Exercise 2
Implementation of Coupled Generative Adversarial Networks on MNIST handwritten digit dataset

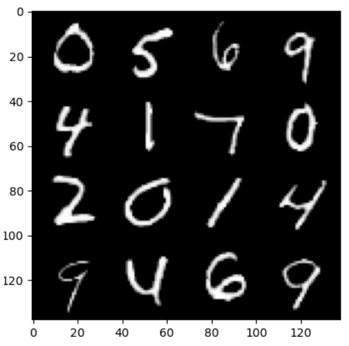
Student: Khuong G.T Diep

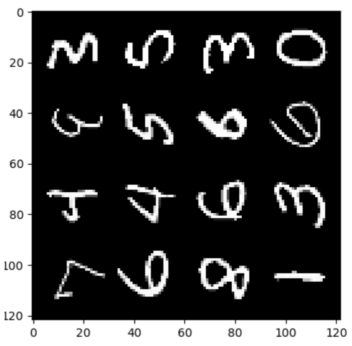
Advisor: Prof. Yong-Guk Kim

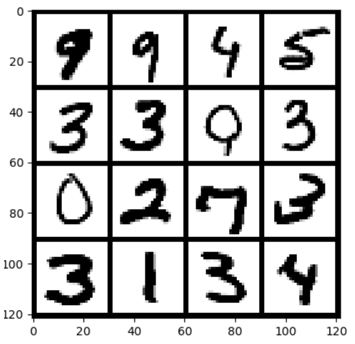
November 4, 2022

1 Introduction

The main point of this report is to implement the Coupled Generative Adversarial Networks (coGan) on MNIST dataset with four experiments: original, rotation, edge and negative images.


(a) Original images


(b) Rotated images


(c) Inverted images

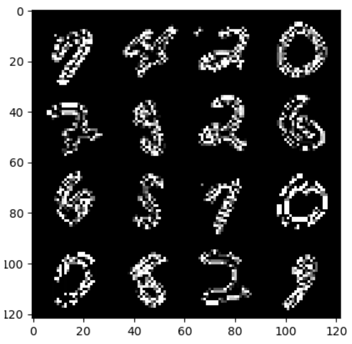

(d) Edge images

Figure 1: Four cases of the MNIST digit dataset used in training

First of all, we need to do some exploration steps in order to get some sense of the data [1]. The MNIST database of handwritten digits, which has a training set of 60,000 examples and a test set of 10,000 examples, is used for this problem. The digits have been size-normalized and centered in a fixed-size image [2]. The table below shows the number of samples for each class in our training and testing dataset.

Number	Training set	Testing set
0	5932	980
1	6742	1135
2	5958	1032
3	6131	1010
4	5842	982
5	5421	892
6	5918	958
7	6265	1028
8	5851	974
9	5949	1009

Table 1: Number of samples in each class

To visualize the dataset, a dimensionality reduction technique called T-distributed stochastic neighbor embedding is used to convert the higher dimensions dataset into lesser dimensions dataset. This technique are widely used when dealing with CNN networks [3].

