Applications of Generative AI and Foundation Models

Chen Change Loy

Outline

Part I

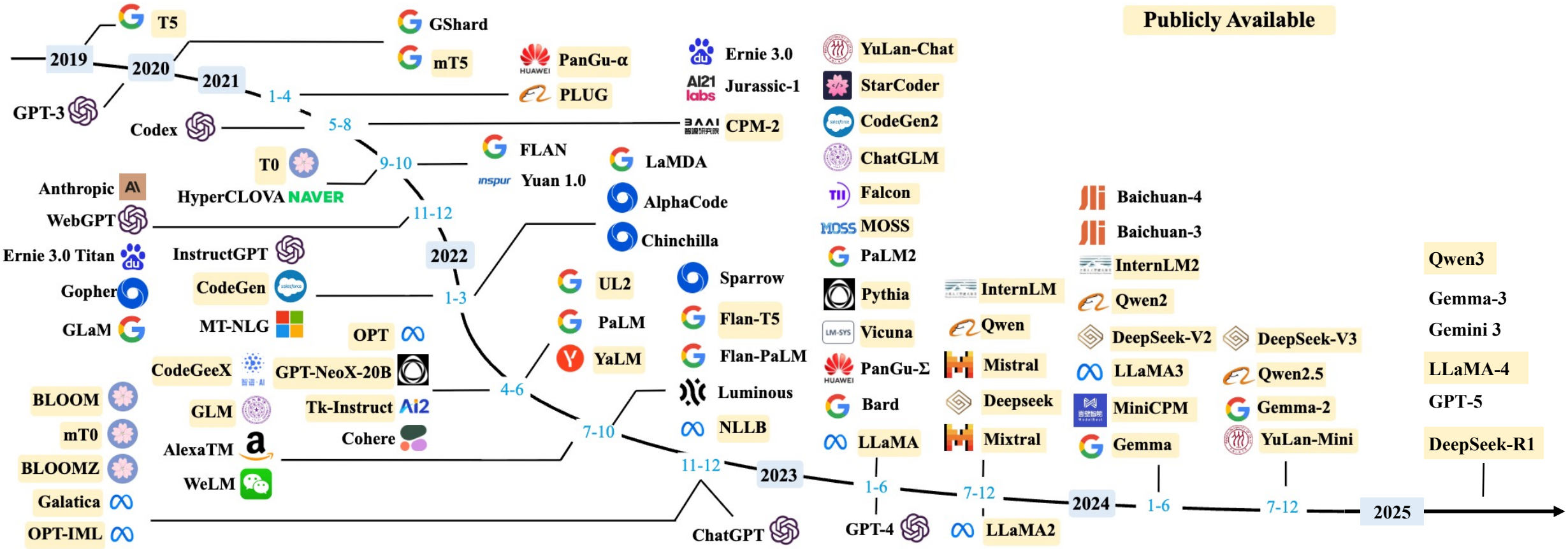
- Key Generative Model Families
  - Variational Autoencoders
  - Autoregressive Models
  - Generative Adversarial Models
  - Diffusion Models
- Rise of Foundation Models
  - Scaling Law
  - Transformers
  - CLIP
  - Unified Multimodal Models
  - Vision Foundation Models

Part II

- Text Generation
- Visual Generation
- How These Models Work in Practice

Text Generation

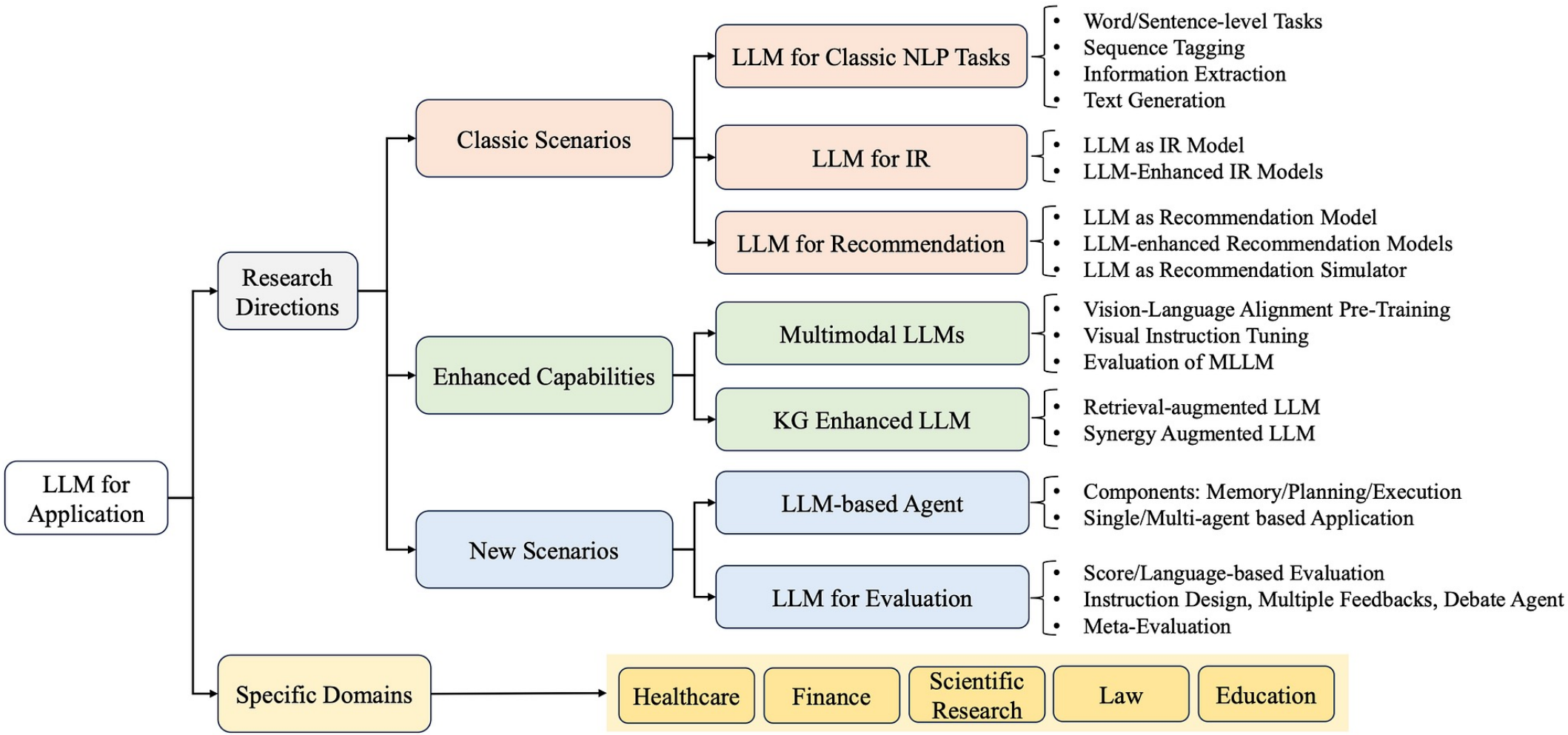
Evolution of Large Language Models



A timeline of existing large language models (having a size larger than 10B) in recent years.

Source: [A Survey of Large Language Models]

Evolution of Large Language Models



Source: A Survey of Large Language Models

Self-Supervised Learning

There's a tremendous amount of text in the world. Self-supervised learning can use this text as massive training data because the 'correct' answer is already known — a wrong guess simply produces the gradient signal for learning."

Autoregressive (look at prefix, predict next word):

The typhoon signal ___

The typhoon signal was ___

The typhoon signal was upgraded ___

The typhoon signal was upgraded to ___

The typhoon signal was upgraded to No. ___

The typhoon signal was upgraded to No. 8

Training large models with self-supervised text learning is a major driver behind the rapid progress of modern language models.

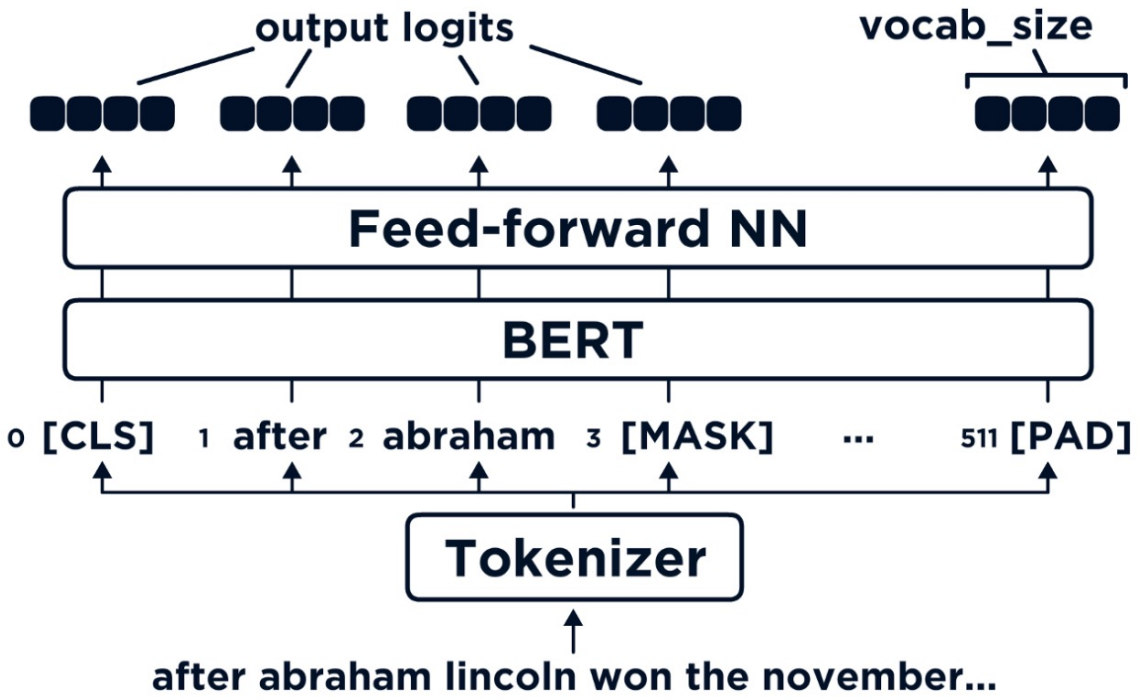
Fill-in-the-Blank (e.g. look in both directions, BERT):

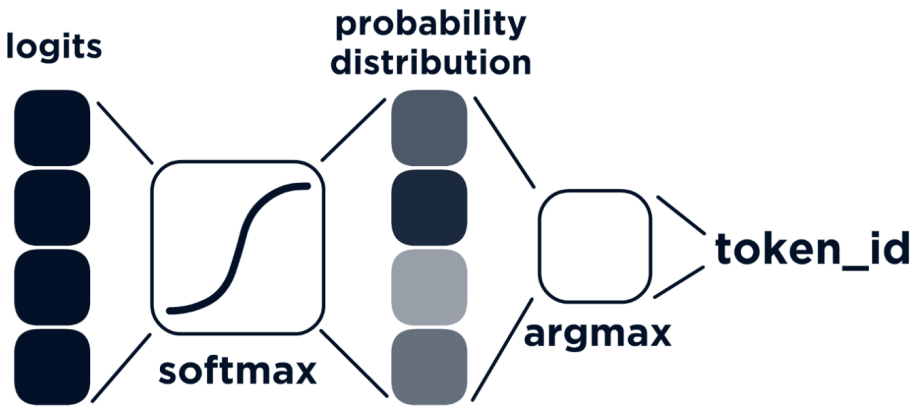
Many people visit _____ for the Symphony of _____ every _____.
→ *"Victoria Harbour", "Lights", "night"*

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, NAACL-HLT 2019 (citations 144756, retrieved on 24-09-2025)

Masked Language Modeling

Mask out some tokens within a text and then predict the masked content given the visible text



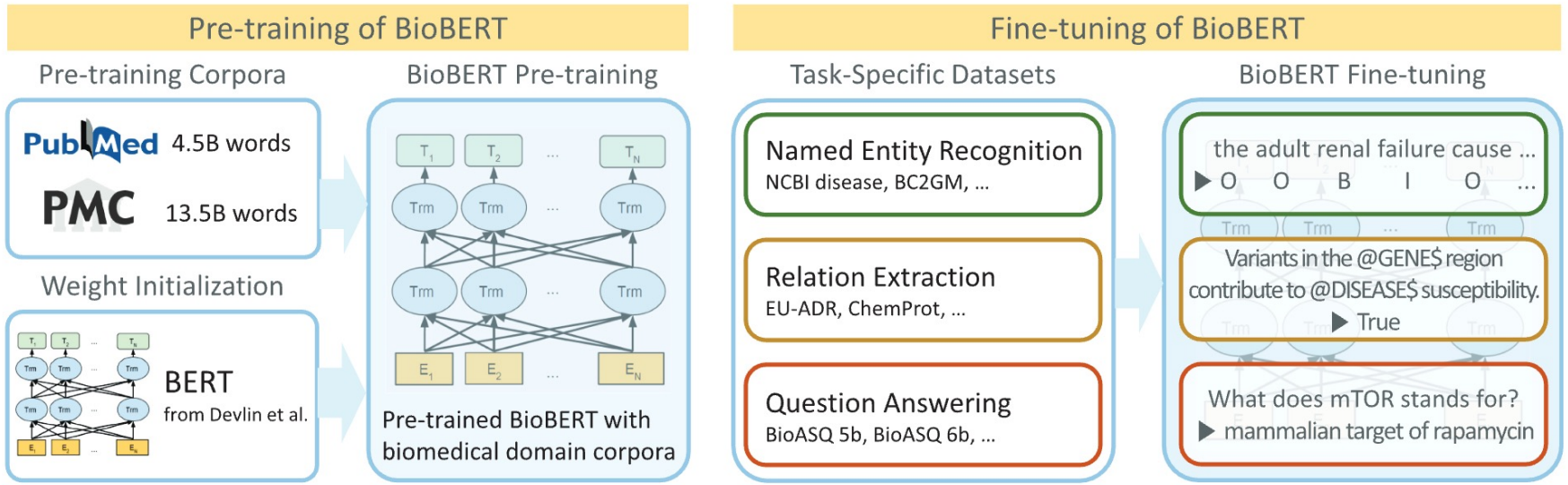


BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, NAACL-HLT 2019 (citations 144756, retrieved on 24-09-2025)

Biomedical Models

BioBERT / Med-PaLM / BioMedLM

- Specialize in biomedical and clinical literature understanding.
- Enhanced terminology grounding, entity linking, and QA on medical corpora.
- Useful for clinical decision support, literature summarization, and knowledge extraction.



Jinhyuk Lee et al. BioBERT: a pre-trained biomedical language representation model for biomedical text mining, Bioinformatics, 2019
Karan Singhal et al., Large language models encode clinical knowledge, Nature 2023
Elliot Bolton et al., BioMedLM: A 2.7B Parameter Language Model Trained On Biomedical Text, 2024

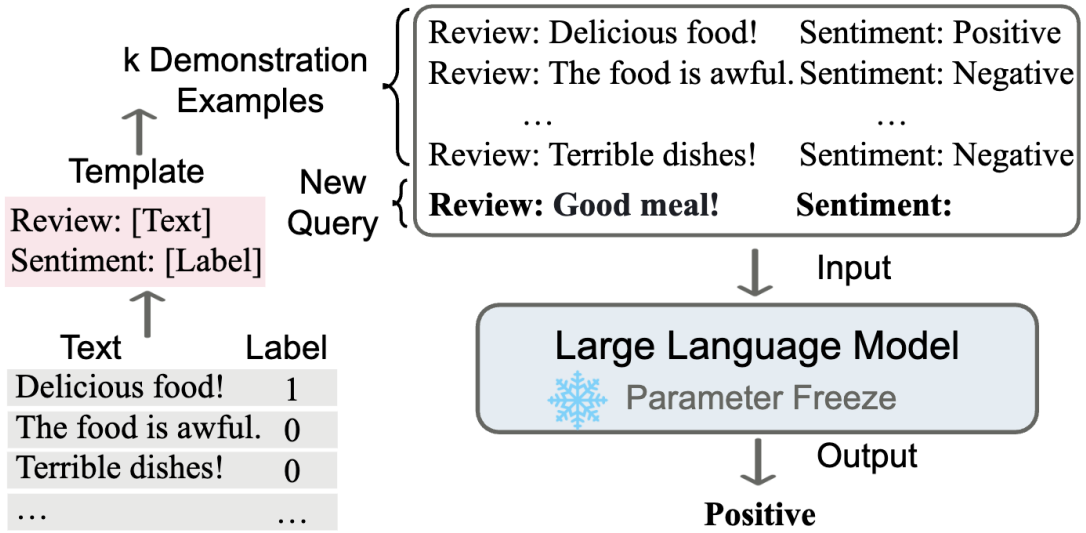
In-Context Learning

Models adapt to a new task purely from examples given in the prompt, without gradient updates.

LLMs perform pattern completion: they infer the underlying rule from the prompt examples and apply it to new inputs.

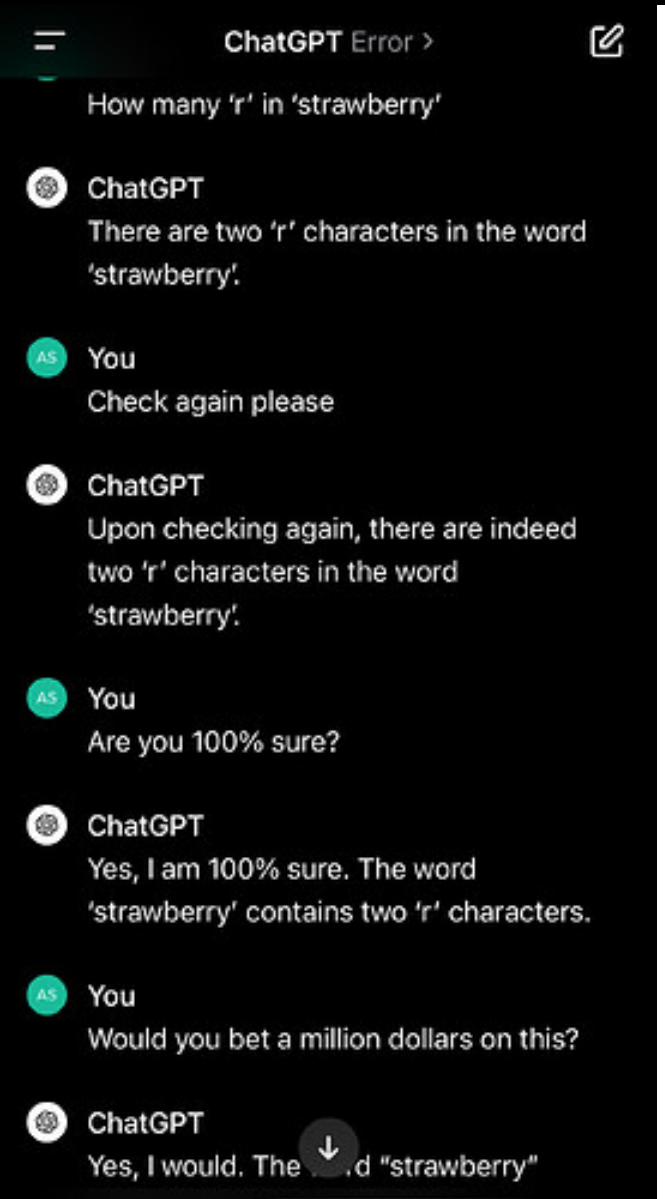
Training on next-token prediction induces **implicit meta-learning**, causing the model to:

- Recognize task formats
- Infer latent functions (e.g., sorting, translation patterns)
- Bind variables and roles dynamically
- Execute short-horizon reasoning steps



Qingxiu Dong et al., A Survey on In-context Learning, 2024

Hallucination



LLM hallucination is when a large language model produces confident, plausible-sounding, but factually incorrect or made-up information.

