Learning to Understand Images

Topic 14

Week 12 – April 3rd, 2019

Topic 14: Learning Image Filters

- **Problems in Computer Vision**
- Computer Vision before 2012
  - Object Recognition
  - Object Class Recognition
  - Deformable Parts Models
- Deep Learning: From Image to Classification
  - Neural Networks
  - Convolutional Neural Networks

Problems in Computer Vision

Object Recognition: What specific object is in this image?

2008 Silver Toyota Camry

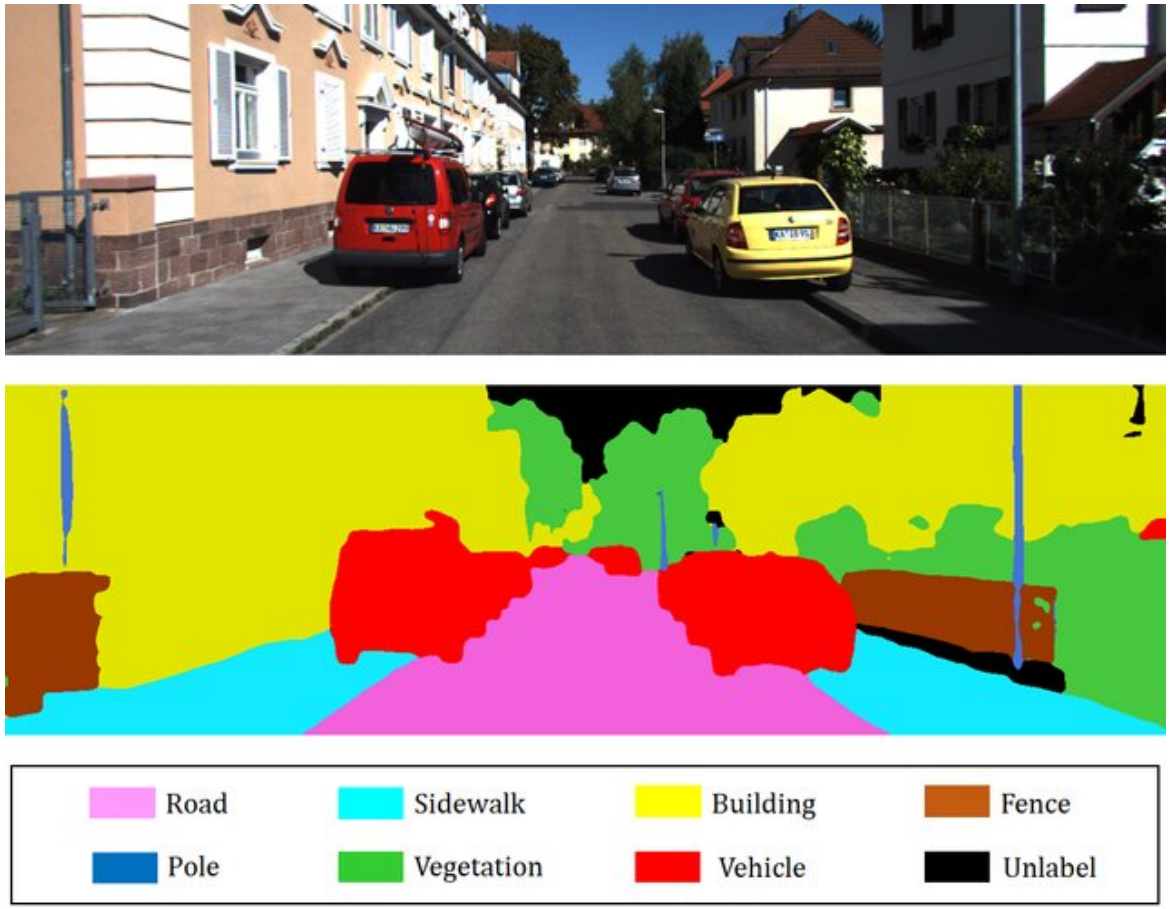
Problems in Computer Vision

Object *Class* Recognition: What type of object is in this image?

Car

Problems in Computer Vision

Semantic Segmentation: Label every pixel in an image



Jeong et al., Sensors, 2018

Problems in Computer Vision

Object Detection: What type of *object**s*** are in this picture, and where?

Car

Problems in Computer Vision

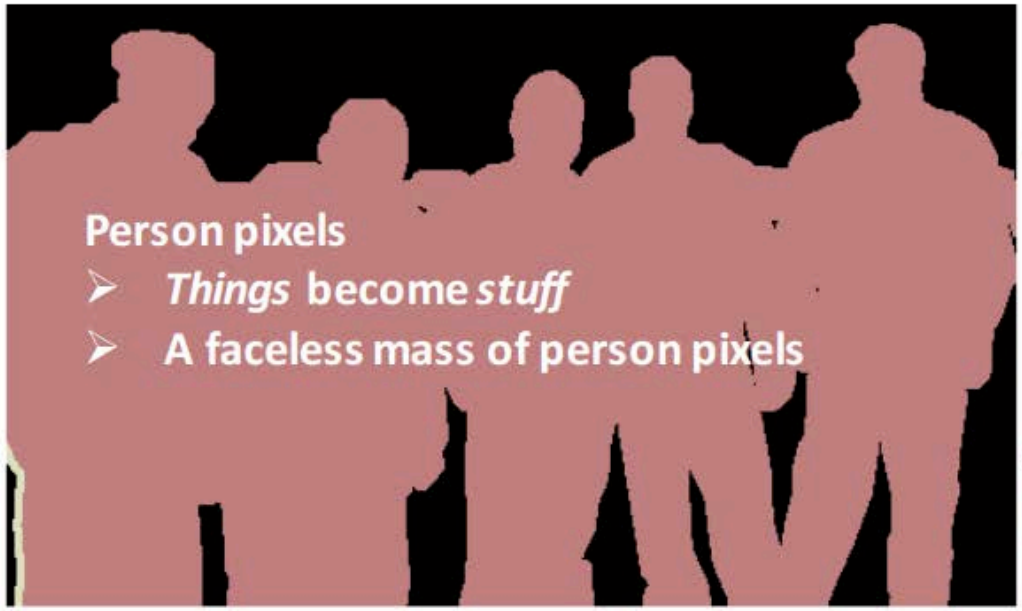
Object Detection: What type of *objects* are in this picture, and where?

More challenging than it seems! Self-occlusion, clutter, etc

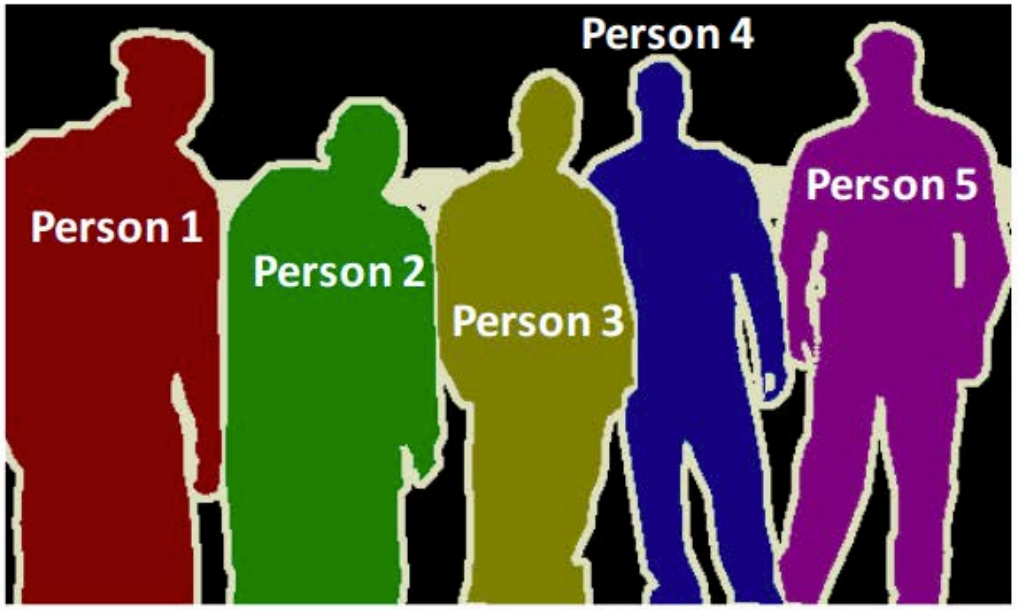
Pascal VOC/Ross Girshick

Problems in Computer Vision

Instance Segmentation: Label every pixel, and identify different object instances



(Semantic Segmentation)

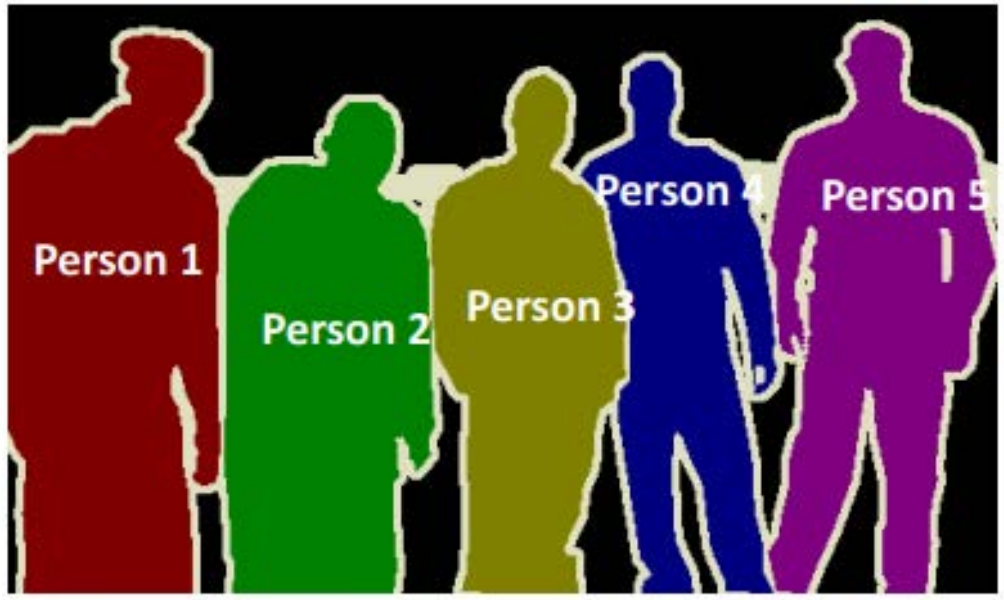


(Instance Segmentation)

Pascal VOC/Ross Girshick

Problems in Computer Vision

Counter-intuitively counting objects is much harder for deep networks than object classification!



Pascal VOC/Ross Girshick

Topic 14: Learning Image Filters

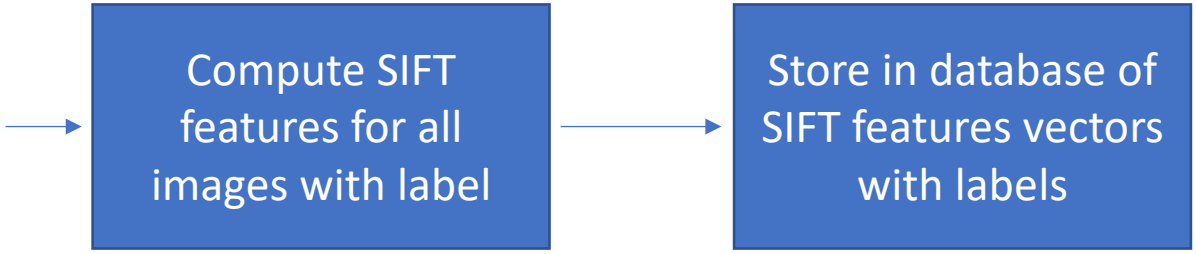
- Problems in Computer Vision
- **Computer Vision before 2012**
  - Object Recognition
  - Object Class Recognition
  - Deformable Parts Models
- Deep Learning: From Image to Classification
  - Neural Networks
  - Convolutional Neural Networks

Object Recognition using Local Features

- ***Object recognition**:* recognizing a specific instance of an object, e.g. **recognizing a specific model of car from a specific year and colour**
- Input: Image on right
- Output: 2008 Silver Toyota Camry
- Appearance is similar across images, we can use local features such as SIFT to try and match to previous objects seen

Object (Car) Recognition Training Pipeline

Image Dataset



- Collect many images of different makes/models of cars in different poses
- Label each set of SIFT features with object label, e.g. “2008 Silver Toyota Camry”
- Train a classifier (e.g. SVM, random forest, neural network) on feature vectors

Object (Car) Recognition Pipeline

Image Input

Compute SIFT features for this image

Compute Local Features

Nearest Neighbours

Classification

2008 Silver Toyota Camry

Label

- Calculate SIFT features for current image
- See what set of SIFT feature vectors in DB are close to this image
- Do some extra filtering based on pose given by close SIFT features

Object Class Recognition

- ***Object Class recognition****:* recognizing a general class of objects, e.g. **recognizing a car in general**
- Input: Images on right
- Output: Car

Object Class Recognition

- *Why is this difficult?*
  - Cars have a wide variety of appearance
  - SIFT features only allow us to compare images by the local appearance
  - Even the same car can have different appearance, e.g. car doors opened or closed – real-world objects are often *deformable*
- Why don't we just memorize all possible cars, i.e. treat this as object recognition
  - Infeasible in general, too many types
  - Humans can recognize cars they've never seen – we want to *generalize*

Object Class Recognition

- What is common to all cars?
- Wheels, headlights, bumpers, rims, door handles (usually) – i.e. parts!

