Flowformer: Linearizing Transformers with Conservation Flows

Haixu Wu [1] **Jialong Wu** [1] **Jiehui Xu** [1] **Jianmin Wang** [1] **Mingsheng Long** [1]

Abstract

Transformers based on the attention mechanism have achieved impressive success in various areas. However, the attention mechanism has a quadratic complexity, significantly impeding Transformers from dealing with numerous tokens and scaling up to bigger models. Previous methods mainly utilize the similarity decomposition and the associativity of matrix multiplication to devise linear-time attention mechanisms. They avoid degeneration of attention to a trivial distribution by reintroducing inductive biases such as the locality, thereby at the expense of model generality and expressiveness. In this paper, we linearize Transformers free from specific inductive biases based on the flow network theory. We cast attention as the information flow aggregated from the sources (values) to the sinks (results) through the learned flow capacities (attentions). Within this framework, we apply the property of *flow conservation* into attention and propose the Flow-Attention mechanism of linear complexity. By respectively conserving the incoming flow of sinks for *source competition* and the outgoing flow of sources for *sink allocation*, Flow-Attention inherently generates informative attentions without using specific inductive biases. Empowered by the Flow-Attention, Flowformer yields strong performance in linear time for wide areas, including long sequence, time series, vision, natural language, and reinforcement learning. The code and settings are available at this repository: https://github.com/thuml/Flowformer.

1. Introduction

Recently, Transformers (Vaswani et al., 2017) have shown immense capability in sequential modeling and been widely used in various areas, such as natural language processing (Devlin et al., 2019; Liu et al., 2019; Brown et al., 2020), computer vision (Dosovitskiy et al., 2021; Liu et al., 2021), time series analysis (Zhou et al., 2021; Wu et al., 2021) and reinforcement learning (Chen et al., 2021b; Janner et al., 2021). Based on attention mechanisms, Transformers can learn the relation between each pair of tokens in a sequence.

However, suffering from the quadratic complexity of pairwise relation modeling, it is computationally prohibitive for Transformers to deal with long sequences and scale up to bigger models. To tackle this essential obstacle for foundation models (Bommasani et al., 2021), efficient and linear Transformers have been explored. One category of methods attempts to utilize the sparsity to reduce the model captured relations (Child et al., 2019; Vyas et al., 2020; Zaheer et al., 2020). By substituting the dense matrix to a sparse version, these models can obtain a lower complexity but inevitably sacrifice some valuable information, leading to the trade-off dilemma between efficiency and performance. Another mainstream category tries to abandon the computation-consuming query-key multiplication in the attention mechanism. The typical method is to substitute or approximate the softmax-based similarity in Transformers. For example, Linear Transformer (Katharopoulos et al., 2020) introduces the decomposition method for similarity calculation and further bypasses the query-key multiplication through the associativity of matrix multiplication. However, without using the softmax function, these methods cannot guarantee the distinguishability of attention. This may result in near-uniform attention of each token to all other tokens, namely the degenerated attention, which damages the effectiveness of the attention mechanism. Although some works try to incorporate the concentration property to avoid the trivial attention (Luo et al., 2021; Zhen et al., 2022), they have to reintroduce specific inductive biases to Transformers, such as the locality in sequence, sacrificing the model generality. Thus, *how to simultaneously obtain the non-trivial attention and maintain the generality as the canonical attention* is the key challenge in the advance of linearizing Transformers.

Previous works demonstrate that the softmax function is essential to avoid the trivial attention (Choromanski et al., 2021; Peng et al., 2021). It is well-known that the softmax function is originally proposed as a differentiable generalization of the "*winner-take-all*" picking maximum operation (Bridle, 1989). Thus, the softmax function can introduce

[1]School of Software, BNRist, Tsinghua University. Haixu Wu <whx20@mails.tsinghua.edu.cn>. Correspondence to: Mingsheng Long <mingsheng@tsinghua.edu.cn>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

the *competition* among tokens in the attention mechanism, enforcing higher attention only to the essential tokens and thereby avoiding near-uniform attention weights. Based on this insight, it is a natural solution to empower transformers with built-in competition property to generate informative attention that guarantees the modeling capability. However, the competition mechanism is irrealizable for linear Transformers because the attention weights to compete will incur the quadratic complexity. To tackle the aforementioned problems, we attempt to reconstruct the attention mechanism from a new view of flow network (Ahuja et al., 1993), where the competition property is naturally achieved. Note that a flow network is a directed graph with information flows from one node to another under the constraint of flow capacity. Correspondingly, the attention mechanism can be reformulated as aggregating the information from sources (i.e., values) to sinks (i.e., results) through the learned flow capacities (i.e., attentions). We further find that by conserving the incoming flow capacity for each sink, the outgoing flow capacities of sources will compete with each other. And by conserving the outgoing flow capacity of sources, we can also obtain the competed incoming flow capacities of sinks. Thus, benefiting from the *flow conservation* in flow network, the competition mechanism can be accomplished without specific inductive biases.

Based on the above insights, we introduce the *flow conservation* to the attention mechanism and further propose the *Flow-Attention* mechanism, which can avoid the trivial attention and simultaneously be free from specific inductive biases. Technically, by conserving the incoming flow of sinks (i.e., results), the *source competition* mechanism is accomplished and then applied for the non-trivial information aggregation. After the information aggregation, the *sink allocation* mechanism is obtained by conserving the outgoing flow of sources (i.e., values) and then applied to filter the aggregated information. Empowered by the Flow-Attention, Flowformer in linear complexity achieves competitive or better performance as the canonical Transformer in extensive areas. The contributions are summarized as follows:

- This paper analyzes the attention mechanism from the new view of the flow network. By introducing the *flow conservation* to both the source and sink aspects, the competition among tokens is naturally achieved.
- Based on flow conservation, we propose the *Flow-Attention* with *source competition* and *sink allocation* mechanisms, which can avoid degenerated attentions without incorporating specific inductive biases.
- Empowered by Flow-Attention, our proposed *Flowformer* yields strong performance in linear time on five benchmarks, covering wide areas: long sequence, language, vision, time series and reinforcement learning.

2. Preliminaries

2.1. General View of Attention Mechanism

The attention mechanism is the key component of Transformers (Vaswani et al., 2017), which can be used to explore the underlying relations among tokens and adaptively aggregate valuable information. The input of attention mechanism contains three parts: queries $\mathbf{Q} \in \mathbb{R}^{n\times d}$, keys $\mathbf{K} \in \mathbb{R}^{m\times d}$ and values $\mathbf{V} \in \mathbb{R}^{m\times d}$. The i-th row of the result $\mathbf{R}$ of the attention mechanism can be calculated as follows:

$$\mathbf{R}_i = \sum_{j=1}^{m} \frac{S(\mathbf{Q}_i, \mathbf{K}_j)}{\sum_{j'=1}^{m} S(\mathbf{Q}_i, \mathbf{K}_{j'})} \mathbf{V}_j, \ i \in \{1, \cdots, n\}, \quad (1)$$

where $*_i$ denotes the i-th row of matrix $*$. $S(\mathbf{Q}_i, \mathbf{K}_j)$ calculates the similarity between queries and keys. Thus, the attention mechanism is to aggregate the information from values based on the attention map calculated from queries and keys. In the canonical Transformer (Vaswani et al., 2017), $S(\mathbf{Q}_i, \mathbf{K}_j)$ is set as $\exp(\mathbf{Q}_i \mathbf{K}_j^{\mathsf{T}})$, corresponding to the softmax function. The softmax function introduces the competition between similarity weights, which is essential to obtain a non-trivial attention. However, because of the calculation of $\mathbf{Q}\mathbf{K}^{\mathsf{T}}$, the computation complexity of Eq. (1) in vanilla Transformer is quadratic in the sequence length, concretely $\mathcal{O}(nmd)$, resulting in the core limitation.

2.2. Efficient and Linear Transformers

To break through the complexity limitation of the canonical attention mechanism, efficient and linear Transformers are widely explored. Categorized by the operation to the attention map, the paradigms roughly involve the similarity-decomposition and attention-sparsification methods.

Similarity-decomposition methods It is notable that the quadratic complexity of canonical attention is caused by the calculation of $\mathbf{Q}\mathbf{K}^{\mathsf{T}}$. This computation-consuming operation is indispensable because of the exponential definition of $S(\cdot, \cdot)$, while similarity-decomposition methods try to linearize the attention by utilizing the decomposition of $S(\cdot, \cdot)$ and the associativity of matrix multiplication. Technically, if $S(\cdot, \cdot)$ can be decomposed as the inner-product between the non-linear projections $\phi(\cdot)$ of queries and keys,

$$S(\mathbf{Q}_i, \mathbf{K}_j) = \langle \phi(\mathbf{Q}_i), \phi(\mathbf{K}_j) \rangle = \phi(\mathbf{Q}_i)\phi(\mathbf{K}_j)^{\mathsf{T}}, \quad (2)$$

then we can adopt associativity to reduce complexity. Under this condition, Eq. (1) will be reformulated as follows:

$$\begin{aligned} \mathbf{R}_i &= \sum_{j=1}^{m} \frac{\phi(\mathbf{Q}_i)\phi(\mathbf{K}_j)^{\mathsf{T}}}{\sum_{j'=1}^{m} \phi(\mathbf{Q}_i)\phi(\mathbf{K}_{j'})^{\mathsf{T}}} \mathbf{V}_j \\ &= \frac{\phi(\mathbf{Q}_i) \sum_{j=1}^{m} \phi(\mathbf{K}_j)^{\mathsf{T}} \mathbf{V}_j}{\phi(\mathbf{Q}_i) \sum_{j=1}^{m} \phi(\mathbf{K}_j)^{\mathsf{T}}}, \ i \in \{1, \cdots, n\}, \end{aligned} \quad (3)$$

in which the direct calculation of $\mathbf{Q}\mathbf{K}^{\mathsf{T}}$ is avoided and replaced by the multiplication of keys and values, namely

