Generative AI Academy

Language / Text Generative Models

Transformers

Transformers are a type of neural network that have a **unique ability to recognize long-range connections within sequences**. They are particularly useful for **tasks like generating text**, as the model needs to comprehend the preceding words in order to produce the next one.

The introduction of transformers in 2018 was a groundbreaking moment for the field of natural language processing.

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* † illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after

Seminal paper published in 2017

Transformers – key capabilities

Transformers are able to learn long-range dependencies in sequences.

Transformers are able to be trained on very large datasets.

Transformers are able to be parallelized

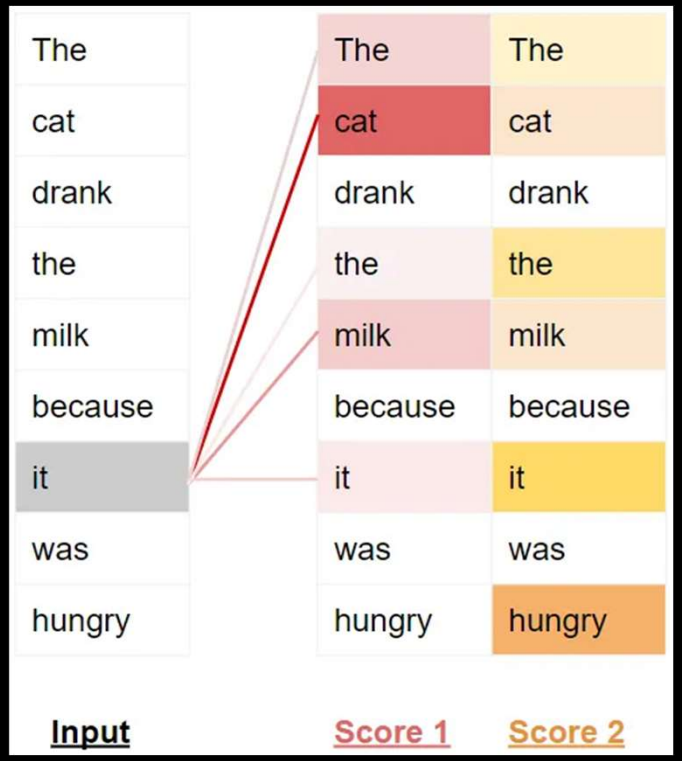
Transformer: Attention Mechanism

Consider two sentences:

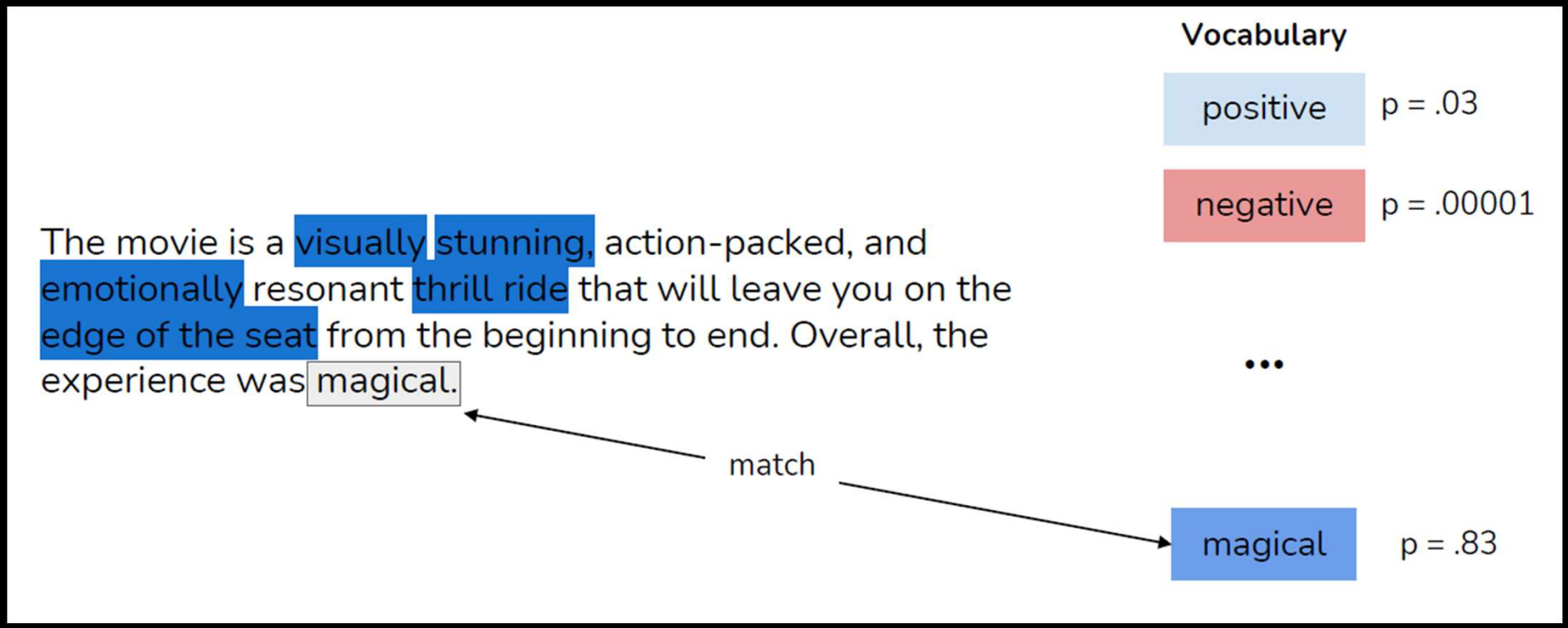
- The ***cat*** drank the milk because **it** was hungry.
- The cat drank the ***milk*** because **it** was sweet.

In the first sentence, the word ‘it’ refers to ‘cat’
in the second it refers to ‘milk.

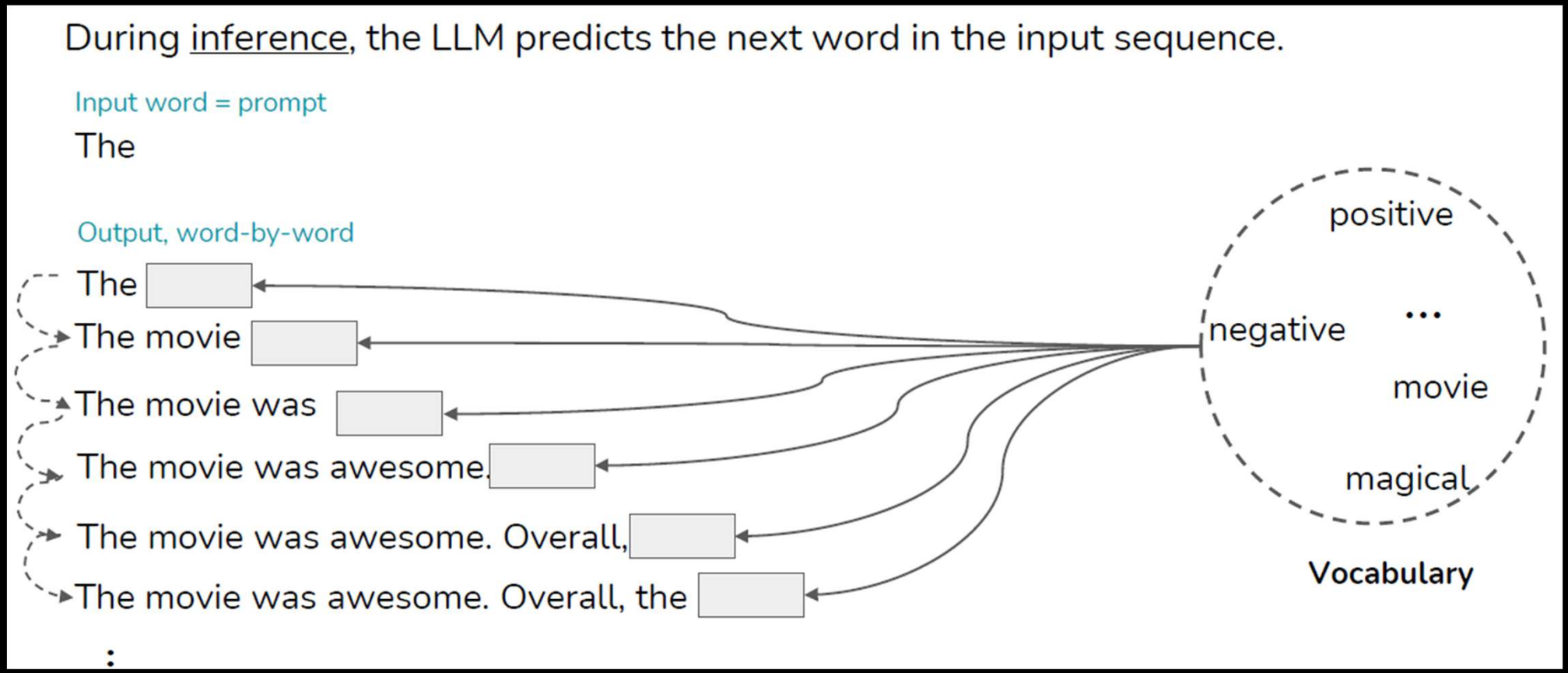
When the model processes the word ‘it’, **self-attention** gives the model more information about its meaning so that it can associate ‘it’ with the correct word.



Transformer: Next token prediction



Transformer: **Generate** text with next token prediction



Transformer: key concepts: Positional encoding

Because self-attention sees the sequence as an unordered bag of vectors, we need a tiny extra hint so

“dog bites man” ≠ “man bites dog”.

We simply add or concatenate a position vector:

```
x_i' = token_embedding_i + position_embedding_i
```

Transformer: key concepts: Self attention

Stage	"What happens" (plain language)	Why it matters functionally
1 · Three views of every token	For each input vector $\mathbf{x}_i$ the layer makes three linear projections: $\mathbf{q}_i$ =query (What am I looking for?) $\mathbf{k}_i$ = key (What do I contain?) $\mathbf{v}_i$ = value (What info can I give?)	Splitting roles lets the model ask: *How relevant is token j to token i?*
2 · All-to-all similarity check	Compute a dot-product matrix $\mathbf{q}_i \cdot \mathbf{k}_j$ for every pair (i,j).	One inexpensive matrix-multiply lets every token "glance" at every other token **in parallel** (no recurrence).

Multi headed attention

Single–head (one lens)	Multi-head (many lenses)
One set of attention weights tries to capture every relationship (subject–verb, adjective–noun, long-range theme) all at once.	The model splits its "vision" into *h* smaller lenses (heads). Each head can specialise: one notices grammar, another spots coreference, another tracks long-distance topics, etc.

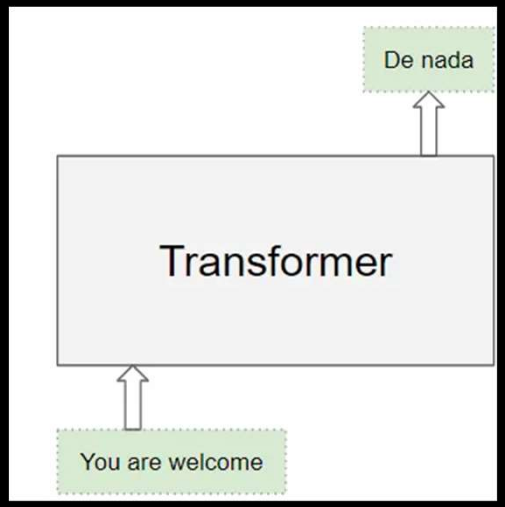
Multi headed attention – under the hood

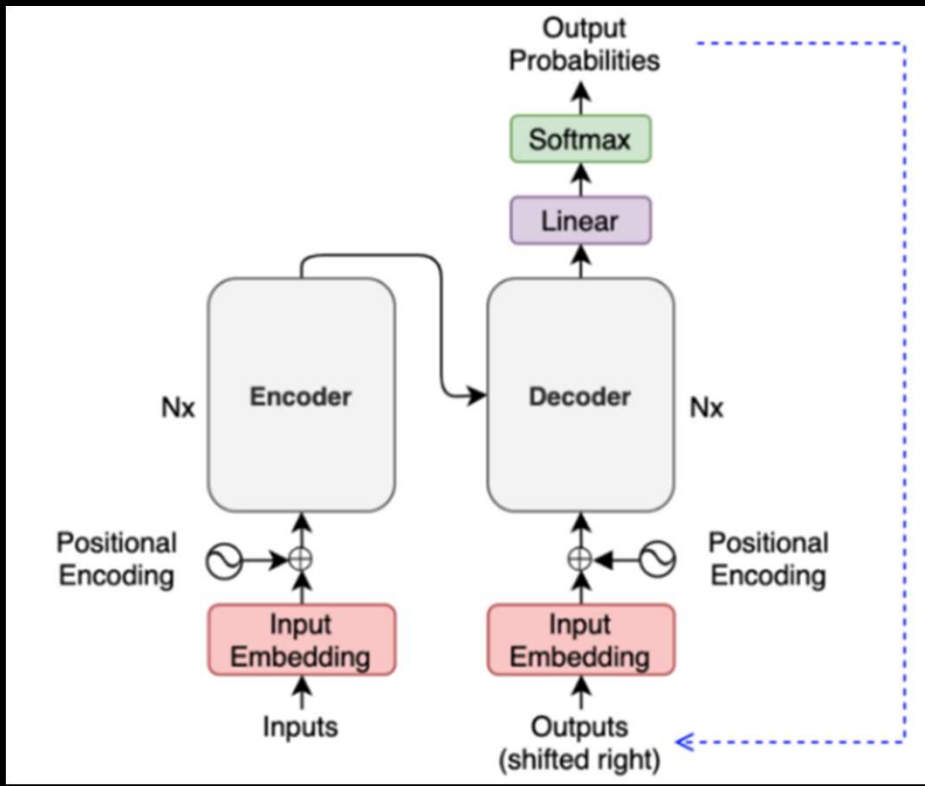
1. **Copy the same input vectors h times.**
2. **Give each copy its own tiny pair of glasses** (different weight matrices) so it sees the sentence from a unique angle.
3. **Let every head do its own attention calculation** (build its own weight map).
4. **Concatenate the heads**—like stacking several transparencies—and run them through one last linear layer so the information blends back together.

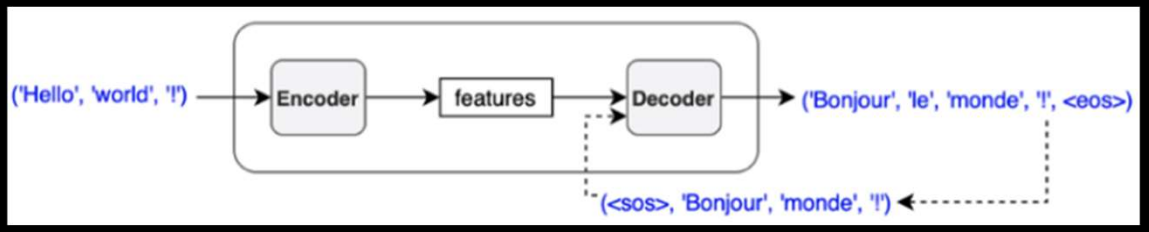
> **Analogy:** Think of a detective squad. All detectives read the same case file, but each focuses on a different clue type—fingerprints, alibis, motives. At the end they pool their findings to solve the case.

