Chapter 10

Convolutional Neural Networks

Convolutional neural networks

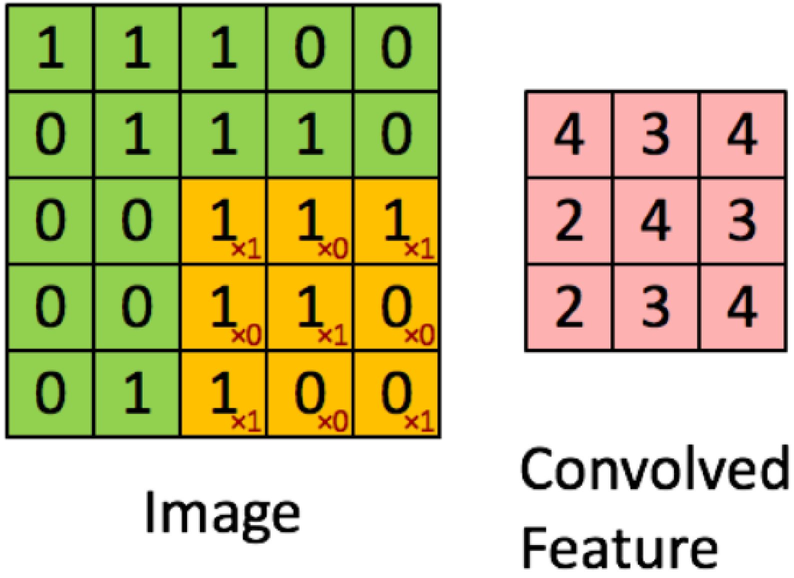
- Special type of Deep Neural Network mainly used for analyzing visual data
- The network works by extracting a set of high-level features
- Multilayer Perceptrons use are fully connected networks, which makes them prone to overfitting. Here the connections between layers are regularized in a hierarchical manner
- Have at least 1 layer that uses the mathematical convolution operation
- Convolution gives us the ability to learn spatial and temporal structures in the data
- Abbreviated as CNN or ConvNet
- Huben and Wiesel found in 1962 that the Visual Cortex of the brain works in a similar way

Convolutional neural networks

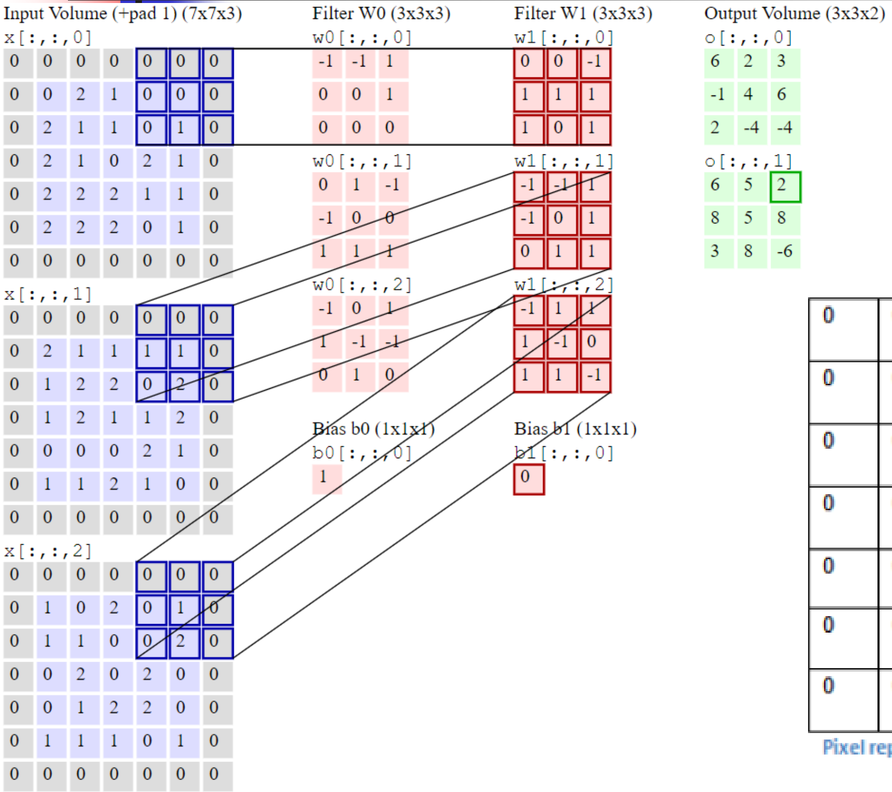
- Types of layers in CNNs:
  - **Convolutional**: Abstracts the images into feature maps (also called **neurons** or **kernels**). Number of feature maps and their size are hyper parameters of the model.
  - **Pooling**: Reduces the dimensions of the data by combining outputs of neurons (also called **downsampling**). Combination can be done by taking the average or the max value of set of neurons.
  - **Dropout**: Randomly sets the output of neurons to zero for each training sample. This forces the network to be redundant and avoid overfitting.
  - **Flatten**: Flattens the higher dimensional data into a single vector.
  - **Fully connected**: A layer where all neurons connect to every neuron from previous layer. Mostly used as the last layer to connect the extracted features and output the predicted labels.

