EEP 596: LLMs: From Transformers to GPT || Lecture 13

Dr. Karthik Mohan

Univ. of Washington, Seattle

February 24, 2025

Deep Learning and Transformers References

Deep Learning

Great reference for the theory and fundamentals of deep learning: Book by Goodfellow and Bengio et al Bengio et al

Deep Learning History

Embeddings

SBERT and its usefulness

SBert Details

Instacart Search Relevance

Instacart Auto-Complete

Attention

Illustration of attention mechanism

Generative AI References

Prompt Engineering

Prompt Design and Engineering: Introduction and Advanced Methods

Retrieval Augmented Generation (RAG)

Toolformer
RAG Toolformer explained

Misc GenAI references

Time-Aware Language Models as Temporal Knowledge Bases

Generative AI references

Stable Diffusion

Diffusion Explainer: Visual Explanation for Text-to-image Stable Diffusion

The Illustrated Stable Diffusion

House Keeping

- Mini-Project 3 coming up early this week
- Combination of Stable Diffusion modeling + working with open-source Llama3-1b model (Kaggle Contest) MP3
- LoRA fine-tuning is an important skill to add to your tool kit
- Any questions/concerns?

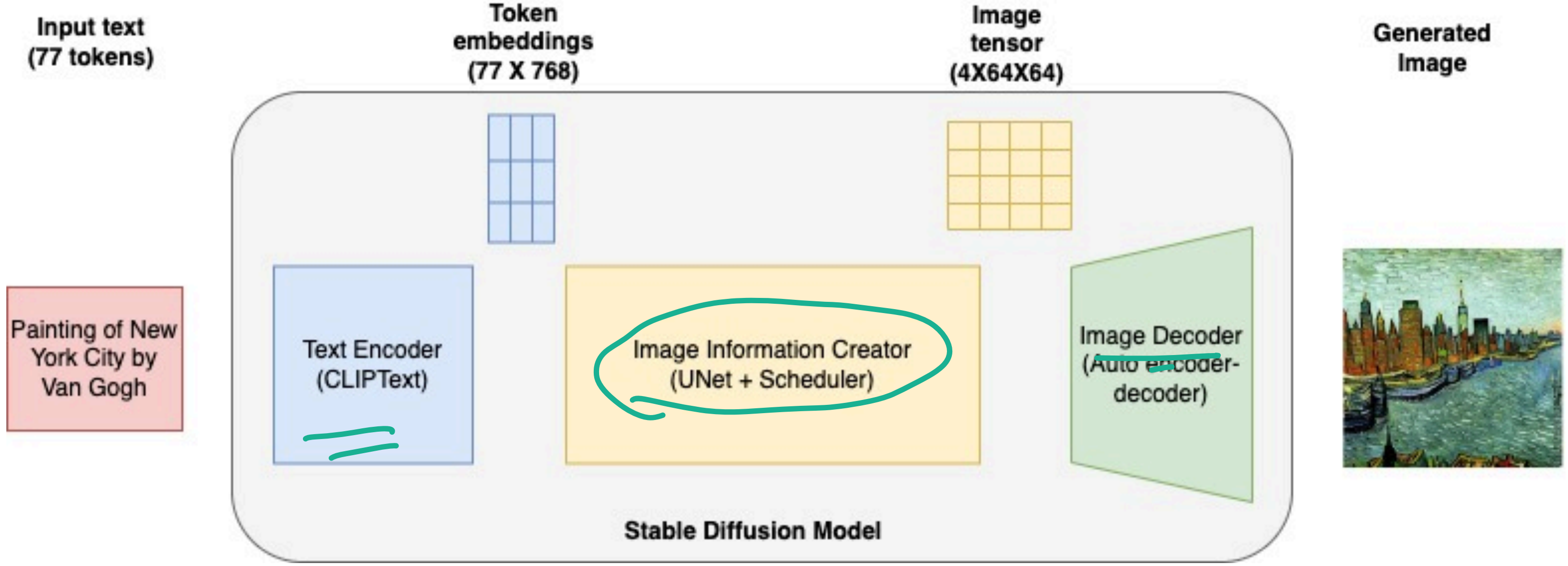
Previous Lecture

- Design of Chatbots
- Introudction to Foundation Vision Models

Today's lecture

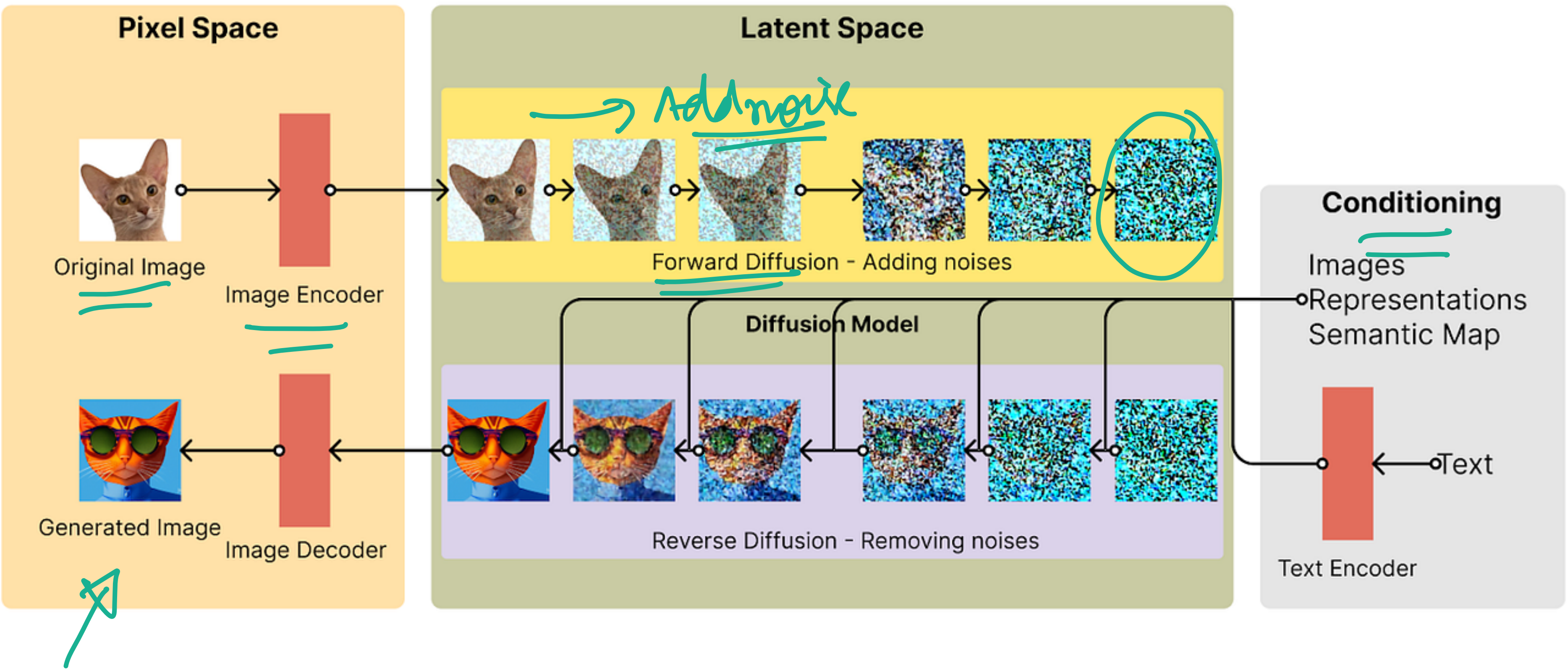
- Image Foundation Models
- Notebook walk through
- Auto Encoders and Stable Diffusion

Foundation Model - Stable Diffusion (Text2Image)

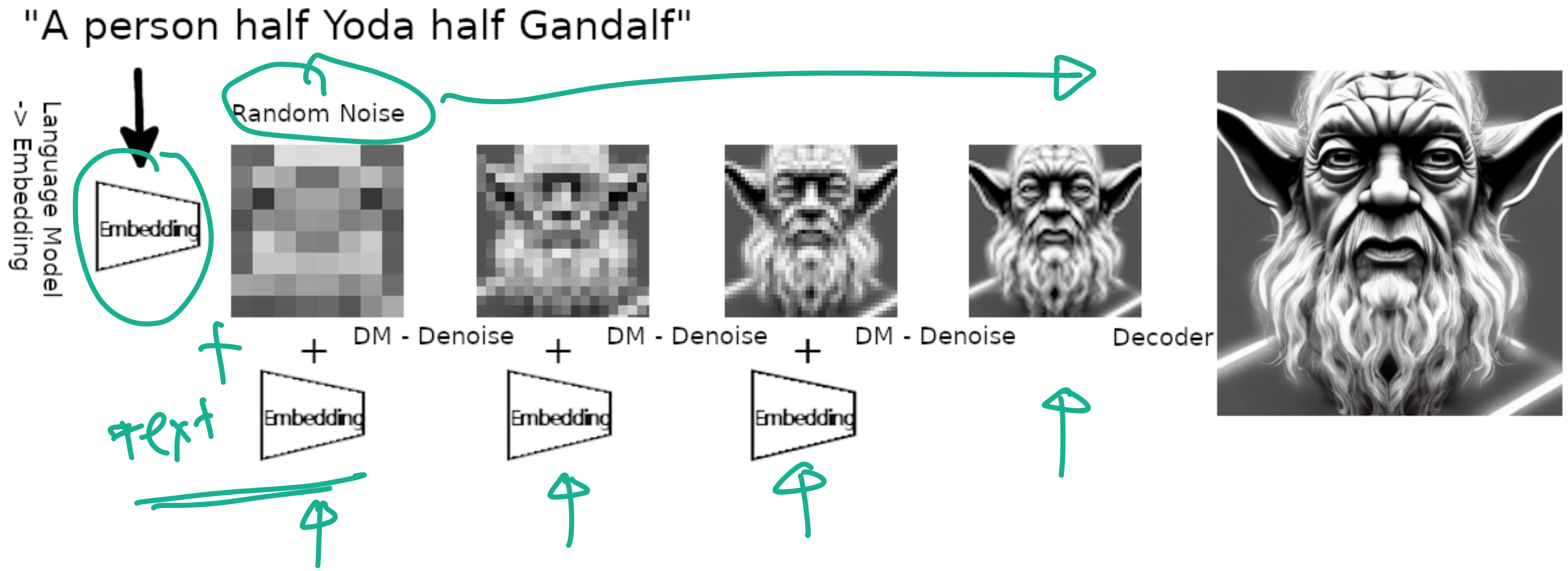


Foundation Model - Stable Diffusion (Text2Image)

Start here



Foundation Model - Stable Diffusion (Text2Image)

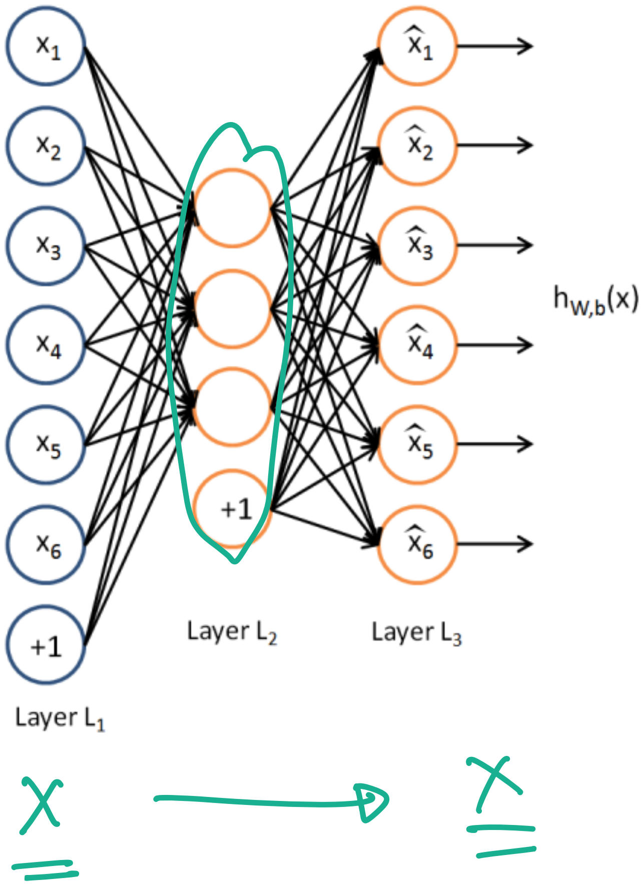


Reference

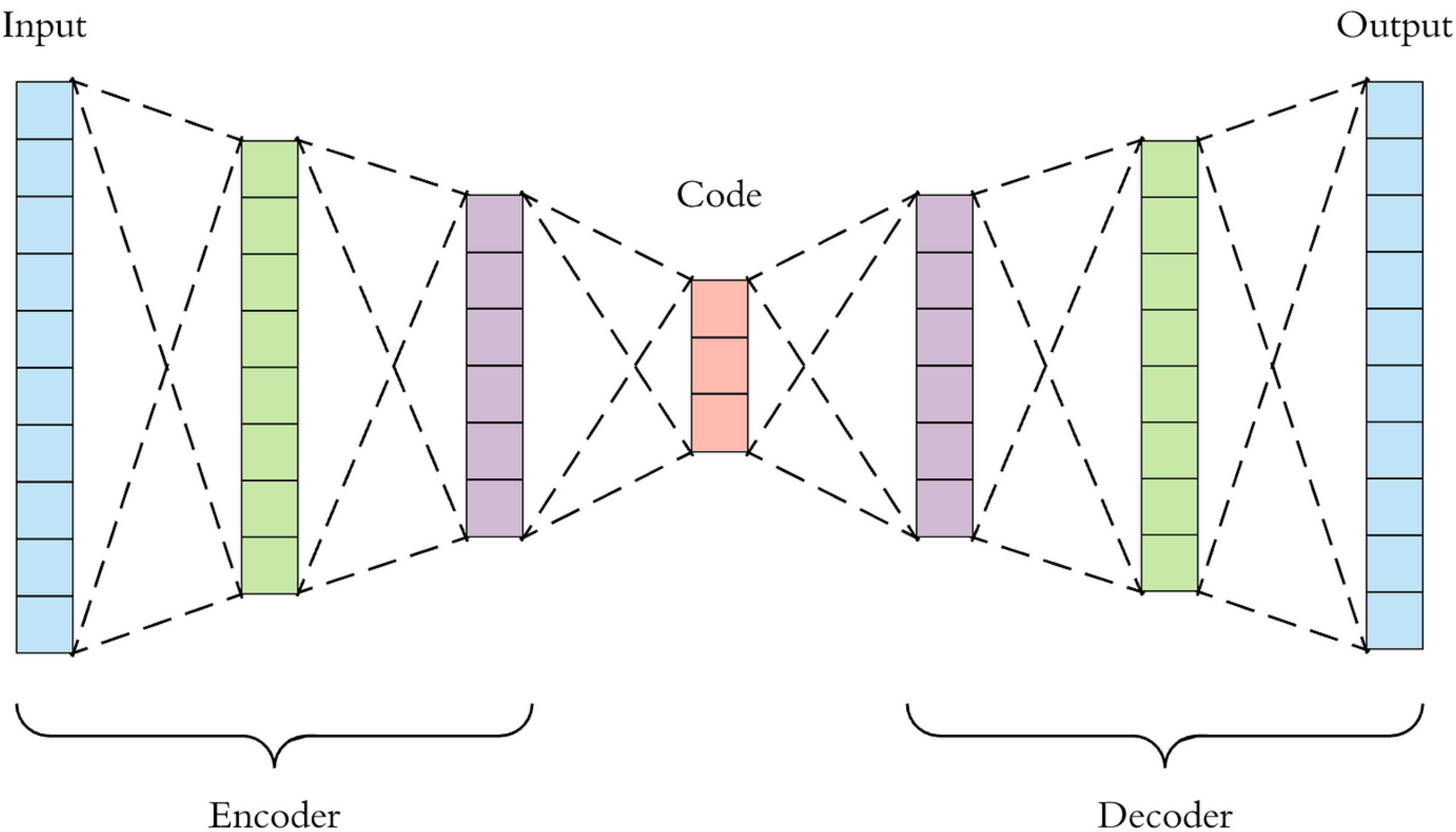
Stable Diffusion Explained

- Based on the concept of "de-noising auto encoders" and the use of text prompt to *guide the de-noising*
- Stable diffusion is also trained to successfully de-noise and increase the resolution of the image using text guidance

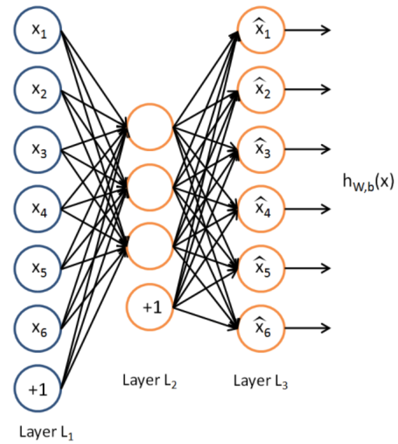
Auto Encoders

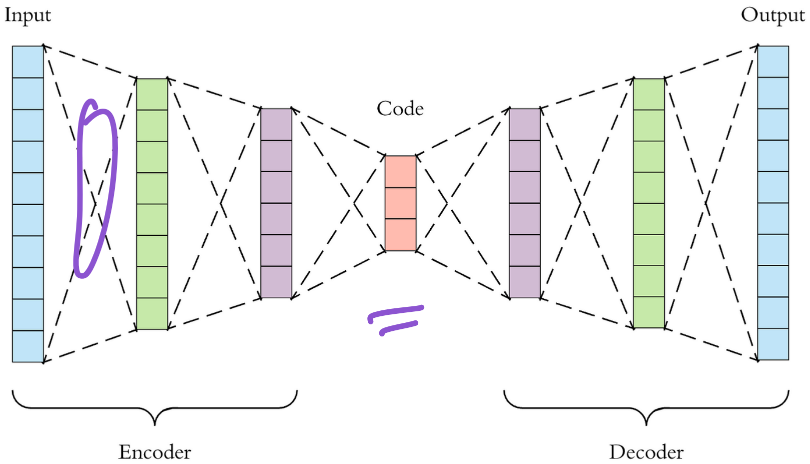


Deep Auto Encoders



PCA vs Auto-Encoders





AEs can use custom compression like convolution

ICE #1

$$\underline{\underline{X}} = \underline{\underline{U}}\Sigma V^T$$

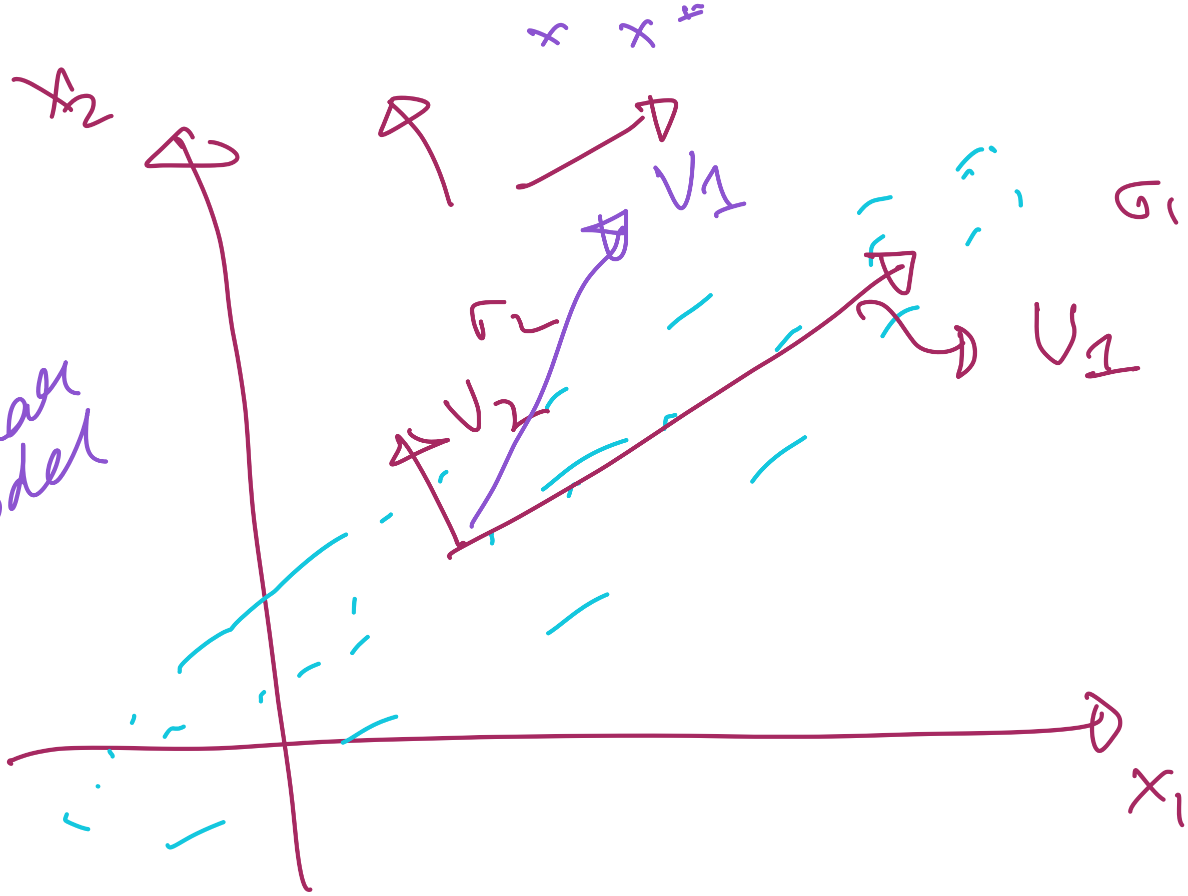
Data Matrix — Principal component — significance of the components

PCA vs Auto Encoder

Which of the following statements are true ?

