Deep Learning Assignment

1 Architecture and Model Design

1.1 Outline

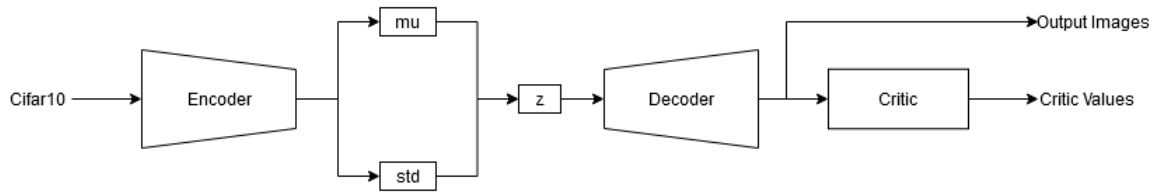
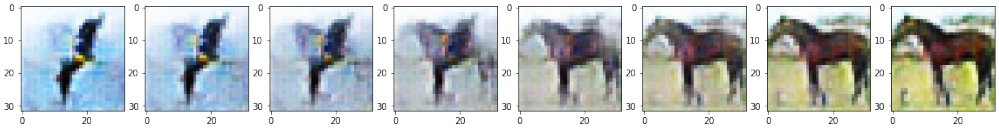


Figure 1: Architecture Pipeline

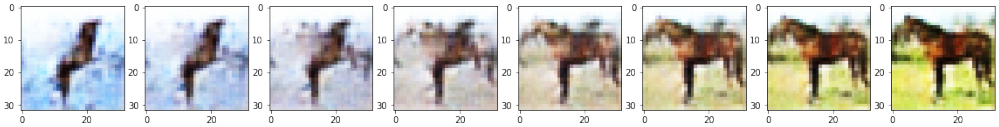
The architecture consists of a Variational Autoencoder (VAE) [3] which trains against a discriminator/critic network. The design takes inspiration from the Adversarial Autoencoders. Using a VAE instead of a traditional autoencoder allows for better-interpolated results from sampling the latent space - the latent space of an autoencoder may not be continuous, as opposed to that of a VAE.
By including a critic network, we hope to improve the generated output images of the VAE, which are typically blurry due to noise, and, as a result, not realistic. In the final implementation, we use a typical Discriminator from a GAN network as our critic, outputting 0 on generated images and 1 on real images, using Binary Cross Entropy (BCE) to calculate the loss. We also tested a critic similar to that in the Adversarial Constrained Autoencoder Interpolation (ACAI) model [2], which predicts a mixing coefficient $/alpha$, using Mean Squared Error (MSE) to calculate the loss. Both produced similar results, but the former trained slightly faster.
Similar prior research has been investigated as well, namely VAE/GAN [1], which suggests to train the VAE on the penultimate layer of the discriminator, and the GAN on real, fake and noise-generated images. In this study, we have compared VAE/GAN with our Adversarial VAE.
The networks are trained on the CIFAR10 dataset. The pipeline is as shown in 1.

1.2 Experiments

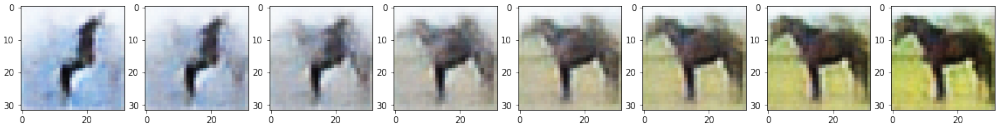
The baseline results are from the Adversarial VAE (described below in 2). All models are trained on solely the horse and bird classes of the training and test dataset in CIFAR10.
Changing Number of Fully Connected Layers seems to change the amount of detail in the animal. Increasing it from one layer to two results in rougher edges.
Changing Number of Convolutional Layers results in slightly blurrier interpolations, but it appears shape information from the original images are reproduced marginally better.
Using spectral normalization increases the amount of blur in the images. Also, the output also contains a checkerboard-like noise pattern.
VAE/GAN seems to result in very high contrast images, and reconstructs the horses well. However, it seems to struggle significantly with recreating the birds. As seen in 2d, the bird image on the left side does not match any of the bird images generated in the other configurations, becoming useless in the interpolation.
ACAI fails to interpolate between images well. Interpolations are similar to interpolating in the image space rather than the latent space, appearing as overlapped images.
Hence, the model we will evaluate is the baseline Adversarial VAE model.



