Independent Study IMT 600

Convolutional Neural Networks

The convolutional neural network, or CNN for short, is a specialized type of neural network model designed for working with two-dimensional image data, although they can be used with one-dimensional and three-dimensional data.

Central to the convolutional neural network is the convolutional layer that gives the network its name. This layer performs an operation called a "*convolution*".

Convolution

In the context of a convolutional neural network, a convolution is a linear operation that involves the multiplication of a set of weights (also called kernel or filter) with the input, much like a traditional neural network. The filter is smaller than the input data and the type of multiplication applied between a filter-sized patch of the input and the filter is a dot product.

Pixel Values

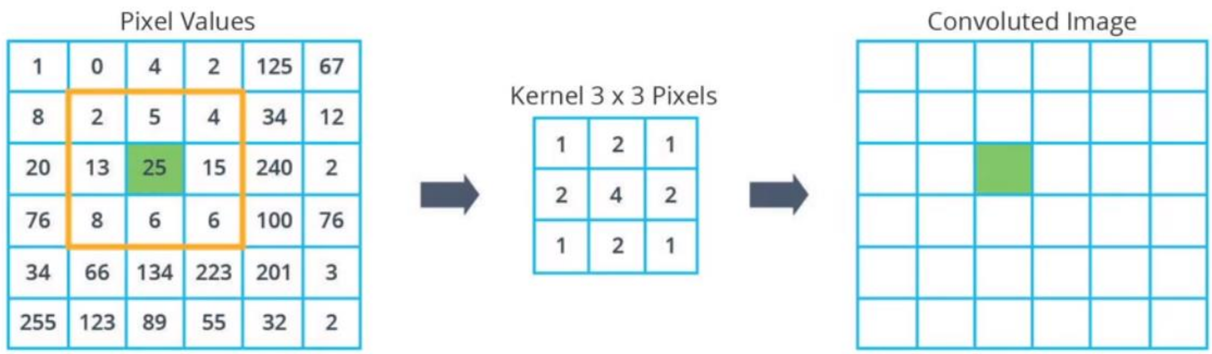
1	0	4	2	125	67
8	2	5	4	34	12
20	13	25	15	240	2
76	8	6	6	100	76
34	66	134	223	201	3
255	123	89	55	32	2

Kernel 3 x 3 Pixels

1	2	1
2	4	2
1	2	1

Using a filter smaller than the input is intentional as it allows the same filter (set of weights) to be multiplied by the input array multiple times at different points on the input. Specifically, the filter is applied systematically to each overlapping part or filter-sized patch of the input data, left to right, top to bottom.

This systematic application of the same filter across an image is a powerful idea. If the filter is designed to detect a specific type of feature in the input (horizontal lines, vertical lines, circular lines, dark spots etc.), then the application of that filter systematically across the entire input image allows the filter an opportunity to discover that feature anywhere in the image.



The values of the filter are learnt iteratively by the neural network during back propagation. In this example, we are using only one filter which captures only one feature/aspect of an image. In reality, we would be using many filters to capture various aspects.

Max Pooling

Convolutional layers in a convolutional neural network systematically apply learned filters to input images in order to create feature maps that summarize the presence of those features in the input.

Convolutional layers prove very effective and stacking **convolutional layers in deep models allows layers close to the input to learn low-level features (e.g. lines) and layers deeper in the model to learn high-order or more abstract features, like shapes or specific objects.**

A limitation of the feature map output of convolutional layers is that they record the precise position of features in the input and occupy a large spatial size (High dimension).

A common approach to addressing this problem from signal processing is called ***down sampling***. This is where a **lower resolution version of an input signal is created that still contains the large or important structural elements, without the fine detail that may not be as useful to the task**.

The function of Pooling is to progressively reduce the spatial size of the input representation. In particular, pooling:

- Makes the input representations (feature dimension) smaller and more manageable
- Reduces the number of parameters and computations in the network, therefore, controlling overfitting.
- Makes the network invariant to small transformations, distortions and translations in the input image (a small distortion in input will not change the output of Pooling)
- Helps us arrive at an almost scale invariant representation of our image (the exact term is "equivariant"). This is very powerful since we can detect objects in an image no matter where they are located.

