Multimodal Image Generation: with CLIP and VQGAN Architectures

29 February 2024

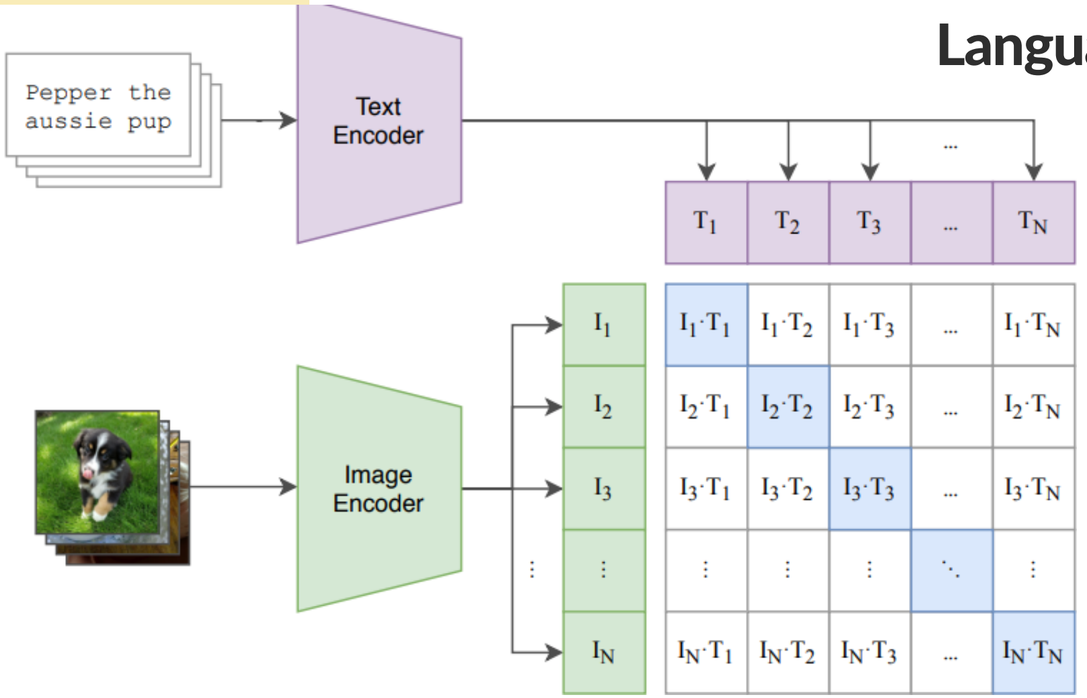
Venkatesh BM 211AI039
Vivek vittal biragoni 211AI041

Project Overview

- Explore generative technology with multimodal image generation
- Seamlessly integrate textual prompts with visual elements
- Utilizes architectures like: CLIP and VQGAN
- Harmoniously fuse diverse data types for creative synthesis
- aid the future of generative art through advanced models

