Data Augmentation Powered by Generative Adversarial Network

Szabolcs Hornyák, Károly Póka

Abstract—The abstract goes here.

Index Terms—Keywords...

1 Introduction

INTRODUCTION goes here.

2 Basic theory of GANs

GAN theory goes here.

3 Experiments with existing architectures

Beacuse the basic architecture of GANs does not provide control over the features of the generated images we looked for more advanced architectures.

3.1 InfoGAN

The first network we examined was the InfoGAN by Chen et al. [2]. This generative architecture attempts to control the features of the generated images via so called code components concatenated to the latent (noise) vector. In order to encourage the generator to connect these code components to meaningful image features (to learn a disentangled representation) the authors introduced a new criteria in the generator's loss function: the generator should maximize the mutual information between the input latent vector and the generated image.

The total loss of the generator (L_G) is defined as:

$$L_G = -L_D(x_G) - \lambda I([z, c], x_G) \tag{1}$$

where $L_D(x_G)$ is the loss of the discriminator for generated images, $I([z, c], x_G)$ is the mutual information between the latent vector and the generated image, λ is a new hyperparameter, tipically 1.

The InfoGAN architecture provides great control over the image features if the type and number of code components are suited to the underlying distribution of the dataset (e.g. use of a categorical component with 10 categories for MNIST). The downside of this architecture is the relative poor quality of the generated images for complex data such as faces from the CelebA dataset. The architectures described below attempts to address this problem and to further remove the entanglement of features in the latent space.

3.2 ST-GAN

One improvement of the InfoGAN architecture is called Semantic Transformation GAN [3]. This networks aims to remove the complicated procedure of approximating the mutual information by introducing an encoder network. The task of this encoder network is to reconstruct the latent vector presented to the generator from the image it generated. This can only be achieved if the generator preserves the latent vector's information in the generated images. In other words, perfect $[z, c]$ reconstruction can only be achieved if the information flow from the input of the generator to the output of the encoder is lossless, and it also means that the generator maps all the information of the latent vector into the generated images.

One can use the ST-GANs encoder network to map an existing image into the latent space, then alter its code components and feed the altered vector to the generator in order to generate an altered version of the original image. With the above described architecture the quality of the altered images are not guaranteed. To overcome this problem the authors of the paper used the reconstructed $[z, c]$ vector from the encoder, altered it ($[z, c']$) and fed back into the generator during training, and definined another objective for the network to improve the quality of the altered images.

This architecture is a significant improvement of InfoGAN, but it has a relatively large number of subnetworks to train and loss functions. Latter leads to conflicting training objectives and slower convergence.

3.3 BEGAN

A further improved InfoGAN-based architecture is called the Boundary Equilibrium GAN.

Because at the beginning of the training the generator creates images greatly different from the real images the discriminator has an advantage. To overcome this and assure ″fair play″ between these networks the BEGAN architecture introduces a balancing process, hence the name ″boundary equilibrium″.

The BEGAN architecture also uses the idea of latent vector reconstruction to remove the need of approximating the true mutual information, but uses a whole autoencoder network as a discriminator.

This discriminator has

4 Our extensions and improvements

The main goal was impossible to complete with the existing architectures, because we wanted to finetune a given picture, but the generator has a multidimensional vector input. Our idea was to train a separated predictor to get noise and code vectors from a given pictures. At first two questions come up: what the predictor net's output should be? Could be whether we can use the Discriminator as a predictor, because it can give us a categorical and continuous code from a given image. The answer is not. The Discriminator was trained to give a judgment about the classification (True or Fake), and the appropriate code vectors. We can not train the Discriminator to give the noise vector also, because that trick would change the training results in a bad way. So, let's make a Noise Predictor net. We used a network based on a same architecture as the Generator has but with noise, categorical and continuous code as outputs. Our network object has on opportunity to train only the predictor network. We trained the InfoGan first and then finetuned the Noise Predictor with the Generator weights. At first, we used the built-in pytorch MSE (Mean Square Error) loss function to compute the loss between the generated image based on the predictor's output and the original image. The mathematically distance between the pixels does not always show the visible difference for humans between two pictures. Therefore, we used the Differentiable Multi-Scale Structural Similarity (SSIM) index the calculate the predictor's loss.

$$MS-SSIM(x,y) = I_M(x,y)^{\alpha M} \prod_{j=1}^{M} C_j(x,y)^{\beta_j} S_j(X,y)^{\gamma_j} \tag{2}$$

Obviously, the goal is for the network to learn to produce visually pleasing images. To the root of the MS-SSIM is the simple SSIM function. Where for pixel p is defined as

$$SSIM(p) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \cdot \frac{2\sigma_{xy} + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} = l(p) \cdot cs(p) \tag{3}$$

Means (μ) and standard deviations (σ) are computed with a Gaussian filter with standard deviation (szigma G kéne ide is). The max output of this metrics is 1 when the two inputs are similar to each other and the minimum is 0 when they are not. In the training loop we can use the loss function like this:

$$\mathcal{L}^{SSIM}(P) = \frac{1}{N} \sum_{p \in P} 1 - SSIM(p) \tag{4}$$

where the p is the pixel and we are looking at its neighbourhood as large as the support of the G(σ_G). So, we can write the loss as

$$\mathcal{L}^{SSIM}(P) = 1 - SSIM(p') \tag{5}$$

because the convolutional nature of the network, where the p(hullám) is the center pixel of patch P. The processed quality depends on the size of (szigma G), but rather then fine-tuning it we can use MS-SSIM. Further information you can visit the cited source of Nvidia Resource.

$$MS-SSIM(p) = l_M^{\alpha} \cdot \prod_{j=1}^{M} cs_j^{\beta_j}(p) \tag{6}$$

5 Results

The results were not as satisfiable as we wanted. It seems the InfoGAN architecture is focused only on the maximization of the mutual information not produce visually pleasing images. We also noticed, that for a good prediction the Noise Predictor needs to have a Generator with good quality of produced images. Also, the resolution is import, because we tried both 64x64 and 128x128 images, and the bigger resolution driven to a better quality of reconstruction. At this area we had to handle a trade-off because the better-quality demand much more compute performance or much more training time. We trained our model with a Nvidia TitanX P with 12Gb GPU memory. We completed more than 30 training and several of them was over 30 hours. On our results obviously visible the bad generation and quality caused fault. Some part of the result pictures contains the information what we want, but overall the generated images are not usable for data augmentation yet.

6 Future work

To improve the generation quality, we will use the improved BeGAN structure. It is possible the extend the existing BeGAN with the latent code and our Noise-Predictor.

Acknowledgments

The authors would like to thank...

References

[1] R. Abdal, Y. Qin, and P. Wonka, "Image2stylegan: How to embed images into the stylegan latent space?" 2019.
[2] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," 2016.
[3] M. Liu, Y. Ding, M. Xia, X. Liu, E. Ding, W. Zuo, and S. Wen, "St-gan: A unified selective transfer network for arbitrary image attribute editing," 2019.