Convolutional Neural Network

Dr. J. Bhuvana
Associate Professor

Dr. P. Mirunalini
Associate Professor

Department of CSE, SSN College of Engineering

Transition from Machine Learning –> Deep Learning :
Text, Image and Speech Processing (MLDLTISP '18)
November 13, 2018

Agenda

1. Introduction
2. CNN Architecture
   - Convolutional Layer
   - Hyperparameters
   - ReLU Layer
   - Max pooling
   - Dropout Layers
   - Fully Connected Layer
3. References

Presentation Outline

1 Introduction

2 CNN Architecture
- Convolutional Layer
- Hyperparameters
- ReLU Layer
- Max pooling
- Dropout Layers
- Fully Connected Layer

3 References

Convolutional neural network

- A specific kind of such a deep neural network is the Convolutional network, which is commonly referred to as CNN or ConvNet.
- It's a deep, feed-forward artificial neural network.
- Feed-forward neural networks are also called multi-layer perceptrons(MLPs),
- These neural networks are successful in many different real-life case studies and applications, like:
    - Image classification, object detection, segmentation, face recognition
    - Self driving cars that leverage CNN based vision systems
    - Classification of crystal structure using a convolutional neural network

Convolutional neural network- Advantages

- It automatically detects the important features without any human supervision
- Once trained, the predictions are pretty fast.
- With any number of inputs and layers, CNN can train.
- Neural networks work best with more data points.

Presentation Outline

1 Introduction

2 CNN Architecture
- Convolutional Layer
- Hyperparameters
- ReLU Layer
- Max pooling
- Dropout Layers
- Fully Connected Layer

3 References

Convolutional Neural Network- Architecture

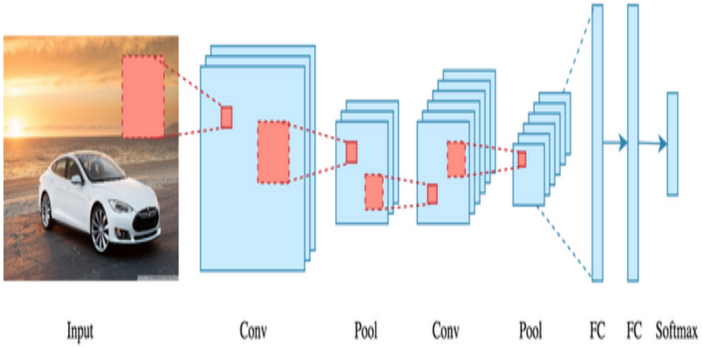


Figure: CNN Architecture [2]

CNN components

- Three basic components to define a basic convolutional network.
  1. The convolutional layer
  2. The Pooling layer[subsampling]
  3. The output layer [fully connected layers]
- CNNs operate over Volumes
- The input is a vector, which is a multi-channeled image (3 channeled in this case)

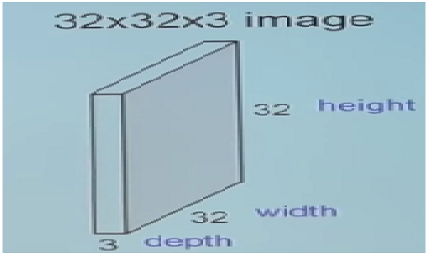


Figure: A RGB image

Convolutional Layer

- The best way to explain a **conv** layer is to imagine a flashlight that is shining over the top left of the image.
- Lets say that the light this flashlight shines covers a 5 x 5 area.
- Lets imagine this flashlight sliding across all the areas of the input image.
- In machine learning terms, this flashlight is called a filter(or sometimes referred to as a **neuron** or a **kernel**) and the region that it is shining over is called the **receptive field**.
- This filter is also an array of numbers (the numbers are called **weights** or parameters).
- **Important note:** The **depth of this filter** has to be the **same** as the **depth of the input** , so the dimensions of this filter is 5 x 5 x 3.

