ASSIGNMENT - 6

1. What is the difference between TRAINABLE and NON-TRAINABLE PARAMETERS?

Ans: Trainable vs. Non-Trainable Parameters:

Trainable Parameters: These are the core elements that get adjusted during the training process. They are typically the weights and biases in neural networks. The training algorithm (like backpropagation) analyzes the errors between predictions and actual outputs, then adjusts these trainable parameters to minimize the error.

Non-Trainable Parameters: These values remain fixed throughout training. They might include:

Hyperparameters: These are settings defined before training, like the number of hidden layers or learning rate. You experiment with different hyperparameter values to find the best configuration for your model.

Predefined Values: Some layers might have inherent non-trainable parameters, like the pooling size in pooling layers (e.g., max pooling).

Fixed Embeddings: In natural language processing, pre-trained word embeddings might be frozen (non-trainable) while training the rest of the network.

2. In the CNN architecture, where does the DROPOUT LAYER go?

Ans: In Convolutional Neural Networks (CNNs), dropout layers are typically inserted between convolutional layers (after ReLU or similar activation functions). This helps prevent overfitting by randomly dropping a certain percentage of activations during training. Here's a common placement:

Input -> Conv Layer -> ReLU -> Dropout -> ... (repeat) -> Fully Connected Layer -> Output

3. What is the optimal number of hidden layers to stack?

Ans: There's no one-size-fits-all answer for the optimal number of hidden layers. It depends on the complexity of your problem and the size of your dataset. Here are some general guidelines:

Simpler problems: Might only need 1-2 hidden layers.

More complex problems: May require 3-5 hidden layers, or even deeper architectures (experimentation is key).

Too many layers: Can lead to overfitting.

Too few layers: May not capture enough complexity in the data.

Start with a reasonable number of layers (e.g., 2-3) and experiment by adding or removing layers to see how it affects performance.

4. In each layer, how many secret units or filters should there be?

Ans: The number of filters/neurons per layer also depends on the specific problem and dataset size. Here are some factors to consider:

Input size: The first layer's filter size should be compatible with the input data dimensions.

Hidden layers: The number of neurons can vary, but it's often beneficial to have a similar or slightly decreasing number of neurons as you move towards the output layer.

Too many neurons: Can lead to overfitting.

Too few neurons: May limit the model's ability to learn complex patterns.

5. What should your initial learning rate be?

Ans: The initial learning rate is crucial for training convergence. A good starting point can be:

0.001: A common default value for many optimizers.

0.01: Might be suitable for smaller datasets or simpler problems.

However, the optimal rate depends on the specific problem and network architecture. You can use techniques like learning rate scheduling or monitoring validation loss to adjust the learning rate during training.

6. What do you do with the activation function?

Ans: Activation functions introduce non-linearity into the network, allowing it to learn complex relationships. Common choices include:

ReLU (Rectified Linear Unit): Popular choice for its efficiency and vanishing gradient mitigation.

Sigmoid/TanH: Traditionally used but can suffer from vanishing gradients in deep networks.

Leaky ReLU: Variant of ReLU that allows a small non-zero gradient for negative inputs.

The best activation function depends on the problem and network architecture.

7. What is NORMALIZATION OF DATA?

Ans: Normalization in neural networks refers to the process of scaling your input data to a specific range. This is important for several reasons:

Improved Training Speed: By having data on a similar scale, the training algorithm (e.g., backpropagation) can converge faster and more efficiently.

Stability for Activation Functions: Some activation functions, like sigmoid and tanh, have more favorable behavior within a specific input range (often between 0 and 1 or -1 and 1). Normalization helps ensure your data falls within that range.

Equal Treatment of Features: If your features have vastly different scales, the network might prioritize features with larger values during training. Normalization helps prevent this by giving all features a more equal footing.

Common normalization techniques include:

Min-Max Scaling: Scales data to a range between 0 and 1 (or -1 and 1).

Z-score normalization: Standardizes data by subtracting the mean and dividing by the standard deviation.

8. What is IMAGE AUGMENTATION and how does it work?

Ans: Image augmentation is a technique used to artificially expand your training dataset by creating variations of existing images. This helps the model become more robust to variations in real-world data, improving its generalization ability. Here are some common image augmentation methods:

Random Cropping: Taking random crops of an image to focus on different parts and simulate zooming.

Flipping (Horizontal/Vertical): Flipping images horizontally or vertically to introduce variations in orientation.

Rotation: Rotating images by small random angles to account for different viewing angles.

Color Jitter: Slightly changing the brightness, contrast, saturation, or hue of images.

Noise Injection: Adding small amounts of random noise to images to simulate imperfections or noise in real-world data.

9. What is DECLINE IN LEARNING RATE?

Ans: The learning rate is a hyperparameter that controls how much the network's weights and biases are adjusted during training. A high learning rate can lead to faster learning but also cause the model to jump past the optimal solution (oscillate and never converge). Conversely, a low learning rate can lead to slow convergence or getting stuck in local minima.

Decline in learning rate (learning rate scheduling) is a technique where you gradually decrease the learning rate throughout training. This allows the network to make larger adjustments initially for faster learning and then fine-tune the weights with smaller adjustments as it gets closer to the optimal solution.

10. What does EARLY STOPPING CRITERIA mean?

Ans: Early stopping is a regularization technique to prevent overfitting. During training, you monitor the model's performance on a validation set (data not used for training). If the validation loss (error) starts to increase even though the training loss keeps decreasing, it's an indication of overfitting. Early stopping allows you to stop training before this happens, saving computational resources and achieving better generalization performance.

Here's a common early stopping approach:

* Define a patience value (e.g., number of epochs without improvement on validation loss).
* Train the model.
* If validation loss doesn't improve for patience epochs, stop training.

This approach prevents the model from memorizing the training data and helps it generalize better to unseen data.