CVIP Project Report

Team Members

Sreeram Melpadi
sreeramm

Piyush Gulhane
pgulhane

Thushar Thorenur Govindaraju
thushart

1. Problem statement

In this project, we aim to develop a generative model that is able to understand the distribution of existing emojis in latent space and produce entirely new, unique emoji designs. The main focus of the project is implementing Deep Convolutional Generative Adversarial Networks (DCGANs) as the core architecture for emoji generation, as GANs are very efficient in generating high-quality and sharper-looking emojis. We encode existing emojis into a latent space representation, allowing us to visualize and analyze how various features and attributes are captured within this space. This will help us assess how well the model captures continuous variations in shape, color, and semantic attributes across different emojis. Looking ahead, as an extension to this work, we propose investigating text-conditioned emoji generation by integrating multimodal models such as CLIP.

2. Dataset Details

To train our emoji generation model, we used the **OpenMoji dataset** as the primary source of image data. OpenMoji (https://openmoji.org/) is an open-source project that provides a comprehensive collection of emoji images in both SVG and PNG formats. It includes over 4,000 emojis in total, but for the scope of our model, we used only the **yellow face smileys** such as 😂🤔😎 etc. These emojis represent expressive human-like facial expressions, and are of uniform shape so that training and inference has good quality.

From the OpenMoji dataset, we picked a set of **108 yellow face smileys**, which capture various expressions such as happiness, sadness, surprise, anger, and more. These emojis were chosen because they form a consistent visual group (round and yellow), and are best for exploring the generation of facial emotions. The images are originally in the PNG format with transparent background, which we later manually converted to have a **black** background. For our training setup, each emoji image is **resized to a standard dimension of 64×64×3**. Some of the emojis in the dataset:

3. Model Overview - DCGAN

For this project, we chose to work with a Generative Adversarial Network (GAN) which is a type of learning model which has two important neural networks: **a generator** and **a discriminator**, which are in a game competing against each other. The generator network tries to produce realistic emojis from random noise, while the discriminator network attempts to differentiate between real emojis and the emojis generated by our generator network. Consequently, the generator network learns to generate highly realistic emojis to trick the discriminator network to classify it as real, while the discriminator also learns to process to identify real and fake emojis more accurately. This adversarial process drives both networks to improve their generation and discrimination tasks, resulting in highly realistic emoji generation.

4. Architecture Details

A. Generator Network

With the help of Generator we transform a low-dimensional latent noise vector into a higher-resolution emoji image. Inorder to perform this upsampling we are using the transposed Convolutional layers.

- **Input**: Input to the generator network is a random noise vector of size **100 × 1 × 1**. This noise is sampled from a normal distribution.

- **Output**: It provides a Emoji image of size 64 x 64 with 3 channels that is sampled from the random noise which is passed as the input.

Generator Architecture Details:

- We are using a 5- Layered generator network consisting of **ConvTranspose2d** operations (upsamping operation).

- With each layer we increase the spatial dimension of image while reducing the channels down to 3 at the output.
  Number of Channels over layers : 512 -> 256 -> 128 -> 64 -> 3

- **Batch Normalization** Along with ReLU activation function is applied at each convolutional layer except the final one.
- **Tanh** activation Function is used to normalize the output (Generated Emoji). and to introduce non-linearity.

B. Discriminator Network

Discriminator network is a CNN based binary classifier which distinguishes real emojis and fake emojis coming from generator. It helps the generator learn to generate more realistic images with the feedback.

- **Input**: Emoji of size 64 x 64 with 3-channels RGB.
- **Output**: Probability of image being real or fake. Value closer to one being real emoji.

Discriminator Architecture Details:

- We are using 5 layers with 4 x 4 kernel size using **Conv2d** operations
- At each layer apart from first and last we are doing Batch normalization followed by LeakyRelu activation function.
- Sigmoid Activation function is used in the final layer.

C. Loss Function

During the training two adversarial networks(generator and discriminator) are optimized using **Binary Cross-Entropy Loss (BCE Loss)**. This loss is used to measure the classification error for classification of real and fake emoji samples.