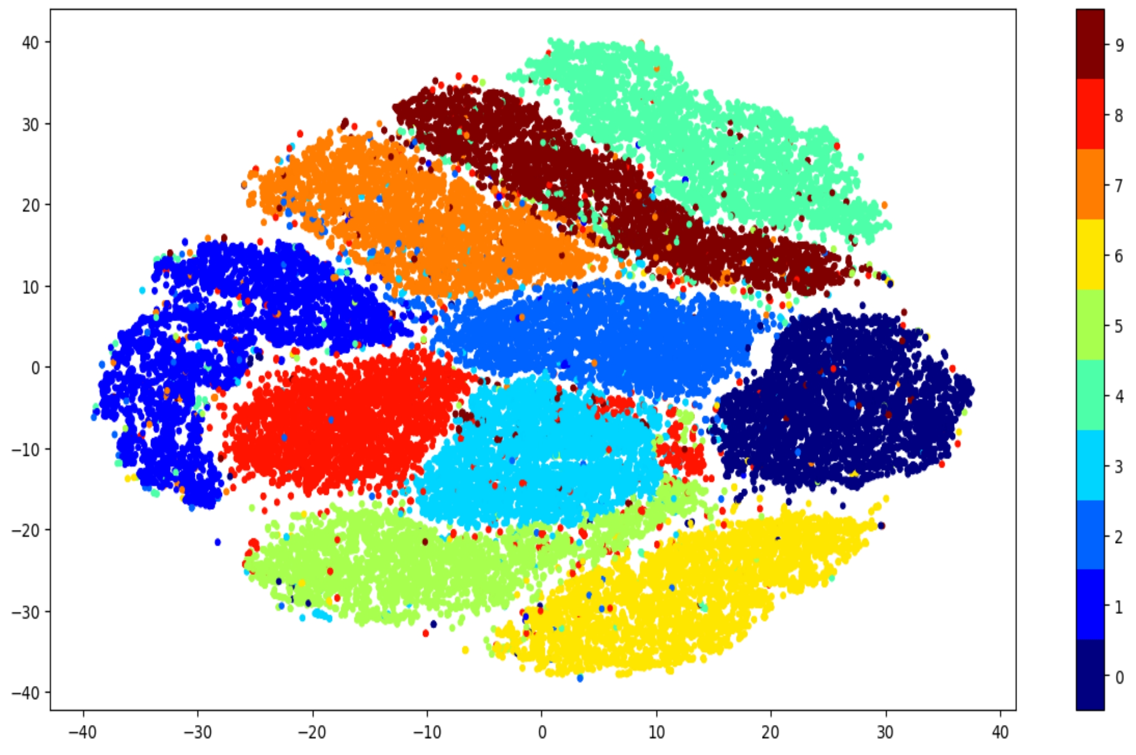


Figure 2: Visualizing the dataset by t-sne method

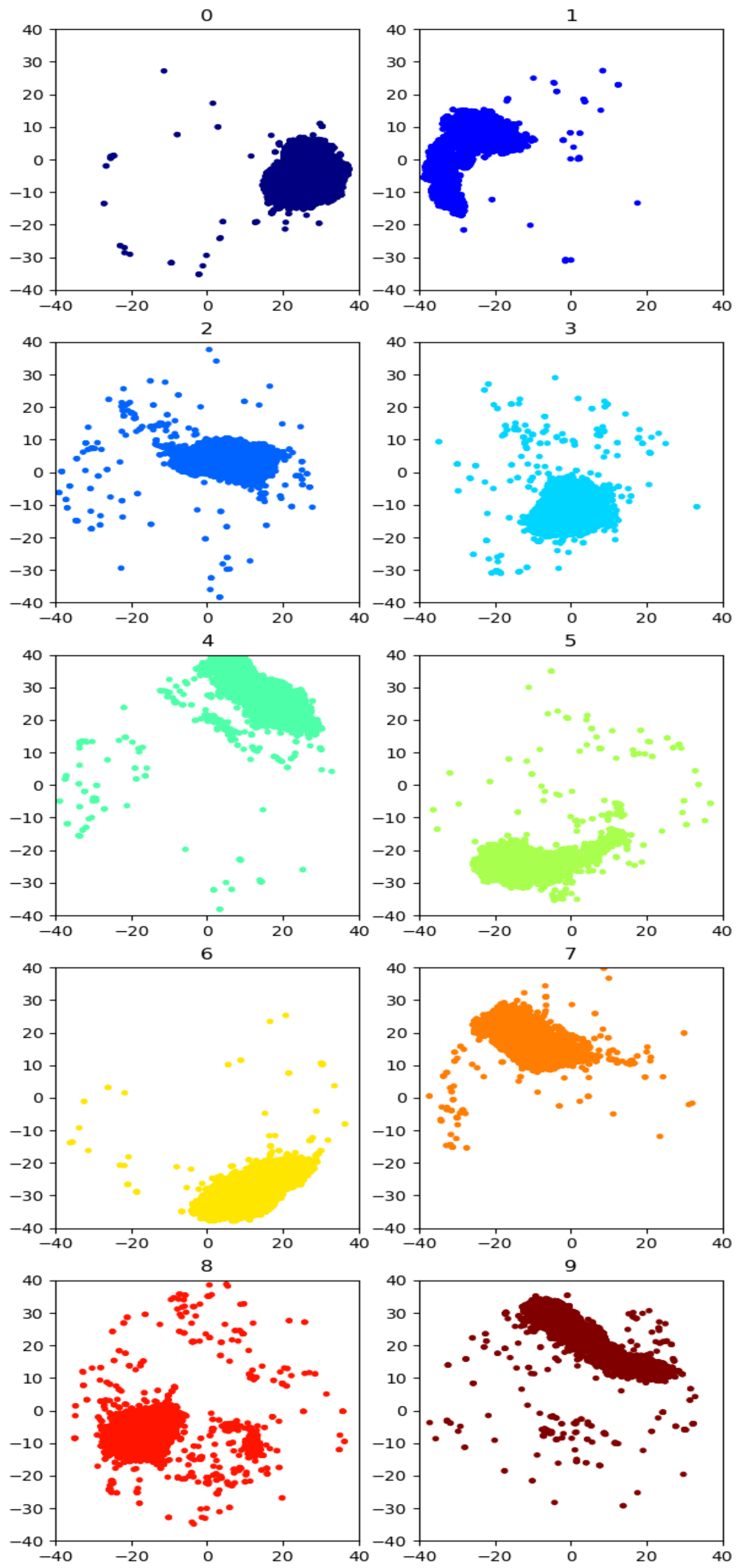


Figure 3: Distribution of 10 classes in the dataset

As can be observed in Fig. 3, the classes 3, 8 and 9 distributions are scattered while others are pretty good especially (0, 1, 6). In machine learning, understanding these distributions is important when you want to do feature engineering/scaling for a dataset.

2 Methods

2.1 Backpropagation

Back-propagation is a process that repeatedly updates the network's weight values in an attempt to minimize the difference between the network's actual output vector and the desired output vector [4]. Backpropagation is a popular approach in machine learning for training feedforward neural networks.

2.2 Convolutional neural network

A convolutional neural network (CNN) is a type of artificial neural network used in deep learning (ANN). It is similar to conventional ANNs in that it is made up of neurons that self-optimise through learning. Each neuron will continue to receive input and conduct an operation (such as a scalar product followed by a non-linear function), which is the foundation of many ANNs. The entire network will still express a single perceptive score function from the input raw picture vectors to the final output of the class score (the weight). The main significant difference between CNNs and normal ANNs is that CNNs are mostly utilized in visual pattern recognition [5].

2.3 Transpose convolutional neural network

The transposed Convolutional Layer, on the other hand, is also (incorrectly) referred to as the Deconvolutional Layer. A deconvolutional layer operates in the opposite direction of a conventional convolutional layer. Typically, a transposed convolutional layer is used for upsampling to provide an output feature map with a spatial dimension higher than the input feature map. The padding and stride parameters define the transposed convolutional layer, same as they do the conventional convolutional layer. These padding and stride values are those that were hypothetically applied to the output to generate the input [6].