Literature Survey

Learning Transferable Visual Models From Natural Language Supervision

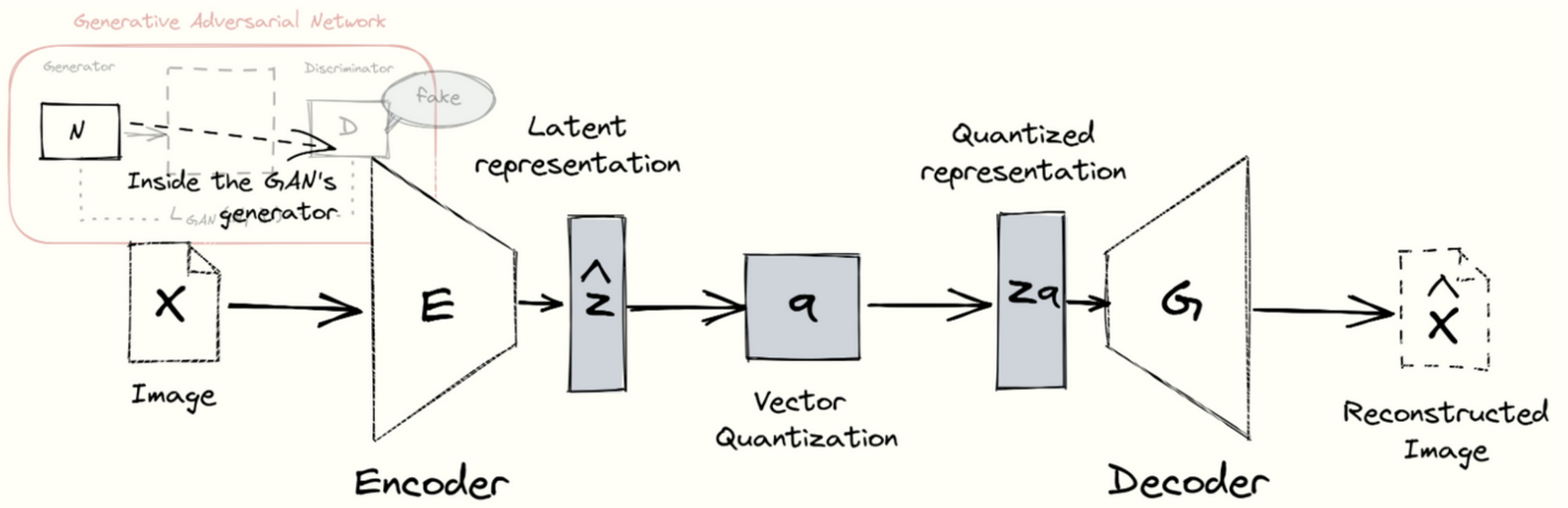


- CLIP stands for Contrastive Language-Image Pre-Training by Open Ai, famous work in multimodal DL.
- Connects Text and images by learning transferable visual models from NL supervision.
- Provides robust methodology for encoding text and images.

