EE7207 Week 10

Self-Supervised Learning

What is self-supervised learning

- Self-supervised learning is a machine learning technique where a model trains itself to learn from unlabeled data by generating labels automatically.
- This approach transforms unsupervised problems into supervised ones by auto-generating labels, allowing the model to learn from vast amounts of unlabeled data.

Why do we need self-supervised learning

High Cost of Labelled Data

Traditional learning methods heavily rely on labeled data, which is expensive and time-consuming to obtain.

Lengthy Data Preparation Lifecycle

The process of preparing labeled data for machine learning models involves cleaning, filtering, annotating, and restructuring data, making it a lengthy process.

Self-supervised learning addresses these challenges by allowing models to learn complex patterns from unlabeled data efficiently, making it a valuable technique in machine learning for various applications like computer vision and natural language processing

Two types of self-supervised learning

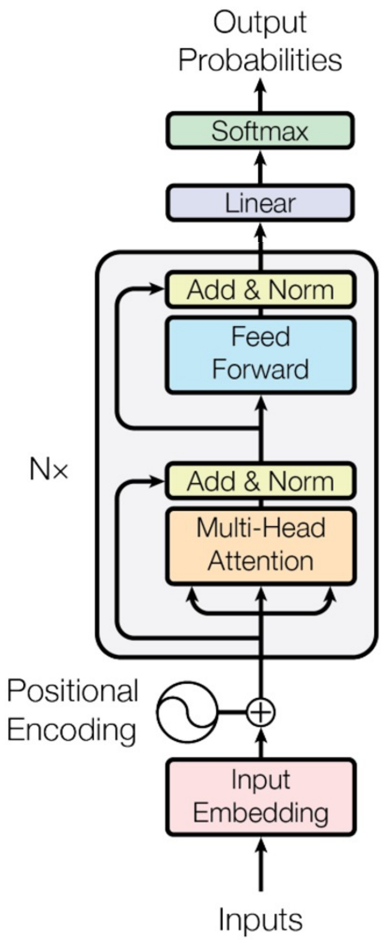
Discriminative Modeling

- Discriminative models focus on learning the decision boundary between classes, aiming to distinguish different categories or classes of data.
- **Strength:**
  - Effective at distinguishing between classes or categories.
  - Can achieve better performance with less data for classification tasks.
- **Weakness:**
  - Do not provide insights into the underlying data distribution.
  - Cannot generate new samples from the learned distribution.
- BERT

Generative Modeling

- Generative models aim to learn the underlying probability distribution of the data to generate new samples that resemble the training data. Suitable for tasks like text generation and image synthesis.
- **Strength:**
  - Can generate new samples resembling the training data.
  - Provide insights into the underlying structure and distribution of the data.
  - Useful for tasks beyond classification like data generation and imputing missing data.
- **Weakness:**
  - Typically have more parameters and are computationally expensive.
  - Might require more data to converge to a meaningful model.
- **GPT**

BERT – Bidirectional Encoder Representation from Transformers



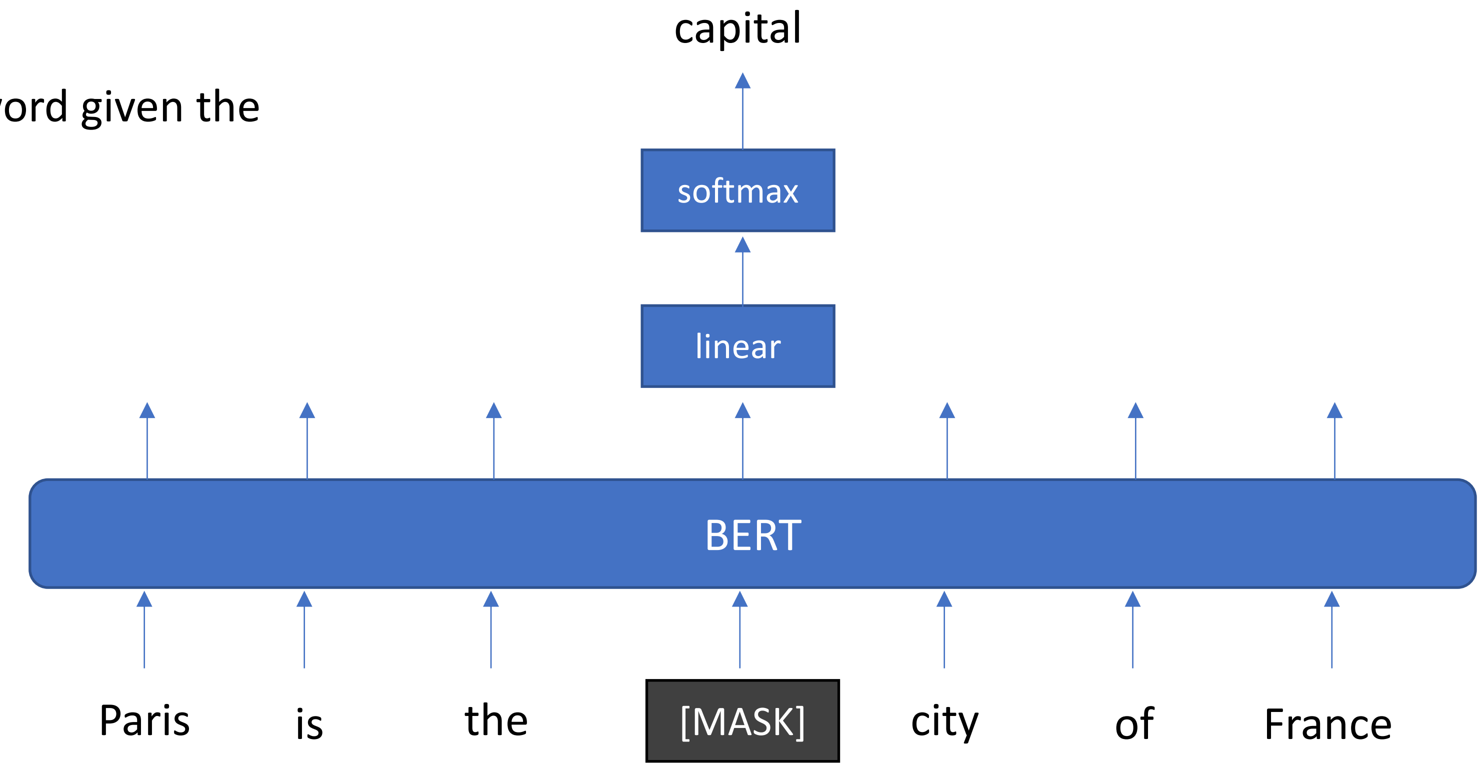
Encoders-only: BERT is made up of layers of encoders of the Transformers model

