Chain-of-Model Learning for Language Model

Kaitao Song

Microsoft Research Asia

Outline

- Background
- Chain-of-Model
- Architecture
- Experiments
- Conclusion

Outline

- Background
- Chain-of-Model
- Architecture
- Experiments
- Conclusion

Background

- **Achilles heel of scaling foundation model**
  - Scaling laws has been considered as the de facto approach to improve model capabilities. But is it perfect?
  - Do we always need a very large foundation model to handle any user instructions?
    - Human intelligent is activated by a dynamic paradigm, we are not activating all neurons of brains.
    - However, existing LLM paradigms (e.g., Dense or MoE) only support a fixed size activation, lacking flexibility and extensibility.

Background

- **Shortcoming of scaling law within training**
  - When we need to increase the scales of foundation model, it always requires us to train from scratch (e.g., 30B → 175B). Is it a correct way?
  - However, the learning stage within human intelligent is a continual learning. We can progressively or incrementally learn new knowledge.
  - So, why we cannot reuse knowledge from previous scales, but always train from scratch?

Background

- **Deficiency in unifying understand and generation**
  - Nowadays, a trend is to unify the understand and generation within one model. So, in existing paradigms:
    - Understanding means the states to handle input content (e.g., Keys / Values in Transformer), including the provided and generated data, refer to prefilling.
    - Generation means how to infer the outputs based on the conditional data, refer to decoding.
  - But does understanding necessitate similar capability as generation?
    - In human, understanding (e.g., Reading a book) is usually simpler than generation (e.g., writing a book).

Background

- **Motivation**
  - So, how can we design a new paradigm, enabling foundation models with better extensibility and flexibility, that satisfies:
    - It allow us to reuse capability from existing scales and enables us to progressively expand model scale.
    - It can offer multiple scales for elastic usage.
    - Besides, it will be better if it can support:
      - Faster Prefilling speed.
      - Dynamic LLM switching during the decoding, without any additional computations.

Outline

- Background
- Chain-of-Model
- Architecture
- Experiments
- Conclusion

Chain-of-Model

- **What is Chain-of-Model?**
  - Chain-of-Model (CoM) is a new scaling paradigm for foundation models, that allow progressive training, elastic inference and so on.
  - It applies the idea of chain into every part within model, from representation, to the layer and finally the model.
  - Chain-of-Representation → Chain-of-Layer → Chain-of-Model

Chain-of-Representation

- **Definition**

Definition 1 (*Chain-of-Representation*) *For representation $x \in \mathbb{R}^D$, it can always be equivalently formulated as a concatenation of n sub-representations, denoted as $\xi(x, n) = \{x_1, \ldots, x_n\}$, where $x_i \in \mathbb{R}^{d_i}$ and $\sum_{i=1}^{n} d_i = D$. We term this as chain-of-representation (CoR) of x.*

- By using different number of chains, we can encode information at different scales, and thus support multi-scale representation.

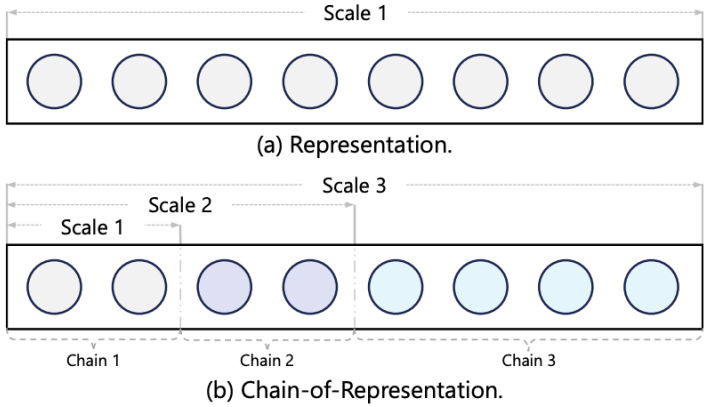


Figure 1: An example of CoR with 3 chains.

Chain-of-Representation

- **Definition**

Definition 1 (*Chain-of-Representation*) *For representation $x \in \mathbb{R}^D$, it can always be equivalently formulated as a concatenation of n sub-representations, denoted as $\xi(x, n) = \{x_1, \ldots, x_n\}$, where $x_i \in \mathbb{R}^{d_i}$ and $\sum_{i=1}^{n} d_i = D$. We term this as chain-of-representation (CoR) of x.*

- By using different number of chains, we can encode information at different scales, and thus support multi-scale representation.

