ADVANCED MACHINE LEARNING

FINAL PROJECT REPORT

SUBMITTED BY

MD MAHIN (PSID: 1900421)
MAHSA REZAEI FIRUZKUHI (PSID: 1864080)
ABHIJIT BAUL (PSID: 2078830)

Department of Computer Science
University of Houston

MAY, 2022

Optimizing the Beta Value of β-VAE using N-arm-bandit and Simulated Annealing

Md Mahin, Mahsa Rezaei Firuzkuhi, Abhijit Baul
mmahin@uh.edu, Mrezaeif@uh.edu, abaul@uh.edu
University of Houston

***Abstract*—β Variational Auto-Encoder(VAE) is a popular variation of VAE proposed to automate the learning process of factorized latent representation more efficiently without any supervision. A β value greater than 1 enforces the model to learn disentangled representation more accurately. Disentangled representations emerge when the right balance is found between information preservation (reconstruction cost as regularisation) and latent channel capacity restriction ($\beta > 1$). However, the optimal value of β is dataset dependent and varies from dataset to dataset. In this paper, we tried to learn beta values based on the loss function automatically. In this method, we assume the optimal β value exists within n beans above 1, and We employ a Reinforcement Learning (RL) based n-arm bandit, where each arm tries to estimate the bean with the lowest loss during the training process. The optimal β value is found by applying simulated annealing within each bean. Result analysis shows that our approach performs better image generation than normal VAE and β VAE with random β, but reconstruction performance is slightly lower than normal VAE.**

Keywords: VAE, $Beta$-VAE, N-arm-bandit, Simulated Annealing

I. Introduction

Unsupervised feature learning is the process of learning representation from unlabeled data. Unsupervised Representation learning frequently seeks to uncover low-dimensional features that represent some structure beneath the high-dimensional input data. There are two main categories of a representation learning algorithm [1]: Deep learning based, Clustering based algorithms. Deep learning-based approaches fall into two categories Autoencoder based approaches and graphical model-based approaches. In autoencoder based approaches convolutional autoencoders [2] or stacked autoencoders [3] are used to separate ladder structures [4] and in general, main components of an input. These approaches encode an input into a compact code and decode the code to reconstruct the input as accurately as possible. This learning process can be labeled as Disentangled Representation Learning.

Disentangled Representation Learning[5][6] indicates learning techniques that break down high dimensional input features into lower dimensional features. In other words, a representation is disentangled if it can be broken down into a number of independent lower dimensional sub-spaces, where each sub-spaces are only dependent on the action of a single subgroup and independent from other subgroups. Disentangled Representation Learning is important for latent factor representations. Latent factor representations indicate the lower dimensional representations that can explain higher dimensional data and using what higher dimensional data can be reproduced[**?**]. Disentangle representations can also boost the transfer of knowledge using transfer learning.

Variational Autoencoders(VAE) [7] are popular methods to learn the latent representations for any high dimensional representations. Higgins et al. [5] have proposed a modified version of VAE β-VAE to learn the disentangled representations better. β-VAE introduces a β term that is multiplied with the KL-divergence term D_{KL} in the loss function. β value greater than 1 enforces the Autoencoder to learn disentangled representation better by restricting latent channels capacity; however, too much pressure from high β value sacrifices reconstruction accuracy. To find an optimal β value, it is important to find a balance between the reconstruction loss and latent channel capacity.

This research aims at finding the optimal β value during the training time using the change in the loss function. This paper first assumes the optimal β value exists within n beans greater than 1. The beans can have any range of values and are non-overlapping. It uses an n-arm bandit to find the bean that provides minimum loss during the training process. β value can be the best value within each bean. However, as the nature of the loss function within each bean can be random, a Simulated Annealing algorithm is applied to find the optimal value from each bean. The optimal β value is the optimal value from the bean that has the lowest loss for the training. The motivation of this research is if the β value can be made dependent on the input samples, it will be easier to find the optimal β value during any knowledge transfer scenario. For that reason, this research tries to make the β value dependent on the loss update, which is dependent on the input.

