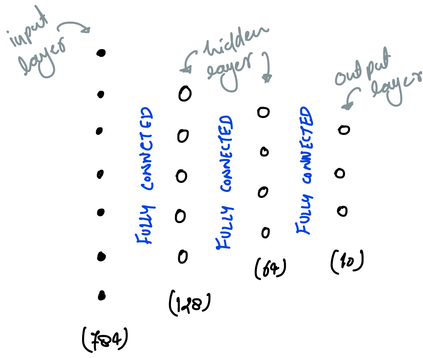
9.1 SOLUTION - DROPOUTS

12 September 2025 01:08 AM Avinash Yadav

1. DROPOUTS :

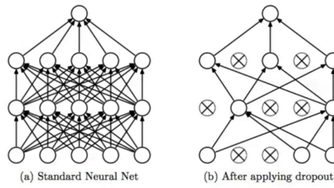


Architecture of our NN

What dropout does is that during each forward pass, it randomly turn offs some neurons

↓

This means deactivating all the incoming and outgoing connection from a particular neuron.



(a) Standard Neural Net (b) After applying dropout.

By doing Dropout →

1) The NN becomes a bit simpler
2) Since in forward pass we are using a different neural network, in way, due to which an effect of regularization can be seen. → And due this the overfitting is reduced.

1. Applied to The **Hidden Layers**
2. Applied **After The ReLU** Activation Function
3. Randomly **Turns off P% Neurons** in The Hidden Layer During Each Forward Pass
4. This Has A Regularization Effect
5. During Evaluation, Dropout Is Not Used

* ORIGINAL ARCHITECTURE :

```python
class MyNN(nn.Module):
    def __init__(self, num_features):
        super().__init__()

        self.model = nn.Sequential(
            nn.Linear(num_features, 128),
            nn.ReLU(),
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Linear(64, 10)
        )

    def forward(self, x):
        return self.model(x)
```

Python

* AFTER APPLYING DROPOUTS :-

```python
class MyNN(nn.Module):
    def __init__(self, num_features):
        super().__init__()

        self.model = nn.Sequential(
            nn.Linear(num_features, 128),
            nn.ReLU(),
            nn.Dropout(p=0.3),   # → dropout in 1st hidden layer
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Dropout(p=0.3),   # → dropout in 2nd hidden layer
            nn.Linear(64, 10)
        )

    def forward(self, x):
        return self.model(x)
```

Python