VAE – GAN: Identifying VAE-GAN Model for Latent Representation Learning

Sruthi Mandalapu

Abstract - As we know the unsupervised learning often requires the features to be trained, coming to VAE-GAN which is one of unsupervised learning, which requires the model to be trained to obtain the realistic and good pixel quality data. The observation we get to notice is, the dataset need to be more trained which often consumes huge time to get stabilized and to generate prolific quality images. Inducing a regularization technique actually reduces the test set loss and also model gets trained quickly to get clutched to a stabilization procuring a adequate quality data as outcome. This process is introduced to denoise the given noisy dataset, generating to achieve impactful results quickly by removing the noise to extract good pixel oriented output.

Keywords - VAE, GAN, Reconstruction Loss, Overfitting, Regularization, Batchnormalization, Denoising, Latent Representation

I Introduction

As we all know supervised learning use an identified data i.e. a classified dataset. This dataset is used to predict the classification of other unnamed data using machine learning algorithms. In other words, the data is already trained, which can be used to predict the unlabeled data (can also be referred as test set). [1] The main process involved in this learning is labeled data is given to the machine which is followed by a Machine Learning model and then allowed to analyze the dataset based on techniques like classification, regression, or clustering. This technique actually helps out the test set to predict almost accurately. [2] what if unlabeled data is given to the machine, this is where unsupervised learning comes into picture. One such aspect that can be observed through this learning is, as it is unlabeled or unspecified data it actually takes the Machine or ML algorithm to be trained and trained continuously in order to achieve accurate results. So, this process involves inputting a raw data and it is given for interpretation which is followed by machine learning algorithm to process the output making sure to put out labeled data.

So, as mentioned above we have seen unsupervised learning is a process the machine needs to be trained continuously to analyze the data, this often consumes time for the machine. In other words, we find this approach of learning, involves a machine to be trained for analyzing and providing out the distributed clustered data sets, which is actually becomes strenuous. Instead, the process of latent representation where input data is given to deep learning model where features are extracted which is represented in latent space which achieves classification, regression and reconstruction in an easier way. It is an abstract of multi-dimensional space which allows to encode in a form of meaningful internal representation. [3] The similar features of the samples are positioned close to each other on latent space. So, here we can demonstrate the latent space as to provide the compressed understanding of the world to the computer, which is projected in a spatial representation.

1.1 Existing System

The same concept of latent space is introduced to Autoencoders. Here, the Autoencoders requires latent representation which allows in simplifying the data to clustered information based on statistical inference. The distribution of data is actually taken care by a probabilistic modelling to partition each cluster across the space. This process makes easier to get the trained data features to give input for generation of new outcome. The Autoencoders actually encodes allows in encoding the data from huge dimension to put into a smaller dimension than the original format. Then, the decoder maps into the lossy version of data to get back to original high dimensional output. So, the main feature what actually it incorporates is while decoding as the input is smaller version, the decoder perfectly identifies the important features and extracts them to produce out enhancable output format. We can find in real scenarios where auto encoders used in removal of water marks of the images, also helps in denoising of picture, where it produces an improved image by removing noise. Few other features which are implemented through the Auto Encoders is neural impainting, this actually helps out in removal of features in background to look more profounded image and also allows to add missed streams from the existing image.

The extended version of Autoencoder is Variational Autoencoders which is termed shortly as VAE. Variational Autoencoders allows to explain: Instead of mapping to fixed vector, you want to map the inputs to distribution, the difference is the bottle neck vector Z is replaced by two separate vectors one representing mean of the distribution and the other one representing standard deviation of the distribution. This VAE is merged with GAN which is Generative Adversarial Network to pull out the real image with high dimensional pixel quality. In GAN, the generator learns to produce reliable data. The created manifestations become negative educational examples for

the discriminator. The discriminator learns to distinguish how much accurate the data is. The discriminator penalizes the generator for improbable results. At the beginning of training, the generator produces obviously false data, and the discriminator quickly learns that it is fake: As the training progresses, the generator approaches to produce power that can fool the discriminator. Finally, if generative learning goes well, the discriminator gets worse at distinguishing between genuine and fake. It starts classifying fake data as real and its accuracy decreases.

1.2 Problem with Existing System

VAE - GAN often suffers from overfitting, which actually performs training of data very poorly. Introducing regularization helps out to prevent overfitting to produce out simpler model. This would be helpful in controlling the complexities of representations. Through this we can achieve semi supervised learning where features can be contemplated easily during distribution in the latent space. So, here the main motivation involves in enhancements in latent space to control. This enables in high pixel quality data reconstructions.

1.3 Proposed Solution

The attempt to eliminate the overfitting problem is by using Regularization technique, where Batch-Normalization is used which is one of Regularization technique, allows almost to equalize the weights helps in learning the data easily. This is introduced to minimize the execution time for predicting a stabilized outcome with high quality pixels data in minimal number of trainings. As quality of image and reconstruction loss is correlated, hence as the quality of images increases, the loss of image indefinitely gets reduced to get the balanced real pixel quality image within shorter time.

1.4 Objectives and Goals

Advantage of Integrating VAE with GAN to avoid cons among them: The main objectives of VAE-GAN are to take advantage of the strengths of both models while addressing some shortcomings of conventional VAEs and GANs. VAE-GANs strive to combine the properties of both VAEs and GANs in order to enhance generative modelling, enhance latent space representations, and gain more control over the generation process while addressing some of the shortcomings of each specific model. Due to these goals, [2] VAE-GANs are a versatile solution for a variety of machine learning applications, such as picture creation, data augmentation, and anomaly detection.

Ability to generate new data samples that closely resemble the training data: GAN component often has a main focal point where a generator network allows to generate a plausible data. The generated instances actually become negative training samples for the discriminator. And successively discriminator penalizes to provide out the generator with an implausible result. These samples would be helpful continuously in a way having a well-trained data for generator.

Reconstructing High Quality Data by Improve of latent space representation: Regularization techniques often allows to avoid a problem of overfitting which gives out more trained data representation in latent space, as it is simpler way of representation the process goes out easier to generate new data, which helps in improving pixel quality. As improved latent space representation is observed, this also helps in the data to be trained easily and reduces the amount of time to predict accurate data.