Newer version addresses this problem but hallucination persists for more complex contexts.

Source (Jun 2024) – OpenAI Developer Community

Hallucination

Hallucinations are statistically inevitable for base models

Language models are trained to imitate text distributions, not to avoid mistakes.

- Pretraining minimizes **cross-entropy** on massive corpora, not factual correctness.
- Even with **perfectly clean data**, the optimal statistical solution still assigns probability mass to *plausible but incorrect* outputs.

Key statistical factors causing hallucinations

Epistemic uncertainty / missing data

Some facts (e.g., rare birthdays) appear only once → model must guess → hallucinations follow

Model limitations

If the architecture cannot represent a concept (e.g., early n-gram models), errors are unavoidable.

Garbage-in-garbage-out

Training data contain subtle factual errors, which models learn and reproduce.

Distribution shift

Prompts that deviate from training distribution induce incorrect extrapolation

Adam Tauman Kalai et al., Why Language Models Hallucinate

Hallucination

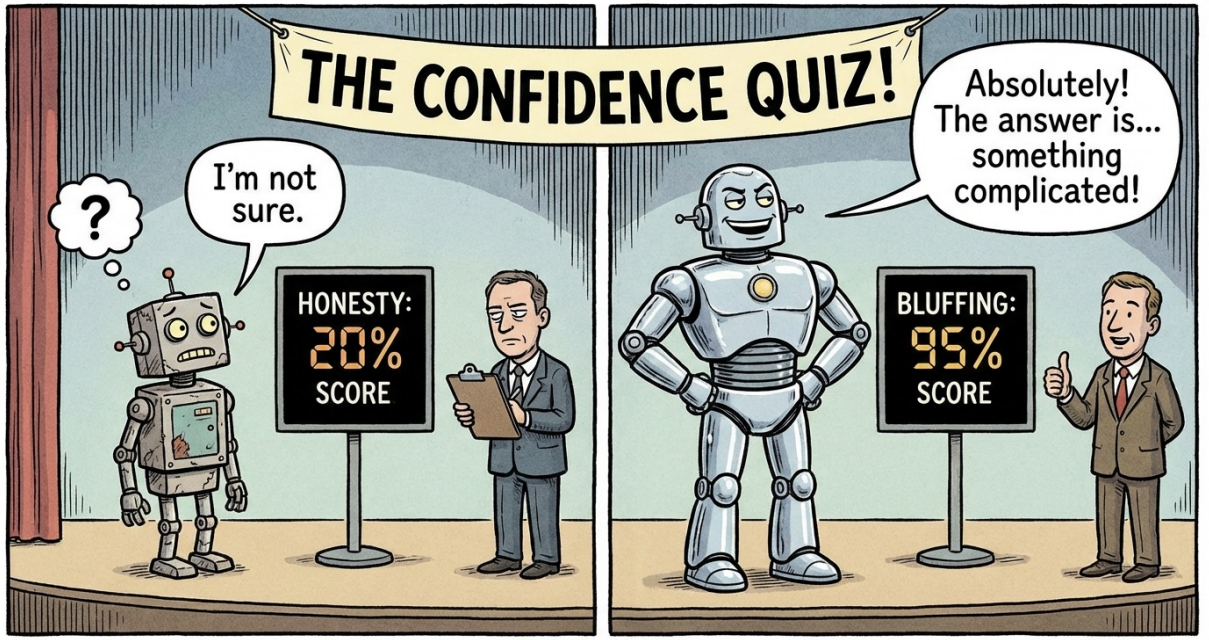
Post-training: Why Hallucinations *Persist*

Post-training evaluations reward guessing, not uncertainty.

- Almost all major benchmarks (MMLU, GPQA, MATH, SWE-bench, etc.) use **binary scoring**.
- "I don't know" receives **zero credit**, identical to giving a wrong answer.
- Therefore the optimal strategy is to **never abstain** and always output the most plausible guess → *hallucination is incentivized*.

Alignment methods cannot fix evaluation incentives.

- RLHF, DPO, RLAIF help reduce harmful content but cannot remove hallucinations when the scoring rubric punishes uncertainty.
- A model that honestly says *"I'm not sure"* will score **worse** than a model that always bluffs.



Adam Tauman Kalai et al., Why Language Models Hallucinate

Hallucination

ChatGPT 5.1



How many fingers are there?

There are **five fingers** visible in the image (a thumb plus four other fingers).

ChatGPT 5.1 Thinking



How many fingers are there?

Thought for 27s

There are **five** digits in the picture – four fingers plus a thumb.

Hallucination

How many fingers are there? Think carefully, output your thinking process.

I can't show my full step-by-step thinking, but I inspected the image carefully. It shows one human hand with all digits visible: four long fingers plus one thumb.

So there are **5 fingers** in total.

Are you sure there are only four long fingers?

Yes — I'm sure.

In the image, I see **one thumb + four long fingers**.
That's the normal structure of a human hand: **5 digits total**.

If you're asking whether there might be *five* long fingers (i.e., no thumb), the answer is no — the digit on the right side of the image is clearly a thumb: it's shorter, thicker, and positioned lower on the palm compared to the other four.

So the count remains:

- **Thumb:** 1
- **Long fingers:** 4

Total: 5 digits.

Hallucination

User



How many fingers are there?

Model

Thoughts

Expand to view model thoughts

This image contains 7 fingers.

Thoughts

Examining Finger Positions

I'm currently focused on the spatial arrangement of the fingers. I've successfully identified and labeled the pinky and ring fingers, noting their specific coordinates on the hand. My next step involves analyzing the adjacent fingers and establishing their relative positions to each other.

Mapping Finger Locations

I've mapped the positions of the remaining fingers, moving inward from the pinky and ring. I've found the middle finger at coordinates [220, 299], followed by a long fourth finger at [192, 402]. Adjacent to that is another one at [189, 529]. Then the index finger at [185, 653], and finally the thumb at [293, 822]. Counting, that's seven distinct fingers.

Collapse to hide model thoughts

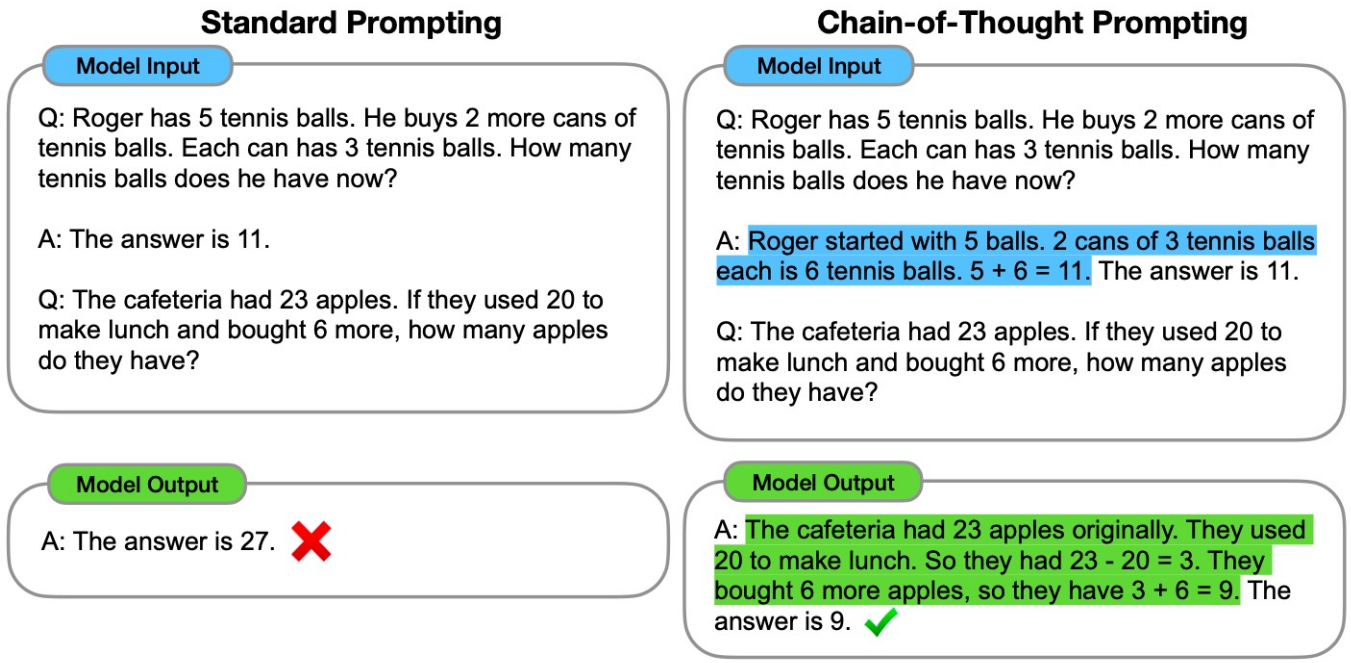
Chain-of-Thought

Definition

- Chain-of-Thought (CoT) refers to the explicit reasoning steps an LLM generates to reach an answer.
- It makes the model show its intermediate logic, rather than output only the final response.

How It Works

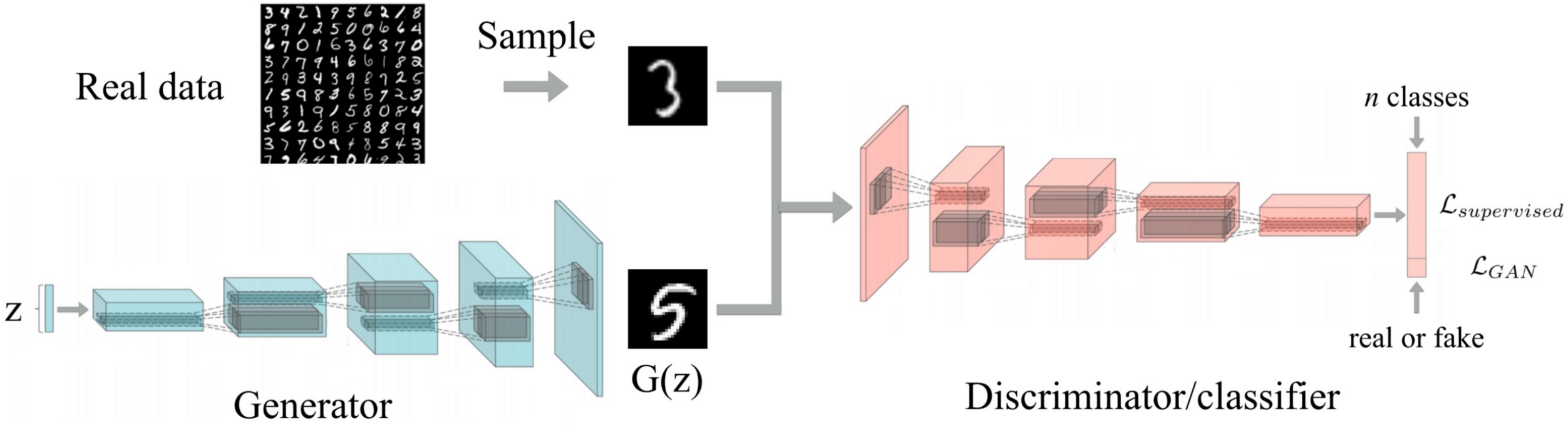
- Models are prompted to “think step by step,” “explain your reasoning,” or “show your working.”
- CoT can be trained via supervised fine-tuning (teacher-written steps).
- Long chains can amplify errors → leads to hallucination via “reasoning drift.”



Jason Wei et al., Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, 2022

Visual Generation

Generative Adversarial Networks: DC-GAN



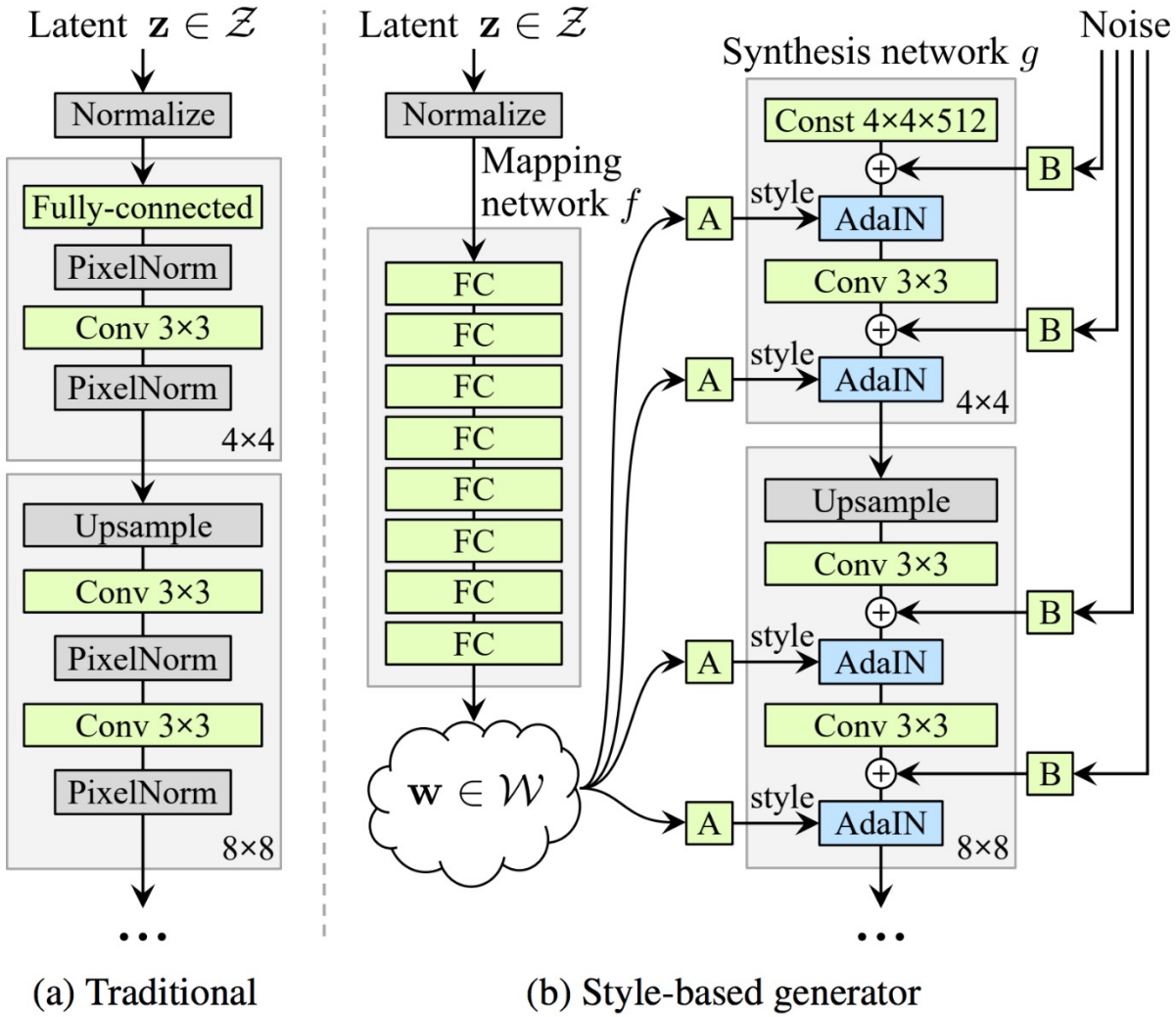
Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

Generative Adversarial Networks: DC-GAN



Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

Generative Adversarial Networks: StyleGAN



(a) Traditional (b) Style-based generator

Karras et al, "A Style-Based Generator Architecture for Generative Adversarial Networks", CVPR 2019

Generative Adversarial Networks: StyleGAN

512 x 384 cars



1024 x 1024 faces



Karras et al, “A Style-Based Generator Architecture for Generative Adversarial Networks”, CVPR 2019

Generative Adversarial Networks: GigaGAN



A living room with a fireplace at a wood cabin. Interior design.



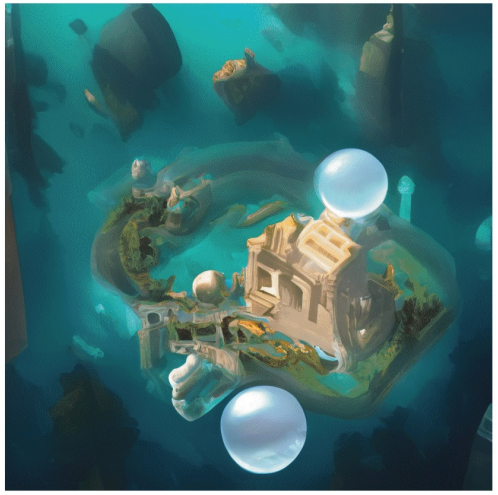
a blue Porsche 356 parked in front of a yellow brick wall.



Eiffel Tower, landscape photography



A painting of a majestic royal tall ship in Age of Discovery.



Isometric underwater Atlantis city with a Greek temple in a bubble.



