Comparative study to augment dataset using ConditionalGAN and InfoGAN

Shantanu Ghosh
4311-4360
University of Florida
shantanughosh@ufl.edu

Kalpak Seal
8241-7219
University of Florida
kalpak.seal@ufl.edu

Abstract

Supervised Learning is expensive since a classifier always needs labelled data. In domains where datasets are small, methods like scaling, shifting and rotating of images have been used to augment the dataset to learn meaningful features. Later, with semi-supervised learning, unlabeled datasets have been leveraged to augment the dataset improving the accuracy of the classifier. Since it's inception, Generative Adversarial Networks (GAN) has been a very popular generative model to learn the true data distribution to generate new data samples which can be used to augment a particular dataset. In this study, we will use two novel variants of GAN - 1) Conditional GAN and 2) InfoGAN to augment the dataset and compare the classifier's performance using a novel dataset augmentation algorithm. Our experiments show that with less training samples from the original dataset and augmenting it using the generative models, the classifier can achieve similar accuracy.

1 Introduction

Unsupervised learning is a domain in machine learning where we only have access to the data points, but without its corresponding labels. Labelling data is cumbersome as it requires significant amount of human effort and time. In the era of big data, there is a plethora of large datasets without labels available. So, if we are able to use unsupervised learning to extract meaningful features from the unlabelled datasets, we can use those features in many downstream tasks - like classification, regression and policy learning in reinforcement learning. Due to this, representation learning has become popular which extracts meaningful features from a large unlabelled dataset.

One of the challenge of using representation learning is to learn the disentangled representation of the features. Disentangled representation aims to impersonate the quick intuition process of a human, using both "high" and "low" dimensional reasoning. It is a technique where each feature is broken down into narrowly defined variables and encodes them as separate dimensions. For example: in a typical prediction task of determining the sex of a person, the network will take high dimensional feature vectors like age, clothing type, height into account. Now a generative network that tries to disentangle the features to produce the images of persons from a dataset, it tries to allocate separate lower dimensional features such as: height of the person, length of the arms and legs, types of shirt and pants etc. Often the disentangled representation is misjudged as distributed representation. In disentangled representation, a single node, or a neuron, can learn a complete feature, independent of other nodes, if the disentangled dimensions match an output's dimension. For example, if trying to tell the difference between a child and an adult, a node checking the disentangled vector of left leg length can discover the entire picture is a child due to the maximum value of the parameter, even though that was not the objective of the node. In distributed representation, objects are represented by their particular location in the vector space, so the algorithm must "run its course" and run through all nodes to arrive at the output of "child."

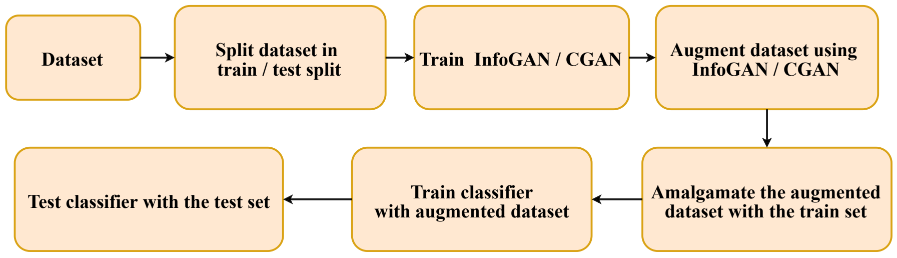
Dataset
Split dataset in train / test split
Train InfoGAN / CGAN
Augment dataset using InfoGAN / CGAN
Amalgamate the augmented dataset with the train set
Train classifier with augmented dataset
Test classifier with the test set

Figure 1: Illustrative diagram of the augmentation algorithm.

Unsupervised representation learning has been dominated by the deep generative models, primarily by Variational Autoencoder(VAE)[13] and Generative Adversarial Network (GAN)[6]. Since the arrival of Generative adversarial networks , it has been used in various domains to generate realistic images. In last couple of years, several variations of GANs have been published which improved the quality of generated images than the original vanilla GAN - two such variations are Conditional GAN [15] and InfoGAN [4] which we have been used in this study to augment the dataset and then, compare the accuracy of the classifier. The InfoGAN explicitly helps us to get the disentangled representation of features which helps in producing more realistic images and showed better performance than the original GAN.

Dataset augmentation can be useful in many tasks from language translation to medical image analysis. However, there is a growing research area in Biomedical Informatics where data-augmentation can be utilised efficiently. In biomedical informatics, there is a technique called propensity score matching(PSM) [3] that matches two groups - treated(who undergo the treatment) or control(who are given a placebo) based on a score, when a group of patients whose outcomes to a particular treatment or drug have to be investigated. Now it has observed that the people who undergo the treatment, i.e the treated group always has less number compared to the control group and as a result of PSM, a large number of samples from control group have to be left out. Using data augmentation, we can generate the treated samples for their control counterpart which helps to cover the entire feature space.

In this study, we initially designed a classifier to test the MNIST dataset on 100% train data. Then we reduced the number of training samples based on various configurations and mixed that training set with the synthetically generated samples from the 2 GANs where as the remaining training data is amalgamated to the test set. The illustrative diagram of the entire algorithm is shown in figure 1.

2 Related Work

There exists a vast literature of unsupervised representation learning. Initially, it's superiority has been showcased in works where Boltzmann machines [9] have been used. Later with the introduction of the autoencoders [2] and it's variations such as sparse autoencoders[16], denoising autoencoders[18] and stacked denoising autoencoders[19], the researchers were able to learn the underlying latent vector in more compact manner. Later this research field was pushed to next level with deep probabilistic generative models like VAE[13] and GAN [6]. While the motivation of VAE lies in the theory of variation inference, the goal of GAN is to learn the true data distribution using a two player minimax game in an adversarial manner achieving the Nash equilibrium[10]. Over second half of the last decade, more variations of the original GAN has come up such as InfoGAN[4], CGAN[15], WGAN[1], StyleGAN[11], CycleGAN [21], enriching this field of research.