- **Idea: Why don't we look for relevant parts?**

Part-based Models

- How do we detect parts?
- Typically parts are based on local features such as HoG or templates
- Let's assume we can easily find common parts that belong to cars – have we solved object class recognition?

Part-based Models

- Are these cars?
- If we classify an image based on seeing a couple of wheels, we are going to get yes!

- So we need to ensure there are different parts present, e.g. bumpers, door handles, headlights, etc.

Part-based Models

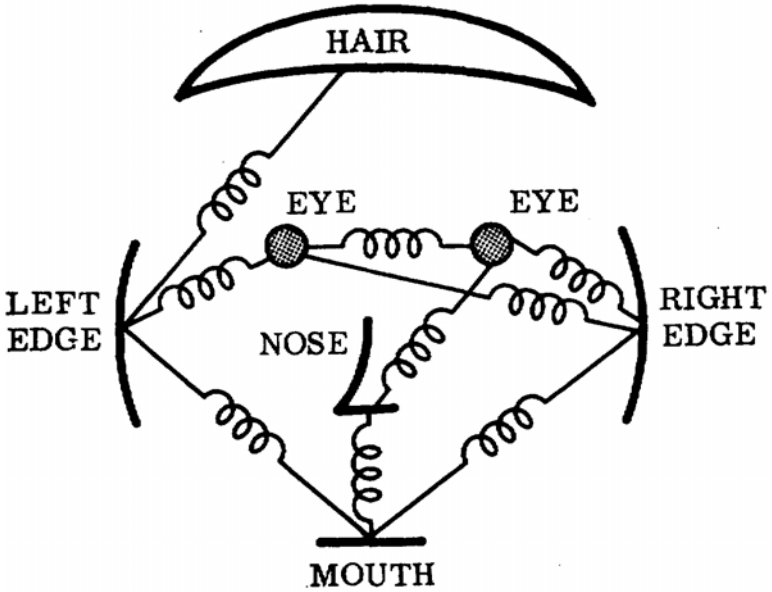
- OK, so we find an image with all relevant parts – done right?
- This image contains eyes, a mouth, and a nose.
- We can probably agree it's not a face!

- It is not enough for the scene to just contain the parts – they **must be in the correct locations relative to other parts**!

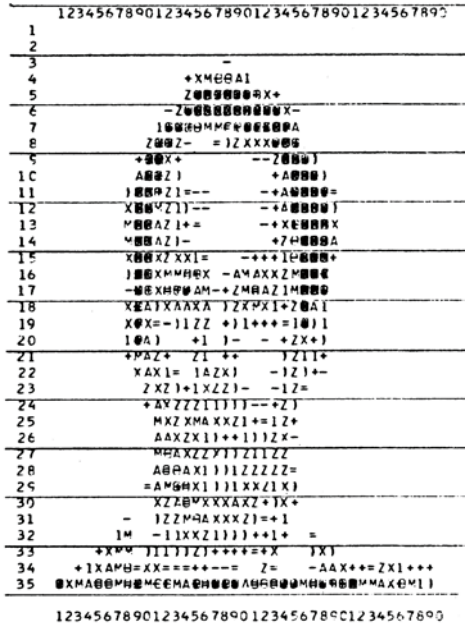
Deformable Part Models

- It is not enough for the scene to just contain the parts – they **must be in the correct locations relative to other parts**!

- This is the basic idea behind *deformable part models (DPM)*
- Old idea – “Pictorial Structures” 1972
- Need some flexibility for locations to allow for deformation/difference between people’s faces

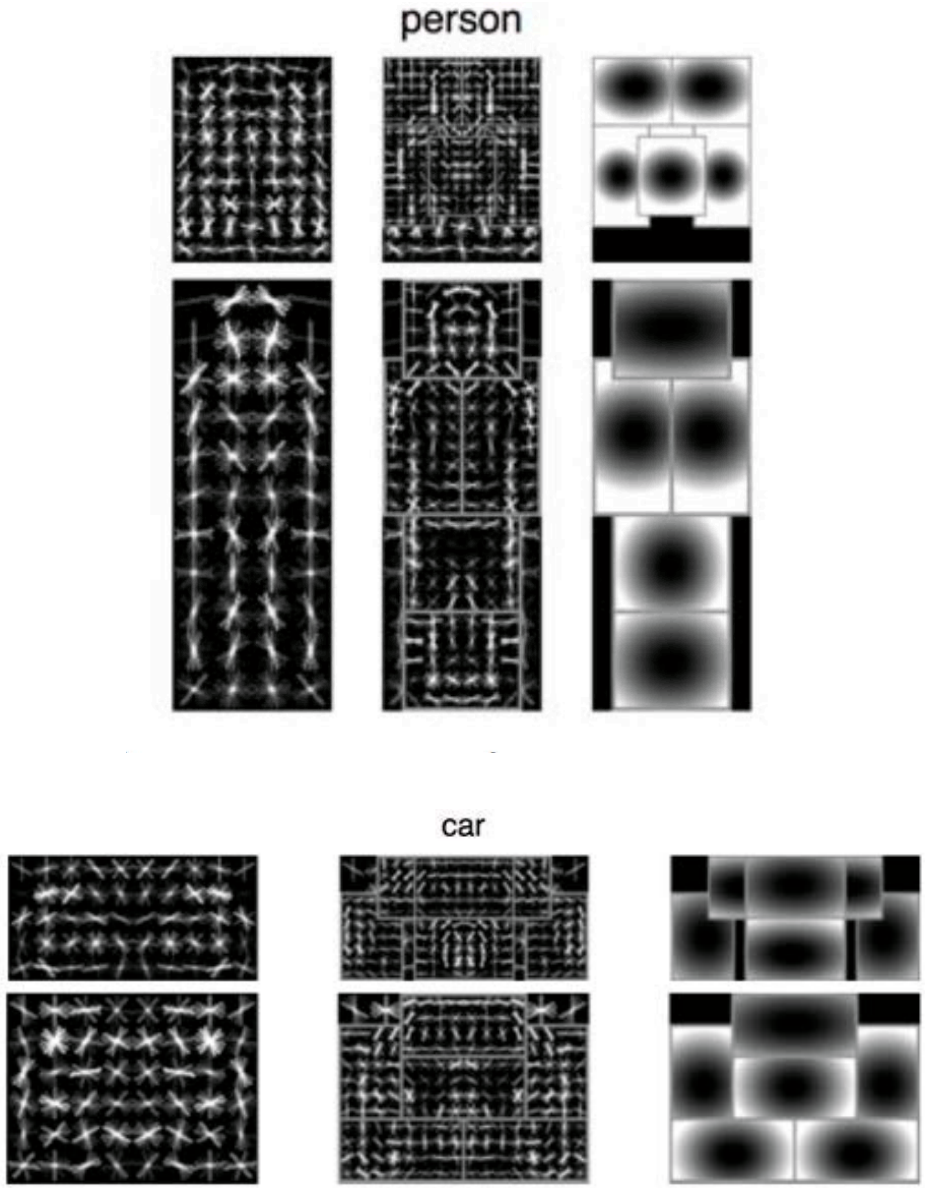


Fischler & Elschlager 1972



Deformable Part Models

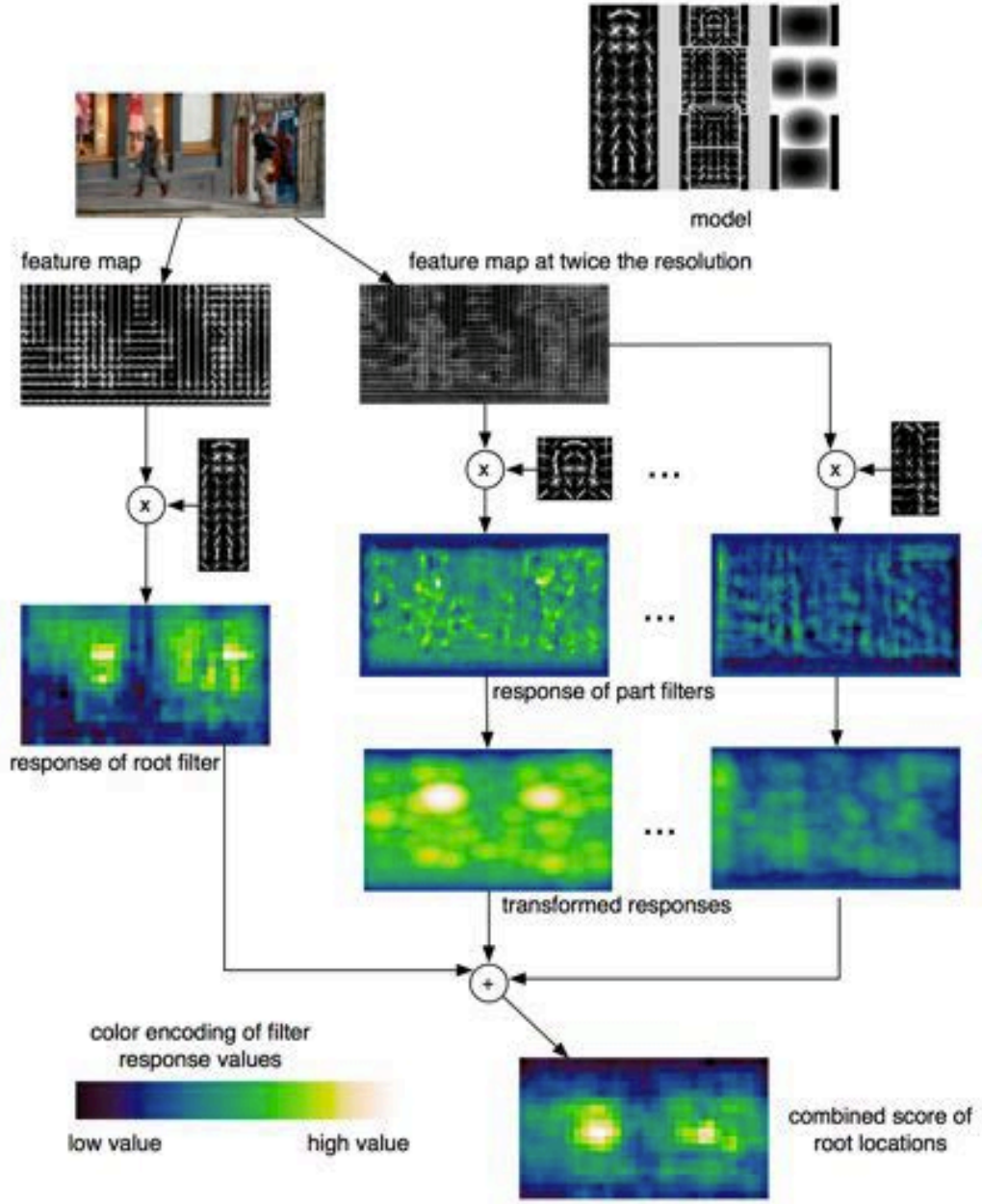
- Basis of many of the works aimed at solving object class recognition and the even harder problem of *object detection* before 2012
- General idea: use a pyramid of HoG features at different scales to find parts
- Weight their relative locations according to a model of where they should be (for different poses/views)



Felzenswalb et al., 2010

Deformable Part Models

- Have to decide how many parts to use a-priori
- Need to learn different models for drastically different poses of object
- DPMs are very expensive to train/optimize – based on graphical models



Felzenswalb et al., 2010

Topic 14: Learning Image Filters

- Problems in Computer Vision
- Computer Vision before 2012
  - Object Recognition
  - Object Class Recognition
  - Deformable Parts Models
- **Deep Learning: From Image to Classification**
  - Neural Networks: A Crash Course
  - Convolutional Neural Networks

