Introduction to Neural Networks and Machine Learning

What is a Neural Network?

Think of a neural network as a team of smart detectives working together to solve a mystery. Each detective has a specific role, and they all collaborate to figure out the solution.

Imagine you have a big puzzle to solve, like figuring out what type of animal is in a picture.

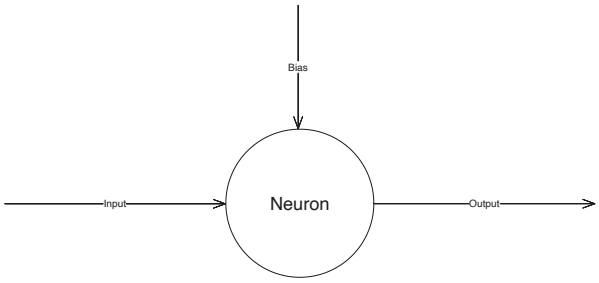
1. **Input Layer (The Lookouts):**
   - This is the first group of detectives. They look at the picture and gather all the important details like colors, shapes, and sizes. These details are like clues they need to solve the puzzle. In a neural network, this layer is where the data enters the system.
2. **Hidden Layers (The Analyzers):**
   - These detectives take the clues from the Lookouts and start analyzing them. They look for patterns and connections between the clues. For example, they might notice that the animal has four legs and fur, which could be important for identifying it. In a neural network, there can be one or more hidden layers that process the data. These layers are called "hidden" because we don't see them directly, but they do a lot of the hard work behind the scenes.
3. **Output Layer (The Decision Makers):**
   - Finally, the last group of detectives makes the decision based on the analysis. They put all the clues and patterns together to conclude what the animal is. They might say, "Based on the clues, it's a dog!" This is the output of the neural network – the final result.

Neurons and Layers

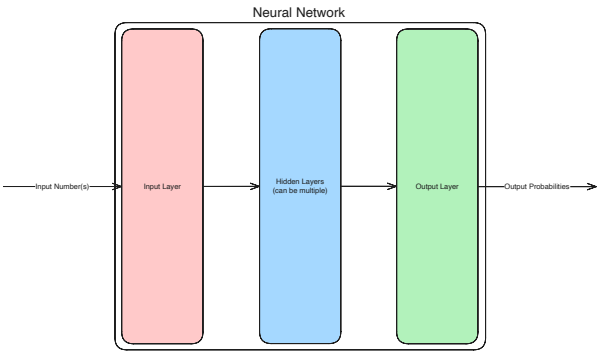
1. What is a Neuron?

A neuron is the basic unit of a neural network, much like a single piece of a puzzle. Each neuron takes in information, processes it, and passes it on to the next neuron. You can think of a neuron as a tiny decision-maker. Imagine each neuron as a light switch that can either be on or off, depending on the information it receives. When a neuron is “on,” it means it’s activated and ready to pass information to the next layer,

this activation-decision uses an Activation Function. The Output of the Neuron is the weighted sum of its inputs.



2. Layers: *How Do Neurons Work Together?*



1. **Input Layer:**
   - The input layer is where the network receives information. Each neuron in this layer represents a feature or piece of data. For example, if you're recognizing hand written numbers in black and white pictures, the input layer might receive the pixels brightness values of the image.
2. **Hidden Layers:**
   - Hidden layers are in the middle of the network and do most of the processing work. They transform the input data into something meaningful by finding patterns. Each neuron in a hidden layer takes input from the previous layer, processes it, and passes it to the next layer. There can be one or multiple hidden layers, depending on the complexity of the task. These layers are also called “Fully Connected Feed Forward Layers” or “Multi-Layer Perceptrons.”
3. **Output Layer:**
   - The output layer produces the final result of the network’s processing. Each neuron in this layer represents a possible outcome or category. For example, in an animal recognition network, the output layer might have neurons for "dog," "cat," "bird," etc., and it will activate the one that best matches the input data. The Output of the Neural Network are the probabilities of all the possible cases the Network knows. Another algorithm

then selects the category with the highest probability, indicating the network's confidence in its prediction.
- This might look like this, here consists the input of a dog:

By working together, these layers allow the neural network to learn from data, find patterns, and make decisions, much like how a team of detectives solves a mystery by piecing together clues.

