AI VIETNAM
@aivietnam.edu.vn

Module 09 – Project Class

Text-Guided Image Generation Using Conditional Flow Matching

Code - Data

Nguyen Quoc Thai
MSc in Computer Science

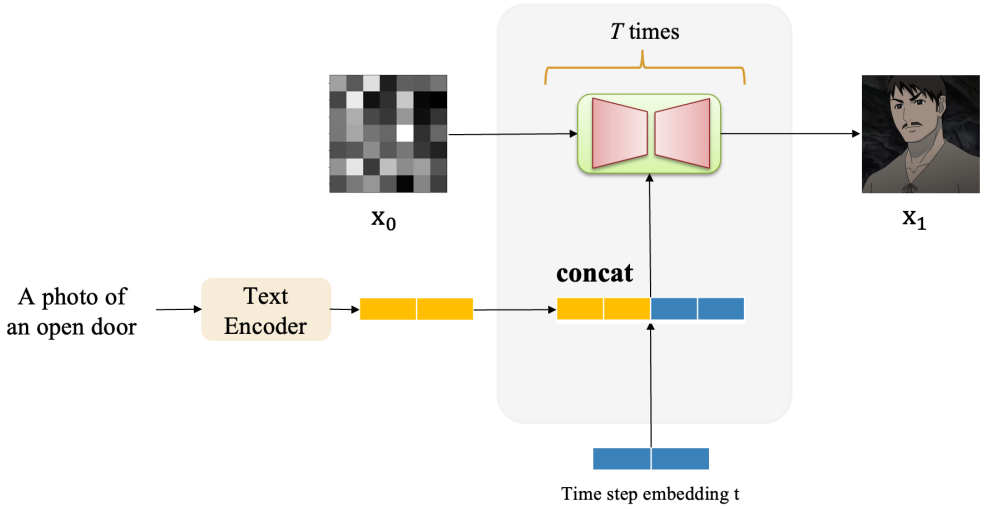
Objectives

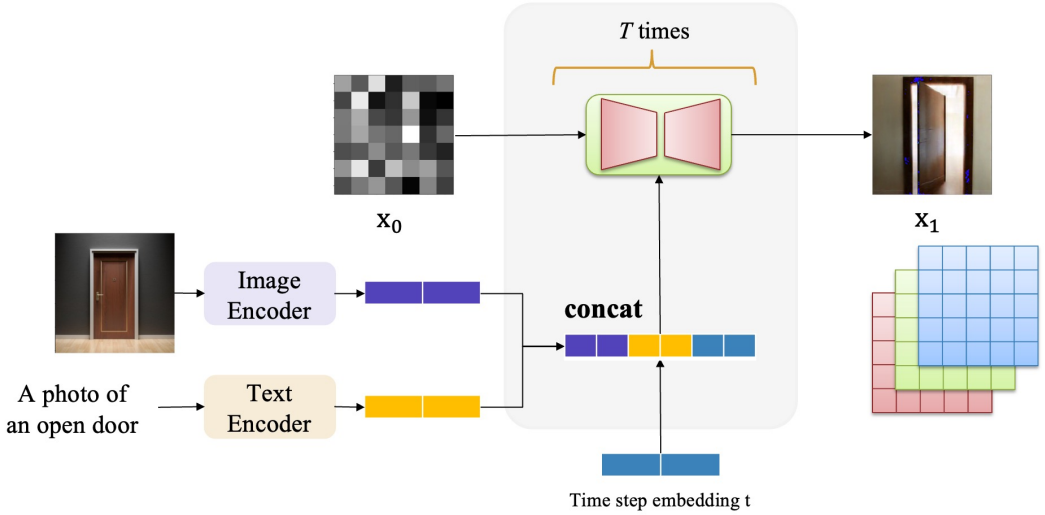
Conditional Flow Matching (Review)

- Flow Matching Model
- Training & Sampling
- Conditional Flow Matching

Text-Guided Image Generation

- Text-To-Image Generation
- Text-Guided Image Generation
- TedBench Dataset

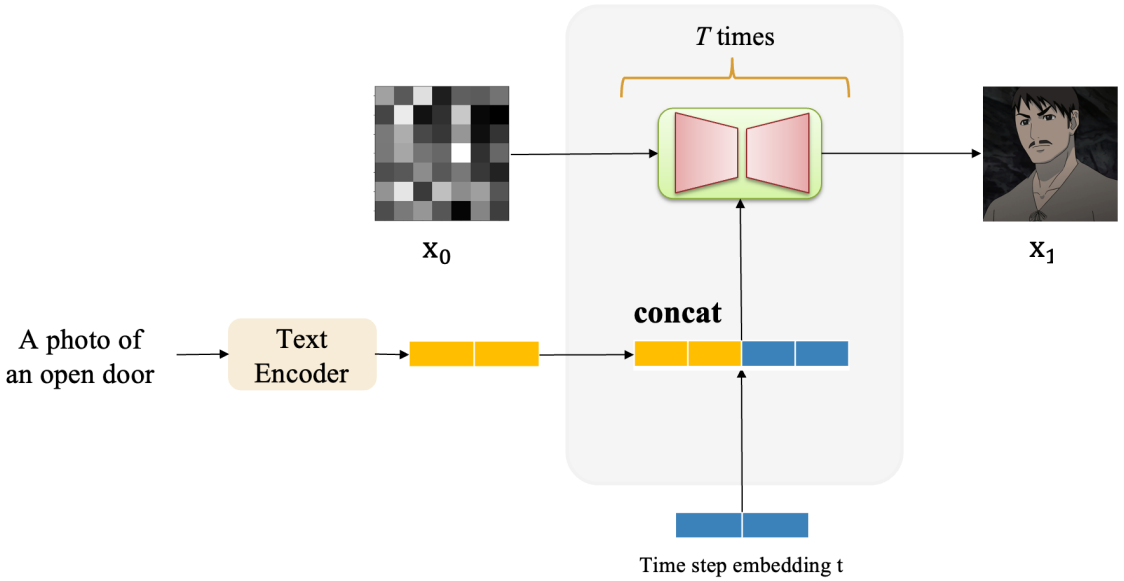




Outline

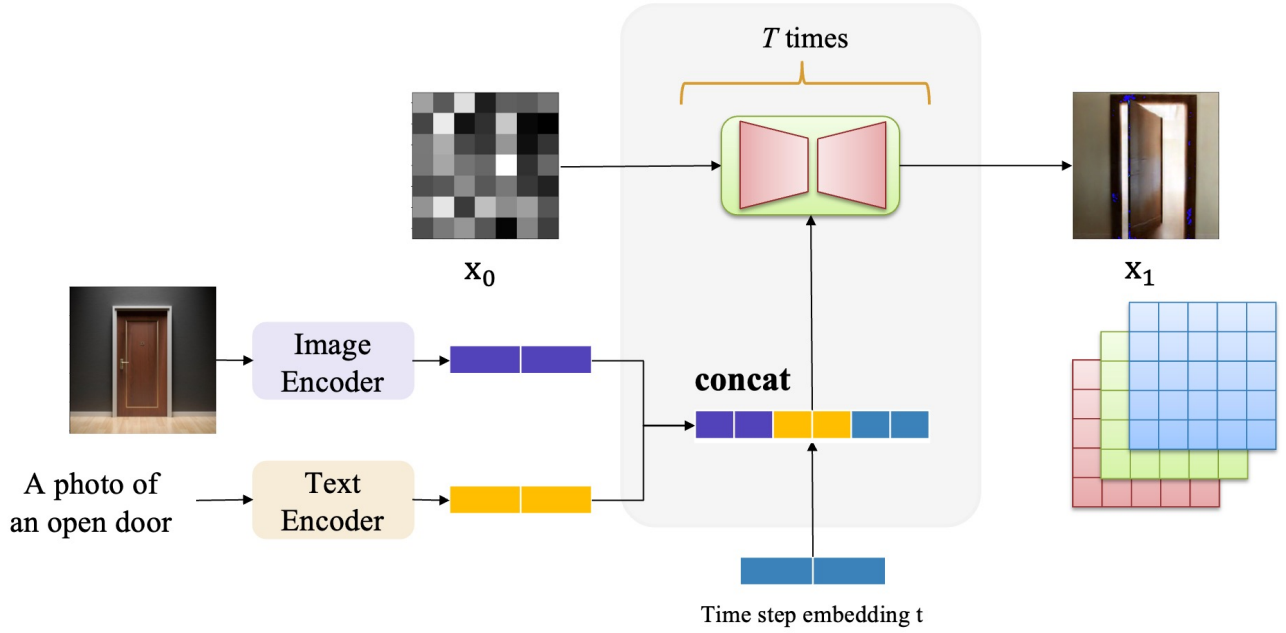
SECTION 1

Conditional Flow Matching



SECTION 2

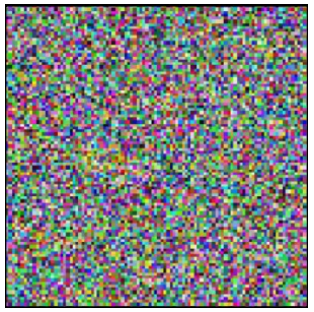
Text-Guided Image Generation



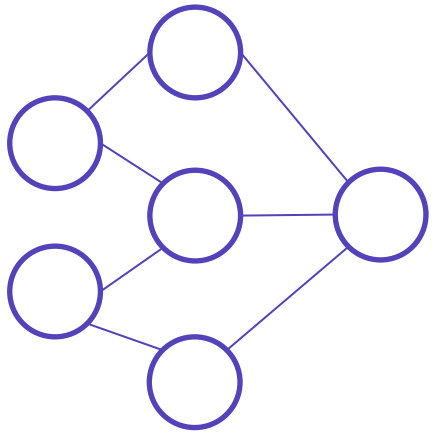
Conditional Flow Matching

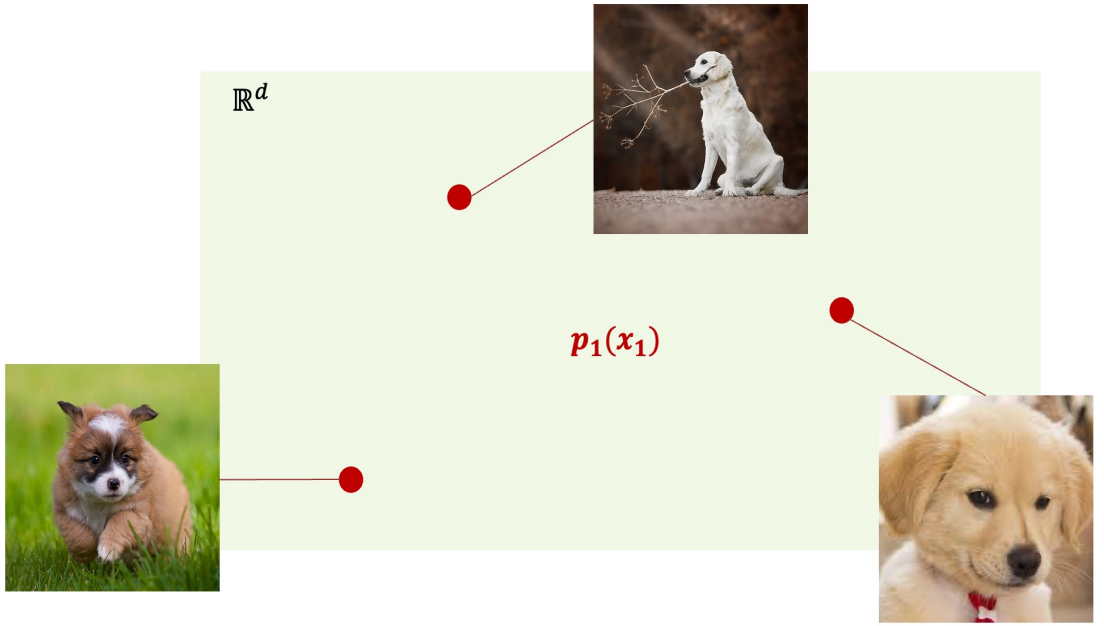
The Generative Modeling

- Generate new samples from an underlying distribution (the training data)



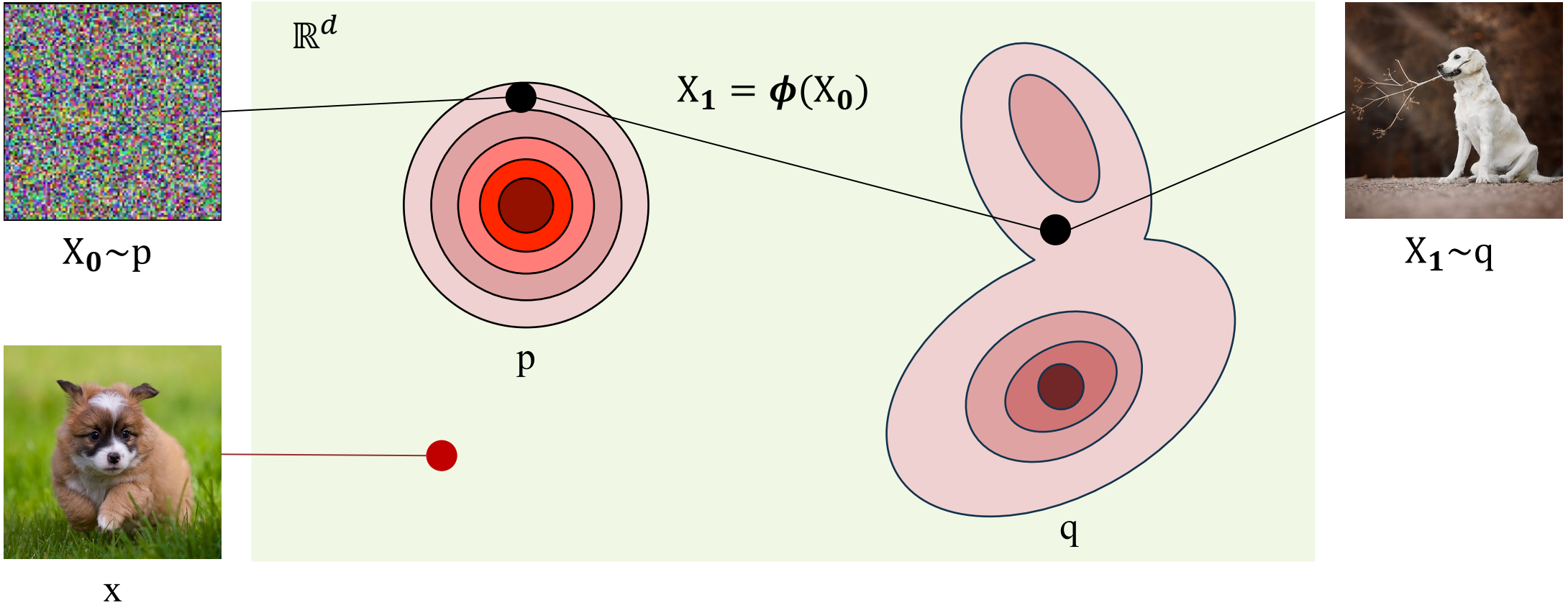
$p_0(x_0)$





Conditional Flow Matching

The Generative Modeling

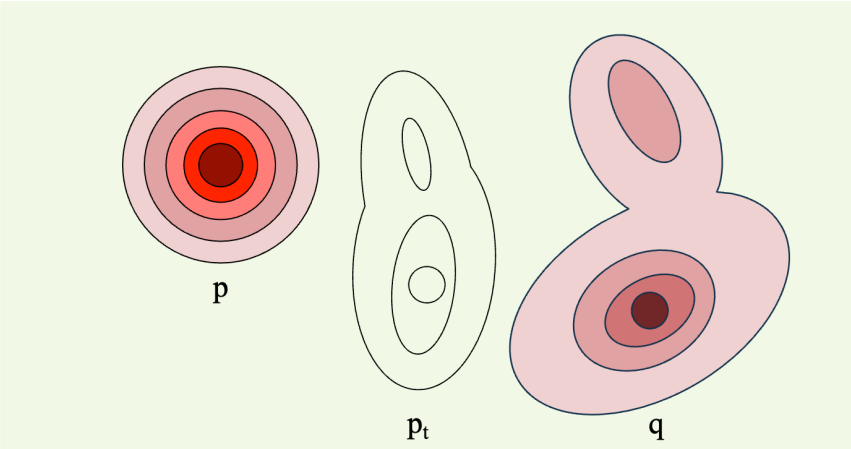


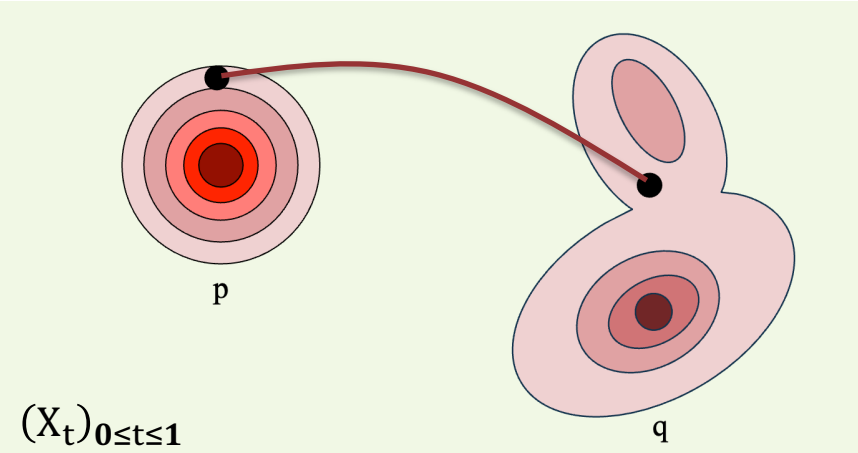
Conditional Flow Matching

Flow Matching

- A scalable method to train flow generative models
- Train by regressing a velocity, sample by following the velocity

$$(X_t)_{0 \le t \le 1}$$
$$X_{t+h} = \boldsymbol{\phi}_{t+h|t}(X_t)$$
$$X_t \sim p_t$$





