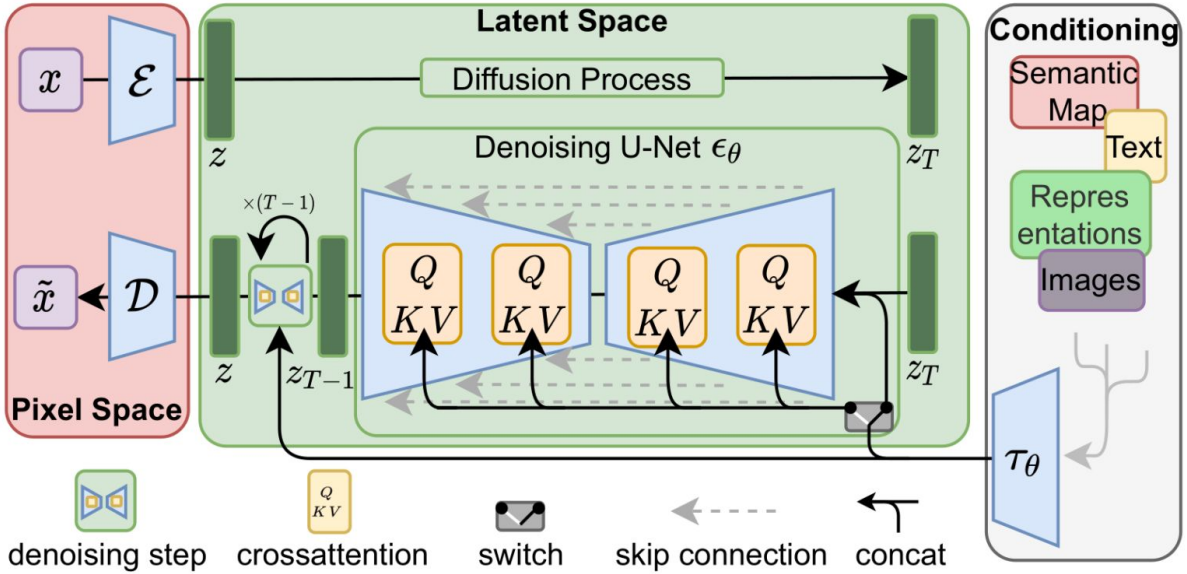
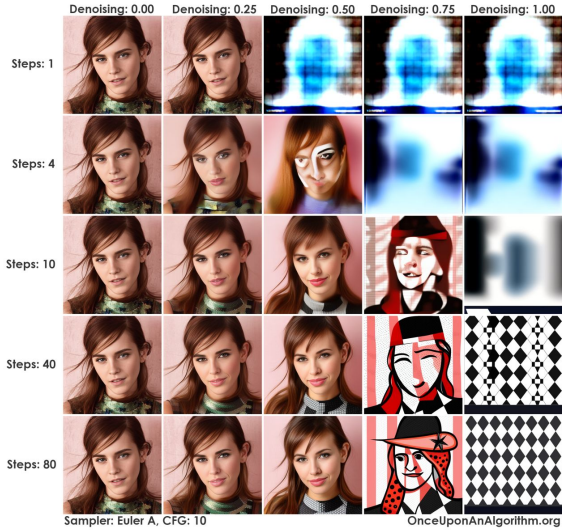

Dreambooth
Fine tuning

A little about me…

- Leader in data science, machine & deep learning and interested in all things MLOps and deep/ machine learning
- Tensorflow Certified Developer and senior fellow of NHS-R Community. Experienced SQL, Python, R and moderate C sharp developer :)
- Background as data scientist, ML engineer, database developer and various leadership and department head roles
- Working for Crisp: A Kroll Company to implement NLP and Computer Vision solutions to detect and prevent online risks as Head of ML & Graph Data Science
- Previously worked for police intelligence, geographical crime analytics, National Health Service, CoreLogic, Ascent and Draper & Dash (Realworld Health).

