Learning Disentangled Latent Spaces using Variational Autoencoder

Manjulata Chivukula
School of Informatics, Computing And Engineering
Indiana University, Bloomington, Indiana 47408
mchivuku@iu.edu

Abstract

Variational Autoencoders (VAEs) [1] are widely used deep generative models that are capable of learning unsupervised latent representations in data. In this paper, I explored the use of vanilla VAE and other VAE models in order to understand and analyze the learned latent representations and their performance in downstream supervised image classification tasks. I demonstrated the results of my experiments using MNIST dataset [2]. The results of my experiments indicate that the vanilla VAE performed better in the image classification tasks over β-VAE (or disentangled VAE) and VAE+GAN model. I also found the reconstruction results of VAE+GAN and adversarial symmetric VAE were sharper than the vanilla VAE and disentangled VAE. In addition, the classification performance of the models indicate that disentangling the latent representation in data did not help with content classification task. The results of TSNE plot indicated that the β-VAE model (or disentangled VAE) disentangled the hidden representations in the data which may be manifested by the dense and overlapped cluster plot compared to Vanilla VAE model where the clusters were found to be far apart. Also, classification performance provided a way to analyze the quality of the encoder. For future work, I would like to rerun the same experiments using a dataset that contains more disentangled representations and analyze the results.

1. Introduction

Deep generative models have shown great success in their ability to successfully model, complex high dimensional data like in images, and natural language etc. Although they are useful for data-generation and feature extraction, it is often still difficult to understand what these networks are learning in their latent representation. For example: the correspondence between particular latent dimension of the latent representation and the aspects of the data it is related is still unclear.

Consider a dataset with N labelled examples:

$$D = \{(x_1, y_1), (x_2, y_2)...(x_N, y_N)\}$$

each with label

$$y_i \in 1....K,$$

data belonging to y_i has some latent structure (can naturally cluster into subclass specified by the label), the goal of the project is to understand the role of VAE model in learning this hidden latent structure of the example, analyze the quality of the generated images from the model and its performance in the classification tasks.

I investigated this problem within the framework of variational autoencoders [1] and generative adversarial networks (GAN) [3]. VAE forms a generative distribution over the data $p_\theta(x) = \int p(z).p_\theta(x\|z)dz$ by introducing latent variable $z \in Z$ and its associated prior p(z). Generative adversarial network (GAN) forms a generative model that have shown tremendous capability of generating realistic data using the models that can mimic the real data. However, both VAE and GANs have drawbacks, reconstruction images of VAE are blurry and GANs are difficult to train while providing sharp reconstructed images. In this project, I explored VAE and β-VAE or disentangled VAE architectures in learning latent representation and, explored the use of adversarial learning interface as in GAN for improving the reconstruction results [4]. In addition, I also explored reformulation of ELBO the idea that was proposed in [5] using symmertic KL-divergence between data and latent codes.

2. Background and Foundations

Consider an observed data sample x, modeled as being drawn from $p_\theta(x|z)$, with the model parameters θ and latent code z. The prior distribution on latent code z is denoted as p(z), typically considered as isotropic Gaussian distribution. The posterior on the latent code can be given as p(z|x), and since this is typically intractable due to the high dimensionality of the data, this posterior distribution is usually approximated using variational distribution $q_\phi(z|x)$ parameterized by parameters ϕ. The conditional distributions

$p_\theta(x|z)$ and $q_\phi(z|x)$ are usually modeled using two neural networks encoder and decoder. Since z is a latent code for x, $q_\phi(z|x)$ is termed as stochastic encoder, with $p_\theta(x|z)$ as corresponding stochastic decoder. The observed data is assumed be drawn from the distribution q(x), that does not have an explicit form but it is learned from the input samples $\{x_i\}_{i=1}^n$ used for learning.

$$z \approx Enc(x) = q(z|x)$$
$$x^{\sim} \approx Dec(z) = p(x|z)$$

The figure 1 shows the architecture of the VAE with the two encoder and decoder networks. VAE regularizes encoder by using the prior on z: p(z) $\approx \mathcal{N}(0,1)$