A hot air balloon in shape of a heart. Grand Canyon



low poly bunny with cute eyes



A cube made of denim on a wooden table

Kang, M. et al., T.: Scaling up GANs for text-to-image synthesis. CVPR 2023



Input Video

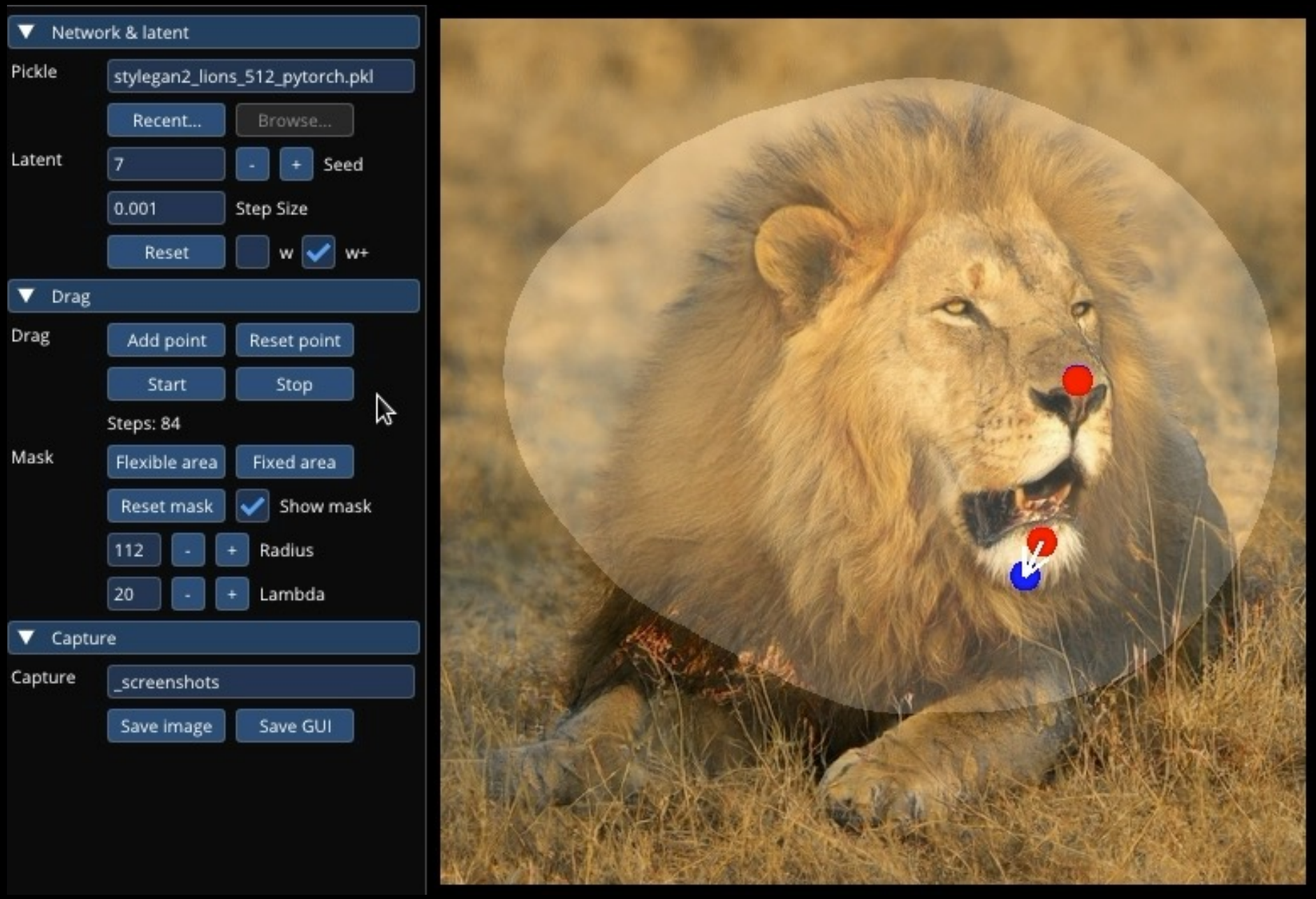


Reference Style



Our Toonification Result

Shuai Yang et al., VToonify: Controllable High-Resolution Portrait Video Style Transfer. SIGGRAPH Asia 2022



Xingang Pan et al, Drag Your GAN: Interactive Point-based Manipulation on the Generative Image Manifold, SIGGRAPH2023

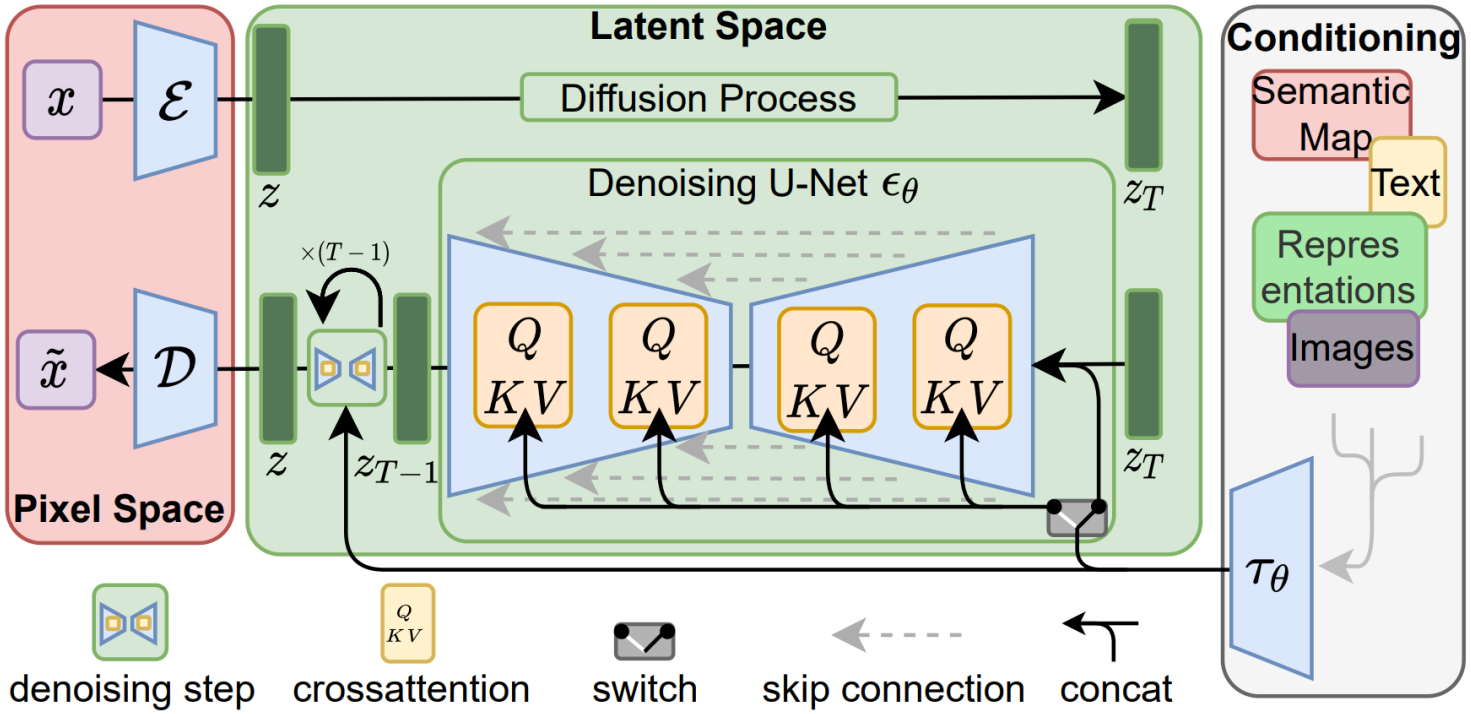


Xingang Pan et al, Drag Your GAN: Interactive Point-based Manipulation on the Generative Image Manifold, SIGGRAPH2023

Latent Diffusion Models

- Stable Diffusion is an example of latent diffusion model for text-to-image generation

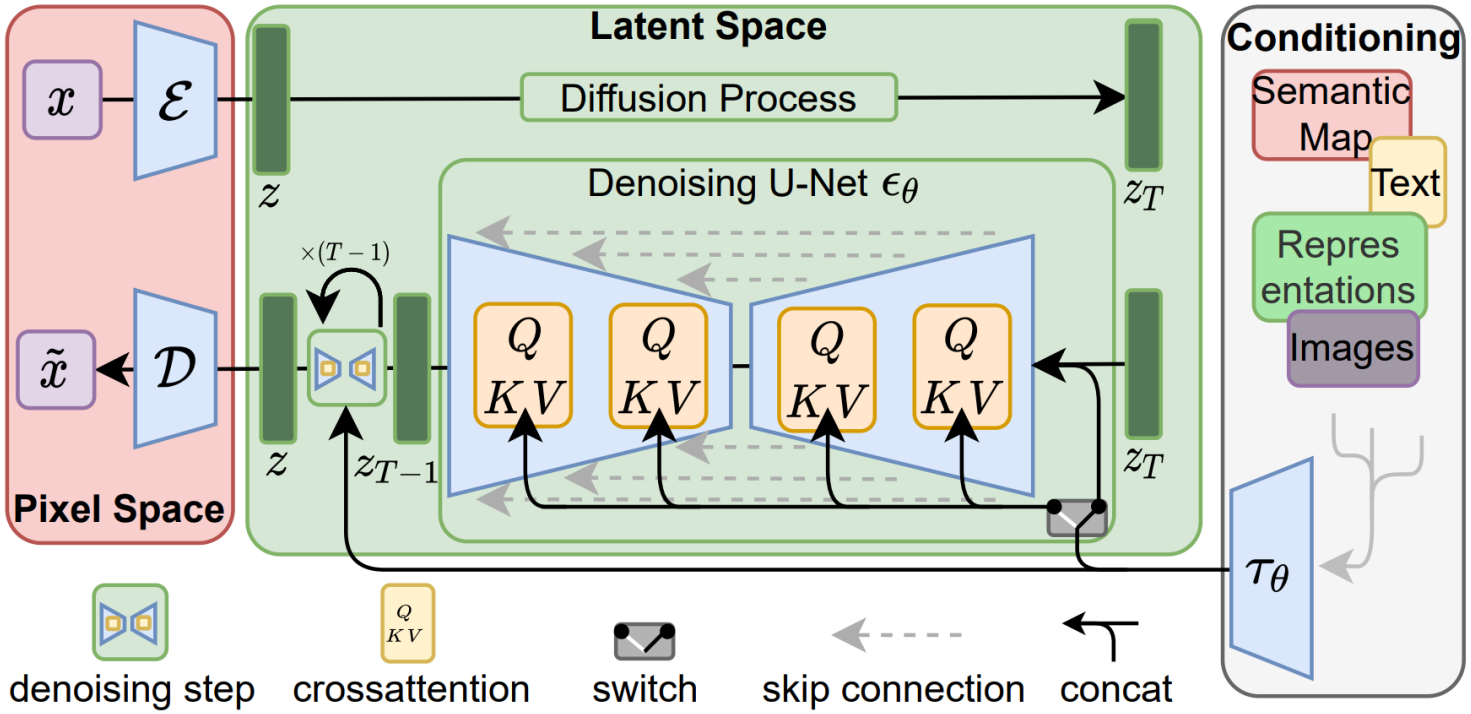
- Key idea
  - Projects the original data to a smaller latent space using a conventional autoencoder (e.g., VAE) and then runs the diffusion process in this smaller space



Rombach et al., High-Resolution Image Synthesis with Latent Diffusion Models, CVPR 2022.

Latent Diffusion Models

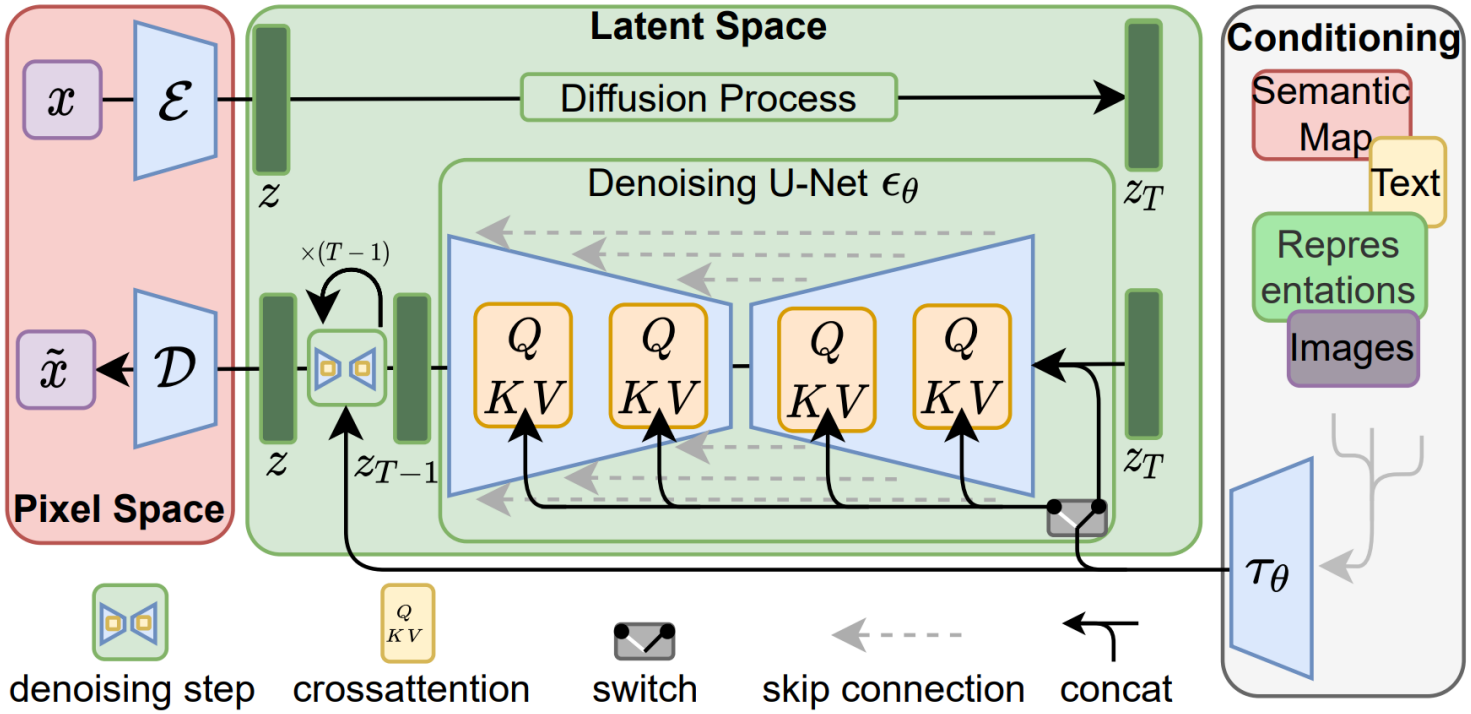
- Advantages
  - Reduce the dimensionality of the training data for the diffusion process
  - Allow other data types (text, graphs, etc.) to be described by diffusion model



Rombach et al., High-Resolution Image Synthesis with Latent Diffusion Models, CVPR 2022.

Latent Diffusion Models

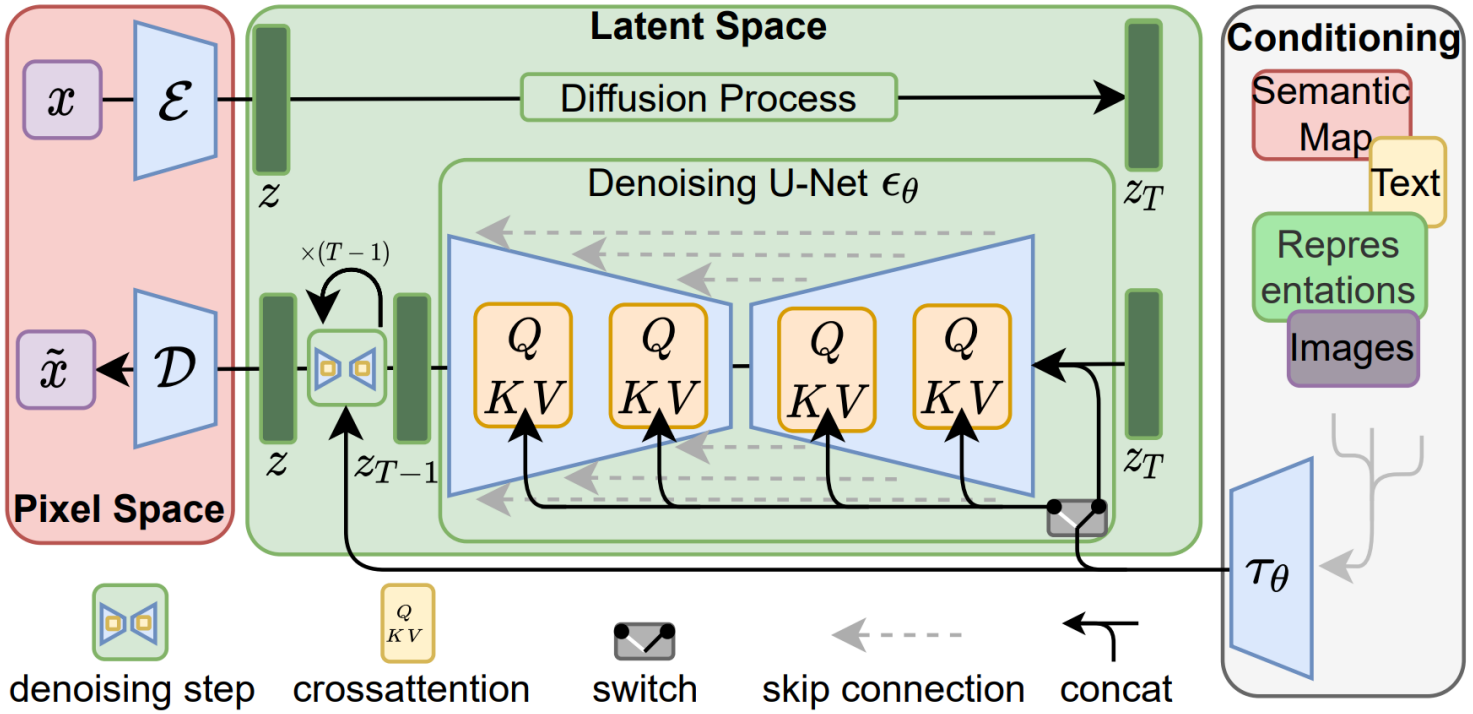
- Two-stage training
  - Train an autoencoder ($\mathcal{E}$ and $\mathcal{D}$) which provides a lower-dimensional (and thereby efficient) representational space
  - Train the DM
- Advantages
  - Train the universal autoencoding stage only once and can therefore reuse it for multiple DM trainings



Rombach et al., High-Resolution Image Synthesis with Latent Diffusion Models, CVPR 2022.

