Generative Adversarial Networks for Computer Vision Tasks

Subjects: Computer Science, Artificial Intelligence

Contributor: Ana-Maria Simion , Șerban Radu , Adina Magda Florea

Computer vision tasks have gained a lot of popularity, accompanied by the development of numerous powerful architectures consistently delivering outstanding results when applied to well-annotated datasets. However, acquiring a high-quality dataset remains a challenge, particularly in sensitive domains like medical imaging, where expense and ethical concerns represent a challenge. Generative adversarial networks (GANs) offer a possible solution to artificially expand datasets, providing a basic resource for applications requiring large and diverse data.

generative adversarial networks (GANs) DCGAN WGAN cGAN CycleGAN ProGAN StyleGAN SRGAN medGAN AMD-GAN

1. Introduction

It is largely acknowledged that generative adversarial networks (GANs) [1] were a major breakthrough in the field of artificial intelligence. The idea behind GAN was first introduced in 2014 by Ian J. Goodfellow and his team and years later it remains one of the most relevant and promising methods used to tackle generative problems in computer vision and many other fields. GANs are great for generating all sorts of data, not only images. It is also used for generating text, tabular data, music and audio, 3D models, etc. It is also the first architecture developed in the field of Deep Learning that was able to produce such high quality results on most datasets they were trained on, no matter the domain.

The GAN is composed of two parts: the generator and the discriminator. The generator generates new instances of data. The discriminator evaluates the authenticity of the generated data. At the beginning of the training the generator will not produce good images, but the discriminator will give feedback to the generator and it will improve. In order to be able to evaluate the authenticity of the generated data, the discriminator is trained on real data. Then, the discriminator receives the image from the generator, and it assigns a probability of the generated image being real. Basically, the discriminator is a standard CNN used for classification, it checks if the generated data falls into the real or fake category. The way the generator works is opposite to the discriminator. The discriminator downsamples the image in order to obtain that probability, meanwhile, the generator takes its input and upsamples it as much as possible to become a good enough piece of data. Both networks try to optimize their specific loss function. During the training process, both networks change and influence one another. Hence the name “Adversarial”, the networks compete in fooling each other. The final goal is to make a generator good enough to approximate its generated distribution to the distribution of the real images. The original paper shows results for the MNIST dataset [2] and CIFAR-10 [3], both of which consist of simple images. MNIST is a dataset that contains 60,000 small images with handwritten digits. The size of each image is 28 × 28 pixels, the digit itself is white and the background is black. Since the image is so small and contains only two colors, the researchers of the paper [1] used noise as input for the generator Network. Using random noise as input for the generator of a GAN is a common way to generate data. However, depending on the complexity and specificity of more advanced tasks, a more sophisticated approach may be necessary [1].

For some scenarios, using random noise can be sufficient. The GAN Generator will take this noise and generate corresponding synthetic data. However, there are situations where it is necessary to provide more meaningful or structured inputs to the generator. For example, when generating realistic images, adding random noise can lead to inconsistent or blurry results. Instead, it may be beneficial to provide the generator with a latent vector.

2. DCGAN—Deep Convolutional GAN

The idea of a deep convolutional GAN was introduced one year after the publication of the original GAN idea. It is similar to the original GAN, but it introduces the use of a deep convolutional network instead of a fully connected network. As mentioned before, CNNs are great when working with images because they look better for spatial correlations. This is making the DCGAN a better option to use when generating images compared to the original GAN [4].

