Project Report - ECE 176

Kaja Erfjord
Computer Science Engineering
PID: U09444878

Priyansh Bhatnagar
Electrical and Computer Engineering
PID : A59019463

Abstract

In this project, we have studied the generative models and dived deep into the implementation of a couple of state-of-art models - Variational Autoencoders(VAE) and Conditional Variational Autoencoders(CVAE). Additionally, this report includes a comparative analysis of both models.

1 Introduction

1.1 Problem definition and Motivation

In this project, we have built VAEs and CVAEs to generate handwritten digit images and compared their performance and computational efficiency.

The motivation for solving the problem of generative modeling using VAEs and CVAEs is rooted in their potential applications across various fields. Moreover, the recent surge in popularity of variational autoencoders has made them an attractive choice for building generative models. The primary aim of exploring this topic is to gain a comprehensive understanding of generative modeling, including the underlying mathematical concepts, and develop the ability to implement cutting-edge models that are currently in use in the industry.

1.2 Our understanding of the problem

The problem of generative modeling involves creating a model that can generate new samples similar to the training data it has seen. VAEs and CVAEs are two popular approaches for generative modeling that use neural networks to learn the underlying probability distribution of the data and generate new samples from it.

VAEs use an encoder-decoder architecture, where the encoder maps the input data to a latent space, and the decoder generates new samples from the latent space. VAEs incorporate a variational lower bound that ensures the generated samples are diverse and representative of the training data.

CVAEs are an extension of VAEs that incorporate additional conditioning information, such as labels, into the model. This allows for a more controlled generation of samples that are specific to a given label or class.

By building and comparing the performance and computational efficiency of VAEs and CVAEs for generating handwritten digit images, we can gain insights into their strengths and limitations and determine which approach is better suited for specific applications.

1.3 Theory - Variational Autoencoders (VAE)

Variational Autoencoders (VAEs) are a type of neural network architecture used for unsupervised learning and generative modeling. VAEs are based on the concept of autoencoders, which are neural networks that learn a compressed representation of the input data, known as the latent space. [1]

In VAEs, the encoder maps the input data to a distribution over the latent space, rather than a fixed point, as in traditional autoencoders. The decoder then generates new samples by sampling from this distribution and reconstructing the input data.

The key innovation of VAEs is the incorporation of a variational lower bound that enforces the generated samples to be diverse and representative of the training data. This lower bound is used to optimize the model's parameters during training and encourages the learned latent space to be smooth and continuous, making it easier to sample from and generate new samples.

VAEs have several advantages over traditional autoencoders, including their ability to generate new samples, handle missing data, and perform unsupervised learning. They have been successfully applied in various fields, such as image and speech recognition, and have become a popular choice for building generative models.

The goal of a VAE is to learn a generative model that can capture the underlying distribution of the data and generate new samples from it. The model is trained using an encoder-decoder architecture that maps the input data to a low-dimensional latent space and then maps the latent space back to the input data.

The encoder maps the input data x to the parameters of a distribution over the latent space z, which is typically assumed to be a multivariate Gaussian distribution with diagonal covariance:

$$q_\phi(z|x) = \mathcal{N}(z|\mu_\phi(x), \sigma^2_\phi(x)I)$$

where $\mu_\phi(x)$ and $\sigma^2_\phi(x)$ are the mean and variance of the distribution, respectively, and ϕ represents the parameters of the encoder. [1]

The decoder maps a sample z from the latent space to the parameters of a distribution over the output data x:

$$p_\theta(x|z) = \mathcal{N}(x|\mu_\theta(z), \sigma^2_\theta(z)I)$$

where $\mu_\theta(z)$ and $\sigma^2_\theta(z)$ are the mean and variance of the distribution, respectively, and θ represents the parameters of the decoder.

