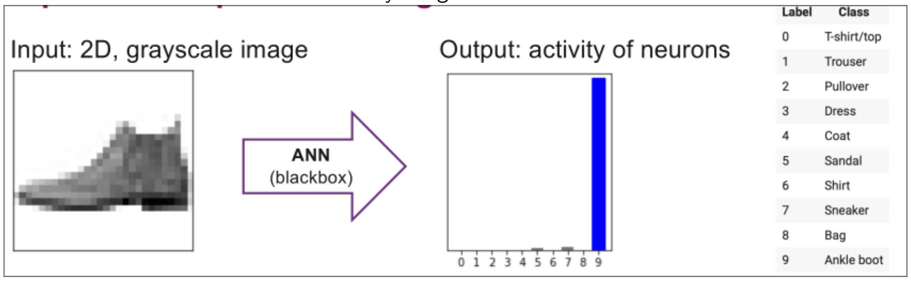
0 Contents

1 CNN Convolutional Neural Networks

1.1 Keras

A python library that wraps tensorflow for classification.
We will use this in this module to classify images like so:



Label	Class
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

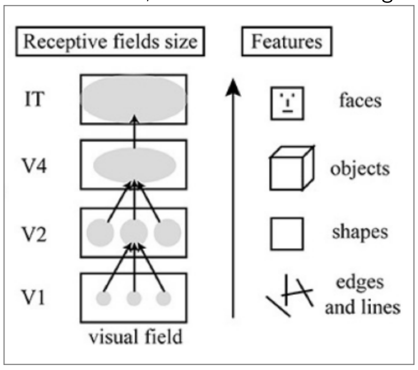
1.2 Flattening

When we convert an image into a long vector, we lose information in the human sense, or rather make it hidden.
todo, explain what is hidden and why

2 Convolution

2.1 Firing of neurons

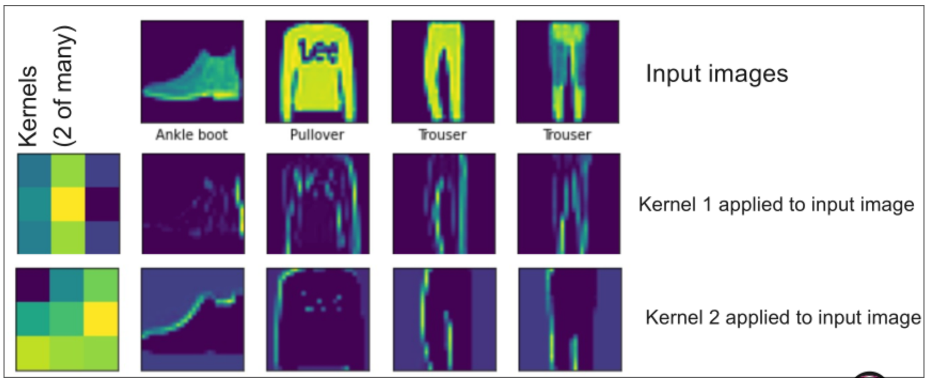
Neurons are clearly connected to something very specific, this would then also be reflected in the artificial neural network. In other words, neuron 1 handles horizontal lines, another a line with a slight angle and so-on.



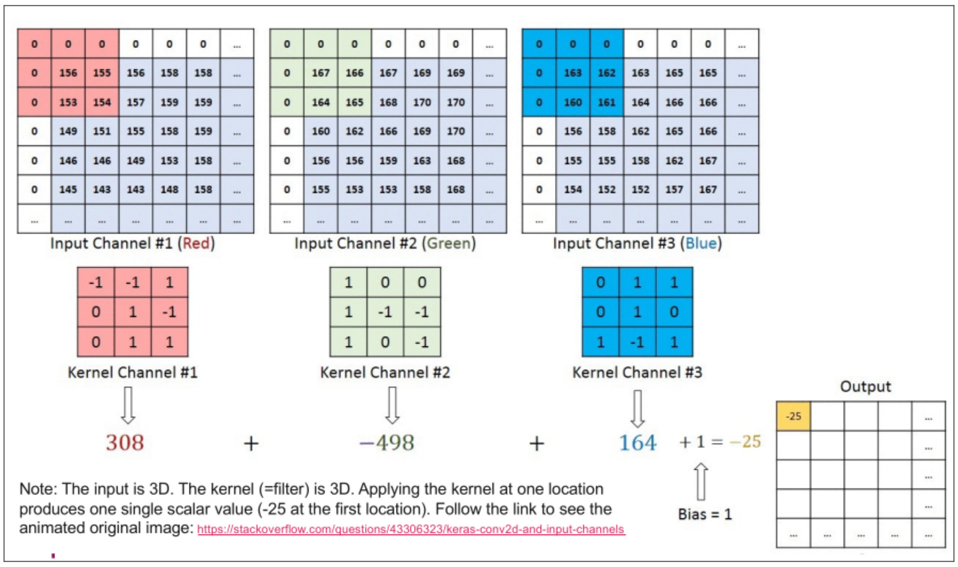
2.2 mathematical model of a feature detector

- Two Inputs
  - a picture
    Note that rgb would give you 3 channels red, green and blue
  - A filter(kernel)
    * an m by n matrix in the simplest case (1 channel, grayscale).
    * an m by n x 3 "stack of matrices" in the case of a 3 channel input (e.g. an RGB image)
    * an m by n x d "stack of matrices". The depth of the kernel must equal the number of input channels.
- One Output
  A feature map (where is the thing that we wanted to search / determine by)
  One convolution produces one feature map. Even if the input and the filter have multiple channels, the output of the convolution has one channel.
- The Operation: Convolution
  We convolve the input image with the convolutional kernel

2.3 Example of convolution



As you can see here, a filter will be used to detect something specific, like a pattern.
This means that you will be combining multiple different filters in order to properly figure out what picture the underlying image is composed of.



