A Variational Autoencoder to Generate Winged Horses

Anonymous author

Abstract

This paper proposes using a convolutional variational autoencoder to generate winged horse (Pegasus) images. To achieve this a variational autoencoder is trained on a images of horses and birds to learn a distribution over the data that permits sampling. The experiments performed suggest that this approach is poor at producing these images and produces blurry images that slightly resemble winged horses.

1 Methodology

1.1 Model

The model used is a convolutional variation autoencoder (VAE) [1] this model was chosen as it is a generative model that produces a structured latent space by feeding in input data, allowing for a better understanding of the latent space when compared with a GAN [2]. Variational autoencoders work by learning a distribution over the input data and imposing and additional prior distribution over the latent space $p(\boldsymbol{Z})$. This distribution is usually, and in the case of this paper, the standard normal distribution meaning $\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. This prior is imposed over the learned distribution in an attempt to force a continuous latent space that can be sampled from, as a normal autoencoder could have gaps in the latent space that produce meaningless results when decoded. In order to impose this prior the architecture of a normal autoencoder is modified as described in [1] that is rather than the encoder network encoding to latent space $\boldsymbol{Z}$ it learns a distribution of encodings for each input parameterised by a mean vector $\boldsymbol{\mu}$ and standard deviation vector $\boldsymbol{\sigma}$. On every forward pass through the decoder these are combined in a random manner to generate the encoding of the input thus learning a continuous latent space. The manner in which they are combined according to $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon$ where $\epsilon \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. This is illustrated below in Figure 1:

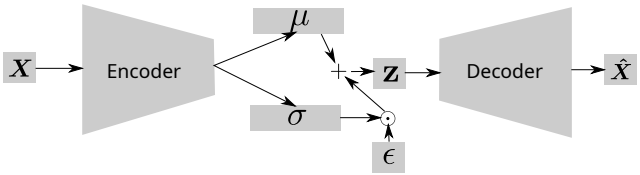

Figure 1: Diagram of VAE arhcitecture

The encoder and decoder are both convolutional neural networks (CNN), with 4 convolutional layers and 4 fully connected layers in the encoder and 4 de-convolutional layers and 4 fully connected layers in the decoder. The large number of fully connected layers were used so that after extracting features from the input image using the convolutional layers the network can better learn how these features represent the input images. Qualitative testing showed that random samples from the model had much more realistic colours when compared with the model without the fully connected layers. Between convolutional layers batch normalisation was used as it has been shown to improve convergence time when training as well as act as a normalisation factor [3]. The networks activation layers were also chosen to be leaky rectified linear activation units (Leaky ReLU) rather than standard linear rectified

activation units (ReLU) as empirical studies have shown that these outperform ReLUs in terms of accuracy [4].

1.2 Loss Function

The loss function being minimised for the VAE is as described in [1]:

$$\mathcal{L}_{VAE} = -\mathbb{E}_{q(\boldsymbol{Z}|\boldsymbol{X})}[\log p(\boldsymbol{X}|\boldsymbol{Z})] + D_{KL}(q(\boldsymbol{Z}|\boldsymbol{X})||p(\boldsymbol{Z})) \tag{1}$$

where D_{KL} is the Kullback-Leibler divergence and $-\mathbb{E}_{q(\boldsymbol{Z}|\boldsymbol{X})}$ is the negated expected log-likelihood for each pixel. The negated log-likelihood is equivalent to cross-entropy therefore in the model cross entropy is used in its place. There also exists a closed form solution to D_{KL} as described in [1] for the Gaussian distributions such as $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ being used in the model. It is:

$$D_{KL} = -\frac{1}{2}\sum_{j=1}^{J}(1 + \log((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2) \tag{2}$$

The cross-entropy portion of the loss function is a measure of how similar the reconstructed image $\hat{\boldsymbol{X}}$ is to the original image $\boldsymbol{X}$ whereas the D_{KL} is used to force the resultant encoding distribution to be as close to $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ as possible.

1.3 Training

In order to generate the winged horses the model was trained on a mix of the STL-10 dataset [5] (resized to 32x32 pixels) and the CIFAR-10 datset [6]. The model was first trained for 100 epochs on the full combined datasets as recommended by [7] this is done to build a useful prior distribution and learn a diverse set of features for image generation. After this all labelled examples if horses and birds were extracted from the datasets and combined into a new dataset of only horses and birds. The model was then trained on this dataset for 300 more epochs, in order to generate a latent space $\boldsymbol{Z}$ that encodes a wide range of features related to horse and bird images, such as wing shape and colour. The optimiser used for the training was the Adam optimiser [8] as is standard in the field.

1.4 Sampling

There are many approaches to sampling from the latent space $\boldsymbol{Z}$, the most naive method would be to randomly sample from the latent space and then decode the samples. If done enough times it is possible that a good image of a winged horse would eventually be produced. However it would be better to take advantage of the structured latent space somehow. Three sampling methods are described in [9], the most desirable of these would be to take advantage of vector mathematics within the latent space to create an "analogy" vector representing wings within the latent space. This wing vector could then be added to any latent space representation of a horse to generate a winged horse. This was not done as the datasets contained very few images of birds that had wings outstretched clearly and none were labeled as such so this would have required manually picking out all images of birds in flight from a dataset of 115000 images which is highly impractical. Instead interpolation was used to generate the images. Specifically spherical interpolations were used rather than linear interpolations for the reasons stated in [9], these are that due to the large number of dimensions in the latent space meaning the mid point between 2 points in the space will be many standard deviations from the mean value. Spherical interpolations address this problem by treating the interpolation as a path on a hypersphere [9]. The equation used to perform these interpolations was:

$$Slerp(q_1, q_2; \boldsymbol{\mu}) = \frac{\sin[(1-\mu)\theta]}{\sin\theta}q_1 + \frac{\sin\mu\theta}{\sin\theta}q_2 \tag{3}$$

where $\theta = \arccos q_1 \cdot q_2$ and q_1, q_2 are latent vectors.

