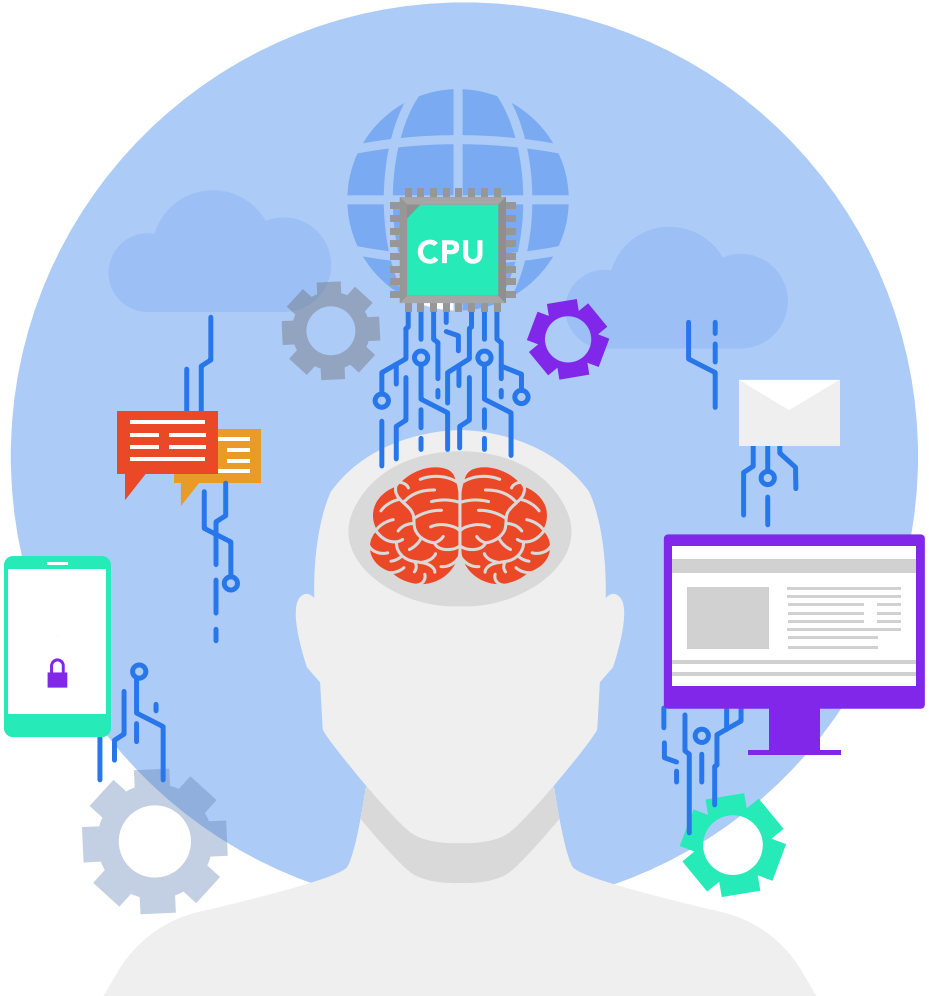
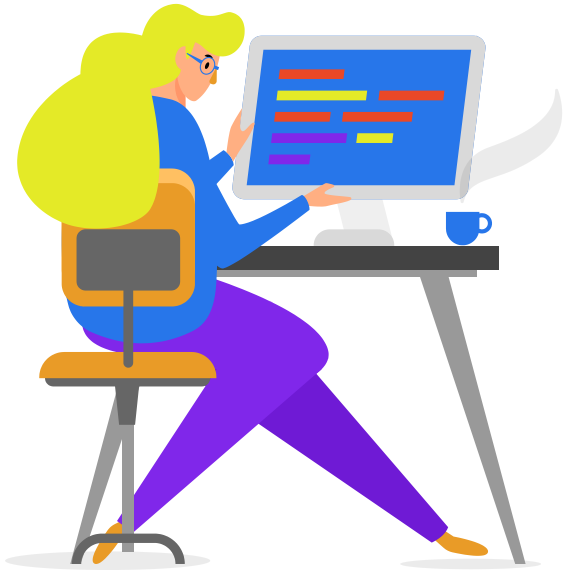



# Taming Transformers for High Resolution Image Synthesis (VQGAN-CLIP-ESRGAN implementation)

Siddhant Bikram Shah 2K19/CO/374
Ayush Karn
2K19/CO/454

# Contents



# Introduction

# How We Perceive Images

## Theory of Perception


"A white cat looking up," or "a cat". Humans appear to perceive pictures via distinct representations such as cat, white, or looking up.

This **theory of perception** suggests that our visual reasoning is **symbolic**, we ascribe meaning through discrete representations and modalities. We may comprehend **relationships** between distinct words or symbols using this symbolic technique.

This is usually referred to as the ability to model **long-range dependencies** in machine learning. For example, we recognised that looking up relates to the **cat's action** and **white** refers to the cat's **description** in the statement "**a white cat looking up**."

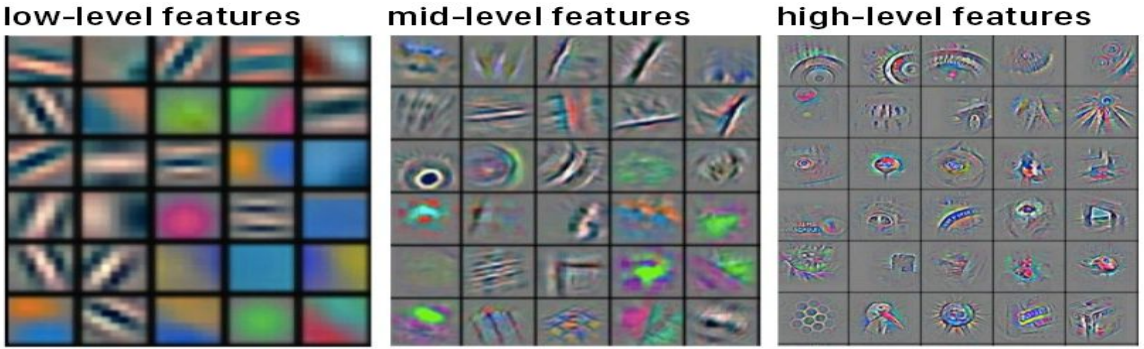
Despite the fact that there has been a lot of research into exploring complex reasoning using discrete representations, most computer vision (CV) approaches do not include modalities. They think in terms of **pixels** instead.

