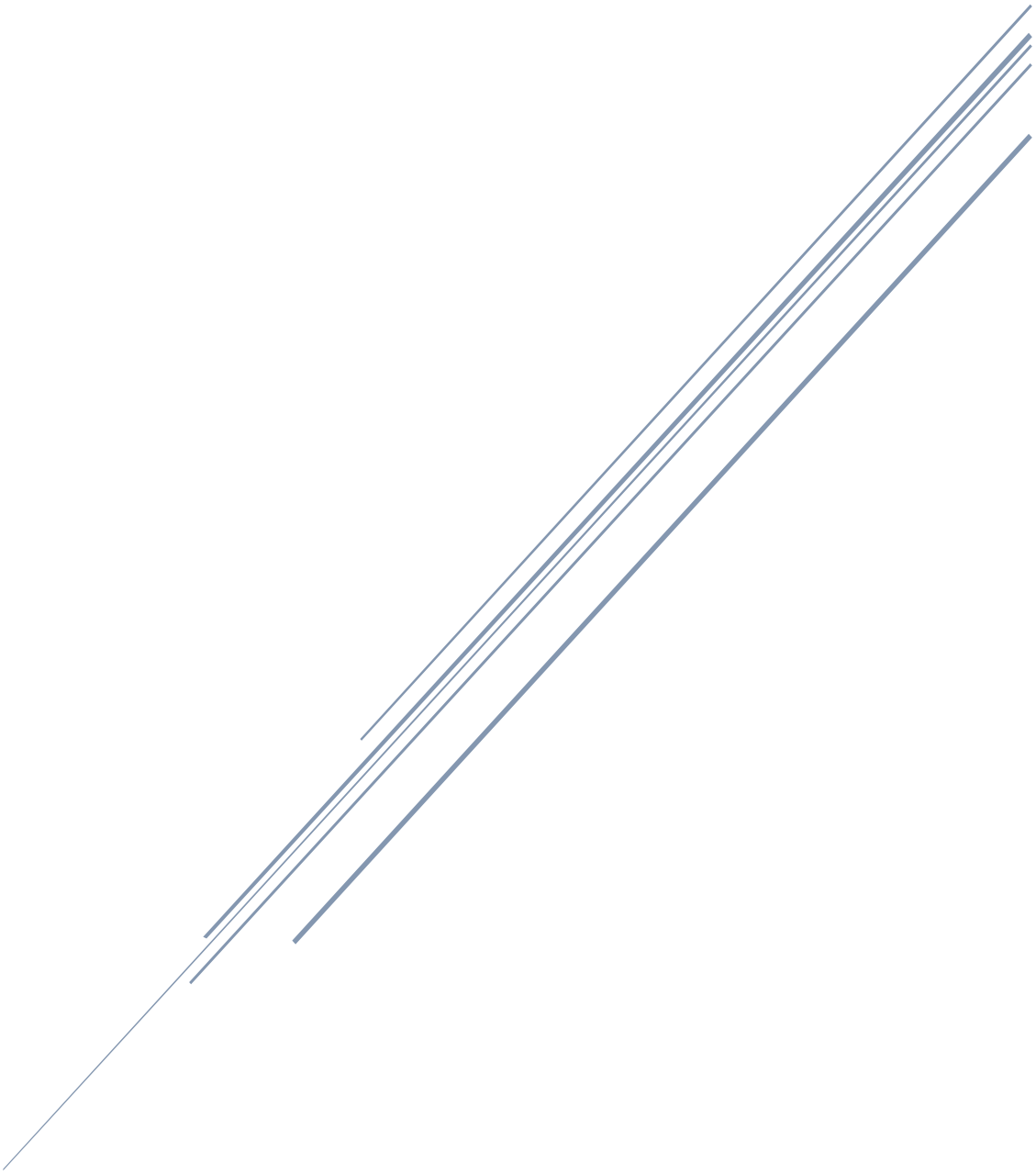
CONVOLUTIONAL NETWORKS

Module 2



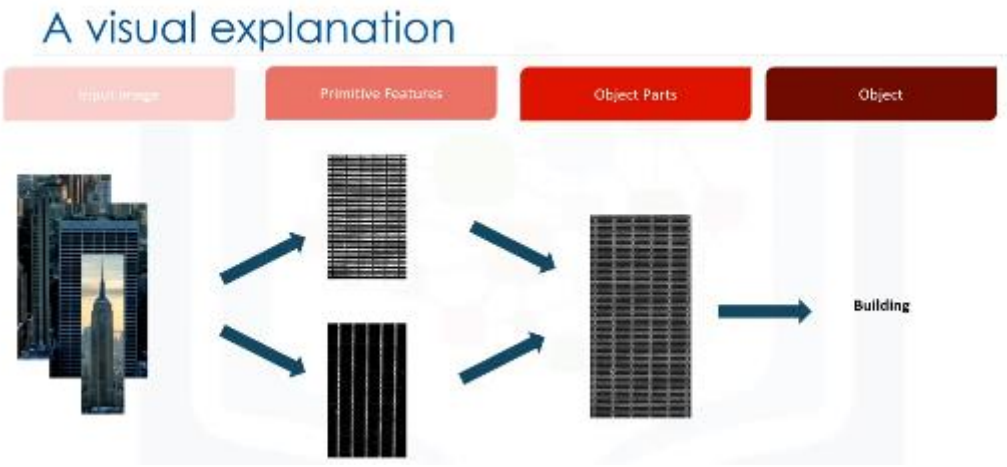
v-cardona

Deep Learning with Tensorflow

Contenido

Intro to CNNs

The CNN solution needed to have two key features to make it viable: First, it needed to be able to detect the objects in the image and place them into the appropriate category. And second, it also needed to be robust against differences in pose, scale, illumination, conformation, and clutter.



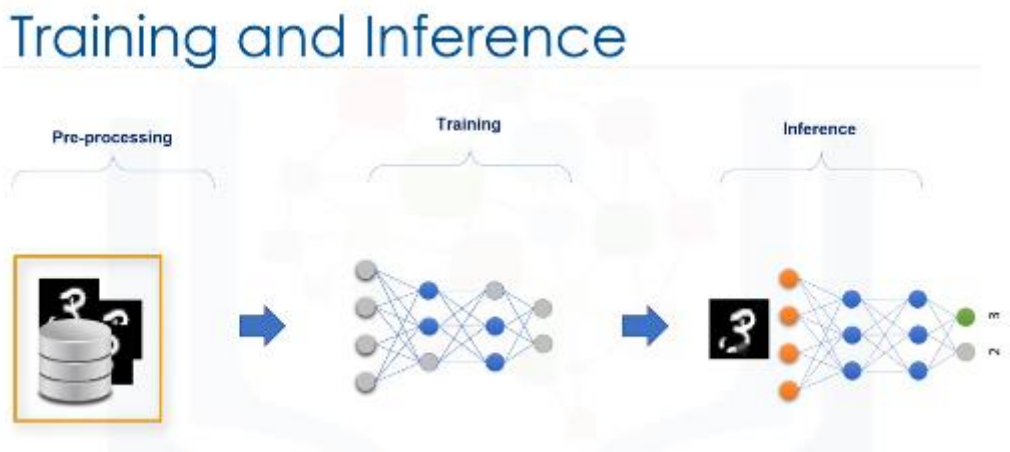
The CNN, is a set of layers, with each of them being responsible to detect a set of feature sets. And these features are going to be more abstract as it goes further into examining the next layer.

