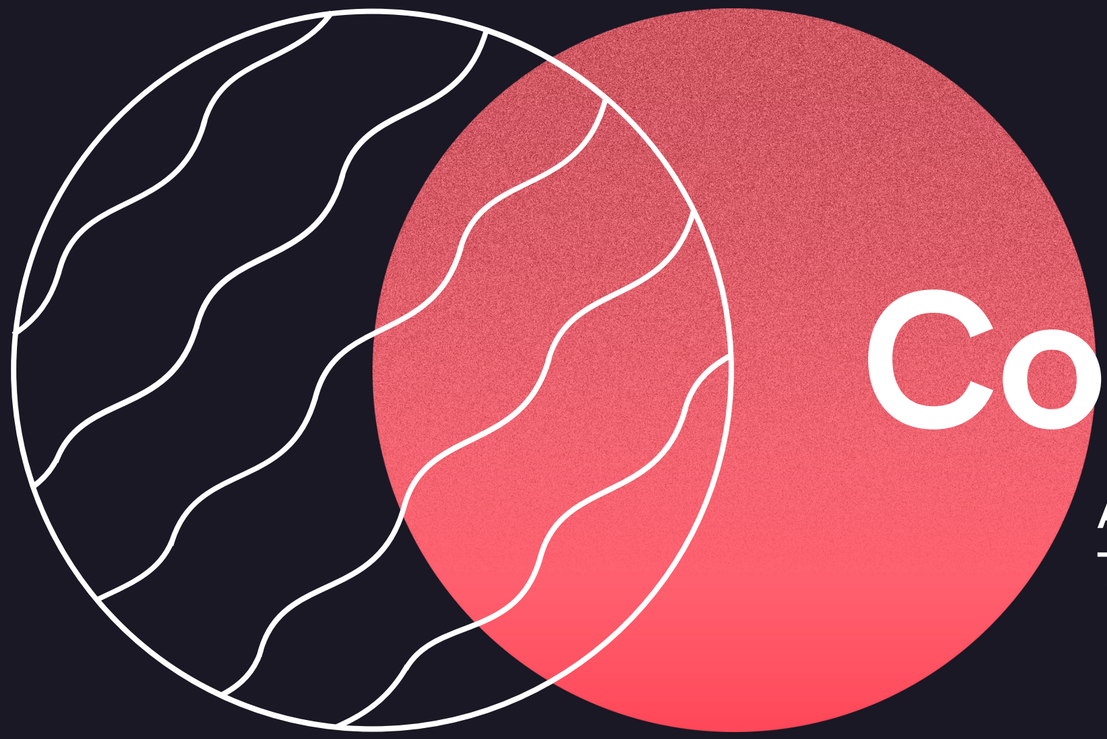


ControlNet

Adding Conditional Control to Text-to-Image Diffusion Models

Intro

I am a 1st Year PhD student in Computer Science at CUNY Grad Center

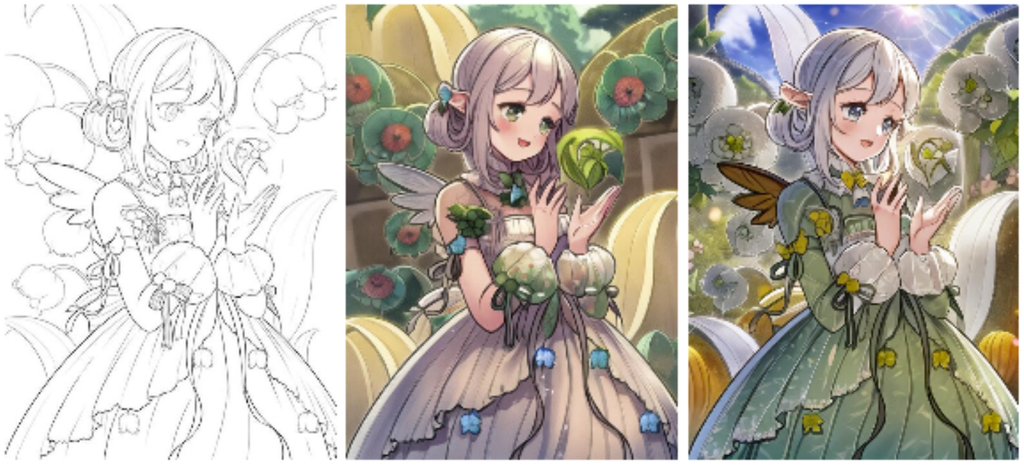
Paper I am summarizing today is *Adding Conditional Control to Text-to-Image Diffusion Models*

WHY THIS PAPER

What is the problem ?

