Notes

Ganics

Adrian Chmiel

June 9, 2024

1 Introduction

This file contains various notes that helped me learn more about how different Generative Adversarial Networks work and what particular layers do. It is a summary of the most important information I found during the research.

2 Generative Adversarial Networks

2.1 Basic GAN

- one generator and one discriminator
- training on paired data aiming to generate realistic general images

2.2 CycleGAN

- two generators and two discriminators
- training on unpaired data aiming for image style conversion
- **cycle consistency loss** - an image translated from A to B and then back from B to A should be similar to the original image

2.3 SRGAN

- one generator and one discriminator
- training aiming to generate high-quality images
- **residual connections** - passing information from previous layers to subsequent layers bypassing intermediate layers

2.4 PatchGAN Discriminator

- discriminator evaluating images at the patch level
- instead of evaluating the entire image, it evaluates individual image patches
- allows for a more detailed evaluation of images
- often used as part of *SRGANs*

3 Layers *tf.keras*

3.1 Sequential

- allows for creating layers in the order they are added
- a simple way to build models in *tf.keras*
- for building models where layers are applied one after the other in a linear manner

3.2 Concatenate

- concatenates a list of tensors along a specified axis
- for combining outputs from different layers, features from earlier layers are combined with features from later layers

3.3 Conv2D

- 2D convolutional layer, used for processing image data
- applies convolutional filters that move across input data to extract features
- for feature extraction from images, such as edges, textures, etc.

3.4 Conv2DTranspose

- inverse 2D convolutional layer, used for generating higher resolution images from lower resolution ones
- for increasing the resolution of images
- for reversing convolution operations

3.5 ZeroPadding2D

- adds zero padding around the edges of input data
- allows for controlling the size of the output image after applying a convolutional layer
- for maintaining spatial size of input data after convolution

3.6 LeakyReLU

- a variant of the ReLU activation function that allows for a small gradient when the unit is not activated (negative value)
- for preventing the "vanishing gradients" problem

3.7 InstanceNormalization / BatchNormalization

- normalizes input data for each sample independently
- allows for stabilizing the training process and rapid convergence
- for normalizing input features in neural networks, especially in generative models and image stylization

3.8 ResizeLayer

- resizes input data to a specified dimension
- for resizing images in neural networks (for standardizing input or output image sizes)