The Mythical Pegasus: A Mysterious Journey

Max Woolterton
Department of Computer Science
Durham University
`max.woolterton@durham.ac.uk`

Abstract

After successfully implementing and experimenting with two flow-based generative models [1], namely Glow [6] and the multi-scale augmented normalizing flow proposed in [9], and various variational autoencoder (VAE) [5] architectures, the study moved on to Generative Adversarial Networks (GAN) [2], specifically the "lightweight GAN" structure proposed in [8]. All models were implemented with the goal of generating diverse and realistic images of white winged horses (Pegasi) from data sets containing no such images, namely CIFAR-10 and STL-10.

1 Methodology

A GAN consists of a generator G and a discriminator D. G maps samples z from a noise distribution Z, such as a Gaussian, to the image space X. D takes an image x as input and tries to differentiate whether x belongs to the data or $G(z)$, by outputting a scalar $p_x \in [0,1]$ (with $p_x = 1$ corresponding to certainty that x is "real").

Traditionally, G and D are set up to play the following minimax game, where G is trying to fool D and D is trying to maximise its differentiation power between real and generated samples, with both D and G being iteratively trained via backpropagation:

$$\min_G \max_D \mathcal{L}_{GAN}(D,G) \tag{1}$$

where

$$\mathcal{L}_{GAN}(D,G) = \mathbb{E}_{x\sim data}[\log D(x)] + \mathbb{E}_{z\sim Z}[\log(1-D(G(z))] \tag{2}$$

In contrast, the lightweight GAN, proposed in [8] and used throughout this paper, utilises the hinge version of the adversarial loss [7][10] to iteratively train D and G, and D outputs an $n \times n$ feature-map of real/fake logits $\in \mathbb{R}^{n\times n}$ with 1 corresponding to "real" and -1 corresponding to "fake":

$$\mathcal{L}_D = -\mathbb{E}_{x\sim data}[\min(0,-1+D(x))] - \mathbb{E}_{z\sim Z}[\min(0,-1-D(G(z))] + \mathcal{L}_{recons_{\{1,2\}}} \tag{3}$$

$$\mathcal{L}_G = -\mathbb{E}_{z\sim Z}[D(G(z))] \tag{4}$$

where $\mathcal{L}_{recons_{\{1,2\}}}$ is the sum of the reconstruction losses defined in (5) for the two decoders used to regularize D. To provide regularization for D it is treated as an encoder and trained alongside two simple decoders, thereby setting D up as a self-supervised discriminator. Each decoder i is trained only on real samples to minimize a simple reconstruction loss:

$$\mathcal{L}_{recons_{\{i\}}} = \mathbb{E}_{\mathbf{f}_i\sim D_{encode}(x)_i, x\sim data}[||\mathcal{G}_i(\mathbf{f}_i) - \mathcal{T}_i(x)||_2^2] \tag{5}$$

where $\mathbf{f}_i$ are the respective intermediate feature-maps from D, the functions $\mathcal{G}_i$ contain the processing on $\mathbf{f}_i$ along with decoder i, and the functions $\mathcal{T}_i$ represent the respective processing on image x (see Figure 1 for full details). $\mathbf{f}_1$ and $\mathbf{f}_2$ are chosen to be the feature-maps at 16^2 and 8^2 respectively. The decoders comprise three upsampling blocks followed by a final conv-layer to produce images at 64×64, given an 8×8 input feature-map. $\mathcal{G}_1$ selects a random quadrant of $\mathbf{f}_1$ and $\mathcal{T}_1$ follows suit by cropping the corresponding real image(s) to the same 8×8 portion to produce I_{part}. $\mathcal{T}_2$ downsamples the real image to get I, and $\mathcal{G}_2$

