CPSC 340:
Machine Learning and Data Mining

Convolutional Neural Networks

Summer 2021

In This Lecture

1. Regularizing neural networks
2. Convolutional Neural Networks
3. Course Summary

Coming Up Next

MORE NEURAL NET ARCHITECTURES

"Residual" Networks (ResNets)

- Impactful recent idea is residual networks (ResNets):

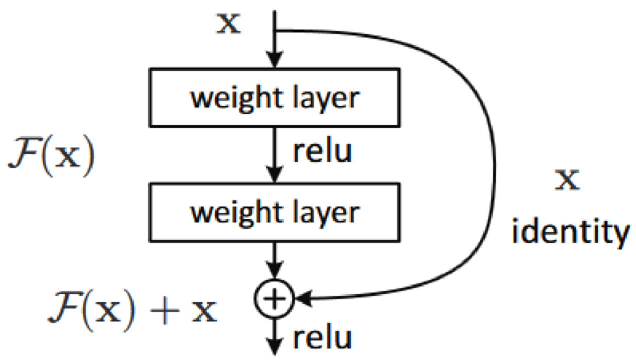


Figure 2. Residual learning: a building block.

$$x_i \rightarrow \underbrace{h(Wx_i)}_{z_i} + x_i$$

  - 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.
- Thanks to differentiable programming, this is surprisingly easy to do

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.

- Thanks to differentiable programming, this is surprisingly easy to do

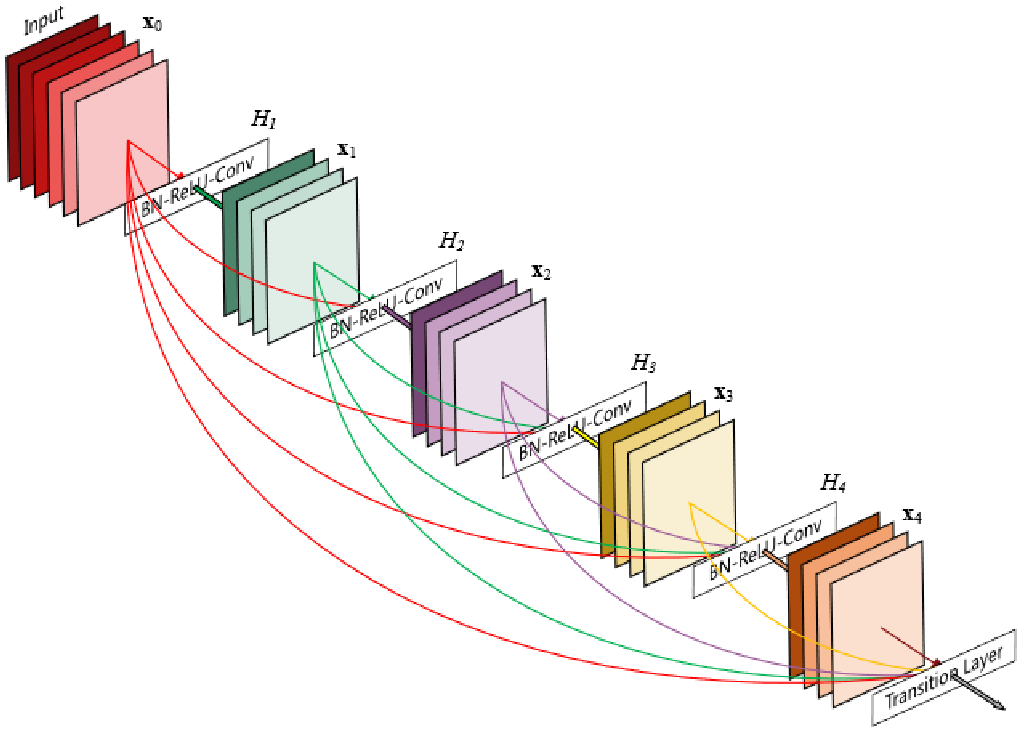


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

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.

Coming Up Next

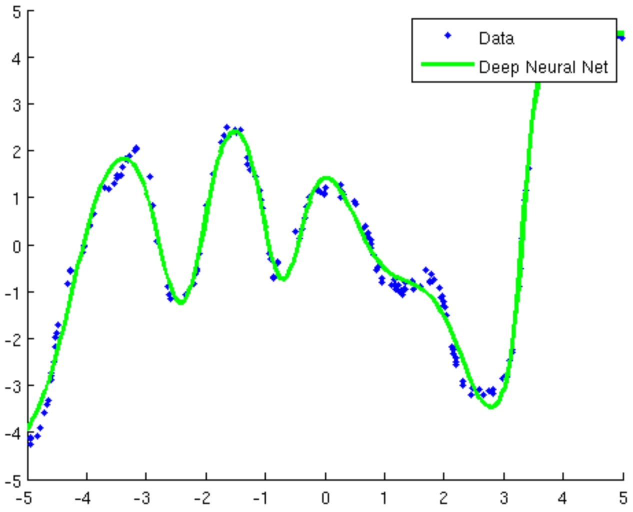
REGULARIZING NEURAL NETS

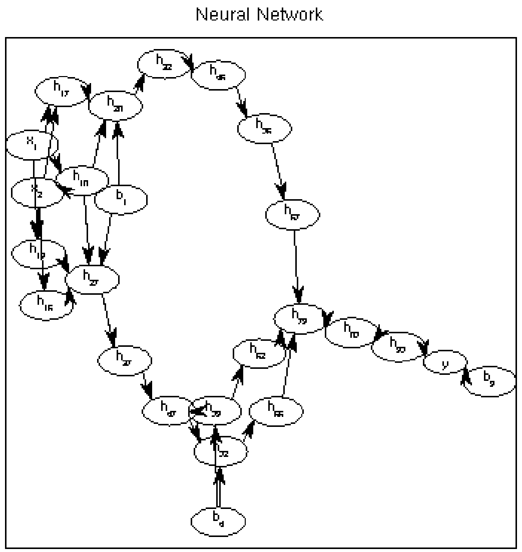
Standard Regularization

- 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.





Standard Regularization

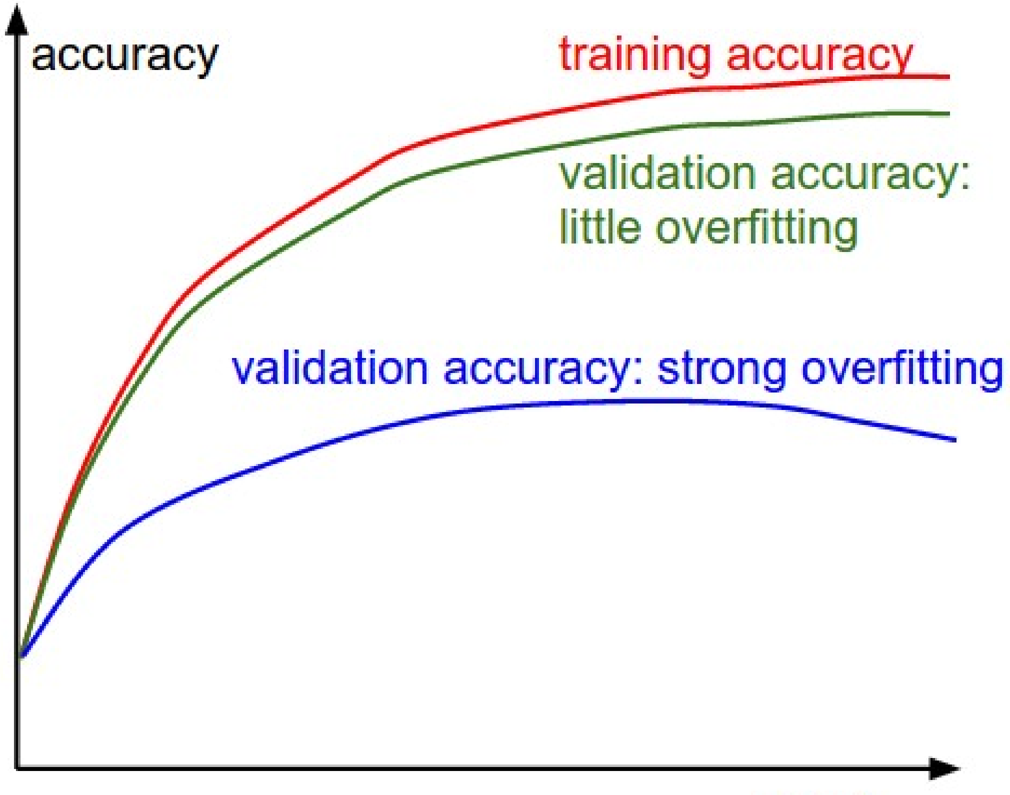
- Traditionally, we've added our usual L2-regularizers:

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

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



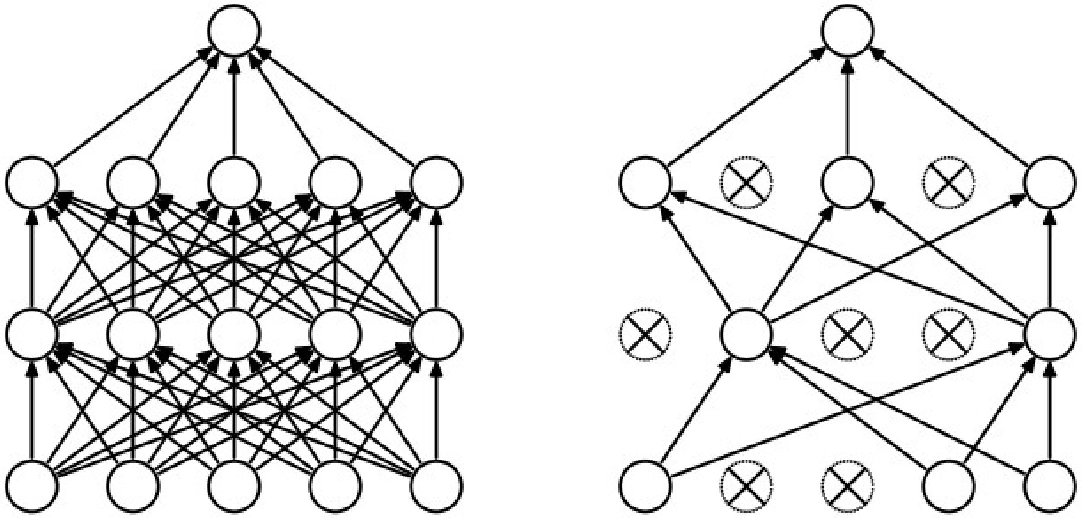
http://cs231n.github.io/neural-networks-3/

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%).



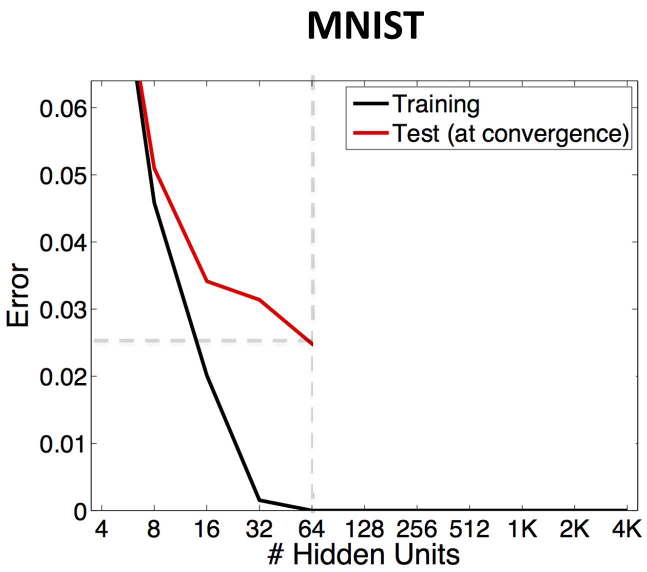
(a) Standard Neural Net (b) After applying dropout.

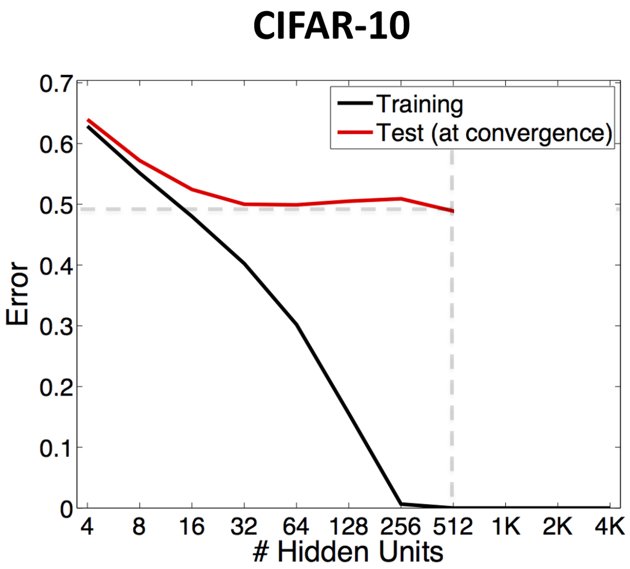
  - Adds invariance to missing inputs or latent factors
    - Encourages distributed representation rather than relying on specific z_i.
  - Can be interpreted as an ensemble over networks with different parts missing.
  - After a lot of success, dropout may already be going out of fashion.

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.

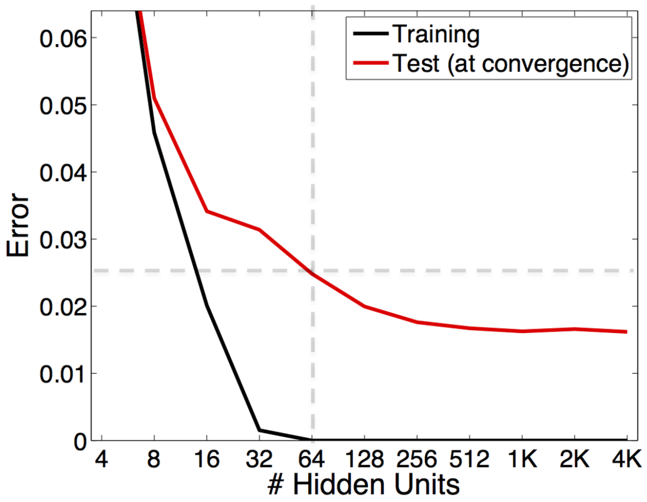
- What should happen to training and test error for larger #hidden?

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

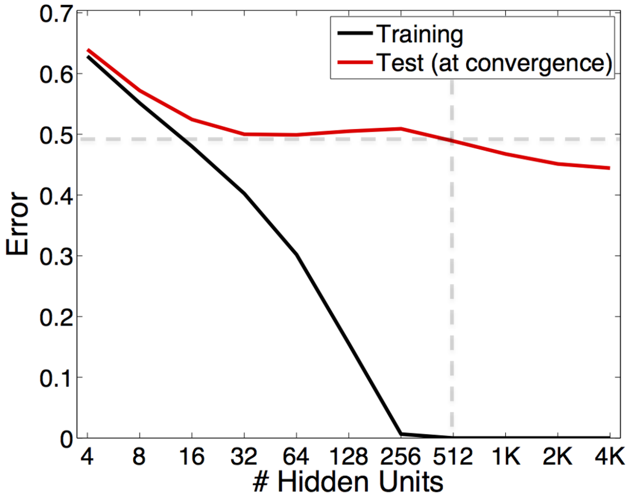
"Hidden" Regularization in Neural Networks

- 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

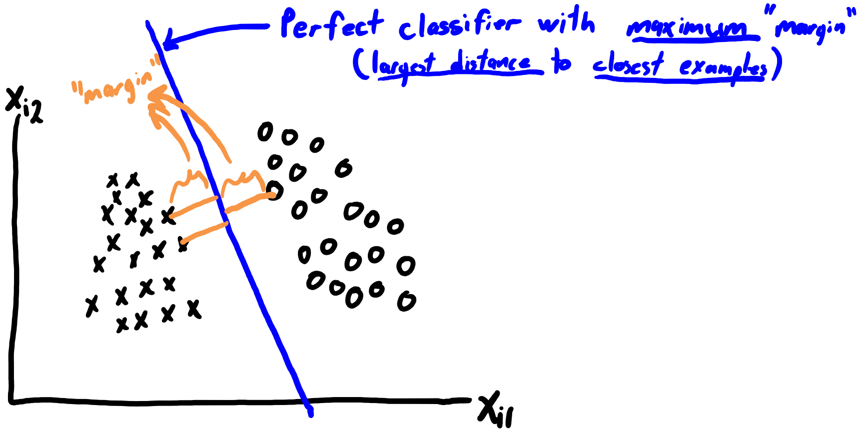
- There is growing evidence that using SGD regularizes parameters.
  - We call this the "implicit regularization" of the optimization algorithm.