2.4 Batch normalization

Batch normalization is a technique used between neural network layers that continuously normalizes the output from the previous layer before transferring it to the next layer. BN is used to stabilize the neural network, maintain data distribution, increase accuracy, and may be used to most models to improve the convergence rate of a neural network [7].

2.5 Activation function

In artificial neural networks, activation functions are utilized specifically to convert input signals into output signals that are then sent as input to the following layer in the stack. A neural network with no activation functions operates as a linear regression model with limited performance and power. A neural network sometimes has to do with more difficult tasks than only learn and compute a linear function, such as modeling complex types of data like images, videos, speech, etc. For this reason, large, high-dimensional, nonlinear datasets with a model that has several hidden layers and a complex structure should use activation functions [8].

The PReLU and Tanh are the two activation function which are used for my progam:

1. While ReLU returns zero when the inputs are negative and Leaky ReLU multiplies the negative input by a small constant (like 0.02) and keeps the positive input as is, PReLU multiplies the negative input by a learnable parameter a.

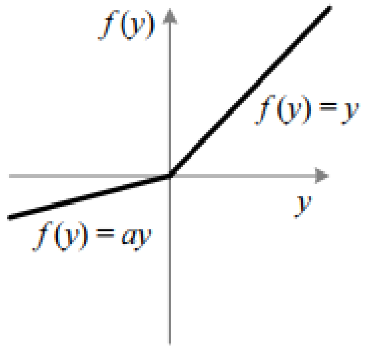


Figure 4: PReLU activation function

$$f(y_i) = \begin{cases} y_i, & \text{if } y_i > 0 \\ a_i y_i, & \text{if } y_i \leq 0 \end{cases} \tag{1}$$

2. Tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph.

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2}$$

3 Experiment

The techniques used in the my program are referenced from the paper ”Coupled Generative Adversarial Networks” [9] and a github repository from Erik Linder-Norén [10]. For further reference, my program can be found on github by the following link: https://github.com/KhuongDiep911/CoGan-MNIST.git

