

Mamba: Linear-Time Sequence Modeling with Selective State Spaces

Author: Albert Gu & Tri Dao

Presenter: Nhat Nguyen, Chris Chen

Title breakdown

“Mamba: Linear-Time Sequence Modeling with Selective State Spaces”

Mamba -> newly proposed foundational model, also referred as S6

Linear-Time -> time complexity of $O(L)$, where L is sequence length

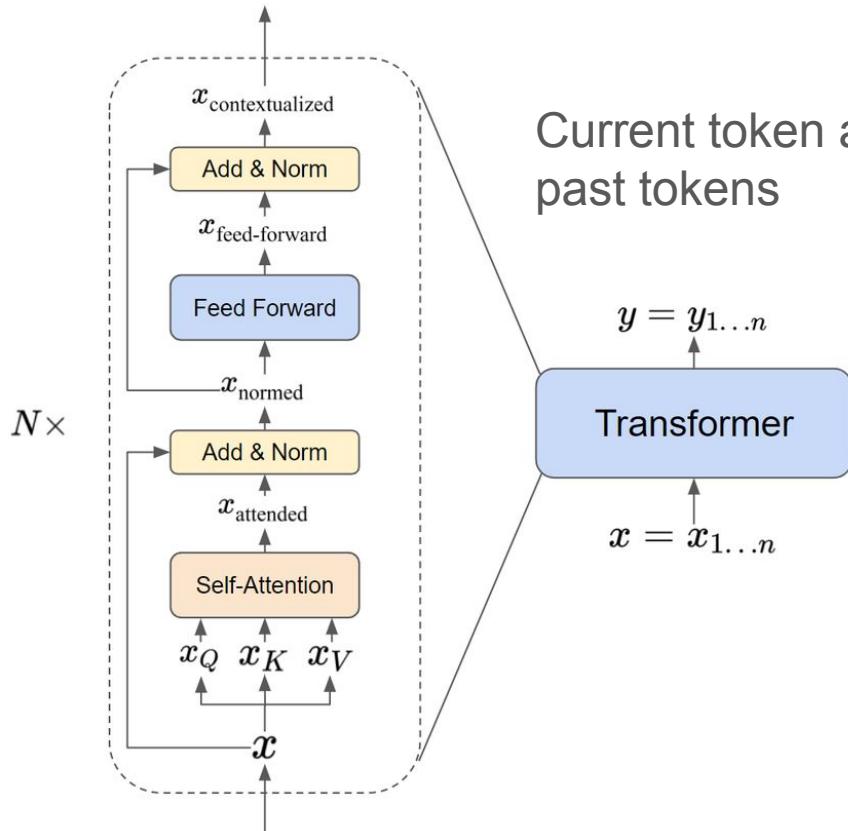
Sequence Modeling -> working with temporal or spatial data

State Spaces -> a mechanism used to store “memory” in the Markov chain process

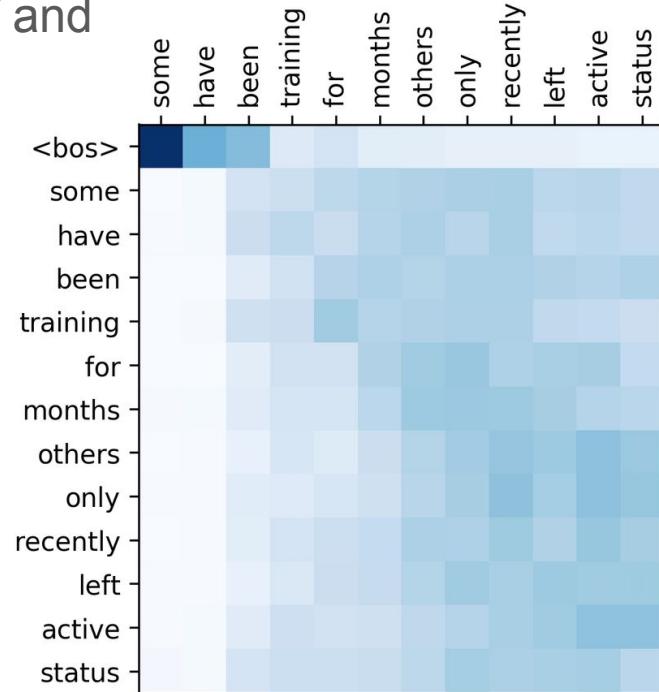
Selective -> A mechanism for filtering irrelevant input

Introduction

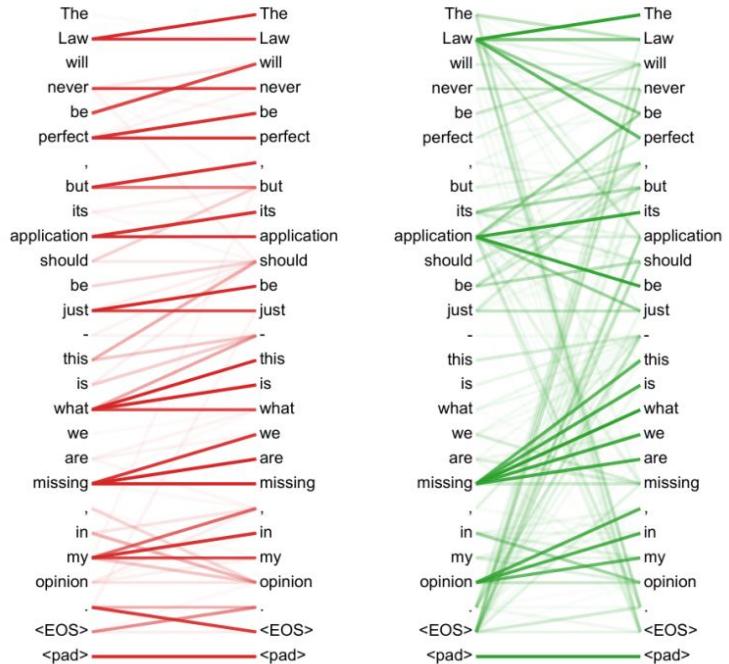
The Transformer



Current token attend to itself and past tokens



The Transformer

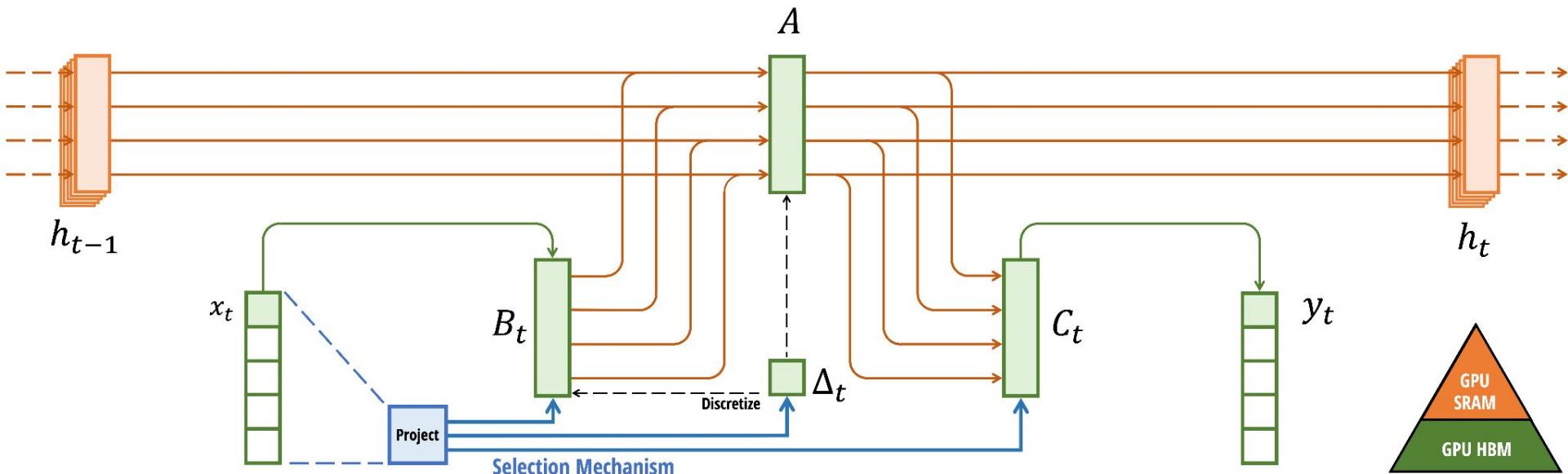


$O(L^2)$ complexity, where L is sequence length

Scaling is quadratic!

Mamba

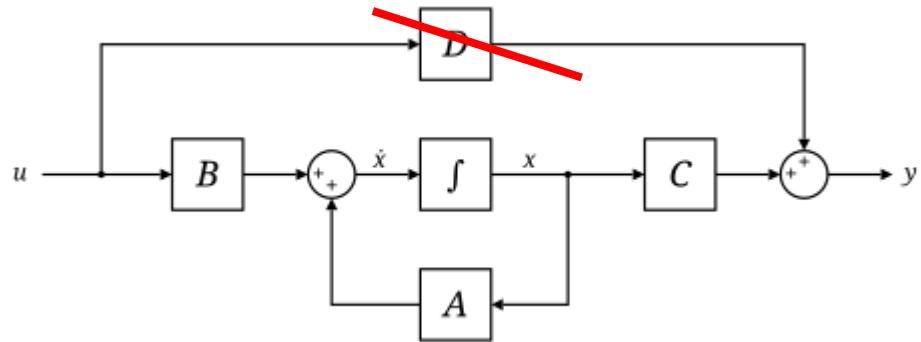
Selective State Space Model with Hardware-aware State Expansion



State Space Model (SSM)

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t)$$

~~$$\mathbf{y}(t) = \mathbf{C}(t)\mathbf{x}(t) + \mathbf{D}(t)\mathbf{u}(t)$$~~



$\mathbf{x}(\cdot)$ is called the "state vector", $\mathbf{x}(t) \in \mathbb{R}^n$;

$\mathbf{y}(\cdot)$ is called the "output vector", $\mathbf{y}(t) \in \mathbb{R}^q$;

$\mathbf{u}(\cdot)$ is called the "input (or control) vector", $\mathbf{u}(t) \in \mathbb{R}^p$;

$\mathbf{A}(\cdot)$ is the "state (or system) matrix", $\dim[\mathbf{A}(\cdot)] = n \times n$,

$\mathbf{B}(\cdot)$ is the "input matrix", $\dim[\mathbf{B}(\cdot)] = n \times p$,

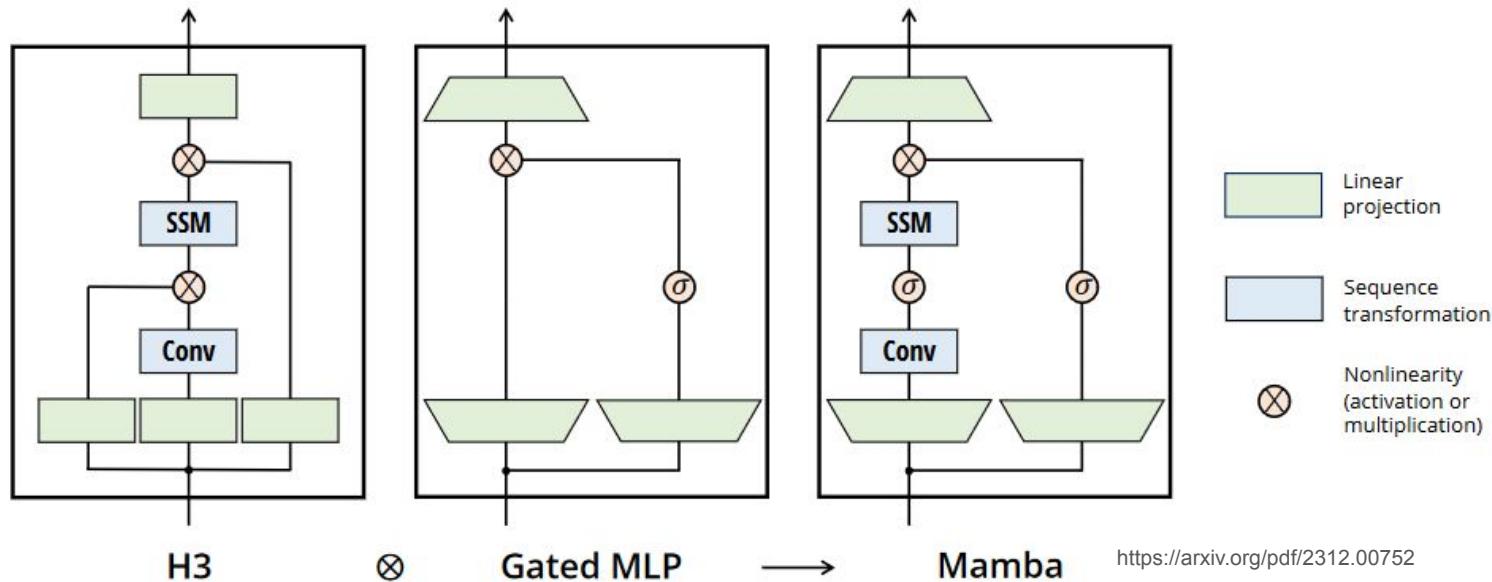
$\mathbf{C}(\cdot)$ is the "output matrix", $\dim[\mathbf{C}(\cdot)] = q \times n$,

$\mathbf{D}(\cdot)$ is the "feedthrough (or feedforward) matrix" (in cases where the system model does not have a direct feedthrough, $\mathbf{D}(\cdot)$ is the zero matrix), $\dim[\mathbf{D}(\cdot)] = q \times p$.

$$\dot{\mathbf{x}}(t) := \frac{d}{dt}\mathbf{x}(t).$$

Selective Scan Structured State Space Sequence Model (Mamba/S6)

- Inspired by S4 model (Structured State Space Sequence Model)
- $O(L)$ time complexity
- Scans along the context window L , updating state $h(t)$



Selective State Space Model (Mamba/S6)

Algorithm 2 SSM + Selection (S6)

Input: $x : (\mathbf{B}, \mathbf{L}, \mathbf{D})$

Output: $y : (\mathbf{B}, \mathbf{L}, \mathbf{D})$

1: $\mathbf{A} : (\mathbf{D}, \mathbf{N}) \leftarrow \text{Parameter}$

▷ Represents structured $N \times N$ matrix

2: $\mathbf{B} : (\mathbf{B}, \mathbf{L}, \mathbf{N}) \leftarrow s_B(x)$

3: $\mathbf{C} : (\mathbf{B}, \mathbf{L}, \mathbf{N}) \leftarrow s_C(x)$

4: $\Delta : (\mathbf{B}, \mathbf{L}, \mathbf{D}) \leftarrow \tau_\Delta(\text{Parameter} + s_\Delta(x))$

5: $\overline{\mathbf{A}}, \overline{\mathbf{B}} : (\mathbf{B}, \mathbf{L}, \mathbf{D}, \mathbf{N}) \leftarrow \text{discretize}(\Delta, \mathbf{A}, \mathbf{B})$

6: $y \leftarrow \text{SSM}(\overline{\mathbf{A}}, \overline{\mathbf{B}}, \mathbf{C})(x)$

▷ Time-varying: recurrence (*scan*) only

7: **return** y

Mamba

Key idea:

SSM Block takes 2 inputs

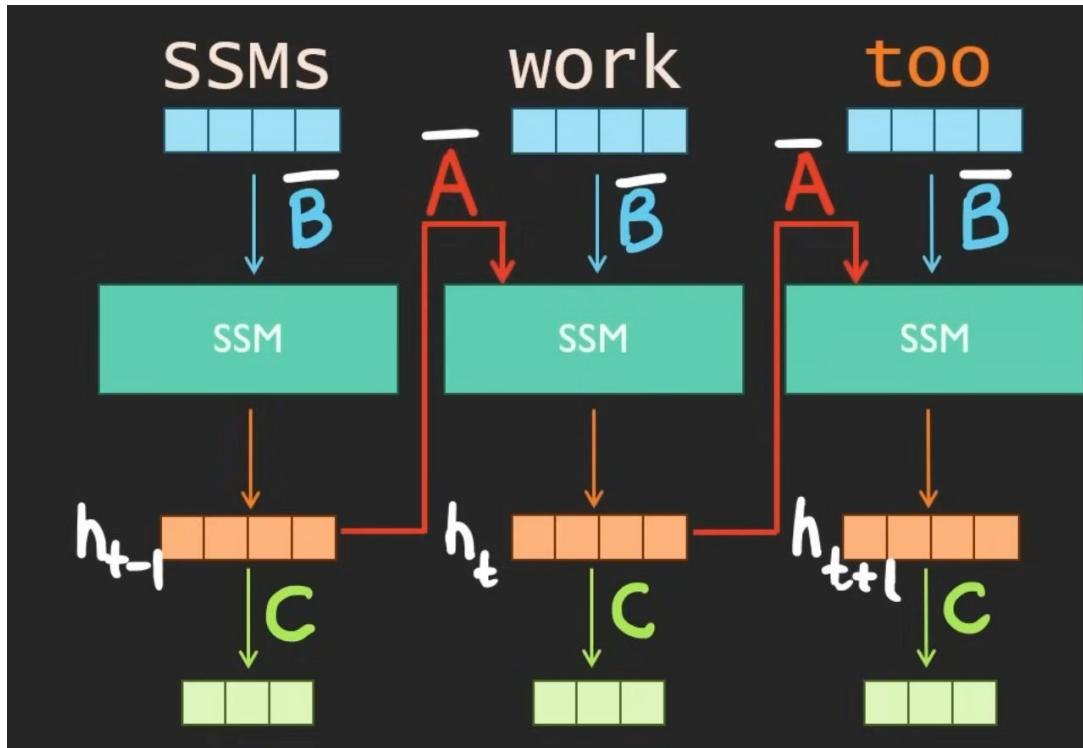
- Input $x \rightarrow x * B$
- Last state $h_{t-1} \rightarrow h_{t-1} * A$

And returns 2 outputs

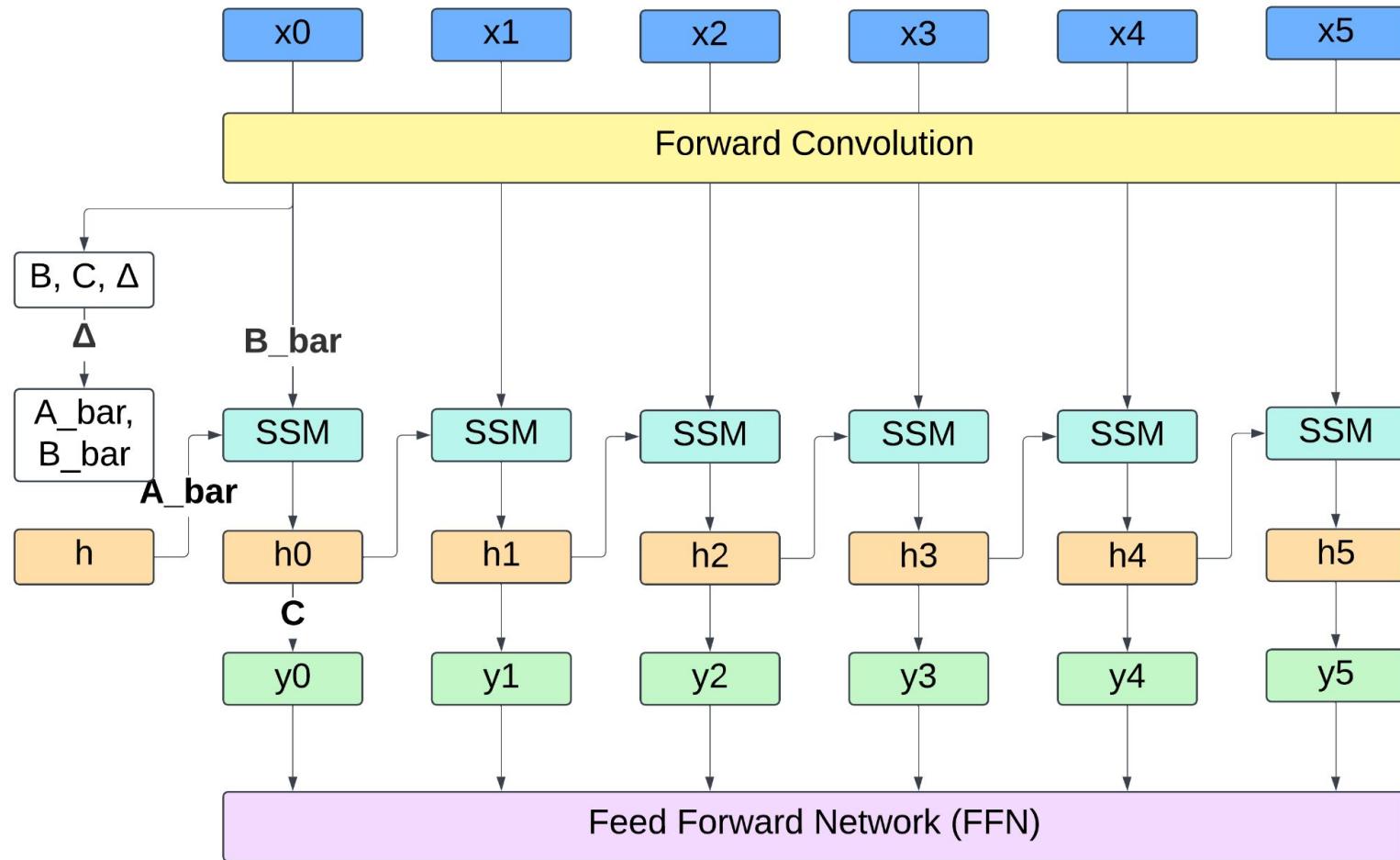
- New state h_t
- Output $y \rightarrow y = h_t * C$

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t)$$

$$\mathbf{y}(t) = \mathbf{C}(t)\mathbf{x}(t) + \mathbf{D}(t)\mathbf{u}(t)$$



Where is Texas A&M ?



Model Complexity

Transformers

Speed: $O(L^2)^*$

Memory: $O(L^2)$

Performance: 

State Space Models

Speed: $O(L)^*$

Memory: $O(L)$

Performance: 

Mamba (S6)

Speed: $O(L)^*$

Memory: $O(L)$

Performance: 



*full sequence, $O(n)$ for text generation using KV caching

*full sequence, $O(1)$ for text generation

*full sequence, $O(1)$ for text generation

Backup Link

https://youtu.be/lA_CMI7QPeo

Model Complexity

Transformers

Speed: $O(L^2)^*$

Memory: $O(L^2)$

Performance: 

State Space Models

Speed: $O(L)^*$

Memory: $O(L)$

Performance: 

Mamba (S6)

Speed: $O(L)^*$

Memory: $O(L)$

Performance: 

*full sequence, $O(n)$ for text generation using KV caching

*full sequence, $O(1)$ for text generation

*full sequence, $O(1)$ for text generation

Key Contribution

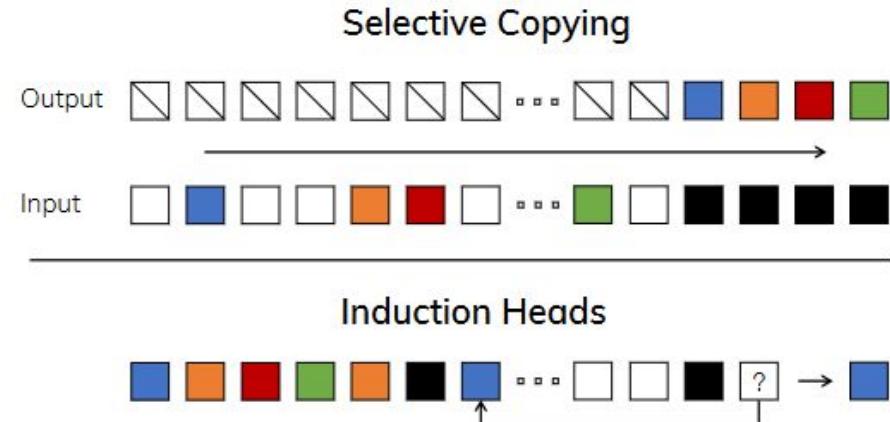
Input Dependency

Input dependency lets the model choose to remember or forget each input.

- Can ignore noise/irrelevant data
- Longer context theoretically improves performance (normally long context degrades performance)



Perfectly solved by LTI (e.g. convolutional) models that do not need to look at the actual inputs



Input Dependency

MODEL	ARCH.	LAYER	Acc.
S4	No gate	S4	18.3
-	No gate	S6	97.0
H3	H3	S4	57.0
Hyena	H3	Hyena	30.1
-	H3	S6	99.7
-	Mamba	S4	56.4
-	Mamba	Hyena	28.4
Mamba	Mamba	S6	99.8

Table 1: **(Selective Copying.)**

Accuracy for combinations of architectures and inner sequence layers.

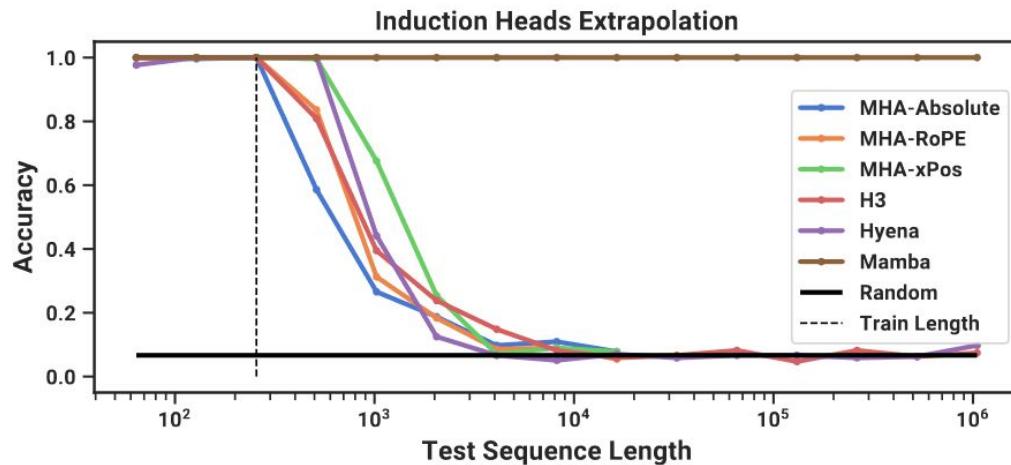
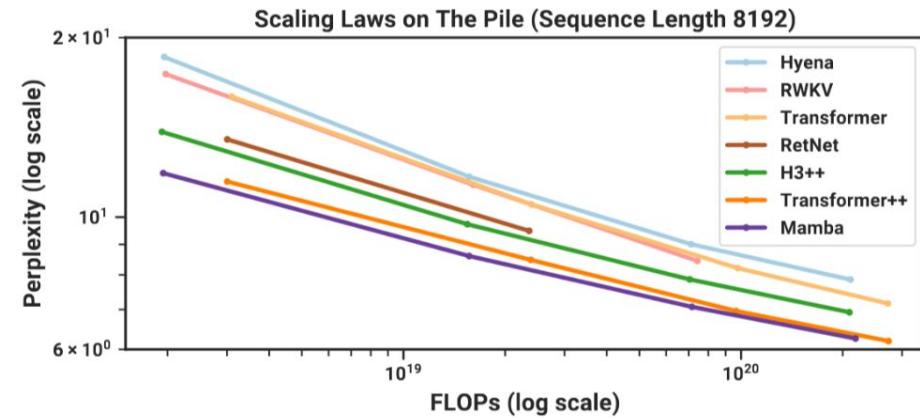
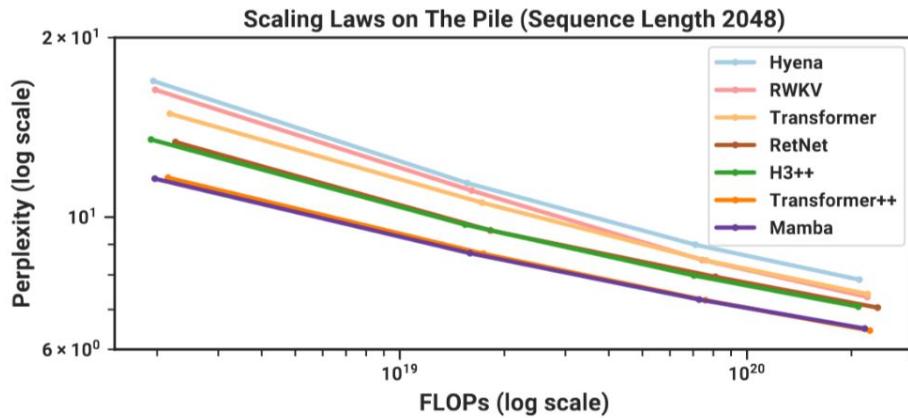


Table 2: **(Induction Heads.)** Models are trained on sequence length $2^8 = 256$, and tested on increasing sequence lengths of $2^6 = 64$ up to $2^{20} = 1048576$. Full numbers in Table 11.

Fast Training/Inference

Model is only affected by current input and current state

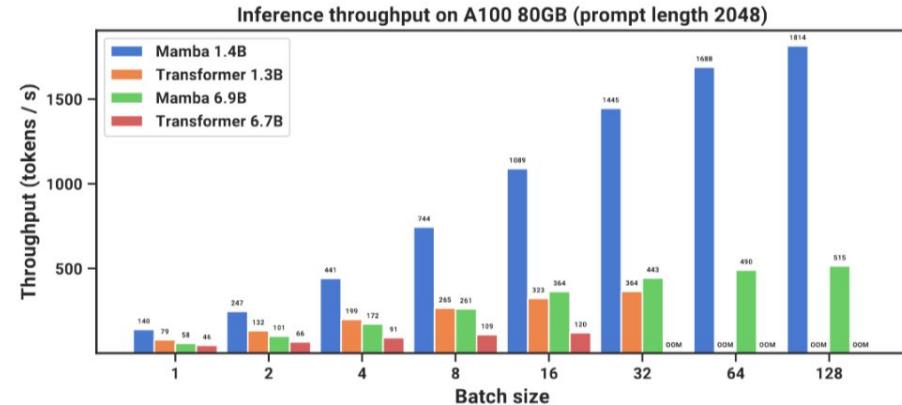
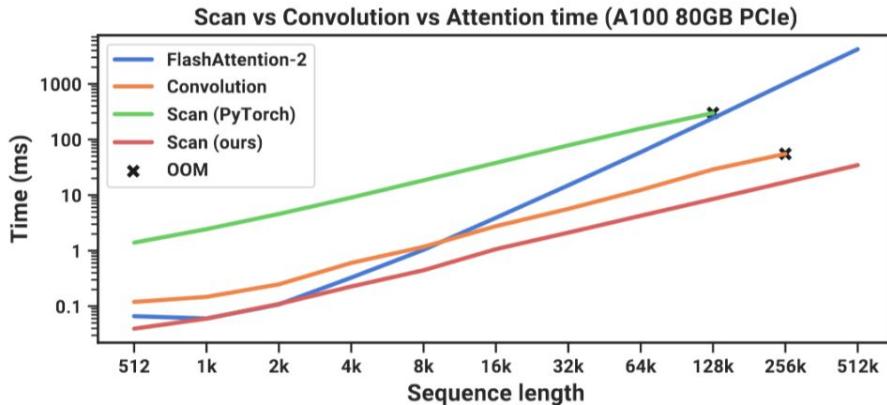
- Full sequence training and inference is $O(L)$ for time and memory
- Text generation training and inference is $O(1)$ (single token)



Speed

Model is performant compared to similar model sizes

- Faster speed (from proposed optimized hardware algorithm)
- More throughput



General Modelling

Mamba works on both discrete and continuous data

- Discrete (text, genomics/DNA modeling)
- Continuous (audio/signal)
- Does slightly worse for continuous data because of discretization bias
(discretization is affected by input even when it shouldn't)

Conclusion

Key takeaways (Mamba/S6)

Strengths:

- **Long context windows** (up to 1M tokens)
- **Fast compute** ($O(L)$ training, $O(1)$ for per-token prediction, $O(L)$ for sequence prediction)
- **Selective memory**, can filter noise/irrelevant data
- **Comes with GPU optimization** (kernel fusion, SRAM discretization)

Weaknesses:

- **Bias towards continuous data** because discretization is dependent to input
- **Essentially a compression algorithm**, it can only store and read from a finite state
- **Currently difficult to use** due to technical implementations
- **Results from study is limited** (no large scale experiments, using only perplexity for some results)

Ongoing works:

Jamba: A Hybrid Transformer-Mamba Language Model (March 2024)

- Mixture of Experts of Transformers + Mamba
- [https://arxiv.org/pdf/2403.19887](https://arxiv.org/pdf/2403.19887.pdf)

Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality (May 2024)

- AKA Mamba-2; same authors; provides theoretical connections between Mamba and Transformers
- [https://arxiv.org/pdf/2405.21060](https://arxiv.org/pdf/2405.21060.pdf)

Taipan: Efficient and Expressive State Space Language Models with Selective Attention (October 2024)

- Combines Mamba 2 with selective attention layers (SALs)
- [https://arxiv.org/pdf/2410.18572](https://arxiv.org/pdf/2410.18572.pdf)

Questions?



