Playing with VAEs

Alexandra Albu*

Babes-Bolyai University, Faculty of Mathematics and Computer Science

October 22, 2017

Index terms— Variational Autoencoder(VAE)

1 Tasks

1. Discrete dataset: Binarized MNIST
   (a) Train VAE on MNIST dataset, assuming a Bernoulli distribution for $p(x \mid z)$, with latent size l=2 vs l=20
   (b) For l=20, do Principal Componenet Analysis (PCA) on the latent space and plot the projection on 2D; compare with same image for l=2.
2. Continuous datasets
   (a) Train MNIST with 2 different distributions for $p(x \mid z)$:
      i. Normal distribution: $\mathcal{N}(\mu, \Sigma)$ with $\mu \in (0, 1)$ - to enforce this use a sigmoid on the output layer
      ii. Logit normal distribution: logit(x) = $\ln \frac{x}{1-x}$
      for $x \in (0, 1) : p(x \mid z) = \frac{1}{x(1-x)} \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(logit(x)-\mu)^2}{2\sigma}\right)$
   (b) Frey Faces dataset
   (c) CIFAR10 dataset - the same analysis as for MNIST
3. Convolutional networks

2 Network structure

I used 2 network structures for the encoder (the decoder is symmetric):

*Electronic address: alexandralbu14@gmail.com

1. Simple network with 2 Fully connected (FC) layers
2. Convolutional network with 3 Convolutional layers:
    - 3x3 filter size, obtain 32 feature maps
    - 7x7 filter size, obtain 64 feature maps
    - 5x5 filter size, obtain 128 feature maps
      each layer is followed by a 2x2 max pooling down sampling
    - 1 FC layer

I used the ADAM optimizer and the Xavier initializer for the weights initialization. The following are given as parameters: size of latent space, size of hidden layers (for FC layers), batch size, learning rate, activation function.

3 Results

3.1 Train binarized MNIST

In this exercise I assumed a Bernoulli distribution of the output distribution $p(x \mid z)$. Since MNIST images are represented using pixel values ranging in [0, 1], the data needs to be binarized. This has been accomplished using the binomial function from the NumPy package. From official documentation: numpy.random.binomial(n, p) draws samples from a binomial distribution with n trials and probability of success p. An array x of flattened images (x has shape [nr_examples, 28*28]) was binarized with np.random.binomial(1, x). Each pixel value between 0 and 1 represents the probability of that pixel being replaced by a 1 in the binarized version (i.e. probability of success in the Bernoulli trial).

Fig. 1a shows images generated by sampling z (the latent vector) from $\mathcal{N}(0, I)$

Fig. 1b shows reconstructions of images from the test set: pass each image through the encoder to obtain the mean μ and covariance matrix Σ of the latent space; then sample from the corresponding normal distribution and pass the result through the decoder. The first column contains the original images.

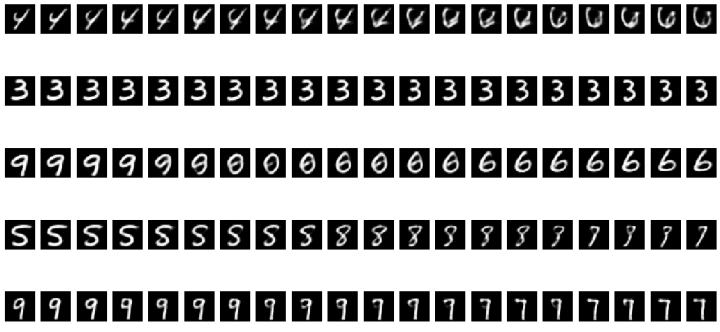
Fig. 1c shows a linear interpolation in the latent space - Choose 2 pictures P_1 and P_2 from the test set. Obtain their representation in the latent space z_1 and z_2 and compute a linear combination between the two: $\alpha * z_1 + (1 - \alpha) * z_2$ for α ranging in [0, 1]. A slow transition between the 2 pictures can be observed. Each row of Fig. 1c shows a pair of pictures drawn from the test set and the transition between them.

3.2 Visualization of latent space

Fig. 2 shows latent space representation for latent space of size 2
Fig. 3 shows a 2D projection of the 20D-latent space using PCA
Fig. 4 shows the singular values of the matrix Z of the latent representation for the test set (Z has a row for each point in the test set and a column for each

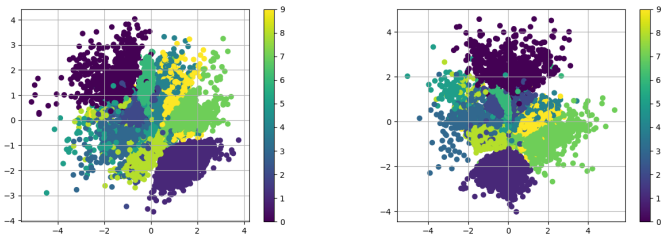
(a) binarized MNIST generated images

(b) binarized MNIST reconstructions



(c) binarized MNIST linear interpolation

Figure 1: latent size=20, 200 neurons on hidden layers, 500 epochs, ReLU activation function, batch size of 100, learning rate of 0.001



(a) 200 neurons on hidden layers, 500 epochs

(b) 500 neurons on hidden layers, 200 epochs

Figure 2: Binarized MNIST, latent size=2, ReLU activation function, batch size of 100, learning rate of 0.001

component of the latent representation). It can be observed that the largest 14 eigenvalues capture 98.5% of the information from the 20D representation.

