Text to image

- 60-90 min: Non-technical
    - Try out different models
    - Prompting
    - Bias, limitations and controversy
- 150-180 min: Understand how some of the models work behind the scenes
    - Diffusion
        - Conditioned models
    - Stable Diffusion
        - CLIP
        - U-Net
    - Textual Inversion
    - InstructPix2Pix

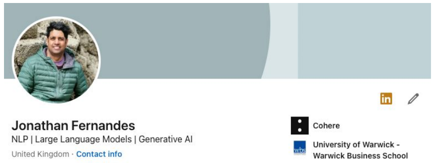
Live Course

Hands-On Transformers for Computer Vision

With Jonathan Fernandes

July 19 • 5pm - 9pm BST

Next iteration of this course will include the latest AI generation



Jonathan Fernandes

NLP | Large Language Models | Generative AI

United Kingdom · Contact info

Cohere

University of Warwick - Warwick Business School

What is diffusion?

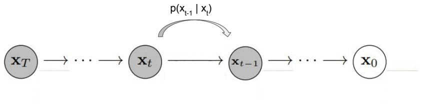
Text to image - try it out (Colab notebook)

What is diffusion?

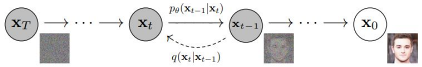
What if we could reverse this process?

Markov chains

A Markov chain is a mathematical system that experiences transitions from one state to another according to certain probabilistic rules. The probability of transitioning to any particular state is dependent solely on the current state and time elapsed.



Diffusion model


$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_{t-1}$
$\mathbf{x}_0$

← Forward - adding noise

→ Reverse - de-noising

Source: Denoising Diffusion Probabilistic Models (Ho et al)

3 key components

- Pipelines - high-level wrappers that make it easy to use the functions.

3 key components

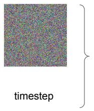
- Pipelines - high-level wrappers that make it easy to use the functions.
- Models - UNet

3 key components

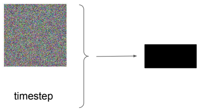
- Pipelines - high-level wrappers that make it easy to use the functions.
- Models - UNet
- Schedulers - the method for iteratively adding noise to an image
  - Why different schedulers?

Colab notebook - diffusion

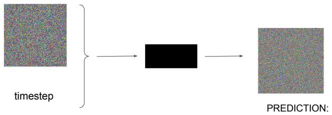
Diffusion models (inference)



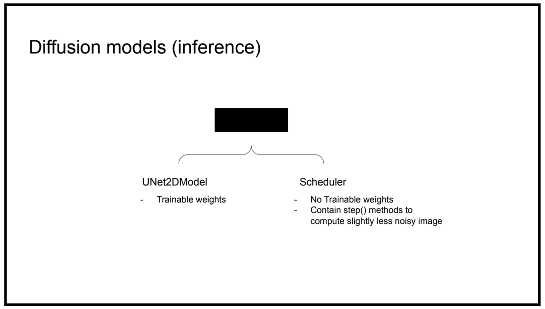
Diffusion models (inference)



Diffusion models (inference)



PREDICTION:
- Slightly less noisy image OR
- Difference between slightly less noisy image and input image OR
- ...



Diffusion models (inference)

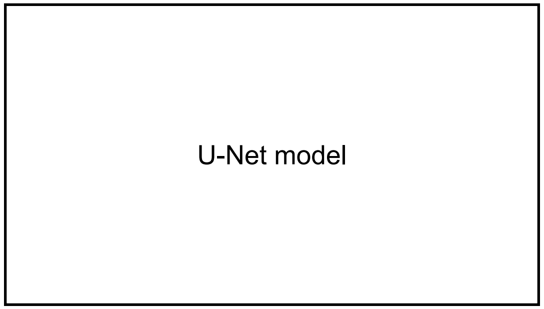
[redacted]

UNet2DModel

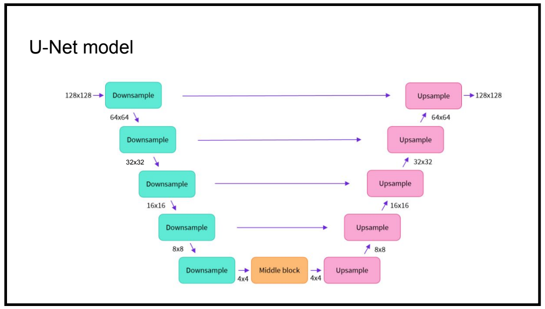
- Trainable weights

Scheduler

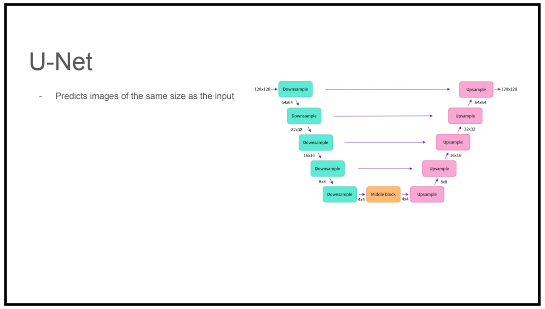
- No Trainable weights
- Contain step() methods to compute slightly less noisy image



