Lecture 9:

Generative AI for Image Creation (Part II)

Visual Computing Systems
Stanford CS348K, Spring 2024

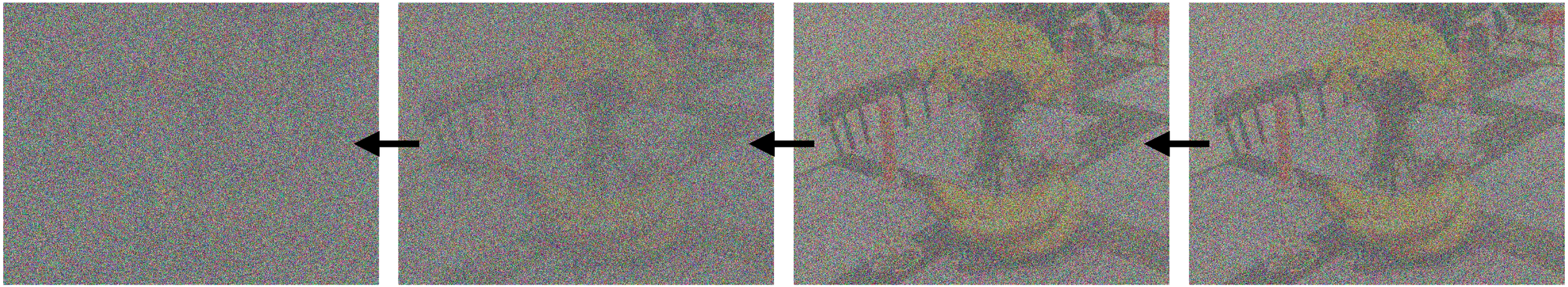
Suppose you are given a data of images x_i

- You'll like to draw a sample according to the underlying data distribution p(x)

Diffusion-based image synthesis

Iterative Markov-chain Monte-carlo (MCMC) process to generate a sample x (an image) from distribution p(x) of observed images

Forward diffusion: iteratively add noise $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$



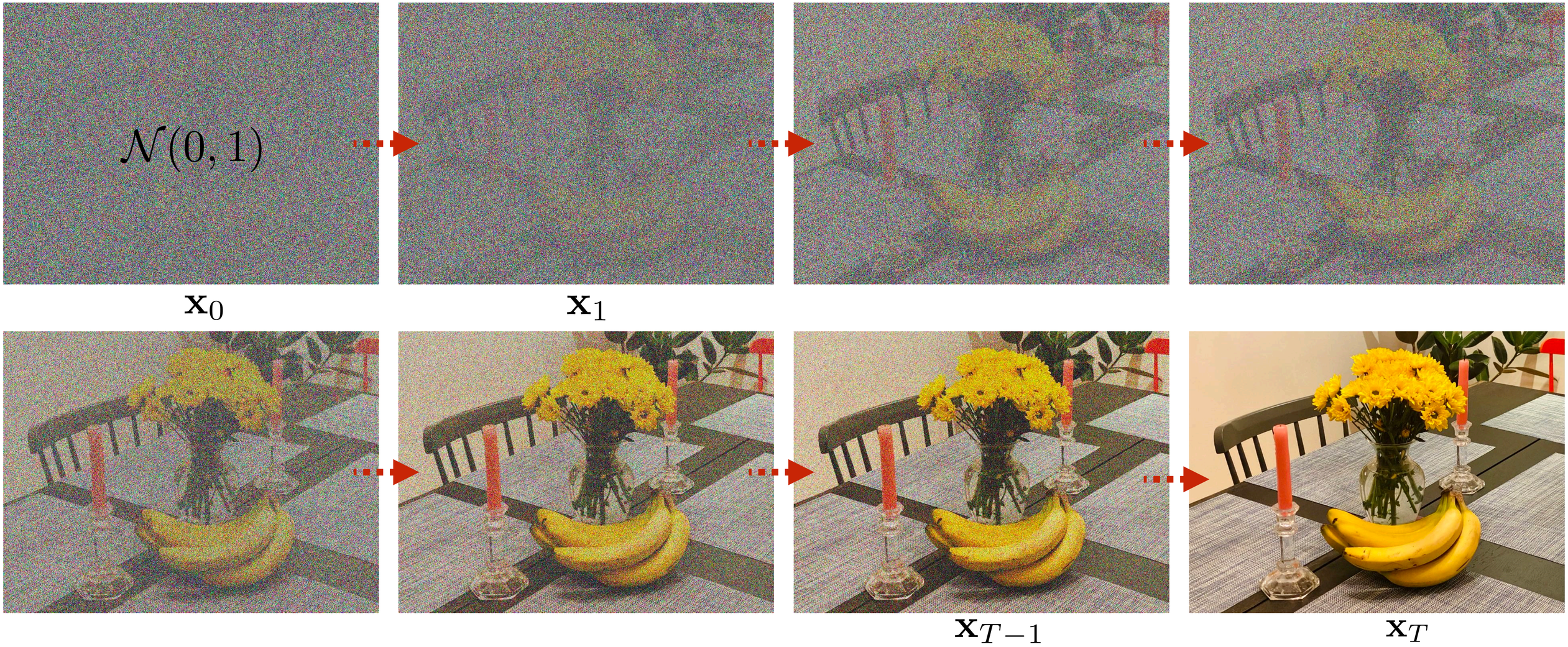
$\mathbf{x}_T$ $\mathbf{x}_{T-1}$ $\mathbf{x}_1$ $\mathbf{x}_0$

Diffusion-based image synthesis

Reverse: iteratively remove noise noise from random sample to obtain image from p(x)

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \sqrt{2\epsilon}\ \mathbf{z}_i, \quad i = 0, 1, \cdots, T$$

("score function")



Guided diffusion

- Assume we know $p(\mathbf{y} \mid \mathbf{x})$ for random variables x and y.
  - Example: x is an image, y is a string describing the image
  - Given an image (x), infer a caption (y)

$$p(\mathbf{x} \mid \mathbf{y}) = p(\mathbf{x})p(\mathbf{y} \mid \mathbf{x}) / \int p(\mathbf{x})p(\mathbf{y} \mid \mathbf{x})\mathrm{d}\mathbf{x}$$ **(Bayes Rule)**

Bayes for score function

$$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x})$$

(Unguided score function)

Modify image x so that image is more likely [to come from the training set]

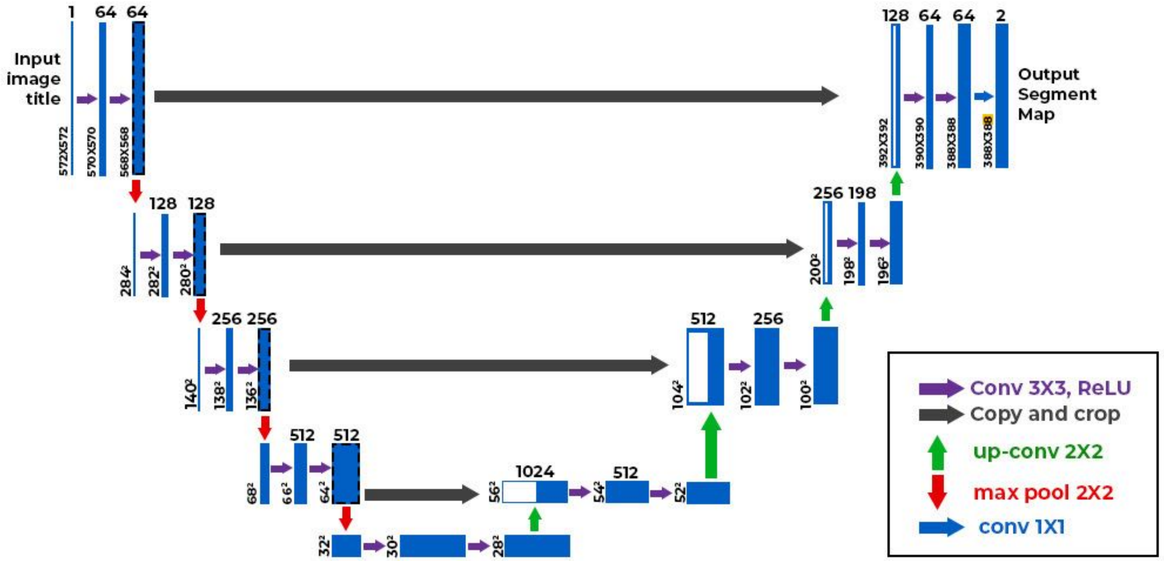
(Prompt guidance)

Modify image x to make the prompt a more likely description of the image

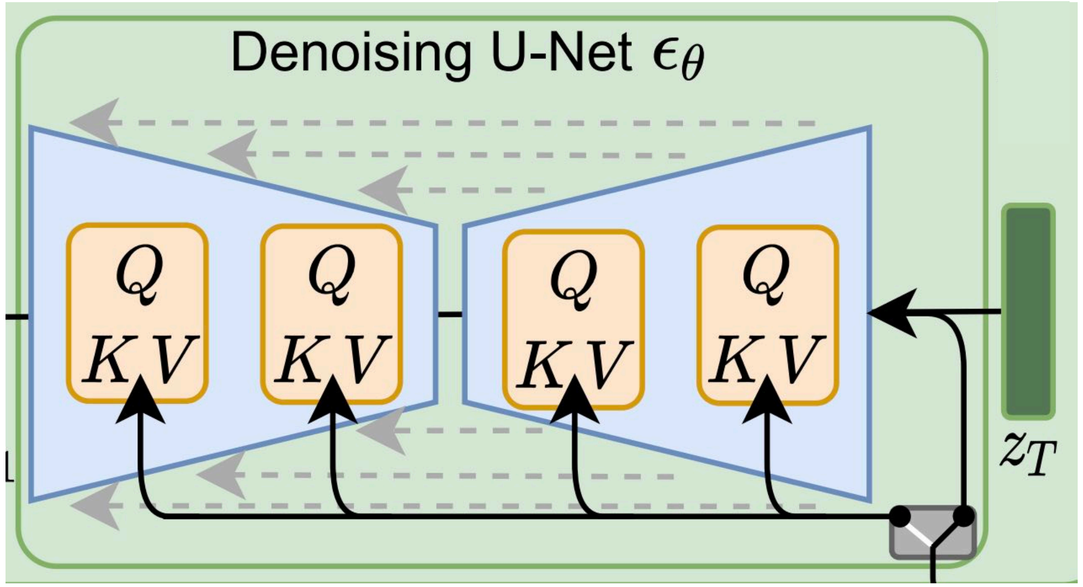
Performance/efficiency optimizations

A basic U-NET architecture

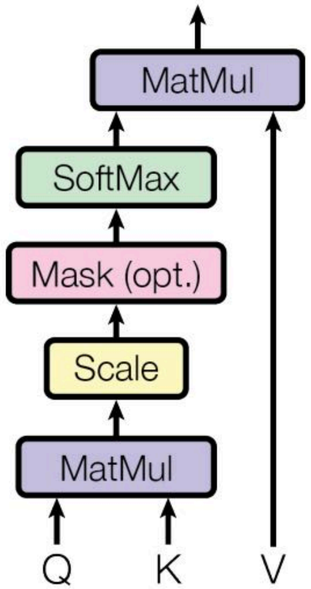
Classic U-Net for image segmentation
(Basic blocks are convolutional layers)