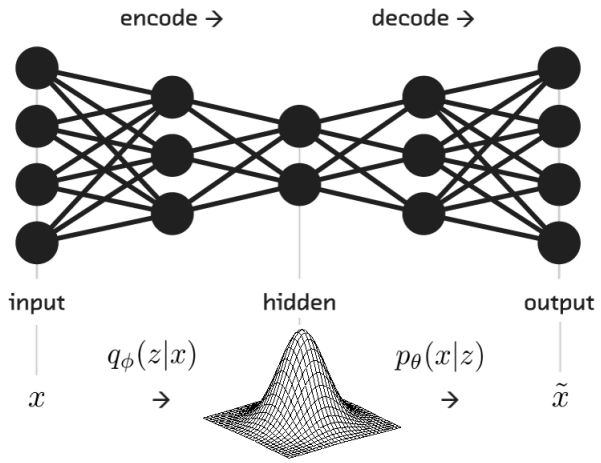

Figure 1. Setup of Variational Autoencoder

2.1. Traditional Variational Autoencoders

VAE [1] seeks to bound $p_\theta(x|z)$ by maximizing the variational lower bound (or evidence lower bound (ELBO)) $L_{VAE}(\theta, \phi)$ that is given by:

$$L_{VAE}(\theta,\phi) = E_{q_\phi(x,z)} log[\frac{p_\theta(x,z)}{q_\phi(z|x)}]$$
$$= E_{q_\phi(x,z)}[log \frac{p(x|z)p(z)}{q(z|x)}]$$
$$= E_{q_\phi(x,z)}[log p(x|z)] + D_{KL}(q(z|x)\|p(z))$$
$$=$$ Reconstruction error + Kullback-Leibler divergence

with expectations $E_{q_\phi(x,z)}$ and $E_{q(x)}$ performed via sampling. VAE setup is an approximation to ML learning of parameters θ. Hence, learning parameters θ is equivalent to minimizing KL divergence between distributions q(z|x) and p(z).

2.1.1 Limitations

VAE is found to produce blurry reconstruction as generation of q(x) depends on distribution of latent code q(z|x) and, if the quality of latent code is very different from p(z|x), then the distribution q(z|x) fails to produce samples that belong to distribution p(x).

2.1.2 Advantages

The advantages of VAE is that they are able to produce inference on the latent distribution unlike other class of generative models like GANs.

2.2. β-VAE

β-VAE [6] is a deep unsupervised generative model used for disentangling latent spaces as independent factors of variation in unsupervised data. The goal of the model is to overcome the disadvantages of traditional VAE in learning disentangled factors by introducing the hyper parameter β that modulates the learning constraints applied to the model causing the model to learn statistically independent latent parameters. The objective of β-VAE is given by:

$$max_{\theta,\phi} E_{x\sim D}[E_{q_\phi(z|x)}[log p_\theta(x|z)]]$$
$$\text{s.t } D_{KL}(q_\phi(z|x)\|p(z)) < \epsilon$$
Rewriting as Lagrangian under KKT conditions
$$L_{Beta}(\theta,\phi,\beta,x,z) =$$
$$E_{q_\phi(z|x)}[log p_\theta(x|z) - \beta(D_{KL}(q_\phi(z|x)\|p(z)))]$$

encourages the encoder to be such that nearby points in the input pixel space (as measured by the L2 norm due to Gaussian assumptions) would be close together in the latent space as well. Minimizing the loss is equivalent to maximizing the Lagrangian and thus works for initial optimization problem of optimizing ELBO.

When β=1, it is same as traditional VAE. When β >1, the model applies a stronger constraint on the latent bottleneck and limits the representation capacity of z. Therefore a higher β encourages more efficient latent encoding and, further encourages the disentanglement. Meanwhile, a higher β may create a trade-off between reconstruction quality and the extent of disentanglement.

2.3. Generative Adversarial Network (GANs)

A GAN [3] consists of two neural networks as shown in the figure 2:

the generator network Gen(z) maps latent z to dataspace while discriminator network assigns probability y = Dis(x) $\in$ [0,1] that x is an actual training sample and probability 1-y that x is generated by the model through x = Gen(z) with

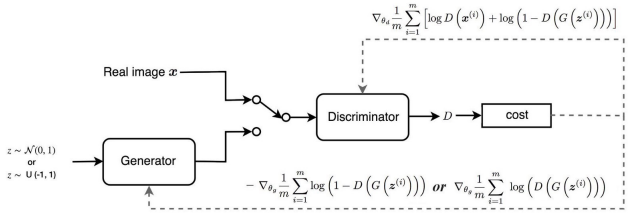

Figure 2. Generative Adversarial Network

z ≈ p(z). The objective of GAN can be given as:

$$min_G max_D V(D,G) = E_{x \approx p(x)}[logD(x)] + E_{z \approx p(z)}[log(1 - D(G(z)))]$$

and, the objective of the network is to find the binary classifier that gives best possible discrimination between the true data and generated data while simultaneously encouraging generator to fit the true data distribution. Thus, the aim is to maximize/minimize the binary cross entropy loss with respect to Discriminator/Generator respectively with x being the training sample and z between the latent distribution.

