Convolutional Neural Networks

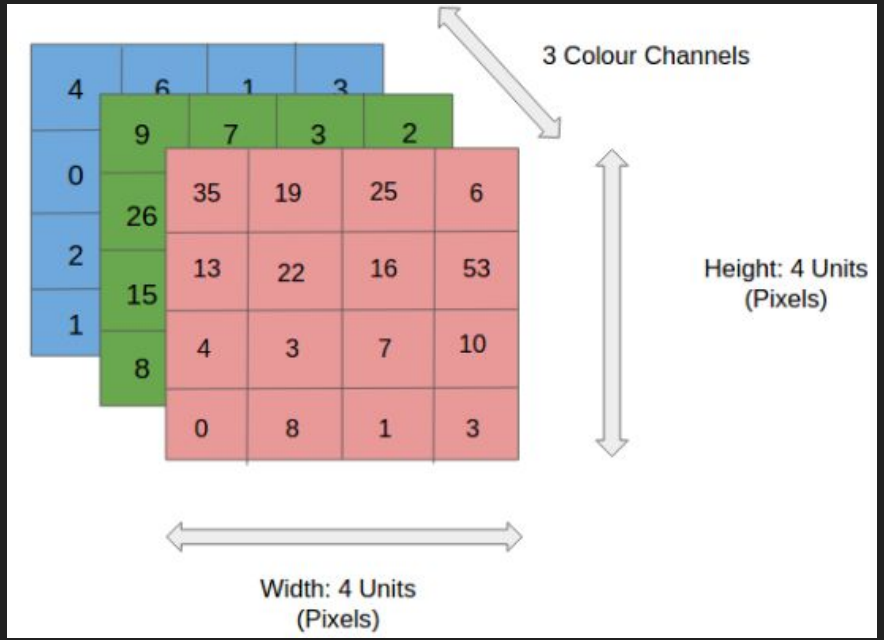
Areeb Gani, Michael Ilie, Vijay Shanmugam

Welcome!

ml.mbhs.edu

Motivation

- Often, with neural networks, we have to deal with images
- Images have lots of data, as they are represented as multiple 2-dimensional arrays

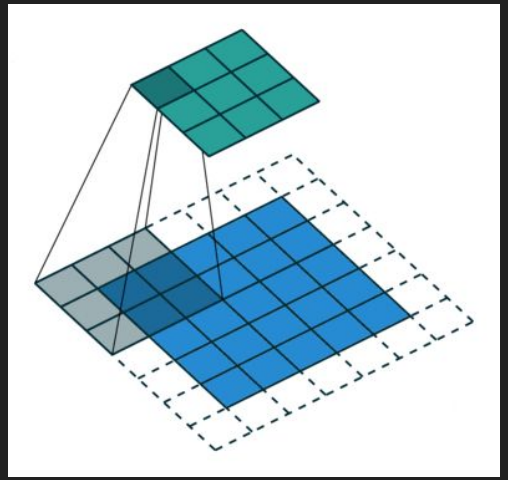


Motivation

- We want to “simplify” the image, keeping its core features, while making it easier for our model to process all the data and perform the normal computations
- We can achieve this using some special functions which we call **kernels**

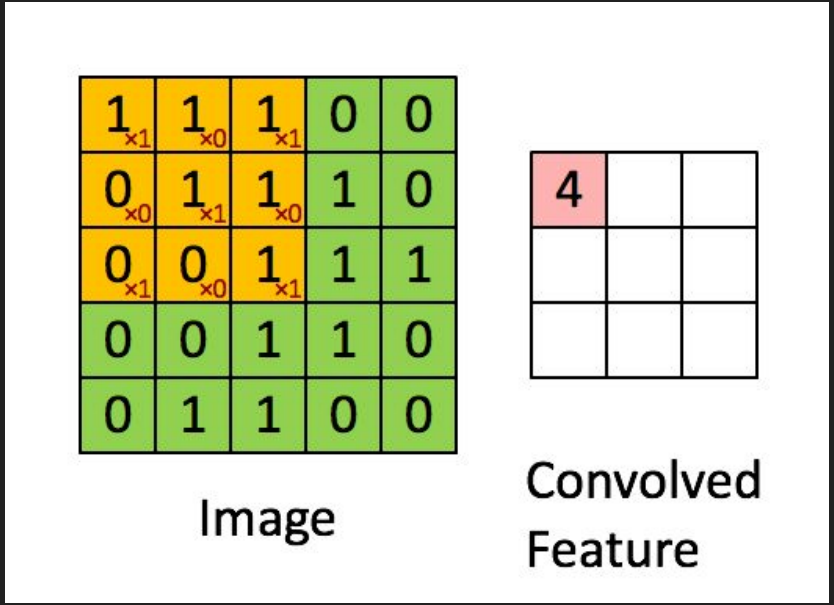
Convolutional Kernel

- “Sliding window,” called **filter**, applies transformation on certain parts of the image sequentially
- Returns a smaller array than original image, but still retains high-level features (the *gist* of the image)



