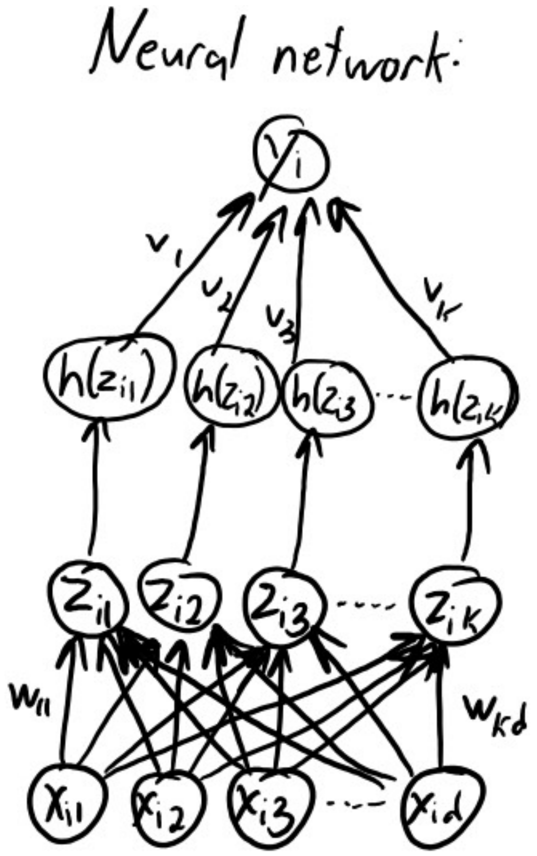
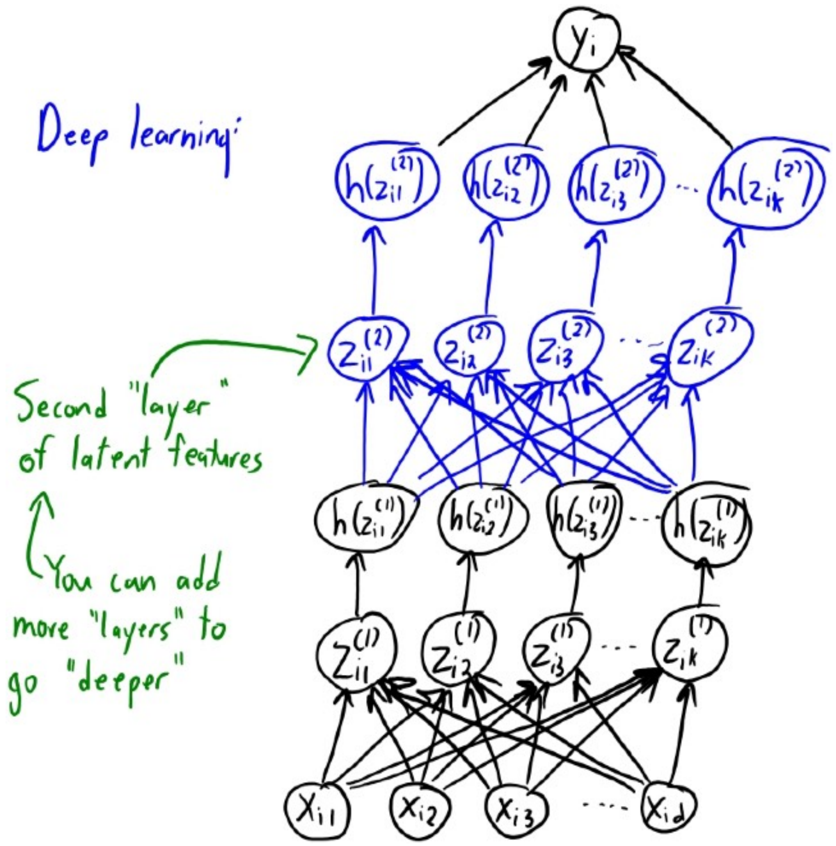
CPSC 340: Machine Learning and Data Mining

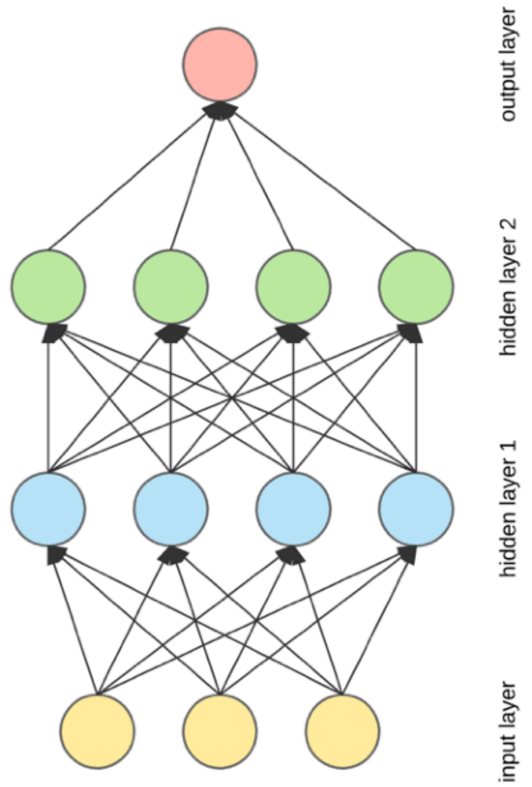
Convolutions

Spring 2022 (2021W2)

Deep Learning







Convolutional Neural Networks

– arguably the most important idea in computer vision

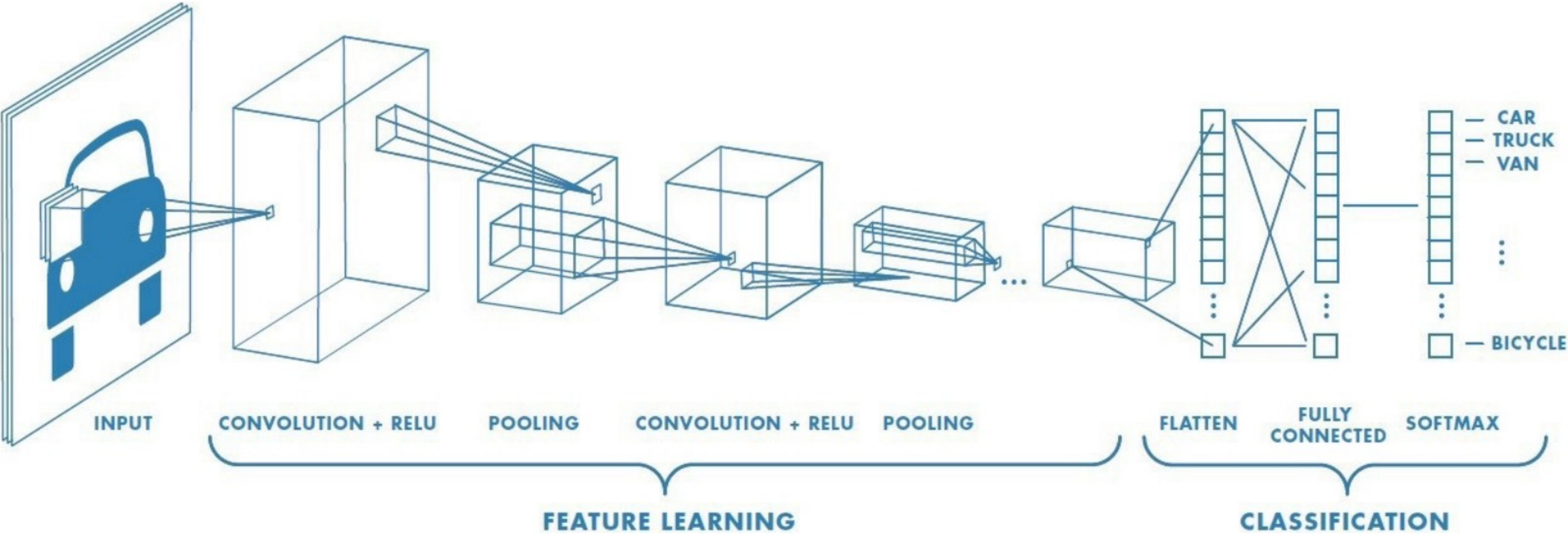
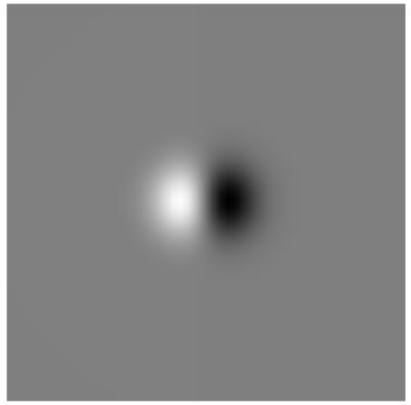

CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

Image Convolution Examples

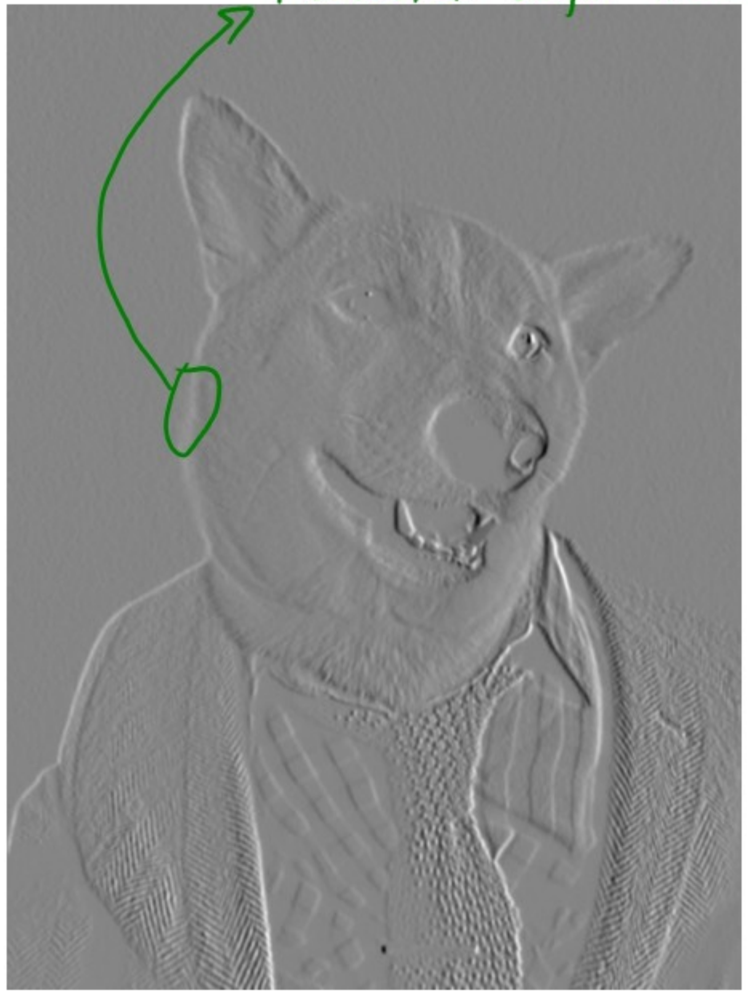
Gabor Filter
(Gaussian multiplied by sine or cosine)

$*$

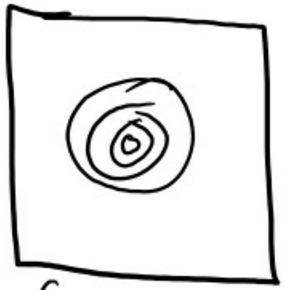


$=$

Horizontal "bright to dark"

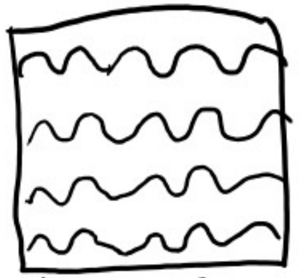


$||$



Gaussian

$.*$

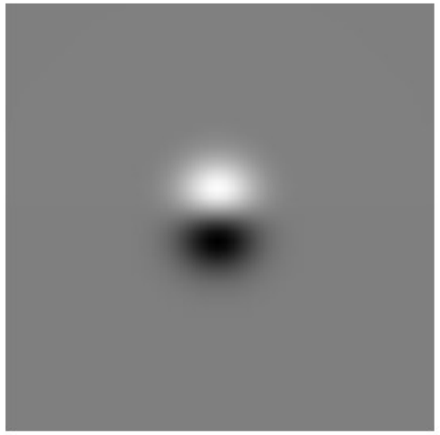


Parallel Sine functions

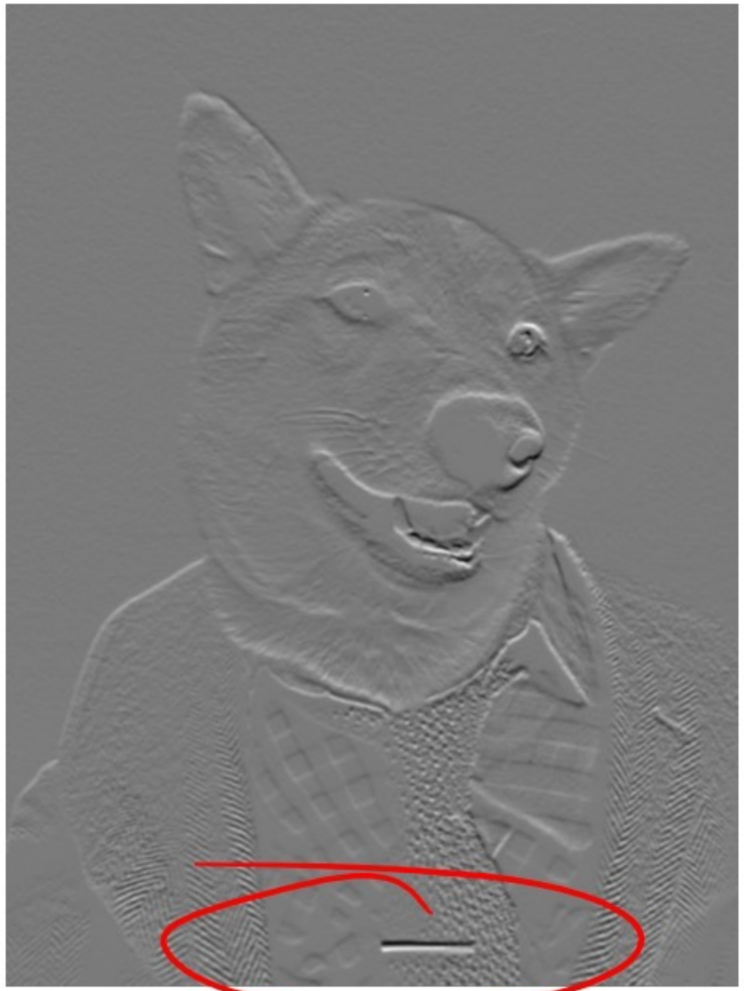
Image Convolution Examples

Gabor Filter
(Gaussian multiplied by sine or cosine)

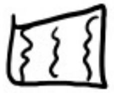
$*$



$=$



Different orientations of the sine/cosine let us detect changes with different orientations.

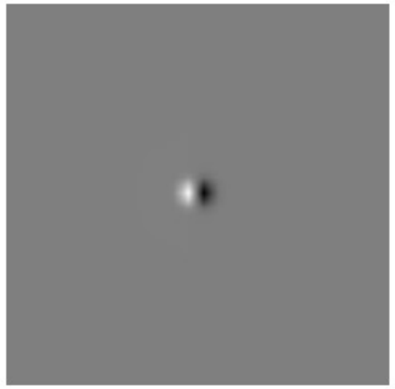


$\rightarrow$ 2d derivatives have a direction.