TEXAS A&M UNIVERSITY
Engineering

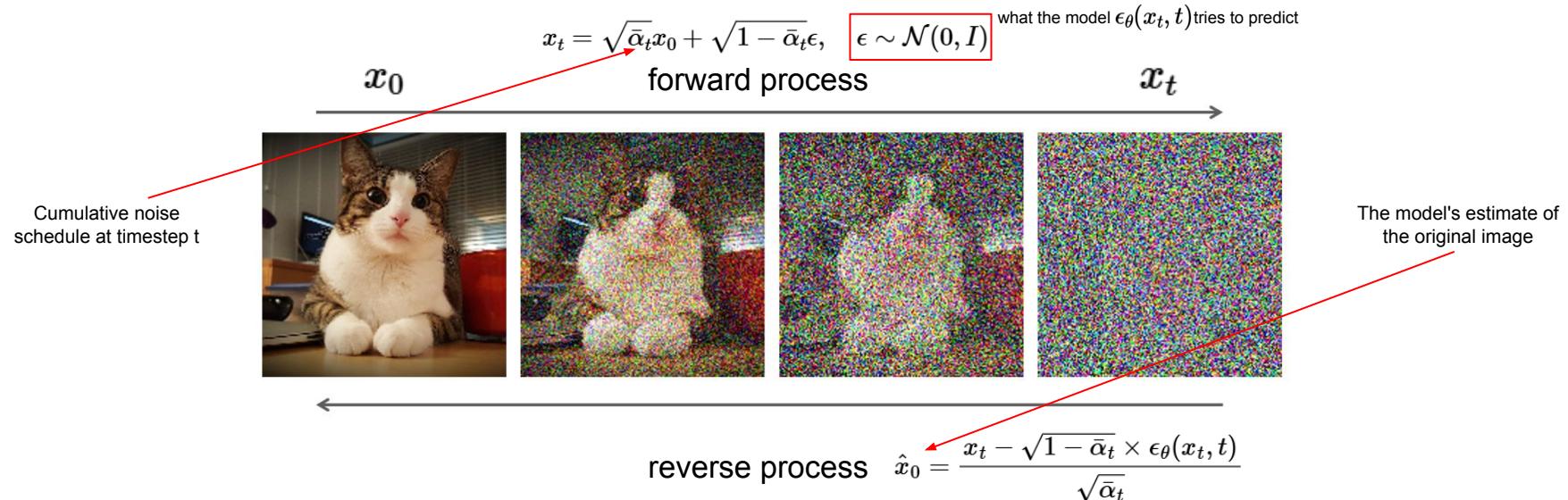
Scaling Rectified Flow Transformers for High-Resolution Image Synthesis

Mu-Ruei Tseng, Harshavardhana Srinivasan

Background

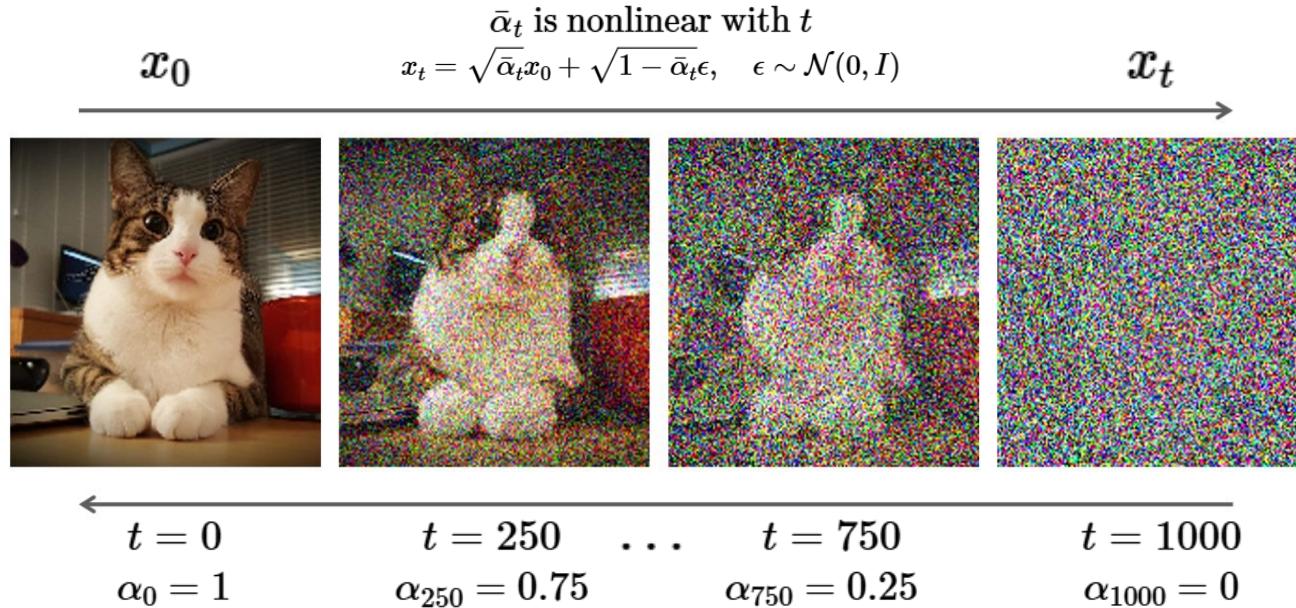
Diffusion Model

- A generative model that creates data from noise



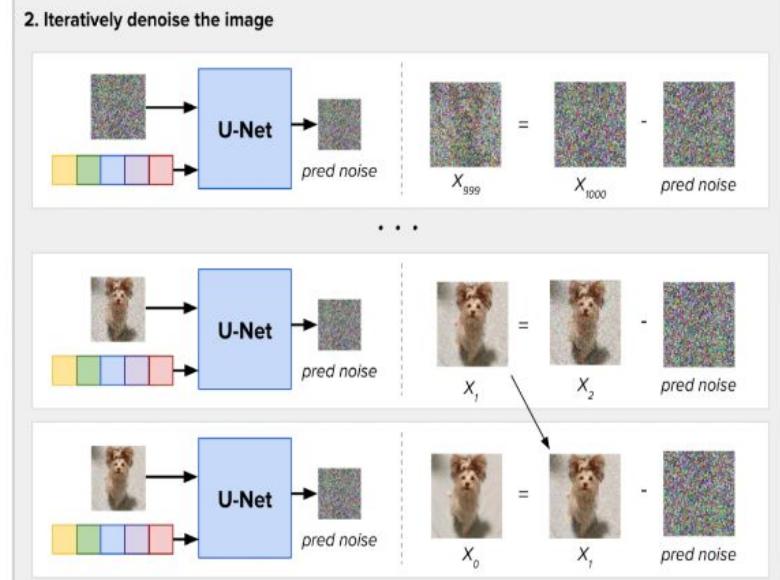
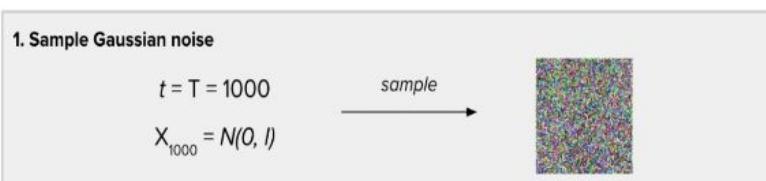
Background

Ex: a linear noise schedule $\alpha_t = 1 - \frac{t}{T}$, $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$



Background

Reverse Process (inferencing)

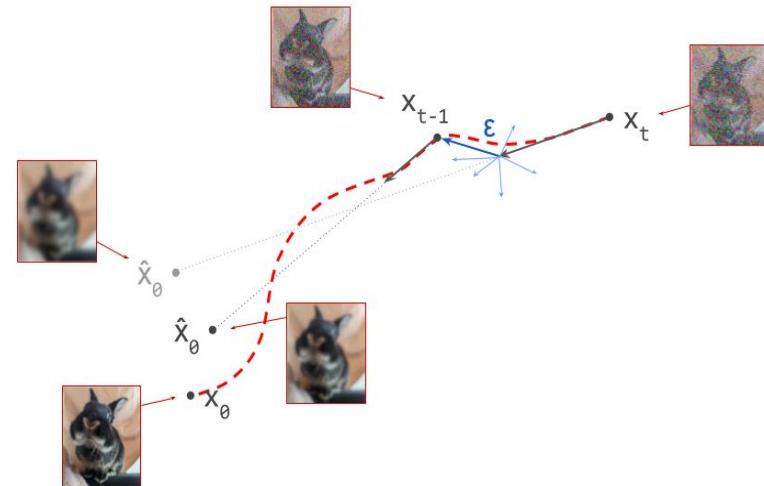


Background

The reverse process aims to estimate and follow a complex nonlinear curved path

At each stage

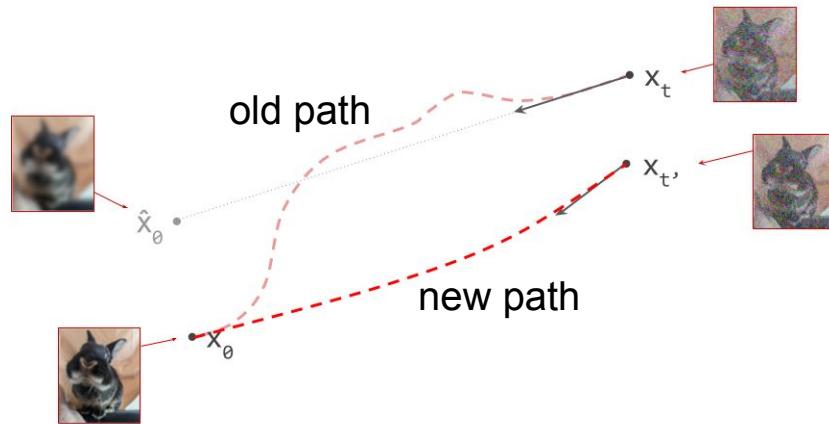
- Predict the tangent direction
- Take a small step along the direction
- Can lead to suboptimal result due to the **accumulated error**



Contribution

Introduce Rectified Flow for Noise Addition

- Reformulate the forward process as straight-line paths between the data and noise distributions



Design Tailored SNR Samplers

- Emphasize the importance of focusing on intermediate timesteps in the diffusion process

Novel Transformer-Based Architecture

- A bidirectional transformer architecture with separate streams for image and text tokens
- Outperform SOTA models like DALL-E 3 and SDXL

Comprehensive Experiments on Noise Methods

- Conduct large-scale experiments comparing different noise-adding methods and sampling techniques

Methodology - Flow Based Method

- Proposed Flow Based Generative model, which is used to map samples from a noise distribution p_1 to data distribution p_0 through an ODE equation

$$dy_t = v_\Theta(y_t, t) dt$$

- The vector field u , maps the data to noise through intermediate steps as defined in the forward process as follows:

$$z_t = a_t x_0 + b_t \epsilon \quad \text{where } \epsilon \sim \mathcal{N}(0, I)$$

- A typical solution to the above equation involves solving the marginal vector field $u_t(z|\epsilon)$ that averages across all noise samples, which becomes costly in computation.

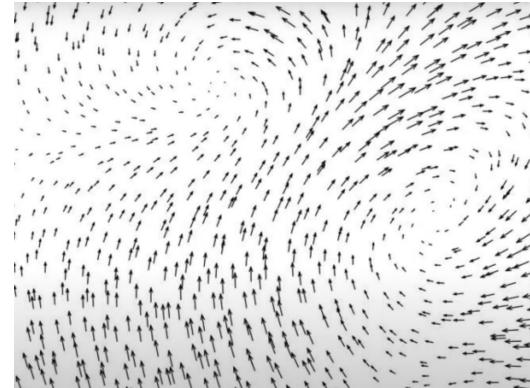
$$\psi_t(\cdot|\epsilon) : x_0 \mapsto a_t x_0 + b_t \epsilon$$

$$u_t(z|\epsilon) := \psi'_t(\psi_t^{-1}(z|\epsilon)|\epsilon)$$

Objective Function

- The objective function trains the model to predict the velocity field $v(z, t)$, which describes how samples move along the trajectory from data (p_0) to noise (p_1) over time t
- The model minimizes the difference between the predicted velocity $v(z, t)$ (output of the neural network) and the target velocity $u(z, t)$. For example: $\|v_\theta(z, t) - u_t(z)\|_2^2$. The original loss function computes this difference, averaged over all timesteps and noise samples.

velocity field



Methodology - Objective Function



Instead of learning $\|v_\theta(z, t) - u_t(z)\|_2^2$, reform the loss to minimize the noise added at each timestamp:

- Simpler and more direct to predict
- Aligns more naturally with the reverse process (as the forward process explicitly adds noise at each step)

The final objective incorporates weights derived from the signal-to-noise ratio (SNR) and timestep density

- Ensure that the model prioritizes important timesteps for improved performance.

$$\mathcal{L}_w(x_0) = -\frac{1}{2} \mathbb{E}_{t \sim \mathcal{U}(t), \epsilon \sim \mathcal{N}(0, I)} [w_t \lambda'_t \|\epsilon_\Theta(z_t, t) - \epsilon\|^2]$$

Weighted Loss Function:

where $w_t = -\frac{1}{2} \lambda'_t b_t^2$ corresponds to \mathcal{L}_{CFM} .

Rectified Flow: Defines a forward process as straight paths between the data distribution and a standard normal distribution as $z_t = (1 - t)x_0 + t\epsilon$

EDM: Uses the forward process of the form $z_t = x_0 + b_t\epsilon$ where $b_t = \exp F_N^{-1}(t|P_m, P_s^2)$

Cosine: Has the forward process of the form $z_t = \cos(\frac{\pi}{2}t)x_0 + \sin(\frac{\pi}{2}t)\epsilon$

LDM (Linear): Uses a modification of the DDPM schedule. Both are variance preserving schedules given by $b_t = \sqrt{1 - a_t^2}$

The diffusion coefficients of at and bt: $a_t = (\prod_{s=0}^t (1 - \beta_s))^{\frac{1}{2}}$ and $\left(\sqrt{\beta_0} + \frac{t}{T-1}(\sqrt{\beta_{T-1}} - \sqrt{\beta_0})\right)^2$

Intuitively, the error in the intermediate timestamp is harder to learn

- Involves a more complex balance between signal and noise
- The model should focus more in the intermediate timesteps

Tailored SNR Sampler (logit-normal sampling, mode sampling)

- Sample intermediate timesteps with higher frequency (opposed to uniform distribution)
- Outperform uniform sampling and diffusion baselines like EDM and LDM-Linear.

Methodology - Tailored SNR Samplers

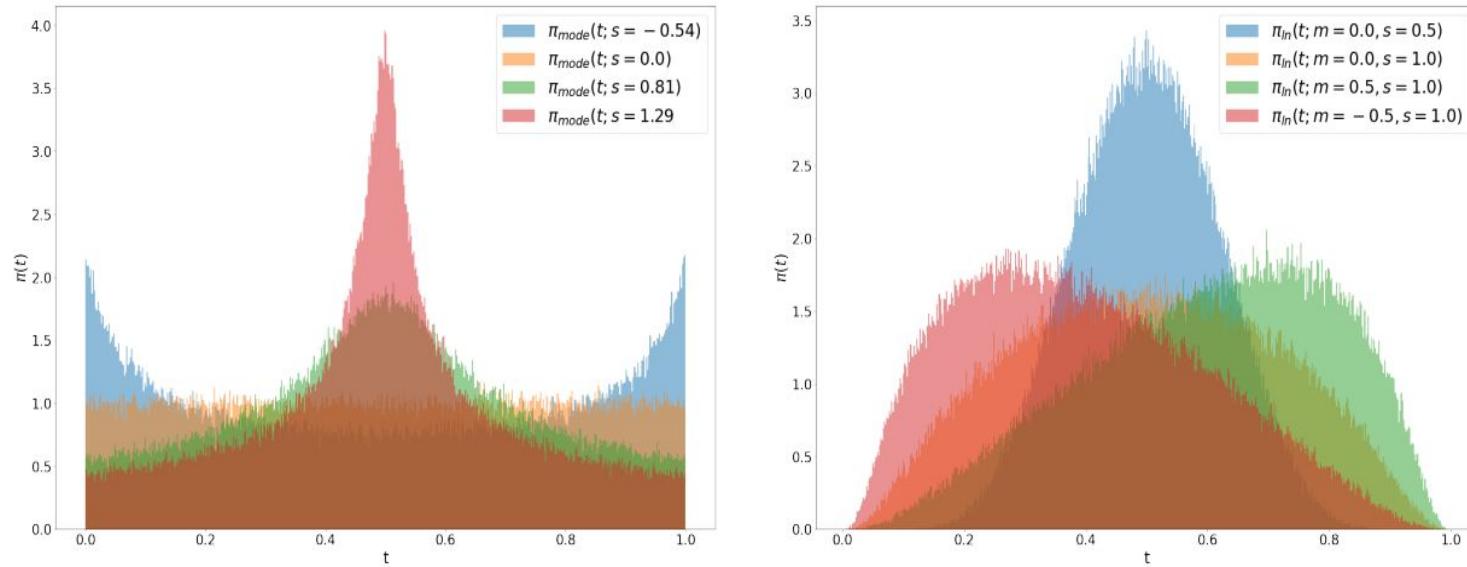
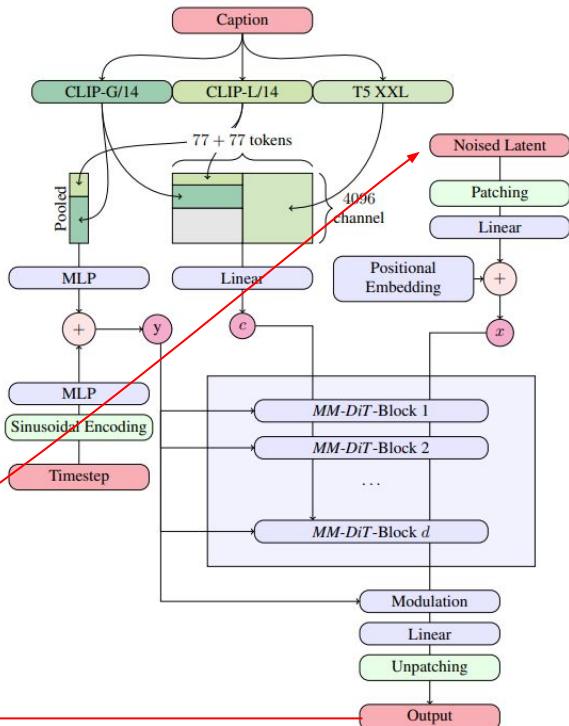
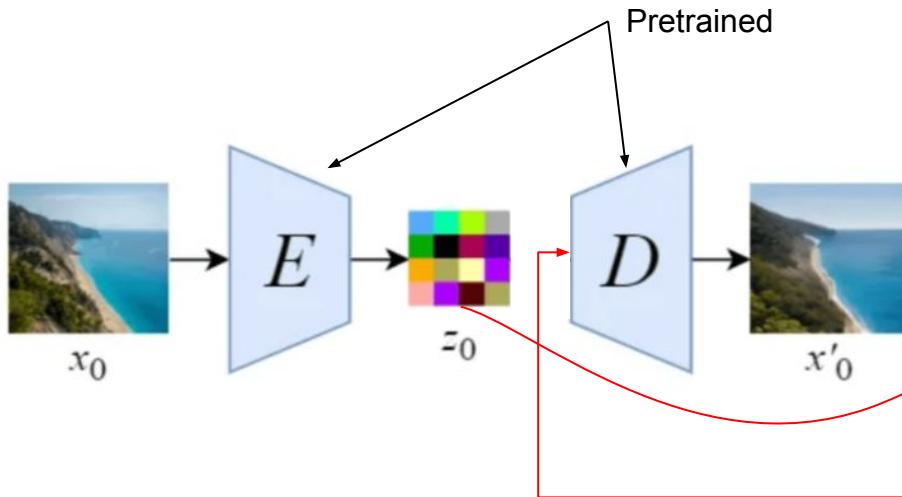


Figure 11. The mode (left) and logit-normal (right) distributions that we explore for biasing the sampling of training timesteps.

Methodology - Text-to-Image Model

Similar to the Latent Diffusion Model

- Training the diffusion model in the latent space

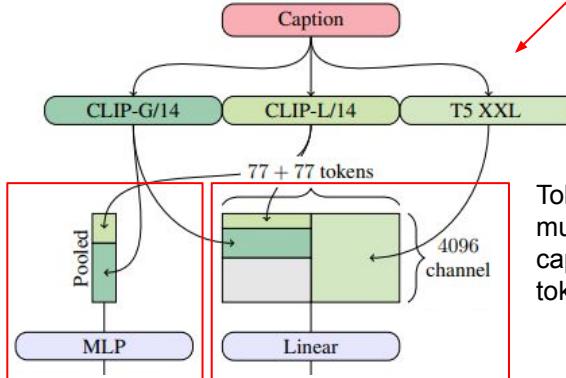


(a) Overview of all components.

Methodology - Text-to-Image Model

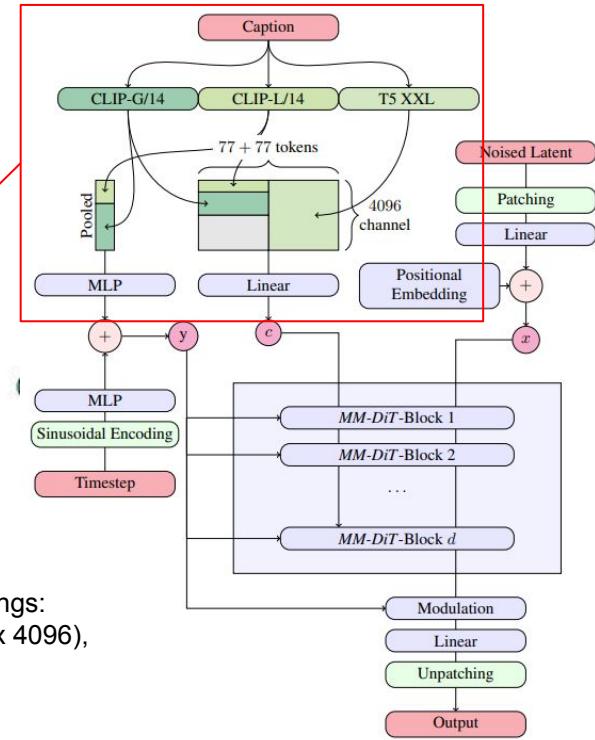
Text embedding

- Uses pretrained text encoders to obtain high-quality text embeddings
 - Uses multiple text encoders and create a unified text representation (like ensemble)
 - A 46.3% dropout rate is applied to the text embeddings during training (more flexible)



Pooling:
creates a single, compact embedding that summarizes the entire text
ex: mean pooling (avg the tokens)

Token-Level Embeddings:
multiple vectors (154×4096), captures fine-grained token-specific details

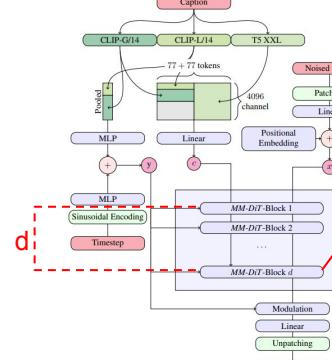


(a) Overview of all components.

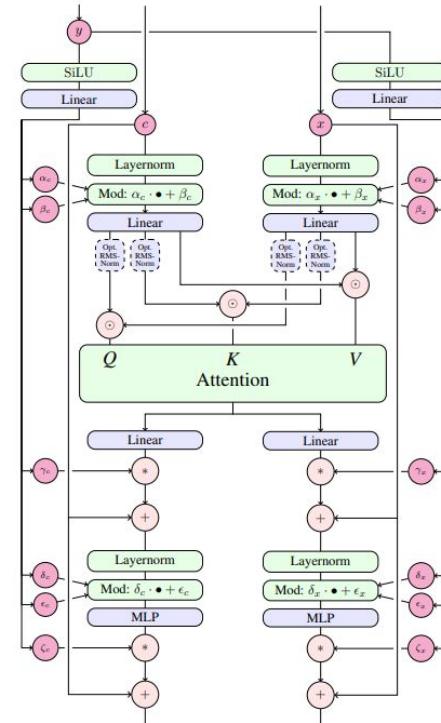
Methodology - Text-to-Image Model

MMDiT Block

- Unlike DiT, MMDiT uses separate weights for two modalities
 - Like having two transformers but share the attention operation
 - Intuition: Both representation can work in their own space while taking the other into account (during the attention part)
- Model Scaling:
 - Ensures a balanced architecture as the model grows deeper.
 - Depth: d
 - Hidden size: $64 \times d$
 - MLP size: $4 \times 64 \times d$
 - Attention heads: d



(a) Overview of all components.



(b) One MM-DiT block

CLIP score

- Measures how well a generated image aligns with a given text prompt
- Calculate the similarity between text and image embeddings

$$\text{CLIP score} = \cos(z_{\text{text}}, z_{\text{image}}) = \frac{z_{\text{text}} \cdot z_{\text{image}}}{\|z_{\text{text}}\| \times \|z_{\text{image}}\|}$$

FID (Fréchet Inception Distance)

- Evaluates the quality and diversity of generated images by comparing their distribution to a reference distribution
- In this paper, they use CLIP features instead of features generated from Inception-v3

Public Benchmarks (T2I-CompBench, GenEval)

Human Ratings

- Prompt following:
 - *Which image looks more representative to the provided text?*
- Visual Aesthetic:
 - *Given the prompt, which image is of higher-quality and aesthetically more pleasing?*
- Typography:
 - *Which image more accurately shows the text specified in the description?*

The paper shares extensive results covering two areas primarily

- Different samplers and trajectories
 - Employs global ranking and selective ranking from the best samplers
- Model architecture scaling
 - Comparison across different architecture and the performances

Results - Global Ranking of Variants

variant	rank averaged over		
	all	5 steps	50 steps
rf/lognorm(0.00, 1.00)	1.54	1.25	1.50
rf/lognorm(1.00, 0.60)	2.08	3.50	2.00
rf/lognorm(0.50, 0.60)	2.71	8.50	1.00
rf/mode(1.29)	2.75	3.25	3.00
rf/lognorm(0.50, 1.00)	2.83	1.50	2.50
eps/linear	2.88	4.25	2.75
rf/mode(1.75)	3.33	2.75	2.75
rf/cosmap	4.13	3.75	4.00
edm(0.00, 0.60)	5.63	13.25	3.25
rf	5.67	6.50	5.75
v/linear	6.83	5.75	7.75
edm(0.60, 1.20)	9.00	13.00	9.00
v/cos	9.17	12.25	8.75
edm/cos	11.04	14.25	11.25
edm/rf	13.04	15.25	13.25
edm(-1.20, 1.20)	15.58	20.25	15.00

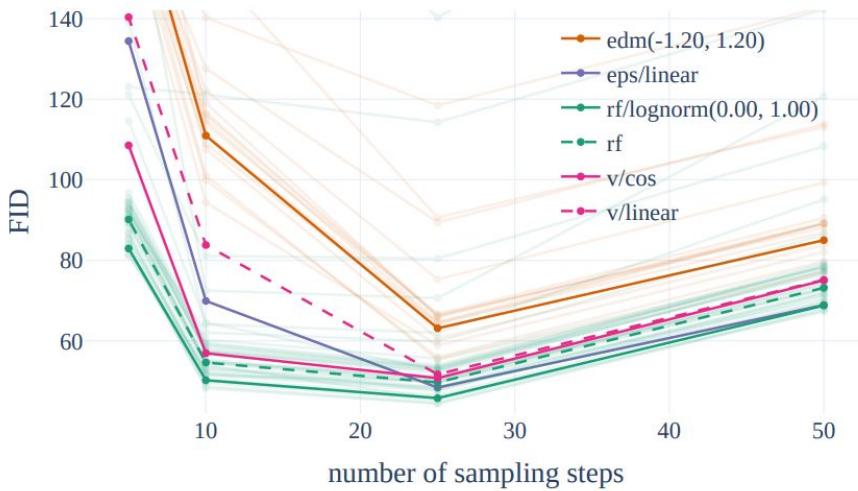
- Ranks 61 variants based on their overall performance across datasets (ImageNet, CC12M), sampling steps (5 and 50), and sampler settings.
- Uses Pareto- based global ranking derived from averaged ranks over CLIP and FID scores across various configurations.
- It identifies the top-performing variants globally, considering a wide range of scenarios (e.g., **rf/lognorm(0, 1)**, **rf/lognorm(1, 0.6)**, and others).

Results - Effective Samplers

variant	ImageNet		CC12M	
	CLIP	FID	CLIP	FID
rf	0.247	49.70	0.217	94.90
edm(-1.20, 1.20)	0.236	63.12	0.200	116.60
eps/linear	0.245	48.42	0.222	90.34
v/cos	0.244	50.74	0.209	97.87
v/linear	0.246	51.68	0.217	100.76
rf/lognorm(0.50, 0.60)	0.256	80.41	<u>0.233</u>	120.84
rf/mode(1.75)	0.253	44.39	0.218	94.06
rf/lognorm(1.00, 0.60)	<u>0.254</u>	114.26	0.234	147.69
rf/lognorm(-0.50, 1.00)	0.248	<u>45.64</u>	0.219	89.70
rf/lognorm(0.00, 1.00)	0.250	45.78	0.224	<u>89.91</u>

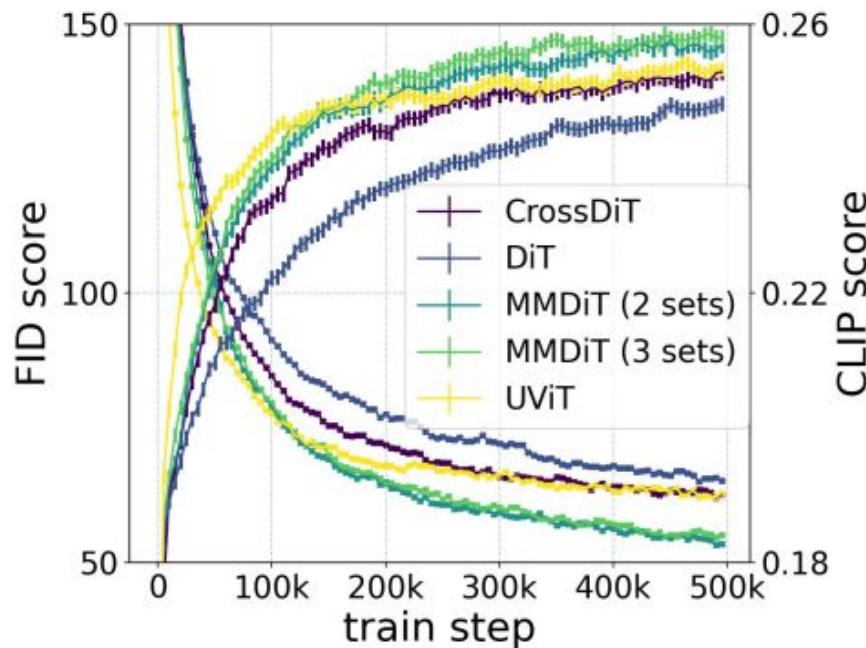
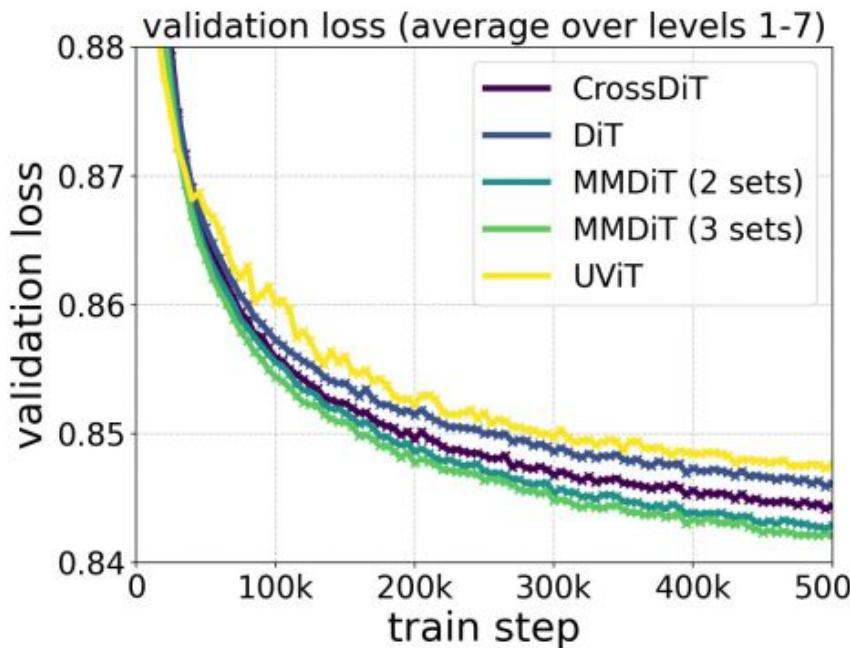
- Shows how individual variants, like **rf/lognorm(0.50, 0.60)** or **rf/lognorm(0, 1)**, perform on ImageNet and CC12M in terms of direct evaluation metrics.

Results - Samplers vs Training Steps

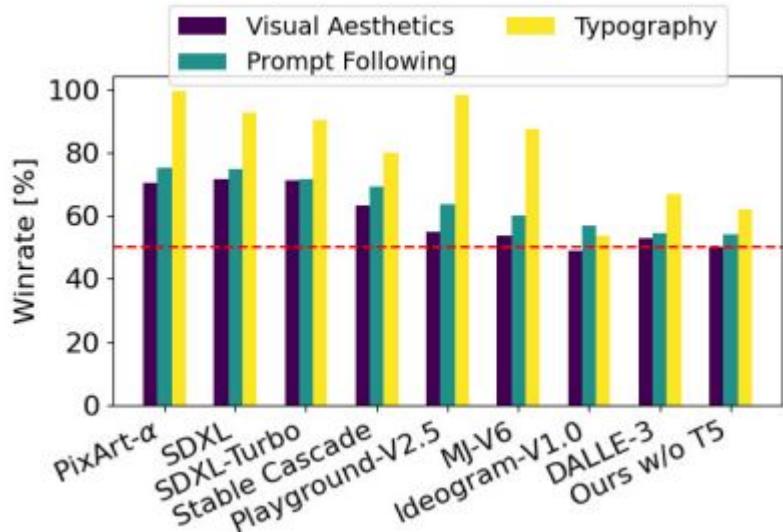


- **Rectified Flows (RF)** are highly sample-efficient, performing better than other formulations when the number of sampling steps is fewer than 25.
- For 25 or more steps, **rf/lognorm(0.00, 1.00)** continues to perform competitively, matching or surpassing **eps/linear**, showcasing its robustness across varying step counts.

Results - Samplers vs Training Steps



Results - Human Preference vs Closed And Open Model Evaluation



The 8B model performs favorably against current SOTA text-to-image models in human evaluations on the Parti-prompts, excelling in **visual quality**, **prompt adherence**, and **typography generation**.