Convolution

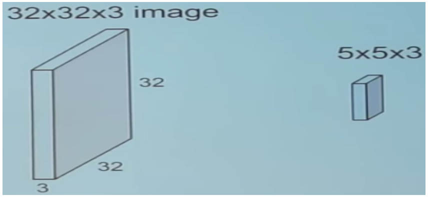
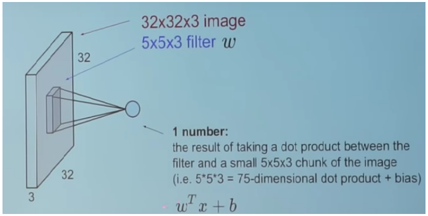


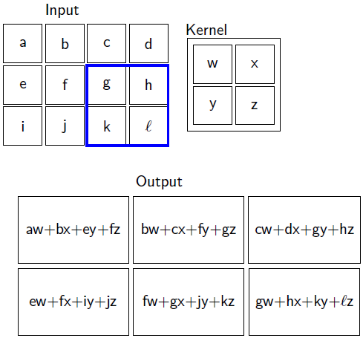
Figure: An image and a filter



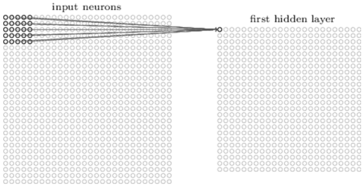
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)
$w^T x + b$

Figure: An image with filter

Convolution



(a) Convolution



(b) Feature map [3]

Figure: Visualizing an activation or feature map

Each time we slide the kernel we get one value in the output.
The resulting output is called a feature map.

Convolution

- Each of these filters will serve as **low level feature** identifiers such as straight edges, simple colors, and curves.
- To predict whether an image is a type of object, the network to be able to recognize higher level features such as hands or paws or ears.
- The output of the network after the 1st conv layer would be a 28 x 28 x 3 volume with three 5 x 5 x 3 filters
- When have another conv layer, the output of the 1st conv layer becomes the **input of the** 2nd **conv layer** i.e. the activation map(s) that result from the 1st layer

Convolution

- When we apply a set of filters on top of that (pass it through the 2nd conv layer), the output will be **activations that represent higher level features**.
- Types of these features could be **semicircles** (combination of a curve and straight edge) or **squares** (combination of several straight edges).
- At the end we get activation maps that represent more and more complex features
- As you go deeper into the network, the filters begin to have a larger and larger receptive field, which means that they are able to consider information from a larger area of the original input volume

Dimensions

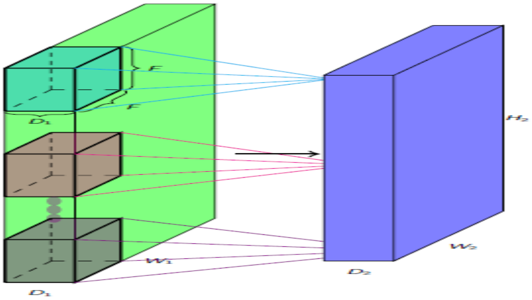


Figure: Dimensions (Image Courtesy: slides of Dr. Mitesh Kaphra, IIT Madras)

We have to define the relations between inputs, filters and outputs

- Width (W1), Height (H1) and Depth (D1) of the original input, Stride S, number of filters K, Spatial extend (F) of each filter (the depth of each filter is same as the depth of each input)
- The output is W2 x H2 x D2

Dimensions

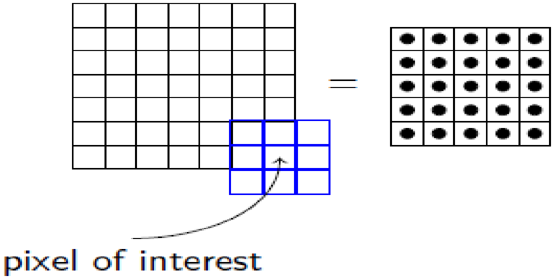


Figure: Kernel (Image Courtesy: slides of Dr. Mitesh Kaphra, IIT Madras)

- We can't place the kernel at the corners as it will cross the input boundary
- This results in an output which is of smaller dimensions than the input
- If we want the output to be of same size as the input : We use padding

