Artificial Neural Networks

4th Assignment

Saeid Cheshmi – 98222027

Exercise 1

Standard autoencoders learn to generate compact representations and reconstruct their input, but aside from a few applications like denoising or sparse autoencoders, they have some limitations.

The main problem with autoencoders, for generation, is that the latent space they convert their inputs to and where the encoded vectors lie, may not be continuous or allow easy interpolation.

If you visualize an encoded vector of latent space of autoencoder you may see distinct encoding for each image type makes it far easier to decoder to reconstruct the image. This is fine because the purpose of autoencoder is replicating the same images.

On the other hand, when you're building a generative model, you want to randomly sample from the latent space, or generate variations on an input image, from continuous space.

If the latent space has discontinuities and you sample a variation from there, the decoder may generate an unrealistic output, because the decoder has no idea how to deal with that region of space.

To address this issue, we can use Variational Autoencoders. VAE, has unique property to generate random samples, its latent space is continuous which allows easy interpolation and random sampling.

VAE does this by making its encoder not output an encoding vector of size n, rather, outputting two vectors of size n, a vector of means and a vector of standard deviations. They form the vector of random variables of size n. The mean vector controls the where the encoding of an input should be centered, while std controls the 'area', how much from the mean the encoding can vary. As encodings are generated at random from anywhere inside the circle, the decoder learns that not a single point referring to a sample of that class. This allows the

decoder to not decode single, specific encodings in the latent space, but it can slightly vary too, as the decoder is exposed to a range of variations of the encoding of the same input during training.

Exercise 2

- In VAEs, what we want are encodings, all of which are as close as possible to each other while still being distinct, allowing smooth interpolation and capability of generating new samples. To do this, Kullback–Leibler divergence is used into the loss function. The KL divergence between two distribution measures how much they diverge from each other. Intuitively this loss encourages the encoder to distribute all encoding, evenly around the center of latent space. If it tries to cluster encodings in different regions, away from origin, it will be penalized.
  Using purely KL loss results in a latent space which encodings densely placed randomly, near the center of latent space regardless of their similarities. Thus, decoder finds it impossible to decode encoding into meaningful image.
- The distributions returned by the encoder are enforced to be close to a standard normal distribution because this ensures that latent space has local and global regularization (local because of the variance control and global because of the mean control). To achieve continuity (two close points in the latent space should not give two completely different contents once decoded) and completeness (a point sampled from the latent space should give “meaningful” content once decoded) in latent space, we must regularize both covariance matrix of and the mean of distributions that returned by the encoder. This regularization is done by enforcing distributions to be close to a standard normal distribution. we require the covariance matrices to be close to the identity, preventing punctual distributions, and the mean to be close to 0, preventing encoded distributions to be too far apart from each other. Hence, we’re assuming our prior follows a normal distribution, we'll output two vectors describing the mean and variance of the latent state distributions. If we must build a true multivariate Gaussian model, we need to define a covariance matrix describing how each of the dimensions are correlated. However, we'll make

a simplifying assumption that our covariance matrix only has nonzero values on the diagonal, allowing us to describe this information in a simple vector.

- First term in VAE loss or reconstruction loss allows us to separate out the classes, that should allow our decoder the ability to reproduce the data points, it has the same functionality line vanilla autoencoder loss. If we just minimize this term here's an uneven distribution of data within the latent space, means there are areas in latent space which don't represent any of our observed data. But if we optimize the two terms simultaneously, we encourage to describe the latent state for an observation with distributions close to the prior but deviating when necessary to describe salient features of the input. In figure below you can see the effect of minimizing each term of loss function in a model which trained on MNIST dataset.

