Supervised Variational Autoencoder

Phong Hoang, Duc Do Huu Nguyen [*]

December 19, 2018

Abstract

Generation and classification of images are among the most important tasks in the field of Computer Vision. Each problem by itself has been studied extensively over the years. In this work, we study the combined task of image generation and classification in the supervised setting. A natural approach to this task is Variational Autoencoder (VAE). We study and compare 2 VAE-inspired model: unsupervised VAE + Neural Network classifier (VAE plus second stage MLP classifier - VAENN) and Semi-supervised VAE modified for the supervised setting (SVAE). We show that VAENN model performs better in image generation but worse in classifying images compared to SVAE. We prove this experimentally and explain the intuitions behind the result. Since each model has its strengths and weaknesses, this result guides us to choose the appropriate model depending on the application.

1 Motivation

Image generation and classification have always been of great interest to Computer Vision researchers. Each task, if successful performed, can be used in a wide variety of application. Image classification is already employed in multiple sectors such as social media, tourism, resale and even self-driving vehicles. On the other hand, image generation has found its application in the game and movie industry. Therefore, each task has been studied thoroughly. However, the combined task of image generation and classification, is a less often studied problem. This problem only arises recently with the trend in the game industry towards ”sandbox” games, also known as open world games. True to their names, these games create a virtual environment in which the player can interact with any of the millions of objects in the environment. Without the aid of machines, the cost to create these games is prohibitive. This motivates the need for computer generated images as well as the classification of these images.

For problems of this kind, the input are images and we want to generate new images from the given images and identify their classes. More rigorously, given an image X, we want to get a reconstructed image X' and its label y.

A natural approach to these problems is the VAE model, introduced by Kingma et al. 2013 [1]. VAE is a type of artificial neural network used to learn efficient data encoding in an unsupervised manner. The aim is to learn a representation (encoding) for a set of data, typically for dimensionality reduction. Both methods that we use to compare in this paper are based on the VAE model. These methods are introduced by Kingma et al. 2014 [2]. The model VAENN we use in our work in based on the M1 model [2], in which an unsupervised VAE is trained first, and then the data is transformed into the low dimensional code space, which is used as input for the neural network classifier. The other model, SVAE, is based on the M2 model [2], which is a semi-supervised VAE. The M2 model deals with unlabelled data by letting the label be a latent variable so that the model can perform inference on. We modify M2 model slightly by removing this part of the model that deal with unlabeled data, thus turning it into a supervised model.

Our goal is to compare the performance of two different models VAENN vs SVAE. The two models both use VAE as the base model, but they make different assumptions about the latent distribution. VAENN assumes that the latent variables z carries information about the class labels, and because the prior is usually a simple distribution in low dimension, we can easily classify the labels in z-space. On the other hand, SVAE attempts to train both the generation and classification of images at the same time. It assumes that the latent variables z and the class labels y are independent, and they jointly generate the image X. Our hypothesis

[*]Department of Computer Science, Halligan Hall, Tufts University, Medford, MA 02155. {*Duc.Nguyen, Phong.Hoang*} @tufts.edu

is that SVAE will give better classification accuracy than VAENN because when images are encoded into low dimension z, they lose information. Therefore using encoding z as input for the classifier will result in worse accuracy then using the entire image to classify as in the SVAE model. For image generation and reconstruction, since VAE makes fewer assumption about z, it will produce better quality images. The assumption that z and y are independent made by the SVAE model is too strong and does not hold on most real world datasets.

2 Methods

2.1 VAENN

An unsupervised VAE is a generative model: it has 2 components an encoder and a decoder. The encoder encodes data X from the input space into a code space z and the decoder transforms z from code space back to input space. The VAE is trained separately. After training the VAE, the encoder transforms data into code space and then a neural network classifies the data using input code space z and label y. Mathematically, assume that each data point x_n has a latent code vector z_n. The prior distribution over this vector is a standard Normal:

$$p(z_n) = Normal(z_n|0, \mathbb{I})$$

We can write a probabilistic model for generating a data vector x_n given its code vector z_n:

$$p(x_n|z_n, \theta) \sim Bern(x_n|decode(z_n, \theta))$$

where θ is the parameter of the decoder network. The negative ELBO is then minimized:

$$logp_\theta(x) \geq \mathbb{E}_{q_\Phi(z|x)}[logp_\theta(x|z) + logp(z) - logq_\Phi(z|x)] \tag{1}$$

where $q_\Phi(z|x) = Normal(z|\mu, \sigma)$ with μ as the mean of the Normal distribution and σ as the diagonal co-variance matrix. Both μ and the diagonal of σ are the output of the encoder network $encode(x_n, \Phi)$. For MNIST dataset, the architecture of the encoder and decoder we are using are as followed: 2 fully-connected 500 nodes layer followed by tanh activation function with dropout rate 0.1. For SVHN dataset, we make use of convolutional layers: the encoder consists of 3 convolutional layers with number of filters 128, 256, 512 respectively. Each convolutional layer has kernel size 2 and strides 2 with ReLU activation and batch normalization after each layer. At the end there are 2 fully-connected layer giving the mean and the diagonal of $q_\Phi(z|x)$. The decoder architecture is similar to the encoder, except that it makes use of deconvolutional layer [3] and the dense layer is replaced by a convolutional layer with 3 filters, kernel size and stride 1 and sigmoid activation.

In the second step, we use z as input to the neural network and train the neural network. For both MNIST and SVHN dataset, we use 4 fully-connected layers with 1000 nodes and dropout rate 0.1.

The neural network is minimized using categorical cross entropy loss. Both VAE and the neural network will be trained with Tensorflow ADAM optimizer [4] and mini-batch gradient descent with batch size 128 and learning rate 1e-4. A held-out test set is used to evaluate the performance of both the VAE and the neural network classifier.

For baseline classifiers, we use Random Forest with number of decision trees equals 25, multiclass SVM with RBF kernel and one vs one strategy and k-nearest neighbors with k equals 30. We use the default parameters of these algorithms given by scikit-learn package [5].

2.2 SVAE

Let x be the generated data, y be latent class variable and z be continuous latent variable. The prior for this model is:

$$p(y) \sim Cat(y|\pi)$$

where π is the parameter of the categorical distribution, a vector of probability of any given class.

$$p(z) \sim Normal(z|0, \mathbb{I})$$

The likelihood for this model is:

$$p_\theta(x|y,z) \sim Bern(x|decode(z,y,\theta))$$

Our objective, modified from the objective for the semi-supervised setting:

$$logp_\theta(x,y) \geq \mathbb{E}_{q_\Phi(z|x,y)}[logp_\theta(x|y,z) + logp_\theta(y) + logp(z) - logq_\Phi(z|x,y)] = -\mathbb{L}(x,y) \tag{2}$$

$$J = \Sigma_{(x,y)}\mathbb{L}(x,y) \tag{3}$$

$$J = J + \alpha E_{(x,y)}[-logq_\phi(y|x)] \tag{4}$$

where (x,y) are all training data points and α is a hyper parameter. $q_\Phi(z|x,y)$ is defined similarly to the VAE case. (2) is modified from (1), to add the label y to the objective so we can train a network that learn both the distribution of x and y. The second term of objective (4) is the classifier $q_\phi(y|x)$: it is added to allow the model to train the classifier and decoder and generator concurrently. This equation is also theoretically motivated: it can be obtained by doing inference over the parameter of the categorical distribution π with symmetric Dirichlet prior.

For MNIST dataset, the architecture of SVAE encoder and decoder is exactly the same as the architecture of VAE, except for that both SVAE encoder and decoder also take y as input. The classification network $q_\phi(y|x)$ has the same architecture as the classifier in the VAENN model. For SVHN dataset, SVAE also makes use of convolutional layers: the encoder and decoder part of SVAE is exactly the same as the VAE. However, the architecture of $q_\phi(y|x)$ is different: we use 3 convolutional layers with filter size 128, 256, 51 respectively. Each convolutional layer has kernel size 2 and stride 1 and is followed by a max pooling layer with pool size 2. The output of the pool layer is batch normalized and has activation function ReLU. At the end of the network, there are 2 fully-connected layers with output size 512, 256 and dropout rate 0.5, 0.3 respectively. The final layer of the network is a Softmax layer.

