FlowPrune: Accelerating Attention Flow Calculation by Pruning Flow Network

Shuo Xu
Southeast University
atmxsp01@gmail.com

Yu Chen
Southeast University
yu_chen@seu.edu.cn

Shuxia Lin
Southeast University
shuxialin@seu.edu.cn

Xin Geng
Southeast University
xgeng@seu.edu.cn

Xu Yang[*]
Southeast University
xuyang_palm@seu.edu.cn

Abstract

The Transformer architecture serves as the foundation of modern AI systems, powering recent advances in Large Language Models (LLMs) and Large Multimodal Models (LMMs). Central to these models, attention mechanisms capture contextual dependencies via token interactions. Beyond inference, attention has been widely adopted for interpretability, offering insights into model behavior. Among interpretability techniques, *attention flow* — which traces global information transfer across layers — provides a more comprehensive perspective than single-layer attention maps. However, computing attention flow is computationally intensive due to the high complexity of max-flow algorithms. To address this challenge, we introduce **FlowPrune**, a novel framework that accelerates attention flow analysis by pruning the attention graph before applying max-flow computations. FlowPrune uses the Max-Flow Min-Cut Theorem and two structural properties of Transformer to identify and eliminate non-critical graph regions. It comprises two components: **Edge Pruning**, which removes insignificant attention edges, and **Layer Compression**, which discards layers with minimal contributions to the flow. We conduct extensive experiments on LLaMA and LLaVA to evaluate the robustness and effectiveness of FlowPrune. Our results show that FlowPrune achieves high agreement with the original attention flow in both absolute and relative error metrics, as well as in identifying influential input tokens. Finally, case studies in both NLP and vision domains demonstrate that FlowPrune produces consistent interpretability outcomes as the original Attention Flow, validating its practical utility. The code for this paper is publicly available.

1 Introduction

The Transformer architecture [1] has become the foundational backbone of modern AI systems, driving major advances in Large Language Models (LLMs) [2–4] and Large Multimodal Models (LMMs) [5–7]. Although various architectural modifications have since been proposed [8, 9], attention remains the core computational primitive, enabling contextual information flow via token interactions. Because attention weights reflect how information is propagated across tokens, they are frequently used as proxies for interpretability. Despite ongoing debate about their faithfulness as explanations [10, 11], attention weights remain valuable tools for interpreting, diagnosing, and analyzing Transformer behavior across diverse applications [12, 13].

[*]Corresponding Author

Compared to single-layer attention maps that capture only local, layer-specific token interactions, attention flow offers a global perspective by tracing how information propagates from input to output across all layers. This makes it a more effective diagnostic tool for understanding model behavior. To quantify attention flow, [14] models the entire network as a Directed Acyclic Graph (DAG), where nodes represent token embeddings and edges denote attention weights. Within this framework, attention flow between input and output tokens is formulated as a maximum flow problem computed by the Highest-Level Preflow Push algorithm [15]. Although this approach captures cross-layer information that single-layer maps overlook, its computational complexity of $O(L^{2.5}N^3)$ — for L layers and N tokens — makes it impractical for modern large-scale models.

To reduce computational complexity, [14] proposes Attention Rollout, which approximates attention flow by recursively multiplying layer-wise attention matrices, reducing complexity to $O(LN^3)$ but introduces significant limitations. Specifically, it neglects global capacity constraints and flow conservation principles inherent to max-flow optimization, potentially distorting information propagation paths [14]. As a result, it risks producing misleading explanations that contradict ground-truth flows — an issue critical to safety-sensitive applications like medical diagnosis or legal reasoning [16, 17].

To address this, we propose **FlowPrune**, a novel method that accelerates attention flow computation without heavily distorting the analyses results. FlowPrune prunes the attention graph before applying max-flow algorithms where the pruning operation is designed based on the Max-Flow Min-Cut Theorem [18], stating that the maximum flow between a source and a sink node equals the total capacity of the minimum cut separating them. This implies that graph regions not contributing to the minimum cut can be safely pruned without affecting the final flow. We leverage two key properties of Transformer-based networks to identify such regions: the attention weights are calculated using the softmax function, and the attention graph is strictly layered.

FlowPrune consists of two components: **Edge Pruning** and **Layer Compression**, both designed to reduce the attention graph while preserving its critical flow structure. First, due to the softmax operation, many attention weights are close to zero [19], contributing little to information propagation. These weights can be safely removed without significantly affecting the overall max-flow value. Second, because the attention graph is strictly layered, we localize minimum cut edges to specific layers, allowing for substantial graph reduction. We use a random sampling heuristic to identify layers with fewer minimum cut edges, which are then removed, and new edges are created between adjacent layers to maintain connectivity. These two procedures significantly compress the attention graph, enabling more efficient max-flow computation.

We validate FlowPrune on two sets of experiments. First, we evaluate the fidelity and efficiency of our graph compression by applying it to two widely used models: the unimodal Llama and the multimodal LLaVA [20, 21]. We compare the max-flow computation on the original and compressed graphs in terms of runtime and absolute/relative error. Since attention flow is often used to assess which input tokens contribute more to model predictions [14], we also measure the ranking discrepancy of input tokens by comparing their relative max-flow orderings. Second, we conduct case studies on real-world scenarios where attention flow has been applied for model analysis. Specifically, we test the practicality of FlowPrune in two real-world scenarios: analyzing Paraphrasing Verification [22] and constructing ViT Heatmaps, where results prove that FlowPrune is more similar with the original Attention Flow than Attention Rollout, validating its practicality. All our code is publicly available, and you can find it at https://github.com/ATMxsp01/FlowPrune.

2 Related Work

Attention Flow and Interpretability. The interpretability of attention mechanisms has attracted significant interest, particularly in NLP and vision models. While some studies suggest that attention weights offer insight into model reasoning [12, 23, 24], others argue that they may not faithfully reflect decision-making processes [10, 11, 25]. Despite this debate, attention remains a widely used tool for attributing model predictions to input elements. For instance, attention-based visualizations have been used to identify important areas in image classification [26, 27], diagnose linguistic errors in translation [28, 29], and trace causal relationships in sentence classification [10, 30]. Beyond direct visualization, recent work has formalized attention flow as a global explanation mechanism [14], tracing information propagation across layers via graph-based analysis. However, such methods are computationally expensive and may become infeasible for large models. Several approximations, including Attention Rollout [14], have been proposed to reduce complexity, though they often sacrifice theoretical properties such as flow conservation. Meanwhile, the observed sparsity of attention matrices [19] motivates efficient pruning-based approaches that preserve interpretive fidelity while improving scalability.

Maximum Flow Algorithms. The maximum flow problem is a classical topic in graph theory, which seeks to determine the maximum amount of flow that can be sent from a source to a sink under capacity constraints. Common algorithms include Edmonds–Karp [31], which uses breadth-first search to find augmenting paths, Dinic's algorithm [32], which improves throughput via level graphs, and Push–Relabel method [33], which maintain a preflow and iteratively update local excess. Among these, the *Highest-Level Preflow-Push* (HLPP) algorithm [15] has demonstrated strong practical efficiency, especially on layered or DAG-structured graphs, making it suitable for attention flow computation [14].

Building on this foundation, our method, **FlowPrune**, accelerates attention flow computation by applying HLPP to a structurally simplified attention graph. We leverage the Max-Flow Min-Cut Theorem [18] to guide pruning decisions, safely removing low-capacity edges and compressible layers that do not contribute to the minimum cut. This design reduces computational complexity while preserving critical attribution paths, enabling scalable analysis of large Transformer models.

3 FlowPrune

To address the scalability limitations of attention flow analysis in large-scale Transformer networks, we propose an efficient approximation method named **FlowPrune**. In this section, we first review the standard Attention Flow formulation and its computational bottlenecks. We then introduce two key approximation strategies, namely **Edge Pruning** and **Layer Compression**, which jointly reduce time complexity while preserving essential attention flow structures to calculate maximum flow.

