# SPARSEFLOW: Accelerating Transformers by Sparsifying Information Flows

**Yeachan Kim**[1] **and SangKeun Lee**[1,2]
[1]Department of Artificial Intelligence, Korea University, Seoul, South Korea
[2]Department of Computer Science and Engineering, Korea University, Seoul, South Korea
{yeachan,yalphy}@korea.ac.kr

## Abstract

Transformers have become the *de-facto* standard for natural language processing. However, dense information flows within transformers pose significant challenges for real-time and resource-constrained devices, as computational complexity grows quadratically with sequence length. To counteract such dense information flows, we propose SPARSEFLOW, a novel efficient method designed to sparsify the dense pathways of token representations across all transformer blocks. To this end, SPARSEFLOW parameterizes the information flows linking token representations to transformer blocks. These parameterized information flows are optimized to be sparse, allowing only the salient information to pass through into the blocks. To validate the efficacy of SPARSEFLOW, we conduct comprehensive experiments across diverse benchmarks (understanding and generation), scales (ranging from millions to billions), architectures (including encoders, decoders, and seq-to-seq models), and modalities (such as language-only and vision-language). The results convincingly demonstrate that sparsifying the dense information flows leads to substantial speedup gains without compromising task accuracy. For instance, SPARSEFLOW reduces computational costs by half on average, without a significant loss in accuracy[1].

## 1 Introduction

Transformers (Vaswani et al., 2017) have brought about a paradigm shift in diverse research areas such as NLP (Brown et al., 2020; Chowdhery et al., 2022) and computer vision (Dosovitskiy et al., 2021; Liu et al., 2021). However, their undeniable effectiveness often comes with a non-negligible computational burden, scaling quadratically with the length of the input sequence. This bottleneck poses a critical challenge in realizing the full potential of Transformers, particularly in resource-constrained or real-time systems (Sun et al., 2020).

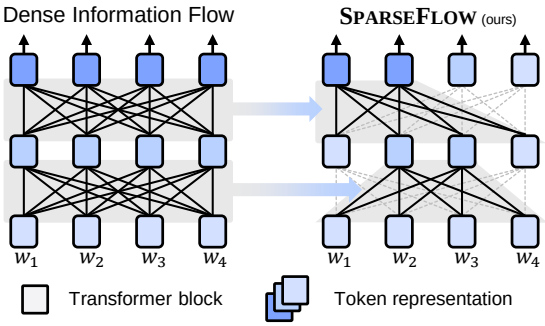



Figure 1: High-level concept of SPARSEFLOW that sparsifies the current dense information flows.

A primary contributor to this inefficiency is the dense *information flows*[2] between all token representations, specifically, the pairwise computations between every pair of tokens. While these dense information flows are a source of strength in transformers, due to their minimal inductive bias, they often lead to redundancy in token representations (Ethayarajh, 2019), resulting in different tokens carrying similar information (Goyal et al., 2020). Such redundancy implies that the current pre-trained language models involve a large amount of expensive yet unnecessary computations. This observation underscores the potential benefits of enhancing efficiency by simplifying the complexity of the current dense information flows.

In this paper, we propose SPARSEFLOW, a novel efficient method designed to sparsify the current dense information flows across all transformer blocks. To this end, SPARSEFLOW parameterizes the information flows linking token representations to each transformer block. These parameters are subsequently optimized to be sparse such that only

[1]Our code is available at https://github.com/yeachan-kr/sparseflow

[2]Following (Abnar and Zuidema, 2020), we represent a information flow as *DAG* (Directed Acyclic Graph) where nodes are input tokens and its representations and edges are the interaction between them (e.g., feed-forward or self-attention).



*Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5937–5948

the salient information flows are preserved. To effectively optimize the sparsified patterns for diverse inputs, we extend SPARSEFLOW through sparsely activated mixture-of-experts (Fedus et al., 2022). By engaging each sample with the optimized sparse pathways, SPARSEFLOW streamlines the transformer inference while preserving the task accuracy, offering practical value across a broad spectrum of applications.

To confirm the efficacy of SPARSEFLOW, we conduct extensive experiments, encompassing diverse tasks (natural language understanding and generation), scales (from millions to billions), architectures (including encoder, decoder, and sequence-to-sequence models), and modalities (language and vision). We further perform a thorough analysis to gain insights into the behavior of SPARSEFLOW by examining the information flow of task-relevant tokens and the sparsified patterns of this flow. In summary, the contributions of this paper include the following:

- We propose SPARSEFLOW, a novel efficient method that learns to sparsify dense information flows, thereby enhancing the efficiency of pre-trained language models.
- We demonstrate experimentally that SPARSEFLOW achieves substantial speedup gains while preserving the task accuracy in comparison to strong baselines.
- We confirm the general applicability of SPARSEFLOW on different modalities, architectures, and its scales, showcasing the practical usefulness in a wide range of applications.

## 2 Related Work

**Representation Removal in Transformers** Dense information flows in self-attention often lead to redundancy in token representations (Ethayarajh, 2019; Goyal et al., 2020). This facilitates numerous studies aimed at removing these redundant representations during a forward pass. The major strategy behind those studies is to reduce computations in transformer blocks depending on the input representations, and they can be categorized in two-fold: (i) depth reduction (also known as early-exit) (ii) width reduction (also known as token pruning).The former approach reduces computations by decreasing the number of computing layers (Zhou et al., 2020; Zhang et al., 2022), while the latter does so by shortening the number of tokens forwarded into transformer blocks (Goyal et al., 2020; Kim et al., 2022; Guan et al., 2022; Kim et al., 2023). Recent works have more focused on token pruning approach as it can optimize the computational resources in a more fine-grained way (Kim et al., 2023). Given these categorizations, SPARSEFLOW aligns more closely with the token pruning approach.

