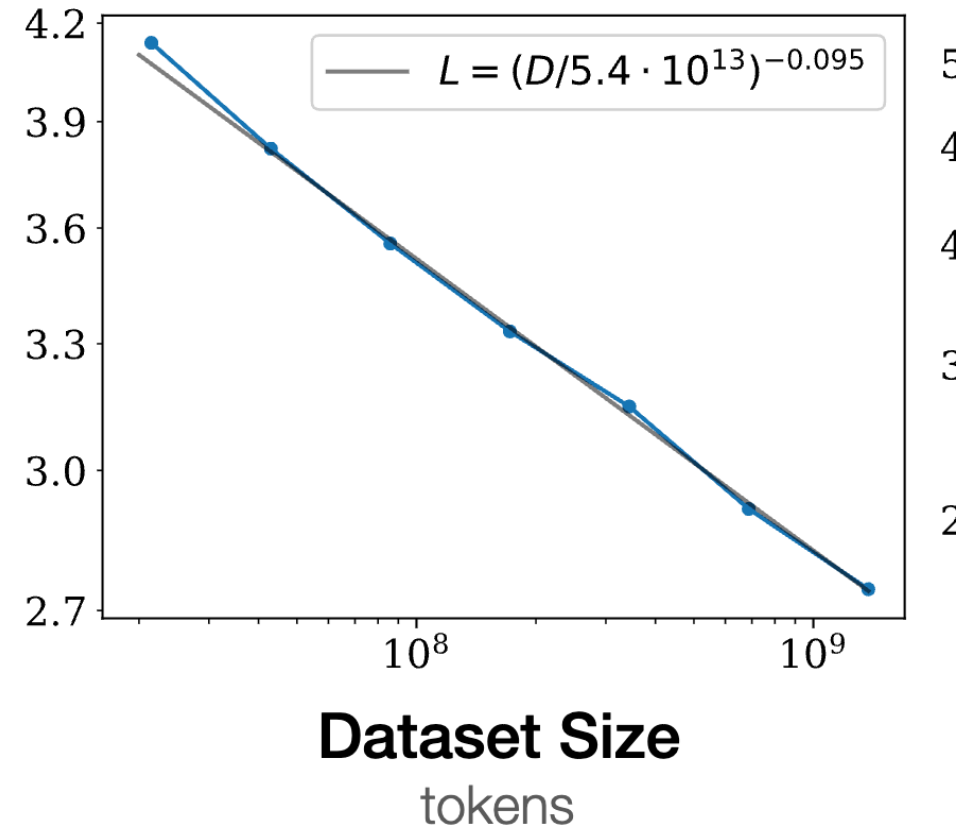
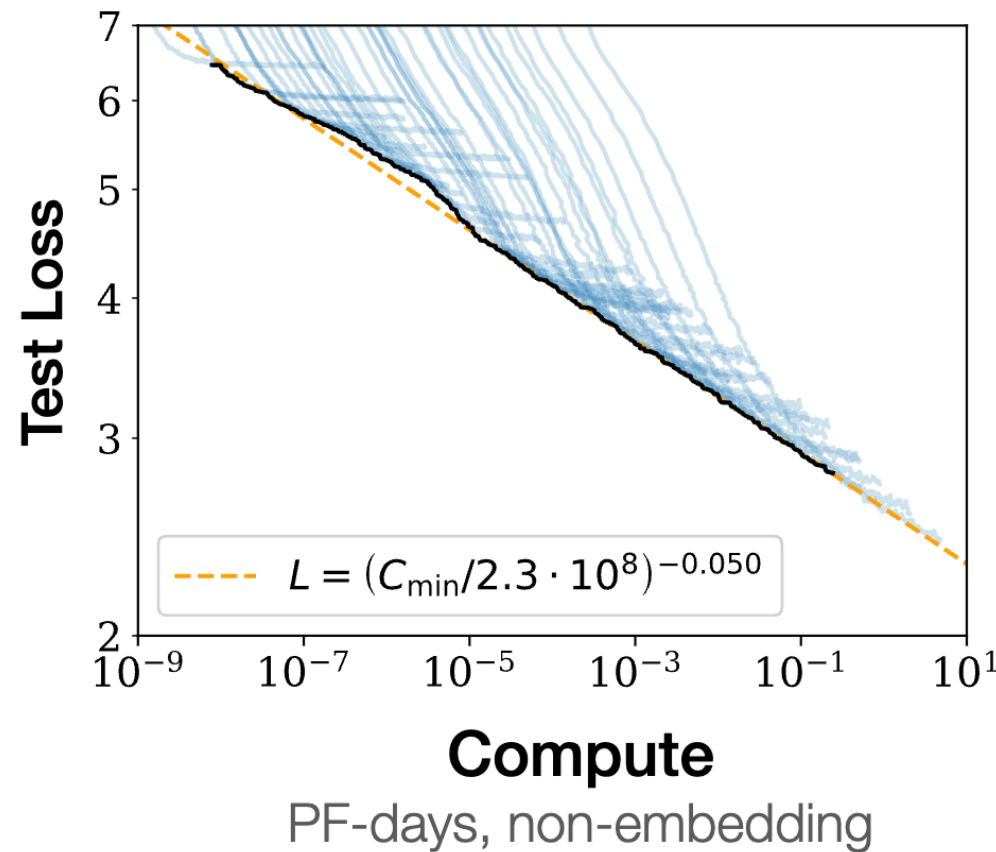


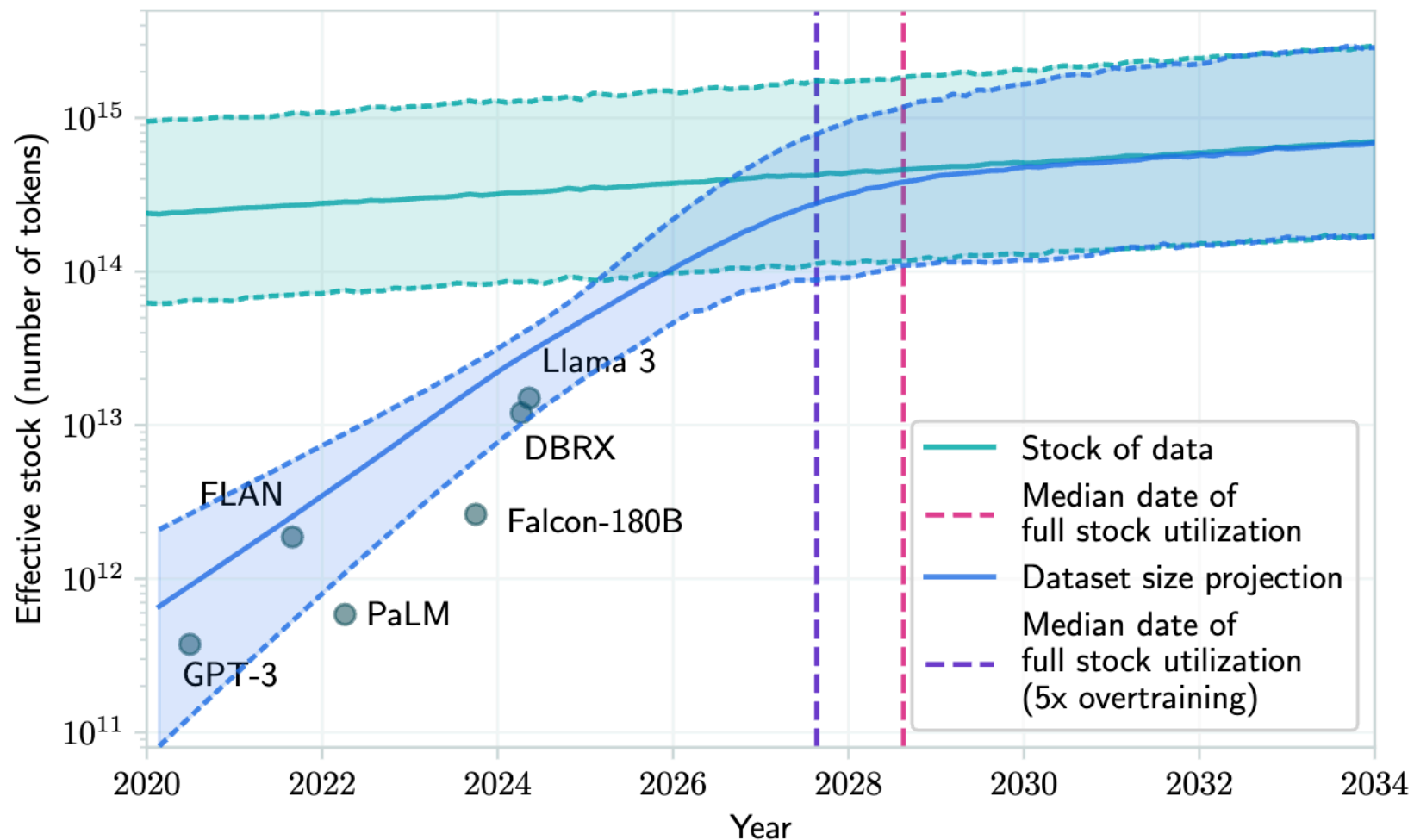
Progress in AI has been due to scaling 2 things:



# While compute is growing data is not growing!



# We will loose out of unique data by 2028



We need more data-efficient algorithms to keep the scaling trend going..



# Primarily there have been 2 algorithms for scaling

Autoregressive objective proposed in Text: GPT 2

Diffusion objective proposed  
in Vision- DDPM

---

## Language Models are Unsupervised Multitask Learners

---

Alec Radford <sup>\*1</sup> Jeffrey Wu <sup>\*1</sup> Rewon Child <sup>1</sup> David Luan <sup>1</sup> Dario Amodei <sup>\*\*1</sup> Ilya Sutskever <sup>\*\*1</sup>

### Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically

competent generalists. We would like to move towards more general systems which can perform many tasks – eventually without the need to manually create and label a training dataset for each one.

---

## Denoising Diffusion Probabilistic Models

---

Jonathan Ho  
UC Berkeley

jonathanho@berkeley.edu

Ajay Jain  
UC Berkeley

ajayj@berkeley.edu

Pieter Abbeel  
UC Berkeley

pabbeel@cs.berkeley.edu

Both objective maximize joint likelihoods

The only difference between the two is the factorization of the joint distribution!

# Success of both Diffusion and AR has created both excitement and confusion.

Language community is exploring diffusion on text.

---

## Structured Denoising Diffusion Models in Discrete State-Spaces

---

Jacob Austin\*, Daniel D. Johnson\*, Jonathan Ho, Daniel Tarlow & Rianne van den Berg<sup>†</sup>  
Google Research, Brain Team  
{jaaustin, ddjohnson, jonathanho, dtarlow, riannevdberg}@google.com

D3PM introduced discrete diffusion on text, that does forward diffusion via random masking

---

## Likelihood-Based Diffusion Language Models

---

Ishaan Gulrajani  
Stanford University  
igul222@gmail.com

Tatsunori B. Hashimoto  
Stanford University  
thashim@stanford.edu

PLAID applied commonly used continuous diffusion (via gaussian noise) on text, by first projecting discrete text tokens first to embeddings and then doing standard diffusion.

# Success of both Diffusion and AR has created both excitement and confusion.

Lot more papers doing diffusion on text:

- **Diffusion-LM** – *Diffusion Improves Controllable Text Generation* (NeurIPS 2022)
- **AR-Diffusion** – *Auto-Regressive Diffusion for Text Generation* (NeurIPS 2023)
- **SEDD** – *Score Entropy Discrete Diffusion* (arXiv 2024, analysis paper)
- **DiffuSeq** – *Sequence-to-Sequence Text Generation with Diffusion Models* (ICLR 2023)
- **SeqDiffuSeq** – *Sequence-to-Sequence Diffusion Model with Self-Conditioning* (NAACL 2024)
- **LLaDA** – *Large Language Diffusion Model from Scratch* (arXiv 2025)
- **DiffusionBERT** – *Improving Generative Masked LMs with Diffusion* (arXiv 2022)
- **CodeFusion** – [A Pre-trained Diffusion Model for Code Generation](#) (EMNLP 2023)
- **Block Diffusion**: *Interpolating Between Autoregressive and Diffusion Language Models* (ICLR 2025)
- **Bit Diffusion** – *Analog Bits: Generating Discrete Data using Diffusion Models* (ICLR 2022)
- **RDLM** – *Continuous Diffusion Model for Language Modeling* (arXiv 2025)
- **EDLM** – *Energy-Based Diffusion Language Models for Text Generation* (arXiv 2024)

Success of both Diffusion and AR has created both excitement and confusion.

Similarly Vision community is exploring Autoregressive on Images.