Encoders and Decoders

The **encoder** extracts features from an input sentence, and the **decoder** uses the features to produce an output sentence







Encoders and Decoders

Paradigm	Masking & Direction	Typical Objectives	Example Models
Encoder-only (bidirectional)	Full context each side; no causal mask.	Masked-Language-Model (MLM), contrastive.	**BERT**, RoBERTa, DeBERTa
Decoder-only (causal)	Left→right; causal mask.	Next-token prediction, RLHF.	**GPT-3**, LLaMA-2, Mixtral
Encoder-Decoder (Seq-to-Seq)	Encoder = bidirectional; Decoder = causal + cross-attn.	Denoising, sequence transduction.	**T5**, BART, FLAN-T5

Encoders and Decoders

Multi-Head Attention
Multiple learned projections let the model attend to different semantic sub-spaces in parallel.

Positional Encoding:
injects order into self-attention (which is permutation-invariant).

Causal Model:
decoder-only transformer where each position can only attend to earlier tokens, enabling generation.

Bidirectional/Masked Model:
encoder-only transformer that can look both left and right to fill in masked tokens; excels at understanding.

Encoder-Decoder: combines both to translate or transform sequences.

Typical training flow

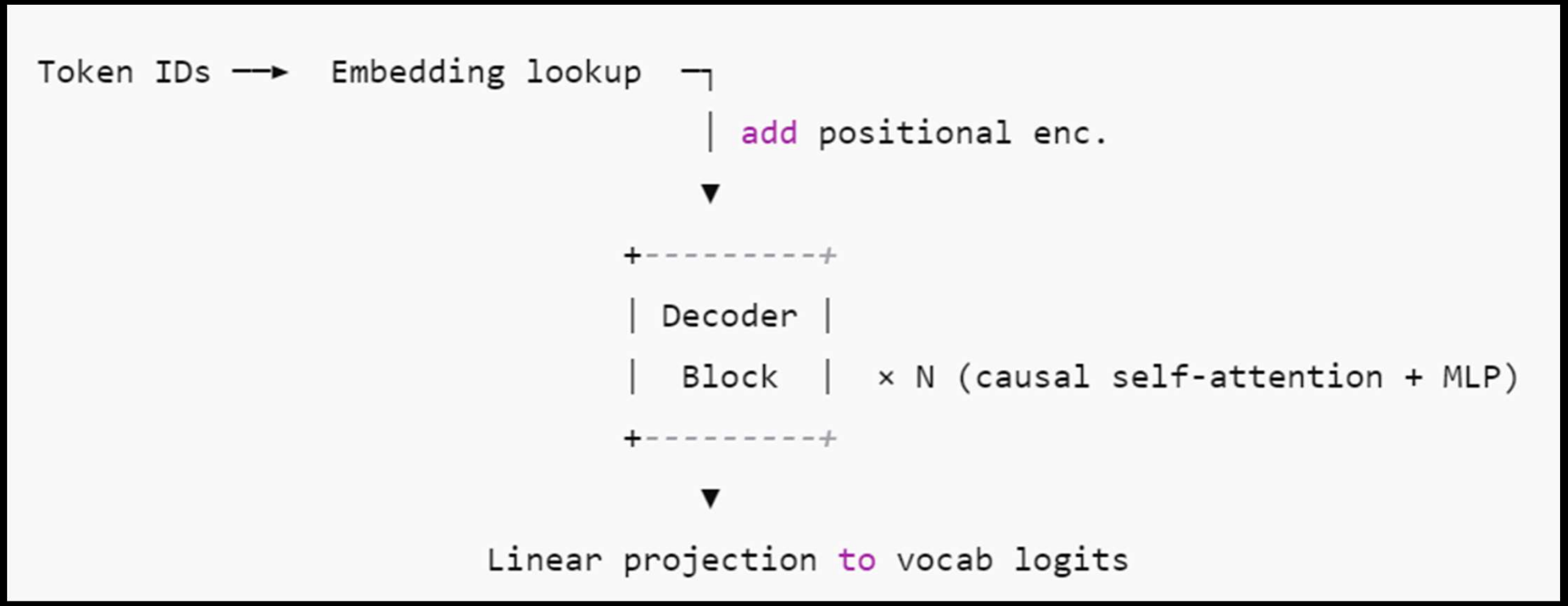
Step 42 318 of training:

- The input pipeline streams 4 096 sequences of 2 048 tokens each (≈ 8 M tokens) onto 128 GPUs.
- Embedding look-ups turn token IDs into vectors; positional encodings are added.
- All 8 M vectors race through the 32 transformer layers in parallel.
- The model outputs a probability for the next token at every position; the loss function compares that to the ground truth already sitting in memory.
- Back-prop computes gradients; GPUs exchange them, average, and adjust weights.
- The optimizer writes the new weights; the step is done.
- New data streams in for step 42 319. None of the old tokens are revisited unless you eventually start a second epoch (rare for 10 T tokens).

Encoder only and **Decoder only** Models

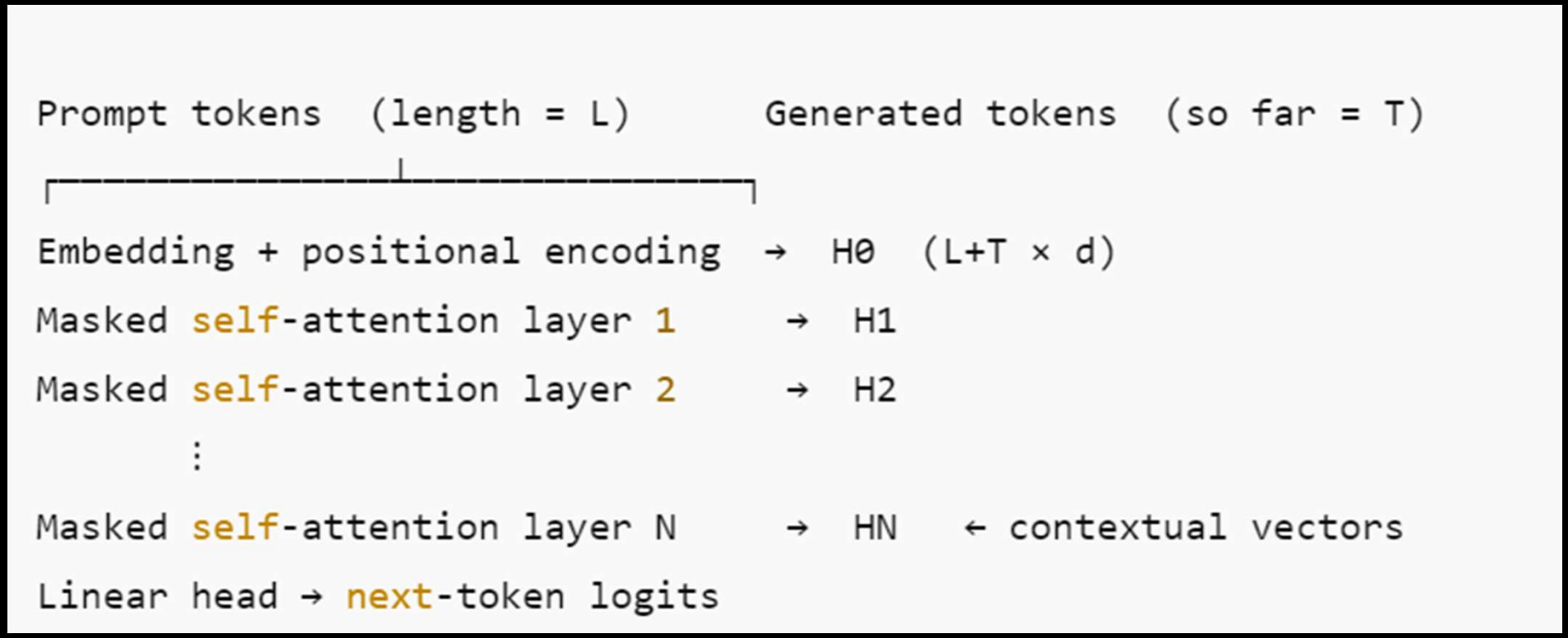
Aspect	Encoder-Only (BERT-style)	Decoder-Only (GPT-style)
Attention mask	**Bidirectional** – every token can attend left **and** right.	**Causal (left-to-right)** – each token sees only earlier tokens; future positions are masked.
Typical pre-training task	Masked-Language Modeling (MLM): hide 15 % of tokens and predict them.	Next-Token Prediction (autoregressive): predict token $t + 1$ from tokens $\leq t$.
Output style at inference	Produces a *fixed-length* vector per input token (or pooled sentence vector). Not designed to stream new tokens.	Generates text *token-by-token*; can keep extending a sequence indefinitely.
Common downstream uses	• Text classification • Named-entity recognition • Embedding extractor • Reranking / retrieval	• Chatbots, story generation • Code completion • Few-/zero-shot reasoning • Text-to-SQL, translation via prompting

Training flow in Decoder only Model



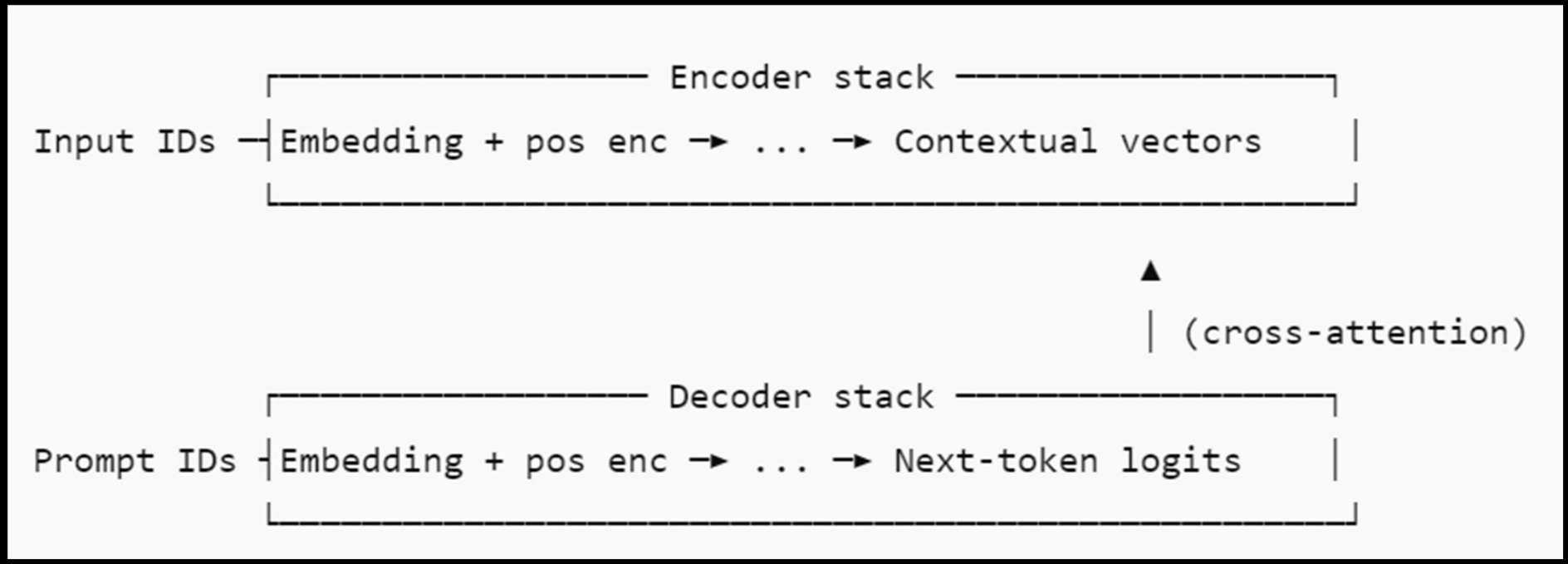
MLP = Multi layer perceptron

Training flow in Decoder only Model



MLP = Multi layer perceptron

Training flow in Encoder + Decoder Model



What would happen without the contextual vector?

What would break without that contextual step?

- **Wrong word choice:** Static embedding might map “Bonjour” near “Bonbon” (candy) or “Bonjourno” (misspelled Italian), confusing the decoder.
- **Poor scaling to longer sentences:** Reordering, agreement, and coreference all rely on rich encoder states; a raw lookup can’t carry that load.
- **Loss of language transfer:** The learned cross-lingual space collapses if encoder vectors aren’t context-aware; the decoder’s queries would have no consistent target to lock onto.

Three concrete benefits of the contextual vector

1. **Disambiguation & register**

   If the French sentence were « *Bonjour, Madame la Présidente* », the encoder's surrounding context would push H_{enc-Bonjour} toward a *formal* greeting, nudging the decoder to pick "Good morning" instead of the casual "Hello".

2. **Polyglot vector alignment**

   Training forces the decoder's query direction for *greeting* to align with the encoder's *greeting* vectors—independent of language. The shared high-dimensional geometry is why translation works at all.

3. **Robustness to typos / sub-word splits**

   Even if "Bon-jour" appears in sub-word form (`_Bon` , `jour`), self-attention fuses them so the decoder receives one coherent "greeting" signal instead of two half-words.

Encoder + Decoder model often superior for language translation

Encoder-decoder	Decoder-only
Bidirectional encoder captures both left & right context → richer source representation.	Source tokens only look *leftwards* within themselves; in practice this is fine but not as expressive as bidirectional encoding.
Cross-attention cleanly separates "read source" from "write target"; easy to train with teacher forcing.	Source + target share context window → longer sequences, harder length limits, possible interference.
More interpretable alignment (attention heat-maps).	Alignment is implicit in self-attention.

Where decoder only (causal language models) shine?

- **Open-ended text generation**
  (stories, marketing copy, creative writing, ideation)
- Chat & multi-turn assistants
- Code completion / synthesis

Popular Language models built on Transformer Architecture

GPT

Generative Pre-trained Transformer

Decoder only

Open AI. GPT-1 released in Jun 2018 (GPT 3.5 in 2022 Nov)

BERT

Bidirectional Encoder Representations from Transformers)

Encoder only, not an LLM (100M odd parameters)

Google. Oct 2018

T5

Encoder and decoder

Google.