3.1 Generative adversarial networks (GAN)

The generative model is fought against an opponent in the proposed adversarial networks framework: a discriminative model that learns to distinguish whether a sample is from the model distribution or the data distribution. The generative model can be compared to a group of counterfeiters attempting to manufacture phony currency and utilize it without detection, whilst the discriminative model can be compared to the police attempting to detect the counterfeit currency. In this game, competition forces both teams to develop their procedures until the counterfeits are indistinguishable from the original products. Both the generative and discriminative models are realized as multilayer perceptrons [11].

$$\min_G \max_D V(D,G) = E_{x\sim p(x)}[\log D(x)] + E_{z\sim p(z)}[\log(1 - D(G(z)))] \tag{3}$$

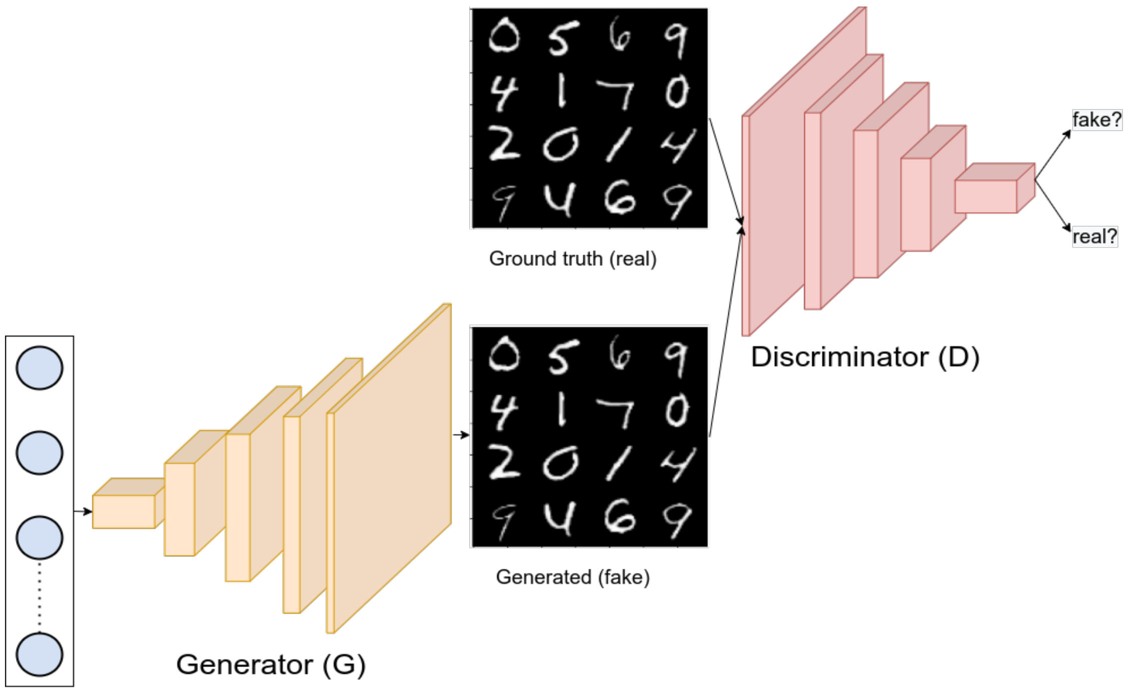


Figure 5: GAN architecture

3.2 Coupled generative adversarial networks (CoGAN)

Deep neural networks are supposed to learn a hierarchical feature representation, which is the basis for the CoGAN architecture. It forces the GANs to decode the high-level semantics in the same manner by requiring the layers that do so to share the weights. In order to confuse the relevant discriminative models, the layers that decode low-level features then translate the shared representation to images in different domains [9].

$$\min_{g_1,g_2}\max_{d_1,d_2} V(d_1,d_2,g_1,g_2) = E_{x_1\sim p(x_1)}[\log d_1(x)] + E_{z\sim p(z)}[\log(1-d_1(g_1(z)))]$$
$$+ E_{x_2\sim p(x_2)}[\log d_2(x)] + E_{z\sim p(z)}[\log(1-d_2(g_2(z)))]$$

