Session 6 - Generative Adversarial Networks

Due Sep 26 by 11:59pm **Points** 3,000 **Submitting** a text entry box or a website url
Available Sep 5 at 9am - Sep 26 at 11:59pm 22 days

This assignment was locked Sep 26 at 11:59pm.

Session 6 - GANs

WHAT ARE GANS?

Adversarial: (of a trial or legal proceedings) in which the parties in a dispute have the responsibility for finding and presenting evidence

GANs

- are DNNs comprised of two nets, pitted one against the other (thus adversarial)
- Generative Adversarial Networks are deep neural net architectures comprised of two neural networks, competing one against the other (adversarial = competing).
- GANs are neural networks that are trained in an adversarial manner to generate data mimicking some distribution.

GAN Applications

- Generate Examples for Image Datasets
- Generate Photographs of Human Faces
- Generate Realistic Photographs
- Generate Cartoon Characters
- Image-to-Image Translation
- Text-to-Image Translation
- Semantic-Image-to-Photo Translation
- Face Frontal View Generation
- Generate New Human Poses
- Photos to Emojis
- Photograph Editing
- Face Aging
- Photo Blending
- Super Resolution
- Photo Inpainting
- Clothing Translation
- Video Prediction
- 3D Object Generation

Two great practical examples of GAN use are:

- Google Brain Project for Creating a Method of Encryption. The project was to send an encrypted message which cannot be intercepted by anyone.
- Drug Research, can be trained on existed drugs and suggest new synthetic chemical structures that improve upon existing drugs.

There are two kinds of models in machine learning:

1. Discriminative model: they discriminate between two classes of data (for example hot-dog, not-hot-dog, i.e. 0/1). They fall into the category of classification problems.
2. Generative models: they are trained on training data X sampled from a distribution D, which given some random distribution z produces a distribution D' which is close to D according to some closeness metrics.

DEEP GENERATIVE MODELS

- AutoEncoders
- Variational AutoEncoders
- Ganerative Adversarial Networks
- Adversarial AutoEncoders
- VAE/GAN
- Adversatial Domain Adaption

OVERVIEW OF DIFFERENT GENERATIVE MODELS

AUTOENCODERS

VARIATIONAL AUTOENCODERS

GANs

GENERATIVE ADVERSARIAL NETWORKS

GANs were introduced in a paper (https://arxiv.org/abs/1406.2661) by Ian Goodfellow and other researchers at the University of Montreal, including Yoshua Bengio, in 2014. Referring to GANs, Facebook's AI research director Yann LeCun **called adversarial training (https://www.quora.com/What-are-some-recent-and-potentially-upcoming-breakthroughs-in-deep-learning)** "the most interesting idea in the last 10 years in ML."

GANs' potential is huge, because they can learn to mimic any distribution of data. That is, GANs can be taught to create worlds eerily similar to our own in any domain: images, music, speech, prose. They are robot artists in a sense, and their **output is impressive (https://www.nytimes.com/2017/08/14/arts/design/google-how-ai-creates-new-music-and-new-artists-project-magenta.html)** – poignant even.

In a surreal turn of events, **Christie's sold a portrait (https://www.theverge.com/2018/10/23/18013190/ai-art-portrait-auction-christies-belamy-obvious-robbie-barrat-gans)** for $432,000 that had been generated by a GAN, based on **open-source code written by Robbie Barrat of Stanford (https://github.com/robbiebarrat/art-DCGAN)**. Like most true artists, he didn't see any of the money, which instead went to the French company, Obviously.

GENERATIVE VS DISCRIMINATIVE ALGORITHMS

To understand GANs, you should know how generative algorithms work, and for that, contrasting them with discriminative algorithms is instructive. Discriminative algorithms try to classify input data; that is, given the features of an instance of data, they predict a label or category to which that data belongs.

A Discriminative algorithm could predict whether a given image is Dog or a Cat, and it does that by learning the features that constitute the input image. So a DA maps features to labels. They are concerned solely with that correlation.

Generative algorithms, on the other hand, try to do the opposite. Instead of predicting a label given a certain image, they attempt to predict the image given a certain label.

