# Skip-Layer Attention: Bridging Abstract and Detailed Dependencies in Transformers

**Anonymous ACL submission**

## Abstract

The Transformer architecture has significantly advanced deep learning, particularly in natural language processing, by effectively managing long-range dependencies. However, as the demand for understanding complex relationships grows, refining the Transformer's architecture becomes critical. This paper introduces Skip-Layer Attention (SLA) to enhance Transformer models by enabling direct attention between non-adjacent layers. This method improves the model's ability to capture dependencies between high-level abstract features and low-level details. By facilitating direct attention between these diverse feature levels, our approach overcomes the limitations of current Transformers, which often rely on suboptimal intra-layer attention. Our implementation extends the Transformer's functionality by enabling queries in a given layer to interact with keys and values from both the current layer and one preceding layer, thus enhancing the diversity of multi-head attention without additional computational burden. Extensive experiments demonstrate that our enhanced Transformer model achieves superior performance in language modeling tasks, highlighting the effectiveness of our skip-layer attention mechanism.

## 1 Introduction

The Transformer architecture has made notable strides in the field of large language models (LLMs) (Devlin et al., 2019; Radford and Narasimhan, 2018; Radford et al., 2019; Brown et al., 2020; Ouyang et al., 2022; OpenAI, 2023). These models have impressively tackled a variety of tasks, including natural language understanding (Hendrycks et al., 2021), general question answering (Rein et al., 2023), coding (Chen et al., 2021), mathematics (Cobbe et al., 2021), and scientific knowledge (Chen et al., 2023). However, as data grows more complex and relationships more intricate (Rein et al., 2023), there's a need for ongoing improvements in the architecture to keep up with these challenges.

The primary strength of the Transformer lies in its self-attention mechanism, which allows each element in the input sequence to compare directly with every other element, thereby capturing dependencies regardless of their distance (Vaswani et al., 2017). Nevertheless, this design faces limitations when handling more complex relationships. The original intra-layer attention in Transformers is often inadequate for capturing the deeper interactions (i.e., high-level abstract features and low-level details) demanded by more complex tasks (Tenenbaum, 2018; Yang et al., 2016).

To address these limitations, researchers have explored various methods employed in earlier models such as ResNet (He et al., 2016) and Highway Networks (Srivastava et al., 2015). Our goal is to refine inter-layer interactions within Transformers. Drawing inspiration from DenseNet (Huang et al., 2017) in convolutional neural networks (CNNs), which employs dense cross-layer connections to facilitate feature propagation, we propose a novel Skip-Layer Attention (SLA) approach to enhance the Transformer model. Our implementation augments the Transformer's capabilities by permitting queries in a given layer to interact not only with keys and values from the current layer but also from the preceding layer. This method enriches the diversity of multi-head attention, while maintaining the same computational efficiency. Unlike DenseNets, which focus on identical tokens across layers, our strategy connects both identical and distinct tokens, thereby enhancing the model's capacity to capture and incorporate both abstract and detailed dependencies. Our contributions are as follows:

- We propose a novel mechanism that enables direct attention between non-adjacent layers, enhancing the ability to capture dependencies