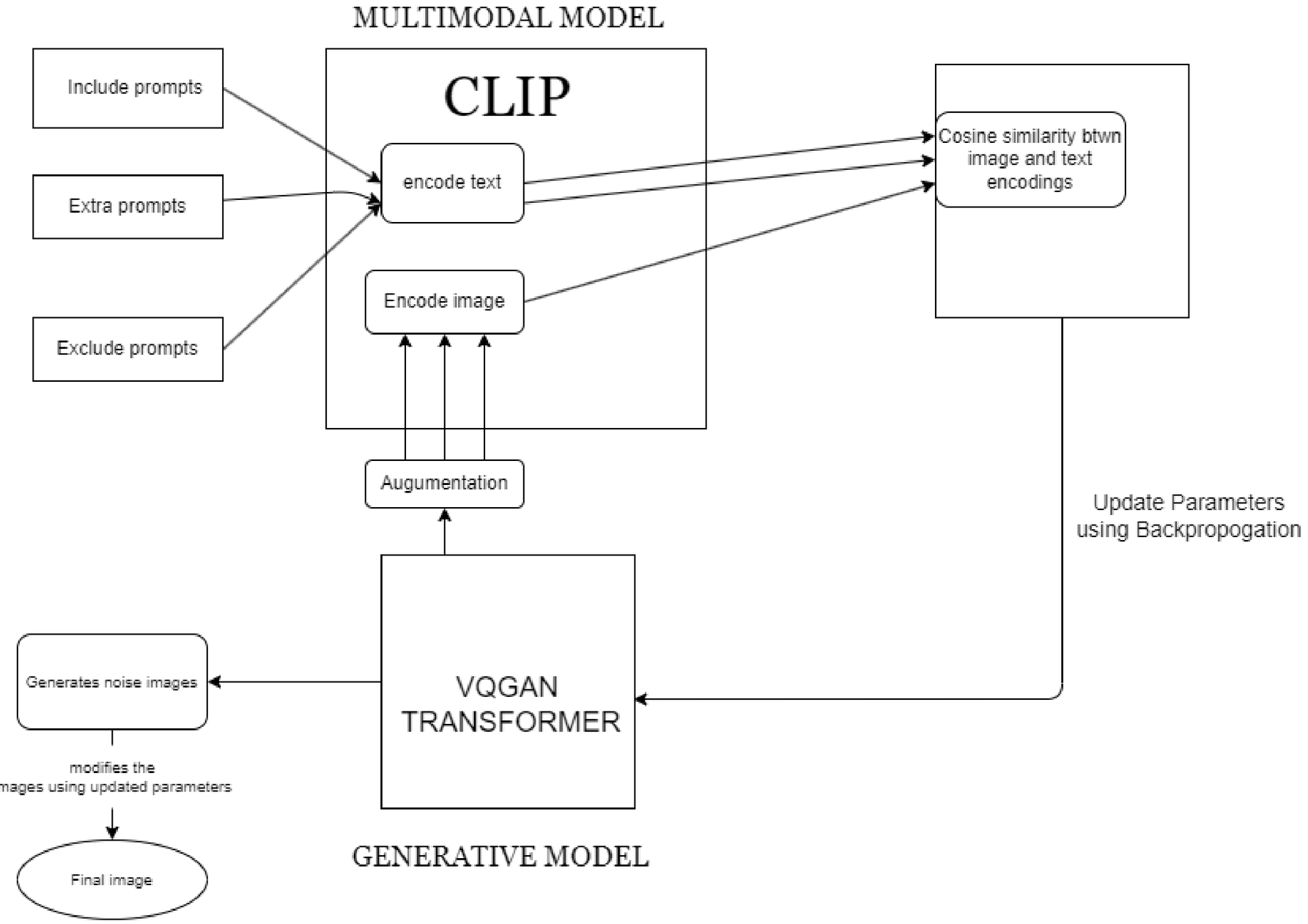
Literature Survey

Taming Transformers for High-Resolution Image Synthesis

- The paper aims to apply transformer models for high-resolution image synthesis, specifically in the megapixel range.
- Instead of pixel representation, images are portrayed as compositions of perceptually rich constituents from a codebook(used in VQGAN)
- The approach efficiently models global interrelations within images using a transformer architecture.



Methodology



Initialization:

Start with a text prompt and a random noise image

Encoding

Use CLIP to encode the text prompt and augmented images.

Cosine Similarity

Calculate cosine similarity between text and image encodings.

Loss Calculation

Derive loss from cosine similarity, aiming to minimize dissimilarity

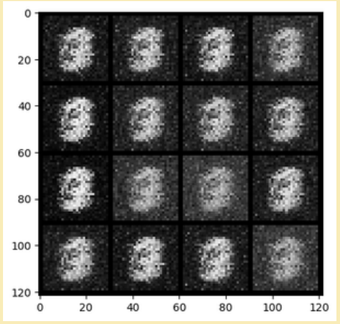
Optimization Objective

Adjust image parameters to minimize the calculated loss.

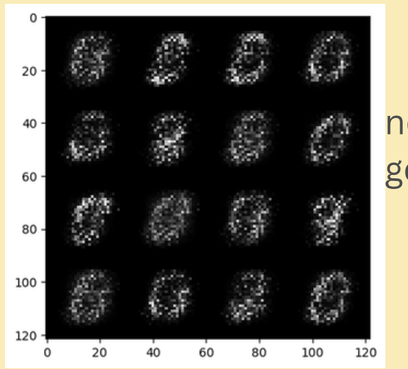
Objectives and Expectations

- Learn the basic GAN and advanced GAN architectures, and why advanced GAN?
- Understand the architecture of CLIP and VQGAN.
- Loss functions used in them.
- Finally get high resolution images generated based on the provided text inputs.
- Try to extrapolate in between the generated images to possibly create artistic visuals.