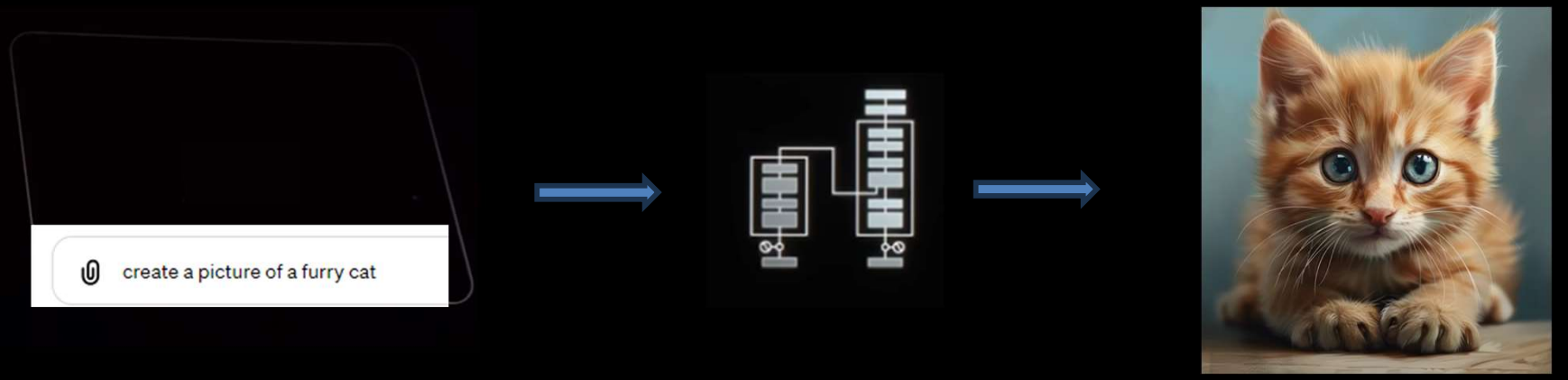
Generative Text Models

- **Transformers** dominate this space now
- RNN, LSTM, GRU still used depending on use-cases
- VAE and GAN have also been adopted (TextGAN, SeqGAN)
- Advanced architectures such as Memory Augmented Networks, Graph Neural Networks are being researched

Video: transformers basics

Video: How transformer does prediction / generation of text

Under the hood – generating picture of cat from text

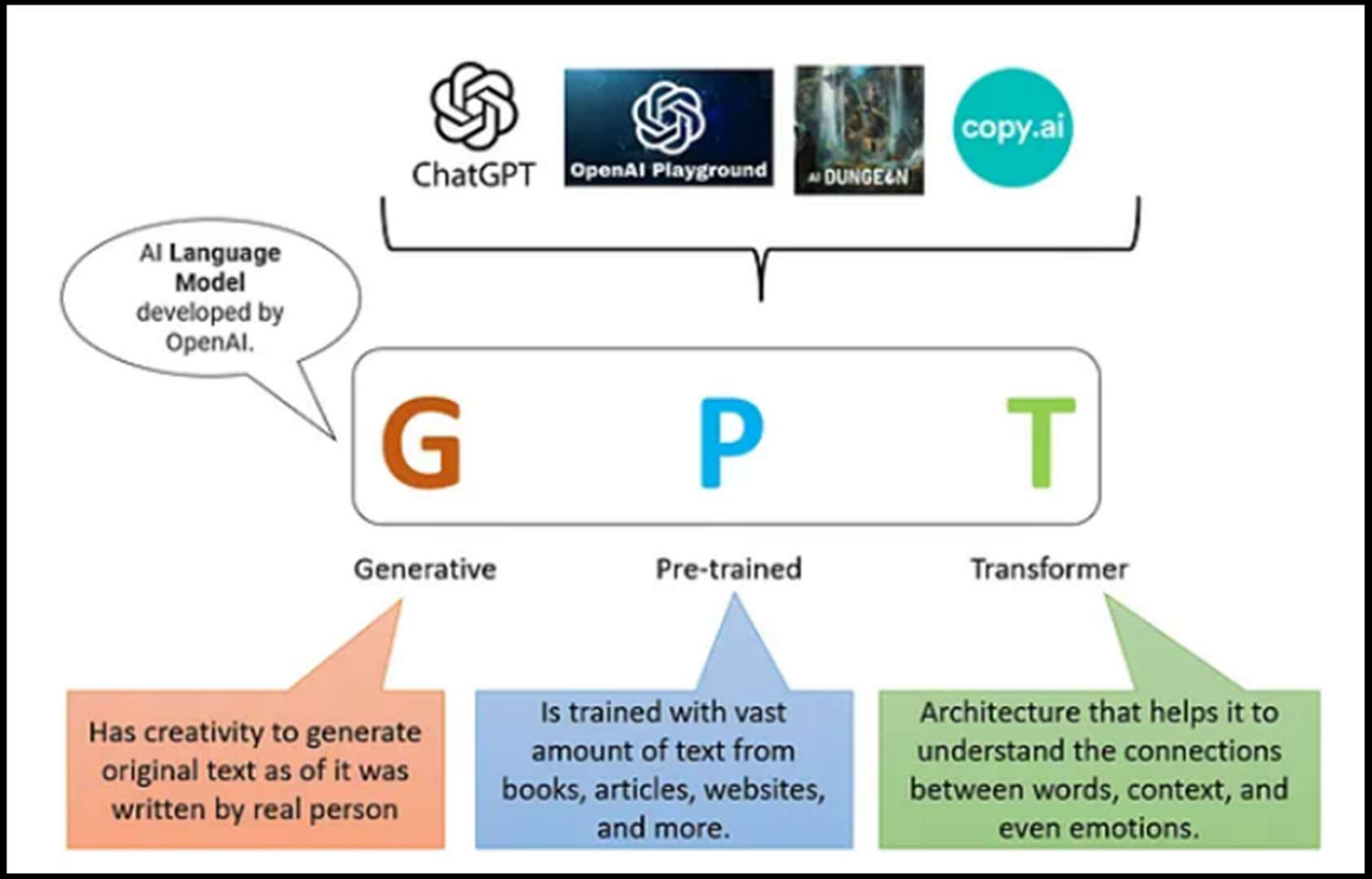


GenAI chatbot - prompt **GPT** **Picture of Cat**

- Features of cat learnt from training data – images of cats (using convolutional neural network) – converted into embeddings
- Text “create picture of…” converted to embeddings using self-attention mechanism (captures dependencies among the input tokens)
- The embeddings are matched to find the common embedding spaces (the text and image will have same embedding positions hence making it possible to generate a picture from text)

Video: Image generation from text - goldfish

GPT

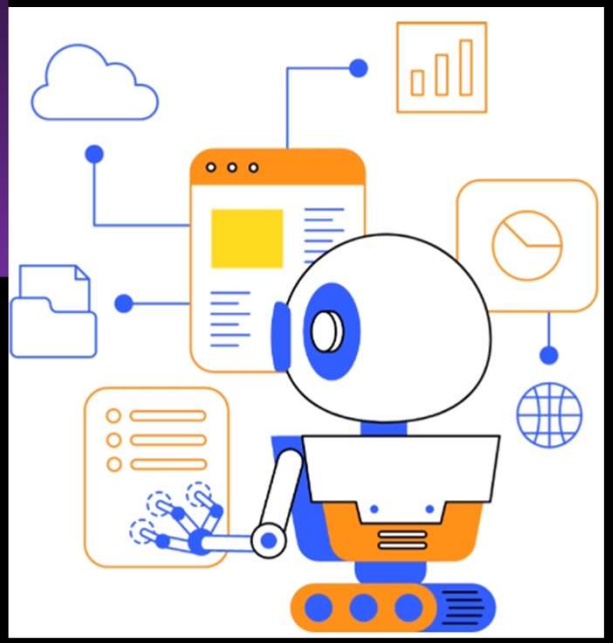

ChatGPT
OpenAI Playground
AI DUNGEON
copy.ai
AI Language Model developed by OpenAI.
G
P
T
Generative
Pre-trained
Transformer
Has creativity to generate original text as of it was written by real person
Is trained with vast amount of text from books, articles, websites, and more.
Architecture that helps it to understand the connections between words, context, and even emotions.

Generative AI

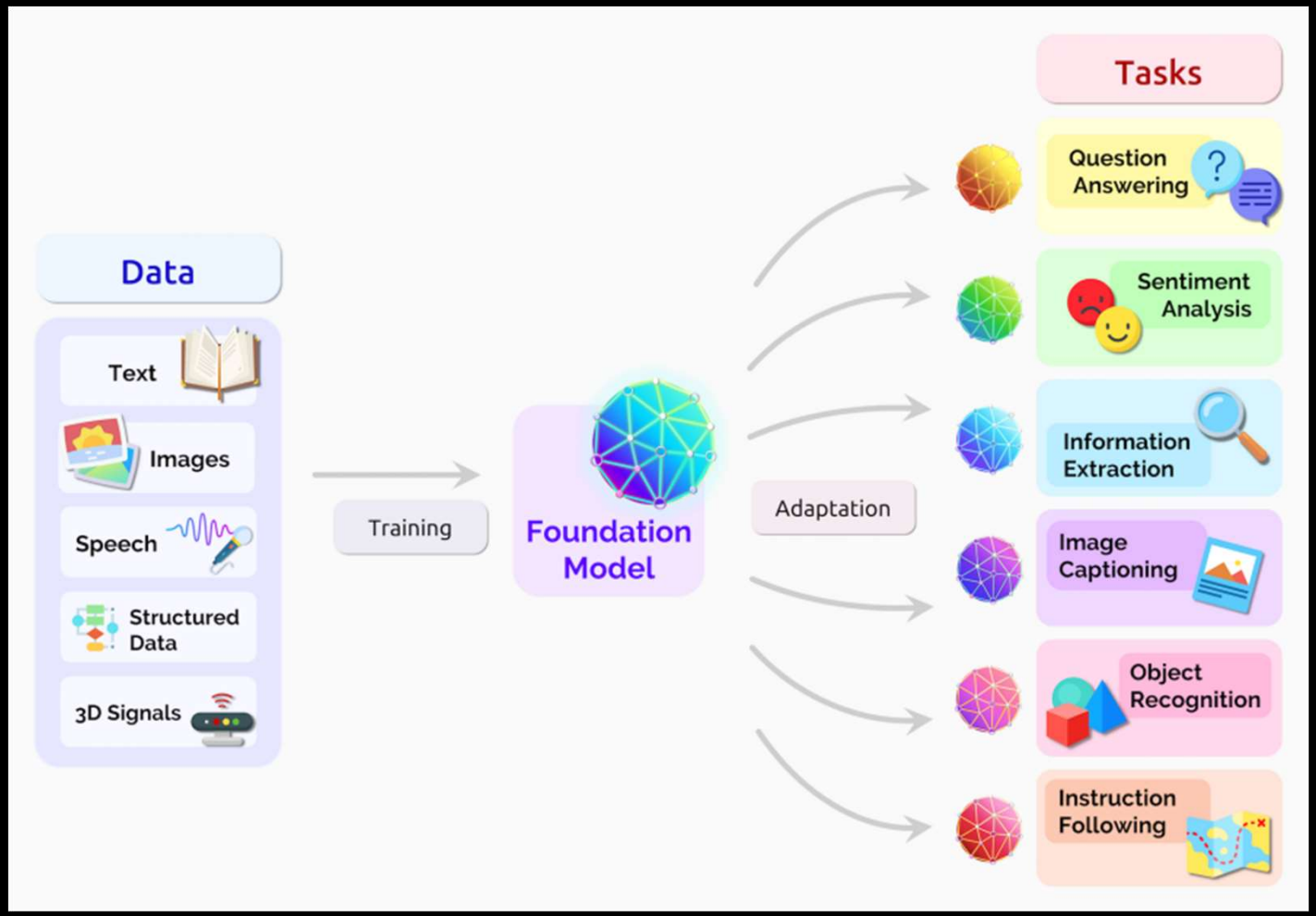
Generative AI

is a type of artificial intelligence that focuses on creating new content. It's a subset of machine learning, drawing from techniques like deep learning and reinforcement learning to generate output that can include text, images, music, video, and more.

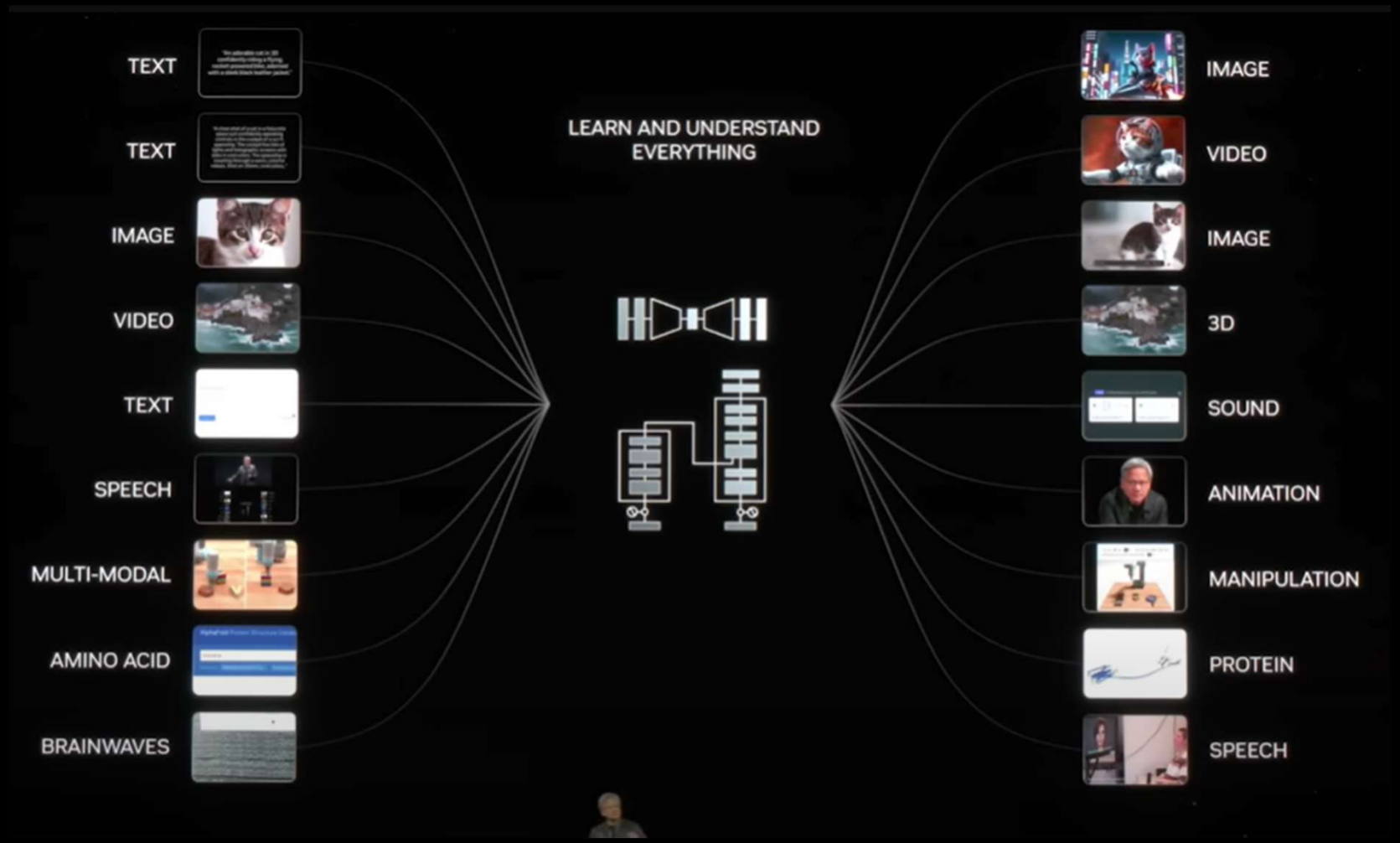
Combination of GPT (or other **generative** models) with a chat interface



Foundation Models



Multi modal models

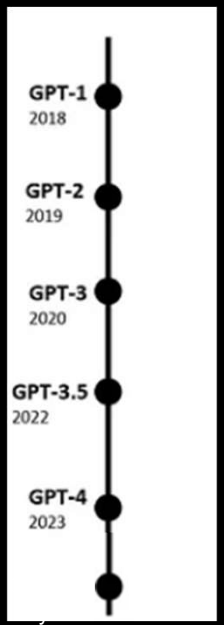


LLM – Large Language Model

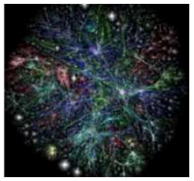
General-purpose language model

- pre-trained on massive data sets
- uses deep learning techniques
- can be fine-tuned for specific purposes
- generate natural language outputs

OpenAI



Steps involved in creating a LLM



Chunk of the internet, ~10TB of text

every ~year

every ~week

Stage 1: Pretraining
1. Download ~10TB of text.
2. Get a cluster of ~6,000 GPUs.
3. Compress the text into a neural network, pay ~$2M, wait ~12 days.
4. Obtain **base model**.

