Efficient Shapley Values Calculation for Transformer Explainability

Tianli Sun[1], Haonan Chen[2], Yuping Qiu[1], and Cairong Zhao[1](✉)

[1] Tongji University, Shanghai, China
zhaocairong@tongji.edu.cn
[2] Alibaba Group, Hangzhou, China

Abstract. Class activation mapping (CAM) methods have achieved great model explainability performance for CNNs. However, these methods do not perform so well for Transformers, whose architectures are fundamentally different from CNNs. Instead, gradient-weighted attention visualization methods, with effective consideration for the self-attention and skip-connection, achieve very promising explainability for Transformers. These methods compute gradients by back-propagation to achieve class-specific and accurate explainability. In this work, to further increase the accuracy and efficiency in Transformer explainability, we propose a novel method which is both class-specific and gradient-free. The token importance is calculated using Shapley value method, which has a solid base on game theory but is conventionally very computational expensive to use in practice. To calculate the Shapley value accurately and efficiently for each token, we decouple the self-attention from the information flow in Transformers and freeze other unrelated values. In this way, we construct a linear version of Transformer so that the Shapley values can be calculated conveniently. Using Shapley values for explainability, our method not only improves the explainability further but also becomes class-specific without using gradients, surpassing other gradient-based methods in both accuracy and efficiency. Furthermore, we show that explainability methods for CNNs and Transformers can be bridged under the 1st-order Taylor expansion of our method, resulting in (1) a significant explainability improvement for a modified GradCAM method in Transformers and (2) new insights into understanding the existing gradient-based attention visualization methods. Extensive experiments show that our method is superior compared to state-of-the-arts methods. Our code will be made available.

Keywords: Transformer · Explainability · Shapley Values

1 Introduction

In the field of explainable AI, in order to understand the decision process of CNNs, explainability methods are used to generate heatmaps which highlight the most influential area of the input image. These methods can be divided into two categories based on their ability to explain different target classes: class-specific and class-agnostic.

Most explainability methods for CNNs are class-specific, e.g., class activation mapping (CAM) methods ([4, 9, 15, 20]). If there are multiple objects with different classes in the image, heatmaps can be visualized for each class.

In recent years, with the fast development in Transformers ([7, 8, 13, 17]), the explainability in Transformers is becoming an important topic, which will be helpful to understand, debug and eventually trust the models. Because Transformers are significantly different from CNNs, some CAM methods (e.g. GradCAM shown in [6]) are applicable, but make degraded explainability. Therefore, some explainability methods specifically designed for Transformers are proposed [1, 3, 18]. These early methods focus on the self-attention and computation process to generate the heatmap, but many of them are class-agnostic, which means they can only generate heatmap for the predicted class.

Gradient is helpful for class-specificity, as shown in GradCAM and other gradient-based methods for CNNs. Therefore, it can be used in Transformer for class-specific explainability methods. The attribution method [6] is the first to introduce gradient into rollout. It computes the relevance scores [2] based on the Deep Taylor Decomposition principle [12] and then propagates the relevance scores through the self-attention layers with gradients. Similarly, transition attention mapping (TAM) method [19] regards the computation process in Transformer as a Markov Chain and propagates the attention with integrated gradients [16] for visualization. Using gradients, these methods introduce class-specific information to the relevance score for explanation results. However, the computation of gradients relies on back-propagation, which is computational expensive.

In addition to class-specificity, using gradient also helps to improve the explainability performance by a large margin. This can be seen in the comparison between the methods of GradCAM [15] and CAM [20], as well as between the methods of attribution [6] and rollout [1]. As for GradCAM, [15] proves that using gradient in GradCAM generalizes CAM for a wide range of CNN-based architectures, which partially illustrate the reason for its superior performance. However, the utilization of gradient in attribution [6] is not so clearly investigated.

In order to find a class-specific, accurate and efficient solution for Transformer explainability, we propose a novel method to utilize Shapley values [11] to compute the heatmap. Shapley value method is based on classic equations from cooperative game theory to compute explanations of model predictions, and it is class-specific. For any particular prediction, it assigns an importance value to each feature which represents its effect on the model prediction. According to [11], this method is the only additive feature attribution method that satisfies three desired properties for model explainability: local accuracy, missingness and consistency. This ensures the accuracy of Shapley values for model explanation.

Computational efficiency is the biggest challenge in using Shapley values to generate heatmaps. Directly computing Shapley values for Transformer is impractical because for the Shapley value of a token, it computes the difference between two simple explanation models (linear models) output, one is trained on a subset of all tokens with the target token present, and the other is trained with the token absent. All possible subsets of the input tokens are needed, resulting in $2^{|F|}$ differences to be computed. Therefore, we solve this problem in two steps.

