Winged Horses with a Vanilla Variational Auto encoder

Anonymous author

Abstract

This paper proposes using an Vanilla Variational Auto Encoder to generate images that look like a Pegasus. Variational Auto Encoders are a type of generative model that aims to find the probability density function of the training data that is being used. The images go through both an encoder and decoder while calculating losses and trying to minimise this where possible. This paper presents findings of an implementation of the Variational Auto Encoder (VAE) and whether or not these batch of images meet the requirement of a Pegasus.

1 Methodology

The method is to train a Vanilla Variational Auto Encoder in order to minimise the loss of the training data and produce images of horses with wings, also known as Pegasus'. An illustration of the variational auto encoder:

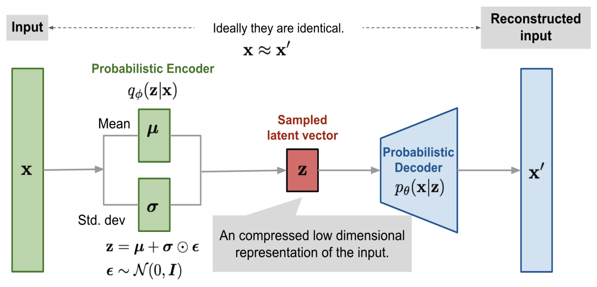

Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
Probabilistic Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
Mean
$\boldsymbol{\mu}$
$\mathbf{x}$
Std. dev
$\boldsymbol{\sigma}$
$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
Sampled latent vector
$\mathbf{z}$
An compressed low dimensional representation of the input.
Probabilistic Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$

Figure 1: VAE model with the multivariate Gaussian assumption [7].

In VAEs, we want to be able to define two processes. One of which is to find at least one latent vector that describes an image and the other process to determine what instructions are needed to reconstruct the image. Below is an equation used to calculate this, where x is the image, the integral tells us to measure over entire latent space, the P is the probability and z is the candidates [8]:

$$\mathbb{P}(x) = \int P(x \mid z) P(z) \, dz \tag{1}$$

$$z \sim p(z|x) = \mathcal{N}(\mu, \sigma^2)$$

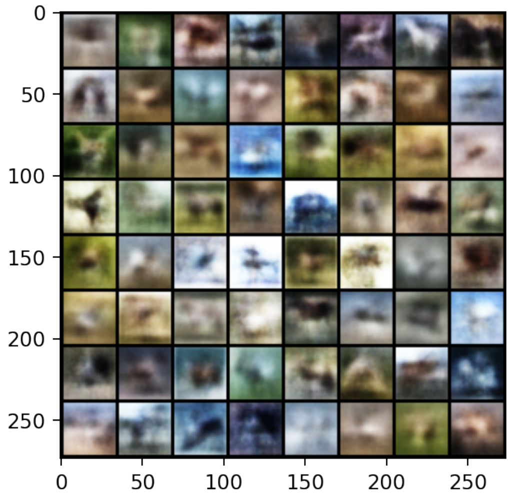
We first let the prior over the latent variables be the centered isotropic multivariate Gaussian. P(z) is then set to be a standard multivariate Gaussian and can be approximated using the Monte Carlo estimate method. This encoder that is used is a 2 layer network that outputs the mean and standard deviation, the latent parameters of distribution [1]. We then take a sample and calculate the losses [3].

The loss term is the summation of two different losses. One of which is the generative loss, this is the mean squared error which is used to determine how accurately the resulting image has been reconstructed. The latent loss which is the KL divergence is the other loss. This loss calculates how closely the latent variables match a unit Gaussian. However, we must optimise KL divergence and this is done by reparameterization to allow vectors of mean and standard deviations to be created [**frans˙2019**].

The decoder takes a sample from the latent dimension and uses that as an input to decode the resulting image [1].

2 Results

The results look very blurry, where the best batch of images looks like this:



From this batch, the most Pegasus-like image (with a bit of stretch of the imagination) is:

From the image, you can make out the image of a dark coloured horse along with what looks like wings that are opened up and lifted, as though the winged horse is ready to fly. From the batch you can see that the neighbours of each image are quite different from each other and not all winged horses look the same.

