black-box inversion attacks:

Black-box attackers collect the auxiliary data set X_{aux} and the corresponding features under protection mechanism M as $M(F_1(X_{aux}))$, maybe over multiple training rounds. The attacker trains an inversion model G over $(X_{aux}, M(F_1(X_{aux})))$ to invert the raw input from features [7, 10, 24] by

$$\underset{G}{\text{minimize}} \; L_{atk}(G(M(F_1(X_{aux}))), X_{aux}). \tag{88}$$

The loss L_{atk} can be the mean square error (MSE) between the reconstructed input $\tilde{X}_{aux}$ and X_{aux}. At convergence, G works as a decoder to invert features into inputs. We adopt the MAE decoder G with base-size Transformer backbone and a Tanh activation layer, pre-trained on ImageNet, as the model inversion model. We train G with an AdamW optimizer with a learning rate of 10^{-4} for 30 epochs.