PyCon India

Hyderabad 2018

AI Assistance in Medical Imaging using PyTorch

Abhishek Kumar

Deep Learning Engineer @Predible Health

Aditya Bagari

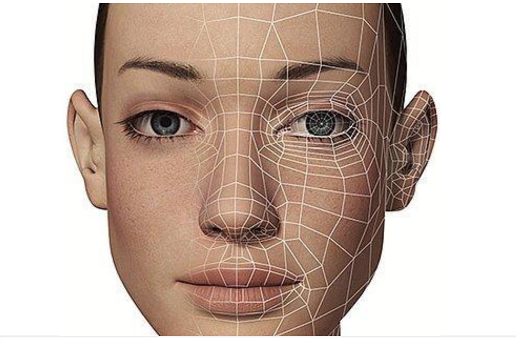
Undergraduate @IITMadras

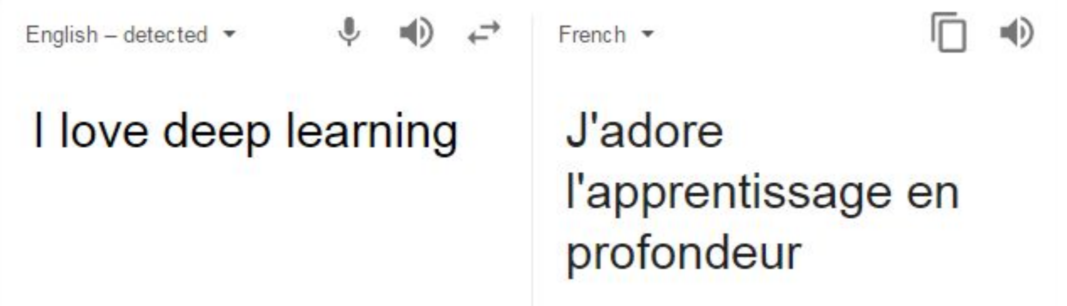
Deep learning is everywhere!

- Presence in:
  - Image Recognition
  - Voice Recognition
  - Machine Translation
  - Self-driving cars etc..

