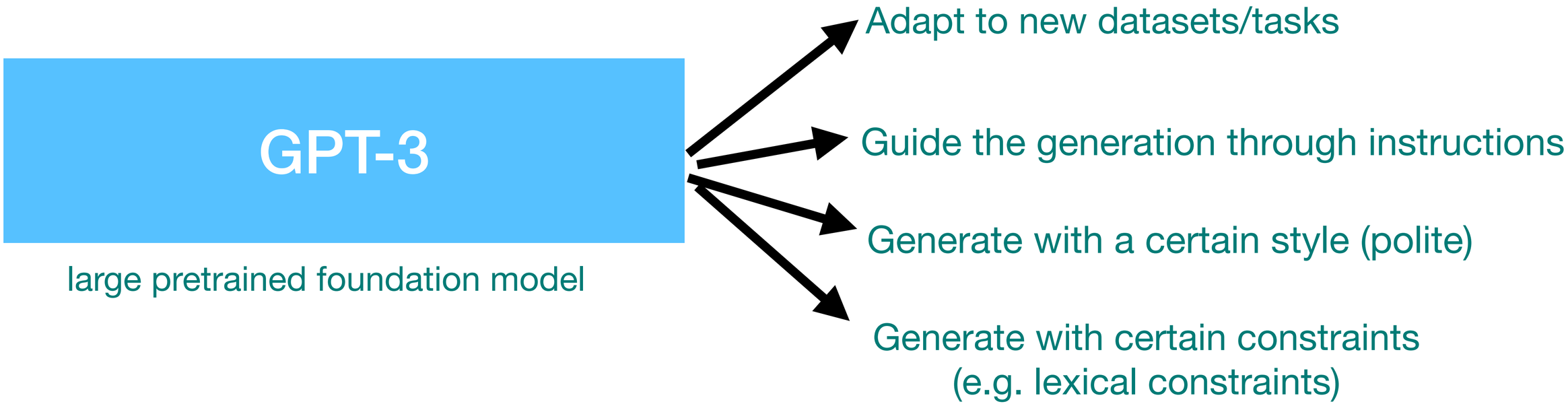
Overview of controllable generation of text and images

Troshin Sergei

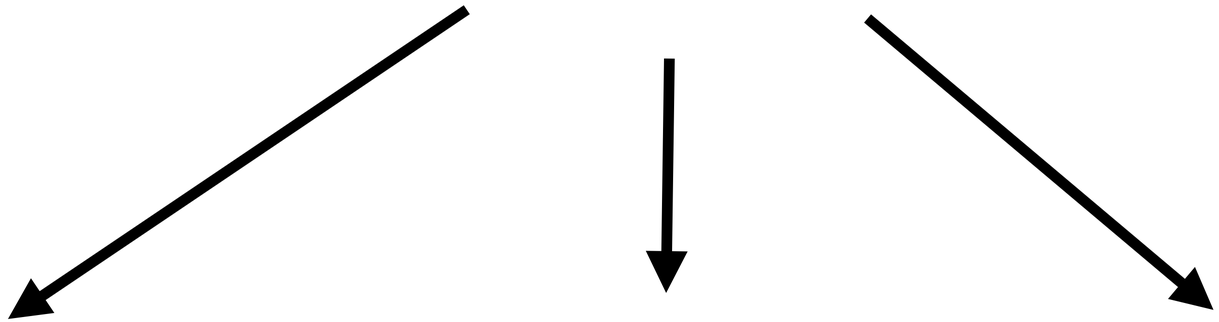
Part 1: Controllable Neural Text Generation

- In-Context Learning (teaching frozen model)
- Finetuning: how to efficiently adapt large pretrained models
- Decoding guidance for constrained generation
- Soft-tokens for gradient-based text guidance

Controllable generation



Adapting pretrained models for text generation



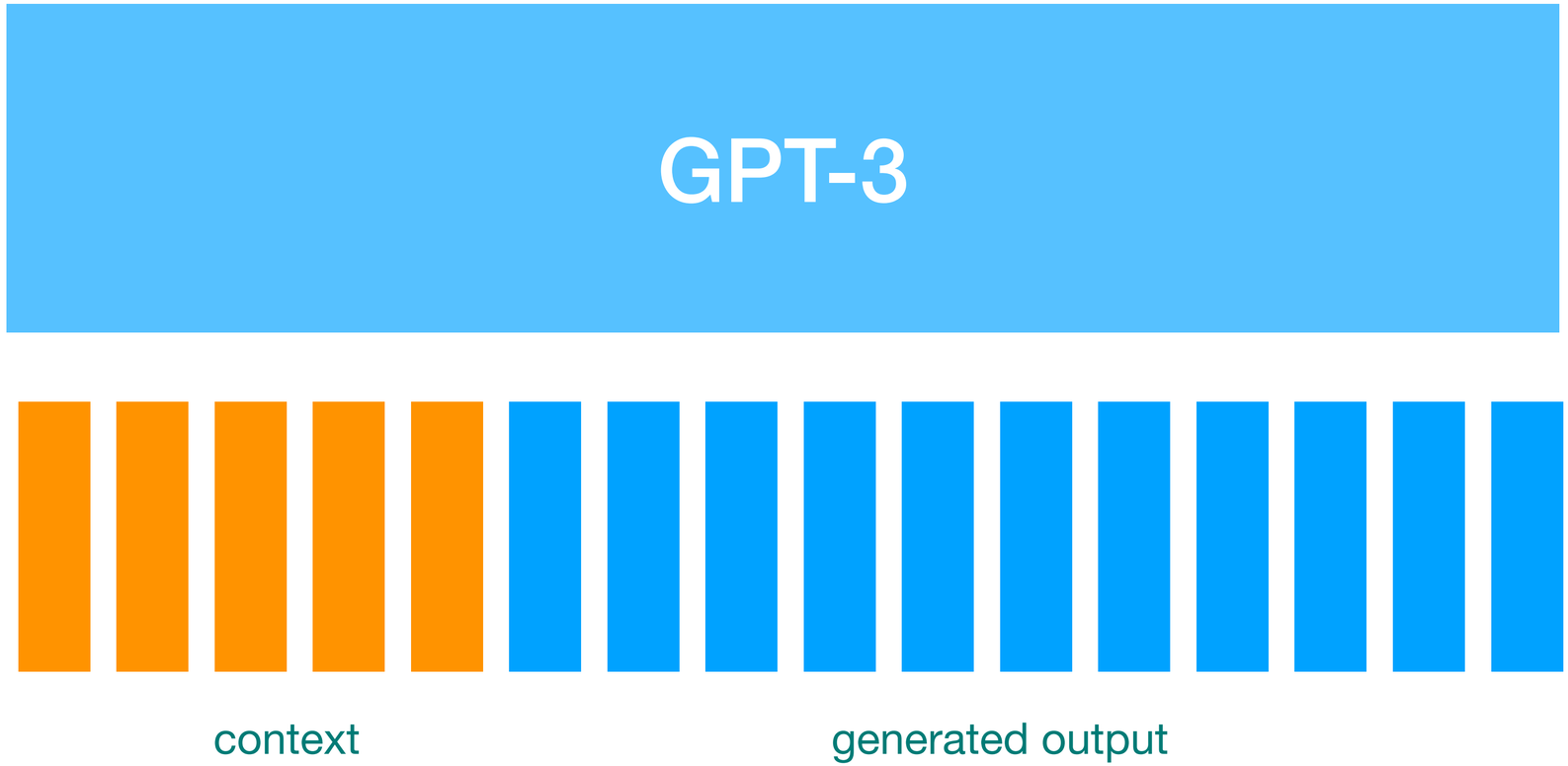
Zero-shot

In Context Learning (ICL)

Fine-tuning

Guided decoding

Zero-shot: prompting



- Condition the GPT-3 model on textual prompt

Prompts for GPT-3

How many ways are there to arrange the numbers from 1 to 9 in a 3 by 3 square, so that in each row the numbers increase from left to right, and in each column they increase from bottom to top?

There are 72 ways to arrange the numbers from 1 to 9 in a 3 by 3 square, so that in each row the numbers increase from left to right, and in each column they increase from bottom to top.

- HSE maths intro test example

Prompts for GPT-3

How many ways are there to arrange the numbers from 1 to 9 in a 3 by 3 square, so that in each row the numbers increase from left to right, and in each column they increase from bottom to top? Let's think step by step.

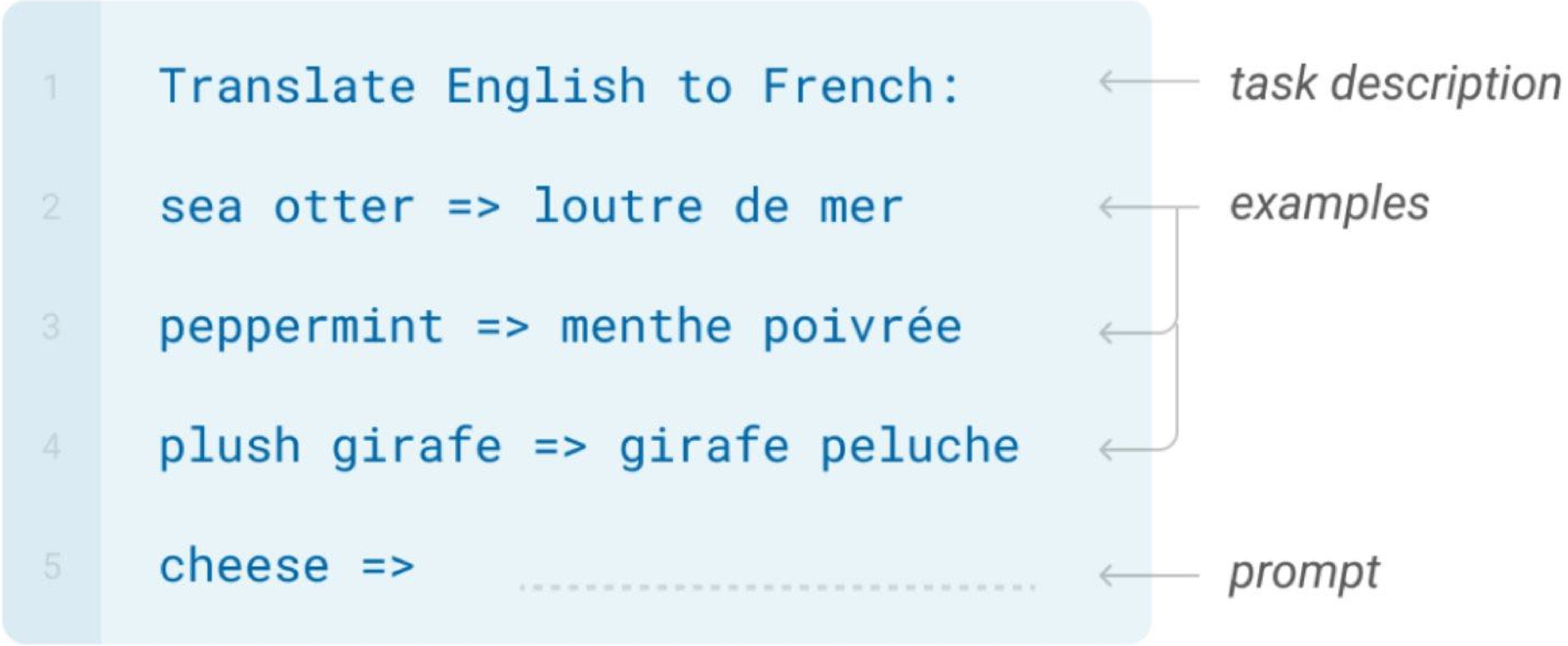
There are 9 places in the square, and we need to fill them with the numbers 1-9.

We can start by filling the bottom row from left to right. We have 3 choices for the first number, 2 choices for the second number, and 1 choice for the third number. So there are 3*2*1=6 ways to fill the bottom row.

Then we need to fill the middle row. The first number in the middle row has 2 choices (it can't be 1 or 9, because those numbers are already in the bottom row), the second number has 1 choice, and the third number has 2 choices. So there are 2*1*2=4 ways to fill the middle row.

Finally we need to fill the top row. The first number in the top row has 1 choice (it can't be 1, 2, or 9, because those numbers are already in the bottom and middle rows), the second number has 2 choices, and the

In-context learning: priming, learning from examples



- Works only for large models e.g. GPT-3

https://twitter.com/colinraffel/status/1524550436203728896

“What Can Transformers Learn In-Context? A Case Study of Simple Function Classes”

Garg et al. 2022

For a given distribution of **inputs** x_i and **functions** f :

Training:

for sampled f, prompt $x^i_{\text{prompt}} = (x_1, f(x_1), x_2, f(x_2), \ldots, x_i, f(x_i), x_{i+1})$

optimize: $\min_\theta \mathbb{E}_{x_{\text{prompt}}} \left[\frac{1}{k+1} \sum_{i=0}^{k} \ell \left(M_\theta \left(x^i_{\text{prompt}} \right), f\left(x_{i+1} \right) \right) \right]$

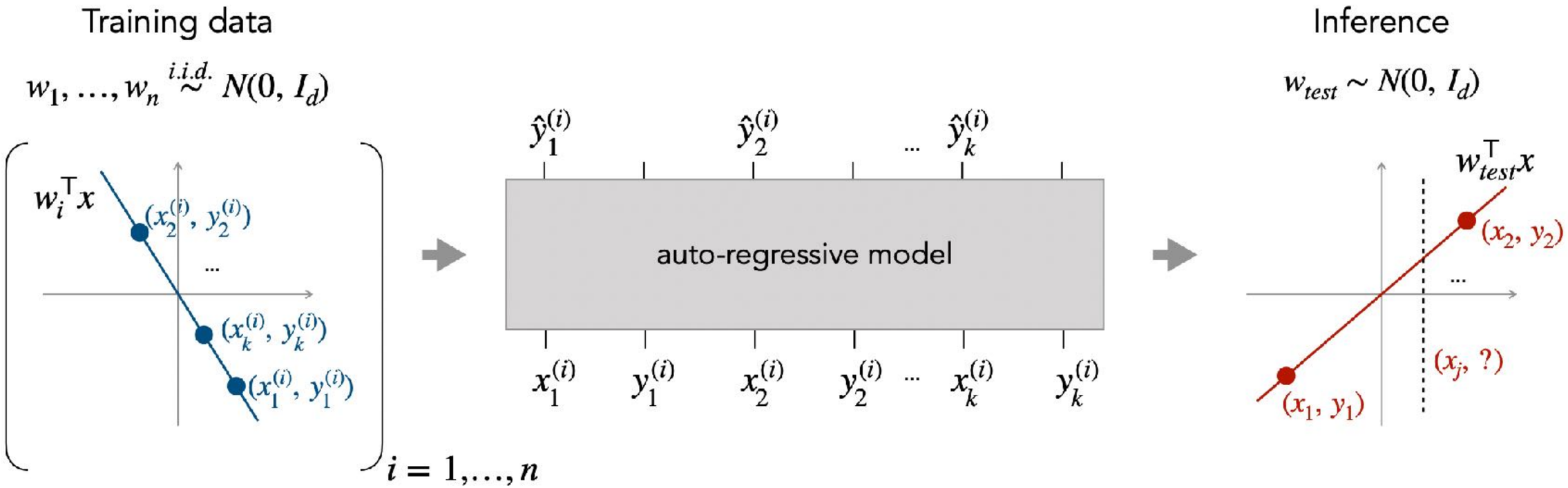
Testing

for sampled f, prompt, x_test: calculate quality (generalize to new functions?):

What function families can Transformer learn in-context?