In this study we tried to showcase the power of unsupervised representation learning by using deep generative models - InfoGAN and CGAN to augment the train set and use those synthetically generated samples to train the classifier and compare it's performance with the one which was trained on the 100% train set.

3 Methods

3.1 Model Description

3.1.1 GAN

In standard GAN [6], there are two neural networks - generator (G) and discriminator (D) - which plays a two player minimax game, trained in the classical adversarial manner, such that the generator learns the true data distribution $\boldsymbol{p(x)}$ and tries to fool the discriminator by producing fake samples. The standard optimization function to train a GAN with data distribution as $\boldsymbol{p_{data}(x)}$ and random noise as $\boldsymbol{p_Z(z)}$ is as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(\boldsymbol{z}))] \tag{1}$$

Here the Generator *G*, independent of the first term, is optimized with m mini-batches as follows

$$\min_G \phi_g(D,G) = \frac{1}{m}\sum_{i=1}^{m}\left[\log\left(1 - D\Big(G\Big(\boldsymbol{z^{(i)}}\Big)\Big)\right)\right] \tag{2}$$

The discriminator *D* in standard GAN whose primary objective is to differentiate the real images and fake generated images. The discriminator *D* is optimized with m mini-batches as follows:

$$\max_{D_G} \phi_d(D,G) = \frac{1}{m}\sum_{i=1}^{m}\left[\log D\Big(\boldsymbol{x^{(i)}}\Big) + \log\left(1 - D\Big(G\Big(\boldsymbol{z^{(i)}}\Big)\Big)\right)\right]$$

3.1.2 Conditional GAN

One of the drawbacks of GAN is that there is no way to control the types of images that are generated other than trying to figure out the complex relationship between the latent space input to the generator and the generated images. CGAN overcomes this problem as in CGAN[15] along with the random noise vector Z, we also provide the class vector explicitly to both generator and discriminator which controls the label of the generated image. In CGAN, $G : \mathbb{R}^{74} \to \mathbb{R}^{784}$, a function that maps 74 - dimensional noise vector to 28x28 dimensional image. This 74 dimensional noise vector is further divided into 64 dimensional random noise vector $z \sim p_z(z)$ and n dimensional vector which represents one hot encoded representation of the class of the generated image. In our project n = 10, as there are 10 classes in the MNIST dataset. CGAN optimizes the following objective function:

$$\min_G \max_D V_C(D,G) = \mathbb{E}_{x\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y}))] \tag{3}$$

where y is the noise vector. So, in CGAN, the generator *G* and discriminator *D* play the minimax game conditioned on y. The architecture of the CGAN model is shown in figure 2.

3.1.3 InfoGAN

In the ConditionalGAN, a noise vector and the given class are given as input to the generator. In contrast to the ConditionalGAN, InfoGAN [4] does not require to be provided with another additional class information externally, it will require an additional algorithm (Information maximization) to capture the class label from the noise vector itself. In InfoGAN, the output of the generator will be feed to the discriminator for classifying real and fake samples and also to another neural network (Q) which will be able to construct the class from the generated image from generator using Information Maximization. The complete objective function of InfoGAN is as follows:

$$\min_G \max_D V_I(D,G) = V_G(D,G) - \lambda I(c; G(z,c)) \tag{4}$$

where $V_G(D,G)$ is denoted as,

$$\min_G \max_D V_G(D,G) = \mathbb{E}_{x\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{z\sim p_z(z),c\sim p_c(c)}[\log(1 - D(G(\boldsymbol{z},\boldsymbol{c}))] \tag{5}$$

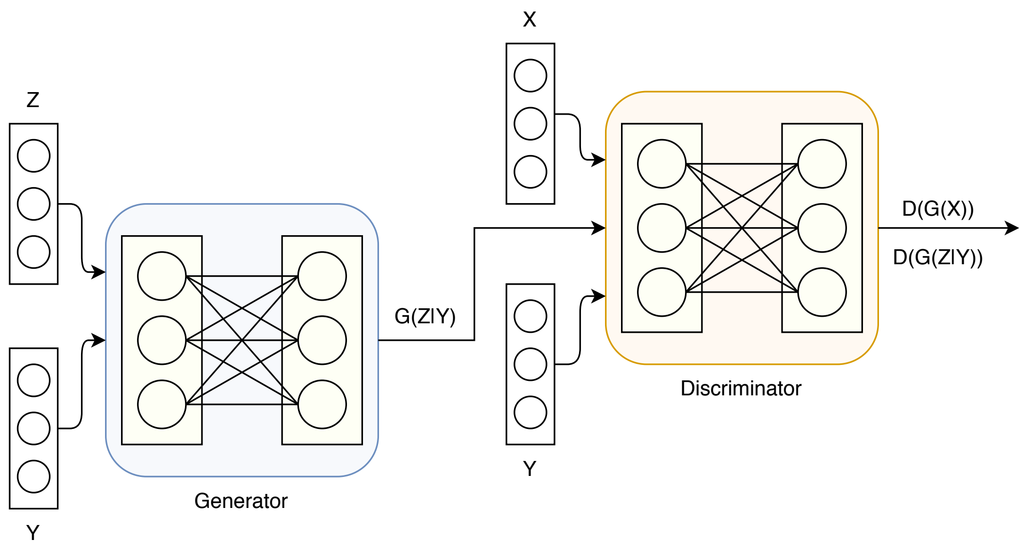

Figure 2: Architecture of Conditional GAN.

