1

EXPLORATION of LLM

Sihao Lin

CONTENTS

01

WEAK-TO-STRONG GENERALIZATION: ELICITING STRONG CAPABILITIES WITH WEAK SUPERVISION

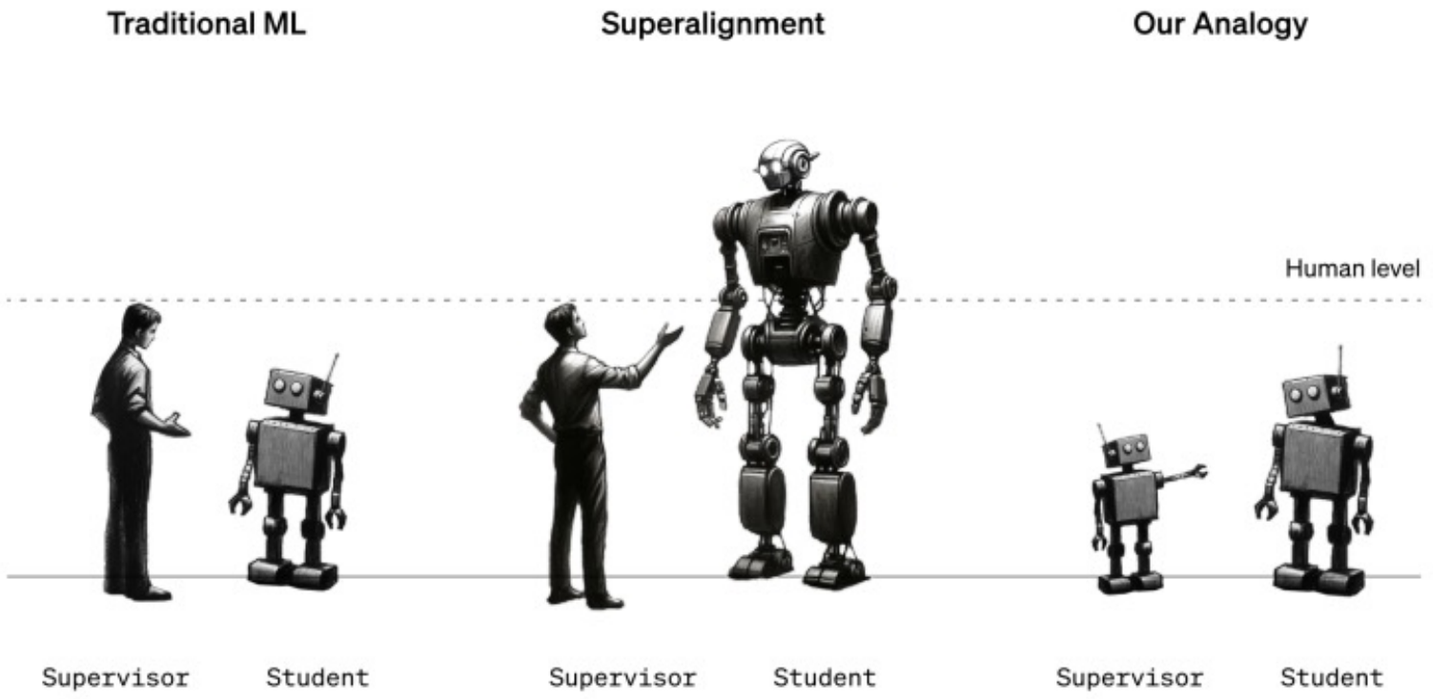
Background

Limitation of human evaluators:

superhuman models will be capable of complex and creative behaviors that humans cannot fully understand.

Problem:

how can weak supervisors control models much smarter than them?



Methodology

replace the weak human supervisor with a weak model supervisor

1. **Create the weak supervisor.**
   create weak supervisors by finetuning small pretrained models on ground truth labels and generate weak labels by taking the weak model's predictions on a held-out set of examples

2. **Train a strong student model with weak supervision.**
   finetune a strong model with the generated weak labels.

3. **Train a strong model with ground truth labels as a ceiling.**
   finetune a strong model with ground truth labels.