https://arxiv.org/pdf/2208.01066.pdf

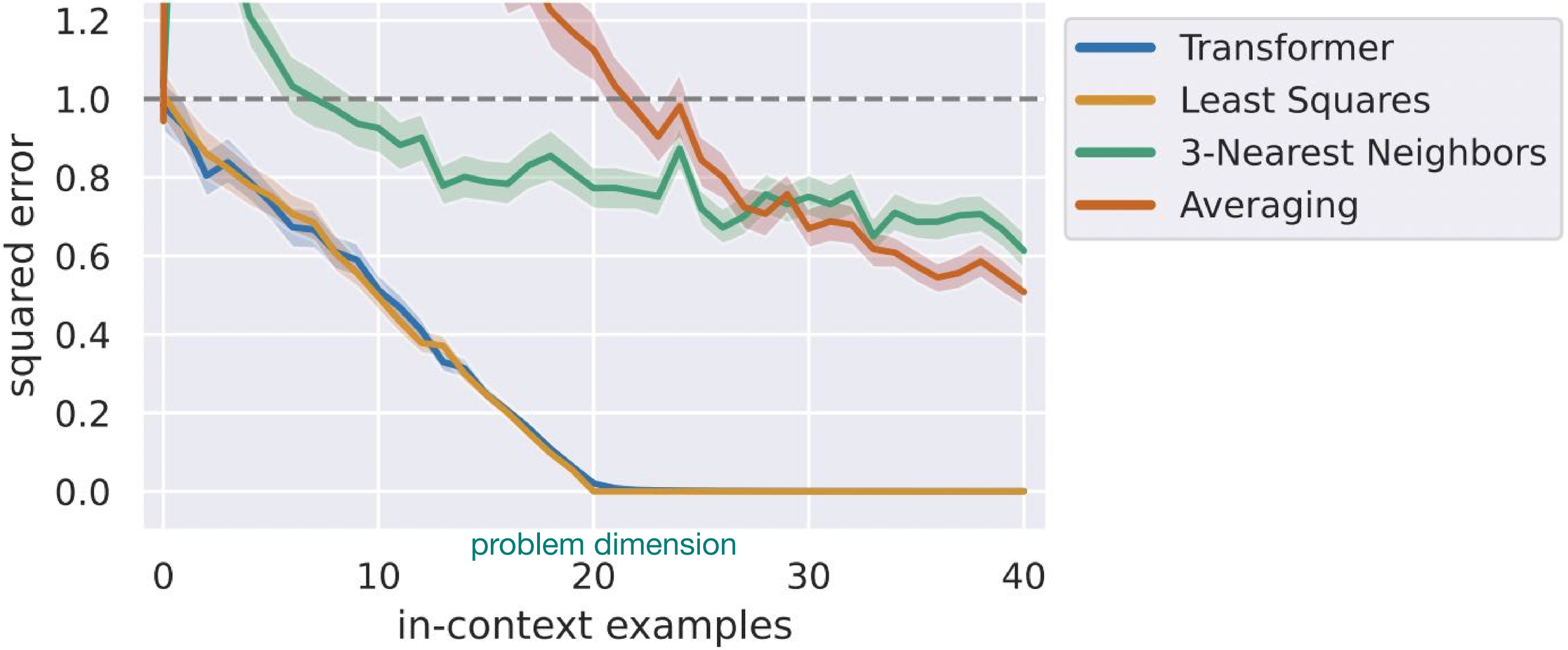
In-context learning: priming



- On test the **functions are different** but from the **same class**

https://arxiv.org/pdf/2208.01066.pdf

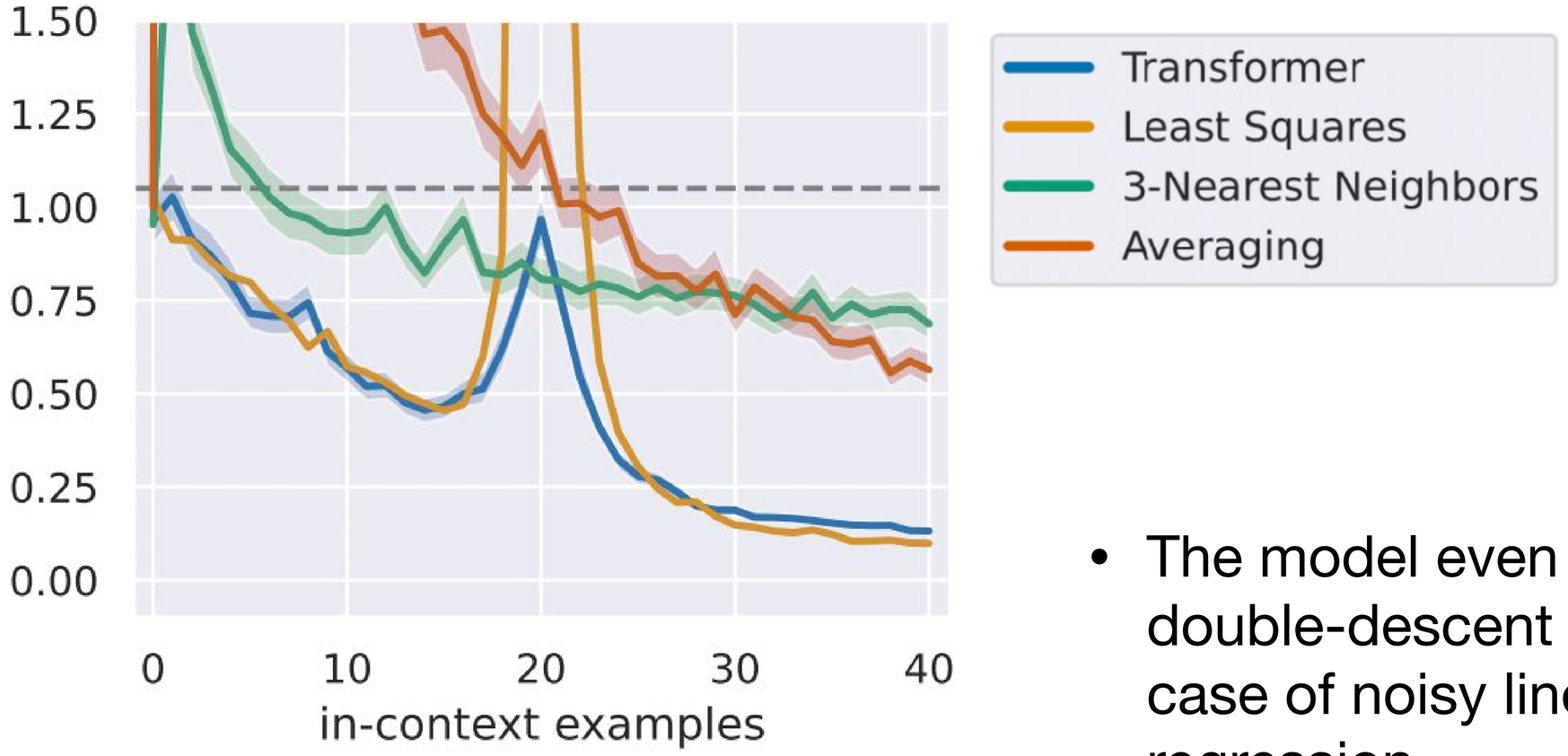
In-context learning: priming



- Transformer can in-context learn **linear functions**

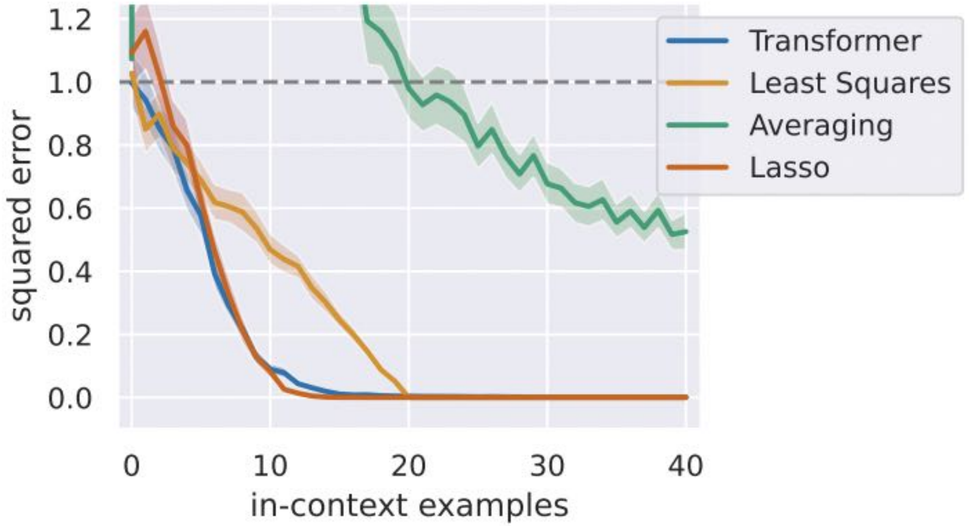
https://arxiv.org/pdf/2208.01066.pdf

In-context learning: priming



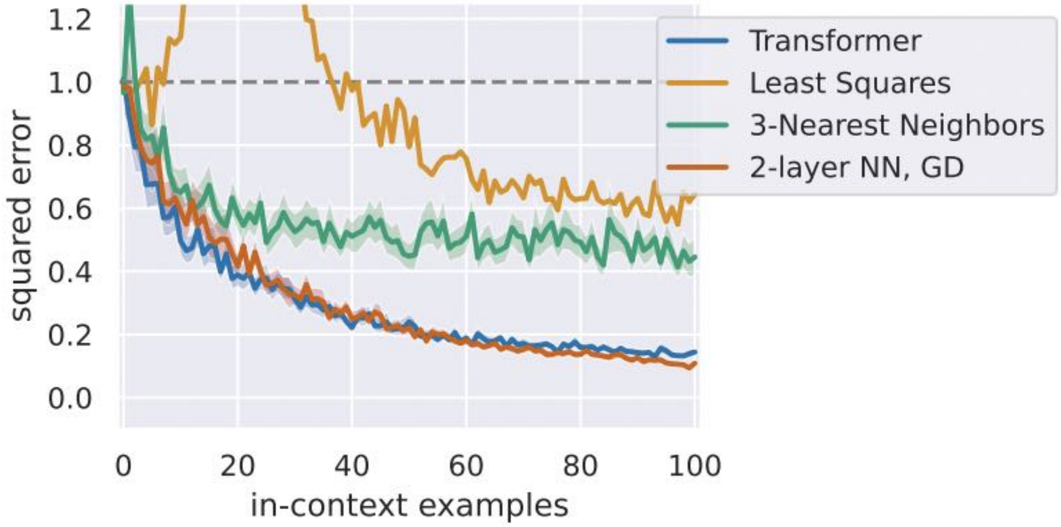
(b) Noisy linear regression

- The model even exhibit double-descent behaviour in case of noisy linear regression

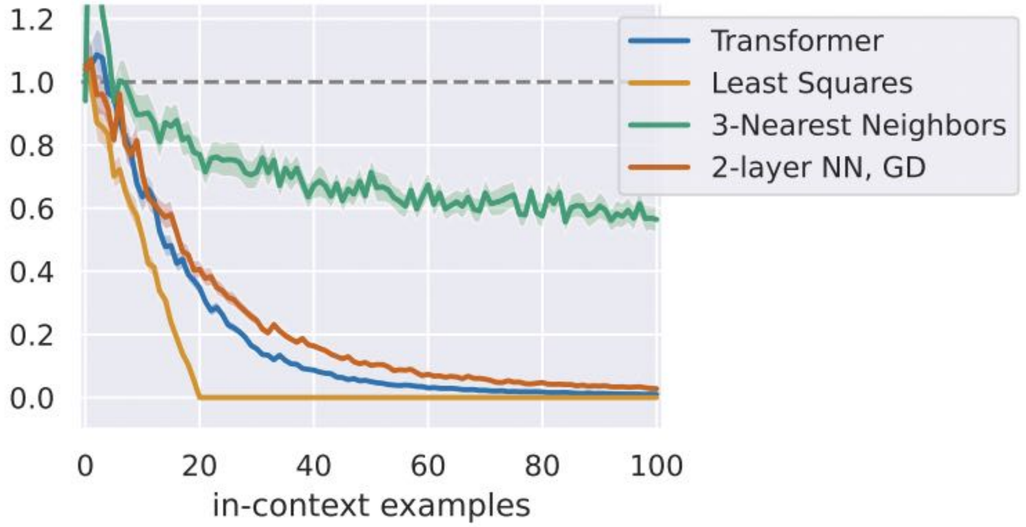


(a) Sparse linear functions

(b) Decision trees



(c) 2-layer NN

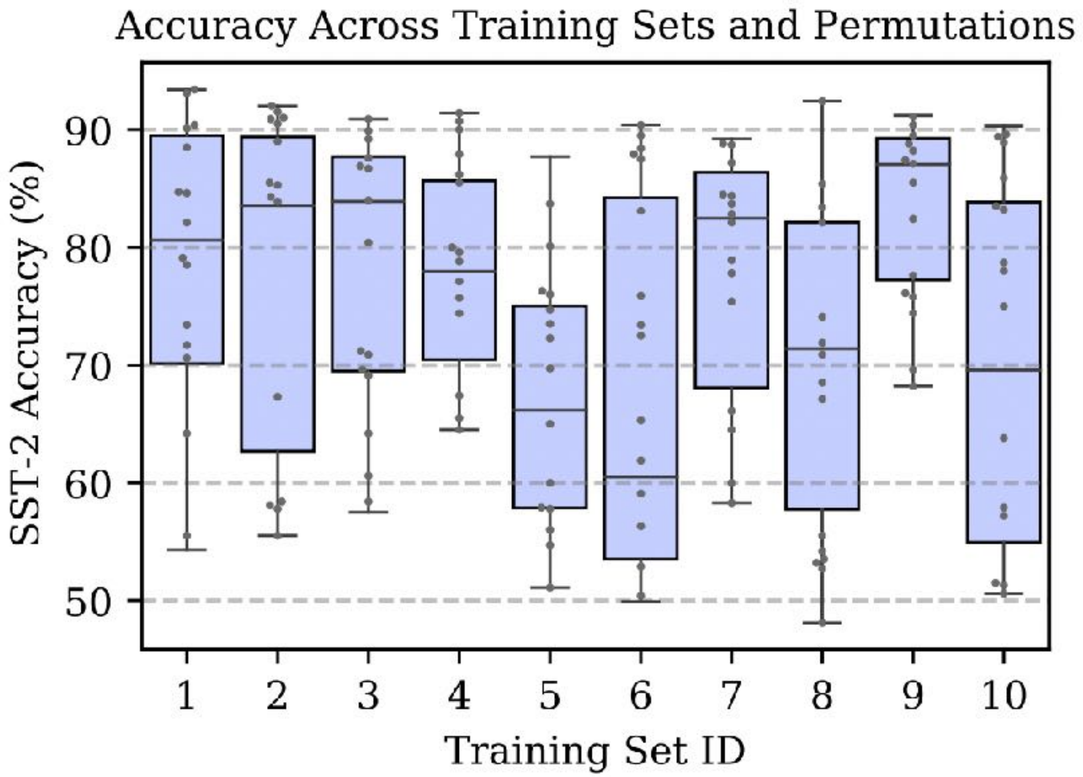


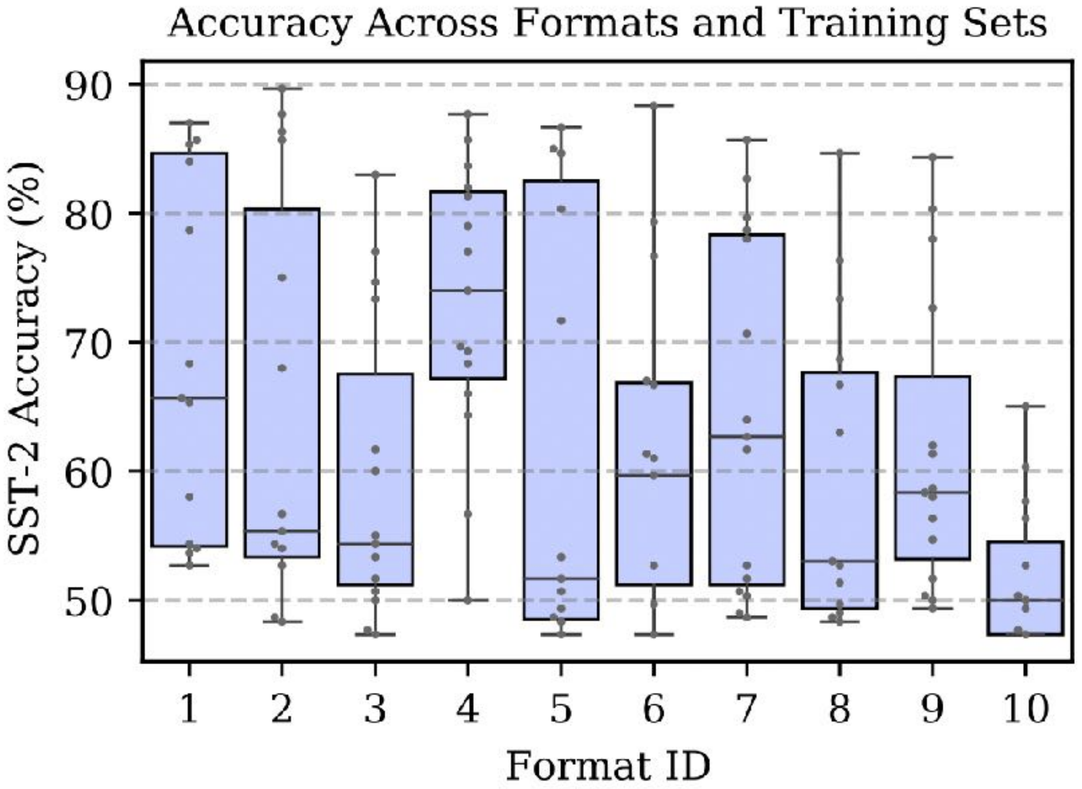
(d) 2-layer NN, eval on linear functions

- Transformer can in-context learn more complex families, such as 2-layer NN

Unstability of In-Context learning

Zhao et al 2022, *Calibrate Before Use: Improving Few-Shot Performance of Language Models*





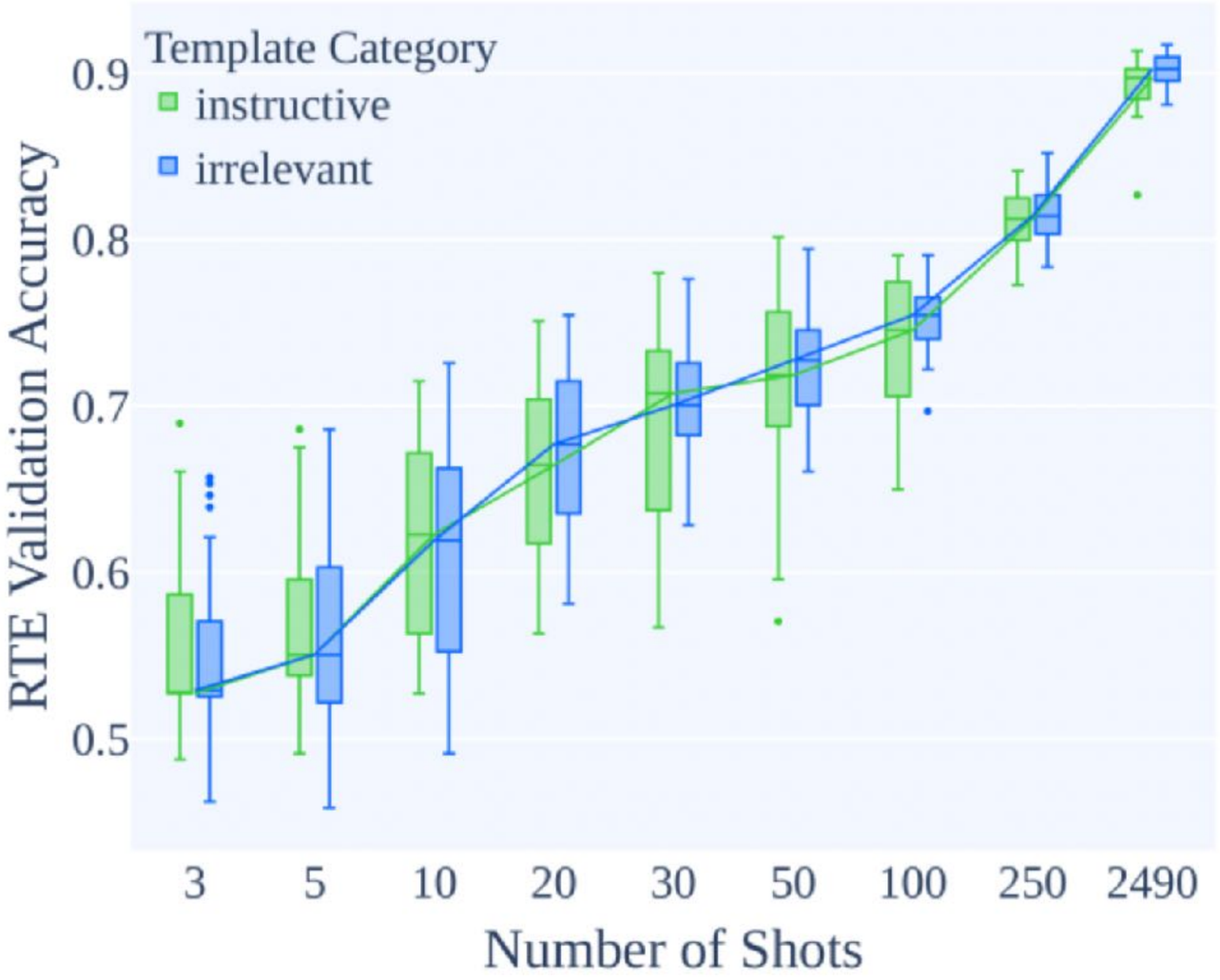
- **high variance** in GPT-3's accuracy as we change the prompt's training examples, as well as the **permutation** of the examples

- **high variance** in GPT-3's accuracy w.r.t **prompt format**

https://arxiv.org/pdf/2102.09690.pdf

Do Prompt-Based Models Really Understand the Meaning of their Prompts?

T0 (3B) on Recognizing Textual Entailment

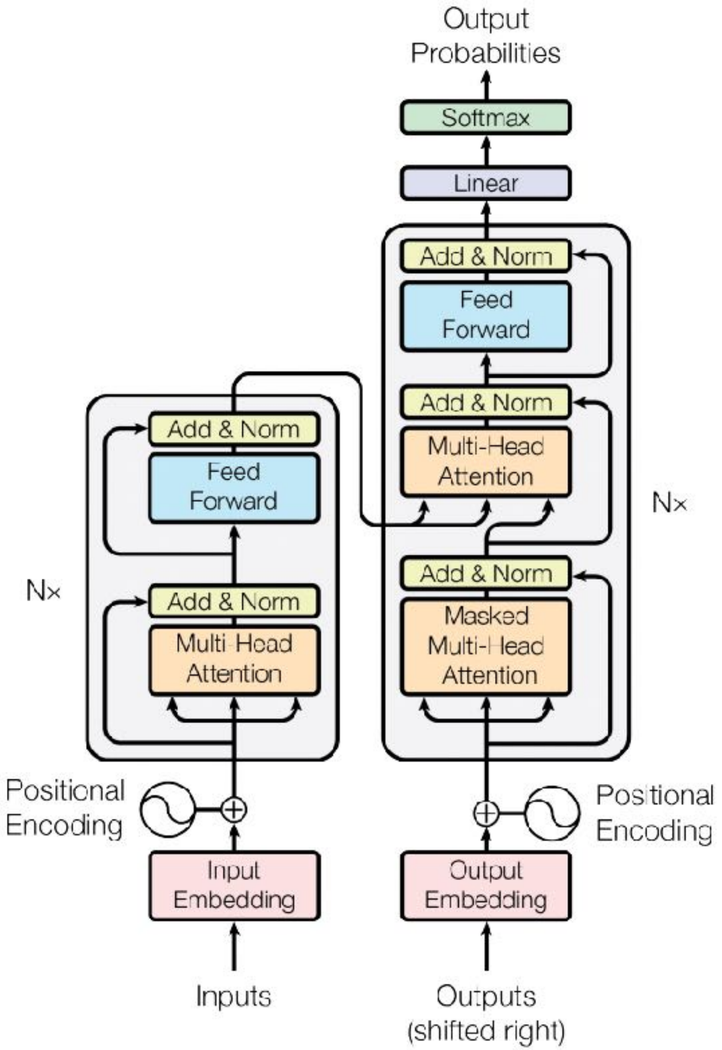


Webson and Pavlick 2022

- We may not fully control the prompting and in-context learning process, and we do not fully understand it

