

Convolutional Neural Networks

UMaine COS 470/570 – Introduction to AI

Spring 2019

Created: 2019-04-29 Mon 21:45

1

Convolutional neural networks

- One of the major kinds of ANNs in use
- One of the reasons deep learning is so successful:
 - addresses computational tractability
 - addresses vanishing/exploding gradient problem
- Start – 80s
- First truly successful modern version: LeNet (LeCun, 1989)
- LeNet-5 (LeCun, 1998): 7-layer CNN for reading numbers on checks

2.1

The problem

- Goal: High-accuracy image recognition
- Standard supervised learning with deep (fully-connected) networks:
 - Images require connections from each pixel \rightarrow each neuron
 - E.g., 1028×768 image \Rightarrow about 789,504 weights *per neuron*
 - Slow to train
 - Vanishing/exploding gradient problem
 - Also no spatial locality exploited
- Can we take inspiration from biological vision systems?

Human visual system

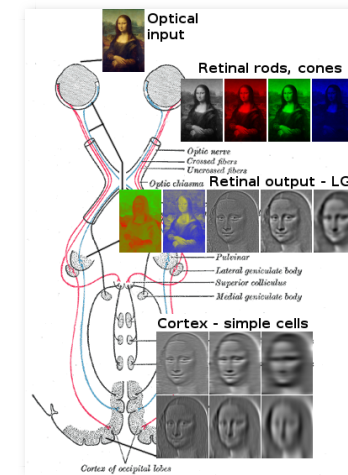
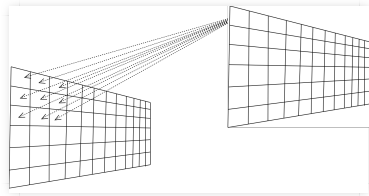


Image credit: user Clock, CC BY-SA 3.0, via Wikimedia Commons

Convolutional layers

- Instead of fully-connected layer, think of using a layer whose neurons each have a *receptive field*:



- Overlapping receptive fields
- Neurons then learn local features, only have a few weight each
- Have multiple feature-detecting layers per convolutional layer
- Problem:
 - Features should be location-independent
 - \Rightarrow Weights for nodes should be shared, learned together

5.1

Shared weights

- So—how to compute the layer?
- For a $n \times n$ *receptive field*:
 - $n \times n$ weights
 - If input layer is $m \times m$, hidden layer is $m - n + 1 \times m - n + 1$
 - For hidden layer neuron at x, y , activation is:

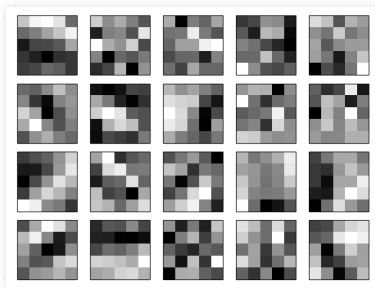
$$\sigma(b + \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} w_{ij} a_{x+i, y+j})$$

- Slide the kernel across, down the image by some *stride*
- b weights = *kernel* or *filter*
- Hidden layer = *feature map*
- Update weights based on entire hidden layer's computed loss function

6.1

Why convolutional layer?

- Learns local spatial features of input
- Location-independent (location-invariant)
- Example (from Nielsen, M: *Neural Networks and Deep Learning*):

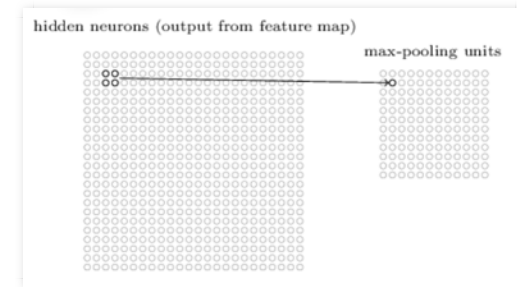


- Typically > 1 feature map/layer \Rightarrow learn different kinds of features

7.1

Pooling layers

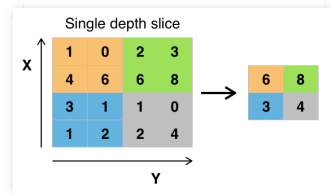
- Convolutional layers are coupled with *pooling layers*
- Each node of pooling layer connected to some $i \times j$ region of feature map



8.1

Pooling layers

- Pool based on some function—max, average, etc.