ImageNet

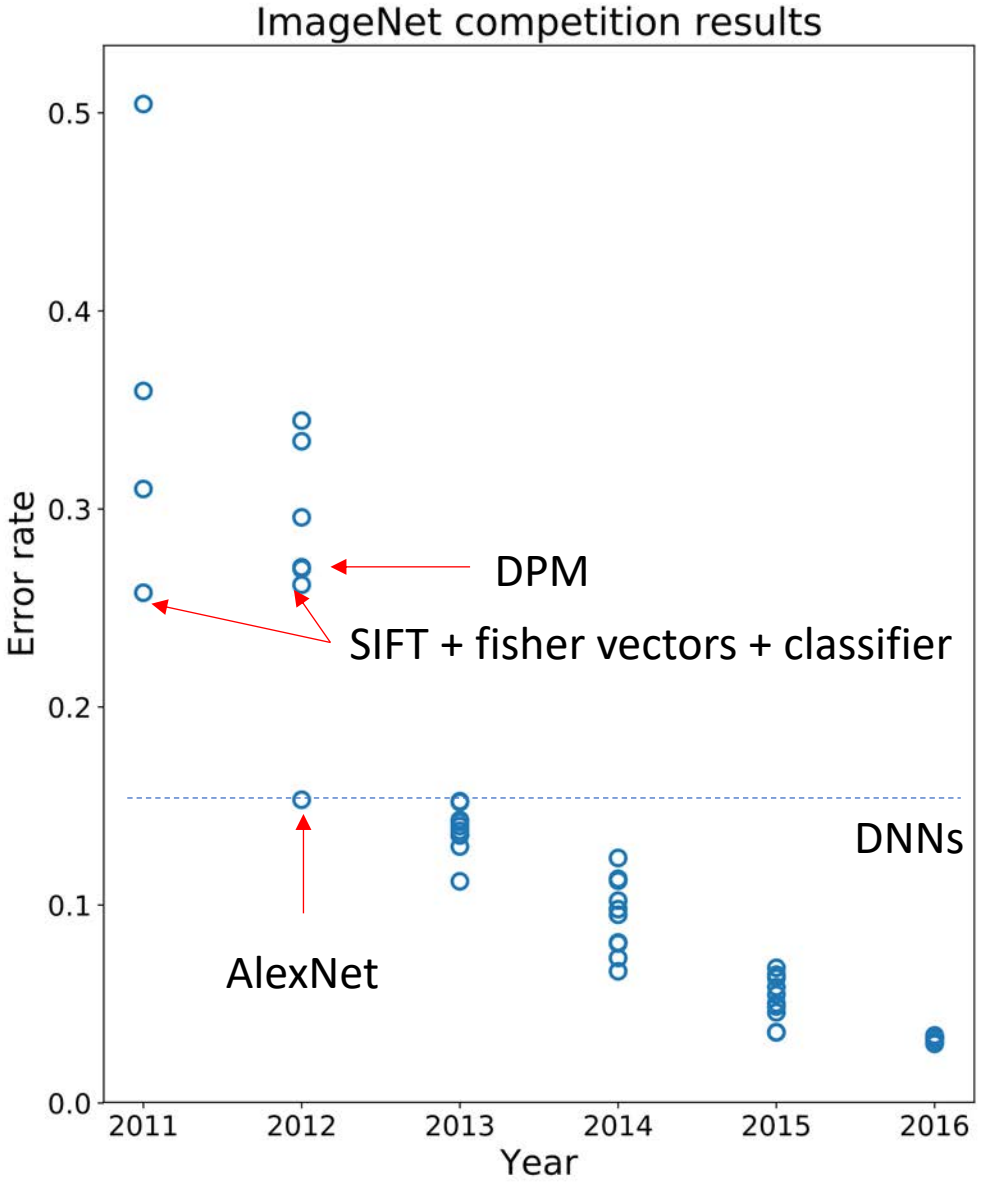
- Often challenges (i.e. contests) push state of the art
- ImageNet Large Scale Visual Recognition Challenge (ILSVRC typically just called imagenet)
- Many object class recognition datasets had been created (e.g. CIFAR 10/100), but ImageNet dataset was of a different order of magnitude!
- Training data:
  - 1 Million total training images
  - Web-resolution images (typically > 100x100)
  - 1000 object classes
  - 1000 images per class
  - 50,000 validation + 50,000 test

Fangyu Cai (Medium)

Deng, Jia; Dong, Wei; Socher, Richard; Li, Li-Jia; Li, Kai; Fei-Fei, Li, "ImageNet: A Large-Scale Hierarchical Image Database" , 2009 conference on Computer Vision and Pattern Recognition

Deep Learning

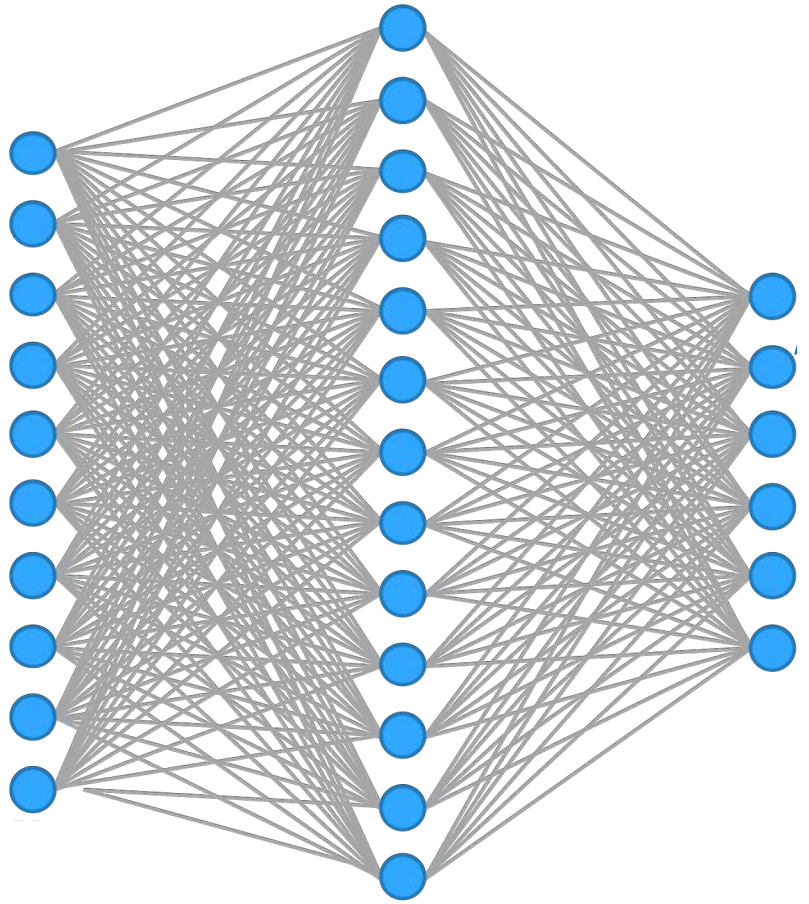
- First challenge in 2010
- AlexNet (from U of T!) was a massive leap in performance, and came out of the blue
- Neural networks had been out of favour for many years
- By 2015 had surpassed human accuracy



From Wikipedia (modified)

Neural Networks: A Crash Course

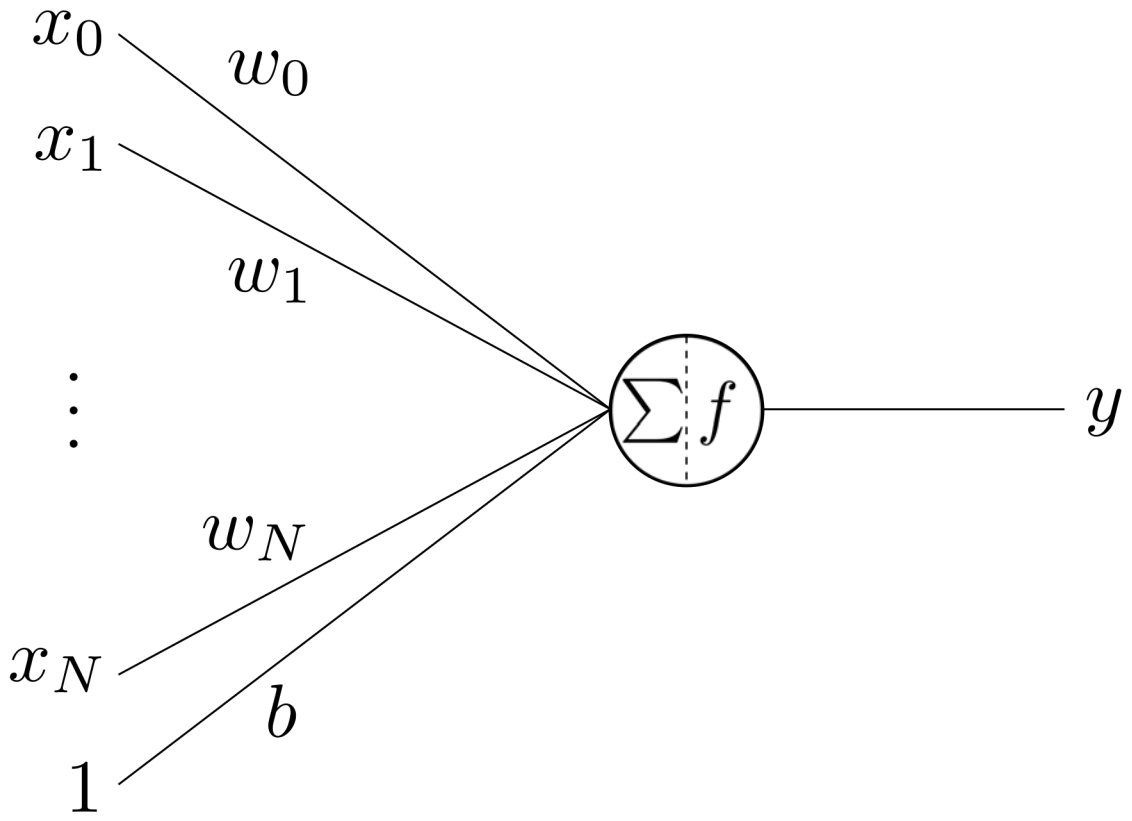
- Artificial neural networks were first proposed in the 1940's MCulloch & Pitts
- However, what we call neural networks now are based mostly on research in the 1980s
- In particular backpropagation – a way of training networks with many layers (Rumelhart, Hinton 1986)



Neural Networks: A Single Neuron

- $\boldsymbol{x}$ is an input vector
- $\boldsymbol{w}$ are the learned *weights*
- f is an *activation function*
- b *is a bias*
- Neuron's output is a scalar:

$$y = f\left(\sum w_i x_i + b\right)$$



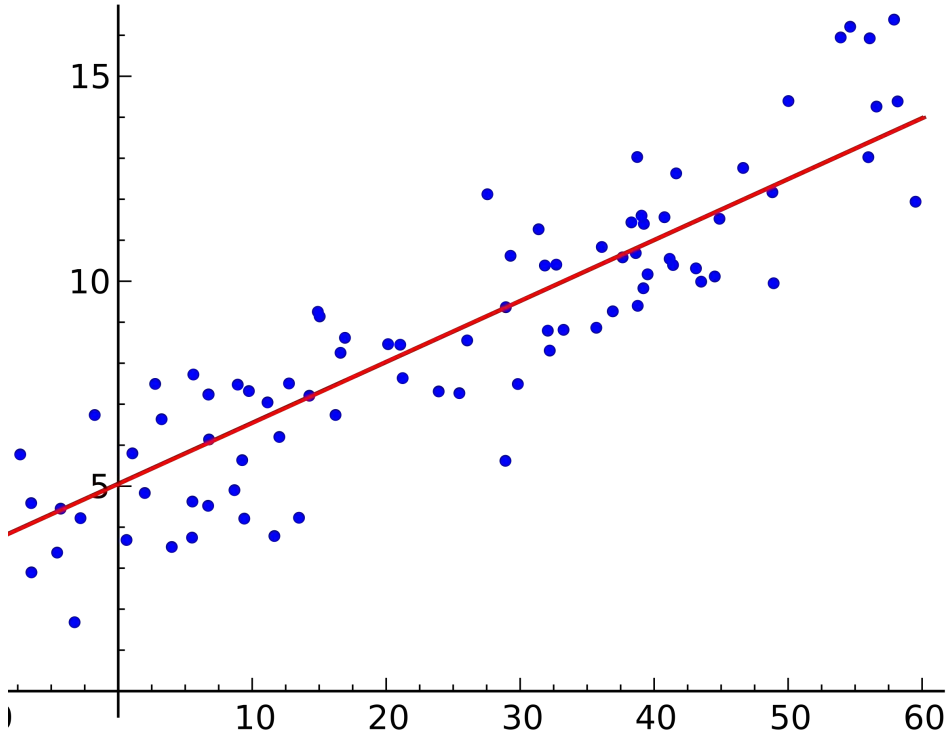
Neural Networks: A Single Neuron

$$y = f\left(\sum w_i x_i + b\right)$$

- If $f(x) = x$ our neuron describes a line
- With a 1D input, we get:

$$y = w_0 x_0 + b$$

- w_0 is slope, b is offset
- In fact, at its very simplest form, fitting a neuron to a dataset is linear regression!



