**COMPUTER VISION ASSIGNMENT\_2**

**1.Explain convolutional neural network, and how does it work?**

A Convolutional Neural Network (CNN) is a type of deep learning artificial neural network used primarily for image and video recognition tasks. It works by applying multiple filters to the input data to extract features or information. The filters move across the input data, computing dot products between the entries of the filter and the input, producing a scalar activation map (also called feature map). This process is repeated several times to extract multiple features from the input data, and the resulting feature maps are then processed by pooling layers, which reduce their dimensions and increase their invariance to small translations in the input data. The final output of the CNN is obtained by applying one or more fully connected layers on the pooled feature maps, producing a prediction for the input data.

**2. How does refactoring parts of your neural network definition favor you?**

Refactoring parts of a neural network definition, or modifying its architecture, can lead to improved performance and accuracy in the tasks it is designed to solve. This can be achieved by:

Introducing new layers to extract better features from the input data, or removing redundant layers to simplify the network.

Changing the hyperparameters of the network, such as the number of filters, filter sizes, pooling sizes, etc., to improve its ability to capture important patterns in the data.

Adding regularization techniques, such as dropout, to prevent overfitting to the training data and improve generalization to new data.

Introducing new loss functions or optimizers that better align with the desired performance metrics for the task.

Refactoring a neural network requires experimentation, fine-tuning and careful evaluation, but it can lead to better results and make the network more suited to the task at hand.

**3. What does it mean to flatten? Is it necessary to include it in the MNIST CNN? What is the reason for this?**

"Flattening" refers to the process of converting a high-dimensional tensor, such as a 2D feature map from a Convolutional Layer, into a 1D vector. This is done so that the output from the convolutional layers can be fed into fully connected (dense) layers, which expect 1D vectors as inputs.

It is necessary to include the flattening step in the MNIST CNN, as the goal is to classify the images into 10 classes (0-9). This requires the feature maps generated by the convolutional layers to be processed by fully connected layers, which can learn to make the final prediction based on the extracted features.

Flattening is an important step in many computer vision tasks and is commonly used in Convolutional Neural Networks to prepare the feature maps for classification. Without it, the information in the high-dimensional feature maps would not be usable by the dense layers.

**4. What exactly does NCHW stand for?**

NCHW stands for "batch size, channels, height, width." It is a data layout convention used to specify the shape of a 4-dimensional tensor in deep learning, especially in computer vision tasks.

In NCHW, the first dimension represents the batch size, or the number of samples in a batch of data. The second dimension represents the number of channels in each sample, such as the number of color channels in an image (e.g., red, green, blue). The third and fourth dimensions represent the height and width of each sample, respectively.

Using NCHW as the data layout can improve performance and memory utilization in certain deep learning libraries and hardware accelerators, as it allows for optimized use of memory and computation resources.

**5. Why are there 7\*7\*(1168-16) multiplications in the MNIST CNN’s third layer?**

The number of multiplications in a convolutional neural network (CNN) layer depends on various factors, including the number of filters, the size of the filters, the number of input channels, and the size of the input feature map. The specific number of multiplications in the third layer of a MNIST CNN you mentioned (7 x 7 x (1168 - 16)) can be calculated based on the following assumptions:

The third layer is a convolutional layer with 7x7 filters and 16 output channels.

The input to the third layer is a feature map with 1168 channels.

To perform one convolution operation in this layer, each of the 7x7 filters would be multiplied by each of the 1168 input channels and then added up. This results in a single output value in the feature map produced by the layer. Doing this operation for all 16 filters results in 16 output values in the feature map. The total number of multiplications required to produce a single output value in the feature map is 7x7x1168. To produce the entire feature map with 16 output channels, the number of multiplications required would be 7x7x1168x16, or 7 x 7 x (1168 - 16).

**6.Explain definition of receptive field?**

The receptive field of a neuron in a convolutional neural network (CNN) refers to the portion of the input image that the neuron is "sensitive" to and uses to produce its output. In other words, the receptive field determines what information a neuron in the network uses to make its prediction.

For example, in an image classification task, a neuron in an early layer of the CNN might have a receptive field of a small, local region of the input image, such as a 3x3 or 5x5 patch. As the network processes the image, the receptive field of neurons in later layers grows to cover larger portions of the input image, allowing the network to consider more global information in its predictions.

The size and shape of the receptive field is determined by the architecture of the network, including the number of convolutional filters and their size, as well as the size of the stride and padding used in the convolution operations. Understanding the receptive field of different neurons in a CNN can help us understand how the network processes information and makes predictions, and can also inform the design of new CNN architectures

**7. What is the scale of an activation’s receptive field after two stride-2 convolutions? What is the reason for this?**

The scale of an activation's receptive field after two stride-2 convolutions is halved. The reason for this is that the stride in a convolutional layer determines the step size with which the filter moves across the input feature map, and a stride of 2 results in the filter skipping over every other element in the input feature map. As a result, the spatial dimensions of the output feature map are reduced by a factor of 2 for each stride-2 convolution applied.

For example, consider an input feature map with spatial dimensions of 28x28. After one stride-2 convolution, the spatial dimensions of the output feature map would be reduced to 14x14. After a second stride-2 convolution, the spatial dimensions of the output feature map would be further reduced to 7x7. As a result, the receptive field of an activation in the 7x7 feature map covers only a quarter of the spatial extent of the receptive field of an activation in the original 28x28 feature map.

Note that this reduction in the scale of the receptive field is trade-off with increased abstraction, as the network is able to combine information from larger portions of the input image to make its predictions. This abstraction can be useful in practice, as it allows the network to recognize larger-scale patterns in the input data and make predictions that are more invariant to changes in the input.

**8. What is the tensor representation of a color image?**

A color image can be represented as a tensor in computer vision. A tensor is a mathematical object that can be thought of as a multi-dimensional array of numbers. In the case of a color image, the tensor would have three dimensions, representing the height, width, and color channels of the image.

Typically, the color channels of an image are represented as Red, Green, and Blue (RGB) channels. The values of these channels range from 0 to 255, and they define the color and intensity of each pixel in the image.

For example, a color image of size 256x256 pixels can be represented as a tensor of shape (256, 256, 3), where the first two dimensions represent the height and width of the image and the third dimension corresponds to the RGB channels. The value of each element in the tensor represents the intensity of one of the RGB channels for a particular pixel in the image.

In some cases, color images may also be represented as grayscale images or in different color spaces, such as the Hue-Saturation-Value (HSV) color space, but the RGB representation is the most commonly used representation for color images in computer vision and deep learning.

**9. How does a color input interact with a convolution?**

In a convolutional neural network (CNN), a color input image is processed through convolutional layers to produce feature maps, which are then processed by subsequent layers to make predictions.

A convolution operation involves sliding a filter (also known as a kernel) across the input image and computing dot products between the values in the filter and the values in the input image. The result of the convolution is a new feature map, where each activation in the feature map corresponds to the result of one dot product operation between the filter and the input image.

When the input image is a color image, it has multiple color channels (e.g., Red, Green, and Blue (RGB) channels). For each convolution operation, a separate filter is applied to each color channel of the input image, resulting in multiple feature maps. These feature maps are then concatenated to produce a single, multi-channel feature map that represents the output of the convolutional layer.

In this way, the convolutional layer is able to process information from each color channel of the input image independently, allowing it to capture information about the color distribution and texture in the image. This information can then be used by subsequent layers to make predictions, such as in an image classification task.