Stage 2: Training

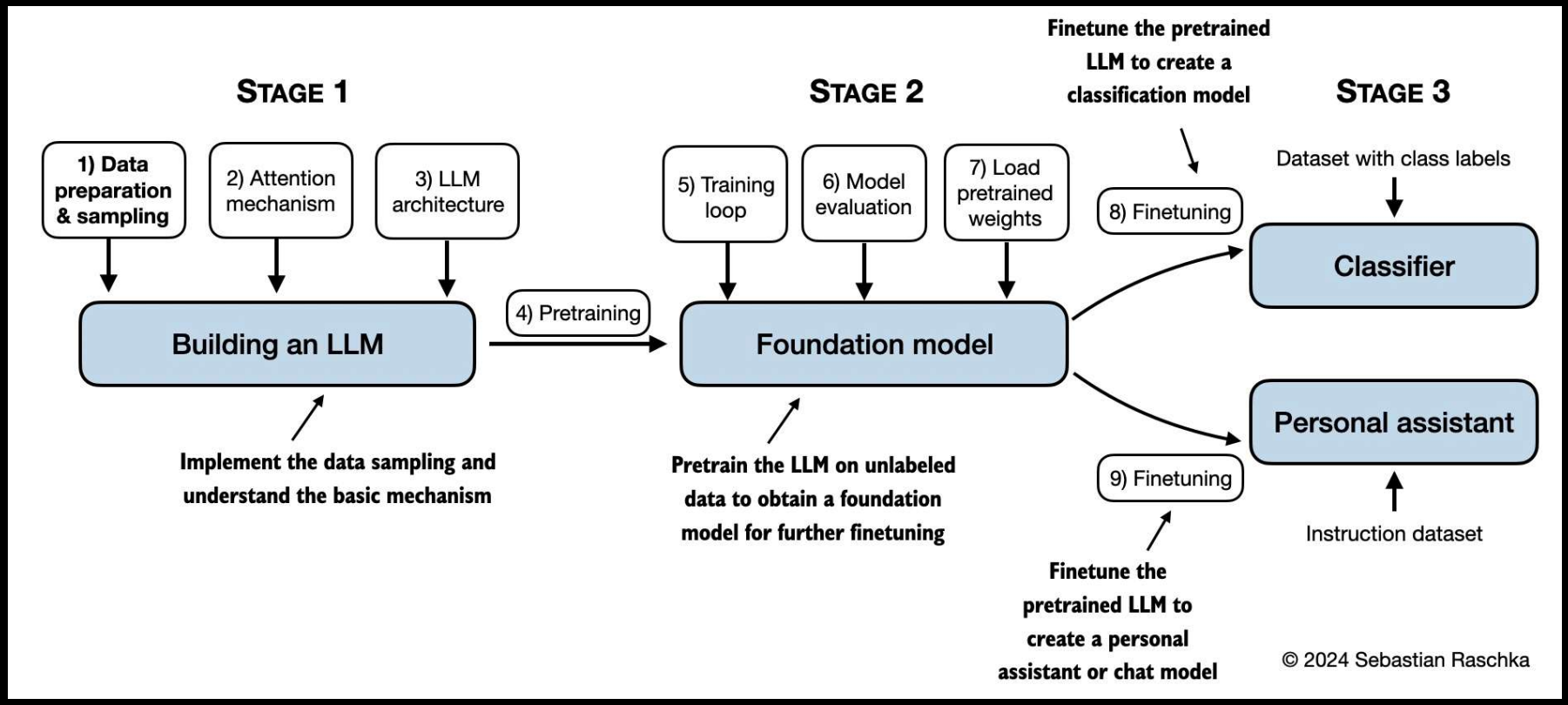
6,000 GPUs for 12 days, ~$2M
~1e24 FLOPS

~140GB file

(for Llama2 70B)

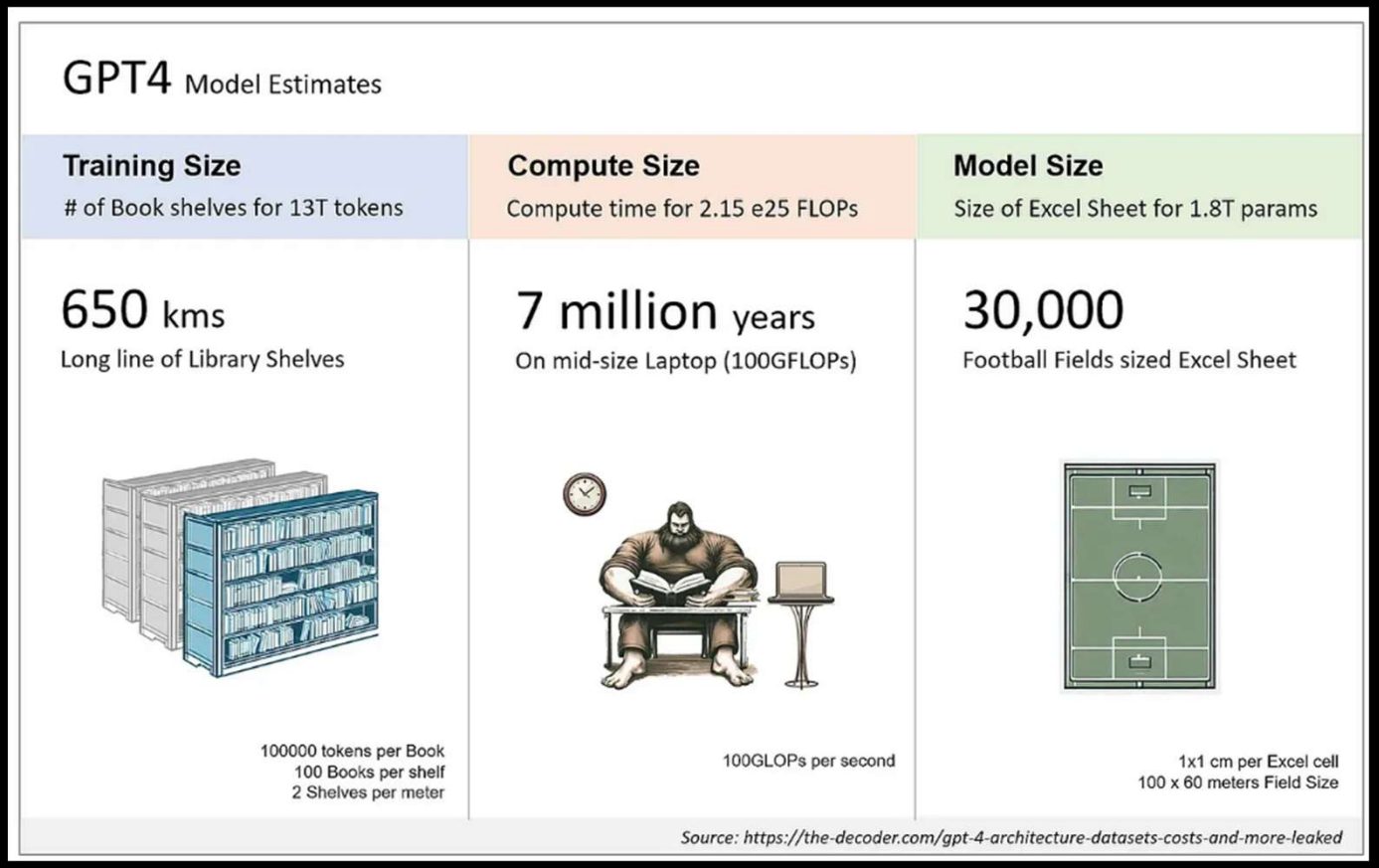
Stage 3 Finetuning
1. Write labeling instructions
2. Hire people (or use scale.ai!), collect 100K high quality ideal Q&A responses, and/or comparisons
3. Finetune base model on this data, wait ~1 day.
4. Obtain **assistant model.**
5. Run a lot of evaluations.
6. Deploy.
7. Monitor, collect misbehaviors, go to step 1.

Steps involved in creating a LLM



https://github.com/rasbt/LLMs-from-scratch

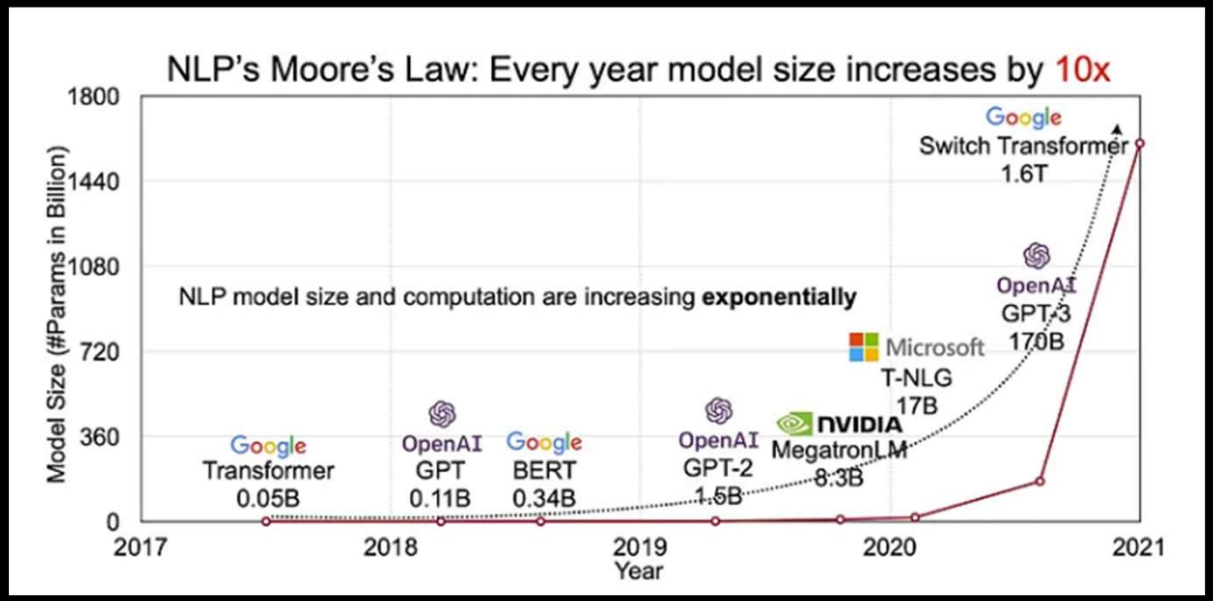
GPT4 training size estimates

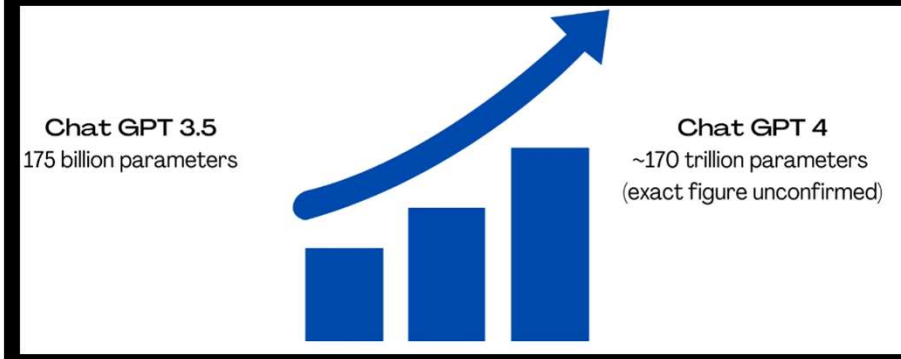


GPT 3.5 training resources and duration
90-100 days to train on 25,000 Nvidia A100 GPUs

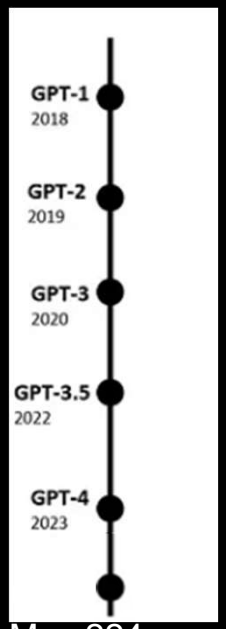
3,125 servers
25,000 GPUs
3,125 * 14,040 to 16,900 KWh
43,875,000 to 52,812,500 KWh

GPT Model size





LLM offerings have mushroomed in less than 2 years!