The optimizer RMSProp [6] is used with learning rate 1e-4. Batch size is 128. All the code for SVHN dataset is implemented using Keras [7].

For initialization, the priors are defined as written before. All the initial weights of neural networks are randomized, and for SVAE objective, we find empirically α equals 25 for MNIST dataset, and α equals 1 for SVHN dataset. A held-out test set is used to evaluate the generation and accuracy performance of SVAE.

All the hyperparameters are chosen based on the performance of the models on the held-out test set.

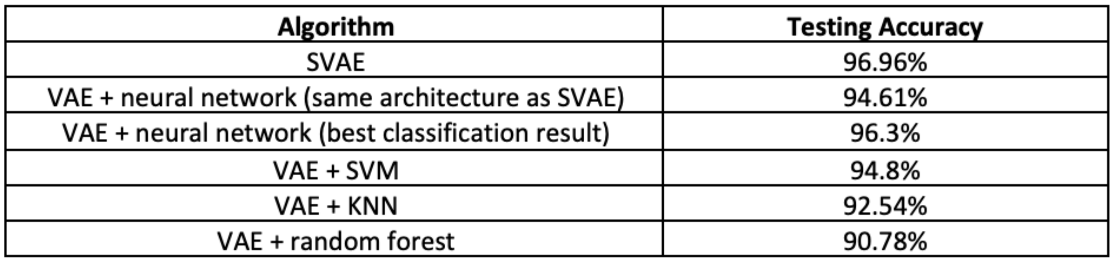
3 Experiments

3.1 MNIST

The baseline dataset we use is MNIST dataset [8]. It is a dataset of handwritten digits 0-9. Each example in the dataset is a grayscale image of size 28x28x1 along with a label. There are 60000 images in the training set and 10000 images in the test set.

We train both the VAENN model and the SVAE model for 30 epochs.

We test our learned models on our held out MNIST test set. Here is the table comparing the performance of VAE + different baseline classifiers and VAENN and SVAE. The VAE + classifiers and VAENN is trained on the code space z with dimension 50.



Algorithm	Testing Accuracy
SVAE	96.96%
VAE + neural network (same architecture as SVAE)	94.61%
VAE + neural network (best classification result)	96.3%
VAE + SVM	94.8%
VAE + KNN	92.54%
VAE + random forest	90.78%

Figure 1: classification result of VAE + classifiers and VAENN with input code space of dimension 50 and SVAE on MNIST dataset

As we can see, SVAE outperforms all other models even VAENN. However, we note that the architecture that we use for both VAENN and for SVAE is used with optimization of SVAE in mind. Therefore it is not the most fair comparison. We also note that by changing the architecture of the neural network of the VAE model, we can achieve an accuracy comparable with that of the SVAE model, as shown in the table.

We also change the dimension of the code space z and see how the dimension affects the performance of the 2 models.