---

### Scaling Autoregressive Models for Content-Rich Text-to-Image Generation

---

Jiahui Yu\* Yuanzhong Xu<sup>†</sup> Jing Yu Koh<sup>†</sup> Thang Luong<sup>†</sup> Gunjan Baid<sup>†</sup>  
Zirui Wang<sup>†</sup> Vijay Vasudevan<sup>†</sup> Alexander Ku<sup>†</sup>  
Yinfei Yang Burcu Karagol Ayan Ben Hutchinson  
Wei Han Zarana Parekh Xin Li Han Zhang  
Jason Baldridge<sup>†</sup> Yonghui Wu\*

PARTI trained a foundational autoregressive model for image generation objective.

---

### Visual Autoregressive Modeling: Scalable Image Generation via Next-Scale Prediction

---

Keyu Tian<sup>1,2</sup>, Yi Jiang<sup>2,†</sup>, Zehuan Yuan<sup>2,\*</sup>, Bingyue Peng<sup>2</sup>, Liwei Wang<sup>1,\*</sup>  
<sup>1</sup>Peking University <sup>2</sup>Bytedance Inc  
keyutian@stu.pku.edu.cn, jiangyi.enjoy@bytedance.com,  
yuanzehuan@bytedance.com, bingyue.peng@bytedance.com, wanglw@pku.edu.cn

VAR introduced scale Autoregressive and achieved state-of-the-art on image generation benchmarks.

# Success of both Diffusion and AR has created both excitement and confusion.

Lot more papers doing Autoregressive on images:

- **ImageGPT** – *Generative Pretraining from Pixels* (OpenAI, 2020)
- **RQ-Transformer** – *Residual Quantization for Scalable Autoregressive Image Modeling* (CVPR 2022)
- **MQ-VAE + Stackformer** – *Masked Vector Quantization for Fast and High-Fidelity AR Models* (CVPR 2023)
- **NFIG** – *Next-Frequency Image Generation* (arXiv 2025)
- **CART** – *Compositional Auto-Regressive Transformer* (arXiv 2024)
- **DALL-E** – *Zero-Shot Text-to-Image Generation* (arXiv 2021)
- **LlamaGen** – *Large Language Model-based Autoregressive Image Generation* (arXiv 2024)
- **RAR** – *Randomized AutoRegressive Models for Bidirectional Visual Generation* (arXiv 2024)
- **Emu3** – *Next-Token Prediction is All You Need* (arXiv 2024)
- **CAR** – *Controllable Autoregressive Modeling for Visual Generation* (ICLR 2025)
- **Chameleon** – *Mixed-Modal Early-Fusion Foundation Models* (arXiv 2024)
- **Anole** – *Open Autoregressive Multimodal Models for Image-Text Generation* (arXiv 2024)

# Similar confusion in Robotics.

## Papers doing Autoregressive

- **FAST**: Efficient Action Tokenization for Vision-Language-Action Models (arXiv 2025)
- **RT-1** – *Robotics Transformer for Real-World Control at Scale* (RSS 2023)
- **ARM4R** – *Pre-training Auto-regressive Robotic Models with 4D Representations* (arXiv 2025)
- **CARP** – *Visuomotor Policy Learning via Coarse-to-Fine Autoregressive Prediction* (arXiv 2024)
- **HMA** – *Heterogeneous Masked Autoregression for modeling action-video dynamics* (arXiv 2024)
- **Decision Transformer** – *Reinforcement Learning via Sequence Modeling* (Chen et al., NeurIPS 2021)
- **ARP** – *Autoregressive Action Sequence Learning for Robotic Manipulation* (Zhang et al., CoRL 2024)
- **Q-Transformer** – *Scalable Offline Reinforcement Learning via Autoregressive Q-Functions* (Chebotar et al., CoRL 2023)

## Papers doing Diffusion

- **Diffusion Policy** – *Visuomotor Policy Learning via Action Diffusion* (Chi et al., RSS 2023)
- **3D Diffuser Actor** – *Policy Diffusion with 3D Scene Representations* (Ke et al., CoRL 2024)
- **3D Diffusion Policy** – *Generalizable Visuomotor Policy Learning via Simple 3D Representations* (Ze et al., RSS 2024)
- **ChainedDiffuser** – *Unifying Trajectory Diffusion and Keypose Prediction for Robotic Manipulation* (Zhou et al., CoRL 2024)
- **SkillDiffuser** – *Interpretable Hierarchical Planning via Skill Abstractions* (Liang et al., CVPR 2024)
- **DiffusionVLA** – *Scaling Robot Foundation Models via Unified Diffusion and Autoregression* (Wang et al., 2024)
- **DiffuserLite** – *Towards Real-time Diffusion Planning for Robot Manipulation* (Liu et al., 2024)
- **ContactDiffusion** – *Learning Contact-Rich Manipulation via Diffusion Models* (Simeonov et al., 2024)

**Background:** How Autoregressive LLMs work -

$$p_{\text{AR}}(x_1, \dots, x_L) = \prod_{t=1}^L p(x_t \mid x_1, \dots, x_{t-1})$$

They model data distribution in a left-to-right manner

## Background: How Diffusion LLMs work -

Forward Noising Process for Continuous Diffusion:

$$q(\tilde{x}_t \mid x) = \mathcal{N} \left( \tilde{x}_t; \sqrt{1 - \gamma_t} x, \gamma_t \cdot \mathbf{I} \right)$$

Forward Noising Process for Discrete Diffusion:

$$q(\tilde{x}_t \mid x) = \prod_{i=1}^L [\gamma_t \cdot \mathbb{I}(\tilde{x}_{t,i} = [\text{MASK}]) + (1 - \gamma_t) \cdot \mathbb{I}(\tilde{x}_{t,i} = x_i)]$$

Objective for Discrete Diffusion:

$$\mathcal{L}_{\text{Diffusion}} = -\mathbb{E}_r \mathbb{E}_{\tilde{x} \sim q_r} \frac{1}{r} \sum_{i \in \mathcal{M}} \log p_{\theta}(x_i \mid \tilde{x}).$$

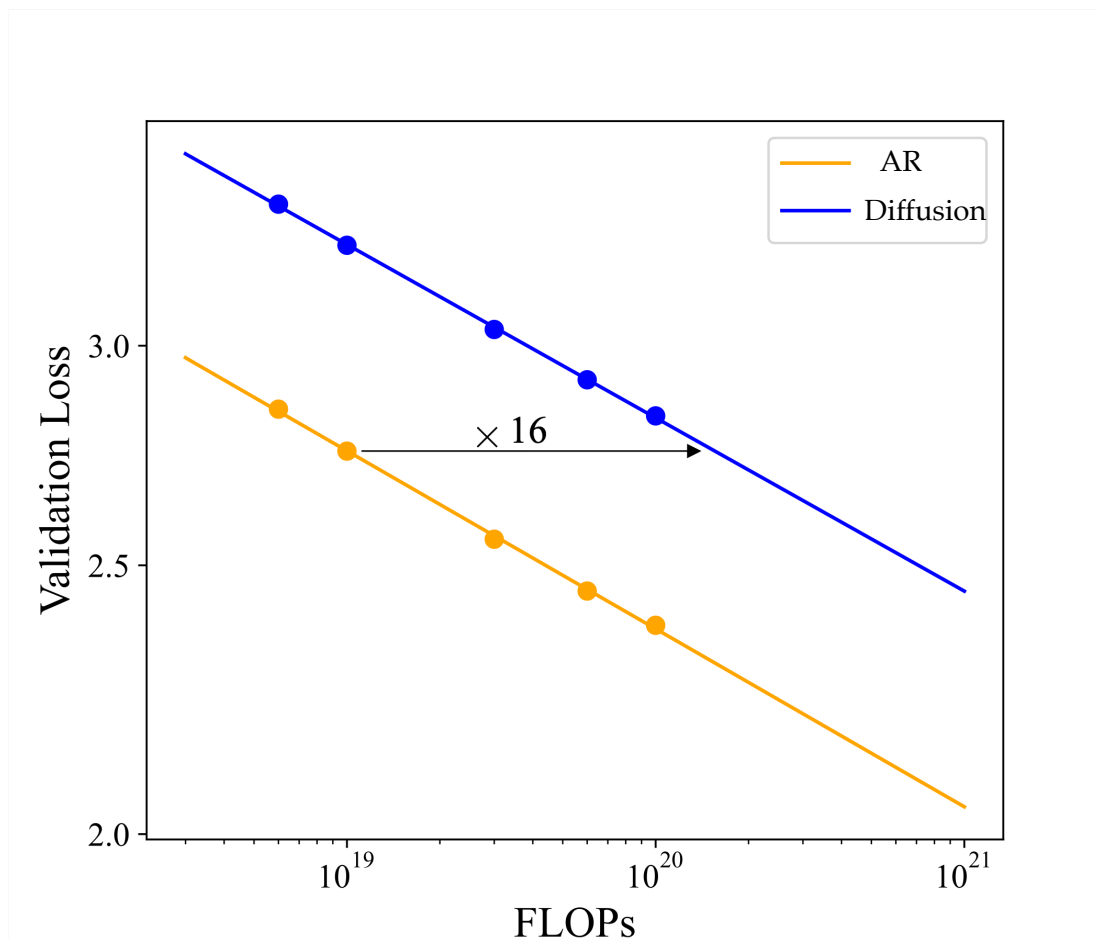


## However Prior methods show worse results for Diffusion on Text

Model Type	Parameters	PPL (↓)
Continuous Diffusion (Diffusion-LM)	80M	118.62
Discrete Diffusion (MDLM)	110M	27.04
Autoregressive (AR)	110M	22.32

- Discrete Diffusion does better than Continuous Diffusion on Text
- Autoregressive still does the best

Prior works show diffusion requires 16x more compute than Autoregressive



Nie et al "Scaling up Masked Diffusion Models on Text"

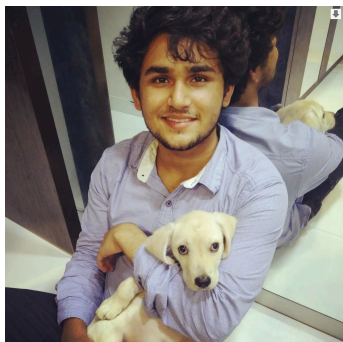
Prior works conflate data and compute in a single plot

Does diffusion require 16x more compute or 16x more data?

We disentangle data and compute —

We study the diffusion and autoregressive models in data-constrained settings

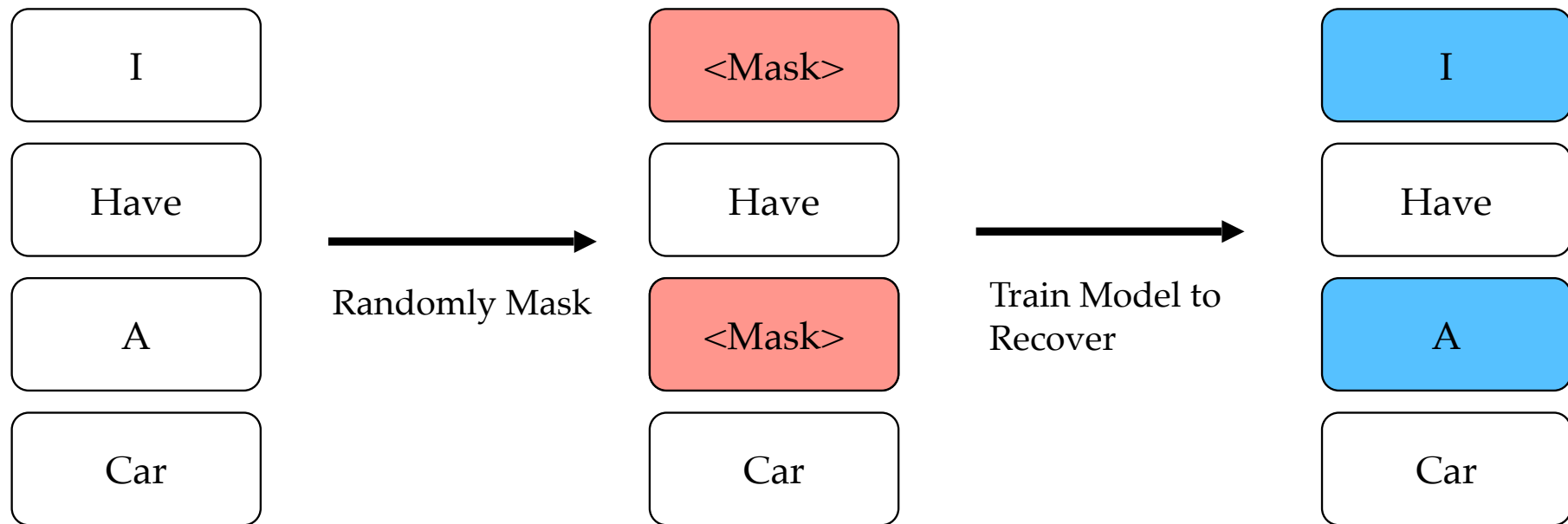
# Diffusion Beats AR in Data-Constrained Settings



