Generating Images with Wasserstein GANs

Anonymous author

Abstract

This paper explores the use of a Wasserstein Generative Adversarial Network (hereafter 'WGAN') to generate novel images based on a large dataset. It explores initial attempts with a WGAN and the reasoning that lead to the introduction of a Gradient Penalty along with other improvements, then discusses the potential for future work.

1 Methodology

A natural starting point for this project was to consider a simple GAN, training a Generator and Discriminator against each other in such a way that the difference between discriminator output and the true input value type (genuine or synthesised) is minimised. However, a GAN in itself is a naïve approach, suffering from a high probably of 'mode collapse' (training weights collapsing such that the generator consistently produces a small set of similar samples), among other issues[2].

This was the primary motivation to explore the usage of a Wasserstein GAN, as first proposed by Arjovsky et al [1]. Here, stability is improved and chances of mode collapse are greatly reduced by minimising an approximation of the Wasserstein (or 'Earth-Movers') distance between the probability distributions of x (real data) and $\tilde{x}$ (generated images). It follows that this gives the desirable feature of the generator being 1-Lipschitz continous. This has been shown to be a more preferable loss metric than the Jensen-Shannon divergence used in traditional GANs. This is in part due to the fact that Wasserstein distance is continuous and hence differentiable at almost all points in the space, yielding better results from training.

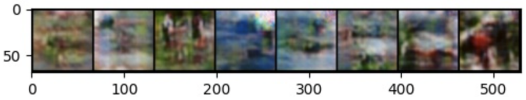


Figure 1: Poor results from a simple WGAN

However, having implemented a WGAN, the results of the model were still relatively poor and stopped converging after a few hundred epochs, leading to incredibly noisey results as shown in Figure 1. This lead to the pursuit the introduction of a gradient penalty term, as discussed by Gulrajani et al [4]. This discard's the standard WGAN technique of 'weight clipping' which has a nature of producing simplistic neural network weights, leading to poorly optimised outputs. The alternative of Gradient Penalties entails using more relaxed constraints, but in such a way which still produces a 1-Lipschitz continous function. It is achieved by uniformly sampling at linear interpolates between the two distributions, and using these samples to calculate a penalty.

This leads to the following loss function for the discriminator, where the first half is a conventional loss and the later the gradient penalty term (with $\lambda = 10$):

$$L = -\mathbb{E}_{x\sim\mathbb{P}_{real}}[D(x)] + \mathbb{E}_{\tilde{x}\sim\mathbb{P}_{generator}}[D(\tilde{x})] + \lambda \cdot \mathbb{E}_{\tilde{x}\sim\mathbb{P}_{\tilde{x}}}[(||\nabla_{\tilde{x}} D(\tilde{x})||_2 - 1)^2]$$

A final improvement to the method implemented was to normalise the images being entered into the model with the distribution $X \sim \mathbb{N}(0.5, 0.5)$ [3]. This lead to much faster convergence of the model, and outputted images were returned to their original value space with the distribution $\tilde{X} \sim \mathbf{N}(-1, 2)$.

Combined, this lead to the creation of the WGAN-GP architecture as shown in Figure 2.

