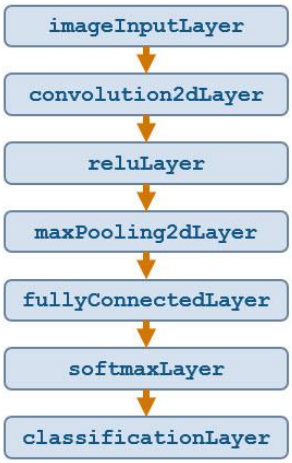
Convolutional Network Layers

As well as traditional fully connected layers described in the previous video, convolutional neural networks also use layers specific to image applications.

CNNs vary in architecture and depth. For example, AlexNet contains 25 layers and GoogLeNet contains 144. However, all CNNs contain some common layers like an image input layer and convolution layers.

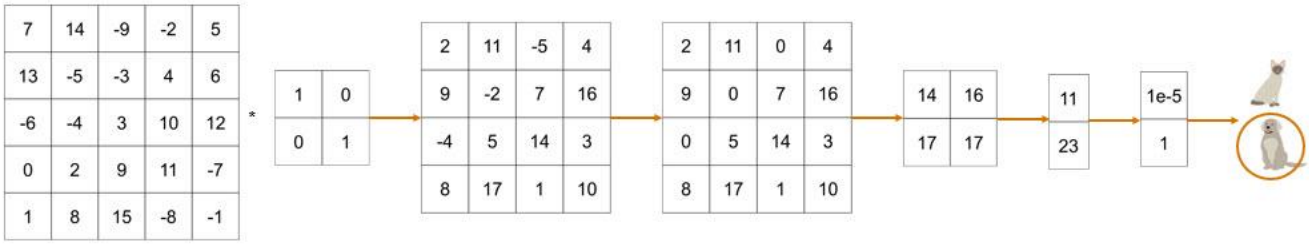
Below you can see how each layer in the land cover classification network would process a simple example image.

1.



The land cover classification network consists of a simple deep network architecture.

2.



Each layer in the network performs some operation on its inputs and outputs a new value.

3.

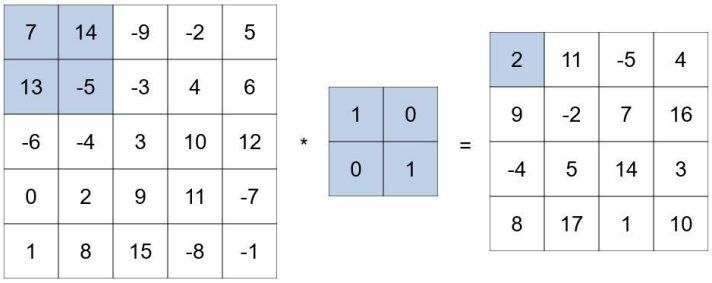
7	14	-9	-2	5
13	-5	-3	4	6
-6	-4	3	10	12
0	2	9	11	-7
1	8	15	-8	-1

imageInputLayer([5 5])

The first layer is an image input layer. This layer defines the input size of the network and normalizes the input images.

By default, an image input layer subtracts the mean image of the training data set. This centers the images around zero.

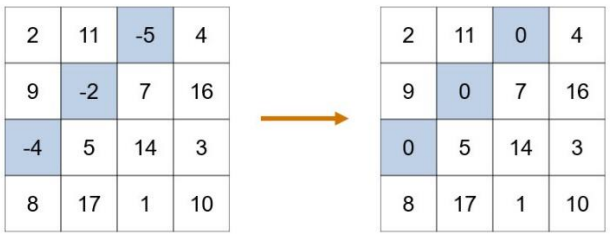
4.



convolution2dLayer([2 2],1)

2-D convolution layers apply sliding filters to the input image. Convolution layers are a key part of the CNN architecture. They rely on the spatial structure of the input image.

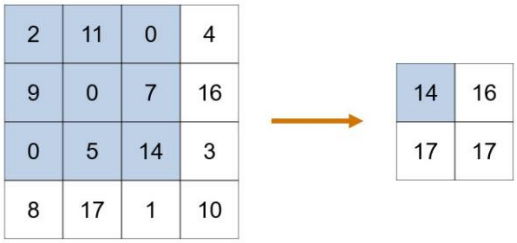
5.



reluLayer()

Convolution layers are usually followed by a nonlinear activation layer such as a rectified linear unit (ReLU). A ReLU layer performs a threshold operation to each element of the input. Any value less than zero is set to zero.

6.

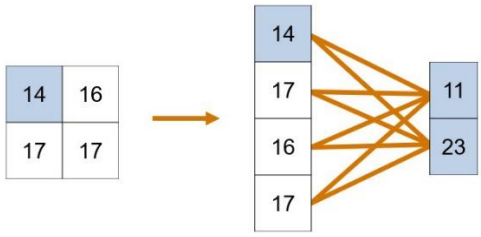


maxPoolingLayer([3 3])

A maximum pooling layer performs down-sampling by dividing the input into rectangular pooling regions and computing the maximum of each region.

Pooling reduces the network complexity and creates a more general network.

7.



fullyConnectedLayer(2)

Features passing through the network are stored in a collection of matrices until they reach the fully connected layer. At the fully connected layer, the input is "flattened" so that it can be mapped to the output classes. This layer is a classical neural network.

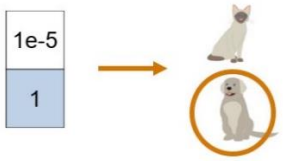
The output size for this layer is the number of classes for your classification problem. For example, if you were classifying cats and dogs, the output size would be two.

8.

softmaxLayer()

The softmax layer converts the values for each output class into normalized scores using a normalized exponential function. You can interpret each value as the probability that the input image belongs to each class.

9.



classificationLayer()

The classification output layer returns the name of the most likely class.