Winged Horses with a Generative Model

Anonymous author

Abstract

This paper proposes using a GLO model to generate images that look like a Pegasus. This model uses a basic generator architecture along with the Laplacian loss function, an informative loss equation which helps at a range of image resolutions. Then, we map each image to a point on the n-dimensional spherical latent space, acting as our probability distribution. Using this spherical distribution, we examine the latent space between horses and birds with the intention of generating a smooth interpolation of the two images.

1 Methodology

1.1 Image Sampling

To simplify generating images of a Pegasus, I decided to reduce the size of my training data-set. I selected all labelled horses and deer, which acted as my generic 4 legged mammal, as well as birds and planes, to learn a latent vector encoding winged features. After reducing the data, I used PCA and K-Means clustering [3] to cluster the images of the horses and birds separately. K-Means clustering minimises the squared euclidean distance between cluster centers and the data-set:

$$\sum_{k=1}^{k} \sum_{i \in C_k} (x_i - m_k)^2 \tag{1}$$

Where $(x_1, x_2, ..., x_i)$ is the image data-set, passed through a PCA dimensionality reduction and m_k is the centroid of cluster C_k. I fitted both PCA models with the gray-scaled STL images, where I selected the clusters with the most white horse like data and flying bird like images.

1.2 Image Reconstruction

The method I propose to reconstruct images is to train a GLO generative model [1]. This model uses the generator from a DCGAN [5], projecting the learned latent value onto a unit l2 sphere and optimises the network by using a Laplacian loss function [2]. The DCGAN generator architecture [Fig 1], increases the size of the third and fourth dimension of the latent space using the transpose of the convolution operation, until $3 \times 64 \times 64$ images is reached. As STL is $3 \times 96 \times 96$, my architecture is similar, however the first layer produces a 2x2 image, with 1536 filters. While the final layer uses a 5×5 kernel and a stride and padding of 3 and 1 respectively to multiply the output dimension by 3, producing a $3 \times 96 \times 96$ image. After the generator produces an image, the Laplacian loss function is performed [2]. This finds the difference between one image and the same image passed through a Gaussian filter [equation 2], which is then repeated by down-sampling the initial image to j levels in the pyramid. Finally, I find the sum of the l1 losses comparing each level in the reconstruction's pyramid and the original's pyramid. More formally, see equation 3 where $L^j(x)$ is the j'th level of x's Laplacian pyramid.

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2}$$

$$\mathcal{L}(x, \hat{x}) = \sum_j 2^{2j} |L^j(x) - L^j(\hat{x})| \tag{3}$$

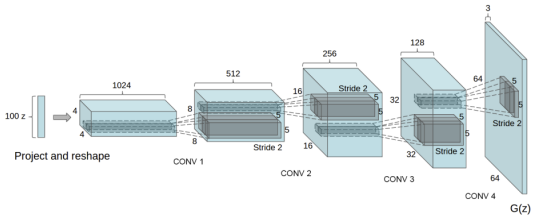


Figure 1: DCGAN Generator [5]

I found better results by not prioritising different levels of the Laplacian pyramid so I have omitted that from my own experiments. It is also clear, that from using this loss function, we can optimise the reproduction of specific features from the original image such as the sharp edges, which are encoded at each level in the Laplacian pyramid. Therefore, improving a fundamental flaw of autoencoders.

After calculating the loss function, I then project the learned latent vector onto a l2 unit sphere, as vectors drawn from a normal distribution are unlikely to land far away from the surface of this manifold [1]. After developing a solution, I experimented here and also tried an embedding network, as seen in step 1 from [4]

1.3 Latent Sampling

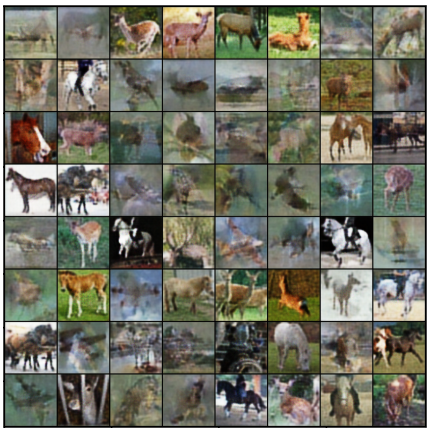
In order to determine the latent vector containing the principal features for this problem, I looked through what each latent vector has learned in order to determine whether the latent space is regularised and semantically meaningful. I experimented with this by iterating through each latent vector and assigning it to the same value a bird may have, as well as linearly interpolating between a horse and bird image.