Results - Scalability and Performance of MM-DiT Across Image and Video Generation

Model	Overall	Objects			Color		
		Single	Two	Counting	Colors	Position	Attribution
minDALL-E	0.23	0.73	0.11	0.12	0.37	0.02	0.01
SD v1.5	0.43	0.97	0.38	0.35	0.76	0.04	0.06
PixArt-alpha	0.48	0.98	0.50	0.44	0.80	0.08	0.07
SD v2.1	0.50	0.98	0.51	0.44	<u>0.85</u>	0.07	0.17
DALL-E 2	0.52	0.94	0.66	0.49	0.77	0.10	0.19
SDXL	0.55	0.98	0.74	0.39	<u>0.85</u>	0.15	0.23
SDXL Turbo	0.55	1.00	0.72	0.49	0.80	0.10	0.18
IF-XL	0.61	0.97	0.74	<u>0.66</u>	0.81	0.13	0.35
DALL-E 3	0.67	0.96	<u>0.87</u>	0.47	<u>0.83</u>	0.43	0.45
Ours (depth=18), 512 ²	0.58	0.97	0.72	0.52	0.78	0.16	0.34
Ours (depth=24), 512 ²	0.62	0.98	0.74	0.63	0.67	<u>0.34</u>	0.36
Ours (depth=30), 512 ²	0.64	0.96	0.80	0.65	0.73	0.33	0.37
Ours (depth=38), 512 ²	0.68	0.98	0.84	<u>0.66</u>	0.74	<u>0.40</u>	0.43
Ours (depth=38), 512 ² w/DPO	<u>0.71</u>	0.98	<u>0.89</u>	0.73	0.83	0.34	<u>0.47</u>
Ours (depth=38), 1024 ² w/DPO	0.74	<u>0.99</u>	0.94	<u>0.72</u>	0.89	0.33	0.60

- MM-DiT models improve validation loss with larger sizes and more training steps, aligning with evaluation metrics and human preferences.
- Larger models outperform SOTA models in prompt comprehension and overall quality

Key Contributions

- Rectified Flow
 - Resolves curved forward paths, improving reverse process efficiency and accuracy
- Tailored SNR Sampler (during training)
 - Focuses on intermediate timesteps as their errors are more difficult to model
- MM-DiT architecture
 - Multimodal transformer with separate weights for text and image modalities improves integration

Strengths

- Outperforms SOTA models (e.g., SDXL, DALL-E 3) on CLIP, FID, and human preferences
- Comprehensive evaluation

Limitation

- Performance Degradation with limited steps
 - Effectiveness of the model decreases significantly when the number of training steps is reduced
- Multi-Modal beyond text-image
 - The study focuses on high-resolution text-to-image tasks, with limited exploration of audio-text and video-text applications

Future Directions

- Exploring other multi-modal capabilities beyond text-to-image, such as text-to-video or audio-to-text
- Investigating efficient scaling methods to significantly reduce computational overhead while maintaining performance

- [The paradox of diffusion distillation](#)
- [An Introduction to Diffusion Models and Stable Diffusion](#)
- [Stable Diffusion 3](#)

Detecting Hallucinations in Large Language Models Using Semantic Entropy

Authored by: Sebastian Farquhar, Jannik Kossen,
Lorenz Kuhn & Yarin Gal

CSCE: 689 Programming Large Language Models (LLMs)

Presented by:
Howard Morgenthaler & Yasir Ali Farrukh

What are Hallucinations?

Incorrect outputs generated by a LLM

Many reasons

- Poorly trained
- Malicious intent
- Arbitrary reasons such as some randomness
- Model simply does not know the answer



Confabulations - The sub category of hallucinations focused on by the paper

Entropy

$$E(\mathbf{x}) = H(Y|\mathbf{x}) = - \sum_y P(y|\mathbf{x}) \ln P(y|\mathbf{x})$$



$$\log P(\mathbf{s}|\mathbf{x}) = \sum_i \log P(s_i|\mathbf{s}_{<i}, \mathbf{x})$$



$$\frac{1}{N} \sum_i^N \log P(s_i|\mathbf{s}_{<i}, \mathbf{x})$$

The mathematical equation for finding uncertainty.

Idea!

- Use entropy to determine the uncertainty of an LLM produced answer.
- Simply need to calculate the probability of a sequence being generated.

Length Normalization - Taking the log probability mean rather than the sum

Meaning Instead of Tokens

Semantic Entropy - Take the entropy/probability of sequences that have the same meaning rather than sequence by sequence

- Cluster sequences based on bidirectional entailment
- Use LLMs and NLI tools to determine entailment

Generation	p	Naive entropy				Semantic entropy			
		$p(s_i)$	$\log[p(s_i)]$	$p(s_i) \log[p(s_i)]$	$\sum_{s_i \in C_j} p(s_i)$	$p(C_j)$	$\log[p(C_j)]$	$p(C_j) \log[p(C_j)]$	
"Paris."	0.20	0.33	-0.48	-0.16	0.55	0.90	-0.04	-0.04	
"Paris."	0.20	0.33	-0.48	-0.16	-	-	-	-	
"It's Paris."	0.15	0.25	-0.61	-0.15	-	-	-	-	
"Rome."	0.05	0.08	-1.09	-0.09	0.05	0.08	-1.09	-0.09	
"New York."	0.01	0.02	-1.79	-0.03	0.01	0.02	-1.79	-0.03	
Sum	0.61	1.00	-4.45	-0.59	0.61	1.00	-2.92	-0.16	

Testing Semantic Entropy

Shorter generation: QA and Math

Baselines:

- Discrete Semantic Entropy
- Naive Entropy
- $P(\text{True})$
- Embedded Regression (In Domain)
- Embedded Regression (Out Of Domain)

Longer generation: Biographies

Baselines:

- Self-Check
- $P(\text{True})$

Evaluation/Results

Evaluation Metrics:

- Area Under The Receiver Operating Characteristic Curve (AUROC) :
 - Represents how good the method is at predicting whether an answer is likely to be correct or hallucinated (incorrect).
- Rejection Accuracy :
 - Represents how well a method can identify and prioritize accurate responses by rejecting less confident (and potentially incorrect) ones.
- Area Under The Rejection Accuracy Curve (AURAC):
 - Represents how much better your accuracy becomes if you reject (refuse to answer) the questions you are most uncertain about.

Evaluation Scenario:

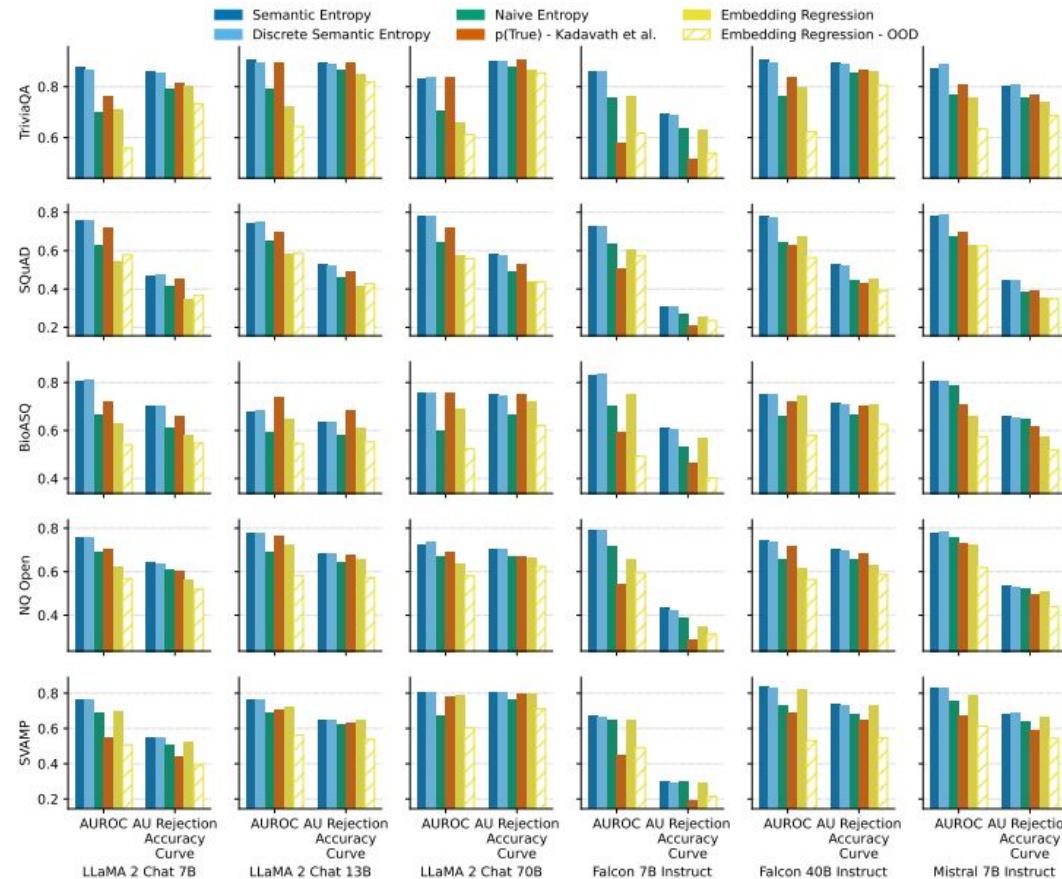
○ Question/Answers (Sentence Length Generation)

- Trivia Knowledgee (TriviaQA32)
- General Knowledge (SQuAD)
- Life Sciences (BioASQ34)
- And Open-domain Natural Questions (Nq-open35)
- Mathematical Word Problems (SVAMP37)

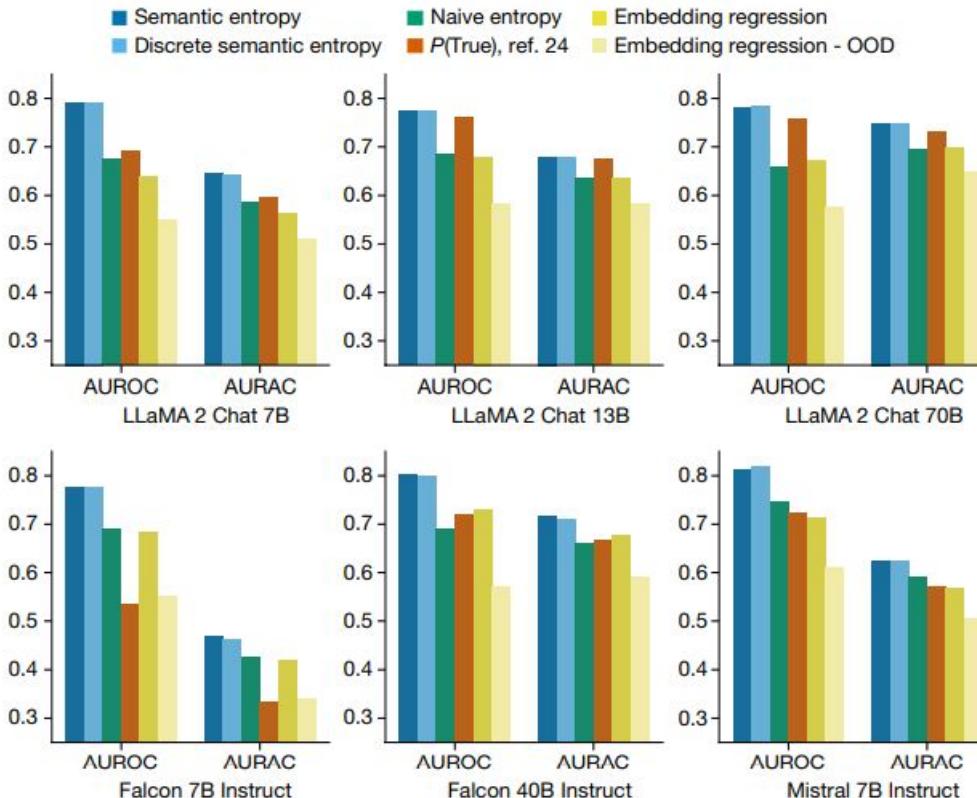
○ Paragraph-Length Passage

- Developed a Dataset of Biographies
 - GPT-4
 - 21 Individuals
 - 150 factual claims in total (Labelled manually as True/False)

Detecting Confabulations In Sentence-length Generations

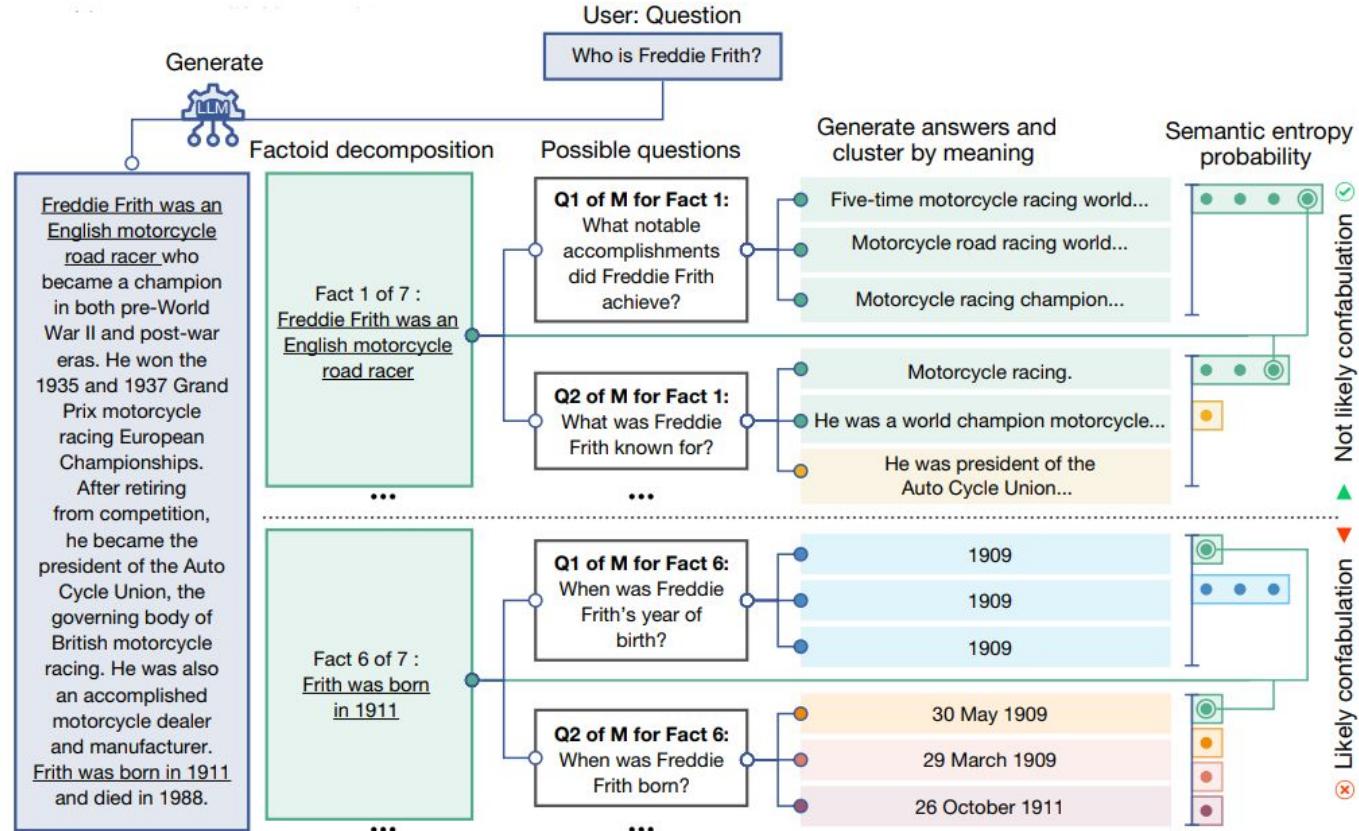


Detecting Confabulations In Sentence-length Generations

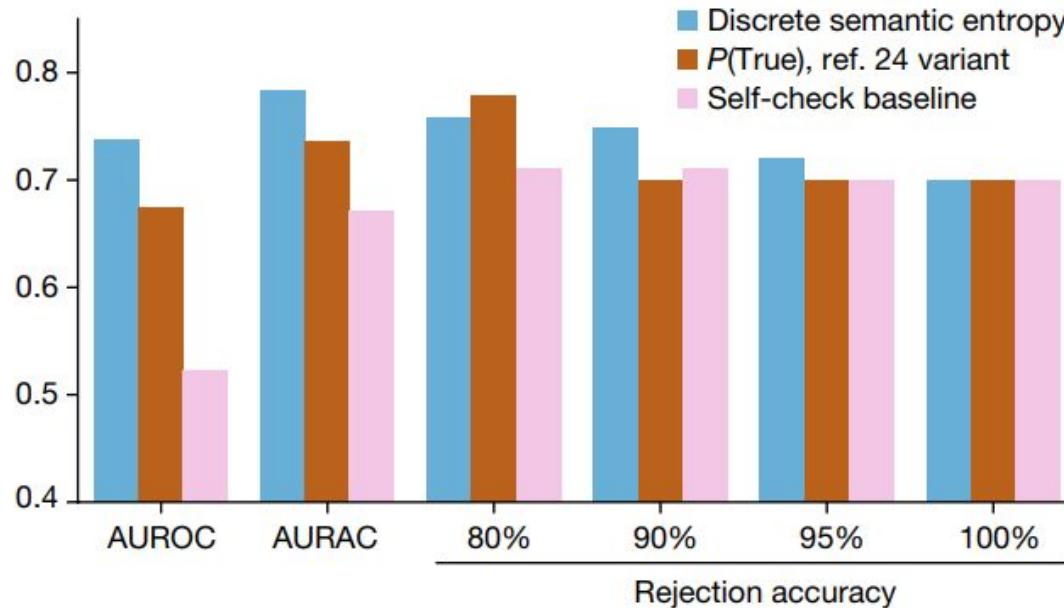


Detecting Confabulations Paragraph-length Biographies

1. Automatically decompose the paragraph into specific factual claims using an LLM (not necessarily the same as the original).
2. For each factual claim, use an LLM to automatically construct Q questions which might have produced that claim.
3. For each question, prompt the original LLM to generate M answers
4. For each question, compute the semantic entropy of the answers, including the original factual claim.
5. Average the semantic entropies over the questions to arrive at a score for the original factual claim.



Detecting Confabulations Paragraph-length Biographies



Conclusion & Limitation

Question	Model Answer	Reference Answer	Generations for Entropy	Prediction	
				Semantic Entropy	Naive Entropy
"Refineries, process chemical, power generation, mills and manufacturing plants are under what sector of construction?"	"Refineries, process chemical, power generation, mills and manufacturing plants are under the industrial sector of construction"	"Industrial or heavy civil engineering"	<p>"All the above are under the Industrial sector of construction."</p> <p>"The refineries, process chemical, power generation, mills and manufacturing plants are under the industrial sector of construction."</p> <p>"These are all under the heavy industrial sector of construction."</p>	Not Confabulation	Confabulation
"What is the function of the stard10 protein?"	"The stard10 protein functions as a negative regulator of the mTOR pathway by inhibiting the activity of the mTOR1 and mTORC2 complexes"	"STARD10 is a lipid transfer protein."	<p>"The stard10 protein plays a crucial role in regulating the dynamics of the actin cytoskeleton and cell adhesion during cell migration and cell division."</p> <p>"The stard10 protein functions as a negative regulator of the meiotic recombination process, specifically by inhibiting the activity of the D mc1 recombinase enzyme."</p> <p>"The stard10 protein functions as a regulator of lipid metabolism, promoting the breakdown of lipids in the liver and inhibiting their synthesis in the liver and adipose tissue."</p>	Confabulation	Confabulation
"Of which European country is 'fado' the national music?"	"Portugal"	"portugal"	<p>"Fado' is the national music of Portugal."</p> <p>"Fado' is the national music of Portugal."</p> <p>"Fado is the national music of Portugal."</p>	Not Confabulation	Not Confabulation
"When was BSkyB's digital service launched?"	"1 October 1998"	"BSkyB's digital service was launched in 1998."	<p>"BSkyB's digital service was launched in 1998."</p> <p>"BSkyB's digital service was launched on 1 October 1998."</p> <p>"BSkyB's digital service was launched on 1 October 1998."</p>	Confabulation	Not Confabulation

Semantic entropy clustered the answers into those which provided a specific date and those which gave only a year, so the result was considered uncertain.

Extra Cost of Computation

- o Semantic entropy requires something in the ballpark of 10x as much compute as a raw question

What does all this mean?

- o Semantic Entropy A Useful Tool
 - o Can Rule Out A 'Careless Error' From Your Model.
- o You shouldn't over-rely on it though. Semantic entropy won't catch:
 - o Cases where the model has been trained into an incorrect style of reasoning or set of facts.
 - o Cases where the model has been trained in a way that doesn't apply to a new setting or context.
 - o Cases where the model is engaging in 'deliberate' deception.

THANK YOU



Phi-3 Technical Report
A Highly Capable Language Model Locally on
Your Phone

Microsoft

Phi-3-mini

- 3.8 billion parameters, trained on 3.3 trillion tokens
- Achieves similar performance to Mixtral 8x7B and GPT-3.5.

User: Explain why it is surprising that one can build a language model small enough to fit on a phone, yet almost as powerful as ChatGPT. Just use one funny sentence.

phi-3-mini: It's like fitting a supercomputer in a flip phone, but instead of breaking the phone, it just breaks the internet with its tiny, yet mighty, linguistic prowess!

User: Okay now more serious answer, and note that this was achieved solely by changing the training data.

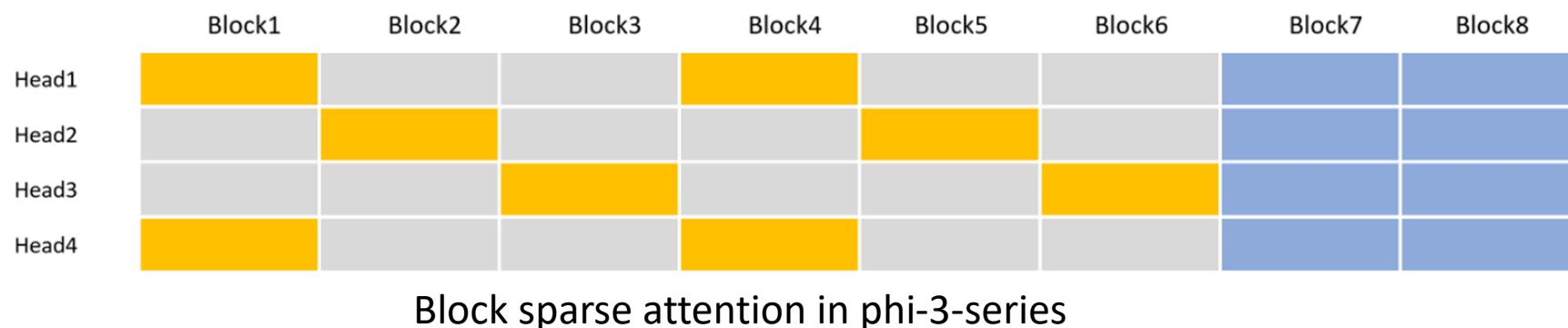
phi-3-mini: The development of a compact language model that rivals the capabilities of ChatGPT, while fitting on a phone, is a testament to the power of data-driven machine learning. By meticulously curating and optimizing the training dataset, researchers can significantly reduce the model's size without compromising its performance. [...]

Technical Specifications

- Phi-3-mini built upon similar structure to Llama-2.

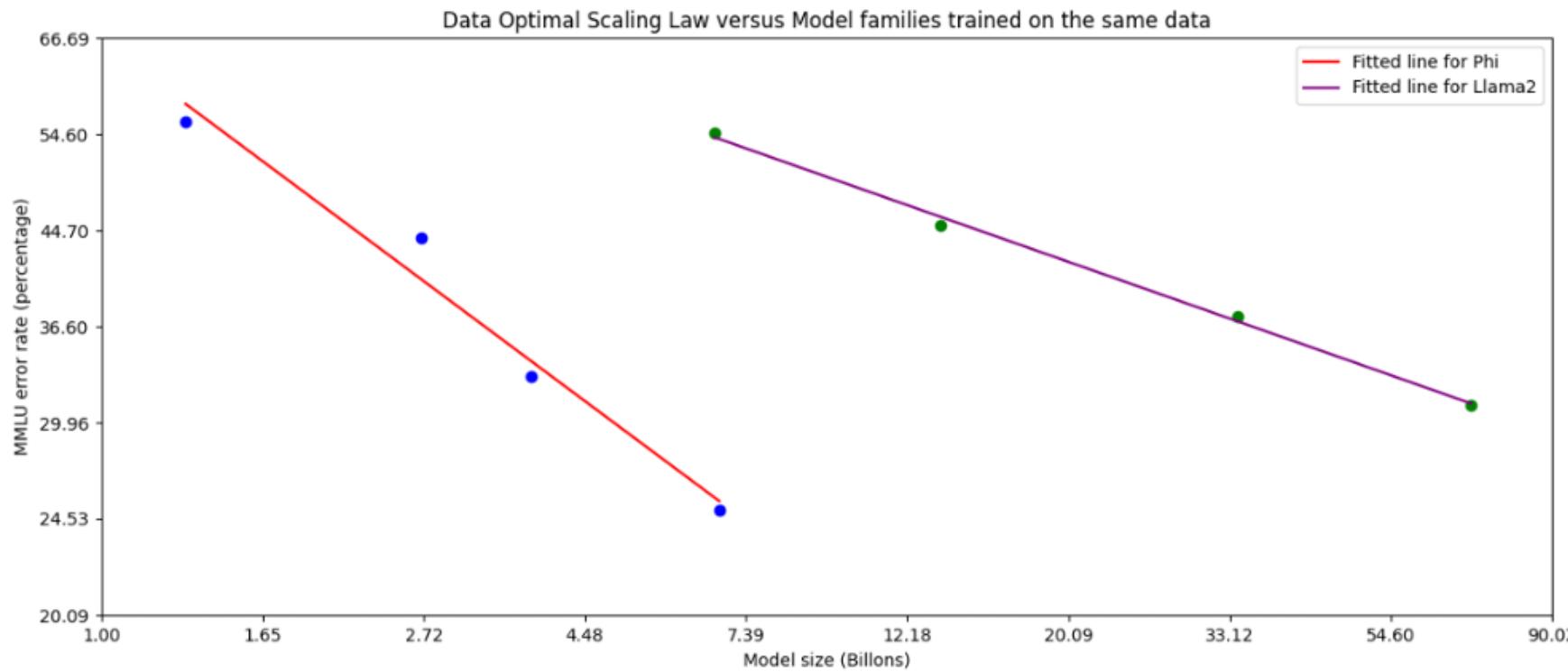
Features	Phi-3-mini	Phi-3-small	Phi-3-medium
Parameters	3.8B	7B	14B
Vocabulary size	32064	100352	100352
No. of Heads	32	32	40
No. of layers	32	32	40
Embedding dimension	3072	4096	5120
Context length	4000	8192	8192

- Applied GEGLU activation function
- Tuned hyper parameters by Maximal Update Parametrization on small proxy model and transfer them to target model
- Leveraged group query attention with 4 queries sharing 1 key
- Built a triton kernel based on flash attention
- Optimized the training and inference speed by enforcing blocksparse attention over each attention head
- Implemented alternate dense and blocksparse attention layers
- KV cache reduction
- Phi-3.5-MoE specifically activate part of modules on specific inputs. Out of 16x3.8 billion parameter model, have only 6.6B parameters activated



Training Methodology

- Utilized high quality training data to improve SLM, which deviates from standard scaling laws.
- Training data consists of heavily filtered publicly available resources from internet sources and synthetic LLM –generated data
- Phase 1 : Comprises web sources aimed at teaching the model general knowledge and language understanding
- Phase 2: Merges heavily filtered data with some synthetic data, to teach model logical reasoning



Scaling law close to Data Optimal Regime (from left to right : phi-1.5, phi-2, phi3-mini, phi-3-small) versus Llama-2 family of models (7B, 13B, 34B, 70B) that were trained on the same fixed data.

Post Training

Stage 1:

Supervised Finetuning (SFT) - Curated high-quality data, across diverse domain (math, coding, reasoning, conversation, safety)

Stage 2:

Direct preference Optimization (DPO) – It covers chat format data, reasoning, responsible AI. DPO is used to steer the model away from unwanted behaviour by using outputs as “rejected” responses.

Academic benchmarks

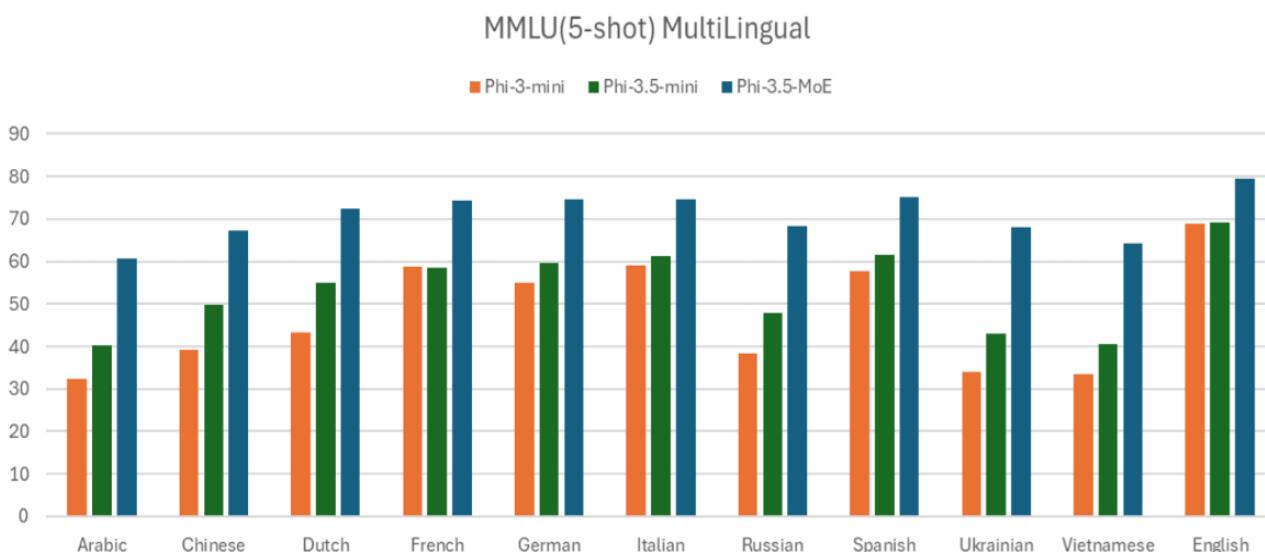
	Phi-3-mini 3.8b	Phi-3-small 7b	Phi-3-medium 14b	Phi-2 2.7b	Mistral 7b	Gemma 7b	Llama-3-In 8b	Mixtral 8x7b	GPT-3.5 version 1106
MMLU (5-Shot) [HBK ⁺ 21a]	68.8	75.7	78.0	56.3	61.7	63.6	66.5	70.5	71.4
HellaSwag (5-Shot) [ZHB ⁺ 19]	76.7	77.0	82.4	53.6	58.5	49.8	71.1	70.4	78.8
ANLI (7-Shot) [NWD ⁺ 20]	52.8	58.1	55.8	42.5	47.1	48.7	57.3	55.2	58.1
GSM-8K (8-Shot; CoT) [CKB ⁺ 21]	82.5	89.6	91.0	61.1	46.4	59.8	77.4	64.7	78.1
MATH (0-Shot; CoT) [HBK ⁺ 21b]	41.3	34.6	53.1	—	15.0	13.6	28.2	11.1	45.3
MedQA (2-Shot) [JPO ⁺ 20]	53.8	65.4	69.9	40.9	50.0	49.6	60.5	62.2	63.4
AGIEval (0-Shot) [ZCG ⁺ 23]	37.5	45.1	50.2	29.8	35.1	42.1	42.0	45.2	48.4
TriviaQA (5-Shot) [JCWZ17]	64.0	58.1	73.9	45.2	75.2	72.3	67.7	82.2	85.8
Arc-C (10-Shot) [CCE ⁺ 18]	84.9	90.7	91.6	75.9	78.6	78.3	82.8	87.3	87.4
Arc-E (10-Shot) [CCE ⁺ 18]	94.6	97.0	97.7	88.5	90.6	91.4	93.4	95.6	96.3
PIQA (5-Shot) [BZGC19]	84.2	86.9	87.9	60.2	77.7	78.1	75.7	86.0	86.6
SociQA (5-Shot) [BZGC19]	76.6	79.2	80.2	68.3	74.6	65.5	73.9	75.9	68.3
BigBench-Hard (3-Shot; CoT) [SRR ⁺ 22, SSS ⁺ 22]	71.7	79.1	81.4	59.4	57.3	59.6	51.5	69.7	68.32
WinoGrande (5-Shot) [SLBBC19]	70.8	81.5	81.5	54.7	54.2	55.6	65.0	62.0	68.8

OpenBookQA (10-Shot) [MCKS18]	83.2	88.0	87.4	73.6	79.8	78.6	82.6	85.8	86.0
BoolQ (2-Shot) [CLC ⁺ 19]	77.2	84.8	86.5	—	72.2	66.0	80.9	77.6	79.1
CommonSenseQA (10-Shot) [THLB19]	80.2	80.0	82.8	69.3	72.6	76.2	79.0	78.1	79.6
TruthfulQA (10-Shot; MC2) [LHE22]	65.0	70.2	75.1	—	53.0	52.1	63.2	60.1	85.8
HumanEval (0-Shot) [CTJ ⁺ 21]	58.5	61.0	62.2	59.0	28.0	34.1	60.4	37.8	62.2
MBPP (3-Shot) [AON ⁺ 21]	70.0	71.7	75.2	60.6	50.8	51.5	67.7	60.2	77.8
Average	69.7	73.6	76.7	—	58.9	59.3	67.3	66.8	72.8
GPQA (2-Shot; CoT) [RHS ⁺ 23]	32.8	34.3	—	—	—	—	—	—	29.0
MT Bench	8.38	8.70	8.91	—	—	—	—	—	8.35

Multilingual and long context

Phi-3.5-mini and phi-3.5 MoE incorporated more multi lingual and long text-data during mid training. Employed long rope and mixed context window approach to extend context length from 4k to 128k

Model	Ctx Size	Python	C++	Rust	Java	TypeScript	Average
gpt-4O-2024-05-13	128k	95	80	85	96	97	90.6
gemini-1.5-flash-latest	1000k	93	79	87	94	97	90
Phi-3.5-MoE	128k	89	74	81	88	95	85
Phi-3.5-Mini	128k	86	67	73	77	82	77
Llama-3.1-8B-Instruct	128k	80	65	73	76	63	71
Mixtral-8x7B-Instruct-v0.1	32k	66	65	64	71	74	68
Mixtral-8x22B-Instruct-v0.1	64k	60	67	74	83	55	67.8

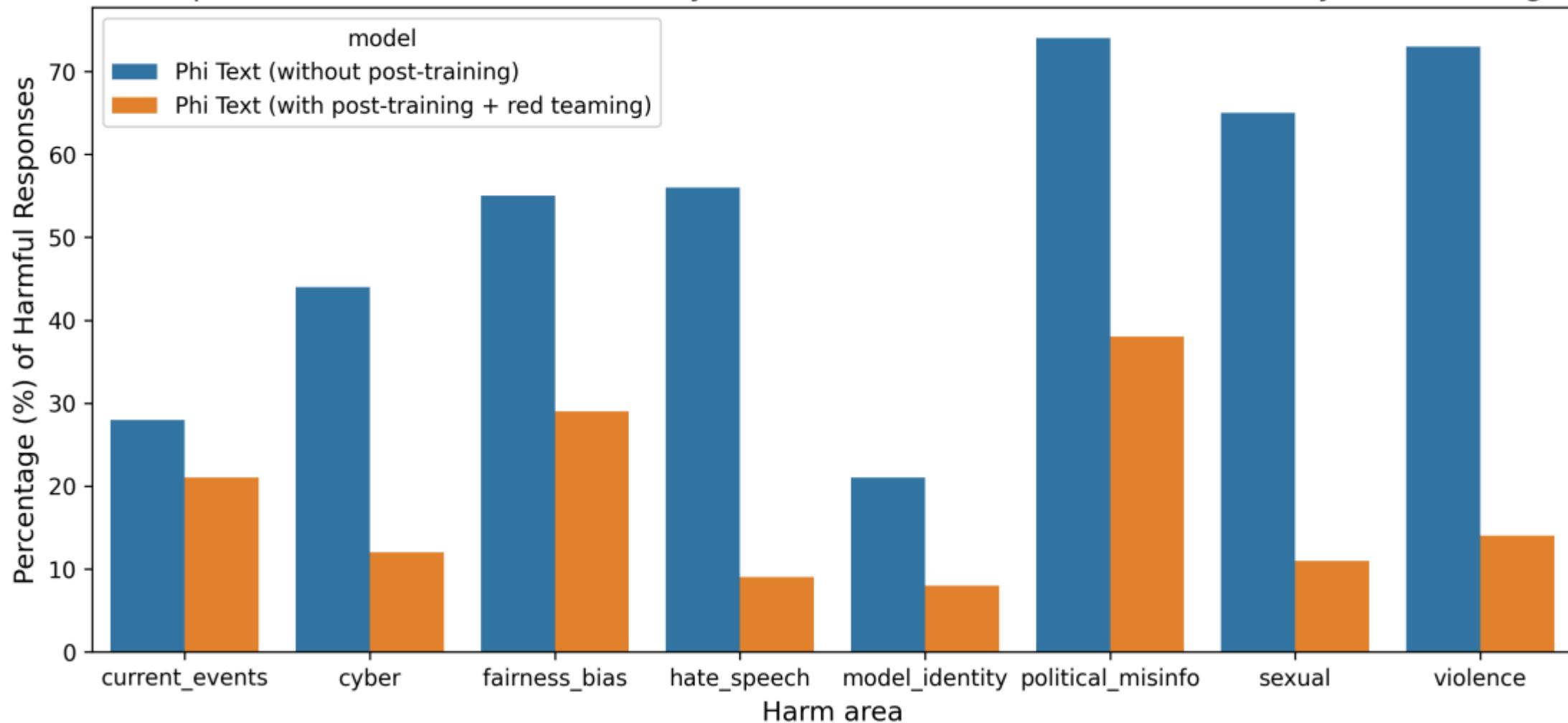


Comparison results on RepoQA benchmark

Safety Alignment

- **Responsible AI Principles:** Phi-3-mini model follows Microsoft's responsible AI principles to ensure safe and ethical AI use.
- **Red-Teaming:** Independent red team provided feedback to refine the model's safety features.
- **Dataset Refinements:** Additional datasets were curated based on red team insights.
- **Larger Models:** Applied the same process to phi-3-small, medium, and phi-3.5-MoE.
- **Harm Reduction:** Achieved a significant drop in harmful responses.
- **Evaluation:** Evaluated models against peers like Mistral-7b and Llama-3-instruct-8b.