1.5 Expected Challenges

Instability during training: Achieving a balance between VAE and GAN components can be difficult, and it is important to find the right hyperparameters such as learning rate and weights to stabilize the training. Training VAE-GANs can be computationally intensive, especially for large datasets and complex architectures. Availability of powerful hardware or cloud resources may be limited.

Avoid Cons of Overfitting Problem: Weight normalization (weight reduction) of models including both encoder and decoder. Regularization favors smaller weight values and can prevent the model from fitting the training data too closely. Here, Models that are too complex, with too many parameters, are prone to overfitting. To avoid this, consider reducing the size of the hidden space, reducing the number of layers and neurons in the networks or using regularization techniques such as dropping or weighting.

Quality of Data: VAE-GANs aim to produce data that closely resemble real-world examples and introduce variations that reflect the diversity and richness of the dataset. This quality is achieved through a competitive training process where the generator competes with the discriminator to produce data indistinguishable from the real data. The quality of the data produced by VAE-GAN can also be affected by the quality of the input data and the effectiveness of the data pre-processing techniques. The data quality, in VAE GAN refers to the characteristics of the dataset that is used to train the model. It's crucial for the data to be accurate, consistent and free from any errors or noise. The datasets relevance to the task, such as using face images for face generation plays a role. While larger datasets can improve the model's robustness, they should still maintain quality. Preprocessing tasks such as resizing, normalization and augmentation also contribute to maintaining data quality. Ultimately the quality of data significantly impacts how well the model learns representations and generates samples.

II Related Work

This section explains introducing VAE with GAN. VAE-GAN combines the concepts of VAE and GAN in an

attempt to leverage the benefits of both models. VAE-GANs have been applied in various domains, including image generation and they are often used when a balance between structured latent representations and high-quality data generation is required. The proposed hybrid feature avoiding overfitting combining VAE with GAN and is motivated to extract feature wise metric for measuring reconstruction quality during training.

2.1 Autoencoder

Autoencoder leverages the benefit of neural networks that allows for a task in representation learning. The design of neural network architecture is to impose a bottleneck with in the network which exerts knowledge representation of original input data set in a compressed way. If the features of input data set are independent of each other this subsequent reconstruction is a very strenuous task. Regardless of how, if some structure exists in the data (that is correlations among input features), this structure can be learnt and consequently leveraged when forcing the input through the bottleneck network's.

Taking an unlabled data set considering it as a supervised learning results in output as $\hat{x}$ which is a reconstruction for the original input x. This network is allowed to train reducing the loss of reconstruction can be termed as reconstruction error representing as: $L(x, \hat{x})$ measuring the amount of loss across the original input and consequent reconstruction. The network then can easily learn to simply remember the input values by passing those values through the network referring to train it again. The ideal autoencoder balances the following:

- Sensitive enough for the inputs to build accurately for reconstruction
- Insensitive enough for the inputs that the model couldn't able to memorize or overfit of trained data.

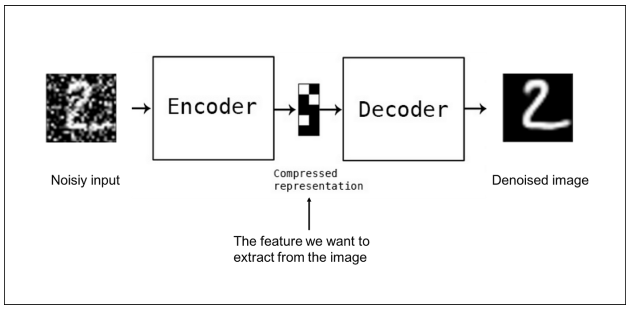

Figure 1: Autoencoder

This trade-off forces the model to preserve only the variations in the data necessary to reconstruct the input, without preserving the redundancy of the input. In most cases, this requires the construction of a loss function where a single expression prompts our model to be sensitive to the inputs (ie, the reconstruction loss $L(x, \hat{x})$) and the second term prevents memorization/overfitting (i.e., the added regularizer). The expression is termed as below:

$$L(x, \hat{x}) + regularizer$$

By penalizing the network according to the reconstruction error, our model can learn the most important attributes of the input data and how best to reconstruct the original input encoded status ideally, this encoding learns and describes the latent attributes of the input data.

2.2 Variational Autoencoder

Instead of outputting the latent state values directly, the encoder model outputs parameters that describe the distribution of each dimension in the latent space. Because we are assuming that our prior follows a normal distribution, we will output two vectors describing the mean and variance of the latent spatial distribution. With this approach, we can now describe the latent attribute of each given input as a probability distribution. When decoding a latent state, we randomly select the distribution of each latent state distribution to create a vector as input to our decoder model. In variational autoencoders, the encoder is sometimes called the recognition model, while the decoder model is sometimes called the generative model.By building our encoder model to produce a set of possible values (a statistical distribution) that are randomly selected to feed our decoder model, we're essentially enforcing a continuous uniform latent state representation. For all samples of the hidden distribution, we assume that our decoder model can accurately reconstruct the input. Thus, values close to each other in the hidden space should correspond to very similar reconstructions.

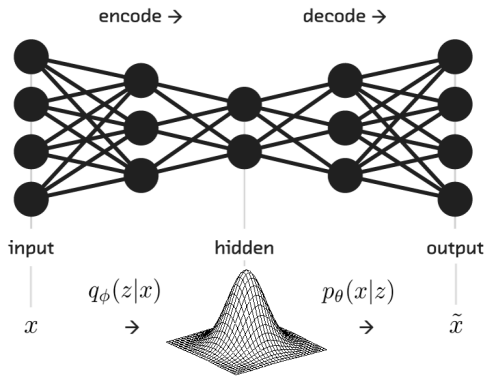

Figure 2: Variational Autoencoder

2.2.1 Statistical Motivation for VAE

Assume that there is a hidden variable z that produces the observation x. We only see x, but we want to infer the characteristics of z. [4] To put it another way, we want to compute $p(z|x)$ as follows:

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

Computing $p(x)$, we get: $p(x) = \int p(x|z)p(z)dz$

Approximating $p(z|x)$ by another distribution $q(z|x)$. If we could define the parameters for $q(z|x)$, so that it is very

similar to $p(z|x)$. KL-divergence is a measure of the difference between two probability distributions. Here, we ensure that $q(z|x)$ is similar to $p(z|x)$, so we get $KL(q(z|x)||p(z|x))$

The loss we get is $E(q(z|x)log(p(x|z)))+KL(q(z|x)||p(z))$

The reconstruction probability is represented by the first component, and the similarity between the learnt distribution q and the true prior distribution p is guaranteed by the second term. For this network, we have a two-term loss function: one that maximizes the reconstruction likelihood and penalizes the reconstruction error, and another that promotes our learned probabilistic distribution $q(z|x)$ to resemble the true prior to the distribution $p(z)$. Thus, the loss that we specify for VAE is:

$$L_{VAE} = L(x, \hat{x}) + \Sigma_j(KL(q_j(z|x)||p(z)))$$

$L(x, \hat{x})$ defines the binary cross entropy of x and $\hat{x}$, Entropy is given as $q(z|x)log(p(x|z)$, binary cross entropy is given by:

$L_{recon} = q(z|x)log(p(x|z) + (1 - q(z|x))log(1 - p(x|z))$

2.3 Generative Adversarial Network

Generative adversarial networks, or GANs, composed of two neural networks: one that creates the data and another that attempts to categorize it. With the generator attempting to trick the discriminator and the discriminator attempting to accurately classify the data, the two networks competes with one another. Generating data that is realistic enough to trick the discriminator is the aim of a GAN.

Training the generator: Compared to separation learning, generative learning necessitates a closer integration between the generator and separator. The GAN's generator-training component consists of the following: Random entry, a network of generators that converts random input into instances of data, a discriminator network that categorizes the output of the data separator that is generated and the generator is penalized for deceiving the separator by means of generator loss.

Two loss functions are connected to the discriminator. The discriminator only employs the discriminator loss during discriminator training, ignoring the generator loss. We employ the generator loss during generator training. While receiving discriminator training: The discriminator uses the generator's data to classify both accuracy and inaccuracy of data. When a real instance is mistakenly identified as fake or a fake instance as real, the discriminator suffers a loss. Using backpropagation from the discriminator loss via the discriminator network, the discriminator modifies its weights.

2.3.1 Discriminator Loss

The goal of the discriminator is to distinguish between accurate an inaccurate data. [4] Here let's denote:

- $D(x)$: The output of the discriminator for real data x.
- $D(G(z))$: The output of the discriminator for generated (fake) data $G(z)$, where G is the generator and z is a random noise vector.

Now, let's use the following labels:

- $real\_labels = 1$: These labels indicate real data.
- $fake\_labels = 0$: These labels indicate fake (generated) data.

The discriminator loss for real data is calculated using binary cross-entropy loss as follows:

$$real\_loss = BCELoss(D(x), real\_labels)$$

This term measures how well the discriminator correctly classifies real data. The BCELoss function is defined as:

$$BCELoss(y, t) = -(tlog(y) + (1 - t)log(1 - y))$$

where y is the predicted probability (output of the discriminator) and t is the target label (1 for real data).

Similarly, the discriminator loss for fake data is calculated as:

$$fake\_loss = BCELoss(D(G(z)), fake\_labels)$$

This term measures how well the discriminator correctly classifies generated (fake) data.

The total discriminator loss is the sum of the losses for real and fake data:

$$total\_discriminator\_loss = real\_loss + fake\_loss$$

This combined loss is used in the backward pass and optimization steps to update the parameters of the discriminator, making it better at distinguishing between real and fake data.

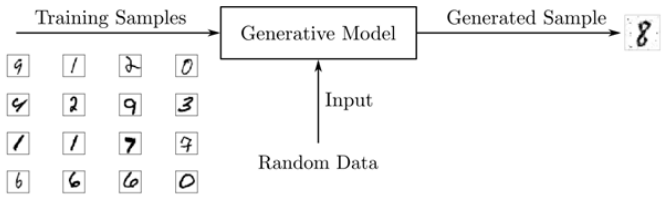

Figure 3: Generative Adversarial Network

III Methodology

This section introduces the model specification of how VAE-GAN is implemented, in addition to also explains Regularization technique to avoid the difficulty of over-fitting. Here, the represenatative architechture is projected maintaining a trace of steps followed during training of dataset.

3.1 Implementation

Use a DataLoader to set up data loading step. Here the data set which was implemented in VAE-GAN is MNIST.

Then the next step is followed by the components of the encoder, decoder, and discriminator for the VAEGAN. Based on the input batch size we select the proper normalization layers, activation functions. Typically here, binary cross-entropy loss for image data is used to define the reconstruction loss for the VAE portion, also the VAE loss is defined by KL divergence considering the average loss. BCE loss is used for the discriminator when defining the adversarial loss for the GAN portion. To sample from the latent space in the VAE, apply the reparameterization approach. The next step is followed to train the data, so setting up a training loop to be iterated over the dataset . For each iteration, pass data through the model and compute the combined loss. And then perform backpropagation and update the model parameters. Select optimizers (such as Adam) for the GAN and VAE components. Specify different optimizers for the GAN and VAE parts. After training, evaluate the model on a test set. Visualize reconstructed samples and generated samples.

