Abstract:

We explored the generational capabilities of the generative models VAEs and GANs keeping in view their inductive biases. One of the fundamental differences between both of these generative models is that VAEs learn the latent space from the dataset, whereas the GANs try to generate useful output from noise. It is a widely accepted concept that VAEs have a bias towards capturing entire data distribution and GANs are biased towards realism at the expense of some nodes.

Introduction:

In this experiment, we had to train a VAE and a GAN to check their generative images. We also evaluated whether the VAE's learned latent space is continuous via interpolation techniques. In addition, we tested the models with OOD data to see how well they generalized. Finally, by comparing the generative results of each model, we aimed to understand what underlying modular architectures and assumptions led to their outcomes.

Methodology

VAE Training Challenges:

My VAE encountered severe KL divergence collapse, where the KL loss dropped to nearly zero (~0.007) after very first iteration, indicating that the encoder was mapping all inputs to the same latent representation. I attempted several techniques to mitigate this:

Trying different beta values - I experimented with beta values of 0.1, 0.3, 4.0, 20, 100, and 1000 a to give more weightage to KL divergence and prevent the reconstruction loss from dominating the optimization.

Adding skip connections at the decoder - I implemented skip connections to force the decoder to utilize latent information rather than relying solely on learned biases.

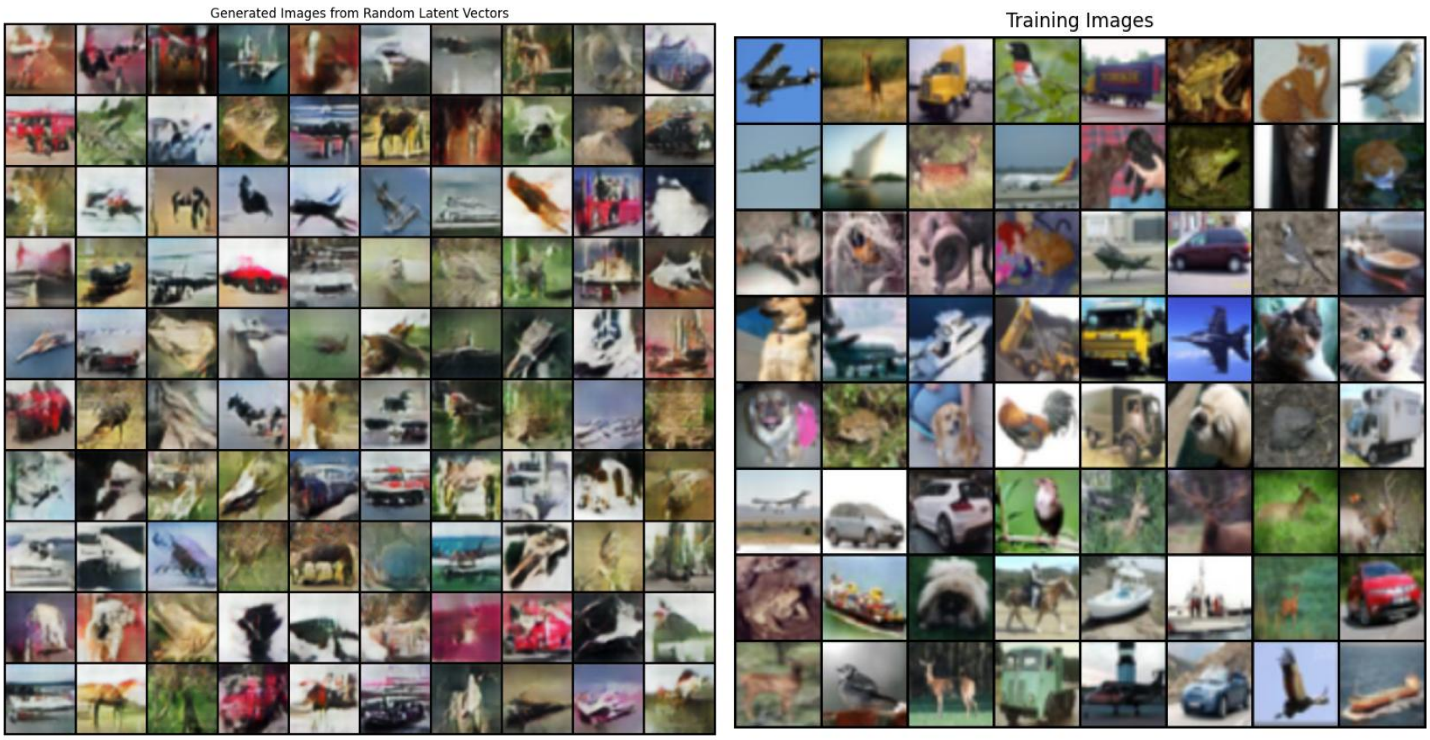
KL annealing - I gradually increased the KL weight during training to allow the model to first learn meaningful representations before enforcing the prior constraint.