# How Most CV Technique process images

Instead of thinking in terms of large chunks of information, common CV techniques focus on an image's smallest unit, the **pixel**. Each pixel channel—red, green, blue—represents a color value in a continuous scale.

Convolutional neural networks (CNN) proved that we can model local interactions between pixels by restricting interactions within their local neighborhood (i.e., the kernel). This allows us to "**compose**" pixels together and **learn visual parts**.

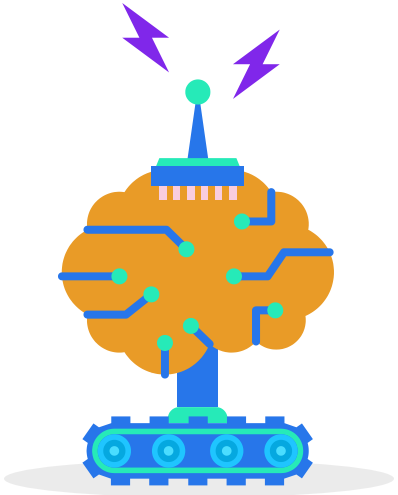
CNN learned how to compose pixels at varying layers of abstraction: **pixels become edges, edges become shapes, and shapes become parts.**



# Transformers: An Overview

## Properties

1. Employs Encoder-Decoder(or Sequence-to-Sequence) structure.
2. Divides images into patches similar to tokens in NLP.
3. Strong long-range modelling capabilities



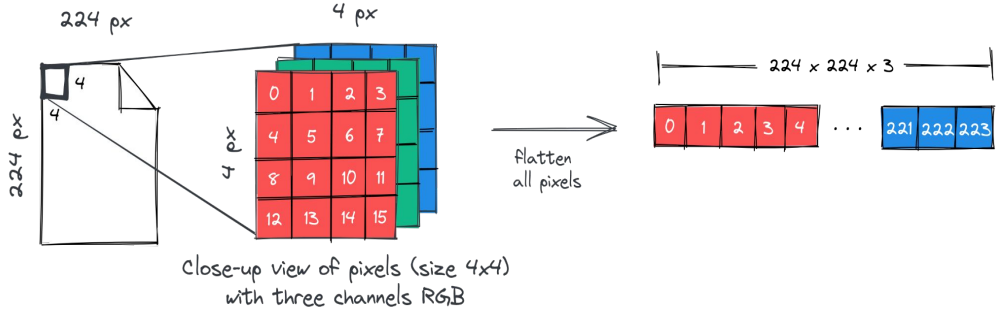
## Salient Features

1. Understand faraway sequential data
2. Fast processing
3. Adaptable to any kind of sequential data

## Applications

1. Natural Language Processing
2. Computer Vision

# Use of Transformer to Model Interaction




03

Reduce the context by downsampling the 224-px image to 32-, 48-, and 64 pixel dimensions. However, instead of downsampling, VQGAN employs **codebooks.**

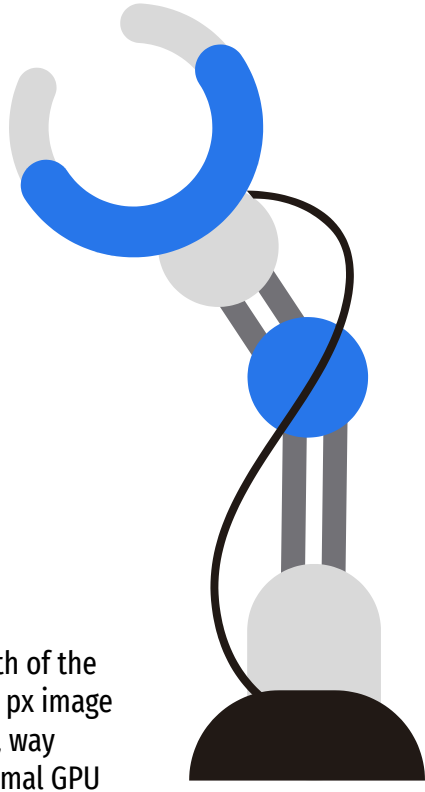
02

Due to its attention mechanism, it computes the pairwise inner product between each pair of the tokens.