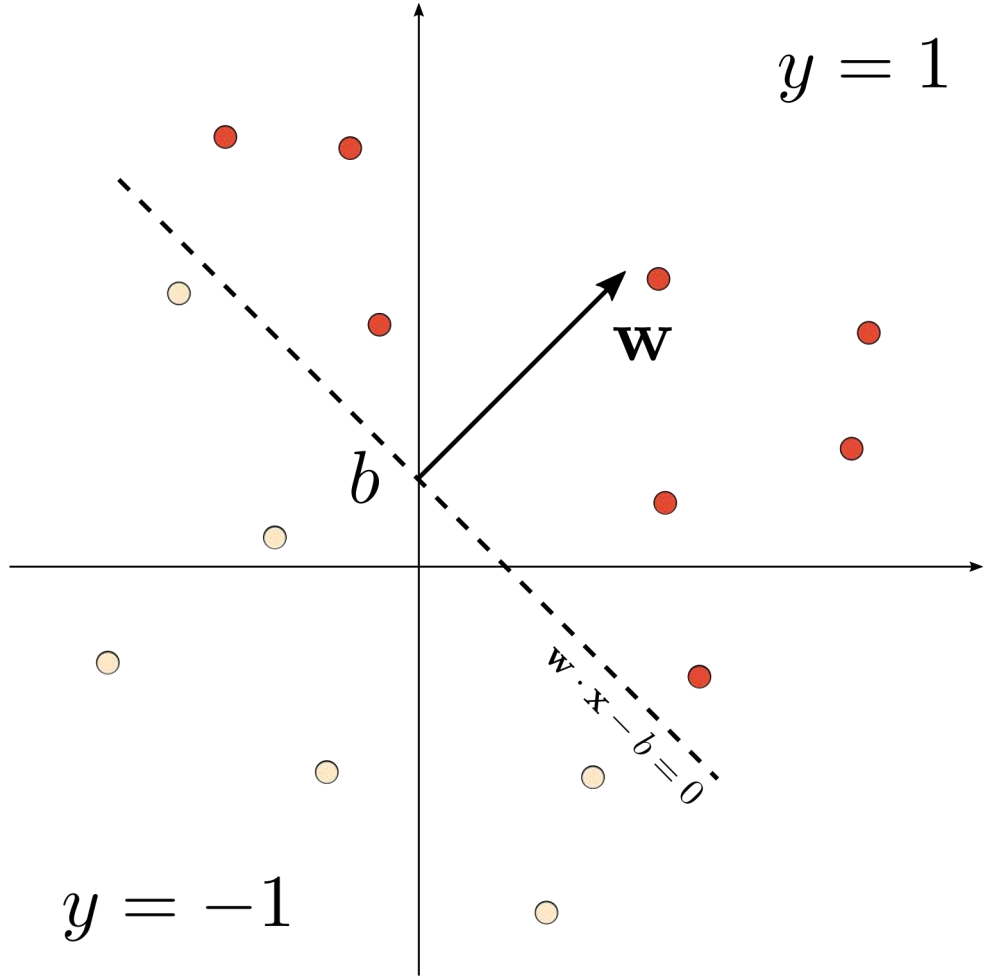
Neural Networks: Linear Activation Function

$$y = \sum w_i x_i + b$$

$$y = \boldsymbol{w} \cdot \boldsymbol{x} + b$$

(in vector form)

- This in general describes a *hyperplane (2D line, 3D plane)*
- *A hyperplane splits the whole of an N-dimensional space into 2!*



Neural Networks: Binary Threshold Functions

$$y = f(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

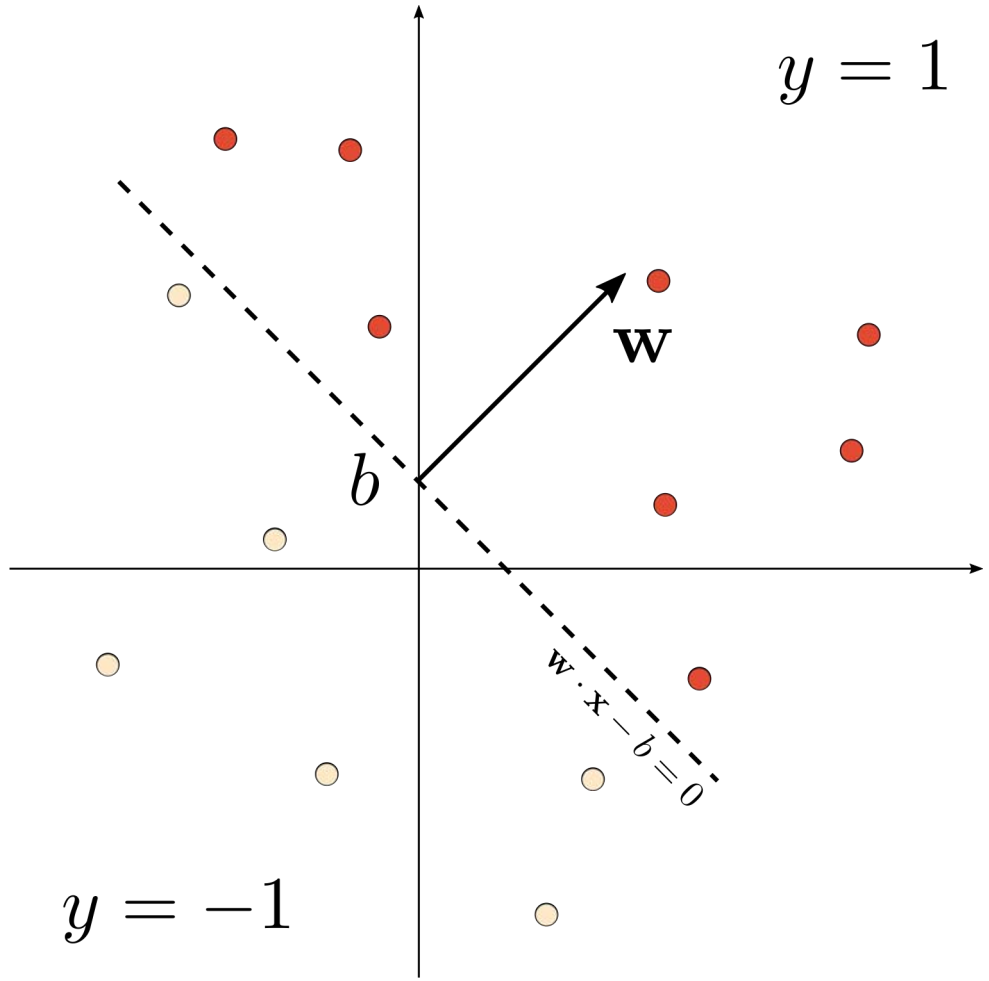
(in vector form)

- Until the 70s neurons were binary, based on sign or threshold:

$$f(x) = \text{sign}(x)$$

$$f(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases}$$

- These give a *decision boundary*



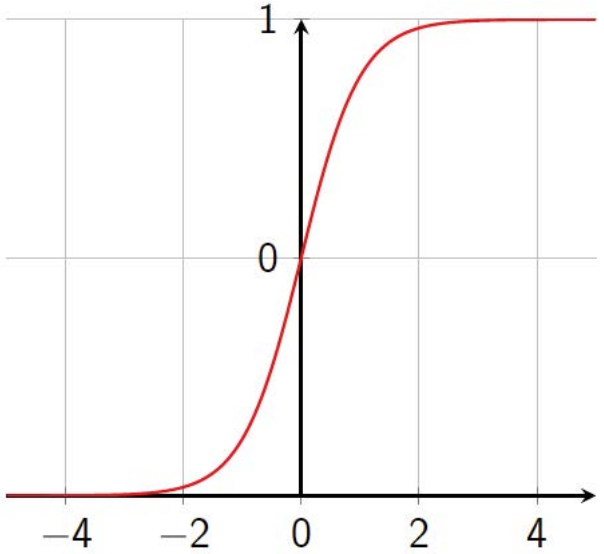
Neural Networks: Activation Functions

$$y = f(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

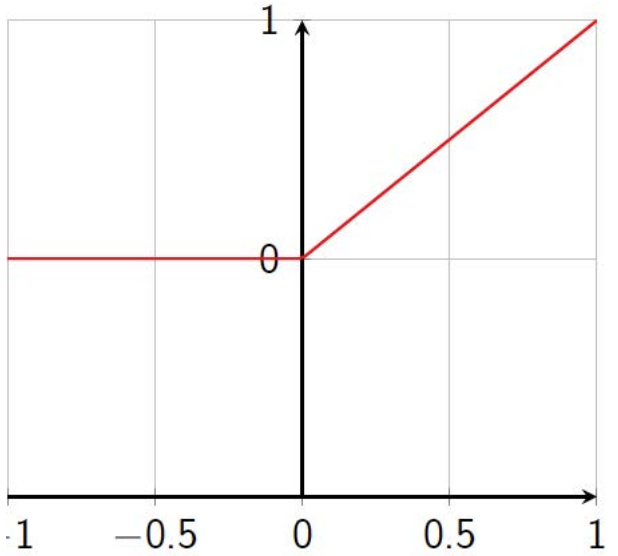
(in vector form)

- Modern neural networks use activation functions with well defined derivatives
  - various types of sigmoids
  - rectified linear unit (ReLU)

$$f(x) = \tanh(x)$$

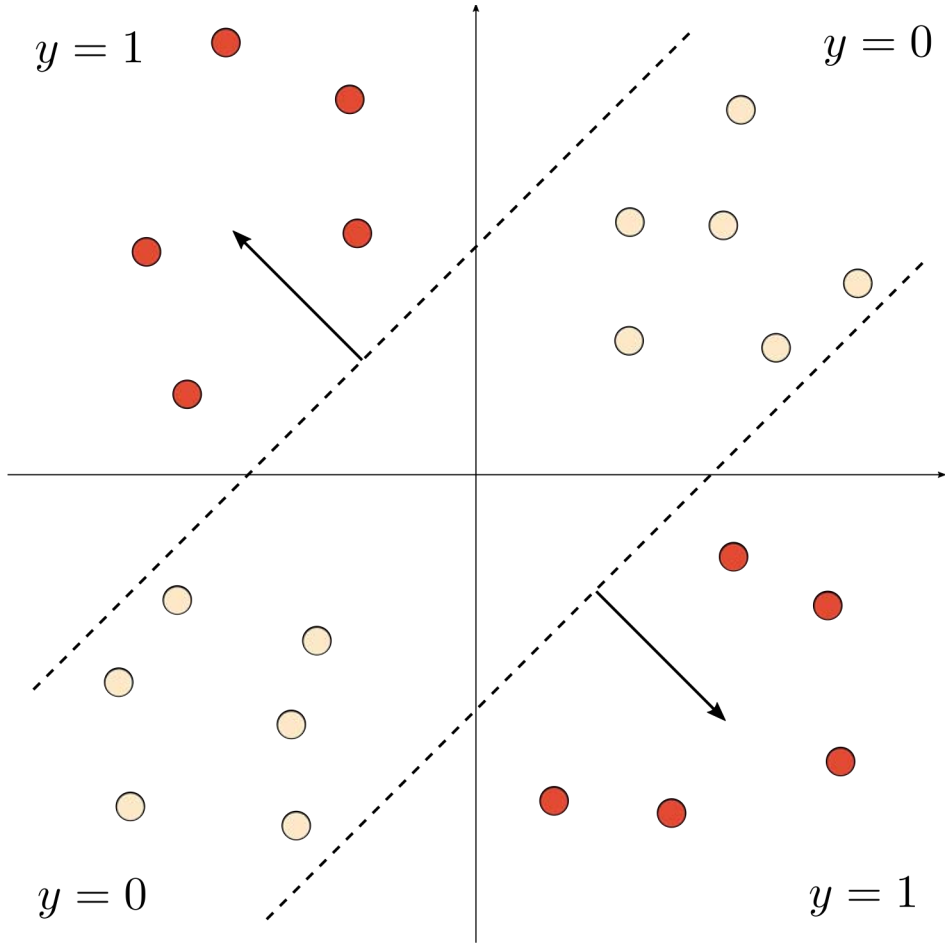


$$f(x) = \begin{cases} 0, x < 0 \\ x, x \geq 0 \end{cases}$$



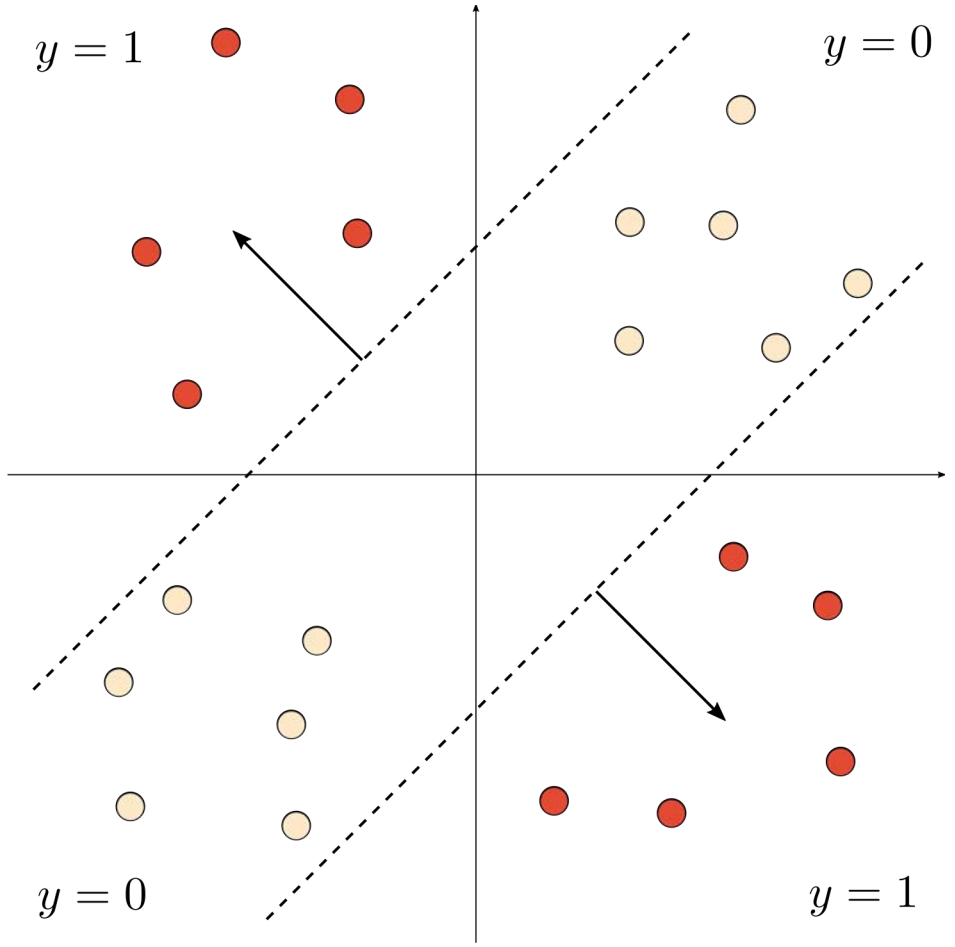
Neural Networks: Single Layer

- A decision boundary is a very simple classifier – a single layer neural network is a collection of these
- However, many problems need more than a single decision boundary
- The most famous such problem is the XOR function which needs two hyperplanes