Discriminator Loss

The discriminator is trained to maximize the probability of correct classifying of emojis. The BCE loss is given as follows:

$$\mathcal{L}_D = -E_{\mathbf{x}\sim p_{data}}\left[\log D(\mathbf{x})\right] - E_{\mathbf{z}\sim p_{\mathbf{z}}}\left[\log\left(1 - D(G(\mathbf{z}))\right)\right]$$

Where:

- D(x) : Discriminator prediction for real image x
- G(z): Generated emoji image from noise z.
- D(G(z)): Discriminator prediction for generated image G(z)

Generator Loss

The generator is trained to generate real looking emojis to trick the discriminator into identifying generated images as real. The Loss is given as

$$\mathcal{L}_G = -E_{\mathbf{z}\sim p_{\mathbf{z}}}\left[\log D(G(\mathbf{z}))\right]$$

The model trains to make D(x) close to 1 and D(G(x)) close to 0.

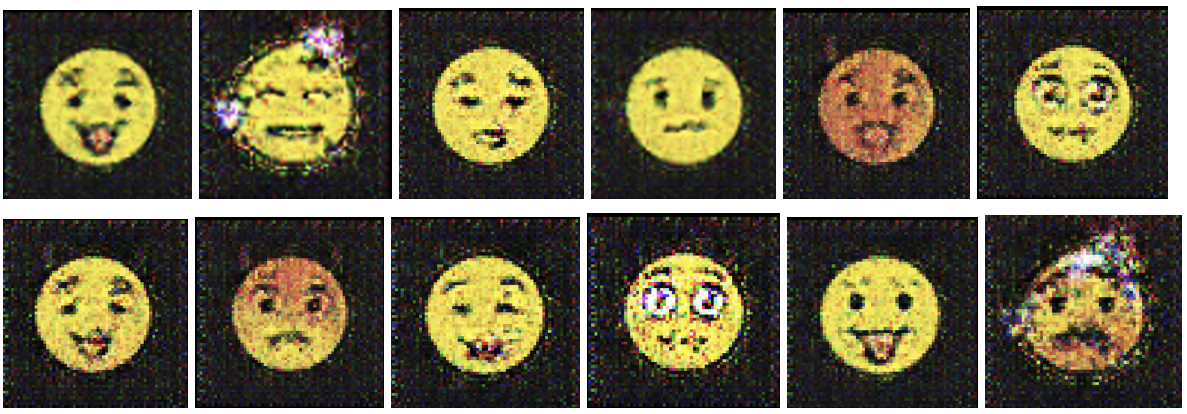
5. Training Details

Dataset	OpenMoji
Image Size	64 x 64
Dataset Length	108
Batch Size	16
Optimizer	Adam (beta1 = 0.4, beta2 = 0.999)
Learning rate	0.00015 (Generator), 0.000105 (Discriminator)
Epochs	500

We had to ensure that the discriminator learns slower than the generator to make sure the gradients to the generator aren't lost due to a very good discriminator. We tried a multitude of learning rate combinations to ensure that the training is good and generator is able to learn to generate good samples.

6. Observations

The GAN (Generative Adversarial Network) model learned different representations of the input and was able to generate some mixed representations of emojis.



Strengths:

1. The generated images combined a broad mix of visual features from the training data, which showed diversity in the output.

2. The model successfully captured high-level features such as the round and yellow (or occasionally red) shapes of emojis, indicating an understanding of global structural patterns.

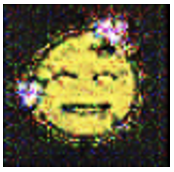
Weaknesses:

1. Significant noise is observable in the generated outputs.

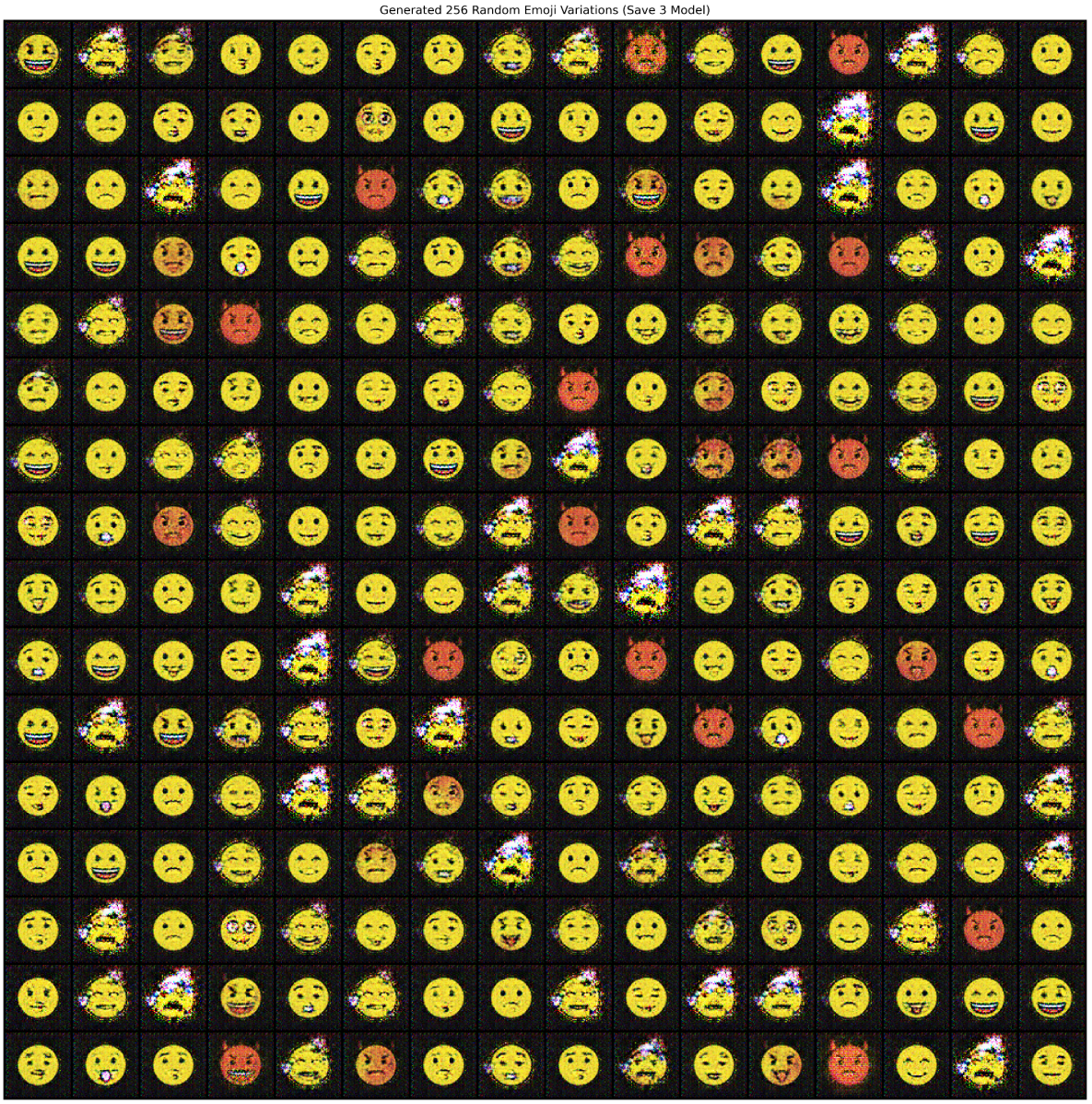
2. Some key features are only partially present, which shows a lack of spatial awareness in the model which can be enhanced with mechanisms like Attention modules or Transformers. For example, the face with clouds emoji:

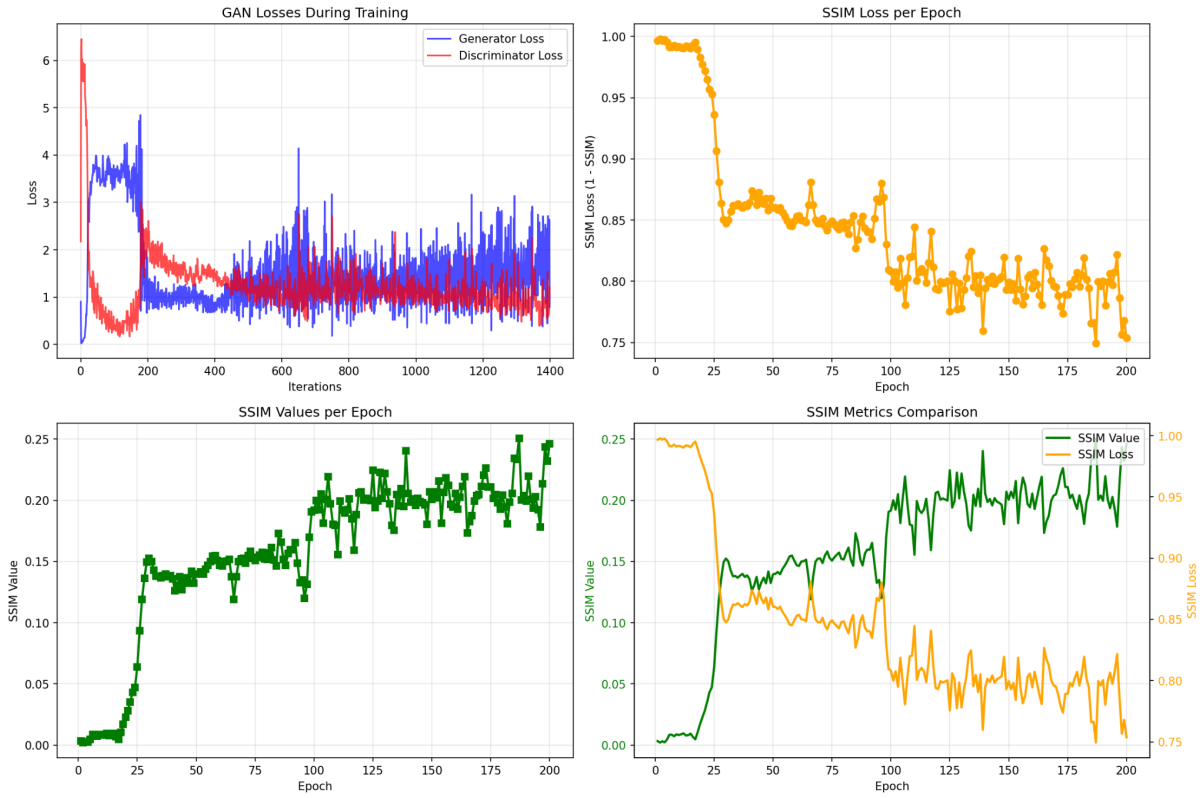
   is not captured entirely and can be seen partially in some emojis:

More samples of the output:



7. Evaluation



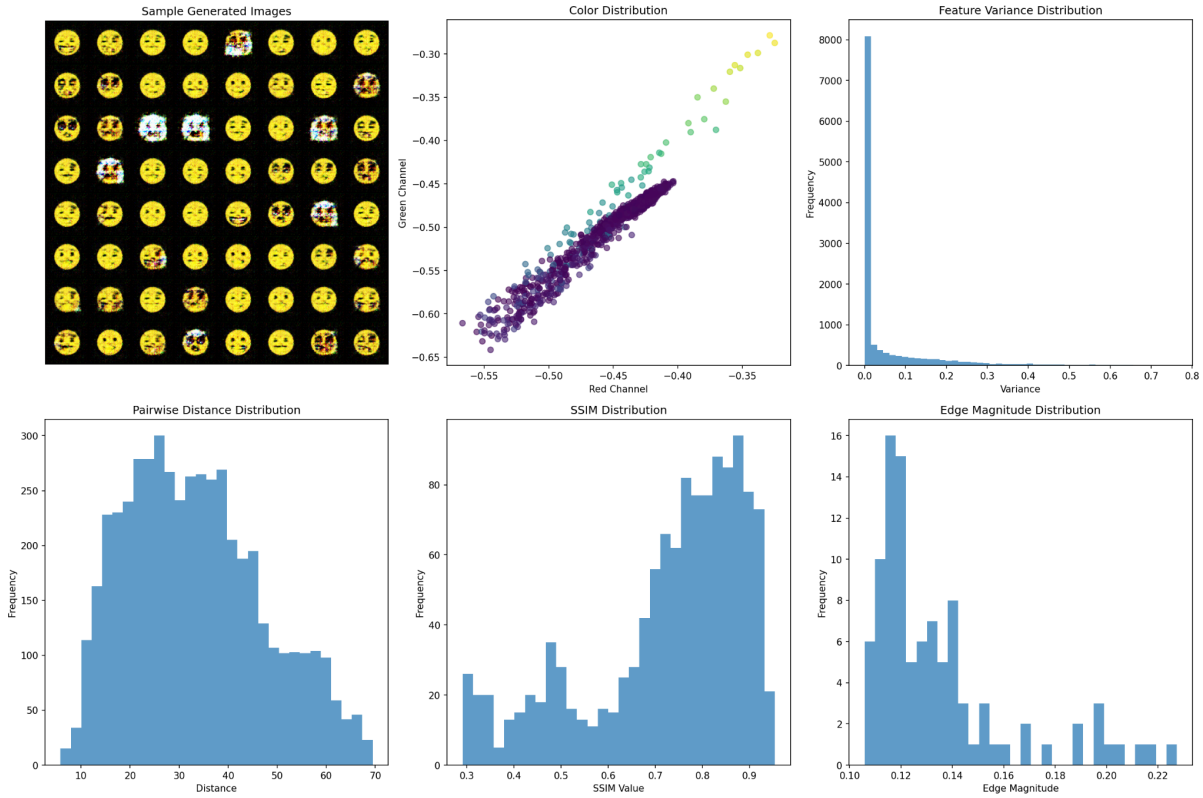
GAN Evaluation: Training Metrics Analysis

