Winged Horses with an Autoencoder

Anonymous author

Abstract

This paper proposes using an autoencoder to generate images that look like a Pegasus. I train an autoencoder, and condition it on birds and horses, identified by the annotated class labels for each image. This gives two latent codes via the encoder network which I linearly interpolate between to give the final outputs via the decoder network.

1 Methodology

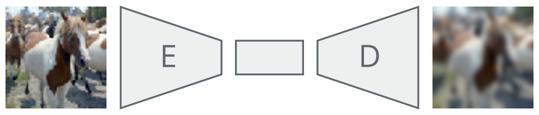
The method is to train an autoencoder [2], by minimising the squared L2 loss:

$$\mathcal{L}_{\text{AE}} = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\,\|\boldsymbol{x} - D(E(\boldsymbol{x}))\|^2] \tag{1}$$

The encoder function $E : \mathbb{R}^n \to \mathbb{R}^m$ compresses the dimensionality of the data $n \ll m$ to a latent encoding $z = E(x)$, which is then recovered by the decoder $D : \mathbb{R}^m \to \mathbb{R}^n$, where $\hat{x} = D(z)$. The autoencoder is conditioned on birds and horses which are identified by the annotated class labels available for the images. This was done by creating a mask from the data source of items which were labelled as either a bird or a horse, and then applying this to the dataset in order to sample only the appropriate images. The latent codes generated by the encoder for the two classes are linearly interpolated between to give the final outputs via the decoder network.

The method was developed on the CIFAR-10 dataset [3] and then tested on the STL-10 dataset [1]. This is because the images in the CIFAR dataset are only 32x32 pixels, so are much faster to train on, meaning that any errors in the method during development would become apparent much more quickly. Adjustments had to be made to the autoencoder to accommodate the change from 32x32 images with CIFAR to 96x96 images with STL, this included an extra convolution block in both the encoding and decoding stages in order to achieve the correct latent space size. Without this extra block, the output images contained unnaturally large blocks of similarly coloured pixels because the latent space was too large.

As visualisation of the autoencoder is as below:

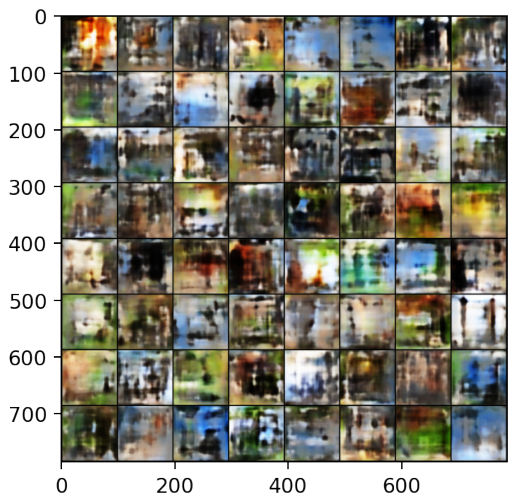


Where 'E' is the encoder, 'D' the decoder, and the block in the middle is the latent space containing the encoded data, which in this case has size 512.

The neural network is trained to predict its inputs, thus learning an identity which it can reproduce as its output.

2 Results

The method ran for 100 epochs on the STL-10 dataset at full resolution, with a learning rate of 0.001. The results look very blurry, where the best batch of images looks like this:



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

This is a white pegasus stood towards the left side of the image and facing to the left (if you use your imagination).

3 Limitations

It's very difficult to see anything that looks like a Pegasus. In the future, this could be improved by training for more than 100 epochs, although this was not possible due to the time constraints. The majority of the winged horses in the batch are not white, this could be improved by manually selecting white birds and white horses from the dataset on which to train, rather than sampling randomly, however this was also not possible due to the time constraints.

Bonuses

This submission has a total bonus of +3 marks, as it is trained with STL-10 at the full 96x96 pixels.

References

[1] Adam Coates, Andrew Ng, and Honglak Lee. "An Analysis of Single-Layer Networks in Unsupervised Feature Learning". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics.* Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. JMLR Workshop and Conference Proceedings, 2011, pp. 215–223. URL: `http://proceedings.mlr.press/v15/coates11a.html`.

[2] Mark A Kramer. "Nonlinear principal component analysis using autoassociative neural networks". In: *AIChE journal* 37.2 (1991), pp. 233–243.

[3] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).