U-Net used in modern diffusion models
(Basic blocks are transformer modules with cross-attention layers)



Attention module



Let $\mathbf{S} = \mathbf{Q}\mathbf{K}^\top \in \mathbb{R}^{N \times N}$

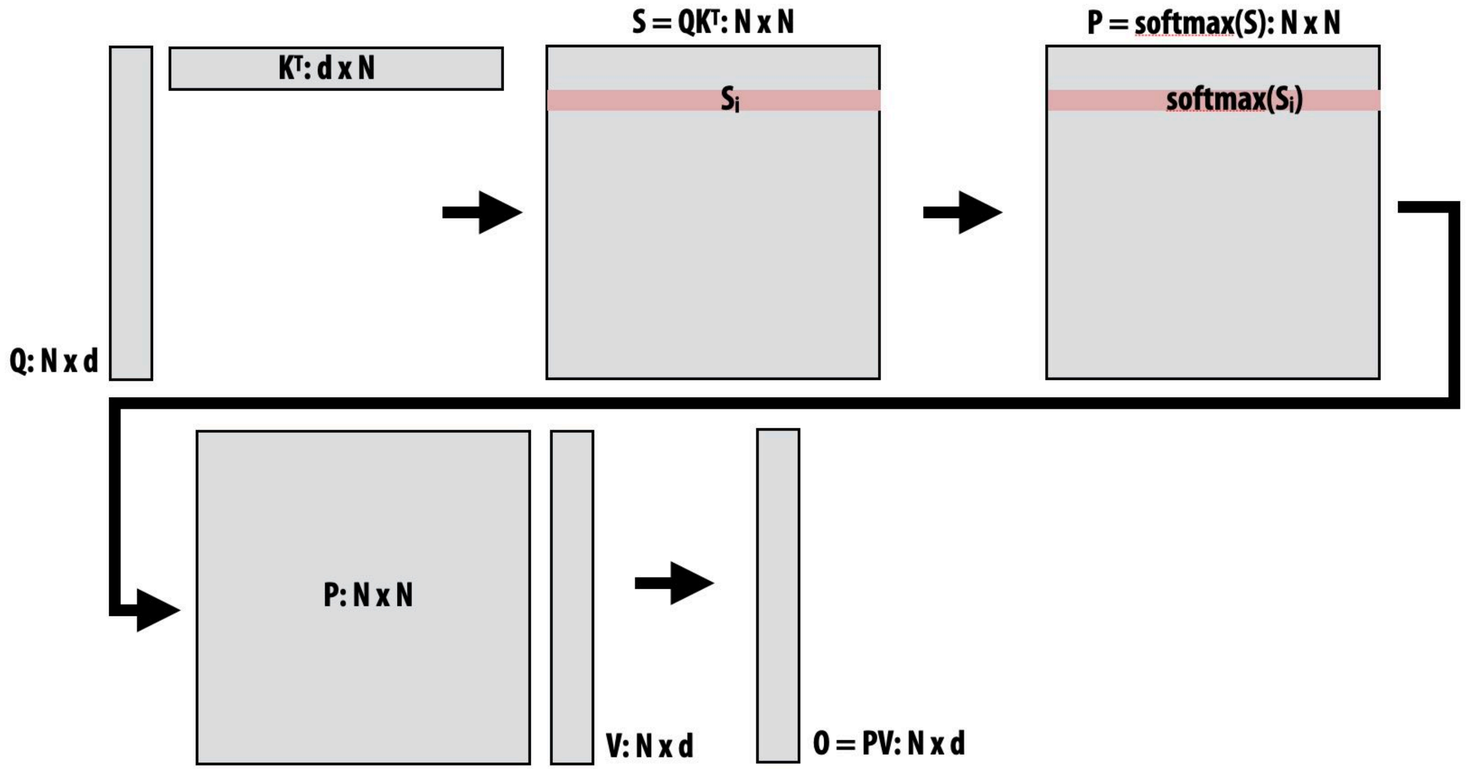
Let $\mathbf{P} = \mathrm{softmax}(\mathbf{S}) \in \mathbb{R}^{N \times N}$

Let $\mathbf{O} = \mathbf{P}\mathbf{V} \in \mathbb{R}^{N \times d}$

Where

$\mathrm{softmax}(\mathbf{S})$

Is softmax over the rows of S



For a row x:

$$\mathrm{softmax}(\mathbf{x}) = \frac{f(\mathbf{x})}{l(\mathbf{x})}$$

Where:

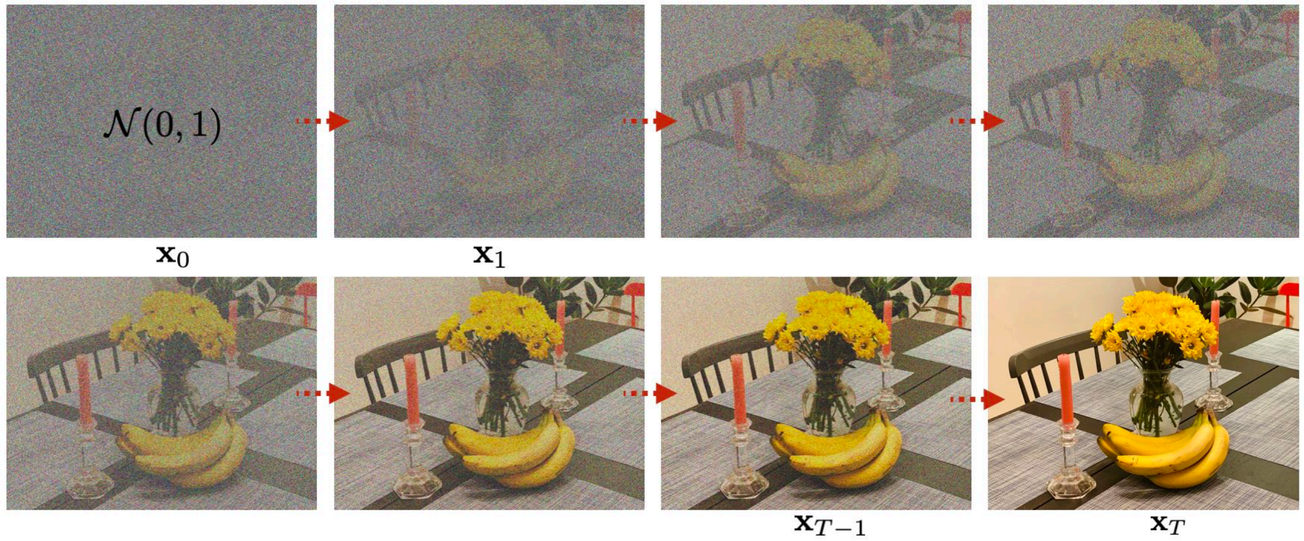
$$f(\mathbf{x}) = \begin{bmatrix} e^{\mathbf{x}_1 - m(\mathbf{x})} & e^{\mathbf{x}_1 - m(\mathbf{x})} & \dots & e^{\mathbf{x}_B - m(\mathbf{x})} \end{bmatrix}$$

$$m(\mathbf{x}) = \max_i(\mathbf{x}_i)$$

$$l(\mathbf{x}) = \sum_i f(\mathbf{x})_i = \sum_i e^{\mathbf{x}_i - m(\mathbf{x})}$$

Performance challenges of inference

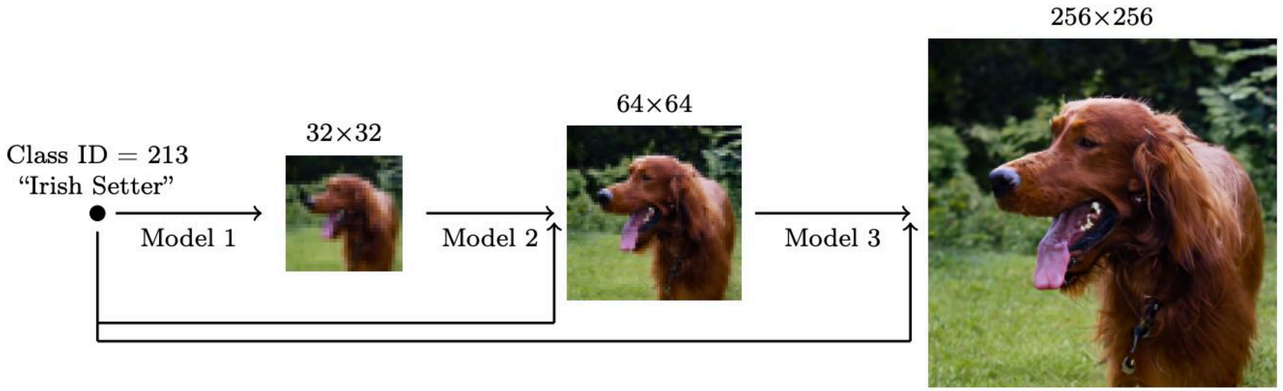
- **Diffusion is an iterative process:**
  - Requires many steps to convergence