CNNs for classification

One of the important steps in the modeling for classification with Shallow Neural Networks, is the feature extraction step. These chosen features could simply be the color, object edges, pixel location, or countless other features that could be extracted from the images. Of course, the better and more effective the feature sets you find, the more accurate and efficient the image classification you can obtain. However, as you can imagine, the process of selecting the best features is tremendously time-consuming, and is often ineffective.

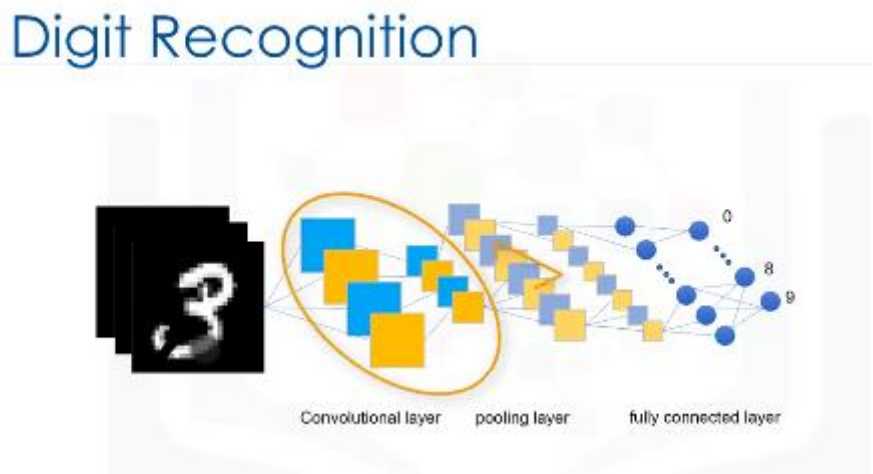
Convolutional neural networks (or CNNs) try to solve this problem by using more hidden layers, and also with more specific layers.

