Developing a Deep Convolutional Generative Adversarial Network with Multi-Scale Gradients

by Aleksander Brekke Røed, Erlend Rønning and Nicolai Thorer Sivesind

Abstract

Generative Adversarial Networks is a relatively new concept within machine learning which recently have seen a considerable amount of applications. While the framework can be applied to any form of data generation, it has proved its potential in the field of image generation. In this work, we propose an expansion of the Deep Convolutional GAN architecture presented by Redford et al. by applying the concept of Multi-Scale Gradients for GANs, presented by Karnewar in 2019. While the MSG technique intends to improve training stability when generating high resolution images, applying it to the DCGAN architecture for low resolution image generation reduces the network's ability to classify correlated features without further improving training stability.

Related Research

The framework for estimating generative models via an adversarial process where a generator and discriminator are simultaneously trained, was first used by Ian Goodfellow (Goodfellow, 2014). His work inspired several unique ways of implementing GANs.

Multi-Scale Gradients (MSG) for Generative Adversarial Networks is a stabilization technique presented by Animesh Karnewar (Karnewar, 2019). The MSG-GAN provides a stable approach for high resolution image generation.

Best practices for using deep convolutional generative adversarial networks (DCGAN) is described in a paper by Alec Radford (Radford, 2016).

Transfer learning in neural networks was invented by Stevo Bozinovski. It concerns training a model in one domain, and applying it in another.

Introduction

Generative Adversarial Networks (GAN), a framework derived from game theory, is now one of the go-to methods for true-to-nature image generation. However, GAN models suffer from two main problems: mode collapse and training instability. Different strategies for stabilizing the training have been put into place since 2014 when GAN first was invented. We will explore Deep Convolutional Generative Adversarial Networks (DCGAN), Multi-Scale Gradients for Generative Adversarial Networks (MSG), and Transfer Learning.

Theory

Generative Adversarial Network

A Generative Adversarial Network (GAN), is a framework for finding the distribution of training data and generating new data from that same distribution (Goodfellow, 2014). The framework consists of two models, a generator (G) and a discriminator (D). The generator's goal is to produce data points that mimic the training data, and the discriminator's goal is to differentiate between real and generated data points. To train the two models, they are put in an adversarial context. The discriminator evaluates both real and generated data points, and the results are then used to calculate loss for each of the models, which in turn is used to adjust the models' weights accordingly.

$$\min_G \max_D \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[1 - \log D(G(z))]$$

MinMax loss function of a GAN

The GAN framework can be applied to generate any type of data, but its potential has been realized within the field of image generation with the use of Convolutional Neural Networks (CNN).

Deep Convolutional GAN

A Deep Convolutional GAN (DCGAN) is an GAN architecture presented by Radford. et al, which has seen great success for efficiently training the framework for image generation. His paper has the following guidelines:

- Strided convolutions instead of pooling functions in discriminator
- Fractional -strided convolutions in generator
- Batch normalization within all intermediate layers
- ReLU-activation in generator intermediate layers
- Leaky ReLU-activation in all discriminator layers

GAN with Multi-Scale Gradients

A GAN with Multi-Scale Gradients (MSG-GAN) connects the layers in the two models by injecting an output batch from the intermediate generator layers with the output batch from the corresponding intermediate discriminator layers. This combination is then forwarded to the next discriminator layer (Karnewar, 2019). The overarching goal of MSG-GANs is to improve stability in GAN training, which is notoriously unstable and hyperparameter sensitive.

Transfer learning

Transfer learning is the practice of using a pre-trained model in a different domain. This approach has been shown to be particularly effective in deep learning image generation. Because real images share similar color value patterns, a model trained in one domain can efficiently be trained to produce images from another.

Methods

Process

Building an MSG-GAN while learning about GANs for the first time required a lot of trial, error and research. Various approaches for all parts of the framework were tested while increasing the complexity of our implementation by applying interesting concepts found in relevant research papers. To learn the basics, a rudimentary vanilla GAN was built with the goal of generating linear sequences of numbers. The next step of moving from a one-dimensional to the two-dimensional domain of DCGAN proved to be difficult.

Having the generator and discriminator properly function in a working training loop took several days of debugging, but once accomplished, made development significantly less complicated. Lastly, the Multi-Scale Gradients were implemented to round off the project.

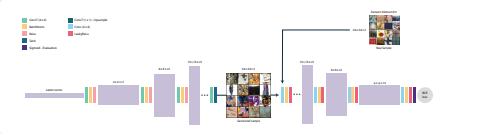
The implementation is written entirely in Python, using the machine learning framework PyTorch. Training was run with CUDA on an Nvidia RTX 3080.