- **Ways to improve inference efficiency**
  - Diffuse in low-dimensional (latent) space
  - Superresolution techniques
  - Learn to take bigger steps

Superresolution

- Use diffusion to produce a low-resolution image
- Then subsequent models perform neural superresolution

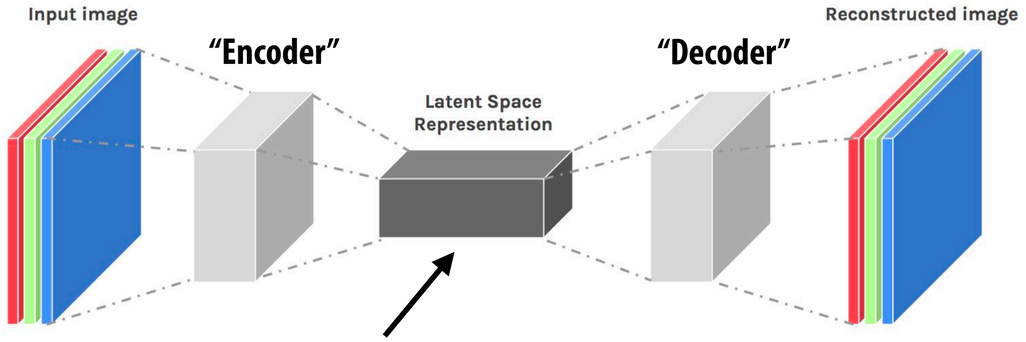


Cascade of diffusion models

Bicubic upsampling vs. two forms of learned upsampling

Perform diffusion in latent space

- Main idea: perform diffusion in the lower dimensional latent space of images, not in high-dimensional RGB pixel space
- After diffusing a latent representation, "decode" latent to final image



If this latent representation is compact, then it is a compressed representation of the input image

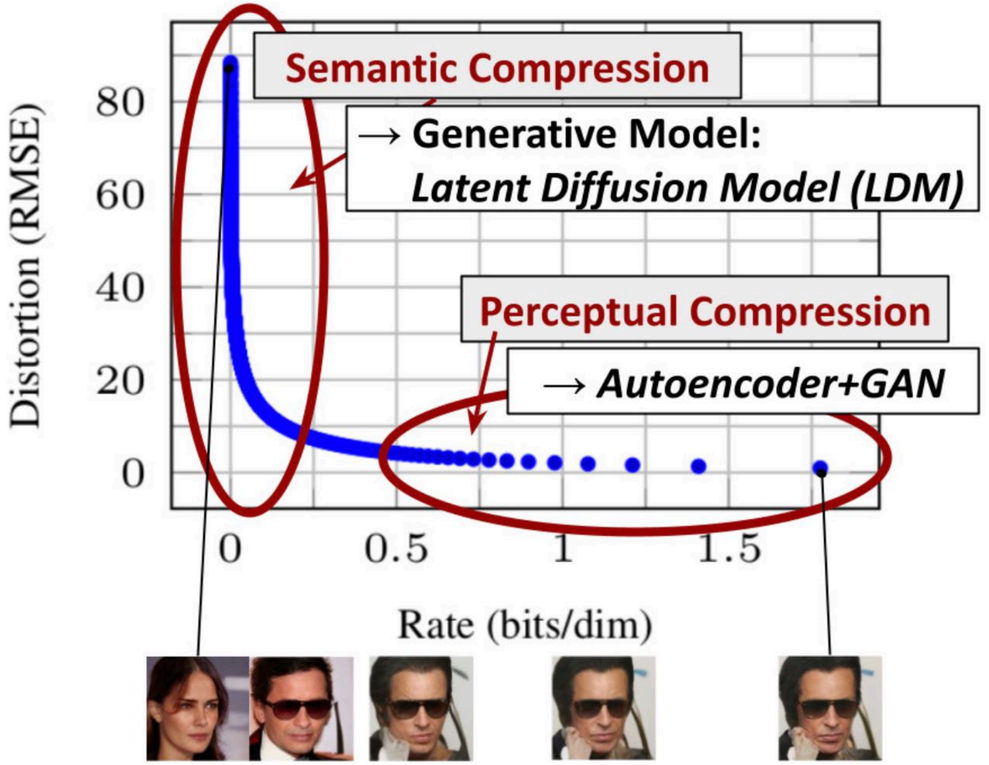
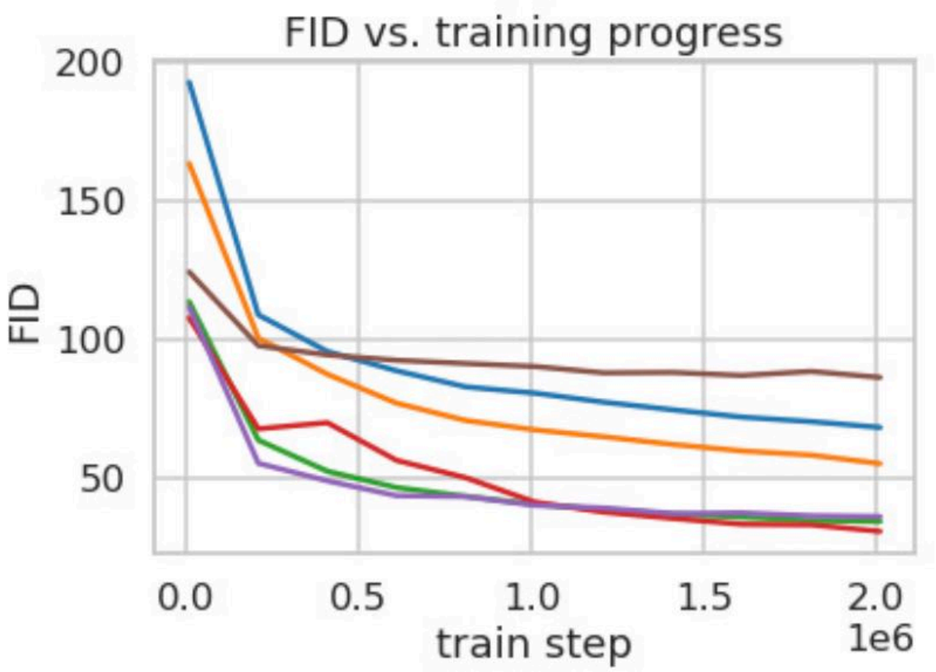


Figure credit: https://medium.com/@birla.deepak26/autoencoders-series-daad78df9350

Perform diffusion in latent space

- Implications to both training efficiency and inference efficiency



Per-pixel representations, can represent data well, but require significant training to learn good models

"Sweet spot": learns good model + trains quickly

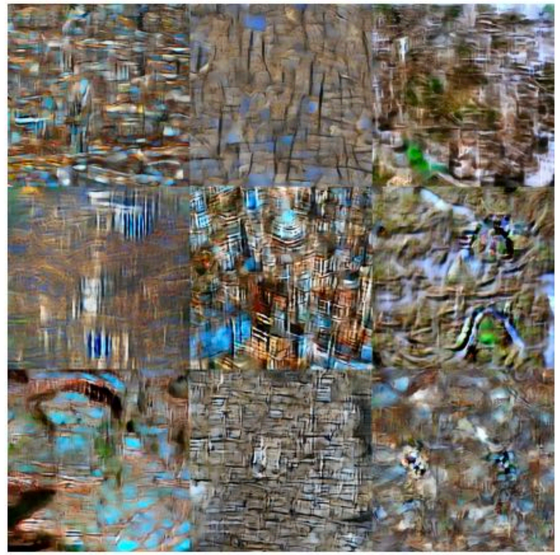
Latent representation too compressed (cannot represent data well)

Learn to take larger steps

- Given a diffusion model, learn a new (second) diffusion model that reproduces multiple steps of the diffusion process
- This is a form of "model distillation": training a "student" model to emulate the output of a teacher. Here, the teacher's output is multiple steps of the diffusion process

Ours 2 steps

Ours 4 steps



DDIM 2×2 steps

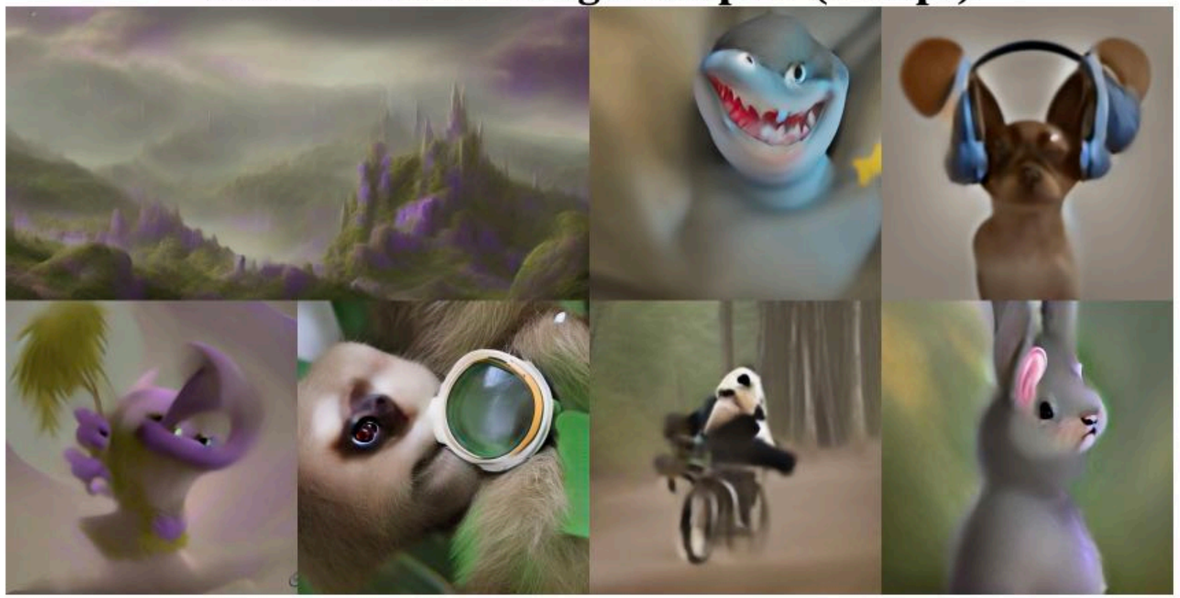
DDIM 4×2 steps

Prompt: "*A beautiful castle, matte painting.*"

