*Classification Loss in GAN Loss to Motivate Generator Behavior*

Ben Lister   
Department of Computer Science  
George Mason UniversityFairfax, U.S.  
blister@gmu.edu

*Abstract*—Generative adversarial networks have shown high capability in learning complex mappings between source and target domains. A limitation of this model is the requirement of available target data for training. In this paper I discuss a technique for training cGANs without target data, motivating generator output only by components of the loss function.

Keywords—GAN, Variational autoencoder, loss

# Introduction

In a GAN scheme, a generator component randomly generates content in some domain, and a discriminator component decides the authenticity of both the generated content and the real content. By having opposing objectives (generator aims to produce content that can fool the discriminator, discriminator aims to be able to detect generated content) in training they force the other to improve at their role, until eventually the generator is ideally able to produce realistic content indistinguishable from what’s real. In a Conditional Generative Adversarial Network (cGAN) the output vector is deterministically created from some input vector, and so through training the network learns the mapping from source domain to target domain.

The aim of this paper is to explore the use of image classification as a component of loss in an Image-to-Image cGAN, either minimizing classification loss or maximizing it depending on the objective. Specifically, given an image in a particular domain (birds) that is one of N classes (species), produce a new image in the same domain that is

1. Same/different predicted class
2. Maximally/minimally distant from input

# Approach

A traditional generator loss function is given by

*)* [3]

Where G(x,z) is the generator output and D(x, G) is the discriminator’s decision on the authenticity of G. As the generator wants the discriminator to misclassify its outputs as authentic, the generator aims to maximize D(x, G), and therefore minimize 1 – D(x, G), making it suitable as a component of the generator’s loss. In many Image-to-Image GANs, an additional component is added to the generator loss – a comparison, usually mean-squared error (MSE) or structured similarity index (SSIM) between the generator output and the target image. As the GAN attempts to minimize loss by adjusting its parameter weights, the difference between known target and generated target also diminishes. The loss function is now

Where is a hyperparameter used to control the weighting of relative to the adversarial loss. If there are no target images, however, there is no *y* so the function of the component (distance between output image to some ground truth) must be modified to be a comparison between generator input and generator output. Care should be taken here to ensure the meaning of this comparison (what is being minimized?) is in line with the overall objective of the GAN.

Another tool that can be used in loss is the output of a pre-trained model. The generator output is input to the model, and the output of that can be used in a new component of the loss function. In this way, generator output can be tuned to produce more specific output, while still satisfying the adversarial loss/pixelwise distance loss. While the aim of this paper is to use image classification as a loss component, many different types of models could apply here, so long as the loss component produced is meaningful and helps define the content the generator should produce.

In summary, when training a GAN without target vectors, there are three sources of loss which contribute to generator loss: adversarial loss (is minimized when the generator fools the discriminator), input-output distance loss, and input-output classification loss. By making the latter two sources of loss contradictory, generator output can be constrained, even without available target data.

The first goal of this project was to train a generator to perform spatial transformations on images. By penalizing both classification loss and input-output pixelwise similarity while encouraging input-output color histogram similarity, the generator would be forced to produce images of birds that were the same class and same color distribution while still maximally different on a pixel level. The best way to accomplish such an objective would be a spatial transform, such as a flip or rotation. Unfortunately, this is likely beyond the capability of the GAN architecture used. The model would simply forego one component of loss and produce images that were desaturated, blurry, or off-color versions of the original.

A close up of a bird

Description automatically generatedA small bird perched on a branch

Description automatically generatedA colorful bird perched on top of a table

Description automatically generated

*Different failure modes (top row original, bottom row generator output)*

The second goal was the opposite, instead encouraging classification loss while also encouraging color histogram similarity. This was met with a higher degree of success, likely due to the relative ease of misclassifying compared to classifying.

A colorful bird perched on a branch

Description automatically generated A small bird perched on a branch

Description automatically generatedA bird standing next to a body of water

Description automatically generated

*Top (original): Rufuos Motmot, Barn Swallow, Flamingo*

*Bottom (prediction of generator output): Crested Caracara, Canary, Cuban Tody*

Interestingly, also encouraging pixelwise similarity with SSIM caused the generator to produce images that were almost exactly the same as the input, forgoing the (mis)classification component of loss, even with low relative to .

# Details

The data is from Kaggle and consists of 27503 images for training, 1000 for testing, and 1000 for validation. The images are colored and are centered and cropped such that at least 50% of the image’s pixels belong to the bird, and then are resized to 224x224 and stored in a jpg format (<https://www.kaggle.com/gpiosenka/100-bird-species>).