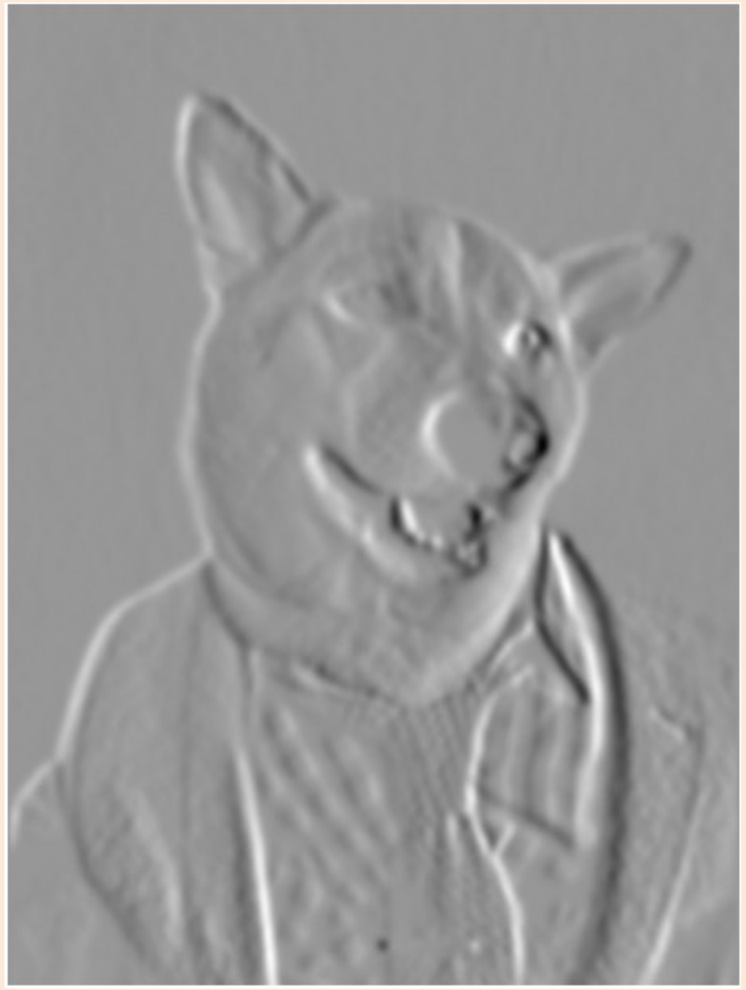
Image Convolution Examples

Gabor Filter
(Gaussian multiplied by sine or cosine)

$*$



$=$



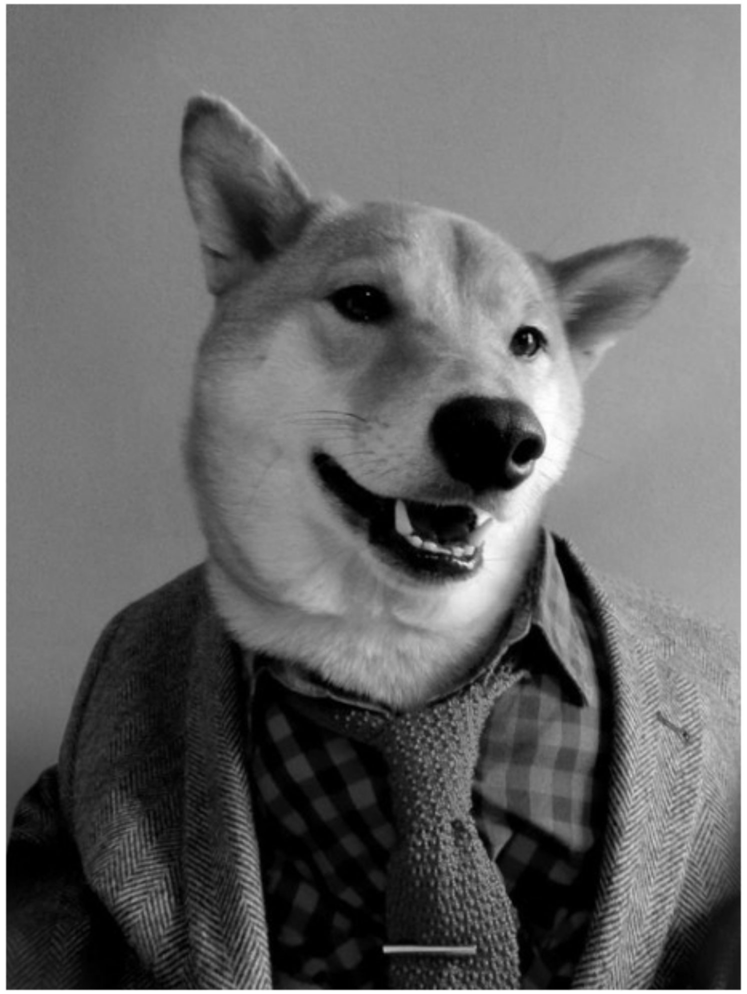
(smaller variance)

Vertical orientation

– Can obtain other orientations by rotating.

– May be similar to effect of V1 "simple cells."

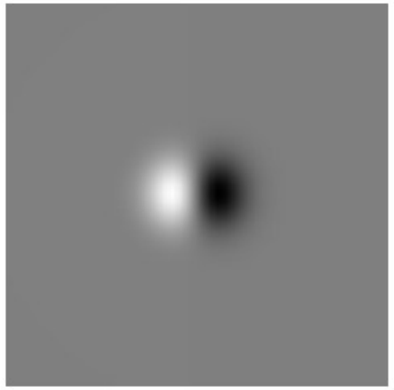
Image Convolution Examples



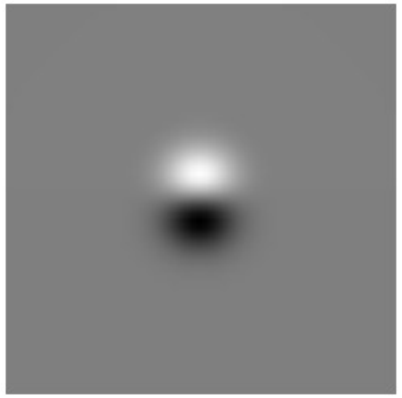
Max absolute value

between horizontal and vertical Gabor:

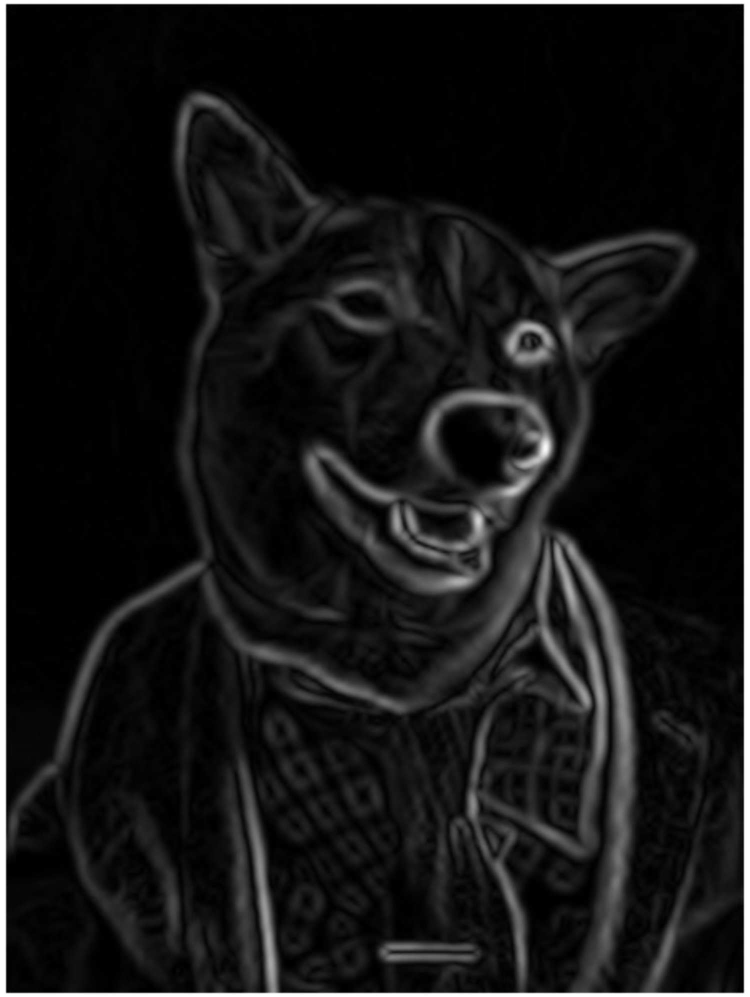
$*$



$*$



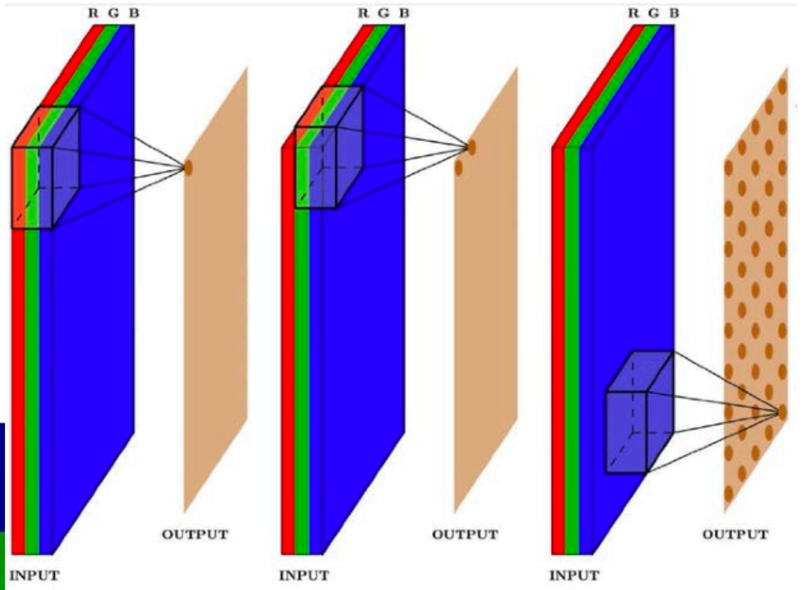
$\longrightarrow$ maximum absolute value $\nearrow$



"Horizontal/vertical edge detector"