3.1 Calculating Maximum Flow

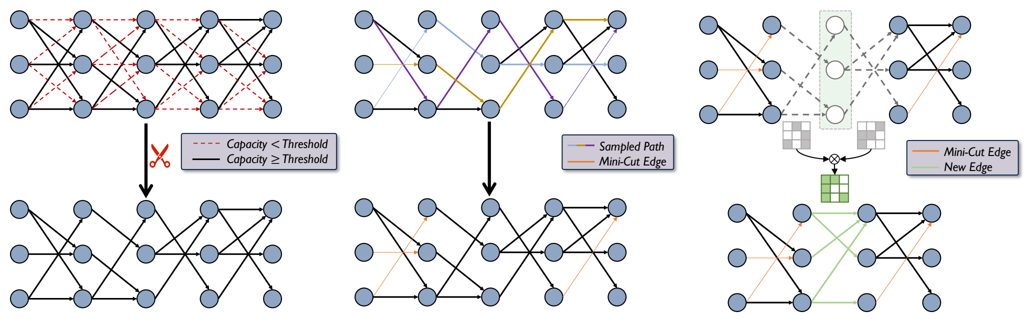
To calculate the maximum flow that quantifies how attention propagates across layers in a L-layer Transformer, [14] models the attention mechanism as a layered directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Specifically, $\mathcal{V} = \{\mathcal{V}_1, \ldots, \mathcal{V}_L\}$, where each $\mathcal{V}_i$ represents the set of nodes corresponding to token representations at the i-th Transformer layer. The edge set is defined as $\mathcal{E} = \{\mathcal{E}_1, \ldots, \mathcal{E}_{L-1}\}$, where $\mathcal{E}_i$ denotes directed edges capturing attention from tokens in layer i to tokens in layer $i + 1$, with normalized attention weights used as edge capacities.

Given this graph $\mathcal{G}$ constructed from an L-layer Transformer, the classical Highest-Level Preflow Push (HLPP) algorithm can be applied to compute the maximum flow, with a time complexity of $O(V^2\sqrt{E})$. Since the number of nodes and edges scale as $V \approx LN$ and $E \approx N^2L$ respectively — where N is the number of tokens per layer and L is the number of layers — the total computational complexity becomes $O(L^{2.5}N^3)$. This high cost makes the method impractical for large-scale Transformer models.

3.2 Pruning the Attention Graph based on Max-Flow Min-Cut Theorem

Our objective is to accelerate the computation of maximum flow in large-scale transformer architectures without largely disturbing the analyses results. To achieve this, we brought ideas from the classic **Max-Flow Min-Cut Theorem** [18] theory, which states that: In a directed graph with non-negative edge capacities, the maximum amount of flow that can be sent from a source node to a sink node equals the total capacity of the minimum cut that separates the source from the sink. This theory provides a useful intuition: if certain regions of the graph do not have the minimum cut, they can be safely removed without significantly changing the overall flow, e.g., by pruning low-capacity edges or compressing intermediate layers without minimum cut.

To apply the above-mentioned insight to prune the attention graph, we also need to exploit two key structural properties of Transformer. First, the computation of attention weights involves a softmax operation and second, Transformer exhibits a strictly layered structure. We next show how to integrate these two properties with Max-Flow Min-Cut Theorem to design **Edge Pruning** and **Layer Compression** strategies for reducing the graph scale.



(a) Edge Pruning. (b) Find Potential Min-cut Edge. (c) Layer Compression.

Figure 1: Pipeline of FlowPrune. (a) The edges with low flow capacity are removed. (b) Three paths are sampled (blue, purple, and brown ones) to find the layers which has less mini-cut edge, where the third layer does not contain min-cut edges. (c) The third layer is removed and the connectivity between the second and fourth layers are re-built by multiplying two binary adjacency matrices.

Edge Pruning. In Transformers, the softmax operation used in attention computation often produces many near-zero attention weights, which contribute minimally to the propagation of information across layers. These negligible weights correspond to edges in the attention graph that carry insignificant flow and can therefore be safely pruned to reduce computational overhead. To eliminate such uninformative edges, as shown in Figure 1a, we set a pruning threshold θ and remove all edges whose capacities fall below this value, effectively treating them as having zero capacity.

Layer Compression. As previously discussed, the attention graph of a Transformer is a strictly layered structure, ensuring all flow paths follow layer-wise transitions. This property offers two key advantages for compression. First, it allows to localize minimum cut edges to specific layers, allowing to treat each layer as a fundamental unit for compression. By removing layers with less critical contributions to the overall attention flow, we can achieve substantial graph reduction while not heavily damaging the maximum flow approximation. Second, the layered structure permits efficient removal of entire intermediate layers and reconstruction of cross-layer connections via matrix multiplication. Next, we show how to identify compressible layers and establish new cross-layer connections.

Motivated by the Max-Flow Min-Cut Theorem, we seek to compress layers containing edges that minimally contribute to the network's minimum cut, as these are less critical for information flow. Since exact minimum cut computation is computationally expensive — equivalent to solving the full max-flow problem [14] — we introduce an efficient sampling-based heuristic, which is shown in Figure 1b. After constructing the attention graph, we randomly sample source-to-sink paths in this graph and identify the lowest-capacity edge in each path as a *potential min-cut edge*. Then we compress layers containing relatively few such edges. Following Transformer architecture principles [13], we preserve the first and last layers to maintain essential signal initiation and termination.

It should be noted that identifying the lowest-capacity edge along a sampled path as a potential min-cut edge is a heuristic rather than a guaranteed criterion. The intuition is as follows: in the case where a source-to-sink path does not overlap with others, the smallest-capacity edge on this path must belong to the minimum cut by definition [18]. However, in more realistic settings where multiple paths often share edges, this assumption becomes less precise. Nevertheless, attention graphs derived from Transformers are densely connected in a strictly layered structure, where each token in layer i typically attends to all tokens in layer $i + 1$. This near-uniform connectivity implies that sampled paths are equally likely to overlap on any edge. As a result, the edge with the lowest capacity on a random path remains statistically more likely to lie on the global min-cut, justifying its use as a proxy in our sampling-based approximation.

After identifying layers for compression, we reconnect the remaining layers to maintain attention flow connectivity. When compressing layer i, we add direct edges from nodes in layer i to layer $i + 2$ whenever a path exists through layer $i + 1$. In practice, this reconnection can be efficiently implemented by multiplying binary adjacency matrices representing layer-wise connectivity, as illustrated in Figure 1c. The new edge capacities are set to 1, which preserves the maximum flow since: (1) all original attention weights are normalized to $[0, 1]$, and (2) flow network theory guarantees that increasing edge capacities beyond actual flow values maintains the maximum flow [34].

Combining edge pruning and layer compression significantly reduces the attention graph size. We then compute the maximum flow using Highest-Level Preflow Push algorithm on this compressed graph, achieving substantial complexity reduction while maintaining flow accuracy.

4 Experiments

We assess the effectiveness of FlowPrune based on computational efficiency, fidelity to the original attention flow, and practical utility in real-world tasks. We evaluate FlowPrune in two cases: (1) generation cases on LLMs and LMMs, and (2) application cases on language and vision tasks. All our experiments were conducted on a single server equipped with 2 AMD EPYC 7453 28-Core Processors and 4 NVIDIA RTX A6000 GPUs.

4.1 Fundamental Evaluation

4.1.1 Experimental Settings and Evaluation Metrics

We evaluate **Llama** [20] and **Qwen** [35], representing language-only transformers, on the other hand **LLaVA** [21] and **QwenVL** [36] representing vision-language transformers, respectively. Each model has 32 layers. We experiment with seven compression configurations: the full model (32 layers), an extreme compression setting (3 layers), and five intermediate retain rates of 80%, 75%, 50%, 30%, and 25%, which correspond to 26, 24, 16, 10, and 8 layers, respectively. The edge pruning threshold is set to 1×10^{-6} throughout all experiments.