First, to calculate the Shapley values for Transformer tokens, we decouple the self-attention flow and freeze the other unrelated values, to construct a linear version of Transformer. Then we sequentially select each token and mask the attention of all the other tokens to get the logits output, which represents the Shapley values. Specifically, in order to freeze unrelated values, we conduct two inferences sequentially in different models. The first inference takes place in the original Transformer (auxiliary model) to record the values that should be frozen, and the second inference is in the Transformer with masked attention (masked model) to calculate Shapley values.

Second, to further lower the time complexity of our method, we compute the Shapley values of each input token within one batch. The direct calculation of such a big batch (196 for ViT-base/16 [8]) has great time and space complexity. Therefore, we propose a fast batch propagation skill which lowers the computational complexity. Specifically, we manage to calculate the batch propagation using only one pair of attention and value matrices in the self-attention layer by permutation.

To the best of our knowledge, our method is the first class-specific explainability method without using gradients in Transformers, which is highly efficient compared to traditional methods relying on back-propagation to compute gradients. Qualitative and quantitative experiments show that our method outperforms states-of-the-arts in both accuracy and efficiency.

2 Related Work

2.1 Explainability for Transformer

Class activation mapping (CAM) methods aim to generate heatmaps with highlighted regions, which indicate the image parts corresponding to the prediction. In this way, the decision-making of CNNs can be explained. One of the most widely used methods, GradCAM [15], is the first to use gradients for class-specific explainability. It utilizes the gradients back-propagated to linearly combine the feature maps.

GradCAM achieves good performance in CNNs, but it has an inferior performance in Transformers [17], because they use self-attention rather than convolution to process images. Naturally, the attention scores in Transformers can be used for better explainability, as the attention on a token is related to its contribution to the prediction. With consideration for other computations in Transformers, including skip-connection and feed forward, attention scores in different layers are combined to visualize a heatmap.

Rollout method [1] tracks the information propagated through the model following the information flow in Transformers, including self-attention and skip-connection. LRP method [2, 3], as a general method for multilayered model, can be used on Transformers, which back-propagates relevance score from the predicated class to the input image. LRP can also be applied only to the last layer for better performance (Partial LRP [6, 18]). The attribution method [6] computes the relevance scores [2] based on the Deep Taylor Decomposition principle [12] and then propagates the relevance scores through the self-attention layers with gradients. Similarly, transition attention mapping (TAM) method [19] regards the computation process in Transformer as a Markov Chain and propagates the attention with integrated gradients [16] for visualization.

2.2 Shapley Values

As an additive feature attribution method, Shapley value method uses an explanation model (linear model) to explain the original complex model. It satisfies three desirable properties for model explanation, which are local accuracy, missingness and consistency. Local accuracy means that when approximating the original model with an explanation model for a specific input x, the two models should have same outputs at least for this specific input. Missingness constrains the features missing in the original input have no impact on the output. Consistency states that if one feature has a larger contribution to the output in a model f compared to another model g, it also has larger attribution.

The exact computation of Shapley values is challenging because of its complexity. For some subset $z \subseteq x$, the attribution of feature i is computed as $f(z) - f(z \backslash i)$. Then the Shapley value of feature i is computed as a weighted average of all possible subsets [11]. For image tasks, because the number of subsets is numerous, Shapley values are too time-consuming to compute in most cases.

3 Method

3.1 Decoupling Self-attention Flow

We start by defining the notations in Transformers. Suppose there are B self-attention layers in a Transformer. Let $x^{(b)}$, $\boldsymbol{Q}^{(b)}$, $\boldsymbol{K}^{(b)}$, $\boldsymbol{V}^{(b)}$ and $\boldsymbol{A}^{(b)}$ represent the input, query, key, value and attention matrix in the b-th self-attention layer, respectively. $\boldsymbol{Q}^{(b)}$, $\boldsymbol{K}^{(b)}$, $\boldsymbol{V}^{(b)} \in \mathbb{R}^{s \times d_h}$, s is input length and d_h is feature dimension. The calculation within the b-th self-attention module is as follows:

$$\boldsymbol{A}^{(b)} = \text{softmax}\left(\frac{\boldsymbol{Q}^{(b)} \cdot \boldsymbol{K}^{(b)T}}{\sqrt{d_h}}\right) \tag{1}$$

$$x_{attn}^{(b)} = \boldsymbol{A}^{(b)} \boldsymbol{V}^{(b)} + x^{(b)} \tag{2}$$

$x_{attn}^{(b)}$ is propagated to the next self-attention layer through a feed forward network (FFN):

$$\begin{aligned} x^{(b+1)} &= FFN\left(x_{attn}^{(b)}\right) + x_{attn}^{(b)} \\ &= FFN\left(x_{attn}^{(b)}\right) + \boldsymbol{A}^{(b)} \boldsymbol{V}^{(b)} + x^{(b)} \end{aligned} \tag{3}$$

