**Generative Adversarial Networks**

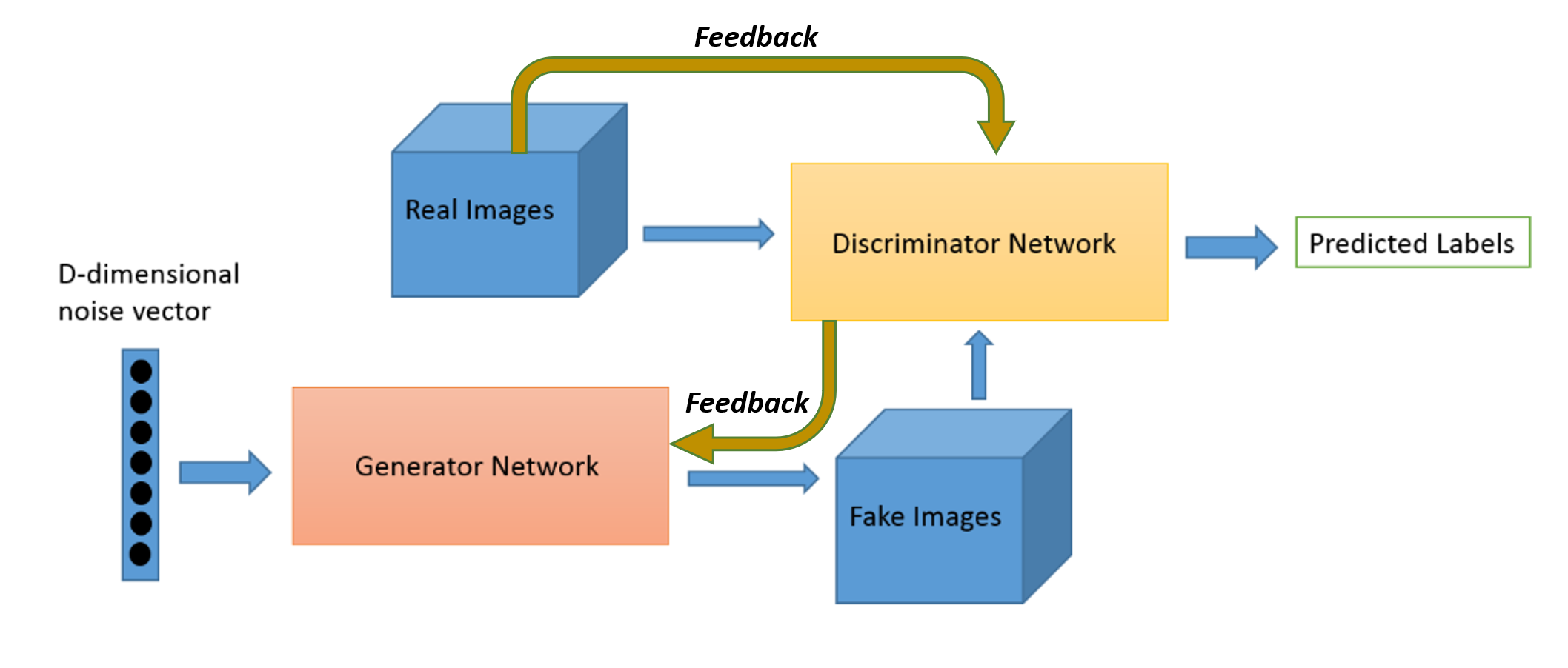
What are GANs? Generative Adversarial Networks consist of two artificial neural networks (sub-models) that compete in zero-sum game: a generator model and discriminator model. GANs are applicable broadly and used to, among other things, to create photo-realistic images for the visualization, to model motion patterns, to create 3D models of objects from 2D images and to process astronomical images. GANs are also used to naturally design user interaction with chatbots. GANs are also used in particle physics to accelerate time-consuming detector simulations.

To understand the difference and the interplay between the two sub-models, we first consider discriminative models.

Discriminator models usually refers to modelling a classification problem, with the purpose to find a discriminant function that maps a given input onto a specific class. A classic example is spam detection: what is the probability that a given e-mail is spam (y) considering all words in that e-mail (x)  p(y|x). This constitutes a supervised learning approach, that fundamentally aims to find the boundary between classes.