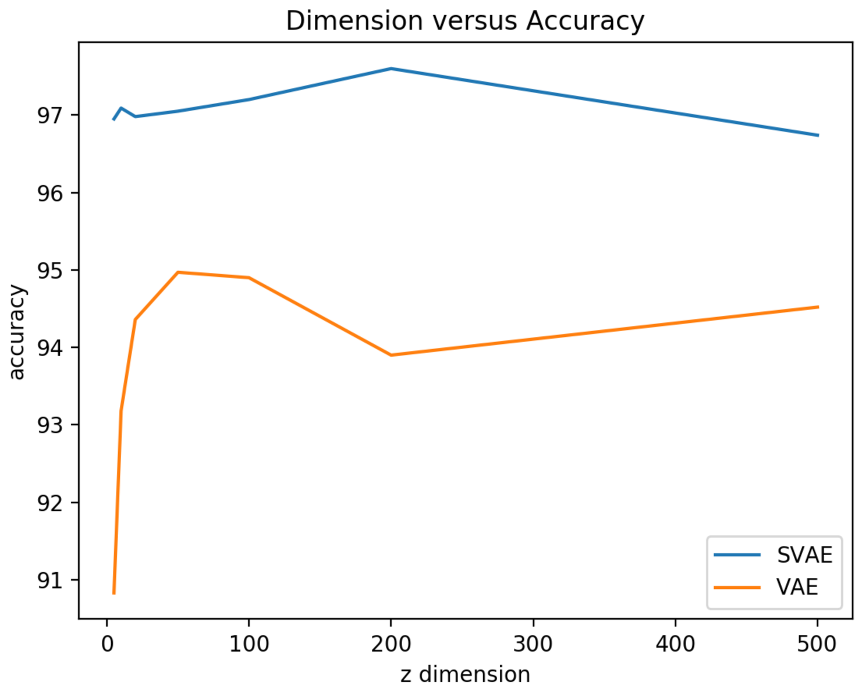


Figure 2: classification accuracy of both VAENN and SVAE with different dimension of code space z on MNIST dataset

As we can see, SVAE outperforms VAENN with all dimensions of z. SVAE classification accuracy is roughly the same regardless of the hidden dimension z, which means that it is very robust. VAENN does not perform as well, possibly because if the dimension is low, than it may lose lots of information when being encoded, but as the dimension increases, the accuracy stays stable in the range 94-95%. This implies that increasing the dimension of z after a certain point does not gain us any extra information. We believe this is because of the prior constraint, forcing z to move towards a Gaussian distribution, which is not necessary the same distribution as that of the images. Therefore, as long as we train on z, we will lose some information and therefore we get worse classification accuracy than SVAE model. The data supports our hypothesis.

Next, we show the generation result for both VAE and SVAE with different dimensions of z.

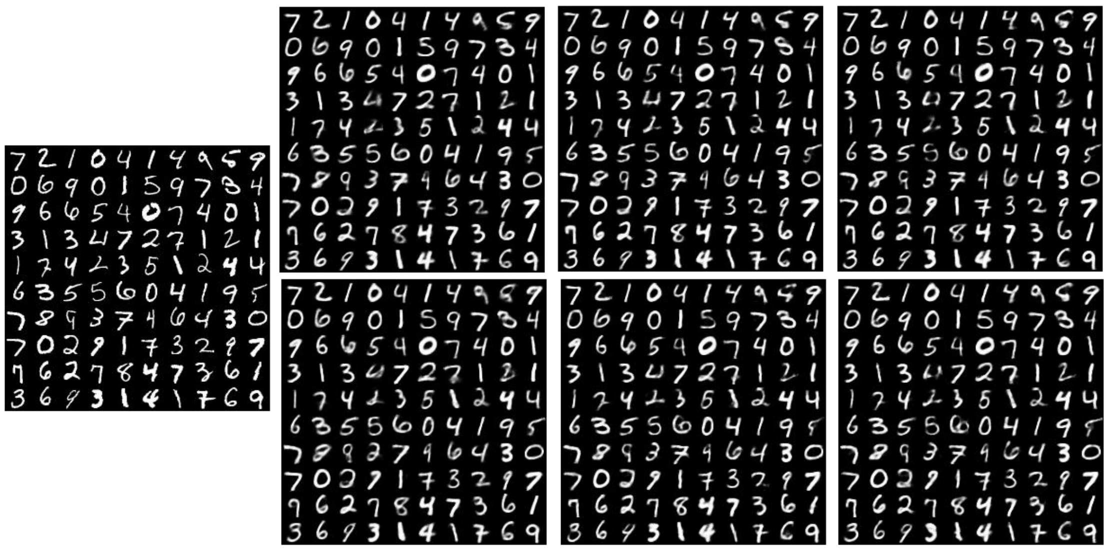


Figure 3: generation result for both VAE and SVAE with different dimension of z on MNIST dataset. First image is the input, and on the first row are SVAE-reconstructed images with z dimension 10,50, 200. On the second row are VAE-reconstructed images with z dimension 10, 50, 200. The generation quality is quite good. The digits are sharp and not blurry. To the human eyes, there are not a lot of difference in these results for VAENN and SVAE

Next, we look at the RMSE result for both VAE and SVAE with different dimensions of z