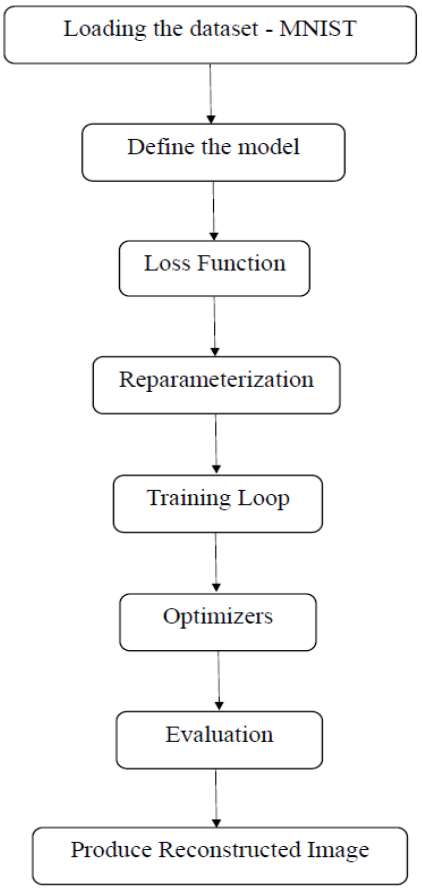

Figure 4: Steps involved in VAE-GAN

3.2 Avoidance of difficulty in Overfitting

When a model becomes very adept at learning the training set, identifying noise and random fluctuations in the dataset, instead of the underlying patterns. In other words, the model does not generalize effectively to new, unknown data and instead becomes too particular to the training dataset. This is when overfitting arises. To overcome the problem we regularize the data i.e we apply a mechanism of regularization. Batch Normalization is one of regularization technique, the basic notion of batch normalization is to divide the batch standard deviation by the batch mean and subtract it from the layer inputs of a neural network. During training, each mini-batch is subjected to this. Before being fed into the following layer, the normalized inputs are first scaled and shifted by learnable parameters. This helps to maintain the distribution of the inputs throughout the training process.

Advantages: Accuracy and better performance: batch normalization aids in bringing each layer's inputs into compliance, which can quicken training and improve convergence. This can enhance the model's overall performance and accuracy., Faster Convergence: By reducing problems like vanishing/exploding gradients, normalizing inputs enables the model to converge more quickly while being trained., Standardization of layer: By preserving a constant mean and variance for every feature, batch normalization serves as a layer of standardization. This may facilitate the model's ability to recognize trends in several batches, leading to more consistent training., Good transformation of values: effective transformation can be acheived through batch normalization which modifies inputs in a way that facilitating model learning. A standardized distribution makes it less likely for the network to become stuck during training., Model can easily deluge the data set: adaptive learning rates is providing inputs in a way that makes it easier for the model to learn

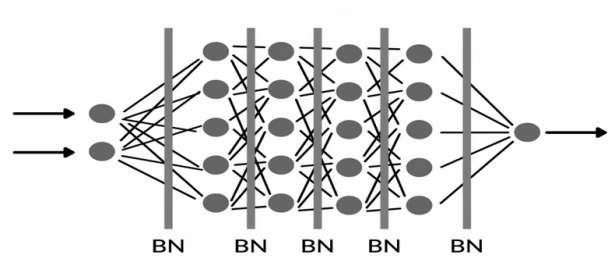

Figure 5: Usage of Batch-Normalization

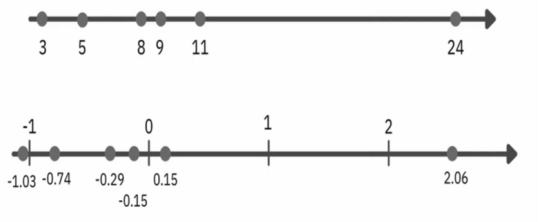

Figure 6: Normalizing the weights

3.3 Pseudocode of VAE-GAN

The below is the pseudocode for VAEGAN:

class VAEGAN { __**init**__(self): {
Initialize encoder:
- Linear layer with input size 784 and output size d^2
- Batch normalization
- ReLU activation
- Linear layer with input size d^2 and output size 2d
- Batch normalization

Initialize decoder:
- Linear layer with input size d and output size d^2
- Batch normalization
- ReLU activation
- Linear layer with input size d^2 and output size 784
- Sigmoid activation

Initialize discriminator:
- Linear layer with input size 784 and output size 256
- Batch normalization
- ReLU activation
- Linear layer with input size 256 and output size 1
- Sigmoid activation

}
void reparametrize(y): {
  if training:
    compute std = sqrt(exp(logvar) / 2)
    generate eps from a normal distribution
    return reparameterized
    sample: eps * std + mu
  else:
    return mu

}
void forward(y): {
  Pass input y through the encoder
  Extract mu and logvar from the encoder output
  Reparameterize to get the latent variable z
  Pass z through the decoder to reconstruct the input
  Return reconstructed input, mu, and logvar

}
void discriminate(x) {
  Pass input x through the discriminator
  Return discriminator output

}
}

Algorithm 1: Pseudocode of VAEGAN

```
class VAEGAN(nn.Module):
    def __init__(self):
        super().__init__()
        # Encoder
        self.encoder = nn.Sequential(
            nn.Linear(784, d**2),
            nn.BatchNorm1d(d**2),
            nn.ReLU(),
            nn.Linear(d**2, d*2),
            nn.BatchNorm1d(d*2)
        )
        # Decoder
        self.decoder = nn.Sequential(
            nn.Linear(d,d**2),
            nn.BatchNorm1d(d**2),
            nn.ReLU(),
            nn.Linear(d**2, 784),
            nn.Sigmoid()
        )
        # Discriminator
        self.discriminator = nn.Sequential(
            nn.Linear(784, 256),
            nn.BatchNorm1d(256),
            nn.ReLU(),
            nn.Linear(256, 1),
            nn.Sigmoid()
        )
    def reparameterise(self, mu, logvar):
        if self.training:
            std=logvar.mul(0.5).exp_()
            eps=std.data.new(std.size()).normal_()
            return eps.mul(std).add_(mu)
        else:
            return mu
    def forward(self,y):
        mu_logvar=self.encoder(y.view(-1,784)).view(-1,2,d)
        mu=mu_logvar[:,0,:]
        logvar=mu_logvar[:,1,:]
        z=self.reparameterise(mu, logvar)
        return self.decoder(z),mu,logvar
    def discriminate(self, x):
        return self.discriminator(x.view(-1, 784))
```

