```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
```

```
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,0.5,0.5),(0.5,0.5,0.5))  # [-1,1]
])

dataset = torchvision.datasets.CIFAR10(
    root='./data',
    train=True,
    download=True,
    transform=transform
)

loader = torch.utils.data.DataLoader(dataset, batch_size=64, shuffle=True)
```

```
100%|██████████| 170M/170M [00:05<00:00, 30.2MB/s]
```

```
class EncoderDecoder(nn.Module):
    def __init__(self):
        super().__init__()

        # Encoder
        self.encoder = nn.Sequential(
            nn.Conv2d(1, 128, 4, 2, 1),   # 32→16
            nn.ReLU(),

            nn.Conv2d(128, 256, 4, 2, 1), # 16→8
            nn.ReLU(),

            nn.Conv2d(256, 256, 4, 2, 1), # 8→4
            nn.ReLU()
        )

        # Decoder
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(256,128,4,2,1), # 4→8
            nn.ReLU(),

            nn.ConvTranspose2d(128,64,4,2,1), # 8→16
            nn.ReLU(),

            nn.ConvTranspose2d(64,3,4,2,1), # 16→32
            nn.Tanh()
        )

    def forward(self,x):
        x = self.encoder(x)
```

```
        x = self.decoder(x)
        return x
```

```
def to_gray(img):
    return img.mean(dim=1, keepdim=True)
```