- $\text{BERT}_{\text{BASE}}$
  - 12 encoder layers
  - 12 attention heads
  - 110M parameters

- $\text{BERT}_{\text{LARGE}}$
  - 24 encoder layers
  - 16 attention heads
  - 340M parameters

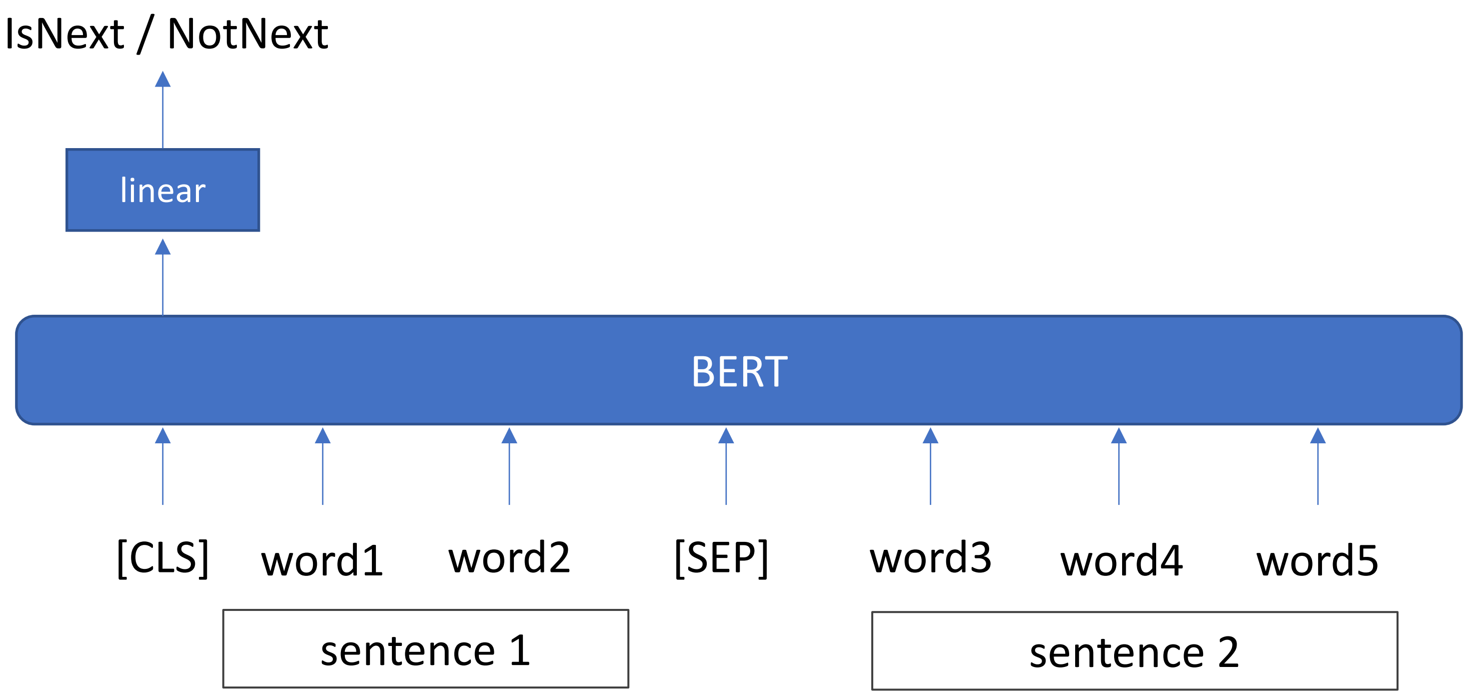
Training task 1: Masked Language Model

- Randomly select words in a sentence and replaced by [MASK].
- BERT is asked to predict the masked word given the left and right context.



Training task 2: Next Sentence Prediction

- Learning relationships between sentences
- 50% of the time, select the actual next sentence
- 50% of the time, select a random sentence from the text



How to use BERT to solve problems

Pretrain

BERT

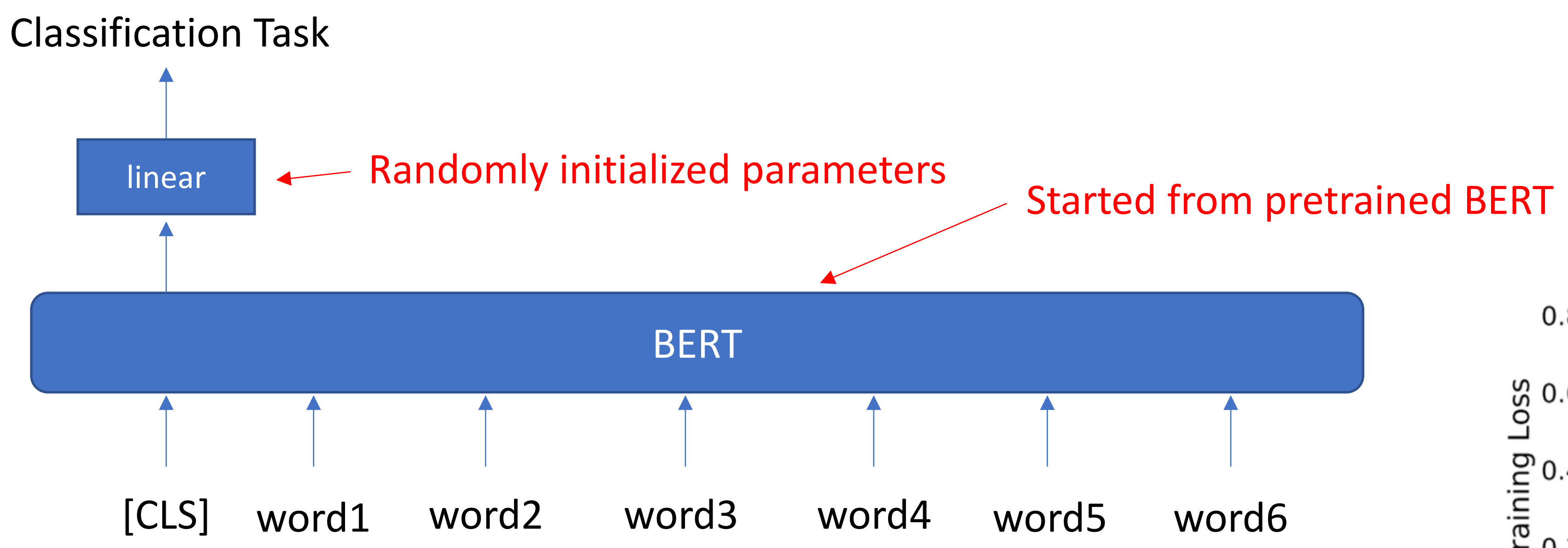
- Masked Word Prediction
- Next Sentence Prediction

