An Introduction to Neural networks

Yi Shen Lim and Aadvik Mohta

6 May 2025

Overview

Mathematical Prerequisites

The Dot Product

We say the dot product of two 3D column vectors is

$$\begin{pmatrix} a_1 \\ b_1 \\ c_1 \end{pmatrix} \cdot \begin{pmatrix} a_2 \\ b_2 \\ c_2 \end{pmatrix} = a_1 a_2 + b_1 b_2 + c_1 c_2$$

This result can be generalised for column vectors of any dimension.

Mathematical Prerequisites

Matrices

We can consider an array of numbers represented in the form of a matrix

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{pmatrix}$$

This matrix has m rows and n columns, and we call it an $m \times n$ matrix.

Mathematical Prerequisites

Matrix Multiplication

$$\mathbf{AB} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1r} \\ b_{21} & b_{22} & \dots & b_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nr} \end{pmatrix}$$

Note that the resultant matrix will be an $m \times r$ matrix, and that to find the ij-entry of this matrix we take the dot product of the i-th row of the first matrix with the j-th column of the second matrix.

Mathematical Prerequisites

Exercise 1

If $\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 2 & 1 & 4 \end{pmatrix}$ and $\mathbf{B} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$, then what is $\mathbf{AB}$?

Exercise 2

What is the condition for us to satisfy in order to multiply two matrices together?

Mathematical Prerequisites

Gradient

In secondary school, we learned that the gradient of a curve at a point is basically the gradient of tangent to the curve at that point. This is nice in the 2D-plane, but the definition of a gradient for functions of several variables is different, because such functions are not simply defined in 2D.

Mathematical Prerequisites

Partial Differentiation

Say we have a function $z = f(x, y, \dots)$ of several variables. Consider holding the other variables as constants and leaving x as the only 'remaining' variable, before differentiating with respect to x. We repeat this process again and again for the other variables until we get the respective partial derivatives of the function with respect to each variable. Such partial derivatives are notated as f_x,,f_y,f_z etc.

Mathematical Prerequisites

The Gradient Vector

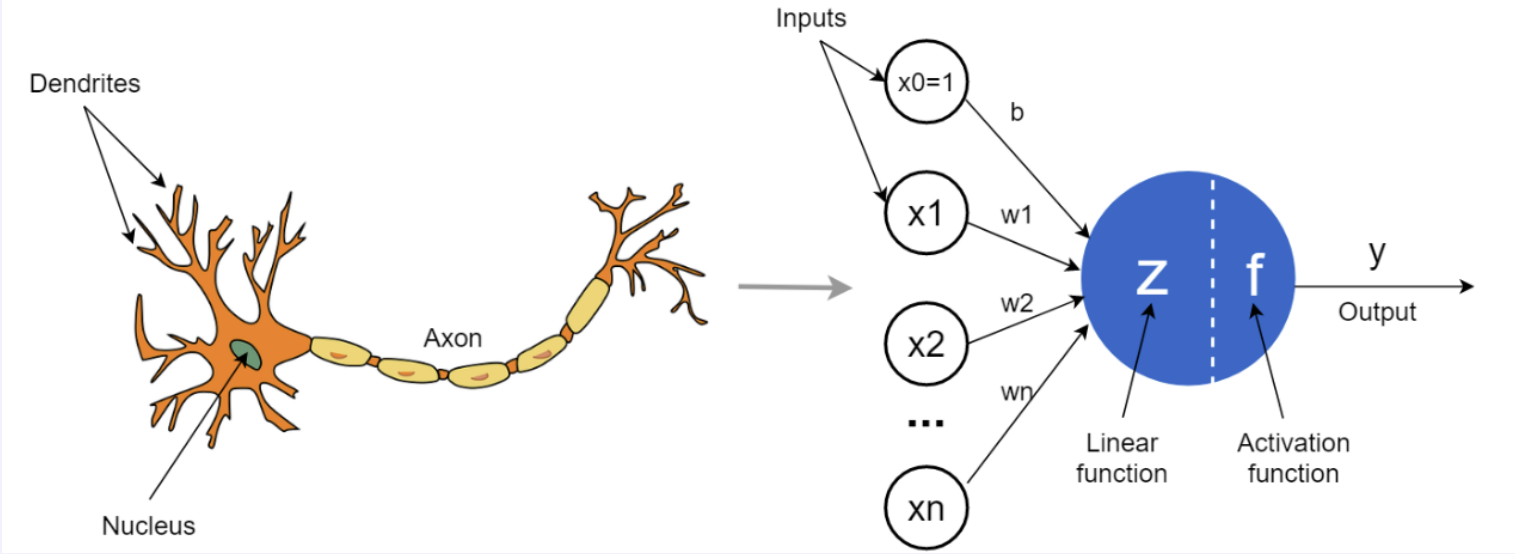
Having obtained the partial derivatives of our function of several variables, we can finally get the "gradient" of this function.

