Faculty of Mathematics
and Computer Science

Machine learning course (ML)

Creating images using Generative adversarial networks

Grigore Mihai Alin

Department of Computer Science, Babeş-Bolyai University
1, M. Kogalniceanu Street, 400084, Cluj-Napoca, Romania
E-mail: griggmihai@yahoo.com

Abstract

The goal of this paper is not to present a single algorithm or method, but to take theoretical steps to fully understand the training dynamics of generative adversarial networks. To support our theoretical analysis, we perform targeted experiments to test our assumptions, illustrate our claims, and quantify the phenomena. In this paper we will present the loss function we will present training methods for GAN and maxout activation. We will also talk about Inception score, Frechet Inception Distance. We will provide examples and suggestive images for a better understanding. Moreover, we show that the corresponding optimization problem is sound, and provide an extensive theoretical work highlighting the deep connections to various distances between distributions. Start with the basics of how GANs work and gradually learn more sophisticated techniques that will improve your models from basic GANs to advanced progressive growing GANs. I will introduce a whole range of Deep Learning concepts with a proper discussion of the mathematics behind the modern models.

Keywords: Generative adversarial networks, Loss function, Maxout Activation, Inception Score, Multi-way Loss Function, Frechet Inception Score, Convolutional GAN

1. Introduction

Generative adversarial networks(GAN) are very highly researched and application space of GANs is ever increasing. About the terminology of GAN (we call it generative adversarial networks), the generative part of this method, as the name suggests, it generates data, which means it's actually creating fake data. In fact the primary application of GANs is generating fake data that looks realistic enough. The adversarial part comes from the point that the generator and something called discriminator are competing to win. Discriminator think of it as our regular machine learning algorithm (For example, we try to classify a pixel for semantic segmentation or if it is an object like you are trying to classify cat as cat or dog as dog, so that is discriminator. He trying to discriminate). Generator and discriminator competing to win meaning, as I mentioned, generator is trying to fake the data and the discriminator is trying that's fake data. The generator gets better and better in the next epoch while the discriminator also gets better. Eventually they comes a point where the generator is good at generating fake images that look realistic enough where

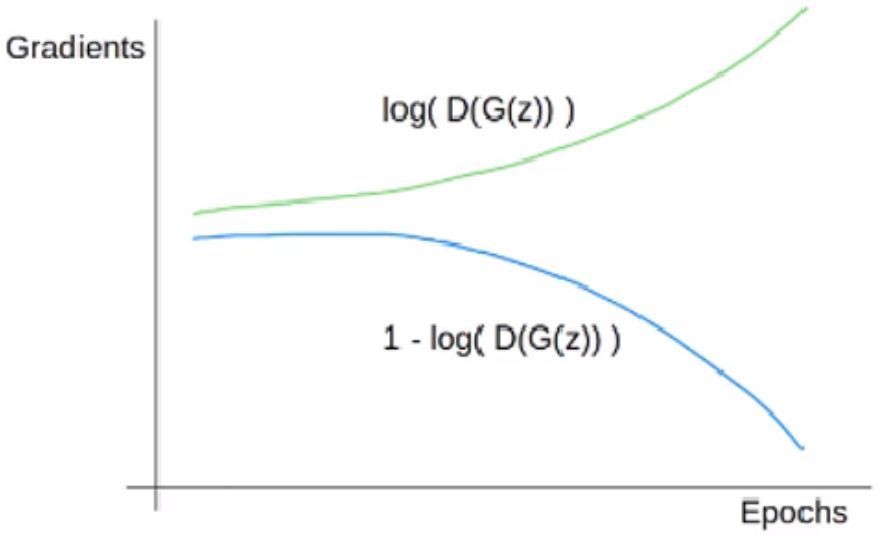
the discriminator cannot discriminate, meaning the probability of that image being a fake or real would be 0.5. The networks part can be deep convolutional or fully connected (Dense only). [4]