In Eq. 3, it can be seen that the output of a Transformer layer, $x^{(b+1)}$, consists of three terms, which are the results of a feed forward module, a self-attention module and a skip-connection, respectively. Therefore, the information flow in Transformer can be separated into three sub-flows: the feed forward flow, the self-attention flow and the skip-connection flow.

According to the researches on attention visualization in Transformers [1, 5, 6, 19], we can make use of the attention matrix $\boldsymbol{A}^{(b)}$ to get explainable heatmaps on model decisions. In our work, we decouple the self-attention flow, and leverage the idea of Shapley value method [11] to get the explainability of Transformers.

We first reformulate Eq. 3 to a linear function with respect to the binary variable $z \in \{0, 1\}^s$:

$$x^{(b+1)}(z) = \sum_{i}^{s} \boldsymbol{A}_i^{(b)} z_i \boldsymbol{V}^{(b)} + \boldsymbol{V}_0^{(b)} \tag{4}$$

where $\boldsymbol{V}_0^{(b)} = \text{FFN}\left(x_{attn}^{(b)}\right) + x^{(b)}$. $\boldsymbol{A}_i^{(b)} \in \mathbb{R}^{s\times s}$ equals $\boldsymbol{A}^{(b)}$ only for the i-th column and zeros in other columns. $z_i = 1$ indicates the presence of attention on the i-th token, and $z_i = 0$ its absence. If all z_i is 1, $\sum_i^s \boldsymbol{A}_i^{(b)} z_i = \sum_i^s \boldsymbol{A}^{(b)}$.

Equation 4 reveals that, if $\text{FFN}\left(x_{attn}^{(b)}\right)$ (feed forward flow), $x^{(b)}$ (skip-connection flow) and $\boldsymbol{V}^{(b)}$ (values in self-attention module) are constant, the output of a self-attention layer, as well as the logits output of a Transformer, are linearly decided by the presence of attention on each token.

According to [11], the Shapley values in a linear model can be calculated using the logits output and the expectations of the input. To calculate the importance of tokens in Eq. 4, if we consider the expectations of each token to be equal, the Shapley values can be calculated using only the logits output. In the last self-attention layer, the importance of the attention of the i-th token with respect to class c is:

$$\phi_i = \omega_c\left(\boldsymbol{A}_i^{(B)} \boldsymbol{V}^{(B)}\right)^T \tag{5}$$

where ω_c represents the weights of fully connected layer f_c with respect to class c.

Consider all self-attention layers, we calculate $\boldsymbol{A}_i^{(b)} \boldsymbol{V}^{(b)}$ in each layer, and propagate through the Transformer to get the importance of the attention of the i-th token. Especially, we use two forward propagation processes (Fig. 1). The first forward propagation takes place in the original Transformer (auxiliary model) to record $\text{FFN}\left(x_{attn}^{(b)}\right)$, $\boldsymbol{V}^{(b)}$ and $\boldsymbol{A}^{(b)}$. The second forward propagation is in the masked model, where we replace $\text{FFN}\left(x_{attn}^{(b)}\right)$ (the feed forward flow) and $\boldsymbol{V}^{(b)}$ in the self-attention flow with the values from the auxiliary model. The values $\boldsymbol{A}^{(b)}$ in self-attention flow is changed to $\boldsymbol{A}_i^{(b)}$ in each layer. The skip-connection flow is untouched so that the value change in every layer can be propagated through the whole network.

3.2 Fast Batch Propagation

To compute the importance for all tokens, according to Eq. 5, we have to make a forward propagation for each token, which is very redundant and time-consuming. For example, in ViT-base/16 [8], there will be 196 forward propagations, which is about 100 times slower than normal gradient-based methods. Alternatively, we can do all the calculations within one batch, whose batch size $N = s$. However, the storage consumption is too large, and the time consumption is still high.