$$\nabla f = \begin{pmatrix} f_x \\ f_y \\ \vdots \\ f_? \end{pmatrix}$$

This allows us to generalise the idea of a "gradient" to several dimensions.

Formulation of Neural Networks

Biological Neuron vs Artificial Neuron



Formulation of Neural Networks

Mathematical Formulation of Neural Networks

For each neuron in a layer, we have:

$$y = f(b(1) + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n)$$

Let us now break down what this means:

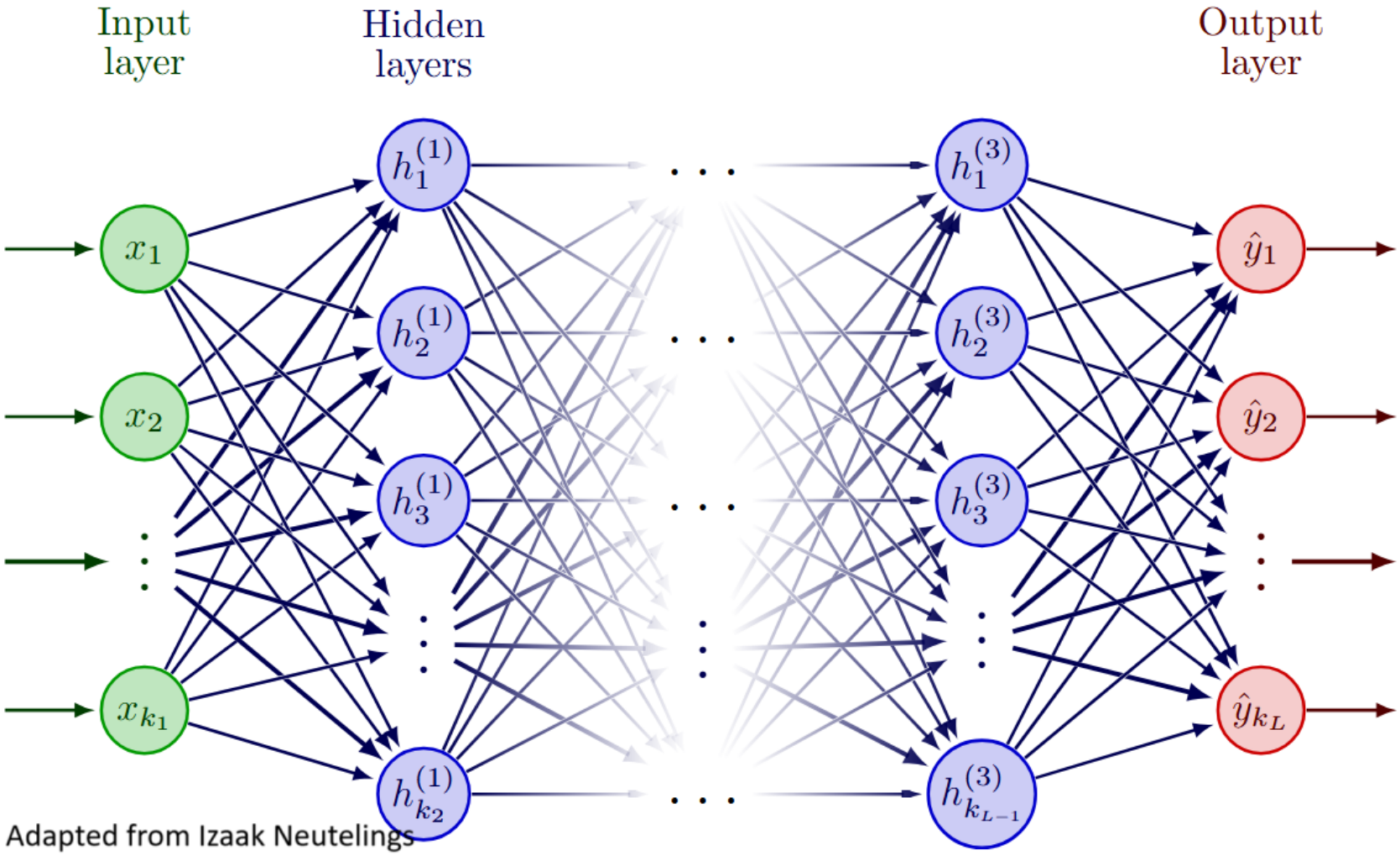
1. $x_1, x_2, \ldots, x_n$ are inputs to the neuron
2. $w_1, w_2, \ldots, w_n$ are weights to each input.
3. b is the bias.
4. f is the activation function.