Activation Functions

What is an Activation Function?

An activation function is a special rule that determines whether a neuron should be "activated" or not, just like deciding if a light should be turned on or off. Think of it as a gatekeeper that decides if the information passing through is important enough to move on to the next layer. Without activation functions, a neural network would simply be a linear model, unable to learn complex patterns.

Common Activation Functions

1. **Sigmoid**
   - The sigmoid function takes any input value and transforms it into a value between 0 and 1. It's like a soft switch that can be anywhere between completely off (0) and fully on (1), making it useful for binary classification tasks. Here is its mathematical equation:

$$Sigmoid(x) = \frac{1}{1 + e^{-x}}$$

1. **ReLU (Rectified Linear Unit)**
   - The ReLU function outputs the input directly if it is positive; otherwise, it outputs zero. This function is very popular because it helps the network learn quickly and handles the problem of vanishing gradients.

$$ReLU(x) = max(0, x)$$

1. **Tanh**
   - The tanh function is similar to the sigmoid function but outputs values between -1 and 1. This makes it useful for tasks where the output needs to capture negative and positive values, allowing the network to center its data around zero.

$$Tanh(x) = tanh(x) = \frac{e^z - e^{-x}}{e^z + e^{e-x}}$$

Why Are Activation Functions Important?

Activation functions are crucial because they allow neural networks to learn and model complex patterns. Without activation functions, the network would be limited to simple linear transformations, which means it wouldn't be able to capture the intricacies of the data.

- **Learning Complex Patterns:** Activation functions introduce non-linearity into the network, enabling it to learn and represent complex patterns and relationships in the data.
- **Gradient-Based Optimization:** They help in gradient-based optimization techniques (like backpropagation) to adjust the weights efficiently and converge to a solution that minimizes error.

In summary, activation functions are like the brain's decision-making process, allowing neural networks to understand and learn from data in a sophisticated way, ultimately making them capable of solving complex tasks.