Using n-arm-bandit is to reduce search space and thus reduces variance within the result. Simulated Annealing is chosen as it provides enough exploration at the beginning so that it is less likely to be stuck at local optima, and thus it is better than random hill-climbing.

The contribution of this paper is a framework to optimize the β value of β-VAE automatically during the training time.

Organization: The rest of the paper is organized as follows. Section II reviews the related works. Section III background, Section IV introduces the proposed method. Section V presents the experiment settings, Section VI results, and evaluations, and finally, section VII concludes the paper.

II. Related Works

A. Deep learning-based generative models

A Generative Model is a way of learning the data distribution using unsupervised learning. The generative models learn the true data distribution of the training sets with some variations to generate new data points from the training set. The two most efficient solutions are using the Variational Autoencoders (VAE) and Generative Adversarial Networks (GAN). VAE aims at maximizing the lower bound of the data log-likelihood, and GAN aims at achieving a balance between generator and discriminator.

The paper [8] was the first known utilization of adversarial learning in generative models. The main contribution of Generative Adversarial Networks is that they don't need any Monte Carlo approximations to train. Compared to variational autoencoders, GANs don't introduce any deterministic bias. GANs generate samples faster than fully visible belief nets. On the other hand, it's hard to learn to generate discrete data, like text. GANs generally suffer collapse mode, which means that the diversity of generated samples doesn't change as expected after synthesizing a certain number of instances.

The paper [**?**] was the first known research for the application of deep autoencoders in generative models. Variational autoencoders have provided a modified bottleneck structure for traditional autoencoders to capture the mean and standard variation among the training instances. These two hidden layers collaborate to form the bottleneck and would be used as a tool to synthesize the novel instances. Since VAE tries to minimize the reconstruction error of the input training data, it can easily ignore outliers that are critical to generating new instances.

The paper [9] was the first known research to justify a clear theoretical approach for a generative model based on both GANs and VAEs named Adversarial Variational Bayes. The main contribution was to make the inference model much more flexible, effectively allowing it to represent almost any family of conditional distributions over the latent variables. They did not propose any solution to obviate collapse mode, which is seen in adversarial-based generative models since the discriminator is trained so quickly and doesn't allow the generator to capture all the manifold.

The paper [10] developed a scalable unsupervised approach for disentangled factor learning called InfoGAN. InfoGAN extends the generative adversarial network [8] framework to additionally maximize the mutual information between a subset of the generating noise variables and the output of a recognition network. It can discover a subset of generative data factors and learns a disentangled representation of these factors. The training instability and reduced sample diversity are the results of the GAN framework on which the InfoGAN is based.

B. Deep autoencoders in generative models

The original VAE framework has been shown to achieve limited disentangling performance on simple datasets, such as MNIST, and disentangling performance does not scale to more complex datasets. The paper [5] proposed a deep unsupervised generative approach for disentangled factor learning called β-VAE that can automatically discover the independent latent factors of variation in unsupervised data. They applied a protocol to quantitatively compare the degree of disentanglement learned by different models on a variety of complex datasets. In the β-VAE model, they increased the Kullback-Leibler term in the VAE objective in order to optimally use the model capacity and learn a meaningful representation. The

VAE model assumes that the observations are independent and identically distributed. In the case of grouped observations, this assumption is no longer true, and the inference is not efficient. The paper [11] proposed the Multi-Level Variational Autoencoder (ML-VAE), a new deep probabilistic model that learns a disentangled representation of a set of grouped observations. This model separates the latent representation into semantically meaningful parts by working both at the group level and the observation level.

The paper [12] developed a model called Factor VAE, which decreases the total correlation of the latent variables and increases the independence of the latent, resulting in a more robust disentangled representation.

