CPSC 340: Machine Learning and Data Mining

Convolutional Neural Networks

Fall 2019

Admin – Lectures this week

- Planned bus strike Wednesday-Friday.

- I’m planning to finish the “testable content” of the course today.
  - I might go a bit over time.

- Wednesday will be about “fun with deep learning”.

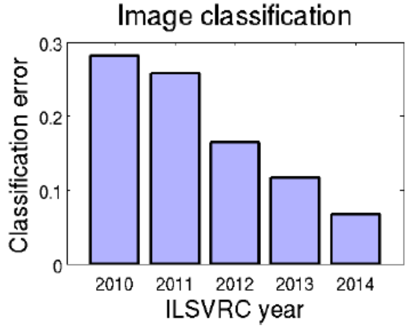
- Friday, I might cover different topics in the different sections:
  - Possible topics include semi-supervised learning, Google’s PageRank, or proofs:
    - How many gradient descent iterations do we need?
    - What does the approximation error depend on?

Last Lectures: Deep Learning

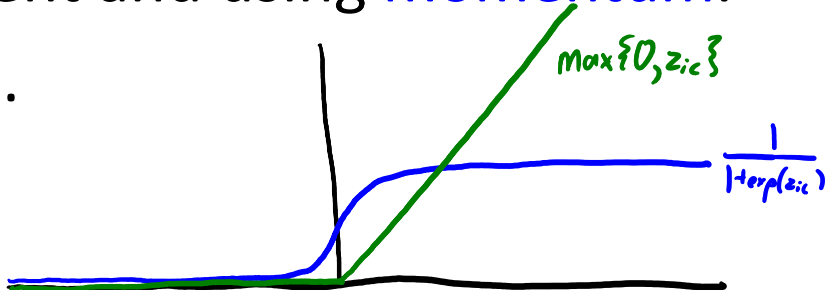
- We've been discussing neural network / deep learning models:

$$\hat{y}_i = v^T h(W^{(m)} h(W^{(m-1)} h(\cdots\cdots W^{(2)} h(W^{(1)} x_i))\cdots))$$

- We discussed unprecedented vision/speech performance.



- We discussed methods to make SGD work better:
  - Parameter initialization and data transformations.
  - Setting the step size(s) in stochastic gradient and using momentum.
  - Alternative non-linear functions like ReLU.



https://arxiv.org/pdf/1409.0575v3.pdf

“Residual” Networks (ResNets)

- Impactful recent idea is residual networks (ResNets):

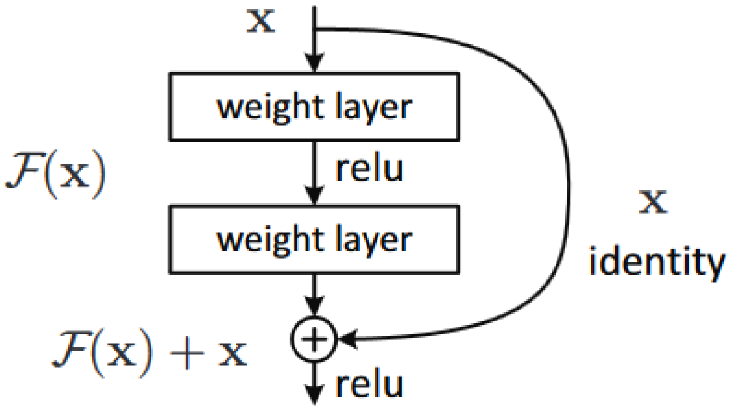

x
weight layer
$\mathcal{F}(\mathrm{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathrm{x}) + \mathrm{x}$
relu

Figure 2. Residual learning: a building block.

  - You can take previous (non-transformed) layer as input to current layer.
    - Also called “skip connections” or “highway networks”.
  - Non-linear part of the network only needs to model residuals.
    - Non-linear parts are just “pushing up or down” a linear model in various places.
  - This was a key idea behind first methods that used 100+ layers.
    - Evidence that biological networks have skip connections like this.

https://en.wikipedia.org/wiki/Residual_neural_network

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.

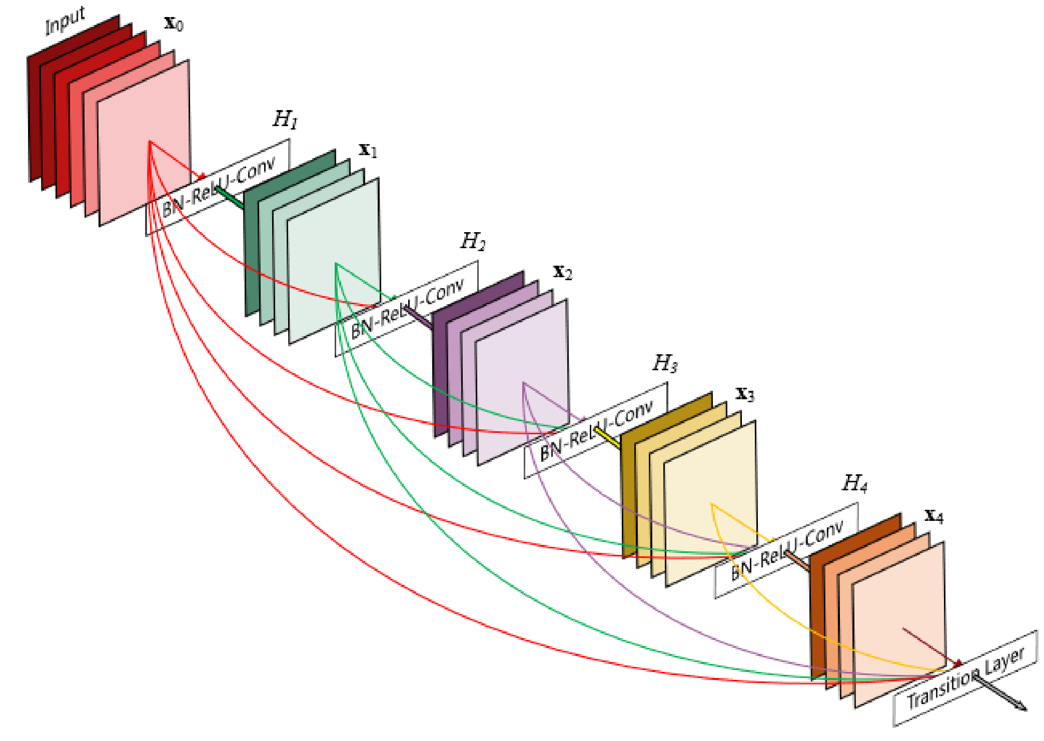


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

https://arxiv.org/pdf/1512.03385v1.pdf

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.

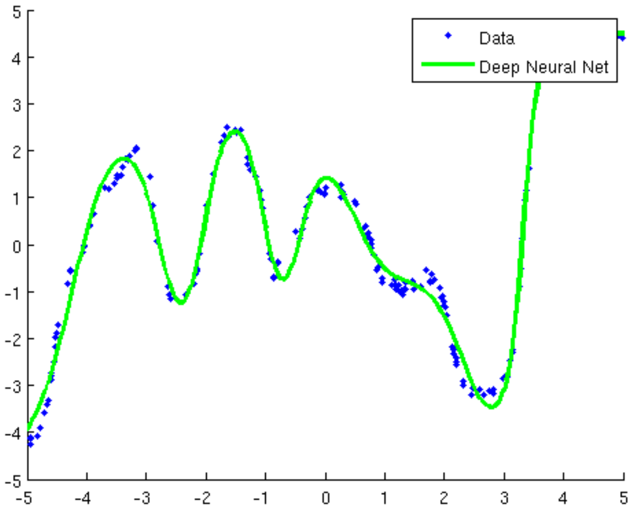
- How could GoogLeNet use 22 layers?
  - Many forms of regularization and keeping model complexity under control.
  - Unlike linear models, typically use multiple types of regularization.