- 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.



Coming Up Next

CONVOLUTIONAL NEURAL NETS

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.
  - ResNets and alternative non-linear functions like ReLU.
  - Different forms of regularization:
    - L2-regularization, early stopping, dropout, implicit regularization from SGD.

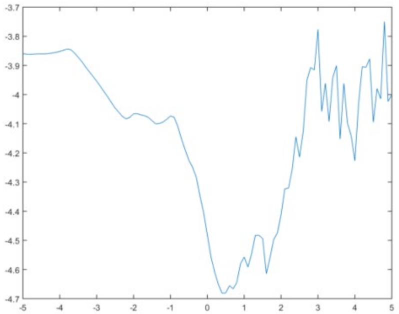
- These are often still not enough to get deep models working.

- 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).

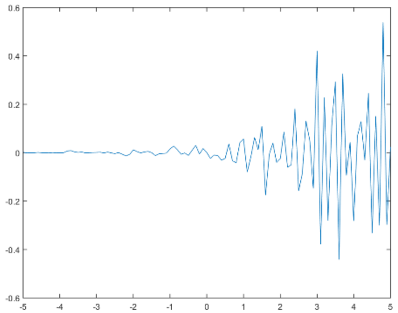
1D Convolution as Matrix Multiplication

- 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:

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

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

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

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

$$z = \tilde{W}x \quad \text{where } \tilde{W} = \begin{bmatrix} \text{———} & w & \text{———} & 0 & 0 & 0 \\ 0 & \text{———} & w & \text{———} & 0 & 0 \\ 0 & 0 & \text{———} & w & \text{———} & 0 \\ 0 & 0 & 0 & \text{———} & w & \text{———} \end{bmatrix}$$

matrix can be very sparse and only has 2m+1 variables.

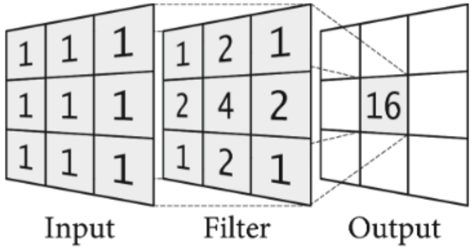
- The shorter 'w' is, the more sparse the matrix is.
- Thanks to differentiable programming, this is surprisingly easy to do

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’:

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

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.

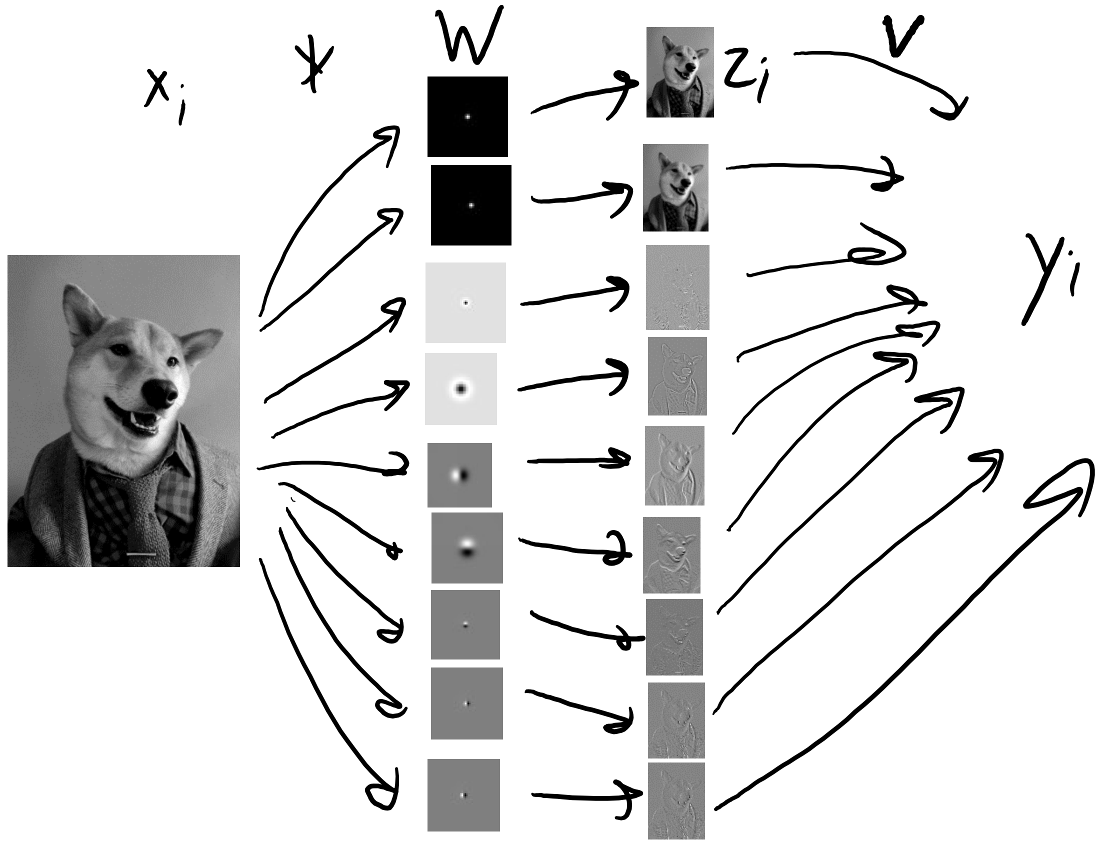
- Forces most weights to be zero, reduces number of parameters.

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

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

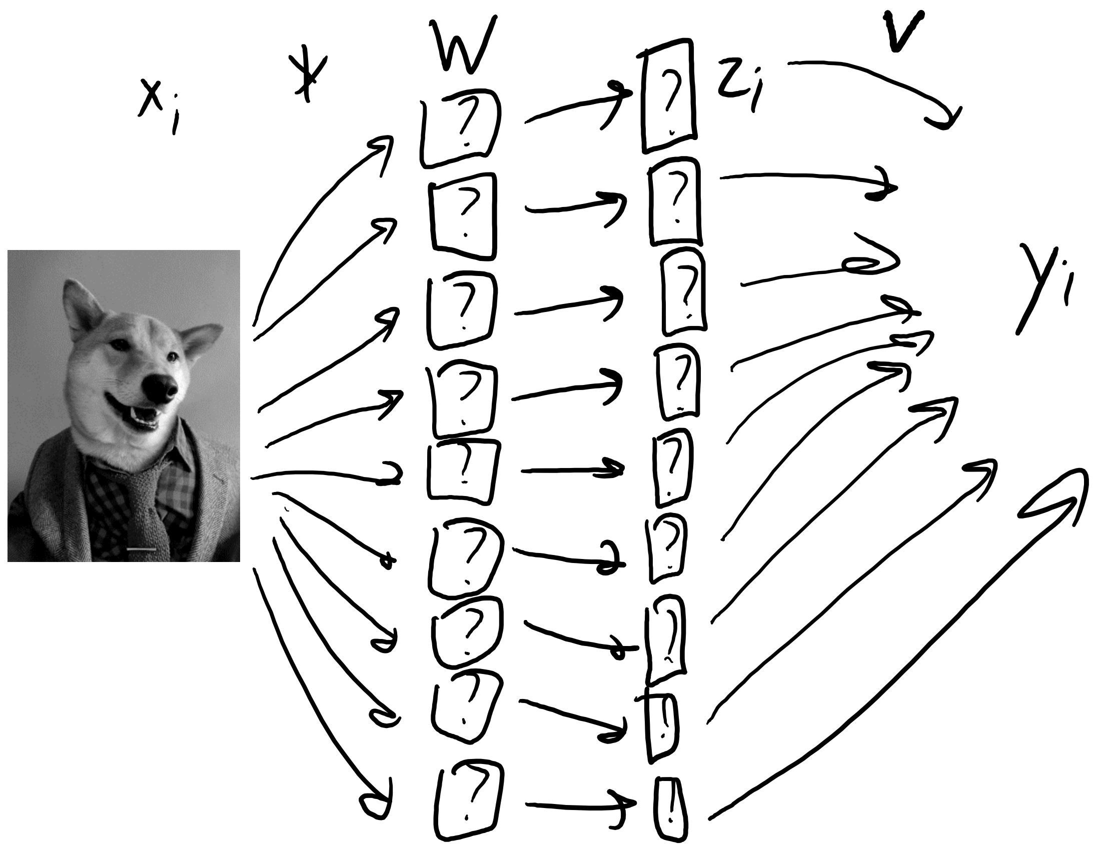
Motivation for Convolutional Neural Networks