To train the model, we optimize a variational lower bound on the log-likelihood of the data, known as the Evidence Lower Bound (ELBO). The ELBO is given by:

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}q\phi(z|x) \left[\log p_\theta(x|z)\right] - D_{KL}\left(q_\phi(z|x)||p(z)\right)$$

where the first term is the reconstruction loss, which measures the difference between the input data and its reconstruction, and the second term is the KL divergence, which measures the difference between the learned distribution over the latent space and the prior distribution $p(z)$ [1], which is assumed to be a standard normal distribution:

$$p(z) = \mathcal{N}(z|0, I)$$

The reconstruction loss can be written as:

$$\mathbb{E}q\phi(z|x) \left[\log p_\theta(x|z)\right] = -\frac{1}{2}\sum_{i=1}^{n}\left(1 + \log(\sigma^2_\theta(z)) - \mu_\theta(z)^2 - \sigma^2_\theta(z)\right)$$

where n is the dimensionality of the input data.

The KL divergence term can be written as:

$$D_{KL}\left(q_\phi(z|x)||p(z)\right) = -\frac{1}{2}\sum_{i=1}^{n}\left(1 + \log(\sigma^2_\phi(x)) - \mu_\phi(x)^2 - \sigma^2_\phi(x)\right) + \frac{1}{2}\sum_{i=1}^{n}(\mu_\phi(x)^2 + \sigma^2_\phi(x) - 1)$$

To optimize the ELBO, we use stochastic gradient descent (SGD) or a related algorithm, such as Adam, to update the parameters θ and ϕ based on the gradients of the ELBO with respect to these parameters.

Overall, the VAE framework provides a powerful approach for learning generative models that can capture the underlying distribution of the data and generate new samples from it, while also ensuring that the model is regularized by imposing a prior distribution over the latent space.

1.4 Theory - Conditional Variational Autoencoders (CVAE)

Conditional Variational Autoencoders (CVAEs) are an extension of Variational Autoencoders (VAEs) that incorporate additional conditioning information into the generative modeling process. In traditional VAEs, the model learns to generate new samples from a latent space that is solely based on the input data. However, in CVAEs, the model also learns to generate samples based on a set of additional conditional variables.

In a CVAE, the input data is first encoded into a low-dimensional latent space using an encoder neural network. The latent representation is then used as input to a decoder neural network, which generates a reconstruction of the original input. Additionally, the CVAE includes a conditional input, which is typically used to control the generation of new data.

The CVAE is trained to minimize a loss function that includes a reconstruction loss and a regularization term to ensure that the latent representation follows a desired distribution. The regularization term is typically implemented using the Kullback-Leibler (KL) divergence between the distribution of the latent representation and the desired prior distribution.

The mathematical formulation of a CVAE is as follows:

Let x be the input data, y be the conditional input, and z be the latent variable. Let the encoder be represented by the function q(z|x,y), the decoder be represented by the function p(x|z,y), and the prior over the latent variable be represented by the distribution p(z|y) [1].

The reconstruction loss is given by the negative log-likelihood of the data under the decoder:

$$L_{rec}(x, y) = -\log p(x|z, y)$$

where z is sampled from the encoder distribution q(z|x,y).

The KL divergence between the encoder distribution q(z|x,y) and the prior distribution p(z|y) is given by:

$$D_{KL}\left[q(z|x, y)||p(z|y)\right] = \frac{1}{2}\sum_{i=1}^{d}\left(1 + \log(\sigma_i^2) - \mu_i^2 - \sigma_i^2\right)$$

where μ and σ are the mean and standard deviation of the encoder distribution q(z|x,y), and d is the dimension of the latent variable z.

The overall loss function for training the CVAE is given by:

$$L(x, y) = L_{rec}(x, y) + \beta D_{KL}\left[q(z|x, y)||p(z|y)\right]$$

where β is a hyperparameter that controls the weight of the regularization term in the loss function [1].

During training, the CVAE is optimized by minimizing this loss function with respect to the parameters of the encoder and decoder networks. Once the CVAE is trained, it can be used for conditional generation by providing a desired conditional input y and sampling a latent variable z from the prior distribution p(z|y). The decoder network is then used to generate a new sample x' from the latent variable z and the conditional input y.