- Artificial Neural Networks are the engines

- Inspired by biological neural networks

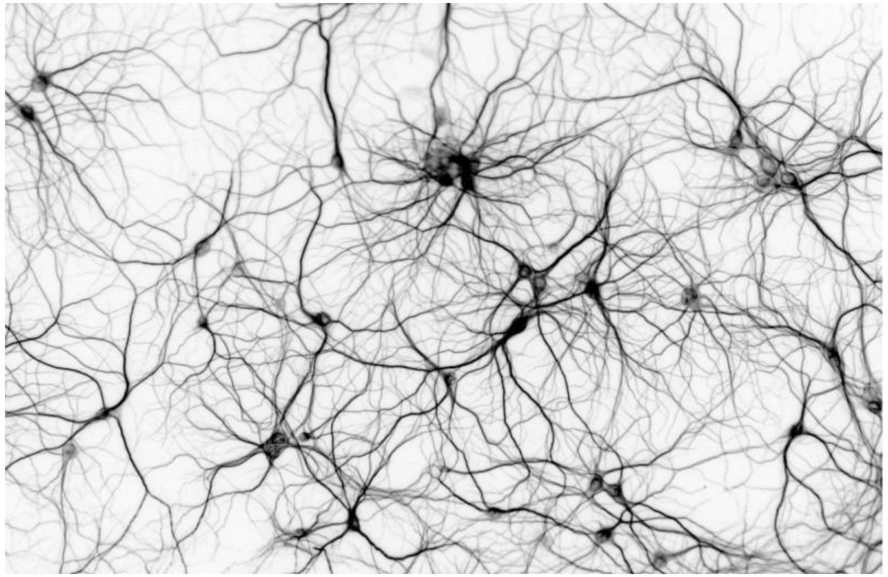


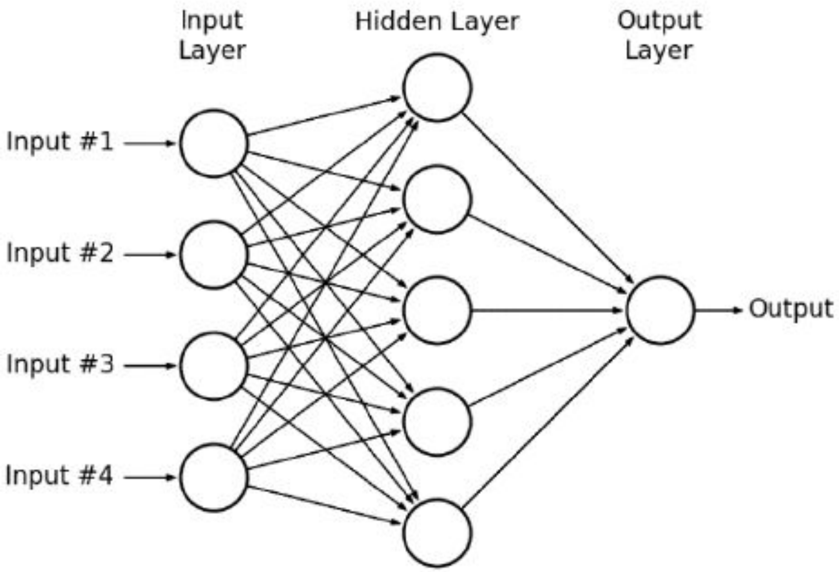


I love deep learning

J'adore l'apprentissage en profondeur

What is a neural network?





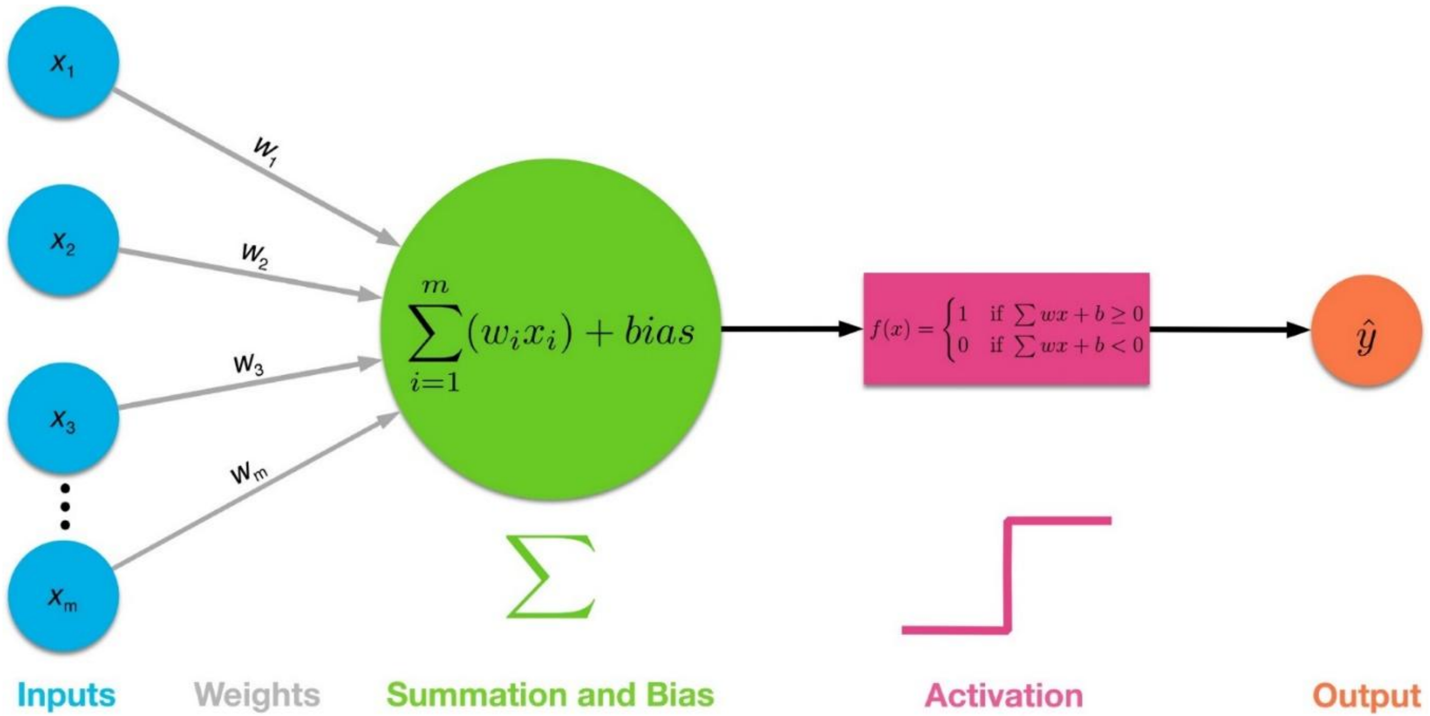
<u>Biological Neural Networks</u>

- Group of connected biological nerve cells
- Human brain is a biological neural network