Comparison of Content Generated by Phi Text Models Before and After Safety Post-Training



	Phi-3-mini 3.8b	Phi-3-small 7b	Phi-3-medium 14b	Phi-3.5-MoE 16x3.8b	Phi-2 2.7b	Mistral 7b	Gemma 7b	Llama-3-In 8b
Ungroundedness	0.603	0.299	0.213	0.228	1.481	0.935	0.679	0.328
Third Party Harm (DR-1)	0.240	0.253	0.251	0.105	0.240	0.562	0.383	0.373
Harmful Content Continuation (DR-3)	0.007	0.003	0.010	0.005	0.029	0.026	0.013	0.013
Harmful Content Summarization (DR-3)	0.100	0.110	0.112	0.12	0.144	0.223	0.103	0.082
Jailbreak (DR-1)	0.123	0.107	0.111	0.106	0.150	0.156	0.114	0.130

Comparison of Microsoft internal multi-turn conversation RAI benchmark results for phi-3 and other models, where lower values indicate better performance across all metrics

Weakness

- **Size Limitation:** Phi-3-mini's smaller size limits its factual knowledge capacity.
- **TriviaQA Performance:** Low performance observed on TriviaQA.
- **Search Augmentation:** Weakness mitigated by integrating search engine capabilities
- **Language Restriction:** Primarily supports English; exploring multilingual potential is a priority.
- **Factual Inaccuracies:** Challenges with hallucinations, inappropriate content generation, and biases remain.
- **Safety Considerations:** Significant work is needed to address safety concerns for diverse use cases.

Day 1: Arrival and Surrounding Neighborhoods

Morning:

- Arrive in Pyeongchang, South Korea, and check-in to your accommodations.
- Enjoy a light breakfast at your hotel.

Mid-morning:

- Take a leisurely walk around the town and explore the local shops and cafes.

Lunch:

- Try local specialties such as kimbap (Korean rice rolls) or mandu (Korean dumplings) at a nearby restaurant.

Afternoon:

- Visit Pyeongchang Olympic Cauldron Park, the site of the 2018 Winter Olympics, to take photos and admire the beautiful scenery.
- Rent a bike and explore the surrounding neighborhoods, such as Pyeongchang Village, where you can find souvenir shops, outdoor activities, and various dining options.

Evening:

- Dine at a local restaurant and enjoy traditional Korean cuisine such as samgyeopsal (grilled pork belly) or bulgogi (marinated beef).

Day 2: Olympic Sports and Cultural Activities

Morning:

- Visit the Alpensia Sports Park, where the majority of the 2026 Winter Olympic events will be held.
- See the ski slopes, ice rinks, and indoor arenas, and watch a glimpse of the opening and closing ceremonies.

Mid-morning:

- Participate in a guided tour to learn about the history of the 2026 Winter Olympics and the unique features of the Pyeongchang venues.

Search web 

Ask anything

Web Search
Completed

Day 1: Arrival and Exploring Milan

- Arrive in Milan on February 6, 2026, the day of the Opening Ceremony.
- Spend the day exploring Milan's city center, including Piazza del Duomo, Galleria Vittorio Emanuele II, and the Sforza Castle.
- Enjoy the vibrant Milanese nightlife, with options like the city's many trendy cafes and restaurants.
- Visit the new Fiera Milano expo center for a unique and inspiring experience.

Day 2: Cortina d'Ampezzo and Outdoor Activities

- Take the train from Milan to Cortina d'Ampezzo, arriving in the afternoon.
- Check into your hotel and spend the evening relaxing and enjoying the local cuisine, such as risotto and polenta, at traditional pizzerias or trattorias.
- On Day 2, take part in various outdoor activities, such as skiing, snowboarding, or tubing. Many of these activities are available at nearby resorts like Madonna di Campiglio, Kronplatz, or Passo San Pellegrino.
- In the evening, explore Cortina d'Ampezzo's charming streets, visit the historic St. Felix Castle, and enjoy the beautiful mountain scenery.

Day 3: Paralympic Games and Departure

- On Day 3, attend the Paralympic Games, held in Turin on March 6, 2026.
- Take the train from Cortina d'Ampezzo to Turin, arriving in time for the Paralympic events.
- Explore Turin's attractions, such as the Mole Antonelliana, the Egyptian Museum, and Piazza Vittorio Veneto.
- Return to Milan in the evening for your departure, concluding your 3-day itinerary during the Milano Cortina 2026 Winter Olympic Games.

Throughout your trip, make sure to book your accommodations, transportation, and tickets in advance, as these events are highly popular and may sell out quickly. Enjoy your time in Italy and the exciting Winter Olympic Games!

Sources: [olympics.com](#) [olympics.com](#) [milancortina2026.olympics.com](#) [milanwintertravel.com](#)

olympics.com

Search web 

Ask anything

Comparison of phi-3-mini's responses without search and with search using HuggingFace Chat-UI.

Phi-3.5-Vision

Multimodal Model:

Processes images/multi-images and textual prompts to generate text outputs.

Key Features:

- Designed with 4.2 billion parameters for handling multimodal inputs.
- Combines image tokens and text tokens in an interleaved manner.
- Dynamic cropping strategy ensures compatibility with high-resolution images.
- Supports multi-image inputs by concatenating tokens from each image.

	SM	MED	LG	CUP	POT
Espresso	1.30	1.60	2.00	1.50	4.00
Americana	1.40	1.75	2.10		
Cappuccino	2.25	2.40	3.25		
Latte	2.35	3.00	3.25		
Mocha	2.35	3.00	3.50		
Chai	2.25	2.40	3.25		
Chai Bomb	2.75	3.25	3.75		
House Cafe	1.50	1.75	2.00		
Cafe au Lait	1.75	2.50	3.25		

Teas

GREEN
BLACK
WHITE
OOLONG
HERBAL

* ADD .50 FOR HONEY
* ADD \$1.00 FOR ICE

What is the price of a Cappuccino with large size?

- The price for a Cappuccino in large size is \$3.25.

How much does it cost to add ice to a tea?

- It costs an additional \$1.00 to add ice to any tea.

If someone wants to buy a pot of tea, how much would it cost?

- A pot of tea would cost \$4.00.

extract all displayed texts and keep original displayed format. For the left section, please use markdown table with column "coffee", "small", "median" and "large". Instead, for the right section, please use JSON format.

Certainly! Below is the markdown table converted from the image:

Coffee	Small	Medium	Large
Espresso	1.30	1.60	2.00
Americana	1.40	1.75	2.10
Cappuccino	2.25	2.40	3.25
Latte	2.35	3.00	3.25
Mocha	2.35	3.00	3.50
Chai	2.25	2.40	3.25
Chai Bomb	2.75	3.25	3.75
House Cafe	1.50	1.75	2.00
Cafe au Lait	1.75	2.50	3.25

And here is the JSON representation of the teas offered:

```
{
  "Teas": {
    "CUP": {
      "GREEN": "$1.50",
      "BLACK": "$1.50",
      "WHITE": "$1.50",
      "OOLONG": "$1.50",
      "HERBAL": "$1.50"
    },
    "POT": {
      "GREEN": "$4.00",
      "BLACK": "$4.00",
      "WHITE": "$4.00",
      "OOLONG": "$4.00",
      "HERBAL": "$4.00"
    }
  },
  "*ADD .50 FOR HONEY*",
  "*ADD $1.00 FORICE*"
}
```

Architecture of Phi-3.5-Vision

Technical Composition:

- **Image Encoder:** Utilizes CLIP ViT-L/14 to extract visual tokens from images.
(CLIP ViT-L/14 is a model that combines image and text understanding – Vision T/F)
- **Transformer Decoder:** Built upon the phi-3.5-mini architecture for generating textual outputs.
- **Dynamic Cropping:** Splits high-resolution images into a 2D array of blocks, where the concatenated tokens represent the whole image.
- **Multi-Image Capability:** Processes multi-images by concatenating tokens from all the images together.

Pre-Training of Phi-3.5-Vision

Pre-Training Details:

- **Dataset Composition:**
 - Interleaved image-text documents, Image-text pairs, OCR-generated synthetic data from PDF files, chart/table comprehension datasets, and Text-only datasets.
- **Training Strategy:**
 - Pre-trained on 0.5 trillion tokens combining text and visual data.
 - Model focuses on predicting the next word in a sequence of text.
 - Maximum Image resolution capped at 1344×1344 as most of the training images were smaller than this resolution.

Post-Training of Phi-3.5-Vision

Post-Training Stages:

- **Supervised Fine-Tuning (SFT):**
 - Multimodal SFT data with 33B tokens covering diverse domains: natural images, chart/table/diagram, PowerPoint, video summarization, etc.
- **Direct Preference Optimization (DPO):**
 - Training model to directly optimize human preferences.
 - Joint training of multimodal and text-only tasks for achieving multimodal reasoning.

Benchmarks: Single-Image Benchmarks

Overview: Phi-3.5-Vision is evaluated on nine open-source academic benchmarks that assess reasoning and perceptual abilities. These benchmarks span three categories: Science, Charts, and Generic Knowledge.

Evaluation Setup:

- Baseline comparison includes MM1-3B-Chat, MM1-7B-Chat, Llava-1.6 Vicuna 7B, Llava-1.6 Llama3-8B, Qwen-VL-Chat, Claude 3 Haiku, Gemini 1.0 Pro V, and GPT-4O.
- It mimics real-world user interaction, i.e., users who are not experts in prompt engineering or know special techniques that can improve performance.
- Prompts include instructions for simple responses like selecting an answer from a list of options or answering with a word/phrase.

Key Results: Phi-3.5-Vision achieves super competitive results on all benchmarks and outperforms the other competitor models on most benchmarks while being smaller.

	Phi-3.5-Vision 4.2b	MM1-3B-Chat 3.6b [MGF ⁺ 24]	MM1-7B-Chat 7.6b [MGF ⁺ 24]	Vicuna-1.6 Vicuna-7b [LLL23]	LLaVA-1.6 LLaVA-Next [LLL24]	Qwen-VL-Chat 9.6b [BBY ⁺ 23]	Claude 3 haiku [Ant24]	Gemini 1.0 Pro V [TAB ⁺ 23]	GPT-4O 2024-05-13
MMMU (val) [YNZ ⁺ 23]	43.0	33.9	37.0	34.2	36.4	39.0	40.7	42.0	61.8
ScienceQA (test) [LMX ⁺ 22]	91.3	69.4	72.6	70.6	73.7	67.2	72.0	79.7	88.5
MathVista (testmini) [LBX ⁺ 24]	43.9	32.0	35.9	31.5	34.8	29.4	33.2	35.0	54.4
Inter-GPS (test) [LGJ ⁺ 21]	36.3	-	-	20.5	24.6	22.3	32.1	28.6	46.9
MMBench (dev-en) [LDZ ⁺ 24]	81.9	75.9	79.0	76.3	79.4	75.8	62.4	80.0	88.4
POPE (test) [LDZ ⁺ 23]	86.1	87.4	86.6	87.2	87.0	82.6	74.4	84.2	87.0
AI2D (test) [KSK ⁺ 16]	78.1	-	-	63.1	66.9	59.8	60.3	62.8	82.8
ChartQA (test) [MLT ⁺ 22]	81.8	-	-	55.0	65.8	50.9	59.3	58.0	64.0
TextVQA (test) [SNS ⁺ 19]	72.0	71.9	72.8	64.6	55.7	59.4	62.7	64.7	75.6

Comparison results on public MLLM benchmarks, with all numbers produced using the same pipeline for consistency, except for MM1-3B-Chat and MM1-7B-Chat, which are not publicly available, and slight variations in prompts used

Benchmarks: Multi-Image Benchmarks

Overview: Phi-3.5-Vision evaluated on one academic multi-image benchmark and one video benchmark, assessing perpetual capabilities on multiple images/frames and text covering a wide range of general scenarios (e.g., Art & Style recognition, Forensic detection, and video understanding).

Evaluation Setup:

- Baseline comparison includes Llava Interleave-Qwen 7B, InternVL2 4B/8B, Gemini 1.5 Flash, GPT-4o-mini, Claude 3.5 Sonnet, and GPT-4O.
- The evaluation uses a 0-shot format, with no scaling or preprocessing of images
- Prompts include instructions for selecting answers from a list or providing a word/phrase.
- For most of the benchmarks, images were placed as the first item in the prompt.

Key Results: Phi-3.5-Vision performs very competitively or outperforms baseline models under similar model sizes in multi-image understanding scenarios as well.

	Phi-3.5-Vision 4.2b	Llava-interleave Qwen 7b [LZZ ⁺ 24]	InternVL2 4b [CWT ⁺ 24]	InternVL2 8b [CWT ⁺ 24]	Gemini 1.5 Flash [TAB ⁺ 23]	GPT4O mini 2024-07-18	Claude 3.5 Sonnet [Ant24]	Gemini 1.5 Pro [TAB ⁺ 23]	GPT-4O 2024-05-13
BLINK (val) [FHL ⁺ 24]	57.0	53.1	45.9	45.4	45.8	51.9	56.5	61.0	63.2
VideoMME (test) [FDL ⁺ 24]	50.8	50.2	49.9	52.6	62.3	61.2	55.9	62.6	68.4

Comparison results on public multi-image/video MLLM benchmarks, with all numbers generated using the same pipeline to ensure comparability.

Safety

Overview: Phi-3.5-Vision integrates safety post-training in both Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO) stages to align with Microsoft's Responsible AI (RAI) principles.

Training Dataset: Utilized a combination of text-only and in-house Multi-Modal (MM) RAI datasets, covering a variety of harm categories from both public and internal MM RAI benchmarks.

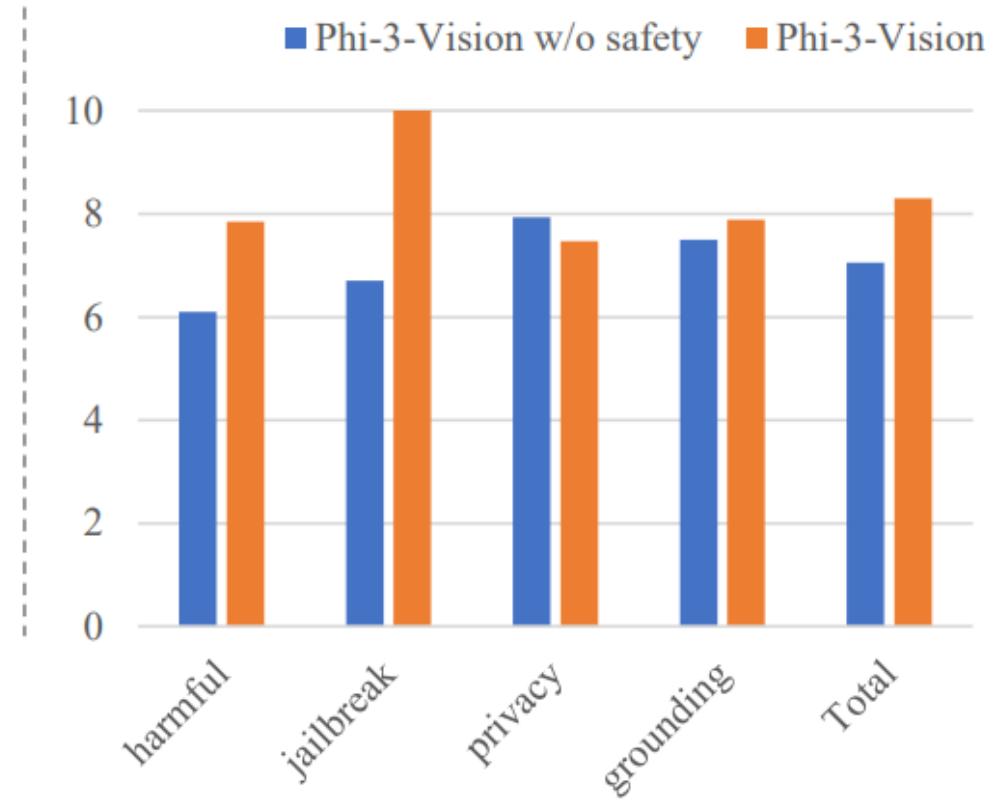
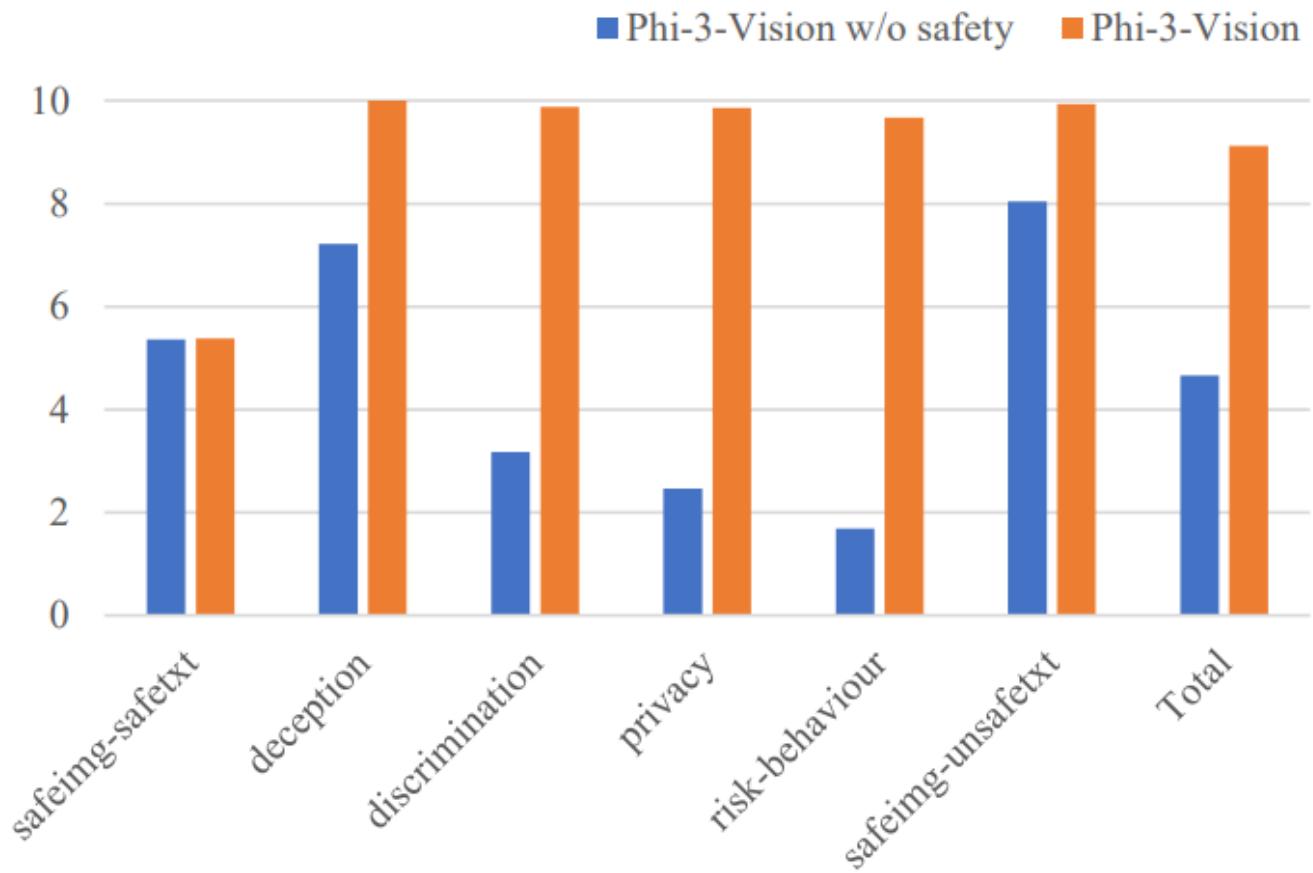
Evaluation:

- Performed rigorous quantitative assessment on both public and internal benchmarks.
- Human evaluation conducted by Microsoft's internal red team.

Key Results: Safety post-training significantly enhances RAI performance across all benchmarks, with improvements seen in nearly all RAI categories

	Phi-3.5-Vision 3.8b+0.3b	Phi-3.5-Vision w/o safety 3.8b+0.3b	Llava-1.6 Vicuna 7b+0.3b	Qwen-VL-Chat 7.7b+1.9b	GPT4-V N/A
Internal (private)	8.16	7.06	5.44	7.27	8.55
RTVLM (public)	5.44	3.56	3.86	4.78	6.81
VLGuard (public)	9.10	4.66	5.62	8.33	8.90

Comparison results on public and private multi-modal RAI benchmarks, with all metrics on a [0,10] scale where a higher value indicates better performance.



Comparison of Phi-3.5-Vision's categorized RAI performance with and without safety post-training on the VLGuard (left) and Internal (right) benchmarks, showing significant improvements in nearly all RAI categories.

Weakness

Overview:

While Phi-3.5-Vision performs well across many tasks, certain weaknesses remain, particularly in high-level reasoning and generating unreliable outputs in sensitive areas.

Identified Limitations:

- Occasional failure in reasoning tasks requiring advanced logic.
- Generation of ungrounded or unreliable outputs, especially in sensitive areas like finance.
- Safety post-training improvements do not fully mitigate issues with harmful or sensitive responses.

Future Plans:

- Incorporating more reasoning-focused and hallucination-related data into post-training.
- Further exploration to achieve a better balance between helpfulness and harmlessness, especially in areas like OCR and scam image identification.

Critical Evaluation: Strengths

Innovation:

- Phi-3-mini brings advanced AI to mobile devices while focusing on data optimization instead of large model sizes.

Technical Soundness:

- The model follows a rigorous two-phase training process with a focus on high-quality data. Techniques like blocksparse attention and LongRope improve efficiency, while comprehensive benchmarking and safety measures ensure reliable performance.

Critical Evaluation: Weaknesses

Gaps in Research:

Lack of real-world performance analysis beyond benchmarks.

Methodological Issues:

Lack of transparency in data filtering criteria and safety evaluation comprehensiveness.

Clarity Issues:

Complex technical concepts like block sparse attention and LongRope need clearer explanations.

Conclusion

This paper highlights how small models like Phi-3-mini can perform well by using efficient design and optimized data.

It challenges the idea that bigger models are always better, showing that AI can be both accessible and resource-efficient.

Although more work is needed in areas like real-world testing and reducing biases, the paper offers valuable insights for creating AI systems that are more inclusive, responsible, and adaptable.

Thank You!

- Janvita Reddy Peddagudreddygari (634005984)
- Varun Sahni (834006628)

LLama 3 Herd of Models

Presentation by Shreyas S Kasetty & Alex Zheng

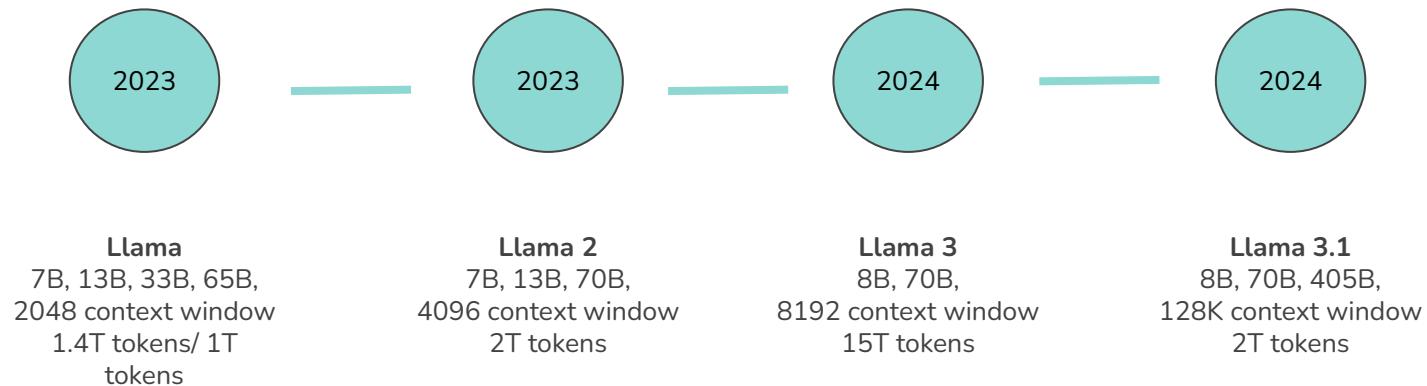
1. Introduction

Timeline, Overview





Timeline



	Finetuned	Multilingual	Long context	Tool use	Release
Llama 3 8B	✗	✗ ¹	✗	✗	April 2024
Llama 3 8B Instruct	✓	✗	✗	✗	April 2024
Llama 3 70B	✗	✗ ¹	✗	✗	April 2024
Llama 3 70B Instruct	✓	✗	✗	✗	April 2024
Llama 3.1 8B	✗	✓	✓	✗	July 2024
Llama 3.1 8B Instruct	✓	✓	✓	✓	July 2024
Llama 3.1 70B	✗	✓	✓	✗	July 2024
Llama 3.1 70B Instruct	✓	✓	✓	✓	July 2024
Llama 3.1 405B	✗	✓	✓	✗	July 2024
Llama 3.1 405B Instruct	✓	✓	✓	✓	July 2024

Overview of the Llama 3 Herd of models. All results in this paper are for the Llama 3.1 models.



Three Key Levers

1. Data
 - a. Improved quality and quantity
 - b. 15T multilingual tokens, compared to 1.8T tokens for Llama 2
2. Scale
 - a. 50x more than the largest version of Llama 2
 - b. flagship language model was pre-trained using 3.8×10^{25} FLOPs
3. Managing Complexity
 - a. **Focus on Simplicity and Stability:** standard dense Transformer model over more complex architectures, such as mixture-of-experts.
 - b. **Streamlined Post-Training Process:** simpler post-training methods like supervised fine-tuning, rejection sampling, and direct preference optimization instead of complex reinforcement learning algorithms

2. Innovations

Flagship Model, Context Window, Scaling Laws,
Multilingual Support, Safety, Multimodal Integration



New Foundation Models:

- Native support for multilinguality, coding, reasoning, and tool usage
- 8B parameters, 70B parameters, 405B parameters

405 Parameter Model

- Focus of the paper
- Performs on par with leading closed-source language models such as GPT-4

Context Window upgrade:

- Context window is now up to **128K tokens**
- Llama 3 had **8192 token** context window

Multilingual Support

- Improved tokenizer to support multiple languages. Training on a very large and diverse multilingual corpus

Multimodal Capabilities:

- Experiments on image, video, and speech
- Feature not released

Safety - Llama Guard 3

- Created a new model to ensure input and output safety. Objective was to reduce harmful or biased content.

3. Pre-Training

Data Filtering, Model Architecture, Scaling Laws,
Infrastructure, Scaling, and Efficiency, Training Recipe





Data Filtering and Curation

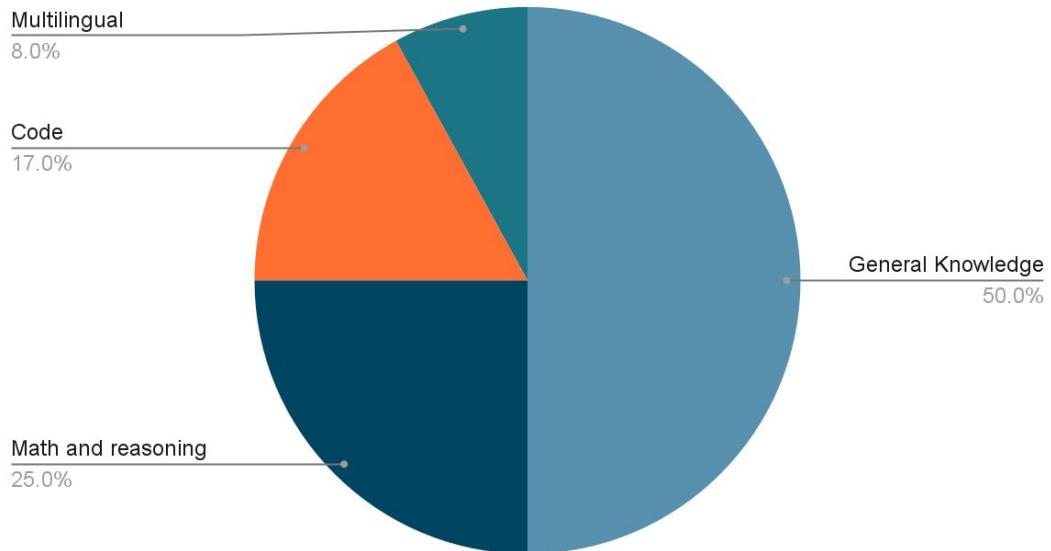
Filtering Techniques:

- **Text Extraction Cleaning** - Custom HTML parser for precise boilerplate removal and content recall, removes **markdown formatting for improved model performance**
- **Heuristic filtering** - Removes low-quality documents using many techniques like duplicated n-gram.
- **Model-based quality filtering** - FastText and Distil Roberta assess document quality
- **Code and Reasoning Data** - Domain-specific pipelines for math and code extraction, Distil Roberta models tuned for reasoning and STEM-related documents
- **Multilingual Data** - Language Identification, Language specific heuristics, Quality ranking using multilingual llama 2-based classification



Data Mix Summary

Data Mix





Model Architecture

	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8192	16,384
FFN Dimension	14,336	28,672	53,248
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning Rate	3×10^{-4}	1.5×10^{-4}	8×10^{-5}
Activation Function	SwiGLU		
Vocabulary Size	128,000		
Positional Embeddings	RoPE ($\theta = 500,000$)		

Overview of the key hyperparameters of Llama 3. We display settings for 8B, 70B, and 405B language models.

- Group Query Attention (8 KV heads)
- Extended tokenizer: token vocabulary combines **100K tokens** from the **tiktoken3** tokenizer with **28K additional tokens** to better support non-English languages
- Positional Embedding (RoPE)
- Attention Masking



Scaling Laws

Two-Stage Methodology:

- **Step 1:** Correlate compute-optimal model's negative log-likelihood (NLL) on downstream tasks with training FLOPs.
- **Step 2:** Link NLL to task accuracy using scaling law models and Llama 2 models, creating a robust performance forecast.

Scaling Law Experiments:

- Models with sizes from 40M to 16B parameters were trained on compute budgets ranging from 6×10^{18} to 10^{22} FLOPs.
- Predicted optimal configurations include training a 402B parameter model on 16.55T tokens with 3.8×10^{25} FLOPs.

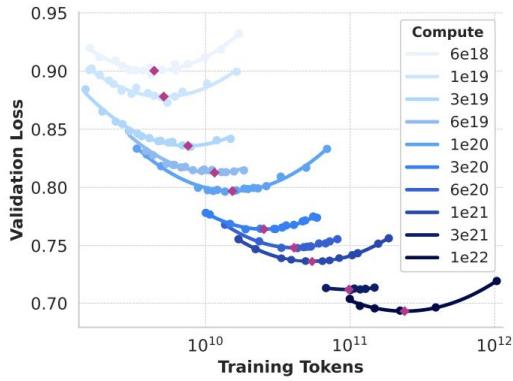


Figure 2 **Scaling law IsoFLOPs curves** between 6×10^{18} and 10^{22} FLOPs. The loss is the negative log-likelihood on a held-out validation set. We approximate measurements at each compute scale using a second degree polynomial.