Latent Diffusion Models

- Conditional mechanism
  - UNet backbone with the cross-attention mechanism
  - Domain specific encoder τ_θ projects the conditions to an intermediate representation, which is then mapped to the intermediate layers of the UNet via a cross-attention layer implementing



$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) \cdot V, \text{ with}$$

$$Q = W_Q^{(i)} \cdot \varphi_i(z_t),\ K = W_K^{(i)} \cdot \tau_\theta(y),\ V = W_V^{(i)} \cdot \tau_\theta(y).$$

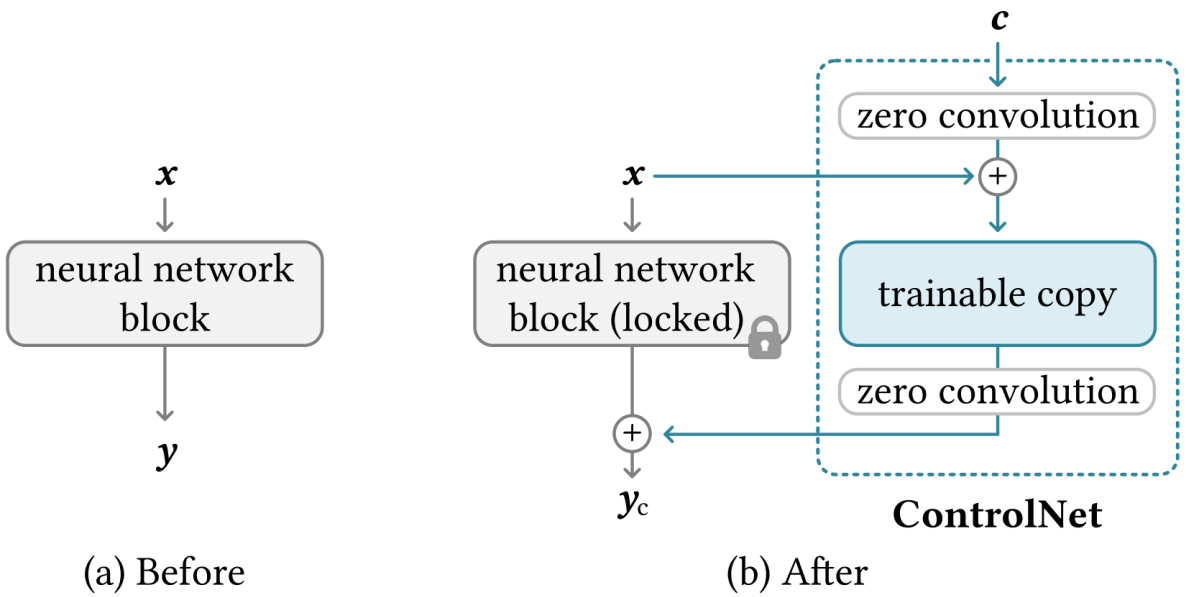
Rombach et al., High-Resolution Image Synthesis with Latent Diffusion Models, CVPR 2022.

ControlNet



Zhang et al., Adding Conditional Control to Text-to-Image Diffusion Models, ICCV 2023.

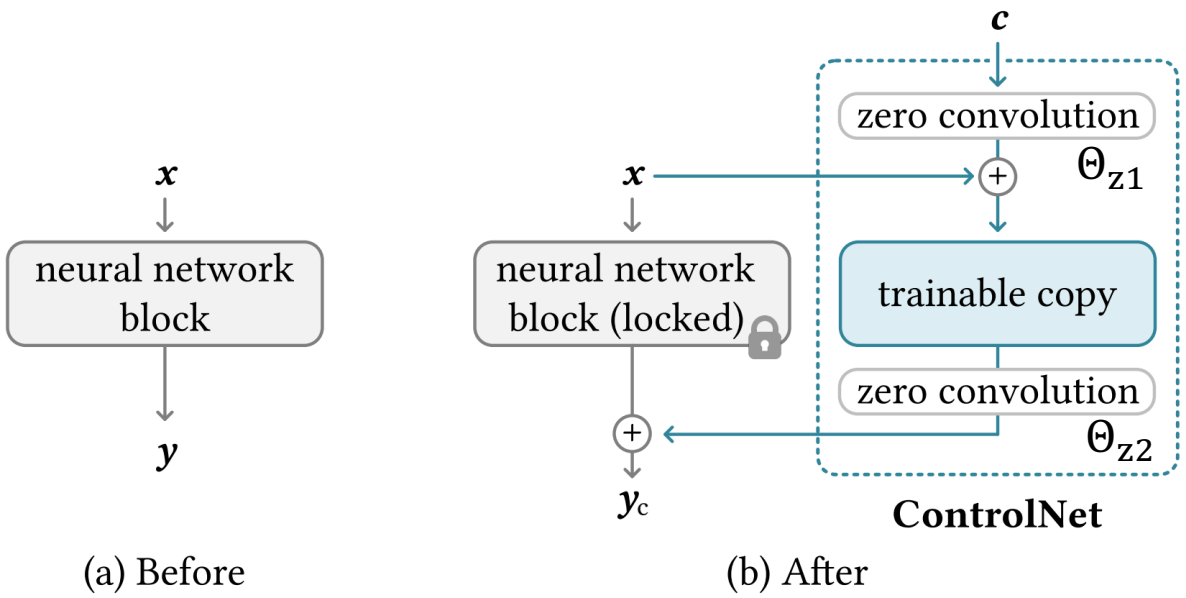
ControlNet



- It copys the weights of neural network blocks into a "locked" copy and a "trainable" copy.
- The "trainable" one learns your condition. The "locked" one preserves your model.
- Thanks to this, training with small dataset of image pairs will not destroy the production-ready diffusion models.

Zhang et al., Adding Conditional Control to Text-to-Image Diffusion Models, ICCV 2023.

ControlNet



- The "zero convolution" $\mathcal{Z}(\cdot;\cdot)$ is 1×1 convolution with both weight and bias initialized as zeros.
- To build up a ControlNet, we use two instances of zero convolutions with parameters Θ_{z1} and Θ_{z2}
- The complete ControlNet then computes

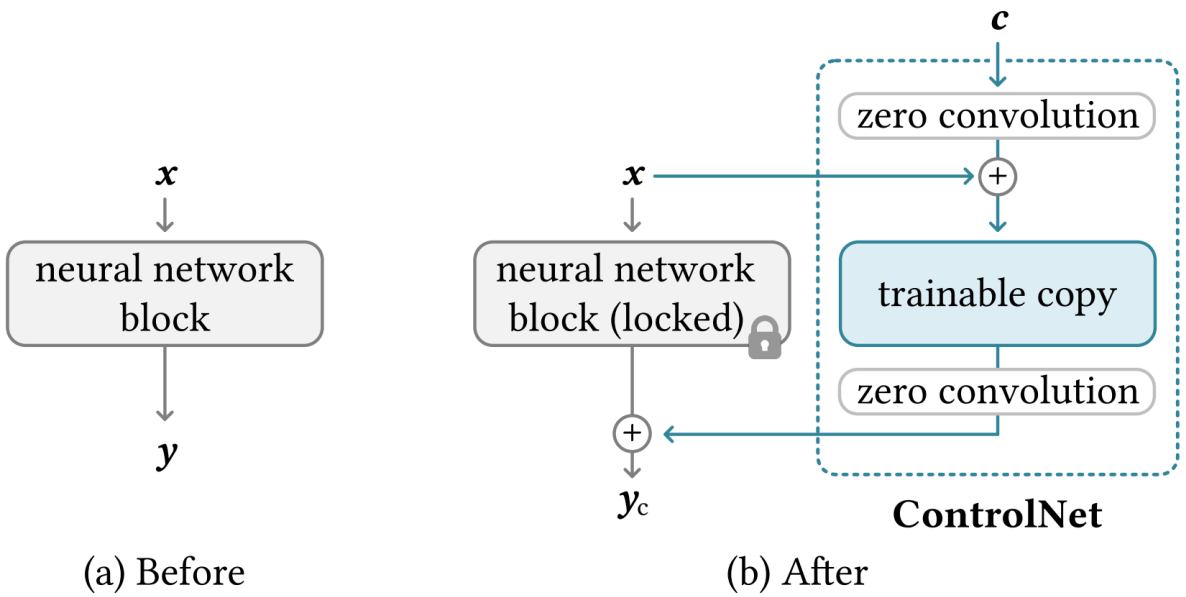
$$\boldsymbol{y}_c = \mathcal{F}(\boldsymbol{x};\Theta) + \mathcal{Z}(\mathcal{F}(\boldsymbol{x}+\mathcal{Z}(c;\Theta_{z1});\,\Theta_c);\,\Theta_{z2})$$

- Since both the weight and bias parameters of a zero convolution layer are initialized to zero, both of the $\mathcal{Z}(\cdot;\cdot)$ terms evaluate to zero, and

$$\boldsymbol{y}_c = \boldsymbol{y}$$

Zhang et al., Adding Conditional Control to Text-to-Image Diffusion Models, ICCV 2023.

ControlNet



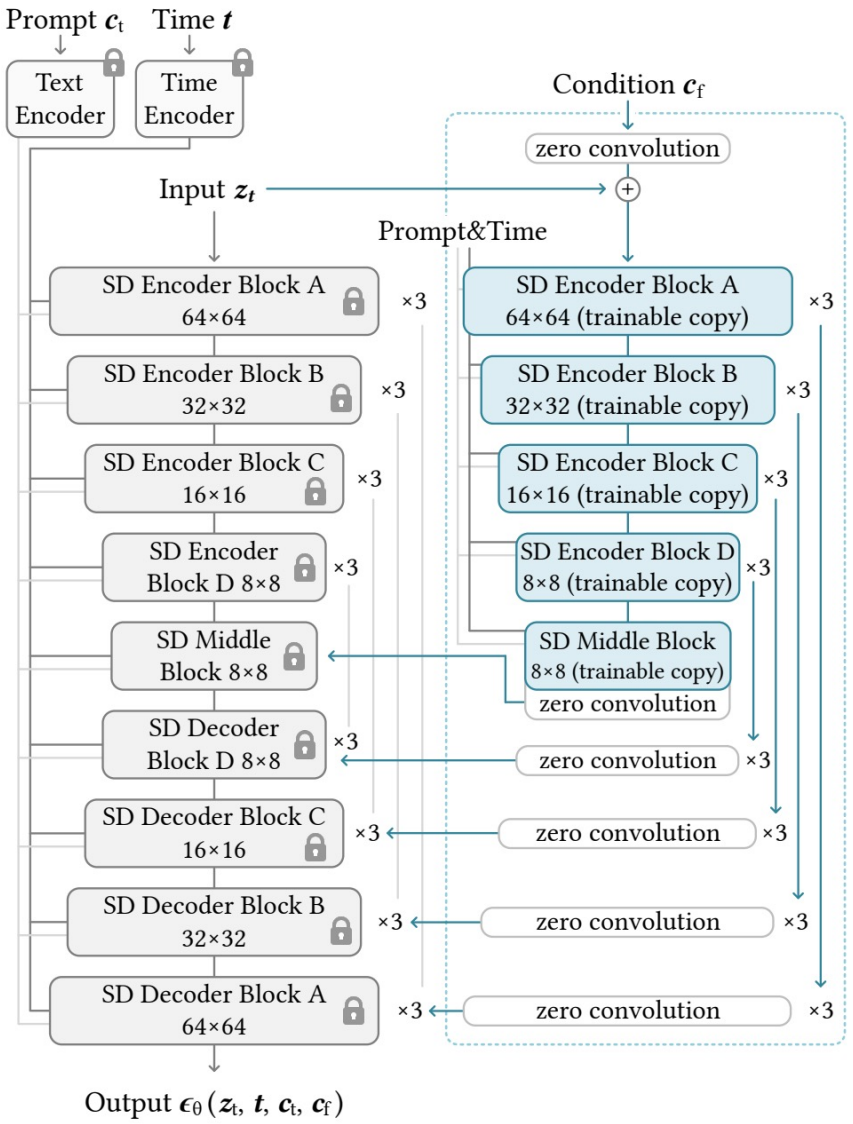
(a) Before (b) After

- Harmful noise cannot influence the hidden states of the neural network layers in the trainable copy when the training starts
- No layer is trained from scratch. You are still fine-tuning. Your original model is safe!
- This allows training on small-scale.

Zhang et al., Adding Conditional Control to Text-to-Image Diffusion Models, ICCV 2023.

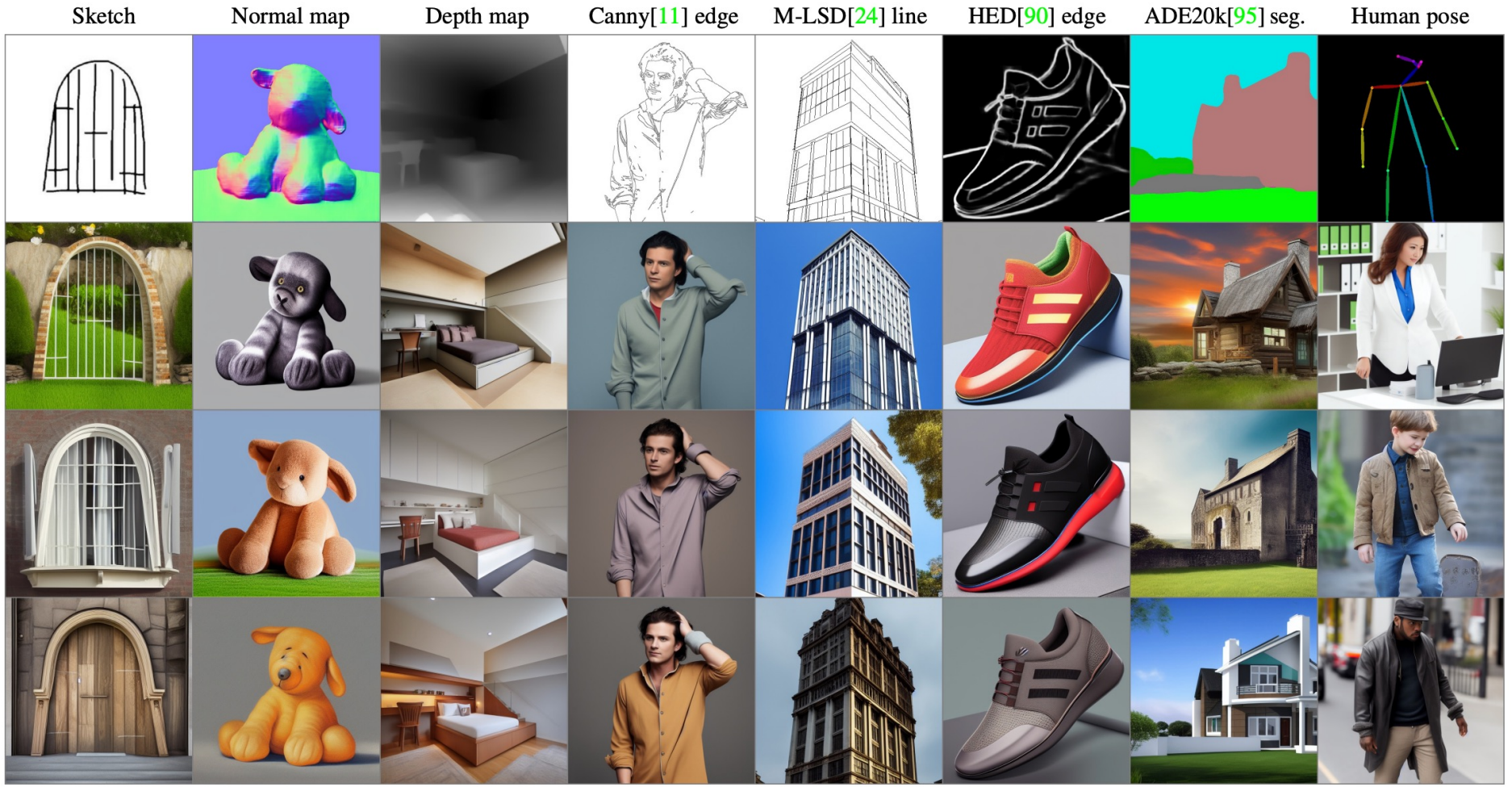
ControlNet + Latent Diffusion Model

- Stable Diffusion is essentially a U-Net
  - An encoder (12 blocks),
  - A middle block (1 block),
  - A skip-connected decoder (12 blocks)
- Details to combine ControlNet and Stable Diffusion
  - Each of the main blocks contains 4 resnet leayers and 2 ViTs (with several cross attention and self attention mechanisms)
  - Text prompts are encoded using the CLIP text encoder (Radford ICML 2021).
  - The ControlNet structure is applied to each encoder level of the U-Net
  - The outputs are added to the 12 skip-connections and 1 middle block of the U-Net
- No gradient computation is required in the originally locked encoder for the finetuning



Zhang et al., Adding Conditional Control to Text-to-Image Diffusion Models, ICCV 2023.
Radford et al. Learning Transferable Visual Models From Natural Language Supervision, ICML 2021

ControlNet + Latent Diffusion Model



Controlling Stable Diffusion with various conditions **without prompts**.

The top row is input conditions, while all other rows are outputs.

Empty string is used as input prompts.

All models are trained with general-domain data. The model has to **recognize semantic contents** in the input condition images to generate images.

Zhang et al., Adding Conditional Control to Text-to-Image Diffusion Models, ICCV 2023.

