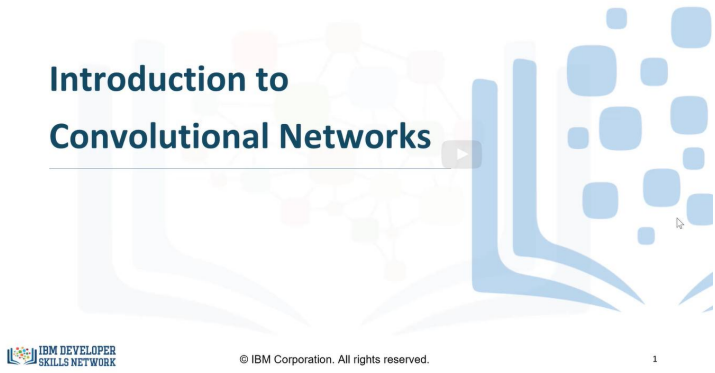
Module 2 - Convolutional Networks

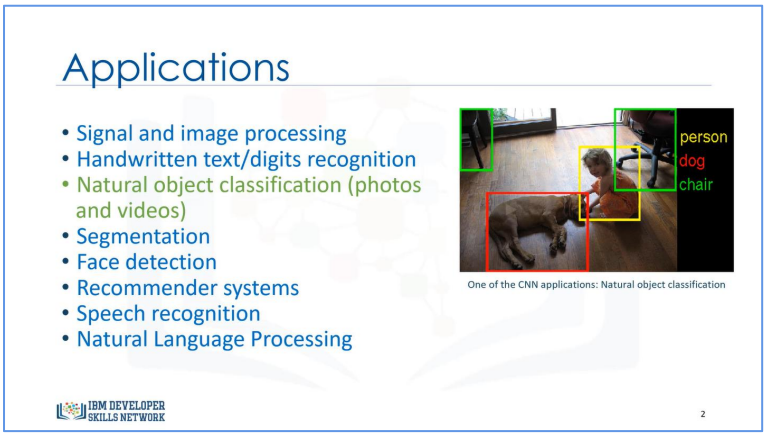
Learning Objectives

In this lesson you will learn about:

- Introduction to Convolutional Networks
- Convolution and Feature Learning
- Convolution with Python and Tensor Flow
- The MNIST Database
- Multilayer Perceptron with Tensor Flow
- Convolutional Network with Tensor Flow

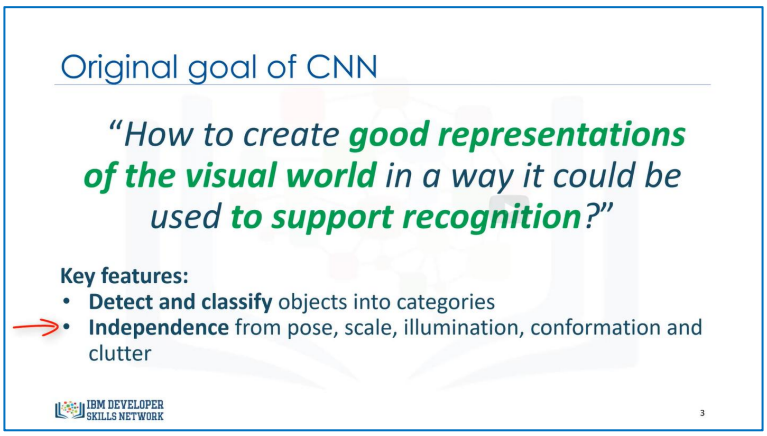
2.1 Intro to CNNs





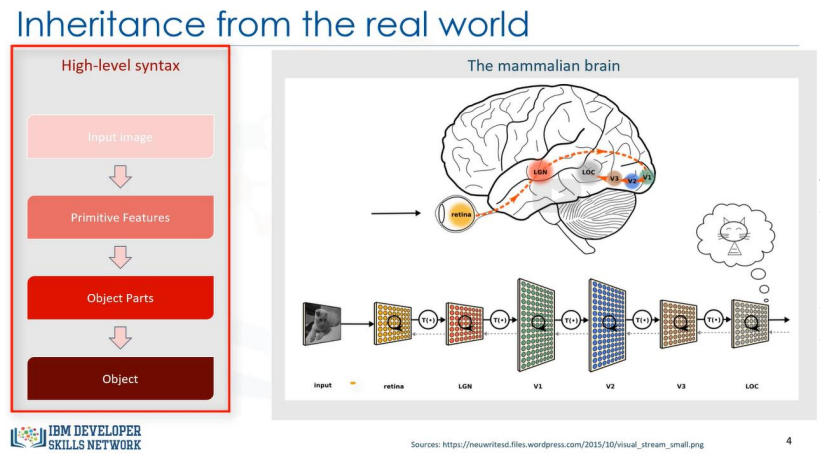
Convolutional neural networks, or CNNs, have gained a lot of attention in the machine learning community over the last few years. This is due to the wide range of applications where CNNs excel, such as object detection and speech recognition.

In a later part of this video, we'll examine the object recognition problem, which is a key consideration when using CNNs. You'll also get an idea of how a CNN's structure allows it to extract the elements from an image.
For example, you'll see how a CNN learns to pick out a person, or a dog, or even chairs from an image, even if the chairs are only partially visible. Indeed, this is not an easy task for computers, and it took years of dedicated research to achieve this.



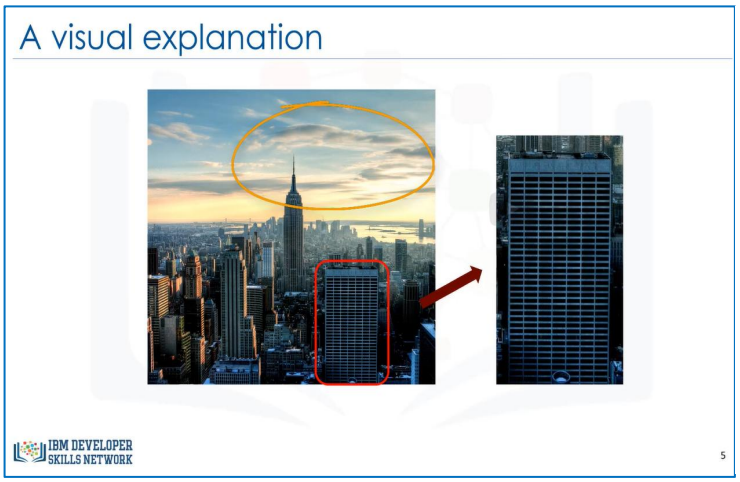
The original goal of the CNN was to form the best possible representation of our visual world, in order to support recognition tasks. The CNN solution needed to have two key featuresto make it viable:

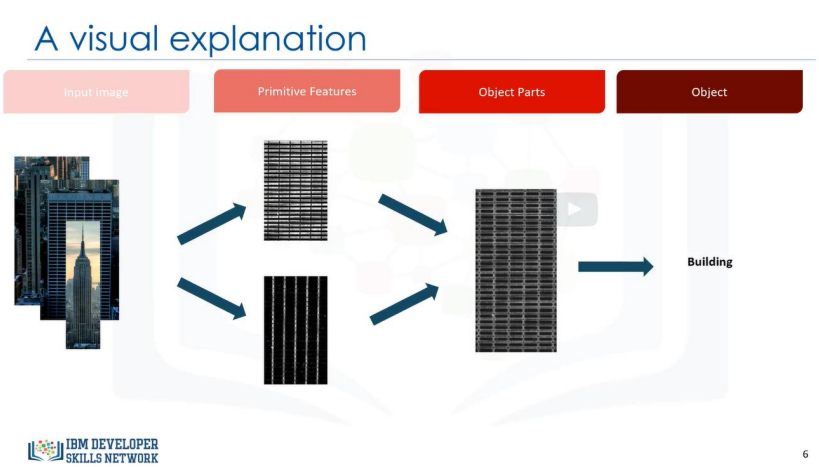
- First, it needed to be able to detect the objects in the image and place them into the appropriate category.
- And second, it also needed to be robust against differences in pose, scale, illumination, conformation, and clutter. The importance of this second feature simply cannot be overstated, since this was historically a huge limitation of hand-coded algorithms.



Interestingly enough, the solution to the object recognition issue was inspired by examining the way our own visual cortex operates. In short, the CNN starts with an input image, it then extracts a few primitive features, and combines those features to form certain parts of the object; and finally, it pulls together all of the various parts to form the object itself.
In essence, it is a hierarchical way of seeing objects. That is, in the first layer, very simple features are detected. Then, these are combined to shape more complicated features in the second layer, and so on to detect things like a cat or a dog or any other object we're needing to detect. The Convolutional Neural Network was developed from this sequence of steps.

总结:

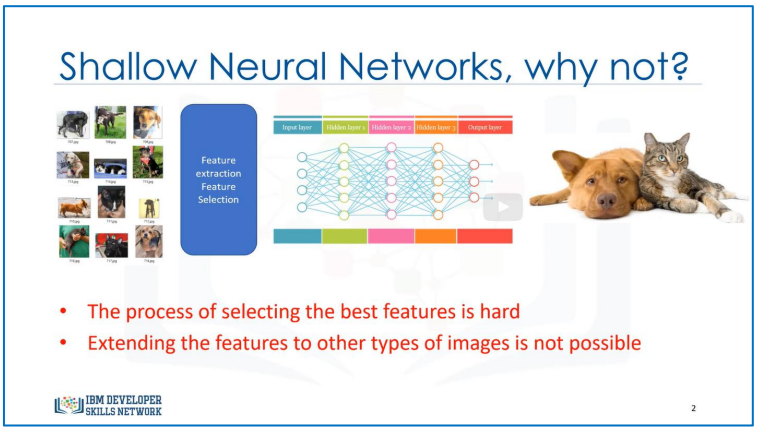




(左图) Let's consider the building in the lower right hand corner of this image. How can a trained CNN learn and recognize that it's an image of building, rather than some of the other elements that are situated in the photograph, such as a person, or even those clouds in the sky?

(右图) Well, in the training phase, CNNs would receive many building images as input. Then, CNNs will automatically find that the best primitive features for a building would be things like horizontal and vertical lines. Much like the neurons firing in a person's visual cortex, the first layer in a CNN reacts, so to speak, when it receives an image that has horizontal or vertical lines. Once these simpler features, such as lines, are combined, CNNs learn to form higher abstract components, like the windows of the building and the building's external shape. It can then use these basic parts to form the complete object, and learn how the entire building essentially looks like.
Later, if it sees an image of a building that it hasn't seen before, CNNs can make a decision if the new input image is of a building, based on the persistence of the various features it has stored. It is the same process for learning and detecting other objects, such as faces, animals, cars, and so on. So, as you can see, 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. So at this point, you should have a basic understanding of the principles behind a convolutional neural network, and how CNNs might be used in an application.

