Winged Horses with a Deep Convolutional Generative Adversarial Network

Anonymous author

Abstract

This paper proposes using a deep convolutional generative adversarial network (DCGAN) to generate images that look like a Pegasus. This abstract should be short and concise, about 8-10 lines long. This abstract should be short and concise, about 8-10 lines long. This abstract should be short and concise, about 8-10 lines long. This abstract should be short and concise, about 8-10 lines long. This abstract should be short and concise, about 8-10 lines long. This abstract should be short and concise, about 8-10 lines long.

1 Methodology

1.1 Underpinning mathematical theory

The method is to train a deep convolutional generative adversarial network (DCGAN) [4], by having two networks, D and G, play the following two-player minimax game with value function $V(D, G)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}\big[logD(x)\big] + \mathbb{E}_{z \sim p_z(z)}\big[log(1 - D(G(z)))\big]. \tag{1}$$

The discriminator network, D, discriminates between real and fake images. D takes as input an image, x, and outputs the scalar probability, $D(x)$, that x came from training data (real) rather than the generator (fake).

The generator network, G, generates fake images. G takes as input a latent space vector, z, sampled from a standard normal distribution, p_z, and outputs the mapped vector, $D(z)$, in data space (e.g. a space of dimension $3 \times 32 \times 32$, which corresponds to the number of channels, height, and width of an image).

GANs proceed by simultaneously training both G and D. We train D to maximise $logD(x)$, the probability of D assigning the correct label to both real and fake images. At the same time, we train G to minimise $log(1 - D(G(z)))$, the probability of D assigning the correct label to a fake image. Alternatively, we can train G to maximise $logD(G(z))$, the probability of D assigning the incorrect label to a fake image (i.e. making a mistake), which results in the same dynamics of G and D. We decide to train G with the latter objective function, since it is supposed to provides much stronger gradients early in learning [2].

There exists a unique solution (global optimum) to the minimax game. This is achieved when G's estimate of the training data distribution, p_g, matches exactly the true training data distribution, p_{data}. When $p_g = p_{data}$, the fake images G generates (by sampling from p_g) are indistinguishable from the real images (sampled from p_{data}), so $D(x)$ equals $\frac{1}{2}$ for all x, since the discriminator can do no better than guess whether x is real or fake.

1.2 Architecture

An architectural diagram of the method is included in Figure 1. More detailed architectural diagrams of the generator and discriminator networks are also included, in Figure 2 and Figure 3 respectively. Figures 2 and 3 were made using the online tool '*NN-SVG*' [1].

[1] https://github.com/alexlenail/NN-SVG

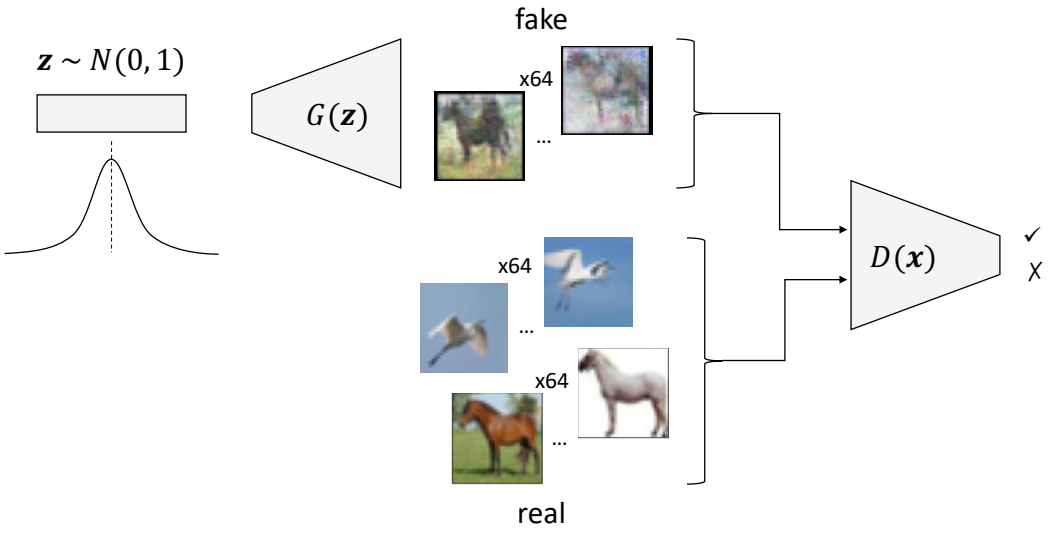

Figure 1: Method architectural diagram. Both generator and discriminator networks process data with a batch size of 64. The image data used to train the discriminator is a manually identified subset of the CIFAR-10 dataset. The motivation and method for doing this is explained in Section 1.3.

