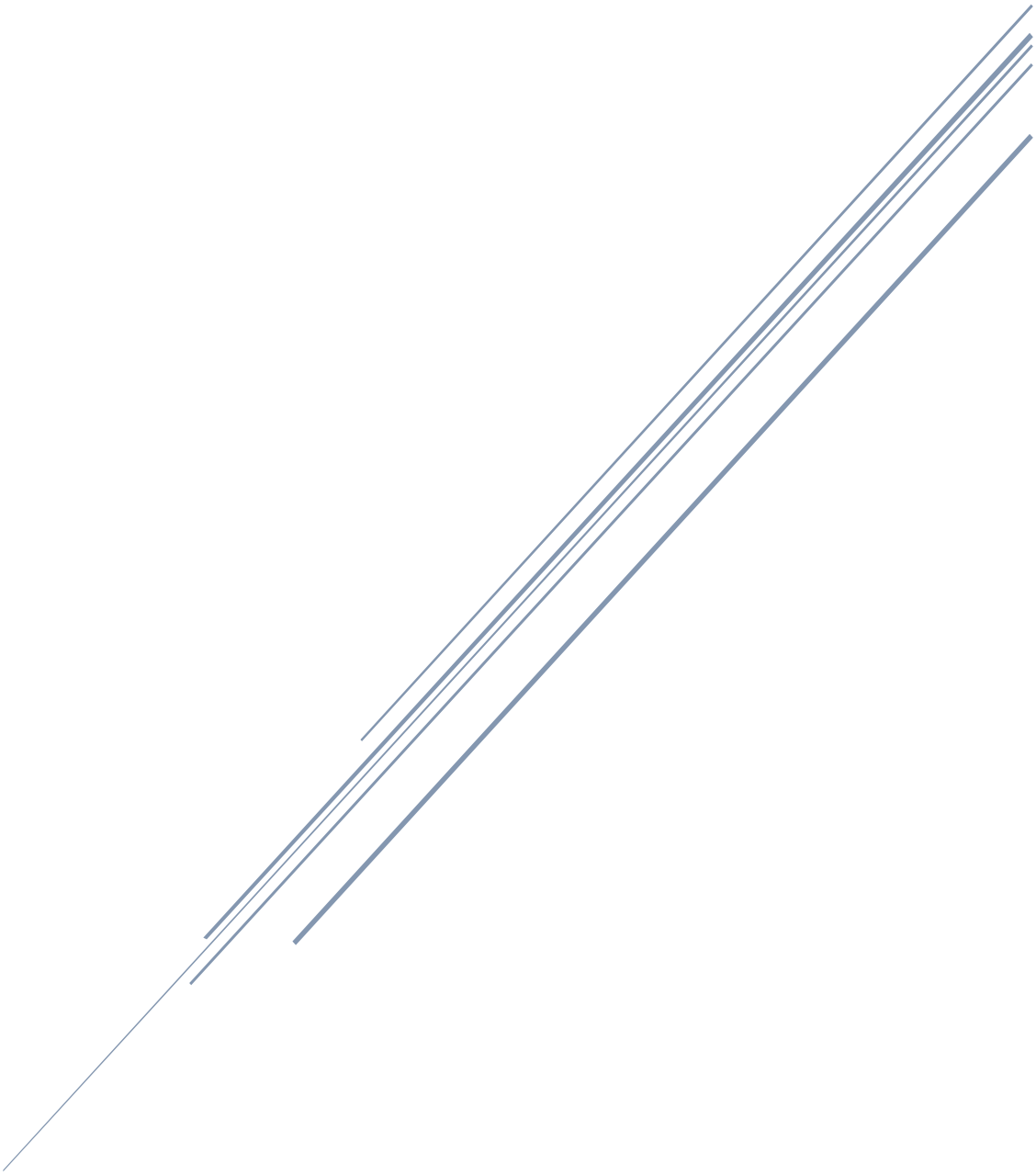
DEEP LEARNING MODELS

Module 4



v-cardona

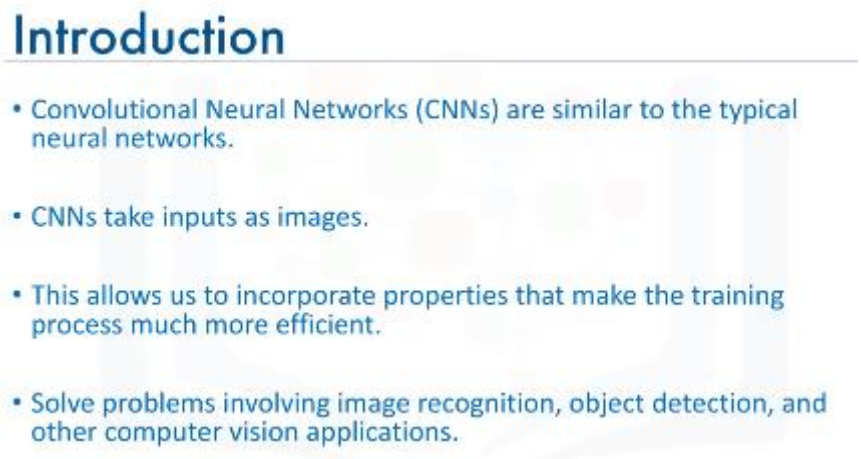
Deep Learning Fundamentals with Keras

Contenido

Shallow versus deep neural networks

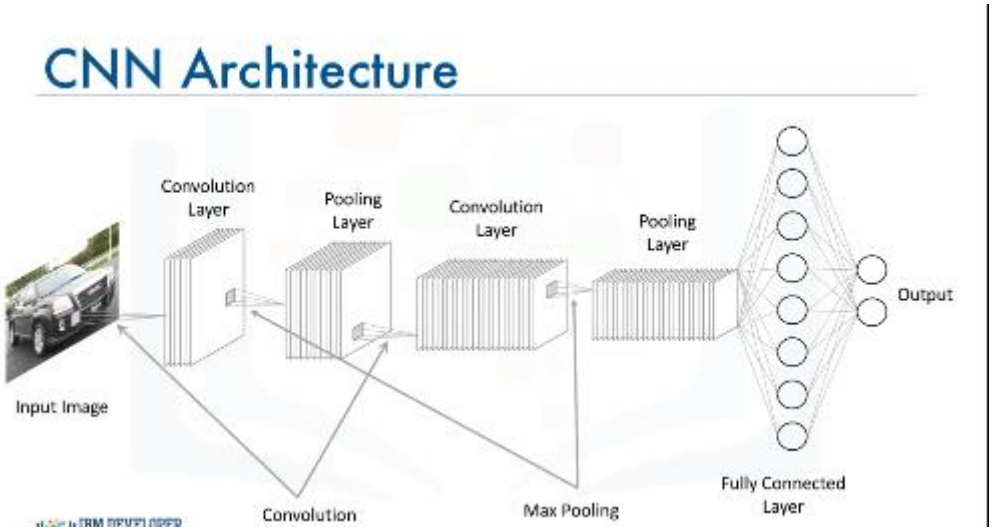
There is not really a consensus on the definition of a shallow neural network but a neural network with one hidden layer is considered a shallow neural network whereas a network with many hidden layers and a large number of neurons in each layer is considered a deep neural network.