**Token Pruning** The earliest token pruning approach is PoWER-BERT (Goyal et al., 2020) that maintains only the pre-determined ratio of tokens based on the attention weights while removing others. However, due to the re-training costs of PoWER-BERT for different environments, LAT (Kim and Cho, 2021) extends it to effectively fit diverse configuration of computational budgets without the retraining. Subsequently, LTP (Kim et al., 2022) removes the tokens by optimizing the trainable threshold for the attention values to identify which tokens are significant to tasks. Beyond manual analysis on attention weights, recent works move to automatic methods that learn to select tokens to be removed during training. For example, TR-BERT (Ye et al., 2021) and Transkimmer (Guan et al., 2022) have suggested token removal strategies that can be learned during training, by using reinforcement learning and re-parameterization, respectively. AdapLeR (Modarressi et al., 2022) and LoT(Kim et al., 2023) have proposed a saliency-based strategy that eliminates tokens by estimating the gradients of the token representations with respect to the predictions.

Compared to token pruning, SPARSEFLOW has distinct characteristics. The token pruning methods learn individual sparse patterns for each data. In contrast, SPARSEFLOW learns the generalized patterns across all data by directly parameterizing information flows. Furthermore, token pruning methods necessitate additional procedures (e.g., attention analysis) or specialized layers (e.g., token classifiers) to identify which tokens to remove. However, SPARSEFLOW eliminates these processes, as it pre-determines the sparsified pathways during the training phase, enabling streamlined inference. Lastly, while token pruning methods progressively discard token information from the input sequence, SPARSEFLOW maintains all token information but reduces connections in the flows. Such a difference renders SPARSEFLOW less susceptible to the information loss commonly associated with token pruning methods (Zhong et al., 2023).

## 3 SPARSEFLOW

In this section, we elaborate on SPARSEFLOW, which is designed to sparsify the dense information flow in current pre-trained language models. We first revisit the dense and sparse information flows in transformer (§3.1). To construct the sparse information flows, we directly parameterize the information flow and introduce a binarization method to sparsify these parameterized flows (§3.2). To cover the diverse inputs, we extend SPARSEFLOW through adaptively optimizing multiple sets of parameterized information flows (§3.3). These parameters are lastly optimized to be sparse (§3.4), allowing only the salient flows to be preserved.

### 3.1 Information Flow in Transformer

**Dense Information Flow.** We start by revisiting the dense information flows in the transformer. Due to the parallel computation over the sequence, the information flows can be decomposed by its sequence index (i.e., token position). Let the token representations of the input $x$ in the $l$-th transformer block be denoted as $\mathbf{X}^{(l)}, l \in [1, L]$ where $L$ is the number of layer, and its representation at $i$-th position be denoted as $\mathbf{X}_i^{(l)}$, the information flow at the $i$-th position starts by passing through the self-attention layer as follows:

$$\mathbf{X}_i^{(l)} = \text{softmax}\left(\frac{q_i \mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \tag{1}$$

where $q_i = \mathbf{X}_i^{(l)}\mathbf{W}_q$, $\mathbf{K} = \mathbf{X}^{(l)}\mathbf{W}_k$, and $\mathbf{V} = \mathbf{X}^{(l)}\mathbf{W}_v$ are the query vector of the $i$-th position, key, and value matrices derived from the input tokens, and $d_k$ is the dimensionality of the keys. Subsequently, the updated representation is then forwarded to the following feed-forward networks[3].

$$\mathbf{X}_i^{(l+1)} = \sigma_{\text{FF}}(\mathbf{X}_i^{(l)}\mathbf{W}_{F_1})\mathbf{W}_{F_2}, \tag{2}$$

where $\mathbf{W}_{F_1}, \mathbf{W}_{F_2}$ are learnable weight matrices, and $\sigma_{\text{FF}}(\cdot)$ denotes an activation function of the feed-forward layers. The above information flows occurs at every token position, and it results in the redundancy with quadratic computational costs with the sequence length (Goyal et al., 2020).

[3]For simplicity, we denote the process of information flows only within self-attention and feedforward layers. In practice, the information flows include processes such as layer normalization and skip connections.

**Sparse Information Flow.** To counteract such dense information flows, we propose SPARSEFLOW that sparsifies the dense information flows to be sparse by reducing the number of positions used for computations (i.e., Eq. (1) and Eq. (2)). Let the reduced subset of the information flows in the $l$-th block of the transformer be denoted as $\mathcal{P}^{(l)}$, the sparse information flow is defined through the computation over this subset:

$$\mathbf{X}^{(l)} \leftarrow \left\{ \text{softmax}\left(\frac{q_i \mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \mid i \in \mathcal{P}^{(l)} \right\}, \tag{3}$$

Similarly, the resulting representations of the subsets are forwarded to the subsequent feed-forward layers as follows:

$$\mathbf{X}^{(l+1)} \leftarrow \left\{ \sigma(\mathbf{X}_i^{(l)}\mathbf{W}_{F_1})\mathbf{W}_{F_2} \mid i \in \mathcal{P}^{(l)} \right\}, \tag{4}$$

By including only a subset $\mathcal{P}^{(l)}$ forwarded to each transformer block, SPARSEFLOW reduces the computation and memory requirement. Specifically, let the size of the subset $\mathcal{P}^{(l)}$ is $m \ll n$ where $n$ is the length of the input sequence, the self-attention complexity is reduced from $\mathcal{O}(n^2 d)$ to $\mathcal{O}(mnd)$, and the complexity of the position-wise feed-forward is reduced from $\mathcal{O}(nd^2)$ to $\mathcal{O}(md^2)$.