Mihir Prabhudesai\*, Mengning Wu\*, Amir Zahed, Katerina Fragkiadaki, Deepak Pathak

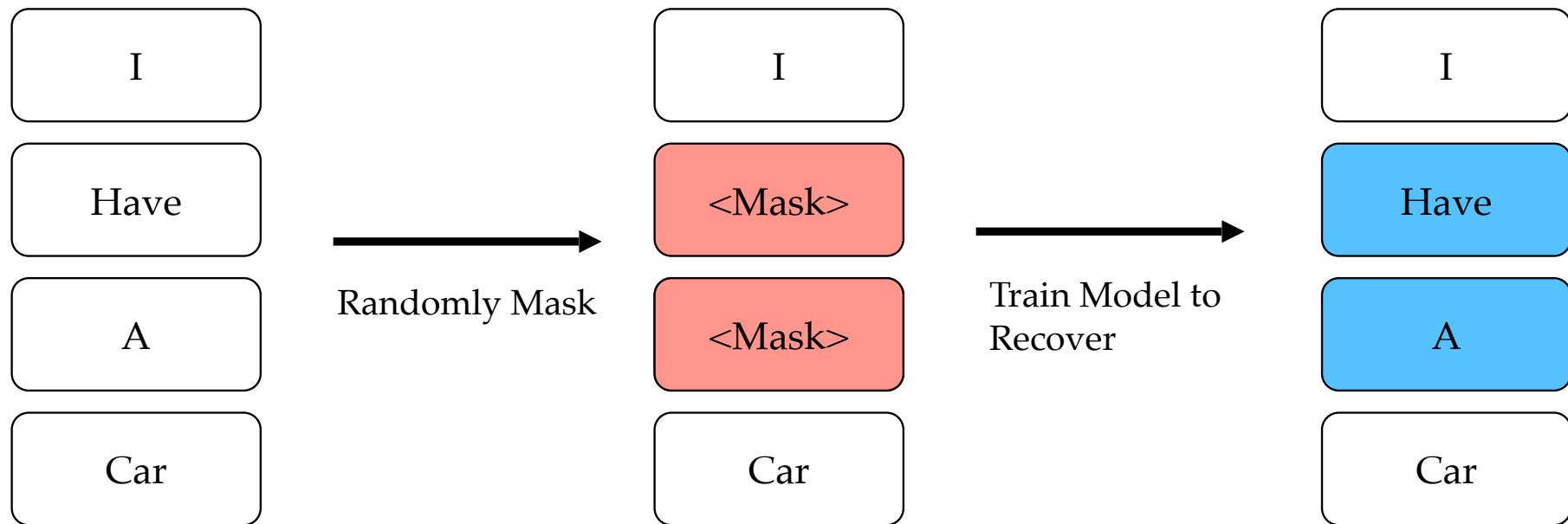
# Our motivation to pursue this project -

Let's go back to diffusion training objective



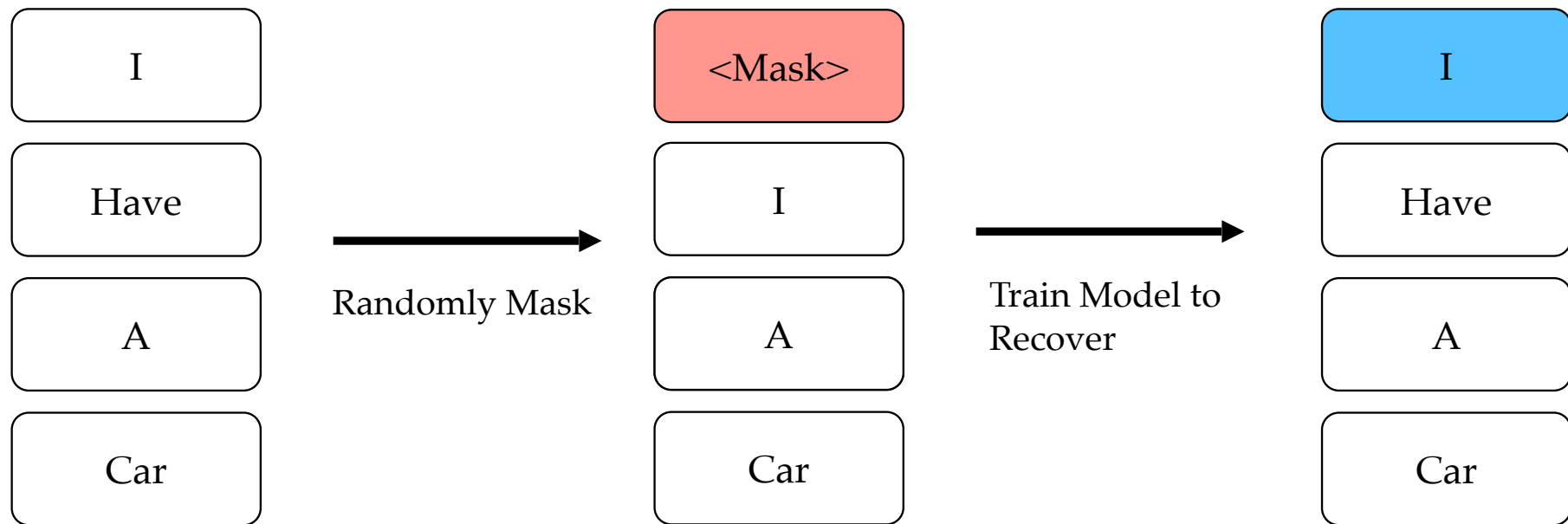
# Our motivation to pursue this project -

Let's go back to diffusion training objective



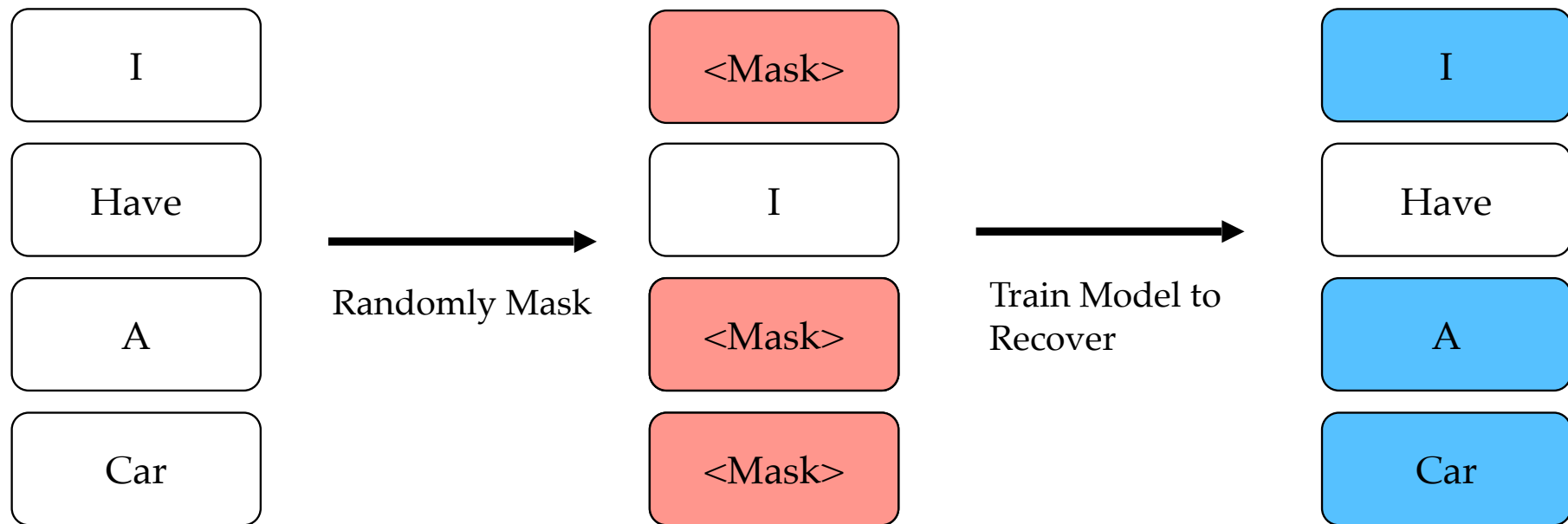
# Our motivation to pursue this project -

Let's go back to diffusion training objective



# Our motivation to pursue this project -

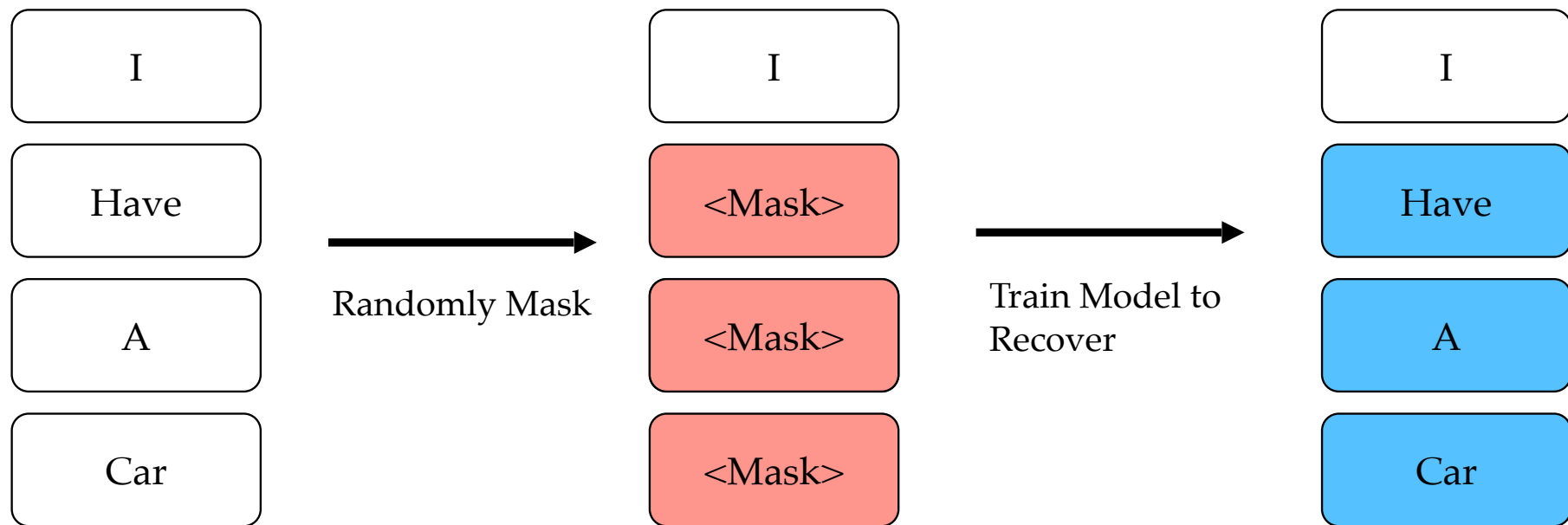
Let's go back to diffusion training objective





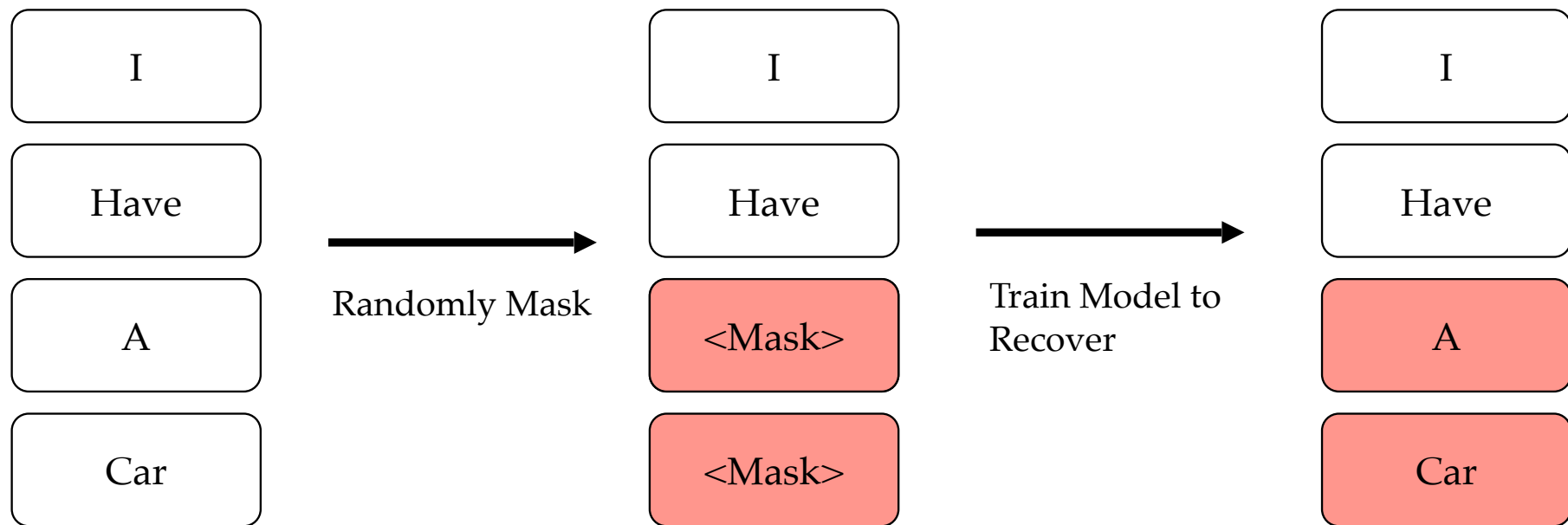
# Our motivation to pursue this project -