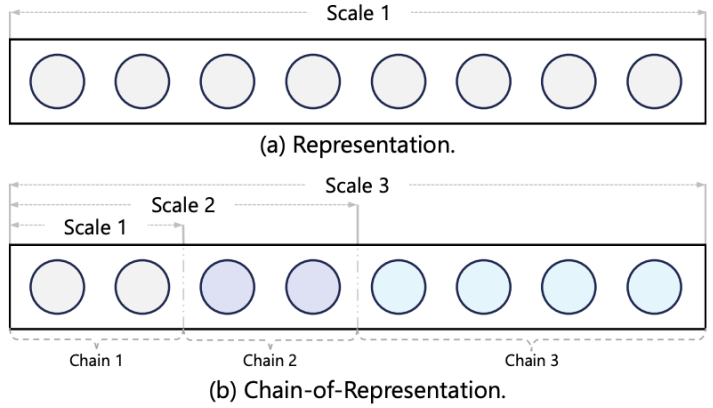


Figure 1: An example of CoR with 3 chains.

Challenge: How to model the connections between two CoR features?

Chain-of-Layer

- **Definition**

> **Definition 2 (*Chain-of-Layer*)** *For a layer $y = f_\theta(x)$, where its input x and output y can both be expressed as the chain-of-representation $\xi(x, n)$ and $\xi(y, n)$. We term it as a Chain-of-Layer (CoL) where $f_\theta(\cdot)$ satisfies that each y_i is only conditioned on $x_{\leq i}$.*

  - That means, for $x = \{x_1, \dots, x_n\}$ and $y = \{y_1, \dots, y_n\}$, both with n chains. In Chain-of-Layer, each y_i is only built upon $x_{\leq i}$.

  - CoL possesses three properties: *Generality*, *Causality*, *Compositionality*.

Chain-of-Layer

- **Generality**
  - Any layer can be considered as a specific case of Chain-of-Layer when the number of chains is 1.
  - Therefore, any layer can be extended to multiple chains, by increasing the number of chains and reusing previous chains.

Corollary 2.1 (*Generality*) *Any layer can be viewed as a case of chain-of-layer when* $n = 1$.

If $n = 1$, CoL can be considered as the standard network layer to process the conventional representation. In other words, any network layer can be generalized to the form of CoL. Therefore, we can gradually increase layer dimension by introducing additional chains upon the existing chains.

Chain-of-Layer

- **Causality**
  - Based on the causality of CoL, to obtain the feature of scale i, it only requires the activation of previous chain $\leq i$.
  - Therefore, it allows us to dynamically calculate the feature at any scale i within one representation.

Corollary 2.2 (*Causality*) *If a layer $y = f(x)$ satisfies the property of chain-of-layer, that means its weights θ can be partitioned into n independent parts $\{\theta_1, \ldots, \theta_n\}$, where each θ_i is used to compute y_i based on $x_{\leq i}$. Therefore, y allows the causal computation, i.e., to obtain feature y_i, we only need to compute $\theta_{\leq i}$ based on $x_{\leq i}$.*

Chain-of-Layer

- **Compositionality**
  - For any two layers, if both of them conform to the chain-of-layer properties, its compositionality also satisfies the setting of CoL.
  - Therefore, it allows us to generalize the setting of Chain-of-Layer to the scope of model level (i.e., Chain-of-Model).

Corollary 2.3 (*Compositionality*) *Assume we have two layers* $y = f_1(x)$ *and* $z = f_2(y)$*, where* $x = \xi(x, n)$*,* $y = \xi(y, n)$ *and* $z = \xi(z, n)$*. If both* f_1 *and* f_2 *adhere to the requirements of CoL, their composition* $z = f_2(f_1(x))$ *also satisfy the setting of CoL, i.e., each* z_i *only depends on* $x_{\leq i}$*.*

Chain-of-Model

- **Definition**

> **Definition 3 (*Chain-of-Model*)** *For a model θ with L layers, we define it as a* ***chain-of-model*** *(CoM) if each of its layers, from the first to the final, conforms to the chain-of-layer property.*

  - If a model, from its first layer to the final layer, both conform to the property of Chain-of-Layer, it also inherit any properties of Chain-of-Layer. For example,
    - Any model can be considered as a special case of Chain-of-Model, when n = 1.
    - To obtain the features / logits of scale i, it only requires us to activate the weights of the corresponding chains.

Outline

- Background
- Chain-of-Model
- Architecture
- Experiments
- Conclusion

Architecture

- Based on the design of CoM, if we can revise each layer of Language model to satisfy the setting of CoL, the final language model is capable of properties of CoL.

