Proceedings of the International Conference on Inventive Computation Technologies (ICICT-2018)
IEEE Xplore Part Number: CFP18F70-ART; ISBN:978-1-5386-4985-5

Auxiliary Conditional Generative Adversarial Networks for Image Data Set Augmentation

Kalpana Devi Bai. Mudavathu
Research Scholar
Department of
Computer Science & Engineering
Acharya Nagarjuna University
dkalpananaik@gmail.com

Dr. M. V. P. Chandra Sekhara Rao
Professor
Department of
Computer Science & Engineering
RVR & JC College of Engineering, Guntur
manukondach@gmail.com

Dr. K. V. Ramana
Professor
Department of
Computer Science & Engineering
JNTUK Kakinada
vamsivihar@gmail.com

***Abstract*—Adversarial models have been widely used for data generation and classification in the fields of Computer Vision and Artificial Intelligence. These adversarial models are defined over a framework in neural networks called Generative Adversarial Networks. In this paper, we use auxiliary conditional generative models which are special kinds of GANs employing label conditioning that result in newly generated images exhibiting global coherence. This conditional version of generative models is constructed by feeding data that we wish to condition on generator network and discriminator network in a GAN. The analysis has experimented on a high-resolution dataset called FMNIST across 60,000 samples of training images with reshaped image resolution size of 28*28. The following procedure is used for image dataset augmentation which improves the accuracy of image classifiers/segmentation techniques.**

Keywords: Generative Adversarial Networks, Convolutional Neural Networks, Dataset Augmentation, Probabilistic Computation, Neural Networks.

I. Introduction

Neural Networks are widely employed for solving real word problems. These neural networks are inspired and imitated from the working of human brains. In the fields of image processing for image classification, segmentation or argumentation we use several kinds of the neural network which include perceptrons networks, multi-layer neural networks, convolutional neural networks. These are widely developed and enhanced in recent years which are able to give until highest possible accuracy and efficiency for predictions.

We employ different kinds of neural network architectures for different kinds of problems which are called neural network models. Different models give different predictions. These models employ several kinds of hyperparameters for model metric analysis. The main hyperparameters the neural network depends are a loss, optimiser, and activation function between the network layers. Most of the image classifiers use convolutional neural networks for object identification and predictions. GANs are special kinds which are used to create new images. This data augmentation can be employed for improving the image classifiers accuracies by adding new images to the training dataset which is achieved by GANS.

Generative Adversarial Networks are widely employed these days for image generation and classification techniques. These are alternative frameworks which are implemented in Neural Networks based on Generative Models. These GANs work like a game with two players, this game is also known as a zero-sum game. The two players in this game are two neural network architectures one is known as a discriminator and the other is known as a generator, The job of the generator is to add randomness to the training dataset using probabilistic functions. While discriminator identifies if the generated image is a fake image or a real image.

In this work, we use auxiliary conditional generative adversarial networks improving image classifier accuracy. These are special kinds of GANs which widely employ label conditioning. These are proposed by Augustus Odena et al. in his work Conditional Image Synthesis with Auxiliary classifier GANs [1].

II. Related Work

GANs utilises two neural networks trained in opposition to one another. These were proposed by Ian Goodfellow et al. in his research Generative Adversarial Networks[2] in the year 2008. In the first network which is a generator G, it takes a random noise as an input and outputs a fake image X_fake = G(z). While the discriminator takes a random sample of training sample of an image that is generated by the generator and outputs a probabilistic distribution P(S—X) = D(X) over a set of possible image sources. Both D(Discriminator) and G(Generator) are trained simultaneously and the hyperparameters are adjusted based on optimising the loss and varying the randomness. With the error and losses that are generated by the discriminator, the randomness in the generator is updated so that generated images are realistic. Mathematically the log likelihood or the rectified randomness that is updated with respect to the discriminator is given by

$$L = E[logP(S = real|X_{real})] + E[logP(S = fake|X_{fake})] \quad \text{... [1]}$$

We augment the GANs using side information by sending training samples to both generator and discriminator. These samples can be of any forms basically we use labels of images in the fields of computer vision using the conditional samples.