<u>Artificial Neural Networks</u>

- Group of connected artificial neurons
- Mathematical approximation of biological NNs

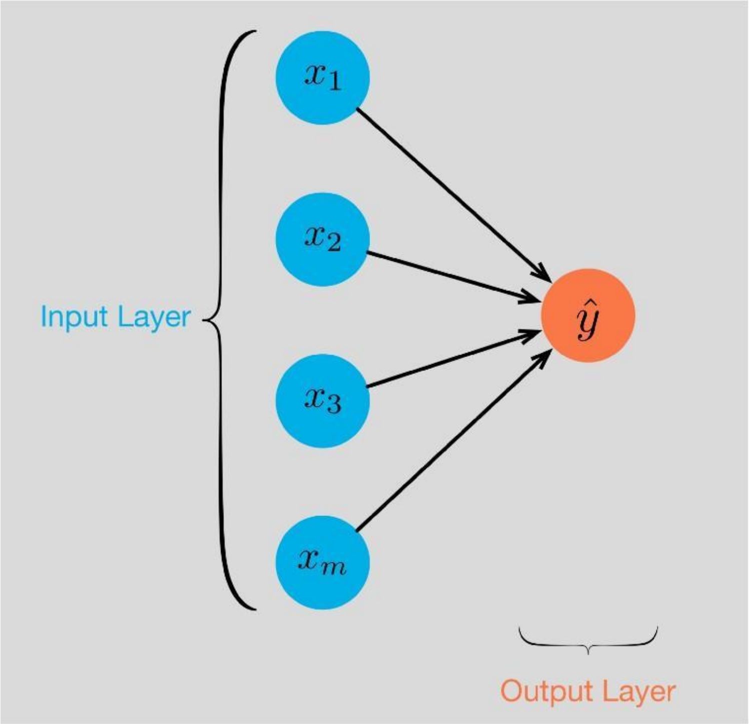
What is an artificial neuron?

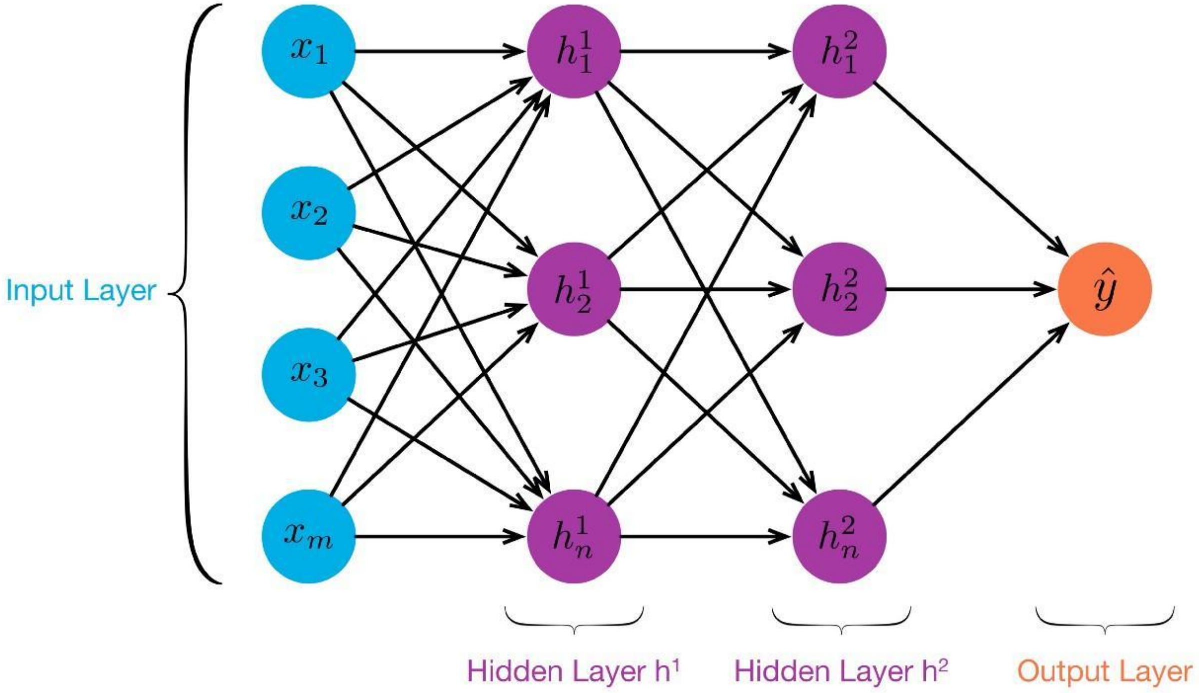

x_1
x_2
x_3
x_m
w_1
w_2
w_3
w_m
$\sum_{i=1}^{m}(w_i x_i) + bias$
$f(x) = \begin{cases} 1 & \text{if } \sum wx + b \geq 0 \\ 0 & \text{if } \sum wx + b < 0 \end{cases}$
$\hat{y}$
Σ
Inputs
Weights
Summation and Bias
Activation
Output

