4. Generative Model Based Simulators

Generative AI is the domain which uses the concept of artificial intelligence to create new instances of data. Initially, Various machine learning methods such as deep learning are used to classify or prediction. However, over the years these models are built to generate data. These Model are trained to learn the density of distributions of the training data effectively various types of architecture are used for various data distributions. For example, images, Convolutional neural networks effectively learn the pattern and features in the images. For data embedded with temporal information, Recurrent neural network is often used to learn the relations between various time steps. At once a model is trained to learn the distribution of the training data, new data points can be generated by sampling from the learned distribution. These generated are usually not in the training corpus and a effectively trained generative models can generate data with features similar to training data. This method can be applied to many fields such as generating new images, music, text…

Based on the learning of density of data distribution, generative modelling can be classified in two types Implicit density explicit density

Explicit Density Models: These models explicitly learn and estimate the probability distribution of the input data. For example, Variational Autoencoders (VAEs) and traditional probabilistic models explicitly model the probability distribution of the data in a defined space. VAEs learn a probabilistic representation of the data, allowing them to generate new samples by sampling from this learned distribution.

Implicit Density Models: On the other hand, implicit density models do not explicitly define the probability distribution. Instead, they learn to generate new data points without directly modelling the probability distribution. Generative Adversarial Networks (GANs) are a prime example of implicit density models. GANs involve a generator network that learns to create samples without explicitly defining the underlying probability distribution. The generator aims to generate data that is indistinguishable from real data, but it doesn't directly model the probability distribution of the data.

The concept of Generative adversarial network was introduced in 2014 by goodfellow et.al has been a break through in the domain of Generative AI. GANs are initially used to generate images. Unlike VAE which generates blurry images, these GANs generate high quality images. However, these models suffer from various issues at the beginning such as Mode failure (inability to generate diverse data), instability in training… Later, various types of GANs had addressed this issue which made it a successful algorithm in the domain of Generative AI. Moreover, these models are generalized for various data making it versatile in this domain.

The ability of generating diverse and high-quality data has various application. In this proposed method these generated data are used to drive a simulator which trains autonomous vehicles. For effective training of a model which drives an autonomous vehicle, it has to trained on diverse scenarios. The existing simulator can offer a limited scenario for training which can be difficult for a model to generalize across diverse scenario. In the proposed method, the simulator is driven by a neural network model which is able to generate sensory data required by an autonomous vehicle. The driving model is trained on sensory data of diverse scenarios.

4.1 Neural networks

Neural networks are the mathematical functions which maps an input value(s) to an output value(s). These functions are made of multiple perceptron which are arranged in layers and connected to each other.

4.1.1 Perceptron

A Perceptron is a linear binary classifier given by the following equation

Z = W.X+ B

Where w is the vector of weights w1, w2, w3…wn. x is the vector of inputs x1, x2, x3…xn. and b is the bias. The weights in the W signifies how much importance every input in vector x influences in the prediction. Figure 15 illustrates a neuron in perceptron with single output and multiple inputs.

4.1.2 Multi-layer perceptron

Neural networks is constructed by connection multiple layers of neurons in layers. The non liner function/ activation functions such as sigmoid, ReLU activation functions are used between the layers which maps the input to some non linear functions. Figure 16 shows the commonly used non linear activation functions. The process of calculating the output from this composite function for given input is called Forward Propagation.

Figure 16 Not linear activation function a) Sigmoid b) ReLU c) tanh

Neural networks are universal approximator which can approximate almost any continuous functions. Multiple layers of neuron and activation function makes it versatile to approximate complex functions. The weights and Bias associated with the neurons are initialized randomly and learned during the training. Figure 17 shows the architecture of a 2 layered Network.

Mathematically, the network function is represented as a proballistic function and the output of the model Y is given by

Y = p(y|x)

4.1.3 Loss functions

Loss function or cost function are the functions which calculates the deviation or difference between the original data point and the predicted data point. The metrics are used to train the network such that a network has to minimize this metric. Various loss functions are used depending upon the data and use case some of the popular loss functions are Mean square error, Entropy loss, KL divergence…

4.1.4 optimizers

Optimizers are the methods which are used to train the neural networks. The Network is trained by learning its parameters (weights and biases), where the cost function of the network is minimum. This is achieved by calculating the gradients of the parameters with respect to the loss function and updating the parameters by the value of step size in a way which its gradients shift in the direction of global minima using the equation

P\_new = p\_old − α∇L(p\_old)

Where ∇L(p\_old) is the gradient of loss function with respect to p\_old. Usually, in a multi layered network, the gradients of parameters are calculated using chain rule. The gradient equation of a parameter in layer n has the terms with gradients of parameters associated with neurons after it. α is the learning rate which signifies the rate at which the parameter update should happen. This rate is reduced over the training in some optimizer like ADAM (Adaptive Moment Estimation) to avoid skipping the minima. This entire process of updating the parameters of a network is called Back propagation. Various versions of optimization technique such as RMSprop, Adagrad, Adadelta are used in Neural network training.

4.1.4 Training

The process of training is iterating the backpropagation over the loop until this loss becomes low and stabilizes. The training data is segmented into batches making computationally effective and training faster. Backpropogation of all the batches in the training data marks an epoch. Various metrics such as accuracy are used to monitor the training. Moreover a small set of data is subjected to validate the model which is not used for training.

4.1.5 Batch Normalization

Batch normalization is a technique in neural networks, improving training process by normalizing layer inputs within mini-batches. By calculating batch-wise mean and variance and normalizing inputs accordingly, it enables faster convergence, prevents vanishing or exploding gradients, and reduces the sensitivity to weight initialization.