1. Both PCA and Auto Encoders serve the purpose of dimensionality reduction ✓
2. They are both linear models but one uses a neural nets architecture and the other is based on projections ✗
3. PCA is robust to outliers while Auto Encoders are not ✗
4. Auto Encoders can compress images better than PCA ✓

PCA
- linear Model



AutoEncoders and Dimensionality Reduction

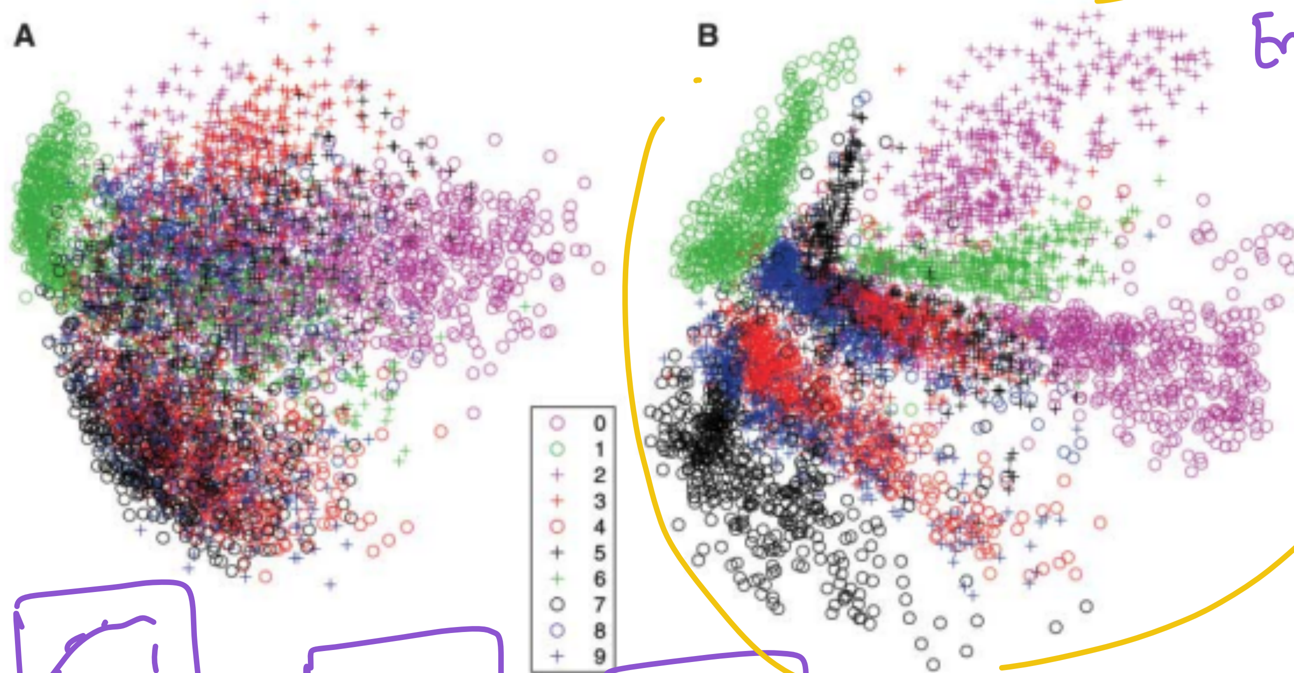
Visualization Performance

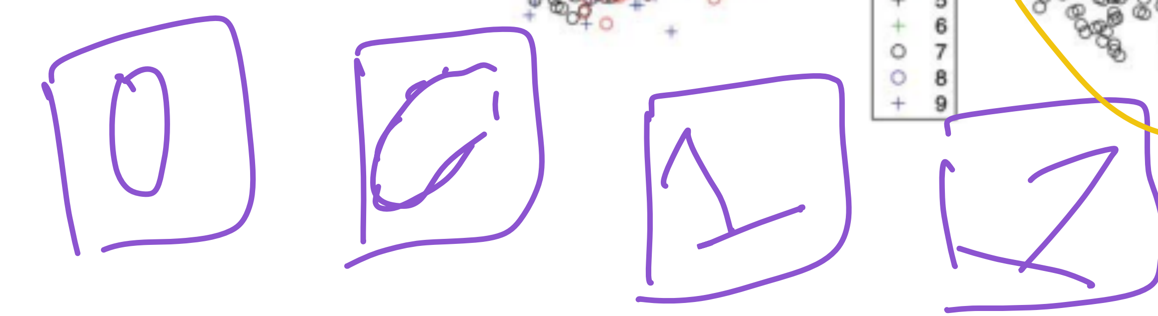
Auto Encoder Reference Paper

AutoEncoders and Dimensionality Reduction

Reading Reference for AE Dimensionality Reduction

Fig. 3. (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (8).

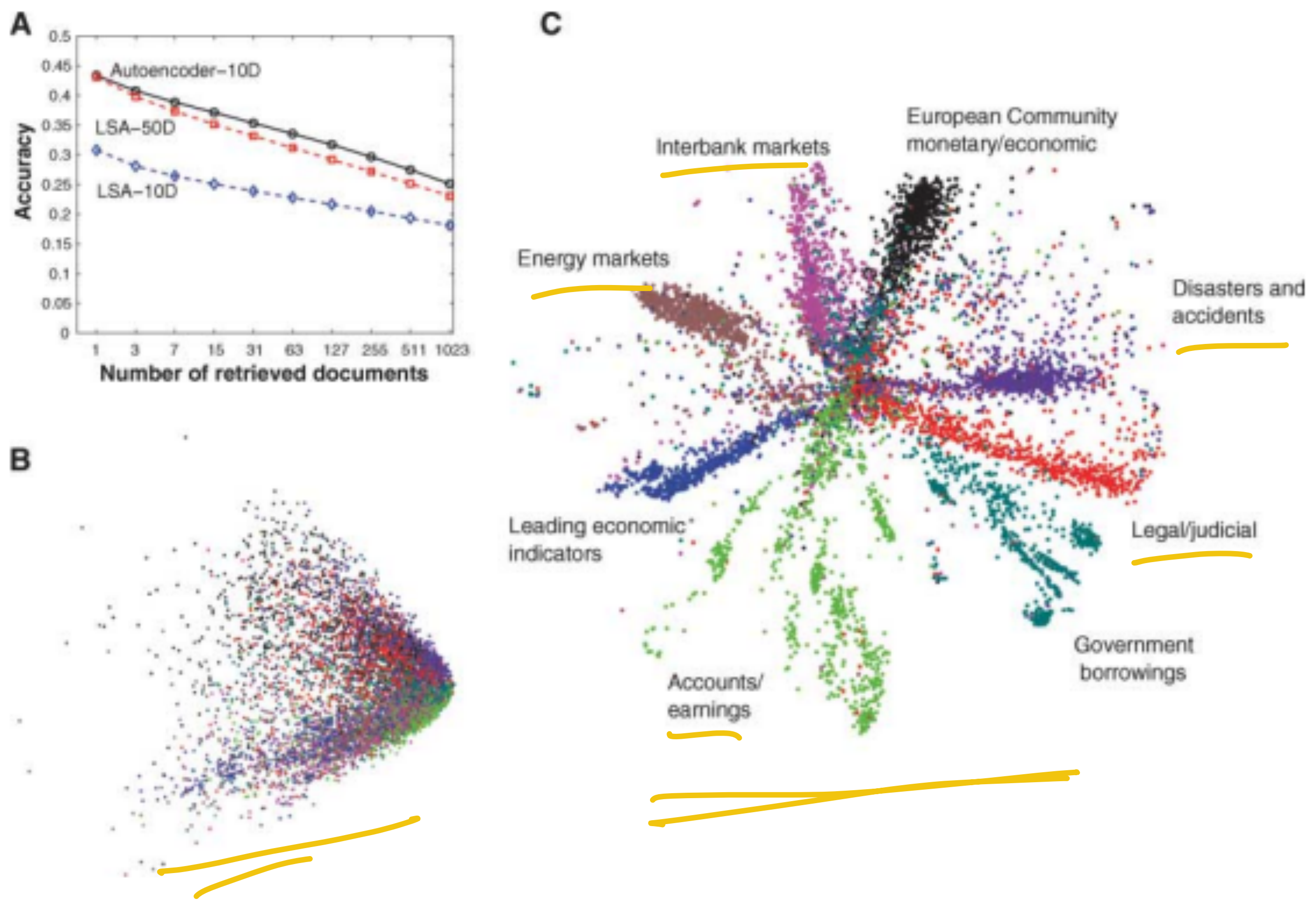




AutoEncoders and Dimensionality Reduction

Reading Reference for AE Dimensionality Reduction

Fig. 4. **(A)** The fraction of retrieved documents in the same class as the query when a query document from the test set is used to retrieve other test set documents, averaged over all 402,207 possible queries. **(B)** The codes produced by two-dimensional LSA. **(C)** The codes produced by a 2000-500-250-125-2 autoencoder.



AutoEncders Summary

1. Auto-Encoders are a method for dimensionality reduction and can do better than PCA for visualization

AutoEncders Summary

1. Auto-Encoders are a method for dimensionality reduction and can do better than PCA for visualization
2. Use Neural Networks architecture and hence can encode non-linearity in the embeddings

AutoEncders Summary

1. Auto-Encoders are a method for dimensionality reduction and can do better than PCA for visualization
2. Use Neural Networks architecture and hence can encode non-linearity in the embeddings
3. Anything else?

AutoEncders Summary

1. Auto-Encoders are a method for dimensionality reduction and can do better than PCA for visualization
2. Use Neural Networks architecture and hence can encode non-linearity in the embeddings
3. Anything else?
4. Auto Encoders can learn convolutional layers instead of dense layers - Better for images! More flexibility!!

ICE #2: Loss Function for Auto-Encoders

What is the loss function used to train Deep Auto-Encoders?

1. Logistic Loss
2. Quadratic Loss
3. Triplet Loss
4. Cross-Entropy Loss

Removing obstacles in images

Deep AE

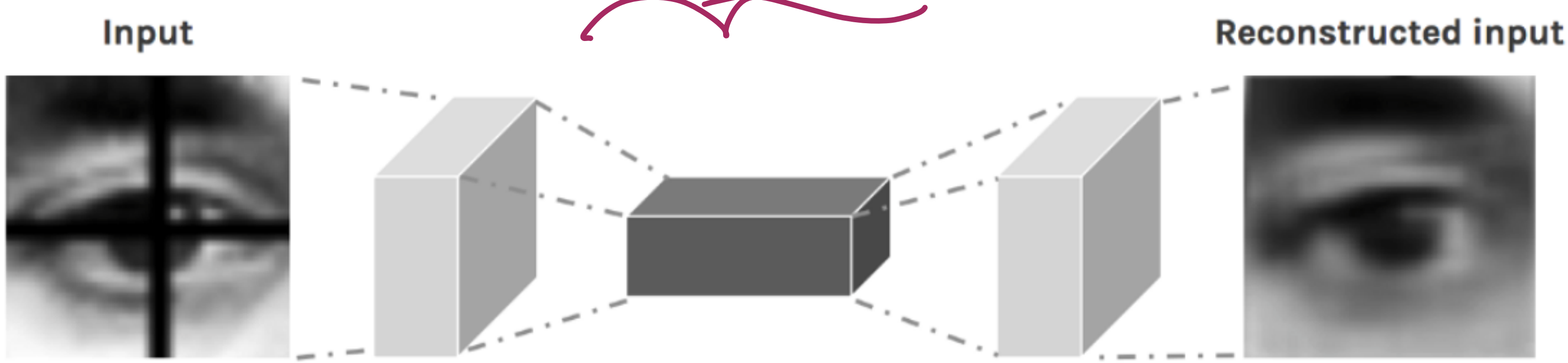


Figure 12: Reconstructed image from missing image [14]

obstacle
occlusion

Targetted De-noising

Removing obstacles in images

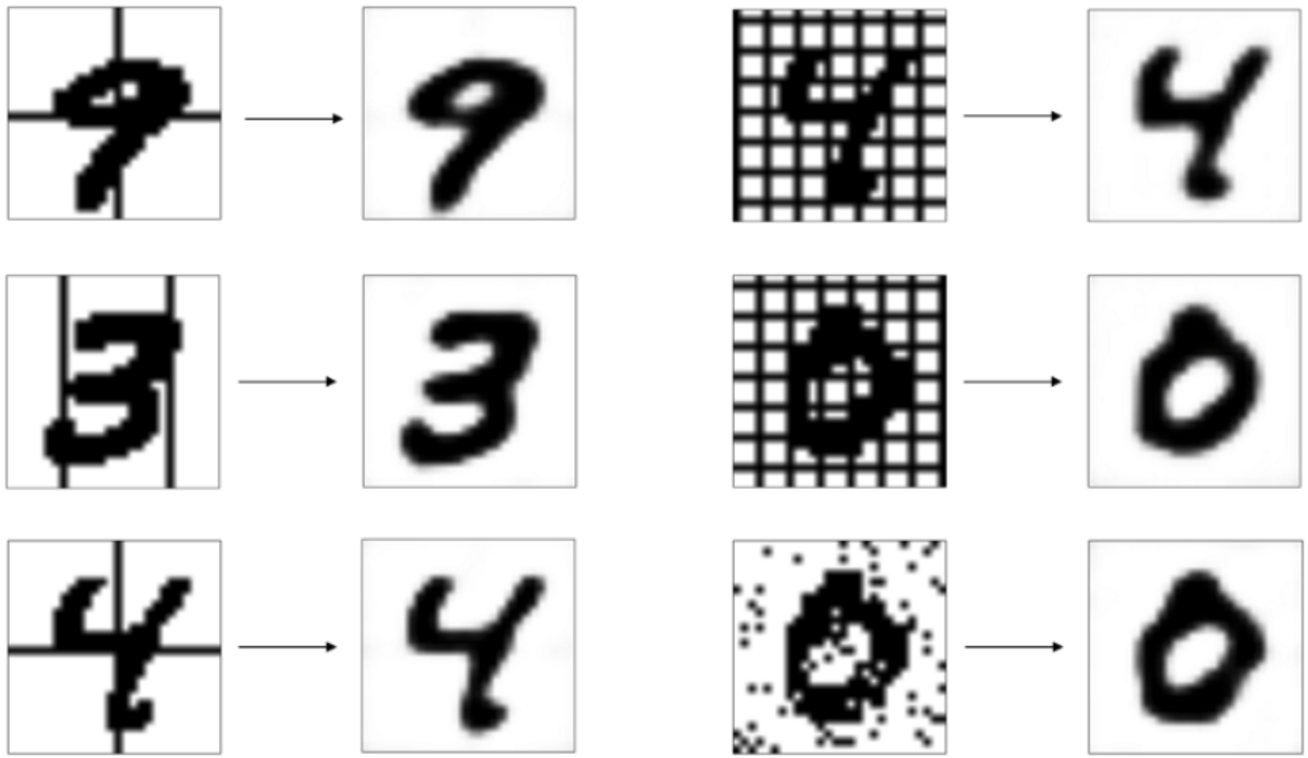
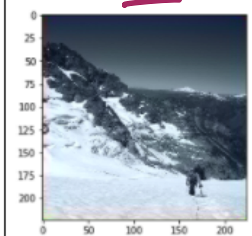
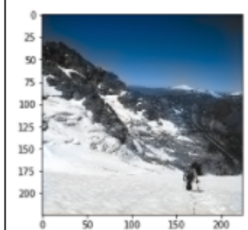
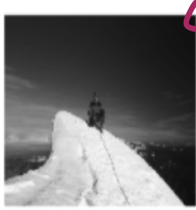
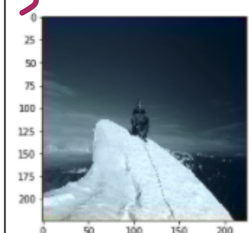
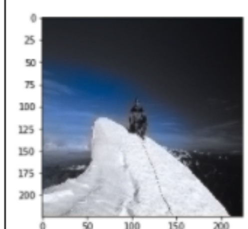
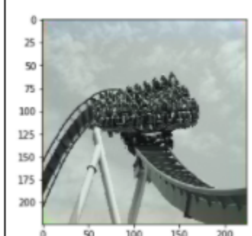
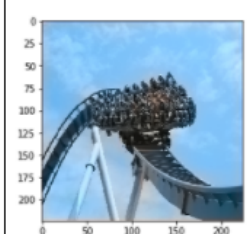
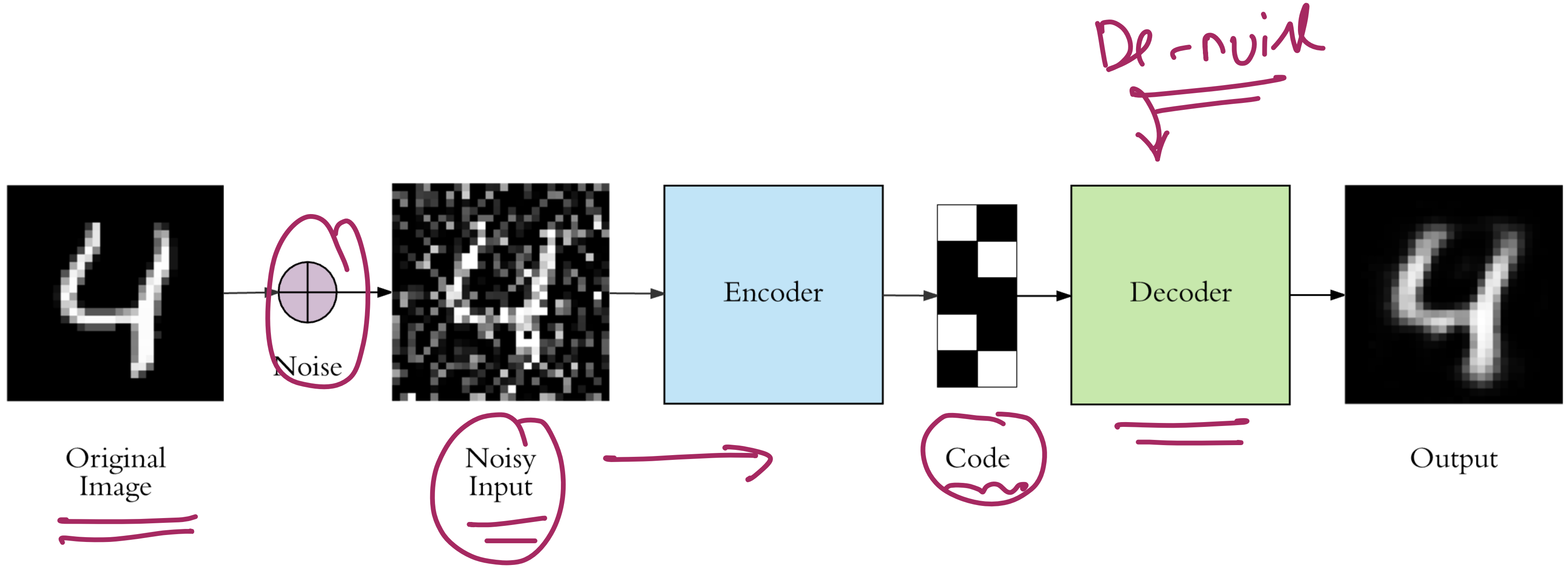


Figure 13: Source [15]

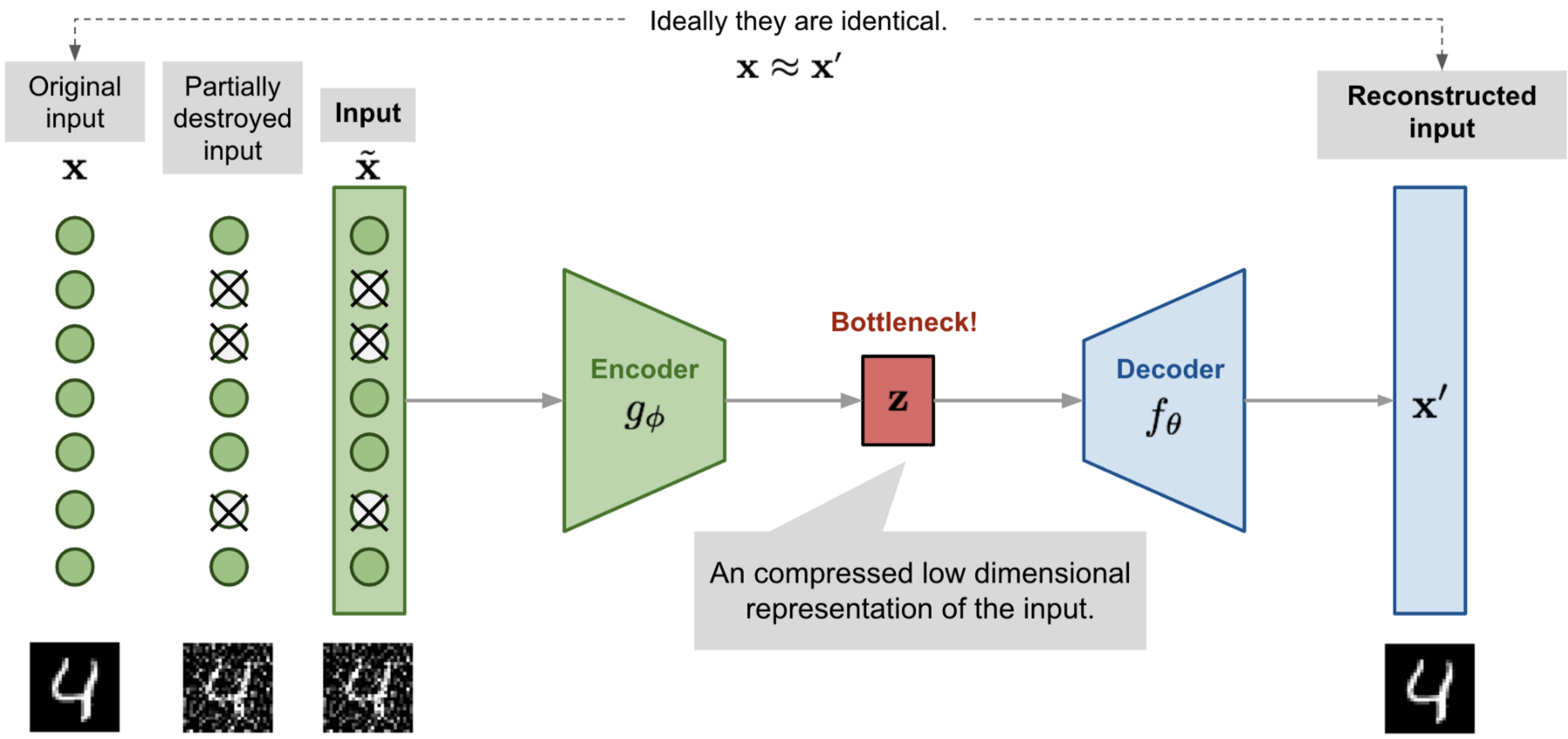
Coloring Images

Gray Image	Vanilla Autoencoder	Merge Model (YCbCr)	Merge Model (LAB)	Original
				
				
				

