# Disentangling in $\beta$-VAE

**Rishabh Mishra**
1213191094

**Goutham H M**
1213159465

{rmishr19, gmanjuna}@asu.edu
Arizona State University

**Abstract**

Unsupervised techniques to interpret and represent data from a data domain has become at most important in the field of deep learning and AI in general. In this paper we try to explain and implement few state-of-the art techniques data generative techniques and analyze them. $\beta$-VAE by Hignes et Al(1) is a successor of VAE(Variational Autoencoders) (Bengio et al 2013)(2), unsupervised InfoGAN(Hilton et al)(3), semi-supervised DC-IGN(Kulkarni et al 2015). We briefly explain autoencoders, variational autoencoders as a primer to better understand $\beta$-VAE(4). We will try to understand how a hyperparameter $\beta$ will help in generating disentangled representations.

## 1 Motivation

Unsupervised techniques to interpret and represent data from a data domain has become at most important in the field of deep learning and AI in general. In this paper we try to explain and implement few state-of-the art techniques data generative techniques and analyze them. $\beta$-VAE by Hignes et. Al is a successor of VAE(Variational Autoencoders) (Bengio et al 2013), unsupervised InfoGAN(Hilton et al), semi-supervised DC-IGN(Kulkarni et al 2015). We briefly explain autoencoders, variational autoencoders as a primer to better understand $\beta$-VAE. We will try to understand how a hyperparameter $\beta$ will help in generating disentangled representations.

## 2 Introduction

Discovering disentangled representation from the data is of significant value for a large variety of tasks and domain. For example to find the hidden independent and interpretable features in the data. Autoencoders plays an important role in finding the hidden representation of the data. Previous attempt to find the disentangled representation from the data required the prior knowledge of number and nature of data generative factors using some kind of semi-supervised learning approach. This is not feasible in the real world. B VAE is the modern deep unsupervised approach to disentangled feature learning. It automatically discovers the independent latent factors in the data. B VAE uses the VAE architecture with an additional hyperparameter B whose value can be estimated using a disentanglement matrix or can be determined heuristically. This paper describes in possible approach and outcome of B VAE architecture in finding the disentangled features of MNIST and SVHN dataset. In this paper, we describe how various values of B affects our results. We have tried with b value of 8 and 12 on MNIST dataset and 10 and 12 on SVHN dataset.

## 3 Variational Autoencoders

Autoencoder is a type a neural network which can be used to find hidden patterns in lower dimensions. The goal here is to represent data in lower dimension and to reconstruct back to same dimension as input. Although Auto-encoders work seemingly well with very minimal reconstruction error. The latent code generated has least significance with respect to input data. Therefore autoencoders do not learn the probability distribution of the the input data. Autoencoders might not work well if we try to create more samples out of input samples. Beingo Et al in 2015 introduces variational autoencoder which is very similar to autoencoders except that they try to fit to a probability distribution. The bottleneck vector z is now replaced by two vectors and representing mean and standard deviation of the distribution as shown in Fig1. The loss function contains two terms reconstruction error and KL divergence distribution between the distribution you are trying to fit and the distribution considered.

## 4 $\beta$-VAE

$\beta$ VAE is a special type of VAE introduced by[2] which by adding a single hyperparameter $\beta$ (¿1) can learn various dis-entanglements from rained latent variables. The idea is to introduce a lagrangian multiplier $\beta$ on latent loss during training by adding more penalty on latent loss. If by using appropriate $\beta$ factor we can visualize what a latent variable has learnt. For example thickness, angle, rotation of digits in MNIST and SVHN.

## 5 Implementation and Architecture

The implementation uses an encoder and a decoder architecture or a B-variational autoencoder. It takes input image as a MNIST(m*28*28*1) image or a SVHN(m*32*32*3) image (where m is number of examples).

### 5.1 Encoder Architecture

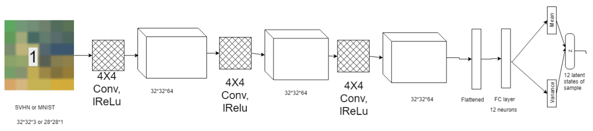

Figure 1: Encoder Architecture

- A Convolution Layer with filter size 4*4 with 64 channels, stride = 2, padding = same and leaky relu activation.
- Another Convolution Layer with filter size 4*4 with 64 channels, stride = 2, padding = same and leaky relu activation.
- Another Convolution Layer with filter size 4*4 with 64 channels, stride = 1, padding = same and leaky relu activation.
- Output of above is flattened and connected to dense fc layer with number of output neurons as 12
- The above layer is used to calculate the mean and standard deviation of the distribution.
- 12 hidden disentangled features are extracted and returned.

### 5.2 Decoder Architecture

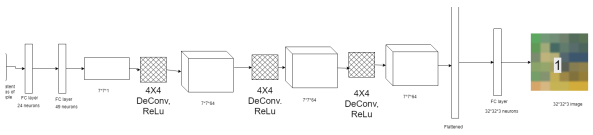

Figure 2: Decoder Architecture

- Sample from the hidden layer distribution is flattened and connected to dense layer with double the output neuron. For example: for 12 latent variables it is fully connected to 24 neurons of the dense layer and is then passed through leaky relu activation function.
- The above layer is fully connected to another dense layer of size 49 and the to leaky relu activation function.
- Output of above is reshaped to shape of (7*7*1). Then.
- A Convolution Layer with filter size 4*4 with 64 channels, stride = 2, padding = same and relu activation is used
- A Convolution Layer with filter size 4*4 with 64 channels, stride = 1, padding = same and relu activation.
- A Convolution Layer with filter size 4*4 with 64 channels, stride = 1, padding = same and relu activation.
- Output of above is flattened and then fully connected to dense layer of size 32*32*3 with sigmoid activation.
- The above output is then reshaped in to size of 32*32*3 and is the output of the decoder.

