Emoji Generation with VQ-VAE

Team ID: 14
Team Members:
Akash Gorakh Chaudhari - MT2024012
Shardul Sisodiya - MT2024140
Course: Generative AI For Vision
Instructor: Prof. Viswanath Gopalakrishnan
Code Repo: https://github.com/chaudhari-akash/Emoji-Generation-with-VQ-VAE

Abstract

This project explores discrete representation learning and generative image modeling using the Vector Quantized Variational Autoencoder (VQ-VAE) combined with a Transformer-based prior. First, we train a VQ-VAE model to encode emojis into a discrete latent space represented by learned codebook vectors. Next, we train a Transformer model to autoregressively model the distribution of these embeddings, enabling sampling of entirely new emoji representations. Additionally, the model enables latent interpolation, which allows smooth morphing between two emojis in latent space to visualize the learned semantic transitions. The system is capable of reconstructing emojis and demonstrating continuous latent structure through interpolation. Experimental evaluations show strong reconstruction quality, demonstrating the effectiveness of discrete latent modeling for visual synthesis.

1 Introduction

VQ-VAE provides a framework for converting high-dimensional images into compact discrete latent representations. These discrete representations can then be modeled using sequence models such as Transformers, treating image synthesis analogously to natural language modeling. This project applies this idea to emoji images, leveraging their structured yet varied expressive features.

The project objectives were:

1. Train a VQ-VAE to compress and reconstruct emoji images.
2. Train a Transformer prior to learn a generative distribution over the discrete latent codes.
3. Generate novel emojis by sampling and decoding latent sequences.
4. Perform latent space interpolation to demonstrate semantic continuity.

2 Dataset and Preprocessing

The dataset used in this project is the HuggingFace dataset `valhalla/emoji-dataset`, containing approximately 2700 emoji images with accompanying text labels.

Data Preparation

In the first part of the project, we implemented the following processing steps (as documented in the first notebook):

- **Image Resizing**: All images were uniformly resized to 64×64 resolution to standardize input dimensions.
- **Tensor Conversion**: Images were converted to PyTorch tensors and normalized to the range $[0, 1]$.
- **Batch Loading**: PyTorch `DataLoader` was used with mini-batches of size 128 for efficient GPU training.

3 VQ-VAE Architecture and Training

3.1 Model Architecture (Part A Implementation)

The VQ-VAE model implemented in the first phase consists of three primary components: an encoder, an EMA-based vector quantizer, and a decoder. The encoder maps an input emoji image into a latent feature map of size $(256, 16, 16)$, which is discretized using a learnable codebook of 512 vectors. The decoder reconstructs the image from quantized latent embeddings.

The quantizer uses Exponential Moving Average (EMA) updates to adjust the codebook vectors during training instead of updating them via gradient descent. This improves training stability and prevents embedding collapse.

3.2 Codebook Collapse and Reset Strategy

In early VQ-VAE training, a common issue is **codebook collapse**, where only a small subset of codebook embeddings are used while others never get selected. To prevent this, we incorporated a **codebook reset mechanism** that periodically resets unused embeddings.

The model maintains a usage counter for each codebook vector. Every **5 epochs**, the following steps are executed:

1. Identify embeddings with zero usage since the last reset.
2. Sample latent vectors from a recent training batch.
3. Replace unused codebook vectors with randomly selected latent vectors.
4. Reset usage counters.

This ensures that the codebook continues to explore meaningful latent directions, avoiding collapse and encouraging diversity.

Observed Effect During Training

During initial epochs, codebook usage rose to approximately 87% before reset. After the first reset, usage temporarily dropped (as many embeddings were reinitialized), followed by rapid recovery approaching nearly 100% utilization. This cyclical pattern repeated until stabilization.

For example:
Epoch 5: Codebook usage = 87.30% (before reset)
Epoch 6: Codebook usage = 17.77% (after reset)
Epoch 11: Codebook usage = 99.41% (recovered and stabilized)

This confirms that the reset mechanism successfully prevents collapse and promotes full codebook utilization, enabling the Transformer prior (Part B) to learn a richer latent distribution.