These can be of any type of either of types one too many samples or one to one samples or one to many samples based on the features that are passed for the network [3].

One of the main most common problems that are widely addressed by Generative Adversarial Networks is when there is a huge dataset the leverage additional modularities for better predictions. Few examples include vector representation for labels in which geometrical relations are semantically meaningful here the prediction errors which are generated by the neural network are generalised to labels by using simple linear mapping from image label space to word representation latency for improving the classification performance[3].

III. Auxiliary Conditional Generative Adveserial Nets

A. Generative Adveserial Models

The two main factors that are needed for training a GAN are the neural network architectures of discriminator and generator. These follow a non-linear mapping function, either a simple perceptron model or a convolutional model. We need to identify two distributions which define the efficiency of the training, primarily the generator distribution which is iterated over several epochs is mathematically defined over a noise distribution P(Z) to data space as G(z,0). The discriminator D(x,0_d) outputs a scalar representing the probability that x that is estimated from the training data.

Below is the diagrammatic representation of a GAN network, the mostly used activation function in the discriminator network is a sigmoid function[4]. This is used for smoothening and final predictions of the image classifier. Real data is sent into discriminator which will be a convolutional neural network model.

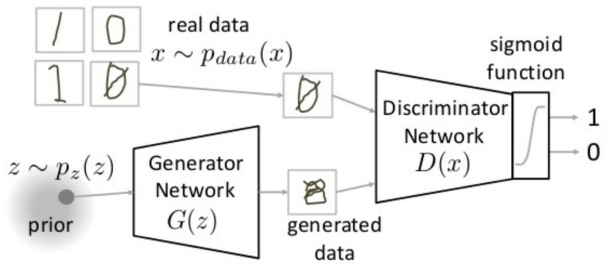


Fig. 1. GAN Network

B. Auxiliary Conditional Adveserial Nets

GANs when extended with some extra information mostly of auxiliary information(class labels or data of other modularities). This information is fed into the network by adding an additional layer. Once the new layer is added to both the generator and discriminatory network we can perform conditioning by feeding in this labelled data. The noise is usually a hidden representation or a layer in the generator network and the adversarial training framework allows for considerable composition[10]. If we consider a discriminator with x as an input x and y as a discriminative function which defines the fakeness or realness of image in a multilayer perceptron then mathematically the objective is calculated as

$$min_G max_D V(D,G) = E_{\infty \sim P_d ata(x)}[logD(x|y)]$$
$$+E_{z \sim P_z(z)}[log(1 - D(G(z|y)))]$$

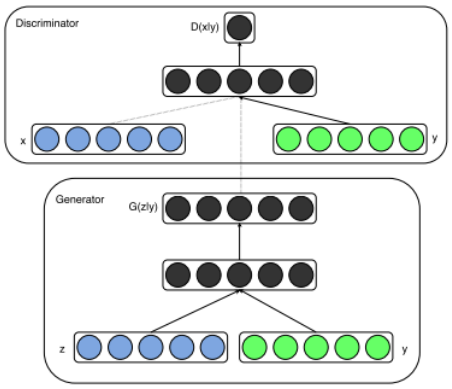


Fig. 2. Conditional Adveserial Neural Network[3

IV. State of Art

A. Generating Random Distributions from GANs

In this section, now lets generate a distribution to add randomness to the training data by the generator. When we consider adversarial models the generator G(Z) should be able to create the randomness and the discriminator should be able to minimise the noise with the number of iterations. In this case, the generator will be able to create a classifier using a sigmoid function and it returns an output of a new image with some probability added to input images/data as a new training data.

B. Implementation of Generator Network

One more main objective of the discriminator network is that the probability of checking the samples should be greater than 0.5 of the newly generated probability distributions. In a generator, we take three main attributes the input, noise and the random probability generator. The activation function used in this neural network architecture is a leaky relu activation to restrict the training values. This leaky RELU is a special kind of activation function also know as a dying relu activation function. We use a Binary Cross entropy function which is given by the equation. The batch sizes and input dimensions can be based on the data which we train with the neural network.