Flow Matching

Conditional Flow Matching

Velocity in Flow Matching

- Velocity: the difference between the noisy and clean data, the rate of transformation between the two states:

$$\boldsymbol{v_t = x_1 - x_0}$$

Flow

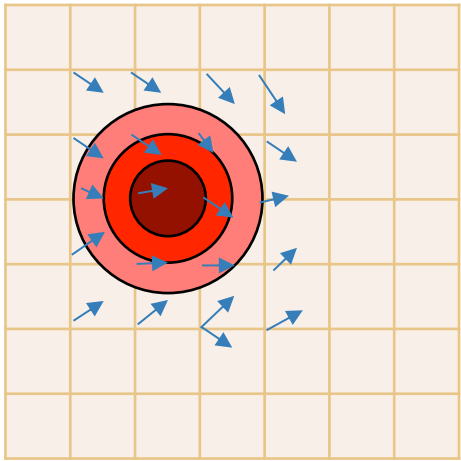
$\boldsymbol{\psi_t(x)}$

Sol ODE ↑ ↓ Differentiate

$\boldsymbol{u_t(x)}$

Velocity

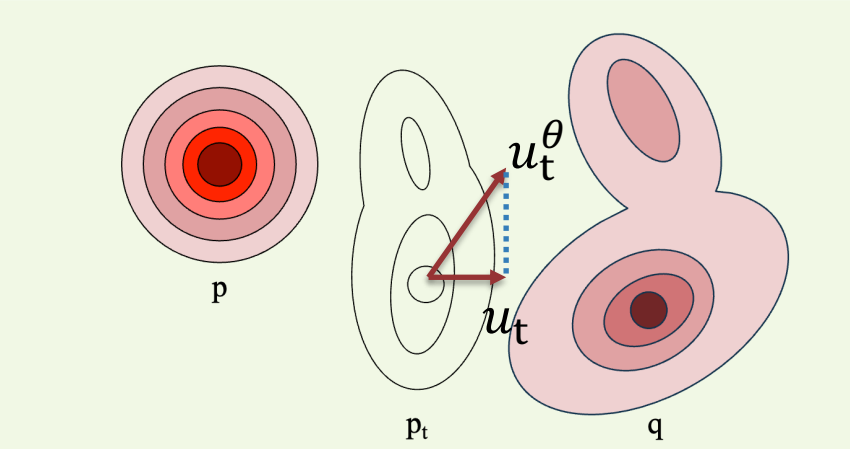
Velocity u_t generates p_t if $X_t = \psi_t(X_0) \sim p_t$



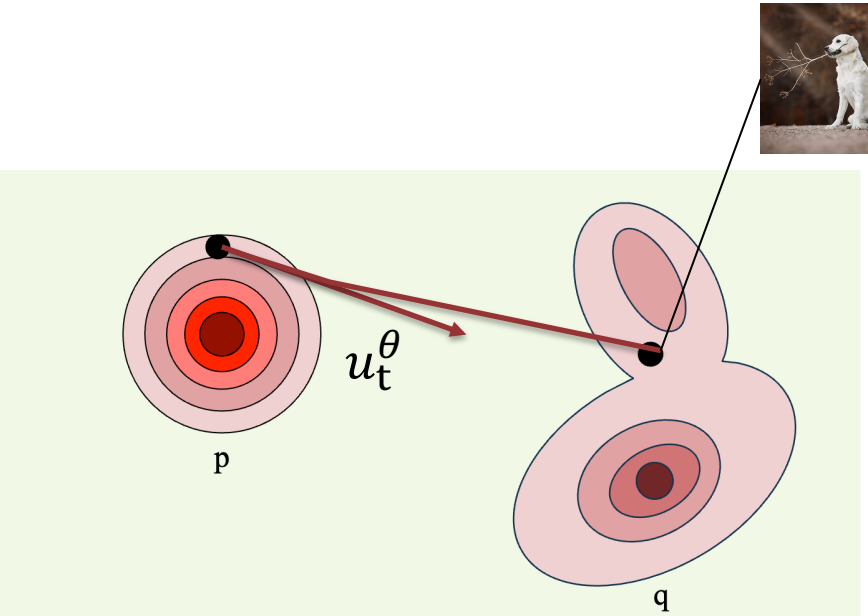
Conditional Flow Matching

Flow Matching

Train a velocity generating p_t with $p_0 = p$ and $p_1 = q$



Sample from $X_0 \sim p$

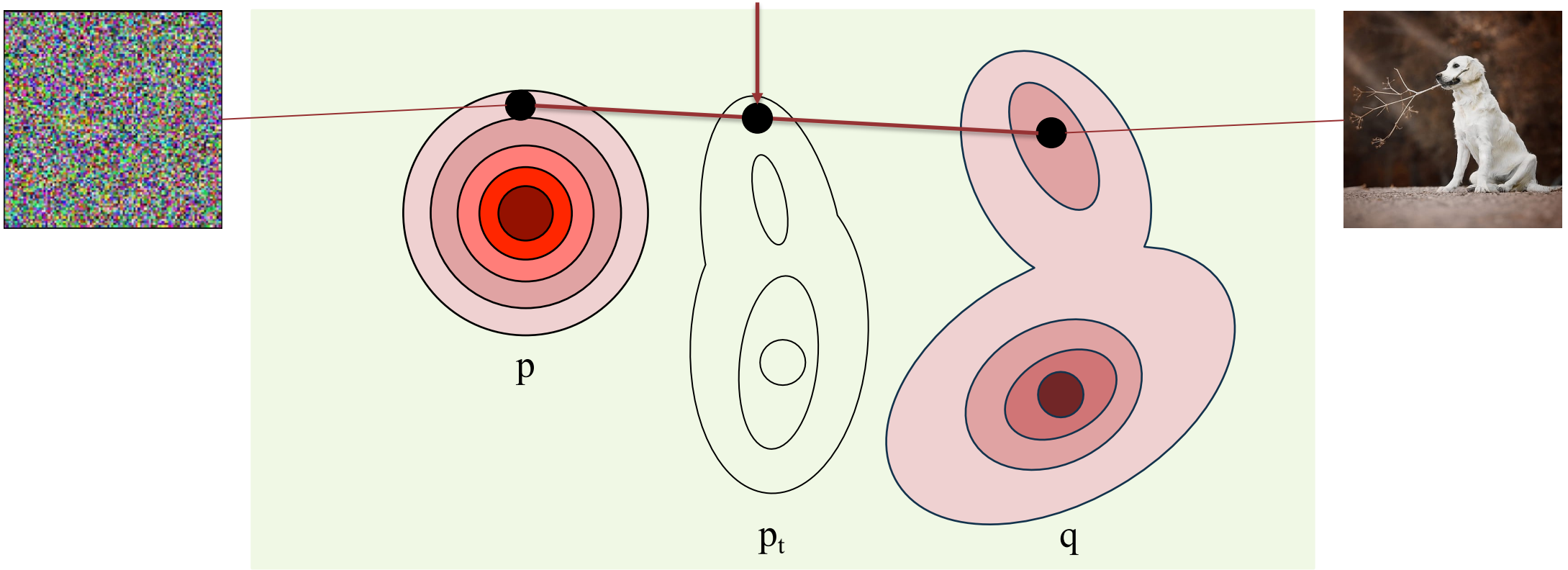


Conditional Flow Matching

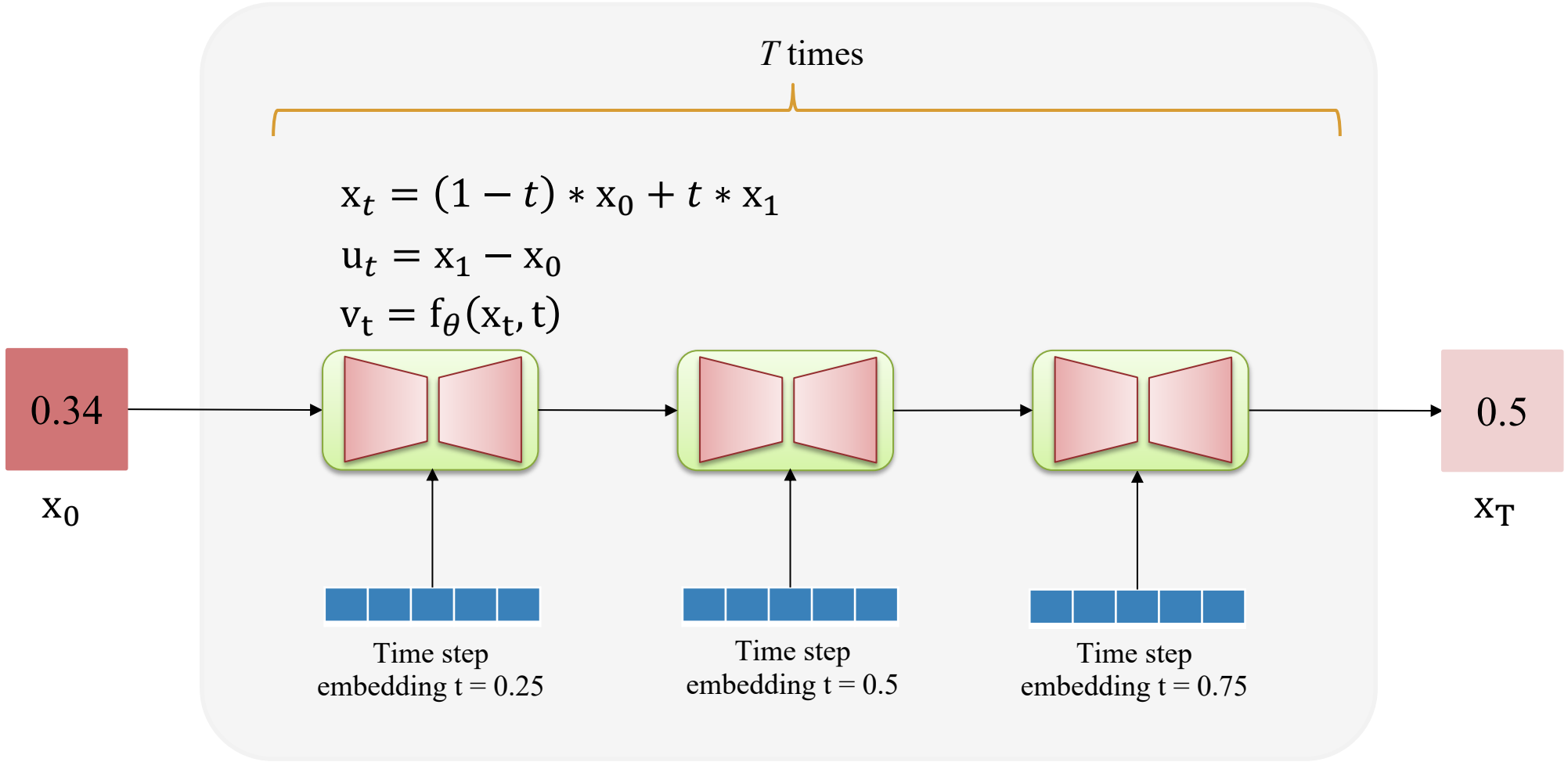
Training in Flow Matching

$$E_{t,X_0,X_1}\left\|u_t^{\theta}(X_t) - (X_1 - X_0)\right\|$$

$$X_t = (1-t) * X_0 + t * X_1$$



Conditional Flow Matching



Conditional Flow Matching

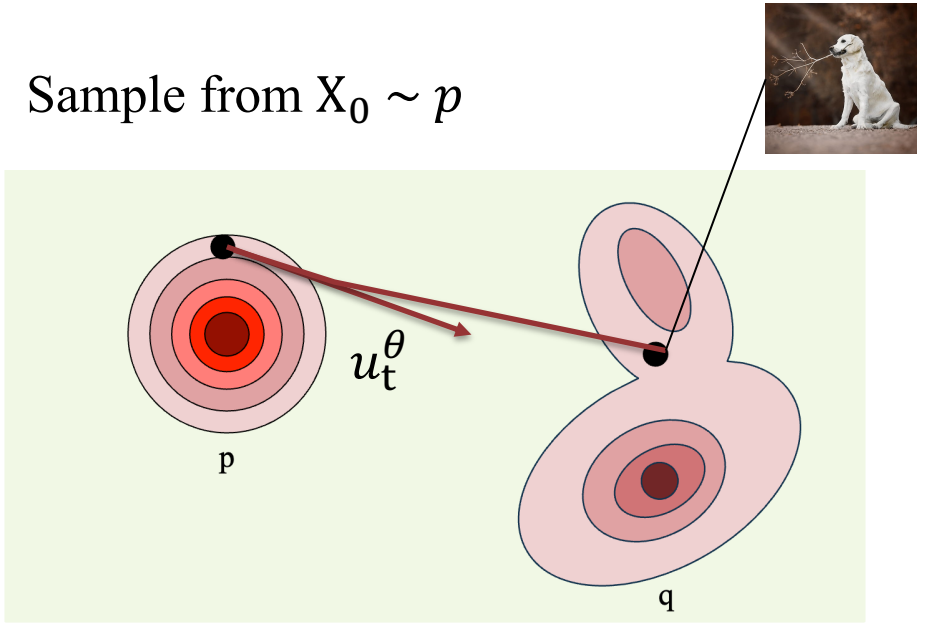
Sampling in Flow Matching

- Euler's method for solving ordinary differential equations (ODEs)