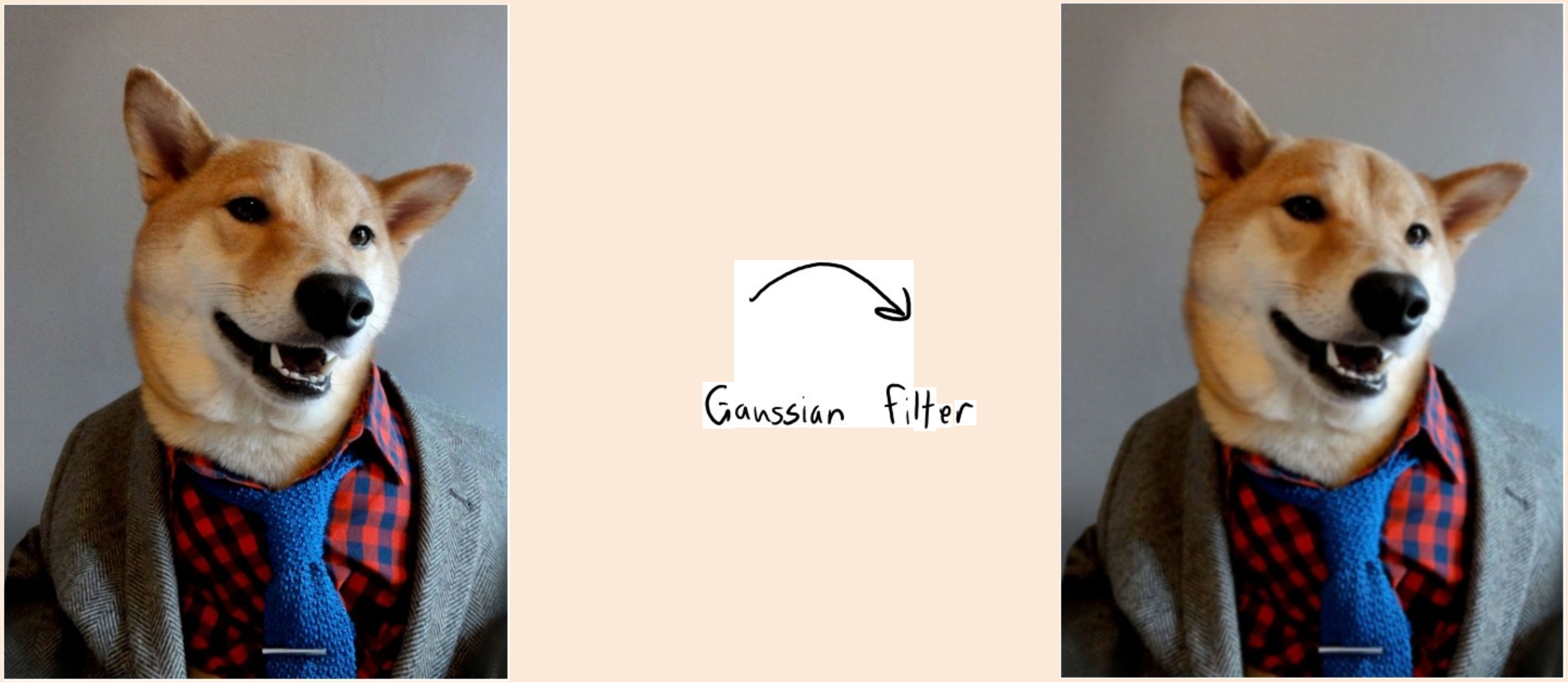
3D Convolution

Represent as RGB

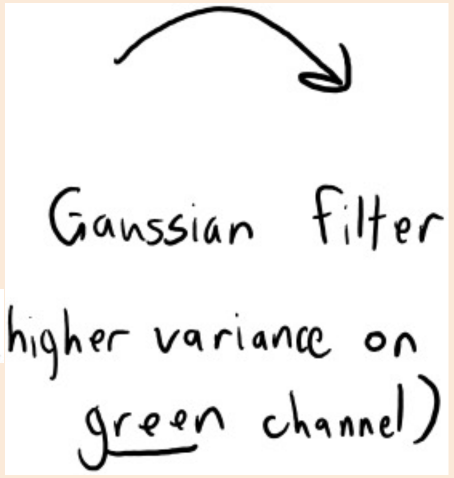


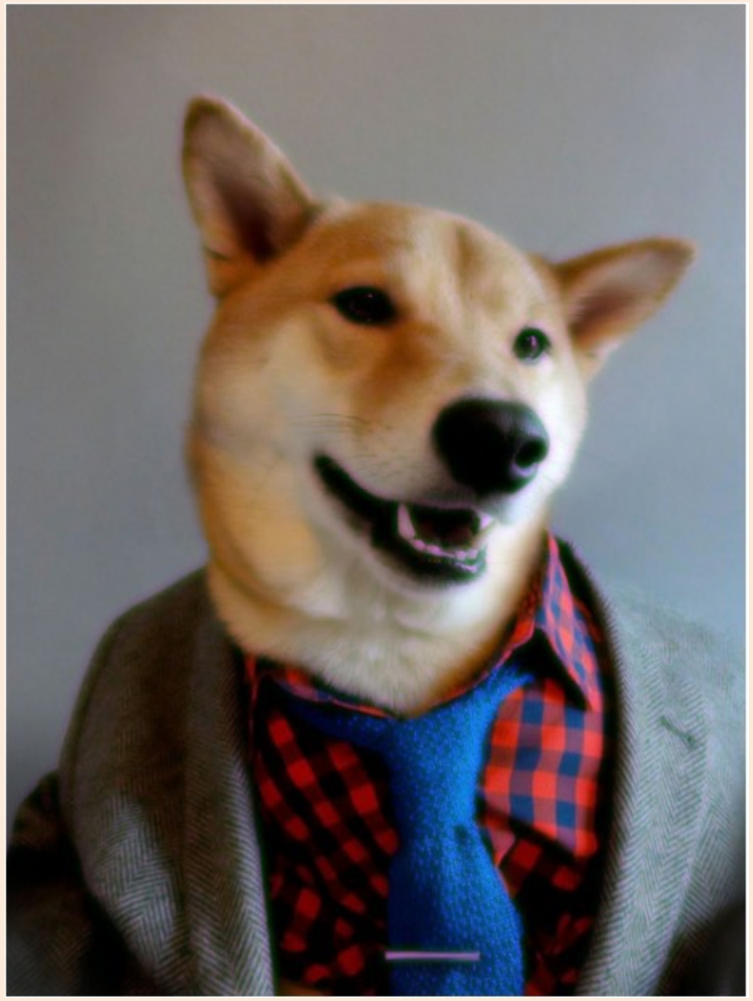
Can apply 3D convolutions

3D Convolution

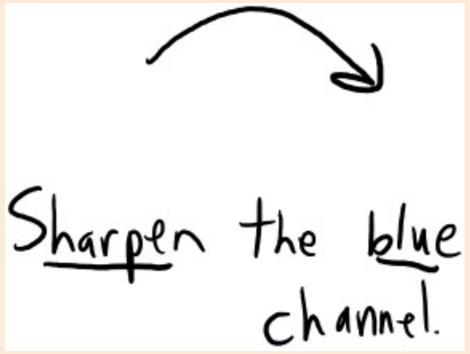


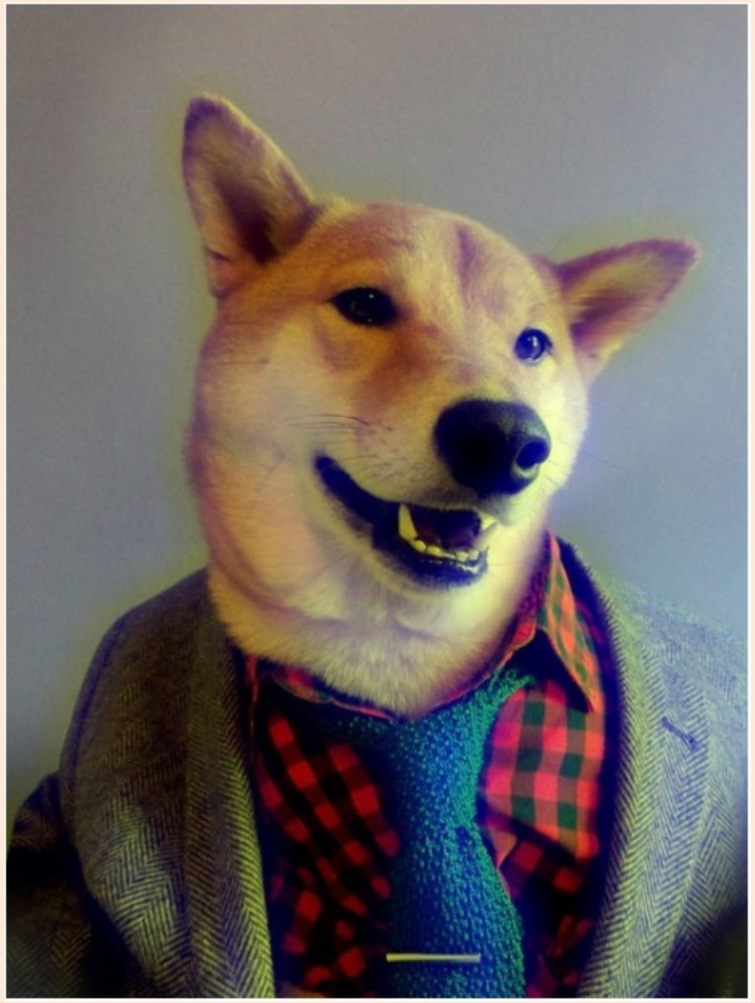
3D Convolution



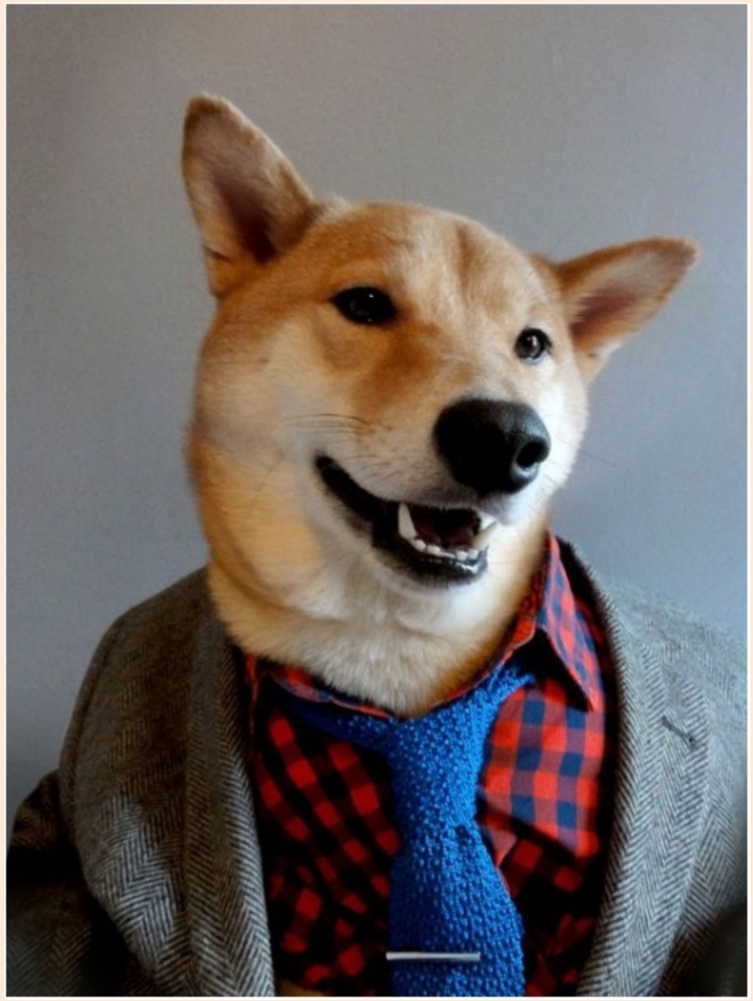


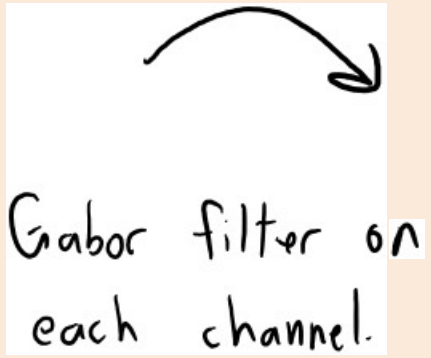
3D Convolution

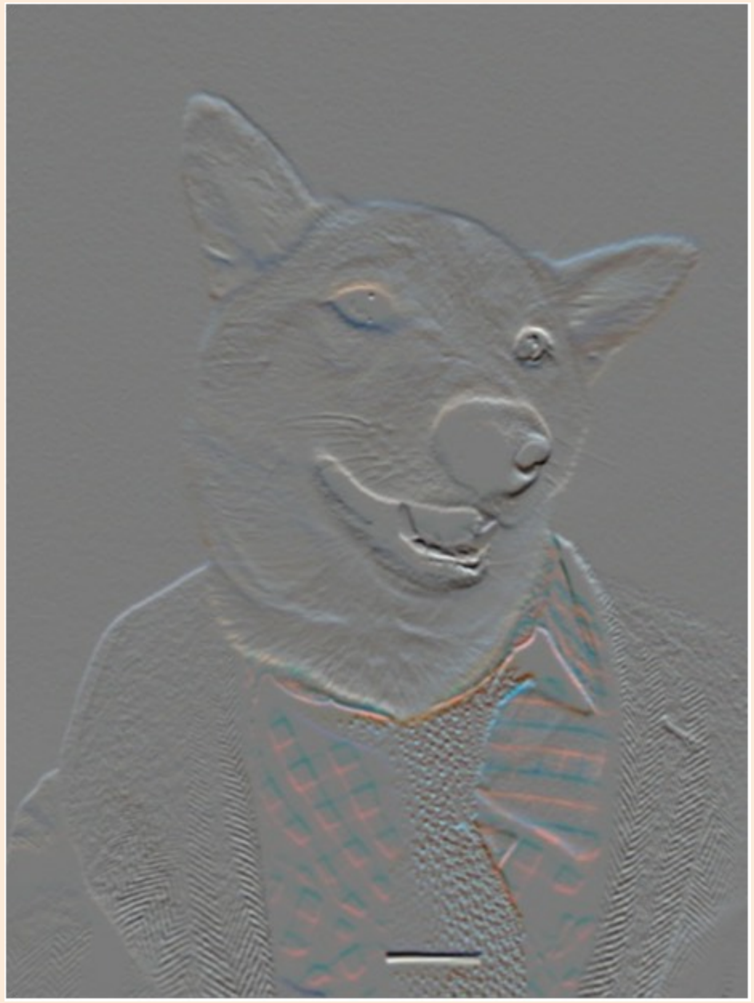




3D Convolution



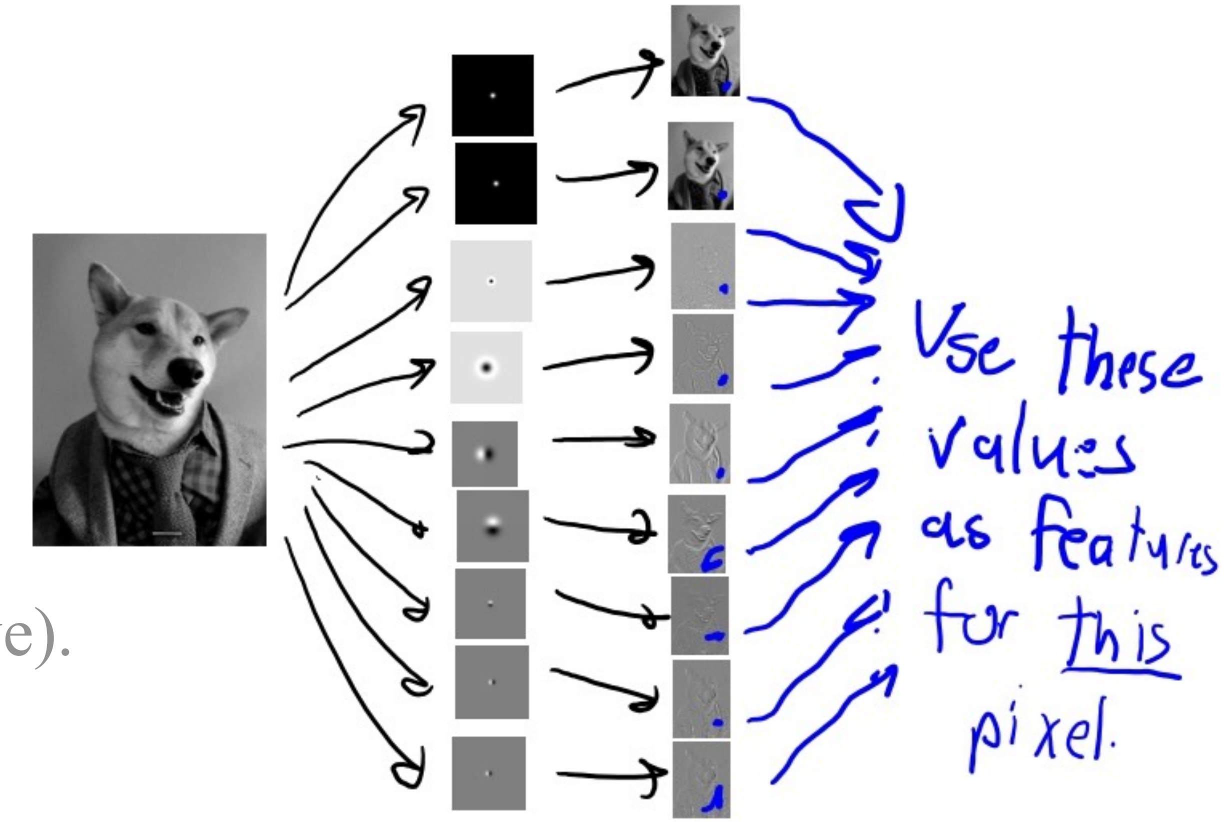




Convolutions as Features

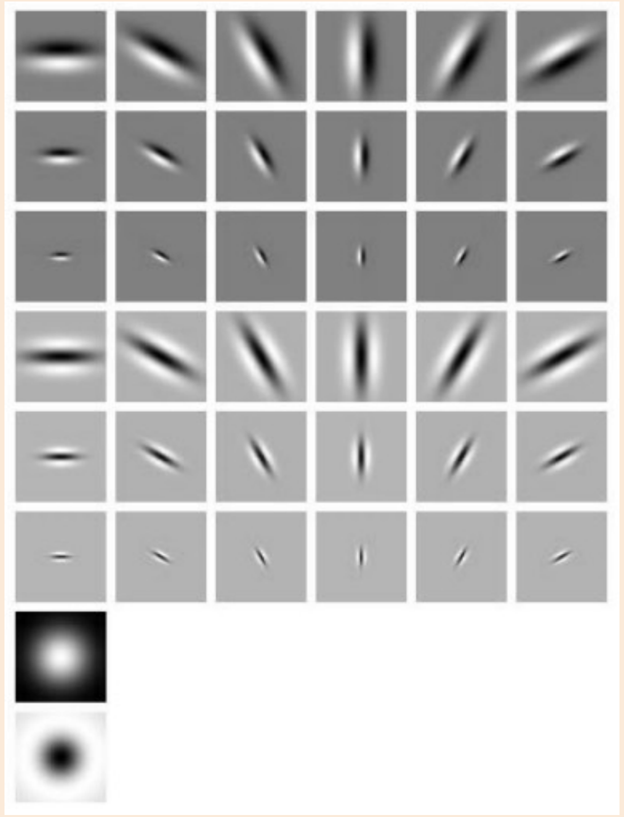
- Classic vision methods use convolutions as features :
  - Usually have different types/variances/orientations.
  - Can take maxes across locations/orientations/scales.

- Notable convolutions:
  - Gaussian (blurring/averaging).
  - Laplace of Gaussian (second- derivative).
  - Gabor filters (directional first- or higher- derivative).



Filter Banks

- To characterize context, we used to use filter banks like “MR8”:
  - 1 Gaussian filter, 1 Laplacian of Gaussian filter.
  - 6 max(abs(Gabor)) filters:
    - 3 scales of sine/cosine (maxed over 6 orientations).



- Convolutional neural networks have replaced filter banks.

http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

bonus!

Global and Local Features for Domain Adaptation

- Suppose you want to solve a classification task, where you have very little labeled data from your domain.
- But you have access to a huge dataset with the same labels, from a different domain.
- Example:
  - You want to label POS tags in medical articles, and pay a few $$$ to label some.
  - You have access the thousands of examples of Wall Street Journal POS labels.
- Domain adaptation: using data from different domain to help.

bonus!