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



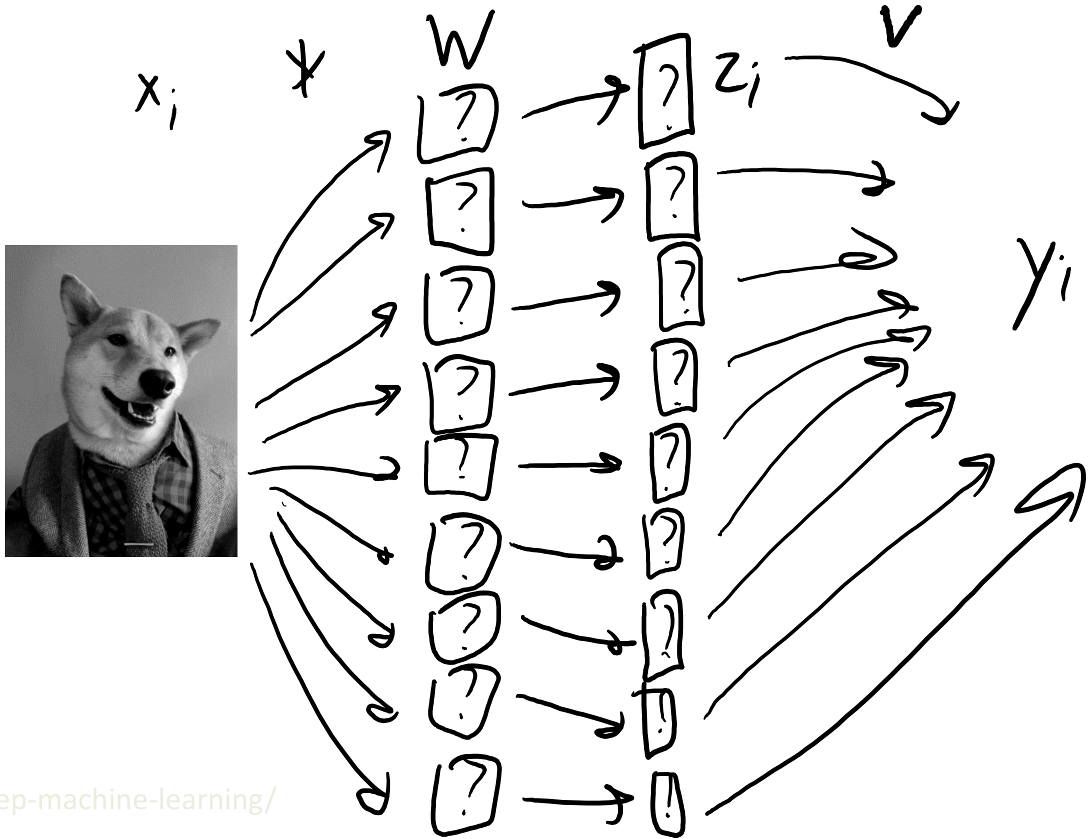
Motivation for Convolutional Neural Networks

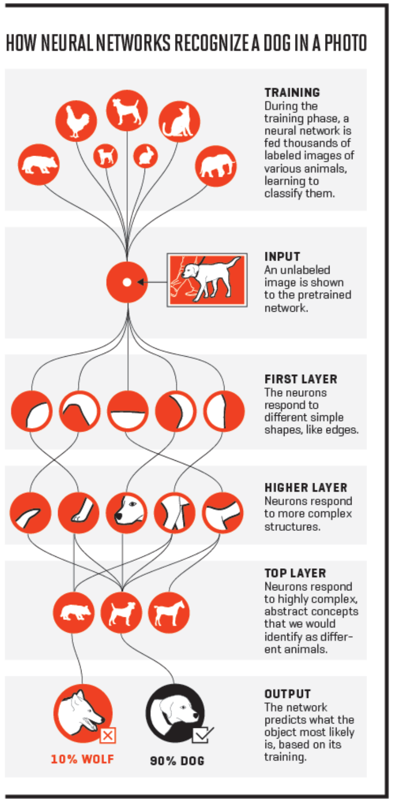
- Convolutional neural networks learn the convolutions:
  - Learning 'W' and 'v' automatically chooses types/variances/orientations.
  - Don't pick from fixed convolutions, but learn the elements of the filters.



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/

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

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

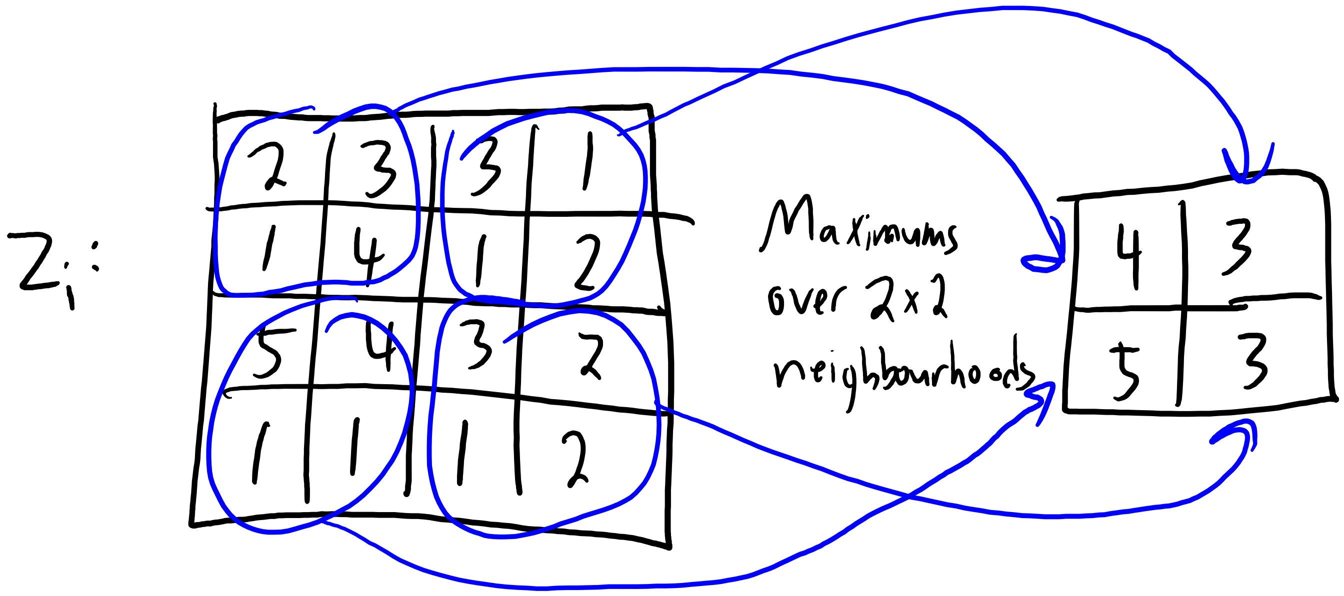
distance between centers of convolution is called "stride"