## 6 Experiments

### 6.1 Training

We have generated models for each individual digits of MNIST/SVHN to see the behaviour of network on individual digits. We use value of $\beta$ as 4. We believe that if similar digits of data is given to the network then network can easily learn the latent representation. So we divided our experiment in two parts.

- Training the network for individual digits: In this phase we extracted 2000 samples of each digits and trained the network on each digits one at a time and extracted the hidden representations. We took 8 and 12 hidden representations respectively. Our idea was to increment the number of hidden representations incrementally as you get more disentangled feature which are interpretable when latent neurons are more than the actual disentanglement present in the data. As we do not know the number of actual disentangled features present in the data, our only option is to increase the number incrementally.

- Training the network for all the digits: Number of samples taken were 10000 ( 1000 from each digits) from MNIST/SVHN dataset and the model was trained on that.

Hence the model was trained for:
(i). Individual digits in MNIST dataset: 2000 samples from each digits are taken and model was trained separately for each of then and analyzed
(ii). All Digits in MNIST dataset: 10000 samples were taken from all the digits and model was trained.
(iii). Individual digits in SVHN dataset: Model was separately trained on the each of the 2000 samples taken from each of the digits in SVHN dataset.
(iv). All digits in SVHN dataset: 10000 samples were taken from SVHN dataset and model was trained on that data.

### 6.2 Testing

During testing, a test image from the test set is taken and an array of latent variables of the image is computed using the encoder of the trained model. Single variable in the latent representation is varied at a time keeping other variables constant. The value of single variable is varied in a heuristically determined range of values to get the specific output image. It was found that different variables changes the generated output image within a different range. This observation shows that different disentangled features vary in a given range in the original data.

## 7 Observations and Results

The intuition and the expected result in each of the experiment above was to vary a single variable in the latent representation at a time and expect the change in one of the values just say for example: thickness, glow, rotation in the output images. Following observation and inferences was found:

### 7.1 MNIST

- On individual digits:
  - Observation: Digits in MNIST were found to vary in two dimension out of 12 latent representations, namely rotation (See Appendix Figure 3), height (See Appendix Figure 5), thickness.
  - Inference: This may be because the data trained was less

- On whole samples:
  - Observation: Training 10000 images for 200 epochs found to be very less intuitive to understand.
  - Inference: Probably training more samples to train and running for more epochs can give better results. Maybe we need to consider adding more convolution layers to learn more hidden features.

### 7.2 SVHN

- On individual digits:
  - Observation: Digits in the SVHN were found to vary in almost all the dimension, this includes changes in lighting conditions(See Appendix Figure 30), thickness of the character(See Appendix Figure 37), contrast(See Appendix Figure 38), shadow of the digit etc.
  - Inference: Model was able to extract the latent representations successfully when trained and tested on the individual digits type in the SVHN dataset.

- On whole samples:
  - Observation: We find out training 1000 SVHN digits for 200 epochs show few disentanglements. This result if compared to MNIST when considering multiple digits proved to be work better.
  - Inference: Since we have another dimension in SVHN than compared to

MNIST we could find more disentanglements. We feel it worked better because of three layers as compared to only one layer in MNIST. Major disentanglement we found was related to lighting and darkness of the digit and its surroundings. Considering CNN for MNIST helped us to incorporate same architecture for SVHN as well.

## 7.3 Reason to choose CNN over dense layer

CNN Preserves spatial features, nearby pixels have similar values. Not storing spatial information will prove to be costly during reconstruction. Convolution layers before a fully connected layer proved to work better has it learned much more features and hidden representations.

## 7.4 Why we trained single digits

To check disentanglement in a simple 4 layered Encoder decoder network. We found that training all digits at once on a 4 layered network didn't seem to work fine because there are way too many features which need to be learnt. Hence to find disentanglement we trained 1000 images of single digit. We found better results and visible disentanglement by doing this.

## 7.5 Number of latent variables(Z)

From our experiments we learnt that choosing number of latent spaces to be for plays an important role. If we choose way to many latent features like 8 in MNIST or SVHN we found that reconstructions were too vague to find disentanglements since it is assumed that the network has combined many disentanglement to a latent variable. When we trained the network to learn 12 latent variables. We found better entanglements as shown in the observations.

## 7.6 Result

The images related to the results of training and testing each type of examples are shown in Appendix A. Z0 to Z 11 are the latent variables varying which we are expected to see changes in the image related to the feature information these variables extracts. The different images corresponding to Z0, Z1, Z2 etc shown in the appendix are formed by varying one of the latent variable either Z0 or Z1 or Z2 etc. For each variable 12 plot are shown which are formed by varying the variable value slightly within a heurustically setermined range. This range is determined heuristically because we assume that the different features present in the image have their own set of measurement criteria which is different from one another. Using this heuristic approach to varying features, from Appendix A, we can find that:

- Model training for a single MNIST digit leads to better reconstructions of that digit. We can also find that varying different latent variables either cause the image to rotate(Appendix Figure 3) or they change the size of the image(Appendix Figure 8).
- Model training on all the samples to find the latent variables couldn't find much information about the internal structures of the digits. Only information we could get from Appendix Figure 21 that the digit fades away initially as the latent parameter Z6 is varied.
- Model gave good result on SVHN dataset when trained on single digits individually. As we can see the reconstructions were better and also that varying a particular latent variable brings a different change for the same image as shown in the images from Figure 29 to Figure 39. For example: In Appendix A Figure 36 we can see that on varying Z8, th brightness of the image increases. So, we can infer that Z8 variable controls the brightness.
- Important thing to notice here is that, usually SVHN digits are not present as single digit like in MNIST, they occur somethimes with other digits as well. For exmaple, if house number is 19 and we want to train on all the digits which have 1, then we will hardly find an image with the single digit one, 1 will appear with 9 in a singe image as in this case or some other digit as well. But still the model was able to extract the latent features when trained on the same kind of digits[Images with reconstructed 1's are in the zip file]. Thus we can infer that we can use the model to extract various features from same kind of digits as well.
- As evident from the Appendix Figure 51, training the model on a subset of SVHN data e extracted a latent feature which affects the 'brightness' of the image. Other images were a bit vague.