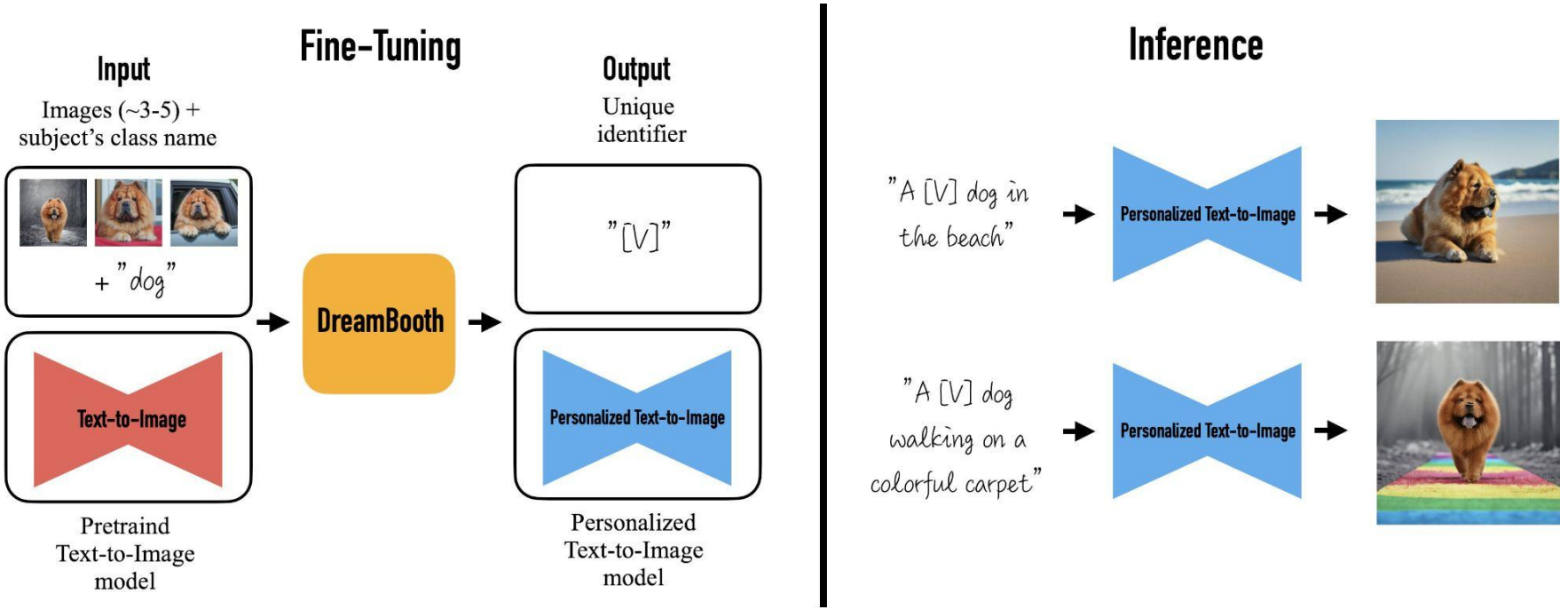
How does Stable Diffusion work?



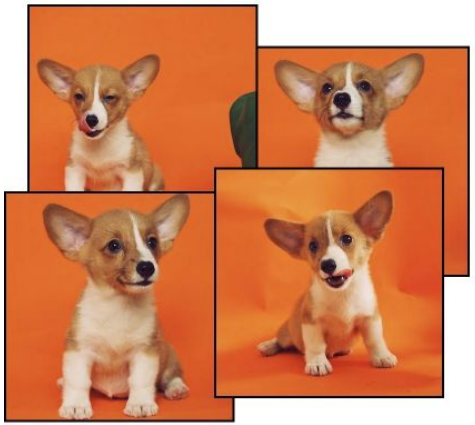


https://jalammar.github.io/illustrated-stable-diffusion/

What is the Dreambooth method?



Put simply…



Input images

in the Acropolis

swimming

sleeping

in a doghouse

in a bucket

getting a haircut

Storing our images in HuggingFace for training

```
from datasets import load_dataset
dataset = load_dataset('imagefolder',
data_dir='images/<your image set>')
dataset.push_to_hub('<name to be displayed in HF>')
```


https://huggingface.co/datasets/StatsGary/dreambooth-hackathon-images

Datasets

Loading our images for training

```
from datasets import load_dataset
# Change dataset ID to your HuggingFace image ID
dataset_id = "StatsGary/dreambooth-hackathon-images"
dataset = load_dataset(dataset_id, split="train")
dataset
```

Get an image grid

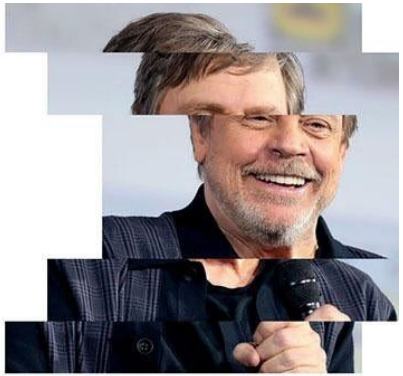
```
from PIL import Image
def image_grid(imgs, rows, cols):
    assert len(imgs) == rows * cols
    w, h = imgs[0].size
    grid = Image.new("RGB", size=(cols * w, rows * h))
    grid_w, grid_h = grid.size
    for i, img in enumerate(imgs):
        grid.paste(img, box=(i % cols * w, i // cols *
h))
    return grid

# Choose number of samples to display
num_samples = 8
image_grid(dataset["image"][:num_samples], rows=1,
cols=num_samples)
```