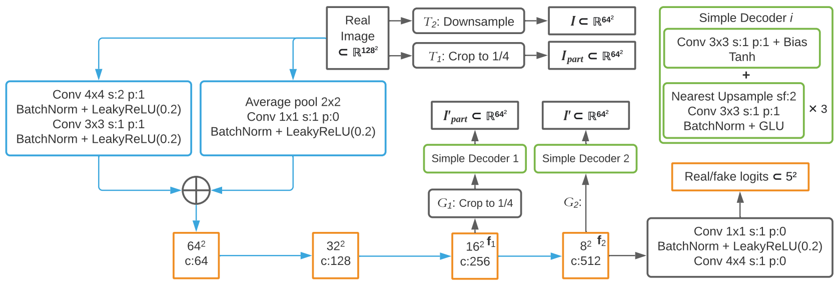

Real Image $\subset \mathbb{R}^{128^2}$
T_2: Downsample
$I \subset \mathbb{R}^{64^2}$
T_1: Crop to 1/4
$I_{part} \subset \mathbb{R}^{64^2}$
Simple Decoder i
Conv 3x3 s:1 p:1 + Bias
Tanh
+
Nearest Upsample sf:2
Conv 3x3 s:1 p:1
BatchNorm + GLU
× 3
Conv 4x4 s:2 p:1
BatchNorm + LeakyReLU(0.2)
Conv 3x3 s:1 p:1
BatchNorm + LeakyReLU(0.2)
Average pool 2x2
Conv 1x1 s:1 p:0
BatchNorm + LeakyReLU(0.2)
$I'_{part} \subset \mathbb{R}^{64^2}$
$I' \subset \mathbb{R}^{64^2}$
Simple Decoder 1
Simple Decoder 2
Real/fake logits $\subset 5^2$
G_1: Crop to 1/4
G_2:
64^2 c:64
32^2 c:128
16^2 $\mathbf{f}_1$ c:256
8^2 $\mathbf{f}_2$ c:512
Conv 1x1 s:1 p:0
BatchNorm + LeakyReLU(0.2)
Conv 4x4 s:1 p:0

Figure 1: The structure and forward flow of the discriminator. Orange boxes represent feature-maps, blue boxes and arrows represent the same residual downsampling structure and green boxes represent the decoder structure shown on the right.

is chosen to be the identity mapping. The decoders then produce I'_{part} from the cropped $\mathbf{f}_1$ and I' from $\mathbf{f}_2$, and are trained to minimise the losses in (5) by computing the squared L2 norm between I'_{part} and I_{part}, and I' and I. This design reduces overfitting from D by ensuring that comprehensive feature-maps are produced up to the final stage, allowing both the global structure (from $\mathbf{f}_2$) and the fine details and texture (from $\mathbf{f}_1$) of the images to be reproduced by simple decoders.

Before being passed to D both the generated and the real images were (differentiably) augmented (as per [4] and [11]) via random translation, square image cut-outs, contrast, brightness and saturation [11] *each* with a 60% chance, along with horizontal mirroring with a 50% chance. This was done to reduce overfitting and increase the quality of generated images [11] due to the very limited number of samples being used to train the GAN (100s rather than 10,000s). Additional augmentations, such as wrap-around translations (offsets) and the addition of noise, were experimented with but proved too detrimental to the quality of images produced due to leaking. This results in the losses in (3) and (4) becoming:

$$\mathcal{L}_D = -\mathbb{E}_{x\sim data}[\min(0, -1 + D(T(x)))] - \mathbb{E}_{z\sim Z}[\min(0, -1 - D(T(G(z)))] + \mathcal{L}_{recons_{\{1,2\}}} \tag{6}$$

$$\mathcal{L}_G = -\mathbb{E}_{z\sim Z}[D(T(G(z)))] \tag{7}$$

where T is the function that composes the differentiable augmentations described above.

To alleviate the longer training times introduced by a deeper G due to increased parameters and weaker gradient flow, Skip-Layer Excitation modules (SLEs) [8] are used (see Figure 2). SLEs provide the same benefits as ResBlock modules [3], with a much lower computation cost and ability to perform skip-connections between different resolutions (allowing for even greater shortcuts in gradient flow), each due to the use of its channel-wise multiplications between activations versus ResBlock's additions. The SLE is defined as:

$$y = \mathcal{F}(\boldsymbol{x}_{low}, \{\boldsymbol{W}_i\}) \cdot \boldsymbol{x}_{high} \tag{8}$$

with $\boldsymbol{x}$ and $\boldsymbol{y}$ representing the input and output feature-maps respectively, $\mathcal{F}$ containing the operations on $\boldsymbol{x}_{low}$ and $\boldsymbol{W}_i$ indicating the module weights to be learned. Figure 2 shows an SLE with input feature-maps of 8×8 and 64×64. Firstly, $\boldsymbol{x}_{low}$ is downsampled into 4×4 and then 1×1 by an adaptive average-pooling layer followed by a conv-layer, with a LeakyReLU then used to model the non-linearity. A further conv-layer is then used to project $\boldsymbol{x}_{low}$ to have the same channel-dimensions as $\boldsymbol{x}_{high}$ so that (after a Sigmoid gating operation) this output from $\mathcal{F}$ can be multiplied with $\boldsymbol{x}_{high}$ along the channel dimension to produce y.

