EVALUATION METRICS FOR GENERATIVE AI & PROJECT DISCUSSION

Biplab Das
biplab.das@iiitb.ac.in

Lecture Notes

Contents

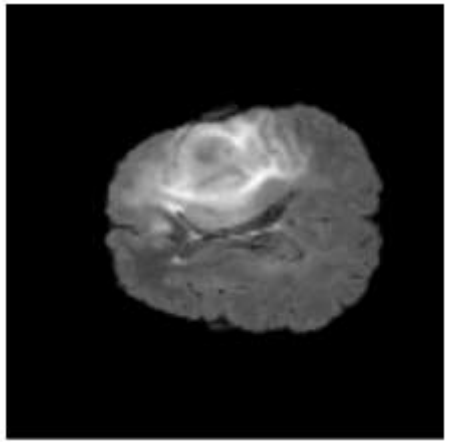
1. **Evaluation Metrics for Generative AI (Images)**
2. **VQVAE2 Overview.**
3. **Straight Through Estimator & Discussion on Prior Learning**
4. **Project Discussion and Guidelines.**

Problem in evaluated image generation

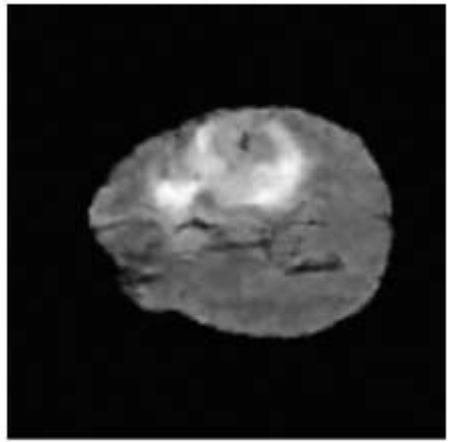
Evaluating generated images needs to be based on two main factors:

- **Image quality** – The generated images must be of high quality and similar to the dataset.

- **Diversity** – The Generator should be able to produce many different images belonging to various classes. If it only generates a few images or images from the same class repeatedly, then it's not very meaningful.



Ground Truth



Generated image

FID-Frechet Inception Distance

Compares the distribution of generated images with the distribution of a set of real images.

A lower FID indicates better-quality images (o is perfect!).

$$d_F\left(\mathcal{N}(\mu,\Sigma),\mathcal{N}(\mu',\Sigma')\right)^2 = \|\mu-\mu'\|_2^2 + \mathrm{tr}\left(\Sigma+\Sigma'-2(\Sigma\Sigma')^{\frac{1}{2}}\right)$$