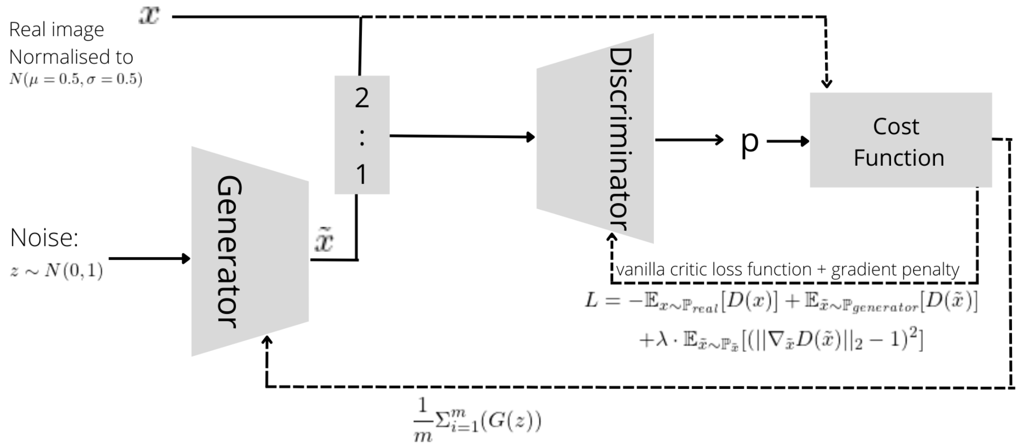
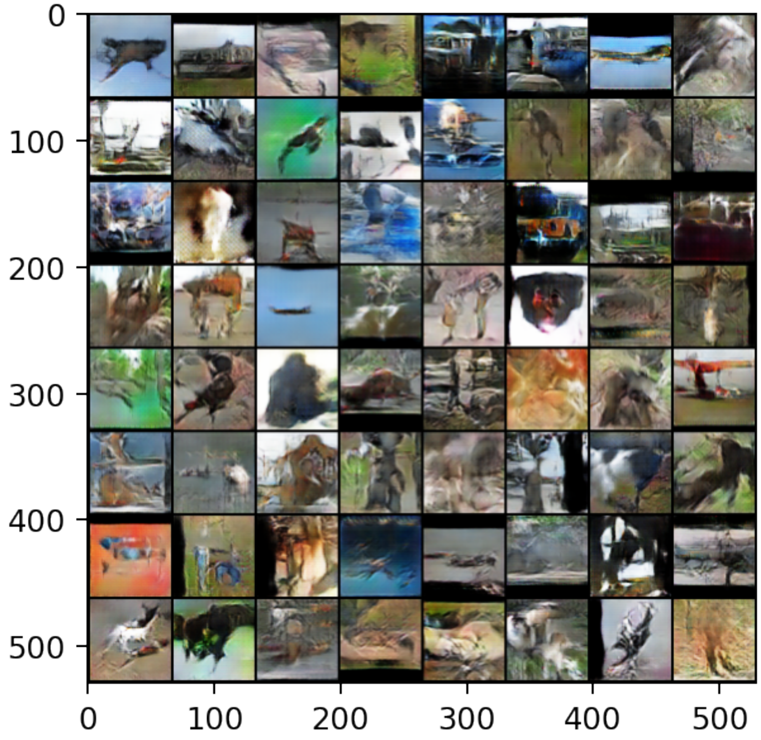

x
Real image
Normalised to
$N(\mu = 0.5, \sigma = 0.5)$
2
:
1
Discriminator
p
Cost
Function
Noise:
$z \sim N(0,1)$
Generator
$\tilde{x}$
vanilla critic loss function + gradient penalty
$L = -\mathbb{E}_{x\sim\mathbb{P}_{real}}[D(x)] + \mathbb{E}_{\tilde{x}\sim\mathbb{P}_{generator}}[D(\tilde{x})]$
$+\lambda \cdot \mathbb{E}_{\tilde{x}\sim\mathbb{P}_{\tilde{x}}}[(||\nabla_{\tilde{x}} D(\tilde{x})||_2 - 1)^2]$
$\frac{1}{m}\Sigma_{i=1}^{m}(G(z))$

Figure 2: The WGAN-GP Architecture used

2 Results

The model was written in PyTorch and (after initial experiments on the MNIST and CIFAR-10 datasets to test for convergence) were executed with the STL-10 dataset at a 96x96 resolution.

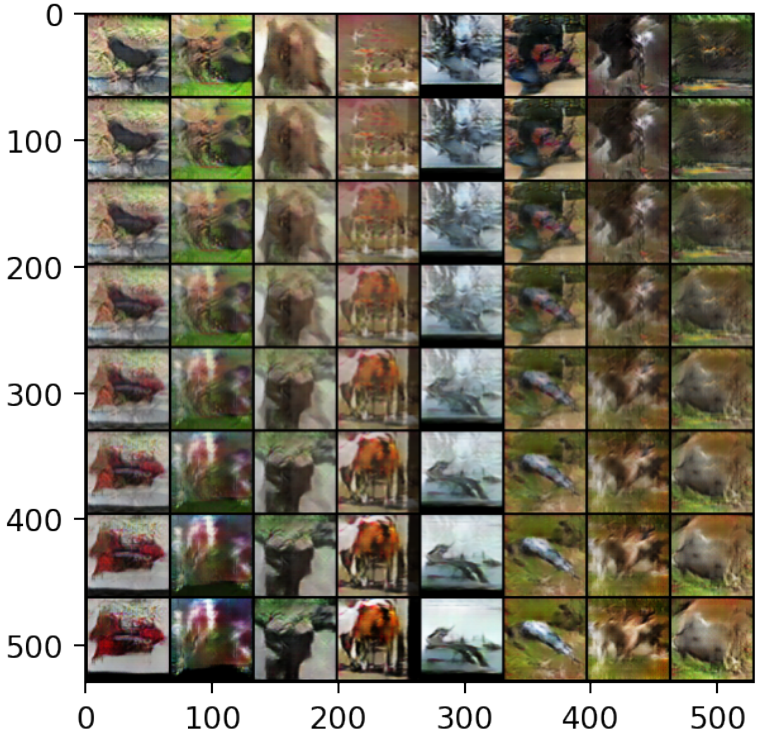
A random sampling of 64 results are shown below. As can be seen, the gradient penalty has led to much better results, with convergence on more well defined objects. However, the realism of the samples could be improved considerably, along with the accuracy of object colour spaces.