Interestingly left-to-right masking gets included in diffusion training



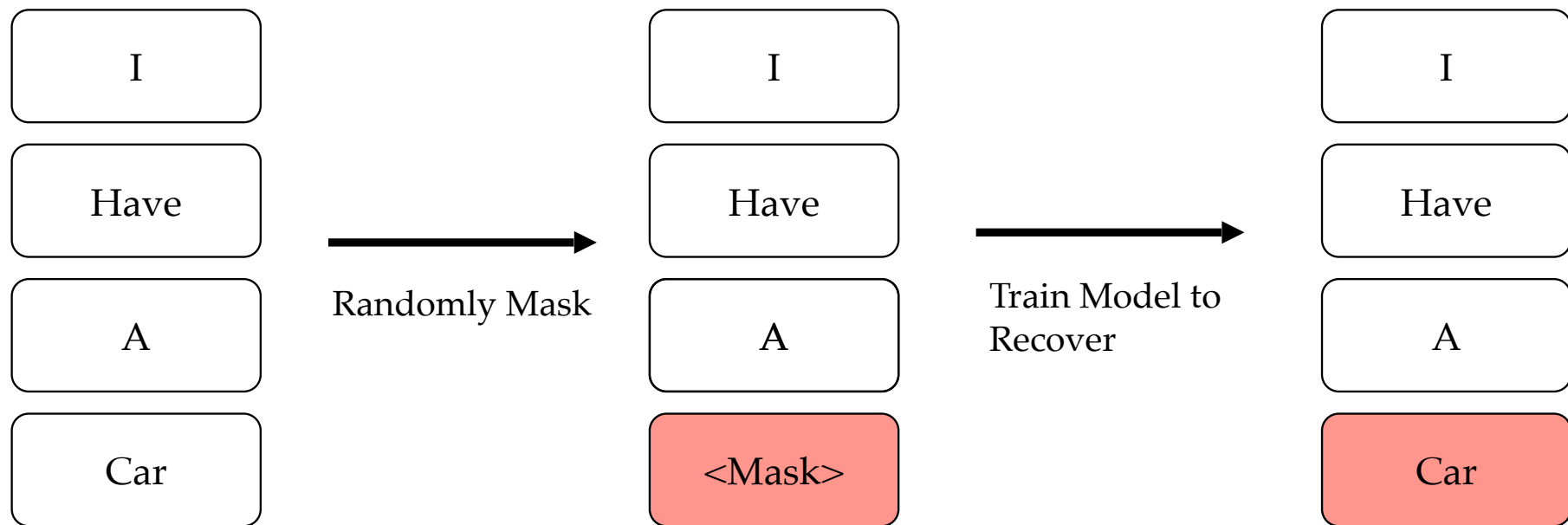
# Our motivation to pursue this project -

Interestingly left-to-right masking gets included in diffusion training



# Our motivation to pursue this project -

Interestingly left-to-right masking gets included in diffusion training



# Our hypothesis -

Diffusion could be understood as implicit **data augmentation** on Autoregressive Training

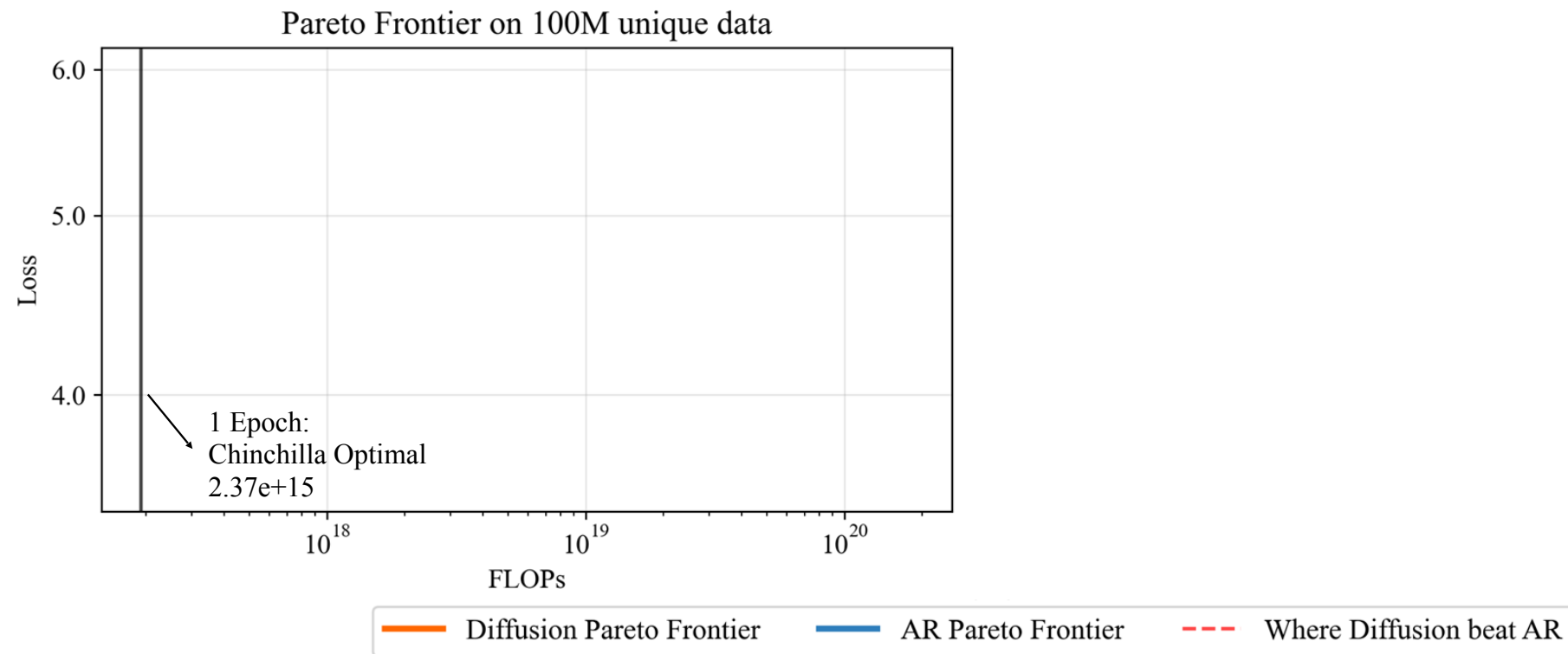
And one should do Data Augmentation only when they are **data bottlenecked!**

Prior works get worse results than Autoregressive, as everyone has been training Diffusion LLMs in **single-epoch settings!**

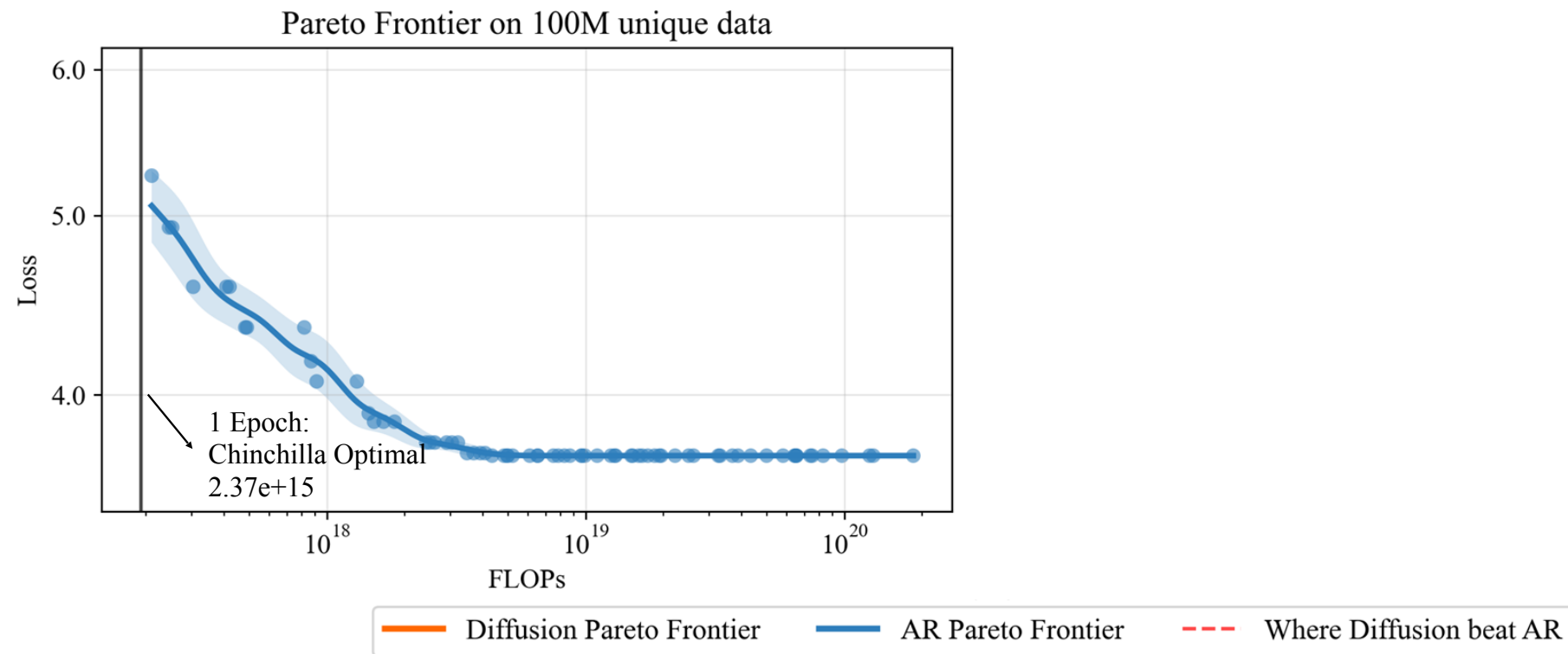
We study Diffusion Models in Data-constrained Settings

We train 200+ Diffusion and AR models at different dataset and Flop budgets

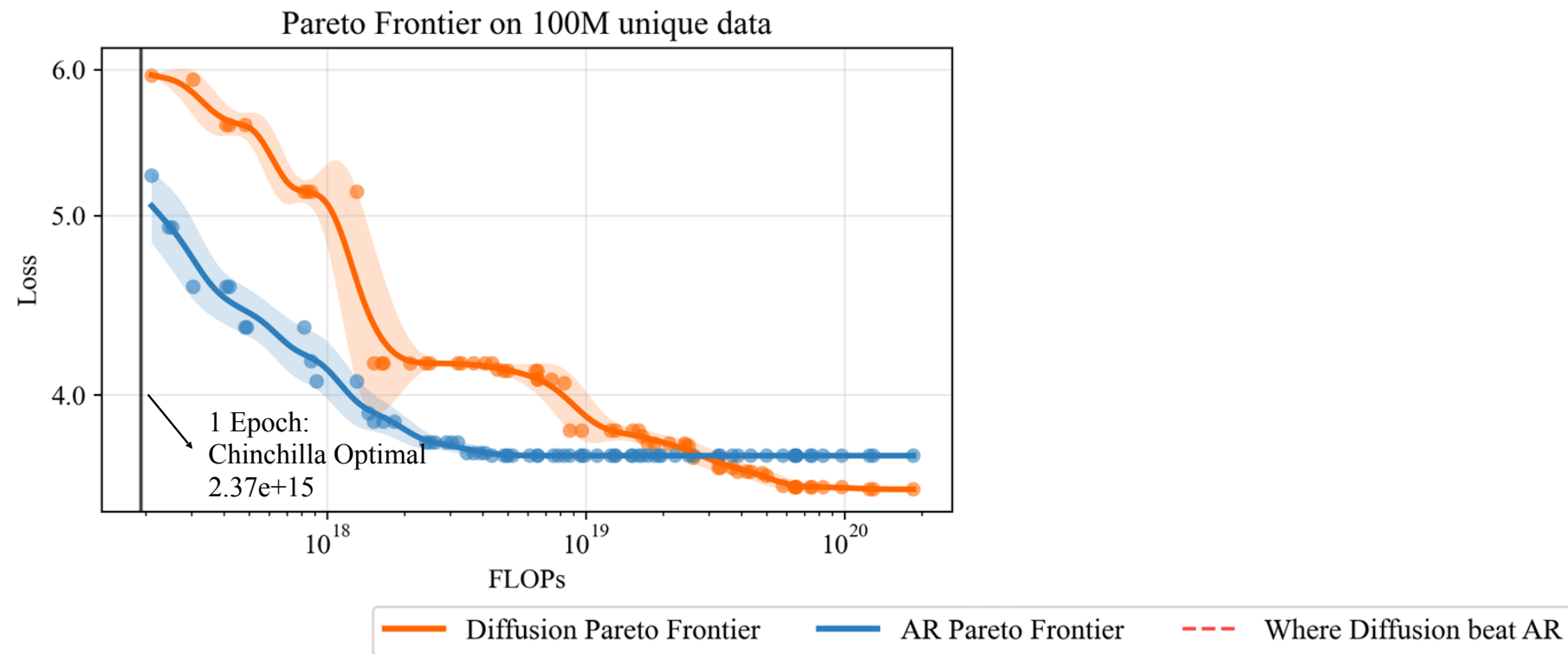
# Pareto optimal tradeoff of validation loss and flops.