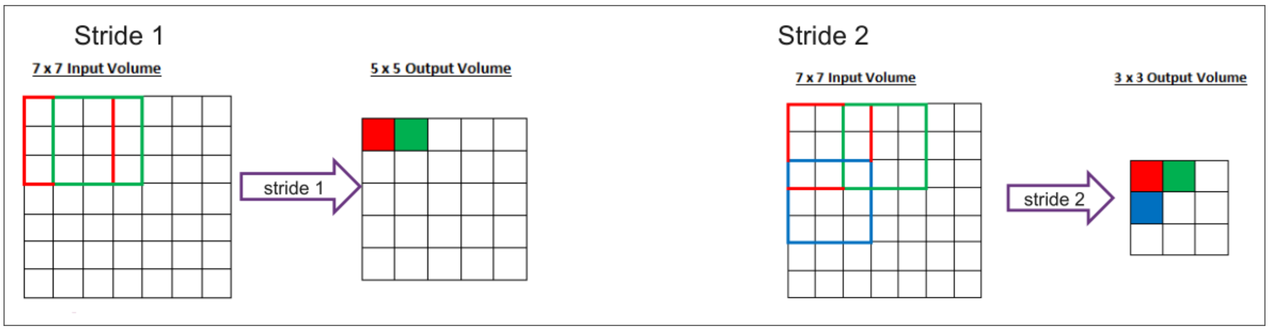
Note: The input is 3D. The kernel (=filter) is 3D. Applying the kernel at one location produces one single scalar value (-25 at the first location). Follow the link to see the animated original image: https://stackoverflow.com/questions/43306323/keras-conv2d-and-input-channels

Explanation for the first calculation: 0 * -1 + 0 * -1 + 0 * 1 + 0 * 0 + 156 * 1 + 155 * -1 + 0 * 0 + 153 * 1 + 154 * 1 = 308
multiply each number in the top red square with the number in the same position in the bottom square. We then proceed to do this for all channels (complexity of input, 3 for rgb), which *will then be combined to 1 single output value*.
This output value will then also be combined with a *bias*.
The entire reason we do this, is so that we can have an easier time calculating the images with a pc.

2.4 Reasons for convolution

- Features can be detected independent of location -> filters will always find what they are supposed/created to find
- This calculation is done in parallel, which is very fast for gpus when doing matrix calculations! -> Hence the use of tensorflow with cuda!
- Shared weights mean using the same *kernel values*, this reduces the use of a singular value for each neuron. -> more processing etc

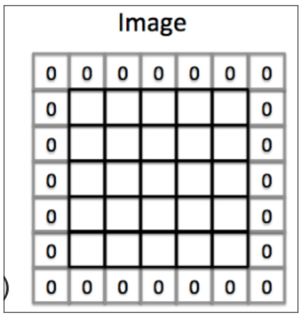
2.5 Stride



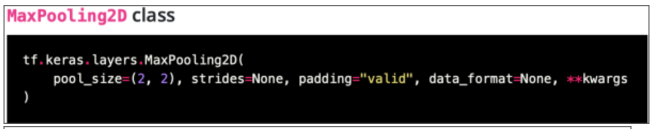
Stride is simply the offset by which we move towards the right and the bottom when we move to the next calculation.
The default value here is 1, which means Stride 1.

2.6 Padding

Padding is simply the edge of the matrix that has been predefined.
A same padding is made entirely of 0's!

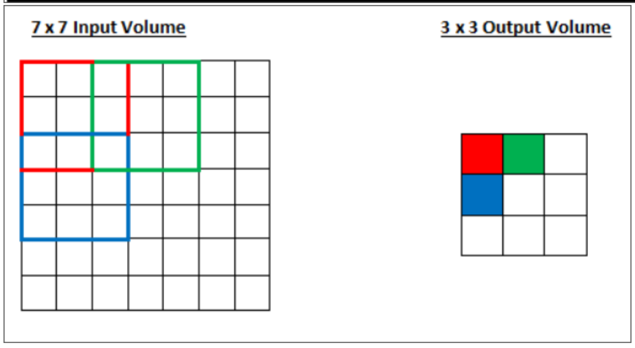


2.7 Max Pooling



MaxPooling2D class

```
tf.keras.layers.MaxPooling2D(
    pool_size=(2, 2), strides=None, padding="valid", data_format=None, **kwargs
)
```



In this case we only take the *max* value instead of some sum.
This leaves you with rgb again instead of a mixed value!!

2.8 Example with Keras

```
model = Sequential([
  layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```