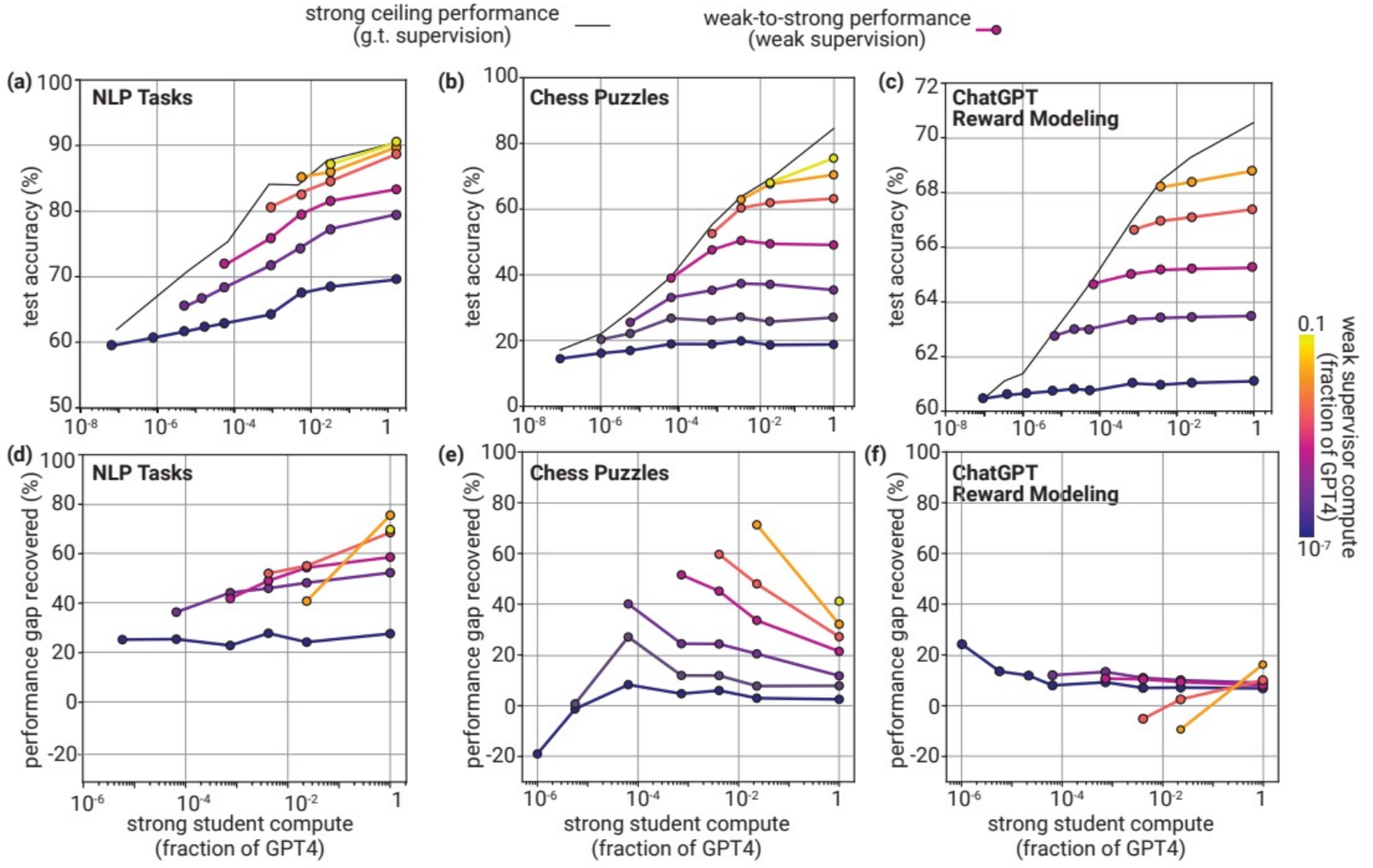
define the performance gap recovered (PGR) to measure the fraction of the performance gap

$$\text{PGR} = \frac{\text{weak-to-strong} - \text{weak}}{\text{strong ceiling} - \text{weak}} = \frac{\text{——}}{\cdots\cdots}$$

Result for naïve finetuning on weak labels

Task : Popular natural language processing benchmarks, Chess puzzles, ChatGPT reward modeling.

Model : study pretrained language models from the GPT-4 family



Observation for naïve finetuning on weak labels

Advantage:

- strong models trained with weak supervision can often generalize to a substantially higher performance than the weak model itself.

Limitation:

- weak-to-strong generalization is poor by default in the ChatGPT reward model setting. We are usually only able to recover roughly 10% of the performance gap between the weak supervisor and the strong student. Even for relatively small gaps in compute between the weak and strong models, PGR almost never exceeds 20%

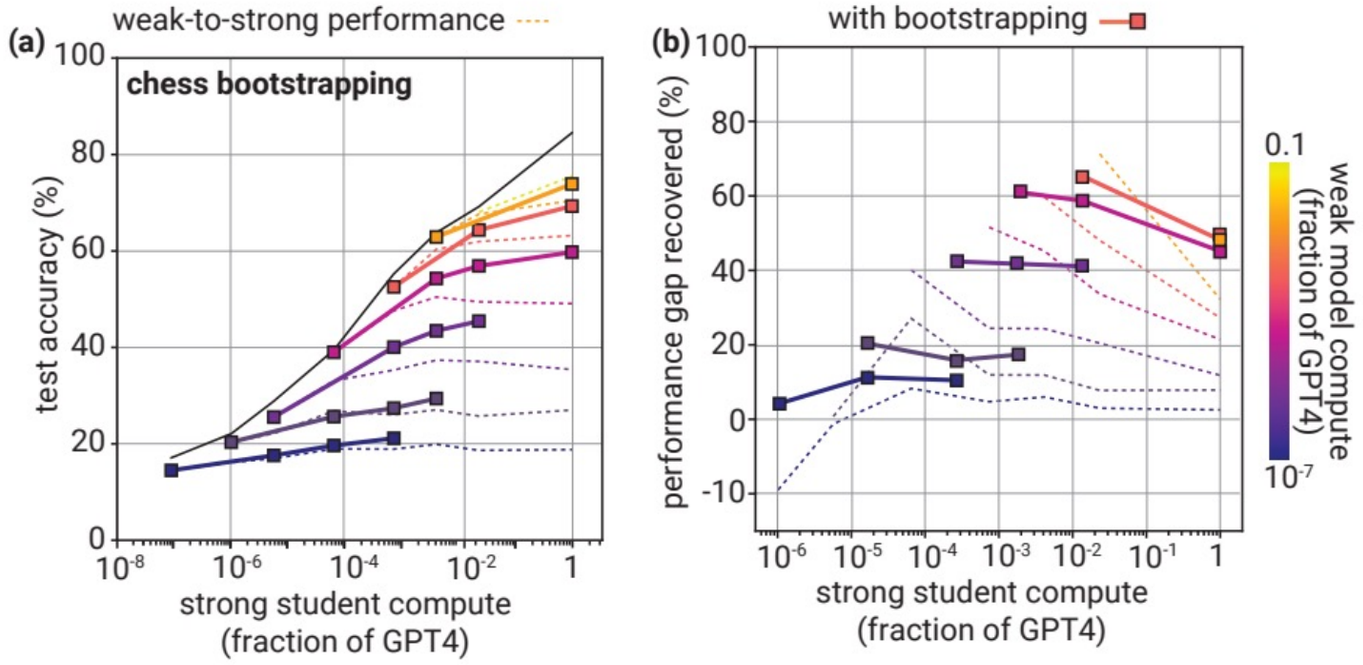
Improving weak-to-strong generalization

Bootstrapping with intermediate model sizes:

Instead of directly aligning very superhuman models, we could first align an only slightly superhuman model, use that to align an even smarter model

$$\mathcal{M}_1 \rightarrow \mathcal{M}_2 \rightarrow \ldots \rightarrow \mathcal{M}_n$$

use the weak labels from M1 to finetune M2, use M2 to generate new weak labels that we can use to finetune the next model in the sequence, M3, and so on



Improving weak-to-strong generalization

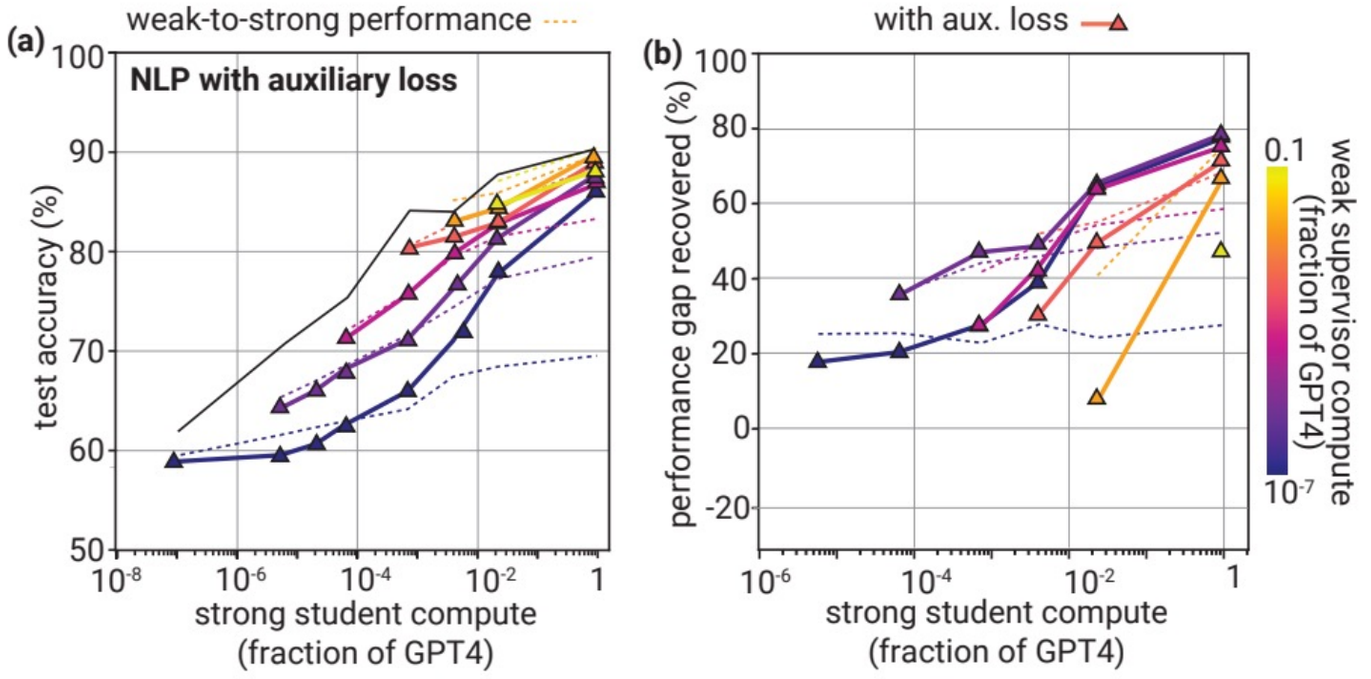
Use auxiliary confidence loss :

instead of directly aligning very superhuman models, we could first align an only slightly superhuman model, use that to align an even smarter model.

$$L_{\text{conf}}(f) = (1-\alpha)\cdot \text{CE}(f(x), f_w(x)) + \alpha \cdot \text{CE}(f(x), \hat{f}_t(x))$$

