Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a class of deep neural networks that are designed for processing grid-like data such as images. They use convolutional layers to automatically detect patterns like edges, textures and shapes in the data, thus making them very useful for applications like **object detection**, **medical imaging** and **facial recognition**. CNNs are widely used in **computer vision** applications due to their effectiveness in processing visual data.

How Convolutional Layers Works?

Convolution Neural Networks are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image), and height (i.e the channel as images generally have red, green, and blue channels). Now imagine taking a small patch of this image and running a small neural network, called a filter or kernel on it, with say, K outputs and representing them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different widths, heights, and depths.

Think of a convolutional layer as a team of feature detectives examining an image.

1. The Filter: The Detective's Magnifying Glass

The most important part is the Filter (or Kernel).

- What it is: It's a tiny, small, square magnifying glass, usually 3×3 pixels in size, filled with specific numbers (the "weights").
- What it does: Each filter is specialized to look for one specific simple feature. One filter might only care about finding vertical lines. Another might look for a 45° diagonal line. A third might look for a specific texture or color spot.

2. The Convolution: The Search-and-Record Process

The convolution is the process of using these filters to scan the image.

1. Sliding: The 3×3 filter is placed over a 3×3 patch of the input image, starting in the top-left corner.
2. Checking for a Match: The filter checks how well the 3×3 patch of the image matches the feature it's looking for (e.g., the vertical line). It does this by multiplying the numbers in the filter by the corresponding pixel numbers in the image patch, and then adding all those results up.

3. Recording the Result:
    - If the result is a high positive number, it means the filter found a strong match for its feature at that exact location (e.g., "Yes! There's a perfect vertical line here!").
    - If the result is near zero, it means no match.
4. Iteration: The filter then slides over (or "strides") to the next patch of the image and repeats the check.

3. The Feature Map: The Treasure Map

After the filter has scanned the entire image, the layer produces its output, which is called a Feature Map.

- What it is: A 2D grid that has the same (x, y) coordinates as the original image, but instead of showing colors, it shows the intensity of the feature detected by that specific filter.
- Example: If the "vertical line detector" filter was used, the resulting Feature Map would be bright wherever it found vertical lines and dark everywhere else.

If the convolutional layer has 32 filters, it produces 32 different Feature Maps, meaning it simultaneously detects 32 different features across the entire image.

4. Downstream Effects (What the Hyperparameters Do)

The way the filter slides and interacts with the edges is controlled by simple settings:

- Stride: This is how many steps the filter takes when it slides.
    - A Stride of 1 means it checks every single pixel location (high detail).
    - A Stride of 2 means it skips every other pixel (faster, smaller output).
- Padding: Since a 3×3 filter can't perfectly cover the edge pixels of an image, we sometimes add a border of zeros around the image edges before the scan. This is called "Same" Padding, and it ensures the output map has the same height and width as the input, preventing the image from shrinking too fast.
- Activation (ReLU): After the convolution sums up its match scores, the ReLU function ($\max(0, x)$) kicks in. It simply deletes any negative results (turning them to zero) and keeps all the positive results. This introduces non-linearity, which is necessary for the network to learn complex, non-straightforward patterns in the real world.

Mathematical Overview of Convolution

Now let's talk about a bit of mathematics that is involved in the whole convolution process.

- Convolution layers consist of a set of learnable filters (or kernels) having small widths and heights and the same depth as that of input volume (3 if the input layer is image input).
- For example, if we have to run convolution on an image with dimensions 34x34x3. The possible size of filters can be axax3, where 'a' can be anything like 3, 5, or 7 but smaller as compared to the image dimension.
- During the forward pass, we slide each filter across the whole input volume step by step where each step is called **stride** (which can have a value of 2, 3, or even 4 for high-dimensional images) and compute the dot product between the kernel weights and patch from input volume.
- As we slide our filters we'll get a 2-D output for each filter and we'll stack them together as a result, we'll get output volume having a depth equal to the number of filters. The network will learn all the filters.