Here we introduce a solution to speed up the batch propagation, with low space complexity. In computation of self-attention, we notice that:

$$\boldsymbol{A}_i \boldsymbol{V} = \boldsymbol{a}_i \boldsymbol{v}_i \tag{6}$$

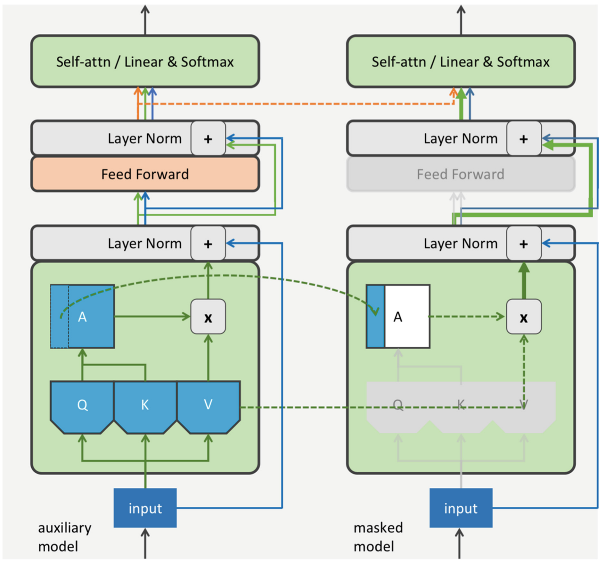

Fig. 1. Overview of our method. Left: auxiliary model which is identical to original model. Right: masked model for importance computation. Information flow from feed forward, self-attention and skip-connection are denoted in orange, green and blue lines, respectively. Dashed lines indicate the values introduced from the auxiliary model.

where $\boldsymbol{a}_i \in \mathbb{R}^{s\times 1}$, $\boldsymbol{v}_i \in \mathbb{R}^{1\times d_h}$ represent the i-th column of $\boldsymbol{A}$ and the i-th row of $\boldsymbol{V}$, respectively. Utilizing Eq. 6, we can compute Eq. 5 for all tokens in a single forward propagation with low cost, by permuting dimensions (1, 3) of $\boldsymbol{A}^{(b)}$ and dimensions (1, 2) of $\boldsymbol{V}^{(b)}$ as follows:

$$\boldsymbol{A}^{(b)} \in \mathbb{R}^{1\times(s\times s)} \rightarrow \boldsymbol{A}'^{(b)} \in \mathbb{R}^{s\times(s\times 1)} \tag{7}$$

$$\boldsymbol{V}^{(b)} \in \mathbb{R}^{1\times(s\times d_h)} \rightarrow \boldsymbol{V}'^{(b)} \in \mathbb{R}^{s\times(1\times d_h)} \tag{8}$$

For a better understanding, we illustrate this permutation in Fig. 2.

Thus, without compute Eq. 5 repeatedly for each token, we can virtually compute the self-attentions for all tokens with the same cost for a single token:

$$\Phi = \omega_c \left(\boldsymbol{A}'^{(B)} \boldsymbol{V}'^{(B)} + \boldsymbol{V}_0'^{(B)}\right)^T \tag{9}$$

where $\Phi = \{\phi_1, \phi_2, \cdots, \phi_B\}$. $\boldsymbol{V}_0'^{(B)} \in \mathbb{R}^{s\times(s\times d_h)}$ is $\boldsymbol{V}_0^{(B)}$ copied s times along batch dimension. We show the calculation of our method in Algorithm 1.

