GAN –Variant's

Building GAN where we are able to control the output so called conditional GAN

Motivation for CGAN

Compare GAN vs CGAN

The CGAN will pass extra information to the generator and critic relating to the label

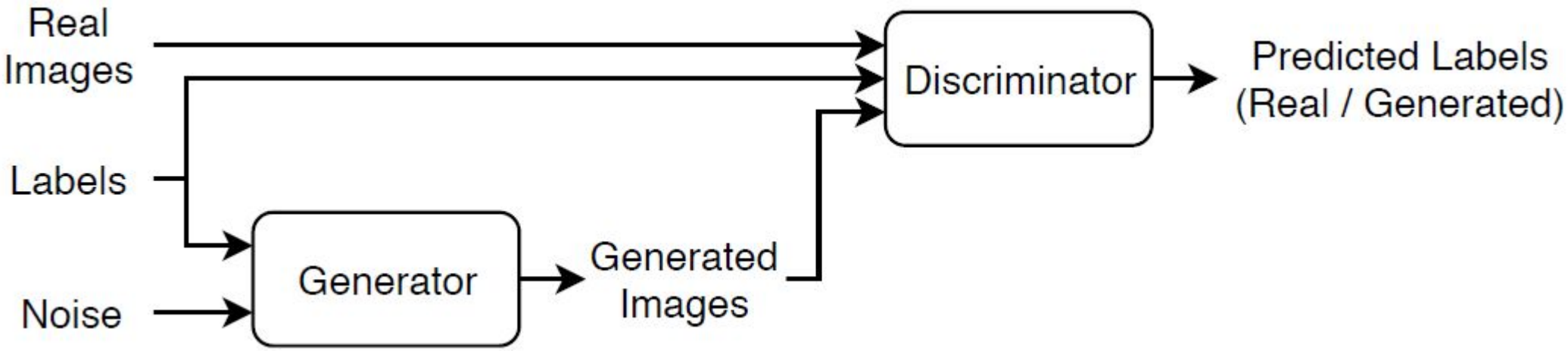
In the generator this is simply appended to the latent space sample as a one hot encoded vector.

In the critic we add the label information as extra channel's to the RGB image. We do this by repeating the one hot encoded vector to fill the sample shape as the input images

CGAN work because the critic now has access to extra information regarding the content of the image so the generator must ensure that its output agrees with the provided label in order to fool critic.

If the generator produced perfect images that disagreed With the image label the critic would be able to tell that They were fake simply because the images and labels did not match.

Conditional GAN



A conditional generative adversarial network (CGAN) is a type of GAN that also takes advantage of labels during the training process.

Generator — Given a label and random array as input, this network generates data with the same structure as the training data observations corresponding to the same label.

Discriminator — Given batches of labeled data containing observations from both the training data and generated data from the generator, this network attempts to classify the observations as "real" or "generated".

Note

Fashion MNIST dataset : 10 different fashion item

One hot encoding label vector of length 10

Incorporated to input of generator and 10 additional one hot encoded label channels into the input of the critic.

Only change is to concatenate the label information to existing inputs of generator and the critic

Analysis of CGAN

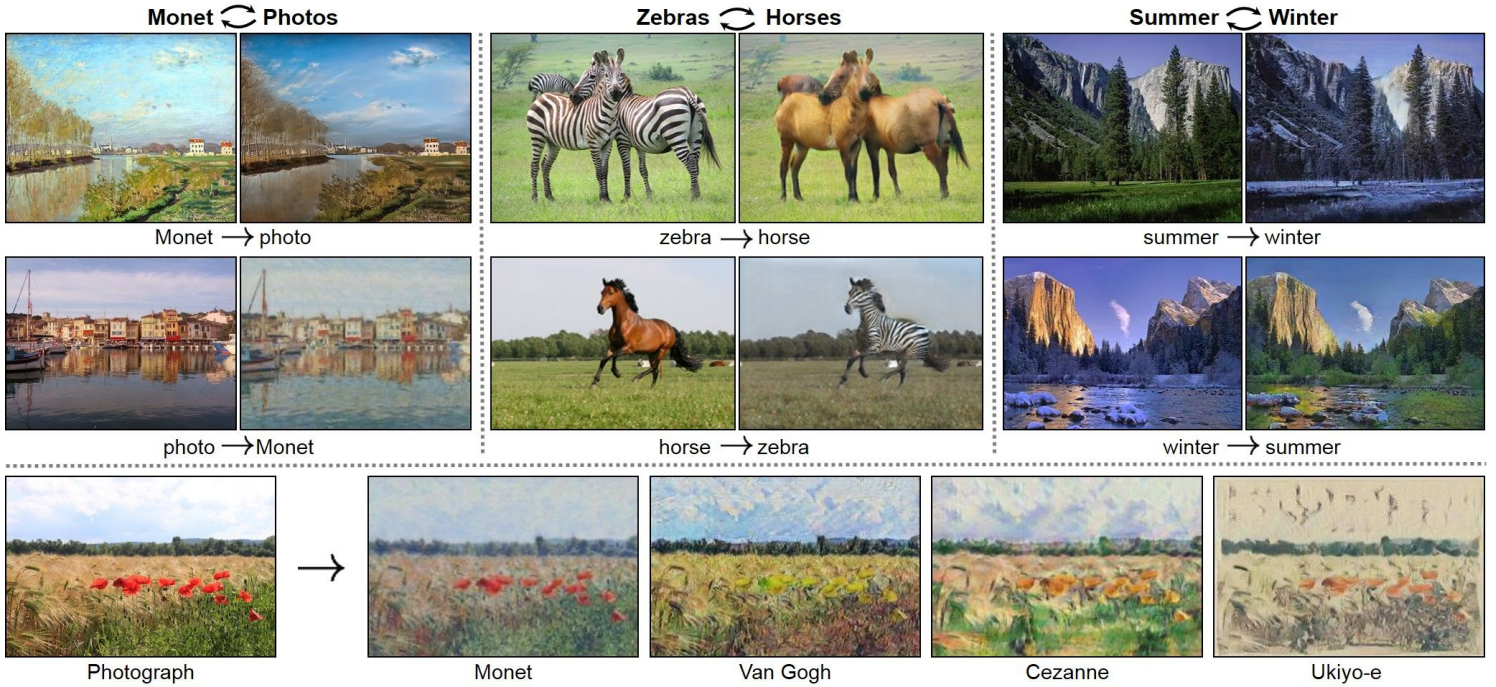
- We keep the random latent vector same across the examples and change only the conditional label vector
- For e.g. [1,0] generate face with nonblond hair
- [0,1] face with blond hair