How to control pretrained large diffusion models to support additional input conditions in image processing tasks.

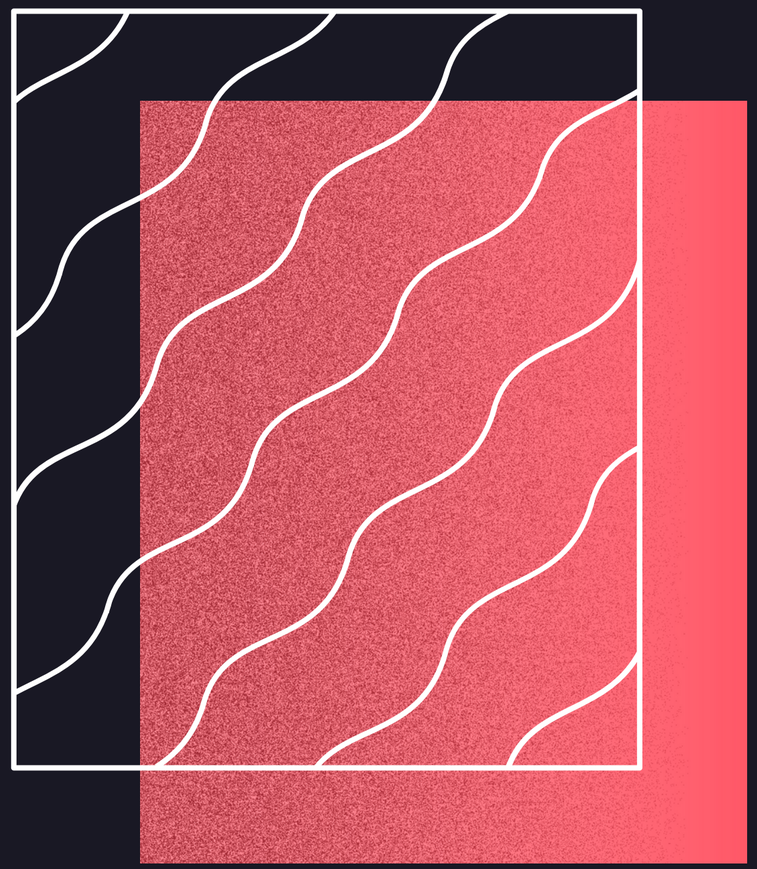
Control Net



Cartoon line drawing

"1girl, masterpiece, best quality, ultra-detailed, illustration"

Stable Diffusion



WHY IT IS IMPORTANT?

In image processing tasks, there are often specific conditions and user controls that need to be taken into account.

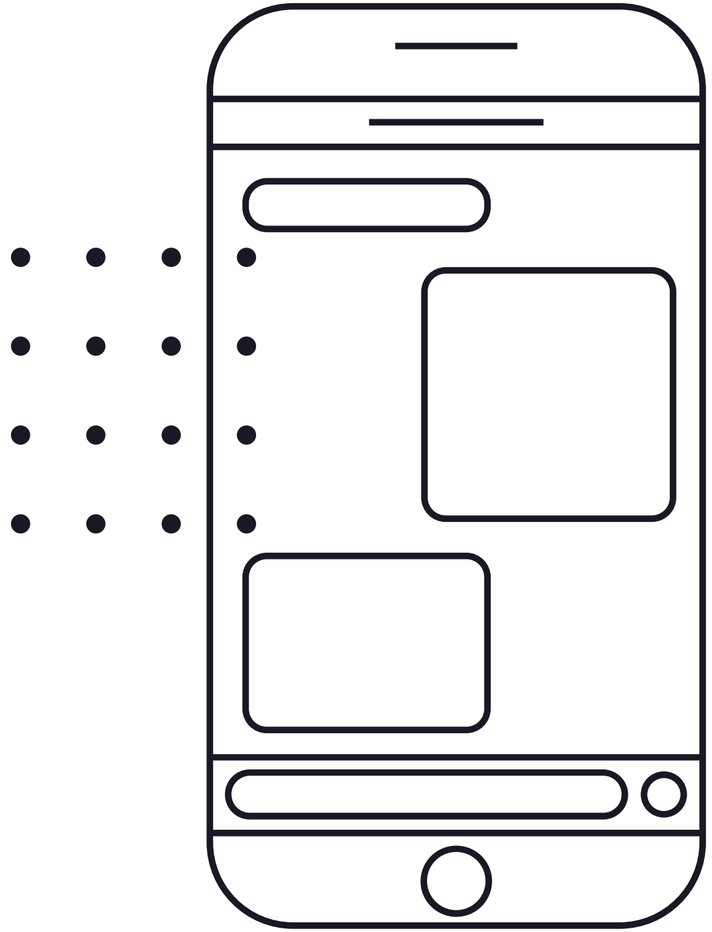
Pretrained large diffusion models may not be able to handle these specific conditions and controls

