Assignment 5: VQ VAE

Soham Padhye

April 13, 2025

1 Introduction

Train a Vector-Quantized Variational Autoencoder (VQ-VAE) on the skin lesion dataset to efficiently encode and decode high-dimensional image data while capturing meaningful latent representations.

Train an Auto Regressive Model of your choice to generate new, realistic images based on the learned latent space representations.

2 Dataset

ISIC 2016 dataset. This dataset has 9015 training images. In this task we used only images to train the model. We also tried trainin the VQ-VAE on sketches but results was not good. We have added both results in the report further.

3 Experimental setup

Used MIL lab elecrical engg dept. PC with NVIDIA GPU. Training of VQVAE took around 5 hours and training of PixelCNN took around 4 hours.

4 Code

In the below code file we have attached the VQVAE code file.

Google colab code link

In the below code file we have attached the VQVAE and pixelcnn code file.

Google colab code link

5 Methodology

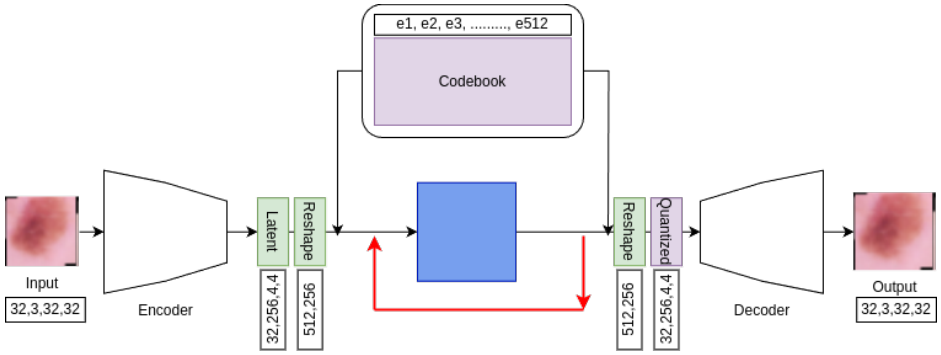

Figure 1: Methodology

6 Pre-Processing

6.1 Data Loader

1. Create the custom dataloader to load the images.
2. Define transform to resize the image and convert it to tensor.

6.2 Augmentation

1. Apply augmentation technique for increasing the size of the dataset and making model to learn the good generalization.
2. Apply augmentation technique like random crop, horizontal flip, rotate, color jitter and Gaussian noise transform as an augmentation.
3. **Random Horizontal Flip**: This flips the image horizontally with a certain probability. This augmentation is especially useful for symmetric objects.
4. **Color Jitter**: This augmentation randomly adjusts brightness, contrast, saturation, and hue of the image. This augmentation introduces variability in color, making the model more robust to different lighting conditions.
5. **Random Resized Crop**: Randomly crops and resizes the image. It's particularly useful for data augmentation as it combines cropping and resizing, allowing the model to learn from various scales and viewpoints of the objects.
6. **Random Rotation**: This augmentation rotates the image by a random angle within a specified range. This helps the model generalize better to rotated versions of the objects in the images.
7. Sample images loaded from the dataloader with augmentation are as shown below

Figure 2: Training images with augmentation

7 Train a Vector-Quantized Variational Autoencoder

For this task use VQ-VAE 2 architecture for generating the realistic images of the disease.

7.1 VQ-VAE

- **Variational Autoencoder (VAE)**: The VAE is a type of neural network architecture designed for learning latent representations of data, particularly for generative modeling tasks. In this case we use continuous data distribution at latent representation.
- **Vector Quantization (VQ)**: In VQ-VAE the latent space representation is discretized using a technique called vector quantization using codebook.
- **Codebook**: The codebook contains a fixed set of embedding vectors that are used as the quantized representations of the input data. Codebook is initially randomly initialized. It get trained while training VQVAE.
- **Codebook loss and commitment loss**: The VQ-VAE model incorporates two additional terms in its objective to align the vector space of the codebook with the output of the encoder. The codebook loss, which only applies to the codebook variables, brings the selected codebook e close to the output of the encoder, E(x). The commitment loss, which only applies to the encoder weights, encourages the output of the encoder to stay close to the chosen codebook vector to prevent it from fluctuating too frequently from one code vector to another

7.2 VQ-VAE-2

The VQ-VAE-2 (Vector Quantized Variational Autoencoder 2) is an advanced version of the original VQ-VAE architecture, which is a type of autoencoder neural network used for generative modeling tasks, particularly in image generation and compression.

