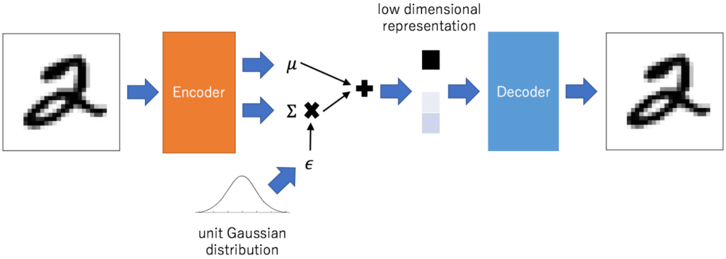
ConVAE Results

Breno Orzari

Sprace

Variational Autoencoder (VAE)

- A Variational Autoencoder is a neural network that has the following structure:



- Its main feature is the capability of generating new outputs through the encoding of the training inputs into a low dimensional representation
- The cost function is given by two terms:
  - The KL divergence and the Reconstruction Likelihood (or reconstruction loss)

Reconstruction Loss Term in Loss Function

- Two different approaches
  - Use the pixel intensity as the third axis of a 3D space (third axis):

$$\sum_i min\ [d_E(p_i,\ \hat{p})]^2 + \sum_i min\ [d_E(p,\ \hat{p}_i)]^2 \tag{1}$$

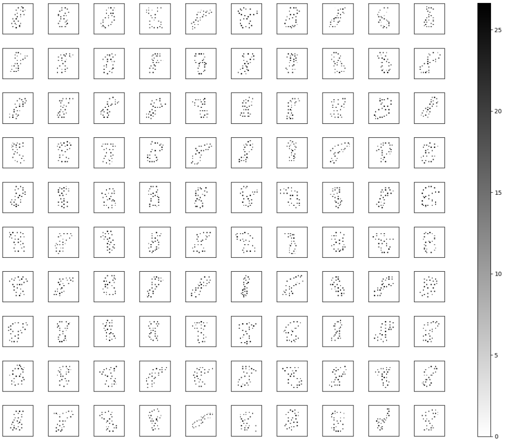
    - $p = (x,\ y,\ I)$ and $\hat{p} = (\hat{x},\ \hat{y},\ \hat{I})$ are input and output pixels features respectively;
  - Suggested by Jean-Roch (JR way):

$$\sum_i min\ [d_E(x_i,\ \hat{x})]^2 + \sum_i min\ [d_E(x,\ \hat{x}_i)]^2 + \sum_i (I_i - \hat{I}_{\hat{k}})^2 + \sum_i (I_k - \hat{I}_i)^2 \tag{2}$$

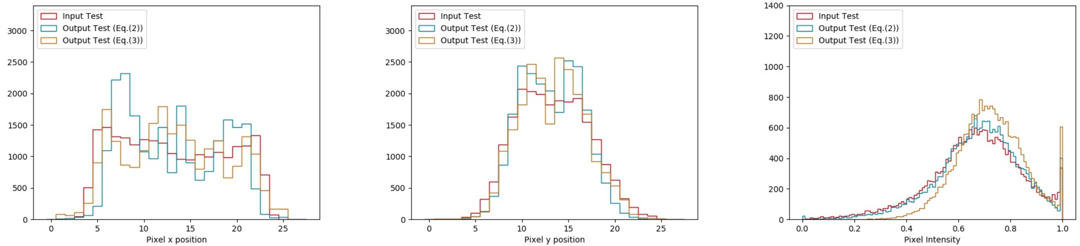
    - $x = (x,\ y)$ and $\hat{x} = (\hat{x},\ \hat{y})$ are the positions of input and output pixels respectively;
    - d_E is the euclidean distance;
    - I and $\hat{I}$ are the intensities of input and output pixels respectively;
    - $\hat{k} =\ argmin\ [d_E(x_i,\ \hat{x})]^2$ and $k =\ argmin\ [d_E(x,\ \hat{x}_i)]^2$;

MNIST Superpixels Dataset

- Only 25 pixels with intensities greater than 0
- Very sparse
- Using only the digit 8
  - The idea is to train other networks for other digits



Best β (from intensity comparison) for TA loss: 0.3



What to try next

- □ Both loss functions are good;
- □ What to try next:
  - Different KL divergence factors in the loss function

$$\text{Loss} = L^{rec} + \beta L^{D_{KL}} \tag{3}$$

    - Problems with reproducibility
  - Normalizing flow technique

Backup

Neural Network for the MNIST Superpixels Dataset

- In the Conv layers: kernel = (1,5) ((3,5) only in input and output layers), stride = (1), padding = (0,1); the order is $channels \times rows \times columns$ (also for ConvTranspose)
- No dropout or pool (for now)
- $ReLU$ activation function after all Conv and ConvTranspose (the output is an exception, it has sigmoids to constrain the values of the features), the first dense layer, and all dense layers that appear after the latent vector