The datasets used by these models to generate attention maps are GSM8K [37] and OKVQA [38], respectively. For each model, we provided 1,000 inputs and randomly selected 100 token pairs from each generated attention map (including those compressed using FlowPrune) to calculate the corresponding results.

We propose four fundamental metrics to assess the efficiency and approximation quality. **(1) Computational Cost (Time).** This metric measures the wall-clock time for computing attention flow using FlowPrune, compared to the original implementation, serving as an indicator of computational efficiency. Lower values indicate faster inference. **(2) Approximation Error.** We compute both the absolute and relative errors between the approximated and original attention flow matrices, averaged across all examples. Smaller values reflect higher accuracy in the approximation. **(3) Top-K Token Retention.** For each input, we extract the top-K highest-flow tokens from both the original and approximated attention flows, and compute their Jaccard overlap [39]. Higher values indicate better retention of key tokens. Specifically, $IOU = \frac{|A \cap B|}{|A \cup B|}$, where A/B respectively represents the top-K token pairs of the attention flows calculated by original/compressed graphs, where $K = 1, 3, 5, 10, 15, 20, 25, 50$ in the experiments. **(4) First Rank Divergence.** We identify the first position in the sorted top-K token rankings where the approximation diverges from the original output, normalizing this position by sequence length. Higher values reflect better consistency with the original ranking.

4.1.2 Results Analysis

Computational Cost (Time). Figure 2 presents the speed-up ratios of Llama and LLaVA under varying retain rates. We use 1,000 samples from the **GSM8K** dataset for Llama and 1,000 samples from **OKVQA** for LLaVA. Each sample corresponds to a unique attention graph, and for each graph, attention flows are computed between 1,000 token pairs. The total computation time is used to derive the speed-up ratios.

As previously discussed, for a Transformer with L layers and N tokens per layer, the computational complexity of attention flow computation is $O(L^{2.5}N^3)$. With **Layer Compression**, retaining only L/R layers reduces the complexity to $O(L^{2.5}N^3/R^{2.5})$, achieving a theoretical speed-up of $R^{2.5}$. The empirical results in Figures 2 and 3 generally align with this analysis. For instance, with a 50% retain rate, Llama's computation time is halved-down from 0.55 seconds per case. When compressed to just three layers, FlowPrune requires only 5.31% of the original attention flow time. On LLaVA, FlowPrune achieves even greater gains, reducing computation time to approximately 35% of the original at a 50% retain rate.

Interestingly, the two models exhibit different speed-up patterns. LLaVA's performance closely matches the theoretical expectation, whereas Llama demonstrates a more linear reduction in average computation time. This discrepancy may be attributed to differences in the attention weight distributions across the two models. Llama may retain more edges during Layer Compression, which causes its time cost to decrease more slowly than the theoretical estimate.

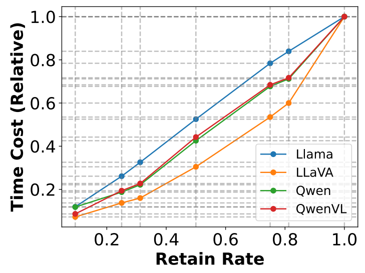
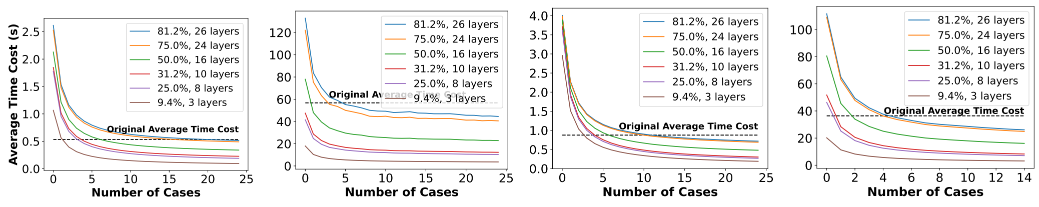


Figure 2: **Speed-up Ratio.** The horizontal axis shows the retain rate (fraction of layers retained), and the vertical axis shows the average time cost normalized by the original Attention Flow runtime, measured over 100 token pairs.

While compressing the attention graph introduces some overhead, this step is performed only once per graph. All subsequent token-pair computations benefit from the reduced graph size. Figure 3 shows the average per-pair computation time — including compression overhead — as the number of token pairs per graph increases. Due to longer input sequences (approximately 500 tokens), LLaVA has a higher average time cost than Llama (typically around 100 tokens). Nonetheless, the compression overhead is amortized as the number of token pairs increases. For a 50% retain rate, the overhead becomes negligible after computing more than 6 cases for Llama and 2 cases for LLaVA.



(a) LLaMA (b) LLaVA (c) Qwen (d) QwenVL

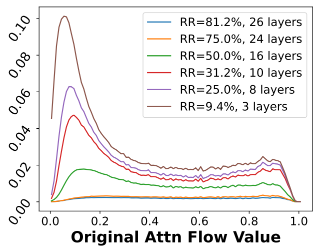
Figure 3: **Comparison of Average Overhead.** The horizontal axis indicates the number of token pairs computed by per attention map. The vertical axis shows the average time cost per token pair, including the overheads. Each curve corresponds to a specific compression setting. The black dashed line denotes the average computational time of the original (uncompressed) Attention Flow.

Approximation Error. Figures 4 provide detailed analyses of how absolute and relative errors vary with the true attention flow values. Since relative error fluctuates with both retain rates and original attention flow values, we present results for all retain rates in Figures 4b and 4e, and separately report results for retain rates of 30% and above in Figures 4c and 4f.

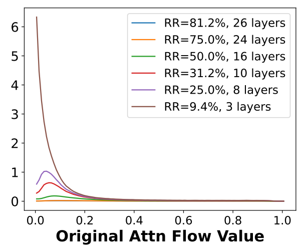
We observe that both absolute and relative errors decrease as the original attention flow approaches 1. This is due to two properties: (1) Since the sum of all edge capacities from a source node in an attention graph is 1, the maximum flow cannot exceed 1. Edge Pruning only reduces edges originating from a node, so this property remains valid for the compressed graph. (2) In Layer Compression, newly added edge capacities are set to 1, which can be regarded as positive infinity in the attention graph, while connectivity between remaining nodes stays unchanged. This ensures that the maximum flow from the compressed graph increases, not decreases.

Both properties constrain the errors when the original attention flow is close to 1, leading to reduced approximation errors. This property is generally favorable, as we are primarily concerned with the top source–destination pairs with the highest attention flow values. These pairs indicate the highest attention weights between token pairs, which are the focus of conventional attention visualization. The small error in this range, as observed with FlowPrune, enhances confidence in its results.

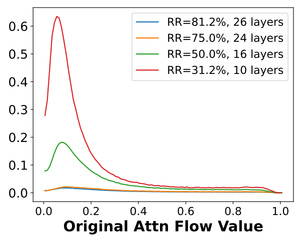
Top-K Token Retention. Many studies use attention flow methods focus on the *top-K token pairs with the highest attention flow*. To address this, we use the Top-K Token Retention metric to calculate the Jaccard overlap (IOU) of these token pairs. Figure 5 shows the results. In Figure 5, the horizontal



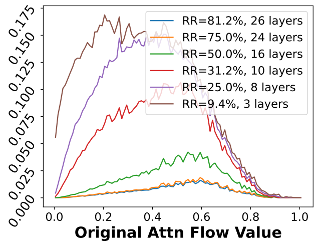
(a) Llama Absolute Error



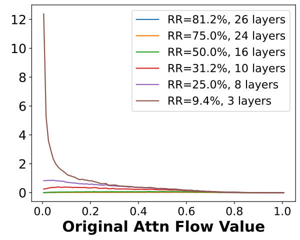
(b) Llama Relative Error



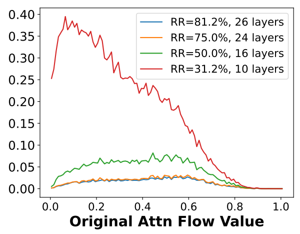
(c) Relative Error (RR $\geq$ 30%))