$$\hat{f}_t(x) = I[f(x) > t] \in \{0, 1\}$$

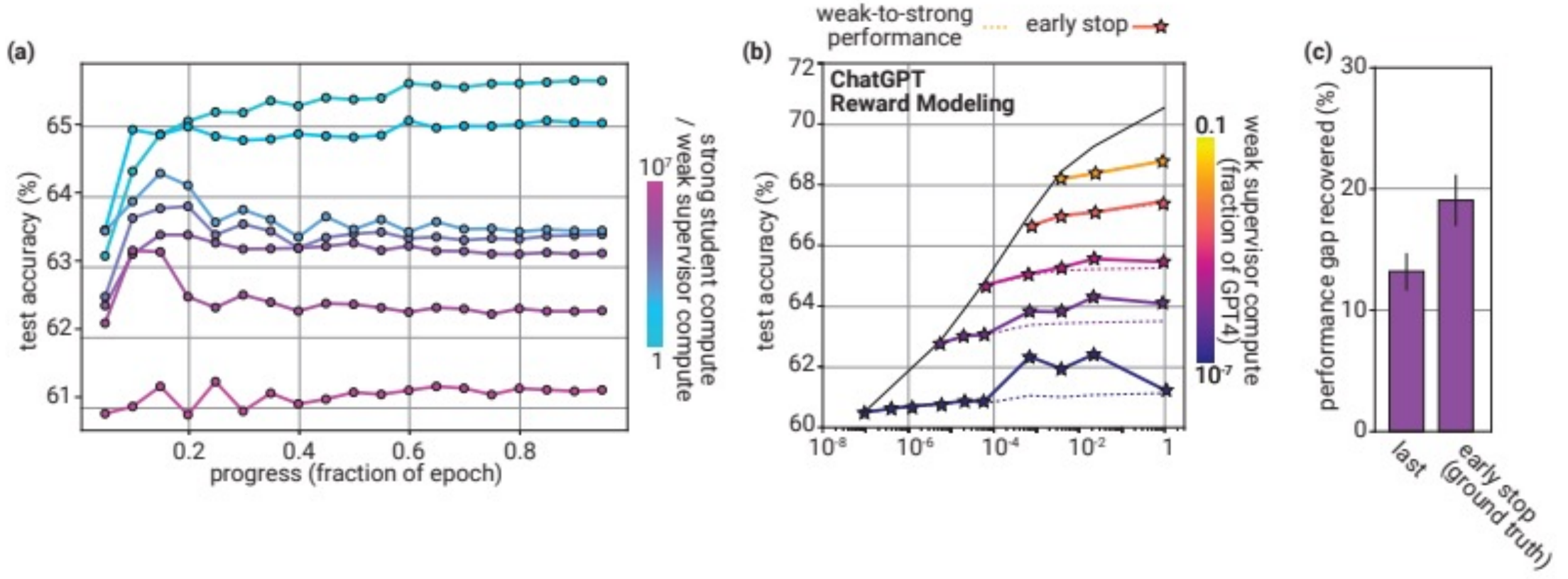
Parameter setting: set $\alpha_{\max} = 0.75$ for largest model
set $\alpha = 0.5$ for otherwise
linearly warm up in first 20% of training eposide



Understanding weak-to-strong generalization

Understanding imitation:
the strong model could simply learn to imitate the weak supervisor, including its errors

- **Overfit to weak supervision:** the strong model might overfit to the weak supervisor labels and its errors, degrading ground truth test accuracy over training even without classic overfitting to any specific training examples



Understanding weak-to-strong generalization

Saliency in the strong model representation:
strong pretrained models should already have good representations of the alignment-relevant tasks we care about.

- **Eliciting strong model knowledge with prompting：**

In particular, it is possible that strong pretrained models can solve many relevant tasks zero-shot with a simple prompt.

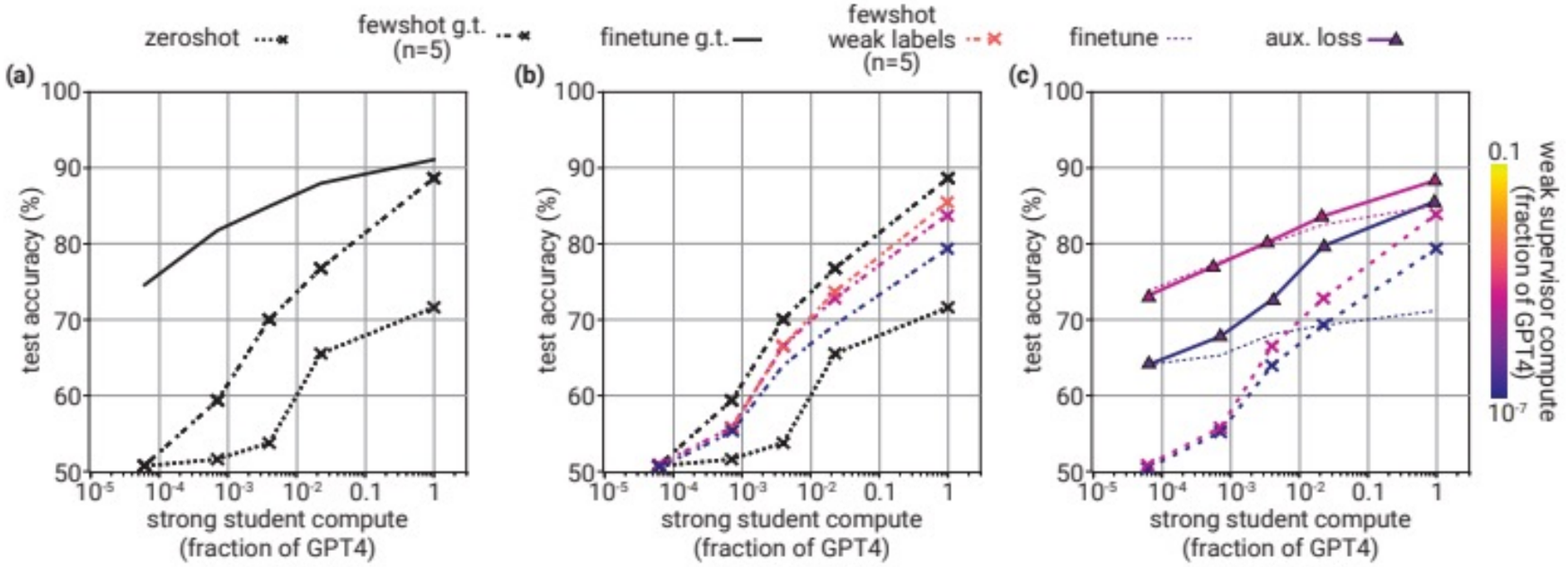


Figure 9: **Few-shot prompting becomes competitive with finetuning for large models; weak-to-strong learning is qualitatively similar in the prompting setting.**

Challenges and future work

Challenges:

Imitation saliency: superhuman models may easily imitate weak errors.