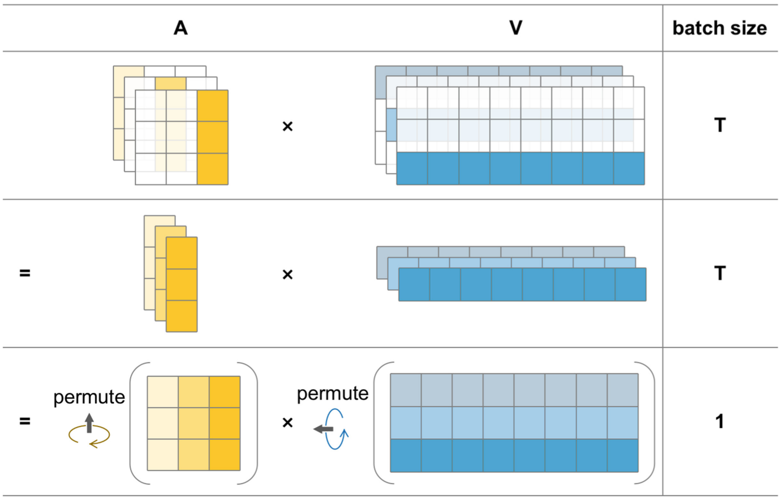

Fig. 2. Permutation for fast batch propagation. Using permutation, the batch size of a forward propagation drops to 1 from T. White cells in the first row depicts the masked tokens, whose values are zero.

```
Algorithm 1 Decoupling Self Attention.
Require: Auxiliary Model $F_a$, Masked Model $F_m$, Input Image $x$
Auxiliary Model
  calculate $y = F_a(x)$
  return $\boldsymbol{A}^{(b)}, \boldsymbol{V}^{(b)}, \mathrm{FFN}^{(b)}$ for $b = 1, 2, \cdots, B$ (Eq. 3)
  permute $\boldsymbol{A}^{(b)} \in \mathbb{R}^{1\times(s\times s)} \longrightarrow \boldsymbol{A}'^{(b)} \in \mathbb{R}^{s\times(s\times 1)}$
  permute $\boldsymbol{V}^{(b)} \in \mathbb{R}^{1\times(s\times d_h)} \longrightarrow \boldsymbol{V}'^{(b)} \in \mathbb{R}^{s\times(1\times d_h)}$
Masked Model
  calculate $\Phi = F_m\big(x, \boldsymbol{A}'^{(1)}, \cdots, \boldsymbol{A}'^{(B)}, \boldsymbol{V}'^{(1)}, \cdots, \boldsymbol{V}'^{(B)}, \mathrm{FFN}'^{(1)}, \cdots, \mathrm{FFN}'^{(B)}\big)$ (Eq. 9)
  return $\Phi \in \mathbb{R}^{s\times 1}$
```

3.3 Bridging Methods of GradCAM and Attribution

Apart from the outstanding explainability performance, our method also shows that the methods of GradCAM and attribution [5, 6] share a similar working principle.

For clarity, suppose an input with length $s = 1$, the self-attention matrix becomes a scalar a. Consider the 2nd-order Taylor expansion for Eq. 5, we have:

$$\phi_i(a) = \phi_i(0) + \phi_i{}'(a)a - \frac{1}{2}\phi_i''(a)a^2 \tag{10}$$

Equation 10 shows a way to approximate the Shapley value of a single token. Applying it to a token series, and only taking the 1st-order part of the expansion, we can get the Shapley value of the i-th token in a way that bridges methods of GradCAM and attribution, which involves the inner product of a self-attention matrix and its gradient:

$$\phi_i(\boldsymbol{A}_i) = \nabla\phi_i(\boldsymbol{A}_i)\boldsymbol{A}_i \tag{11}$$

While in CNNs, GradCAM calculates the inner product of the feature and gradient of the last layer, with a global average pooling (GAP), Eq. 11 shows that in Transformers, GradCAM can be used on the self-attention matrix and its gradient of the last layer, without GAP. Our experiments show that GradCAM without a GAP can largely improve the explainability performance.

Furthermore, utilizing the 2nd-order part of Eq. 10, we can optimize GradCAM:

$$\phi_i(\boldsymbol{A}_i) = \nabla\phi_i(\boldsymbol{A}_i)\boldsymbol{A}_i - \frac{1}{2}\nabla^2\phi_i(\boldsymbol{A}_i)\boldsymbol{A}_i^2 \tag{12}$$

Our experiments show that Eq. 12 further improves the performance of GradCAM. However, the improvement is marginal, and the calculation of $\nabla^2\phi_i$ is too time-consuming.

4 Experiments

4.1 Evaluation Settings

In this section, we conduct experiments to demonstrate that our method can accurately and efficiently explain the prediction of Transformer.

For qualitative evaluation, we compare the explainability heatmaps of our method with others. For quantitative evaluation, we follow [5, 6, 19] and conduct the perturbation and segmentation experiments to show the accuracy of our method.

Perturbation Tests. Perturbation tests [6] consists of positive and negative perturbations on the testing set of ImageNet dataset. In positive perturbation, image patches (tokens) are masked from the highest activations to the lowest in the explainability heatmap, while in the negative version, from the lowest to the highest. In positive perturbation, good explanation results will result in a steep decrease in performance, which indicates that the masked tokens are important to the classification score. In negative perturbation, good explanation results maintain the accuracy of the model, because the removed tokens are not related to the prediction. Therefore, the positive perturbation test with lower AUC means better explainability. In negative test, the higher AUC is better.

