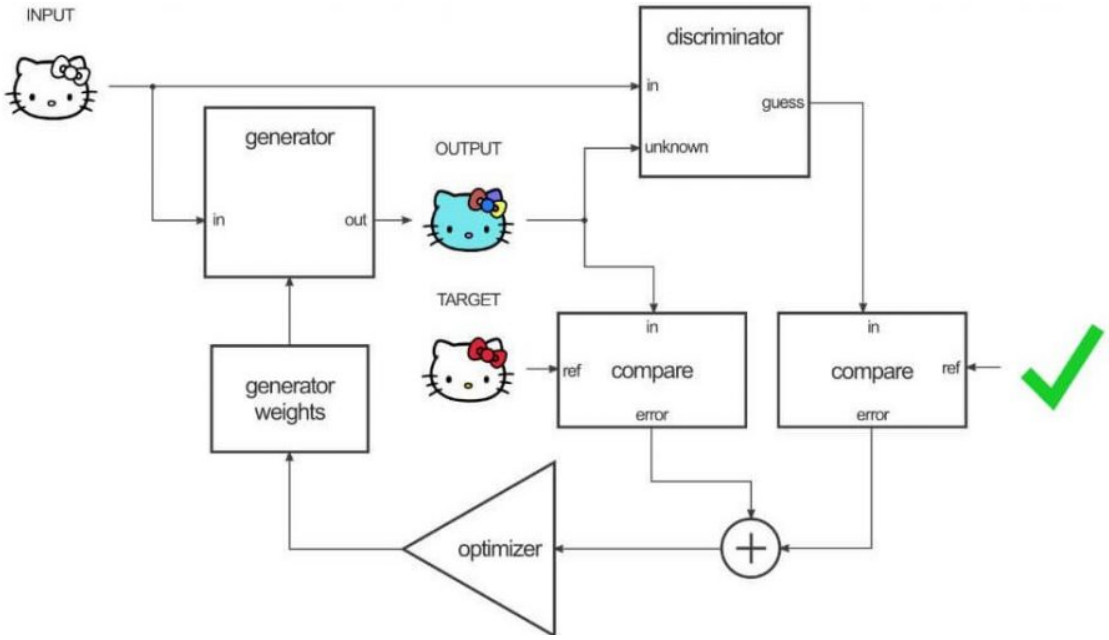
Pix2Pix – Image-to-Image Translation Neural Network

27 November 2018



Pix2pix architecture was presented in 2016 by researchers from Berkeley in their work “Image-to-Image Translation with Conditional Adversarial Networks.” Most of the problems in image processing and computer vision can be posed as “translating” an input image into a corresponding output image. For example, a scene may be rendered as an RGB image, a gradient field, an edge map, a semantic label map, etc. In analogy to automatic language translation, we define automatic image-to-image translation as **the task of translating one possible representation of a scene into**

another, given a large amount of training data. But with the rise of deep learning, CNN becomes the common workhorse behind a wide variety of image prediction problems.

The paper: https://arxiv.org/pdf/1611.07004.pdf

DataSet

- Cityscapes (which is a large-scale dataset that contains a diverse set of stereo video sequences recorded in street scenes from 50 different cities, with high-quality pixel-level annotations of 5 000 frames. The dataset is thus an order of magnitude larger than similar previous attempts)datasets with 2975 training images from the Cityscapes training set trained for 200 epochs, with random jitter and mirroring. Validation set has been used for testing.
- Maps aerial photograph 1096 training images scraped from Google Maps, trained for 200 epochs, batch size 1. Data was then split into train and test set (with a buffer region added to ensure that no training pixel appeared in the test set).
- BW-> color 1.2 million training images (Imagenet training set), trained for 6 epochs, batch size 4, with only mirroring, no random jitter. Tested on a subset of Imagenet validation set.
- Edges ->shoes 50k training images from UT Zappos50K dataset trained for 15 epochs, batch size 4. Data were split into train and test randomly.

Pix2pix architecture

Pix2pix uses a **conditional generative adversarial network** (cGAN) to learn a function to map from an input image to an

output image. The network is made up of two main pieces, the Generator, and the Discriminator. The Generator transforms the input image to get the output image.
The Discriminator measure the similarity of the input image to an unknown image (either a target image from the dataset or an output image from the generator) and tries to guess if this was produced by the generator.

