🧨 diffusers for Democratizing Diffusion Models

IISc, June 11 2023

Sayak Paul, Hugging Face 🤗
@RisingSayak

Presenting this work on behalf of the Diffusers team (past and current) and the dear community 🤗





Plan of attack

(1) Generative models - a brief intro

(2) 🧨 diffusers for image generation and beyond

(3) Making 🧨 diffusers research-friendly

Feel free to interrupt with questions anytime :)

Generative Models

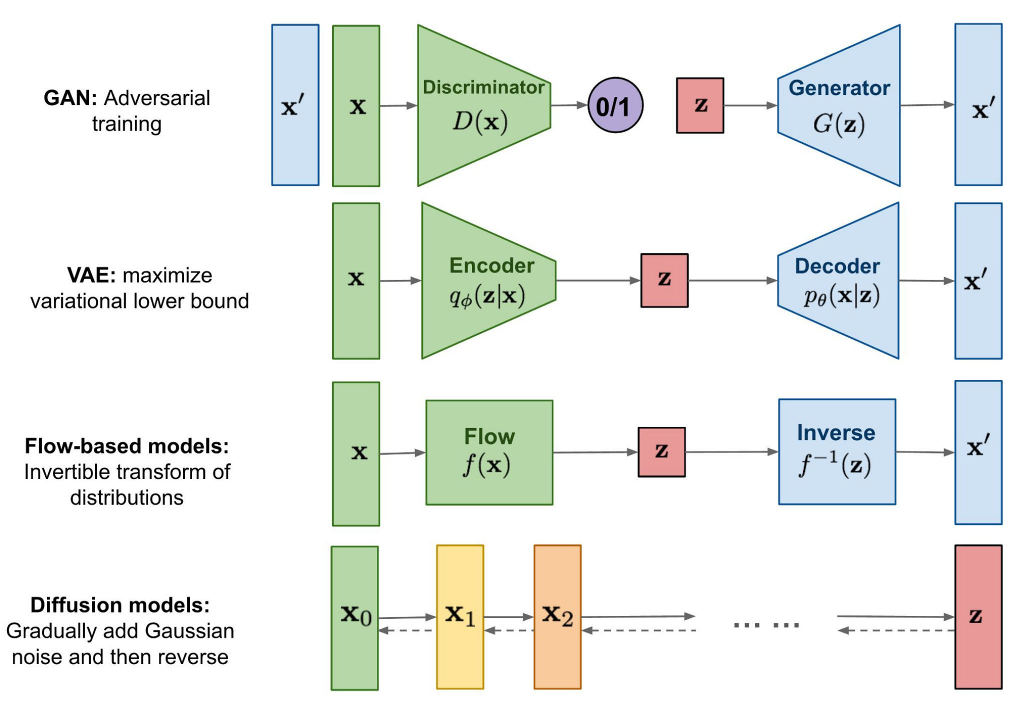


Image from “What are Diffusion Models” by Lilian Weng

Generative Models

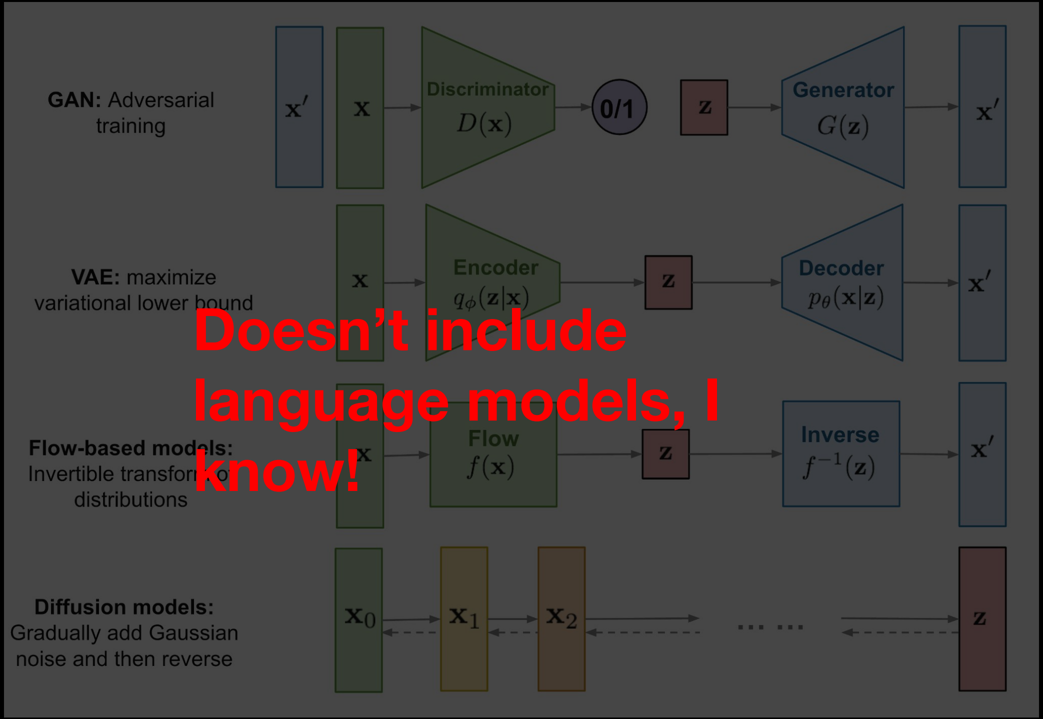


Image from "What are Diffusion Models" by Lilian Weng

Generative Models

Today's focus

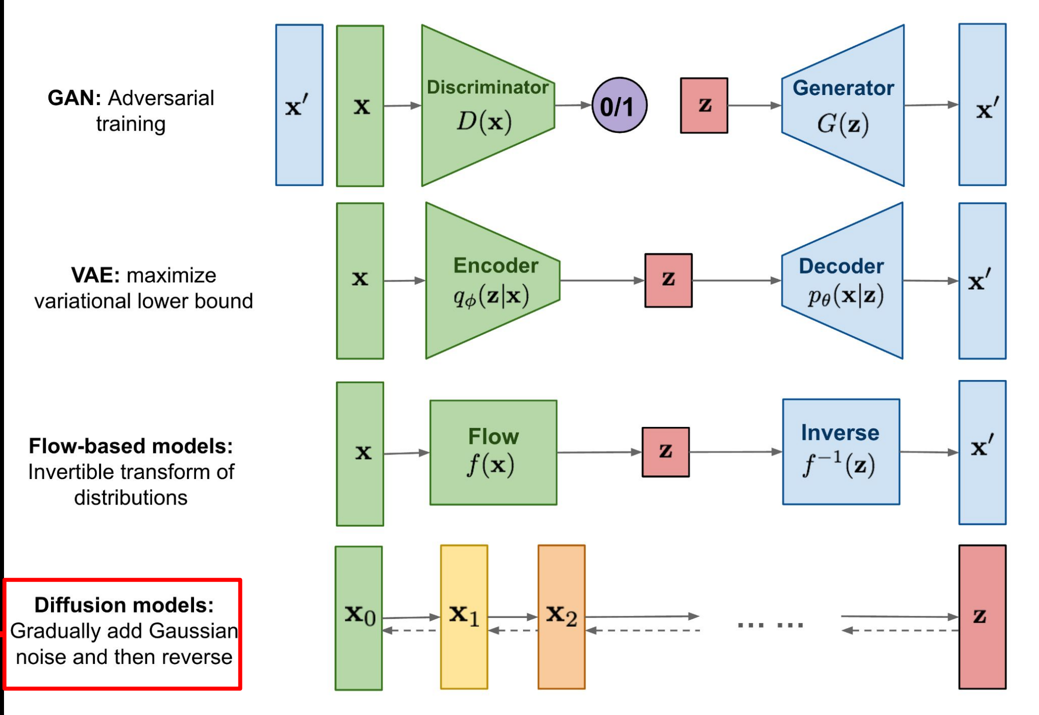
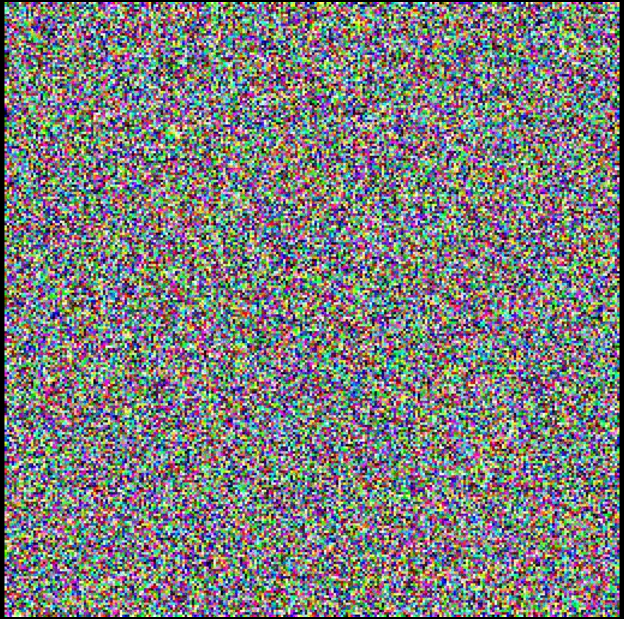