Segmentation Tests. Segmentation tests are conducted on ImageNet-Segmentation dataset using the generated heatmaps as soft-segmentation results of the input images. Higher segmentation accuracy represents a better explainability method. We use pixel-accuracy (acc), mean-intersection-over-union (mIoU) and mean-Average-Precision (mAP) as evaluation metrics.

In addition, we also measure the computation speed in segmentation test to show the effectiveness of our method.

4.2 Explainability Results

In this part, we report the performance of our method in visualization, segmentation tests, perturbation tests and computational efficiency. The Transformer we use is ViT-base/16 [8]. CAM-T1 and CAM-T2 are modified GradCAM methods with Eq. 11 and Eq. 12, respectively.

Visualization. To qualitatively evaluate our method, in Fig. 3, we show the visualization results of different explainability methods with respect to the target class. It can be seen that, (1) our method generates more accurate visualizations comparing to others, and (2) CAM-T1 and CAM-T2 visually outperform GradCAM by a large margin.

Segmentation and Perturbation. In order to quantitatively evaluate our method, we perform segmentation and perturbation experiments following the evaluation settings in recent papers on Transformer explainability [6, 19].

In segmentation tests, we utilize the mean value of heatmap as the threshold of segmentation on ImageNet-Segmentation dataset [10]. Pixel-accuracy (acc), mean-intersection-over-union (mIoU) and mean-Average-Precision (mAP) are used to evaluate segmentation performance. Table 1. reports the results, in which our method significantly outperforms all others.

Table 1. Segmentation results on the ImageNet-segmentation dataset (percent).

Method	Pixel acc	mAP	mIOU
GradCAM [15]	65.47	71.37	41.06
CAM-T1	75.61	83.92	56.73
CAM-T2	75.64	83.94	56.76
rollout [1]	73.54	84.76	55.42
partial LRP [18]	76.32	84.67	57.95
attribution [6]	78.96	85.93	61.15
TAM [19]	74.55	85.29	56.66
Ours (last)[1]	78.21	84.82	60.25
Ours	80.25	87.6	63.36

In perturbation tests, we sequentially mask out the tokens of input according to their importance, and compute AUC of the top-1 classification accuracy on ImageNet validation set [14]. In positive perturbation, pixels are masked from the highest importance to the lowest, and a lower AUC means better explainability, while in the negative version, higher AUC is better. Table 2 shows the results in which our method achieves the 1st and 2nd in positive and negative tests, respectively.

Efficiency. In traditional gradient-based methods, back-propagation is the most time-consuming computation. In our method, we manage to be class-specific with two forward propagations instead of using gradients. With the fast batch propagation method as we described in Sect. 3.2, the efficiency of our method greatly outperforms attribution method (Fig. 4). Rollout is the fastest, but it is class-agnostic. GradCAM is faster than our method, but the performance in Transformers is inferior. TAM and CAM-T2 are not shown in the experiment because they are too time-consuming.

[1] Only computes SHAP values for the last layer.

