ASSIGNMENT - 3

1. After each stride-2 conv, why do we double the number of filters?

Ans: When you use a stride-2 convolution, the output activation map's spatial dimensions (width and height) are halved compared to the input. To compensate for this reduction and maintain the network's capacity to learn complex features, we often double the number of filters in the following convolutional layer.

Here's the reasoning:

Stride-2 convolution reduces spatial resolution, leading to fewer activation locations in the output.

Doubling the number of filters increases the number of feature channels, allowing the network to learn a more diverse set of features from the reduced number of spatial locations.

This maintains a similar level of overall model complexity and computational cost compared to using the same number of filters with stride-1 convolution.

2. Why In a simple CNN for MNIST (handwritten digit recognition), using a larger kernel size (e.g., 5x5) in the first convolutional layer can be beneficial for a few reasons:

Capturing Spatial Information: MNIST images are relatively small (28x28 pixels). A larger kernel can capture a wider range of contextual information within an image patch, potentially aiding in recognizing digits.

Learning Lower-Level Features: Early convolutional layers often extract basic features like edges, lines, and shapes. A larger kernel might be more effective at learning these low-level features directly from the input image.

Reducing Number of Parameters: Although a larger kernel has more parameters, you might be able to compensate by using fewer filters compared to a smaller kernel with more filters. This can sometimes lead to a more efficient model, especially for smaller datasets like MNIST.do we use a larger kernel with MNIST (with simple cnn) in the first conv?

Ans:

3. What data is saved by ActivationStats for each layer?

Ans: the ActivationStats object might store information about the activation values during training, such as:

Mean: The average activation value across all elements in the activation map for a given layer.

Standard Deviation: The spread of activation values around the mean, indicating how diverse the activations are.

Minimum and Maximum Values: The lowest and highest activation values encountered.

This information can be used for various purposes, including:

Normalization: Normalizing activations (e.g., using techniques like batch normalization) can improve training stability. Statistics about activations might be used to calculate normalization parameters.

Diagnostics and Debugging: Analyzing activation statistics can help identify potential issues like vanishing/exploding gradients.

The specific data saved in ActivationStats might vary depending on the framework.

4. How do we get a learner’s callback after they’ve completed training?

Ans: The exact way to get a learner's callback after training completion depends on the deep learning library you're using. Here are some general strategies:

Built-in Callbacks: Many frameworks offer built-in callbacks that are automatically triggered after training finishes. Look for functions or classes related to training completion events.

Custom Callbacks: You can create a custom callback function that performs the actions you want after training is done. This function can be registered with the learner object before training starts.

For example, in PyTorch Lightning, you could use the Trainer class's callbacks argument to register a custom callback.

5. What are the drawbacks of activations above zero?

Ans: Activations above zero can have some drawbacks in CNNs:

Internal Covariate Shift: Activations that are always positive can lead to an internal covariate shift within the network, where the distribution of activations changes across layers during training. This can make it harder for the network to learn effectively.

Limited Expressive Power: Activations restricted to positive values can limit the network's ability to express certain types of features, especially when features involve both positive and negative components.

To address these issues, activation functions like ReLU (Rectified Linear Unit) are commonly used. ReLU sets negative activations to zero, introducing non-linearity while still allowing positive activations.

6.Draw up the benefits and drawbacks of practicing in larger batches?

Ans: Benefits:

Faster Training: Larger batches can lead to faster training convergence due to more efficient utilization of computational resources.

Smoother Gradients: Averaging gradients over a larger batch can result in smoother gradients, potentially leading to better convergence and avoiding local minima.

Drawbacks:

Memory Requirements: Training with large batches requires more GPU or system memory to store the input data and gradients.

Variance Reduction: Overly large batches may reduce variance in the gradients, leading to potentially suboptimal solutions.

7. Why should we avoid starting training with a high learning rate?

Ans: There are several reasons to avoid starting training with a high learning rate:

Overshooting the Minimum: Imagine the loss function (error) as a valley you want your model to navigate towards the lowest point (minimum). A high learning rate can cause the model to take large steps, potentially overshooting the minimum and getting stuck on a suboptimal solution.

Unstable Training: High learning rates can lead to large updates to the model's weights during backpropagation. These large updates can cause the loss to oscillate wildly, making training unstable and potentially preventing convergence.

Ignoring Fine Details: High learning rates might focus on making large adjustments initially, potentially overlooking subtle features or local minima that could lead to better performance.

8. What are the pros of studying with a high rate of learning?

Ans: In some cases, a high learning rate can be beneficial, but it should be used cautiously and often as part of a learning rate schedule that reduces it over time. Here are some potential advantages:

Faster Initial Progress: A high learning rate can help the model make rapid progress early in training, especially when dealing with simple problems or large datasets.

Escaping Local Minima: In some cases, a high learning rate can help the model escape from shallow local minima in the loss function and explore a wider range of solutions.

9. Why do we want to end the training with a low learning rate?

Ans: As training progresses, the model gets closer to the optimal solution. Here's why a low learning rate is often preferred towards the end:

Fine-Tuning: A low learning rate allows for more precise adjustments to the weights, helping the model refine its solution and potentially achieve better accuracy.

Stability: A low learning rate prevents large oscillations in the loss function, promoting smooth convergence and avoiding getting stuck in suboptimal regions.

Generalization: By carefully adjusting weights with smaller steps, a low learning rate can help the model generalize better to unseen data.