Convolutional Neural Networks

Introduction

Why to Learn CNN?

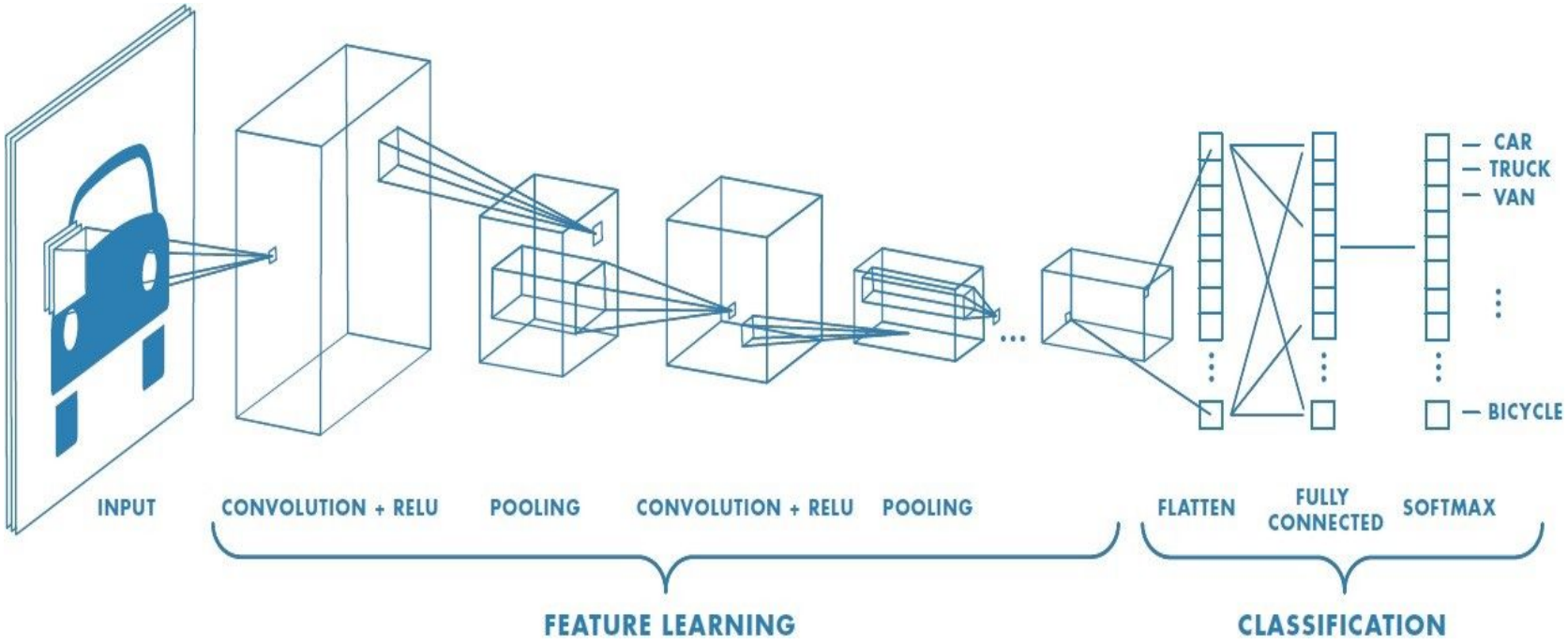
Advantage

- ➢ A class of deep neural networks
- ➢ Computationally efficient
- ➢ Automatic feature extraction
- ➢ Provide superhuman accuracy

Uses

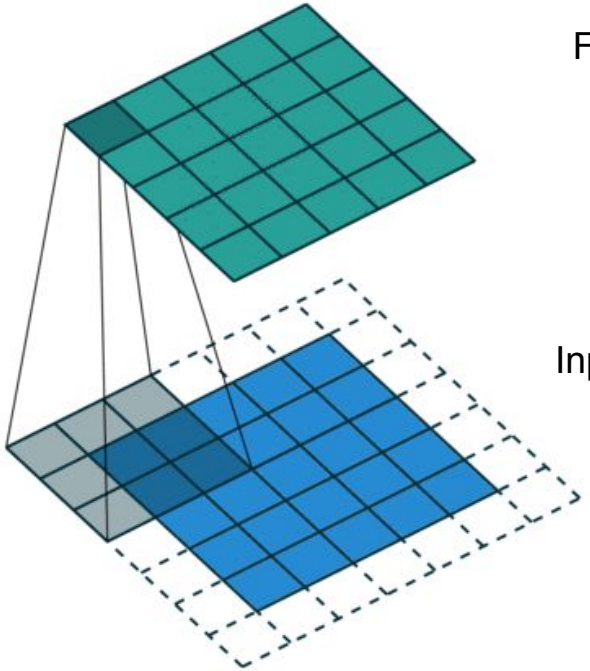
- Image classification
- Recommender systems, natural language processing

What is CNN?