https://arxiv.org/abs/2109.01247

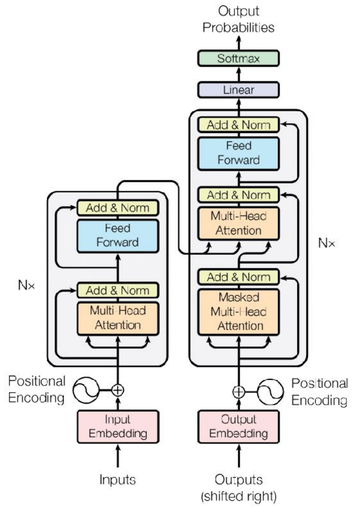
Finetuning



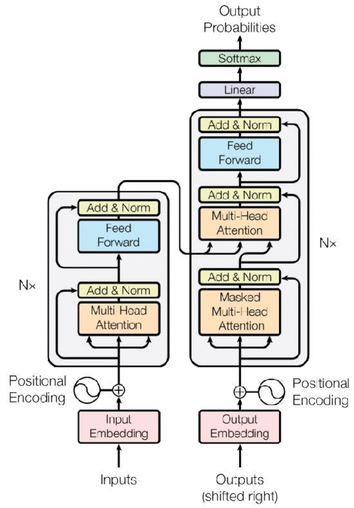
pretrained model

finetune

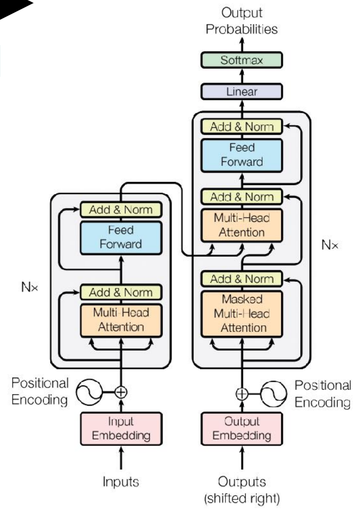
separate model for each task



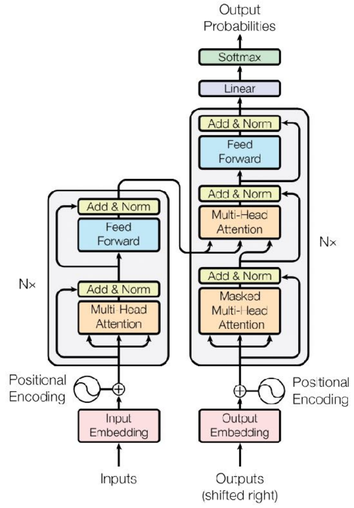
Task 1



Task 2



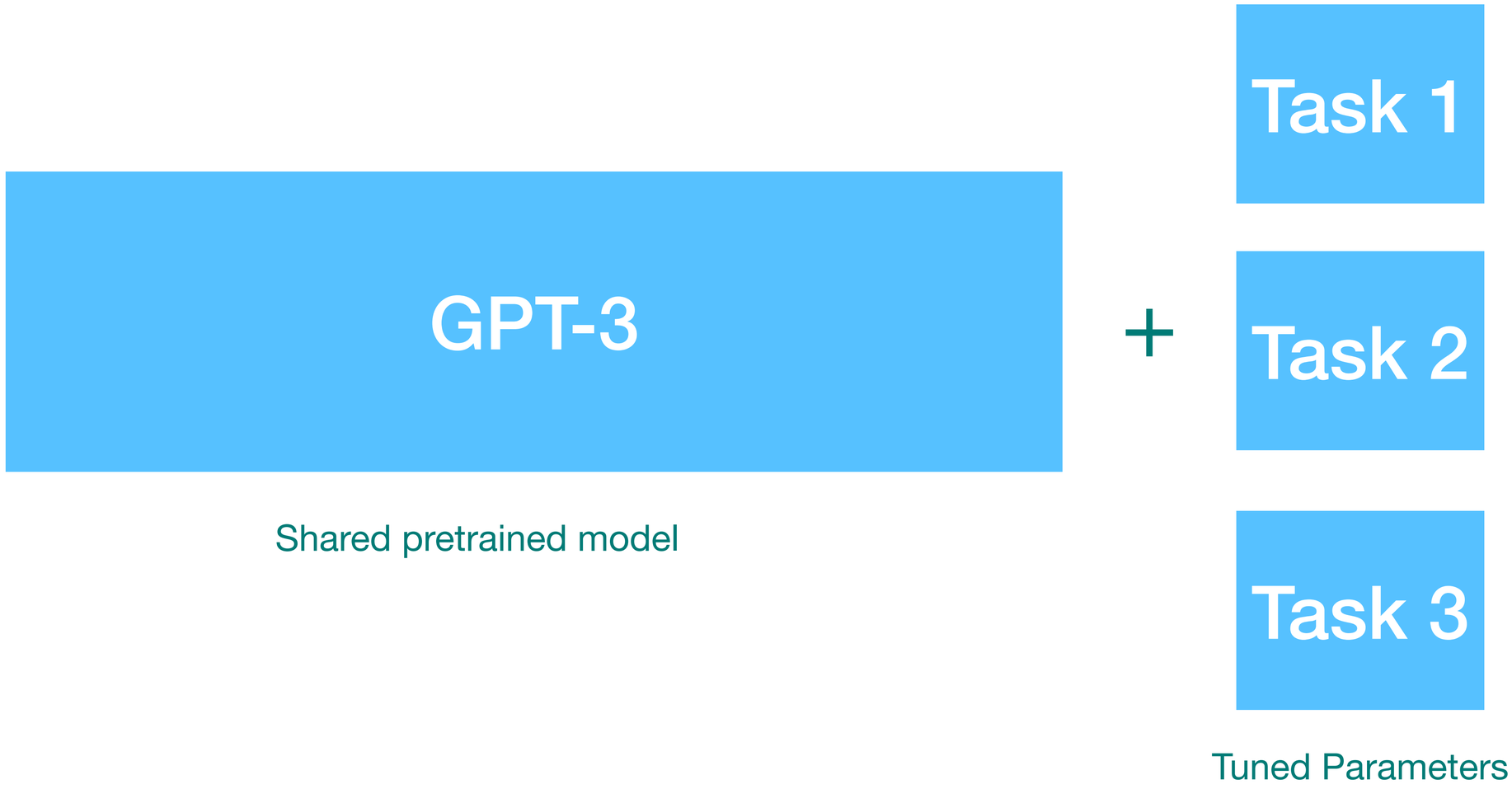
Task 3



Task 4

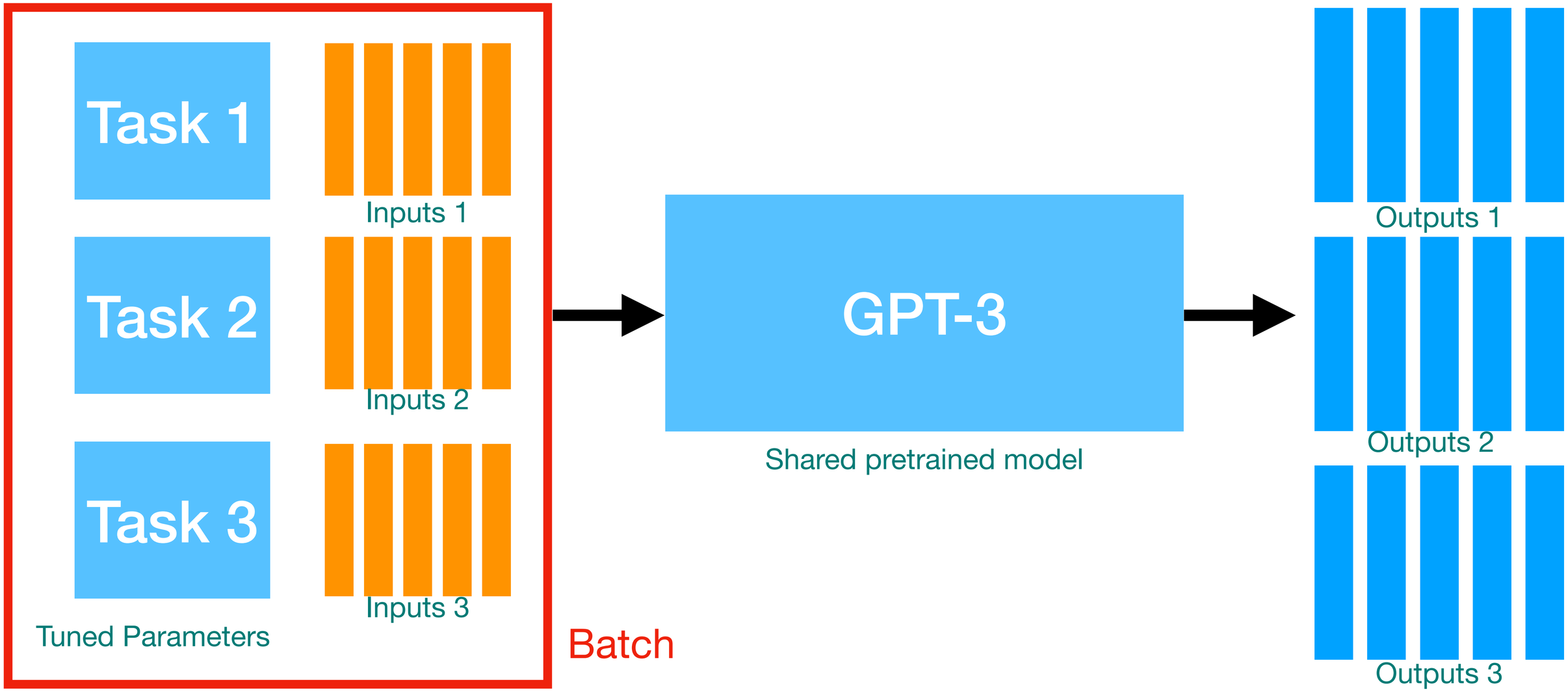
- Finetuning require storying separate set of parameters for each task

Parameter Efficient Finetuning



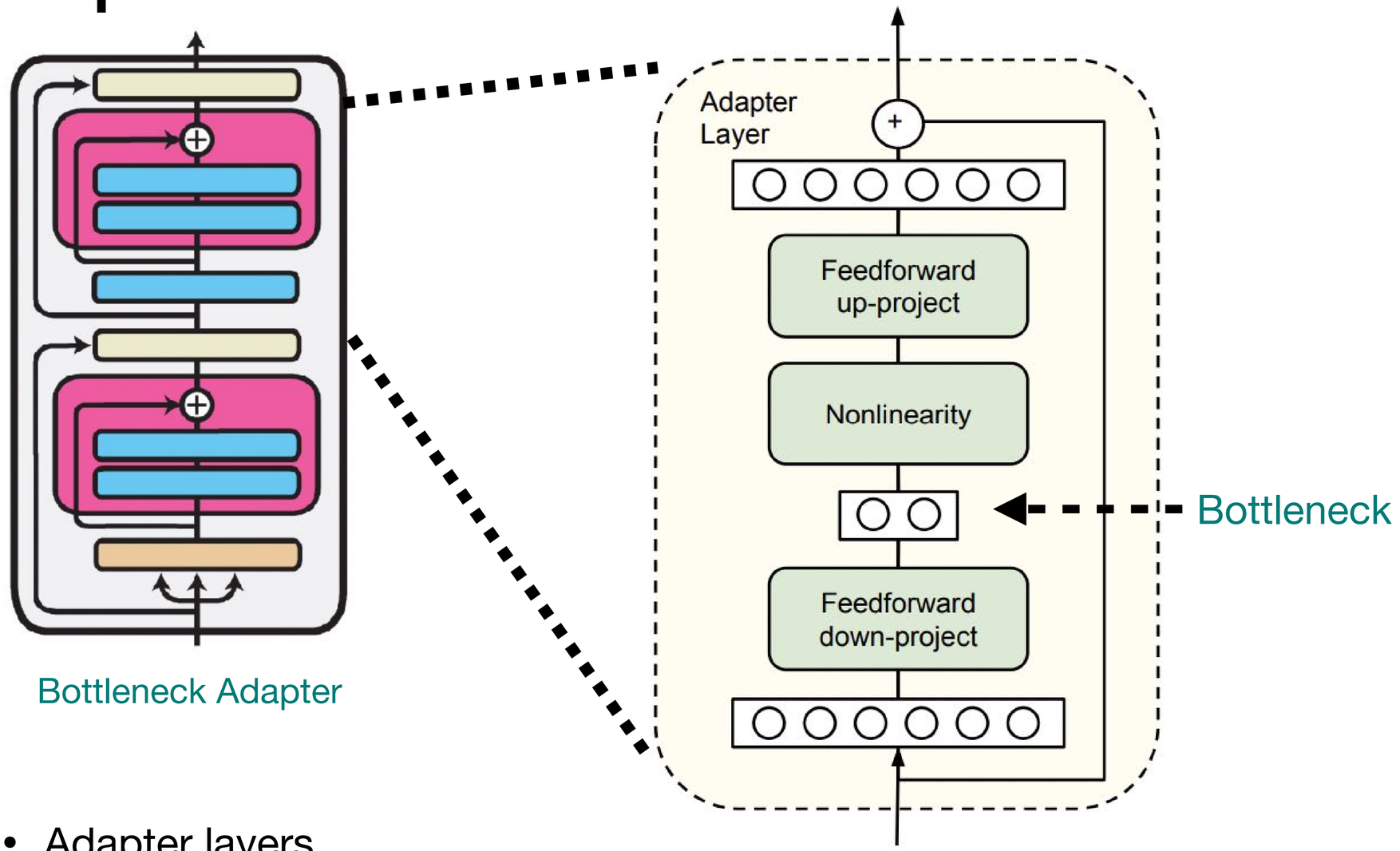
- We want to finetune only a very small number of task-specific parameters

Parameter Efficient Finetuning



- Ideally we want to be able to pass the tasks-specific parameters as if it is part of the input

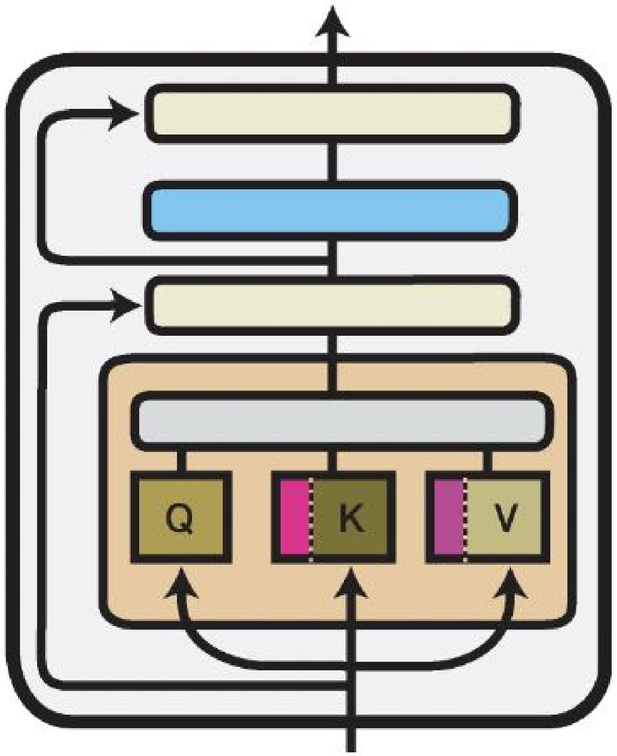
Parameter Efficient Finetuning: Bottleneck Adapters



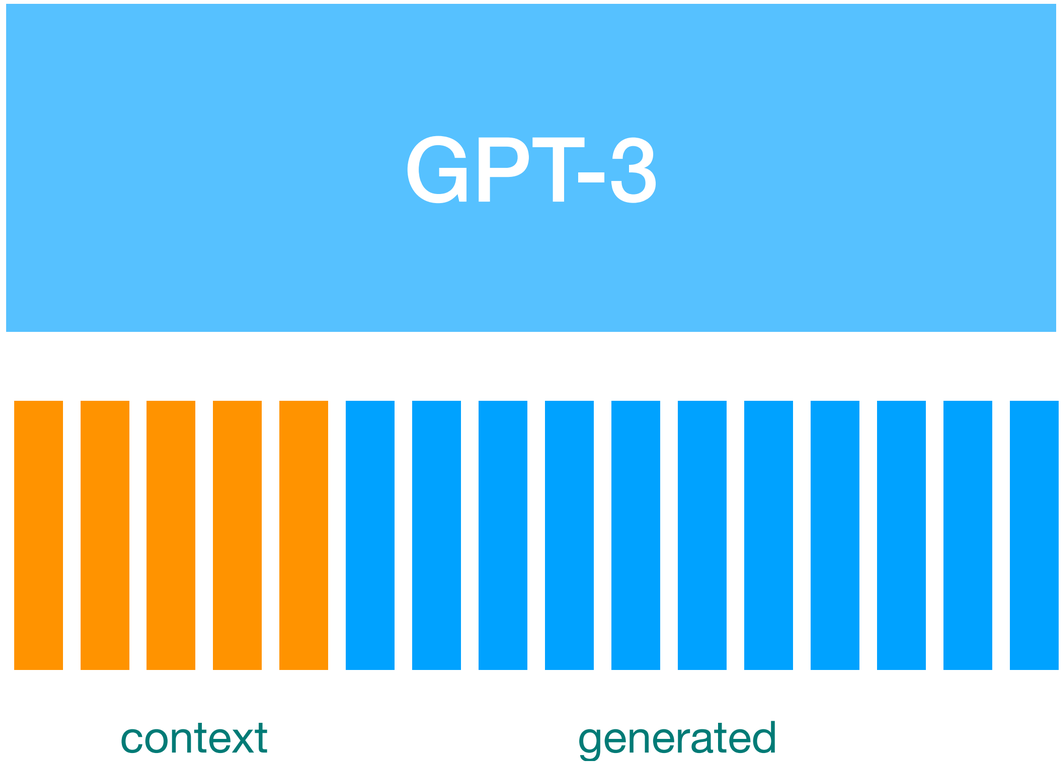
- Adapter layers

https://adapterhub.ml/blog/2022/03/adapter-transformers-v3-unifying-efficient-fine-tuning/

Parameter Efficient Finetuning approaches: Prefix tuning, low rank



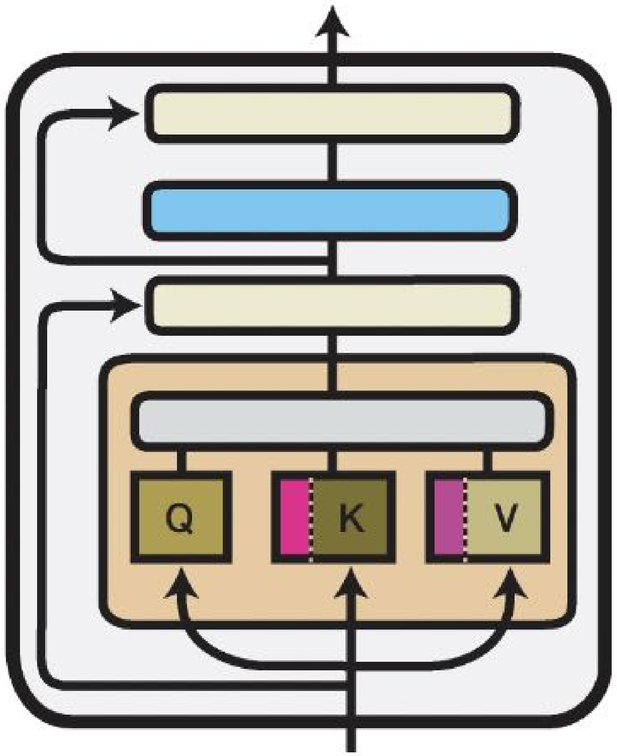
Prefix Adapter (Prefix tuning)