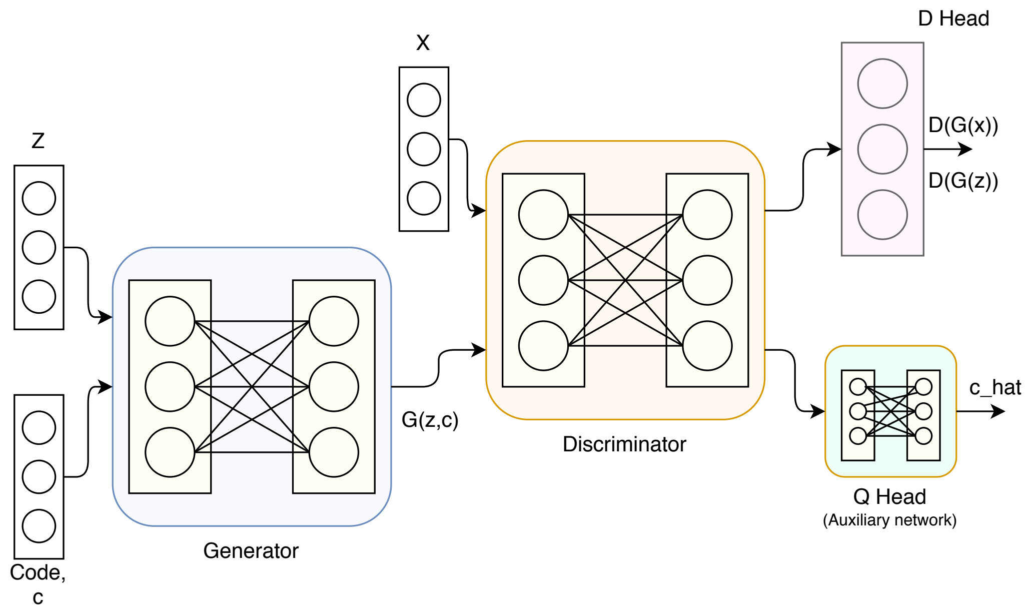

Figure 3: Architecture of InfoGAN.

In InfoGAN, the input noise to generator *G*, is divided into two vectors $z \sim p(z)$ and $c \sim p(c)$. The generator takes the concatenated input $\{z, c\}$. The generated sample from the generator is denoted by $G(z, c)$ and it will be fed to discriminator and another auxiliary network (Q) which captures some noticeable feature on to real data by maximising the mutual information between c and $G(z, c)$. The mutual information between c and $G(z, c)$ is denoted by $I(c; G(z, c))$. In this study the z is represented as 62 dimensional vector. As discussed in the paper [4], latent code c is further divided into discrete and continuous latent codes. The discrete latent codes or the categorical codes are 10 dimensional vectors (representing number of classes of MNIST dataset) and the two continuous latent codes are sampled from $Unif(-2, 2)$ denoting the rotation of digits and variations of width respectively. Through out this article, we denote the latent codes controlling the rotation of digits as type 1 noise, and variations of width as type 2 noise. So in InfoGAN, $G : \mathbb{R}^{74} \rightarrow \mathbb{R}^{784}$, a function that maps 74 - dimensional (random + categorical + continuous) noise vector to 28x28 dimensional

Table 1: Configuration of various datasets used in this study

Dataset	**Train Data**	**Test Data**
MNIST(100% train)	60000	10000
CGAN + MNIST(50% train)	(30+30) x 1000 = 60000	40000
CGAN + MNIST(20% train)	(12+48) x 1000 = 60000	58000
CGAN + MNIST(10% train)	(6+54) x 1000 = 60000	64000
InfoGAN with type1 noise + MNIST(50% train)	(30+30) x 1000 = 60000	40000
InfoGAN with type1 noise + MNIST(20% train)	(12+48) x 1000 = 60000	58000
InfoGAN with type1 noise + MNIST(10% train)	(6+54) x 1000 = 60000	64000
InfoGAN with type2 noise + MNIST(50% train)	(30+30) x 1000 = 60000	40000
InfoGAN with type2 noise + MNIST(20% train)	(12+48) x 1000 = 60000	58000
InfoGAN with type2 noise + MNIST(10% train)	(6+54) x 1000 = 60000	64000

image. We believe that the complete mathematical derivation of the evidence lower bound (ELBO) from the information maximization theory in the main paper of InfoGAN [4] is too short, so based on our understanding we derived the ELBO in details in Appendix A1. The complete architecture of the InfoGAN model is shown in figure 3.

4 Experimental setup

4.1 Dataset

We have used the MNIST dataset to perform our experiments where each image is of 28x28 pixels. The dataset consists of hand written digits from 0 to 9, uniformly distributed. The training dataset consists of 60000 images and the test dataset has 10000 images. First, we run our model on this available train and test data of MNIST and note down the model performance. Next, we use our synthetically generated data from CGAN and InfoGAN to augment the existing train datasets and perform several experiments. After augmentation the test dataset has images from the original MNIST test set comprises of 10000 images and the part of the MNIST train data not used for training.

The complete configurations of all the variations of MNIST datasets used for in this study is described in the table 1. This project has been developed in MacBook Pro i7, 16GB memory mounting Mac OS Big Sur and we ran all the experiments on Google Cloud Platform.

4.2 Deep learning modules' settings

This section describes the configurations of the various deep learning modules used in this project. This project is developed in PyTorch framework. Unless specified explicitly, all the parameters are the default parameters specified by PyTorch. The complete algorithms of augmenting the dataset using CGAN and InfoGAN are discussed in the Appendix A1.

4.2.1 CGAN

In CGAN, for the generator a 4 layered convolutional neural network is used. All the layers except the final layer use ReLU [14] as activation function. The final layer of the generator uses tanh as the activation function. For the discriminator, a 2 layered convolutional neural network is used. All the layers except the last layer use Leaky ReLU [20] with negative slope of 0.2 as the activation function. The final layer of the discriminator uses sigmoid activation function.

The CGAN is trained for 100 epochs with a batch size of 128. All the parameters of the discriminator and generator are updated with the ADAM [12] optimizer with learning rate as 0.00002. The complete details of the architecture of the CGAN model is discussed in Appendix A2.1.