Another way to think about it is to distinguish discriminative from generative like this:

- Discriminative models learn the boundary between classes
- Generative models model the distribution of individual classes

GAN INTERNALS

One neural network, called the generator, generates new data instances, while the other, the discriminator, evaluates them for authenticity; i.e. the discriminator decides whether each instance of data that it review belongs to the actual training dataset or now.

Let's say we're trying to do something banaler than mimic the Mona Lisa. We're going to generate hand-written numerals like those found in the MNIST dataset, which is taken from the real world. The goal of the discriminator, when shown an instance from the true MNIST dataset, is to recognize those that are authentic.

Meanwhile, the generator is creating new, synthetic images that it passes to the discriminator. It does so in the hopes that they, too, will be deemed authentic, even though they are fake. The goal of the generator is to generate passable hand-written digits: to lie without being caught. The goal of the discriminator is to identify images coming from the generator as fake.

discriminator is to identify images coming from the generator as fake.

Here are the steps a GAN takes:

- The generator takes in random numbers and returns an image
- This generated image is fed into the discriminator alongside a stream of images taken from the actual, ground-truth dataset
- The discriminator takes in both real and fake images and returns values between 0 and 1, with 1 representing a prediction of authenticity and 0 representing a fake.

So you have a double feedback loop:

- The discriminator is in a feedback loop with the ground truth of the images, which we know.
- The generator is in a feedback loop with the discriminator.

Actual Example:

Real and Fake are values 1 and 0. We are targeting 0.5.

You can think of a GAN as the opposition of a counterfeiter and a cop in a game of cat and mouse, where the counterfeiter is learning to pass false notes, and the cop is learning to detect them. Both are dynamic; i.e. the cop is in training, too (to extend the analogy, maybe the central bank is flagging bills that slipped through), and each side comes to learn the other's methods in a constant escalation.

Let us consider MNIST

For MNIST, the discriminator network is a standard convolution network that can categorize the images fed to it, a binomial classifier labeling images as real or fake.

The generator is an **inverse convolution network**. While a standard convolution classifier takes an image and down-samples it to produce a soft-max number, the generator model takes a vector of random noise and up-samples it to an image.

The first throws away the data through down-sampling techniques like max-pooling, and the second generates new data.

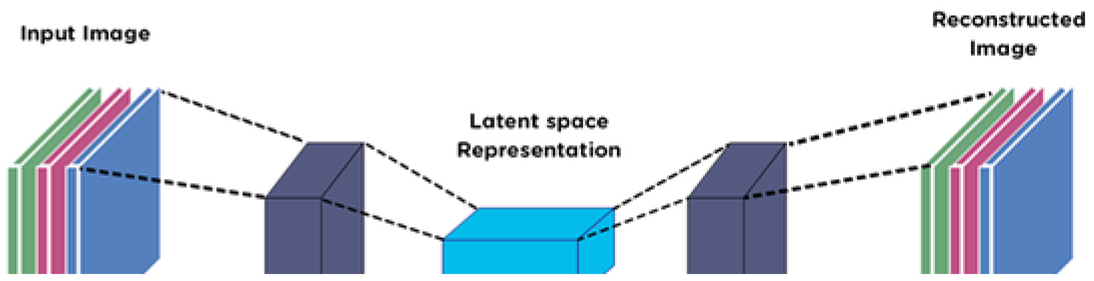
Both networks are trying to optimize a different and opposing objective function, or loss function, in a zero-sum game. This is essentially an actor-critic model. As the discriminator changes its behavior, so does the generator, and vice versa. Their losses push against each other.

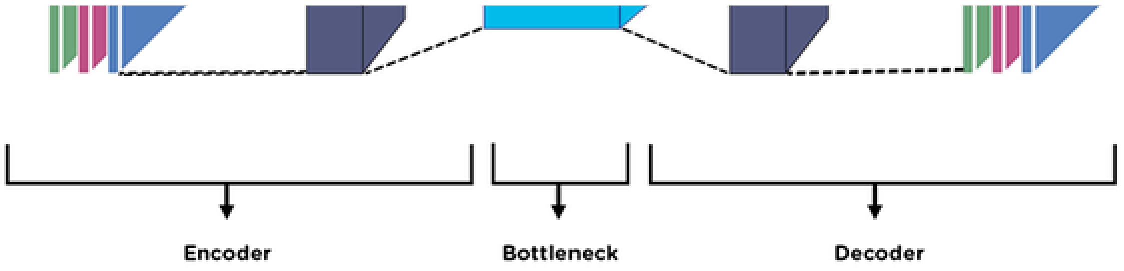
Our Discriminator and Generator can take many DNN forms, most of which we already are aware of:

GANS, AUTOENCODERS AND VAES

It may be useful to compare generative adversarial networks to other neural networks, such as autoencoders and variational autoencoders.