- So, we propose Chain-of-Language-Model (CoLM), that apply CoM into Linear, and each module within Transformer (e.g., Self-Attention, FFN, Normalization and so on).

Linear

- Linear serves as the fundamental layer in neural network, to transform input features. Its standard mathematic expression is: $y = Wx + b,$ where $W \in \mathcal{R}^{D_y \times D_x}$, and $b \in \mathcal{R}^{D_y}$.

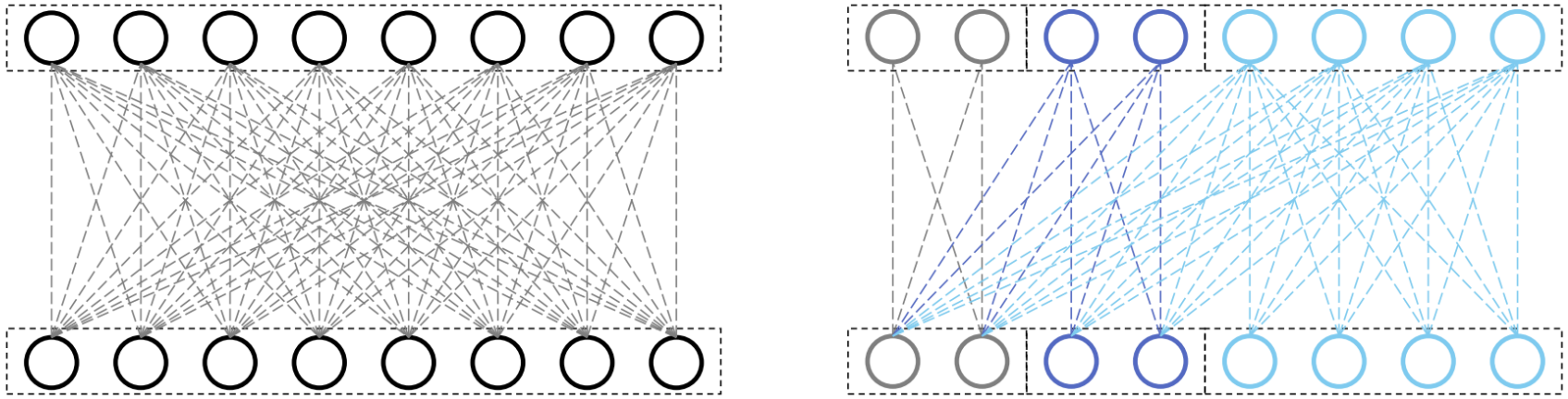
- For linear to process CoR, we denote $x = \{x_1, \dots, x_n\}$ and $y = \{y_1, \dots, y_n\}$. So, its formulation is as:

$$y = \|_{i=1}^{n} (y_i) = \|_{i=1}^{n} (W_i x_{\leq i} + b)$$

- where $\|$ means the concatenate operation. And here, we introduce a hyper-parameter $C = \{c_1, \dots, c_n\}$ to determine the size of $x_i \in D^{x_i}$ and $y_i \in D^{y_i}$, where $D_{x_i} = \frac{c_i}{\sum C} D_x$, and $D_{y_i} = \frac{c_i}{\sum C} D_y$.

Linear

- Comparisons



(a) Linear. (b) Chain of Linear.

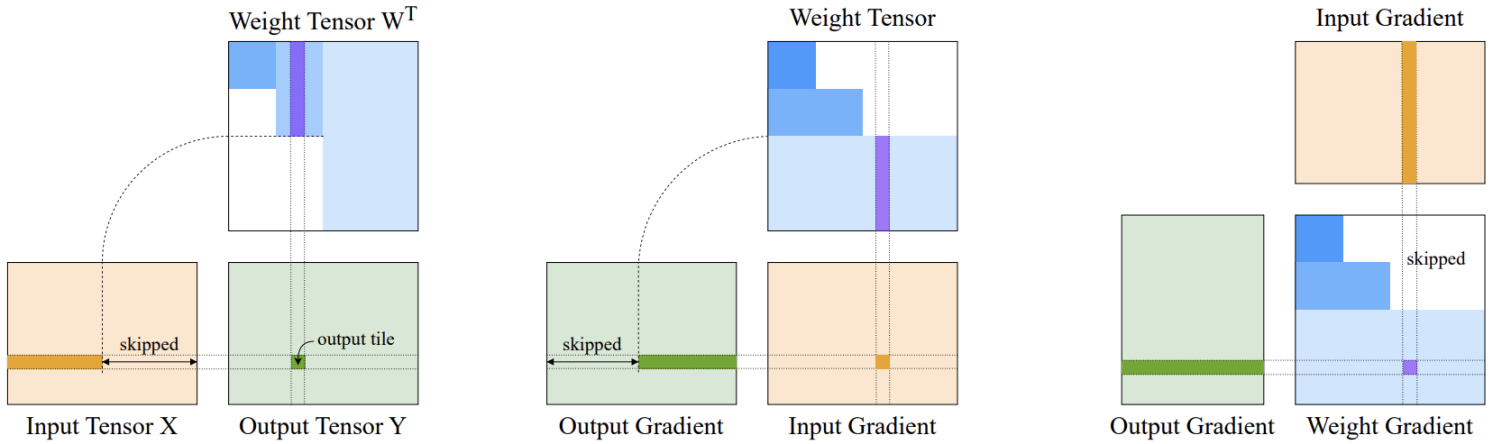
Figure 2: Comparisons between Linear and Chain-of-Linear layer.