Initially, we set up the dimensions of the training data by reshaping them and sending them into dense and dropout

layers of the neural network. Dropout layers in neural networks are used for reducing the overfitting of the training procedures. Once the training data is resized as per the dimensions for the first layer of the generator network the input is sent into a convolutional transpose network with few parameters. These parameters include strides(which are used for multiplication/kernel skipping operation), padding(extra dimensions added to the input images so that the dimensions of the output images stay as the same). The momentum factor the network is taken as 0.99. This is an ideal metric to increase the efficiency and accuracy of the training process. In this implemented GAN we considered the kernel size of the convolutional operator to be 5. Keeping the matrix with constant 5 rows and 5 columns of operations. We use four similar convolutional layers which are dropout layer and batch normalisation layers. At last, once the input data is sent the output will be a single dimension which will be the data that is generated with a complete forward propagation.

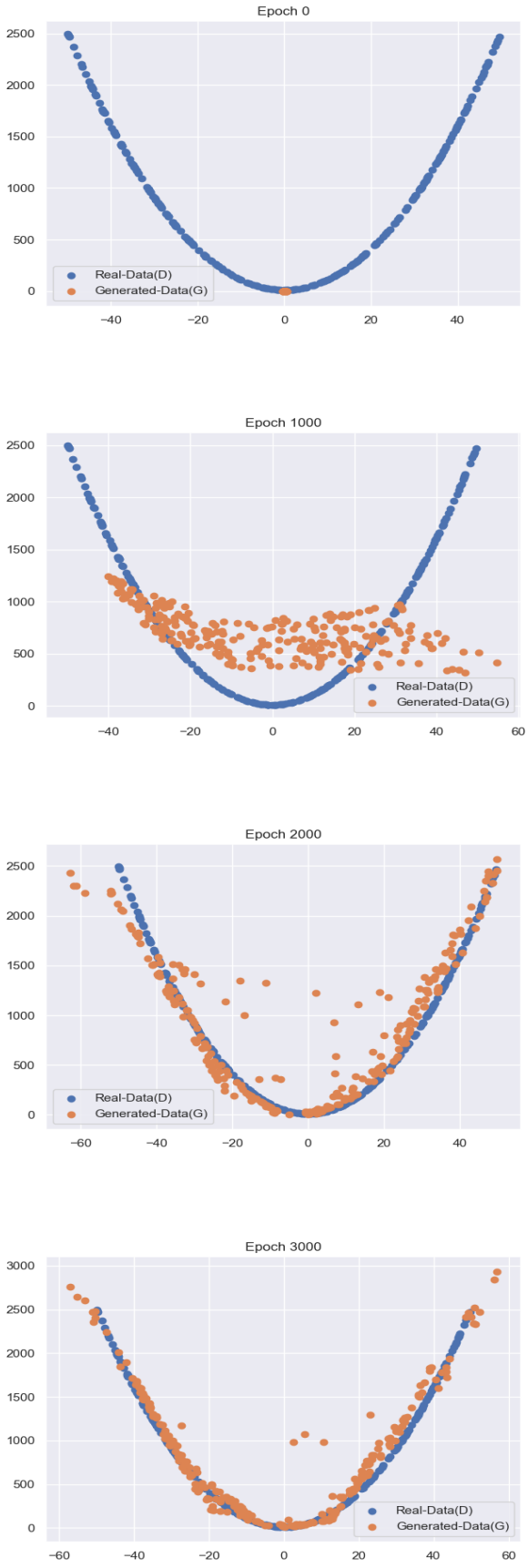
$$H(p,q) = E_p[-logq] = H(p) + D_K L(p||q)$$

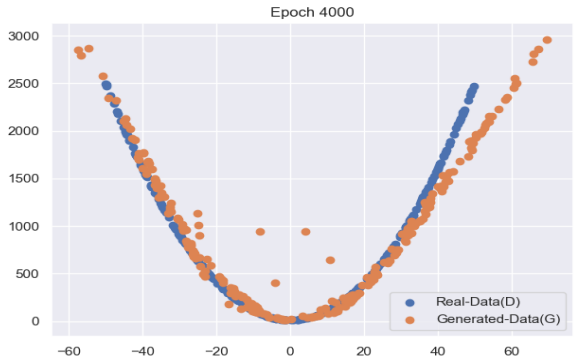
C. Implementation of Discriminator Network

