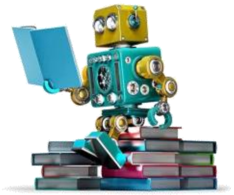
SDML
ML Paper Review

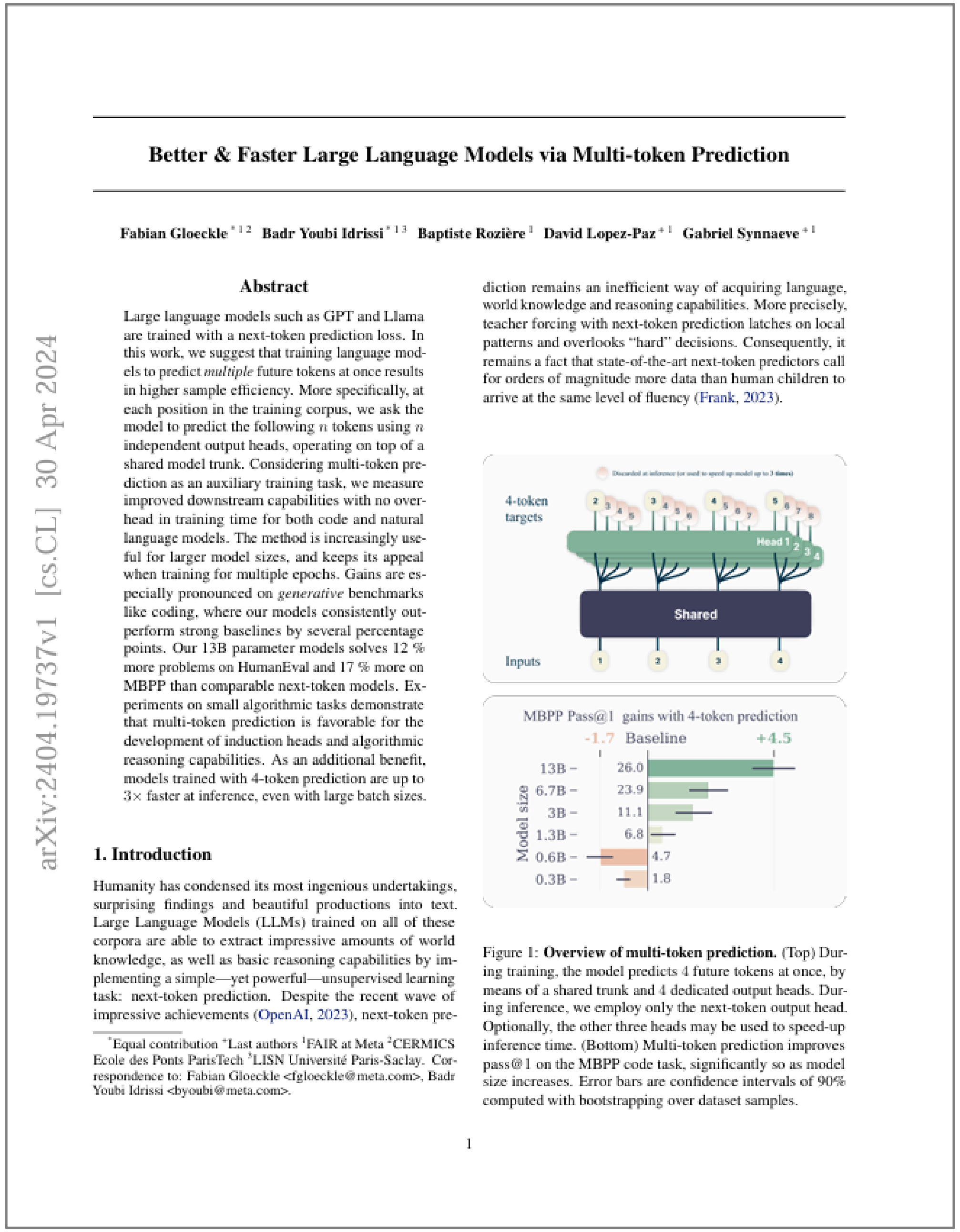
June 2024



Better & Faster Large Language Models via Multi-token Prediction

Fabian Gloeckle et al., FAIR

https://arxiv.org/abs/2404.19737



arXiv:2404.19737v1 [cs.CL] 30 Apr 2024

Better & Faster Large Language Models via Multi-token Prediction

Fabian Gloeckle [* 1 2] **Badr Youbi Idrissi** [* 1 3] **Baptiste Rozière** [1] **David Lopez-Paz** [+ 1] **Gabriel Synnaeve** [+ 1]

Abstract

Large language models such as GPT and Llama are trained with a next-token prediction loss. In this work, we suggest that training language models to predict *multiple* future tokens at once results in higher sample efficiency. More specifically, at each position in the training corpus, we ask the model to predict the following n tokens using n independent output heads, operating on top of a shared model trunk. Considering multi-token prediction as an auxiliary training task, we measure improved downstream capabilities with no overhead in training time for both code and natural language models. The method is increasingly useful for larger model sizes, and keeps its appeal when training for multiple epochs. Gains are especially pronounced on *generative* benchmarks like coding, where our models consistently outperform strong baselines by several percentage points. Our 13B parameter models solves 12 % more problems on HumanEval and 17 % more on MBPP than comparable next-token models. Experiments on small algorithmic tasks demonstrate that multi-token prediction is favorable for the development of induction heads and algorithmic reasoning capabilities. As an additional benefit, models trained with 4-token prediction are up to 3× faster at inference, even with large batch sizes.

1. Introduction

Humanity has condensed its most ingenious undertakings, surprising findings and beautiful productions into text. Large Language Models (LLMs) trained on all of these corpora are able to extract impressive amounts of world knowledge, as well as basic reasoning capabilities by implementing a simple—yet powerful—unsupervised learning task: next-token prediction. Despite the recent wave of impressive achievements (OpenAI, 2023), next-token prediction remains an inefficient way of acquiring language, world knowledge and reasoning capabilities. More precisely, teacher forcing with next-token prediction latches on local patterns and overlooks "hard" decisions. Consequently, it remains a fact that state-of-the-art next-token predictors call for orders of magnitude more data than human children to arrive at the same level of fluency (Frank, 2023).