Formulation of Neural Networks

Further Clarification

1. A weight is basically a measure of how important each input is to a neuron's output. Think of a student's score in some subjects. Some subjects(inputs) are more important(higher weights) then other subjects(lower weights).
2. A bias is a parameter allowing the model to shift the activation function. Intuitively, consider the line $y = mx + b$. The intercept b lets the line move up or down, just like the bias lets a neuron adjust its output baseline.

Formulation of Neural Networks



Adapted from Izaak Neutelings

Formulation of Neural Networks

Exercise 3

Say we have m neurons connected to n inputs. For the j-th neuron, each input x_i is scaled by the weight w_{ji} and a bias b_j is added:

$$b_j + w_{j1}x_1 + w_{j2}x_2 + \ldots w_{jn}x_n$$

So the question is, what is the corresponding expression for n neurons?

Formulation of Neural Networks

Answer to Exercise 3

We can represent the list of inputs as

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

So for n neurons, we have

$$\begin{pmatrix} b_1 + w_{11}x_1 + w_{12}x_2 + \cdots + w_{1n}x_n \\ b_2 + w_{21}x_1 + w_{22}x_2 + \cdots + w_{2n}x_n \\ \vdots \\ b_m + w_{m1}x_1 + w_{m2}x_2 + \cdots + w_{mn}x_n \end{pmatrix} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

Formulation of Neural Networks

Remark on Answer to Exercise 3

We can visualise $\mathbf{Wx}+\mathbf{b}$ as scaling(stretching/squashing) and translation(shifting).