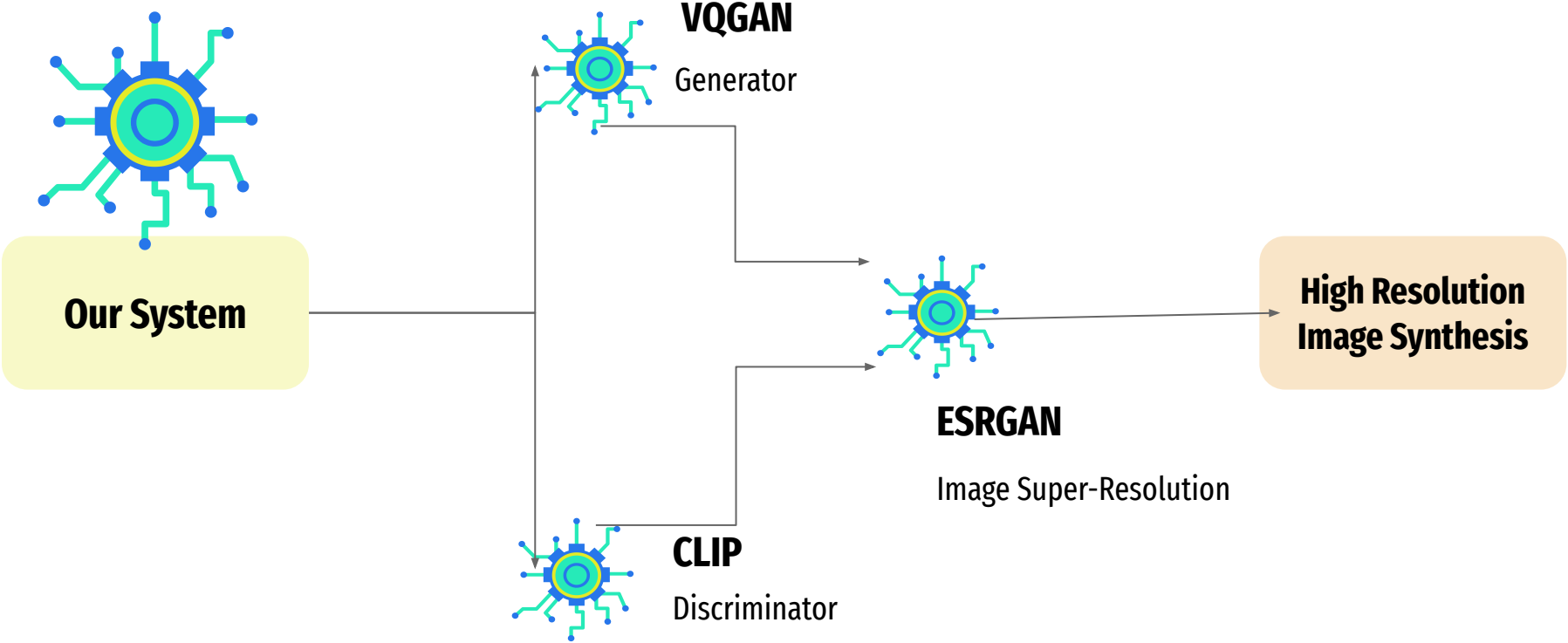
01

Flatten pixel of image into sequence and use that as input

Note: Computation scales quadratically with the length of the input sequence. A 224 x 224 px image will have a length of $224^2 \times 3$, way above the capacity of a normal GPU

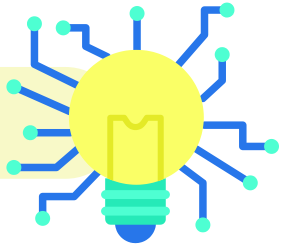


# System Flowchart



# Motivation

# Why Transformers?



## CNN vs Transformers

### CNN

1. Inductive Bias
2. Low expressivity
3. Learns complex input relationships by exploiting prior knowledge
4. Low training complexity

**Learns images with context**

### Transformers

1. No Inductive Bias
2. High expressivity
3. Cannot learn complex input relationships efficiently
4. High training complexity(quadratic)

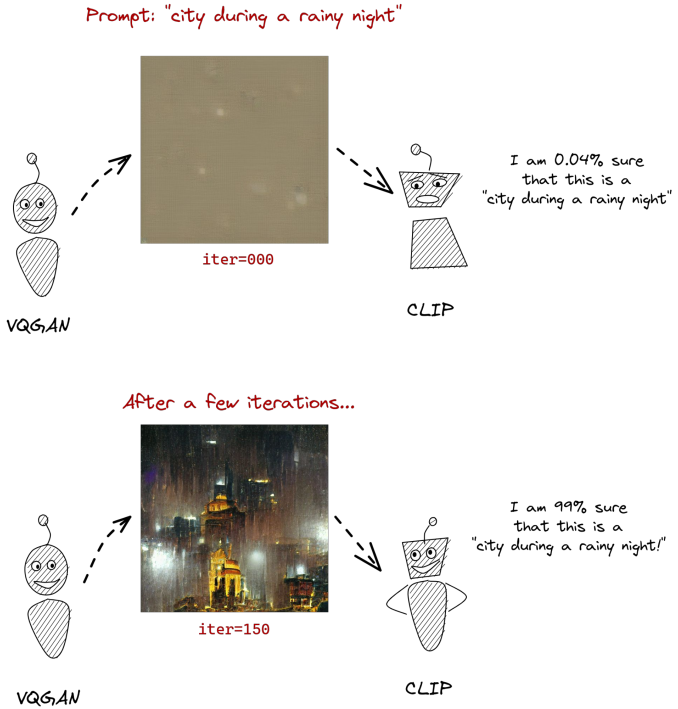
**Models composition**

# System Overview

# VQGAN

VQGAN stands for **V**ector **Q**uantized **G**enerative **A**dversarial **N**etwork, while CLIP stands for **C**ontrastive **I**mage-**L**anguage **P**retraining. The interaction between these two networks is referred to as VQGAN-CLIP. They're two different models that function together.

VQGAN creates the pictures, whereas CLIP assesses how well an image fits our text query. Our generator is guided by this interaction to create more accurate images.

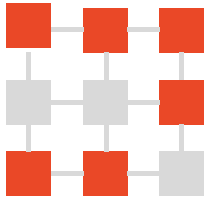


01



Model **long-range dependencies** by representing images discreetly

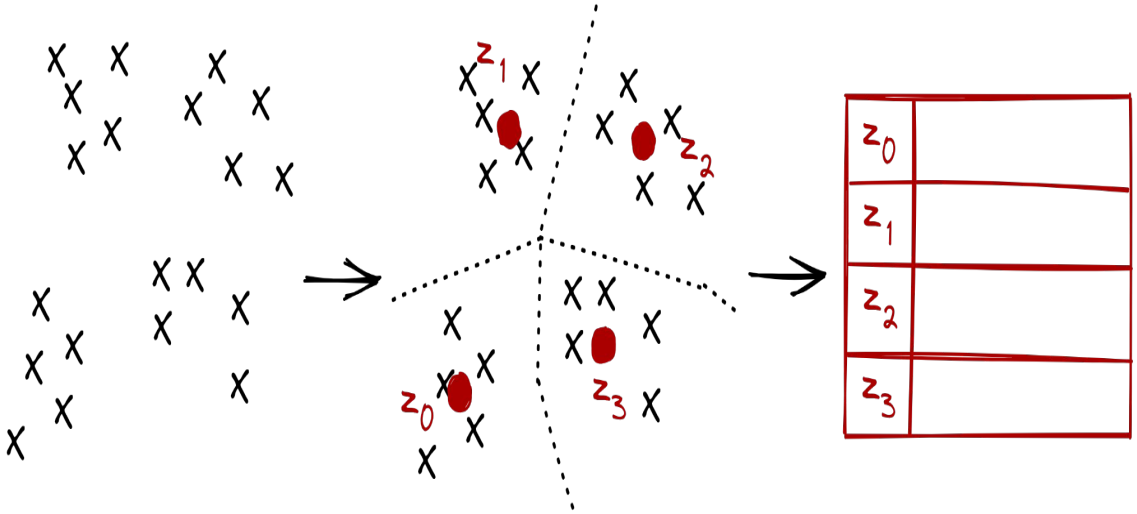
02



Pixel-based approach to learn **local interactions** and **visual parts**

**VQGAN** was able to combine both of them. It can learn
(1) visual parts of an image
(2) relationship between these parts.