2.3.1 Limitations

Some limitations of GAN networks:

- Non-convergence - model parameters oscillate, destabilize and never converge. It is difficult to achieve Nash equilibrium shown below:

$$min_G max_D V(D,G) = xy$$

Nash equilibrium is x = y = 0

- Mode collapse - the generator collapses producing limited varieties of samples i.e. network produces sharp pictures, although they don't represent the entire data distribution

- Diminished gradient: the discriminator gets too successful that the generator gradient vanishes and learns nothing;

- Unbalance between generator and discriminator causing over fitting of the data;

- Highly sensitive to hyper parameter selections;

2.3.2 Advantages

The advantages of GAN is that they are able to produce sharp reconstruction results.

2.4. Autoencoding beyond pixels using a learned similarity metric (VAE + GAN)

VAE + GAN [4] is a network that is modeled using the VAE and GAN objective, shown below.

$$L_{VAE} = L_{like}^{pixel} + L_{prior}$$
$$L_{prior} = D_{KL}(q(z|x) \| p(z))$$
where
$$L_{like}^{pixel} = -E_q(z|x)[logp(x|z)] \text{ and}$$
$$L_{GAN} = log(Dis(x)) + log(1 - Gen(Dis(x)))$$

The architecture of the network is shown in the figure 3: Hence, the total loss of network is sum of VAE + GAN

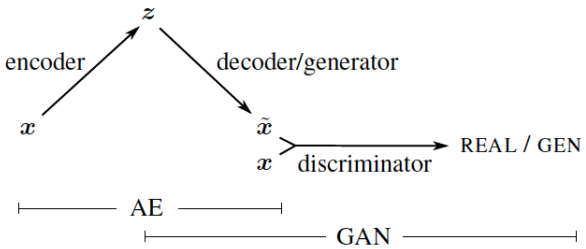

Figure 3. VAE+GAN

Loss:

$$L = L_{prior} + L_{llike}^{Dis_l} + L_{GAN}$$

where $L_{llike}^{Dis_l}$ is l-th layer features of discriminator

$$L_{llike}^{Dis_l} = -E_{q(z|x)}[logp(Dis_l(x)|z]$$

$L_{llike}^{Dis_l}$ in the computed in the following way and, can be described using the figure 4:

1. Feed the latent vector of z, into decoder to get, reconstruction result for x: $x^{\sim} \approx$ Dec(z)
2. Feed the result to the Discriminator to extract the l-th layer features for the reconstructed output: $Dis_l(x^{\sim})$
3. Feed the input of the VAE part x as an input to the discriminator, to extract the l-th layer features: Dis_l(x)
4. Form a Gaussian-based likelihood:

$$p(Dis_l(x)|z) = \mathcal{N}(Dis_l(x)|Dis_l(x^{\sim}), I)$$

and then compute: $E_{q(z|x)}[logp(Dis_l(x)|z]$

5. This formulation enables computing VAE reconstruction error that is measured in the feature space.

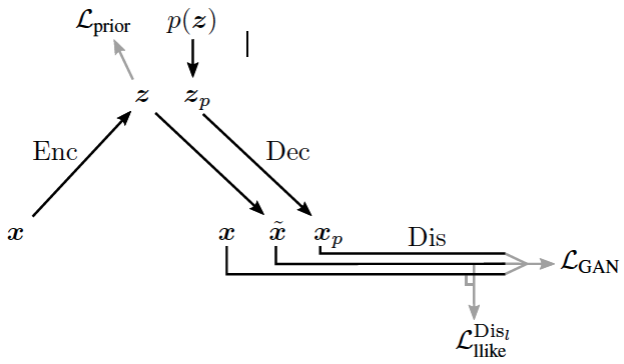

Figure 4. VAE+GAN process flow

2.4.1 Advantages

The advantage of the model is better reconstruction results and the network is able to better learn abstract embedding space.

2.5. Adversarial Symmetric Variational Autoencoder

Adversarial Symmetric Variational Autoencoder [7] is a new form of VAE network that is developed where joint distribution of data and latent codes is considered in the two symmetric forms

- observed data fed through encoder to yield latent codes
- from latent code drawn from a simple prior and propagated through decoder to manifest data.

and reformulation of the ELBO as shown below. The framework is desired to assign high probability to the points in the data distribution and low probability to the points not in the data distribution. This is different from maximum likelihood approach for VAE where the objective of the VAE is to assign high probability for points in the data distribution. Thus, the framework helps the network learn the lower bounds on marginal log-likehoods for data and the latent codes.