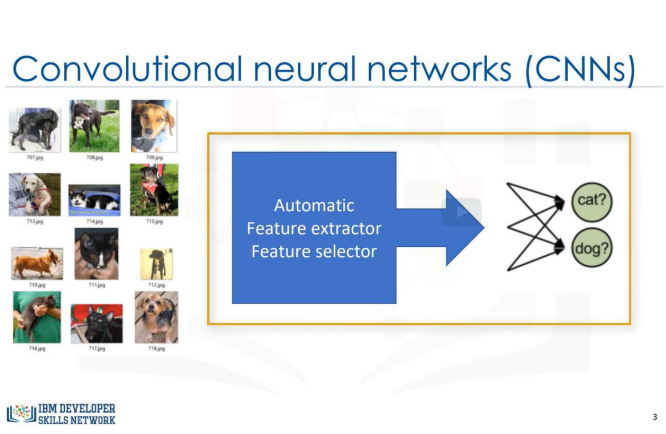
2.2 CNNs for Classification

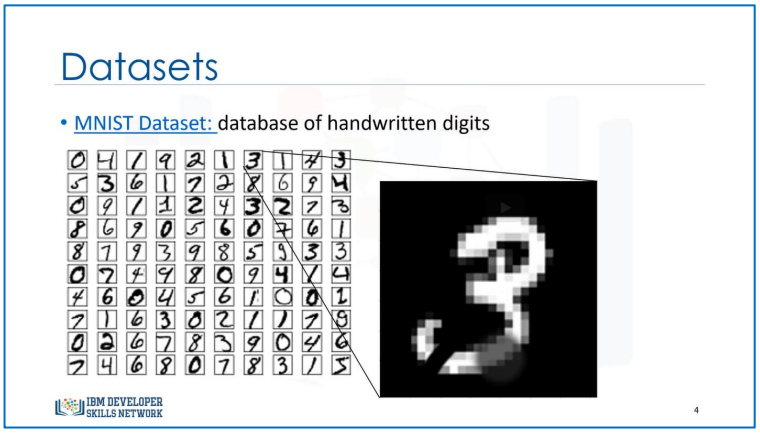


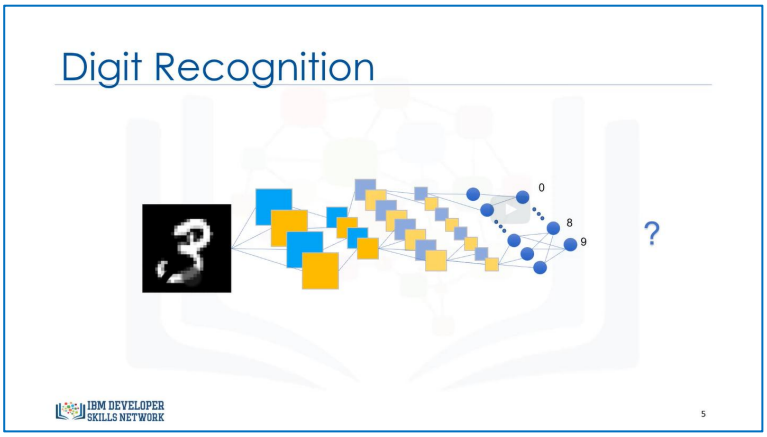
In this video, we'll explain how CNN can solve an image classification problem such as digit recognition.

First, let's see why traditional shallow neural networks do not work as well as CNN. As mentioned before, 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. Also, extending the features to other types of images becomes an even greater problem. Also, extending the features to other types of images becomes an even greater problem.

总结:

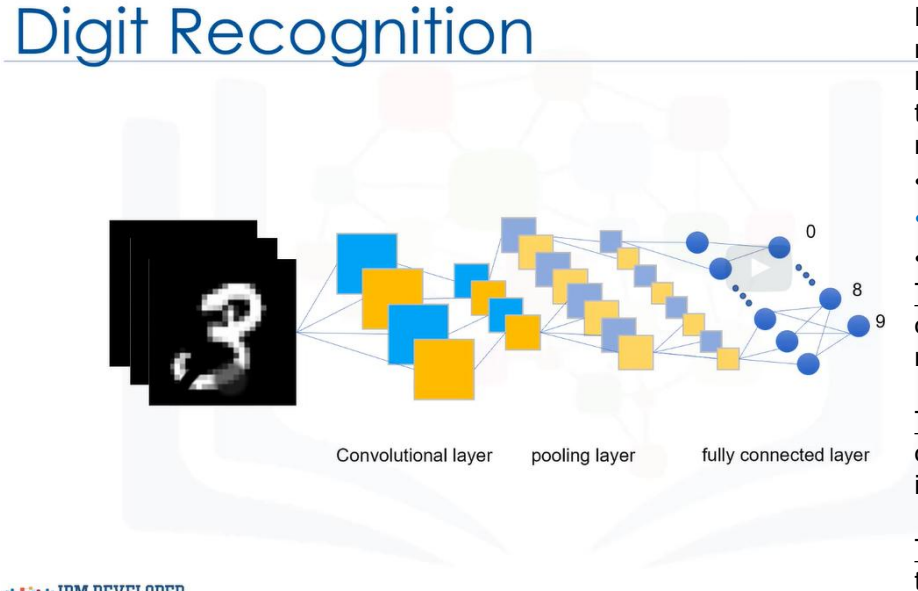






(上右) Convolutional neural networks (or CNNs) try to solve this problem by using more hidden layers, and also with more specific layers. So, when using CNN, instead of you choosing image features, to classify dogs vs. cats, for instance, CNNs can automatically find those features and classify the images for you.
Instead of using a dogs and cats dataset, though, let's use a more practical example here. The MNIST dataset is a database of handwritten digits that has a training set of 60,000 examples. The great thing about the MNIST dataset is that the digits have been size-normalized and centered in a fixed-size image".

(下左) Now, consider the digit recognition problem. We would like to classify images of handwritten numbers, where the observations are the intensity of the pixels. And, the target will be a digit, from 0 to 9. So, our objective here is to build a digit recognition system using CNN.
(下右) Now let's look at it from a higher level. Basically, if we look at the pipeline of our deep learning process, we can see the following phases: First, pre-processing of input data. Second, training the deep learning model. And third, inference and deployment of the model. In the first part, we have to convert the images to a readable and proper format for our network. 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. 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.



Now let's 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.

The convolutional layer applies a convolution operation to the input, and passes the result to the next layer.

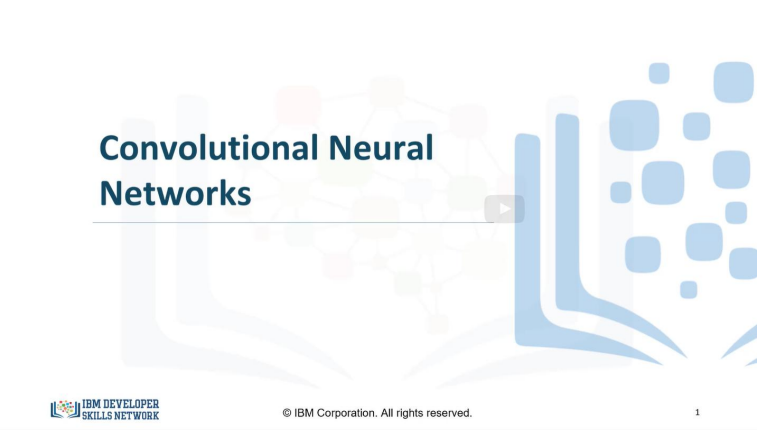
The pooling layer combines the outputs of a cluster of neurons in the previous layer into a single neuron in the next layer.