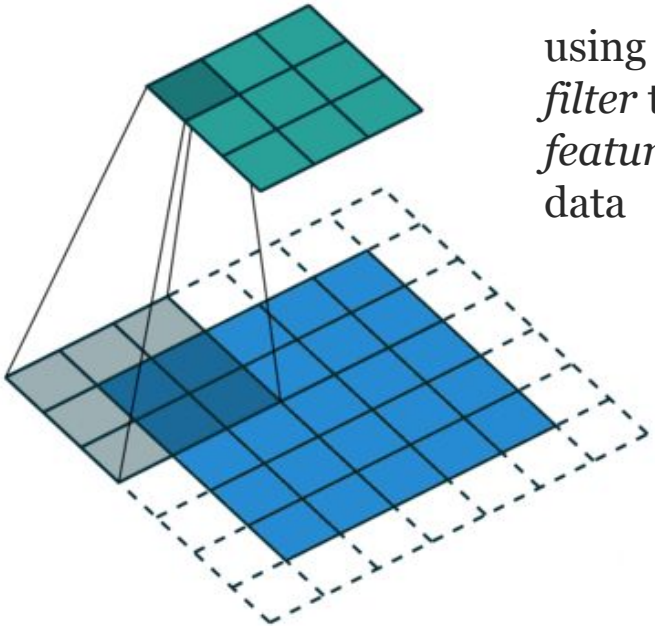
Basic CNN structure : Convolutional layers,Pooling layers, Fully connected layers

Convolution Layer



Stride 1,padding 1
convolution filter(3x3)

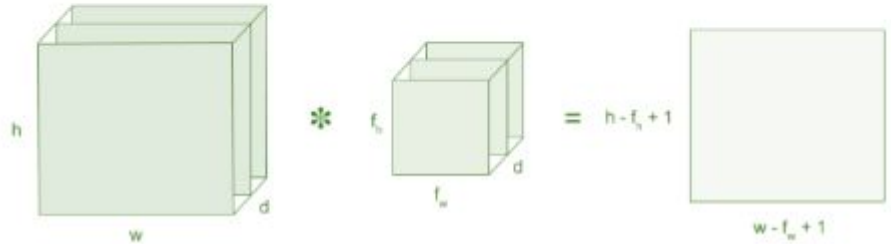
using a *convolution filter* to produce a *feature map* on input data



Stride 2,padding 1
Convolution filter(3x3)

Dimension Calculation

- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f_h x f_w x d)**
- Outputs a volume dimension **(h - f_h + 1) x (w - f_w + 1) x 1**



With Padding and Stride

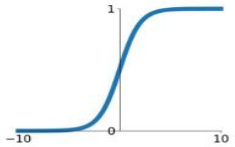
([(h−fh+2P)/S]+1)x([(w-fw +2P)/S]+1)

Activate function

Activation Functions

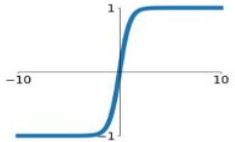
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



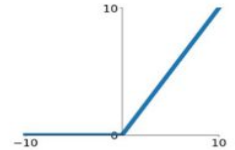
tanh

$$\tanh(x)$$



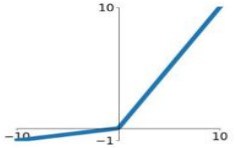
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

