Introduction to CycleGAN

Abdulkader Helwan · Follow
5 min read · Jan 12, 2024

In this article, we discuss the CycleGAN architecture.

Here we discuss the CycleGAN architecture and explain how each architectural component can be implemented.

Introduction

In this series of articles, we'll present a Mobile Image-to-Image Translation system based on a Cycle-Consistent Adversarial Networks (CycleGAN). We'll build a CycleGAN that can perform unpaired image-to-image translation, as well as show you some entertaining yet academically deep examples. We'll also discuss how such a trained network, built with TensorFlow and Keras, can be converted to TensorFlow Lite and used as an app on mobile devices.

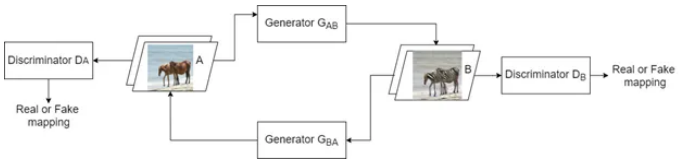
We assume that you are familiar with the concepts of Deep Learning, as well as with Jupyter Notebooks and TensorFlow. You are welcome to download the project code.

In the previous article of this series, we discussed the concepts of conditional generative adversarial networks (CGAN). In this article, we'll have a closer look at the CycleGAN structure, explain how it works, and show how it can be implemented using TensorFlow and Keras.

CycleGAN Model

A CycleGAN model enables training a deep convolutional neural network (deep CNN) to perform image-to-image translations by mapping input and output images from unpaired datasets. The network learns how to map an input image to an output image using a training set of images.

The typical CycleGAN architecture includes two generators and two discriminators that work in sync to map images from the source domain to the target domain and vice versa using the same model. The diagram below shows a simple CycleGAN architecture.



CycleGAN. Draw by author

We'll use the horse-to-zebra translation as an example to explain the working mechanism and training of the CycleGAN.

As you can see in the above diagram, the network has two generators (GAB and GBA) and two discriminators (DA and DB). GAB generates images for the first domain (Domain A). In other words, it transforms images from Domain A to Domain B — from horses to zebras. The second generator, GBA, generates images for the second domain (Domain B). In other words, it transforms images from Domain B to Domain A — from zebras to horses.

Each generator has a corresponding discriminator that detects whether the generated image is real or fake. DA detects images generated by GAB, while DB detects images from GBA.

So we have two mapping functions — G and F:

- G: A→B
- F: B→A

A represents images from Domain A, and B represents the desired output images from Domain B.

CycleGAN Generator

A CycleGAN generator is an autoencoder that takes an input image, extracts features from it, and generates another image. The generator network consists of three main stages:

- Encoder (convolutional block)
- Transformer (residual block)
- Decoder (transposed convolutional block)

The encoder stage includes three 2D convolutional layers followed by an instance normalization layer and an activation function (ReLU). The encoder extracts features from input images using convolutions, reducing the representation by 25 percent of the input image size.

The encoder output passes through the transformer stage which mainly consists of 6 to 9 residual blocks. Each block is a set of 2D convolutional layers, with every two layers followed by an instance normalization layer with a momentum.

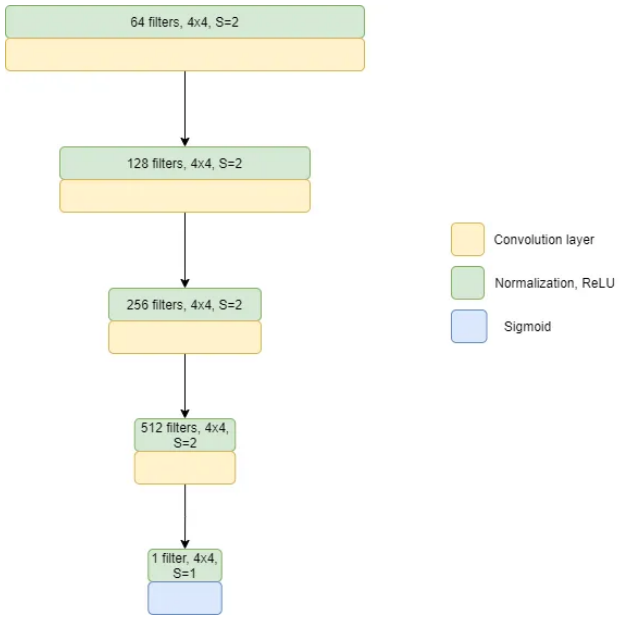
The transformer output will pass through the decoder stage, which consists of two upsampling blocks. Each upsampling block is a transposed convolution layer followed by a ReLU activation function. These two deconvolution blocks increase the size of representation of the processed images coming from the transformer to its original value.

The generator features a final 2D convolutional layer that uses the tanh activation function. This layer allows the generation of images of size equal to the size of the original input images.

CycleGAN Discriminator

A CycleGAN discriminator is a typical CNN that includes multiple convolutional layers. This network takes an input image and classifies it as real or fake. The CycleGAN discriminator is different from that used in the regular GAN. The latter maps input from a 256 by 256 image to a single scalar

output, which represents “real” or “fake.” The CycleGAN discriminator maps from the 256 by 256 image to an N by N array of outputs X. In that array, each Xij signifies whether the patch ij in the image is real or fake. The diagram below shows the typical architecture of a CycleGAN discriminator.



CycleGAN discriminator. By author

CycleGAN Loss Functions

A CycleGAN has two loss functions:

- Adversarial Loss
- Cycle-Consistency Loss

Adversarial Loss: This loss is similar to the one used in the regular GAN. However, in CycleGAN, adversarial loss is applied to both generators that are trying to generate images of their corresponding domains. A generator aims to minimize the loss against its discriminator in order to finally generate real images. The adversarial loss is calculated as follows:

$$Loss_{advers}\left(G, D_y, X, Y\right) = \frac{1}{m}\sum\left(1 - D_y\left(G\left(x\right)\right)\right)^2$$
$$Loss_{advers}\left(F, D_x, Y, X\right) = \frac{1}{m}\sum\left(1 - D_x\left(F\left(y\right)\right)\right)^2$$

Where G is the “X to Y” domain, and F is the inverse “Y to X” domain. Dx and Dy are the discriminators.

Cycle-Consistency Loss: To understand this loss, we should first understand the cycle-consistency approach practiced in the CycleGAN. This approach was first proposed in the CycleGAN article, and it represents the inverse mapping, F: Y→X. Cycle-consistency was meant to make the inverse mapping of a mapped image produce the same results as the original image. In other words, if we want to transform an image from horse to a zebra, and then translate it back from zebra to horse, we should get the initial image.

CycleGAN uses two cycle-consistency losses to regularize the mappings:

- Forward Cycle-Consistency Loss

- Backward Cycle-Consistency Loss

These two are calculated in order to make sure that if an image is translated from one domain to another (A to B) and then back (B to A), the same results will be obtained.

Next Steps

In the next article, we'll show you how to implement a CycleGAN using the Keras framework. Stay tuned!

If you like the article and would like to support me make sure to:
📰 View more content on my medium profile
🔔 Follow Me: LinkedIn | Medium | GitHub | Facebook

👏Clap for this article
🚀👉 Read more related articles to this one on Medium and AI-ContentLab

Cyclegan Gans Adversarial Network Python Style Transfer

Written by Abdulkader Helwan

712 Followers

Research Scientist, AI in Medicine, Technical Reviewer at ContentLab.io.
https://bio.site/AbdulkaderH