Phases:



1. In the first part, we have to convert the images to a readable and proper format for our network.
2. Then, an untrained network is fed with a big dataset of the images in the Training phase, so as to allow the network to learn. That is, we build a CNN, and train it with many hand-written images in the training set.
3. Finally, we use the trained model in the Inference phase, which classifies a new image by inferring its similarity to the trained model. So, this model can be deployed and used as a digit recognition model for unseen cases.

Focus on the training process. As mentioned before, a deep neural network not only has multiple hidden layers, the type of layers and their connectivity also is different from a shallow neural network, in that it usually has multiple Convolutional layers, pooling layers, as well as fully connected layers.



1. The convolutional layer applies a convolution operation to the input, and passes the result to the next layer.
2. The pooling layer combines the outputs of a cluster of neurons in the previous layer into a single neuron in the next layer.
3. And then, the fully connected layers connect every neuron in the previous layer to every neuron in the next layer.

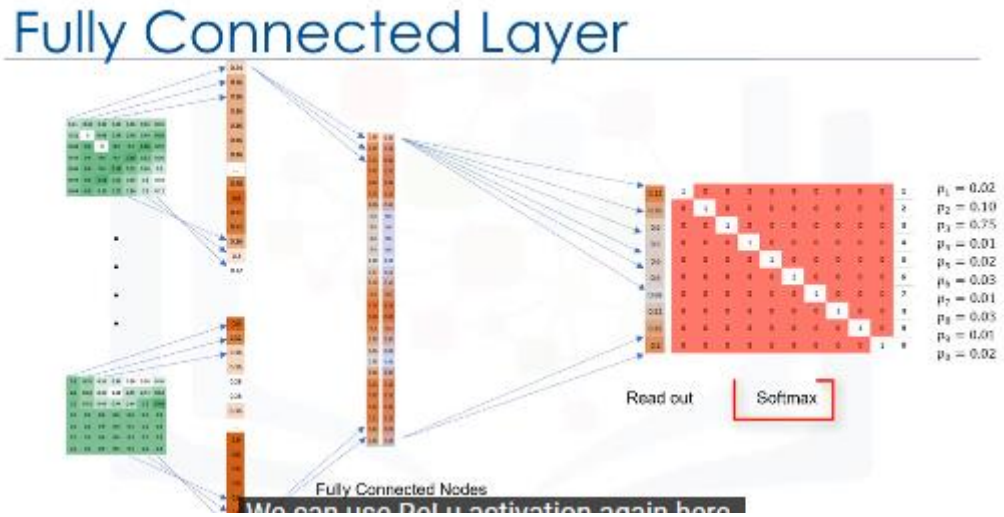
CNN architecture

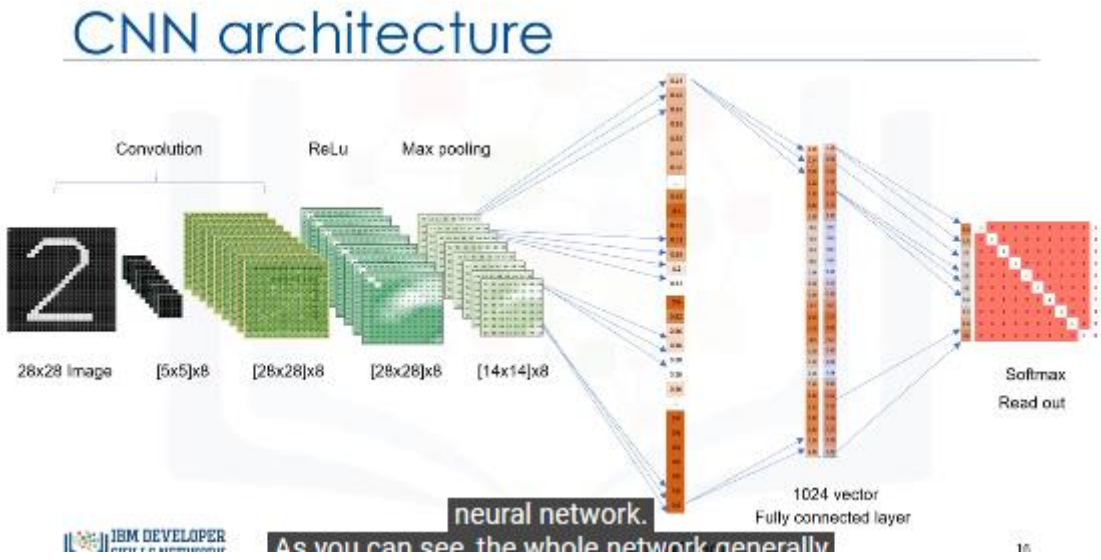
The main purpose of the Convolutional layer is to detect different patterns or features from an input image, for example, its edges. Assume that you have an image, and you want to find the edges from the photo ... so that you can define a filter, which is also called a kernel. Leveraging different kernels allows us to find different patterns in an image, such as edges, curves, and so on. Typically, you initialize the kernels with random values, and during the training phase, the values will be updated with optimum values, in such a way that the digit is recognized.