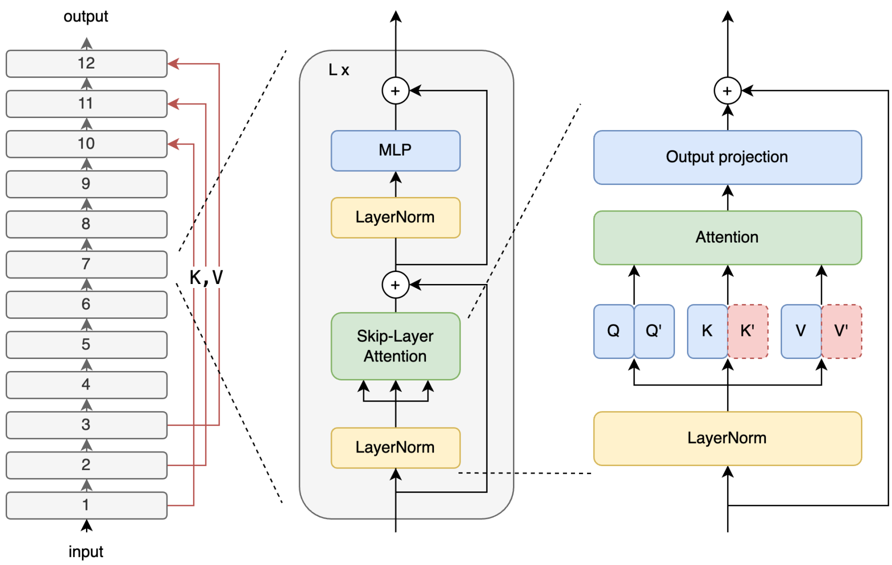


Figure 1: Model architecture of the Transformer with skip-layer attention. The left figure illustrates a Transformer model with 12 layers, each equipped with an additional skip-layer attention connection (e.g., layer 1 to layer 10, layer 2 to layer 11, layer 3 to layer 12). The center figure provides a zoomed-in view of each layer, highlighting the skip-layer attention and MLP sublayers. The right figure details the skip-layer attention mechanism, with red indicating keys and values from the preceding layer.

between high-level abstract features and low-level details.

- Our method extends the Transformer's functionality without significantly increasing computational complexity, making it practical for large-scale applications.

- Through extensive experiments against Transformer baselines, we demonstrate the effectiveness of our enhanced architecture in language modeling tasks.

## 2 Related Work

The concept of enhancing network connectivity originates from earlier architectures such as ResNet (He et al., 2016), which introduces residual connections. These residual connections enable the training of much deeper networks by facilitating the flow of gradients during backpropagation. Highway Networks (Srivastava et al., 2015) introduce gated connections to regulate information flow across layers, making the end-to-end training of deep networks more feasible.

DenseNet (Huang et al., 2017) advances this idea by creating an intricate connectivity pattern where each layer connects to every other layer in a feed-forward manner. This dense connectivity promotes feature reuse and significantly reduces the number of parameters, directly inspiring the skip-layer connectivity pattern explored in our work. However, DenseNet primarily targets CNNs and is mainly applied to computer vision tasks, with connections occurring between the same tokens in subsequent layers. Our approach extends this concept to Transformers by incorporating direct connections among both identical and distinct tokens.

More recently, Brandon et al. (2024) propose sharing key/value heads across adjacent layers to reduce the size of the KV cache. This strategy draws inspiration from the success of Multi-Query Attention (Shazeer, 2019) and Grouped-Query Attention (Ainslie et al., 2023). Our skip-layer attention, however, enables direct attention between non-adjacent layers, thus bridging the dependencies between high-level abstract features and low-level details. This approach encapsulates the depth of interactions required by more demanding tasks.

## 3 Method

Our novel Transformer model enhances the standard Transformer architecture by incorporating skip-layer attention, aimed at improving the information flow between non-adjacent layers. As shown in Figure 1, our approach replaces the conventional multi-head attention sublayer with skip-layer attention sublayer. This upgrade establishes direct connections between layers that are not immediately adjacent, thereby promoting a more efficient and comprehensive exchange of information across the entire network.

Our model retains the Transformer's original multi-head attention mechanism, utilizing queries, keys, and values, but extends this framework by enabling queries in a given layer to interact with keys and values from both the current layer and one preceding layer. This extension is fundamental to the implementation of skip-layer attention.

We denote the queries after linear projection for all heads as the tensor $Q \in \mathbb{R}^{h \times T \times d}$, where $h$ is the number of heads, $T$ is the sequence length, and $d$ is the hidden size for each head. Similarly, the keys and values are represented as tensors $K \in \mathbb{R}^{h \times T \times d}$ and $V \in \mathbb{R}^{h \times T \times d}$, respectively. The number of skip layers is denoted as $n_l$, and the number of skip heads as $n_h$.