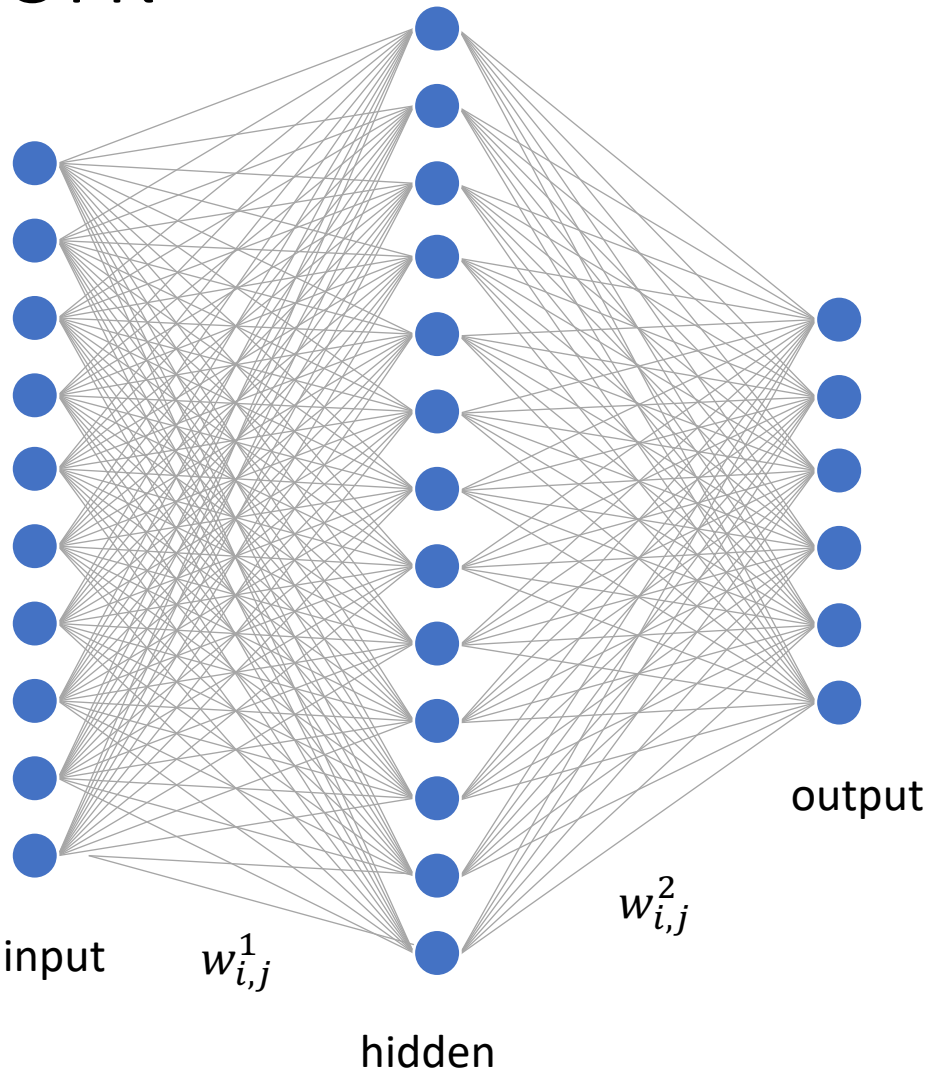
Neural Networks: Multiple Layers

- By having multiple layers, we can combine decision boundaries!
- In fact it has been shown that a neural network with a single *hidden layer* (i.e. a neural network with 2 or more layers of neurons) with infinite width is a *universal function approximator*
- But how to train neural networks of multiple layers was not obvious until the late 80s



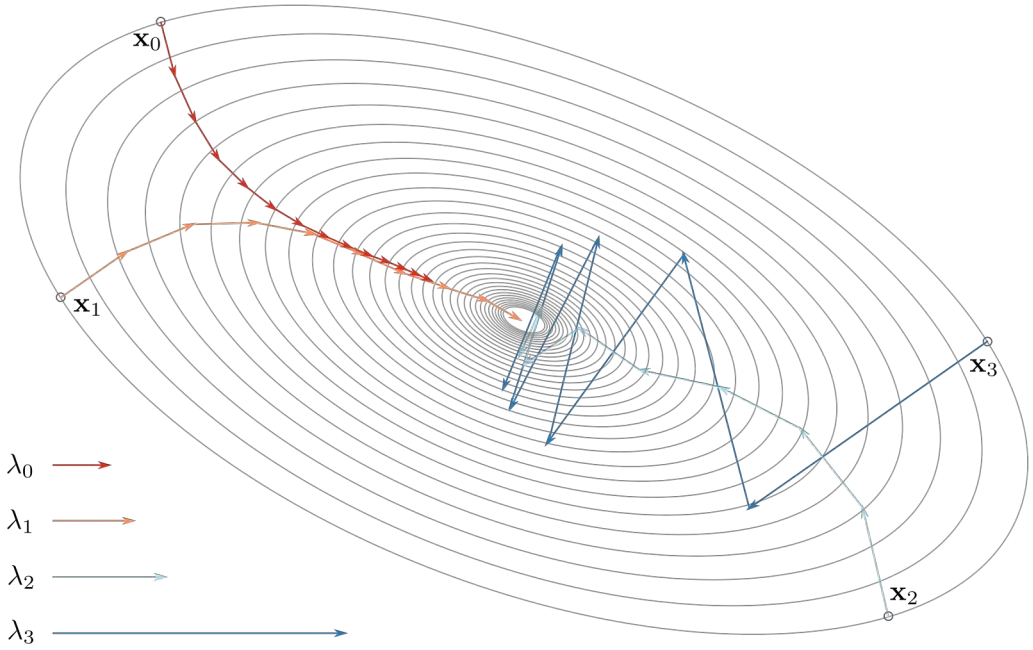
Fully-Connected Neural Network

- A general neural network is *fully connected* – i.e. every neuron has a weight for every neuron/input in a previous layer
- The number of weights can be massive!
- We know what the input is, and what the output is, what is the hidden representation?
- An ***embedding:*** like learning a representation in a different basis, but with a non-linear transformation!



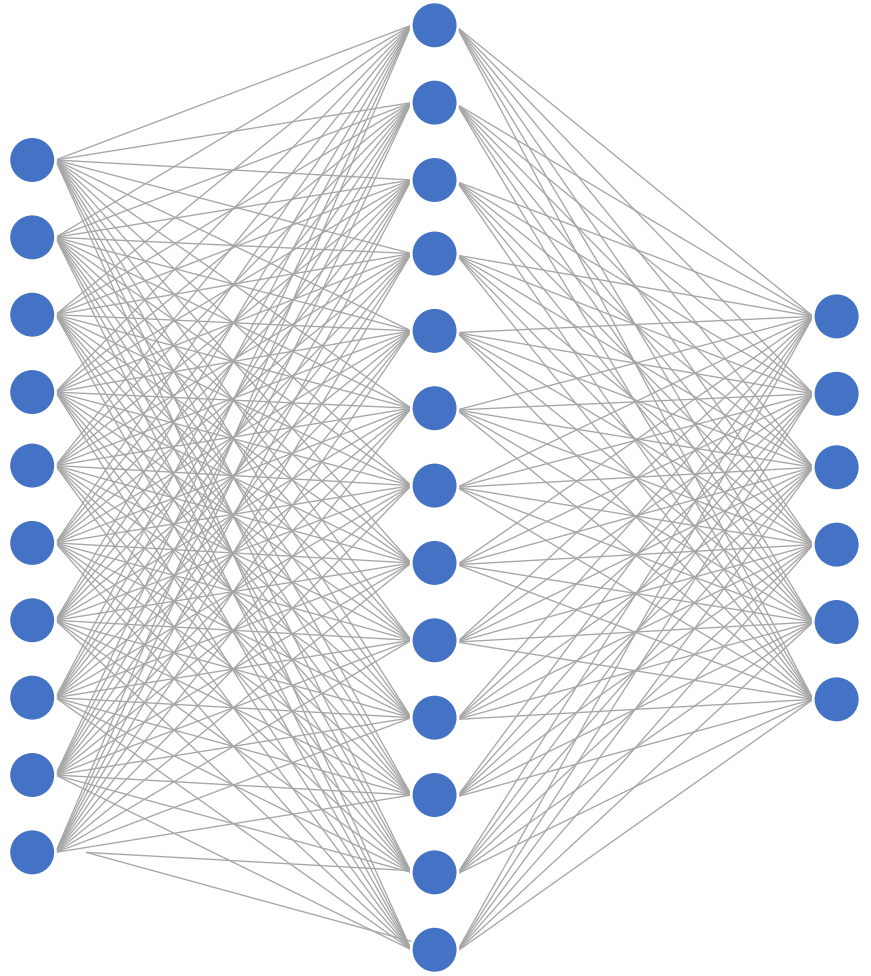
Neural Networks: Training with SGD in One Slide

- Training data and labels:
$$X = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^n\}, T = \{t^1, t^2, \cdots, t^n\}$$
- Output of neural network:
$$Y = \{y^1, y^2, \cdots, y^n\}$$
- We have the true labels, and the output of the network, we can calculate an error E, for example Mean Squared Error (MSE) for regression:
$$E = \frac{1}{n}\sum_i \left(t^i - y^i\right)^2$$
- *Backpropagation* lets us calculate the partial derivative of any weight in the model W.R.T. the error, no matter which layer it is on, and out pops our old friend the ***gradient***!
$$\nabla E = \frac{\partial E}{\partial w_i}$$
- Update weights using *gradient descent* (λ is learning rate)
$$\boldsymbol{w}^{t+1} = \mathbf{w}^{\mathrm{t}} - \lambda \nabla E$$