IMPACTS

ControlNet can increase the functionality and applicability of pretrained large diffusion models

ControlNet learns task-specific conditions even when the training dataset is small (<50k) makes it a promising solution for practical applications where data may be limited

It is up for 2 months and we can already see 15.6k stars on Github with various applications, including a WebUI



BACKGROUND

HyperNetwork

StyleGAN

Text-to-Image Diffusion

Using CLIP, such as Disco Diffusion

Image-to-Image Translation

Taming Transformer
Very similar, but it is domain focused and Control Net is task focused

Diffusion Probabilistic Model

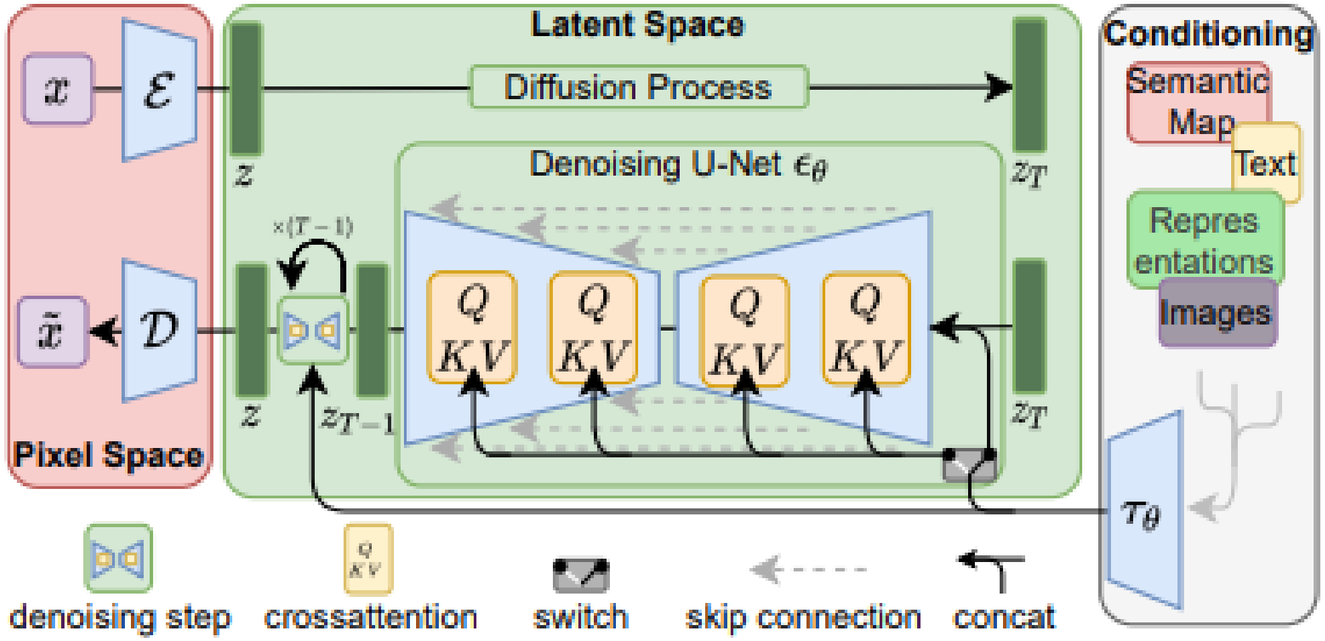
Stable Diffusion

Personalization of Diffusion Model

Textual Inversion

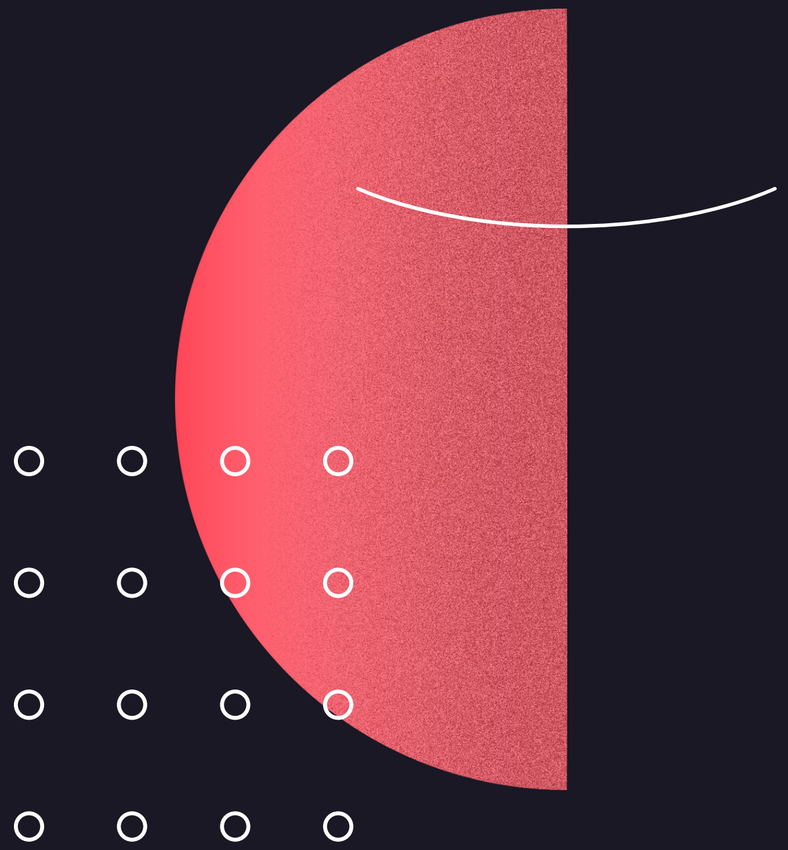
BACKGROUND
STABLE DIFFUSION

it turns text prompt to high quality images



08

The Model

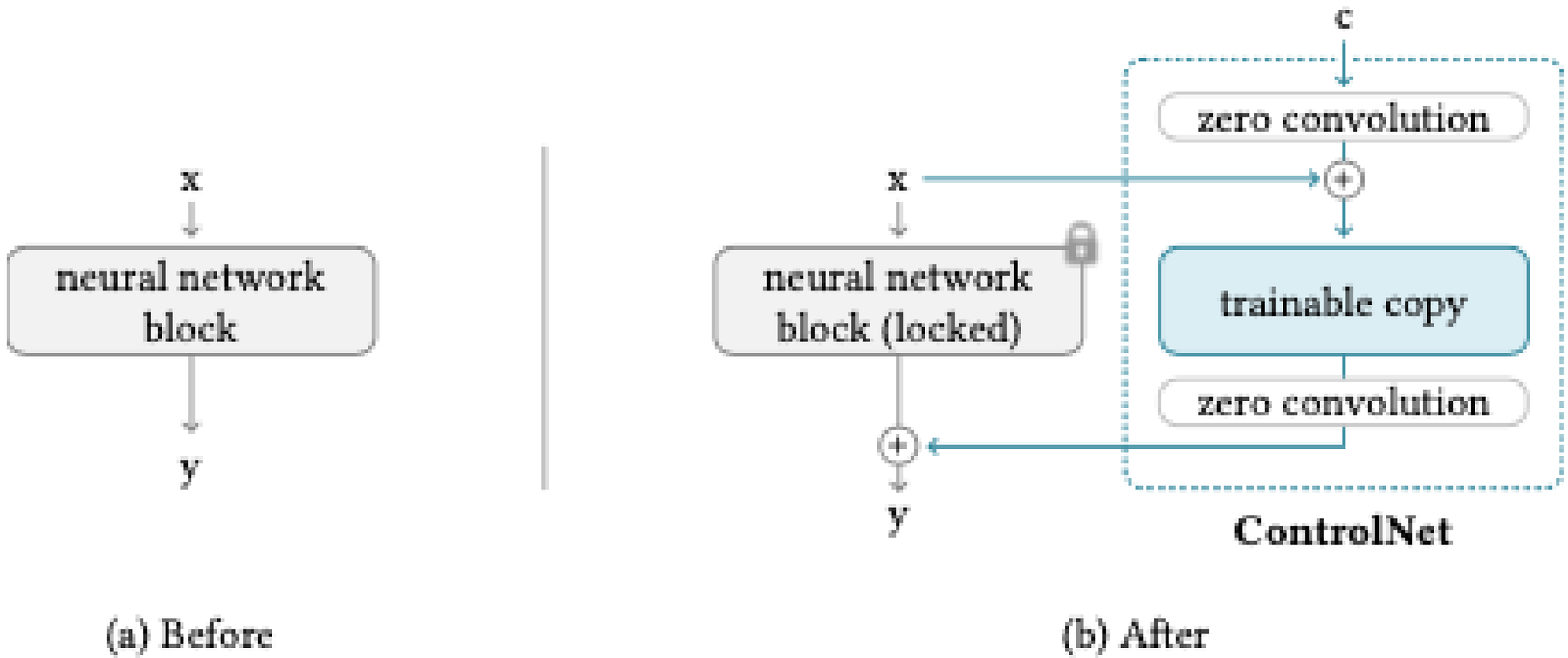


