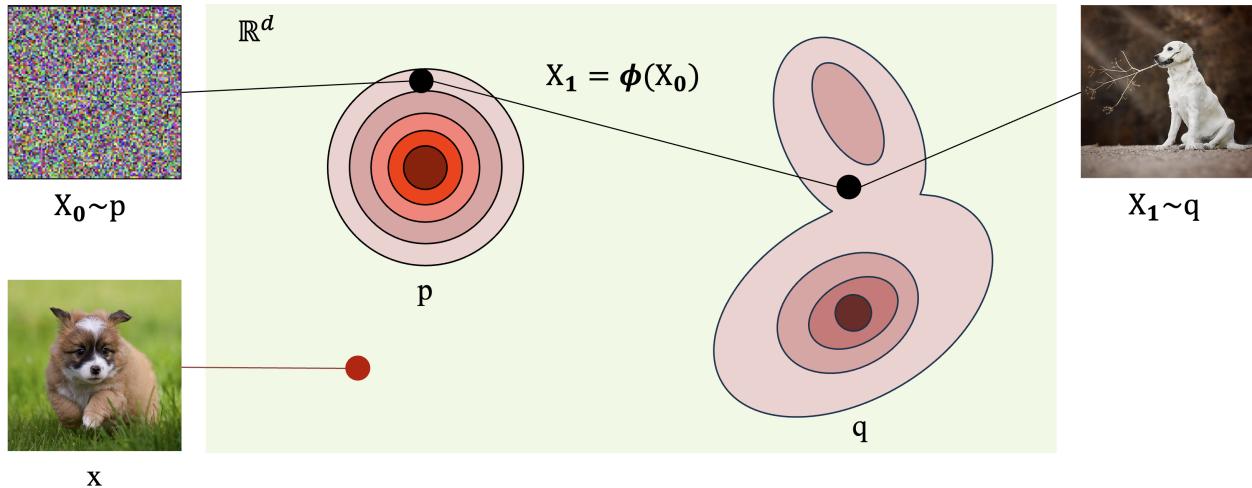


Exercise: Conditional Flow Matching

Quoc-Thai Nguyen, Quang-Hien Ho và Quang-Vinh Dinh

Ngày 29 tháng 3 năm 2025

Phần 1. Flow Matching



Hình 1: Flow Matching.

Flow Matching là một phương pháp hiệu quả để huấn luyện các mô hình sinh (generative models) dựa trên dòng (flow). Khác với Diffusion Models, Flow Matching tập trung vào việc học vận tốc chuyển đổi giữa phân phối dữ liệu và phân phối noise, sau đó tạo mẫu bằng cách đi theo vận tốc này.

Trong bài toán mô hình sinh, mục tiêu là tạo ra các mẫu mới từ một phân phối cơ bản (phân phối dữ liệu huấn luyện). Gọi:

- $p_0(x_0)$ là phân phối dữ liệu thực
 - $p_1(x_1)$ là phân phối noise (thường là phân phối Gaussian)
- Mục tiêu của mô hình sinh là học một hàm ánh xạ ϕ sao cho:
- $X_0 \sim p_0$ (dữ liệu thực)
 - $X_1 = \phi(X_0) \sim p_1$ (noise)

Flow Models định nghĩa một quá trình chuyển đổi liên tục giữa hai phân phối. Đối với mỗi thời điểm $t \in [0, 1]$, ta có một phân phối trung gian p_t và một ánh xạ $\phi_{t+h|t}$ sao cho:

- $X_t \sim p_t$
- $X_{t+h} = \phi_{t+h|t}(X_t)$

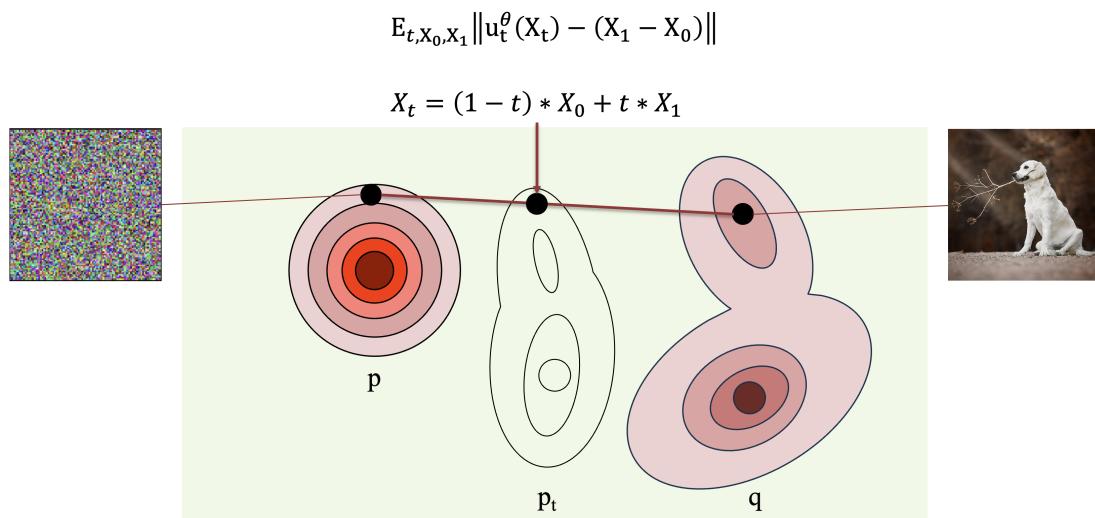
Flow Matching là một phương pháp hiệu quả để huấn luyện các mô hình flow bằng cách:

- Học một trường vận tốc (velocity field) thay vì học trực tiếp hàm ánh xạ
- Lấy mẫu bằng cách theo dõi vận tốc này qua phương trình vi phân

Trong Flow Matching, vận tốc đại diện cho tốc độ chuyển đổi giữa dữ liệu sạch và dữ liệu nhiễu. Cụ thể:

- v_t : vận tốc tại thời điểm t
- $\psi_t(x)$: flow ánh xạ tại thời điểm t
- $u_t(x)$: vận tốc thực, được tính bằng $x_1 - x_0$

1.1. Huấn luyện mô hình



Hình 2: Training Flow Matching.

Để cập nhật trọng số mô hình dự đoán θ , quá trình huấn luyện Flow Matching cho mỗi epoch bao gồm các bước sau:

- Chọn ngẫu nhiên một mẫu dữ liệu sạch $x_0 \sim p_0$
- Chọn ngẫu nhiên một mẫu noise $x_1 \sim p_1$
- Chọn ngẫu nhiên một thời điểm $t \in [0, 1]$
- Tính điểm nội suy (interpolated point) $x_t = (1 - t)x_0 + tx_1$
- Tính vận tốc thực $u_t = x_1 - x_0$
- Dự đoán vận tốc bằng mô hình $v_t = f_\theta(x_t, t)$
- Tính hàm mất mát $l_t = |v_t - u_t|^2$

(h) Cập nhật tham số mô hình θ bằng gradient descent

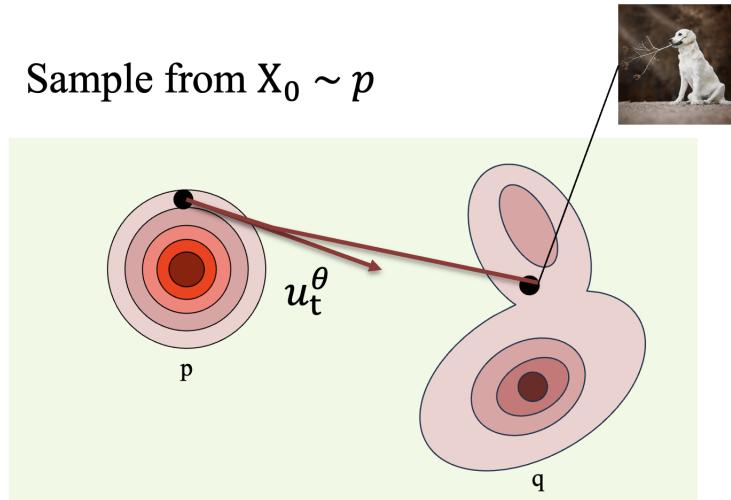
Hàm mất mát tổng quát có thể viết dưới dạng:

$$\mathcal{L} = \mathbb{E}_{t, X_0, X_1} |u_t^\theta(X_t) - (X_1 - X_0)|^2$$

Trong đó:

- $X_t = (1-t)X_0 + tX_1$
- $X_0 \sim p_0$
- $X_1 \sim p_1$
- $t \sim \mathcal{U}(0, 1)$

1.2. Lấy mẫu



Hình 3: Sampling in Flow Matching.