2 Related Work

In our project, we explored two state-of-the-art generative models - Variational Autoencoders (VAEs) and Conditional Variational Autoencoders (CVAEs). Our work is related to several papers, including "Auto-Encoding Variational Bayes" by Kingma and Welling [2], which introduced VAEs and their training algorithm, and "Learning Structured Output Representation using Deep Conditional Generative Models" by Sohn et al.[5], which proposed the CVAE model.

Additionally, we analyzed and compared the performance of these models on various datasets, including MNIST, in line with the work presented in "Variational Inference for Monte Carlo objectives" by Mnih and Gregor [4], which evaluated VAEs on the MNIST dataset. In addition to the paper called "Deep Convolutional Inverse Graphics Network" by Kulkarni et al. [3], which demonstrated the applicability of VAEs for generating images of physical scenes.

3 Method

As mentioned above we have tried to explore generative modelling using one of the popular techniques that is Variational Autoencoders. We have gone a bit beyond traditional 'vanilla' variational autoencoders and tried to explore Conditional variational autoencoders (CVAE) as well.

We have built our custom encoders, decoders, and loss functions for both variational and conditional variational autoencoders. Morever, there is a bottleneck present in the architecture of any kind of autoencoder. This bottleneck is the primary area of focus while training our model. It represents/ learns a latent space (of less dimensions than the input). As a addition to generating images, we have also visualized the best-estimated/learnt latent space.

We have tried several architectures for VAE and CVAE and finally come up with a model with maximum performance and least parameters, so that it is computationally efficient. Following is the architecture of VAE and CVAE that we have come up with and implemented.

The architecture that we have finalized for Variational Autoencoder is as follows. First there is an encoder with 3 fully connected layers .2 parameters - mean and standard deviation are learnt from the encoder and a sample is generated from a Gaussian distribution with the trained mean and standard deviation. This sample is passed through a decoder with 3 layers in order to generate the reconstructed images.

For MNIST, the image is given as an array of 784 values (28*28) and 256 values (16*16) for USPS. The size of hidden layers in encoder is 512 and 256 for MNIST and 128 and 64 for USPS. The size of hidden layers in decoder is 256 and 512 for MNIST and 64 and 128 for USPS.

The architecture that we have finalized for Conditional Variational Autoencoder is as follows. First there is an encoder with 2 fully connected layers. 2 parameters - mean and standard deviation are learnt from the encoder with the feature size increased to size of image array plus the one hot encoded label for a particular class. A sample is generated from a Gaussian distribution with the trained mean and standard deviation. This sample is passed through a decoder with 2 layers in order to generate the reconstructed images.

For MNIST, the input is given as an array of 784 values (28*28) plus 10 (one hot encoding) and 256 values (16*16) plus 10 for USPS. The size of hidden layer in encoder and decoder is 400 for MNIST and USPS.