Figure 7: VAE-GAN Algorithm

3.4 Tools and Technologies Used

Here, the main tools and technologies used in VAE-GAN are: Python a primary tool which is used to create machine learning models such as VAE-GAN, here we does also use Jupyter Notebook as an interactive programming environment that lets to make and distribute the code for representations, and graphics. The technologies implemented in VAE-GAN are torchvision in PyTorch, this module is used to manage computer vision tasks. The MNIST dataset, which is frequently used for digit recognition, is loaded and transformed within the code. Here, nn from torch an essential part for creating neural network design. It offers a range of tools, loss functions, and layers for building and refining neural networks. The encoder, decoder, and discriminator of a custom neural network components that you define in the code is by using the nn.Module which is a base class of all neural networks classes. DataLoader is used to generate an wide range of varied datasets, by using the library torch.utils.data. During training, it makes possible for you to load and loop through batches of data quickly. Few

other modules are: Transformations (from torchvision), this module offers preprocessing and augmentation functions for images. VAE-GAN uses it to create a set of modifications that the input photos from the MNIST dataset will undergo. The MNIST dataset is loaded using this class (from torchvision.datasets). You will create an MNIST dataset object in the code by giving the train mode, the appropriate transformations, and the root directory where the dataset will be stored. The pyplot module (from matplotlib) is used to plot photos and create other types of visualizations. To view the produced pictures or any other pertinent visualizations during or after training, import the code.

3.5 Architecture

We find a compact representation of encoder, a decoder, and a discriminator, each consisting of two fully connected layers. Regularization such as batch normalization, can be added between the layers helps to mitigate overfitting. BatchNorm helps stabilize and speed up training by normalizing the input to each layer, reducing internal covariate shift, and acting as a form of regularization.

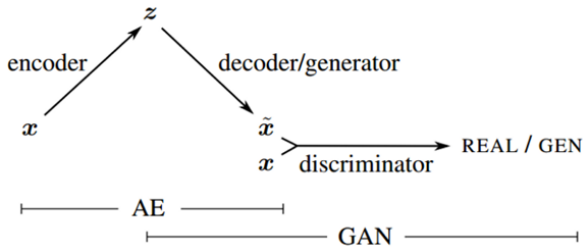

Figure 8: Architecture of VAE-GAN

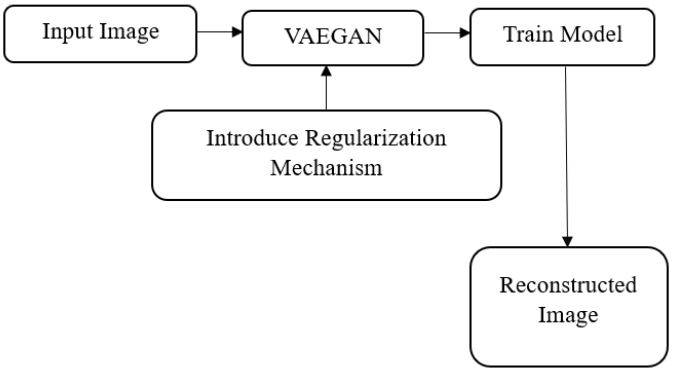

Figure 9: Introduce BatchNorm to VAE-GAN

IV Results

This section diverges in understanding the results procuring the proposed implementation of vaegan. So we find mainly concering to measure the effect of avoidance of overfitting problem using regularization (batch normalization).

4.1 Measure of Reconstructed Images

Here, we can observe the reconstructed image loss before and after regularization differs, the below is the test set 3 that is epoch 3 reconstructions.

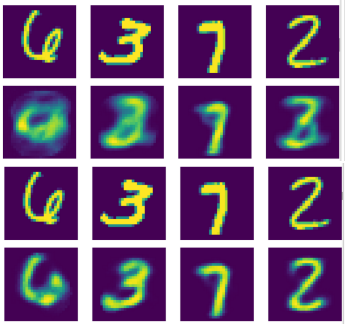

Figure 10: Reconstructions before and after Batchnorm

```
====> Epoch: 1 Average loss: 469043.53020833334
====> Test set loss:  230.3283
====> Epoch: 2 Average loss: 372118.0802083333
====> Test set loss:  180.3972
====> Epoch: 3 Average loss: 334059.9197916667
====> Test set loss:  157.0591
====> Epoch: 4 Average loss: 305111.078125
====> Test set loss:  142.4545
====> Epoch: 5 Average loss: 286867.453125
====> Test set loss:  134.2129
====> Epoch: 6 Average loss: 274090.73125
====> Test set loss:  128.9089
====> Epoch: 7 Average loss: 266304.3421875
====> Test set loss:  126.7280
====> Epoch: 8 Average loss: 261820.54010416666
====> Test set loss:  124.5679
====> Epoch: 9 Average loss: 257664.67864583334
====> Test set loss:  122.2477
====> Epoch: 10 Average loss: 253247.05572916666
====> Test set loss:  119.8634
```

```
====> Epoch: 1 Average loss: 399735.6302083333
====> Test set loss:  147.8428
====> Epoch: 2 Average loss: 296214.32916666666
====> Test set loss:  125.5780
====> Epoch: 3 Average loss: 269985.890625
====> Test set loss:  114.9232
====> Epoch: 4 Average loss: 253746.15989583332
====> Test set loss:  107.7456
====> Epoch: 5 Average loss: 241967.37864583332
====> Test set loss:  103.7279
====> Epoch: 6 Average loss: 233630.771875
====> Test set loss:  101.1813
====> Epoch: 7 Average loss: 227900.6109375
====> Test set loss:  99.6840
====> Epoch: 8 Average loss: 223998.0765625
====> Test set loss:  98.5898
====> Epoch: 9 Average loss: 221302.01145833332
====> Test set loss:  98.2488
====> Epoch: 10 Average loss: 218911.9984375
====> Test set loss:  97.3294
```

Figure 11: Before and After implementing Regularization

4.2 Measure of Denoised Images

Here, we can find the denoised image losses before and after implementing the Batch Normalization. The denoised images for the 1st epoch has a greater difference, if we don't implement regularization it could unable to predict at initial trainings, this results in huge time of execution as

number of training's increases. When compared over the test set that is regularized at initial epoch it is actually predicting the data. Below is the comparison of epoch1 results for denoising of image. Here the dimensionality of neural network is 100 (putting in lower dimensionality).