### 3.2 Parameterized Information Flows

Given the definition of SPARSEFLOW, its efficacy hinges on how we optimize the subset $\mathcal{P}^{(l)}$. Specifically, the subset should remain sufficiently small to enhance efficiency without compromising task accuracy. To achieve this, we directly parameterize this subset to control the complexity of information flows. Let the maximum length of transformers be denoted as $N_{\text{max}}$, the subset of information flows in the $l$-th block is parameterized as $p^{(l)} \in \mathbb{R}^{N_{\text{max}}} = [p_1^{(l)}, p_2^{(l)}, \ldots, p_{N_{\text{max}}}^{(l)}]$ where the value of $p_i^{(l)}$ is a scalar indicator of the information flow. To represent existence in the subset, the parameters are constrained to have binary values (i.e., 1 for the utilized information flows, 0 for the skipped ones). These parameters are then multiplied to token representations to bypass the computation in each block, i.e., $p^{(l)} \odot \mathbf{X}^{(l)}$.

However, optimization with the binary constraints is non-trivial as the binarization function is not differentiable. Inspired by the continuous relaxation trick for the discrete function (Jang et al., 2017), we approximate the binary constraints through the Gumbel trick with sigmoid function,

i.e., Gumbel-sigmoid. Specifically, as the sigmoid function can be viewed as a 2-class softmax where logits are learnable parameter $p_i^{(l)}$ and the fixed value of 0, the Gumbel-sigmoid is calculated as follows:

$$p_i^{(l)} = \frac{\exp((p_i^{(l)} + g')/\tau)}{\exp((p_i^{(l)} + g')/\tau) + \exp(g''/\tau)} = \sigma((p_i^{(l)} + g' - g'')/\tau), \quad (5)$$

where $\sigma(\cdot)$ denotes the sigmoid function, $g'$ and $g''$ are independent samples drawn from a standard Gumbel distribution[4], and $\tau$ is a temperature parameter controlling the steepness of the sigmoid function. As the temperature $\tau$ approaches zero, the sigmoid function becomes more like a step function, enabling the optimization of the subset parameters using gradient-based methods.

### 3.3 Mixture of Sparse Information Flows

The parameterized information flows are optimized to generalize across all training samples. However, employing a single sparse pattern to cover all data can be sub-optimal, as different inputs may require different patterns. Inspired by the concept of sparsely activated mixture-of-experts (MoE) (Fedus et al., 2022), we extend SPARSEFLOW to a dynamic method where multiple sparse patterns are selectively activated for different inputs.

To this end, we first duplicate the subset parameters, ensuring that $p^{(l)} \in \mathbb{R}^{K \times N_{\max}}$ where $K$ is the number of sets of parameterized information flows (i.e., the number of experts in MoE). We then introduce a learnable router $w(\cdot)$, which consists of a single feed-forward layer. The router takes the average of input embeddings $\overline{\mathbf{X}}^{(0)}$ over the sequence as input and produces the routing probability to select the subset parameters, as follows:

$$w(x) = \text{softmax}(\mathbf{W}^\top \overline{\mathbf{X}}^{(0)}) \quad (6)$$

where $\mathbf{W} \in \mathbb{R}^{d \times K}$ is a learnable weights. With the routing probability, the selected subset parameter is derived through the soft weighted average of the subset parameters[5]:

$$p_i^{(l)} = \sum_{k=1}^{K} w(x)_k \cdot p_{k,i}^{(l)} \quad (7)$$

where $p_{k,i}^{(l)}$ indicates the $i$-th connection parameter of the $l$-th transformer block in the $k$-th pattern. These derived parameters are subsequently binarized using the Gumbel-sigmoid function (as in Eq (5)).

### 3.4 Information Flow Optimization

**Sparse Regularization** To inject the sparsity into the parameterized information flows, we regularize the subset parameters to be zero, thereby resulting in the sparse subset $\mathcal{P}$. This is achieved by adding L2 regularization term to the loss function, defined as follows:

$$\mathcal{L}_{\text{sparse}} = \frac{1}{L} \sum_{l=1}^{L} \left( \alpha^{(l)} - \frac{1}{n} \sum_{i=1}^{n} p_i^{(l)} \right)^2, \quad (8)$$

where $\alpha^{(l)}$ is the target sparse ratio in the $l$-th block to prevent the model from overly removing the information flow, and $\lambda$ is a hyper-parameter that controls the degree of sparsity. This regularization sparsifies the information flow until having the minimum size of subsets (i.e., $\alpha^{(l)}$) while learning the given task.

**Overall Objective** The overall loss function comprises a task-specific loss (such as cross-entropy for classification tasks) and sparsity regularization. Additionally, we add a consistency regularization loss to ensure predictions remain consistent before and after sparsification, as follows:

$$\mathcal{L}_{\text{cons}}(x, y) = \text{CE}(p(y; x), p(y; x_{\text{sparse}})) \quad (9)$$

where $\text{CE}(\cdot)$ denote the cross-entropy, $p(y; x)$ and $p(y; x_{\text{sparse}})$ are the prediction over classes produced by the model with the dense information flow and SPARSEFLOW, respectively. With this regularization, the total loss function for a given input $x$ and its label $y$ can be expressed as:

$$\mathcal{L}(x, y; \theta) = \mathcal{L}_{\text{task}}(x, y; \theta) + \lambda_{\text{sparse}} \mathcal{L}_{\text{sparse}} + \lambda_{\text{cons}} \mathcal{L}_{\text{cons}}(x, y), \quad (10)$$

where $\mathcal{L}_{\text{task}}(x, y; \theta)$ is the loss function for the primary task (such as classification or translation), $\lambda_{\text{sparse}}$ and $\lambda_{\text{cons}}$ are hyper-parameters that balance the importance of sparsity and consistency regularization, respectively. Through the above optimization procedures, SPARSEFLOW learns to use the most salient information flows for each transformer blocks, thereby leading to improved efficiency.