III. Background

A. VAE Loss Function

In the VAE model, a network (the encoder) encodes an observation ($x \in X$) into its latent representation z and a generative network (the decoder) decodes an observation from the latent code z. Here, X is all samples. The VAE model learns latent representation in form of probability distribution denoted by $P(z)$. An isotropic unit Gaussian distribution $p(z) = N(0, I)$ is selected as a prior to learn the distribution $P(z)$. Later the model can sample many latent variables from $P(z)$ and generate x from these variables. VAE uses variational inference to infer $P(z)$ using $P(z|X)$ [13]. VAE first model $P(z|X)$ using simpler distribution $Q_\phi(z|X)$ and minimize the difference between $P(z|X)$ and $Q_\phi(z|X)$ using KL-divergence metric approach. As a result, the objective function of variational autoencoder is as the following equation:

$$D_{KL}[Q_\phi(z|X)|| < \epsilon] \tag{1}$$

$$F(\theta, \phi, \beta; x, z) = logP_\theta(X) - D_{KL}[Q_(z|X)||P(z|X)] \tag{2}$$

we can write the equation as:

$$= E(logP_\theta(X|z) - \beta(D_{KL}[Q_(z|X)||P(z|X)] - \epsilon)) \tag{3}$$

The term $Q_\phi(z|X)$ is representing the encoder network, z is the encoded representation of data x , where $(x \in X)$

and $P(X|z)$ is the decoder network. As mentioned above, based on equation 2, the goal is to maximize the log-likelihood of our data distribution considering the error given by $D_{KL}[Q_(z|X)||P(z|X)]$. The loss function will contain two terms. The first one is the reconstruction loss of the input to output and the second loss is the KL-divergence term. Since $P(z|X)$ is not tractable and the KL-divergence term is always greater than or equal to zero, the VAE is trying to minimize the lower bound of $logP_\theta(X)$. So, the model is trying to maximize the $E(logP(X|z))$ and minimize the $D_{KL}[Q_\phi(z|X)P(z|X)]$.

B. *β-VAE Loss Function*

As proposed by the paper [5], a multiplier $\beta \geq 1$, which is the regularization coefficient, is added to the loss function. This coefficient constrains the capacity of the latent information z and puts implicit independence pressure on the learned posterior due to the isotropic nature of the Gaussian prior $p(z)$. The KL-divergence is a measure of dependence between the latent dimensions. Increasing β forces the model to find statistically independent factors of variation in the data distribution. Setting $\beta > 1$ puts a stronger constraint on the latent bottleneck than in the original VAE formulation from paper [**?**] and limits the capacity of z. This constraint, in addition to maximizing the log-likelihood of the training data x, forces the model to learn the most efficient representation of the data. As mentioned in the paper [5], increasing the value of β should force the model to learn a disentangled representation of v. On the other hand, the extra pressures coming from high β values provide a trade-off between reconstruction and the quality of disentanglement within the learned latent representations. The right balance between information preservation and latent channel capacity restriction ($\beta > 1$) provides the disentangled representations. Since the hyperparameter β directly affects the degree of learned disentanglement and depends on the value of ϵ in 2, finding the optimal value of β for each dataset and model architecture is necessary.

C. *Multi-Arm Bandit using Upper Confidence Bound*

In the Multi-armed bandit problem, the model must choose between multiple actions, and each selection will have an unknown payout. The goal is to determine the most profitable outcome through a series of choices. The Multi-Arm Bandit problem is modeled as an iterative interaction between a learner and a stochastic environment. The environment is a set of K unknown distributions where at each round t, the learner selects an action and therefore receives a random sample from action distribution at round t (independent of the past samples). The goal is to define an allocation strategy over the actions to estimate their expected values uniformly well. Since the variances of the actions (distributions) are not initially known, the learner should follow an online allocation strategy that adapts its behavior as samples are collected. The upper confidence bound (UCB) algorithm [14] assigns a confidence boundary to each action on each round of exploration when the learner chooses one action. In other words, the UCB algorithm focuses on exploitation which is selecting the action with the highest estimated reward and changes its exploration-exploitation balance as it realizes more knowledge of the environment. Initially, when it has low confidence in the best actions to take, the exploration part of its equation causes it to search through the set of all possible actions. Reward values for each action will be updated as exploration makes progress and better estimates are formed for each action. As a result, the level of exploration can be decreased, the selection of the good actions that have been found can increase, and the algorithm focuses more on the exploitation and reduces its regret.