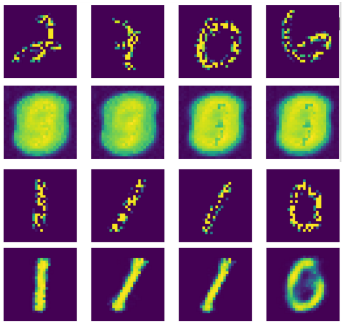

Figure 12: Denoising before and after Batchnorm

```
====> Epoch: 1 Average loss: 402239.553125
====> Test set loss:  347.0229
====> Epoch: 2 Average loss: 379431.6458333333
====> Test set loss:  247.7857
====> Epoch: 3 Average loss: 349705.42604166665
====> Test set loss:  207.1779
====> Epoch: 4 Average loss: 328458.85520833335
====> Test set loss:  186.8587
====> Epoch: 5 Average loss: 314208.3697916667
====> Test set loss:  178.0751
====> Epoch: 6 Average loss: 303869.925
====> Test set loss:  168.4420
====> Epoch: 7 Average loss: 296035.18333333335
====> Test set loss:  162.5893
====> Epoch: 8 Average loss: 290033.70625
====> Test set loss:  155.9546
====> Epoch: 9 Average loss: 284849.74375
====> Test set loss:  150.0192
====> Epoch: 10 Average loss: 280185.8104166667
====> Test set loss:  146.4455
====> Epoch: 1 Average loss: 373148.0833333333
====> Test set loss:  164.1581
====> Epoch: 2 Average loss: 341214.85104166664
====> Test set loss:  154.7473
====> Epoch: 3 Average loss: 324038.1041666667
====> Test set loss:  154.7070
====> Epoch: 4 Average loss: 312138.60520833335
====> Test set loss:  152.6874
====> Epoch: 5 Average loss: 303007.6510416667
====> Test set loss:  150.3859
====> Epoch: 6 Average loss: 295980.44375
====> Test set loss:  144.7938
====> Epoch: 7 Average loss: 290391.465625
====> Test set loss:  133.7135
====> Epoch: 8 Average loss: 285511.27291666664
====> Test set loss:  132.9777
====> Epoch: 9 Average loss: 281577.54375
====> Test set loss:  124.2333
====> Epoch: 10 Average loss: 278828.403125
====> Test set loss:  121.4781
```

Figure 13: Denoising loss before and after Batchnorm

4.3 Denoised Images over Higher Dimensionality

Here you can observe on a increased dimentionality over 500 neural networks, this effect of denoising can be noticed, by generating noise cancelled images at initial epochs, as the training iterates, the quality of images tends to look more prolific.

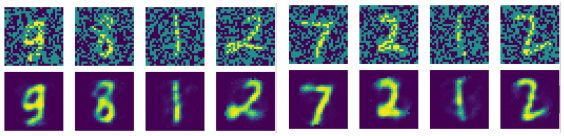

Figure 14: Denoising Images at higher dimension

4.4 Latent Space Analysis

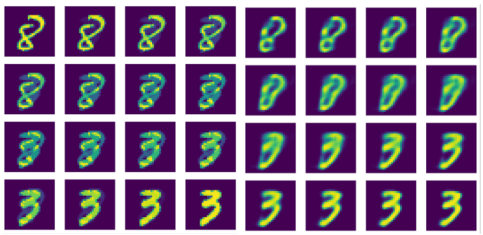

Figure 15: Mapping of 2 number in latent space

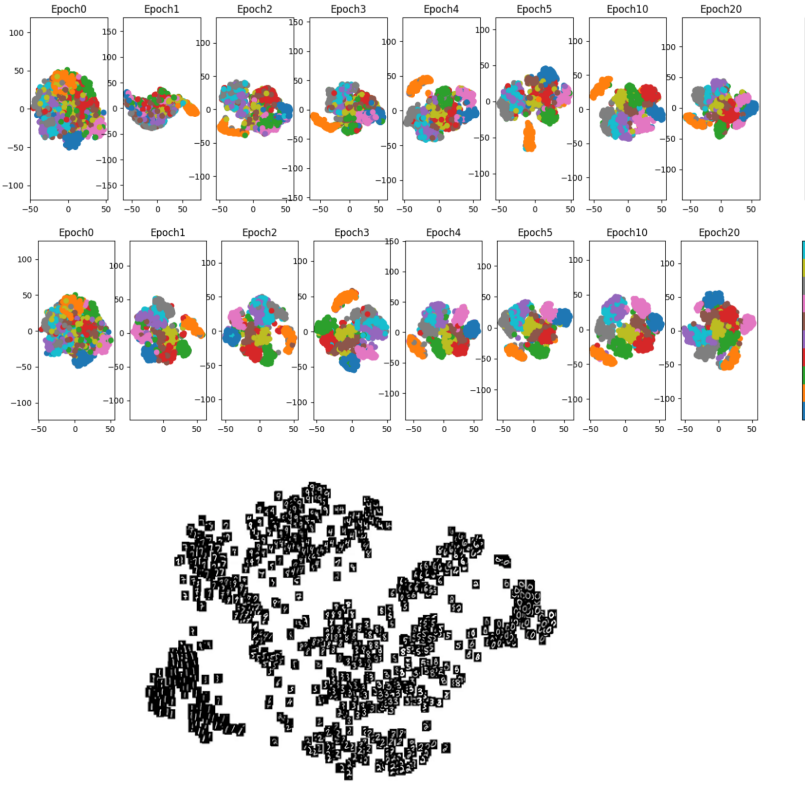

Figure 16: Latent Space Representation

The upper image specifies beefore and after Regularization (Batch-Norm), on capture of data in a 2- dimensional latent space. The bottom one specifies the representative data on a dimensional space.

4.5 Analysis

As you can observe the test set results at few epcohs before and after application of Regularization technique which is projected in a graphical view. We can clearly

obesrve, after regularizing the data obsviously having a lesser test set loss which is intended as high quality pixel images when compared to normally trained data. The same scenario is observed on average loss, for a normal training reflecting the dataset need to be more trained. Next we introduce denoised data projecting the simillar comparison as above.