U-Net model

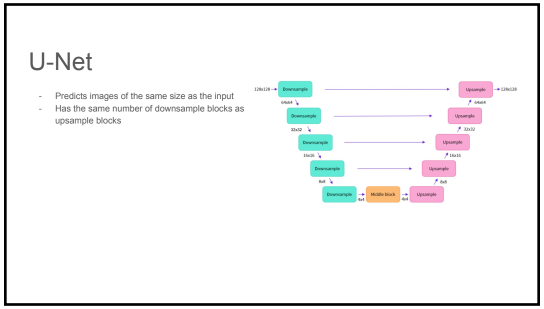


U-Net model



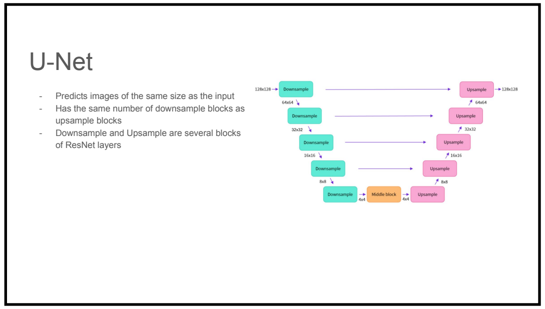
U-Net

- Predicts images of the same size as the input



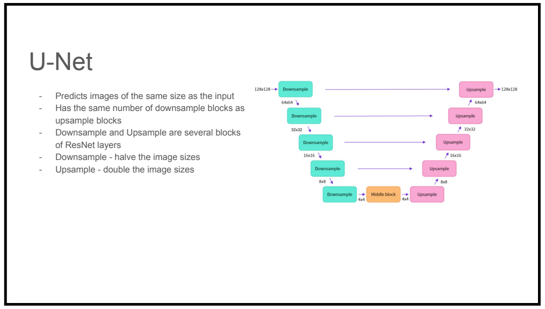
U-Net

- Predicts images of the same size as the input
- Has the same number of downsample blocks as upsample blocks



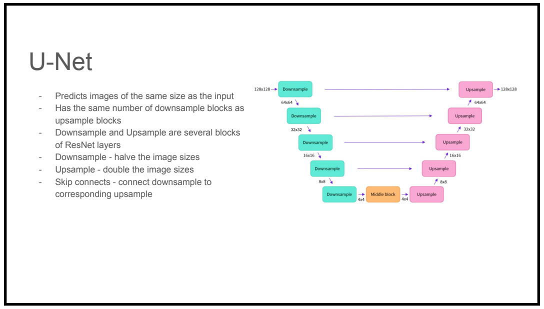
U-Net

- Predicts images of the same size as the input
- Has the same number of downsample blocks as upsample blocks
- Downsample and Upsample are several blocks of ResNet layers



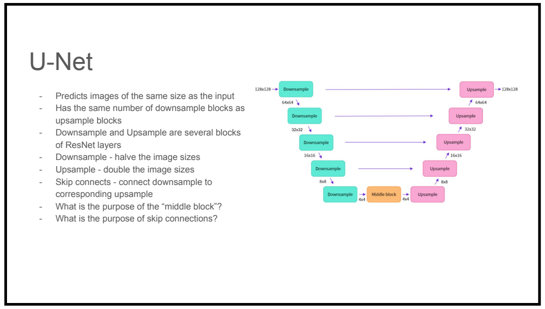
U-Net

- Predicts images of the same size as the input
- Has the same number of downsample blocks as upsample blocks
- Downsample and Upsample are several blocks of ResNet layers
- Downsample - halve the image sizes
- Upsample - double the image sizes



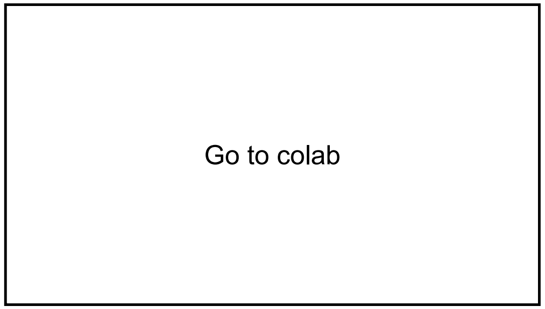
U-Net

- Predicts images of the same size as the input
- Has the same number of downsample blocks as upsample blocks
- Downsample and Upsample are several blocks of ResNet layers
- Downsample - halve the image sizes
- Upsample - double the image sizes
- Skip connects - connect downsample to corresponding upsample

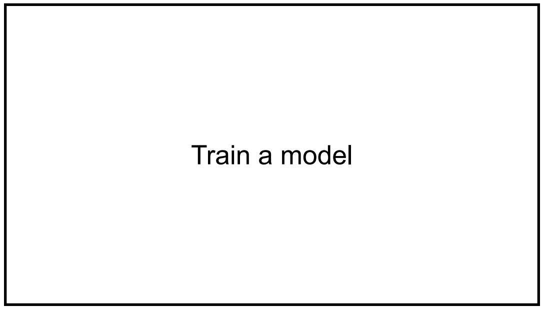


U-Net

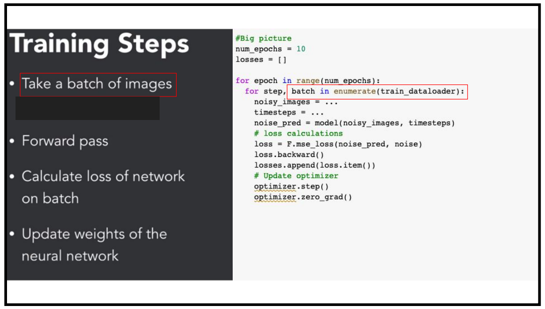
- Predicts images of the same size as the input
- Has the same number of downsample blocks as upsample blocks
- Downsample and Upsample are several blocks of ResNet layers
- Downsample - halve the image sizes
- Upsample - double the image sizes
- Skip connects - connect downsample to corresponding upsample
- What is the purpose of the “middle block”?
- What is the purpose of skip connections?