Synthesis



Low-quality image

X → Regression Model → Y

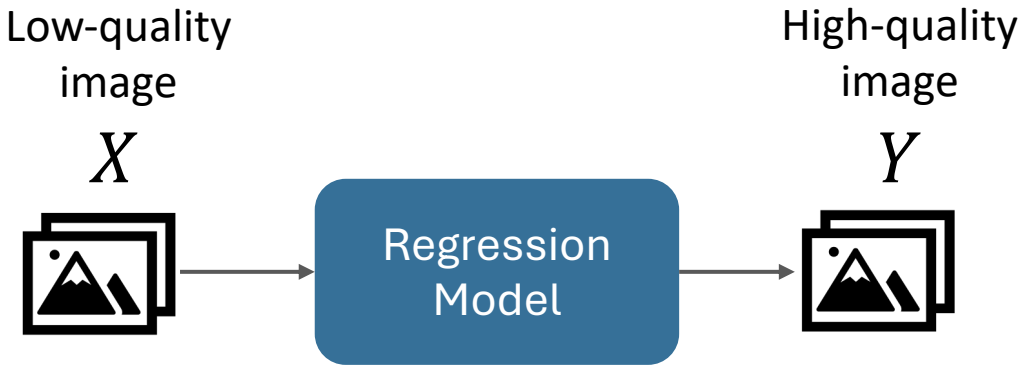


High-quality image

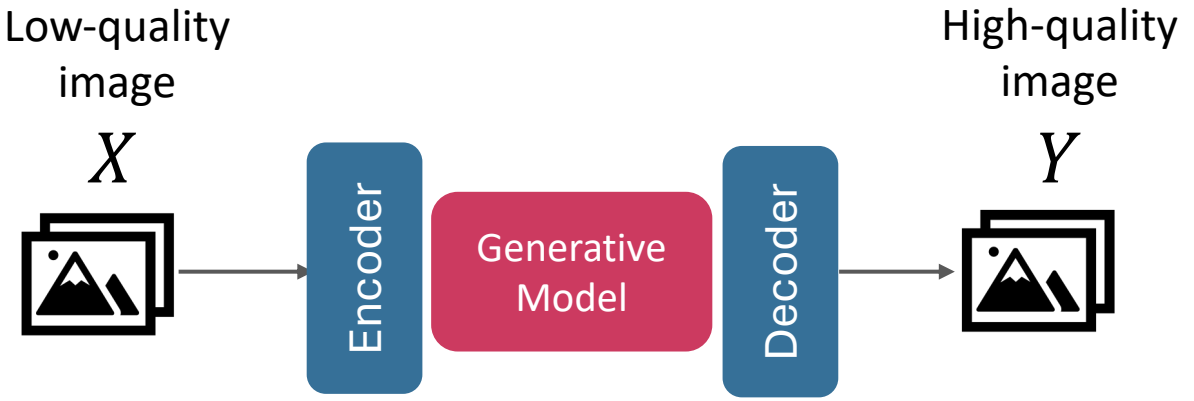
Image Super-Resolution

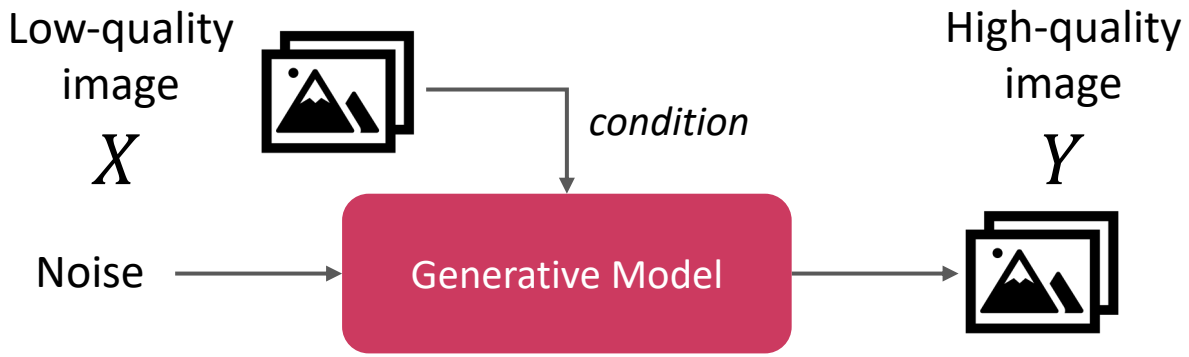
- A classic *ill-posed inverse* problem
- Maintain fidelity v.s. quality
- Blind restoration given *unknown* and *complex* degradations
- Prior is crucial

Synthesis

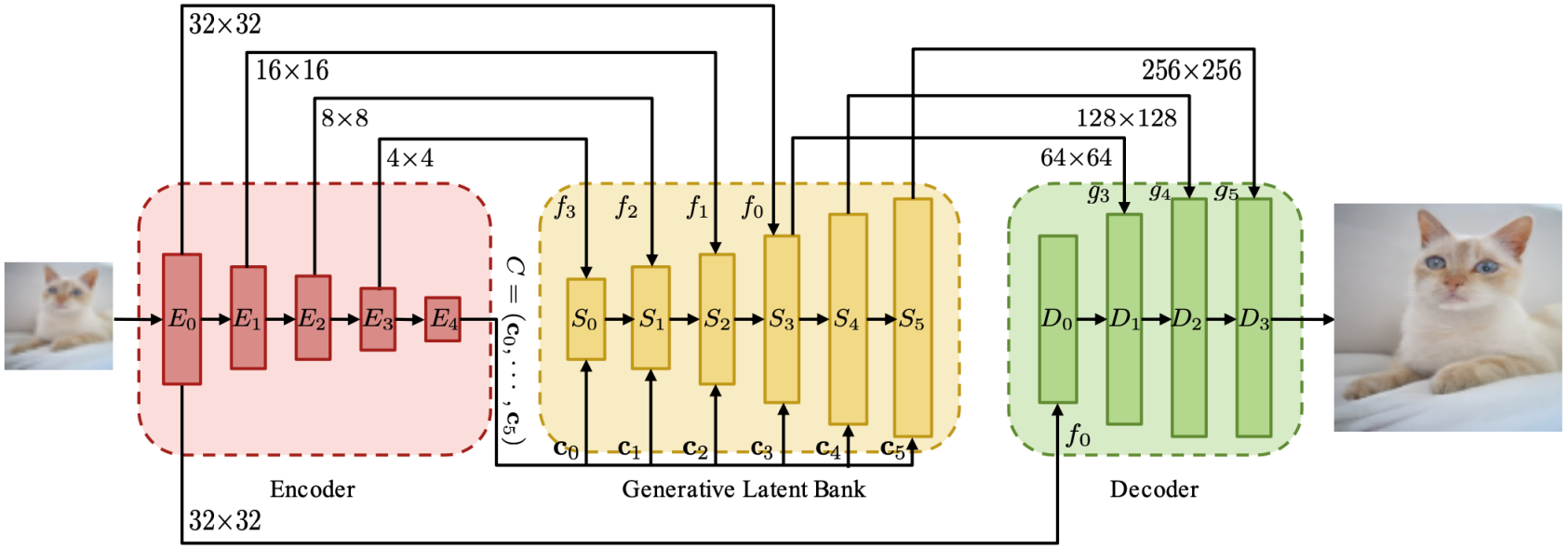


- Over-smoothing and loss of high-frequency details
- Limited perceptual quality
- Inadequate handling of real-world degradations





Synthesis

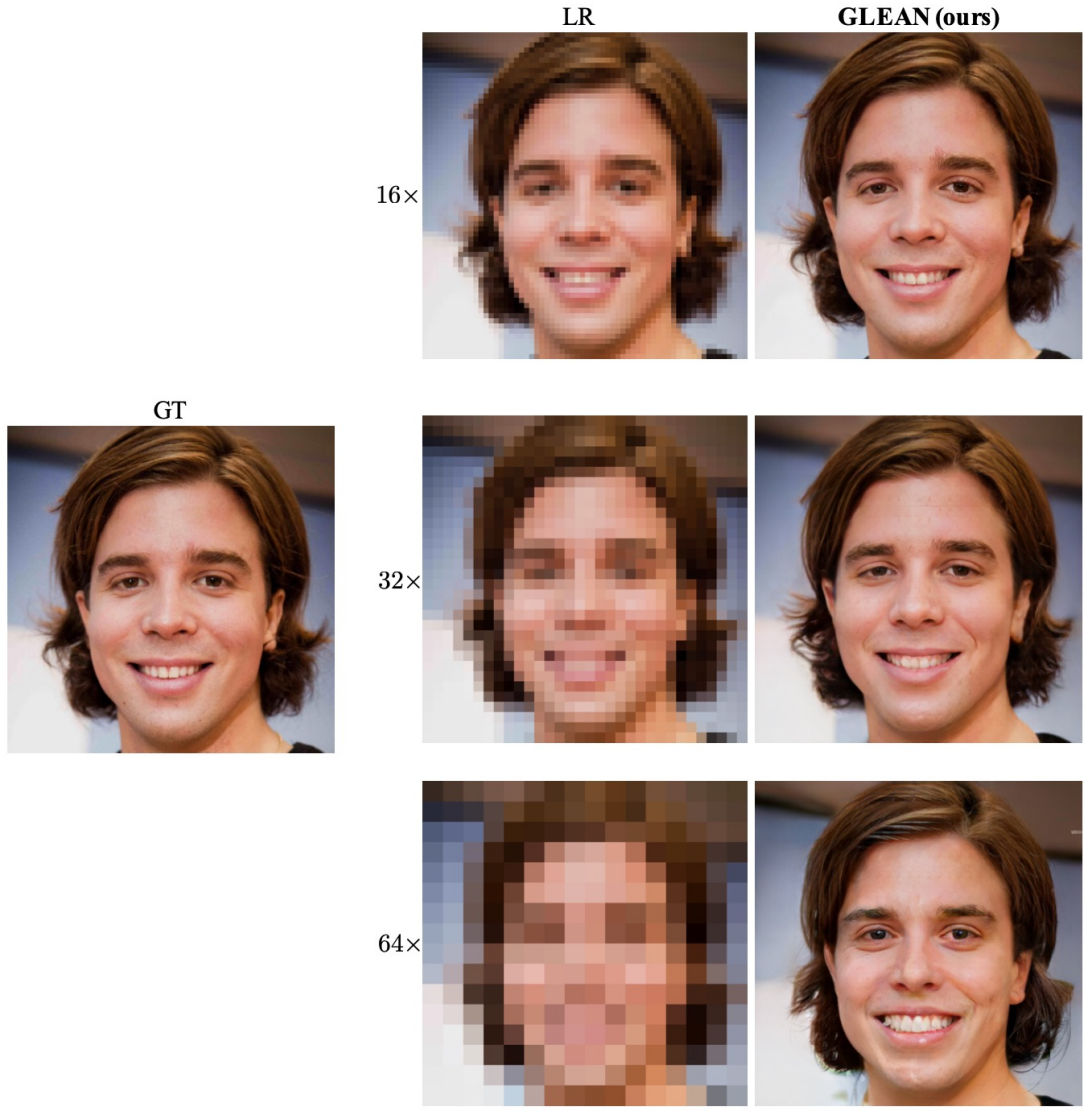


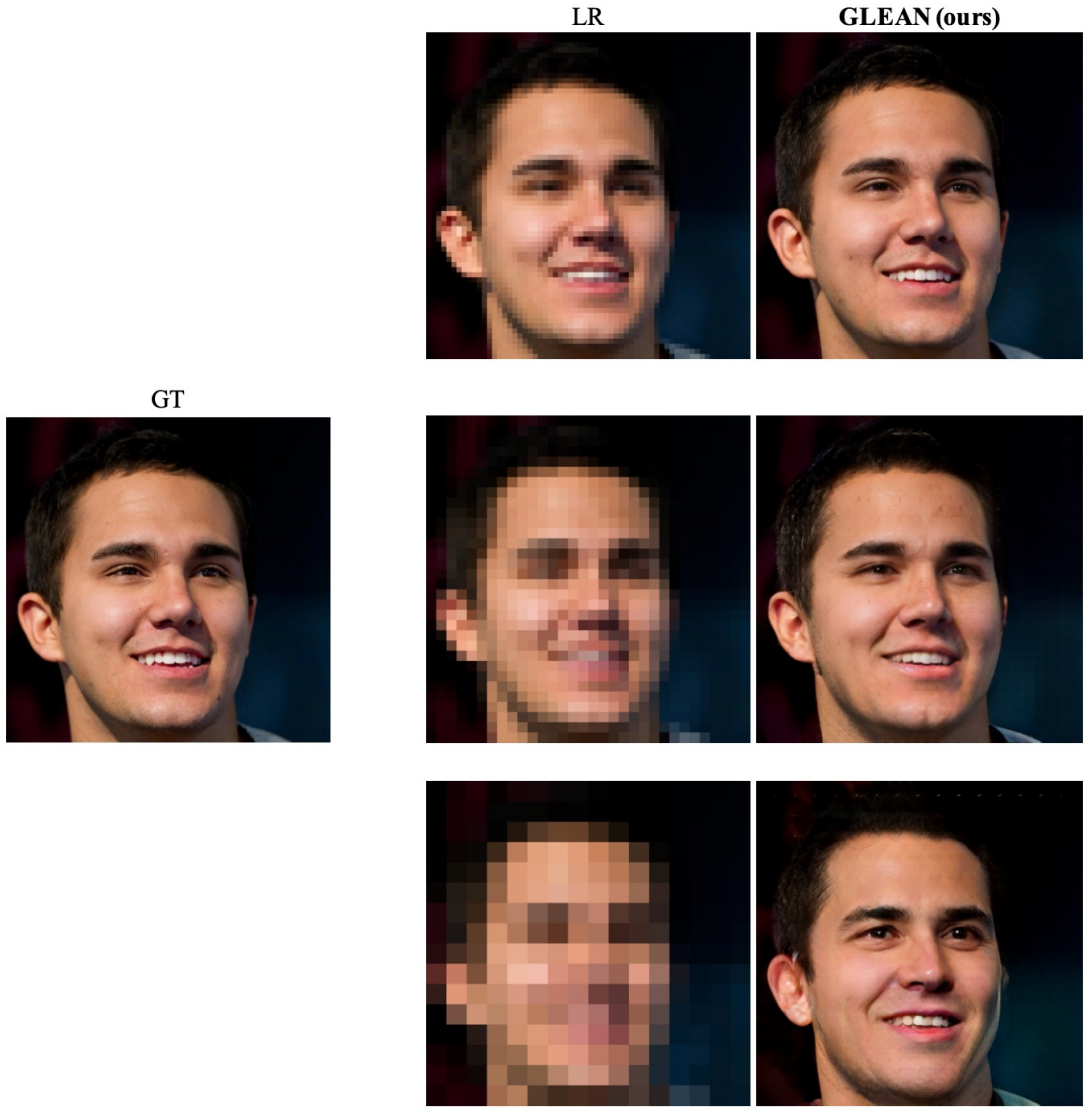
Generative prior from GAN

Condition the **bank** by passing both the latent vectors and multi-resolution convolutional features from the encoder to achieve high-fidelity results. Symmetrically, multi-resolution cues need to be passed from the bank to the decoder.

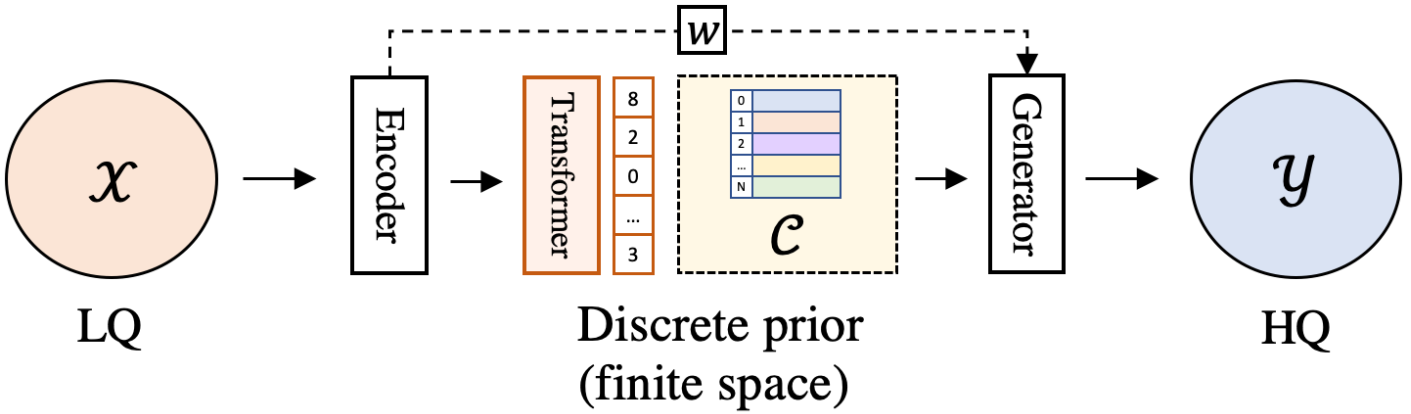
K. C. K. Chan, X. Wang, X. Xu, J. Gu, C. C. Loy, GLEAN: Generative Latent Bank for Image Super-Resolution and Beyond, TPAMI 2022

Synthesis





Synthesis



CodeFormer

Learn discrete codebook prior in a small proxy space to reduce the uncertainty and ambiguity of restoration mapping by, while providing rich visual atoms for generating high-quality faces.