Global and Local Features for Domain Adaptation

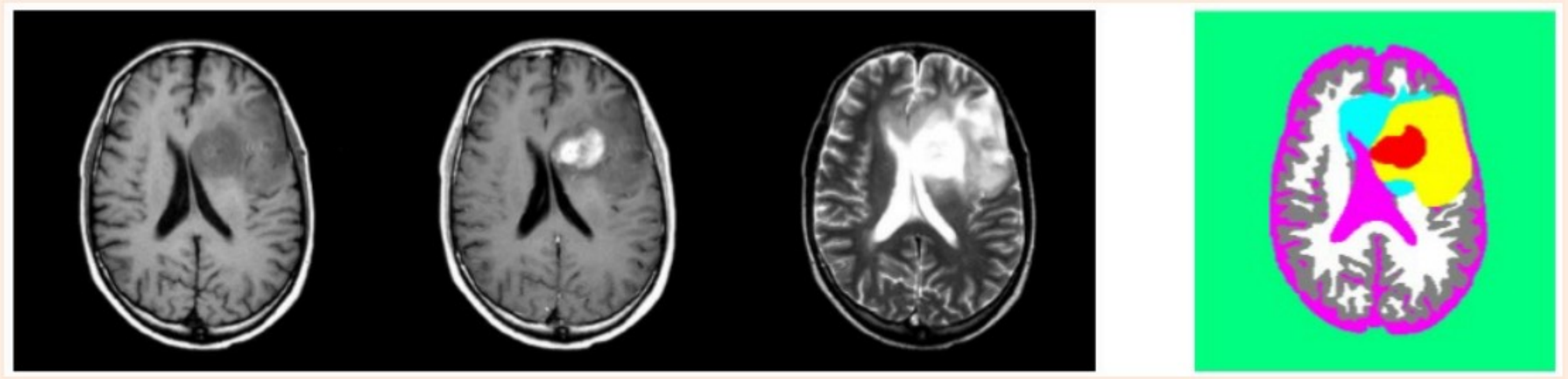
- "Frustratingly easy domain adaptation":
  - Use "global" features across the domains, and "local" features for each domain.
  - "Global" features let you learn patterns that occur across domains.
    - Leads to sensible predictions for new domains without any data.
  - "Local" features let you learn patterns specific to each domain.
    - Improves accuracy on particular domains where you have more data.
  - For linear classifiers this would look like:

$$\hat{y}_i = \text{sign}(\underbrace{w_g^T x_{ig}}_{\text{features used across domains}} + \underbrace{w_d^T x_{id}}_{\text{features/weights specific to domain}})$$

bonus!

Image Coordinates

- Should we use the image coordinates?
  - E.g., the pixel is at location (124, 78) in the image.

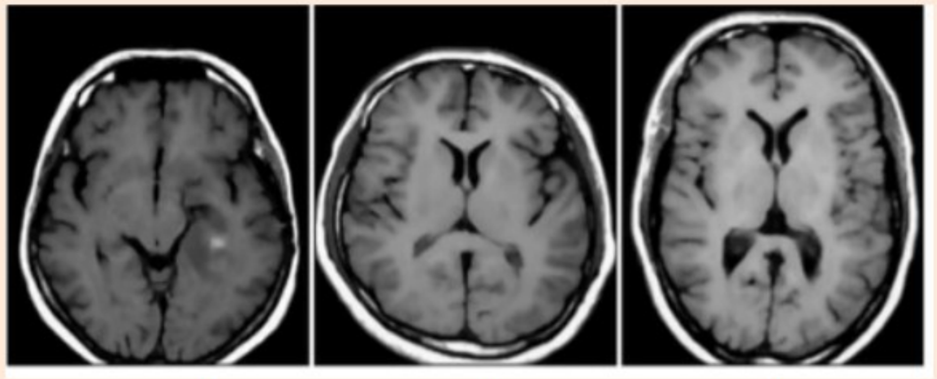


- Considerations:
  - Is the interpretation different in different areas of the image?
  - Are you using a linear model?
    - Would "distance to center" be more logical?
  - Do you have enough data to learn about all areas of the image?

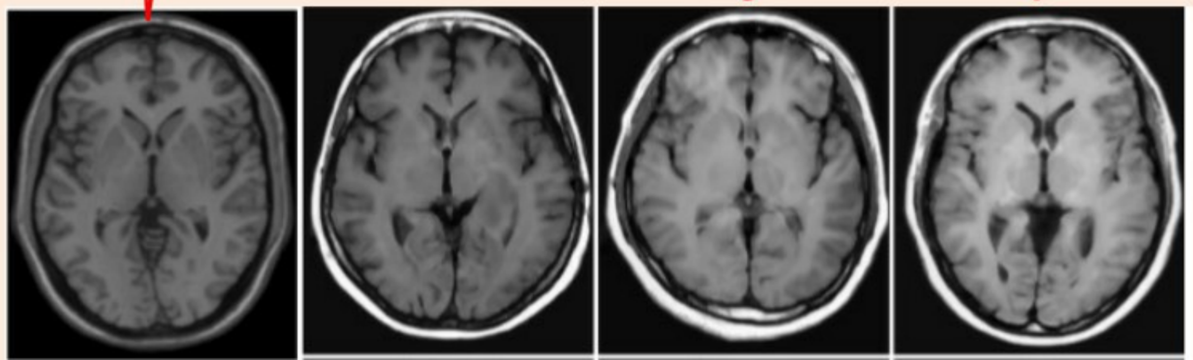
bonus!

Alignment- Based Features

- The position in the image is important in brain tumour application.
  - But we didn't have much data, so coordinates didn't make sense.
- We aligned the images with a "template image".



Template

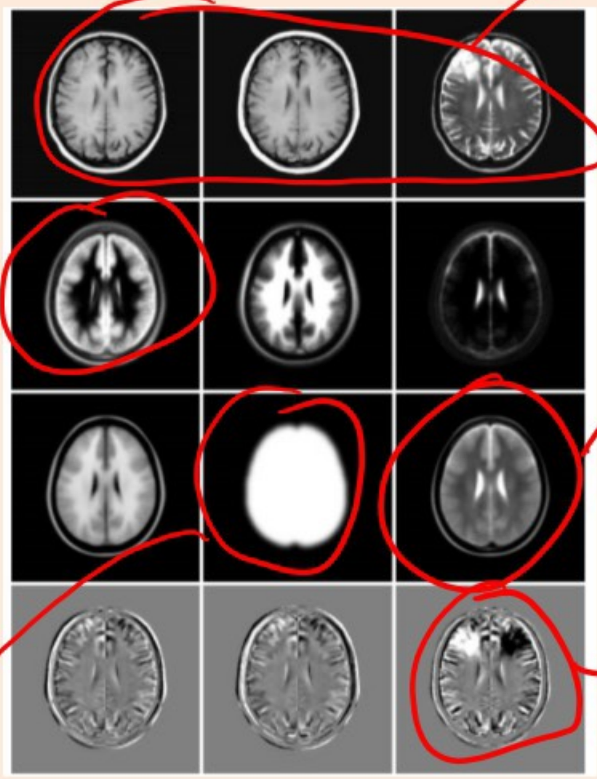


(Look different because we're showing middle slice and alignment is in 3D.)

bonus!

Alignment- Based Features

- The position in the image is important in brain tumour application.
  - But we didn't have much data, so coordinates didn't make sense.
- We aligned the images with a "template image".
  - Allowed "alignment- based" features:



Original pixel values

Probability of gray matter at this pixel among tons of people aligned with template.

Actual pixel value of template image at this location.

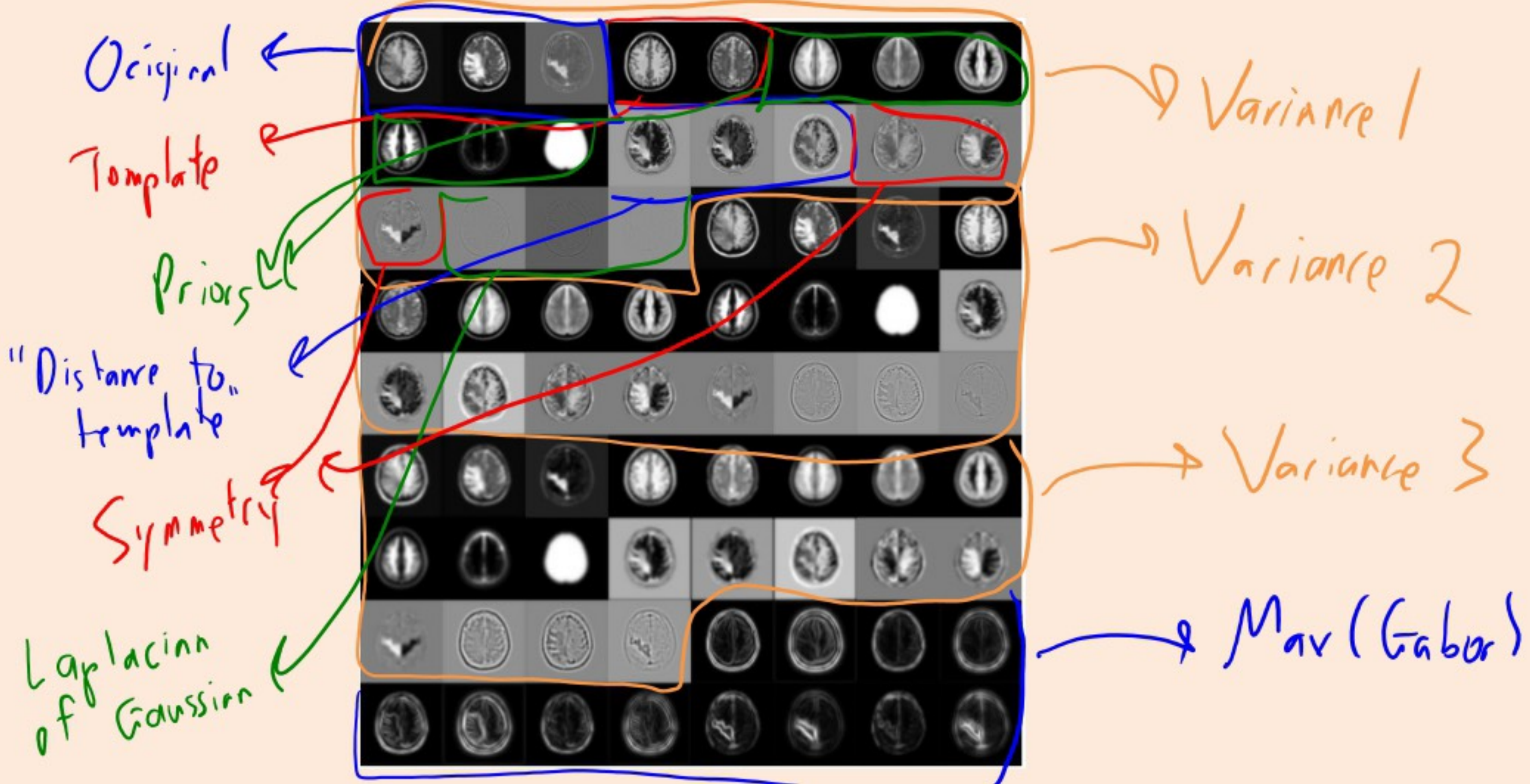
Probability of being brain pixel.

Left-right symmetry difference.

bonus!

Motivation: Automatic Brain Tumor Segmentation

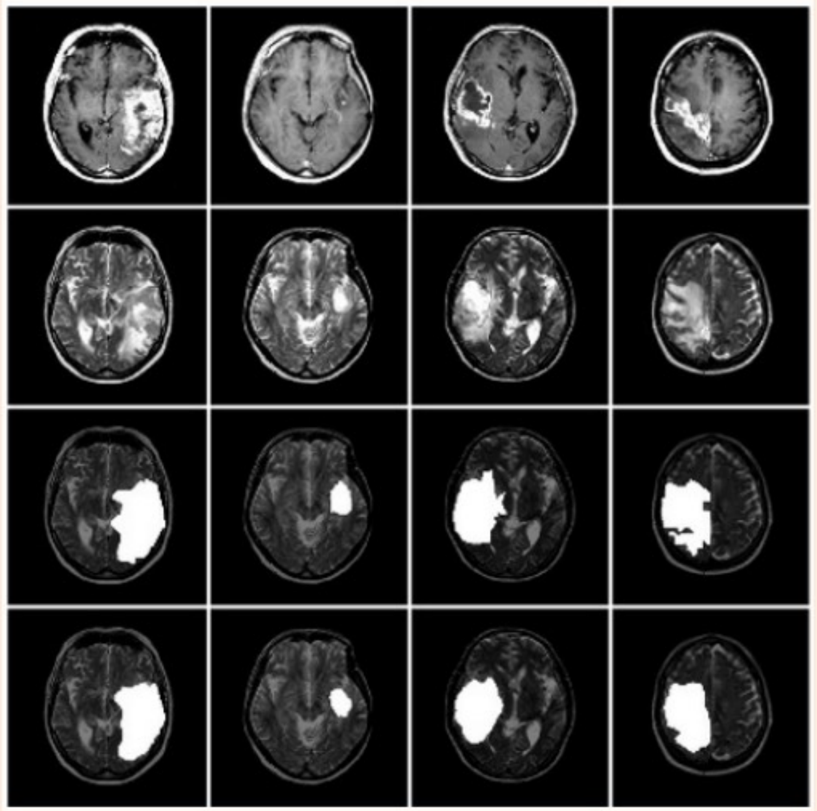
- Final features for brain tumour segmentation:
  - Gaussian convolution of original/template/priors/symmetry, Laplacian of Gaussian on original.
    - All with 3 variances.
    - Max(Gabor) with sine and cosine on orginal (3 variances).



bonus!

Motivation: Automatic Brain Tumour Segmentation

- Logistic regression and SVMs among best methods.
  - When using these 72 features from last slide.
  - If you used all features I came up with, it overfit .
- Possible solutions to overfitting:
  - Forward selection was too slow.
    - Just one image gives 8 million training examples.
  - I did manual feature selection ("guess and check").
  - L2- regularization with all features also worked.
    - But this is slow at test time .
    - L1- regularization gives best of regularization and feature selection.



bonus!

FFT implementation of convolution

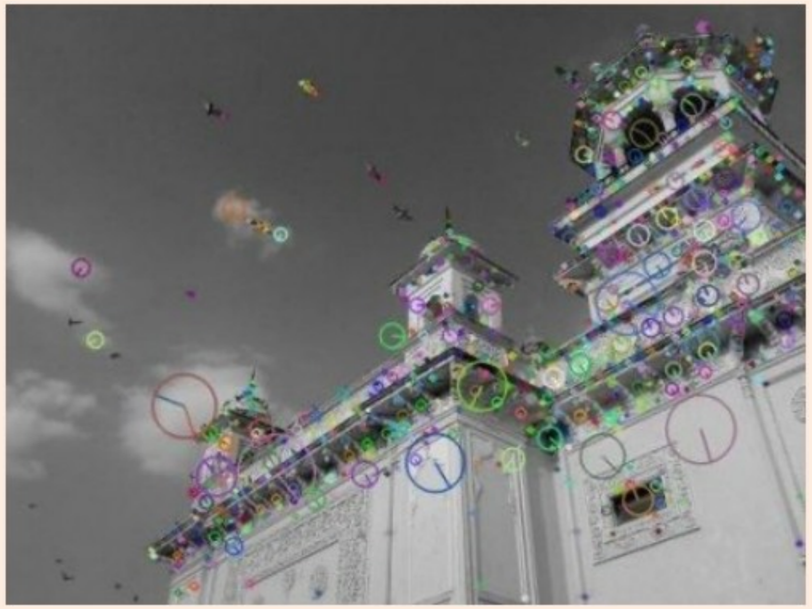
- Convolutions can be implemented using fast Fourier transform:
  - Take FFT of image and filter, multiply elementwise, and take inverse FFT.

- It has faster asymptotic running time but there are some catches:
  - You need to be using periodic boundary conditions for the convolution.
  - Constants matter: it may not be faster in practice.
    - Especially compared to using GPUs to do the convolution in hardware.
  - The gains are largest for larger filters (compared to the image size).

bonus!

SIFT Features

- Scale-invariant feature transform (SIFT):
  - Features used for object detection ("is particular object in the image"?)
  - Designed to detect unique visual features of objects at multiple scales.
  - Proven useful for a variety of object detection tasks.

