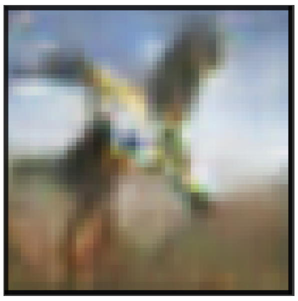
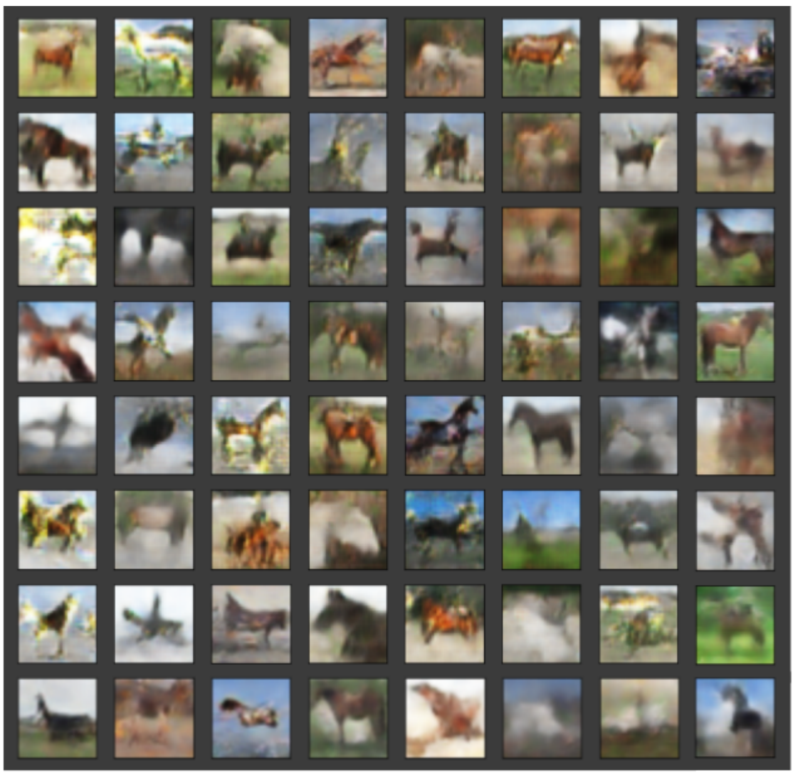
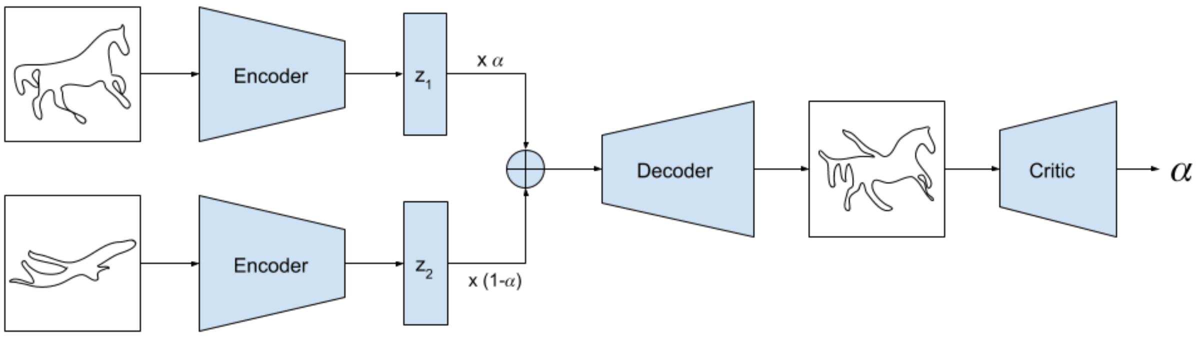
Best output:



Sample batch:



Model diagram:



The architecture of this model is based on the Adversarially Constrained Autoencoder Interpolation (ACAI) model by Bertherlot et al [1]. The aim of ACAI is to produce realistic images from the linear interpolation of the latent codes of two different images. This is achieved through modifying the standard Autoencoder model with the addition of a critic network d_ω. During training, the role of the critic is to try and work out the mixing factor α between the two latent codes, while the goal of the autoencoder is to try and 'fool' the critic into outputting α as zero (in addition to the reconstruction loss). The loss function for the autoencoder is given by

$$\mathcal{L}_{f,g} = \|x - g_\phi(f_\theta(x))\|^2 + \lambda\|d_\omega(\hat{x}_\alpha)\|^2$$

where λ is a hyperparameter. As both models are trained, if the autoencoder eventually 'wins', the critic will output zero for images produced from interpolated latent codes as well as non-interpolated ones (single images). If the critic is sufficiently good, this means that the interpolated images will look just as realistic as the reconstructions of single images.

In my architecture, the encoder consists of three convolutional layers, going from 3 to 32, 64, and 128 channels respectively, followed by a fully connected layer. The size of the latent code is 32. The decoder mirrors the encoder and consists of a fully connected layer followed by three transpose convolutional layers. The critic architecture is identical to the encoder except the output is the mean of the activations of the final layer so that a single number (for α) is produced.

I chose to train my model on only the images of horses and planes from the CIFAR-10 dataset. The reason for this is because the proportion of bird images with visible wings was very low, making them not very useful for generating a pegasus, while almost all of the plane images had long, visible wings that could be used. The data was augmented with a random horizontal flip to increase training data volume.

To get good results from my model, I had to find the right weighting λ between the reconstruction and interpolation loss functions. To do this, I started with a value of zero, so that the reconstructions of single images were good quality but interpolations were nonsense, which I kept track of by displaying two reconstructions of training images and 9 interpolations between them after each training phase. I gradually increased λ until the interpolations looked realistic and the reconstructions were still good quality. I further quantified this by plotting graphs of the reconstruction loss and interpolation loss separately during training. I settled on a value of λ=0.05.

To generate images of pegasuses, once my model was trained I randomly chose 64 horse images and 64 aeroplane images from the dataset. Then, each of these was encoded to produce the latent codes. The latent codes of the horses and planes were added in the weighting 0.6 to 0.4 to give the latent codes of the pegasuses. Then these were decoded to give the final output.

References

[1] D. Berthelot, C. Raffel, A. Roy, I. Goodfellow, Understanding and improving interpolation in autoencoders via an adversarial regularizer, International Conference on Learning Representations (ICLR), 2019.