Image from "What are Diffusion Models" by Lilian Weng

For the next couple of slides, we will concentrate on “image” diffusion models 🌠

Diffusion models

What happens when you refine a noise vector to become a realistic image?



adityaramesh.com/posts/dalle2/dalle2.html

Diffusion models

What happens when you refine a noise vector to become a realistic image?

https://nvlabs.github.io/denoising-diffusion-gan/

Diffusion models

When you “condition” the denoising process with text:

DALL-E 2 prompt: “A photo of a white fur monster standing in a purple room”

Diffusion models – the path

- Deep unsupervised learning using nonequilibrium thermodynamics (2015)
- Denoising Diffusion Probabilistic Models (2020)
- Denoising Diffusion Implicit Models (2020)
- Diffusion Models Beat GANs on Image Synthesis (2021)
- Classifier-Free Diffusion Guidance (2021)
- GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models (2022)
- High-Resolution Image Synthesis with Latent Diffusion Models (2022)[1]
- Elucidating the Design Space of Diffusion-Based Generative Models (2022)[2]
- Hierarchical Text-Conditional Image Generation with CLIP Latents (2022)[3]
- Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding (2022)[4]

1 - Scaled to Stable Diffusion
2 - The “Karras paper”
3 - DALLE-2
4 - Imagen

Some popular diffusion models for images

- DALL-E 2 (OpenAI)
- Stable Diffusion (Stability AI, CompVis, RunwayML, LAION)
- Imagen (Google)
- IF (DeepFloyd)
- Kandinsky (AI Forever)

But …

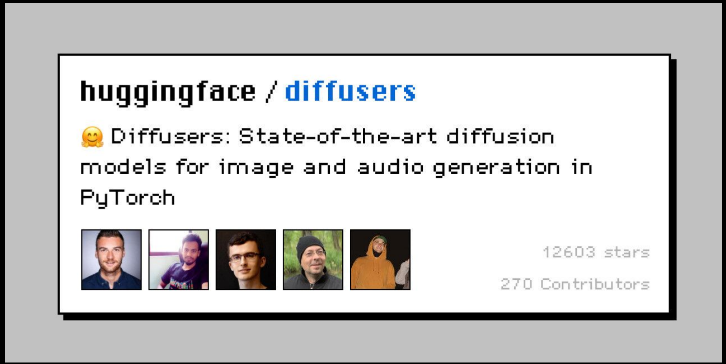
Not all these models are open-source!