OpenAI	Microsoft CoPilot	Google Gemini	Meta Llama	Anthropic Claude	Mistral
	• Azure OpenAI **Jan 2023** • Integrated with Office365, GitHub, Power Platform, Teams	• Bard Mar 2023 • Gemini **Feb 2024**	• Opensource • Llama2 **Jul 2023** • Llama3 **Apr 2024**	• AWS adoption • Claude2 **Jul 2023** • Claude3 **Mar 2024**	• Opensource • Mistral 7b **Sept 2023** • Mistral **small Feb 2024** • Mistral 8x 22B **Apr 2024**

May 224

Generative Image Architectures

- Generative Adversarial Network (GAN)

  Invented: 2014 **By:** Ian Goodfellow et al

- Variational Auto Encoder (VAE)
  Invented: 2013 **By:** Kingma, Diederik P., and Max Welling

- Stable Diffusion
  Invented: 2022 **By:** Stability AI – led by Emad Mostaque
  Paper: *"High-Resolution Image Synthesis with Latent Diffusion Models"*

- Vision Transformers (ViT)

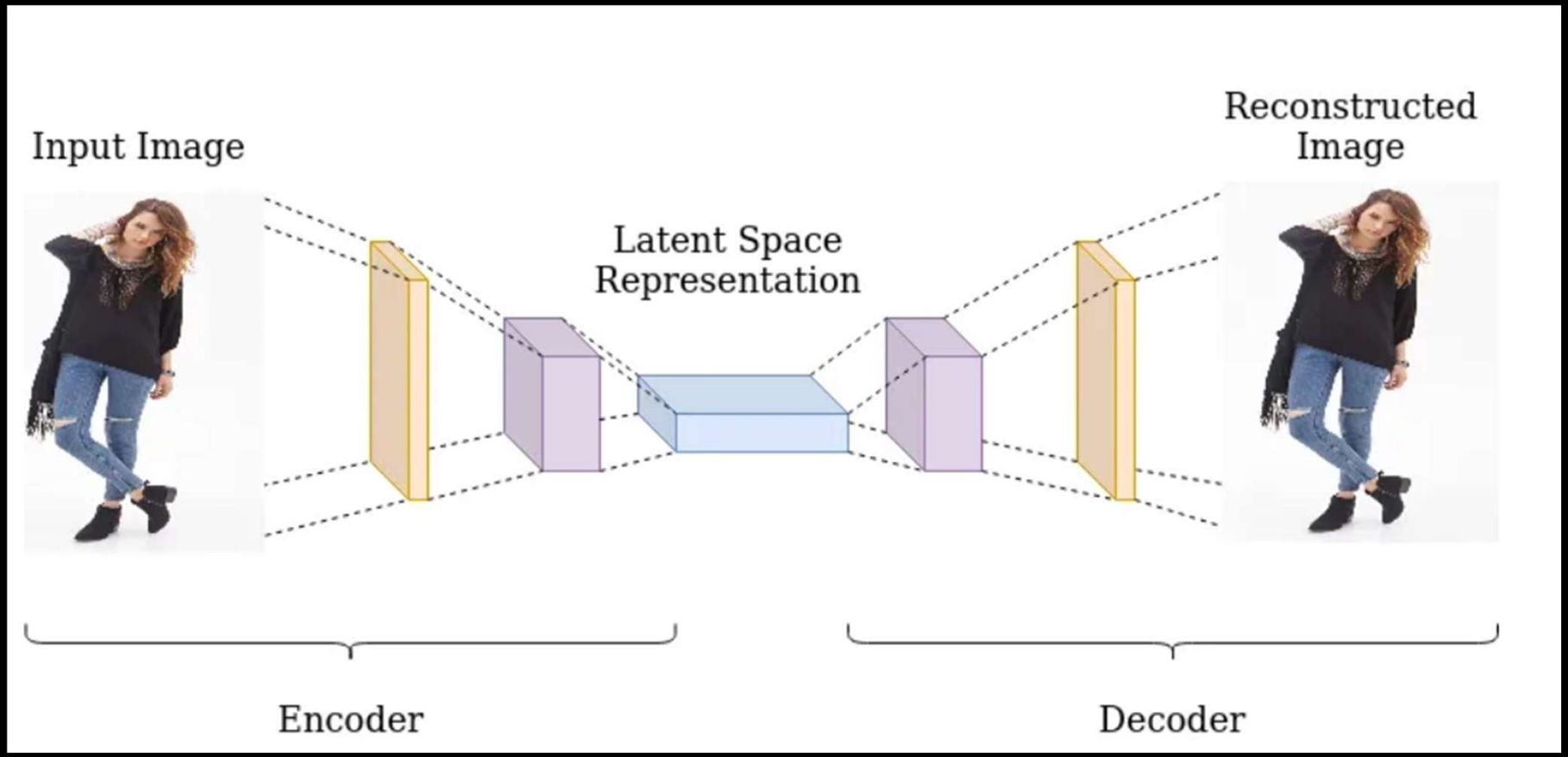
- DALL-E

- Midjourney

Popular image generation models

- DALL-E by Open AI
- Midjourney by Midjourney
  initial version: 2022, v6: Dec 2023
- Stable Diffusion XL
  Opensource
- Imagegen
  Google
- Firefly
  By Adobe

Encoders and decoders



Encoding techniques: PCA and Autoencoders

Feature	PCA (Principal Component Analysis)	Autoencoders
	Linear transformation technique	Non-linear neural network-based encoding
Supervision	Unsupervised (no labels required)	Unsupervised (self-supervised learning)
Feature Extraction	Identifies linear correlations between features	Learns complex, non-linear feature representations
Dimensionality Reduction	Reduces dimensions by projecting data onto principal components	Compresses data using a neural network bottleneck
Computational Complexity	Low (based on eigenvalue decomposition/SVD)	High (requires training a neural network)
Interpretability	Easy to interpret (principal components are linear combinations of features)	Hard to interpret (learned latent space is abstract)
Non-linearity Handling	❌ Cannot capture non-linear relationships	✅ Captures complex, non-linear patterns
Scalability	Efficient for small to medium datasets	Scales well but requires more computational power
Data Reconstruction	Approximate reconstruction (lossy) using principal components	Can reconstruct input but may not be perfect (depends on loss function)
Flexibility	Fixed transformation (does not adapt to new data)	Can be trained on specific datasets and fine-tuned
Robustness to Noise	Sensitive to noise in data	Denoising autoencoders can effectively remove noise

Encoding and Decoding example

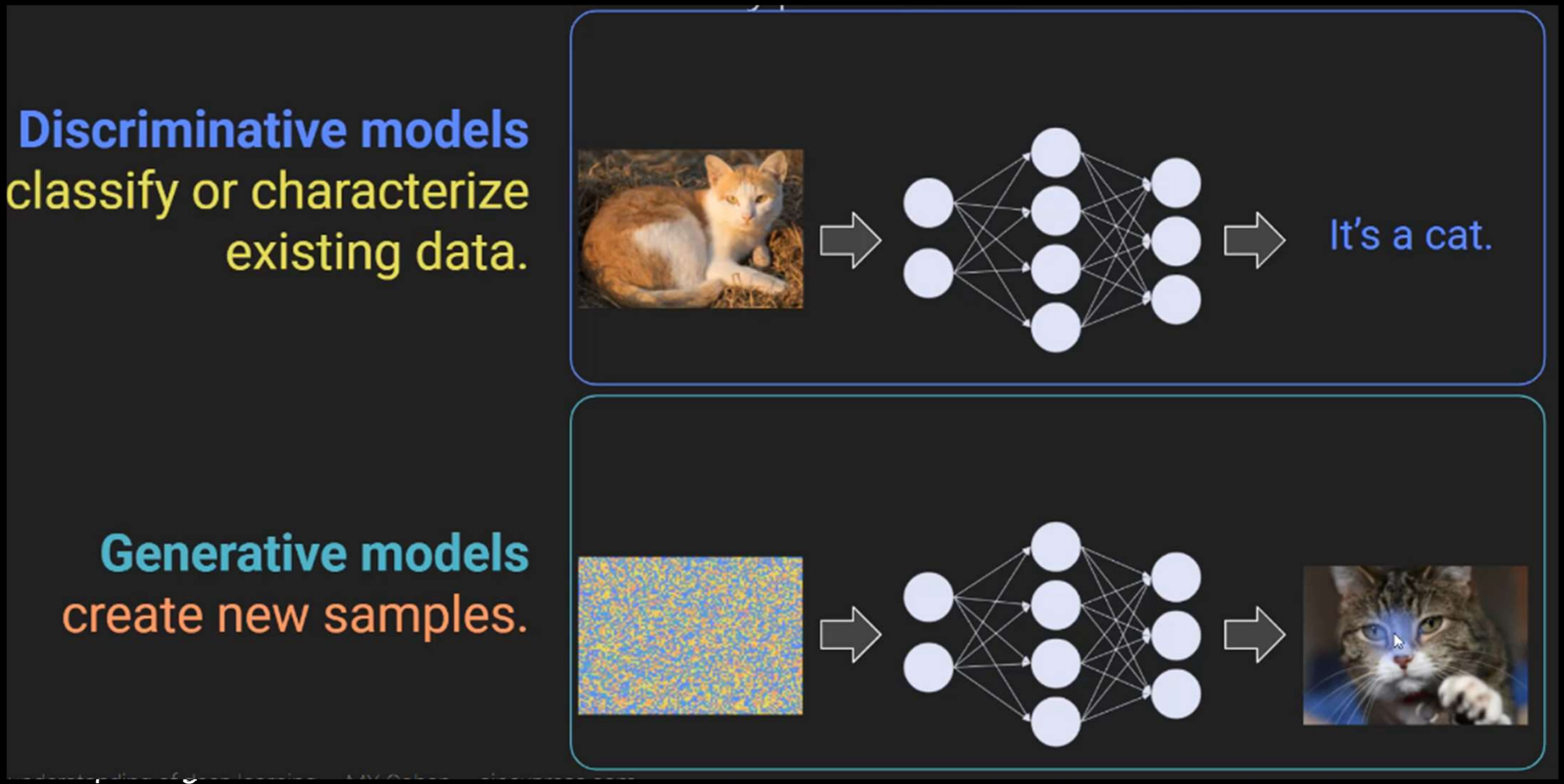
Example: Image Compression

Consider a high-resolution image (128×128 pixels).

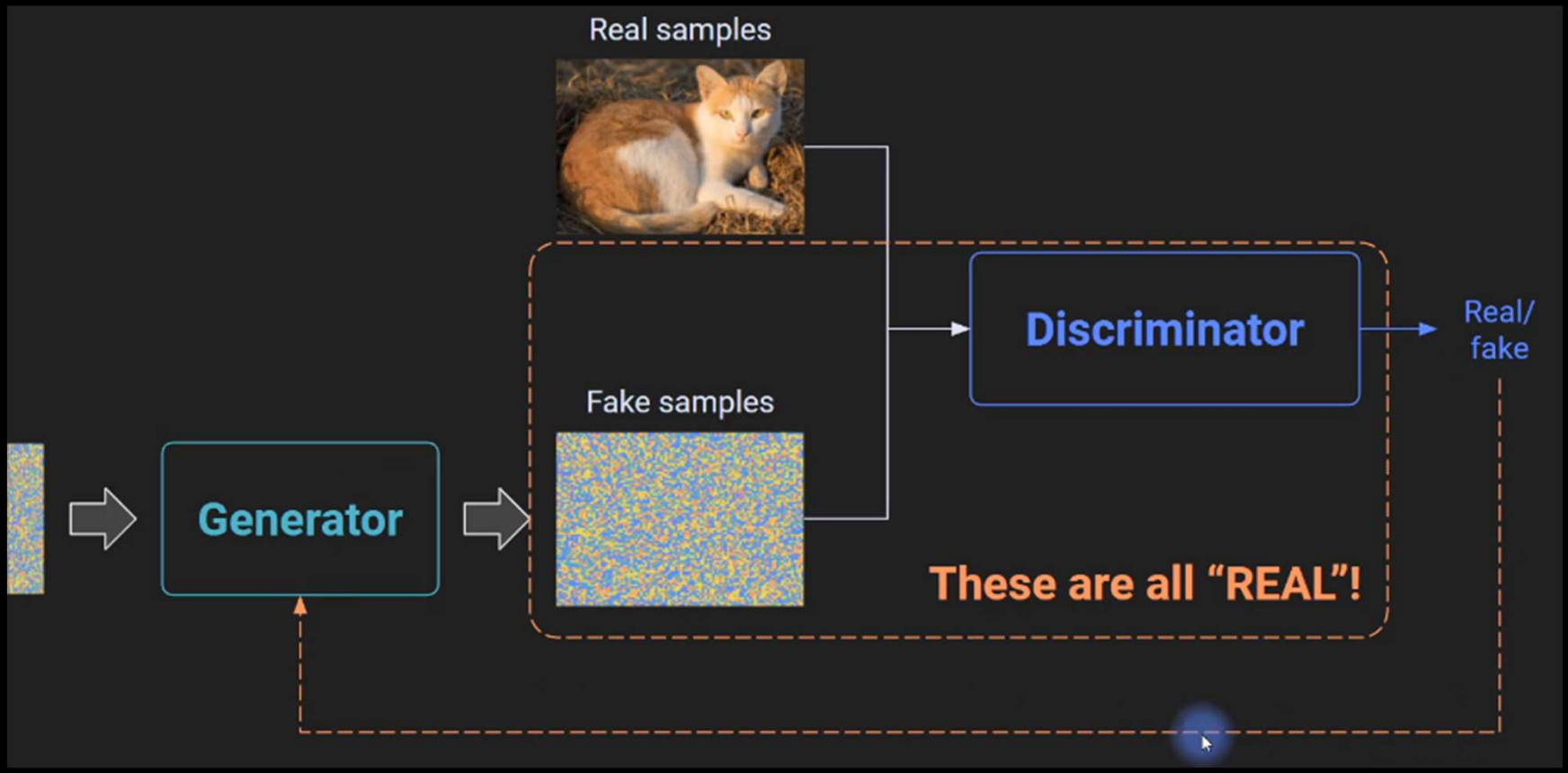
Instead of storing all 16,384 pixels (128×128), an autoencoder might encode it into a vector of just 100 numbers.

The decoder then tries to reconstruct the original image from just those 100 numbers.