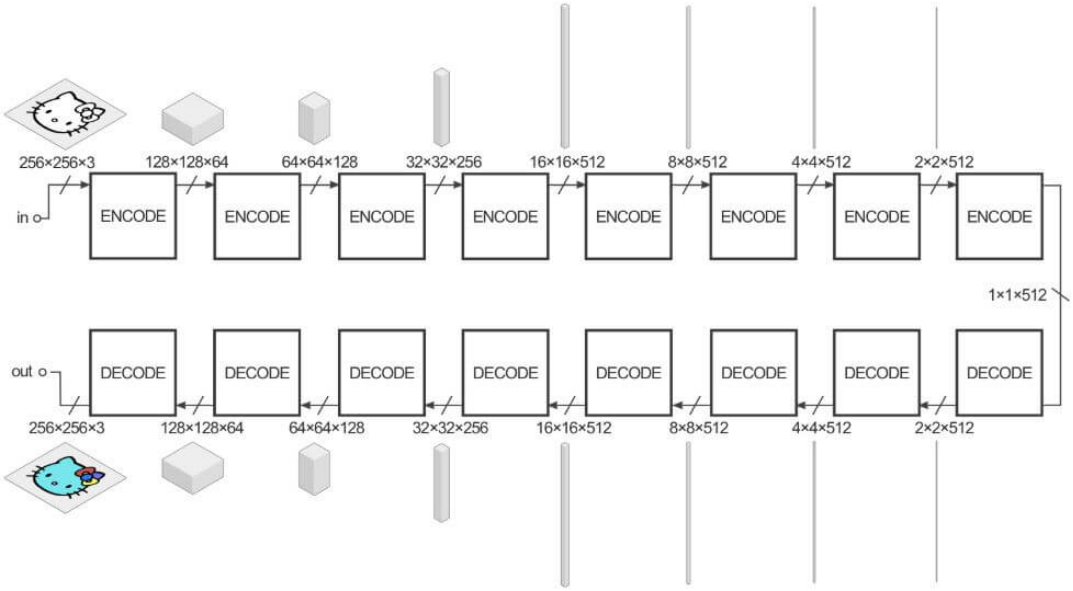
Generator

The Generator has the job of taking an input image and performing the transform to produce the target image. The encoder-decoder architecture consists of:

encoder: *C64-C128-C256-C512-C512-C512-C512-C512*

decoder: *CD512-CD512-CD512-C512-C256-C128-C64*

An example input could be an image (black and white), and the output of that image is to be a colorized version. The structure of the generator is called an “encoder-decoder,” and in pix2pix the encoder-decoder looks more or less like this:



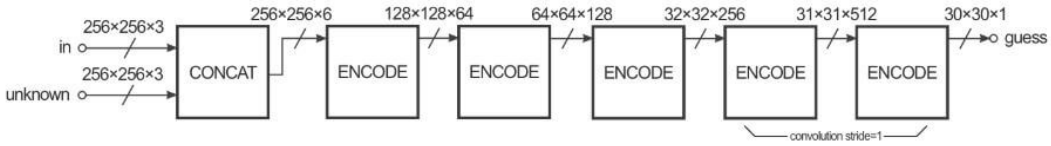
A convolution is applied after the last layer in the decoder to map the number of output channels (3 in general, except in

colorization, where it is 2), followed by a Tanh function. Batch-Normalization is not applied to the first C64 layer in the encoder. All ReLUs in the encoder is leaky, with slope 0.2, while ReLUs in the decoder is not leaky. The U-Net architecture is identical except with skip connections between each layer *i* in the encoder and layer *n-i* in the decoder, where *n* is the total number of layers. The skip connections concatenate activations from layer *i* to layer *n-i.*

Discriminator

The Discriminator has the job of taking two images, an input image and an unknown image (which will be either a target or output image from the generator), and also decide if the other image was produced by the generator or not. The 70×70 discriminator architecture is:

C64-C128-C256-C512



A convolution is applied after the last layer to map to a 1-dimensional output, followed by a Sigmoid function. BatchNorm is not applied to the first C64 layer. All ReLUs are leaky, with slope 0.2. All other discriminators follow the same basic architecture, with depth varied to modify the receptive field size:

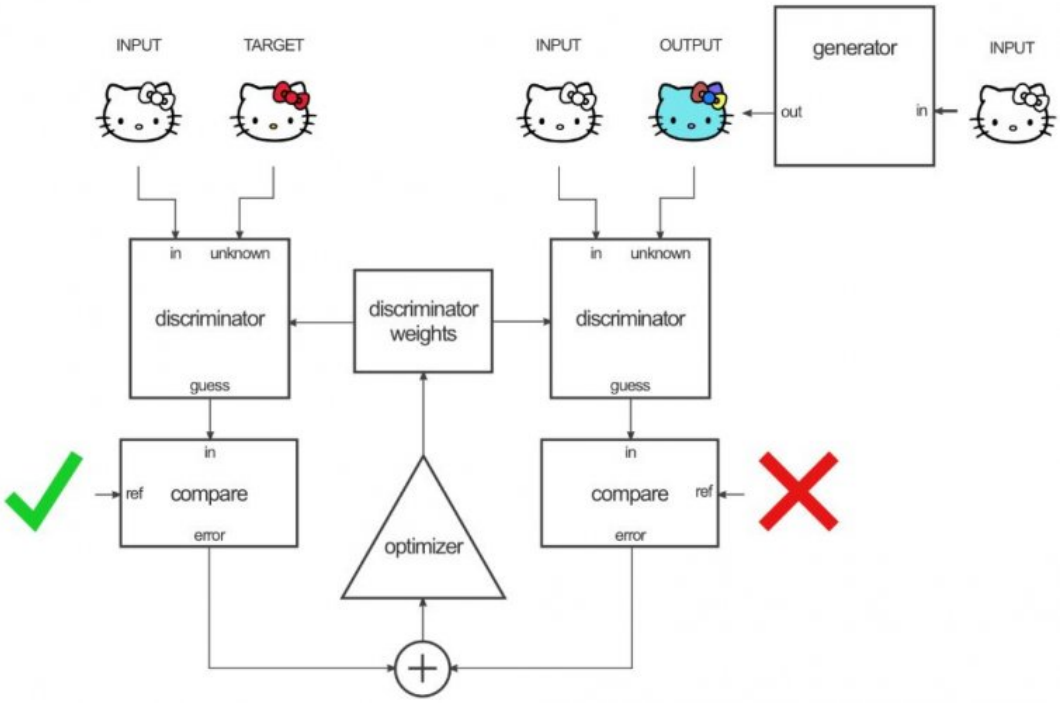
- 1 x1 discriminator: *C64-C128* (note, in this special case, all convolutions are1 x1 spatial filters)
- 16×16 discriminator: *C64-C128*
- 286×286 discriminator: *C64-C128-C256-C512-C512-C512*

In the pix2pix implementation, each pixel from this 30×30 image corresponds to the believability of a 70×70 patch of the input image (the patches overlap a lot since the input images are 256×256). The architecture is called a “PatchGAN”.

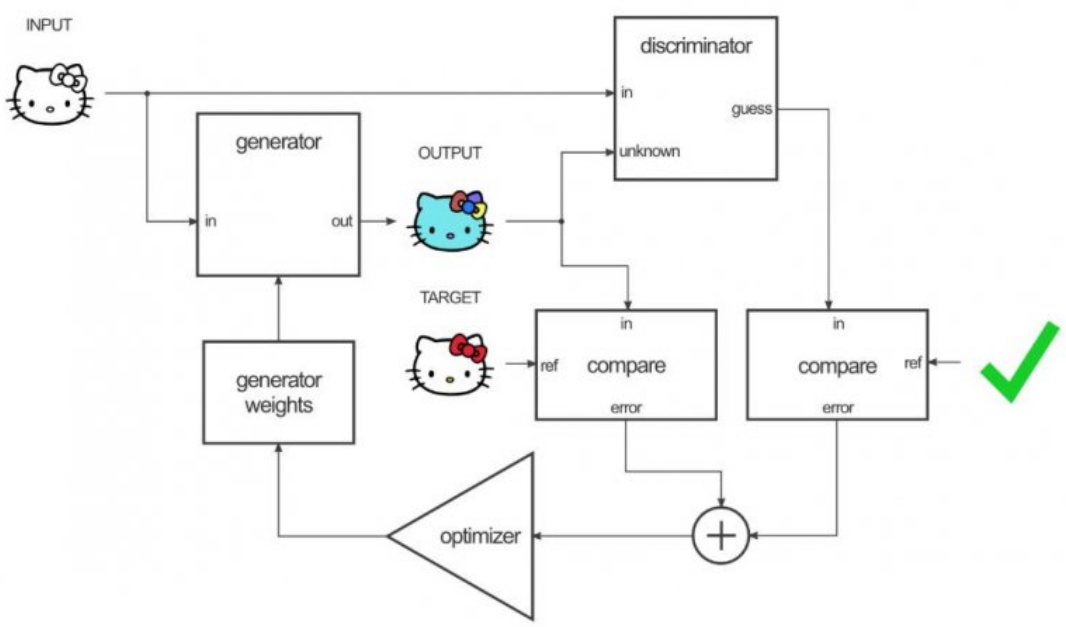
Training

For training the network, two steps have to be taken: training the discriminator and training the generator. All networks were trained from scratch. Weights were initialized from a Gaussian distribution with mean 0 and standard deviation of 0.02.

First, the generator generates an output image. The discriminator looks at the input/target pair and the input/output pair and produces its guess about how realistic they look. The weights vector of the discriminator is then adjusted based on the classification error of the input/output pair and the input/target pair.



The generator’s weights are then adjusted based on the output of the discriminator as well as the difference between the output and the target image.



The thing to remember here, when the generator is trained on the output of the discriminator, actually, the gradient is calculating through the discriminator, which means that while the discriminator improves, you're training the generator to beat the discriminator. If the discriminator is good at its job and the generator is capable of learning the correct mapping function through gradient descent, you should get generated outputs that could fool a human.

The objective function, we want to minimize during training is of conditional GAN, which can be expressed as:

$$\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x,z))],$$

Use Cases and Implementation

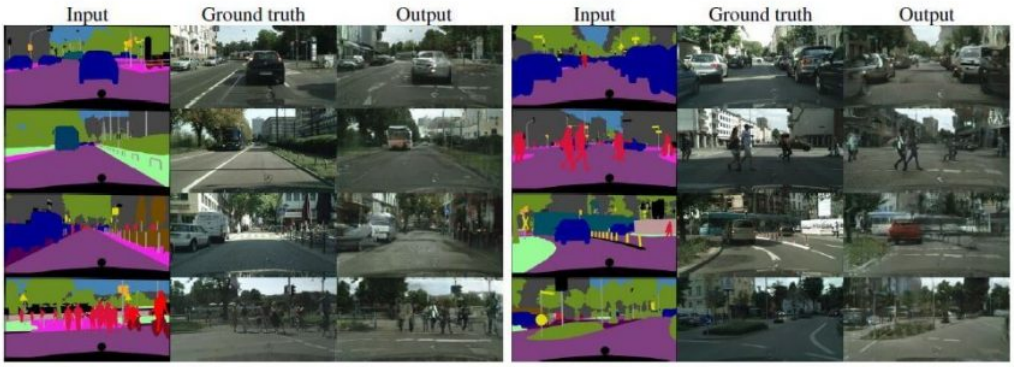
The nice thing about pix2pix is that it is not specific to some class i.e. it is generic. It does not require to define any relationship between the two types of images. It makes no assumptions about the relationship and instead learns the objective during training, by comparing the defined inputs and outputs during training and inferring the objective. This

makes pix2pix highly adaptable to a wide variety of situations, including ones where it is not easy to verbally or explicitly define the task we want to model.

[Tensorflow][Pytorch][Keras]

Result

Pix2pix suggest that conditional adversarial networks are a promising approach for many image-to-image translation tasks, especially those involving highly structured graphical outputs. These networks learn a loss adapted to the task and data at hand, which makes them applicable to a wide variety of settings. The result achieved by pix2pix is state of the art.



Example results of our method on Cityscapes labels->photo, compared to ground truth

Example results of our method on automatically detected edges -> shoes, compared to ground truth.

Author: Muneeb ul Hassan

Source: https://arxiv.org/pdf/1611.07004.pdf

Github: https://github.com/affinelayer/pix2pix-tensorflow

You Might Also Like

The StyleGAN Code Released: Neural Network for Faces Generation by NVIDIA

NVIDIA Research Proposed New Style-Based Generator Architecture for GANs

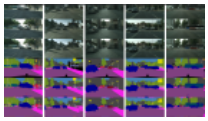
Dissecting GANs for Better Understanding and Visualization

Be the first to comment.

Recent posts

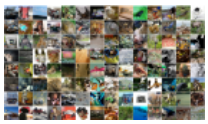
OpenAI's MuseNet AI Can Generate Novel 4-Minute Music Compositions

29 April 2019



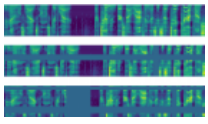
"Segmenting The Future": New Method Predicts Segmentation Masks From Past Video Frames

26 April 2019



OpenAI Released Sparse Transformer Models

25 April 2019



SpecAugment: New and Simple Data Augmentation Technique for Audio Data

24 April 2019

Algorithm Approximates Images Using Geometric Shapes

23 April 2019

Facebook’s AI System Can Convert One Singer’s Voice Into Another

20 April 2019

Up-to-date research in the field of neural networks: machine learning, computer vision, nlp, photo processing, streaming sound and video, augmented and virtual reality.