

## 1 A Supplementary Materials

2 To provide additional qualitative evidence for the effectiveness of the proposed FlowPrune framework,  
 3 this appendix presents five supplementary examples for both **Paraphrasing Verification** and **Vision**  
 4 **Transformer (ViT) Heatmaps** Analysis in Section ?? . We also included an application scenario  
 5 experiment on **Question Answer Verification** [? ], which is also based on BERT. In addition, we  
 6 provide a detailed description of the dynamic programming algorithm used to determine which  
 7 attention layers to compress in FlowPrune.

### 8 A.1 Paraphrasing Verification

9 To further validate the effectiveness of FlowPrune in interpretability tasks, we conduct additional  
 10 experiments on the MRPC dataset for paraphrasing verification. This task involves determining  
 11 whether two input sentences convey the same meaning. For example, the sentence pair [CLS] *It*  
 12 *affected earnings per share by a penny.* [SEP] *The company said this impacted earnings by a penny a*  
 13 *share.* [SEP] is labeled as a paraphrase (positive example), since both sentences express semantically  
 14 equivalent content.

15 In this setting, we compute attention flow values between token pairs from both sentences using  
 16 FlowPrune, and compare the results with those obtained from the original Attention Flow and  
 17 Attention Rollout algorithms. The analysis is performed using both the pre-trained and fine-tuned  
 18 BERT models. FlowPrune is applied to reduce the computational complexity while preserving  
 19 essential interpretability signals.

20 Figures 1 through 5 show five representative examples of the resulting attention heatmaps. As  
 21 demonstrated, FlowPrune consistently aligns with the original Attention Flow in highlighting key  
 22 semantic alignments and distinguishing features between the sentences. Notably, the differences  
 23 between pre-trained and fine-tuned models are also clearly reflected in the FlowPrune heatmaps,  
 24 showcasing its ability to retain meaningful interpretability insights under compression.

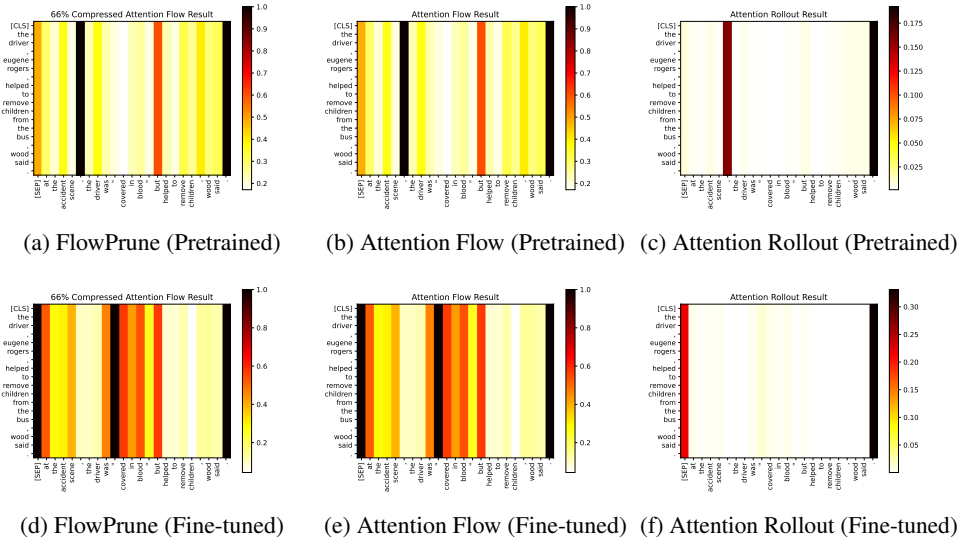


Figure 1: **Paraphrasing Verification Example 1.** The two input sentences are: (1) “The driver , Eugene Rogers , helped to remove children from the bus , Wood said .” and (2) “At the accident scene , the driver was ” covered in blood ” but helped to remove children , Wood said .”