Learn to take larger steps

Distilled Text-to-Image samples (4 steps)

Native Text-to-Image samples (4 steps)



Native Text-to-Image samples (8 steps)

Controlling the output of diffusion models

Text conditioning

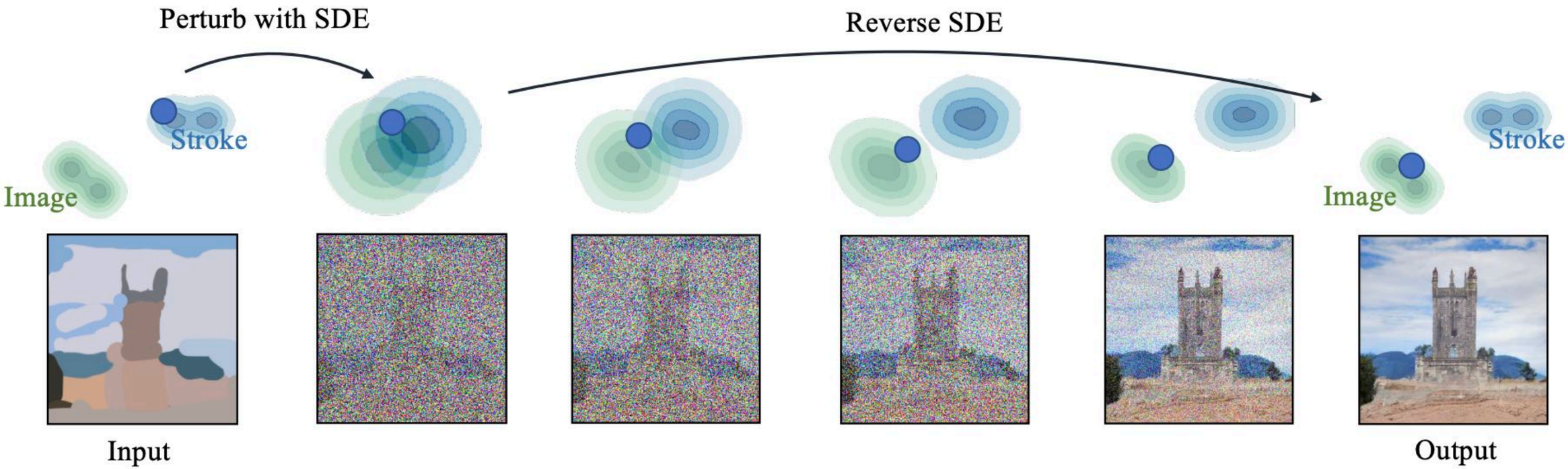
"A classroom of many attentive college students"

I want the visual output to look LIKE THIS.
(Other forms of conditioning)

img2img (RGB image conditioning)

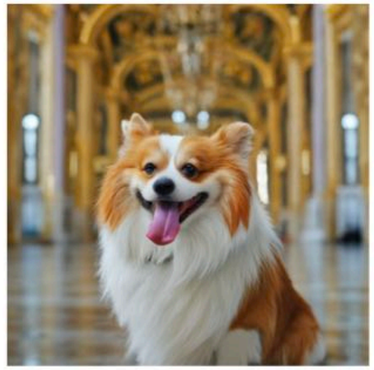
1. Start with a guide image (a target)
2. Add "small" amount of noise
3. Iteratively denoise to produce sample from p(x)

"Guide toward a visual target"

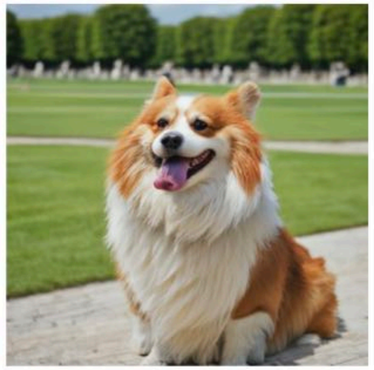


Specialization to a specific concept

Input images



A [V] dog in the Versailles hall of mirrors



A [V] dog in the gardens of Versailles

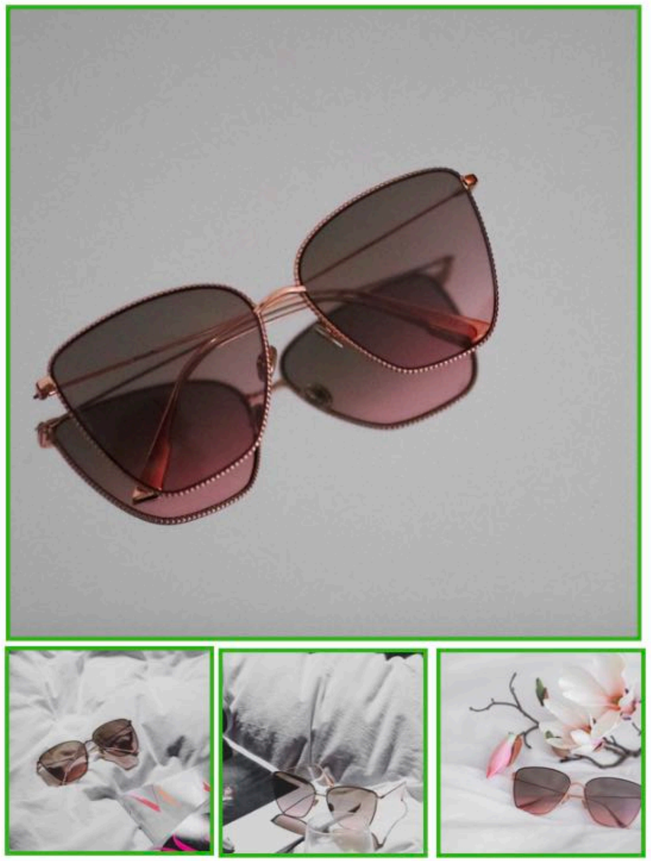
A [V] dog in Coachella

A [V] dog in mountain Fuji

A [V] dog with Eiffel Tower in the background

Specialization to a concept

Input images



A [V] sunglasses in the jungle

A [V] sunglasses worn by a bear

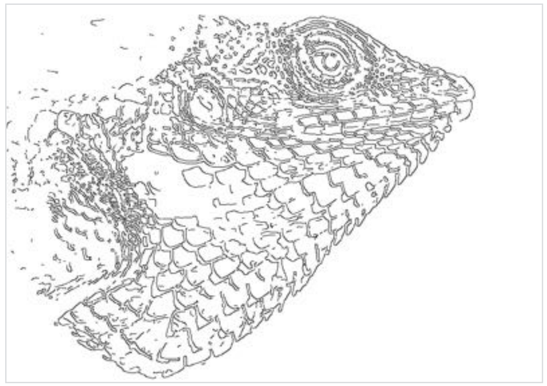
A [V] sunglasses at Mt. Fuji

A [V] sunglasses on top of snow

A [V] sunglasses with Eiffel Tower in the background

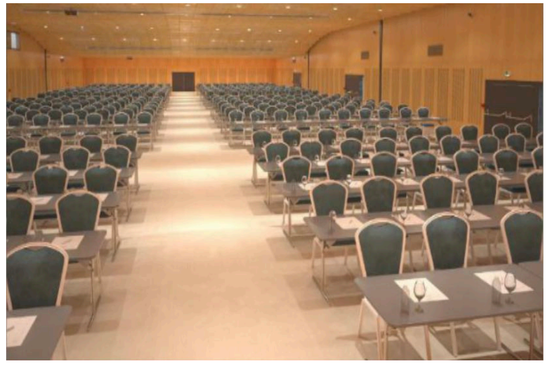
Other forms of conditioning

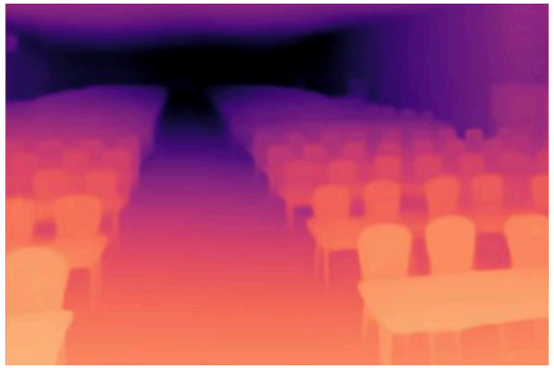
Common tactic: automatically create paired data (via image processing/analysis)