# Vector Quantization




Vectors in a 2D space

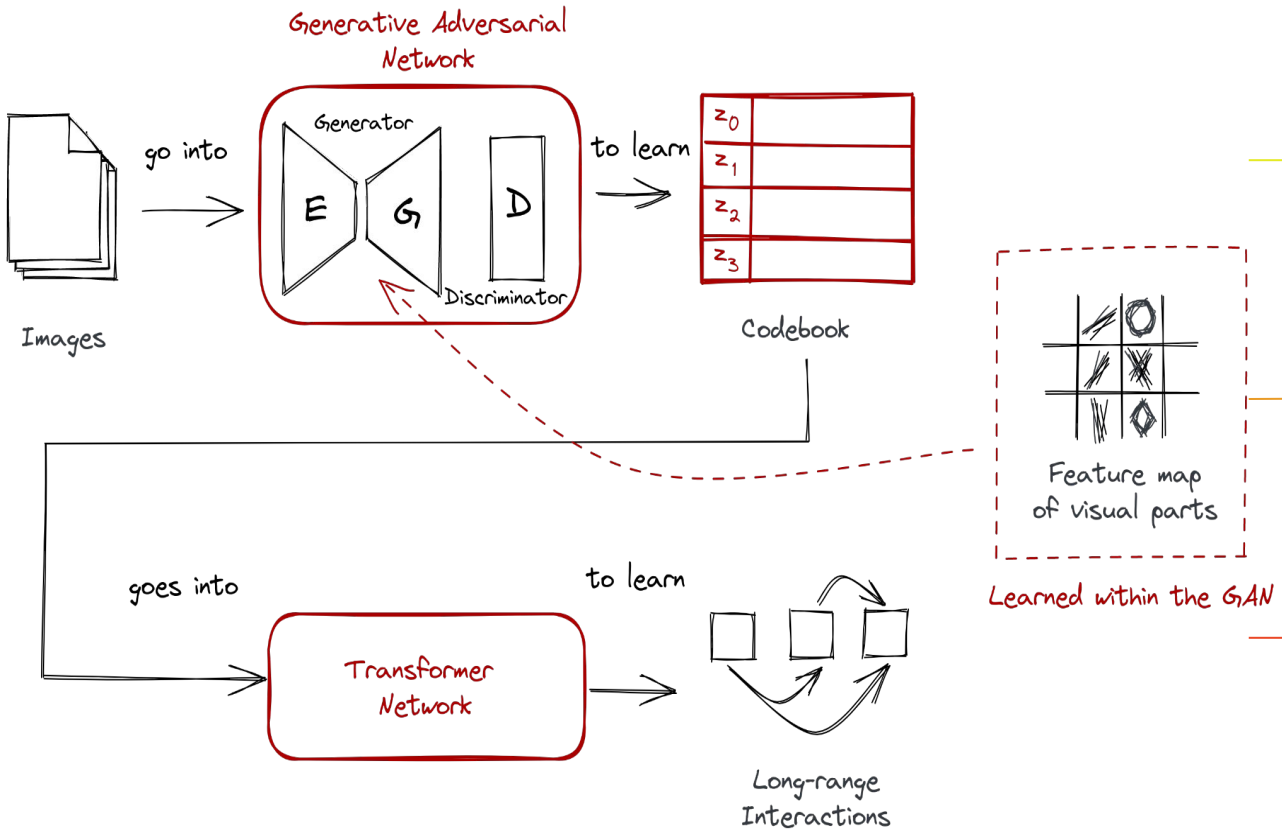
Codewords that represent each cluster

The resulting dictionary of centroids or codebook

Vector Quantization is the process of dividing vectors into groups that have approximately the same number of points closest to them.

Each group is then represented by a centroid (codeword), obtained via k-means or other clustering algorithm. Finally, one learns a dictionary of centroids (codebook) and their corresponding members.

# Codebook




The **codebook** is generated through a process called vector quantization (VQ). i.e., the "VQ" part of "VQGAN."

Vector quantization is a signal processing technique for encoding vectors.

It represents all visual parts found in the convolutional step in a quantized form, making it less computationally expensive once passed to a transformer network

# Training the system



## Training the GAN

- We train the GAN on a dataset of images to learn not only its visual parts, but also their codeword representation, i.e., the codebook.

## Training the Transformer

- We train the transformer on top of the codebook with sliding attention to learn long-range interactions across visual parts.

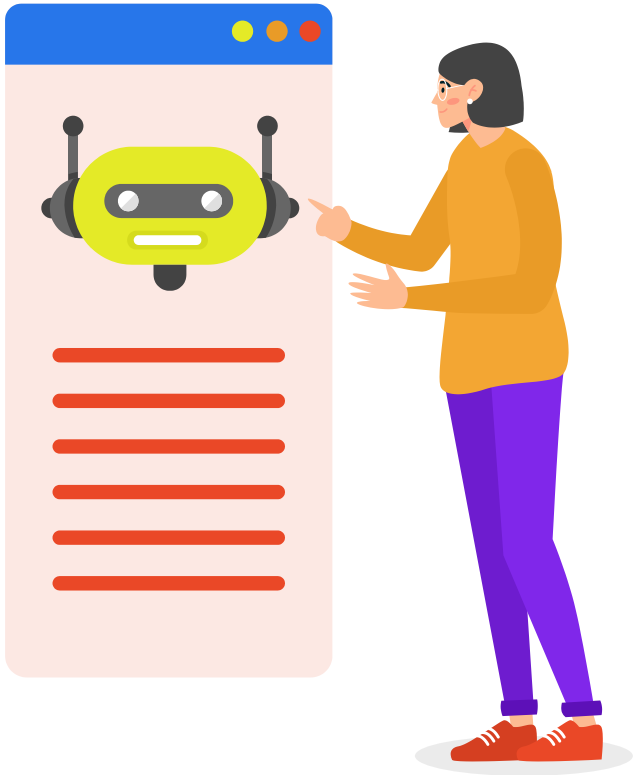
# Training the GAN

**01 Generator N**

for creating new samples

**02 Discriminator D**

classifies the samples as either real or fake



**03 logD(x)**

measures the probability of the discriminator D to say that a real data instance x is actually real