Pretraining leakage: superhuman knowledge may be latent, not observable.

Future work:

ANALOGOUS SETUPS

it is important that we have metrics which provide strong signal about whether we are making real progress toward the problem we ultimately care about.

SCALABLE METHODS

future work should identify additional unsupervised properties that can be used to specify the desired generalization

SCIENTIFIC UNDERSTANDING:

need a thorough understanding of precisely when and why our methods work.

02

Hungry Hungry Hippos: Towards Language Modeling with State Space Models

Background

Challenges:

- State space models (SSMs) have demonstrated state-of-the-art sequence modeling performance in some modalities(time series analysis, audio generation), but underperform attention in language modeling.

- specialized hardware support for attention, ranging from tensor cores to transformer chips but not state space models.

Contribution:
propose H3 (Hungry Hungry Hippo), a new SSM-based layer designed to solve these language modeling tasks.

Scaling SSMs to improve the efficiency of SSMs on modern hardware, to reduce the hardware barrier between attention and SSMs.

Background

State Space Models:
discrete-time state-space representation

$$x_i = \mathbf{A}x_{i-1} + \mathbf{B}u_i$$
$$y_i = \mathbf{C}x_i + \mathbf{D}u_i.$$

A state-space model (SSM) uses these representations as a layer in a deep learning pipeline

SSMs as Convolution:
given the entire sequence of the input u1,...,uN , the output sequence y1,...,yN can also be written as the convolution of the input with the filter

$$f = [\mathbf{CB}, \mathbf{CAB}, \mathbf{CA}^2\mathbf{B}, \ldots, \mathbf{CA}^{N-1}\mathbf{B}].$$

$$y_i = \mathbf{CA}^i\mathbf{B}x_0 + (f * u)_i + \mathbf{D}u_i, \quad \Longrightarrow \quad y = \mathrm{SSM}(u)$$

Background

Linear Attention:

General form of attention:

$$O_i = \frac{\sum_{j=1}^{i} \mathrm{Sim}(Q_i, K_j) V_j}{\sum_{j=1}^{i} \mathrm{Sim}(Q_i, K_j)} \in \mathbb{R}^d.$$

$$\begin{cases} Q = W^q x & Query\left(to\ match\ others\right) \\ K = W^k x & Key\left(to\ be\ matched\right) \\ V = W^v x & Value\left(information\ to\ be\ extracted\right) \end{cases}$$

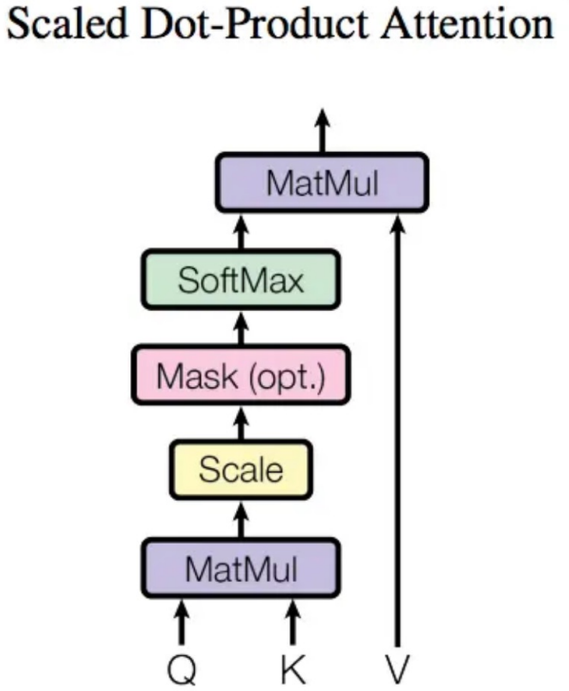
Consider as an weighted average of value

Softmax Attention:

$$\mathrm{Sim}(q, k) = e^{q^\top k} \quad \Rightarrow \quad Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})_i = \frac{\sum_{j=1}^{n} e^{\boldsymbol{q}_i^\top \boldsymbol{k}_j} \boldsymbol{v}_j}{\sum_{j=1}^{n} e^{\boldsymbol{q}_i^\top \boldsymbol{k}_j}}$$

Linear Attention:

$$\mathrm{Sim}(q, k) = \phi(q)^\top \phi(k) \quad \Rightarrow \quad O_i = \frac{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j) V_j^\top}{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j)}$$



Motivation

Synthetic Language Modeling Tasks: demonstrate the gap between SSM layer and attention layer

The Induction Head task tests how well a model can recall content after a special token

Associative Recall task tests how well a model can recall content after a special token key value pairs

Table 1: Synthetic language modeling tasks.

Task	Input	Output	Sequence Length	Vocab Size
Induction Head	$a\ b\ c\ d\ e \vdash f\ g\ h\ i \ldots\ x\ y\ z \vdash$	f	30	20
Associative Recall	$a\ 2\ c\ 4\ b\ 3\ d\ 1\ a$	2	20	10

Table 2: Evaluation of 2-layer models on synthetic language tasks.