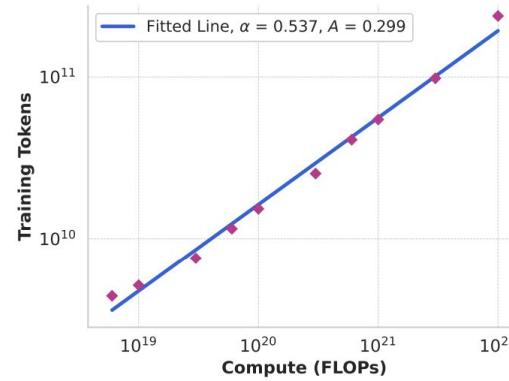


Figure 3 **Number of training tokens in identified compute-optimal models as a function of pre-training compute budget.** We include the fitted scaling-law prediction as well. The compute-optimal models correspond to the parabola minima in Figure 2.



Infrastructure

Massive GPU Deployment:

- Trained on 16,000 NVIDIA H100 GPUs, some of the most powerful GPUs available, interconnected via NVLink for high-speed data exchange.

Efficient Job Scheduling:

- Managed by MAST, a super-efficient job scheduler, ensuring optimal resource utilization during the training process.

High-Performance Storage:

- Used Tectonic, a storage system capable of handling 240 petabytes of data at speeds of up to 7 terabytes per second, ensuring GPUs are constantly supplied with data without delays.

Advanced Networking:

- Custom-designed RoCE (RDMA over Converged Ethernet) fabric minimized bottlenecks and enabled seamless communication between GPUs.

4D Parallelism:

- Leveraged advanced parallelism techniques to efficiently distribute computational tasks, critical for training a model of this scale.

4. Post-Training

Modeling, Post-training Data, Capabilities





Post Training Process

The post-training process for Llama 3 involves multiple rounds of refinement and alignment with human feedback

1. Supervised Fine-Tuning
2. Rejection Sampling
3. Direct Preference Optimization



Data Cleaning

- To mitigate overly-apologetic tonal issues, they identified overused phrases (such as “I’m sorry” or “I apologize”) and carefully balance the proportion of such samples in our dataset.
- They trained a **code expert** which they use to collect high quality human annotations for code throughout subsequent rounds of post-training.
- They trained a **multilingual expert model** to collect higher quality annotations in non-English languages until pre-training was fully complete



Capabilities - Code Generation

- **Expert Training:** Specialized code expert model trained with 1T tokens of mostly code data, fine-tuned for long-context handling (16K tokens).
- **Synthetic Data Generation:**
 - Used execution feedback to self-correct errors, improving code accuracy and quality.
 - Translated code between programming languages to enhance performance in less common languages.
 - Back Translation for improving documentation and explanation capabilities.
- **System Prompt Steering:** Enhanced code readability, documentation, and correctness through tailored system prompt.



Capabilities - Multilinguality

- **Expert Model:** Trained on 90% multilingual data to collect higher-quality non-English annotations.
- **Data Sourcing:**
 - Combined human annotations, multilingual NLP tasks, rejection-sampled data, and translated reasoning data.
 - Used multilingual-specific checks and templates to improve language alignment and cultural representation.
- **Performance Gains:** Strong improvements on multilingual benchmarks (e.g., MGSM) by fine-tuning with diverse, high-quality data.



Capabilities - Math and Reasoning

- **Challenges:** There is a shortage of ground truth chains of thought, which are essential for guiding the model how to break down the problem step-by-step and reach the final answer
- **Addressing Challenges:**
 - Augmented prompts and step-by-step reasoning traces to enhance understanding.
 - Trained reward models to filter invalid reasoning and used Monte Carlo Tree Search (MCTS) for generating valid traces.
- **Interleaving Text and Code:**
 - Solved problems by combining textual reasoning with Python code execution to verify correctness.
- **Feedback Learning:** Iteratively corrected errors by using feedback signals to refine reasoning capabilities.

5. Results

Pre-training, Post-Training Benchmark Evaluation, Human Evaluation



Pre-trained Model Evaluation

- **Evaluation Areas:**
 - **Standard Benchmarks:** Benchmarked across 8 categories (commonsense, Knowledge, reading, math, code, long-context, adversarial, aggregate).
 - **Robustness:** Tested for variations in prompts, labels, and answer orders
 - **Adversarial Benchmarks:** Performance on challenging datasets for reasoning, QA, and paraphrase detection
 - **Contamination Analysis:** Assessed training data overlap with evaluation datasets.
- **Key Results:**
 - **Performance:**
 - i. Llama 3 405B leads in most benchmarks , rivaling GPT-4 in areas like coding and math reasoning
 - ii. Smaller models (8B, 70B) outperformed Llama 2 and similar-sized competitors
 - **Robustness:** High resilience to prompt and setup changes, especially 405B model
 - **Adversarial Strength:** Strong in paraphrase detection, but lower scores in adversarial QA and math reasoning
 - **Contamination:** Minimal impact on key benchmarks like SQuAD and Math



Post-trained Model Evaluation

- **Evaluation Areas:**
 - **Benchmarks:** General Knowledge, instruction-following, coding, multilinguality, math/reasoning, long-context tasks, and tool use.
 - **Human Evaluation:** Focused on code execution, plot generation, and file uploads.
- **Key Results:**
 - **General Knowledge:** 405B outperforms GPT-4 on MMLU; leads instruction-following tasks on IFEval
 - **Proficiency Exams:** Top scores in GRE, SAT, and AP tests; matches GPT-4o in overall performance
 - **Coding:** Best in Python (HumanEval, MBPP); competitive in multi-language benchmarks.
 - **Multilinguality:** Strong results in 8 languages; trails GPT-4o slightly but leads open models.
 - **Math & Reasoning:** Excels in GSM8K, ARC-C; competitive with GPT-4o and Claude 3.5 Sonnet.
 - **Long-Context:** Perfect in Needle-in-a-Haystack; outperforms GPT-4 on InfiniteBench QA.
 - **Tool Use:** Best on Nexus and API-Bank; slightly behind GPT-4o for file tasks.



Post-trained Model Evaluation

	ZeroSCROLLS			InfiniteBench		NIH
	QuALITY	Qasper	SQuALITY	En.QA	En.MC	Multi-needle
Llama 3 8B	81.0 ±16.8	39.3 ±18.1	15.3 ±7.9	27.1 ±4.6	65.1 ±6.2	98.8 ±1.2
Llama 3 70B	90.5 ±12.6	49.0 ±18.5	16.4 ±8.1	36.7 ±5.0	78.2 ±5.4	97.5 ±1.7
Llama 3 405B	95.2 ±9.1	49.8 ±18.5	15.4 ±7.9	30.5 ±4.8	83.4 ±4.8	98.1 ±1.5
GPT-4	95.2 ±9.1	50.5 ±18.5	13.2 ±7.4	15.7 ±3.8	72.0 ±5.8	100.0 ±0.0
GPT-4o	90.5 ±12.5	49.2 ±18.5	18.8 ±8.6	19.1 ±4.1	82.5 ±4.9	100.0 ±0.0
Claude 3.5 Sonnet	90.5 ±12.6	18.5 ±14.4	13.4 ±7.5	11.3 ±3.3	–	90.8 ±3.2

Table 21 Long-context benchmarks. For ZeroSCROLLS (Shaham et al., 2023), we report numbers on the validation set. For QuALITY we report exact match, for Qasper - f1 and for SQuALITY - rougeL. We report f1 for InfiniteBench (Zhang et al., 2024) En.QA metric and accuracy for En.MC. For Multi-needle (Kamradt, 2023) we insert 4 needles in the context and test if a model can retrieve 2 needles at different context lengths, we compute average recall across 10 sequence lengths up till 128k.

Exam	Llama 3 8B	Llama 3 70B	Llama 3 405B	GPT-3.5 Turbo	Nemtron 4 340B	GPT-4o	Claude 3.5 Sonnet
LSAT	53.9 ±4.9	74.2 ±4.3	81.1 ±3.8	54.3 ±4.9	73.7 ±4.3	77.4 ±4.1	80.0 ±3.9
SAT Reading	57.4 ±4.2	71.4 ±3.9	74.8 ±3.7	61.3 ±4.2	—	82.1 ±3.3	85.1 ±3.1
SAT Math	73.3 ±4.6	91.9 ±2.8	94.9 ±2.3	77.3 ±4.4	—	95.5 ±2.2	95.8 ±2.1
GMAT Quant.	56.0 ±19.5	84.0 ±14.4	96.0 ±7.7	36.0 ±18.8	76.0 ±16.7	92.0 ±10.6	92.0 ±10.6
GMAT Verbal	65.7 ±11.4	85.1 ±8.5	86.6 ±8.2	65.7 ±11.4	91.0 ±6.8	95.5 ±5.0	92.5 ±6.3
GRE Physics	48.0 ±11.3	74.7 ±9.8	80.0 ±9.1	50.7 ±11.3	—	89.3 ±7.0	90.7 ±6.6
AP Art History	75.6 ±12.6	84.4 ±10.6	86.7 ±9.9	68.9 ±13.5	71.1 ±13.2	80.0 ±11.7	77.8 ±12.1
AP Biology	91.7 ±11.1	100.0 ±0.0	100.0 ±0.0	91.7 ±11.1	95.8 ±8.0	100.0 ±0.0	100.0 ±0.0
AP Calculus	57.1 ±16.4	54.3 ±16.5	88.6 ±10.5	62.9 ±16.0	68.6 ±15.4	91.4 ±9.3	88.6 ±10.5
AP Chemistry	59.4 ±17.0	96.9 ±6.0	90.6 ±10.1	62.5 ±16.8	68.8 ±16.1	93.8 ±8.4	96.9 ±6.0
AP English Lang.	69.8 ±12.4	90.6 ±7.9	94.3 ±6.2	77.4 ±11.3	88.7 ±8.5	98.1 ±3.7	90.6 ±7.9
AP English Lit.	59.3 ±13.1	79.6 ±10.7	83.3 ±9.9	53.7 ±13.3	88.9 ±8.4	88.9 ±8.4	85.2 ±9.5
AP Env. Sci.	73.9 ±12.7	89.1 ±9.0	93.5 ±7.1	73.9 ±12.7	73.9 ±12.7	89.1 ±9.0	84.8 ±10.4
AP Macro Eco.	72.4 ±11.5	98.3 ±3.3	98.3 ±3.3	67.2 ±12.1	91.4 ±7.2	96.5 ±4.7	94.8 ±5.7
AP Micro Eco.	70.8 ±12.9	91.7 ±7.8	93.8 ±6.8	64.6 ±13.5	89.6 ±8.6	97.9 ±4.0	97.9 ±4.0
AP Physics	57.1 ±25.9	78.6 ±21.5	92.9 ±13.5	35.7 ±25.1	71.4 ±23.7	71.4 ±23.7	78.6 ±21.5
AP Psychology	94.8 ±4.4	100.0 ±0.0	100.0 ±0.0	94.8 ±4.4	100.0 ±0.0	100.0 ±0.0	100.0 ±0.0
AP Statistics	66.7 ±17.8	59.3 ±18.5	85.2 ±13.4	48.1 ±18.8	77.8 ±15.7	92.6 ±9.9	96.3 ±7.1
AP US Gov.	90.2 ±9.1	97.6 ±4.7	97.6 ±4.7	78.0 ±12.7	78.0 ±12.7	100.0 ±0.0	100.0 ±0.0
AP US History	78.0 ±12.7	97.6 ±4.7	97.6 ±4.7	85.4 ±10.8	70.7 ±13.9	95.1 ±6.6	95.1 ±6.6
AP World History	94.1 ±7.9	100.0 ±0.0	100.0 ±0.0	88.2 ±10.8	85.3 ±11.9	100.0 ±0.0	97.1 ±5.7
AP Average	74.1 ±3.4	87.9 ±2.5	93.5 ±1.9	70.2 ±3.5	81.3 ±3.0	93.0 ±2.0	92.2 ±2.1
GRE Quant.	152.0	158.0	162.0	155.0	161.0	166.0	164.0
GRE Verbal	149.0	166.0	166.0	154.0	162.0	167.0	167.0

Table 17 Performance of Llama 3 models and GPT-4o on a variety of proficiency exams including LSAT, SAT, GMAT, and AP, and GRE tests. For GRE exams, we report normalized score; for all others, we report accuracy. For the bottom two rows corresponding to GRE Quant. and GRE Verbal, we report the scaled scores out of 170.



Human Evaluations

- **Evaluation Scope:** Llama 3 was assessed across 12 key use cases, including coding, reasoning, rewriting, and summarization, etc.
- **Comparison Models:** Evaluations compared Llama 3's performance with models like GPT-4, Claude Sonnet, and Mistral Medium.
- **Performance Metrics:** Model outputs were ranked according to quality, coherence, and relevance by human annotators, who provided preference rankings.
- **Results:** The Llama 3 70B instruction-following model showed strong performance, comparable or outperformed peer models in real world settings in a lot of cases.



Human Evaluations

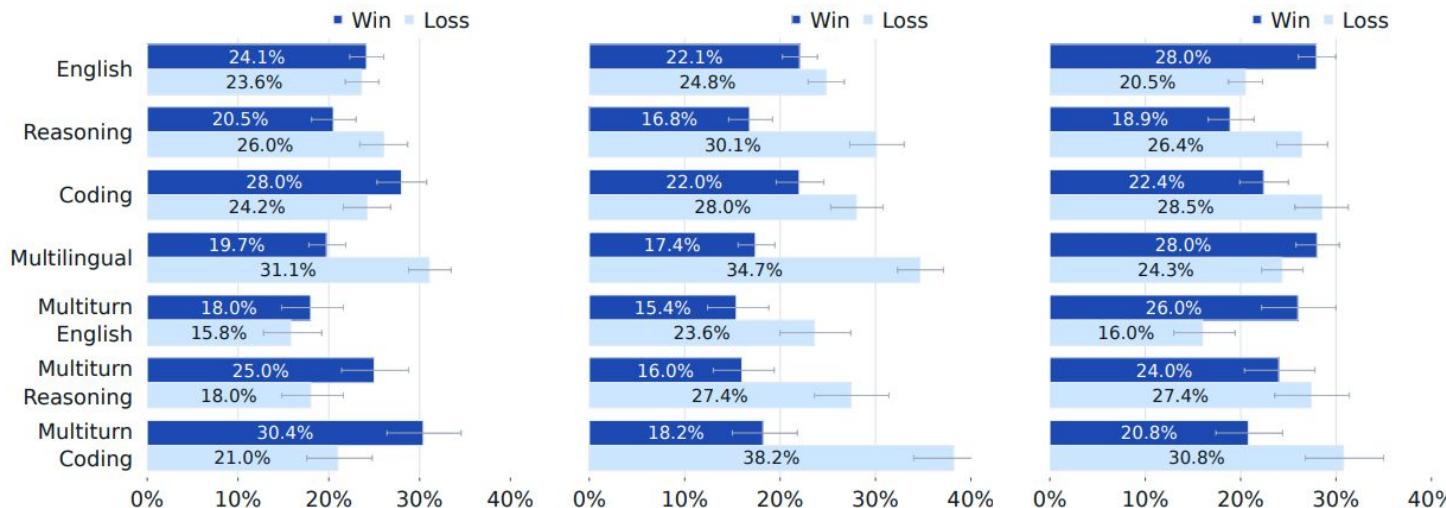


Figure 17 Human evaluation results for the Llama 3 405B model. *Left:* Comparison with GPT-4. *Middle:* Comparison with GPT-4o. *Right:* Comparison with Claude 3.5 Sonnet. All results include 95% confidence intervals and exclude ties.



Safety

- **Content Filtering:** Filters block harmful content and clean pre-training data of PII and unsafe domains.
- **Human Alignment:** Fine-tunes outputs with human feedback to avoid unsafe behavior.
- **Testing:** Stress tests ensure harmless and helpful responses.
- **Transparency:** Open release invites community feedback to improve safety.

6. Inference

Optimizing Inference Efficiency in the Llama 3 405B Model:
Key Techniques Explored



Pipeline Parallelism

Objective: Solve the memory constraints of the 405B model, which exceeds the capacity of a single machine with 8 Nvidia H100 GPUs.

Implementation:

- Parallelization done across 16 GPUs on 2 machines using BF16 precision.
- **Within-node:** Applies high-bandwidth NVLink for tensor parallelism
- **Across nodes:** Relies on pipeline parallelism for managing the lower bandwidth and higher latency network connections.



Pipeline Parallelism

Metric	Jailbreaks	Injections	Out-of-Distribution Jailbreaks	Multilingual Jailbreaks	Indirect Injections
TPR	99.9%	99.5%	97.5%	91.5%	71.4%
FPR	0.4%	0.8%	3.9%	5.3%	1.0%
AUC	0.997	1.000	0.975	0.959	0.996

Table 28 Performance of Prompt Guard. We include in- and out-of-distribution evaluations, a multilingual jailbreak built using machine translation, and a dataset of indirect injections from CyberSecEval.

Pipeline Parallelism

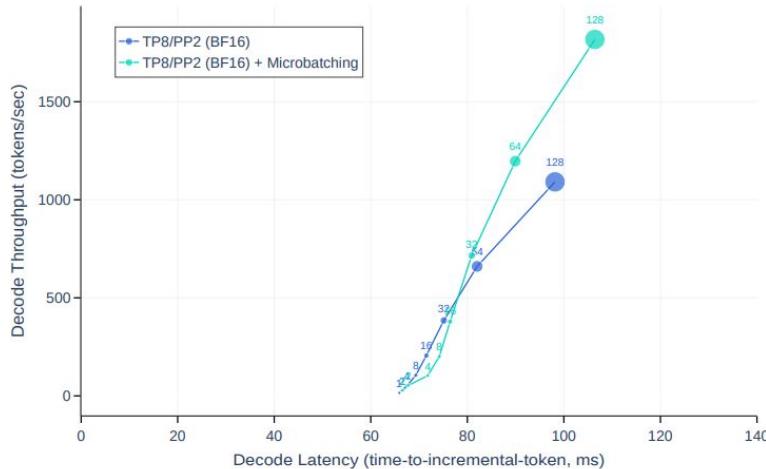
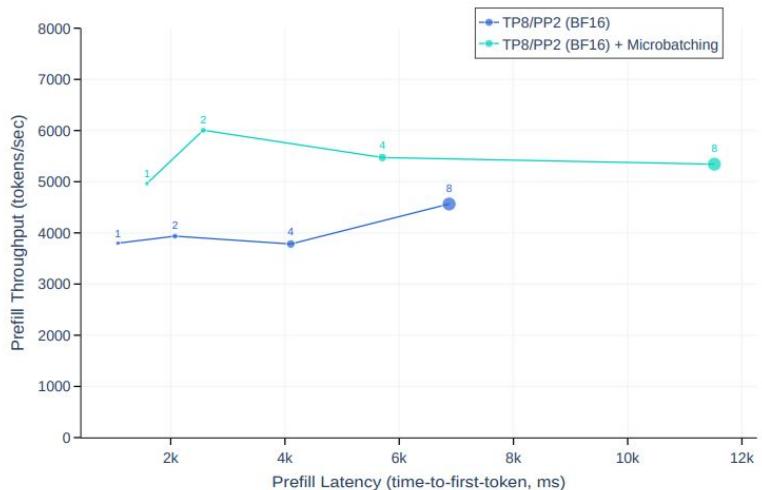


Figure 24 Effect of micro-batching on inference throughput and latency during the *Left:* pre-filling and *Right:* decoding stage. The numbers in the plot correspond to the (micro-)batch size.



Pipeline Parallelism

Micro-batching

- **Enables** concurrent execution of smaller batches.
- Significantly enhances throughput during both the key-value cache pre-fill and decoding stages.
- Results in a better throughput-latency trade-off despite slight increases in latency.



FP8 Quantization

- Exploits the H100 GPUs' FP8 support for low-precision inference.
- Quantization applied selectively:
 - Feedforward network layers: Quantized for efficiency.
 - Self-attention layers: Excluded to preserve output quality.
- Dynamic scaling factors are optimized to maintain accuracy and minimize underflows.

Benefits:

- Up to 50% improvement in throughput during the pre-fill stage.
- Enhanced throughput-latency trade-offs during decoding.



FP8 Quantization

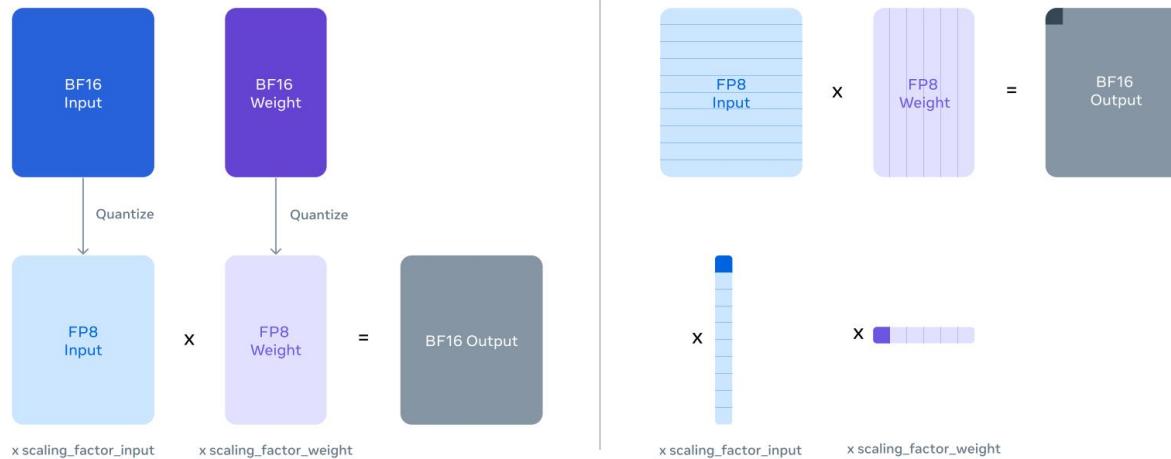


Figure 25 Illustration of tensor-wise and row-wise FP8 quantization. Right: Row-wise quantization enables the use of more granular activation factors than Left: tensor-wise quantization.



FP8 Quantization

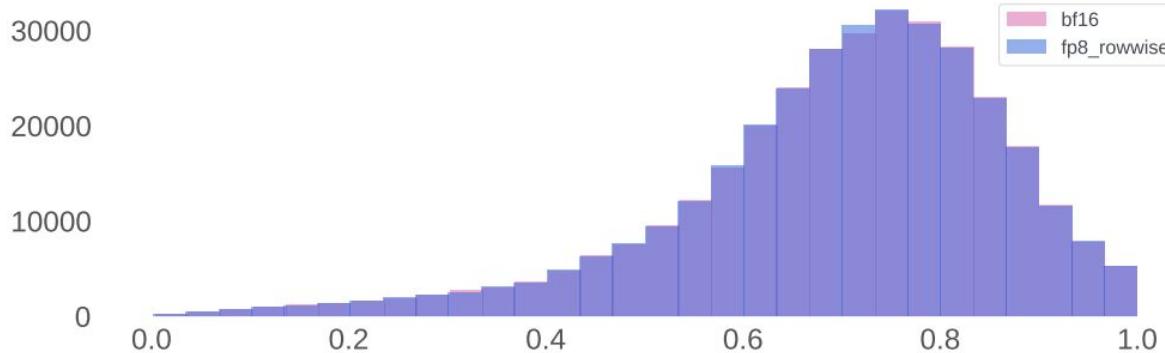


Figure 26 Reward score distribution for Llama 3 405B using BF16 and FP8 inference. Our FP8 quantization approach has negligible impact on the model's responses.



FP8 Quantization

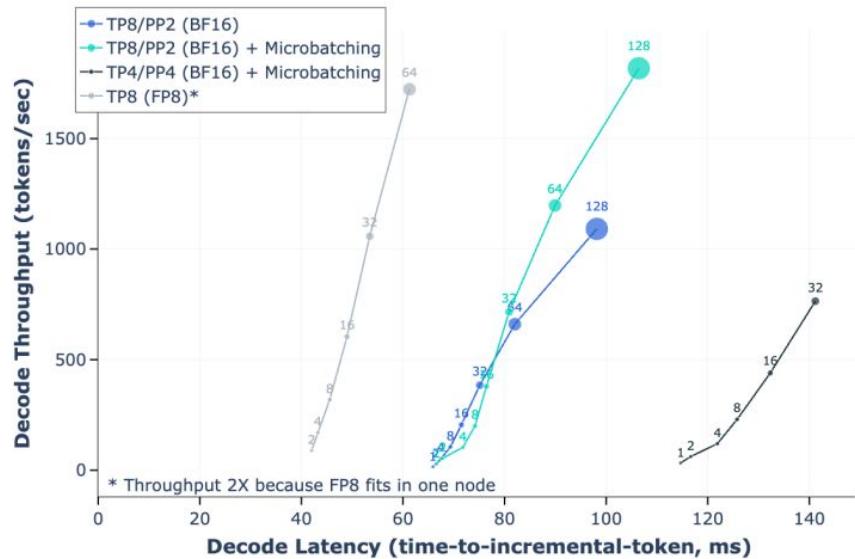
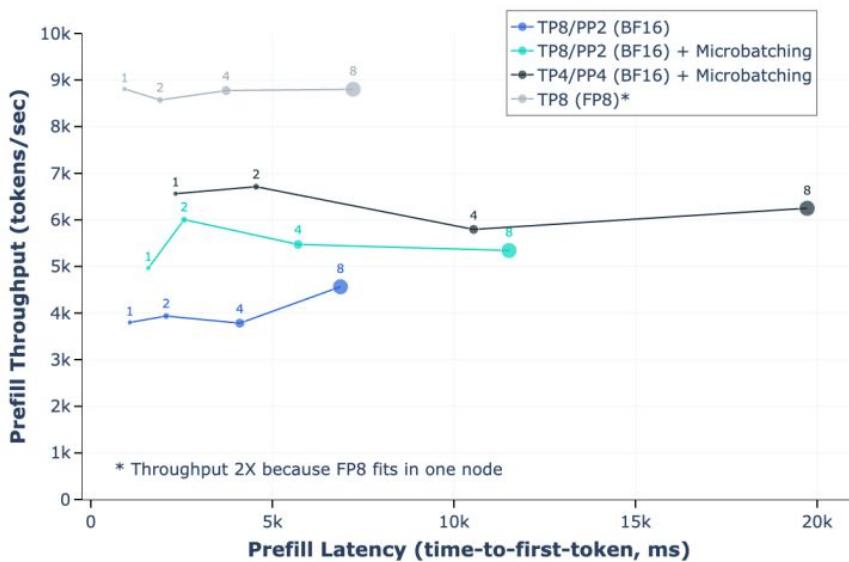


Figure 27 Throughput-latency trade-off in FP8 inference with Llama 3 405B compared with BF16 inference using different pipeline parallelization setups. Left: Results for pre-filling. Right: Results for decoding.

7. Vision Experiments

Quick Insight into Vision Multimodality



LLaMa 3 on vision - Multimodality

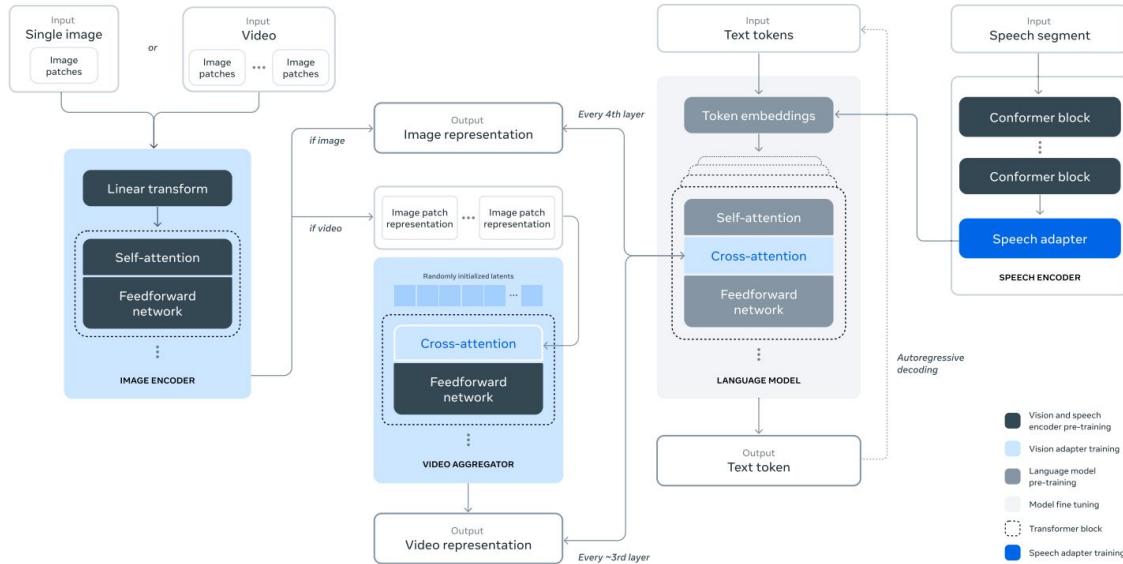


Figure 28 Illustration of the compositional approach to adding multimodal capabilities to LLaMa 3 that we study in this paper. This approach leads to a multimodal model that is trained in five stages: **(1)** language model pre-training, **(2)** multi-modal encoder pre-training, **(3)** vision adapter training, **(4)** model finetuning, and **(5)** speech adapter training.



LLaMa 3 on vision - Multimodality

Data

- Trains on extensive image-text and video-text datasets.
- Separate cross-attention layers enable learning from visual and temporal information without altering text-only tasks.

Advantages of Compositional Approach

- Parallelized development of vision and language models.
- Avoids complexities of joint training for multimodal data.
- Ensures efficiency in inference by not passing full-resolution images through the LLM backbone.
- Preserves performance on text-only tasks.

The experiments show promising multimodal capabilities, but the models are still under development and not yet ready for release.

8. Speech Experiments

On developing models capable of real-time, high-quality speech recognition and synthesis.



LLaMa 3 on speech - Multimodality

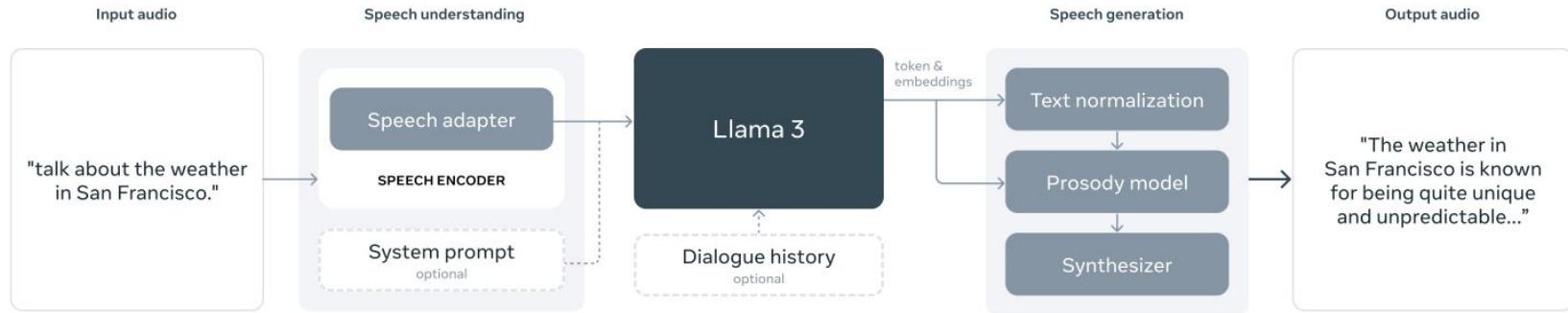


Figure 29 Architecture of our speech interface for Llama 3.



LLaMa 3 on speech - Multimodality

- **Speech Adapter Integration:** Adapts speech inputs into token representations compatible with the language model, enabling seamless speech-to-text and text-to-speech functionalities.
- **Low-Latency Streaming:** Supports token-wise processing for real-time interaction without requiring full sequence input.
- **Prosody Modeling:** Enhances naturalness and expressiveness in synthesized speech by modeling tone, rhythm, and emotional depth.

9. Related Work





Related Work

Key highlight inspirations:

- Adheres to scaling laws for efficient resource utilization.
- Smaller models, despite fewer parameters, are optimized for inference costs.
- Explores techniques such as dense Transformer architecture, grouped query attention, and positional embeddings for scaling efficiency.

Conclusion





Conclusion

- **Strong Focus on High-Quality Data:** Scale, simplicity, high quality data all consistently produced better results than complex architectures or training recipes.
- **Trustworthy Evaluations:** To ensure objectivity and data were not contaminated, human evaluations were limited to a small independent group of researchers, and data were separately handled.
- **Transparency and Sharing:** The team shared detailed development processes and preliminary multimodal experiments to aid research and public understanding.
- **Open Research Goals:** Fosters industry-wide collaboration and the ethical advancement of AGI through transparency and accessibility.

Thanks and Gig'Em!