2.5.1 Approach - Reformulation of the lower bound

The key component of the approach described in the paper is the reformulation of Variational lowerbound or ELBO using symmetric Kullback-Leibler divergence of joint density functions from i) observed data to latent codes, ii) latent codes to decoder to reconstructed data. Optimizing the ELBo is equivalent to maximizing the marginal log-likelihood of the distribution which is further equivalent to minimizing symmetric Kullback-Leibler distance between the data and latent codes. The author further utilizes the adversarial training interface to learn the variational lower bound so that the better reconstruction of data can be achieved. The new formulation of the ELBO and the loss of VAE to maximize marginal likehood of data distribution $\log p_\theta(x)$, and latent code distribution $\log p_\theta(z)$ are formulated as shown below:

$$L_{VAE}(\theta,\phi) = E_{q_\phi(x,z)} log[\frac{p_\theta(x,z)}{q_\phi(z|x)}]$$
$$= E_{q(x)}[logp_\theta(x) - KL(q_\phi(z|x)\|p_\theta(z|x))]$$
$$= -KL(q_\phi(x,z)\|p_\theta(x,z)) + constant$$
$$L_{VAE_x}(\theta,\phi) = E_{q(x)} logp_\theta(x) - KL(q_\phi(x,z)\|p_\theta(x,z))$$
$$L_{VAE_z}(\theta,\phi) = E_{p(z)} logq_\phi(z) - KL(p_\theta(x,z)\|q_\phi(x,z))$$

The total variation loss of data and latent codes is then given by:

$$L_{VAE_{xz}} = L_{VAE_x}(\theta,\phi) + L_{VAE_z}(\theta,\phi)$$
$$= E_{q_{\phi(x,z)}}[logp_\theta(x)p(z) - logq_\phi(x,z) + logp_\theta(x|z)] +$$
$$E_{p_\theta(x,z)}[logq_\phi(z)q(x) - logp_\theta(x,z)]$$
$$= E_{q_{\phi(x,z)}}[logp_\theta(x|z) - f_{\psi_1^*}(x,z)]$$
$$+$$
$$E_{p_{\theta(x,z)}}[logq_\phi(z|x) - f_{\psi+2^*}(x,z)]$$

Each of the function $f_{\psi_1^*}(x,z)]$ and $f_{\psi_2^*}(x,z)]$ are learnt using adversarial training with the use of discriminators. With the use of symmetric KL divergence property and use of adversarial training the author claims to achieve state of art results in reconstruction of data.

3. Experiments and Results

3.1. Implementation Details

3.1.1 Unsupervised Learning

For training VAE, I have used MNIST [2] dataset, with batch size of 128 and trained for 100 epochs. I used relatively small batch size in order to get benefits of stochastic gradient descent. I used Adam optimizer with learning rate of 0.001. I did not experiment with different optimizers or weight initialization methods like Xavier init etc. I chose number of epochs solely based on the convergence rate for the loss.

For training VAE+GAN with components Encoder, Decoder and Discriminator, I have used MNIST [2] dataset with batch size of 128 andt trained for 250 epochs. I have used RMSProp optimizer with learning rate 0.0001.

3.1.2 Supervised Learning

For training Linear classifier, I chose single layer linear classifier with log softmax of the output. For the optimizer I choose Adam optimizer with learning rate of 0.001. For supervised learning, I first trained the VAE and detached the VAE to train the classifier based on the learned feature vector. The number of epochs chosen was based on the convergence of the loss of the model.

3.2. Experiment Setup

In order to analyze and explore the latent space disentanglement and reconstruction results for the various VAE based models, I performed following experiements using MNIST [2] dataset. I have used following CNN architecture of encoder and decoder. I ran the models on **NVIDIA Tesla K80 GPU** available on Google cloud infrastructure.