# Pareto optimal tradeoff of validation loss and flops.

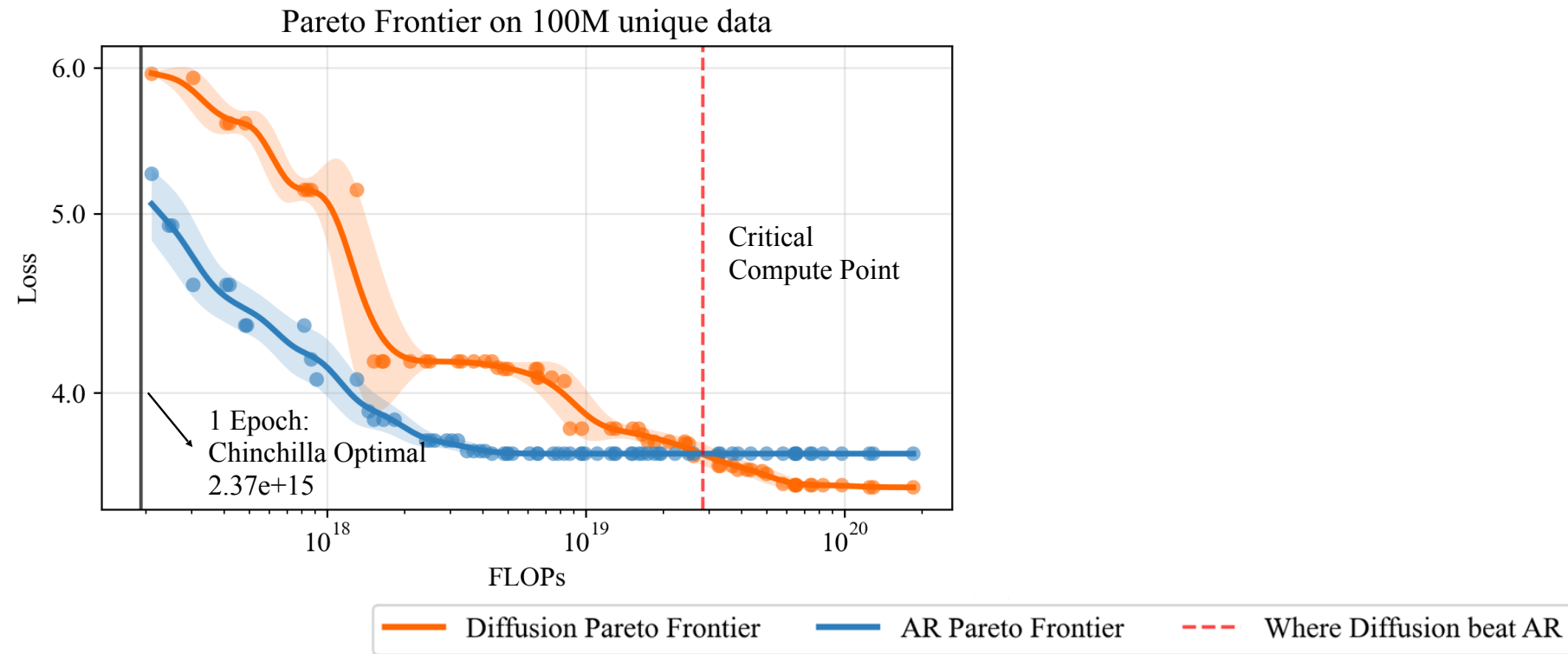


# Pareto optimal tradeoff of validation loss and flops.

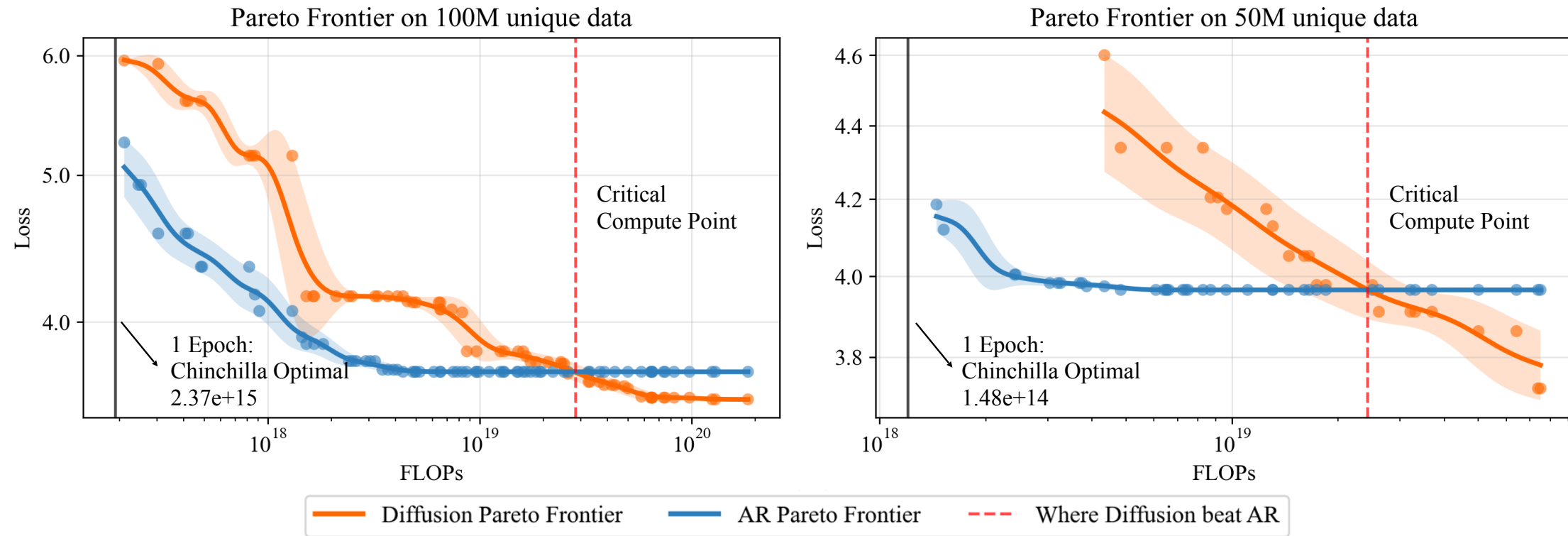




# Pareto optimal tradeoff of validation loss and flops.

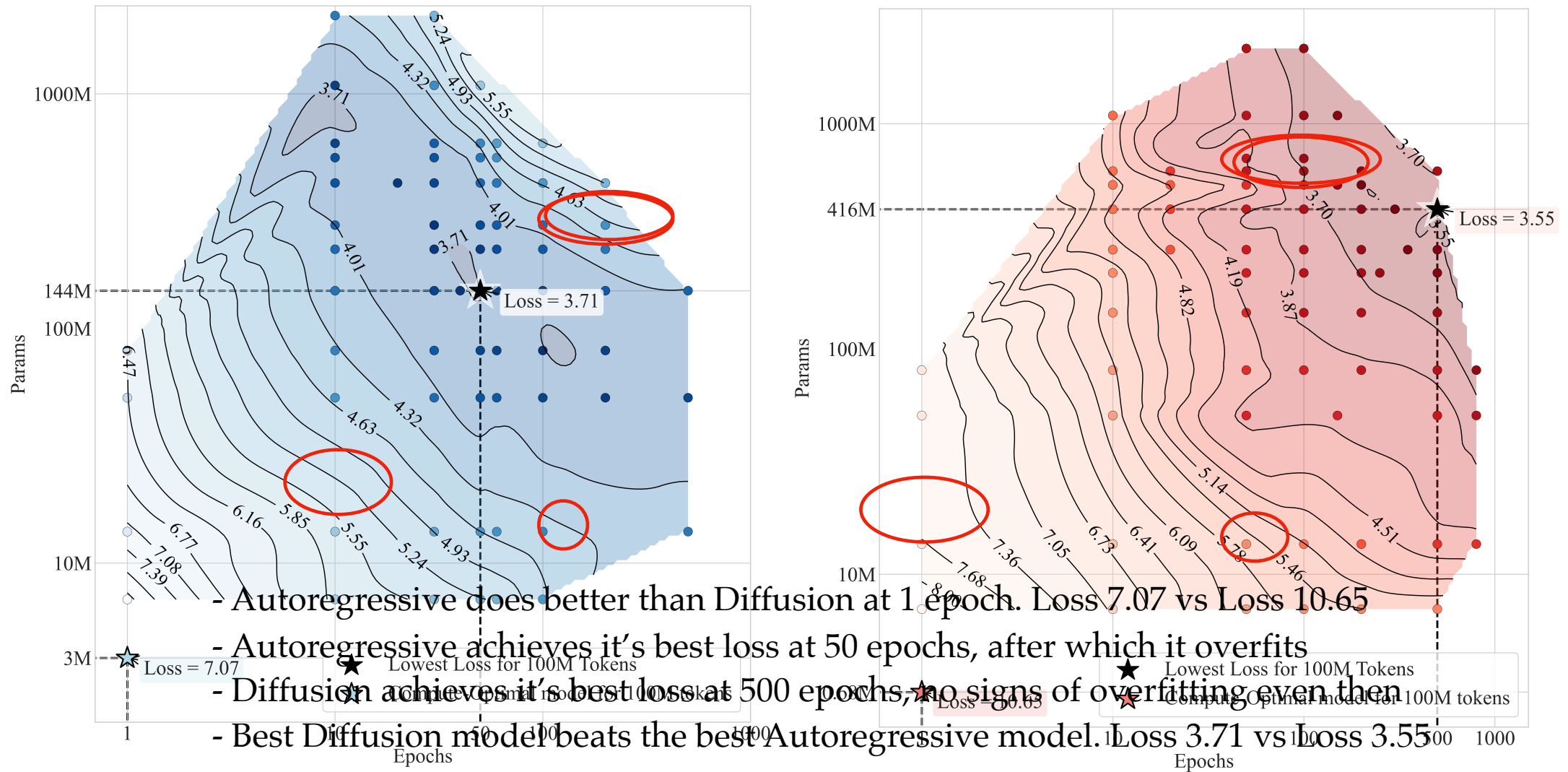


# Pareto optimal tradeoff of validation loss and flops.



- Diffusion Beats AR after a certain number of flops, which is much higher than Chinchilla 1 epoch flop count.

# How compute is distributed over parameters and epochs



We fit data-constrained scaling laws on these models.

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

**Learnt constants**

**Parameters**

**Training Tokens**  
found for AR in single-epoch setting

Villalobos et al “Scaling Laws for Neural Language Models”

Hoffmann et al “Training Compute-Optimal Large Language Models”

We fit data-constrained scaling laws on these models.

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

$$D' = U + (1 - \delta)U + (1 - \delta)^2U + \dots + (1 - \delta)^{R_D}U$$

Effective number  
of Tokens

$$D' = U_D + U_D R_D^* (1 - e^{-\frac{R_D}{R_D^*}})$$

Effective  
number of  
Training Tokens

Half-Life of Data Reuse

found for AR in multi-epoch setting

Muennighoff et al "Scaling Data-Constrained Language Models"

We fit data-constrained scaling laws on these models.

Table 1: Fitting metrics of the scaling law model for Diffusion and AR. Diffusion and AR achieves a stronger fit across both phases.

(a) Initial fit.