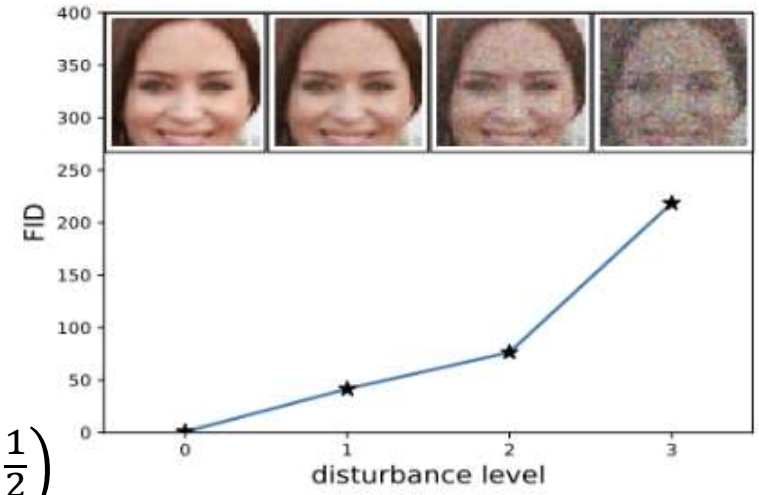


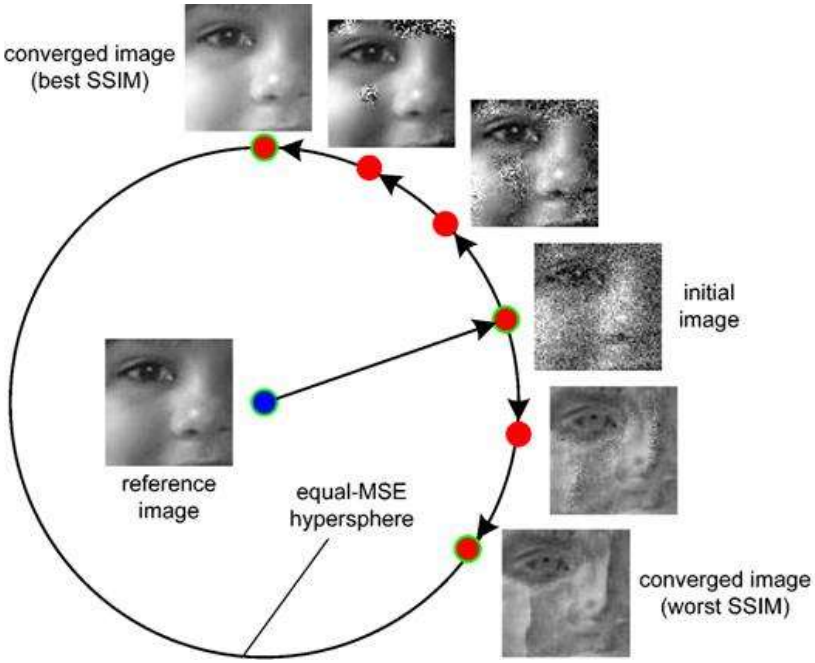
Fig 1: Example of How Increased Distortion of an Image Correlates with High FID Score.[1]

SSIM-Structural Similarity Index Measurement

Based on three parameters for comparison

- Luminance
- Contrast
- Structure

$$\mathrm{SSIM}(\mathrm{x},\mathrm{y}) = [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma}$$



SSIM-Structural Similarity Index Measurement

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2}$$

$$s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3}$$

The SSIM for each block is then a weighted combination of those comparative measures:

$$\mathrm{SSIM}(x,y) = l(x,y)^{\alpha} \cdot c(x,y)^{\beta} \cdot s(x,y)^{\gamma}$$

PSNR: Peak Signal-to-Noise Ratio

PSNR is used to calculate the ratio between the maximum possible signal power and the power of the distorting noise

PSNR is measured in decibels

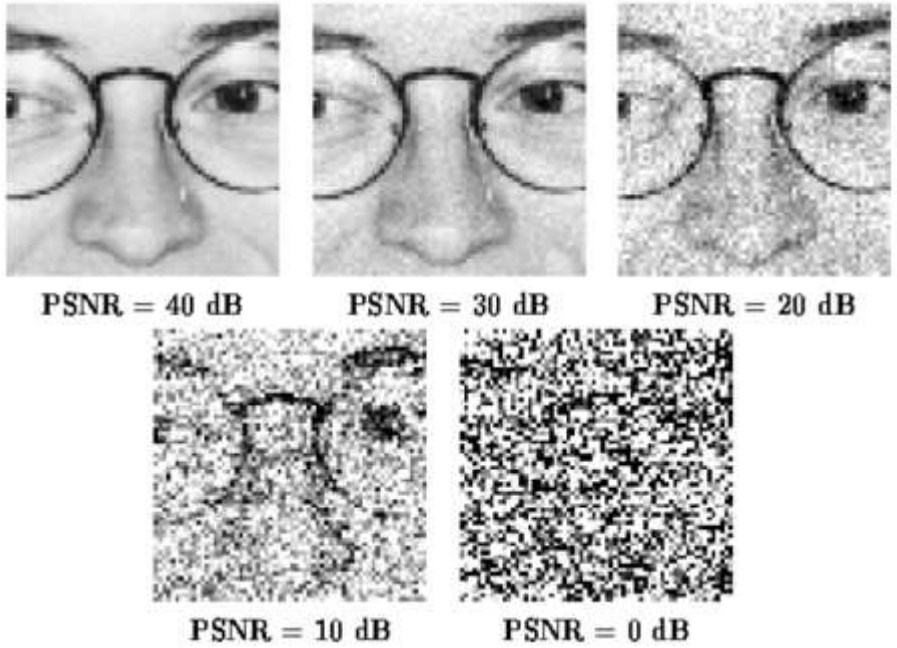


Fig 3. Illustration of the PSNR measure[3]