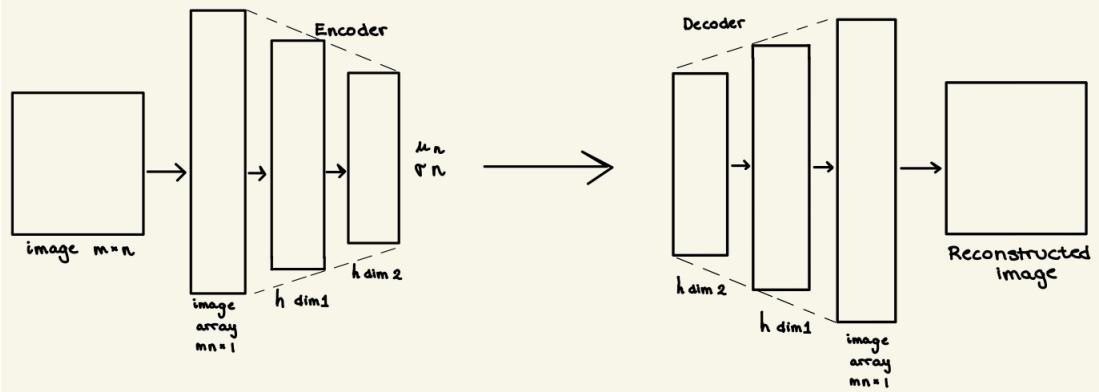

Figure 1: Visualisation of the VAE architecture

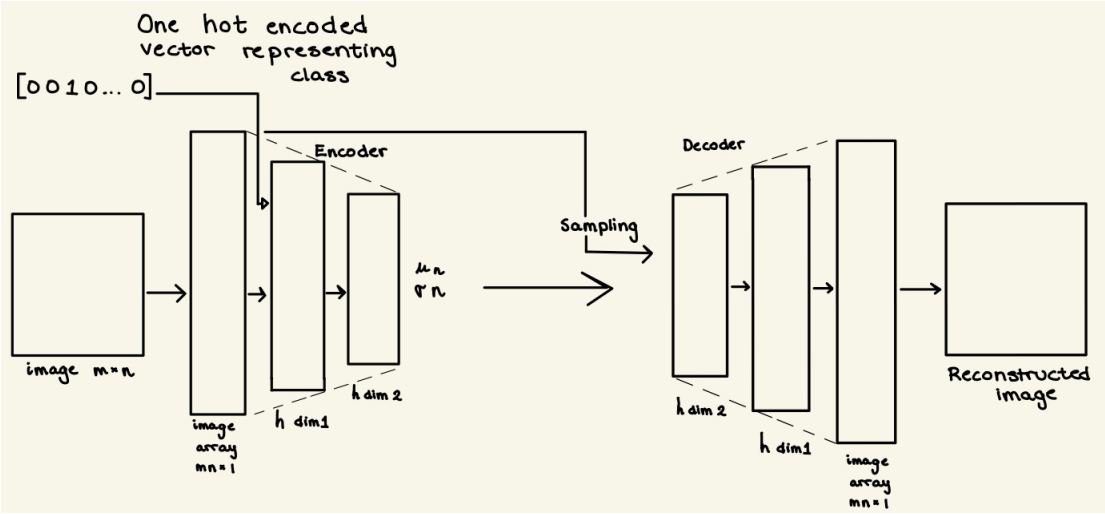

Figure 2: Visualisation of the CVAE architecture

4 Experiments

As in the original paper, the variational autoencoders have been implemented on MNIST dataset, we have also performed the analysis and modelling on 2 datasets - MNIST and USPS.

Datasets -

1. MNIST - This dataset contains 60,000 28*28 greyscale images of hand written digits from 0 to 9.

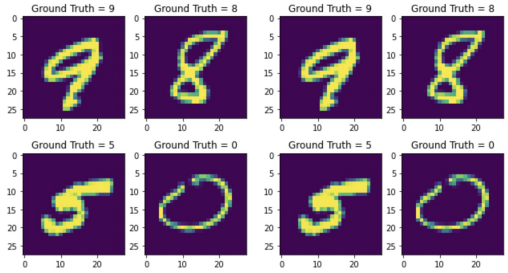

Figure 3: Data Visualization for MNIST

2. USPS - This dataset contains 9,298 16*16 greyscale images of hand written digits from 0 to 9 scanned automatically from envelopes by U.S. Postal Service. These images are normalized, centered and have a wide range of font styles.

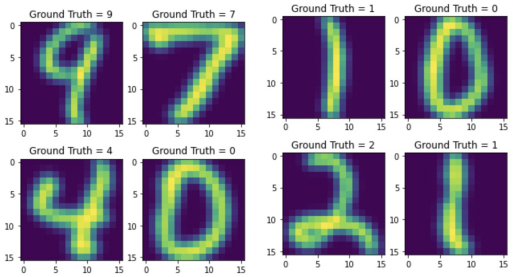

Figure 4: Data Visualization for USPS

The following parameters had been trained -

1. Batch Size
2. Learning Rate
3. Size of Latent Space
4. Optimizer

Though we had tried several different values for the above hyperparameters, we include only some of the plots for batch sizes and learning rate.