$\phi(\mathbf{K})^{\top}\mathbf{V}$. Correspondingly, the complexity is $\mathcal{O}(nd^2)$ and linear in sequence length. One typical instance proposed by Linear Transformer (Katharopoulos et al., 2020) is to set the non-linear projection $\phi(\cdot)$ as $\text{elu}(\cdot)+1$ using the exponential linear unit. However, it is hard for Linear Transformer to avoid degenerated attention without the softmax function. Thus, RFA (Peng et al., 2021) and Performer (Choromanski et al., 2021) adopt the random Fourier features (Rahimi & Recht, 2007) and positive random features to approximate the softmax respectively. But these two methods suffer from the approximate error and have to choose specific kernels to meet the theoretical guarantee of approximation. Besides, SOFT (Lu et al., 2021), YOSO (Zeng et al., 2021) and Nyströmformer (Xiong et al., 2021) approximate softmax function by Nyström method, while they will bring tedious iterations into calculation and cannot implement the causal version of attention for autoregressive tasks. Recently, cosFormer (Zhen et al., 2022) decomposes the similarity metric based on the decomposition of cosine function, where $S(\mathbf{Q}_i, \mathbf{K}_j)$ is set as $\phi(\mathbf{Q}_i)\phi(\mathbf{K}_j)^{\top}\cos(\frac{\pi}{2}\times\frac{i-j}{M})$ and M is a hyperparameter. Although it can concentrate the learned attention, the design in cosFormer explicitly includes the locality inductive bias in temporal dimension, overlooking the spatial position in vision, thereby sacrificing the generality.

Attention-sparsification methods This paradigm does not change the similarity metric $S(\cdot,\cdot)$ in the canonical attention but attempts to reduce the model captured relations. Typically, Sparse Transformer (Child et al., 2019) only calculates the similarity between pre-selected query-key pairs, thereby able to obtain the sparse attention matrix for efficiency. Based on the low-rank hypothesis, Linformer (Wang et al., 2020) adopts the projector to map the queries and keys to a low dimension. Reformer (Kitaev et al., 2020) replaces the dense dot-product attention with the locality-sensitive hashing for similarity calculation. Clustered attention (Vyas et al., 2020) reduces the complexity by grouping the queries. BigBird (Zaheer et al., 2020) enhances the sparse attention mechanism with the global token to accomplish the more powerful information aggregation. Note that all the above methods sacrifice information utilization. Thus, they have to suffer the efficiency-performance dilemma.

Unlike prior methods, Flowformer adopts the similarity decomposition and brings the flow conservation into design, which can naturally achieve the competition among tokens to avoid trivial attentions. Thus, Flowformer escapes from the efficiency-performance dilemma and eliminates specific inductive biases, thereby empowering stronger generality.

3. Method

As aforementioned, the core of linearizing Transformers is to obtain the non-trivial attention and maintain the generality simultaneously. To break with the above challenges, we reanalyze the attention mechanism from a novel view of the flow network. Inspired by the flow network theory, we propose the *Flow-Attention* mechanism by conducting the flow conservation on both the source and sink aspects. This design can bring the competition mechanism to sources and the allocation mechanism to sinks, avoiding trivial attentions without incorporating specific inductive biases.

3.1. Attention Mechanism: A Flow Network View

From a new perspective, this paper attempts to reformulate the attention mechanism from the view of flow network (Ahuja et al., 1993). As stated in Eq. (1), the canonical attention mechanism presents the procedure that the information is aggregated from the values $\mathbf{V}$ to the results $\mathbf{R}$. And the aggregation weights (i.e., attention map) are calculated from the similarity between queries $\mathbf{Q}$ and keys $\mathbf{K}$.

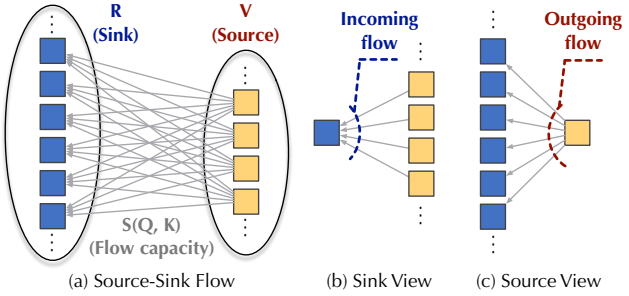


Figure 1. The flow network view for attention. The blue boxes (sinks) represent the results $\mathbf{R}$. The orange boxes (sources) represent the values $\mathbf{V}$. The gray arrows (flow capacity) denote the attention weights calculated from queries $\mathbf{Q}$ and keys $\mathbf{K}$.

Correspondingly, we can interpret the attention mechanism from the flow network view, as shown in Figure 1. Here, we take the results $\mathbf{R}$ as sinks, which have only incoming information flow for source aggregation, and the values $\mathbf{V}$ as sources, which have only outgoing flow for providing information to sinks. Following the similarity-decomposition method (Eq. (2)–(3)), we can define the similarity function $S(\mathbf{Q},\mathbf{K})$ as $\phi(\mathbf{Q})\phi(\mathbf{K})^{\top}$ to achieve linear complexity, where $\phi(\cdot)$ is the element-wise non-linear projection. Due to the property of flow network, we choose $\phi(\cdot)$ as a non-negative function to keep flow capacity positive.

As an important concept in flow network, the incoming and outgoing flow of each node can reflect the global interaction between each node and the whole flow network, which can provide valuable global information. Under the attention mechanism context, the flow capacity is calculated by queries and keys. Suppose we have n sinks and m sources. The incoming flow $I_i \in \mathbb{R}$ of the i-th sink and the outgoing flow $O_j \in \mathbb{R}$ of the j-th source can be calculated as:

$$I_i = \phi(\mathbf{Q}_i)\sum_{j=1}^{m}\phi(\mathbf{K}_j)^{\top},\ O_j = \phi(\mathbf{K}_j)\sum_{i=1}^{n}\phi(\mathbf{Q}_i)^{\top}, \quad (4)$$

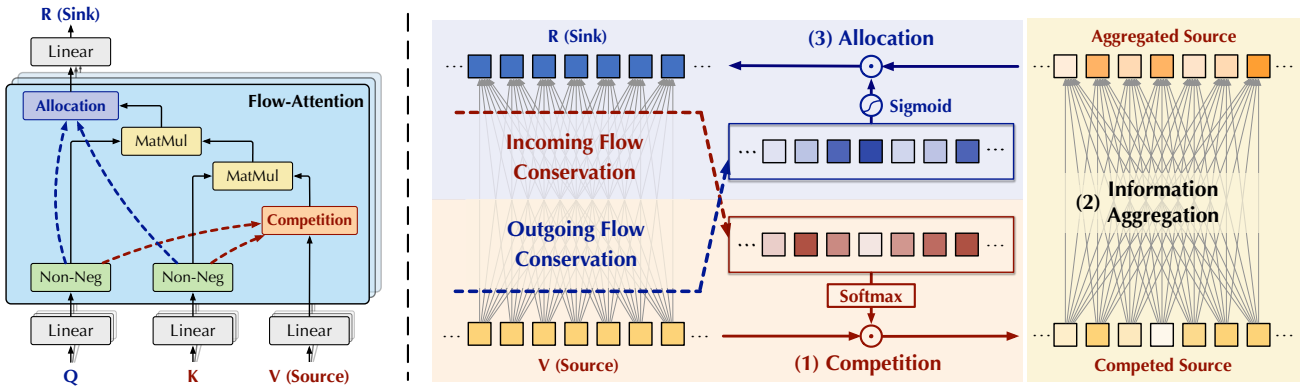


Figure 2. The overall procedure of Flow-Attention. The *source competition* mechanism (red dotted line) is obtained by incoming flow conservation to sinks. The *sink allocation* mechanism (blue dotted line) is accomplished by outgoing flow conservation to sources.

where $i \in \{1, \cdots, n\}$ and $j \in \{1, \cdots, m\}$. Note that the incoming flow $\mathbf{I} \in \mathbb{R}^{n\times 1}$ and the outgoing flow $\mathbf{O} \in \mathbb{R}^{m\times 1}$ can be calculated in linear complexity by Eq. (4). Then, we derive the Flow-Attention mechanism based on $\mathbf{I}$ and $\mathbf{O}$.

3.2. Flow-Attention Mechanism

Recall that the softmax function can obtain non-trivial attention by introducing *competition* among tokens, which is the indispensable component for avoiding the trivial attention. However, the competition mechanism is irrealizable for linear Transformers because the attention weights to compete will incur quadratic complexity. To tackle this dilemma, we propose the Flow-Attention by introducing the flow conservation into design in the spirit that "fixed resource will cause competition". Intuitively, as shown in Figure 3, outside the attention mechanism, the information of values ($\mathbf{V}$) is obtained from previous layer and the information of results ($\mathbf{R}$) will be provided to the next layer. Without loss of generality, we set the information incoming from or outgoing to other layers as the default value 1 for *fixing* the resource.

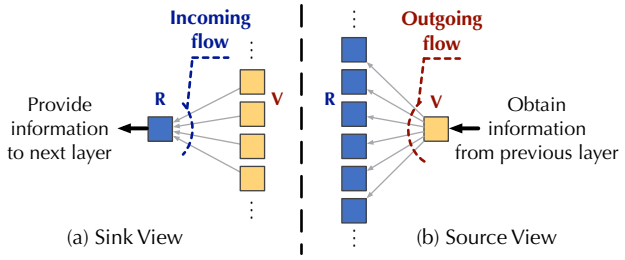


Figure 3. Information flow outside the attention mechanism.