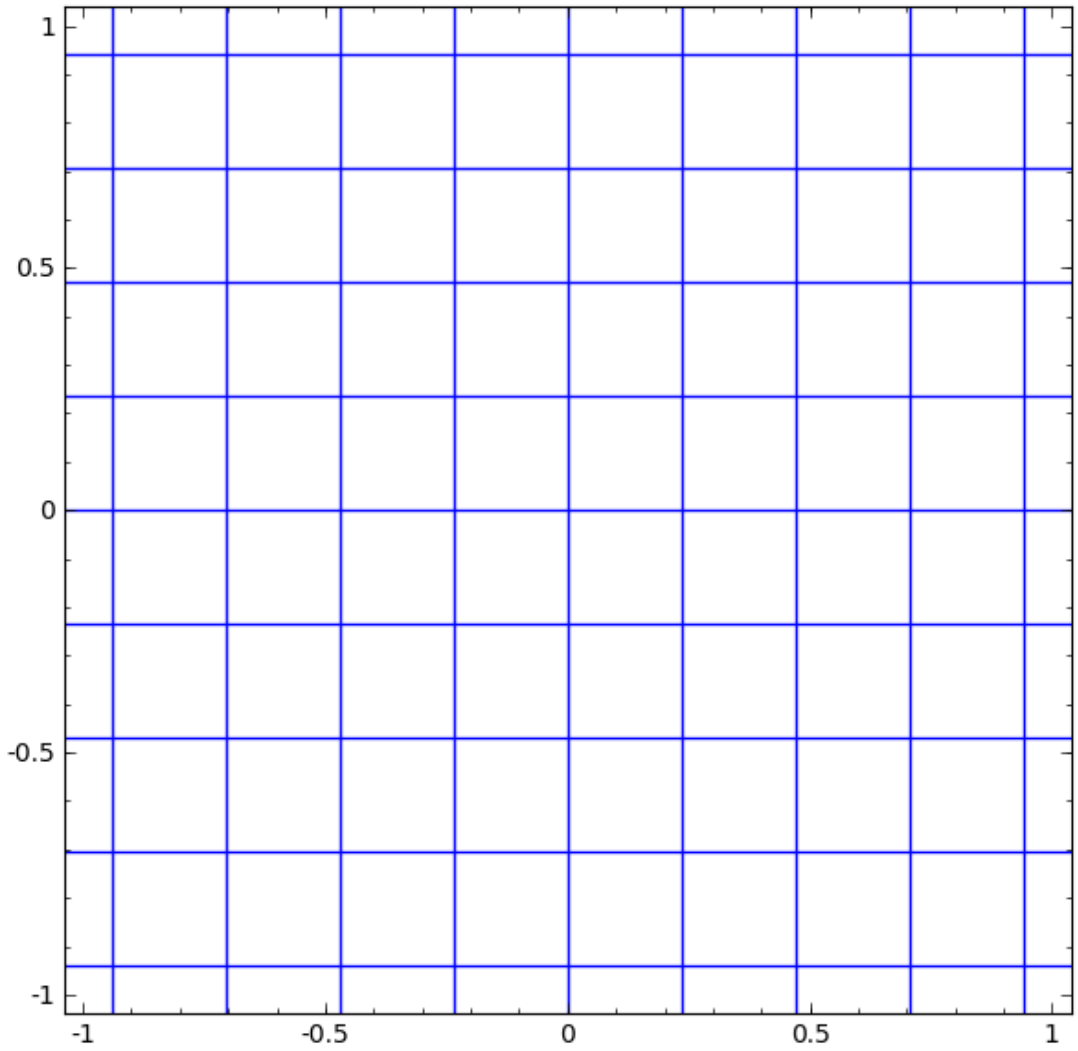
Formulation of Neural Networks

A Layer of the Neural Network

A layer of a neural netowrk can thus be written as

$$\mathbf{h}^{(l)} = f\left(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right) \tag{1}$$

Formulation of Neural Networks



Gradient Descent

Loss Function

A loss function is a mathematical formula that tells a machine learning model how wrong its predictions are. It measures the difference between the predicted output and the true(actual) value.

Gradient Descent

An Example of a Loss Function

Say we want to carry out some good old classification between two classes. The Binary Cross-Entropy Loss Function is:

$$\text{Loss} = -[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})]$$

Where y is the true label(0 or 1) and $\hat{y}$ is the predicted probability that the instance belongs to class 1.

Gradient Descent

Example Calculation 1

Let $y = 1$ and $\hat{y} = 0.9$. Then the loss calculation is given by

$$\text{Loss} = -[y \cdot \log(\hat{y}) + (1-y) \cdot \log(1-\hat{y})] = -\log(0.9) = 0.1054$$

Example Calculation 2

Let $y = 1$ and $\hat{y} = 0.1$. Then the loss calculation is given by

$$\text{Loss} = -[y \cdot \log(\hat{y}) + (1-y) \cdot \log(1-\hat{y})] = -\log(0.1) = 2.3026$$

The loss is higher when the model is confident but wrong, and lower when the model is confident and correct.

