Experiment No: 08

Aim: To define the CNN to predict the knowledge from image classification Visualizing the learned CNN Model

Theory:

Convolutional Neural Networks (CNNs) are a class of deep learning models primarily designed for processing and analyzing grid-like data, with a strong emphasis on their application in computer vision tasks. CNNs have proven to be exceptionally effective in image classification, object detection, and various other visual recognition tasks. Here are the key concepts and theory behind CNNs:

Convolutional Layers:

Convolution Operation: CNNs use convolutional layers to process input data. Convolution is a mathematical operation that involves a small filter or kernel sliding over the input data to perform element-wise multiplications and aggregations, allowing the network to detect local patterns.

Feature Maps: The output of a convolutional operation is called a feature map or activation map. Each feature map corresponds to a specific filter and highlights a particular pattern or feature in the input data.

Pooling Layers:

Pooling Operation: Pooling layers reduce the spatial dimensions of the feature maps while retaining the most essential information. Common pooling operations include max pooling and average pooling.

Downsampling: Pooling helps decrease the computational complexity and control overfitting by reducing the number of parameters in the network.

Activation Functions:

CNNs typically use non-linear activation functions like ReLU (Rectified Linear Unit) to introduce non-linearity into the model. ReLU is preferred due to its simplicity and effectiveness.

Convolutional Neural Network Architecture:

CNNs consist of multiple convolutional layers, interspersed with activation functions and pooling layers. The final layers are typically fully connected layers.

The first layers capture low-level features like edges and textures, while deeper layers capture higher-level features and representations.

CNN architectures often involve skip connections, batch normalization, and other techniques to improve training and performance.

Weight Sharing:

One of the fundamental principles of CNNs is weight sharing. Each filter in a convolutional layer is shared across the entire input space. This sharing of weights allows the network to learn to recognize the same features in different parts of the input, making CNNs highly efficient.

Striding:

Convolutional layers can have a "stride" parameter that defines the step size at which the filter moves across the input data. Striding can reduce the spatial dimensions of the feature maps.

Padding:

Padding involves adding extra pixels to the input data before convolution. Padding helps preserve the spatial dimensions of feature maps, which is especially important in deeper layers.

Transfer Learning:

CNNs are often pre-trained on large datasets like ImageNet and then fine-tuned for specific tasks. This approach, known as transfer learning, leverages the learned features from the large dataset to boost performance on smaller, task-specific datasets.

Applications:

CNNs are widely used in image classification, object detection, image segmentation, facial recognition, medical image analysis, and more. They have also found applications beyond computer vision, such as in natural language processing and speech recognition.

Training: CNNs are trained using stochastic gradient descent (SGD) or its variants. Backpropagation is employed to update the network's weights and biases based on the gradient of the loss function.

Convolutional Neural Networks have revolutionized computer vision and significantly improved the state-of-the-art in various visual recognition tasks, making them a cornerstone of modern deep learning. Their ability to automatically extract hierarchical features from images has made them indispensable in many real-world applications.

Conclusion:

A simple CNN was built to classify images as either horizontal or vertical. The trained model learned to distinguish between the two orientations, achieving accuracy on both the training and validation datasets. The model's architecture and learned features can be visualized to better understand its performance.