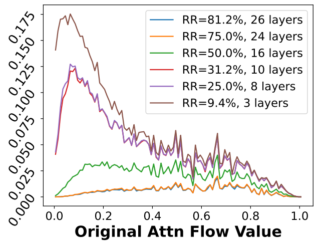
(d) LLaVA Absolute Error



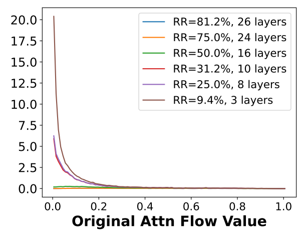
(e) LLaVA Relative Error



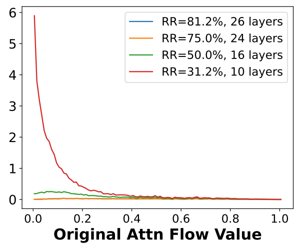
(f) Relative Error (RR $\geq$ 30%))



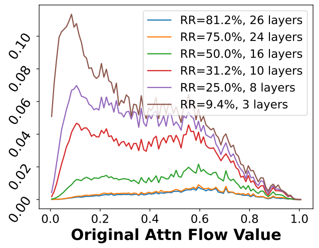
(g) Qwen Absolute Error



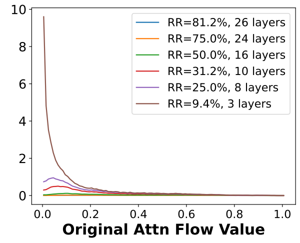
(h) Qwen Relative Error



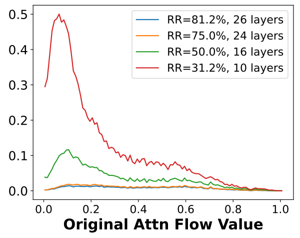
(i) Relative Error (RR $\geq$ 30%))



(j) QwenVL Absolute Error

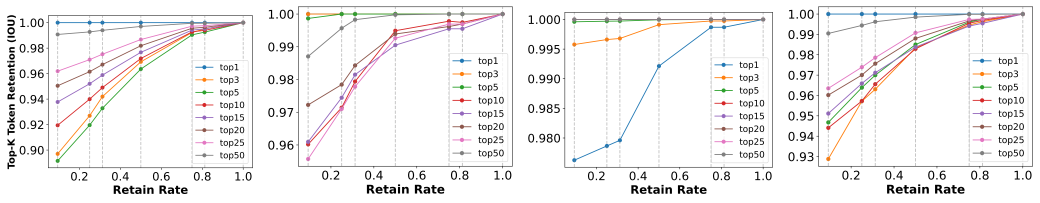


(k) QwenVL Relative Error



(l) Relative Error (RR $\geq$ 30%))

Figure 4: **Approximation Error vs. True Attention Flow Value.** Each subplot shows the approximation error (either absolute or relative) between FlowPrune and the original Attention Flow. The horizontal axis denotes the original attention flow values, and the vertical axis shows the average error magnitude for all token pairs falling within each value. Subplots (a)–(c) show the absolute and relative errors for Llama, and subplots (d)–(f), (g)-(i),(j)-(l) show the same for LLaVA, Qwen and QwenVL. RR represents retain rates.



(a) Llama (b) LLaVA (c) Qwen (d) QwenVL

Figure 5: **Top-K Token Retention.** We evaluate the overlap between the top-K tokens selected by FlowPrune and those selected by the original Attention Flow method for different values of K. Retention is measured using the Jaccard index (IoU), defined as IoU $= \frac{|A \cap B|}{|A \cup B|}$. Higher values indicate better preservation of salient attention targets.

axis represents the retain rate, defined as the ratio of layers in the compressed attention map to those in the original map. A lower retain rate corresponds to fewer remaining layers. The vertical axis shows Jaccard overlap values. The leftmost point represents the case where the retain rate is minimal, corresponding to compressing the attention map to three layers. Even in this case, the Top-K Token Retention remains around 0.90, demonstrating that our approximation algorithm maintains high fidelity for the top token pairs. This result aligns with findings from the Approximation Error experiment, which examined error variation with original attention flow values.

First Rank Divergence. Figure 6 shows the results of this metric for two models. The curves depict the average First Rank Divergence (FRD) across all attention map samples under varying retain rates. When the retain rate is restricted to 50%, the average FRD of both models exceeds 0.4. Notably, even at the highest compression level (retaining only 3 layers), the average FRD of Llama is close to 0.2, while the average FRD of LLaVA exceeds 0.3. This result is fully sufficient for practical applications with lower precision requirements. While the results at a retain rate 50% are adequate for tasks with higher precision demands.

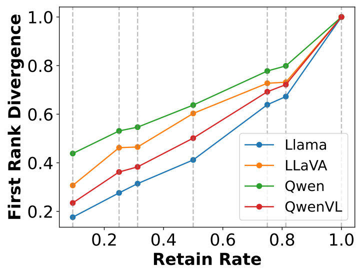


Figure 6: **First Rank Divergence.** This metric records the first position in the sorted top-K token list where the approximation diverges from the original output, normalized by the sequence length.

4.2 Application Scenarios

We use two application scenarios in both NLP and Vision, which are **Paraphrasing Verification** and **Vision Transformer (ViT) Heatmaps**, to show the practicality of FlowPrune. Paraphrase Verification [22] involves determining whether two sentences express the same or similar meaning. Attention Flow offers a visual analytics framework for examining self-attention patterns, illustrating how fine-tuning sharpens attention toward task-relevant details. In ViT, attention heatmaps depict the spatial distribution of attention weights, highlighting the model's focus regions during inference.

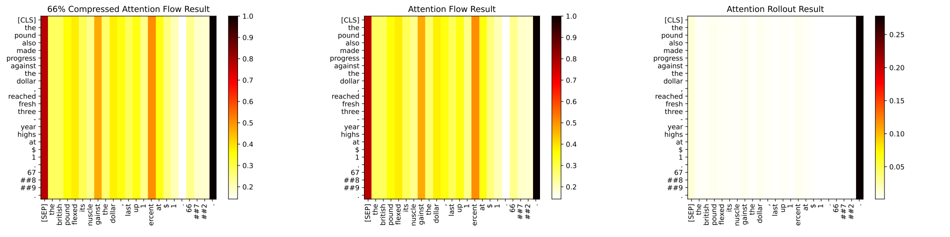
4.2.1 Paraphrasing Verification

In this task, we need to trace the attention flow from the classification token in the final layer backward through the network, across layers and attention heads. By comparing attention patterns between pre-trained and fine-tuned models, it reveals how fine-tuning sharpens attention on task-relevant distinctions, such as phrases that differentiate sentence meaning. This visual comparison offers valuable insights into the model's reasoning and decision-making process. Here, we replace the original Attention Flow algorithm with FlowPrune and compare its results against those of Attention Flow and Attention Rollout, demonstrating the practical efficiency and effectiveness of FlowPrune.

