Generating Winged Horses with a Modified DCGAN

Anonymous author

Abstract

This paper proposes using a Deep Convolutional Generative Adversarial Network (DCGAN) in order to generate images of Pegasus, the winged white horse depicted in Greek Mythology, using images from the CIFAR-10 dataset. Experiments were then made on modifications of the standard DCGAN in order to improve stability and result quality.

1 Methodology

Firstly, the CIFAR-10 dataset [7] was filtered, using the provided labels, into a set of images of horses. This decision was made based on a personal review of the 'bird' images in CIFAR-10 revealing mostly dormant birds of various colours, making the required white outspread wings of a Pegasus an unlikely feature, in contrast, airplanes are always depicted with wings outstretched however rarely featuring details distinguishable in a Pegasus as many of the images feature planes seen from a distance or silhouetted.

Instead of merging images of various subjects, the approach was thus taken to train a DCGAN on images of just horses, in order to develop strong recognition of the details of a horse, then to add noise to the generator in the form of dropouts, with the goal of blending the white, detailed horse images into having wing-like shapes.

A DCGAN [9, 4] is an adaptation of the standard GAN [2, 12]. In which two generative neural networks: the discriminator and generator, attempt to learn a distribution matching the sample data in order to accurately judge and generate images in-line with the samples respectively. The two neural networks play the minimax game:

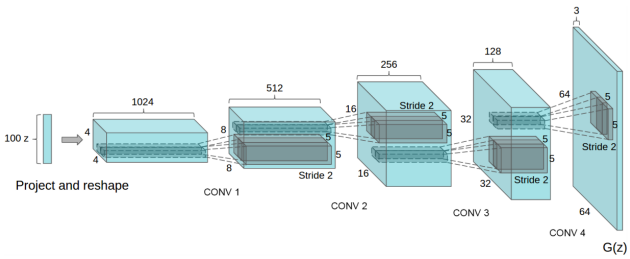
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \tag{1}$$

[2] for which in this paper, $V(D,G)$ is defined as the Binary Cross Entropy (BCE) used to evaluate the loss between the outputs of D and G for a test batch. BCE was shown in [8] to be effective in adversarial image-based generation and in [1] is described as following:

$$\ell(x,y) = L = \{l_1,\dots,l_N\}^\top, \quad l_n = -w_n\left[y_n \cdot \log x_n + (1-y_n)\cdot \log(1-x_n)\right]$$

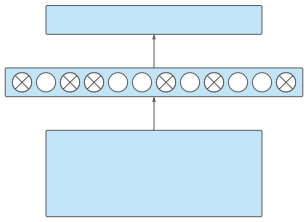
Which simply means that the label for an item y_n is multiplied by the probability of the item being in that category according to the model $logx_n$ resulting in an array of floating point numbers, one for every item in the batch, regarding how accurate the generated probability was for that item.

The DCGAN instance of the GAN involves the use of strided convolutions to transform between images and distributions. This process is depicted in [9] as:



Following this methodology recommended by [9], this paper proposes using a series of four convolutions to transform from a latent vector to an image in the generator and the reverse in the discriminator.

It is within the generator that the key modification is made to the standard DCGAN, following the procedure in [5], dropouts were used to add noise to the generator at three layers of convolution in order to make the generator's task include not just fooling the discriminator but also removing noise from the convolution by tending towards the original images. This aimed to reduce the likelihood of mode collapse and deterministic results. Dropouts were applied with 50% probability to omit values between convolution layers, as depicted below:



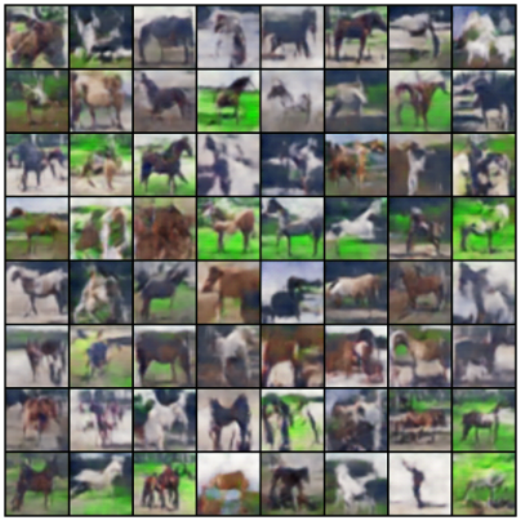
Based on the procedure followed in [9, 2], experiments were also undertaken (see Appendix) using Stochastic Gradient Descent optimisation however in all experiments on the test data, the Adam optimiser [6] provided better-looking results in far fewer training epochs.