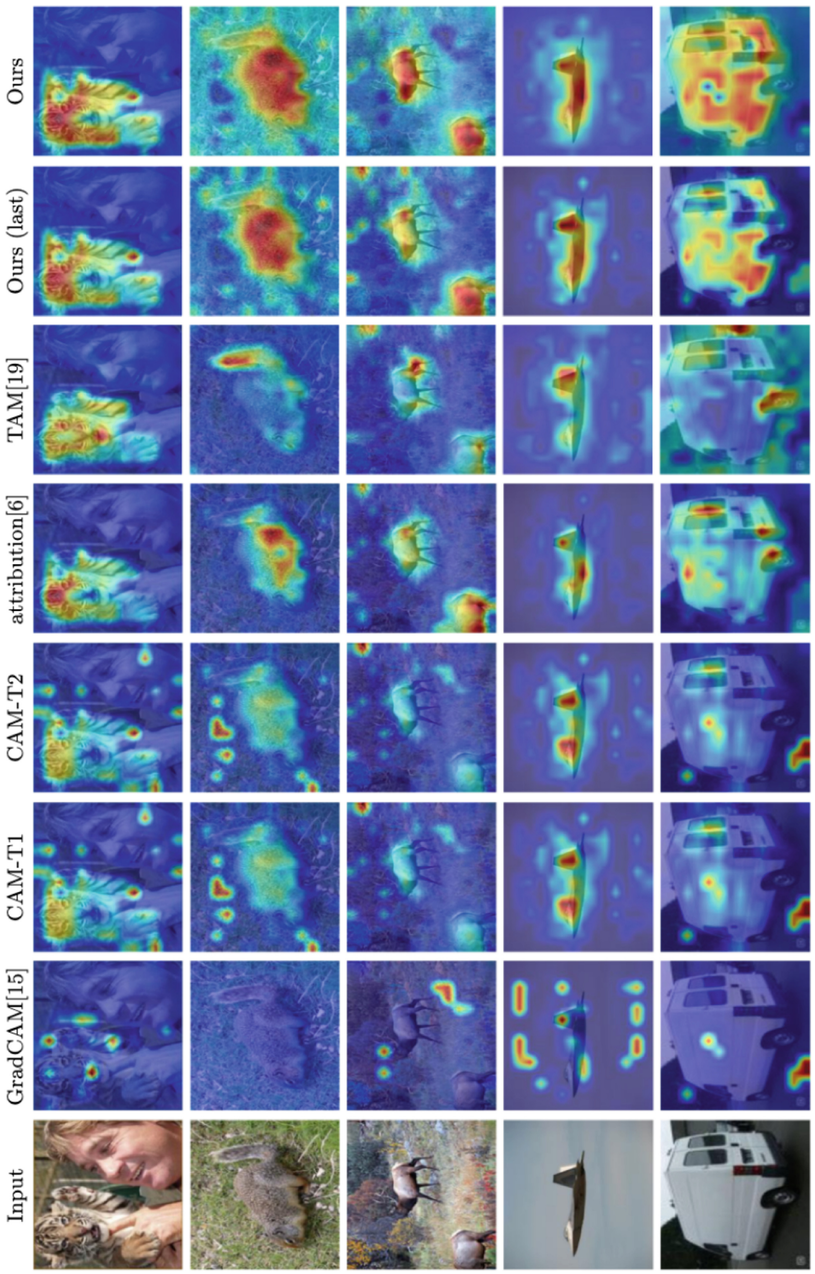

Fig. 3. Visualization results of some SOTA methods. It can be seen that heatmaps from our method are more explainable, and CAM-T1 and CAM-T2 generate better heatmaps than GradCAM.

Table 2. Perturbation AUC results (percent) for the predicted classes on the ImageNet validation set.

Method	positive	negative
GradCAM [15]	34.07	41.52
CAM-T1	19.27	50.85
CAM-T2	19.24	50.87
rollout [1]	20.05	53.11
partial LRP [18]	19.81	50.3
attribution [6]	17.03	54.21
TAM [19]	16.27	55.21
Ours (last)	17.9	52.5
Ours	16.2	55.07

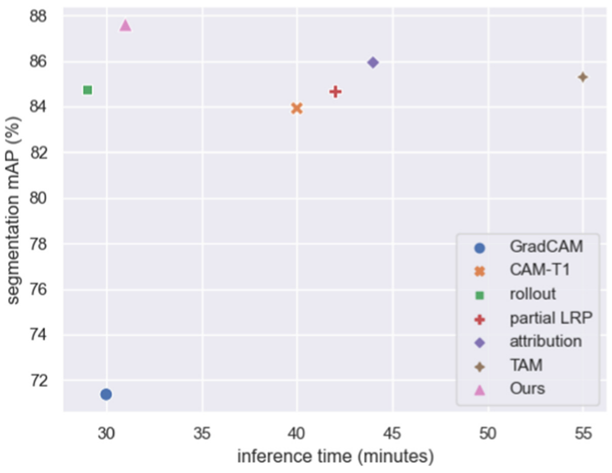

Fig. 4 Speed (minutes) versus mAP (%) in segmentation tests. Rollout is the fastest, but it is class-agnostic. Our method achieves the best accuracy with comparative efficiency.

4.3 Ablation Study

We conduct ablation studies on segmentation and perturbation tests to investigate the influence of different compositions in Transformer's information flow. We start with a naive masking method applied to input. To compute the *i*-th token's importance, we skip it over and mask all the other tokens. The logits output is used to represent the importance.

Next, we turn to mask attention matrix. We conduct the experiment on masking attention only (method A), masking attention and keeping $\boldsymbol{V}$ constant (method A + V), and finally our complete method, masking attention and keep $\boldsymbol{V}$ and feed forward flow constant (method A + V + FF).

Table 3 and 4 show gradual performance improvement on perturbation and segmentation tests from the naive method to our method, which proves that the decoupled self-attention flow is essential for Transformer explainability.

Table 3. Ablation study on perturbation.

Method	positive	negative
Naive	31.85	35.21
A	25.28	39.79
A + V	16.67	54.31
A + V + FF	16.2	55.07

Table 4. Ablation study on segmentation (percent).