3. WGAN—Wasserstein GAN

WGAN was introduced in 2017 and received its name from the Wasserstein loss that it is using. They propose the use of the Wasserstein distance (the effort that it takes to transform one distribution into another) instead of the Jensen–Shannon divergence to compare the distributions of the generated data and the training data. The researchers call the discriminator a “critic”. WGAN uses weight clipping in order to enforce the 1-Lipschitz constraint. To ensure the validity of the Wasserstein distance calculation, WGAN enforces the 1-Lipschitz constraint on the critic (also known as the discriminator in a GAN). The Lipschitz constraint ensures that the critic’s output does not change dramatically with small changes in the input. This constraint helps stabilize the training process and improves the quality of the generated samples. The critic must satisfy the Lipschitz constraint. To enforce the 1-Lipschitz constraint, WGAN uses weight clipping. Weight clipping involves limiting the values of the weights in the critic to a predefined range, typically by setting a threshold. This prevents the weights from growing too large and helps ensure that the critic’s output remains within a reasonable range. However, it’s worth noting that weight clipping has some limitations and can lead to issues. Overall, the 1-Lipschitz constraint, enforced through weight clipping in WGAN, is a technique used to stabilize the training process and improve the convergence of the Wasserstein GAN model. The researchers of the work in [5] emphasize that by using this loss, they obtained better stability in training and that it solves common problems like vanishing gradients and mode collapse.

Mode collapse in generative adversarial networks (GANs) refers to a situation where the generator of the GAN fails to capture the full diversity of the true data distribution and instead produces limited variations or replicates a few specific modes of the data. In other words, mode collapse occurs when the generator generates similar or identical samples, regardless of the diversity present in the training data. For example, for the MNIST dataset, which contains digits from 0 to 9, the generator tends to produce digits from only two classes: 1 and 8. Mode collapse can be problematic because it leads to a lack of diversity in the generated samples. Instead of capturing the full complexity and variation of the real data distribution, the generator may focus on a subset of modes or produce repetitive outputs. This can result in low-quality or unrealistic generated samples that do not represent the full range of the desired data distribution. Because the generator is overfitting, the discriminator does not improve, so the generated data remains limited to a small number of classes.

Vanishing gradients can hinder the training process and lead to slow convergence or poor performance in GANs. When the discriminator is too good, the generator does not have any chance to improve. When the gradients are too small, they do not provide sufficient information to guide the generator in improving its generated samples. As a result, the generator may struggle to learn and adapt effectively to produce better-quality samples. The vanishing gradient problem is particularly challenging in GANs, because of the adversarial nature of the training. The generator and discriminator constantly try to outperform each other, and if the discriminator's gradients diminish, the generator's updates become less informative and effective.

Wasserstein loss tackles both problems. If the critic is not getting stuck in local minimum, it is forcing the generator to try something new when it is stabilizing and returning outputs from only a few classes. Also, Wasserstein loss is a good indicator for the quality of the image. As the number of epochs increases, the value of the loss function decreases, proving a better alignment of the generated distribution to the original one used for training.

The Critic maximizes this difference, while the target of the generator is to minimize it. The lower the difference, the better the quality of the generated data.

The downside of the WGAN and Lipschitz Constraint is that the quality of the output is very dependent on the hyperparameter c used in weight clipping. In the context of weight clipping in Wasserstein GAN, the hyperparameter "c" represents the threshold or maximum value to which the weights of the critic (discriminator) network are clipped. Setting "c" too small can lead to gradient vanishing or hinder the learning process, while setting it too large may result in weak enforcement of the Lipschitz constraint, allowing the critic to produce unstable or inaccurate gradients. Finding the optimal value for "c" often requires experimentation and empirical observation to strike a balance between stability and effective training in WGAN [5].

To overcome this problem, [6] proposed the use of a gradient penalty. In WGAN, the gradient penalty is applied by adding a regularization term to the loss function. The regularization term encourages the gradients of the critic's output with respect to the input samples to have a consistent and controlled magnitude. Specifically, during training, random interpolations are generated between pairs of real and generated samples. These interpolations are used to calculate the gradients of the critic's output, with respect to the interpolated points. The gradient penalty term is then computed as the norm (e.g., L2 norm) of these gradients. The objective is to encourage the gradients to have a magnitude of 1, indicating Lipschitz continuity [6].