Discriminative and Generative Models

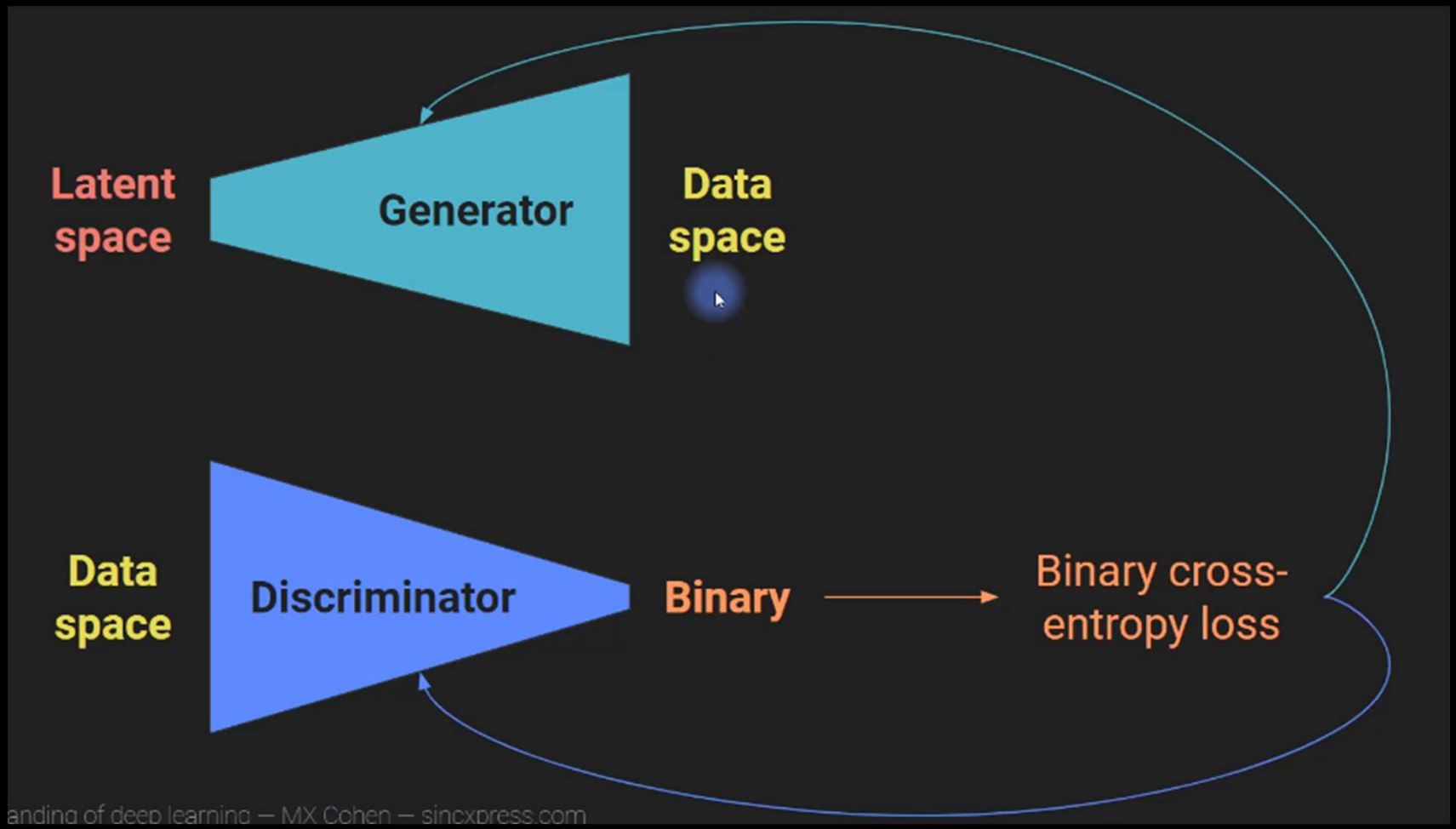


Generative Adversarial Network (GAN)



Adversarial Network / training
Discriminator model training (we give the model labelled data – real images)
Generator model starts from noise and corrects itself by backpropagation to come close to real sample

Generative Adversarial Network (GAN)



Applications of GAN

Combine two categories to increase training set.

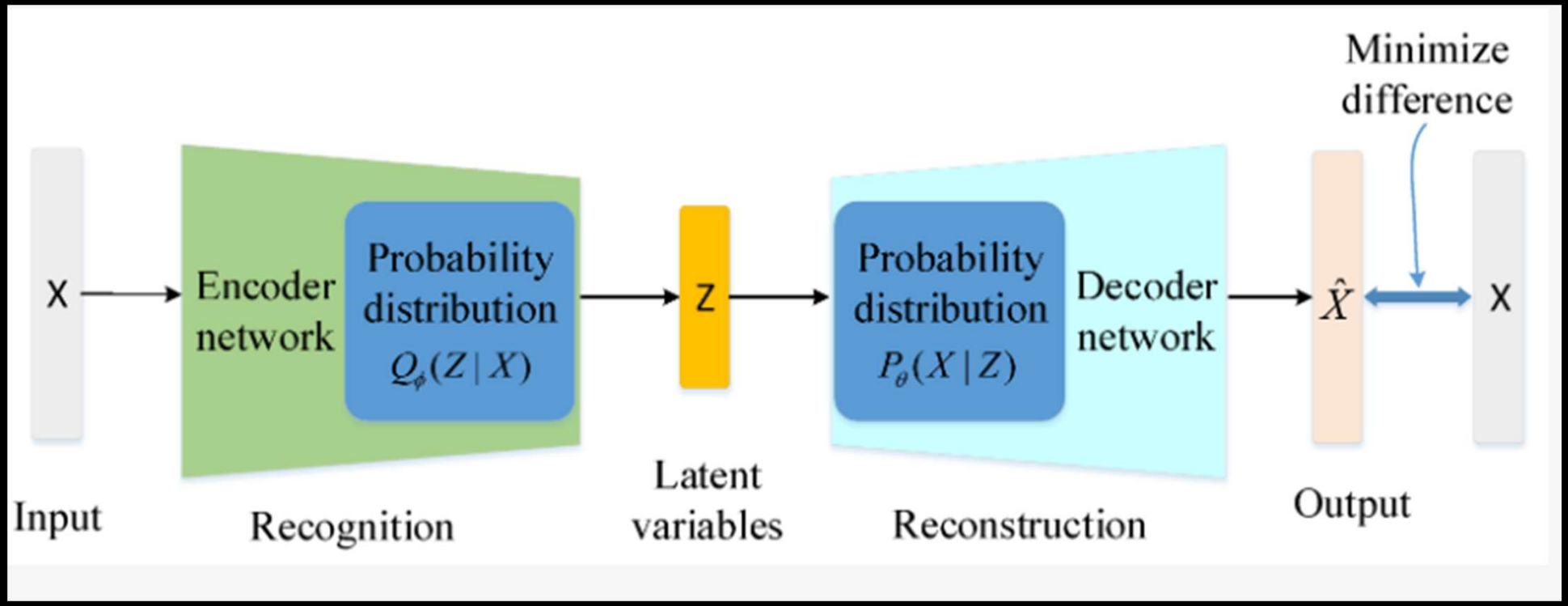
Super-resolution upscaling.

Personalized medical devices.

Evil (e.g., deep fakes).

Learning from private data without privacy concerns (learn from data, then delete original data).

Variational Auto Encoder (VAE)



GAN Vs VAE

Feature	VAE (Variational Autoencoder)	GAN (Generative Adversarial Network)
Architecture	Encoder-Decoder	Generator-Discriminator (Adversarial)
Learning Approach	Probabilistic (learns latent space distribution)	Game-theoretic (two networks compete)
Latent Space Representation	Continuous, structured, and smooth	Unstructured, often chaotic
Training Objective	Maximizes log-likelihood with KL-divergence regularization	Minimax game (Generator vs. Discriminator)
Optimization	Uses reconstruction loss + KL divergence	Uses adversarial loss (binary cross-entropy)
Sample Quality	Lower quality, but smooth interpolation	High quality, but may have mode collapse
Stability of Training	More stable, but blurry images	Less stable, often difficult to train
Mode Collapse Issue	❌ No mode collapse	✅ Can suffer from mode collapse (generator produces limited variety)

VAE Demo and assignment

1. **Input Image:**

   $$(28, 28, 1) \quad \text{(grayscale image)}$$

2. **Encoder Output:**
   - Mean (μ): $[1.0, -0.5]$
   - Log Variance ($\log(\sigma^2)$): $[0.1, 0.2]$

3. **Latent Vector (Sampled):** Stochastic encoding

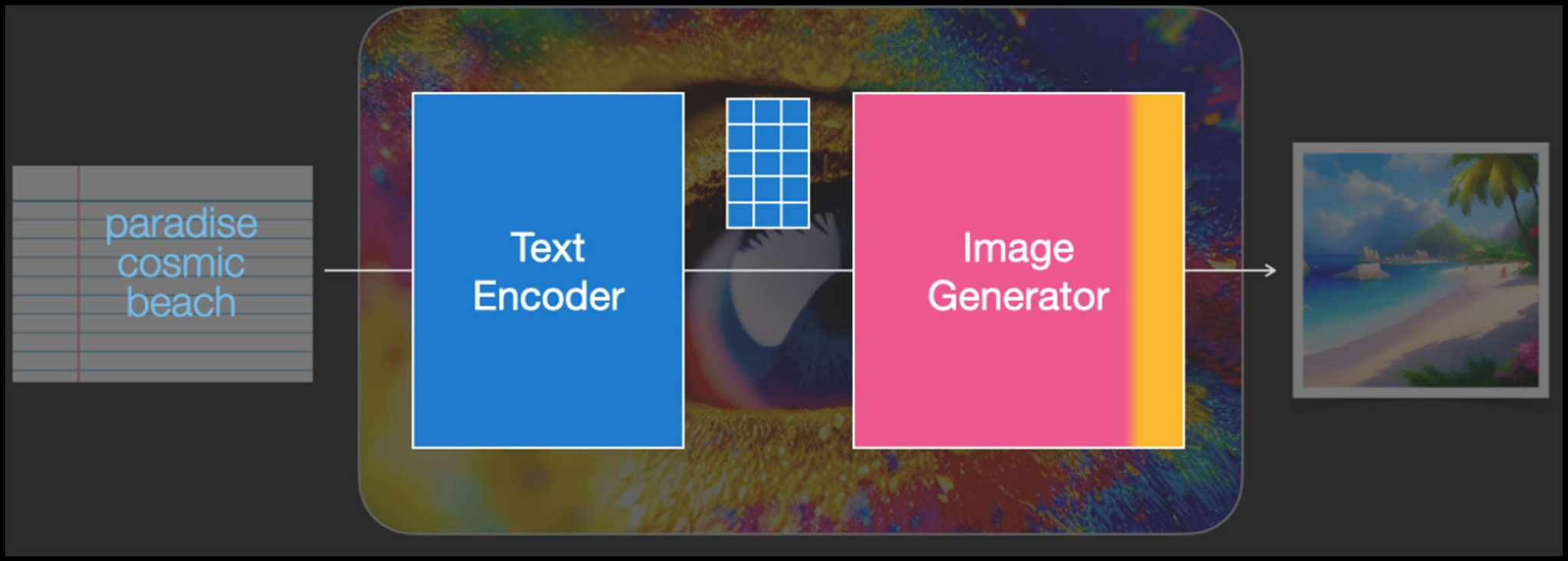
   $$z = \mu + \sigma \cdot \epsilon$$

   Example: $[1.2, -0.6]$

4. **Decoder Output:**

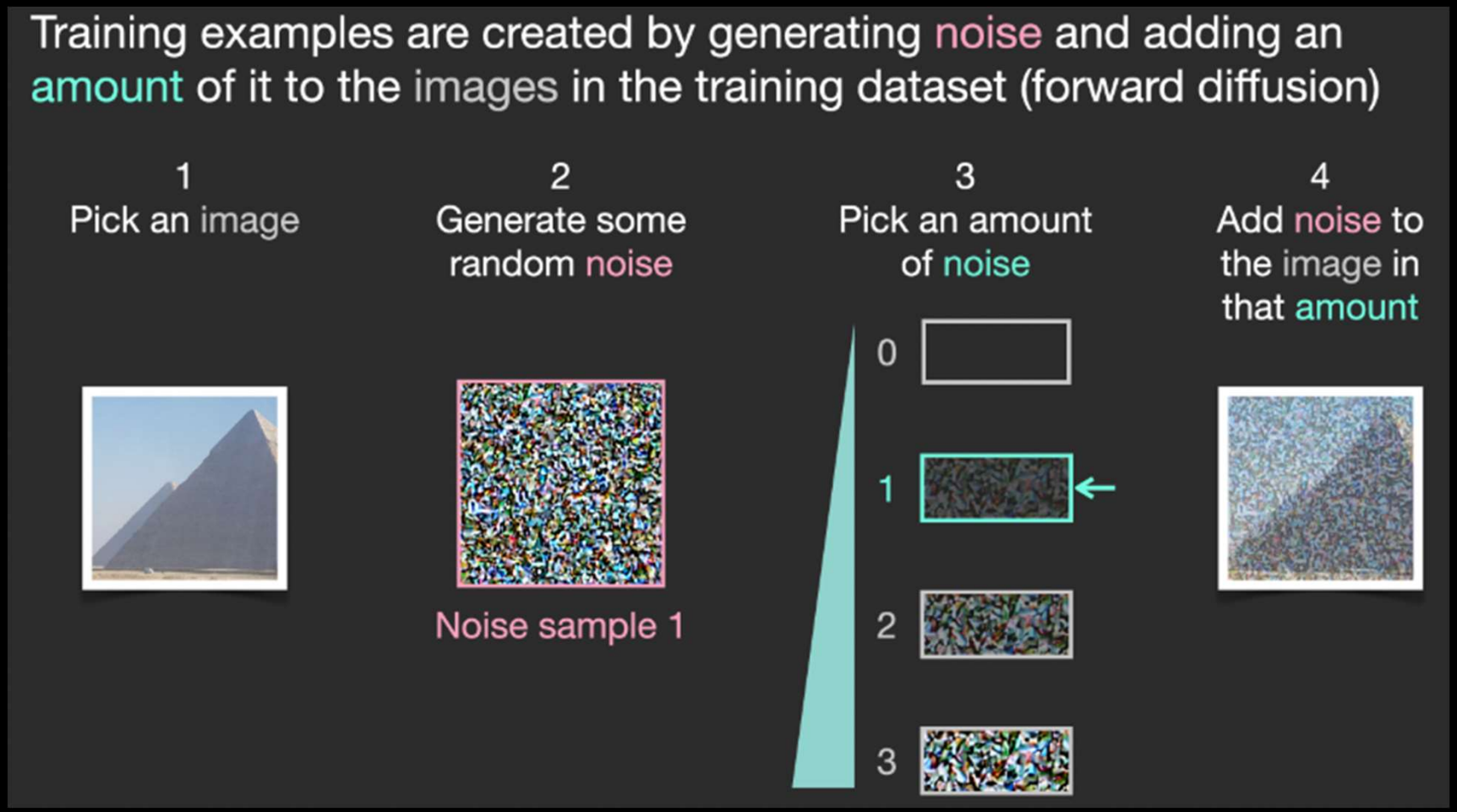
   Reconstructed image of shape `(28, 28, 1)`.