Method	Pixel acc	mAP	mIOU
Naive	51.58	57.34	33.31
A	56.07	61.48	37.18
A + V	77.53	84.56	59.65
A + V + FF	80.25	87.6	63.36

5 Conclusion

In this work, we propose a class-specific yet gradient-free heatmap generation method for Transformer explainability, which has an outstanding performance in both the accuracy and efficiency. The contribution of each token is calculated based on the Shapley value method. To efficiently calculate the Shapley values, we decouple the self-attention information flow from the whole model, and turn the Transformer into a linear model. In addition, we use a permutation method to further speed up the calculation. Experiments show that our gradient-free method has a better accuracy and efficiency than other gradient-based methods. Furthermore, we show that explainability methods for CNNs and Transformers can be bridged under the 1st-order Taylor expansion of our method, resulting a significant accuracy improvement of the explainability using a modified GradCAM in Transformers. This also brings some new insights into understanding the existing gradient-based attention visualization methods.

In the future, we will continue to improve our method, and investigate its application for other models. With the accurate segmentation ability of our method, we will also explore its utility in guiding the training of image classification and object detection models.

References

1. Abnar, S., Zuidema, W.H.: Quantifying attention flow in transformers. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, 5–10 July 2020, pp. 4190–4197. Association for Computational Linguistics (2020)
2. Bach, S., Binder, A., Montavon, G., Klauschen, F., M¨uller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one **10**(7), e0130140 (2015)
3. Binder, A., Montavon, G., Lapuschkin, S.: Layer-wise relevance propagation for neural networks with local renormalization layers. In: Villa, A.E.P., Masulli. P. (eds.) ICANN 2016. LNCS, vol. 9887, pp. 63–71. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44781-0_8
4. Chattopadhyay, A., Sarkar, A., Howlader, P., Balasubramanian, V.N.: Grad- cam++: generalized gradient-based visual explanations for deep convolutional networks. In: 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018, Lake Tahoe, NV, USA, 12–15 March 2018, pp. 839–847. IEEE Computer Society (2018)
5. Chefer, H., Gur, S., Wolf, L.: Generic attention-model explainability for interpreting bimodal and encoder-decoder transformers. In: 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, 10- 17 October 2021, pp. 387–396. IEEE (2021)
6. Chefer, H., Gur, S., Wolf, L.: Transformer interpretability beyond attention visualization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 782–791 (2021)
7. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter ofthe Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, 2–7 June 2019, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics (2019)
8. Dosovitskiy, A., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, 3–7 May 2021. OpenReview.net (2021)
9. Fu, R., Hu, Q., Dong, X., Guo, Y., Gao, Y., Li, B.: Axiom-based grad-cam: Towards accurate visualization and explanation of cnns. In: 31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, 7–10 September 2020. BMVA Press (2020)
10. Guillaumin, M., K¨uttel, D., Ferrari, V.: Imagenet auto-annotation with segmentation propagation. Int. J. Comput. Vis. **110**(3), 328–348 (2014)
11. Lundberg, S.M., Lee, S.: A unified approach to interpreting model predictions. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4–9 December 2017, Long Beach, CA, USA, pp. 4765–4774 (2017)
12. Montavon, G., Lapuschkin, S., Binder, A., Samek, W., M¨uller, K.R.: Explaining nonlinear classification decisions with deep, taylor, decomposition. Pattern Recogn. **65**, 211–222 (2017)
13. Radford, A., et al.: Learning transferable visual models from natural language supervision. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18–24 July 2021, Virtual Event. Proceedings of Machine Learning Research, vol. 139, pp. 8748–8763. PMLR (2021)
14. Russakovsky, O., et al.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vis.Comput. Vis. **115**(3), 211–252 (2015)

15. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad- cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 618–626 (2017)
16. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: International Conference on Machine Learning, pp. 3319–3328. PMLR (2017)
17. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
18. Voita, E., Talbot, D., Moiseev, F., Sennrich, R., Titov, I.: Analyzing multi-headself-attention: Specialized heads do the heavy lifting, the rest can be pruned. In: Korhonen, A., Traum, D.R., M'arquez, L. (eds.) Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, 28 July- 2 August, 2019, Volume 1: Long Papers, pp. 5797–5808. Association for Computational Linguistics (2019)
19. Yuan, T., Li, X., Xiong, H., Cao, H., Dou, D.: Explaining information flow inside vision transformers using markov chain. In: Explainable AI Approaches Fordebugging And Diagnosis (2021)
20. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2921–2929 (2016)