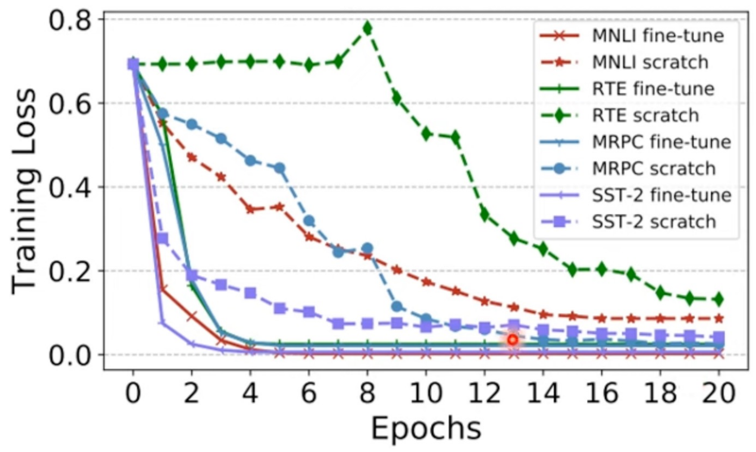
Finetune

Task of interest

- Can achieve better performance with less labelled data

How to finetune





Source of image: https://arxiv.org/abs/1908.05620

Which tasks is BERT good at

Tasks requiring a deep understanding of bidirectional context

- Sentiment analysis
- Text classification
- Neural machine translation
- Named entity recognition (NER)
- Question answering

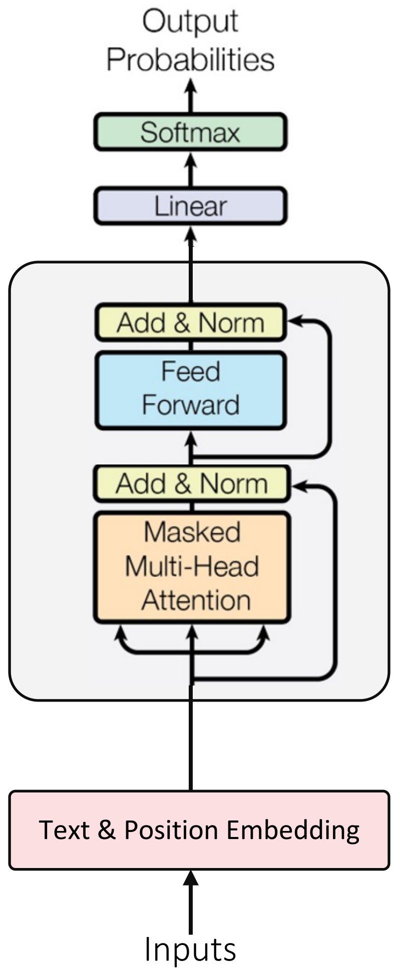
Potential issue with finetuning

- Still highly depending on labelled data
- Usually perform badly on related tasks not directly finetuned on

GPT is very ambitious to try to avoid the necessity for fine-tuning on each specific task and leverage on prompt engineering and zero-shot learning

- prompt engineering: providing detailed and specific inputs to the model
- zero-shot learning: expecting models to perform tasks without explicit training on them

GPT – Generative Pre-trained Transformer



Decoders-only: GPT is made up of layers of decoders of the Transformers model, where the input data is directly fed into the decoder without prior transformation by an encoder

- GPT2
  - Released by OpenAI in 2019
  - 1.5 billion parameters
- GPT3
  - Released by OpenAI in 2020
  - 175 billion parameters
- GPT4
  - Released by OpenAI in 2023
  - Rumored to contain 1.76 trillion parameters

Training task: next token prediction

- Uni-directional
- Autoregressive
- Causal language model
- Look back at previous words to predict the next token
- Trained specifically for text generation

How to use GPT

- Handle specific tasks with **prompts**
- Zero-shot learning: perform a specific task given an instruction and input
  - Considered a very hard problem, even for humans

Prompt	Classify the text into positive, neutral or negative: Text: This movie is awesome! Classification:
Response	Positive

Few-shot learning

- One-shot learning: one example is included in the context
- Few-shot learning: multiple examples are included in the context