$$\frac{dy}{dt} = f_\theta(y, t)$$

$$y(t + \Delta t) \approx y(t) + \left(\frac{dy(t)}{dt}\right)\Delta t$$

$$y(t + \Delta t) \approx y(t) + f_\theta(y, t)\Delta t$$

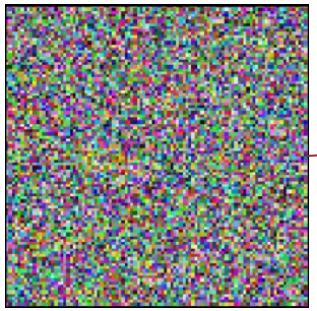


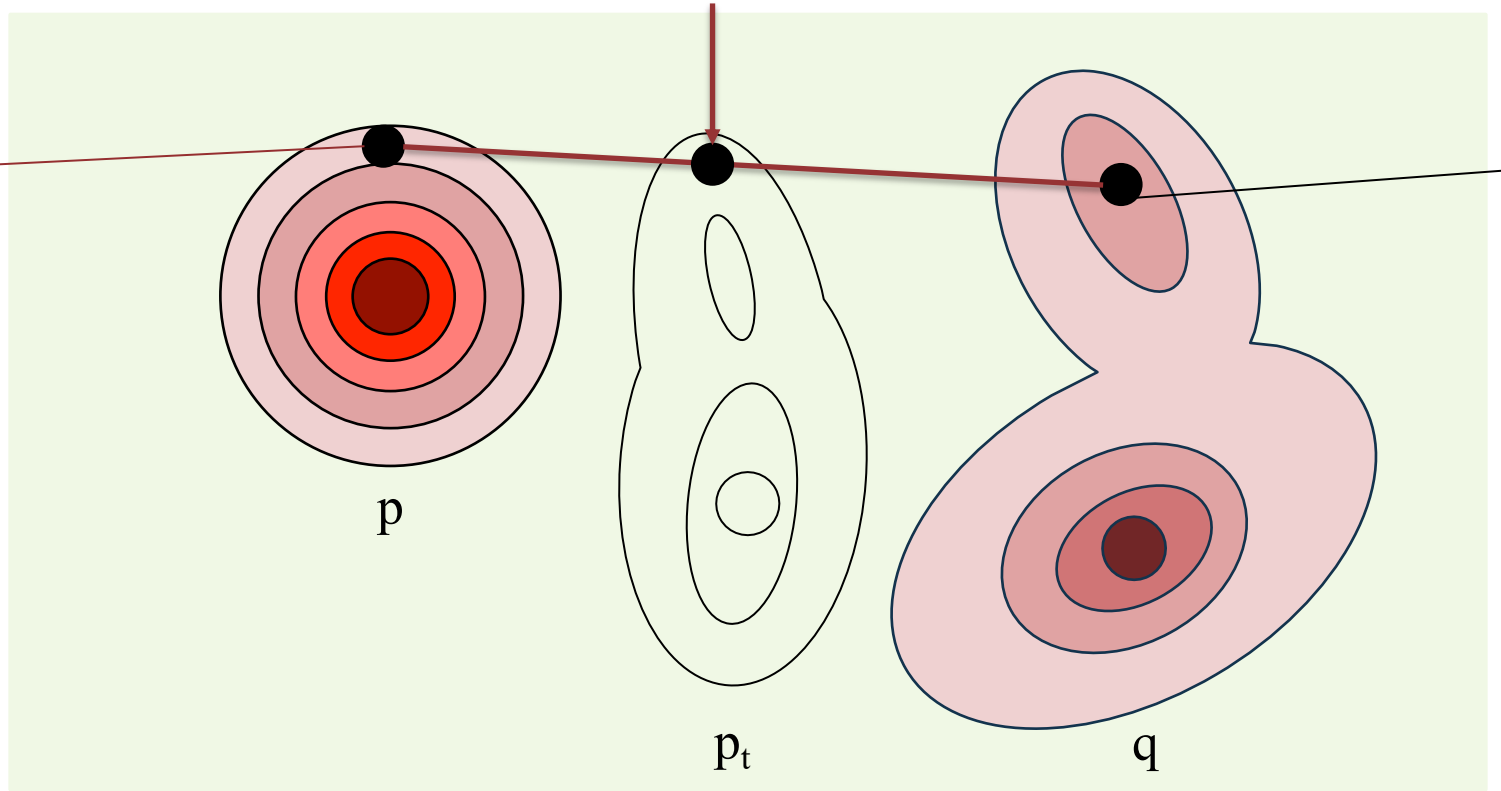
Conditional Flow Matching

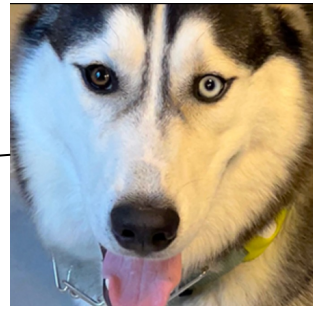
Training in Conditional Flow Matching

$$E_{t,X_0,X_1}\left\|u_t^{\theta}(X_t) - (X_1 - X_0)\right\| \longrightarrow E_{t,X_0,X_1,Y}\left\|u_t^{\theta}(X_t, Y) - (X_1 - X_0)\right\|$$

$$X_t = (1-t) * X_0 + t * X_1$$



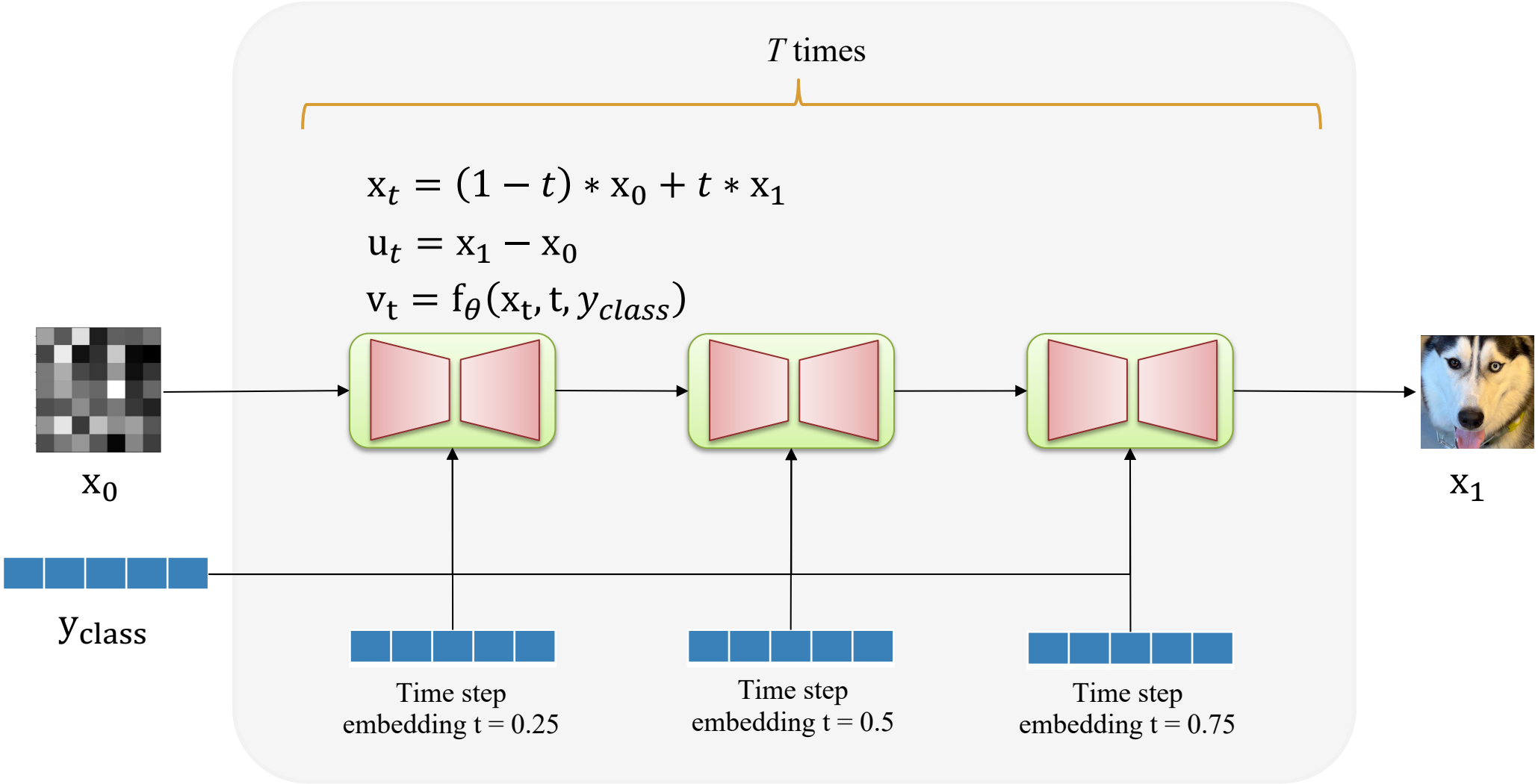




$(X_1, Y) \sim q$

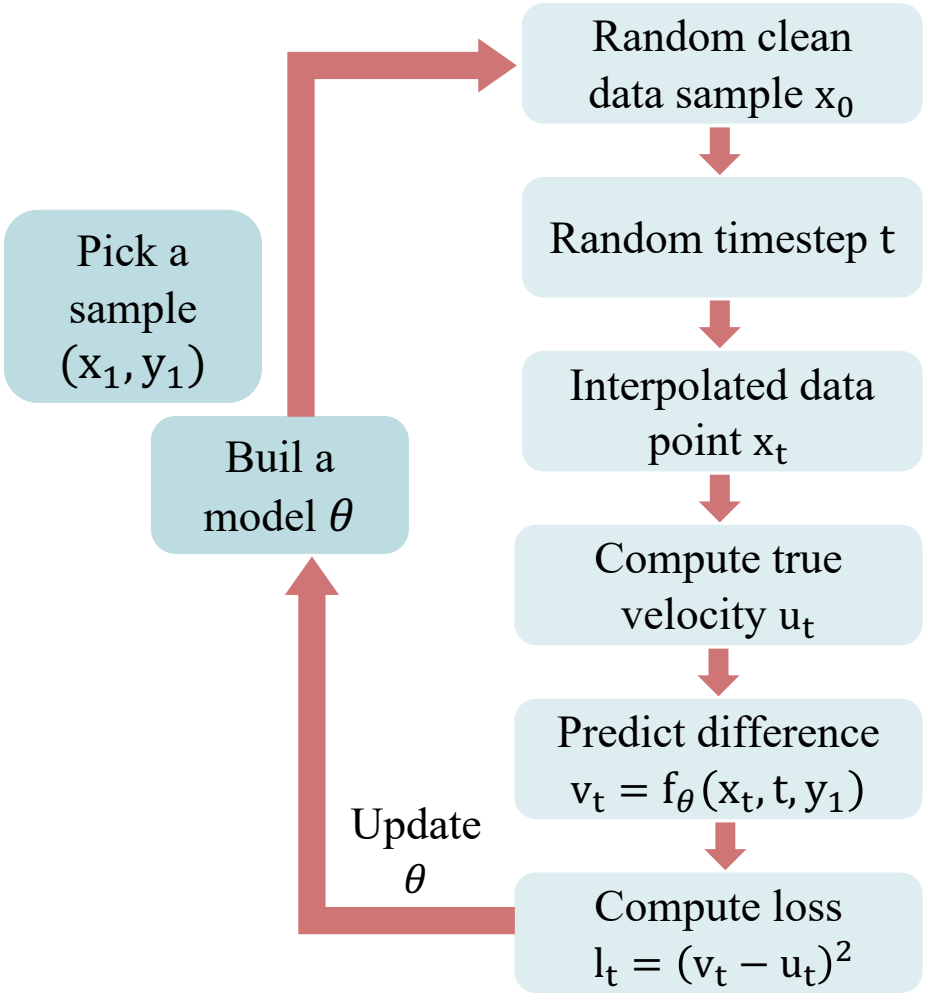
Label = "Husky"

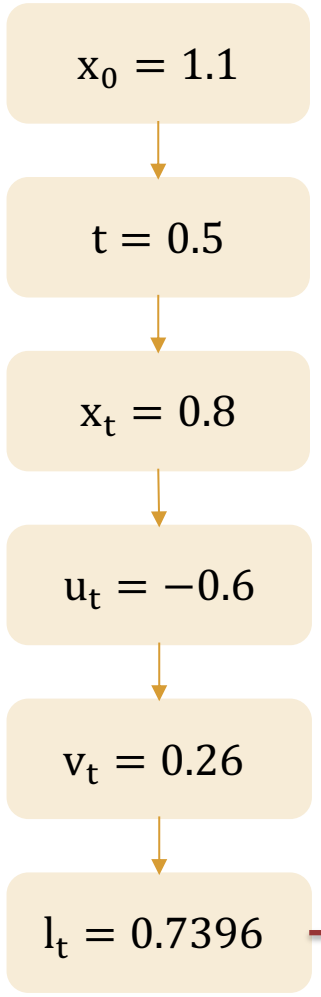
Conditional Flow Matching



Conditional Flow Matching

Training in Conditional Flow Matching

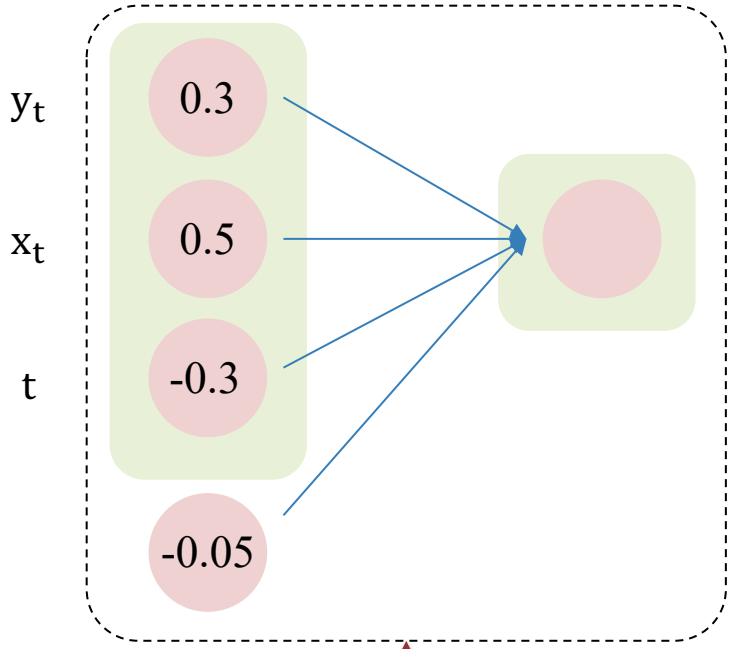




5 classes $y_1 = \frac{1}{5} = 0.2$ $x_1 = 0.5$

$$x_t = (1 - t) * x_0 + t * x_1$$

$$u_t = x_1 - x_0$$



Conditional Flow Matching

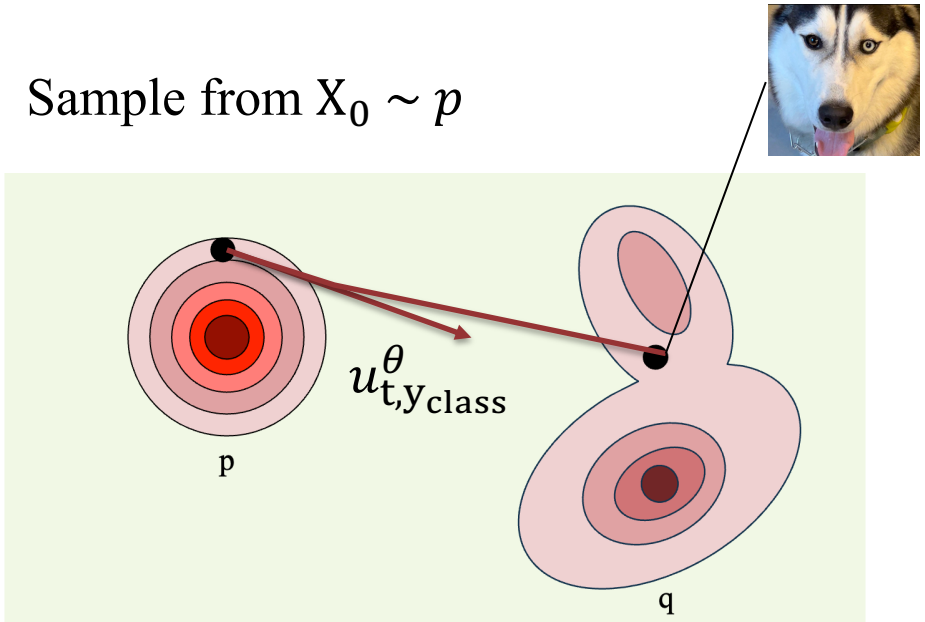
Sampling in Conditional Flow Matching

- Euler's method for solving ordinary differential equations (ODEs)

$$\frac{dy}{dt} = f_\theta(y, t, y_{class})$$

$$y(t + \Delta t) \approx y(t) + \left(\frac{dy(t)}{dt}\right)\Delta t$$

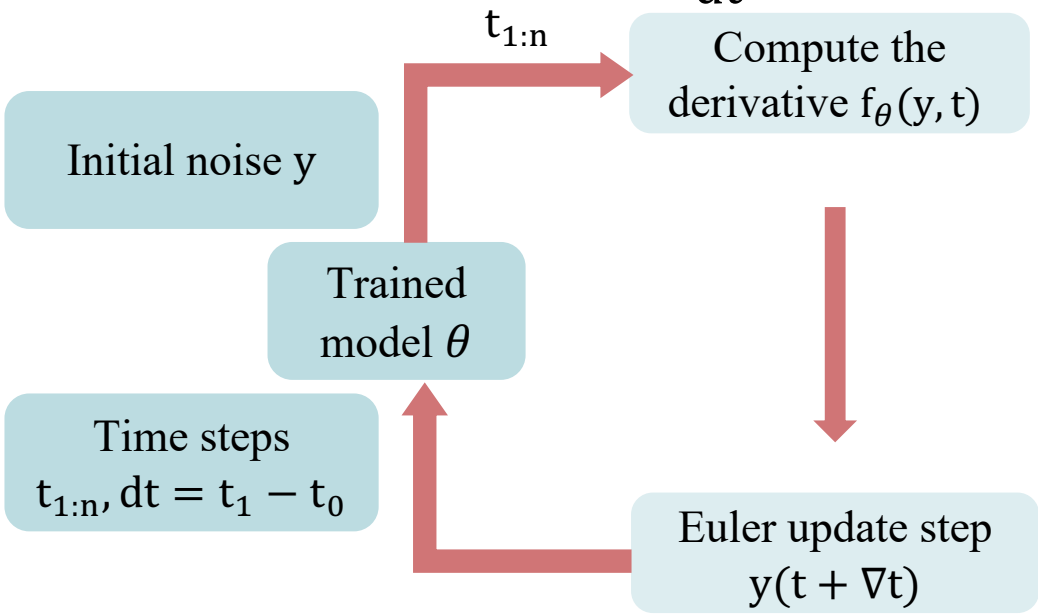
$$y(t + \Delta t) \approx y(t) + f_\theta(y, t, y_{class})\Delta t$$