De-noising Auto Encoders



De-noising Auto Encoders

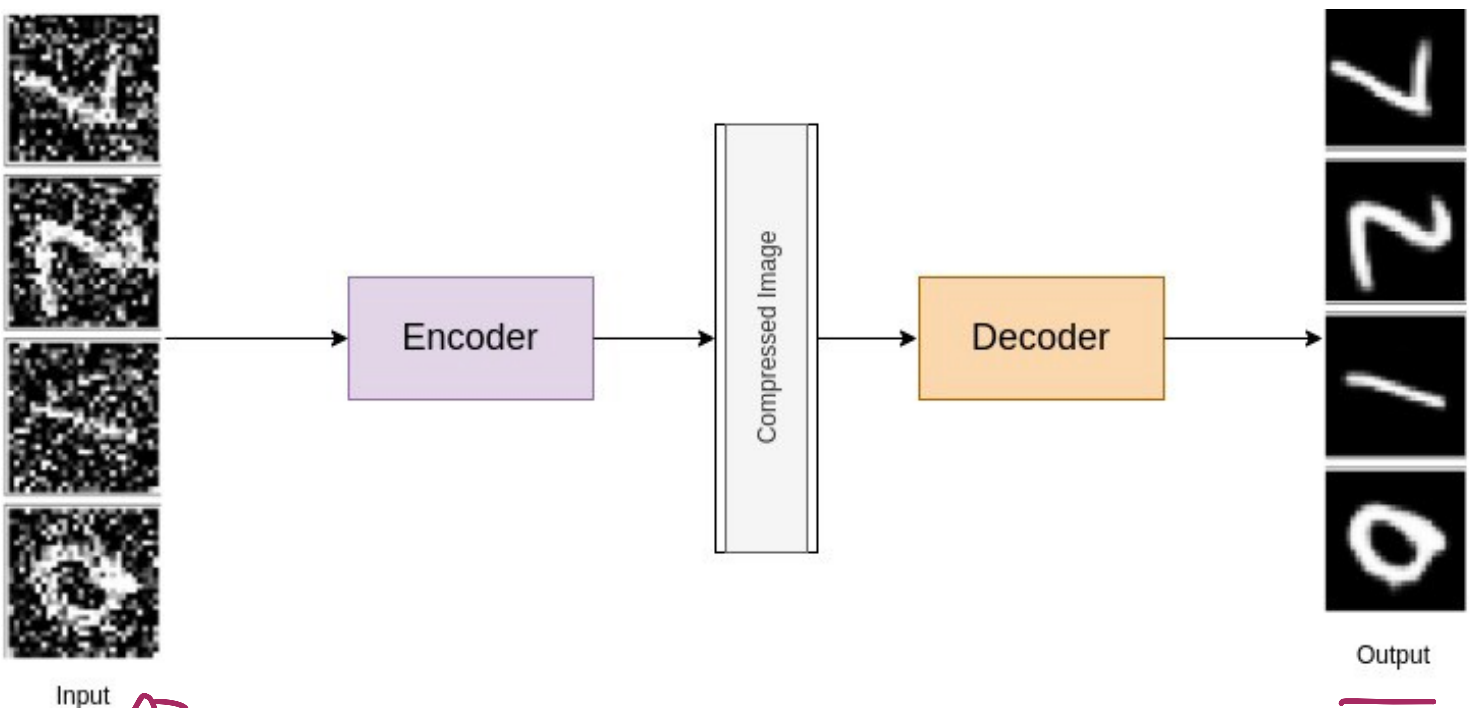


De-noising Auto Encoders

Ex1

Ex2

Ex3



Training

De-noising Auto Encoders

Details

- Just like an Auto Encoder

De-noising Auto Encoders

Details

- Just like an Auto Encoder
- Difference: Noise is injected in the inputs on purpose but output is a clean data point.

De-noising Auto Encoders

Details

- Just like an Auto Encoder
- Difference: Noise is injected in the inputs on purpose but output is a clean data point.
- This forces the Auto Encoder to "de-noise" data, esp. useful for images!

De-noising Auto Encoders

Details

- Just like an Auto Encoder
- Difference: Noise is injected in the inputs on purpose but output is a clean data point.
- This forces the Auto Encoder to "de-noise" data, esp. useful for images!
- Esp. useful for a category of objects or images (e.g. digit recognition or face recognition, etc)

ICE #3

Unsupervised Learning

Which of these is NOT an example of unsupervised learning?

1. Perceptron → supervised
2. Auto Encoder → (self-supervised) (unsupervised)
3. De-noising Auto Encoder — ||
4. K-means++ unsupervised
5. None of the above
6. All of the above

Breakouts Time #1

5 mins

Discuss in your groups what are some real-world applications of any or many of the Auto Encoder Architectures we discussed so far you can think of in your area of work or in a standard context e.g. images.

Stable Diffusion High-Level (only text)

Magic

Text

Image

Reference: The Illustrated Stable Diffusion

Stable Diffusion High-Level (text and image input)

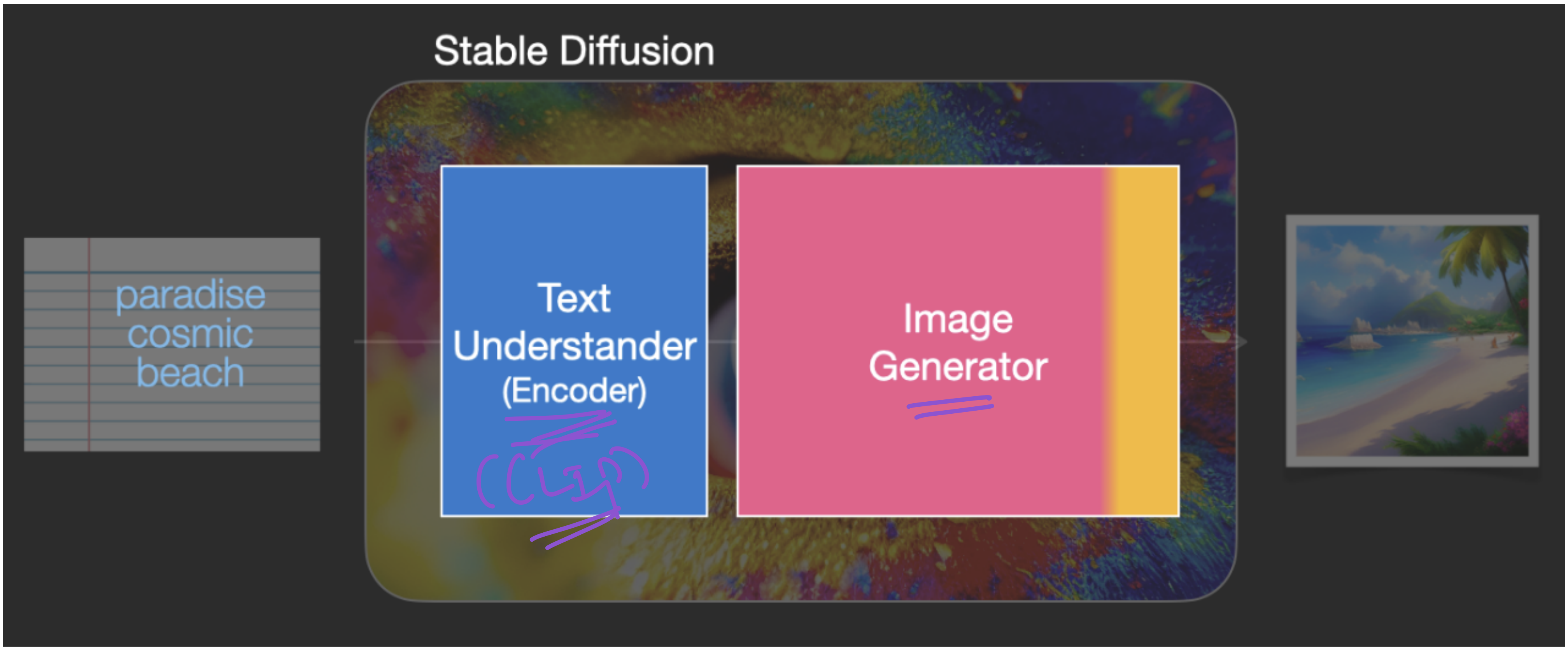
Text

Pirate ship

Ref Image

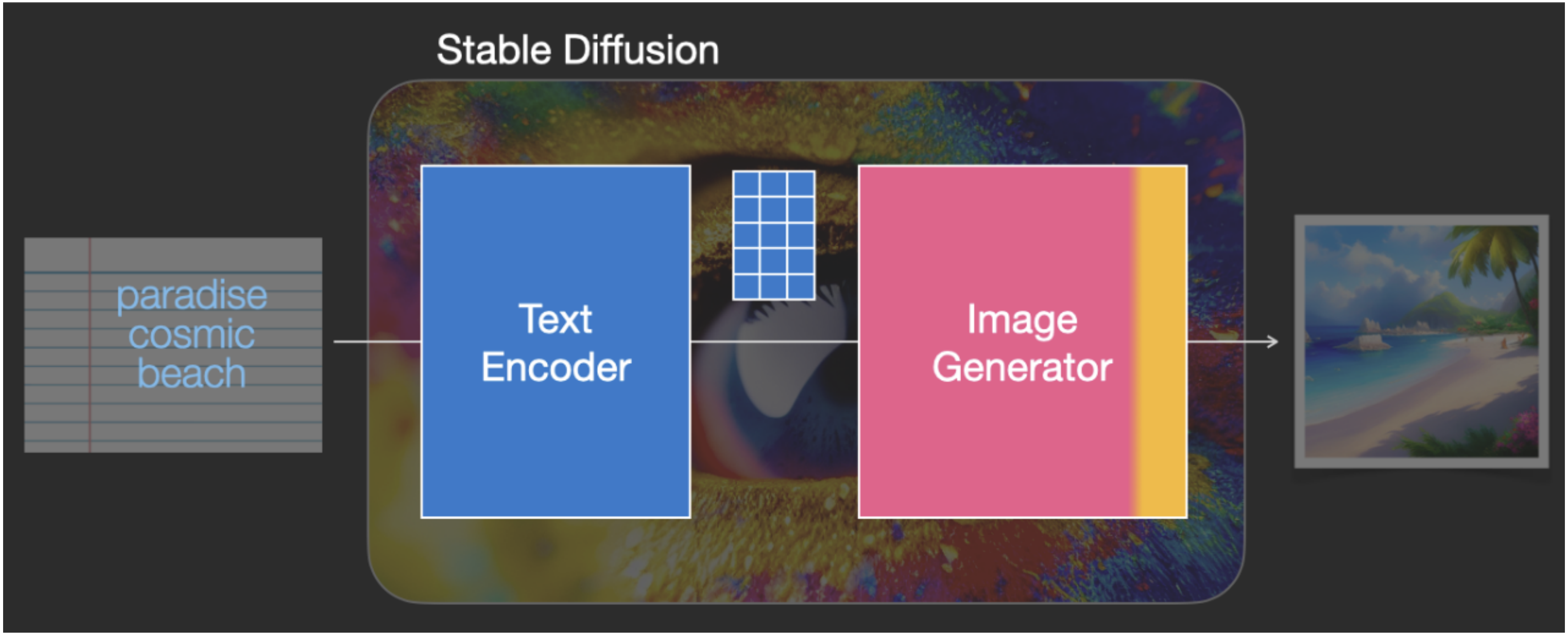
Reference: The Illustrated Stable Diffusion

Stable Diffusion Components



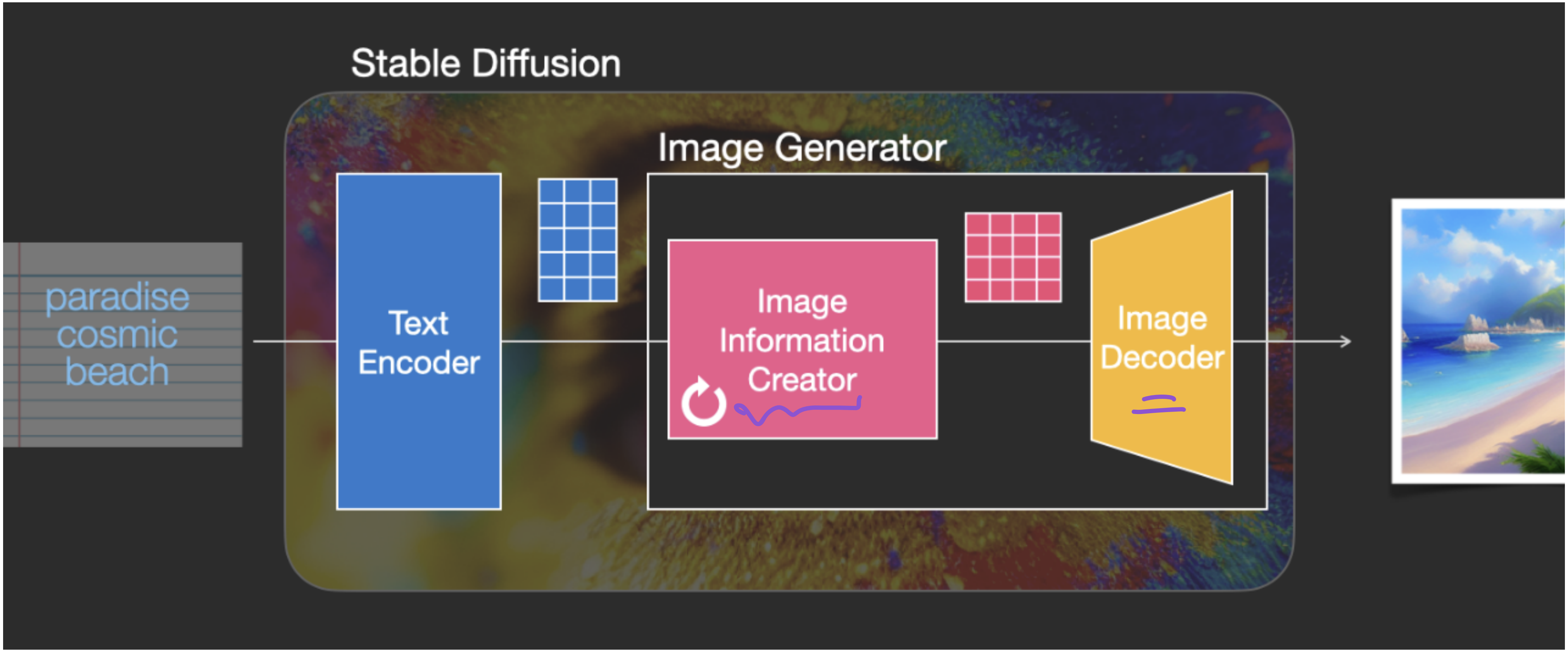
Reference: The Illustrated Stable Diffusion

Stable Diffusion Components



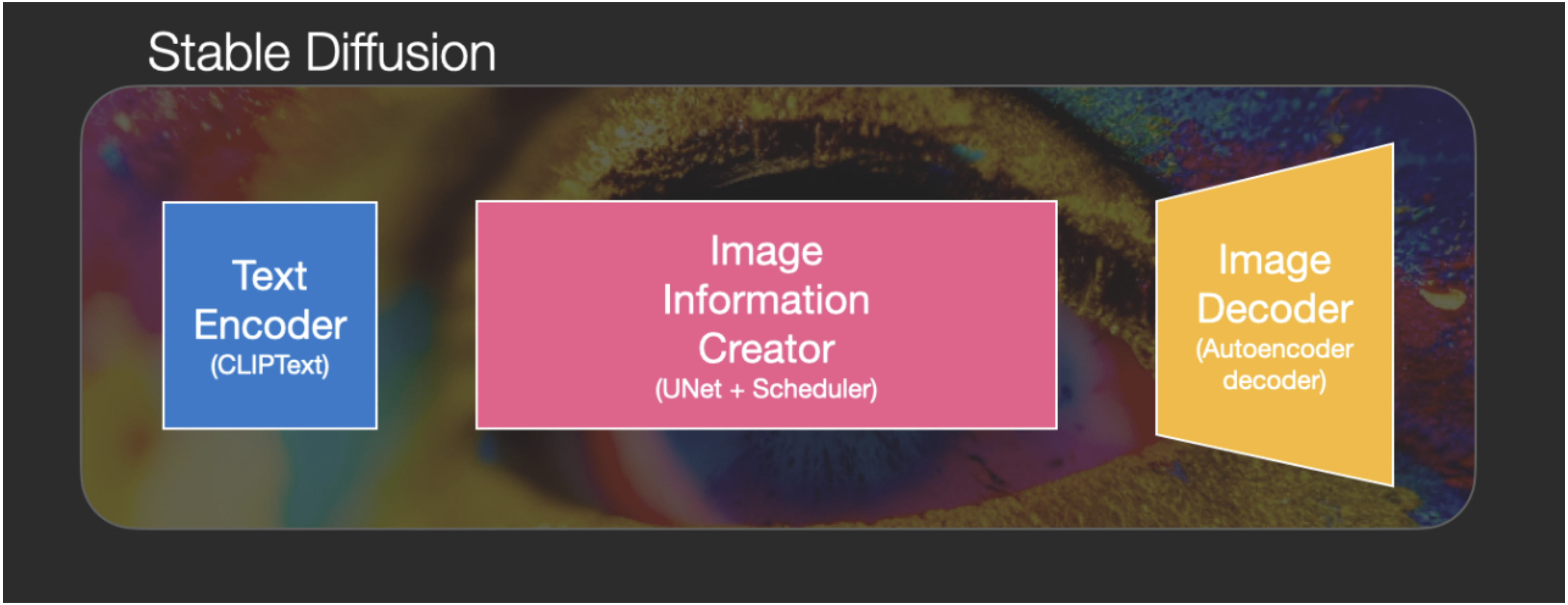
Reference: The Illustrated Stable Diffusion

Image Information Creator



Reference: The Illustrated Stable Diffusion

Image Decoder



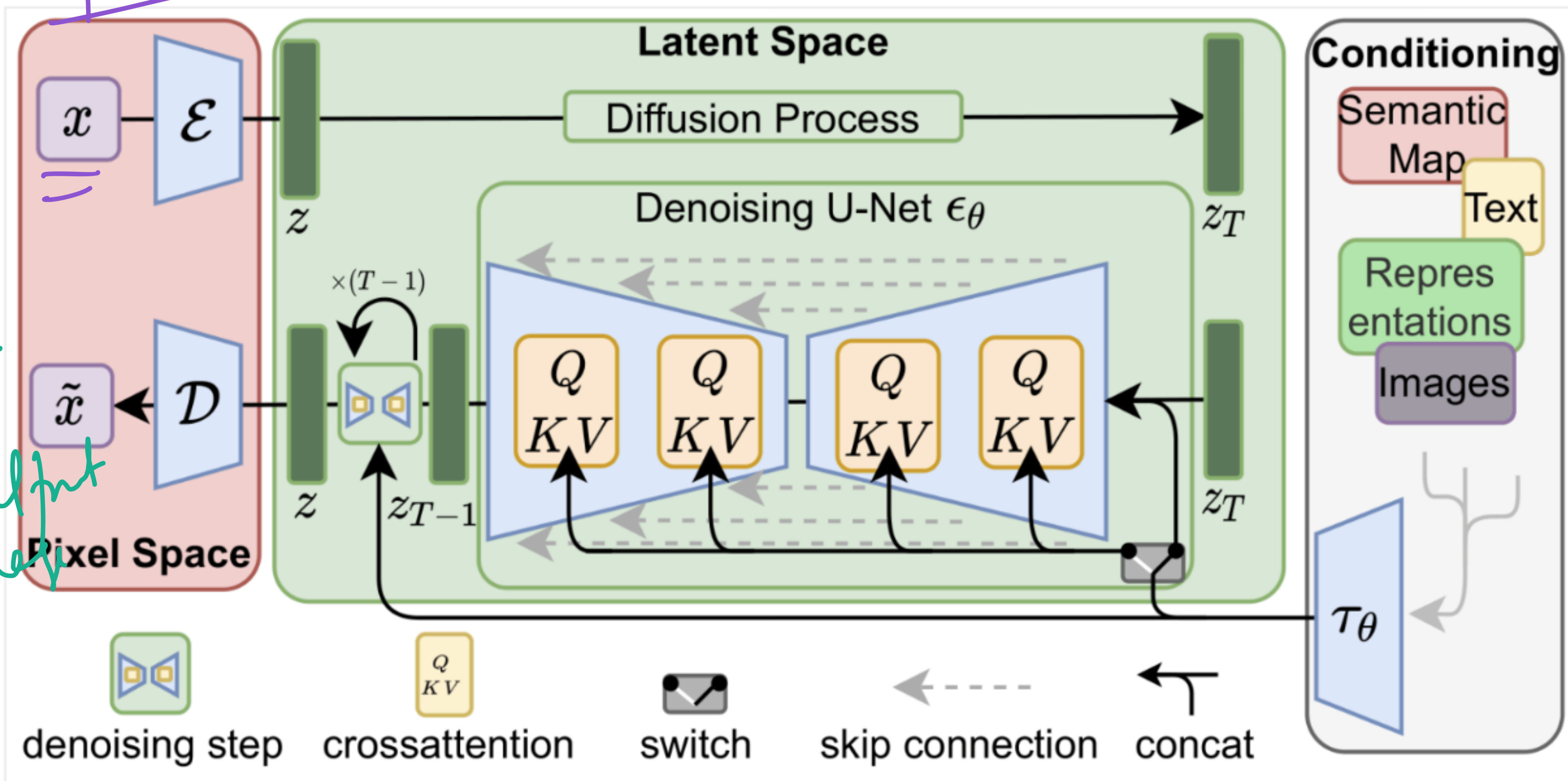
Reference: The Illustrated Stable Diffusion