- Stable Diffusion - ✅
- Stable unCLIP (Stability AI version) - ✅
- unCLIP (Kakao Brain version) - ✅
- IF (Imagen-like model from DeepFloyd and Stability AI) - ✅
- Kandinsky - ✅
- DALL-E 2 - ❌
- Imagen - ❌

Why make them open?

- Study the risk factors and failure cases
- Evaluate safety measurements
- Build on top of them
- Improve them

🧨 diffusers



https://github.com/huggingface/diffusers

A Python library maintained at 🤗

- Providing open and responsible access to pre-trained diffusion models.
- Democratizing the ecosystem of diffusion models by making them easy to use.

Text-to-image with 🧨 diffusers

```
from diffusers import StableDiffusionPipeline

model_id = "runwayml/stable-diffusion-v1-5"
pipeline = StableDiffusionPipeline.from_pretrained(model_id)
pipeline = pipeline.to("cuda")

image = pipeline("An astronaut riding a tiger").images[0]
image.save("image.png")
```

https://hf.co/docs/diffusers/api/pipelines/stable_diffusion/overview

Text-to-image with 🧨 diffusers

Striving for photorealism

```python
from diffusers import DiffusionPipeline
import torch

# prior model
pipe_prior = DiffusionPipeline.from_pretrained(
    "kandinsky-community/kandinsky-2-1-prior", torch_dtype=torch.float16
)
pipe_prior.to("cuda")

# text-to-image model
t2i_pipe = DiffusionPipeline.from_pretrained(
    "kandinsky-community/kandinsky-2-1", torch_dtype=torch.float16
)
t2i_pipe.to("cuda")

prompt = "A car exploding into colorful dust"
image_embeds, negative_image_embeds = pipe_prior(prompt,
generator=generator).to_tuple()
image = t2i_pipe(
    prompt, image_embeds=image_embeds, negative_image_embeds=negative_image_embeds
).images[0]
image.save("image.png")
```

https://hf.co/docs/diffusers/main/en/api/pipelines/kandinsky

Image variations with Stable unCLIP and 🧨 diffusers

```
from diffusers import StableUnCLIPImg2ImgPipeline
from diffusers.utils import load_image
import torch

pipe = StableUnCLIPImg2ImgPipeline.from_pretrained(
    "stabilityai/stable-diffusion-2-1-unclip",
    torch_dtype=torch.float16,
    variation="fp16"
)

init_image = load_image(<image_url>)
images = pipe(init_image, num_images_per_prompt=3).images
```

https://hf.co/docs/diffusers/api/pipelines/stable_unclip

Text-to-video with 🧨 diffusers

```
import torch
import imageio
from diffusers import TextToVideoZeroPipeline

pipe = TextToVideoZeroPipeline.from_pretrained(
    "runwayml/stable-diffusion-v1-5",
    torch_dtype=torch.float16
)

prompt = "A panda is playing guitar on times square"
result = pipe(prompt=prompt).images
result = [(r * 255).astype("uint8") for r in result]
imageio.mimsave("video.gif", result, "GIF", fps=4)
```