3.3 Training Details

The model was trained for 200 epochs using the AdamW optimizer with a learning rate of 2×10^{-4} and cosine annealing. Gradient clipping was applied for stability.

4 Reconstruction and Codebook Usage Analysis

4.1 Reconstruction Quality

The VQ-VAE demonstrates strong reconstruction capabilities, retaining both color and structural emoji characteristics.

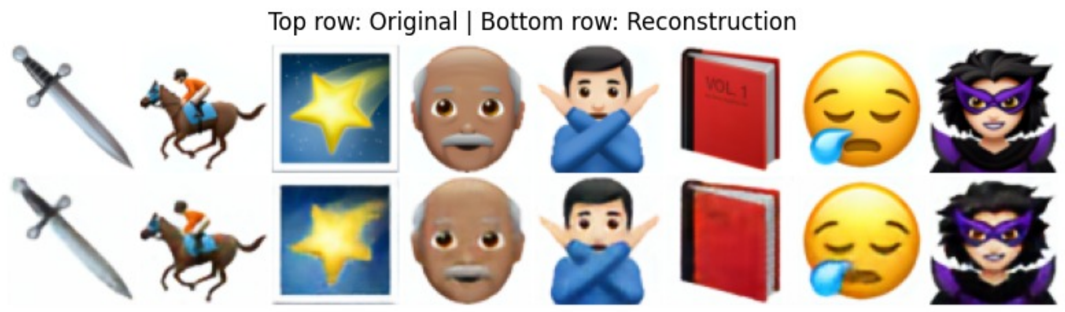


Figure 1: Original emojis (top) and reconstructed emojis (bottom).

4.2 Codebook Usage Monitoring

To ensure that the codebook embeddings were actively utilized and to prevent embedding collapse, we monitored the percentage of codebook vectors that were assigned at least once per training epoch. The usage percentage increased over time and stabilized near full utilization after the periodic reset strategy was introduced.

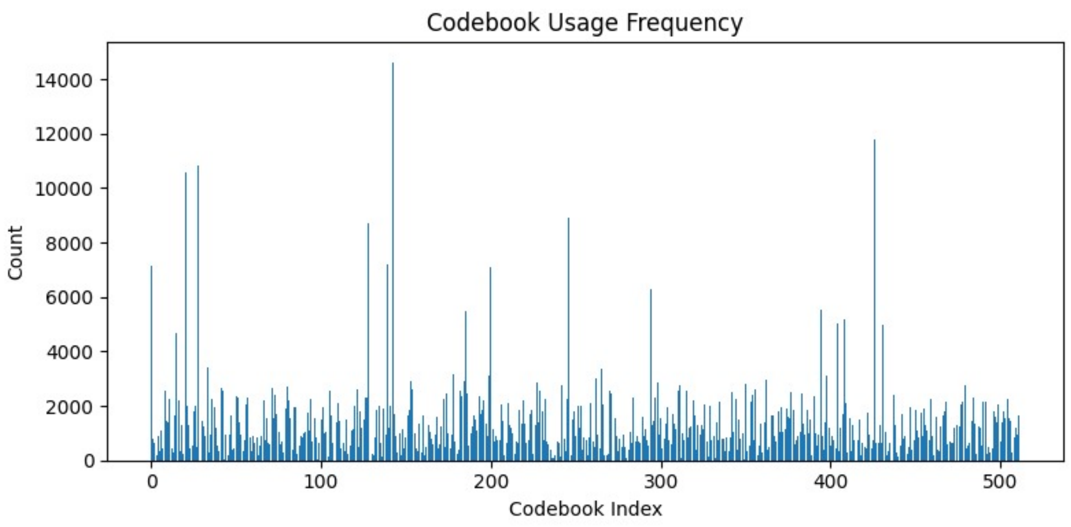


Figure 2: Codebook usage frequency across training epochs. The periodic drop corresponds to the reset interval, followed by rapid recovery as embeddings are reused.

