

Solution

02 January 2025

16:50

60000

1. Adding more data ✓✗
2. Reducing the complexity of NN architecture
3. Regularization
4. Dropouts
5. Data Augmentation
6. Batch Normalization
7. Early Stopping

Dropout - OneNote

File Home Insert Draw History Review View Help

Sticky Notes Share

Dropout

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

$p = 0.5$ 50%
0.3

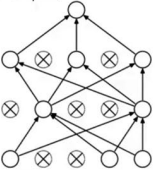
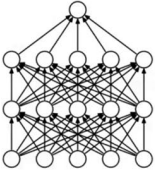
each forward pass

(784) (128) (64) (10)

simplify

(a) Standard Neural Net

(b) After applying dropout.



Batch Norm

03 January 2025 18:40

- Applied to Hidden Layers:
 - Typically applied to the hidden layers of a neural network, but not to the output layer.
- Applied After Linear Layers and Before Activation Functions:
 - Normalizes the output of the preceding layer (e.g., after nn.Linear) and is usually followed by an activation function (e.g., ReLU).
- Normalizes Activations:
 - Computes the mean and variance of the activations within a mini-batch and uses these statistics to normalize the activations.
- Includes Learnable Parameters:
 - Introduces two learnable parameters, gamma (scaling) and beta (shifting), which allow the network to adjust the normalized outputs.
- Improves Training Stability:
 - Reduces internal covariate shift, stabilizing the training process and allowing the use of higher learning rates.
- Regularization Effect:
 - Introduces some regularization because the statistics are computed over a

Batch Normalisation is used to ensure that output normalisation and data distribution is same

Applied after linear layer but before Activation func

L2 Regularization

03 January 2025 18:58

Applied to Model Weights:

- Regularization is applied to the weights of the model to penalize large values and encourage smaller, more generalizable weights.

Introduced via Loss Function or Optimizer:

- Adds a penalty term $\lambda \sum w_i^2$ to the loss function in L2 regularization.

$$\text{Loss}_{\text{reg}} = \text{Loss}_{\text{original}} + \lambda \sum w_i^2$$

$\lambda (w_1^2 + w_2^2 + w_3^2 + w_4^2)$

- In weight decay, directly modifies the gradient update rule to include λw , effectively shrinking weights during training.

$$w \leftarrow w - \eta (\nabla \text{Loss} + \lambda w)$$

weight decay

Penalizes Large Weights:

- Encourages the network to distribute learning across multiple parameters, avoiding reliance on a few large weights.

Reduces Overfitting:

- Helps the model generalize better to unseen data by discouraging overly complex

Handwritten notes: optimization, mse, logloss, weights/biases