Figure 1: **Overview of multi-token prediction.** (Top) During training, the model predicts 4 future tokens at once, by means of a shared trunk and 4 dedicated output heads. During inference, we employ only the next-token output head. Optionally, the other three heads may be used to speed-up inference time. (Bottom) Multi-token prediction improves pass@1 on the MBPP code task, significantly so as model size increases. Error bars are confidence intervals of 90% computed with bootstrapping over dataset samples.

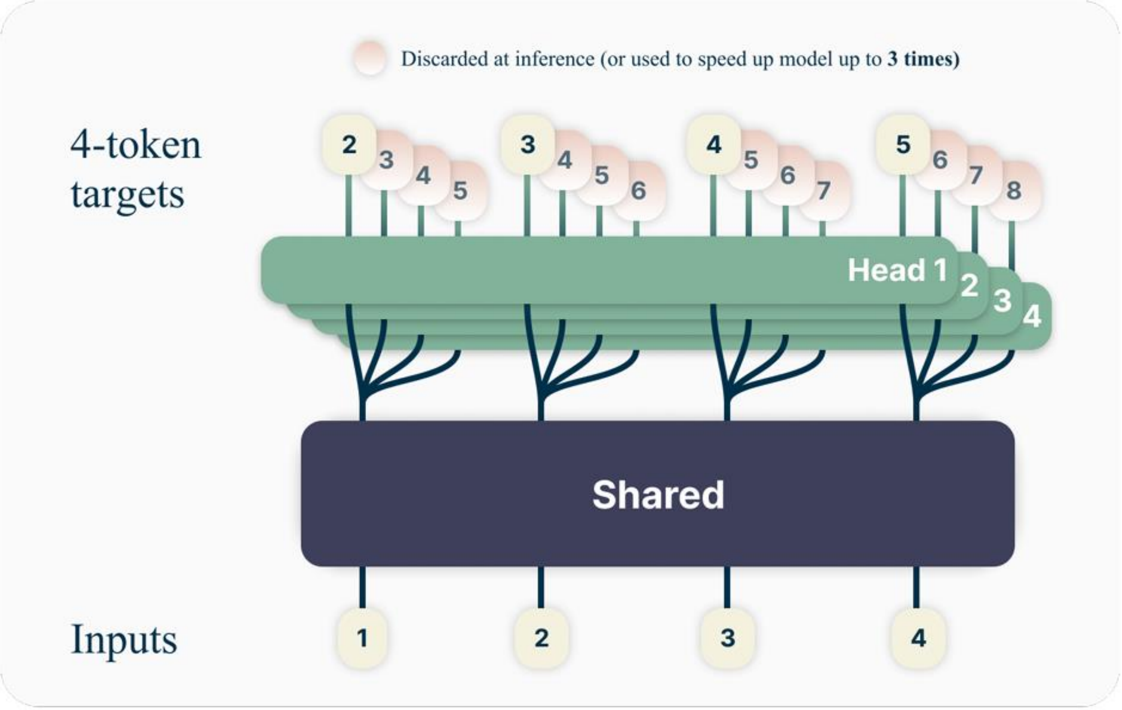
*Equal contribution +Last authors [1]FAIR at Meta [2]CERMICS Ecole des Ponts ParisTech [3]LISN Université Paris-Saclay. Correspondence to: Fabian Gloeckle <fgloeckle@meta.com>, Badr Youbi Idrissi <byoubi@meta.com>.

Multi-token prediction overview

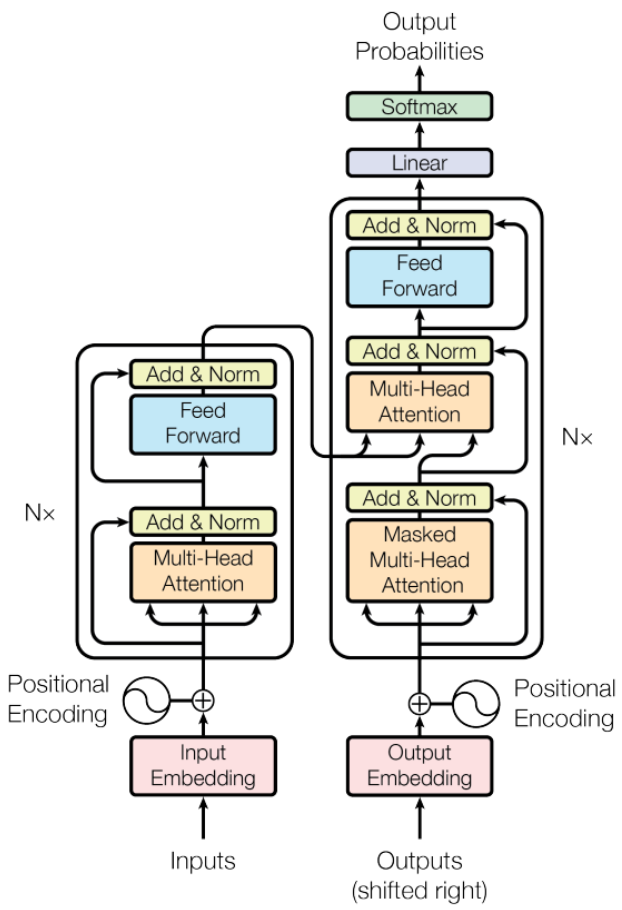
- Standard LLMs like LLaMA are trained (pre-training) with cross-entropy loss on predicting the next token
- Simple concept is to share the transformer body but add multiple prediction heads for, say, the next 4 tokens
  - During inference, the addition prediction heads are discarded
- Does the model train faster?
- Model seems to perform better on downstream tasks like coding and reasoning tasks
- Optionally, the additional prediction heads can be used for speculative decoding to speed up inference

Multi-token prediction method

- The main transformer body is unchanged and shared across all prediction heads
  - This minimizes increase in params
- Additional prediction heads, in addition to predicting the next token, also predict the token after the next one, etc.
- Generally compared performance with normal LLMs with same number of parameters, avoiding unfair advantage