Autoencoders encode input data as vectors. They create a hidden or compressed representation of the raw data. They are useful in dimensionality reduction; that is, the vector serving as a hidden representation compresses the raw data into a smaller number of salient dimensions. Autoencoders can be paired with a decoder, which allows you to reconstruct input data based on its hidden representation.





Variational Autoencoders are generative algorithms that add an additional constraint to encode the input data, namely that the hidden representations are normalized. VAEs are capable of both compressing data like an autoencoder and synthesizing data like a GAN. However, while GANs generate data in fine, granular details, images generated by VAEs tend to be more blurred (this statement changed on [9 June 2019 (https://arxiv.org/pdf/1906.00446.pdf)](https://arxiv.org/pdf/1906.00446.pdf)

Coming back to GAN

When you train the discriminator, hold the generator values constant, and when you train the generator, hold the discriminator constant. You need to train the discriminator first. You also need to train the discriminator more than the generator.

Each side of the GAN can overpower the other. If the discriminator is too good, it will return values so close to 0 or 1, that the generator will struggle to read the gradient. If the generator is too good, it will persistently exploit a weakness in the discriminator that lead to false negatives. This may be mitigated by their respective learning rates.

GANs take a long time to train. On a single GPU a GAN might take hours and on a single CPU more than a day.