PSNR: Peak Signal-to-Noise Ratio

The PSNR is usually calculated as the logarithm term

$$\mathrm{PSNR} = 10\log_{10}((\mathrm{peakval}^{\ 2})/\mathrm{MSE})$$

Here, peakval (Peak Value) is the maximum in the image data

Ex: 8-bit unsigned integer data type, the peakval is 255

Mean Squared Error (MSE) between two images such as $g(x, y)$ and $\hat{g}(x, y)$ is defined as:

$$\mathrm{MSE} = \frac{1}{MN}\sum_{n=0}^{M}\sum_{m=1}^{N}[\hat{g}(n,m) - g(n,m)]^2$$

References for Metrics

[1] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*. arXiv. https://arxiv.org/abs/1706.08500

[2] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh and Eero P. Simoncelli, (2011,Feb. 11), The SSIM Index for Image Quality Assessment, https://www.cns.nyu.edu/~lcv/ssim/

[3] *Todd Veldhuizen , 1998 ,* Measures of image quality https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/VELDHUIZEN/node18.html

VQVAE-2 OVERVIEW

Main Ideas:

1) **Vector Quantized Variational AutoEncoder.**

2) **Learning the codebook.**

3) **Hierarchical VQ-VAE.**

4) **Learning priors over the latent codes.**

Hierarchical VQ-VAE and Codebook Learning

In the first stage, a hierarchical autoencoder with multiple stacked vector quantized variational autoencoders (VQ-VAE) is trained on the dataset. The latent spaces on each level are compressed by quantization.

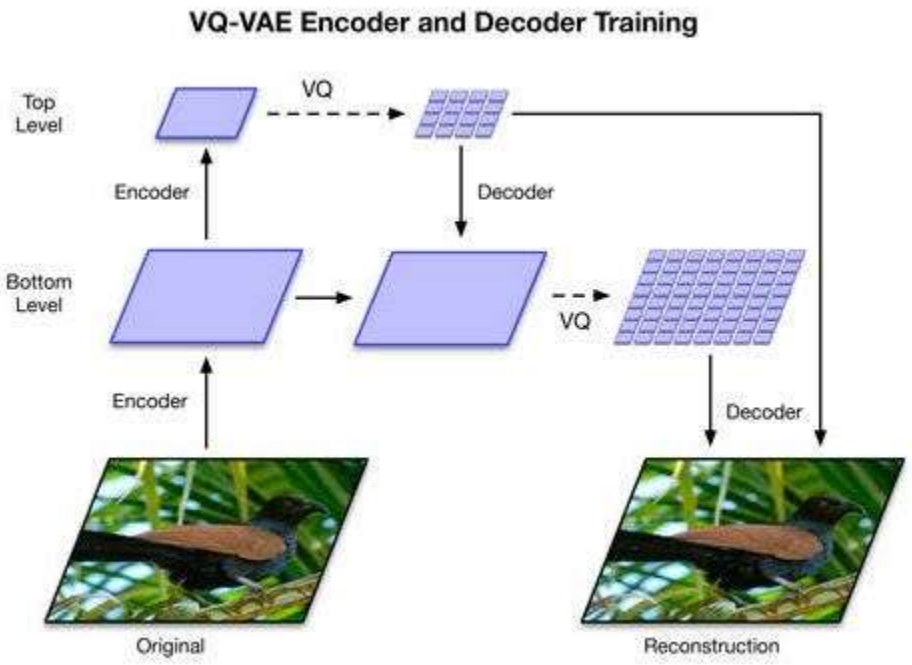
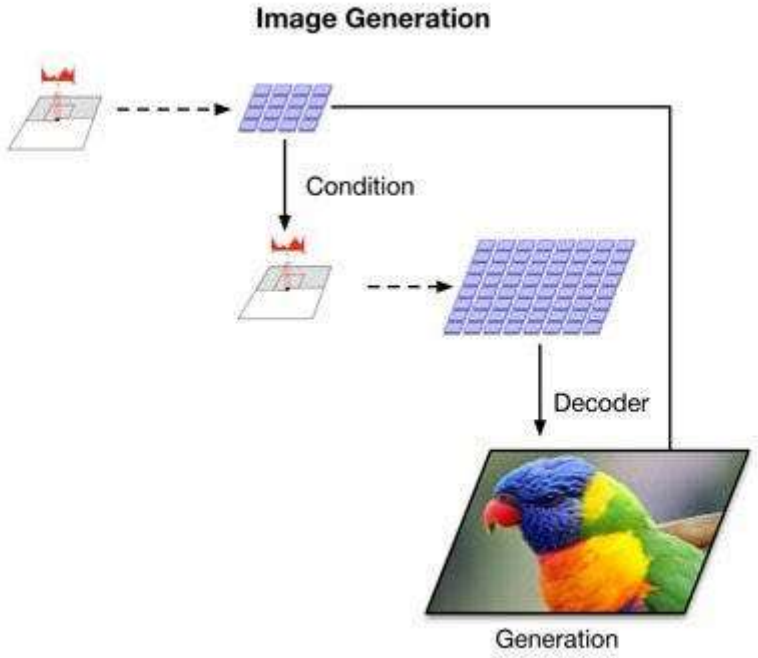