Linear (optimized)

- Please note, although Chain-of-Linear introduce fewer parameters, the naïve implementation for training is slow due to more data access and all-reduce operation.

- Therefore, we also introduce a block-wise sparse kernel implemented by Triton for acceleration.

Linear (optimized)



(a) Forward. (b) Backward for input gradient. (c) Backward for weight gradient.

Figure 8: Block-sparse GeMM kernel for Linear (Chain) layer.

Table 15: Speed comparisons between different kernels. The "fwd", "bwd_w" and "bwd_i" represent the speed of forward, backward (weight) and backward (inputs). Chain-of-Linear (Naive) refers to Algorithm 1, and Chain-of-Linear (Triton) corresponds to our block-wise sparse kernel implementation.

Linear (optimized)

- Speedup Comparisons

Table 15: Speed comparisons between different kernels. The "fwd", "bwd_w" and "bwd_i" represent the speed of forward, backward (weight) and backward (inputs). Chain-of-Linear (Naive) refers to Algorithm 1, and Chain-of-Linear (Triton) corresponds to our block-wise sparse kernel implementation.

			Computation		
Implementation	Dim	Params	fwd	bwd_w	bwd_i
$\mathcal{C} = \{8, 8, 8, 8\}$, *tensor size is (4096, 4096).*					
Linear	4096	167K	0.67	0.62	0.64
Chain-of-Linear (Naive)	4096	104K	0.58	0.49	0.72
Chain-of-Linear (Triton)	4096	104K	0.47	0.46	0.56
$\mathcal{C} = \{4, 4, 4, 4, 4, 4, 4, 4\}$, *tensor size is (8192, 8192).*					
Linear	8192	671K	1.56	1.61	1.56
Chain-of-Linear (Naive)	8192	419K	0.96	0.84	1.82
Chain-of-Linear (Triton)	8192	419K	0.77	0.76	0.90

Attention

- Standard Formulation

$$\mathrm{MHA}(x) = \mathrm{O}\big(\|_{i=1}^{h} \mathrm{Attention}(q_i, k_i, v_i)\big) = \mathrm{O}\big(\|_{i=1}^{h} \mathrm{softmax}(\frac{q_i k_i^T}{\sqrt{d_k}}) v_i\big), \quad (2)$$
$$\text{where } \|_{i=1}^{h} q_i = \mathrm{Q}(x), \|_{i=1}^{h} k_i = \mathrm{K}(x), \|_{i=1}^{h} v_i = \mathrm{V}(x),$$

- In the above function, MHA contains 4 linear layers, which includes Q, K, V, O. Hence, we just need to replace all Linear by Chain-of-Linear layer.

- However, we also need to guarantee the dot product also adheres to the setting of CoL.