2. Working of GAN Loss Function

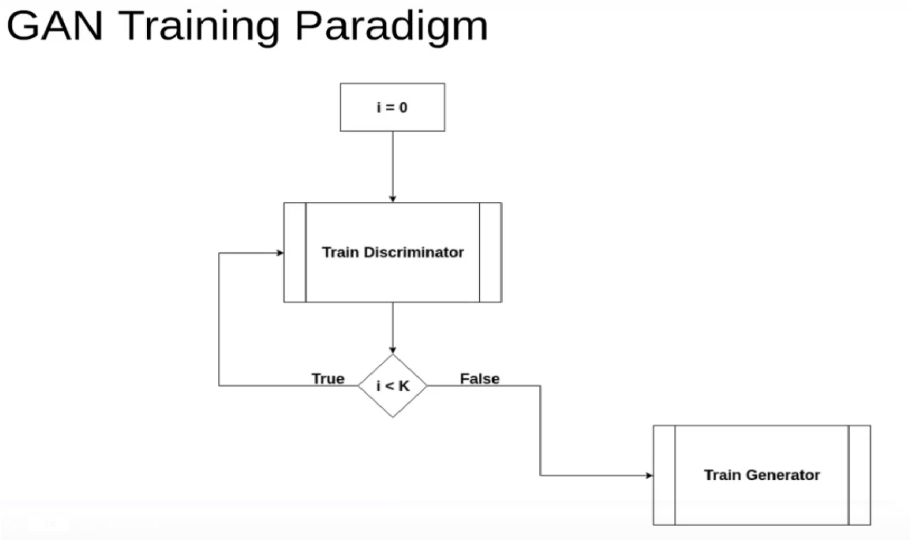
In this section we will working on loss function. We shall discuss the min max game, the training pattern of gains max out activation functions, in a quick review of gain evaluation methods. Generative adversarial networks can be formulated mathematically as an min max game. The generator is trying to minimize the number of symbols that a discriminator detects. On the other hand the discriminator is trying to maximize the number of fake symbols it detects.

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{x\sim p_z(z)}[log(1 - D(G(z)))] \text{ [4] [3] [6]}$$

In this equation x denotes real samples and Z denotes noise. So D of X is discriminator output for real samples and D of G of Z is discriminator output for generated samples. Since the output for generated samples is expected to be zero, we subtract it from 1 to make the formula correct. It is however found in practice that the discriminator can easily find out fake symbols at the beginning of training. Hence the gradients are almost zero and the generator cannot update it weights to match the efficiency of the discriminator. Because of this gradient saturation problem is solved by changing the objective of the generator, that is by making the generator maximize the probability that it's output is not detected by the discriminator. This way the generator has a better change of learning. In the graph below, the blue curve represents the gradients in the case of min max formulation, which is almost straight for the first reigning iterations. The green curve represents the new gradient which of higher values.



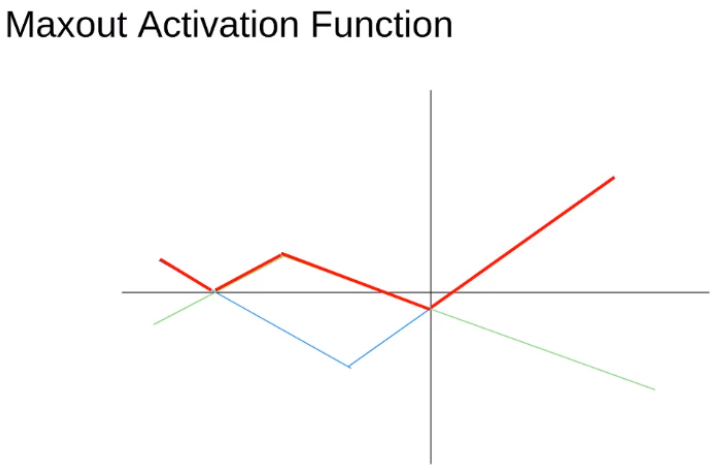
The discriminator will learn to mark all generator outputs SFE. Both the generator and the discriminator are trained in tandem. The discriminator is trained for first four key steps where key is an integer. In the original paper key was equal to one but it is found in practice that K equals four yields much better results. That is the discriminator is trained for four steps and then the generator is trained once. [6]



The training ends when the probability density function of the generator is equal to that of the original data set. All of the following mathematical equations are equivalent:

$$Pdata = Pmodel <=> D^* = Pdata/(Pdata + Pmodel) <=> D(X) = D(G(z)) = 1/2$$ [5] [4]

Maxout Activation Function. Maxout is a more robust and avoids overfitting.It does not also suffer from vanishing RELU problem. The downside however is that the number of parameters is doubled for every single neuron which leads to a high number of parameters.