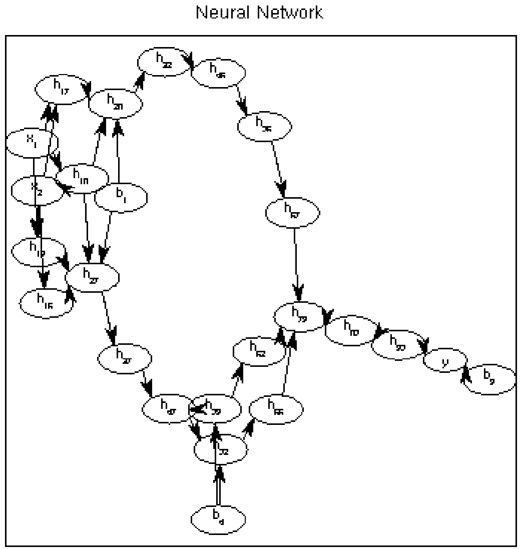
Standard Regularization

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

$$f(v, W^{(3)}, W^{(2)}, W^{(1)}) = \frac{1}{2}\sum_{i=1}^{n}\left(v^T h(W^{(3)} h(W^{(2)} h(W^{(1)} x_i))) - y_i\right)^2 + \frac{\lambda_4}{2}\|v\|^2 + \frac{\lambda_3}{2}\|W^{(3)}\|_F^2 + \frac{\lambda}{2}\|W^{(2)}\|_F^2 + \frac{\lambda_1}{2}\|W^{(1)}\|_F^2$$

- 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:

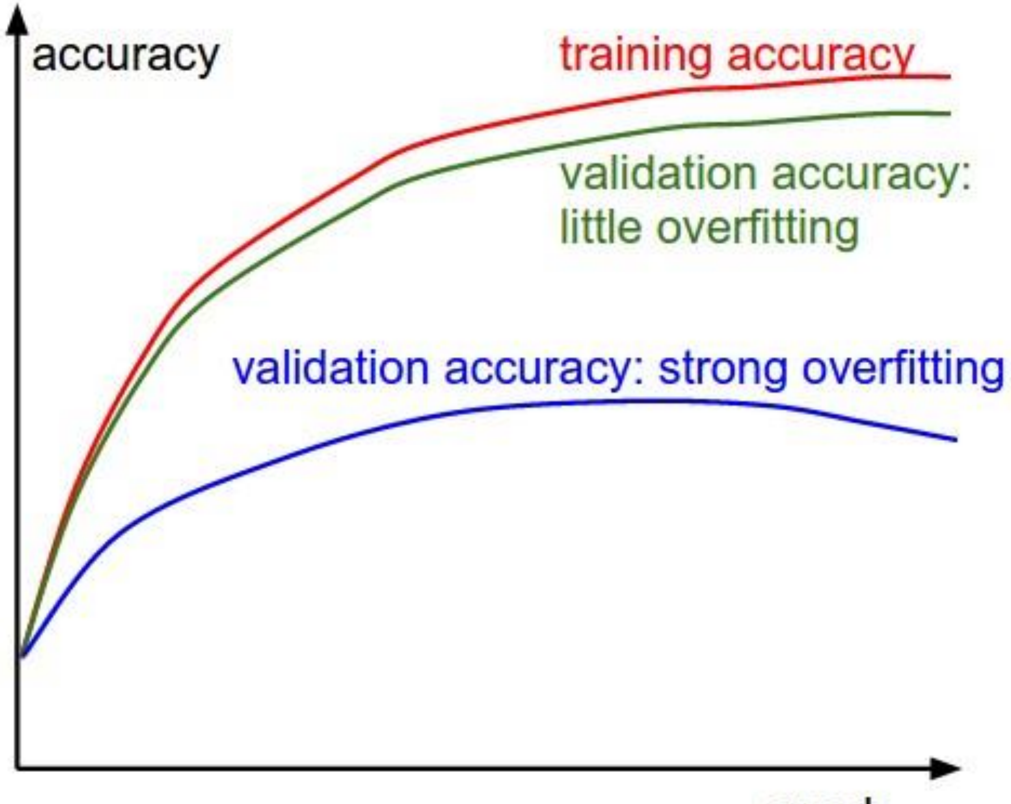
$$f(v, W^{(3)}, W^{(2)}, W^{(1)}) = \frac{1}{2}\sum_{i=1}^{n}\left(v^T h(W^{(3)} h(W^{(2)} h(W^{(1)} x_i))) - y_i\right)^2 + \frac{\lambda_4}{2}\|v\|^2 + \frac{\lambda_3}{2}\|W^{(3)}\|_F^2 + \frac{\lambda_2}{2}\|W^{(2)}\|_F^2 + \frac{\lambda_1}{2}\|W^{(1)}\|_F^2$$

- L2-regularization often called “weight decay” in this context.
  - Could also use L1-regularization: gives sparse network.
- “Hyper-parameter” optimization:
  - Try to optimize validation error in terms of λ_1, λ_2, λ_3, λ_4.

- Recent result:
  - Adding a regularizer in this way creates bad local optima.

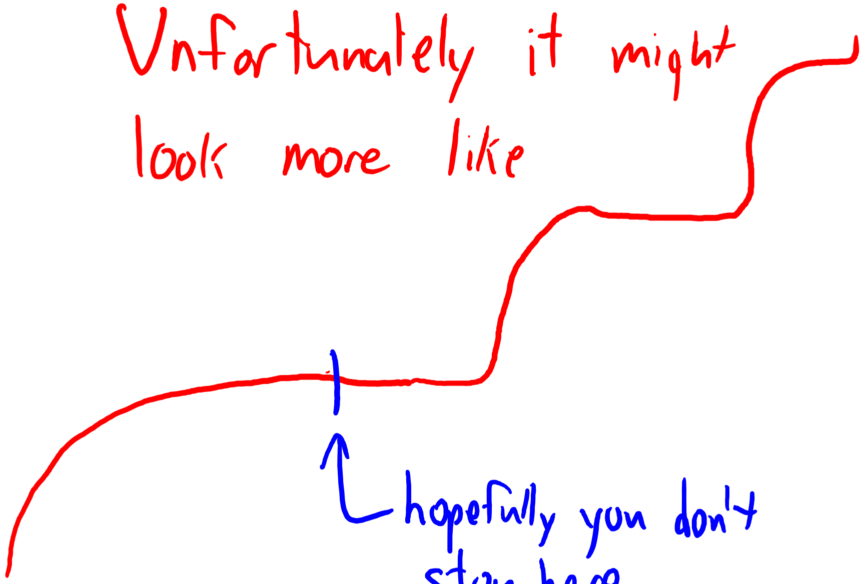
Early Stopping

- 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.



http://cs231n.github.io/neural-networks-3/

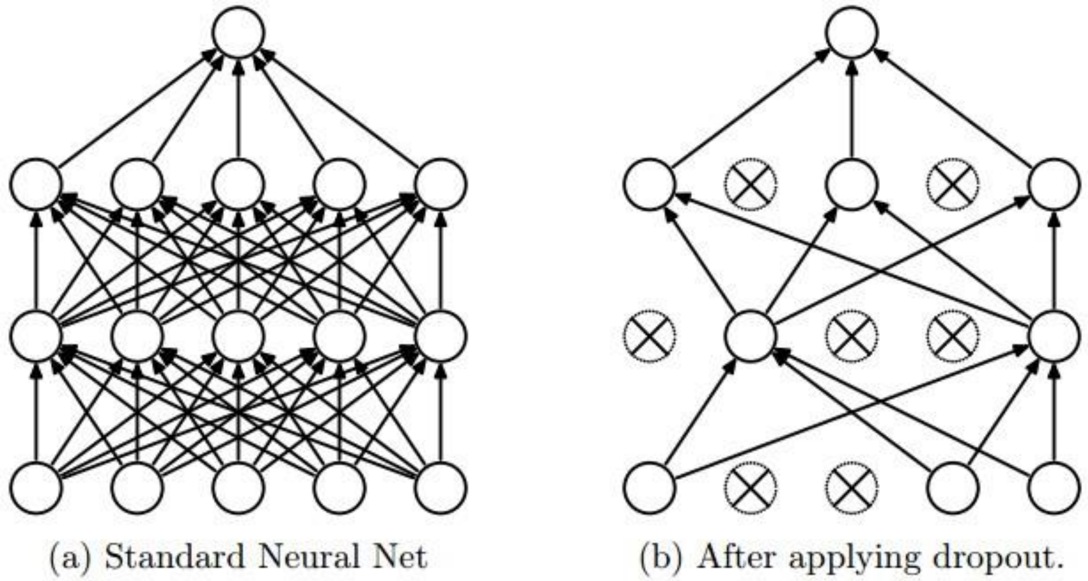
Unfortunately it might look more like



hopefully you don't stop here.

