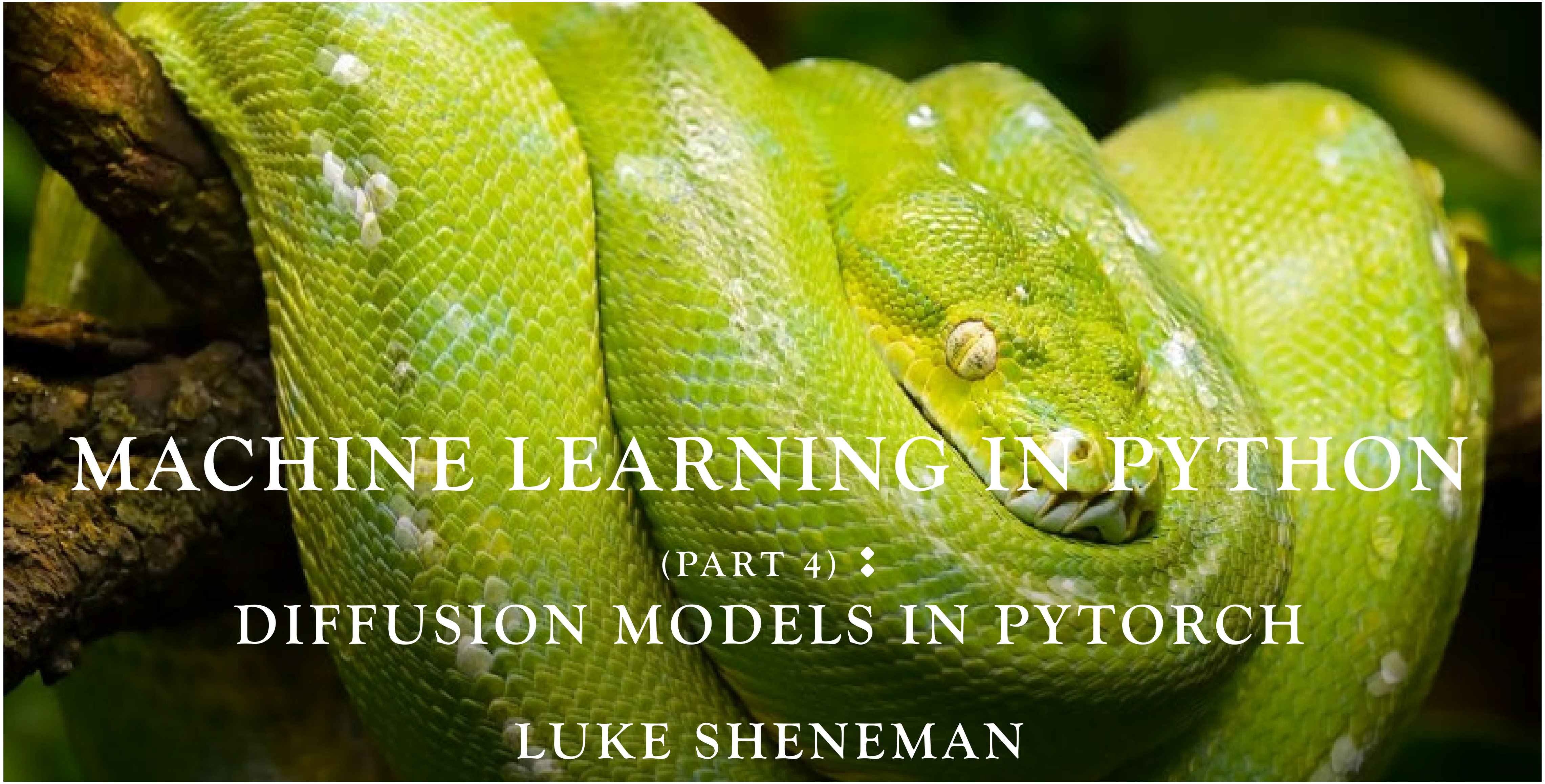

MACHINE LEARNING IN PYTHON
(PART 4)
DIFFUSION MODELS IN PYTORCH
LUKE SHENEMAN

GENERATIVE ARTIFICIAL INTELLIGENCE

Text to Image (Stable Diffusion)

Text to Video

EDUCATION

ChatGPT passes medical, law, and business exams

The AI text generator has educators debating its role in the classroom.

Real estate agents say they can't imagine working without ChatGPT now

By Samantha Murphy Kelly, CNN Business
Updated 4:06 PM EST, Sat January 28, 2023

NEWS | 18 January 2023

ChatGPT listed as author on research papers: many scientists disapprove

At least four articles credit the AI tool as a co-author, as publishers scramble to regulate its use.

Chris Stokel-Walk

Science journals ban listing of ChatGPT as co-author on papers

Some publishers also banning use of bot in preparation of submissions but others see its adoption as inevitable

19 Ways to Use ChatGPT in Your Classroom

ChatGPT Is Making Universities Rethink Plagiarism

Students and professors can't decide whether the AI chatbot is a research tool—or a cheating engine.

MUSICO

An AI that writes music

Musico is an AI-driven software engine that generates music. It can react to gesture, movement, code or other sound.

LISTEN TO DEMOS

Google has developed a music-making AI bot

The AI takeover hits the music industry.

By Chase DiBenedetto on January 29, 2023

MY VOICE IS NO LONGER MY PASSWORD —

Microsoft's new AI can simulate anyone's voice with 3 seconds of audio

Text-to-speech model can preserve speaker's emotional tone and acoustic environment.

BENJ EDWARDS - 1/9/2023, 2:15 PM

Denoising EEG Signals for Real-World BCI Applications Using GANs

Eoin Brophy[1,2*], Peter Redmond[1,3], Andrew Fleury[4], Maarten De Vos[5,6], Geraldine Boylan[2] and Tomás Ward[1,3]

Review

Generative adversarial networks unlock new methods for cognitive science

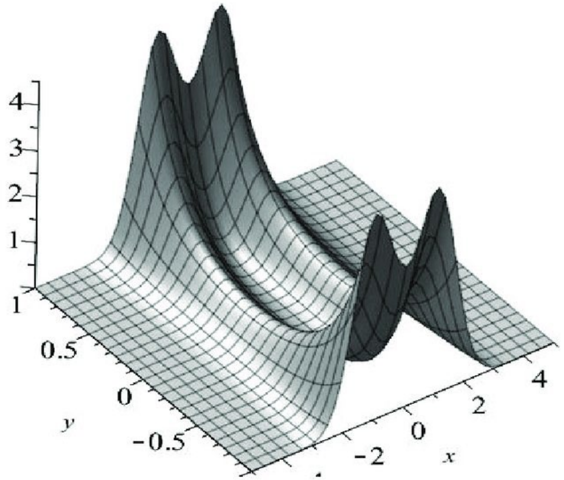
Lore Goetschalckx[1,2], Alex Andonian[3],

GANs for Driving Biological Research

GitHub Copilot

Generative AI: Learn a *latent* representation of the distribution of our complex training data and then sample from it

Training Data

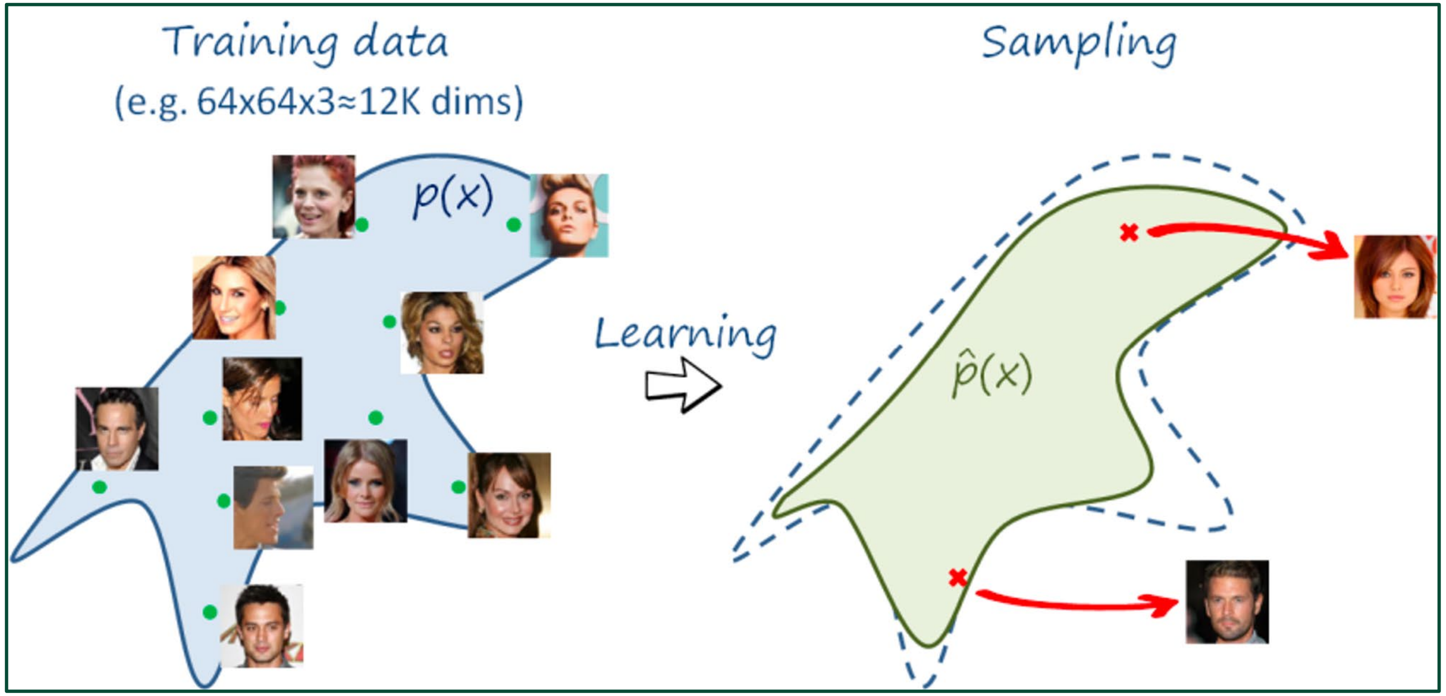


Deep Learning

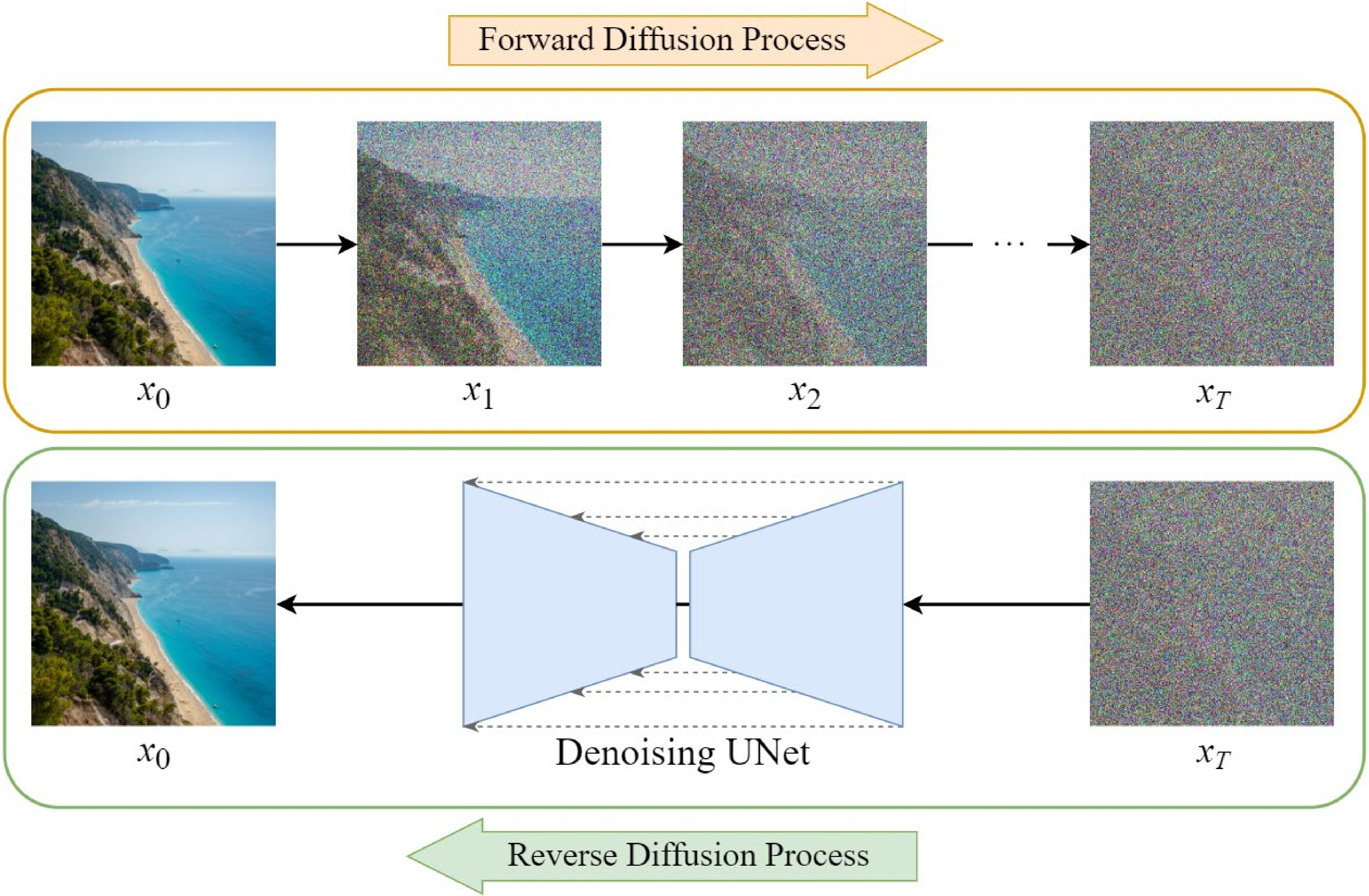
Diffusion, etc.