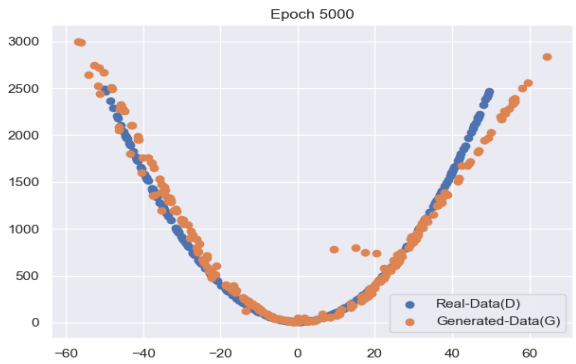
In this discriminator network, we use a 4 layer convolutional neural network each followed by dropout layer and a leaky RELU activation function. Each convolutional layers reduces the dimensions of the input data, these are reverted back using fully connected layers which are implemented in the last two layers of the neural network. The sigmoid function is used for interpreting the input image with the considered training data[6]. End output layer of the generator network is connected by a sigmoid function preceded by few flattening layers. Flattening Layers in neural networks are used to bring back the dimensions of original data since once we apply convolutional operations the size of the dimensions changes with every kernel. This output layer returns whether the generated image a real or a fake image.

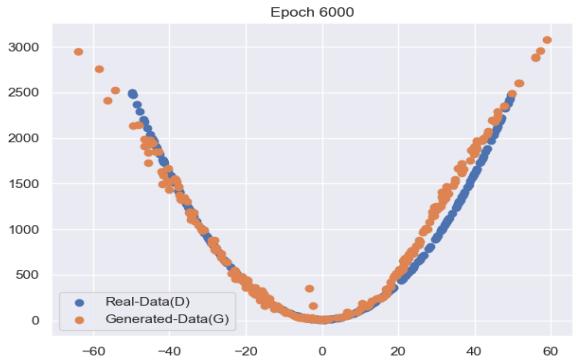
V. Training the GAN

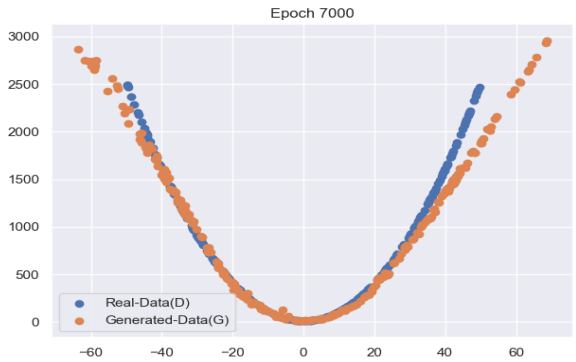
We need to train the gan simultaneously so that the coordination of both generator and discriminator are uniform. Initially, we feed random values to the generator and which lets us create the new data. The discriminatory should be run parallelly [7] where its should manage the losses as well as inheriting the training process from the previous iterated steps. The total number of iteration for the entire GAN architecture are set to be 10,000 steps. If the generator loss is less than the discriminator loss we consider the generator training to be false, if the discriminator loss is less than the generator loss then we stop the discriminator training. We now run the generator network and discriminator network simultaneously. Below is the generated data for a randomly created a dataset, for every iteration new data is generated and retrained for generating the new data with a maximum probability of realness.

