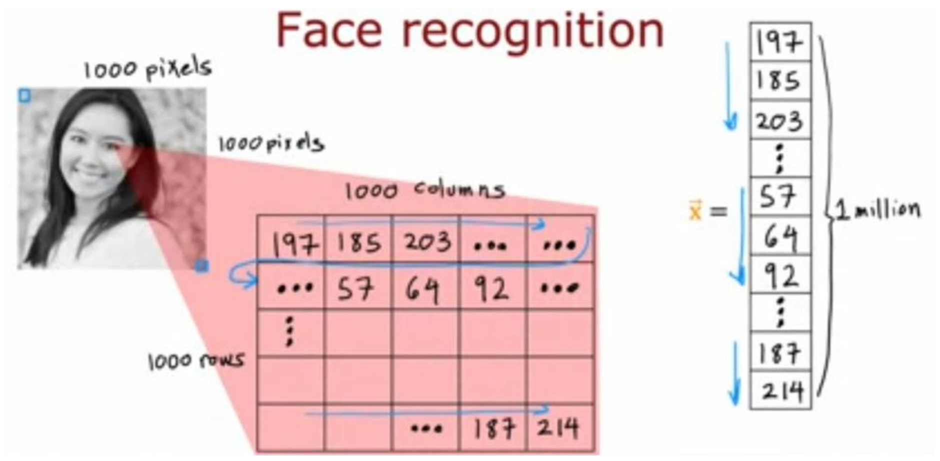
Exemple : Reconnaissance d'images

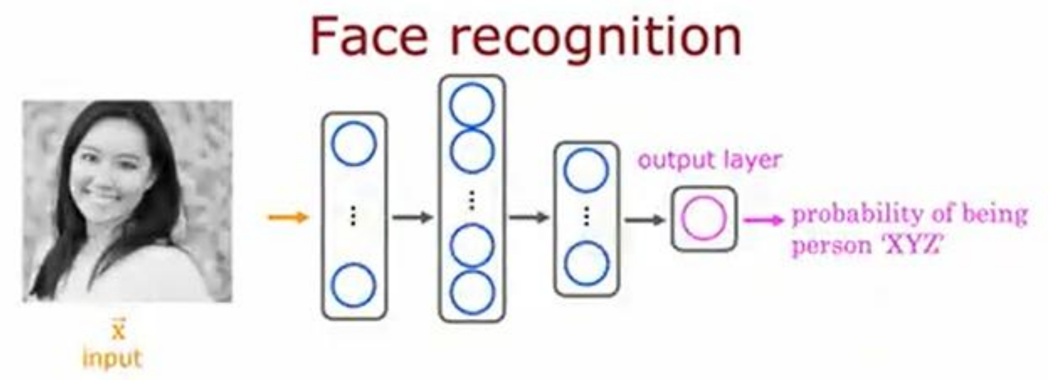
Let's dive in. If you're building a face recognition application, you might want to train a neural network that takes as input a picture like this and outputs the identity of the person in the picture. This image is 1,000 by 1,000 pixels. Its representation in the computer is actually as 1,000 by 1,000 grid, or also called 1,000 by 1,000 matrix of pixel intensity values.

In this example, my pixel intensity values or pixel brightness values, goes from 0-255 and so 197 here would be the brightness of the pixel in the very upper left of the image, 185 is brightness of the pixel, one pixel over, and so on down to 214 would be the lower right corner of this image.



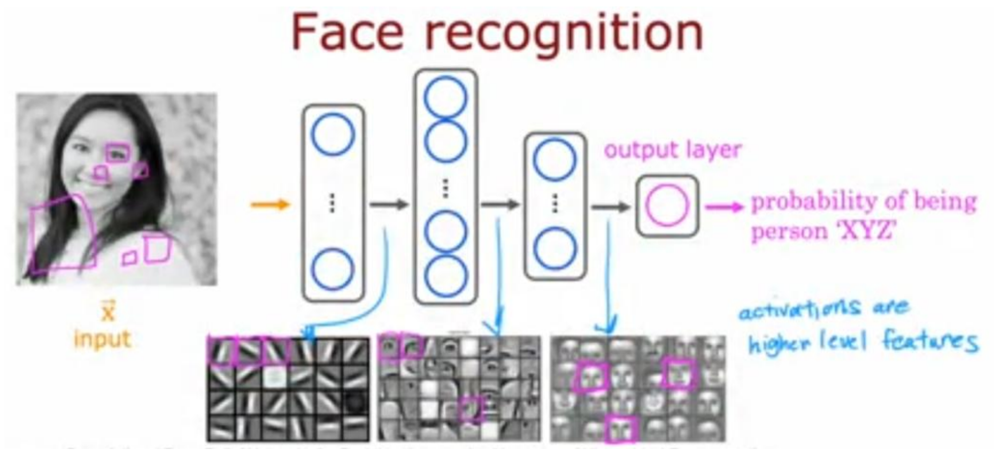
If you were to take these pixel intensity values and unroll them into a vector, you end up with a list or a vector of a million pixel intensity values. One million because 1,000 by 1,000 square gives you a million numbers. The face recognition problem is, can you train a neural network that takes as input a feature vector with a million pixel brightness values and outputs the identity of the person in the picture.