D. *Simulated Annealing*

Simulated Annealing (SA) is a global search optimization algorithm that can be used in the combinatorial optimization tasks to minimize the defined cost function in a system with multiple degrees of freedom [15]. SA is a popular search method when searching any value within a random function, as it provides enough exploration at the beginning and explores less with time. As a result, it will be less stuck to sub-optimal regions, which makes it better in comparison to random hill-climbing. SA is used to improve the performance of Convolution Neural Network (CNN) using a modern optimization technique called metaheuristic algorithm. It finds the global extreme for the function that has local minimums and large search space with a substantial reduction in computation time.

IV. Methodology

Here we will discuss our overall β optimization technique. The β optimization algorithm can mainly be divided into three steps:

- β selection Step
- β evaluation Step
- Exception step

The algorithm 1 shows all steps combined. Next, we will divide the functionality of the steps within the algorithm.

A. *β Selection Step*

At the β selection step, we first select a β range using an n-arm bandit. In the algorithm 1, line 13 to 31 represents the selection step, excluding 16 to 26. Line 16 to 25 represents the exception step. In line 14, the algorithm selects the β range. Next, a candidate value within the β range is selected using the standard simulating annealing approach from lines 26 to 31. Here, the candidate beta value is generated using a random value, the current beta position within the arm, and the current step size for the arm. Next, we directly back-propagate the loss function and evaluate its performance during the next iteration. After a candidate value is generated, β generation is turned off for the next iteration at line 31.

B. *β Evaluation Step*

In the evaluation step, we evaluate how much our training performance improved or degraded due to the previous candidate value. In the algorithm 1, line 32 to 47 represents the evaluation step. Here at first, we calculate the same loss using the previous candidate value. Next, we try to find if the current

```
1  Input: number of arm n, range of arms d, temperature
     T, step size percent s, step size update rate p
2  Output: $\beta$ value
3  initialize selection_count vector for n arms
4  initialize total_reward vector for n arms
5  initialize best_value vector for n arms
6  initialize current_value vector for n arms
7  initialize best_loss vector for n arms
8  initialize current_loss vector for n arms
9  initialize step_size for n arms using range and percent
10 initialize previous_loss = 0 and arm_selection = True
11 for each epoch do
12     for each iteration do
13         if arm_selection == True then
14             Select an arm from n arms using UCB
15             update selection_count[arm] += 1
16             if selection_count[arm]==1 then
17                 candidate = mean(arm_range[arm])
18                 best_value[arm]= candidate
19                 current_value[arm]= candidate
20                 calculate and back-propagate loss
21                 previous_loss = loss
22                 best_loss[arm]= 1/loss
23                 current_loss[arm]= 1/loss
24                 total_reward[arm] += 1/loss
25             end
26             if $selection\_count[arm] > 1$ then
27                 candidate = current_value[arm] +
                     random_number*step_size[arm]
28                 calculate and back-propagate loss
29                 arm_selection == False
30             end
31         end
32         if arm_selection == False then
33             calculate candidate_loss
34             if $1/candidate\_loss >$
                 $best\_loss\_per\_arm[arm]$ then
35                 best_value[arm] = candidate_value
36                 best_loss[arm] = 1/candidate_loss
37             end
38             difference = 1/candidate_loss -
                 current_loss[arm]
39             Check candidate_probability if candidate
                 being selected over current_value[arm]
                 using the temperature
40             if candidate_probability is okay then
41                 current_value[arm] = candidate_value
42                 current_loss[arm] = 1/candidate_loss
43             end
44             back-propagate current_loss[arm]
45             total_reward[arm] += (previous_loss -
                 current_loss[arm]) previous_loss =
                 1/current_loss[arm]
46             update temperature and step_size[arm]
47         end
48     end
49 end
```

