Using Variational Autoencoders as Generative Models

Anonymous author

Abstract

In this paper we explore the use of the established variational autoeconder architecture modified with residual skip connections as a generative model. We will train the autoencoder on the STL-10 dataset resized to 64x64 size images. And we will demonstrate the architecture's generative capabilities by generating a batch of 64 pegasus-like winged horses. Note that no such winged horses exist in the dataset but horses and birds do indeed make up some of the dataset.

1 Methodology

1.1 Autoencoders

Autoencoders (AEs) are feedforward neural networks thar are trained on their inputs. The most basic of AEs are not necessarily generative models; they can be viewed as learned compression functions over some data distribution. AEs consist of two parts, the encoder and the decoder. Typically the encoder takes some high dimesnional input (for example images) and maps it to some lower dimensional latent space which can be thought of as a 1-dimensional vector $\boldsymbol{z} = E(\boldsymbol{x})$, also known the latent representation of $\boldsymbol{x}$. The decoder then reconstructs the image from the latent representation $\boldsymbol{z}$, giving us an image $D(\boldsymbol{z})$ or $E(D(\boldsymbol{x}))$. The whole model is trained by minimising the reconstruction loss between the original image and the output image, given by,

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

For some loss function $\mathcal{L}$

1.2 Variational Autoencoders

Variational Autoencoders (VAEs) build upon this idea but with some additional regularization loss term. VAEs impose a prior distribution on the latent space, typically the unit normal distribution $\boldsymbol{z} \sim \mathcal{N}(0, 1)$, using the Kullback-Liebler (KL) divergence measure.

$$D_{\text{KL}}(p\|q) = \int p(\boldsymbol{x}) \log(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}) d\boldsymbol{x} \tag{2}$$

As we train the model the KL divergence loss makes the latent space become normally distributed, and this is something we can sample from in a meaningful way. Effectively this modification makes our AE a generative model.

1.3 Additional Tricks

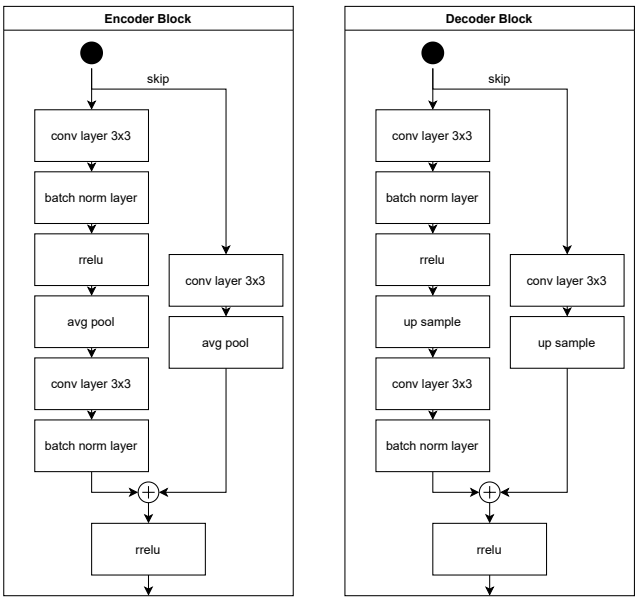
Dropout is an effective strategy for dealing with overfitting [1]. The main idea is to randomly turn off neurons by setting their weights to zero, this helps prevent the neural network from having too much dependence on certain neurons.

In our proposed model we will not use dropout but instead we will use Batch Normalization [2]. Batch normaliztaion can be used as an alternative to dropout, it works by normalizing the mini batches that we train on. Because we now know that the inputs to our model have nice properties we can speed up training by increasing the learning rate.

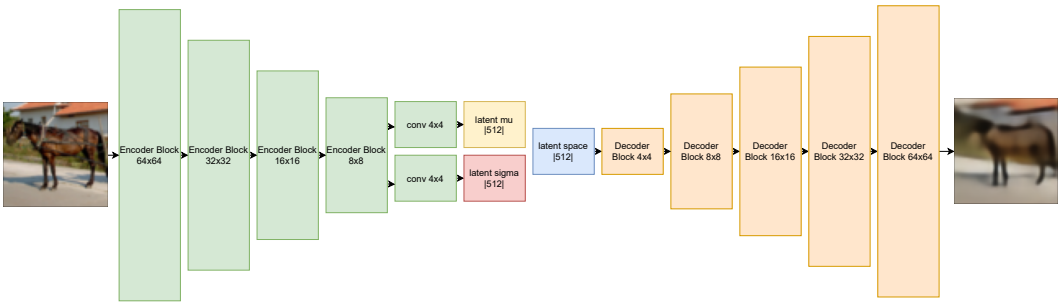
In addition our model will use residual skip connections which have been widely used since their introduction [3]. In our model each residual will effectively skip just one convolutional, two batch norm layers, and one non-linear layer. During training we will also use simple linear annealing of the learning rate.

1.4 Implementation Details

For the implementation of the VAE, we split the encoder and decoder into two seperate python classes. And for the the encoder and decoder we split the layers into encoder blocks and decoder blocks respectively. The architecture for both blocks is given bellow.



Note that we use nearest neighbour for upsampling. The full architecture of our VAE is given below.