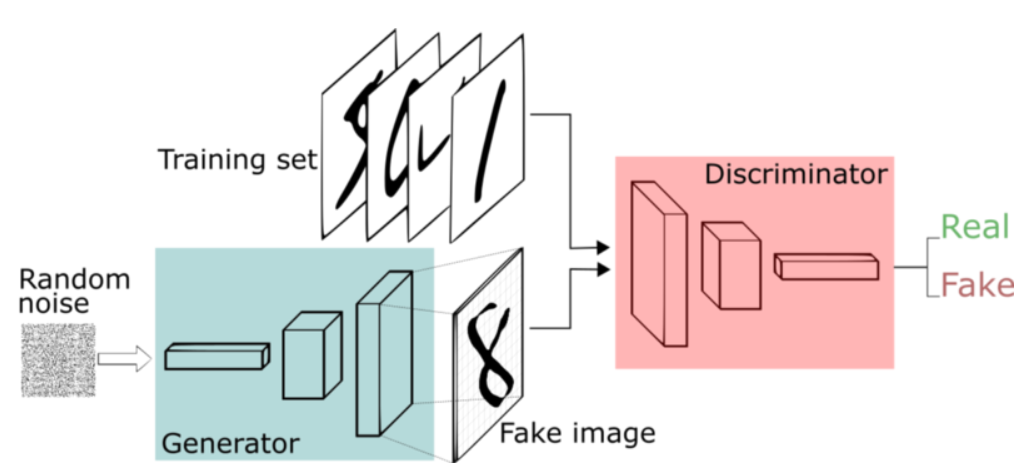
Generative modelling on the other hand falls into the category of unsupervised learning within the machine learning domain, and aims to discover patterns within a given data set to then generate output that mimics the underlying data. Considering the spam detection example again, a generative approach would be as follows: (1) assume a given e-mail in span; (2) what is the probability of seeing these words (relevant for spam detection) in a particular e-mail. Expressed in statistical terms, a generative approach models the joint probability of an observable variable and the target variable.

Generally, a GAN architecture combines these two approaches where the generative model first creates (generates) new data from a vector of latent variables to the desired result space. The generators’ aim is to learn to generate results based on a given data distribution to ultimately generate output that is indistinguishable from the ground truth. The discriminator, on the other hand is trained to distinguish the results of the generator from the (fake) data from the real data and labels the generators’ output accordingly. In this constellation, the discriminator provides feedback to the generator while simultaneously receiving feedback from the ground truth, the underlying data.



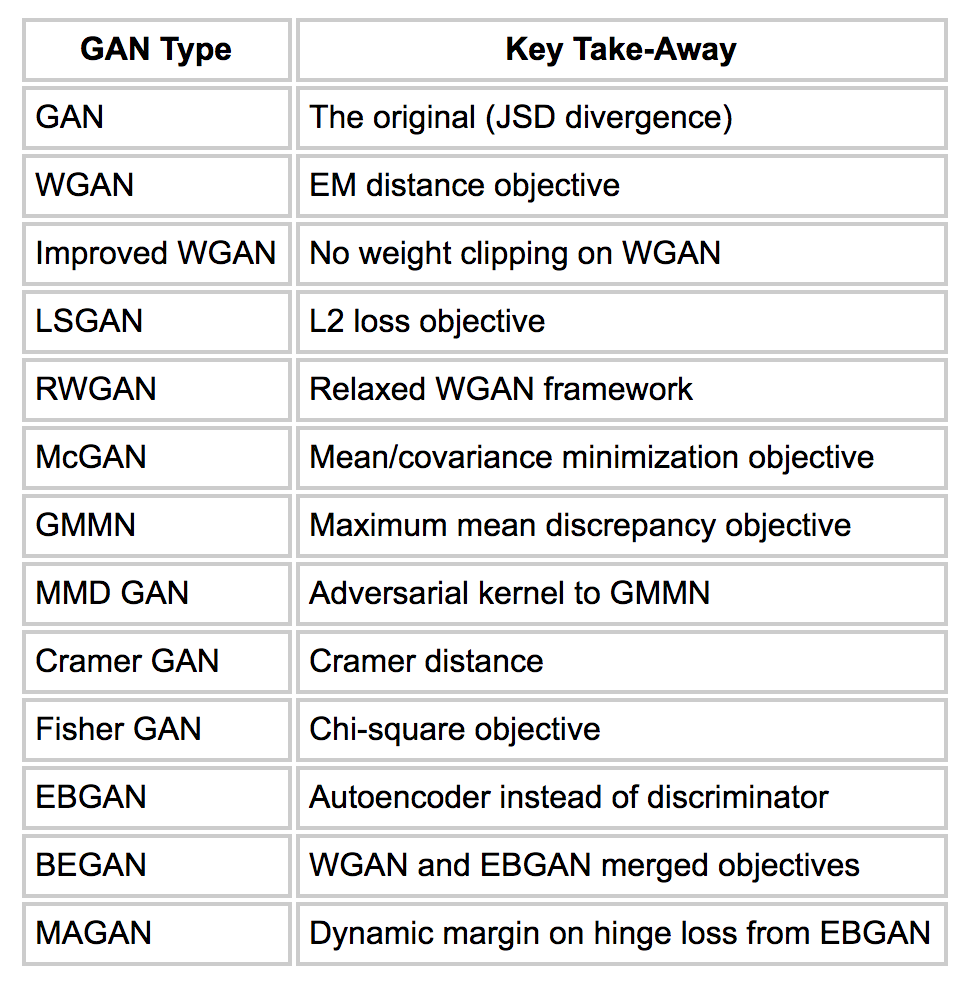
The two models are organized such that they compete in a zero-sum game (a concept rooted in game theory, where gains and losses cancel each other out, resulting in zero), hence the term “adversarial”. For instance, the discriminator can be a convolutional network for binary classification, say images. The generator, in a sense, can be seen as an inverse convolutional network that takes random data to produce images. Both models aim to optimize their opposing loss function. The result is a natural (Nash) equilibrium, where the generator produces output that is classified as real 50% of times. Through this combination of models, a unsupervised learning approach is transformed to a supervised approach. The following analogy describes this area of tension.

*„We can think of the generator as being like a counterfeiter, trying to make fake money, and the discriminator as being like police, trying to allow legitimate money and catch counterfeit money. To succeed in this game, the counterfeiter must learn to make money that is indistinguishable from genuine money, and the generator network must learn to create samples that are drawn from the same distribution as the training data.“*



**GAN Variations**

There exists a myriad GAN variations and evolutions as can be seen in the following table:



However, a good starting point for image-synthesis-based is Deep Convolutional GANs (DCGAN), based on Radfort et al’s groundbreaking work, that condenses to five best practice points guideline points when designing an DCGAN:

Architecture guidelines for stable Deep Convolutional GANs

1. The authors suggest to replace “…deterministic spatial pooling functions (such as maxpooling) with strided convolutions…“ in order to allow the convolutional network its own spatial down sampling.
2. Further it is recommended to use flattened layers that are directly connect to the output layer (instead of fully connected layers) in the discriminator model, as this yield more model stability and leads to faster convergence. The first layer, then, could be seen as fully connected to the output layer
3. Normalizing each input unit to have zero mean and variance (batch normalization) further stabilizes the learning process, since this supports gradient flow and initialization. Note, however, batch normalization should not be applied to the generator’s output layers and the discriminator’s input layers, due to arising problems of model fluctuations and instability.
4. Furhter, Radfort et al found that using ReLU activation results in faster learning rates in the domain of image classification, when used in the generator model (all layers except the output layer where Tanh function should be used).
5. A leaky rectified activation should be used in the gernerator.

In our analysis we use binary cross entropy as loss fuction, since GAN can be seen as a game of two players A and B that compete towards the same objective. Thus, both A and B need to be optimized in order to reach equilibrium.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Sources:**

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

NIPS 2016 Tutorial: Generative Adversarial Networks

https://en.wikipedia.org/wiki/Generative\_model

https://www.freecodecamp.org/news/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394/

https://pathmind.com/wiki/generative-adversarial-network-gan

https://towardsdatascience.com/gan-objective-functions-gans-and-their-variations-ad77340bce3c