Stable Diffusion - Break down of components

1. **Text Encoding:** Uses ClipText
2. **Image diffusion process:** Take in an image and adds noise to the image.
3. **Reverse diffusion process:** Take in a text embedding and successfully generate an image embedding that can then be converted to an image. Each step here de-noises the image embedding.
4. **Image Decoder:** This is an AutoEncoder-Decoder that takes in an Image embedding and returns an image in a pixel format (512x512x3)

Stable Diffusion Full Architecture

Input Image

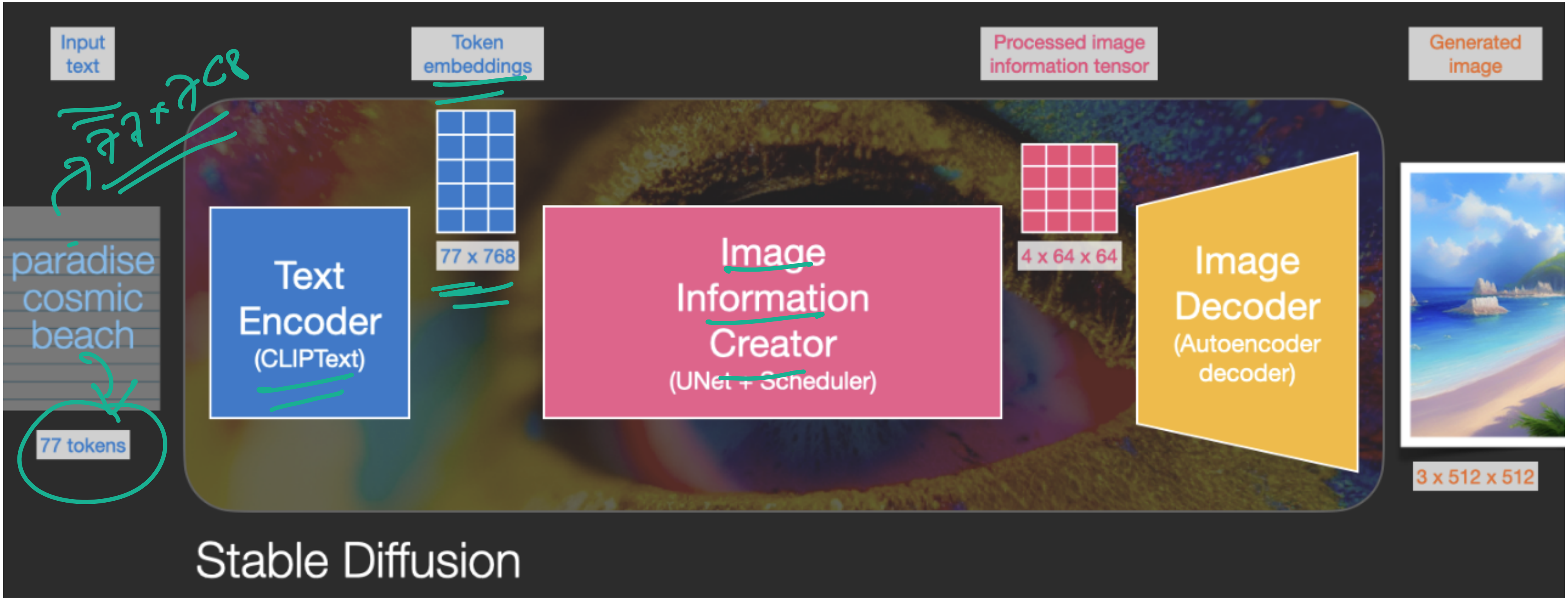


x (Pixel) $\tilde{x}$

Output Image

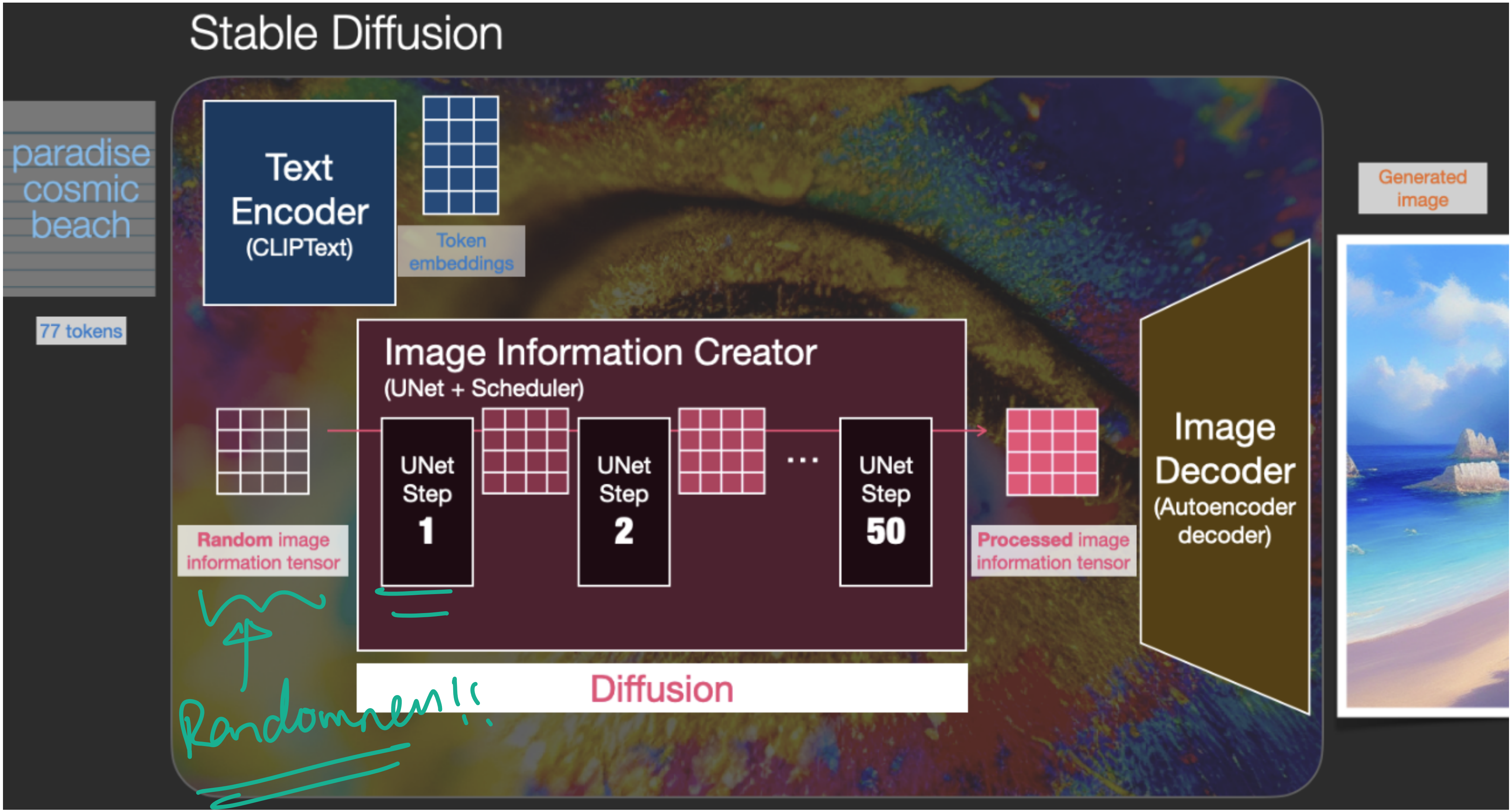
Reference: The Illustrated Stable Diffusion

High-level of Image Generation



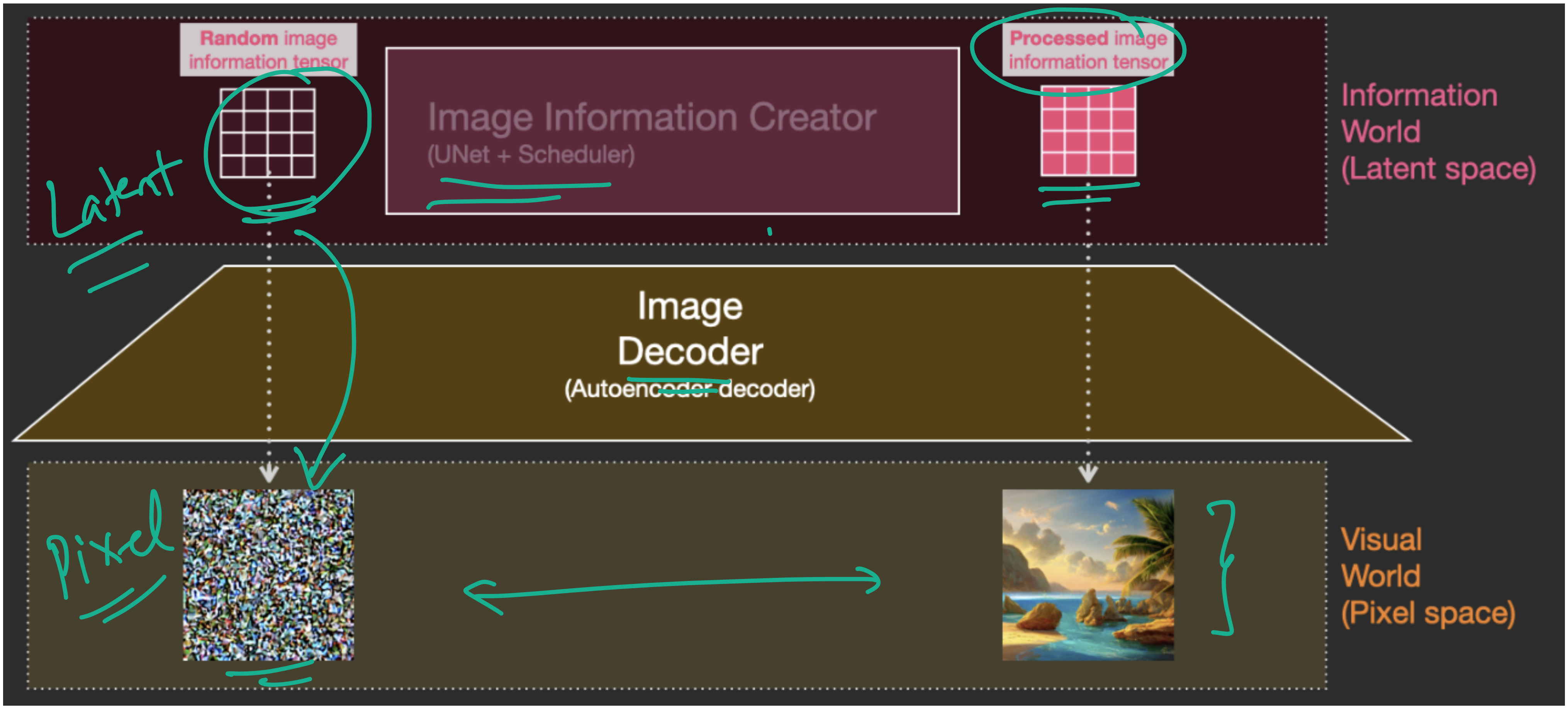
Reference: The Illustrated Stable Diffusion

Understanding Diffusion



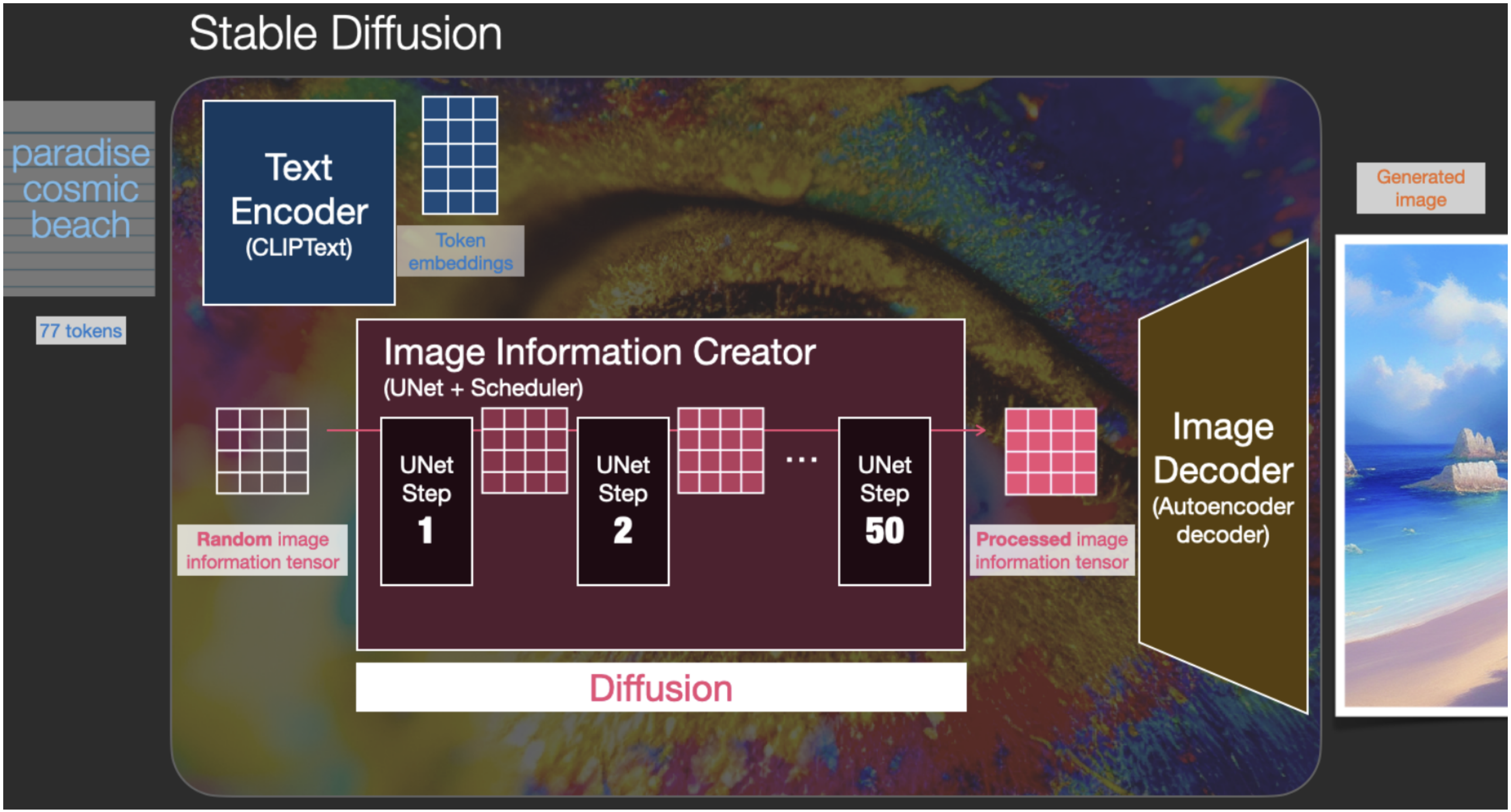
Reference: The Illustrated Stable Diffusion

Understanding Diffusion



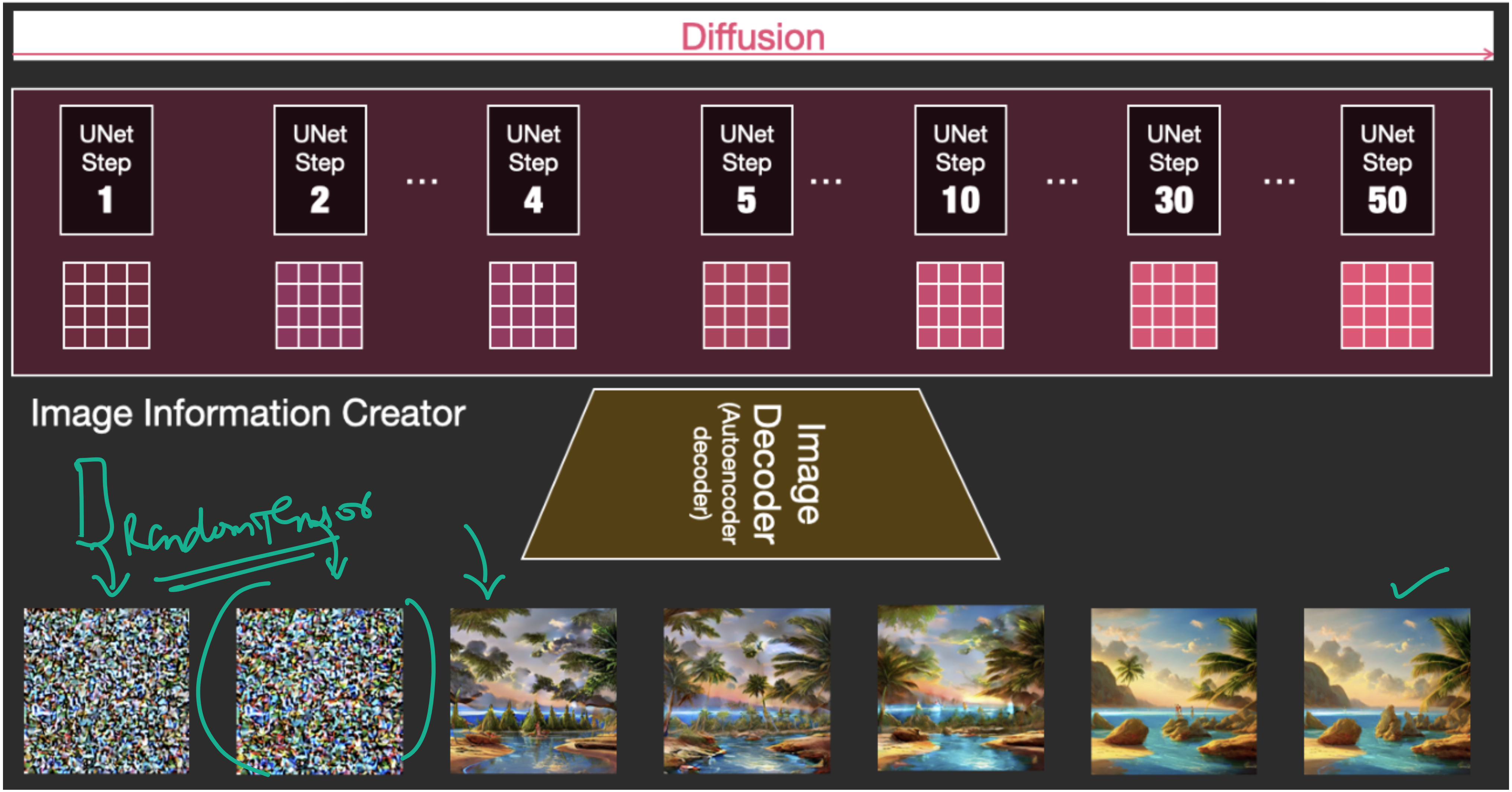
Reference: The Illustrated Stable Diffusion