Configure config file

```
train_params:
  stable_diffusion_backbone:
CompVis/stable-diffusion-v1-4
  feature_extractor: openai/clip-vit-base-patch32
  hugging_face_image_store:
StatsGary/dreambooth-hackathon-images
  learning_rate: 2e-06
  max_train_steps: 400
  resolution: 512
  train_bs: 1
  grad_accum_steps: 8
  max_gradient_norm: 1.0
  sample_batch_size: 2
  model_checkpoint_name: norweigen-fjords-dreambooth
  random_shuffle_train_set: True
  use_8bit_optimizer: True
eval_params:
  image_save_path: images
  eval_prompt: a viking on a boat in a fjord
```

Mark Hamill



Mark Yaml

Train the model - getting your imports right

```
from dreambooth.dataloader import
pull_dataset_from_hf_hub, DreamBoothDataset
from dreambooth.image import image_grid
from dreambooth.collator import collate_fn
from dreambooth.train import train_dreambooth
from transformers import CLIPTokenizer
from diffusers import AutoencoderKL,
UNet2DConditionModel
from transformers import CLIPFeatureExtractor,
CLIPTextModel
import logging
import yaml

# SET project constants and variables
with open('dreambooth_param.yml', 'r') as train:
    params = yaml.safe_load(train)
```

Train the model - loading data and naming concept prompt

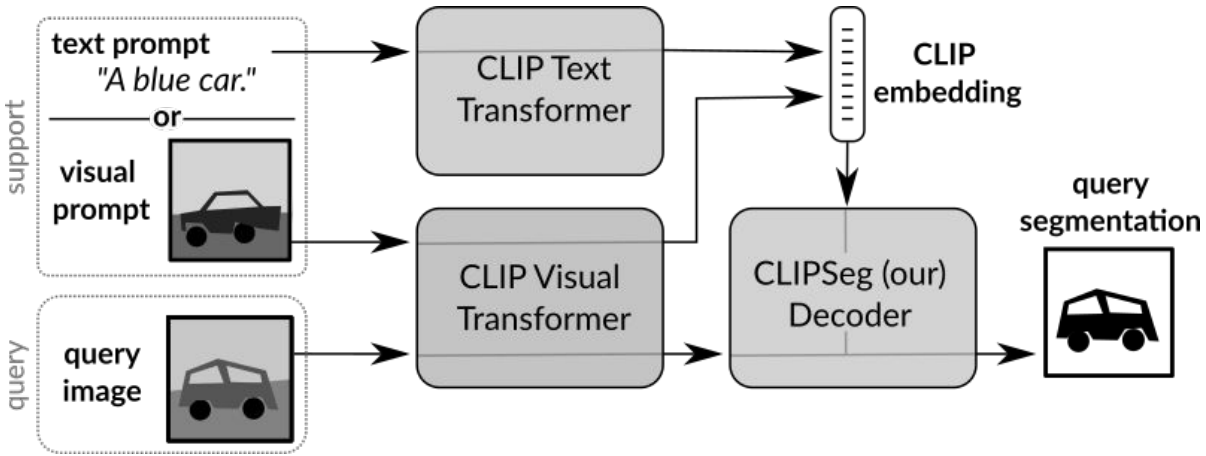
```
# Load the image dataset from HuggingFace hub
dataset =
pull_dataset_from_hf_hub(dataset_id=hf_data_location)

# Name your concept and set of images
name_of_your_concept = name_of_your_concept
type_of_thing = object_type
instance_prompt = f"a photo of {name_of_your_concept}
{type_of_thing}"
print(f"Instance prompt: {instance_prompt}")
```

huggingface_hub

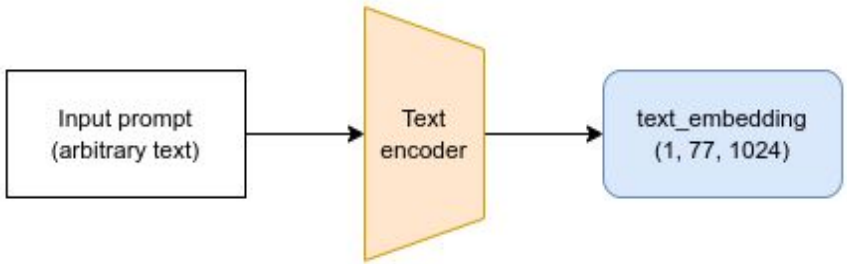
Train the model - CLIP tokenizer

```
# Load the CLIP tokenizer
model_id = STABLE_DIFFUSION_NAME
tokenizer = CLIPTokenizer.from_pretrained(
              model_id,
              subfolder="tokenizer")
```