- As opposed to vanilla VQ-VAE, in this they used a hierarchy of vector quantized codes to model large images. The main motivation behind this is to model local information, such as texture, separately from global information such as shape and geometry of objects. The prior model over each level can thus be tailored to capture the specific correlations that exist in that level.
- The structure of our multi-scale hierarchical encoder is illustrated in Fig. 7, with a top latent code which models global information, and a bottom latent code, conditioned on the top latent, responsible for representing local details.

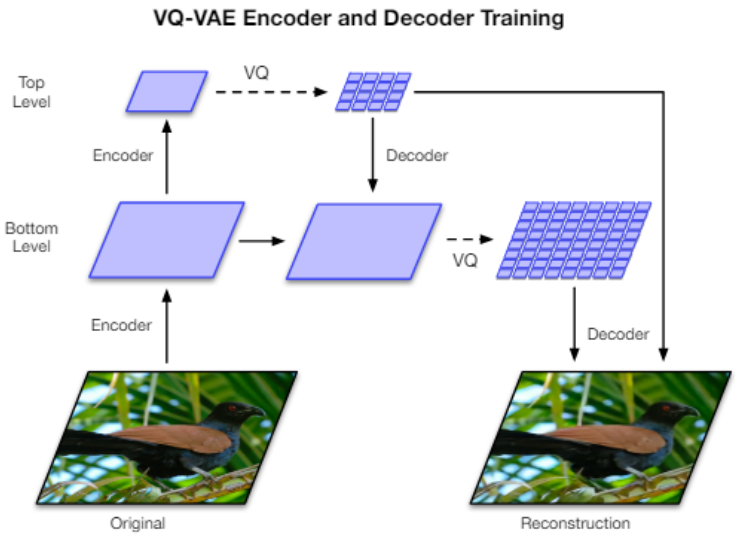

Figure 3: Training images and mask

8 Model Training

1. Iterating over the dataset in batches. Batch sieze is 32 in our case
2. Than pass each batch of images through the encoder to obtain the latent representations.
3. After this reshape (B*H*W, C) the latent representations and calculate the distances between the codebook vectors and the latent representations.
4. Map each latent representation to the nearest codebook vector. Code is random;y initialized.
5. Reshape (B,C,H,W) the quantized latent representations.
6. Pass the quantized latent representations through the decoder to reconstruct the images.
7. Calculate the reconstruction loss and the vector quantization loss.
8. Last update the parameters of the encoder, decoder, and codebook using backpropagation to minimize the total loss.

8.1 Loss function

- Loss function plays an important role in the model optimization. we tried MSE loss function. Calculating the MSE loss between the generated image and groundtruth image.
- As training progress the loss was decreasing and generated images was good so we fixed this loss as it is good loss for image reconstruction.

9 Plots and observations

9.1 Training loss

Figure 4: Training loss

From the figure 4 of loss versus the number of epochs, it can be observed that training decreases as the number of epochs increases. This kind of behavior indicates that the model is continuously improving over the training data and it has ability to generalize well to unseen validation data.

But from the graph of loss vs no. of epochs it can be seen that after 15 epochs the model is converged. So we trained the model for 30 epochs to see any further improvement.

