**Key Terminologies and Parameters**

1. **Neural Network**

A neural network is a series of algorithms that attempt to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. For MNIST, we'll use a simple neural network with an input layer, one or more hidden layers, and an output layer.

1. **Neuron**

A neuron is a basic unit of a neural network that receives input, processes it, and passes it to the next layer. It mimics a biological neuron.

1. **Layer**

**Input Layer**: The first layer of the neural network that receives the input data. For MNIST, it would have 784 neurons (28x28 pixels).

**Hidden Layer**: Intermediate layers between input and output layers. They perform transformations on the input data to find patterns.

**Output Layer**: The final layer of the neural network that produces the output. For MNIST, it has 10 neurons (one for each digit).

1. **Convolutional Layer**

A layer in a Convolutional Neural Network (CNN) that applies convolutional operations to the input, capturing spatial hierarchies in the data.

1. **Convolutional Neural Network (CNN)**

A type of neural network particularly effective for image recognition tasks. It includes convolutional layers, pooling layers, and fully connected layers.

1. **Recurrent Neural Network (RNN)**

A type of neural network designed for sequential data. It uses its internal state (memory) to process sequences of inputs.

1. **Activation Function**

Functions applied to neurons' outputs. They introduce non-linearity into the network, helping it learn complex patterns. Examples include ReLU, Sigmoid, Tanh, and Softmax.

1. **ReLU (Rectified Linear Unit)**

An activation function that outputs zero if the input is negative, and the input itself if positive. Helps mitigate the vanishing gradient problem.

1. **Sigmoid**

An activation function that maps input values to a range between 0 and 1. Often used in binary classification problems.

1. **Tanh (Hyperbolic Tangent)**

An activation function that maps input values to a range between -1 and 1. It’s zero-centered, which helps in centering the data.

1. **Softmax**

An activation function often used in the output layer of a neural network for multi-class classification. It outputs a probability distribution.

1. **Forward Propagation**

The process by which input data passes through the network layers to produce an output.

1. **Backpropagation**

The process by which the network learns from the error by adjusting weights in the opposite direction of the gradient of the loss function.

1. **Loss Function**

A function that measures the difference between the actual output and the predicted output. Examples include Mean Squared Error, Cross-Entropy Loss.

1. **Cost Function**

Similar to the loss function but often refers to the overall error of the model. It’s the average of the loss function over the training dataset.

1. **Gradient Descent**

An optimization algorithm used to minimize the cost function by iteratively adjusting the model’s parameters.

1. **Learning Rate**

A hyperparameter that controls the step size of the gradient descent updates.

1. **Batch Size**

The number of training samples used in one forward/backward pass. Smaller batch sizes generally result in noisier updates but can lead to faster convergence.

1. **Epoch**

One complete pass through the entire training dataset.

1. **Overfitting**

When a model learns the training data too well, including noise and details, causing poor performance on new, unseen data.

1. **Underfitting**

When a model is too simple to capture the underlying patterns in the data, leading to poor performance even on the training data.

1. **Training Set**

The subset of data used to train the model.

1. **Validation Set**

The subset of data used to tune the model’s hyperparameters and to evaluate the model’s performance during training.

1. **Test Set**

The subset of data used to evaluate the model’s performance after training.

1. **Cross-Validation**

A technique for evaluating the model’s performance by dividing the dataset into multiple training and validation sets.

1. **Hyperparameters**

Parameters that are set before training the model, such as learning rate, batch size, and the number of epochs.

1. **Model Parameters**

Parameters learned during training, such as weights and biases.

1. **Regularization**

Techniques used to prevent overfitting, such as L1/L2 regularization, dropout.

1. **Dropout**

A regularization technique where randomly selected neurons are ignored during training, reducing overfitting.

1. **Weight Initialization**

The method used to initialize the weights of the neural network before training.

1. **Normalization**

The process of scaling input features to a specific range, typically [0, 1] or [-1, 1].

1. **Standardization**

The process of scaling input features so that they have zero mean and unit variance.

EXAAMPLE TEST DATA

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Image ID | Pixel1 | Pixel2 | … | Pixel699 | Pixel784 | label |
| 1 | 0 | 0 |  | 0 | 0 | 5 |
| 2 | 0 | 0 |  | 0 | 0 | 0 |
| 3 | 0 | 0 |  | 0 | 0 | 4 |
| 4 | 0 | 0 |  | 0 | 0 | 1 |

**Neural Network**: Our neural network will process the pixel values of each image to classify it as a digit from 0 to 9.

**Neuron**: Each pixel value will be input to a neuron in the input layer.

**Layer**: The network consists of multiple layers including input, hidden, and output layers.

**Input Layer**: The 784 pixel values are fed into the 784 neurons in the input layer.

**Hidden Layer**: These intermediate layers process the input data to detect features such as edges or corners.

**Output Layer**: The final layer will have 10 neurons, each representing a digit from 0 to 9.

**Convolutional Layer**: In a CNN, the convolutional layer would detect patterns like lines or textures in the images.

**Convolutional Neural Network (CNN)**: A CNN could be used for this task to improve accuracy by capturing spatial hierarchies in the images.

**Recurrent Neural Network (RNN)**: Not typically used for image data but more for sequential data like time series or text.

**Activation Function**: Each neuron uses an activation function to determine the output.

**ReLU**: Commonly used in hidden layers for its simplicity and effectiveness.

**Sigmoid**: Might be used in the output layer for binary classification, but not in this multi-class scenario.

**Tanh**: Another activation function that could be used in hidden layers.

**Softmax**: Used in the output layer to convert the raw scores into probabilities for each class (digit).

**Forward Propagation**: The process where the pixel values move through the network to produce an output.

**Backpropagation**: The process of adjusting weights based on the error of the output.

**Loss Function**: Measures the difference between the predicted and actual labels.

**Cost Function**: The average loss over the entire training set.

**Gradient Descent**: Optimization method used to minimize the cost function.

**Learning Rate**: Controls the size of the steps taken during gradient descent.

**Batch Size**: Number of images processed before updating the weights.

**Epoch**: One complete pass through the entire training dataset.

**Overfitting**: If our model performs well on training data but poorly on unseen data.

**Underfitting**: If our model performs poorly even on training data.

**Training Set**: The data used to train the model.

**Validation Set**: Used to tune hyperparameters and validate the model during training.

**Test Set**: Used to evaluate the model’s performance after training.

**Cross-Validation**: Dividing the dataset into multiple training and validation sets to evaluate the model.

**Hyperparameters**: Predefined parameters like learning rate, batch size.

**Model Parameters**: Parameters learned during training such as weights and biases.

**Regularization**: Techniques like dropout used to prevent overfitting.

**Dropout**: Randomly ignoring certain neurons during training to prevent overfitting.

**Weight Initialization**: Properly initializing weights to avoid issues like vanishing or exploding gradients.

**Normalization**: Scaling pixel values to a specific range.

**Standardization**: Ensuring pixel values have zero mean and unit variance.