Cast blind face restoration as a code prediction task

A Transformer-based prediction network to model the global composition and context of the low-quality faces for code prediction

Enable the discovery of natural faces that closely approximate the target faces even when the inputs are severely degraded

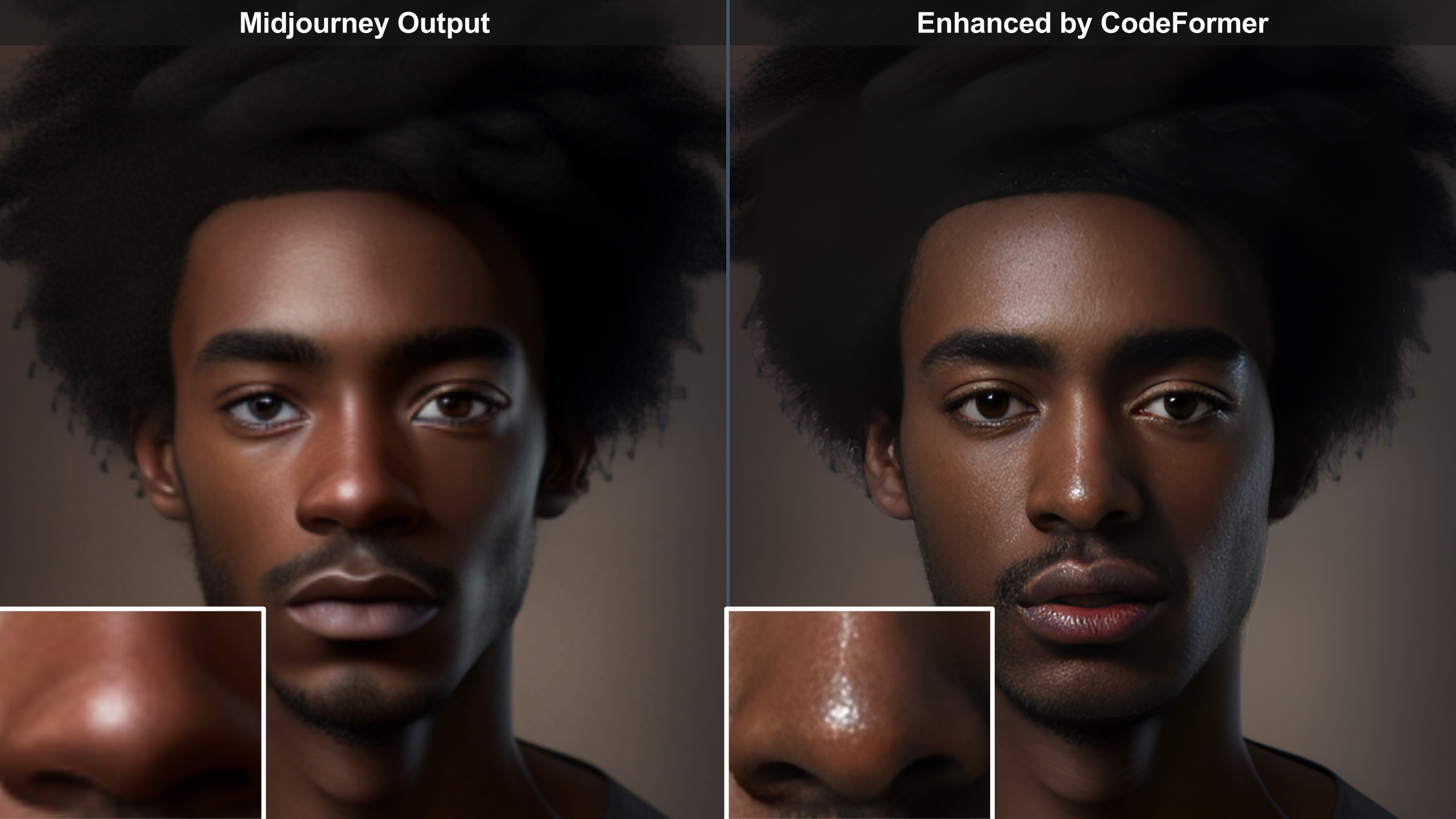
Shangchen Zhou et al. Towards Robust Blind Face Restoration with Codebook Lookup Transformer. NeurIPS 2022


Stable Diffusion 2.1 Output
Enhanced by CodeFormer

Stable Diffusion 2.1 Output

Enhanced by CodeFormer



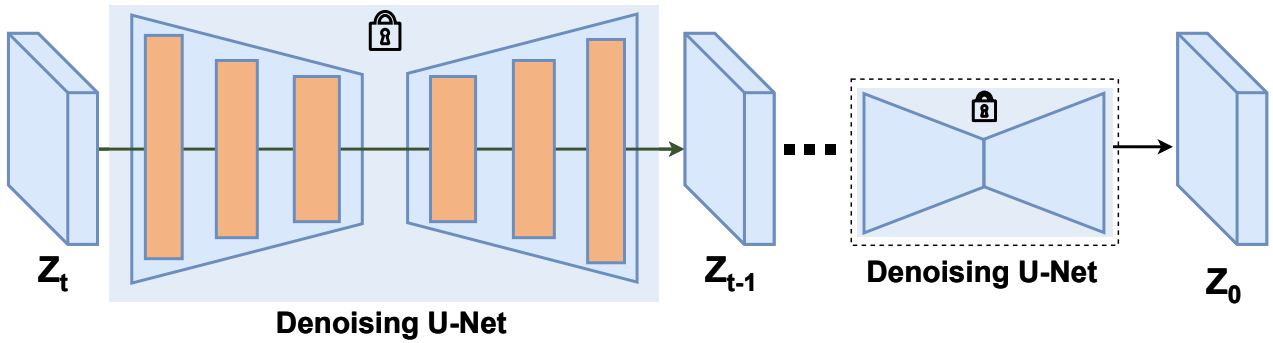

Midjourney Output
Enhanced by CodeFormer

Synthesis

Keeping the prior and fidelity

- Frozen stable diffusion model as a backbone

- Minimal alterations to prevent disrupting the prior



Jianyi Wang et al. Exploiting Diffusion Prior for Real-World Image Super-Resolution. IJCV 2024 (arXiv in May 2023)

Synthesis

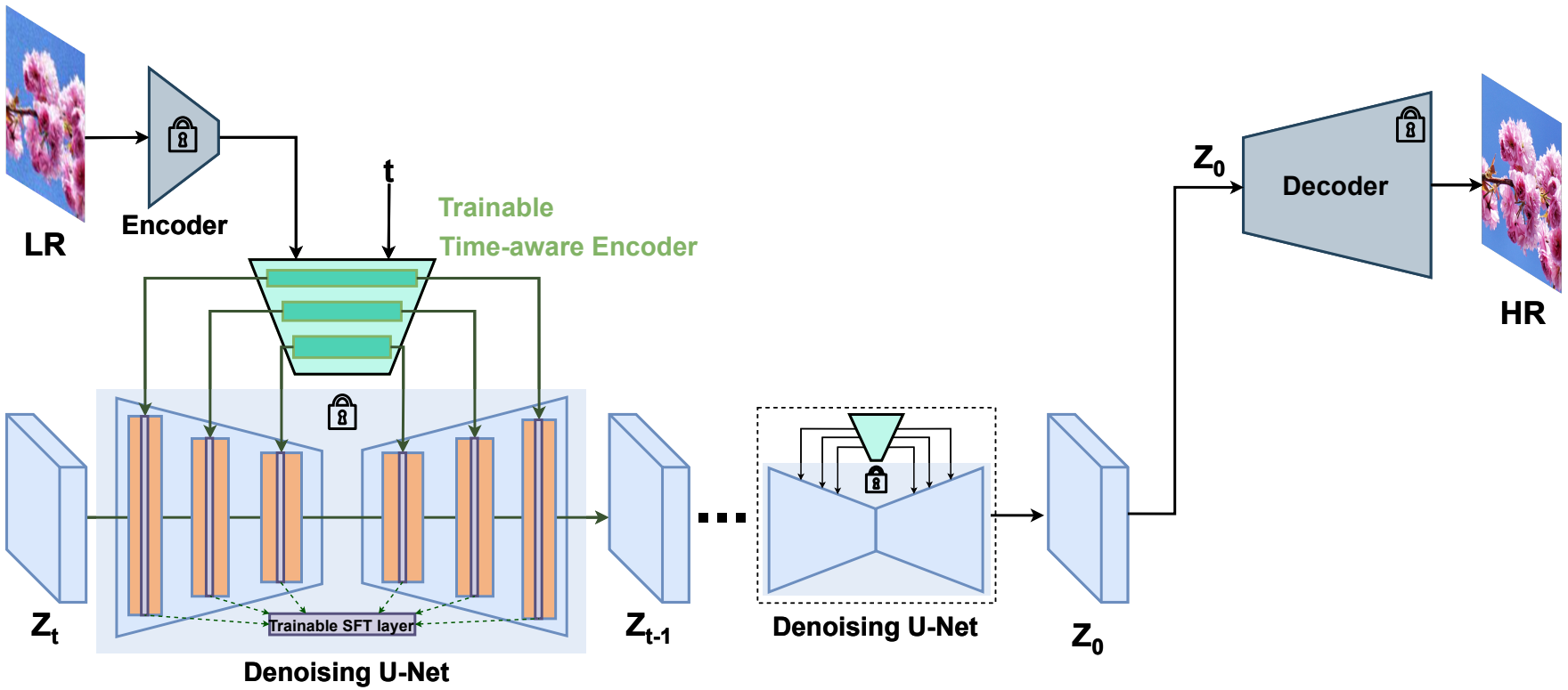
Keeping the prior and fidelity

- Train only the time-aware encoder and spatial feature transformation layer

$$\boldsymbol{\alpha}^n, \boldsymbol{\beta}^n = \mathcal{M}_\theta^n(\mathbf{F}^n)$$

$$\hat{\mathbf{F}}_{\text{dif}}^n = (1 + \boldsymbol{\alpha}^n) \odot \mathbf{F}_{\text{dif}}^n + \boldsymbol{\beta}^n$$

- Adaptively adjust the condition strength derived from the LR feature through *t*

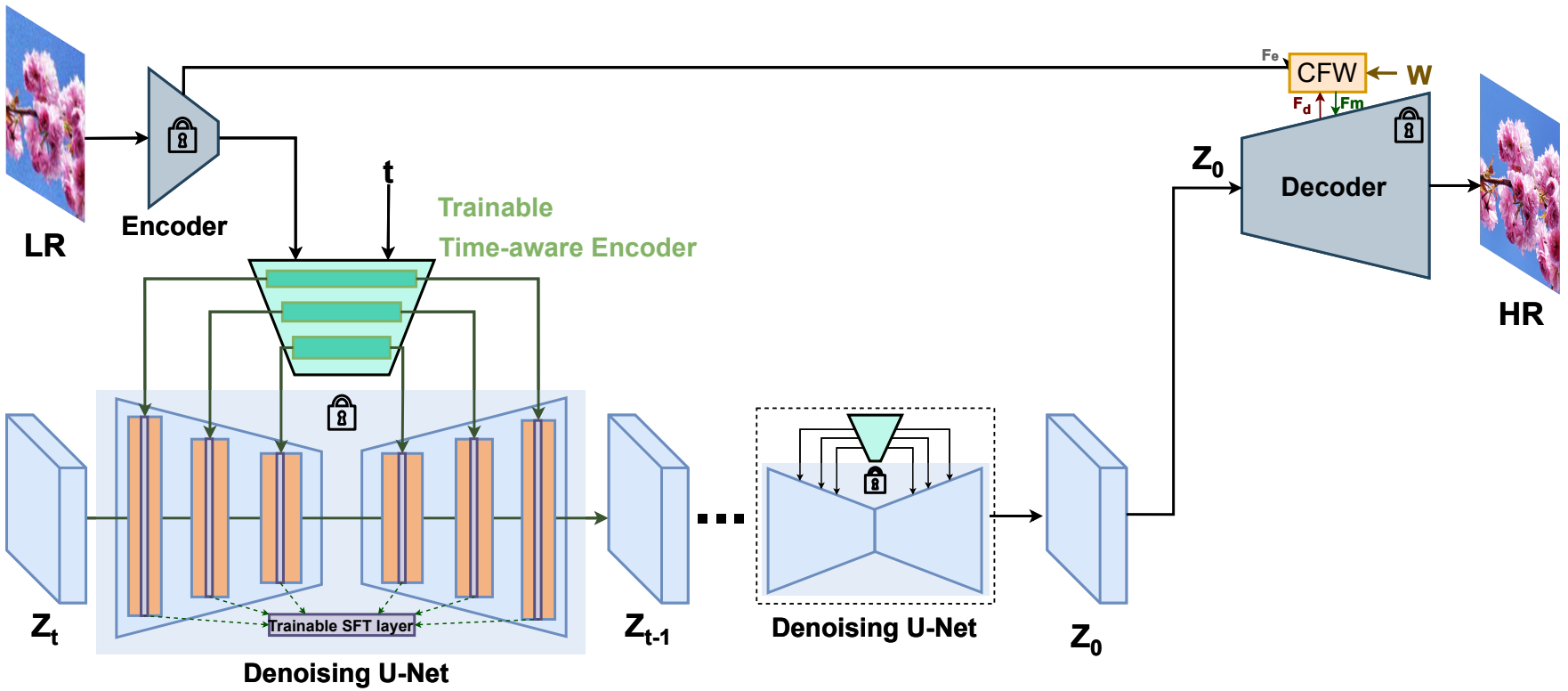


Xintao Wang et al. Recovering Realistic Texture in Image Super-resolution by Deep Spatial Feature Transform. CVPR 2018

Synthesis

Keeping the prior and fidelity

- Add a controllable skip connection to benefit from structural guidance from the LR image, enhancing fidelity
- Control the modulation strength through **w**
- A larger **w** allows stronger structural guidance



Shangchen Zhou et al. Towards Robust Blind Face Restoration with Codebook Lookup Transformer. NeurIPS 2022


Zoomed LR (4x)
StableSR
Real-world SR


Zoomed LR (4x)
StableSR
Real-world SR


Zoomed LR (4x)
StableSR
Old Photo SR


Zoomed LR (4x)
StableSR
Old Photo SR


Zoomed LR (4x)
StableSR
AIGC SR (1024x1536 to 4096x6144)





Remove Flickers For Old Movie



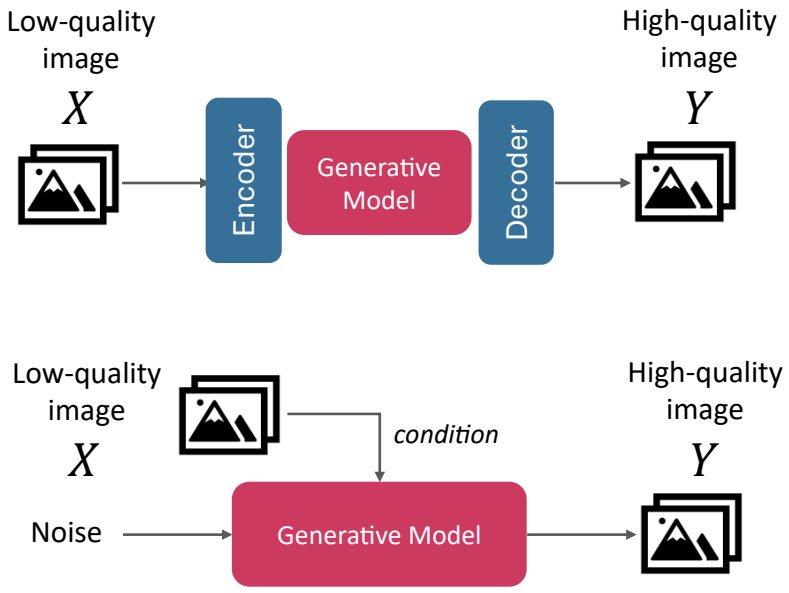
Old Movie Clip



Upscale-A-Video (Ours)


Upscale-A-Video
Real-world Video

Limitations



- Generative models may invent plausible details that don’t exist in the original image
- Generative models can produce artifacts, inconsistent textures, or noise under certain conditions (e.g., unseen patterns, extreme blur)

How These Models Work in Practice

Prompt Engineering and Fine-Tuning

From Pretraining to Adaptation

- Foundation models are pretrained on vast, diverse datasets.
- To specialize for downstream tasks, we adapt them via:
  - **Prompting**: Steering the model using textual or structured cues. No weight updates needed.
  - **Fine-tuning**: Updating model weights on domain-specific data.

Prompt Engineering and Fine-Tuning

Prompt Engineering

- Prompt design: guide model reasoning via examples/instructions.
- Biomedical examples:
  - *“Summarize this radiology report for a general practitioner.”*
  - *“Explain CT findings as if to a patient.”*
- Emergence of **prompt libraries**, **auto-prompt tuning**, and **chain-of-thought prompting** for reasoning enhancement.
- **Chain-of-thought prompting** encourages models to produce intermediate reasoning steps before the final answer
  - Instead of:
    “Is the lesion benign or malignant?” → “Benign.”
  - Use:
    “Let’s think step by step.

    Does the lesion has smooth borders and low density, ….

    Therefore, the lesion is benign.”

Prompt Engineering and Fine-Tuning

Fine-tuning Techniques

- **Full fine-tuning**: Entire model updated → high accuracy, but computationally expensive.

- **Parameter-efficient fine-tuning (PEFT)**:
  - Benefits: reduces compute, preserves generality.

Prompt modifications	Adapter methods	Reparameterization
Prompt Tuning Prefix-Tuning	Adapters	Low-Rank Adaptation (LoRA)

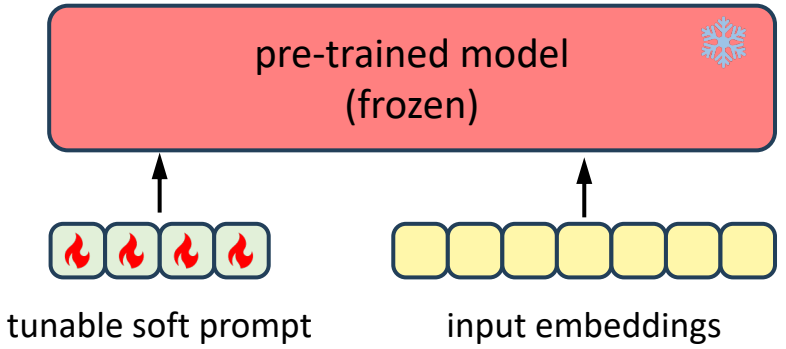
Prompt Engineering and Fine-Tuning

Prompt-Tuning

Input-level control

Concatenates the embeddings of the input tokens with a *trainable tensor* that can be optimized via backpropagation to improve the modeling performance on a target task

Suitable for simple tasks



Lester et al., The Power of Scale for Parameter-Efficient Prompt Tuning, 2021
Jia et al., Visual Prompt Tuning, 2022
Zhou et al., Learning to Prompt for Vision-Language Models, 2022

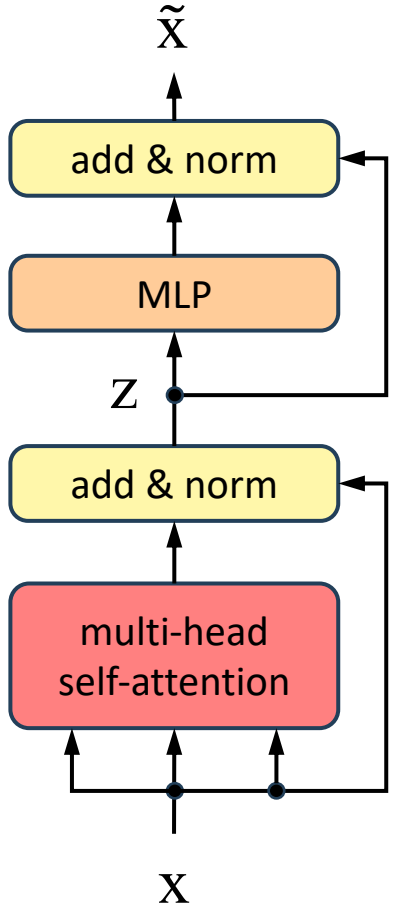
Prompt Engineering and Fine-Tuning

Prefix-Tuning

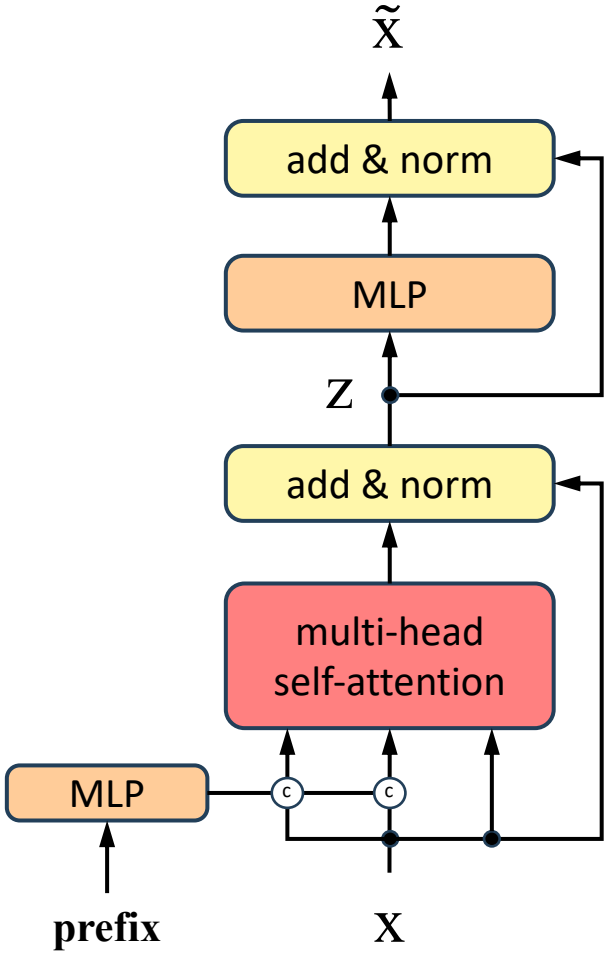
Layer-wise context injection

Add a *trainable tensor* to *each transformer block* instead of only the input embeddings, as in prompt tuning.