- CGAN is able to control only hair attribute of the image using label vector

cycle GAN

CycleGAN is a powerful model for generating new images from existing ones.

It uses two generative adversarial networks (GANs) to learn the mapping between two domains of images. The model is trained to capture the characteristics of the target domain and to generate new images from the source domain that share the same characteristics.



Cycle GAN

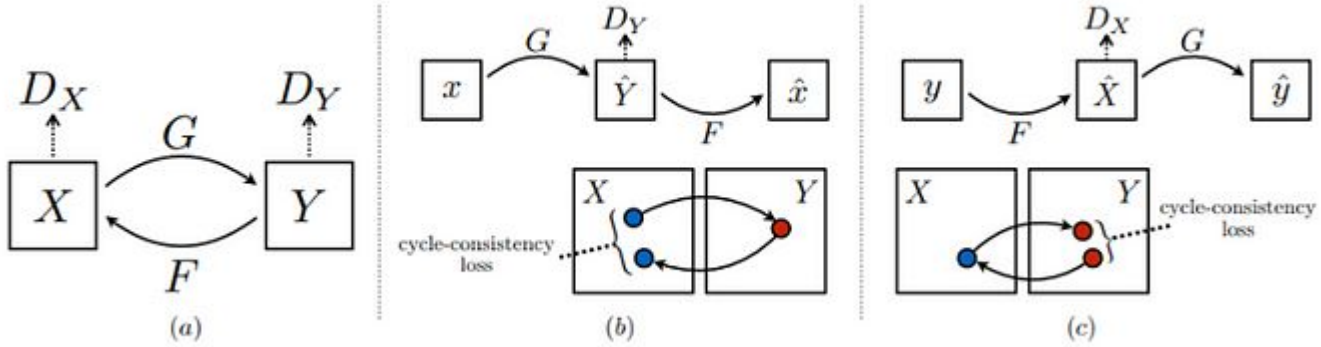
The model is initialized by training the two GANs separately

First, the generator of the first GAN is trained to generate a target domain image from a source domain image. The discriminator of the first GAN is trained to identify real target domain images from generated target domain images.

the generator of the second GAN is trained to generate a source domain image from a target domain image.

The discriminator of the second GAN is trained to detect real source domain images from generated source domain images. Once the two GANs have been trained separately, the CycleGAN is defined.

- The generators of the two GANs form the generators of the CycleGAN, and the discriminators of the two GANs form the discriminators of the CycleGAN. The model is then trained to generate an image from the source domain that is similar to an image from the target domain and vice versa.



CycleGANs (Cycle Generative Adversarial Networks) are a type of deep learning architecture used for image-to-image translation tasks. Unlike standard image-to-image models that require paired training data (where each input image has a corresponding desired output image), CycleGAN can learn the mapping between domains using unpaired datasets. This makes them particularly useful for tasks where obtaining perfectly paired d is difficult or expensive.

Two Generators (G and F):
Generator G takes an image from the first domain (X) and translates it into an image that resembles the second domain (Y).
Generator F takes an image from the second domain (Y) and translates it back to the first domain (X').
Two Discriminator Networks (X and Y):
Discriminator X differentiates between real images from domain X and the generated images (X') that F produces (supposed to be from X).
Discriminator Y differentiates between real images from domain Y and the generated images (Y) that G produces (supposed to be from Y).

Application of cycle GAN

The CycleGAN is trained using a cycle-consistent loss, which encourages the model to generate images that are indistinguishable from real images from the target domain

The model is further improved by adding an identity mapping component. This component helps to preserve the characteristics of the source domain images when they are translated to the target domain.

The top 5 CycleGAN applications

Image-to-image translation: from sketches to photographs

Domain adaptation:This is often used to adapt models trained on synthetic data to real data.

Data augmentation::This can be used to generate new data from existing data. This can be useful for increasing the size of a training dataset.

Denoising: This can be used to remove noise from data. This is often used in image and audio applications.

Overall Objective:

The overall objective of CycleGAN training is to minimize a combined loss function that includes both the adversarial loss (fooling the discriminators) and the cycle consistency loss. By minimizing this combined loss, the generators learn to translate images between domains while ensuring the translated images maintain their content and structure.

Training Process:

Generator Training: Both generators are trained simultaneously in an adversarial fashion:
Generator G tries to generate images (Y) from domain X that fool discriminator Y into believing they are real images from domain Y.
Generator F tries to generate images (X') from domain Y that fool discriminator X into believing they are real images from domain X.
Cycle Consistency Loss: A key aspect of CycleGANs is the cycle consistency loss. This loss encourages the generated images to be able to be translated back to the original domain while maintaining the content.
For example, if G translates an image from X to a seemingly realistic image in Y, F should then be able to translate that generated Y image back to an image in X' that closely resembles the original image in X. This enforces consistency prevents the generators from creating nonsensical outputs.