ChatGPT

Transformers, etc.



DIFFUSION MODELS


Forward Diffusion Process
x_0
x_1
x_2
$\cdots$
x_T
x_0
Denoising UNet
x_T
Reverse Diffusion Process

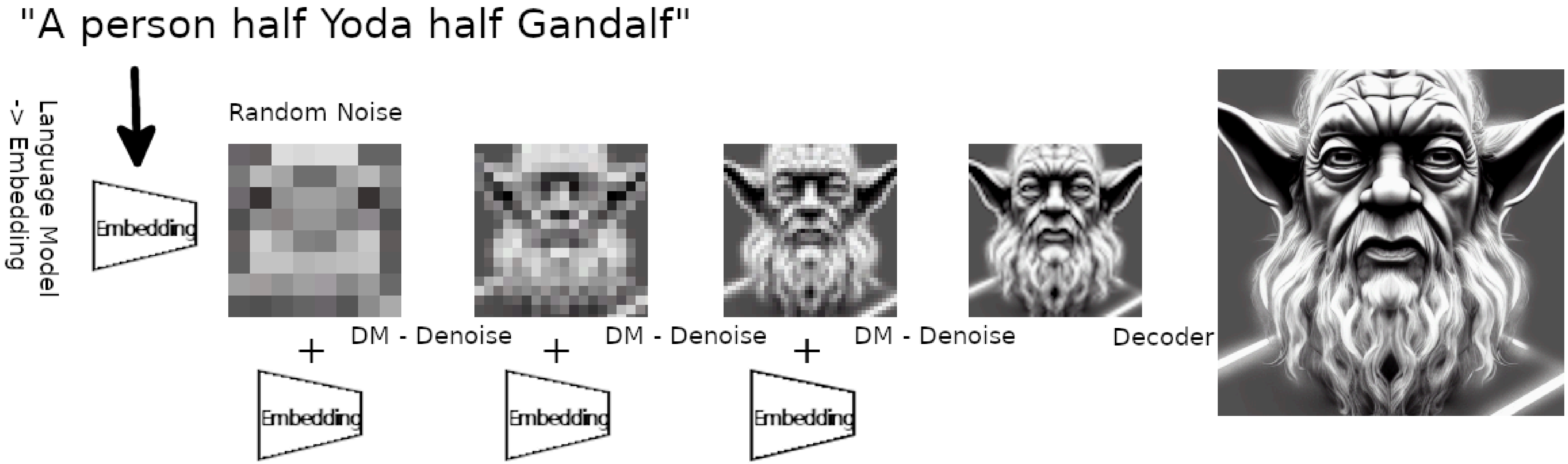
CONDITIONING IMAGE GENERATION

- Provide natural language text prompts to *guide* reverse diffusion process

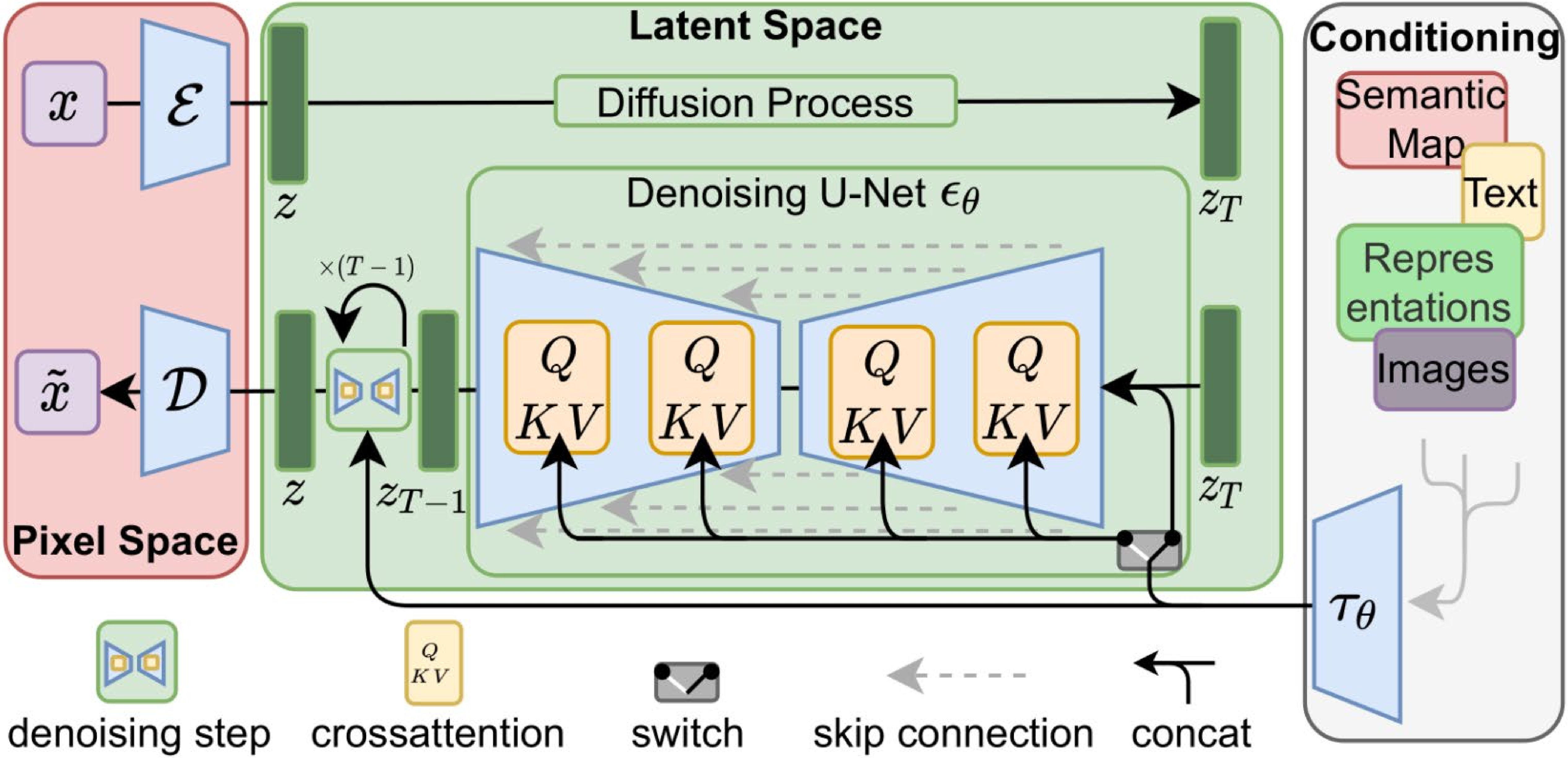
- Text-to-Image Diffusion Models <u>are both:</u>
  - Image Generation Models
  - Language Models

PROMPT #49

Midjourney prompts : *a mini golden doodle in space marine armour, chibi –v 4*


"A person half Yoda half Gandalf"
Language Model
-> Embedding
Embedding
Random Noise
DM - Denoise
DM - Denoise
DM - Denoise
Decoder
+
+
+
Embedding
Embedding
Embedding

Stable Diffusion Architecture

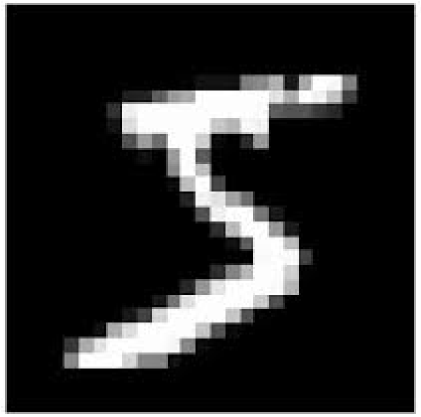


OVERVIEW

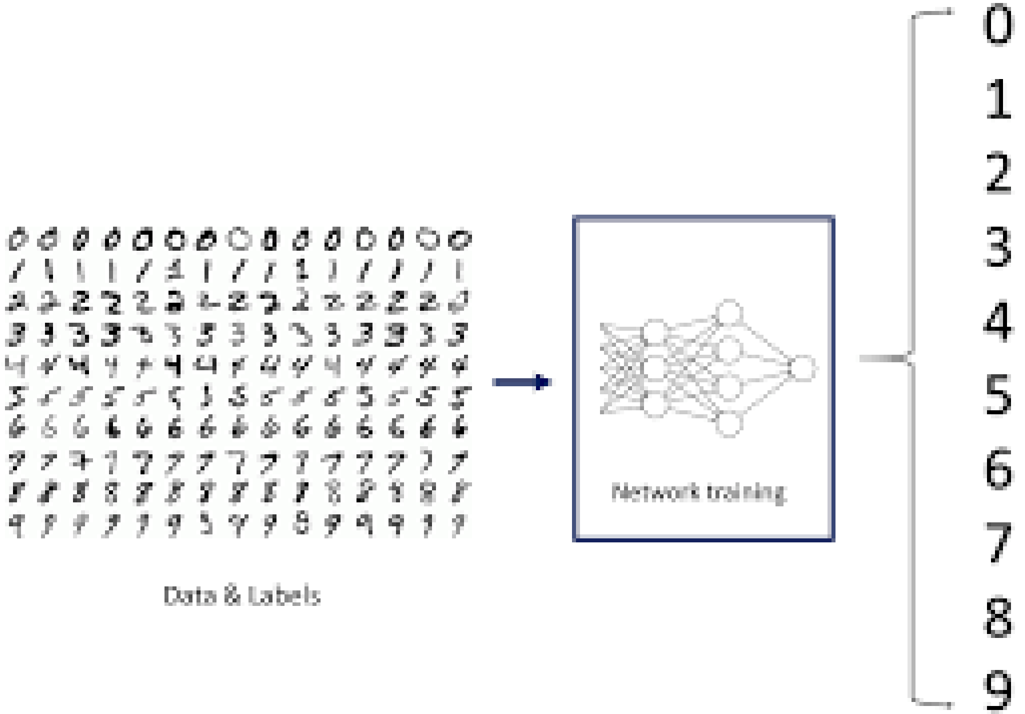
- Recap from Parts 1-3
  - Machine Learning Basics
  - Neural Networks
- Tensors
- Convolutional Neural Networks (CNNs)
- GPUs and CUDA
- PyTorch
  - Why use PyTorch?
  - Implementing a Diffusion Model in Python
  - Train and Test our Diffusion Model

REVIEW OF BASICS

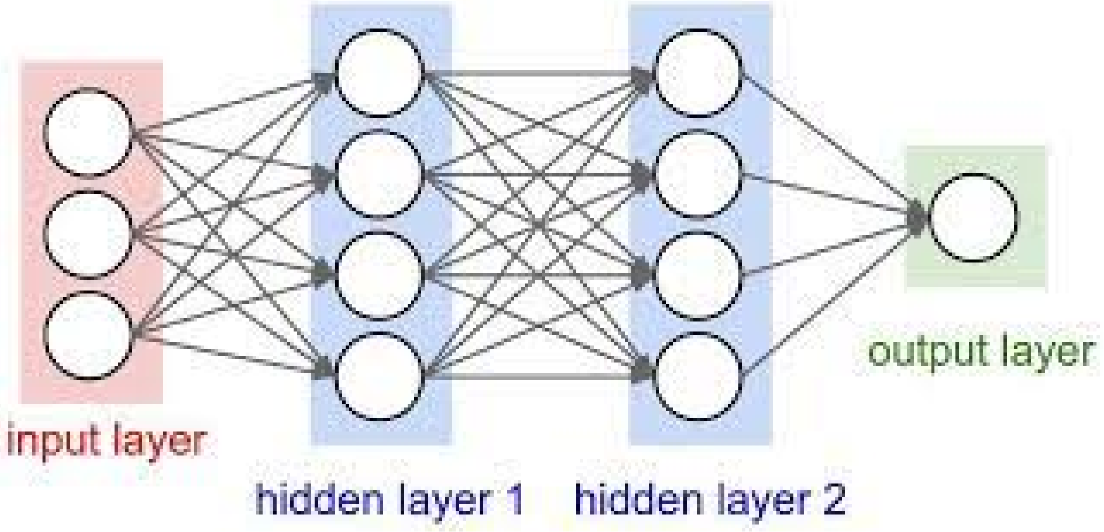
- Machine learning is a data-driven method for creating models for prediction, optimization, classification, generation, and more
- Python and scikit-learn
- MNIST
- Artificial Neural Networks (ANNs)