- Parameters to be learnt – weights and biases

source: towardsdatascience.org

Multilayer Perceptron

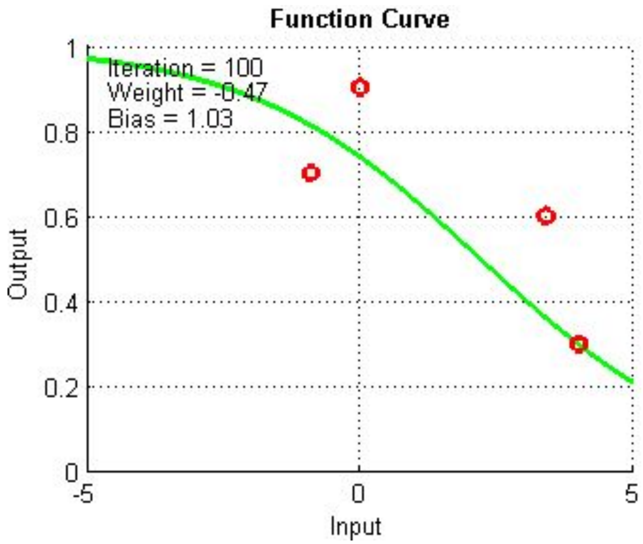


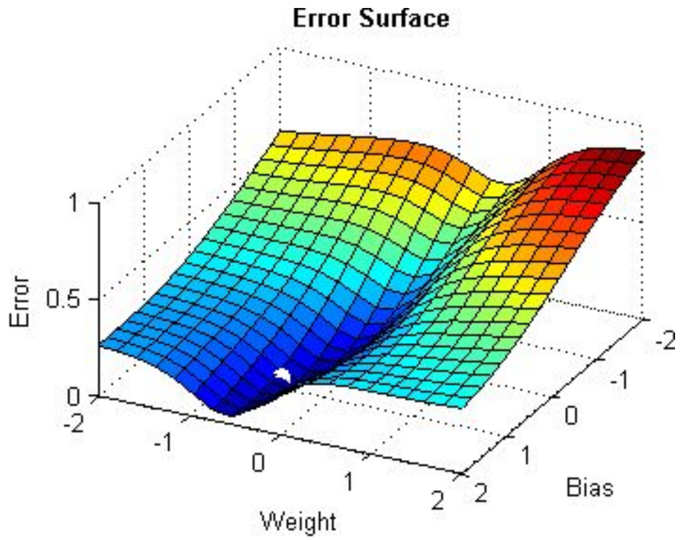


source: towardsdatascience.org

How do the networks learn?

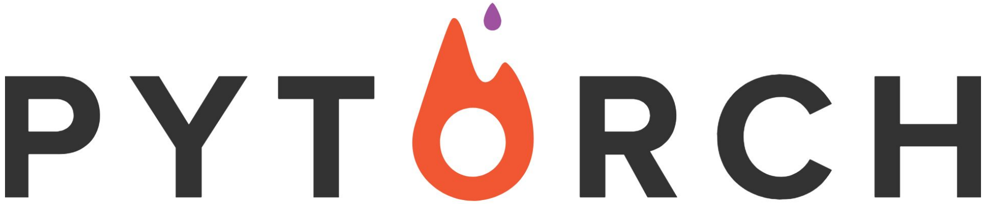
- Cost function – compute error between prediction and label
- Learning by gradient descent – adjust parameters to reduce cost
- **Backpropagation** – compute gradients by applying chain rule of differentiation
- Gradient descent - Blind man walking down a hill step-by-step in search of a valley





Firing up your Jupyter Notebooks!