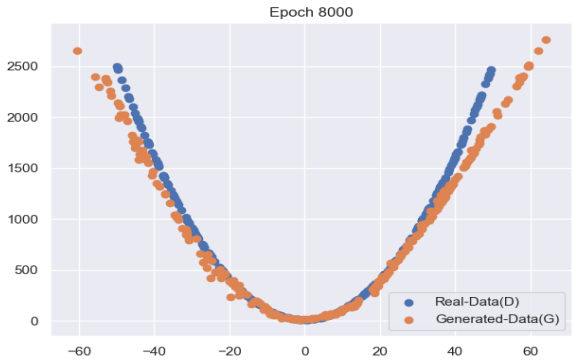


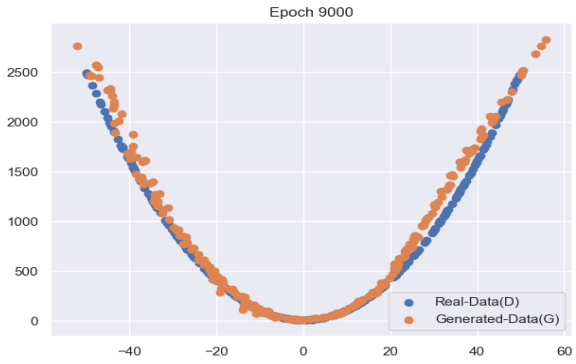












Below are the loss values that are iterated by the discriminator and the generator network for every 1000 epochs.

Iterations: 0000	Discriminator loss: 0.8340	Generator loss: 0.7104
Iterations: 1000	Discriminator loss: 0.8630	Generator loss: 0.6736
Iterations: 2000	Discriminator loss: 1.0807	Generator loss: 0.6518
Iterations: 3000	Discriminator loss: 1.3804	Generator loss: 0.6879
Iterations: 4000	Discriminator loss: 1.3716	Generator loss: 0.7298
Iterations: 5000	Discriminator loss: 1.3645	Generator loss: 0.6561
Iterations: 6000	Discriminator loss: 1.3575	Generator loss: 0.6952
Iterations: 7000	Discriminator loss: 1.3085	Generator loss: 0.5756
Iterations: 8000	Discriminator loss: 1.4518	Generator loss: 0.7381
Iterations: 9000	Discriminator loss: 1.2495	Generator loss: 0.5534
Iterations: 10000	Discriminator loss: 1.3530	Generator loss: 0.8886

Loss graph that is produced by the GAN.

VI. Image Dataset Augmentation on FMNIST Data

Fashion MNIST is a labelled dataset image of collected fashion items which includes clothing and few accessories. The dataset has over 70,000 fashion images each of similar shape 28*28 concerning resolution. These have ten classes which are T-shirt/top, Trouser, Pullover Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot. We apply AC Gans for generating a new FMNIST image which can be used for dataset augmentation. We split the dataset into training and testing traditionally with 70%, 30% ratio respectively. Below are the few samples of MNIST dataset.

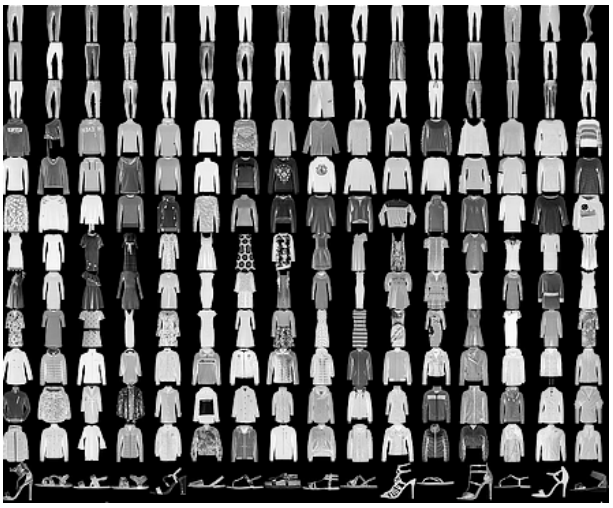


Fig. 3. Few samples of MNIST dataset

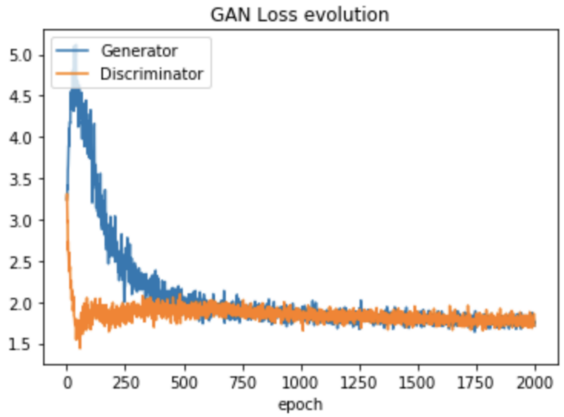
VII. Proposed Auxiliary Conditional Generative Adversarial Networks