3.2.1 Encoder/Decoder (VAE, β-VAE)

For encoder I used convolution layer for down-sampling, and for decoder I used de-convolution layer for up-sampling and reconstruction. Architecture of the network is as follows:

Encoder

- 32 filters, 5x5 kernel, stride = 1, padding = 2, ReLU and Maxpool
- 64 filters, 5x5 kernel, stride = 1, ReLU and Maxpool
- 128 filters, 5x5 kernel, stride = 1, padding = 2, ReLU and Maxpool
- 256 filter, 2x2 kernel, stride = 1

Decoder

- 64 filters, 2x2 kernel, stride = 2 , ReLU
- 32 filters, 2x2 kernel, stride = 2 , Sigmoid

Following are other parameters of the network:

- z-dim: 10
- num_epochs: 100
- classifier epochs: 50
- β: 6
- Adam optimizer with 0.001 learning rate for VAE
- Linear classifier is with single layer with softmax on the output.
- Used Adam optimizer for Linear classifier with learning rate = 0.001

Training and test plot for Vanilla VAE and Disentangled VAE are shown in the figures 5, 6:

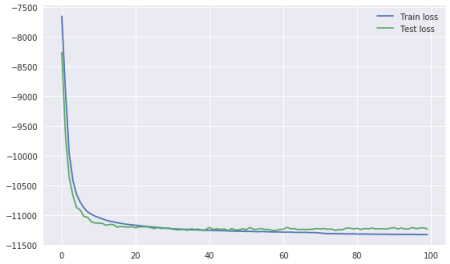

Figure 5. Train and test loss for Vanilla VAE

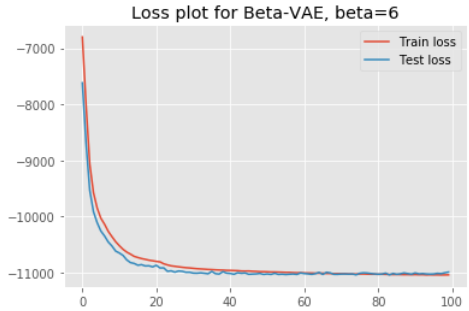

Figure 6. Train and test loss for β VAE

3.2.2 Disentangled Latent Space

In order, to analyze the latent spaces, I performed two kinds of experiments and captured following plots of the models:

1. Experiment 1 - I trained the VAE, detached VAE and trained the classifier
2. Experiment 2 - Trained VAE model and the classifier together

The aim of the experiments is to understand the performance of the latent code generated by the VAE model when the classifier is trained and VAE model is detached, and, when both of them are trained. First experiment is merely to understand the performance of latent code in the classification task and, the second experiment is performed to understand how the performance of the latent code impacted with the additional content loss or classification loss being added through the classifier.

- TSNE plot of the latent space using the maximum variance along two dimensions, the plots of two experiment scenarios is shown in the figure 7

- Captured the plot for the traversal through the latent space for couple of sample test points: 5, 0 and 4 respectively, and the plots are shown in the figures: 8,

- Train and test accuracy plot for downstream classifier, the table 1 discusses the train accuracy and 2 discusses the test accuracy.

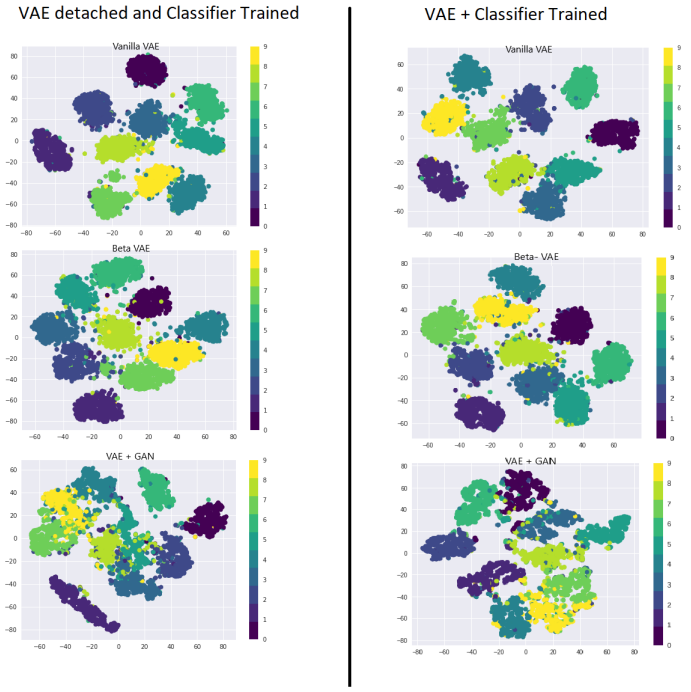

Figure 7. TSNE plot: Left: Vanilla VAE, Right: β-VAE, Bottom: VAE+GAN model

Model	Classifier trained (%)	VAE+ Classifier trained(%)
Vanilla VAE	95.95	96.025
Beta VAE	94.76	95.09
VAE + GAN	89.45	91.08