Layers in CNN and its working

A Convolutional Neural Network (CNN) has several layers that process data in sequence: the **Input Layer** takes in data like an image, the **Convolutional Layer** uses filters to detect features, the **Pooling Layer** downsamples the data to reduce computation, the **Flatten Layer** converts the 2D feature maps into a 1D vector, and the **Fully Connected Layer** performs classification based on the high-level features, finally leading to the **Output Layer** that gives the final result.

Input Layer

- **What it is:** The first layer that receives the raw input data, such as an image represented as a 3D matrix (height x width x color channels).
- **How it works:** It simply passes the input data to the next layer for processing.

Convolutional Layer

- **What it is:** The core building block of a CNN that extracts features from the input.
- **How it works:** It uses a set of learnable filters (or kernels) that slide across the input image. Each filter is designed to detect a specific feature, like an edge or a texture. The output of this operation is a feature map, which highlights where the features were detected in the image.

Activation Layer

- **What it is:** A layer that introduces non-linearity into the network.
- **How it works:** It applies an activation function, such as the Rectified Linear Unit (ReLU), to the output of the convolutional layer. This allows the network to learn more complex patterns that a simple linear function cannot capture.

Pooling Layer

- **What it is:** A layer used to reduce the spatial dimensions (height and width) of the feature maps.
- **How it works:** It operates on small regions of the feature map and summarizes them. A common method is max pooling, which takes the maximum value from each region. This helps reduce computational complexity and makes the network more robust to small variations in the position of features (translation invariance).

Flatten Layer

- **What it is:** A layer that reshapes the 2D feature maps into a 1D vector.
- **How it works:** It converts the output from the final pooling or convolutional layer into a single, long vector. This is necessary to feed the data into the fully connected layers that follow.

Fully Connected Layer

- **What it is:** A standard neural network layer where every neuron is connected to every neuron in the next layer.
- **How it works:** It takes the high-level features from the flattened vector and uses them to perform the final classification or regression. These layers are responsible for combining the features to make a prediction.

Output Layer

- **What it is:** The final layer of the network.
- **How it works:** It produces the final output of the CNN. For classification tasks, it might use an activation function like softmax to output a probability for each possible class.

Other common layers

- **Normalization Layer:** Helps improve training stability and speed up convergence.
- **Dropout Layer:** Randomly "drops" a percentage of neurons during training to prevent the network from becoming too reliant on any single neuron, which helps prevent overfitting.

Layers Used to Build ConvNets

A complete Convolution Neural Networks architecture is also known as covnets. A covnets is a sequence of layers, and every layer transforms one volume to another through a differentiable function.

Let's take an example by running a covnets on of image of dimension 32 x 32 x 3.

- **Input Layers:** It's the layer in which we give input to our model. In CNN, Generally, the input will be an image or a sequence of images. This layer holds the raw input of the image with width 32, height 32, and depth 3.
- **Convolutional Layers:** This is the layer, which is used to extract the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are smaller matrices usually 2x2, 3x3, or 5x5 shape. it slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch. The output of this layer is referred as feature maps. Suppose we use a total of 12 filters for this layer we'll get an output volume of dimension 32 x 32 x 12.
- **Activation Layer**: By adding an activation function to the output of the preceding layer, activation layers add nonlinearity to the network. it will apply an element-wise activation function to the output of the convolution layer. Some common activation functions are **RELU**: max(0, x), **Tanh**, **Leaky RELU**, etc. The volume remains unchanged hence output volume will have dimensions 32 x 32 x 12.
- **Pooling layer**: This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are **max pooling** and **average pooling**. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.
- **Flattening:** The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.
- **Fully Connected Layers:** It takes the input from the previous layer and computes the final classification or regression task.

References

GeeksforGeeks. (2025, July 12). *Introduction to Convolution Neural Network*. GeeksforGeeks. https://www.geeksforgeeks.org/machine-learning/introduction-convolution-neural-network/