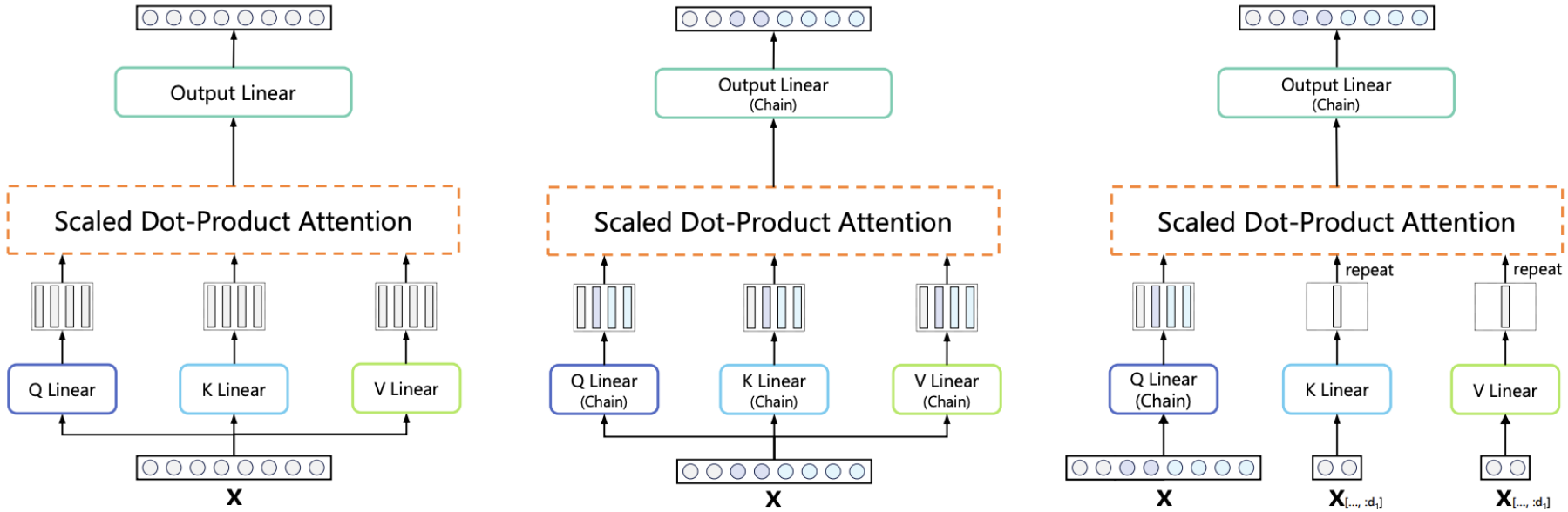
Attention

- For $Attention(q, k, v)$, if a single head (q_i, k_i, v_i) contains information from 2 or more chains, its output will blend these information and break the rules of CoL.

- Therefore, we expect each head only contains information from single scale. To this end, we just need to ensure $\sum C = h$, so that each c_i represent how many heads are assigned to chain i.

Attention

- Implementation

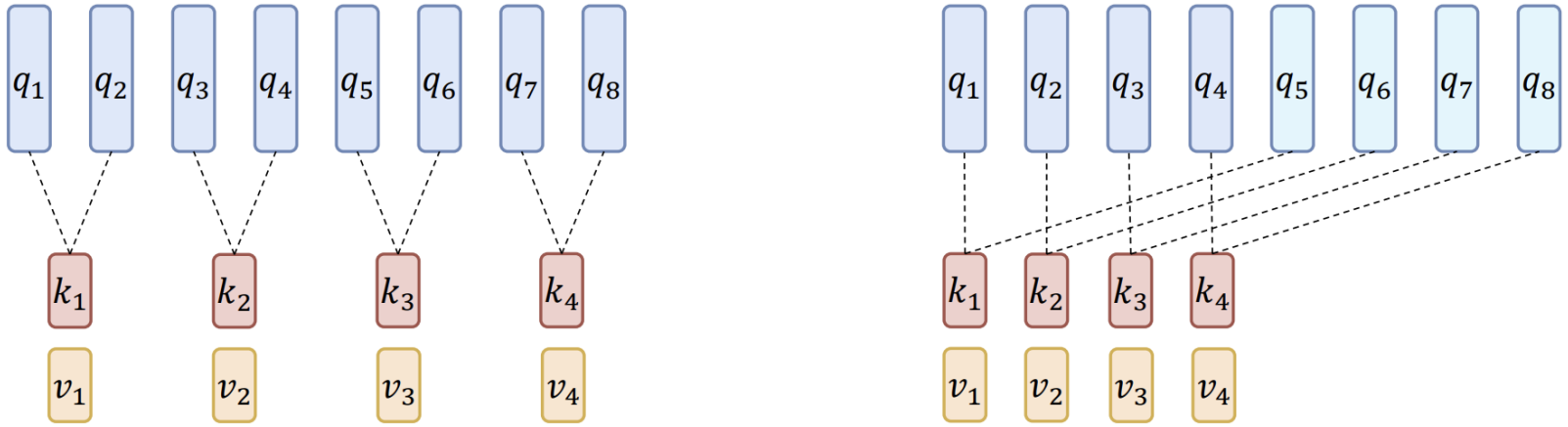


(a) Attention. (b) Chain of Attention. (c) Chain of Attention w/ KV sharing.

Figure 3: Differences between Attention, Chain of Attention and Chain of Attention with KV sharing.