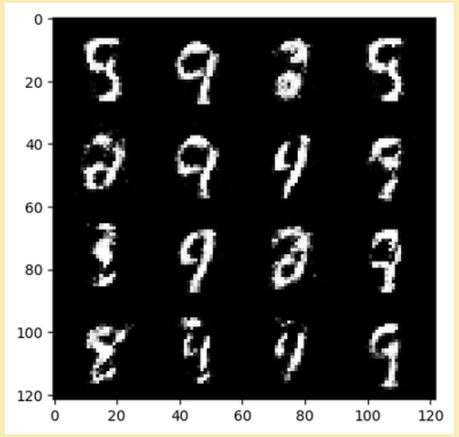
basic_gan_dl.ipynb



Initialliy generated noise image



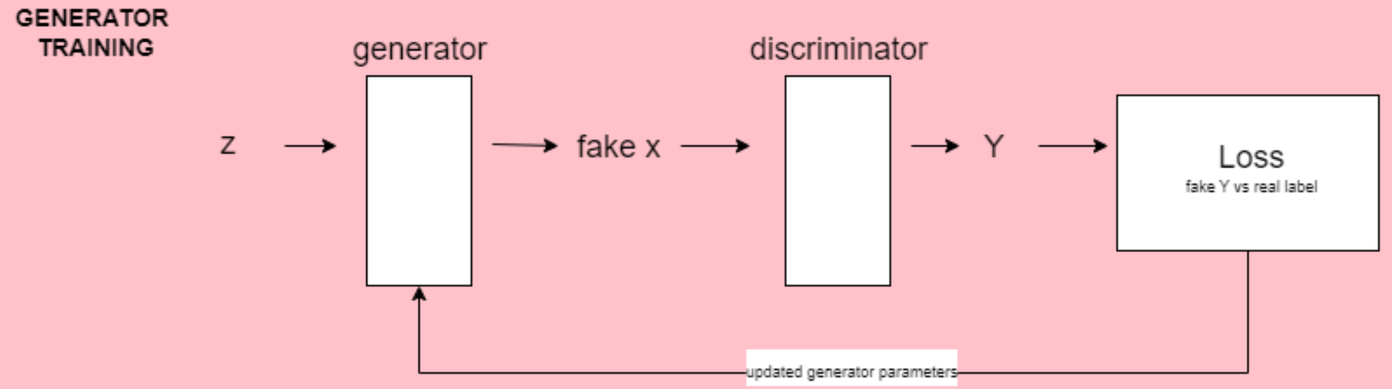
noise image getting updated

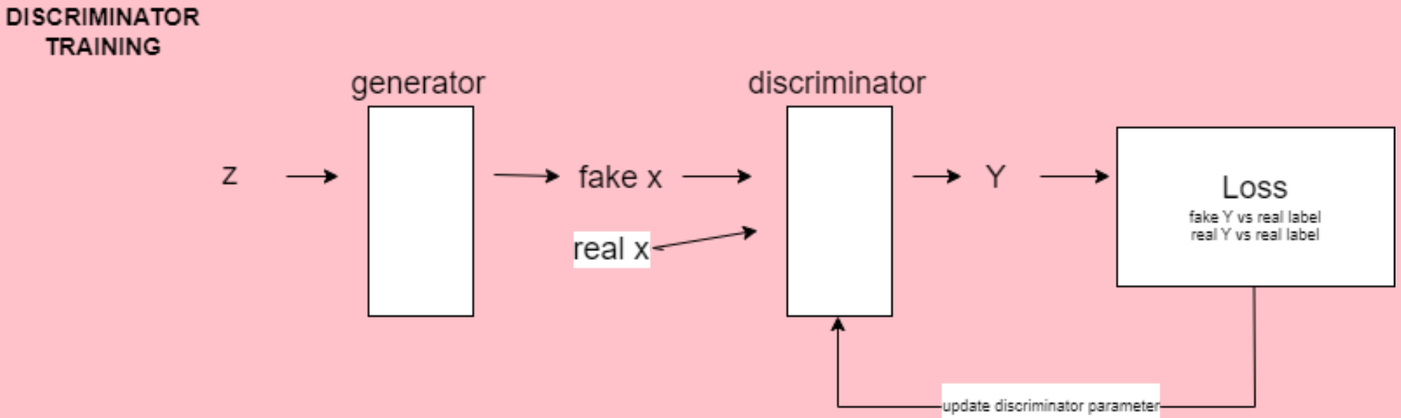


noise image at 80th epoch

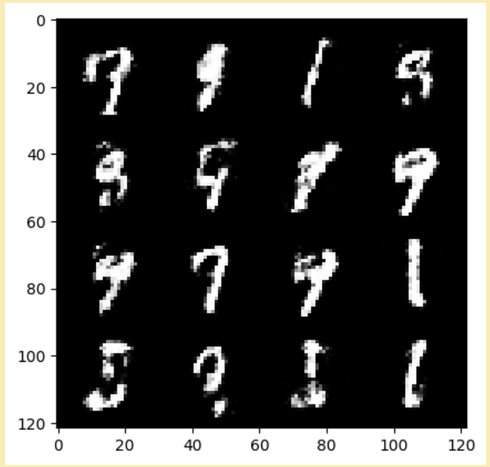
Present work progress

Basic gan architecture

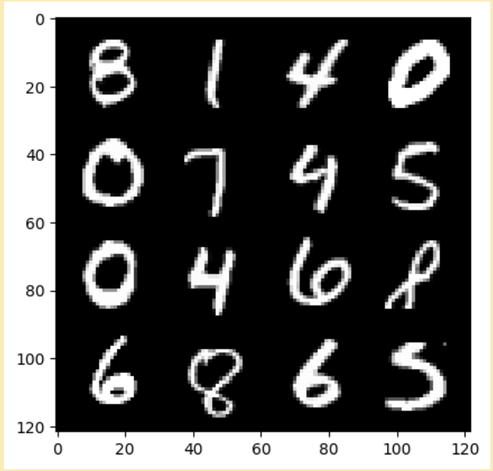




basic_gan_dl.ipynb



After running for 100 epochs



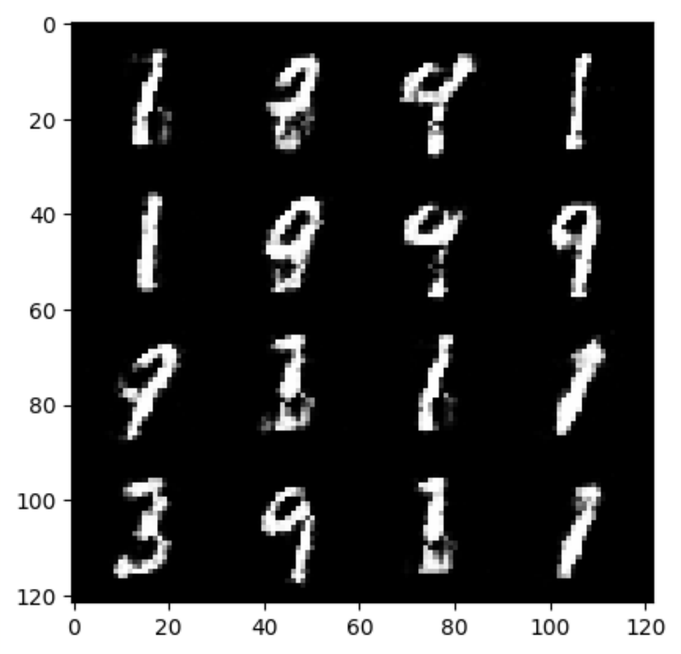
This is an image from the MNIST dataset

Present work progress

Basic gan architecture

- We used the MNIST dataset, which consists of handwritten digits from 0 to 9
- Generative Adversarial Networks (GANs): Implemented a basic GAN architecture.
  - A Generator model tries to create new, realistic images of handwritten digits (fakes).
  - A Discriminator model tries to distinguish between real images (from the MNIST dataset) and the fake images generated by the Generator.
- Used the nn.BCEWithLogitsLoss function to calculate both the generator and discriminator losses.
  - This function is suitable for binary classification tasks like distinguishing real and fake images.

basic_gan_dl.ipynb



Here we see that our generator has already moved to repeat so many same numbers in this small batch of 16 images

Present work progress

Observed problems with the Basic gan architecture

- **Training Instability:** The training process of basic GANs can be unstable and prone to mode collapse.
- **Vanishing Gradients:** The loss function used in basic GANs (often the Binary Cross Entropy loss) can lead to vanishing gradients for the discriminator(towards the extreme values like the 0 and 1s for sigmoid functions, not large enough to make enough learning). This makes it difficult for the discriminator to learn effectively and also mode collapse for the generator.
- **Non-interpretable Loss:** The loss function in basic GANs doesn't directly correspond to the quality of the generated images, making it difficult to interpret and assess the training progress.