[4] $G_k = -\log(-\log(U_k)), U_k \sim U(0, 1)$ with the uniform distribution $U$.

[5] While we adopt learning-based routing, there exists various alternatives, such as Top-k selection (Zhou et al., 2022) and Random selection (Wang et al., 2022). Moreover, semantically-driven routing can be a promising approach. We leave the exploration of these routing variants as future work.

Table 1: Evaluation results of test accuracy (%) and speedup ratio on NLU tasks. The speedup ratio (denoted as **SpeedUp**) is computed by comparing with the backbone. The best and second best results are highlighted in **boldface** and underline, respectively.

| Method | SST-2 | | OpenbookQA | | QNLI | | MRPC | | STS-B | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. | SpeedUp | Acc. | SpeedUp | Acc. | SpeedUp | F1. | Speed | Pearson. | SpeedUp |
| Baseline | 94.1 | 1.00× | 59.6 | 1.00× | 91.3 | 1.00× | 88.1 | 1.00× | 88.9 | 1.00× |
| LTP (2022) | 93.3 | 1.31× | 59.4 | 1.25× | 90.2 | 1.21× | 87.5 | 1.21× | 88.4 | 1.21× |
| Transkimmer (2022) | 93.0 | 1.42× | 58.6 | 1.34× | 89.4 | 1.45× | 86.9 | 1.54× | 88.1 | 1.64× |
| LoT (2023) | 93.5 | **1.96**× | 59.1 | 1.45× | 90.4 | 1.86× | 87.2 | 1.82× | 87.8 | 1.42× |
| SPARSEFLOW | 93.3 | 1.85× | 59.2 | **1.58**× | 90.7 | **2.12**× | 87.5 | **2.41**× | 88.4 | **1.85**× |

## 4 Experiments

In this section, we experimentally demonstrate the efficacy of the proposed method. Specifically, we answer the following three questions through extensive experiments and analysis:

**Q1 (Efficiency)** Does SPARSEFLOW offer better efficiency than token pruning methods across a wide range of tasks? (§4.2)

**Q2 (Generality)** Can SPARSEFLOW be generally applicable to larger scales, different architectures, and even other modalities? (§4.3)

**Q3 (Insights)** What flow patterns and behaviors are learned from SPARSEFLOW? (§4.4)

### 4.1 Experimental Setups

**Baselines** As SPARSEFLOW is closely related to the token pruning, we mainly compare our method with following token pruning methods with a backbone model (denoted as Baseline in the tables): **LTP** (Kim et al., 2022) which utilizes the attention maps to eliminate the tokens; **Transkimmer** (Guan et al., 2022) that removes tokens with the learnable token predictors; **LoT** (Kim et al., 2023) that learns to skip per-token computations through gradient-based routers. We follow the original setups for each baseline. As for SPARSEFLOW, the hyper-parameters and ablation results including the effects of the number of mixtures ($K$) are described in Appendix.

**Tasks and Datasets** To confirm the broad applicability of the methods, we evaluate each baseline on both natural language understanding (NLU) and natural language generation (NLG) tasks. For NLU tasks, we evaluate each baseline on five tasks, which are Stanford Sentiment Treebank (SST-2) (Socher et al., 2013) for sentence classification, OpenbookQA (Mihaylov et al., 2018) for multiple-choice question answering, Question-answering NLI (QNLI) (Rajpurkar et al., 2016) for natural language inference, Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005) for paraphrasing task, and Semantic Textual Similarity (STS-B) (Cer et al., 2017). For NLG tasks, we perform summarization tasks on SamSum (Gliwa et al., 2019) and XSum (Narayan et al., 2018).

Table 2: Evaluation results of test accuracy (%) and speedup ratio on the NLG tasks. The speedup ratio (denoted as **SpeedUp**) is computed by comparing with the backbone. The best and second best results are highlighted in **boldface** and underlined, respectively.

| Method | SamSum | | XSum | |
|---|---|---|---|---|
| | RL | SpeedUp | RL | SpeedUp |
| Baseline | 39.1 | 1.00× | 28.0 | 1.00× |
| LTP (2022) | 38.8 | 1.51× | 27.6 | 1.49× |
| Transkimmer (2022) | 38.6 | 1.42× | 27.9 | 1.57× |
| LoT (2023) | 38.8 | 1.64× | 27.4 | 1.71× |
| SPARSEFLOW | 38.9 | **1.75**× | 27.8 | **1.92**× |

**Models** For NLU task, we compare all baselines on the $\text{BERT}_{\text{Large}}$ (Devlin et al., 2019) due to the suitability of the encoder model for understanding tasks. For the generation tasks, we utilize $\text{T5}_{\text{Small}}$ (Raffel et al., 2020) which consists of the encoder and decoder transformers. To confirm the scalability and general applicability, we additionally perform experiments on the larger models, which includes $\text{RoBERTa}_{\text{Large}}$ (encoder, 337M parameters) (Liu et al., 2019), $\text{T5}_{\text{Large}}$ (encoder-decoder, 770M parameters), and $\text{GPT2}_{\text{XL}}$ (decoder, 1.5B parameters) (Radford et al., 2019). Note that, for models comprising a decoder transformer, such as T5 and GPT-2, SPARSEFLOW is applied to the context token processing components (i.e., the encoder in T5 and conditional token computation in GPT-2) because our primary focus is on the quadratic costs

associated with encoding. The exploration to the decoding phase is left as future work.