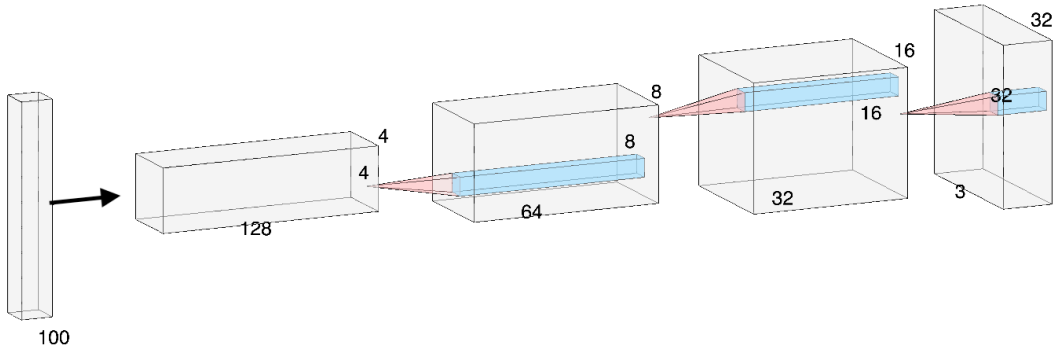

Figure 2: Generator architectural diagram. Transformations happen from left to right.

The generator architecture consists of four layers. The first layer takes as input a $100 \times 1 \times 1$ vector of noise (random numbers) sampled from the standard normal distribution, and consists of a convolutional transpose layer, with a 4×4 kernel, stride 1, and 0 padding, paired with a batch normalisation layer and ReLU activation function. The second and third layers are like the first, except with stride 2 and 1 padding. The fourth and final layer consists of a convolutional transpose layer, with a 4×4 kernel, stride 2, and 1 padding, with *tanh* activation function, and outputs a $3 \times 32 \times 32$ image.

The discriminator architecture also consists of four layers. The first layer takes as input a $3 \times 32 \times 32$ image, and consists of a convolutional transpose layer, with a 4×4 kernel, stride 2, and 1 padding, with Leaky ReLU activation function. The second and third layers are like the first, each paired with an additional batch normalisation layer. The fourth and final layer consists of a convolutional transpose layer, with a 4×4 kernel, stride 1, and 0 padding, with Sigmoid activation function, and outputs a probability.

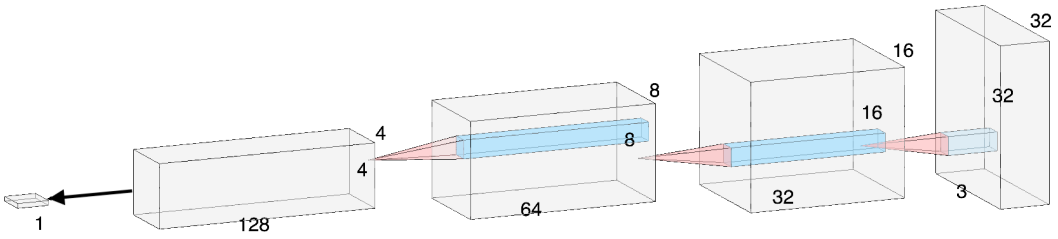

Figure 3: Discriminator architectural diagram. The sequence of transformations flows from right to left.

1.3 Implementation

Our goal was to create unique and diverse images of winged horses that all looked like a Pegasus. We chose to use the CIFAR-10 dataset to train our DCGAN to synthesise a unique batch of 64 images.

In order to create an image that looked like a Pegasus, we had to combine a horse's body with a bird's wings. The first step would be to identify good examples of images that exhibited these features clearly. Therefore, we manually identified birds and horses using the annotated class labels available for the CIFAR-10 images. However, upon inspecting the subset, it was noted that the required features were not often displayed clearly: not all horse images showed the horse's body — instead, showing just the head — and not all bird images showed birds with their wings outstretched — instead, showing their beak or neck.