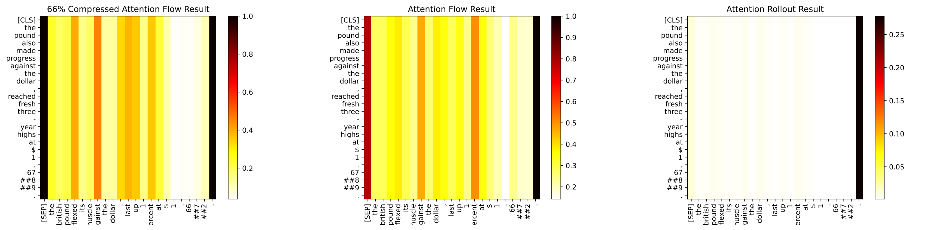
In this case, we implement the experiments on **MRPC** (Microsoft Research Paraphrase Corpus) [40] dataset with a 12-layer BERT model, which asks the model to determine whether two sentences have the same meaning. We compare the results of FlowPrune and Attention Rollout with the original

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

Max-n	FP Pretrained	AR Pretrained	FP Fine-tuned	AR Fine-tuned
$n = 3$	$98.50 \pm 0.86\%$	$63.40 \pm 2.58\%$	$95.00 \pm 1.50\%$	$84.50 \pm 2.32\%$
$n = 5$	$97.93 \pm 1.19\%$	$84.71 \pm 1.83\%$	$92.50 \pm 2.26\%$	$76.62 \pm 1.93\%$
$n = 10$	$97.69 \pm 1.33\%$	$66.38 \pm 1.58\%$	$91.45 \pm 2.58\%$	$70.85 \pm 1.56\%$



(a) FlowPrune (Pretrained) (b) Attention Flow (Pretrained) (c) Attention Rollout (Pretrained)



(d) FlowPrune (Fine-tuned) (e) Attention Flow (Fine-tuned) (f) Attention Rollout (Fine-tuned)

Figure 7: **Paraphrasing Verification Example.** Visualization of attention flow results across three methods—FlowPrune, original Attention Flow, and Attention Rollout—on a sentence-pair classification task. The two input sentences are: *(1) "The pound also made progress against the dollar, reached fresh three-year highs at $1.6789."* and *(2) "The British pound flexed its muscle against the dollar, last up 1 percent at $1.6672."*

Attention Flow in both pre-trained and fine-tuned models. In this task, we primarily focus on the attention metrics from tokens in the initial layer pointing to the *[cls]* token in the final layer. The tokens with the highest computed attention metrics are considered the most important.

We calculate the **Max-n Overlap** between the most important tokens obtained using FlowPrune/Attention Rollout and the original Attention Flow. For FlowPrune, we use a retain rate of 67%, reducing the original 12-layer attention map to 8 layers. Similar to the Top-K Token Retention, we also use the Jaccard overlap (IOU) as the corresponding metric. We take $n = 3, 5, 10$ and calculate the corresponding results. Table 1 shows the results, where we can see that FlowPrune has higher Max-n Overlap with original Attention Flow than Attention Rollout, proving FlowPrune is a better approximation than Attention Rollout in analyzing the model behaviour. We also show one example from MPRC in Figure 7. We find that the analysis results from FlowPrune are almost identical to those from the original Attention Flow, and both are capable of identifying the focal points of attention, effectively distinguishing between pretrained and fine-tuned models. In contrast, the Attention Rollout method is limited by its inherent flaws, resulting in inferior outcomes compared to the other two analysis methods.

4.2.2 Vision Transformer (ViT) Heatmaps

When conducting an attention interpretability analysis for the ViT model, we may be interested in understanding the attention relevance of a specific token to all other tokens to construct an attention heatmap. To test whether FlowPrune is useful here, we use the **Deit-Small** [41] model and implement the experiments on **ILSVRC2012** [42], an image classification dataset. Similarly, we primarily focus on the attention metrics from tokens in the initial layer pointing to the *[cls]* token in the final layer, and compare the relevant analysis results of Attention Flow, FlowPrune, and Attention Rollout. For FlowPrune, we use a retain rate of 25%, reducing the 12-layer attention map to 3 layers.

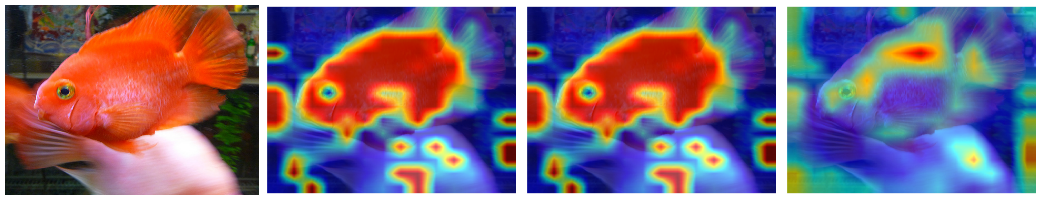


Figure 8: **ViT Heapmap Example.** We visualize the attention attribution maps produced by three methods — FlowPrune, the original Attention Flow and Attention Rollout — on a ViT model. All heatmaps correspond to the [CLS] token and show which regions of the image contribute most to the model's final prediction.

Table 2: **Max-n Overlap for ViT Heatmap.** We report the top-n region overlap between FlowPrune (FP), Attention Rollout (AR), and the reference attention map computed via full Attention Flow. (Mean ± SE)

Max-n	FP	AR
$n = 10$	$100.0 \pm 0.00\%$	$28.18 \pm 3.06\%$
$n = 20$	$99.53 \pm 0.12\%$	$50.84 \pm 3.74\%$
$n = 30$	$99.59 \pm 0.11\%$	$78.52 \pm 3.43\%$

Like Paraphrasing Verification, we calculate the **Max-n Overlap** between Original Attention Flow and FlowPrune/Attention Rollout. Considering that ViT has more input tokens, We take $n = 10, 20, 30$ and get results in Table 2. We can see FlowPrune has a much higher Max-n Overlap with the original Attention Flow than Attention Rollout, indicating FlowPrune is a better approximation than Attention Rollout in constructing the heatmaps for ViT. Figure 8 presents an example of a ViT heatmap, which was generated using three different methods. **Deit-Small** model utilizes the *[CLS]* token for image classification tasks. Therefore, we focus on the attention metrics from all tokens to the *[CLS]* token to construct heatmaps. The analysis results of FlowPrune once again align with the original Attention Flow, while the Attention Rollout method, due to its inherent flaws, fails to effectively analyze the propagation of attention.

5 Conclusion and Limitation

In this work, we propose **FlowPrune**, a novel framework to accelerate attention flow computation, aiming to facilitate the analysis of Transformer-based networks. Motivated by the Max-Flow Min-Cut Theorem, FlowPrune leverages two structural properties of Transformers: the use of the softmax operation in attention computation and the strictly layered architecture of attention graphs. FlowPrune introduces two key components: **Edge Pruning**, which removes insignificant attention edges, and **Layer Compression**, which discards layers with minimal contributions to the flow. Together, these techniques effectively reduce the scale of the attention graph while preserving the fidelity of attention flow analysis. Our method provides a practical and efficient solution for scalable interpretability in large Transformer models, and holds promise for guiding further analysis and application of Large Language Models [43] [44] [45].

While **FlowPrune** effectively accelerates attention flow computation with minimal loss in accuracy for most cases, its relative error tends to correlate with the magnitude of the original attention flow values. In particular, when the original flow value is very small, the relative error introduced by FlowPrune can become excessively large, which may render the results unreliable for interpretability in those regions. This suggests that caution should be exercised when analyzing low-magnitude flows, as the pruning process may disproportionately distort their values.

6 Acknowledgment

This work is supported by the National Science Foundation of China (62576091). This research work is also supported by the Big Data Computing Center of Southeast University. This work was also partially supported by the Southeast University Kunpeng & Ascend Center of Cultivation and the Big Data Computing Center of Southeast University.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[2] 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.

[3] Simeng Han, Frank Palma Gomez, Tu Vu, Zefei Li, Daniel Cer, Hansi Zeng, Chris Tar, Arman Cohan, and Gustavo Hernandez Abrego. Ateb: Evaluating and improving advanced nlp tasks for text embedding models. *arXiv preprint arXiv:2502.16766*, 2025.

[4] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

[5] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

[6] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[7] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

[8] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[9] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

[10] Sarthak Jain and Byron C. Wallace. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3541–3556, 2019.

[11] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4090–4100, 2019.

[12] Jesse Vig and Yonatan Belinkov. Analyzing the structure of attention in a transformer language model. *arXiv preprint arXiv:1906.04284*, 2019.

[13] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the association for computational linguistics*, 8:842–866, 2021.

[14] Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. *arXiv preprint arXiv:2005.00928*, 2020.