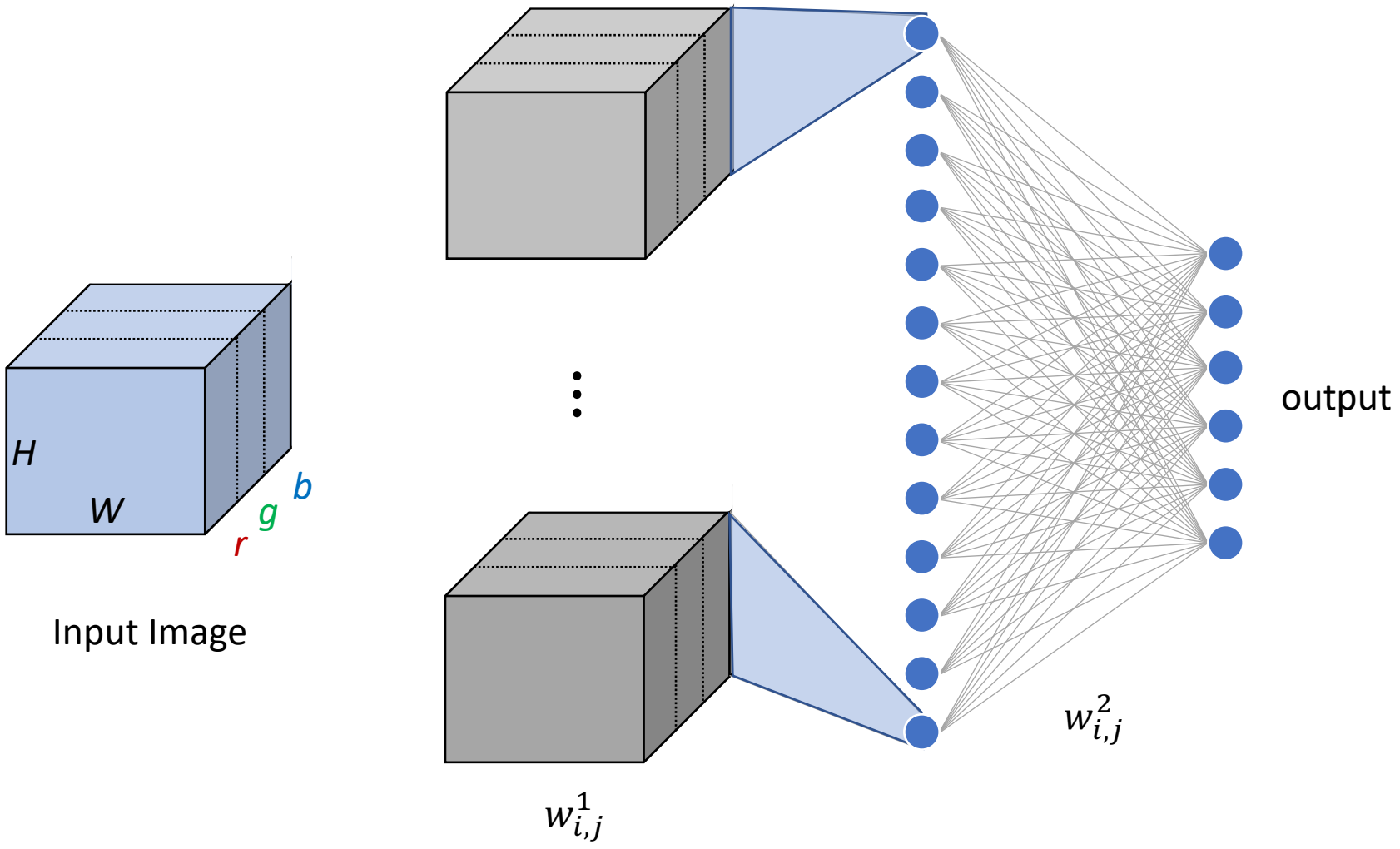


Neural Networks: Image Inputs

- Let's get back to images!
- We can simply feed in an image vector as the input vector $\boldsymbol{x}$
- If we do this, *each neuron* has the same number of weights as pixels in the image!
- Assume image of M pixels and N neurons in first layer, $N \times M$ weights in first layer!
- 32×32 pixels * 10 = 10,240 weights

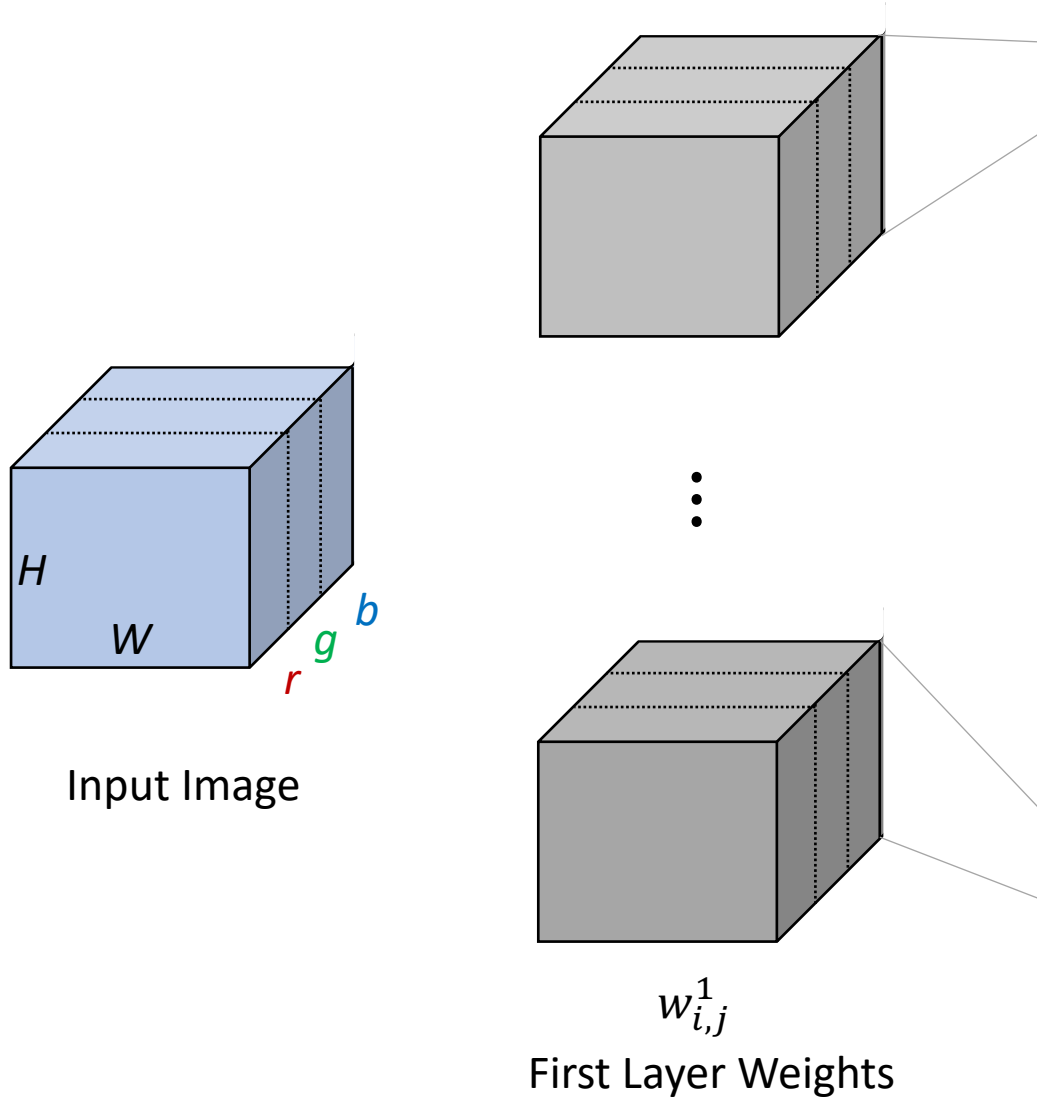


Neural Networks: Image Inputs



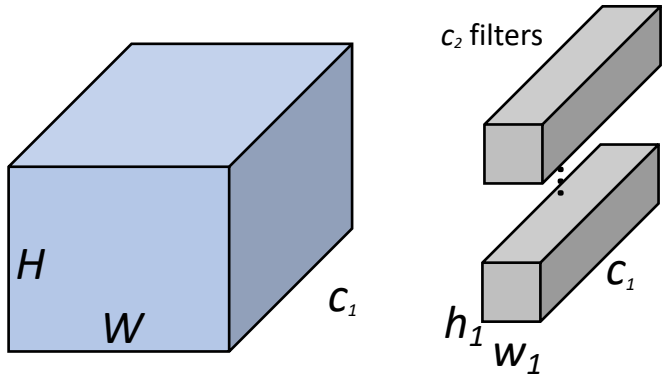
Neural Network for Images

- With a fully-connected neural network we are learning massive image-sized "filters"!
- This is a poor model – should always try to encode what we know about the domain (i.e. images) into our model
- We know local features are enough to represent an image's appearance/textures!
- Don't need whole-image connectivity



Neural Network for Images

- We know that we can model appearance well based on local neighbourhoods – i.e. image filters
- Learn spatially small filters
- But now we don't cover all the pixels spatially in the image!
- Let's apply this filter for all pixels in the image – this is basically image filtering

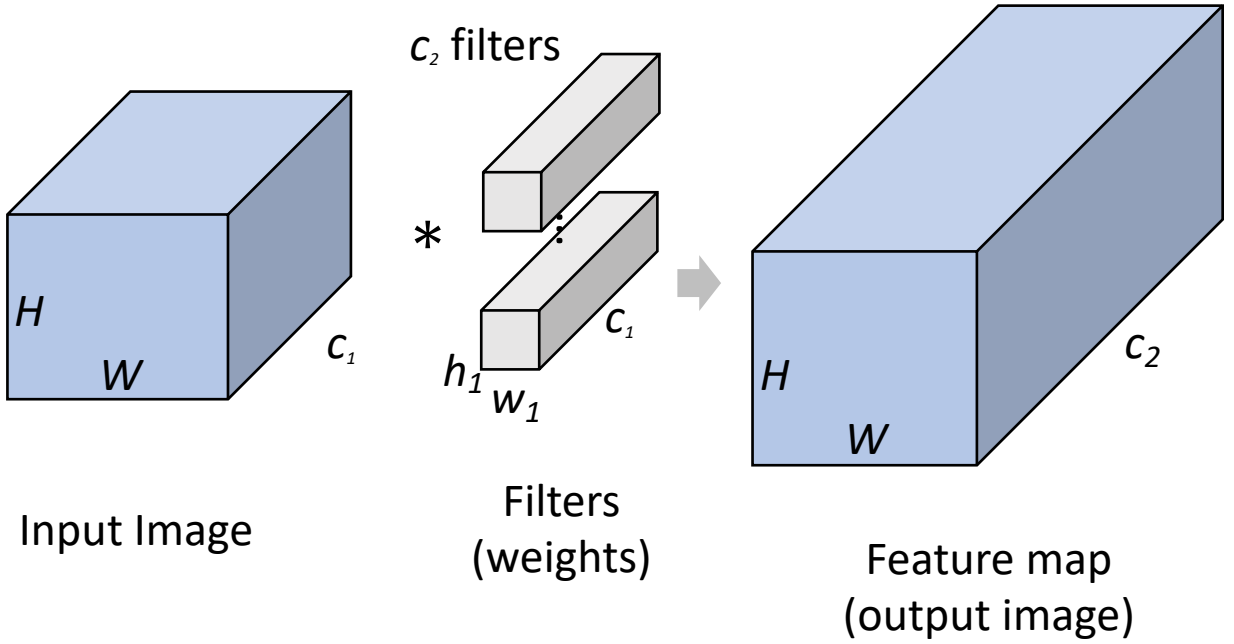


Input Image

Filters (weights)

Convolutional Layer

- This is a layer from a CNN (Convolutional Neural Network)
- Learn a set of filters that are "convolved" over every pixel spatially in the image
- Every filter outputs a single image channel *per filter*
- This stack of output images is called a feature map



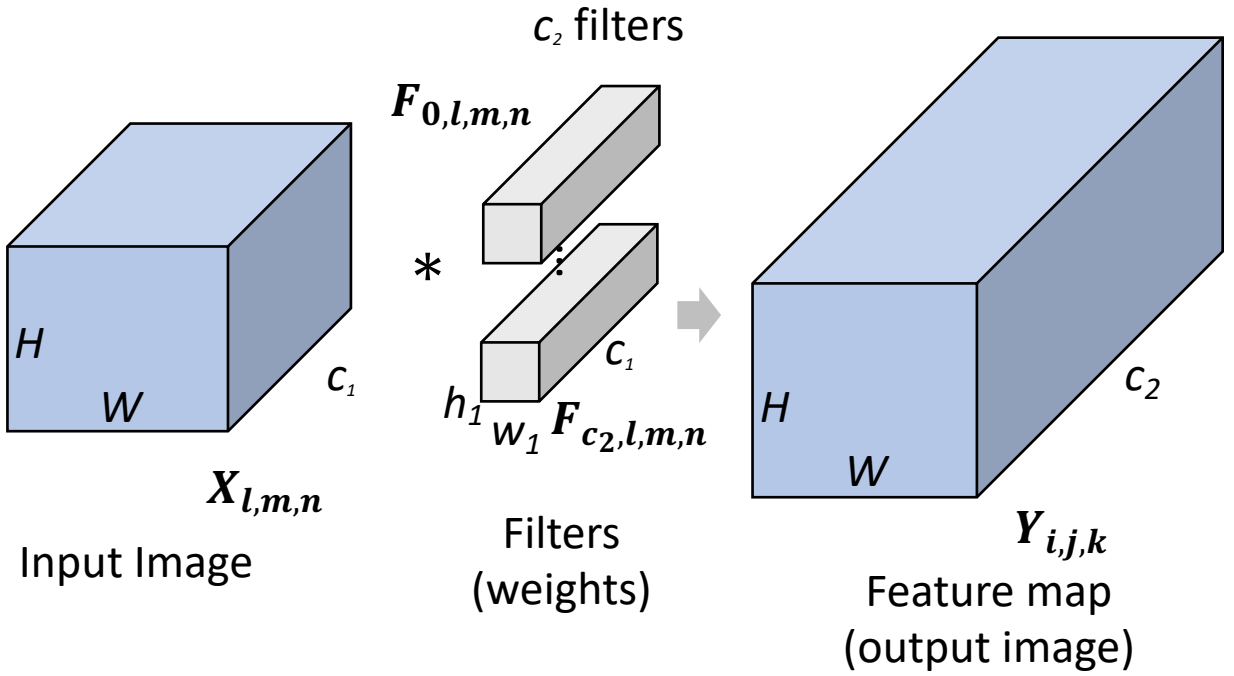
Convolutional Layer

- This is a layer from a CNN (Convolutional Neural Network)

