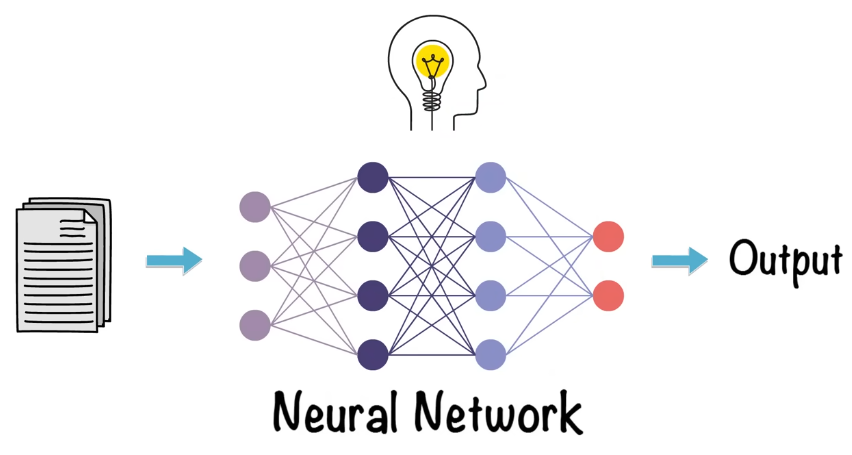
Resource one: <https://youtu.be/UOvPeC8WOt8?si=E10_J_SLDRSv4udk>

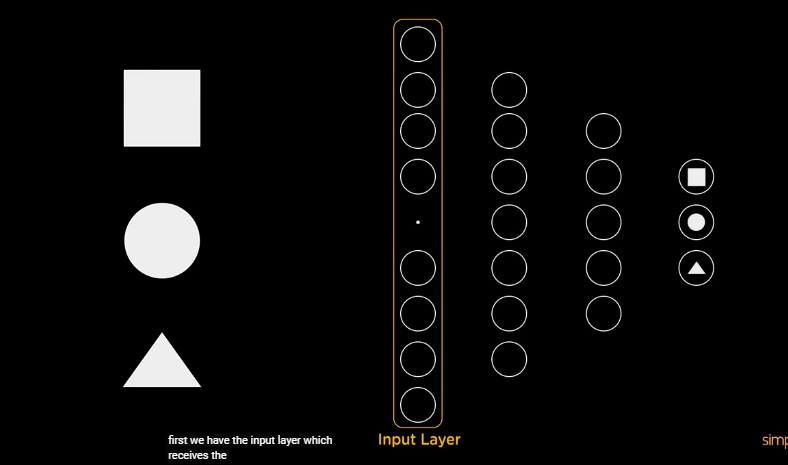


In simple term, neural networks are inspired by human brain that takes input , process the data or do internal presentation of data and give the output.



Now above is the neural network that differentiate between Square, Circle, Triangle.

Basically the circles we are seeing in above picture are **neurons** also known as **nodes.**



First layer is the input layer which takes the input .

A screen shot of a black and white screen

Description automatically generated

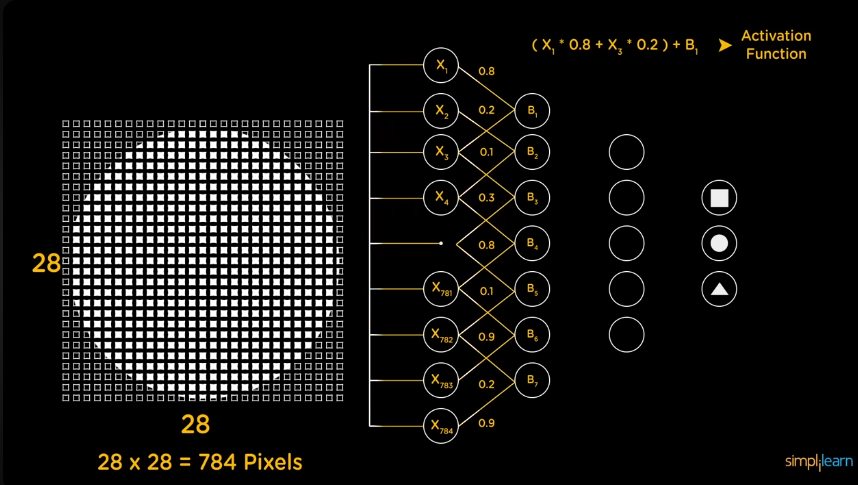
The last layer is the **Output layer** which predicts our output.