No parameters in GPT are updated

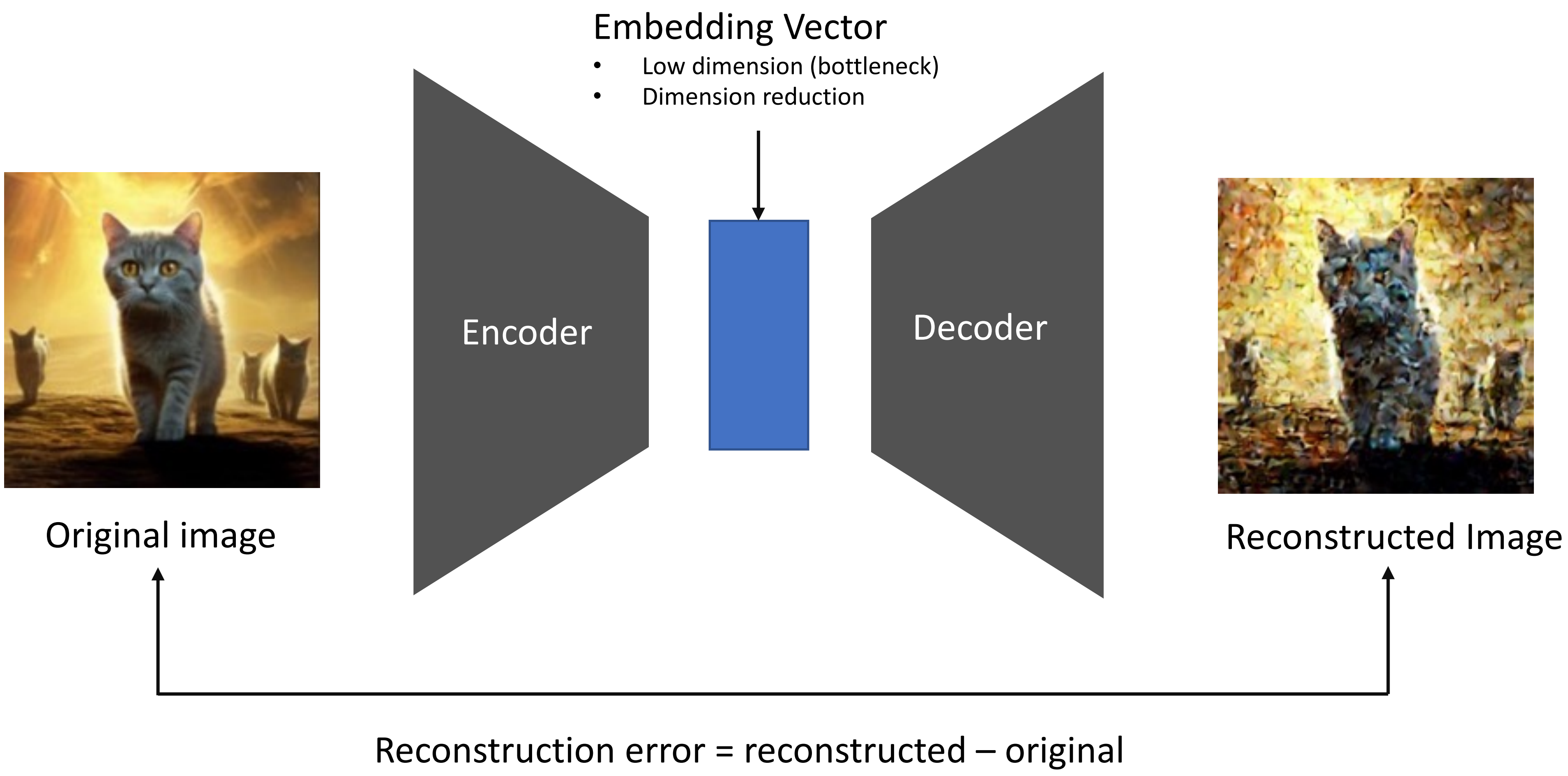
Prompt	Classify the text into positive, neutral or negative: Text: This movie is awesome! Classification: Positive Text: Lot of silly plot holes in the film. Classification: Negative Text: This movie was obscenely obvious and predictable. Classification: Negative Text: This movie has great style and fantastic visuals! Classification:
Response	Positive

Which tasks is GPT good at

Tasks requiring text generation and context-based responses, such as

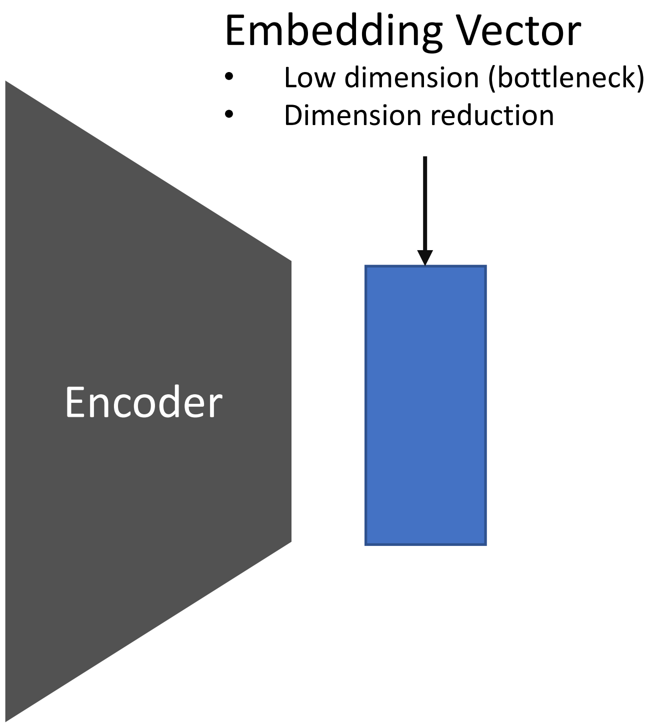
- Text generation, where the model generates coherent and contextually relevant text given a prompt or initial input.
- Dialogue systems and chatbots, where GPT generates responses based on the conversation history. GPT's ability to produce contextually relevant responses makes it suitable for creating engaging conversational agents.

Auto-encoder



How to use auto-encoder – Dimensionality reduction

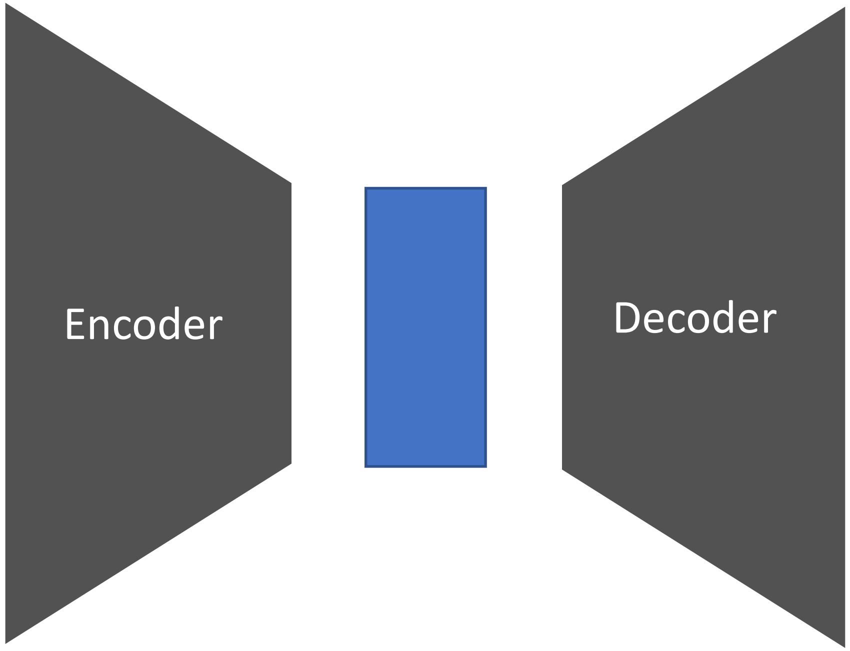
- Dimensionality reduction
- Representation learning / feature extractor
  - Categorical feature embedding
  - Numeric feature embedding