In the deep model implementation, as shown in Figure 2, Flow-Attention adopts the non-negative and non-linear projection ϕ for the computation of flow capacity. Inspired by the flow network theory, we find that the competition mechanism is naturally achieved by introducing the *conservation* property in both source and sink aspects. Specifically, by conserving the incoming flow capacity for each sink as the default value 1, *i.e.* fixing the information provided to next layer (Figure 3), the outgoing flow capacities of sources will compete with each other since their sum is constrained as 1. Similarly, by conserving the outgoing flow capacity for each source as the default value 1, *i.e.* fixing the information obtained from previous layer (Figure 3), we can also obtain the competed incoming flow capacities of sinks. The above two conservation processes can be achieved by the following normalizing operations, which can well calibrate the subsequently calculated flow capacities:

$$\frac{\phi(\mathbf{K})}{\mathbf{O}}, \frac{\phi(\mathbf{Q})}{\mathbf{I}}, \tag{5}$$

where the ratio represents the element-wise division, $\frac{\phi(\mathbf{K})}{\mathbf{O}}$ is for the source conservation and $\frac{\phi(\mathbf{Q})}{\mathbf{I}}$ is for the sink conservation. Through the normalization, the conservation of flow capacity for each source and sink token are guaranteed, which can be verified by the following equations:

$$\begin{aligned}
\text{source-}j\text{:}\ & \frac{\phi(\mathbf{K}_j)^\top}{O_j} \sum_{i=1}^{n} \phi(\mathbf{Q}_i) = \frac{\sum_{i=1}^{n} \phi(\mathbf{Q}_i)\phi(\mathbf{K}_j)^\top}{O_j} = 1 \\
\text{sink-}i\text{:}\ & \frac{\phi(\mathbf{Q}_i)^\top}{I_i} \sum_{j=1}^{m} \phi(\mathbf{K}_j) = \frac{\sum_{j=1}^{m} \phi(\mathbf{K}_j)\phi(\mathbf{Q}_i)^\top}{I_i} = 1
\end{aligned} \tag{6}$$

which follows the same calculation as Eq. (4). The first equation is for the outgoing flow capacity of j-th source after the normalization $\frac{\phi(\mathbf{K})}{\mathbf{O}}$ and the second equation is for the incoming flow capacity of i-th sink after the normalization $\frac{\phi(\mathbf{Q})}{\mathbf{I}}$. Both capacities are equal to the default value 1.

After the conservation processes in both source and sink aspects, the competition is realized among the incoming

flow of sink tokens and the outgoing flow among source tokens respectively. The conserved information flows are:

$$\widehat{\mathbf{I}} = \phi(\mathbf{Q}) \sum_{j=1}^{m} \frac{\phi(\mathbf{K}_j)^{\mathsf{T}}}{O_j}, \ \widehat{\mathbf{O}} = \phi(\mathbf{K}) \sum_{i=1}^{n} \frac{\phi(\mathbf{Q}_i)^{\mathsf{T}}}{I_i}, \quad (7)$$

where $\widehat{\mathbf{I}} \in \mathbb{R}^{n\times 1}$ and $\widehat{\mathbf{O}} \in \mathbb{R}^{m\times 1}$ denote the capacity of conserved incoming flow and outgoing flow respectively.

Benefiting from the competition brought by the incoming flow conservation of sinks, $\widehat{\mathbf{O}}$ denotes the information provided by the sources under the fixed sum of flow capacity, which indicates the importance of each source. As for $\widehat{\mathbf{I}}$, it denotes the information obtained by each sink when the source outgoing capacity is fixed as 1, which reflects the capacity of aggregated information that each sink is allocated. Thus, as shown in Figure 2, we present the Flow-Attention based on the above conserved information flows, which includes the competition mechanism for sources and the allocation mechanism for sinks. The overall equations of the Flow-Attention mechanism can be formalized as follows:

$$\begin{aligned} \text{Competition: } \widehat{\mathbf{V}} &= \mathrm{Softmax}(\widehat{\mathbf{O}}) \odot \mathbf{V} \\ \text{Aggregation: } \mathbf{A} &= \frac{\phi(\mathbf{Q})}{\mathbf{I}} \left(\phi(\mathbf{K})^{\mathsf{T}} \widehat{\mathbf{V}}\right) \\ \text{Allocation: } \mathbf{R} &= \mathrm{Sigmoid}(\widehat{\mathbf{I}}) \odot \mathbf{A}, \end{aligned} \quad (8)$$

where $\odot$ denotes the element-wise multiplication. Note that both Softmax and Sigmoid can be computed in linear time. $\widehat{\mathbf{V}} \in \mathbb{R}^{m\times d}$ represents the competed sources, which is non-trivially re-weighted based on the incoming flow conservation. $\mathbf{A} \in \mathbb{R}^{m\times d}$ is the aggregated source information and calculated by the associativity of matrix multiplication. After the allocation mechanism with $\widehat{\mathbf{I}}$ to filter the incoming flow capacity for each sink, we can obtain the results $\mathbf{R} \in \mathbb{R}^{n\times d}$ of Flow-Attention. With above designs, Flow-Attention involves the competition in both source and sink aspects, thereby able to avoid trivial attention efficiently. More implementation details of both the normal and causal versions are provided in Appendix A.

Further, by replacing the attention mechanism in the Transformer family (Vaswani et al., 2017) with Flow-Attention, we can obtain the Flowformer without changing other designs but empower previous models with linear complexity.

Note that both the competition and allocation mechanisms are conducted based on the flow capacity directly calculated from queries and keys. Thus, different from cosFormer (Zhen et al., 2022) or other linear variants, Flowformer is without specific inductive bias, which empowers our model with great generality. The calculation of Eq. (4) and (7) is in linear complexity with respect to the sequence length. By further utilizing the associativity of matrix multiplication, Flow-Attention can be accomplished in linear complexity.

4. Experiments

To testify the effectiveness and generality of Flowformer, we extensively experiment on five well-established benchmarks, covering long sequence modeling, language processing, computer vision, time series and reinforcement learning. As shown in Table 1, the tasks on language modeling and reinforcement learning can verify the performance of causal-version Flow-Attention. See Appendix A.3 for more details.

Table 1. Summary of experiment benchmarks.

BENCHMARKS	TASK	VERSION	LENGTH
LRA (2020C)	SEQUENCE	NORMAL	1000∼4000
WIKITEXT (2017)	LANGUAGE	CAUSAL	512
IMAGENET (2009)	VISION	NORMAL	49∼3136
UEA (2018)	TIME SERIES	NORMAL	29∼1751
D4RL (2020)	OFFLINE RL	CAUSAL	60

4.1. Long Sequence Modeling

Setup. Long-Range Arena (LRA, Tay et al. 2020c) is a benchmark specially designed for efficient Transformers with long input sequence. It contains five different tasks: long sequence equation calculation (ListOps, Nangia & Bowman 2018), byte-level text classification (Text, Maas et al. 2011), document retrieval with the ACL Anthology Network (Retrieval, Radev et al. 2013), image classification based on the pixel sequence on CIFAR-10 (Image, Krizhevsky 2009) and long-range spatial dependencies discovery of images (Pathfinder, Linsley et al. 2018). For fair comparison, we follow the official implementation and experiment protocol of Long-Range Arena in Jax (Bradbury et al., 2018) and replace the full attention mechanism in vanilla Transformer with Flow-Attention. All the experiments are conducted on 2 NVIDIA 2080 Ti GPUs.

Results. As shown in Table 2, Flowformer achieves state-of-the-art performance in both the ListOps and Retrieval tasks and competitive performance in other tasks. Overall, Flowformer achieves competitive performance over previous methods (55.23→56.48). Besides, *our model consistently surpasses the vanilla Transformer in all five tasks*, even though the latter adopts the full attention mechanism with quadratic complexity. In addition, we conduct the ablation study to testify the effectiveness of each module in our design. As we show, the competition and allocation mechanisms bring 1.23 (55.25→56.48) and 0.9 (55.58→56.48) averaged promotion respectively.

Efficiency. We conduct experiments on LRA to evaluate the efficiency of our model in dealing with long sequences. As shown in Table 3, Flowformer presents great efficiency in both the training and inference phases under different input sequence lengths (1K∼4K). Especially compared to Performer (Choromanski et al., 2021), Flowformer achieves the

Table 2. Results on the Long-Range Arena. The best result is in bold and the second best is underlined. Classification accuracy is recorded.

Model	ListOps ↑	Text ↑	Retrieval ↑	Image ↑	Pathfinder ↑	Avg ↑
Local Attention (Tay et al., 2021)	15.82	52.98	53.39	41.46	66.63	46.06
Linear Trans. (Katharopoulos et al., 2020)	16.13	**65.90**	53.09	42.34	75.30	50.55
Reformer (Kitaev et al., 2020)	37.27	56.10	53.40	38.07	68.50	50.67
Sparse Trans. (Child et al., 2019)	17.07	63.58	59.59	**44.24**	71.71	51.24
Sinkhorn Trans. (Tay et al., 2020b)	33.67	61.20	53.83	41.23	67.45	51.29
Linformer (Wang et al., 2020)	35.70	53.94	52.27	38.56	<u>76.34</u>	51.36
Performer (Choromanski et al., 2021)	18.01	<u>65.40</u>	53.82	42.77	**77.05**	51.41
Synthesizer (Tay et al., 2020a)	36.99	61.68	54.67	41.61	69.45	52.88
Longformer (Beltagy et al., 2020)	35.63	62.85	56.89	42.22	69.71	53.46
Transformer (Vaswani et al., 2017)	36.37	64.27	57.46	42.44	71.40	54.39
BigBird (Zaheer et al., 2020)	36.05	64.02	59.29	40.83	74.87	55.01
cosFormer (Zhen et al., 2022)	<u>37.90</u>	63.41	61.36	43.17	70.33	55.23
Flowformer w/o Competition	36.80	63.48	<u>61.66</u>	42.39	71.90	55.25
Flowformer w/o Allocation	37.00	63.78	61.33	42.52	73.26	<u>55.58</u>
Flowformer	**38.70**	64.29	**62.24**	<u>43.20</u>	73.95	**56.48**

Table 3. Efficiency analysis (steps per second) on the Long-Range Arena in both inference and training phases. Experiments are conducted on 2 NVIDIA 2080 Ti GPUs. The best performance is in bold and the second is underlined. "-" indicates the out-of-memory situation.