Now according to my understanding its means that in ML and DL , jo Ml hai usme jo input data kay features hain wo denay hotay hain but in DL we just input the data it will extract the feature automatically BUT like in supervise ML we define output label to validate the prediction/inference ussi tarah say agar above pic may dekhein toh neural network mabi jo output layer hai usme be humnay output labels diya hain takay at the end wo identify kray kay prediction and jo accurate output hai usme kitna difference hai and then on the basis of that it validate the output.

A screenshot of a computer

Description automatically generated

Then, the layers b/w input and output are the **hidden layers ,** this layer perform most of the computation, data processing and internal representation of data.



Let's break this down step by step, starting from the image you uploaded and continuing with the deeper process of how the neural network identifies shapes like a circle, square, or triangle.

**Step 1: Input Layer (28x28 Pixels)**

* The image you provided shows a 28x28 grid of pixels. These pixels represent the image of the circle we want to classify.
* **Each pixel is represented by a value (X₁, X₂, X₃, …, X₇₈₄)**, where each of these values can be a number between 0 (black) and 255 (white), indicating how bright the pixel is. In total, there are 28x28 = 784 pixels.

**Step 2: Flattening the Image**

* Before the image enters the neural network, it gets flattened. This means that the 2D grid (28x28) is transformed into a 1D list of 784 pixel values. Now, each pixel becomes a feature/input to the network, labeled as **X₁, X₂, X₃, …, X₇₈₄**.

**Step 3: Input Layer to Hidden Layers**

* Each pixel value (X₁ to X₇₈₄) is fed into the **input layer**. The network has several layers, starting with the input layer that directly connects to the **hidden layers**.
* The hidden layers consist of **neurons** (the circles you see in the middle of the image). These neurons take the weighted sum of inputs, which means each neuron gets the pixel data from all 784 inputs (X₁ to X₇₈₄).

**Step 4: Weighted Connections**

* Each connection between input pixels and neurons in the hidden layers has a **weight (W₁, W₂, …)** associated with it. This weight determines the importance of that specific input.
* For instance, if a certain pixel (like a boundary of the circle) is more important, its corresponding weight will be higher. These weights are learned during training.

Q- Ab yaha question yeh haka yeh jo weights hotay hain yeh kasay sum hotay hain and isme jo further jo bias hota hai wo kasay add hota hai?

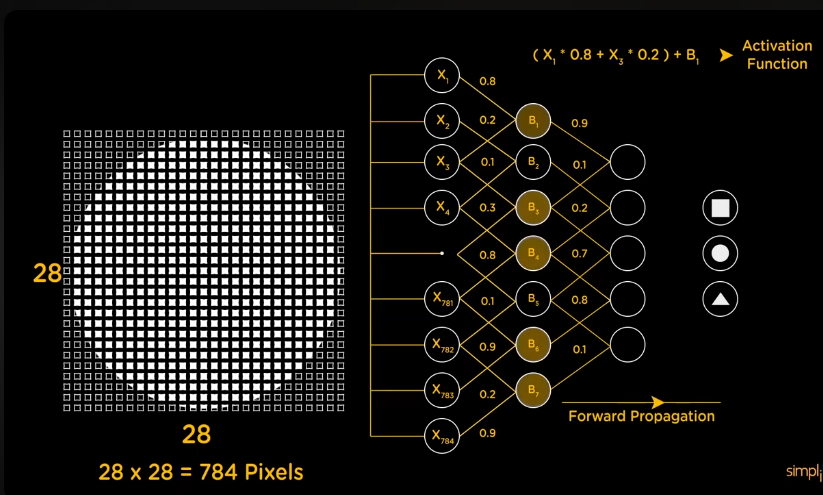
Now iska answer yeh haka if you see in above image, so jo first input layer hai usme jo nodes hain unka connection build ho rha hai **first hidden layer** say, Ab jo connection apas may create ho rahy hain unka **ek specify weight hota hai joka ek numeric value hi hoti hai** jo yeh define krti haka inn do neurons/nodes may jo relation/connection wo kitna strong hai.