How to use auto-encoder – Anomaly detection

- Autoencoders are used to replicate the input dataset
- A reconstruction error is generated upon prediction
- Higher the reconstruction error, higher the possibility of the data point being an anomaly

Original image

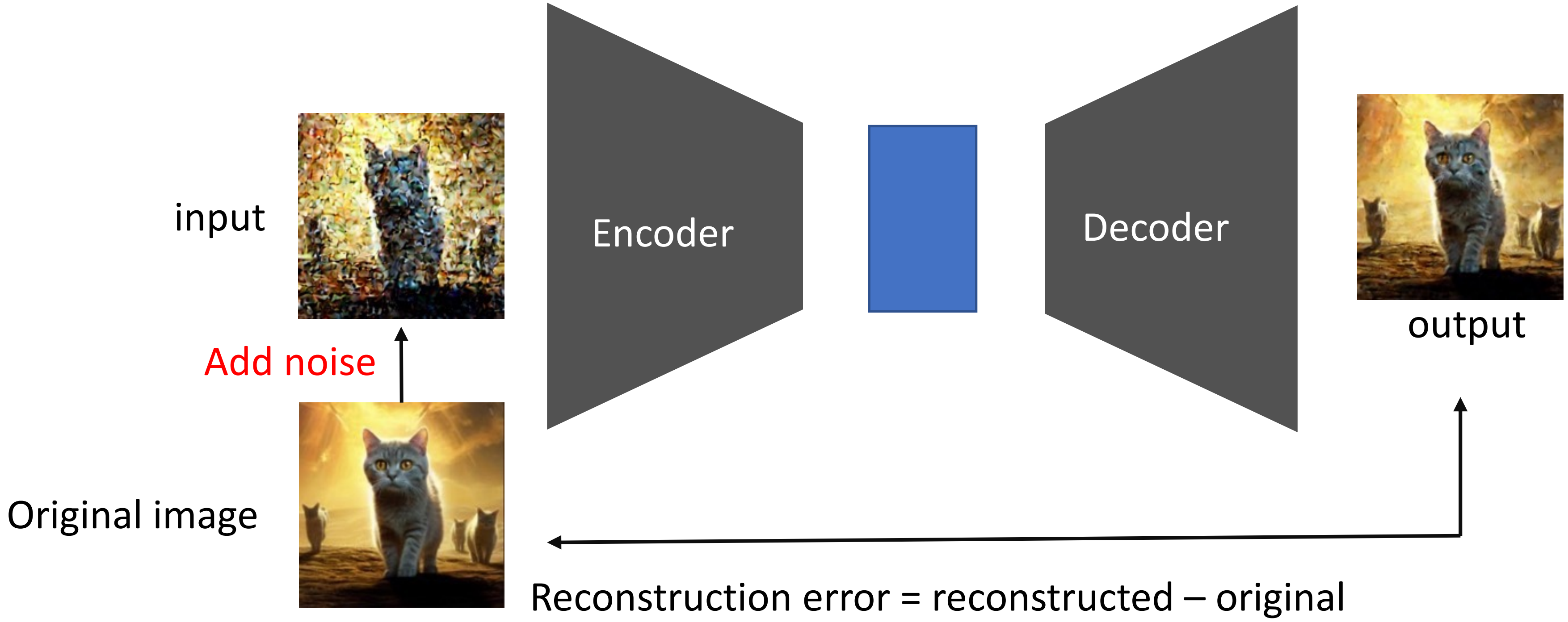


Reconstructed Image

Reconstruction error = reconstructed – original

Denoising autoencoder

- Sometimes autoencoder just memories inputs
- Denoising autoencoder
  - Introduced by Yan LeCun in 1987
  - Add noise to the input, train the autoencoder to reconstruct the input from a corrupted version
  - Why is it useful?
    - Force model not just to memories inputs
    - Extract the most important features
    - Learn a more robust representation of the data



Diffusion model is a type of autoencoder

side view, contemporary painting, fast braush works, photo style, pink hair, a tattooed happy woman with

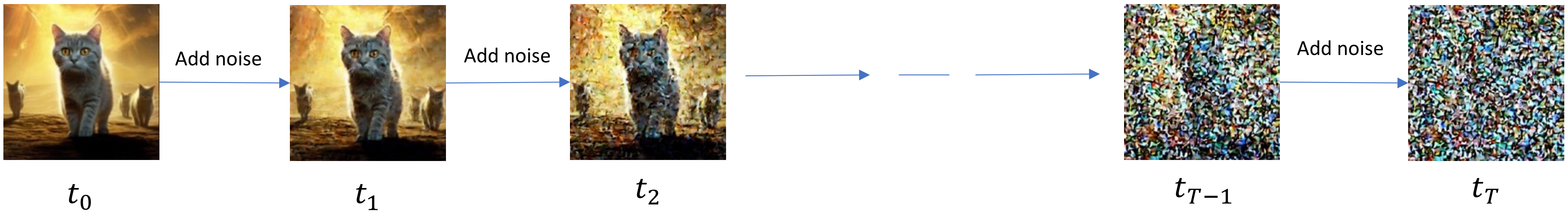
Photo of smallest cat Wearing a red coat and a white sweater are getting on the bus(standing and walking on both

Abstract ethereal sculpturework, a grandiose butterfly made from an aesthetically-pleasing arrangement of