Training

The models are trained separately within the same training loop, which handles one batch of images at a time. The discriminator is trained by evaluating images from both the real and generated domain. Its classification scores are then passed to a BCE-loss function along with the correct labels to calculate the discriminator loss. The generator is similarly trained by passing a new batch of generated images to the discriminator and backpropagating the classification loss to the generator's optimizer.

Implementing a DCGAN

To implement the DCGAN we have followed the guidelines presented by Ratford. et al.



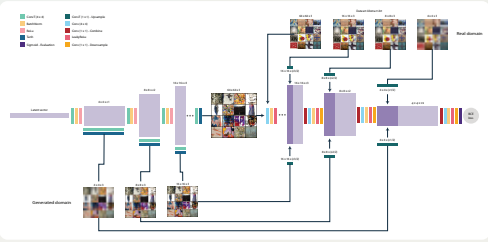
Implemented DCGAN architecture

Image generation is done by running random noise from a normal distribution through a dynamic number of two-dimensional transposed convolutional layers, which varies depending on the desired image resolution. For each generator intermediate layer (GIL), the output dimensions are doubled until the desired resolution is achieved. To render a GIL output to an image, it is passed through a transposed convolutional layer, before applying a hyperbolic tangent activation function.

The discriminator has a similarly reversed architecture which is implemented with the inverse intermediate layer scaling function of the generator. Standard convolutional layers are used for downsampling to allow the network to learn its own pooling function, and Leaky ReLU is used as activation to avoid the dying ReLU problem, which occurs when its outputs stagnate at zero, halting the updation of model weights.

Applying Multi-Scale Gradients

We implement Multi-Scale Gradients by rendering images from each GIL and passing them to the corresponding discriminator intermediate layer (DIL).



Implemented MSG-DCGAN architecture

Due to the fact that the discriminator has the inverse intermediate layer scaling function as the generator, we can use the image rendering layer implemented in the DCGAN to generate an image at each GIL which matches the corresponding DIL's output resolution. After each DIL, the image passed from the corresponding GIL is upsampled to a desirable proportionate sample size and injected into the DIL output. This should, according to the MSG-paper, improve discriminator robustness by increasing the overlap between real and generated images which in turn should give the generator more informative feedback and reduce the possibility of loss convergence.

Datasets

To evaluate progress and test our GAN implementations, we trained our models on multiple datasets of varying complexity and size:

- MNIST Handwritten Numbers, 60 000 images
- Abstract Art, 2782 images
- Bored Ape Yacht Club (NFT Cartoons), 10 000 images
- CelebA (Celebrity faces), 200 000 images

Results

DCGAN

By following the guidelines of Redford et al. our DCGAN produced decent results with all datasets. The epoch thresholds for producing satisfactory results was inversely proportional with the size of the training sets. For the Abstract Art dataset, that threshold seemed to be around 100 epochs, and results peaked at 200-300 epochs. Beyond 300 epochs the generator slowly starts overfitting, and at around 1700 epochs there is a complete mode collapse.

Generated images, epoch 200

Mode collapse, epoch 2000

A subset of the training set for comparison

Multi-Scale Gradients

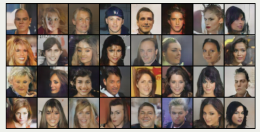
Applying Multi-Scale Gradients to our DCGAN-implementation did not improve results. For the abstract art dataset, the MSG-DCGAN quickly mode collapsed, but for the remaining sets, decent results were produced by experimenting with various learning rates and injection sizes. By injecting GIL images of equal (100%) channel size as the DIL output it is injected into, the generator produces pixelated images. With an injection size of 50% as shown in our architecture figure, the results improved, but previously unencountered characteristics appeared.

100% Injection size, epoch 50

50% injections size, epoch 11

50% injection size, epoch 50

With an injection size of 50%, lattice structures are prominent in early stages of training. This is due to the kernel overlapping during deconvolution in the generator. In the final epoch of 50, the lattice structure is less prevalent, but wavelike contours are apparent within complex features like hair. In addition, the MSG-DCGAN struggles with classifying correlated features, resulting in little to no differentiation between male and female faces in the case of the Celeba dataset.

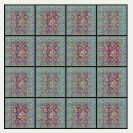


DCGAN without Multi-Scale Gradients, Epoch 50

In comparison, the regular DCGAN mostly succeeds in male-female differentiation, and produces images without the wavelike contours. Small occurrences of lattice structure are still present in end result images produced both with and without Multi-Scale Gradients.