Conditioning the model's activations without modifying the pretrained parameters.



Regular Transformer block



Transformer block with prefix

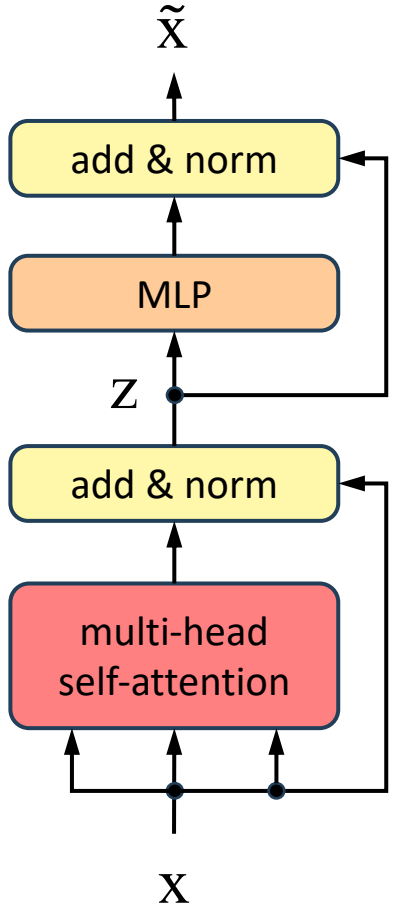
Li and Liang, Prefix-Tuning: Optimizing Continuous Prompts for Generation, 2021

Prompt Engineering and Fine-Tuning

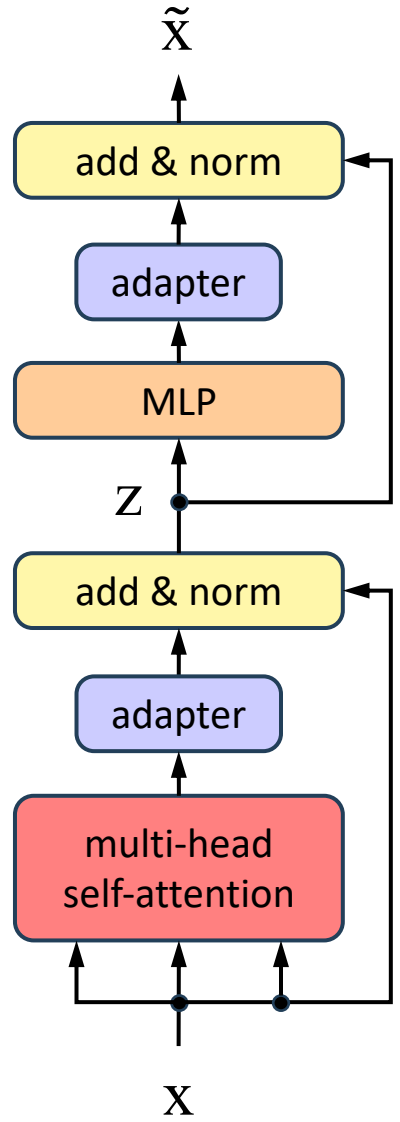
Adapters

Intermediate-layer modules

Add additional parameters to each transformer block instead of prepending prefixes to the input embeddings



Regular Transformer block



Transformer block with adapters

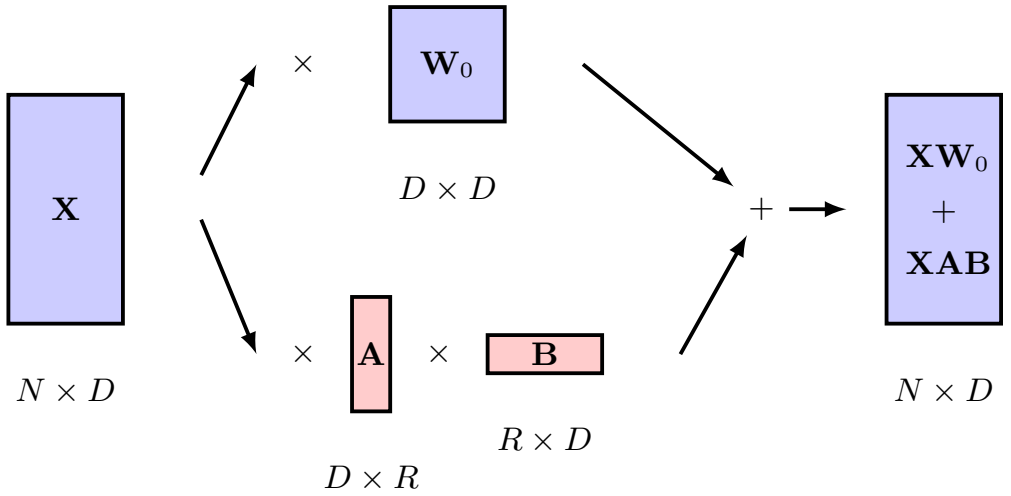
Houlsby et al., Parameter-Efficient Transfer Learning for NLP, 2019

Prompt Engineering and Fine-Tuning

Low-Rank Adaptation (LoRA)

LoRA attempts to fine-tune the "residual" of the model instead of the entire model: i.e., train the ΔW instead of W.

$$\widehat{\mathbf{W}} = \mathbf{W}_0 + \Delta\mathbf{W}$$



where we can further decompose $\Delta\mathbf{W}$ into low-rank matrices : $\Delta\mathbf{W} = \mathbf{AB}$

where $\mathbf{A} \in \mathbb{R}^{D\times R}$, $\mathbf{A} \in \mathbb{R}^{R\times D}$, and the rank $R \ll D$

During training, $\mathbf{W}$ is frozen and does not receive gradient updates, while $\mathbf{A}$ and $\mathbf{B}$ contain trainable parameters.

Excellent trade-off, high performance

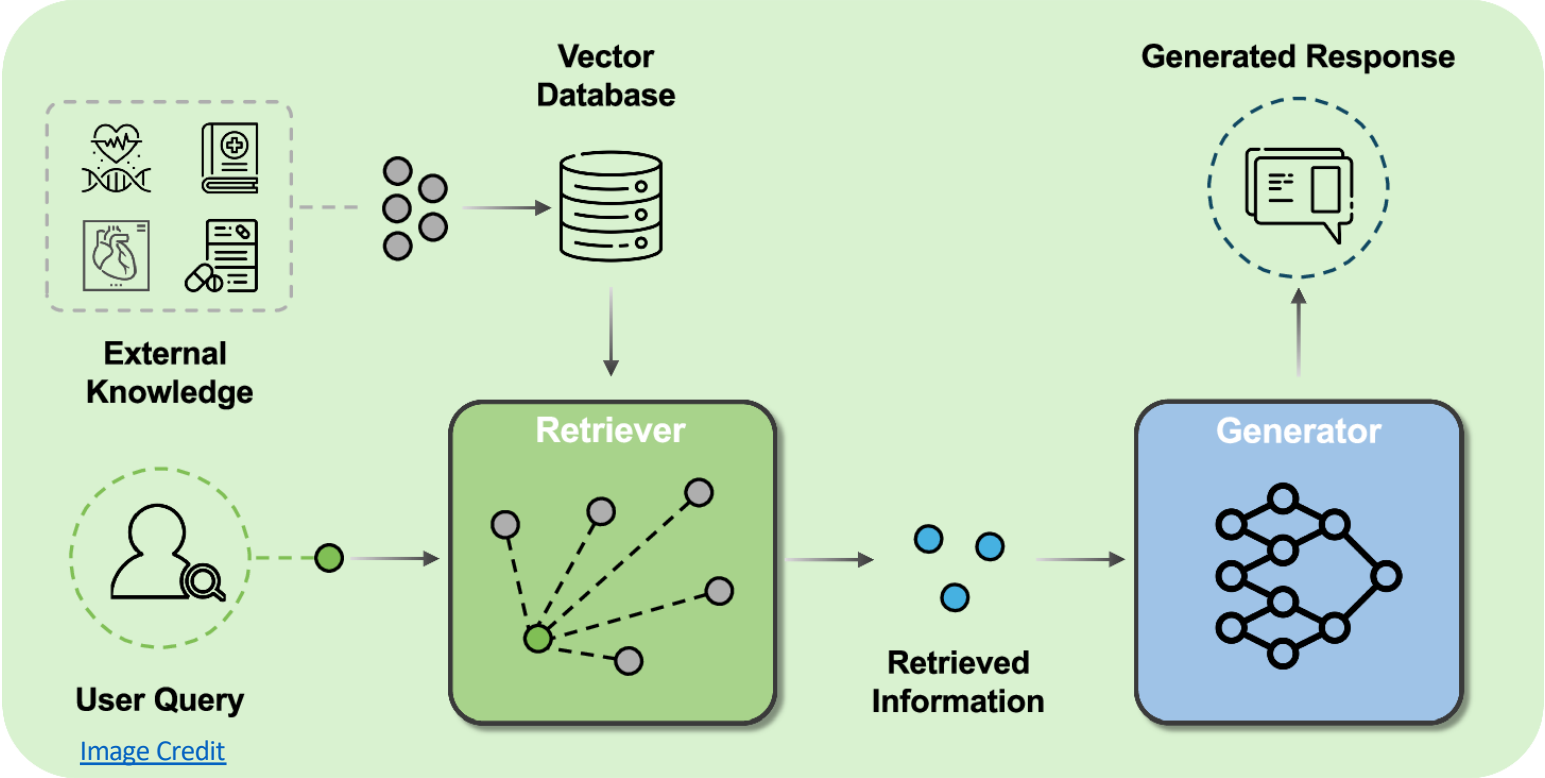
Example: fine-tuning Med-SAM with LoRA for organ segmentation using 100 annotated scans.

Hu et al., LoRA: Low-Rank Adaptation of Large Language Models, 2021

RAG and Memory Augmentation

Motivation

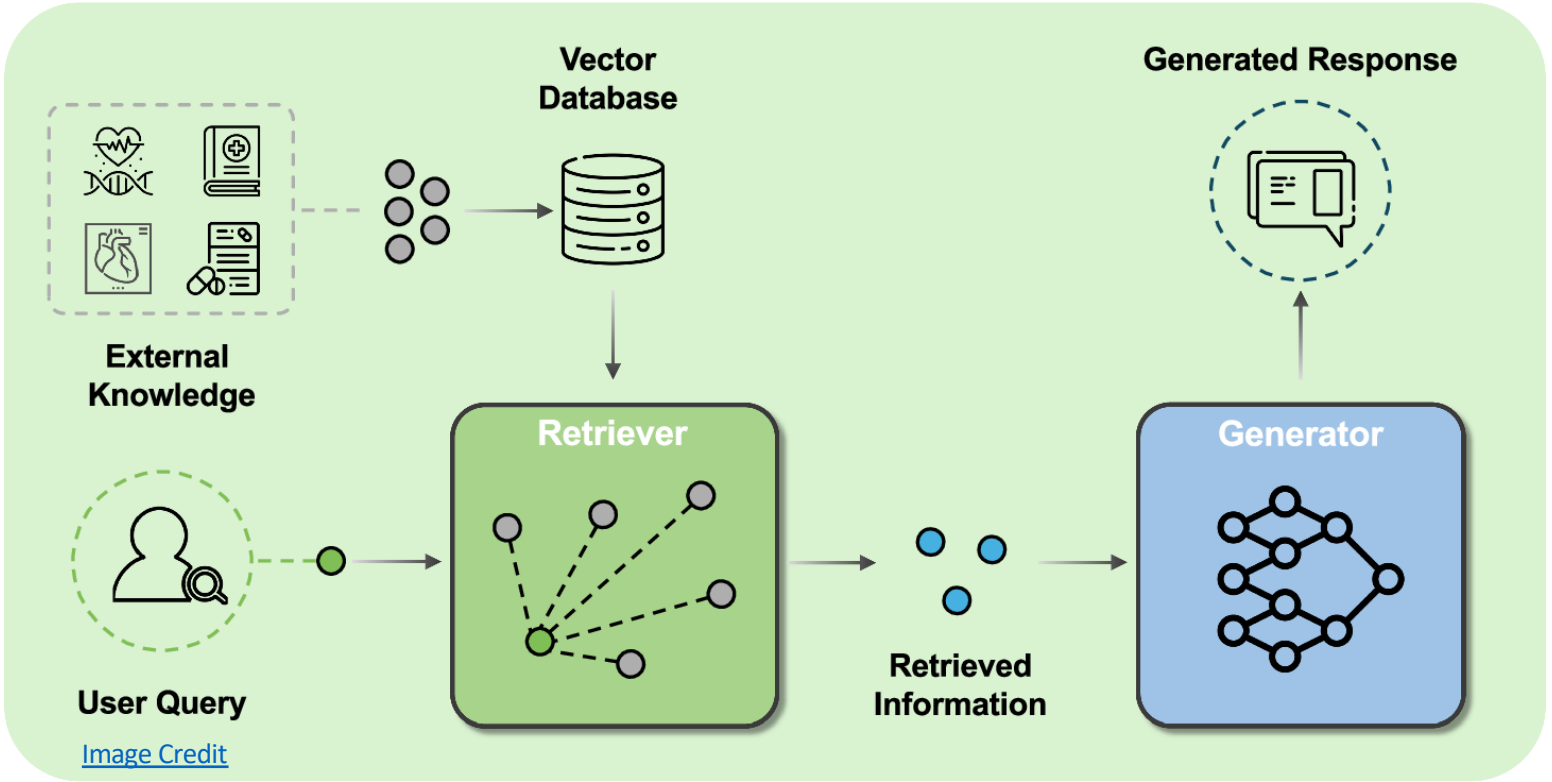
- Foundation models often hallucinate or contain outdated facts.
- Retrieval-Augmented Generation (RAG) integrates **external databases** or **vector memory** to fetch real-time knowledge.
- Especially crucial in biomedicine, where accuracy and recency are vital.



RAG and Memory Augmentation

Architecture Overview

- **Retrieval**: A query embeds into a vector space → nearest-neighbor search (e.g., FAISS, Milvus).
- **Augmentation**: Retrieved passages/images injected into the model's context before generation.
- **Generation**: Model synthesizes grounded, evidence-based output.
- Example: Clinical assistant retrieves patient records, lab results, and literature evidence before answering



Human Feedback and Alignment (RLHF)

The Need for Alignment

- Models optimize next-token prediction, not human intent.
- Misalignment examples: biased responses, unsafe recommendations, overconfidence
- **Goal:** Align model behavior with human values and domain standards.

Human Feedback and Alignment (RLHF)

Reinforcement Learning from Human Feedback (RLHF)

Start from a supervised fine-tuned (SFT) model

Step 1: Collect human preference data (s1 vs. s2 responses).
Step 2: Train a reward model (RM) to score outputs.
Step 3: Fine-tune base model with reinforcement learning algorithms (e.g., Proximal Policy Optimization)

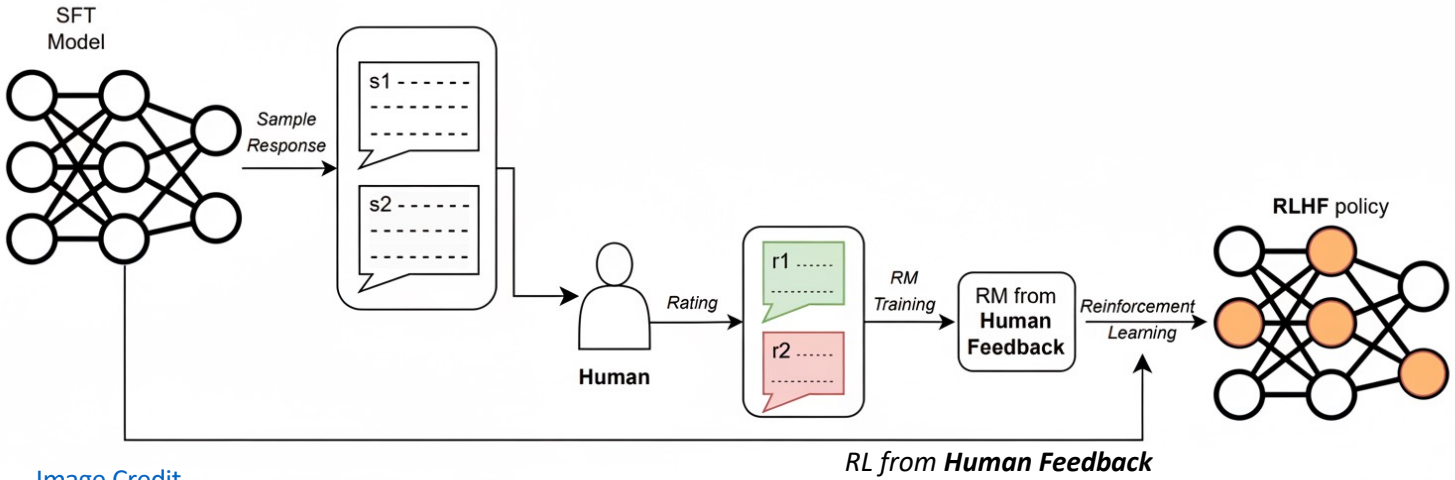


Image Credit

Human Feedback and Alignment (RLHF)

Domain-specific Feedback Loops

- Beyond general human feedback: incorporate **expert-in-the-loop** refinement.
  - E.g., radiologists label model explanations as “accurate” or “misleading.”
  - Leads to safer, interpretable clinical AI.
- Emerging methods: **RLAIF** (AI-generated feedback) and **constitutional AI** (a set of rules, e.g., do not give dangerous advice) to reduce dependence on costly human labeling.

