Deep Learning Assignment

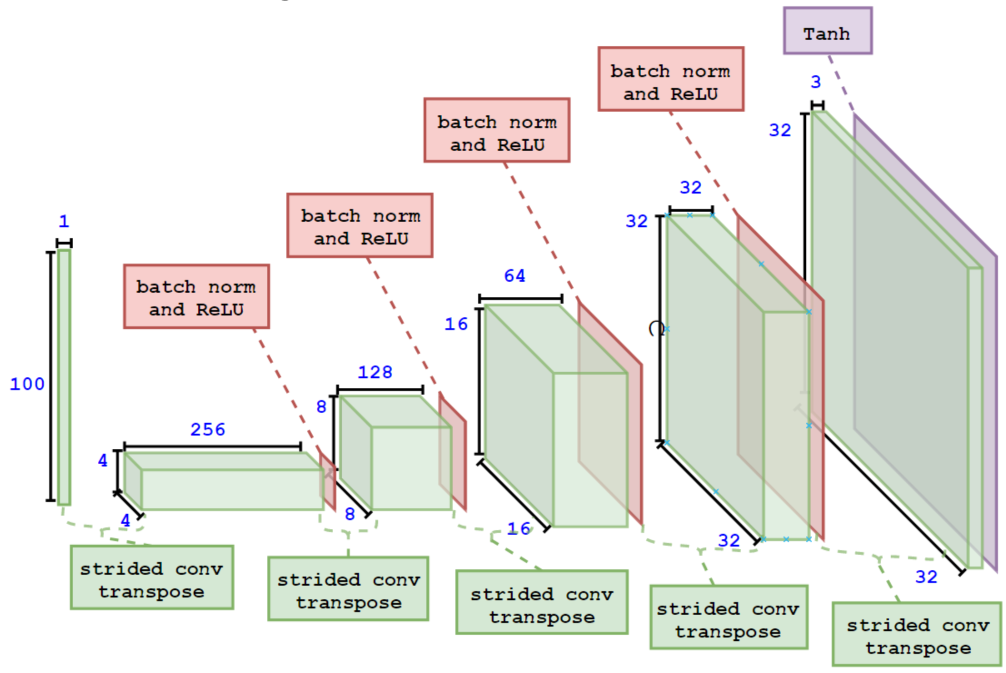
Student: kgxj22
Coursework Deadline: 03/04/2020

1 Introduction

A Generative Adversarial Network (GAN) [1] consists of two interacting models - the generator and the discriminator. The generator produces increasingly realistic fake samples and the discriminator must determine the probability that an input image is an authentic sample from the training set, or a synthetic sample produced by the generator. During training, the discriminator minimises the classification error, while the generator attempts to maximise it, coaxing the discriminator to wrongly classify samples.

We implement a deep convolutional generative adversarial network (DCGAN) [4] based on the PyTorch tutorial [1] by Nathan Inkawhich. In a DCGAN, strided transposed convolutional layers with ReLU activations are added between the input and output layers of the generator (*figure 1a*), and strided convolutional layers with leaky ReLU activations are used in the discriminator (*figure 1b*). Batch normalisation is used in both the generator and discriminator to promote healthy gradient flow. An overview of the model architecture is provided in the figures below, although we further improve upon this architecture through a series of experiments.