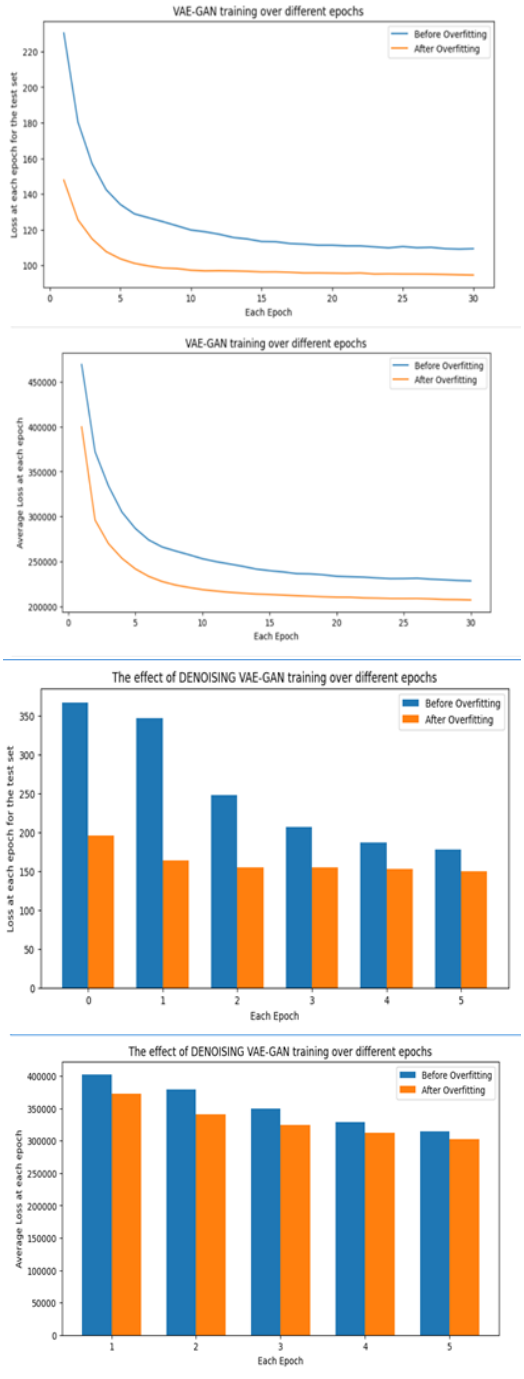

Figure 17: Comparision of Overfitting

4.6 Comparison of Results

Epochs	Normal Training	Regularizing Data
1	230.3283	147.8428
2	180.3972	125.5780
3	157.0591	114.9232
4	142.4545	107.7456
5	134.2129	103.7279

Table 1: Comparison of test set loss

Epochs	Normal Training	Regularizing Data
1	469043.5302	399735.6302
2	372118.0802	296214.3291
3	334059.9197	269985.8906
4	305111.0781	253746.1598
5	286867.4531	241967.3786

Table 2: Comparison of average loss

Epochs	Normal Training	Regularizing Data
1	347.0229	164.1581
2	247.7857	154.7473
3	207.1779	154.7070
4	186.8587	152.6874
5	178.0751	150.3859s

Table 3: Comparison of denoised data test set loss

Epochs	Normal Training	Regularizing Data
1	402239.5531	373148.0833
2	379431.6458	341214.8510
3	349705.4260	324038.1041
4	328458.8552	312138.6052
5	314208.3697	303007.6510

Table 4: Comparison of denoised data average loss

V Discussion

This section discusses the significance how robust the model is used, and also cites the limitations of this model.

5.1 Significance

Improved Latent Space Representations: VAEs encourage a more interpretable and structured latent space by regularizing the distribution of latent variables. Combining this with the adversarial training of GANs can lead to a latent space that is both controllable and capable of capturing complex data distributions. Batch normalization can help enhance the generalization of the

entire network, including the encoder and decoder of the autoencoder, by normalizing inputs at each layer. A more reliable and significant latent space representation may come from this enhanced generalization. The training of the neural network as a whole, including the encoder and decoder of an autoencoder, is stabilized with the aid of batch normalization. This can avoid problems such as vanishing gradients during training, which can facilitate the autoencoder's learning of a meaningful latent representation.

Stabilized Training with faster convergence: By reducing problems like vanishing gradients, batch normalization aids in the stabilization of the training process. This lessens the possibility of divergence during training and enables the model to converge more quickly. As a result, it can take the network less training cycles to achieve an acceptable performance level. During training, batch normalization may result in speedier convergence. It contributes to a uniform activation distribution across the network by normalizing each layer's inputs. As a result, the optimization method can converge to a solution more quickly, possibly requiring fewer training rounds.

5.2 Improved Robustness

VAE-GAN includes a reconstruction loss that promotes the similarity of the generated data to the real data. The job of the discriminator is to maximize its ability to distinguish between real and synthetic data. At the same time, the Generator tries to minimize the separator's ability to do so. The adversarial component of VAE-GAN training ensures higher accuracy and authenticity of the generated samples. This iterative process refines VAE's latent spatial representation, improving its ability to produce high-quality and consistent data. While VAEs are good at learning structured and controllable latent space representations, GANs are recognized for their capacity to produce realistic, high-quality samples. Developing generative models with more precise control over the latent space by merging these models to generate realistic examples. By combining the regularization techniques of VAEs with the adversarial training of GANs, researchers aim to create models that are more robust and less prone to overfitting. This can result in improved generalization to unseen data.

5.3 Major Implications

Applicability to Image Generation: When used in conjunction with batch normalization, denoising, and vae-gan, picture generating tasks can produce realistic, high-quality images. [5] This has consequences for computer vision tasks like data augmentation, style transfer, and picture synthesis.

Potential for Anomaly Detection: The regularization effects of denoising and batch normalization could improve the model's capacity to identify and discard noisy or abnormal data. This might be useful in situations where it's important to spot abnormalities or outliers.

Adaptable Architecture Design: [6] The architecture of vae-gans can be designed with flexibility thanks to the use of batch normalization and denoising. By experimenting with different configurations, practitioners can take advantage of these techniques' advantages and attain the ideal balance between generalization and model complexity.

5.4 Limitations

Need for Extensive Training Data: Large training data sets are often beneficial for generative models, such as VAEGANs, as they enable them to capture the diversity of the underlying distribution. The model's effectiveness might be constrained in situations when gathering a sufficient amount of diverse data is difficult.