4. cGAN—Conditional GAN

cGAN was introduced in order to control the type of image that wants to generate. For example, for the MNIST dataset, it cannot control the generation of a specific digit with the methods presented so far. cGAN emerged as a solution to address the challenge of understanding the connection between the random input provided to the generator and the actual characteristics present in the training images. Without this, it would not be able to influence specific features that would generate a specific result.

In conditional GAN architecture, both the generator and the discriminator will be provided with some additional information to generate images of specific classes. Compared to the original GAN, some additional information for the generator and the same information to the discriminator are provide. That information is a set of fake class labels. The discriminator with the real labels for the training data is also provided. The advantage of cGAN is having more control over what is generated, and convergence is faster. cGAN architecture is the foundation for some very important image to image translation architectures like Pix2Pix and CycleGAN [7].

5. CycleGAN

Unlike Pix2Pix it's no need to use paired datasets. In Pix2Pix data from both categories are needed in order for the model to understand how to correlate and translate them. CycleGAN applies the learned style from one dataset to the other dataset. For example, if we want to perform translation from edges to images, we need a dataset that contains images with edges and a corresponding dataset that includes the real image of that item. What makes CycleGAN unique is its ability to utilize unpaired data, meaning we can have a dataset of castle images and another dataset featuring a specific painting style. By the end of the process, we can successfully apply the painting style to the castle images, despite the fact that the images were never paired together at any point.

The generating process begins with an input image from either domain A or domain B. For instance, if we have a photograph of a horse, it belongs to domain A. To transform the input image into the target domain, we pass it through generator A (GA). GA takes the input image and generates an image in the target domain, which is domain B in this case. The output image produced by GA possesses the style and characteristics of domain B. If we desire to revert the image back to its original domain, we can use generator B (GB). By passing the image generated by GA through GB, GB takes the generated image from domain B and transforms it back to domain A. The resulting image should ideally resemble the original input image of the horse photograph. Therefore, by utilizing generators GA and GB, we can seamlessly translate an image from one domain to the other. For example, starting with a horse photograph in domain A, GA can generate an image in domain B that resembles a zebra painting. Then, by passing this generated image through GB, we can obtain an image back in domain A that closely resembles the original horse photograph. This is also called cycle consistency loss.

The discriminator process is also made up of two parts. The first discriminator receives as input the generated image and outputs a classification matrix. The second discriminator receives real data from domain B as input and outputs a classification matrix. In the end we apply least squares loss between the two classification matrices. The first discriminator encourages the first generator to translate the input into fake images as indistinguishable as possible from images in domain B and vice versa for the second discriminator and the second generator. This creates a consistent cycle, hence the name of the architecture.

CycleGAN is a great option when dealing with unpaired data, but if paired data is available, Pix2Pix architecture is preferred because it is faster in training. The results are realistic only when the style translation is made between objects with a similar shape or features. Also, the results are very good in tasks that include texture, style, or color changes [8].

6. ProGAN—Progressively Growing GAN

T. Kerras and his team from NVIDIA introduced the ProGAN architecture in 2017 and its purpose was to generate faces of people that do not exist in the real world. The name of the architecture comes from the fact that they progressively increase both the generator and the discriminator. The layers of the network are organized in the form of a pyramid. The latent vector starts from a low resolution and increases with every layer. For the experiments they used a dataset of images with celebrities of size 1024 × 1024 pixels and the generated results were very good. Naturally, the generated faces bear a resemblance to the individuals present in the training dataset. The generated results can be controlled if different features are specified, for example we can control the direction the person is looking at [9].

7. StyleGAN

StyleGAN is the state-of-the-art method for generating new images. It is also the work of T. Kerras and the other team members from NVIDIA. StyleGAN is based on ProGAN. It employs a distinct generator architecture inspired by style transfer, specifically adaptive instance normalization. This architecture enables the model to learn high level features, such as facial positions, hairstyles, beards, and various other characteristics. In StyleGAN the latent vector is mapped to an intermediate latent space that controls the generator through adaptive instance normalization. Multiple improvements and versions are available for StyleGAN. The most recent one is called StyleGAN3 [10].

