KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY
AN AUTONOMOUS INSTITUTION - ACCREDITED BY NAAC WITH 'A' GRADE
Narayanaguda, Hyderabad.

Deep Learning

28-10-2024

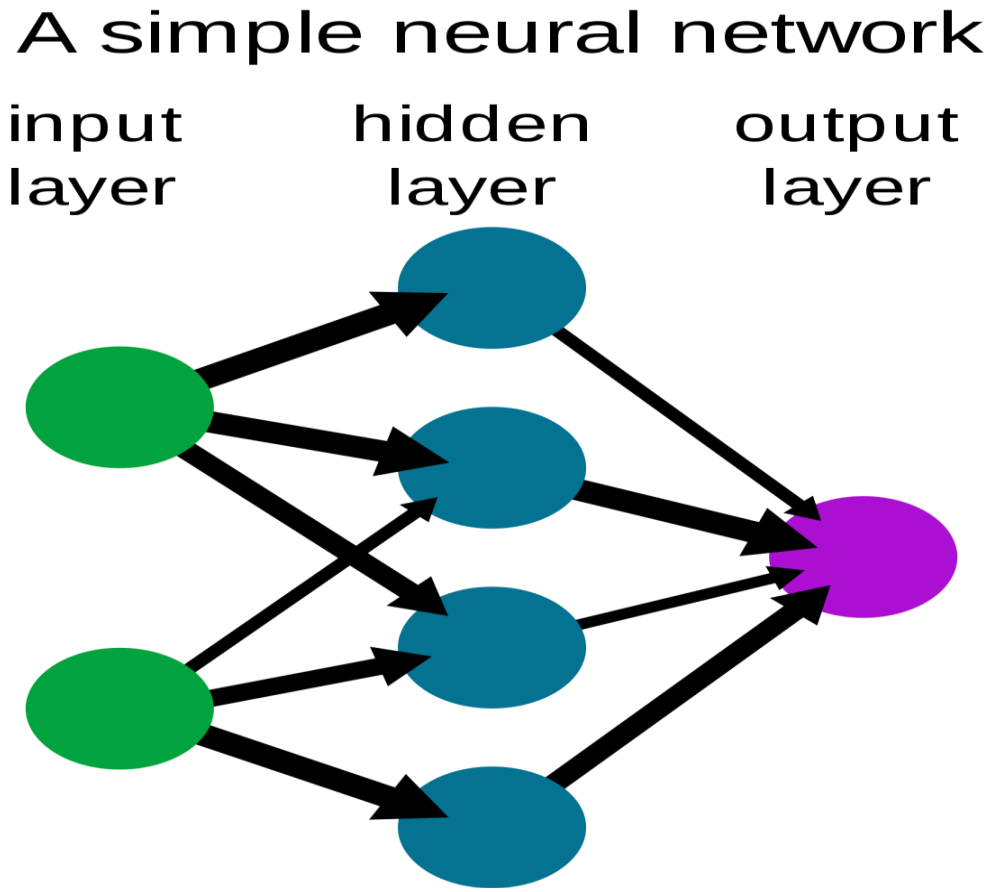
BY
ASHA

Building ANN from Scratch

Neural Networks, Activation Functions, Loss Functions, and Optimizers

What are Neural Networks?

- At the heart of AI systems, **neural networks** are computational models inspired by the structure and function of the human brain. They consist of interconnected nodes, called **neurons**, that are organized in layers. These networks are particularly powerful for tasks that require pattern recognition, such as image classification, language processing, and time series prediction.



A simple neural network consists of three components :

- Input layer
- Hidden layer
- Output layer

- **Input layer**: The layer where raw data (features) is fed into the network.
- **Hidden layers**: Intermediate layers where the network processes and transforms the input data through weights and biases.
- **Output layer**: The final layer, producing the model's prediction.

Why Neural Networks?

Traditional machine learning algorithms may struggle with complex, non-linear problems. Neural networks, however, excel at:

- Learning from data and improving their performance.
- Handling a variety of tasks, from regression to classification, across multiple domains (vision, speech, text, etc.).
- Detecting intricate patterns and relationships that traditional models can miss, particularly in unstructured data like images or audio.

Why Do We Use Neural Networks?

We use neural networks because of their ability to:

- **Learn complex, non-linear relationships**: Unlike traditional models like linear regression, neural networks can model highly complex relationships between input and output data.
- **Generalize well to unseen data**: With proper training, neural networks can learn to generalize from the training set to new, unseen data, improving predictive performance.
- **Versatility across domains**: Neural networks can be applied to a wide range of tasks including classification, regression, and generation of new content in domains like computer vision, natural language processing, and more.
- **Automatic feature extraction**: Deep neural networks, particularly convolutional neural networks (CNNs), can automatically extract important features from raw data without the need for manual feature engineering.

How Do We Build an Artificial Neural Network (ANN)?

Step 1: Initialize the Network Architecture

A basic ANN consists of three types of layers:

Input Layer: Receives the input features (e.g., pixel values of an image, words in a text, etc.).

Hidden Layers: Consist of neurons that apply a transformation to the input data using weights and biases.

Output Layer: Produces the final prediction, typically a classification or regression output.

Each connection between neurons has a **weight**, and each neuron has a **bias**.

Step 2: Apply an Activation Function

Each neuron takes a weighted sum of its inputs, applies a bias, and passes the result through an **activation function**. The role of activation functions is to introduce **non-linearity**, allowing the network to model complex data patterns.

Step 4: Define a Loss Function

The **loss function** quantifies how far off the network's predictions are from the actual values. Common loss functions include:

Mean Squared Error (MSE) for regression tasks.

Cross-Entropy Loss for classification tasks. The goal is to minimize this loss by adjusting the network's weights and biases.

Step 3: Forward Propagation – Generating Predictions

Once the architecture is set, the data is fed forward through the network:

Each neuron in the hidden layers calculates the weighted sum of its inputs, adds the bias, and applies the activation function.

The final layer outputs a prediction, often using activation functions like **Softmax** for multi-class classification or **Sigmoid** for binary classification.

Step 5: Backpropagation – Learning from Errors

After generating predictions, the network needs to learn from its mistakes. **Backpropagation** is the process of calculating the **gradient** of the loss function with respect to each weight in the network, using the chain rule.

This gradient indicates how each weight should be adjusted to reduce the overall loss.

Step 6: Use an Optimizer to Update Weights

The **optimizer** is responsible for adjusting the weights based on the gradients from backpropagation. Popular optimizers include:

Stochastic Gradient Descent (SGD): Updates weights using small, random batches of data.

Adam: Combines the advantages of SGD and other adaptive algorithms, allowing for faster and more efficient convergence.

Optimizers are key to ensuring that the model converges to an optimal solution efficiently.

Step 7: Train the Network

The training process consists of feeding the model with batches of data (called **epochs**), performing forward and backward passes, and updating weights to reduce the loss function. After several iterations, the model learns to make accurate predictions.