GANs

Visual Quality: The generated images are sharp in the sense that the generated classes have clear boundaries, but they lack some features, as can be seen in the image below which was generated from random latent space.

Although the images are blur, that could be because of the poor feature learning during training. Perhaps more epochs with the solution to address mode dropping might help.

Diversity: I have generated 100 images from 70 epoch checkpoint, and they look as shown below. I can see a bunch of distorted classes, i.e., birds, horses with no faces, deer with broken legs, and car-like creatures. These results exhibit a typical example of mode collapse, as all other classes such as trucks, ships, and frogs are nowhere to be seen. Also, I observe that my GAN is generating animal images more often (even though broken ones) than other classes. This is probably because we have 7 classes that correspond to animals and hence more data for animals than other classes. This means that the GAN gave more emphasis to generating animals, which correspond to a larger number of "fools" calls towards the discriminator, and hence rests for easier ways to fool the discriminator.

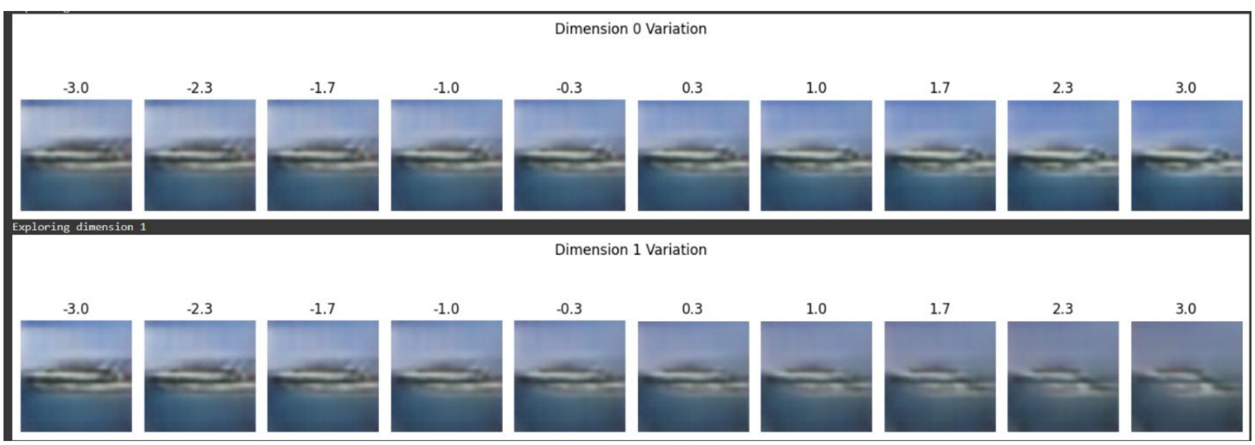


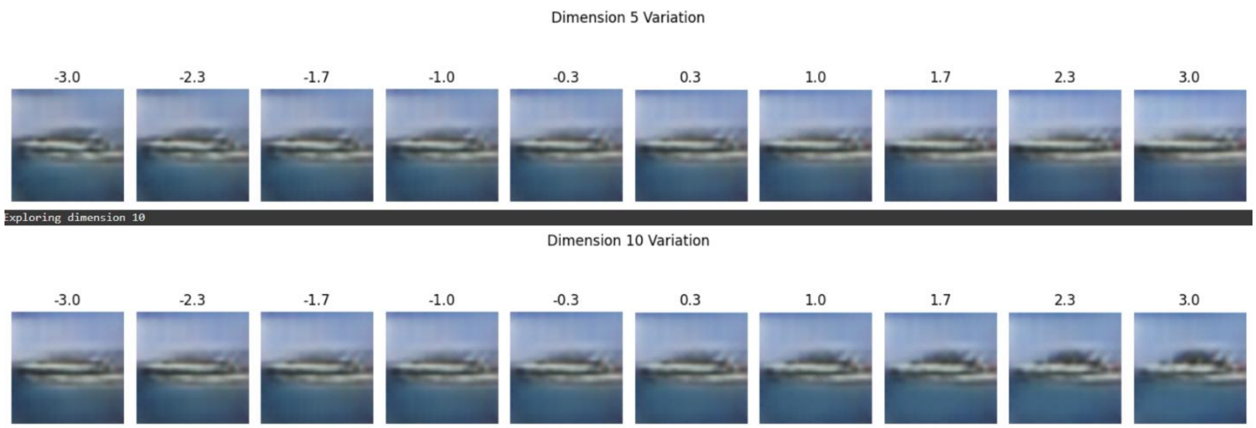
Left side: GAN generated Images, Right side: CIFAR-10 Dataset Images