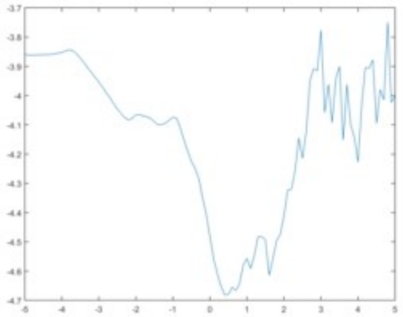


http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html

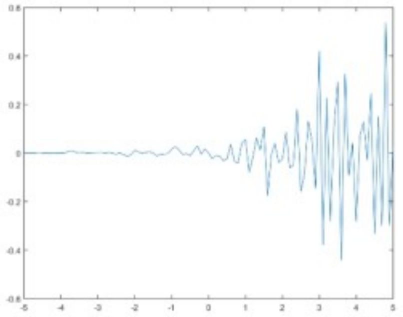
1D Convolution as Matrix Multiplication

- 1D convolution:
  - Takes signal ‘x’ and filter ‘w’ to produces vector ‘z’:

$$x \ast w = z$$



$$[1 \ -2 \ \ 1]$$



  - Can be written as a matrix multiplication:

$$W_x = \begin{bmatrix} 1 & -2 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ & & & \vdots & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 & -2 & 1 \end{bmatrix} x = z$$

1D Convolution as Matrix Multiplication

- Each element of a convolution is an inner product:

$$z_i = \sum_{j=-m}^{m} w_j x_{i+j}$$

$$= w^T x_{(i-m:i+m)}$$

$$= \tilde{w}^T x \quad \text{where } \tilde{w} = [0 \;\; 0 \;\; 0 \; \overbrace{\text{———} \; w \; \text{———}}^{\text{positions } i-m \text{ through } i+m} \; 0 \;\; 0]$$

- So convolution is a matrix multiplication (I'm ignoring boundaries):

$$z = \tilde{W}x \quad \text{where } \tilde{W} = \begin{bmatrix} \text{———} & w & \text{———} & 0 & 0 & 0 \\ 0 & \text{———} & w & \text{———} & 0 & 0 \\ 0 & 0 & \text{———} & w & \text{———} & 0 \\ 0 & 0 & 0 & \text{———} & w & \text{———} \end{bmatrix} \Bigg\} \text{matrix can be very sparse and only has } 2m+1 \text{ variables.}$$

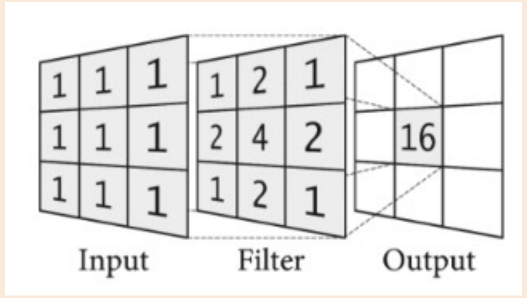
- The shorter 'w' is, the more sparse the matrix is.

2D Convolution as Matrix Multiplication

- 2D convolution:
  - Signal ‘x’, filter ‘w’, and output ‘z’ are now all images/matrices:

x

$$* \quad w = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \quad = \quad z$$



  - Vectorized ‘z’ can be written as a matrix multiplication with vectorized ‘x’:

$$W = \left[\begin{array}{ccc|cccc|c|ccc|cc} -2 & -1 & 0\;0\;0\;0 \cdots 0 & -1 & 0 & 1 & 0 & 0\;0 \cdots 0 & 0 & 1 & 2 & 0\;0\;0 \cdots 0 & 0\;0 \cdots \\ 0 & -2 & -1\;0\;0\;0 \cdots 0 & 0 & -1 & 0 & 1 & 0\;0 \cdots 0 & 0 & 0 & 1 & 2\;0\;0 \cdots 0 & 0\;0 \cdots \\ & & \vdots \cdots 2\;{-1}\;0 & 0 & 0 & 0 & & \cdots -1\;0\;1 & 0 & 0 & 0 & \cdots 0\;1\;2 & 0\;0 \cdots \\ 0 & 0 & 0 \cdots & & & & & & & & & & \\ 0 & 0 & 0\;0\;0\;0 \cdots 0 & 2 & -1 & 0 & 0 & 0\;0 \cdots 0 & -1 & 0 & 1 & 0\;0\;0 \cdots 0 & 0\;1\;2 \\ 0 & 0 & 0\;0\;0\;0 \cdots 0 & 0 & 2 & -1 & 0 & 0\;0 \cdots 0 & 0 & -1 & 0 & 1\;0\;0 \cdots 0 & 0\;0\;1 \cdots \\ & & \vdots & & & & & \vdots & & & & \vdots & \end{array}\right]$$

Motivation for Convolutional Neural Networks

- Consider training neural networks on 256 by 256 images.
  - This is 256 by 256 by 3 ≈ 200,000 inputs.
- If first layer has k=10,000, then it has about 2 billion parameters .
  - We want to avoid this huge number (due to storage and overfitting).

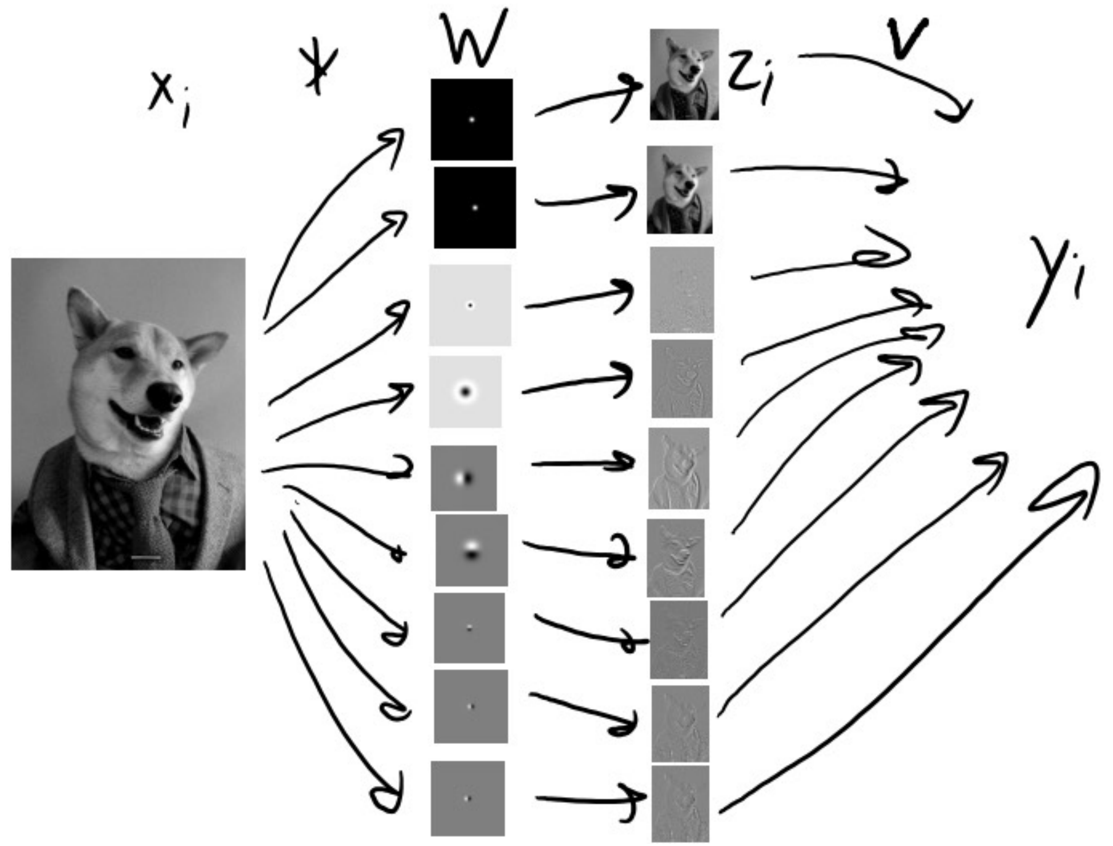
- Key idea: make Wx_i act like several convolutions (to make it sparse):
  1. Each row of W only applies to part of x_i .
  2. Use the same parameters between rows.

$$w_1 = [0 \quad 0 \quad 0 \;\text{——}\; w \;\text{——}\; 0 \quad 0 \quad 0]$$
$$w_2 = [0 \;\text{——}\; w \;\text{——}\; 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

- Forces most weights to be zero, reduces number of parameters.

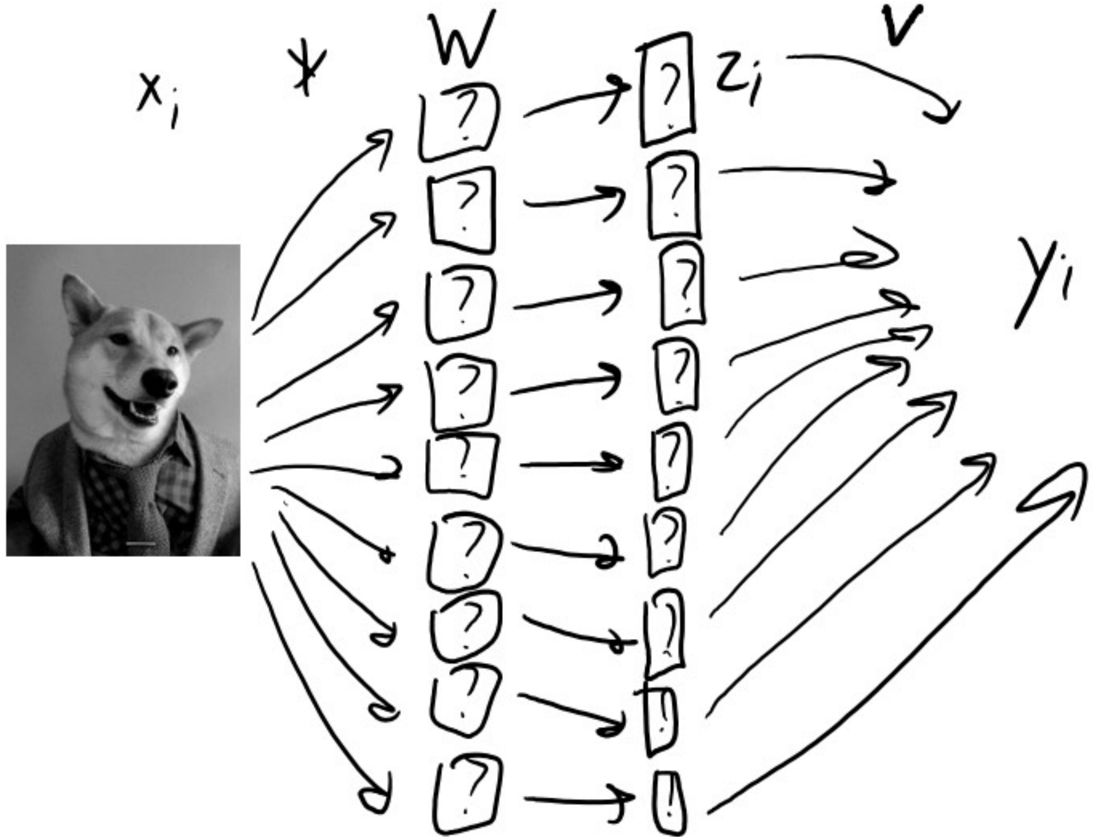
Motivation for Convolutional Neural Networks

- Classic vision methods uses fixed convolutions as features:
  - Usually have different types/variances/orientations.
  - Can do subsampling or take maxes across locations/orientations/scales.



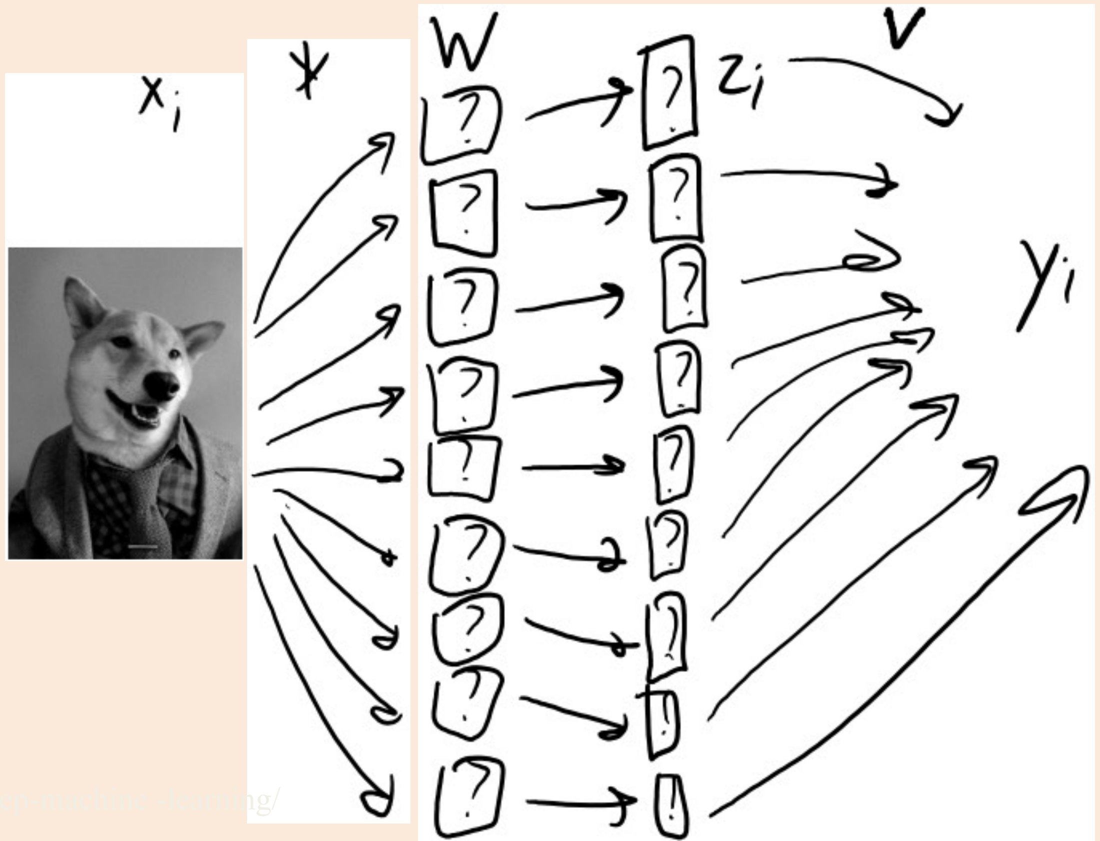
Motivation for Convolutional Neural Networks

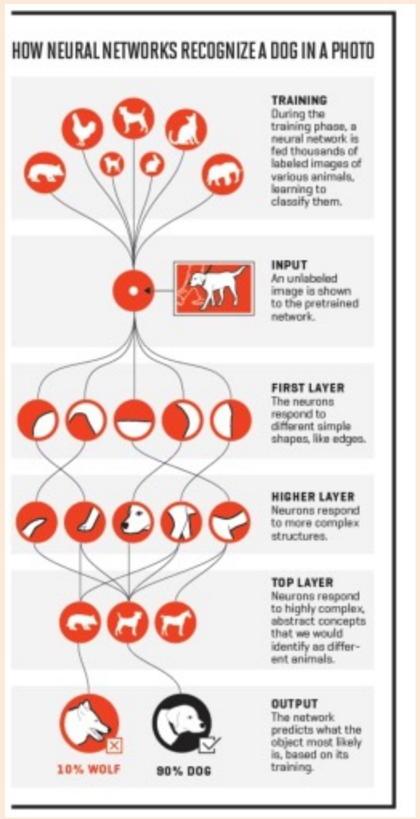
- Convolutional neural networks learn the convolutions:
  - Learning ‘W’ and ‘v’ automatically chooses types/variances/orientations.
  - Don’t pick from fixed convolutions, but learn the elements of the filters.



Motivation for Convolutional Neural Networks

- Convolutional neural networks learn the convolutions:
  - Learning ‘W’ and ‘v’ automatically chooses types/variances/orientations.
  - Can do multiple layers of convolution to get deep hierarchical features.