Attention

- Implementation



(a) *Group Query Attention.* (b) *Attention (Chain) w/ KV sharing.*

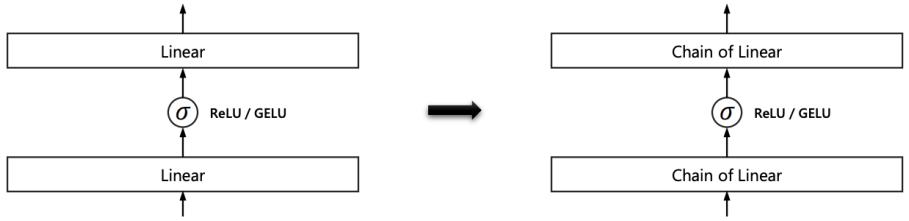
Figure 9: Differences between Group Query Attention (GQA) and Attention (Chain) with KV sharing in the attention implementation.

Attention (Extension)

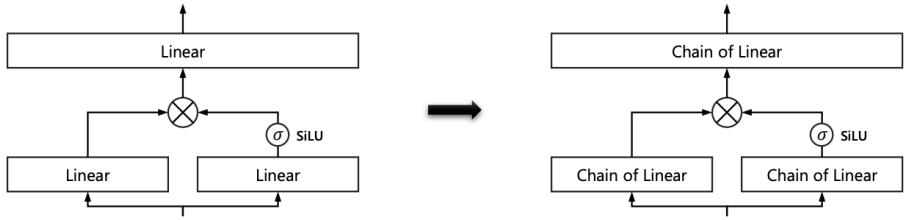
- **KV Sharing**
  - Based on the above design, each chain hold its dedicated keys and values.
  - If we want to use the largest scales or switch different scales during the decoding, we also need to calculate all weights. So is it possible to bridge the connections between different chains?
    - Let's require all keys and values to be calculated within the first chain.
    - It can be considered as a case of y_i is only conditioned on x_1. And thus, it also satisfy the requirements of CoL.

FeedForward Network

- FFN is usually composed of multiple linear layer, with the connected activated function. So, we just need to replace all linear as Chain-of-Linear.



(a) FFN (ReLU / GELU).



(b) FFN (SwiGLU).

Figure 10: FFN for CoR. We replace all Linear by using Chain-of-Linear layer.

Normalization

- A simple trick for normalization is to apply normalization over each chain.

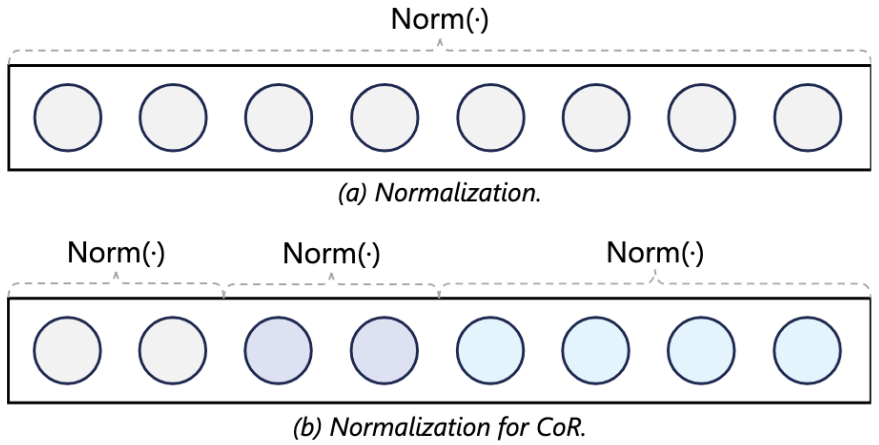


Figure 4: Normalization (CoR).

Embedding

- For Embedding, which often acts as the first layer, we do not involve any specific design, but just activate the specific chains when using it.