Go to colab



Train a model

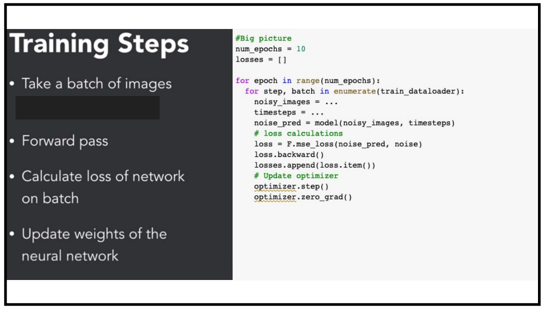


Training Steps

- Take a batch of images
- Forward pass
- Calculate loss of network on batch
- Update weights of the neural network

```
#Big picture
num_epochs = 10
losses = []

for epoch in range(num_epochs):
  for step, batch in enumerate(train_dataloader):
    noisy_images = ...
    timesteps = ...
    noise_pred = model(noisy_images, timesteps)
    # loss calculations
    loss = F.mse_loss(noise_pred, noise)
    loss.backward()
    losses.append(loss.item())
    # Update optimizer
    optimizer.step()
    optimizer.zero_grad()
```

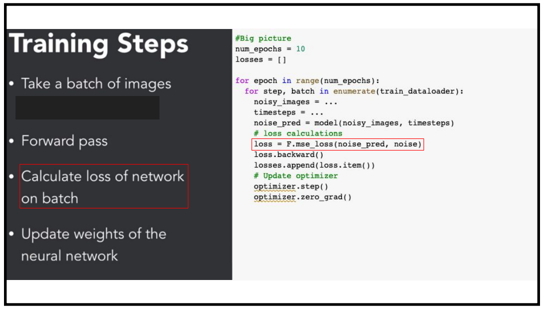


Training Steps

- Take a batch of images
- Forward pass
- Calculate loss of network on batch
- Update weights of the neural network

```
#Big picture
num_epochs = 10
losses = []

for epoch in range(num_epochs):
  for step, batch in enumerate(train_dataloader):
    noisy_images = ...
    timesteps = ...
    noise_pred = model(noisy_images, timesteps)
    # loss calculations
    loss = F.mse_loss(noise_pred, noise)
    loss.backward()
    losses.append(loss.item())
    # Update optimizer
    optimizer.step()
    optimizer.zero_grad()
```

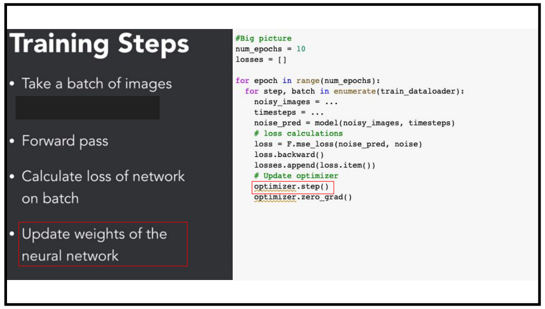


Training Steps

- Take a batch of images
- Forward pass
- Calculate loss of network on batch
- Update weights of the neural network

```
#Big picture
num_epochs = 10
losses = []

for epoch in range(num_epochs):
  for step, batch in enumerate(train_dataloader):
    noisy_images = ...
    timesteps = ...
    noise_pred = model(noisy_images, timesteps)
    # loss calculations
    loss = F.mse_loss(noise_pred, noise)
    loss.backward()
    losses.append(loss.item())
    # Update optimizer
    optimizer.step()
    optimizer.zero_grad()
```

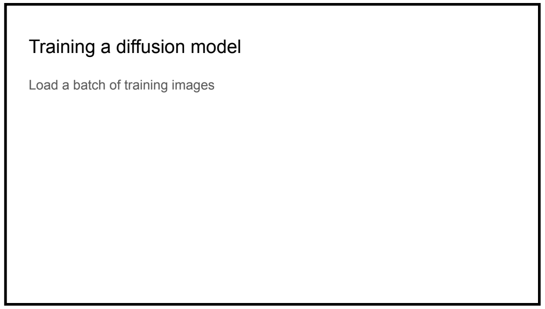


Training Steps

- Take a batch of images
- Forward pass
- Calculate loss of network on batch
- Update weights of the neural network