Interpolation: So I took two images as shown above and interpolated between them. As per the interpolated results, I observed a smooth transition from one image to another show casing the continuity of the learnt latent spaces. Both interpolation endpoints appear to be from very similar classes, which doesn't fully test cross-class interpolation capability. Hence, abrupt change couldn't be observed in this case.

Interpolated Images from GAN

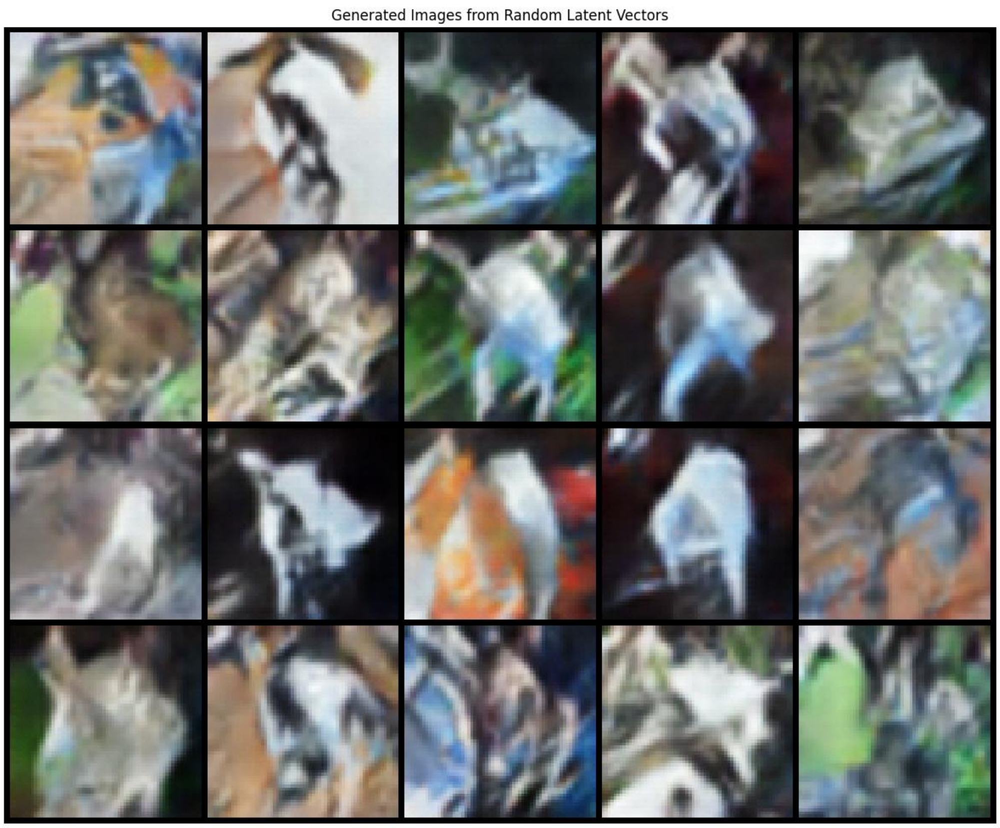
Semantic Meaning: It is observed that varying different dims results in the reduction of features but in most of the cases the distortion has uncertain results. For example, see the variation in dimension 10 results in the disappearance of the boat's base. Whereas dimension 1 affects the sharpness of the image. However, varying dims 50, 70, and 90 all have blurry affect. This is testament to indeterministic nature of the learnt mapping of the latent space in GANs.





