# 1 Theoretical Background:

## 1.1 Probabilities and Statistics:

Generative Models like GANs are probabilistic by definition and they operate in stochastic fashion rather than a deterministic one, if the generative models learns a fixed calculation, such as merely taking the average value of each pixel in the dataset, it is not regraded as a generative one because the model produces the same output every time. Therefore, Statistics and information theory [1] are important to the understanding of how these models work and consequently enhancing their results.

In this section we will define the terms and ideas we incorporated from the field of probabilities and their mathematical formulas [2].

- The ***Sample Space*** is the complete set of all values an observation *x* can take.
- A ***Probability Density Function*** (PDF), $\mathrm{P}(\mathrm{x})$, is a function that maps a point $\mathrm{x}$ in the sample space to a number between 0 and 1.
- A ***Parametric Model***, $\mathrm{P}_\theta(\mathrm{x})$, is a family of density functions that can be described using a finite number of parameters, θ.
- The ***Likelihood*** $\mathscr{L}(\theta \mid \mathrm{x})$ of a parameter set θ is a function that measures the plausibility of θ, given some observed point x. It is defined as follows:

$$\mathscr{L}(\theta \mid \mathrm{x}) = \mathrm{P}_\theta(\mathrm{x}) \qquad \text{(eq.1)}$$

- ***Maximum Likelihood Estimation*** is the technique that allows us to estimate $\theta\grave{}$—the set of parameters $\theta$ of a density function, $\mathrm{P}_\theta(\mathrm{x})$, that are most likely to explain some observed data $\mathrm{X}$. More formally:

$$\theta\grave{} = \operatorname{argmax}_\theta \mathscr{L}(\theta \mid \mathrm{X}) \qquad \text{(eq.2)}$$

- ***Kullback-Leibler Divergence*** between two probability distributions $\mathrm{P}(\mathrm{x})$ and $\mathrm{Q}(\mathrm{x})$ that are defined over the same random variable is defined by eq.3 where N is number of samples in the sample space for a discrete random variable.

$$\mathrm{KL}(\mathrm{P}\|\mathrm{Q}) = \sum_i^N P(x_i) log \frac{P(x_i)}{Q(x_i)} \qquad \text{(eq.3)}$$

- ***Bayesian Inference*** ,Bayes' rule:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} \qquad \text{(eq.4)}$$

$\mathrm{P}(\mathrm{x})$: prior probability.
$\mathrm{P}(\mathrm{x} \mid \mathrm{y})$: posterior probability.
$\mathrm{P}(\mathrm{y} \mid \mathrm{x})$: likelihood.
$\mathrm{P}(\mathrm{y})$: evidence.

## 1.2 Notations of deep Learning:

### 1.2.1 Neural Networks:

A neural network [3] is a series of linear and nonlinear computations that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates.

A "neuron" in a neural network is a mathematical function that collects and classifies information according to a specific architecture.

A neural network contains layers of interconnected nodes. Each node is a perceptron that feeds its signal into an activation function.

In a multi-layered perceptron (MLP), perceptrons are arranged in interconnected layers. The input layer collects input patterns. The output layer has classifications or output signals to which input patterns may map.

Hidden layers fine-tune the input weightings to minimize the network's error. Hidden layers extrapolate salient features in the input data that have predictive power regarding the outputs. This describes feature extraction, which accomplishes a utility similar to statistical techniques such as principal component analysis.

### 1.2.2 Loss function:

A loss function [4] is a method for evaluating how well your algorithm models your dataset. Loss functions map a set of parameter values for the network onto a scalar value that indicates how well those parameter accomplish the task the network is intended to do.

Some of the most popular loss function in deep learning are Mean Squared Error, Log Likelihood and Cross Entropy loss.

### 1.2.3 Backpropagation:

Back-propagation is the essence of neural net training. It was first introduced in 1960s [5] and almost 30 years later (1989) popularized by Rumelhart, Hinton and Williams [6].

Backpropagation Tune the parameters $\theta$ of the model to minimize the value of the loss function. It adjust the parameters by small steps in towards the minimum of the loss function $\mathrm{L}$. The direction of each step for each parameter is found by calculating the partial derivative $\frac{\partial L}{\partial \theta}$ which can be computed using the multivariate chain rule.

The parameters can be updated as:

$$\theta = \theta - \alpha * \frac{\partial L}{\partial \theta} \qquad \text{(eq.5)}$$

Where $\alpha$ is the learning rate or the step size. The models usually doesn't work with one sample, but rather a batch of samples for which the loss is computed with respect to.
The update is performed, using the average derivative of that parameter where the average is taken over all the examples in the minibatch. Thus, bigger mini-batches help reduce the variance of the updates but introduce additional computational cost at the same time.

### 1.2.4 Activation Functions:

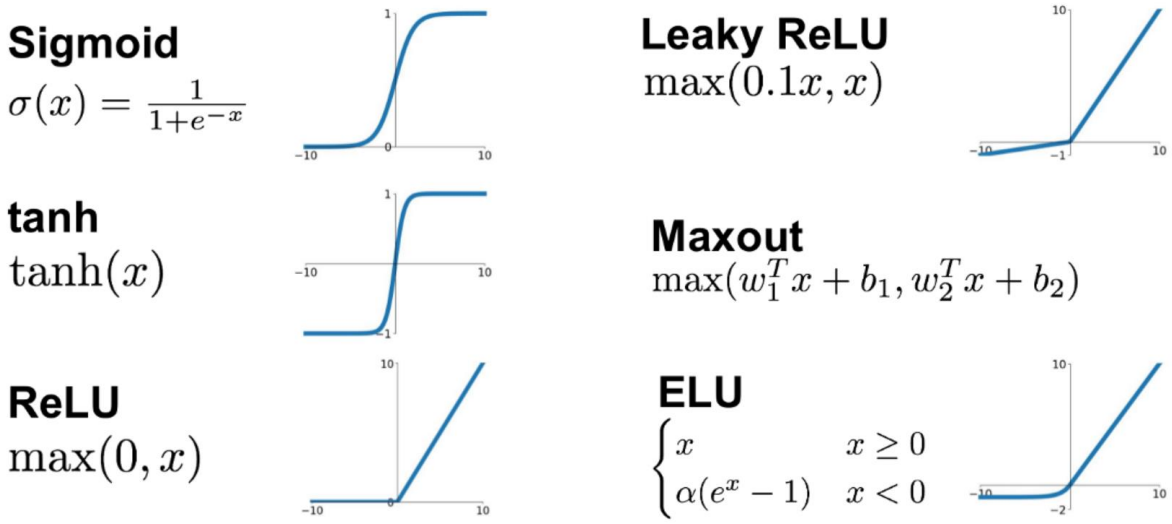


**Figure1:** activations functions and their garphs (Introduction to Different Activation Functions for Deep Learning [7])

The activation function provide the model more capacity to learn more complex functions. The activation function maps its input -each input unit multiplied by its weight- nonlinearly to an output (figure1).

This nonlinearity is what provide the capacity of neural network to learn different data distributions. Without the activation functions well have a bunch of stacked together linear layers which have the same capacity as a single linear layer.

### 1.2.5 Convolution Layers:

The name convolutional neural network [8] indicates that the network employs a mathematical operation called *convolution*. Convolution is a specialized kind of linear operation [9]. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

Convolutional Layers operate on 3D array of values called tensors and produce as output another 3D array of possibly different dimensions. Images are one such tensor with dimensions $\mathrm{width} \times \mathrm{height} \times 3$ in the case of RGB images. Convolutional layers are composed of filters. A filter has a reduced spatial resolution such as $3\mathrm{x}3$, each filter is convoluted with regions of the input tensor by sliding it across the width and height of the tensor.

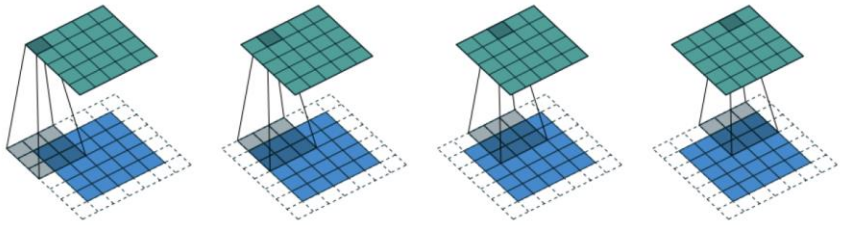

Figure 2: A convolutional filter with padding, stride one and filter size of 3x3 (Dumoulin & Visin, 2016 [10])

The stride is a parameter which determines by how many units the filter is moved in each direction during sliding. Another one is the amount of zero padding which refers to padding the borders of the input tensor with zeros (figure 2).

Our work uses a special type of convolution know as transpose convolution. A transposed convolutional layer carries out a regular convolution but reverts its spatial transformation. This layer is used in generative models for the task of upsampling feature maps.

## 1.3 Generative models:

The problem of text to image synthesis falls into the core of the domain of problems that generative models attempts to solve. So far, generative adversarial networks provide the best results in the task of text to image synthesis, but before we go into the details of how GANs work and the reason for their superior results we need to understand how other generative models work and the different in concepts between the two.

First we need to clarify the difference between the generative and discriminative models. Given a set of data instances X and a set of labels Y:

- **Generative** models capture the joint probability p(X, Y), or just p(X) if there are no labels.
- **Discriminative** models capture the conditional probability p(Y | X).

Generative models task is more difficult because they have to model more. A generative model for images might capture correlations like "things that look like boats are probably going to appear near things that look like water" and "eyes are unlikely to appear on foreheads." These are very complicated distributions. In contrast, a discriminative model might learn the difference between "dog" and "not dog" by just looking at few samples. It could ignore many of the correlations that the generative model must get right.

To achieve this task, the model needs to learn the probability distribution of the training data. *Variational Autoencoder* (VAE) [11] is one of the most popular approach to learn the complicated data distribution such as images using neural networks in an unsupervised fashion. It is a probabilistic graphical model rooted in Bayesian inference. The idea is to learn a low-dimensional latent representation of the training data called *latent variables* (variables which are not directly observed but are rather *inferred* through a mathematical model) which we assume to have generated our actual training data. The probability distribution of latent variables $\boldsymbol{z}$ is denoted by $\boldsymbol{P(z)}$ and a Gaussian distribution is selected as a prior to learn this distribution. VAE assumes that a low dimensional vector z has generated our sampled data x and try to maximize the log likelihood of the training data X with respect to the parameters of the models.

The power of VAE is that it learns both the generative model and an inference model even though GANs yield better results as aforementioned. In VAE, we optimize the lower variational bound whereas in GAN, there is no such assumption. In fact, GANs don't deal with any explicit probability density estimation. The failure of VAE in generating sharp images implies that the model is not able to learn the true posterior distribution and we can now see in details how GANs mitigated this problem.

## 1.4 Generative Adversarial Networks:

Generative Adversarial Networks -GANS- as proposed by Ian Goodfellow [12] are a deep learning framework that's represent the underlying probability distributions of different kinds of data. This proposed model overcomes the shortcoming of its predecessor generative models; yielding higher quality samples that's further mimic the true distribution of the data without the need to learn an explicit density which -for complex distribution- might not even exist.

GANs achieve this by pitying a generative model –called the generator ***G***- against an adversary discriminative model -called the discriminator ***D***- corresponding to a minimax two-player game [13]. The adversarial network can be thought of as the opposition of a counterfeiter and a cop, where the counterfeiter is learning to pass false notes, and the cop is learning to detect them. Both are dynamic and each side comes to learn the other's methods in a constant escalation.

The adversarial model framework suggested by Ian describe a case were both networks are multilayer perceptrons i.e. a simple neural network. However different architectures can be used for the two networks by applying the same techniques.

Before going into details of how GANs operate we will introduce some necessary notations. Starting with a Dataset $\boldsymbol{X} = \{ x_1, x_2, .., x_n\}$ composed of **n** samples and each sample $x_i$ is a vector and in our particular case $x_i$ is an image encoded as vector of pixel values. We assume this dataset is generated from an unknown underlying true distribution $\mathbf{P_{data}}$.

The Generator take as input a noise vector **Z** that's act as prior with distribution $\mathbf{P_z}$ which is sampled from a Gaussian distribution $\boldsymbol{P_z} = N\ (\mu\ ;\ \sigma)$. The then learns to map this input to a data space that represent a distribution $\mathbf{P_g}$ which estimate $\mathbf{P_{data}}$ of the data we are trying to generate**.** This mapping $\mathbf{G_{\theta g}: Z \rightarrow X}$ is defined as $\mathbf{G}(\mathbf{Z}; \boldsymbol{\theta_g})$ where **G** is a differentiable multilayer perceptron parametrized by $\boldsymbol{\theta_g}$. The second multilayer perceptron $\mathbf{D}(\mathbf{X}; \boldsymbol{\theta_d})$ outputs a single scalar and represents the probability that **X** came from the data $\mathbf{P}_{\text{data}}$ rather than $\mathbf{P}_{\text{g}}$.

The value or loss function in $\text{eq}.6$ was proposed by Ian to train the network. **D** is trained to maximize the probability of assigning the correct label to both training examples and samples from **G** while simultaneously **G** is trained to minimize the probability of assigning the correct label to fake samples from.

$$min_G max_D V(D,G) = E_{x\sim P_{data(x)}}[\log D(x)] + E_{z\sim P_z(z)}[\log(1 - D\big(G(z)\big))] \qquad \text{(eq.6)}$$

This framework differs from the previously discussed maximum likelihood models is that samples generated by the generator have an implicit distribution $\mathbf{P_g}$ determined by the values of $\boldsymbol{\theta_g}$. $\mathbf{P_g}$ cannot be explicitly evaluated. The loss form the value function when the discriminator correctly identify generated samples as fake drive $\mathbf{P_g}$ close to $\mathbf{P_{data}}$ (figure 3)

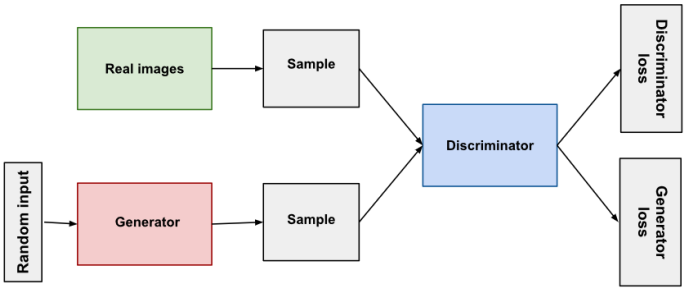


**Figure 3:** GANs architecture and loss produced by discriminator as described by the value function(Background: What is a Generative Model?[14])

A unique solution exists, with **G** recovering the training data distribution ($\mathbf{P_g = P}_{data}$) and **D** equal to 0.5 everywhere (figure 4). This case is known as *nash equilibrium* [15]. The equality $\mathbf{P_g = P_{data}}$ that happens at convergence is another difference between GANs and maximum likelihood generative models which doesn't guarantee it because it contains a biased estimator.

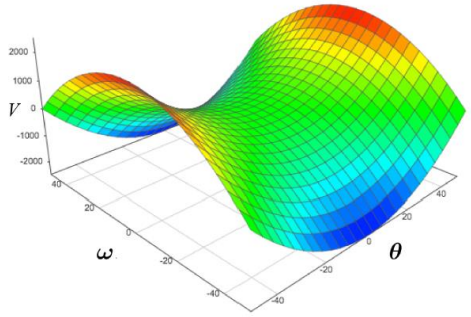


**Figure 4:** The Nash Equilibrium corresponds to a saddle point in space where the value function **V** is at a minimum with respect to $\boldsymbol{\theta_g}$ and at a maximum with respect to $\boldsymbol{\theta_d}$. Neither of the networks has any interest to change their parameters ([16]).

## 1.5 Spatial transformer networks (STN):

Convolutional Neural Networks are designed to be spatially invariant, that is, even when an image is rotated, sized differently or viewed in different illumination an object will be recognized as the same object, all in a computationally and parameter efficient manner.

Local max-pooling [17] layers in CNNs has helped to satisfy this property but due to the typically small spatial support for max-pooling (e.g. 2 × 2 pixels) the intermediate feature maps –the image in our case- are not actually invariant to large transformations of the input data.

The Spatial Transformer [18] can be included into a standard neural network architecture to perform spatial transformations on the entire feature map and can include scaling, cropping, rotations, as well as non-rigid deformations. This allows networks to be spatially invariant and also able to transform specific regions to an expected pose to simplify recognition and/or generation.

### 1.5.1 Localization Network:

The localization network takes the input image or feature map **U** of height (**H**) x width (**W**) x channels (**C**) and outputs the parameters of the transformation $\mathbf{T_\theta}$ which can be learnt as affine transform or to be more constrained to specific transformation (eq. 7).

$$A_\theta = \begin{bmatrix} s & 0 & t_x \\ 0 & s & t_y \end{bmatrix} \quad \text{(eq.7)}$$

### 1.5.2 Parameterized Sampling Grid:

The warping of the input feature map to output pixel – *A pixel is an element of a generic feature map, not necessarily an image* - is computed by applying a sampling kernel centered at a particular location in the input feature map. The output pixels are defined to lie on a regular grid **G** of pixels $\mathbf{G_i} = (\mathbf{x^t_i}, \mathbf{y^t_i})$ forming an output feature map **V** of **H'**x**W'**x**C**, where **H'** and **W'** are the height and width of the grid, and **C** is the number of channels, which is the same in the input and output.

### 1.5.3 Differentiable Image Sampling:

In theory, any sampling kernel can be used to produce the sampled output feature map **V** from a transformation $\mathbf{T_\theta}$, along with the input feature map **U** as long as (sub-)gradients can be defined with respect to $\mathbf{x^t_i}$ and $\mathbf{y^t_i}$.

## 1.6 Conditioning Augmentation:

Latent space for the text embedding for fine grained classification [19] is usually high dimensional (> 100 dimensions). This causes discontinuity in the latent data manifold [20], especially in the case of limited amount of data. This discontinuity disrupts the learning of the generator, conditioning augmentation helps mitigate this problem by sampling the latent variable $\dot{C}$ from a Gaussian distribution $N(\mu(\varphi), \sum(\varphi))$ where the mean and covariance matrix are conditioned on the text embedding thus encouraging robustness to small perturbations along the conditioning manifold. To further enforce the smoothness over the conditioning manifold and avoid overfitting the objective is regularized by the Kullback-Leibler divergence (KL divergence) [21] between the standard Gaussian distribution and the conditioning Gaussian distribution (eq. 8):

$$D_{KL}\left(N\left(\mu(\varphi_t), \sum(\varphi_t)\right) \,||\, N\,(0\,, I\,)\right) \quad \text{(eq.8)}$$

# 2 Refrencers:


[1] K. P. Burnham and D. R. Anderson, "Information and Likliehood Theory," in *Model Selection and Multimodel Inference - A Practical Information-Theoretic Approach*, 2nd ed., New York: Springer Science, 2002, pp. 49–96.

[2] K. Ahirwar, *Generative Adversarial Networks Projects*. 2019.

[3] S. S. Haykin, *Neural networks : a comprehensive foundation*, 2nd ed. Prentice Hall, 1999.

[4] C. P. Robert, *The Bayesian Choice*, 2nd ed. New York: Springer New York, 2007.

[5] "Multiple Linear Regression – MLR Definition." [Online]. Available: https://www.investopedia.com/terms/m/mlr.asp. [Accessed: 16-Feb-2020].

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, 1986, doi: 10.1038/323533a0.

[7] Introduction to Different Activation Functions for Deep Learning. (n.d.). Retrieved February 17, 2020, from https://medium.com/@shrutijadon10104776/survey-on-activation-functions-for-deep-learning-9689331ba092

[8] L. Cun *et al.*, "Handwritten Digit Recognition with a Back-Propagation Network," 1990.

[9] Rader, C. M. (1972). Discrete Convolutions via Mersenne Transforms. IEEE Transactions on Computers, C–21(12), 1269–1273. https://doi.org/10.1109/T-C.1972.223497

[10] Dumoulin, V., & Visin, F. (2016). A guide to convolution arithmetic for deep learning. http://arxiv.org/abs/1603.07285

[11] Kingma, D. P., & Welling, M. (2014, December 20). Auto-encoding variational bayes. 2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings.

[12] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. Advances in Neural Information Processing Systems, 3(January), 2672–2680. https://doi.org/10.3156/jsoft.29.5_177_2

[13] Osborne, M. J., & Rubinstein, A. (1994). A course in game theory. MIT Press.

[14] Background: What is a Generative Model? (n.d.). Retrieved February 9, 2020, from https://developers.google.com/machine-learning/gan/generative

[15] Zamir, S., Maschler, M., Solan, E., Hellman, Z., & Borns, M. (n.d.). Game theory.

[16] Bodnar, C. (2018). Text to Image Synthesis Using Generative Adversarial Networks. https://doi.org/10.13140/RG.2.2.35817.39523

[17] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-December, 770–778. https://doi.org/10.1109/CVPR.2016.90

[18] Jaderberg, M., Simonyan, K., Zisserman, A., & Kavukcuoglu, K. (2015). Spatial transformer networks. Advances in Neural Information Processing Systems, 2015-Janua, 2017–2025.

[19] Reed, S., Akata, Z., Schiele, B., & Lee, H. (2016). Learning Deep Representations of Fine-grained Visual Descriptions. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-December, 49–58. http://arxiv.org/abs/1605.05395

[20] Borsboom, D., Mellenbergh, G. J., & Van Heerden, J. (2003). The Theoretical Status of Latent Variables. In Psychological Review (Vol. 110, Issue 2, pp. 203–219). https://doi.org/10.1037/0033-295X.110.2.203

[21] Kullback, S., & Leibler, R. A. (1951). On Information and Sufficiency. The Annals of Mathematical Statistics, 22(1), 79–86. https://doi.org/10.1214/aoms/1177729694