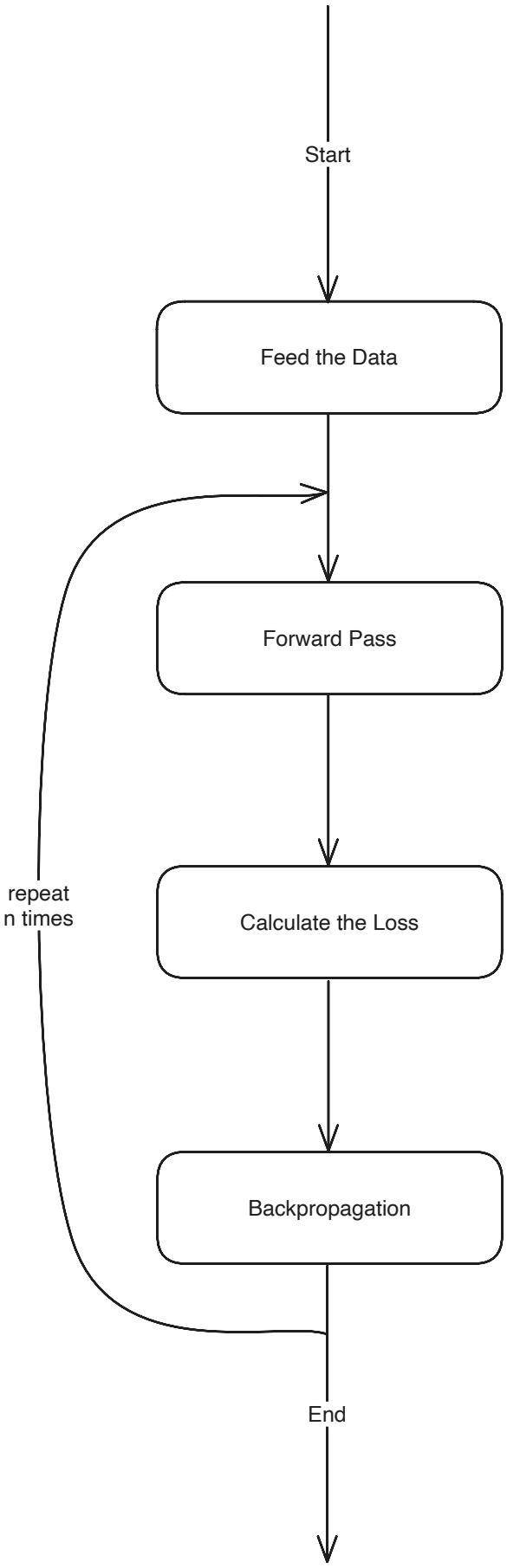
Training a Neural Network

3. What is Training?

Training a neural network involves feeding it data and adjusting its internal parameters (all the weights and biases in the Network) to minimize the error in its predictions. This process is similar to teaching a team of detectives to get better at solving puzzles by learning from their mistakes.

Steps in Training:

1. **Feed Data:** Provide the network with input data and the corresponding correct output (label).
2. **Forward Pass/Fowardpropagation:** Pass the data through the network to get predictions.
3. **Calculate Error/Loss:** Compare the predictions to the actual output to determine the error/loss.
4. **Backward Pass/Backpropagation:** Adjust the network's weights and/or biases to reduce the error / loss.
5. **Repeat:** Iterate the process with multiple data samples until the network's predictions are accurate enough.



Forward Propagation/Forward Pass

• Forward propagation is the process by which input data is passed through the layers of the neural network to produce an output. It involves computing the

activations of neurons layer by layer, starting from the input layer and moving through the hidden layers to the output layer.

Loss/Error

What is Loss/Error?

The error or loss is a measure of how well the network's predictions match the actual results. It quantifies the difference between the predicted output and the true output. A high loss indicates poor performance, while a low loss indicates better performance.

Why is Minimizing Error Important?

• The goal of training a neural network is to make the predictions as accurate as possible.
• Minimizing the loss is crucial because the primary objective of training a neural network is to achieve accurate predictions. By reducing the loss, the network becomes more reliable and effective in making correct predictions or decisions based on the input data. This is achieved through optimization techniques in the Backpropagation phase that adjusts the network's parameters (weights and biases) to improve its performance.

Backward Propagation

What is Backward Propagation?

• Backward Pass, or backpropagation, is a method used to minimize the error by adjusting the weights and biases in the network. It involves calculating the gradient of the loss function with respect to each weight and bias and then updating these same parameters in the direction that reduces the loss.

Steps of Backward Propagation with the calculated loss:

1. Adjust weights in the network to reduce error.
   • The loss value is then used to propagated backward through the network. Using the calculated loss, the gradients of the loss with respect to each weight are computed. These gradients indicate how much the weights need to be adjusted to minimize the error.
2. Repeat until the error is minimized.
   • This process is repeated for multiple iterations or epochs, adjusting the weights and biases each time until the error is sufficiently minimized.

Gradient Descent and Gradients

What is Gradient Descent?

Gradient descent is an optimization algorithm used to minimize the loss function by iteratively moving towards the minimum value of the function. It works by taking steps proportional to the negative of the gradient of the function at the current point.

What are Gradients?

Gradients are the partial derivatives of the loss function concerning the network's weights and biases. They indicate the direction and rate of change of the loss function with respect to each parameter.

The Gradients we need are the. Gradient of the loss with respect to the Neural Networks (NN) output, Gradient of the NN output with respect to the output of the Neuron.

We then combine the gradients to get the gradient of the loss with respect to the weights and biases.

And lastly we update the weights and bias with the said Gradients.

Why Do We Need Gradients?

Gradients are essential because they provide the information needed to update the network's parameters in a way that reduces the error. By using gradients in the gradient descent algorithm, we can efficiently find the optimal parameters that minimize the loss function, thereby improving the network's performance.

And we finished our first training step, now repead from the Forward Pass.