Present work progress

advanced_gan_dl.ipynb

weights and biases link

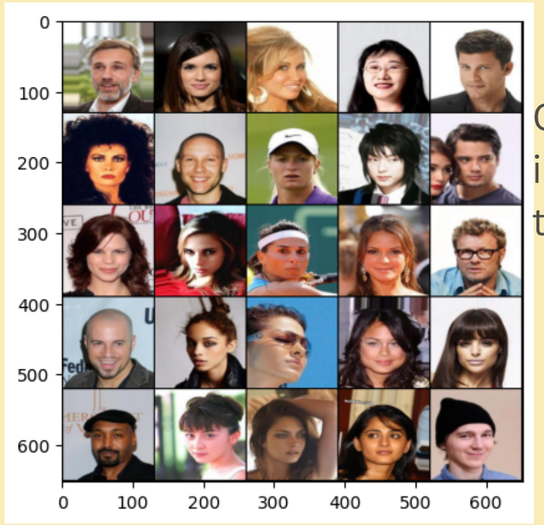
- Analogy:
  - distinguishing real from fake coins (real vs. fake data).
  - Traditional GAN: Only compare real and fake coins (limited learning).
  - WGAN-GP: Also examine slightly altered coins (mixed images) to better understand subtle differences (balanced gradients).

WGAN-GP (Wasserstein GAN with Gradient Penalty)

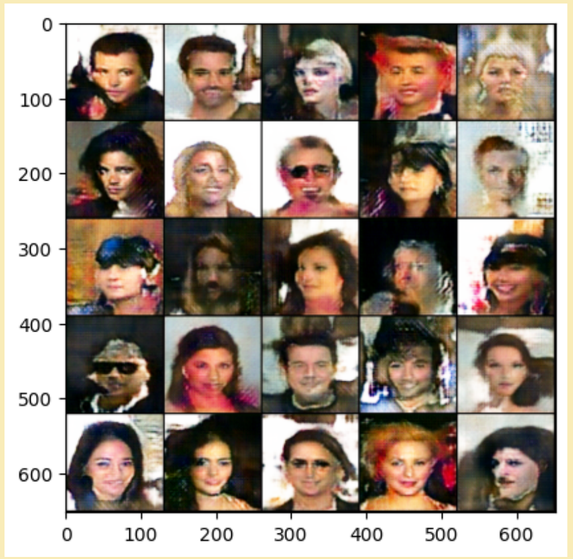
- The gradient penalty encourages the critic to maintain balanced and informative gradients throughout its decision landscape. In simpler terms, the penalty discourages the critic from having gradients that are either too large or too small, preventing it from becoming overly confident and promoting more effective learning.
- WGAN-GP uses a technique called **mixed images** to estimate gradients at intermediate points in the data space. These mixed images are created by **interpolating** (blending) between real and fake images.
- By calculating the gradients for these mixed images, WGAN-GP can understand how the critic's decision function is changing throughout the feature space. The gradient penalty then uses this information to adjust the critic's behavior and ensure it maintains informative gradients, leading to improved learning and stability in the training process.