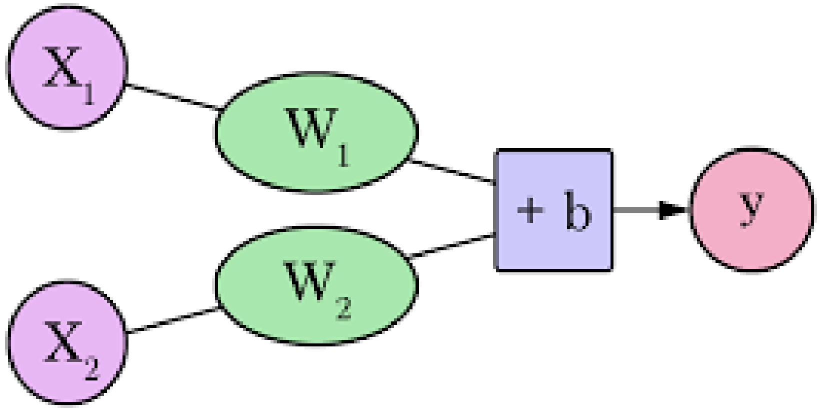
MNIST



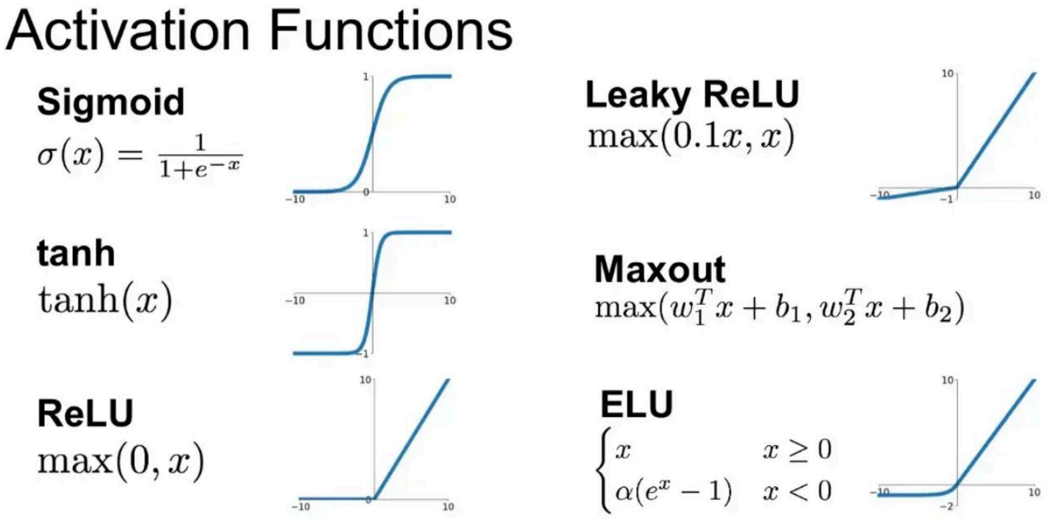
NEURAL NETWORK BASICS



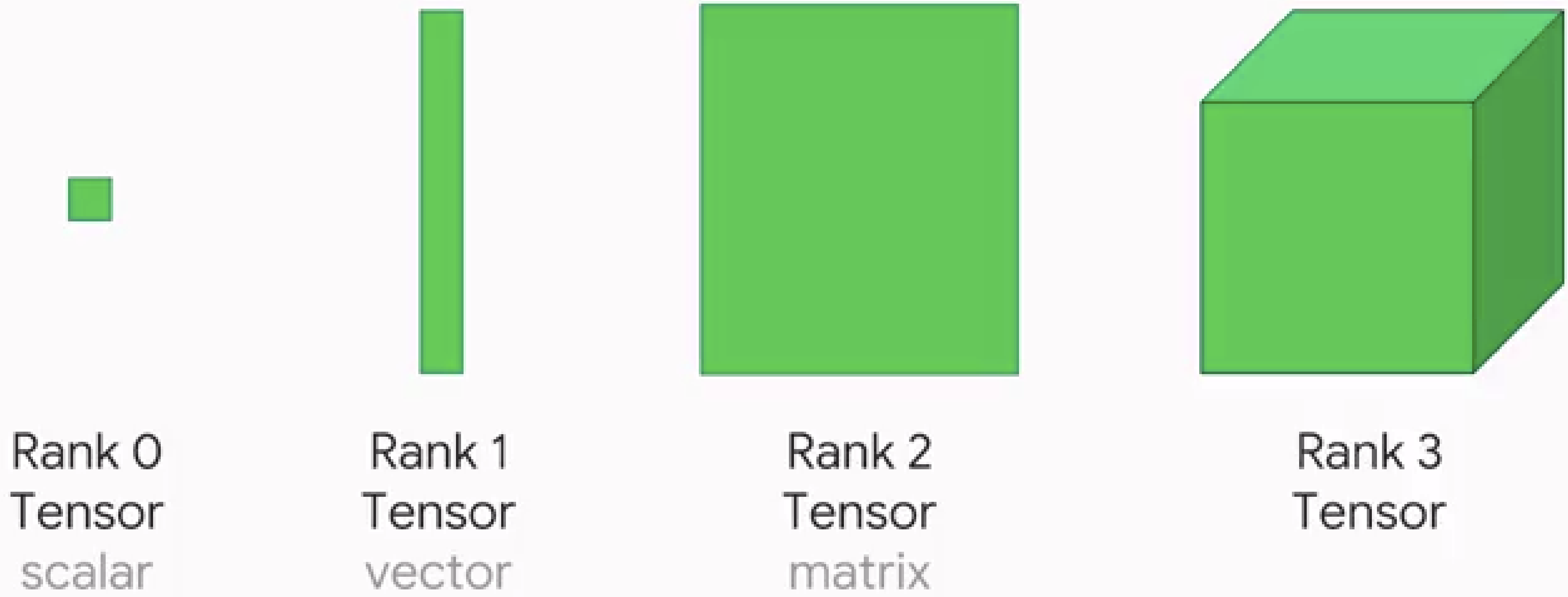

```
from sklearn.neural_network import MLPClassifier

# Instantiate a Neural Network and train our model
nn = MLPClassifier(hidden_layer_sizes=(50,25),
                   max_iter=50,
                   n_iter_no_change=50,
                   activation = 'relu',
                   solver='adam',
                   random_state=42,
                   verbose=True)
nn.fit(X_train, y_train)
```



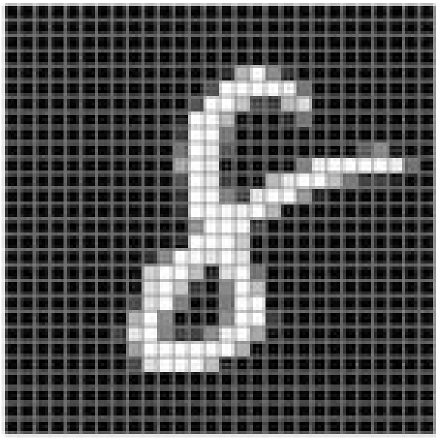
Weights and Biases



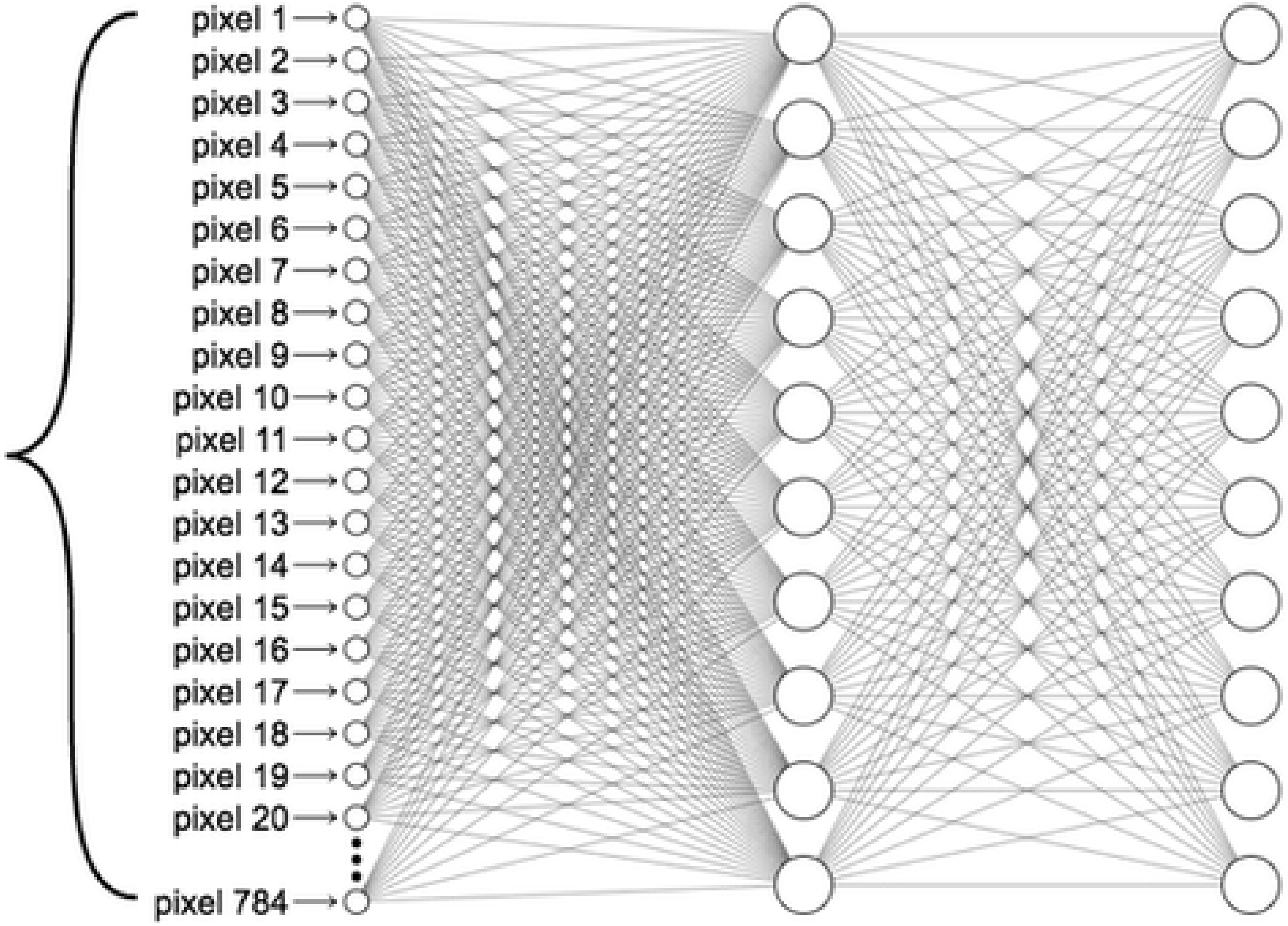
A tensor is an N-dimensional array of data



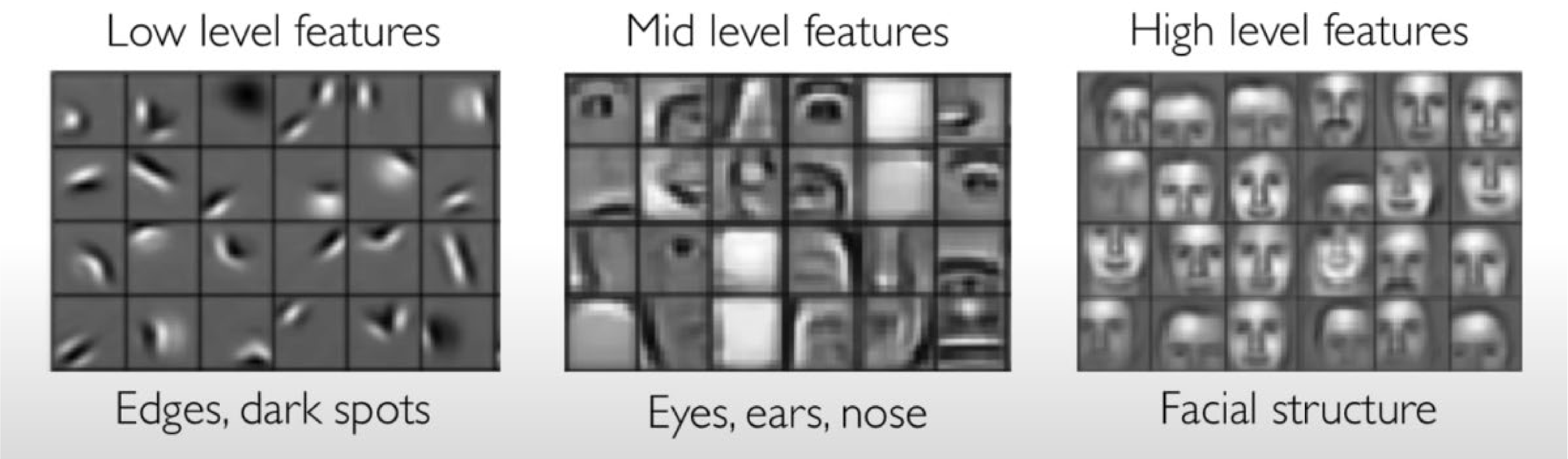
FULLY-CONNECTED NEURAL NETWORKS



Images are tensors!