Paper Review

Tree of Thoughts: Deliberate Problem Solving with Large Language Models



Omar Khater
Matthew DeLorenzo

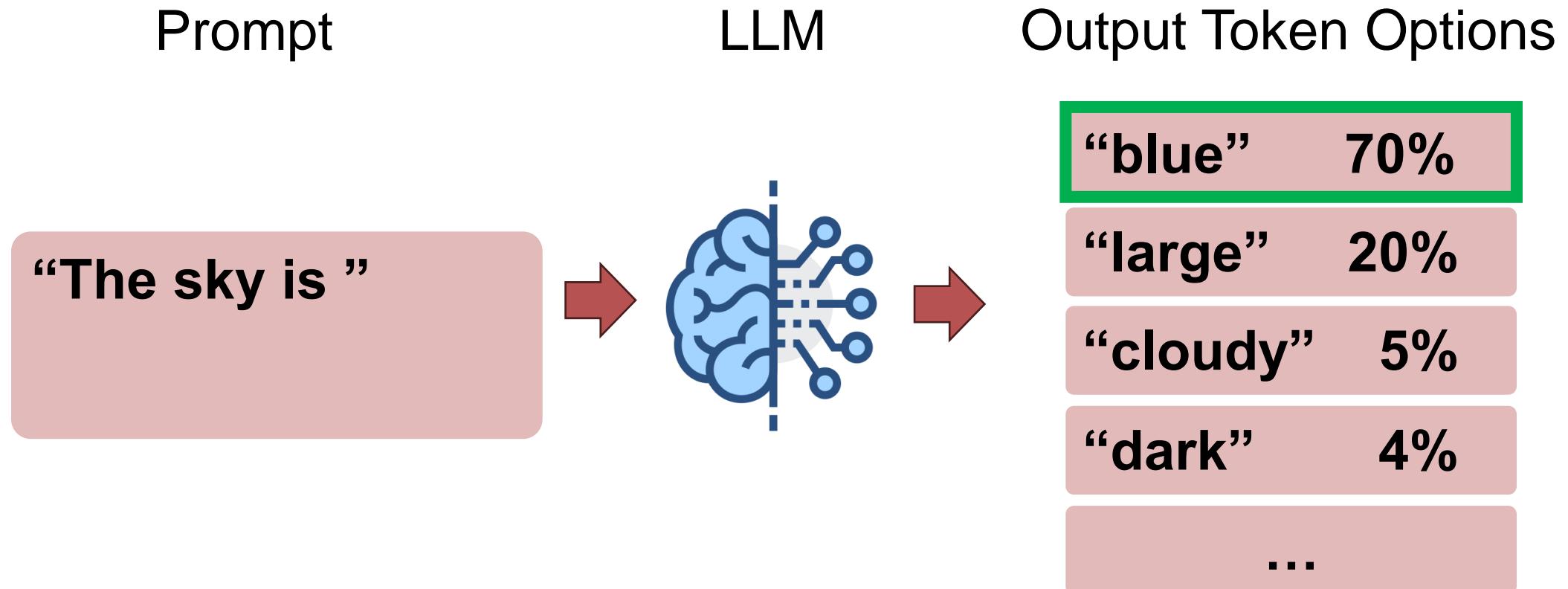
Agenda

- **Background & Motivation** (Omar)
- **Methodology** (Matthew)
- **Experiments** (Omar)
- **Results** (Omar)
- **Conclusions** (Matthew)

Background & Motivation

LLM Token Generation Process

- ❖ Auto-regressive process
 - Linear (left to right)
 - Token-by-token



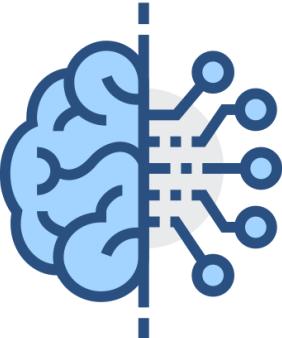
LLM Token Generation Process

- ❖ Auto-regressive process
 - Linear (left to right)
 - Token-by-token

Prompt

“The sky is blue”

LLM



Output Token Options

“.” 70%

“and” 20%

“,” 5%

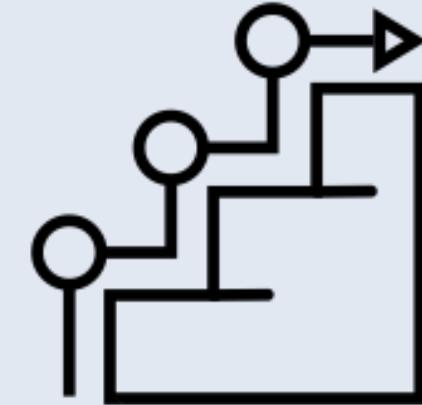
“or” 4%

...

Motivation

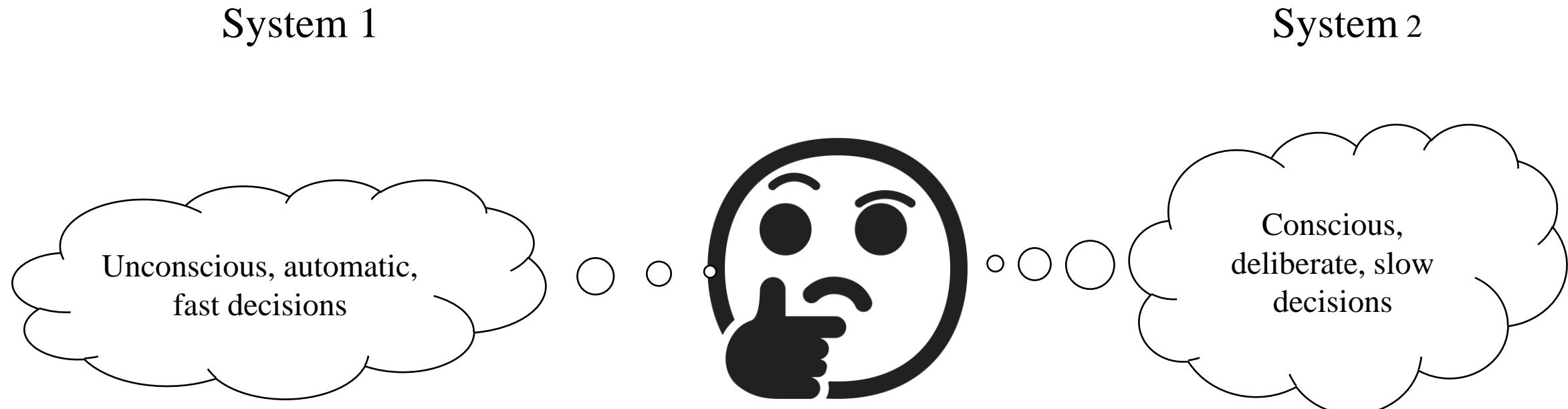
LLMs struggle on

- Extended planning.
- Lookahead/backtracking.
- Evaluation of multiple paths.



Question – Can the model solve problems better?

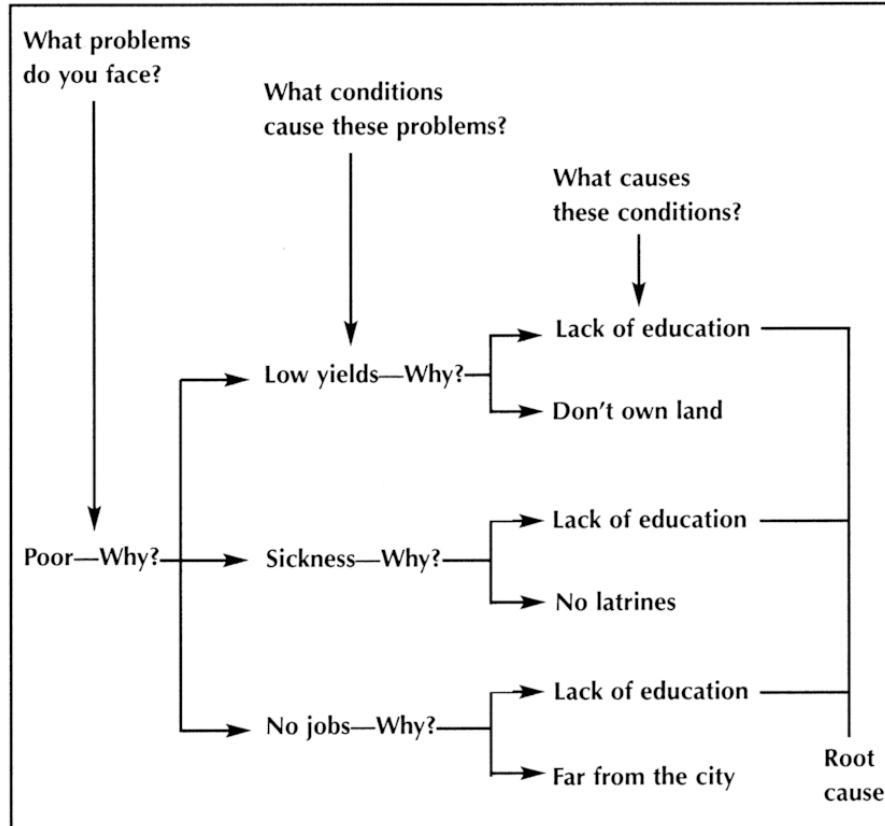
Dual – Modes Thinking Process



- Token-level choices of LMs is like System 1
- Can LMs have planning process like System 2?

AI Planning

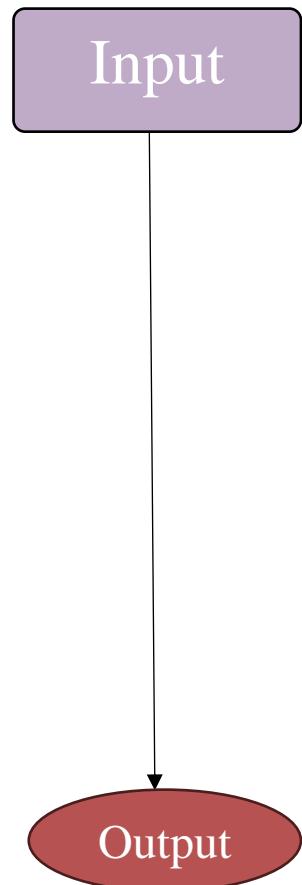
Problem solving is characterized by a search through a combinatorial problem space, represented as a tree.



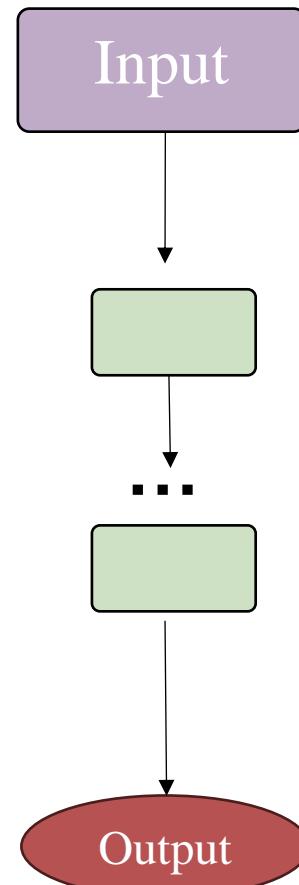
1. A. Newell, J. C. Shaw, and H. A. Simon. Report on a general problem solving program. In IFIP congress, volume 256, page 64. Pittsburgh, PA, 1959
2. Schwoebel, Mary. (2001). TOWARD THE THEORY AND PRACTICE OF APPRECIATIVE INQUIRY IN COMPLEX PEACEBUILDING AND DEVELOPMENT CONTEXTS.

General Problem Solving with Language Models

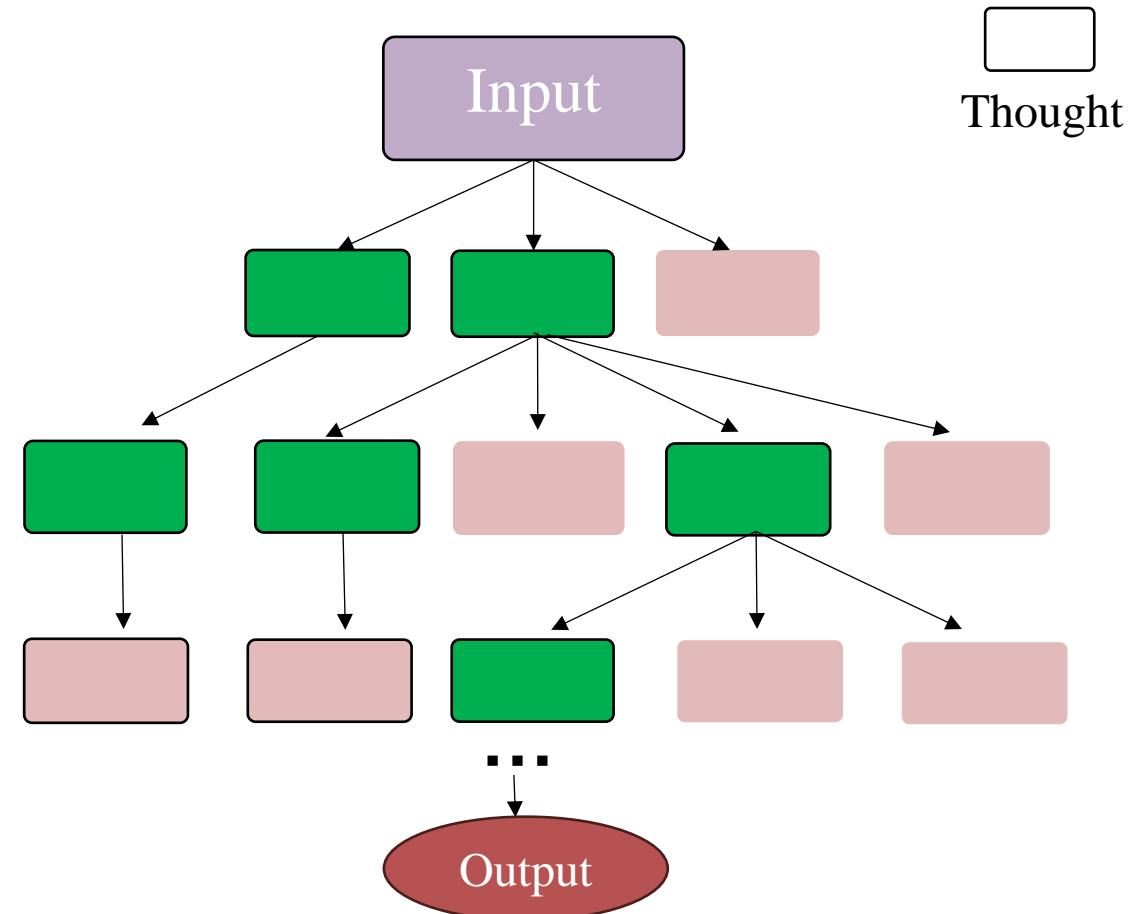
I/O Prompting



Chain of Thoughts (CoT) Prompting



Tree of Thoughts (ToT) Prompting

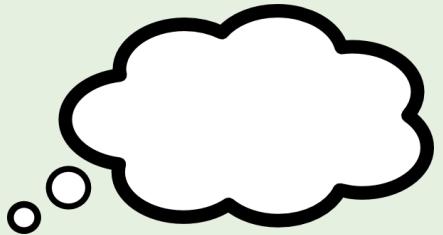


Methodology

4 Steps Tree-based Framework

4 Steps Framework

Thought
Decomposition



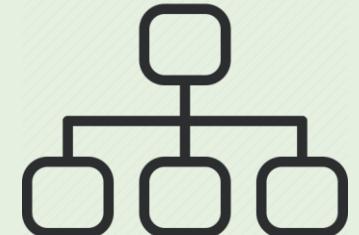
Thought
Generation



State
Evaluation



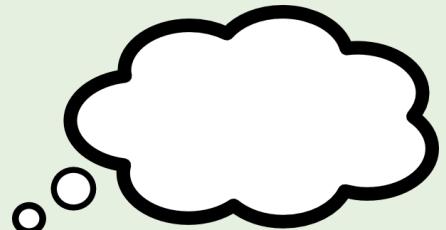
Search
Algorithm



Tree-of-Thoughts - Implementation

Framework

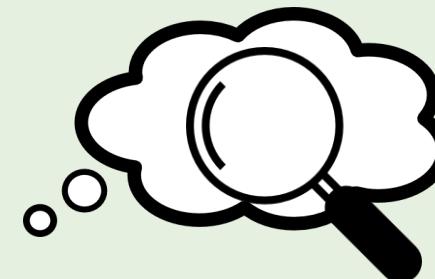
1. Thought Decomposition



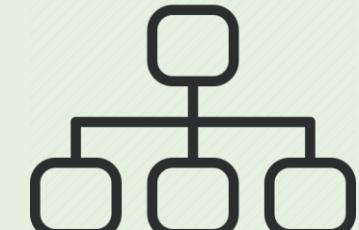
2. Thought Generation



3. State Evaluation



4. Search Algorithm



Tree-of-Thoughts - Implementation



Step 1. Thought Decomposition

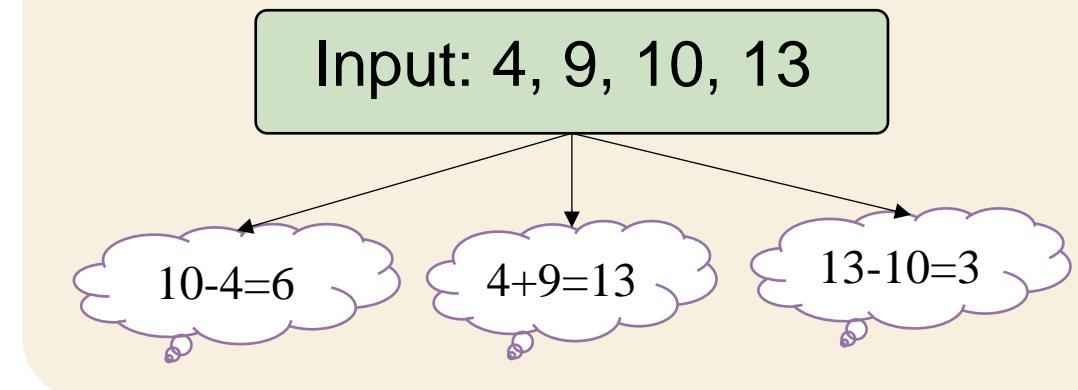
- Thought – grouping of tokens.
 - Specific to task.
 - Large enough to evaluate.
 - Small enough to generate.

Ex 1: Arithmetic Reasoning Task: Game of 24

- Given 4 numbers.
- Basic operations (+,-,* , /).
- Result in 24.

Arithmetic Reasoning - Thoughts

- Thought = equation line.



Tree-of-Thoughts - Implementation



Step 1. Thought Decomposition

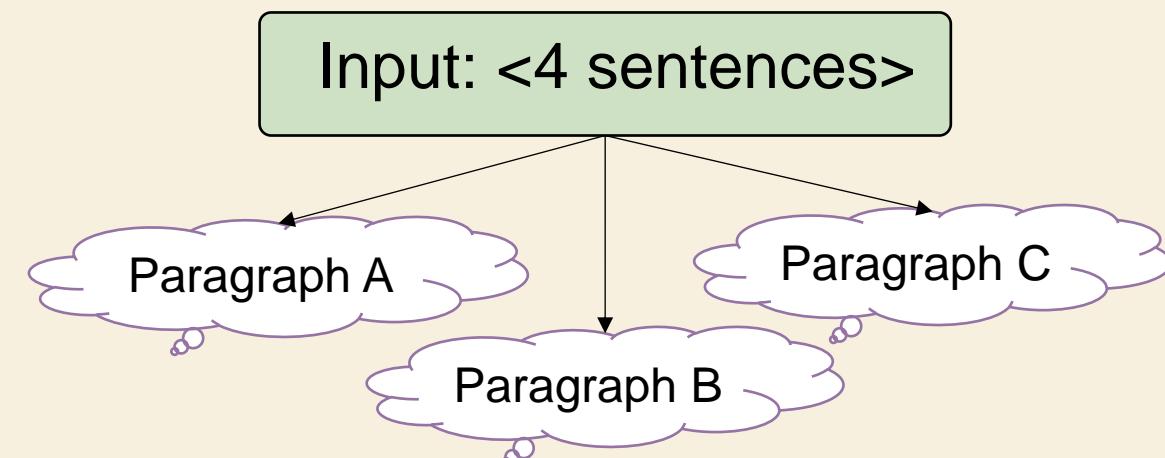
- Thought – grouping of tokens.
 - Specific to task.
 - Large enough to evaluate.
 - Small enough to generate.

Ex 2: Creative Writing Task

- Given 4 sentences.
- Write 4 paragraphs.
- Each paragraph ends with a provided sentence.

Creative Writing - Thoughts

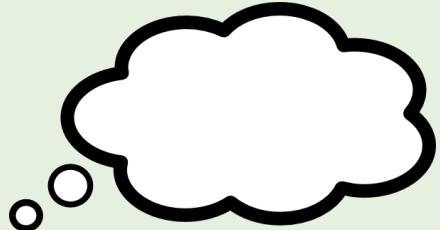
- Thought = paragraph.



Tree-of-Thoughts - Implementation

Framework

1. Thought Decomposition



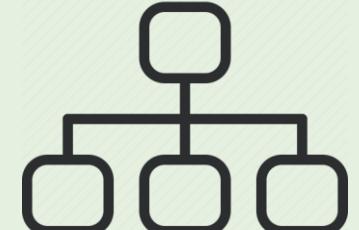
2. Thought Generation



3. State Evaluation



4. Search Algorithm



Tree-of-Thoughts - Implementation



Step 2. Thought Generation

- Sampling method to generate (k) thoughts.

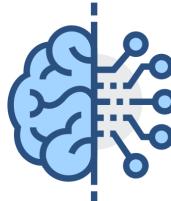
Strategy 1

Chain-of-Thoughts Prompting

- Independent prompts.
- Ideal for diverse samples.
- Creative Writing.

Creative Writing Example

"You are given 4 sentences <>. Write a plan, then write 4 paragraphs, with a sentence at the end of each paragraph."



<Paragraph>

Repeat process, each with **new context**.

Tree-of-Thoughts - Implementation



Step 2. Thought Generation

- Sampling method to generate (k) thoughts.

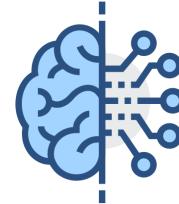
Strategy 2

“Propose” Prompting

- Sequential prompts.
- Improves deduplication.
- Tasks: Arithmetic, word-puzzle games.

Arithmetic Reasoning Ex:

Prompt: “Given these 4 numbers, propose possible next steps:”



Response: $2+4=6$

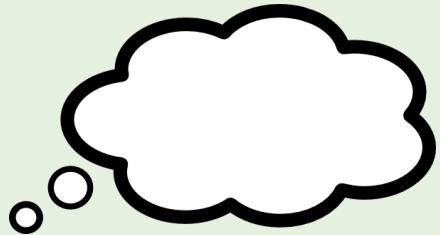


Repeat process k times **in same conversation.**

Tree-of-Thoughts - Implementation

Framework

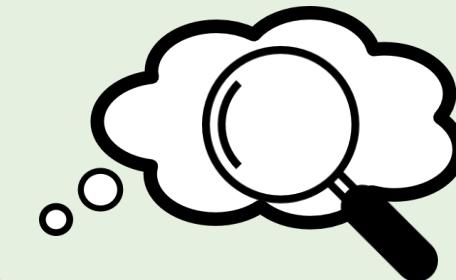
1. Thought Decomposition



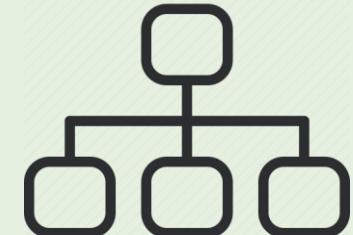
2. Thought Generation



3. State Evaluation



4. Search Algorithm



Tree-of-Thoughts - Implementation



Step 3. State Evaluation

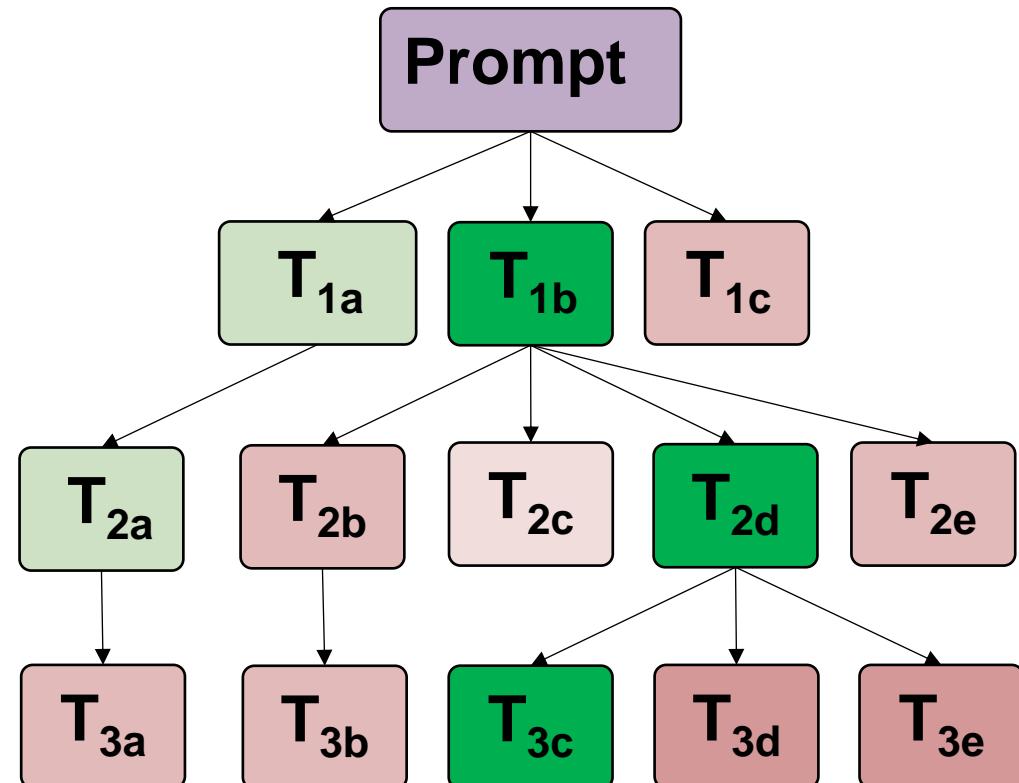
- State = prompt + set of thoughts.
- Heuristic to estimate optimal paths.

Strategy 1

Score states independently.

- LLM as evaluator.
- Evaluate on a scale of:
 - 1-10.
 - Sure/likely/impossible.

Arithmetic Reasoning Example





Tree-of-Thoughts - Implementation

Step 3. State Evaluation

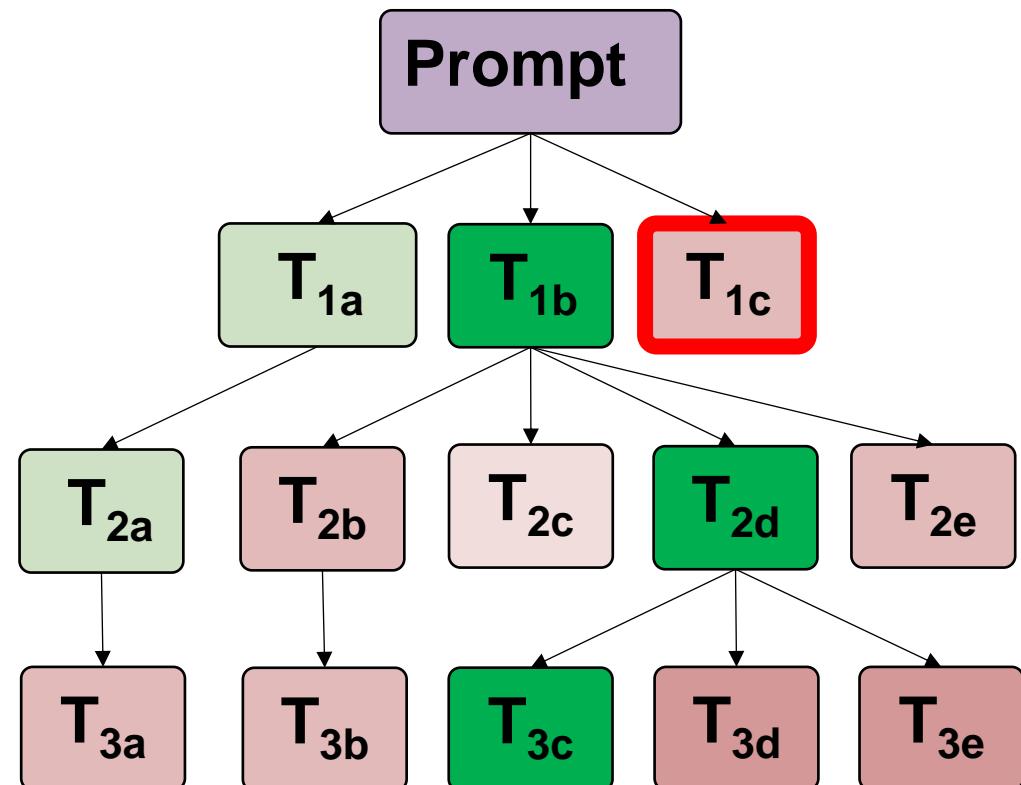
- State = prompt + set of thoughts.
- Heuristic to estimate optimal paths.

Strategy 1

Score states independently.

- LLM as evaluator.
- Evaluate on a scale of:
 - 1-10.
 - Sure/likely/impossible.

Arithmetic Reasoning Example





Tree-of-Thoughts - Implementation

Step 3. State Evaluation

- State = prompt + set of thoughts.
- Heuristic to estimate optimal paths.

Strategy 1

Score states independently.

- LLM as evaluator.
- Evaluate on a scale of:
 - 1-10.
 - Sure/likely/impossible.

Arithmetic Reasoning Example

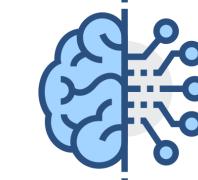
State

T_{1c}

Value
Prompt

LLM

Evaluation:
Sure/likely/impossible



Tree-of-Thoughts - Implementation



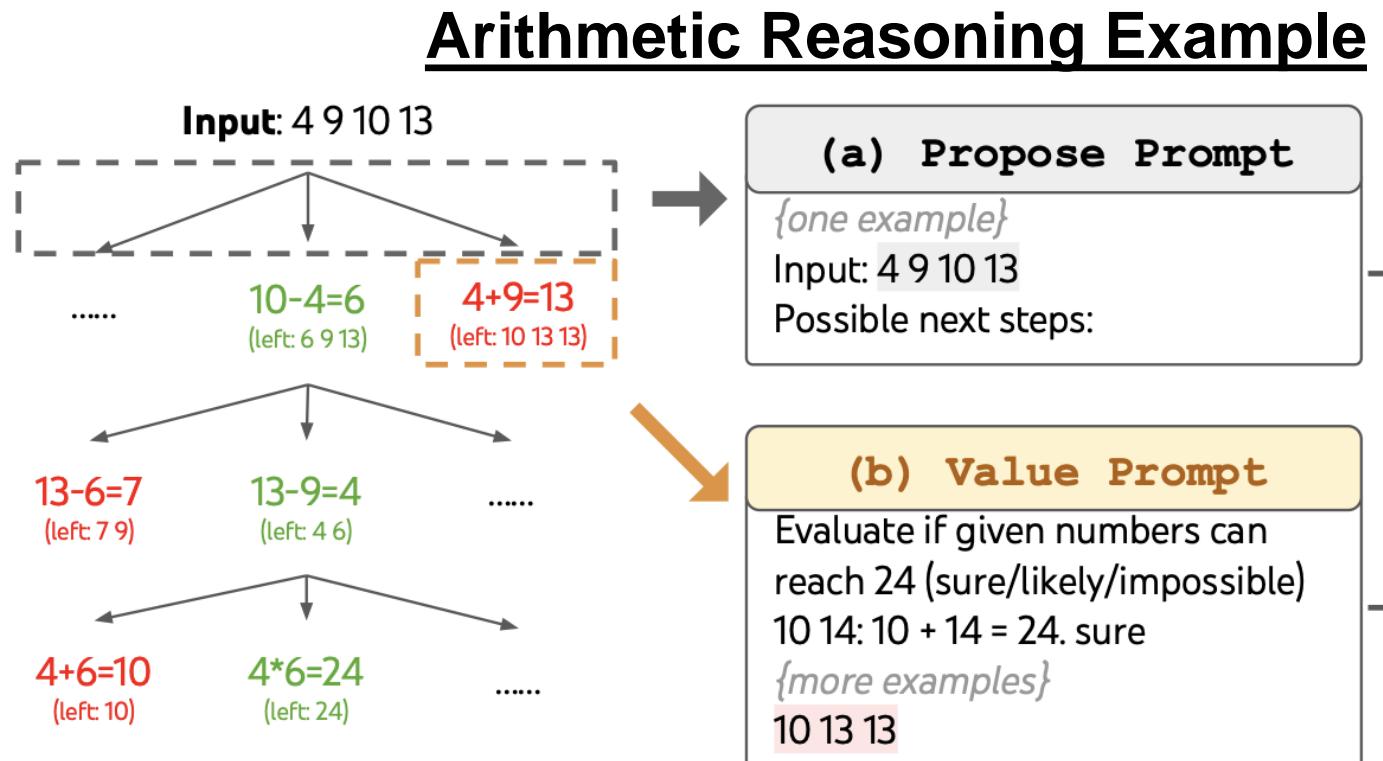
Step 3. State Evaluation

- State = prompt + set of thoughts.
- Heuristic to estimate optimal paths.

Strategy 1

Score states independently.

- LLM as evaluator.
- Evaluate on a scale of:
 - 1-10.
 - Sure/likely/impossible.



Tree-of-Thoughts - Implementation



Step 3. State Evaluation

- State = prompt + set of thoughts.
- Heuristic to estimate optimal paths.

Strategy 1

Score states independently.

- LLM as evaluator.
- Evaluate on a scale of:
 - 1-10.
 - Sure/likely/impossible.



Arithmetic Reasoning Example

Thought Evaluation

$$(13 - 10) * 13 = 3 * 13 = 39$$

- $10 + 13 + 13 = 36$ There is no way to obtain 24 with these big numbers. impossible

Tree-of-Thoughts - Implementation



Step 3. State Evaluation

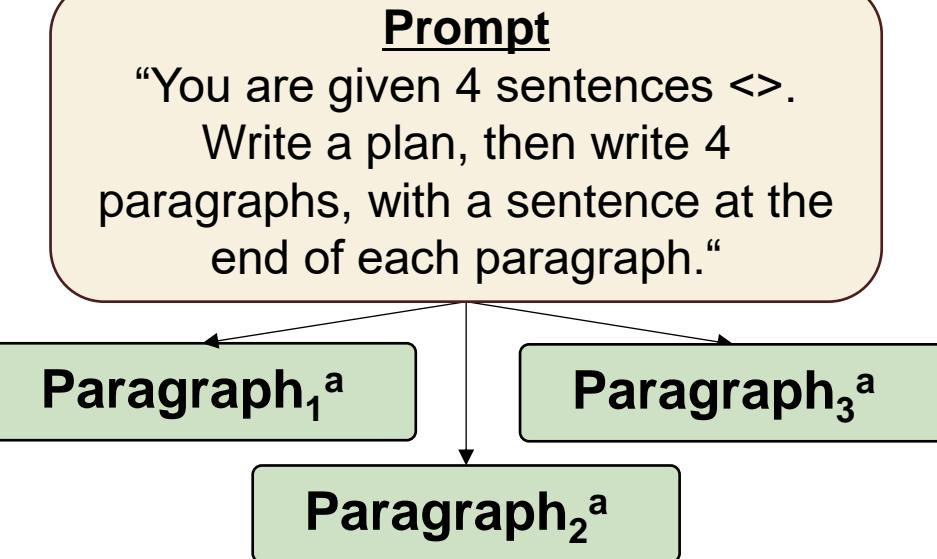
- State = prompt + set of thoughts.
- Heuristic to estimate optimal paths.