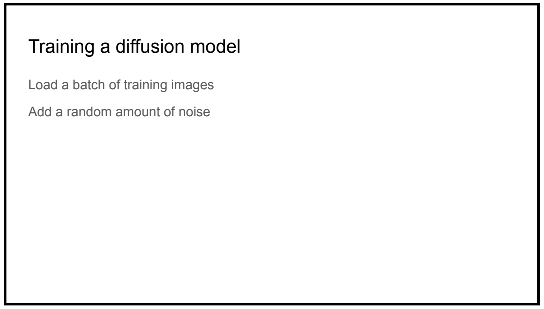
```
#Big picture
num_epochs = 10
losses = []

for epoch in range(num_epochs):
  for step, batch in enumerate(train_dataloader):
    noisy_images = ...
    timesteps = ...
    noise_pred = model(noisy_images, timesteps)
    # loss calculations
    loss = F.mse_loss(noise_pred, noise)
    loss.backward()
    losses.append(loss.item())
    # Update optimizer
    optimizer.step()
    optimizer.zero_grad()
```



Training a diffusion model

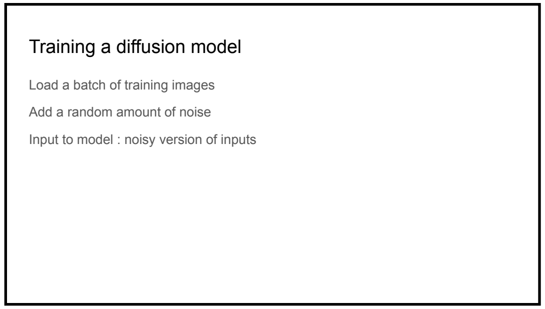
Load a batch of training images



Training a diffusion model

Load a batch of training images

Add a random amount of noise



Training a diffusion model

Load a batch of training images

Add a random amount of noise

Input to model : noisy version of inputs

Training a diffusion model

Load a batch of training images

Add a random amount of noise

Input to model : noisy version of inputs

Use the loss to determine how well the model does at de-noising the inputs

Training a diffusion model

Load a batch of training images

Add a random amount of noise

Input to model : noisy version of inputs

Use the loss to determine how well the model does at de-noising the inputs

Update the model weights

Training a diffusion model

Load a batch of training images

Add a random amount of noise

Input to model : noisy version of inputs

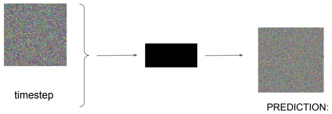
Use the loss to determine how well the model does at de-noising the inputs

Update the model weights

Train a model (colab notebook)

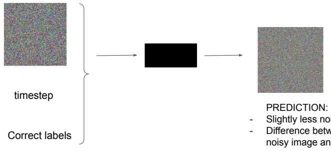
Conditioned models

Diffusion models (training)



PREDICTION:
- Slightly less noisy image OR
- Difference between slightly less noisy image and input image OR
- …

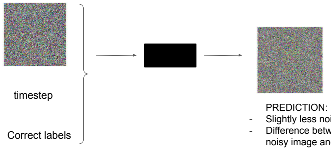
Conditioned Diffusion models (training)



PREDICTION:
- Slightly less noisy image OR
- Difference between slightly less noisy image and input image OR
- …

Conditioned Diffusion models (inference)

Inference: We pass the labels we want and the model should generate images that match.



PREDICTION:
- Slightly less noisy image OR
- Difference between slightly less noisy image and input image OR
- …

Conditioned models

- Add additional channels in the input to the Unet
- Add cross-attention layers that can attend to a sequence passed.
    - Conditioning is text
    - Stable Diffusion uses this

Add additional channels in the input to the Unet

- UNet2DModel with additional input channels

Add additional channels in the input to the Unet

- UNet2DModel with additional input channels
- Map class labels (expected output == conditioned) to a learned vector e.g. an embedding layer

Add additional channels in the input to the Unet

- UNet2DModel with additional input channels
- Map class labels (expected output == conditioned) to a learned vector e.g. an embedding layer
- Add this information as extra channels for the Unet input

Add additional channels in the input to the Unet

- UNet2DModel with additional input channels
- Map class labels (expected output == conditioned) to a learned vector e.g. an embedding layer
- Add this information as extra channels for the Unet input
- Submit this as input to UNet2DModel as before, with timestep.

Conditioned models - adding additional channels
(colab notebook)

Add additional channels in the input to the Unet

- UNet2DModel with additional input channels
- Map class labels to a learned vector (embedding layer)
- Add this information as extra channels for the Unet input
- Submit this as input to UNet2DModel as before, with timestep.

REVIEW: This is what we have done in the notebook

Stable Diffusion

What makes this different to other text to image solutions?

- DALL-E
- DALLE-2
- Imagen

Can run on commodity hardware

Model and training details

Model

Text encoder - CLIP ViT-L/14

UNet = 860M parameter model

Autoencoder - downsampling factor of 8.

The model was pretrained on 256x256 images and then finetuned on 512x512 images.

Training time

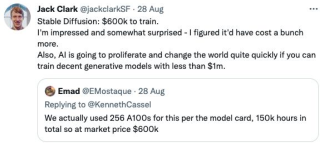
- Hardware Type: A100 PCIe 40GB
- Hours used: 150000

Training data

The core dataset was trained on LAION-Aesthetics, a soon to be released subset of LAION 5B.

LAION-Aesthetics was created with a new CLIP-based model that filtered LAION-5B based on how "beautiful" an image was, building on ratings from the alpha testers of Stable Diffusion.