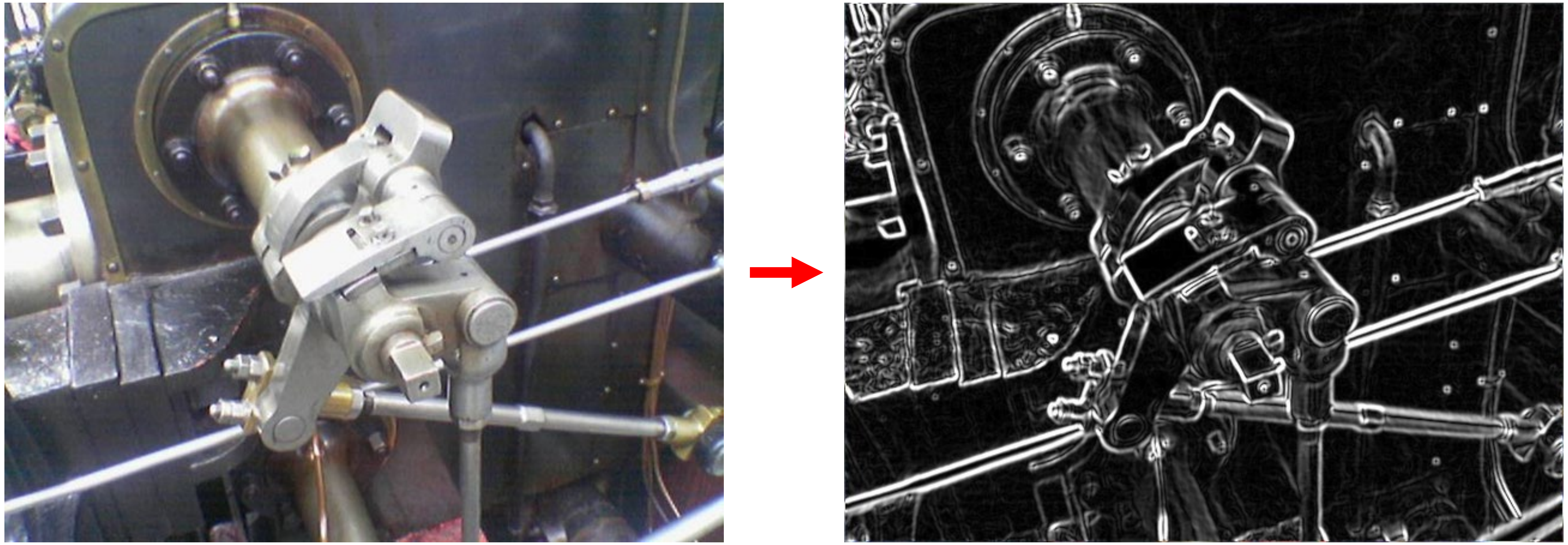


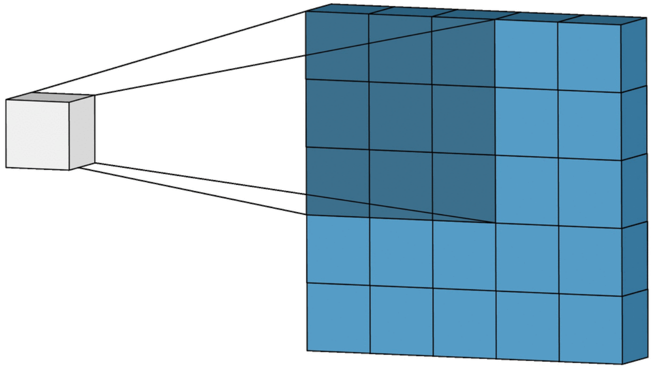
FEATURE HIERARCHIES



We need image *filters* to help us extract features

EXAMPLE: SOBEL FILTER





Sobel kernels =

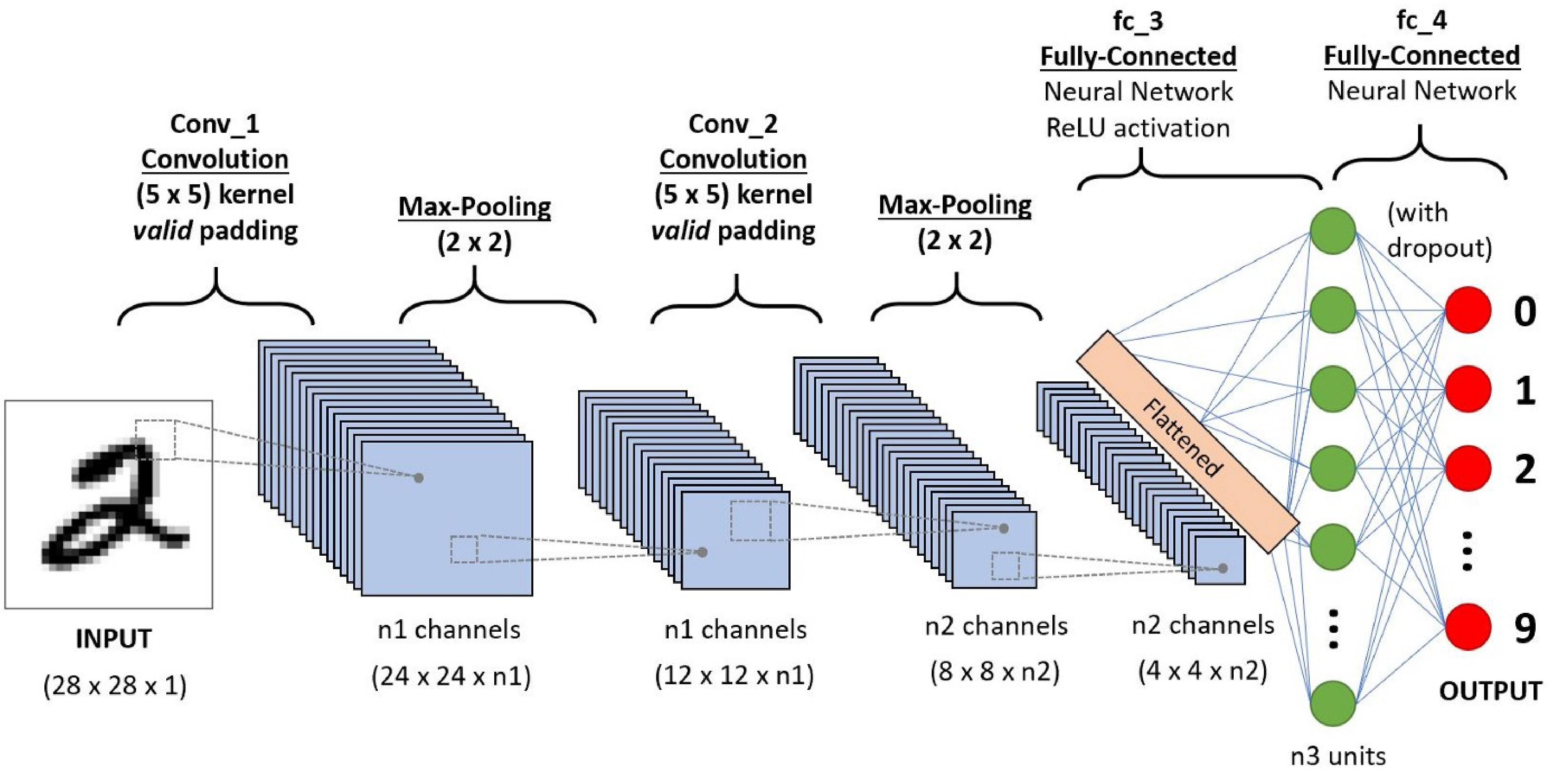
-1	0	+1
-2	0	+2
-1	0	+1

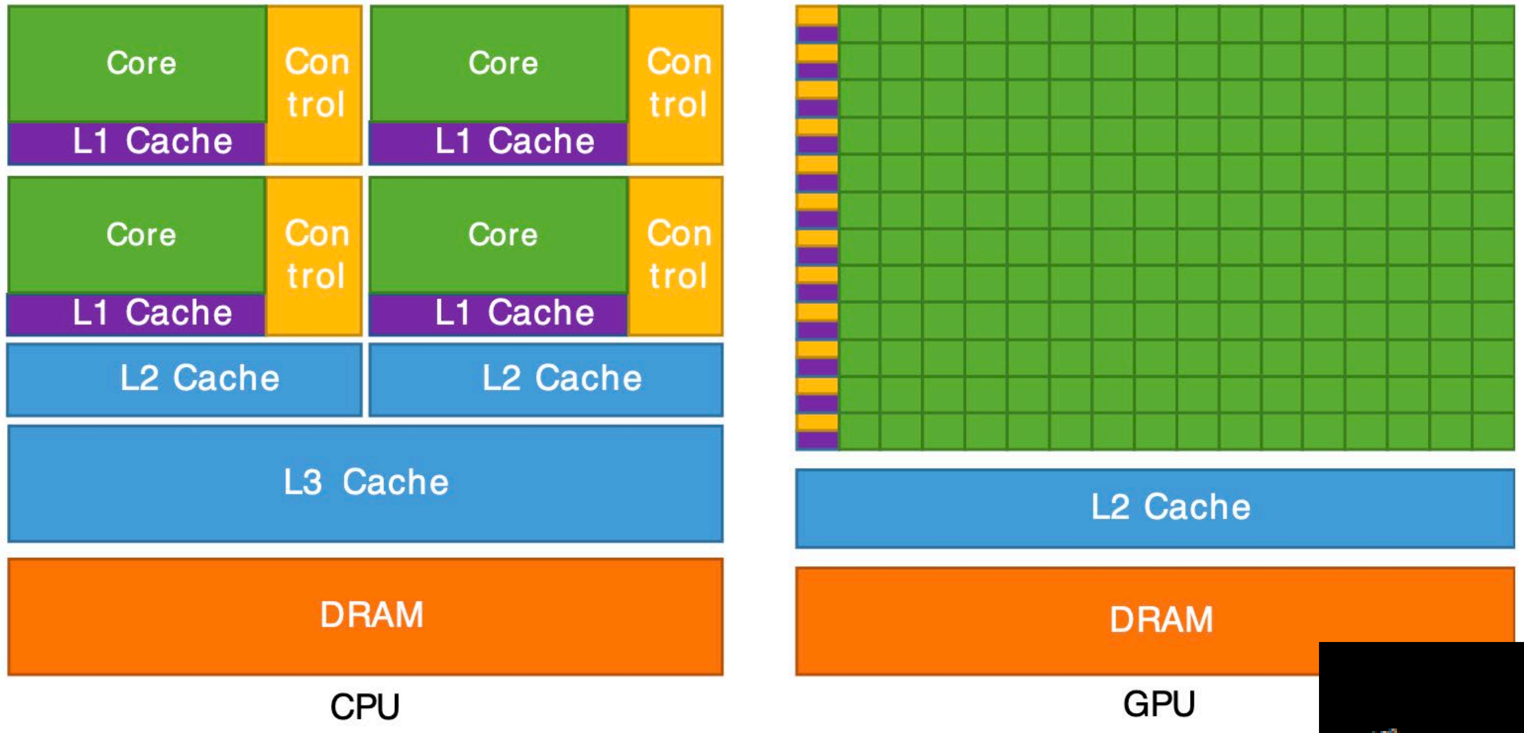
Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

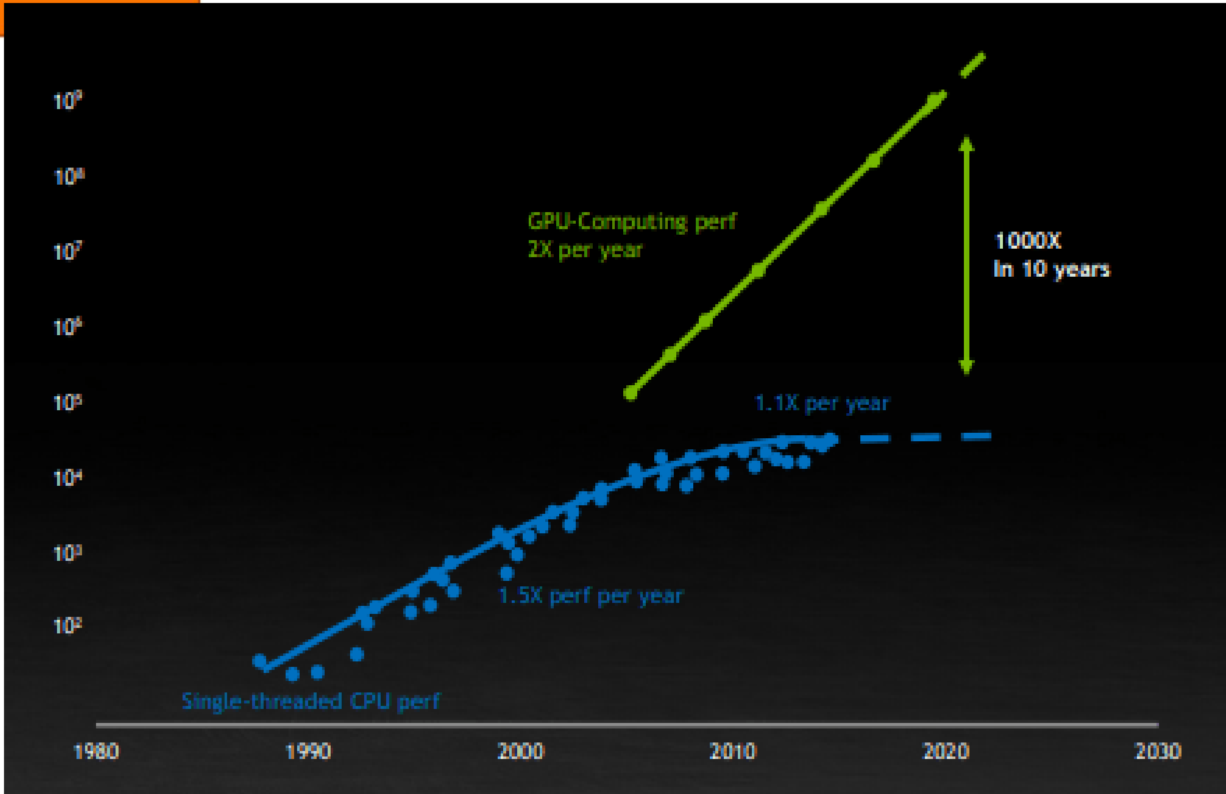
CONVOLUTIONAL NEURAL NETWORK





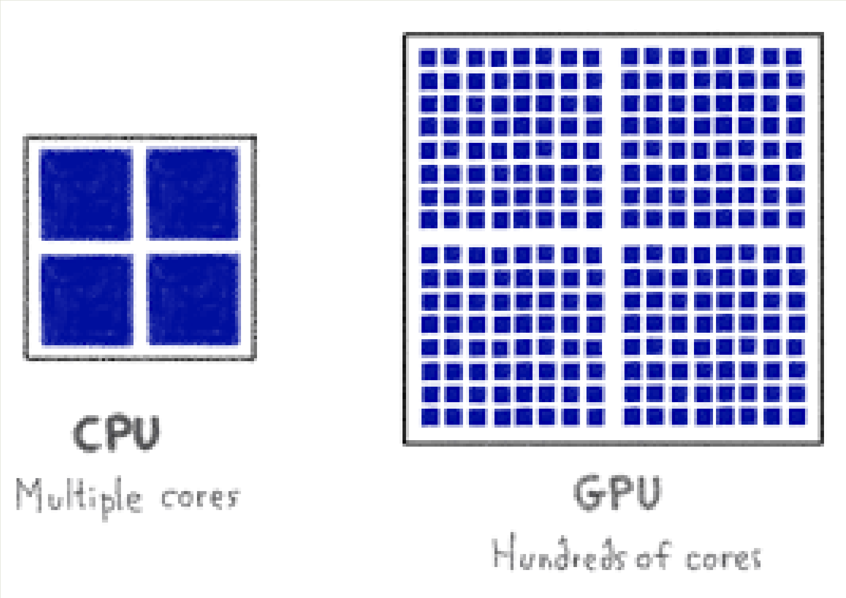
GPU vs. CPU

"Moore's Law for CPUs is Dead"



WHY GPUS EXACTLY?