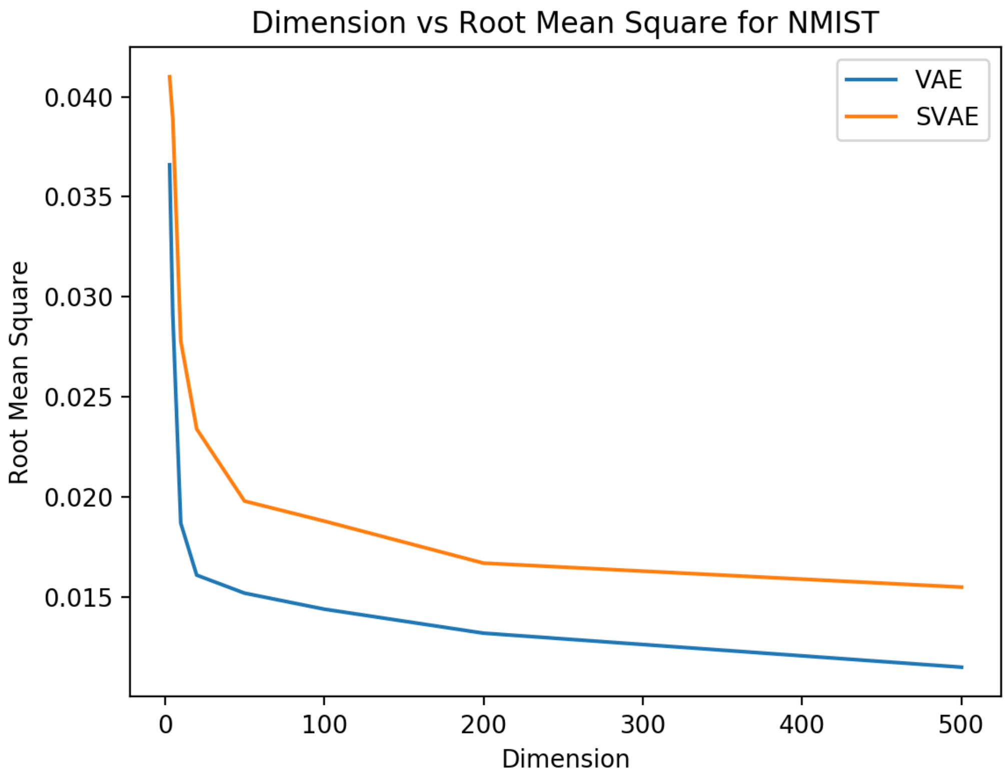


Figure 4: RMSE for VAE and SVAE with different dimensions of z on MNIST dataset

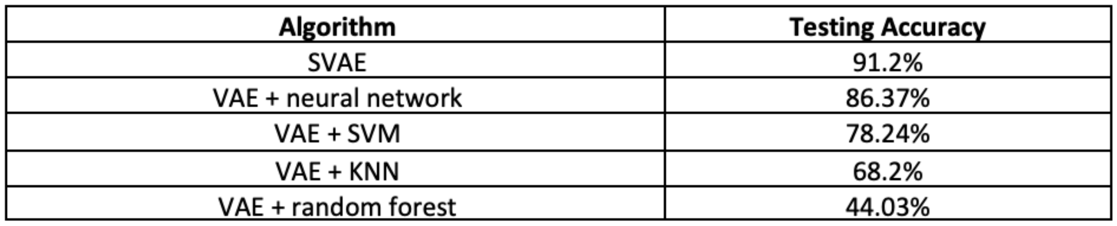
From the figure, clearly SVAE outperforms VAE in RMSE metric. For both VAE and SVAE model, RMSE reduces as the z dimension increases. This makes sense as larger z means we lose less information when we encode, therefore the reconstructed image will be more similar to the original image. The intution behind why VAE outperforms SVAE in image generation is that the assumption of SVAE that label y and code z are independent are really strong and not easily satisfied. Another reason is that while VAE model only optimizes for image generation, SVAE optimizes for both image generation and classification, therefore the quality of the image is lower.

3.2 SVHN

The focus dataset we are using is SVHN dataset [9]. It consists of 73257 32x32x3 images in the training set and 26032 32x32x3 images in the test set. Each image is a color image of street view house digits 0-9.