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

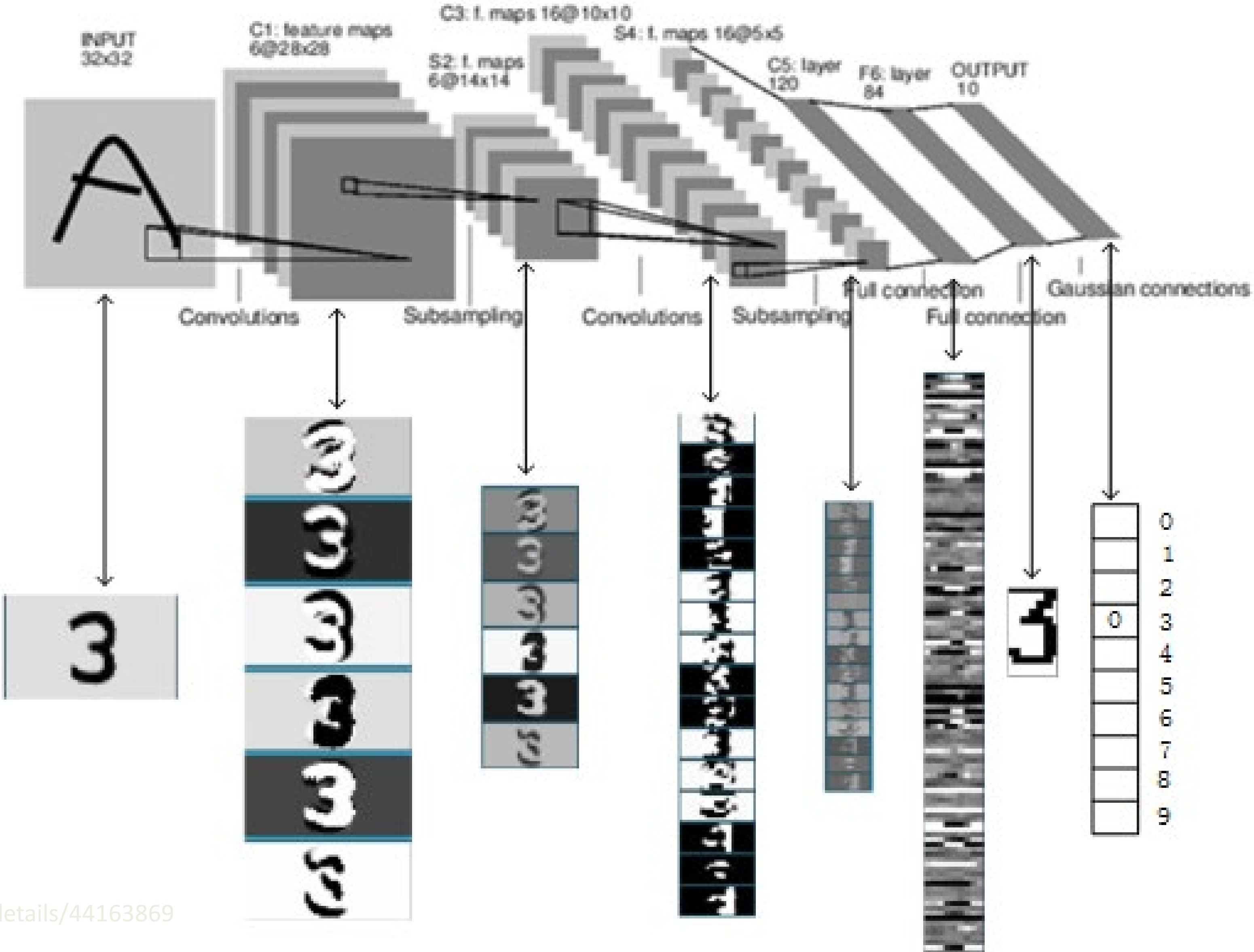
Sparse and small number of parameters

Convolutional Neural Networks

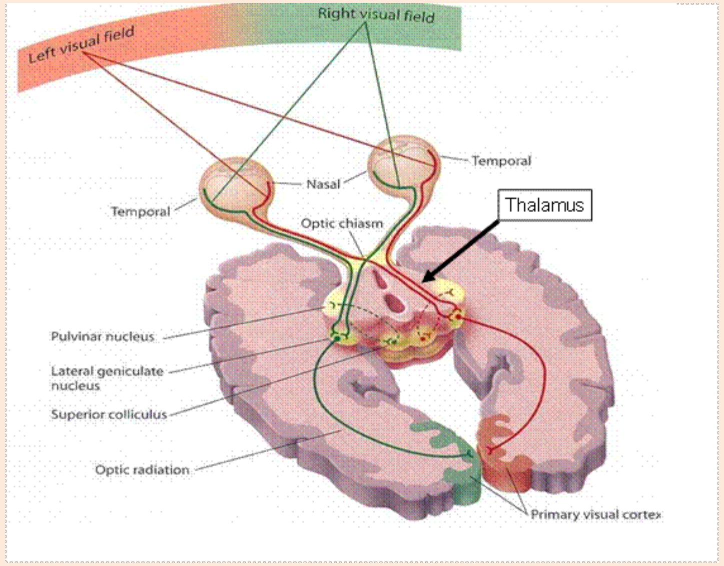
- Convolutional Neural Networks classically have 3 layer "types":
  - Fully connected layer: usual neural network layer with unrestricted W.
  - Convolutional layer: restrict W to act like several convolutions.
  - Pooling layer: combine results of convolutions.
    - Can add some invariance or just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