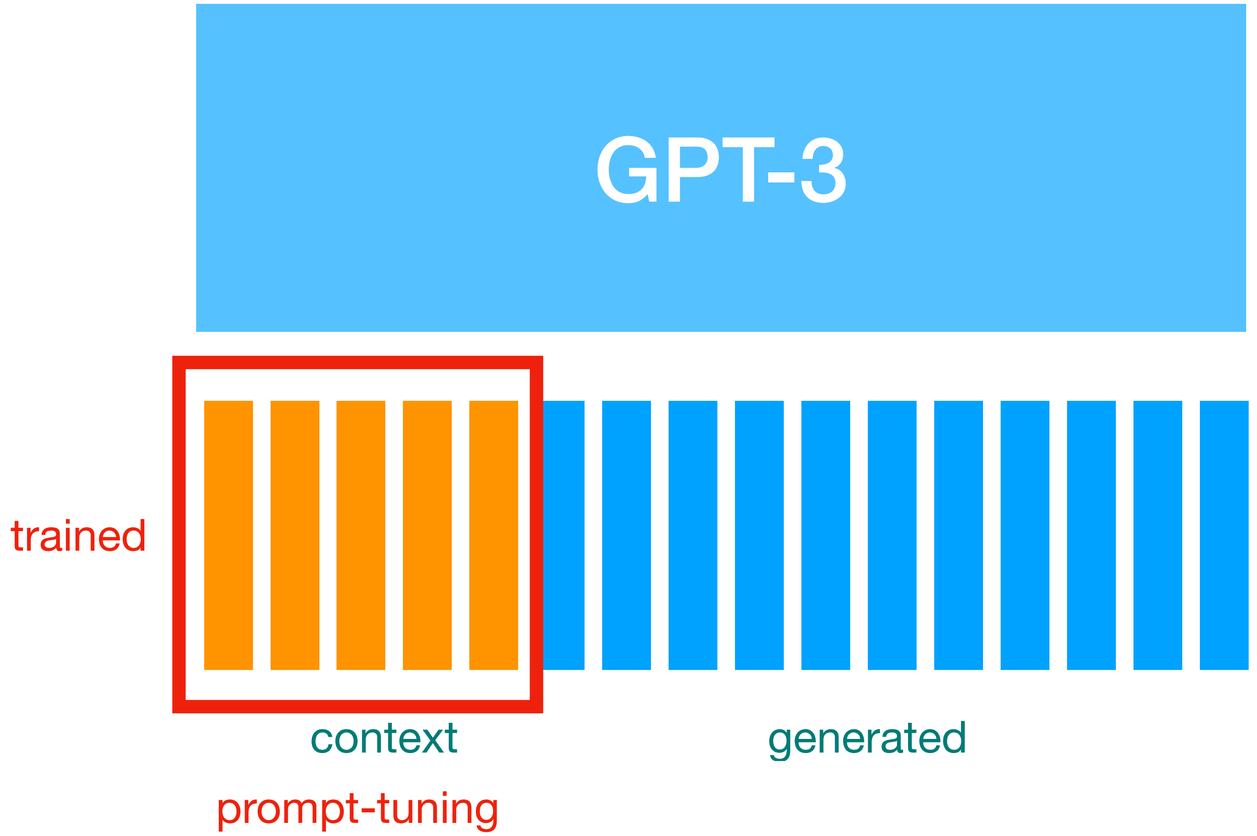
- Introduce tuned prefix activation;

https://adapterhub.ml/blog/2022/03/adapter-transformers-v3-unifying-efficient-fine-tuning/

Parameter Efficient Finetuning approaches: Prefix tuning, low rank



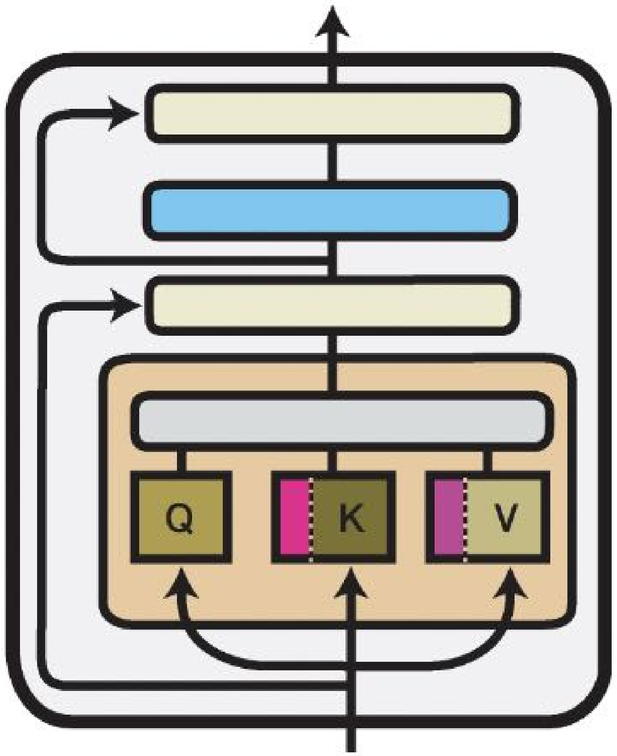
Prefix Adapter (Prefix tuning)



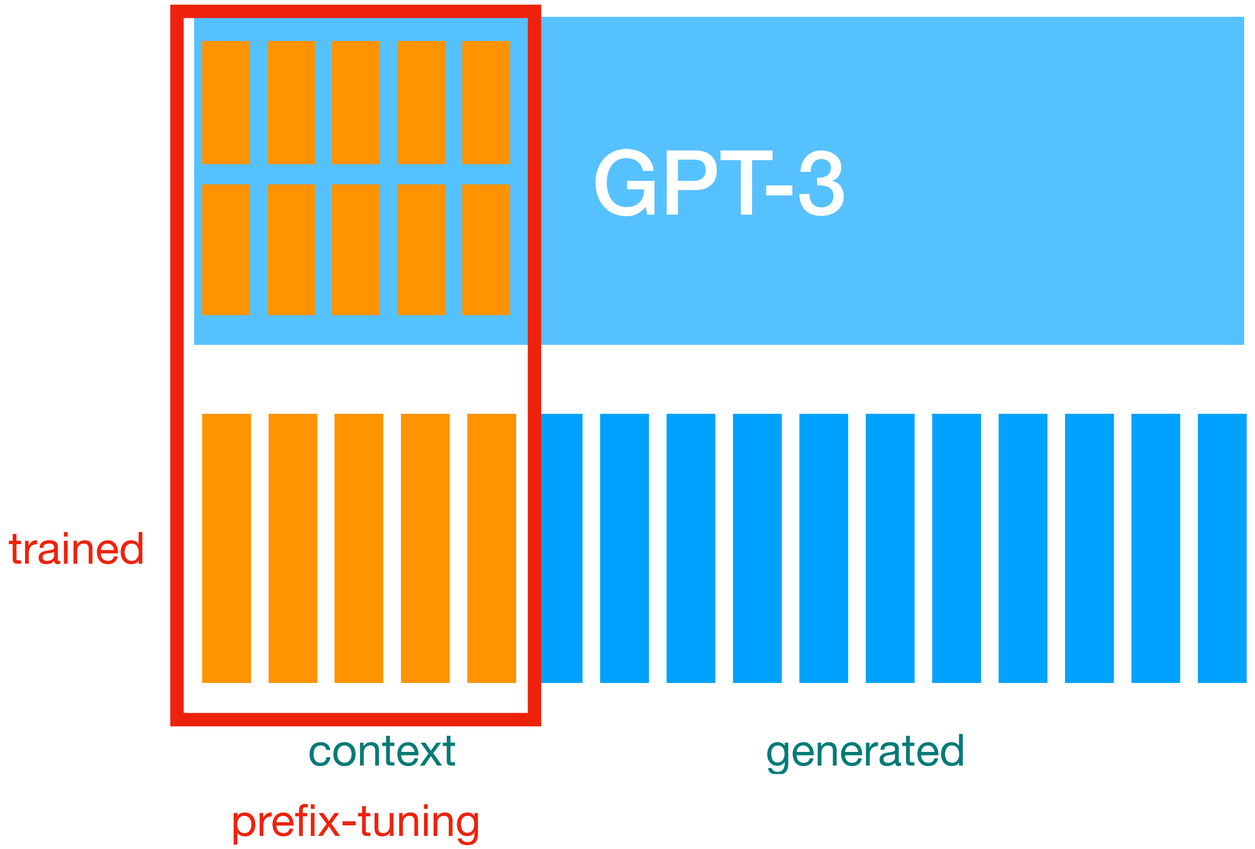
- Introduce tuned prefix activation;

https://adapterhub.ml/blog/2022/03/adapter-transformers-v3-unifying-efficient-fine-tuning/

Parameter Efficient Finetuning approaches: Prefix tuning, low rank



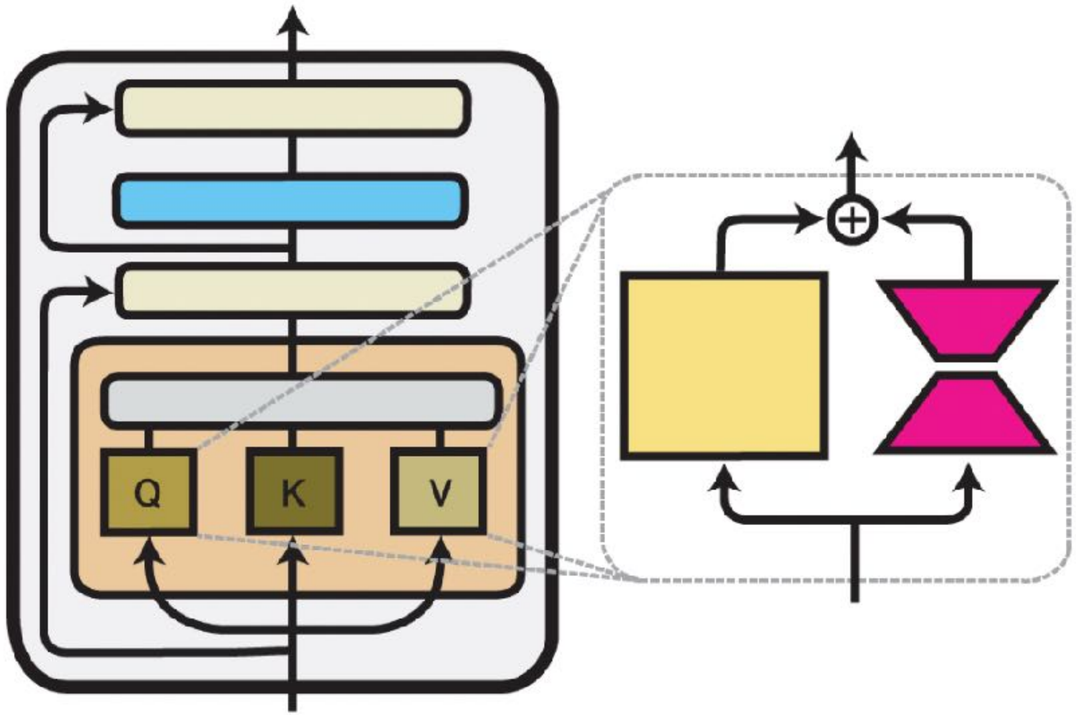
Prefix Adapter (Prefix tuning)



- Introduce tuned prefix activation;

https://adapterhub.ml/blog/2022/03/adapter-transformers-v3-unifying-efficient-fine-tuning/

Parameter Efficient Finetuning approaches: LoRA: Low rank adapters



Low-Rank Adaptation (LoRA)

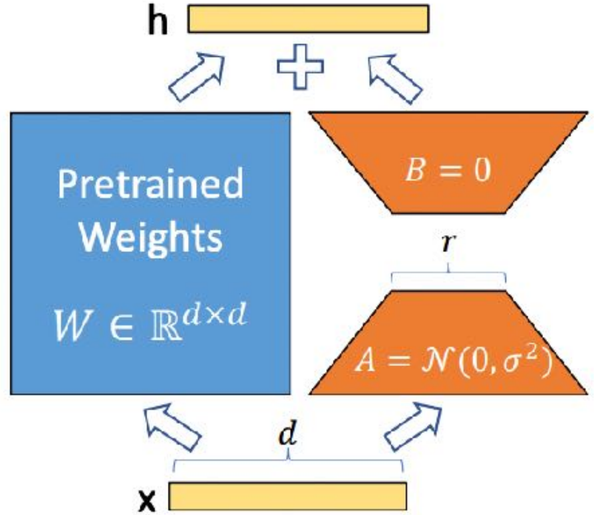


Figure 1: Our reparametrization. We only train A and B.

- For GPT-3 175B: reduce the number of trainable parameters 10000x
- reduce GPU memory requirements 3x

https://adapterhub.ml/blog/2022/03/adapter-transformers-v3-unifying-efficient-fine-tuning/

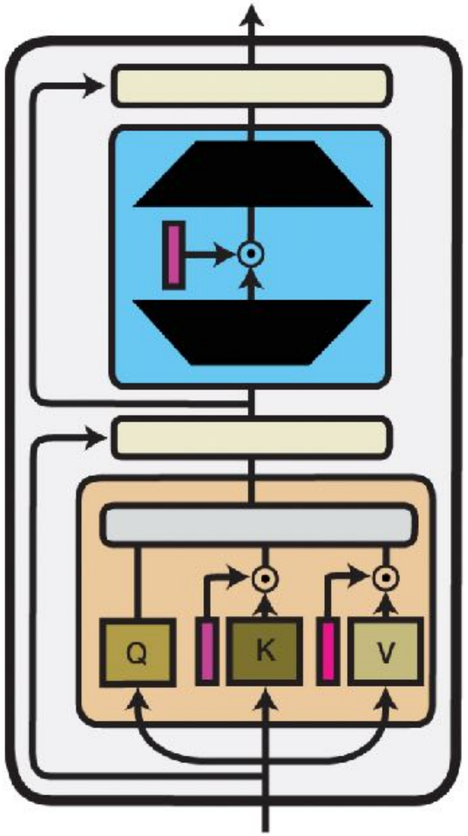
Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning

Haokun Liu* **Derek Tam*** **Mohammed Muqeeth***

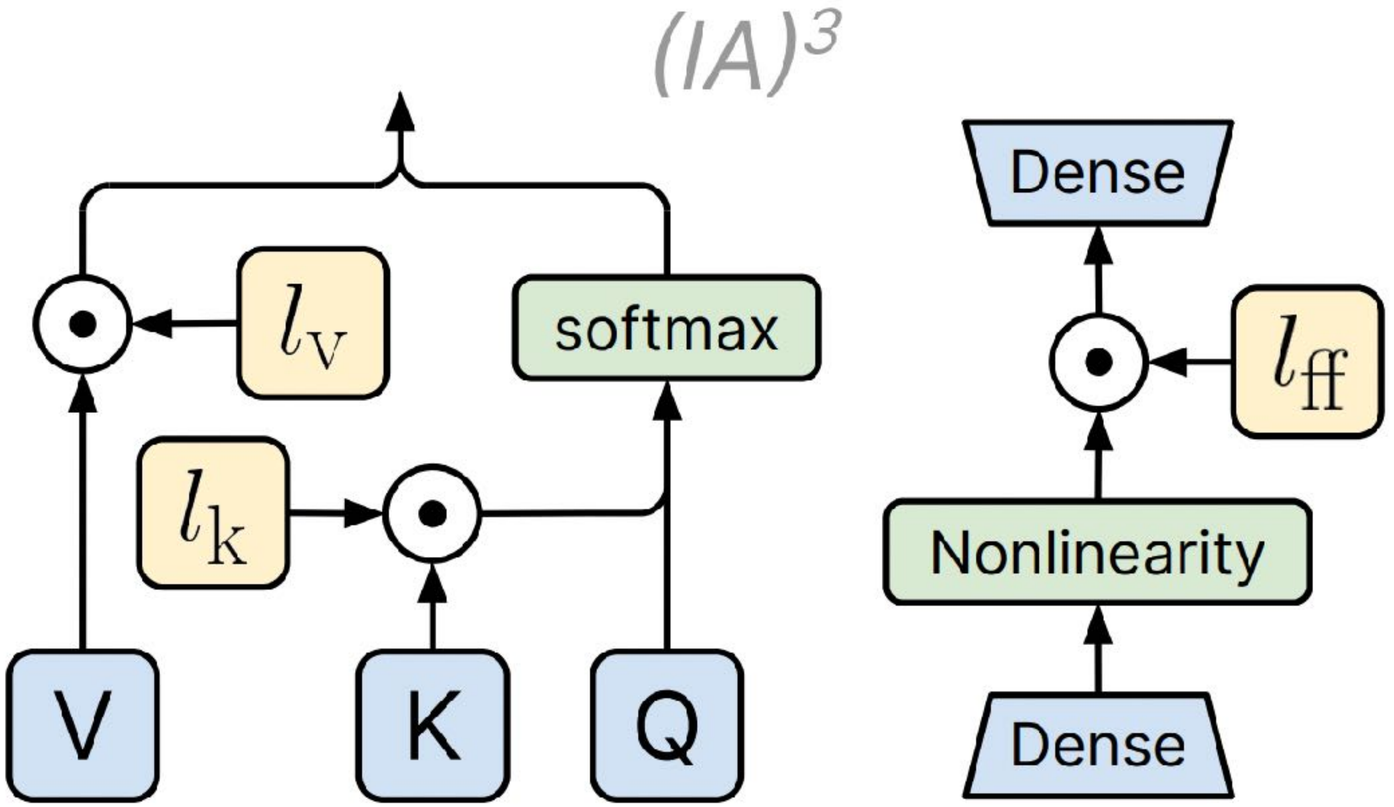
Jay Mohta **Tenghao Huang** **Mohit Bansal** **Colin Raffel**

Department of Computer Science
University of North Carolina at Chapel Hill
{haokunl,dtredsox,muqeeth,craffel}@cs.unc.edu