**04 log(1-D(^x))**

Measures discriminator estimate of the probability that a fake instance is real

$$L_{GAN}(N,D)=[logD(x)+log(1-D(\hat{x}))]$$

# Training the GAN

Vector quantization is done between encoder and decoder. After encoding the input x into ^z=E(X), we perform an element-wise operation q to obtain **discrete version of the input**:

$$z_q = \mathbf{q}(\hat{z}) := \arg\min_{z_k \in \mathcal{Z}} ||\hat{z}_{ij} - z_k||$$

Instead of reconstructing from the encoder output ^z, we do it from its quantized form $z_q$ to obtain the **reconstructed image**:

$$\hat{x} = G(z_\mathbf{q}) = G(\mathbf{q}(E(x)))$$

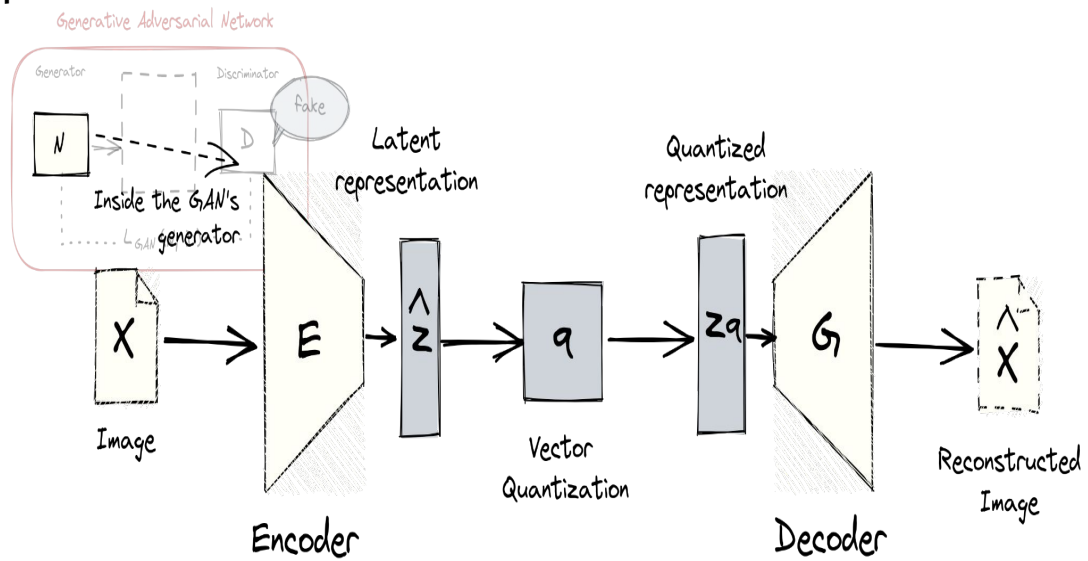
The autoencoder model and the codebook were trained together using the **loss function**:

$$\mathcal{L}_{VQ}(E, G, Z) = ||x - \hat{x}||^2 + ||sg[E(x)] - z_\mathbf{q}||_2^2 + ||sg[z_\mathbf{q}] - E(x)||_2^2$$

**Discrete version of the input**

**Reconstructed image ^x**

**Loss function**



# Training the GAN

$$L_{VQ}(E,G,Z) = \|x-\hat{x}\|^2 + \|sg[E(x)]-z_q\|^2_2 + \|sg[z_q]-E(x)\|^2_2$$

**01** $\|x-\hat{x}\|2$

**Reconstruction loss**, it checks how well our network was able to approximate (via ^x) our input x when given only its quantized version zq.This is computed as perceptual loss, not in a per-pixel basis

**02** $\|sg[E(x)]-z_q\|^2_2$

**Embedding Optimizer**. The operation sg stands for “stop-gradient.” It is an identity function during forward pass, and has zero partial derivatives, thus constraining it to be constant