[15] Joseph Cheriyan and Kurt Mehlhorn. An analysis of the highest-level selection rule in the preflow-push max-flow algorithm. *Information Processing Letters*, 69(5):239–242, 1999.

[16] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1721–1730, 2015.

[17] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[18] Lester R Ford Jr and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of mathematics*, 8(3):399–404, 1956.

[19] Liu Liu, Zheng Qu, Zhaodong Chen, Yufei Ding, and Yuan Xie. Transformer acceleration with dynamic sparse attention, 2021.

[20] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[21] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[22] Joseph F DeRose, Jiayao Wang, and Matthew Berger. Attention flows: Analyzing and comparing attention mechanisms in language models, 2020.

[23] Sofia Serrano and Noah A Smith. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2931–2951, 2019.

[24] Jasmijn Bastings and Katja Filippova. The elephant in the interpretability room: Why use attention as explanation when we have saliency methods? *arXiv preprint arXiv:2010.05607*, 2020.

[25] Danish Pruthi, Mansi Gupta, Bhuwan Dhingra, Graham Neubig, and Zachary C Lipton. Learning to deceive with attention-based explanations. *arXiv preprint arXiv:1909.07913*, 2019.

[26] Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 782–791, 2021.

[27] Qiu-Feng Wang, Xin Geng, Shu-Xia Lin, Shi-Yu Xia, Lei Qi, and Ning Xu. Learngene: From open-world to your learning task. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8557–8565, 2022.

[28] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[29] Hamidreza Ghader and Christof Monz. What does attention in neural machine translation pay attention to? *arXiv preprint arXiv:1710.03348*, 2017.

[30] Simeng Han, Tianyu Liu, Chuhan Li, Xuyuan Xiong, and Arman Cohan. Hybridmind: Meta selection of natural language and symbolic language for enhanced llm reasoning. *arXiv e-prints*, pages arXiv–2409, 2024.

[31] Jack Edmonds and Richard M Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

[32] Efim A Dinic. Algorithm for solution of a problem of maximum flow in networks with power estimation. In *Soviet Math. Doklady*, volume 11, pages 1277–1280, 1970.

[33] Andrew V Goldberg and Robert E Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, 35(4):921–940, 1988.

[34] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[35] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024.

[36] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution, 2024.

[37] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[38] Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pages 3195–3204, 2019.

[39] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579, 1901.

[40] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019.

[41] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers distillation through attention, 2021.

[42] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. pages 248–255, 2009.

[43] Xu Yang, Hanwang Zhang, and Jianfei Cai. Deconfounded image captioning: A causal retrospect. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):12996–13010, 2021.

[44] Xu Yang, Yongliang Wu, Mingzhuo Yang, Haokun Chen, and Xin Geng. Exploring diverse in-context configurations for image captioning. *Advances in Neural Information Processing Systems*, 36:40924–40943, 2023.

[45] Fu Feng, Jing Wang, Xu Yang, and Xin Geng. Learngene: Inheritable "genes" in intelligent agents. *Artificial Intelligence*, page 104421, 2025.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The Flowprune algorithm introduced in the abstract and the introduction is presented in Section 3. The experiments are described in Section 4.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: In Section 5, we introduce the limitations of the Flowprune algorithm.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: The time complexity of the attention flow using the HLPP algorithm is analyzed in Section 3. The acceleration in time due to the compression ratio is analyzed in Section 4.1.2 under Computational Cost (Time) part.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: In Section 4, we have provided all the information needed to reproduce the experiments.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We have provided open access to our code. the code link is provided in abstract.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: We specify all the training and test details necessary to understand the results in Section 4

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [No]

Justification: The models we used are relatively large and the datasets are extensive. However, the random seed settings for all experiments are identical to ensure the fairness of the comparisons.

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: All the information required to reproduce the experiments is mentioned at the beginning of Section 4.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have read and complied with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [NA]

    Justification: There is no societal impact of the work performed

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: This paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The creators or original owners of the assets we used have been properly credited. These can be verified in Section 4.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [NA]

    Justification: This paper does not release new assets.

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Supplementary Materials

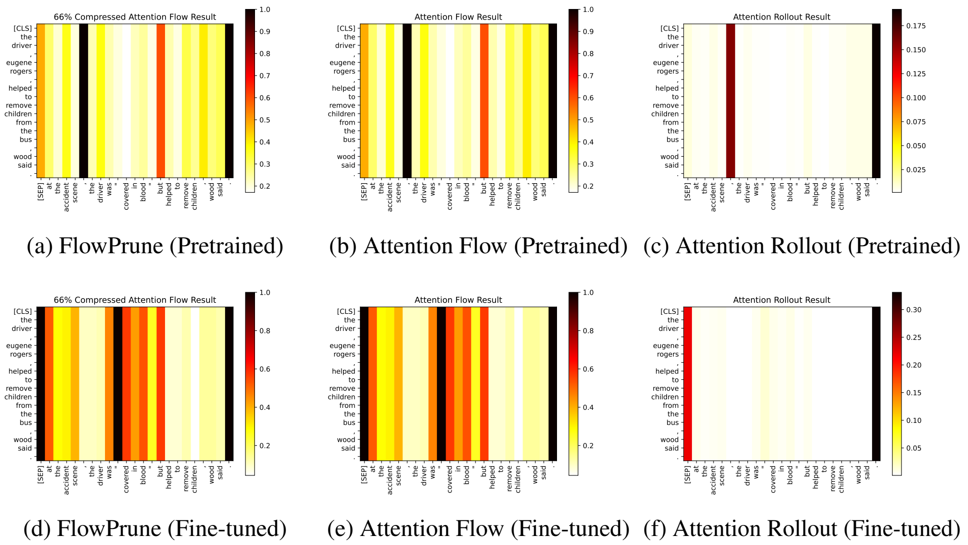
To provide additional qualitative evidence for the effectiveness of the proposed FlowPrune framework, this appendix presents five supplementary examples for both **Paraphrasing Verification** and **Vision Transformer (ViT) Heatmaps** Analysis in Section 4. We also included an application scenario experiment on **Question Answer Verification** [22], which is also based on BERT. In addition, we provide a detailed description of the dynamic programming algorithm used to determine which attention layers to compress in FlowPrune.

A.1 Paraphrasing Verification

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

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

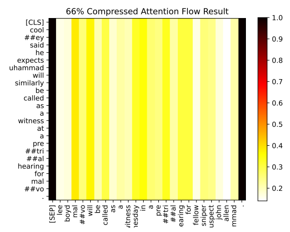
Figures 9 through 13 show five representative examples of the resulting attention heatmaps. As demonstrated, FlowPrune consistently aligns with the original Attention Flow in highlighting key semantic alignments and distinguishing features between the sentences. Notably, the differences between pre-trained and fine-tuned models are also clearly reflected in the FlowPrune heatmaps, showcasing its ability to retain meaningful interpretability insights under compression.