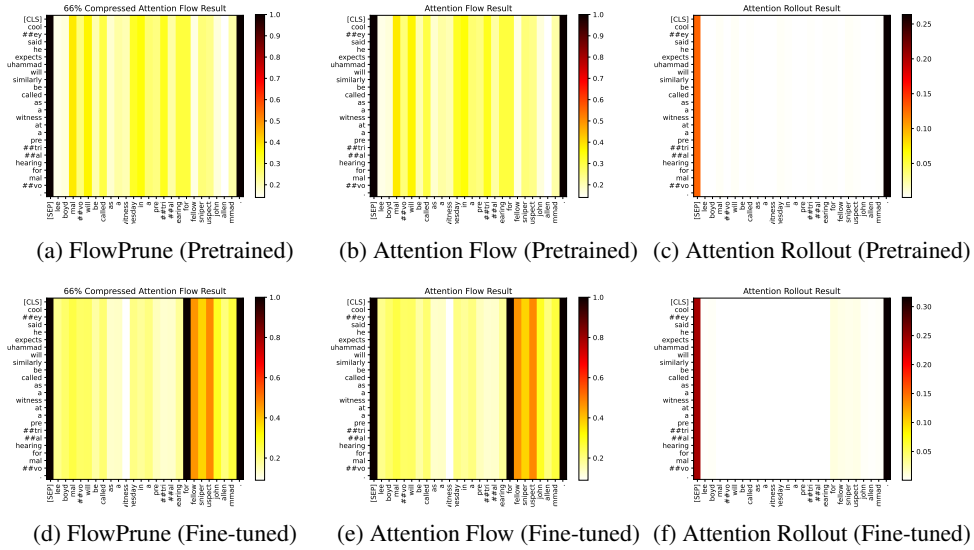


Figure 2: **Paraphrasing Verification Example 2.** The two input sentences are: (1) “Cooley said he expects Muhammad will similarly be called as a witness at a pretrial hearing for Malvo .” and (2) “Lee Boyd Malvo will be called as a witness Wednesday in a pretrial hearing for fellow sniper suspect John Allen Muhammad .”

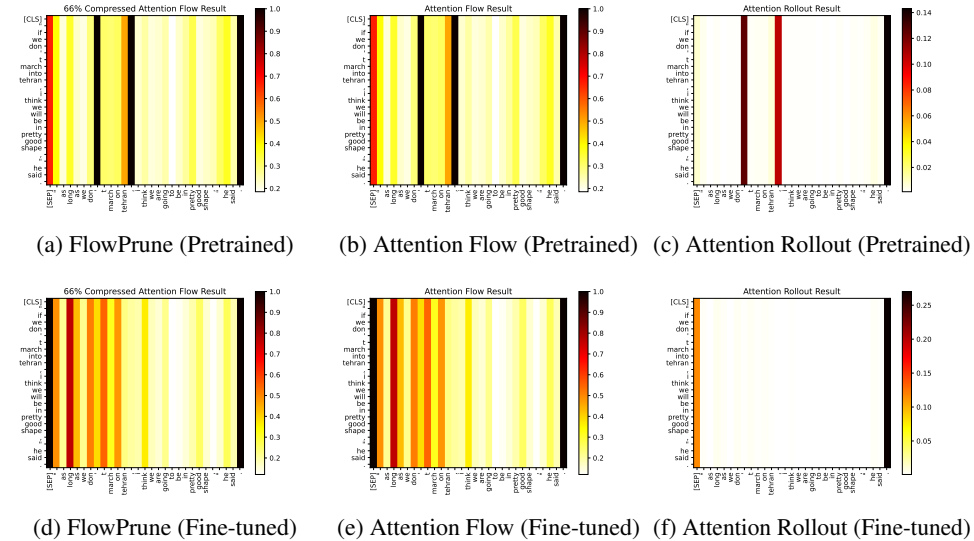


Figure 3: **Paraphrasing Verification Example 3.** The two input sentences are: (1) “ “If we don ’t march into Tehran , I think we will be in pretty good shape , ” he said .” and (2) “ “As long as we don ’t march on Tehran , I think we are going to be in pretty good shape , ” he said .”