Cost



Jack Clark @jackclarkSF · 28 Aug

Stable Diffusion: $600k to train.
I'm impressed and somewhat surprised - I figured it'd have cost a bunch more.
Also, AI is going to proliferate and change the world quite quickly if you can train decent generative models with less than $1m.

> **Emad** @EMostaque · 28 Aug
> Replying to @KennethCassel
> We actually used 256 A100s for this per the model card, 150k hours in total so at market price $600k

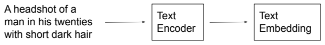
Controversy

- Image regurgitation
- Copying artist styles
  - Getty Images
  - Shutterstock

Applications

https://www.reddit.com/r/StableDiffusion/comments/wyduk1/show_rstablediffusion_integrating_sd_in_photoshop/

What components do we need?

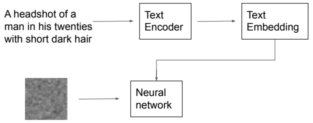


A headshot of a man in his twenties with short dark hair

Text Encoder

Text Embedding

What components do we need?



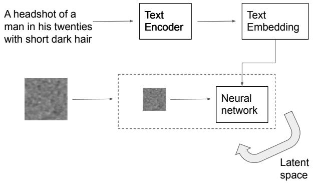
A headshot of a man in his twenties with short dark hair

Text Encoder

Text Embedding

Neural network

What components do we need?



A headshot of a man in his twenties with short dark hair

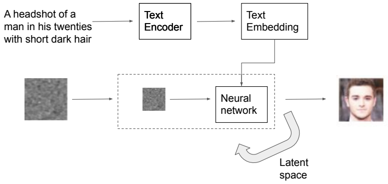
Text Encoder

Text Embedding

Neural network

Latent space

What components do we need?



A headshot of a man in his twenties with short dark hair

Text Encoder

Text Embedding

Neural network

Latent space

The key difference between latent and standard diffusion is that latent diffusion model is trained to generate latent (compressed) representations of the images

What 3 components do we need for latent diffusion?

- A text encoder (CLIP's Text Encoder)

What 3 components do we need for latent diffusion?

- A text encoder - CLIP's Text Encoder
- Neural network - UNet

U-Net

+ Time encoding

U-Net

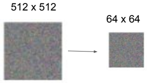
+ Time encoding + Text encoding = Conditioned Image

What 3 components do we need for stable diffusion?

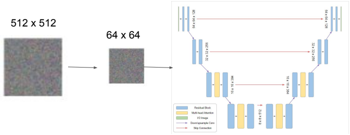
- A text encoder - CLIP's Text Encoder
- Neural network - UNet
- Autoencoder

Autoencoder

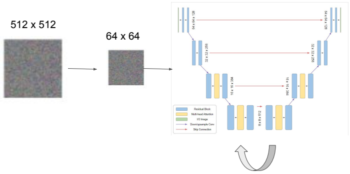
Autoencoder



Autoencoder



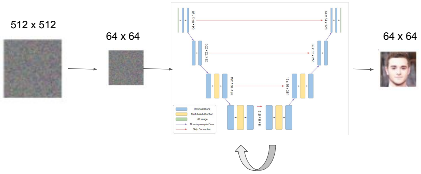
Autoencoder



512 x 512

64 x 64

Autoencoder

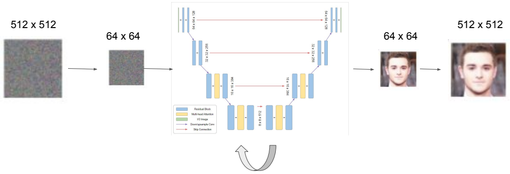


512 x 512

64 x 64

64 x 64

Autoencoder



512 x 512

64 x 64

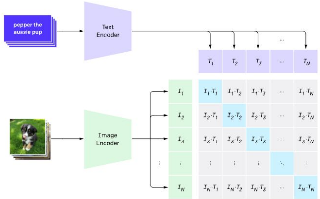
64 x 64

512 x 512

Colab notebook - autoencoder

Text encoder - CLIP

CLIP - Contrastive pre-training



pepper the aussie pup

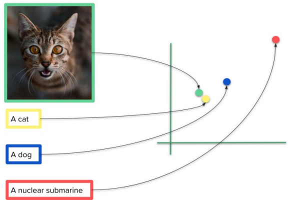
Text Encoder

Image Encoder

	T_1	T_2	T_3	...	T_N
I_1	$I_1 \cdot T_1$	$I_1 \cdot T_2$	$I_1 \cdot T_3$	...	$I_1 \cdot T_N$
I_2	$I_2 \cdot T_1$	$I_2 \cdot T_2$	$I_2 \cdot T_3$	...	$I_2 \cdot T_N$
I_3	$I_3 \cdot T_1$	$I_3 \cdot T_2$	$I_3 \cdot T_3$	...	$I_3 \cdot T_N$
⋮	⋮	⋮	⋮	⋱	⋮
I_N	$I_N \cdot T_1$	$I_N \cdot T_2$	$I_N \cdot T_3$	...	$I_N \cdot T_N$