advanced_gan_dl.ipynb

weights and biases link



Original images from the dataset



our model generated images, after several 1000s epochs

Present work progress

Implementation of WGAN-GP

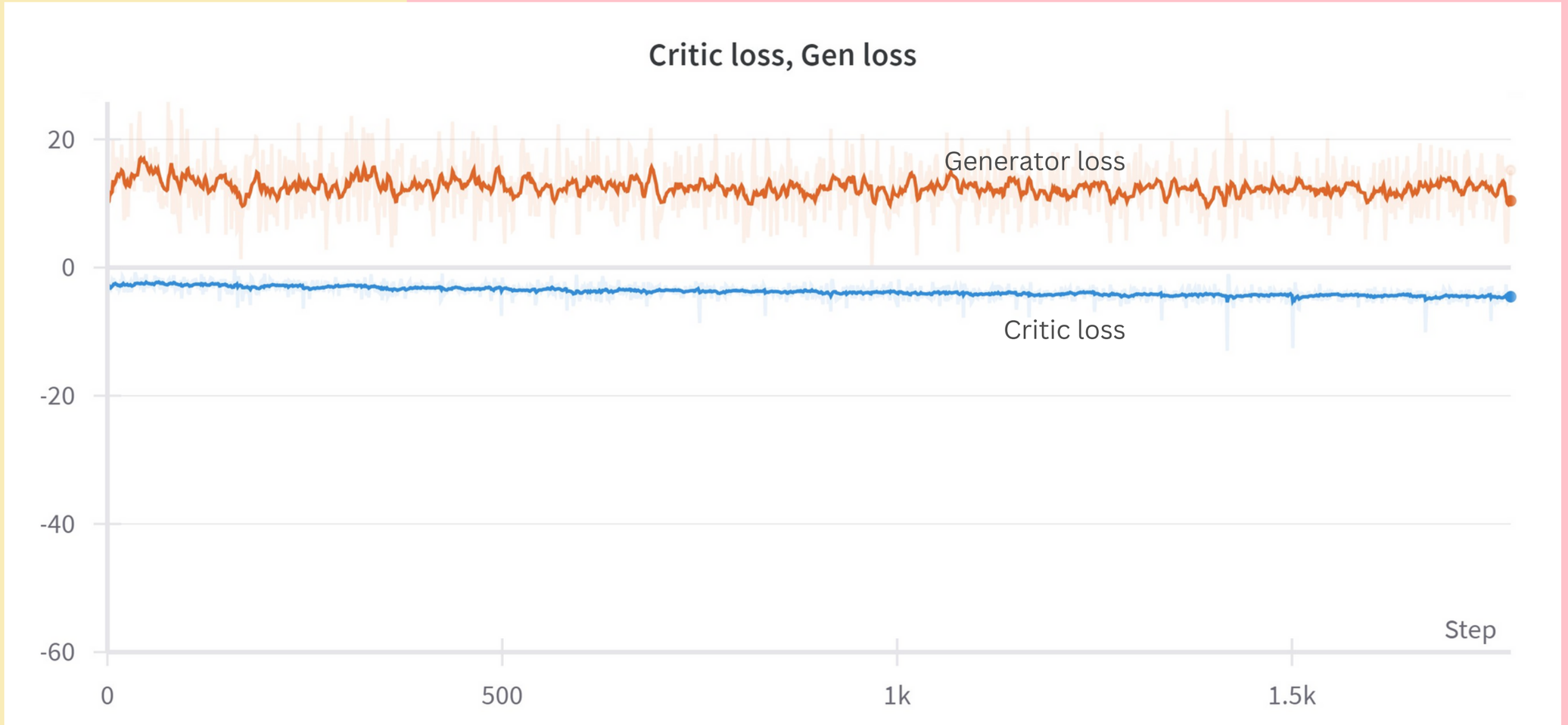
- **Models:**
  - Generator (Gen): Takes random noise as input and generates realistic images.
  - Critic (Crit): Evaluates both real and generated images, trying to distinguish between them.
- **Dataset:**
  - CelebA Dataset: Used in this example, containing images of celebrity faces.
- **Training Loop:**
  - **Critic Training:**
    - Critic is trained to distinguish real and fake images using gradient descent.
    - The gradient penalty term is added to the critic's loss, encouraging balanced gradients.
  - **Generator Training:**
    - Generator is trained to fool the critic by generating images that the critic classifies as real.
- **Logging and Monitoring:**
  - Weights & Biases (W&B) Integration Tracks training progress and visualizes generated images.
- **Checkpointing:** Periodically saves model states for potential resumption later.
- **Loss and Image Visualization:** Plots training losses and displays generated images at specific intervals.