Sau khi huấn luyện mô hình dự đoán vận tốc $f_\theta(y, t)$, chúng ta có thể lấy mẫu bằng cách giải phương trình vi phân:

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

Phương pháp phổ biến để giải phương trình này là phương pháp Euler:

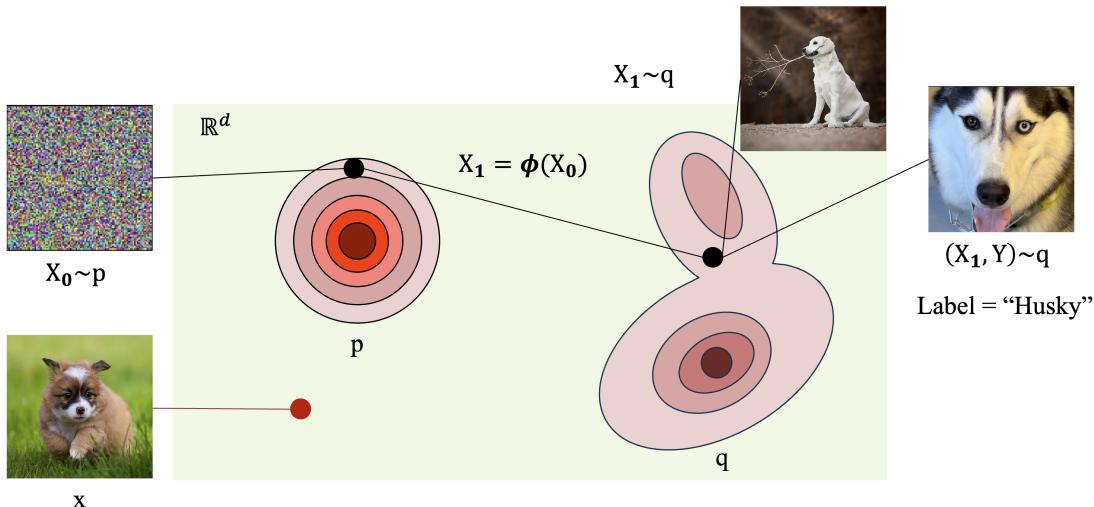
$$y(t + \Delta t) \approx y(t) + f_\theta(y, t)\Delta t$$

Quá trình lấy mẫu bao gồm các bước:

- (a) Khởi tạo y từ phân phối noise p_1
- (b) Chia khoảng thời gian $[0, 1]$ thành n bước nhỏ: t_1 , với $\Delta t = t_i - t_{i-1}$
- (c) Tại mỗi bước i :
 - Tính vận tốc $f_\theta(y, t_i)$
 - Cập nhật $y = y + f_\theta(y, t_i)\Delta t$
- (d) Kết quả cuối cùng là mẫu từ phân phối p_0

Phần 2. Conditional Flow Matching

Conditional Flow Matching mở rộng Flow Matching với khả năng điều kiện hóa (conditioning). Điều này cho phép mô hình tạo ra dữ liệu có điều kiện dựa trên các thuộc tính hoặc thông tin bổ sung.



Hình 4: Conditional Flow Matching.

2.1. Dự đoán theo điều kiện

Trong mô hình có điều kiện, chúng ta làm việc với:

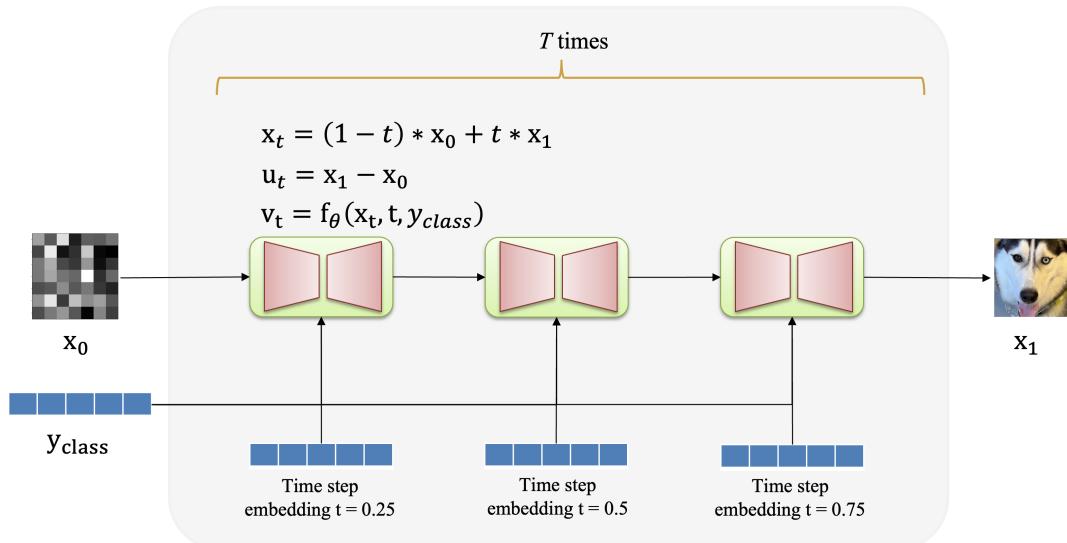
- $X_0 \sim p$: dữ liệu thực
- $X_1 \sim q$: noise
- $(X_1, Y) \sim q$: cặp noise và thông tin điều kiện

Thông tin điều kiện Y có thể là các nhãn lớp, văn bản, hoặc các dữ liệu khác giúp hướng dẫn quá trình sinh.

2.2. Huấn luyện Conditional Flow Matching

Quá trình huấn luyện cho Conditional Flow Matching tương tự như Flow Matching, nhưng bổ sung thông tin điều kiện vào mô hình học sâu để dự đoán các kết quả theo điều kiện:

- (a) Chọn ngẫu nhiên một mẫu có điều kiện $(x_1, y_1) \sim q$
- (b) Chọn ngẫu nhiên một mẫu dữ liệu sạch $x_0 \sim p_0$
- (c) Chọn ngẫu nhiên một thời điểm $t \in [0, 1]$
- (d) Tính điểm nội suy $x_t = (1 - t)x_0 + tx_1$
- (e) Tính vận tốc thực $u_t = x_1 - x_0$
- (f) Dự đoán vận tốc bằng mô hình có điều kiện $v_t = f_\theta(x_t, t, y_1)$
- (g) Tính hàm mất mát $l_t = |v_t - u_t|^2$
- (h) Cập nhật tham số mô hình θ bằng gradient descent



Hình 5: Flow Matching.

Hàm mất mát tổng quát trở thành:

$$\mathcal{L} = \mathbb{E}_{t, X_0, X_1, Y} |u_t^\theta(X_t, Y) - (X_1 - X_0)|^2$$

2.3. Lấy mẫu từ Conditional Flow Matching

Quá trình lấy mẫu cũng tương tự nhưng sử dụng thông tin điều kiện:

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

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

Trong đó y_{class} là thông tin điều kiện (ví dụ: nhãn lớp) mà chúng ta muốn sử dụng để hướng dẫn quá trình sinh.

Phần 3. Image Inpainting using Conditional Flow Matching

Image Inpainting là kỹ thuật điền vào những phần bị thiếu trong hình ảnh, đảm bảo rằng kết quả:

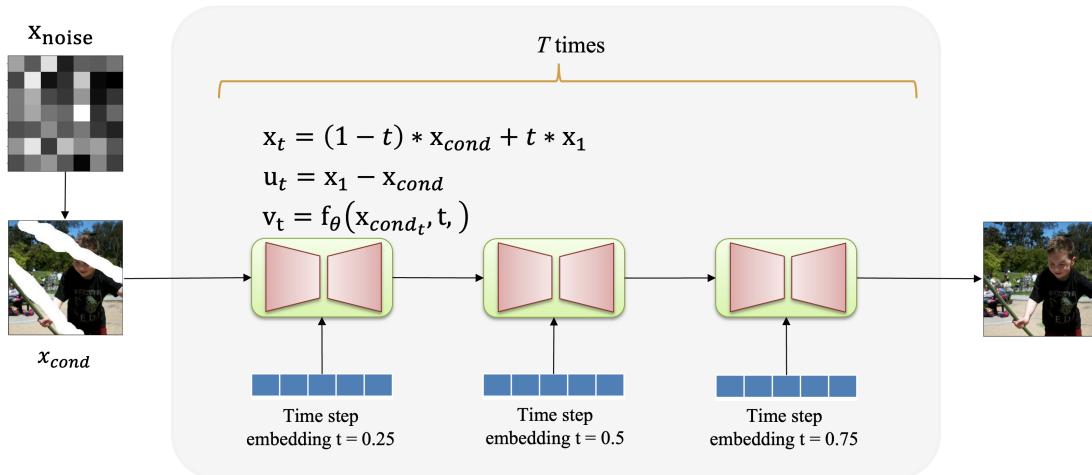
1. Thực tế về mặt thị giác
2. Hợp lý về ngữ nghĩa