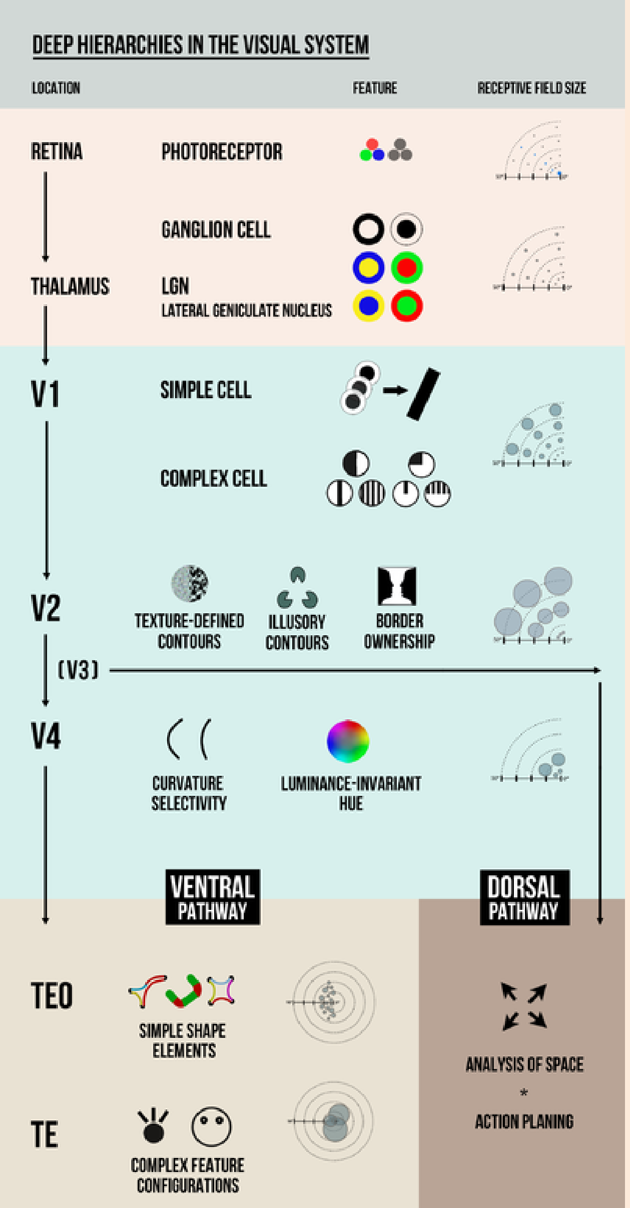


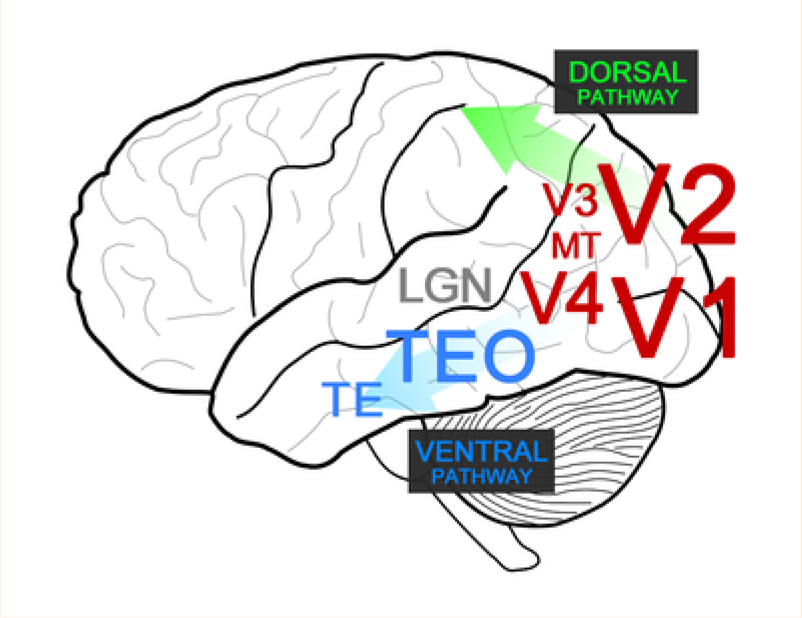
LeNet for Optical Character Recognition



Deep Hierarchies in the Visual System

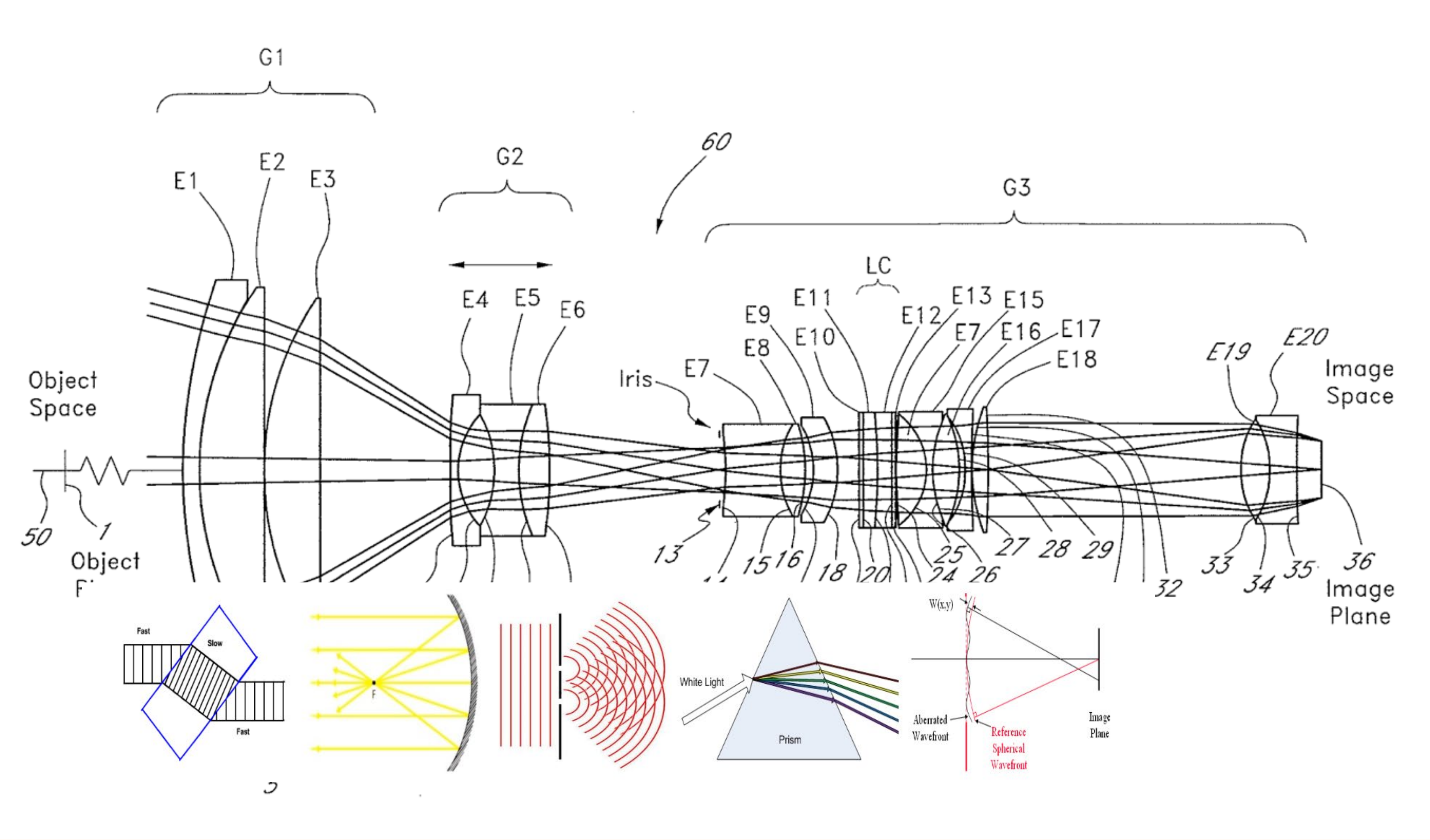






http://www.strokenetwork.org/newsletter/articles/vision.htm
https://en.wikibooks.org/wiki/Sensory_Systems/Visual_Signal_Processing

Deep Hierarchies in Optics



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

Coming Up Next

WORDS OF CAUTION

Mission Accomplished?

- For speech recognition and object detection:
  - No other methods have ever given the current level of performance.
  - Deep models continue to improve performance on these and related tasks.
  - We don't know how to scale up other universal approximators.
  - There is likely some overfitting to popular datasets like ImageNet.
    - Recent work showed accuracy drop of 4-10% by using a different test set on CIFAR 10.

- CNNs are now making their way into products.
  - Face recognition.
  - Amazon Go: https://www.youtube.com/watch?v=NrmMk1Myrxc
    - Trolling by French company Monoprix here.
  - Self-driving cars.

Mission Accomplished?

- We're still missing a lot of theory and understanding deep learning.

From: Boris
To: Ali

On Friday, someone on another team changed the default rounding mode of some Tensorflow internals (from truncation to "round to even").*

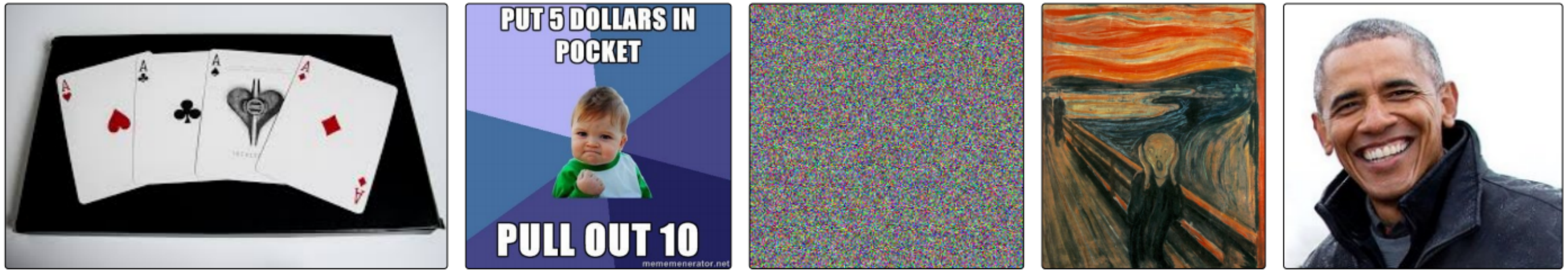
*Our training broke. Our error rate went from <25% error to ~99.97% error (on a standard 0-1 binary loss).