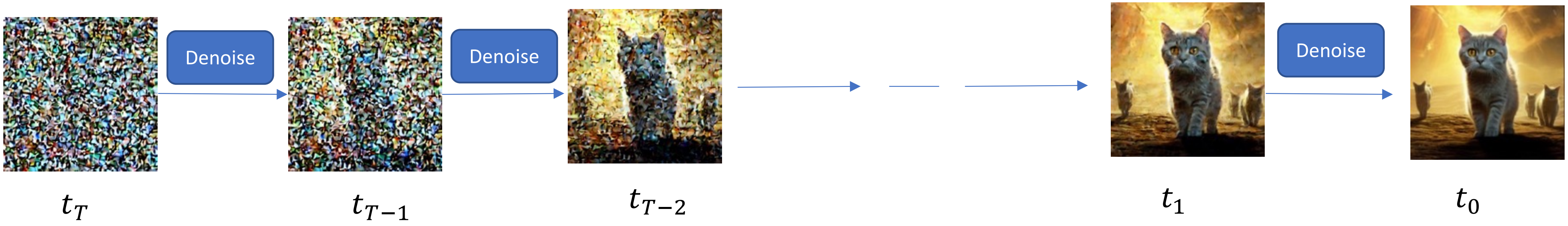
A traditional Chinese painting, portrait of a woman, background, waterfall in the middle, rolling mountains, three-distance method, Chinese calligraphy

Forward and Reverse Process

Forward Process (Add Noise)

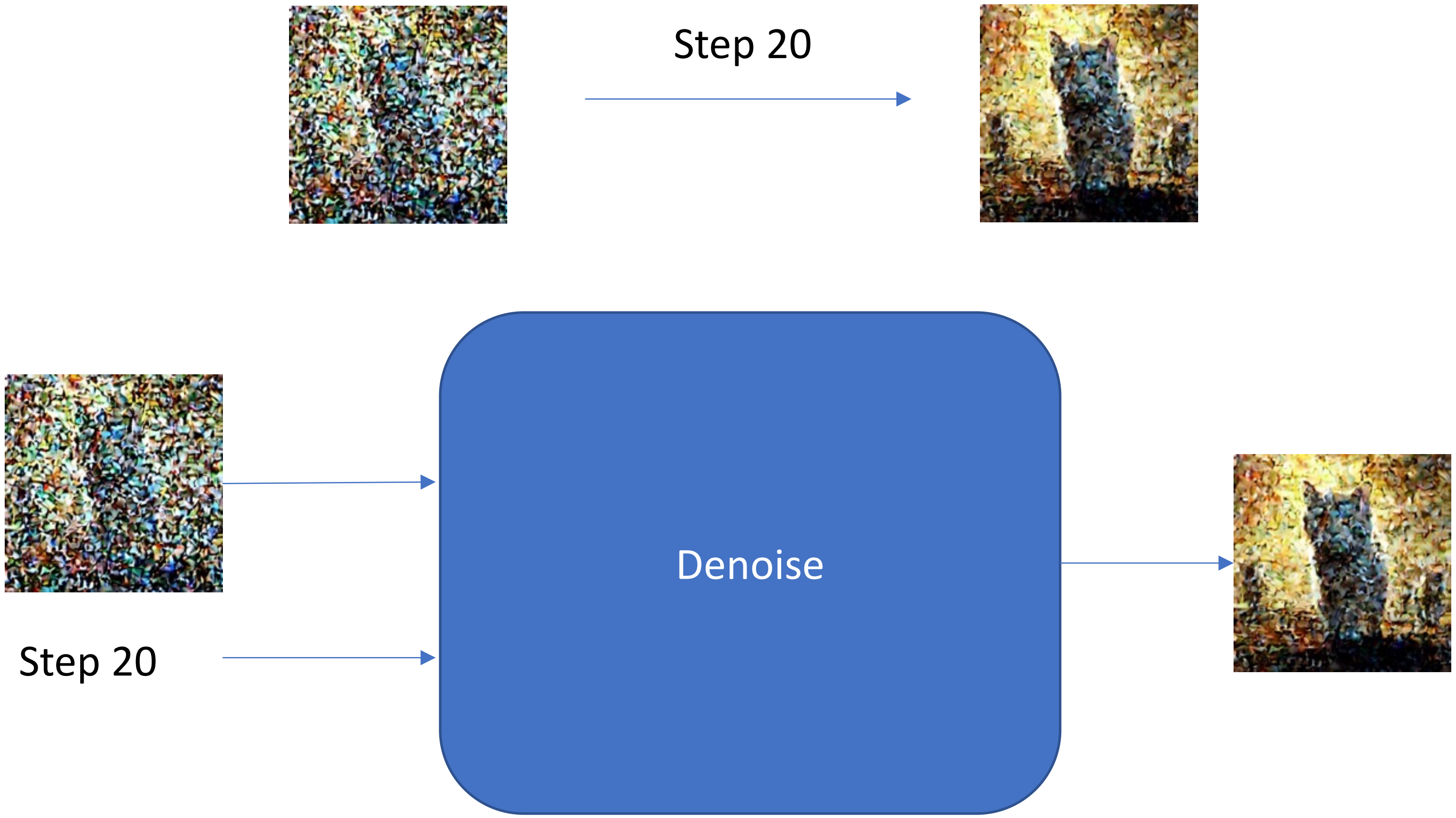


Reverse Process (Denoise)