This is how you might build a neural network to carry out this task. The input image X is fed to this layer of neurons. This is the first hidden layer, which then extract some features. The output of this first hidden layer is fed to a second hidden layer and that output is fed to a third layer and then finally to the output layer, which then estimates, say the probability of this being a particular person.



One interesting thing would be if you look at a neural network that's been trained on a lot of images of faces and to try to visualize what are these hidden layers, trying to compute. It turns out that when you train a system like this on a lot of pictures of faces and you peer at the different neurons in the hidden layers to figure out what they may be computing this is what you might find. In the first hidden layer, you might find one neuron that is looking for the low vertical line or a vertical edge like that. A second neuron looking for a oriented line or oriented edge like that.

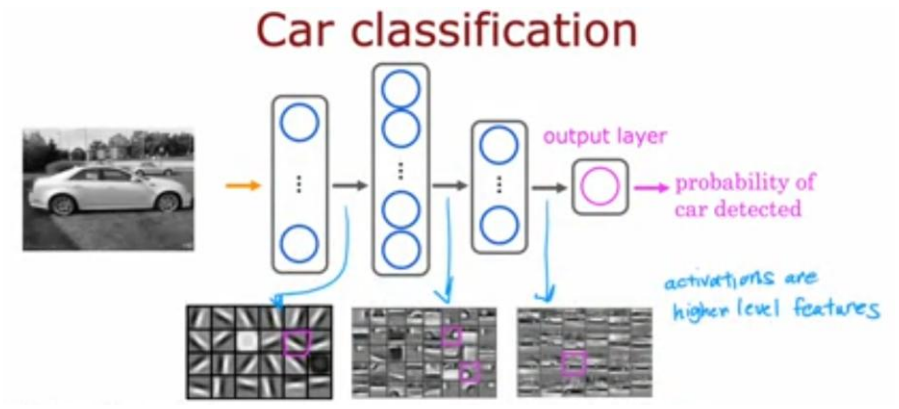
The third neuron looking for a line at that orientation, and so on. In the earliest layers of a neural network, you might find that the neurons are looking for very short lines or very short edges in the image. If you look at the next hidden layer, you find that these neurons might learn to group together lots of little short lines and little short edge segments in order to look for parts of faces.


Face recognition
$\vec{x}$
input
output layer
probability of being person 'XYZ'
activations are higher level features

For example, each of these little square boxes is a visualization of what that neuron is trying to detect. This first neuron looks like it's trying to detect the presence or absence of an eye in a certain position of the image. The second neuron, looks like it's trying to detect like a corner of a nose and maybe this neuron over here is trying to detect the bottom of an ear.

Then as you look at the next hidden layer in this example, the neural network is aggregating different parts of faces to then try to detect presence or absence of larger, coarser face shapes. Then finally, detecting how much the face corresponds to different face shapes creates a rich set of features that then helps the output layer try to determine the identity of the person picture.

A remarkable thing about the neural network is you can learn these feature detectors at the different hidden layers all by itself. In this example, no one ever told it to look for short little edges in the first layer, and eyes and noses and face parts in the second layer and then more complete face shapes at the third layer.


Car classification
output layer
probability of car detected
activations are higher level features

The neural network is able to figure out these things all by itself from data. Just one note, in this visualization, the neurons in the first hidden layer are shown looking at relatively small windows to look for these edges. In the second hidden layer is looking at bigger window, and the third hidden layer is looking at even bigger window.

These little neurons visualizations actually correspond to differently sized regions in the image. Just for fun, let's see what happens if you were to train this neural network on a different dataset, say on lots of pictures of cars, picture on the side. The same learning algorithm is asked to detect cars, will then learn edges in the first layer. Pretty similar but then they'll learn to detect parts of cars in the second hidden layer and then more complete car shapes in the third hidden layer.

Just by feeding it different data, the neural network automatically learns to detect very different features so as to try to make the predictions of car detection or person recognition or whether there's a particular given task that is trained on. That's how a neural network works for computer vision application. In fact, later this week, you'll see how you can build a neural network yourself and apply it to a handwritten digit recognition application.

So far we've been going over the description of intuitions of neural networks to give you a feel for how they work. In the next video, let's look more deeply into the concrete mathematics and a concrete implementation of details of how you actually build one or more layers of a neural network, and therefore how you can implement one of these things yourself.