https://hf.co/docs/diffusers/api/pipelines/text_to_video_zero

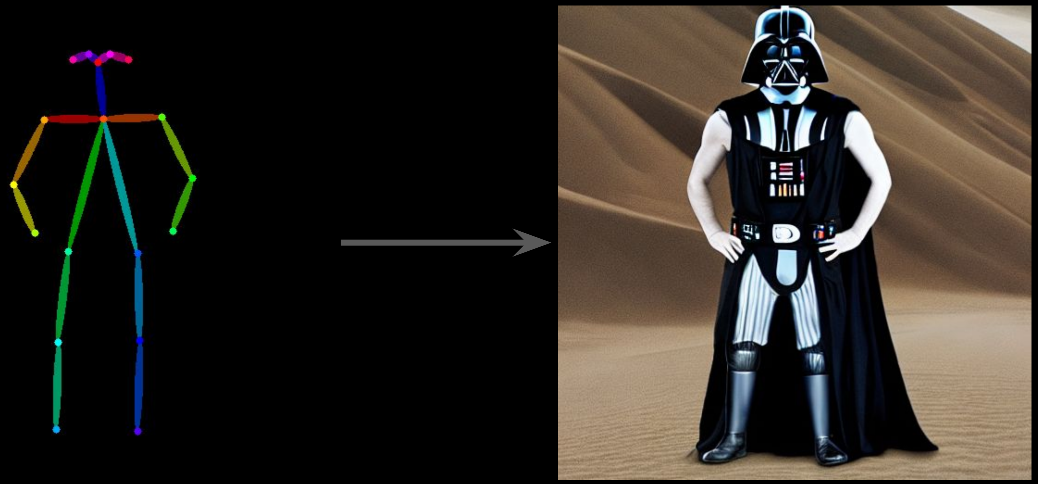
Community's favorite: ControlNet 🎨

```python
from diffusers import StableDiffusionControlNetPipeline, ControlNetModel

controlnet = ControlNetModel.from_pretrained("lllyasviel/sd-controlnet-openpose")
pipe = StableDiffusionControlNetPipeline.from_pretrained(
    "runwayml/stable-diffusion-v1-5", controlnet=controlnet
)

prompt = "Darth Vader dancing in a desert"
image = pipe(prompt, image=openpose_image).images[0]
```

https://hf.co/docs/diffusers/api/pipelines/stable_diffusion/controlnet

Community's favorite: ControlNet 🎨

"Darth Vader dancing in a desert"

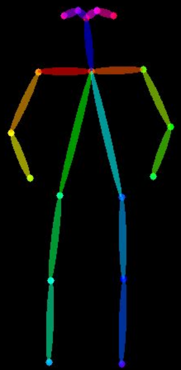


Community's favorite: ControlNet 🎨

"a giant standing in a fantasy landscape, best quality"

Canny map

+



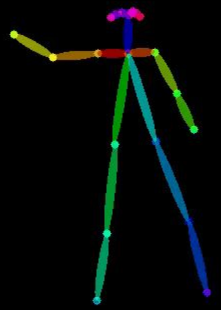
Pose

=

Final image

ControlNet + Video 🎨🎥

"Darth Vader dancing in a desert"

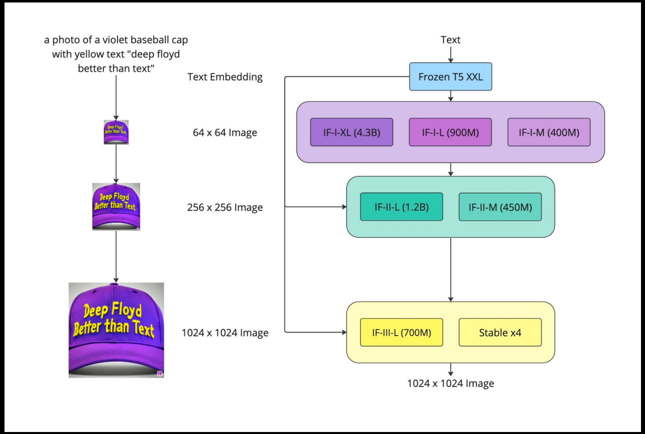


https://hf.co/docs/diffusers/main/en/api/pipelines/text_to_video_zero

You also asked for models that can spell characters well :)

Fresh off the press - IF

Running a **43.2 GB system** in a *free-tier Colab Notebook* (T4 GPU)