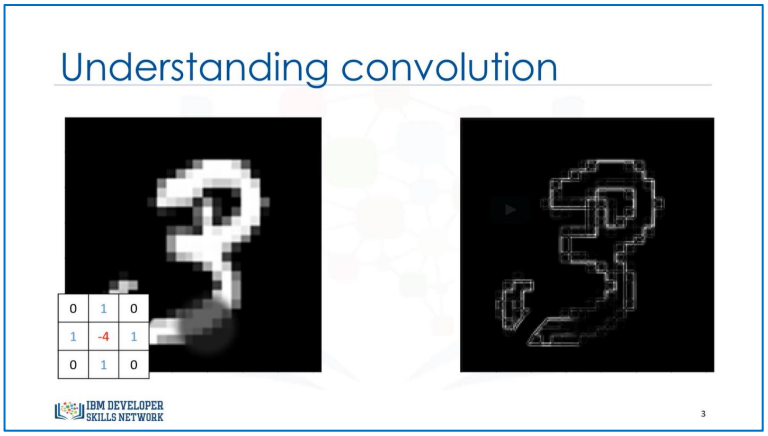
The fully connected layers connect every neuron in the previous layer to every neuron in the next layer.

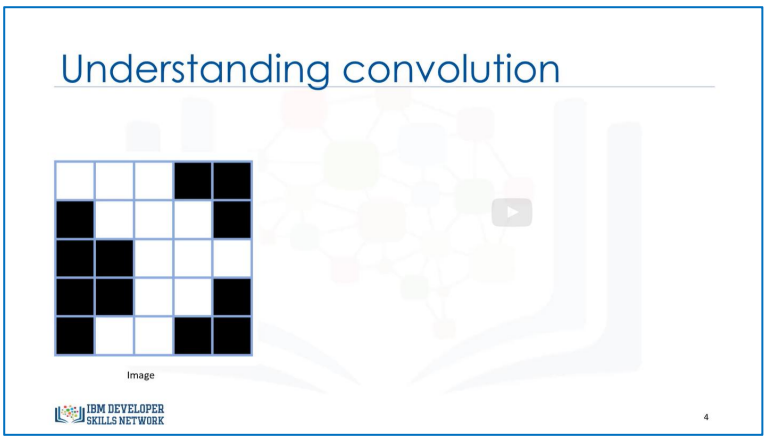
In the next video, you will learn more about these layers and their specifications, in detail.

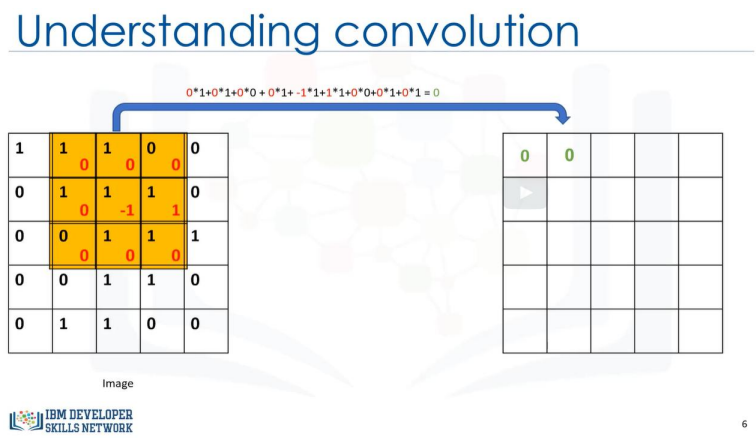
总结:

2.3 CNN Architecture









In this video, we'll be examining the architecture of the convolutional neural network model. As previously mentioned, CNN is a type of neural network, empowered with some specific hidden layers, including the Convolutional layer, the Pooling layer, and the Fully-Connected layer.

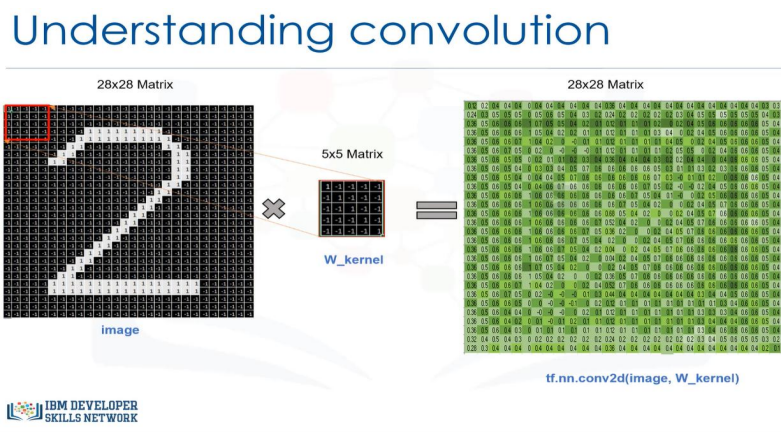
(右上) Let's start with the first layer, the Convolutional layer. The main purpose of the Convolutional layer is to detect different patterns or features from an input image, for example, its edges. Assume that you have an image, and you want to find the edges from the photo ... so that you can define a filter, which is also called a kernel. Now, if you slide the filter over the image, and apply the dot product of the filter to the image pixels, the result would be a new image with all the edges. That is, applying the filter to the left image will result in the right features that are typically useful for the initial layers of a CNN. In fact, the output is one of the very first primitive feature sets in the hierarchy of features. It is one of the key concepts behind CNNs.

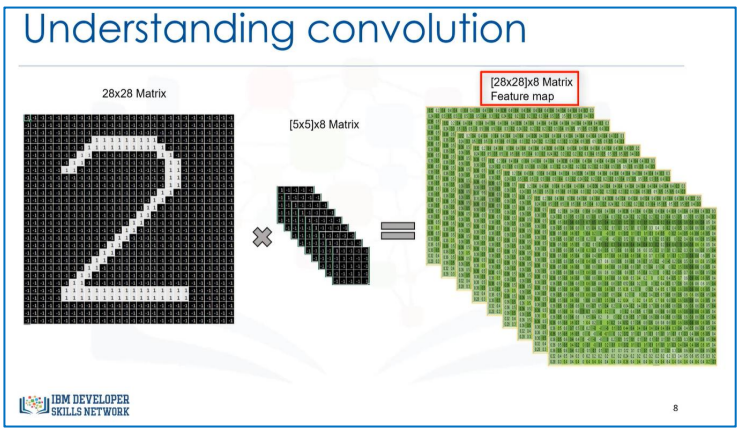
(左下) Now, the question is: what really happens mathematically when we apply a filter or kernel on an image? This brings us to the Convolution process. Convolution is a function that can detect edges, orientations, and small patterns in an image. The convolution operation can be seen as a mathematical function. Imagine that we have a simple black and white image as you see here. We can convert the image to a matrix of pixels with binary values, where 1 means a white pixel and 0 represents a black pixel. Please keep in mind, however, that the most common usage is a value between 0 and 255 for a grayscale image, or a 3-channel image for a color image. But for now and for the sake of simplicity, let's use 0 and 1 as our values.