Parameter Efficient Finetuning approaches: reweighting activations



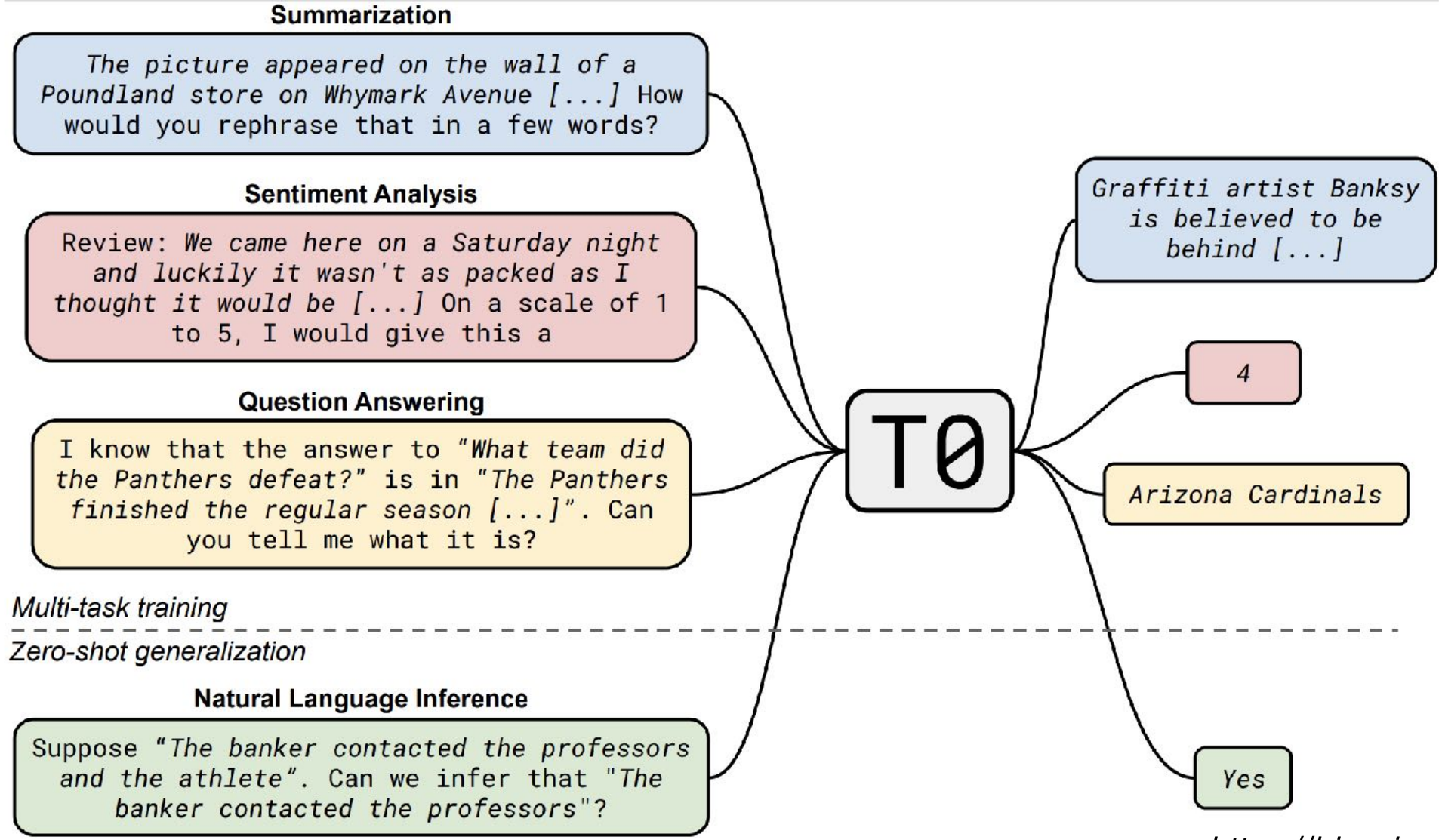
(IA)^3



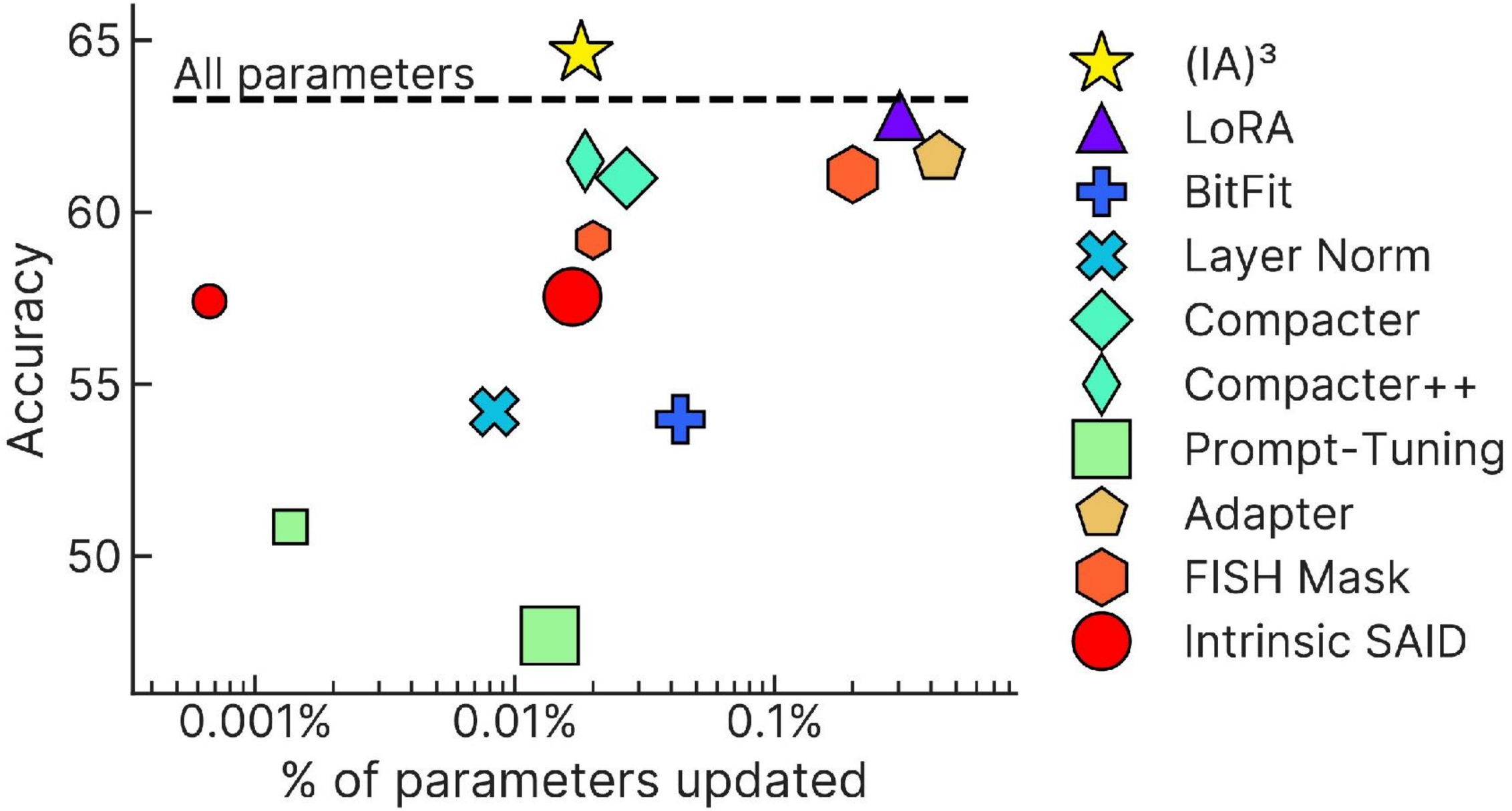
- T-few adapts previous methods into efficient approach
- Total new params: $L(d_k + d_v + d_{ff})$ initialised by ones

T0: zero-shot generalisation to unseen downstream tasks

"if we explicitly train a language model on a massive mixture of diverse NLP tasks, would it generalize to unseen NLP tasks?"



https://bigscience.huggingface.co/blog/t0



- Held-out few-shot datasets: T-few outperforms full fine-tuning

T0 + finetune

ICL

Method	Inference FLOPs	Training FLOPs	Disk space	Accuracy
T-Few	1.1e12	2.7e16	4.2 MB	72.4%
T0 [1]	1.1e12	0	0 B	66.9%
T5+LM [14]	4.5e13	0	16 kB	49.6%
GPT-3 6.7B [4]	5.4e13	0	16 kB	57.2%
GPT-3 13B [4]	1.0e14	0	16 kB	60.3%
GPT-3 175B [4]	1.4e15	0	16 kB	66.6%

Table 1: Accuracy on held-out T0 tasks and computational costs for different few-shot learning methods and models. T-Few attains the highest accuracy with 1,000× lower computational cost than ICL with GPT-3 175B. Fine-tuning with T-Few costs about as much as performing ICL on 20 examples with GPT-3 175B.

- T-few outperforms ICL (GPT-3) being more fast on inference

Decoding guidance

- Want to use the **freezed** pretrained model and guide the decoding process

Decoding guidance

Common Decoding Methods help to increase the quality/diversity of samples (no additional information/constrains)

- Greedy search
- Beam search
- Top-k sampling (Fan et al., 2018)
- Nucleus sampling (Holtzman et al. 2019)
- Penalized sampling (Keskar et al. 2019)

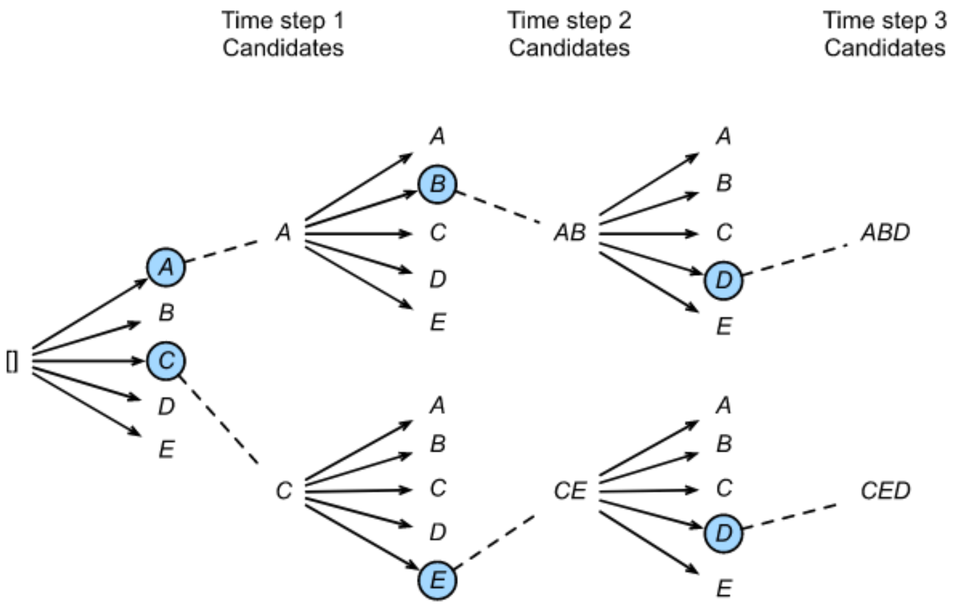


Image source: https://d2l.ai/chapter_recurrent-modern/beam-search.html

Decoding guidance

Constrained LM decoding with manually designed features
Ghazvininejad et al. (2017)

$$\text{score}(x_{t+1}, b_t) = \text{score}(b_t) + \log p(x_{t+1}) + \sum_i \alpha_i f_i(x_{t+1})$$

e.g. block some words from generation

Meister et al. (2020). Regularizations.

$$\mathbf{y}^* = \arg\max_{\mathbf{y} \in \mathcal{Y}} \big(\underbrace{\log p_\theta(\mathbf{y}|\mathbf{x})}_{\text{MAP}} - \underbrace{\lambda \mathcal{R}(\mathbf{y})}_{\text{regularizer}} \big)$$

- Hard to pass global constraints, also problem with left-to-right decoding

https://lilianweng.github.io/posts/2021-01-02-controllable-text-generation/

Energy models to constrain the decoding

$$P(w_t \mid w_{1:t})$$

freezed pretrained model

$$P_{toxic} < 0.05$$

$$'amazing' = w_4$$

constraints

$$E(w_{1:T})$$

Energy function

$$x \sim \frac{e^{-E(x_{1:T})}}{Z}$$

Sample

- Global & local constraints, flexible approach

https://arxiv.org/pdf/2004.11714.pdf

Energy models to constrain the decoding

Residual energy-based models, Deng et al. 2020

Language Model

Energy term

$$P_\theta(x_{p+1}, \cdots, x_T | x_1, \cdots, x_p) = \frac{P_{LM}(x_{p+1}, \cdots, x_T | x_1, \cdots, x_p) \exp(-E_\theta(x_1, \cdots, x_T))}{Z_\theta(x_1, \cdots, x_p)}$$

Partition function

Training of energy model: Noise Contrastive Estimation (NCE)

Assign low energy to data points, and high energy to LM data

$$\max \mathbb{E}_{x_+ \sim P_{data}} \log \frac{1}{1 + \exp(E_\theta(x_+))} + \mathbb{E}_{x_- \sim P_{LM}} \log \frac{1}{1 + \exp(-E_\theta(x_-))}$$

Sampling: importance sampling (not practical)

$$P(x_t | x_{<t}) = P_{LM}(x_t | x_{<t}) \frac{\mathbb{E}_{x'_{t+1}, \cdots, x'_T \sim P_{LM}(\cdot | x_{\leq t})}[\exp(-E_\theta(x_{\leq t}, x'_{t+1}, \cdots, x'_T))]}{\mathbb{E}_{x'_t, \cdots, x'_T \sim P_{LM}(\cdot | x_{\leq t-1})}[\exp(-E_\theta(x_{\leq t-1}, x'_t, \cdots, x'_T))]}$$

https://arxiv.org/pdf/2004.11714.pdf

Energy models to constrain the decoding

Algorithm 1: Top-k Joint Sampling

Input: number of samples n drawn from P_{LM}, value of k in top-k

`// Get a set of samples from` P_{LM}

sample n samples $\{x^1, \cdots, x^n\}$ from P_{LM} with top-k sampling

calculate energies $s^i = E_\theta(x^i)$ for each $x^i \in \{x^1, \cdots, x^n\}$

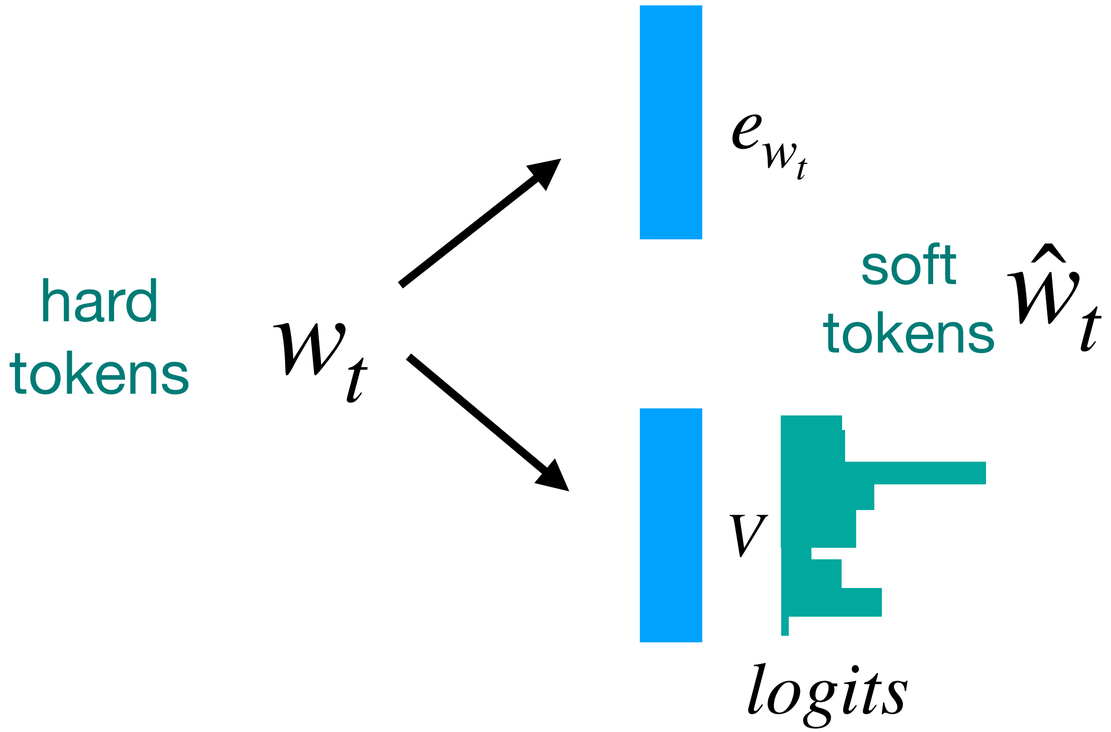
`// Resample from the set of LM samples`

sample $x = x^i$ with probability $\frac{\exp(-s^i)}{\sum_{j=1}^{n} \exp(-s^j)}$

return x

More practical sampling algorithm but still expensive

Gradient-based guidance



$$\hat{w}_t = \hat{w}_{t-1} - \mu \nabla_{\hat{w}} E(\hat{w}_{t-1}) + \epsilon^t$$

Langevin updates over soft tokens

$$w_t \leftarrow \hat{w}_t$$

Decode back to hard tokens
relying on LM

Incorporate constraints and LM into energy model
using **gradient-based** methods for finding low energy solutions

Problem: text is discrete
Shift to **soft** tokens i.e. probabilities over vocabulary

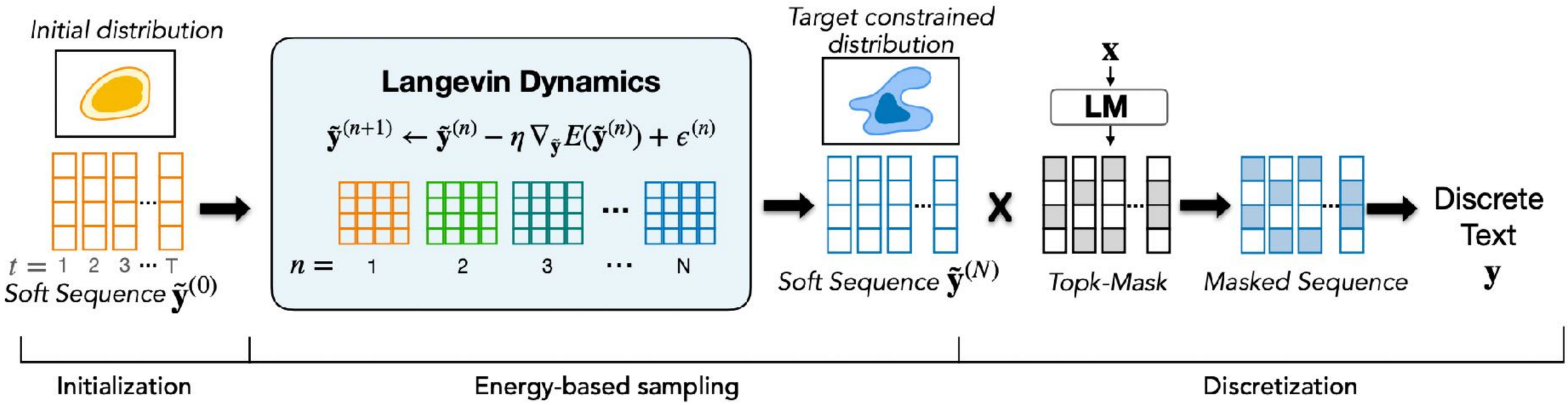
COLD Decoding: Energy-based Constrained Text Generation with Langevin Dynamics

Qin et al 2022

hard token w_t ⟶ V soft tokens

L x V parameters

logits



https://arxiv.org/pdf/2202.11705.pdf

MuCoLA: Constrained Sampling from Language Models via Langevin Dynamics in Embedding Spaces

Kumar et al 2022

hard tokens w_t ⟶  soft tokens $\hat{w}_t$ (embeddings)

e_{w_t}

$$\hat{w}_t = Proj(\hat{w}_{t-1} - \mu \nabla_{\hat{w}} E(\hat{w}_{t-1}) + \epsilon^t)$$

projecting embeddings to word vectors (crucial in practice)

Formulate constrains as Lagrangian (e.g. p_toxic < eps)

- More parameter efficient ($L \cdot d$ *vs* $L \cdot V$ parameters)
- Main results: in general high fluency with 95% sat. constraints, lower diversity

https://arxiv.org/pdf/2205.12558.pdf

Results on lexically constrained decoding

	Coverage		Perplexity
	Count	Percent	(GPT2-XL)
TSMH	2.72	71.27	1545.15
Neurologic	3.30	91.00	**28.61**
COLD	**4.24**	**94.5**	54.98
MuCoLa	4.07	93.8	31.48

coverage as average count of **desired keywords** in the output and the fraction of the outputs containing all desired keywords

- COLD is less fluent than MuCoLa, but more constraint-satisfiable

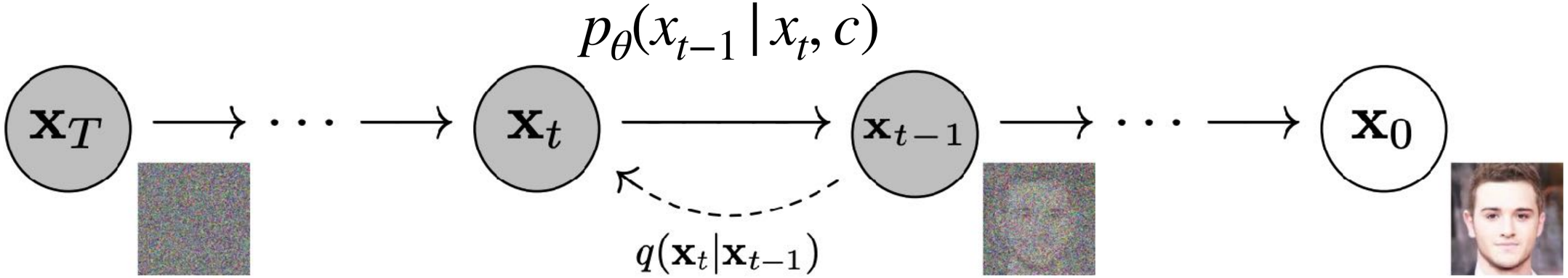
Summary

- For the new tasks, in-context learning can be used to adapt the model when we cannot allow training. Efficient fine-tuning should be used when we can afford some training

- When we want to leave the model intact (preserve the fluency) but constrain the output we may use the decoding guidance techniques (the model remains frozen)

Part 2: Controllable Text-to-Image Generation

DDPM: Denoising Diffusion Probabilistic Models



https://arxiv.org/pdf/2006.11239.pdf

- Conditional generation. “c” may be class label, text, another image, …

Text-2-image diffusion models

- **GLIDE**, https://arxiv.org/abs/2112.10741
- **Imagen**, https://imagen.research.google/paper.pdf
- **Stable diffusion** (latent diffusion), https://github.com/CompVis/stable-diffusion
- **DALL-E-2**, https://cdn.openai.com/papers/dall-e-2.pdf

Main text control ingredients

Text Conditioning
Cross Attention

$$p_\theta(x_{t-1} \mid x_t, c)$$

Classfier guidance

$$\hat{\mu}_\theta(\mathbf{x}_t|y) = \mu_\theta(\mathbf{x}_t|y) + s \cdot \Sigma_\theta(\mathbf{x}_t|y) \nabla_{\mathbf{x}_t} \log p_\phi(y|\mathbf{x}_t)$$

gradient from classifier

CLIP guidance

$$\hat{\mu}_\theta(\mathbf{x}_t|c) = \mu_\theta(\mathbf{x}_t|c) + s \cdot \Sigma_\theta(\mathbf{x}_t|c) \nabla_{\mathbf{x}_t} \log(f(\mathbf{x}_t) \cdot g(c))$$