Gradient Descent

Goal of Gradient Descent

The goal of gradient descent is to find the best parameters(such as weights and biases) that minimise the error between predictions and actual results.

Gradient Descent

An Analogy for Gradient Descent

Think of the loss function as a hilly landscape, which tells you how *wrong* the model is. A peak on the hill indicates a big error, while a valley indicates a low error.

Gradient Descent

How Gradient Descent Works

1. Start somewhere on the hill(random weights).
2. Find the gradient of the hill at that point as it signifies which direction is downhill.
3. Take a small step in that direction.
4. Repeat this process until we have reached a valley(YIPPEE!!!!).

Gradient Descent

Mathematical Formulation of Gradient Descent

$$x_{n+1} = x_n - \alpha f'(x_n)$$

where α is the step size and $f'(x_n)$ is the steepness.

Gradient Descent

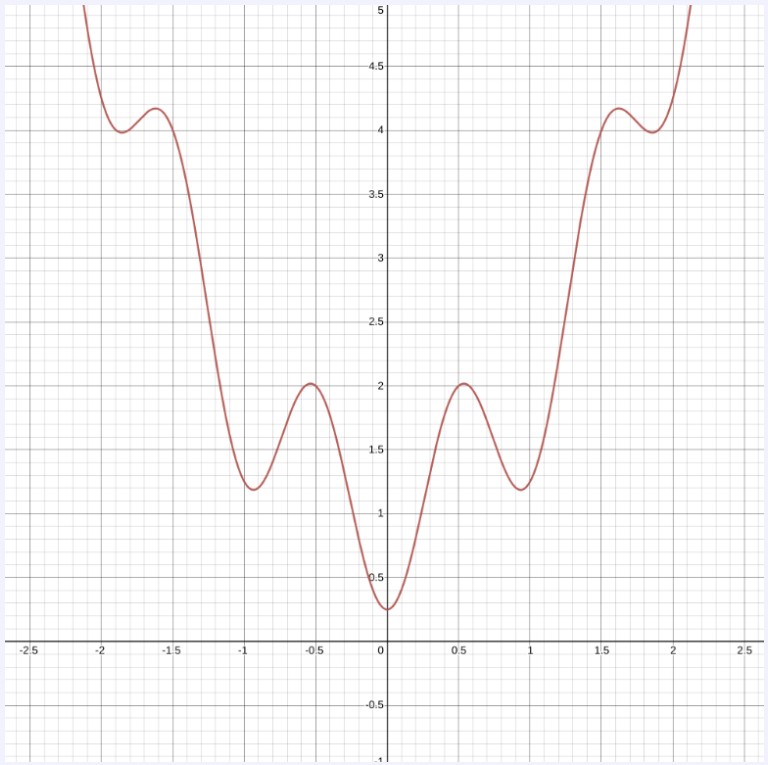
Problems Associated with Gradient Descent

1. The value of α chosen is too small.
2. The value of α chosen is too large.

Gradient Descent

Problems Associated with Gradient Descent

The algorithm does not necessarily know if it has reached the **global** minimum, as opposed to simply reaching a **local** minimum. Consider the following graph:



Convolution

A Simple Question

Consider a 2D image which can be represented as a matrix. Since a neural network can only accept a 1D vector as an input, how can we give the neural network an input that shows the positional information of the pixels?

Convolution

The Solution

The solution to this question lies in a mathematical operation called **convolution**!

Convolution

Why Use a Convolution?