## 8 Conclusion

The experiment revealed that the number small number of neurons in the latent representations do not give disentangled representations. To get the disentangled representations we need to have more latent variables than the features present in the original dataset. This can be determined heuristically by incrementing the nodes in the latent representation one at a time. Another direct conclusion observed from the above experiment was $\beta$ variational autoencoders learns the effectively when it is given only one kind of digit at a time. As we can see from the experiments, the inputs images often contained characters ( other than which was given) together with the given character. In spite of the presence of other characters B-VAE disentangles the features in an effective manner. The above architecture was effective in extracting the features when SVHN dataset with single kind of digits at a time was given when compared to the MNIST dataset.

## References

[1] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, Alexander Lerchner. *$\beta$-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework, 2017.*

[2] Y. Bengio, A. Courville, and P. Vincent. *Representation learning: A review and new perspectives.*

[3] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, Pieter Abbeel *InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets.*

[4] Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, Alexander Lerchner *Understanding disentangling in $\beta$-VAE*

## 9 Contribution

- Goutham H M - Literature Review, Coding, MNIST, Report(50%)

- Rishabh Mishra - Literature Review, Coding, SVHN, Report(50%)

## 10 Appendix A: Figures

### 10.1 MNIST Single Digit Training: Varying Different Latents [Fig. 3 - 14]

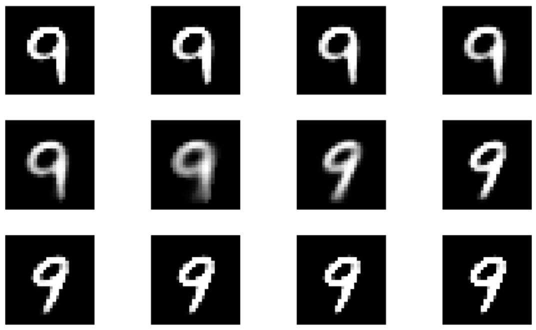

Figure 3: MNIST Single Digit Latent Z0

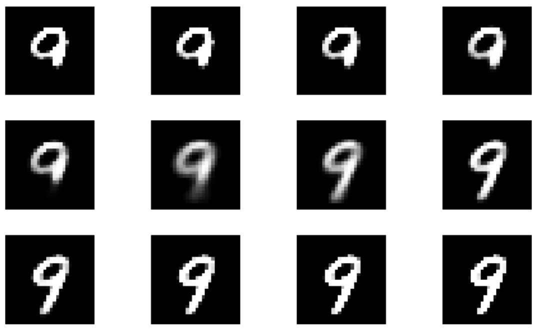

Figure 4: MNIST Single Digit Latent Z1

### 10.2 MNIST All Digit Training: Varying Different Latents [Fig. 15 - 26]

### 10.3 SVHN Single Digit Training: Varying Different Latents [Fig. 27 - 39]

### 10.4 SVHN All Digit Training: Varying Different Latents [Fig. 40 - 51]

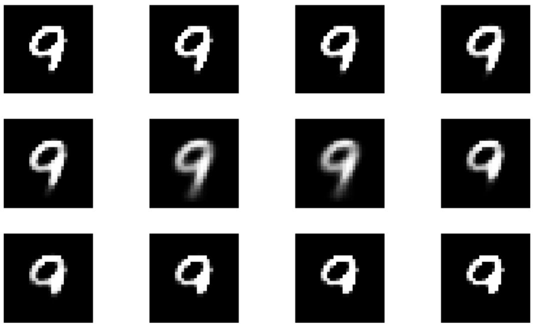
Figure 5: MNIST Single Digit Latent Z2

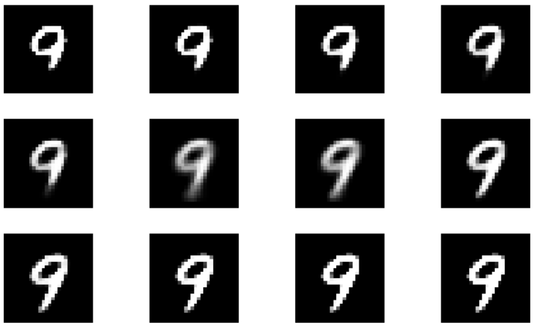
Figure 8: MNIST Single Digit Latent Z5

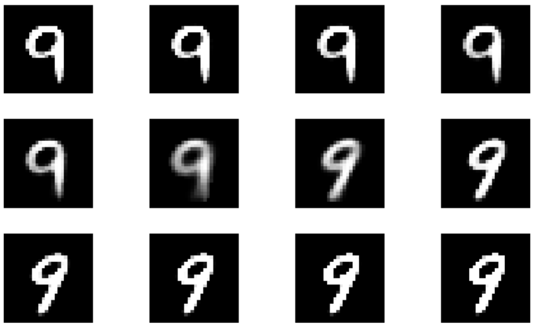
Figure 6: MNIST Single Digit Latent Z3

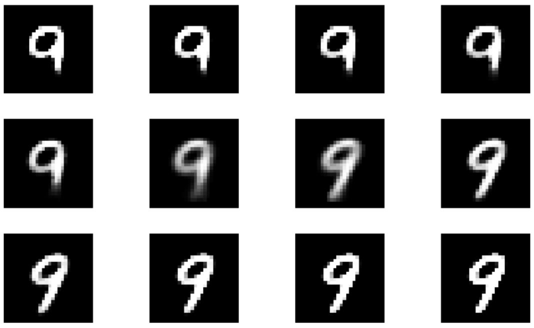
Figure 9: MNIST Single Digit Latent Z6

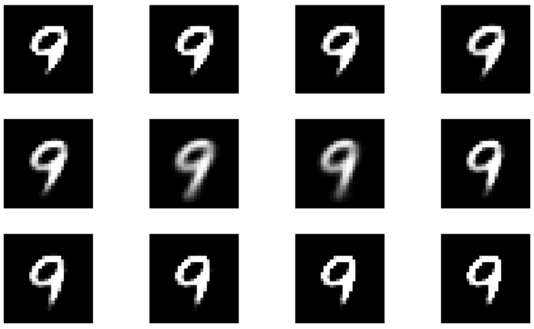
Figure 7: MNIST Single Digit Latent Z4

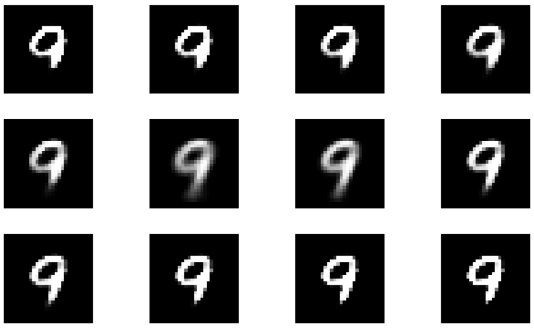
Figure 10: MNIST Single Digit Latent Z7

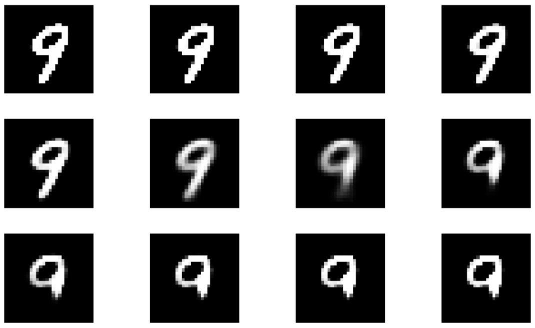
Figure 11: MNIST Single Digit Latent Z8

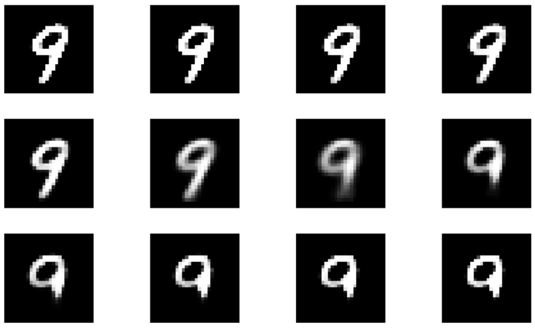
Figure 14: MNIST Single Digit Latent Z11

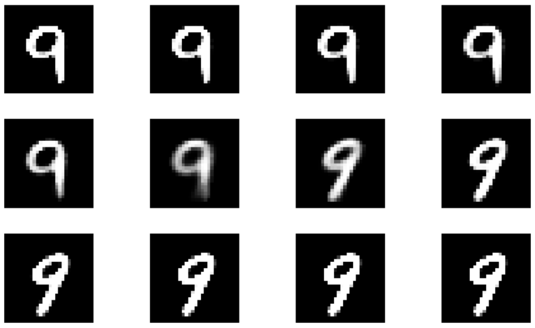
Figure 12: MNIST Single Digit Latent Z9

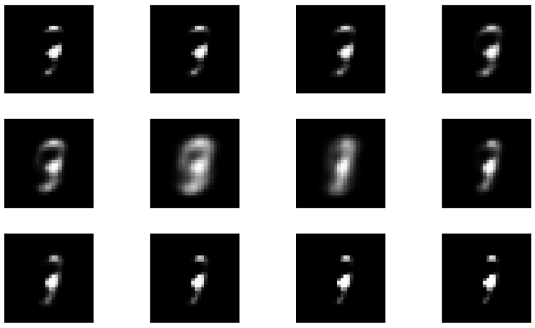
Figure 15: MNIST All Digit Latent Z0

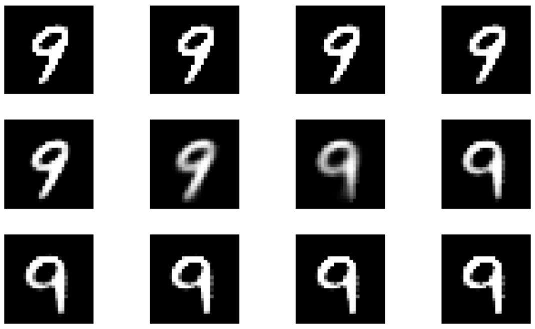
Figure 13: MNIST Single Digit Latent Z10