Ab yeh jo hum continuously uper yeh cheez keh rhay hain kay bhai pehla jo input layer may neuron may jo pixel ki value hai wo multiply hogi with their corresponding weight value , then sab weights ka sum hoga or then phr usme bias add hoga , then jo finally value hogi after adding bias wo jayegi activation function (sigmoid or ReLU) kay pass then phr wo decide krega kay neuron fire hoga ya nhi. Ab yeh pura jo flow define kia hai yeh hoga kasay ?

Toh mainly jo flow uper btya hai wo iss tarah say execute hoga kay like if you see in above image jo input layer may first node hai X1 it has pixel value = 1 and the value of weight is **0.8** , isi tarah jo input layer may jo X3 node hai wobi hidden layer may usi neuron say connected hai jissay X1 connected hai BUT both connection have different values of weight which means that the weight with X1 is **0.8** and weight with X3 is **0.2 .** Now isme jo hum kehtay haina kay Weighted Sum or sum of weight toh wo yehi hota haka ek layer ma jo neurons/nodes next layer may neuron say connected hain unka jo weight hota hai usko sumup krdena that is known as **sum of weight**. In our case we will do it like this **Sum of weight = (X1 \* W1) + (X3 \* W2) = (1 \* 0.8) + (3 + 0.2) = 1.56**

Now we have add the bias value, So basically if you see in above pic toh after input layer jo first hidden layer hai usme jo neuron hain unme b1,b2 and so on likhein hain toh basically jo yeh b1,b2 likha hai yehi **bias** hai or inki jaga koi na koi value hoti hai that is known as **bias value** , So wo jo bias value hoti hai wo hum final **Sum of Weight** jo uper kia hai usme add krtay hain and after adding bias value jo final value ati hai usko agay **activation function** may pass krdetay hain , now here first we calculate final value which will be **final value = Sum of weights + Bias value** for e,g we assume the bias value as 1 so final value = 1.56 + 1 = **2.56**

Now This **final value = 2.56** will be passed to **activation function** (ReLU or sigmoid) so this function will simply check that if the final value is passing the threshold/target value so it activate or fire the neuron and its mean kay jab neuron fire hojayega toh uska jo output value hai wo next layer may jo neuron hoga usko pass hojayegi or jo output value hogi wo wohi value hogi jo humna final value define ki thi.



Now you can see that In above Image that jo first hidden layer may neurons hain unme say jo neuron activate huye that are highlighted unki jo **final value** wo as input pass ki next layer may jo neuron hain and Now the **B1** **and B3 are connected to first neuron in 2nd hidden layer.** And isi process ko jo data passing hoti hai after prediction from one layer to another is known as **forward propagation**. (propagation means that spreading or multiplication of any thing)

Q- Ab ek sawal yeh be haka jo Weight and bias ki value hoti hai wo toh hum input nhi krtay toh wo value kasay milti hai ?

Artificial Neural Network (ANN) mein **weights aur bias** ki values training ke process ke dauran adjust hoti hain. Initially, ye values randomly assign hoti hain, lekin training ke baad backpropagation algorithm ke zariye inhe adjust kiya jata hai. Ab mai tumhe step-by-step samjhata hoon ki ye kaise hota hai:

**1. Initialization (Shuruwati Values):**

* Jab hum ANN ko start karte hain, to **weights aur bias** ki initial values random hoti hain. Yeh ek random number generator ke zariye decide ki jati hain, takay network kisi specific direction mein biased na ho.
* For example, ek input neuron se kisi hidden neuron tak ka weight random value se start hota hai, jaise W1=0.5W\_1 = 0.5W1​=0.5.

**2. Feedforward Process:**

* Jab hum ANN ko input dete hain (jaise image ya kisi number ka data), to ye input forward pass ke through layers mein pass hota hai. Har neuron weighted sum calculate karta hai (jo tum pehle samajh chuke ho), bias add karta hai, aur phir activation function use karke output deta hai.

**3. Error Calculation:**

* Jab final output layer mein result aata hai, hum ise expected output ke saath compare karte hain. Is comparison ke zariye hume **error** pata chalta hai, jo basically yeh hota hai ki model ne galat jawab diya hai ya sahi. For example, agar hum "2" digit ko recognize karwana chahte hain, lekin network ne "6" diya, to error hogi.