(右下) In this step, we define a filter. It can be any type of filter, even a random filter, but let's use the edge detector filter for now.

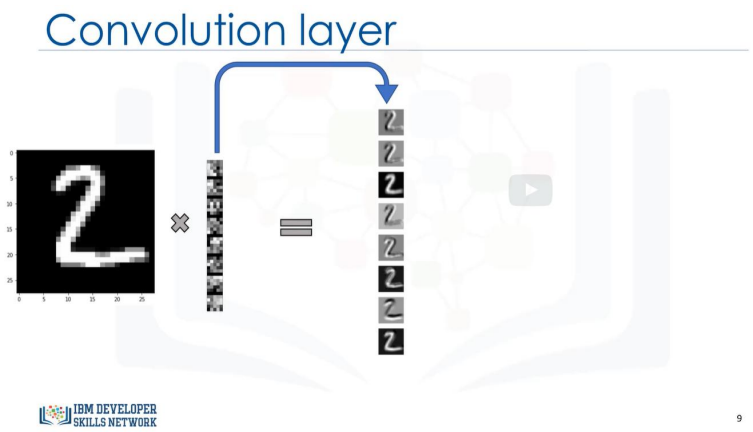
- Ok, now let's slide it over the image. And then calculate the convolved value for the first slide. Taking the values −1 and 1 on two adjacent pixels and zero everywhere else for the filter, results in the following image.
- Ok, now let's slide the kernel again. It will calculate the second element of our convolved image. We repeat this process, to complete the convolved matrix. So, in essence, the Convolution Function is a simple matrix multiplication, and in this instance, the result shows us the edges of the image.

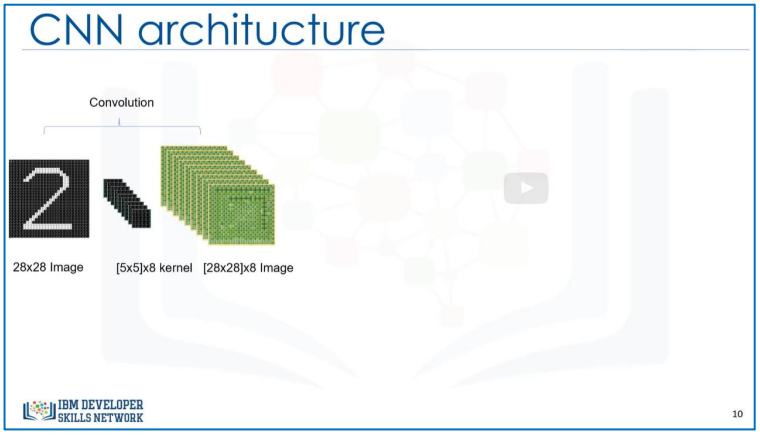
总结:

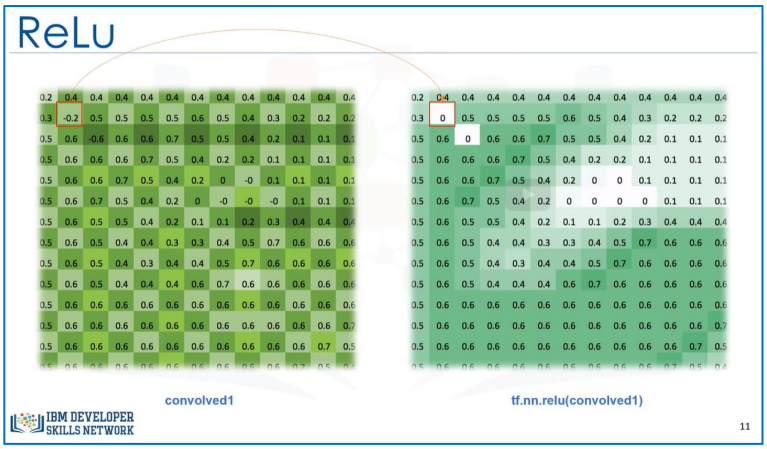




Now, let's consider a hand-written digit that we want to recognize. We can apply a kernel to it. It is, in fact, the sum of "element-wise multiplication" of the kernel matrix and the image matrix, as shown on the previous slide. So, we can show the result as convolving a kernel on the image.
Applying multiple kernels on one image will result in multiple convolved images. Now you can understand that leveraging different kernels allows us to find different patterns in an image, such as edges, curves, and so on.
The output of the convolution process is called a 'feature map'. At this point, you might be asking yourself: "How do I choose or initialize the proper kernels?" Well, 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.







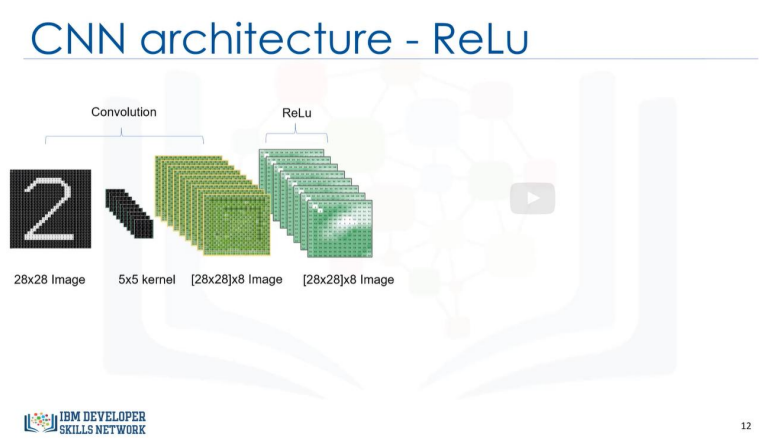
(左上) For example, here is the result of applying 8 different kernels on the digit 2. As you can see, each kernel will recognize a particular pattern in the digit.