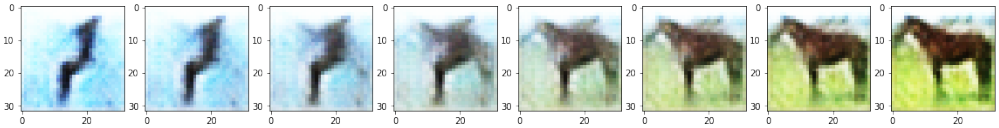
(a) Baseline: 1000 Epochs, 3 Convolutions, 1 Linear Layer, no Spectral Normalization



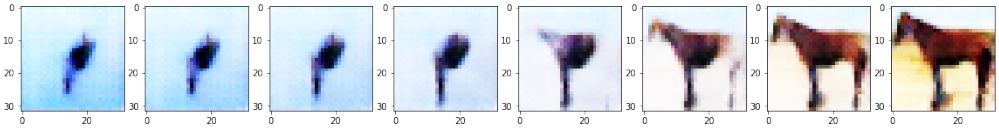
(b) 300 Epochs, 3 Convolutions, 2 Linear Layers, no Spectral Normalization



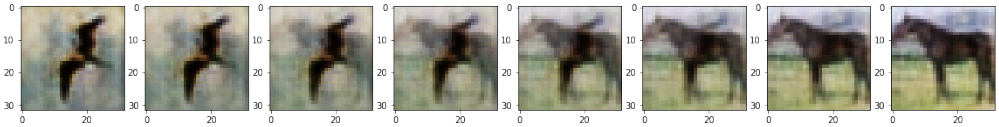
(c) 300 Epochs, 4 Convolutions, 1 Linear Layer, no Spectral Normalization



(d) 300 Epochs, 3 Convolutions, 1 Linear Layer, with Spectral Normalization



(e) VAE/GAN: 1000 Epochs, 3 Convolutions, 1 Linear Layer



(f) ACAI: 1000 Epochs, 4 Convolutions

Figure 2: Comparison of different models and layer counts

1.3 Model Design

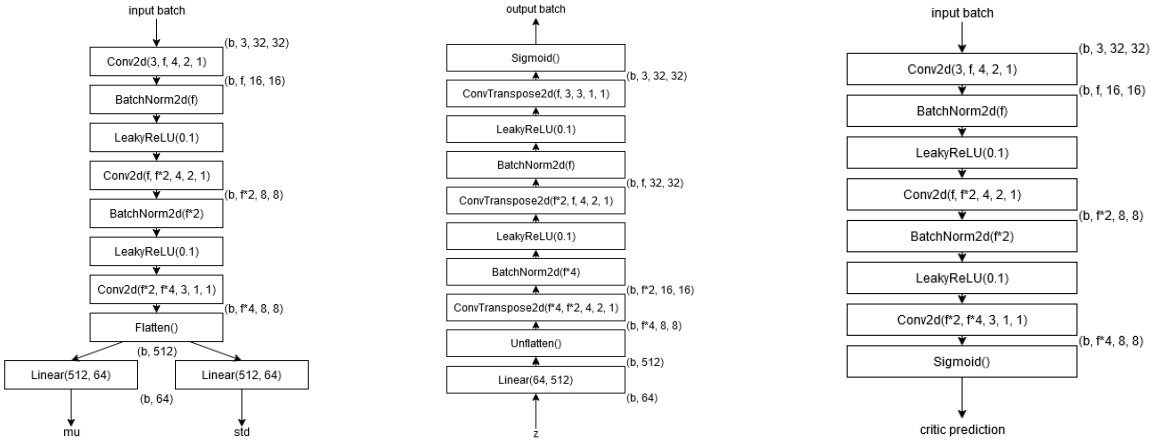
The encoder takes in a batch of images, and outputs 3 values: μ, σ and z. The encoder contains 3 convolutional layers. This is followed by two identical, but separate, fully connected layers, resulting in two outputs from the encoder, μ and σ. The reparameterization trick from [3] is used to sample the latent space, and obtain our single latent code, z.

The decoder/generator is configured in the reverse manner, using convolutional transpose layers instead. It takes a single input, z, and outputs a generated batch of images.

The critic/discriminator is similar to the encoder, but excludes the fully connected layers. Furthermore, the last convolution is separated from the rest of the layers.

All 3 components have batch normalization layers between each convolution, and use the LeakyReLU activation function between convolutions. The decoder and discriminator use Sigmoid as a final activation before output, to constrain outputs to the range [0,1].

Full model details can be seen in 3.



(a) Encoder Model (b) Decoder Model (c) Critic Model

Figure 3: Model Diagrams

1.4 Loss Functions

1.4.1 Adversarial VAE

The VAE is trained on the the KL divergence loss (KLD) as mentioned in [3], the sum binary cross-entropy (BCE) loss between the output batch and input batch, and BCE loss between the predicted critic value of the output batch, and 1:

$$L_{VAE} = KLD + BCE_{sum}(D(E(input)), input) + BCE_{sum}(C(D(E(input))), 1)$$

Where $D(x)$ is the decoder function, $E(x)$ is the encoder function, $C(x)$ is the critic function, and *input* is the real batch of input images.
The Critic is trained on the BCE loss between the predicted critic values of the fake batch and 0, and the predicted critic values of the real batch and 1:

$$L_C = BCE_{sum}(C(D(E(input))), 0) + BCE_{sum}(input, 1)$$

2 Results

To generate an image of a pegasus, the following steps are taken:

1. Encode an image of a horse and an image of a bird
2. Combine the latent codes in latent space through interpolation with some ratio in range [0.4,0.6]
3. Decode the combined latent codes

The results are obtained after 1000 epochs, training on only the bird and horse classes.
The batch of 64 samples is obtained by randomly selecting a horse and a bird image with similar colour values for their center pixel, and doing as above. This is done to encourage better interpolation between the two images, where the animals are of similar colour.