- conda create env -f med-torch.yml
- conda activate med-torch / source activate med-torch

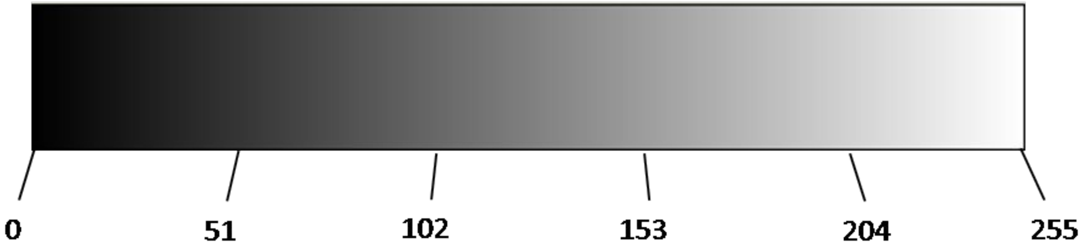

PYTORCH

Introduction to PyTorch – hands on

- Tensor handling ops – matrix multiplication, addition etc..
- Define layers
- Neural network modules
- Forward pass
- Loss computation
- Optimization step
- NumPy to Torch tensor conversion

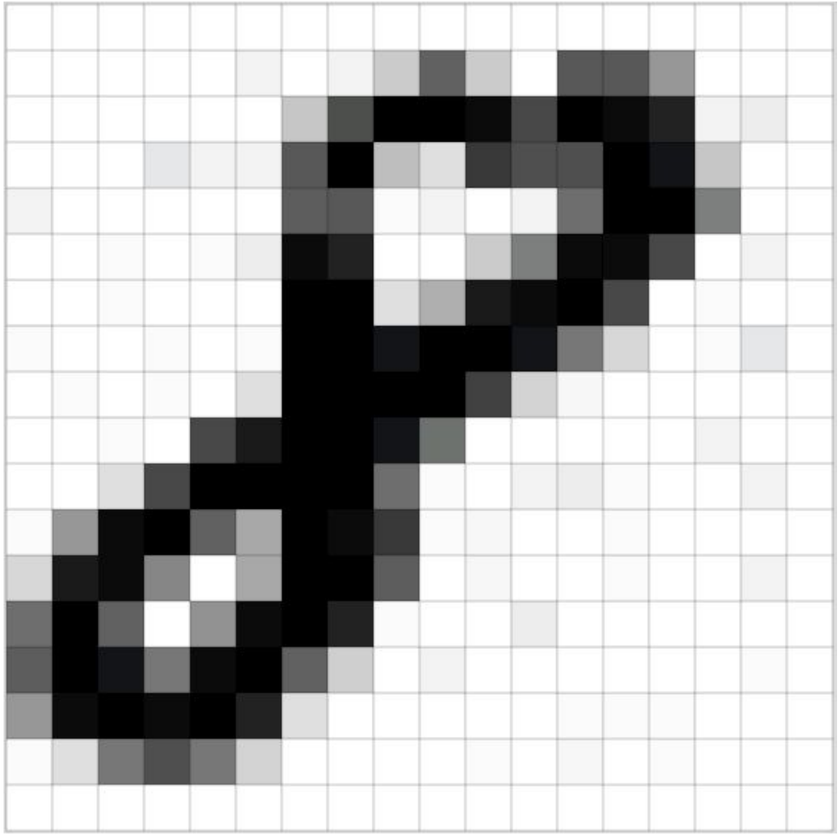
Applying deep learning on hand-written digits

Images are numbers too!



Importance of normalization:

- “Centers” the data
- Stabilizes weight updates

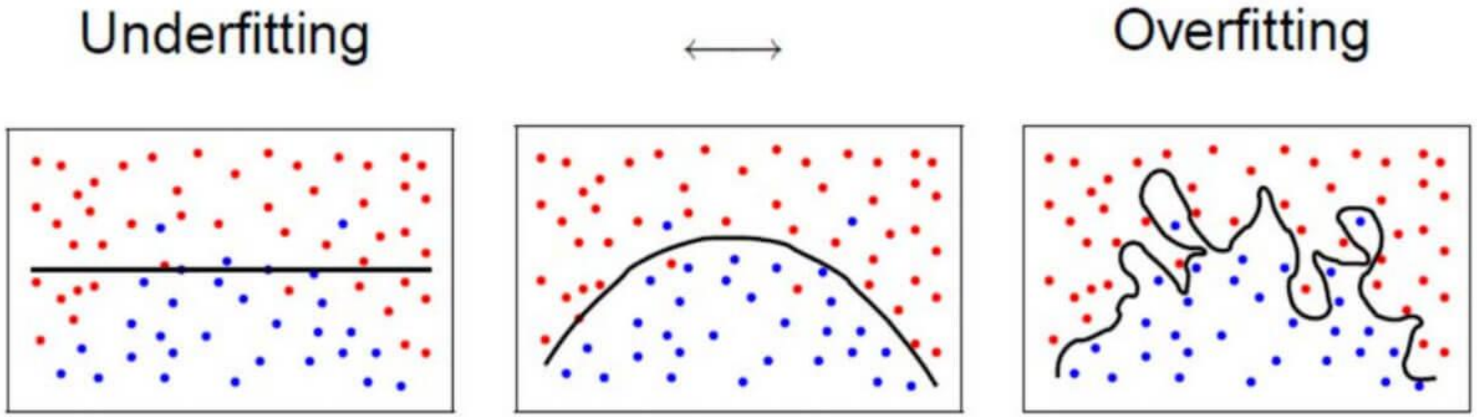


MNIST demo – hands on

- Dataset of 60,000 images
- 10 labels [0-9]
- 28x28 images
- Multi-layer Perceptron
- Learning by gradient descent

Classify digits using a neural network

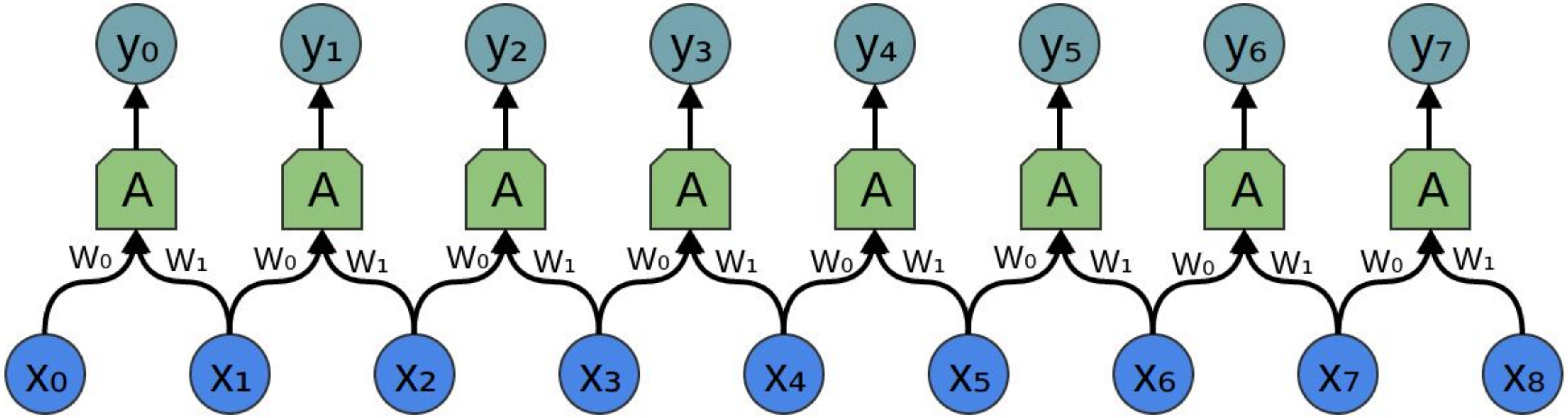
The problem of overfitting



- Too many parameters can result in over-fitting
- Too less can result in under-fitting
- Choose the right number of parameters

Convolutional Neural Networks

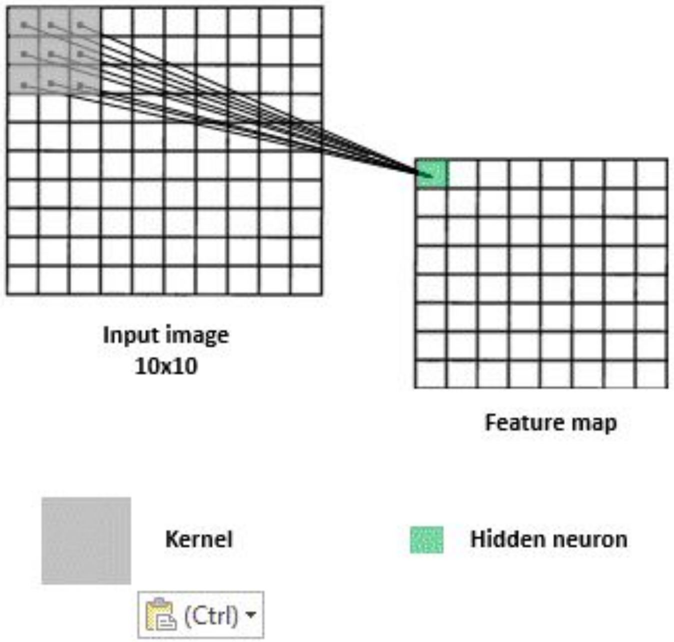
- Weights are shared between neurons – w0 and w1
- Prevents networks from overfitting