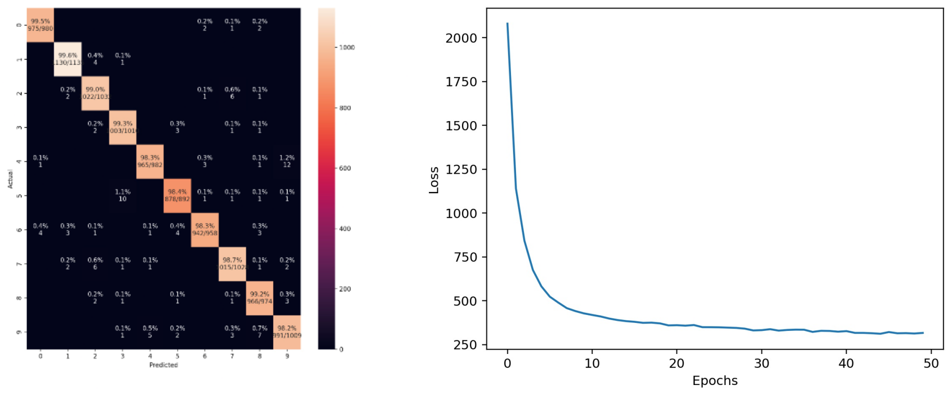

Figure 4: Confusion Matrix (left) and the loss vs epoch (right) plot of the CNN classifier trained on the 100% training data of MNIST dataset.

4.2.2 InfoGAN

In InfoGAN, for the generator a 4 layered convolutional neural network is used. All the layers except the final layer use ReLU [14] as activation function. The final layer of the generator uses Sigmoid as the activation function. For the discriminator, a 3 layered convolutional neural network is used. All the layers except the last layer use Leaky ReLU [20] with negative slope of 0.2 as the activation function.

The InfoGAN is trained for 100 epochs with a batch size of 128. All the parameters of the discriminator and generator are updated with the ADAM [12] optimizer with learning rate as 0.00002. As per the paper [4] the auxiliary network Q shares the same initial convolution layers as the discriminator. To accomplish this task in this study, we divide the discriminator into two separate networks - 1) QHead - which learns the categorical and continuous latent codes and 2) DHead - which discriminates between the real and fake samples. The complete details of the architecture of the InfoGAN model is discussed in Appendix A2.2.

4.2.3 CNN Classifier

For the classifier module, we used a 3-layer convolutional neural network. The parameters of the network are optimized using Adam optimizer[12] and learning rate is set to 0.0001. All the other parameters are the default parameters, set by the PyTorch framework. We ran the classifier for 50 epochs and used ReLU as the activation function of all the convolution and the fully connected layers. The fully connected layers are regularized using a dropout of probability 0.8. The weights of the Convolutional Neural Networks are initialised using Kaiming [8] for the convolutional layers and Xavier [5] for the fully connected layers. The detailed configuration of the classifier network is discussed in Appendix A2.3.

4.3 Validation and Performance metrics

We report the precision, recall, f1 score and accuracy, tested each of test set as mentioned in the table 2.

4.4 Program Execution

Set the path of **model_dir** in **convNet_manager.py** corresponding to the folder present under **experiments** directory depending upon the configuration of your choice.

For example : **experiments/mnist_infoGan_30k_noise1** has the configurations detailed under **config.json** and set **model_dir='experiments/mnist_infoGan_30k_noise1** in **convNet_manager.py**.

Then, execute the command: **python3 convNet_manager.py**

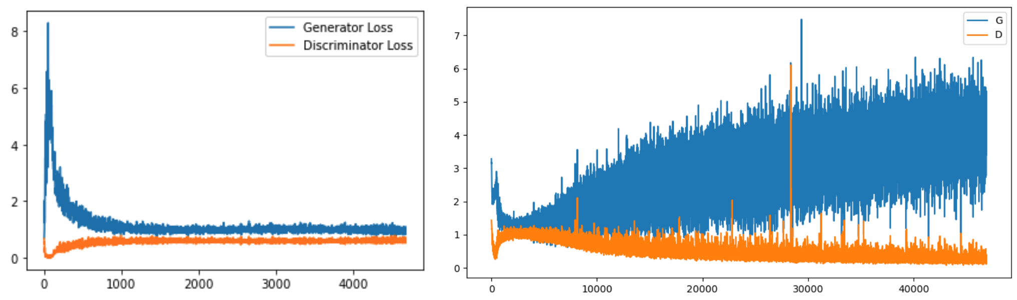

Figure 5: The loss vs accuracy plots of CGAN(left) and InfoGAN(Right). The X and Y axes denote the iterations and the loss value respectively.

A detailed description about the code is added to README.md file which is attached to the codebase, uploaded with the report.

5 Results & analysis

The detailed results of all the experiments are shown in table 2. We see that the CNN classifier achieves a 98.6% accuracy when trained on the MNIST dataset with 100% training samples. Now with CGAN, we achieve our best accuracy of 98.3% when trained on 50% training samples from the MNIST dataset and the rest from synthetically generated data. As we decreased the MNIST training data size to 20% and 10%, the accuracy also dropped to 97% and 96.5% along with other metrics - precision, recall and F1 score. This is expected as we were relying more on the synthetic images which generated by CGAN from a limited training set.

Compared to CGAN, InfoGAN - trained on 50% train set - performed marginally well and almost matched with the accuracy of the main classifier (trained on 100% train set) for both the types of continuous noise samples - 1) type1 as the rotation of the digits and 2) type2 as the variation along the width of the digits. The classifier trained on 50% trained synthetic image samples, generated by InfoGAN, achieves an accuracy of 98.5% compared to the 98.6% accuracy of the classifier, trained on 100% original training samples. This is probably because the InfoGAN learns the labels from the hidden categorical noise given to the generator as input using the Information maximization theory, discussed in Appendix 2.3. As per the expectation, the performance of the classifier dropped when we trained the classifier on 20 and 10 % of the training data and 80 and 90 % of InfoGAN generated synthetic images. However, the features learned by both the GANs have a significant impact and as they generated realistic data, the classifier achieved a significant accuracy for all the configurations.

The confusion matrix and the loss vs epoch plot of the classifier is shown on figure 4. The confusion matrices of the performance and the loss vs accuracy plot of the different classifiers trained with GAN generated synthetic images are discussed in Appendix A4 and A5 respectively. We also plotted how the loss function for both CGAN and InfoGAN converged and it is shown in figure 5.

