

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**NAME: R.Siva Kumar**

**REG.NO: 121012012767**

**COURSE NAME: DEEP LEARNING**

**ASSIGNMENT - 1**

**INTRODUCTION :**

Neural networks are the fundamental building blocks of deep learning, a subfield of machine learning that focuses on training models to perform tasks without explicit programming. Modeled after the human brain's structure and functioning, neural networks excel at learning complex patterns and representations from data.

**TYPES OF ARTIFICIAL NEURAL NETWORK :**

1. Input Layer:

The first layer that receives the initial input data. Each node in this layer represents a feature or attribute of the input.

2. Hidden Layers:

These are intermediate layers between the input and output layers. They process the input data through weighted connections to produce an output. Neural networks can have multiple hidden layers.

3.Output Layer:

The final layer that produces the network's output. The number of nodes in this layer depends on the problem type (e.g., binary classification, multi-class classification, regression).

4. Dense/Fully Connected Layer:

In a dense layer, each node is connected to every node in the previous and subsequent layers. This is a common type of layer in a neural network.

5. Convolutional Layer (Convolutional Neural Network - CNN):

Used for processing grid-like data, such as images. Convolutional layers apply convolution operations to learn spatial hierarchies of patterns.

6. Pooling Layer (CNN):

Usually follows convolutional layers. It reduces the spatial dimensions of the input volume, reducing the amount of computation in the network.

7. Recurrent Layer (Recurrent Neural Network - RNN):

Designed for sequential data, it maintains a hidden state that captures information about previous inputs. This allows RNNs to process sequences.

8. LSTM (Long Short-Term Memory) Layer (RNN):

-A specialized type of recurrent layer designed to capture long-term dependencies in sequential data.

9. GRU (Gated Recurrent Unit) Layer (RNN):

Similar to LSTM but with a simpler architecture. It also aims to capture long-term dependencies in sequential data.

10.Dropout Layer:

Used for regularization. It randomly drops a proportion of the connections during training to prevent overfitting.

11. Batch Normalization Layer:

Normalizes the input of a layer, often accelerating the training process and improving convergence.

12. Activation Layer:

Introduces non-linearity to the network. Common activation functions include ReLU (Rectified Linear Unit), Sigmoid, and Tanh.



**3 . Optimizers**

1. **Stochastic Gradient Descent (SGD):**

Updates the parameters in the opposite direction of the gradient of the loss function with respect to the parameters.

Computationally efficient but may converge slowly and get stuck in local minima.

1. **Adaptive Moment Estimation (Adam):**

Maintains per-parameter learning rates that are adapted based on the average of past gradients and the past squared gradients.

Combines the advantages of both AdaGrad and RMSProp.

Widely used due to its good performance across various tasks.

1. **RMSProp (Root Mean Square Propagation):**

Adapts the learning rate for each parameter based on the average of recent magnitudes of the gradients for that parameter.

Helps to alleviate the diminishing learning rates problem encountered in AdaGrad.

1. **Adagrad (Adaptive Gradient Algorithm):**

Adapts the learning rate for each parameter based on the historical gradients for that parameter.

Scales down the learning rate for frequently occurring features.

1. **Adadelta:**

An extension of Adagrad that seeks to reduce its aggressive, monotonically decreasing learning rate.

Adapts the learning rate based on a moving window of the gradient updates.

1. **Nesterov Accelerated Gradient (NAG):**

A variant of SGD with momentum that first calculates the gradient using an intermediate parameter update and then adjusts the momentum update.

Helps to prevent the momentum update from overshooting the minimum.

1. **Momentum:**

Accumulates a moving average of the gradients and uses this information to update the parameters.

Helps accelerate SGD in the relevant direction and dampens oscillations.

1. **AdaMax:**

A variant of Adam based on the infinity norm.

Can be more stable than Adam for very large datasets and high-dimensional parameter spaces.

1. **Nadam:**

Nesterov Adam optimizer, combining Adam with Nesterov momentum.

1. **FTRL-Proximal (Follow-The-Regularized-Leader Proximal):**

An online optimization algorithm designed for large-scale linear models.

1. **AdaBound**

An optimizer that combines the benefits of AdaGrad, Adam, and other adaptive optimizers.

Converges faster than traditional optimizers while achieving better generalization.

1. **AMSGrad**

An improvement over Adam that addresses the problem of Adam's inability to effectively converge on certain non-convex surfaces.

Maintains a maximum of past squared gradients to prevent the learning rate from being too aggressive.