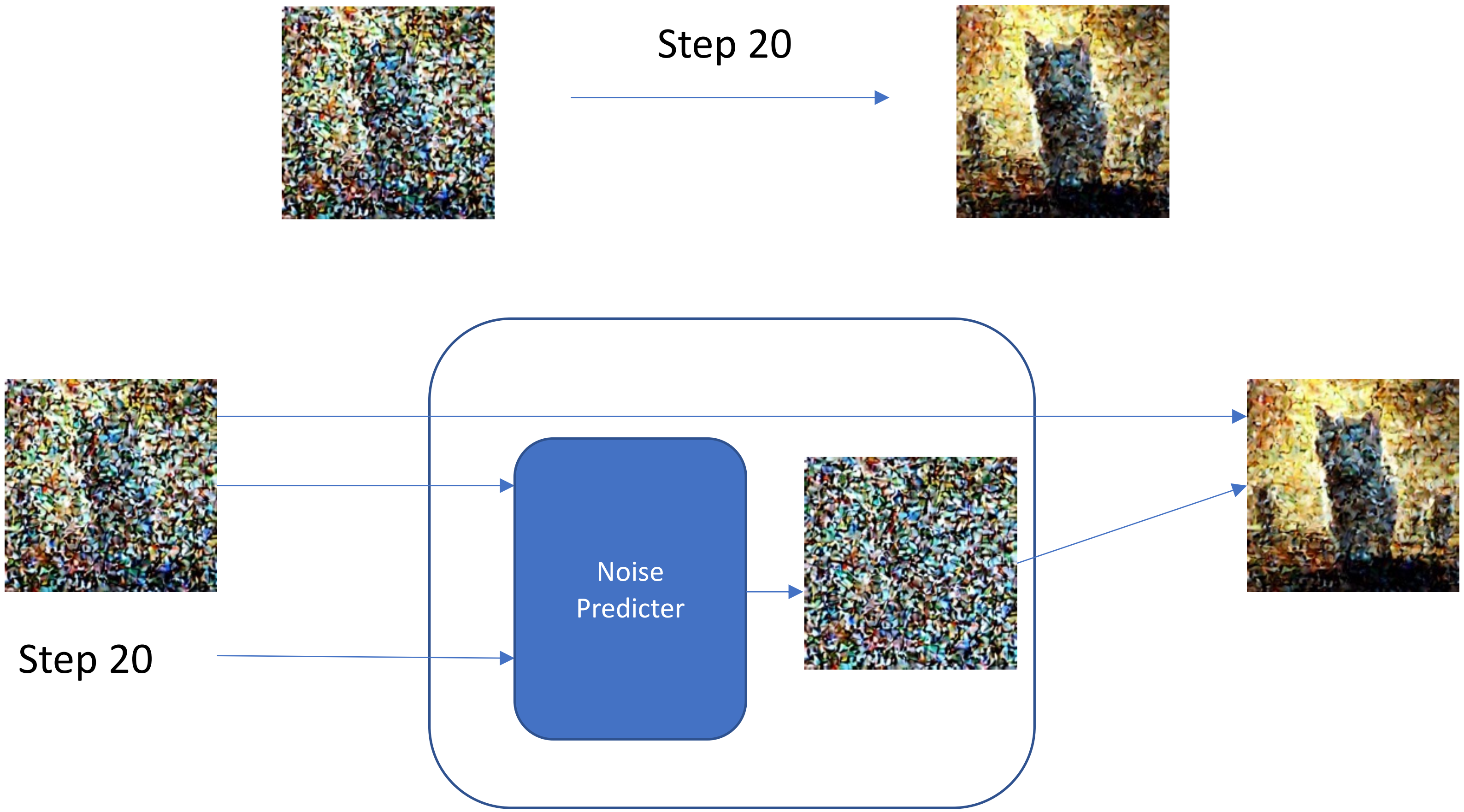


Another angle of looking at this: gradient descent on input images

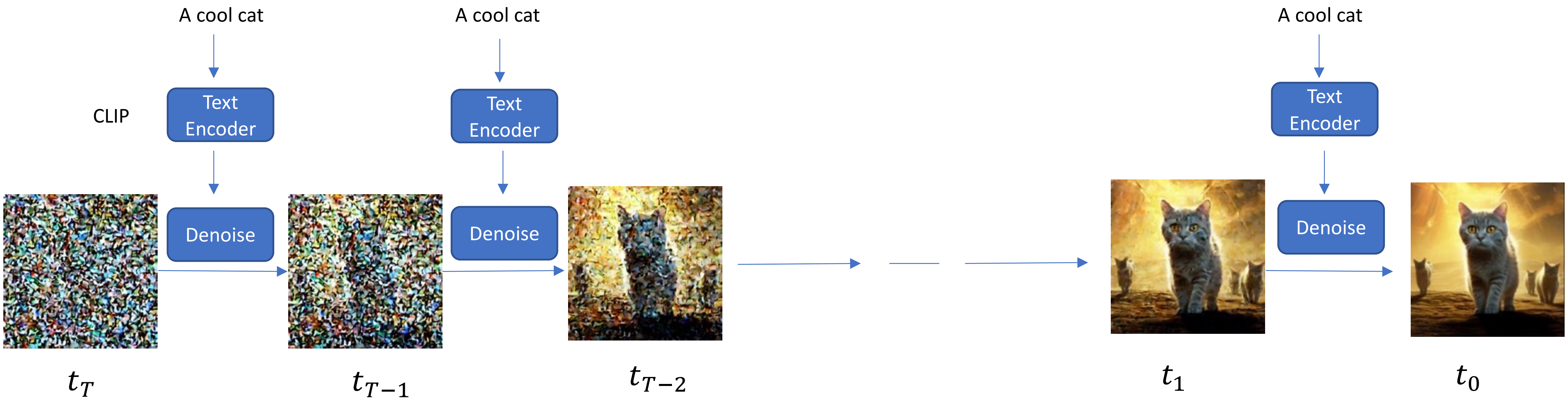
What does denoise module do