6 Discussion

In this project, we trained both the GANs with MNIST dataset. One limitation of our study was the limited access of GPU and time; so with access to more computational power, this study can be extended to more datasets like CIFAR 10 or CIFAR 100. Also it will be interesting to see how the different configurations of GAN will perform with more variations of different hyper-parameters. Also another possible extension of this project will be the use of other generative models like Variational autoencoder (VAE) and check how the generated images from the VAE will impact the performance of the classifier. Also we have used a simple convolution neural network(CNN) in the generator module for the GANs, so usage of complicated CNN architecture like Resnet-50 [7] or Incepetion [17] can possibly extract more meaningful features and generate more realistic images with significant variations. Lastly usage of models pretrained on large dataset like ImageNet can be have a large impact of leveraging our approach to a wide-range of research – especially in the area of medical image analysis where the dataset is often limited by a small number of samples.

Table 2: Performance of classifiers trained on various configuration of MNIST dataset

Dataset	**Precision**	**Recall**	**F1 score**	**Accuracy**
100% Train set	**0.986**	**0.986**	**0.986**	**0.986**
CGAN(type1 noise)				
+ 50% Train set	**0.983**	**0.983**	**0.983**	**0.983**
+ 20% Train set	0.970	0.970	0.970	0.970
+ 10% Train set	0.965	0.965	0.965	0.965
InfoGAN(type1 noise)				
+ 50% Train set	**0.985**	**0.985**	**0.985**	**0.985**
+ 20% Train set	0.979	0.978	0.978	0.979
+ 10% Train set	0.965	0.964	0.964	0.964
InfoGAN(type2 noise)				
+ 50% Train set	**0.985**	**0.985**	**0.985**	**0.985**
+ 20% Train set	0.977	0.977	0.977	0.977
+ 10% Train set	0.969	0.968	0.968	0.968

7 Conclusion

In this study, we performed the data-augmentation of the training set using two variations of GANs - CGAN and InfoGAN. This study once again showcased the power of using generative models like GAN which showed that a classifier trained on a small training set and the synthetically generated samples from GAN can achieve similar comparable results with the one that was trained on the original 100% train-set. This approach can be an alternative to research areas where labeled-dataset is often limited by a small number of samples and can take the large amount of unlabelled data into account to learn meaningful features to boost the performance of the downstream classifier.

8 Authors' contributions

Both the team member contributed equally to the project and report. SG was responsible in understanding the CGAN and InfoGAN theory, design and training, generating synthetic images to augment the dataset for both the noise while KS was responsible for writing data preprocessing pipelines, classifier model design, training, reporting the accuracies, recording the performance for each dataset and plotting pipelines.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *arXiv preprint arXiv:2003.05991*, 2020.

[3] Marco Caliendo and Sabine Kopeinig. Some practical guidance for the implementation of propensity score matching. *Journal of economic surveys*, 22(1):31–72, 2008.

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

[5] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

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

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. corr abs/1512.03385 (2015), 2015.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[9] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[10] Charles A Holt and Alvin E Roth. The nash equilibrium: A perspective. *Proceedings of the National Academy of Sciences*, 101(12):3999–4002, 2004.

[11] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[15] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[16] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[17] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[18] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[19] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

[20] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015.

[21] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.

Appendix

A1 Algorithm

The complete algorithm to augment the train set and use it to train the classifier is discussed in Algorithm 1 and 2 respectively.

A2 InfoGAN optimization

A2.1 Optimal Discriminator

Combining equation (4) and (5),

$$D^* = \max_D V_I(D,G) = \max_D V_G(D,G) \tag{6}$$

Algorithm 1 Data-augmentation using CGAN

```
Input: Training set $\boldsymbol{X}$ = {($x^{(1)}$, $y^{(1)}$),..., ($x^{(n)}$, $y^{(n)}$) }, loss function $L(.,.)$, hyperparameter $\beta > 0$, generator network $G$ with initial parameters $\theta_g$, discriminator network $D$ , classifier network with initial parameters $\theta_d$, epochs $K$
1: Divide the train set as per requirement. For example: if the train set is divided into 20-80 split, use the 20% for training and add the remaining 80% to the test set which will be used for testing
2: for $epochs = 1, 2, \ldots K_s$ do
3:     Sample minibatch of $m$ noise samples $\{z^{(i)}\}_{i=1}^{m}$ where $z \sim p_z(z)$ and $z^{(i)} \in \mathbb{R}^{64}$
4:     Sample minibatch of $m$ samples of one-hot representation of the 10 dimensional label vector and concatenate with z
5:     Update the discriminator $D$ as per the equation (3) by ascending its stochastic gradient: $\nabla_{\theta_d}(\phi_d(D, G))$
6:     Sample minibatch of $m$ noise samples $\{z^{(i)}\}_{i=1}^{m}$ where $z \sim p_z(z)$
7:     Update the generator $G$ as per the equation (3) by descending its stochastic gradient: $\nabla_{\theta_g}(\phi_g(D, G))$
8: end for
9: Remove the discriminator $D$ and get the generated samples from the generator $G$ and amalgamate them with the train set.
10: Train the classifier with the synthetically generated train set from CGAN model
11: Test the classifier with the test set and record the performance metrics.
```

