Introduction:
"ShopNow," a large e-commerce retailer, is looking to improve its ability to predict which users are likely to make purchases on their platform. They decide to implement a **neural network** model that can process complex data from users' interactions—such as the number of product views, the duration of time spent on the platform, and previous purchase history—to generate predictions about customer behavior.
To understand how a neural network works in this context, let's go through the various components of a neural network in detail.

---

1. Neurons and Layers
A neural network is built using units called **neurons**, much like how our brains work. These neurons are organized into layers. Each neuron in a layer is connected to neurons in the next layer.

- **Input Layer**: This is where the raw data enters the network. In our case, for each customer, the input data would include:
    - Number of product views
    - Time spent browsing products
    - Total number of items added to the cart
    - Past purchases
    - Location, device used, etc.

Let's say a customer viewed 5 products, spent 12 minutes on the site, and added 2 items to the cart. This data enters the **input layer**.

- **Hidden Layers**: After the input layer, the data moves into the hidden layers, where the real processing happens. Hidden layers learn from the data and capture underlying patterns, such as how the combination of high browsing time and repeated product views might indicate high purchase intent.

For instance, ShopNow's hidden layer might discover that users who view the same product more than 3 times and spend over 10 minutes browsing have a 60% chance of making a purchase.

- **Output Layer**: The output layer provides the result—whether the customer is likely to make a purchase (yes or no), or a probability score (e.g., "this customer has a 70% likelihood of buying a product").

---

2. Weights and Biases
In a neural network, connections between neurons are assigned **weights** and each neuron has a **bias**. These are like dials that adjust the flow of information and help the network "learn" by emphasizing certain inputs over others.

- **Weights**: These values determine how much influence one neuron has on the next. In ShopNow's case, the system might assign a higher weight to "time spent on product pages" compared to "total clicks on ads," because time on a product page is more strongly associated with purchase behavior.

For example:

- Time spent = 10 minutes (weight = 0.8)
- Total clicks on ads = 3 clicks (weight = 0.3)

Here, the weight for time spent is higher because it is more predictive of purchase behavior.

- **Bias**: Bias adds flexibility to the model. If every customer who viewed a product more than 3 times ended up buying it, bias would adjust to account for that pattern, ensuring the model doesn't miss consistent trends.

In technical terms, the weighted sum of inputs is passed to a neuron, and then bias is added to shift the value slightly. This helps the network adapt to different scenarios.

---

3. Forward Propagation

Forward propagation is the process of sending input data through the network, from the input layer through the hidden layers, and finally to the output layer, generating a prediction.

Let's walk through a simplified example with ShopNow:

- **Input**: A customer viewed 4 products, spent 15 minutes on the website, and clicked on 2 ads.

In forward propagation:

  - Each of these inputs is multiplied by their respective weights.
  - The results are added together, and biases are applied to shift the results.
  - The output is a prediction: "This customer has an 80% chance of making a purchase."

Forward propagation is essentially the "prediction phase" of the neural network.

---

4. Activation Functions

After each neuron processes the inputs, it applies an **activation function**. The activation function determines whether the information should be passed to the next layer and in what form. This step is essential for introducing non-linearity into the model, enabling it to learn complex patterns.

- **ReLU (Rectified Linear Unit)**: This is a commonly used activation function in hidden layers. It outputs zero for negative values and keeps positive values unchanged. This makes it ideal for filtering out irrelevant information while focusing on significant data points.

For example:

  - If the weighted sum from the forward propagation is -5, ReLU will output 0 (since negative values are set to zero).
  - If the sum is +3, ReLU will output 3.

- **Sigmoid Function**: Often used in the output layer for binary classification problems. Sigmoid squashes the output into a value between 0 and 1, representing probabilities. For example, a 0.85 output from the sigmoid function could be interpreted as an 85% chance that the customer will make a purchase.

---

5. Loss Function

Once the neural network makes a prediction, we need to measure how far off that prediction was from the actual outcome. The **loss function** calculates the error in prediction.

- **Loss Example**: ShopNow's model predicts an 85% chance that a customer will buy something, but the customer doesn't make a purchase. The loss function computes the difference between the predicted probability (0.85) and the actual result (0). A large difference means a high loss.

Common loss functions include:

- **Mean Squared Error (MSE)**: Typically used for regression tasks.
- **Cross-Entropy Loss**: Ideal for classification problems (like predicting purchases).

In our case, we'd use **Cross-Entropy Loss** to calculate how "off" our model was in predicting whether or not a customer would buy.

---

6. Backward Propagation

Now that we know how far off our predictions were, we need to adjust the model to improve future predictions. This is done through **backward propagation**.

- In backward propagation, the network calculates how much each weight contributed to the error. For example, it might discover that the weight for "time spent on the website" was too low and should be increased to give that feature more influence in the prediction process.

This process allows the neural network to "learn" from its mistakes by tweaking the weights and biases to make better predictions in the future.

---

7. Optimizer

The **optimizer** is the algorithm that updates the weights and biases during the backward propagation process. It determines how much the weights should be adjusted based on the errors.

- **Gradient Descent**: This is the most commonly used optimizer. It calculates the gradient (or slope) of the loss function and adjusts the weights in the direction that reduces the loss. Imagine you're trying to find the lowest point on a hill—gradient descent helps you figure out which direction to step to get there faster.

For ShopNow, gradient descent will tweak the weights for features like "product views" or "total time spent" so the model becomes more accurate at predicting whether a customer will make a purchase.

---

8. Batch Size

Neural networks don't process all the data at once. Instead, they divide the data into smaller chunks called **batches**.

- **Batch Size Example**: ShopNow might have data on 100,000 customers. Instead of processing all that data at once, the neural network can take batches of 500 or 1000 customers at a time, making training faster and more efficient.

Larger batch sizes make the training faster but less frequent, while smaller batch sizes result in more frequent updates but can be slower overall.

---

9. Metrics

In addition to loss functions, metrics help measure how well the model is performing. For ShopNow, the following metrics would be essential:

- **Accuracy**: The percentage of correct predictions. If the model predicts that 70 out of 100 customers are likely to buy and 65 actually do, the accuracy is 65%.
- **Precision**: Out of all the customers the model predicted would buy, how many actually made a purchase? For example, if the model predicted 70 customers would buy but only 50 did, the precision is 50/70 = 71%.

- **Recall**: Out of all the customers who made a purchase, how many did the model correctly identify? If 80 customers made a purchase but the model only predicted 60 of them, the recall is 60/80 = 75%.

These metrics help ShopNow understand the quality of their predictions and refine the model accordingly.

---

10. Regularization

One problem neural networks can face is **overfitting**. This happens when the model learns too much from the training data and performs poorly on new data.

Regularization is used to prevent overfitting.

- **L2 Regularization (Ridge)**: This method adds a penalty for large weights, keeping the weights smaller and making the model more generalizable.
- **Dropout**: A technique where random neurons are "dropped out" (ignored) during training. This forces the network to learn more robust patterns by not relying too heavily on any one neuron.

For ShopNow, regularization would prevent the model from over-learning specific details of the training data and instead focus on general patterns of purchase behavior, ensuring that predictions for new customers are accurate.