Transfer learning

To apply transfer learning, the regular DCGAN models were trained for 200 epochs on the abstract art dataset, before being saved and loaded onto a new DCGAN-instance with Bored Ape Yacht Club as targeted training set.



DCGAN on BAYC dataset, epoch 1

DCGAN BAYC dataset, epoch 50

Abstract art model on BAYC dataset, epoch 1

The transfer learned model at its first epoch outperforms the untrained model, and produces results comparable to the untrained model at epoch 50.

Bored Ape Yacht Club

To test the network's ability to produce images in distinct domains, we tested it on the Bored Ape Yacht Club which consists of digital illustrations. In comparison with photographs, which the Abstract Art and CelebA datasets consist of, the DCGAN struggled to produce unique results with clearly defined features. The images also contain more noise than the ones generated from photographs.

Discussion

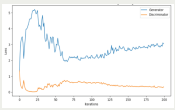
DCGAN vs MSG-DCGAN

Images generated by our DCGAN implementation were satisfactory. Little to no tweaking of hyperparameters where necessary when switching datasets. This only became a factor when generating higher resolution images. In addition to the architecture having a lower complexity than the MSG-DCGAN, it is also better documented. Having several available implementations to learn from, and informative articles from reputable sources may also be the reason that our DCGAN produced images that exceeded the images produced with Multi-Scale Gradients.

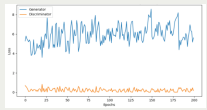
While we have benchmarked the DCGAN and the MSG-DCGAN by comparing image results, this may be a misleading approach for comparison. In the MSG-paper, the concept is implemented with complex GAN-architectures such as ProGan and StyleGan, and trained over several days to produce 1024x1024 images. In comparison, we trained our model for a maximum of 12 hours to produce 64x64 and 128x128 images. The intent of the original MSG-architecture was to stabilize training in complex GAN-architectures for high resolution images without having to fine tune hyperparameters. It might be that by applying the MSG-concept to the DCGAN-architecture, which is famous for being a recipe for stable and efficient training, we are applying a solution for a problem we do not have. Another factor may be that our target image resolutions were too low to encounter the problems of which MSGs are intended to solve.

Loss graphs

By analyzing the loss graph of the MSG-DCGAN, we can see that the discriminator loss is too low compared to the generator loss. Ideally, the discriminator loss should converge towards 0.5 as it does for the regular DCGAN (Radford, 2016). The MSG-DCGAN seems to be learning slowly and the weight adjustments from back propagation are small. This can be corrected by increasing the learning rate, but too big of a correction makes the model prone to converging on local minimums during gradient descent. This may explain why the MSG-DCGAN mode collapses with no indications of rectification when trained on the abstract art dataset, in conjuction with the dataset's relatively small size.



DCGAN loss evolutions



MSG-DCGAN loss evolutions

Transfer learning

The results demonstrate that the trained models may efficiently be applied to other datasets. As a result, we can examine more datasets with reduced effort compared to training brand new models. The versatility of transfer learning can be seen after a single epoch. This suggests that different datasets contain more similar features than initially anticipated. By exposing a GAN to multiple datasets, it may also improve its versatility and robustness.

Bored Ape Yacht Club

The DCGAN seems to produce better results with real life photographs rather than illustrations. This may be because photographs naturally consist of more noise and that the flaws of its output are better hidden within photographs. The network might also have problems with classifying related features within the Bored Ape Yacht Club dataset as it mainly consists of the same character with varying background and clothing. A different illustration dataset might have improved the generated results.

Conclusion

In this work, we presented our implementation of different types of GANs. Our DCGAN implementation was able to generate decent images mimicking the training data well. The loss values of the MSG-GAN did not indicate intended behavior and the generated images remain lacking compared to those generated by the DCGAN. With respect to transfer learning, the network finds correlations between images of separate domains that makes it able to quickly produce decent images when exposed to new target datasets.

To conclude, our implementation of the DCGAN has been a success, showing decent results using various datasets. It was also further enhanced by transfer learning. The MSG approach might not be optimally paired with the DCGAN-architecture, but the process of implementing it has been highly educational.

Acknowledgments

We would like to thank Espen Bomann Fosseide (Master student at NTNU) for his guidance on GANs, and the idea of implementing an MSG and transfer learning. Finally we extend a special thanks to our lecturers Donn Morrison (Associate professor at NTNU) and Ole Christian Eidheim (Associate professor at NTNU) for providing us with an informative course and an interesting project.