

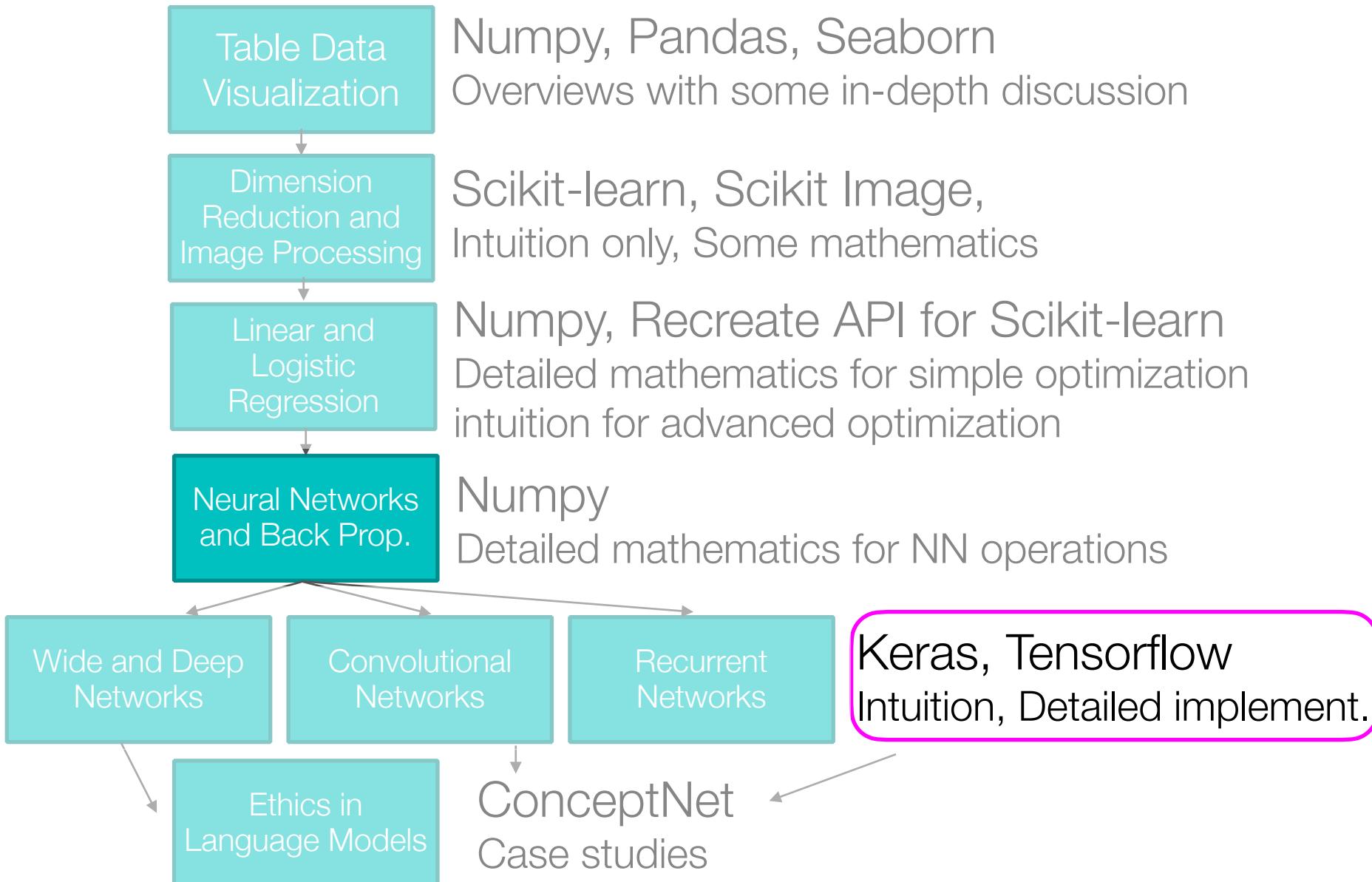
Lecture Notes for **Machine Learning in Python**

Revisiting CV A “Too Early” History of Deep Learning

Logistics and Agenda

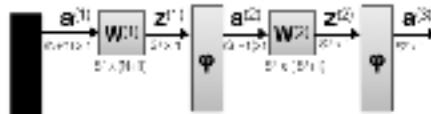
- Logistics
 - None!
- Agenda
 - Revisiting Cross Validation
 - Town Hall
 - “Deep Learning” History

Class Overview, by topic



Last time:

Back propagation summary



$$J(\mathbf{W}) = \sum_k^M (\mathbf{y}^{(k)} - \mathbf{a}^{(L)})^2$$

$$w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(l)}} a_j^{(l)}$$

1. Forward propagate to get \mathbf{z}, \mathbf{a} for all layers
2. Get final layer gradient
3. Update back propagation variables
4. Update each $\mathbf{W}^{(l)}$

for each $y^{(k)}$: $\frac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(L)}} = -2(\mathbf{y}^{(k)} - \mathbf{a}^{(L)}) \times \mathbf{a}^{(L)} + (1 - \mathbf{a}^{(L)})$

$\frac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(l)}} = \text{diag}[\mathbf{a}^{(l+1)} \times (1 - \mathbf{a}^{(l+1)})] \cdot \mathbf{W}^{(l+1)} \frac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(l+1)}}$

$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta \frac{\partial J(\mathbf{W}^{(l)})}{\partial \mathbf{z}^{(l)}} \cdot \mathbf{a}^{(l)}$

Practical Implementation of Architectures

- A new cost function: **Cross entropy**

$$J(\mathbf{W}) = -[\mathbf{y}^{(l)} \ln \mathbf{a}^{(L)} + (1 - \mathbf{y}^{(l)}) \ln(1 - \mathbf{a}^{(L)})]$$

speeds up initial training

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(l)}} = (\mathbf{a}^{(l+1)} - \mathbf{y}^{(l)})$$

vectorized backpropagation
signal1 = (A1-Y_enc) # <- this is only line
signal2 = (M2.T * signal1)*A2*(1-A2)

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(2)}} = (\mathbf{a}^{(3)} - \mathbf{y}^{(l)})$$

grad1 = signal2[1:, :] * A1
grad2 = signal3 * M2.T

new update

vectorized backpropagation
signal1 = -2*(Y_enc-A1)*A2*(1-A2)
signal2 = (M2.T * signal1)*A2*(1-A2)

grad1 = signal2[1:, :] * A1
grad2 = signal3 * M2.T

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(2)}} = -2(\mathbf{y}^{(l)} - \mathbf{a}^{(3)}) \times \mathbf{a}^{(3)} + (1 - \mathbf{a}^{(3)}) \text{ old update}$$

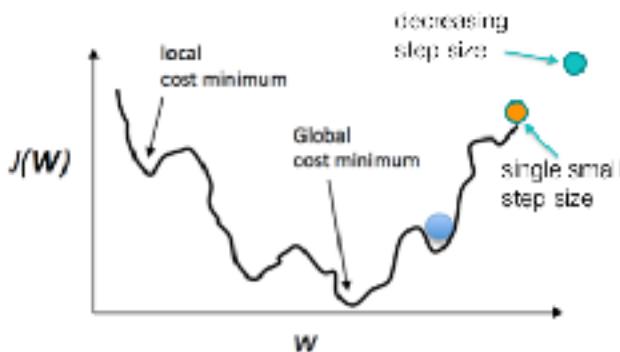
bp-5
10

Problems with Advanced Architectures

- Space is no longer convex

One solution:

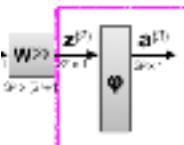
- start with large step size
- *cool down* by decreasing step size for higher iterations



8

Practical Implementation of Architectures

- A new nonlinearity: **rectified linear units**



$$\phi(\mathbf{z}^{(l)}) = \begin{cases} \mathbf{z}^{(l)}, & \text{if } \mathbf{z}^{(l)} > 0 \\ 0, & \text{else} \end{cases}$$