Taken from
https://hf.co/DeepFloyd/IF-I-XL-v1.0

https://hf.co/blog/if

Exploring 🧨 diffusers

"🤗 Diffusers is the go-to library for state-of-the-art pretrained diffusion models for generating images, audio, and even 3D structures of molecules. Whether you're looking for a simple inference solution or want to train your own diffusion model, 🤗 Diffusers is a modular toolbox that supports both. Our library is designed with a focus on usability over performance, simple over easy, and customizability over abstractions." - https://hf.co/docs/diffusers

Exploring 🧨 diffusers

Various pipelines exploration:

- Image translation (think of CycleGAN like stuff)
- Text to video generation
- Latent space manipulation
- Image editing with human-readable instructions
- Semantic guidance
- and more (including AUDIO pipelines)!

https://hf.co/docs/diffusers/main/en/api/pipelines

Exploring 🧨 diffusers

Various pipelines exploration:

- Image translation (think of CycleGAN like stuff)
- Text to video generation
- Latent space manipulation
- Image editing with human-readable instructions
- Semantic guidance
- and more (including AUDIO pipelines)!

→ `DiffusionPipeline`

https://hf.co/docs/diffusers/main/en/api/pipelines

Exploring 🧨 diffusers

Swapping components of a pipeline for rapid experimentation:

```
import torch
from diffusers import StableDiffusionPipeline, UniPCMultistepScheduler

pipe = StableDiffusionPipeline.from_pretrained(
    "runwayml/stable-diffusion-v1-5",
    torch_dtype=torch.float16
)
pipe.scheduler = UniPCMultistepScheduler.from_config(pipe.scheduler.config)

prompt = "A panda is playing guitar on times square"
result = pipe(prompt=prompt).images
```

New scheduler

https://hf.co/docs/diffusers/using-diffusers/schedulers

Exploring 🧨 diffusers

Swapping components of a pipeline for rapid experimentation:

```
import torch
from diffusers import StableDiffusionPipeline, UNet2DConditionModel

unet = UNet2DConditionModel.from_pretrained(
    "valhalla/sd-pokemon-model",
    subfolder="unet",
    torch_dtype=torch.float16
)
pipe = StableDiffusionPipeline.from_pretrained(
    "CompVis/stable-diffusion-v1-4",
    unet=unet,
    torch_dtype=torch.float16
)

prompt = "cute Sundar Pichai character"
result = pipe(prompt=prompt).images
```

New UNet

https://hf.co/docs/diffusers/using-diffusers/loading

Exploring 🧨 diffusers

"cute Sundar Pichai character"

Exploring 🧨 diffusers

Train your own models 🔥

- community
- controlnet
- custom_diffusion
- dreambooth
- inference
- instruct_pix2pix
- research_projects
- rl
- text_to_image
- textual_inversion
- unconditional_image_generation

https://github.com/huggingface/diffusers/tree/main/examples/

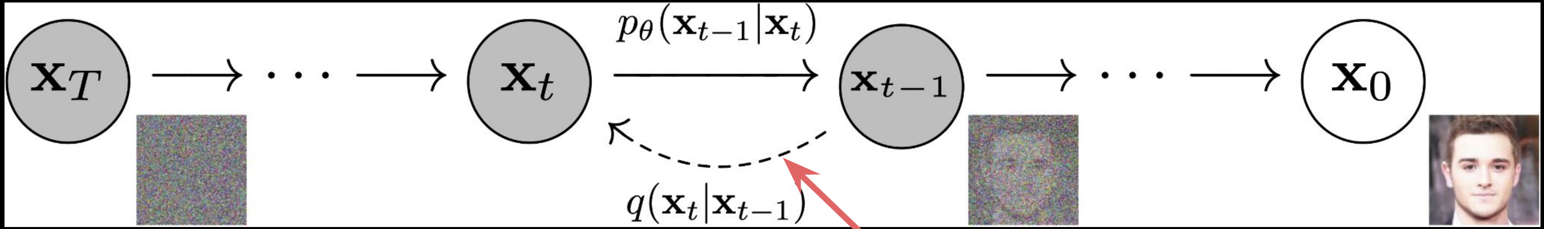
Modular design for research

Training for unconditional generation:

Sample dataset

Modular design for research

Training for unconditional generation:



To learn

Image from the DDPM paper.