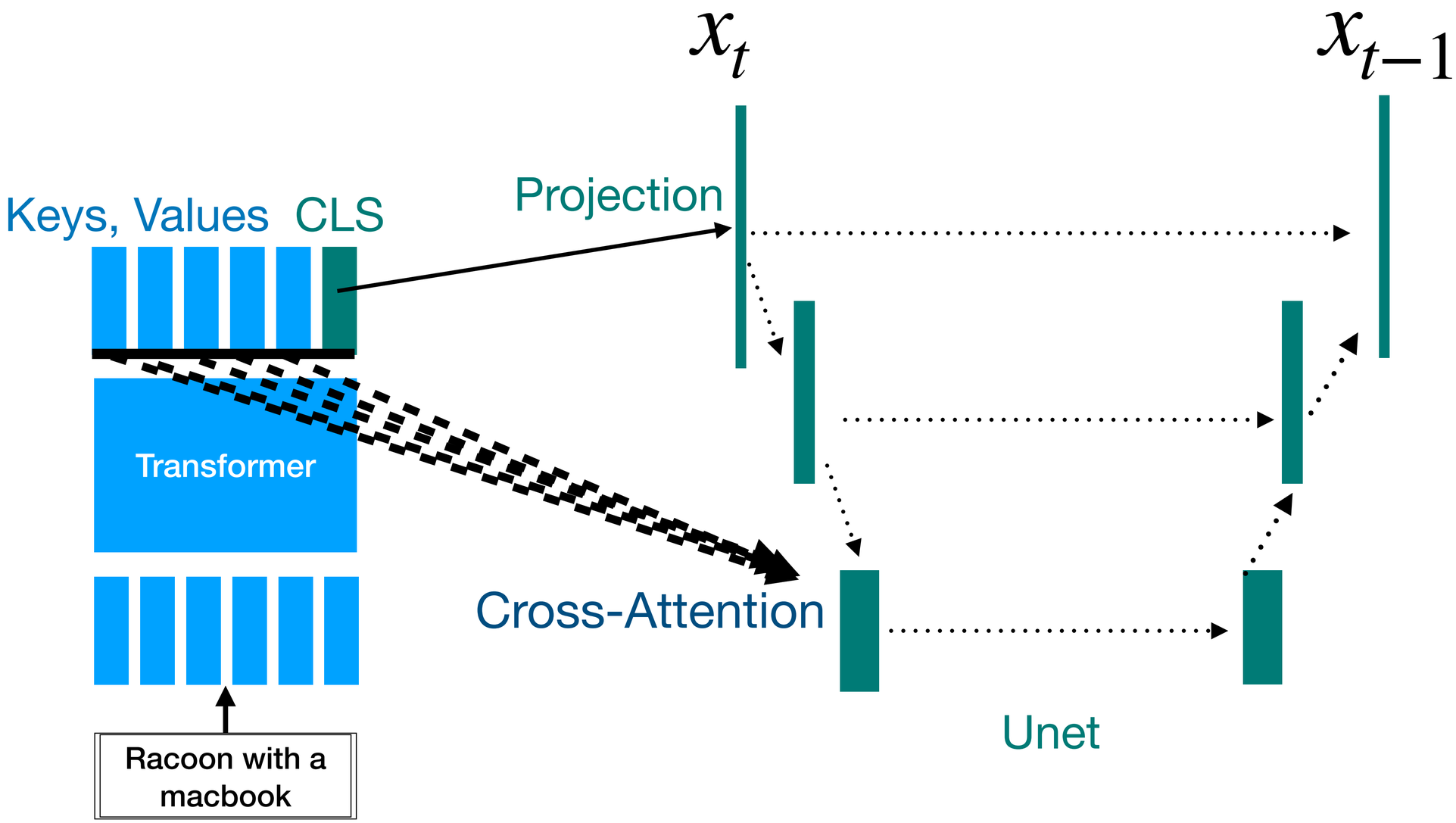
gradient from CLIP score

Classifier-free
guidance

$$\hat{\epsilon}_\theta(\mathbf{x}_t) = \epsilon_\theta(\mathbf{x}_t) + s \cdot (\epsilon_\theta(\mathbf{x}_t|y) - \epsilon_\theta(\mathbf{x}_t))$$

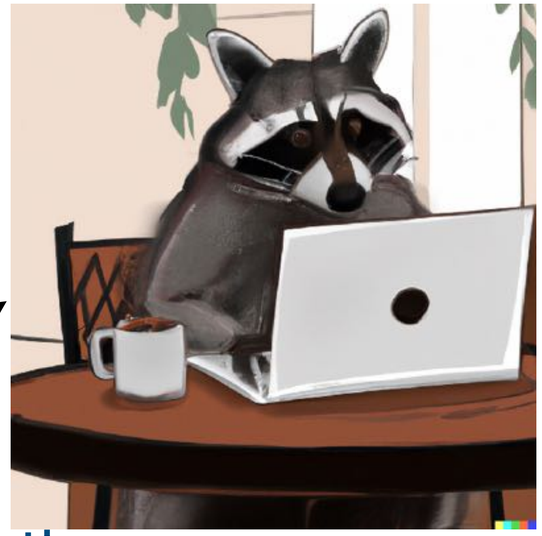
extrapolation towards conditioned output

Text conditioning



Text conditioning: super resolution

64x64



256x256

1024x1024

Cross-Attention

Transformer

Racoon with a macbook

- Upscaling the images

Classifier-guidance weight

$$\hat{\mu}_\theta(\mathbf{x}_t|y) = \mu_\theta(\mathbf{x}_t|y) + s \cdot \Sigma_\theta(\mathbf{x}_t|y) \nabla_{\mathbf{x}_t} \log p_\phi(y|\mathbf{x}_t)$$

"Pembroke Welsh corgi"

classifier scale 1.0 (FID: 33.0)

classifier scale 10.0 (FID: 12.0)

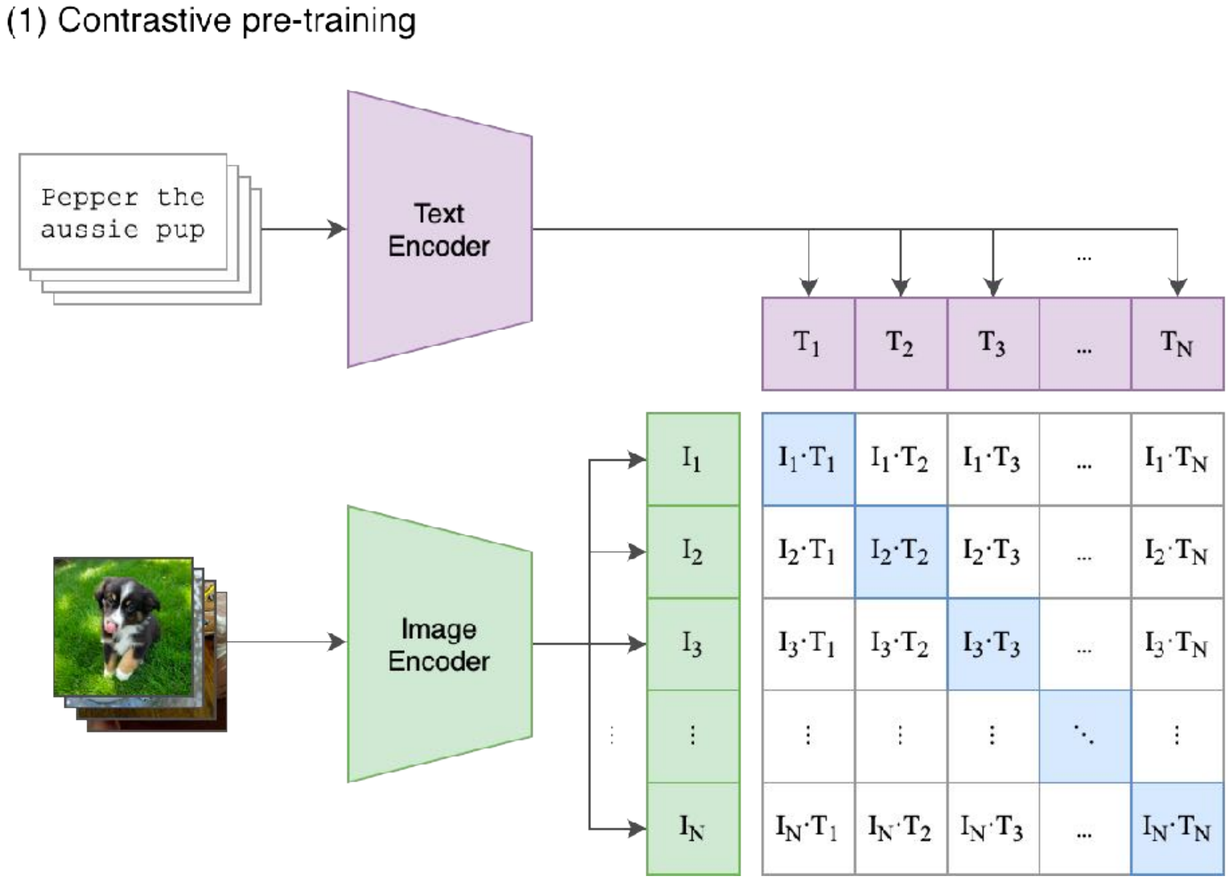
- Increasing the weight of the **classifier guidance** improves the FID (at the cost of diversity)

"Diffusion Models Beat GANs on Image Synthesis"

CLIP guidance

$$\hat{\mu}_\theta(\mathbf{x}_t|c) = \mu_\theta(\mathbf{x}_t|c) + s \cdot \Sigma_\theta(\mathbf{x}_t|c)\, \nabla_{\mathbf{x}_t} \log(f(\mathbf{x}_t) \cdot g(c))$$

CLIP score



Radford et al. 2021

Zero-shot capabilities

“More Control for Free! Image Synthesis with Semantic Diffusion Guidance”

$\nabla_{x_t} F(x_t, \cdot)$

Language Guidance $$F(x_t, l, t) = E'_I(x_t, t) \cdot E_L(l),$$

Image Content Guidance $$F(x_t, x'_t, t) = E'_I(x_t, t) \cdot E'_I(x'_t, t).$$

Image Content Guidance (Spatial) $$F(x_t, x'_t, t) = -\sum_j \frac{1}{C_j H_j W_j} ||E'_I(x_t, t)_j - E'_I(x'_t, t)_j||_2^2$$

Image Style Guidance $$F(x_t, x'_t, t) = -\sum_j ||G'_I(x_t, t)_j - G'_I(x'_t, t)_j||_F^2,$$

Multimodal Guidance $$F_{\phi_0}(x_t, y, t) = s_1 F_{\phi_1}(x_t, y, t) + s_2 F_{\phi_2}(x_t, y, t).$$

- Control different aspects of generation through classifier-guidance

Image Content Guidance

$$F(x_t, x'_t, t) = E'_I(x_t, t) \cdot E'_I(x'_t, t).$$

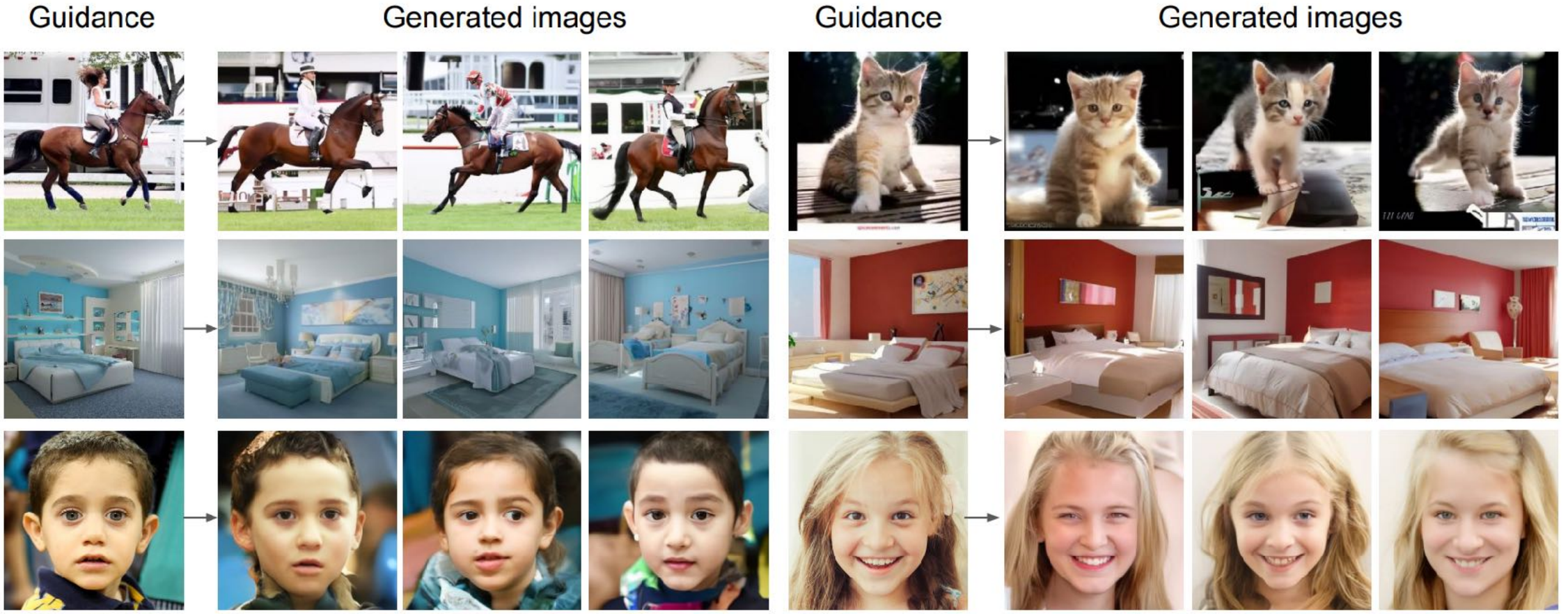
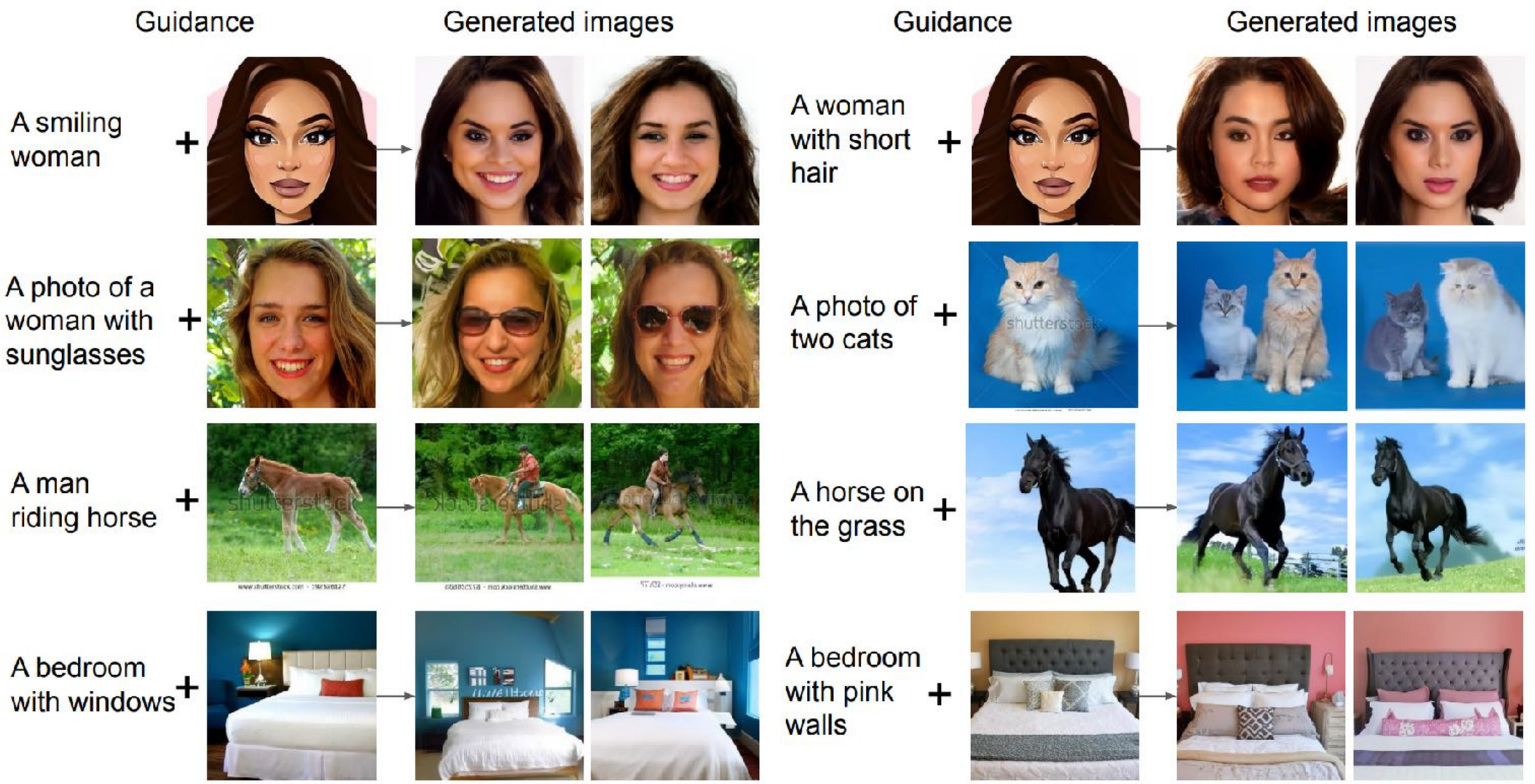


Image content + language content guidance

$$E'_I(x_t, t) \cdot E_L(l) + E'_I(x_t, t) \cdot E'_I(x'_t, t)$$



Structure-preserving & style guidance

(a) Style guidance

$$-\sum_j ||G'_I(x_t,t)_j - G'_I(x'_t,t)_j||_F^2,$$

Gram matrices

(b) Structure-preserving guidance

$$\sum_j \frac{1}{C_j H_j W_j} ||E'_I(x_t,t)_j - E'_I(x'_t,t)_j||_2^2$$

Spatial features

Classifier-free guidance

$$\hat{\epsilon}_\theta(x_t|y) = \epsilon_\theta(x_t|\emptyset) + s \cdot (\epsilon_\theta(x_t|y) - \epsilon_\theta(x_t|\emptyset))$$

unconditioned (empty text) — conditioned — unconditioned (empty text)

Extrapolating predictions

- A single model! Do not require separate classifier
- Requires unconditional + conditional pre-training

https://arxiv.org/pdf/2112.10741.pdf

Power of classifier-free guidance

GLIDE (unguided)

GLIDE (CLIP guided, scale 2.0)

"a stained glass window of a panda eating bamboo"

GLIDE (classifier-free guidance, scale 3.0)

- **Classifier-free guidance** wins in terms of photorealism and similarity to caption

Power of classifier-free guidance

- **Classifier-free guidance** wins in terms of photorealism and similarity to caption

https://arxiv.org/pdf/2112.10741.pdf

(Text-) Controllable Image Editing & Generation

Importance of image-text cross-attention

- “Imagen: Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”
  Importance of text encoder

- “Prompt-to-Prompt Image Editing with Cross Attention Control”
  Control via cross-attention

- “DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation”
  Personalisation

Imagen: the importance of text encoder

Sprouts in the shape of text 'Imagen' coming out of a fairytale book.

A photo of a Shiba Inu dog with a backpack riding a bike. It is wearing sunglasses and a beach hat.