A convolution turns a small patch of an image into a single value in a feature map, helping detect patterns like edges or textures

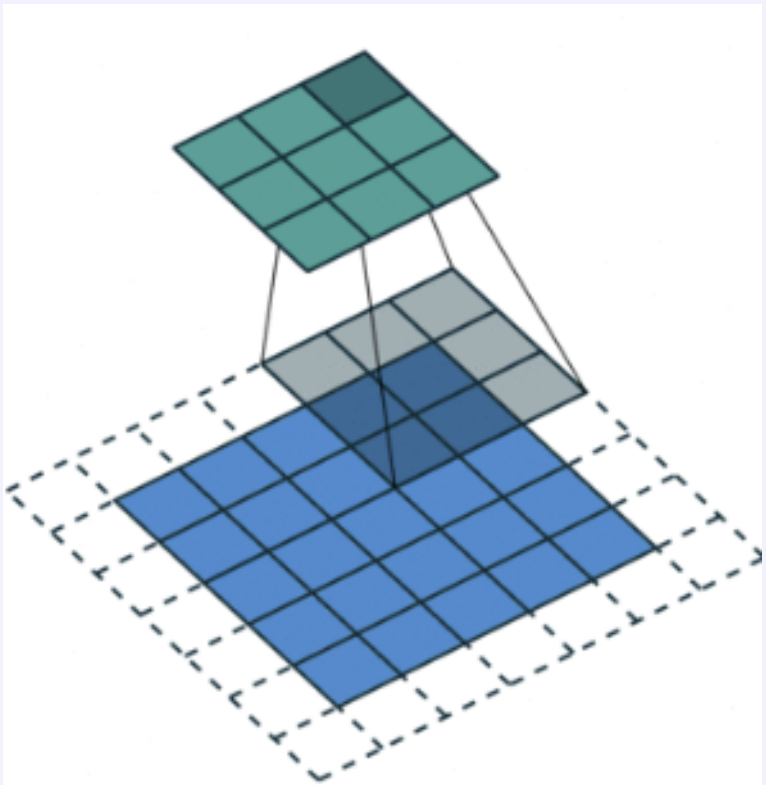
Convolution

Definition of a Convolution

A convolution is a mathematical operation that combines two functions(or matrices) to produce a third. In the context of image processing, it involves applying a small matrix, the kernel/filter, to an input image to produce an output image that highlights certain features such as edges, textures and patterns.

Convolution

Visualisation of Convolution



Backpropagation

Why Backpropagation?

Backpropagation is how a neural network learns. It helps adjust the weights in the network so the connections get more accurate over time.

The goal of backpropagation is to minimise the loss by tweaking the weights using gradient descent. Basically, it is a tool to find the optimal combination of weights in the network.

Backpropagation

How Backpropagation Works

Consider a simple 3-layer neural network.

$$\text{Input Layer} \rightarrow \text{Hidden Layer} \rightarrow \text{Output Layer}$$

For this neural network, we do the following 3 steps:

1. Forward Pass
2. Compute Loss
3. Backward Pass(Backpropagation!!!)

Backpropagation

Forward Pass and Computing Loss

1. Inputs pass through the network.
2. Each neuron applies weights, sums inputs, applies activation, and passes values forward.
3. Final output is compared with the true value to compute loss.