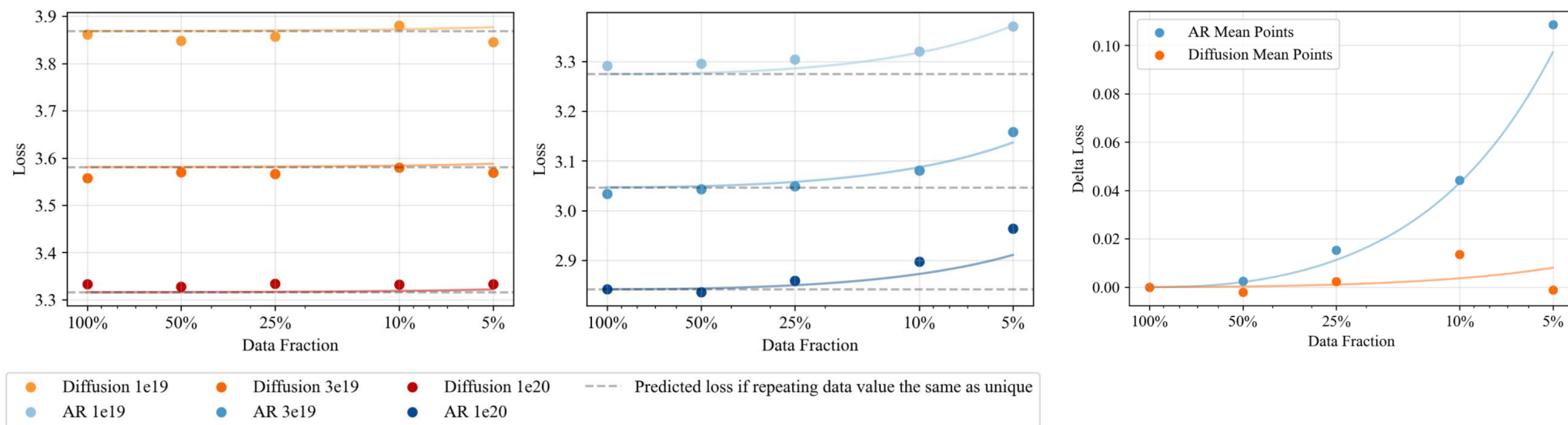
Model	$R^2$	Loss
Diffusion	0.9447	0.0002
AR	0.9439	7.7532e−05

(b) Second step fit with extracted scaling parameters.

Model	$R^2$	Loss	$R_D^*$	$R_N^*$
Diffusion	0.9784	0.00079	493.89	1265.65
AR	0.7628	0.00361	31.19	55.16

Half-life of data-reuse

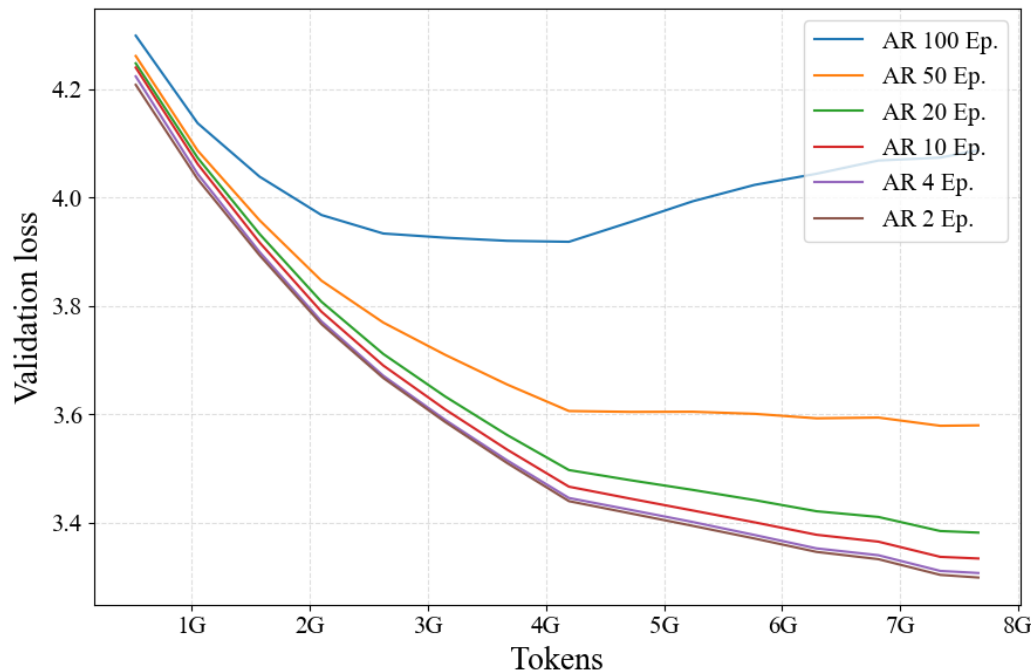
# Decay rate of data value under repetition.



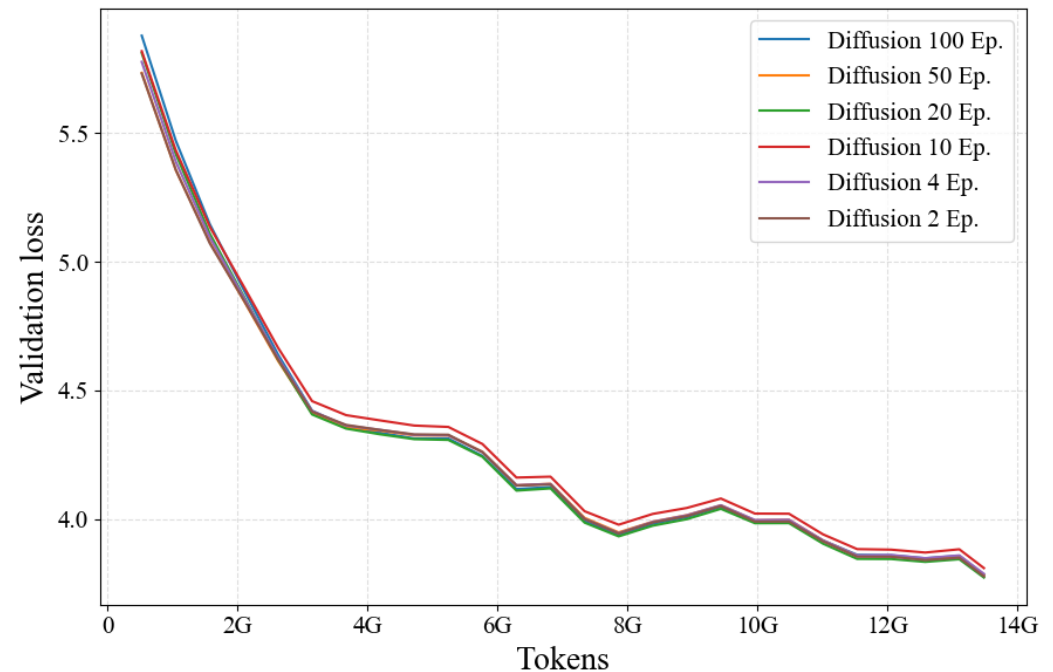
- Fitted scaling law curves (solid line), accurately follows the empirical datapoints
- Diffusion models have a much lower decay rate of data value under repetition.

# Iso-Flop Training curves with different epochs

Autoregressive



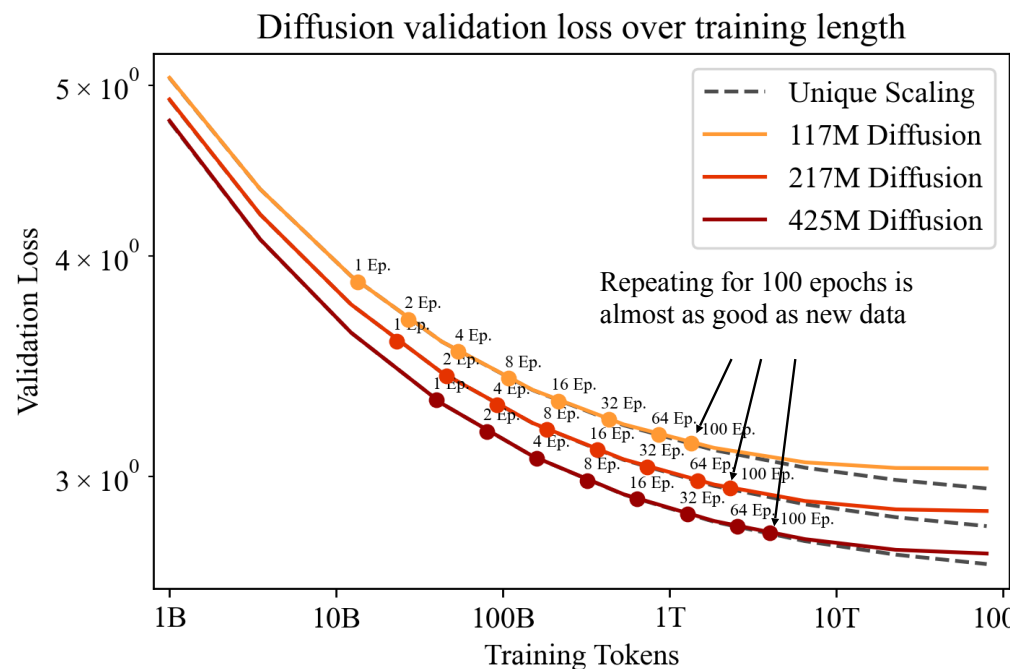
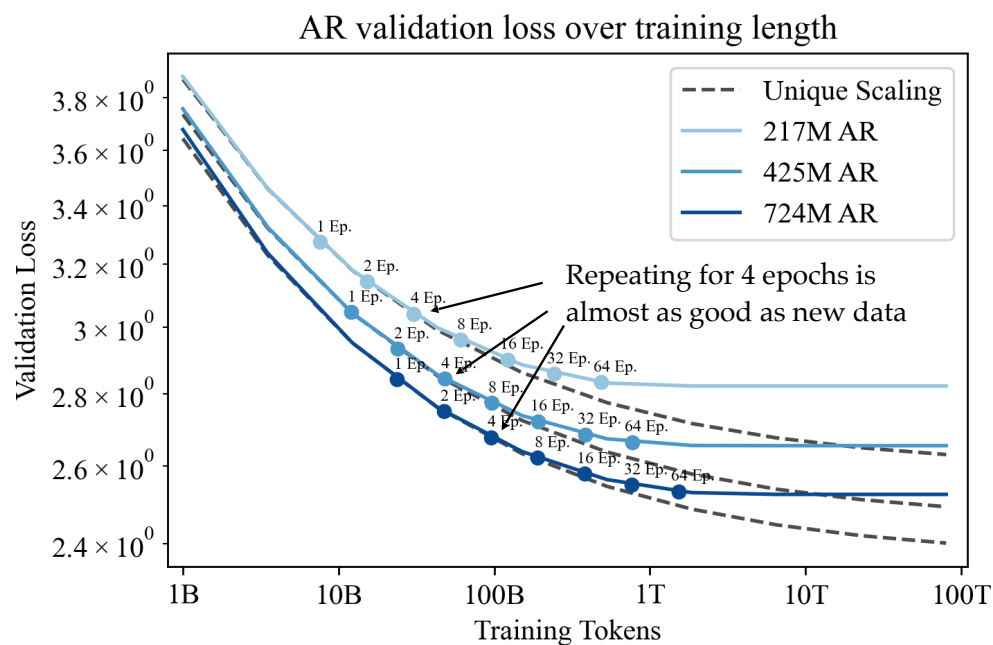
Diffusion



- We maintain the same number of Flops, and train for different epochs and data fractions
- For instance, 2 epoch with 50% unique data or 100 epochs with 1% unique data
- For diffusion, the training curves look similar but for AR there is a meaningful gap



# Extrapolated Scaling laws



- At different Flop counts, AR and Diffusion loss curves, solid Line represents data-constrained setting (multi epoch), while dotted line represents unique data setting (single epoch)
- AR gives the same loss as fresh data until 4 epochs, diffusion gives the same loss as fresh data until 100 epochs.

# Downstream Results

Benchmarks	Random Baseline	100M unique tokens		500M unique tokens	
		AR	Diffusion	AR	Diffusion
ARC-Easy [7]	25.00	35.63	<b>37.84</b>	43.79	<b>45.95</b>
BoolQ [6]	50.00	46.00	<b>49.38</b>	51.87	<b>55.26</b>
COPA [32]	50.00	56.33	<b>59.00</b>	<b>67.00</b>	64.83
HellaSwag [46]	25.00	27.37	<b>30.24</b>	32.28	<b>35.33</b>
PiQA	50.00	<b>60.94</b>	60.72	<b>65.71</b>	65.61
RACE [19]	25.00	25.28	<b>28.96</b>	28.28	<b>31.44</b>
WinoGrande XL [35]	50.00	48.87	<b>50.97</b>	50.61	<b>51.51</b>
SciQ [17]	25.00	58.05	<b>68.67</b>	67.82	<b>79.13</b>
Lambada [29]	00.00	10.91	<b>15.19</b>	15.07	<b>22.30</b>

*Note:* All values represent accuracy (%). Best results shown in bold.