To get around this issue, we trained a convolutional neural network classifier by Chris Willcocks [2] for 20 epochs, and used it to choose our model training data. By selecting only horse images that the classifier predicted correctly and with over 80% confidence, it was almost guaranteed that the resulting horse images would be of the horse's entire body. Similarly, by selecting only bird images that the classifier misclassified as airplanes with over 80% confidence, the resulting bird images would be of birds in mid-flight with outstretched wings.

From this, we made 10 batches of bird images and 10 batches of horse images, with a batch size of 64. Using a weighted random sampler, we trained the DCGAN on this data, and varied the number of epochs the model trained for, as well as the probability of the discriminator training on horse data versus bird data. Our best results were when the odds of training on a batch of horses to a batch of birds was 1:5.

2 Results

The generated images are low resolution, but they are not blurry since there are clear-defined edges which outline the objects in them. The objects have realistic shapes because one can easily identify the body of a horse and distinguish between its features, including the torso, head, legs, and, in some cases, tail. The objects in them also have some texture, in particular where the shading highlights the mane and musculature. Although there is some noise, the generated images look real enough to classify as horse-like.

The images are not that different to their nearest neighbours in the dataset, indicating that the network has learned well the data distribution of horse images. Although the objects in the generated images all hold the same side-view pose, there is diversity between the samples within the batch of 64 provided, in the orientation of pose (left-facing and right-facing), colour (various shades of brown, black, and white), as well as stance (standing and moving). Unfortunately, few clearly identifiable winged horses were made — the closest were horses with humps instead of wings. There is definitely mode collapse, since we observed

[2] https://colab.research.google.com/gist/cwkx/3a6eba039aa9f68d0b9d37a02216d385/convnet.ipynb

the generator rotated through outputting horse-like objects against a green, grassy backdrop and outputting bird-like objects against a blue, cloudless backdrop (see Appendix).

The best batch of images looks like this:

From this batch, the most Pegasus-like image is:

3 Limitations

It's very difficult to see anything that looks like a Pegasus. This was anticipated since GANs are notoriously difficult to train due to having the following properties: non-convergence, diminishing gradient, difficult to balance, and mode collapse [].

In the future, mode collapse in the GAN could be reduced by lowering the Lipschitz constant for the discriminator function, as was achieve by Wasserstein GANs [1] or by spectral normalisation [3].

Furthermore the diversity and quantity of training data could have been improved by incorporating data from the STL-10 dataset.

although this was not possible due to the time constraints.

Bonuses

This submission has a total bonus of -8 marks (a penalty), as it used an adversarial training method, is trained only on CIFAR-10, and the Pegasus has a dark body colour.

4 Appendix

First, we trained a deep convolutional generative adversarial network (DCGAN) [] on only the images with the 'horse' label in the CIFAR-10 dataset. We did this with the hope of generating images of horses which we can later transform into pegasi. We trained the network for 5, 50, 100, 150, and 200 epochs separately, with a batch size of 64, and generated fake images each time (see Appendix A). By qualitative comparison of the results, it was decided that training for 100 epochs generated the most realistic images (see Section 3 for the limitations of using DCGANs).

Second, we tried fine-tuning the pre-trained network on only the images with the 'bird' label in the CIFAR-10 dataset. We did this thinking that the network might generate images of horses with some bird features, thus resembling a pegasi. We fine-tuned the network for 5, 25, and 50 epochs separately, with a batch size of 64, and generated fake images each time (see Appendix B). By inspection of the results, it was decided that this was not a suitable approach for producing pegasi.

Third, we tried fine-tuning the pre-trained network on a hand-selected image (see Appendix C). We did this hoping that the network would recognise the key feature shared by birds and pegasi — wings — and reflect this in its output. We fine-tuned the network for 5, 25, and 50 epochs separately, with a batch size of 1, and generated fake images each time (see Appendix D). By inspection of the results, and given time constraints, it was decided that this method would have to suffice.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML].

[2] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[3] Takeru Miyato et al. "Spectral Normalization for Generative Adversarial Networks". In: *CoRR* abs/1802.05957 (2018). arXiv: 1802.05957. URL: http://arxiv.org/abs/1802.05957.

[4] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: (Nov. 2015).