Modular design for research

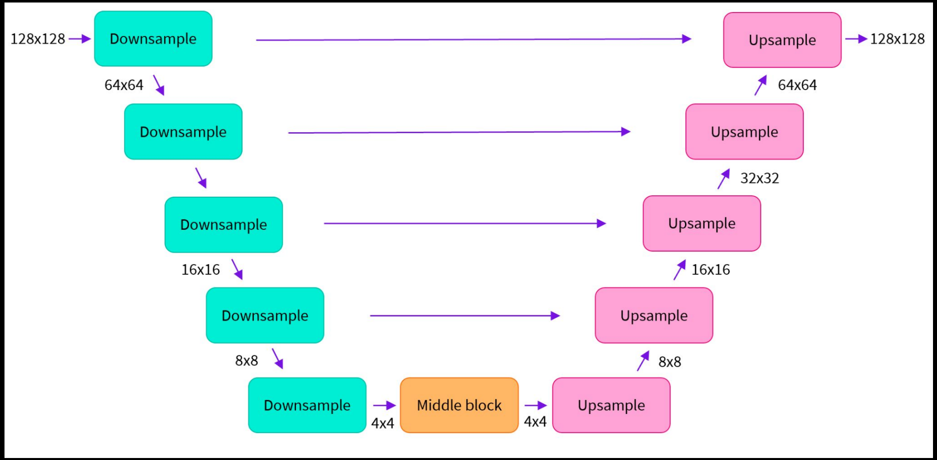
We add noise to an image according to a <u>noise schedule</u>:

```
noise_scheduler = DDPMScheduler(num_train_timesteps=1000)

noise = torch.randn(sample_image.shape)
timesteps = torch.LongTensor([50])

noisy_image = noise_scheduler.add_noise(sample_image, noise, 
timesteps)
noisy_image = ((noisy_image.permute(0, 2, 3, 1) + 1.0) * 127.5)
Image.fromarray(noisy_image.type(torch.uint8).numpy()[0])
```

Modular design for research

We need a model (neural net) to predict the less noisy image:

```
from diffusers import UNet2DModel

model = UNet2DModel(...)
```



Modular design for research

A minimal training loop:

```
for epoch in range(epochs):
    for clean_image_batch in dataset:
        # Sample noise to add to the images.
        noise = torch.randn(clean_image_batch.shape).to(clean_image_batch)
        bs = clean_image_batch.shape[0]

        # Sample a random timestep for each image.
        timesteps = torch.randint(0, noise_scheduler.num_train_timesteps, (bs,))

        # Add noise to the clean images according to the noise magnitude at
        # each timestep (this is the forward diffusion process).
        noisy_images = noise_scheduler.add_noise(clean_image_batch, noise, timesteps)

        # Predict the noise residual
        noise_pred = model(noisy_images, timesteps, return_dict=False)[0]
        loss = F.mse_loss(noise_pred, noise)

        # Backprop.
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
```

Modular design for research

And voila!

Full notebook: https://github.com/huggingface/notebooks/blob/main/diffusers/training_example.ipynb

Modular design for research

Training a text-conditioned latent space diffusion model:

- The dataset will have image-prompt pairs:

image (image)	text (string)
	"a drawing of a green pokemon with red eyes"
	"a green and yellow toy with a red nose"
	"a red and white ball with an angry look on its face"
	"a cartoon ball with a smile on it's face"
	"a bunch of balls with faces drawn on them"

https://hf.co/datasets/lambdalabs/pokemon-blip-captions

Modular design for research

Training a text-conditioned latent space diffusion model:

- We need ways to:
  - Embed the images
  - Embed the prompts
  - Use a UNet to pass BOTH image and text embeddings for denoising