As discussed in the section Auxiliary Conditional Adveserial Nets they are GANs with few auxiliary information with few conditional constraints. The proposed solution for the neural network architecture includes building two neural network architectures. These two neural network architecture are sequential in geometry. In the generator, the first layer takes in all the dimensions of all input features of FMNIST dataset. Here, as the resolution of each image is 28*28 the total number of neurons in the initial layer are limited to 784. We invert the image to normalise every image, and the layer followed by the input is the batch normalisation layer which increases the efficiency and accuracy of the training [5]. Next, we have the upsampling layer followed by the first convolution layer of kernel size and same image padding. The activation function used after the first convolutional layer is relu activation which softens the output that is produced by the layer. The activation function is followed by batch normalisation and upsampling with two dimensions. The same layer is added for two more convolutional layers followed by activation and batch normalisation.

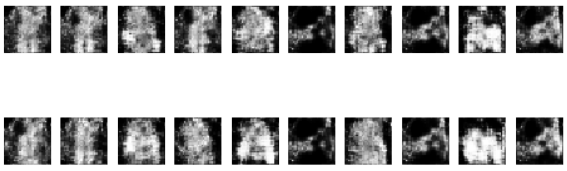
The discriminator is an image classifier which comprises a convolutional neural network. The proposed discriminator architecture consists of four convolutional architecture which is followed by leaky relu activation function with learning rate 0.2 and dropout layer. Dropout layer is used for overfitting the model[9]. This is achieved by ignoring a few neurons in the training process. The last layer is a flattening layer which brings back dimensions of the original training image.

VIII. Results and Evaluation

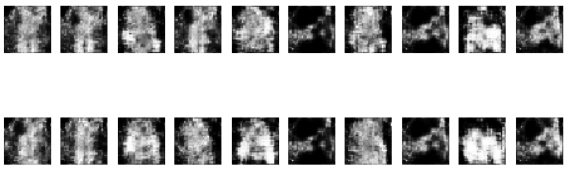
The proposed adversarial model was applied to the described dataset which is FMNIST. The adversarial network is trained for over 1400 epochs with batch size 100 and sample iteration rate of 100. As we see the results generated by the first 800 epochs have greater noise when compared to images that are produced after 800 images. This is because the generator adjusts the random noise based on the loss that is regulated by the discriminator network. The loss generated by the discriminator and generated network is visualised below found concerning the number of epochs the model is trained. It can be seen that in the first epochs the generator has substantial loss values, while the discriminator has lower loss values, meaning that the generator is not being able to trick the discriminator yet. The loss for the generator decreases after some epochs, arriving at a stable point after period epoch 700.



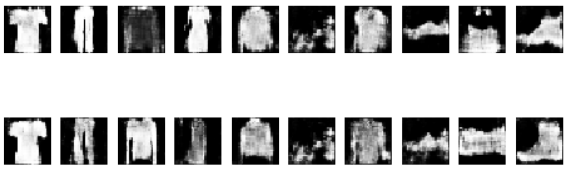
Below are the results of the images that are generated by auxiliary conditional gans of newly created accessories.