Conditional Flow Matching

Sampling

$$\frac{dy}{dt} = f_\theta(y, t)$$



$$y(t + \Delta t) \approx y(t) + f_\theta(y, t)\Delta t$$

$t = 0.5$

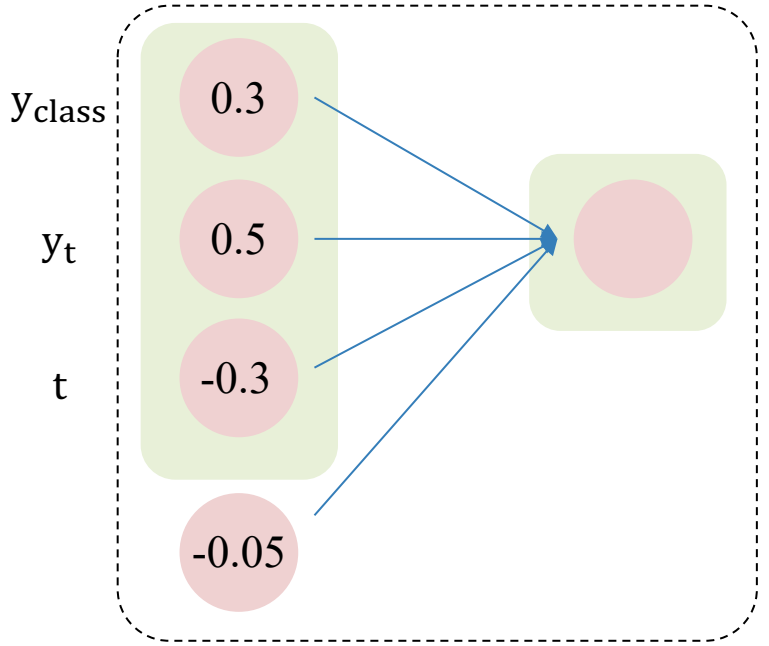
$f_\theta(y, t, y_{class}) = 0.28$

$y_{t+\Delta t} = 0.64$

$t = \{0.0, 0.5, 1.0\}$

$y = 0.5\ ;\ y_{class} = \frac{3}{5}$

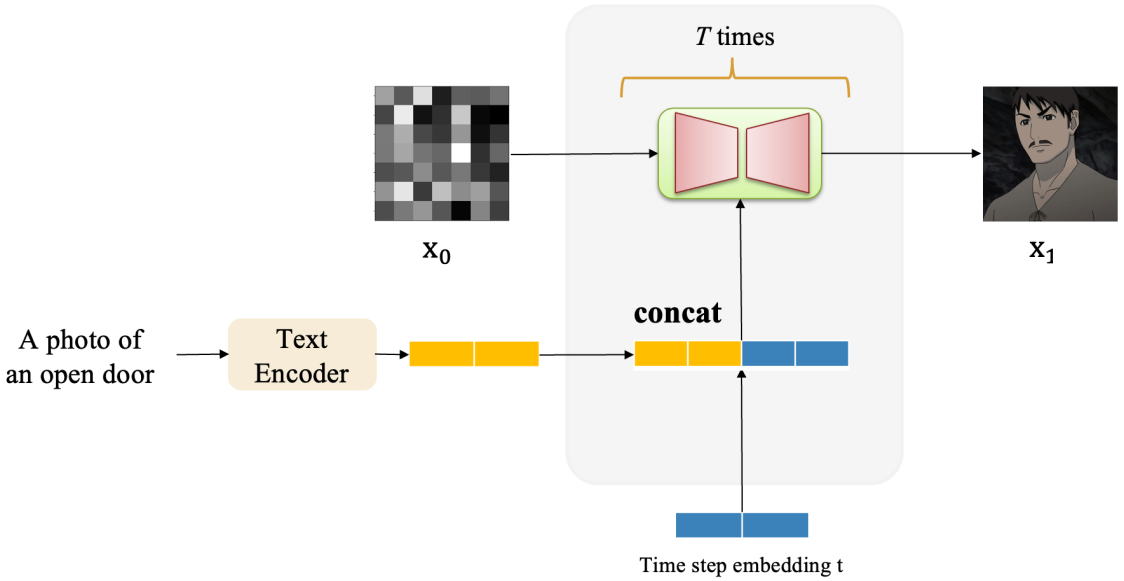
5 classes



Outline

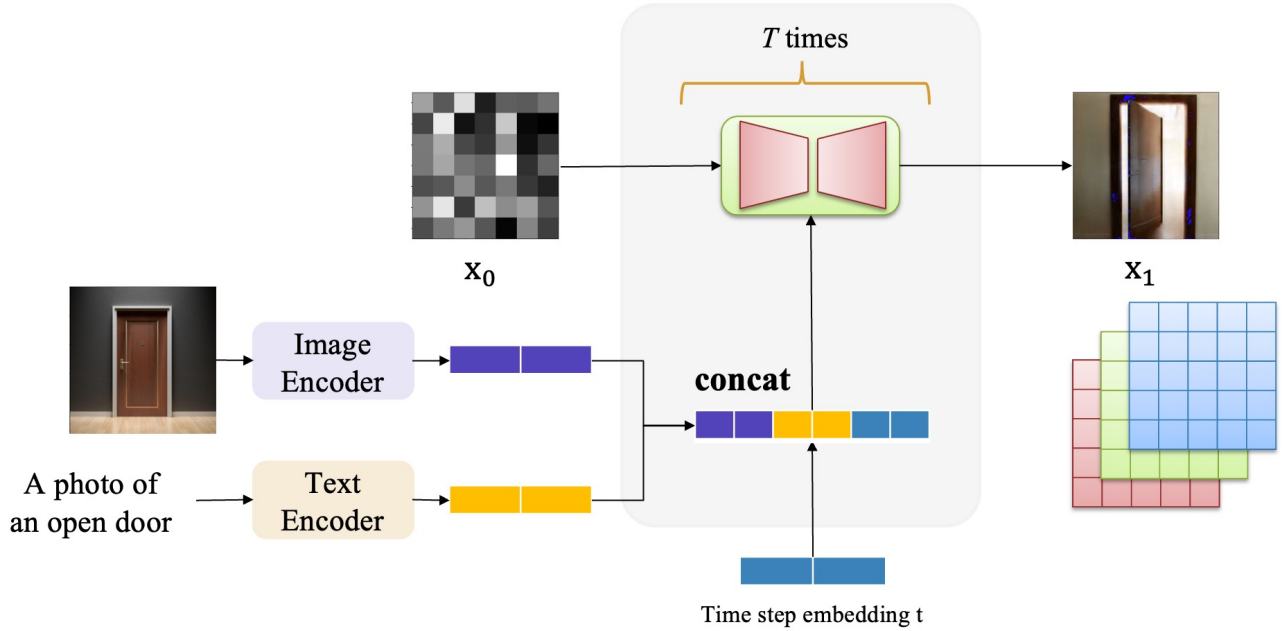
SECTION 1

Conditional Flow Matching



SECTION 2

Text-Guided Image Generation



Text-to-Image Generation

Description

- **Text-to-Image Generation** is a process that combines text embeddings with image generation techniques to create images from text description
- Datasets: Oxford-102 Flower, CUB-200, Naruto-Captions,…

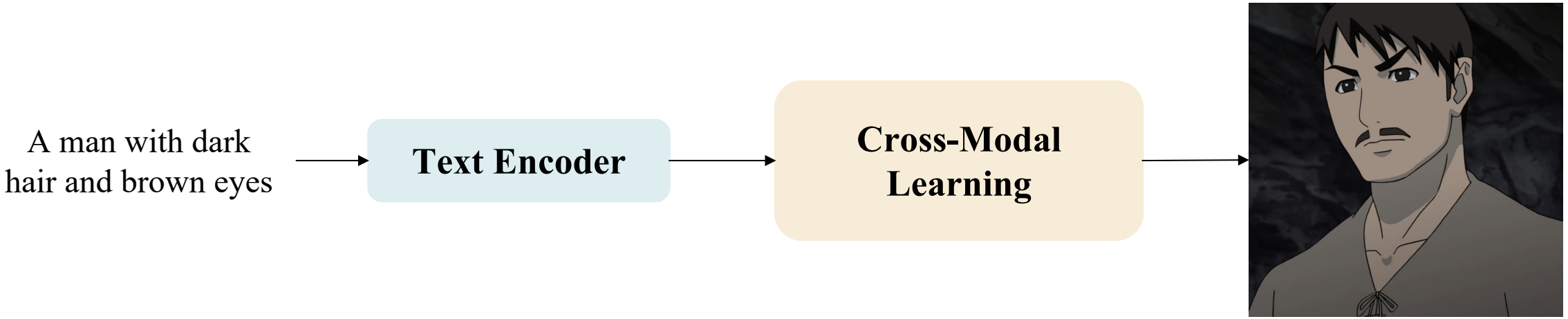
This pink and yellow flower has a beautiful yellow center with many stamens

A man with dark hair and brown eyes

Text-to-Image Generation

Methods

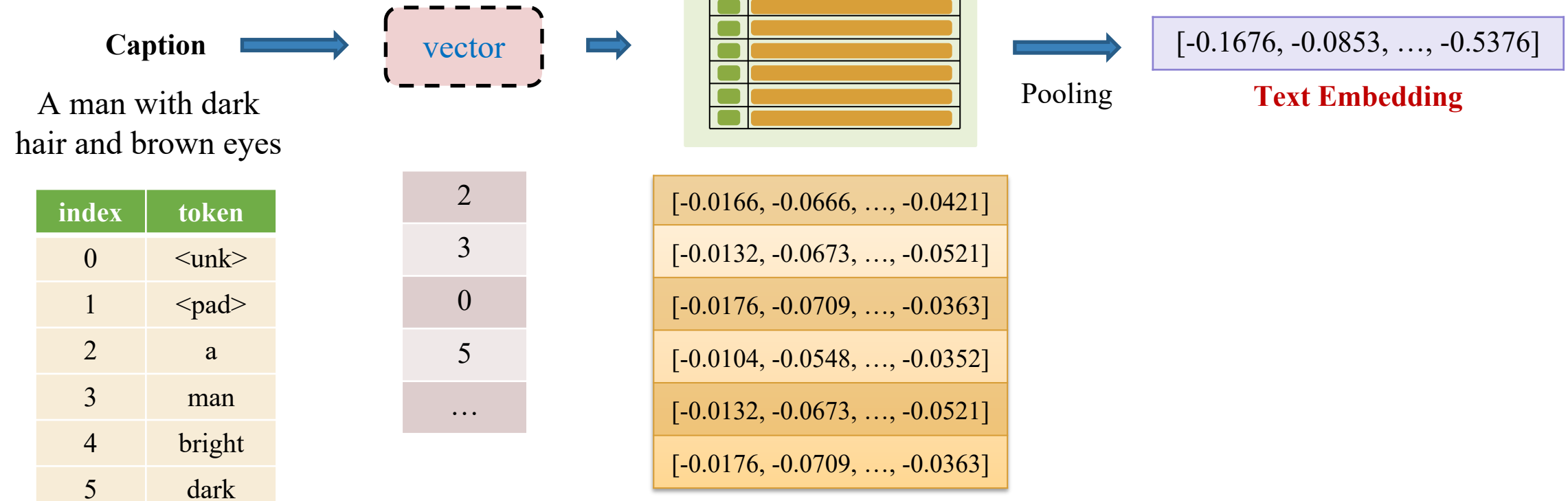
- **Text-to-Image Generation** is a process that combines text embeddings with image generation techniques to create images from text description
- Datasets: Oxford-102 Flower, CUB-200, Naruto-Captions,…



Text-to-Image Generation

Preprocessing

Text Representation (Text Encoder)

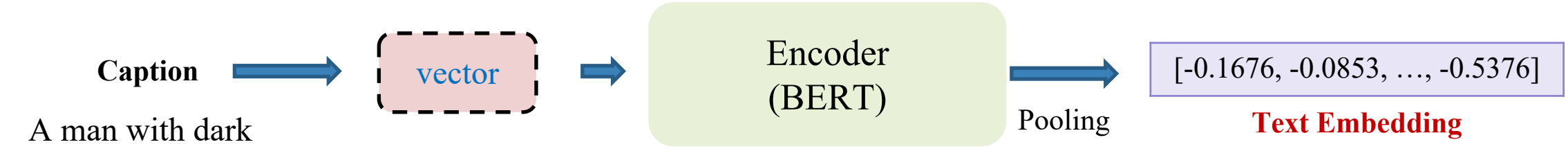


index	token
0	<unk>
1	<pad>
2	a
3	man
4	bright
5	dark
…	…

Text-to-Image Generation

Preprocessing

- **Text Representation (Text Encoder):** Pre-trained LMs



A man with dark hair and brown eyes

index	token
0	<unk>
1	<pad>
2	a
3	man
4	bright
5	dark
…	…

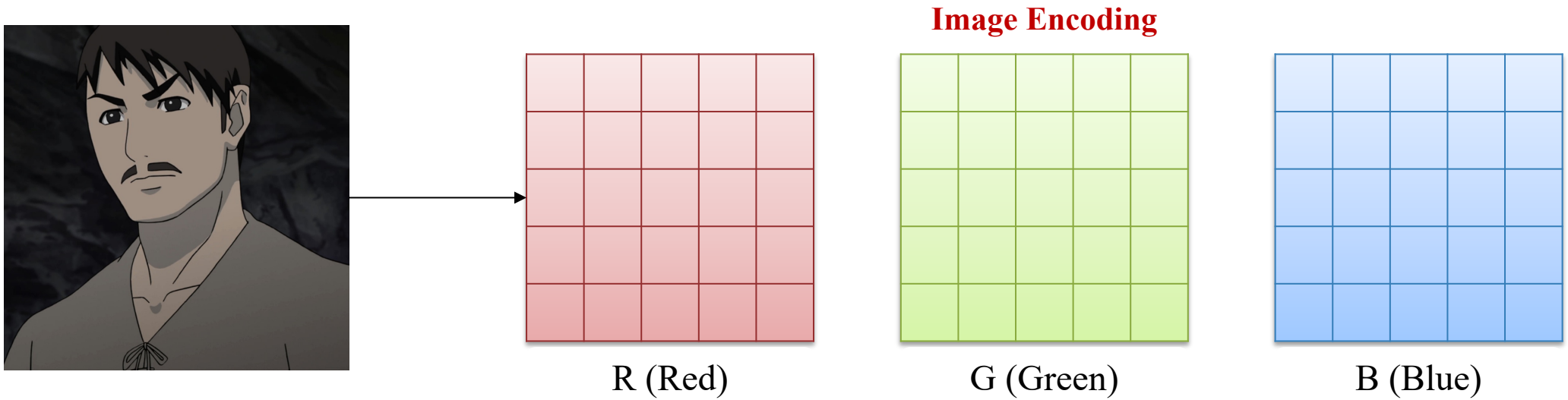
2
3
0
5
…

[-0.0166, -0.0666, …, -0.0421]
[-0.0132, -0.0673, …, -0.0521]
[-0.0176, -0.0709, …, -0.0363]
[-0.0104, -0.0548, …, -0.0352]
[-0.0132, -0.0673, …, -0.0521]
[-0.0176, -0.0709, …, -0.0363]