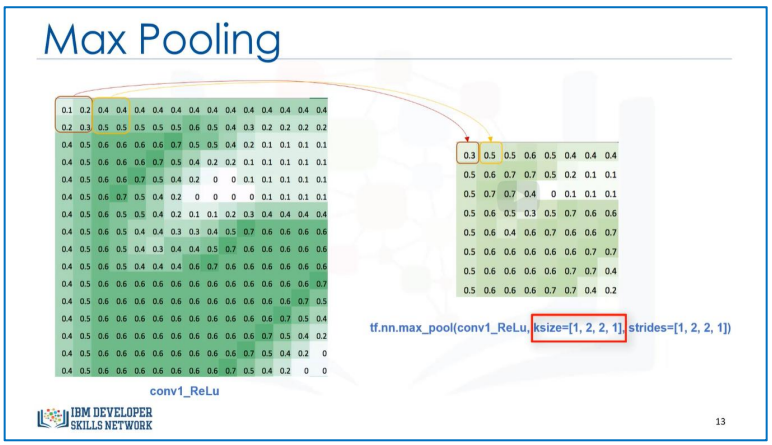
(右上) This is the first layer of Convolutional Deep Learning. We can interpret this layer to traditional neural network terminology.

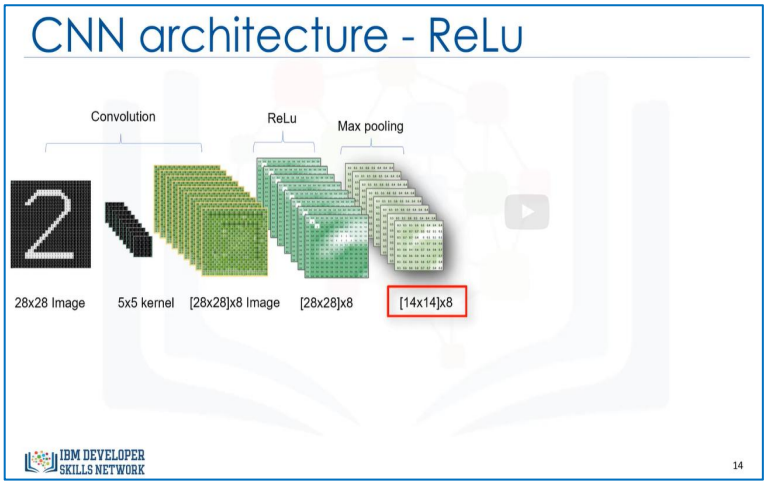
- The input image can be considered as a matrix that we feed into the neural network through input nodes. For example, in this case, the input layer of the network has 28-by-28 nodes. The output of the co-evolution process, are 8 of [28-by-28] neurons. We can assume those as hidden nodes.
- So, what are the weights between the input and hidden nodes? Yes, the 8 kernels, which are 5-by-5.

(左下)Now we need to make a decision as to whether each hidden neuron should be "fired" or not. So, we have to add "activation functions" to the neurons. 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. So, how can we apply it on Convolution layer? Well, we just go through all outputs of the Convolution layer, covolved1, and wherever a negative number occurs, we swap it out for a 0. This is called the ReLu activation Function.

总结:



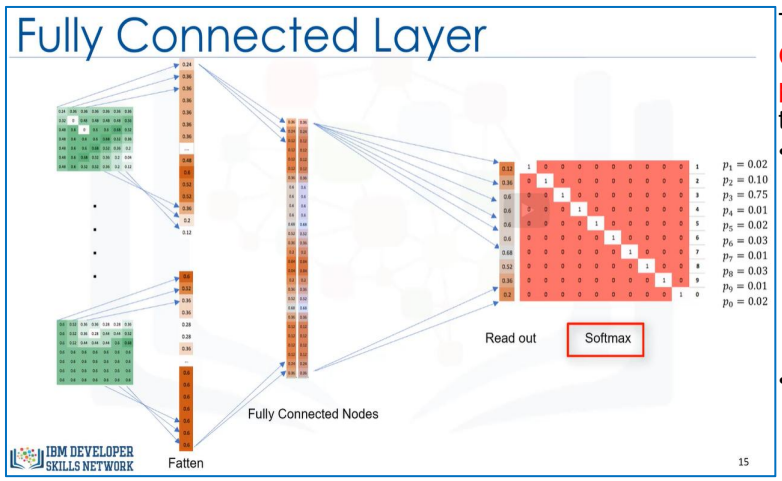




Now, we have gone one step further. In this step, we have to down-sample the output images from the ReLu function, to reduce the dimensionality of the activate neurons. So, we use another layer, which is called Max-Pooling. "Max-Pooling" is an operation that finds the maximum values and simplifies the inputs. In other words, it reduces the number of parameters within the model. In a sense, it turns the low level data into higher level information. For example,

- we can select a window of size of 2-by-2, and then select the maximum value for this matrix. That is, if the image is a 2-by-2 matrix, it would result in one output pixel.
- Also, in this step, we can define Strides. 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.

Ok, now we should have our output matrices, but with lower dimensions, for example 14-by-14.