Figure 4: Best Pegasus generated

Figure 5: 64 Generated Images

For more semi-successful generations, see 7.

2.1 Recognisability

In 4, we can clearly see the shape of the horse, as well as a part protruding from the horse's back, resembling a small wing. Ideally, the wing would be bigger. However, the interpolation and smoothing where the wing meets the horse is almost natural. As for in 5, most of the generated samples do not resemble a pegasus - instead appearing to be one of the two classes. Some images contain too much noise for us to discern any animals. Overall, we can conclude that certain combinations of images are more likely to create a pegasus, specifically ones where the bird's wings extend out from where the horse's body would be.

2.2 Realism

The generated images are noisy, especially if the image is generated from the interpolation of two latent codes. However, many remain recognisable and edges are usually retained well. 4 shows this. Colours on the image are vibrant, but not extreme. Overall, the images are reasonably realistic.

2.3 Uniqueness

5 shows much variation - each generated image is different, due to the number of possible unique combinations of bird and horse images. Upon further testing, we have also found that the same pair of horse and bird images could result in slightly different outputs, as shown in 6.

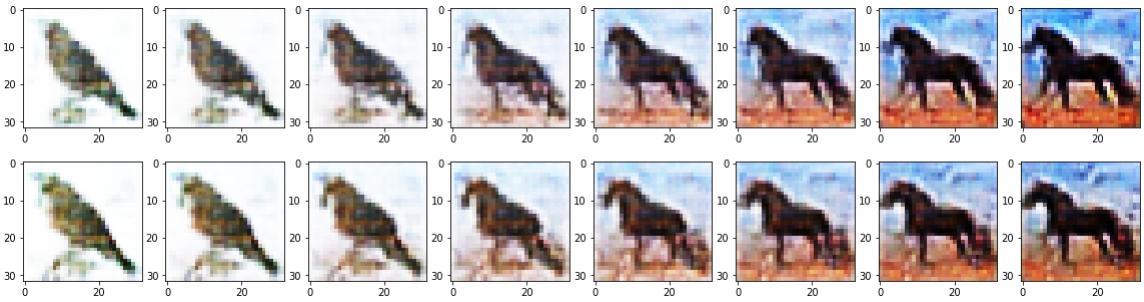


Figure 6: Comparison of interpolation sequences for identical images, across different repetitions.

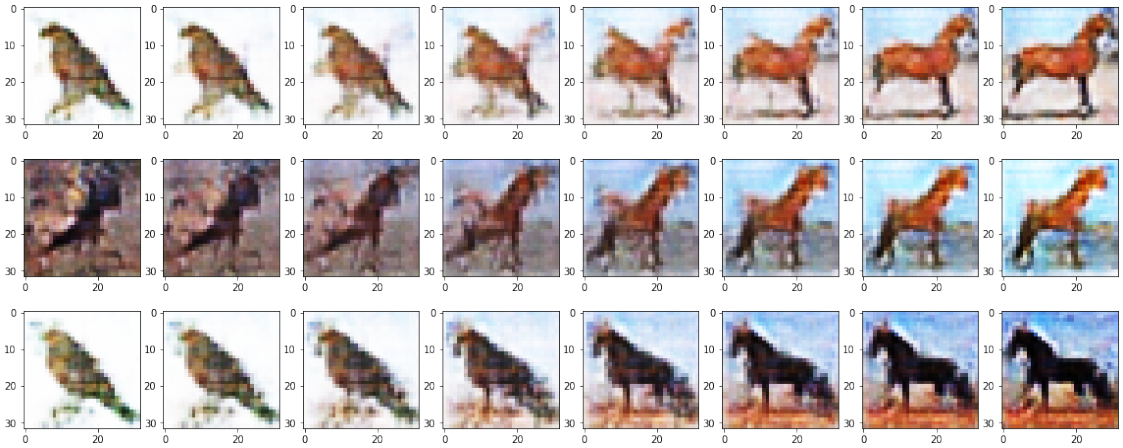


Figure 7: Other Successful Generations

References

[1] A.B.L. Larsen, S.K. Sonderby, H. Larochelle and O. Winther. Autoencoding beyond pixels using a learned similarity metric. arXiv:1512.09300v2. (2016)

[2] D. Berthelot, C. Raffel, A. Roy and I. Goodfellow. Understanding and Improving Interpolation in Autoencoders via an Adversarial Regularizer. arXiv:1807.07543v2. (2018)

[3] D.P. Kingma and M. Welling. Auto-Encoding Variational Bayes. arXiv:1312.6114v10. (2014)