Understanding Diffusion



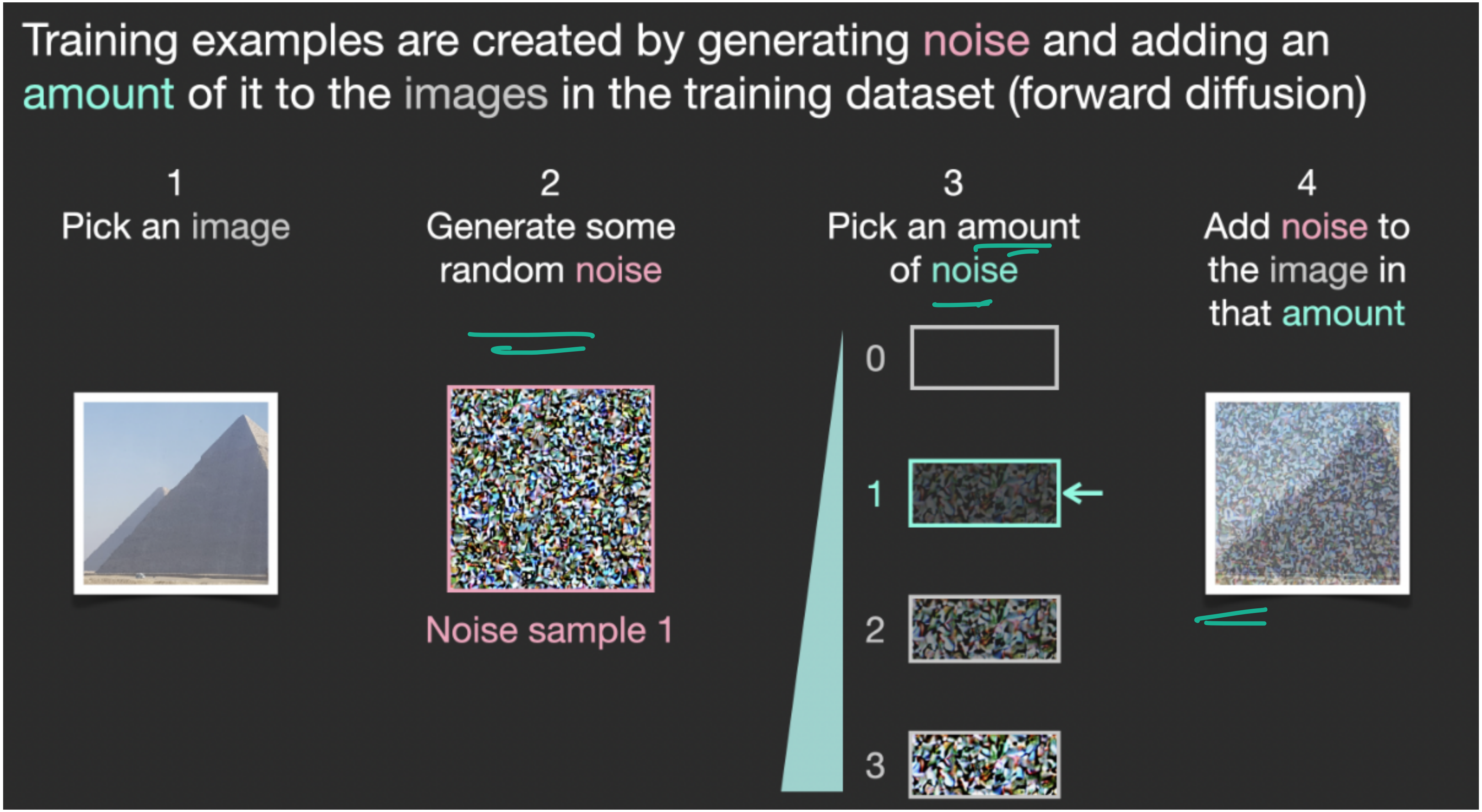
Reference: The Illustrated Stable Diffusion

Understanding Diffusion



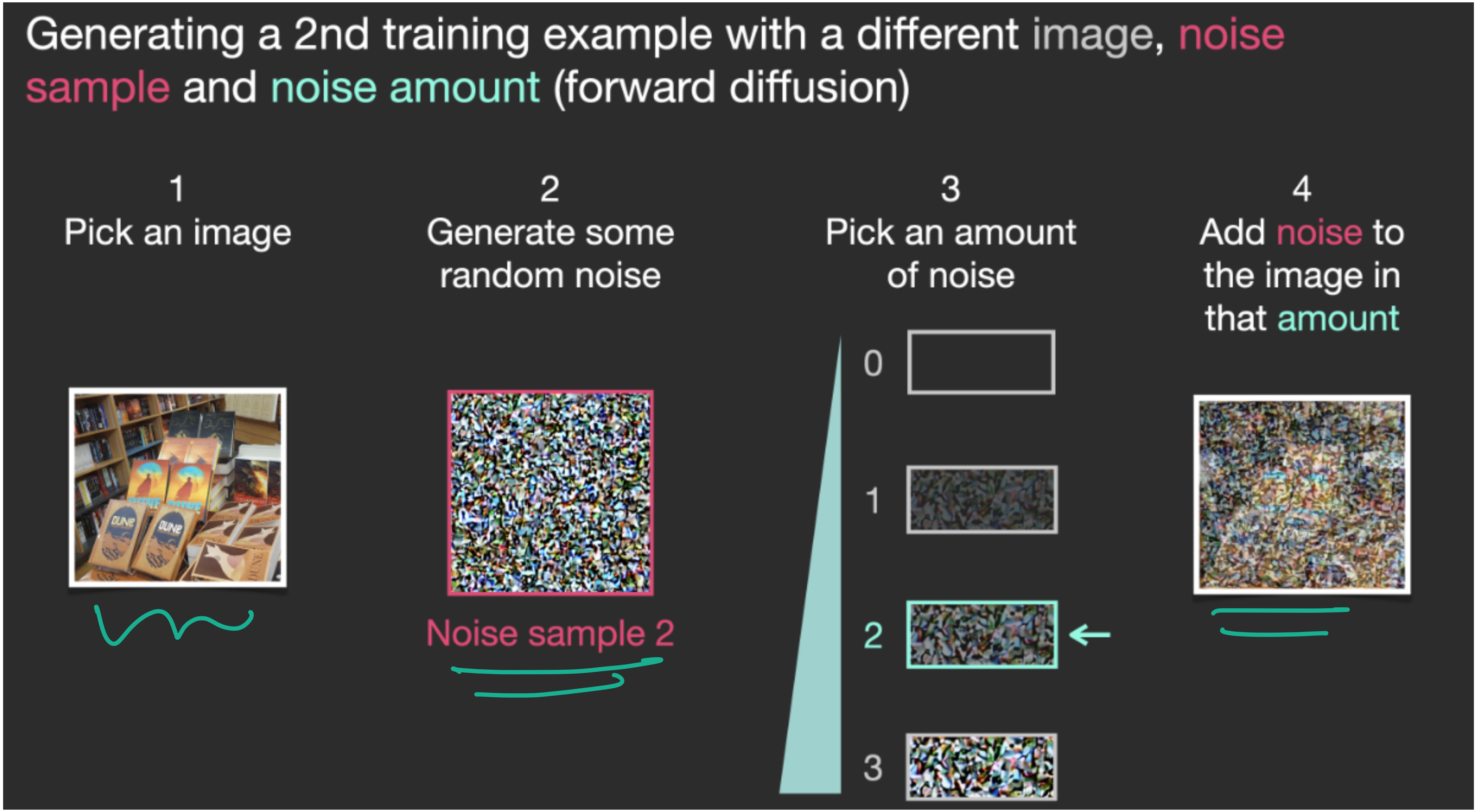
Reference: The Illustrated Stable Diffusion

Generating Training Examples with Noise Addition



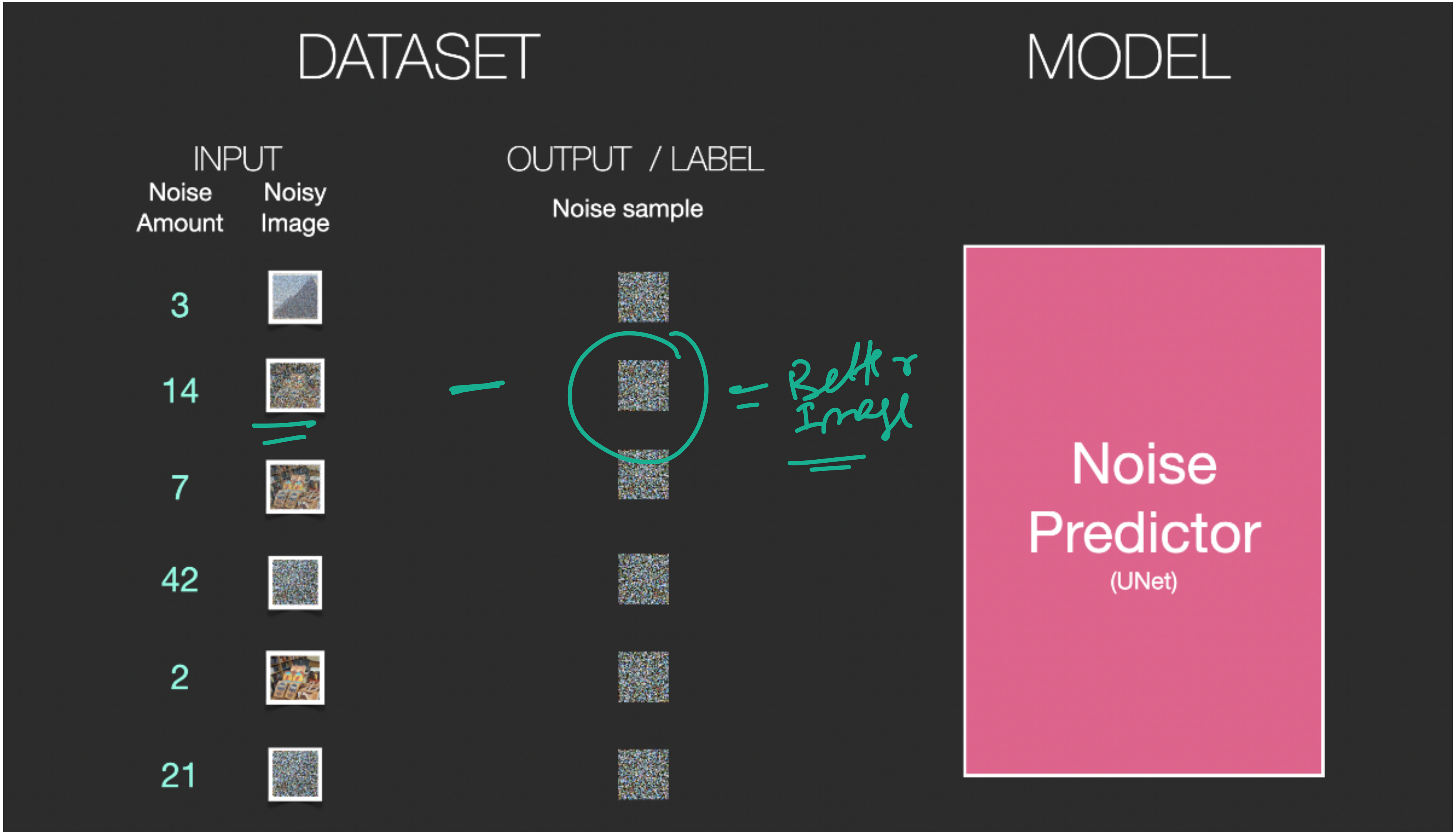
Reference: The Illustrated Stable Diffusion

Generating Training Examples with Noise Addition



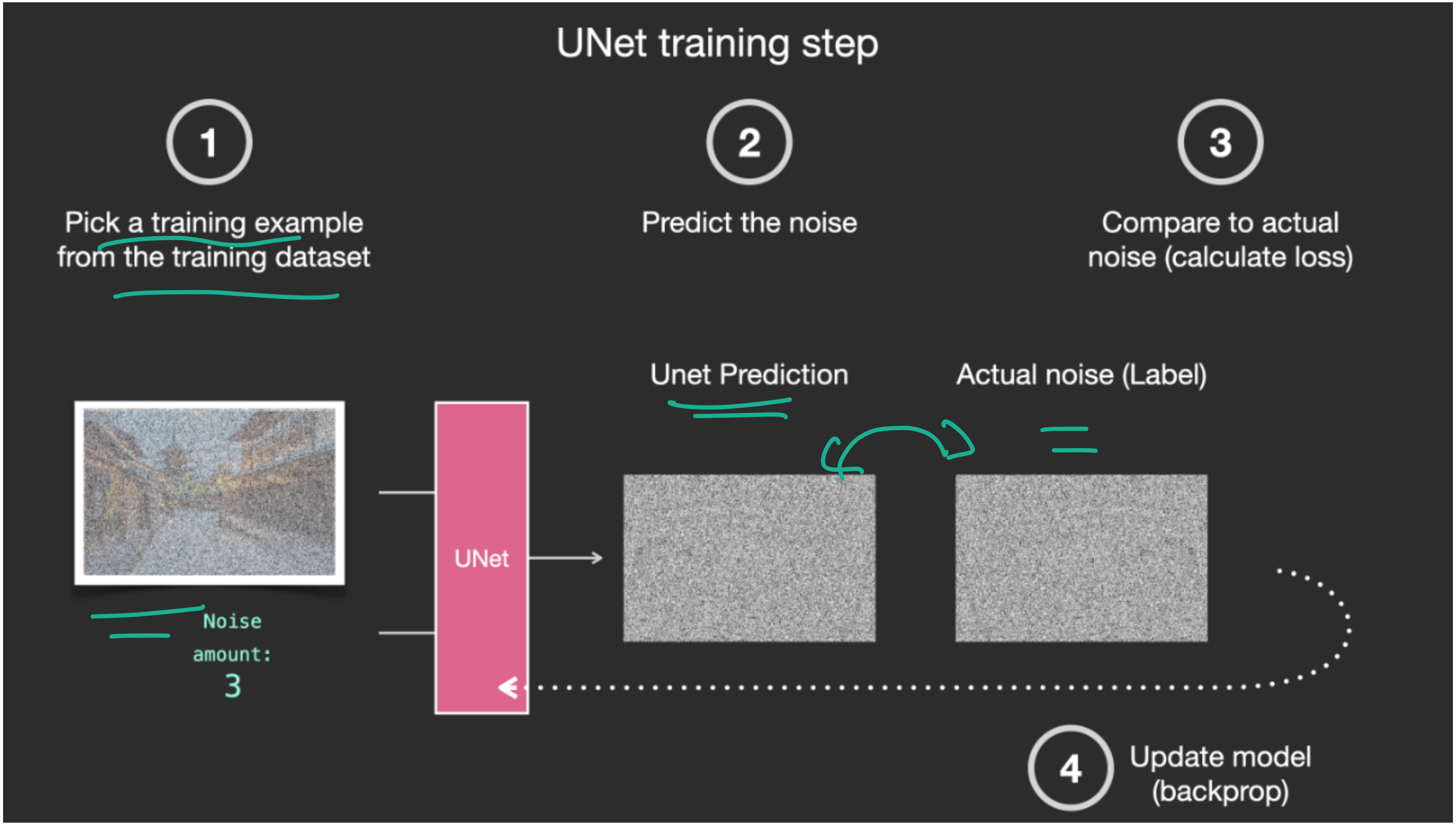
Reference: The Illustrated Stable Diffusion

Generating Training Examples with Noise Addition



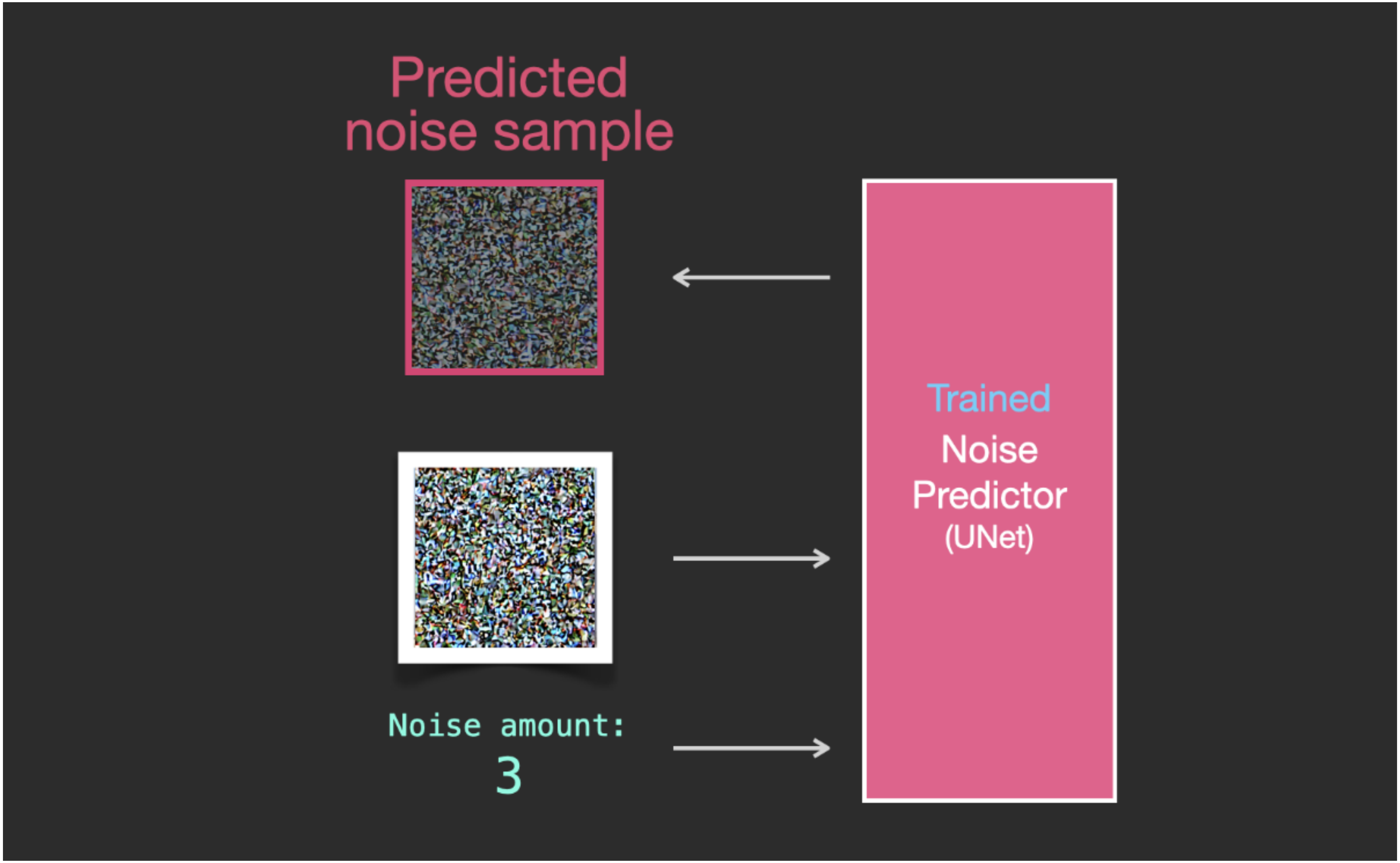
Reference: The Illustrated Stable Diffusion

Training Process



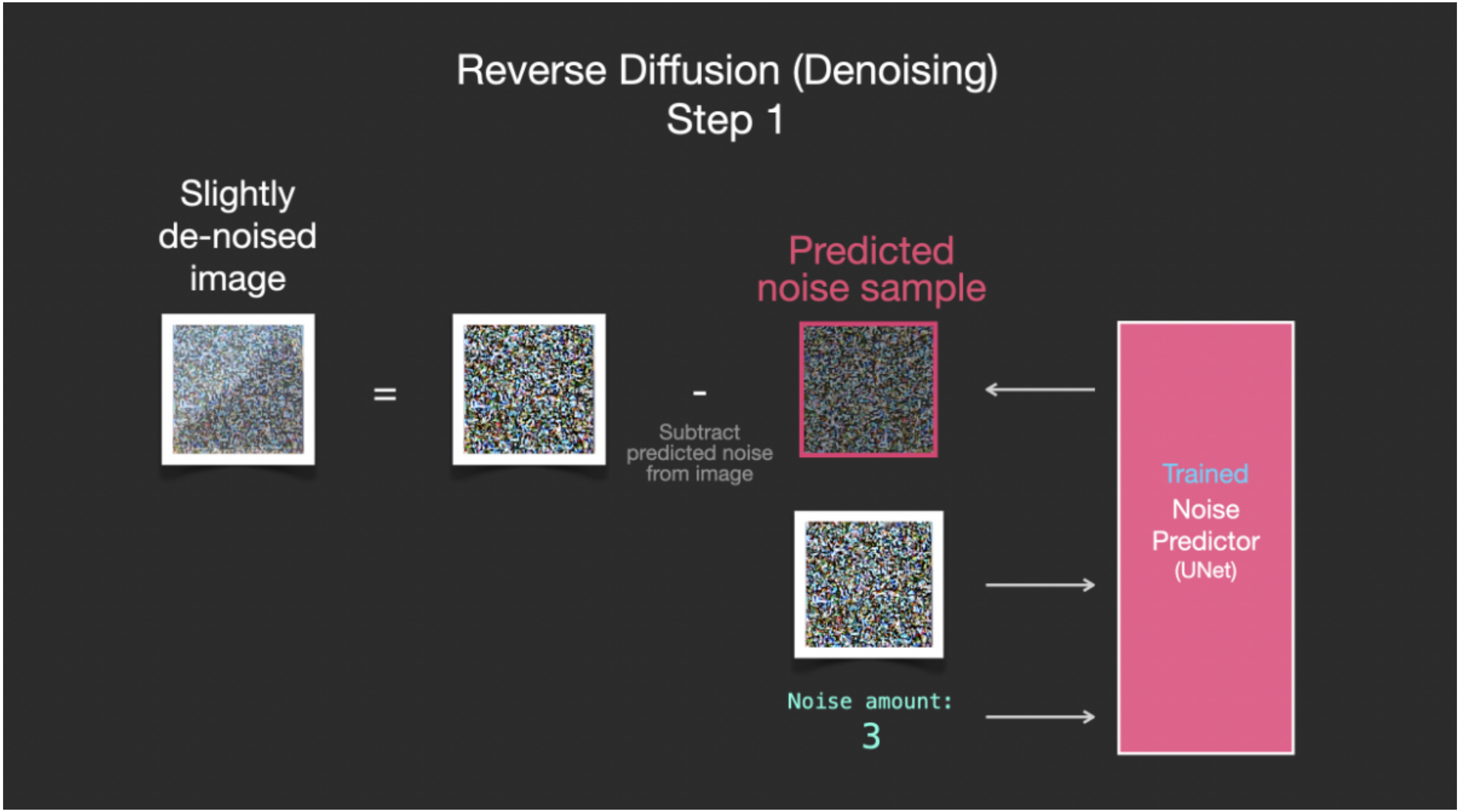
Reference: The Illustrated Stable Diffusion