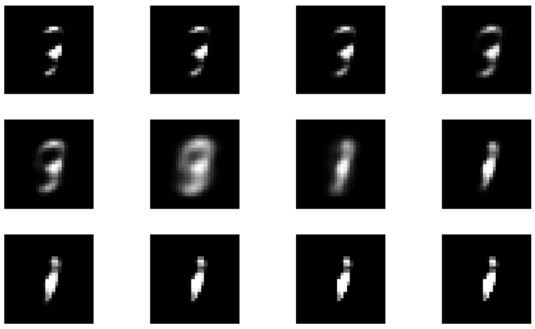
Figure 16: MNIST All Digit Latent Z1

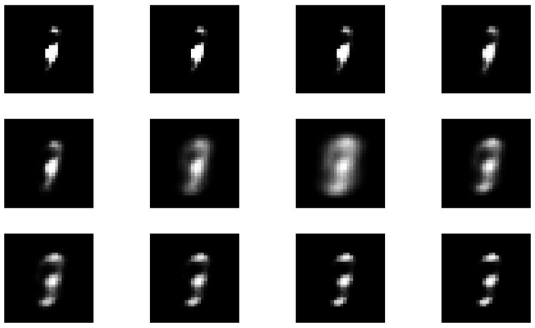
Figure 17: MNIST All Digit Latent Z2

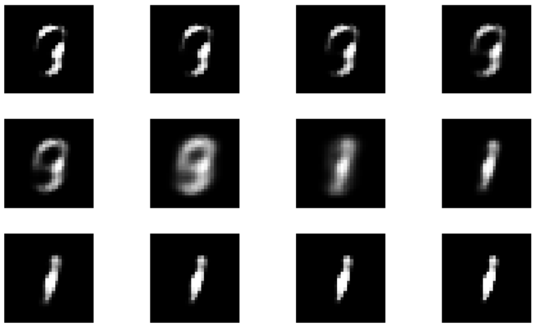
Figure 20: MNIST All Digit Latent Z5

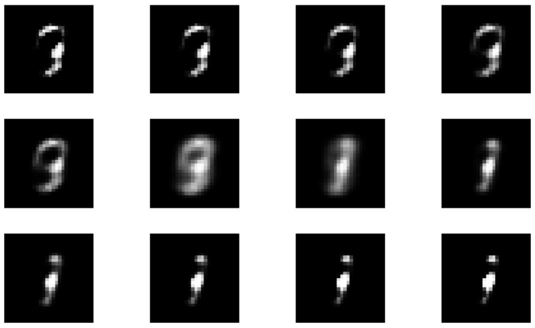
Figure 18: MNIST All Digit Latent Z3

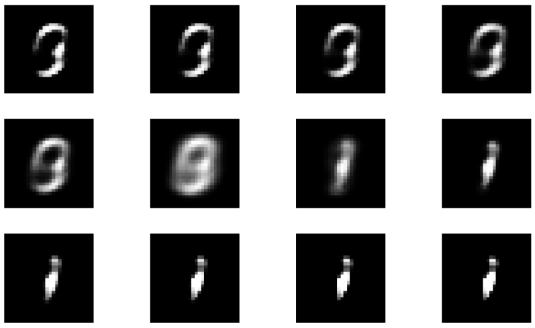
Figure 21: MNIST All Digit Latent Z6

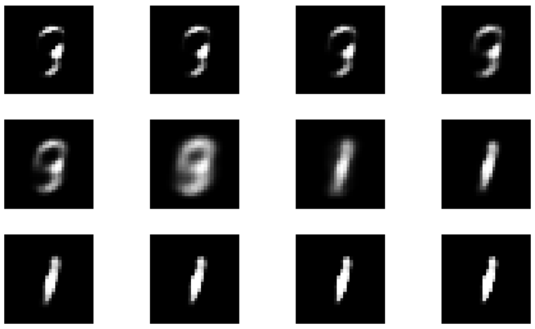
Figure 19: MNIST All Digit Latent Z4

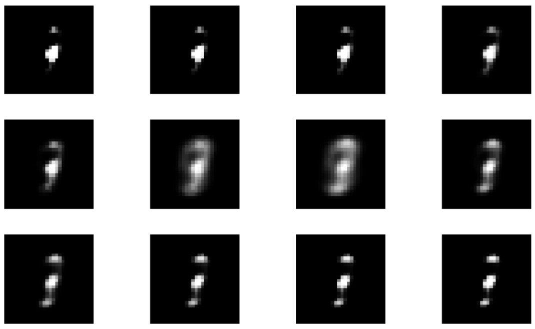
Figure 22: MNIST All Digit Latent Z7

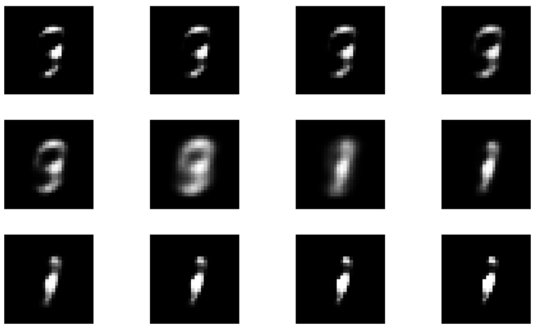
Figure 23: MNIST All Digit Latent Z8

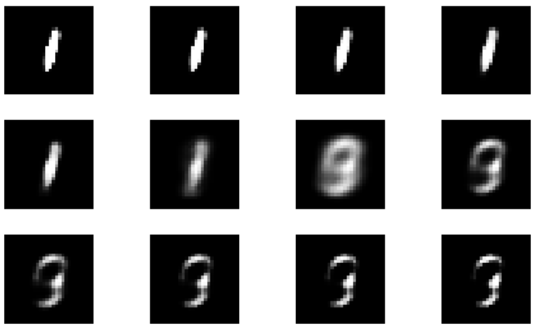
Figure 26: MNIST All Digit Latent Z11