Stable Diffusion



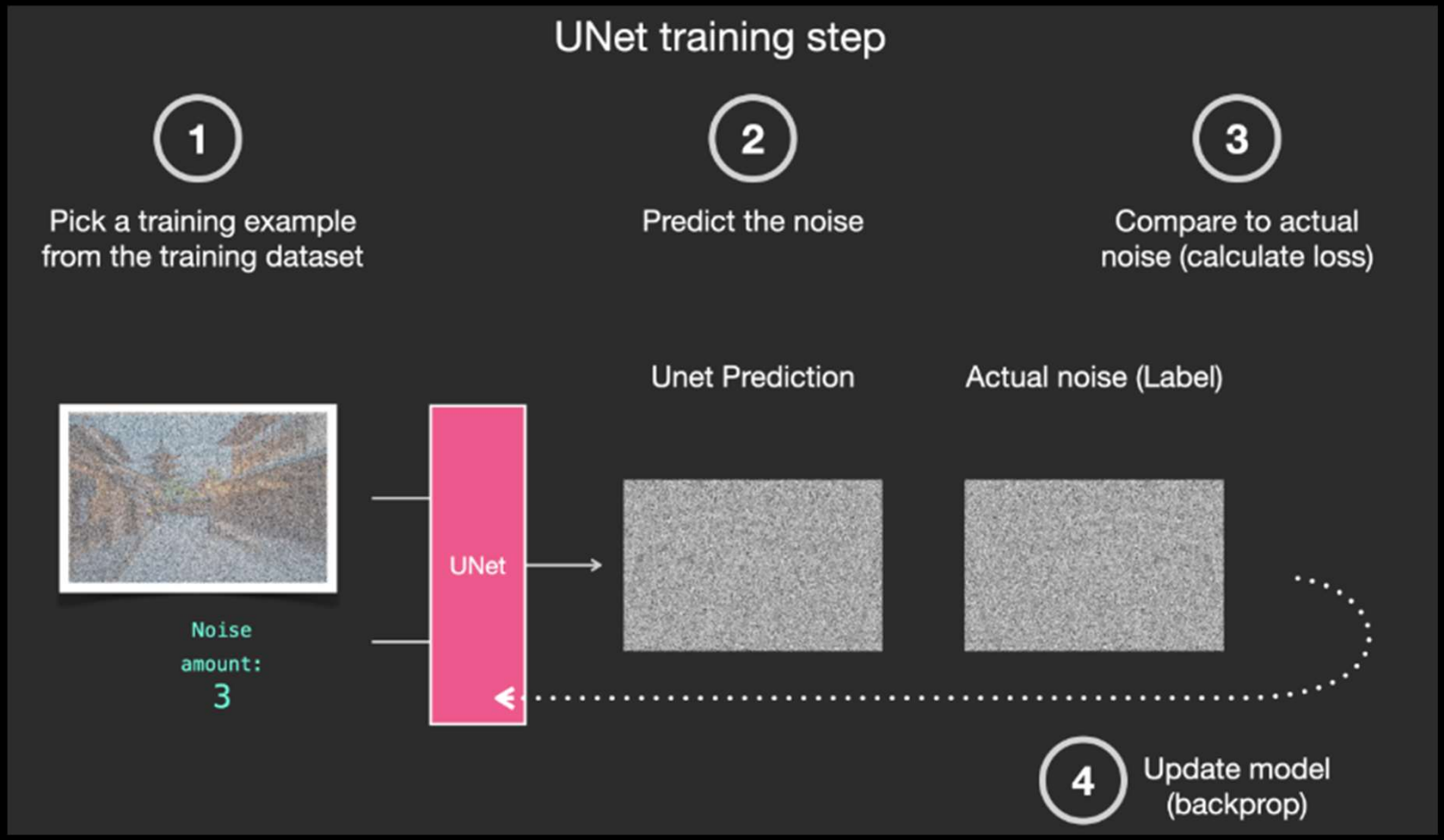
https://jalammar.github.io/illustrated-stable-diffusion/

Stable Diffusion training dataset creation



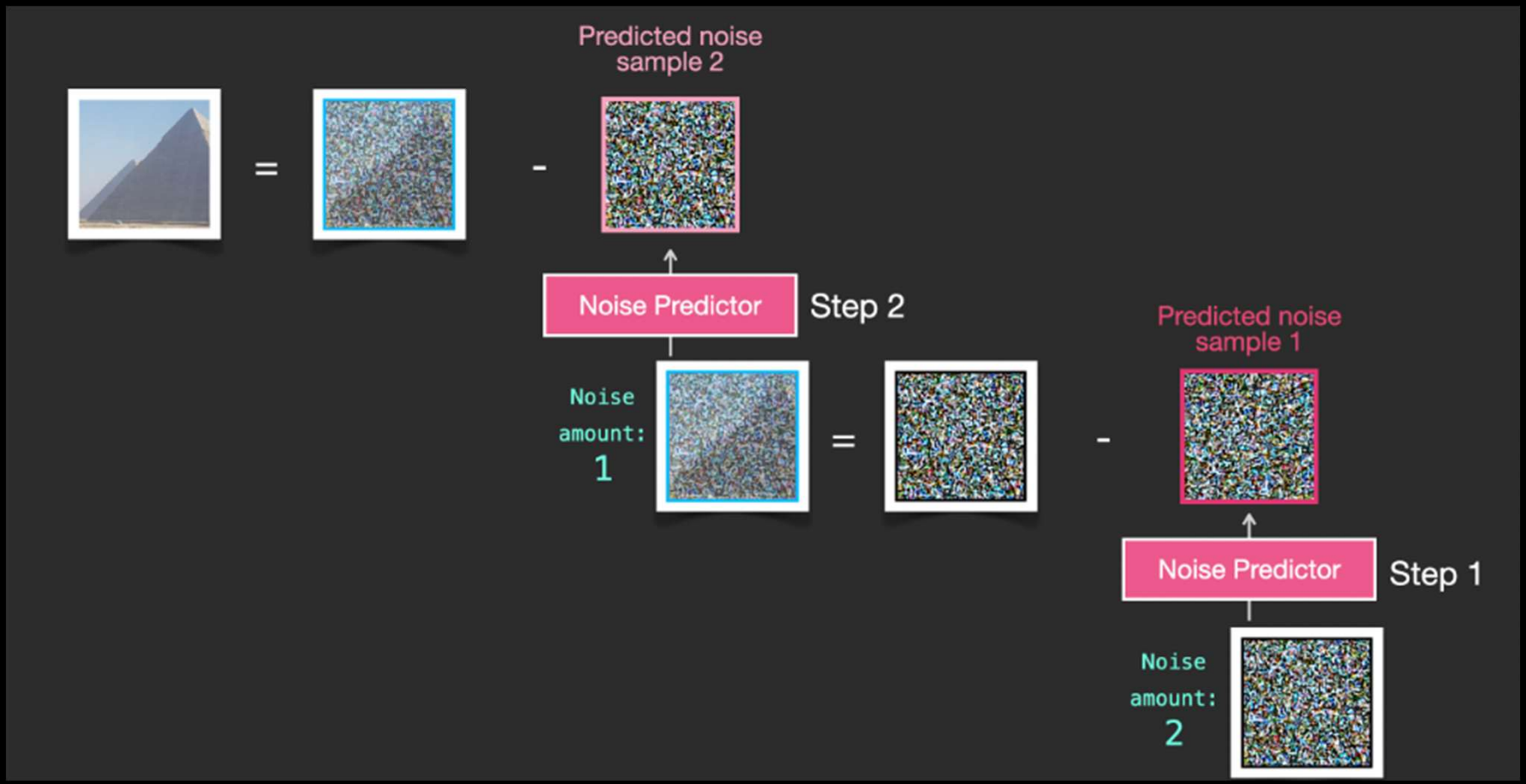
https://jalammar.github.io/illustrated-stable-diffusion/

Stable Diffusion training



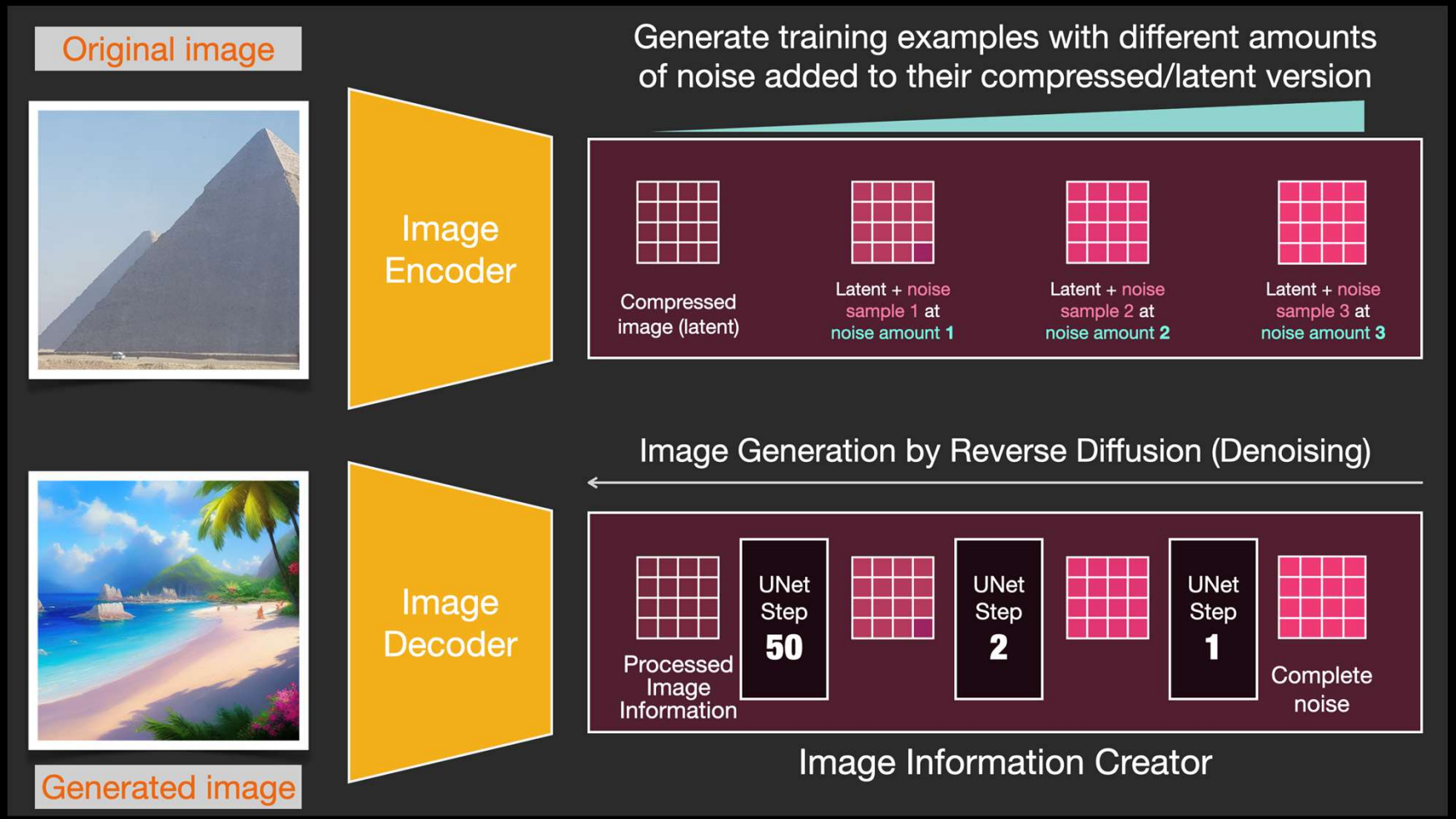
https://jalammar.github.io/illustrated-stable-diffusion/

Stable Diffusion – image generation by de-noising



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

Stable Diffusion – image generation by de-noising



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

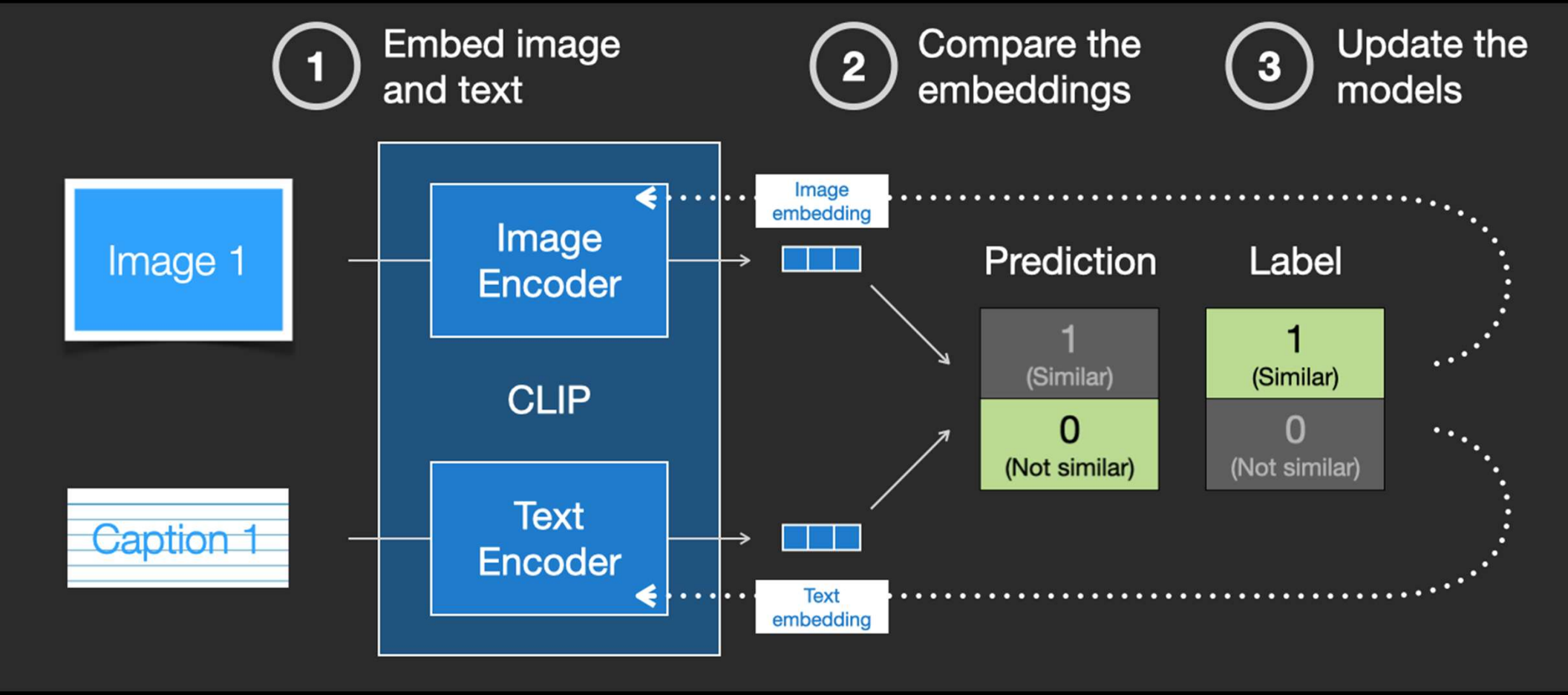
Stable Diffusion – leveraging CLIP

Contrastive Language Image Pre-training (CLIP) is trained on a dataset of images and their captions (~400 million images and their captions)