- "[Good CS expert says: Most firms that thinks they want advanced AI/ML really just need linear regression on cleaned-up data]."

http://www.argmin.net/2017/12/05/kitchen-sinks/

Mission Accomplished?

- Despite high-level of abstraction, deep CNNs are easily fooled:
  - Hot research topic at the moment.

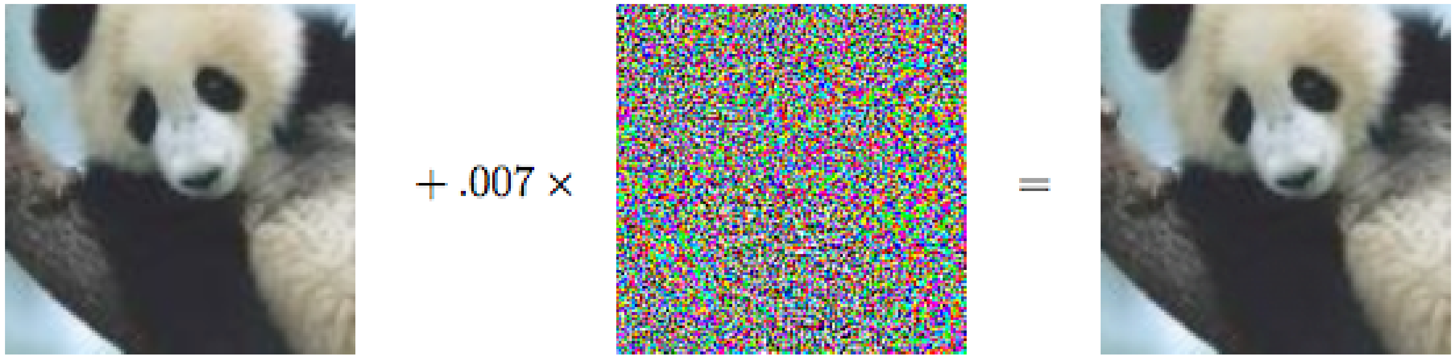


DenseNet 161 (2017)	Envelope 31%	Balance Beam 52%	Chainlink Fence 31%	Chest 37%	Tench 36%
SqueezeNet (2016)	Binder 43%	Balance Beam 18%	Poncho 32%	Jean 30%	Suit 21%
ResNet 152 (2015)	Envelope 40%	Pacifier 33%	Chain Mail 29%	Dust Cover 52%	Sweatshirt 25%
VGG 19 (2014)	Binder 51%	Dust Cover 44%	Window Screen 5%	Chest 11%	Sweatshirt 46%
AlexNet (2012)	T-shirt 16%	Dust Cover 22%	Cardigan 12%	Theater Curtain 3%	Coho 37%

Figure 1: The arbitrary predictions of several popular networks [2, 3, 4, 5, 6] that are trained on ImageNet [1] on unseen data. The red predictions are entirely wrong, the green predictions are justifiable, the orange predictions are less justifiable. The middle image is noise sampled from $\mathcal{N}(\mu = 0.5, \sigma = 0.25)$ without any modifications. This unpredictable behaviour is not limited to demonstrated architectures. We show that merely thresholding the output probability is not a reliable method to detect these problematic instances.

Mission Accomplished?

- Despite high-level of abstraction, deep CNNs are easily fooled:
  - Hot research topic at the moment.
- Recent work: imperceptible noise that changes the predicted label.
  - “Adversarial” examples (can change to any other label).



$\boldsymbol{x}$	$\mathrm{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$	$\boldsymbol{x} + \epsilon \mathrm{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
“panda” 57.7% confidence	“nematode” 8.2% confidence	“gibbon” 99.3 % confidence

Mission Accomplished?

- Can someone repaint a stop sign and fool self-driving cars?

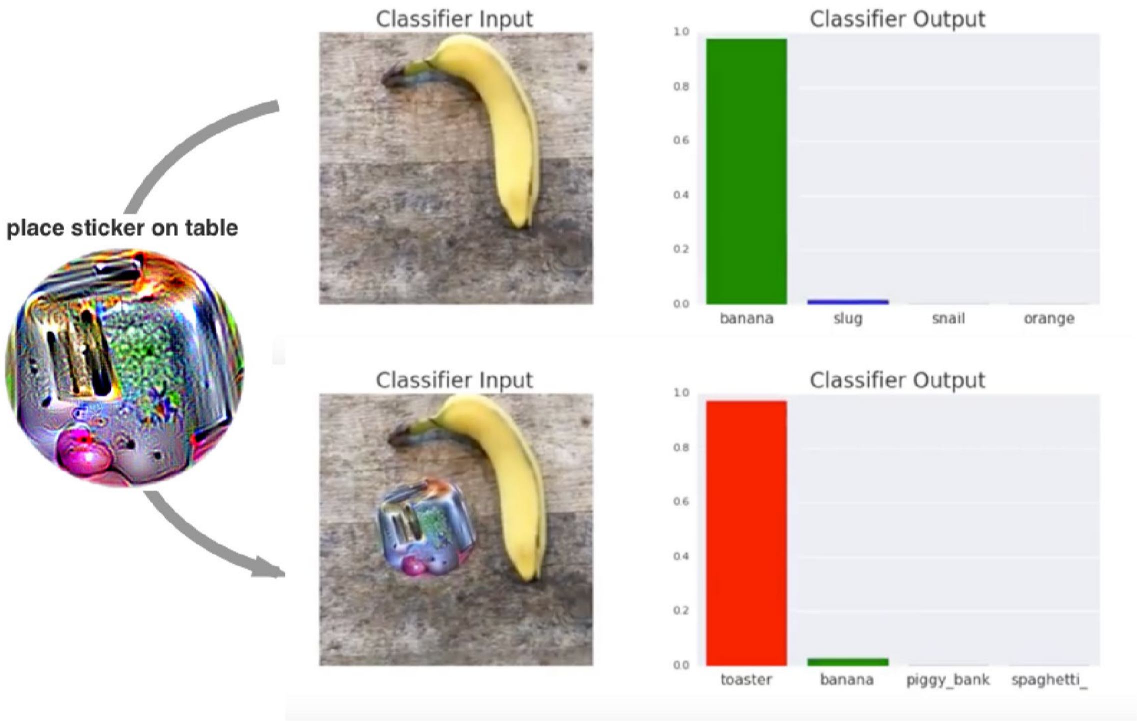


Figure 1: A real-world attack on VGG16, using a physical patch generated by the white-box ensemble method described in Section 3. When a photo of a tabletop with a banana and a notebook (top photograph) is passed through VGG16, the network reports class 'banana' with 97% confidence (top plot). If we physically place a sticker targeted to the class "toaster" on the table (bottom photograph), the photograph is classified as a toaster with 99% confidence (bottom plot). See the following video for a full demonstration: `https://youtu.be/i1sp4X57TL4`

https://arxiv.org/pdf/1712.09665.pdf

Mission Accomplished?