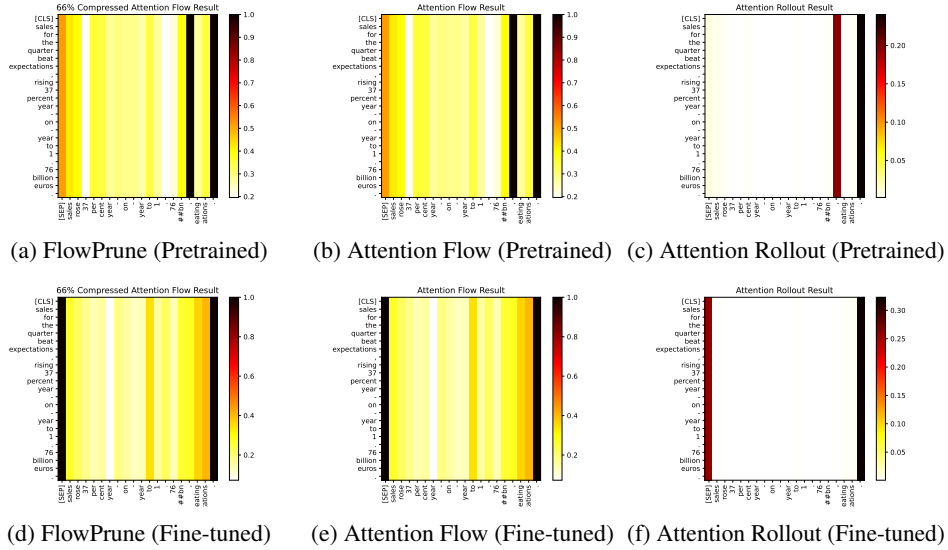


Figure 4: **Paraphrasing Verification Example 4.** The two input sentences are: (1) "Sales for the quarter beat expectations, rising 37 percent year-on-year to 1.76 billion euros." and (2) "Sales rose 37 per cent year-on-year to 1.76bn, beating expectations."

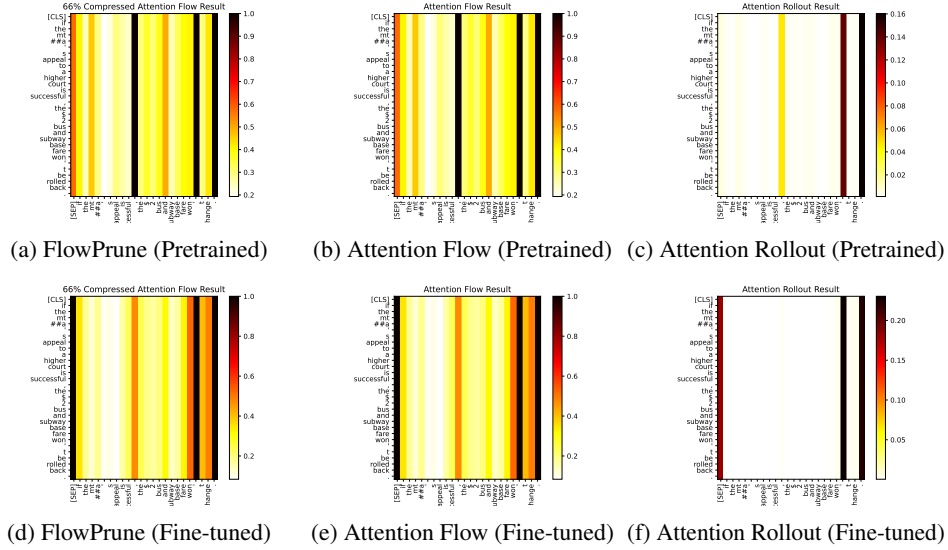


Figure 5: **Paraphrasing Verification Example 5.** The two input sentences are: (1) "If the MTA's appeal to a higher court is successful, the \$2 bus and subway base fare won't be rolled back." and (2) "If the MTA's appeal is successful, the \$2 bus and subway base fare won't change."

## 25 A.2 Vision Transformer (ViT) Heatmaps

26 To further evaluate the utility of FlowPrune in visual interpretability tasks, we provide additional  
 27 examples using the Vision Transformer (ViT) model. Specifically, we apply FlowPrune to the **DeiT-**  
 28 **Small**[?] model on the **ILSVRC2012**[?] image classification dataset. The aim is to generate  
 29 attention heatmaps that reflect the relevance between each input image token and the **[CLS]**  
 30 token, which is responsible for final classification decisions. In this setting, we compare three interpretability  
 31 methods: FlowPrune, the original Attention Flow, and Attention Rollout.