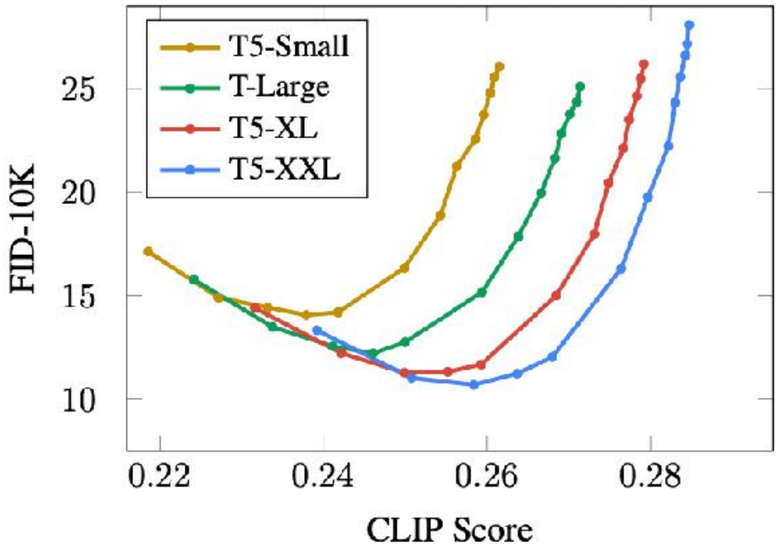
A high contrast portrait of a very happy fuzzy panda dressed as a chef in a high end kitchen making dough. There is a painting of flowers on the wall behind him.

https://imagen.research.google/paper.pdf

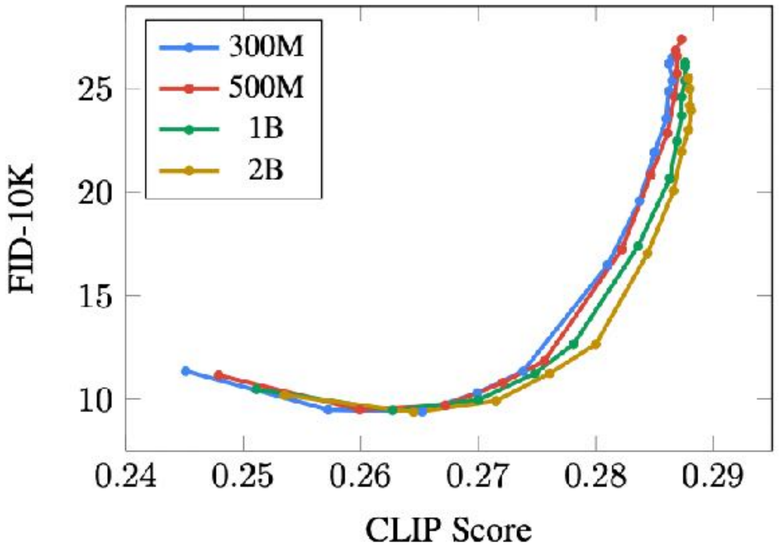
Imagen takeaways

Imagen used powerful frozen T5-XXL text encoder. Scaling text encoder size is more important than U-Net size

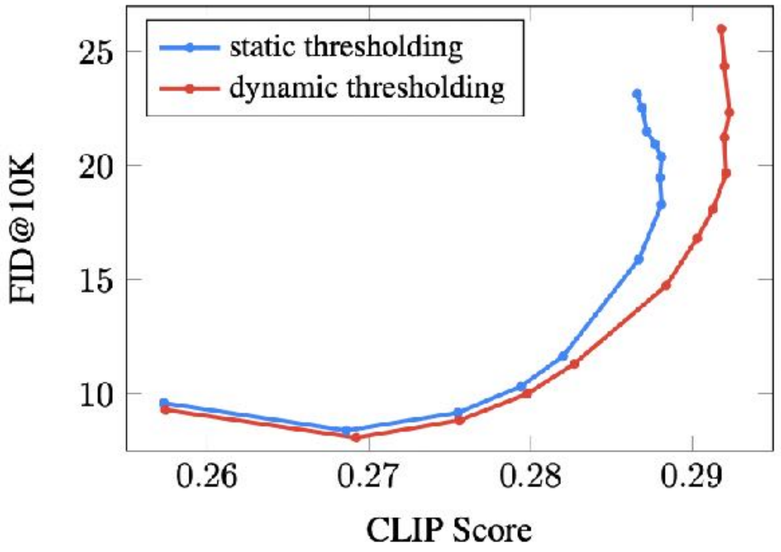
FID w.r.t. text alignment



(a) Impact of encoder size.



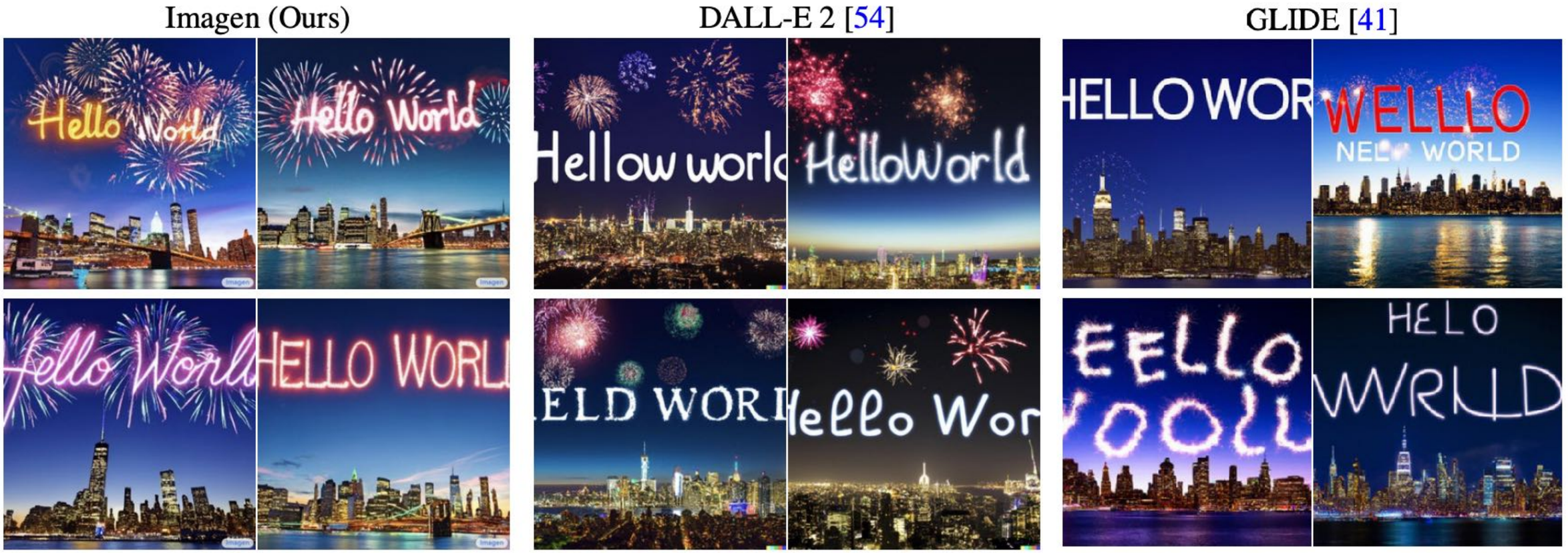
(b) Impact of U-Net size.



(c) Impact of thresholding.

https://imagen.research.google/paper.pdf

Imagen takeaways

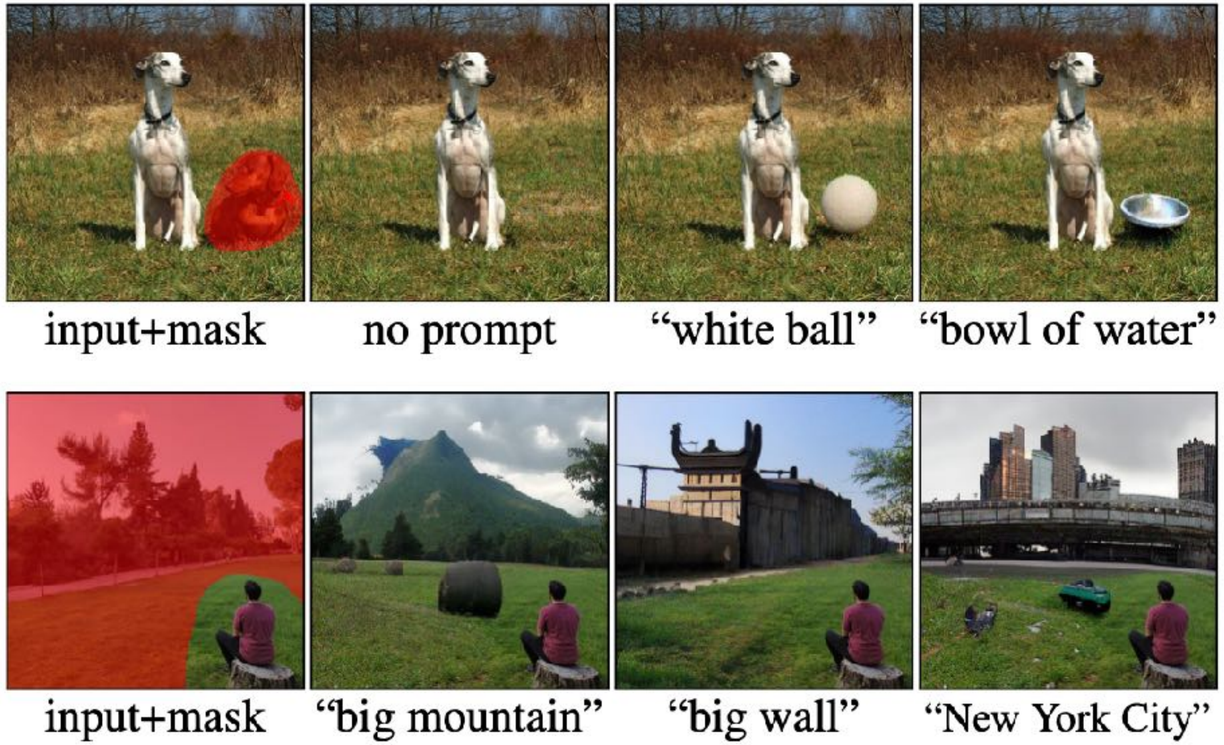


New York Skyline with Hello World written with fireworks on the sky.

- Imagen can render text thanks to the large text T5-XXL encoder

https://imagen.research.google/paper.pdf

Local edits: how to edit an object preserving the scene?

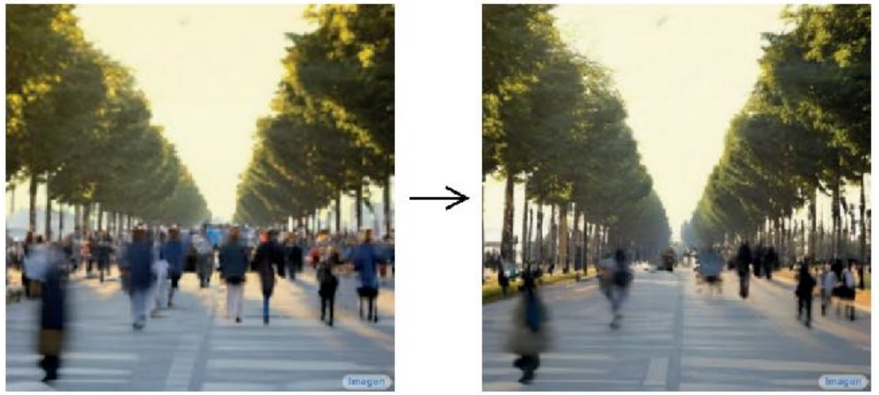


“Blended Diffusion for Text-driven Editing of Natural Images”, Kim et al. 2022

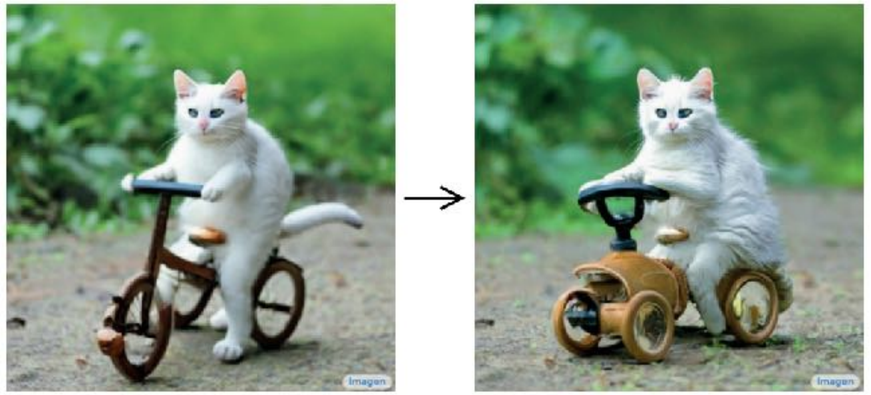
https://arxiv.org/pdf/2111.14818.pdf

- CLIP + masks allow the user to select the area to regenerate

“Prompt-to-Prompt Image Editing with Cross Attention Control”



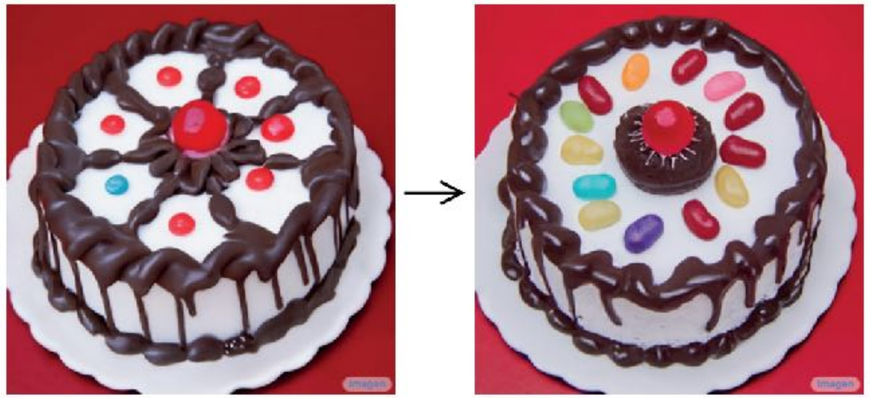
“The boulevards are crowded today.”



“Photo of a cat riding on a bicycle.”

car

“Children drawing of a castle next to a river.”



“a cake with decorations.”

jelly beans

- Control by reusing attention (**no masks required**)

https://arxiv.org/pdf/2208.01626.pdf

Source image and prompt:

"photo of a cat riding on a bicycle."

bicycle → motorcycle

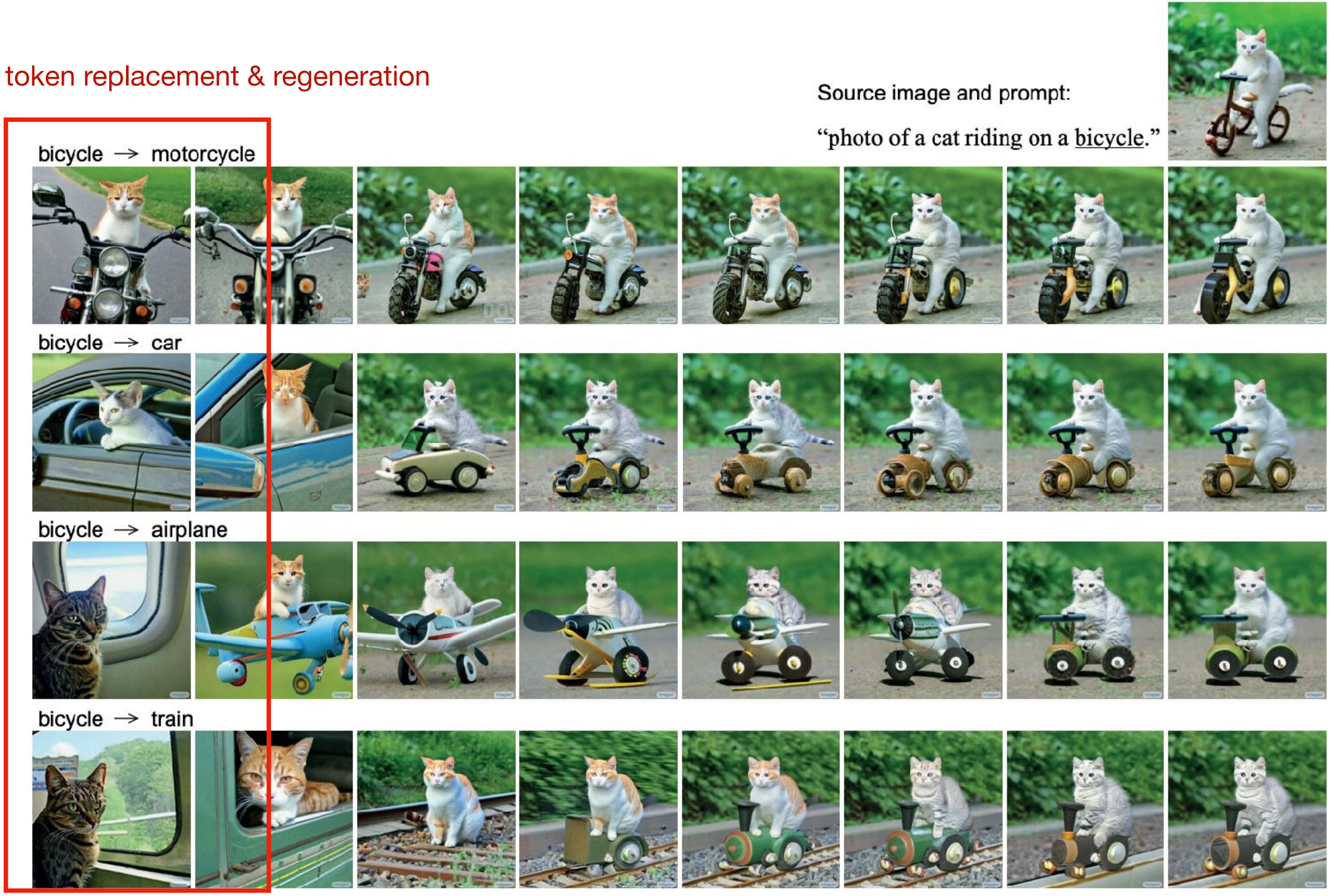
bicycle → car

bicycle → airplane

bicycle → train

W.O. attention injection → Full attention injection

Simple token replacement & regeneration



Fixing attention & regeneration

Source image and prompt:

"photo of a cat riding on a bicycle."

bicycle → motorcycle

bicycle → car

bicycle → airplane

bicycle → train

W.O. attention injection → Full attention injection

Fixed random seed

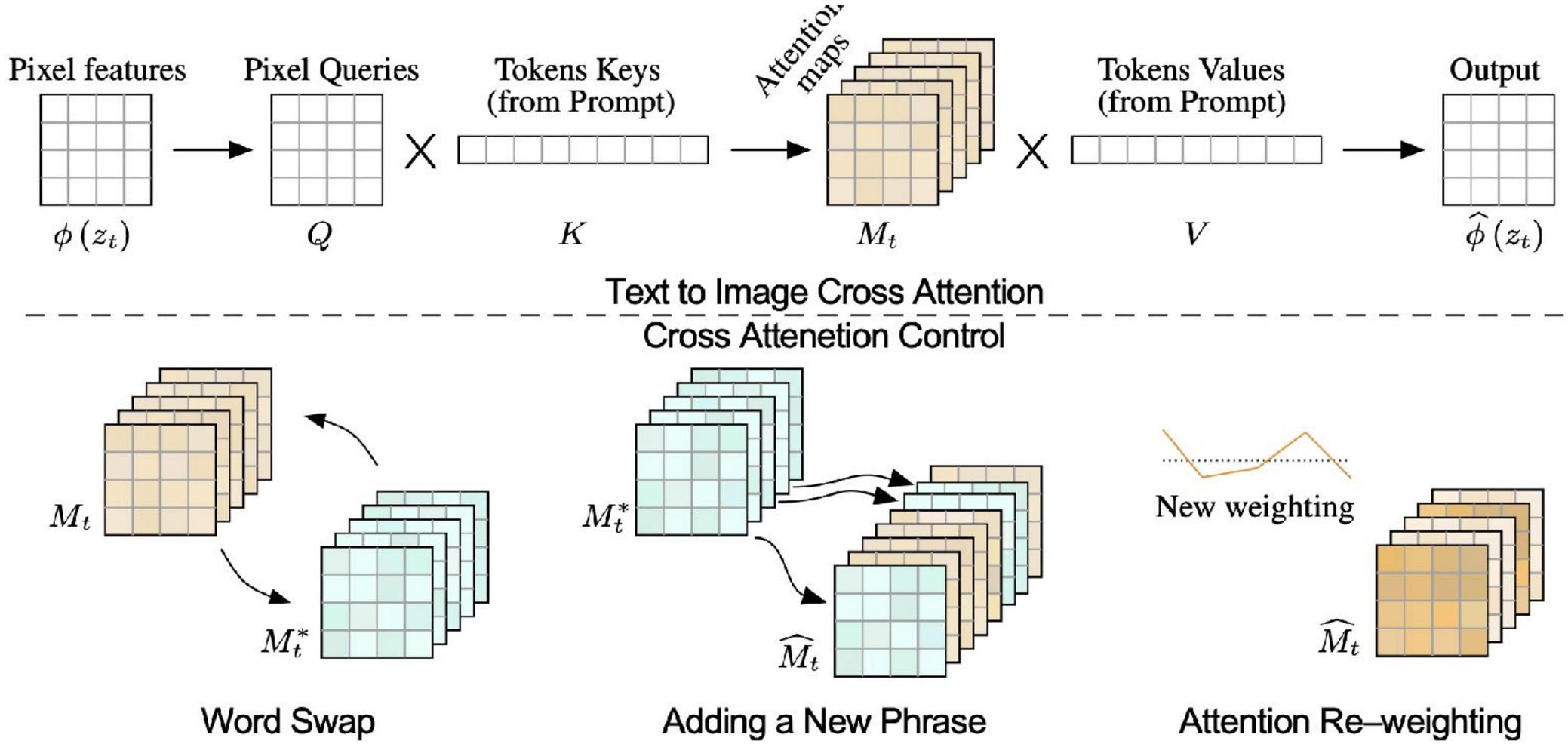
- Fixing random seed is not enough for generating consistent images