- CNNs are all about matrix and vector operations (multiplication, addition)
- GPUs can perform parallel multiplication and addition steps per each clock cycle.

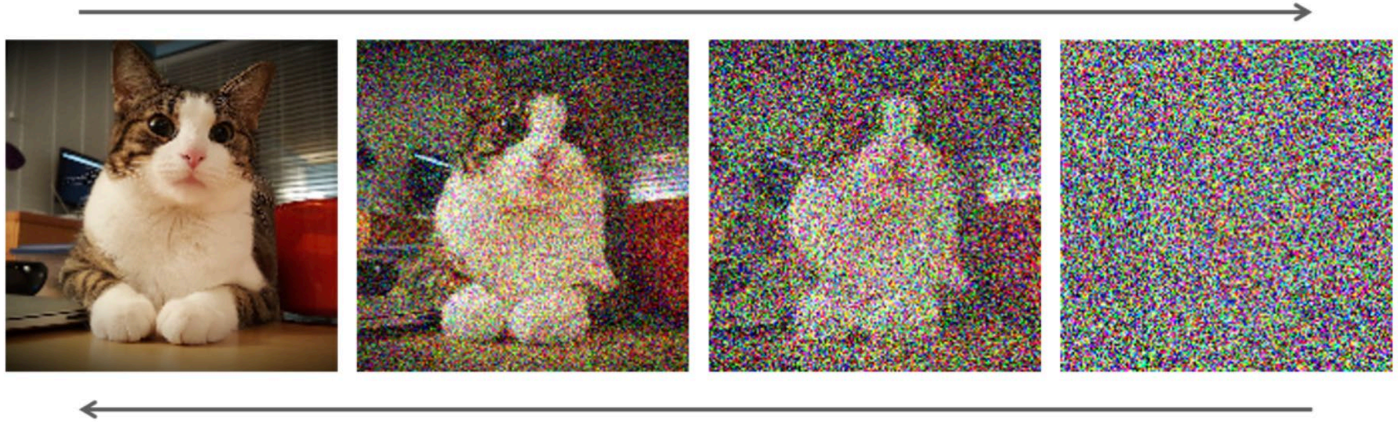


Frameworks make GPUs Easy

TensorFlow NVIDIA CUDA PyTorch

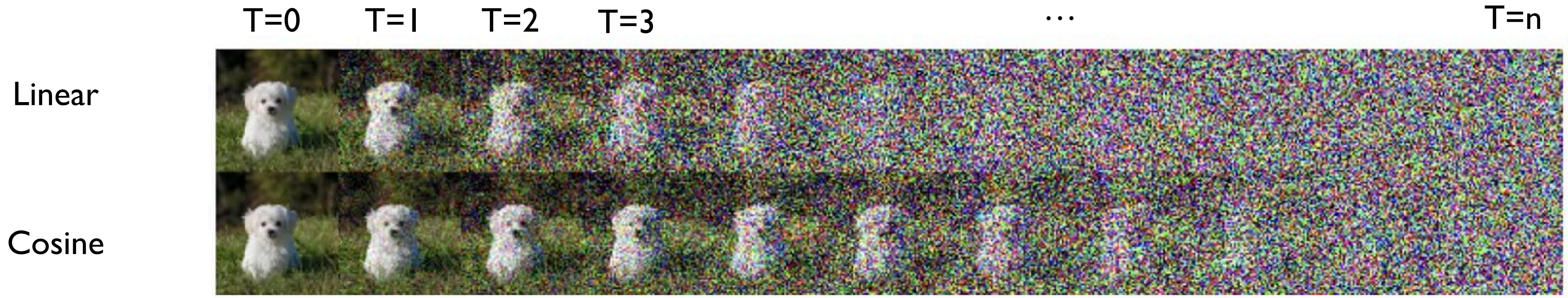
PyTorch | AMD INSTINCT

DIFFUSION MODELS



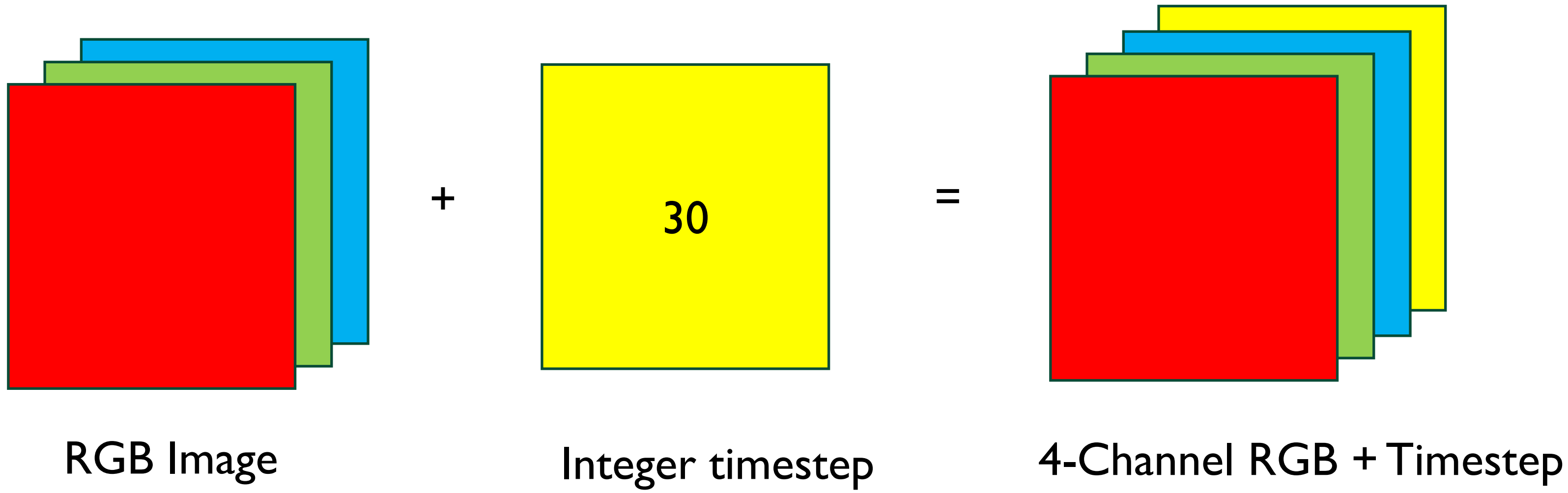
FORWARD DIFFUSION

- Define how many **time steps** will be used (common to use hundreds or more)
- Establish a **noise schedule** which describes the rate at which Gaussian noise is added

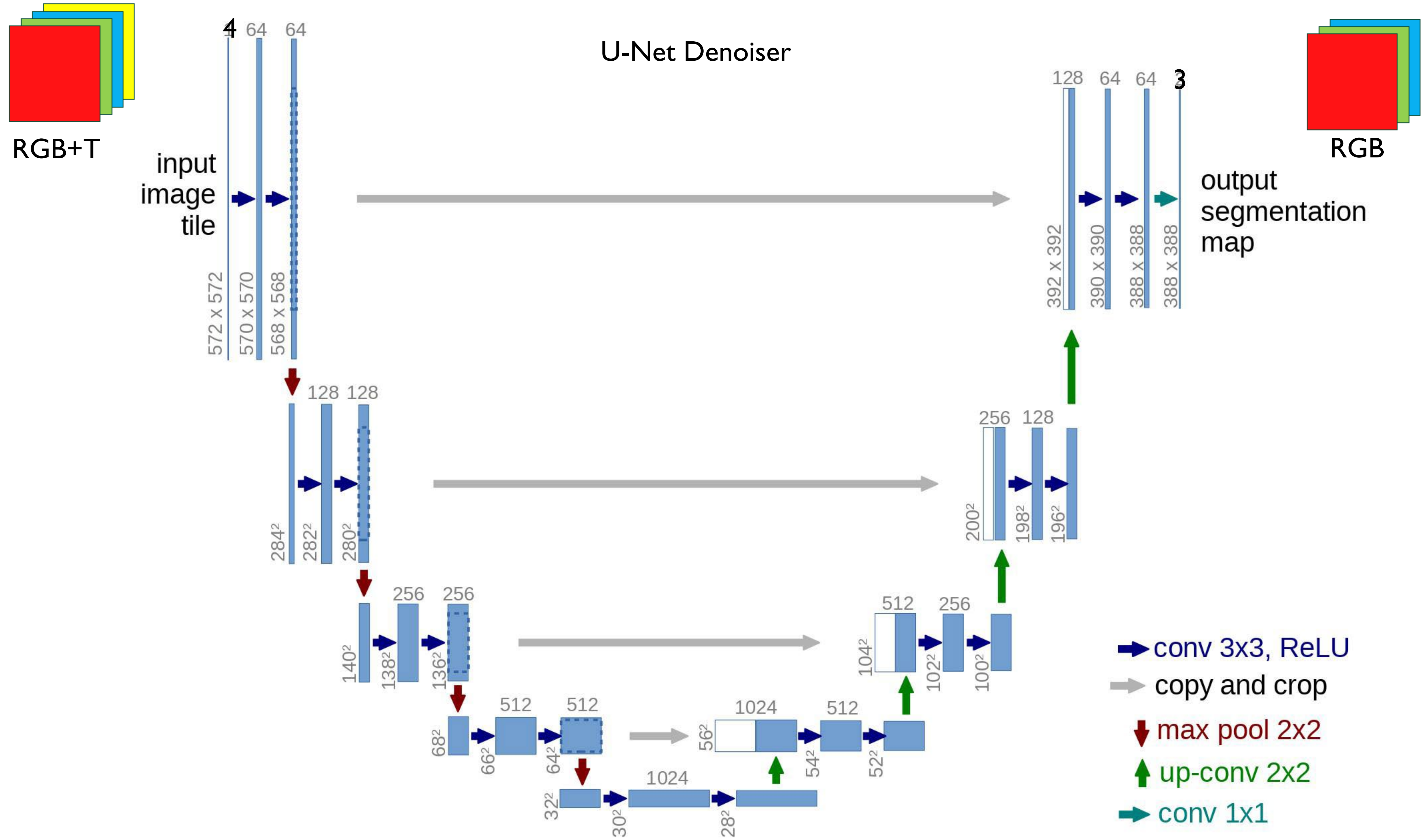


I used 100 timesteps.   Larger models like Stable Diffusion use thousands of smaller steps.
I used a cosine noise schedule.

TIME STEP ENCODING

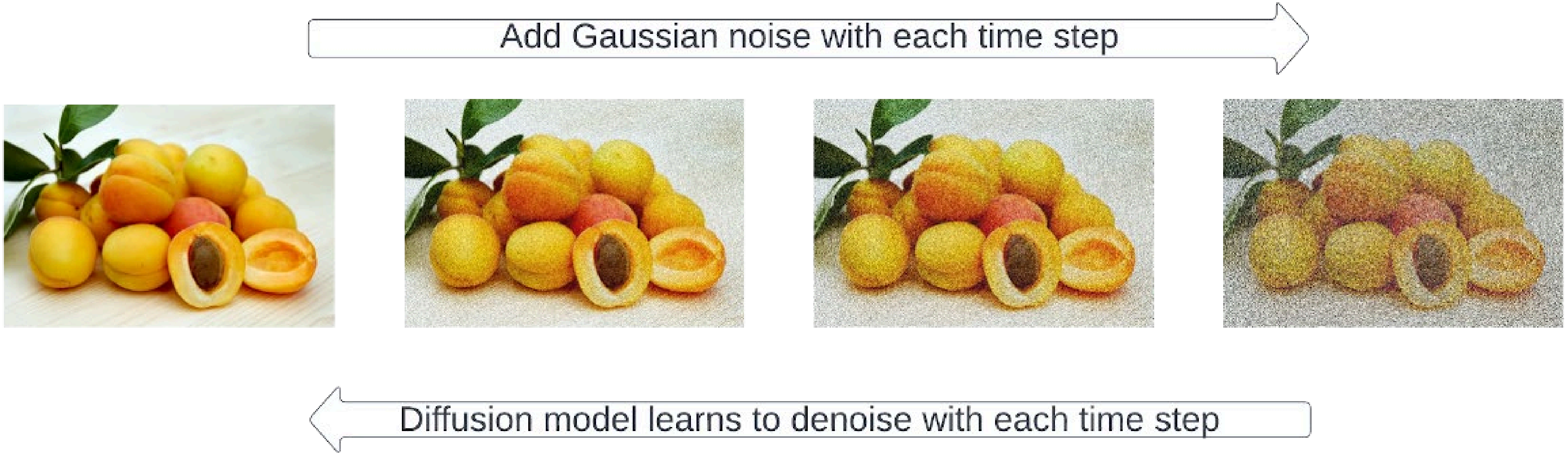


I encode timestep as another band in the image in pixel-space



U-Net Denoiser
RGB+T
RGB
4
64
64
input
image
tile
572 x 572
570 x 570
568 x 568
128
128
284^2
282^2
280^2
256
256
140^2
138^2
136^2
512
512
68^2
66^2
64^2
1024
32^2
30^2
28^2
1024
56^2
512
54^2
52^2
512
256
104^2
102^2
100^2
256
128
200^2
198^2
196^2
128
64
64
3
392 x 392
390 x 390
388 x 388
388 x 388
output
segmentation
map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

Training our Neural Network



Possible loss functions for our U-Net

$$loss_1 = MSE(pred_t, original)$$
$$loss_2 = MSE(pred_t, noisy_{t-1})$$
$$loss_3 = pred_t - noisy_t$$

Other Hyperparameters:

Epochs = 100
Timesteps = 100
Batch Size = 1250
Optimizer = Adam
Learning Rate = 0.001