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%).

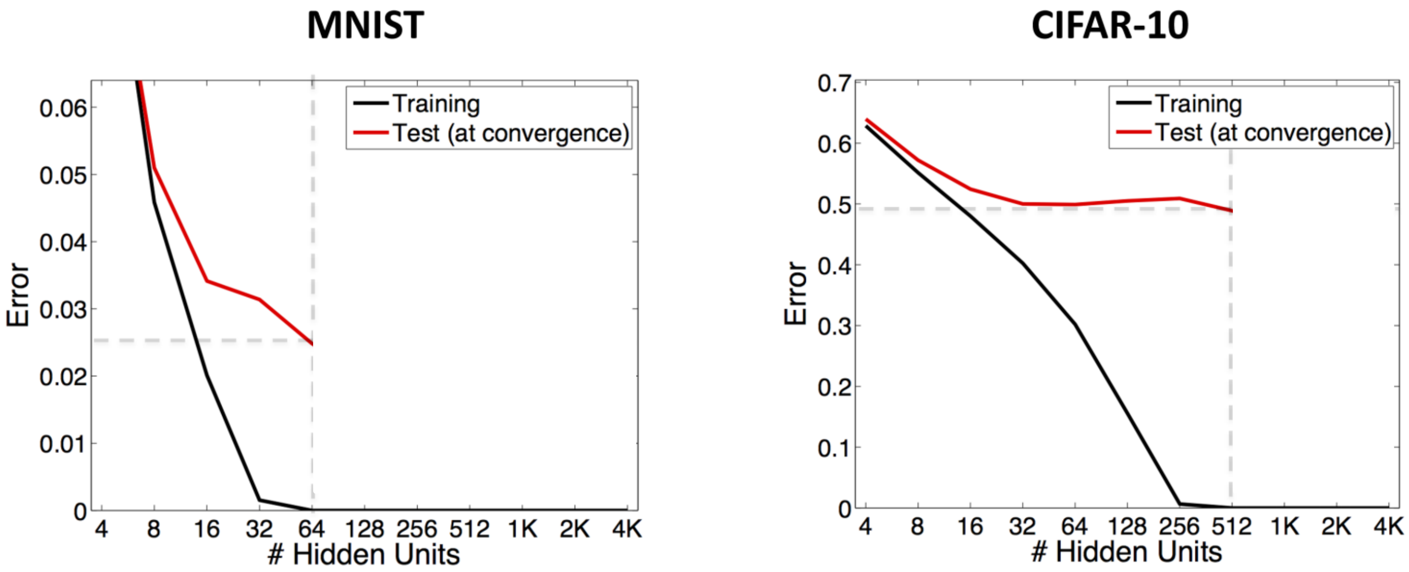


  - Encourages distributed representation rather than relying on specific z_i.
    - Alternately, you are adding invariance to missing inputs or latent factors.
  - After a lot of success, dropout may already be going out of fashion.

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

“Hidden” Regularization in Neural Networks

- Fitting single-layer neural network with SGD and no regularization:



- Training goes to 0 with enough units: we’re finding a global min.

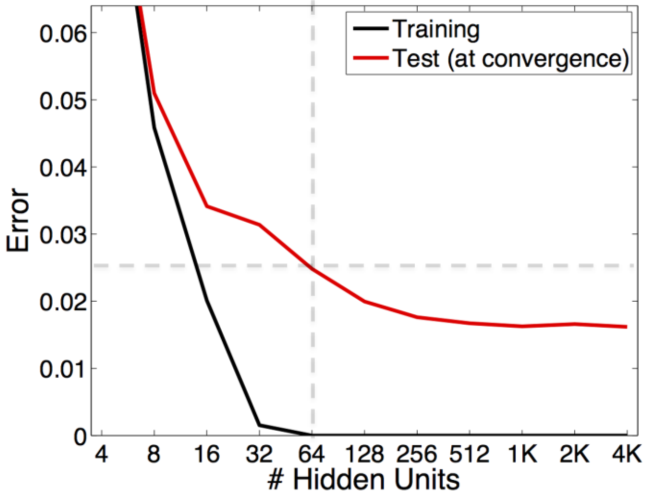
- What should happen to training and test error for larger #hidden?

https://www.neyshabur.net/papers/inductive_bias_poster.pdf

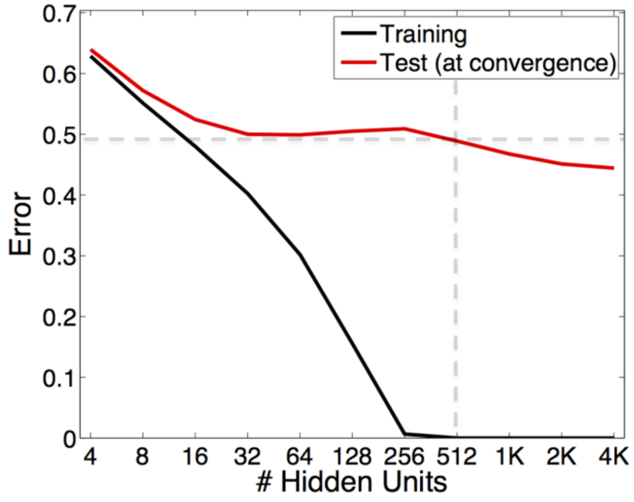
“Hidden” Regularization in Neural Networks

- Fitting single-layer neural network with SGD and no regularization:

MNIST



CIFAR-10



- Test error continues to go down!?! Where is fundamental trade-off??
- There exist global mins with large #hidden units have test error = 1.
  - But among the global minima, SGD is somehow converging to “good” ones.

https://www.neyshabur.net/papers/inductive_bias_poster.pdf

Implicit Regularization of SGD

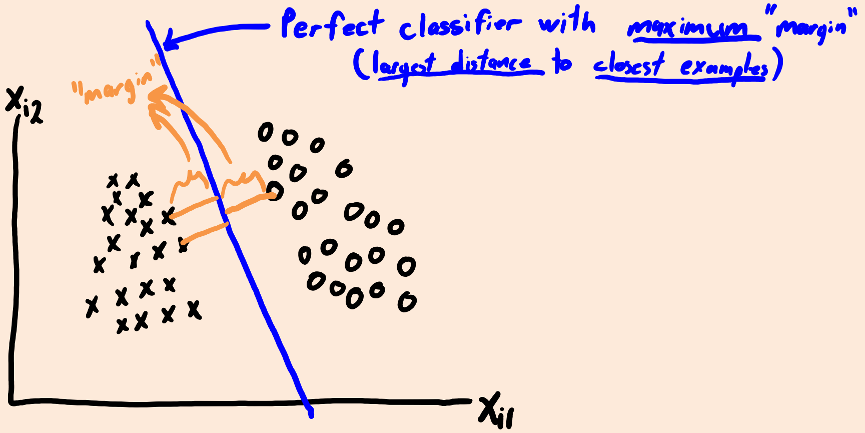
- There is growing evidence that using SGD regularizes parameters.
  - We call this the "implicit regularization" of the optimization algorithm.

