Introduction to Convolutional Neural Networks(CNN)

Convolutional Neural Networks (CNNs) are similar to traditional Artificial Neural Networks (ANNs) in that they consist of neurons that self-optimize through learning. Each neuron receives an input, performs an operation (like a scalar product followed by a non-linear function), and contributes to the network's overall output score. The final layer includes loss functions for classification, and traditional ANN techniques apply to CNNs as well. The key difference is that CNNs are designed for pattern recognition in images, incorporating image-specific features that make them more suitable for image-focused tasks and reduce the model's parameters.

CNN Architechture

CNNs are designed specifically for image data. Unlike traditional ANNs, CNN neurons are organized in three dimensions: height, width, and depth. The depth here refers to the number of channels, not the number of layers. Neurons in a CNN layer connect only to a small region of the previous layer. For example, an input image with dimensions 64×64×3 (height, width, depth) is condensed into an output layer of 1×1×n, where n is the number of possible classes. This structure allows CNNs to efficiently handle image data.

Overall Architechture

CNNs are comprised of three types of layers. These are convolutional layers, pooling layers and fully-connected layers. When these layers are stacked, a CNN architecture has been formed.

The example CNN's basic functionality is divided into four key areas:

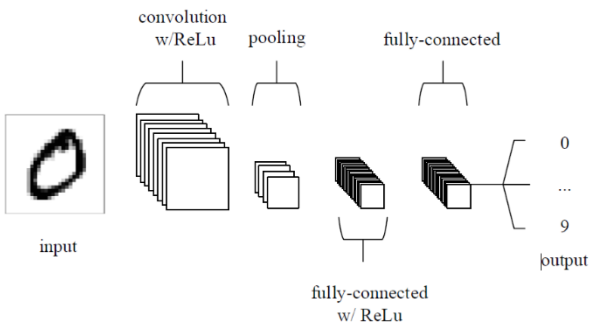
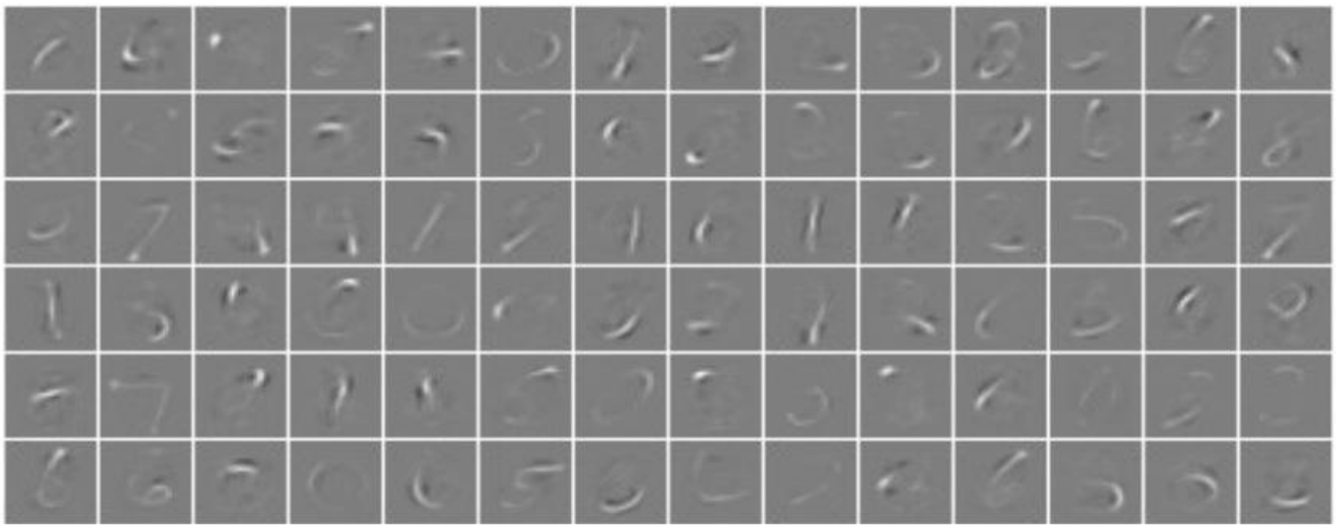


Fig. 2: An simple CNN architecture, comprised of just five layers

1. **Input Layer**: Holds the pixel values of the image.
2. **Convolutional Layer**: Computes the output of neurons connected to local regions of the input by calculating the scalar product of their weights and the input region, often applying an activation function like ReLU.
3. **Pooling Layer**: Performs downsampling along the spatial dimensions to reduce the number of parameters.
4. **Fully-Connected Layers**: Functions like standard ANN layers to produce class scores for classification, potentially using ReLU for better performance



If you look carefully, you can see that the network has successfully picked up on characteristics unique to specific numeric digits.

Convolutional Layer Summary:

1. **Function**: Uses learnable kernels to create 2D activation maps by convolving filters over the input.
2. **Receptive Field**: Neurons connect only to small input regions, reducing weights.
3. **Optimization**: Utilizes depth, stride, and zero-padding to control output size and reduce complexity.
4. **Formula**: Output size calculation:

$$\frac{(V-R)+2Z}{S}+1$$

   where V is input size, R is receptive field, Z is zero-padding, and S is stride.

5. **Parameter Sharing**: Shares weights across output to minimize parameters.

Pooling Layer Summary:

1. **Purpose**: Reduces dimensionality and computational complexity.
2. **Function**: Uses max-pooling (typically 2×2 kernels with a stride of 2) to scale down activation maps by 75% while maintaining depth.
3. **Methods**: Commonly uses non-overlapping (2×2) or overlapping (3×3 with stride 2) pooling.
4. **Variants**: Includes general pooling methods like average pooling and L1/L2 normalization, but max-pooling is most common.

Fully-Connected Layer Summary:

1. The fully-connected layer contains neurons of which are directly connected to the neurons in the two adjacent layers, without being connected to any layers within them.