We train both the VAENN model and the SVAE model for 30 epochs.

We test our learned models on our held out MNIST test set. Here is the table comparing the performance of VAE + different baseline classifiers and VAENN and SVAE. The VAE + classifiers and VAENN is trained on the code space z with dimension 100.



Algorithm	Testing Accuracy
SVAE	91.2%
VAE + neural network	86.37%
VAE + SVM	78.24%
VAE + KNN	68.2%
VAE + random forest	44.03%

Figure 5: classification result of VAE + classifiers and VAENN with input code space of dimension 100 and SVAE on SVHN dataset

As we can see, SVAE and VAENN outperform other methods by a huge margin. Given the complicated distribution of the SVHN dataset, this is expected. SVAE outperforms VAENN in accuracy. This supports our hypothesis that when we train a classifier on the code space z, some of the information is lost because we transform the image into low dimension z. One particular information that is lost is image locality. For SVHN dataset, we use convolutional neural network, and because z is not an image, we cannot use convolutional neural network to classify z in the VAENN pipeline. On the other hand, we use convolutional neural network to classify image in SVAE model, which allows it to retain image locality. This causes a huge difference in the classification performance of VAENN and SVAE.

We also change the dimension of the code space z and see how the dimension affects the performance of the 2 models.

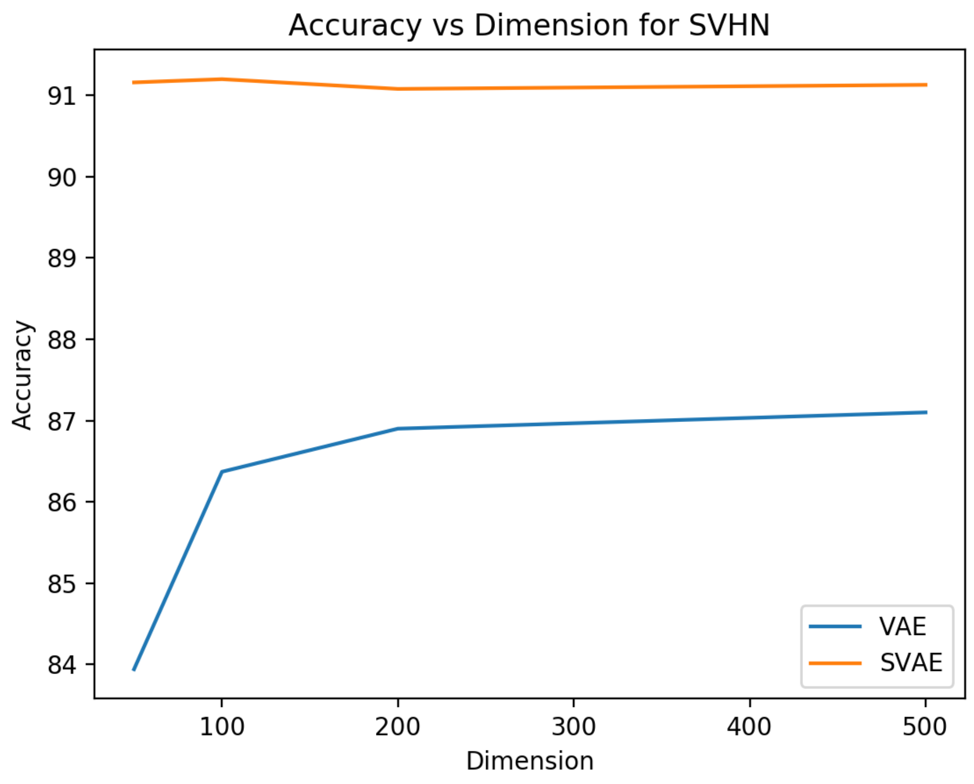


Figure 6: classification accuracy of both VAENN and SVAE with different dimension of code space z on SVHN dataset

From Figure 6, we can see that accuracy of SVHN is very stable with different dimension of z. We believe this is because for SVAE, we always get extra label information from the classifier $q_\phi(y|x)$ no matter the dimension of z. For VAE, with low dimension of z, the accuracy is low but increased as the dimension of z increases. This makes sense as with larger dimension of z we lose less information. SVAE outperforms VAE in classification because when we train on low dimensional space z, some information is already lost. One such information is image locality as mentioned in previous analysis.