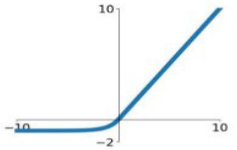


Maxout

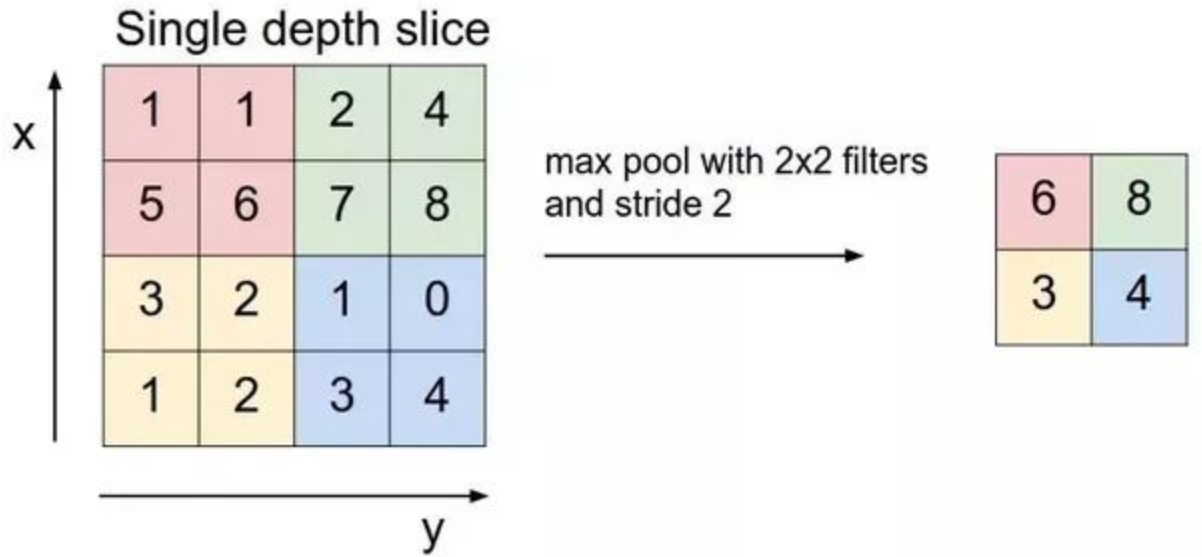
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

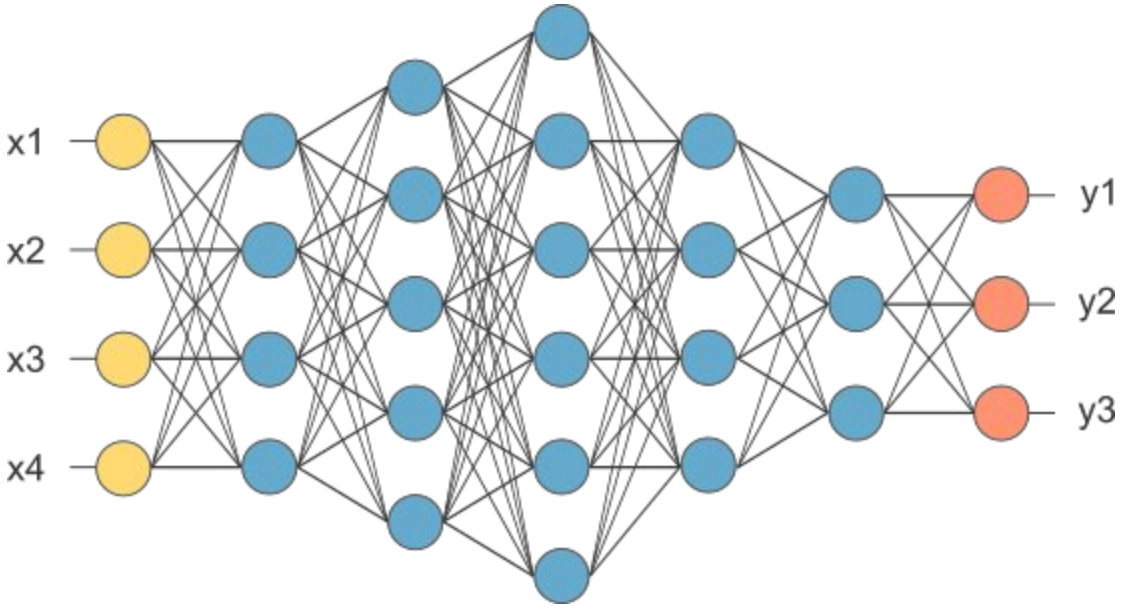
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Pooling Layer(Max,AVg)



Fully Connected Layer



How does CNN work

1. A **forward** phase, where the input is passed completely through the network.
2. A **backward** phase, where gradients are backpropagated (backprop) and weights are updated.

Loss Function

Loss is the quantitative measure of deviation or difference between the predicted output and the actual output in anticipation. It gives us the measure of mistakes made by the **network** in predicting the output.

In simple words, the Loss is used to calculate the gradients. And gradients are used to update the weights of the Neural Net. This is how a Neural Net is trained.

