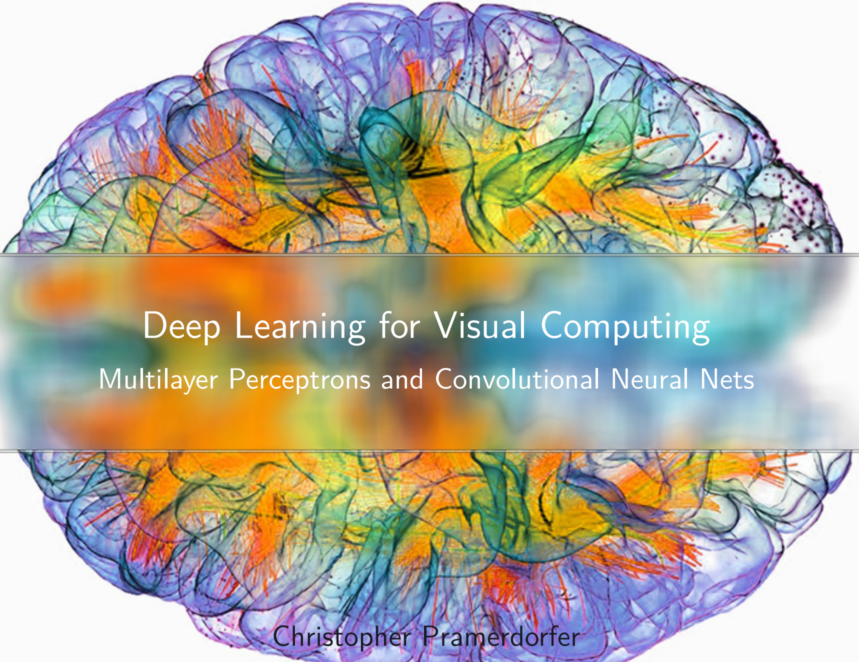


Deep Learning for Visual Computing

Multilayer Perceptrons and Convolutional Neural Nets

Christopher Pramerdorfer
Computer Vision Lab, TU Wien

Image by Neuroscape Lab

Topics

Image classification recap

Multilayer perceptrons

Representation learning and deep learning

Convolutional neural networks

Image Classification Recap

We are concerned with image classification

Image from youtube.com

$\Rightarrow$ cat

Image Classification Recap

We've covered the family of parametric models

How they can be used for classification

- Predict vector $\mathbf{w}$ of class scores

How they can be trained

- Optimize cross-entropy loss using SGD
- Using regularization strategies

Image Classification Recap

Performance has been lackluster so far

Test accuracies using HOG features and softmax classifier

- About 42% on `TinyCifar10Dataset`
- About 48% on `Cifar10Dataset`

Two directions for improvements

- Better classifier
- Better features

Multilayer Perceptrons
Motivation

Recall that softmax classifier is linear

- Restricted to linear decision boundaries

But classes not linearly separable in HOG feature space

- Unable to reach low training error

Multilayer Perceptrons (MLPs) are a powerful alternative

MLPs are universal approximators

- Can approximate any function to any degree (regression)
- Can represent arbitrary decision boundaries (classification)

Trained in the same SGD & loss function framework

A (artificial) neural network

- Is a directed computational graph
- Vertices (neurons) are scalar functions of input
- Edges define data flow

And thus a function $f : \mathbf{x} \in \mathbb{R}^D \mapsto \mathbf{w} \in \mathbb{R}^T$

- That is composed of other functions (neurons)
- Neurons operate on (subset of) $\mathbf{x}$ and/or neuron output

Multilayer Perceptrons
Feedforward Neural Networks

In feedforward neural networks this graph is acyclic

Neurons at same level in hierarchy form a layer

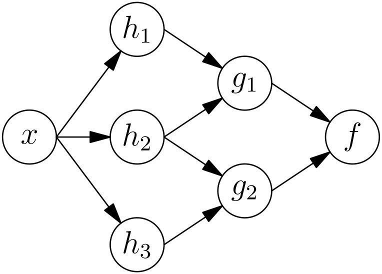


Image from wikipedia

Multilayer Perceptrons
Feedforward Neural Networks

D input units (x) and T output units (f)

Flexible number of hidden units (h,g)

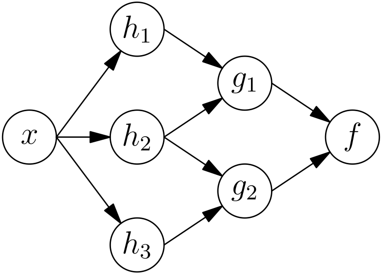


Image from wikipedia

Multilayer Perceptrons

Feedforward Neural Networks

Input units only provide data (no computations)