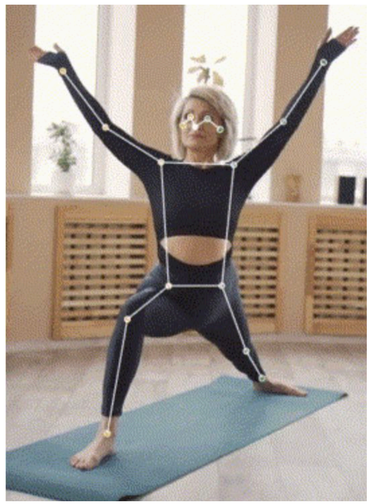
Edge detection

Segmentation



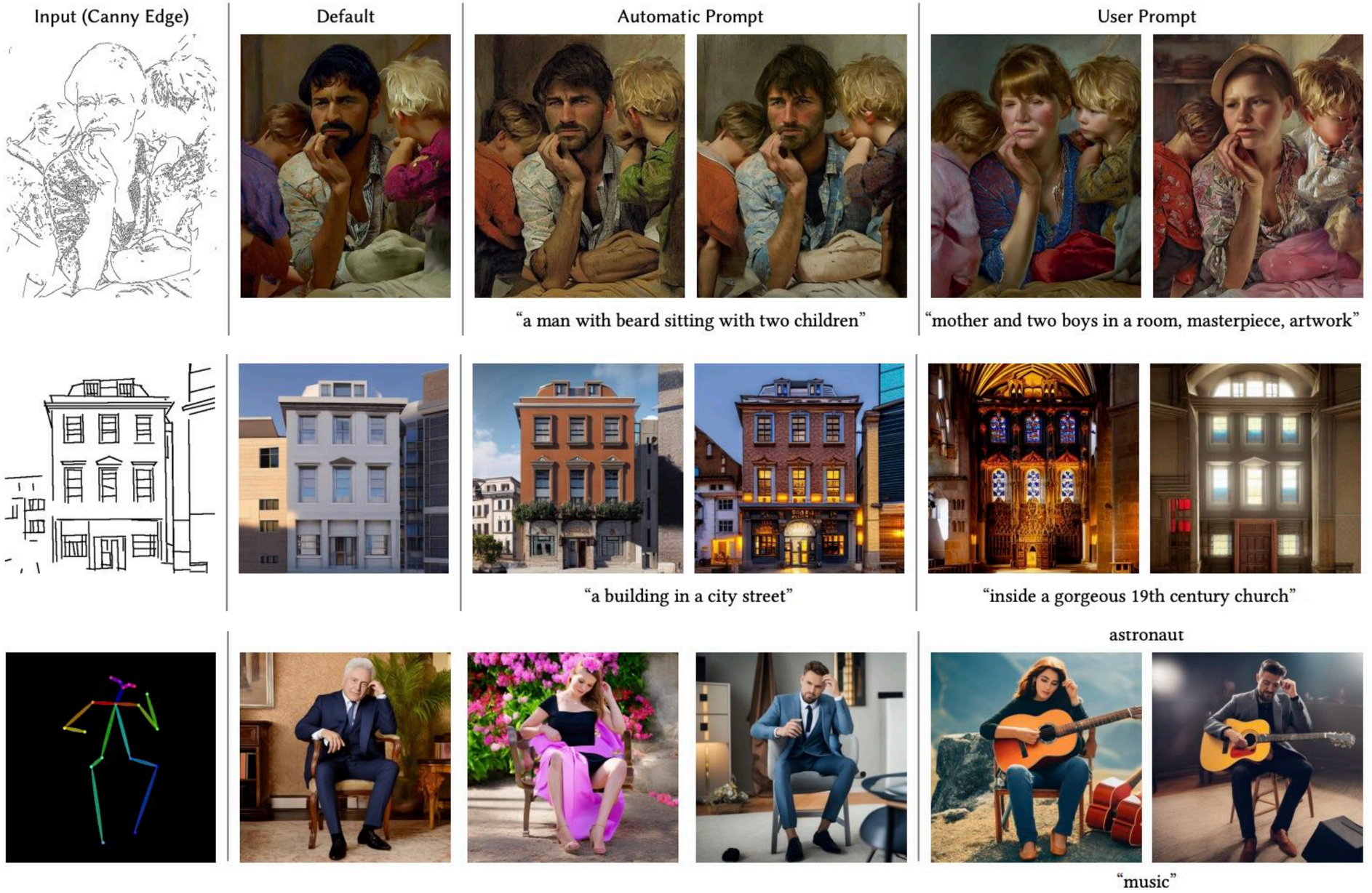


Depth



Another example:
Pose Estimation

Alternative forms of control



Reducing the cost of training

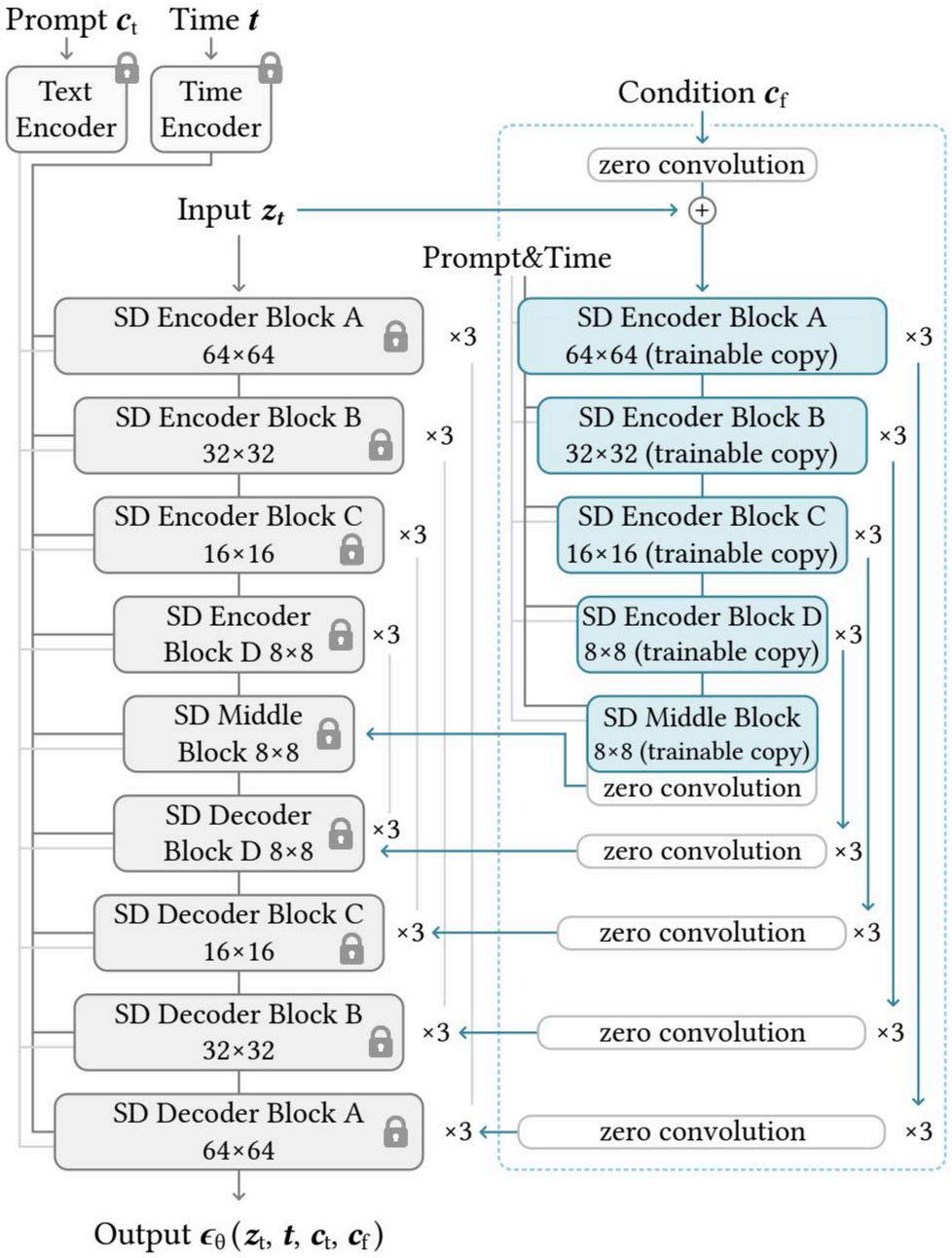
- Example open source text/image training dataset:
  - LAION-5B (5.5B image/test training pairs)

- Significant compute cost to train a model

ControlNet

- **Key idea: duplicate diffusion network**
  - **Original copy retains original weights**
  - **ControlNet copy learns how to modify activations to respect new control signals**

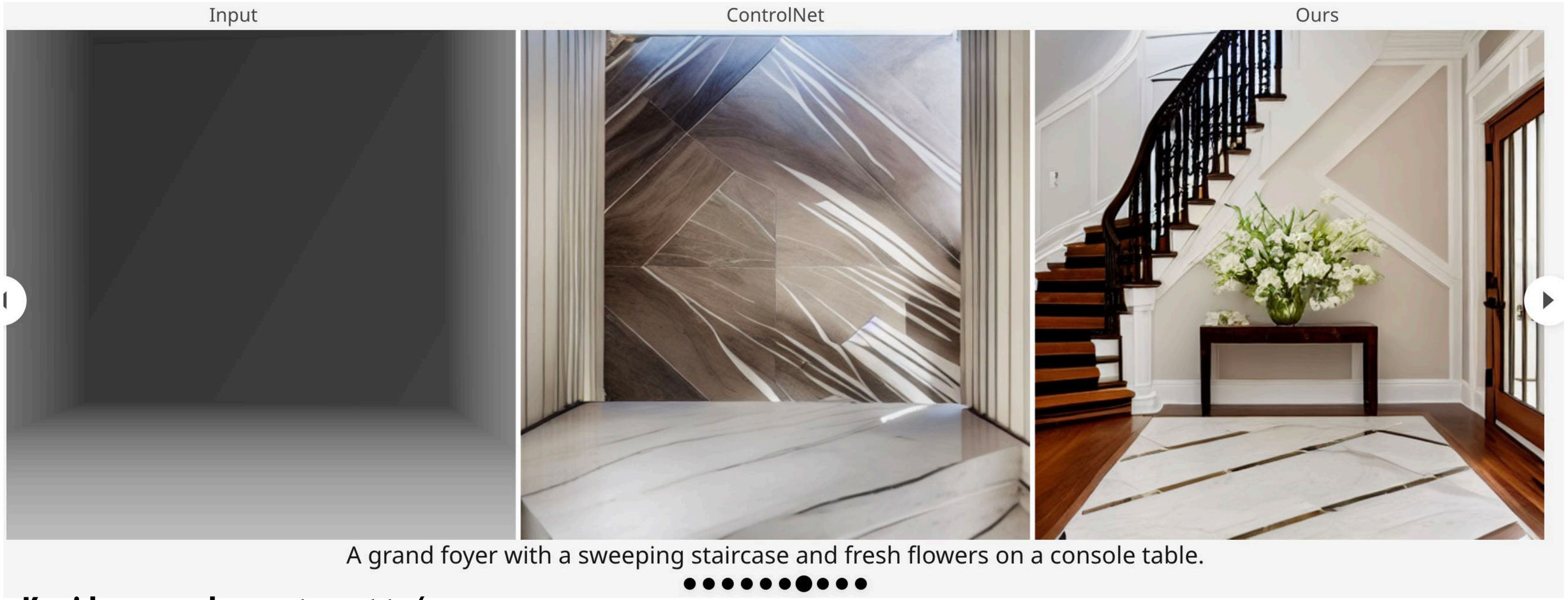
- **Intuition:**
  - **Retain strong priors of backbone trained on large body of images (expensive training)**
  - **Learn how to respect new control signal from a much smaller number of images and few optimization steps (inexpensive)**