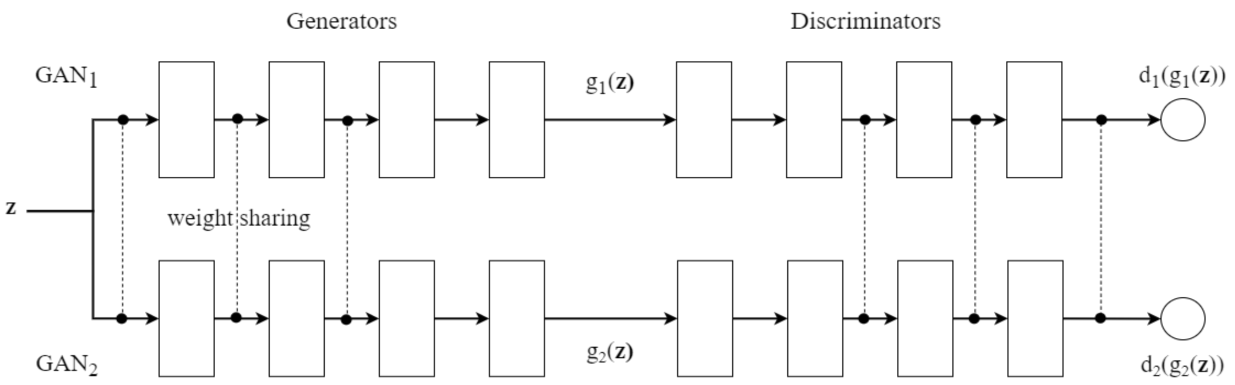


Figure 6: CoGAN architecture

There are two teams in the game, each with two players. The generative models collaborate to synthesize a pair of images from two separate domains in order to confuse the discriminative models. The discriminative models attempt to distinguish images obtained from the training data distribution in the relevant domains from those drawn from the generative models. The weight-sharing constraint establishes collaboration among players on the same team. CoGAN, like GAN, may be trained via back propagation with an alternating gradient update.

Generative models			
Layer	Domain 1	Domain 2	Shared?
1	FCONV(N1024,K4x4,S1),BN,PReLU	FCONV(N1024,K4x4,S1),BN,PReLU	Yes
2	FCONV(N512,K3x3,S2),BN,PReLU	FCONV(N512,K3x3,S2),BN,PReLU	Yes
3	FCONV(N256,K3x3,S2),BN,PReLU	FCONV(N256,K3x3,S2),BN,PReLU	Yes
4	FCONV(N128,K3x3,S2),BN,PReLU	FCONV(N128,K3x3,S2),BN,PReLU	Yes
5	FCONV(N1,K6x6,S1),Tanh	FCONV(N1,K6x6,S1),Tanh	No

Discriminative models			
Layer	Domain 1	Domain 2	Shared?
1	CONV(N20,K5x5,S1),Pool-MAX2	CONV(N20,K5x5,S1),Pool-MAX2	No
2	CONV(N50,K5x5,S1),Pool-MAX2	CONV(N50,K5x5,S1),Pool-MAX2	Yes
3	FC(N500),PReLU	FC(N500),PReLU	Yes
4	FC(N1),Sigmoid	FC(N1),Sigmoid	Yes