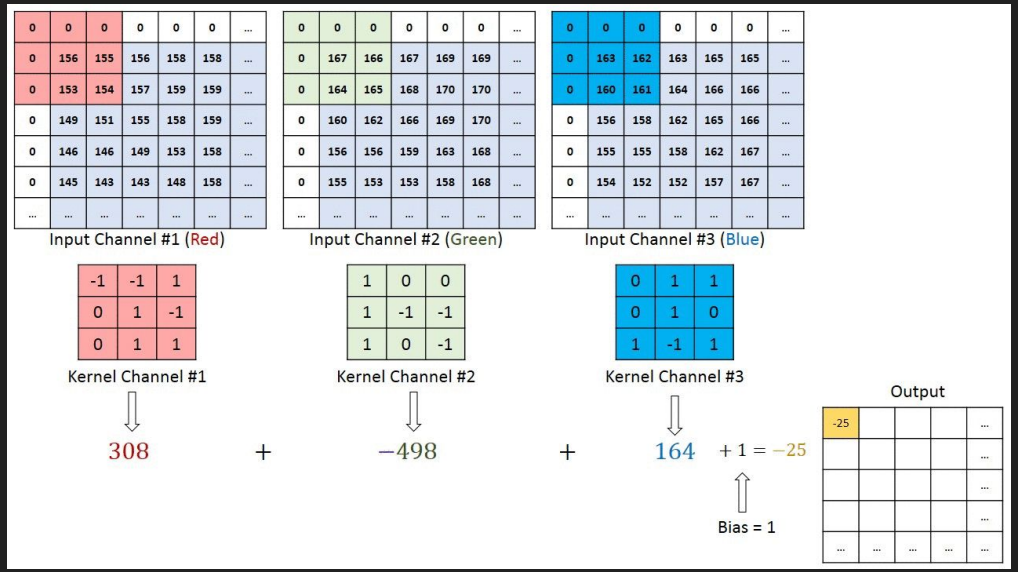
Convolutional Kernel

- Mathematically, our filter is just a matrix - we multiply the element of the image array with the elements of our matrix to get our output



Convolutional Kernel

- For higher dimensions, we add together the output of each filter in each color channel to our final output matrix

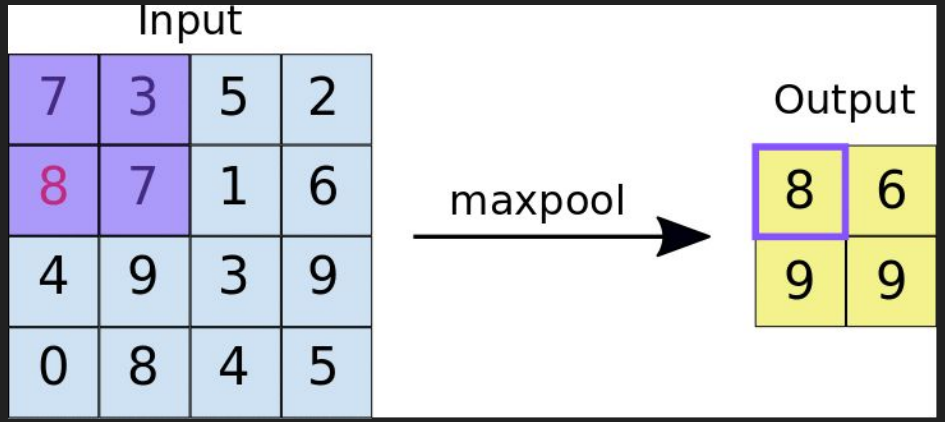


Pooling Layer

- After applying the convolution filter, we run our output matrix through a **pooling layer**
- There are different types of pooling layers, but the most common is called “maximum pooling,” which is what we’ll focus on