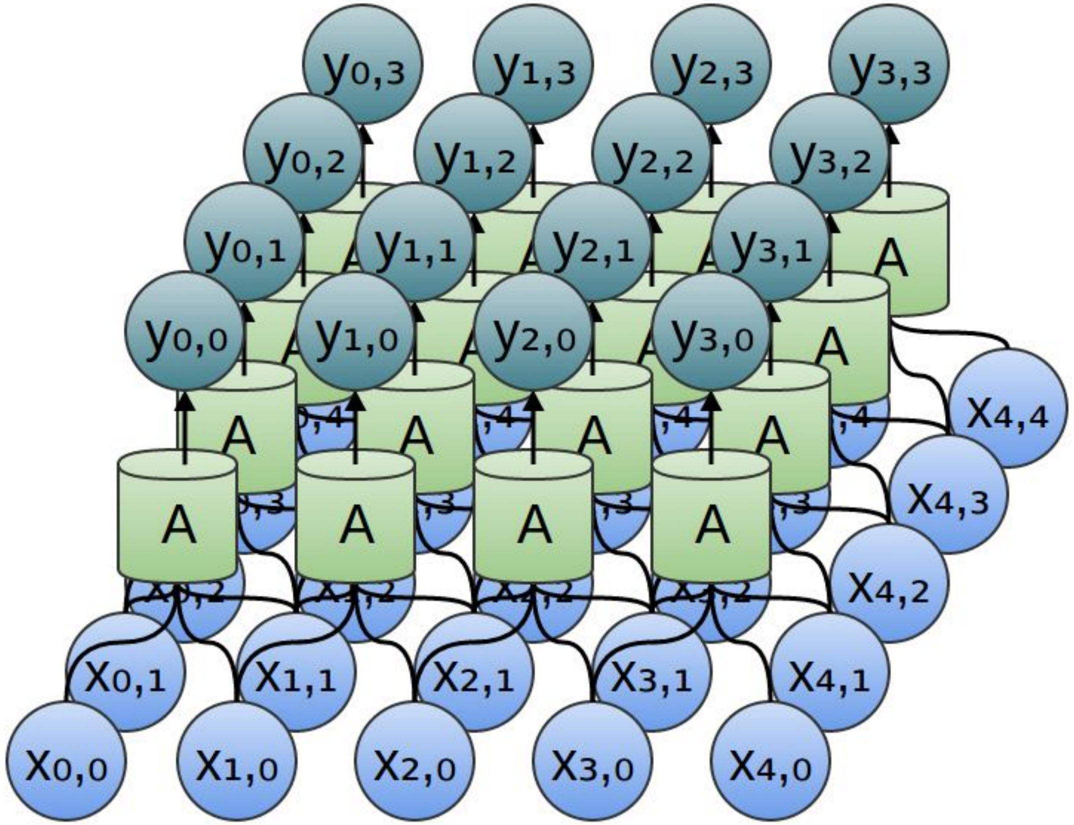


source: colah.github.io

Understanding convolutions in 2D

- “Roll” the layer over the entire image

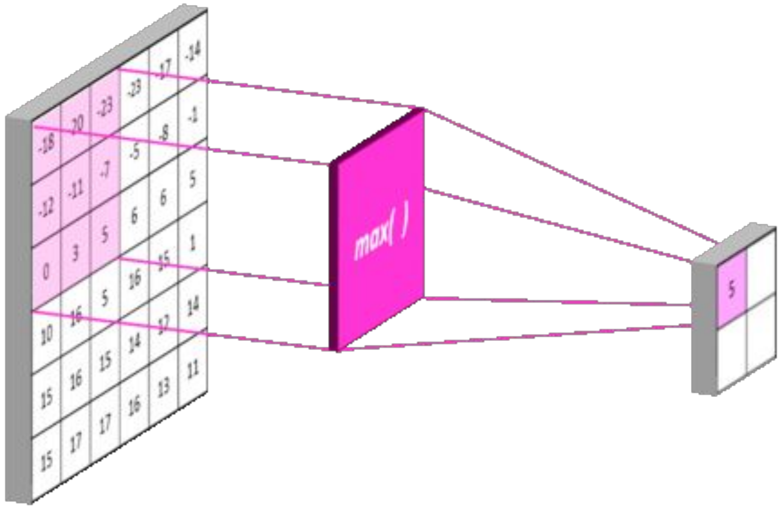




source: colah.github.io

The max-pooling layer

- Take only the maximum defined by the kernel size
- Helps in reducing the size of the network
- Retains important features



-18	20	-23	-23	17	-14
-12	-11	-7	-5	-8	-1
0	3	5	6	6	5
10	16	5	16	15	1
15	16	15	14	17	14
15	17	17	16	13	11

Build a Convolutional Neural Network

- Put together a few convolutions, max-poolings and fully connected layers.