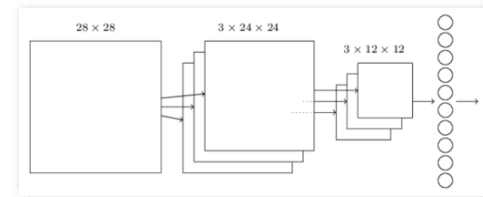
(Aphex34 [CC BY-SA 4.0], via Wikimedia Commons)

- Purpose(s):
 - Reduce # weights needed
 - Blur/average/smooth feature map
 - Determining if a feature is in a particular region

9.1

Using the features

- Pooled layers' output \Rightarrow fully-connected layer – e.g., for MNIST:



(From Nielson))

- Learn configuration of features
- Could have multiple fully-connected layers, too

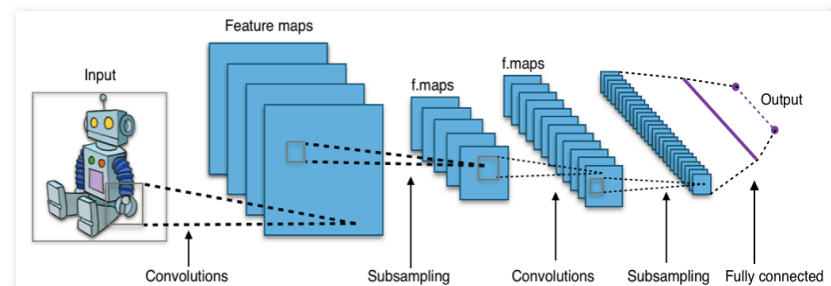
10.1

Learning in CNNs

- Backpropagation learning, gradient descent
- Equations for fully-connected nets have to be modified, though
- Theano, TensorFlow, PyTorch – all have support for training CNNs

Multiple convolutional layers

- Multiple convolutional + pooling layers:

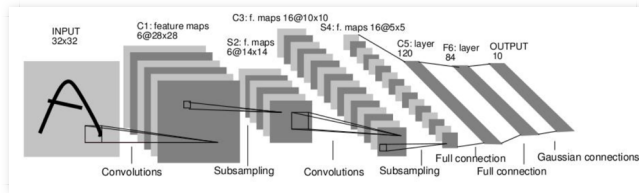


(Aphex34 [CC BY-SA 4.0], via Wikimedia Commons)

- Deeper layers \Rightarrow more complex features

Multiple convolutional layers

- LeNet-5:
 - 7 layers
 - Recognize numbers on checks

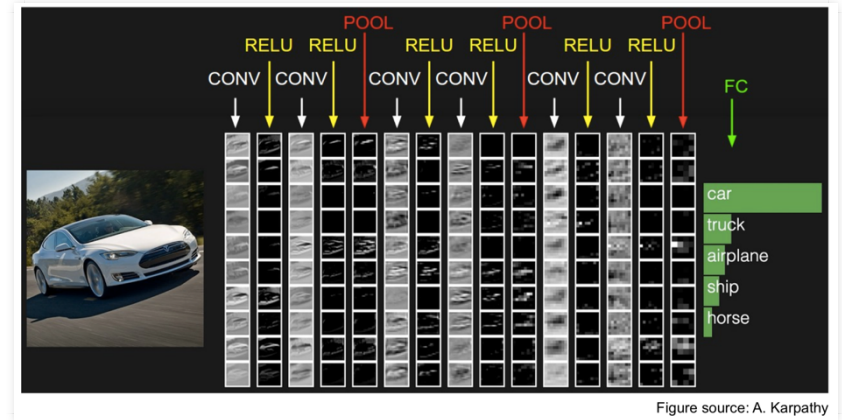


- Recall the DQN we talked about used CNNs
- Many additional variants of CNNs now
- ResNet: 152 layers, general image recognition, lots of additions to LeNet's basic architecture

13 . 1

Feature detection in CNNs

- From ConvNet:



14 . 1

Progress in image recognition competition

Team	Year	Place	Error (top-5)	External data
SuperVision – Toronto (AlexNet, 7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai – NYU (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG – Oxford (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
ResNet (152 layers)	2015	1st	3.57%	
Human expert*			5.1%	

<http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/>

(Aphex34 [CC BY-SA 4.0], via Wikimedia Commons)

15 . 1

Your turn

1. Build a CNN
 - Get into groups, one of whom has a laptop with Keras on it
 - Create a simple CNN for MNIST
2. Explain a CNN
 - Get into groups with at least 2 laptops
 - Part of group: Look up an “inception” layer in (e.g.) GoogleNet
 - Other part: Look up ResNet
 - Explain them to each other after a few minutes