HOW NEURAL NETWORKS RECOGNIZE A DOG IN A PHOTO

TRAINING During the training phase, a neural network is fed thousands of labeled images of various animals, learning to classify them.

INPUT An unlabeled image is shown to the pretrained network.

FIRST LAYER The neurons respond to different simple shapes, like edges.

HIGHER LAYER Neurons respond to more complex structures.

TOP LAYER Neurons respond to highly complex, abstract concepts that we would identify as different animals.

OUTPUT The network predicts what the object most likely is, based on its training.

http://fortune.com/ai-artificial-intelligence-deep-machine-learning/

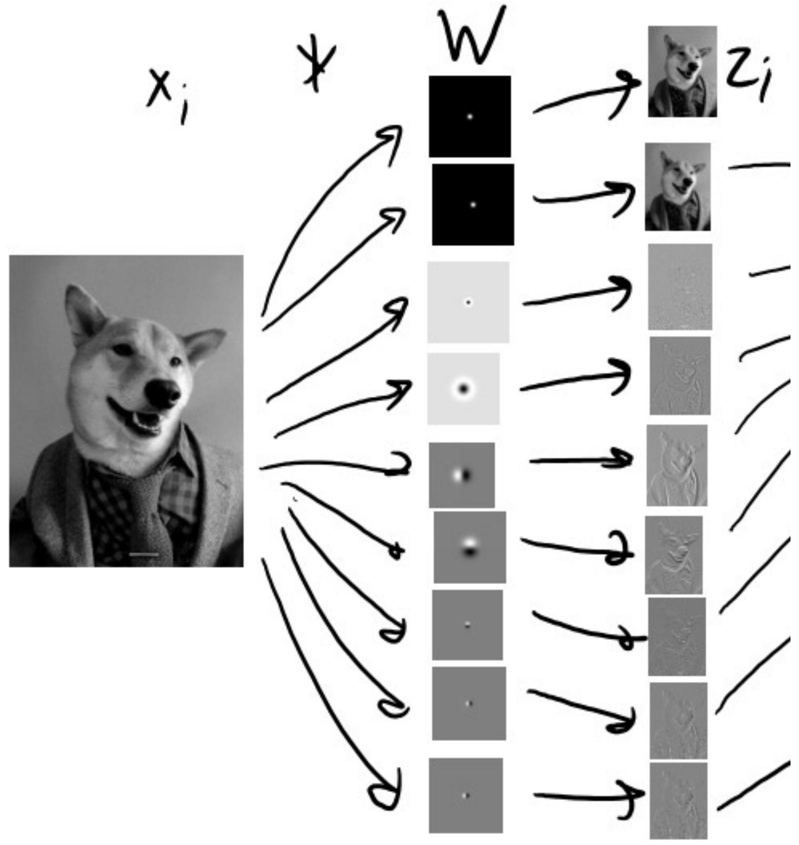
Convolutional Neural Networks

- Convolutional Neural Networks classically have 3 layer "types":
  - Fully connected layer : usual neural network layer with unrestricted W.

$$W^{(m)} = \begin{bmatrix} \text{———} \; w_1^{(m)} \; \text{———} \\ \text{———} \; w_2^{(m)} \; \text{———} \\ \vdots \\ \text{———} \; w_k^{(m)} \; \text{———} \end{bmatrix}$$

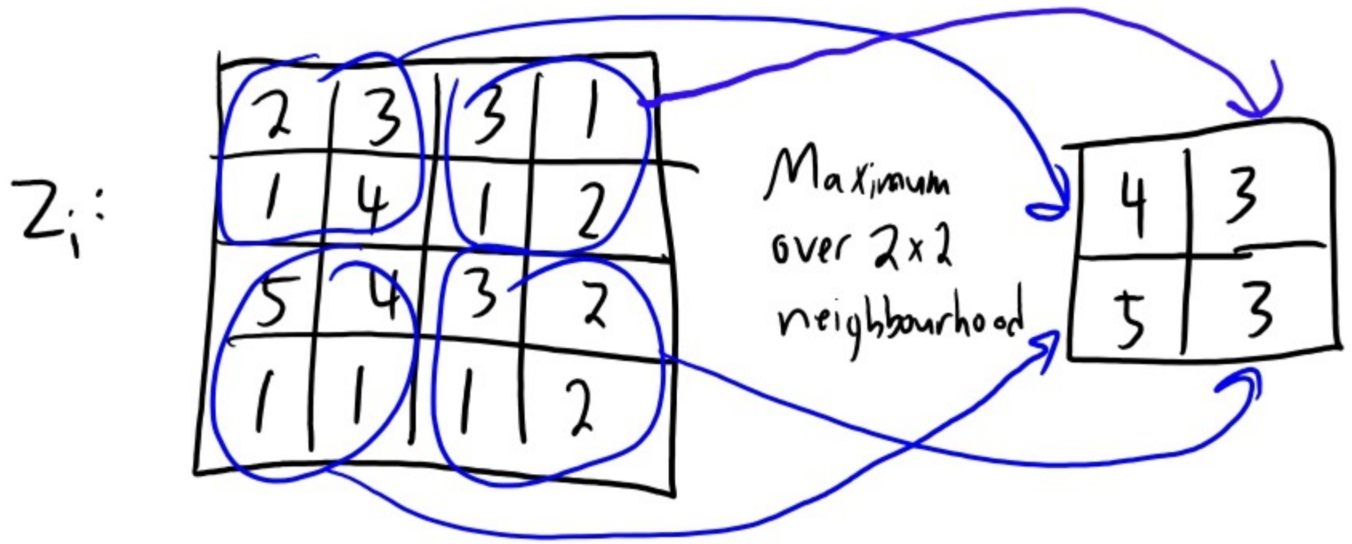
Convolutional Neural Networks

- Convolutional Neural Networks classically have 3 layer "types":
  - Fully connected layer : usual neural network layer with unrestricted W.
  - Convolutional layer : restrict W to act like several convolutions.



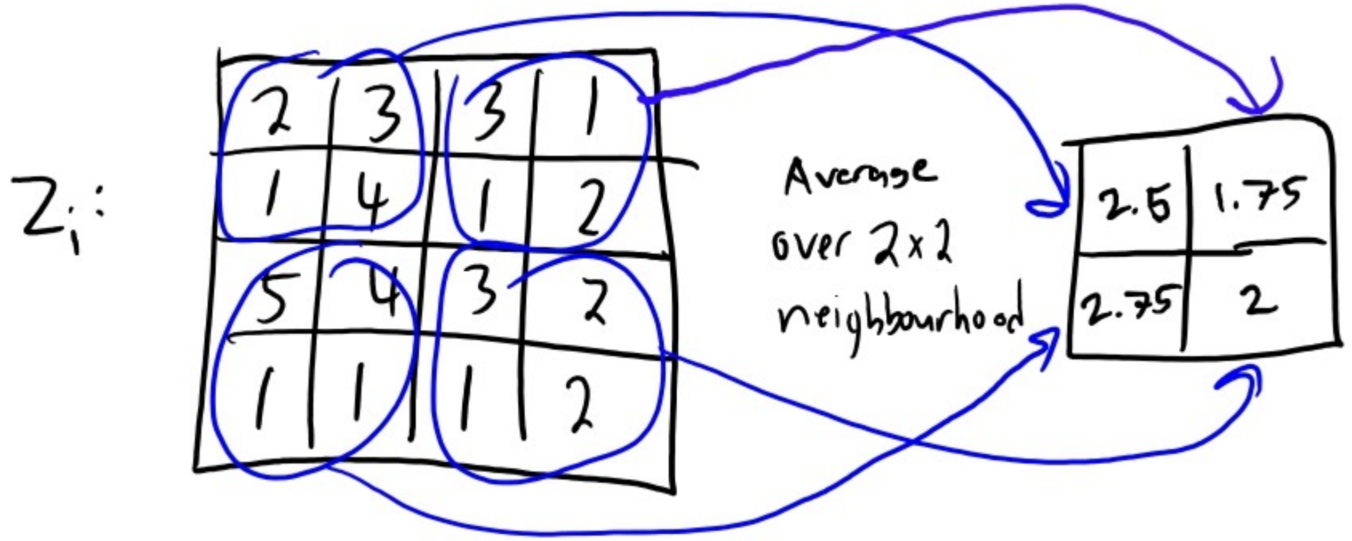
Convolutional Neural Networks

- Convolutional Neural Networks classically have 3 layer "types":
  - Fully connected layer : usual neural network layer with unrestricted W.
  - Convolutional layer : restrict W to act like several convolutions.
  - Pooling layer : combine results of convolutions.
    - Can add some invariance or just make the number of parameters smaller.
    - Often 'max pooling ':



Convolutional Neural Networks

- Convolutional Neural Networks classically have 3 layer "types":
  - Fully connected layer : usual neural network layer with unrestricted W.
  - Convolutional layer : restrict W to act like several convolutions.
  - Pooling layer : combine results of convolutions.
    - Can add some invariance or just make the number of parameters smaller.
    - Often 'max pooling' or else 'average pooling ':



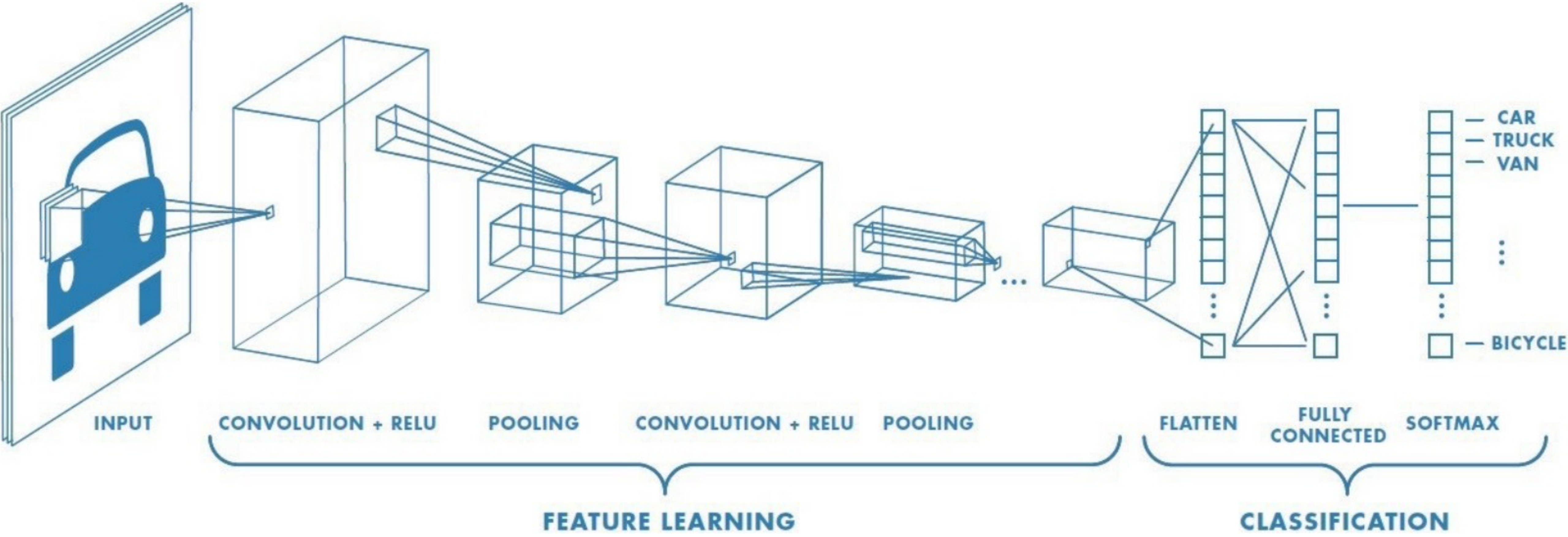
bonus!