For both USPS and MNIST, the best batch size and learning rate came out to be 256 and 0.001. Also the optimizer that gave the minimum loss was Adam amongst RMSProp, Adagrad and Adam. These values and plots are for the Conditional Variational Autoencoder that we trained for both the datasets.

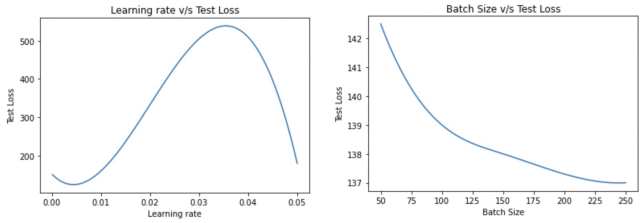

Figure 5: Fine tuning for Learning Rate and Batch Size for MNIST

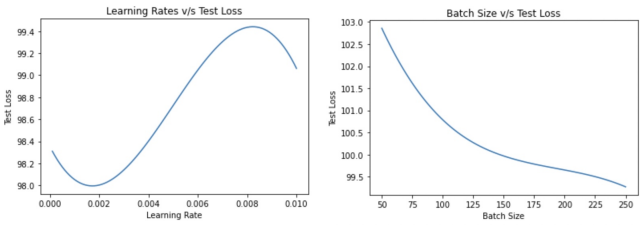

Figure 6: Fine tuning for Learning Rate and Batch Size for USPS

We also visualized the latent space for the Conditional Variational Autoencoder for MNIST and USPS. The distributions learned for each class (0 - 9) are visualized using different colour colour schemes. It can be seen that these distributions have different parameters (mean and standard deviation) for learned for each class.

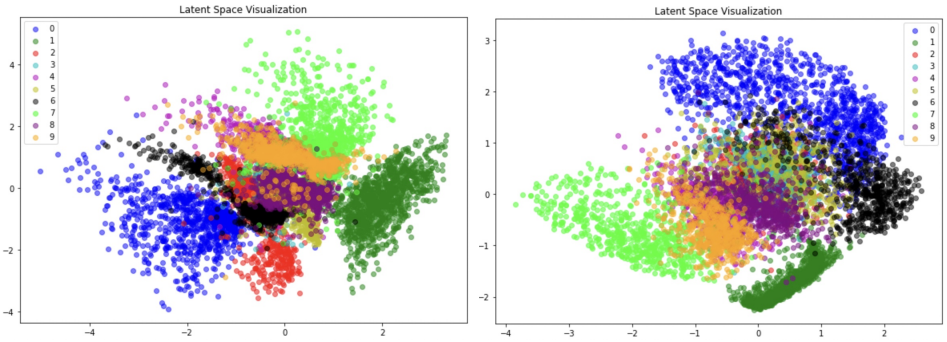

Figure 7: Latent Space Visualized for MNIST (left) and USPS (right)

We also kept track of the reconstructed images while training our CVAE models after each epoch. Here is an example showing the contrast of a the reconstructed images after the first epoch and the last epoch.

1. MNIST -

Figure 8: Reconstructed Images after Epoch 1(left) and Epoch 10 (right)

2. USPS -

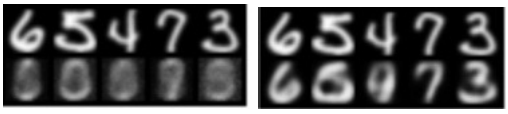

Figure 9: Reconstructed Images after Epoch 1(left) and Epoch 10 (right)

Moreover, after we trained our models, and fine tuned our models to achieve the best hyperparameters, we saved the model weights corresponding to these hyperparameters. We then used the same to reconstruct some of the images for different classes using the trained latent space and the model decoder. The following are the reconstructed images for MNIST and USPS for both Variational Autoencoder and Conditional VAE.