Algorithm 1: β Optimization Algorithm

beta value is the best beta value within the arm by comparing the improvement of their loss. We use $\frac{1}{loss}$ every step for certain reasons discussed in our reward function section. Here if the loss improves, the fraction will be greater than the best loss, and both the best loss and best value will be updated. Next, we try to update the current search position from lines 40 to 43. Here we first calculate the difference between the candidate loss and current loss and use the temperature value. If the probability is higher than a random number as in standard simulated annealing, we update the current position and current loss. Next, we back-propagate the loss and save it as a previous loss to compare it with the next arm. The reward of each arm is also dependent on how much the current arm improved from the loss generated by the previous arm.

C. Exception Step

In the algorithm 1, line 16 to 25 represents the exceptions of normal steps. During the first iteration for each arm, we do not have any previous step; we just select the mean of the arm as a candidate and back-propagate that, and the reward is updated by considering the previous loss is 0. These updates are compared during the next iteration for the arm.

D. Reward Function

Our algorithm is highly dependent on the accuracy of the reward calculation for the n-arm bandit, as the reward will determine which range provides the most optimal β. We have to consider several circumstances before coming up with our reward function. The circumstances are:

- At the beginning of training difference in losses will be huge, and they will go down with time
- High reward at the beginning will overshadow lower rewards at the end
- Reward function should increase the reward with lower losses

  To consider all these situations we developed the following reward function:

  $$reward = \frac{1}{previous\_loss} - \frac{1}{current\_loss} \quad (4)$$

  In equation 4 we calculate $\frac{1}{loss}$. It will make bigger loss smaller. As a result, initial higher losses will have smaller values, and lower losses from the end will have lower values. Also, it will solve the overshadowing problem as a fraction will make the number very small. The next difference between previous and current loss will generate a positive value if the current loss is smaller or the reward will be negative.

V. Experiment Settings

A. Model Architecture

For our experiment we have created a simple VAE encoder and decoder architecture with five fully connected layers. Our VAE architecture configuration mentioned below:

- Total number of layers: 5
- Layers type: Fully connected

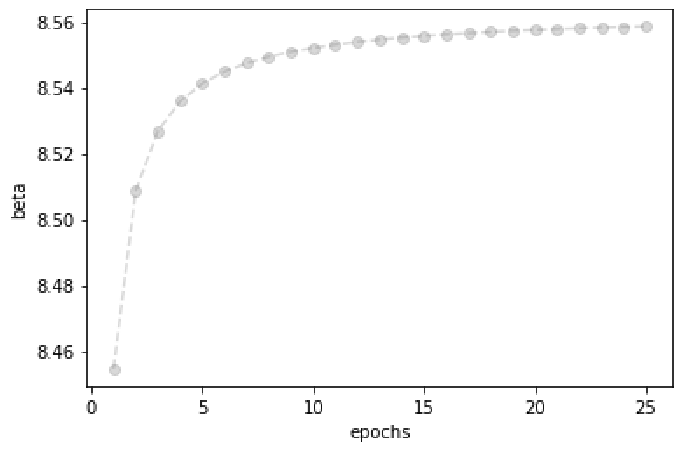


Fig. 1: β Update for Different Epochs

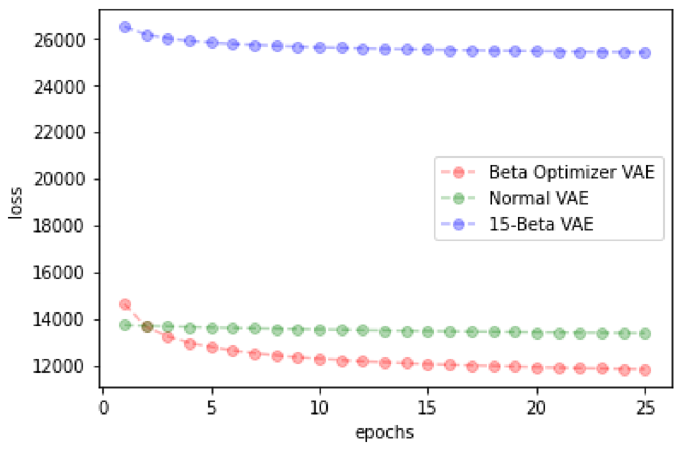


Fig. 2: Loss Functions Update for Three Models

- Input and output dimension: 784
- Hidden layer dimension: 400
- Latent layers dimension: 20
- Optimizer: Adam Optimizer