The overall model architecture is a modified version of pix2pix, implemented in pytorch (<https://github.com/eriklindernoren/PyTorch-GAN/tree/master/implementations/pix2pix>). Because pix2pix expects source-target pairs for training data, its discriminator takes two inputs [3]. This, the aforementioned changes to the loss function, such as using SSIM instead of MSE (<https://github.com/Po-Hsun-Su/pytorch-ssim>), and other necessary changes to the supporting classes such as dataloaders were made.

The model was trained for on a Windows 10 machine with a NVIDIA GeForce RTX 2060 GPU, 16 GB RAM, and an Intel i9 2.60 GHz CPU. Training was done for one epoch over 26762 batches using Adam optimizer with learning rate of 0.0001. Data augmentation to mitigate overfitting included a bicubic resize to 256x256, random horizontal flip, random brightness/saturation change by -20% to 20% and normalizing with mean 0.5 and standard deviation 0.5.

The classifier uses an EfficientNet architecture [2] and is implemented in fastai using pytorch (<https://github.com/lukemelas/EfficientNet-PyTorch/blob/master/efficientnet_pytorch/model.py>). It was trained on the same machine on the same dataset for one epoch, achieving a classification accuracy of 97.3%. Data augmentation was identical to that done for the GAN.

# Results

While the initial goal of encouraging the generator to spatially transform the input was unsuccessful, the secondary goal of maximizing classification error while minimizing color difference had some success. To evaluate, images from the test data set were given to the generator, and classification was performed on the generator output. The classifier achieved an accuracy of 82.53% - significantly lower than its 97% accuracy on non-transformed images. Unfortunately, the generator is only capable of changing the color of the input image, and the classifier is decently robust against this.

When the generator was successful in producing an image that classified to a different species, the result was usually an image negative. This is a trivial and noninteresting solution to the problem.

The GAN also appeared to overfit in training, as the difference between the output images in the 2000th batch and the 25000th batch was minimal.

# Related Works

## Privacy-Protective GAN for Face De-identification

Wu, Yang, Ling et al. suggest a novel PP-GAN architecture to deidentify images of human faces. They define an ideal deidentified face as one that:

1. Cannot be matched to the original through facial recognition (causes verification to fail)
2. Is still a human face
3. Is minimally different from the original

Due to the nature of the problem, however, paired deidentified faces for all the input faces are not available, as the authors aimed to find a better strategy for deidentification than the alternatives, not mimic the existing strategy by training a GAN to do it. Lacking target images to compare the generator output to, they instead achieve the above three objectives by 1) maximizing verification loss on a pre-trained one-shot Siamese network for facial verification, 2) adversarially-trained discriminator as per usual, and 3) maximizing SSIM between input and generator output [1].

![A picture containing drawing, clock

Description automatically generated]()

*PP-GAN architecture. Generator is a U-Net autoencoder similar to pix2pix, discriminator receives single images rather than pairs, verificator is a pre-trained Light CNN-9 model [1].*

Both the verificator component and the GAN component were trained on the MORPH dataset (n=55,00 images of over 13,000 different people, mostly Black and White men). They used OpenFace for data augmentation to detect, align, and crop face areas. To ensure utility preservation (related to 2nd objective – deidentified image is still a human face) they divide data into six groups based on age (young/middle/old) and race (Black/White). On each group they train a new verificator (achieving over 95% verification accuracy across all groups) and test using four different model architectures – using cGAN only (control), cGAN with SSIM loss, cGAN with pretrained verificator loss, and cGAN with SSIM loss and verificator loss [1].

![A person looking at the camera

Description automatically generated]()

![A person making a face for the camera

Description automatically generated]()

*Comparison between deidentified faces across the four experimental setups [1]*

The architecture the authors propose, cGAN + SSIM + Verification, was then evaluated using the verificator’s deidentifcation rate (1-verification accuracy), achieving 100%/97%/93.7% deidentification on Black young/middle/old, and 94.4%/90.8%/84.7% deidentification on White young/middle/old [1].

##### Conclusion

Elaborate loss functions are no replacement for paired training data. In PP-GAN, the dataset had far less inherent variance, as the facial images were all close-cropped, looking at the camera, and in grayscale. Furthermore, the data was further divided into subcategories. The data used in this project was much more varied, with almost 200 different kinds of colorful birds in very different poses. PP-GAN also featured a more robust verification loss “to eliminate the overfitting caused by the pre-trained verificator” [1] while the equivalent component in this model, the classifier, had no such feature.

There may still be some use for pre-trained models as a component of loss, but they should not be seen as a tool for training a GAN.

##### References

1. Y. Wu, F. Yang, and H. Ling. Privacy-protective-gan for face de-identification. arXiv preprint arXiv:1806.08906, 2018.
2. M. Tan and Q. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in Proc. 36th Int. Conf. Mach. Learn., 2019, pp. 6105–6114.
3. P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-toimage translation with conditional adversarial networks. In CVPR, 2017.