Modular design for research

A minimal training loop:

```
for epoch in range(epochs):
    for images, prompts in dataset:
        # Convert images to latent space
        latents = vae.encode(batch["pixel_values"]).latent_dist.sample()

        # Sample noise to add to the images.
        noise = torch.randn_like(latents)
        bs = noise.shape[0]

        # Sample a random timestep for each image.
        timesteps = torch.randint(0, noise_scheduler.num_train_timesteps, (bs,))

        # Add noise to the image latents according to the noise magnitude at each timestep
        # (this is the forward diffusion process).
        noisy_images = noise_scheduler.add_noise(latents, noise, timesteps)
```

Modular design for research

A minimal training loop:

```
for epoch in range(epochs):
    for images, prompts in dataset:
        # Convert images to latent space
        latents = vae.encode(batch["pixel_values"]).latent_dist.sample()

        # Sample noise to add to the images.
        noise = torch.randn_like(latents)
        bs = noise.shape[0]

        # Sample a random timestep for each image.
        timesteps = torch.randint(0, noise_scheduler.num_train_timesteps, (bs,))

        # Add noise to the image latents according to the noise magnitude at each timestep
        # (this is the forward diffusion process).
        noisy_images = noise_scheduler.add_noise(latents, noise, timesteps)
```

Modular design for research

A minimal training loop:

```
for epoch in range(epochs):
    for images, prompts in dataset:
        ...

        # Compute text embeddings.
        text_embeddings = text_encoder(prompts)[0]

        # Predict the noise residual
        model_pred = unet(noisy_latents, timesteps, text_embeddings).sample
        loss = F.mse_loss(model_pred, noise)

        # Backprop.
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
```

Modular design for research

A minimal training loop:

```
for epoch in range(epochs):
    for images, prompts in dataset:
        ...

        # Compute text embeddings.
        text_embeddings = text_encoder(prompts)[0]

        # Predict the noise residual
        model_pred = unet(noisy_latents, timesteps, text_embeddings).sample
        loss = F.mse_loss(model_pred, noise)

        # Backprop.
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
```

Modular design for research

And voila!

"cute Sundar Pichai character"

Full example: https://github.com/huggingface/diffusers/blob/main/examples/text_to_image

Modular design for research

Prediction targets can be configured:

```
if noise_scheduler.config.prediction_type == "epsilon":
    target = noise
elif noise_scheduler.config.prediction_type == "v_prediction":
    target = noise_scheduler.get_velocity(latents, noise, timesteps)

loss = F.mse_loss(model_pred, target)
```

Modular design for research

There many more features in our training examples:

- Faster convergence with Min-SNR
- Offset noise for learning better contrast and brightness
- Noise perturbation
- Qualitative validation

Other good-to-have features

- Support for distributed training with 🤗 accelerate
- Memory optimization:
  - Easy FP16 training
  - Memory-efficient attention
  - Attention slicing
  - VAE tiling
  - LoRA for parameter-efficient fine-tuning

Docs: https://huggingface.co/docs/diffusers/main/en/optimization/fp16

Some implementations for reference

The following works are built on top of 🧨 diffusers that perform training:

- Tune-A-Video: One-Shot Tuning of Image Diffusion Models for Text-to-Video Generation, Wu et al., 2022.
- Training Diffusion Models with Reinforcement Learning, Black et al., 2023.
- Instruction-tuning Stable Diffusion, Paul et al., 2023.

Inference-time optimizations are also easy

- Training-free improvements to Diffusion systems:
    - Attend and Excite
    - Zero-shot Image Translation
    - Semantic Guidance
- All of these are subclassed from `DiffusionPipeline`.
- Refer to the source code of these pipelines to know more.
- Community pipelines reference:

  https://hf.co/docs/diffusers/main/en/using-diffusers/contribute_pipeline

IF prompt: A cute panda standing amidst a mountain and holding a placard saying “Thank you!”

SCAN ME