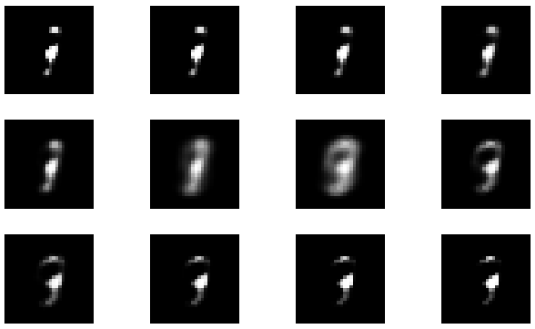
Figure 24: MNIST All Digit Latent Z8

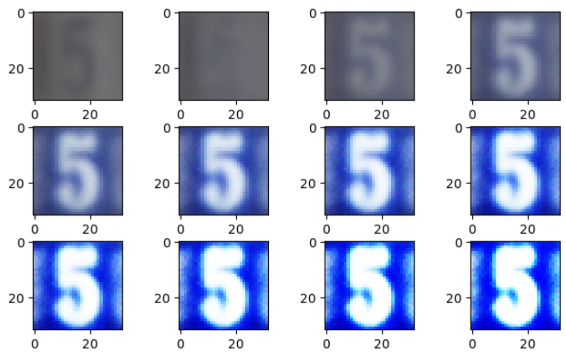
Figure 27: SVHN Single Digit Latent Z0

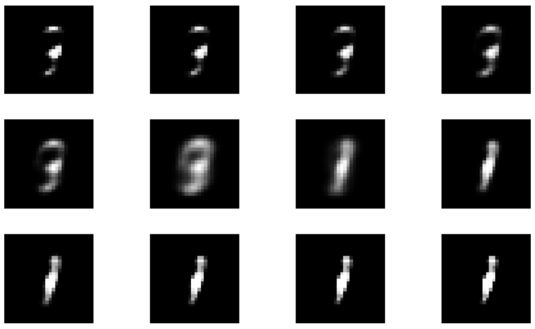
Figure 25: MNIST All Digit Latent Z10

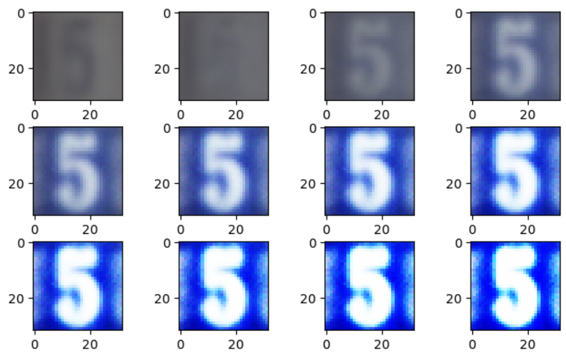
Figure 28: SVHN Single Digit Latent Z0

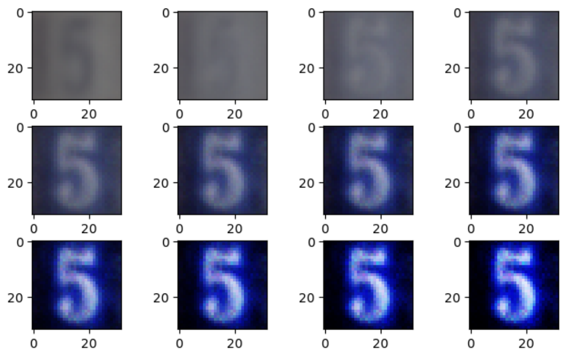

Figure 29: SVHN Single Digit Latent Z1

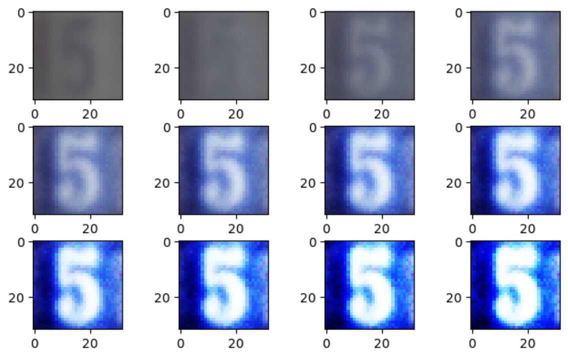

Figure 32: SVHN Single Digit Latent Z4

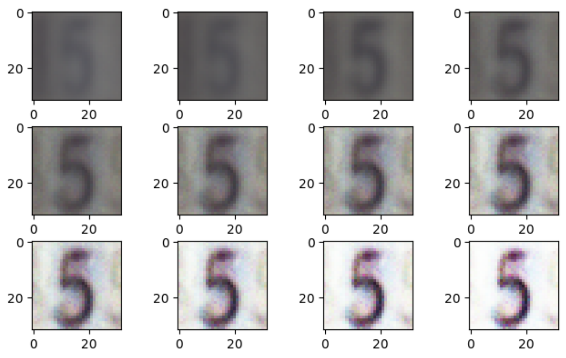

Figure 30: SVHN Single Digit Latent Z2

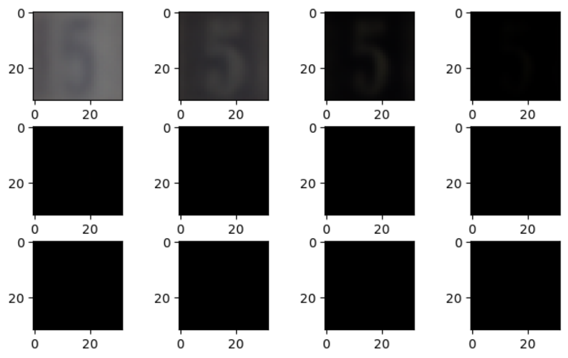

Figure 33: SVHN Single Digit Latent Z5

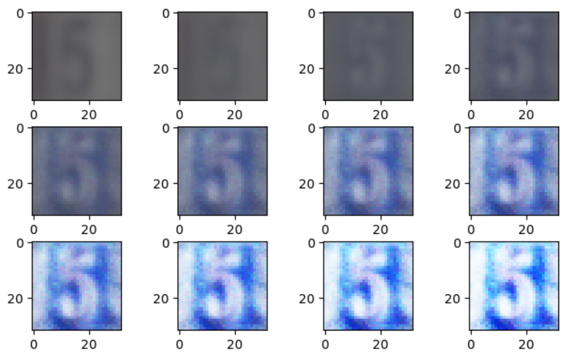

Figure 31: SVHN Single Digit Latent Z3