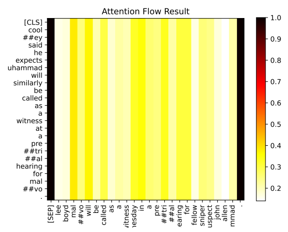
(a) FlowPrune (Pretrained) (b) Attention Flow (Pretrained) (c) Attention Rollout (Pretrained)

(d) FlowPrune (Fine-tuned) (e) Attention Flow (Fine-tuned) (f) Attention Rollout (Fine-tuned)

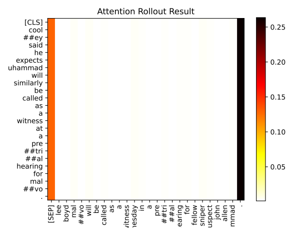
Figure 9: **Paraphrasing Verification Example 1.** The two input sentences are: *(1) "The driver , Eugene Rogers , helped to remove children from the bus , Wood said ."* and *(2) "At the accident scene , the driver was " covered in blood " but helped to remove children , Wood said ."*



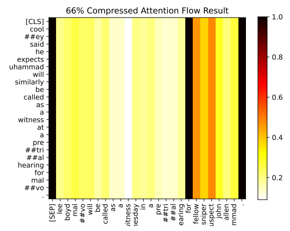
(a) FlowPrune (Pretrained)



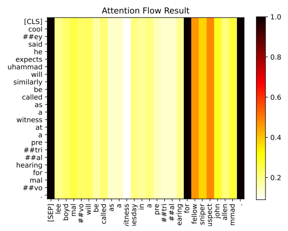
(b) Attention Flow (Pretrained)



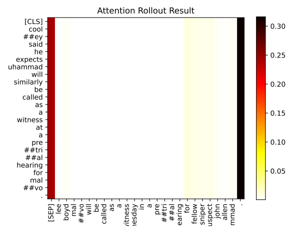
(c) Attention Rollout (Pretrained)



(d) FlowPrune (Fine-tuned)

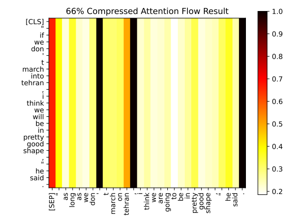


(e) Attention Flow (Fine-tuned)

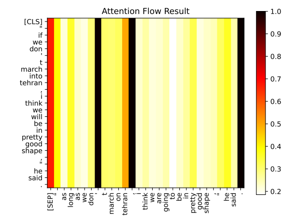


(f) Attention Rollout (Fine-tuned)

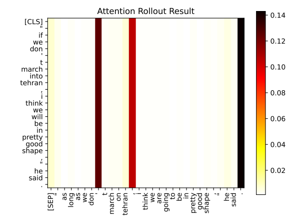
Figure 10: **Paraphrasing Verification Example 2.** The two input sentences are: *(1) "Cooley said he expects Muhammad will similarly be called as a witness at a pretrial hearing for Malvo ."* and *(2) "Lee Boyd Malvo will be called as a witness Wednesday in a pretrial hearing for fellow sniper suspect John Allen Muhammad ."*



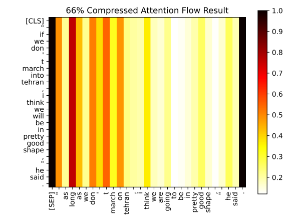
(a) FlowPrune (Pretrained)



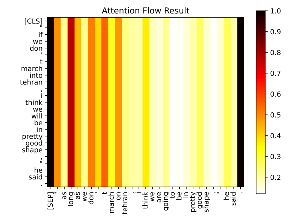
(b) Attention Flow (Pretrained)



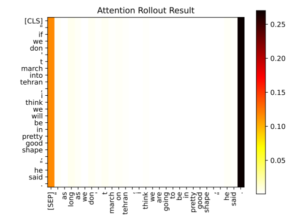
(c) Attention Rollout (Pretrained)



(d) FlowPrune (Fine-tuned)

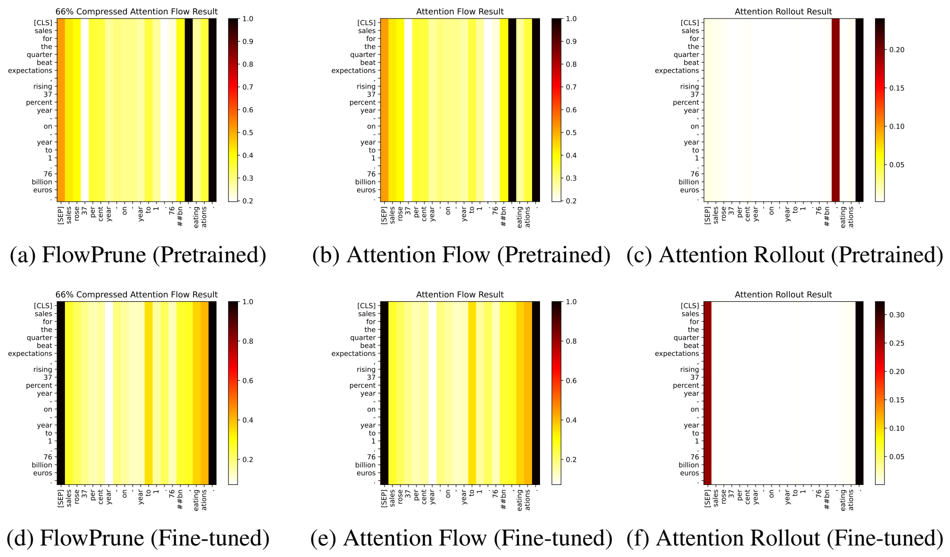


(e) Attention Flow (Fine-tuned)



(f) Attention Rollout (Fine-tuned)

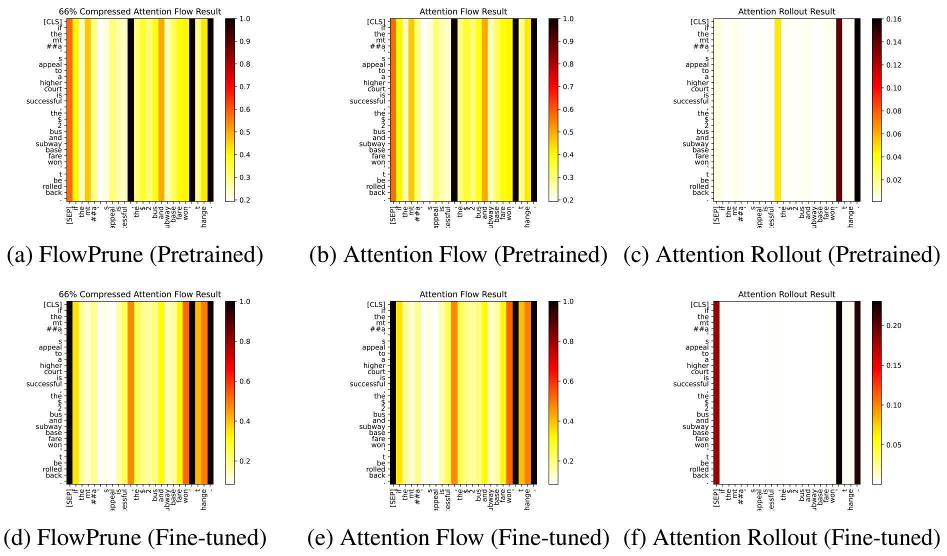
Figure 11: **Paraphrasing Verification Example 3.** The two input sentences are: *(1) "" If we don 't march into Tehran , I think we will be in pretty good shape , " he said ."* and *(2) "" As long as we don 't march on Tehran , I think we are going to be in pretty good shape , " he said ."*



(a) FlowPrune (Pretrained) (b) Attention Flow (Pretrained) (c) Attention Rollout (Pretrained)

(d) FlowPrune (Fine-tuned) (e) Attention Flow (Fine-tuned) (f) Attention Rollout (Fine-tuned)

Figure 12: **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 ."*



(a) FlowPrune (Pretrained) (b) Attention Flow (Pretrained) (c) Attention Rollout (Pretrained)

(d) FlowPrune (Fine-tuned) (e) Attention Flow (Fine-tuned) (f) Attention Rollout (Fine-tuned)

Figure 13: **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 ."*

A.2 Vision Transformer (ViT) Heatmaps

To further evaluate the utility of FlowPrune in visual interpretability tasks, we provide additional examples using the Vision Transformer (ViT) model. Specifically, we apply FlowPrune to the **DeiT-Small**[41] model on the **ILSVRC2012**[42] image classification dataset. The aim is to generate attention heatmaps that reflect the relevance between each input image token and the *[CLS]* token, which is responsible for final classification decisions. In this setting, we compare three interpretability methods: FlowPrune, the original Attention Flow, and Attention Rollout.

Figures 14 through 18 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.