2 Results

Generally, the reconstructions of the images are of a high quality, especially considering the 96×96 resolution and the non-adversarial method. My best reconstructions of white horses are excellent reconstructions and interpolating with the "right" bird would produce fantastic images of a Pegasus, as shown in 2. However, as seen from the best reconstructed batch in

Figure 2: Best White Horse Reconstructions

figure 3a, it is easy to see that the method heavily results in over-fitting, causing adjacent images in the latent space to "bleed" together. The images selected from this batch should contain only horses, however birds and planes can still be seen. Therefore, demonstrating that the space between images in the latent space is not semantically meaningful, and visually similar images are not close in the latent space. I attempted to interpolate between a selection of "good" birds in the data-set and the collection of horses, this gave me my best batch seen in figure 3b. I selected this as my best batch, as it was able to produce an image of a Pegasus, while also producing diverse and mostly realistic images. With my best Pegasus as seen in figure 4; where you can see a white 4 legged mammal, with it's wing spread out in shadow.



(a) Best Batch Reconstructions

(b) Best Interpolated Batch

Figure 3

Figure 4: Best Pegasus from the batch

3 Limitations

There were 3 main limitations for the results produced above. The first limitation was the time constraints using Google Colab to train the model, alongside other coursework deadlines. The second was due to my methodology. I was unable to produce a semantically meaningful latent space via GLO which made finding a vector encoding winged features or a vector encoding horse colour, nearly impossible. In the future, it would be ideal to train another model which can make sense of the meaningless latent space. For example, I could sample from a regularised distribution such as a multivariate normal distribution, which I pass through a dense linear network, such that it maps to a point in the latent space which I have found above. Let's call this mapping function T, so in equation 4, the sampled noise e, would become a point z on the latent space. I can now write a loss equation to optimise this function. Using equation 5, I can find the l2 norm of the difference between the output of the mapping function T and latent vector z_t, and over time this converges onto a maximum likelihood estimate where the point on the normal distribution maps to the latent space. This can also be seen in figure 5.

$$z = T(e) \tag{4}$$

$$T = argmin \sum_{t} ||z_t - \tilde{T}(e_t)||_2^2 \tag{5}$$

My final limitation was due to the lack of images in my dataset, and therefore I only had a handful of white horses to generate images from. Ideally, in the future, I could improve my PCA sampling method, by using the encoder half of an autoencoder, a convolutional network. With this I could classify images containing bird with wings spread out versus a horse versus neither of those. This would require custom labels, and would require lots of time being spent on labelled images. To speed this up, a transfer learning approach could be used (however this was not allowed for this assignment), by using the pretrained weights of another discriminator network.

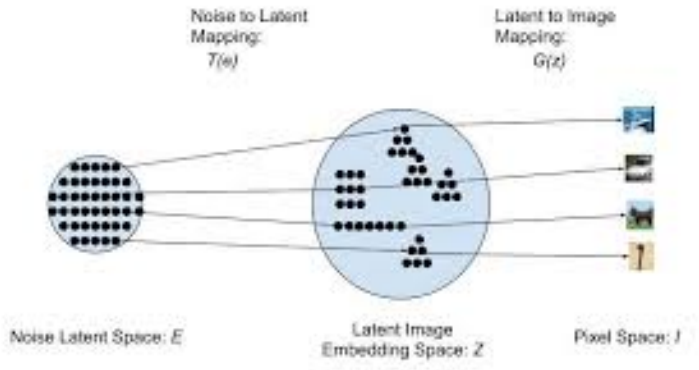


Figure 5: GLANN Architecture [4]

Bonuses

This submission has a total bonus of +3 marks, as it is trained on STL at the full 96x96 resolution. As for the majority white batch bonus points, it is possible to sample a batch of white horses (as shown earlier in figure 2) from my model. I did not include this in my final batch image as showing a different part of the latent space was more informative.

References

[1] Piotr Bojanowski et al. "Optimizing the latent space of generative networks". In: *arXiv preprint arXiv:1707.05776* (2017).

[2] Emily Denton et al. "Deep generative image models using a laplacian pyramid of adversarial networks". In: *arXiv preprint arXiv:1506.05751* (2015).

[3] Chris Ding and Xiaofeng He. "K-means clustering via principal component analysis". In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, p. 29.

[4] Yedid Hoshen, Ke Li, and Jitendra Malik. "Non-adversarial image synthesis with generative latent nearest neighbors". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5811–5819.

[5] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).