GAN OOD: To experiment with the response of a trained GAN on an out-of-distribution dataset, I generated latent spaces with exponential distribution. It only generates positive latent values and has a very different probability density. As we know that GANs typically use normal distribution, which makes it a perfect test case to check GAN robustness with respect to OOD. As can be seen below, the output is meaningless and has no similarity with the CIFAR-10 dataset. But it is also important to note that we do get some output rather than a complete blackout; hence, reflecting on its generative prowess.

When I tested with OOD data, the GAN was still producing some output. The results were not useful, but the model still returned some pixel combinations instead of going completely blank. This shows the GAN's ability to remain somewhat robust in generating fake data, even with OOD input.



OOD Latent Generation using GAN

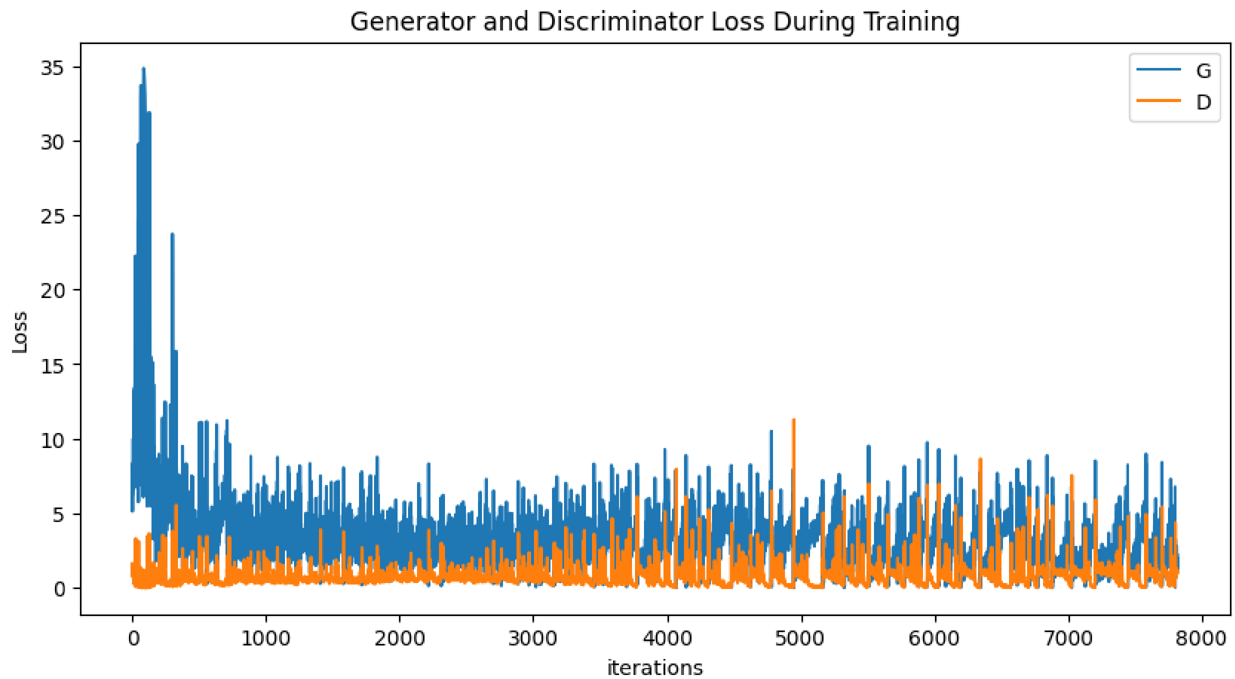
Training Dynamics and Stability: While training the GAN, the model was generating outputs, but mode dropping was visible in the first 100 epochs. After 100 epochs, I saw that the picture quality went down, and by 160 epochs the model fell into mode collapse, where it was only producing one image of a bird. However, at epoch 70, the model was still making images and had some diversity, but most of them were of animals, while others looked random and did not belong to any CIFAR-10 class.

This also shows that continued training degraded the model performance reseulting in mode collapse, generating only one type of image for all latent spaces. It means that model has lost its diversity and learnt to map all latent spaces to the same output.

Model Output at Epoch 170 Onwards

In the interpolation experiment, I saw that as we move from one latent space point to another, the change was smooth. The first image slowly changed shape into the second image.

Moreover, the training loss progression reflects instability inherent to the mixmax game.


Generator and Discriminator Loss During Training
G
D
Loss
iterations
0
5
10
15
20
25
30
35
0
1000
2000
3000
4000
5000
6000
7000
8000