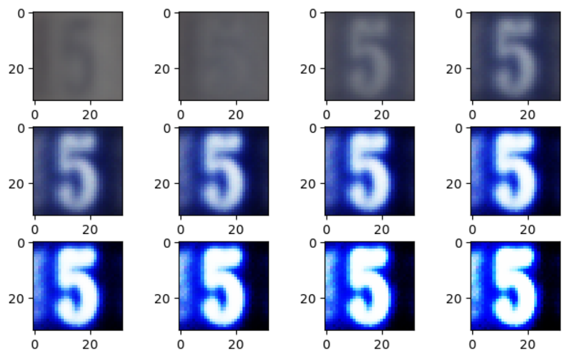

Figure 34: SVHN Single Digit Latent Z6

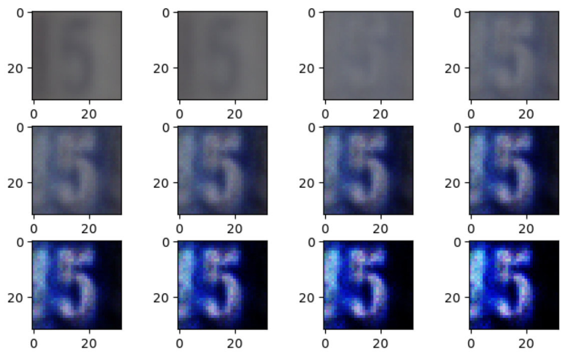
Figure 35: SVHN Single Digit Latent Z7

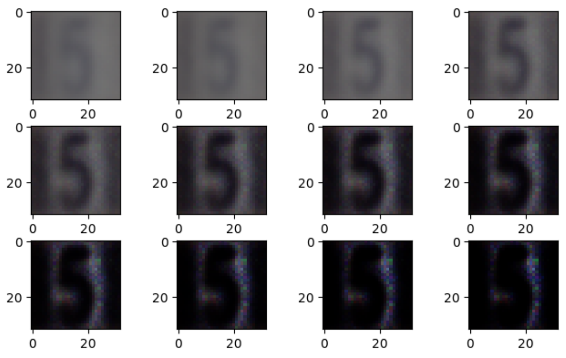
Figure 38: SVHN Single Digit Latent Z10

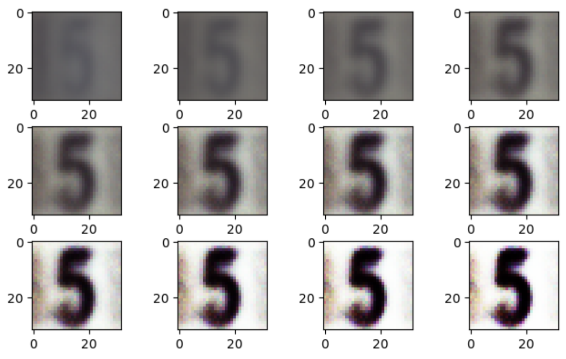
Figure 36: SVHN Single Digit Latent Z8

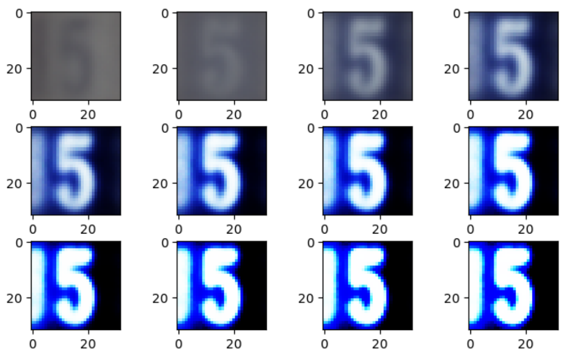
Figure 39: SVHN Single Digit Latent Z11

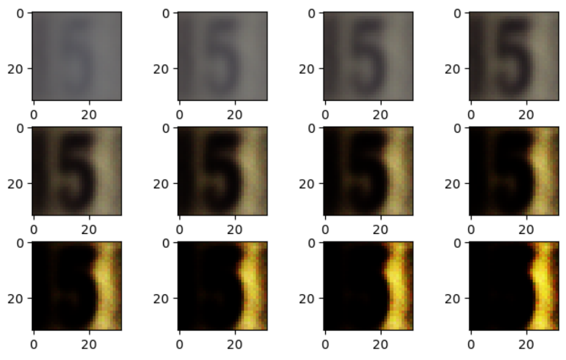
Figure 37: SVHN Single Digit Latent Z9

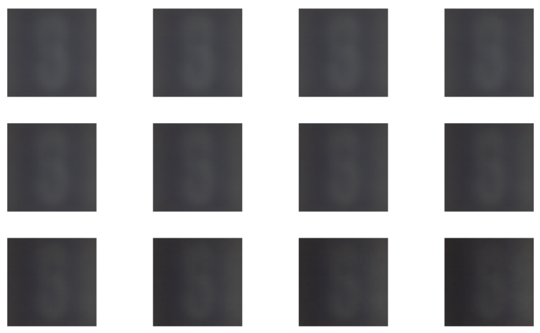
Figure 40: SVHN All DigitLatent Z0

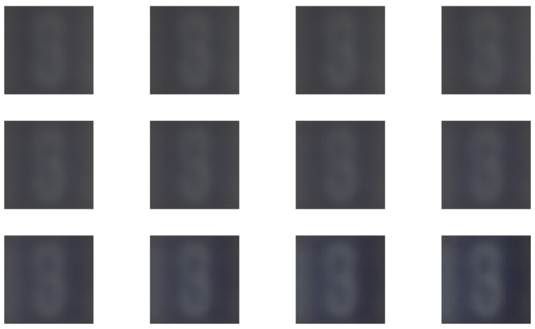

Figure 41: SVHN All DigitLatent Z1

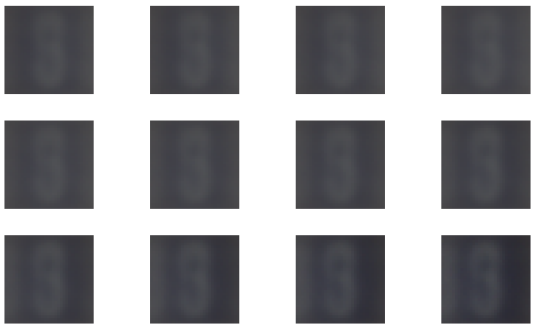