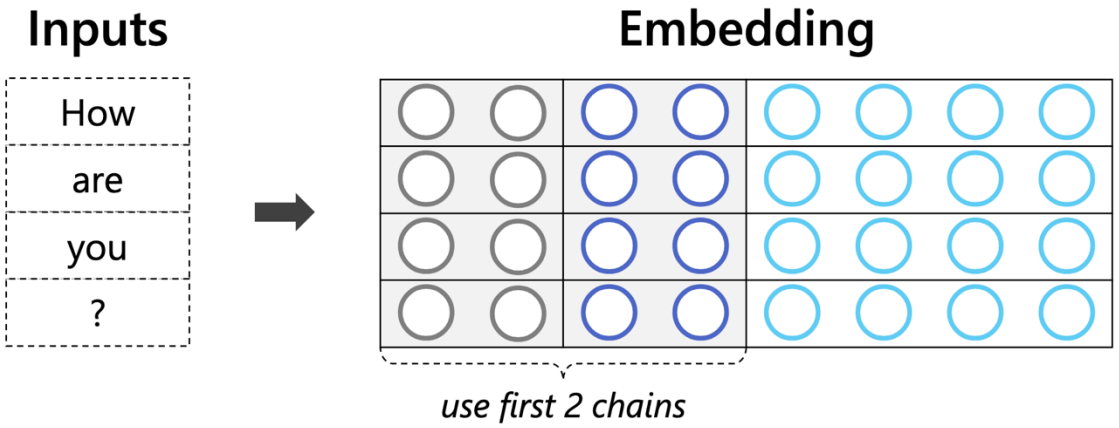


Figure 11: Example of Embedding in CoLM architecture. In this example, we design the number of chains as 3, while the dimensions for each chain are set as $\{2, 2, 4\}$. If we expect to use the first 2 chains, we only need to activate the first 4 neurons for each word in the embedding layer.

Objective Function

- For the output features $x \in R^D$ from CoLM, we can use cross-entropy loss. But we also need to guarantee each chain for prediction.

- Motivated by MRL loss, we respectively calculate the objective function of each chain, as $Loss_i = \mathcal{L}(W^i x_{\leq i})$.

Outline

- Background
- Chain-of-Model
- Architecture
- Experiments
- Conclusion

Experiments

- Performance

Table 1: Performance of baseline and CoLM on commonsense reasoning tasks. Models are trained for 50K steps. Each task is reported by acc_norm metric.

Chain($\mathcal{C}$)	**Dims**	**Params**	**HellaSwag↑**	**Obqa↑**	**WinoGranda↑**	**ARC-e↑**	**ARC-c↑**	**Boolq↑**	**Piqa↑**	**Avg↑**
{32}	2048	1.10B	40.01	31.19	52.72	43.52	23.63	57.43	67.30	45.11
CoLM										
{16, 16}	2560	1.11B	40.25	31.39	52.41	43.73	23.81	58.01	67.30	45.27
{16, 16}	2048	0.86B	37.12	29.18	51.07	40.82	22.61	61.74	65.48	44.00
{8, 8, 8, 8}	3072	1.18B	38.63	31.99	51.62	42.80	23.89	56.70	66.00	44.51
{8, 8, 8, 8}	2048	0.74B	34.49	27.57	50.20	39.02	22.78	57.65	63.00	42.10
CoLM-Air										
{16, 16}	2560	1.11B	39.85	31.19	52.09	44.30	23.63	56.72	66.76	44.80
{16, 16}	2048	0.86B	36.82	28.77	51.62	49.19	22.70	61.31	65.94	43.90
{8, 8, 8, 8}	3072	1.18B	36.98	29.78	49.80	41.46	24.57	55.81	65.51	43.41
{8, 8, 8, 8}	2048	0.74B	33.77	27.97	49.41	38.93	22.61	57.65	62.10	41.77

Experiments

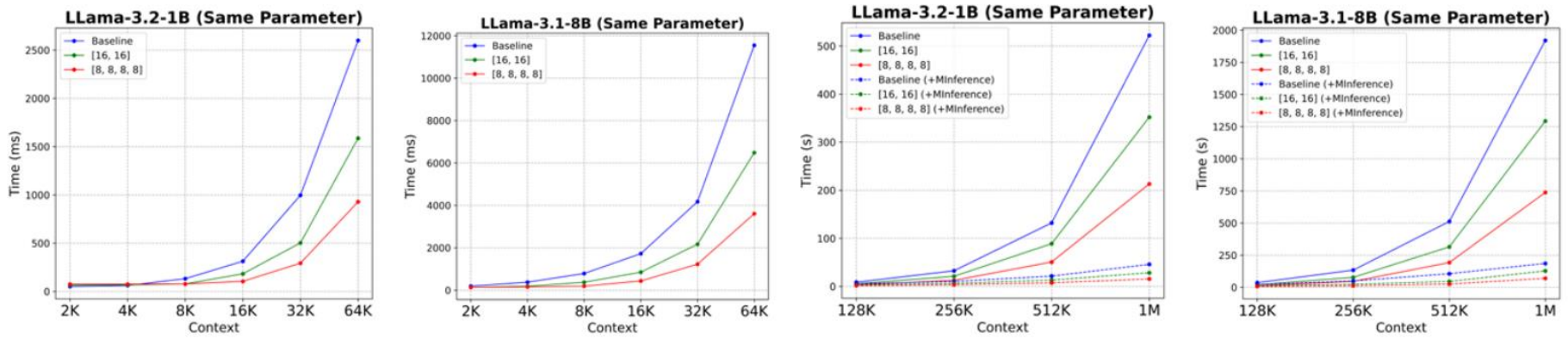
- Model Expansion

Table 2: Performance of baseline versus expanded models on commonsense reasoning tasks.