Padding

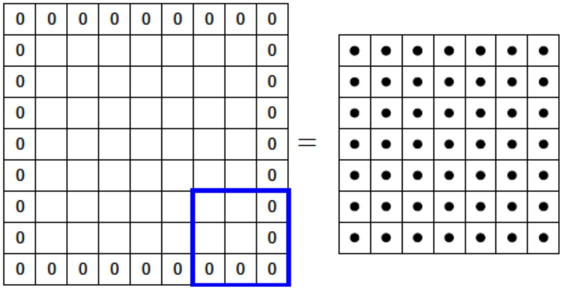


Figure: Padding (Image Courtesy: slides of Dr. Mitesh Kaphra, IIT Madras)

- Let us use pad P = 1 with a 3 X 3 kernel
- This means we will add one row and one column of 0 inputs at the top, bottom, left and right

Stride

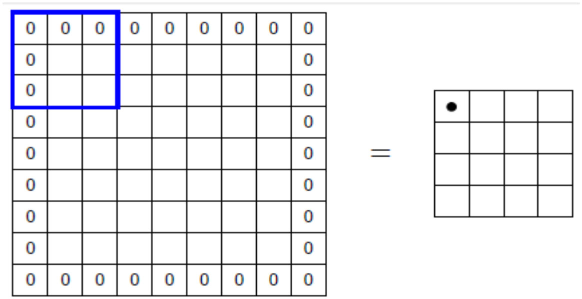


Figure: Stride (Image Courtesy: slides of Dr. Mitesh Kaphra, IIT Madras)

Stride

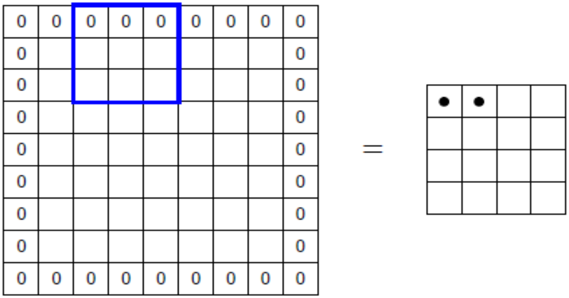


Figure: Stride (Image Courtesy: slides of Dr. Mitesh Kaphra, IIT Madras)

Stride

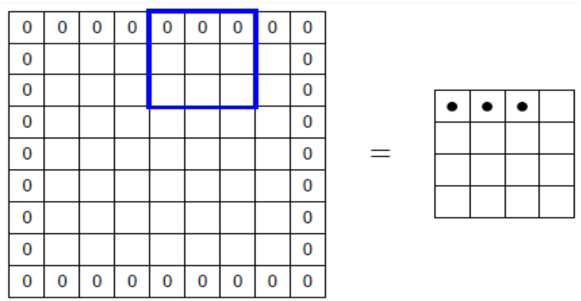


Figure: Stride (Image Courtesy: slides of Dr. Mitesh Kaphra, IIT Madras)

Stride

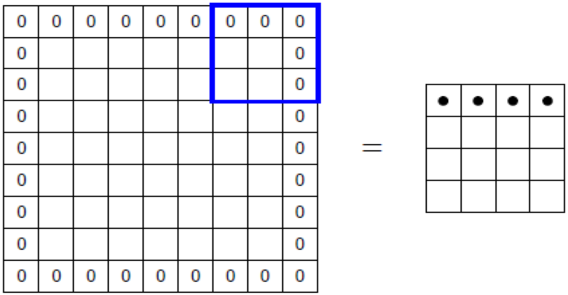


Figure: Stride (Image Courtesy: slides of Dr. Mitesh Kaphra, IIT Madras)

Stride

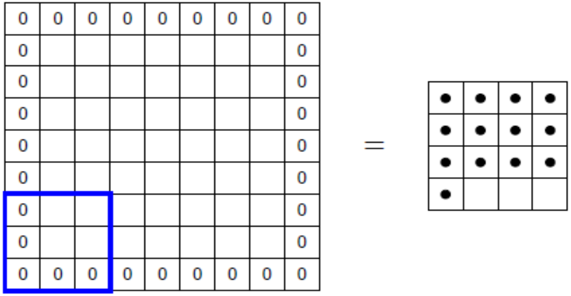


Figure: Stride (Image Courtesy: slides of Dr. Mitesh Kaphra, IIT Madras)