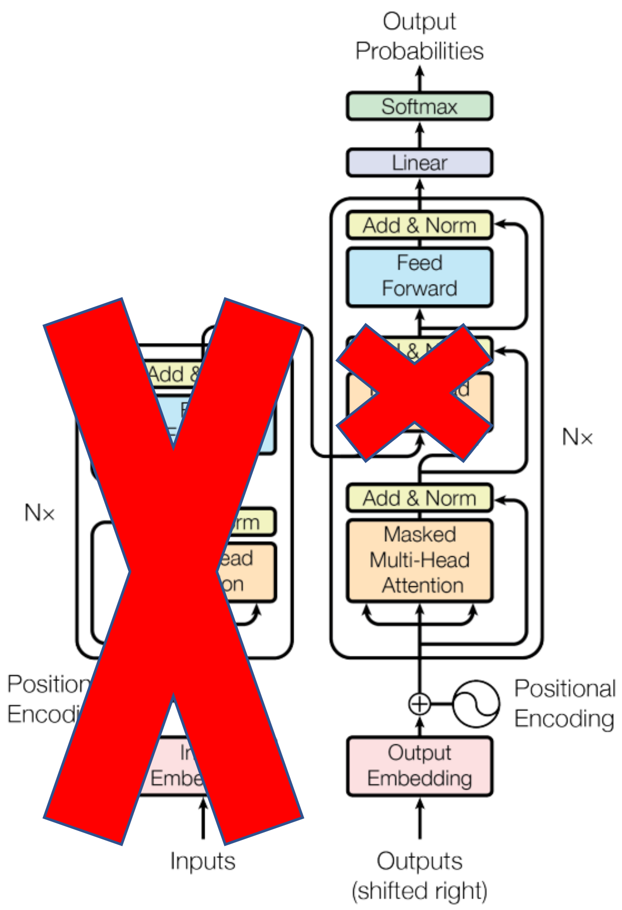


Original Transformer

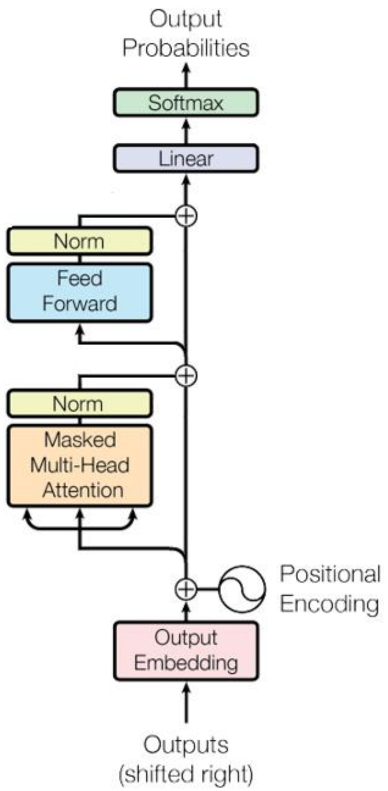


"Attention Is All You Need," Vaswani et al., https://arxiv.org/abs/1706.03762