Model	HellaSwag↑	Obqa↑	WinoGranda↑	ARC-e↑	ARC-c↑	Boolq↑	Piqa↑	Sciq↑	Avg↑
Tiny-LLaMA-v1.1									
Baseline	61.47	36.62	59.43	55.47	32.68	55.99	73.56	84.20	57.43
+ Expansion	61.66	36.62	60.62	56.27	32.94	58.44	74.05	86.20	**58.35**
LLaMA-3.2-1B									
Baseline	63.78	36.42	59.83	60.56	36.03	63.70	74.37	88.40	60.39
+ Expansion	63.19	37.02	60.69	60.31	36.69	63.55	74.65	88.20	**60.53**

Experiments

- **Prefilling**



(a) Llama-1B (2K - 64K). (b) Llama-8B (2K - 64K). (c) Llama-1B (128K - 1M). (d) Llama-8B (128K - 1M).

Figure 5: Comparisons on Prefilling speed on LLaMa-1B and LLaMa-8B settings (2K - 1M).

Experiments

- **Elastic Inference**

Table 3: Elastic Inference to offer multiple sub-models by using different number of chains.

Scale	Params	HellaSwag	Obqa	WinoGranda	ARC-e	ARC-c	Boolq	Piqa	Avg
CoLM-Air, $\mathcal{C} = \{16, 16\}$, Dims $= 2048$									
Chain 1 + 2	0.86B	36.82	28.77	51.62	40.19	22.70	61.31	65.94	43.90
Chain 1	0.33B	29.31	25.75	52.96	34.50	22.01	62.23	61.15	41.13

- **Chain Tuning**

Table 4: GLUE dev set results. Each task is reported by accuracy. "+ CT" means Chain Tuning, and the values in brackets mean the fine-tuned parameters and the total parameters.

Model	SST-2	COLA	MNLI	MRPC	QNLI	QQP	RTE	WNLI	Avg.
Baseline	67.89	32.98	35.87	49.02	51.77	51.65	54.87	43.66	48.46
+ CT (0.8B/1.9B)	92.32	54.87	82.26	62.01	82.87	55.22	59.21	53.52	**67.79**

Outline

- Background
- Chain-of-Model
- Architecture
- Experiments
- Conclusion

Conclusion

- In this paper, we introduce a new paradigm, termed Chain-of-Model, to enable foundation model with better extensibility and flexibility in processing multi-scale information.
- We extend the idea of CoM into each layer within Transformer, and propose Chain-of-Language-Model (CoLM).
- Experimental results also demonstrate the capability of CoLM, and both exhibit better extensibility and flexibility.

Discussion

- **Depth-up vs Width-up**
  - Depth-up means increase the number of layers, while CoM can be considered as a kind of width-up scaling, to increase the feature dimension.
  - Depth-up is better at feature learning and can reuse previous weight, but it cannot retain the logits (i.e., the prediction) at the model level.
  - Width-up is better at extensibility, so that it can perfectly preserve knowledge from previous method.
  - Depth-up is compatible with width-up.

Discussion

- **CoM vs MoE**

Table 14: Comparisons between MoE and CoM from different aspects.

	Mixture-of-Expert (MoE)	**Chain-of-Model (CoM)**
Expert Capability	Equivalent	Weak $\rightarrow$ Strong
Expert Range	FFN	Full Model
Expert Activation	Sparse	Nested

- Besides, CoM is completely orthogonal to MoE.

Discussion

- **CoM vs NAS**
  - Neural Architecture Search (NAS) is a classical algorithm to offer elastic inference. It built a Super-Net, and then require sampling policy to obtain sub-net.
  - CoM integrate multi-scale model within one model, and can train it jointly, without any sampling policy.

Discussion

- **Limitations**
  - Infrastructure
    - the chain-of-layer is compatible with DP, PP and CP, but not well-suited for tensor parallelism (TP).
    - Chain-of-Layer can be considered as an imbalanced tensor, so it may require well-designed parallelism.
  - Optimal Setting
    - Chain-of-Model ensure each model can be considered as a case of CoM. Hence, any scaling laws can be applied to the CoM setting.
    - But, how to determine the optimal setting of each chain need exploration. Specifically, each chain will affect its latter chains.