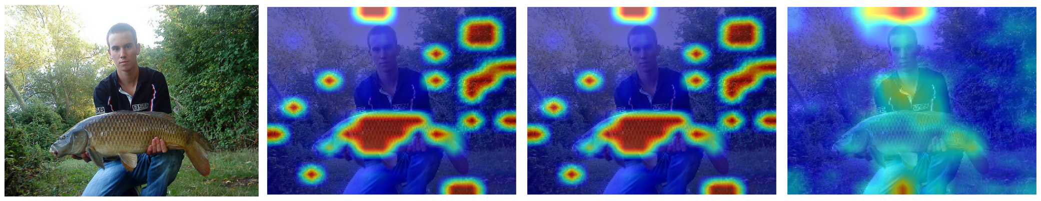

(a) Original Image (b) Attention Flow (c) FlowPrune (d) Attention Rollout

Figure 14: **ViT Heapmap Example 1.**

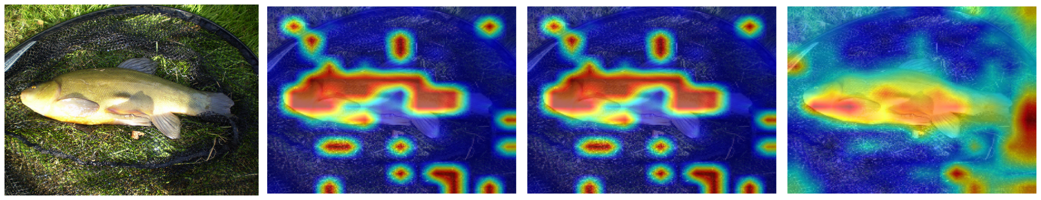

(a) Original Image (b) Attention Flow (c) FlowPrune (d) Attention Rollout

Figure 15: **ViT Heapmap Example 2.**

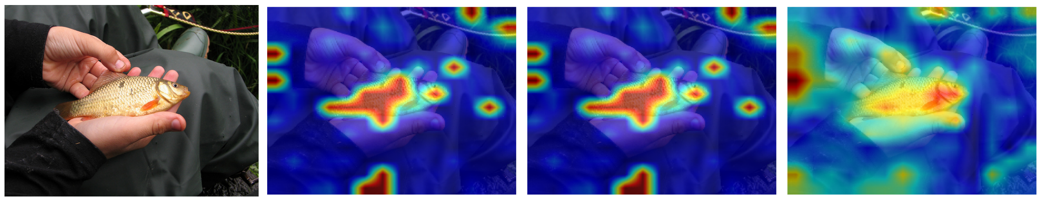

(a) Original Image (b) Attention Flow (c) FlowPrune (d) Attention Rollout

Figure 16: **ViT Heapmap Example 3.**

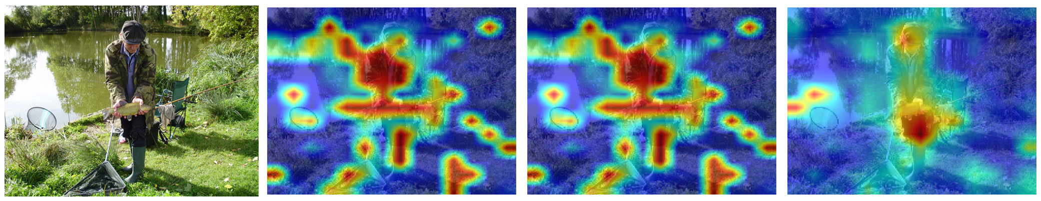
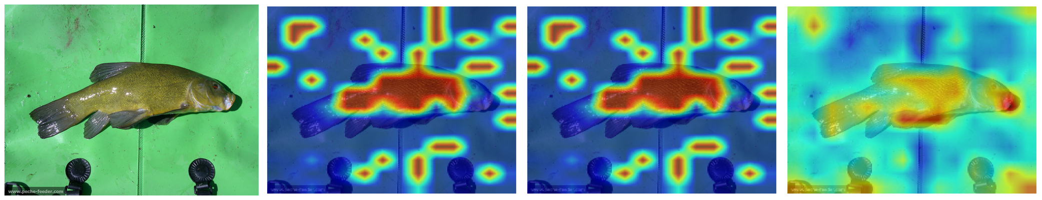

(a) Original Image (b) Attention Flow (c) FlowPrune (d) Attention Rollout

Figure 17: **ViT Heapmap Example 4.**



(a) Original Image (b) Attention Flow (c) FlowPrune (d) Attention Rollout

Figure 18: **ViT Heapmap Example 5.**

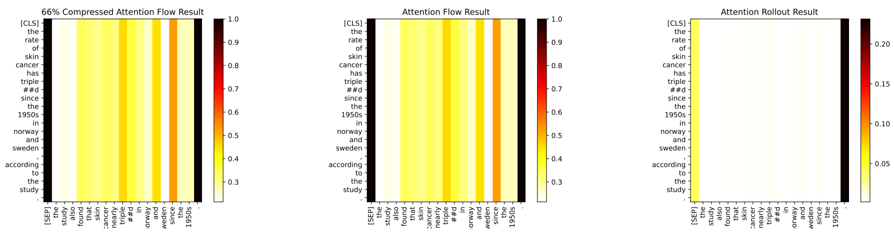
A.3 Question Answer Verification

The Question Answer Verification (QAV) experiment aims to investigate how fine-tuning affects BERT's ability to validate whether a given answer correctly addresses a question. Using the QNLI [40] dataset from GLUE, the study fine-tunes a BERT-base model and employs Attention Flows to analyze changes in the model's attention mechanisms. The results show that fine-tuning enables the model to better focus on task-relevant details, such as key connecting words and phrases between the question and answer, thereby improving its accuracy in determining the validity of the answer.

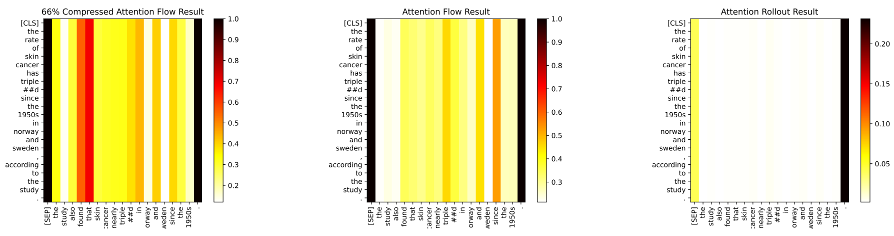
In this experiment, we continue to utilize a 12-layer BERT model. We aim to compare the outcomes of FlowPrune and Attention Rollout with those of the original Attention Flow, both in pre-trained and fine-tuned models. The primary focus of this task is the attention metrics from tokens in the initial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

Table 3: **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
$n = 3$	$98.05 \pm 1.11\%$	$64.68 \pm 2.91\%$	$96.10 \pm 1.54\%$	$85.71 \pm 2.59\%$
$n = 5$	$97.31 \pm 1.54\%$	$84.48 \pm 2.14\%$	$94.16 \pm 2.31\%$	$76.31 \pm 2.21\%$
$n = 10$	$97.01 \pm 1.72\%$	$66.50 \pm 1.88\%$	$93.42 \pm 2.60\%$	$71.03 \pm 1.76\%$



(a) FlowPrune (Pretrained) (b) Attention Flow (Pretrained) (c) Attention Rollout (Pretrained)



(d) FlowPrune (Fine-tuned) (e) Attention Flow (Fine-tuned) (f) Attention Rollout (Fine-tuned)

Figure 19: **Question Answer Verification.** Visualization of attention flow results across three methods—FlowPrune, original Attention Flow, and Attention Rollout.

We quantify the **Max-n Overlap** between the most salient tokens derived via FlowPrune/Attention 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 3. Additionally, an illustrative example from the MPRC dataset is depicted in Figure 19.

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 $\{a_1, a_2, \ldots, a_{L-2}\}$, 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:

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

The boundary conditions are:

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

The optimal value is given by:

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

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.