Max Pooling vs Average Pooling

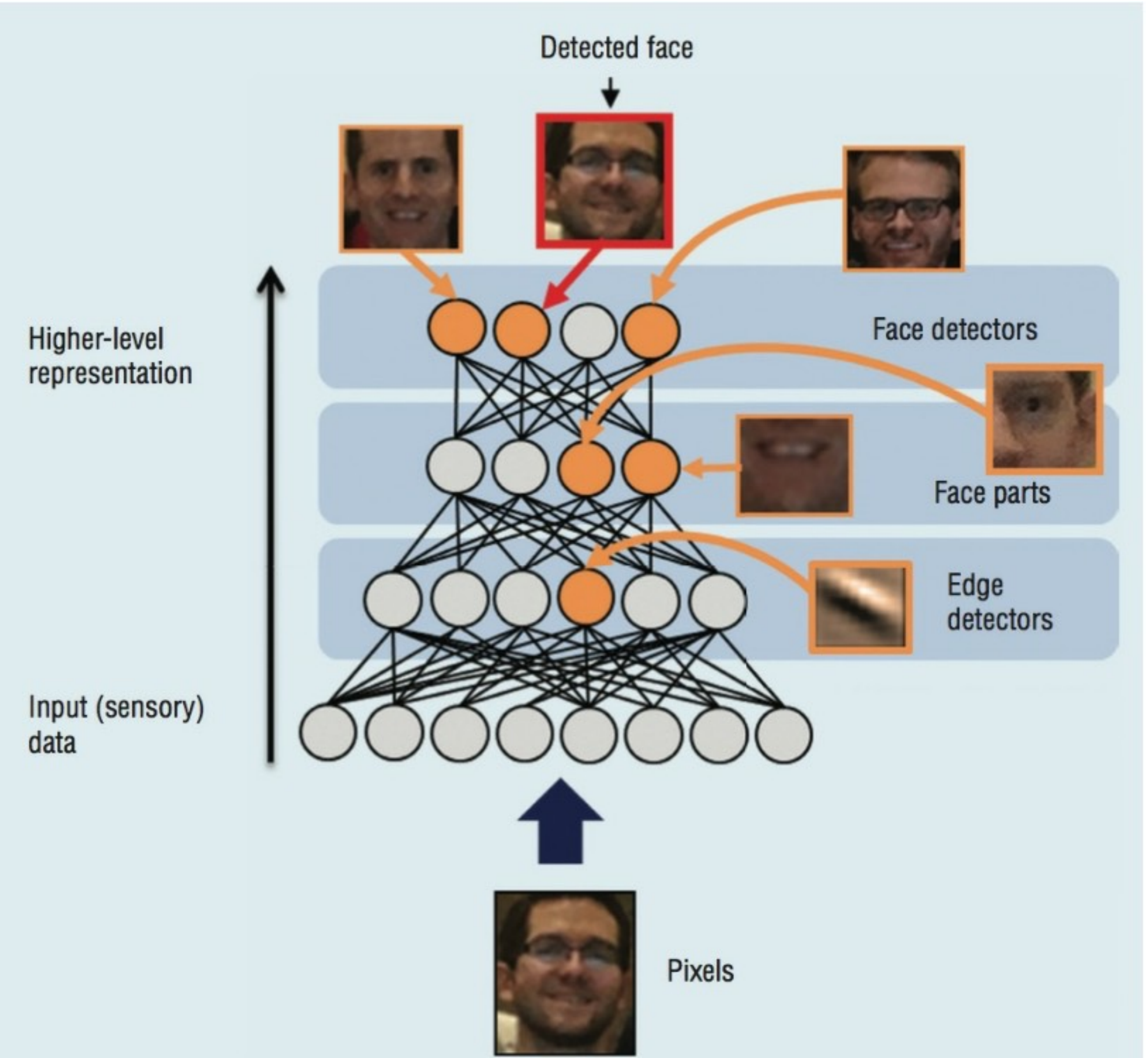
- Both downsample the image

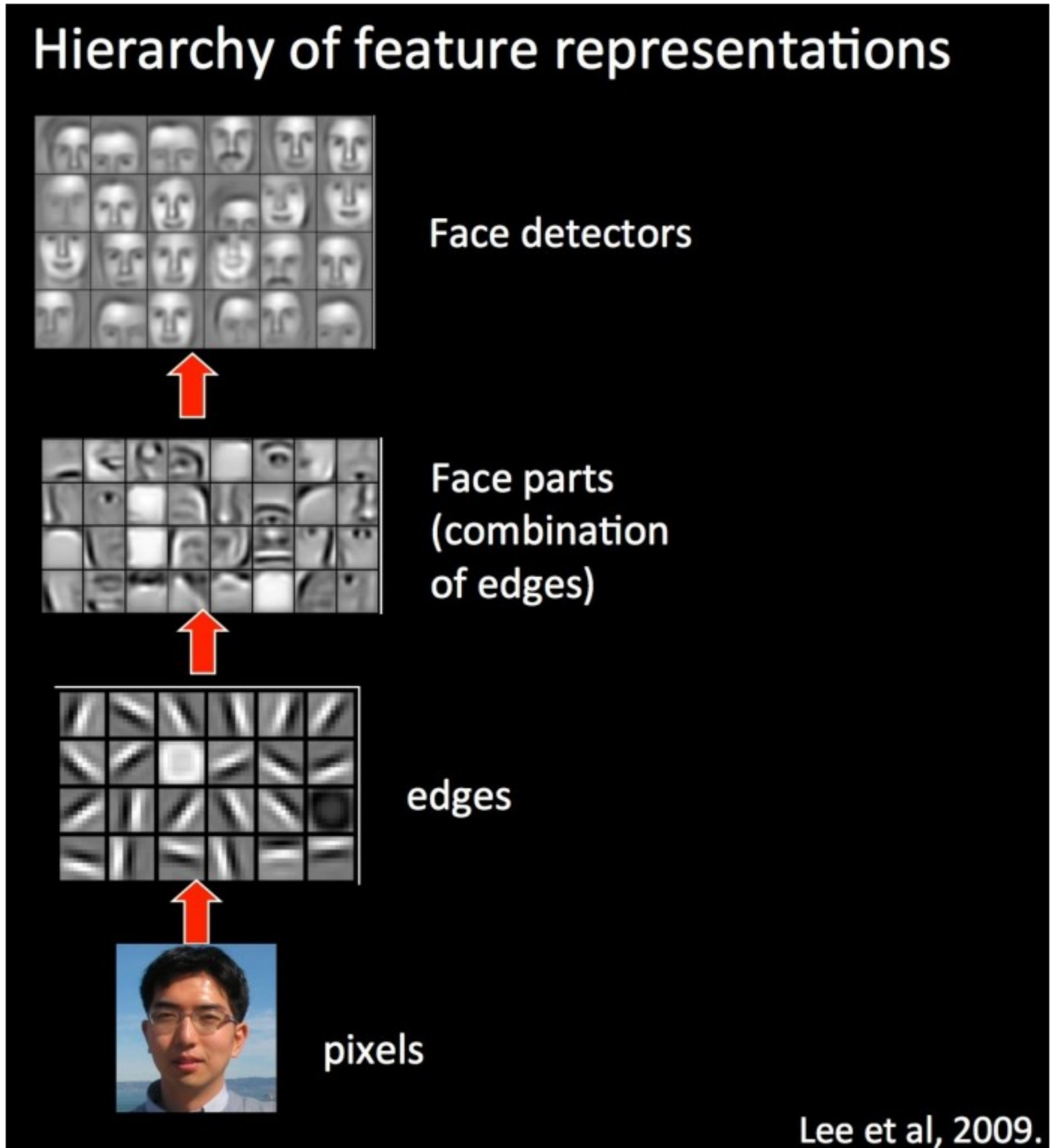
- Max pooling: “any of these options is present”
  - Much more common, especially in early layers
  - “There’s an edge here, but I don’t really care how thick it is”

- Average pooling: “all/most of these options are present”
  - If used, more often at the end of the network
  - “Most of the big patches look like a picture of a train“


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

Hierarchically composed feature representations

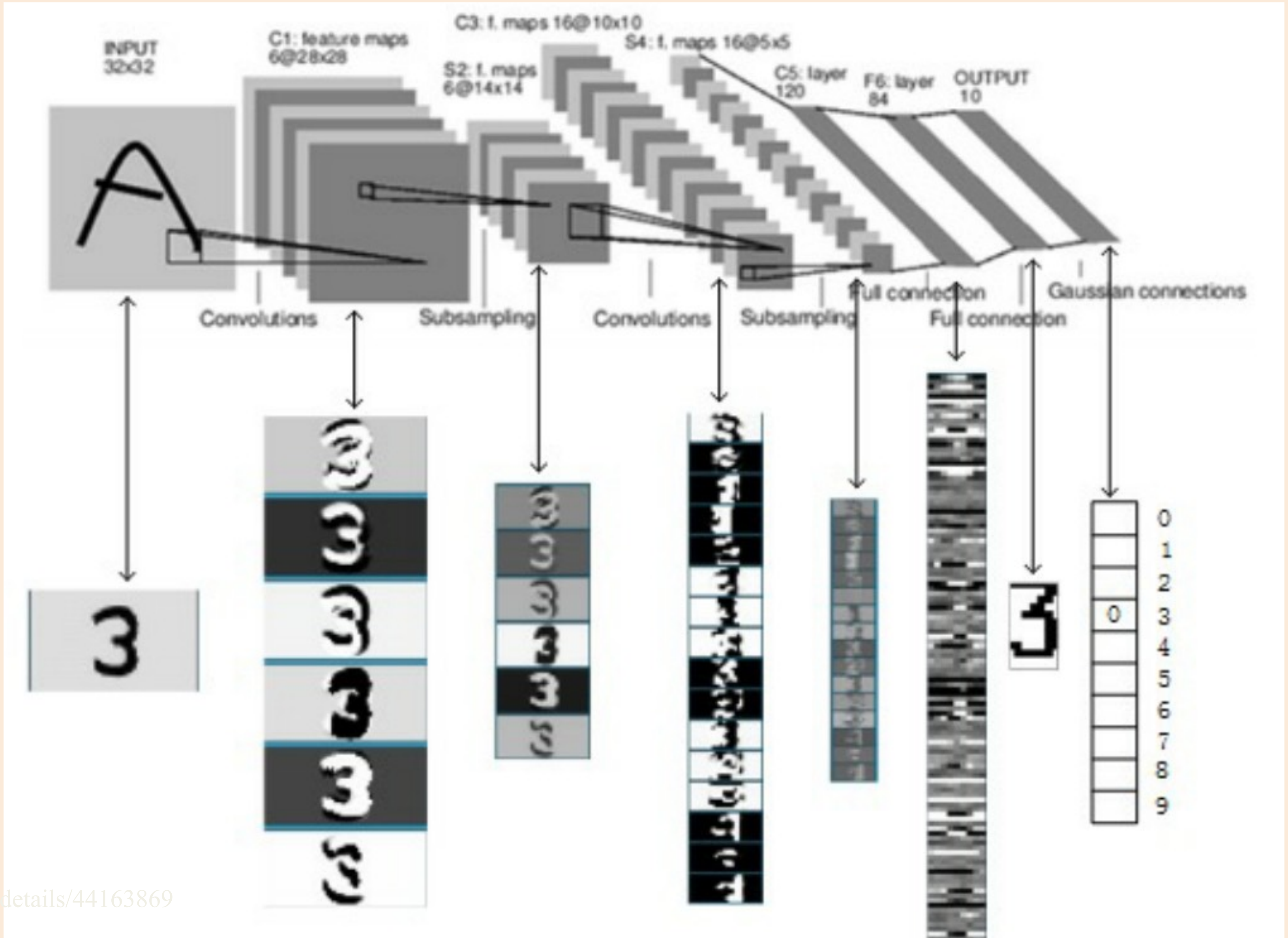



Hierarchy of feature representations
Face detectors
Face parts (combination of edges)
edges
pixels
Lee et al, 2009.

DeepViz Toolbox

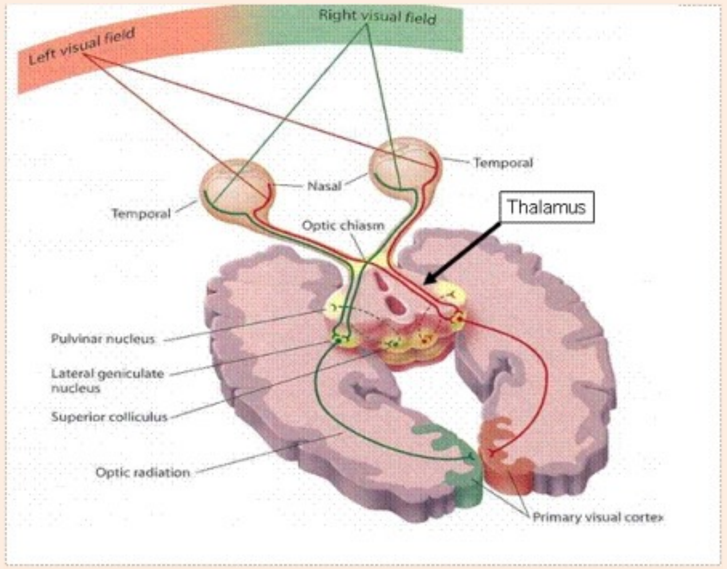
- We're ready to watch this now!
- https://youtu.be/AgkfIQ4IGaM

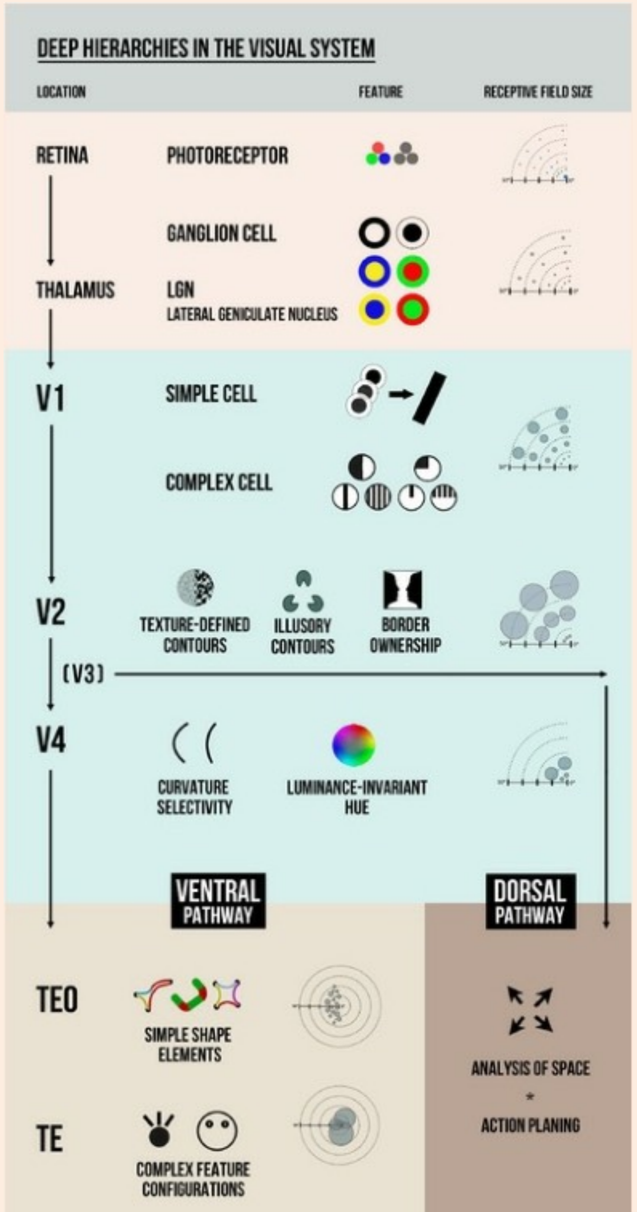
LeNet for Optical Character Recognition

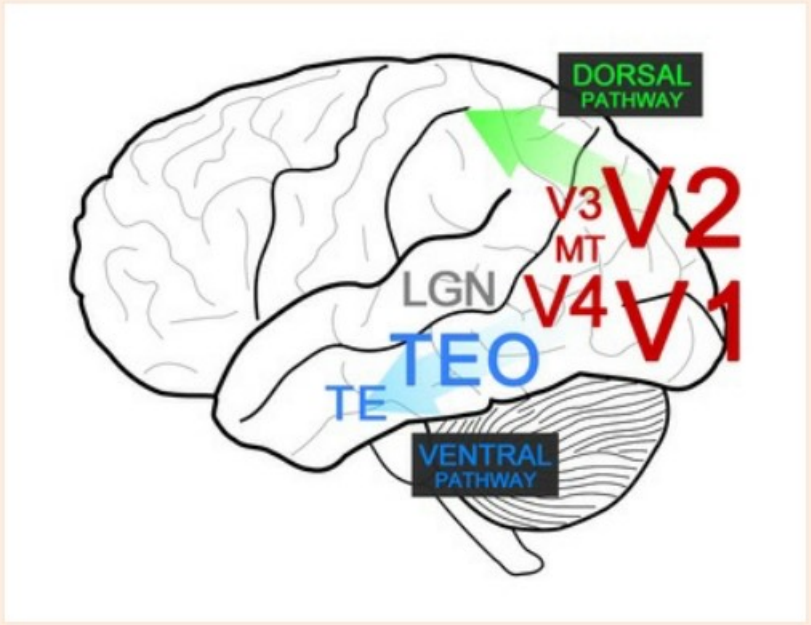


bonus!

Deep Hierarchies in the Visual System



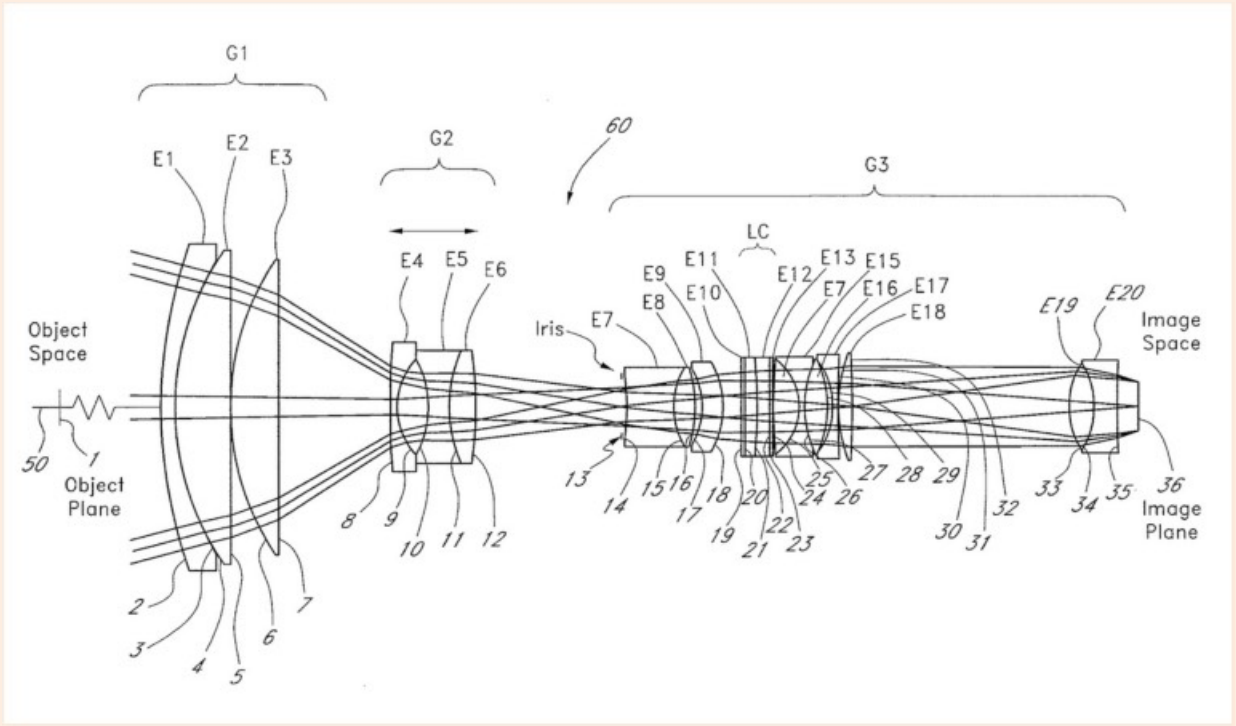


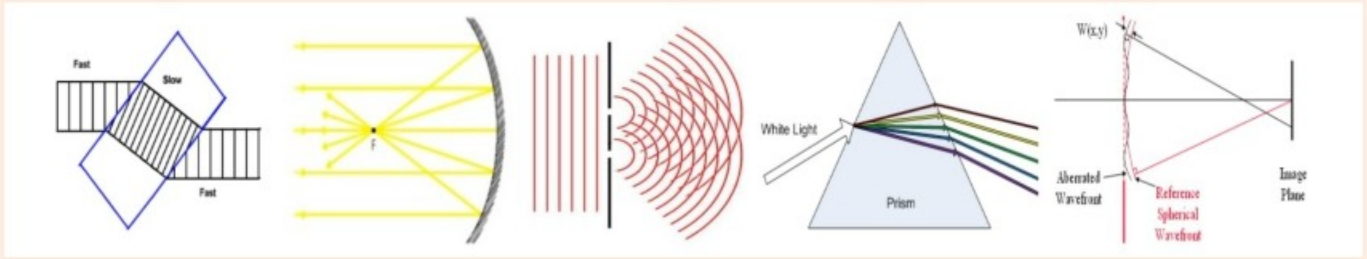


http://www.strokenetwork.org/newsletter/articles/vision.htm
https://en.wikibooks.org/wiki/Sensory_Systems/Visual_Signal_Processing

bonus!