Text-to-Image Generation

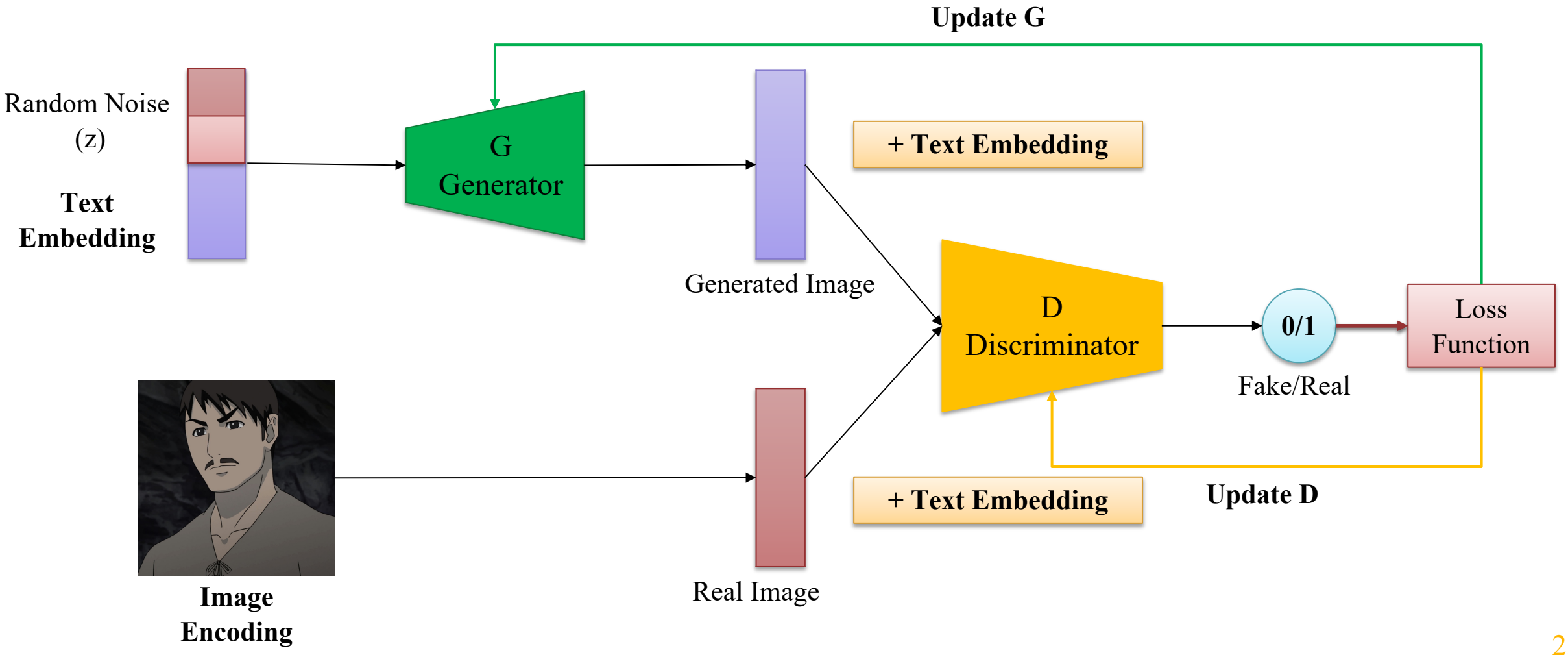
Preprocessing

- **Image Representation:** The generated image is output as a tensor or array representing pixel values, typically of shape (channels, height, width) for neural network processing



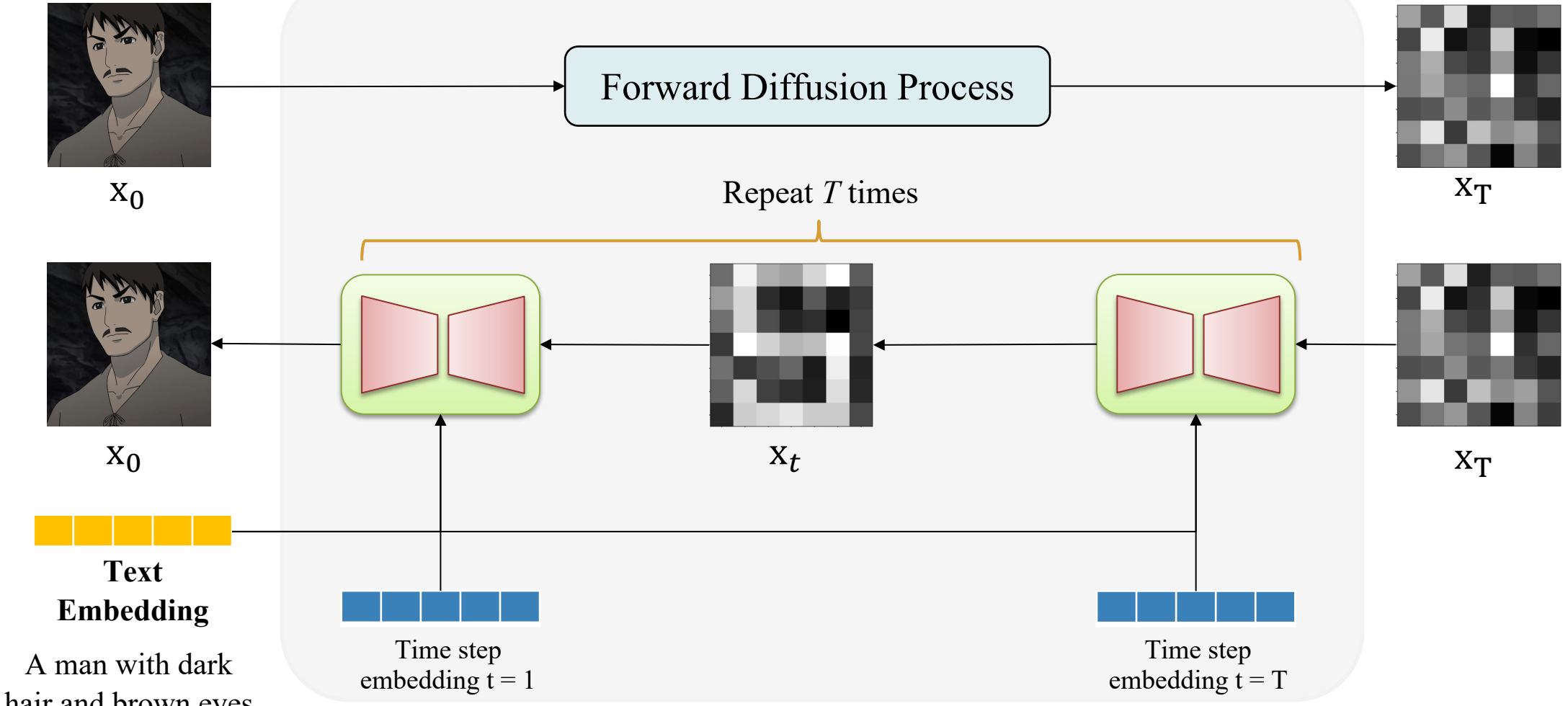
Text-to-Image Generation

DCGANs



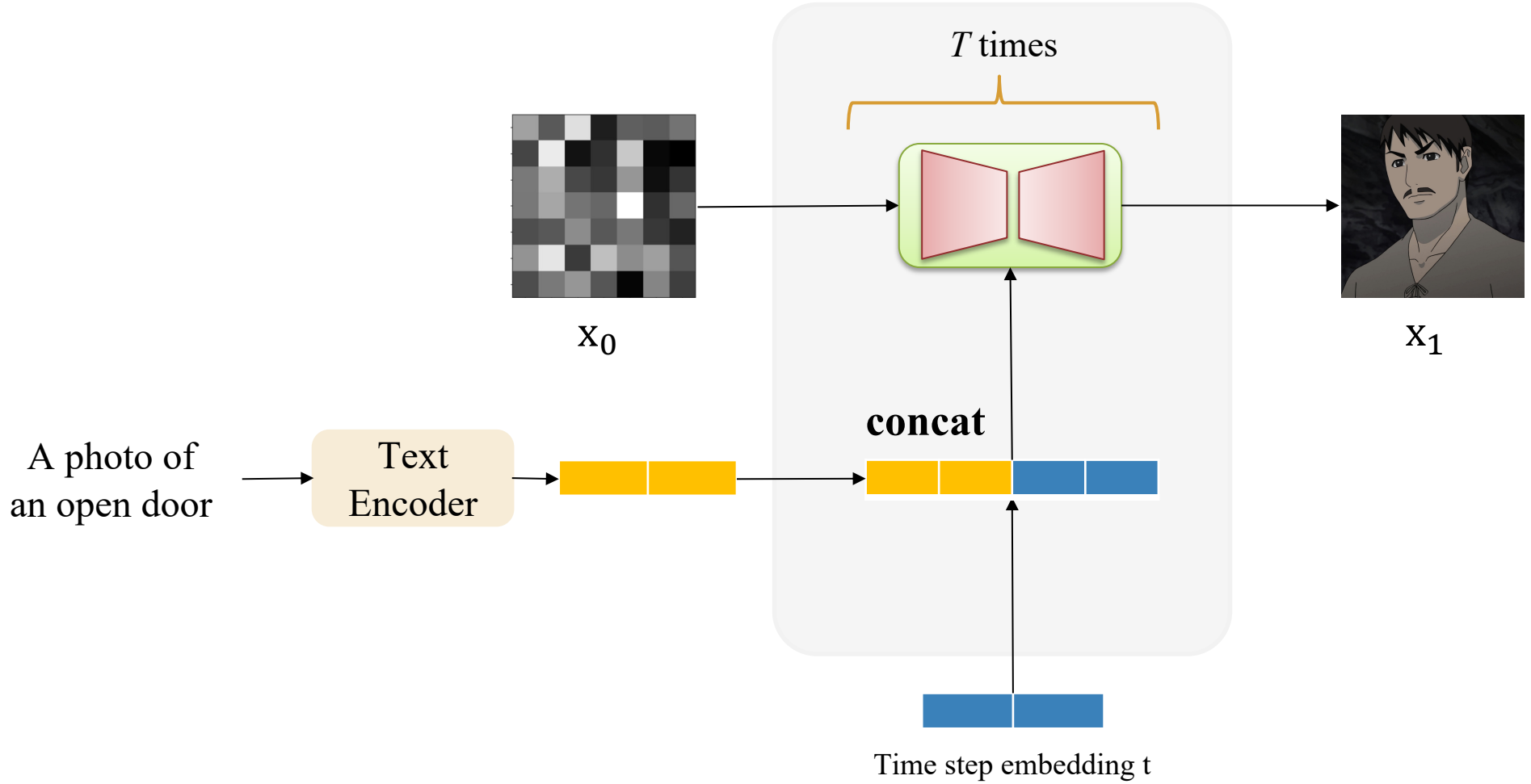
Text-to-Image Generation

Diffusion Models



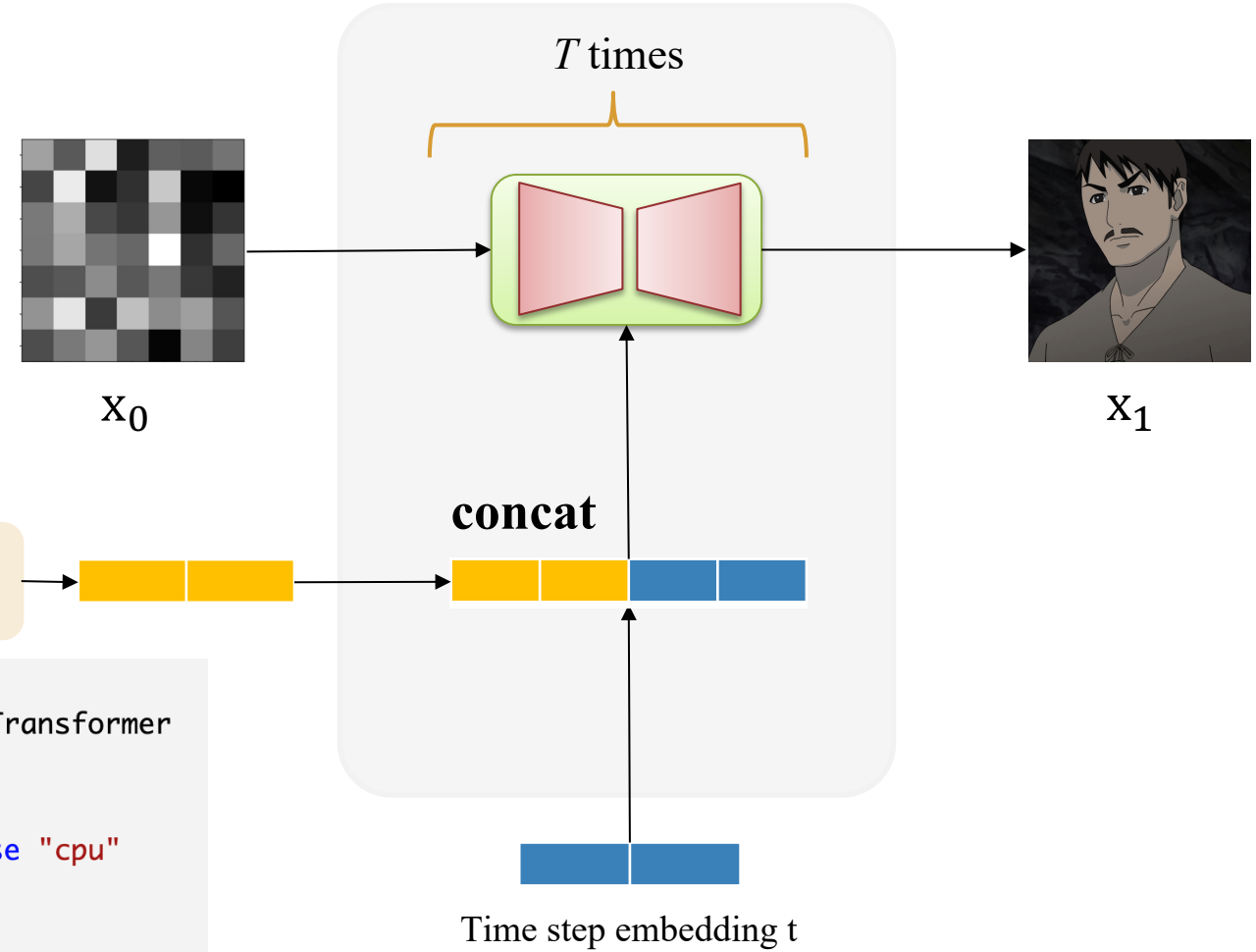
Text-to-Image Generation

Conditional Flow Matching



Text-to-Image Generation

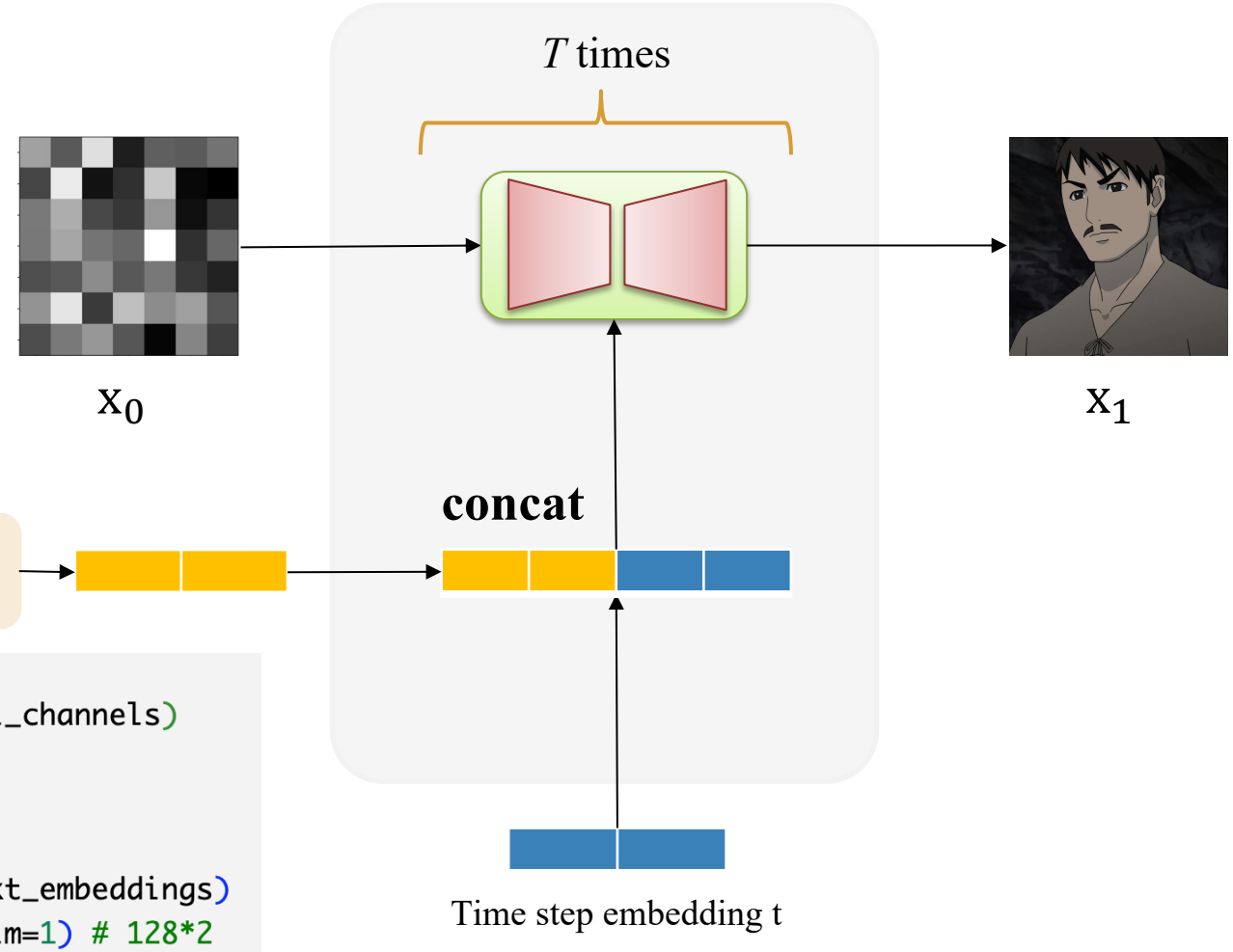
Conditional Flow Matching




```python
import torch
from sentence_transformers import SentenceTransformer

device = torch.device(
    "cuda" if torch.cuda.is_available() else "cpu"
)
text_encoder = SentenceTransformer(
    "all-mpnet-base-v2"
).to(device)
```