8. SRGAN—Super Resolution GAN

Super resolution GAN tackles the challenge of estimating a higher resolution image from a low resolution version. It uses a perceptual loss function consisting of an adversarial loss and a content loss. The discriminator is trained to distinguish between original low resolution images and super-resolution generated images. Moreover, the researchers proposed the use of a content loss based on perceptual similarity, rather than pixel space similarity.

The generator uses residual blocks and skip connections. This is meant to keep the information from previous layers, so that the network could choose more features adaptively. The input used for the generator is the low-resolution image. The used loss function is the perceptual loss function. The perceptual loss function is made up of two parts: content loss and adversarial loss. The content loss measures how much of the original content is going to get lost during training. When we perform super resolution, new pixels need to be added to the image. The neural network needs to produce new pixels from the image. When this happens, we can lose some context from the original image. New pixels need to be added to the image and they should be related to the original image, that is what the content loss indicates [11].

9. ESRGAN—Enhanced SRGAN

It is an improvement compared to the original SRGAN. There are some changes in the architecture and loss function. A new deep neural network model instead of the residual blocks was adopted. A relativistic average GAN [12] is used instead of the vanilla GAN. The perceptual loss is also improved before applying the activation functions. The ESRGAN produces less blurry and better quality images than SRGAN [13].

10. MedGAN

MedGAN appeared as a solution to the problem that not enough medical data is available for experiments. The researchers of [14] indicate that accessing medical databases can be difficult, particularly due to the sensitive nature of the data. Furthermore, the researchers highlight that large medical organizations, which have access to such databases, often demand significant financial resources for their use. medGAN is to accomplish two important objectives. The first one is to preserve the privacy of the patients, so that it would be practically impossible to gain knowledge about real people from the synthetic data. The second one is to produce data of a good enough quality, so that the models trained on synthetic data could perform as well as the ones trained on real data.

The particularity of the medGAN architecture is that an encoder–decoder model was added. The discrete real data input is encoded into a continuous embedding vector. Then, the decoder is used to convert back to a discrete representation of the input in the data space. Both the encoded input data as well as some random noise are put through that decoder. Then, the discriminator takes the data provided by the decoder as well as the real data and decides the probability of the data being true or false. For the autoencoder, cross-entropy loss is used.

Different GAN architectures were experimented with to see which one would provide the best dimension-wise probability distribution. MedGAN showed the best performance [14].

11. AMD-GAN

The researchers of [15] present the difficulty in detecting fundus diseases, particularly in the context of retinal imaging, using scanning laser ophthalmoscopy (SLO) images with an ultra-wide field.

A primary obstacle encountered in the approach of an artificial intelligence-based solution is the persistent issue of limited available data. Usually, a method based on GANs could be used to augment the dataset, but in this case, a constraint specific to this medical concern and its associated image data type lies in the inefficacy of GANs. The small size of salient objects within SLO images causes the GAN difficulties in extracting high-level features.

To overcome these problems, the researchers of [15] proposed a method called AMD-GAN. The method uses a generator and a discriminator. The generator has two parts: an attention encoder (AE) and a multi-branch (MB) structure. The encoder network uses real data and random Gaussian noise as input. The AE and generation modules help extract details from low-level information and generate features at different scales. The AE is made of the following layers: convolution, batch normalization, ReLU, and maxpooling. An attention mechanism is used to combine these extracted features from real data with features generated at the same scale. The fake images are generated from a random Gaussian noise input. The random noise is passing through three pairs of upsampled residual blocks followed by the AE module, and after this, passing through another three upsampled residual blocks (RU block). This output (the fake generated data) along with the real data is then passed to the discriminator. The RU block is made of the following layers: batchnorm, ReLU up-sampling, 3 × 3 conv, batchnorm, ReLU, 3 × 3 conv. The loss used by the generator is described in Equation (1).