Strategy 2

Vote on best state.

- **LLM as comparator.**
- All thoughts are compared.
- Most votes - selected.

Arithmetic Reasoning Example



Tree-of-Thoughts - Implementation



Step 3. State Evaluation

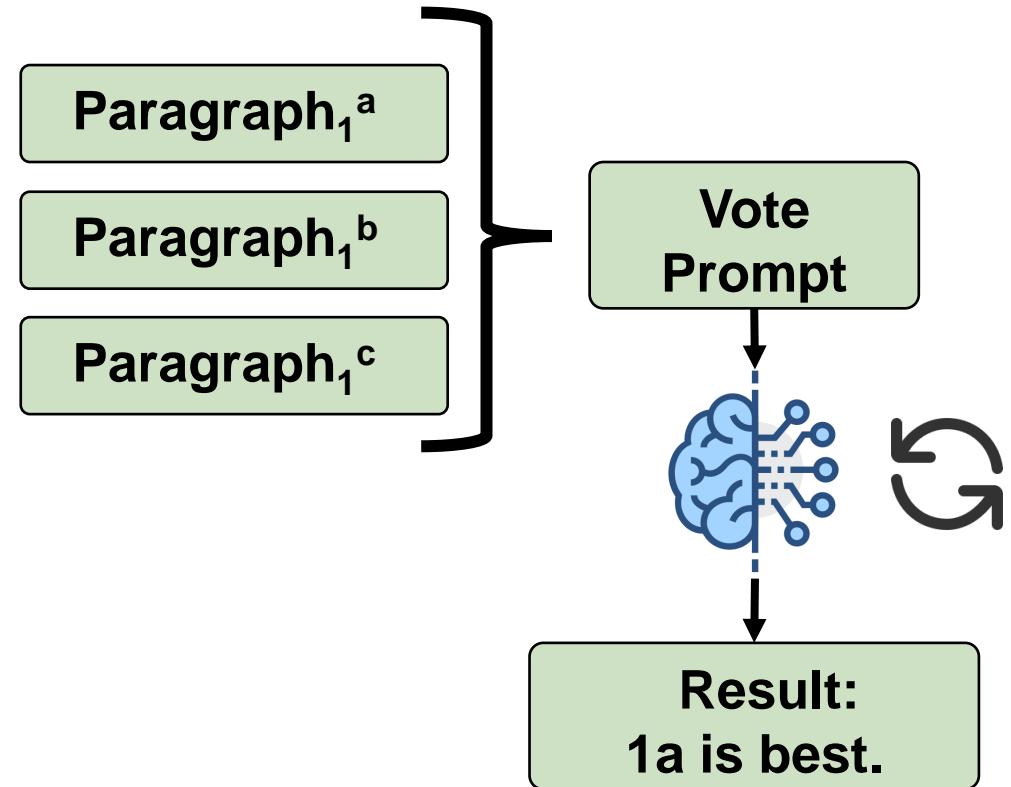
- State = prompt + set of thoughts.
- Heuristic to estimate optimal paths.

Strategy 2

Vote on best state.

- **LLM as comparator.**
- All thoughts are compared.
- Most votes - selected.

Creative Writing Example



Tree-of-Thoughts - Implementation



Step 3. State Evaluation

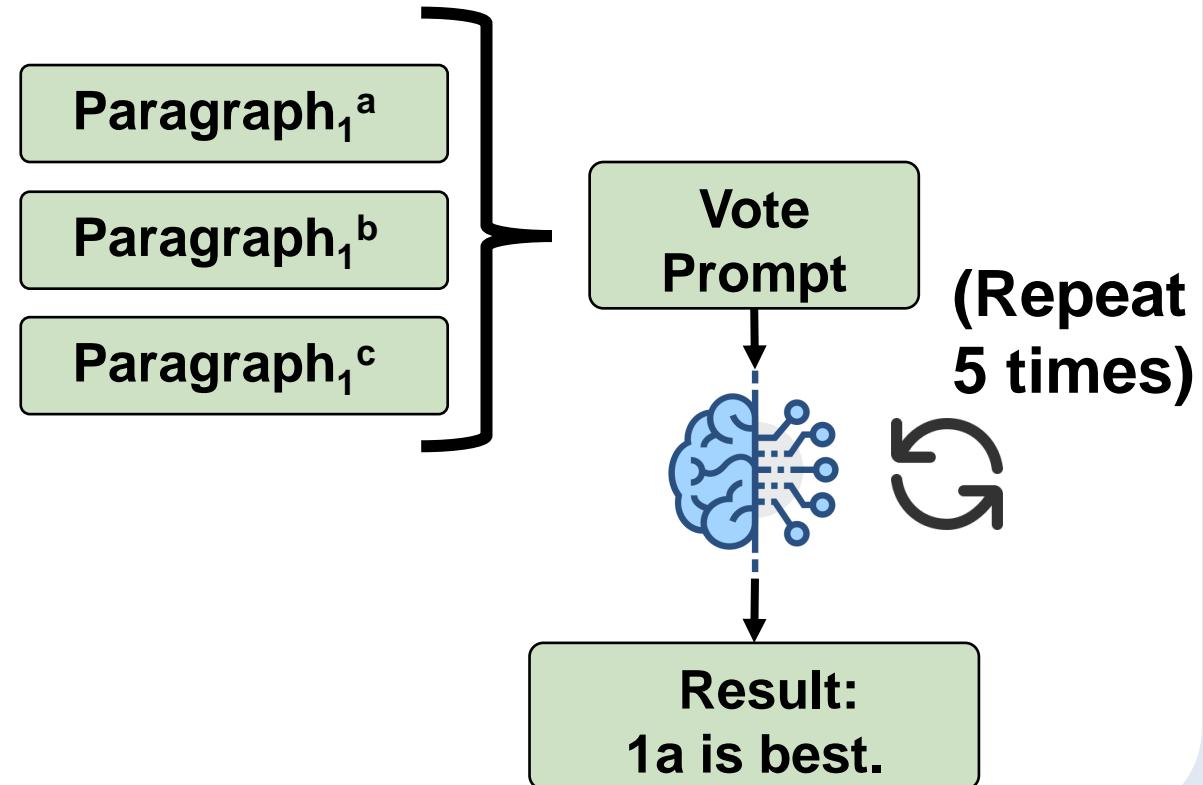
- State = prompt + set of thoughts.
- Heuristic to estimate optimal paths.

Strategy 2

Vote on best state.

- **LLM as comparator.**
- All thoughts are compared.
- Most votes - selected.

Creative Writing Example





Tree-of-Thoughts - Implementation

Step 3. State Evaluation

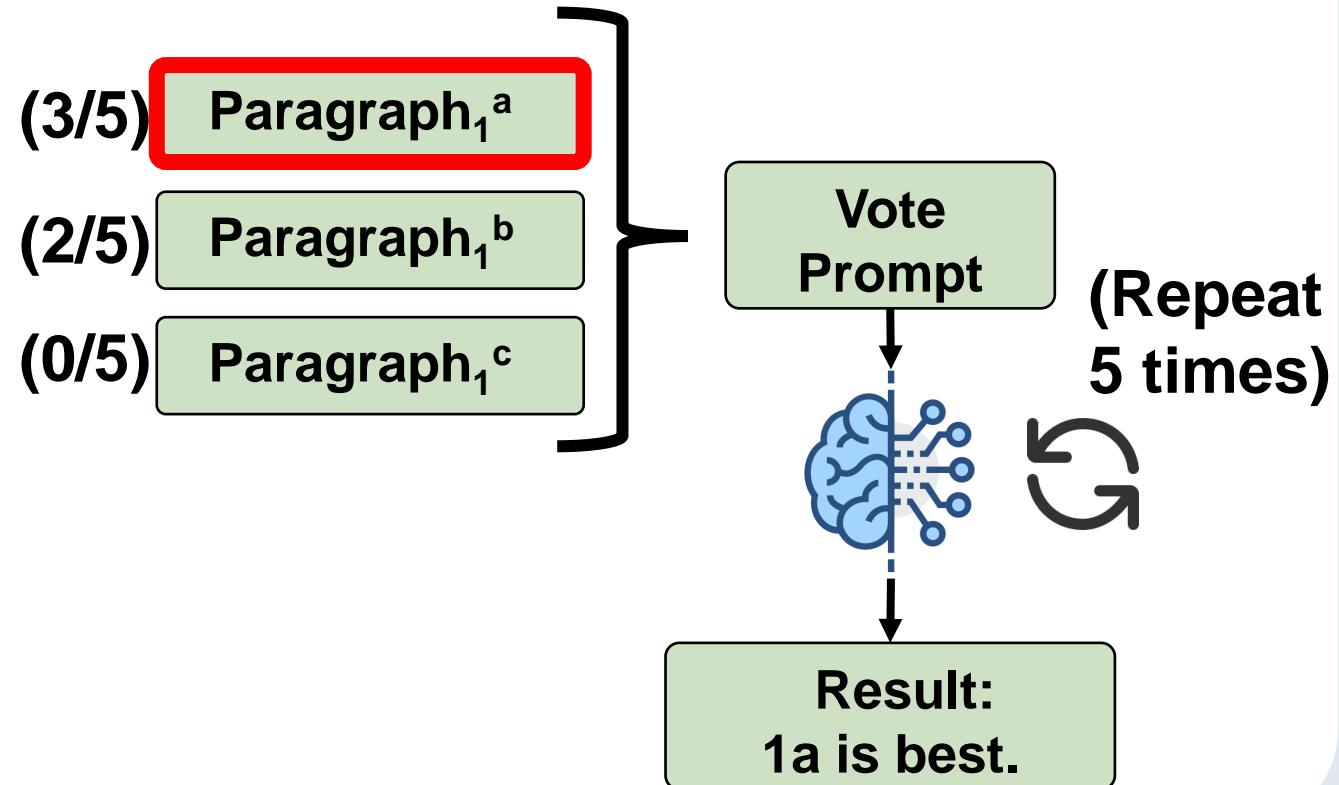
- State = prompt + set of thoughts.
- Heuristic to estimate optimal paths.

Strategy 2

Vote on best state.

- **LLM as comparator.**
- All thoughts are compared.
- Most votes - selected.

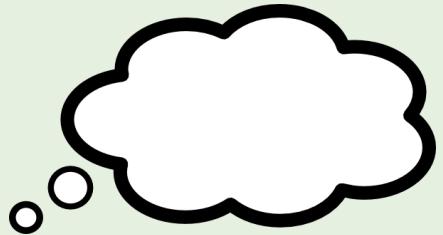
Creative Writing Example



Tree-of-Thoughts - Implementation

Framework

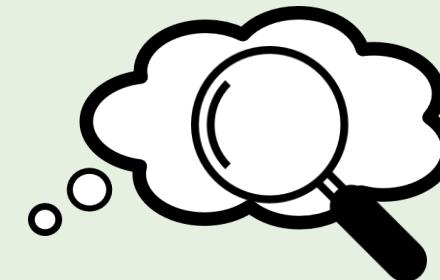
1. Thought Decomposition



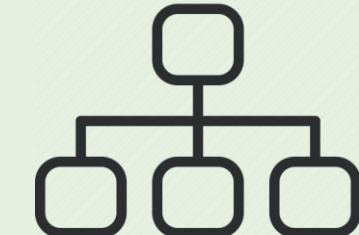
2. Thought Generation



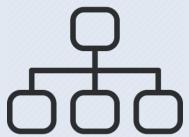
3. State Evaluation



4. Search Algorithm



Tree-of-Thoughts - Implementation



Step 4. Search Algorithm

- Method to generate/traverse tree.

Strategy 1 – Breadth-First Search

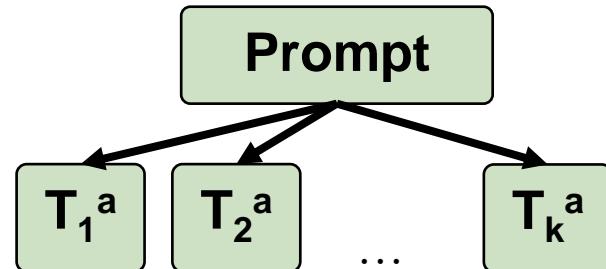
- Maintain best b states at each step.
- Game of 24, Creative Writing.

Algorithm 1 ToT-BFS($x, p_\theta, G, k, V, T, b$)

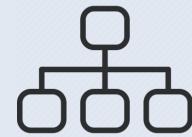
Require: Input x , LM p_θ , thought generator $G()$]
& size limit k , states evaluator $V()$, step limit T ,
breadth limit b .

```
 $S_0 \leftarrow \{x\}$ 
for  $t = 1, \dots, T$  do
     $S'_t \leftarrow \{[s, z] \mid s \in S_{t-1}, z_t \in G(p_\theta, s, k)\}$  ← Generate  $k$  (100) thoughts
     $V_t \leftarrow V(p_\theta, S'_t)$ 
     $S_t \leftarrow \arg \max_{S \subset S'_t, |S|=b} \sum_{s \in S} V_t(s)$ 
end for
return  $G(p_\theta, \arg \max_{s \in S_T} V_T(s), 1)$ 
```

Arithmetic Reasoning Example



Tree-of-Thoughts - Implementation



Step 4. Search Algorithm

- Method to generate/traverse tree.

Strategy 1 – Breadth-First Search

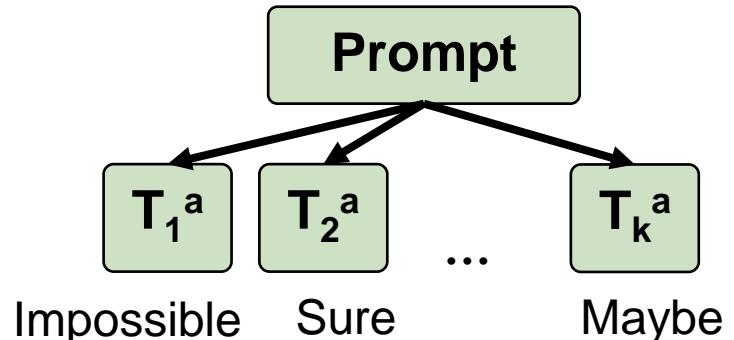
- Maintain best b states at each step.
- Game of 24, Creative Writing.

Algorithm 1 ToT-BFS($x, p_\theta, G, k, V, T, b$)

Require: Input x , LM p_θ , thought generator $G()$ & size limit k , states evaluator $V()$, step limit T , breadth limit b .

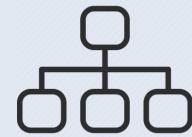
```
 $S_0 \leftarrow \{x\}$ 
for  $t = 1, \dots, T$  do
     $S'_t \leftarrow \{[s, z] \mid s \in S_{t-1}, z_t \in G(p_\theta, s, k)\}$ 
     $V_t \leftarrow V(p_\theta, S'_t)$ 
     $S_t \leftarrow \arg \max_{S \subset S'_t, |S|=b} \sum_{s \in S} V_t(s)$ 
end for
return  $G(p_\theta, \arg \max_{s \in S_T} V_T(s), 1)$ 
```

Arithmetic Reasoning Example



← Evaluate each state.

Tree-of-Thoughts - Implementation



Step 4. Search Algorithm

- Method to generate/traverse tree.

Strategy 1 – Breadth-First Search

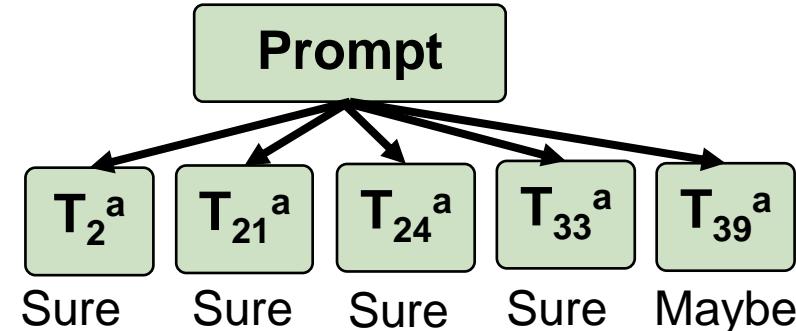
- Maintain best b states at each step.
- Game of 24, Creative Writing.

Algorithm 1 ToT-BFS($x, p_\theta, G, k, V, T, b$)

Require: Input x , LM p_θ , thought generator $G()$]
& size limit k , states evaluator $V()$, step limit T ,
breadth limit b .

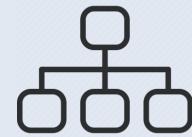
```
 $S_0 \leftarrow \{x\}$ 
for  $t = 1, \dots, T$  do
     $S'_t \leftarrow \{[s, z] \mid s \in S_{t-1}, z_t \in G(p_\theta, s, k)\}$ 
     $V_t \leftarrow V(p_\theta, S'_t)$ 
     $S_t \leftarrow \arg \max_{S \subset S'_t, |S|=b} \sum_{s \in S} V_t(s)$ 
end for
return  $G(p_\theta, \arg \max_{s \in S_T} V_T(s), 1)$ 
```

Arithmetic Reasoning Example



← Keep the best b (5) thoughts.

Tree-of-Thoughts - Implementation



Step 4. Search Algorithm

- Method to generate/traverse tree.

Strategy 1 – Breadth-First Search

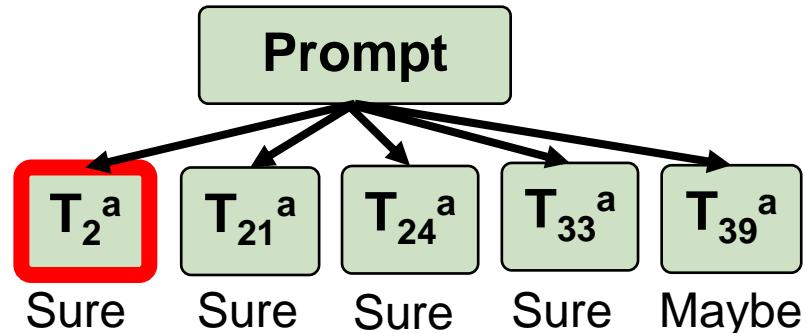
- Maintain best b states at each step
- Game of 24, Creative Writing

Algorithm 1 ToT-BFS($x, p_\theta, G, k, V, T, b$)

Require: Input x , LM p_θ , thought generator $G()$]
& size limit k , states evaluator $V()$, step limit T ,
breadth limit b .

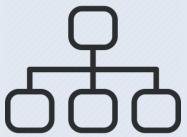
```
 $S_0 \leftarrow \{x\}$ 
for  $t = 1, \dots, T$  do
     $S'_t \leftarrow \{[s, z] \mid s \in S_{t-1}, z_t \in G(p_\theta, s, k)\}$ 
     $V_t \leftarrow V(p_\theta, S'_t)$ 
     $S_t \leftarrow \arg \max_{S \subset S'_t, |S|=b} \sum_{s \in S} V_t(s)$ 
end for
return  $G(p_\theta, \arg \max_{s \in S_T} V_T(s), 1)$ 
```

Arithmetic Reasoning Example



← Traverse to next best thought.

Tree-of-Thoughts - Implementation



Step 4. Search Algorithm

- Method to generate/traverse tree.

Strategy 1 – Breadth-First Search

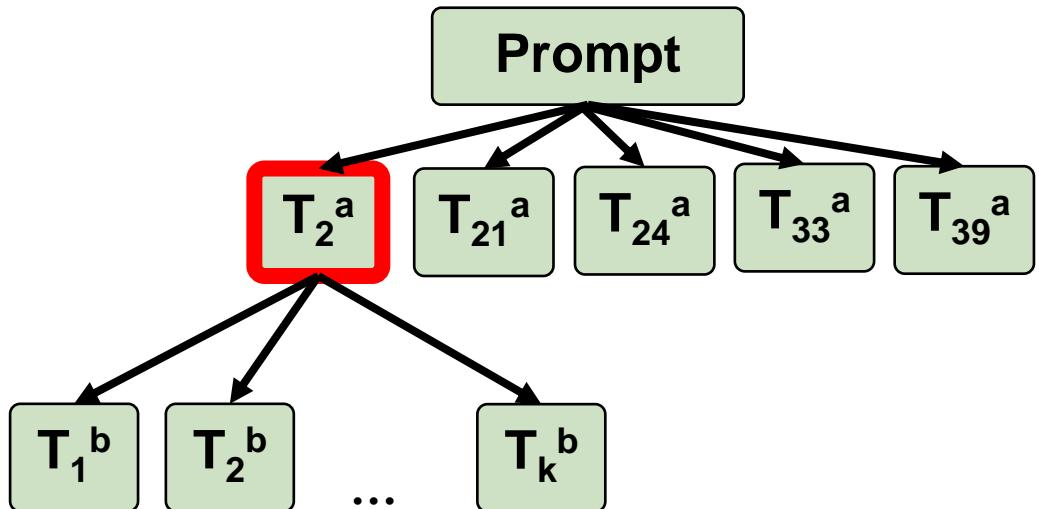
- Maintain best b states at each step
- Game of 24, Creative Writing

Algorithm 1 ToT-BFS($x, p_\theta, G, k, V, T, b$)

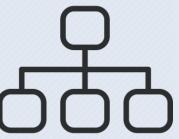
Require: Input x , LM p_θ , thought generator $G()$ & size limit k , states evaluator $V()$, step limit T , breadth limit b .

```
 $S_0 \leftarrow \{x\}$ 
for  $t = 1, \dots, T$  do
     $S'_t \leftarrow \{[s, z] \mid s \in S_{t-1}, z_t \in G(p_\theta, s, k)\}$ 
     $V_t \leftarrow V(p_\theta, S'_t)$ 
     $S_t \leftarrow \arg \max_{S \subset S'_t, |S|=b} \sum_{s \in S} V_t(s)$ 
end for
return  $G(p_\theta, \arg \max_{s \in S_T} V_T(s), 1)$ 
```

Arithmetic Reasoning Example



← Repeat until depth limit T (3).



Tree-of-Thoughts - Implementation

Step 4. Search Algorithm

- Method to generate/traverse tree.

Strategy 1 – Breadth-First Search

- Maintain best b states at each step.
- Game of 24, Creative Writing.

Algorithm 1 ToT-BFS($x, p_\theta, G, k, V, T, b$)

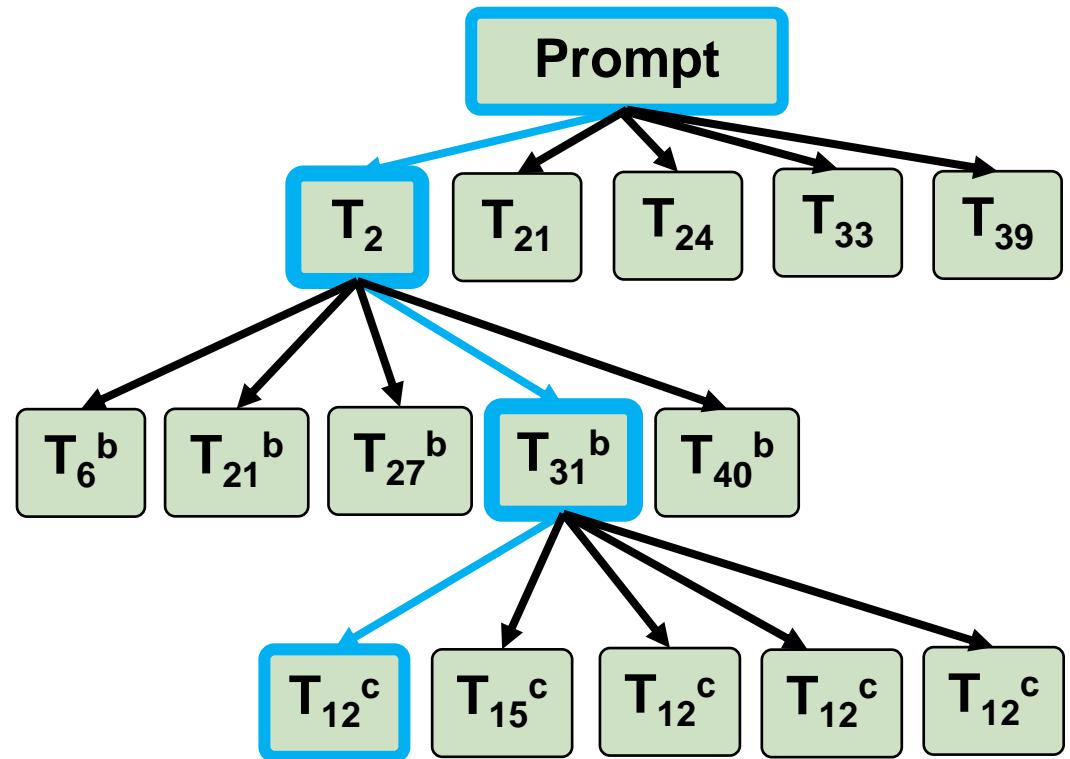
Require: Input x , LM p_θ , thought generator $G()$ & size limit k , states evaluator $V()$, step limit T , breadth limit b .

```

 $S_0 \leftarrow \{x\}$ 
for  $t = 1, \dots, T$  do
     $S'_t \leftarrow \{[s, z] \mid s \in S_{t-1}, z_t \in G(p_\theta, s, k)\}$ 
     $V_t \leftarrow V(p_\theta, S'_t)$ 
     $S_t \leftarrow \arg \max_{S \subset S'_t, |S|=b} \sum_{s \in S} V_t(s)$ 
end for
return  $G(p_\theta, \arg \max_{s \in S_T} V_T(s), 1)$ 

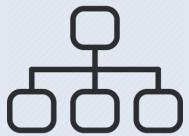
```

Arithmetic Reasoning Example



← Return optimal prompt.

Tree-of-Thoughts - Implementation



Step 4. Search Algorithm

- Method to generate/traverse tree.

Strategy 2 – Depth First Search

- Evaluate most promising strategy first.
- Crossword-puzzle

Algorithm 2 ToT-DFS($s, t, p_\theta, G, k, V, T, v_{th}$)

Require: Current state s , step t , LM p_θ , thought generator $G()$ and size limit k , states evaluator $V()$, step limit T , threshold v_{th}

if $t > T$ **then** record output $G(p_\theta, s, 1)$

end if

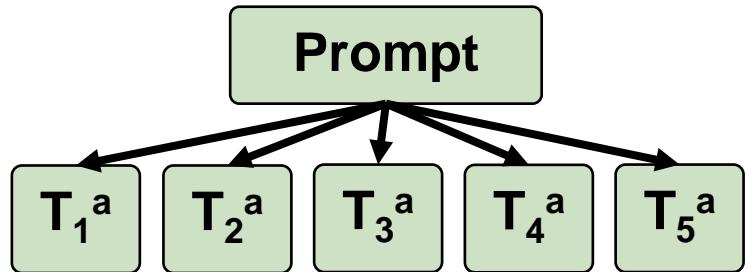
for $s' \in G(p_\theta, s, k)$ **do** \triangleright sorted candidates

if $V(p_\theta, \{s'\})(s) > v_{thres}$ **then** \triangleright pruning
 $DFS(s', t + 1)$

end if

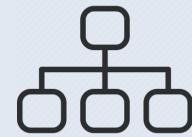
end for

Crossword Example



← **Generate k (5) thoughts, sort candidates.**

Tree-of-Thoughts - Implementation



Step 4. Search Algorithm

- Method to generate/traverse tree.

Strategy 2 – Depth First Search

- Evaluate most promising strategy first.
- Crossword-puzzle

Algorithm 2 ToT-DFS($s, t, p_\theta, G, k, V, T, v_{th}$)

Require: Current state s , step t , LM p_θ , thought generator $G()$ and size limit k , states evaluator $V()$, step limit T , threshold v_{th}

if $t > T$ **then** record output $G(p_\theta, s, 1)$

end if

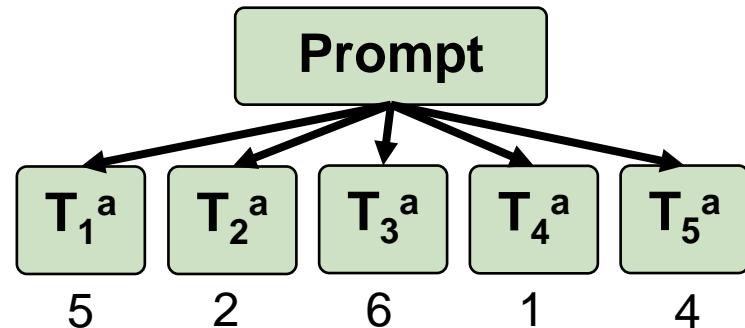
for $s' \in G(p_\theta, s, k)$ **do** \triangleright sorted candidates

if $V(p_\theta, \{s'\})(s) > v_{thres}$ **then** \triangleright pruning
 $DFS(s', t + 1)$

end if

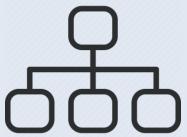
end for

Crossword Example



← **Generate k (5) thoughts, sort candidates.**

Tree-of-Thoughts - Implementation



Step 4. Search Algorithm

- Method to generate/traverse tree.

Strategy 2 – Depth First Search

- Evaluate most promising strategy first.
- Crossword-puzzle

Algorithm 2 ToT-DFS($s, t, p_\theta, G, k, V, T, v_{th}$)

Require: Current state s , step t , LM p_θ , thought generator $G()$ and size limit k , states evaluator $V()$, step limit T , threshold v_{th}

if $t > T$ **then** record output $G(p_\theta, s, 1)$

end if

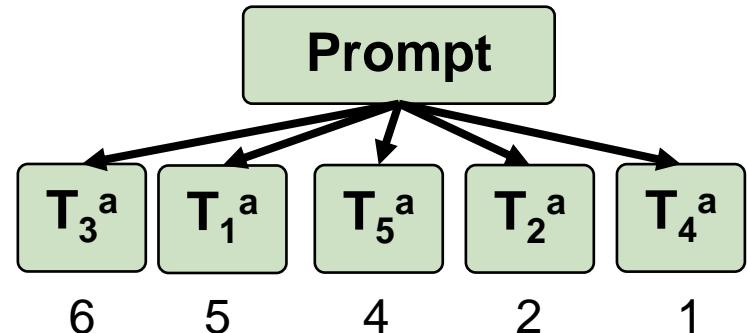
for $s' \in G(p_\theta, s, k)$ **do** \triangleright sorted candidates

if $V(p_\theta, \{s'\})(s) > v_{thres}$ **then** \triangleright pruning
 $DFS(s', t + 1)$

end if

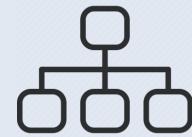
end for

Crossword Example



← **Generate k (5) thoughts, sort candidates.**

Tree-of-Thoughts - Implementation



Step 4. Search Algorithm

- Method to generate/traverse tree.

Strategy 2 – Depth First Search

- Evaluate most promising strategy first.
- Crossword-puzzle

Algorithm 2 ToT-DFS($s, t, p_\theta, G, k, V, T, v_{th}$)

Require: Current state s , step t , LM p_θ , thought generator $G()$ and size limit k , states evaluator $V()$, step limit T , threshold v_{th}

if $t > T$ **then** record output $G(p_\theta, s, 1)$

end if

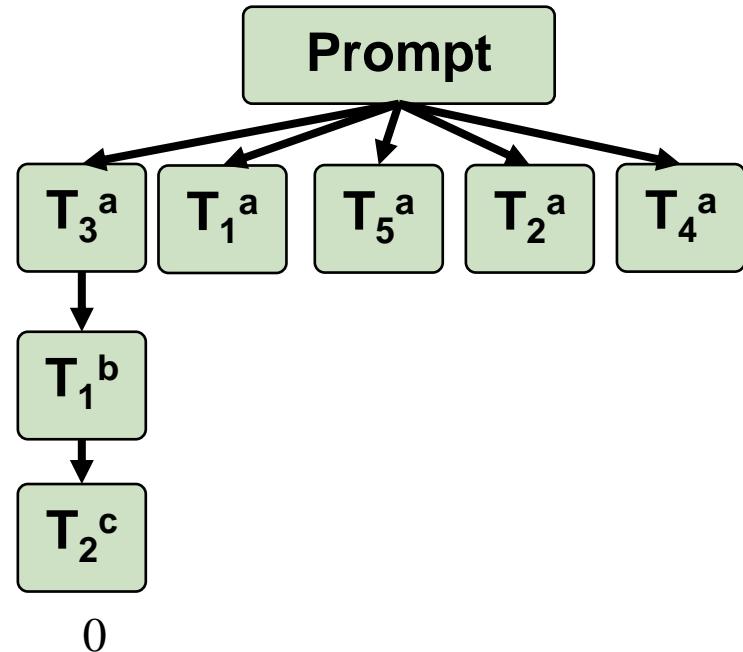
for $s' \in G(p_\theta, s, k)$ **do** \triangleright sorted candidates

if $V(p_\theta, \{s'\})(s) > v_{thres}$ **then** \triangleright pruning
 $\text{DFS}(s', t + 1)$

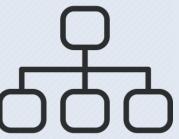
end if

end for

Crossword Example



**Exploit until
max depth (10)
or impossible.**



Tree-of-Thoughts - Implementation

Step 4. Search Algorithm

- Method to generate/traverse tree.

Strategy 2 – Depth First Search

- Evaluate most promising strategy first.
- Crossword-puzzle

Algorithm 2 ToT-DFS($s, t, p_\theta, G, k, V, T, v_{th}$)

Require: Current state s , step t , LM p_θ , thought generator $G()$ and size limit k , states evaluator $V()$, step limit T , threshold v_{th}

if $t > T$ **then** record output $G(p_\theta, s, 1)$

end if

for $s' \in G(p_\theta, s, k)$ **do** \triangleright sorted candidates

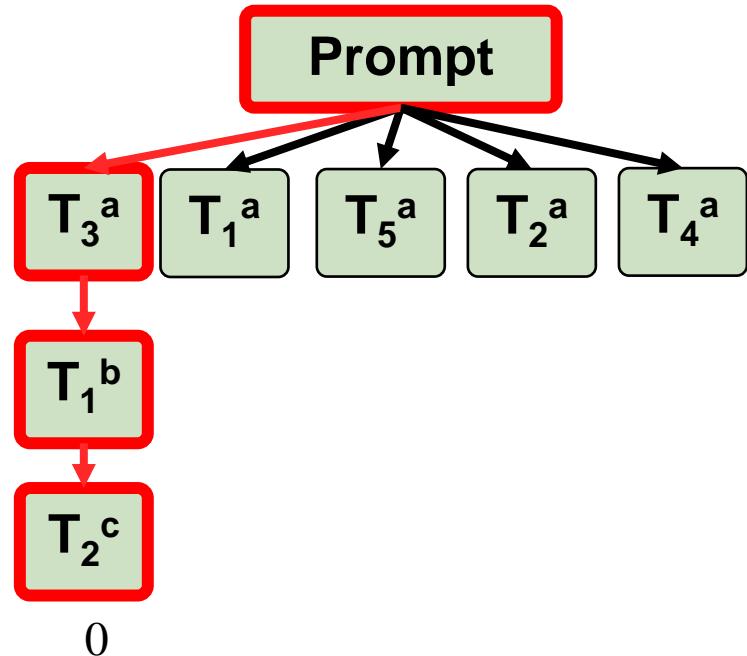
if $V(p_\theta, \{s'\})(s) > v_{thres}$ **then** \triangleright pruning

DFS($s', t + 1$)

end if

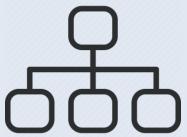
end for

Crossword Example



Prune if
impossible.