3.3 Continuous datasets

Problems encountered: numerical issues (obtaining nan) caused (I think) by the division by σ^2 in the pdf of the normal distribution (if $\log \sigma$ is very small (negative), $\sigma^2 = 2\exp(log\sigma)$ is very close to 0, thus making $\frac{(x-\mu)^2}{2\sigma^2}$ go to ∞). (Kind of) solved the problem by adding a small quantity to σ.

Fig. 5 shows images generated from the Frey Faces dataset assuming a normal distribution of the pixels. The reconstruction of the last picture (last row) is not accurate. Also, the transitions from one picture to another are not smooth.

Fig. 6 shows Frey Faces images assuming a logit normal distribution of the pixels. Sampling from the standard normal does not generate very good images.

Fig. 7 shows generated images from the SVHN dataset. The reconstructions are not accurate.

Fig. 8 shows generated images from the CIFAR10 dataset. Neither the reconstructions nor the generated images are accurate.

4 Details for running the code

4.1 Train VAE

To train the VAE, run the script train.py with the following parameters:

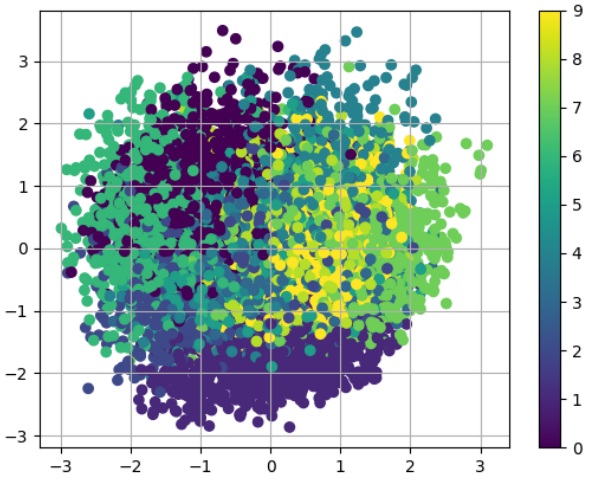


Figure 3: PCA on the space of the mean for 20D VAE(200 neuons per hidden layer, 500 epochs, ReLU activation function, batch size of 100, learning rate of 0.001)

Parameter	Description, Possible values	Default value
dataset	choose from: mnist, frey_faces, cifar10 (cifar images cropped to 8x8), cifar10_full (32x32), svhn	mnist
vae_type	type of network used choose from: flat_binarized, flat_cont_normal, flat_cont_logit, conv3	flat_binarized
latent_dim	Number of neurons used for latent space representation	20
num_epochs	Number of epochs	500
learning_rate	Learning rate	0.001
batch_size	Batch size	100
hidden_size	Number of neurons on hidden layers	200
restore	True if you want to continue training an already saved model, False otherwise	False

Example:

```
python train.py --dataset frey_faces --vae_type flat_cont_normal
```

will train the model on the Frey Faces dataset with a neural network having 2 fully connected layers each with 200 neurons, for 500 epochs with a learning rate of 0.001 and batch size of 100.

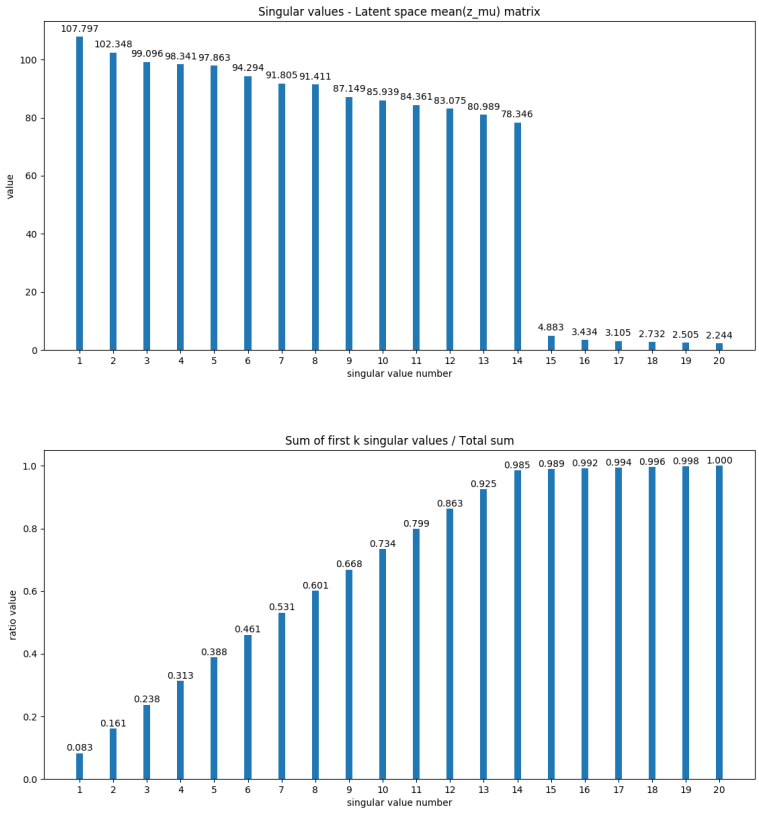
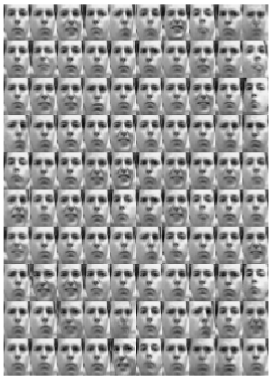
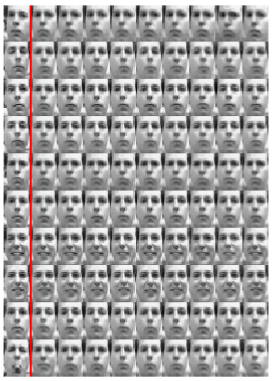