Cross-entropy loss (3)

- For a set of n inputs $\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_i$

labels (one-hot)

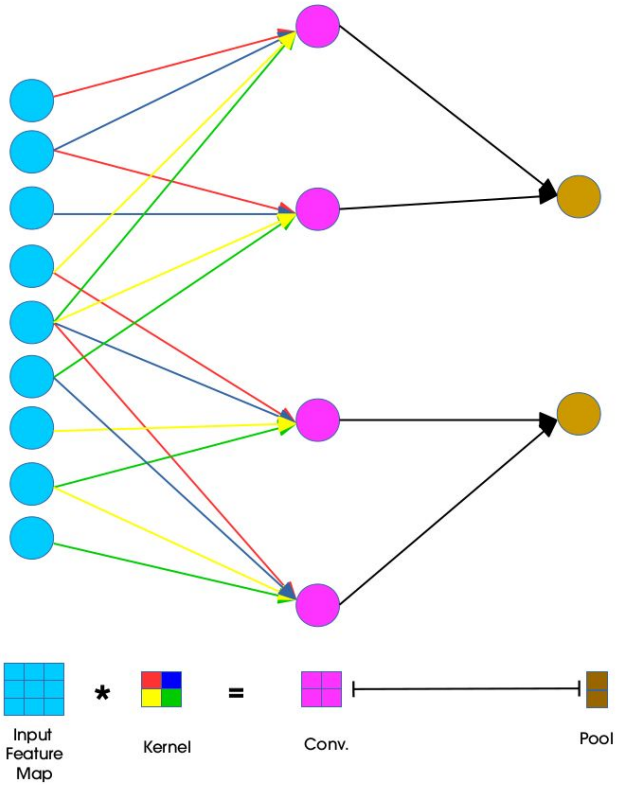
$$\mathcal{L} = -\sum_{i=1}^{n}\mathbf{y}_i \log(S(f_\theta(\mathbf{x}_i)))$$

Softmax

Foward Propagation

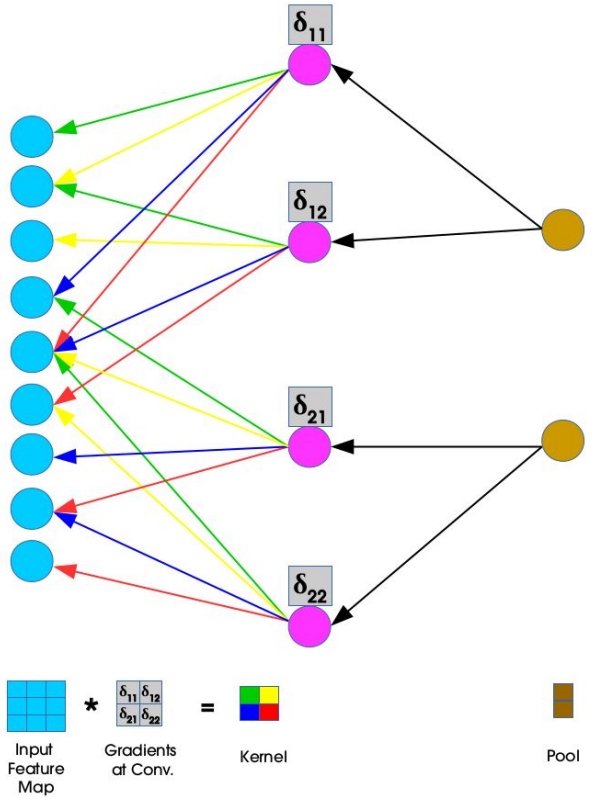
Dot product (input * kernel) is computed and the results summed up to obtain the output at that current location.

weight sharing



Backpropagation

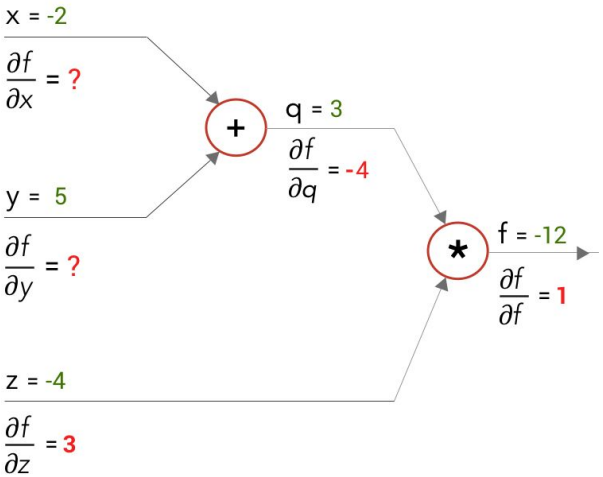
two updates performed, for the weights and the deltas.



Backpropagation

Using chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} * \frac{\partial q}{\partial x}$$



$f = q * z$

$\frac{\partial f}{\partial q} = z \mid z = -4$

$\frac{\partial f}{\partial z} = q \mid q = 3$

$q = x + y$

$\frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$

Example: CNN Image Classification using Pytorch

Performance Tuning

1. Tune Parameters

2. Image Data Augmentation

3. Deeper Network Topology

4. Handel Overfitting and Underfitting problem