Convolutional neural networks

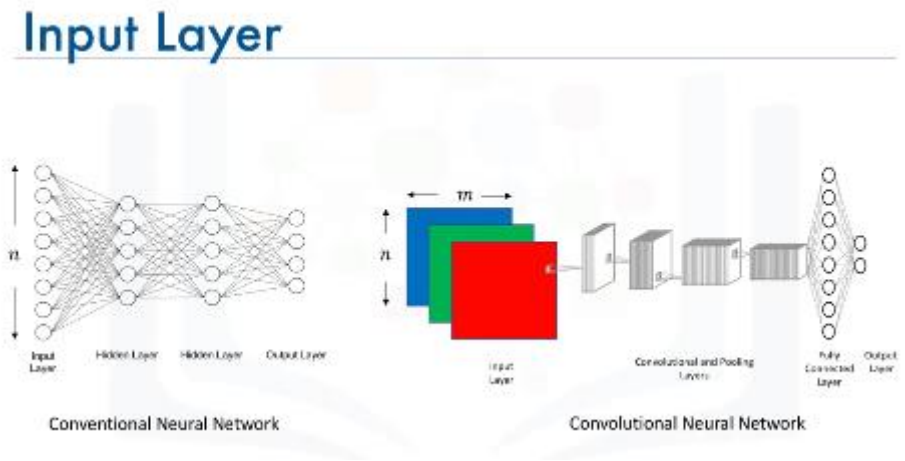


Convolutional neural networks, or CNNs for short, make the explicit assumption that the inputs are images, which allows us to incorporate certain properties into their architecture. These properties make the forward propagation step much more efficient and vastly reduces the amount of parameters in the network. Therefore, CNNs are best for solving problems related to image recognition, object detection, and other computer vision applications.

Architecture



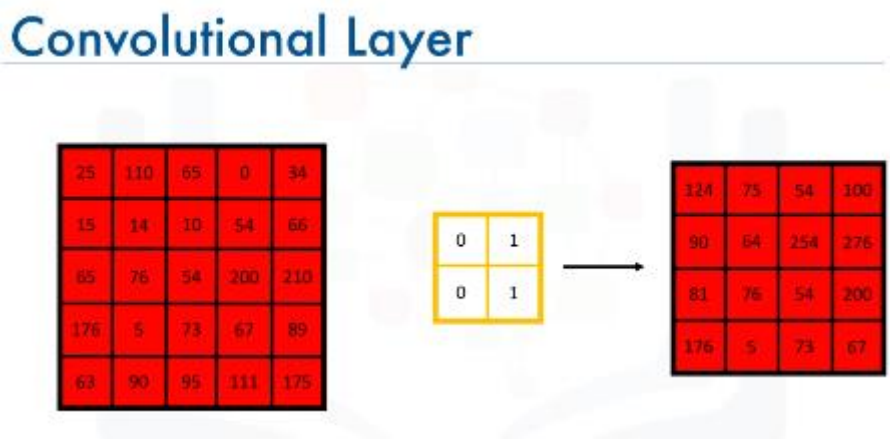
Input layer



Convolutional layer

In the convolutional layer, we basically define filters and we compute the convolution between the defined filters and each of the three images.

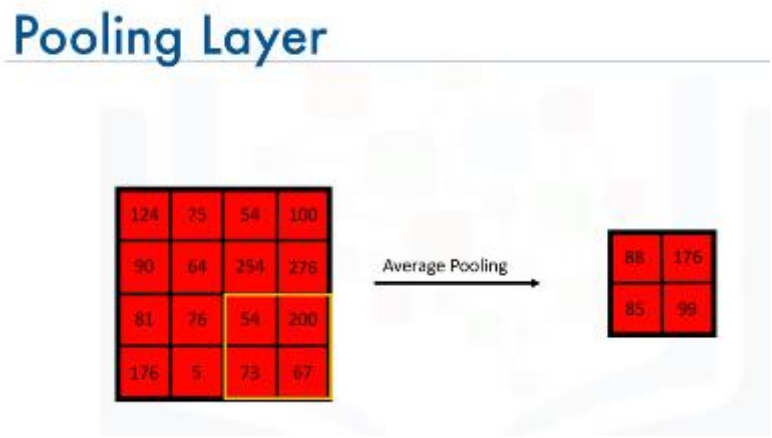
We start by sliding the filter over the image and computing the dot product between the filter and the overlapping pixel values and storing the result in the empty matrix. We repeat this step moving our filter one cell, or one stride. You can apply more than one filter. The more filters we use, the more we are able to preserve the spatial dimensions better.