Figures 6 through 10 illustrate five representative examples of attention heatmaps produced by these three methods. As shown in the visualizations, FlowPrune yields results that closely match those of the full Attention Flow, successfully highlighting semantically relevant regions in the image. In contrast, Attention Rollout generates more diffuse and less informative maps, often failing to localize key visual areas.

These results demonstrate that FlowPrune not only retains the interpretability of the original attention mechanism but also achieves substantial computational savings, making it a practical and efficient alternative for ViT visualization.

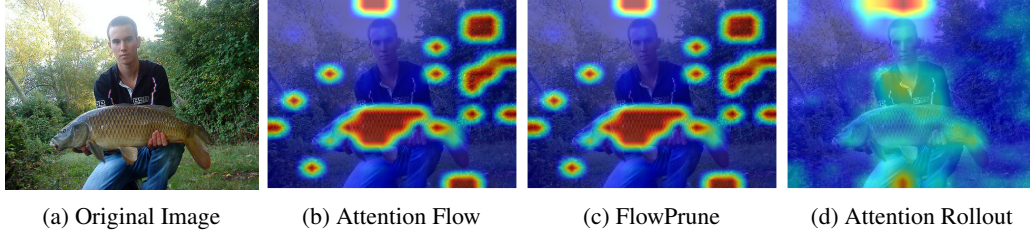


Figure 6: ViT Heapmap Example 1.

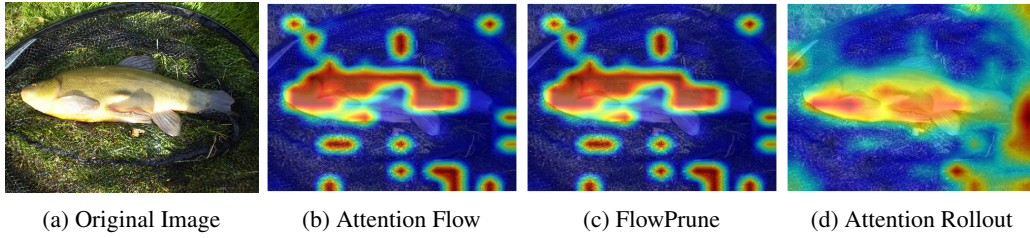


Figure 7: ViT Heapmap Example 2.

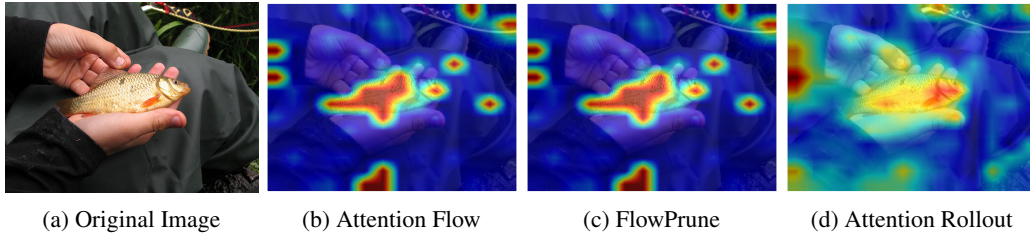


Figure 8: ViT Heapmap Example 3.

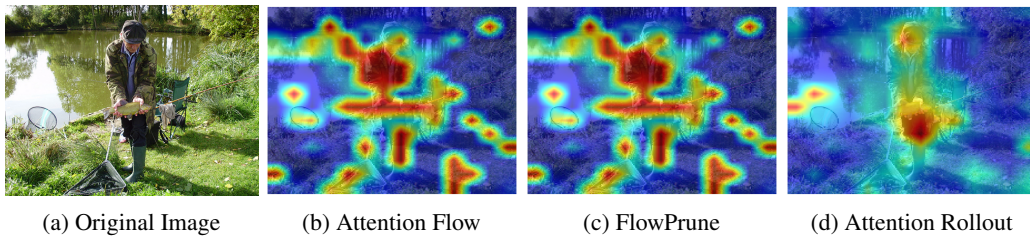


Figure 9: ViT Heapmap Example 4.