The attention mechanism for each head is described by the following equations:

$$\mathrm{H}_i^l = \begin{cases} \mathrm{Att}(Q_i^l, K_i^l, V_i^l) & \text{if } i \in \{1, ..., h - n_l\} \\ \mathrm{Att}(Q_i^l, K_i^{l-n_l}, V_i^{l-n_l}) & \text{if } i \in \{h - n_l + 1, ..., h\} \end{cases} \quad (156)$$

where $i$ represents the index of the head and $l$ represents the index of the layer. This formulation allows our model to effectively bridge abstract and detailed dependencies and improve information flow throughout the network. The core implementation in PyTorch can be found in Appendix A.1.

# 4 Experimental Setup

## 4.1 Dataset

We use the OpenWebText corpus[1], an open-source recreation of the WebText dataset. It comprises approximately 8 million documents sourced from Reddit-linked web content. The corpus is divided into a training set with about 9 billion tokens and a validation set with around 4 million tokens.

## 4.2 Training Setup

Training batches contain 524,288 tokens, stabilizing the process across model scales. The training spans a maximum of 18,000 steps, processing roughly 9.4 billion tokens, equivalent to one full epoch on the training set. We experiment with three sizes of the GPT-2 architecture: GPT2 (124M), GPT2-Medium (350M) and GPT2-Large (774M).

Table 1: Optimal number of skip layers.

| #SkipLayer | #SkipHead | Loss | Abs. Impr. |
|---|---|---|---|
| 0 (Baseline) | 0 | 3.3826 | - |
| 1 | 6 | 3.4030 | -0.0204 |
| 3 | 6 | 3.2958 | 0.0868 |
| 6 | 6 | 3.2853 | 0.0973 |
| **9** | **6** | **3.2750** | **0.1076** |
| 11 | 6 | 3.3734 | 0.0092 |

Table 2: Optimal number of skip heads.

| #SkipLayer | #SkipHead | Loss | Abs. Impr. |
|---|---|---|---|
| 0 (Baseline) | 0 | 3.3826 | - |
| 9 | 3 | 3.3543 | 0.0283 |
| 9 | 6 | 3.2750 | 0.1076 |
| **9** | **9** | **3.2497** | **0.1329** |
| 9 | 12 | 3.2643 | 0.1183 |

Flash Attention (Dao, 2023) is incorporated to accelerate attention operations. Training is conducted on NVIDIA V100 GPUs with 32GB of memory. We experiment with sequence lengths of 4096, 8192, and 16,384 tokens to explore long-range dependencies. Code implementation and optimization are managed using the nanoGPT framework[2].

All models start with a learning rate of 1.5e-4, determined to offer a good balance between rapid convergence and stability. Model performance is evaluated based on the loss on the validation dataset.

# 5 Result

## 5.1 Number of skip layers

We initially explore the impact of varying the number of skip layers using a GPT-2 model (124M parameters) as our default backbone. This model has a hidden size of 768, 12 heads, 12 layers, and supports a sequence length of 16,384. The default number of skip heads is set to 6. As shown in Table 1, our findings indicate that the optimal performance enhancement via the skip-layer attention method is achieved with 9 skip layers, resulting in a substantial absolute improvement of 0.1076 over the baseline. Configurations with 3 and 6 skip layers also demonstrate notable progress. However, employing just a single skip layer yields no benefit; this might be attributed to the similarity between the key and value heads among adjacent layers, as discussed in Brandon et al. (2024). Similarly, setting the number of skip layers to 11 does

[1] https://skylion007.github.io/OpenWebTextCorpus/

[2] https://github.com/karpathy/nanoGPT/

Table 3: Model size and sequence length variations.