Table 1. Supervised Classification Train Accuracy

Model	Classifier trained (%)	VAE+ Classifier trained(%)
Vanilla VAE	95.88	96.04
Beta VAE	94.72	95.09
VAE + GAN	90.07	91.71

Table 2. Supervised Classification Test Accuracy

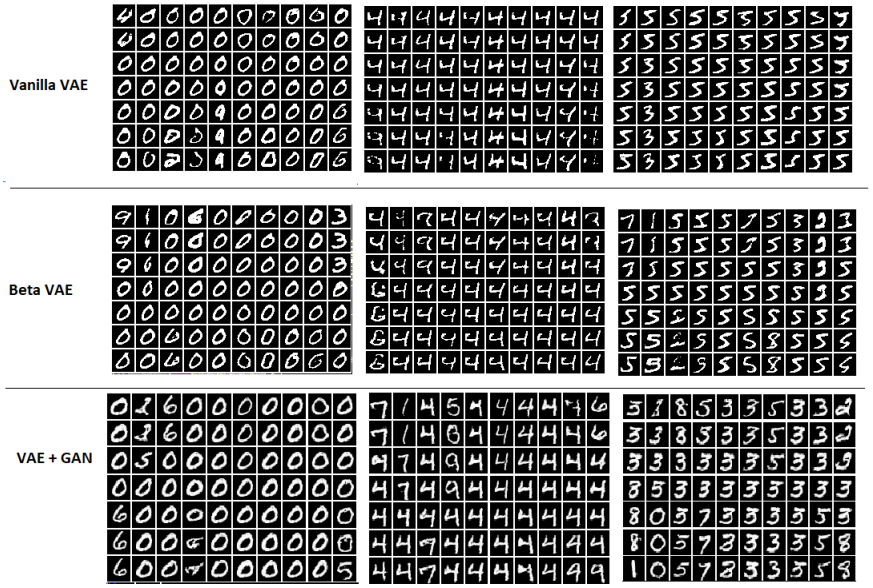

Figure 8. Latent traversal plot for Experiment 1 - for***0,4,5***: Top: Vanilla VAE, Middle: β-VAE, Bottom: VAE+GAN model

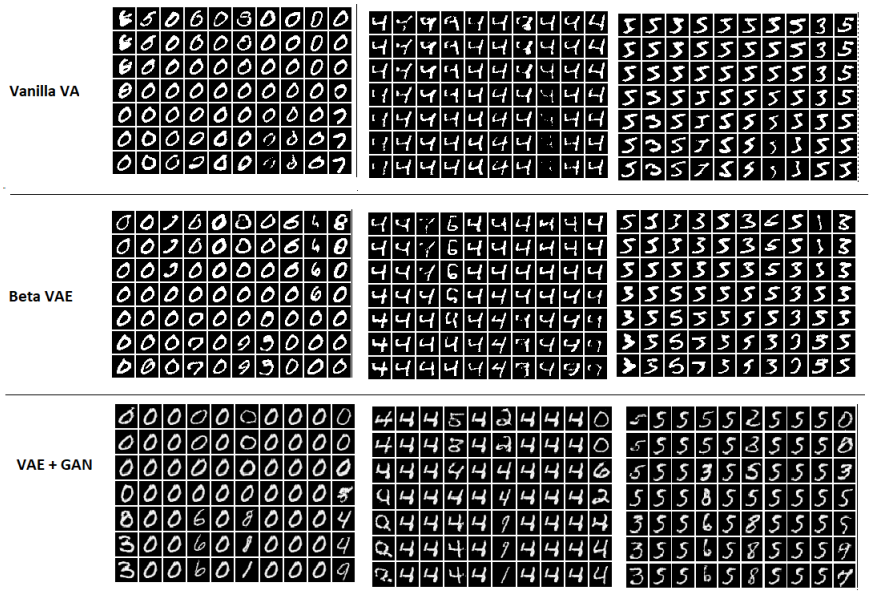

Figure 9. Latent traversal plot for Experiment 2 - for***0,4,5***: Top: Vanilla VAE, Middle: β-VAE, Bottom: VAE+GAN model

Figure 10. Reconstruction results (test): Left: Vanilla VAE, Middle: β-VAE, Right: VAE+GAN model

3.2.3 Reconstructions

In addition to capturing the results of the latent space, I also captured the results to analyze the quality of the reconstruction. I captured following results for each of the models:

- Reconstruction results for test outputs, plot shown in the figure: 10
- Generated samples of the models, plot shown in the figure: 11
- In order to quantify the generated output, I computed Laplacian blur or variance of the generated images. Laplace variance is a quantity that indicates the amount of blur in the image. The laplacian variance increases with increased focus of an image or decreases with increased blur. Further, images with a smaller amounts of edges have smaller laplacian variance. Therefore, I first analyzed the laplacian variance for digits 0-9 in the MNIST test dataset before computing the blur for the generated images. The plot is shown in the figure: 12. Based on the box plot, the laplacian variance of all digits are comparable except for digit class 1. Hence, I compared the laplacian variance for the rest of the reconstructed output ignoring the digit 1 for the models. The plot is shown in the figure:

3.2.4 Adversarial Symmetric Variational Autoencoder

I re-implemented the original tensor flow implementation of [7] available by the author of the paper in pytorch framework [8] and captured the results of the experiment on CIFAR dataset. The figure shows the reconstruction and generated results of the experiment after 10 epochs respectively: 14, 15

3.3. Results

3.3.1 Evaluation Metrics

Classifier Performance: I found the results of classification performance were surprising to me. The table 1 and 2 shows the results from classification performance of my experiments. I found the performance of classifier using vanilla VAE had performed the best reaching highest classification accuracy of $\approx$ 96% compared to other models.

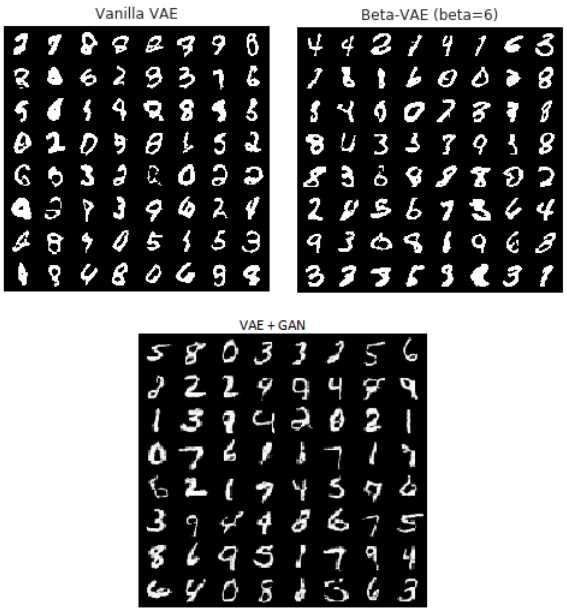

Figure 11. Generated samples: Left: Vanilla VAE, Middle: β-VAE, Right: VAE+GAN model

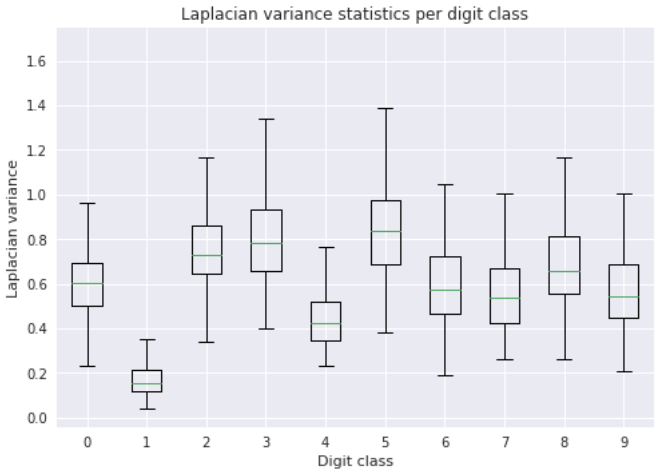

Figure 12. Laplacian variance statistics per digit class

I also found the number of epochs used for training VAE did not have any affect on the performance of linear classifier. The results indicate that the disentangling latent factors in data do not improve the classification performance of content classifier.In addition, the results showed small improvements in performance in my second experiment when the classifier and VAE were trained together. This might or might not be true with other real world datasets that have more latent factors as opposed to MNIST dataset as, MNIST is not a very good representative of the data containing a lot of latent factors.