Stride

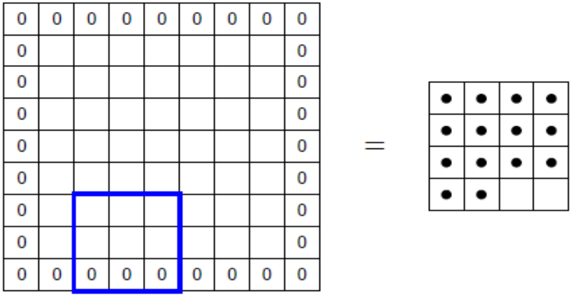


Figure: Stride (Image Courtesy: slides of Dr. Mitesh Kaphra, IIT Madras)

Stride

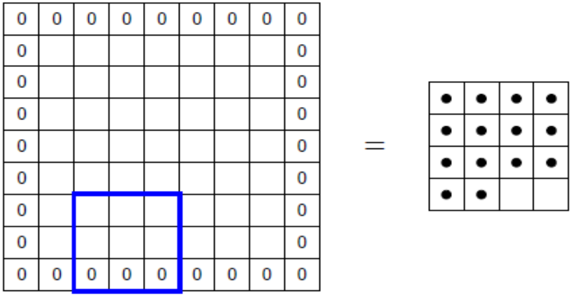


Figure: Stride (Image Courtesy: slides of Dr. Mitesh Kaphra, IIT Madras)

Stride

The intervals at which the filter is applied (here $S = 2$)

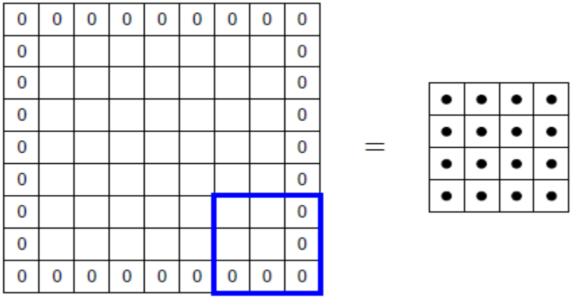


Figure: Stride (Image Courtesy: slides of Dr. Mitesh Kaphra, IIT Madras)

Here we are skipping every 2_{nd} pixel which will again result in an output which is of smaller dimensions

Hyperparameters

The final dimensions will be

$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1$$

$$D_2 = K$$

- Each filter gives us one 2d output.
- K filters will give us K such 2D outputs
- We can have many kernels but the kernels will be shared by all locations in the image calling it as weight sharing

ReLU (Rectified Linear Units) Layer

- After each conv layer, it is convention to apply a nonlinear layer (or activation layer) immediately afterward.
- The purpose is to introduce nonlinearity to a system that basically has just been computing linear operations during the conv layers.
- In the past, nonlinear functions like *tanh* and *sigmoid* were used, but **ReLU** layers work far better because the network is able to train a lot faster without making a significant difference to the accuracy.
- It also helps to **alleviate the vanishing gradient** problem, which is the issue where the lower layers of the network train very slowly because the gradient decreases exponentially through the layers
- The ReLU layer applies the function $\mathbf{f(x) = max(0, x)}$ to all of the values in the input volume.
- In basic terms, this layer just **changes all the negative activations to 0**.

Pooling Layer

Sometimes when the images are too large, we would need to reduce the number of trainable parameters.