When to use Diffusion over AR?

$$\Delta\mathcal{L}(C, U) = \mathcal{L}_{\text{Diffusion}}(C, U) - \mathcal{L}_{\text{AR}}(C, U),$$

$$\Delta\mathcal{L}(C_{\text{crit}}, U) = 0$$




Solve for  $C_{\text{crit}}$

## When to use Diffusion over AR?

$$\log_{10}(U) = 0.460 \cdot \log_{10}(\text{FLOPs}) - 7.052.$$

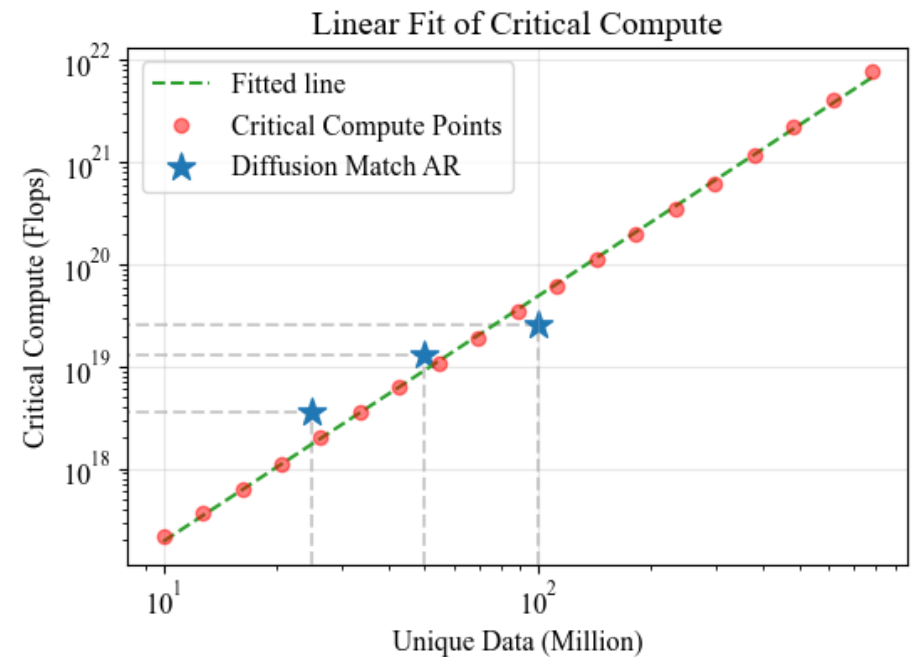
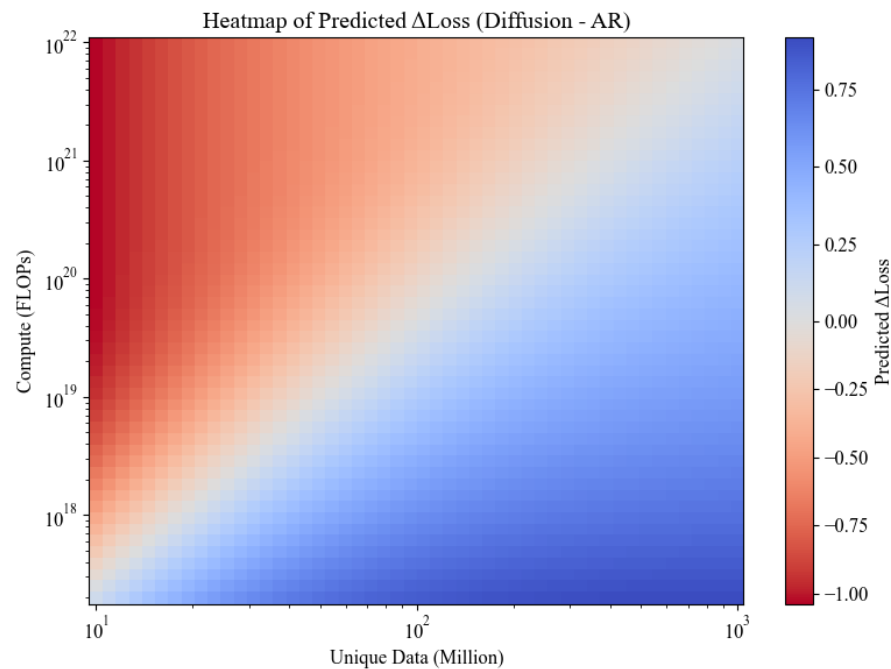
$$C_{\text{crit}}(U) = 10^{\frac{\log_{10}(U) + 7.052}{0.460}} = 2.12 \times 10^{15} \cdot U^{2.174}$$

$$C_{\text{crit}}(U) \propto U^{2.174}.$$



Equation to predict the flop counts after which Diffusion beats AR for any given unique data tokens (U)

# When to use Diffusion over AR?



- For any given number of unique data tokens ( $U$ ), we can predict the exact number of flops (Critical compute point) after which Diffusion beats AR
- We find the critical compute point has a log linear relationship with the number of unique tokens

## Conclusion

If you are compute bottlenecked  $\rightarrow$  AR.

If you are data bottlenecked  $\rightarrow$  Diffusion.

## In Robotics

Simulation Training  $\rightarrow$  AR.

Real world Training  $\rightarrow$  Diffusion.

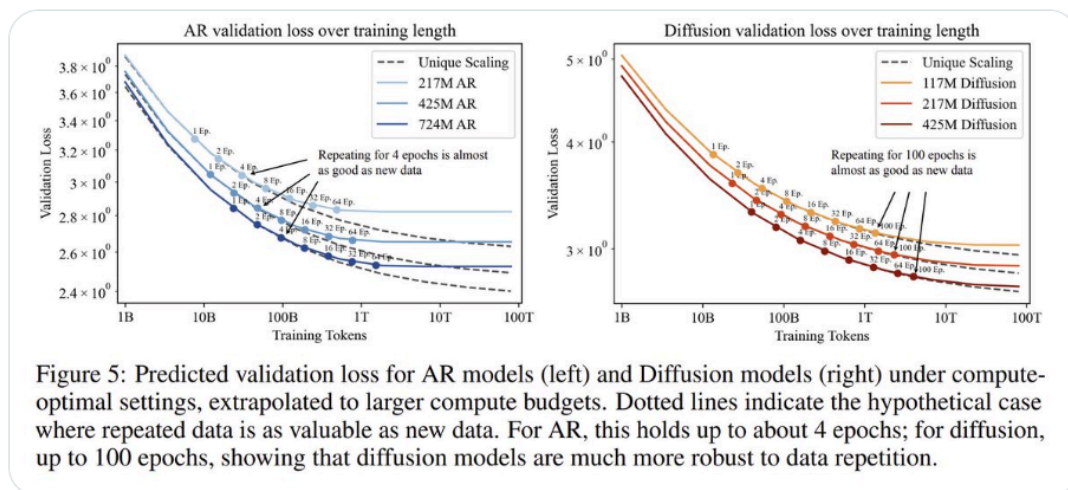
# Twitter Peer Review



Simo Ryu    
@cloneofsimo



Everyone get your top 1% quality dataset and train 100 epochs right now



2:40 AM · Jul 22, 2025 · 124.7K Views



6



33



475



351







**Lucas Beyer (bl16)**

@giffmana



AKA data augmentation. The numbers actually match my experience exactly. This is something i think LLM people will slowly rediscover from vision people.

Not sure how they can write up the whole paper and not even once think of running the AR with augmentation or dropout?



**Simo Ryu** @cloneofsimon · Jul 22

Everyone get your top 1% quality dataset and train 100 epochs right now

11:46 AM · Jul 22, 2025 · **129.5K** Views



21



34



627



513



# Our rebuttal:

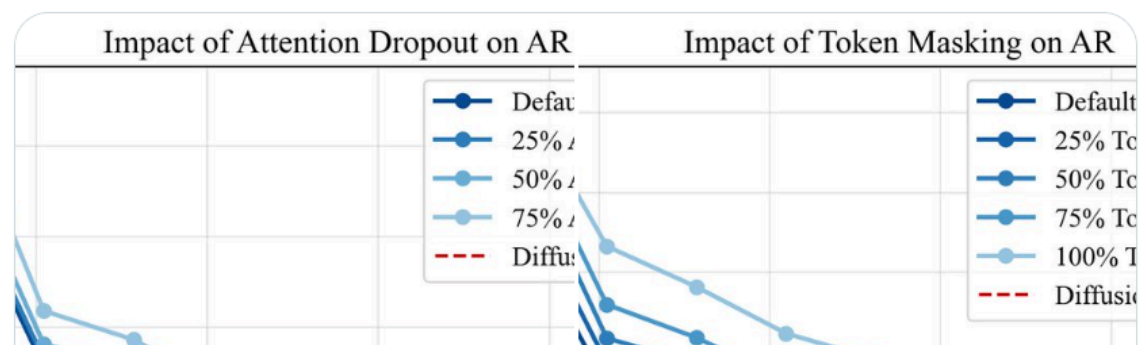


**Mihir Prabhudesai** @mihirp98 · Jul 25



We ran more experiments, with random token masking, and attention dropout in autoregressive training. Consistent with our earlier ablations, we find these augmentations still overfit quite quickly and are still quite behind diffusion models trained for 500+ epochs. Diffusion

[Show more](#)



**Lucas Beyer (bl16)** @giffmana · Jul 22

AKA data augmentation. The numbers actually match my experience exactly. This is something i think LLM people will slowly rediscover from vision people.

Not sure how they can write up the whole paper and not even once ...



