# Progressive Pegasus Production

**Anonymous author**

## Abstract

This paper proposes using a GAN to generate grayscale images that look like a Pegasus and using a second network to add colour to these images. A progressive GAN is used, improving both the speed of training and the quality of the final results.

## 1 Methodology

The method is to progressively train a GAN [1] using images of horses from the CIFAR10 and STL10 datasets. This trains two competing networks, G and D, in a two-player minimax game with value function V(G, D):

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

GANs are notoriously unstable to train. To partially overcome this I have used a progresive training scheme [3] that begins with a 4x4 resolution and scales up during training to produce higher resolution images. This speeds up training by only using a fraction of the two models at any one time. The task of transforming a latent vector into an image, is subdivided into several subtasks of increasing refinement.

To further simplify the GAN training I used only grayscale images of horses. Colour images contain detail that is superfluous to the underlying structure of the horse and attempting to produce RGB images tended to result in mode collapse. I used horses because they are the class most similar to a pegasus in both datasets. When using the full dataset the model tended to prefer to generate simpler classes e.g. trucks.

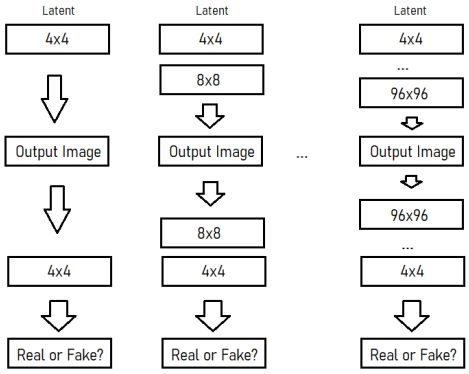
I trained G and D using LSGAN loss [4]. Least Squares loss attempts to minimise the following loss functions:

$$\min_D V_{LSGAN}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}[(D(x) - b)^2] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(z)}[(D(G(z)) - a)^2] \tag{2}$$

$$\min_G V_{LSGAN}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(z)}[(D(G(z)) - c)^2] \tag{3}$$

where a, b and c are hyperparameters. The closest approximation to the normal GAN loss function would involve setting b=c e.g. a binary encoding system where D outputs a value between 0 and 1 to denote fake or real accordingly. In practice, this loss function showed signs of mode collapse almost immediately so I have used the other suggested values from the paper i.e. a=-1, b=1, c=0.

The following diagram shows the increasing architecture of the model throughout training. At each stage the discriminator is asked to distinguish between fake images and real images that have been scaled down to the same resolution. Transistions between resolutions are performed smoothly. Training begins with 30 epochs of producing 4x4 images (each epoch uses 250 batches of data). Then to begin training the next 8x8 layer the 4x4 images are upscaled and a weighted average of the upscaled 4x4 images and the 8x8 images is used. Over the next 30 epochs this weighting shifts linearly to give more weighting to the new resolution. Finally, 30 epochs of training are done using only the 8x8 output, before the process repeats to introduce 16x16, 32x32 and finally 96x96 resolutions. The discriminator follows a similar scheme to introduce each subsequent resolution.




Finally, to produce RGB images I trained an additional colourising network, using the images from STL10. This is a simple network that takes grayscale single-channel images as input and outputs three-channel colour images. The network, C, minimizes a simple reconstruction loss:

$$\mathcal{L}_{\text{RECON}} = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\,\|\boldsymbol{x} - C(\boldsymbol{x})\|^2] \tag{4}$$