Model Speed	Inference (Steps per second)				Train (Steps per second)			
Sequence Length	1K	2K	3K	4K	1K	2K	3K	4K
Transformer (Vaswani et al., 2017)	81.83	25.26	-	-	22.12	7.50	-	-
Local Attention (Tay et al., 2021)	98.28	96.51	94.60	95.60	46.75	43.05	35.42	30.34
Linear Trans. (Katharopoulos et al., 2020)	97.33	96.14	94.03	93.69	<u>48.66</u>	**48.78**	<u>41.66</u>	35.44
Reformer (Kitaev et al., 2020)	60.92	60.30	39.37	26.98	46.07	22.93	14.34	9.56
Sparse Trans. (Child et al., 2019)	78.30	23.33	-	-	21.74	7.30	-	-
Sinkhorn Trans. (Tay et al., 2020b)	91.42	92.21	92.72	80.67	45.93	36.21	28.11	23.83
Linformer (Wang et al., 2020)	96.56	**96.84**	94.74	93.59	45.57	44.11	37.28	31.58
Performer (Choromanski et al., 2021)	**99.60**	<u>96.80</u>	**96.52**	**96.42**	47.34	<u>48.30</u>	41.00	<u>36.14</u>
Synthesizer (Tay et al., 2020a)	65.44	-	-	-	5.16	-	-	-
Longformer (Beltagy et al., 2020)	73.56	-	-	-	13.09	-	-	-
BigBird (Zaheer et al., 2020)	82.50	54.12	37.83	29.34	27.34	16.95	12.00	9.33
cosFormer (Zhen et al., 2022)	96.46	95.58	95.19	94.69	46.50	45.24	39.49	35.09
Flowformer	<u>98.83</u>	96.21	<u>95.65</u>	<u>95.82</u>	**49.76**	47.18	**41.93**	**36.79**

competitive speed for inference and the better efficiency for training. Not only with comparable efficiency, Flowformer also surpasses Performer by a large margin (51.41→56.48).

4.2. Language Modeling

Setup. We conduct the language modeling experiment on the WikiText-103 (Merity et al., 2017), which is to estimate the probability distribution of a token given the previous ones. We use this task to testify the causal version of Flow-Attention. Following the well-established experiment setting (Peng et al., 2021), the sequence length is set as 512 for both training and evaluation. The model architecture consists of 6 decoder layers with 8 heads and 512 hidden channels for attention mechanism (Ott et al., 2019). The number of hidden channels for the feed-forward layers is set as 2048. All the models are trained from scratch without pre-training on 4 NVIDIA TITAN RTX 24GB GPUs for 150K updates after a 6K-steps warm-up.

Table 4. Results on language modeling with WikiText-103 (Merity et al., 2017). A lower perplexity indicates the better results.

Model	Perplexity ↓
Transformer (2017)	33.0
Linear Trans. (2020)	38.4
Reformer (2020)	33.6
Performer (2021)	37.5
TRF-Transformer (2021)	33.6
TRF-Transformer-GATE (2021)	31.3
cosFormer (2022)	34.1
Flowformer w/o Competition	31.2
Flowformer w/o Allocation	32.2
Flowformer	**30.8**

Results. We can find that Flowformer achieves the best performance in language modeling task from Table 4 and even outperforms the vanilla Transformer (Vaswani et al., 2017). Since the language modeling is an autoregressive

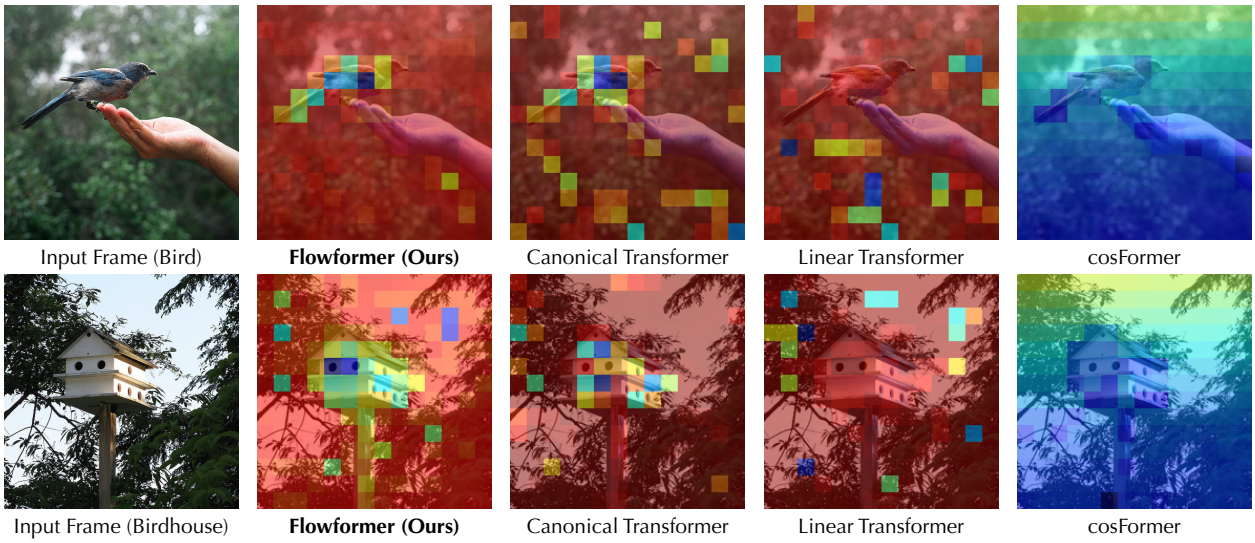


Figure 4. Visualization of learned attention. We present the sum of attention weights to each frame patch in the last layer of the model. For Flowformer, we visualize the competition weights $\mathrm{Softmax}(\widehat{\mathbf{O}}) \in \mathbb{R}^{m\times 1}$, which is applied to sources for non-trivial aggregation.

task, these experiments can also verify the effectiveness of Flow-Attention in the causal version. Besides, we conduct the ablation study for both the competition and allocation mechanisms. The results prove the effectiveness of these two modules, where competition and allocation mechanisms bring 1.4 and 0.4 absolute reductions in perplexity respectively. It is also notable that, both the competition and allocation mechanisms are based on flow conservation. Thus, any one of them can avoid trivial attention to some extent.

4.3. Image Recognition

Setup. We testify the capability of Flowformer in image recognition by experimenting on the ImageNet-1K (Deng et al., 2009). This dataset contains 1.28M training images and 50K validation images with 1,000 classes. Each image is in the resolution of 224×224. The Top-1 accuracy and Top-5 accuracy are recorded as the metrics. To fully evaluate our proposed Flowformer, we demonstrate the experiments in the following two aspects:

- Compare different attentions under the same Transformer architecture. We present Flowformer with 19 layers in a four-stage hierarchical structure, where the channels are in $\{96, 192, 384, 768\}$ and the input sequence length for each stage is in $\{3136, 784, 196, 49\}$ correspondingly. Global average pooling and linear projection are employed at the end of the model for classification. We take extensive efficient Transformers as baselines, as well as ViT (2021).
- Apply the Flow-Attention to the specific-designed vision Transformer, such as DeiT (Touvron et al., 2021), which adopts the token distillation for data efficiency.

All the experiments are conducted on 8 NVIDIA TITAN RTX 24GB GPUs for 300 epochs.

Table 5. Accuracy results (%) on ImageNet-1K (Deng et al., 2009). A higher accuracy indicates the better performance.

Model	Complex.	Params (MB)	Flops (G)	Top-1 Acc.	Top-5 Acc.
ViT-base (2021)	$\mathcal{O}(n^2d)$	86	55.4	77.9	/
ViT-large (2021)	$\mathcal{O}(n^2d)$	307	190.7	76.5	/
Full Attn. (2017)	$\mathcal{O}(n^2d)$	41	6.7	78.7	94.3
Linear Trans. (2020)	$\mathcal{O}(nd^2)$	41	6.3	79.0	94.1
Reformer (2020)	$\mathcal{O}((n\log n)d)$	37	6.0	79.6	94.7
Longformer (2020)	$\mathcal{O}(nd^2)$	38	6.3	77.6	93.1
Performer (2021)	$\mathcal{O}(nd^2)$	41	6.3	78.1	93.2
Nyströmformer (2021)	$\mathcal{O}(nd^2)$	41	6.3	77.2	93.0
YOSO-E (2021)	$\mathcal{O}(nd^2)$	41	5.8	79.0	94.3
SOFT (2021)	$\mathcal{O}(nd^2)$	37	5.8	79.2	94.5
cosFormer (2022)	$\mathcal{O}(nd^2)$	41	6.3	68.3	88.0
Flowformer	$\mathcal{O}(nd^2)$	41	6.3	**80.6**	**94.9**
DeiT-S (2021)	$\mathcal{O}(n^2d)$	22	4.6	79.8	**95.0**
DeiT+Flowformer	$\mathcal{O}(nd^2)$	22	**4.2**	**80.0**	94.8

Results. Table 5 shows that Flowformer achieves the strong performance in both the Top-1 and Top-5 accuracy metrics along with the linear complexity. Besides, Flowformer is the best linear Transformer and even surpasses the vanilla Transformer (Top-1: 78.7 v.s. 80.4). It is notable that, comparing to Flowformer, cosFormer (Zhen et al., 2022) provides a relatively poor result (Top-1: 68.3 v.s. 80.4). This is because that cosFormer directly incorporates the locality inductive bias along the patch sequence and does not consider the spatial position. This experiment also demonstrates the importance of specific-inductive-bias-free design, which can be naturally satisfied in Flowformer.

Table 6. Accuracy results (%) on time series classification. A higher accuracy indicates the better performance. As for the baselines of the Transformer family, we include the the canonical Transformer (Trans.), Linear Transformer (Linear.), Reformer (Re.), Longformer (Long.), Performer (Per.), cosFormer (cos.) and etc for a comprehensive comparison.