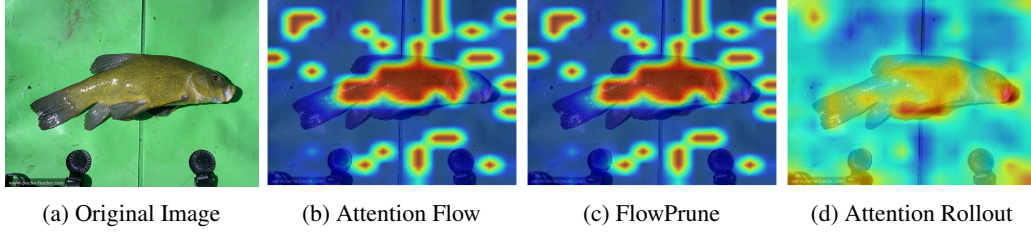


Figure 10: ViT Heapmap Example 5.

### 40 A.3 Question Answer Verification

41 The Question Answer Verification (QAV) experiment aims to investigate how fine-tuning affects  
 42 BERT’s ability to validate whether a given answer correctly addresses a question. Using the QNLI [?  
 43 ] dataset from GLUE, the study fine-tunes a BERT-base model and employs Attention Flows to  
 44 analyze changes in the model’s attention mechanisms. The results show that fine-tuning enables the  
 45 model to better focus on task-relevant details, such as key connecting words and phrases between the  
 46 question and answer, thereby improving its accuracy in determining the validity of the answer.

47 In this experiment, we continue to utilize a 12-layer BERT model. We aim to compare the outcomes  
 48 of FlowPrune and Attention Rollout with those of the original Attention Flow, both in pre-trained  
 49 and fine-tuned models. The primary focus of this task is the attention metrics from tokens in the  
 50 initial layer that point to the *[cls]* token in the final layer. Tokens with the highest computed attention  
 51 metrics are deemed the most significant.

Table 1: **Max- $n$  Overlap for Question Answer Verification.** FP represents Flowprune and AR represents Attention Rollout.(Mean  $\pm$  SE)

Max- $n$	FP Pretrained	AR Pretrained	FP Fine-tuned	AR Fine-tuned
$n = 3$	98.05 $\pm$ 1.11%	64.68 $\pm$ 2.91%	96.10 $\pm$ 1.54%	85.71 $\pm$ 2.59%
$n = 5$	97.31 $\pm$ 1.54%	84.48 $\pm$ 2.14%	94.16 $\pm$ 2.31%	76.31 $\pm$ 2.21%
$n = 10$	97.01 $\pm$ 1.72%	66.50 $\pm$ 1.88%	93.42 $\pm$ 2.60%	71.03 $\pm$ 1.76%