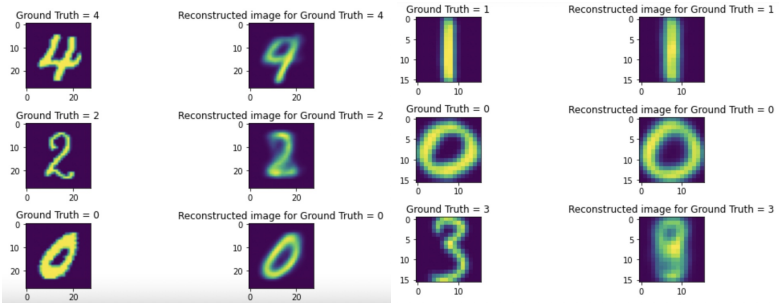

Figure 10: Reconstructed Images for VAE for MNIST (left) and USPS (right)

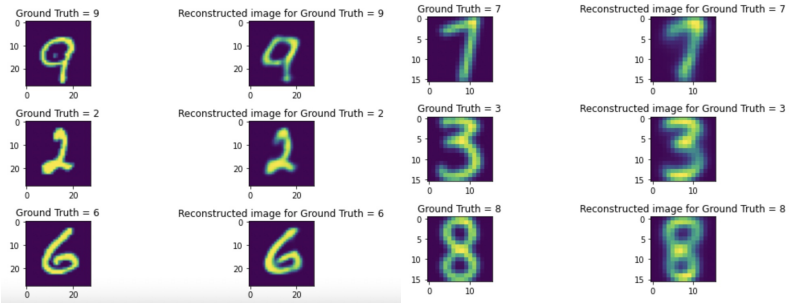

Figure 11: Reconstructed Images for CVAE for MNIST (left) and USPS (right)

5 Contribution of Team Members

Priyansh Bhatnagar -
1. Developed the VAE class including encoder and decoder
2. Implemented Conditional Variational Autoencoder and visualization of reconstructed images
3. Hyperparameter Tuning - a. Batch Size b. Learning Rate
4. Contribution in report - Method and Experiments

Kaja Erfjord -
1. Latent Space Visualization of VAE and CVAE for both datasets MNIST and USPS
2. Hyperparameter tuning - a. Dimensions of Latent Space b. Number of Epochs
3. Implemented different architectures for Variational Autoencoders
4. Contribution in report - Introduction, Related Work, Model Architecture

6 Supplementary Material

We have also prepared a Presentation for our project. Below is the link for the presentation -
Project Presentation

We have also prepared a video for our project. Below is the link for the video -
Video Presentation/Demo

References

[1] Doersch, C. (2016). Tutorial on Variational Autoencoders. Revised on 3 Jan 2021.

[2] Kingma, D. P., Welling, M. (2013). Auto-Encoding Variational Bayes. Revised on 10 Dec 2022.

[3] Kulkarni, T.D., Whitney, W., Kohli, P. and Tenenbaum, J.B. (2015). Deep Convolutional Inverse Graphics Network.

[4] Mnih, A., Rezende, D. J. (2016). Variational inference for Monte Carlo objectives. Revised 1 Jun 2016.

[5] Sohn, K., Lee, H., Yan, X. (2015). Learning Structured Output Representation using Deep Conditional Generative Models.

[6] Md Ashiqur Rahman. (2020). Understanding Conditional Variational Autoencoders. Towards Data Science. https://towardsdatascience.com/understanding-conditional-variational-autoencoders-cd62b4f57bf8

[7] Asperti, Andrea, and Matteo Trentin. (2020). Balancing reconstruction error and kullback-leibler divergence in variational autoencoders. IEEE Access 8: 199440-199448.