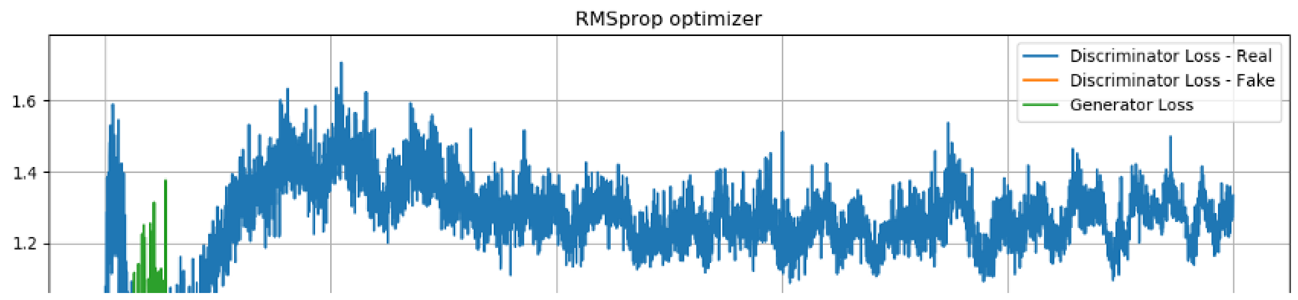
GAN VIA CODE

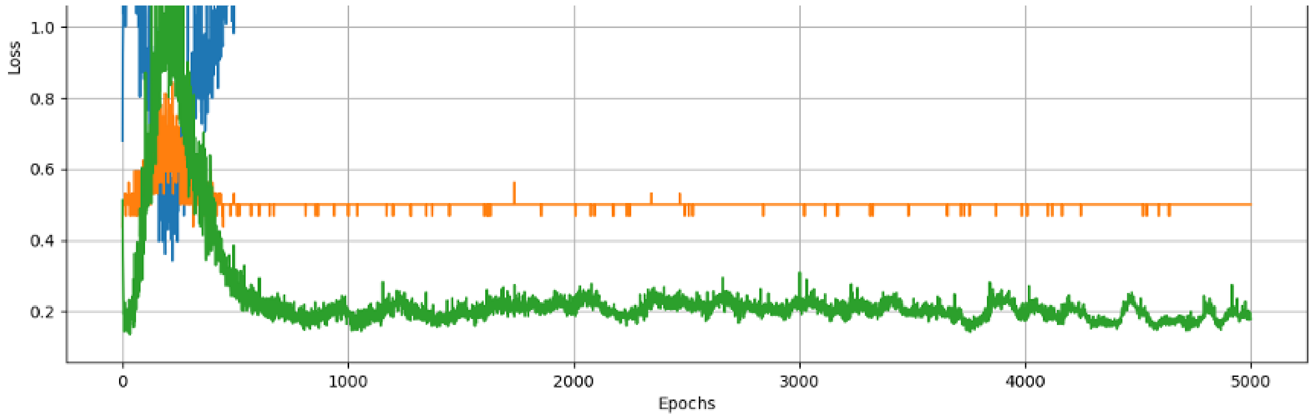
References

1. https://towardsdatascience.com/training-gans-using-google-colaboratory-f91d4e6f61fe (https://towardsdatascience.com/training-gans-using-google-colaboratory-f91d4e6f61fe)
2. https://colab.research.google.com/github/smartgeometry-ucl/dl4g/blob/master/gan.ipynb#scrollTo=NRMryrDVLFrr (https://colab.research.google.com/github/smartgeometry-ucl/dl4g/blob/master/gan.ipynb#scrollTo=NRMryrDVLFrr)
3. https://github.com/Yangyangii/GAN-Tutorial/blob/master/CARS/DCGAN.ipynb (https://github.com/Yangyangii/GAN-Tutorial/blob/master/CARS/DCGAN.ipynb)
4. https://github.com/Yangyangii/GAN-Tutorial/blob/master/CelebA/R1GAN.ipynb (https://github.com/Yangyangii/GAN-Tutorial/blob/master/CelebA/R1GAN.ipynb)

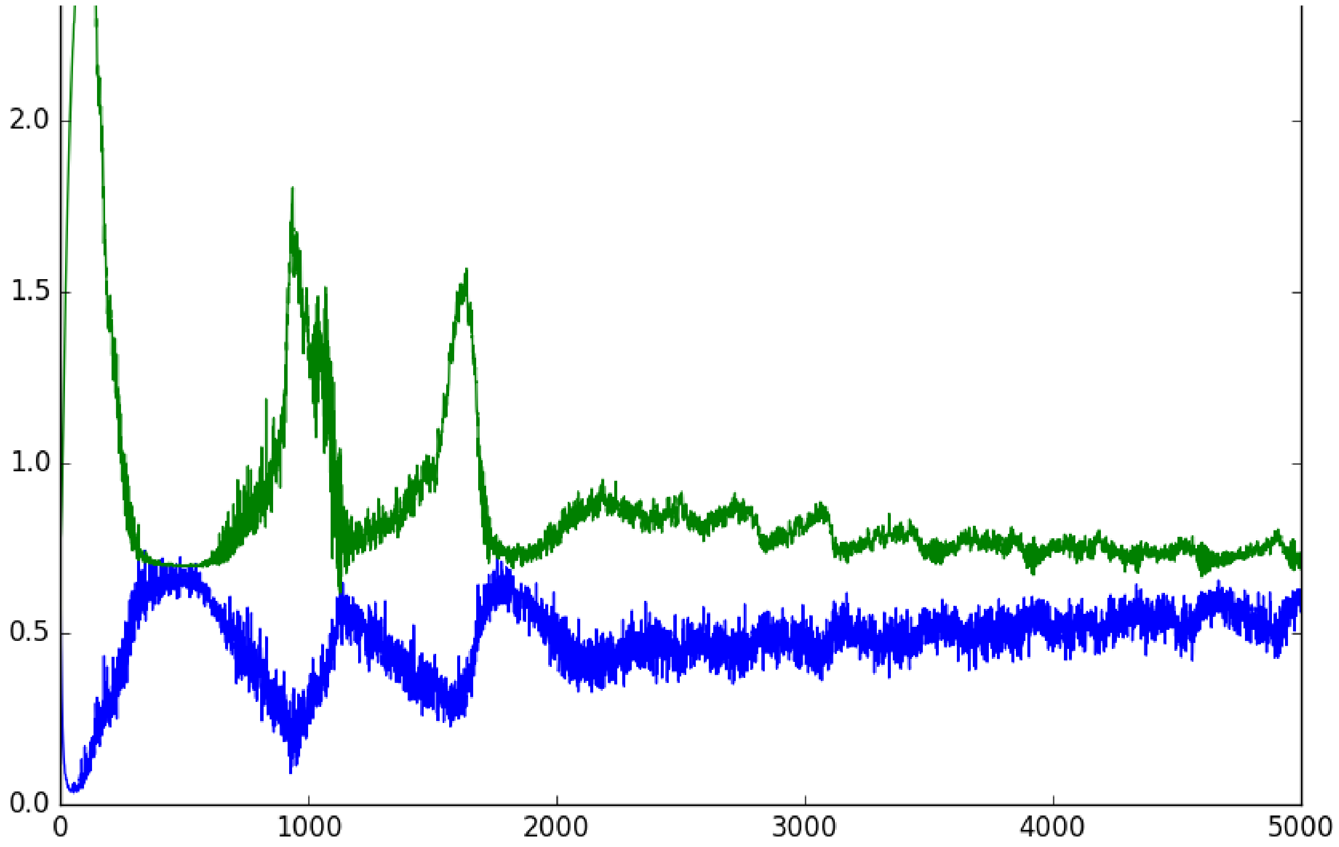
A stable GAN will have a discriminator loss around 0.5, typically between 0.5 and maybe as high as 0.7 or 0.8. The generator loss is typically higher and may hover around 1.0, 1.5, 2.0, or even higher.