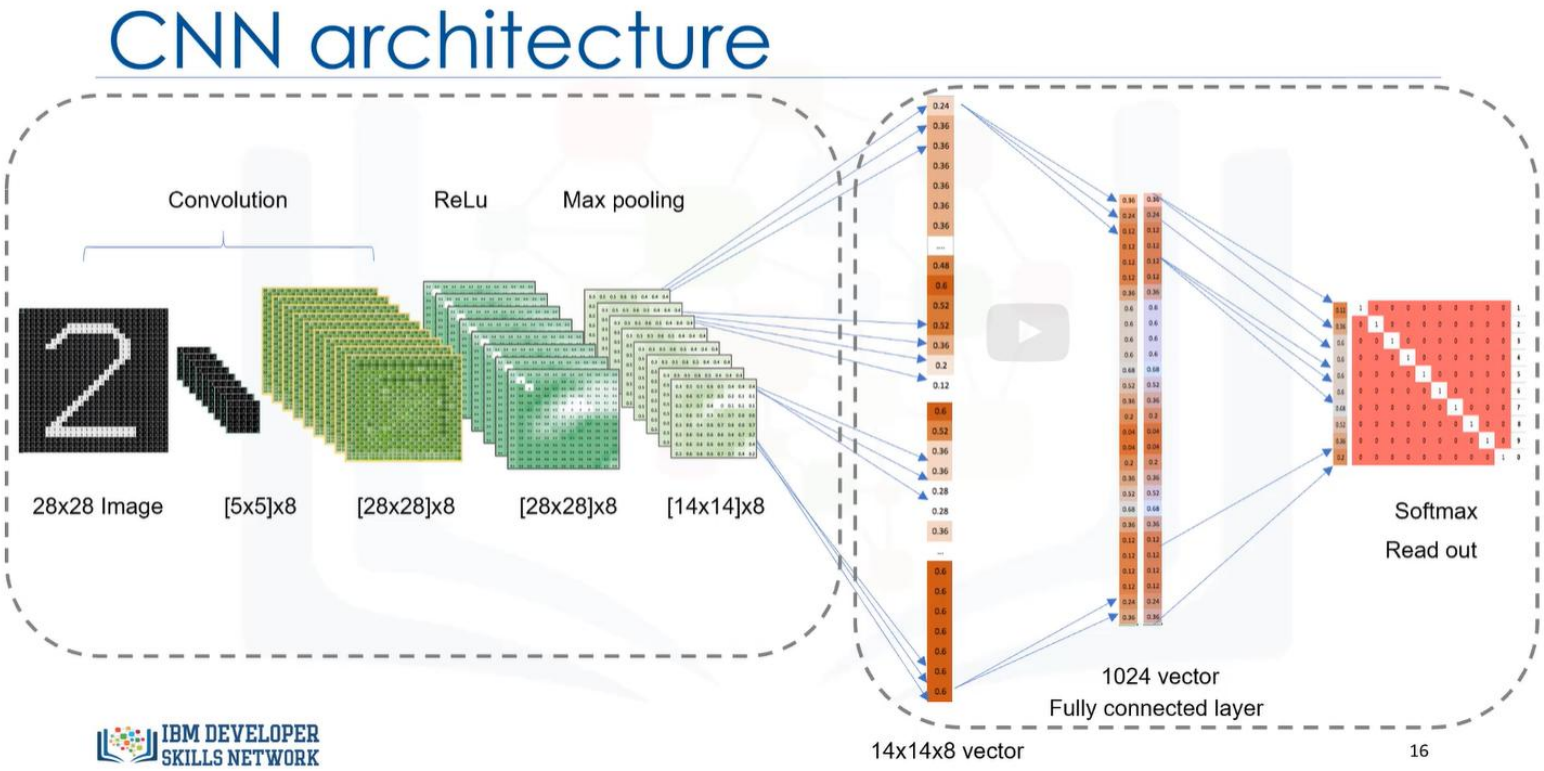


The next layer is the Fully-Connected layer. Fully-Connected layers take the high-level filtered images from the previous layer, that is, all 8 matrices in our case, and convert them into a vector.

- 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. We can use ReLu activation again here.
- 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.

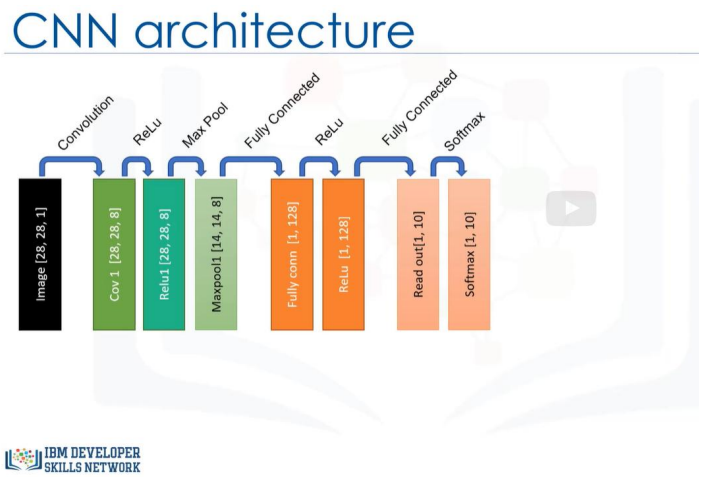
For example, our model will not be 100% sure that one digit is the number 3; instead, the answer will be a distribution of probabilities where, if the model is right, the 3 will be assigned the larger probability. That is, Softmax can output a multiclass categorical probability distribution.

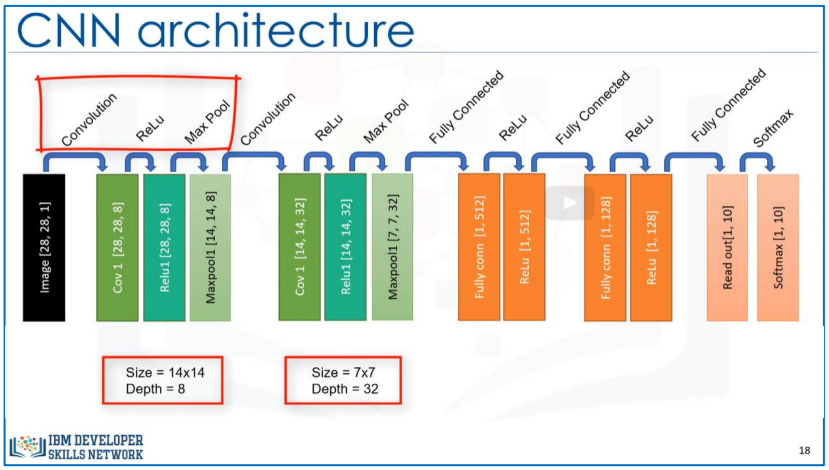
总结:



Now, let's put all these layers together. This chart shows the main layers of a convolutional neural network. As you can see, the whole network generally is doing two tasks.

- The first part of this network is all about feature learning and extraction.
- The second part revolves around classification.