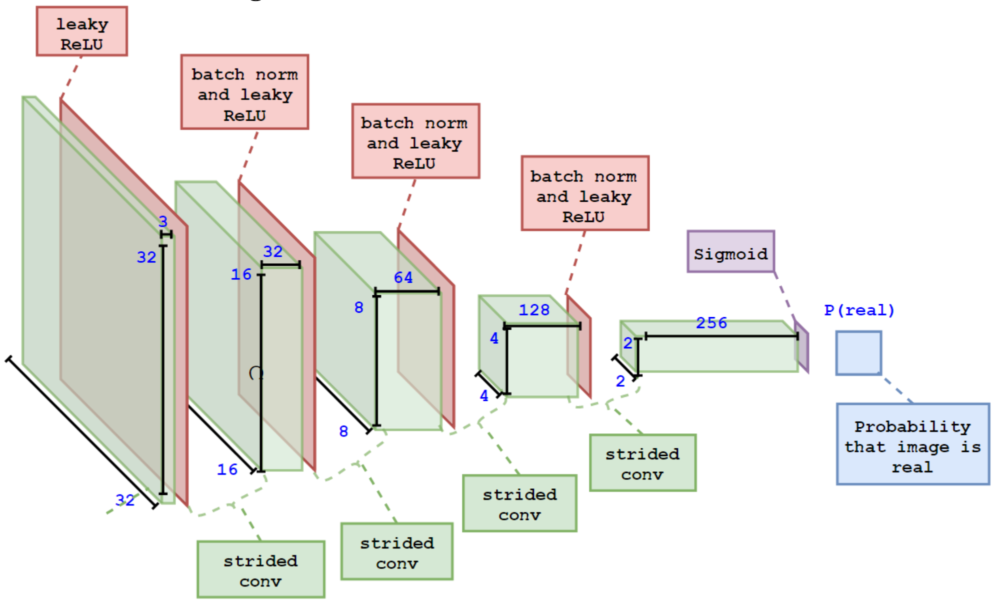
Figure 1a: Model architecture of Generator



The generator maps a $100 \times 1 \times 1$ dimensional latent space vector to a $3 \times 32 \times 32$ dimensional output vector over a series of strided upsampling convolutions. This produces a 3-channel 32×32 image.

[1] https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html

Figure 1b: Model architecture of Discriminator


leaky ReLU
batch norm and leaky ReLU
batch norm and leaky ReLU
batch norm and leaky ReLU
Sigmoid
P(real)
3
32
32
32
16
16
64
8
8
128
4
4
256
2
2
Probability that image is real
strided conv
strided conv
strided conv
strided conv

The discriminator takes a 3-channel 32 × 32 image, and following a series of strided downsampling convolutions, it outputs a continuous value (between 0 and 1) through a Sigmoid activation function - this is the probability that the input image is real.

Experimental Settings – Initially, the whole dataset is shuffled and used without augmentation in batch sizes of 128. We use the Adam [2] optimiser, setting the learning rate to 0.0002 and the beta1 hyperparameter to 0.5 as described in Radford et al. (2015) [4]. Classifying samples as real or fake is a binary classification problem therefore the Binary Cross Entropy loss function is used. Model weights are initialised from a zero-centered normal distribution.

Alternative Architectures – Preliminary experiments on a Spectral Normalisation GAN (SNGAN) [3] were conducted. This implementation was based on the SNGAN model by Chris Willcocks[2]. After training for 200 epochs on horses and birds, the resulting images were pixelated and did not adequately resemble a Pegasus (*figure 2b*). Furthermore, only marginal improvements to image quality occurred after 50 training epochs, as the generator and discriminator losses began to plateau (*figure 2a*).

Figure 2a: Training loss of the discriminator (left) and generator (right)

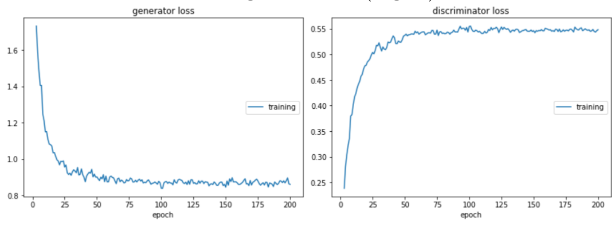
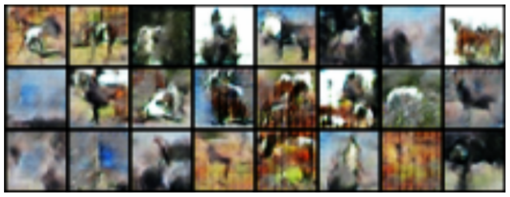


Figure 2b: Images produced by our SNGAN



[2]https://colab.research.google.com/gist/cwkx/f4b49cd3efc0e624bc22c89c90921931/spectral-norm-gan.ipynb

2 Experimentation

Experimental modifications are made to the dataset and the model, and we qualitatively interpret the results using visual information. Where improvements occur, these modifications are permanently included.