Source: https://openai.com/blog/clip/

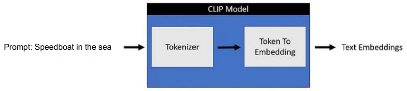
CLIP



A cat

A dog

A nuclear submarine



Prompt: Speedboat in the sea

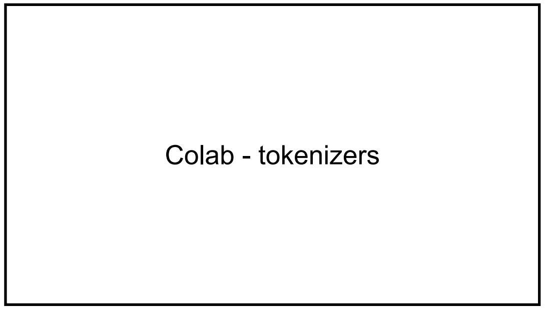
CLIP Model

Tokenizer

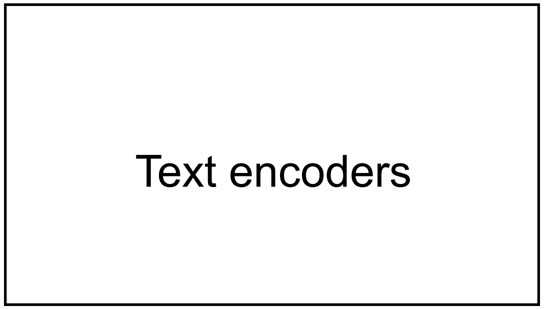
Token To Embedding

Text Embeddings

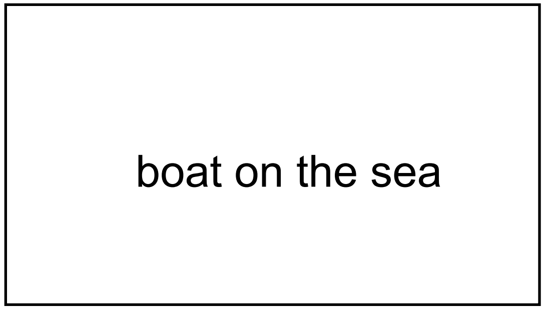
Colab - CLIP Model



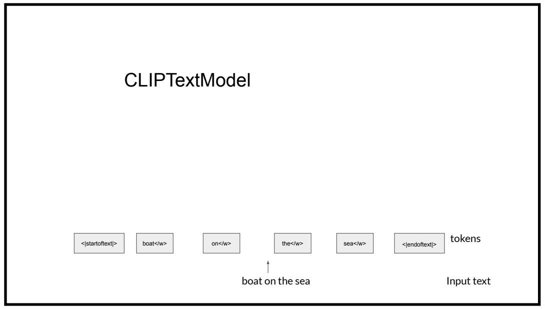
Colab - tokenizers



Text encoders



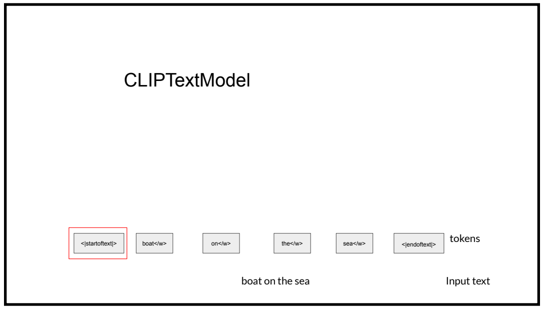
boat on the sea



CLIPTextModel

<|startoftext|> boat</w> on</w> the</w> sea</w> <|endoftext|> tokens

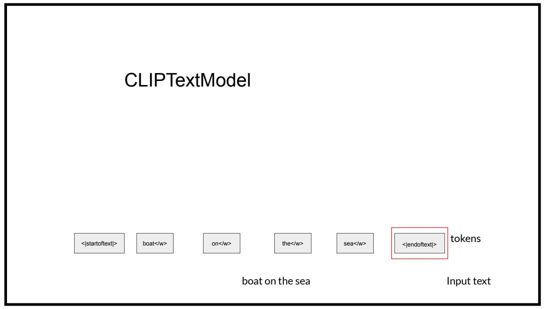
boat on the sea Input text



CLIPTextModel

<|startoftext|> boat</w> on</w> the</w> sea</w> <|endoftext|> tokens

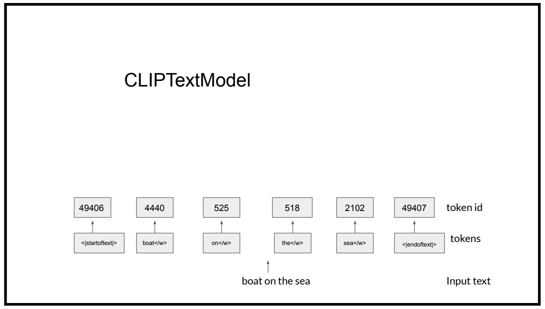
boat on the sea Input text



CLIPTextModel

<|startoftext|> boat</w> on</w> the</w> sea</w> <|endoftext|> tokens

boat on the sea Input text

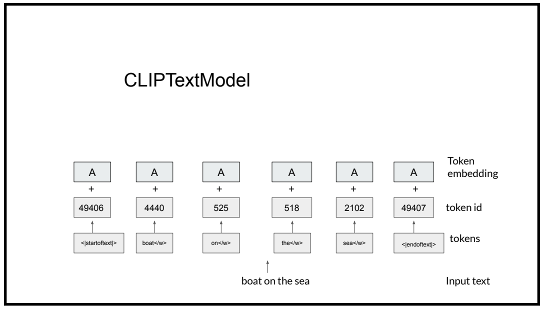


CLIPTextModel

49406 4440 525 518 2102 49407 token id

<|startoftext|> boat</w> on</w> the</w> sea</w> <|endoftext|> tokens

boat on the sea Input text



CLIPTextModel

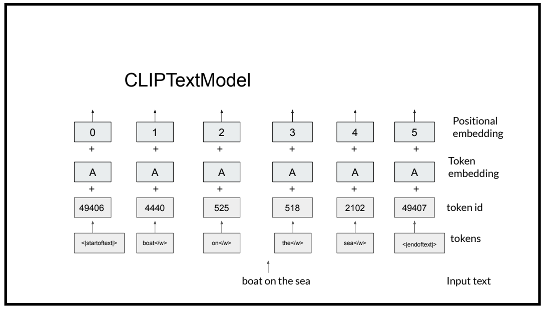
A A A A A A Token embedding

+ + + + + +

49406 4440 525 518 2102 49407 token id

<|startoftext|> boat</w> on</w> the</w> sea</w> <|endoftext|> tokens

boat on the sea Input text



CLIPTextModel

0 1 2 3 4 5 Positional embedding

+ + + + + +