Decreasing the number of parameters would definitely help in preventing the model from overfitting the training data, for that, convolutional networks are a good solution.

It is worth mentioning that a convolutional layer also consists of ReLU's which filter the output of the convolutional step passing only positive values and turning any negative values to 0.

Pooling layer



The pooling layer's main objective is to reduce the spatial dimensions of the data propagating through the network. There are two types of pooling that are widely used in convolutional neural networks. Max-pooling and average pooling. In max pooling which is the most common of the two, for each section of the image we scan we keep the highest value. With average pooling, we compute the average of each area we scan. In addition to reducing the dimension of the data, pooling, or max pooling in particular, provides spatial variance, which enables the neural network to recognize objects in an image even if the object does not exactly resemble the original object.

Connected layer

Finally, in the fully connected layer, we flatten the output of the last convolutional layer and connect every node of the current layer with every other node of the next layer. This layer basically takes as input the output from the preceding layer, whether it is a convolutional layer, ReLU, or pooling layer, and outputs an n-dimensional vector, where n is the number of classes pertaining to the problem at hand.

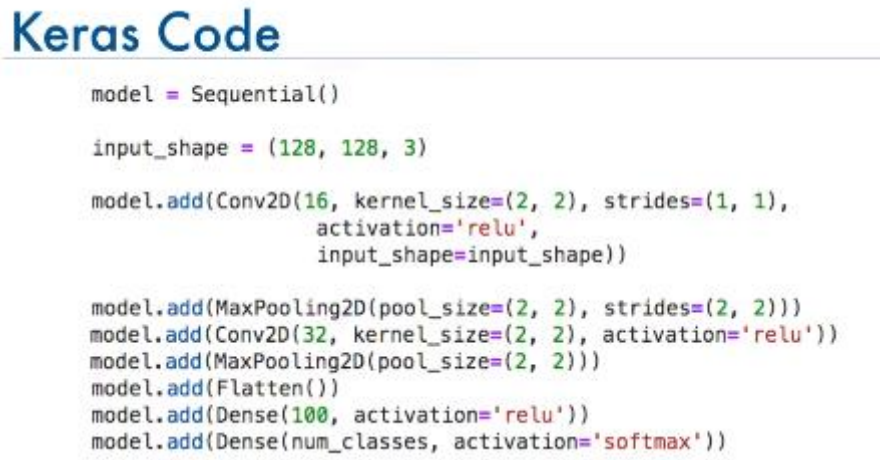
Keras CNN

We define our input to be the size of the input images.

We start with a convolutional layer, with 16 filters, each filter being of size 2x2 and slides through the image with a stride of magnitude 1 in the horizontal direction, and of magnitude 1 in the vertical direction. And the layer uses the ReLU activation function.

Then, we add a pooling layer and we're using max-pooling here with a filter or pooling size of 2 and the filter slides through the image with a stride of magnitude 2. Next, we add another set of convolutional and pooling layers. The only difference here is we are using more filters in the convolutional layer, actually twice as many filters as the first convolutional layer.

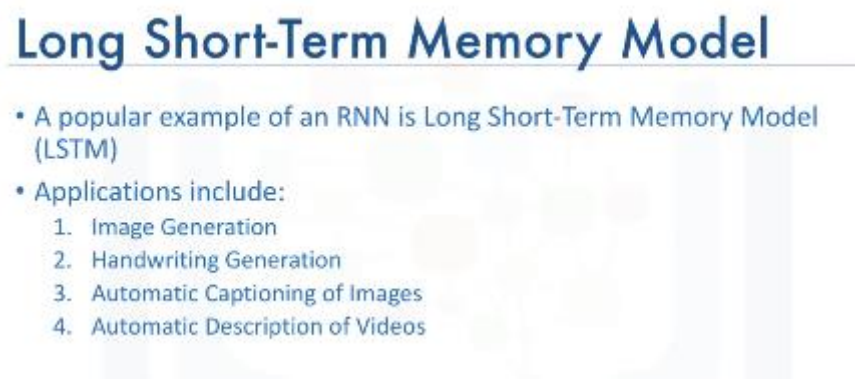
Finally, we flatten the output from these layers so that the data can proceed to the fully connected layers. We add another dense layer with 100 nodes and an output layer that has nodes equal to the number of classes in the problem at hand. And we use the softmax activation function in order to convert the outputs into probabilities.