Figure 42: SVHN All DigitLatent Z2

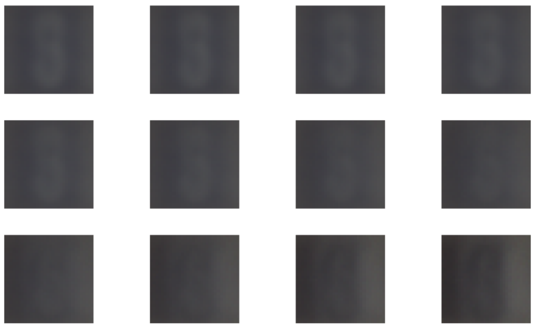

Figure 43: SVHN All DigitLatent Z3

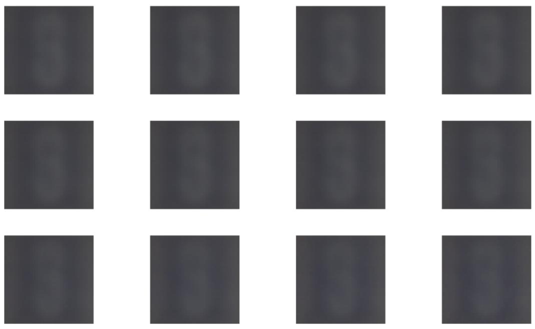

Figure 44: SVHN All DigitLatent Z4

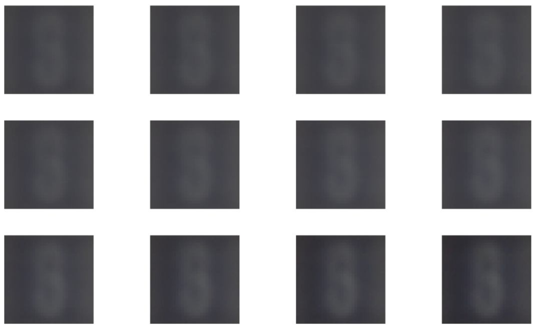

Figure 45: SVHN All DigitLatent Z5

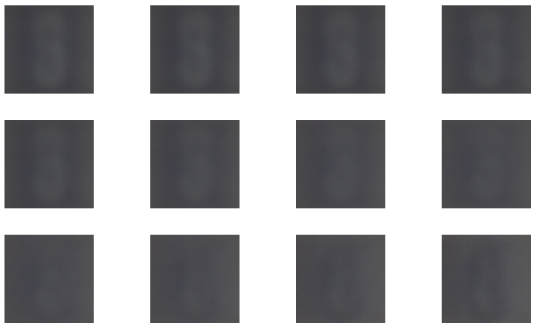

Figure 46: SVHN All DigitLatent Z6

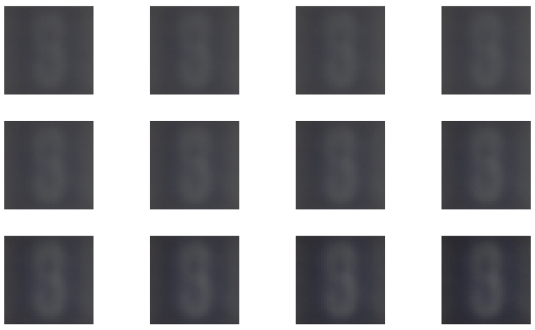
Figure 47: SVHN All DigitLatent Z7

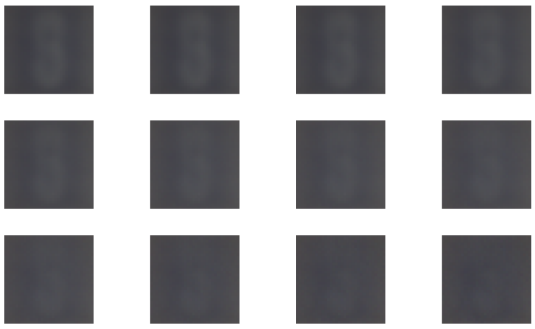
Figure 50: SVHN All DigitLatent Z10

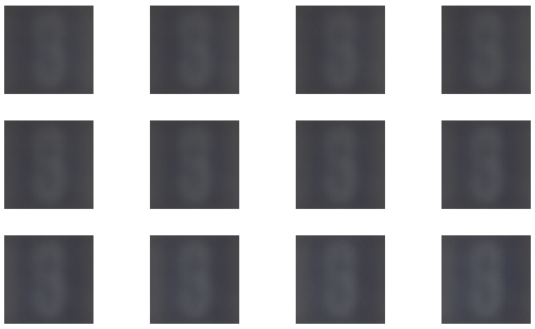
Figure 48: SVHN All DigitLatent Z8

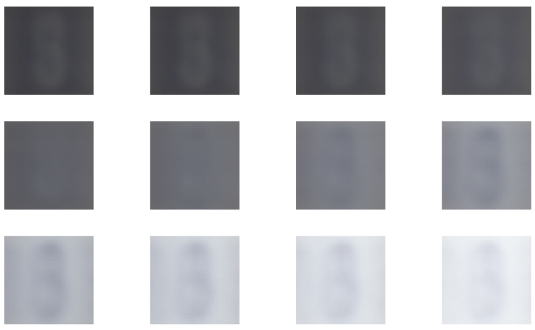
Figure 51: SVHN All DigitLatent Z11

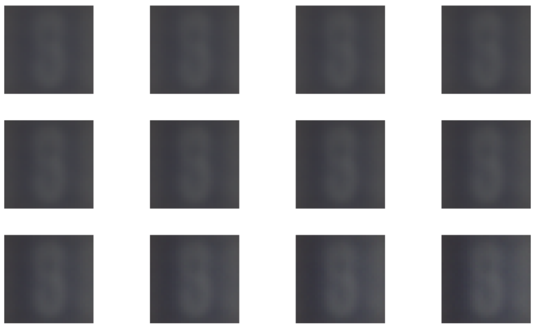
Figure 49: SVHN All DigitLatent Z8