3.1. Các kỹ thuật tạo mặt nạ (masking)

Có nhiều kỹ thuật để tạo mặt nạ cho Image Inpainting:

- Center Box Mask: mặt nạ hình chữ nhật ở giữa
- Random Box Mask: mặt nạ hình chữ nhật ở vị trí ngẫu nhiên
- Irregular Mask: mặt nạ có hình dạng không đều
- Free-Form Mask: mặt nạ tự do
- Hybrid Mask: kết hợp nhiều loại mặt nạ

3.2. Mô hình Inpainting sử dụng Conditional Flow Matching



Hình 6: Flow Matching.

Trong Image Inpainting với Conditional Flow Matching, thông tin điều kiện là hình ảnh không đầy đủ (có phần bị che khuất) x_{cond} . Mô hình sẽ học cách điền vào phần bị thiếu dựa trên thông tin còn lại. Quá trình huấn luyện:

- (a) Chọn một hình ảnh đầy đủ x_1
- (b) Tạo mặt nạ và áp dụng lên x_1 để tạo ra hình ảnh điều kiện x_{cond}
- (c) Tính điểm nội suy $x_t = (1 - t)x_{cond} + tx_1$

- (d) Tính vận tốc thực $u_t = x_1 - x_{cond}$
- (e) Dự đoán vận tốc bằng mô hình $v_t = f_\theta(x_t, t, x_{cond})$
- (f) Tính hàm mất mát và cập nhật mô hình

Quá trình sinh (inference):

- (a) Khởi tạo x_1 từ phân phối noise
- (b) Định nghĩa x_{cond} là hình ảnh không đầy đủ cần điền
- (c) Chia khoảng thời gian $[0, 1]$ thành các bước
- (d) Tại mỗi bước:
 - Tính vận tốc $v_t = f_\theta(x_t, t, x_{cond})$
 - Cập nhật x_t theo phương pháp Euler
- (e) Kết quả cuối cùng là hình ảnh đã được hoàn thiện

3.3. Thực nghiệm

Trong phần này chúng ta sẽ tiến hành thực nghiệm huấn luyện mô hình với các bước sau:

- (a) Tài về bộ dữ liệu:

```

1 # Dataset
2 !gdown 194DqJkUjjt1UCp7Sd2DkXgwu7SuBFsrj
3 !unzip celeba_hq_256.zip
4
5 import os
6 import random
7
8 file_names = os.listdir("./celeba_hq_256")
9 img_paths = ["../celeba_hq_256/" + file_name for file_name in file_names]
10 len(img_paths)
11 sample_size = int(len(img_paths) * 0.9)
12 train_imgpaths = random.sample(img_paths, sample_size)
13 val_imgpaths = [img_path for img_path in img_paths if img_path not in
14     train_imgpaths]
15 len(train_imgpaths), len(val_imgpaths)

```

- (b) Tiền xử lý dữ liệu:

```

1 import numpy as np
2 def bbox2mask(img_shape, bbox, dtype='uint8'):
3     """ bbox (tuple[int]): Configuration tuple, (top, left, height, width)
4     height, width = img_shape[:2]
5     mask = np.zeros((height, width, 1), dtype=dtype)
6     mask[bbox[0]:bbox[0] + bbox[2], bbox[1]:bbox[1] + bbox[3], :] = 1
7     return mask