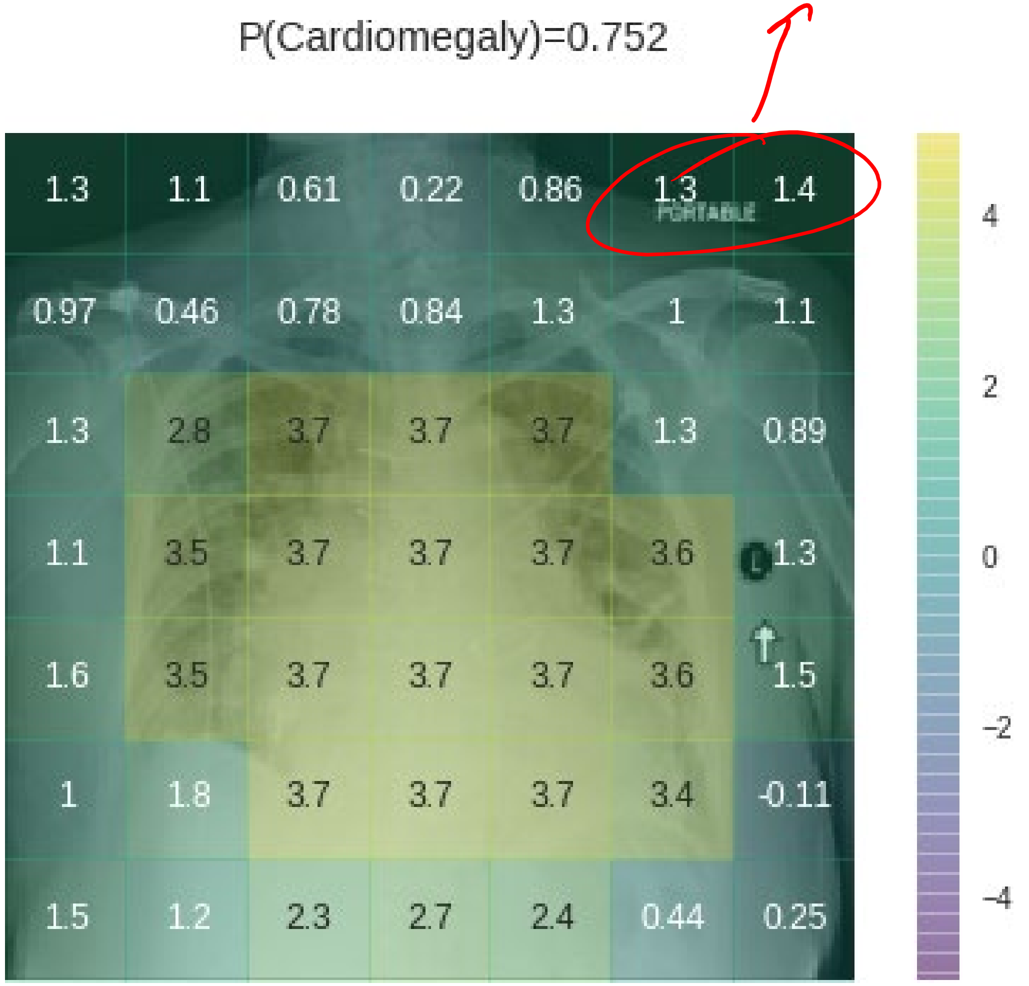
- Are the networks understanding the fundamental concepts?
  - Is being "surrounded by green" part of the definition of cow?
  - Do we need to have examples of cows in different environments?
    - Kids don't need this.

https://www.onegreenplanet.org/news/cows-enjoy-the-beach/

Mission Accomplished?

- CNNs may not be learning what you think they are.
  - CNN for diagnosing enlarged heart:
    - Higher values mean more likely to be enlarged:
  - CNN says “portable” protocol is predictive:
    - But they are probably getting a “portable” scan because they’re too sick to go the hospital.
  - CNN was biased by the scanning protocol.
    - Learns the scans that more-sick patients get.
    - This is not what we want in a medical test.

????



https://medium.com/@jrzech/what-are-radiological-deep-learning-models-actually-learning-f97a546c5b98

(Racially-)Biased Algorithms?

- Major issue: are we learning representations with harmful biases?
  - Biases could come from data (if data only has certain groups in certain situations).
  - Biases could come from labels (always using label of "ball" for certain sports).
  - Biases could come from learning method (model predicts "basketball" for black people more often than they appear in training data for basketball images).

Fig. 8: Pairs of pictures (columns) sampled over the Internet along with their prediction by a ResNet-101.

  - This is a major problem/issue when deploying these systems.
    - E.g., "repeat-offender prediction" that reinforces racial biases in arrest patterns.

https://arxiv.org/pdf/1711.11443.pdf

Energy Costs

- Current methods require:
  - A lot of data.
  - A lot of time to train.
  - Many training runs to do hyper-parameter optimization.

- Recent paper regarding recent deep language models:
  - Entire training procedure emits 5 times more CO_2 than lifetime emission of a car, including making the car.

Machine Learning: "Soft Query"

- Suppose I have a dictionary (mapping of key → value)

Key	1	3	5	7	9
Value	5	15	25	35	45

Q: What do I get if I ask for dict[2]?

```
>>> d = dict()
>>> d[1] = 5
>>> d[3] = 15
>>> d[5] = 25
>>> d[7] = 35
>>> d[9] = 45
>>> d[1]
5
>>> d[2]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 2
```

Machine Learning: "Soft Query"

- Suppose I have a 1-d linear regression model

x_i	1	3	5	7	9
y_i	5	15	25	35	45

Q: What do I get if I ask for model.predict([2])?

- Instead of KeyError, we will get a number!

$$y_i = w x_i \,. \quad w = 5$$

$$\hat{y}_i = 5 \cdot 2 = 10$$

- Learned model: a more robust version of a dictionary.

Machine Learning: "Soft Query"

- Suppose I have a 1-d linear regression model

x_i	1	3	5	7	9
y_i	5	15	25	35	45

Q: What do I get if I ask for model.predict([11])?

$$\hat{y}_i = 5 \cdot 11 = 55$$

x_i	1	3	5	7	9	11	13
y_i	5	15	25	35	45	-20	-30

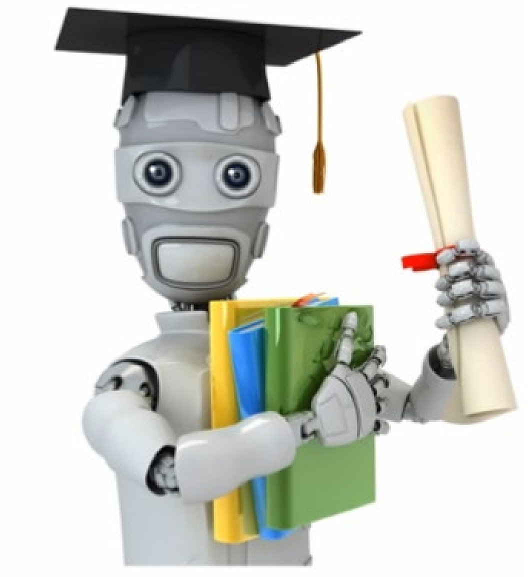
Bottom line 1:
ML is not a solved problem

Bottom line 2:
Data is the real bottleneck of ML

Coming Up Next

CONCLUSION:
WHAT IS MACHINE LEARNING?

What is Machine Learning?



- Machine learning: a special type of optimization
  - Extract statistical regularities from data
  - Translate regularities into numerical structures
- We studied mechanisms enabling prediction, compression, encoding (and all at the same time)

CPSC 340: Overview

1. Intro to supervised learning (using counting and distances).
   - Training vs. testing, parametric vs. non-parametric, ensemble methods.
   - Fundamental trade-off, no free lunch, universal consistency.
2. Intro to unsupervised learning (using counting and distances).
   - Clustering, outlier detection, finding similar items.
3. Linear models and gradient descent (for supervised learning)
   - Loss functions, change of basis, regularization, feature selection.
   - Gradient descent and stochastic gradient.
4. Latent-factor models (for unsupervised learning)
   - Typically using linear models and gradient descent.
5. Neural networks (for supervised and multi-layer latent-factor models).

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.
  - Boosting: another ensemble of decision trees that works very well.

- 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.

Where to Go From Here

- 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:
  - Reinforcement learning for simulation and control
- 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.
  - I'm a first-time instructor!
  - We had 2 fewer lectures than usual term

- Good luck with finals and the next steps!

Please do course evaluation!