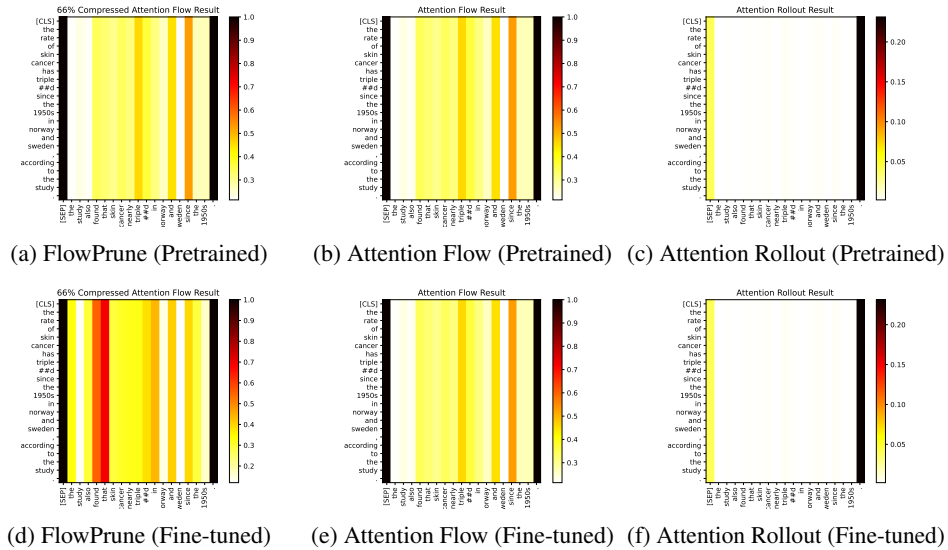


Figure 11: **Question Answer Verification.** Visualization of attention flow results across three methods—FlowPrune, original Attention Flow, and Attention Rollout.

52 We quantify the **Max- $n$  Overlap** between the most salient tokens derived via FlowPrune/Attention  
 53 Rollout and the original Attention Flow. For FlowPrune, a retain rate of 67% is employed, thereby

condensing the original 12-layer attention map into 8 layers. Analogous to the Top- $K$  Token Retention method, the Jaccard overlap (IOU) is utilized as the evaluation metric. The results are presented in Table 1. Additionally, an illustrative example from the MPRC dataset is depicted in Figure 11.

#### A.4 Layer Selection via Dynamic Programming for FlowPrune

To identify which attention layers to compress in the FlowPrune framework, we formulate the selection process as a dynamic programming (DP) problem that aims to **preserve critical regions in the attention flow graph** while reducing computational cost.

As described in the main text, the attention structure of Transformers forms a strictly layered directed acyclic graph (DAG), in which each token in layer  $i$  attends to all tokens in layer  $i + 1$ . This layered property enables layer-wise flow analysis and facilitates compression strategies based on edge removal and reconnection.

To estimate the relative importance of each layer, we apply a sampling-based heuristic inspired by the Max-Flow Min-Cut Theorem. Specifically, we sample multiple source-to-sink paths through the attention graph and treat the lowest-capacity edge on each path as a *potential min-cut edge*. For each intermediate layer  $i$ , we count the number  $a_i$  of such edges it contains. A lower count indicates that the layer contributes less to the overall minimum cut and is thus a better candidate for compression.

Given these layer-wise scores  $\{a_1, a_2, \dots, a_{L-2}\}$ , where the first and last layers are excluded from compression for structural consistency, our goal is to select  $N$  layers to retain (i.e., not compress) such that the total number of potential min-cut edges in the retained layers is maximized.

We define the dynamic programming state as:

- $f(i, j, 0)$ : the maximum total number of potential min-cut edges when the first  $i$  layers are reduced to  $j$  layers, and the  $i$ -th layer is compressed.
- $f(i, j, 1)$ : the same, but the  $i$ -th layer is retained.

The recurrence relations are:

$$\begin{aligned} f(i, j, 0) &= \max(f(i-1, j, 0), f(i-2, j-1, 1)) \\ f(i, j, 1) &= \max(f(i-1, j-1, 0), f(i-1, j-1, 1)) + a_i \end{aligned}$$

The boundary conditions are:

$$\begin{aligned} f(0, 0, 0) &= -\infty, & f(0, 0, 1) &= 0 \\ f(i, j, 0) &= -\infty & \text{if } i \leq j \\ f(i, j, 1) &= -\infty & \text{if } i < j \\ f(i, 1, 0) &= 0 & \text{for } i > 1 \end{aligned}$$

The optimal value is given by:

$$\max(f(L, N, 0), f(L, N, 1))$$

where  $L$  is the total number of intermediate layers (excluding the first and last).

To reconstruct the compression scheme, we backtrack from the optimal DP state and trace the transition path:

- If the current state is  $(l, n, 0)$ , we move to the maximum of  $f(l-1, n, 0)$  or  $f(l-2, n-1, 1)$ , and record that layer  $l$  is compressed.
- If the current state is  $(l, n, 1)$ , we move to the maximum of  $f(l-1, n-1, 0)$  or  $f(l-1, n-1, 1)$ , and record that layer  $l$  is retained.

This dynamic programming approach ensures that layers essential to maintaining attention flow—those with higher concentrations of potential min-cut edges—are preserved. Meanwhile, layers with relatively minor contributions are removed.