Error=Actual output−Expected output\text{Error} = \text{Actual output} - \text{Expected output}Error=Actual output−Expected output

**4. Backpropagation Algorithm:**

* **Backpropagation** ek method hai jo yeh error ko reverse direction mein layers ke through propagate karta hai, peeche ki taraf. Is algorithm ka main maksad yeh hai ke har weight aur bias ko update kiya jaye based on error.
* **Gradient Descent** naam ka technique use hota hai jisme error ko minimize karne ke liye weights aur bias ko update kiya jata hai. Gradient descent ke through hum "error ka slope" find karte hain, aur phir us slope ke opposite direction mein weights ko adjust karte hain.

**Step 5: Activation Function**

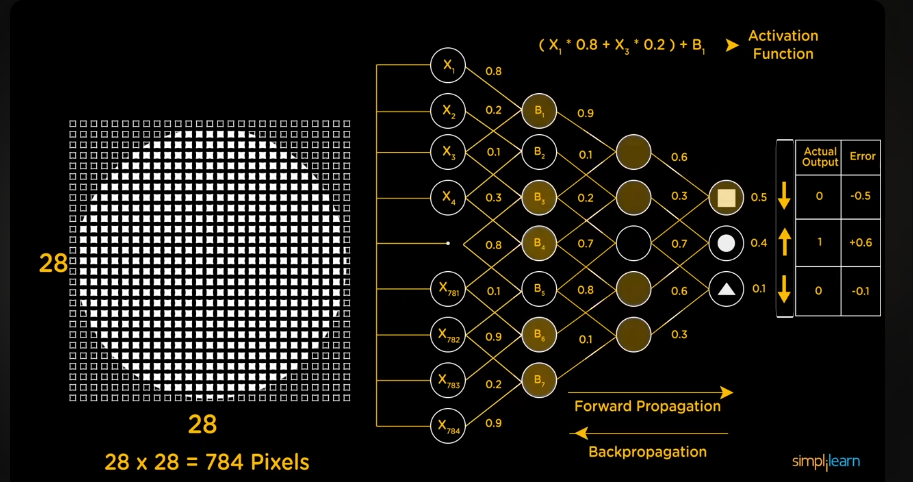
* Once the weighted sum is computed for each neuron, the output passes through an **activation function** (such as ReLU or sigmoid). The activation function decides whether the neuron should "fire" or stay inactive.
* If the neuron fires (i.e., passes a certain threshold), it sends its output to the next layer.

**Step 6: Passing Through Hidden Layers**

* The process repeats across multiple hidden layers. Each hidden layer takes inputs from the previous layer, computes weighted sums, applies the activation function, and passes the result to the next layer.
* The hidden layers help the network learn complex patterns in the data. For example, one layer might learn to detect edges, another might detect shapes, and eventually, the network will be able to recognize whether the input is a circle, square, or triangle.

**Step 7: Output Layer**

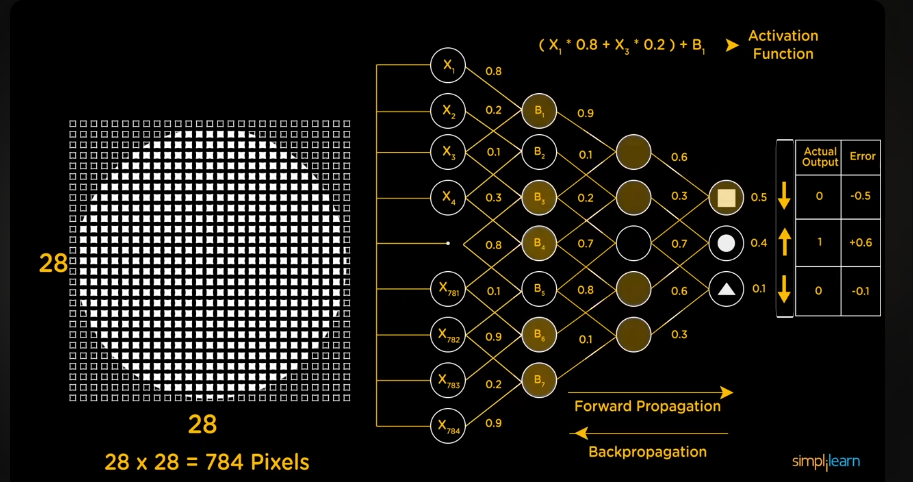
* Finally, the output from the last hidden layer goes to the **output layer**. In this case, the output layer has three neurons representing the three possible shapes: circle, square, and triangle.
* Each of these neurons will output a probability score. For example:
  + Circle neuron: 0.9 (90% chance it’s a circle)
  + Square neuron: 0.05 (5% chance it’s a square)
  + Triangle neuron: 0.05 (5% chance it’s a triangle)