Algorithm 2 Data-augmentation using InfoGAN

```
Input: Training set $\boldsymbol{X}$ = {($x^{(1)}$, $y^{(1)}$),..., ($x^{(n)}$, $y^{(n)}$) }, loss function $L(.,.)$, hyperparameter $\beta > 0$, generator network $G$ with initial parameters $\theta_g$, discriminator network $D$ , classifier network with initial parameters $\theta_d$, epochs $K$
1: Divide the train set as per requirement. For example: if the train set is divided into 20-80 split, use the 20% for training and add the remaining 80% to the test set which will be used for testing
2: for $epochs = 1, 2, \ldots K_s$ do
3:     Sample minibatch of $m$ noise samples $\{z^{(i)}\}_{i=1}^{m}$ where $z \sim p_z(z)$ and $z^{(i)} \in \mathbb{R}^{62}$
4:     Sample minibatch of $m$ categorical noise samples $\{c^{(i)}\}_{i=1}^{m}$ where $c \sim p_c(z)$ and and $c^{(i)} \in \mathbb{R}^{10}$
5:     Sample minibatch of $m$ continuous noise samples denoting the rotation of digits $\{c^{(i)}\}_{i=1}^{m}$ where $c \sim Unif(-2, 2)$ and and $c^{(i)} \in \mathbb{R}$
6:     Sample minibatch of $m$ continuous noise samples denoting the variation in width $\{c^{(i)}\}_{i=1}^{m}$ where $c \sim Unif(-2, 2)$ and and $c^{(i)} \in \mathbb{R}$
7:     Amalgamate the categorical and 2 continuous noise samples with the random noise sample $z$
8:     Update the discriminator $D$ as per the equation (4) by ascending its stochastic gradient: $\nabla_{\theta_d}(\phi_d(D, G))$
9:     Sample minibatch of $m$ noise samples $\{z^{(i)}\}_{i=1}^{m}$ where $z \sim p_z(z)$
10:    Update the generator $G$ as per the equation (4) by descending its stochastic gradient $\nabla_{\theta_g}(\phi_g(D, G))$
11: end for
12: Remove the discriminator $D$ and get the generated samples from the generator $G$ and amalgamate them with the train set.
13: Train the classifier with the synthetically generated train set from CGAN model
14: Test the classifier with the test set and record the performance metrics.
```

A2.2 Optimal Generator

Combining equation (4) and (5),

$$\begin{aligned} G^* &= \min_G V_I(D,G) \\ &= \min_G \left\{ \mathbb{E}_{z\sim p_z(z), c\sim p_c(c)}[\log(1 - D(G(\boldsymbol{z},\boldsymbol{c}))] - \lambda I(c; G(z,c)) \right\} \end{aligned} \tag{7}$$

Now to get G^*, we need to minimise equation (7) which implies the maximization of $I(c; G(z,c))$. Now the mutual information $I(c; G(z,c))$ can be expressed in terms of entropy as,

$$I(c; G(z,c)) = H(c) - H(c|G(z,c))$$

$I(c; G(z,c))$ will be maximum if $H(c|G(z|c)) = 0$ which indicates that the latent code c will be completely determined by $G(z,c)$, which is the generated image from the generator. This indicates that if we maximize the mutual information $I(c; G(z,c))$, we can the complete latent code information, including the class label from the generated image $G(z,c)$.

A2.3 Variational information maximization

$$\begin{aligned} I(c; G(z,c)) &= H(c) - H(c|G(z,c)) \\ &= H(c) + \int\int p(C=c', X=G(z,c)) \log p(C=c'|X=G(z,c))\, dc'dx \\ &= H(c) + \mathbb{E}_{x\sim G(z,c)}\mathbb{E}_{c'\sim p(c|x)} \log(p(c'|x)) && (8) \\ &= H(c) + \mathbb{E}_{x\sim G(z,c)}\mathbb{E}_{c'\sim p(c|x)} \log\left[\frac{p(c'|x)}{Q(c'|x)} Q(c'|x)\right] \\ &= H(c) + \mathbb{E}_{x\sim G(z,c)}\mathbb{E}_{c'\sim p(c|x)} \log\left[\frac{p(c'|x)}{Q(c'|x)}\right] + \mathbb{E}_{x\sim G(z,c)}\mathbb{E}_{c'\sim p(c|x)} \log\left[Q(c'|x)\right] \\ &= H(c) + \mathbb{E}_{x\sim G(z,c)} \int p(c'|x) \log\frac{p(c'|x)}{Q(c'|x)} dc' + \mathbb{E}_{x\sim G(z,c)}\mathbb{E}_{c'\sim p(c|x)} \log\left[Q(c'|x)\right] \\ &= H(c) + \mathbb{E}_{x\sim G(z,c)}\left[D_{KL}\big(p(c'|x||Q(c'|x))\big)\right] + \mathbb{E}_{x\sim G(z,c)}\mathbb{E}_{c'\sim p(c|x)} \log\left[Q(c'|x)\right] \\ &\geq H(c) + \mathbb{E}_{x\sim G(z,c)}\mathbb{E}_{c'\sim p(c|x)} \log\left[Q(c'|x)\right] \\ &\geq H(c) + \mathbb{E}_{c\sim p(c)}\mathbb{E}_{x\sim G(z,c)}\mathbb{E}_{c'\sim p(c|x)} \log\left[Q(c'|x)\right] && (9) \\ &\geq H(c) + \mathbb{E}_{c\sim p(c)}\mathbb{E}_{x\sim G(z,c)} \log\left[Q(c|x)\right] = L_I(G,Q) && (10) \end{aligned}$$

The posterior density $p(c'|x)$ in equation (8) is hard to optimize, so we added the auxiliary distribution $Q(c'|x)$ which will can be optimized easily. This technique of lower bounding mutual information is known as Variational Information Maximization. Equation (9) is still hard to optimize as the latent c' is still sampled from posterior density $p(c|x)$, so we can not take the Monte Carlo approximation of Expectation. To get rid of this, we use Lemma 5.1 of InfoGAN paper[4] and get to equation (10). Thus we get rid of all the posterior density $p(c|x)$ and can use the Monte Carlo approximation. Now, $L_I(G,Q)$ can be directly maximised w.r.t Q and via reparametrization trick w.r.t G. As per the InfoGAN paper, we use neural network as Q to learn the conditional density c given x.

A3 Network configurations

A3.1 CGAN

All the layers of the generator and the discriminator comprise with a convolution and batch normalization layers. The detailed configuration of the generator and discriminator are discussed in table 3 and 4 respectively.