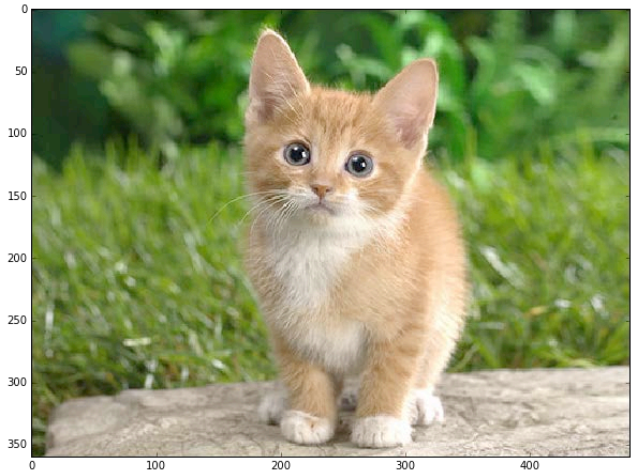
$$Y_{i,j,k} = \sum_{l,m,n} X_{l,j+m,i+n} \, F_{i,l,m,n}$$

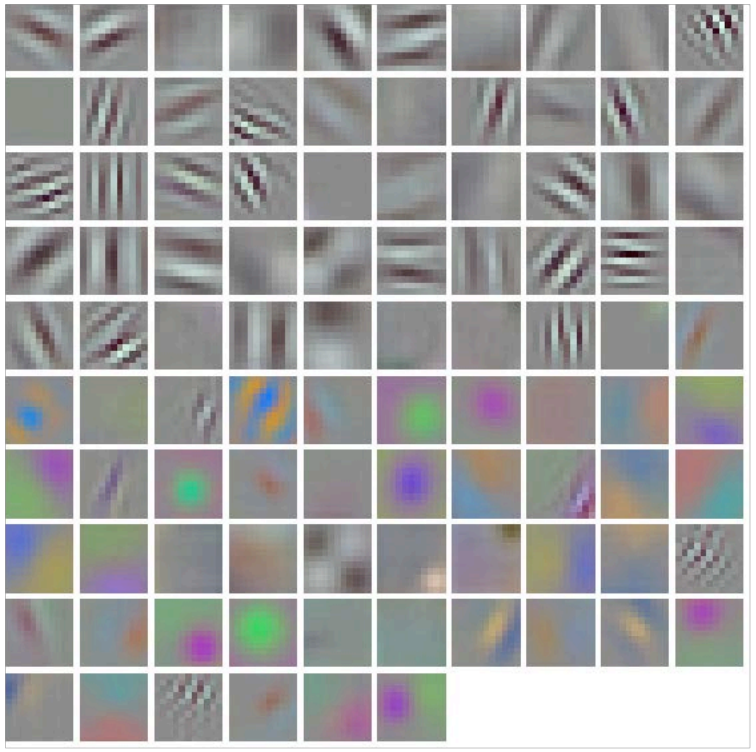
Where i, j are the spatial and k the channel coord. of the pixel in Y, m, n, l are pixel coord. in filter F

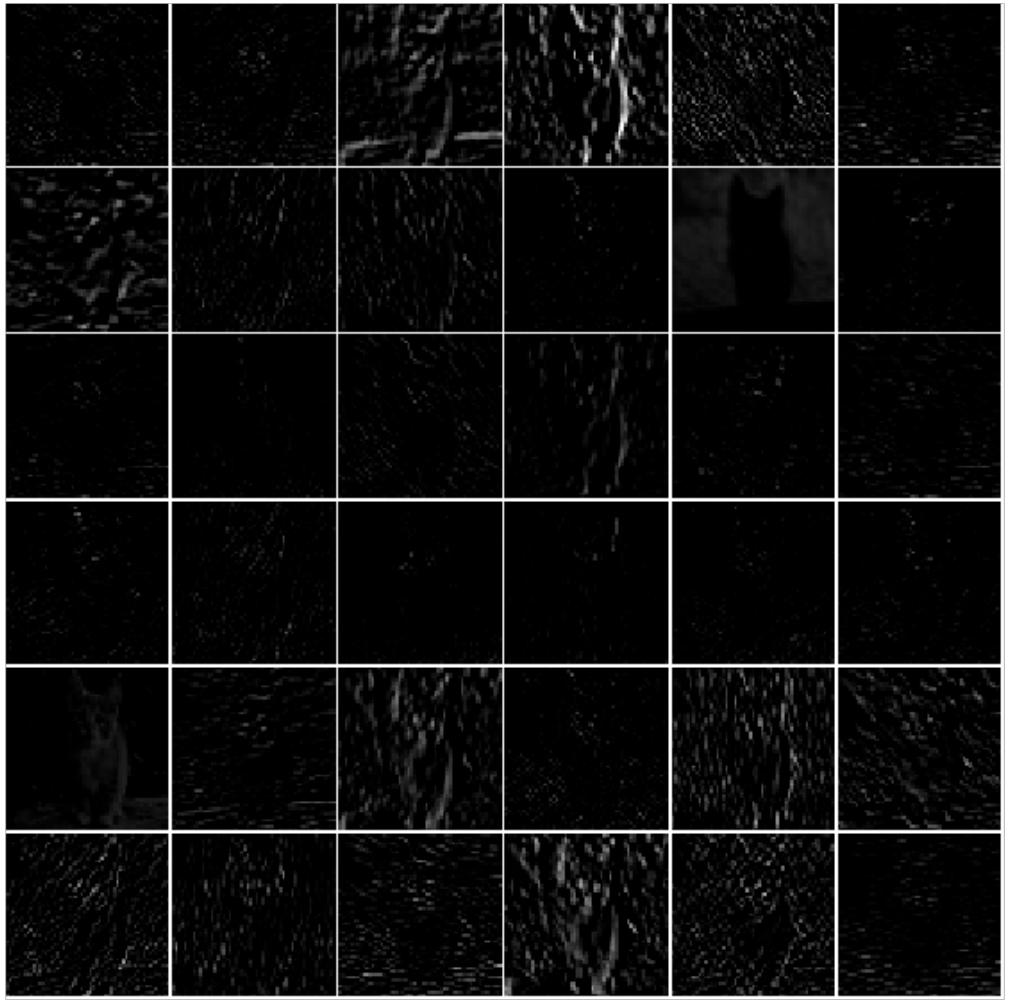
- Actually this is cross-correlation not convolution!
- You can also think of these filters as template patches and this as template matching with cross-correlation



First Layer Filters and Featuremaps

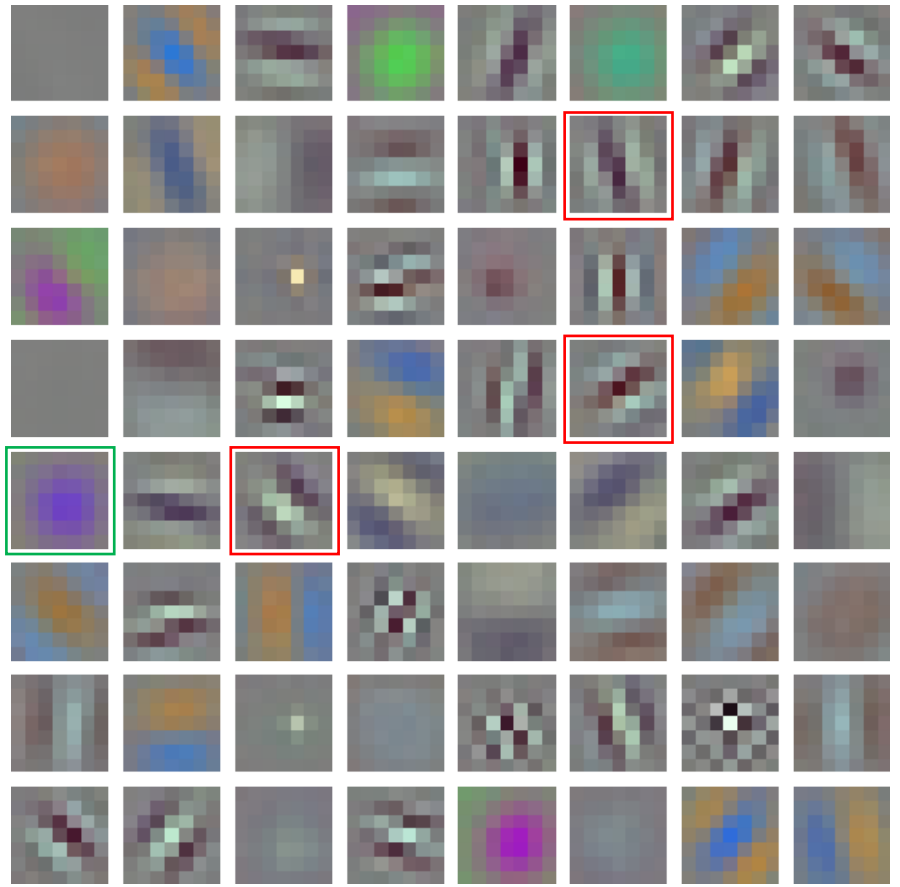






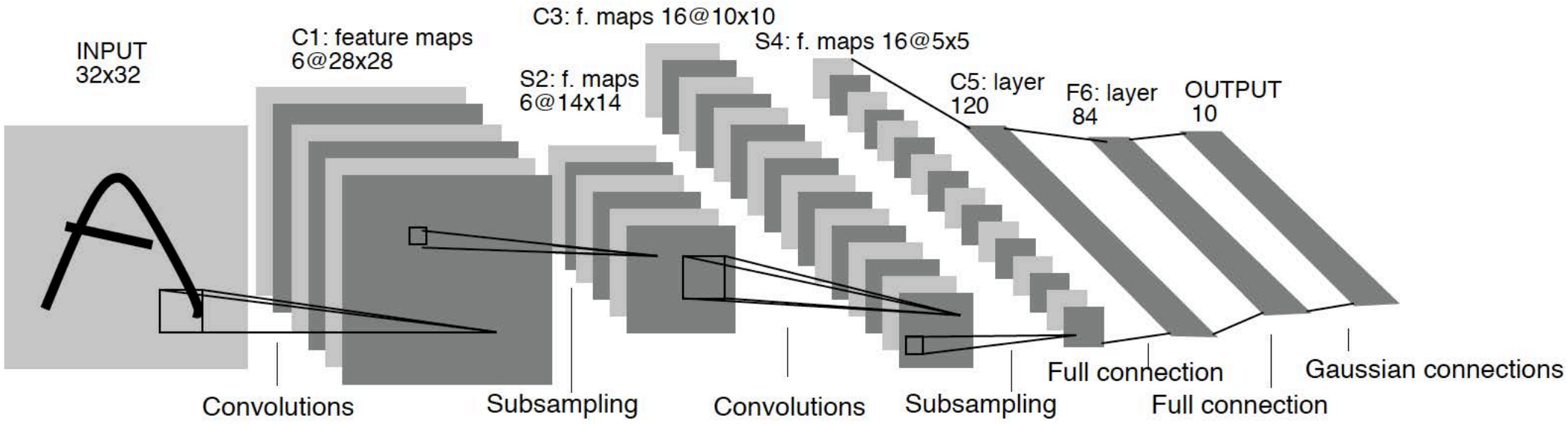
Convolutional Neural Networks

- Example of *first-layer* filters learned in a CNN
- Resemble a mixture of edges and Gaussians
- Edge like filters are *Gabor-like filters*
  - Gabor filters are a standard set of steerable band-pass filters based on Gabor wavelets
  - Sinusoids multiplied by Gaussian



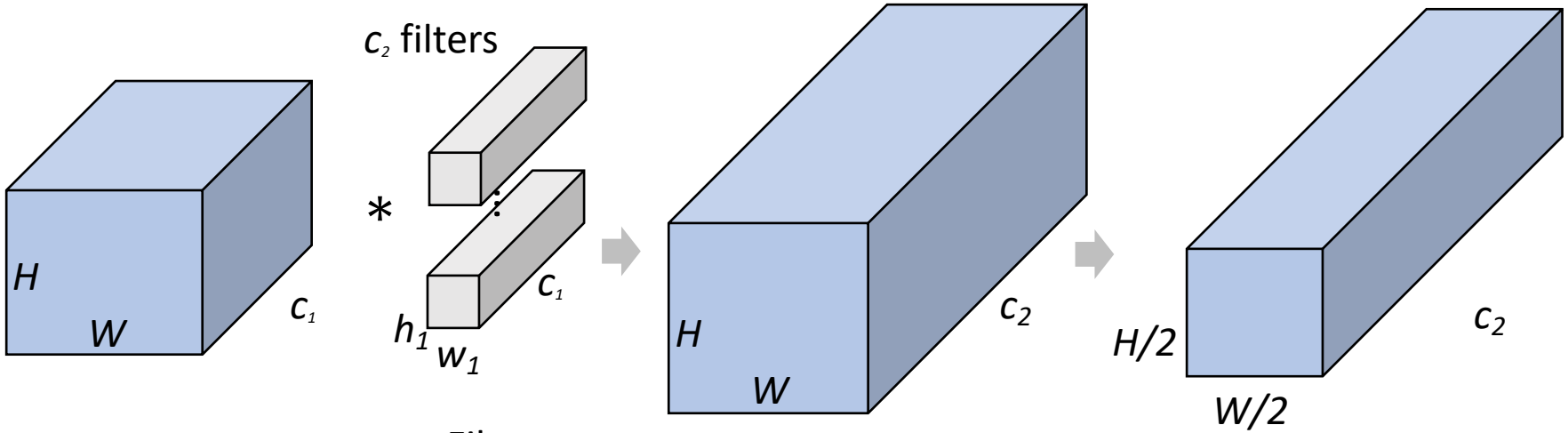
- *Gaussian-like filters are centre-surround*