B. Experiment Settings

For our analysis we have trained three models with same parameters:

- β optimization model (Our proposed model)
- Normal Variational Autoencoder (VAE) without any β
- Random β Variational Autoencoder (VAE) with $\beta = 15$

Our choice of hyper-parameters are:

- Learning rate: 0.001
- Number of epoch: 25
- Batch size: 128

All models are trained in the Google-Colab.

C. Dataset

For our analysis, we have used MINST handwritten image dataset [16]. MINST dataset contains 60000 training and 10000 testing grayscale images with a resolution of 28×28 pixels.

VI. Result Analysis

A. Beta Learning

Figure 1 shows our β update for different epochs. We can see, that the β value changes first at the beginning of the training procedure but stabilize around 8.56 at the end of the 25th epoch slowly. Most frequent changes occur first few epochs, but from epoch three, it starts stabilizing.

B. Loss Function Comparison

Figure 2 shows the average loss update from the three models during each epoch. From the figure, we can see that our model Beta Optimization VAE improves mostly and gives the lowest loss as the training goes on. On the other hand, Normal VAE gives a constant loss, which is very close to our loss. Random VAE performs worst here, which makes sense as the optimal β value is unknown for the training sample, and a random value is used.

C. Sample Generation Performance Comparison

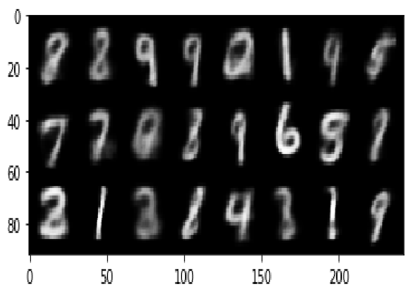
Figure 3a, 3b , 3c shows the generation of different handwritten characters from three different models. From the generated images, it can be seen that normal VAE learns specific parts better and generates them better. However, generation from normal VAE is not uniform. On the other hand, both β VAEs learn more uniformly than the normal VAE. Among the β VAE, out β Optimizer VAE generates more clear images, and it performs best in all three models.

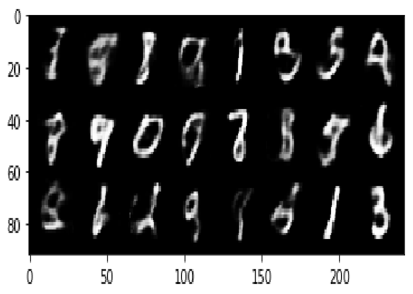
D. Sample Reconstruction Performance Comparison

Figure 4a, 4b , 4c shows the reconstruction of different handwritten characters from three different models. Every left character is the image provided as input to the encoder and the right character is the output of the decoder. From the reconstructed images, it appears normal VAE performs most better. Out β Optimizer VAE performs second, better than random β VAE with $\beta = 15$. It indicates that during the training process we are sacrificing some reconstruction accuracy. However, the loss of performance can be due to limited training too even though all the models are trained the same number of epochs and iterations, β Optimizer VAE back-propagates loss 50% times during the training process. As a result, the reconstruction accuracy falls behind the normal VAE.

