Adaptive Layer-skipping in Pre-trained LLMs

Xuan Luo, Weizhi Wang, Xifeng Yan UCSB

open-sourced

COLM

Concept

Vanilla Decoder-only Transformer

- **Uniform Effort**: Every token is generated using the same number of transformer layers
- **Example**: It takes the same effort to generate a token for copying a string as for solving complex math problems.
- **Opportunity**: Underutilized computation can be reduced by skipping unnecessary layers.

Our Idea: FlexiDepth

- **Adaptive Layer Allocation**: FlexiDepth dynamically skips transformer layers based on the input.
- **Example**: Tokens needing high computation or uncertainty use more layers, while repetitive tokens or fixed phrases need fewer:

[Context] The grey seal was found at Cruden Bay on Tuesday. The Scottish SPCA removed the seal from beach and took it to the charity's rescue centre... Thankfully there were no injuries in this case and we were able to release him relatively quickly.
[Prompt] Where the grey seal was found?
The grey seal was found at Cruden Bay.

[Prompt] Please summarize this paragraph into a single sentence:
A grey seal was found tangled in netting at Cruden Bay , rescued by the Scottish S PCA , and released back into the water with no injuries.

[Prompt] Please calculate the sum of the 8 numbers in the list: [99, 45, 12, 78, 33, 66, 21, 54]
To calculate the sum of all 8 numbers in the list, I'll add each number one by one:
99 + 45 = 144
144 + 12 = 156
156 + 78 = 234
234 + 33 = 267
267 + 66 = 333
333 + 21 = 354
354 + 54 = 408

The sum of all 8 numbers in the list is 408.

Layer-skipping patterns (Llama-3-8B-Instruct) for a language task (left) and a math task (right). The light-to-dark blue gradient represents layer usage from 16 to 32.

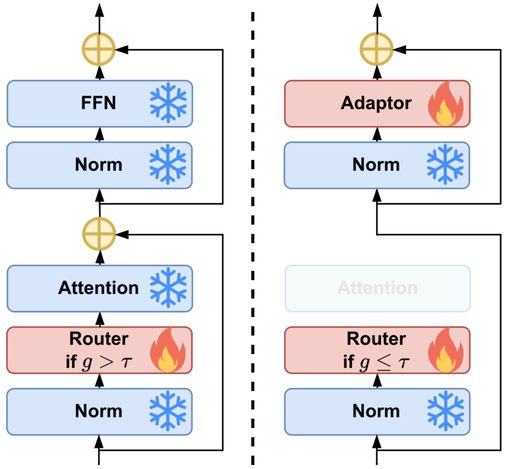
Overall Logic

To enable adaptive layer-skipping, we convert the latter half of vanilla decoder layers in a pre-trained LLM into FlexiDepth layers. Each FlexiDepth layer allows individual tokens to dynamically decide whether to skip the layer. By stacking multiple FlexiDepth layers, the model can adaptively allocate different number of layers.

Method

We introduce two lightweight components to each FlexiDepth layer: a router and an adapter.

- **Router**: A bottlenecked MLP that generates gating scores to decide if a token skips the layer.
- **Adapter**: A bottlenecked MLP that aligns skipped hidden states with processed ones.
- **Training**: Only the router and adapter are trainable; pre-trained LLM parameters remain frozen.



Left: Full-processing path where hidden states undergo the pre-trained attention and FFN modules. Right: Skipping path where hidden states bypass the attention module and processed by a lightweight adapter.

Let $X = [x_1, x_2, \ldots, x_T]$ denote the input hidden states. The router computes their corresponding gating scores $G = [g_1, g_2, \ldots, g_T]$ as:

$$G = \sigma(\text{Router}(\text{X})).$$

For each hidden states x_i, we use a predefined threshold τ for routing:

$$x_i' = \begin{cases} g_i \cdot \text{FFN}(\text{Norm}(\text{Attn}(\text{Norm}(x_i)) + x_i)) + \text{Attn}(\text{Norm}(x_i)) + x_i, & \text{if } g_i > \tau \\ (1 - g_i) \cdot \text{Adapter}(\text{Norm}(x_i)) + x_i, & \text{if } g_i \leq \tau \end{cases}$$

where x_i' is the output hidden state of this layer. When skipping the attention module, we still compute its KV Cache for sustained generation.

Loss Function

To balance efficiency and quality, we optimize a skipping loss alongside the next-token prediction loss. The skipping loss is:

$$\mathcal{L}_{\text{skip}} = \frac{1}{T} \sum_{t=1}^{T} \left(\sum_{l=1}^{L} g_t^l \right)^2$$

where g_t^l is the gating score for layer l at time step t.
The total loss is:
$\mathcal{L} = \alpha \cdot \mathcal{L}_{\text{skip}} + \mathcal{L}_{\text{lm}}$

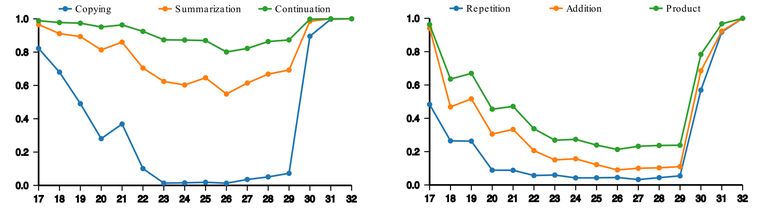
Results

Comparison

Methods	Single-Token Generation			Multi-Token Generation			Retain %
	MMLU	Hellaswag	Winogrande	GSM8K	HumanEval	CoQA	
Vanilla	0.673	0.706	0.744	0.679	0.299	0.784	100.0%
Skip 4 Layers							
LayerSkip	0.659	0.636	0.676	0.004	0.0	0.350	54.0%
ShortGPT	0.664	0.662	0.700	0.536	0.092	0.145	69.1%
LaCo	0.671	0.693	0.724	0.581	0.031	0.778	81.7%
MindSkip	0.664	0.698	0.722	0.378	0.189	0.720	84.2%
Ours	0.663	0.724	0.756	0.695	0.390	0.810	106.5%
Skip 8 Layers							
LayerSkip	0.650	0.525	0.640	0.0	0.0	0.049	43.9%
ShortGPT	0.307	0.462	0.597	0.001	0.0	0.005	32.0%
LaCo	0.656	0.628	0.695	0.065	0.006	0.707	65.3%
MindSkip	0.602	0.650	0.646	0.039	0.024	0.620	60.2%
Ours	0.616	0.705	0.735	0.662	0.341	0.801	100.7%

Performance comparison based on Llama-3-8B-Instruct, which consists of 32 layers. Retain % represents the percentage of average retained benchmark performance.

Layer-skipping Pattern



Percentage of tokens processed by transformer layers 17 to 32. The x-axis represents the layer index, and the y-axis represents the percentage of tokens processed by the layer.