- Beyond empirical evidence, we know this happens in simpler cases.

- Example of implicit regularization:
  - Consider a least squares problem where there exists a 'w' where Xw=y.
    - Residuals are all zero, we fit the data exactly.
  - You run [stochastic] gradient descent starting from w=0.
  - Converges to solution Xw=y that has the minimum L2-norm.
    - So using SGD is equivalent to L2-regularization here, but regularization is "implicit".

Implicit Regularization of SGD

- Example of implicit regularization:
  - Consider a logistic regression problem where data is linearly separable.
    - We can fit the data exactly.
  - You run gradient descent from any starting point.
  - Converges to max-margin solution of the problem.
    - So using gradient descent is equivalent to encouraging large margin.


Perfect classifier with maximum "margin"
(largest distance to closest examples)
"margin"
x_{i2}
x_{i1}

- Similar result known for boosting.

(pause)

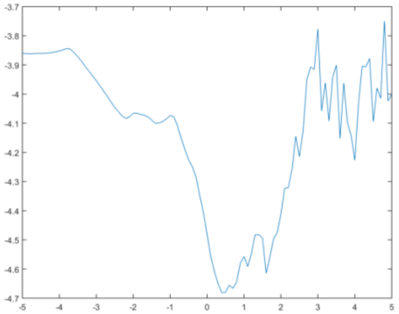
Deep Learning “Tricks of the Trade”

- We’ve discussed heuristics to make deep learning work:
  - Parameter initialization and data transformations.
  - Setting the step size(s) in stochastic gradient and using momentum.
  - RestNets and alternative non-linear functions like ReLU.
  - Different forms of regularization:
    - L2-regularization, early stopping, dropout, implicit regularization from SGD.
- These are often still not enough to get deep models working.
- Deep computer vision models are all convolutional neural networks:
  - The $W^{(m)}$ are very sparse and have repeated parameters (“tied weights”).
  - Drastically reduces number of parameters (speeds training, reduces overfitting).

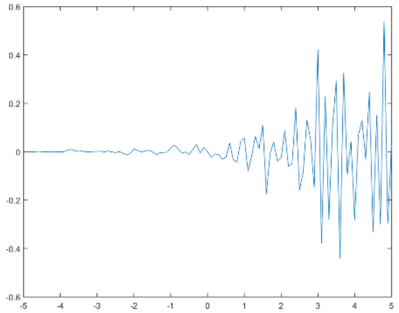
1D Convolution as Matrix Multiplication

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

$x * w = z$



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



  - Can be written as a matrix multiplication:

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

1D Convolution as Matrix Multiplication

- Each element of a convolution is an inner product:

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

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

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

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

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

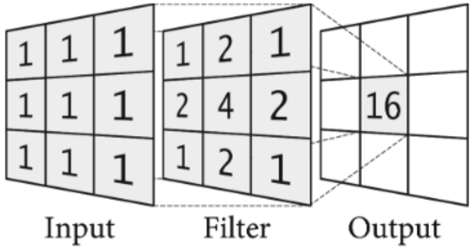
- The shorter 'w' is, the more sparse the matrix is.

2D Convolution as Matrix Multiplication

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

$$x \quad * \quad w \quad = \quad z$$

$$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$



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

$$W = \left[\begin{array}{cccccccc|cccccccc|cccccccc|ccc}
-2 & -1 & 0 & 0 & 0 & 0 & \cdots & 0 & -1 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 2 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots \\
0 & -2 & -1 & 0 & 0 & 0 & \cdots & 0 & 0 & -1 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 & 2 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots \\
 & & \vdots & & \cdots & 2 & -1 & 0 & 0 & 0 & 0 & \cdots & & -1 & 0 & 1 & 0 & 0 & 0 & \cdots & & 0 & 1 & 2 & 0 & 0 & \cdots \\
0 & 0 & 0 & \cdots & & & & & & & & & & & & & & & & & & & & & & & \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 2 & -1 & 0 & 0 & 0 & 0 & \cdots & 0 & -1 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 2 & -1 & 0 & 0 & 0 & \cdots & 0 & 0 & -1 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\
 & & & \vdots & & & & & & & & \vdots & & & & & & & & & \vdots & & & & & &
\end{array}\right.$$

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.

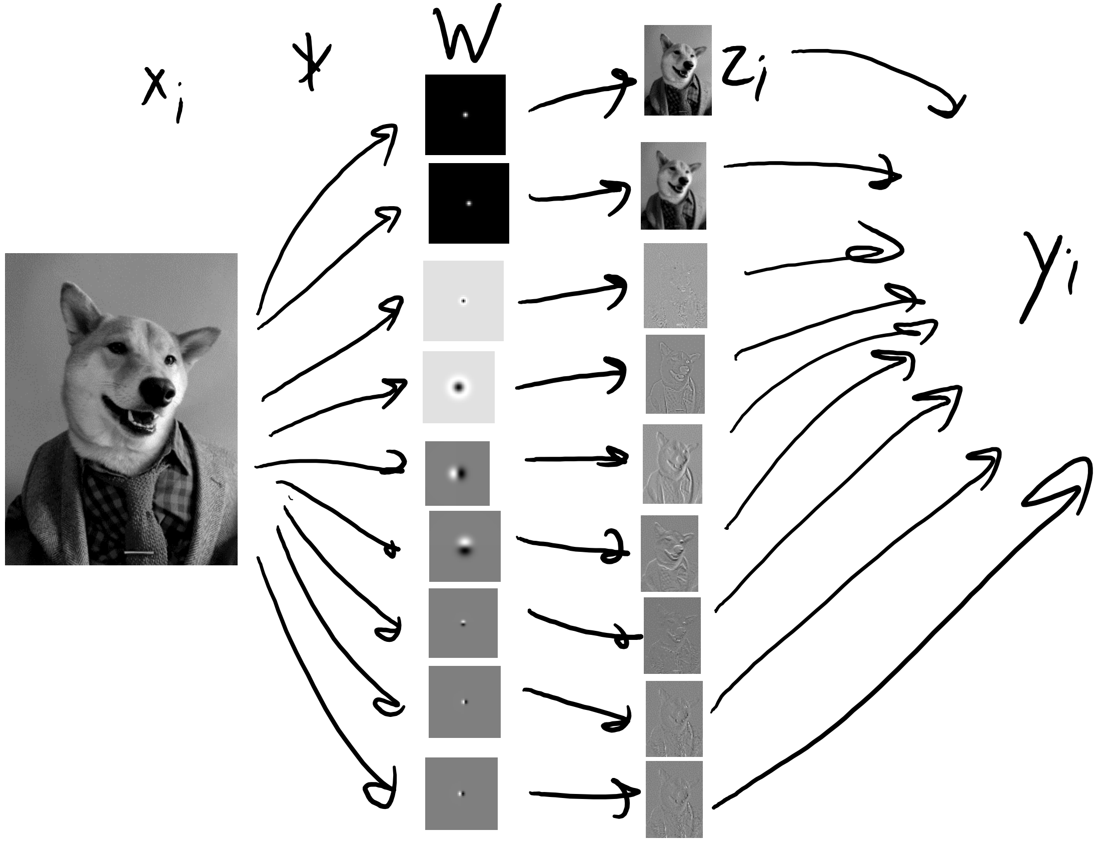
$$w_1 = [0 \quad 0 \quad 0 \ \text{——} \ w \ \text{——} \ 0 \quad 0 \quad 0]$$