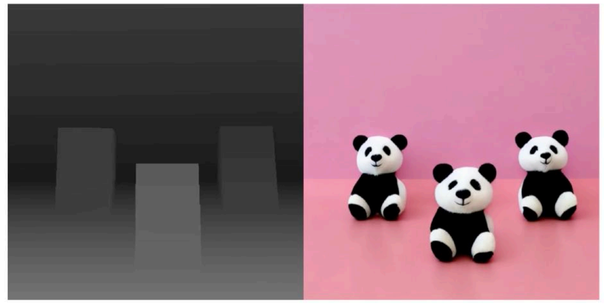
(a) Stable Diffusion (b) ControlNet

I kind of have an idea of what I want visually,
but it is challenging to produce the conditioning

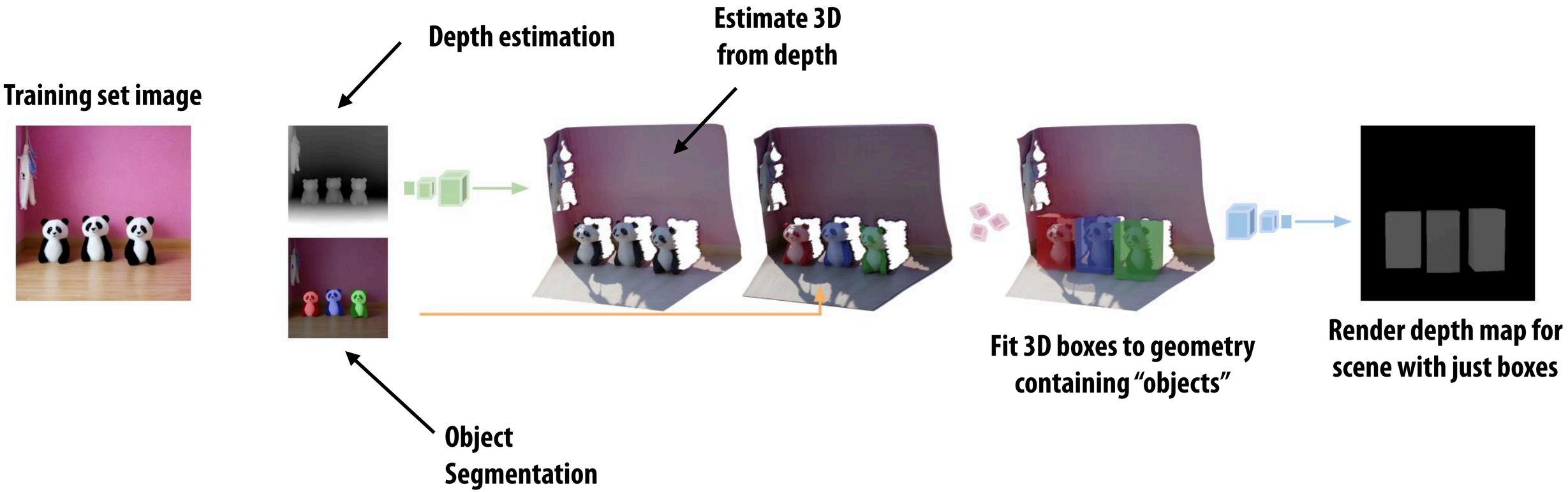
Loose control



Key idea: user does not want to (or may not have capability to) specific visual controls precisely. Just have the user "block out" the basic shape of the scene.



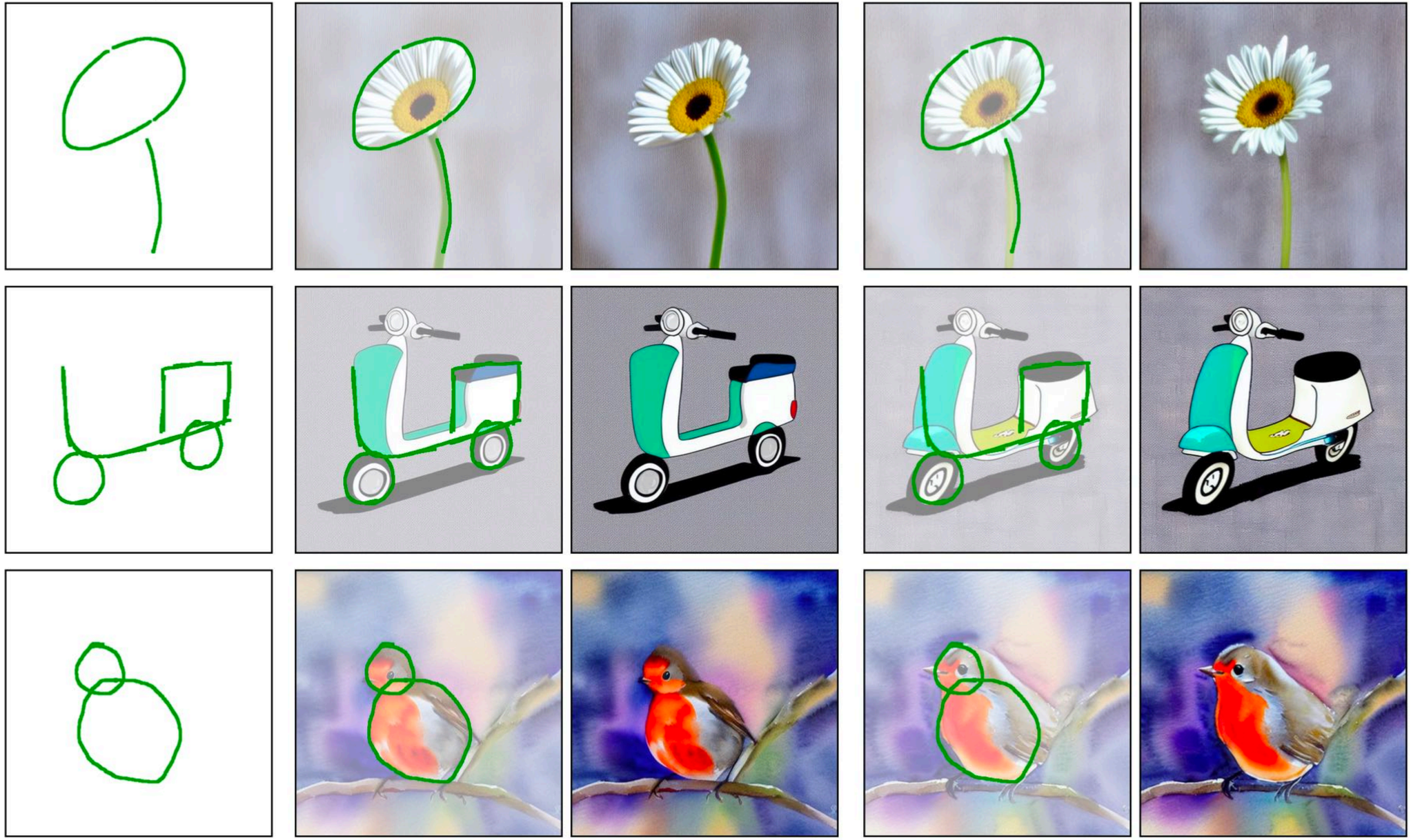
Generating training data for loose control



Sketch-to-image

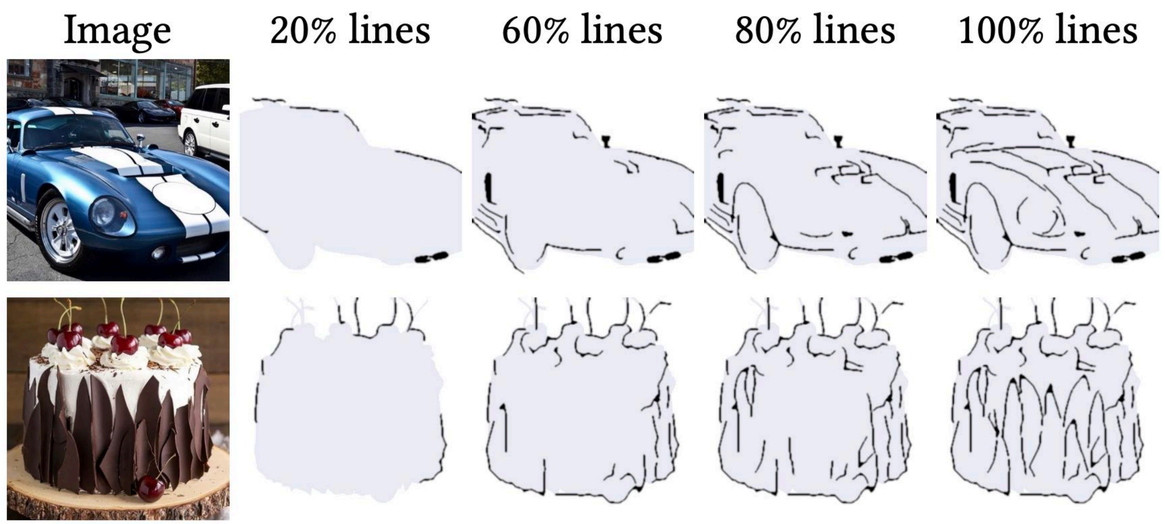
Tight control (ControlNet)