The red curve represents the output of a maxout activation which is giving to linear functions. In this case the green and the blue lines it is also called Beast Y linear activation. This has the advantage over RELU since it has steeper slopes which allow for better gradient flow in the network.
The evaluation of generated models: Parzen Window, Euclidean Distance, Maximum Likelihood Estimation and Inception Score. There are a few ways to evaluate the results of GANs. Some of these methods also work for generating models in general but each has it's own downside. For example Parzen Window which is a kernel based density estimator fails to capture high dimensional features such as of images. Second, the Euclidean distance. This one is

pruned to overfititing sensor generated images don't have to be exactly the same like the original ones. Third we have Maximum likelihood estimation which in most generated models is intractable. That's another way to saying that it cannot be solved analytically. Maximum Likelihood Estimation works by giving the wrong benchmark. The downside is apparent when it gives it wrong benchmark for a lower bound. For example, if we have one model and that one has a lower bound on it's value and this lower bound is bot used on the other model which we are comparing. In this case we will not be comparing an apple to an apple. The last evaluation, the Inception Score, which can be misleading as well because it is relevant for images similar to imagine it dataset but irrelevant otherwise. [4]

3. GAN Evaluation Metrics. Inception Score.

The inception score takes it's name after the inception architecture proposed by Google in the year 2014. One of the big on hearing architectures and convolution of neural networks that performed a high score on the imaging it classification problem. The imaging it has thousand classes and the network is designed so that it has thousand of good neurons for each label of the classes in the dataset. The architecture of the inception of score allows for very deep networks by having auxiliary networks additional output layers that are put in the intermediate layers. This way they provide gradient flow in the shallow layers of the network meaning that because of the vanishing gradient problem inherent in very deep neural networks. The designers of the inception architecture designed an auxiliary output that's a secondary output not actually essentially one for the output and this one is placed not at the end of the network but at intermediate places of the network. And then when the loss is calculated on the output of these intermediate layers there will be a gradient flow in the network that finds it's way through to the very first layers of the network and hence it solves the vanishing gradient problem inherent in deep learning. The idea behind the inception is core that we are going to use a pretrained Inception network to evaluate the performance of the GAN model. [4]
How it works? The Inception score aims at providing a metric for two values. The first is whether the image is correctly classified by the Inception network. That means if we have a network, a GAN model that generates images, of let's say canines, we feed the image to the inception network which is pretrained on thousand classes of image it and these thousand causes, they do actually include images of canines and cats and so on. So the output of the network will be a classification at distribution overall of the Southern classes with a beak a very high value, a very high probability value add the label corresponding to the canine class. By reporting these value what we get is that we have a measure, an objective measure of the image quality that is very close very similar. This is very close and very similar to the evaluation of a human being, That means if given the same image to a human and given the same image to an inception network and calculating the score, the value whether the image is real or fake of a desired object it can AI, for example, will be very close. So that helps us in the design and evaluation of generative adversarial networks because we need an objective way to assess the quality of generated images. We had to look at every epoch when the images are generated or every ten epochs and then manually inspect the quality of the images. This is not feasible at larger scale and it cannot be very decisive in deciding the best model, so the must be a mechanical a mathematical term that help us better evaluate the performance of GAN's. On the other hand, we want the model the output a variety of classes, not necessarily as we done an amnesty number one or number two or number three, but we want it to have a uniform distribution equal number of samples generated from each class. So for each class in the dataset should have it's equal proportion to the total output of the images. This is called that intraclass diversity and this is also one of the qualities that is measured by the inception of score. [4]