The graphs below represent the relationship that epoch runs have with both the train and test loss results. The variational auto encoder was run with an epoch number of 300 and the results can be seen below:

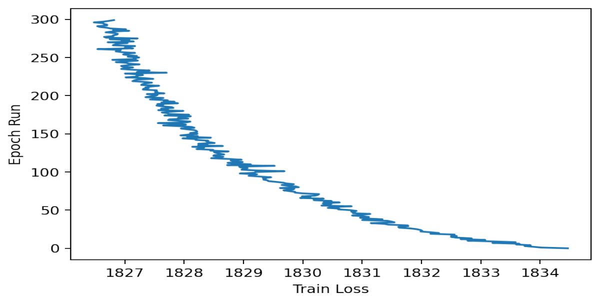


Figure 2: An graph showing epoch runs against the train loss

From (2), we can see that as more epoch runs are made, the train loss becomes smaller and smaller over its course. This is good since that means the error when training the data is decreasing over time resulting in better final images.

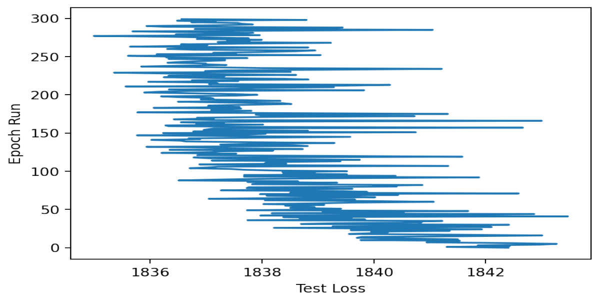


Figure 3: An graph showing epoch runs against the test loss

It is quite difficult the relation between epoch runs and test loss in (3) as the line seems to be erratic no matter the epoch run. At around the 150 epoch run mark, it shows the biggest spike in test loss suggesting that the run might become worse at that point. Overall, the graph suggests that the test loss does very slowly decrease over the epoch runs however the constant spikes in loss makes it hard to distinguish the best number of epoch runs to take. It can be also be predicted that even with more epoch runs the test loss would remain about the same from looking at the graph as the test loss starts to hover around a certain range with more epoch runs.

3 Limitations

From the images that were obtained using the variational autoencoder, it can be seen that is quite difficult to see what is in the images, however you can still make out some horse like objects within the batch. In the future, this could be improved upon by increasing the current 300 epoch run to a larger number, although this was not possible due to time constraints. The blurry, noisy images are a result of using the variational autoencoder since it can be seen as a poor model of data when certain insufficient distributions are used. Basic variational auto encoders (VAEs) ”suffer from multiple deficiencies that stem from the mathematically convenient yet simplistic distributional assumptions” as stated in [4]. This could be improved up by introducing normalising flows using a series of Householder transformations which could lead to an improvement in posterior flexibility [6].

Bonuses

This submission has a total bonus of -4 marks (a penalty), as it is trained only on CIFAR-10, and the Pegasus has a dark body colour.

References

[1] Raviraja G. *Vanilla Variational Autoencoder (VAE) in Pytorch.* Feb. 2019. URL: `https://ravirajag.dev/machine%20learning/data%20science/deep%20learning/generative/neural%20network/encoder/variational%20autoencoder/2019/02/09/vanillavae.html`.

[2] Graviraja. *graviraja/pytorch-sample-codes.* Mar. 2019. URL: `https://github.com/graviraja/pytorch-sample-codes/blob/master/simple_vae.py`.

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

[4] Frantzeska Lavda, Magda Gregorová, and Alexandros Kalousis. “Data-dependent conditional priors for unsupervised learning of multimodal data”. In: *Entropy* 22.8 (2020), p. 888.

[5] Peterhessey. *peterhessey/DeepLearning*. Mar. 2020. URL: `https://github.com/peterhessey/DeepLearning/blob/master/dcgan.py`.

[6] Jakub M. Tomczak and Max Welling. *Improving Variational Auto-Encoders using Householder Flow*. 2017. arXiv: `1611.09630 [cs.LG]`.

[7] Lilian Weng. *From Autoencoder to Beta-VAE*. Aug. 2018. URL: `https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html`.

[8] Yoel Zeldes. *Variational Autoencoders Explained*. Sept. 2018. URL: `https://anotherdatum.com/vae.html`.