6



17



174

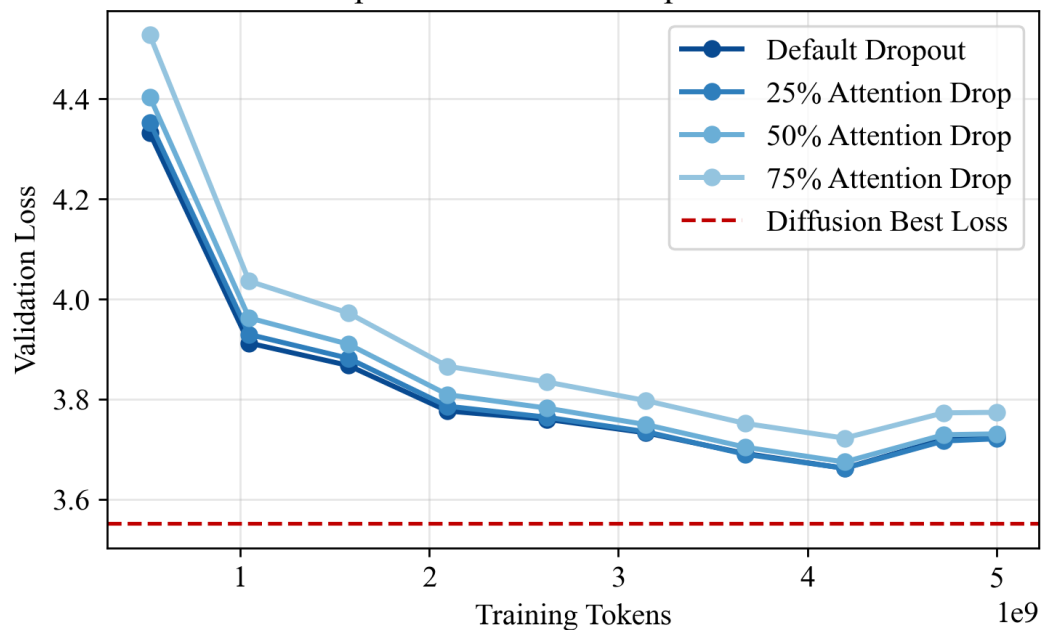


40K

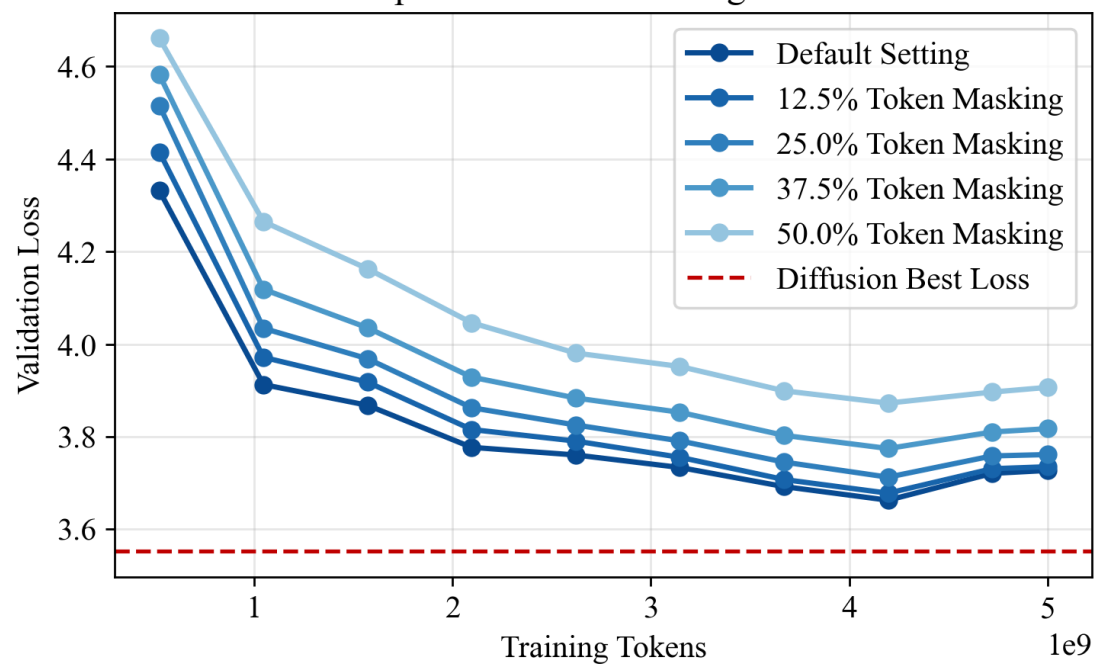


# Our rebuttal:

Impact of Attention Dropout on AR



Impact of Token Masking on AR



# Our rebuttal:



**Mihir Prabhudesai** ✓

@mihirp98



We ran more experiments, with random token masking, and attention dropout in autoregressive training. Consistent with our earlier ablations, we find these augmentations still overfit quite quickly and are still quite behind diffusion models trained for 500+ epochs. Diffusion models randomly factorize the joint, which enables them to generate tokens in random orders, which we think can't simply be recreated by just random input masking while still having the next token prediction objective. We have updated the arxiv with these new results.

We had run some of these experiments before release, however we didn't report them in the paper as they didn't really help us explain the data efficiency of diffusion models (should have included!). We would also like to point out that prior works have also found some of these augmentations to not help much, e.g., [x.com/pratyushmaini/...](https://x.com/pratyushmaini/). They find rephrasing the text to mainly be the augmentation that helps with AR training, however we don't see a reason on why even diffusion models cannot benefit out of this augmentation.

# Lucas replied to our rebuttal:



Lucas Beyer (bl16) ✓  
@giffmana



Interesting, thanks!

So this is your next hypothesis then, right?

> Diffusion models randomly factorize the joint, which enables them to generate tokens in random orders, which we think can't simply be recreated by just random input masking while still having the next token prediction objective

The ideal next step is to design some experiment(s) that can support or disprove this hypothesis.

I maybe have one idea for one, though it's not perfect. You could at the start of training sample  $N$  permutations, and during training use one of them at random for each sequence.  $N=1$  would just be regular AR training.  $N=2$  would be training a mixed forward and backwards AR LM. And then you can continue increasing  $N$  and make a plot. I guess one question would be whether at eval time to also randomize which permutation to use, or ensemble them.

This experiment would add your hypothesis to the AR model, so if your hypothesis for the reason is correct performance should improve at least a bit.

I've actually seen something close to this with AR models before: i had a severely overfitting (many epochs) AR model on task A, then i add a completely unrelated task B and see performance on A increasing, section 5.4 and especially 5.4.4 in [arxiv.org/pdf/2303.17376](https://arxiv.org/pdf/2303.17376). So I'm genuinely curious about this experiment

Another way would be to try an experiment that removes the hypothesis from the diffusion model and see if it deteriorates, though i don't have a good idea without making it ridiculous.

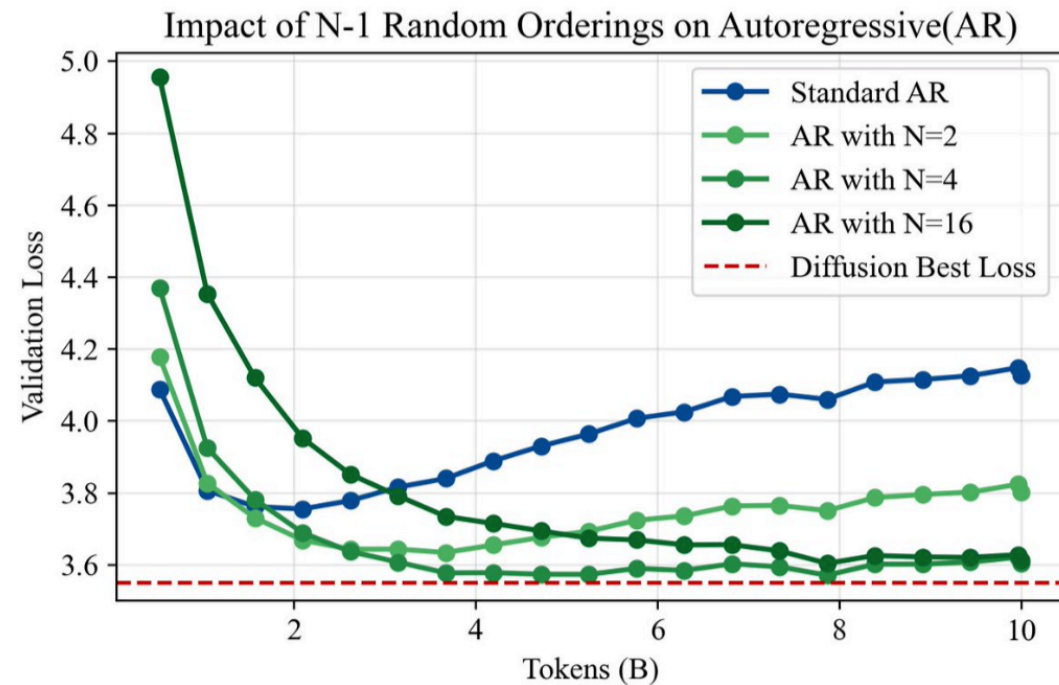
# We tried this experiment out:



**Mihir Prabhudesai** ✓ @mihirp98 · Aug 6

We ran more experiments to better understand “why” diffusion models do better in data-constrained settings than autoregressive. Our findings support the hypothesis that diffusion models benefit from learning over multiple token orderings, which contributes to their robustness and [x.com/giffmana/statu...](https://x.com/giffmana/status...)

[Show more](#)



# We tried this experiment out:



Mihir Prabhudesai  
@mihirp98



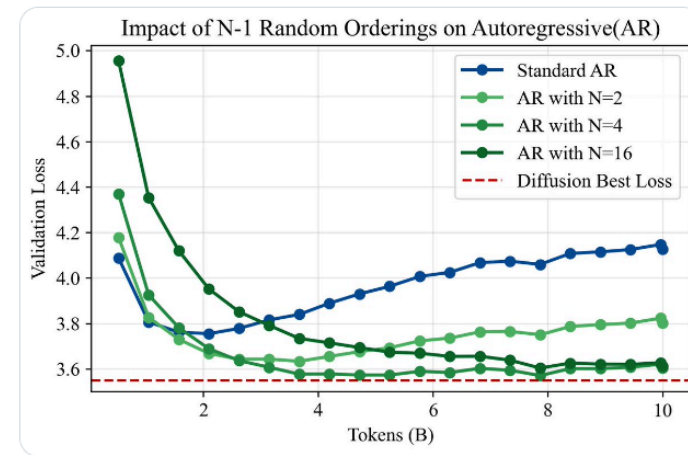
We ran more experiments to better understand “why” diffusion models do better in data-constrained settings than autoregressive. Our findings support the hypothesis that diffusion models benefit from learning over multiple token orderings, which contributes to their robustness and reduced overfitting.

To test this, we trained autoregressive (AR) models with varying numbers of token orderings:  $N=1$  corresponds to the standard left-to-right ordering, while  $N=k$  includes the left-to-right order plus  $k-1$  additional random permutations. As  $N$  increases, we observe that AR models become more data-efficient, exhibiting improved validation loss and reduced overfitting.

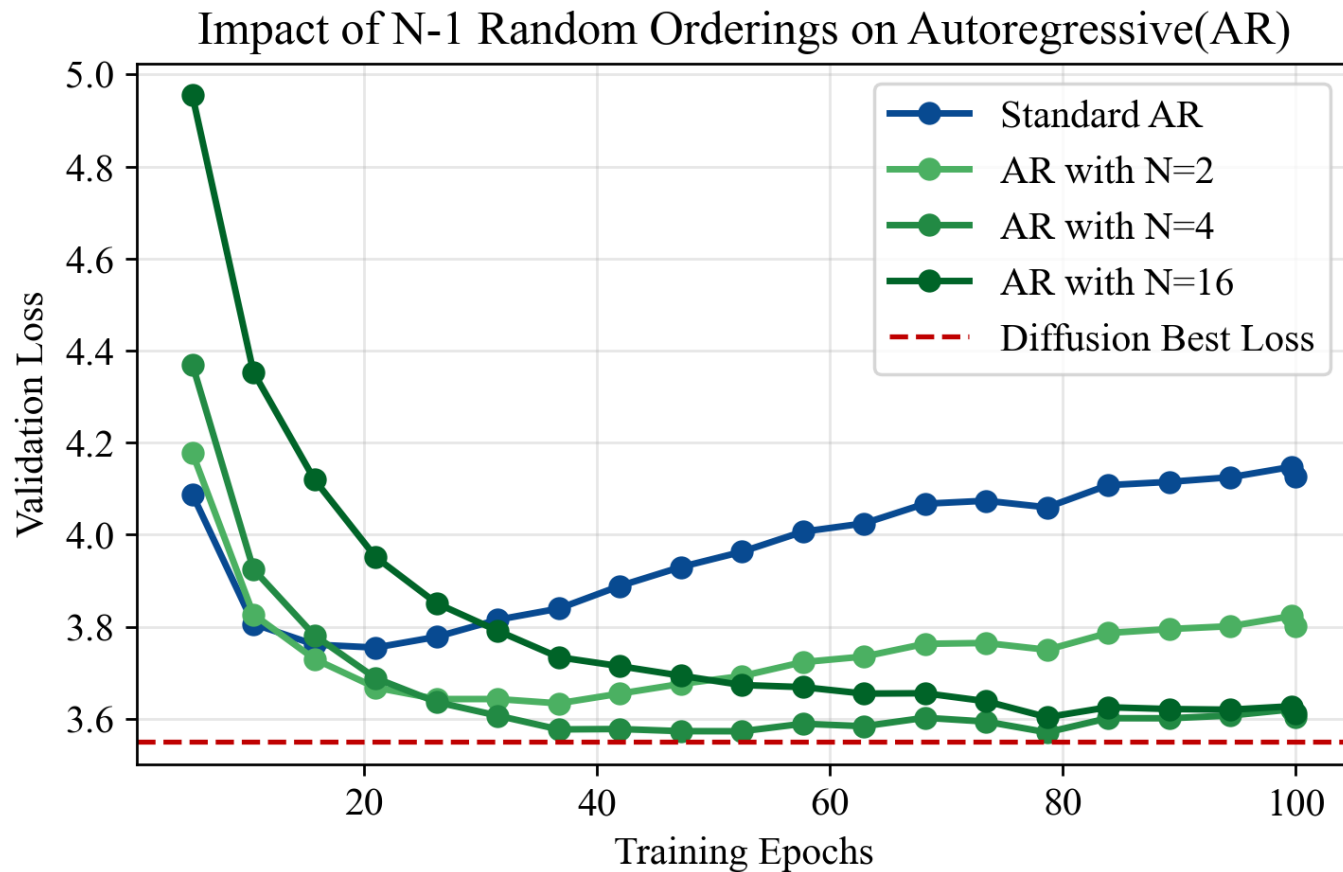
All models were trained for 100 epochs, and were evaluated using the standard left-to-right factorization. We also experimented with related approaches, such as RAR and  $\sigma$ -GPT, and observed consistent trends -- introducing more random factorizations led to better generalization and less overfitting.

We have updated our arXiv submission with these new results. We thank @giffmana and @YouJiacheng for suggesting these experiments.

Original paper post - [x.com/mihirp98/statu...](https://x.com/mihirp98/status/1744444444444444444)



# Token ordering explains diffusion's data efficiency.



- Potentially could allow one to interpolate between data & compute efficiency.



A concurrent work (came after 3 weeks of our release), also validated our core findings:

