

CGANs

Normal neural networks face 2 major issues with tasks related to one-to-many mappings or tasks where the number of categorical outputs increase a lot.

One way to deal with this is by using adding in other related information (other modalities) to the model, like it could be word embeddings as they ensure that related words are closer together and hence the loss function is relevant, this has proved to increase classification performance.

To solve the one-to-many mapping we use, Conditional GANs where condition the variables. i.e., we set an extra variable that tells us about the training data. For e.g. In MNIST, it could be the label of the image and due to the randomness in z , we can get different outputs each time for the same conditional variable.

Architecture:

Generator:

As usual, we start with a normal distribution $P_z(z)$ and take sample a random noise z from this, then we use a neural network and side by side to z we take in the conditional variable y and train it along a different set of layers finally we merge the 2 layers and thereby concatenate z and the conditional variable, to create a generated array of pixels of the given conditional variable. These outputs are then turned into a probability distribution using the Gaussian-Parzen-Window $P_g(x|y)$ where $x=G(z|y)$. Note that each y will have its own probability distribution, also, z is completely random and independent from y .

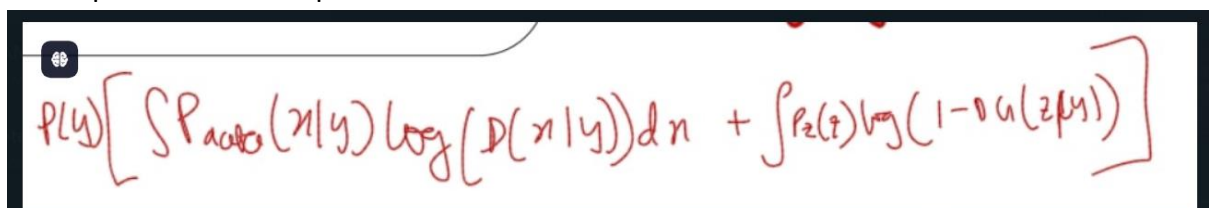
Discriminator:

The discriminator predicts 1 if it feels the image is not AI generated and 0 if it feels that the image is AI generated. If we are sampling from the real dataset, we take $P_{data}(x|y)$ as the probability of choosing x from the real data space. The discriminator is then fed in the value of x along with y like so: $D(x|y)$. Same for AI generated data, we pass in the following: $D(G(z|y))$.

This is the new min max game:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x|y)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z|y)))].$$

The expansion of the expected value is as follows:



The image shows a handwritten equation in red ink on a white background, enclosed in a black rectangular frame. The equation is:

$$P(y) \left[\int P_{data}(x|y) \log(P(x|y)) dx + \int P_z(z) \log(1 - D(G(z|y))) \right]$$

It is not $P_z(z|y)$ because z is not based on y , we concatenate the y factor during the neural network training and not while randomly choosing z .

Unimodal- generates the same output all the time for a conditioned CGAN. Ie, in MNIST if we ask it to predict an 8 it can only produce one unique image of the 8.

Multimodal-It is designed such that each time the output is different and different outputs can be generated for the same input, this resolves the one-to-many mapping problem we initially had. Use case: suppose you have an image and you want to create labels for it. There can be many labels assigned to the same image, by using a combination of vector embedding where you represent words as multidimensional arrays, and CGANs, we can do so by inputting the same image multiple times.

Note to self:

Maxout layers were frequently mentioned during training. This can be thought of as a trainable activation function with parameters. In a nutshell, it has many linear functions in its layers and we compute them for each of the outputs from the previous layer of perceptron and then we take the maximum from that. To be more precise, there are multiple units(number of units=number of outputs), and each unit has k number of pieces. Each piece has a bias and n number of weights where n is the number of input features.