Predicting noise and Noise Removal



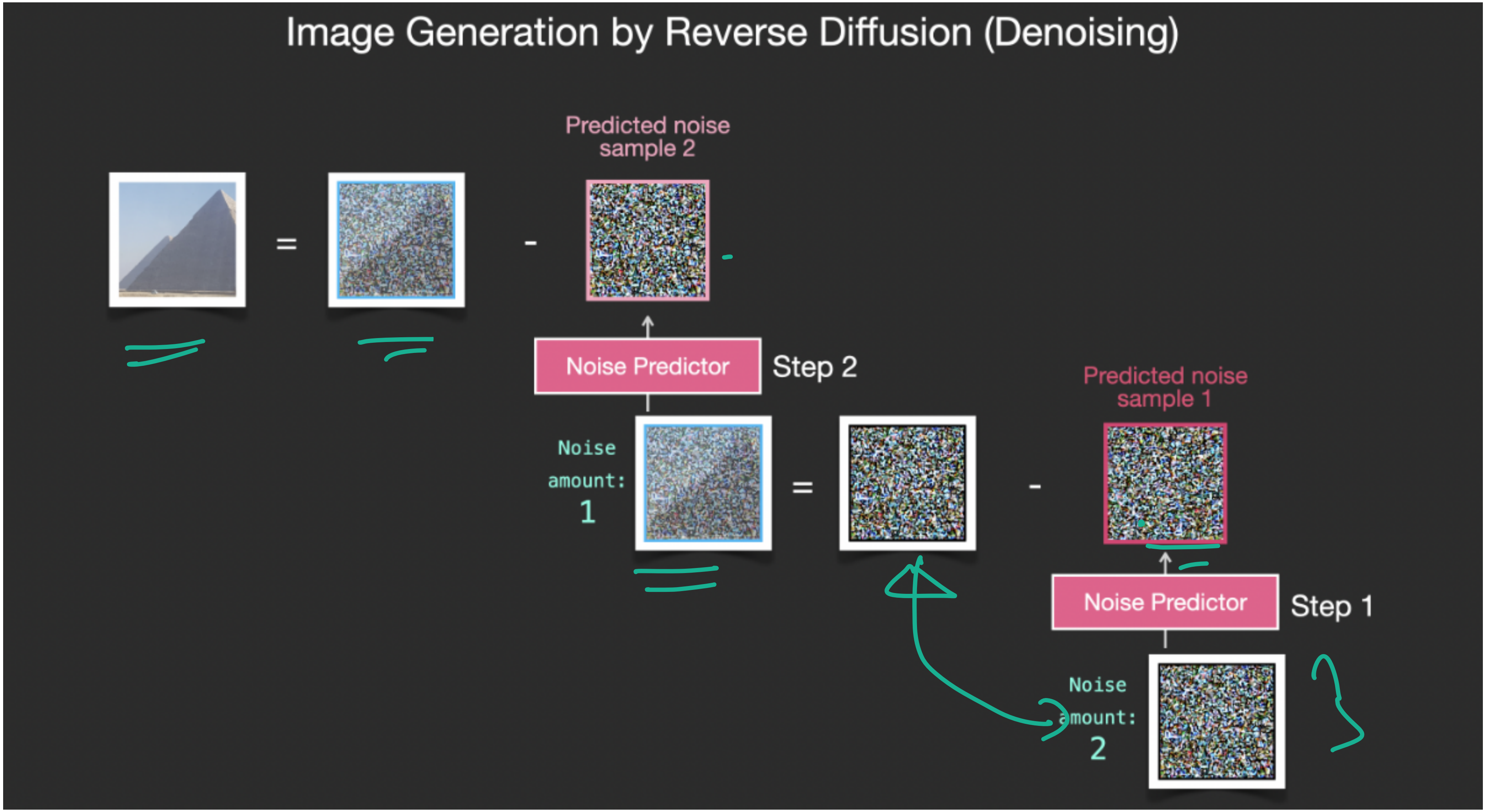
Reference: The Illustrated Stable Diffusion

Predicting noise and Noise Removal



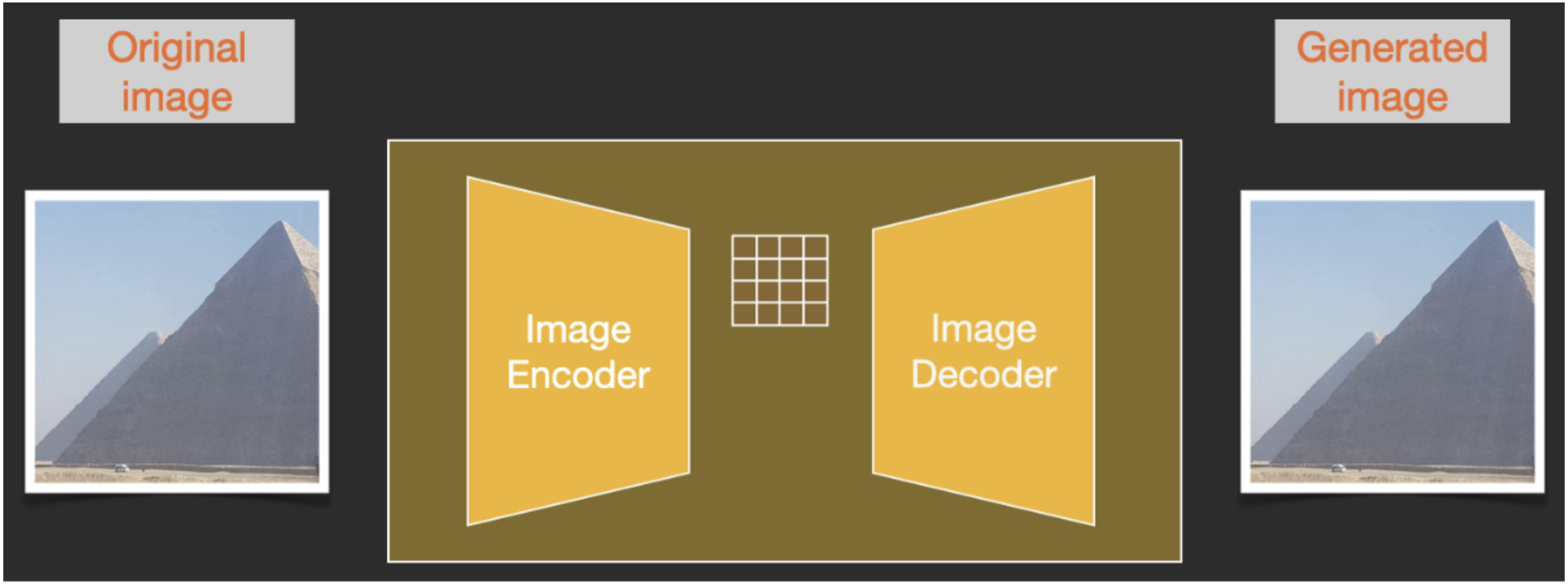
Reference: The Illustrated Stable Diffusion

Predicting noise and Noise Removal



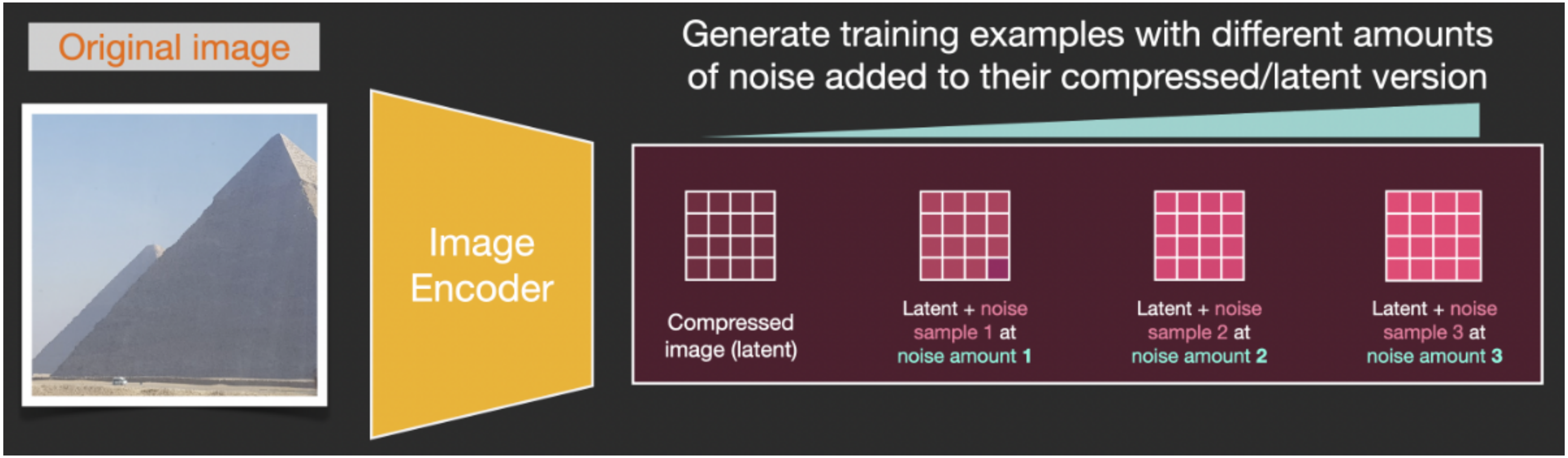
Reference: The Illustrated Stable Diffusion

Speeding up Stable Diffusion



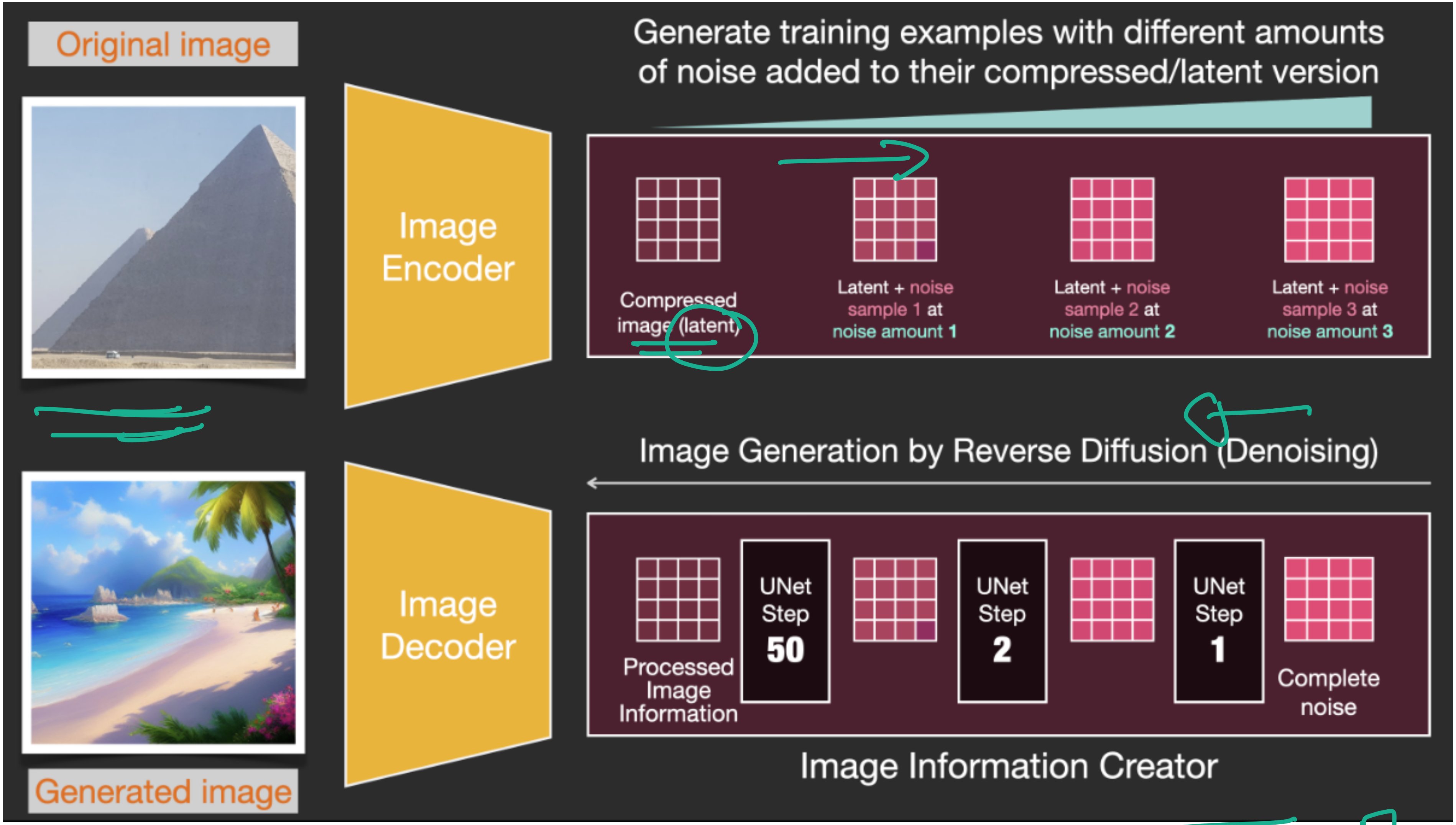
Reference: The Illustrated Stable Diffusion

Speeding up Stable Diffusion



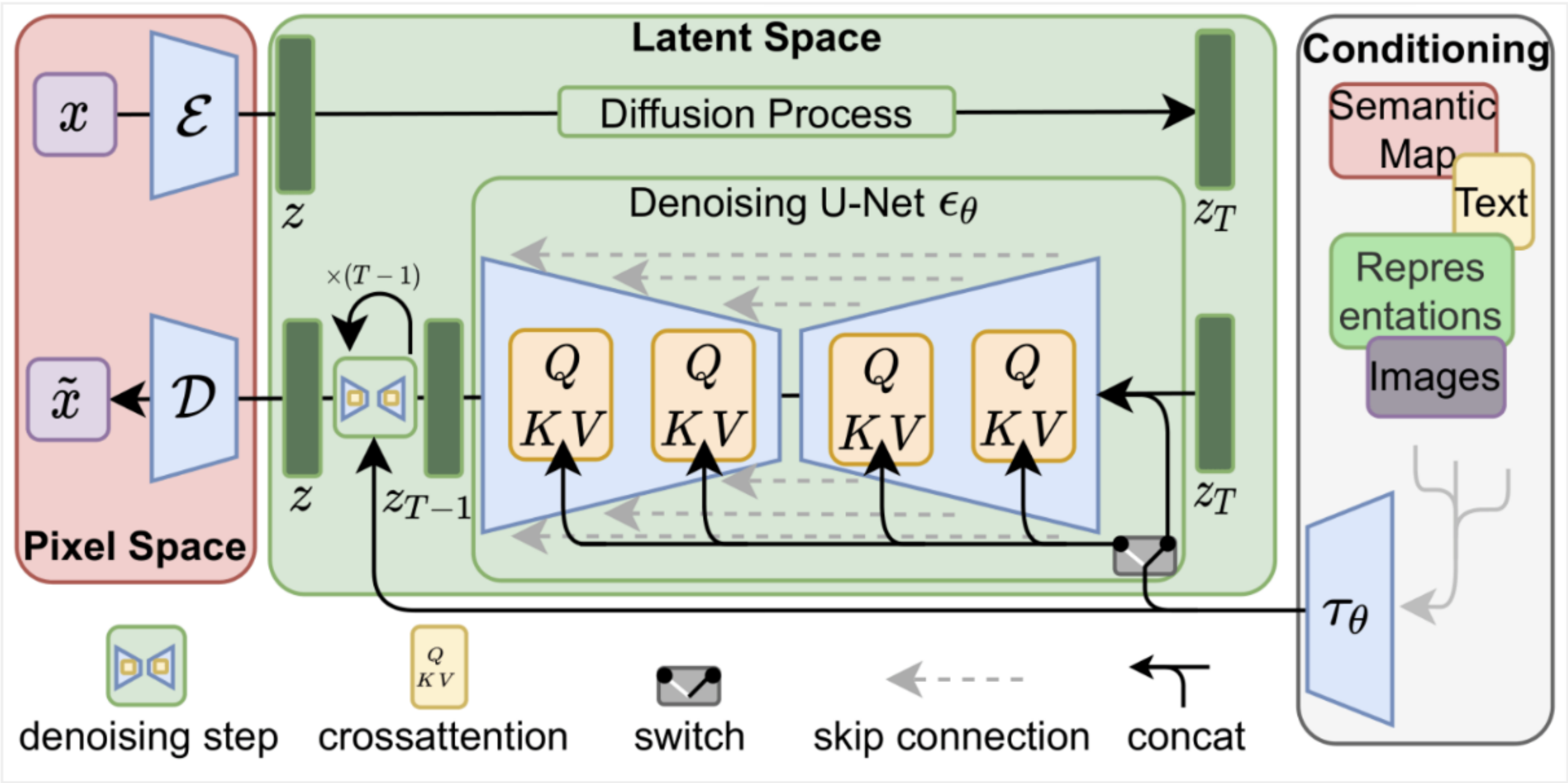
Reference: The Illustrated Stable Diffusion

Speeding up Stable Diffusion



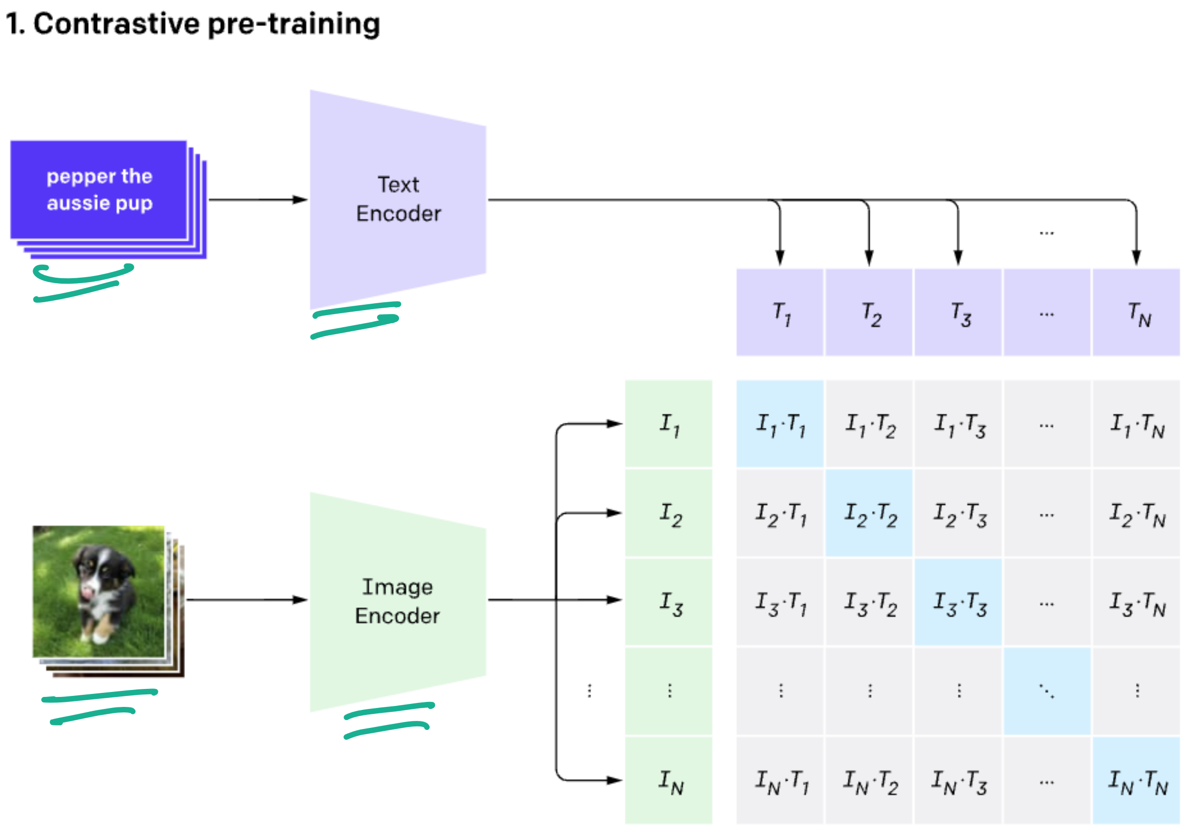
Reference: The Illustrated Stable Diffusion

Stable Diffusion Full Architecture



Reference: The Illustrated Stable Diffusion

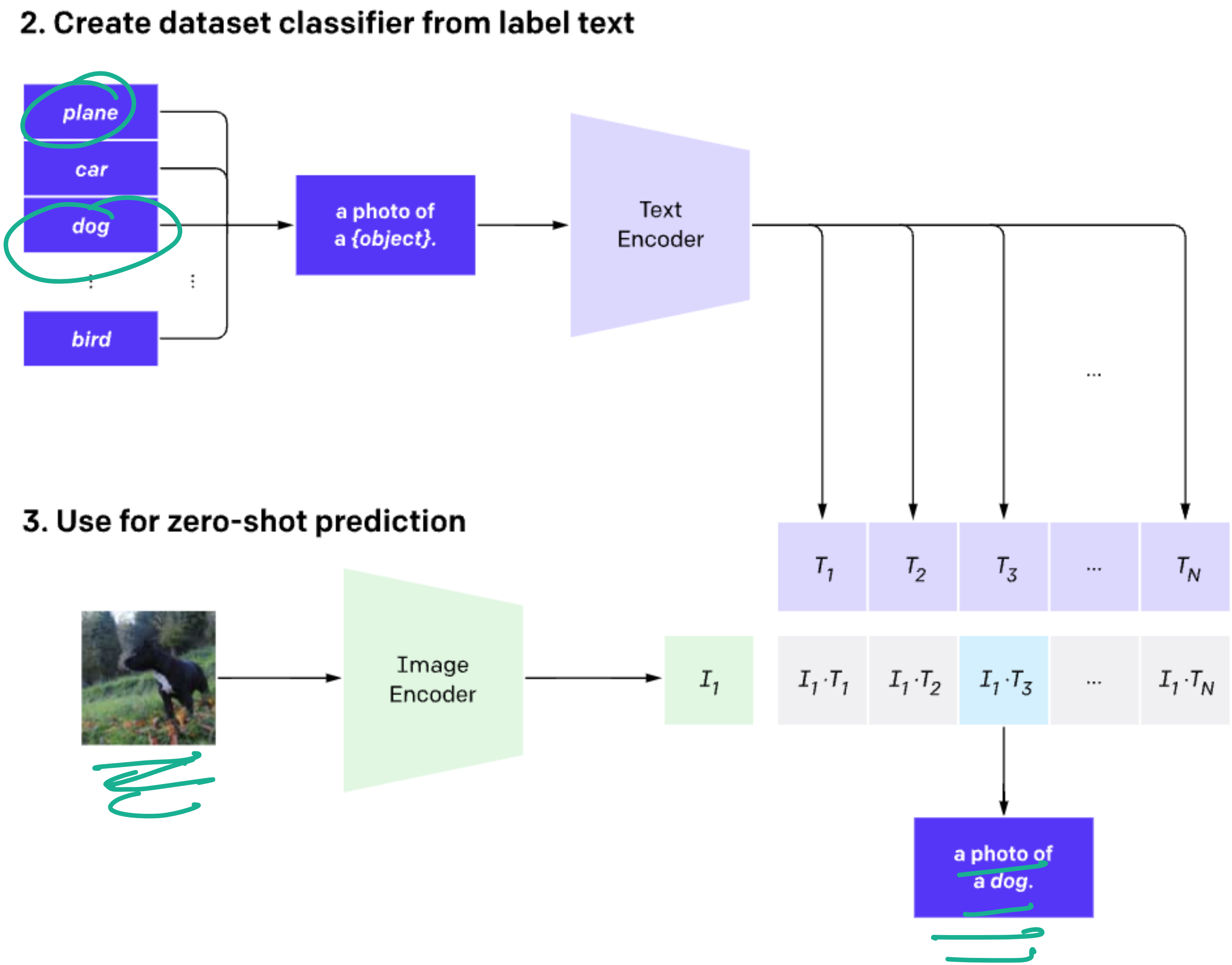
Clip Pre-Training Architecture



CLIP pre-trains an image encoder and a text encoder to predict which images were paired with which texts in our dataset. We then use this behavior to turn CLIP into a zero-shot classifier. We convert all of a dataset's classes into captions such as "a photo of a dog" and predict the class of the caption CLIP estimates best pairs with a given image.