Within the training process of iteratively optimising the discriminator and generator, one-sided label smoothing was experimented with as defined in [10] by adding distributed noise to real labels (eg. 1) and none to fake labels (eg. 0). This was attempted due to the findings of [3] in that label smoothing was shown to reduce the vulnerability of neural networks to adversarial examples.

These iterations of optimisation continue until the established number of epochs is reached and the final batch of generated images is output, utilising the methodology of [11].

2 Results

The results of this methodology are varied but with a few recognisable Pegasus in the resulting batch of 64.



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

This is due to the clear horse-like outline, the colour of the horse being white, as well as the white wing shapes extending from its back, covering what could be perceived as a white tail.

3 Limitations

Despite utilising methodologies adapted from various resources in the literature, it is not immediately apparent as to whether these methods are suitable when applied in tandem with each other. Despite some experimentation, it would be beneficial to further study the influence these adaptations have on each other when applied to this particular task so as to find a combination that produces a more appealing visual result. It would be beneficial to establish a key understanding of how these different methodologies might function together without imbalancing one another.

This model is also limited in that it only utilises the CIFAR-10 dataset which is quite limited in scope. Ideally, given greater time and resources, the model could be trained on further examples of horses and birds in order to more concretely learn the features needed to produce a Pegasus image.

One further possible improvement would be to examine the result of training three DCGANs on planes, birds and horses separately then examining the result of integrating their resulting images, this would allow for modification of the strength of each network to appropriately suit the accuracy required from each aspect of the Pegasus.

4 Bonuses

This submission is expecting a deduction of 5 points due to the methodology utilising adversarial training (-4), nearly all winged horses in the batch are white (+1) and the model was trained only on the CIFAR-10 dataset (-2).

5 Appendix

Figure 1: Noise added to labels, trained for 160 epochs

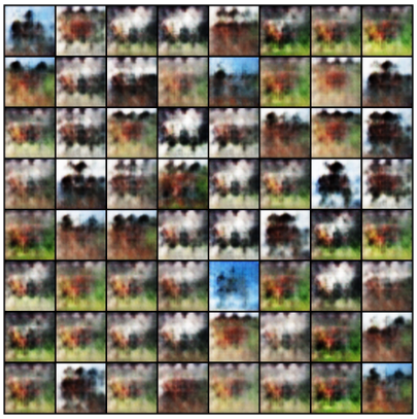


Figure 2: SGD optimiser used on Discriminator, trained for 250 epochs

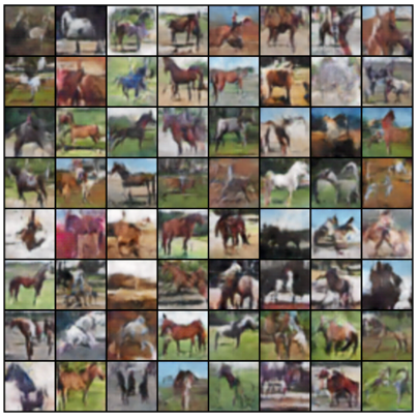


Figure 3: One-sided noise added to labels, dropouts in Discriminator, trained for 160 epochs

References

[1] *bceloss pytorch 1.8.0 documentation*. URL: https://pytorch.org/docs/stable/generated/torch.nn.BCELoss.html.

[2] Ian J Goodfellow et al. "Generative adversarial networks". In: *arXiv preprint arXiv:1406.2661* (2014).

[3] Tamir Hazan, George Papandreou, and Daniel Tarlow. "Adversarial Perturbations of Deep Neural Networks". In: (2017).

[4] Nathan Inkawhich. *DCGAN Tutorial*. URL: https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html.

[5] Phillip Isola et al. "Image-to-image translation with conditional adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134.

[6] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[7] Alex Krizhevsky. URL: https://www.cs.toronto.edu/~kriz/cifar.html.

[8] Junting Pan et al. "Salgan: Visual saliency prediction with generative adversarial networks". In: *arXiv preprint arXiv:1701.01081* (2017).

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

[10] Tim Salimans et al. "Improved techniques for training gans". In: *arXiv preprint arXiv:1606.03498* (2016).

[11] Chris Willcocks. *DL-Assignment.ipynb*. 2020. URL: https://gist.github.com/cwkx/e63ea58890a496d65467761a879c717a.

[12] Chris Willcocks. *simple-gan.ipynb*. 2019. URL: https://gist.github.com/cwkx/74e33bc96f94f381bd15032d57e43786#file-simple-gan-ipynb.