The accuracy of the discriminator on both real and generated (fake) images will not be 50%, but should typically hover around 70% to 80%.




Loss
1.0
0.8
0.6
0.4
0.2
0
1000
2000
3000
4000
5000
Epochs

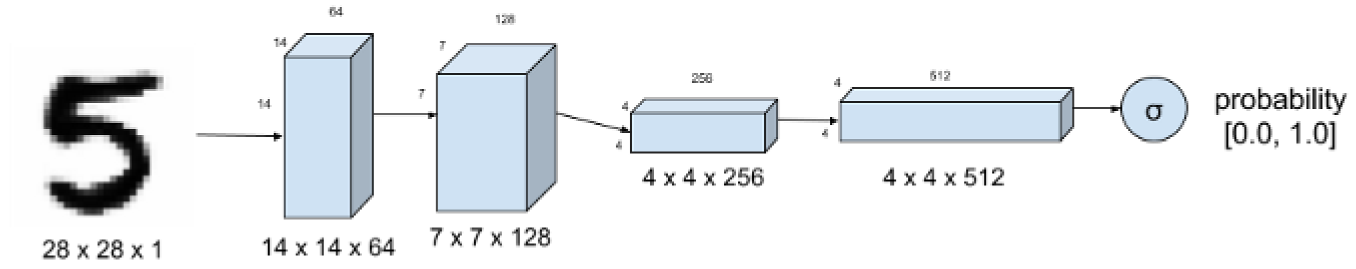
3.0
2.5
discriminitive loss
generative loss

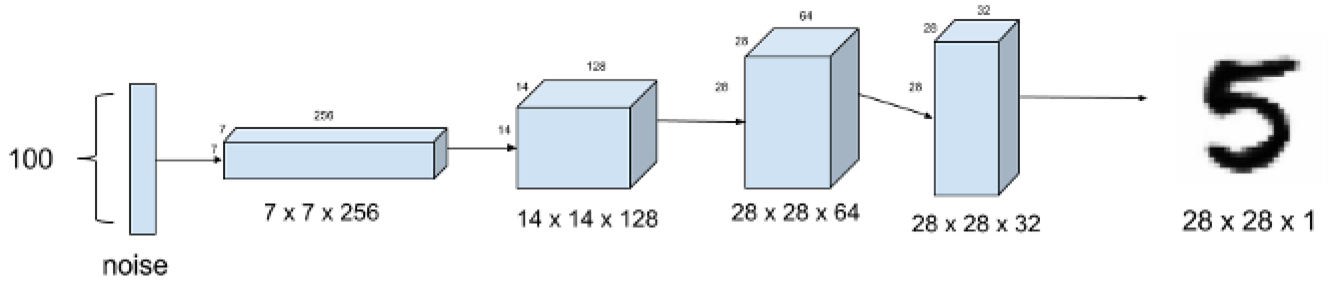


TYPES OF GANS

DEEP CONVOLUTIONAL GAN (DCGAN)