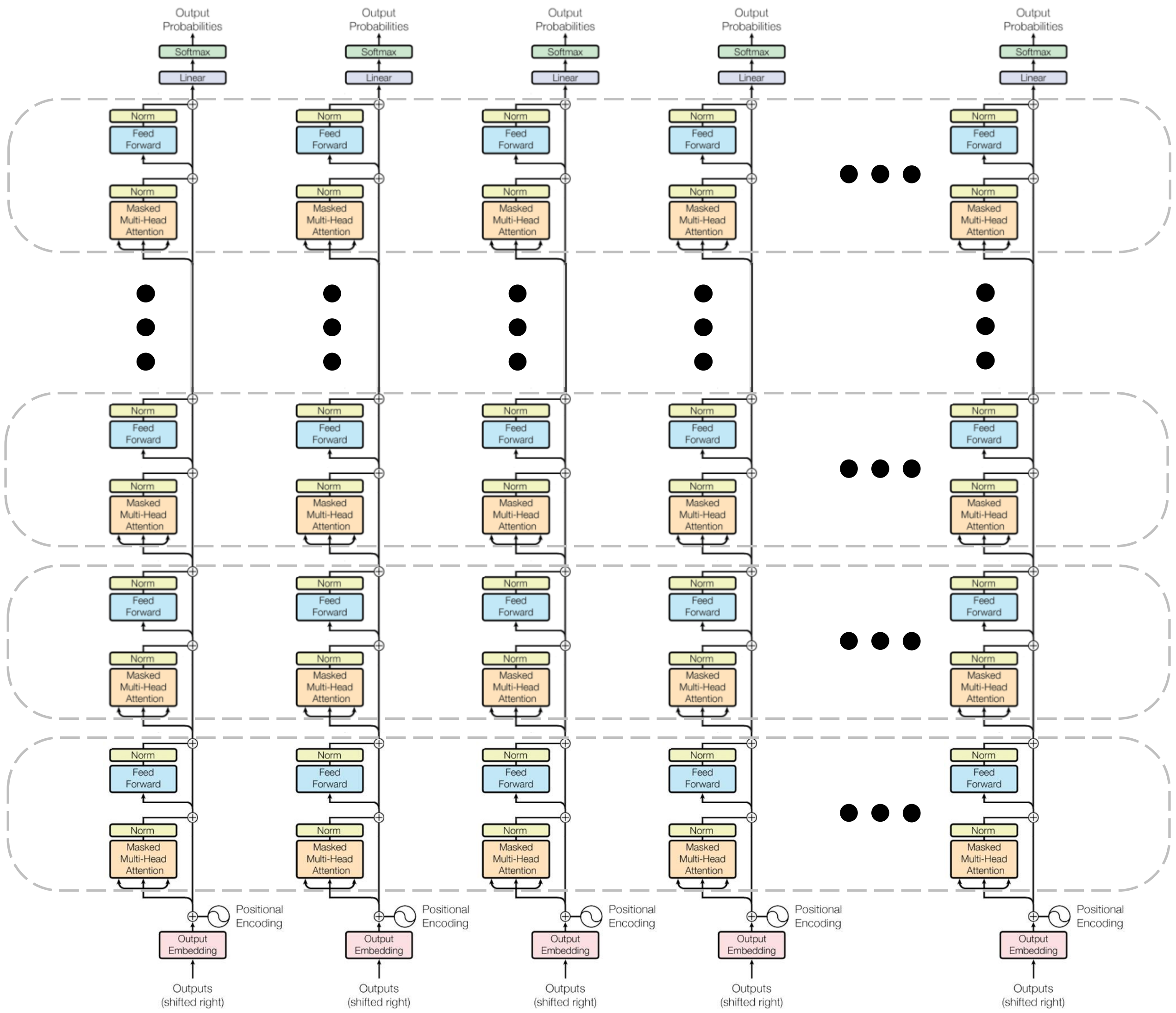
Decoder-only Transformer



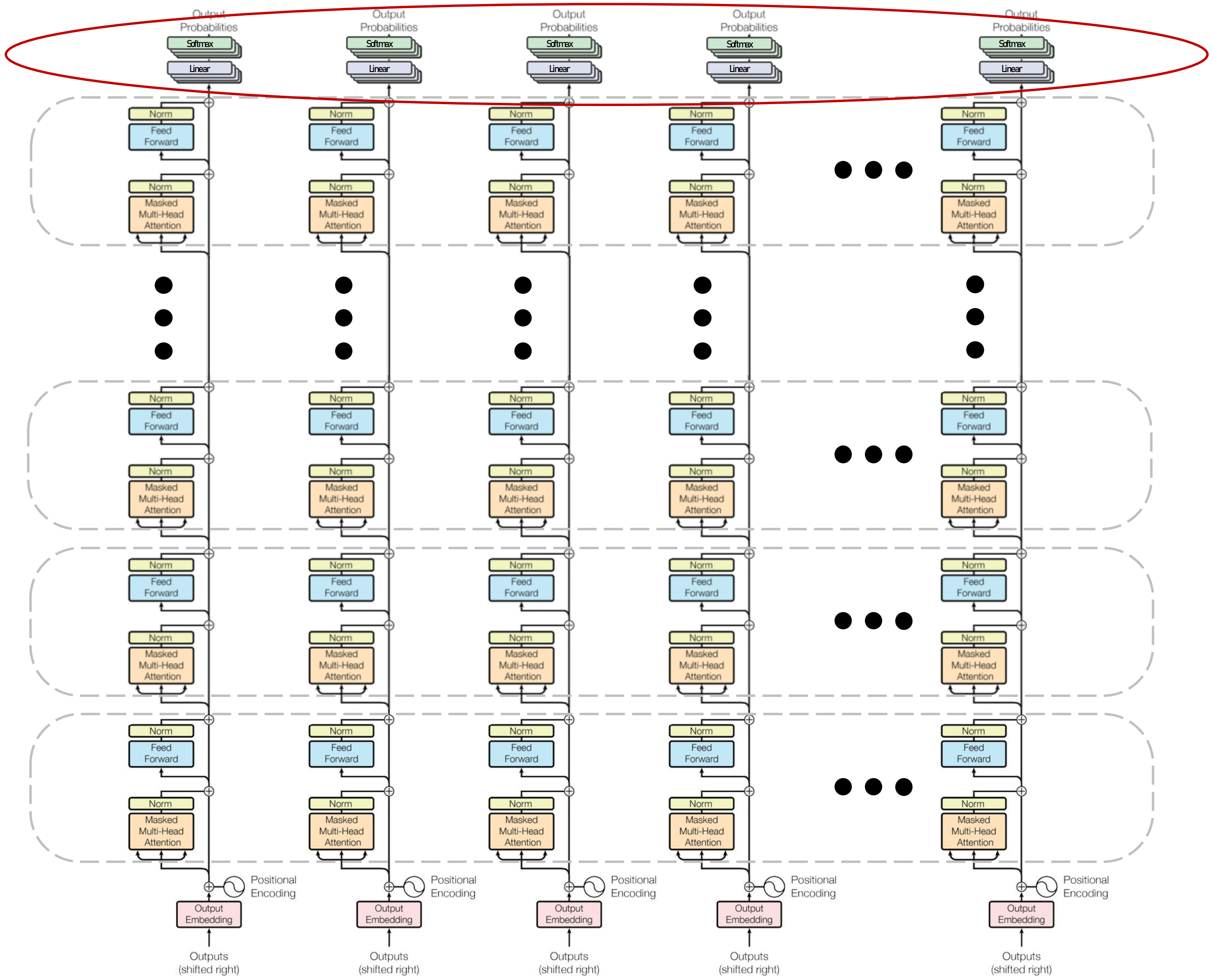
Decoder-only Transformer, Straight Through