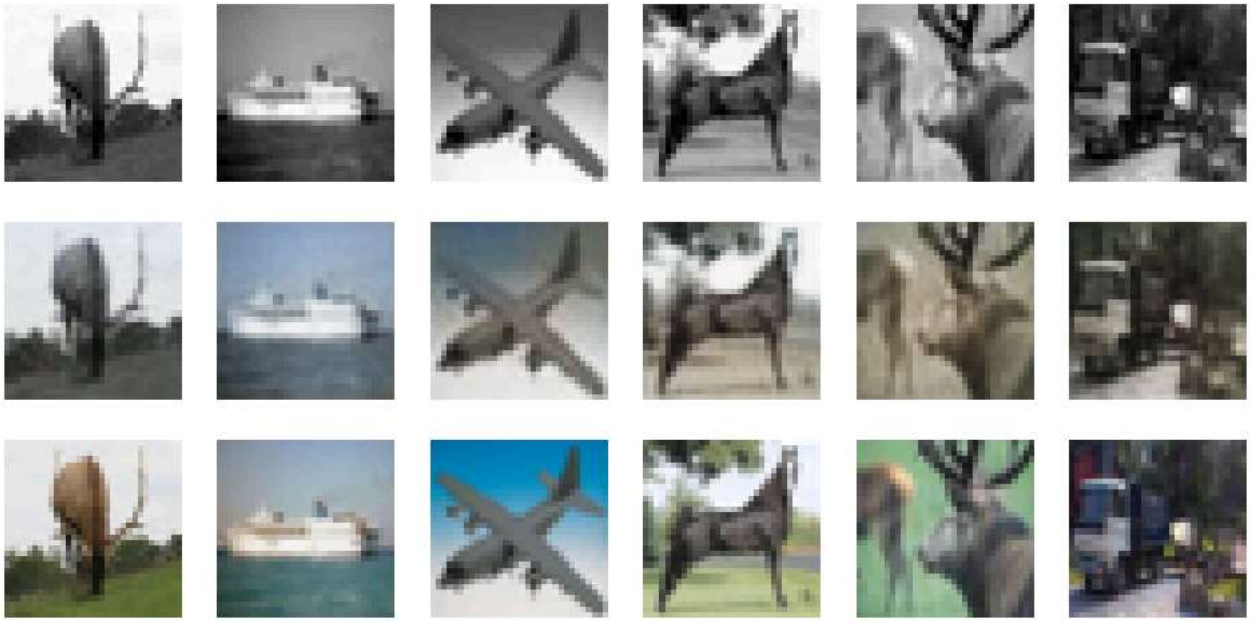
```
from tqdm import tqdm
```

```
device = 'cuda' if torch.cuda.is_available() else 'cpu'

model = EncoderDecoder().to(device)

criterion = nn.L1Loss()   # try MSE later
optimizer = optim.Adam(model.parameters(), lr=0.0002)

epochs = 10

for epoch in range(epochs):

    for imgs,_ in tqdm(loader):

        imgs = imgs.to(device)
        gray = to_gray(imgs)

        output = model(gray)

        loss = criterion(output, imgs)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    print(f"Epoch {epoch+1} Loss: {loss.item():.4f}")
```

```
100%|██████████| 782/782 [00:14<00:00, 52.90it/s]
Epoch 1 Loss: 0.1131
100%|██████████| 782/782 [00:13<00:00, 57.89it/s]
Epoch 2 Loss: 0.1251
100%|██████████| 782/782 [00:13<00:00, 57.59it/s]
Epoch 3 Loss: 0.1131
100%|██████████| 782/782 [00:13<00:00, 58.93it/s]
Epoch 4 Loss: 0.1160
100%|██████████| 782/782 [00:13<00:00, 59.62it/s]
Epoch 5 Loss: 0.1019
100%|██████████| 782/782 [00:20<00:00, 38.84it/s]
Epoch 6 Loss: 0.0921
100%|██████████| 782/782 [00:13<00:00, 59.28it/s]
Epoch 7 Loss: 0.1054
100%|██████████| 782/782 [00:13<00:00, 59.29it/s]
Epoch 8 Loss: 0.0973
100%|██████████| 782/782 [00:13<00:00, 59.41it/s]
Epoch 9 Loss: 0.1096
100%|██████████| 782/782 [00:13<00:00, 58.94it/s]Epoch 10 Loss: 0.0803
```

```
def show_images(gray, output, target):

    gray = gray[:6].cpu()
    output = output[:6].cpu()
    target = target[:6].cpu()

    output = (output + 1) / 2
    target = (target + 1) / 2

    fig, ax = plt.subplots(3,6, figsize=(12,6))

    for i in range(6):
        ax[0,i].imshow(gray[i].permute(1,2,0).squeeze(), cmap='gray')
        ax[1,i].imshow(output[i].permute(1,2,0))
        ax[2,i].imshow(target[i].permute(1,2,0))

        ax[0,i].axis('off')
        ax[1,i].axis('off')
        ax[2,i].axis('off')

    ax[0,0].set_ylabel("Input")
    ax[1,0].set_ylabel("Output")
    ax[2,0].set_ylabel("Target")

    plt.show()
```

```
imgs,_ = next(iter(loader))
imgs = imgs.to(device)
gray = to_gray(imgs)

with torch.no_grad():
    out = model(gray)

show_images(gray, out, imgs)
```



```
import torch
from diffusers import StableDiffusionInstructPix2PixPipeline, EulerAncestralDiscreteScheduler
from PIL import Image
import matplotlib.pyplot as plt
import os

# ========================================
# 1. USER SETTINGS
# ========================================
INPUT_IMAGE = "/content/WhatsApp Image 2026-01-15 at 23.40.09 (1).jpeg"  # Ensure this file exists in the folder

# List of prompts to generate
PROMPT_LIST = [
    "make me sit in open mountains",
    "make me sit in a modern classroom",
    "turn the background into a cyberpunk city",
    "make it look like a pencil sketch",
    "Make me fly through sahara desert",
    "Make 2 balloons grow on the head",
    "Make me sit in cafe drinking coffee",
    "Make me study under the moon",
    "Make me hospitalized in hospital bned",
    "Make me drive a supercar",
    "Make me ride a superbike"

]

# Tuning Knobs
IMAGE_STABILITY = 2  # 1.5 is balanced. Higher = keeps face better.
TEXT_STRENGTH = 10.0    # How strongly to apply the prompt.
```

```
# ========================================
# 2. Setup Model & Hardware
# ========================================
device = "cuda" if torch.cuda.is_available() else "cpu"
dtype = torch.float16 if device == "cuda" else torch.float32

print(f"--- Multi-Prompt Generator (Running on {device}) ---")

# Load Model
model_id = "timbrooks/instruct-pix2pix"
pipe = StableDiffusionInstructPix2PixPipeline.from_pretrained(
    model_id, torch_dtype=dtype, safety_checker=None
).to(device)
pipe.scheduler = EulerAncestralDiscreteScheduler.from_config(pipe.scheduler.config)
```

```
Flax classes are deprecated and will be removed in Diffusers v1.0.0. We recommend migrating to PyTorch classes or pinning your version of Diffusers.
Flax classes are deprecated and will be removed in Diffusers v1.0.0. We recommend migrating to PyTorch classes or pinning your version of Diffusers.
--- Multi-Prompt Generator (Running on cuda) ---
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart yo
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
Loading pipeline components...: 100%                                    6/6 [00:14<00:00,  2.08s/it]
Loading weights: 100%                                    196/196 [00:02<00:00, 149.75it/s, Materializing param=text_model.final_layer_norm.weight]
CLIPTextModel LOAD REPORT from: /root/.cache/huggingface/hub/models--timbrooks--instruct-pix2pix/snapshots/31519b5cb02a7fd89b906d88731cd4d6a7bbf88d/text_encoder
Key                               | Status     |  |
----------------------------------+------------+--+-
text_model.embeddings.position_ids | UNEXPECTED |  |

Notes:
- UNEXPECTED    :can be ignored when loading from different task/architecture; not ok if you expect identical arch.
Loaded image. Processing 11 prompts...

🎨 Generating 1/11: 'make me sit in open mountains'...
100%                                    40/40 [00:08<00:00,  4.82it/s]
```

Original

Result: make me sit in open mountains

```
✅ Saved to output_0_make_me_si.png
----------------------------------------
🎨 Generating 2/11: 'make me sit in a modern classroom'...
100%                                    40/40 [00:08<00:00,  4.68it/s]
```

Original

Result: make me sit in a modern classroom

✅ Saved to output_1_make_me_si.png

🎨 Generating 3/11: 'turn the background into a cyberpunk city'...

100% 40/40 [00:08<00:00, 4.56it/s]

Original

Result: turn the background into a cyberpunk city

✅ Saved to output_2_turn_the_b.png

🎨 Generating 4/11: 'make it look like a pencil sketch'...

100% 40/40 [00:08<00:00, 4.49it/s]

Original

Result: make it look like a pencil sketch