Task	Random	S4D	Gated State Spaces	H3	Attention
Induction Head	5.0	35.6	6.8	**100.0**	**100.0**
Associative Recall	25.0	86.0	78.0	99.8	**100.0**

H3 layer

H3 uses SSMs with shift and diagonal matrices, along with multiplicative operations against projections of the input to capture the missing capabilities identified by the synthetics.

$\text{SSM}_{\text{shift}}$: use shift matrix as A in SSM, to create a "memory" of the previous states

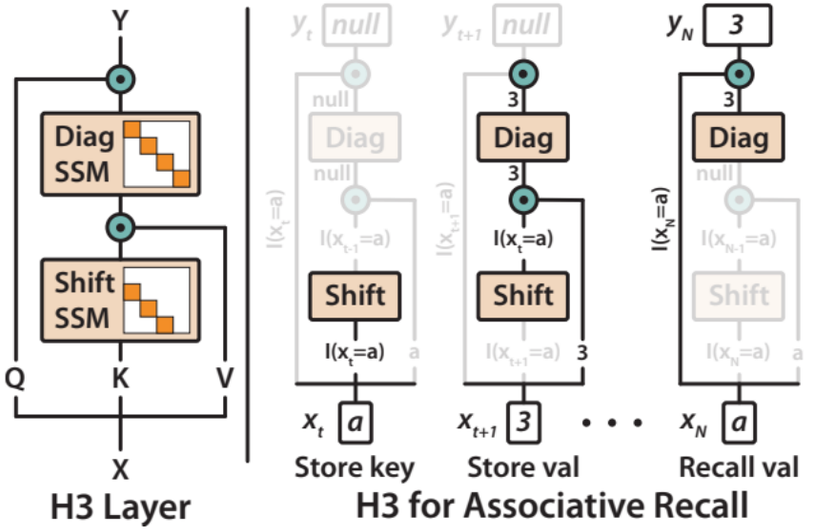
$$\text{mapping } [a, b, c] \to [0, a, b] \qquad A = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

SSM_{diag} : use diagonal matrix as A in SSM, to remember state over the entire sequence

$$\mathbf{Q} \odot \text{SSM}_{\text{diag}}(\text{SSM}_{\text{shift}}(\mathbf{K}) \odot \mathbf{V}),$$

$$O_i = \frac{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j) V_j^\top}{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j)}$$

$$x_i = \mathbf{A}x_{i-1} + \mathbf{B}u_i$$
$$y_i = \mathbf{C}x_i + \mathbf{D}u_i.$$



H3 layer

Algorithm 1 H3 Layer

Require: Input sequence $u \in \mathbb{R}^{N \times d}$ from the previous layer, weight matrices $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V, \mathbf{W}_O \in \mathbb{R}^{d \times d}$, a shift SSM $\mathrm{SSM}_{\mathrm{shift}}$, a diagonal SSM $\mathrm{SSM}_{\mathrm{diag}}$, head dimension d_h.

1: Compute $\mathbf{Q} = u\mathbf{W}_Q, \mathbf{K} = u\mathbf{W}_K, \mathbf{V} = u\mathbf{W}_V \in \mathbb{R}^{N \times d}$.

2: Pass $\mathbf{K}$ through the shift SSM: $\overline{\mathbf{K}} = \mathrm{SSM}_{\mathrm{shift}}(\mathbf{K}) \in \mathbb{R}^{N \times d}$.

3: Split $\mathbf{Q}, \overline{\mathbf{K}}, \mathbf{V}$ into H "heads" ($\mathbf{Q}^{(h)}, \overline{\mathbf{K}}^{(h)}, \mathbf{V}^{(h)}$ for $h = 1, \ldots, H$), each a sequence of N vectors of size $d_h = d/H$.

4: **for** $1 \le h \le H$ **do**

5: Take the batched outer product $\overline{\mathbf{K}}^{(h)}(\mathbf{V}^{(h)})^\top \in \mathbb{R}^{N \times d_h \times d_h}$ (batched in the N-dimension) and pass it through a diagonal SSM: $\mathbf{KV}^{(h)} = \mathrm{SSM}_{\mathrm{diag}}(\overline{\mathbf{K}}^{(h)}(\mathbf{V}^{(h)})^\top) \in \mathbb{R}^{N \times d_h \times d_h}$.

6: Batch-multiply by $\mathbf{Q}$: $\mathbf{O}^{(h)} = [\mathbf{Q}_1^{(h)}\mathbf{KV}_1^{(h)}, \ldots, \mathbf{Q}_N^{(h)}\mathbf{KV}_N^{(h)}] \in \mathbb{R}^{N \times d_h}$ (batched in the N-dimension).

7: **end for**

8: Concatenate the output $\mathbf{O}^{(h)}$ of each head, and multiply by the output projection matrix $\mathbf{W}_O \in \mathbb{R}^{d \times d}$.