Decoder, Multiple Tokens and Multiple Layers



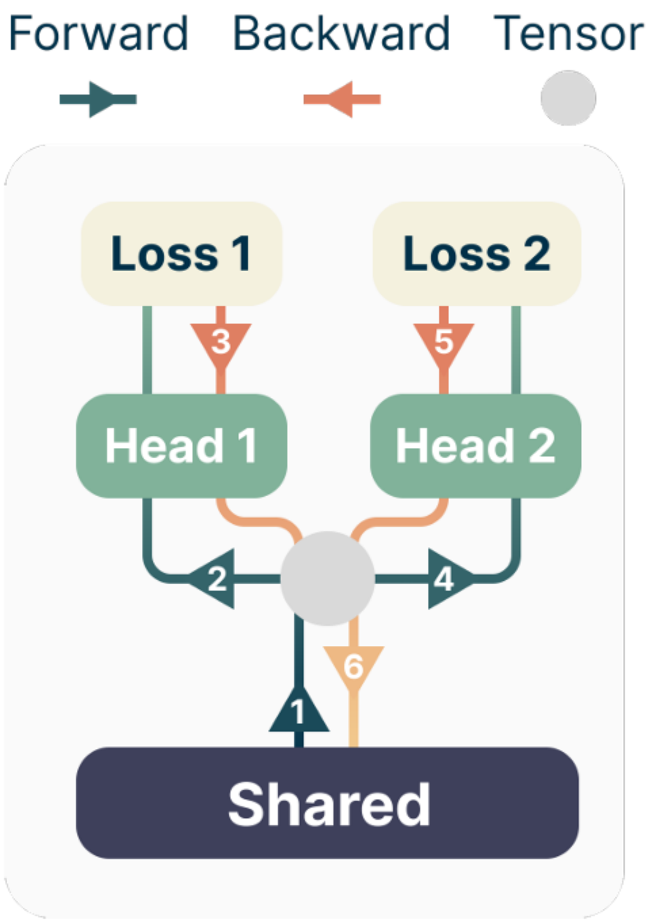
Decoder, Multiple Prediction Heads



This is the only architecture change

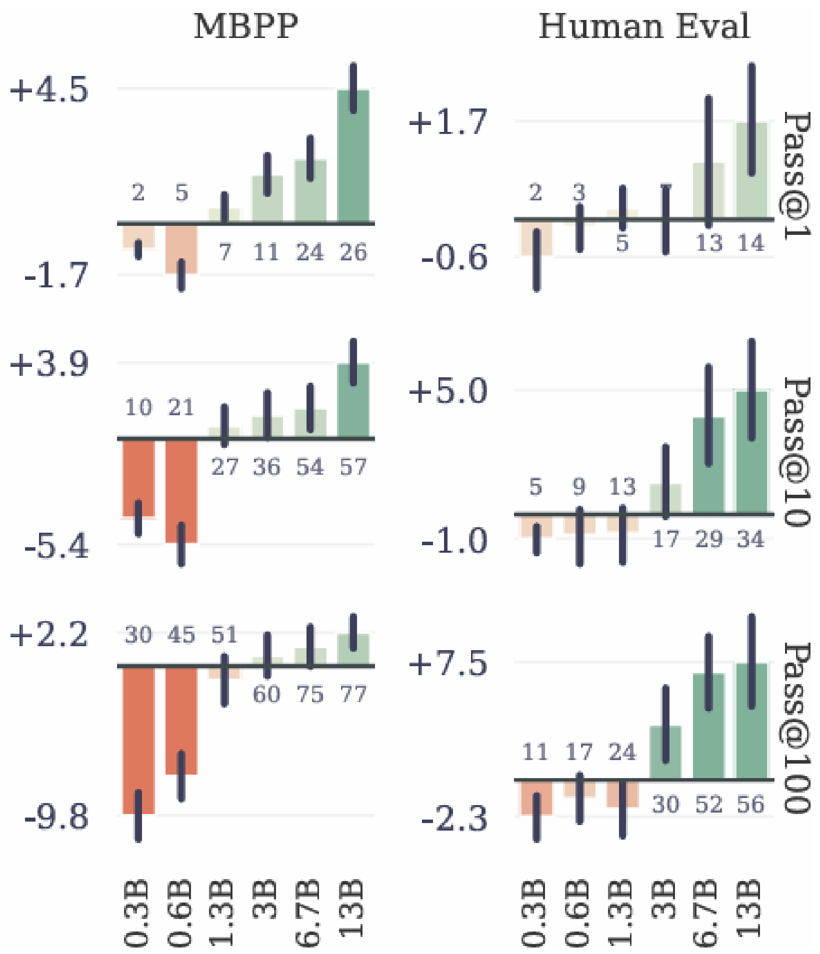
Memory-efficient training

- Clever tweak reduces GPU memory for training
- Normal flow is you do the forward pass over your entire model before staring backward pass
  - A lot of memory is used to keep track of gradients
- Prediction heads are large matrices that are model_dim x vocab_size, and vocab is very big
- After forward pass for each head, they do a small backward step and save results
  - The accumulated gradient is much smaller than what would have been stored normally



Results for scaling model size

- Tested models trained on code
- Varied size from 300M to 13B params
- Multi-token prediction did worse for the smaller models, but did better and better as the size scaled up
- Authors hypothesize that the usefulness only for larger models is why people haven’t noticed before
- Ted’s hypothesis: the model dimension on smaller models is too small, lacking bandwidth for multi-token information



Byte-level tokenization, optimal *n*, multi-epoch

- Tested variant where all text is tokenized into single bytes
  - Information is spread over many more tokens. Requires longer range context.
  - The multi-byte prediction beat next-byte by larger margins
- For tokens, best was predicting 4 next tokens
  - Depends on data?
- Generally works when training for multiple epochs

Training data	Vocabulary	n	MBPP			HumanEval			APPS/Intro		
			@1	@10	@100	@1	@10	@100	@1	@10	@100
313B bytes (0.5 epochs)	bytes	1	19.3	42.4	64.7	18.1	28.2	47.8	0.1	0.5	2.4
		8	**32.3**	**50.0**	**69.6**	**21.8**	**34.1**	**57.9**	**1.2**	**5.7**	**14.0**
		16	28.6	47.1	68.0	20.4	32.7	54.3	1.0	5.0	12.9
		32	23.0	40.7	60.3	17.2	30.2	49.7	0.6	2.8	8.8
200B tokens (0.8 epochs)	32k tokens	1	30.0	53.8	73.7	22.8	36.4	62.0	2.8	7.8	17.4
		2	30.3	55.1	76.2	22.2	38.5	62.6	2.1	9.0	21.7
		4	**33.8**	**55.9**	**76.9**	**24.0**	**40.1**	**66.1**	1.6	7.1	19.9
		6	31.9	53.9	73.1	20.6	38.4	63.9	**3.5**	**10.8**	**22.7**
		8	30.7	52.2	73.4	20.0	36.6	59.6	3.5	10.4	22.1
1T tokens (4 epochs)	32k tokens	1	40.7	65.4	83.4	31.7	57.6	83.0	**5.4**	**17.8**	**34.1**
		4	**43.1**	65.9	83.7	31.6	57.3	**86.2**	4.3	15.6	33.7

Fine-tuning results

- Previous results were just the pre-trained model
- Also fine-tuned for code
- Tried fine-tuning with the 4 prediction heads or just with 1 prediction head
- Both beat the model that was pretrained with next token prediction only