Dataset / Model	Classical methods			Deep Models										
				RNN	TCN	Transformer and its Efficient Variants								
	DTW (1994)	XGBoost (2016)	Rocket (2020)	LSTM (1997)	Unsuper. (2019)	Trans. (2017)	Linear. (2020)	Re. (2020)	Long. (2020)	Per. (2021)	YOSO-E (2021)	SOFT (2021)	cos. (2022)	**Flow. (ours)**
EthanolConcentration	32.3	43.7	45.2	32.3	28.9	32.7	31.9	31.9	32.3	31.2	31.2	32.3	33.5	33.8
FaceDetection	52.9	63.3	64.7	57.7	52.8	67.3	67.0	68.6	62.6	67.0	67.3	64.8	67.1	67.6
Handwriting	28.6	15.8	58.8	15.2	53.3	32.0	34.7	27.4	39.6	32.1	30.9	28.9	34.7	33.8
Heartbeat	71.7	73.2	75.6	72.2	75.6	76.1	76.6	77.1	78.0	75.6	76.5	77.1	75.6	77.6
JapaneseVowels	94.9	86.5	96.2	79.7	98.9	98.7	99.2	97.8	98.9	98.1	98.6	98.3	99.2	98.9
PEMS-SF	71.1	98.3	75.1	39.9	68.8	82.1	82.1	82.7	83.8	80.9	85.2	83.2	80.9	83.8
SelfRegulationSCP1	77.7	84.6	90.8	68.9	84.6	92.2	92.5	90.4	90.1	91.5	91.1	91.1	91.8	92.5
SelfRegulationSCP2	53.9	48.9	53.3	46.6	55.6	53.9	56.7	56.7	55.6	56.7	53.9	55.0	55.6	56.1
SpokenArabicDigits	96.3	69.6	71.2	31.9	95.6	98.4	98.0	97.0	94.4	98.4	98.9	98.4	98.8	98.8
UWaveGestureLibrary	90.3	75.9	94.4	41.2	88.4	85.6	85.0	85.6	87.5	85.3	88.4	85.6	85.0	86.6
Average Accuracy	67.0	66.0	<u>72.5</u>	48.6	70.3	71.9	72.4	71.5	72.0	71.9	72.2	71.5	72.2	**73.0**

Efficiency. In addition to the theoretical complexity analysis, we also provide the parameter and computation efficiency comparison in Table 5. Note that, compared to the canonical attention mechanism, Flow-Attention brings no extra model parameters. Thus, a promising way is to incorporate the Flow-Attention along with other well-designed vision Transformers, which can obtain comparable results with linear complexity, such as DeiT v.s. DeiT+Flowformer (Top-1: 79.8 $\rightarrow$ 80.0, Flops: 4.6 $\rightarrow$ 4.2). Since Flowformer is linear in sequence length, with the longer patch sequence, the efficiency promotion will be more remarkable, which is important for the models to scale up.

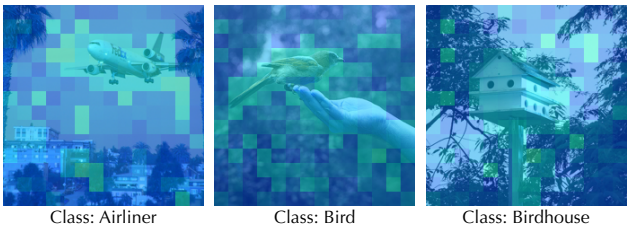


Figure 5. Allocation weights $\mathrm{Sigmoid}(\widehat{\mathbf{I}})$ heatmap in Flowformer.

Attention visualization. To further elaborate the difference between Flowformer and other Transformers, we visualize the learned attention in Figure 4. We can find that both the Flowformer and canonical Transformer (Vaswani et al., 2017) can capture the essential parts correctly, while the latter will consume the quadratic complexity and the attention may be distracted by the background context. In contrast, without competition mechanism, Linear Transformer (Katharopoulos et al., 2020) fails in attending to the right area and presents a degenerated attention map. As for the cosFormer (Zhen et al., 2022), due to the unsuitable introduction of sequence dimension locality and the overlook of spatial position, the attention is only concentrated on the upper part of frames, which will impede the model capability. Based on above comparisons, we can verify the advantages of Flow-Attention in informative attention learning.

Allocation visualization. As stated in Eq. (8), the learned allocation weights can reflect the capacity that each sink accepts. From Figure 5, we find that inflow capacities remain large on the essential parts, which means that these parts require more global information from sources for classification and matches our expectation.

4.4. Time Series Classification

Setup. We adopt the time series classification task to evaluate the model performance for temporal sequences. We select 10 multivariate datasets from UEA Time Series Classification Archive (Bagnall et al., 2018) for experiments and follow the data pre-processing in (Zerveas et al., 2021). We use 2 layers for Transformer-based models with 512 hidden channels and 8 heads for the attention mechanism. In addition to the deep models, we also compare our model with classical methods: DTW (Berndt & Clifford, 1994), XGBoost (Chen & Guestrin, 2016) and Rocket (Dempster et al., 2020). Rocket is the state-of-the-art model for time series classification. All the experiments are conducted on one single NVIDIA TITAN RTX 24GB GPU for 100 epochs.

Results. Flowformer achieves the best performance on the time series classification benchmark (Table 6). Besides other linear Transformers and deep models, Flowformer also surpasses the state-of-the-art classical methods Rocket (Dempster et al., 2020) and is *the only method that beats classical methods on the averaged performance*. This result

Table 7. Reward results on D4RL (Fu et al., 2020). A higher reward or a lower deviation indicates the better performance.

Environment	BC (1989)	AWAC (2020)	DT (2021a)	Linear Trans. (2020)	Reformer (2020)	Performer (2021)	cosFormer (2022)	**Flowformer (Ours)**
Medium-Expert								
HalfCheetah	55.2	42.8	83.8±3.3	78.2±3.2	81.5±1.6	85.1±2.1	85.5±2.9	90.8±0.4
Hopper	52.5	55.8	104.0±2.5	107.2±0.9	104.2±9.8	93.5±13.9	98.1±7.4	109.9±1.0
Walker	107.5	74.5	107.7±0.6	67.2±27.3	71.4±1.8	72.6±2.4	100.5±14.5	108.0±0.4
Medium								
HalfCheetah	42.6	43.5	42.4±0.1	42.3±0.2	42.2±0.1	42.1±0.2	42.1±0.3	42.2±0.2
Hopper	52.9	57.0	64.2±1.1	58.7±0.4	59.9±0.7	59.7±7.5	59.8±3.8	66.9±2.5
Walker	75.3	72.4	70.6±3.2	57.9±10.6	65.8±4.9	63.3±10.7	71.4±1.2	71.7±2.5
Medium-Replay								
HalfCheetah	36.6	40.5	34.6±0.6	32.1±1.5	33.6±0.7	31.7±0.9	32.8±3.6	34.7±1.5
Hopper	18.1	37.2	79.7±7.4	74.3±7.0	66.1±2.6	64.6±24.2	59.3±16.5	75.5±14.5
Walker	26.0	27.0	62.9±5.0	62.1±7.4	50.1±3.5	61.3±6.7	60.5±9.9	62.0±3.1
Avg Reward	51.9	50.1	72.2±**2.6**	64.4±6.5	63.9±2.9	63.8±7.6	67.8±7.6	**73.5**±2.9

can verify the temporal modeling capacity of Flowformer, which is essential for sequential data. See Appendix D.2 for the case study in attention visualization.

4.5. Reinforcement Learning

Setup. We consider the continuous control tasks from D4RL benchmark (Fu et al., 2020) to evaluate the model performance on the offline reinforcement learning (Offline RL) (Lange et al., 2012; Levine et al., 2020). We select the HalfCheetah, Hopper and Walker as experiment environments, which are to control the movement of robot. To obtain a comprehensive evaluation, we experiment on different datasets pre-collected with three different behavior policies: Medium-Expert, Medium and Medium-Replay. Since the offline RL is an autoregressive task, it can also be used to testify the causal-version Flow-Attention. For comparison, we include the Decision Transformer (DT, Chen et al. 2021a), Behavior Cloning (BC, Pomerleau 1989), AWAC (Nair et al., 2020), Linear Transformer (Linear Trans., (Katharopoulos et al., 2020)), Reformer (Kitaev et al., 2020), Performer (Choromanski et al., 2021) and cosFormer (Zhen et al., 2022) as baselines, where DT is the state-of-the-art models for offline RL and adopts the canonical Transformer as the backbone. We adopt 3 layers with 256 hidden channels and 4 heads in all experiments for Flowformer and other Transformers. We repeat each experiment three times with different seeds on one single NVIDIA 2080 Ti GPU for 10 epochs.

Results. As shown in Table 7, it is notable that compared to the vanilla Transformer used in DT, previous efficient Transformers degenerate a lot and cannot provide a stable result. Especially, the averaged rewards of Reformer (Kitaev et al., 2020) and Performer (Choromanski et al., 2021) decrease seriously (72.2 v.s. 63.9 and 63.8 respectively), indicating that the locally sensitive hashing or random Fourier features may be not suitable for the global dependency modeling under the reinforcement learning context. In contrast, Flowformer still shows a competitive performance on this challenging control task in the comparison with DT (72.2 v.s. 73.5), justifying the generality of our proposed Flowformer in offline reinforcement learning.

5. Conclusions

This paper focuses on Transformer linearization and attempts to tackle this problem from a new view of the flow network. By introducing the flow conservation to the attention mechanism, we present the Flow-Attention mechanism, which can naturally achieve the competition mechanism for sources and the allocation mechanism for sinks to filter the aggregated source information. Empowered by Flow-Attention, Flowformer can achieve the linear complexity and avoid degenerated attention without specific inductive biases. With great generality, Flowformer achieves the strong performance on extensive areas, covering vision, language, long sequence, time series, and reinforcement learning.

Our future work includes scaling up the proposed efficient Flowformer to build general-purpose pre-trained models facilitating a wider range of upstream and downstream tasks.