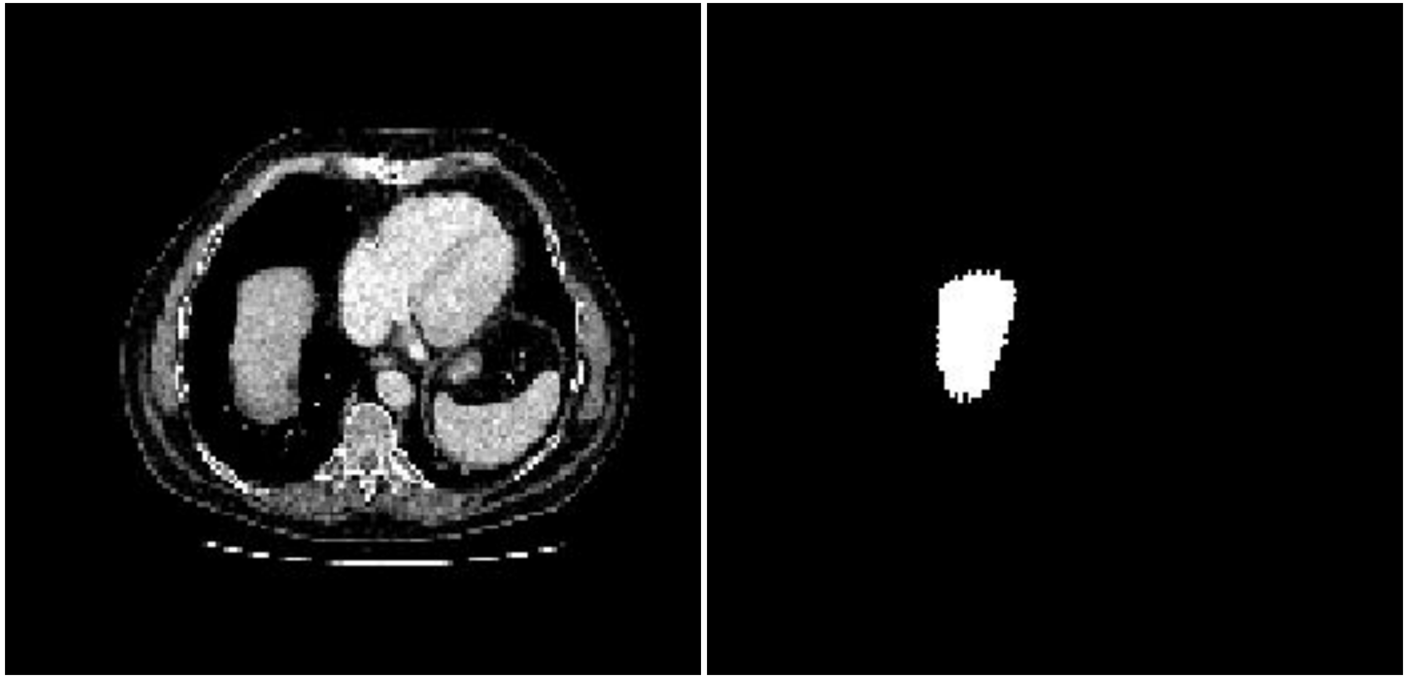
```
class MnistModel(nn.Module):
    def __init__(self):
        super(MnistModel, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, 5, padding=2)
        self.conv2 = nn.Conv2d(32, 64, 5, padding=2)
        self.fc1 = nn.Linear(64*7*7, 1024)
        self.fc2 = nn.Linear(1024, 10)

    def forward(self, x):
        x = F.max_pool2d(F.relu(self.conv1(x)), 2)
        x = F.max_pool2d(F.relu(self.conv2(x)), 2)
        x = x.view(-1, 64*7*7)   # reshape Variable
        x = F.relu(self.fc1(x))
        x = F.dropout(x, training=self.training)
        x = self.fc2(x)
        return F.log_softmax(x)
```

MNIST demo with CNNs – hands on

Pixel-wise segmentation

- Classify every pixel in the image by “rolling” the network on patches
- Process 32x32 patches with stride 1



WiFi : pycon_workshop
Password: wspycon2018