| Model | Length | Baseline | Skip-Layer Attention | Abs. Impr. | Training Speedup |
|---|---|---|---|---|---|
| GPT2(124M) | 4096 | 3.1858 | 3.1762 | 0.0096 | -1.69% |
| | 8192 | 3.2077 | 3.2020 | 0.0057 | -0.28% |
| | **16384** | **3.3826** | **3.2497** | **0.1329** | **0.83%** |
| GPT2-Medium(350M) | 4096 | 2.9538 | 2.9399 | 0.0139 | -0.76% |
| | **8192** | **3.0335** | **2.9506** | **0.0829** | **-2.34%** |
| GPT2-Large(774M) | 4096 | 2.8271 | 2.8156 | 0.0115 | -1.24% |

not produce noticeable advancements, possibly due to the presence of only one skip-layer attention in this setup. Based on these results, we recommend that 3/4 of the total number of layers is the most effective number of skip layers.

### 5.2 Number of skip heads

Subsequently, we investigate the impact of varying the number of skip heads, also using a GPT-2 model (124M) and a sequence length of 16,384 as our default backbone. The default number of skip layers is set to 9, based on the optimal configuration identified in the previous section. As presented in Table 2, our results indicate that the optimal performance enhancement is observed with 9 skip heads, yielding a significant absolute improvement of 0.1329 over the baseline. Configurations with 6 and 12 skip heads also demonstrate commendable improvements. Notably, the configuration with 12 skip heads suggests that the keys and values from the last 3 layers all use the keys and values from the first 3 layers. This implies that direct attention modeling between high-level abstract features and low-level detail features is more important than attention modeling purely between high-level abstract features. Conversely, employing only 3 skip heads shows no substantial benefit, indicating that more skip-attention heads are necessary than the original attention heads. Based on these findings, we recommend setting the number of skip heads to 3/4 of the total number of heads.

### 5.3 Model Size and Sequence Length Variations

In this section, we explore the effects of varying model sizes and sequence lengths on performance. We maintain the default configuration of using 3/4 of the total number of layers as skip layers and 3/4 of the total number of heads as skip heads. Due to the 32GB memory limit of the V100 GPU, we restrict our tests to model sizes and sequence lengths that do not trigger out-of-memory (OOM) errors when using Distributed Data Parallel (DDP).

As shown in Table 3, we observe an absolute improvement of 0.1329 over the baseline when using GPT-2 (124M) with a sequence length of 16,384. However, no significant improvements are noted for sequence lengths of 4,096 and 8,192. This suggests that longer sequences benefit more from our skip-layer attention method, likely because they encompass more abstract and detailed dependencies. Furthermore, for a sequence length of 8,192, GPT-2 Medium (350M) achieves an absolute improvement of 0.0829 over the baseline, while GPT-2 (124M) shows no noticeable improvement. This indicates that larger models gain a greater advantage from our skip-layer attention method.

The training time with our skip-layer attention method is slightly longer than the baseline, experiencing a maximum decrease in training speed of 2.34%. This is likely due to the additional storage requirements for keys and values in the lower layers.

## 6 Conclusion

In this paper, we propose a Skip-Layer Attention (SLA) mechanism to enhance the Transformer architecture's ability to capture complex dependencies within input data. By enabling direct attention between non-adjacent layers, our approach improves the model's capacity to integrate high-level abstract features with low-level details without significantly increasing computational complexity. Extensive experiments demonstrate that our enhanced Transformer model outperforms standard Transformer baselines in language modeling tasks, validating the effectiveness of our method. Our findings pave the way for further innovations in optimizing neural network architectures.

## 7 Limitation

While our research has demonstrated the effectiveness of skip-layer attention in Transformer models, several avenues for future work remain to be explored:

**Scaling to Larger Models**: Future research could extend skip-layer attention to larger Transformer architectures, such as GPT-3 (Brown et al., 2020) and beyond, to assess its effectiveness across different scales and complexity levels. Evaluating the performance and efficiency on these larger models will offer valuable insights into its scalability.

**Real-World Applications**: Evaluating the skip-layer attention mechanism in various real-world applications, such as natural language understanding (Hendrycks et al., 2021), general question answering (Rein et al., 2023), coding (Chen et al., 2021), and mathematics (Cobbe et al., 2021), will be critical to fully understand its practical benefits and limitations.