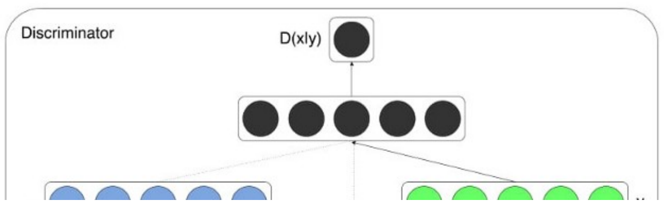
The core idea is we use a convolutional neural network instead of a vanilla neural network at both discriminator and generator

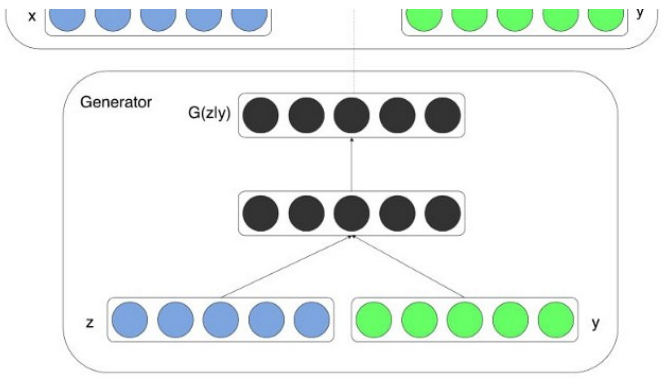




CONDITIONAL GAN

The core idea is to train a GAN with a conditioner, e.g. for MNIST data, we provide the label as well.





AUXILLARY GANS

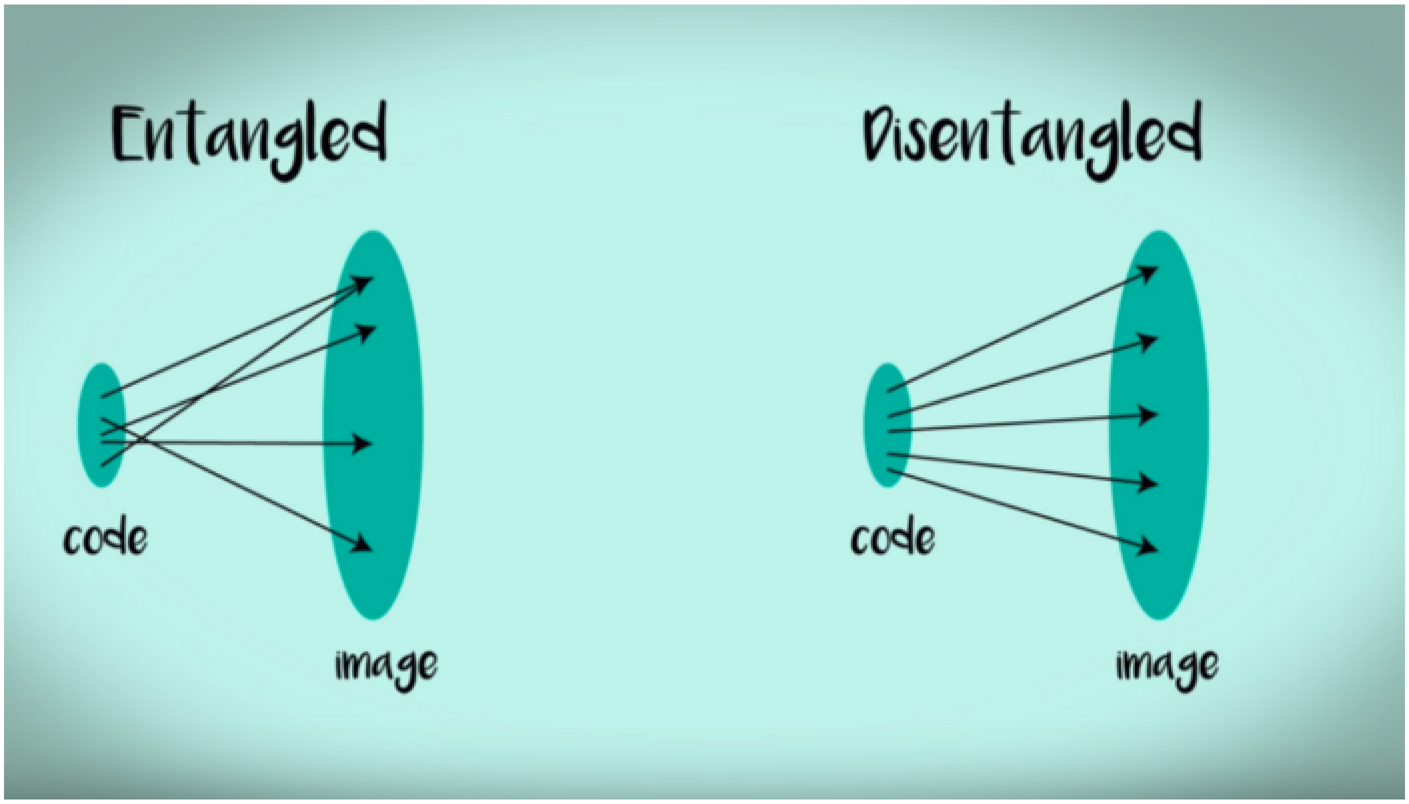
every generated sample has a corresponding class label:

$$X_{fake} = G(c, z)$$

Think of ACGAN as CGAN except D is not told about the label, though the additional network is added to give class probs at D.

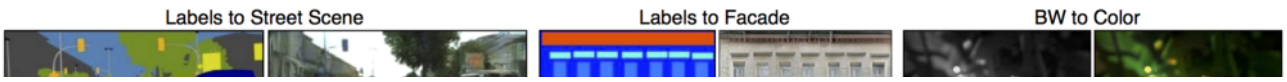
INFOGANS

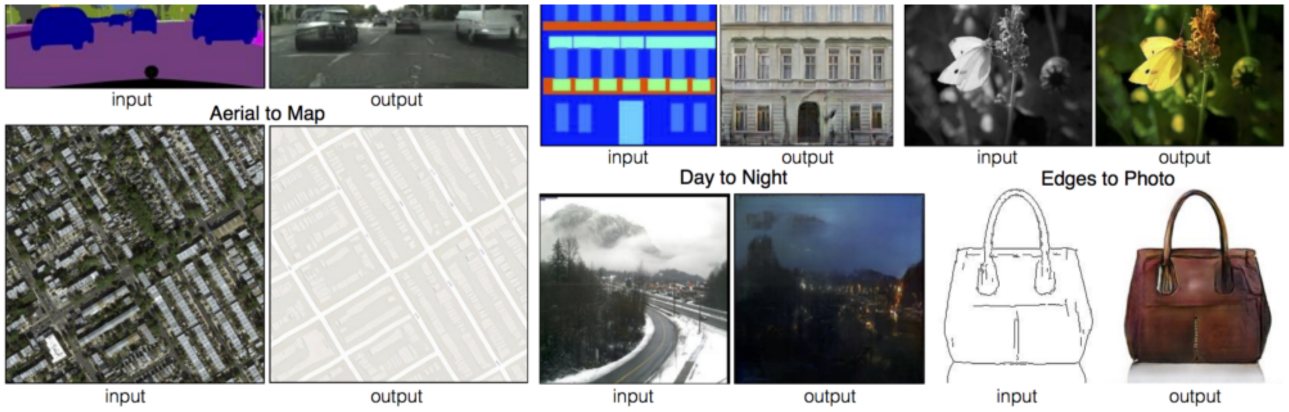
We seed GAN with random noise. We are not sure what is generating what. This is spooky. Can we change this? can we untangle this?