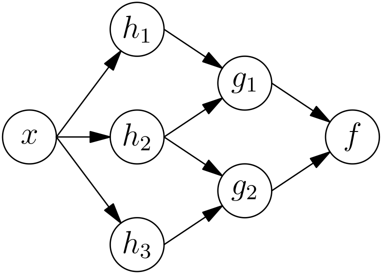


Image from wikipedia

MLPs are feedforward neural networks

- With one input layer and one output layer
- Neurons in layer l connected to all neurons in layer $l-1$

Every (non-input) neuron i computes $n_i(\mathbf{a}_i^\top \mathbf{x}_l + b_i)$

- Linear transformation of input $\mathbf{x}_l$ (varies per layer)
- Followed by some activation function n_i

Multilayer Perceptrons
Activation Functions

Activation function of output units depends on problem

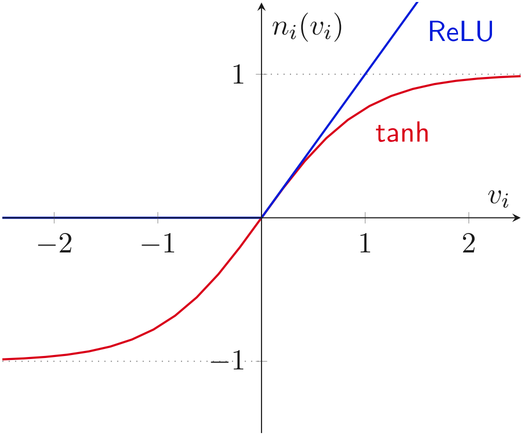
- Identity for regression
- Softmax for classification unless $L(\boldsymbol{\theta})$ includes it

Common hidden layer activation functions (non-linearities)

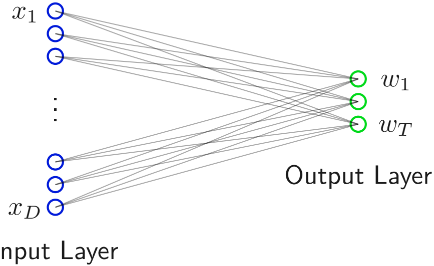
- $n_i(v_i) = \tanh(v_i)$
- $n_i(v_i) = \max(0, v_i)$ (Rectified Linear Unit, ReLU)

Multilayer Perceptrons

Activation Functions

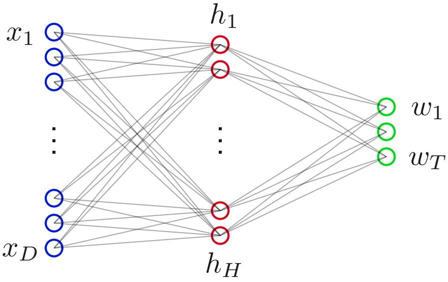


MLPs without hidden units are linear models



Most MLPs have single hidden layer with H neurons

- Depth of 2 (two layers with parameters)



Representational capacity depends on (hyperparameters)

- Number of hidden units H
- Type of non-linearities (usually ReLU)

Neural networks (including MLPs and CNNs) work best if

- They have enough capacity to overfit
- Are regularized properly to avoid this

Networks with higher capacity perform better

- Because larger networks are easier to train

Recall what we learned about gradient descent limitations

- Applies only for sufficiently large networks
- Particularly, smaller networks have bad local minima

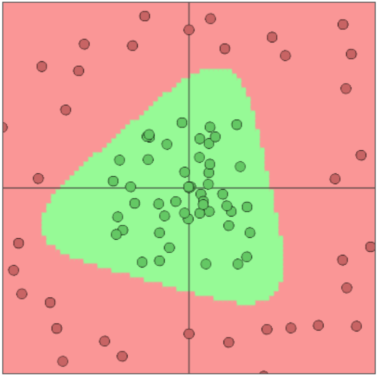


Image from cs.stanford.edu

CIFAR10 dataset, HOG features, MLP : $\approx 60\%$ test accuracy

- Much better than before
- But still far from state of the art (90% and more)

HOG features are limiting factor

Representation Learning
Motivation

Could extract additional features

- Features that retain color information
- HOG features obtained using different extractor settings
- Other features (SIFT, LBP, ...)