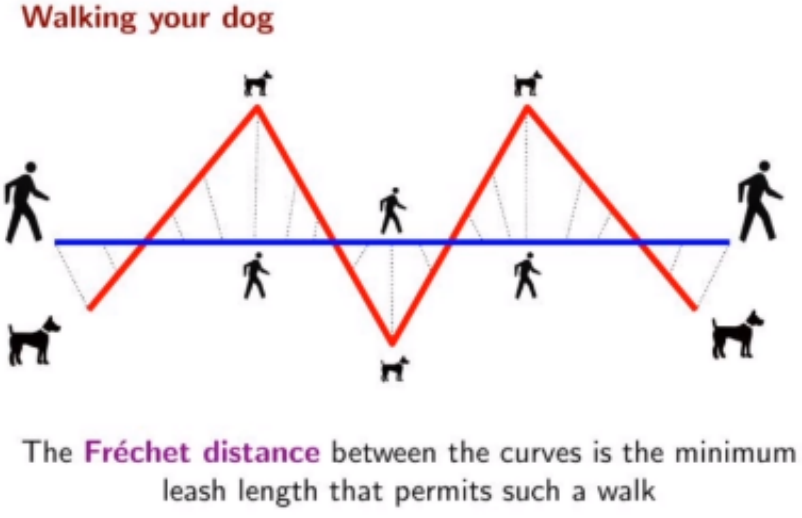
So as we can see in this picture, on the left side we have the kind of probability distribution that is given when we have a classifier and given an image and then we have a high probability at the label or the neuron corresponding to that class and this is how it looks like if we try to block the output of the soft max layer. On the right side we have a uniform distribution. It means if we have three classes: elephant, cat and dog then we want each one of them to have the same number of generated images done by the GAN model.
The Kullback Leibler divergence is a distance metric between two probability distributions. The higher the difference between the two probabilities the higher the Kullback Leibler divergence distance that more similar, the two probability distributions the lower the value of the Kullback Leibler are divergence distance. So we can benefit from this kind of metric to actually maximize the inception score because the inception score will be at the end, the mean or the average Kullback Leibler divergence between the two distributions. The distribution of the class Leibler given the generated images and then their average values over all the samples in the dataset or in the generated dataset for all the classes. So this way we have the conditional probability working on the horizontal axis and the probability of the class Leibler working on the vertical axis and then by giving them as to probability distributions to Kullback Leibler diversions we make sure that we get a very high distance, meaning that we always get this kind of distribution and we are getting a uniform distribution when the images are very diversified and different from each other for different labels.

$$IS(x) = exp(E_x[KL(p(y|x)||p(y))]) \text{ [1]}$$

The mathematical formulation is the mean of the Kullback Leibler divergence between two probabilities, probability of Y given X and the probability of Y. Y here stands for the label, the class label and existence for the generated images. So what we are trying to calculate is actually a distance. The further the distance between these two probabilities, the better the inception of score the more close they are each other, the lower the inception score and hence it is worse. The inception score helps us to measure the interest plus diversity but it still fails on the enter class diversity. [4]

4. GAN Evaluation Metrics. Frechet Inception Distance(FID)

In this section we will introduce the Frechet Inception Distance or FID score. The Frechet distance is a distance metric that is bet fit for Curves, polygons and it is perfect for pattern matching and shape matching. It doesn't require the two curves to be identical it allows them to be within a given approximation range from each other.



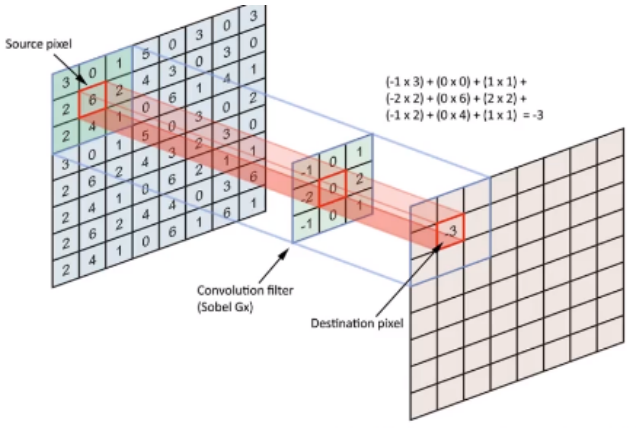
As an example of this, if you are going to walk your dog outside, you have a leash and the dog is on the leash. It is allowed it to be in the same path with you but it can also go on it's own both as long as the leash allows for it. The shortest leash that is going to allow you and your dog to walk on the same path on the same curve without diverging from each other is the minimum Frechet distance. This is the intuition behind the Frechet distance, the real intuition behind using it for generative adversarial networks and their evaluation. Even though we talked about Inception score in the previous section, it is found to be inferior to the Frechet Inception distance because it doesn't take into account the statistics of both the real image and the generated image. The Frechet distance or the Frechet Inception distance is designed to allow to study the mean and the variance of both vision rated images and the generated images and the real images using the Inception network. Unlike the Inception score which uses the final output of the inception network in the Frechet Inception distance, we are going to use the last layer of the second to last layer before the last output in the Inception network. That means we are using the inception network as a feature extractor. A very common concept in deep learning is called representation learning. It was found that in DB archtectures such as AlexNet, VGGNet, ResNet and Inception network some particular layers are valid as feature extractions because all of the convolution layers in the network, the learn the different features such as curves, corners, edges and so on of the input image and the they keep on accumulating this knowledge in a hierachical topology from the lower layer up to the higher layers in the network till we arrive at a flat in vector which is a valid feature representation that maximizes the entire class difference and minimizes the entire class difference. That meaning that images for the same object will have similar features and images from different objects will be very far from each other according to this feature representation. We can think of it as similar to BCE or principal component analysis for feature reduction, will you try to project the input features to different coordinates that try to maximize the difference between the features and allow us to have linear decision boundaries between the classes. So the idea of Frechet Inception distance that we are going to use the statistics of the mean and the variance of the features learned by an inception network on both the real images and the generated images. [4]
FID Score