Deep Hierarchies in Optics





http://www.argmin.net/2018/01/25/optics/

bonus!

DenseNet

- More recent variation is “DenseNets ”:
  - Each layer can see all the values from many previous layers.
  - Gets rid of vanishing gradients.
  - May get same performance with fewer parameters/layers.

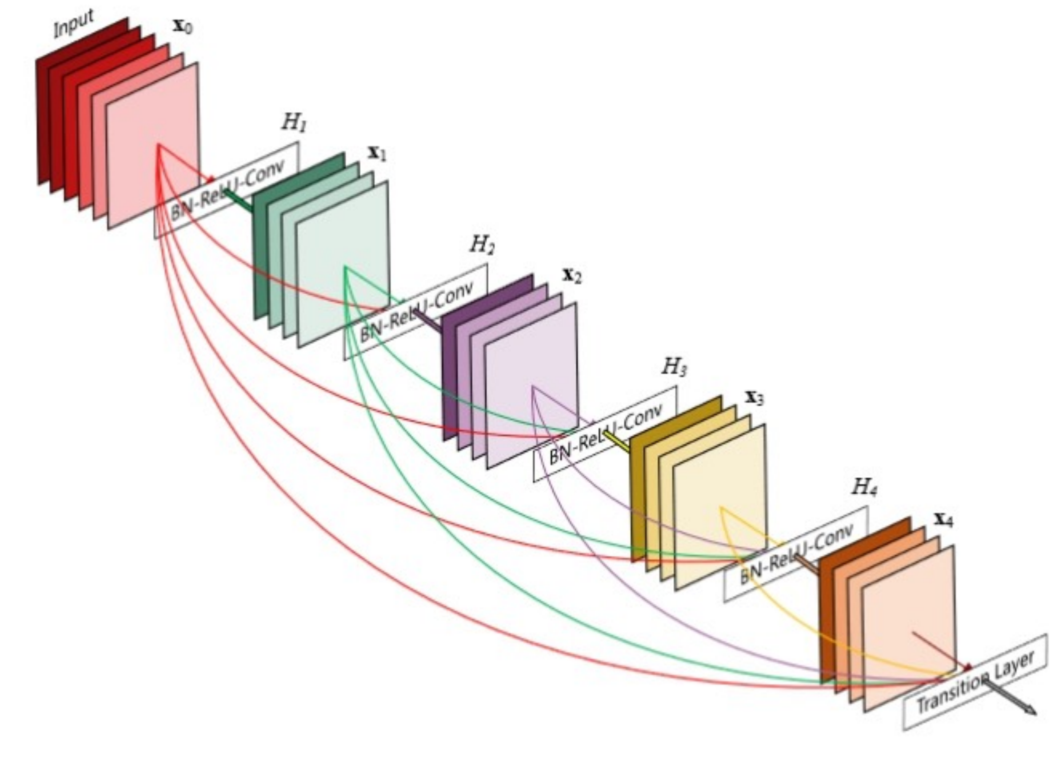


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

https://arxiv.org /abs /1512.03385

Deep Learning and the Fundamental Trade- Off

- Neural networks are subject to the fundamental trade-off:
  - With increasing depth, training error of global optima decreases.
  - With increasing depth, training error may poorly approximate test error.

- We want deep networks to model highly non-linear data.
  - But increasing the depth can lead to overfitting.

- How could GoogLeNet use 22 layers?
  - Many forms of regularization and keeping model complexity under control.
  - Unlike linear models, typically use multiple types of regularization.

Standard Regularization

- Traditionally, we've added our usual L2 -regularizers :

$$f(v,W^{(3)},W^{(2)},W^{(1)}) = \frac{1}{2}\sum_{i=1}^{n}(v^T h(W^{(3)}h(W^{(2)}h(W^{(1)}x_i))) - y_i)^2 + \frac{\lambda_4}{2}\|v\|^2 + \frac{\lambda_3}{2}\|W^{(3)}\|_F^2 + \frac{\lambda}{2}\|W^{(2)}\|_F^2 + \frac{\lambda_3}{2}\|W^{(1)}\|_F^2$$

- L2 -regularization often called "weight decay " in this context.
  - Could also use L1- regularization: gives sparse network .

Standard Regularization

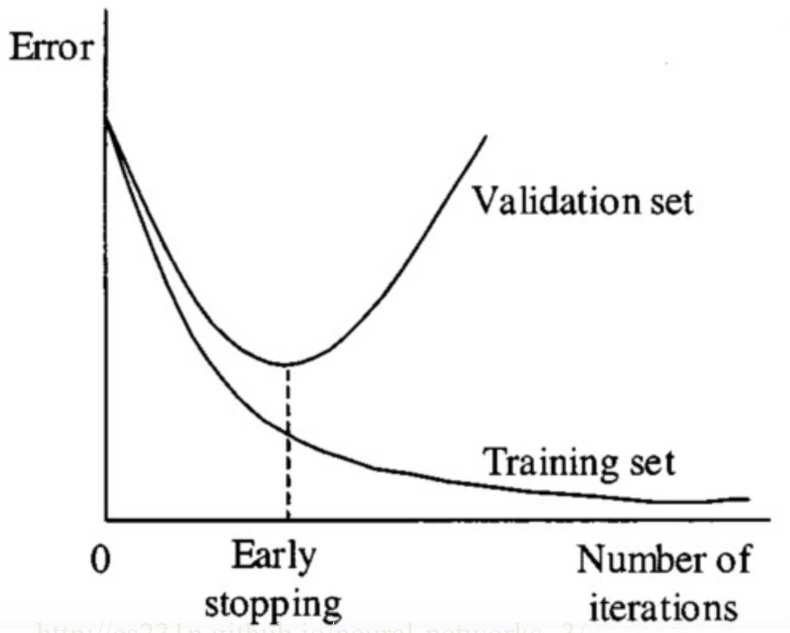
- Traditionally, we've added our usual L2 -regularizers :

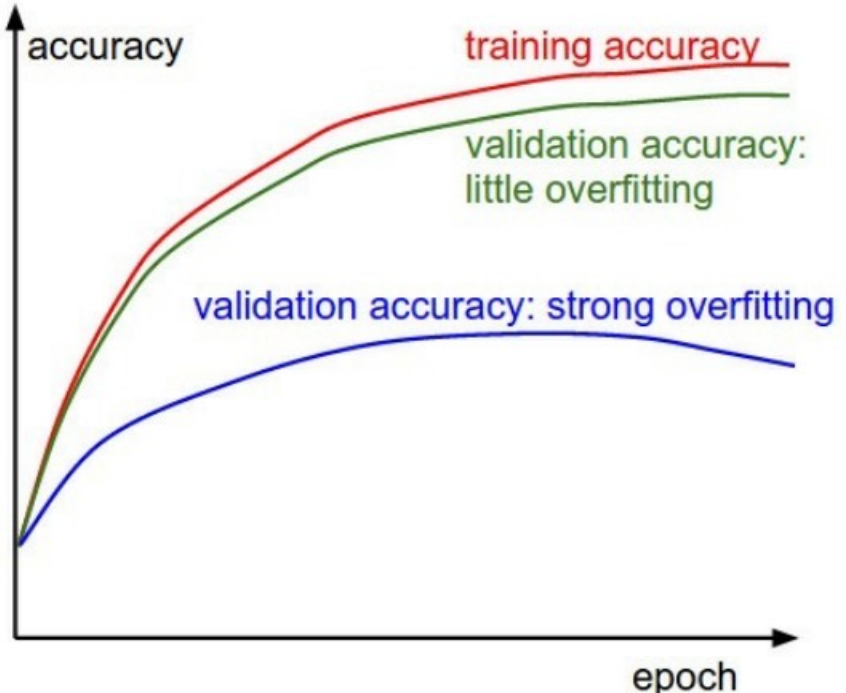
$$f(v, W^{(3)}, W^{(2)}, W^{(1)}) = \frac{1}{2}\sum_{i=1}^{n}\left(v^T h(W^{(3)} h(W^{(2)} h(W^{(1)} x_i))) - y_i\right)^2 + \frac{\lambda_4}{2}\|v\|^2 + \frac{\lambda_3}{2}\|W^{(3)}\|_F^2 + \frac{\lambda}{2}\|W^{(2)}\|_F^2 + \frac{\lambda}{2}\|W^{(1)}\|_F^2$$

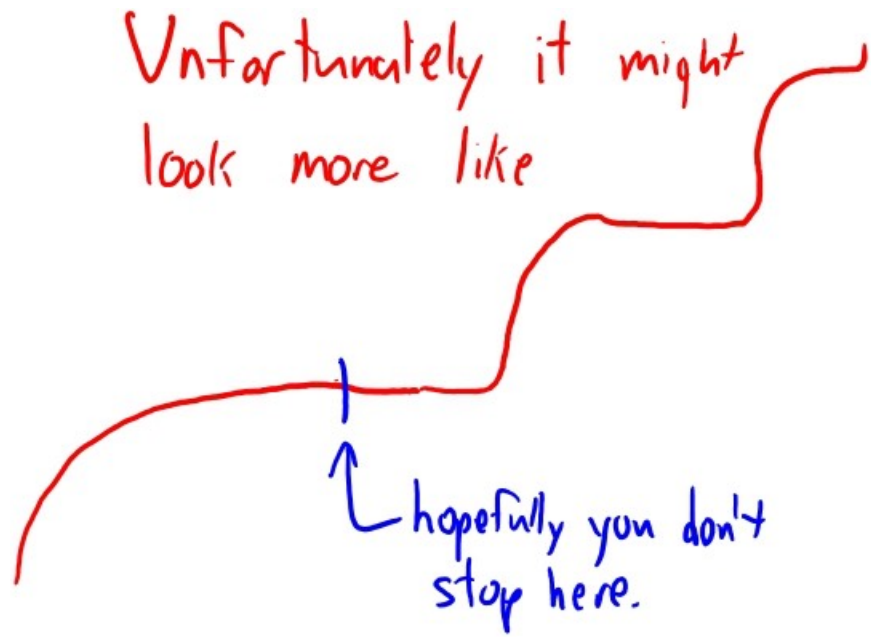
- L2 -regularization often called "weight decay " in this context.
  - Adds ! W to gradient, so (S)GD "decays" the weights 'W 'at each step
  - Could also use L1- regularization: gives sparse network .
- Hyper-parameter optimization gets expensive:
  - Try to optimize validation error in terms of λ_1 , λ_2 , λ_3 , λ_4 .
  - In addition to step- size, number of layers, size of layers, initialization.
- Recent result:
  - Adding a regularizer in this way can create bad local optima .

Early Stopping

- Another common type of regularization is "early stopping":
  - Monitor the validation error as we run stochastic gradient .
  - Stop the algorithm if validation error starts increasing .



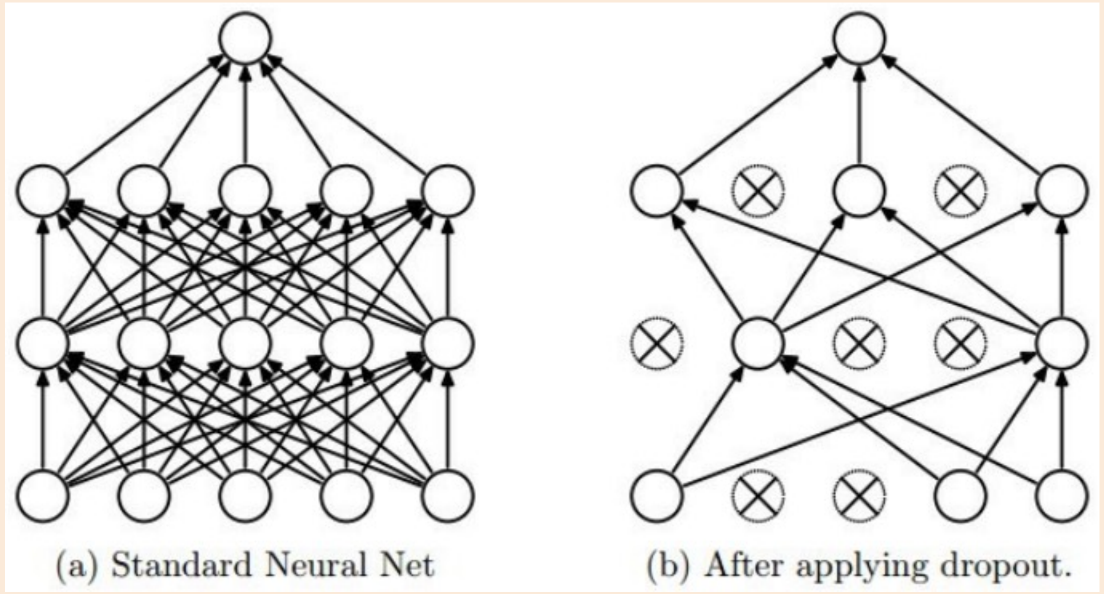




bonus!

Dropout

- Dropout is a more recent form of explicit regularization:
  - On each iteration, randomly set some x_i and z_i to zero (often use 50%).



  - Adds invariance to missing inputs or latent factors
    - Encourages distributed representation rather than relying on specific z_i .
  - Can be interpreted as an ensemble over networks with different parts missing.
  - After a lot of early success, dropout is already kind of going out of fashion.

http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

Discussion & Summary of CNNs

- Convolutional layers reduce the number of parameters in several ways:
  - Each hidden unit only depends on small number of inputs from previous layer.
  - We use the same filters across the image .
    - So we do not learn a different weight for each "connection" like in classic neural networks.
  - Pooling layers decrease the image size .

- CNNs give some amount of translation invariance :
  - Because same filters used across the image, they can detect a pattern anywhere in the image .
    - Even in image locations where the pattern has never been seen.

- CNNs are not only for images!
  - Can use CNNs for 1D sequences like sound or language or biological sequences.
  - Can use CNNs for 3D objects like videos or medical image volumes.
  - Can use CNNs for graphs.
- But you do need some notion of "neighbourhood " for convolutions to make sense.
- Regularization is crucial to neural net performance (convolutional or otherwise)

  – L2- regularization, early stopping, dropout, implicit regularization of SGD.

(end, tested technical material)

Some high-level principles, and ethical issues we will cover are still testable