2.1 Experimenting with the Dataset

2.1.1 Removing redundant image classes

Removing redundant image classes from the dataset may produce output images that better resemble a pegasus. A subset of the CIFAR-10 dataset is generated, such that only images tagged with specific class names remain in the dataset; these classes are *horse*, *bird* and *plane*. The model is then trained for 80 epochs on two separate datasets.

Examining a batch from the output data shows an improvement in image quality. Horse features are more prevalent in figure 3a, whereas figure 3b shows blurry, less defined output samples.

Fig. 3a: Training on Horses, Birds and Planes

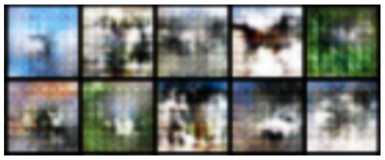
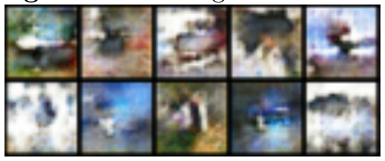


Fig. 3b: Training on CIFAR-10



2.1.2 Enlarging and augmenting the dataset

Horizontally flipping the images with probability 0.5 as they enter the discriminator may increase the variation in the output samples. Additionally, combining the testing and training set may also promote output diversity, as this will provide more data for the model to train on.

This subjectively increases variation, although the training time increases due to the increased amount of training data. This modification is permanently included as the increased training time is a worthwhile tradeoff for increased output diversity.

2.2 Experimenting with the Model

2.2.1 Incorporating spectral normalisation in the discriminator

We hypothesise that by incorporating spectral normalisation on the convolutional layers in the discriminator, as outlined in Miyato et al. (2018) [3], the output samples will be more diverse. Spectral normalisation controls the Lipschitz constant which helps to stabilise training, and prevents mode collapse. This is where the generator model is only capable of producing a limited set of samples.

Output samples were at least as good as previous experiments, and this technique certainly did not hinder the model, although there was no obvious improvement in output diversity. This is believed to be because previous models had not suffered from mode collapse. This modification was permanently included in the model, as this will prevent mode collapse in future training.

2.2.2 Finetuning a trained model

By finetuning the model after training exclusively on horses, the intuition is that this will retain the horse features while incorporating wings. Pegasi are majoritively comprised of horse features, therefore training on horses for a prolonged period before incorporating other image classes may produce better Pegasi. The model is initially trained on 10000 images (from the testing and training set) for 150 epochs and the model weights are stored. We conduct four experiments - finetuning with birds, planes, horses and birds, horses and planes.

When finetuning with only birds and only planes, many of the horse features were lost and the images were less defined. Incorporating horses in the finetuning set led to more horse features in the model output, however these features were warped. This training technique was not used to produce the best outputs.