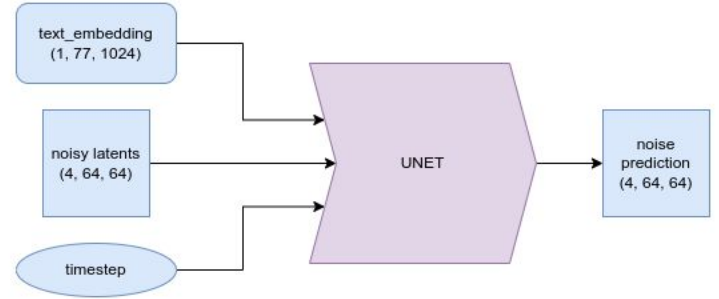
Train the model - use Dreambooth dataloader and text encoder

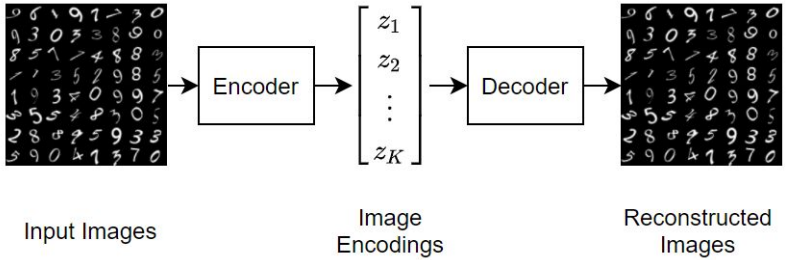
```
# Create a train dataset from the Dreambooth data
loader
train_dataset = DreamBoothDataset(dataset,
instance_prompt, tokenizer)
# Get text encoder - CLIP
text_encoder = CLIPTextModel.from_pretrained(model_id,
subfolder="text_encoder")
```



Train the model - load VAE, UNET amd feature extractor

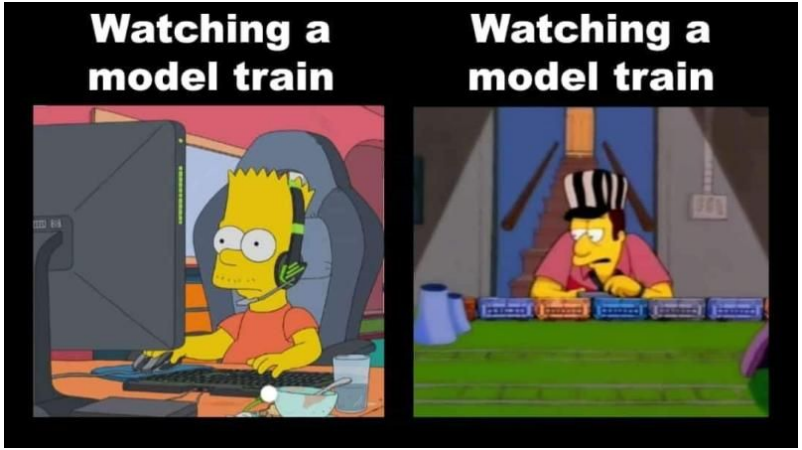
```
vae = AutoencoderKL.from_pretrained(model_id,
subfolder="vae")
unet = UNet2DConditionModel.from_pretrained(model_id,
subfolder="unet")
feature_extractor =
CLIPFeatureExtractor.from_pretrained(FEATURE_EXTRACTOR
)
```