- It is then desired to periodically introduce pooling layers between subsequent convolution layers.
- It progressively reduce the spatial size of the representation to reduce the amount of parameters and computation and also controls overfitting.
- Pooling is done independently on each depth dimension, therefore the depth of the image remains unchanged.
- The most common form of pooling layer generally applied is the max pooling, with filters of size 2x2 applied with a stride of 2
- This subsampling of the pixels makes image smaller with fewer parameters to characterize the image

Max pooling

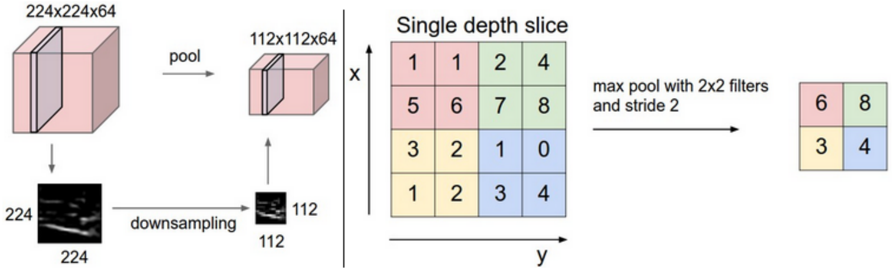


Figure: Max pooling [1]

- **Left:** The input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64].
- **Right:** With stride of 2, That is, each max is taken over 4 numbers (little 2x2 square).

Dropout Layers

- This layer **drops out** a random set of activations in that layer by setting them to zero.
- To show that a network can provide the right classification or output for a specific example even if some of the activations are dropped out.
- It helps to alleviate the overfitting problem.
- This layer is only used during training, and not during test time.

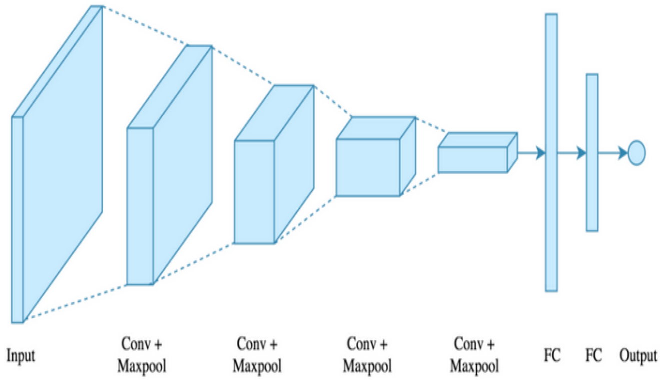
Fully Connected (FC) Layer

The convolution and pooling layers would only be able to extract features and reduce the number of parameters from the original images.

- To generate the final output we need to apply a fully connected layer to generate an output equal to the number of classes we need.
- Neurons in a FC layer have full connections to all activations in the previous layer.
- Their activations can hence be computed with a matrix multiplication followed by a bias offset.
- A fully connected layer expects a 1D vector of numbers, hence Flattening is applied to simply arranging the 3D volume of numbers into a 1D vector
- The output layer has a loss function like categorical cross-entropy, to compute the error in prediction.
- Once the forward pass is complete the backpropagation begins to update the weight and biases for error and loss reduction.

Putting it all together - How does the entire network look like

A CNN model can be thought as a combination of two components: feature extraction part and the classification part. The convolution + pooling layers perform feature extraction.



CNN - a walkthrough

1. Pass an input image to the 1_{st} convolutional layer. The convoluted output is an activation map, the filters extract relevant features from the input image.
2. If we need to retain the size of the image, we use same padding(zero padding), other wise valid padding is used.
3. Pooling layers are added to reduce the number of parameters
4. Several conv and pooling layers are added before the prediction is made.
5. The output layer is a FC layer, where the input from the other layers is flattened and sent, this transforms the output into the number of classes.
6. A loss function is defined in FC output layer to compute the mean square loss. The gradient of error is then calculated.
7. The error is then backpropagated to update the filter(weights) and bias values.
8. One training cycle is completed in a single forward and backward pass.

Presentation Outline

1 Introduction

2 CNN Architecture
- Convolutional Layer
- Hyperparameters
- ReLU Layer
- Max pooling
- Dropout Layers
- Fully Connected Layer

3 References

References I

CS231n: Convolutional Neural Networks for Visual Recognition. .
http://cs231n.github.io/convolutional-networks/,.

Arden Dertat.
Applied Deep Learning - Part 4: Convolutional Neural Networks.
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2, Nov 8, 2017.

Michael A Nielsen.
Neural networks and deep learning, volume 25.
Determination press USA, 2015.

THANK YOU