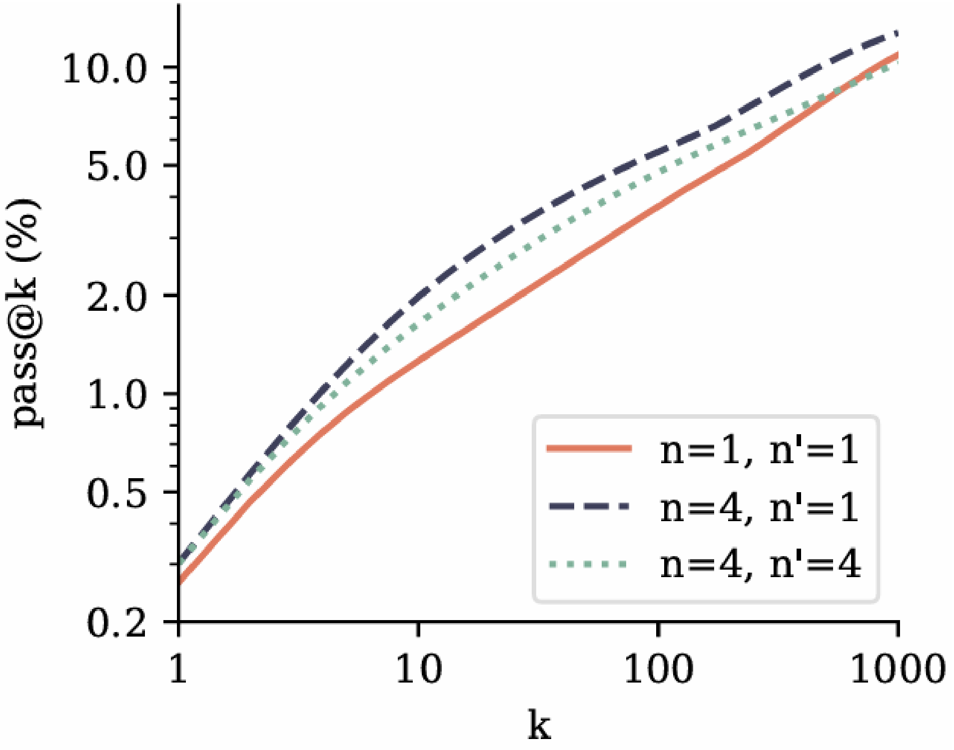


Figure 4: **Comparison of finetuning performance on CodeContests.**

Natural language results

- Previous results were on code. Next, studied natural language.
- Using standard benchmarks for NLP (some older), for 7B models, the 2-token prediction is okay, but the 4-token model does worse most of the time
  - Perhaps this is due to reduced non-prediction parameters

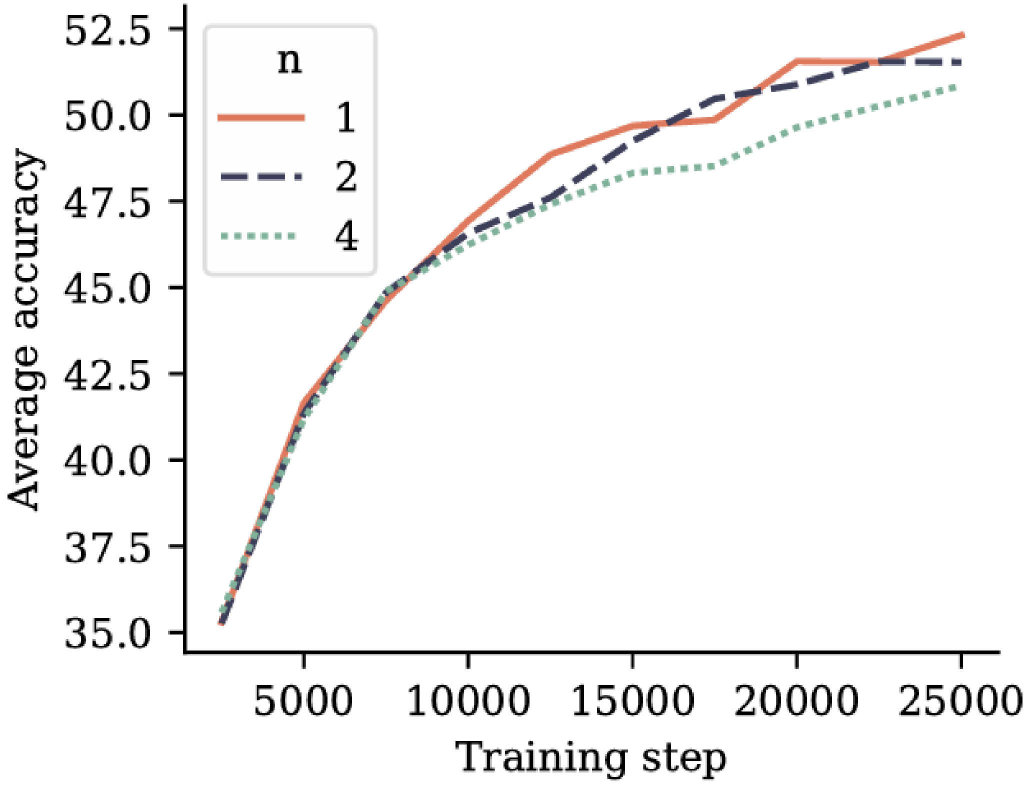


Figure 5: **Multi-token training with 7B models doesn't improve performance on choice tasks**

Natural language results

- Also tried text summarization, which is more of a generative task
- Tried eight benchmarks that use a form of ROUGE metric
  - ROUGE isn't the best metric, but it's well studied
- Multi-token prediction beats next token consistently
- 4-token is better with more fine-tuning data