1.5 Training

During training we train on the whole STL-10 dataset *"train+unlabelled"*. We train on batches of 64 images for 100 epoches, using a learning rate of 0.0004 that is linearly annealed. For each batch we minimize the reconstruction loss using pytroch's functional *binary_cross_entropy_with_logits*, with the additional KL divergence loss. If we follow this version of the KL divergence formula,

$$D_{\mathrm{KL}}(q\|p) = \mathrm{E}[\log(q) - \log(p)] \tag{3}$$

Then for VAEs using the prior $N(0, 1)$ we can derive the following loss function,

$$\mathcal{L}_{\text{KLD}} = -\frac{1}{2}\Sigma(\exp(\Sigma \boldsymbol{z}) + 1 - \mu^2(\boldsymbol{z}) - \Sigma(\boldsymbol{z})) \tag{4}$$

The whole loss function that we minimize is as follows,

$$\mathcal{L}_{\text{VAE}} = \text{BCE\_with\_logits}_{\text{recon}} + 0.01 * \mathcal{L}_{\text{KLD}} \tag{5}$$

1.6 Sampling Strategy

The pegasus batch generation is quite straightforward but we do some data augmentation first. Firstly, we construct a hand-picked batch of 64 horses from the STL-10 *"test"* set, we choose horses that are predominantly white and clearly visible. This construction was done through human inspection of the dataset. We then construct a hand-picked batch of 64 birds from the STL-10 *"test"* set, again we choose birds that are predominantly white or birds with large outstretched wings.

We pass these two batches seperately to the encoder, giving us two batches of 64 encodings, one for horses and one for birds. Some of the birds in our batch have black or dark wings, to remedy this we multiply the encodings of the birds in our batch that have dark wings by -1. This appears to flip the colours in the reconstructed image to some extent, giving us white wings where there used to be black wings.

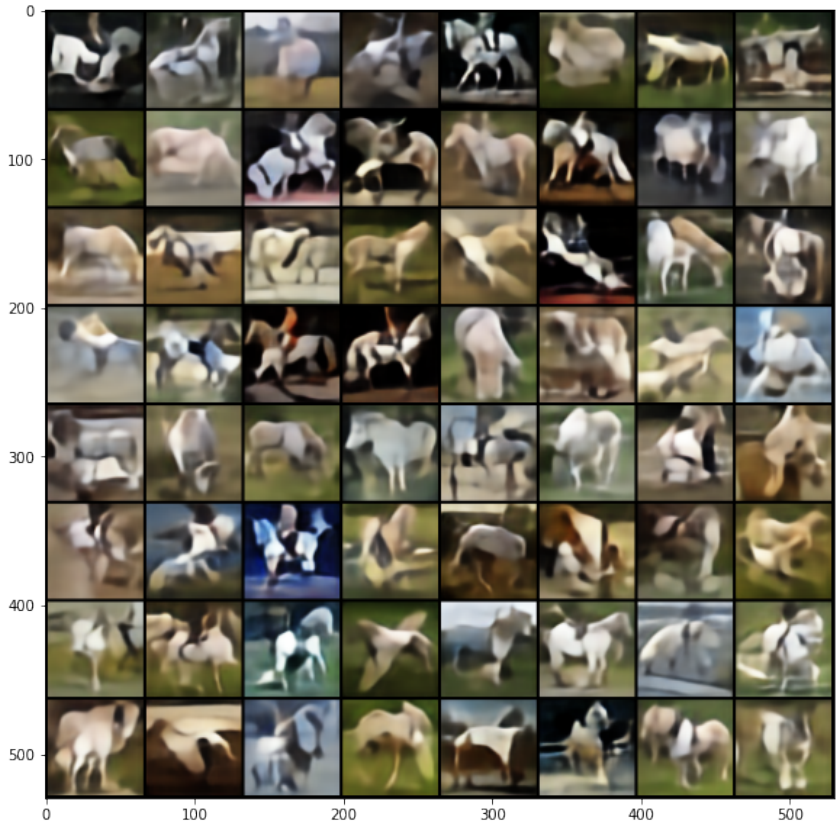
The final step is easy, we just linearly interpolate between the latent encodings of the horses and the modified bird encodings, with weighting $\boldsymbol{w} = 0.6$ in favour of the horse encodings.

2 Results

After 100 epoches of training we acheived the following reconstruction,

The reconstruction is indeed blurry, but the general outline of the horses and some texture is retained. Note that these images are from the STL-10 test set and at no point was the model trainined on these images.

The following batch of pegasus was generated from the trained model,



Clearly most if not all the horses are white, and in every batch there are usually 1 or 2 pegasus-like winged horses. From this batch the most pegasus-like image is the following figure, but if you look closely you may be able to find more in the batch.

3 Limitations

We have shown that VAEs can indeed be used as generative models to some extent. The main limitations of this model are its image reconstruction and dependence on the dataset for generating new images. If perhaps we used a deeper architecture with more residual layers we could perhaps get a better image reconstruction that is less blurry and retains interesting textures.

Understanding the high level meanings of latent space values could help us generate more clear pegasus images, and also allow us to be less dependent on the data set for generating new images. Architectures such as Conditional Subspace VAE [4] aim to provide us with a better understanding of the latent space. Using this understanding we could turn on and off certain high level features such as wings or object colour.

Bonuses

This submission expects a total bonus of +2 marks, the model was trained on the STL-10 dataset (resized to 64 x 64 size images), and nearly all the pegasus in the batch are white.

References

[1] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *The journal of machine learning research* 15.1 (2014): 1929-1958.

[2] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *International conference on machine learning.* PMLR, 2015.

[3] He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016.

[4] Klys, Jack, Jake Snell, and Richard Zemel. "Learning latent subspaces in variational autoencoders." *arXiv preprint arXiv:1812.06190* (2018).