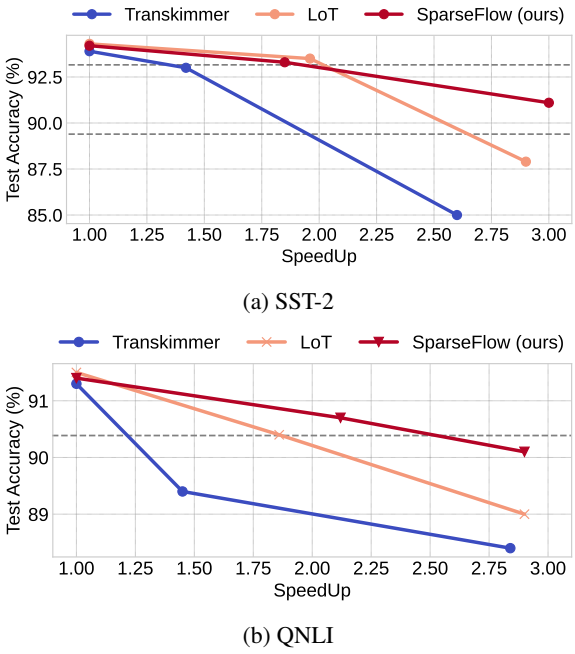
**Evaluation** To evaluate the efficiency of each baseline in maintaining task accuracy while reducing computational demand, in every experiment, we report the speedup gains achieved by each method up to the point where it retains 99% of the original accuracy for each task. Following the previous work (Guan et al., 2022; Kim et al., 2023), these speedup gains are calculated as the ratio of reduced FLOPs compared to the backbone model.

### 4.2 Main Results

**Comparison to baselines** Table 1 shows the overall comparison results. It is noticeable that SPARSEFLOW achieves substantial speedup gains better than other token pruning methods on NLU tasks (i,e., best speedups among four out of five datasets). This result demonstrates the efficacy in sparsifying the dense information flows. In addition, the evaluation result of the generation tasks in Table 2 presents that the strength of SPARSEFLOW can be generalized to generation tasks. Regarding the superiority to token pruning methods, we believe that a significant contributing factor lies in the preservation of all token information while decreasing computations, preventing the model from losing semantic information (Zhong et al., 2023). Furthermore, it is worth noting that the minimal overhead compared to other approaches also contributes to its superior efficiency. These overall results clearly verify our hypothesis that sparsifying information flow works quite well to improve the efficiency in transformers.

**Trade-off between Accuracy and Speedup** In Figure 2, we also analyze the trade-off between the task accuracy and computational costs while retaining 95% and 99% of original task accuracy. Our results suggest that SPARSEFLOW shows a superior performance over a wide range of speedup gains with a smoother decline. This result also supports the efficacy of our method, demonstrating that the proposed method better preserves the task-significant information than other token pruning methods.

**Throughput Increase and Memory Reduction** To confirm the actual speedup on a specific GPU hardware, we measure the throughput increase and memory reduction achieved by SPARSEFLOW. Table 3 presents the improvement in each metric on an NVIDIA RTX 2080Ti. Our results provide that SPARSEFLOW indeed brings speedup gains in actual GPU setups while lowering the memory requirements, similar to the reduced ratio of FLOPs. Such reduction implies that the memory and computationally-constrained environments can indeed benefit from SPARSEFLOW.




(a) SST-2




(b) QNLI

Figure 2: Trade-off between speedup gains and test accuracy on two representative tasks.

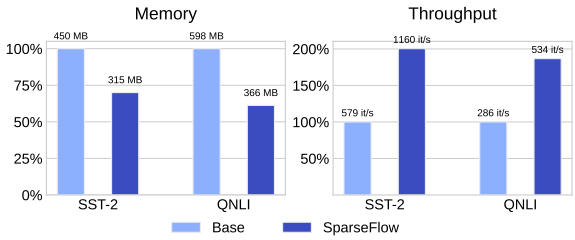



Figure 3: Throughput increase and memory reduction compared to the backbone. Evaluations are conducted in a single NVIDIA RTX 2080Ti GPU.

### 4.3 General Applicability of SPARSEFLOW

**Different Modality** SPARSEFLOW can be easily applicable to any transformer-based models with minor modifications. We thus investigate whether the effectiveness of SPARSEFLOW can be generalized to the different modalities other than text. We specifically test a multimodal (vision-and-language) transformer, as such multi-modality often necessitates longer sequence lengths to en-

Table 3: Evaluation results of test accuracy (%) and speedup ratio on the vision-language tasks. Best results are highlighted in **boldface**.

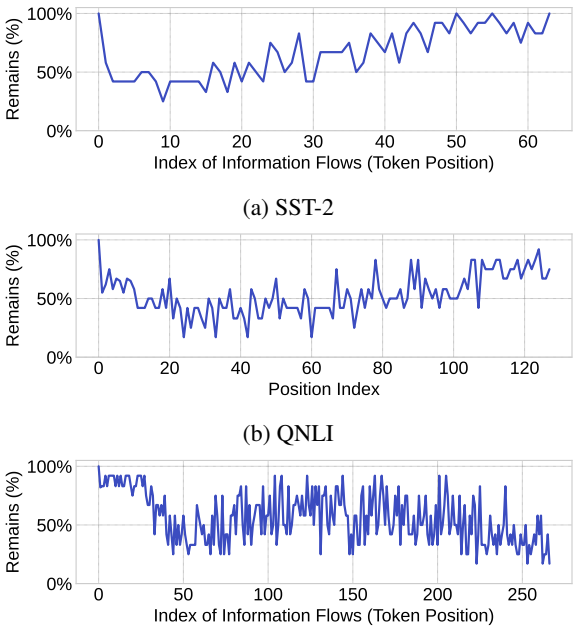
| Method | VQAv2 | | NLVR2 | |
|---|---|---|---|---|
| | Acc. | SpeedUp | Acc. | SpeedUp |
| ViLT | 70.7 | 1.00× | 75.5 | 1.00× |
| PuMer (2023) | 68.9 | 1.84× | **74.9** | **1.84**× |
| SPARSEFLOW | **69.5** | **1.92**× | 74.6 | 1.78× |