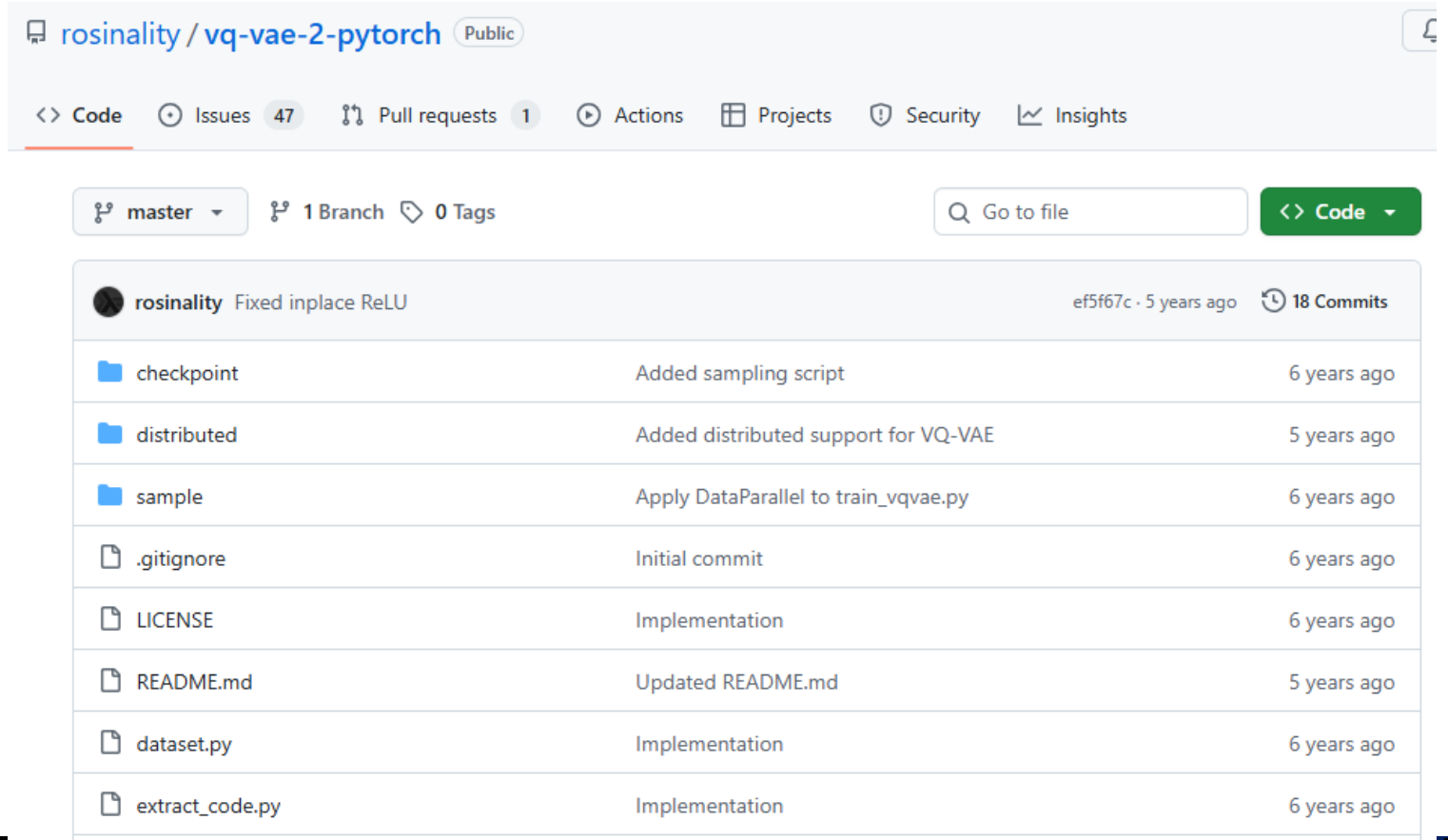
Image Generation

In a second stage for each level in the hierarchy, a PixelCNNs is trained. The PixelCNN only models the latents, allowing it to spend its capacity on the global structure and most perceivable features. In the lowest hierarchical level, the PixelCNN is conditioned on the class. In the higher levels, the PixelCNNs are conditioned on the previous quantized latent map.



Reference Implementation of VQ VAE-2

[vq-vae-2-pytorch/vqvae.py at master · rosinality/vq-vae-2-pytorch · GitHub]



Straight Through Estimator

Straight-Through Estimator in VQ-VAE

In vector quantization, the challenge is that the quantization step (choosing the nearest embedding) is **non-differentiable**. To allow gradients to flow through this discrete operation during backpropagation, we use the **straight-through estimator (STE)**.

In this code, the STE is implemented in this line inside the `Quantize.forward()` method:

Python Copy

```
quantize = input + (quantize - input).detach()
```

Learning Prior

🧠 What Does "Learning a Prior" Mean in VQ-VAE?

In VQ-VAE, the encoder maps input data (e.g., images) to **discrete latent codes** from a learned codebook. These codes are great for compression, but to generate new data, we need a **prior** — a model that tells us how likely each code is, and how they relate to each other spatially.

PixelSNAIL is used to **model the distribution of these codes**:
$P(z) = \prod_{i,j} P(z_{i,j} \mid z_{<i,j})$

This means it predicts each code $z_{i,j}$ conditioned on all previous codes in raster scan order.

Learning Prior

How PixelSNAIL Learns the Prior

1. Training Setup

- After training the VQ-VAE, you encode your dataset into discrete code indices (e.g., `id_t`, `id_b` from `VQVAE.encode()`).