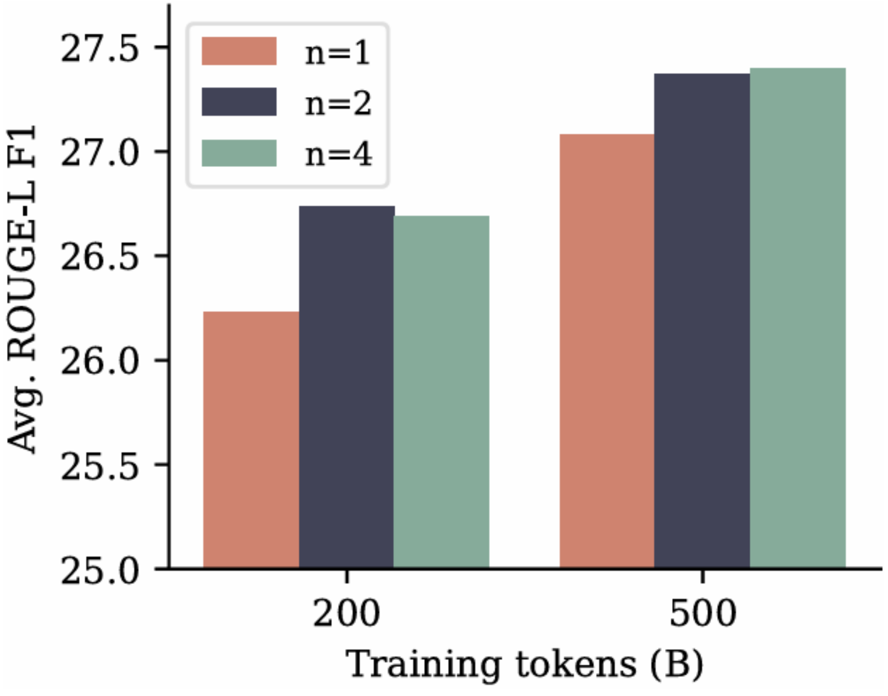


Figure 6: **Performance on abstractive text summarization.** Average ROUGE-L (longest common subsequence overlap) F_1 score for 7B models trained on 200B and 500B tokens of natural language on eight summarization benchmarks.

Synthetic test of induction heads

- Synthetic data to test ability to form induction heads
- Took children's stories, and replaced names with random two-token names
- Predicting first name token requires text context, but the induction heads can see the 1st then predict the 2nd
- Conclude that induction heads get former earlier in training

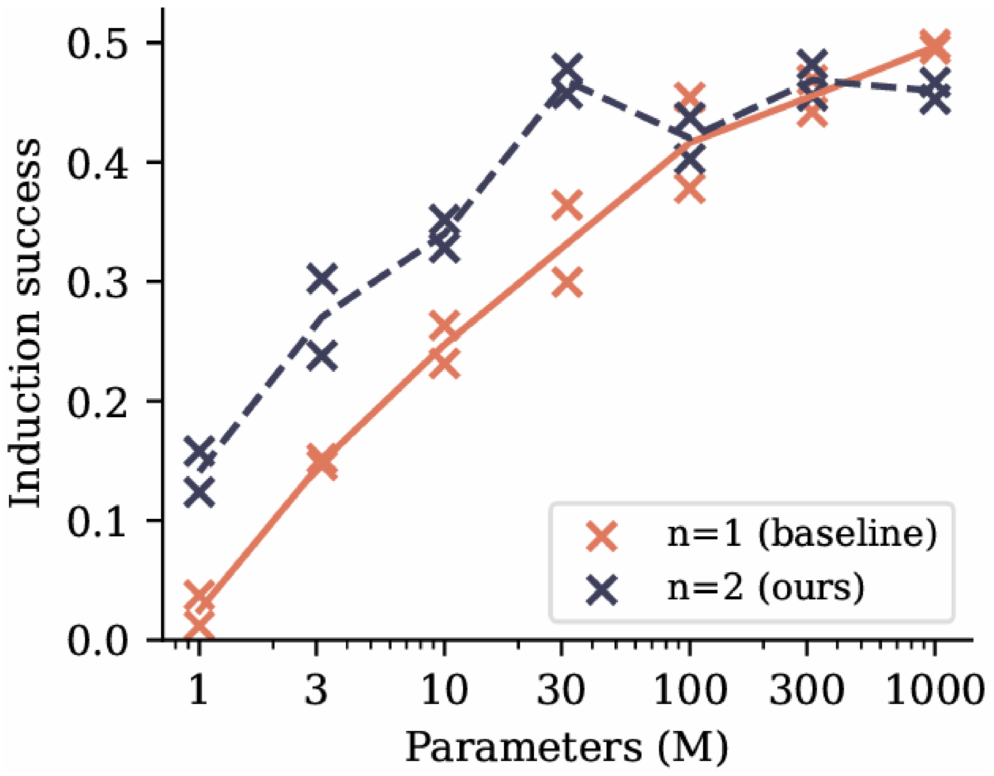


Figure 7: **Induction capability of n-token prediction models.** Shown is accuracy on the second token of two token names that have already been mentioned previously.

Algorithmic reasoning

- Trained small models on a math task on the ring $\mathbb{F}^7[X]/(X^5)$
- Multi-token prediction models did better than next token
- They also did relatively better on problems that were out of domain from what they had been trained on