Table 4: Evaluation results of test accuracy (%) and speedup ratio with different architectures and larger scales. In this experiment, $\text{RoBERTa}_{\text{Large}}$, $\text{T5}_{\text{Large}}$, and $\text{GPT2}_{\text{XL}}$ have 330M, 770M and 1.5B parameters.

| Method | SST-2 | | QNLI | |
|---|---|---|---|---|
| | Acc. | SpeedUp | Acc. | SpeedUp |
| $\text{RoBERTa}_{\text{Large}}$ | 95.6 | 1.00× | 92.9 | 1.00× |
| LoT (2023) | 95.2 | 1.56× | 92.3 | 1.49× |
| SPARSEFLOW | 95.1 | 2.21× | 92.0 | 2.14× |
| $\text{T5}_{\text{Large}}$ | 95.9 | 1.00× | 94.1 | 1.00× |
| LoT (2023) | 94.8 | 1.54× | 93.4 | 1.65× |
| SPARSEFLOW | 95.3 | 1.86× | 93.8 | 1.81× |
| $\text{GPT2}_{\text{XL}}$ | 95.4 | 1.00× | 91.1 | 1.00× |
| LoT (2023) | 94.2 | 1.21× | 90.7 | 1.43× |
| SPARSEFLOW | 94.7 | 1.67× | 90.5 | 1.61× |

code various types of inputs (e.g., image patches and textual tokens). For this purpose, we employ the Vision-and-Language Transformer (ViLT) as the backbone model. Evaluations are conducted on Visual Question Answering (VQAv2) (Goyal et al., 2017) and Natural Language for Visual Reasoning (NLVR2) (Suhr et al., 2019). VQAv2 is to predict the answers for the questions requiring an understanding of vision, language, and commonsense knowledge. NLVR2 is to predict whether the given sentence is true about two input images. To confirm the competitiveness of SPARSEFLOW, we compare ours with PuMer (Cao et al., 2023), a token pruning method specifically designed for vision and language tasks.

Table 3 shows the evaluation result on two tasks. Interestingly, without a modality-specific design, SPARSEFLOW demonstrates substantial speedup gains on both tasks, comparable to or even surpassing those of modality-specific methods. This result suggests that sparsifying dense information is a generally effective strategy for enhancing the efficiency of pre-trained transformers. In Section 4.4, we further explore how SPARSEFLOW works differently on image and text representations.




(a) SST-2

(b) QNLI

(c) VQAv2

Figure 4: Average utilization ratio of each position in three different tasks. This ratio is the average of the remaining information flows across all layers. In VQAv2, tokens from 0 to 40 correspond to the textual tokens.

**Different Scales and Architectures** As the size of pre-trained models keeps increasing recently, it is important to confirm whether the proposed method can be scaled to larger models. We therefore examine the scalability of SPARSEFLOW by applying ours to $\text{RoBERTa}_{\text{Large}}$ (Liu et al., 2019), $\text{T5}_{\text{Large}}$ (Raffel et al., 2020), and $\text{GPT}_{\text{XL}}$ (Radford et al., 2019). Table 4 provides the evaluation results on this setup. This shows that the performance trend in ther other architecture is similar to the main experiments, demonstrating the general applicability of SPARSEFLOW to larger and even different architectures.

### 4.4 Learned Patterns of SPARSEFLOW

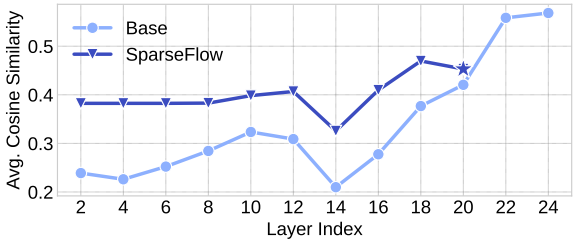
**Sparse Distribution** SPARSEFLOW learns to remain crucial information flow throughout the training. To investigate whether the remaining information flows have interpretable patterns or not, we analyze the distribution of sparse information flows through the lens of token positions[6] in Figure 4. In the language understanding tasks (i.e., SST-2 and QNLI), we observe that initial positions tend

[6]In Appendix, we include the analysis on the remaining information flows across different layers.

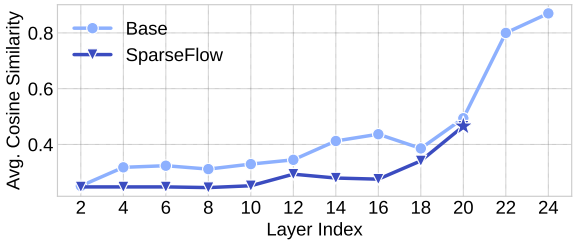
to be more pruned than the later positions. We believe the frequent optimization on initial positions contributes this result since the parameters of the latter positions are rarely optimized during training (More discussion can be found in the Limitation part). We also observe the distinct patterns from the vision-language tasks (i.e., VQAv2). It shows that image tokens (positions after 40) are more pruned than text tokens (positions before 40), indicating the significant of textual tokens to solve the given tasks or the potential existence of short-cut in textual inputs. As for the pruned positions, we also find that the pruned image tokens are noticeably located in the near start and end positions which are corresponding to the edge of an image. This suggests that SPARSEFLOW can learn the different behavior depending on the modality and task.