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

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

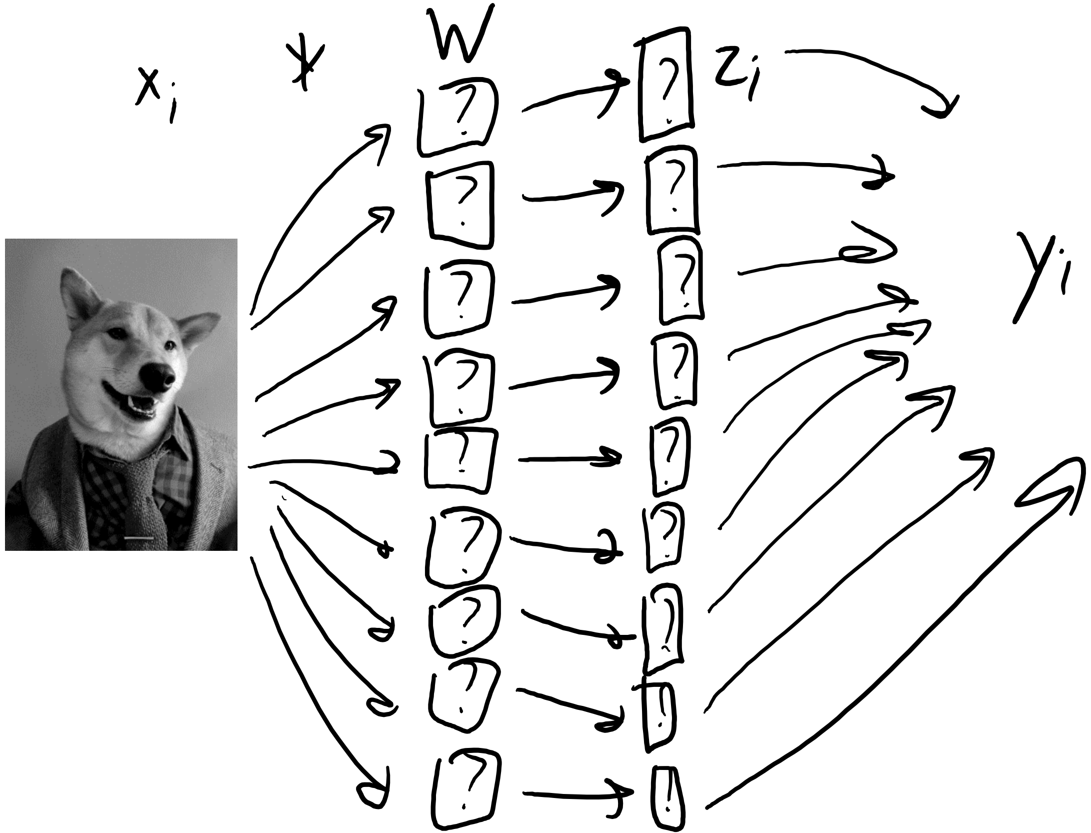
Motivation for Convolutional Neural Networks

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



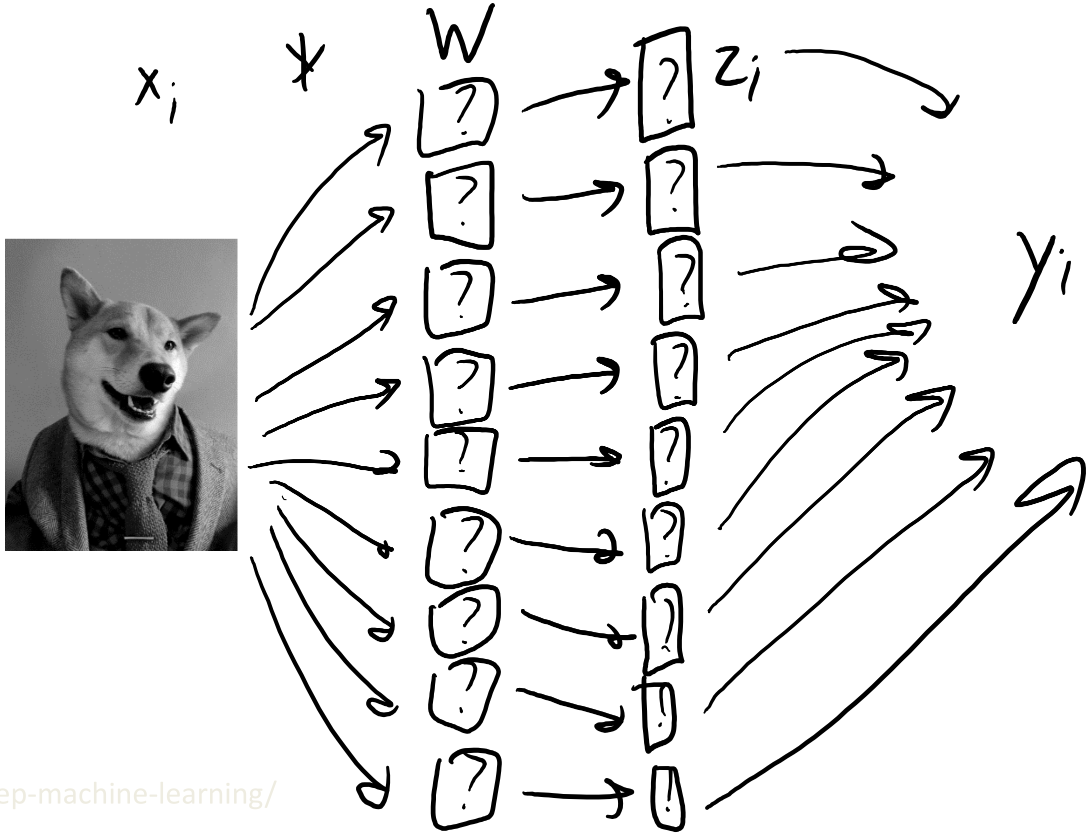
Motivation for Convolutional Neural Networks

- 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.

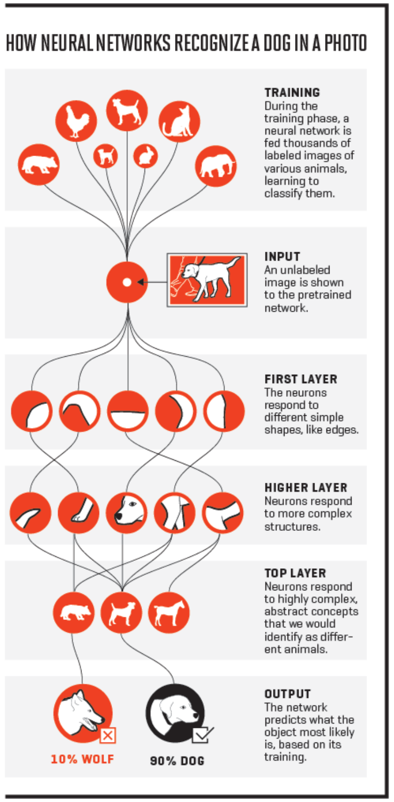


Motivation for Convolutional Neural Networks

- 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.



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



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.

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.