Training was performed solely on STL-10 at 128×128 (upsampled from the original 96×96). Upsampled CIFAR-10 images were originally included with STL-10 when training at 64×64, however, the GAN selectively chose to reproduce only the easier CIFAR-10 images, so they were excluded from that point onwards. All training samples were selected by hand from the two datasets, resulting in a final image count of 564 horse-like images and 143 winged-bird-like images (e.g. birds, airplanes...). The two types of images were randomly sampled

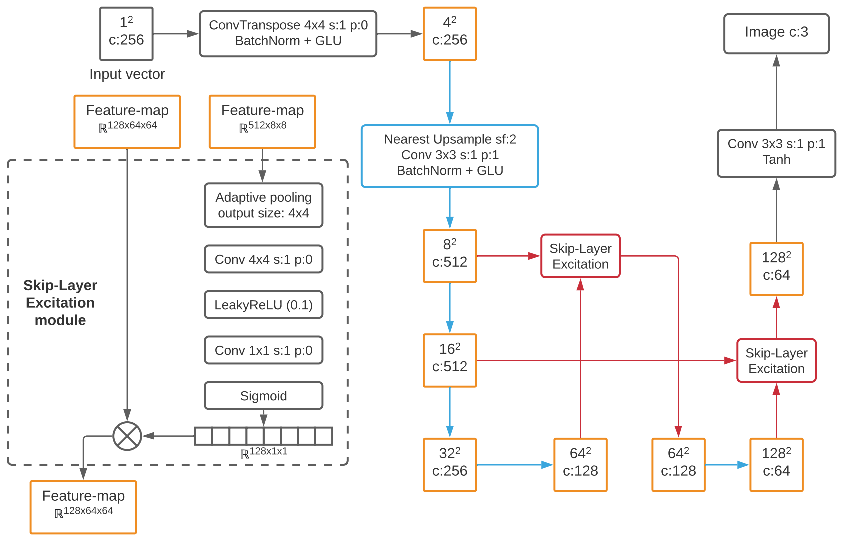
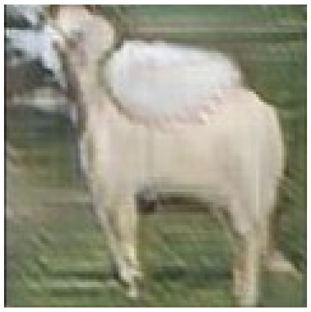


Figure 2: The forward flow of the generator. Orange boxes represent feature-maps, blue boxes and arrows represent the same upsampling structure and red boxes contain the SLE module shown on the left.

with equal importance when training the GAN with a batch size of 8 (4 from each class). Before being passed to D, the bird-like real images (only) underwent an additional stage of augmentation. They were randomly cropped and resized to between 50-100% of the original image with a 30% chance and then rotated 90° clockwise or anti-clockwise with a 20% chance each. This was done to encourage D to recognize wings, in various positions, rather than birds, while also helping to reduce the chances of the GAN opting to overfit and generate only the less diverse and easier to model bird-like images. These augmentations are also applied to the horse-like images but with far lower chances of 5% and 2% respectively.

2 Results

Both of the flow models experimented with suffered from severe capacity issues given the small size and complexity of the datasets and tended to recreate only the grassy or sky backgrounds. The VAE architectures tested struggled to represent the global structure of either class while also producing blurry images. However, the tailored lightweight GAN described above produced some excellent results from the limited datasets, due to its extensive use of Data Augmentation and its lightweight design, as can be seen in Figures 3, 4 & 5.


(a) Best Pegasus?

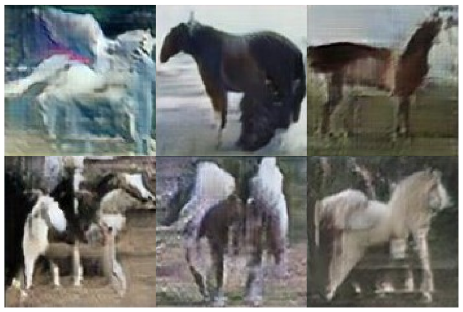

(b) The Other Contestants.

Figure 3: Our Best Pegasi.

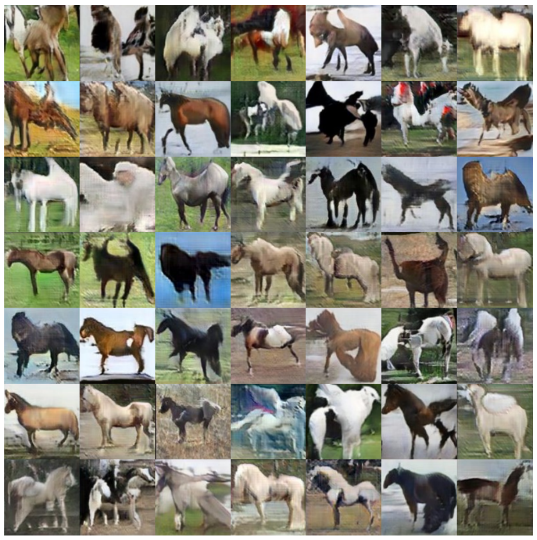


Figure 4: Pegasi picked from training from start (top left) to end (bottom right) going across.