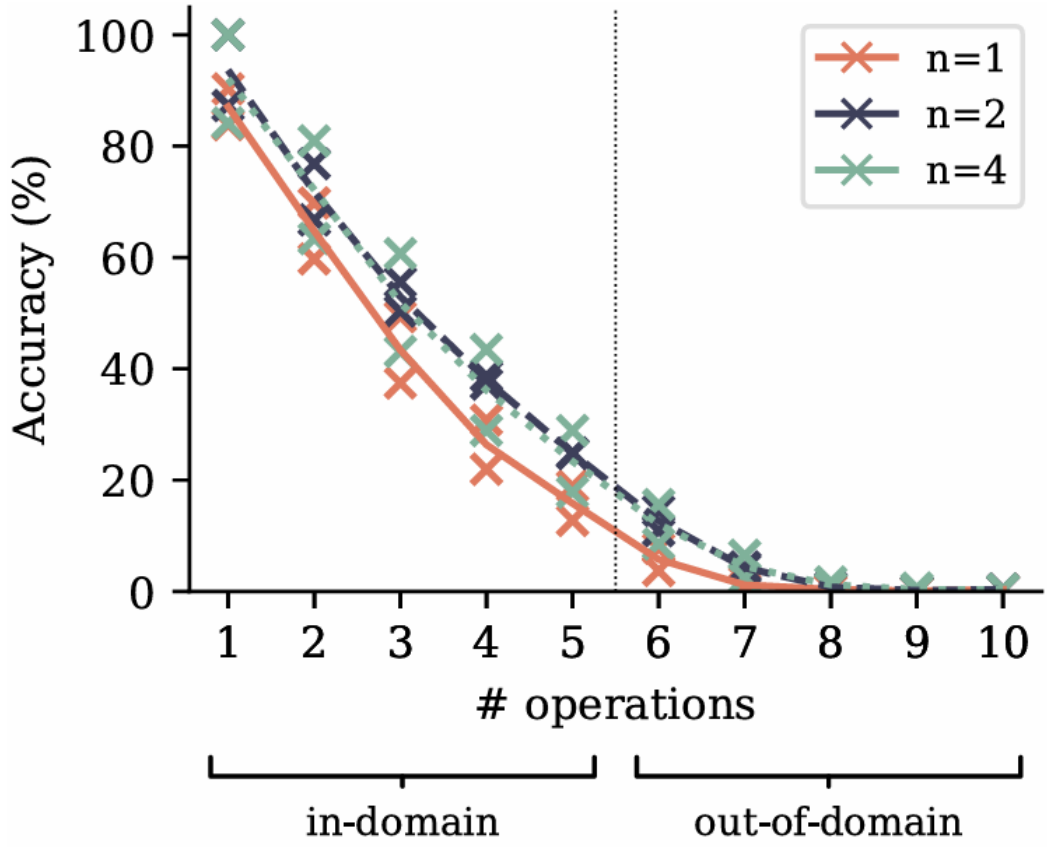


Figure 8: **Accuracy on a polynomial arithmetic task with varying number of operations per expression.**

Faster inference

- Tested self-speculative decoding using the addition prediction heads
- For 4-token prediction got around 3x inference speedup
- Code accepts 2.5 out of 3 suggestions, text was 2.7 out of 3

	Wikipedia		Books		Code	
# Heads used	Rel. speedup	Tokens / forward	Rel. speedup	Tokens / forward	Rel. speedup	Tokens / forward
1	1.00	1.00	1.00	1.00	1.00	1.00
2	1.79	1.88	1.77	1.87	1.85	1.94
3	2.35	2.57	2.32	2.56	2.54	2.78
4	2.74	3.12	2.67	3.09	**3.05**	**3.50**

Multi-token prediction conclusion

- Trained LLMs that predicted multiple tokens in parallel instead of just predicting one single next token
  - Also tried some variants instead of parallel (e.g., causal) but didn't pursue
- Multi-token prediction seems to work better for complex tasks like generating code and when reasoning required
  - Didn't work well for multiple choice benchmarks
- Some speculation about earlier creation of induction heads and domain shift from training with teacher forcing to real generation
- Worked better for larger models up to 13B, and would be interesting to see further analysis for larger, top of the line LLMs
  - Authors mention experiments such as vocab size, but I'd like to see expanded model dimensions

References

- Blockwise Parallel Decoding for Deep Autoregressive Models
  Mitchell Stern et al. (2018)
  https://arxiv.org/abs/1811.03115
- In-Context Learning and Induction Heads
  Catherine Olsson et al. (2022)
  https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html

Addendum – Speculative decoding

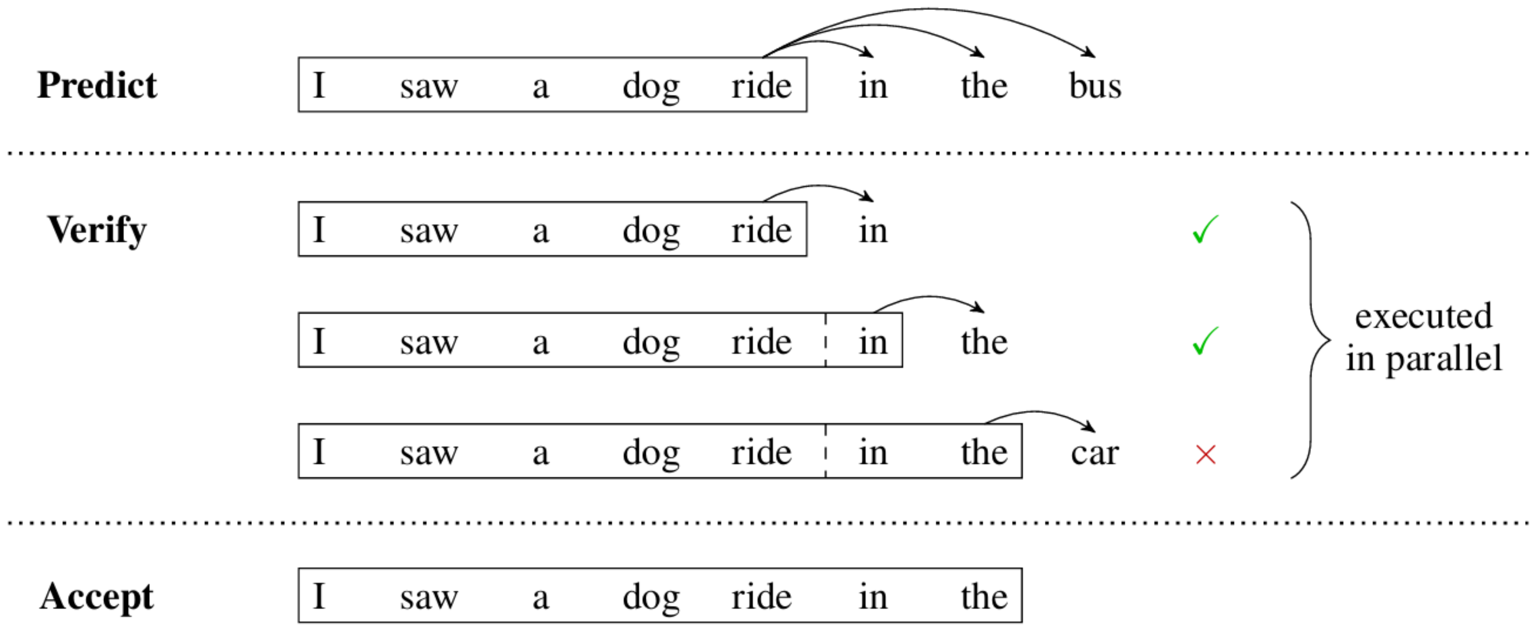


Figure 1: The three substeps of blockwise parallel decoding. In the **predict** substep, the greedy model and two proposal models independently and in parallel predict "in", "the", and "bus". In the **verify** substep, the greedy model scores each of the three independent predictions, conditioning on the previous independent predictions where applicable. When using a Transformer or convolutional sequence-to-sequence model, these three computations can be done in parallel. The highest-probability prediction for the third position is "car", which differs from the independently predicted "bus". In the **accept** substep, $\hat{y}$ is hence extended to include only "in" and "the" before making the next k independent predictions.