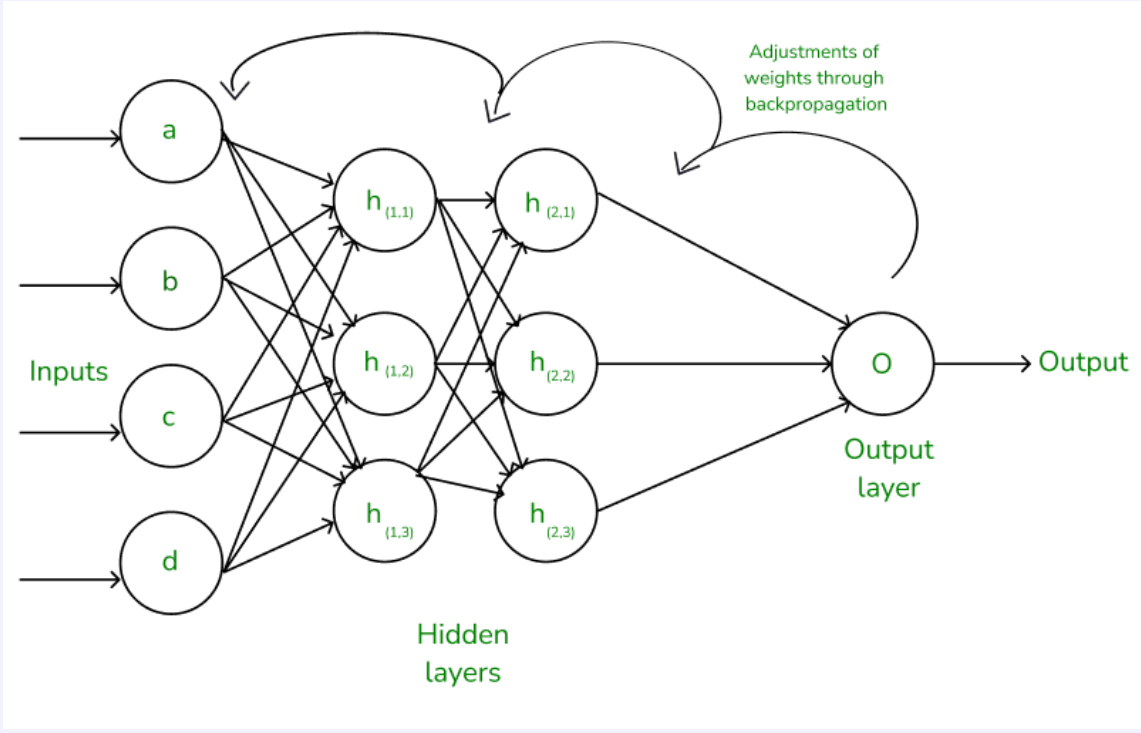
Backpropagation

Backward Pass(Backpropagation)

1. Calculate how much each weight contributed to the error.
2. Use the chain rule to propagate the error backwards from output to input.
3. Update weights using gradient descent.

Backpropagation

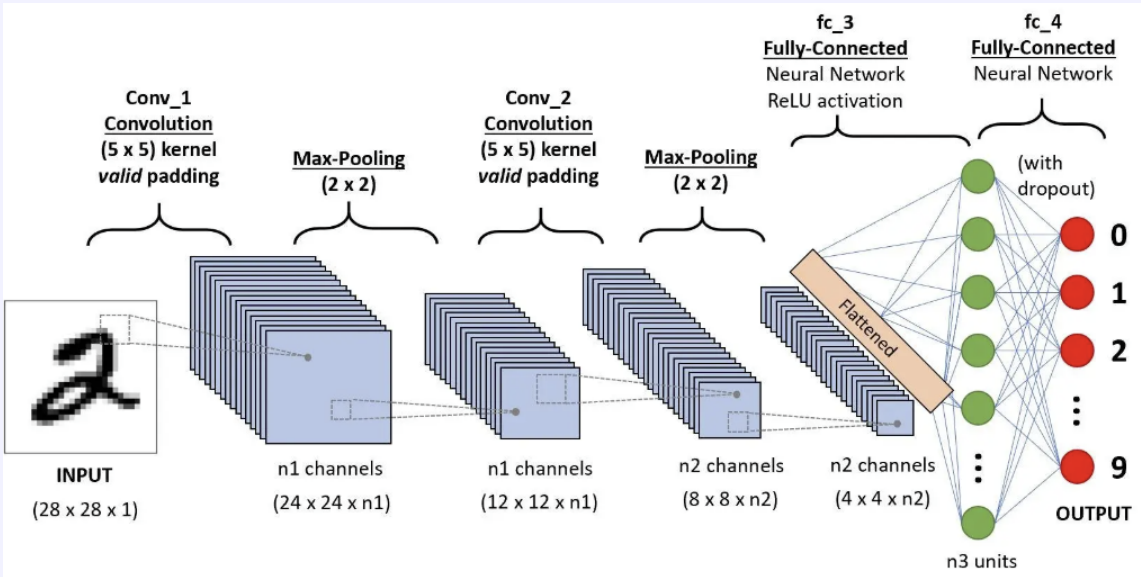
Visualising Backpropagation



Convolutional Neural Networks

What is a Convolutional Neural Network(CNN)?