CONTROL NET ARCHITECTURE

Use a "trainable copy" for specific tasks

CONTROL NET ARCHITECTURE

A single layer



(a) Before

(b) After

In zero convolution, the convolution weights are initialized to zeros and then progressively grow to optimized parameters in a learned manner during training.

CONTROL NET ARCHITECTURE

Original feature map

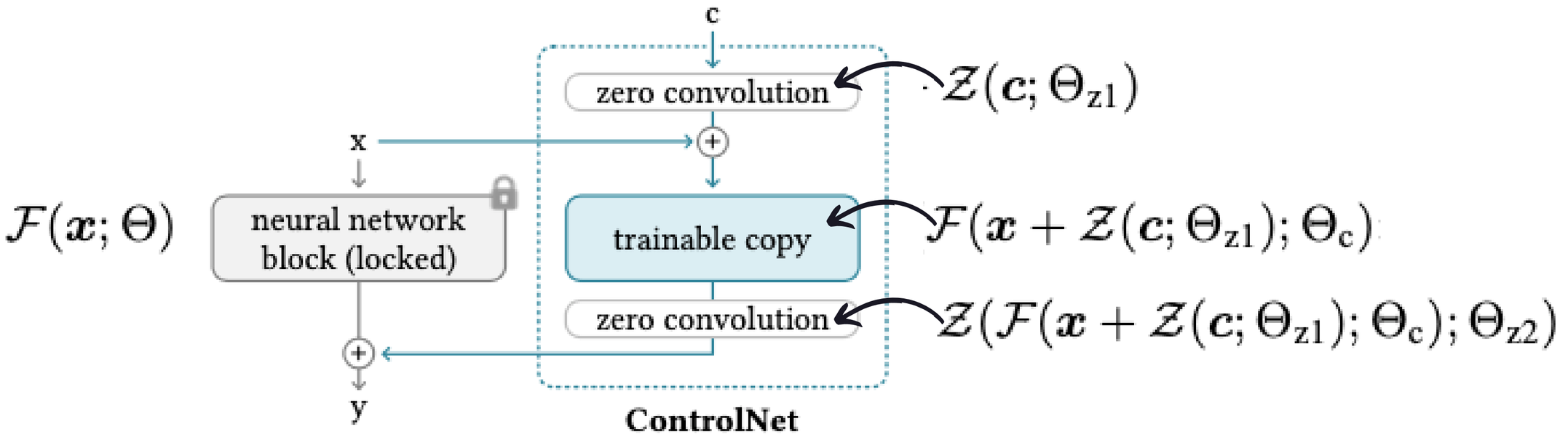
$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}; \Theta)$$