VII. Conclusion

β-VAE, a variation of VAE, is proposed to improve disentangled representation learning in a complete unsupervised approach. However, the β value from β-VAE is completely dataset dependent, and the model is not truly trained unsupervised if it is not learned automatically. In this paper, we propose an n-arm-bandit and Simulated Annealing based β optimization technique, which learns the β value automatically


(a) Sample Generation using β Optimizer VAE


(b) Sample Generation using Normal VAE

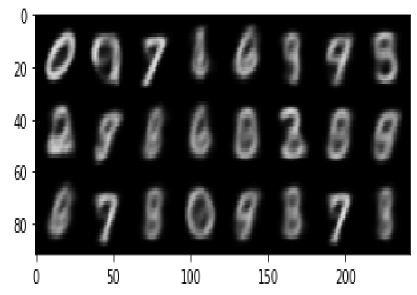
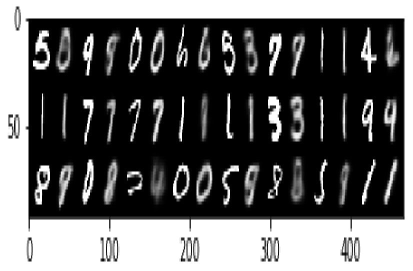
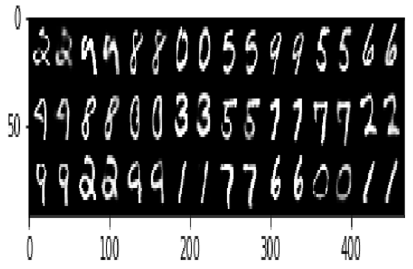

(c) Sample Generation using β VAE where β = 15

Fig. 3: Image Generation Comparison from Three Models


(a) Sample Reconstruction using β Optimizer VAE


(b) Sample Reconstruction using Normal VAE

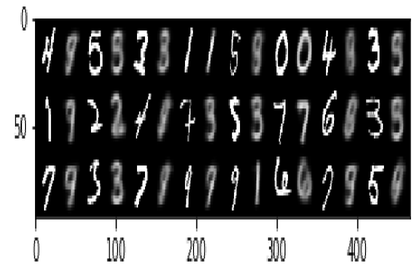

(c) Sample Reconstruction using β VAE where β = 15

Fig. 4: Image Reconstruction Comparison from Three Models

based on the update of the loss function. Our approach improved the image generation performance of VAE but lost reconstruction accuracy slightly. Another problem with our approach is run-time almost get doubled. In the future, we can try to optimize the β value by keeping the run-time fixed and making the β value more dataset dependent, so it can directly be utilized in transfer learning.

References

[1] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[2] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." *Journal of machine learning research*, vol. 11, no. 12, 2010.

[3] J. Zhao, M. Mathieu, R. Goroshin, and Y. Lecun, "Stacked what-where auto-encoders," *arXiv preprint arXiv:1506.02351*, 2015.

[4] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with ladder networks," *Advances in neural information processing systems*, vol. 28, 2015.

[5] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," 2016.

[6] I. Higgins, D. Amos, D. Pfau, S. Racaniere, L. Matthey, D. Rezende, and A. Lerchner, "Towards a definition of disentangled representations," *arXiv preprint arXiv:1812.02230*, 2018.

[7] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *arXiv preprint arXiv:1906.02691*, 2019.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[9] L. Mescheder, S. Nowozin, and A. Geiger, "Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2391–2400.

[10] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," *Advances in neural information processing systems*, vol. 29, 2016.

[11] D. Bouchacourt, R. Tomioka, and S. Nowozin, "Multi-level variational autoencoder: Learning disentangled representations from grouped observations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[12] H. Kim and A. Mnih, "Disentangling by factorising," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2649–2658.

[13] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[14] A. Carpentier, A. Lazaric, M. Ghavamzadeh, R. Munos, and P. Auer, "Upper-confidence-bound algorithms for active learning in multi-armed bandits," in *International Conference on Algorithmic Learning Theory*. Springer, 2011, pp. 189–203.

[15] L. R. Rere, M. I. Fanany, and A. M. Arymurthy, "Simulated annealing algorithm for deep learning," *Procedia Computer Science*, vol. 72, pp. 137–144, 2015.

[16] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.