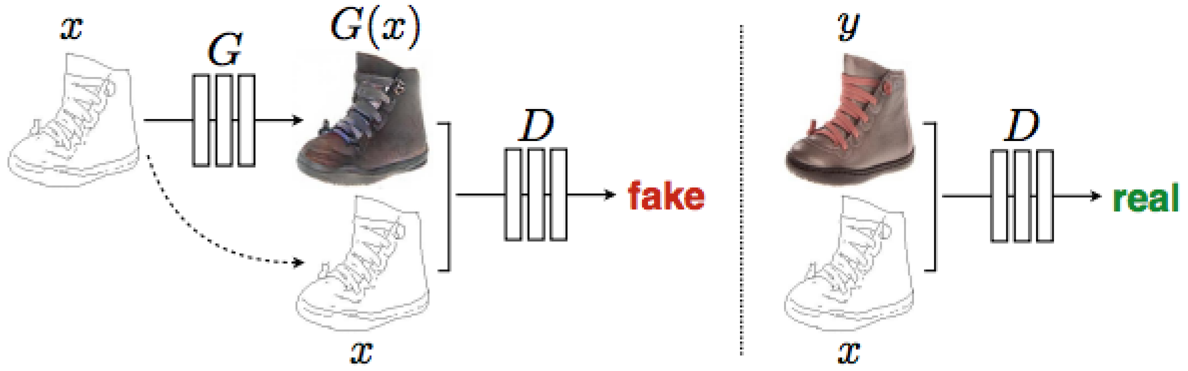


1. take a separate network called Q.
2. pick a random number (say x, like a label)
3. feed it to G, along with the noise
4. train D as you'd
5. feed the image generated by G and make Q network predict **x**

PIXELGANS

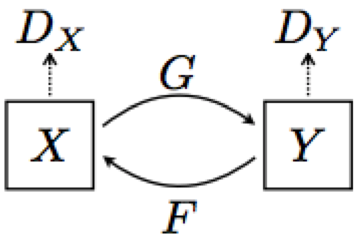


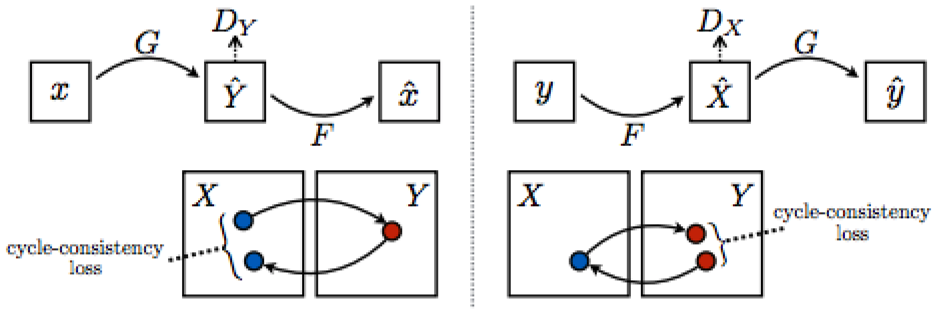




Basically feed both the image to both the networks.

CYCLEGANS





1. Take the 2 images x and y (1 from Domain X and 1 from Domain Y)
2. Run the two generators (x2y and y2x) → generate 2 fake images(y’,x’)
3. Run the two discriminators (DX and DY) DX takes x and x’ , DY takes y and y’
4. Calculate the discriminator's losses from the above equations
5. Run again the two generators, x2y (x’ as the input) and y2x (y’ is the input) generates two-cycle images (y_cycle, and x_cycle)
6. Calculate the cycle l1 loss from (x,x_cycle and y,y_cycle)
7. Finally, calculate the generator loss

7. Finally, calculate the generator loss

GAN PROBLEMS

Many GAN models suffer the following major problems:

- **Non-convergence**: the model parameters oscillate, destabilize and never converge,
- **Mode collapse**: the generator collapses which produces limited varieties of samples,
- **Diminished gradient**: the discriminator gets too successful that the generator gradient vanishes and learns nothing,
- Unbalance between the generator and discriminator causing overfitting, &
- Highly sensitive to the hyperparameter selections.

MODE COLLAPSE

Real-life data distributions are multimodal. For example, in MNIST, there are 10 major **modes** from digit ‘0’ to digit ‘9’. The samples below are generated by two different GANs. The top row produces all 10

modes while the second row creates a single mode only (the digit “6”). This problem is called **mode collapse** when only a few modes of data are generated.

ASSIGNMENT

1. Train any GAN to create 100x100 images of Indian Cars. [(https://github.com/simontomaskarlsson/CycleGAN-Keras)](https://github.com/simontomaskarlsson/CycleGAN-Keras)
2. KISS!
3. Move to Lambda and on your site.

VIDEO