Neural Network Training Techniques

Initialization and Dropout

Introduction to Neural Network Training

Neural networks are powerful computational models inspired by the human brain, designed to recognize patterns and make predictions. Their ability to learn from data has revolutionized fields from image recognition to natural language processing.

Mimicking the Mind

Neural networks process information in layers, much like biological neurons.

The Training Imperative

Effective training ensures models learn optimal weights, crucial for accurate performance.

Optimizing Performance

Proper techniques prevent issues like underfitting or overfitting, maximizing model utility.

Training Process of a Neural Network

1

Initialize Parameters

- Randomly set **weights** (W) and **biases** (b) using methods like Xavier, He, or LeCun initialization.

2

Choose Optimization Algorithm

- Example: Gradient Descent, SGD, Adam, RMSprop

3

Repeat for many epochs:

- **Forward Pass** : Input -> Output Prediction
- **Calculate Loss:** Compare prediction with correct answer
- **Backpropagation** : Find how much each weight contributed to the error
- **Update parameters**: Adjust weight to reduce errors.

What is Initialization?

- **Initialization** is the process of setting the initial values of a neural network..
- **Weight Initialization** is the process of assigning the starting values for the weights of a neural network before training begins.

Problems with Poor Weight Initialization:

Vanishing Gradients

If initial weights are too **small**, the signals shrink as they move through layers.
Eg: (0.5)^10 = 0.00098 → almost zero signal

Exploding Gradients

Conversely, large weights can cause gradients to grow uncontrollably, destabilizing the network
.If the weights are too large, values grow exponentially as they pass through the layers.

Breaking Symmetry

If all weights starts with same value, neurons in the same layer learn identical features

.

Common Initialization Techniques

1

Zero Initialization

All weights are set to zero. This fails immediately due to the symmetry problem: all neurons in a layer learn identical features.

2

Random Initialization

Weights are set to small random values. Better than zero, but still prone to vanishing or exploding gradients, especially in deep networks.

3

Xavier (Glorot) Initialization

Designed for tanh and sigmoid activations. Balances input and output variances to keep activations in a reasonable range.

$$W_{ij} \sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}} + n_{\text{out}}}\right)$$

4

He (Kaiming) Initialization

Specifically for ReLU activations. Focuses on maintaining variance of activations as they propagate forward.

$$W_{ij} \sim \mathcal{N}\left(0, \frac{1}{n_{\text{in}}}\right)$$

5

LeCun Initialization

Tailored for SELU activation functions. It helps activations converge to a specific mean and variance, supporting self-normalization.

$$W_{ij} \sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}}}\right)$$

Comparison of Initialization Techniques

Choosing the right initialization technique is crucial for faster convergence and improved model accuracy. It depends heavily on the activation function used in your network.

Xavier/Glorot

- Best for sigmoid, tanh
- Balances input/output
- Speeds up convergence

He/Kaiming

- Ideal for ReLU, Leaky ReLU
- Prevents dying ReLUs
- Maintains variance

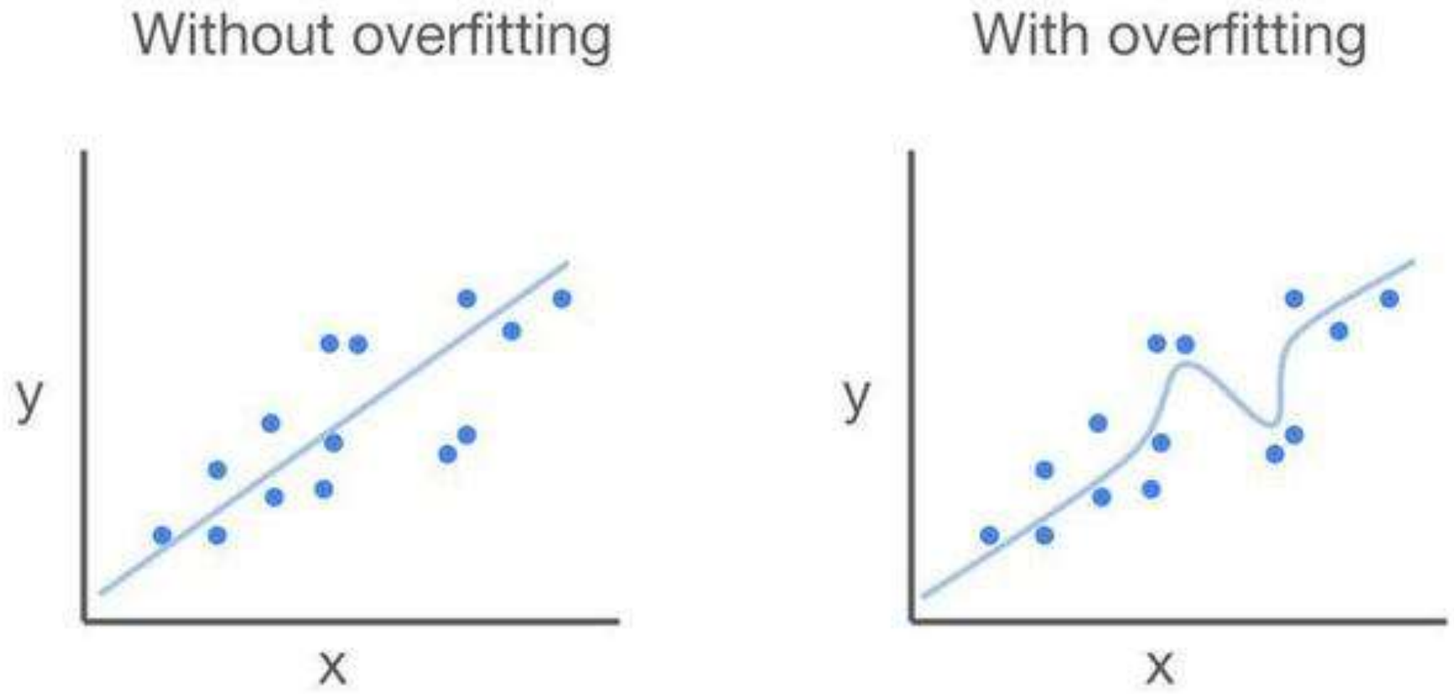
LeCun

- Specific to SELU
- Enables self-normalization
- Deep network stability

Overfitting

Overfitting occurs when a model learns the training data too well, capturing noise and specific examples rather than underlying patterns.