ControlNet feature map

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

CONTROL NET ARCHITECTURE

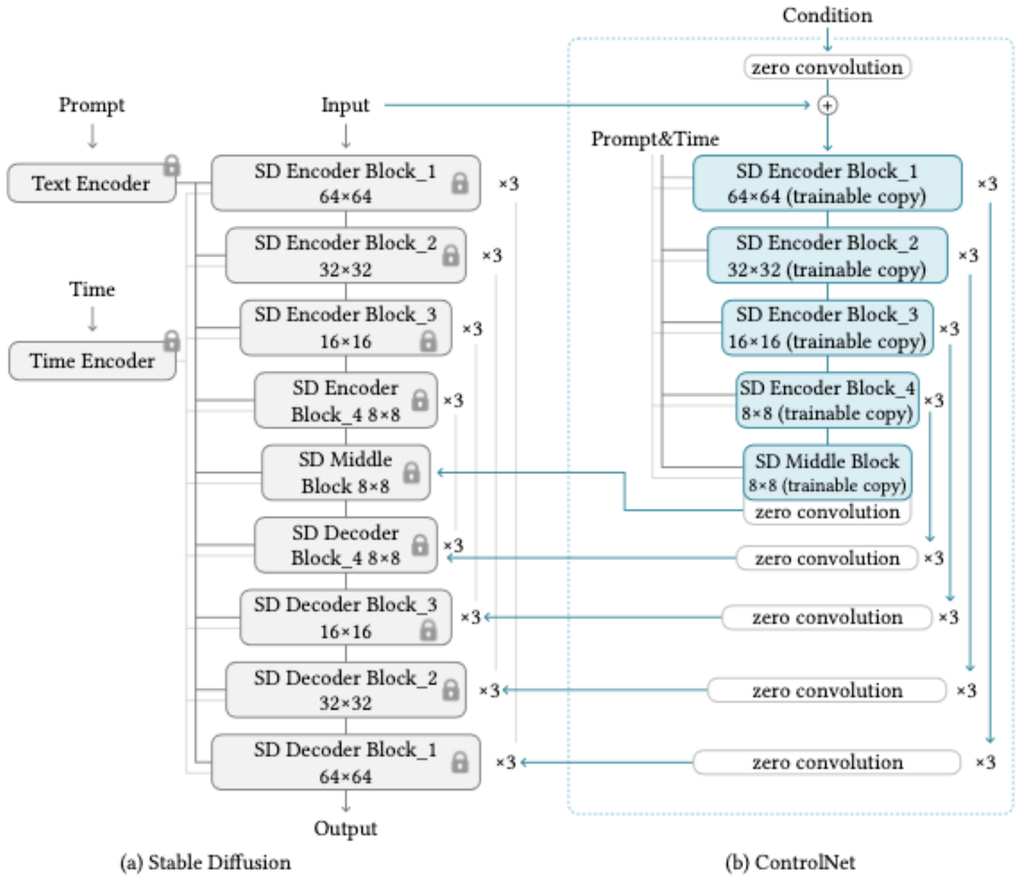
A single layer



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

ControlNet feature map

CONTROL NET ARCHITECTURE

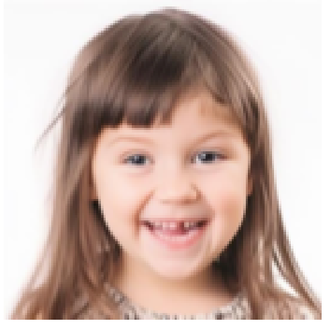


The whole net

The original weights are locked, no gradient computation on the original encoder is needed for training. This can speed up training and save GPU memory, as half of the gradient computation on the original model can be avoided.

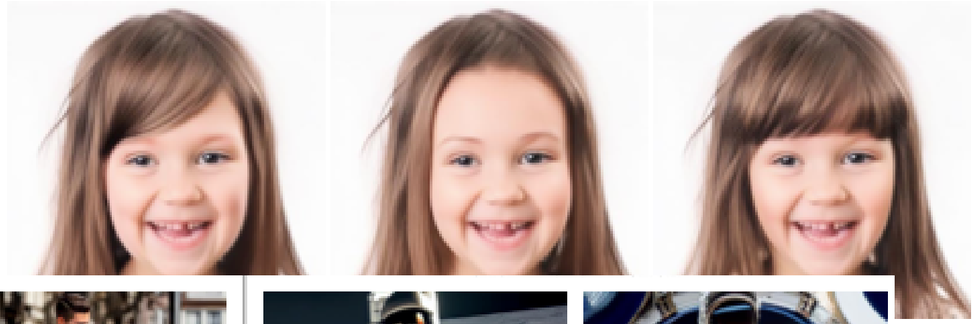
SOME RESULTS

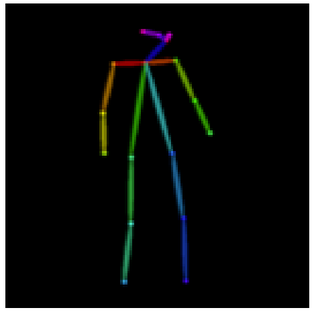
SOME RESULTS



Source image

User input





"astronaut"

REGIONAL PROMPT