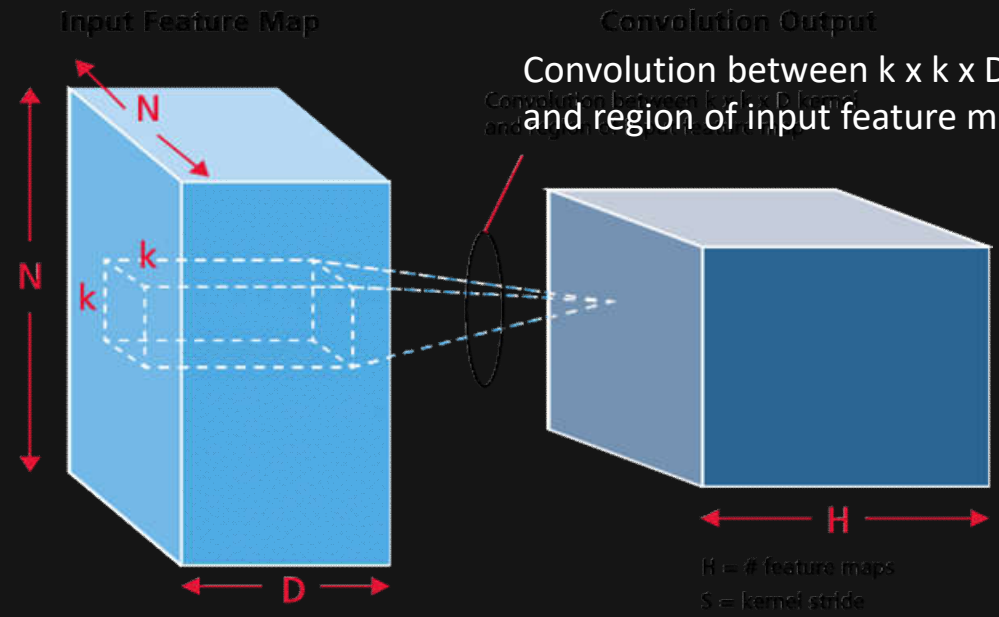
CNN Layers

Convolutional Layers



N = input height and width
k = kernel height and width
D = input depth

H = # feature maps

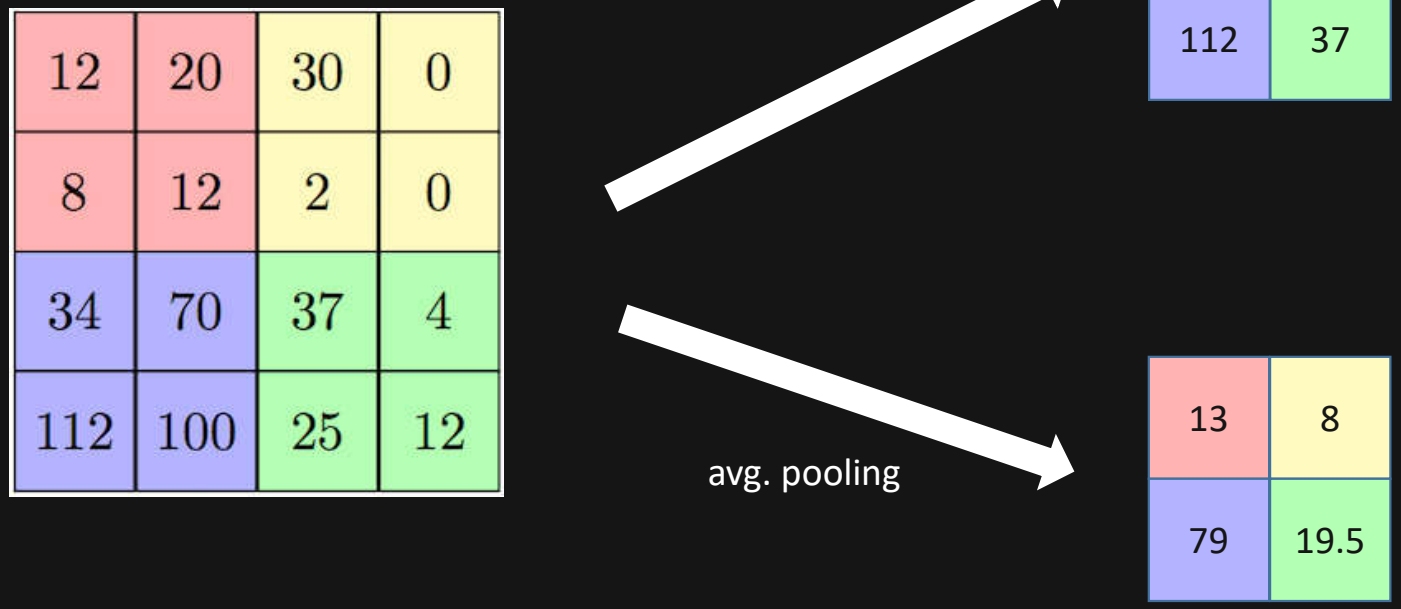
Image source (link)