1D example

distance between centers of convolution is called "stride"

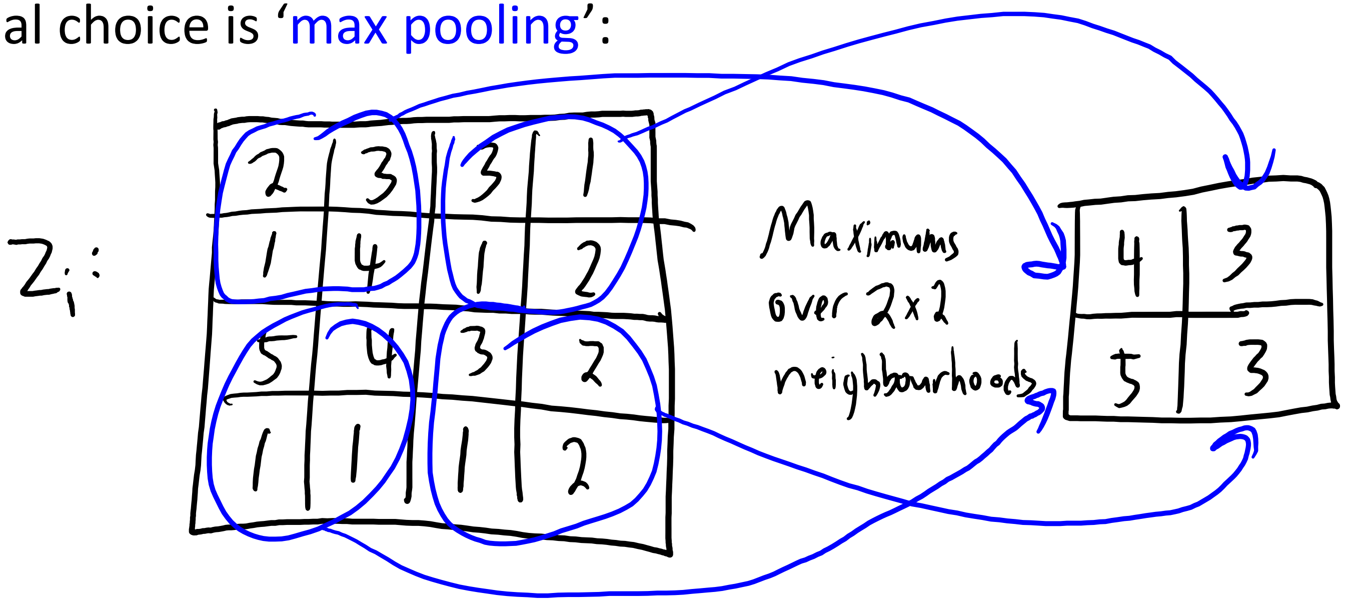
$$W^{(m)} = \begin{bmatrix} \text{———} & w_1^{(m)} & \text{———} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \text{———} & w_1^{(m)} & \text{———} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \text{———} & w_1^{(m)} & \text{———} \\ \text{———} & w_2^{(m)} & \text{———} & 0 & 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & \text{———} & w_2^{(m)} & \text{———} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \text{———} & w_2^{(m)} & \text{———} \end{bmatrix}$$

Same $w_1^{(m)}$ used across multiple rows.

Sparse and small number of parameters

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.
    - Usual choice is 'max pooling':



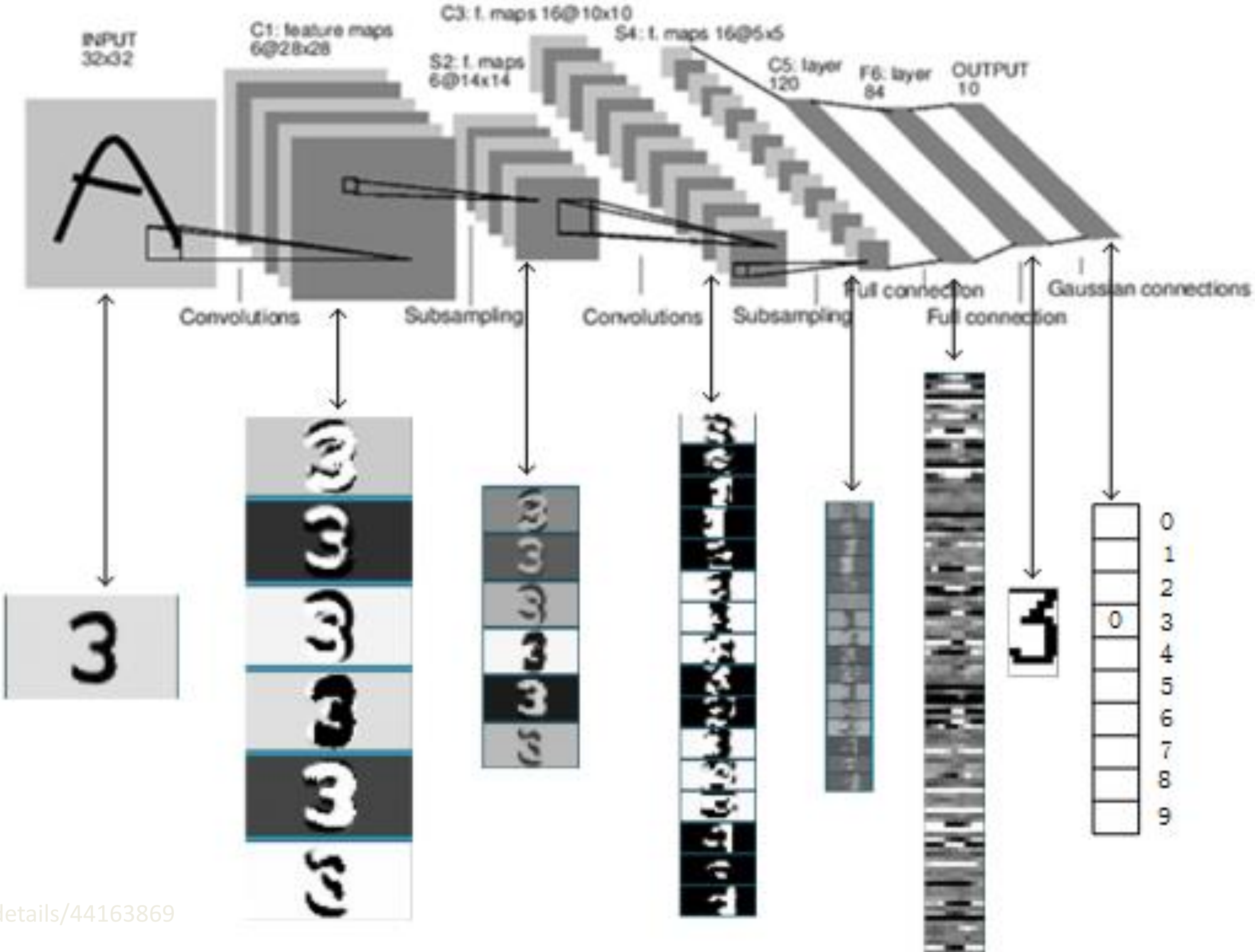
z_i:

2	3	3	1
1	4	1	2
5	4	3	2
1	1	1	2

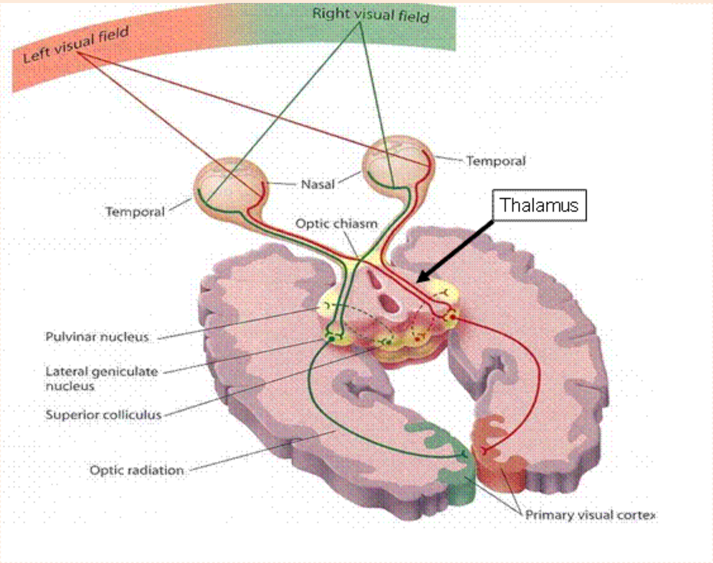
Maximums over 2x2 neighbourhoods

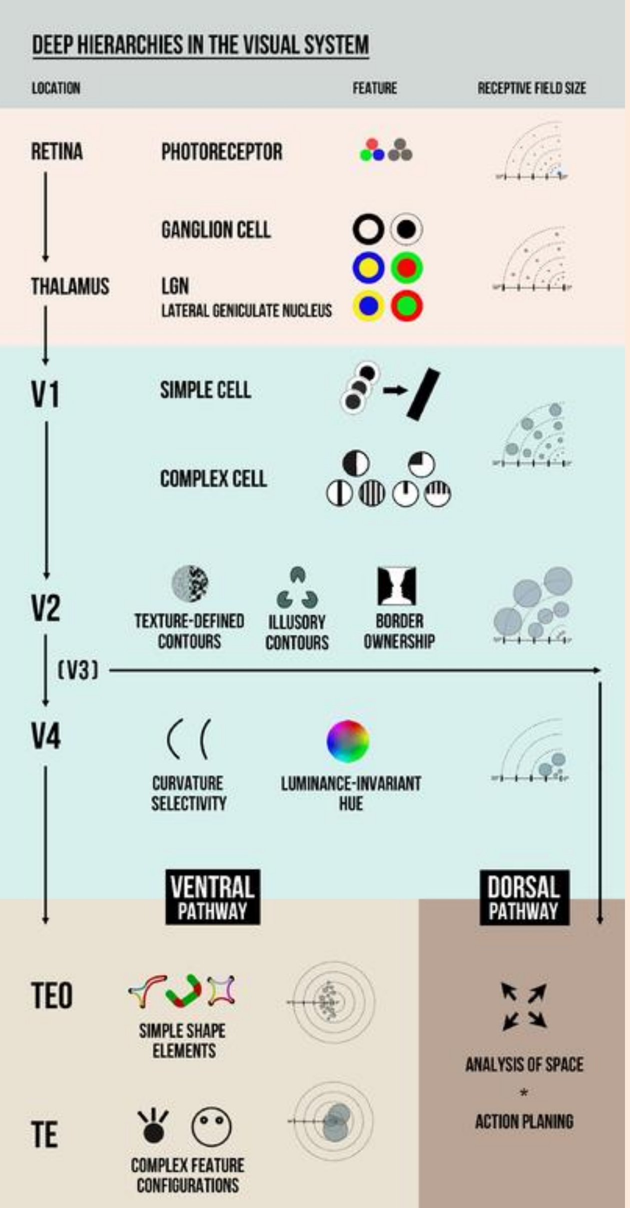
4	3
5	3

LeNet for Optical Character Recognition



Deep Hierarchies in the Visual System





DEEP HIERARCHIES IN THE VISUAL SYSTEM

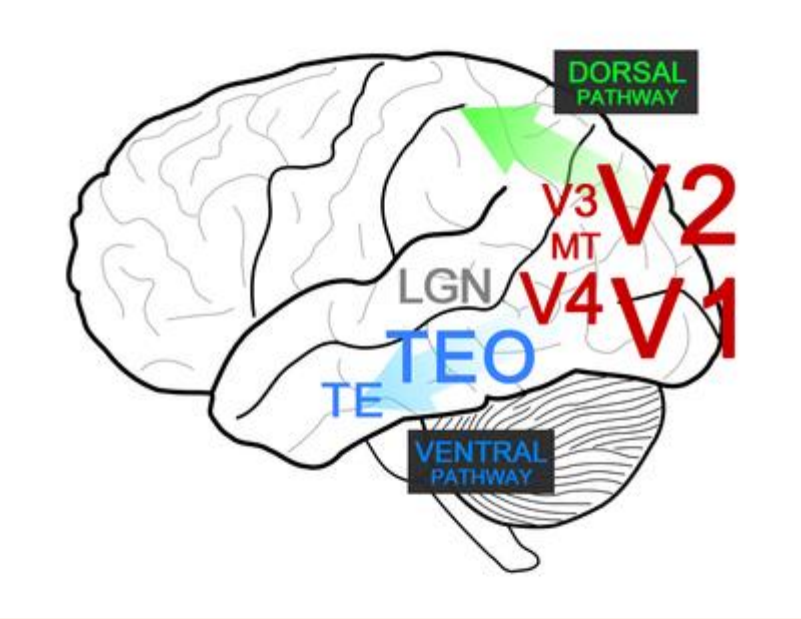
LOCATION		FEATURE	RECEPTIVE FIELD SIZE
RETINA	PHOTORECEPTOR		
	GANGLION CELL		
THALAMUS	LGN LATERAL GENICULATE NUCLEUS		
V1	SIMPLE CELL		
	COMPLEX CELL		
V2	TEXTURE-DEFINED CONTOURS ILLUSORY CONTOURS BORDER OWNERSHIP		
(V3)			
V4	CURVATURE SELECTIVITY LUMINANCE-INVARIANT HUE		