Pooling Layer

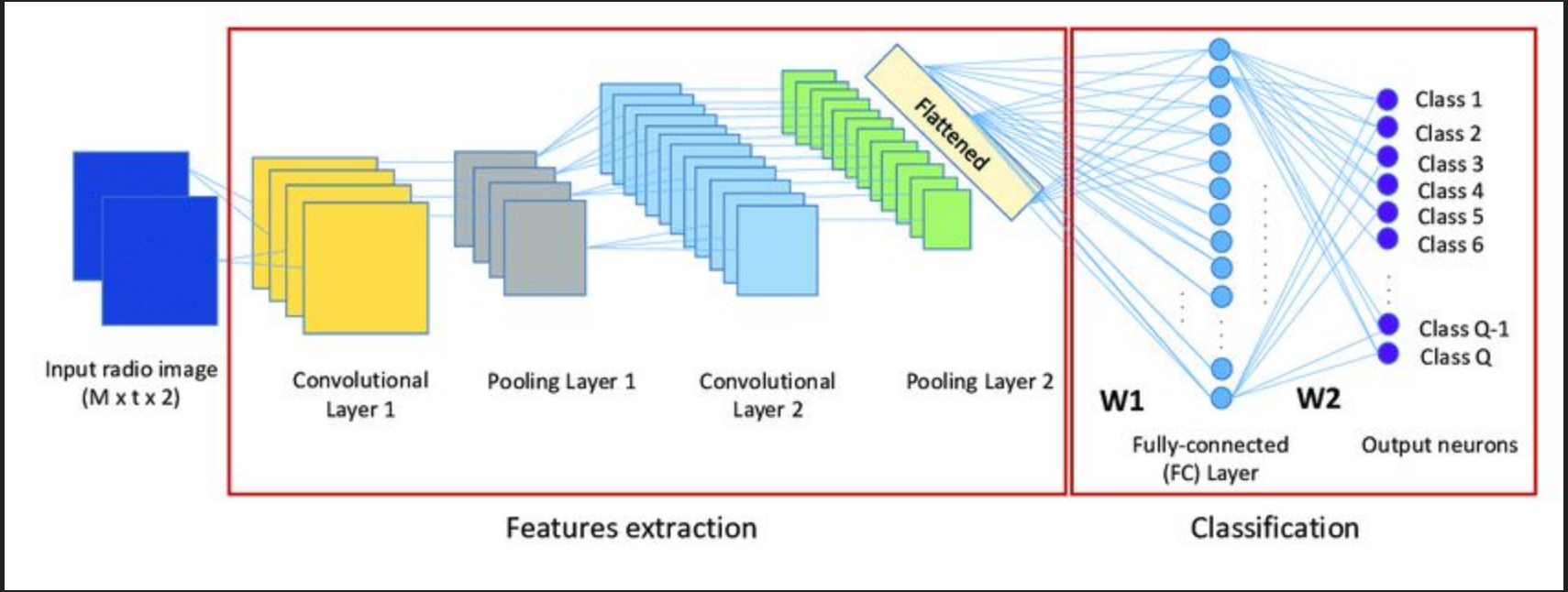
- Max pooling has a similar “sliding window” process to convolutional kernels, but instead of multiplying by a matrix, we simply take the maximum value of each window in our output
- This will find the most dominant features in the image, which is what we need to make our prediction



Fully Connected Layer and Output

- We apply the previous two steps (Convolutional Kernel and Pooling Layer) repeatedly, to minimize the size of the image, for as long as we wish
- Then we use a **fully connected layer** to get our output - this is just a standard neural network layer like we have discussed before
- Finally, we get our output using an activation function such as *sigmoid* or *softmax*

Full Model


Input radio image
(M x t x 2)
Convolutional
Layer 1
Pooling Layer 1
Convolutional
Layer 2
Pooling Layer 2
Flattened
W1
W2
Fully-connected
(FC) Layer
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class Q-1
Class Q
Output neurons
Features extraction
Classification

Some Practice Tutorials

- https://www.tensorflow.org/tutorials/images/cnn (TensorFlow)
- https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html (PyTorch)
- https://www.analyticsvidhya.com/blog/2019/10/building-image-classification-models-cnn-pytorch/ (also PyTorch)

Join Our Groups

- Sign up for Discord (https://discord.gg/3Z5YuPqt)
- Join Deepnote (https://deepnote.com/join-team?token=af3af0284bc8497)
- Fill out our form (https://forms.gle/Fr31aFLWx8cHdtTY8)
    - Join mailing list + Github organization