Acknowledgements

This work was supported by the National Key Research and Development Plan (2020AAA0109201), National Natural Science Foundation of China (62022050 and 62021002), Beijing Nova Program (Z201100006820041), and BNRist Innovation Fund (BNR2021RC01002).

References

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. Network flows - theory, algorithms and applications. 1993.

Bagnall, A. J., Dau, H. A., Lines, J., Flynn, M., Large, J., Bostrom, A. G., Southam, P., and Keogh, E. J. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Berndt, D. J. and Clifford, J. Using dynamic time warping to find patterns in time series. In *KDD Workshop*, 1994.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P. W., Krass, M., Krishna, R., Kuditipudi, R., Kumar, A., Ladhak, F., Lee, M., Lee, T., Leskovec, J., Levent, I., Li, X. L., Li, X., Ma, T., Malik, A., Manning, C. D., Mirchandani, S., Mitchell, E., Munyikwa, Z., Nair, S., Narayan, A., Narayanan, D., Newman, B., Nie, A., Niebles, J. C., Nilforoshan, H., Nyarko, J., Ogut, G., Orr, L., Papadimitriou, I., Park, J. S., Piech, C., Portelance, E., Potts, C., Raghunathan, A., Reich, R., Ren, H., Rong, F., Roohani, Y., Ruiz, C., Ryan, J., Ré, C., Sadigh, D., Sagawa, S., Santhanam, K., Shih, A., Srinivasan, K., Tamkin, A., Taori, R., Thomas, A. W., Tramèr, F., Wang, R. E., Wang, W., Wu, B., Wu, J., Wu, Y., Xie, S. M., Yasunaga, M., You, J., Zaharia, M., Zhang, M., Zhang, T., Zhang, X., Zhang, Y., Zheng, L., Zhou, K., and Liang, P. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018.

Bridle, J. S. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In *NeurIPS*, 1989.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *NeurIPS*, 2020.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*, 2021a.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *NeurIPS*, 2021b.

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. *KDD*, 2016.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlós, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L. J., and Weller, A. Rethinking attention with performers. *ICLR*, 2021.

Dempster, A., Petitjean, F., and Webb, G. I. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Min. Knowl. Discov.*, 2020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Franceschi, J.-Y., Dieuleveut, A., and Jaggi, M. Unsupervised scalable representation learning for multivariate time series. In *NeurIPS*, 2019.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 1997.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. In *NeurIPS*, 2021.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, 2020.

Kitaev, N., Kaiser, L., and Levskaya, A. Reformer: The efficient transformer. In *ICLR*, 2020.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

Lange, S., Gabel, T., and Riedmiller, M. Batch reinforcement learning. In *Reinforcement learning*. 2012.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Linsley, D. A., Kim, J., Veerabadran, V., and Serre, T. Learning long-range spatial dependencies with horizontal gated-recurrent units. *NeurIPS*, 2018.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. C.-F., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. *ICCV*, 2021.

Lu, J., Yao, J., Zhang, J., Zhu, X., Xu, H., Gao, W., Xu, C., Xiang, T., and Zhang, L. Soft: Softmax-free transformer with linear complexity. In *NeurIPS*, 2021.

Luo, S., Li, S., Cai, T., He, D., Peng, D., Zheng, S., Ke, G., Wang, L., and Liu, T.-Y. Stable, fast and accurate: Kernelized attention with relative positional encoding. In *NeurIPS*, 2021.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A., and Potts, C. Learning word vectors for sentiment analysis. In *ACL*, 2011.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *ICLR*, 2017.

Nair, A., Dalal, M., Gupta, A., and Levine, S. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Nangia, N. and Bowman, S. R. Listops: A diagnostic dataset for latent tree learning. In *NAACL*, 2018.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. fairseq: A fast, extensible toolkit for sequence modeling. In *NAACL-HLT*, 2019.

Peng, H., Pappas, N., Yogatama, D., Schwartz, R., Smith, N. A., and Kong, L. Random feature attention. In *ICLR*, 2021.

Pomerleau, D. A. Alvinn: An autonomous land vehicle in a neural network. Technical report, Carnegie Melon Univ. Pittsburgh, PA. Artificial Intelligence and Psychology., 1989.

Radev, D. R., Muthukrishnan, P., Qazvinian, V., and Abu-Jbara, A. The acl anthology network corpus. *Lang. Resour. Eval.*, 2013.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *NeurIPS*, 2007.

Tay, Y., Bahri, D., Metzler, D., Juan, D.-C., Zhao, Z., and Zheng, C. Synthesizer: Rethinking self-attention in transformer models. In *ICML*, 2020a.

Tay, Y., Bahri, D., Yang, L., Metzler, D., and Juan, D.-C. Sparse sinkhorn attention. In *ICML*, 2020b.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena: A benchmark for efficient transformers. *ICLR*, 2020c.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena : A benchmark for efficient transformers. In *ICLR*, 2021.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and J'egou, H. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Vyas, A., Katharopoulos, A., and Fleuret, F. Fast transformers with clustered attention. *NeurIPS*, 2020.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Wu, H., Xu, J., Wang, J., and Long, M. Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting. In *NeurIPS*, 2021.

Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G. M., Li, Y., and Singh, V. Nyströmformer: A nyström-based algorithm for approximating self-attention. *AAAI*, 2021.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontañón, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. Big bird: Transformers for longer sequences. In *NeurIPS*, 2020.

Zeng, Z., Xiong, Y., Ravi, S. N., Acharya, S., Fung, G., and Singh, V. You only sample (almost) once: Linear cost self-attention via bernoulli sampling. In *ICML*, 2021.

Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., and Eickhoff, C. A transformer-based framework for multivariate time series representation learning. *KDD*, 2021.

Zhen, Q., Sun, W., Deng, H., Li, D., Wei, Y., Lv, B., Yan, J., Kong, L., and Zhong, Y. cosformer: Rethinking softmax in attention. In *ICLR*, 2022.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI*, 2021.

A. Implementation Details

A.1. Pseudo-code for Flow-Attention

We present the pseudo-code of normal Flow-Attention in Algorithm 1 and the causal version in Algorithm 2. Especially, in the causal version, we adopt the `Causal-Dot-Product` (Katharopoulos et al., 2020) for the information aggregation.

Algorithm 1 Multi-head Flow-Attention Mechanism (normal version).

Input: $\mathbf{Q} \in \mathbb{R}^{n\times d}, \mathbf{K} \in \mathbb{R}^{m\times d}, \mathbf{V} \in \mathbb{R}^{m\times d}$

$\mathbf{Q}, \mathbf{K}, \mathbf{V} = \texttt{Split}(\mathbf{Q}), \texttt{Split}(\mathbf{K}), \texttt{Split}(\mathbf{V})$ // $\mathbf{Q} \in \mathbb{R}^{n\times h\times \frac{d}{h}}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{m\times h\times \frac{d}{h}}$

$\mathbf{Q}, \mathbf{K} = \texttt{Sigmoid}(\mathbf{Q}), \texttt{Sigmoid}(\mathbf{K})$

$\mathbf{I} = \texttt{Sum}\Big(\mathbf{Q} \odot \texttt{Broadcast}\Big(\texttt{Sum}(\mathbf{K}, \texttt{dim=0}), \texttt{dim=0}\Big), \texttt{dim=2}\Big)$ // $\mathbf{I} \in \mathbb{R}^{n\times h}$

$\mathbf{O} = \texttt{Sum}\Big(\mathbf{K} \odot \texttt{Broadcast}\Big(\texttt{Sum}(\mathbf{Q}, \texttt{dim=0}), \texttt{dim=0}\Big), \texttt{dim=2}\Big)$ // $\mathbf{O} \in \mathbb{R}^{m\times h}$

$\widehat{\mathbf{I}} = \texttt{Sum}\Big(\mathbf{Q} \odot \texttt{Broadcast}\Big(\texttt{Sum}(\mathbf{K}/\mathbf{O}, \texttt{dim=0}), \texttt{dim=0}\Big), \texttt{dim=2}\Big)$ // $\widehat{\mathbf{I}} \in \mathbb{R}^{n\times h}$

$\widehat{\mathbf{O}} = \texttt{Sum}\Big(\mathbf{K} \odot \texttt{Broadcast}\Big(\texttt{Sum}(\mathbf{Q}/\mathbf{I}, \texttt{dim=0}), \texttt{dim=0}\Big), \texttt{dim=2}\Big)$ // $\widehat{\mathbf{O}} \in \mathbb{R}^{m\times h}$

$\mathbf{R} = \texttt{Matmul}\Big(\mathbf{Q}/\mathbf{I}, \texttt{Matmul}\Big(\mathbf{K}, \mathbf{V} \odot \texttt{Softmax}(\widehat{\mathbf{O}})\Big)\Big) \odot \texttt{Sigmoid}(\widehat{\mathbf{I}})$ // $\mathbf{R} \in \mathbb{R}^{n\times h\times \frac{d}{h}}$

Return $\mathbf{R}$

Algorithm 2 Multi-head Flow-Attention Mechanism (causal version).