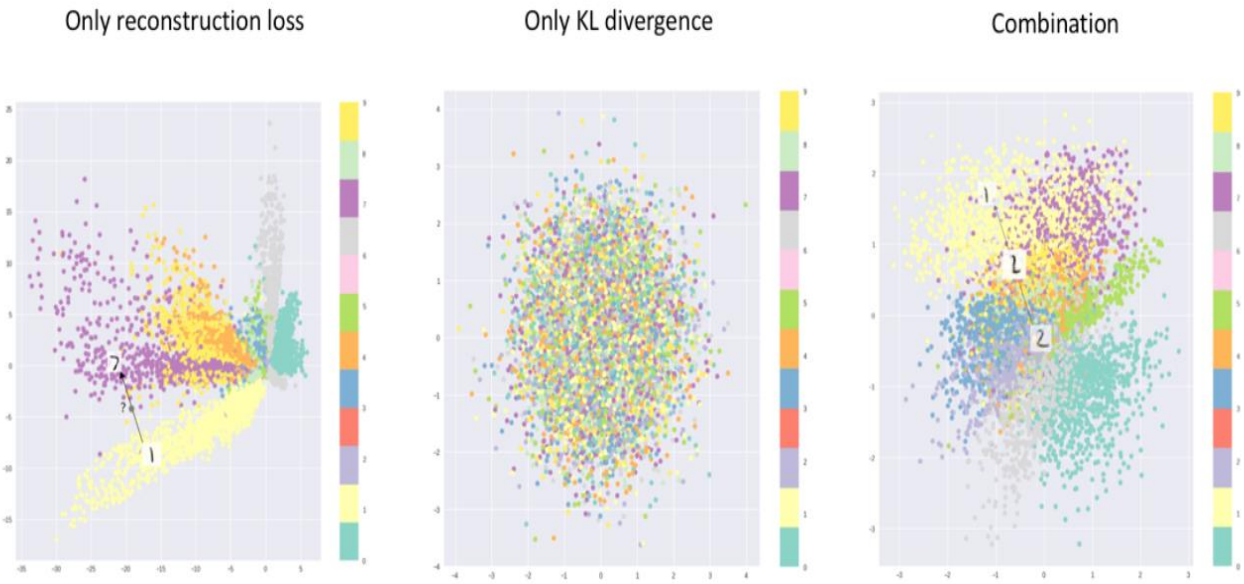

Figure 1. Effects of minimizing each term of loss

Exercise 3

In this exercise asked us to develop a model which inputs average of two images and reconstruct both of them. Here, we used CIFAR10 as our training dataset.

- **Data loading and splitting**
  CIFAR10 consists of 10 classes in order to get appropriate results we as mentioned in the exercise we just took 1000 samples from dataset and train our model on them. We split 1000 samples into two sets of 500 as in each of them we have 100 samples of each class. We wrote a custom dataset class to handle data loading efficiently and used batch size of 32 for data loader. Our dataset randomly select one pair of images and computes corresponding average then returns first image, second image and their average.
- **Model**
  It is an appropriate option to use encoder-decoder architecture to solve this task so, we tried different encoder-decoder models like Autoencoder, Variational Autoencoder and U-Net. After trying all these models, we found out that VAE had the poor results and using standard autoencoder just improved our results a little bit. After that we came to use U-Net, a well-known architecture which is used mostly for image segmentation, we implemented a U-Net with one encoder and two decoders which gives average image and outputs two image. We have 32, 64, 128, 256 convolution filters respectively in its encoder and decoder, and 512 convolution filters for its bottleneck. For training we used Adam as optimizer with learning rate of 0.001 and mean squared error as loss function. In training, we calculated first loss with respect to output of first decoder and first image and second loss with respect to second image and output of second decoder, final loss will be summation of them. We trained our model for 2500 epochs, then we got 0.0797 as final loss on train set.

- **Results**

  In below figure you can see the inference of model on a random pair. As you can see, we got almost proper result on training set, our images were reconstructed well. We also evaluated our model on test dataset, as figure 3 shows the results aren't as good as first results, because we just trained our model on 1000 samples of train set. Using the whole training set and train the model longer will result in better results.