Standard approach until breakthrough of deep learning

- Limitations of features remain limiting factor
- Works only to some extent

Representation Learning
Motivation

Recall that manually designed features are low-level

- Capture basic properties of image
- Such as contrast changes, dominant colors

We want task-specific high-level features

- Features with semantic meaning
- E.g. presence of wheel for car classification

Representation Learning
Motivation

We cannot design reliable high-level feature extractors

- But we can try to learn them
- This task is called representation learning

Representation is feature vector used for classification

- Representation learning = learning to extract features

In our applications, input are images after preprocessing

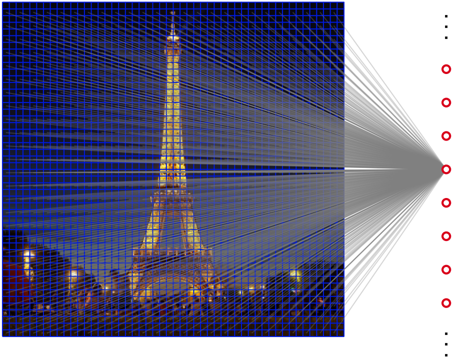
MLPs are not suited for this task

- Not designed for images

Number of parameters increases quickly with D and H

- $D \cdot H$ weights in first hidden layer
- 224 by 224 RGB image and $H = 500:\ 75$ million

Representation Learning
Challenges



Representation Learning
Challenges

Can think of a two-layer MLP as

- A stage that learns to extract H features from $\mathbf{x}$
- A stage that trains a linear classifier on these features

Must learn high-level features in single step

- This does not work in general
- Similar difficulty as solving original problem

Deep learning solves this problem using divide and conquer

- Learn to extract features in hierarchical way
- Later features build upon earlier (simpler) ones

Hierarchy has many levels, hence the name deep learning

Representation Learning
Deep Learning

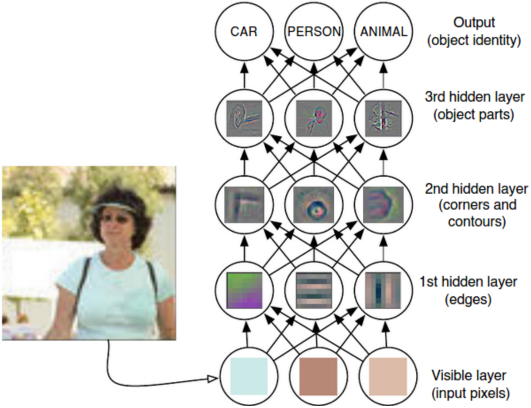


Image from [1]

Cannot just use MLPs with several hidden layers

- Usually no improvements as depth exceeds 3

MLPs are unable to learn good image features either way

- Not surprising since no understanding of images

We'll now switch to models that do

Convolutional Neural Networks

We'll now adapt above MLP architecture to images

- And other grid-like data
- Result is no longer a MLP

Should make use of spatial structure of images

- Not appropriate to flatten images to vector $\mathbf{x}$

Retain structure by arranging input neurons accordingly

- If input images have size $W_0 \times H_0$ and C_0 channels
- Input neurons form $W_0 \times H_0 \times C_0$ grid
- W_0 is width, H_0 is height, C_0 is depth of layer

Input Image

Input Layer

Spatially close pixels are highly correlated, others are not

- Nearby pixels correspond to same object (or part)
- Want to learn features using nearby pixels

Realized by

- Arranging hidden layer neurons in $W_l \times H_l$ grid
- With each neuron having a sparse connectivity

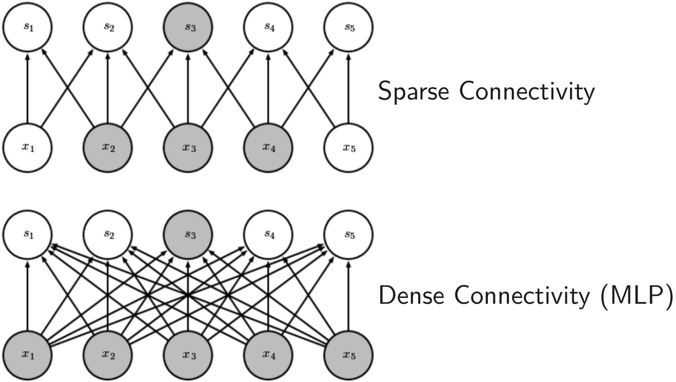


Image adapted from [1]

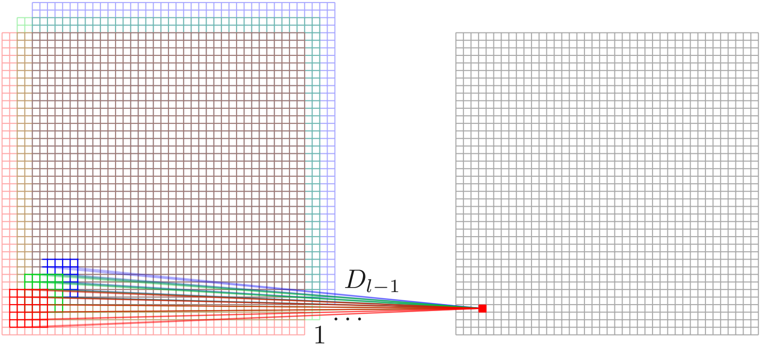
W_l and H_l depend on input width and height