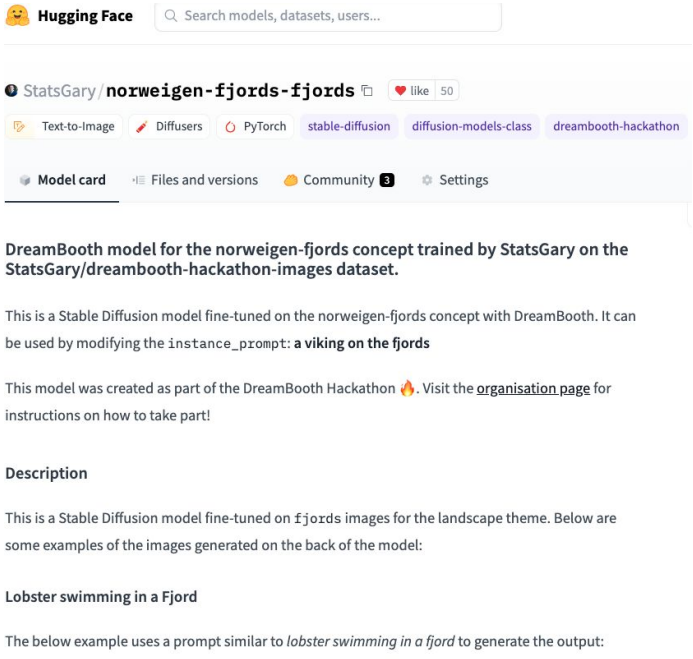
Train the model - set it to run

```
# Train the model
model = train_dreambooth(
       text_encoder=text_encoder,
        vae = vae,
        unet = unet,
        tokenizer=tokenizer,
        feature_extractor=feature_extractor,
        train_dataset=train_dataset,
        train_batch_size=train_batch_size,
        max_train_steps=max_train_steps,
        shuffle_train=shuffle_train,
        gradient_accumulation_steps=grad_accum_steps,
        use_8bit_ADAM=True,
        learning_rate=learning_rate,
        max_grad_norm=max_gradient_norm,
        output_dir=model_checkpoint_name)
```



Push your model to the hub

```python
from huggingface_hub import HfApi, ModelCard, create_repo,
get_full_repo_name
# Create a name for your model on the Hub. No spaces allowed.
model_name = f"{name_of_your_concept}-{type_of_thing}"
description = f"""
This is a Stable Diffusion model fine-tuned on `{type_of_thing}`
images for the {theme} theme.
"""
# Set up repo and upload files
hub_model_id = get_full_repo_name(model_name)
create_repo(hub_model_id)
api = HfApi()
api.upload_folder(folder_path=args.output_dir, path_in_repo="",
repo_id=hub_model_id)
content = f'<Model card description>'
card = ModelCard(content)
hub_url = card.push_to_hub(hub_model_id)
print(f"Upload successful! Model can be found here: {hub_url}")
```



Inference options via hub

StatsGary / **norweigen-fjords-fjords** like 50

Text-to-Image | Diffusers | PyTorch | stable-diffusion | diffusion-models-class | dreambooth-hackathon | landscape | License: creativeml-openrail-m

Model card | Files and versions | Community 3 | Settings | Deploy | Use in Diffusers

Edit model card

DreamBooth model for the norweigen-fjords concept trained by StatsGary on the StatsGary/dreambooth-hackathon-images dataset.

This is a Stable Diffusion model fine-tuned on the norweigen-fjords concept with DreamBooth. It can be used by modifying the `instance_prompt`: **a viking on the fjords**

This model was created as part of the DreamBooth Hackathon 🔥. Visit the organisation page for instructions on how to take part!

Description

This is a Stable Diffusion model fine-tuned on `fjords` images for the landscape theme. Below are some examples of the images generated on the back of the model:

Lobster swimming in a Fjord

The below example uses a prompt similar to *lobster swimming in a fjord* to generate the output:

Downloads last month
32

Hosted inference API

Text-to-Image | Examples

painting of a fjord in van goch style | Compute

Computation time on gpu: 5.026 s






Inference options in script

```
if __name__ == '__main__':
    # Load in fine tuned model
    model_name =
train_params['model_checkpoint_name']
    pipe = StableDiffusionPipeline.from_pretrained(
        model_name,
        torch_dtype=torch.float16,
    ).to("cuda")

    # Use config or user prompt
    if get_user_input==False:
        prompt = str(eval_params['eval_prompt'])
    else:
        prompt = input()

    guidance_scale = 7

    num_cols = 2
    all_images = []
    for _ in range(num_cols):
        images = pipe(prompt,
guidance_scale=guidance_scale).images
        all_images.extend(images)
    plt = image_grid(all_images, 1, num_cols)
    save_path =
f"{eval_params['image_save_path']}/{str(prompt.replac
e(' ','')[-10:])}.jpg"
    plt.save(save_path)
```

```
eval_params:
  image_save_path: images
  eval_prompt: a viking on a boat in a fjord
```

Where to get the code?

https://github.com/StatsGary/stable-diffusion-leeds-data-science

Dark side of generative AI

- Paedophiles using AI to create child abuse imagery: https://www.thetimes.co.uk/article/paedophiles-using-ai-to-create-child-abuse-images-mrmxfd03s
- Generative porn: https://www.forbes.com/sites/rashishrivastava/2023/05/11/reddit-ai-generated-porn/
- Deep fakes of celebrities and other politicians can be easily recreated with tools such as Midjourney, Dalle and Dreambooth

Questions?