```

```
1 import torch
2 from PIL import Image
3 from torchvision import transforms
4 from torch.utils.data import Dataset
5 class InpaintingDataset(Dataset):
6     def __init__(self, img_paths, image_size=[256, 256]):
7         self.img_paths = img_paths
8         self.tfs = transforms.Compose([transforms.Resize((image_size[0],
9             image_size[1])), transforms.ToTensor()])
10        self.image_size = image_size
11    def __getitem__(self, index):
12        img_path = self.img_paths[index]
13        img = Image.open(img_path).convert('RGB')
14        img = self.tfs(img)
15        mask = self.get_mask()
16        mask_img = img*(1. - mask) + mask
17        return {
18            "gt_image": img, "cond_image": cond_image,
19            "mask": mask, "path": img_path
20        }
21    def __len__(self):
22        return len(self.img_paths)
23    def get_mask(self):
24        h, w = self.image_size # Center mask
25        mask = bbox2mask(self.image_size, (h//4, w//4, h//4, w//4))
26        return torch.from_numpy(mask).permute(2,0,1)
27 train_dataset = InpaintingDataset(train_imgpaths)
28 batch_size = 64 # (GPU 24GB)
29 train_loader = torch.utils.data.DataLoader(
30     train_dataset, batch_size=batch_size, shuffle=True, drop_last=True
31 )
```

Hiển thị một số hình ảnh để kiểm tra:



Hình 7: Example.

(c) Huấn luyện mô hình:

```
1 # Model
2 ! pip install -q torchcfm
3
4 from torchcfm.models.unet import UNetModel
5 from tqdm import tqdm
6 device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
7 model = UNetModel(dim=(3, 256, 256), num_channels=32, num_res_blocks=1).to(device)
8 optimizer = torch.optim.Adam(model.parameters())
9
10 n_epochs = 1000
11 for epoch in range(n_epochs):
12     losses = []
13     for i, data in tqdm(enumerate(train_loader)):
14         optimizer.zero_grad()
15         x1 = data["gt_image"].to(device)
16         mask = data["mask"].to(device)
17         x0 = torch.randn_like(x1).to(device)
18
19         x_noise = (1.0-mask)*x1 + mask*x0
20         t = torch.rand(x0.shape[0], 1, 1, 1).to(device)
21         xt = t * x1 + (1 - t) * x_noise
22         ut = x1 - x_noise
23
24         t = t.squeeze()
25         x_cond = xt*mask + (1.0-mask)*x1
26         vt = model(t, x_cond)
27
28         loss = torch.mean(((vt - ut) ** 2)*mask)
29         loss.backward()
30         optimizer.step()
31         losses.append(loss.item())
32     avg_loss = sum(losses) / len(losses)
33
34     print(f"epoch: {epoch}, loss: {avg_loss:.4}")
```

(d) Dự đoán:

```
1 model.eval()
2 def euler_method(model, cond_image, t_steps, dt, mask):
3     y = cond_image
4     y_values = [y]
5     with torch.no_grad():
6         for t in t_steps[1:]:
7             t = t.reshape(-1, )
8             dy = model(t.to(device), y)
9             y = y + dy * dt
10            y = cond_image*(1. - mask) + mask*y
11            y_values.append(y)
12    return torch.stack(y_values)
13
14 # Initial random image and class (optional)
15 sample = next(iter(train_loader))
16 gt_image = sample['gt_image'].to(device)
17 noise = torch.randn_like(gt_image, device=device)
18 mask = sample['mask'].to(device)
19 cond_image = gt_image*(1. - mask) + mask*noise
20
21 # Time parameters
22 t_steps = torch.linspace(0, 1, 50, device=device) # Two time steps from 0 to 1
23 dt = t_steps[1] - t_steps[0] # Time step
24
25 # Solve the ODE using Euler method
26 traj = euler_method(model, cond_image, t_steps, dt, mask)
```

Kết quả thử nghiệm



Hình 8: Kết quả thực nghiệm mô hình sau khi huấn luyện.

(e) Triển khai ứng dụng:

Triển khai mô hình trên streamlit. Tham khảo về [code](#) và [demo](#).

Image Inpainting using Conditional Flow Matching

Model: Conditional Flow Matching. Dataset: CelebA-HQ

How would you like to give the input?

Run Example Image



Hình 9: Kết quả thực nghiệm mô hình sau khi huấn luyện.

Phần 4. Câu hỏi trắc nghiệm

Câu hỏi 1 Mục tiêu chính của Flow Matching là gì?

- a) Tạo mẫu ngẫu nhiên từ phân phối đã cho
- b) Học một trường vận tốc (velocity field) để biến đổi một phân phối dữ liệu sang phân phối khác
- c) Tính toán ma trận Jacobian của mạng nơ-ron
- d) Giảm nhiễu trong dữ liệu hình ảnh

Câu hỏi 2 Mục tiêu của việc học velocity trong Flow Matching là gì?