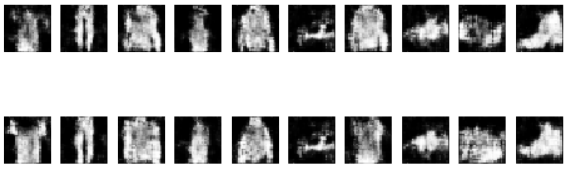
Epoch: 0



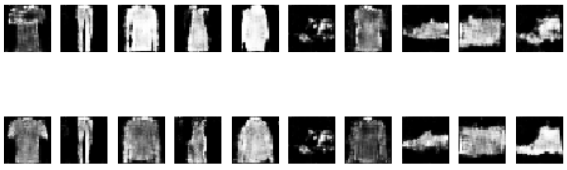
Epoch: 200



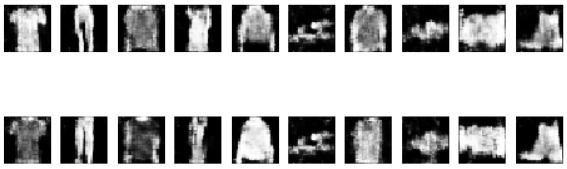
Epoch: 1000



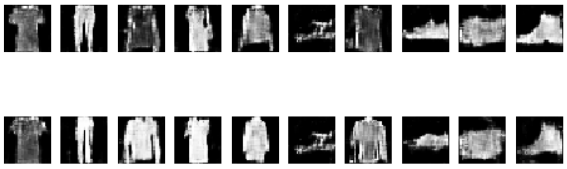
Epoch: 400



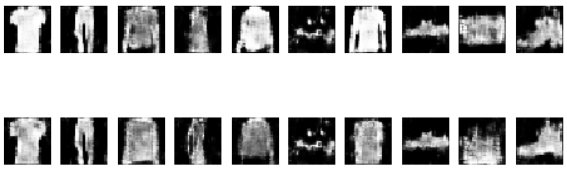
Epoch: 1200



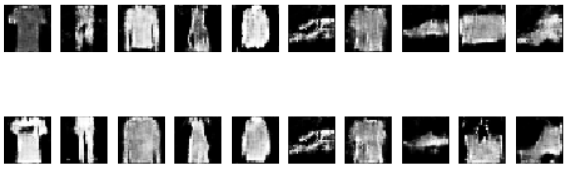
Epoch: 600

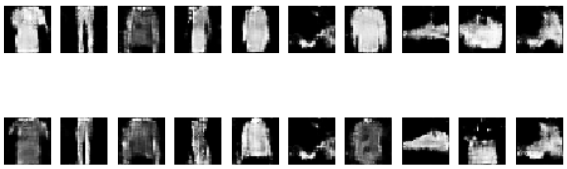


Epoch: 1200



Epoch: 800





Epoch: 1400

Once the neural network generates the new images and is labelled as FMNIST images by the discriminator network, they are added to the dataset. This is how we increase the dataset size by augmenting the generated images into a real dataset using adversarial systems. With this Convolution, neural networks can perform well in image classification and segmentation problems. Most of the convolutional architectures fail to train the model properly because of inconsistent datasets, with this solution CNN's can perform in a better way. Below are the newly generated images which are produced by AC GAN.

IX. Conclusion

In this work, we proposed a new modified auxiliary conditional adversarial neural network and wrote the implementation in the Python framework called keras. The proposed neural network architecture is applied to the FMNIST dataset where we tried generating new FMNIST datasets and added to the training datasets. However, to use the proposed architecture for other datasets, we need to modify the architecture input and output dimensions as the shapes and labels of the input images changes. The model gave an accuracy of 88% in generating the new FMNIST images which are used for image dataset augmentation. Further work includes image dataset augmentation with spherical GANS.

References

[1] Conditional Image Synthesis With Auxiliary Classifier GANs Augustus Odena, Christopher Olah, Jonathon Shlens arXiv:1610.09585
[2] Generative Adversarial Networks Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio arXiv:1406.2661
[3] Conditional Generative Adversarial Nets Mehdi Mirza, Simon Osindero arXiv:1411.1784
[4] Searching for Activation Functions Prajit Ramachandran, Barret Zoph, Quoc V. Le arXiv:1710.05941v2
[5] ImageNet Classification with Deep Convolutional Neural Networks Alex Krizhevsky Ilya Sutskever Geoffrey E. Hinton : Advances in Neural Information Processing Systems 25 (NIPS 2012)
[6] Adversarial Learning with Local Coordinate Coding ICML2018 Jiezhang CaoYong GuoQingyao WuChunhua ShenJunzhou Huang Mingkui Tan
[7] Understanding Convolutional Neural Networks Jayanth Koushik arXiv:1605.09081
[8] Automatically Designing CNN Architectures Using Genetic Algorithm for Image Classification Yanan Sun, Bing Xue, Mengjie Zhang, Gary G. Yen arXiv:1808.03818
[9] Some Theoretical Properties of GANs G. Biau (LPSM), B. Cadre (ENS Rennes), M. Sangnier (LPSM), U. Tanielian (LPSM) arXiv:1803.07819
[10] Image-to-Image Translation with Conditional Adversarial Networks Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros CVF, arXiv:1611.07004, 2016