Recurrent neural networks

Recurrent neural networks or (RNNs) for short, are networks with loops that don't just take a new input at a time, but also take in as input the output from the previous dat point that was fed into the network.

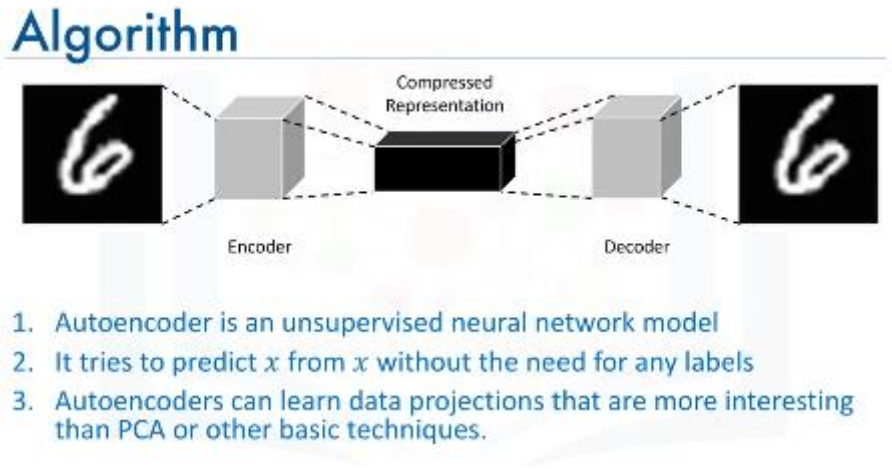
Long Short-Term Memory Model (LSTM)



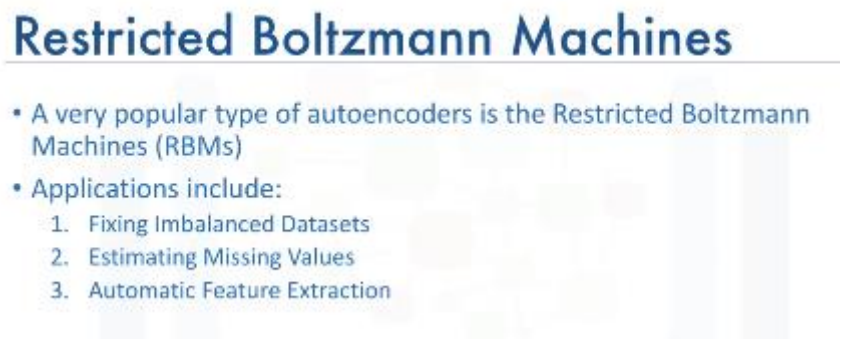
Autoencoders

Autoencoding is a data compression algorithm where the compression and the decompression functions are learned automatically from data. instead of being engineered by a human. Such autoencoders are built using neural networks. Autoencoders are data specific, which means that they will only be able to compress data similar to what they have been trained on. Therefore, an autoencoder trained on pictures of cars would do a rather poor job of compressing pictures of buildings, because the features it would learn would be vehicle or car specific. Some interesting applications of autoencoders are data denoising and dimensionality reduction for data visualization.

It uses backpropagation by setting the target variable to be the same as the input. In other words, it tries to learn an approximation of an identity function. Because of non-linear activation functions in neural networks, autoencoders can learn data projections that are more interesting than a principal component analysis PCA or other basic techniques, which can handle only linear transformations.



A very popular type of autoencoders is the Restricted Boltzmann Machines or (RBMs) for short.



Supervised networks: Convolution, Recurrent, LSTM

Unsupervise networks: Autoencoders, RBM