- These indices form a grid of integers — essentially an image of codebook entries.
- PixelSNAIL is trained to **predict the next code index** given previous ones, using cross-entropy loss.

2. Input to PixelSNAIL

- The input is a 2D tensor of code indices (e.g., shape `[B, H, W]`).
- These are converted to one-hot vectors and passed through causal convolutions to preserve autoregressive ordering.

Learning Prior

3. Conditioning (Optional)

- You can condition PixelSNAIL on top-level codes (`id_t`) when modeling bottom-level codes (`id_b`).
- This is done via the `condition` argument and `CondResNet`, allowing hierarchical modeling.

4. Output

- PixelSNAIL outputs logits over the codebook size (`n_class`), predicting the probability of each code at each position.
- During training, you compare these logits to the true code indices using cross-entropy.

Sample from the Prior

Sampling from the Prior

Once trained, you can **sample new code grids** from PixelSNAIL:

1. Start with an empty grid.
2. Fill in each position sequentially using the predicted distribution.
3. Decode the sampled codes using `VQVAE.decode_code()` to generate new images.

This gives you **high-quality, diverse samples** that respect the learned structure of the latent space.

Project Guidelines

Details in the attached document

Project Guidelines: Emoji Generation with VQ-VAE

Generative AI Course Project

Part A: Core (Required)

1. Dataset Preparation

- Collect a dataset of emojis (minimum ~2500 images; e.g., Unicode emojis, custom smiley sets, or open datasets).
- Preprocess to uniform size (e.g., 32×32 or 64×64).
- https://huggingface.co/datasets/valhalla/emoji-dataset

2. Model Training

- Train a VQ-VAE for reconstruction.
- Report reconstruction quality (MSE, SSIM, FID).
- Visualize original vs. reconstructed emojis.

Thank You