**Detailed examples of SPARSEFLOW** To get a deeper understanding of the SPARSEFLOW's behaviors, we analyze how the information of task-significant tokens are flowed through the remaining positions of the information flows. Here, we trace the similarity between the task-significant tokens and remaining tokens (i.e., remaining information flows) on the example of "*the movie fails to live up to the sum of its parts*" from SST-2 (sentiment analysis) where the task-significant word is *fails*. As token representations in transformers grow increasingly similar across layers, the degree of similarity between important tokens and others can indicate the effectiveness with which task-specific features are transferred throughout the network. Figure 5 shows the similarity changes compared to the base model. Interestingly, we observe that the similarity of the task-significant token (i.e., *fails*) to remaining tokens is higher than that of the base model. It implies that the the sparsified patterns (i.e., remaining tokens) can convey task-significant features well. In contrast, the similarity to the insignificant word (e.g., *the*) also support our findings by showing a similar or even lower similarity compared to the base model on the less significant word.

We also see the sparsified positions on specific images to see how the model differently performs on image modality. Here we sample the image from VQAv2, and Figure 6 shows the pruned positions (i.e., image patch) on the given image. Noticeably, in the lower layers, it is evident that the removed information flows are primarily found from the edge positions. In other words, the model learns to decide that tokens in center positions are crucial




(a) Similarity with a word '*fails*'




(b) Similarity with a word '*the*'

Figure 5: Similarity between the selected words (*fails, the*) and words in masked positions over different layers. The example is "*the movie fails to live up to the sum of its parts*" from SST-2 (sentiment analysis). The star mark in the graph indicates that all information flows are removed on this example in SPARSEFLOW.

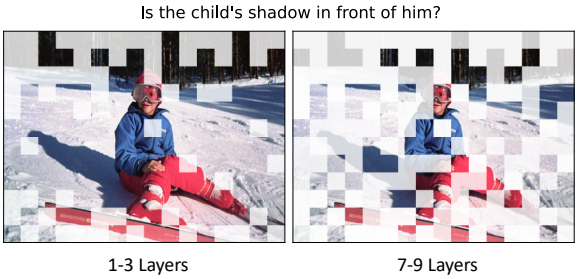



Figure 6: Removed information flows from lower layers (1∼3) and upper layers (7∼9). On these images, the white box on images indicates the removed information flows at the imagee patches.

to understand the image. This result shows that the model automatically learns *center bias* (Tseng et al., 2009) in image processing, and SPARSEFLOW is able to learn modality-specific information during the training.

## 5 Conclusion

In this paper, we have introduced SPARSEFLOW, a novel efficient method that sparsifies the dense information flow to improve efficiency in transformers. To this end, we parameterize the information flow of the pre-trained models and learn to sparsify the information flow such that only the necessary flows are preserved. We have performed extensive

experiments on diverse benchmarks, scaled models, and modalities to verify the efficacy and general applicability. Comprehensive result convincingly demonstrates that sparsifying the information flows brings significant speedup gains without compromising the task accuracy. Moreover, we have also shown that SPARSEFLOW can learn modality-specific information automatically (e.g., center bias in an image processing).

## Limitations

While SPARSEFLOW enables the streamlined inference of pre-trained language models, there exist potential limitations. First, since the sparsified patterns are pre-determined during the training, the effectiveness of SPARSEFLOW can be limited to the samples deviating from the training distributions, such as those longer than the maximum sequence. However, we believe that composing sufficiently diverse inputs can mitigate the limitation. Second, in this work, we mainly focus on the encoder part of the transformers and the context encoding part of the decoder model, which necessitate quadratic computations in the forward pass. Therefore, the efficiency of SPARSEFLOW during the decoding phase is yet to be verified. However, we believe that integrating SPARSEFLOW with KV-caching tricks effectively reduces the computation costs with memory requirements. We leave this direction of improvement as a promising avenue for future research.

## Acknowledgment

This work was supported by the Basic Research Program through the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2021R1A2C3010430) and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2019-II190079, Artificial Intelligence Graduate School Program (Korea University)).

## References

Samira Abnar and Willem H. Zuidema. 2020. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4197. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Qingqing Cao, Bhargavi Paranjape, and Hannaneh Hajishirzi. 2023. Pumer: Pruning and merging tokens for efficient vision language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 12890–12903. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko E Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and cross-lingual focused evaluation. In *Proceedings of The Eleventh International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *CoRR*, abs/2204.02311.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop*

*on Paraphrasing*, pages 9–16. Asian Federation of Natural Language Processing.

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

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 55–65. Association for Computational Linguistics.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 70–79. Association for Computational Linguistics.

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan T. Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. Power-bert: Accelerating BERT inference via progressive word-vector elimination. In *Proceedings of the International Conference on Machine Learning*, volume 119, pages 3690–3699. PMLR.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6904–6913. IEEE.

Yue Guan, Zhengyi Li, Jingwen Leng, Zhouhan Lin, and Minyi Guo. 2022. Transkimmer: Transformer learns to layer-wise skim. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 7275–7286. Association for Computational Linguistics.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *Proceedings of the Fifth International Conference on Learning Representations*. OpenReview.net.

Gyuwan Kim and Kyunghyun Cho. 2021. Length-adaptive transformer: Train once with length drop, use anytime with search. In *The Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 6501–6511. Association for Computational Linguistics.

Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, and Kurt Keutzer. 2022. Learned token pruning for transformers. In *Proceddings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 784–794. ACM.

Yeachan Kim, Junho Kim, Jun-Hyung Park, Mingyu Lee, and SangKeun Lee. 2023. Leap-of-thought: Accelerating transformers via dynamic token routing. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15757–15769. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9992–10002. IEEE Computer Society.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

Ali Modarressi, Hosein Mohebbi, and Mohammad Taher Pilehvar. 2022. Adapler: Speeding up inference by adaptive length reduction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 1–15. Association for Computational Linguistics.