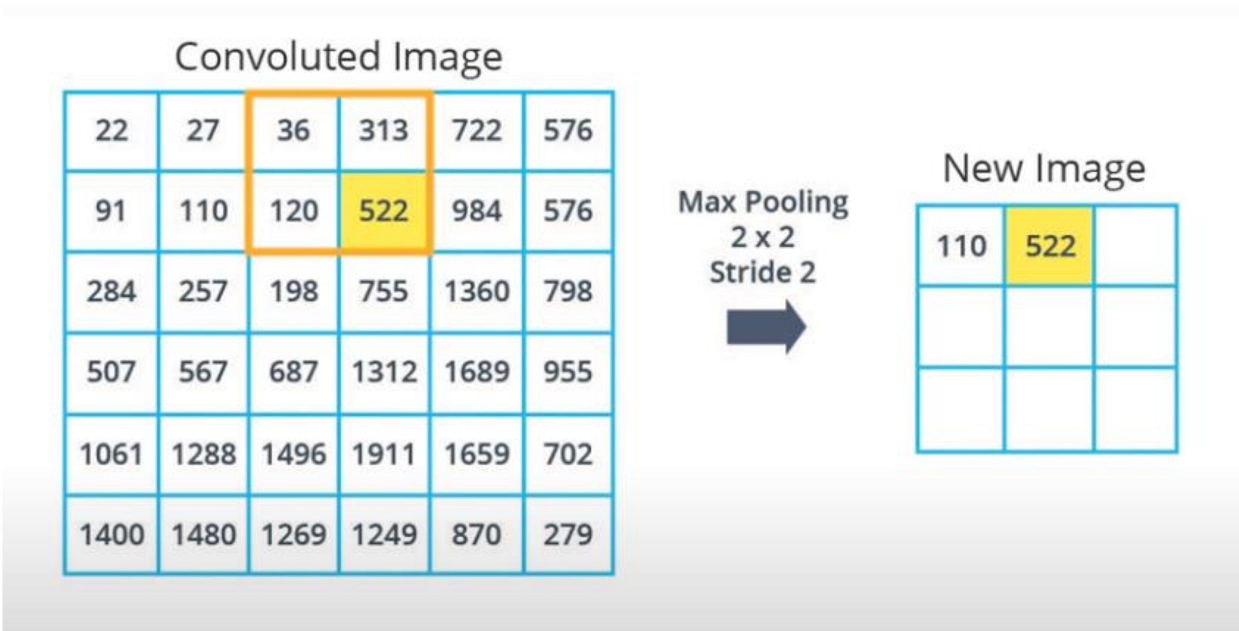
A pooling layer is a new layer added after the convolutional layer. Specifically, after a nonlinearity (e.g. ReLU) has been applied to the feature maps output by a convolutional layer; for example, the layers in a model may look as follows:

- Input Image
- Convolutional Layer
- Nonlinearity
- Pooling Layer

Pooling involves selecting a pooling operation, much like a filter to be applied to feature maps. The size of the pooling operation or filter is smaller than the size of the feature map; specifically, it is almost always 2×2 pixels applied with a stride of 2 pixels.

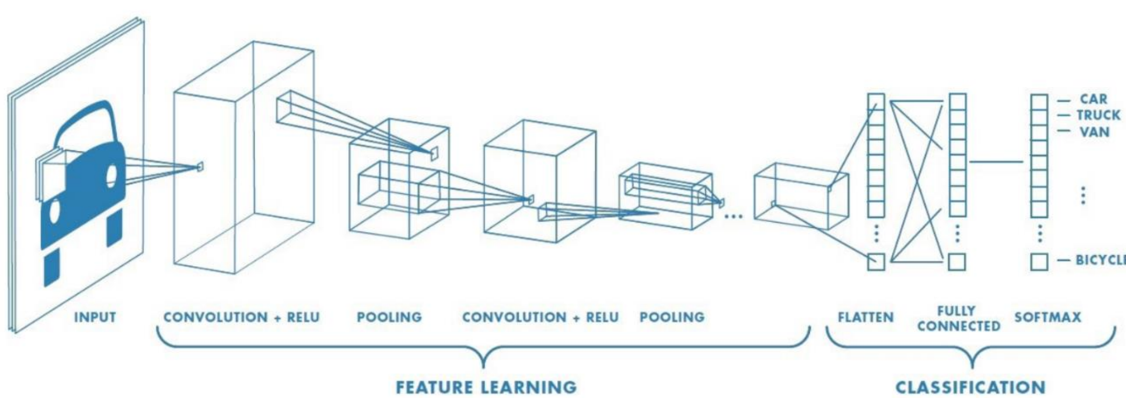
This means that the pooling layer will always reduce the size of each feature map by a factor of 2, e.g. each dimension is halved, reducing the number of pixels or values in each feature map to one quarter the size. For example, a pooling layer applied to a feature map of 6×6 (36 pixels) will result in an output pooled feature map of 3×3 (9 pixels).

Maximum pooling is an operation which calculates the maximum value for each patch of the feature map.



The result of using a pooling layer and creating down sampled or pooled feature maps is a summarized version of the features detected in the input.

The whole process is as follows:



Demonstration with an example

In our previous example, we built an image classifier using a multi-layer neural network.

```
[ ]  model = tf.keras.Sequential([
         tf.keras.layers.Flatten(input_shape=(28, 28, 1)),
         tf.keras.layers.Dense(128, activation=tf.nn.relu),
         tf.keras.layers.Dense(10, activation=tf.nn.softmax)
     ])
```

We add the convolution and max pool layers to the same neural network to make it into a Convolutional Neural Network.

```
[ ]  model = tf.keras.Sequential([
         tf.keras.layers.Conv2D(32, (3,3), padding='same', activation=tf.nn.relu,
                                input_shape=(28, 28, 1)),
         tf.keras.layers.MaxPooling2D((2, 2), strides=2),
         tf.keras.layers.Conv2D(64, (3,3), padding='same', activation=tf.nn.relu),
         tf.keras.layers.MaxPooling2D((2, 2), strides=2),
         tf.keras.layers.Flatten(),
         tf.keras.layers.Dense(128, activation=tf.nn.relu),
         tf.keras.layers.Dense(10, activation=tf.nn.softmax)
     ])
```

Adding the convolution and max pool layers increased the model's accuracy from 81% to 97%.

Additional Resources

This blog has good visual representations and animations depicting the techniques of convolution and max pool layers: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/