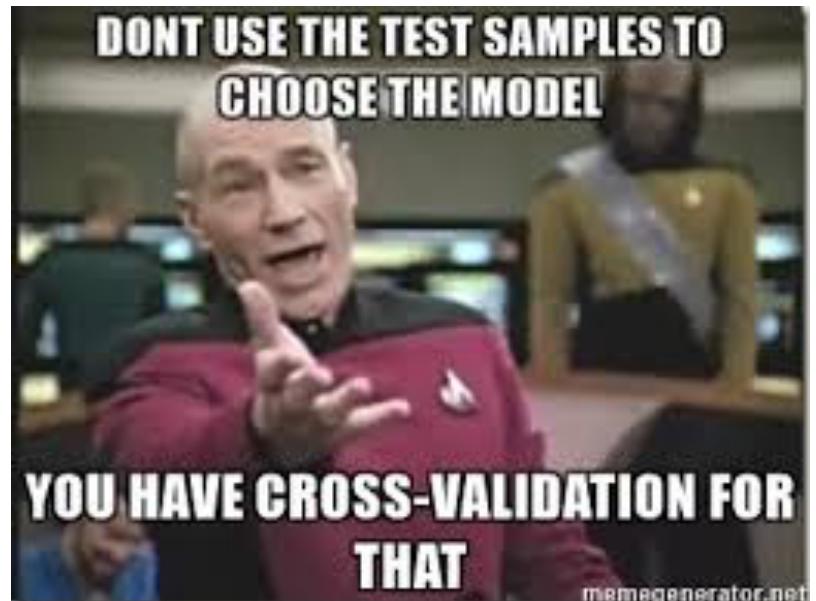
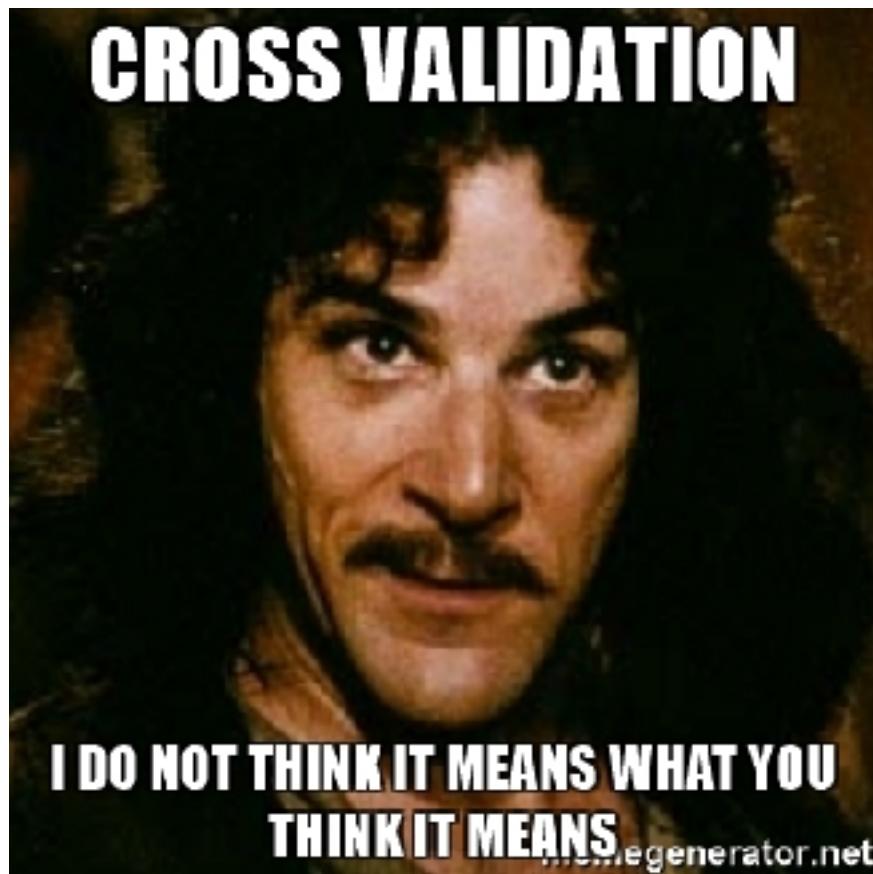
it has the advantage of **large gradients** and **extremely simple** derivative

$$\frac{\partial \phi(\mathbf{z}^{(l)})}{\partial \mathbf{z}^{(l)}} = \begin{cases} 1, & \text{if } \mathbf{z}^{(l)} > 0 \\ 0, & \text{else} \end{cases}$$

79

4

Revisiting Cross Validation



Review: Grid Searching

Trying to find the best parameters

LR: $C_1 = [1, 10, 100]$ $C_2 = [1e3, 1e4, 1e5]$

		C_1		
		(1, 1e3)	(10, 1e3)	(100, 1e3)
		(1, 1e4)	(10, 1e4)	(100, 1e4)
C_2	(1, 1e5)	(1, 1e5)	(10, 1e5)	(100, 1e5)

Review: Grid Searching

For each value, want to run cross validation...