**03** $\|sg[z_q]-E(x)\|^2_2$

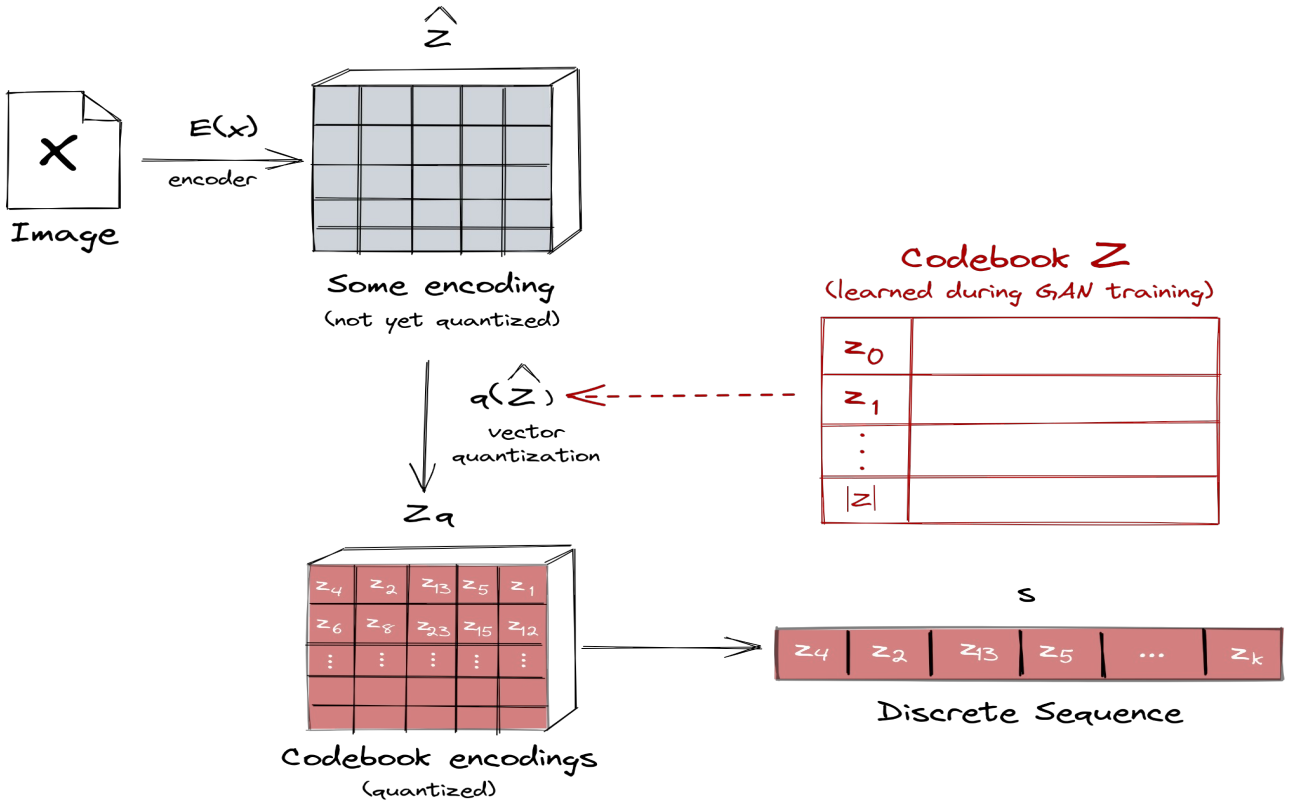
**Commitment loss**. It ensures that the encoder
E commits to a particular representation of the image.

# Training the Transformer

Codebook Z provides a discrete representation of our visual parts that can readily be fed to a transformer.

We represent the images in terms of the codebook-indices of their embeddings

This means that the encoding $z_q$ of an image x can be represented as a sequence s from our codebook Z, that is, $s \in \{0,.....|Z|-1\}$



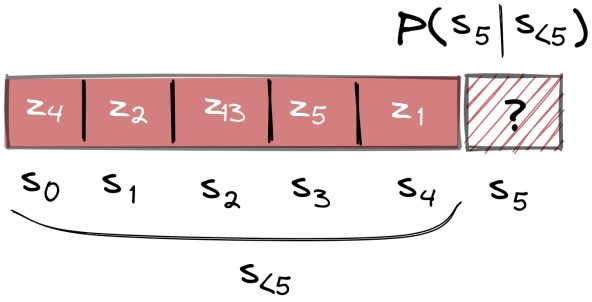
# Training the Transformer

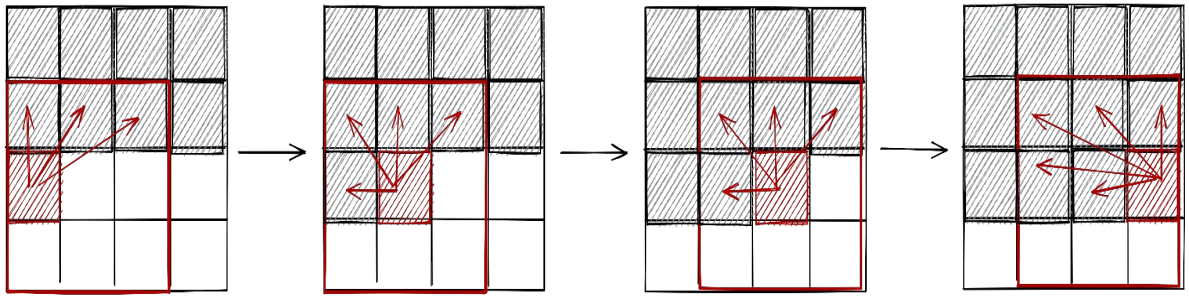
If we have indices $s_{<i}$, the transformer predicts the distribution of possible **next indices**:

$$p(s_i|s_{<i})$$

We get to compute the **likelihood** as $p(s)=\Pi_i p(s_i|s_{<i})$ and minimize the **transformer loss**