Table 3: Configuration of generator used in CGAN

Layer	Layer type	Depth	Kernel	Stride
1	ConvoTranspose2d	256	3*3	2
2	BatchNorm2d	256	-	-
3	ConvoTranspose2d	128	4*4	1
4	BatchNorm2d	128	-	-
5	ConvoTranspose2d	64	3*3	2
6	BatchNorm2d	64	-	-
7	ConvoTranspose2d	1	4*4	2

Table 4: Configuration of discriminator in CGAN

Layer	Layer type	Depth	Kernel	Stride
1	Convolution	64	4*4	2
2	BatchNorm2d	64	-	-
3	Convolution	128	4*4	2
4	BatchNorm2d	128	-	-
5	Convolution	1	4*4	2

A3.2 InfoGAN

In InfoGAN, all the layers of the generator comprise with a convolution and batch normalization layer. The discriminator is divided into two separate neural networks - DHead and QHead - with 3 shared convoltuion layers. The DHead consists of a single layer neural network with one output node and sigmoid activation function. The QHead consists of two layered convolutional neural network. The detailed architecture of the generator, discriminator and auxiliary (Q) network are discussed in table 5, 6 and 7 respectively.

A3.3 Classifier Network

The classifier used in this study is a typical convolutional neural network with convolution, max pool, batch normalization and fully connected layers. Each convolution layer uses a 3x3 filter with padding and stride set to 1. The network comprises of 3 such layers, after which the information is flattened and sent to a fully connected Neural Network having two hidden layers. The configuration of the classifier is discussed in the table 8.

A4 Confusion matrices of the different classifiers

The confusion matrices of the different classifiers trained on different configurations of the generated image samples from CGAN and InfoGAN are shown in figures 6 and 7 respectively.

Table 5: Configuration of generator used in InfoGAN

Layer	Layer type	Depth	Kernel	Stride	Padding
1	ConvoTranspose2d	1024	1*1	1	-
2	BatchNorm2d	1024	-	-	-
3	ConvoTranspose2d	128	7*7	1	-
4	BatchNorm2d	128	-	-	-
5	ConvoTranspose2d	64	4*4	2	1
6	BatchNorm2d	64	-	-	-
7	ConvoTranspose2d	1	4*4	2	1

Table 6: Configuration of discriminator in InfoGAN

Layer	**Layer type**	**Depth**	**Kernel**	**Stride**	**Padding**
1	Convolution	64	4*4	2	1
2	Convolution	128	4*4	2	1
3	BatchNorm2d	128	-	-	-
4	Convolution	1024	7*7	1	-
5	BatchNorm2d	1024	-	-	-

Table 7: Configuration of QHead in InfoGAN

Layer	**Layer type**	**Depth**	**Kernel**	**Stride**
1	Convolution	128	1*1	1
2	BatchNorm2d	128	-	-
3	Convolution	10	1*1	1
4	BatchNorm2d	10	-	-

A5 Loss vs accuracy plot of the different classifiers

The loss vs accuracy plots of the different classifiers trained on the CGAN and InfoGAN generated synthetic images are shown in figure 8 and 9 respectively.

GAN generated image samples

The synthetic samples generated by CGAN and InfoGAN are shown in figure 10.

Table 8: Configuration of CNN Classifier

Layer	**Layer type**	**Depth**	**Kernel**	**Stride**	**Padding**
1	Convolution	6	3*3	1	1
2	BatchNorm2d	6	-	-	-
3	Convolution	12	3*3	1	1
4	BatchNorm2d	12	-	-	-
5	Convolution	24	3*3	1	1
6	BatchNorm2d	24	-	-	-
7	Fully Connected	120	-	-	-
8	BatchNorm2d	120	-	-	-
9	Fully Connected	60	-	-	-
10	BatchNorm2d	60	-	-	-
11	Fully Connected	10	-	-	-

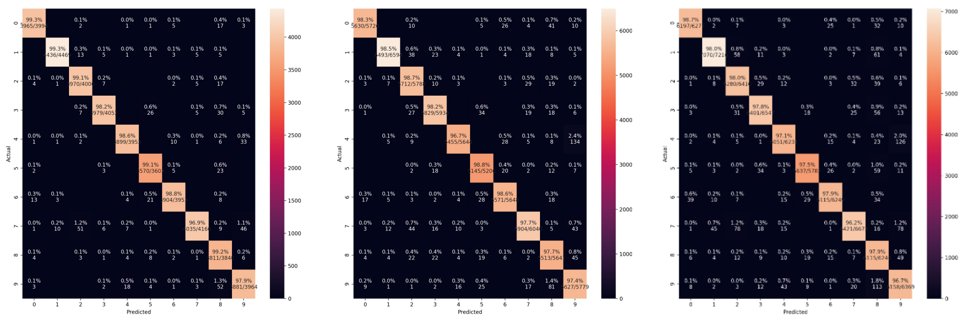

Figure 6: Confusion Matrices(from left to right) of the CNN classifiers trained on the 50%, 20% and 10% train set + augmented dataset generated by CGAN respectively.