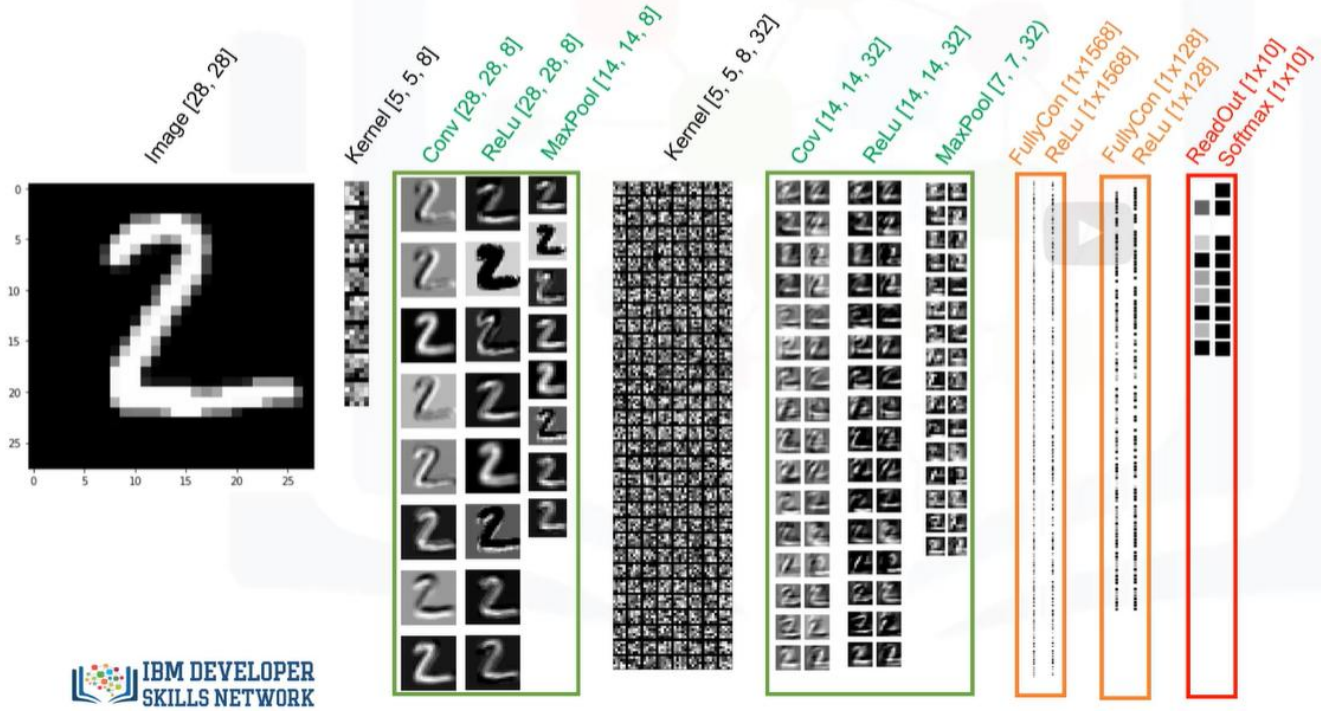


So, if we look at each operation as a building block, we can see that a typical convolutional neural network might look like this diagram. As you can imagine, though, a typical CNN architecture for an image classification can be much more complicated. It can be a chain of repeating Conv, ReLu, Max-Pool operations. For example, here we see more than one convolutional layer, followed by a few Fully-Connected layers.

Note the effect of each single Conv, ReLu and Max-Pool operations pass through the image: it reduces height and width of the individual image, and then it increases the depth of the images. For example, the output of the first convolutional layer is an image of depth 8, and the output of the second layer is an image of depth 32. This is very important, because using multiple layers, CNN will be able to break down the complex patterns into a series of simpler patterns.

总结:

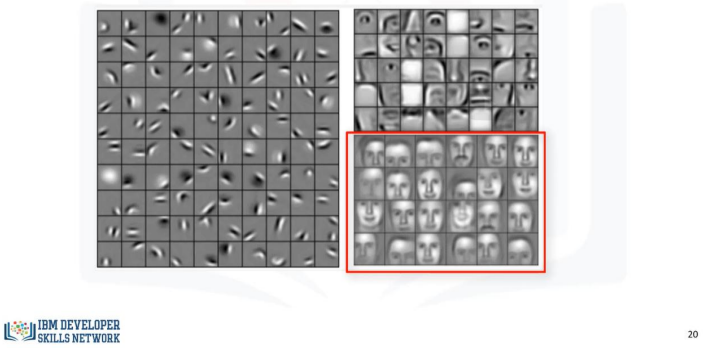
CNN architecture



If we pass one of the digits through the network, you can see the outputs.

- In the first convolutional layer, we apply 8 kernels of size 5-by-5. Then we apply ReLu and Max-Pool.
- In the second convolutional layer, we use 32 kernels. Again, we apply ReLu and Max-Pool.
- We connect the outputs to a Fully-Connected layer.
- Also, we use a second Fully-Connected layer with a lower number of nodes.
- And, finally, we use Softmax in the last layer.

CNN on faces



CNN Training

(左图) And now you can imagine that if we use a CNN trained on human faces,

- the first layers will represent mostly primitive features, for example the details of each part of the faces;
- and the next layers represent more abstract patterns such as the nose and eyes;
- with the last layers representing more face-like patterns.

(右图) Now, let's take a closer look at this architecture and see what happens in the training phase of Convolutional Neural Networks . As mentioned, a CNN is a type of feed-forward neural network, consisting of multiple layers of neurons. Each neuron in a layer receives some input, processes it, and optionally follows it with a non-linear output using an activation function. So, what is the network going to learn through the training process? Well, it learns the connections between the layers. These are the learnable weights and biases matrices. In fact, the whole network is about a series of dot products between the weight matrices and the input matrix. This means we must first initialize the weights of the network randomly. Then, we will keep feeding the network with a big dataset of images. We then check the output of the network and depending on how far the output is from the expected one, we change/update the weights. We keep repeating this process until it reaches high accuracy of prediction. Of course, this is a very high-level picture of the whole training process.

In any case, I hope you've absorbed the main ideas behind CNNs. For a better understanding of Convolutional Neural Networks, I recommend that you run the labs of this module, which will walk you through different layers of CNN.

总结:

What can be achieved with "convolution" operations on Images?

- ○ Noise Filtering
- ○ Image Smoothing
- ○ Image Blurring
- ○ Edge Detection
- ◉ All of the above

✔

For convolution, it is better to store images in a TensorFlow Graph as:

- ◉ Placeholder
- ○ CSV file
- ○ Numpy array
- ○ Variable
- ○ None of the above

✔

Which of the following statements is TRUE about Convolution Neural Networks (CNNs)?

- ○ CNN can be applied ONLY on Image and Text data
- ◉ CNN can be applied on ANY 2D and 3D array of data
- ○ CNN can be applied ONLY on Text and Speech data
- ○ CNN can be applied ONLY on Image data
- ○ All of the above

✔

总结：

Which of the following Layers can be part of Convolution Neural Networks (CNNs)

- ○ Dropout
- ○ Softmax
- ○ Maxpooling
- ○ Relu
- ◉ All of the above

✔

The objective of the Activation Function is to:

- ○ Increase the Size of the Network
- ◉ Handle Non-Linearity in the Network
- ○ Handle Linearity in the Network
- ○ Reduce the Size of the Network
- ○ None of the above

✔

总结：