IMAGE			
CAPTION	Photo pour Japanese pagoda and old house in Kyoto at twilight - image libre de droit	Soaring by Peter Eades	far cry 4 concept art is the reason why it 39 s a beautiful game vg247. Black Bedroom Furniture Sets. Home Design Ideas

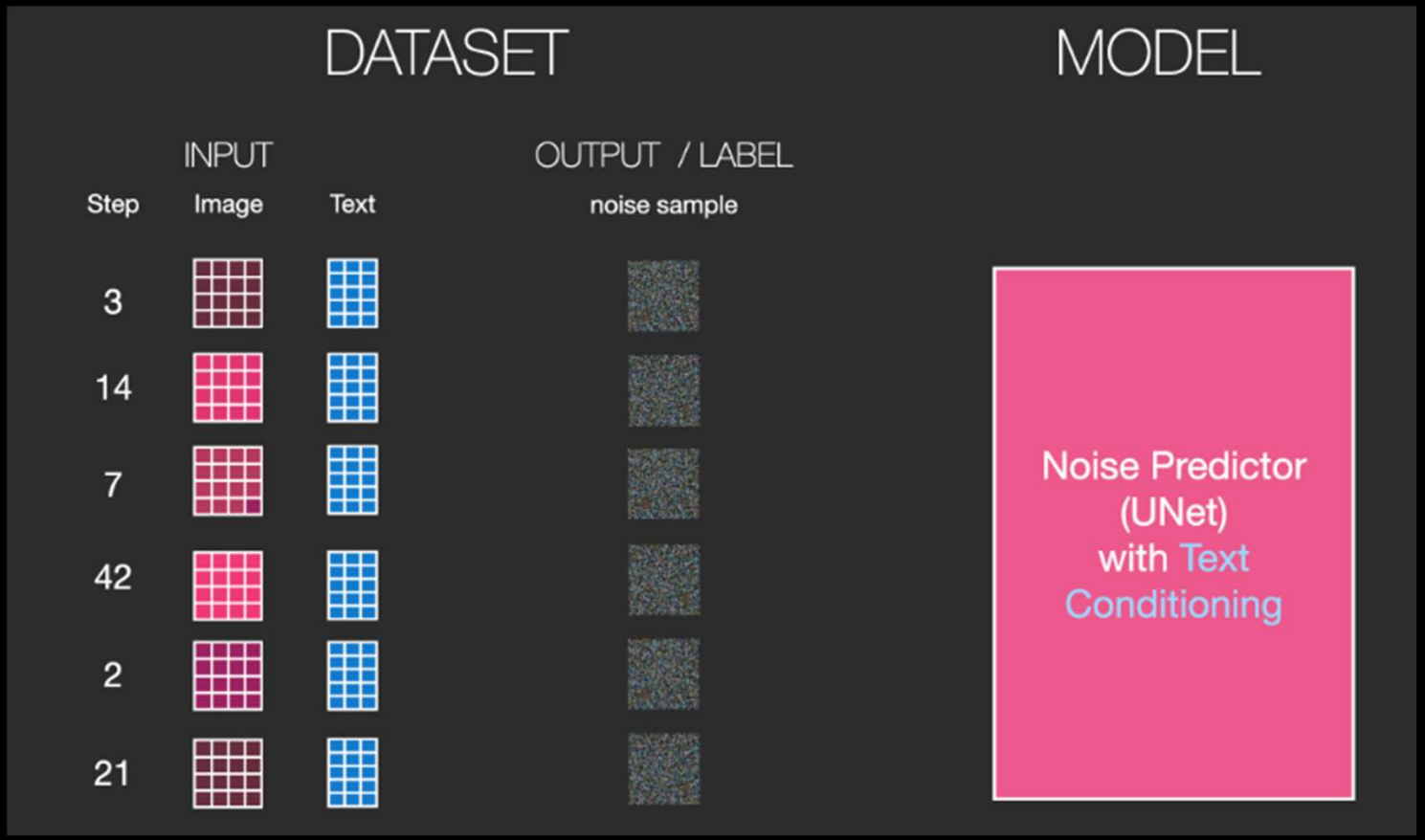
https://jalammar.github.io/illustrated-stable-diffusion/

Stable Diffusion – adding text embeddings encoder



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

Stable Diffusion – adding text embeddings encoder



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

Stable Diffusion demo and assignment

Speech to Text Models

Open AI **Whisper**
Uses a Transformer-based encoder-decoder architecture trained on a large dataset of diverse audio.

Google Speech to Text
Google's cloud service for speech recognition.
Based on large-scale neural network models (originally LSTM/RNN-based, now moving towards Transformer-based architectures).

Commercial services:
Amazon Transcribe, Microsoft Azure Speech, Apple Siri

https://huggingface.co/spaces/openai/whisper

Text to speech models – how do they work?

Text Analysis / Front-End
The input text is normalized (expanding abbreviations, handling numbers, punctuation),

Acoustic Model
A neural network (e.g., Tacotron, FastSpeech, or VITS) predicts an intermediate acoustic representation—commonly a mel-spectrogram (similar to embeddings creation)

Vocoder
A separate model (e.g., WaveNet, HiFi-GAN, WaveRNN) that converts the spectrogram to raw audio waveforms.

Post-Processing
The generated audio can be enhanced or normalized (e.g., for loudness), voice style transfer or emotional prosody control, which modifies rhythm and intonation for different contexts.

Text to speech models

- Tacotron (encoder–decoder approach) +vocoder
- Vocoder (e.g., WaveGlow, Deepmind Wavenet)
  transforms the “predicted” acoustic features into the actual waveform you hear.
  Wavenet is one of initial vocoders, trained on audio data, can synthesize highly realistic speech.
- VITS (Variational Inference TTS)
- Commercial services
  Amazon Polly, Microsoft Azure TTS, Google Cloud TTS

Multi modal models

Multimodal input refers to the ability of AI systems to accept and process and generate different types of data simultaneously, such as text, images, audio, and video.

Examples:

GPT-4, Gemini, DALL-E, Stable diffusion

Comparing the image generation architectures (1/3)

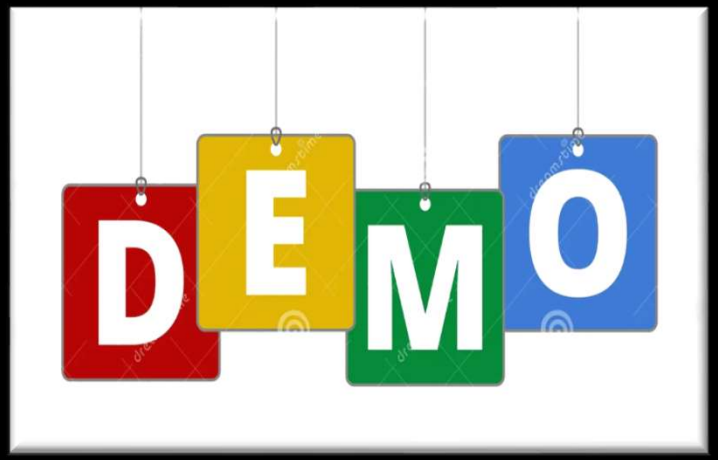
Feature	GAN (Generative Adversarial Network)	VAE (Variational Autoencoder)	Stable Diffusion
Architecture	Generator and Discriminator in adversarial setup	Encoder-Decoder with probabilistic latent space	U-Net with noise diffusion in a latent space
Working Principle	Generator creates images; Discriminator judges real vs fake	Encodes input into a latent distribution and reconstructs it	Iterative denoising from noise to image
Training Complexity	High; adversarial training can be unstable	Moderate; stable but can lead to blurry outputs	High; stable but computationally intensive
Inference Complexity	Low; single forward pass	Low; single forward pass	High; multiple denoising steps
Training Cost	High due to adversarial competition	Low to moderate	Very high due to iterative learning
Inference Cost	Low	Low	High; slower due to iterative refinement

Comparing the image generation architectures (2/3)

Feature	GAN (Generative Adversarial Network)	VAE (Variational Autoencoder)	Stable Diffusion
Output Quality	High, but prone to mode collapse	Moderate; often blurry	Very high; detailed, sharp, and diverse
Diversity of Outputs	Limited due to mode collapse	Moderate	High; better at generating unique and varied results
Latent Space	Unstructured and harder to navigate	Smooth and continuous	Compressed, structured, and interpretable
Robustness	Low; sensitive to hyperparameters	High	High; consistent across conditions
Main Use Cases	Image generation, face generation, deepfakes	Anomaly detection, data compression, reconstruction	Text-to-image, image editing (inpainting, outpainting), art generation
Conditional Generation	Challenging to implement	Limited	Built-in; excels in text-to-image generation

Comparing the image generation architectures (3/3)

Feature	GAN (Generative Adversarial Network)	VAE (Variational Autoencoder)	Stable Diffusion
Mode Collapse Risk	High	Low	Very low
Training Time	Long	Short to moderate	Very long
Interpretability	Low	High	High
Resolution Support	High, but less effective for very large images	Low to moderate	High; built for high-resolution generation
Realism of Output	High, photorealistic	Moderate; less sharp	Very high; fine details and textures
Noise Tolerance	Low	Moderate	High; denoising is part of the process
Deployment Ease	Easy after training	Easy	Complex due to high inference cost
Energy Efficiency	Moderate	High	Low; requires significant GPU resources



SORA DEMO:

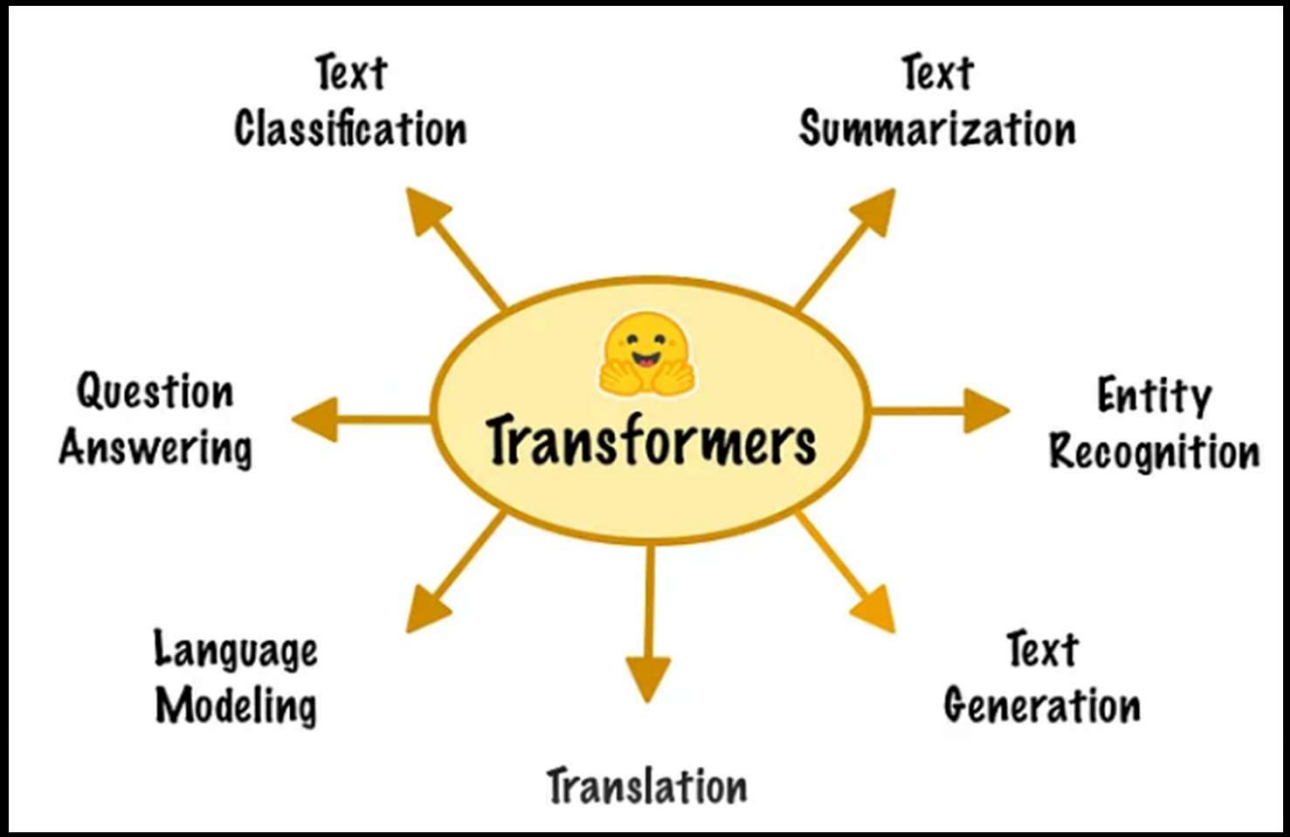
https://openai.com/sora

Figure 01 Robot demo

https://www.youtube.com/watch?v=Sq1QZB5baNw

Generative Models Quiz

Core Capability: NLP



Core capability - inferencing

Inferencing involves generating outputs based on **patterns** and **knowledge** learned during their **training**. This can include answering questions, completing sentences, translating text, and more

Prompt: "What is the capital of France?"
Response: "Paris."

Prompt: A detailed article about climate change.
Response: "Climate change is a significant global issue caused by human activities leading to rising temperatures and environmental impacts."

Translation: Translating text from one language to another.
Prompt: "Hello, how are you?" (English)
Response: "Hola, ¿cómo estás?" (Spanish)

Text Generation: Producing creative content such as stories, poems, or essays.
Prompt: "Once upon a time in a faraway land,"
Response: "there was a small village surrounded by towering mountains and lush forests."

Sentiment Analysis: Determining the sentiment or emotional tone of a text.

Example: Input: "I love this product!"
Output: "Positive sentiment."

Core capability - summarization

It can summarize large documents, videos, images, flowcharts, audio – any modality, multiple languages

Extractive Summarization:

The LLM selects and concatenates key sentences or phrases from the original text.

Abstractive Summarization:

The LLM generates new sentences that paraphrase the original content.

Practical applications
News summarization, scientific papers summarization, legal documents, product reviews

Challenges and limitations:
May miss critical details or emphasize less important information affecting comprehensiveness
Coherence and relevance: While generally good, can be lacking sometimes
Bias and errors: Biases present in training data can result in skewed or inaccurate summaries

Core capability - reasoning

Reasoning in AI involves higher-level cognitive processes (compared to inferencing) where the AI system makes decisions based on logic, understanding relationships, and drawing conclusions from given information.

Deductive reasoning:

All humans are mortal. Socrates is a human. Therefore, Socrates is mortal.

Inductive reasoning

Every time I water my plant, it grows taller. Watering plants regularly helps them grow taller.

Abductive reasoning

Ability to choose best possible inference. Lawn is wet – rain? Or sprinkler?

Practical applications

Analyse market data to analyse patterns, customer behaviour analysis, drug discovery

Core capability – generate **new, unseen** outputs

Generate **new, unseen** text, image, audio, video