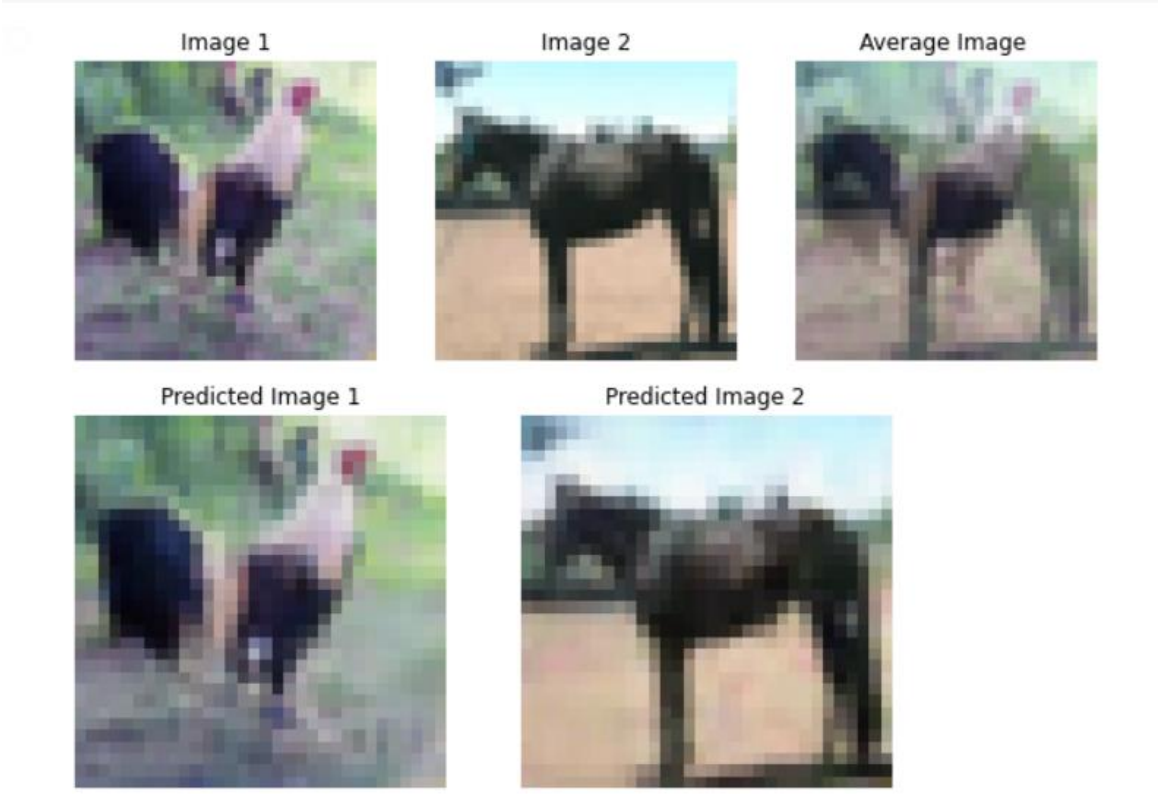

Figure 2. Reconstruction results on train set

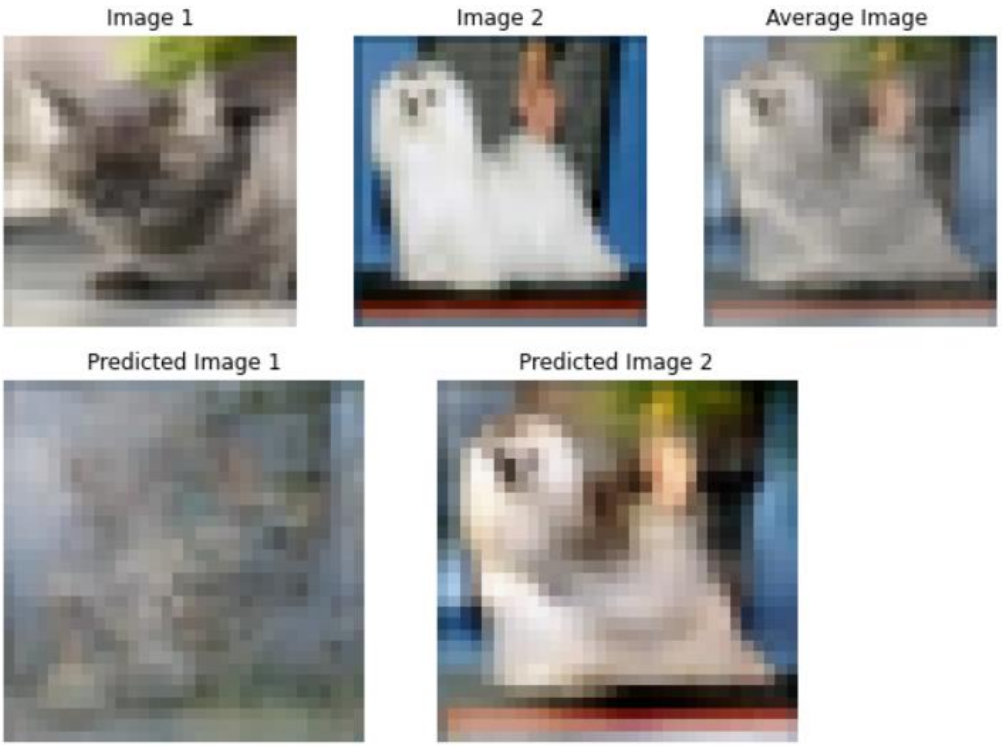

Figure 3. Reconstruction results on test set

References

- https://www.jeremyjordan.me/variational-autoencoders/
- https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73
- Ronneberger, O., Fischer, P., Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab, N., Hornegger, J., Wells, W., Frangi, A. (eds) Medical Image Computing and Computer-Assisted Intervention-MICCAI 2015