advanced_gan_dl.ipynb

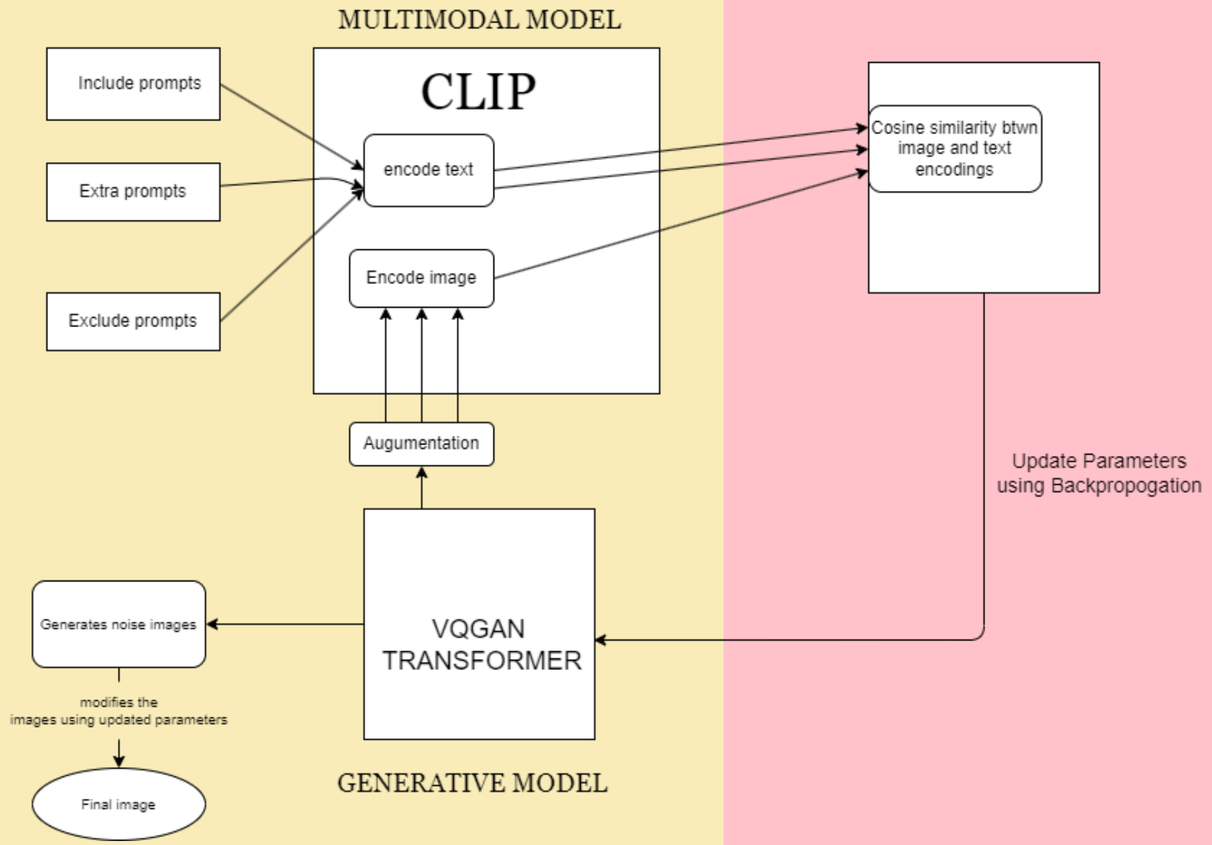
weights and biases link

Present work progress

This code We are running it on the previously saved checkpoints and loaded them, so this is after a huge number of iterations



Plan ahead



- With this we will are now in a position to understand GAN architecture and tweak it if required.
- Implement the CLIP+VQGAN as described earlier(main part of the project).
- additional: Try to include the stable diffusion.
- Play with the latent space of VQGAN to produce artistic images.

References

- https://medium.com/@anuj.1306.gupta/vqgan-an-introduction-to-its-architecture-training-process-and-applications-in-machine-learning-332910248e6e
- https://medium.com/one-minute-machine-learning/clip-paper-explained-easily-in-3-levels-of-detail-61959814ad13
- https://ljvmiranda921.github.io/notebook/2021/08/08/clip-vqgan/
- https://arxiv.org/pdf/2103.00020.pdf
- Generative AI on udemy.
- https://arxiv.org/abs/2012.09841
- And several other articles on medium.

Thank You

Venkatesh BM 211AI039
Vivek vittal biragoni 211AI041