Looser control (Blended Renoising)



Partial sketch to image training data generation

Creating "partial-sketch" training data



Find edge lines

Identify contour lines (via segmentation)

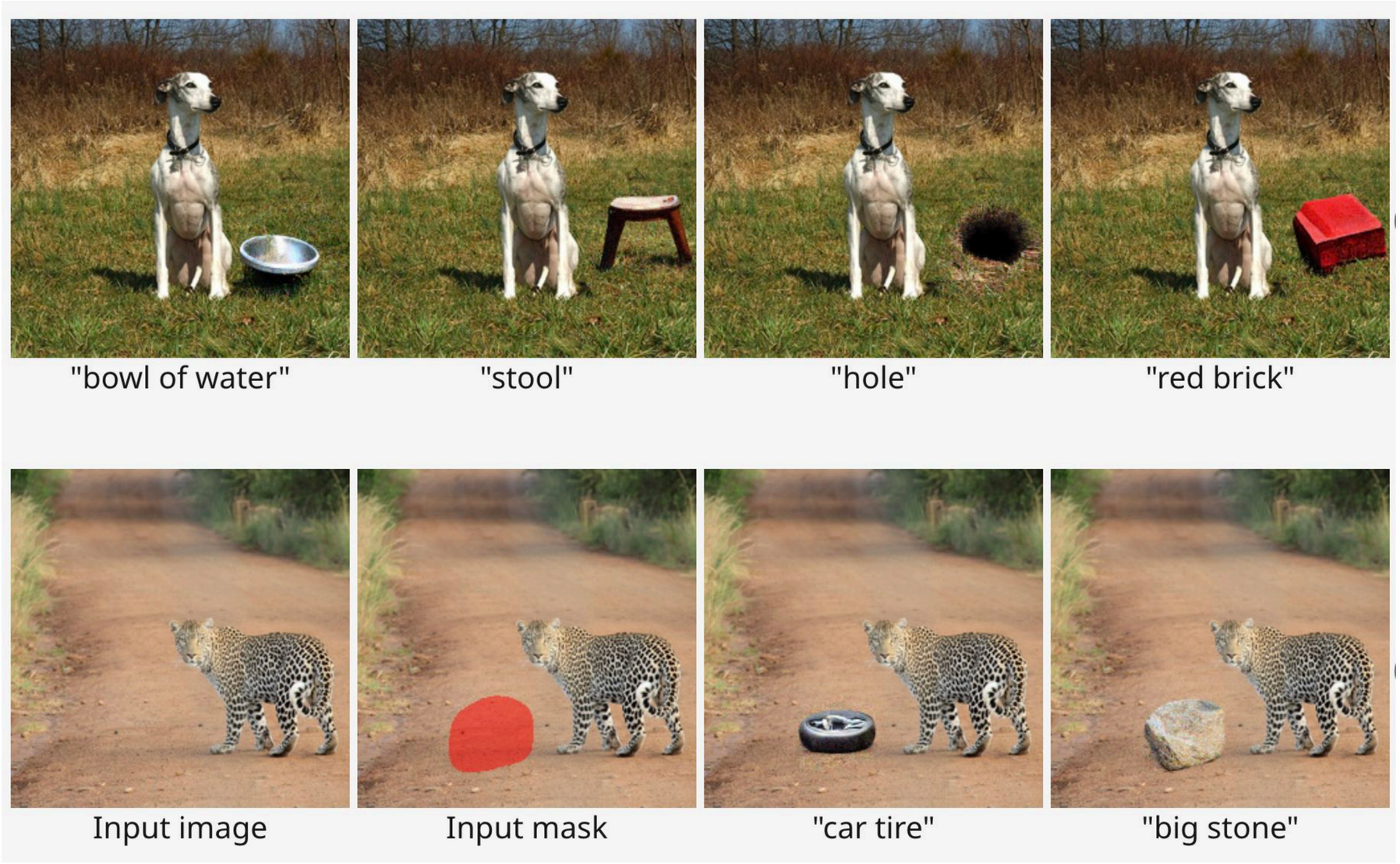
Progressively remove edge lines that are farthest from contours.

I want to modify the output in a SPECIFIC REGION.
I want to control COMPOSITION.

Inpainting (apply [new] prompt to a region)

User specifies mask for region of interest and text prompt for that region.

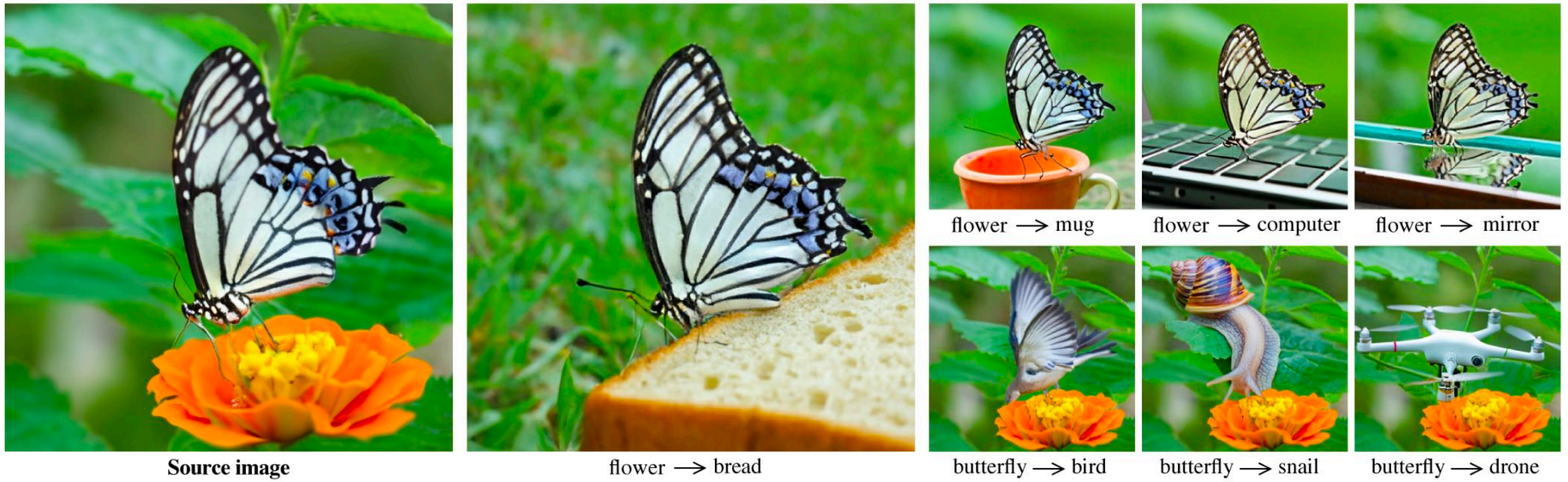
Image outside of region remains almost the same.



Use change in text prompt to trigger change in image

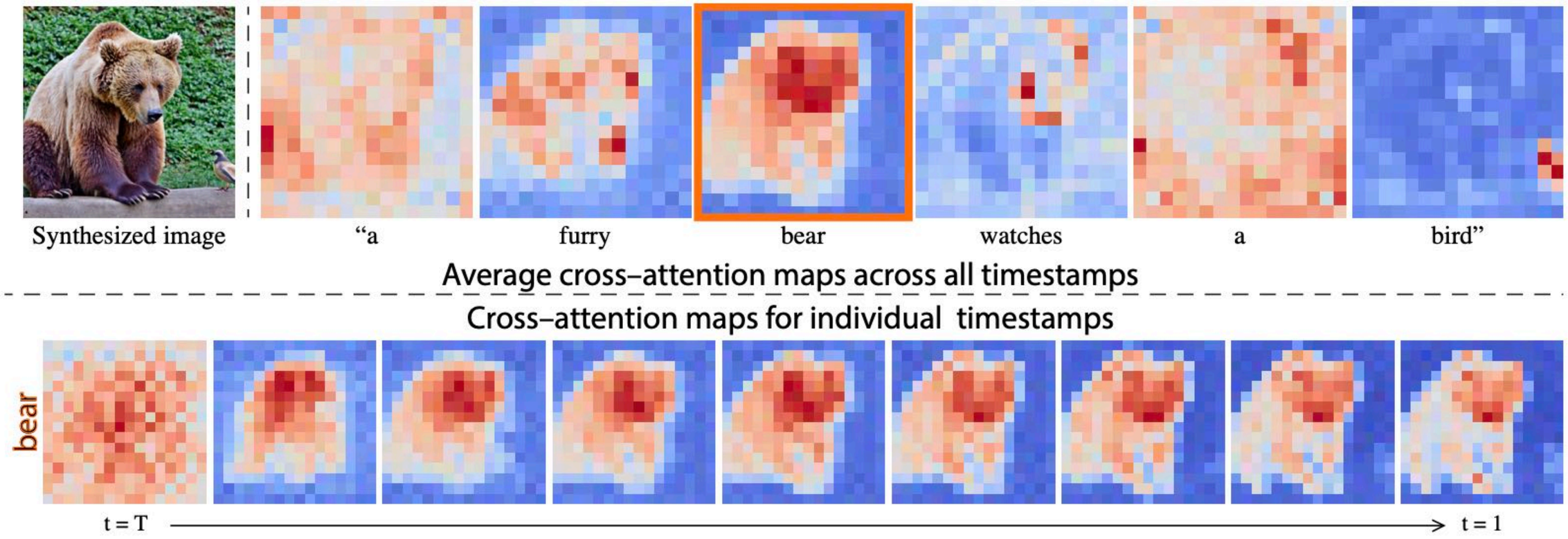
"A basket full of apples."

"A photo of a butterfly on a flower."