Text-to-Image Generation

Conditional Flow Matching

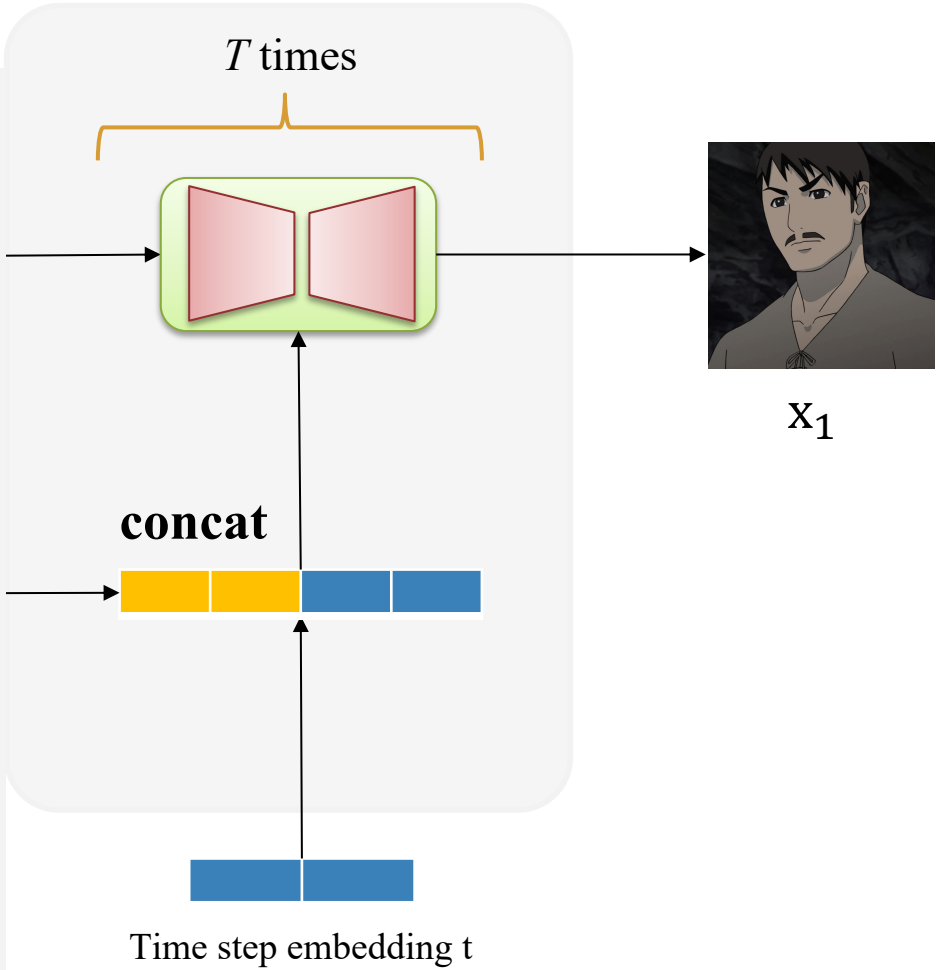



```
emb = self.time_embed(
    timestep_embedding(timesteps, self.model_channels)
)

if text_embeddings is not None:
    text_embedded = self.embedding_layer(text_embeddings)
    emb = torch.cat([emb, text_embedded], dim=1) # 128*2
    emb = self.fc(emb)
```

Text-to-Image Generation

Conditional Flow Matching

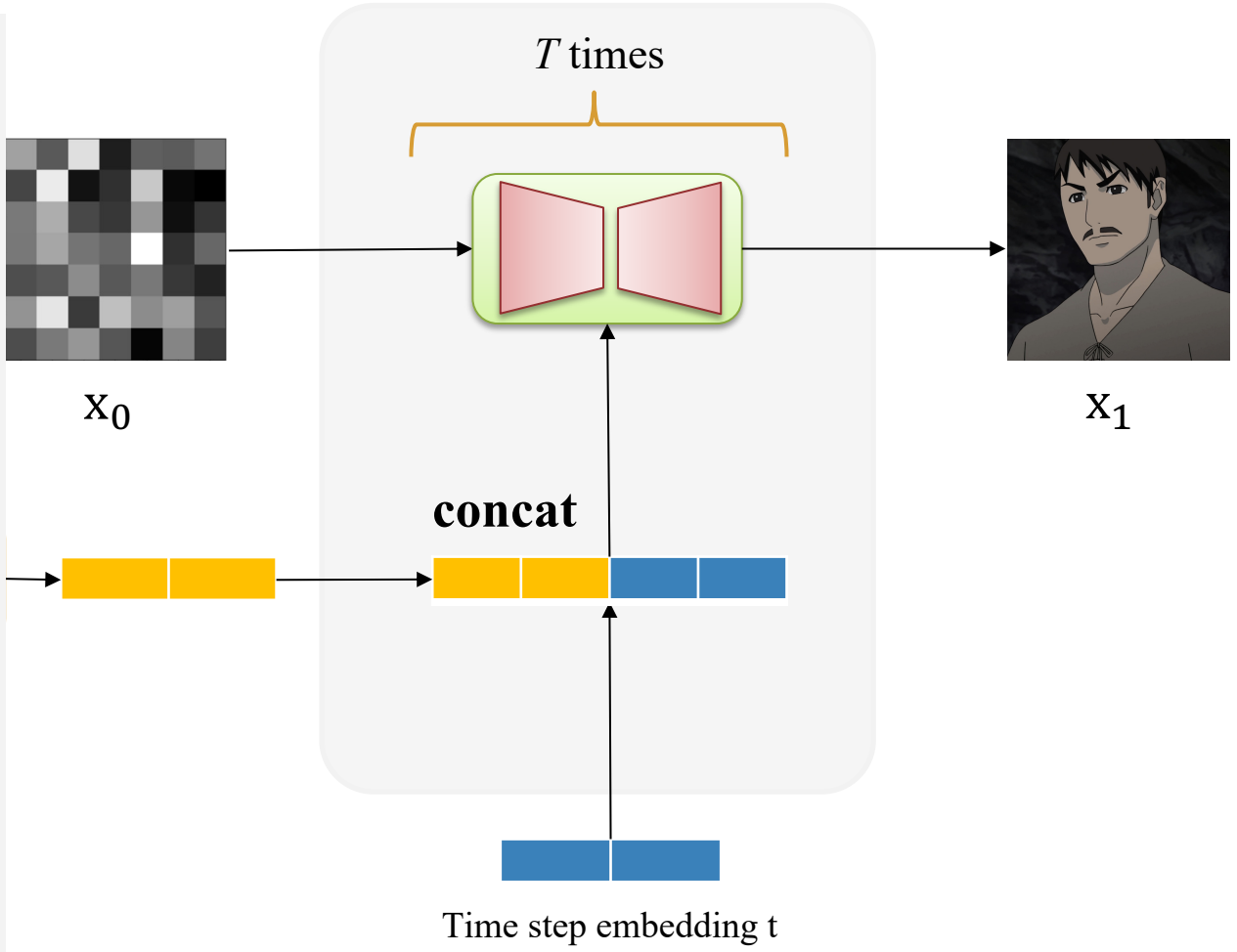
```
x1 = batch["image"].to(device)
text_embeddings = batch["caption_embedding"].to(device)

x0 = torch.randn_like(x1).to(device)

t = torch.rand(x0.shape[0], 1, 1, 1).to(device)

xt = t * x1 + (1 - t) * x0
ut = x1 - x0

t = t.squeeze()

vt = model(t, xt, text_embeddings=text_embeddings)

loss = torch.mean((vt - ut) ** 2)

loss.backward()
optimizer.step()
```



Text-to-Image Generation

Inference

```
model.eval()
def euler_method(model, text_embedding, t_steps, dt, noise):
    y = noise
    y_values = [y]
    with torch.no_grad():
        for t in t_steps[1:]:
            t = t.reshape(-1, )
            dy = model(
                t.to(device), y,
                text_embeddings=text_embedding
            )
            y = y + dy * dt
            y_values.append(y)
    return torch.stack(y_values)

# Initial random image and class (optional)
noise = torch.randn((3, 64, 64), device=device).unsqueeze(0)
text_embedding = text_embeddings[1].unsqueeze(0).to(device)

# Time parameters
t_steps = torch.linspace(0, 1, 100, device=device)
dt = t_steps[1] - t_steps[0]

# Solve the ODE using Euler method
results = euler_method(model, text_embedding, t_steps, dt, noise)
```



Text-Guided Image Generation

Description

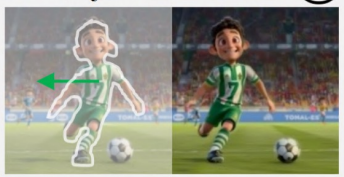
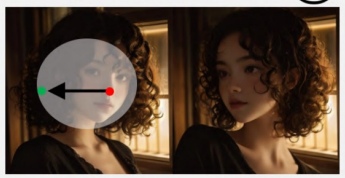
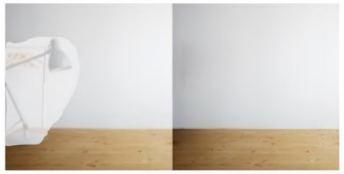
- **Text-Guided Image Generation** aims to edit the given synthetic or real image to meet the specific requirements from the prompt.
- Based on the development of text-to-image models, which generate images according to text prompts

This pink and yellow flower has a beautiful yellow center with many stamens

A photo of an open door

Text-Guided Image Generation

Content-Aware Editing: Local Editing

Object Manipulation	"... with a glasses" 	"Remove red flowers" 	"... ~~basketball~~ football ..." 	Replacement by Exemplar Image 
Attribute Manipulation	"...is standing" 	"A fluffy (↑) doll" 	"...~~round~~ square..." 	Appearance Transfer 
Spatial Transformation	Object Translation 	Object Scaling 	Local Distortion 	
Inpainting	" " 	"A parrot" 	"A vase " 	Inpainting with Exemplar Image 

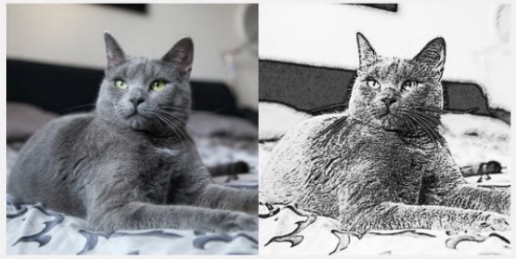
A survey of Multimodal-Guided Image Editing

Text-Guided Image Generation

Content-Aware Editing: Global Editing

Style Change

"... in penciled style"



"Turn it to Van Goph style"

Style Transfer

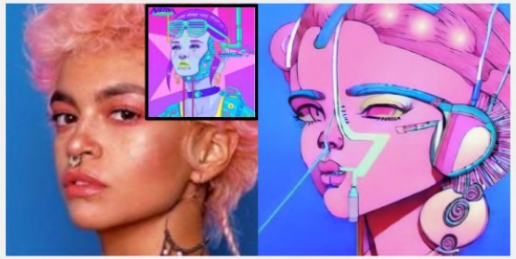
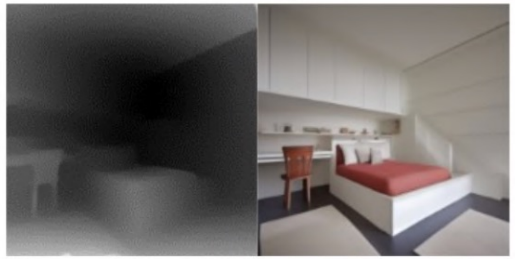
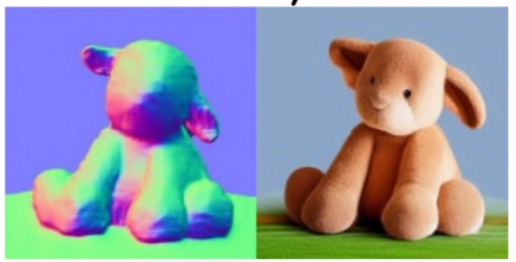


Image Translation

"A room"



"A toy"

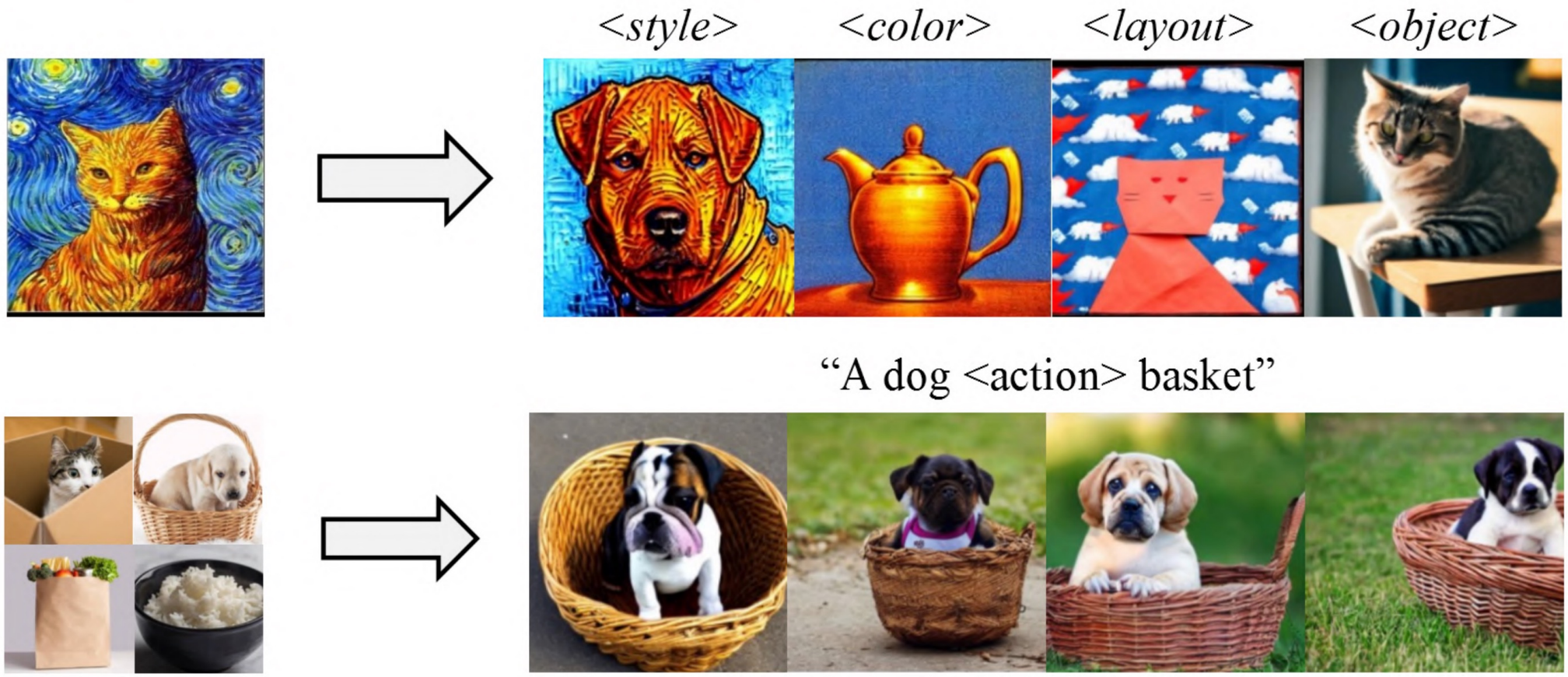


"A man"