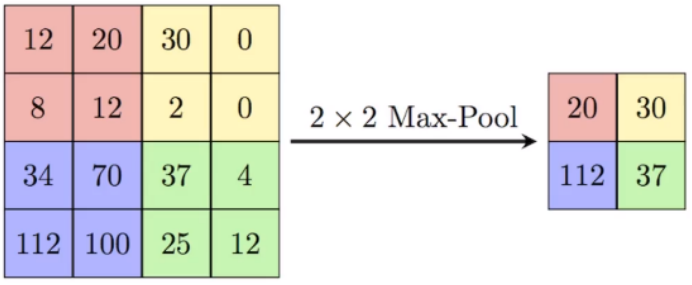
$$FID = \|\mu_r - \mu_g\|^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}) \text{ [1]}$$

We have the mean μ_r which is the mean of the real images and μ_g the mean of the generated images. We subtract them and then we have the square product of μ_r and μ_g difference and then we have a new operator Tr which is the crease operator. It allows us to take the sum along that the diagonal of a matrix and this operator is applied to the variance. The variance here is taken for the representation of the real images and the generated images. So because we know that, the variance matrix is the diagonal, we take the sum along these diagonal, then we take the product, Σ_r multiplied by Σ_g and then we take the square root multiply by 2 and substract from $\Sigma_r + \Sigma_g$. A very simple equation it gives us a distance, a Frechet distance between the generated images, the representation, and the real images, the representation, using the mean and the variance of the feature representation learned of the inception network.

5. Diving Deeper with a Deep Convolutional GAN

Convolution filters is a concept that has it's origins in the signal processing digital image processing and it carried on to be the cornestone of the modern deep learning revolution. The convolution filter is lid over every Bexhill of the source image and product is calculated to find out the new pixel value in that location. The values in the filter are initialized with random values and then the optimal values are learned along with the network parameters by back propagation. Unlike digital image processing where one designs at the values of the filter on his own, to perform a specific task such as blurring an image or extracting edges and convolution networks we allow the network to design it's own filters. It starts to learn useful features such as edges, corners and shapes. The opposite operation of convolution is the transpose convolution. The process is reversed and the resulting matrix is larger than the original one. For every pixel of the input, the filter is applied to produce a region equal to the size of the filter. Usually convolutions are followed by a max pooling layer to perform dimensionality reduction. For example, a max pooling of two by two results and shrinking every four pixels to only one pixel. The value of the new pixel is the maximum value in the original two by two region. Again, the max pooling has an inverse operation which goes sometimes by

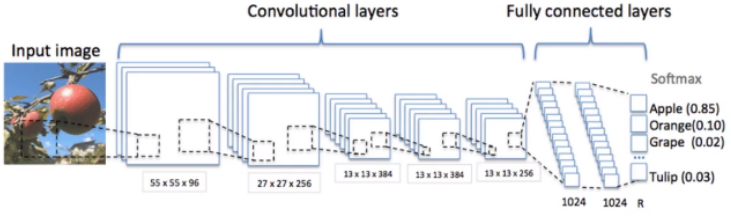




the name Max on pooling. The model has however to bookmark the location of the max values so that it can fill them back. That is while doing Max pooling the location of the max value is reserved, then when applying Max on pooling that location is filled and the others are set to zero. There is a variation of this approach which fills all the pixels with the same values. In basic GAN's we have used a Sigmoid Activation at the output layer to question the values between 0 and 1. In deep convolution GAN's it is recommended to use Tanh Activation. This will result in values squashed between negative 1 and plus 1. It was found empirically that this produces more appealing results. However we should not forget to scale the values of the training images to be in the same range. That is instead of zero to one. It should be also between negative 1 and plus 1.

$$\bar{x} = \frac{x}{127.5} - 1 \text{ [4]}$$

This equation does exactly that. We can verify that by substituting 255 for X and calculate the new x value and again substitute the value 0 for x and calculate once again.