To generate the batch of 64 winged horse images from the samples a set of 8 horse images were manually selected from the dataset and a set of 21 bird images were also selected. For each horse image a bird image was randomly selected from the bird set and then spherical

interpolations were performed with μ being 8 evenly spaced values from the interval $[0.4, 0.7]$, this interval was chosen as qualitatively it produced results that neither looked completely like the bird or horse image bu instead was clearly a merger of the 2.

2 Results

The best batch of winged horses is shown in Figure 2. The results are quite noisy but nowhere near as much as with a normal autoencoder. The diversity of the batch is not great, however that is to be expected due to the sampling method used to generate the batch, as each row is an interpolation between 2 images so all images in said row will be similar. The vast majority of the images in the sample are white, and the images on rows 4 and 5 do look like winged horses with a small stretch of the imagination.

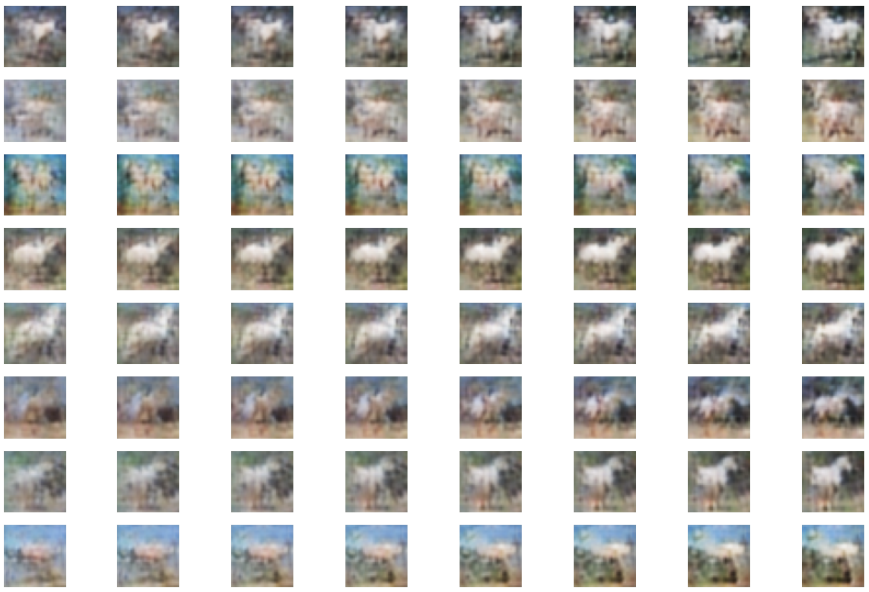

Figure 2: Best batch of 64 winged horses from spherical interpolations

The best winged horse image from Figure 2 is shown in Figure 3 This depicts a white horse

Figure 3: Best Pegasus image from batch

looking at you with its head located in the centre of the image and a clear wing shape in the upper right as well as a less clear wing to the left.

3 Limitations

The obvious limitation of the approach presented in this paper is the apparent "blurriness" of the generated images. This blur is actually noise that is caused by using a VAE and manner in which VAEs approximate maximum likelihood [10].

The model also only generates 32x32 pixel images this is due to it only being trained on CIFAR-10 which is already 32x32 and STL-10 which had been resized to 32x32 as a preprocessing step. This was done as training the model on the full resolution (96x96) images from STL-10 took an extremely long amount of time which was not practical due to time constraints.

The chosen sampling method (spherical interpolations) although good was in itself a limitation as better sampling methods for VAEs exist such as Adversarially Constrained Autoencoder Interpolation [11]. Unfortunately this method of interpolation is an adversarial method thus would incur the -4 point penalty if used, this may have produced more realistic results but there was no way of knowing if they would be better enough to balance the penalty.

Both CIFAR-10 and STL-10 are very poor datasets with regard to bird images. As a large number of such images are of bird heads or birds that are not in flight (wings not outstretched). This made training the model to represent wings in its latent space very difficult as very few images had those desirable features. For better results a dataset with labeled examples of birds in flight would be far more appropriate.

Bonuses

The submission should have a total bonus of +1 as both CIFAR-10 and STL-10 were used albeit at a reduced resolution and nearly all winged horses in the batch (Figure 2) are white.

References

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

[2] Marissa C. Connor, Gregory H. Canal, and Christopher J. Rozell. *Variational Autoencoder with Learned Latent Structure*. 2020. arXiv: 2006.10597 [stat.ML].

[3] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[4] Bing Xu et al. *Empirical Evaluation of Rectified Activations in Convolutional Network*. 2015. arXiv: 1505.00853 [cs.LG].

[5] Adam Coates, Honglak Lee, and Andrew Ng. *An Analysis of Single-Layer Networks in Unsupervised Feature Learning*. Jan. 2011.

[6] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. May 2012.

[7] 2021. URL: http://cs.stanford.edu/~acoates/stl10.

[8] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[9] Tom White. *Sampling Generative Networks*. 2016. arXiv: 1609.04468 [cs.NE].

[10] Shengjia Zhao, Jiaming Song, and Stefano Ermon. *Towards Deeper Understanding of Variational Autoencoding Models*. 2017. arXiv: 1702.08658 [cs.LG].

[11] David Berthelot et al. *Understanding and Improving Interpolation in Autoencoders via an Adversarial Regularizer*. 2018. arXiv: 1807.07543 [cs.LG].