Now if you see in our case so the model is predicting wrong as its highlighting the output as **square.** But we will see that in next step which **Step 8.**

**Step 8: Prediction**

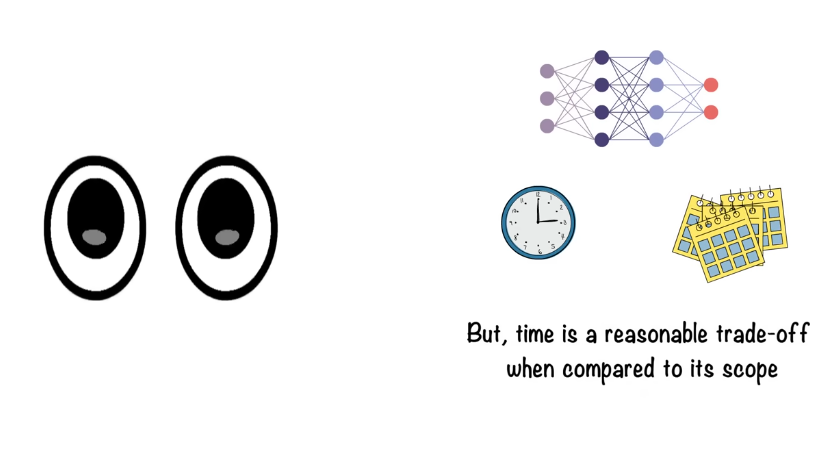
* The network will pick the shape with the highest probability as the final prediction. Since the circle neuron has the highest value (0.9), the model will predict that the shape is a circle.



Now as in our case we can see in above image that prediction is wrong , So how to fix it. So basically if you the right corner we have a chart as well where **Actual output label** and the **error detected is mention** , which is clearly saying that jo square wala output node hai uski value **0 honi chaiya bcuz jo downward arrow hai who yeh keh rha haka iski value actual value say upward hai yeh downward honi chaiya or jo error may value -0.5 aa rhi hai uska yehi mtlb haka jo predicted value hai square walay output node ki usme say 0.5 minus krdo so then it will become 0 which is accurate.** Isi tarah jo circle wala node hai uski predict value **0.4** hai halaka kay it should be **1** bcuz in actual output it is **1** so is ilia error may it is saying to add +0.6 into 0.4 so that it become **1.** And same rule go for triangle node.

Now ab on the basis of these comparisons which we created done in above paragraph So as we are training model toh isilia jo information humna uper paragraph may find ki hai kay yeh yeh adjustment honi chaiya yeh info backward transfer hojayegi network ko and it is known as **Back propagation** and that will adjust the values of **weight** and **bias** and then again testing hogi agr toh thk prediction hui toh OK agar nhi toh dobara say **Back propagation** hogi and yeh process iteratively chlta rahega until kay ek neural network achay say train na hojaye.