9.2 Hyper parameter tuning

- In the above fig 4 the red line is for SGD optimizer and Blue curve is for ADAM optimizer. From the curves it can be seen that there is sharp jumo in SGD. But sfter 7 epochs both optimizers converged. We fixed ADAM optimizer for training.
- We tried different learning rates from 0.03 to 3e-4. We were getting best resuts for 3e-4 and fluctuating results on 0.03 learning rate. So we fixed the learning rate to 0.0003 value.

10 Results

With above mentioned setup we have run the model for 30 epochs to train the VQ-VAE. The generated images are as shown below. First row denotes the groundtruth images and second row denotde the generated images from VQ-VAE-2.

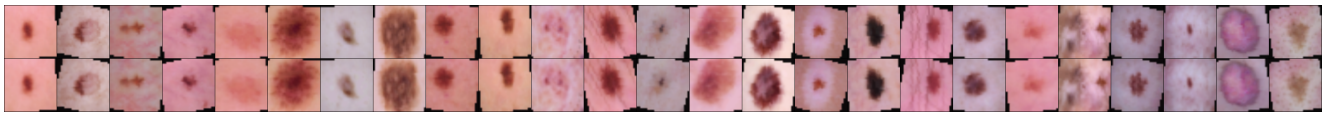

Figure 5: Results for VQ-VAE

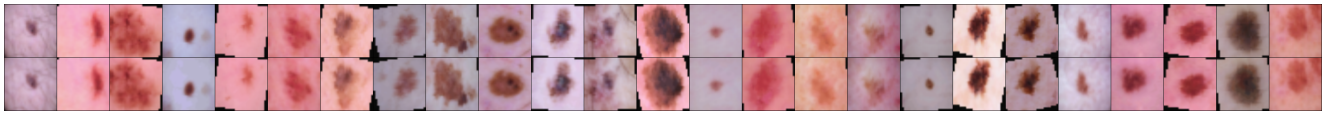

Figure 6: Results for VQ-VAE

11 PixelCNN to improve the prior

1. Stage 1: Learning Hierarchical Latent Codes from encoder
2. Stage 2: Learning Priors over Latent Codes. So in this section we will discuss more about this stage.

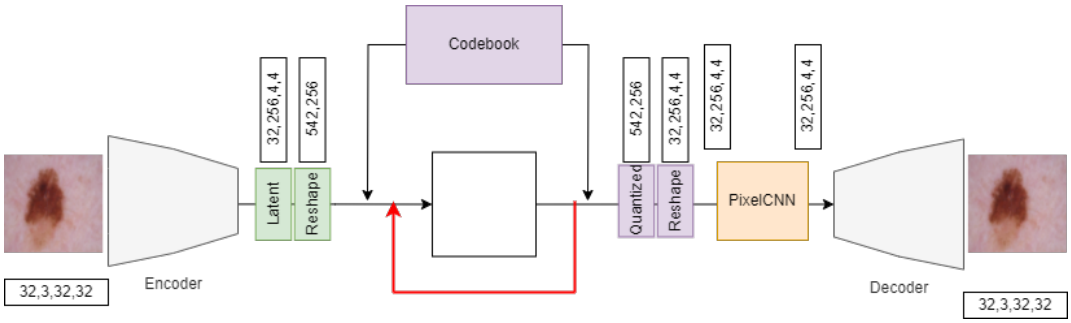

Figure 7: VQVAE-2 with pixel CNN

In this section we will learn a prior over the latent codes and then pass it to the decoder. This procedure also reduces the gap between the marginal posterior and the prior. Thus, latent variables sampled from the learned prior at test time are close to what the decoder network has observed during training which results in more coherent outputs.

In this task auxiliary prior is modeled with a powerful autoregressive neural network PixelCNN. The process of fitting a prior to the learned posterior can be considered as lossless compression of the latent space by re-encoding the latent variables with a distribution that is a better approximation of their true distribution,

12 PixelCNN training

12.1 Training loss

Figure 8: Training loss for VQVAE and pixelCNN

1. From the figure of loss versus the number of epochs, it can be observed that training lose first increase till 0.04 than start decreases as the number of epochs increases which indicate model is adopting the dataset and then decrease represent that after every epoch over model starting to saturate This kind of behavior indicates that the model is continuously improving over the training data.