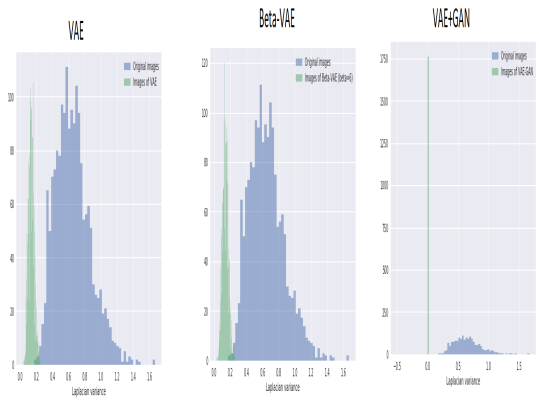

Figure 13. Laplacian Blur for the reconstructed test images

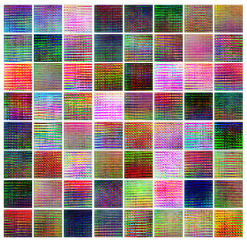

Figure 14. Generated output

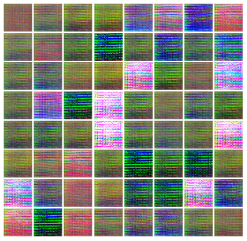

Figure 15. Reconstructed output

3.3.2 TSNE of Latent Space

The results of TSNE plot 7 of the latent space for β-VAE and VAE+GAN was interesting to me. The plot was dense with lot of overlap between the clusters indicating the latent factors were shared between classes. Although, the performance of the classifier was not improved, plots showed that the data was successfully disentangled. TSNE plot of VAE+GAN also had overlap indicating the data was disentangled into its latent features. TSNE plot for Vanilla VAE had isolated clusters which was further manifested in the high classification performance. The plots of all the models from the second experiment showed 7 additional content classifier helped with breaking the overlap.

3.3.3 Latent Traversal

The results of latent traversal related closely with the TSNE plots as shown in the figures 8, 9. As indicated in the TSNE plot, latent traversal of Vanilla VAE was showing traversal for the tested digit while the latent traversal for β-VAE and VAE+GAN model displayed other possible digits that are possible at that latent position (similar to the overlap in their TSNE plot).

3.3.4 Laplacian Blur

The results of Laplacian Blur were more obvious to me as reconstructed output of GAN models are more sharper than VAE models. Laplacian variance was highest for VAE+GAN indicating generated results had a lot of focus. The laplacian blur for Vanilla VAE is higher than β-VAE as β-VAE loss model has more emphasis on disentanglement as opposed to reconstruction due to β value.

4. Conclusion

I compared the performance of various VAE based models in my work and analyzed the performance of the models in terms of their learned latent vector using linear classifier. I trained all the models on MNIST dataset. I found the disentangled VAE and VAE+GAN did not perform well for the content classification task. Along with these models, I also ran the models InfoGAN [9] and Disentangling by factorizing [10] using the code provided by the author of the paper. However, I did not capture enough metrics to compare these models with my current models. From the results of InfoGAN [9] I found the reconstruction results were sharper and comparable to VAE+GAN results. The figure describe the results of InfoGAN 16 For future work,

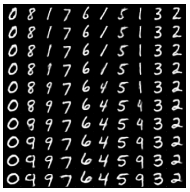

Figure 16. Reconstructed output for InfoGAN

I would like to redo the experiments using the dataset that contains lot more disentangled features and would like to compare the classification performance of such models in the supervised and semi-supervised setting. I would like to complete my experiment of using Adversarial Symmetric Variational autoencoder using MNIST dataset and understand the use of symmetric KL divergence in supervised and semi-supervised classification tasks. I also would like to complete implementation for my proposed paper [11]. In addition, during my current work I analyzed the quality of

the latent code or the output of the encoder and, for the future tasks I would like to analyze the quality of the decoder.

4.1. Challenges

Some of the challenges I found during my experiments is availability of GPU resources. Although, google cloud and google colab resources were available I frequently ran into timeout and connectivity issues. I was also constrained by limit on GPU memory which in turn was cost dependent and, that limited my ability to run models simultaneously and capture more metrics and analyze more models. Due to time constraints and resource constraints, I felt I couldn't reach my intended target.

References

[1] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[2] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[4] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

[5] Yuchen Pu, Weiyao Wang, Ricardo Henao, Liqun Chen, Zhe Gan, Chunyuan Li, and Lawrence Carin. Adversarial symmetric variational autoencoder. In *Advances in Neural Information Processing Systems*, pages 4330–4339, 2017.

[6] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, volume 3, 2017.

[7] Yuchen Pu, Weiyao Wang, Ricardo Henao, Liqun Chen, Zhe Gan, Chunyuan Li, and Lawrence Carin. Adversarial symmetric variational autoencoder. In *Advances in Neural Information Processing Systems*, pages 4330–4339, 2017.

[8] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[9] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

[10] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

[11] Adrià Ruiz, Oriol Martinez, Xavier Binefa, and Jakob Verbeek. Learning disentangled representations with reference-based variational autoencoders. *arXiv preprint arXiv:1901.08534*, 2019.