Trade-off in Latent Space Quality: The quality of the learned latent space representation and the richness of generated samples may be traded off, even if a VAE-GAN may reduce overfitting. Finding the ideal balance can be difficult, particularly when tailoring optimization for particular use cases.

Sensitivity to Hyperparameters: The balance between the VAE and GAN components, learning rates, and architectural factors are among the hyperparameters that might affect a VAEGAN's performance. Suboptimal selections may have an impact on the caliber of the generated samples and the latent space representation, so it's important to fine-tune these hyperparameters.

Mode Collapse: Mode collapse can still be an issue even though the VAE component regularizes the GAN. When a generator generates only a small number of sample types while disregarding the diversity found in the data distribution, mode collapse takes place. Getting rid of mode collapse totally is difficult.

5.5 Findings Compared with Existing Work

Incorporating [6] denoising techniques in the context of VAE-GANs is a logical step to improve the model's ability to handle noisy or imperfect data. The denoising process can contribute to better generalization by enabling the model to focus on the underlying patterns in the data rather than fitting to noise. Machine learning models frequently face the issue of overfitting, and VAE-GANs are not immune to it. To avoid overfitting, methods such as regularization, denoising, and appropriate model design are essential. In this situation, denoising could function as a regularization technique by keeping the model from fitting the noise too closely.

VI Conclusion and Future Work

Here, we demonstrate conclusion and future scope to work for enhancing Denoised VAE-GAN with improval features.

6.1 Conclusion

VAE-GAN fuses the twinned ease of benefits leveraging more prolific outcomes. Though, it gives enough advantages, it lags in taking huge time to train the data. This is because of the problem of overfitting, which allows in huge data loss. To prevent from this, we introduce Regularization technique (Batch-Norm). This technique shows enough impact reducing the test set loss and also minimizes the execution time to acheive principled quality data in lesser number of tranings. [4] The entire idea of notion is summed up for denoised images giving out most impactful productive data. To conclude this model allows to acheive good results by maximixing data quality in shorter duration.

6.2 Future Work

One crucial hyperparameter in the training of machine learning models, such as neural networks, is the learning rate. During optimization, its value dictates the step size for each iteration. Adjust the learning rate during training by implementing learning rate scheduling. Reducing the learning rate gradually can help with convergence and keep the model from going over the better results.

References

[1] Alla Abdella and Ismail Uysal. A statistical comparative study on image reconstruction and clustering with novel vae cost function. *IEEE Access*, 8:25626–25637, 2020.

[2] Zengrui Jin, Xurong Xie, Mengzhe Geng, Tianzi Wang, Shujie Hu, Jiajun Deng, Guinan Li, and Xunying Liu. Adversarial data augmentation using vae-gan for disordered speech recognition. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023.

[3] Haziq Yusoff bin Abdul Wahab and Alexei Sourin. Application of generative adversarial networks and latent space exploration in music visualisation. In *2021 International Conference on Cyberworlds (CW)*, pages 125–128, 2021.

[4] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566. PMLR, 2016.

[5] Weiming Xiang, Dong Chen, Yingbo Cui, and Shaoliang Peng. H-vae: A hybrid variational autoencoder with data augmentation in predicting crispr/cas9 off-target. In *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 550–555, 2021.

[6] Antonia Creswell and Anil Anthony Bharath. Denoising adversarial autoencoders. *IEEE transactions on neural networks and learning systems*, 30(4):968–984, 2018.

[7] Bjørn Uttrup Dideriksen, Kristoffer Derosche, and Zheng-Hua Tan. ivae-gan: Identifiable vae-gan models for latent representation learning. *IEEE Access*, 10:48405–48418, 2022.

[8] Lev Yasenko, Yaroslav Klyatchenko, and Oksana Tarasenko-Klyatchenko. Image noise reduction by denoising autoencoder. In *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, pages 351–355, 2020.

[9] Sujay Jadhav and Pooja Kulkarni. Image denoising using deep auto-encoder network for production monitoring in real-time. In *2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, pages 1–7, 2021.

[10] Taisuke Kobayashis. q-vae for disentangled representation learning and latent dynamical systems. *IEEE Robotics and Automation Letters*, 5(4):5669–5676, 2020.

[11] Jiayi Chen and Wei Song. Gan-vae: Elevate generative ineffective image through variational autoencoder. In *2022 5th International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*, pages 765–770, 2022.

[12] A Kiran and S Saravana Kumar. A comparative analysis of gan and vae based synthetic data generators for high dimensional, imbalanced tabular data. In *2023 2nd International Conference for Innovation in Technology (INOCON)*, pages 1–6, 2023.

[13] Chihiro Nakatsuka and Satoshi Komorita. Denoising 3d human poses from low-resolution video using variational autoencoder. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4625–4630, 2021.

[14] Savvas Karatsiolis and Christos N. Schizas. Conditional generative denoising autoencoder. *IEEE Transactions on Neural Networks and Learning Systems*, 31(10):4117–4129, 2020.

[15] Vinayak Abrol, Pulkit Sharma, and Arijit Patra. Improving generative modelling in vaes using multimodal prior. *IEEE Transactions on Multimedia*, 23:2153–2161, 2021.

[16] Hegen Yan and Zhihua Lu. A disentangled recurrent variational autoencoder for speech enhancement. In *2023 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1697–1702, 2023.

[17] Oyebade K. Oyedotun and Djamila Aouada. A closer look at autoencoders for unsupervised anomaly detection. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3793–3797, 2022.

[18] Qian Xiang and Xuliang Pang. Improved denoising autoencoders for image denoising. In *2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–9, 2018.

[19] Zehao Fan, Yi Wang, Lihua Meng, Guangyao Zhang, Yi Qin, and Baoping Tang. Unsupervised anomaly detection method for bearing based on vae-gan and time-series data correlation enhancement (june 2023). *IEEE Sensors Journal*, 23(23):29345–29356, 2023.

[20] Matthieu Molinier and Jorma Kilpi. Avoiding overfitting when applying spectral-spatial deep learning methods on hyperspectral images with limited labels. In *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 5049–5052, 2019.