Input: $\mathbf{Q} \in \mathbb{R}^{n\times d}, \mathbf{K} \in \mathbb{R}^{n\times d}, \mathbf{V} \in \mathbb{R}^{n\times d}$

$\mathbf{D} = \texttt{Broadcast}\Big(\texttt{Arrange}(n), \texttt{dim=1}\Big)$ // $\mathbf{D} \in \mathbb{R}^{n\times h}$

$\mathbf{Q}, \mathbf{K}, \mathbf{V} = \texttt{Split}(\mathbf{Q}), \texttt{Split}(\mathbf{K}), \texttt{Split}(\mathbf{V})$ // $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n\times h\times \frac{d}{h}}$

$\mathbf{Q}, \mathbf{K} = \texttt{Sigmoid}(\mathbf{Q}), \texttt{Sigmoid}(\mathbf{K})$

$\mathbf{I} = \texttt{Sum}\Big(\mathbf{Q} \odot \texttt{Cusum}(\mathbf{K}, \texttt{dim=0}), \texttt{dim=2}\Big)/\mathbf{D}$ // $\mathbf{I} \in \mathbb{R}^{n\times h}$

$\mathbf{O} = \texttt{Sum}\Big(\mathbf{K} \odot \texttt{Cusum}(\mathbf{Q}, \texttt{dim=0}), \texttt{dim=2}\Big)/\mathbf{D}$ // $\mathbf{O} \in \mathbb{R}^{n\times h}$

$\widehat{\mathbf{I}} = \texttt{Sum}\Big(\mathbf{Q} \odot \texttt{Cusum}(\mathbf{K}/\mathbf{O}, \texttt{dim=0}), \texttt{dim=2}\Big)/\mathbf{D}$ // $\widehat{\mathbf{I}} \in \mathbb{R}^{n\times h}$

$\widehat{\mathbf{O}} = \texttt{Sum}\Big(\mathbf{K} \odot \texttt{Cusum}(\mathbf{Q}/\mathbf{I}, \texttt{dim=0}), \texttt{dim=2}\Big)/\mathbf{D}$ // $\widehat{\mathbf{O}} \in \mathbb{R}^{n\times h}$

$\widehat{\mathbf{O}} = \Big(\texttt{Exp}(\widehat{\mathbf{O}})/\texttt{Cusum}\Big(\texttt{Exp}(\widehat{\mathbf{O}}), \texttt{dim=0}\Big)\Big) \odot \mathbf{D}$ // $\widehat{\mathbf{O}} \in \mathbb{R}^{n\times h}$

$\mathbf{R} = \texttt{Causal-Dot-Product}\Big(\mathbf{Q}/(\mathbf{I} \odot \mathbf{D}), \mathbf{K}, \mathbf{V} \odot \widehat{\mathbf{O}}\Big) \odot \texttt{Sigmoid}(\widehat{\mathbf{I}})$ // $\mathbf{R} \in \mathbb{R}^{n\times h\times \frac{d}{h}}$

Return $\mathbf{R}$

A.2. Flowformer Architecture

Suppose the model contains L layers with length-n input $\mathbf{X} \in \mathbb{R}^{n\times d_{\text{input}}}$. The overall equations of the l-th layer are:

$$\begin{aligned}\mathbf{Z}^l &= \text{Layer-Norm}\Big(\text{Flow-Attention}(\mathbf{X}^{l-1}, \mathbf{X}^{l-1}, \mathbf{X}^{l-1}) + \mathbf{X}^{l-1}\Big)\\ \mathbf{X}^l &= \text{Layer-Norm}\Big(\text{Feed-Forward}(\mathbf{Z}^l) + \mathbf{Z}^l\Big),\end{aligned} \tag{9}$$

where $\mathbf{X}^l \in \mathbb{R}^{n\times d}, l \in \{1, \cdots, L\}$ denotes the output of the l-th layer with d channels. The initial input $\mathbf{X}^0 = \text{Embedding}(\mathbf{X}) \in \mathbb{R}^{n\times d}$ represents the embedded raw data. $\mathbf{Z}^l \in \mathbb{R}^{n\times d}$ is the l-th layer's hidden representation.

A.3. Experiment Configuration

For the Long-Range Arena (Tay et al., 2020c) and the WikiText-103 (Merity et al., 2017) benchmarks, we just following the official public protocol in Long-Range Arena[1] and Fairseq[2]. Here are the configurations for the other three benchmarks. All details can be found in our code: https://github.com/thuml/Flowformer.

ImageNet-1K The image with resolution 224×224 is split into several patches by convolution at the beginning. Inside the model, we adopt the convolution layer for down sampling between different stages. Thus, different from the model architecture in ViT (Dosovitskiy et al., 2021), we adopt a hierarchical architecture of Transformer without classification token. The global information is aggregated with the global average pooling layer in the end with the last projector for classification. The details are summarized in Table 8.

Table 8. Hierarchical architecture for vision recognization task.

PARAMETERS	STAGE 1	STAGE 2	STAGE 3	STAGE 4
LAYERS	3	3	10	3
CHANNELS	96	192	384	786
HEADS	16	16	16	16
INPUT SEQUENCE LENGTH	3136	784	196	49

UEA Here is the statistical details of the UEA time series classification dataset. As shown in Table 9, this benchmark includes various types of subsets, such as the low or high dimension, long or short length, sufficient or insufficient data, many or few class numbers. Thus, experiments on this benchmark can provide a comprehensive comparison.

Table 9. Statistical Results of the UEA dataset.

DATASET	TRAINSIZE	TESTSIZE	NUMDIMENSIONS	SERIESLENGTH	NUMCLASSES
ETHANOLCONCENTRATION	261	263	3	1751	4
FACEDETECTION	5890	3524	144	62	2
HANDWRITING	150	850	3	152	26
HEARTBEAT	204	205	61	405	2
JAPANESEVOWELS	270	370	12	29	9
PEMS-SF	267	173	963	144	7
SELFREGULATIONSCP1	268	293	6	896	2
SELFREGULATIONSCP2	200	180	7	1152	2
SPOKENARABICDIGITS	6599	2199	13	93	10
UWAVEGESTURELIBRARY	120	320	3	315	8

D4RL This benchmark is for offline RL (Lange et al., 2012; Levine et al., 2020), which is to learn the policy from a pre-collected dataset and then perform the action in the online environment. This task is challenging not only because of the difficulty of continuous control also due to the extrapolation error caused by the out-of-distribution actions. We experiment on the HalfCheetah, Hopper and Walker environments (Figure 6) and train the model with the datasets collected by different policies. We follow the configuration in Decision Transformer (Chen et al., 2021b) for experiments.

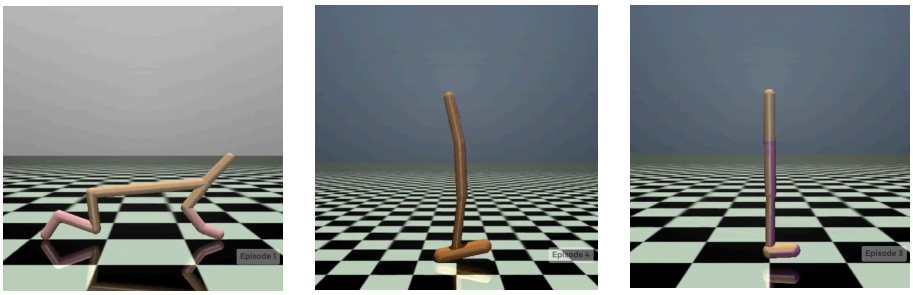


Figure 6. HalfCheetah, Hopper and Walker environments.

[1] https://github.com/google-research/long-range-arena
[2] https://github.com/pytorch/fairseq

B. Ablation studies

B.1. Ablation on Activate Function for Non-negative Operation

As stated in Eq. (4), to satisfy the non-negative requirement in flow network, we set the activate function ϕ as the sigmoid function. In section, we compare the Flowformer performance under different choices of ϕ, such as $\text{elu}(\cdot) + 1.0$, $\text{Relu}(\cdot)$. As shown in Table 10, all the choices of $\phi(\cdot)$ can achieve state-of-the-art performance. Benefiting from the optimization property and numerical stability, the sigmoid function achieves the best averaged performance.

Table 10. Ablation results on the Long-Range Arena under different choices of activate function.

MODEL	LISTOPS ↑	TEXT ↑	RETRIEVAL ↑	IMAGE ↑	PATHFINDER ↑	AVG ↑
$\text{ELU}(\cdot) + 1.0$	38.65	64.09	61.76	43.75	71.80	56.01
$\text{RELU}(\cdot)$	38.45	63.90	62.17	42.85	72.96	56.07
SIGMOID FUNCTION (FINAL VERSION)	38.70	64.29	62.24	43.20	73.95	**56.48**

B.2. Ablation on Activate Functions for Competition and Allocation

As shown in Eq. (8), the final design of Flowformer chooses the Softmax for Competition and the Sigmoid for Allocation, denoted by Softmax-Sigmoid. Obviously, there are 4 different types of choices. As shown in Table 11, we experiment on every choice exhaustively and find that Softmax-Sigmoid is the best. These results indicate that in Flow-Attention, Softmax is more suitable for Competition and Sigmoid fits the Allocation better. It is because the former is to highlight the tokens and the latter is to control the information flow as stated in the main text.

Table 11. Accuracy results (%) on LRA (averaged from 5 sub-tasks).

MODEL	COMPETITION	ALLOCATION	AVG ACC.
CHOICE 1	Sigmoid	Sigmoid	52.36
CHOICE 2	Sigmoid	Softmax	53.11
CHOICE 3	Softmax	Softmax	55.41
FLOWFORMER	Softmax	Sigmoid	**56.48**

C. Preliminaries of Flow Network

A flow network (Ahuja et al., 1993) is a directed graph where each edge has a capacity and each edge receives a flow. The amount of flow on an edge cannot exceed the capacity of the edge. The conservation property can be intuitively explained as follows: for each node, the incoming flow capacity is equal to the outgoing flow capacity.

In our paper, in addition to considering the inner source-sink flow shown in Figure 1, we demonstrate that attention mechanisms also exchange information with model's other parts. As presented in Figure 3, *Outside* the attention mechanism, the information of values ($\mathbf{V}$) is obtained from previous layer and the information of results ($\mathbf{R}$) will be provided to the next layer. Without loss of generality, we set the information incoming from or outgoing to other layers as the default value 1. Eq. (6) of the main text is to prove that, by conducting normalization operations formalized in Eq. (7), the incoming flow of each sink and the outgoing flow of each source are equal to 1 respectively, which is exactly the default value that we set for information flow between other layers. Thus, our design follows the flow conservation property well. It is also notable that our design is inspired by flow conservation property and the key insight is "fixed resource will cause competition".

D. More Attention Visualizations

We provide more attention visualizations on different tasks for a better intuitive understanding.