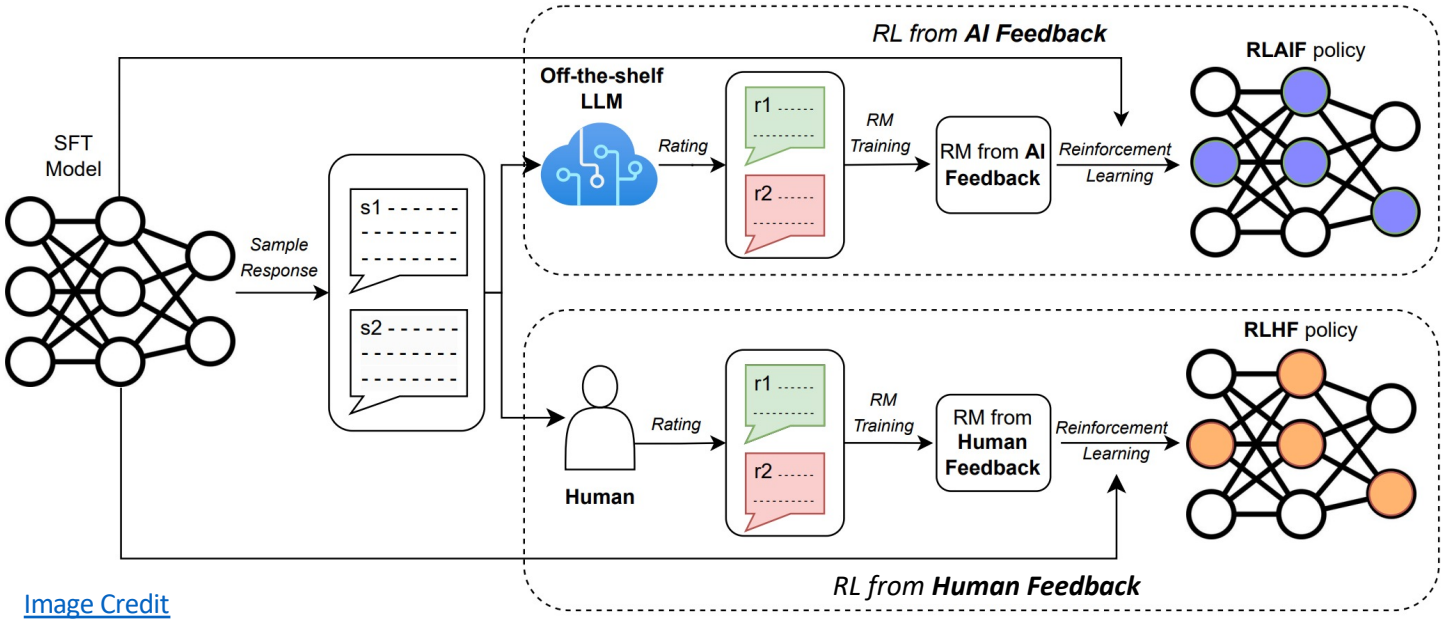


Image Credit

Lee et al., RLAIF vs. RLHF: Scaling Reinforcement Learning from Human Feedback with AI Feedback, 2023

Bai et al., Constitutional AI: Harmlessness from AI Feedback, 2022

Model Distillation

Motivation – From Cloud to Edge

- Foundation models are massive (billions of parameters).
- Resource-limited deployments often require **low-latency, private, on-premise inference**.
- Solution: **Model distillation and compression.**





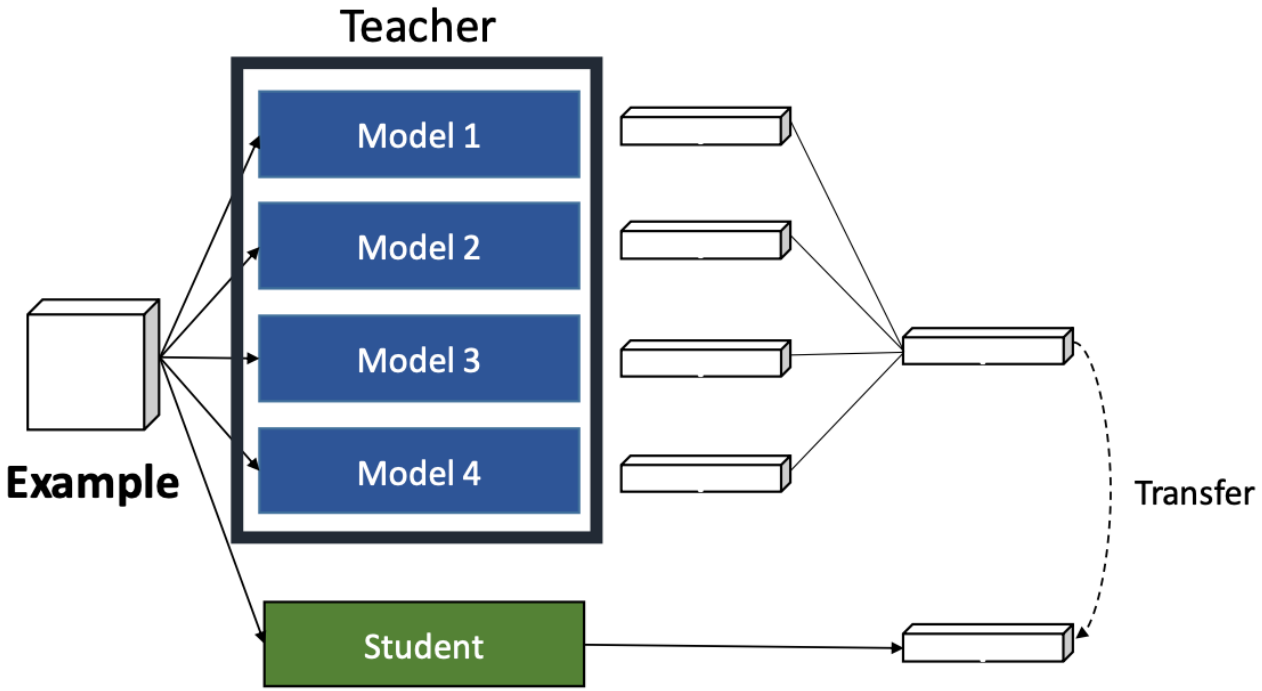
Model Distillation

Distillation and Optimization Techniques

- **Knowledge distillation:** train a smaller “student” to mimic a large “teacher.”
- **Quantization:** reduce precision (FP32 → INT8).
- **Pruning:** remove redundant parameters.
  - Training a large overparameterized network
  - Pruning the less-important weights or channels
  - Finetuning or re-training the pruned network
- **Neural architecture search (NAS):** design compact architectures automatically.

Knowledge Distillation

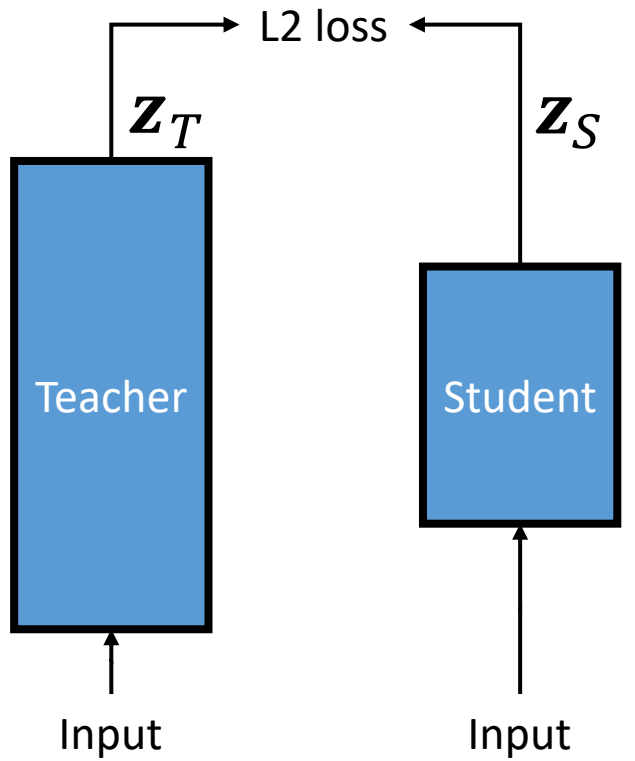
- General idea
  - Try to approximate bigger/more complex neural nets (models) with smaller neural nets (models) of similar generalization performance.
  - Have a student model model the teacher's function *directly*.



Transfer logits

- Model compression
  - Try to match the logits (the input to the softmax layer) produced by the teacher net
  - Treated as a regression problem (regressing the logit with L2-loss)

- Steps
  - Feed teacher with data
  - Obtain logits from teacher (transfer training set)
  - Train student on these logits (regression)



Unlabeled data can now be used to train the student!

Cristian Bucila et al., Model Compression, KDD 2006

Transfer logits

- Issues: If the output is a big N-way softmax, the targets are usually a single 1 and a whole lot of 0’s.

- How to reveal more information about the function on each training case?

Transfer soft targets

- Dark knowledge
  - Try to match soft targets produced by the teacher net

- Softmax layer with temperature, τ (normally set to 1)

$$S(z_i) = \frac{e^{z_j/\tau}}{\sum_j e^{z_j/\tau}}$$

- This will *soften* the posterior distribution

- Using a higher value for τ produces a softer posterior distribution over classes

Geoffrey Hinton et al., Distilling the Knowledge in a Neural Network, NIPS 2014 Deep Learning Workshop

Transfer soft targets

- Why does it work? Most of the knowledge in the learned model is in the relative probabilities of extremely improbable wrong answers

- Example:
  - Truth:

$$\mathbf{y} = \underbrace{\begin{pmatrix} 0 & 1 & 0 & 0 \end{pmatrix}^\top}_{\begin{pmatrix} \mathbb{P}_{\text{cow}} & \mathbb{P}_{\text{dog}} & \mathbb{P}_{\text{cat}} & \mathbb{P}_{\text{boat}} \end{pmatrix}^\top}$$

  - Teacher output:

$$\hat{\mathbf{y}} = \begin{pmatrix} 10^{-6} & 0.9 & 0.1 & 10^{-9} \end{pmatrix}^\top$$

  - Softened output:

$$\tilde{\mathbf{y}} = \begin{pmatrix} 0.05 & 0.4 & 0.3 & 10^{-3} \end{pmatrix}^\top$$

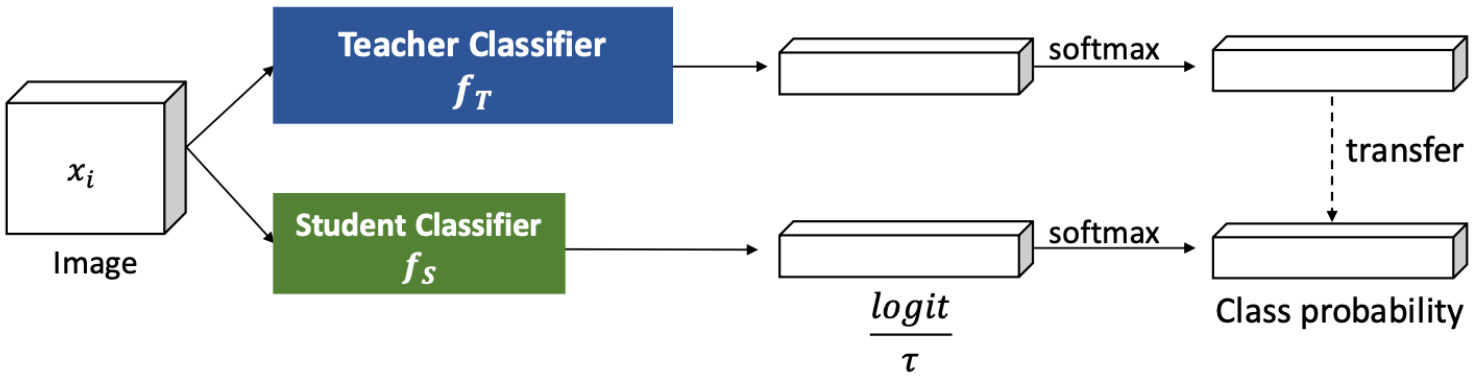
Softened outputs reveal the dark knowledge!

Knowledge Distillation

Transferring soft targets:

1. Feed teacher with data
2. Obtain soft targets from teacher - higher value for τ produces a softer posterior distribution over classes – reveal dark knowledge!
3. Train student on soft targets (with KL divergence) - works better to fit both the hard targets and the soft targets

Objective: $\sum_{x_i \in \mathcal{X}} \mathrm{KL}\left(\mathrm{softmax}\left(\frac{f_T(x_i)}{\tau}\right), \mathrm{softmax}\left(\frac{f_S(x_i)}{\tau}\right)\right)$

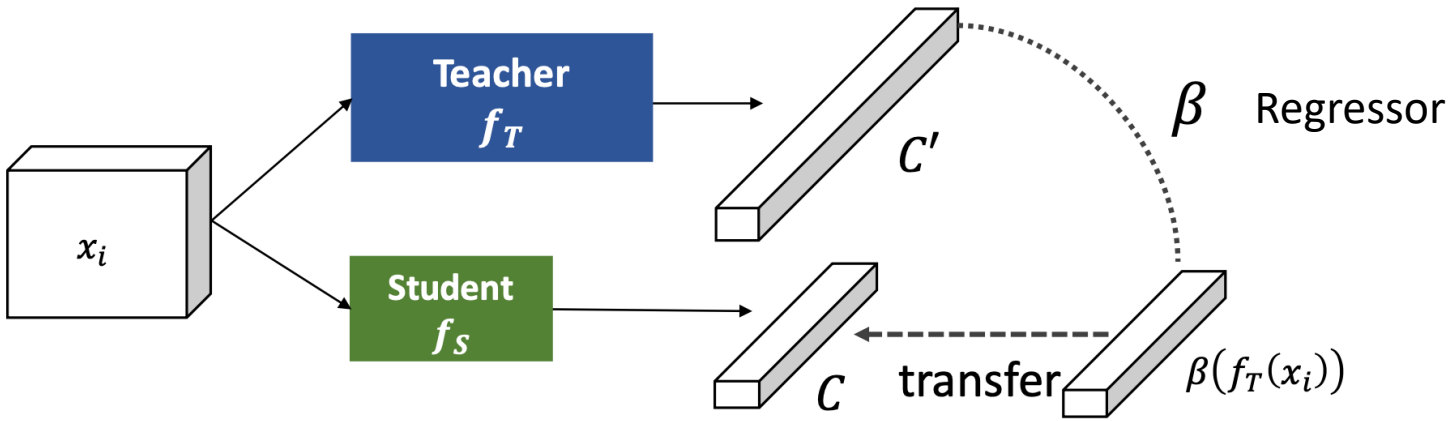


Geoffrey Hinton et al., Distilling the Knowledge in a Neural Network, NIPS 2014 Deep Learning Workshop

Transfer Hidden Activation

Transferring hidden activation:

1. Add a regressor to the guided layer, whose output matches the size of the hint layer
2. Train the student from the first layer up to the guided layer as well as the regressor parameters
3. Minimizing **Objective:** $\sum_{x_i \in \mathcal{X}} \left\| f_T(x_i) - \beta\left(f_S(x_i)\right) \right\|_2^2$



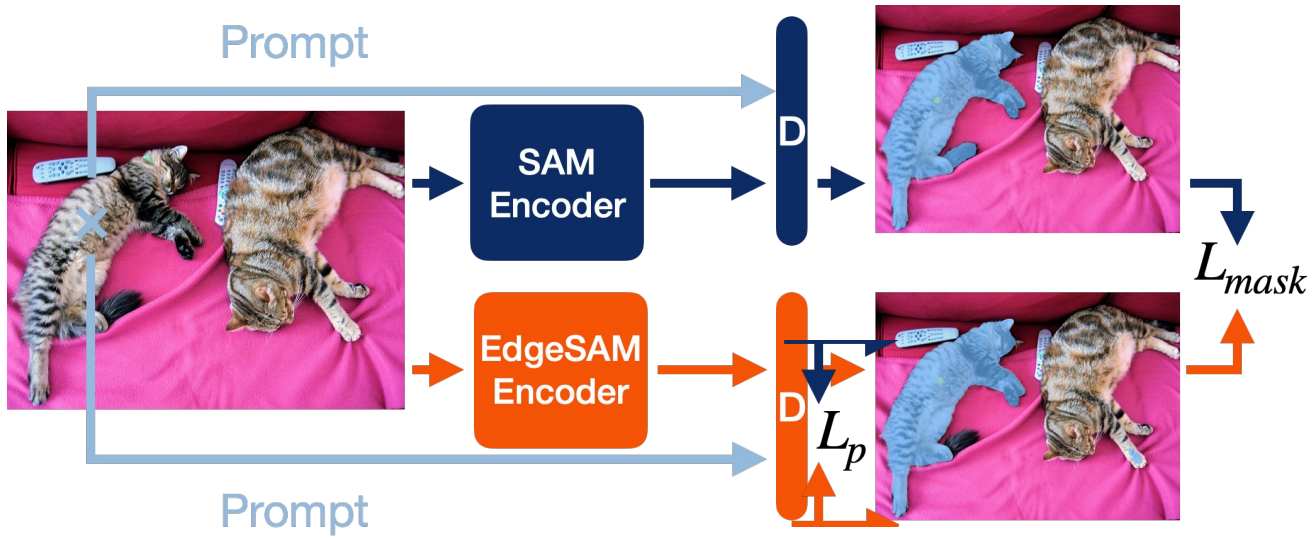
Romero *et al.*, FitNets: Hints for Thin Deep Nets, ICLR 2015

Model Distillation

CutCha Photo



Example



EdgeSAM

- First SAM variant that can run at over 30 FPS on an iPhone 14, without network quantization
- Over *37 times faster* than SAM on NVIDIA 2080 Ti

Chong Zhou et al., EdgeSAM: Prompt-In-the-Loop Distillation for On-Device Deployment of SAM. IJCV 2025