The discriminator (used as a classifier in this problem scenario) uses a multi-branch structure based on a ResNet-34 backbone model to capture rich high-level features. The researchers added a deep-wise asymmetric dilated convolution (DADC) [16] block to refine high-level features and speed up training with fewer parameters. The loss used by the discriminator is described in Equation (2).

To generate good fake images, the researchers use adversarial loss which tries to maximize the probability of real images being recognized as real and generated images being considered fake. To ensure that the generated images not only look real but are also correctly classified, the researchers use a classification loss. The classification loss is different for real and fake data. To ensure that the generated images capture the overall content information of the input images, they also use the mean square error (MSE, or L2 norm) between the generated images and the real images. This is called the content loss. The final loss is made of the previously mentioned losses, and they propose the use of two coefficients α and β (which are set between 1 and 10 in their experiments) to control the classification and content losses.

$$L_{generator} = L_{adversarial} + \alpha L_{classificaation}^{fake\ data} + \beta L_{content} \tag{1}$$

$$L_{discriminator} = -L_{adversarial} + \alpha L_{classificaation}^{real\ data} \tag{2}$$

References

1. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. arXiv 2014, arXiv:1406.2661v1.
2. Deng, L. The mnist database of handwritten digit images for machine learning research. IEEE Signal Process. Mag. 2012, 29, 141–142.
3. The CIFAR-10 Dataset. Available online: https://www.cs.toronto.edu/~kriz/cifar.html (accessed on 22 August 2023).
4. Radford, A.; Metz, L.; Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv 2016, arXiv:1511.06434v2.
5. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein GAN. arXiv 2017, arXiv:1701.07875v3.
6. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A. Improved Training of Wasserstein GANs. arXiv 2017, arXiv:1704.00028v3.
7. Mirza, M.; Osindero, S. Conditional Generative Adversarial Nets. arXiv 2014, arXiv:1411.1784v1.
8. Zhu, J.-Y.; Park, T.; Alexei, P.I.; Efros, A. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. arXiv 2020, arXiv:1703.10593v7.
9. Karras, T.; Aila, T.; Laine, S.; Lehtinen, J. Progressive growing of GANs for improved quality, stability, and variation. arXiv 2018, arXiv:1710.10196v3.
10. Karras, T.; Aila, T.; Laine, S. A Style-Based Generator Architecture for Generative Adversarial Networks. arXiv 2019, arXiv:1812.04948v3.
11. Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; et al. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 105–114.
12. Papers with Code. Available online: https://paperswithcode.com/method/relativistic-gan (accessed on 1 June 2023).
13. Wang, X.; Yu, K.; Wu, S.; Gu, J.; Liu, Y.; Dong, C.; Loy, C.C.; Qiao, Y.; Tang, X. ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. arXiv 2018, arXiv:1809.00219v2.
14. Armanious, K.; Jiang, C.; Fischer, M.; Küstner, T.; Hepp, T.; Nikolaou, K.; Gatidis, S.; Yang, B. MedGAN: Medical Image Translation using GANs. arXiv 2019, arXiv:1806.06397v2.
15. Xie, H.; Lei, H.; Zeng, X.; He, Y.; Chen, G.; Elazab, A.; Wang, J.; Zhang, G.; Lei, B. AMD-GAN: Attention encoder and multi-branch structure based generative adversarial networks for fundus

disease detection from scanning laser ophthalmoscopy images. Neural Netw. 2020, 132, 477–490.

16. Li, G.; Yun, I.; Kim, J.; Kim, J. DABNet: Depth-wise asymmetric bottleneck for real-time semantic segmentation. arXiv 2019, arXiv:1907.11357.