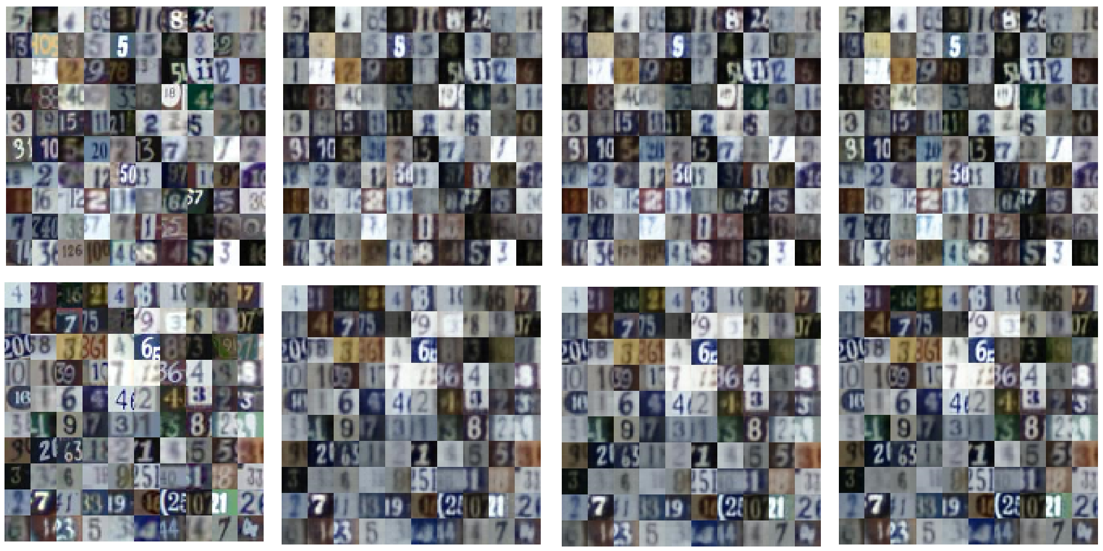


Figure 7: generation result for both VAE and SVAE with different dimension of z on SVHN dataset. First image is the input, and on the first row are SVAE-reconstructed images with z dimension 100, 200,500. On the second row are VAE-reconstructed images with z dimension 100,200,500. The reconstructed images are quite blurry for z = 100 but become quite good for larger z. However, we can see that VAE reconstruct higher quality images

Next, we look at the RMSE result for both VAE and SVAE with different dimensions of z.