- a) Dự đoán sự thay đổi giữa dữ liệu nhiễu và dữ liệu sạch
- b) Dự đoán sự phân bố của dữ liệu nhiễu
- c) Tạo dữ liệu có điều kiện từ phân phối gốc
- d) Giảm nhiễu trong dữ liệu âm thanh

Câu hỏi 3 Công thức nào sau đây biểu diễn quá trình sampling trong Flow Matching?

- a) $\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + f(\mathbf{x}(t), t)\Delta t$
- b) $\mathbf{x}(t + \Delta t) = \mathbf{x}(t) - f(\mathbf{x}(t), t)\Delta t$
- c) $\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \sigma(t) \cdot \epsilon(t)\Delta t$
- d) $\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + f_\theta(\mathbf{x}(t), t)\Delta t$

Câu hỏi 4 Công thức nào sau đây mô tả sự thay đổi giữa dữ liệu nhiễu và dữ liệu sạch trong Flow Matching?

- a) $v_t = x_1 - x_0$
- b) $v_t = x_1 + x_0$
- c) $v_t = x_1/x_0$
- d) $v_t = \sigma(t) + x_0$

Câu hỏi 5 Quá trình image inpainting sử dụng Conditional Flow Matching có mục tiêu gì?

- a) Sinh hình ảnh từ nhãn lớp
- b) Phục hồi các phần bị thiếu trong hình ảnh sao cho chúng hợp lý về mặt trực quan và ngữ nghĩa
- c) Thêm nhiễu vào hình ảnh
- d) Dự đoán các giá trị bị thiếu trong dữ liệu âm thanh

Câu hỏi 6 Trong Image Inpainting sử dụng Conditional Flow Matching, kỹ thuật masking nào sau đây được sử dụng để che khuất phần dữ liệu?

- a) Random Masking
- b) Irregular Masking
- c) Free-Form Masking
- d) Tất cả các kỹ thuật trên

Câu hỏi 7 Trong image inpainting với Conditional Flow Matching, công thức nào sau đây mô tả quá trình cập nhật dữ liệu bị thiếu?

- a) $x_t = (1 - t) \cdot x_{\text{cond}} + t \cdot x_1$
- b) $x_t = x_1 + (1 - t) \cdot x_0$
- c) $x_t = t \cdot x_{\text{cond}} + (1 - t) \cdot x_0$
- d) $x_t = x_0 + x_1$

Câu hỏi 8 Bộ dữ liệu được sử dụng trong thực nghiệm là?

- a) CelebA-HQ
- b) MNIST
- c) CIFAR10
- d) CIFAR100

Câu hỏi 9 Kích thước đầu vào của mô hình UNET trong phần thực nghiệm là?

- a) 3x256x256
- b) 3x224x224
- c) 3x160x160
- d) 3x64x64

Câu hỏi 10 Các timesteps trong phần Sampling được khởi tạo thông qua phương pháp nào?

- a) sin
- b) cos
- c) linspace
- d) tanh

Phần 5. Phụ lục

1. **Hint:** Dựa vào file tải về **Image-Inpainting-Conditional-Flow-Matching** để hoàn thiện các đoạn code.
2. **Solution:** Các file code cài đặt hoàn chỉnh và phần trả lời nội dung trắc nghiệm có thể được tải về [tại đây](#) (Lưu ý: Sáng thứ 3 khi hết deadline phần project, admin mới copy các nội dung bài giải nêu trên vào đường dẫn).

3. **Rubric:**

Phần	Kiến Thức	Đánh Giá
1	<ul style="list-style-type: none">- Hiểu rõ bài toán Image Inpainting- Hiểu rõ mô hình Conditional Flow Matching	<ul style="list-style-type: none">- Các bước thực hiện trong mô hình với Flow Matching và Conditional Flow Matching
2.	<ul style="list-style-type: none">- Hiểu rõ mô hình UNet sử dụng trong Conditional Flow Matching- Các kỹ thuật cải tiến mô hình UNet	<ul style="list-style-type: none">- Xây dựng mô hình Image Inpainting dựa trên bộ dữ liệu CelebA-HQ- Cải tiến mô hình với các kỹ thuật huấn luyện mô hình.

- *Hết* -