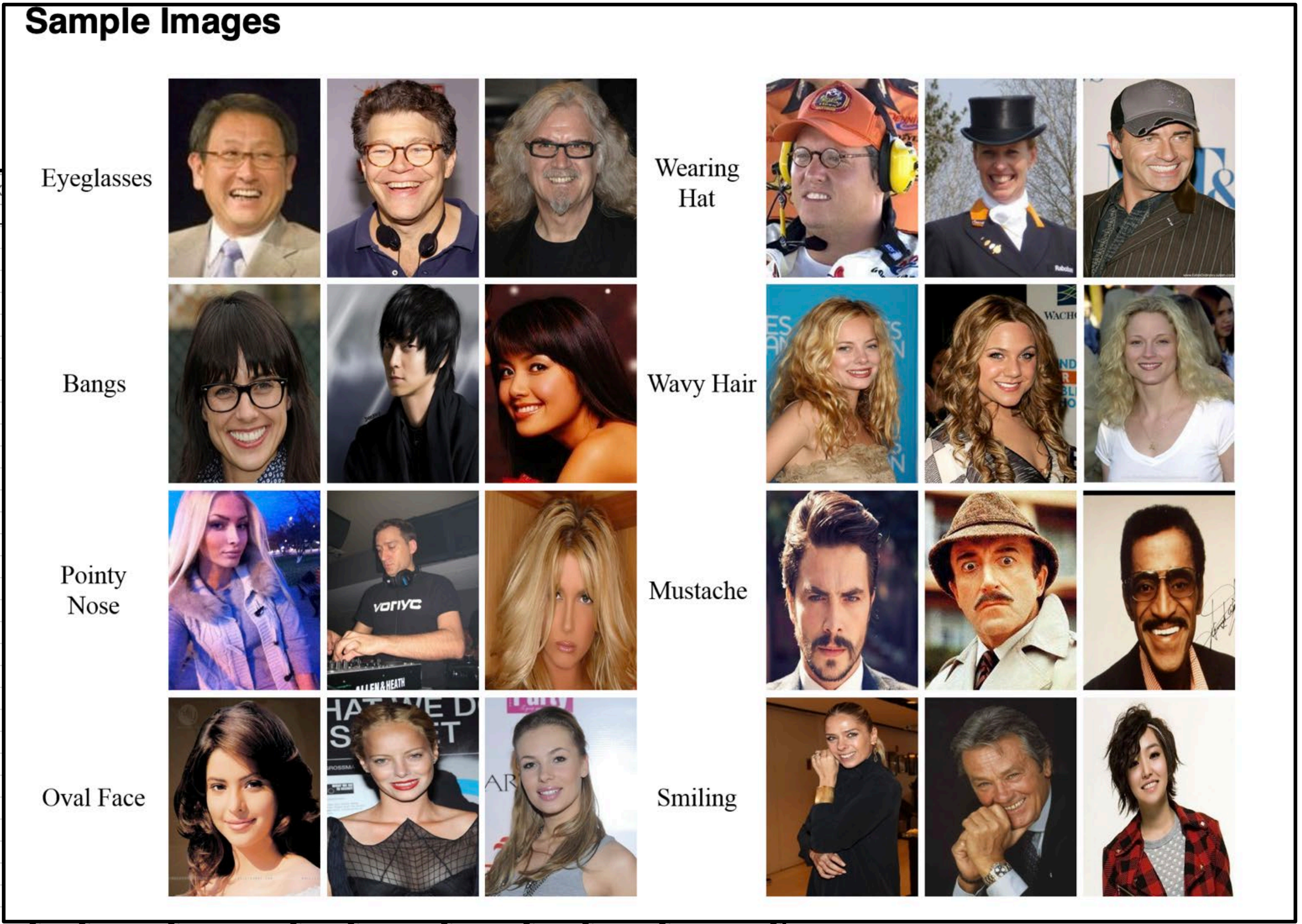
Core Training Loop

```
schedule  = cosine_schedule(TIMESTEPS)
for each Epoch:
        for each Batch b:
                for each Timestep t:
                        img = add_gaussian_noise(img, schedule(t))
                        predicted = UNet(img)
                        loss = loss_function(img, predicted)
                        backward_propagation and optimization
```

CELEB FACES ATTRIBUTES (CELEBA) DATASET

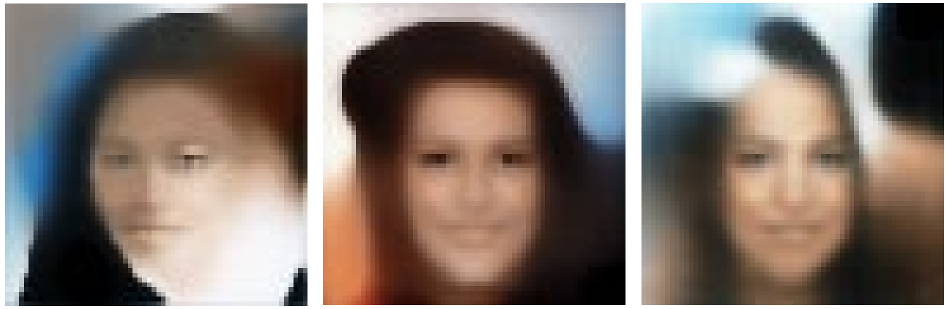
- 202,599 number of face images of various celebrities
- 10,177 unique identities, but names of identities are not given
- 40 binary attribute annotations per image
- 5 landmark locations

Images ”in the wild” <u>or</u> Cropped/Aligned



	A	B	C	D	E	F	G	H	I	J	K									
1	filename	5_o_Clock_Shadow	Arched_Eyebrows	Attractive	Bags_Under_Eyes	Bald	Bangs	Big_Lips	Big_Nose	Black_Hair	Blond_									
2	000001.jpg	-1	1	1	-1	-1	-1	-1	-1	-1										
3	000002.jpg	-1	-1	-1	1	-1	-1	-1	1	-1										
4	000003.jpg	-1	-1	-1	-1	-1	-1	1	-1	-1										
5	000004.jpg	-1	-1	1	-1	-1	-1	-1	-1	-1										
6	000005.jpg	-1	1	1	-1	-1	-1	1	-1	-1										
7	000006.jpg	-1	1	1	-1	-1	-1	1	-1	-1										
8	000007.jpg	1	-1	1	1	-1	-1	1	1	1										
9	000008.jpg	1	1	-1	1	-1	-1	1	-1	1										
10	000009.jpg	-1	1	1	-1	-1	1	1	-1	-1										
11	000010.jpg	-1	-1	1	-1	-1	-1	-1	-1	-1										
12	000011.jpg	-1	-1	1	-1	-1	-1	-1	-1	1										
13	000012.jpg	-1	-1	1	1	-1	-1	-1	-1	1										
14	000013.jpg	-1	-1	-1	-1	-1	-1	-1	-1	-1										
15	000014.jpg	-1	1	-1	-1	-1	-1	-1	1	1										
16	000015.jpg	1	-1	-1	1	-1	-1	-1	1	-1										
17	000016.jpg	1	-1	-1	1	-1	-1	-1	-1	-1										
18	000017.jpg	-1	-1	-1	-1	-1	-1	-1	-1	1										
19	000018.jpg	-1	1	-1	-1	-1	-1	-1	1	-1										
20	000019.jpg	-1	1	1	-1	-1	-1	-1	-1	-1										
21	000020.jpg	-1	-1	-1	-1	-1	-1	-1	-1	1										
22	000021.jpg	-1	-1	-1	-1	-1	-1	-1	1	-1										
23	000022.jpg	-1	1	-1	-1	-1	-1	1	-1	-1										
24	000023.jpg	1	-1	1	-1	-1	-1	-1	1	-1										
25	000024.jpg	-1	1	1	-1	-1	-1	1	-1	-1										
26	000025.jpg	1	-1	-1	1	-1	-1	-1	1	-1										
27	000026.jpg	-1	-1	1	-1	-1	-1	1	-1	-1										
28	000027.jpg	-1	1	1	-1	-1	-1	1	-1	1										
29	000028.jpg	-1	-1	1	-1	-1	-1	-1	-1	-1										
30	000029.jpg	-1	1	1	-1	-1	1	-1	-1	-1										
31	000030.jpg	-1	-1	-1	1	-1	-1	-1	1	-1										
32	000031.jpg	-1	-1	-1	1	-1	-1	-1	-1	-1										
33	000032.jpg	-1	-1	-1	1	-1	-1	-1	1	-1										
34	000033.jpg	-1	-1	1	-1	-1	-1	-1	-1	1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
35	000034.jpg	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	1

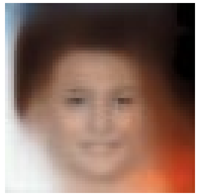
SOME PRELIMINARY OUTPUT



Oh no!

Use separate AI model for *upsampling*

64



64

SRResNet

512

512

https://github.com/twtygqyy/pytorch-SRResNet

My Model

Might not be terrific, but…

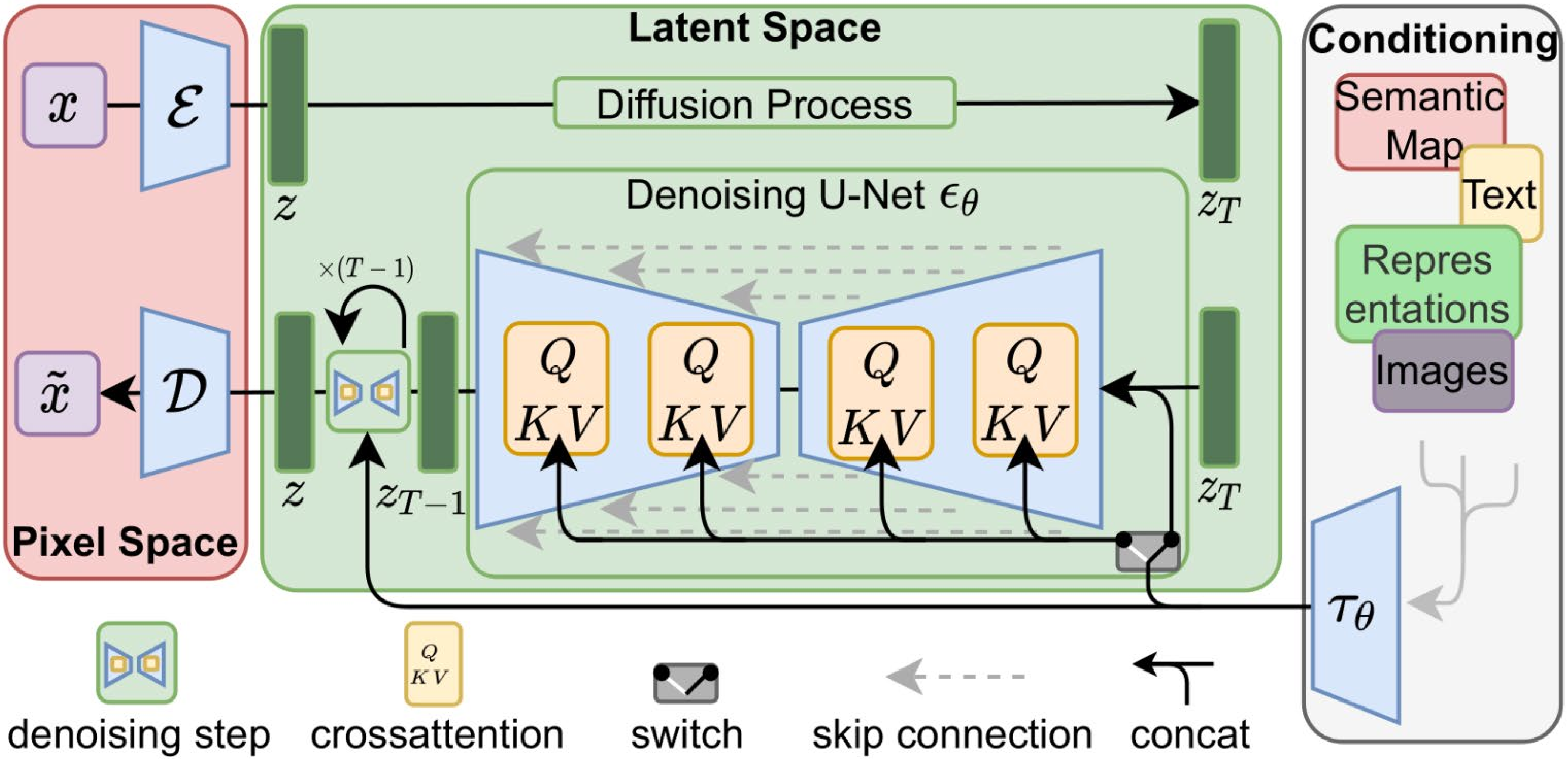
It was trained on only 5000 images for a few hours on a single RTX 4090 GPU

Stable Diffusion

Stable Diffusion was trained on 600 million captioned images

Took 256 NVIDIA A100 GPUs on Amazon Web Services a total of 150,000 GPU-hours
At a cost of $600,000