Reference: CLIP

Clip Zero-Shot Prediction Process



Reference: CLIP

ICE #1

What pre-trained encoders would CLIP probably have used for text and image encodings?

1. CNN for both
2. Word2Vec and CNN
3. Transformer and Vi Transformer
4. Glove and CNN

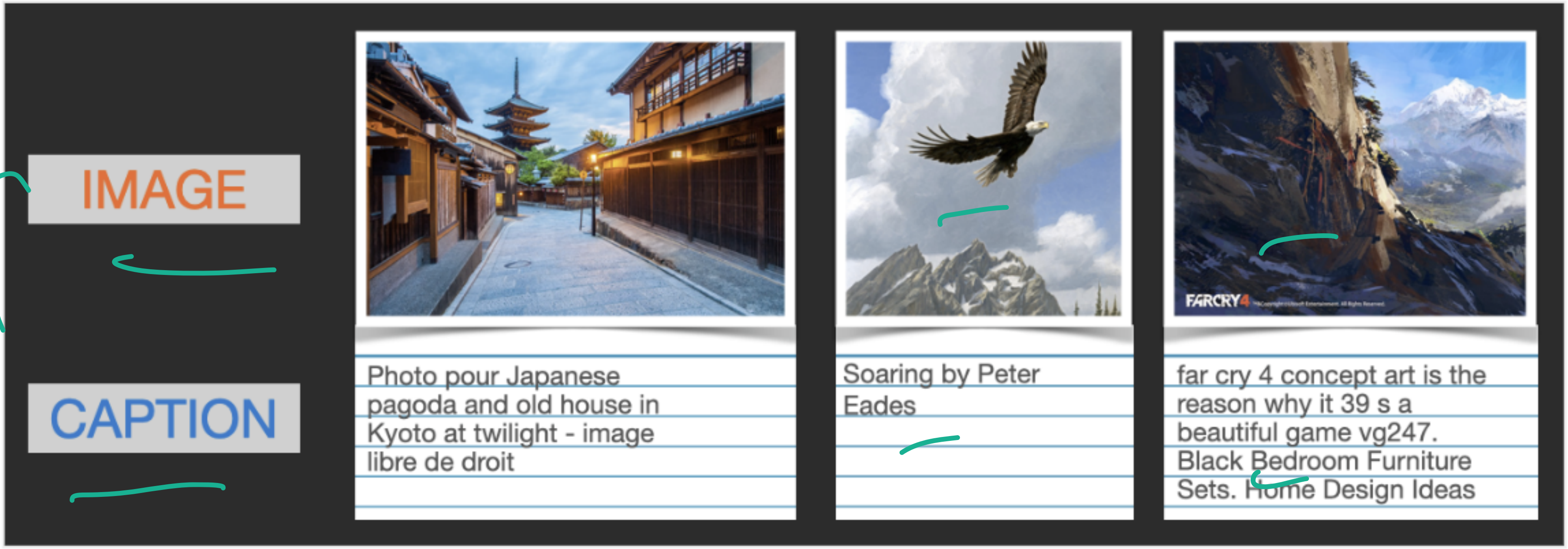
Clip Implementation Pseudo-code

```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

Figure 3. Numpy-like pseudocode for the core of an implementation of CLIP.

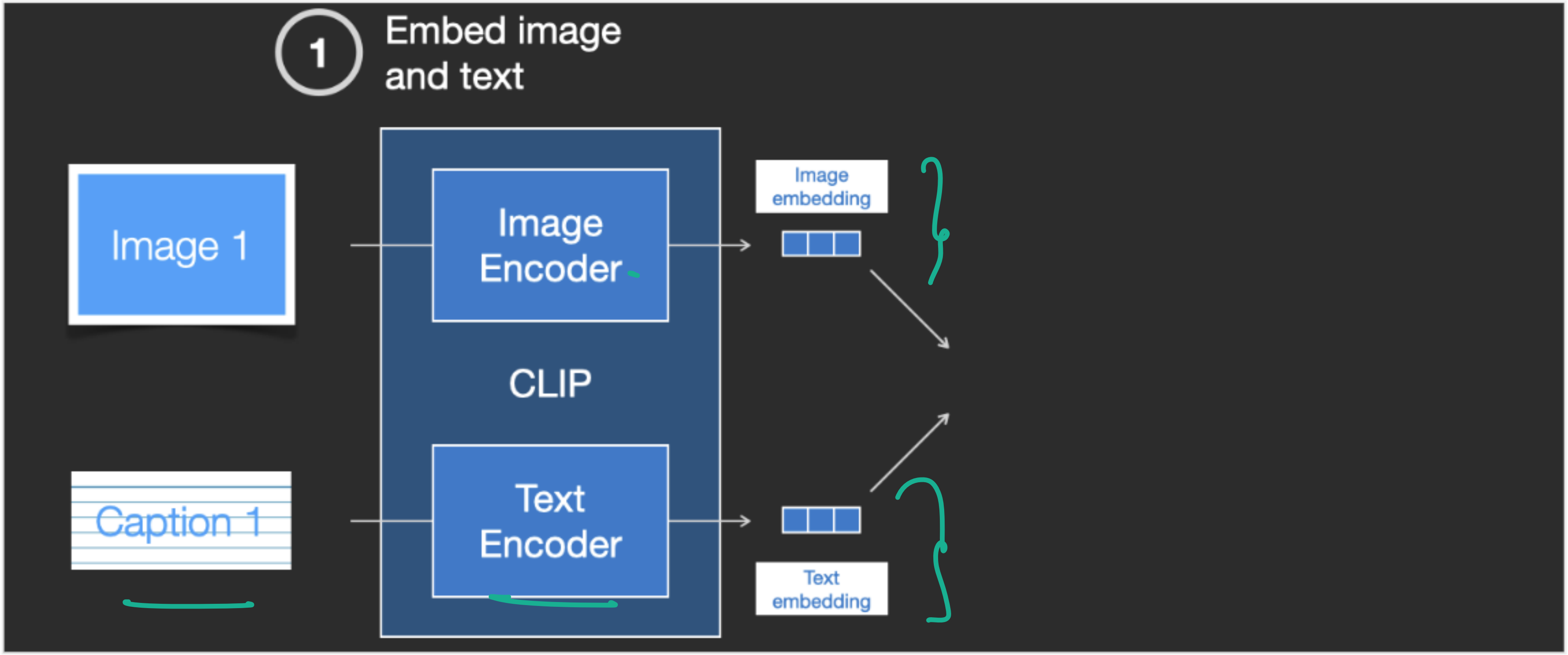
Reference: CLIP

Clip Training Examples



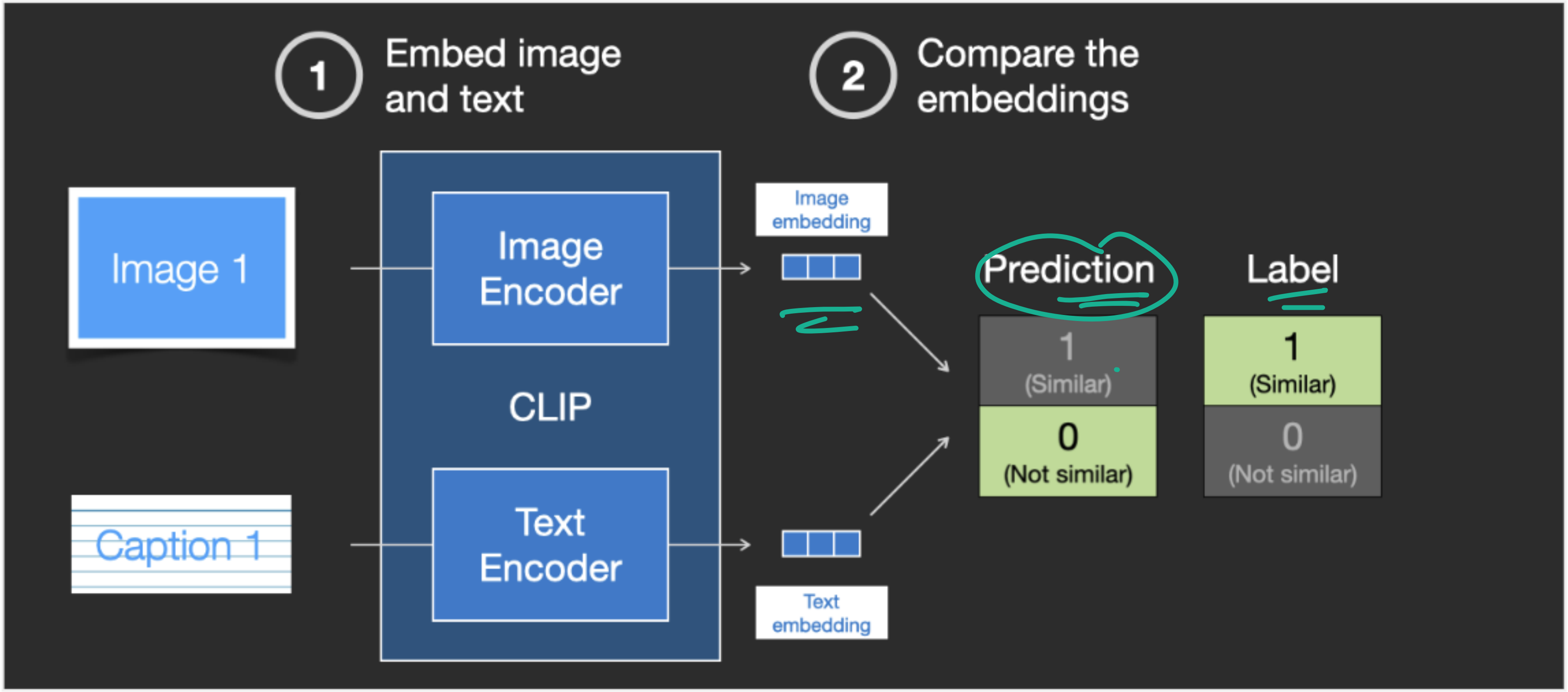
Question: How many examples used for Training? Reference: The Illustrated Stable Diffusion

Clip Training Process

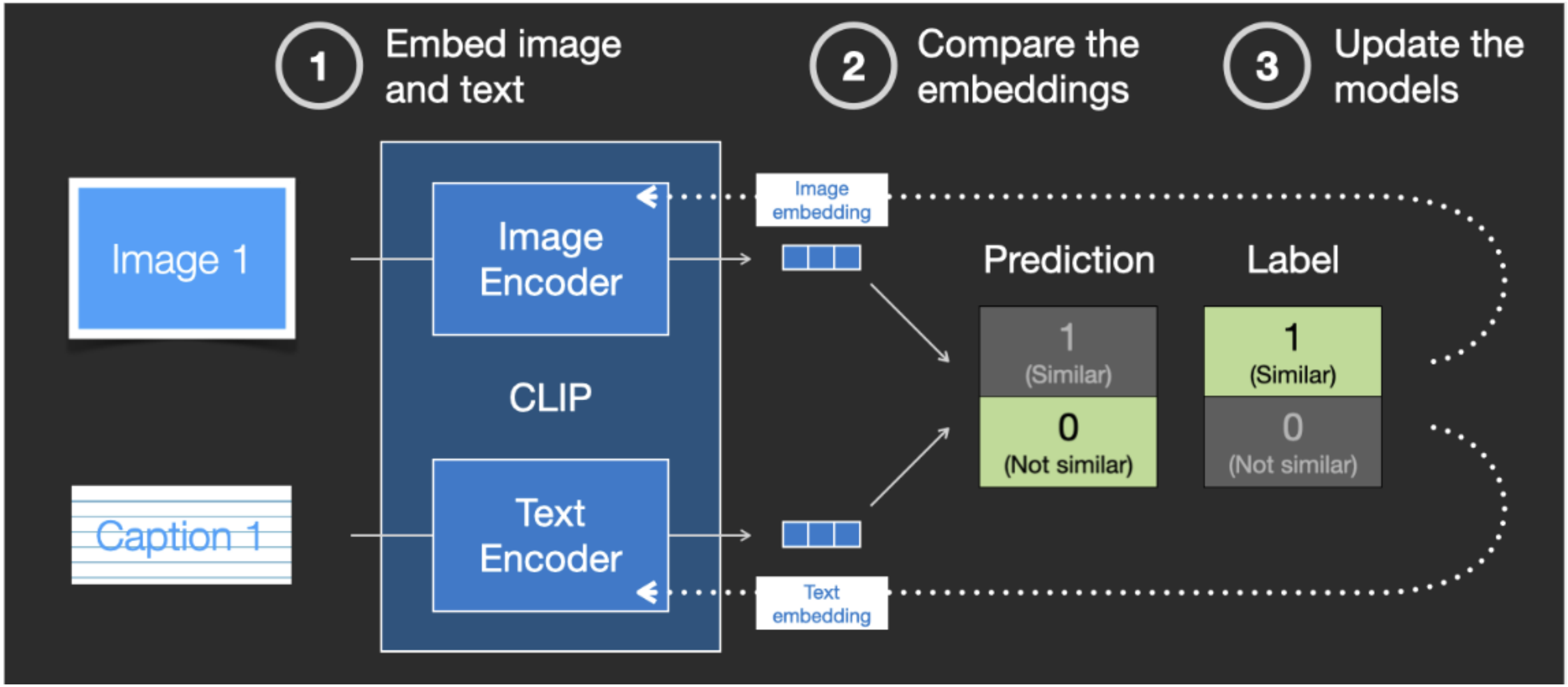


Reference: The Illustrated Stable Diffusion

Clip Training Process

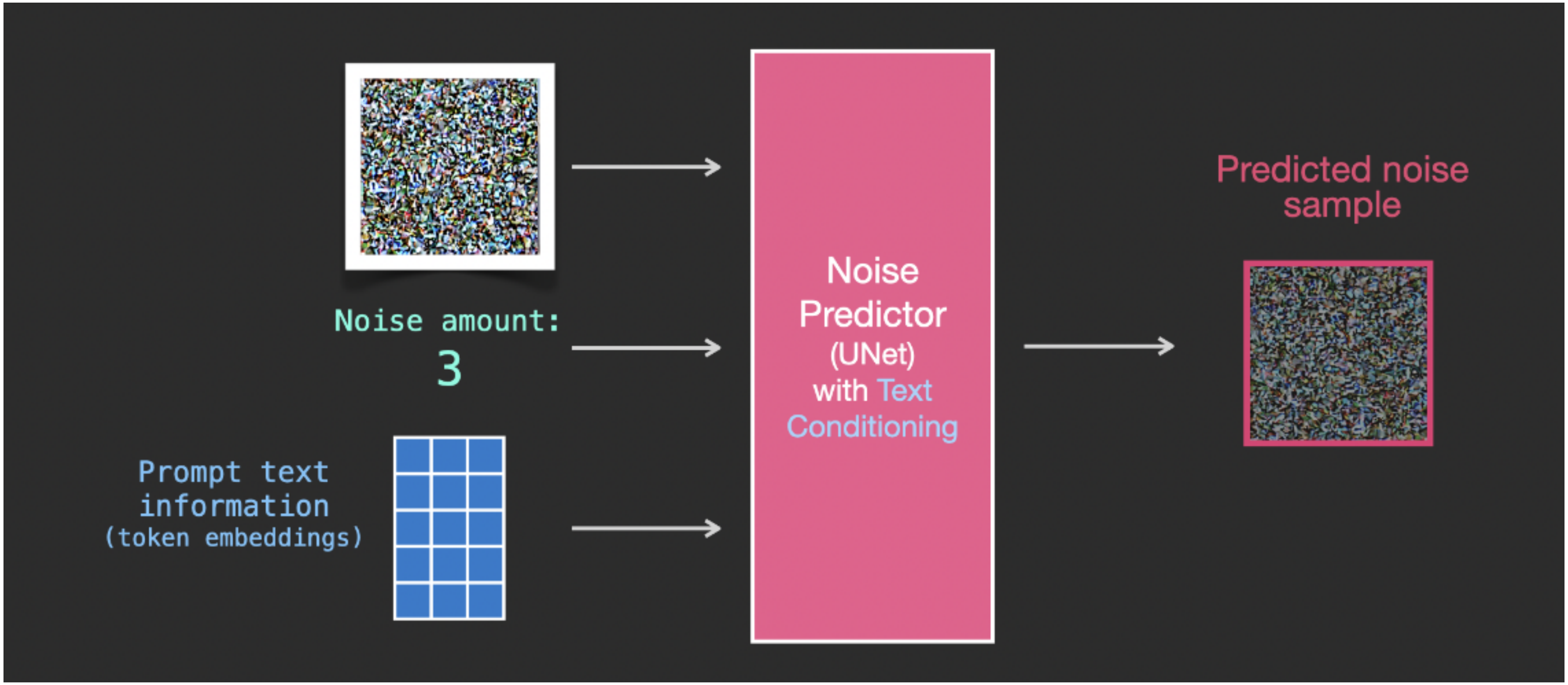


Reference: The Illustrated Stable Diffusion



Reference: The Illustrated Stable Diffusion

Image Generation Process