A A A A A A Token embedding

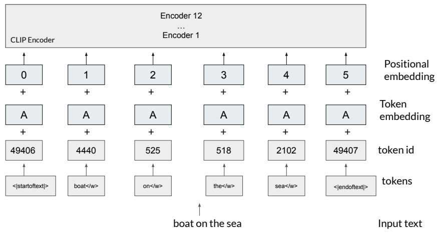
+ + + + + +

49406 4440 525 518 2102 49407 token id

<|startoftext|> boat</w> on</w> the</w> sea</w> <|endoftext|> tokens

boat on the sea Input text

CLIPTextModel



Encoder 12
...
Encoder 1

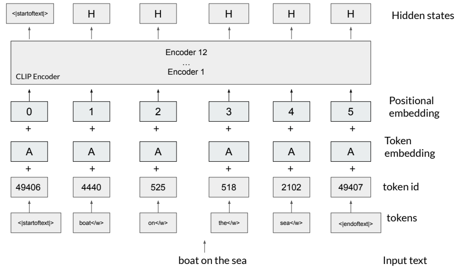
CLIP Encoder

0	1	2	3	4	5	Positional embedding
A	A	A	A	A	A	Token embedding
49406	4440	525	518	2102	49407	token id
<|startoftext|>	boat</w>	on</w>	the</w>	sea</w>	<|endoftext|>	tokens

boat on the sea

Input text

CLIPTextModel



<|startoftext|>	H	H	H	H	H	Hidden states

Encoder 12
...
Encoder 1

CLIP Encoder

0	1	2	3	4	5	Positional embedding
A	A	A	A	A	A	Token embedding
49406	4440	525	518	2102	49407	token id
<|startoftext|>	boat</w>	on</w>	the</w>	sea</w>	<|endoftext|>	tokens

boat on the sea

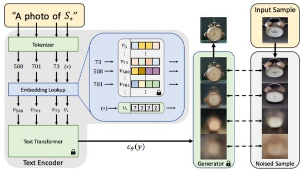
Input text

Colab - text encoders

Textual Inversion

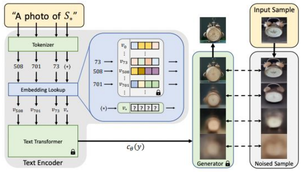
Textual Inversion

- Capture concepts from a small number of example images to control text-to-image pipelines



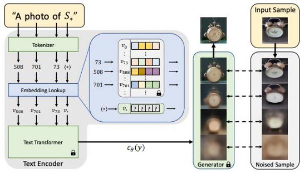
Textual Inversion

- A prompt containing the placeholder word is first converted to tokens



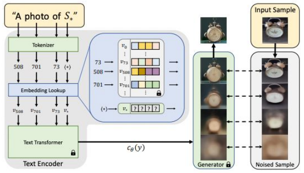
Textual Inversion

- The tokens are converted to embeddings



Textual Inversion

- Finally, the embedding vectors are transformed into a single conditioning code cθ(y) which guides the generative model.