Figure 4: Binarized MNIST, latent size=20, ReLU activation function, batch size of 100, learning rate of 0.001, 500 neurons per hidden layer, 200 epochs



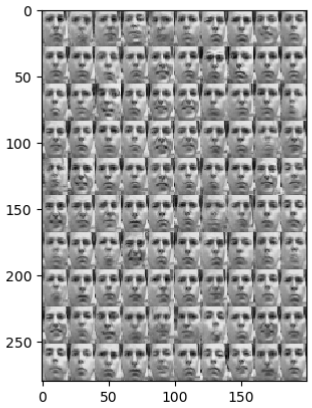
(a) Frey Faces generated images, normal distribution



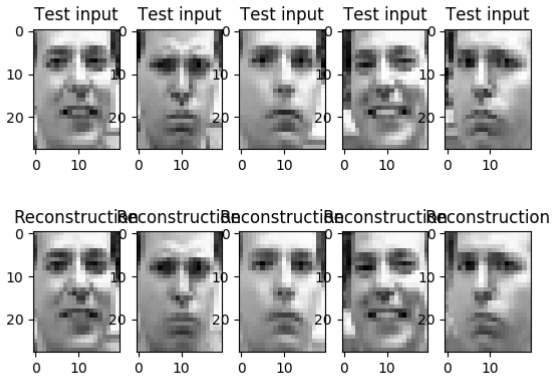
(b) Frey Faces reconstructions, normal distribution

(c) Frey Faces interpolation, normal distribution

Figure 5: latent size=20, 200 neurons on hidden layers, 500 epochs, ReLU activation function, batch size of 100, learning rate of 0.001



(a) Frey Faces generated images, logit normal distribution



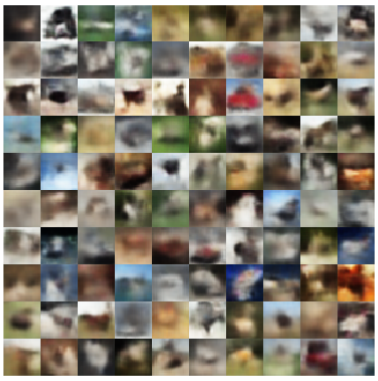
(b) Frey Faces reconstructions, logit normal distribution

Figure 6: latent size=20, 500 neurons on hidden layers, 200 epochs, ReLU activation function, batch size of 100, learning rate of 0.001

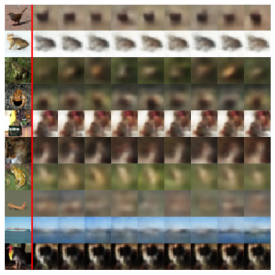
(a) SVHN generated images

(b) SVHN reconstructions

Figure 7: latent size=20, 500 epochs, ReLU activation function, batch size of 100, learning rate of 0.001, convolutional network



(a) CIFAR10 generated images



(b) CIFAR10 reconstructions

Figure 8: latent size=20, 1000 epochs, ReLU activation function, batch size of 100, learning rate of 0.001, convolutional network

4.2 Latent space visualization

To plot the latent space visualizations run

- plots_latent_space.py with the same parameters as train.py, except for the "restore" parameter (the model is always restored)
- plots_latent_space_2d.py with the parameters: dataset, vae_type, num_epochs, learning_rate, batch_size, hidden_size defined as for train.py

4.3 Singular values plots

To plot the singular values run the script plot_singular_values.py. The parameters are the same as for train.py, except for the "restore" parameter (the model is restored)

4.4 Generate images

To generate the 3 types of pictures from above (generate images by sampling z, reconstruct images from test set and obtain a linear interpolation between 2 images) run the script make_pictures.py. The parameters are the same as for train.py, except for the "restore" parameter. Here, the model is restored from a previously saved one.

Before running the code, fix the path to the folder where/from where the checkpoints will be saved/restored as well as the path to the folder where the pictures will be saved.