ASSIGNMENT - 10

1. Why don’t we start all of the weights with zeros?

Ans: Initializing all weights in a CNN to zero is a bad idea for two main reasons:

Symmetry: If all weights are zero, all neurons in a layer will perform the same calculation, essentially becoming redundant. This prevents the network from learning diverse features.

Vanishing gradients: With zero weights, the gradient during backpropagation might also be zero, leading to no updates and preventing the network from learning at all (especially for activation functions like tanh or ReLU where the derivative at zero is zero).

2. Why is it beneficial to start weights with a mean zero distribution?

Ans: Initializing weights with a mean of zero helps ensure gradients flow efficiently during backpropagation. Here's why:

Balanced activations: A mean of zero prevents any bias towards positive or negative activations in the early layers. This allows the network to learn both positive and negative features in the data.

Efficient gradient updates: With a mean of zero, gradients tend to have a more balanced distribution, facilitating smoother learning through backpropagation.

3. What is dilated convolution, and how does it work?

Ans: Dilated convolution (also known as atrous convolution) is a technique used in CNNs to capture long-range dependencies in the input while maintaining the resolution of the feature maps. It works by introducing a dilation rate (often denoted as d) between filter elements:

* The filter itself remains the same size.
* During convolution, the filter is "dilated" by skipping d - 1 elements between filter taps in each direction.

This allows the receptive field (the area of the input the filter "sees") to grow exponentially without increasing the filter size itself. This is helpful for tasks like image segmentation where capturing context over a larger area is important.

4. What is TRANSPOSED CONVOLUTION, and how does it work?

Ans: Transposed convolution (sometimes called deconvolution, although it's not mathematically a true deconvolution) is an operation that learns to upsample a feature map. It's useful in tasks like image generation or dense prediction where you want to increase the spatial resolution of the output.

Here's how it works:

* A transposed convolution filter has learned weights.
* The filter is flipped (rotated by 180 degrees) compared to a standard convolution.
* The filter is applied to the input feature map, but instead of sliding across the input, it produces an output that is larger than the input.
* The specific way the output size is calculated depends on the padding and stride used in the transposed convolution layer.

5.Explain Separable convolution

Ans: Separable convolution is an optimization technique that reduces the computational cost of standard convolution. It decomposes a standard convolution into two depthwise and pointwise convolutions:

Depthwise convolution: Applies a single filter per input channel, capturing spatial information within each channel.

Pointwise convolution (1x1 convolution): Combines the outputs from the depthwise convolution using 1x1 filters, effectively learning linear combinations of the spatially filtered outputs.

This reduces the number of parameters compared to standard convolution, especially for large filter sizes and many input channels.

6.What is depthwise convolution, and how does it work?

Ans: Depthwise convolution is one part of a separable convolution. It applies a single filter per input channel of the feature map. This captures spatial information within each channel independently.

Here's a breakdown:

* The filter has the same height and width as the standard convolution filter, but its depth is equal to 1 (one filter per input channel).
* During convolution, each filter is applied only to its corresponding channel in the input feature map.

7.What is Depthwise separable convolution, and how does it work?

Ans: Depthwise separable convolution combines the concepts of depthwise and pointwise convolutions. Here's the process:

Depthwise convolution: Apply separate filters to each input channel, capturing spatial information.

Pointwise convolution (1x1 convolution): Combine the outputs from the depthwise convolution using 1x1 filters, effectively learning linear combinations of the spatially filtered outputs.

8.Capsule networks are what they sound like.

Ans: This is a partially accurate statement. Capsule networks are a type of artificial neural network architecture inspired by the hierarchical structure of the visual cortex. They aim to address limitations of standard CNNs, particularly in representing object

9. Why is POOLING such an important operation in CNNs?

Ans: Pooling layers play a crucial role in CNNs for several reasons:

Dimensionality reduction: Pooling downsamples the feature maps, reducing their height and width. This:

Makes the network more computationally efficient by reducing the number of parameters and calculations required in subsequent layers.

Helps prevent overfitting by reducing the amount of data the network needs to learn from.

Translation invariance: Pooling introduces a degree of invariance to small shifts in the input image. This means the network becomes less sensitive to minor variations in the position of features within the image. For example, a cat's eye might be slightly higher in one image compared to another. Pooling helps the network still recognize the presence of an eye even with this small shift.

Focus on prominent features: Pooling often captures the most dominant feature within a local region (e.g., max pooling takes the maximum value). This can help the network focus on the most important aspects of the input for further processing.

10. What are receptive fields and how do they work?

Ans: In CNNs, the receptive field of a neuron in a specific layer refers to the area of the input image that contributes to its activation. It's essentially the "window" a neuron "sees" through its filter.

Here's how receptive fields work:

* The size of the receptive field depends on the filter size and the number of preceding convolutional layers.
* In the first convolutional layer, the receptive field is simply the size of the filter.
* As you add more convolutional layers, the receptive field grows. Each layer "sees" not only the direct input but also the combined features learned by the previous layer's filters within its receptive field.
* This expanding receptive field allows the network to learn increasingly complex features by combining information from larger portions of the input image.