[A survey of Multimodal-Guided Image Editing]

Text-Guided Image Generation

Content-Free Editing



A survey of Multimodal-Guided Image Editing

Text-Guided Image Generation

Dataset

- Oxford 120 Flowers, CUB-200, Recipe1M, …
- TedBench Dataset

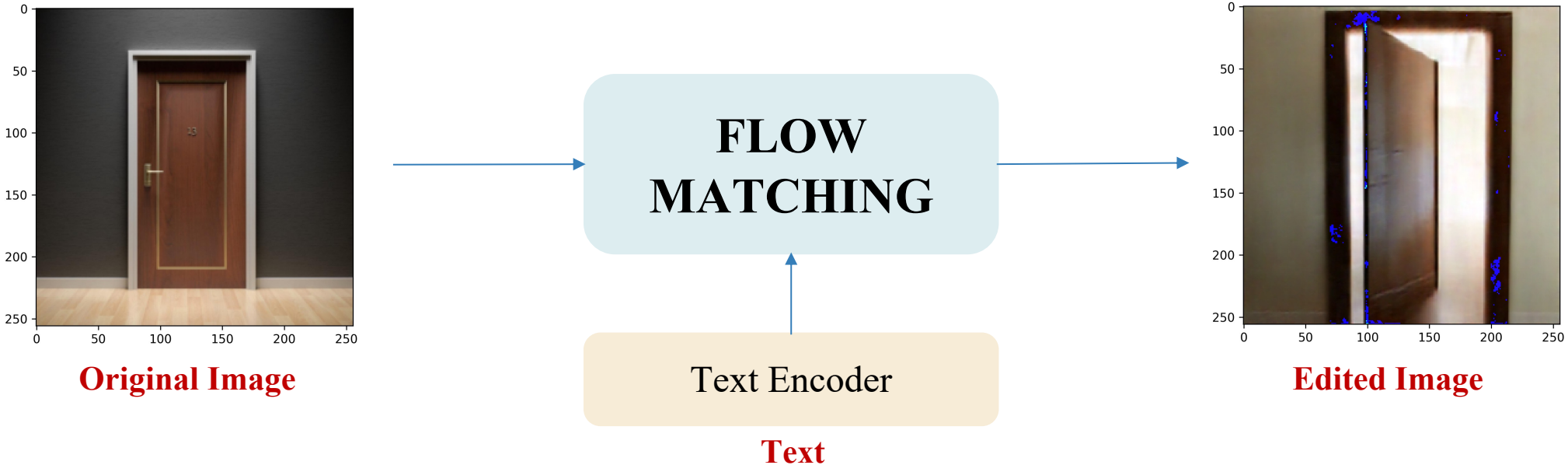
A photo of a chair swed in half

A photo of an open door

Text-Guided Image Generation

Methods

- **Text Embedding**: Convert text into a vector representation
- **Image Encoder**: The original image is encoded to extract its feature
- **Text-Image Alignment**: the image aligns with the text description requires robust cross-modal learning
- **Generator**: The decoder reconstructs the image from the modified features



Text-Guided Image Generation

Preprocessing

Text Representation (Text Encoder)

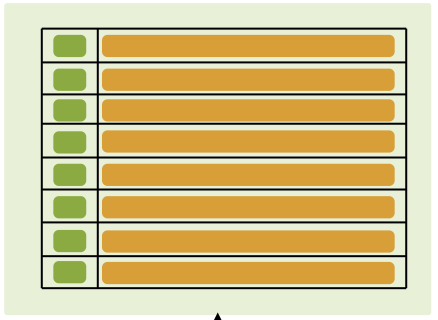
Caption

A man with dark hair and brown eyes

vector

index	token
0	<unk>
1	<pad>
2	a
3	man
4	bright
5	dark
…	…

2
3
0
5
…



Encoder (BERT)

Pooling

[-0.1676, -0.0853, …, -0.5376]

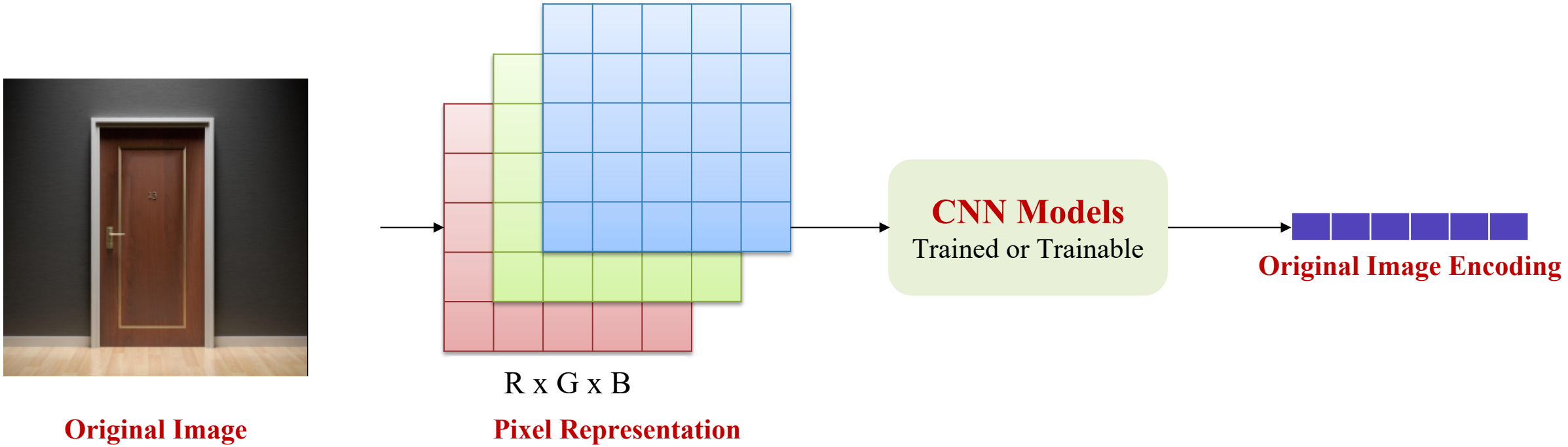
Text Embedding

[-0.0166, -0.0666, …, -0.0421]
[-0.0132, -0.0673, …, -0.0521]
[-0.0176, -0.0709, …, -0.0363]
[-0.0104, -0.0548, …, -0.0352]
[-0.0132, -0.0673, …, -0.0521]
[-0.0176, -0.0709, …, -0.0363]

Text-Guided Image Generation

Preprocessing

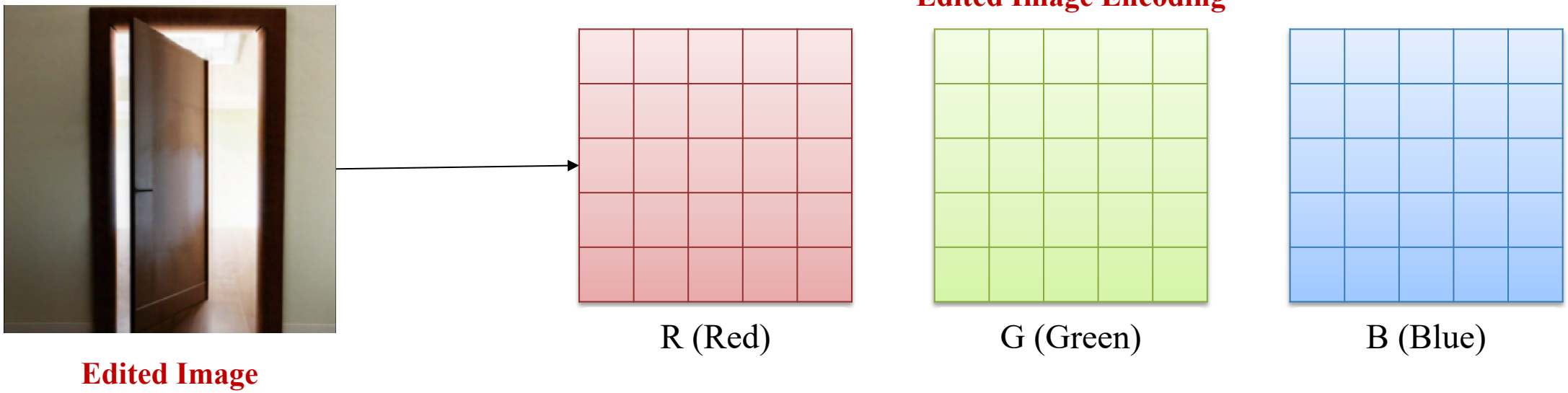
- **Original Image Encoding:** Encoded to extract high-level features that can guide the generation progress through a CNN or a pre-trained model as a image encoder.



Text-Guided Image Generation

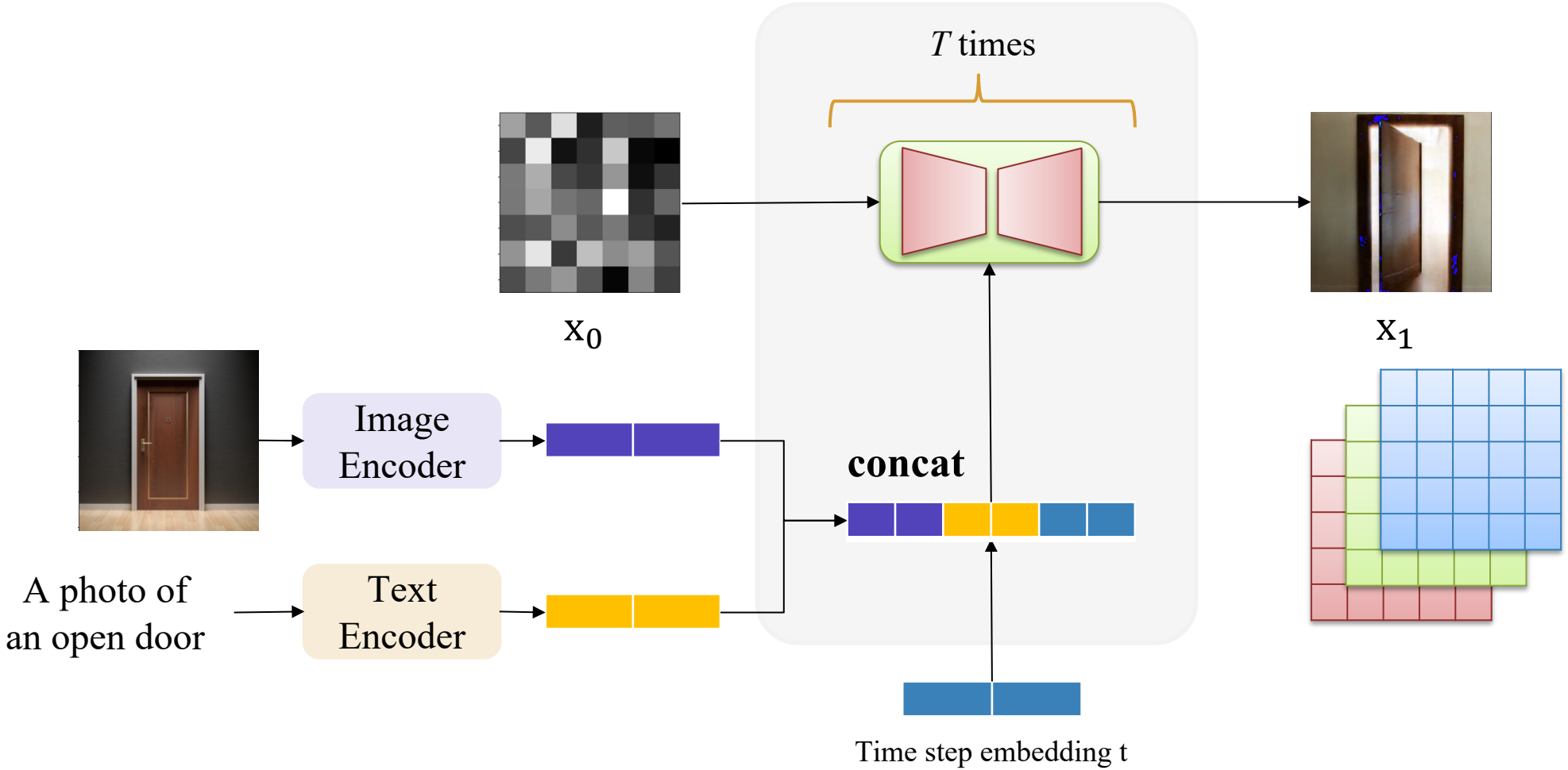
Preprocessing

- **Edited Image Encoding:** The generated image is output as a tensor or array representing pixel values, typically of shape (channels, height, width) for neural network processing, would be normalized



Text-Guided Image Generation

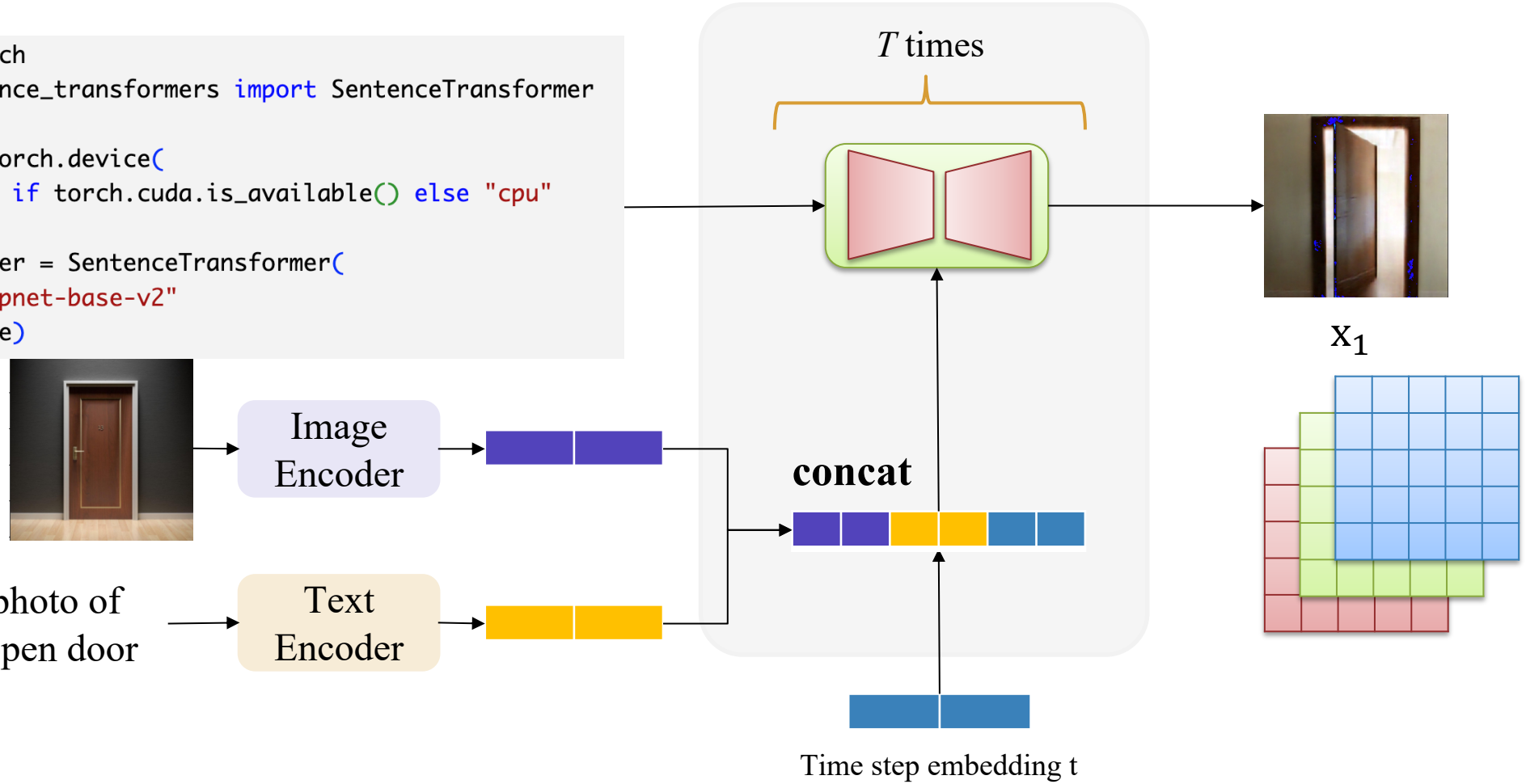
Conditional Flow Matching



Text-Guided Image Generation

Conditional Flow Matching

```python
import torch
from sentence_transformers import SentenceTransformer

device = torch.device(
    "cuda" if torch.cuda.is_available() else "cpu"
)
text_encoder = SentenceTransformer(
    "all-mpnet-base-v2"
).to(device)
```