Convolutional Neural Networks



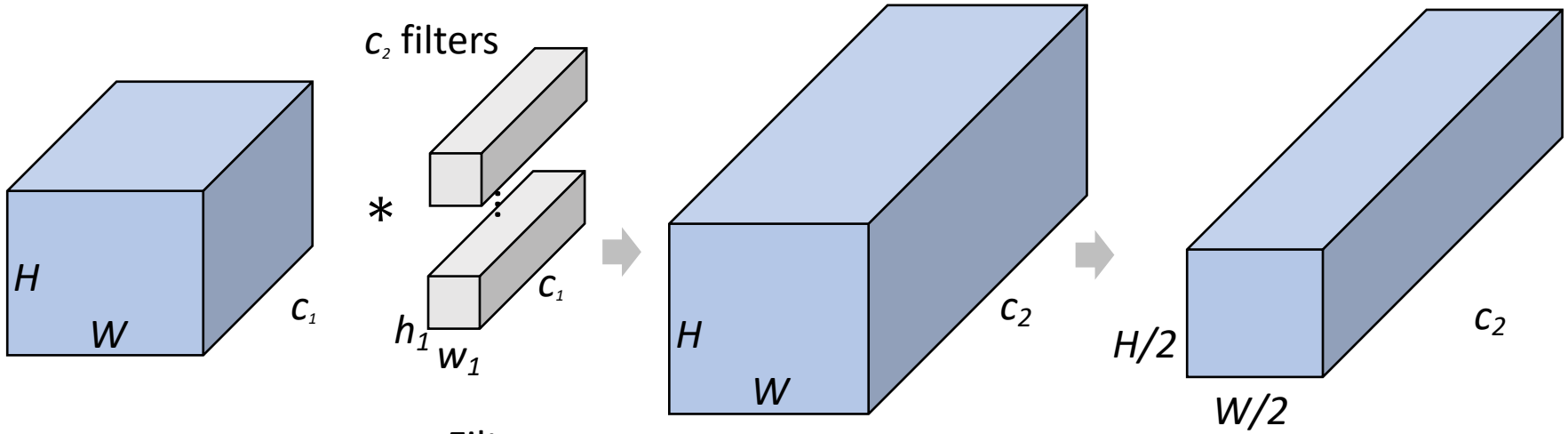
- CNNs were introduced by Yann LeCun in 1995 for the problem of handwritten digit classification
- Two successive layers of convolution, and subsampling, followed by fully-connected
- Notice the similarity to an image pyramid! Except here we are performing many operations at different scales

Convolutional Neural Networks



- Spatial Pooling (subsampling) gives us:
  - Invariance (to translation, rotation, some deformation)
  - Ability to learn filters at different scales

Convolutional Neural Networks



Input Image /Feature Map

Filters (weights)

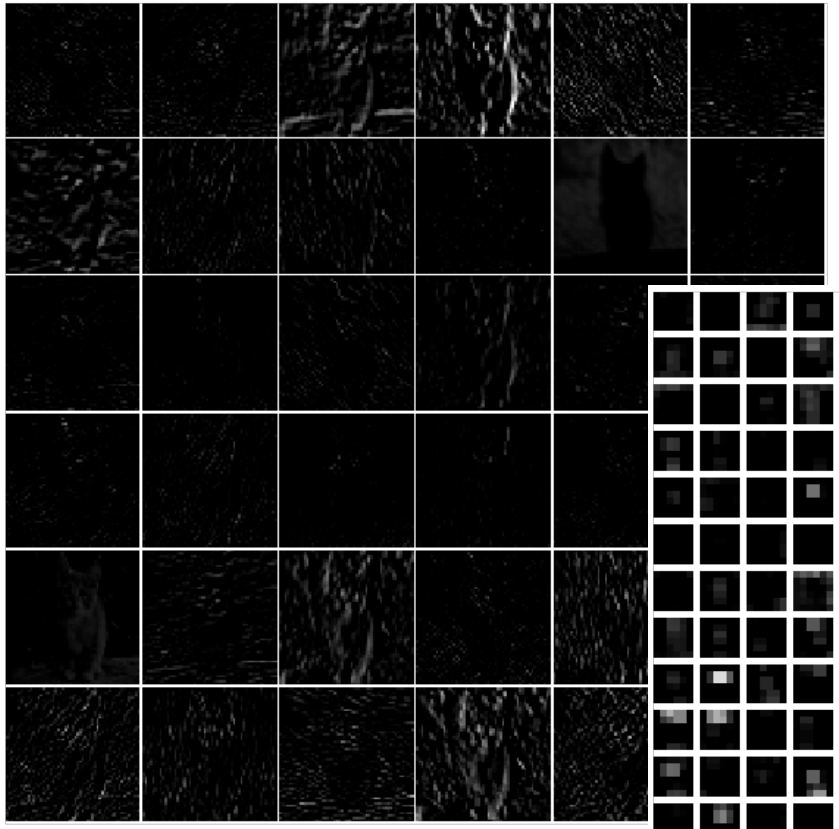
Feature map (output image)

Average/Max Pooling (subsampling)

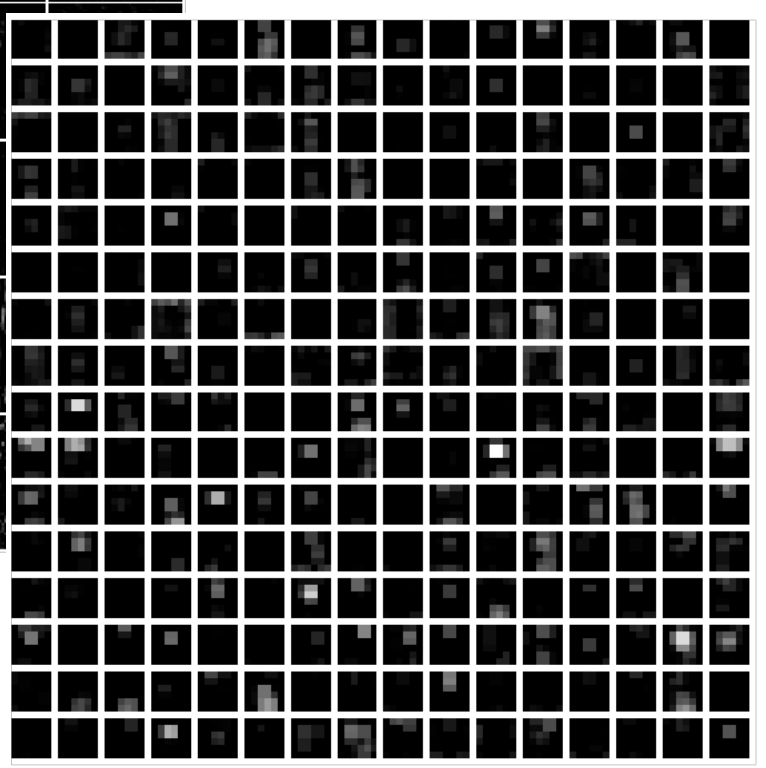
- The first filters act on images – easy to understand their functions and outputs
- As we go deeper, filters are on feature maps! What are these filters doing?

Convolutional Neural Networks

- Tricky question to answer!
- Visualizing featuremaps directly doesn't help much…



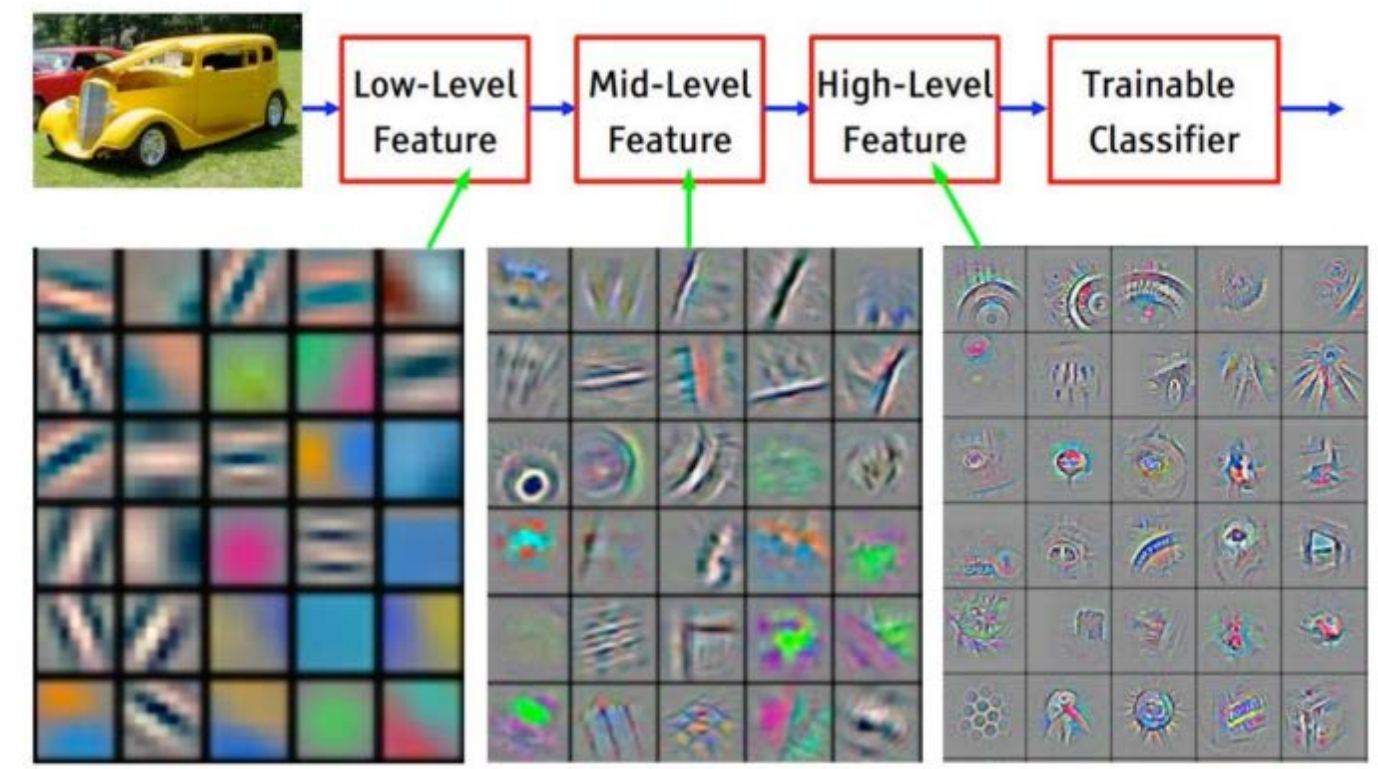
conv1



conv5 (after pooling)

Convolutional Neural Networks

- Lots of work on trying to understand the features being learned, often with *saliency maps*, i.e. understanding what was important in the original image
- Note the parts being highlighted!



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Convolutional Neural Networks v.s. DPMs

- Interpretability is difficult (and a major open problem)
- *However*, clear that CNNs learn high level concepts by learning parts

- DPM is a good intuition as to what's going on in a CNN – in fact it has been shown that a DPM is a CNN (not vice versa)
- Each filter is matching part responses from previous layers and assembling those responses spatially
- CNN incorporates idea of an image pyramid, and at least the first layer has band-pass filters

Why This Course Still Matters

- While we don't handcraft features anymore (i.e. SIFT), we do design network architectures and training methods
- Hundreds of papers at each conference on how to apply deep learning to new problems in computer vision – all based on incorporating what we know about learning from images
- CNNs themselves incorporate aspects of almost everything you've learned about
  - Image pyramids, gradients, band-pass filters, convolution, translation invariance, scale, low-dimensional embeddings (non-linear dimensionality reduction), wavelets (Gabor), convolution theorem (Gabor)
- And of course you still have to understand where the images themselves come from…

Convolutional Neural Networks: The Big Questions

- Understanding why they make decisions is difficult, but important (Interpretability)
- Can we learn these network architectures (i.e. connectivity) instead of hand-designing it (Neural Architecture Search)
- CNNs are typically massive (AlexNet is 60 million parameters), and are definitely overparameterized (Efficient Deep Learning)
- We don’t understand why deep networks *generalize* so well – all current theory says they shouldn’t, but empirical evidence disagrees (Lack of Theory)

Make the most of U of T!

- This is the school that started the deep learning revolution
- Has amongst the best faculty in the world in machine learning/computer vision
- Reach out and engage with faculty and ask about helping out on research projects!
  - Always better to do this in person – some get thousands of e-mails a day

- This is a good way to decide if you want to do a MSc/PhD or research
- Unfortunately I'm not research faculty here, but I'm happy to help you decide who to contact