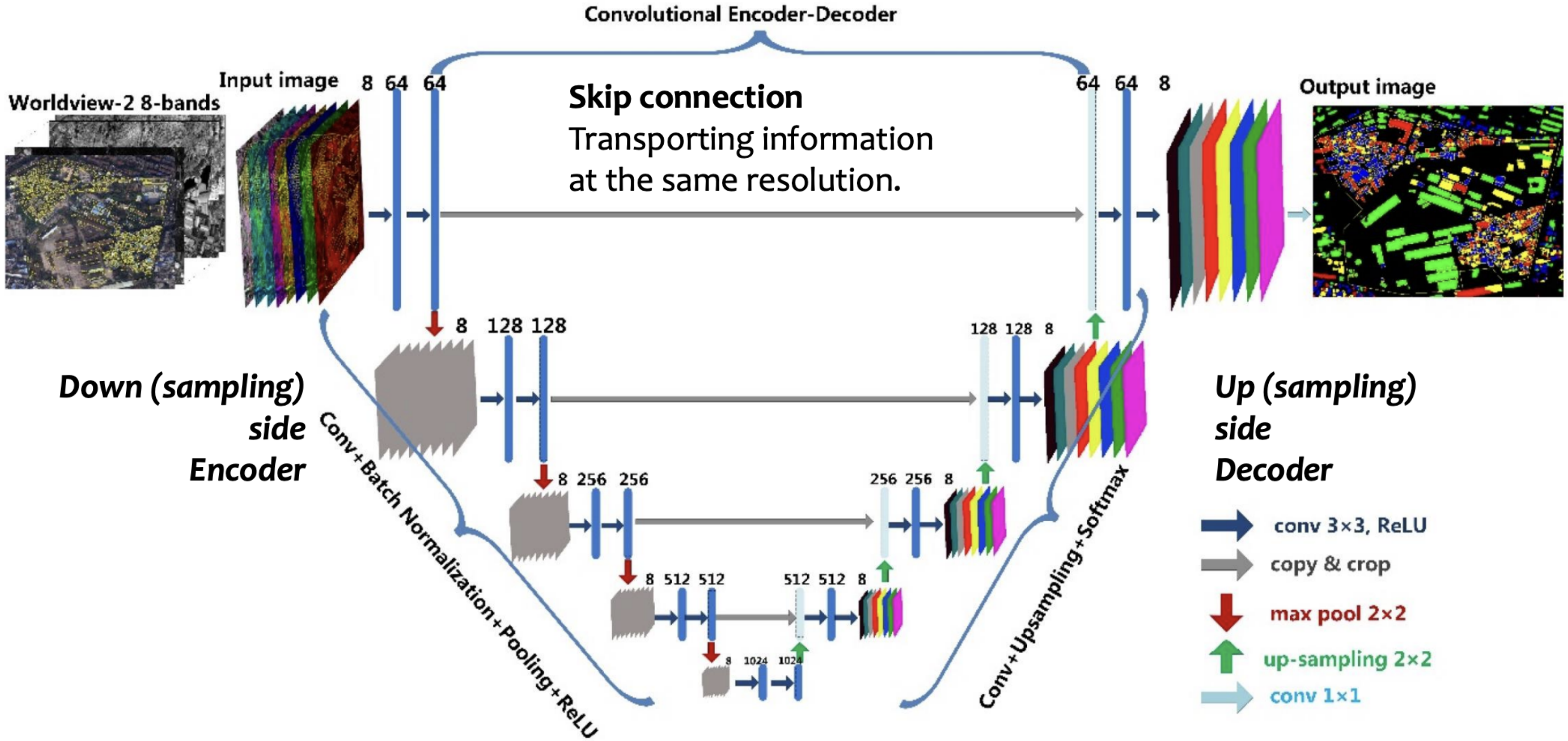


Reference: The Illustrated Stable Diffusion

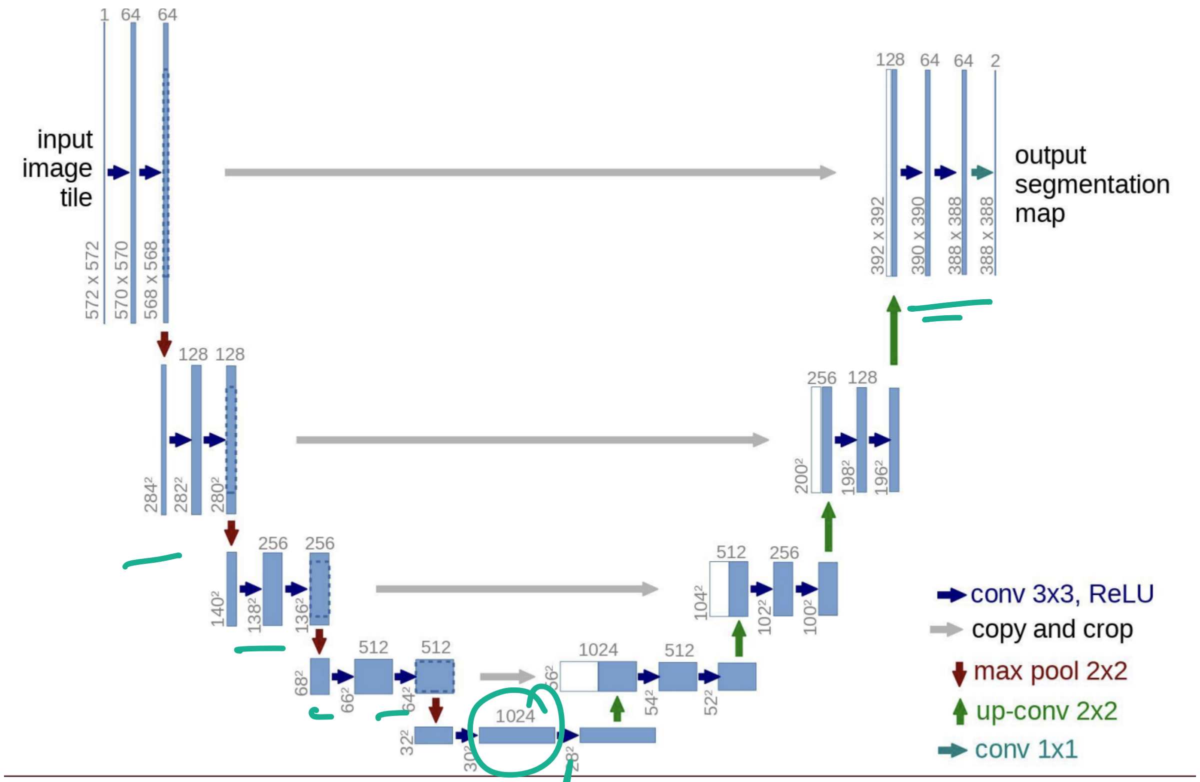
Image Generation: Training Data

Reference: The Illustrated Stable Diffusion

Unet Architecture

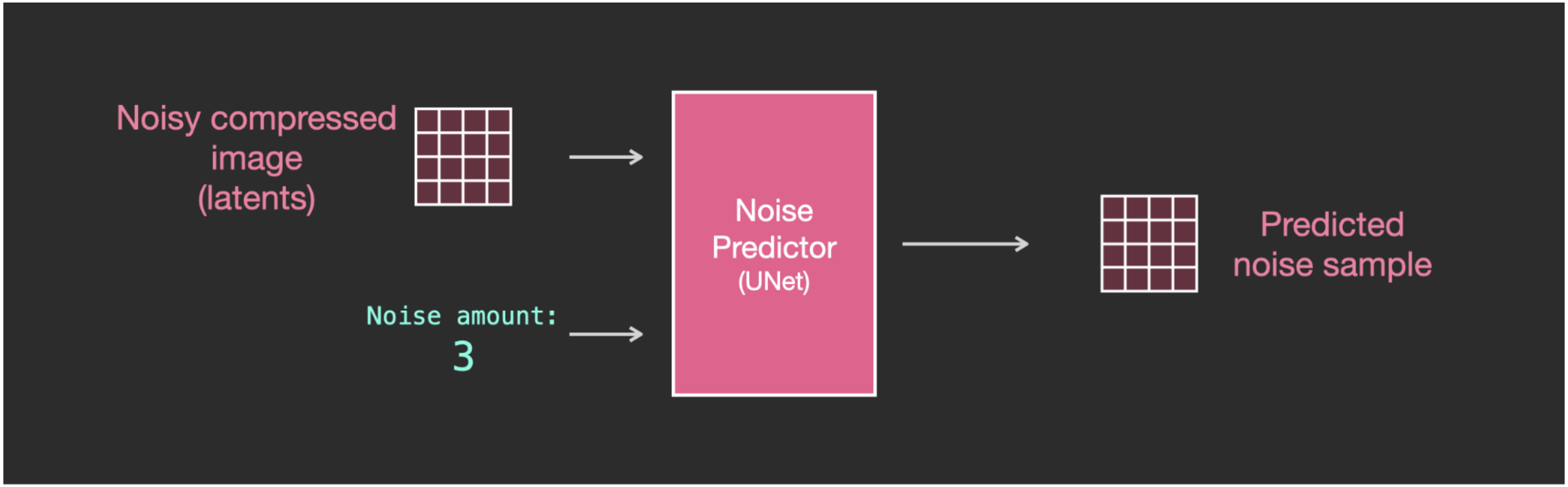


Unet Architecture



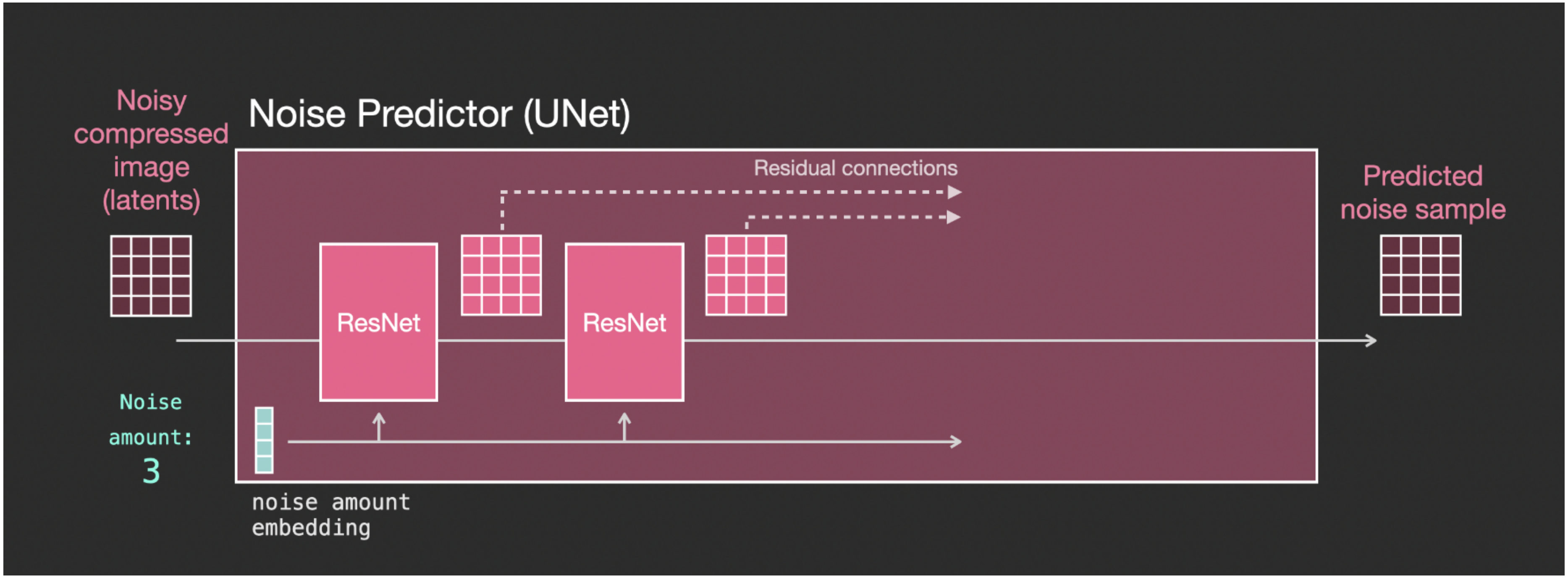
Code

Unet Predictor (Without Text)



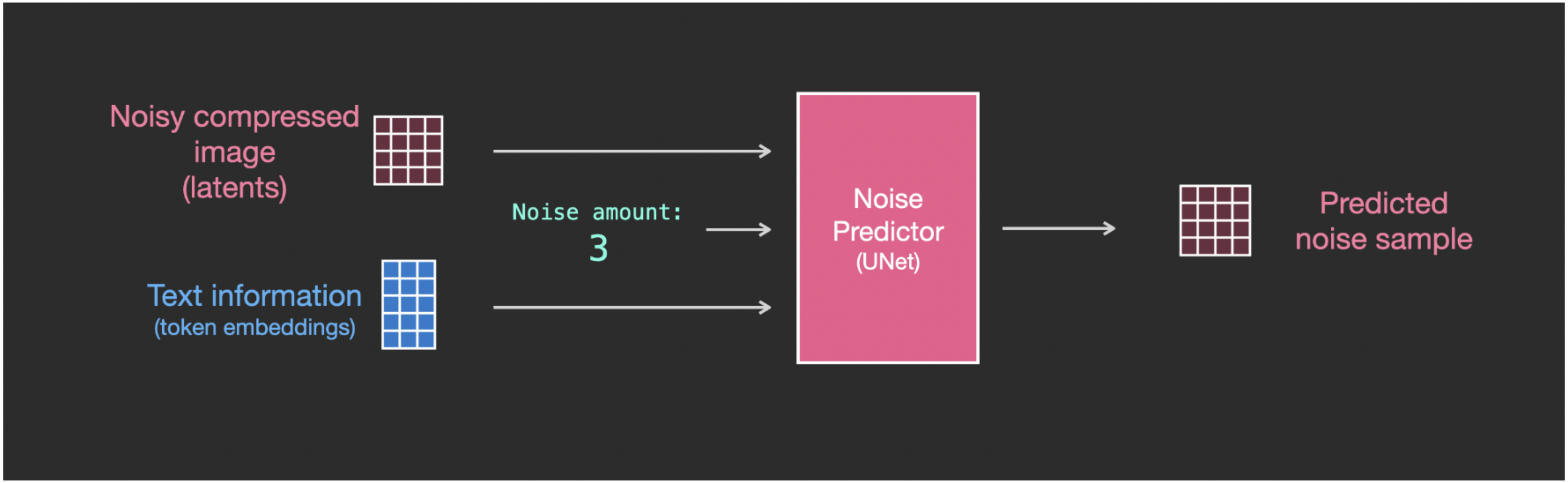
Reference: The Illustrated Stable Diffusion

Unet Predictor (Without Text)



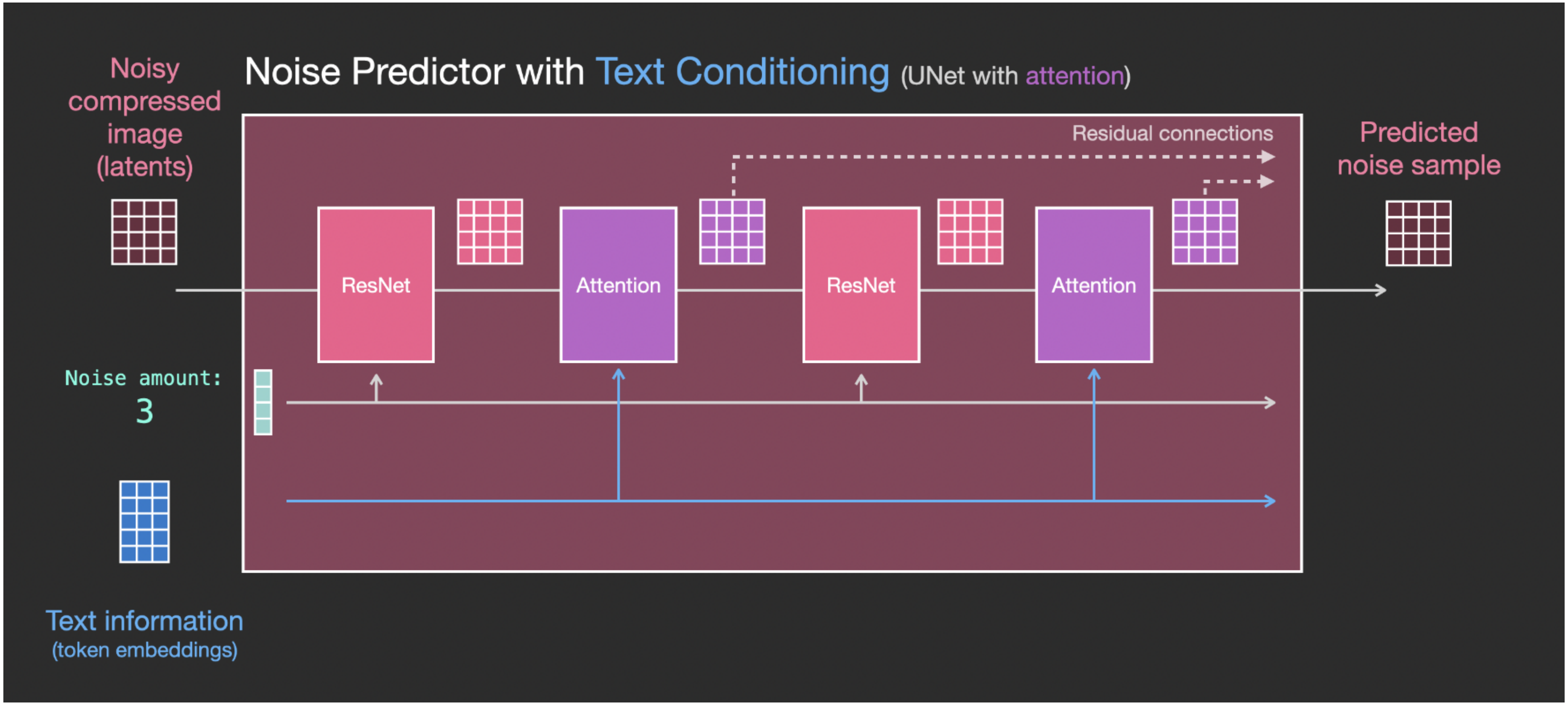
Reference: The Illustrated Stable Diffusion

Unet Predictor (With Text)



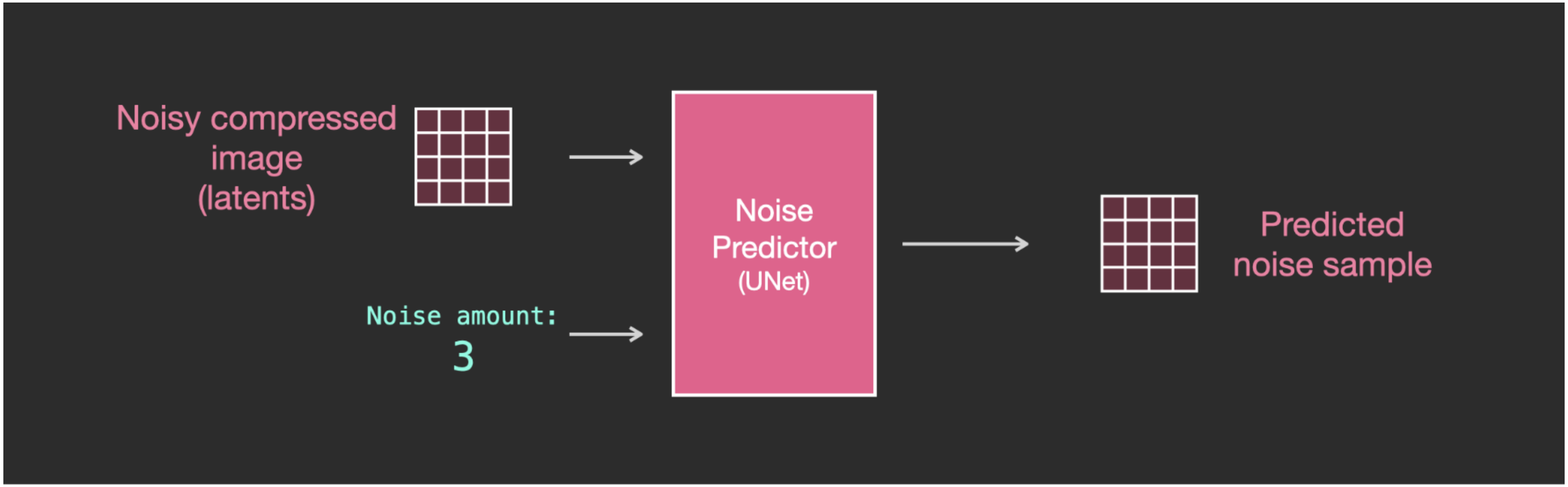
Reference: The Illustrated Stable Diffusion

Unet Predictor (With Text)



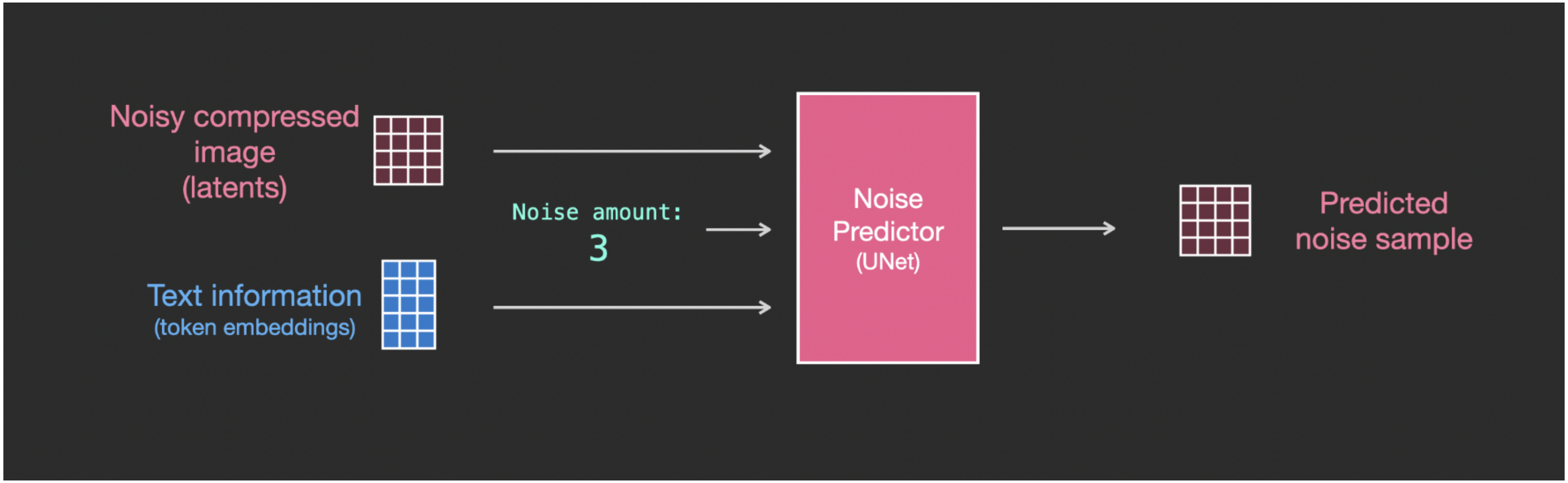
Reference: The Illustrated Stable Diffusion

Unet Predictor (Without Text)



Reference: The Illustrated Stable Diffusion

Unet Predictor (With Text)



Reference: The Illustrated Stable Diffusion