H3 layer

Efficiency:

	H3	Attention
Time complexity	$O(d^2N + dN\log N)$	$O(N^2d)$
Space complexity	$O(dN)$	$O(N^2)$

Expressivity:

H3	H3 Hybrid (2 Attn)	S4D	GSS	GSS Hybrid (2 Attn)	Transformer
21.0	**19.6**	24.9	24.0	19.8	20.6

Perplexity of SSM variants compared to Transformers on OpenWebText

hybrid model simply retains two self-attention layers: one in the second layer, and one in the middle (layer 2 + N/2 for an N-layer model, N even)

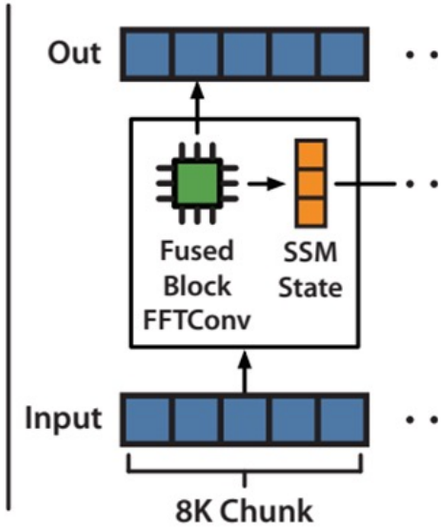
FlashConv : Efficiently training SSMs

For short sequences(<8K): by using kernel fusion, we can fuse the entire FFTConv into a single kernel and compute it in SRAM

Fused Block FFTConv:

Kernel fusion : Kernel fusion addresses IO bottlenecks due to reading and writing of intermediate results

Block FFT : the FFT-based convolution to utilize specialized matrix multiplication units.



FlashConv : Efficiently training SSMs

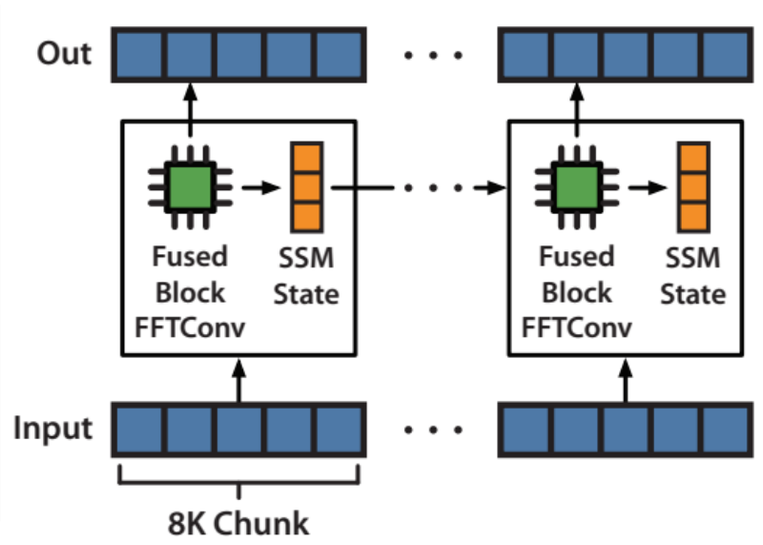
For long sequences(>8K): the computation no longer fits in GPU SRAM

State passing Algorithm:

Algorithm 2 State Passing Algorithm

Require: Input $u \in \mathbb{R}^N$, SSM parameterized by matrices $\mathbf{A} \in \mathbb{R}^{m \times m}$, $\mathbf{B} \in \mathbb{R}^{m \times 1}$, $\mathbf{C} \in \mathbb{R}^{1 \times m}$, $\mathbf{D} \in \mathbb{R}^{1 \times 1}$, chunk size N' where N is a multiple of N'.

1: Precompute $\mathbf{A}^{N'} \in \mathbb{R}^{m \times m}$, $\mathbf{M}_{ux} = [\mathbf{A}^{N'-1}\mathbf{B}, \ldots, \mathbf{B}] \in \mathbb{R}^{m \times N'}$, $\mathbf{M}_{xy} = [\mathbf{C}, \mathbf{CA}, \ldots, \mathbf{CA}^{N'-1}] \in \mathbb{R}^{N' \times m}$.
2: Split the inputs $u_{1:N}$ into $C = N/N'$ chunks $u^{(c)}_{1:N'}$ for $c = 1, \ldots, C$.
3: Let the initial state be $x^{(0)}_{N'} = 0 \in \mathbb{R}^m$.
4: **for** $1 \leq c \leq C$ **do**
5: Compute $y^{(c)} = \mathbf{M}_{xy} x^{(c-1)}_{N'} + \textsc{BlockFFTConv}(f, u_j) + \mathbf{D}u^{(c)} \in \mathbb{R}^{N'}$.
6: Update state: $x^{(c)}_{N'} = \mathbf{A}^{N'} x^{(c-1)}_{N'} + \mathbf{M}_{ux} u^{(c)}$.
7: **end for**
8: Return $y = [y^{(1)} \ldots y^{(C)}]$.



H3 Evaluation

Compare with language model:

Table 6: 3-shot acc. on SuperGLUE with logit scoring. Best results in bold, second best underline.

Model	WSC	WIC	RTE	CB	MultiRC	ReCoRD	BoolQ	COPA	Average
OPT-125M	36.5	**50.2**	47.3	44.6	**57.9**	44.9	41.9	60.0	47.9
GPT-Neo-125M	38.5	50.0	53.1	17.9	56.3	39.6	**62.1**	60.0	47.2
Hybrid H3-125M	**43.3**	49.1	**58.1**	**51.8**	48.9	**55.0**	56.1	**67.0**	**53.7**
GPT-2 medium (355M)	36.5	**50.5**	48.0	8.9	43.5	53.3	58.8	65.0	45.6
OPT-350M	37.5	50.0	45.8	**44.6**	49.8	51.4	**61.7**	60.0	50.1
Hybrid H3-355M	**42.3**	47.5	**50.5**	28.6	**59.7**	**62.3**	60.5	**69.0**	**52.6**
OPT-1.3B	**44.2**	**51.1**	53.4	16.1	**59.9**	62.1	38.3	70.0	49.4
GPT-Neo-1.3B	35.6	50.6	47.3	**32.1**	**59.9**	55.7	**61.2**	67.0	51.2
Hybrid H3-1.3B	36.5	49.2	**55.2**	23.2	59.3	**67.6**	56.9	**76.0**	**53.0**
OPT-2.7B	44.2	50.5	**53.4**	17.9	59.2	66.0	**62.0**	71.0	53.0
GPT-Neo-2.7B	**49.0**	**51.9**	51.6	21.4	57.0	60.0	56.0	68.0	51.9
Hybrid H3-2.7B	36.5	45.6	47.3	**46.4**	**59.4**	**71.1**	60.6	**77.0**	**55.5**