We use ReLu (which is short for rectified linear unit) as the activation function on top of nodes in the Convolution layer. ReLu is a nonlinear activation function, which is used to increase the nonlinear properties of the decision function.

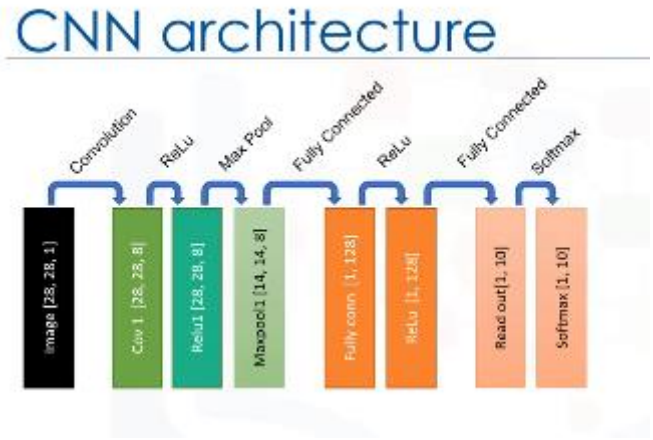
Next layer is Max pooling, and his purpose is to reduce the dimensionality of the activate neurons. “Max-Pooling” is an operation that finds the maximum values and simplifies the inputs. In other words, it reduces the number of parameters within the model. A Stride dictates the sliding behavior of the Max-Pooling. For example, if we select stride=2, the window will move 2 pixels every time, thus not overlapping.

The next layer is the Fully-Connected layer. Fully-Connected layers take the high-level filtered images from the previous layer, and convert them into a vector. First, each previous layer matrix will be converted to a one-dimensional vector. Then, it will be Fully-Connected to the next hidden layer, which usually has a low number of hidden units. We call it Fully-Connected because each node from the previous layer is connected to all the nodes of this layer. It is connected through a weight matrix. And finally, we use Softmax to find the class of each digit. Softmax is an activation function that is normally used in classification problems. It generates the probabilities for the output.





The whole network generally is doing two tasks. The first part of this network is all about feature learning and extraction and the second part revolves around classification.



We can repeated layers, by adding again a convolution layer or relu and max pool.

Feature engineering is the process of extracting useful patterns from input data that will help the prediction model to understand better the real nature of the problem. A good feature learning will present patterns in a way that significantly increase the accuracy and performance of the applied machine learning algorithms in a way that would otherwise be impossible or too expensive by just machine learning itself.

Feature learning algorithms finds the common patterns that are important to distinguish between the wanted classes and extract them automatically. After this process, they are ready to be used in a classification or regression problem.

The great advantage of CNNs is that they are uncommonly good at finding features in images that grow after each level, resulting in high-level features in the end. The final layers (can be one or more) use all these generated features for classification or regression.

- Convolution operation between two arrays (sliding x windows over h):
  y = np.convolve(x, h)

- Convolution with full operation:
  y = np.convolve(x, h, "full")

```
[2  6]
 |  |
 V  V
 0 [1 2 5 4]
```

= 2 * 0 + 6 * 1 = 6

Second step:

```
  [2  6]
   |  |
   V  V
0 [1  2  5  4]
```

= 2 * 1 + 6 * 2 = 14 (the ar
input)

Third step:

```
     [2  6]
      |  |
      V  V
0 [1  2  5  4]
```

= 2 * 2 + 6 * 5 = 34

Fourth step:

```
        [2  6]
         |  |
         V  V
0 [1  2  5  4]
```

= 2 * 5 + 6 * 4 = 34

Fifth step:

```
           [2  6]
            |  |
            V  V
0 [1  2  5  4] 0
```

- Convolution with same operation: In this approach, we just add the zero to left (and top of the matrix in 2D). That is, only the first 4 steps of "full" method:
  y = np.convolve(x, h, "same")

- Convolution with no padding operation:
  y = np.convolve(x, h, "valid")

```
[2  6]
 |  |
 V  V
[1  2  5  4]
```

= 2 * 1 + 6 * 2 = 14 (the ar
input)

Second step:

```
   [2  6]
    |  |
    V  V
[1  2  5  4]
```

= 2 * 2 + 6 * 5 = 34

Third step:

```
      [2  6]
       |  |
       V  V
[1  2  5  4]
```

= 2 * 5 + 6 * 4 = 34

You have two basic options when using TensorFlow to run your code:

- [Build graphs and run session] Do all the set-up and THEN execute a session to evaluate tensors and run operations (ops)
- [Interactive session] create your coding and run on the fly.