Figure 4: Results of training the model for 150 epochs on 10000 horse images

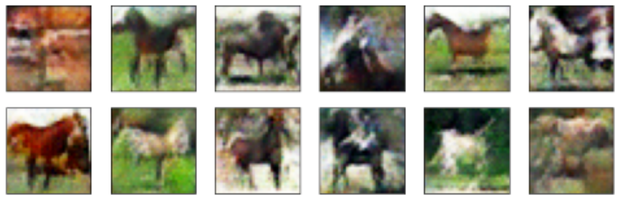


Fig. 4a – Finetuning with birds

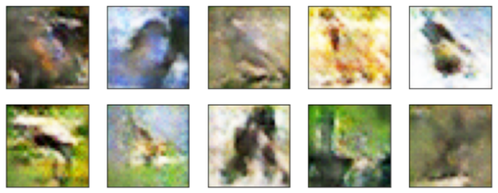


Fig. 4b – Finetuning with planes

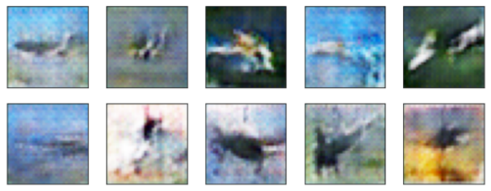


Fig. 4c – Finetuning with horses and birds

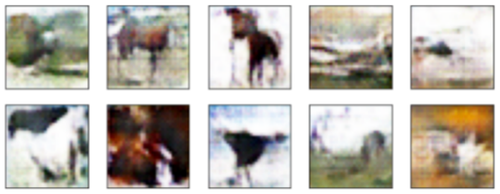
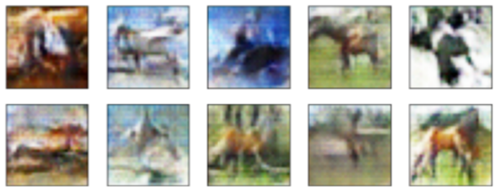


Fig. 4d – Finetuning with horses and planes



3 Results

The best output is shown in *figure 5* and a batch of 64 samples are provided in *figure 6*.

Figure 5: Best Pegasus

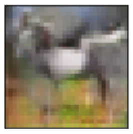
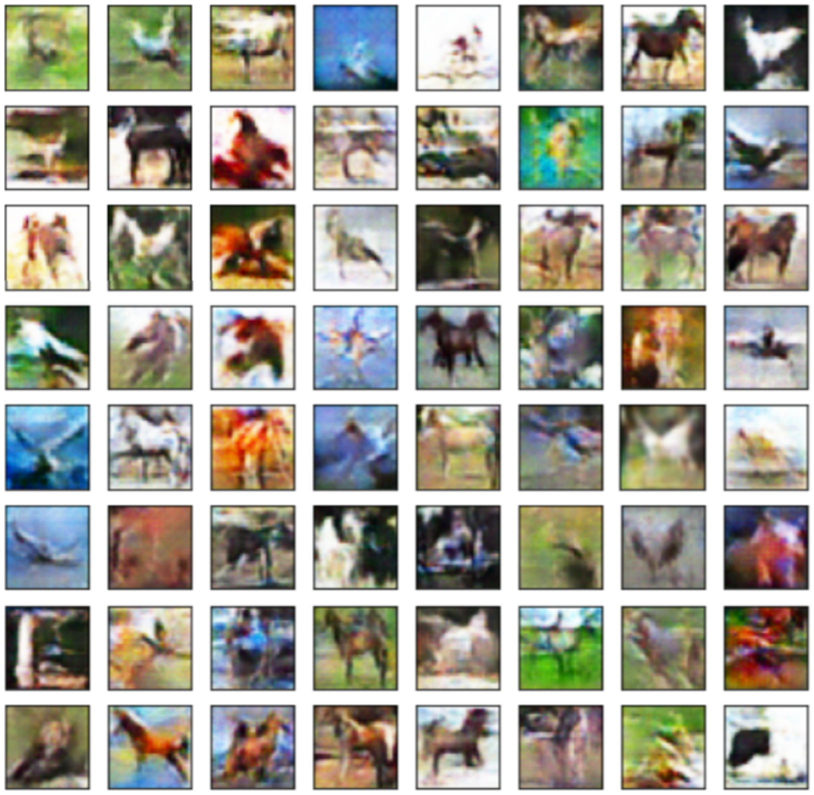


Figure 6: Batch of 64 output samples



References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[3] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

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