Language Modeling Inference

Table 7: Inference throughput on A100 80GB, 1.3B models. Batch size 64, prompt length 512, 1024, or 1536, and generating 128 tokens per sequence in the batch (i.e., 64 × 128 tokens in a batch). Hybrid H3 is up to 2.4× faster than a Transformer of similar size in inference. The difference is larger for longer sequences.

Tokens/s	Prompt length 512	Prompt length 1024	Prompt length 1536
Transformer-1.3B	1340	770	520
Hybrid H3-1.3B	1980	1580	1240

FlashConv Evaluation

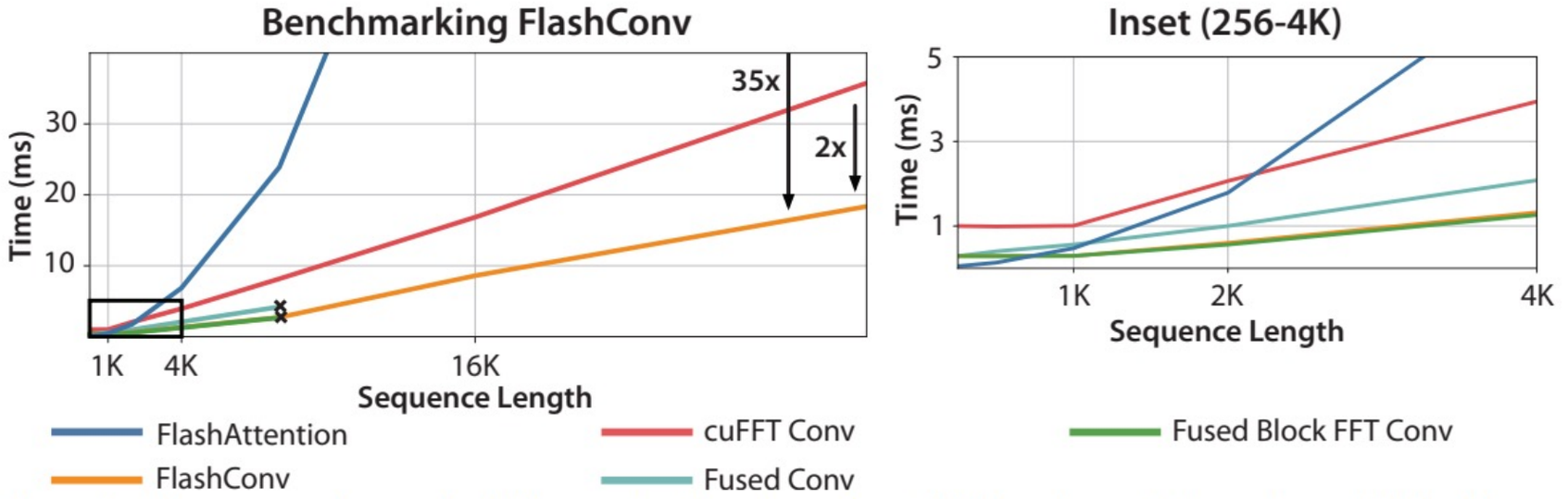


Figure 2: We compare the speed of different algorithms to perform FFT-based convolution, along with FlashAttention [15] (the fastest attention implementation we know of). We use batch size 8, hidden dimension 1024, and varying sequence length from 256 to 32k, and measure on an A100-SMX4-40GB GPU. We see that kernel fusion gives up to 3.4× speedup over naive FFTConv for short sequences (up to 512), block FFT gives up to 2× speedup for medium length sequences (1k - 8k), and state-passing allows 2.3× faster FFTConv for long sequences (16k and above).

Table 8: Speedup on the LRA benchmark.

Models	Speedup
Transformer	1×
FlashAttention [15]	2.4×
Block-sparse FlashAttention [15]	2.8×
S4 [28]	2.9×
S4 with FlashConv	5.8×

Summary

Goal:
understand and narrow the gap between attention and SSMs in language modeling in terms of modeling capabilities and hardware efficiency

Conclusion:
Use synthetic language tasks to evaluate the performance of model

propose H3 (Hungry Hungry Hippo), a new SSM-based layer designed to solve these language modeling tasks.

Scaling SSMs to improve the efficiency of SSMs on modern hardware, to reduce the hardware barrier between attention and SSMs.

Future work
combining the complementary strengths of SSMs and attention in the future

References:

[1]Burns, C., Izmailov, P., Kirchner, J. H., Baker, B., Gao, L., Aschenbrenner, L., Chen, Y., Ecoffet, A., Joglekar, M., Leike, J., Sutskever, I., & Wu, J. (2023). Weak-to-Strong Generalization: Eliciting Strong Capabilities With Weak Supervision. *ArXiv*. /abs/2312.09390

[2]Fu, D. Y., Dao, T., Saab, K. K., Thomas, A. W., Rudra, A., & Ré, C. (2022). Hungry Hungry Hippos: Towards Language Modeling with State Space Models. ArXiv. /abs/2212.14052


Q&A


THANK YOU