Tree-of-Thoughts - Implementation



Step 4. Search Algorithm

- Method to generate/traverse tree.

Strategy 2 – Depth First Search

- Evaluate most promising strategy first.
- Crossword-puzzle

Algorithm 2 ToT-DFS($s, t, p_\theta, G, k, V, T, v_{th}$)

Require: Current state s , step t , LM p_θ , thought generator $G()$ and size limit k , states evaluator $V()$, step limit T , threshold v_{th}

if $t > T$ **then** record output $G(p_\theta, s, 1)$

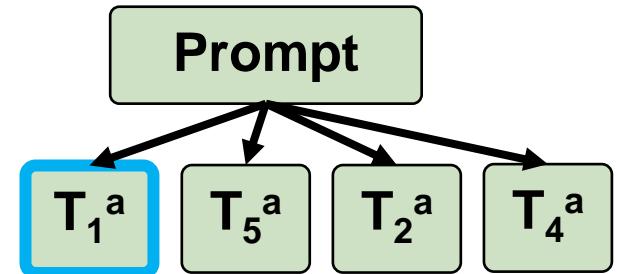
end if

for $s' \in G(p_\theta, s, k)$ **do** \triangleright sorted candidates
if $V(p_\theta, \{s'\})(s) > v_{thres}$ **then** \triangleright pruning
 DFS($s', t + 1$)

end if

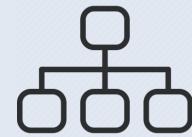
end for

Crossword Example



**Explore next
most probable
state.**

Tree-of-Thoughts - Implementation



Step 4. Search Algorithm

- Method to generate/traverse tree.

Strategy 2 – Depth First Search

- Evaluate most promising strategy first.
- Crossword-puzzle

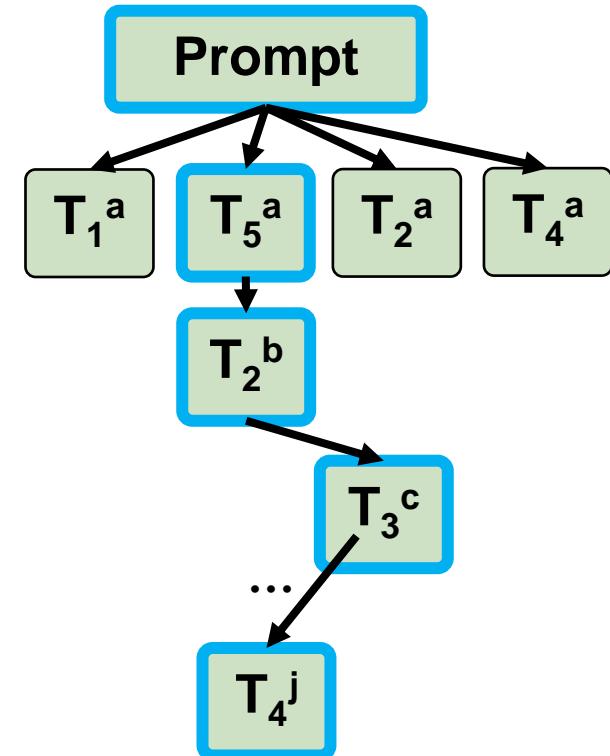
Algorithm 2 ToT-DFS($s, t, p_\theta, G, k, V, T, v_{th}$)

Require: Current state s , step t , LM p_θ , thought generator $G()$ and size limit k , states evaluator $V()$, step limit T , threshold v_{th}

```
if  $t > T$  then record output  $G(p_\theta, s, 1)$  ←  
end if  
for  $s' \in G(p_\theta, s, k)$  do    ▷ sorted candidates  
  if  $V(p_\theta, \{s'\})(s) > v_{thres}$  then ▷ pruning  
    DFS( $s', t + 1$ )  
  end if  
end for
```

**After t (100)
steps, return
deepest state.**

Crossword Example



Experiments

Main Tasks

Criteria: Hard even when sampling from state-of-the-art (SoTA) LMs

Arithmetic Reasoning

Game of 24

Consider the 4 numbers:
4, 9, 10, 13.

Write an equation, with operations (+,-,/,*) that results in 24.

Commonsense Reasoning

Creative Writing

Given the sentences:

1. "The dog ran fast."
2. "The man went to work."
3. "A car drove by."
4. "The floor is clean."

Write 4 paragraphs, each ending with the corresponding sentence above.

Lexical Reasoning

Mini Crosswords

S	L	U	F	F
L	O	S	E	R
U		A	L	I
R	E	G	L	E
P	L	E	A	D

Horizontal

- 1 To discard; to slough
- 2 One who loses a game; a person
- 3 Oorali, a poison used on arrows
- 4 A groove for a sliding door
- 5 To entreat

Vertical

- 1 To drink noisily
- 2 A worthless person
- 3 A habit or custom
- 4 Slang for "fellow"
- 5 Cooked by frying

Extensions to other tasks (Zero-shot ToTs)

Only 100 random test questions from the following tasks

Grade School Math 8K (GSM8K)

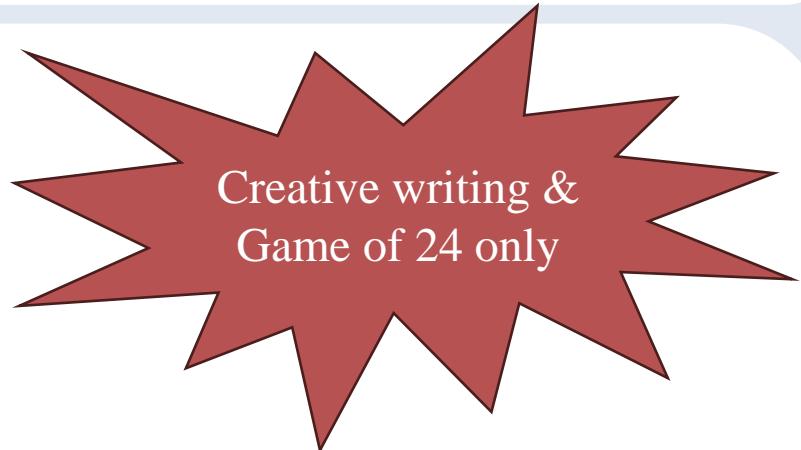
- 8500 high-quality, linguistically diverse math word problems designed for grade school students
- Created by OpenAI to evaluate and enhance the problem-solving and reasoning of LMs

StrategyQA

- Question-Answering benchmark designed to evaluate the model's ability to handle questions that require implicit multi-step reasoning.
- Necessary steps must be inferred.

Other Experiments

- Extensions to new LMs
 - o GPT-3.5 turbo for Creative Writing & Game of 24
- Cost and efficiency
 - o Generative/Prompt tokens, cost per case, success rate comparison
- Generation Vs Evaluation
 - o GPT-4 generation + GPT-3.5 evaluation Vs GPT-3.5 generation + GPT-4 evaluation



Results

Arithmetic Reasoning

Game of 24

Task Setup

1. Scraped 1362 games sorted from easy to hard by human solving time
2. Testing with subset of relatively 100 hard games
3. The output is success if and only if the equation equals 24 and uses each number exactly once.
4. Report the success rate across 100 games
5. CoT self-consistency takes the majority output from 100 CoT samples.
6. Iterative-refine on top of IO sample for at most 10 iterations
7. Different models for Generation vs evaluation

Results

Method	Success
IO prompt	7.3%
CoT prompt	4.0%
CoT-SC ($k=100$)	9.0%
ToT (ours) ($b=1$)	45%
ToT (ours) ($b=5$)	74%
IO + Refine ($k=10$)	27%
IO (best of 100)	33%
CoT (best of 100)	49%

Table 2: Game of 24 Results.

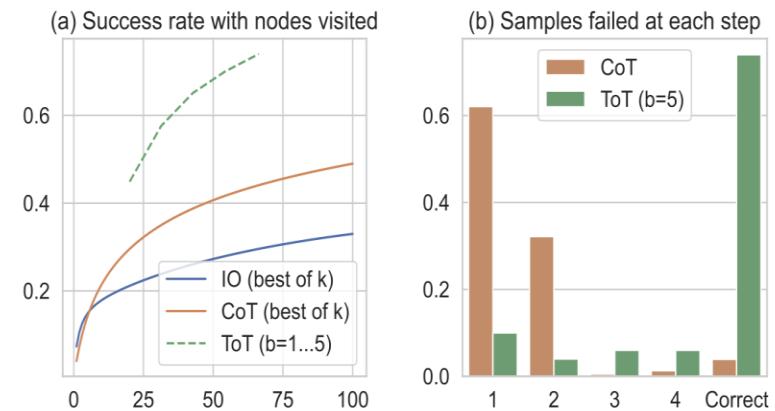
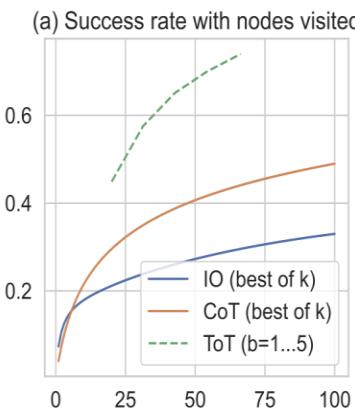


Figure 3: Game of 24 (a) scale analysis & (b) error analysis.

Generation	Evaluation	Success rate
GPT-4	GPT-3.5	64%
GPT-3.5	GPT-4	31%

Takeaways

- Relatively shallow ToT is enough (at most 3 thought steps).
- Thought generation is bottleneck.
- Different generation / evaluation models might attain decent results at lower costs

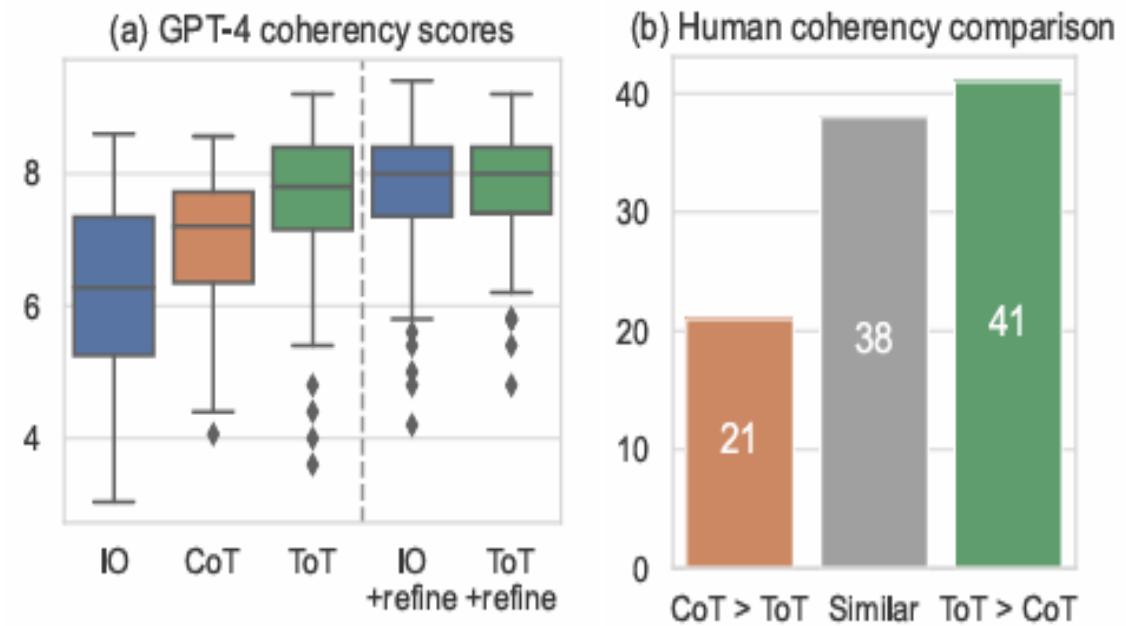
Commonsense Reasoning

Creative Writing

Task Setup

1. Sample random unlabeled sentences to form 100 inputs
2. Zero-shot prompting to provide score (1-10) by averaging 5 sampled scores.
3. Blind study on authors to compare coherency for CoT vs ToT generated passage pairs.
4. Iterative-refine ($k < 5$) on top of a random IO sample for each task.

Results



Takeaways

- ToT wrote most coherent paragraphs on average (7.56) vs IO (6.19) and CoT (6.93).
- Iterative-refine improves the IO coherence from 6.19 to 6.67 and ToT from 7.56 to 7.91
- Human evaluators confirmed the observation (IO < CoT < ToT)
- Relatively shallow ToT is enough (1 thought step).

Lexical Reasoning

Mini Crosswords

Task Setup

1. Scrape 156 games of $5 * 5$ mini crosswords
2. Use 20 games for testing and 5 for prompting
3. The input describes the 5 horizontal and vertical clues. The output should be a board of $5 * 5 = 25$ letters to solve the puzzle
4. 3 levels of success: portion of correct letters (25 per game), words (10 per game), and games.

Results

Method	Success Rate (%)		
	Letter	Word	Game
IO	38.7	14	0
CoT	40.6	15.6	1
ToT (ours)	78	60	20
+best state	82.4	67.5	35
-prune	65.4	41.5	5
-backtrack	54.6	20	5

Table 3: Mini Crosswords results.

Takeaways

- DFS on ToT with at most 100 steps gives the best result
- Better heuristics for DFS pruning are critical for problem solving in this case

Extensions to other tasks

Zero-shot ToTs

Task Setup

1. Generic zero-shot ToT-BFS for both GSM8K and Strategy QA
2. Sample 5 strategies and then vote for the best one before sampling 5 solutions based on the best strategy then vote for the best one.
3. Evaluated on subset of random 100 questions for both tasks.

Results

	GSM8K	StrategyQA
IO	51	73
CoT	86	82
ToT	90	83

Table 4: New tasks with zero-shot ToT and GPT-4.

Takeaways

- Extending ToT to new tasks could be straight forward
- CoT is already very good on such tasks.
- It is more suitable to try smaller LLMs+ ToT for traditional NLP tasks or GPT4 + ToT for hard tasks that challenge GPT-4 + CoT's reasoning

Other Experiments

Cost Analysis

Extensions to Other LMs

Game of 24	Generate/Prompt tokens	Cost per case	Success
IO (best of 100)	1.8k / 1.0k	\$0.13	33%
CoT (best of 100)	6.7k / 2.2k	\$0.47	49%
ToT	5.5k / 1.4k	\$0.74	74%

Table 7: Cost analysis on Game of 24.

	GPT-4	GPT-3.5
IO	7.3%	6%
CoT	4.0%	3%
ToT	74%	19%

Table 5: Game of 24 with GPT-4 vs GPT-3.5.

Creative Writing	Generate/Prompt tokens	Cost per case
IO	0.9k / 0.4k	\$0.06
CoT	0.9k / 0.4k	\$0.07
ToT	4k / 2.9k	\$0.32

Table 8: Cost analysis on Game of 24.

	GPT-4	GPT-3.5
IO	6.19	4.47
CoT	6.93	5.16
ToT	7.56	6.62

Table 6: Creative Writing with GPT-4 vs. GPT-3.5.

Takeaways

- ToT needs significantly more computations than IO or CoT prompting
- Performance gain worths the additional cost
- Total cost for Game of 24 + Creative Writing = \$106
- Estimated cost for Crosswords ~ \$100

Conclusions

Contributions/Impact

Reproducibility

Contributions

- 1. Significant improvements on s.o.t.a LLM (GPT-4) for planning.**
- 2. Provides the ToT framework.**
 1. Open-source, accessible.
 2. Applicable to other tasks.
- 3. Developed benchmark for LLM planning abilities.**
 1. Game of 24, creative writing, crosswords.

Impact

- 1. High: >1000 citations**
 - 1. Inspired works:**
 - **Graph-of-Thoughts**
 - **Improvements, limitations, integrations.**

Reproducible? Yes*

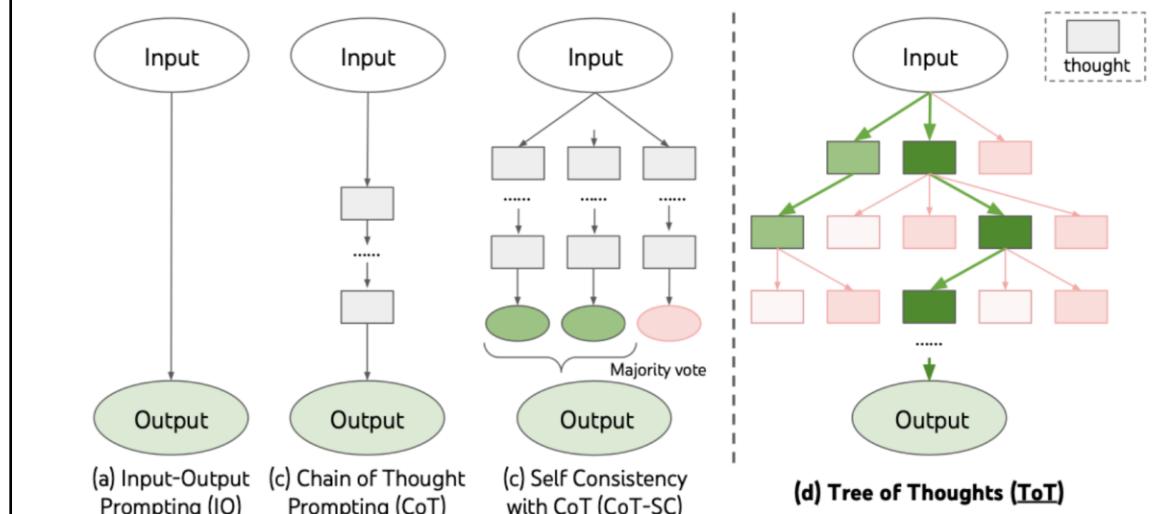
1. LLMs utilized are **accessible** (GPT-4, GPT-3.5).
2. **Framework** open-source (w/ prompt logs).

Potential Concerns

1. **API cost** (lots of inferencing)
2. **Temperature** – non-deterministic inferencing

Official Repo of Tree of Thoughts (ToT)

pypi package 0.1.0 Python 3.7+ License MIT DOI 10.5281/zenodo.8118487



<https://github.com/princeton-nlp/tree-of-thought-llm>

Conclusions

Limitations

Future Work

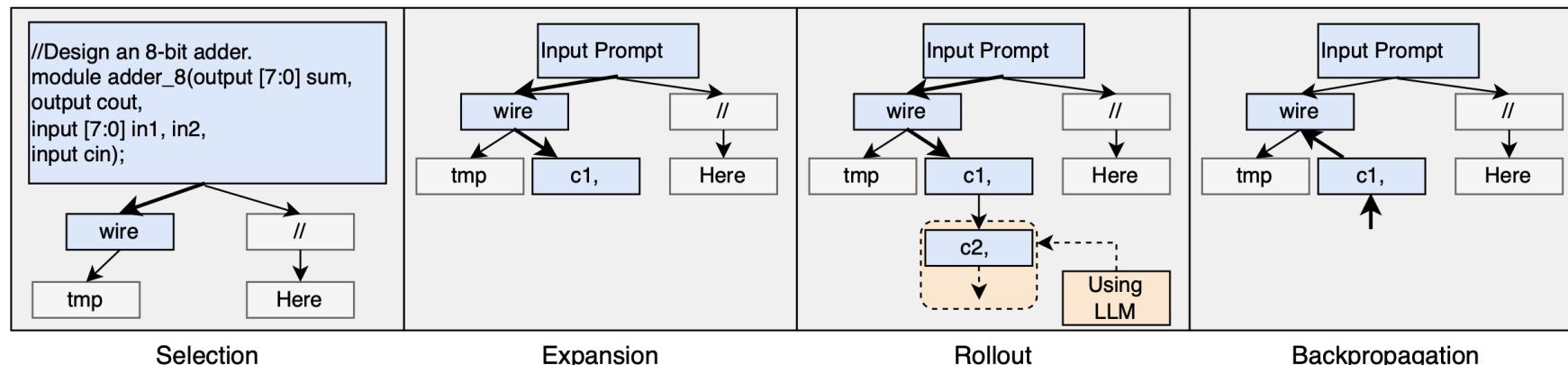
Limitations

- 1. Significant computation overhead**
 - ~100x tokens generated vs CoT.
- 2. No tool learning strategies**

Timo Schick, J. D.-Y. (2023). Toolformer: Language Models Can Teach Themselves to Use Tools. arXiv.

Future Work

- Additional **tree-search strategies**
 - Monte-Carlo Tree Search (feedback).
- **Applications to programming** (C, Python, Verilog).
 - Define “thoughts,” tool feedback in heuristics.
- **“Make Every Move Count”** explores this.
 - Verilog generation – functionality, PPA.
- Multi-agent frameworks where specialized agents are used to prompt the models themselves instead of hand-crafting the prompts



M.DeLorenzo,A.B.Chowdhury,V.Gohil,S.Thakur,R.Karri,S.Garg, and J. Rajendran, “Make every move count: LLM-based high-quality RTL code generation using MCTS,” *arXiv preprint arXiv:2402.03289*, 2024.



Review: Training Compute-Optimal Large Language Models

Manoj Kumar Reddy Peta

Vamshi Krishna Battala

Background

- **Prevailing Trend & Challenges:**
 - ◆ Larger model size → Better performance
 - ◆ Traditional scaling laws (Kaplan et al. 2020):
 - Recommend 70:30 ratio for model size vs. dataset size increase
- **Key Observations:**
 - ◆ Current large language models are under-fit
 - ◆ Focus has been on increasing model size without proportional increase in training data
 - ◆ Optimal training strategy may not be just increasing model size
- Larger models face challenges: higher costs, slower performance, limited deployment options, and environmental impact

We already have “Scaling Laws”. Why this again?



→ **Key Issues with Original Scaling Laws:**

- ◆ Fixed LR schedule for all models → Underestimated performance on fewer tokens
- ◆ Focused on small models (<100M params) → Missed trends in larger scales
- ◆ Recommended dramatic model size increase (0.73) vs token increase (0.27)

→ **What This Paper Fixed:**

- ◆ Tested 400+ models up to 16B params with varied schedules
- ◆ Matched LR schedules to actual training duration
- ◆ **Found equal scaling (0.5) for both model size and tokens**

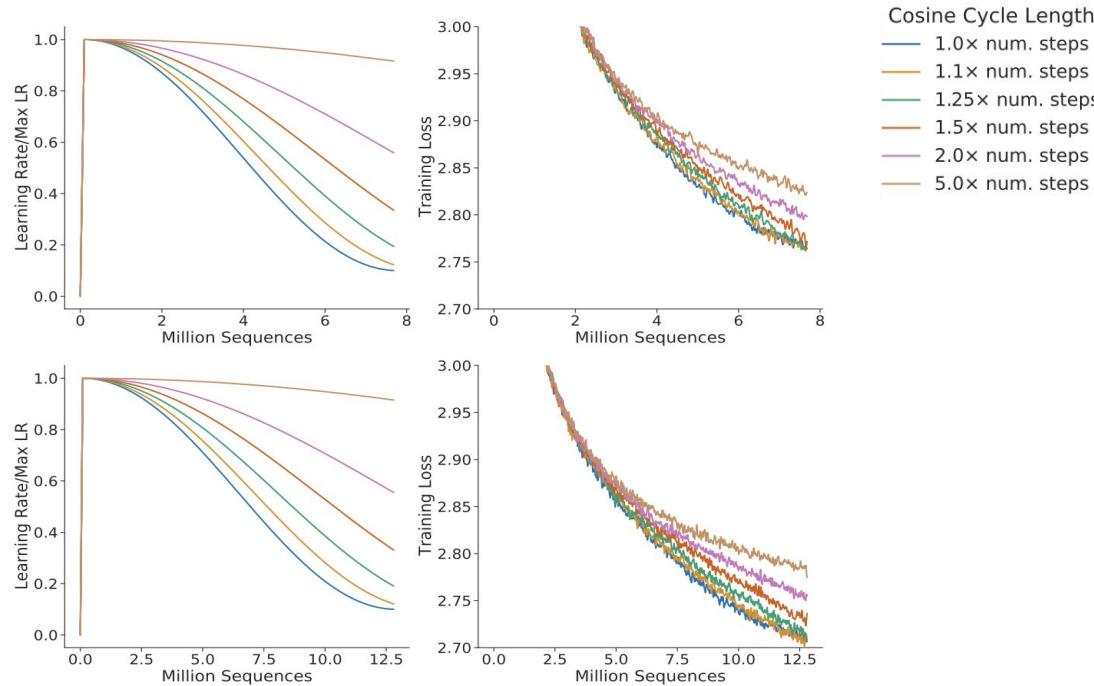
→ **Result:** Chinchilla (70B) outperformed Gopher (280B) using same compute but more optimal allocation

We already have “Scaling Laws” paper? Why this again?



TEXAS A&M
UNIVERSITY

→ Learning Rate Analysis



Overview

- The paper answers the question - “**Given a fixed compute how should we balance model size and training data?**”
- Over 400 language models ranging from 70 million to over 16 billion parameters on 5 to 500 billion tokens are trained and it is found that for compute-optimal training, the model size and the number of training tokens should be **scaled equally**.
- Three different approaches are presented in this paper to evaluate the relationship between the model size and the size of the dataset.
- Tests show that Chinchilla, a model with 70B parameters trained on four times more data within the same compute budget as Gopher, consistently and significantly outperforms larger models like Gopher (280B), GPT-3 (175B), Jurassic-1 (178B), and Megatron-Turing NLG (530B) across a variety of evaluation tasks.

Methodology- Approach 1



Fix model sizes and vary number of training tokens

Training Setup

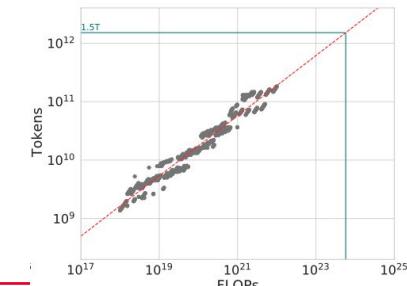
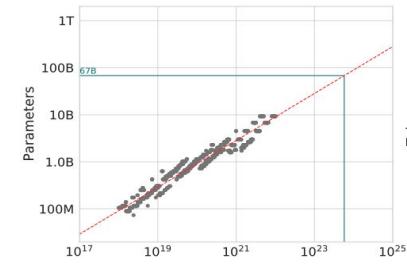
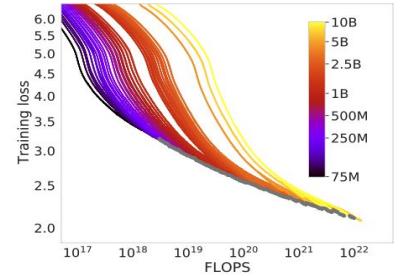
- Range of models: 70M → 10B parameters
- Four different training horizons per model
- Used cosine schedule for learning rate decay
learning rate decays by 10x during training

Empirical Loss Analysis

- Smoothed and interpolated training loss curves
- Created continuous mapping: FLOP count → training loss
- For each FLOP count, identified which run achieved lowest loss

Finding Optimal Scaling

- Found optimal model size at each FLOP count
- Fitted power laws to estimate optimal scaling
- Found relationship: $N_{opt} \propto C^{0.50}$, $D_{opt} \propto C^{0.50}$



Methodology- Approach 2

IsoFLOP

Training Setup

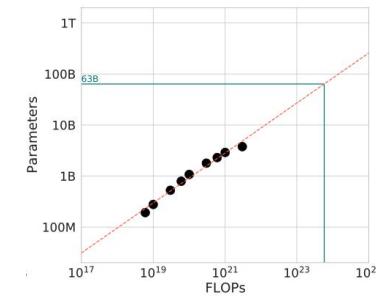
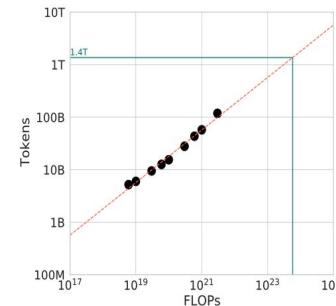
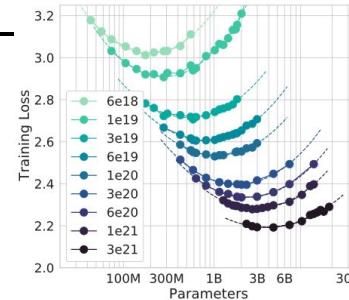
- Fixed FLOP budgets approach
- 9 different budgets (6×10^{18} to 3×10^{21} FLOPs)
- Varied model sizes up to 16B parameters
- Let training tokens be determined by FLOP budget

Experimental Setup

- Each FLOP budget → multiple model sizes
- Cosine schedule matched to training length
- Found clear minimum loss for each budget

Analysis Process

- Plotted final loss vs. parameter count
- Fitted parabola to each IsoFLOP curve
- Located minimum point for each curve
- Found scaling law: $N_{opt} \propto C^{0.49}$, $D_{opt} \propto C^{0.51}$



Methodology- Approach 3



Fitting a parametric loss function

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^\alpha} + \frac{B}{D^\beta}$$

Where:

- E: Base loss (text entropy)
- A/N^α: Model size dependent term
- B/D^β: Training data dependent term

- They fit this function to all final losses from Approaches 1 & 2 by minimizing the difference between the predicted and actual losses using Huber loss
- The optimal allocation of compute budget C can be derived mathematically from this equation, giving:
 - $N \propto C^{0.46}$
 - $D \propto C^{0.54}$
- This approach predicted even smaller optimal models than Approaches 1 & 2

Chinchilla



- Based on the findings, the optimal model size for the Gopher compute budget is somewhere between 40 and 70 billion parameters.
- To test this, an optimal model, **Chinchilla, with 70 billion parameters, but with 4 times more training data is trained.**
- Chinchilla is trained on the MassiveText dataset with a tailored subset distribution to manage more training tokens, uses **AdamW** for better language modeling and performance after fine tuning, and incorporates a slightly modified **SentencePiece tokenizer** that enhances representation in fields like mathematics and chemistry.
- Both Chinchilla and Gopher have been trained for the same number of FLOPs but differ in the size of the model and the number of training tokens.

Model	Layers	Number Heads	Key/Value Size	d_{model}	Max LR	Batch Size
Gopher 280B	80	128	128	16,384	4×10^{-5}	3M → 6M
Chinchilla 70B	80	64	128	8,192	1×10^{-4}	1.5M → 3M

Results



MMLU

Random	25.0%
Average human rater	34.5%
GPT-3 5-shot	43.9%
<i>Gopher</i> 5-shot	60.0%
<i>Chinchilla</i> 5-shot	67.6%
Average human expert performance	89.8%
June 2022 Forecast	57.1%
June 2023 Forecast	63.4%

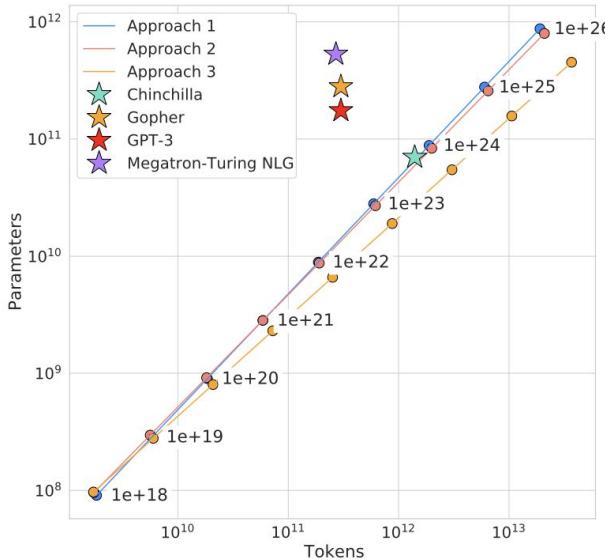
Gender Bias

	<i>Chinchilla</i>	<i>Gopher</i>
All	78.3%	71.4%
Male	71.2%	68.0%
Female	79.6%	71.3%
Neutral	84.2%	75.0%

Common Sense Benchmarks

	<i>Chinchilla</i>	<i>Gopher</i>	GPT-3	MT-NLG 530B	Supervised SOTA
HellaSWAG	80.8%	79.2%	78.9%	80.2%	93.9%
PIQA	81.8%	81.8%	81.0%	82.0%	90.1%
Winogrande	74.9%	70.1%	70.2%	73.0%	91.3%
SIQA	51.3%	50.6%	-	-	83.2%
BoolQ	83.7%	79.3%	60.5%	78.2%	91.4%

Results



opt Parameters	FLOPs	opt Tokens	Model	Actual Tokens
400 Million	1.92×10^{19}	8.0 Billion		
1.5 Billion	1.21×10^{20}	30 Billion	GPT-2	8 Billion
10 Billion	1.23×10^{22}	205.1 Billion		
67 Billion	5.76×10^{23}	1.5 Trillion	Chinchilla	1.4 Trillion
175 Billion	3.85×10^{24}	3.7 Trillion	GPT-3, Jurassic-I	300 Billion
280 Billion	9.90×10^{24}	5.9 Trillion	Gopher	300 Billion
520 Billion	3.43×10^{25}	11.0 Trillion	MT-NLG	270 Billion
1 Trillion	1.27×10^{26}	21.2 Trillion		
10 Trillion	1.30×10^{28}	216.2 Trillion		

Future Work



- Multi-Epoch Training Dynamics
 - ◆ How do scaling laws change with multiple epochs?
 - ◆ When does data repetition help vs. harm?
- Scaling Limits Investigation
 - ◆ Understanding diminishing returns at high FLOPS
 - ◆ Investigating loss convexity patterns
- Data Quality and Domain Adaptation
 - ◆ Developing data quality metrics
 - ◆ Cross-domain validation of scaling laws

THANK YOU