$$L_{Transformer}=E_{x\sim p(x)}[-\log p(s)]$$

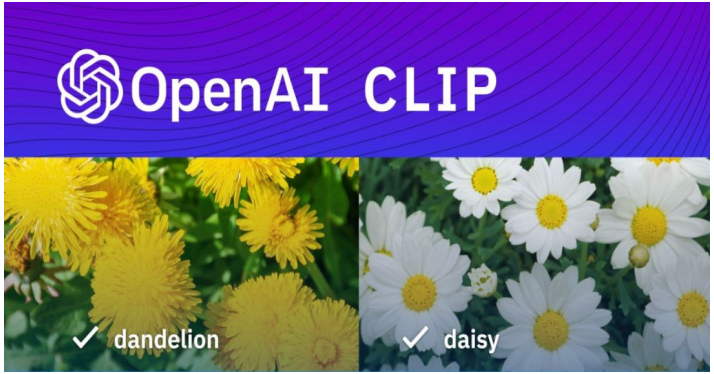


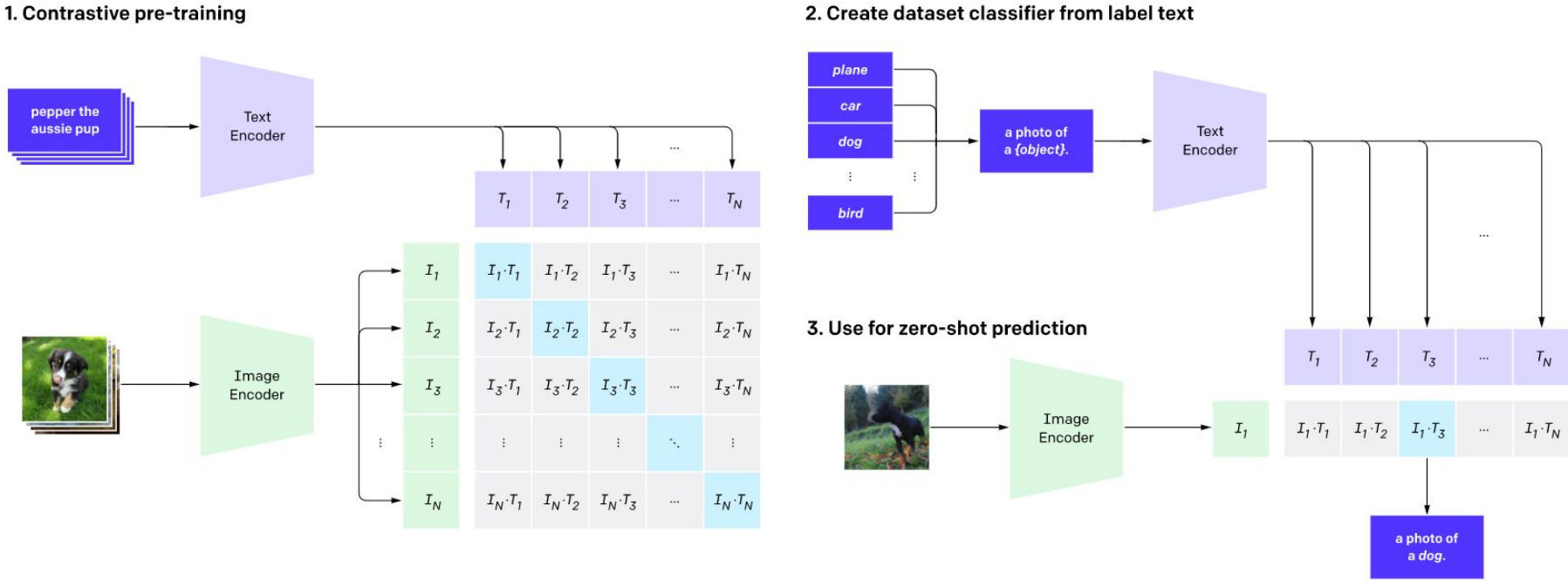
# CLIP

CLIP stands for **C**ontrastive **L**anguage-**I**mage **P**retraining. It builds on a large body of work on zero-shot transfer, natural language supervision, and multimodal learning.

CLIP models need to learn to recognize a wide variety of visual concepts in images and associate them with their names. As a result, CLIP models can then be applied to nearly arbitrary visual classification tasks.



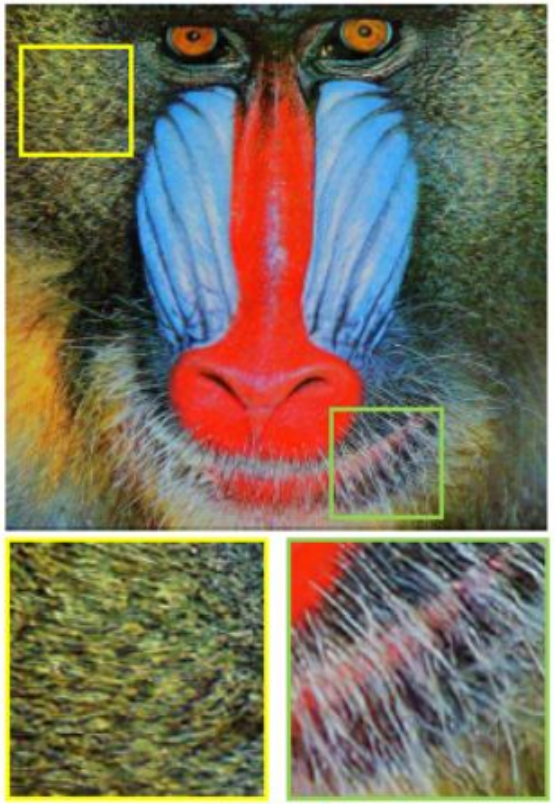
# How CLIP works



# ESRGAN

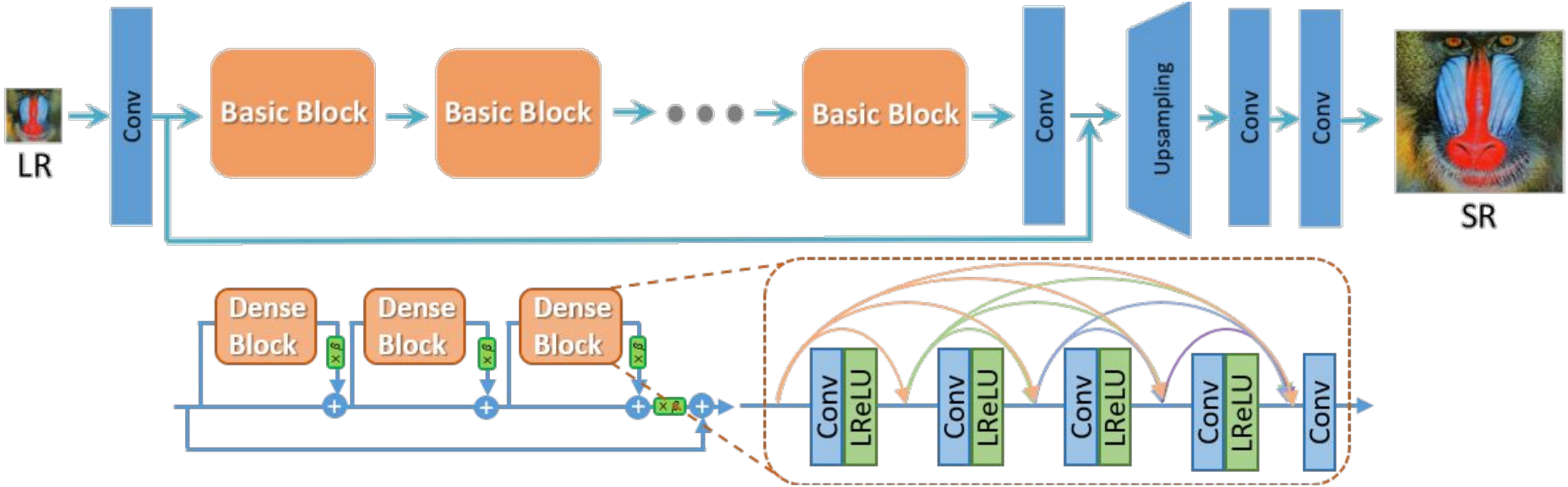
ESRGAN stands for **E**nhanced **S**uper-**R**esolution **G**enerative **A**dversarial **N**etwork. Image super-resolution techniques reconstruct a higher-resolution image or sequence from the observed lower-resolution images, e.g. upscaling of 720p image into 1080p.