Table 2: Coupled GAN network

4 Results

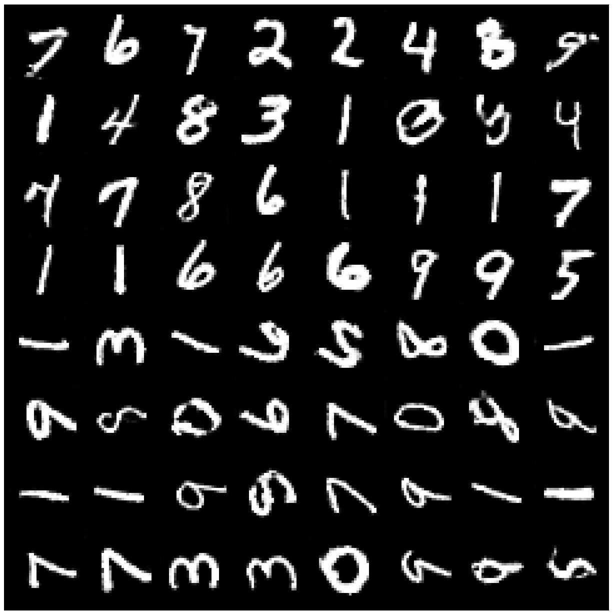


Figure 7: The generative images for original and 90-rotated cases

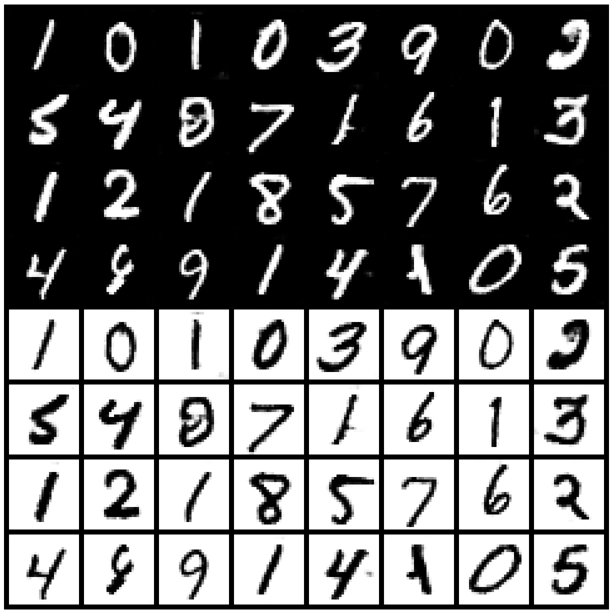


Figure 8: The generative images for original and negative cases

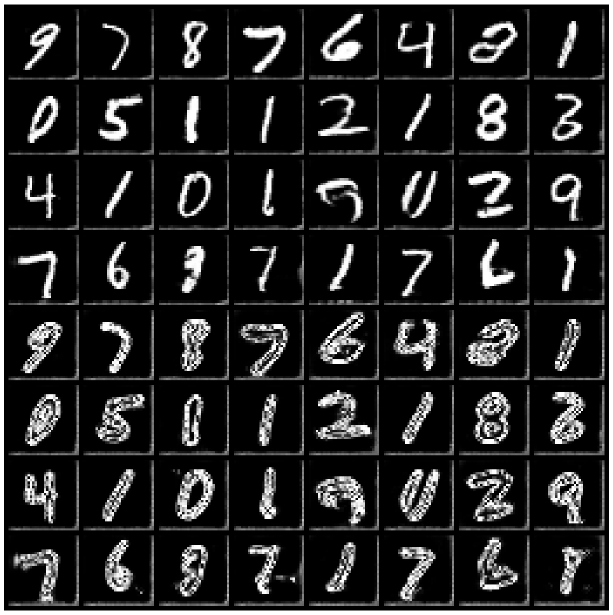


Figure 9: The generative images for original and edge cases

5 Conclusion

The generated images for the rotated and inverted scenarios are quite good and clear, as illustrated in Figs. 7 and 8. While the outcomes in the edge images are a little blurry. The explanation could be due to the technique used to detect edges in the original dataset.
This homework provides us with an understanding of GAN as well as the ability to use the CoGAN framework to train a joint distribution of multi-domain images. By enforcing a simple weight-sharing constraint to the layers that are responsible for decoding abstract semantics, the CoGAN learnt the combined distribution of images by simply drawing samples from the marginal distributions individually.

References

[1] M. Sanjeevi, *Mnist-exploration to execution.* [Online]. Available: https://medium.com/datadl-ai/mnist-exploration-to-execution-25136ca00570.

[2] Y. LeCun, *The mnist database of handwritten digits.* [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[3] K. Erdem, *T-sne clearly explained.* [Online]. Available: https://towardsdatascience.com/t-sne-clearly-explained-d84c537f53a.

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[5] K. O'Shea and R. Nash, *An introduction to convolutional neural networks*, 2015. arXiv: 1511.08458 [cs.NE].

[6] A. Anwar, *What is transposed convolutional layer?* [Online]. Available: https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11.

[7] N. Bjorck, C. P. Gomes, B. Selman, and K. Q. Weinberger, "Understanding batch normalization," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/36072923bfc3cf47745d704feb489480-Paper.pdf.

[8] S. S. Siddharth Sharma and A. Athaiya, "Activation functions in neural networks," *International Journal of Engineering Applied Sciences and Technology*, vol. 4, pp. 310–316, 2020.

[9] M.-Y. Liu and O. Tuzel, *Coupled generative adversarial networks*, 2016. DOI: 10.48550/ARXIV.1606.07536. [Online]. Available: https://arxiv.org/abs/1606.07536.

[10] E. Linder-Norén, *Pytorch-gan*, 2021. [Online]. Available: https://github.com/eriklindernoren/PyTorch-GAN.

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, *Generative adversarial networks*, 2014. DOI: 10.48550/ARXIV.1406.2661. [Online]. Available: https://arxiv.org/abs/1406.2661.