Fixed attention and random seed

Fixed attention maps and random seed

- With attention & random seed fixing

Prompt-to-Prompt Image Editing with Cross Attention Control



https://arxiv.org/abs/2208.01626

Prompt-to-Prompt

Algorithm 1: Prompt-to-Prompt image editing

Input: A source prompt $\mathcal{P}$, a target prompt $\mathcal{P}^*$, and a random seed s.
Output: A source image x_{src} and an edited image x_{dst}.
$z_T \sim N(0, I)$ a unit Gaussian random variable with random seed s;
$z_T^* \leftarrow z_T$;
for $t = T, T-1, \ldots, 1$ **do**
    $z_{t-1}, M_t \leftarrow DM(z_t, \mathcal{P}, t, s)$;
    $M_t^* \leftarrow DM(z_t^*, \mathcal{P}^*, t, s)$;
    $\widehat{M}_t \leftarrow Edit(M_t, M_t^*, t)$;
    $z_{t-1}^* \leftarrow DM(z_t^*, \mathcal{P}^*, t, s_t)\{M \leftarrow \widehat{M}_t\}$;
end
Return (z_0, z_0^*)

calculate z with old prompt

Prompt-to-Prompt

Algorithm 1: Prompt-to-Prompt image editing

Input: A source prompt $\mathcal{P}$, a target prompt $\mathcal{P}^*$, and a random seed s.
Output: A source image x_{src} and an edited image x_{dst}.
$z_T \sim N(0, I)$ a unit Gaussian random variable with random seed s;
$z_T^* \leftarrow z_T$;
for $t = T, T-1, \ldots, 1$ **do**
    $z_{t-1}, M_t \leftarrow DM(z_t, \mathcal{P}, t, s)$;
    $M_t^* \leftarrow DM(z_t^*, \mathcal{P}^*, t, s)$;
    $\widehat{M}_t \leftarrow Edit(M_t, M_t^*, t)$;
    $z_{t-1}^* \leftarrow DM(z_t^*, \mathcal{P}^*, t, s_t)\{M \leftarrow \widehat{M}_t\}$;
end
Return (z_0, z_0^*)

get attention map from an old prompt

Prompt-to-Prompt

Algorithm 1: Prompt-to-Prompt image editing

Input: A source prompt $\mathcal{P}$, a target prompt $\mathcal{P}^*$, and a random seed s.
Output: A source image x_{src} and an edited image x_{dst}.
$z_T \sim N(0, I)$ a unit Gaussian random variable with random seed s;
$z_T^* \leftarrow z_T$;
for $t = T, T-1, \ldots, 1$ **do**
    $z_{t-1}, M_t \leftarrow DM(z_t, \mathcal{P}, t, s)$;
    $M_t^* \leftarrow DM(z_t^*, \mathcal{P}^*, t, s)$;
    $\widehat{M}_t \leftarrow Edit(M_t, M_t^*, t)$; inject attention map
    $z_{t-1}^* \leftarrow DM(z_t^*, \mathcal{P}^*, t, s_t)\{M \leftarrow \widehat{M}_t\}$;
end
Return (z_0, z_0^*)

Prompt-to-Prompt

Algorithm 1: Prompt-to-Prompt image editing

Input: A source prompt $\mathcal{P}$, a target prompt $\mathcal{P}^*$, and a random seed s.
Output: A source image x_{src} and an edited image x_{dst}.
$z_T \sim N(0, I)$ a unit Gaussian random variable with random seed s;
$z_T^* \leftarrow z_T$;
for $t = T, T-1, \ldots, 1$ **do**
    $z_{t-1}, M_t \leftarrow DM(z_t, \mathcal{P}, t, s)$;
    $M_t^* \leftarrow DM(z_t^*, \mathcal{P}^*, t, s)$;
    $\widehat{M}_t \leftarrow Edit(M_t, M_t^*, t)$;
    $z_{t-1}^* \leftarrow DM(z_t^*, \mathcal{P}^*, t, s_t)\{M \leftarrow \widehat{M}_t\}$;
end
Return (z_0, z_0^*)

calculate z with new prompt and injected attention

apples → oranges

apples → chocolates

apples → cookies

apples → kittens

apples → smoke

"A ball between two chairs on the beach.

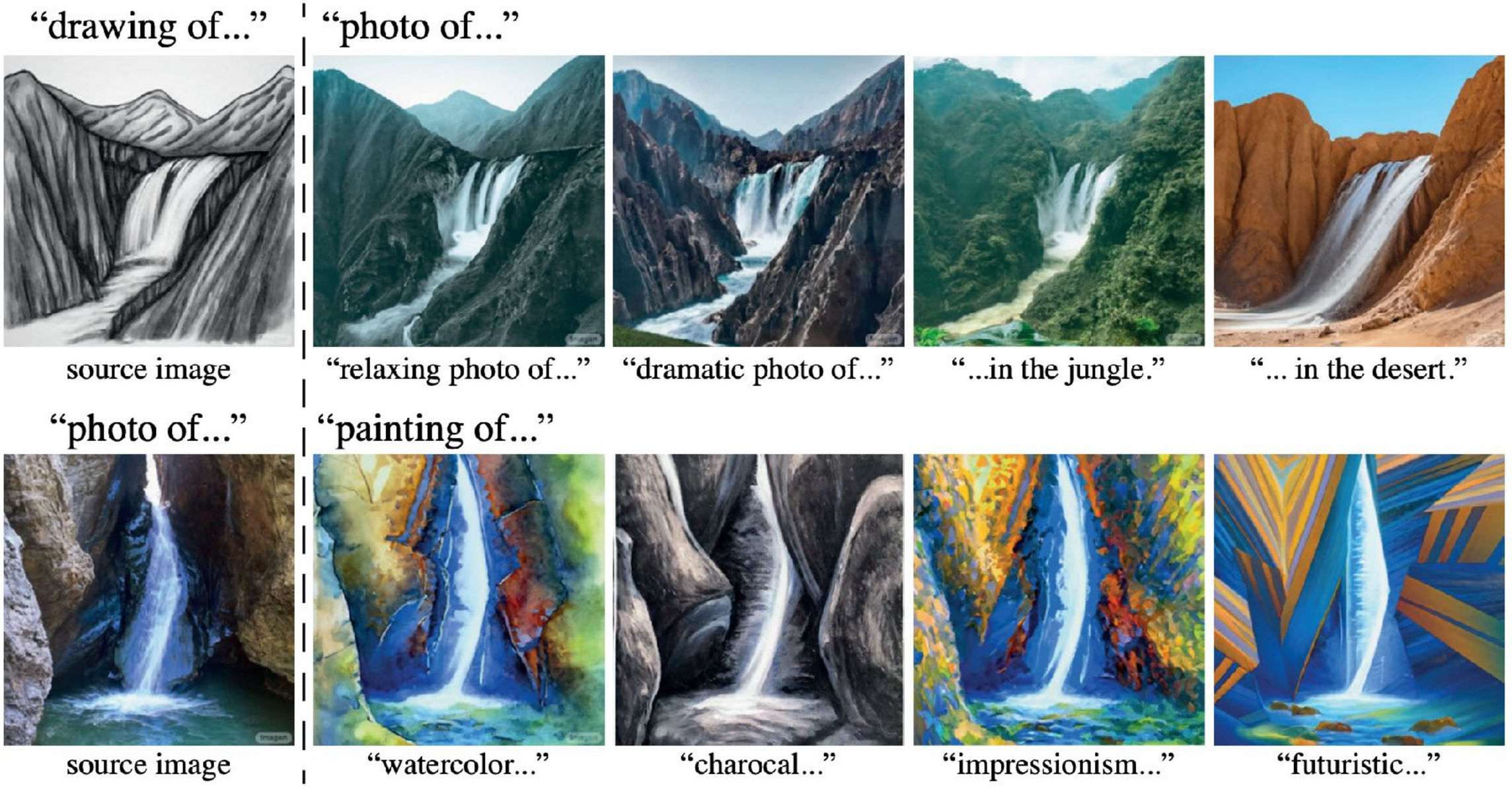
source image

ball → frisbee

ball → bucket

ball → palace

ball → turtle



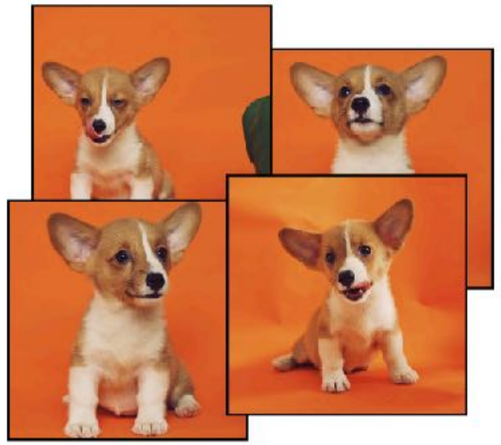
- Adding a New Phrase

"A photo of a birthday(↓) cake next to an apple."

"The picnic is ready under a blossom(↓) tree."

- Attention Re–weighting

Personalised Edits



Input images

in the Acropolis

swimming

sleeping

in a doghouse

in a bucket

getting a haircut

Input images

worn by a bear

in the jungle

on red fabric

at Mt. Fuji

on top of snow

with Eiffel Tower

https://arxiv.org/pdf/2208.12242.pdf

DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation

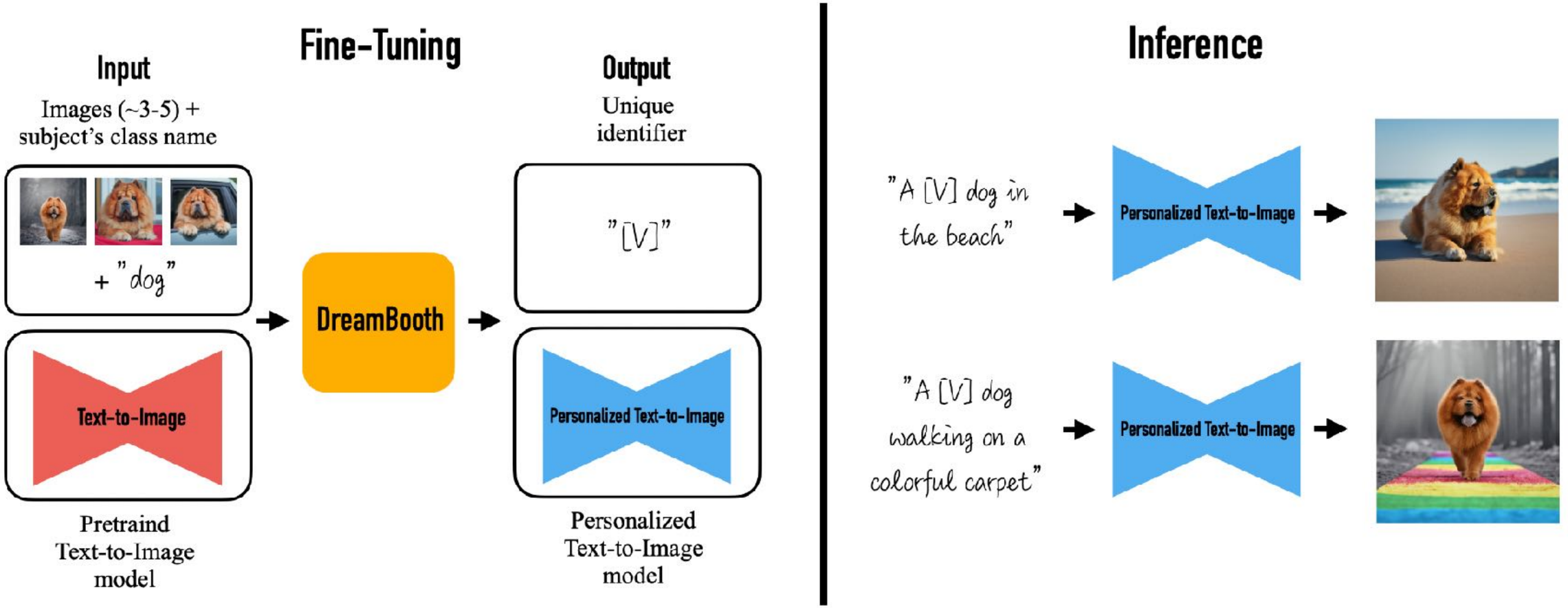
Ruiz et al. 2022

a [V] clock in the jungle

- Image/text guided approach are not good for generating of particular objects

https://dreambooth.github.io/

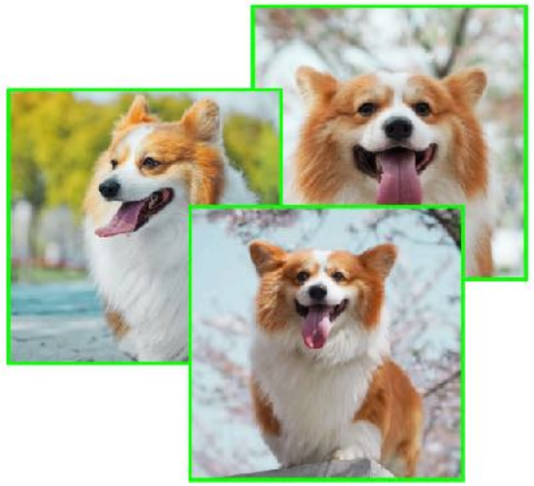
Personalization recipe



- Recipe: Take 3-5 images, finetune a rare token (e.g.“xxy5syt00”), use textual prompts
- Problem: overfitting, forgetting of similar objects

https://dreambooth.github.io/

Input images



Generating “A dog”

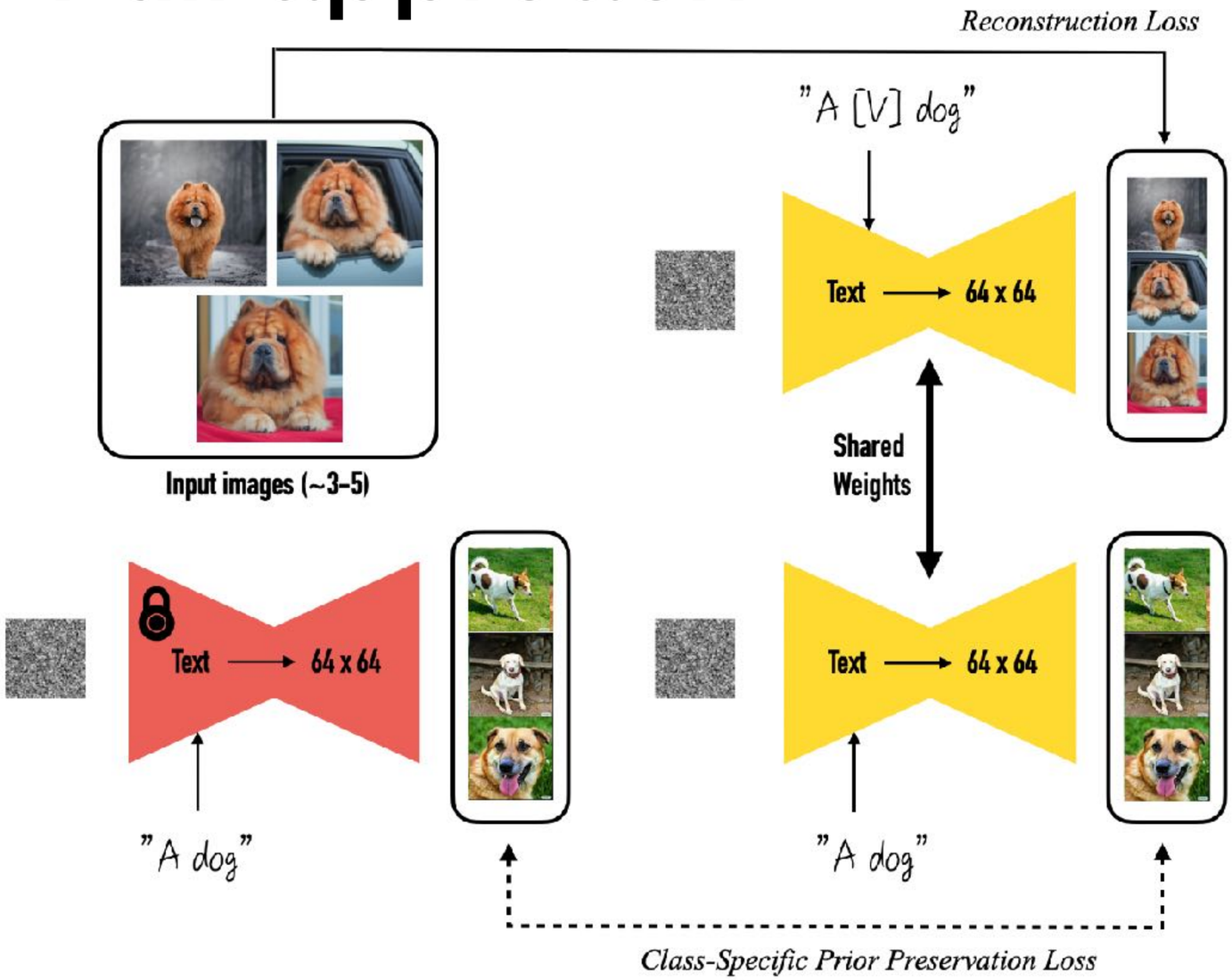
Vanilla model

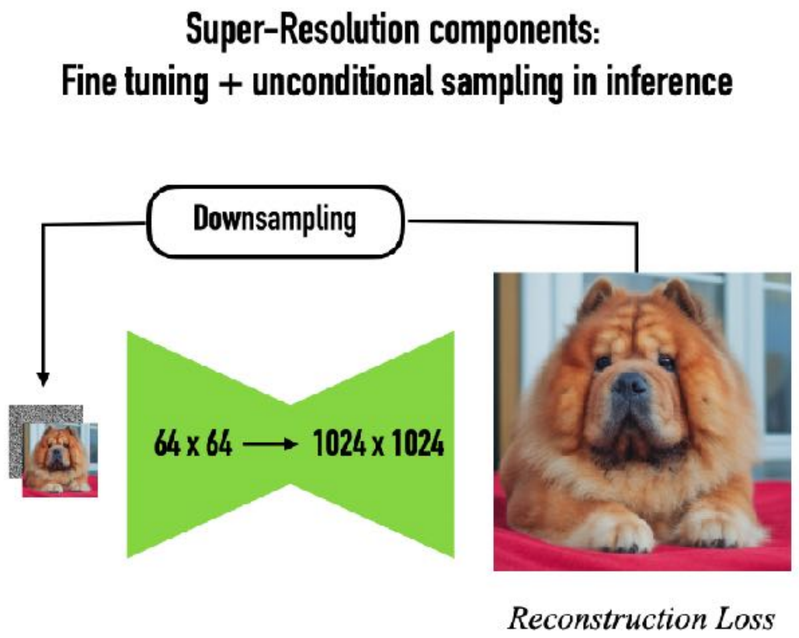
Ours w/o prior-preservation loss

Ours (full)

- Problem with naive finetuning: the model learns that all dogs are like yours

Full approach





- Expensive finetune with prior preservation loss

https://dreambooth.github.io/

Input images

A [V] backpack in the Grand Canyon

A [V] backpack with the night sky

A [V] backpack in the city of Versailles

A wet [V] backpack in water

A [V] backpack in Boston

https://dreambooth.github.io/

Summary

- Classifier-free guidance is most efficient for fidelity and caption matching (compared to CLIP guidance)
- Choosing the right classifier-guidance we can control for text matching, image matching, style, and spatial matching
- Cross-attention plays vital role for text-to-image synthesis, attention maps can be used for fine-grained editing
- Inverting human-provided images into tokens allows to use them for personalised generation