- Usually $W_l = W_{l-1}$ and $H_l = H_{l-1}$

Connectivity c along with and height dimensions

- Is configurable but usually $c = 3$, that is 3×3

Connectivity along depth dimension is always D_{l-1}



Layer $l-1$

Layer l $(c \times c \times D_{l-1} = 5 \times 5 \times 3)$

Every neuron learns

- How to extract a feature
- By combining inputs in local neighborhood

Operation is same as before, $n_i(\mathbf{A}_i \cdot \mathbf{X}_i + b_i)$

- Weights and inputs are now $c \times c \times D_{l-1}$ matrices
- $\mathbf{A}_i \cdot \mathbf{X}_i$ is dot product of matrices
- Input now varies with spatial location

Such layers are called locally connected layers

Every neuron extracts different feature (different parameters)

Usually given feature is useful anywhere in image

- E.g. ability to detect car wheel anywhere in image

Enforced by weight sharing between neurons

- Neurons compute $n_i(\mathbf{A}_l \cdot \mathbf{X}_i + b_i)$
- Same weights for all neurons in layer

Now every neuron in layer

- Takes input in local neighborhood
- Computes linear combination with identical weights
- Adds bias and applies non-linearity

First two operations equal to 3D convolution operation

- Hence such layers are called convolutional layers
- Fundamental layer of convolutional neural networks

But layer can learn only single feature (not sufficient)

To overcome this problem, we replicate neurons D_l times

- Resulting in $W_l \times H_l \times D_l$ grid of neurons
- Only neurons with same depth D_l share weights
- Every 2D depth slice is called feature map

Layer with D_l feature maps can learn D_l different features

- D_l is another hyperparameter

Number of weights $\mathbf{A}_l$ depends only on c, D_{l-1}, D_l

- $c = 3, D_{l-1} = 3, D_l = 32 \implies 864$ weights
- $c = 3, D_{l-1} = 32, D_l = 64 \implies 18.5$k weights

Few parameters per layer

- Can use several convolutional layers in network
- Layer l learns to combine layer $l - 1$ features to new ones

Convolutional Neural Networks
Definition

Convolutional Neural Networks (CNNs, convnets)

- Are feedforward neural networks
- That include convolutional layers

CNNs are optimized for data with grid-like structure

- Most important models for image analysis

Receptive field increases with depth

- Input region the neuron "sees" (indirect connection)

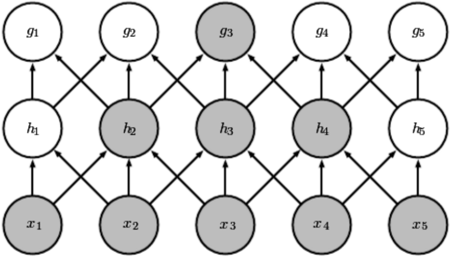


Image from [1]

So in CNNs

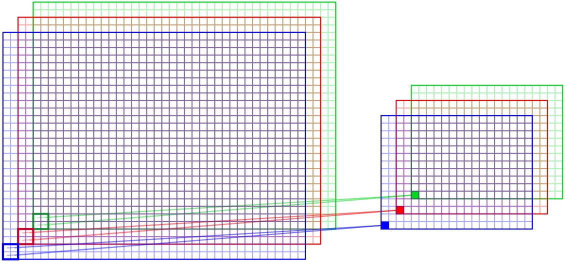
- Direct connections are sparse
- But receptive field can span most/all of image

Feature extraction approach is essentially part-based

- Earlier layers learn more local features
- Learn local features (e.g. presence of eye or nose)
- Learn global features (e.g. presence of face) from those

Most CNNs include pooling layers

- Reduce spatial resolution (but not depth) of input
- To reduce number of parameters and computations



Most common form is 2×2 max-pooling with stride 2

- $W_l = W_{l-1}/2$, $H_l = H_{l-1}/2$, and $c = 2$
- Output of neuron i is $\max(\mathbf{X}_i)$ with $\mathbf{X}_i \in \mathbb{R}^{2\times 2}$

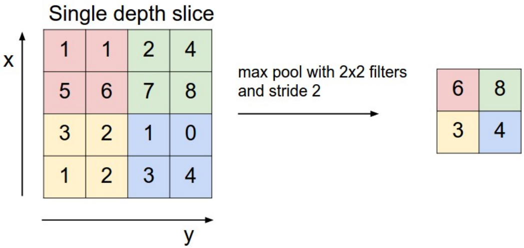


Image from cs231n.github.io

Convolutional layers are fundamental layer type in CNNs

Most CNNs also include pooling layers

Locally connected layers are used in some applications

- If features should vary with spatial location

Many other kinds of layers have been proposed

- Most are not used in current CNNs

Next lecture

- More on layer composition
- State-of-the-art CNN architectures

Bibliography

[1] *Deep learning*, 2016, [Online]. Available: http://www.deeplearningbook.org.