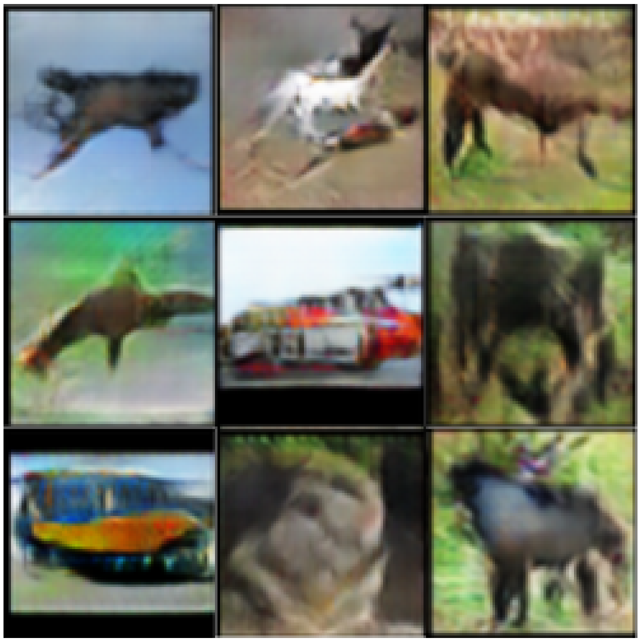
Mode collapse appears to have been successfully avoided here - the samples produced are reasonably diverse and unique from one another, and this is reproduced across repeated sampling.

The model generally seems to be able to generate broad shapes and structures of realistic objects, however is unable to produce fine details with any clarity of consistency. This could be due to the networks of the generator and discriminator being too restrictive - that is, not having a large enough number of neural connections to sufficiently model the complex distributions in the training data.

The next results show images generated by interpolating between points in the latent space. On some samples, midpoints are not entirely realistic with behaviour of alpha blending being exhibited.



And here are some cherry-picked samples that show the best outputs the model has generated:



3 Limitations

While the model does produce distinct objects, to the human eye these are not realistic and it is easy to distinguish them from the training data. One potential factor in this may have been due to a lack of capacity in the neural network - the changes made to the model's layer weights when changing from CIFAR-10 to STL-10 could have been more substantial in adding more layers / increasing the capacity of each layer to cope with the more complex distributions. However, a conscious decision to not do so was made due to time limitations - increasing the model complexity by even 1 layer would have added a substantial increase to training time which could unfortunately not be afforded.

Another limitation is the lack of conditionally - that is, the model is unable to distinguish between classes of objects in it's latent space (e.g. 'bird' vs 'dog' vs 'ship') and generate them on demand, nor to generate the classes proportionally in the sampling. This is caused by the architecture of the model used, and would require significant alterations to be able to respect object classes.

If future work were to be undertaken on these models, I would experiment with how batches are handled (for instance removing or altering the batch normalisation processes) and would run several trials with various layer connectivity setups within the neural network to find the best training-time/result-quality balance.

Bonuses

This submission has a total bonus of -2 marks (a penalty), as it utilises adversarial training (-4) but does train on STL-10 at 96x96 pixels (+2).

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. DOI: `10.48550/ARXIV.1701.07875`. URL: `https://arxiv.org/abs/1701.07875`.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 214–223.

[3] AKSU Gökhan, Cem Oktay Güzeller, and Mehmet Taha Eser. “The effect of the normalization method used in different sample sizes on the success of artificial neural network model”. In: *International Journal of Assessment Tools in Education* 6.2 (2019), pp. 170–192.

[4] Ishaan Gulrajani et al. *Improved Training of Wasserstein GANs*. 2017. DOI: `10.48550/ARXIV.1704.00028`. URL: `https://arxiv.org/abs/1704.00028`.