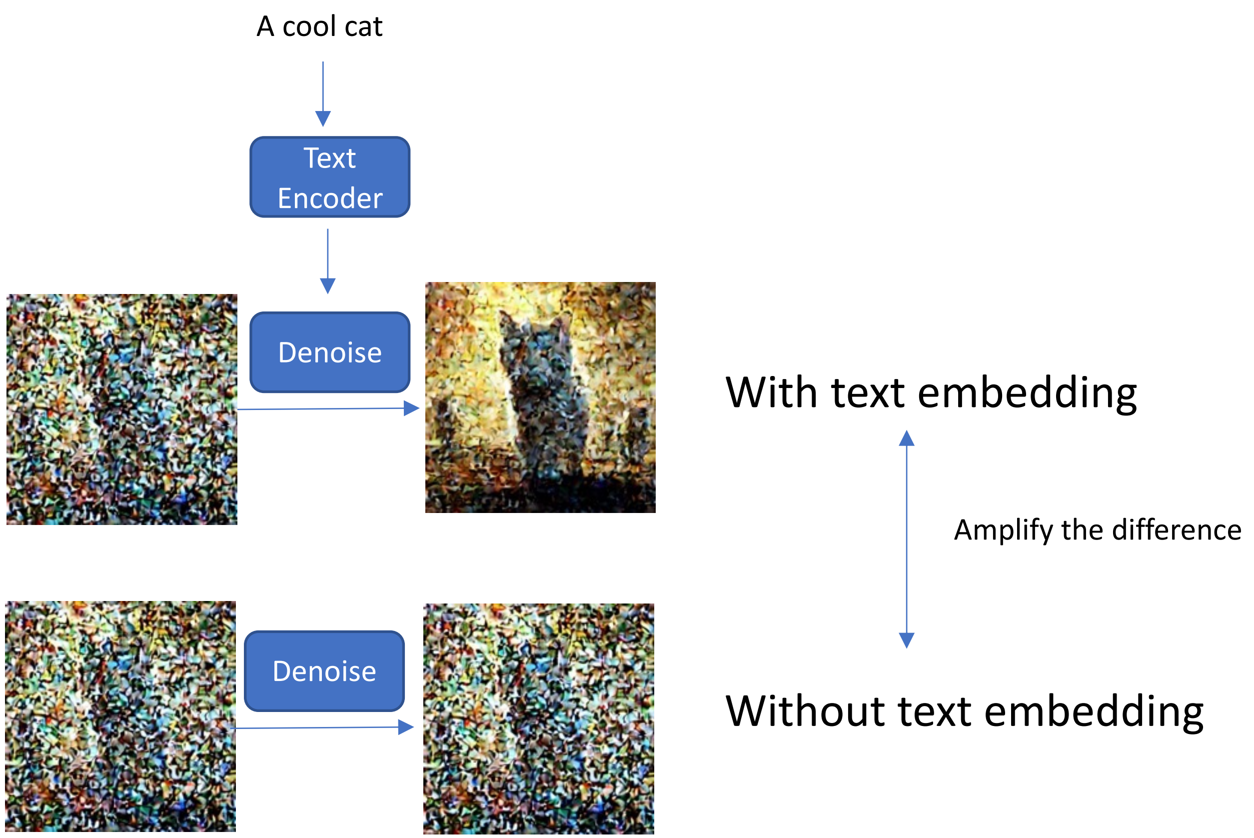
What does denoise module do



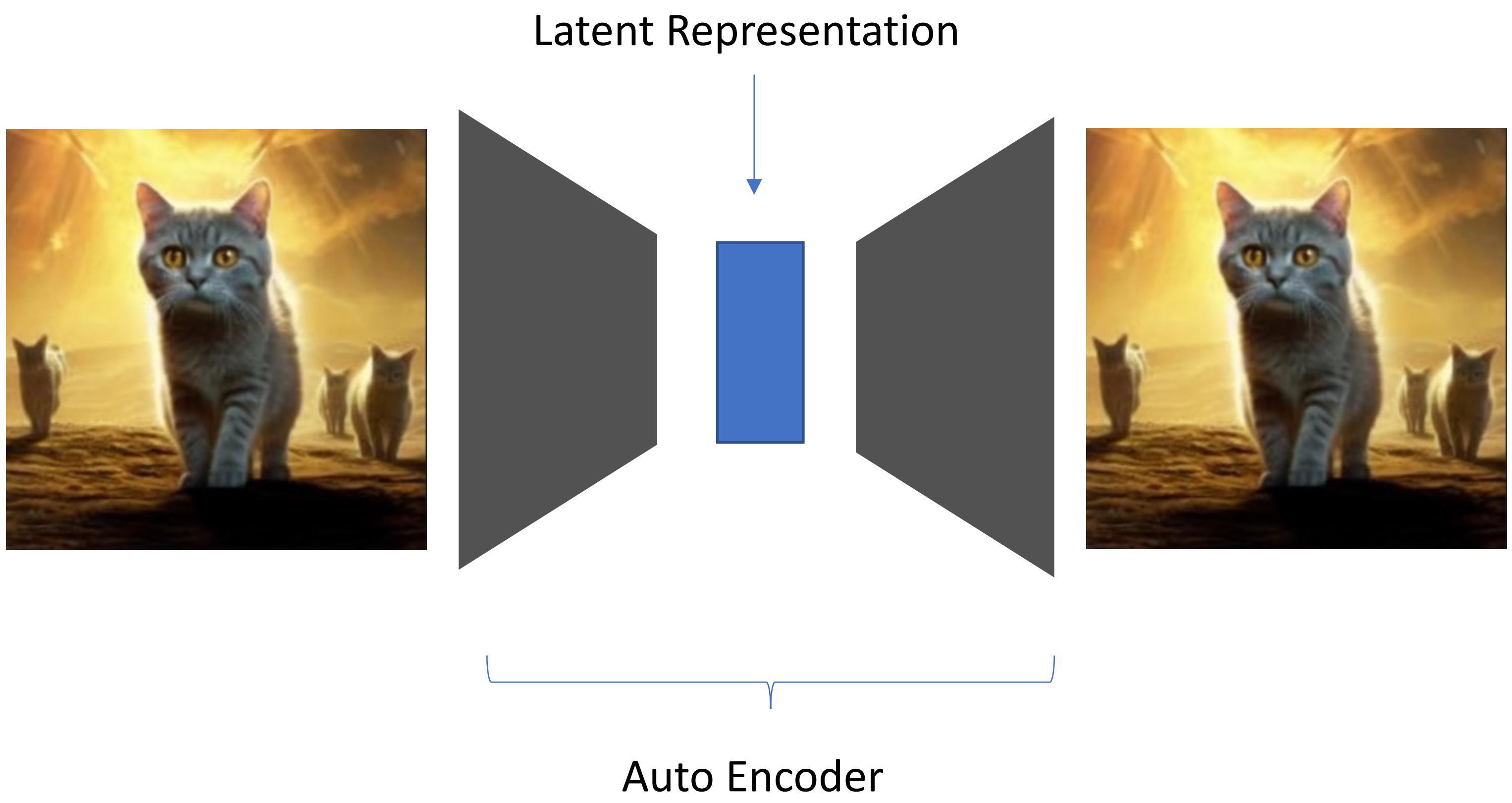
Text to image



Classifier-Free Guidance



Auto-encoder!



Overall framework for text-to-image generator