Convolutional Layer

- Creates feature maps from input data using filters
- A convolutional layer can have many filters
- Each filter strides through the entire image doing matrix multiplication to create a feature map
- Stride step size can be controlled to reduce output spatial size
- We can add zero-padding to the border around the input data to control spatial size of the convolved feature
- Output size of a filter: $$\frac{W - K + 2P}{S} + 1$$
  - W is the input size
  - K is the kernel size
  - P is the padding size
  - S is the stride step size

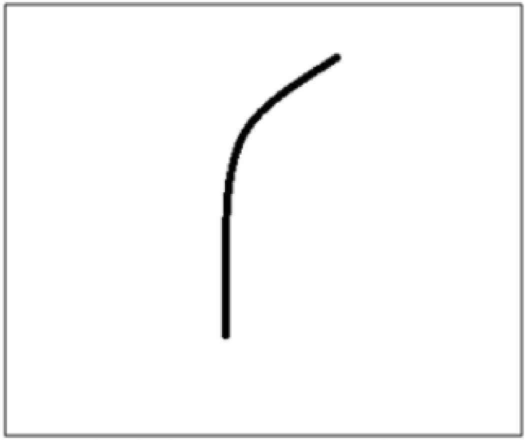


Convolutional Layer



0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

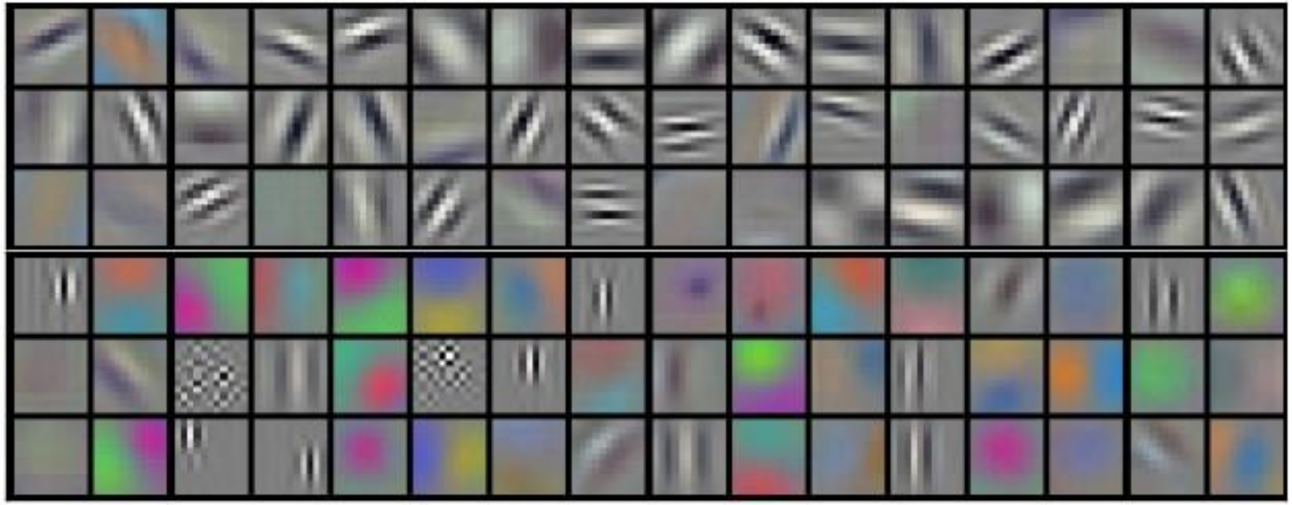
Pixel representation of filter



Visualization of a curve detector filter

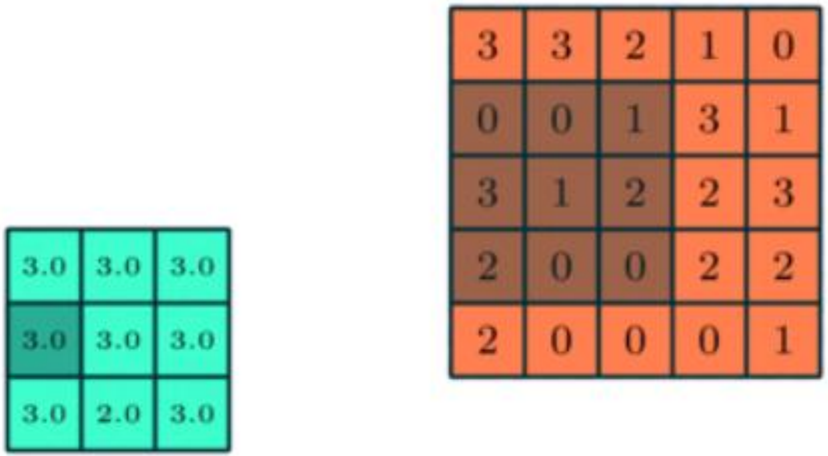
Convolutional Layer

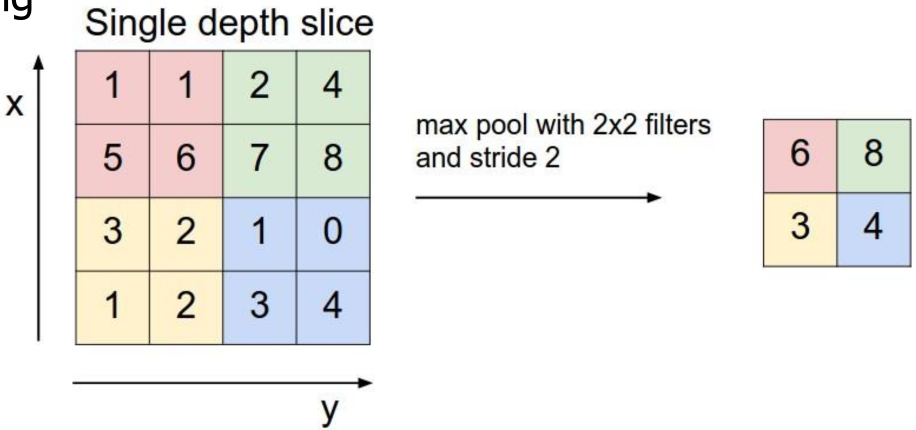
- We can visualize the learned filters
- https://youtu.be/AgkfIQ4IGaM



Pooling Layer

- Dimensionality reduction layer
- A pooling layer also has size, stride and padding just like a convolutional layer
- Thus output size calculation is the same as with convolutional layer
- The output of the pool can be determined using maximum value of the pool or average value