VENTRAL PATHWAY

TEO	SIMPLE SHAPE ELEMENTS	
TE	COMPLEX FEATURE CONFIGURATIONS	

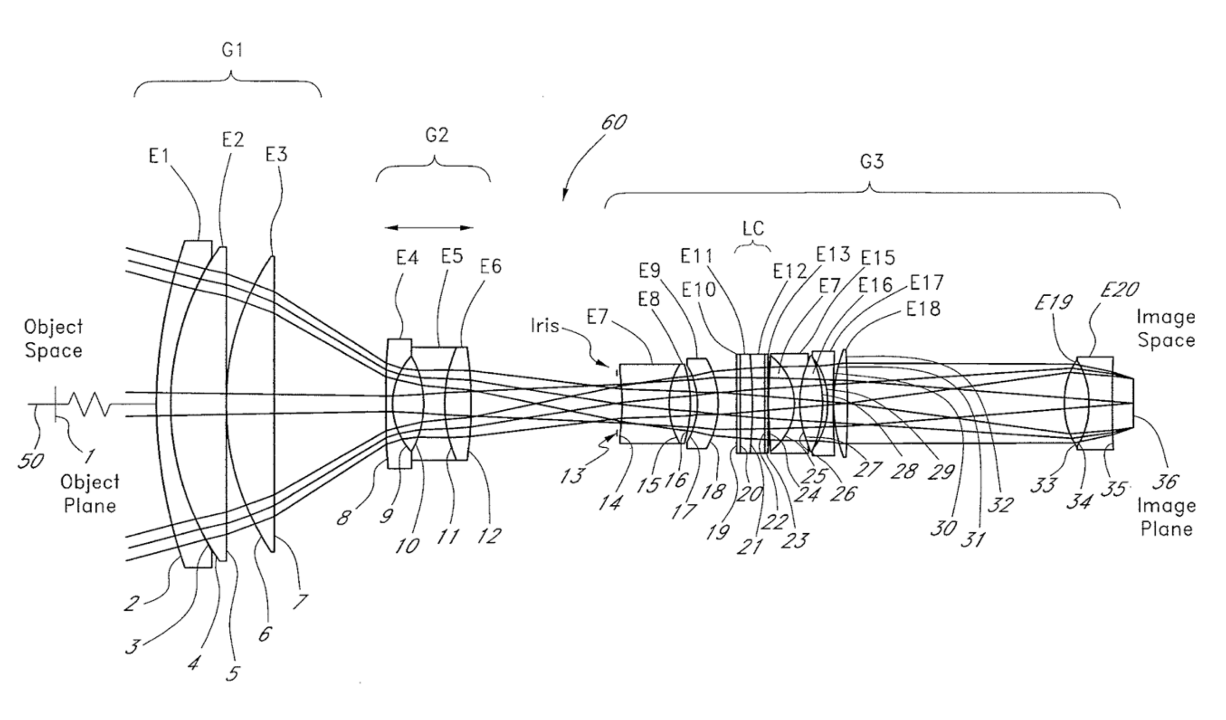
DORSAL PATHWAY

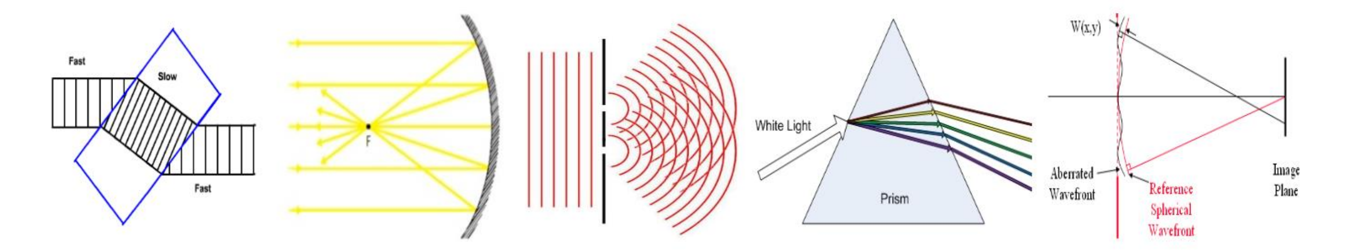
ANALYSIS OF SPACE
*
ACTION PLANING



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

Deep Hierarchies in Optics

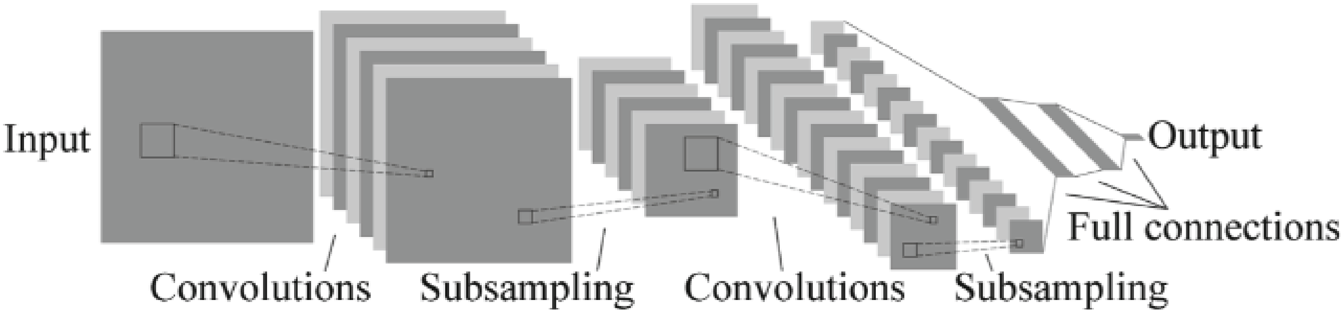




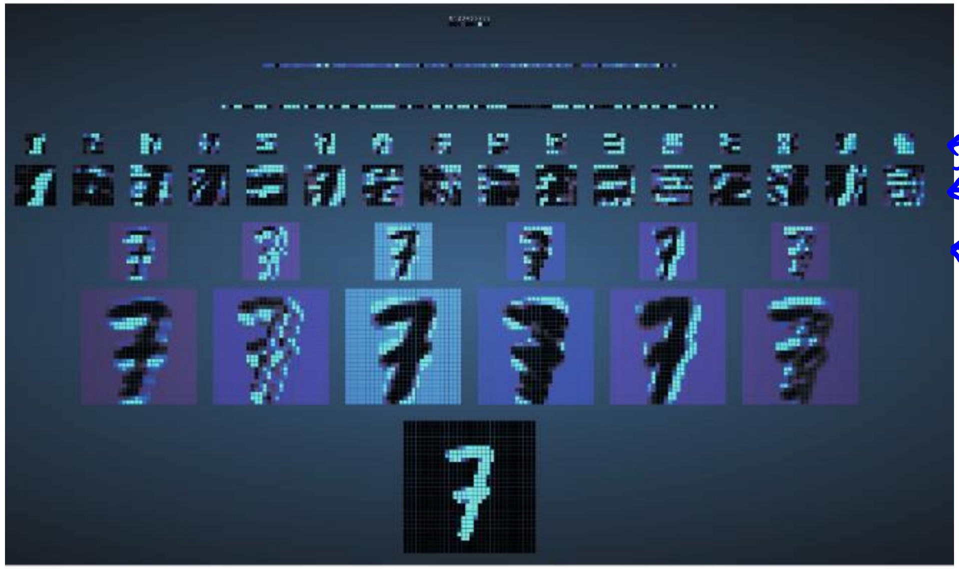
http://www.argmin.net/2018/01/25/optics/

Last Time: Convolutional Neural Networks

- Classic convolutional neural network (LeNet):



- Visualizing the "activations" of the layers:
  - http://scs.ryerson.ca/~aharley/vis/conv
  - http://cs231n.stanford.edu



softmax

2 "fully-connected"

max pooling

3D convolutions

max pooling

2D convolutions

http://scs.ryerson.ca/~aharley/vis/harley_vis_isvc15.pdf

Partial Summary

- ResNets include untransformed previous layers.
  - Network focuses non-linearity on residual, allows huge number of layers.
- Regularization is crucial to neural net performance:
  - L2-regularization, early stopping, dropout, implicit regularization of SGD.
- Convolutional neural networks:
  - Restrict $W^{(m)}$ matrices to represent sets of convolutions.
  - Often combined with max (pooling).

- Next time: modern convolutional neural networks and applications.
  - Image segmentation, depth estimation, image colorization, artistic style.

(End of testable content for final exam)

Topics from Previous Years

- Slides for other topics that were covered in previous years:
  - Ranking: finding “highest ranked” training examples (Google PageRank).
  - Semi-supervised: using unlabeled data to help supervised learning.
  - Sequence mining: approximate matching of patterns in large sequences.

- In previous years we did a course review on the last day:
  - Overview of topics covered in 340, and topics coming in 540.
  - Slides here: this could help with studying for the final.

CPSC 330 vs. 340

- CPSC 330 starts next semester: “Applied Machine Learning”.
  - Not intended as a sequel to 340 (or even a prequel).

- There is some overlap in content, but focus is different:
  - More emphasis on the other steps of the data processing pipeline:
    - Data cleaning, feature extraction, reproducible workflows, communicating results.
  - More emphasis of “how to use packages”, less on “how stuff works”.

- If you found 340 too hard to keep up with, 330 might make sense.
  - In this situation, tell your friends about 330.

CPSC 330 vs. 540

- Next semester I'm teaching CPSC 540.
  - Intended as a direct sequel to 340.
  - We're basically starting with CNNs and going from there.

- Main focuses:
  - What if y_i is a sentence or an image or a protein?
  - Giving you the background to understand the latest advances.

- Prerequisites:
  - I expect you to know everything in this course and CPSC 320.

- Longer term, I expect this course to also be listed as CPSC 440.
  - 540 next semester is a "trial run" for CPSC 440.
  - I removed topics related to optimization research from the course.

Other ML-Related Courses

- CPSC 406:
  - Numerical optimization algorithms (like gradient descent).
- CPSC 422:
  - Includes topics like time series and reinforcement learning.
- CPSC 532R/533R:
  - Deep learning for vision, sound, and language.
- CPSC 533V:
  - Deep learning for computer graphics.
- EECE 592:
  - Deep learning and reinforcement learning.
- STAT 406:
  - Similar/complementary topics.
- STAT 460/461:
  - Advanced statistical issues (what happens when ‘n’ goes to ∞?)

Final Slide

- "Calling Bullshit in the Age of Big Data":
  - https://www.youtube.com/playlist?list=PLPnZfvKID1Sje5jWxt-4CSZD7bUI4gSPS
  - Every "data scientist" should watch all these lectures.
  - You should be able to recognize non-sense,
    and not accidently produce non-sense!

- Thank you for your patience.
  - This was the first time someone has taught multiple sections of this class.

- Good luck with finals/projects and the next steps!