בס"ד

Deep Generative Models – Final Project

Daniel Levi | Adi Cohen

Abstract:

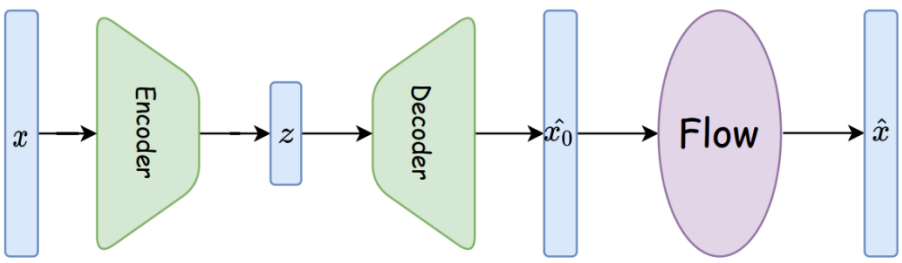
VAE models have a bottleneck to transform high dimensional representation into a much lower one - an Encoder network. The low dimension in many cases is described by a simple distribution such as an i.i.d gaussian samples. This distribution in many cases does not correctly represent the distribution of the data. In this case, the decoder's output gaussian samples may be different than the original samples. To obtain a more powerful decoder output, we would like to add a more complex dependency between the pixels by combining a flow model in the VAE output. The model is validated on MNIST and Fashion MNIST datasets and compared to separate methods of VAE and NICE (flow).

Introduction:

In our day's deep neural network architecture are good at various tasks. The fact that the amount of data is available and getting large improves the models' performance. Today there are models that are very effective at learning compressed representations, such as Variational Autoencoders (VAE) and Flow architectures. These neural networks are good as inference networks to model the latent data distributions and therefore can be good generative models. Although successful, part of these models usually initializes a Gaussian prior distribution with the assumption that each latent variable is sampled independently, and the output samples are also gaussian, it does not always describe the right scenario. Ignoring the dependencies between variables can limit the model's flexibility to fit the data.

A common approach that we saw to improving VAE performance is to add flexible inference networks in the bottleneck, like flow models. This helps the model to learn a mapping from a simple posterior distribution to a more complex posterior. This approach was done in many works and achieve a good improvement.

In this work, we propose to improve VAE performance by combining a flow model after the VAE, in the VAE's output. We expect to have a better performance than the VAE with gaussian samples at the output and flow model separately.



Method:

Our method is to add more dependency between the pixels by combining a flow model in the VAE output. We choose NICE as our flow model.

We want to compare the model performance to the subnet that build it, VAE and NICE.

For comparison, we built each subnet individually with as similar parameters as possible. Moreover, the loss functions of each model are defined with a Gaussian distribution.

The loss function for each model:

VAE:

$$ELBO(\theta,\lambda) = \sum_t \left\{ logN\left(x_t; \mu_\theta(z_t), \Sigma_\theta(z_t)\right) - D\left(N\left(\mu_\lambda(x_t), \Sigma_\lambda(x_t)\right)||N(0,1)\right)\right\}$$

Process:

$$\tilde{z}_t \sim N(0,1)$$

$$x_t \overset{encoder}{\rightarrow} \mu_\lambda(x_t), \Sigma_\lambda(x_t) \rightarrow z_t = \tilde{z}_t \cdot \sigma_\lambda(x) + \mu_\lambda(x) \overset{decoder}{\rightarrow} \mu_\theta(z_t), \Sigma_\theta(z_t)$$

Note: The loss function commonly used in the community for the VAE model is the Binary cross-entropy. This loss function helps in training and significantly improves the samples. Because we want to compare this model to our integrated model, we used this loss function.

Flow:

$$\log\left(p_{X_0}(x)\right) = \sum_t \left\{ \log\left(p_{X_T}\left(f(x_t)\right)\right) + \sum_{j=1}^{k} \log\left(\left|\det\left(\frac{\partial f_j(x_t)}{\partial(x_t)}\right)\right|\right)\right\}$$

$$\overset{NICE}{=} \sum_t \left\{ \log(N(0,1)) + \sum_{j=1}^{k} \log\left(\left|s(x_{1,j})\right|\right)\right\}$$

Where f is a bijection mapping from x_i to x_k.