Conditioning reverse Diffusion on Text prompts



PRE-PROCESSING CELEBA DATASET

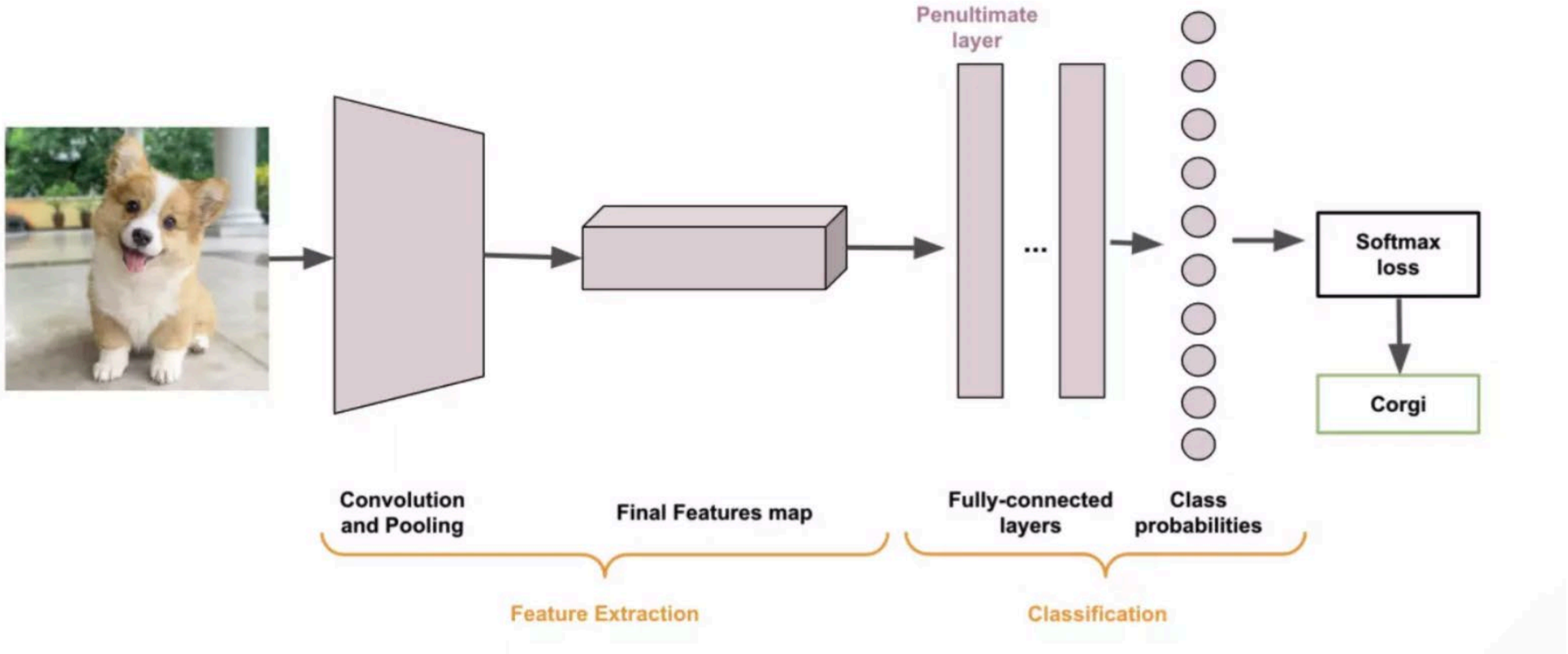
- Read first 5000 annotations into PANDAS dataframe (easy!)
- For each image, get the heading names for positive attributes
- Convert heading names into a text prompt:

  e.g. "Photo of person <attribute_x>, <attribute_y>, <attribute_z>, …"

  e.g. "Photo of person bushy eyebrows, beard, mouth slightly open, wearing hat."
- Crop the largest square from the image, then resize to 64x64x3 numpy array
- Use OpenAI CLIP model to find the image embeddings and text embeddings for every image/prompt pair.
- Create a 5000 element Python list of 4-tuples:
  - (filename, 64x64xRGB image_array, image_embedding, prompt_embedding)
- Pickle list to a file we can quicky load into ***memory*** when we train our model!

OPENAI CLIP MODEL *(CONTRASTIVE LANGUAGE–IMAGE PRE-TRAINING)*

- Open source/weights multi-modal AI model trained on image, caption pairs
- Shared embedding space!
- Use transformer model (GPT-2) to create token embeddings from text
- Use vision transformer (VIT) to create token embeddings from images



CLIP Examples

https://openai.com/research/clip

✓ a photo of **guacamole**, a type of food.

✗ a photo of **ceviche**, a type of food.

✗ a photo of **edamame**, a type of food.

✗ a photo of **tuna tartare**, a type of food.

✗ a photo of **hummus**, a type of food.

✓ a photo of a **2012 honda accord coupe**.

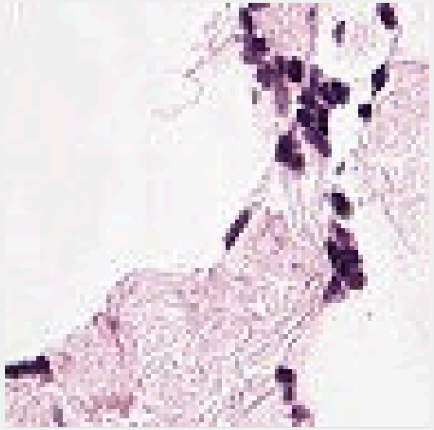
✗ a photo of a **2012 honda accord sedan**.

✗ a photo of a **2012 acura tl sedan**.

✗ a photo of a **2012 acura tsx sedan**.

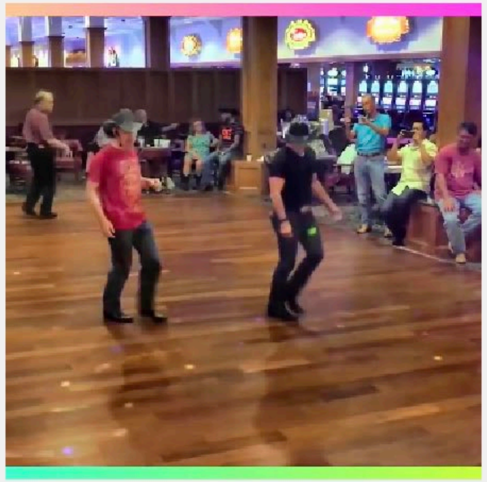
✗ a photo of a **2008 acura tl type-s**.

healthy lymph node tissue (77.2%) Ranked 2 out of 2 labels



✗ this is a photo of **lymph node tumor tissue**

✓ this is a photo of **healthy lymph node tissue**



✓ a photo of **country line dancing**.

✗ a photo of **square dancing**.

✗ a photo of **swing dancing**.

✗ a photo of **dancing charleston**.

✗ a photo of **salsa dancing**.

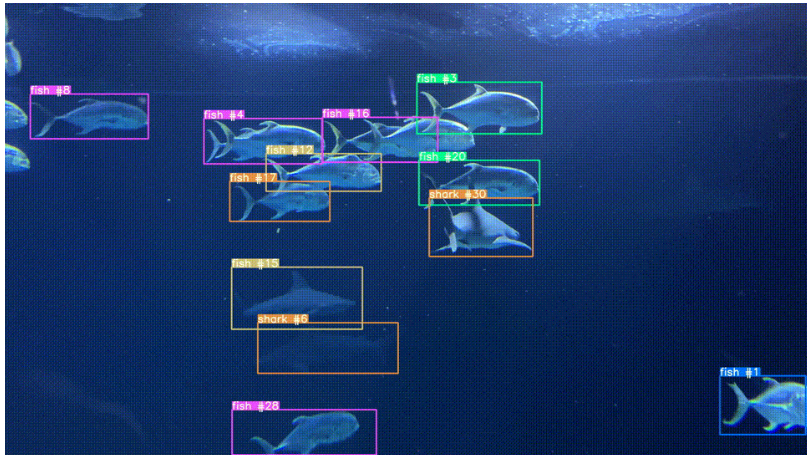
USING CLIP IS TRIVIAL

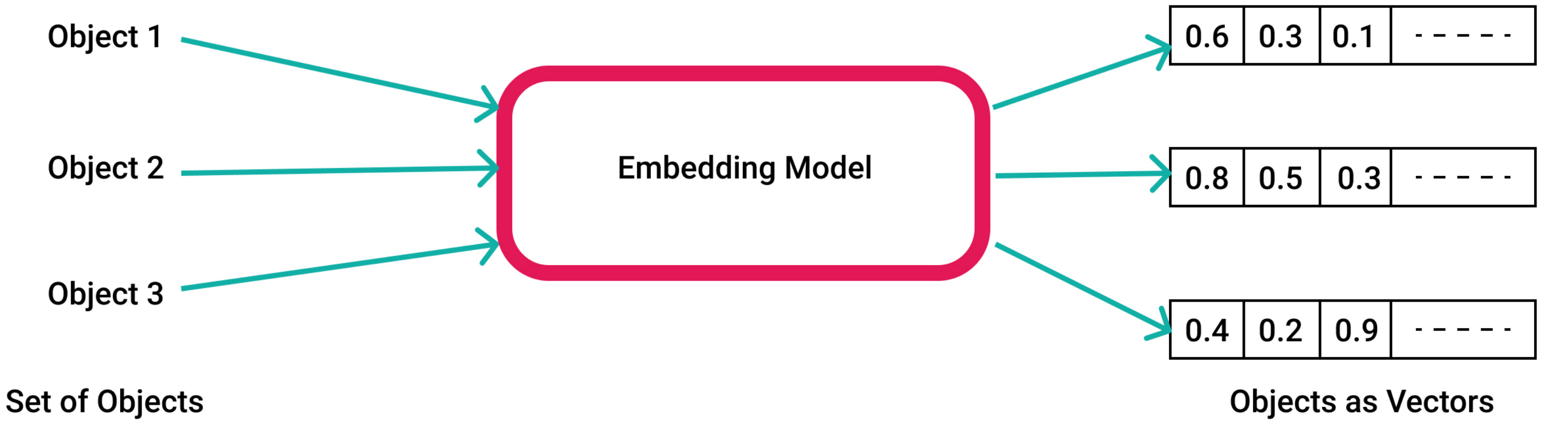
```python
import torch
import clip
from PIL import Image

device = "cuda" if torch.cuda.is_available() else "cpu"
model, preprocess = clip.load("ViT-B/32", device=device)

image = preprocess(Image.open("CLIP.png")).unsqueeze(0).to(device)
text = clip.tokenize(["a diagram", "a dog", "a cat"]).to(device)

with torch.no_grad():
    image_features = model.encode_image(image)
    text_features = model.encode_text(text)

    logits_per_image, logits_per_text = model(image, text)
    probs = logits_per_image.softmax(dim=-1).cpu().numpy()

print("Label probs:", probs)  # prints: [[0.9927937  0.00421068 0.00299572]]
```

https://github.com/openai/CLIP

Zero-shot classifications!
Conditioning Generative AI (DALL-E)
Generating captions for images or video
Image similarity search
Content Moderation
Object Tracking

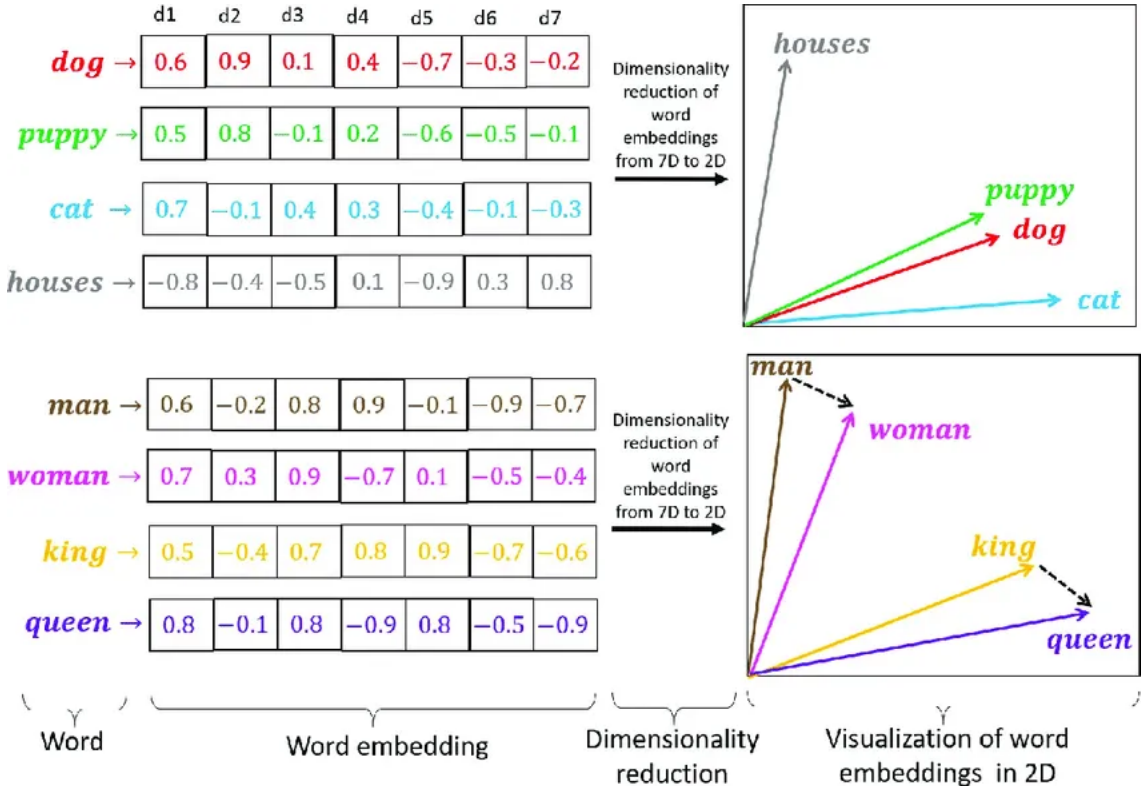
A couple of people standing next to an elephant.





CLIP uses vectors with 512 dimensions

GPT3 (Davinci) uses 12888 dimensions



Vector embeddings capture the deeper semantic context of a word or text chunk…or image…or anything.