“Masks” come from learned attention

- Use attention masks from original generation process to constrain what pixels can change after prompt is edited



Using text to describe how to change the image

"Swap sunflowers with roses"

"Add fireworks to the sky"

"Replace the fruits with cake"

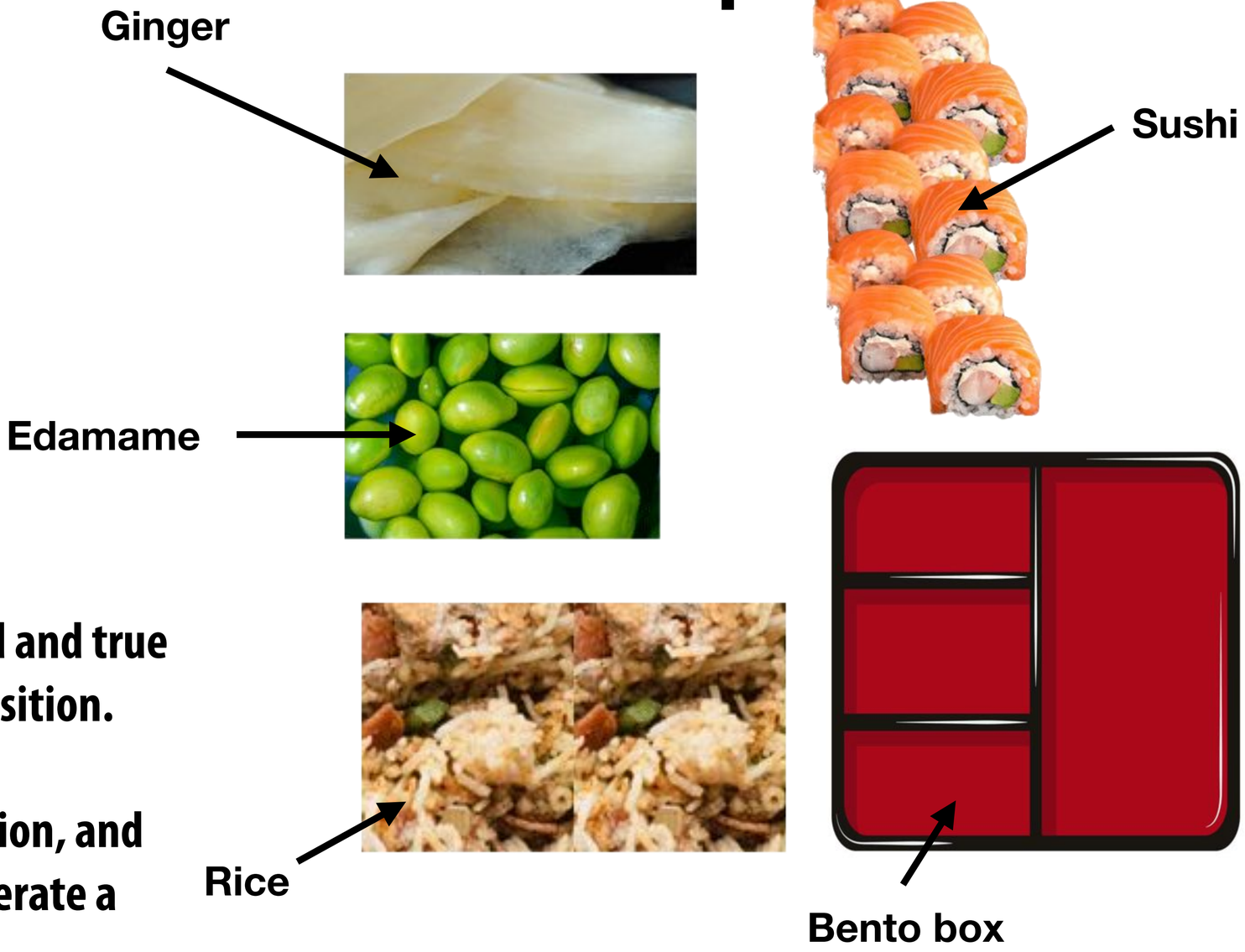
"What would it look like if it were snowing?"

"Turn it into a still from a western"

"Make his jacket out of leather"

Leveraging layer information to control composition

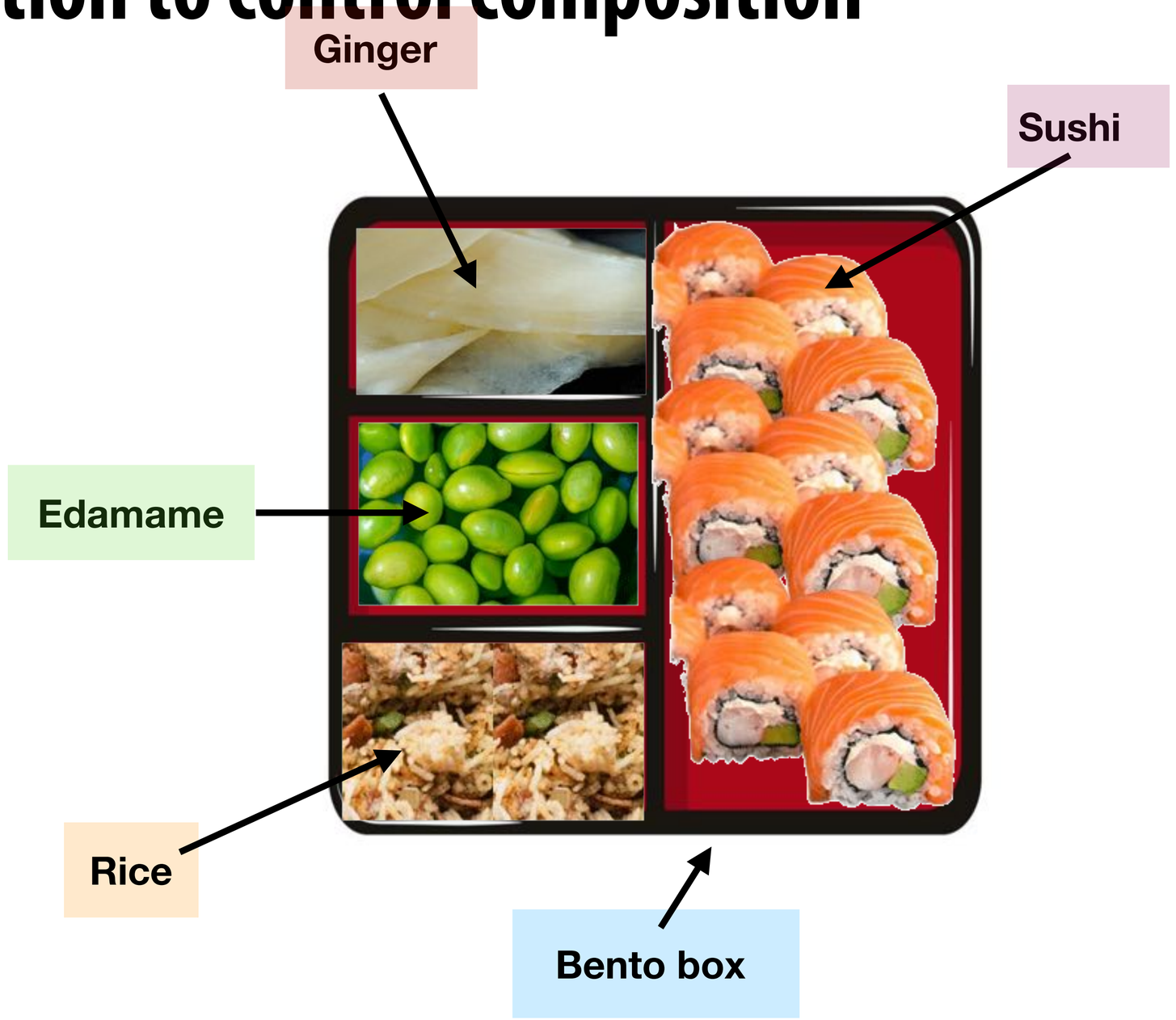
Prompt + a rgba per layer



- Key idea: layer-based edited is a tried and true way to manipulate an image's composition.
  - User manipulates layers
  - Model receives per-layer information, and leverages this information to generate a globally harmonious image

Leveraging layer information to control composition

"A bento box with rice, edamame, ginger, and sushi."



Leveraging layer information to control composition

"A bento box with rice, edamame, ginger, and sushi."

Reminder: key aspect in the design of any system

Choosing the "right" representations for the job

- Good representations are productive to use:
  - Embody a "preferred" way of thinking about a problem

- Good representations enable the system to provide useful services:
  - Validating/providing certain guarantees (correctness, resource bounds, conversion of quantities, type checking)
  - Performance optimizations (parallelization, vectorization, use of specialized hardware)
  - Implementations of common, difficult-to-implement functionality (complex array indexing code, texture mapping in 3D graphics, auto-differentiation, etc.)

Here: choosing the right representation is choosing controls that are most useful to an editing task

- What is the type of control that aligns with the users thought process / mental model?
  - Text is often an ambiguous, imprecise, or flat out inefficient way to describe visual intent

- Examples:
  - Users want to control spatial composition
    - "Dog on the left" vs. dragging a layer to the right location
  - Users want to "block out" an idea, and have the diffusion model "fill in the details", "correct proportions", "harmonize the image"
  - Users want to express intent via an example: "I want it to look LIKE THIS!"

Discussion:

Propose a type of edit that you would like to make to images

How does the user "think" about what they are trying to change (are they worried about details, composition, a particular "axis" of change (e.g, adjust smile but not eyes)

How could to generate supervision to train a model to support this type of control?

Summary

- Diffusion-based generation produces high quality ("plausible") image output
  - Step 1: get generation to produce output that models the training data well

- Step 2: ongoing research on
  - New ways to help users exert guide/control over the generation process
  - Improving the efficiency of diffusion model training/evaluation

- This line of work is a great example of many of the issues and concerns we've discussed in this class
  - Are we optimizing for the right metrics?
  - How to achieve high performance (through better algorithms, or systems techniques)
  - What are the implications of these technologies?