Text-Guided Image Generation

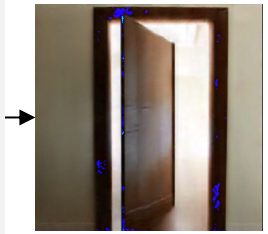
Conditional Flow Matching

Image Encoder

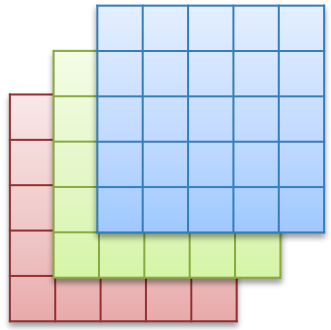
A photo of an open door

Text Encoder

```
# get an original image
original_image = self.original_images[idx]
original_image = self.transform(original_image)

# get a output image
edited_image = self.edited_images[idx]
edited_image = self.transform(edited_image)

# get a text
caption = self.captions[idx]
caption_embedding = self.embed_captions[idx]

return {
    "original_image": original_image,
    "edited_image": edited_image,
    "caption": caption,
    "caption_embedding": caption_embedding,
}
```



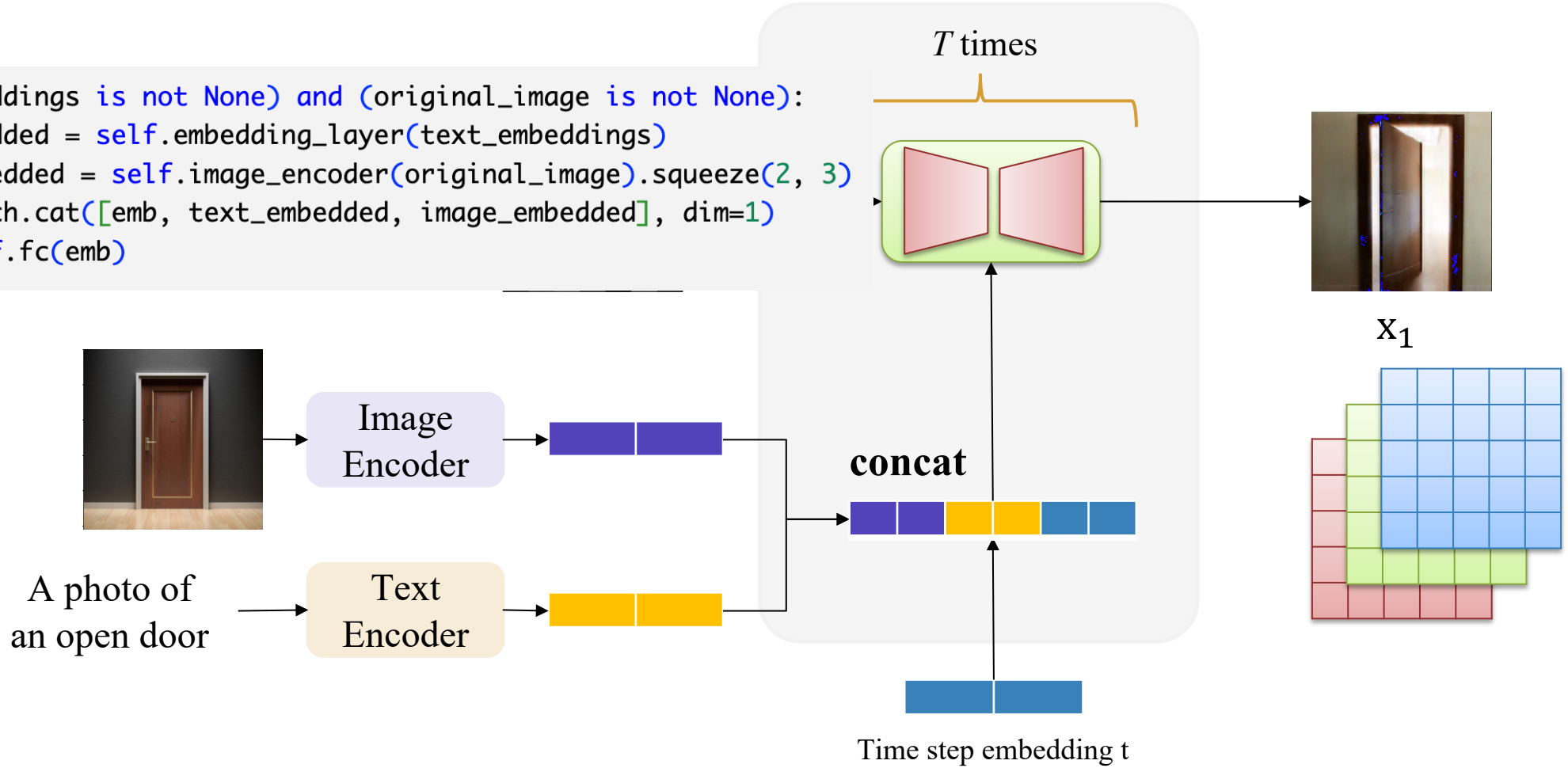
x_1



Text-Guided Image Generation

Conditional Flow Matching

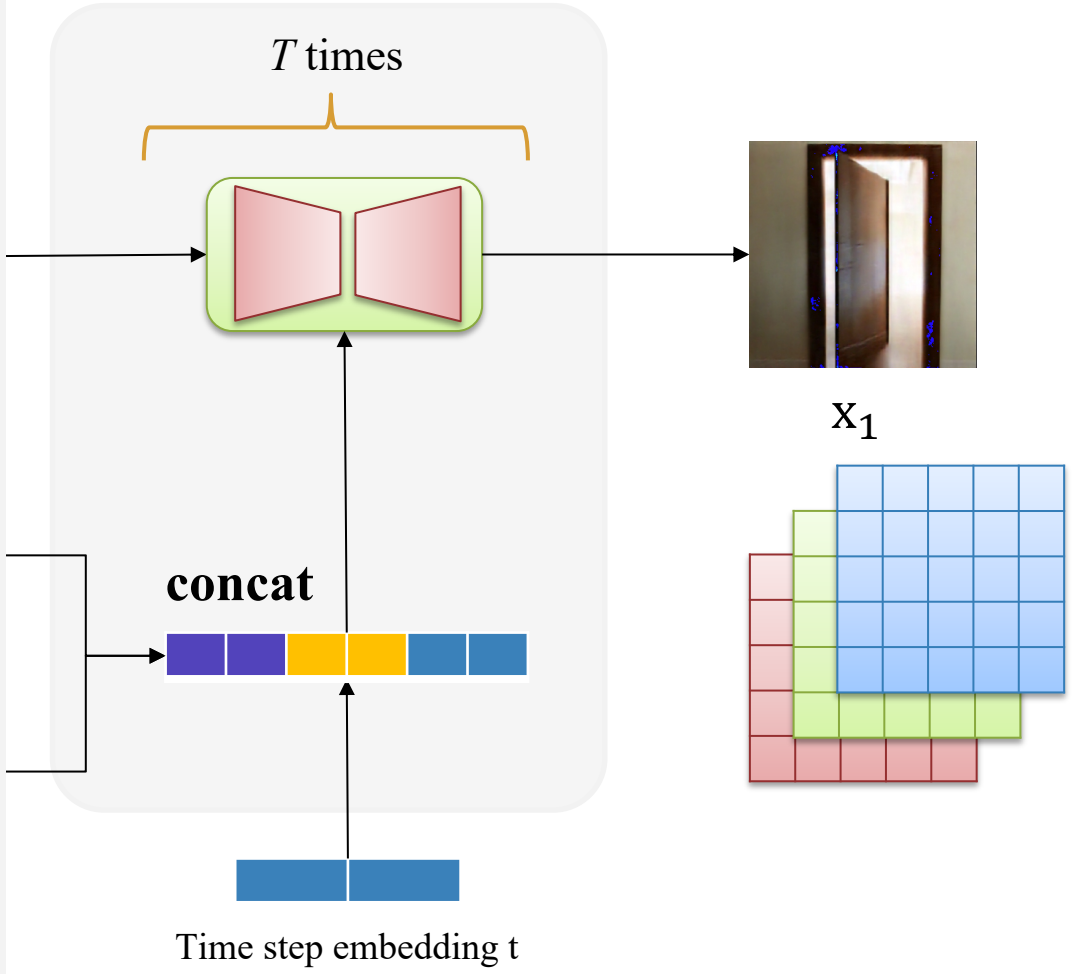
```
if (text_embeddings is not None) and (original_image is not None):
    text_embedded = self.embedding_layer(text_embeddings)
    image_embedded = self.image_encoder(original_image).squeeze(2, 3)
    emb = torch.cat([emb, text_embedded, image_embedded], dim=1)
    emb = self.fc(emb)
```



Text-Guided Image Generation

Conditional Flow Matching

```
optimizer.zero_grad()
x1 = edited_image

x0 = torch.randn_like(x1).to(device)

t = torch.rand(x0.shape[0], 1, 1, 1).to(device)

xt = t * x1 + (1 - t) * x0
ut = x1 - x0

t = t.squeeze()

vt = model(
    t, xt,
    text_embeddings=caption_embedding,
    original_image=original_image
)

loss = torch.mean(((vt - ut) ** 2))

loss.backward()
optimizer.step()
```

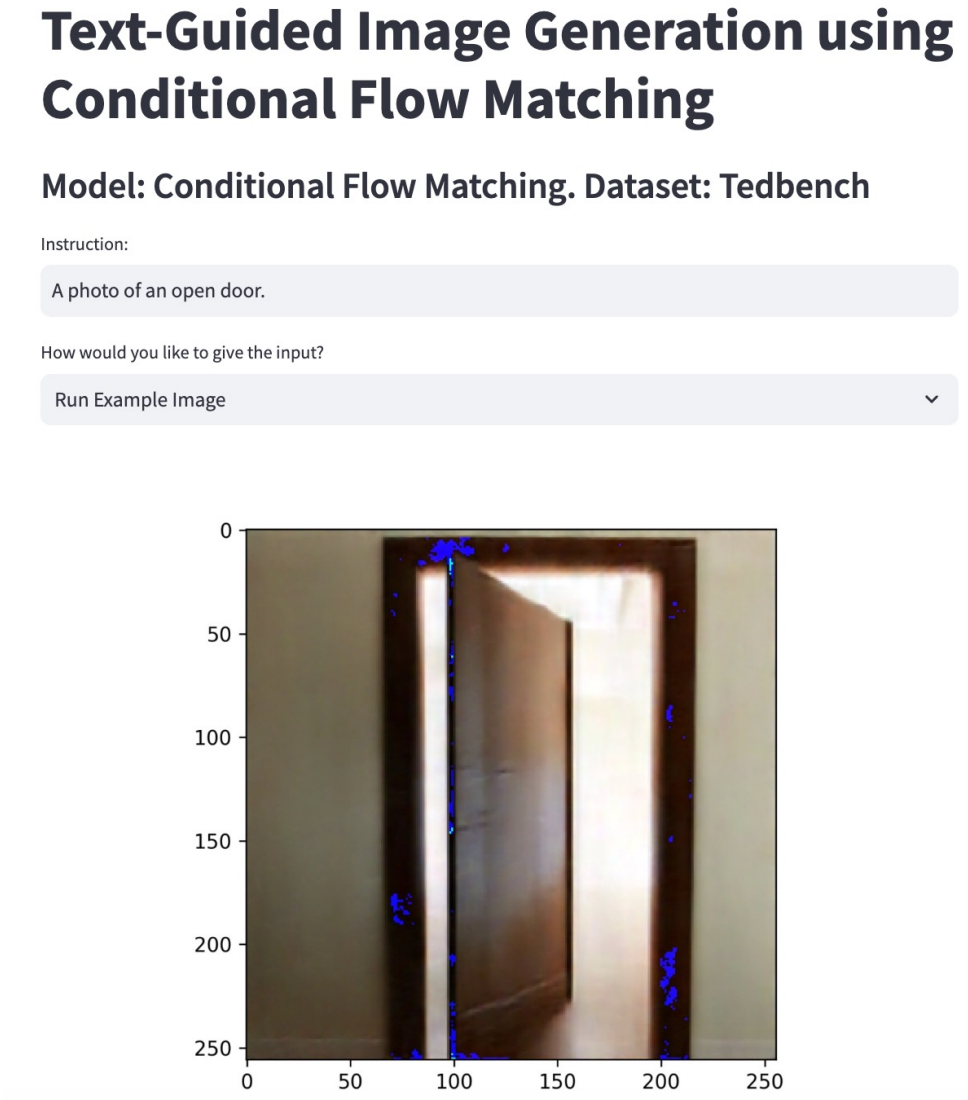


Text-Guided Image Generation

Deployment

- Github - Demo

A photo of an open door



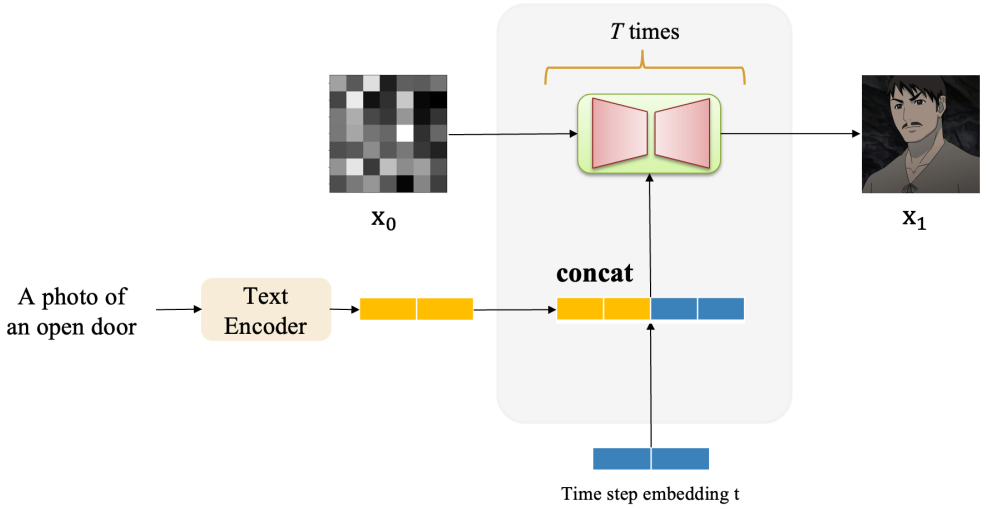
Objectives

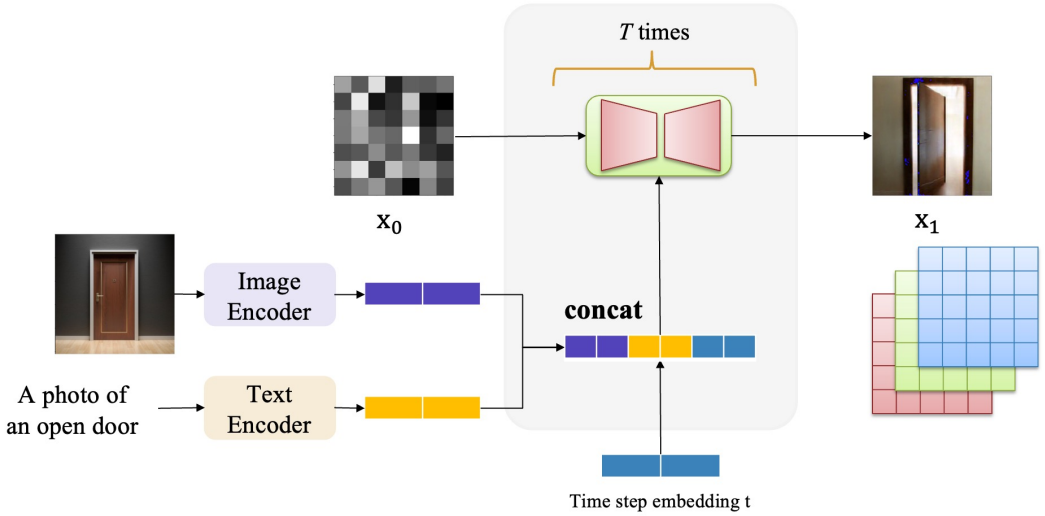
Conditional Flow Matching (Review)

- Flow Matching Model
- Training & Sampling
- Conditional Flow Matching

Text-Guided Image Generation

- Text-To-Image Generation
- Text-Guided Image Generation
- TedBench Dataset





Thanks!

Any questions?