5 Transformer Prior and Sampling (Part B)

Once the VQ-VAE was trained, each (16×16) grid of codebook indices was flattened into a sequence of 256 discrete tokens. A GPT-style causal Transformer was trained to model sequences autoregressively. After training, new emojis were generated by sampling token sequences and decoding them back through the VQ-VAE decoder.

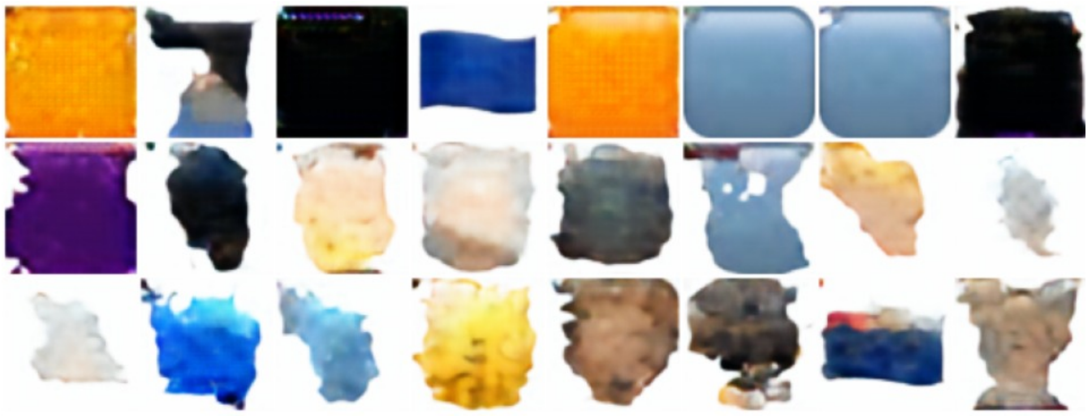


Figure 3: Newly generated emojis sampled from the Transformer prior.

5.1 Latent Interpolation (Part C)

To demonstrate the continuity and semantic structure of the learned latent space, we performed interpolation between two emojis:

- "Smiling Face with Smiling Eyes"
- "Smiling Face with Heart-Shaped Eyes"

First, each emoji image was passed through the encoder of the trained VQ-VAE model to obtain its latent representation. We denote these latent representations as:

$$\text{latent1} = \text{Encoder(smiling eyes emoji)}, \qquad \text{latent2} = \text{Encoder(heart eyes emoji)}$$

These latent tensors represent the compressed feature spaces of the respective emojis. To visualize how the model transitions between these two expressions, we performed linear interpolation in latent space.

We defined a list of interpolation step counts:

$$\text{interpolation\_steps\_list} = \{10, 5\}$$

For each chosen number of steps, we generated a series of intermediate latent vectors using the standard linear interpolation formula:

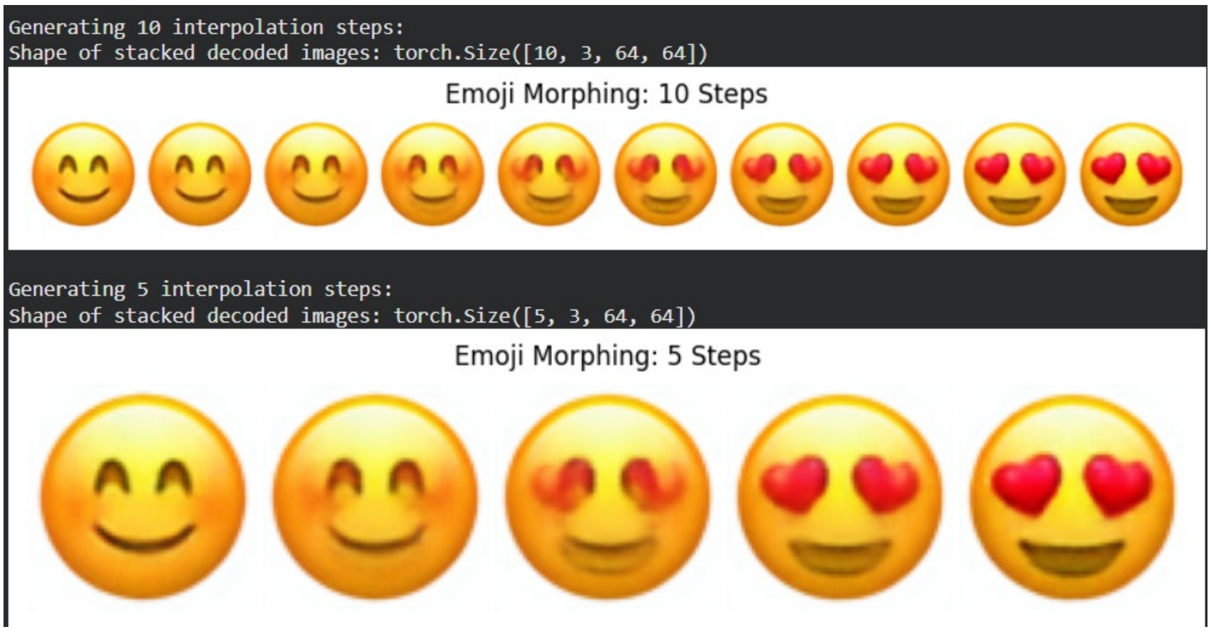
$$\text{latent}(t) = (1 - t) \cdot \text{latent1} + t \cdot \text{latent2}, \qquad t \in [0, 1]$$

where t increases evenly from 0 to 1 across the interpolation steps. When $t = 0$, the resulting representation equals `latent1`; when $t = 1$, it equals `latent2`. Intermediate values of t yield latent points that lie along the direct path between the two emojis in latent space.

Each interpolated latent tensor was then passed through the VQ-VAE decoder to reconstruct an image:

$$\hat{x}(t) = \text{Decoder}(\text{latent}(t))$$

This produced a sequence of images that smoothly transition from the smiling-eyes emoji to the heart-eyes emoji. The results clearly show gradual changes in expression and eye shape rather than abrupt jumps, indicating that the latent space learned by the VQ-VAE is **semantically meaningful and continuous**.



```
Generating 10 interpolation steps:
Shape of stacked decoded images: torch.Size([10, 3, 64, 64])
```

```
Generating 5 interpolation steps:
Shape of stacked decoded images: torch.Size([5, 3, 64, 64])
```

Figure 4: Latent interpolation between two emojis showing semantic continuity.

6 Quantitative Evaluation

Metric	Result
MSE (Reconstruction)	0.001424
PSNR (dB)	28.472 dB
SSIM	0.938347

Table 1: Reconstruction Evaluation Metrics

7 Discussion

The model demonstrates several strengths, most notably the VQ-VAE's ability to learn compact and meaningful discrete latent representations that preserve the core structure and expressive features of emojis. The reconstruction quality is strong, and the periodic codebook reset strategy effectively prevents collapse and maintains high embedding utilization. Furthermore, latent interpolation reveals smooth semantic transitions between expressions, indicating a well-structured latent space. However, the system also has limitations: reliance on an MSE-dominant reconstruction objective can result in slightly blurred textures. The relatively small dataset constrains diversity in generated samples, and sampling quality is sensitive to hyperparameters such as temperature and top-k/p. Despite these limitations, the project provides valuable insights, showing that discrete latent modeling bridges visual synthesis and sequence-based generative modeling, and that meaningful semantic structure can emerge without explicit supervision. The model's interpolation behavior, in particular, suggests that the latent space captures expressive variation rather than simply memorizing pixel patterns.

8 Conclusion

The combined VQ-VAE and Transformer architecture successfully models the emoji domain, demonstrating robust reconstruction, and coherent interpolation. Future extensions include improving the generation quality of novel emojis, emotion-conditioned generation and cross-style emoji blending.