A CNN is a type of neural network that is especially good at processing images. Simply speaking, a CNN is a fancy visual pattern detector.



Convolutional Neural Networks

How CNNs Work

1. The network uses filters that slide over the image to detect features(this is like in convolutions).
2. The filters "shrink" the image by only keeping the "essence" of the area, allowing the network to focus on major features.
3. After extracting features, the CNN uses regular neural network layers to interpret those features, and makes a prediction.

Convolutional Neural Networks

Making Sense of How Neural Networks Learn

Imagine you are identifying an animal in a picture.

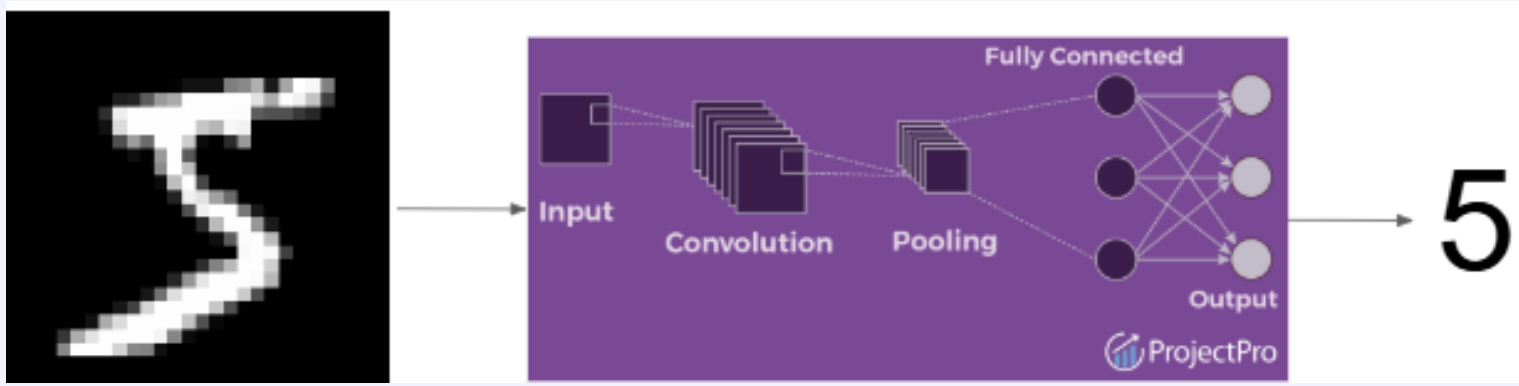
1. You first notice shapes(ears,tails,whiskers etc).
2. You combine those shapes into higher-order features(face of a cat).
3. You conclude what the animal is.

This is exactly how a CNN learns.

Handwritten Digit Recognition

Schematic Flow of Digit Recognition by CNNs

This is perhaps the most famous application of Convolutional Neural Networks.



Handwritten Digit Recognition

Handwritten Digit Recognition

Let's now try a Python simulation of a simple CNN!

Advantages and Disadvantages of Neural Networks

Advantages of Neural Networks

1. They are good at identifying patterns in data(well, obviously).
2. By utilising large datasets, neural networks can learn patterns that more traditional models(such as linear regression and decision trees) cannot detect.
3. Neural networks require less programming effort as compared to building other models from scratch. One just needs a pre-trained model which can be adapted for one's needs.

Advantages and Disadvantages of Neural Networks

Disadvantages of Neural Networks

1. It can be difficult or impossible to explain the decisions made by neural networks due to their "black-box" nature. Because of this, researchers may be unaware of important features present in their data being learned by the model.
2. Neural networks are extremely data-hungry because they rely on large amounts of data for training and optimisation of the models.