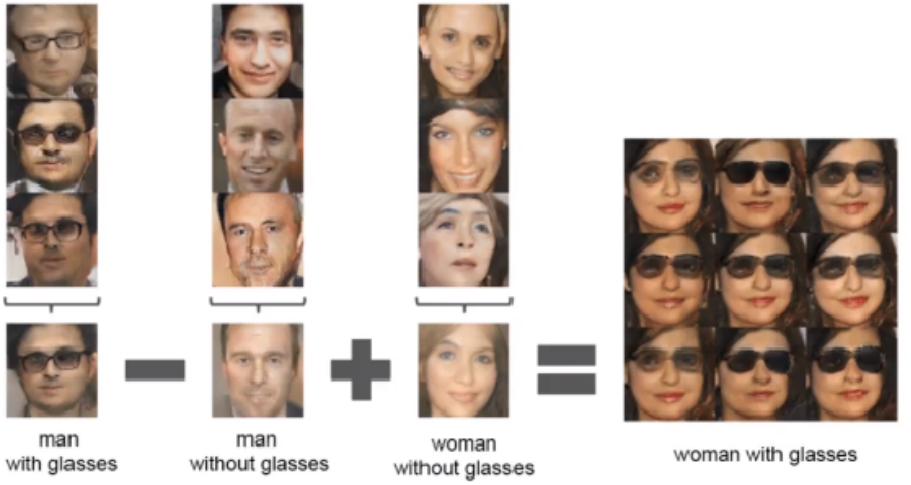


Let's take for example AlexNet architecture in which the network consists of convolutional blocks followed by fully

connected layers. The convolutional blocks function as feature extractors learning interested patterns such as shapes edges and corners. The convolutional blocks also learn these patterns while preserving the hierarchical topology of features. For example if the network is trained on crop the human faces, then it will learn that the eyes and the nose will be located inside a face and so on. The output of the convolutional block is called a feature map or a vector representation of the input. For example in AlexNet, the first dense layer of four thousand ninety six neurons will be a valid representation for image. The same idea extends to auto encoders where the output of the encoder service as a compressed version of the input. It turns out that the output of the discriminator in the DC GAN architecture serves the same purpose. That is the discriminator can learn useful features from unlabeled dataset in an unsupervised learning. This is quite useful in cases where there are not many labelled symbols available for training. [5] [4]
In a paper published by MCA love on learning feature vectors on word embedding buildings for natural language processing, it was found that interesting arithmeti like addition and subtraction a apply on the resulting vectors.

$$King - Man + Woman = Queen$$

The word the men can be substracted from the word King and then added to the word woman to give the result of a queen. That is a vector addition and subtraction on the learned representation. So is this applicable to GAN's? In fact it holds true for the representation learned by the discriminator of a DC GAN.

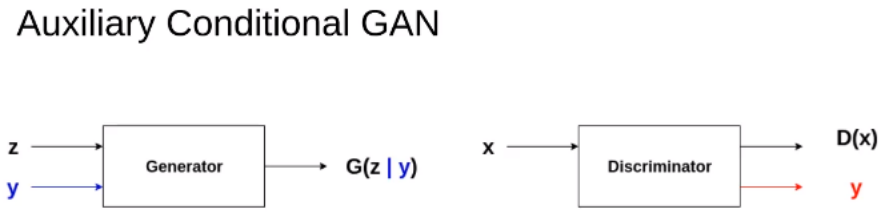


The authors have average three vector representations of three simple images in three categories. The first category was a man with glasses. The second was a man without glasses. The third was a woman without glasses. Then perform it the arithmetics and here you get a woman with glasses.
This is a very common technique to test the model's robustness by simply adding Gaussian noise to the inputs and observe the results. It was also found that the vector representation can be used to change the emotion on the resulting image. For example they subtract smiling faces from angry faces and then add the difference to an angry face to make it smile. [4]