   But from the graph of loss vs no. of epochs it can be seen that after 15 epochs the model is converged. So we trained the model for 30 epochs to see any further improvement.

13 Results from PixelCNN

With above mentioned setup we have run the model for 30 epochs to train the PixelCNN. While training the PixelCNN we freeze the decoder and codebook also. We have achieved this by loading the pretrained weights from VQ-VAE2 model trained earlier. We used pytorch Detach() function and detached the decoder and codebook from computational graph of pytorch.

The generated images are as shown below. First row denotes the groundtruth images and second row denote the generated images from PixelCNN and pretrained VQ-VAE-2.

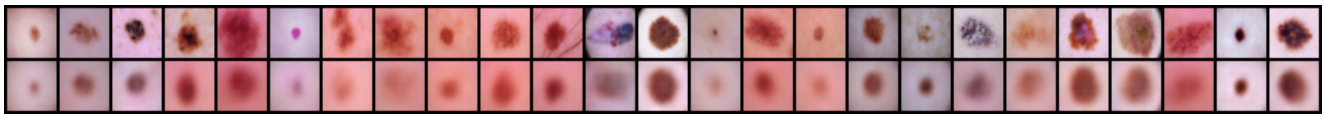

Figure 9: Results for PixelCNN and VQVAE

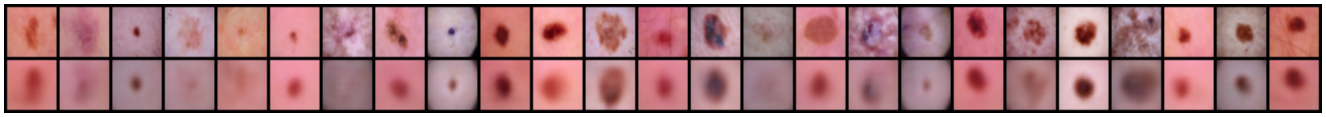

Figure 10: Results for PixelCNN and VQVAE

14 Results on test dataset

We have saved the checkpoits for both the models. Then we run the test datast to get the generated samples.

14.1 Observations

1. From the images we can observe that the images generated on vqvae-2 are really good. Reconstruction is mostly similar to the groundtruth image. This is because quantized representation is trained good so even after quantization it is able to generate similar image as input.

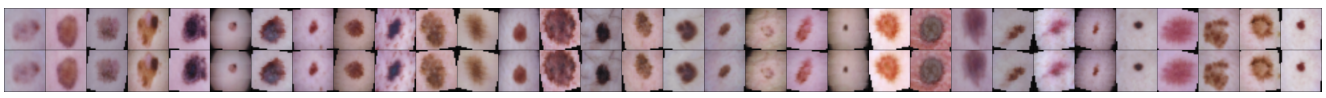

Figure 11: Results for VQVAE

2. We can observe from below results that the images generated on vqvae-2 and pixel cnn combined are decent while training process. Reconstruction is mostly similar to the ground

truth image but at the time of testing the results are not that good. This may be because the prior that is improved may have overfitted to certain kind of disease and now that trained prior by pixelcnn is forcing the model to generate image of single type only. Therefore we can see that all the generated images are similar while testing.

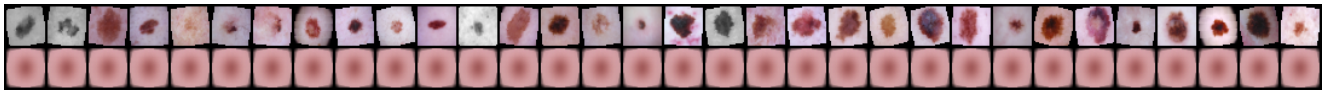

Figure 12: Results for PixelCNN

15 References

- Generating Diverse High-Fidelity Images with VQ-VAE-2
- Dynamic Vector quantization
- VQ-VAE-2 github
- ChatGPT