**Beyond Text**: Extending the applicability of the skip-layer attention mechanism to other domains, such as computer vision and speech processing, will help determine its versatility and potential for cross-modal impact.

**Ablation Studies**: Conducting comprehensive ablation studies to understand the contributions of different components within the skip-layer attention mechanism could provide deeper insights. For instance, exploring the impact of connections to multiple preceding layers rather than just one could reveal additional enhancements and inform design choices.

## References


Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. GQA: training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 4895–4901. Association for Computational Linguistics.

William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan Kelly. 2024. Reducing transformer key-value cache size with cross-layer attention. *Preprint*, arXiv:2405.12981.

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, Jeff 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. *ArXiv*, abs/2005.14165.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2023. Theoremqa: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 7889–7901. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *CoRR*, abs/2307.08691.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision* *and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2017. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269. IEEE Computer Society.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. 2022. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pretraining.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *CoRR*, abs/1911.02150.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

Josh Tenenbaum. 2018. Building machines that learn and think like people. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, page 5. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM.

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 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1480–1489. The Association for Computational Linguistics.

# A Appendix

## A.1 Code

In this paper, the model code is based on nanoGPT, which is publicly available on GitHub[3]. The implementation of the skip-layer attention method is detailed in the code listing provided below.

```python
import torch.nn.functional as F

class CausalSelfSkipLayerAttention(nn.Module):

    def __init__(self, config):
        super().__init__()
        assert config.n_embd % config.n_head == 0
        # key, query, value projections for all heads, but in a batch
        self.c_attn = nn.Linear(config.n_embd, 3 * config.n_embd, bias=config.bias)
        # output projection
        self.c_proj = nn.Linear(config.n_embd, config.n_embd, bias=config.bias)
        # regularization
        self.attn_dropout = nn.Dropout(config.dropout)
        self.resid_dropout = nn.Dropout(config.dropout)
        self.n_head = config.n_head
        self.n_embd = config.n_embd
        self.dropout = config.dropout
        self.num_skip_layer = config.num_skip_layer
        self.split_num = config.n_head - config.num_skip_head

    def forward(self, x, prev_k_list=[], prev_v_list=[]):
        B, T, C = x.size()  # batch size, sequence length, n_embd

        q, k, v = self.c_attn(x).split(self.n_embd, dim=2)
        k = k.view(B, T, self.n_head, C // self.n_head).transpose(1, 2)  # (B, nh, T, hs)
        q = q.view(B, T, self.n_head, C // self.n_head).transpose(1, 2)  # (B, nh, T, hs)
        v = v.view(B, T, self.n_head, C // self.n_head).transpose(1, 2)  # (B, nh, T, hs)

        # causal self-attention; Self-attend: (B, nh, T, hs) x (B, nh, hs, T) -> (B, nh, T, T)
        # efficient attention using Flash Attention CUDA kernels
        if len(prev_k_list) >= self.num_skip_layer and len(prev_v_list) >= self.num_skip_layer:
            k_combine = torch.cat([k[:, :self.split_num, :, :], prev_k_list[-self.num_skip_layer]], dim=1)
            v_combine = torch.cat([v[:, :self.split_num, :, :], prev_v_list[-self.num_skip_layer]], dim=1)
            y = F.scaled_dot_product_attention(q, k_combine, v_combine, attn_mask=None,
                                               dropout_p=self.dropout if self.training else 0,
                                               is_causal=True)
        else:
            y = F.scaled_dot_product_attention(q, k, v, attn_mask=None,
                                               dropout_p=self.dropout if self.training else 0,
                                               is_causal=True)

        y = y.transpose(1, 2).contiguous().view(B, T, C)
        # output projection
        y = self.resid_dropout(self.c_proj(y))
        return y, k[:, self.split_num:, :, :], v[:, self.split_num:, :, :]
```

[3] https://github.com/karpathy/nanoGPT/