The ESRGAN uses a Relativistic discriminator to better approximate the probability of an image being real or fake; thus, the generator produces better results.



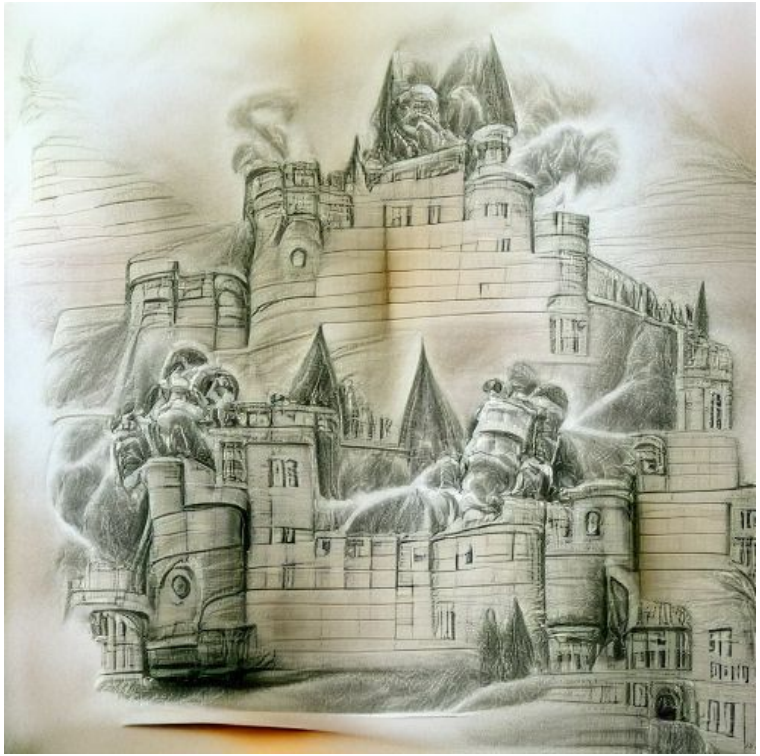
ESRGAN

# How ESRGAN works

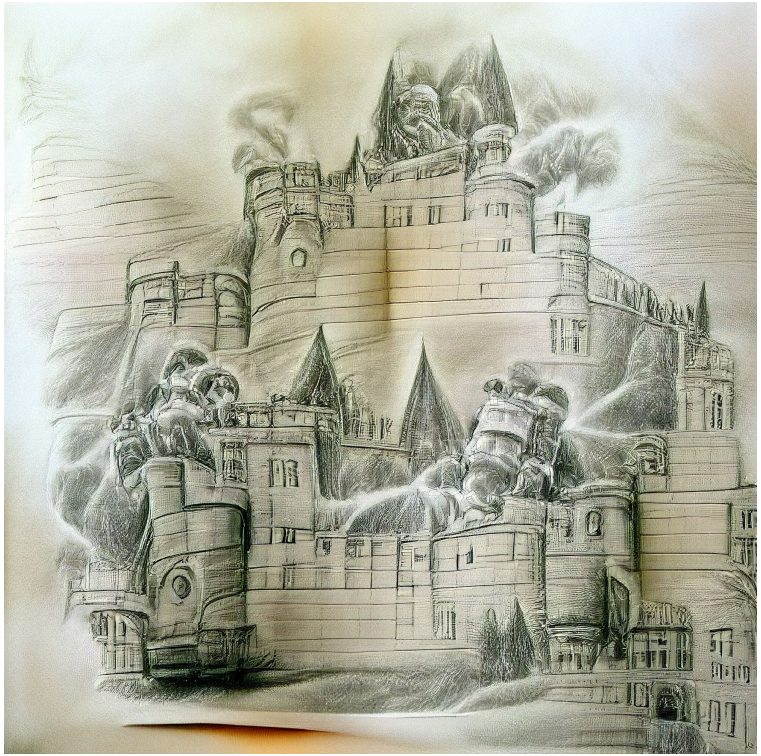


# Results

# Prompt: Hogwarts Castle Pencil Sketch



496x496 px



1984x1984 px

# Prompt: Himalayas Cloud Sky Heavenly

496x496 px

1984x1984 px

# Prompt: Snowy Mountains Sunset Pencil Sketch

496x496 px

1984x1984 px

# Conclusion and Future Scope

# Conclusion and Future Scope

In this project we have presented VQGAN-CLIP, a method of generating and manipulating high resolution images based on only human written text prompts. We tried to introduce novelty to the project by integrating ESRGAN to it as well.

Compared to its competitors like BigGan, VQGAN-CLIP performs well even when the image content and text prompt have low semantic similarity.

Research on the use of augmented images during optimization proves that it is faster and shows more quality than the traditional codebook method.

# References


1. Esser, Patrick, Robin Rombach, and Bjorn Ommer. "Taming transformers for high-resolution image synthesis." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.

2. Li, Yangguang, et al. "Supervision exists everywhere: A data efficient contrastive language-image pre-training paradigm." arXiv preprint arXiv:2110.05208 (2021).

3. Wang, Xintao, et al. "Esrgan: Enhanced super-resolution generative adversarial networks." Proceedings of the European conference on computer vision (ECCV) workshops. 2018.

4. “Illustrated VQGAN” https://ljvmiranda921.github.io/notebook/2021/08/08/clip-vqgan/

5. “Guide To Image Super-Resolution By ESRGAN” https://analyticsindiamag.com/guide-to-image-super-resolution-by-esrgan/

6. “CLIP: Connecting text and images” https://openai.com/blog/clip/