Process:

$$x_{t,0} \overset{flow}{\rightarrow} x_{t,T}\ ;\ x_{t,T} \sim N(0,1)$$

VAE with Flow:

$$ELBO = \sum_t \left\{ q(z_t|x_t;\lambda) \log\left(p\left(x_{t,T}\middle|z_t;\theta\right)\right) - D\left(q(z_t|x_t;\lambda)||\, p(z_t;\theta)\right)\right\}$$

$$= \sum_t \left\{ N\left(\mu_\lambda(x_t), \Sigma_\lambda(x_t)\right) \log\left(N\left(\mu_\theta(z_t), \Sigma_\theta(z_t)\right) \prod_{j=1}^{T} \left|det\left(J^{-1}(x_{j-1})\right|\right.\right.$$

$$\left.\left. - D\left(N\left(\mu_\lambda(x_t), \Sigma_\lambda(x_t)\right)||N(0,1)\right)\right\}\right.$$

Process:

$$\tilde{x}_t, \tilde{z}_t \sim N(0,1)$$

$$x_t \to \mu_\lambda(x_t), \Sigma_\lambda(x_t) \to z_t = \tilde{z}_t \cdot \sigma_\lambda(x) + \mu_\lambda(x) \to \mu_\theta(z_t), \Sigma_\theta(z_t) \to x_{t,0}$$

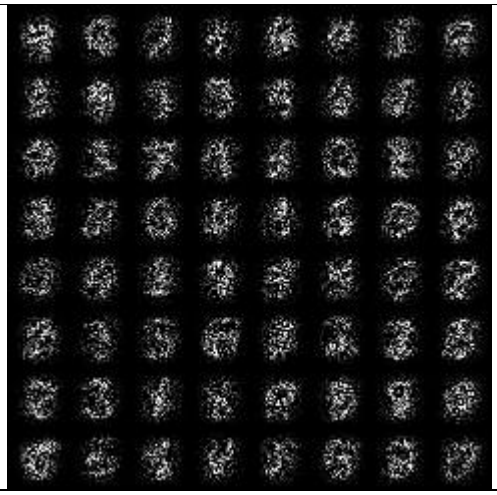
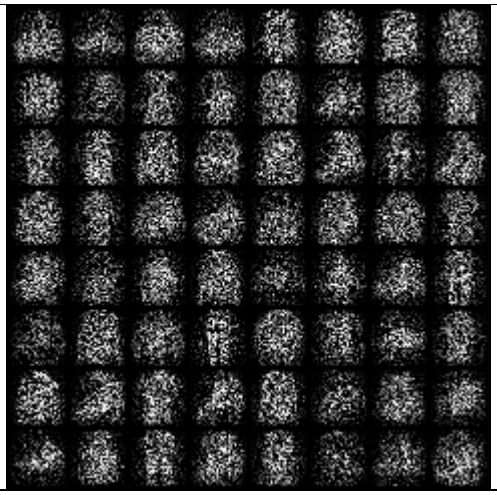
$$= \tilde{x}_t \cdot \sigma_\theta(x) + \mu_\theta(x) \overset{flow}{\to} x_{t,T}$$

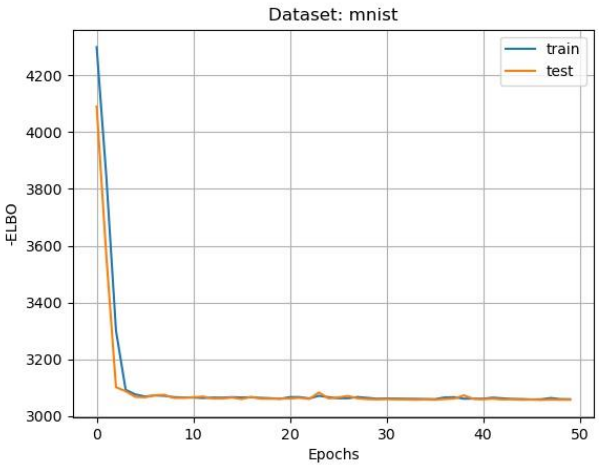
Results:

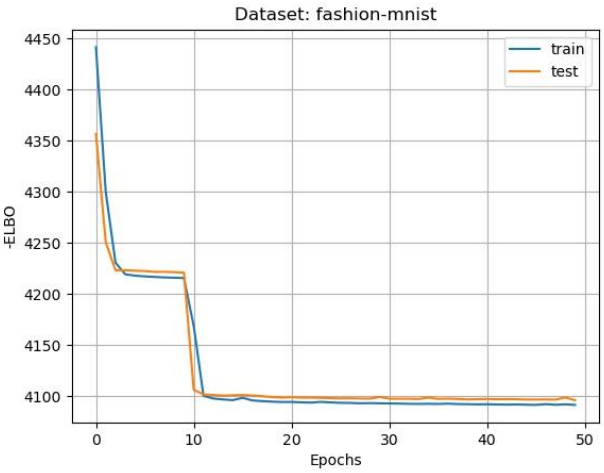
The models were tested on MNIST and Fashion MNIST data sets.

VAE model:

The results from the VAE model

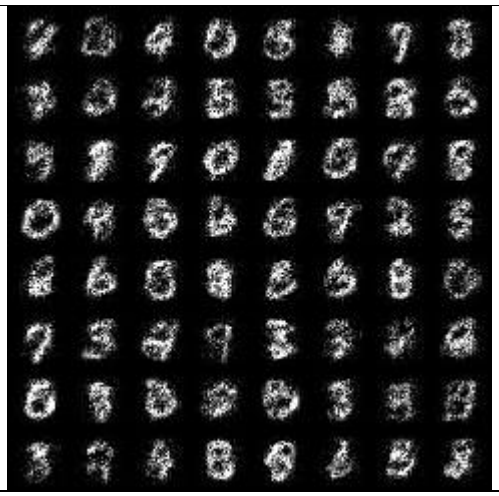
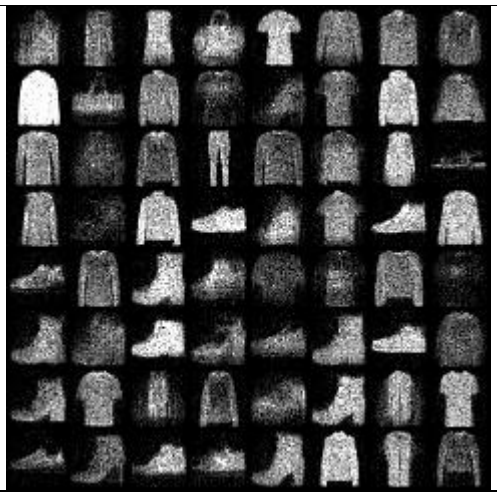
NMIST	Fashion - NMIST
	