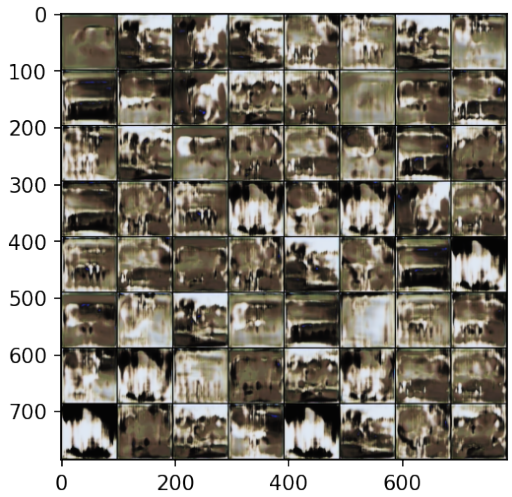
The grayscale GAN has a tendency to produce images that resemble silhouettes more than actual horses. The use of a recolouring network is therefore intended to improve the realism of the final images. Additionally, to help outupt lighter horses I used the negative of the images generated by the GAN, 1-G(z). This allows the bulk of the generated horses to be of a lighter colour. The GAN prefers to output darker horses because the bulk of the horses in the dataset are black or brown.

The recolouring model uses a very simple convolutional design, with each layer taking as input some number of 96x96 channels. The final layer is then passed through a sigmoid function to ensure the model outputs meaningfully coloured pixels.

A number of design choices were made for both the generator and discriminator. Each layer uses convolutions followed by leaky ReLUs and batch normalization [2][6]. The discriminator uses spectral normalization at each downsampling stage [5] and minibatch discrimination in the final layer, to help mitigate the risk of mode collapse [7].

## 2 Results

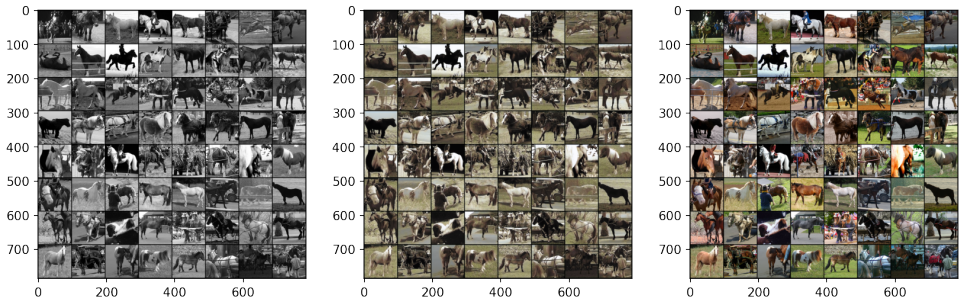
The colouring network has limited capability and produces a vaguely sepia-toned batch:



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

(It has a black head and is sitting on its hing legs, facing left. Its wings occupy most of the upper-right quadrant.)

An additional demonstration of the recolouring network's capabilities:




## 3 Limitations

The final pegasus is clearly an outlier. This can be seen by comparison to the full batch and also by noting that the pegasus has wings. Having been trained on wingless horses the appearance of wings on the final result can only be accidental. Time constraints are likely to have played a significant factor. Training a similar architecture to produce mega-pixel resolution images took 96 hours of training [3]; results shown above are the result of one afternoon's training time.

A more sophisticated colouring network may also have been of benefit. The network was also trained for only several minutes (although it is a relatively simple network).

## Bonuses

This submission has a total bonus of +1 marks, as it is a GAN but trains on both CIFAR10 and STL10 at the full 96x96 resolution.

## References

[1] I Goodfellow et al. "Generative adversarial nets In: Advances in Neural Information Processing Systems (NIPS)". In: (2014).

[2] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[3] Tero Karras et al. "Progressive growing of gans for improved quality, stability, and variation". In: *arXiv preprint arXiv:1710.10196* (2017).

[4] Xudong Mao et al. *Least Squares Generative Adversarial Networks*. 2017. arXiv: 1611.04076 [cs.CV].

[5] Takeru Miyato et al. "Spectral normalization for generative adversarial networks". In: *arXiv preprint arXiv:1802.05957* (2018).

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

[7] Tim Salimans et al. *Improved Techniques for Training GANs*. 2016. arXiv: 1606.03498 [cs.LG].