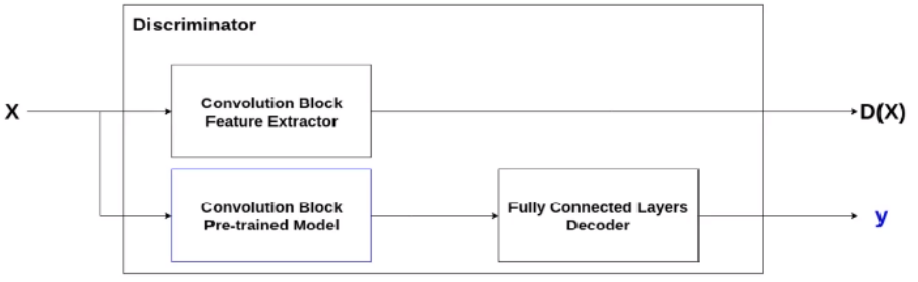
6. Multi-way Loss Function

In this section we will present the basics of auxiliary GAN's or AC Gan's. The first is the generation of appealing high resolution images. The second is the generation of different symbols for the same label and instead of mode collapse to generate the same image for a specific class. We passed the condition label to both, the generator and discriminator since we didn't use convolution and layers back then we simply concatenated the condition to the input.

AC GAN's, on the contrary encourages the discriminator to reconstruct the label. This led to a major improvement in the image quality and diversity. So now the discriminator should produce two outputs. The first one is whether is a sample is fake or real and the second one is the sample classification. This reconstruction technique should apply to other metadata such as bounding boxes. It is still valid to pass the condition label to the generator in order to produce relevant samples. In this case the discriminator will still need to reconstruct the label along with the fake or real result. The reason that the discriminator does not sheet and copy the label from the generator could be that the adversarial training makes one of them fix it at a time.



Inside the discriminator box, the image is first passed through a convolutional block which functions as a feature extractor. Then the result is though a fully connected layer that performance the classification task and reconstructs the label. It could also perform a regression task and reconstruct the bounding boxes or even both of them. This is called multitask learning. It is also possible to use a pretrained model. This is done by passing the input image through two different pathes. The first one is responsible for determining whether the samples is real or fake. The second path reconstructs the label. [4]



The lost function for AC GAN's. The basic GAN loss is reformulated for convenience.

$$L_S = E[logP(S = real|X_{real})] + E[logP(S = fake|X_{fake})]$$

L_S stands for the source loss that is whether the image in the real or fake.

$$L_C = E[logP(C = c|X_{real})] + E[logP(C = c|X_{fake})]$$

L_C stands for the class loss that is whether the label is reconstructed correctly. So the discriminator now, tries to maximize the sum of both losses. That is it does it's best to find out whether the symbol is real or fake and at the same time reconstruct the label. The generator is trying to minimize the probability that it gets caught as fake. [2] [4]

7. Conclusions and future work

Generative adversarial networks are a promising class of generative models that have been hampered by unstable training and the lack of a suitable evaluation metric. This work presents partial solutions to these two problems. We propose several techniques for stabilizing training that allow us to train models that were previously untrainable. In addition, our proposed evaluation metric (the inception score) provides a basis for comparing the quality of these models. We apply our techniques to the problem of semi-supervised learning and obtain the best results on a number of different datasets from the computer vision domain. The contributions made in this work are practical in nature; we hope to develop a more rigorous theoretical understanding in future work. [6]

References

[1] Arjovsky, M., Bottou, L., 2017. Towards principled methods for training generative adversarial networks. arXiv preprint arXiv:1701.04862 .

[2] Arjovsky, M., Chintala, S., Bottou, L., 2017. Wasserstein generative adversarial networks, in: International conference on machine learning, PMLR. pp. 214–223.

[3] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., Bharath, A.A., 2018. Generative adversarial networks: An overview. IEEE Signal Processing Magazine 35, 53–65.

[4] Qamaruddin, M., 2021. Introduction to generative adversarial networks with pytorch. URL: https://www.udemy.com/course/introduction-to-generative-adversarial-networks-with-pytorch/.

[5] Tan, W.R., Chan, C.S., Aguirre, H.E., Tanaka, K., 2017. Artgan: Artwork synthesis with conditional categorical gans, in: 2017 IEEE International Conference on Image Processing (ICIP), IEEE. pp. 3760–3764.

[6] Zhu, J.Y., Park, T., Isola, P., Efros, A.A., 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks, in: Proceedings of the IEEE international conference on computer vision, pp. 2223–2232.