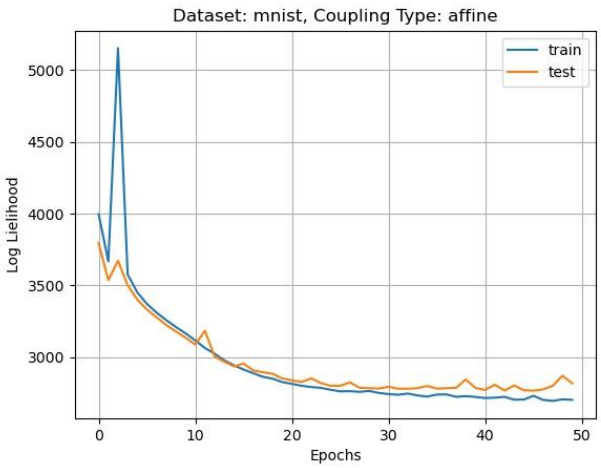


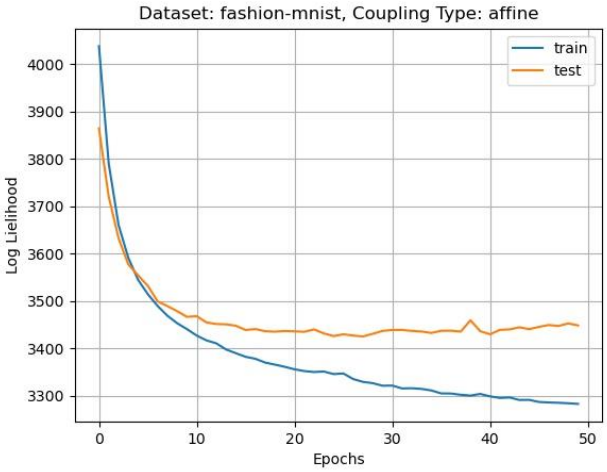


We can see that the model does not have a good result, the gaussian samples did not fit to generate pixels with a strong dependence.

The results from the NICE model

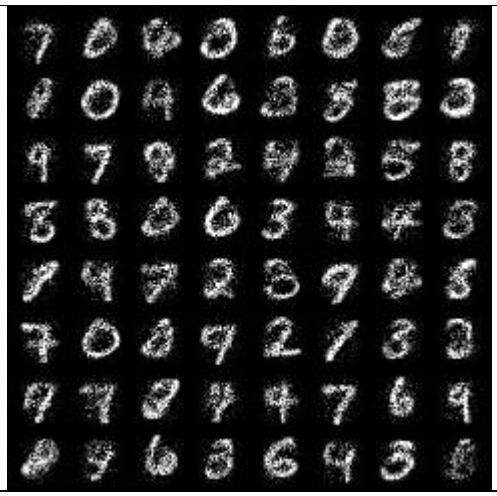
NMIST	Fashion - NMIST
	

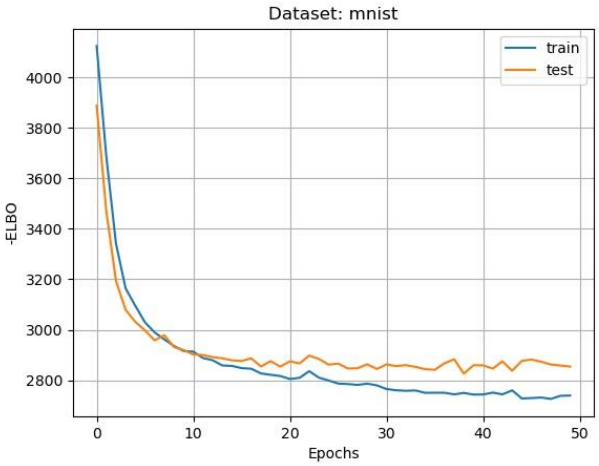


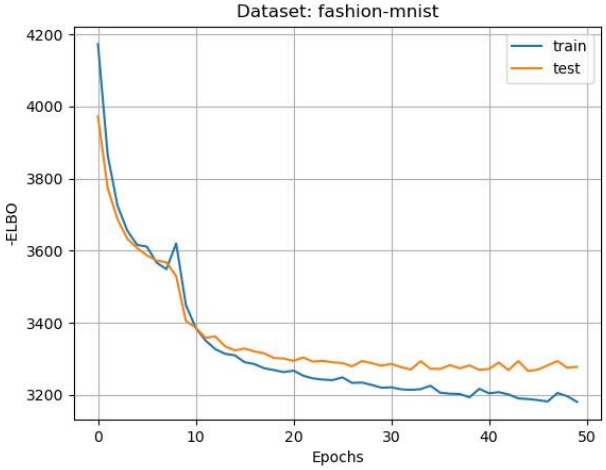


We can see that the model has a better result than the VAE model.

The results from our proposed model (VAE + NICE)

NMIST	Fashion - NMIST
	





Our method has significantly better results than the results obtained by the VAE model. In addition, our method yields only a slight improvement compared to the results obtained for the NICE model.

Note:

We tried to improve the results by changing the layer type, and network parameters, and by adding the sigmoid function. These experiments did not yield an improvement in results.

Code:

The code and the results can be found at the following link:
https://github.com/levidaniel96/VAE-NICE

Reference:

1. *Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013).*
2. *Dinh, Laurent, David Krueger, and Yoshua Bengio. "Nice: Non-linear independent components estimation." arXiv preprint arXiv:1410.8516 (2014).*