**And must remember that error is always calculated at the last layer of neural network which is output layer.**

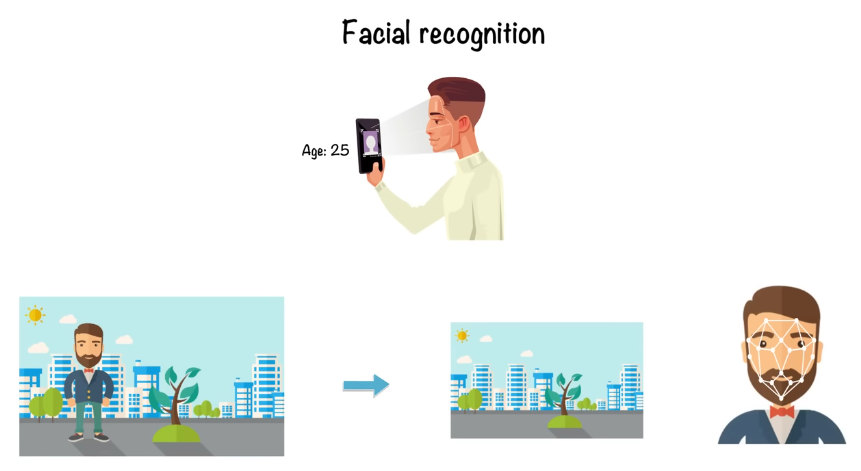


Above pic is telling us that these neural network can take weeks, months or even years as well to train.

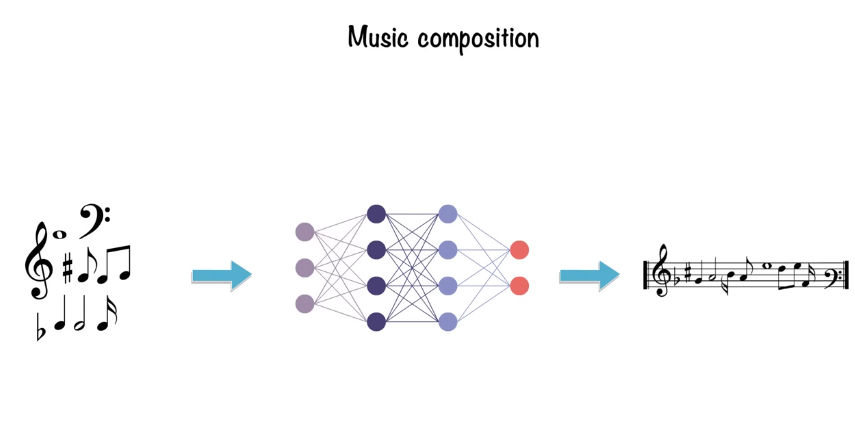
**Summary of the Process:**

1. **Input**: 28x28 pixels of the circle image, converted into 784 values (X₁ to X₇₈₄).
2. **Flatten**: The image is flattened into a 1D list.
3. **Hidden Layers**: Each pixel value is passed through neurons in hidden layers, where they get combined with weights and activated.
4. **Weights & Biases**: The neurons use weights and biases to adjust the importance of each input.
5. **Activation Function**: Decides which neurons fire.
6. **Output**: The model predicts whether it’s a circle, square, or triangle based on the output neuron with the highest value.

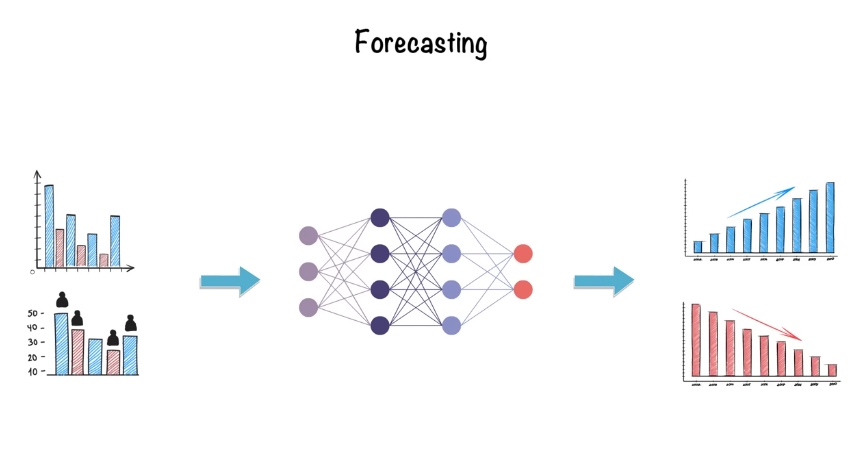
**Applications**

****

Now in above application which is of Facial recognition , so the first layer is detecting the background in image and then the next layer is pointing out the features of face by some points.



Above application is telling that neural networks can also be used in music composition like it takes music as input and finds the pattern in it.



Above example is of weather prediction and stock prediction, that when we provide the input to network so it predict the output .

Q-Which essential component of Artificial Neural Network performs weighted summation and applies activation function on input data to produce an output?

* + Answer is Neuron