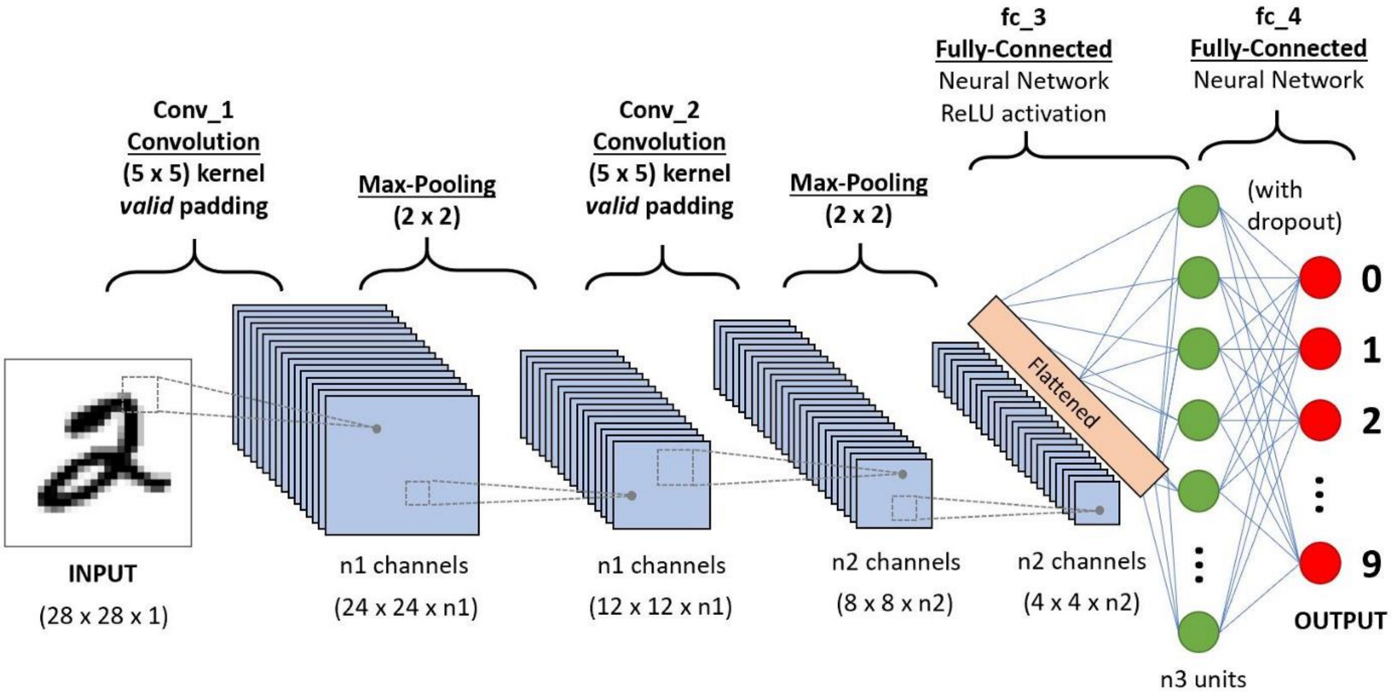
Convolutional neural networks

- As with Multi-layer Perceptrons, we also use activation functions here. The most commonly used are:
  - ReLU – used on convolutional layers
  - Softmax – used on the output layer for single label multiclass classification tasks. Normalizes the output vector, so that the sum of probabilities add up to 1
  - Sigmoid – used on the output layer for binary classification tasks and multilabel multiclass classification tasks
  - Linear – used on the output layer for regression tasks
- Training is also done via backpropagation. Main steps:
  1. Forward pass – Feeding an input sample to the CNN and getting the output
  2. Loss function – calculate the error gradient of our output
  3. Backward pass – Go backwards from the output through the network and update the weights according to their contribution for the error

Convolutional neural networks

- Many training **optimizers** have been created in the past years
- The main issue is trying to avoid local optimums and converge towards the global optimum in a timely manner
- Most common optimizers:
  - Stohastic Gradient Descent – flat learning rate
  - Adam – Adaptive learning rate
- We have various **loss functions** for calculating the error of the network. Most common loss functions:
  - Mean Squared Error – used for regression tasks
  - Binary Cross Entropy – used for binary classification tasks and multilabel multiclass classification tasks
  - Categorical Cross Entropy – used for single label multiclass classification tasks

Example CNN Architecture



References, Literature, further reading

- https://www.deeplearningbook.org/
- https://cs231n.github.io/convolutional-networks/
- https://towardsdatascience.com/neural-network-optimization-7ca72d4db3e0
- https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
- https://www.pyimagesearch.com/2019/01/21/regression-with-keras/
- https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/
- https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

Questions?