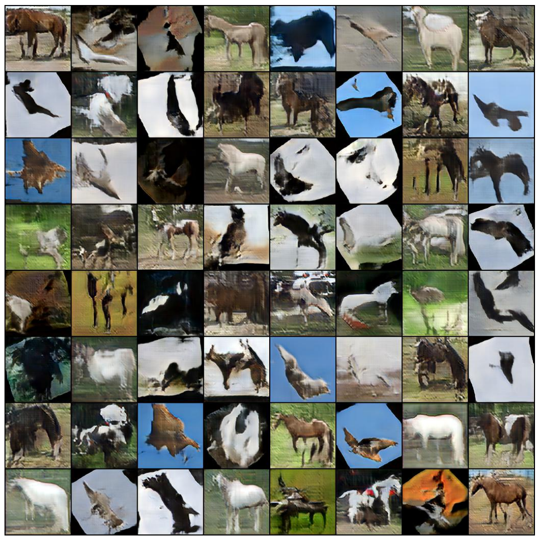


Figure 5: A batch of 64 randomly sampled outputs containing our best Pegasus.

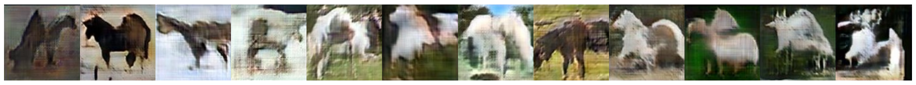


Figure 6: Pegasi picked from our Dual Discriminator training, start to finish.

Extensive experimentation with G setup in an adversarial game against two discriminators simultaneously, one trained 90:10 birds to horses and the other 90:10 horses to birds, was carried out with the results shown in Figure 6. However, although outlines and global structure were promising, the realism and texture of outputs was lacking, leaving our single discriminator setup the preferred option.

3 Conclusions, Limitations and Further Developments

The rotational augmentation starts to leak into some of the birds produced (as can be seen from the corners of the images) as the generator tries to optimize between the 90° rotations applied. However, all Pegasi and horses remain artefact free and are thus able to display various angles of wings. As the Pegasi are the sole aim for this project this rotational augmentation is still considered a success. Overall, considering the limited data used and resolution of images trained upon, the shape, texture and realism of the generated images are impressive, although some of the random outputs are still occasionally unrecognizable in shape. Perhaps this is due to the model not having yet fully converged (evident from the lack of any modal collapse in the random batch, as is customary in a GAN), thus this may improve given more training. Additionally, as is evident from the absence of winged horses in the dataset, the model is able to produce images dissimilar to those it is trained upon, which at the same time remain realistic.

This model could be improved via the incorporation of additional, potentially higher quality/more appropriate (e.g. images of winged horses), training data, the enhancement of the network architecture via the inclusion of any new/novel contributions to the field, or the use of more forms of Differentiable Data Augmentation (although the study in [11] reviews many such combinations and concludes that the current augmentations utilised are hard to beat).

Bonuses

This submission has a total bonus of +2, as it is trained on STL-10 at 96×96 upsampled to 128×128 (+5), most winged horses in the batch are white (+1), and an adversarial training approach is used (-4).

References

[1] Laurent Dinh, David Krueger, and Yoshua Bengio. *NICE: Non-linear Independent Components Estimation*. 2015. arXiv: `1410.8516 [cs.LG]`.

[2] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: `1406.2661 [stat.ML]`.

[3] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: `1512.03385 [cs.CV]`.

[4] Tero Karras et al. *Training Generative Adversarial Networks with Limited Data*. 2020. arXiv: `2006.06676 [cs.CV]`.

[5] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2014. arXiv: `1312.6114 [stat.ML]`.

[6] Diederik P. Kingma and Prafulla Dhariwal. *Glow: Generative Flow with Invertible 1x1 Convolutions*. 2018. arXiv: `1807.03039 [stat.ML]`.

[7] Jae Hyun Lim and Jong Chul Ye. *Geometric GAN*. 2017. arXiv: `1705.02894 [stat.ML]`.

[8] Bingchen Liu et al. "Towards Faster and Stabilized {GAN} Training for High-fidelity Few-shot Image Synthesis". In: *International Conference on Learning Representations*. 2021. URL: `https://openreview.net/forum?id=1Fqg133qRaI`.

[9] Didrik Nielsen et al. *SurVAE Flows: Surjections to Bridge the Gap between VAEs and Flows*. 2020. arXiv: `2007.02731 [cs.LG]`.

[10] Dustin Tran, Rajesh Ranganath, and David M. Blei. *Hierarchical Implicit Models and Likelihood-Free Variational Inference*. 2017. arXiv: `1702.08896 [stat.ML]`.

[11] Shengyu Zhao et al. *Differentiable Augmentation for Data-Efficient GAN Training*. 2020. arXiv: `2006.10738 [cs.CV]`.