Shashi Narayan, Shay Cohen, and Maria Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807. Association for Computational Linguistics.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits

of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1631–1642. Association for Computational Linguistics.

Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. 2019. A corpus for reasoning about natural language grounded in photographs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6418–6428.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170. Association for Computational Linguistics.

Po-He Tseng, Ran Carmi, Ian GM Cameron, Douglas P Munoz, and Laurent Itti. 2009. Quantifying center bias of observers in free viewing of dynamic natural scenes. *Journal of vision*, 9(7):4–4.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Adamix: Mixture-of-adaptations for parameter-efficient model tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, pages 5744–5760. Association for Computational Linguistics.

Deming Ye, Yankai Lin, Yufei Huang, and Maosong Sun. 2021. TR-BERT: dynamic token reduction for accelerating BERT inference. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5798–5809. Association for Computational Linguistics.

Zhen Zhang, Wei Zhu, Jinfan Zhang, Peng Wang, Rize Jin, and Tae-Sun Chung. 2022. PCEE-BERT: Accelerating BERT inference via patient and confident early exiting. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 327–338. Association for Computational Linguistics.

Qihuang Zhong, Liang Ding, Juhua Liu, Xuebo Liu, Min Zhang, Bo Du, and Dacheng Tao. 2023. Revisiting token dropping strategy in efficient BERT pretraining. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 10391–10405. Association for Computational Linguistics.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit. In *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V. Le, and James Laudon. 2022. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems*, volume 35, pages 7103–7114.

## A Sparse Distribution of SPARSEFLOW

In Figure 7, we additionally analyze the number of remaining information flows across layers. We first notice that SPARSEFLOW learns to prune more information flows located in the deeper layers. This indicates the large redundancy in the deeper layer, and it is closely aligned with the previous finding that token representations get similar through multiple layers (Ethayarajh, 2019; Goyal et al., 2020).

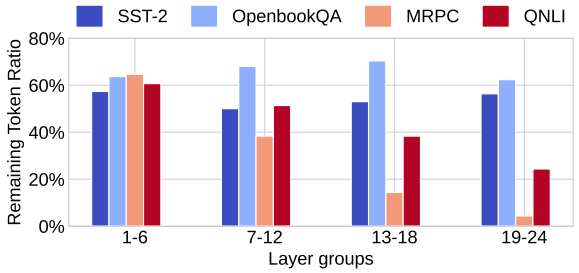



Figure 7: Remaining token distribution across various layer groups (grouped by three) and datasets.

## B Hyper-parameter Setup of SPARSEFLOW

In our experiments, we control the sparsity of SPARSEFLOW through adjusting $\alpha$ in Eq. (8), and we find that $\alpha = 0.3$ generally works well. For the number of sparsified patterns in MoE, we empirically set $K$ to 3 as it generally performs well over the experiments.

Table 5: Ablation study of SPARSEFLOW, and 'w/o' indicates the model without the corresponding component.

| Method | SST-2 | | QNLI | |
|---|---|---|---|---|
| | Acc. | Speed | Acc. | Speed |
| SPARSEFLOW (ours) | 93.3 | 1.85× | 90.7 | 2.12× |
| w/o Consistency | 93.1 | 1.78× | 90.3 | 1.81× |
| w/o Target Ratio | 93.5 | 1.59× | 90.1 | 1.92× |
| w/ Mixture ($K$=1) | 93.1 | 1.69× | 90.3 | 1.91× |
| w/ Mixture ($K$=2) | 92.8 | 1.84× | 90.1 | 1.95× |
| w/ Mixture ($K$=3) | 93.3 | 1.85× | 90.7 | 2.12× |
| w/ Mixture ($K$=4) | 93.1 | 1.83× | 90.2 | 2.07× |

## C Ablation on SPARSEFLOW

To confirm what components of SPARSEFLOW are crucial to achieve better efficiency, we perform an ablation study of individual components, which are consistency regularization (Eq. (9)) and setting a target ratio (Eq. (8)) and different configurations of MoE. Table 5 shows the ablation results. Consistent with other experiments, we maintained the 99% original accuracy of each method to evaluate speedup gains. The findings indicate that omitting consistency regularization and the target ratio results in reduced speedup gains, thereby empirically justifying the importance of each component. In terms of MoE patterns, we observed that integrating MoE into SPARSEFLOW significantly enhances speedup gains. These results confirm the effectiveness of our proposed method in accommodating diverse inputs, which in turn leads to enhanced efficiency.

## D Additional Experiments with Large Models

we have performed additional experiments with larger models on other datasets (MRPC and STS-B). Table 6 shows the evaluation results when applying LoT (Kim et al., 2023) and the proposed method to $T5_{Large}$ and $GPT2_{XL}$. Similar to the experiments in others, SPARSEFLOW shows superior trade-off between accuracy and speedups compared to recent token pruning method, highlighting the scalability of the proposed method.

Table 6: Evaluation results of test accuracy (%) and speedup ratio with larger scales. In this experiment, $T5_{Large}$ and $GPT2_{XL}$ have 770M and 1.5B parameters.

| Method | MRPC | | STS-B | |
|---|---|---|---|---|
| | F1. | Speed | Pearson. | Speed |
| $T5_{Large}$ | 95.9 | 1.00× | 94.1 | 1.00× |
| LoT (2023) | 94.8 | 1.54× | 93.4 | 1.65× |
| SPARSEFLOW | 95.3 | 1.86× | 93.8 | 1.81× |
| $GPT2_{XL}$ | 95.4 | 1.00× | 91.1 | 1.00× |
| LoT (2023) | 94.2 | 1.21× | 90.7 | 1.43× |
| SPARSEFLOW | 94.7 | 1.67× | 90.5 | 1.61× |