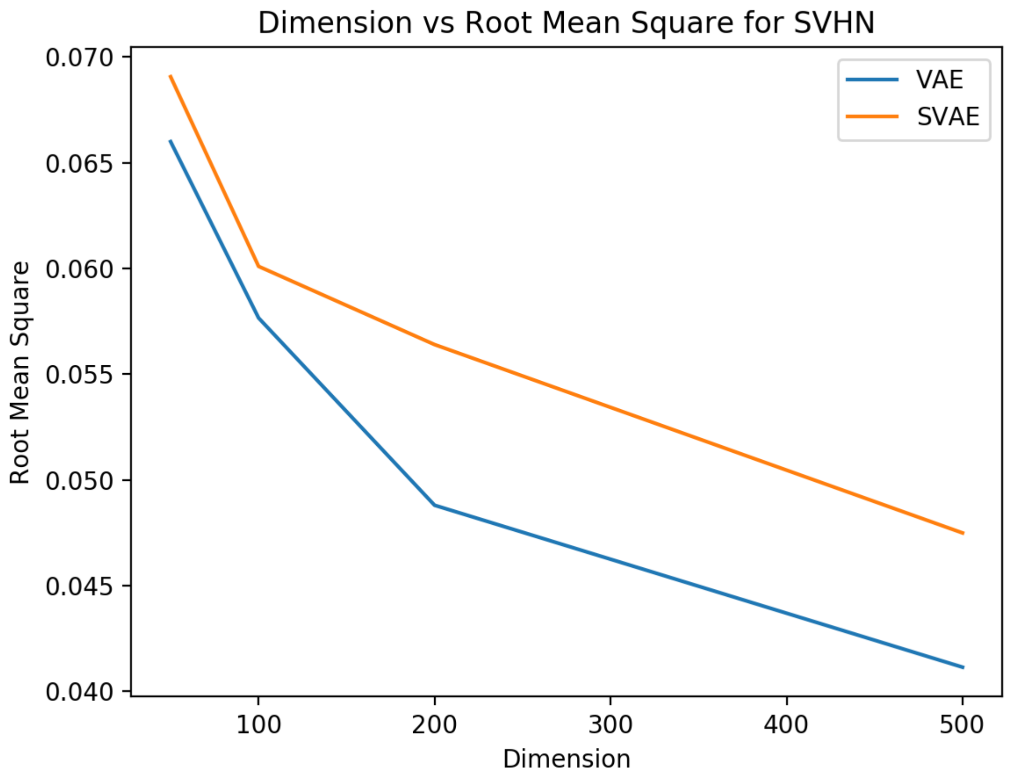


Figure 8: RMSE for VAE and SVAE with different dimensions of z on SVHN dataset

We can clearly see that as the dimension increases, RMSE decreases, which agrees with our belief that with larger z we lose less information when we encode so reconstructed image will be more similar to input image. Similar to the MNIST case, VAE outperforms SVAE in RMSE. This is because the assumption that y and z are independent are quite strong and not satisfied. Therefore, when we reconstruct the image and the decoder takes in hidden code z and label y, since the independence assumption is not satisfied, the generation quality is not as good. In contrast, VAE does not make such assumption, therefore the model is more "free" to encode z and decode back into image, giving it better reconstruction error.

4 Reflection and Outlook

For SVHN, our variational autoencoders combine with random forest only achieves 44.02% accuracy, and SVAE also achieves only 91.2% which is reasonable, but not very good. We want to improve the accuracy by tuning the parameters and figuring out the sensitive domain of the parameter that may lead to overfitting. Also, we can try more value of z and allow the number of epoch to be large. It took roughly 5 hours with epoch equals 30 and we need to run the model many times, so it's quite impossible for us to go beyond that in a short time.

For long term goal, we want to find a way to successfully apply our model and algorithms to CIFAR10 dataset. We tried, but did not succeed due to the characteristic of CIFAR10 images. Given more time, we believed that we can find a way to tune the model so that it can produce images that are close enough to original CIFAR10 dataset. Also, we want to apply the normalizing flow and autogressive flow to the prior instead of assuming that the prior is Gaussian distribution. We believe that by doing so, we will obtain better result.

The most significant take-home lesson from BDL method is that even though Bayesian deep learning is grounded in probability theory, there are still lots of things about it that requires practical knowledge instead of theoretical understanding. Even if you understand the mathematical model perfectly, you still have to tune parameters, choose some exact model architecture without understanding why to boost performance, fix numerical issues,... Even though Bayesian theory is established, BDL is still a new field and many people who works in this lacks comprehensive understanding of everything that it does.

References Cited

[1] Diederik P. Kingma, Max Welling, *Auto-Encoding Variational Bayes*, arXiv preprint arXiv (2013).

[2] Diederik P. Kingma, Max Welling, *Semi-Supervised Learning with Deep Generative Models*, arXiv preprint arXiv (2014).

[3] Jonathan Long, Evan Shelhamer, *Fully Convolutional Networks for Semantic Segmentation*, arXiv preprint arXiv (2014).

[4] Tensorflow, *https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/training/adam.py* , github (2015).

[5] Scikit-learn, *https://github.com/scikit-learn/scikit-learn* , github (2012).

[6] Daniel Gutteriez, *RMSprop Optimization Algorithm for Gradient Descent with Neural Networks* , insidebigdata (2017).

[7] Keras, *https://keras.io* , Keras (2015).

[8] Yann Lecun, *http://yann.lecun.com/exdb/mnist/* , MNIST (1999).

[9] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng, *Reading Digits in Natural Images with Unsupervised Feature Learning* , SVHN (2011).