D.1. Image Recognition

In this section, we provide the attention visualization from various classes in the ImageNet-1K. As shown in Figure 9, Flowformer can capture the essential part in the image precisely. While the canonical Transformer (Vaswani et al., 2017) may appear the distraction problem and Linear Transformer (Katharopoulos et al., 2020) could show a trivial attention

without the attention concentration. We also visualize the learned allocation weights for corresponding classes in Figure 7, which can also cover the most informative parts for classification.

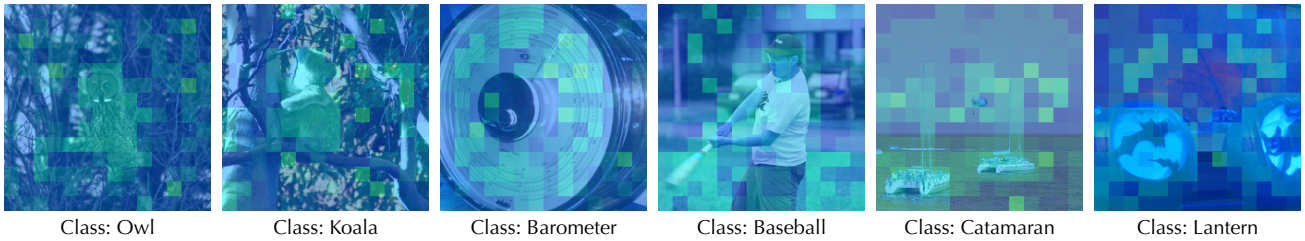
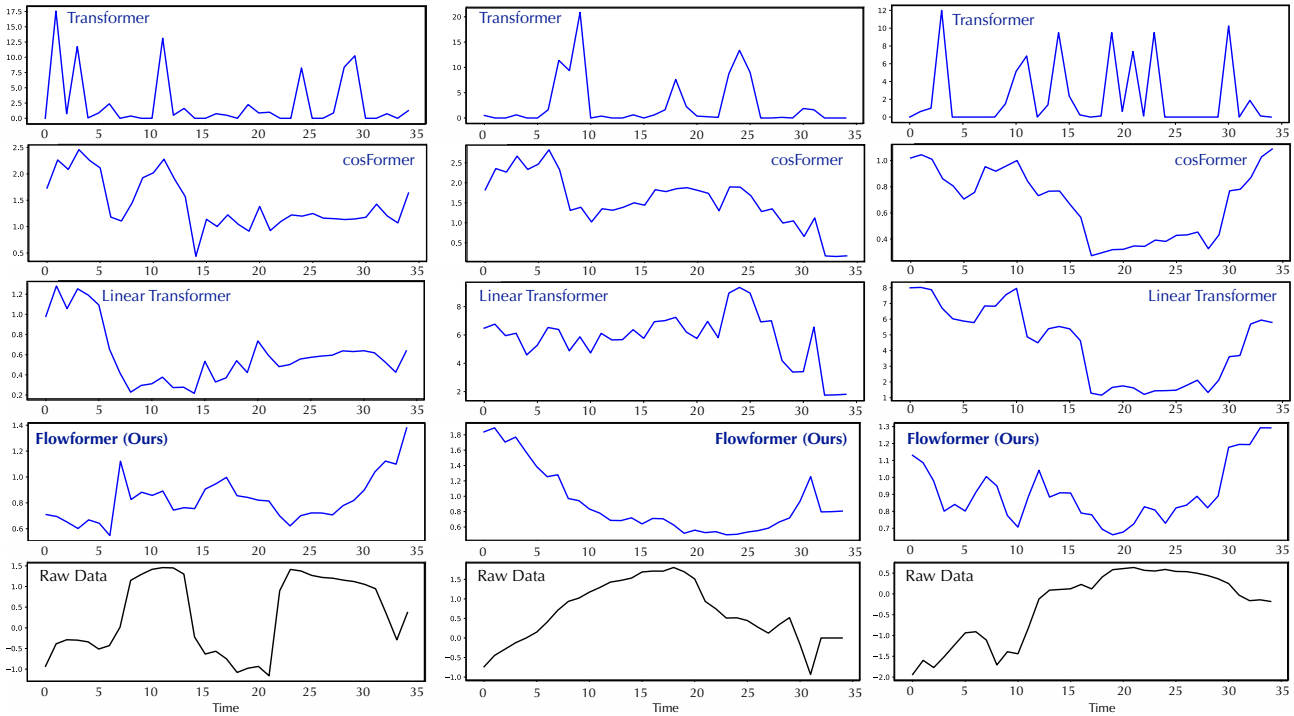


Figure 7. Visualization of learned allocation weights $\mathrm{Sigmoid}(\widehat{\mathbf{I}})$ of Flowformer.

D.2. Time Series Classification

We provide the attention visualization for the SpokenArabicDigits dataset. This dataset contains times series of mel-frequency cepstrum coefficients (MFCCs) corresponding to spoken Arabic digits from 0 to 9. Note that all the Transformer-based models achieve a great performance in this dataset, including Transformer (Vaswani et al. 2017, 98.4% accuracy), Linear Transformer (Katharopoulos et al. 2020, 98.0%), cosFormer (Zhen et al. 2022, 98.8%) and Flowformer (ours, 98.8%). Thus, we provide the visualization only to demonstrate how these attentions work in time series.

Generally, we can find that Flowformer successively pays non-trivial attention to different phases of the time series for different classes (Figure 8). Concretely, as shown in Figure 8(a), Flowformer pays more attention on the intervals $[5, 10]$ and $[30, 35]$, where are important for the classification of the two-stage pronunciation for "zero" ([ˈziroʊ]). Besides, Flowformer pays more attention to the distinguishable part in "two" ([tuː], Figure 8(b)) and "three" ([θriː], Figure 8(c)). While for the vanilla Transformer, it may be distracted by unimportant fluctuations (Figure 8(c)). As for the Linear Transformer and cosFormer, they may fail to capture the second phase in the pronunciation of "zero" (Figure 8(a)).



(a) Class: Zero (b) Class: Two (c) Class: Three

Figure 8. Learned attention in the SpokenArabicDigits dataset. We plot the competition weights $\mathrm{Softmax}(\widehat{\mathbf{O}})$ for Flowformer.

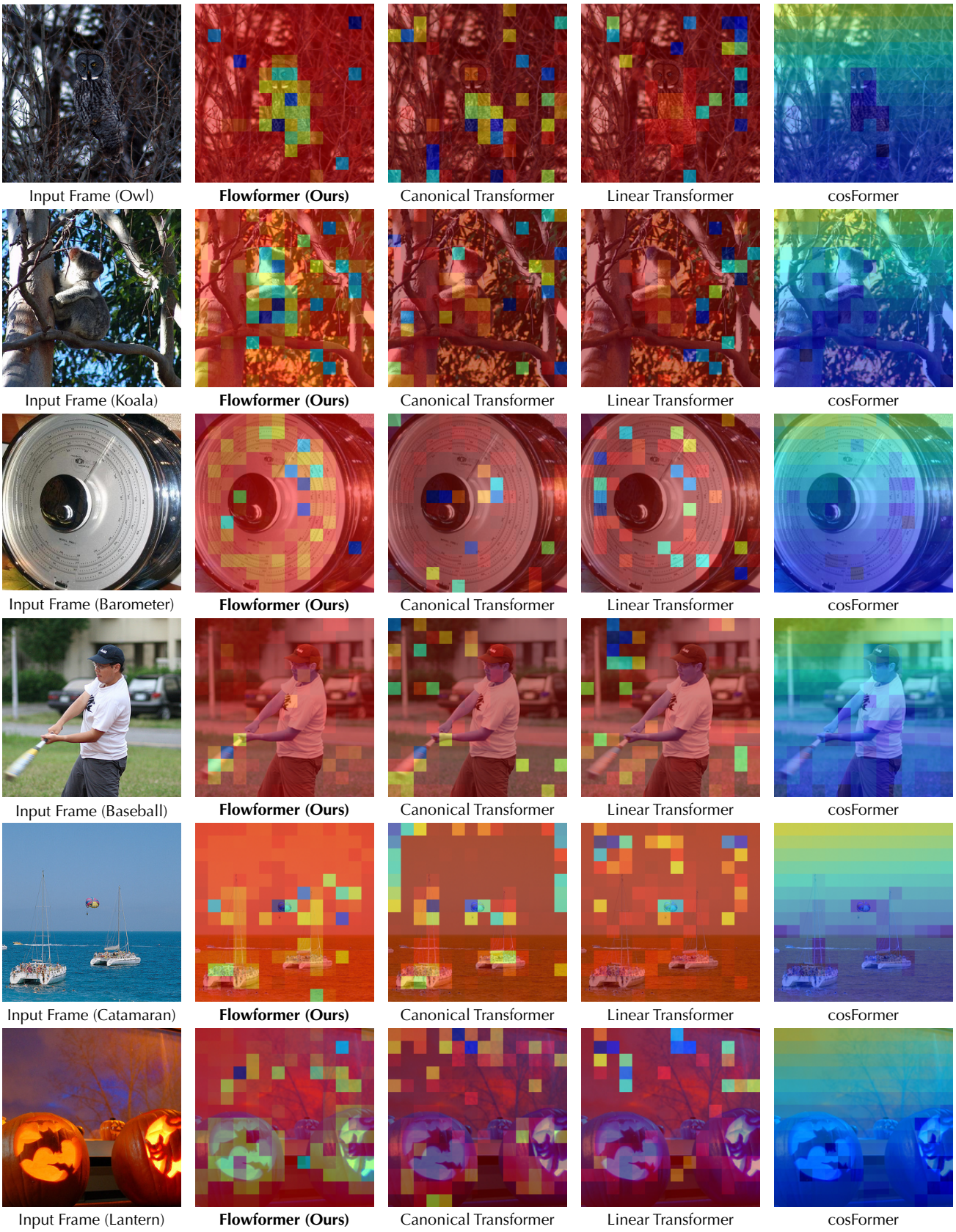


Figure 9. Comparison of learned attention in the ImageNet-1K dataset.