GAN Loss Progression:

During training, GAN showed 3 distinct phases. In the early stage (0–200 iterations), the training was highly unstable, with the generator struggling to produce realistic outputs and its loss reaching as high as 6. In the middle stages (200–400 iterations), the process became little more stable, with it's loss reducing to around 1 to 1.5 and the discriminator's loss around 0.5 to 1. By the later stages (400–1400 iterations), the model's both the losses fluctuated between 0.5 and 2.5 but remained under control.

SSIM Quality Improvement:

A similar trend was observed in the SSIM image quality over epochs. The quality improved rapidly in the first 25 epochs, jumping from 0.00 to 0.15. Between epochs 25 and 100, progress slowed and SSIM hovered around 0.14 to 0.15. After epoch 100, the improvements were gradual, with the SSIM score reaching its peak of 0.24 around epoch 140.

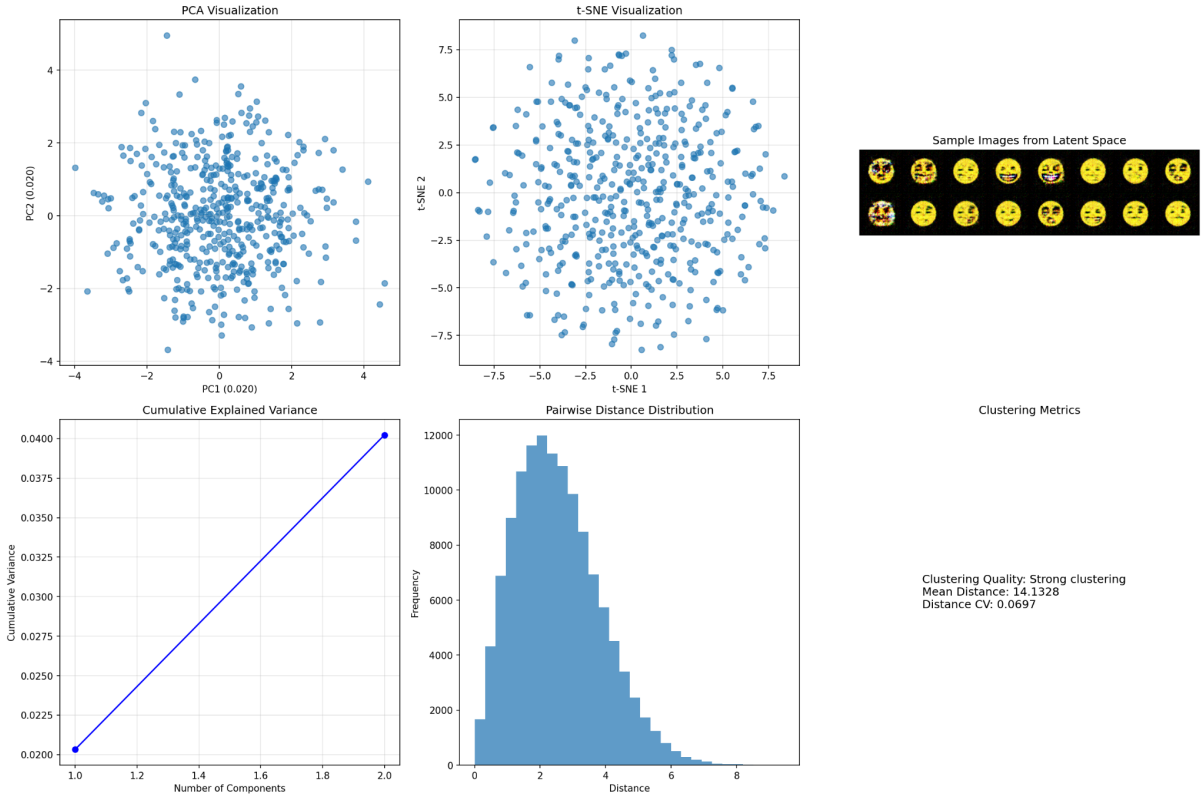


Color Distribution:
The generated images show a dense cluster of colors within a very narrow spectrum, which results in low color diversity, making the outputs appear visually similar. (explains because there are only a few colors - yellow, red, black, white)

Feature Variance:
The variance of features is heavily left skewed, with a high variance at low values. This indicates that the images are mostly similar looking, with very little variation in their patterns.

Pairwise Distance:
The distances between image samples form a bell-shaped distribution, peaking around 25 to 30. This means the images are moderately distinct from each other, showing some controlled variation without being too spread out.

Edge Quality:
The edge magnitudes are relatively low, ranging between 0.11 and 0.12. This produces soft and subtle edges, which can make the images look less sharp and reduce their overall visual clarity. (This can be seen from the generations that the edges have noise and less sharp)



PCA Visualization:
The PCA plot forms a dense circular cluster centered around (0,0). This shows low variance, i.e, PCA has difficulty capturing any complex structure in the data.

t-SNE Visualization:
The t-SNE plot also shows a dense cluster, and the points are slightly more evenly spread, and there are no clear or distinct clusters, suggesting that the samples remain quite similar in feature space.

Distance Distribution:
The distance distribution peaks around 2, with only a few pairs exceeding 6. This pattern shows that most samples are highly similar, and only a small portion exhibit notable differences.

8. Conditional DC-GAN

In the conditional DC-GAN, the Generator is conditioned by concatenating a learned class label embedding to the random noise vector. The class label embedding is used to control the generation of emojis. The Discriminator is also trained along with the class label, allowing it to check if an image is both realistic and a correct match for its given class label..

Model Architecture

Generator:.A 100-dimensional random noise vector is concatenated with a 50-dimensional class embedding. This combined 150-dimensional vector is then passed through a Generator model (five transposed convolutional layers). Each layer is stabilized with Batch Normalization and uses a ReLU activation function. The Tanh activation function is used in the last layer to scale the output pixels(128x128 pixel image).

Discriminator: The class label is converted into an embedding dimension(50) and concatenated with the image along the channel dimension. This combined input is processed by a series of five convolutional layers that downsample the image. Each layer uses Batch Normalization and a LeakyReLU activation. The final layer is passed through a Sigmoid activation to produce a probability score determining the realness of the image and its closeness to the provided label.