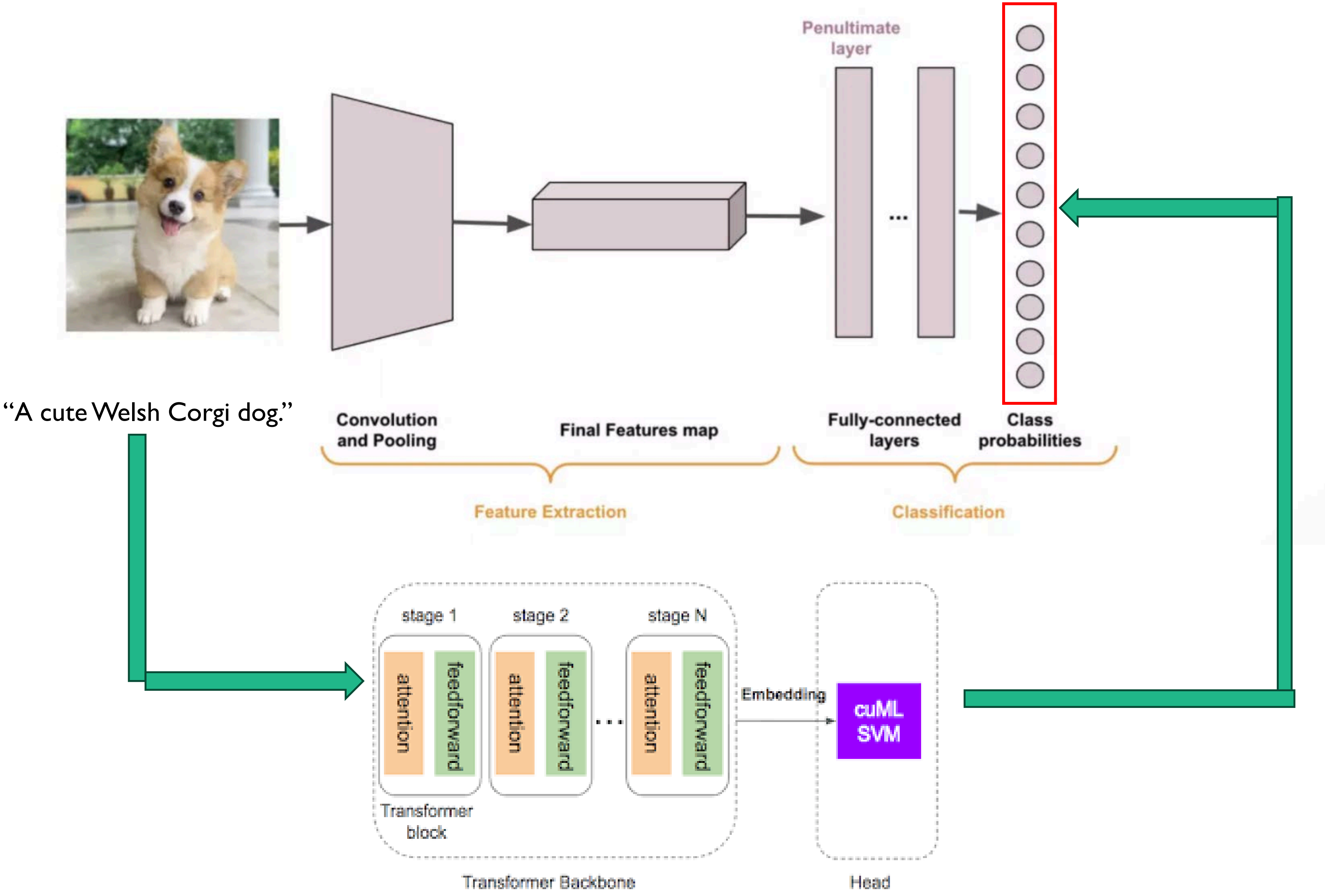
The semantics of an object are defined by its multi-dimensional and multi-scale co-occurrence and relationships with other objects in the training data

Semantic vector embeddings are learned from vast amounts of data.

400,000,000 (image, text) pairs
CLIP was trained on 256 large GPUs for 2 weeks.

One way to train a multi-modal embedding layer

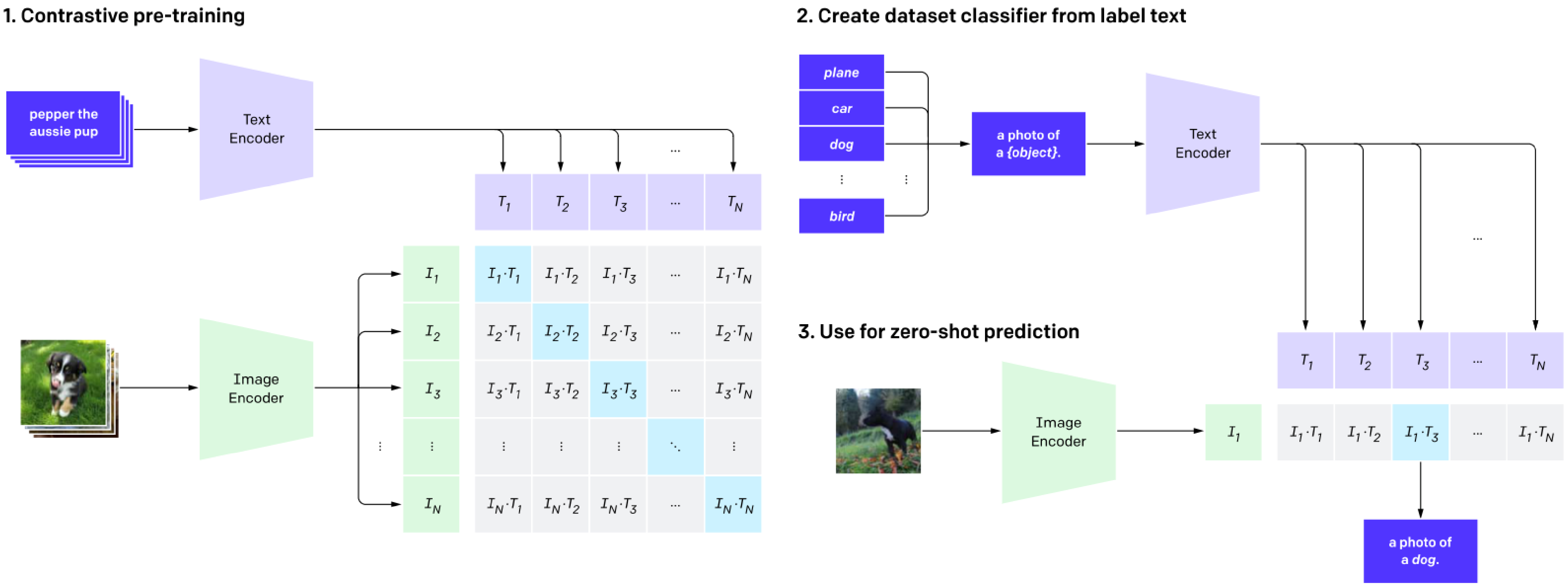
Size of Embedding Vector

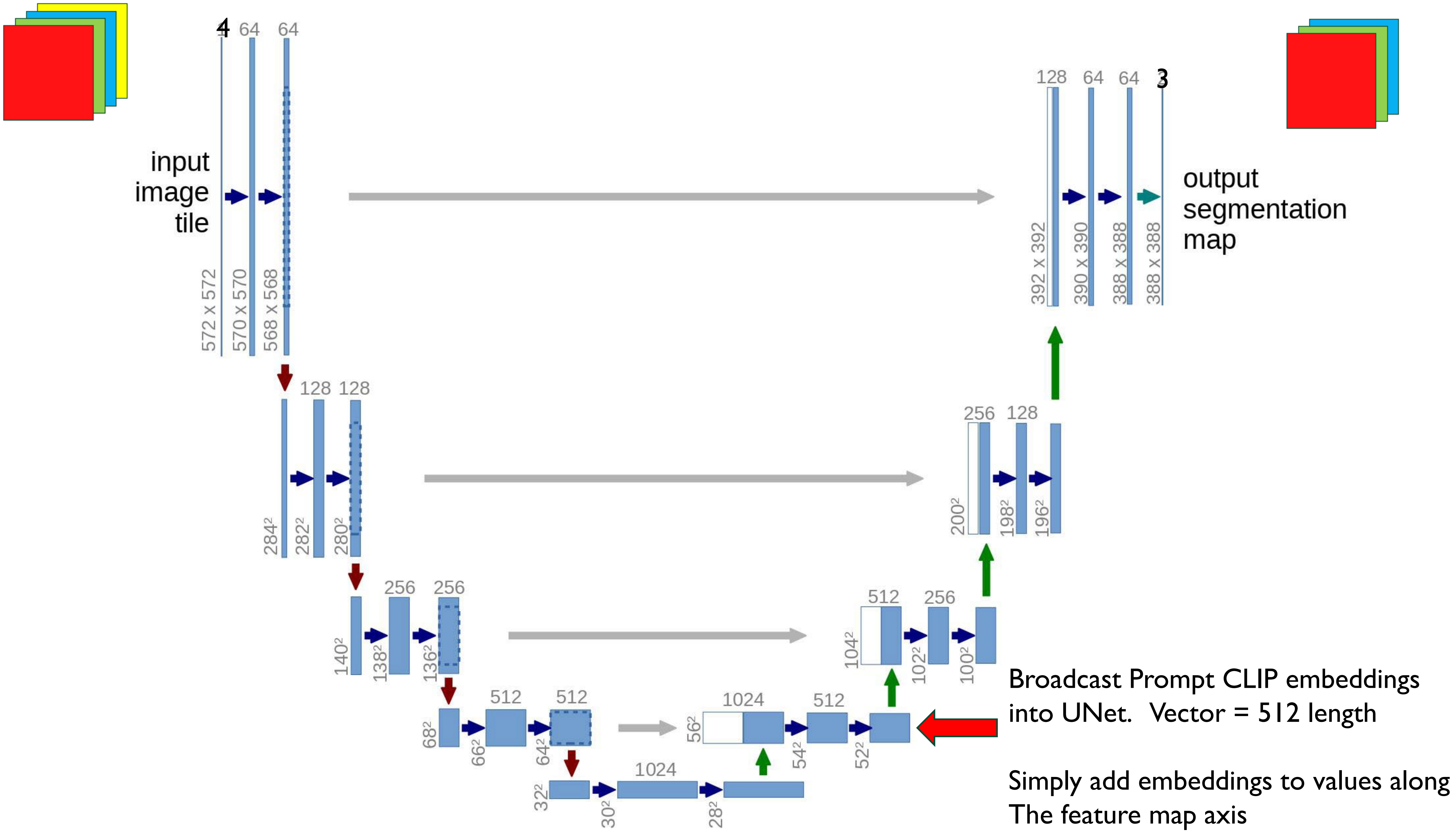


Learning your semantic embeddings first (GPT)

Train your CNN to predict matching embedding vectors

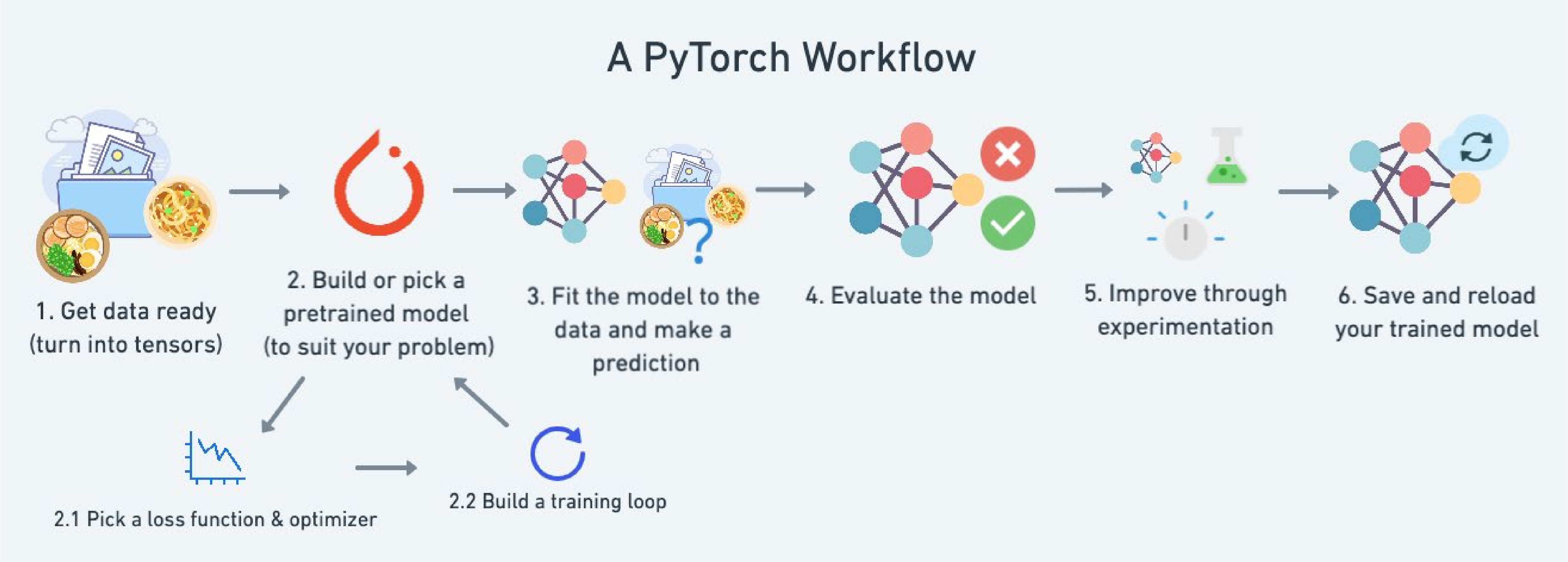
How CLIP handle multi-model semantic embeddings





4
64
64
input image tile
572 x 572
570 x 570
568 x 568
128
128
284²
282²
280²
256
256
140²
138²
136²
512
512
68²
66²
64²
1024
32²
30²
28²
1024
56²
54²
512
52²
512
256
104²
102²
100²
256
128
200²
198²
196²
128
64
64
3
392 x 392
390 x 390
388 x 388
388 x 388
output segmentation map
Broadcast Prompt CLIP embeddings into UNet. Vector = 512 length
Simply add embeddings to values along The feature map axis

Now for Code



github.com/sheneman/diffusion