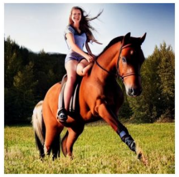


InstructPix2Pix

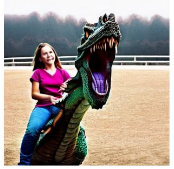
InstructPix2Pix

- https://huggingface.co/spaces/timbrooks/instruct-pix2pix
-

“A photograph of a girl riding a horse”



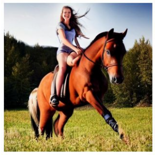
“A photograph of a girl riding a dragon”

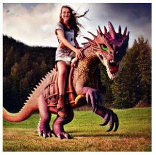


Different images generated

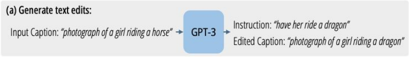
With Prompt-to-Prompt
“A photograph of a girl riding a horse”

With Prompt-to-Prompt
“A photograph of a girl riding a dragon”





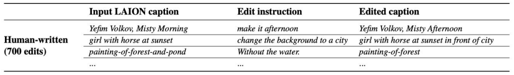
Training Data Generation



(a) Generate text edits:

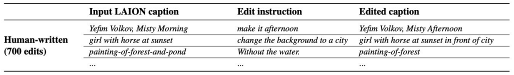
Input Caption: *“photograph of a girl riding a horse”* → GPT-3 → Instruction: *“have her ride a dragon”*
Edited Caption: *“photograph of a girl riding a dragon”*

Training Data Generation



	Input LAION caption	Edit instruction	Edited caption
Human-written (700 edits)	*Yefim Volkov, Misty Morning*	*make it afternoon*	*Yefim Volkov, Misty Afternoon*
	girl with horse at sunset	*change the background to a city*	*girl with horse at sunset in front of city*
	painting-of-forest-and-pond	*Without the water.*	*painting-of-forest*
	...	...	...

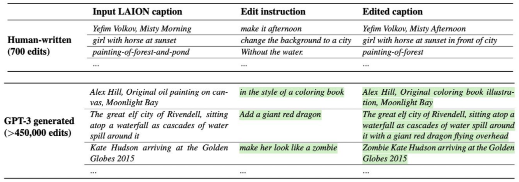
Training Data Generation



	Input LAION caption	Edit instruction	Edited caption
Human-written (700 edits)	*Yefim Volkov, Misty Morning*	*make it afternoon*	*Yefim Volkov, Misty Afternoon*
	girl with horse at sunset	*change the background to a city*	*girl with horse at sunset in front of city*
	painting-of-forest-and-pond	*Without the water.*	*painting-of-forest*
	...	...	...

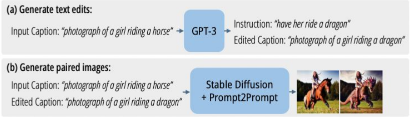
Fine-tune GPT-3 to generate a large dataset of text triplets

Training Data Generation



	Input LAION caption	Edit instruction	Edited caption
Human-written (700 edits)	*Yefim Volkov, Misty Morning*	*make it afternoon*	*Yefim Volkov, Misty Afternoon*
	girl with horse at sunset	*change the background to a city*	*girl with horse at sunset in front of city*
	painting-of-forest-and-pond	*Without the water.*	*painting-of-forest*
	...	...	...
GPT-3 generated (>450,000 edits)	*Alex Hill, Original oil painting on canvas, Moonlight Bay*	*in the style of a coloring book*	*Alex Hill, Original coloring book illustration, Moonlight Bay*
	The great elf city of Rivendell, sitting atop a waterfall as cascades of water spill around it	*Add a giant red dragon*	*The great elf city of Rivendell, sitting atop a waterfall as cascades of water spill around it with a giant red dragon flying overhead*
	Kate Hudson arriving at the Golden Globes 2015	*make her look like a zombie*	*Zombie Kate Hudson arriving at the Golden Globes 2015*
	...	...	...

Training Data Generation



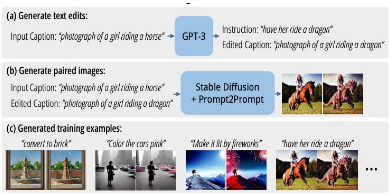
(a) Generate text edits:

Input Caption: *"photograph of a girl riding a horse"* → GPT-3 → Instruction: *"have her ride a dragon"*
Edited Caption: *"photograph of a girl riding a dragon"*

(b) Generate paired images:

Input Caption: *"photograph of a girl riding a horse"*
Edited Caption: *"photograph of a girl riding a dragon"* → Stable Diffusion + Prompt2Prompt →

Training Data Generation



(a) Generate text edits:

Input Caption: *"photograph of a girl riding a horse"* → GPT-3 → Instruction: *"have her ride a dragon"*
Edited Caption: *"photograph of a girl riding a dragon"*

(b) Generate paired images:

Input Caption: *"photograph of a girl riding a horse"*
Edited Caption: *"photograph of a girl riding a dragon"* → Stable Diffusion + Prompt2Prompt →

(c) Generated training examples:

"convert to brick" *"Color the cars pink"* *"Make it lit by fireworks"* *"have her ride a dragon"* ...

What have we looked at?

- Non-technical
  - Try out different models
  - Prompting
  - Bias, limitations and controversy
- 180 min: Understand how some of the models work behind the scenes
  - Diffusion
    - Conditioned models
  - CLIP
  - U-Net
  - Stable Diffusion
  - Textual Inversion
  - InstructPix2Pix

What have we looked at?

- We've tried out different models
- We've looked at prompting and prompting guides to improve
- Looked at Bias, limitations and controversy with AI generation.
- Diffusion
  - Conditioned models
- Stable Diffusion
  - CLIP
  - U-Net
- Textual Inversion
- InstructPix2Pix

Next iteration of this course will include the latest AI generation

Live Course

Hands-On Transformers for Computer Vision

With Jonathan Fernandes

July 19 • 5pm - 9pm BST

END