Training Details

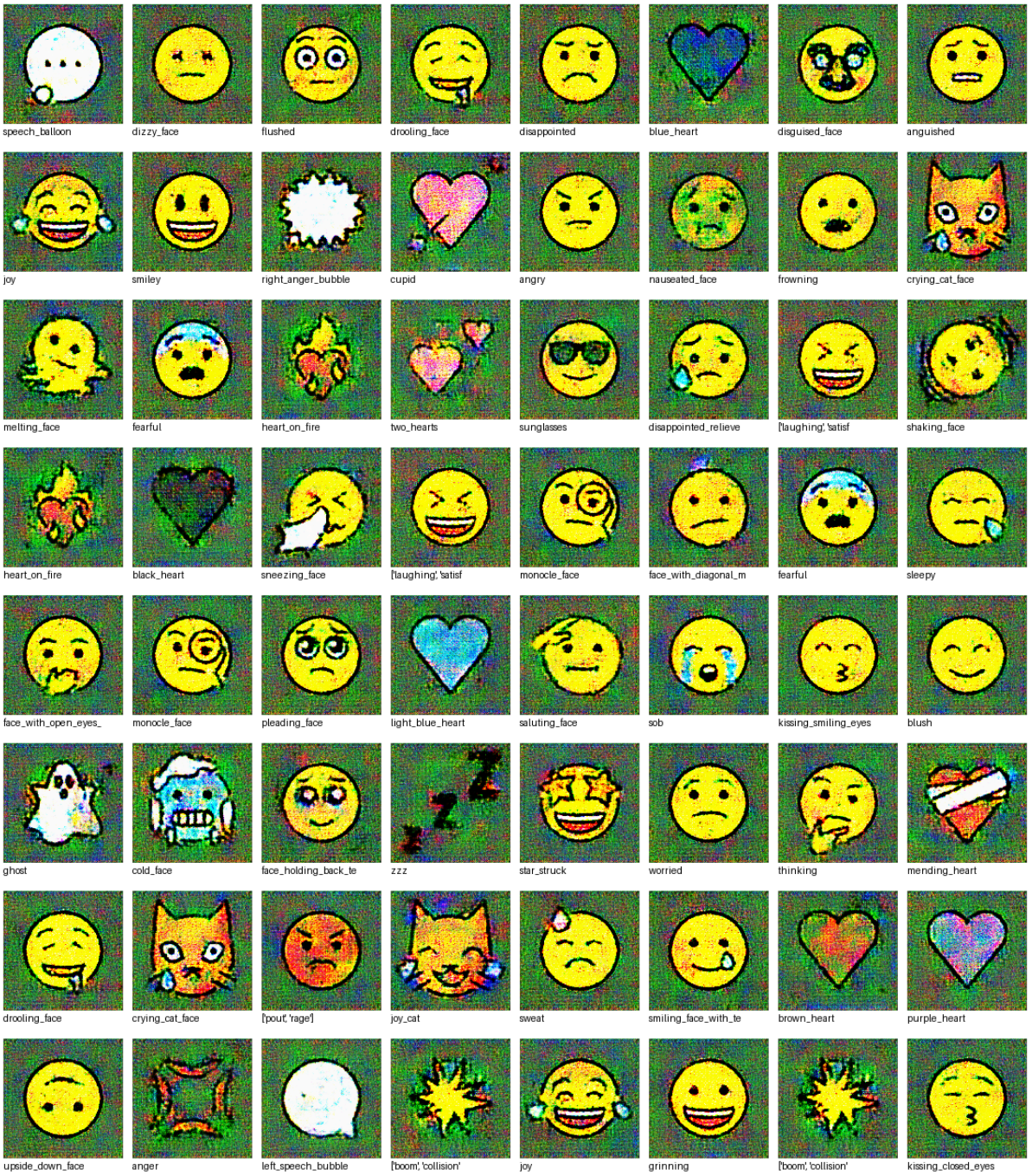
The models were trained using the same subset of emoji images We used 128x128 emoji images and their corresponding text-based "shortcode" labels from a CSV file

Hyperparameters:

- **Optimizer:** Adam
- **Learning Rate (LR):** 4×10^{-4}
- **Batch Size:** 16
- **Number of Epochs:** 200
- **Latent Vector (z) Dimension:** 100
- **Class Embedding Dimension:** 50
- **Loss Function: Binary Cross-Entropy (BCELoss)**
- **Training Enhancements: Label Smoothing** was employed to improve training stability. Instead of using hard labels (1 for real, 0 for fake), "soft" labels were used (e.g., 0.9 for real, 0.1 for fake). This regularization technique prevents the Discriminator from becoming overconfident in its predictions.

Results and Visualization

1) Samples generated after 200 epochs, a fixed set of 64 noise vectors and 64 fixed class labels was passed to the Generator.



2) Generating Emoji sample based on label input:

Kissing Heart Emoji

Winking Emoji

Crying Cat Face

3) Interpolating between Two emojis in latent space

a) Interpolating between Heart Eyes and Big Roll Eyes

b) Interpolating between Smiley and star struck emoji

9. Future Work

Though the model was able to generate a various combination of features, below are the things that can be improved as an extension for this project:

- Text based conditional generation - to generate the emoji based on a prompt.
- Add Attention blocks to capture spatial awareness to the models.
- Add more emojis to the training data - such as humanoid emojis, Hand emojis etc.