Convolutional Layer

- Performs filtering on the image or result from the previous layers
- Saves weights by using the same weights for all filters of a feature map
- Multiple filters can be applied to the result of the previous layer resulting in multiple feature maps

- Params:
  - nxn: the filter size (typically 3x3, 5x5, 7x7). Can be 1x1!
  - depth: the number of feature maps
  - stride: shift to apply next filter (1 applies to every pixel, 2 skips a row/column, …)
  - padding: can be used to create a border of zeros around the image(s) from the previous layer (preserves dimensionality)

Pooling


12 20 30 0
8 12 2 0
34 70 37 4
112 100 25 12
max pooling
20 30
112 37
avg. pooling
13 8
79 19.5

Pooling

- Enables dimensionality reduction
- Typical kernel = 2x2 => reduction by half in each dimension
- Typical stride for kernel 2x2 = 2 (no overlap)
- Most common is max pooling

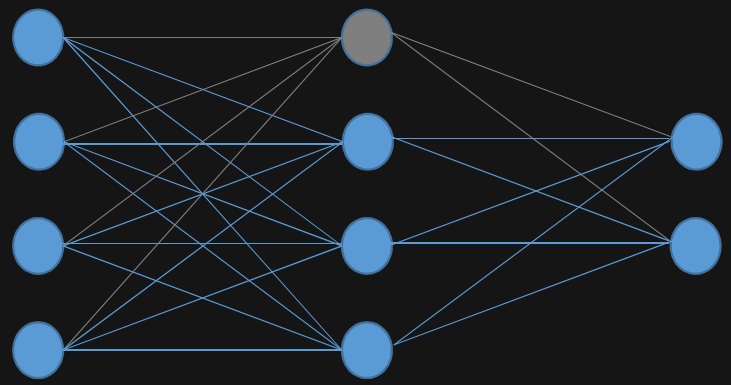
- Considering an input of 28 x 28 x 256 the output will be 14 x 14 x 256

- Pooling layer is commonly placed after a convolutional layer

- There is no consensus about pooling. Some prefer to use Convolutional layer with stride: STRIVING FOR SIMPLICITY: THE ALL CONVOLUTIONAL NET (link)

Dropout

- Randomly selects neurons to discard during feedforward and backward passes.



Droupout

- Srivastava, et al. (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting

- Should provide better generalization, as it is forcing the network to have multiple paths to achieve the same result.

Regularization

- Penalization for large weights
- CNNs with smaller weights tend to generalize better.

In Keras

```
model = Sequential()

model.add(Conv2D(64, (5, 5), padding='same',
                 input_shape=(imgSize, imgSize, channels),
                 activation='relu'))
model.add(Conv2D(64, (5, 5), activation='relu') )
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu') )
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(classCount, activation='softmax‘))
```