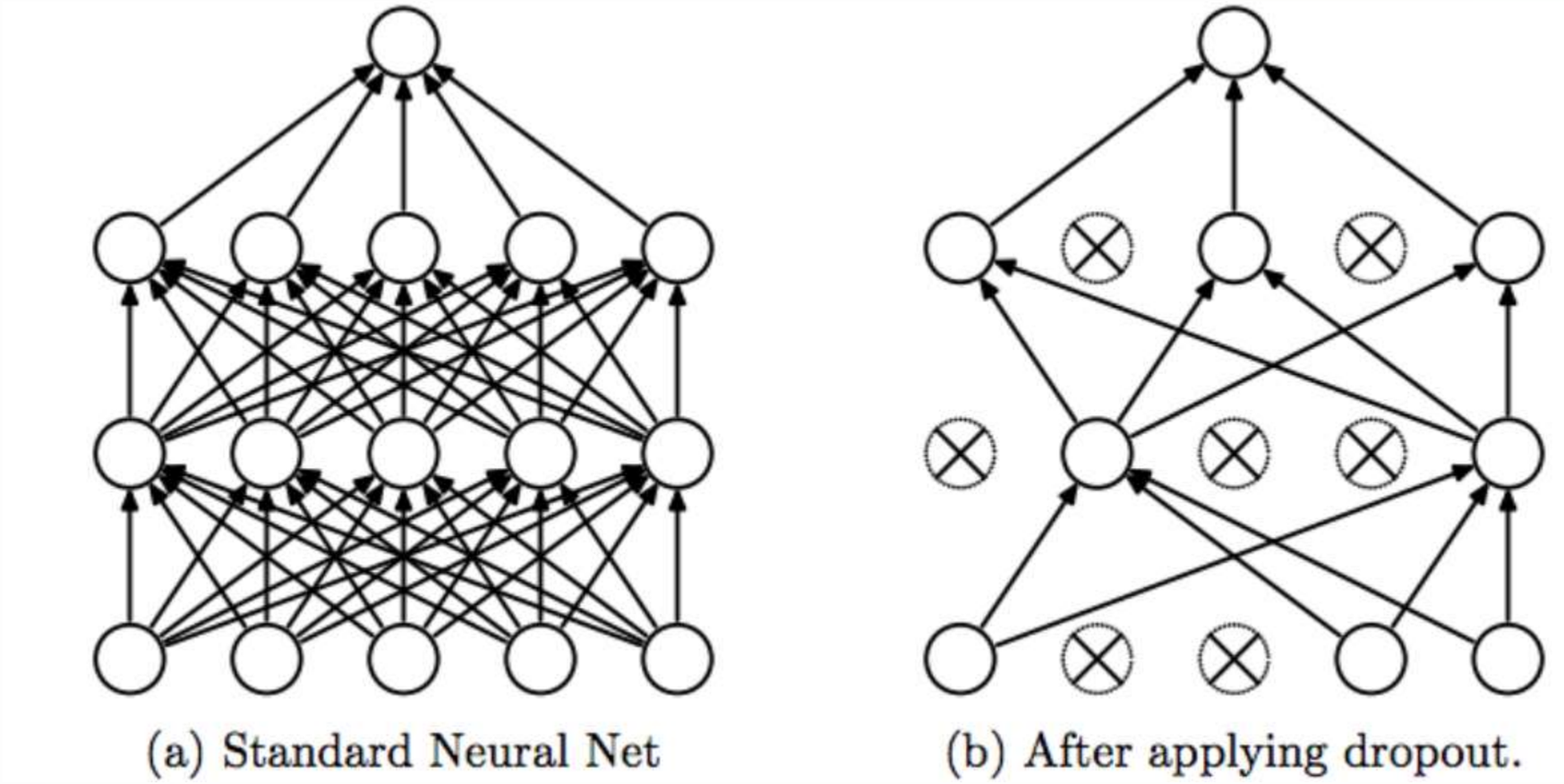
This leads to excellent performance on training data but poor generalization on unseen data.



Training accuracy rises steadily, but test accuracy peaks and then drops

Dropout

Dropout is a **regularization technique** used during the training of neural networks to prevent **overfitting**.



(a) Standard Neural Net

(b) After applying dropout.

How Dropout Works:

- **Choose a dropout rate.**
  Eg : p = 0.5 means 50% of neurons are dropped during training.

- **During each forward pass in training:**
  Random select neurons to drop
  Set their output to zero for that pass

- **During backpropagation** :
  Only the active neurons weight are updated

- **During testing**:
  No neurons are dropped.

Simple Dropout Example

Let's see dropout in action. Imagine a layer with neuron activations before dropout.

Before Dropout

```
Layer Output: [2.5, 1.8, 3.1, 0.9, 2.2]
```

After Dropout (p=0.5)

```
Dropout Mask: [1, 0, 1, 0, 1]
Layer Output: [2.5, 0.0, 3.1, 0.0, 2.2](Neurons at
index 1 and 3 were dropped out)
```