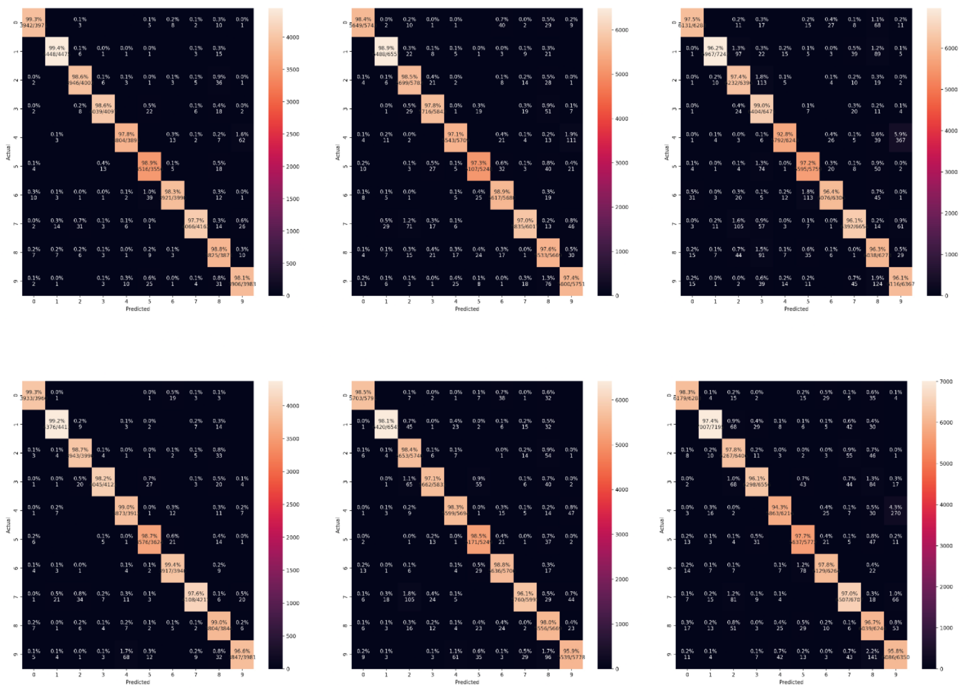

Figure 7: Confusion Matrices (from left to right) of the CNN classifiers trained on the 50%, 20% and 10% train set + augmented dataset generated by InfoGAN based on noise of type 1 - rotation of digits(top row) and type 2 - variations in width(bottom row) respectively.

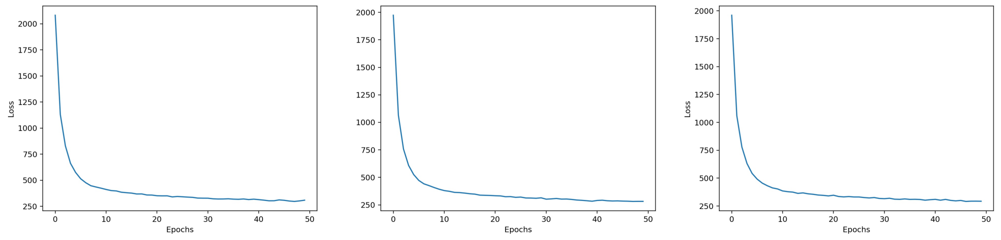

Figure 8: Loss vs epochs plot (from left to right) of the CNN classifiers trained on the 50%, 20% and 10% train set + augmented dataset generated by CGAN respectively.

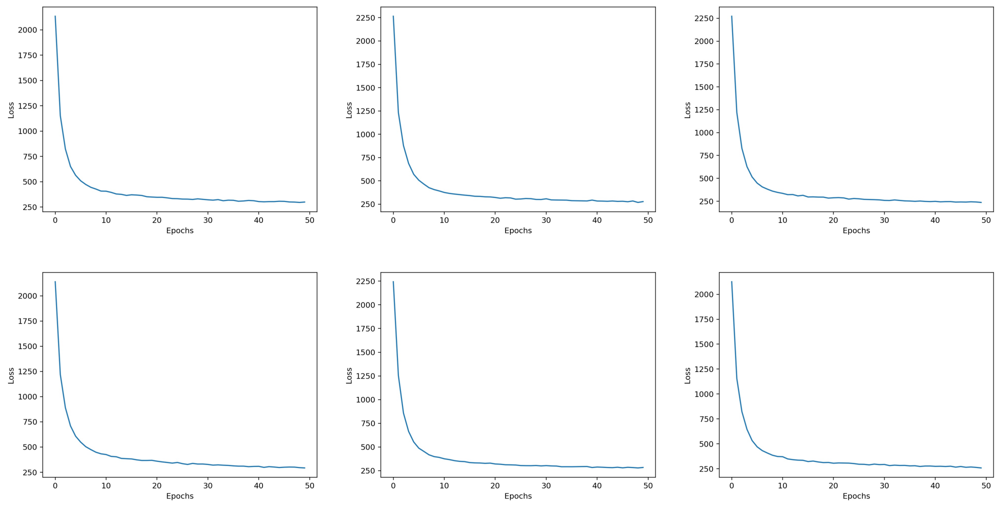

Figure 9: Loss vs epochs plot (from left to right) of the CNN classifiers trained on the 50%, 20% and 10% train set + augmented dataset generated by InfoGAN based on noise of type 1 - rotation of digits(top row) and type 2 - variations in width(bottom row) respectively.

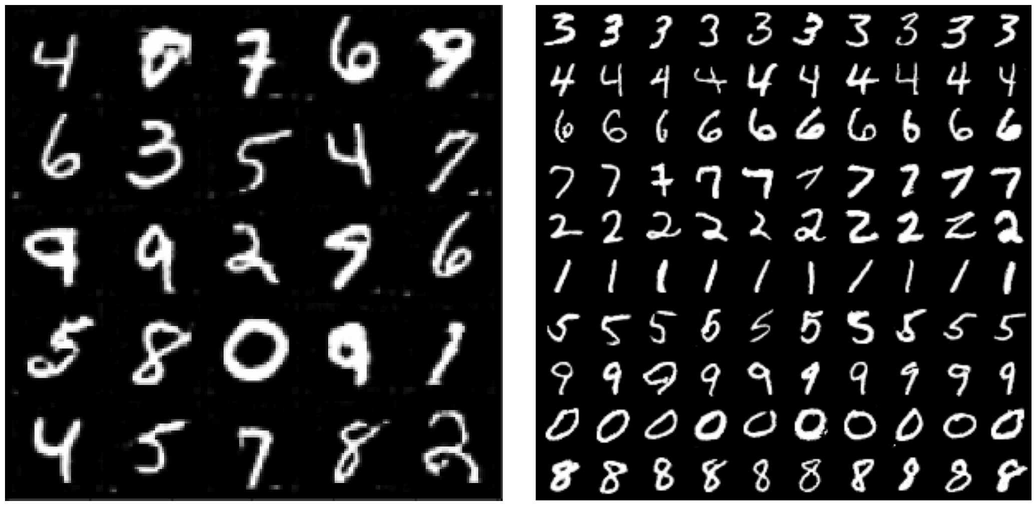

Figure 10: Generated images by CGAN(left) and InfoGAN(right) respectively after training for 100 epochs.