C1																																																								
C2	(1, 1e3)	(10, 1e3)	(100, 1e3)																																																					
	<table border="1"><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>A</td><td>C</td><td>B</td></tr><tr><td>A</td><td>B</td><td>B</td></tr><tr><td>B</td><td>C</td><td>A</td></tr><tr><td>B</td><td>C</td><td>A</td></tr></table>	A	B	C	A	B	C	A	C	B	A	B	B	B	C	A	B	C	A	<table border="1"><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>A</td><td>C</td><td>B</td></tr><tr><td>A</td><td>C</td><td>B</td></tr><tr><td>B</td><td>C</td><td>A</td></tr><tr><td>B</td><td>C</td><td>A</td></tr></table>	A	B	C	A	B	C	A	C	B	A	C	B	B	C	A	B	C	A	<table border="1"><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>A</td><td>C</td><td>B</td></tr><tr><td>A</td><td>C</td><td>B</td></tr><tr><td>B</td><td>C</td><td>A</td></tr><tr><td>E</td><td>C</td><td>A</td></tr></table>	A	B	C	A	B	C	A	C	B	A	C	B	B	C	A	E	C
A	B	C																																																						
A	B	C																																																						
A	C	B																																																						
A	B	B																																																						
B	C	A																																																						
B	C	A																																																						
A	B	C																																																						
A	B	C																																																						
A	C	B																																																						
A	C	B																																																						
B	C	A																																																						
B	C	A																																																						
A	B	C																																																						
A	B	C																																																						
A	C	B																																																						
A	C	B																																																						
B	C	A																																																						
E	C	A																																																						
(1, 1e4)	(10, 1e4)	(100, 1e4)																																																						
<table border="1"><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>A</td><td>C</td><td>B</td></tr><tr><td>A</td><td>C</td><td>B</td></tr><tr><td>B</td><td>C</td><td>A</td></tr><tr><td>B</td><td>C</td><td>A</td></tr></table>	A	B	C	A	B	C	A	C	B	A	C	B	B	C	A	B	C	A	<table border="1"><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>A</td><td>C</td><td>B</td></tr><tr><td>A</td><td>C</td><td>B</td></tr><tr><td>B</td><td>C</td><td>A</td></tr><tr><td>B</td><td>C</td><td>A</td></tr></table>	A	B	C	A	B	C	A	C	B	A	C	B	B	C	A	B	C	A	<table border="1"><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>A</td><td>C</td><td>B</td></tr><tr><td>A</td><td>C</td><td>B</td></tr><tr><td>B</td><td>C</td><td>A</td></tr><tr><td>E</td><td>C</td><td>A</td></tr></table>	A	B	C	A	B	C	A	C	B	A	C	B	B	C	A	E	C	A
A	B	C																																																						
A	B	C																																																						
A	C	B																																																						
A	C	B																																																						
B	C	A																																																						
B	C	A																																																						
A	B	C																																																						
A	B	C																																																						
A	C	B																																																						
A	C	B																																																						
B	C	A																																																						
B	C	A																																																						
A	B	C																																																						
A	B	C																																																						
A	C	B																																																						
A	C	B																																																						
B	C	A																																																						
E	C	A																																																						
(1, 1e5)	(10, 1e5)	(100, 1e5)																																																						
<table border="1"><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>A</td><td>C</td><td>B</td></tr><tr><td>A</td><td>C</td><td>B</td></tr><tr><td>B</td><td>C</td><td>A</td></tr><tr><td>B</td><td>C</td><td>A</td></tr></table>	A	B	C	A	B	C	A	C	B	A	C	B	B	C	A	B	C	A	<table border="1"><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>A</td><td>C</td><td>B</td></tr><tr><td>A</td><td>C</td><td>B</td></tr><tr><td>B</td><td>C</td><td>A</td></tr><tr><td>B</td><td>C</td><td>A</td></tr></table>	A	B	C	A	B	C	A	C	B	A	C	B	B	C	A	B	C	A	<table border="1"><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>A</td><td>C</td><td>B</td></tr><tr><td>A</td><td>C</td><td>B</td></tr><tr><td>B</td><td>C</td><td>A</td></tr><tr><td>E</td><td>C</td><td>A</td></tr></table>	A	B	C	A	B	C	A	C	B	A	C	B	B	C	A	E	C	A
A	B	C																																																						
A	B	C																																																						
A	C	B																																																						
A	C	B																																																						
B	C	A																																																						
B	C	A																																																						
A	B	C																																																						
A	B	C																																																						
A	C	B																																																						
A	C	B																																																						
B	C	A																																																						
B	C	A																																																						
A	B	C																																																						
A	B	C																																																						
A	C	B																																																						
A	C	B																																																						
B	C	A																																																						
E	C	A																																																						

Review: Grid Searching

Could perform iteratively

C1

(1, 1e3)

A	B	C
A	B	C
A	C	B
A	B	B
B	C	A
B	C	A

(10, 1e3)

A	B	C
A	B	C
A	C	B
A	C	B
B	C	A
B	C	A

(100, 1e3)

A	B	C
A	B	C
A	C	B
A	C	B
B	C	A
B	C	A

C2

(1, 1e4)

A	B	C
A	B	C
A	C	B
A	C	B
B	C	A
B	C	A

(10, 1e4)

A	B	C
A	B	C
A	C	B
A	C	B
B	C	A
B	C	A

(100, 1e4)

A	B	C
A	B	C
A	C	B
A	C	B
B	C	A
B	C	A

(1, 1e5)

A	B	C
A	B	C
A	C	B
A	C	B
B	C	A
B	C	A

(10, 1e5)

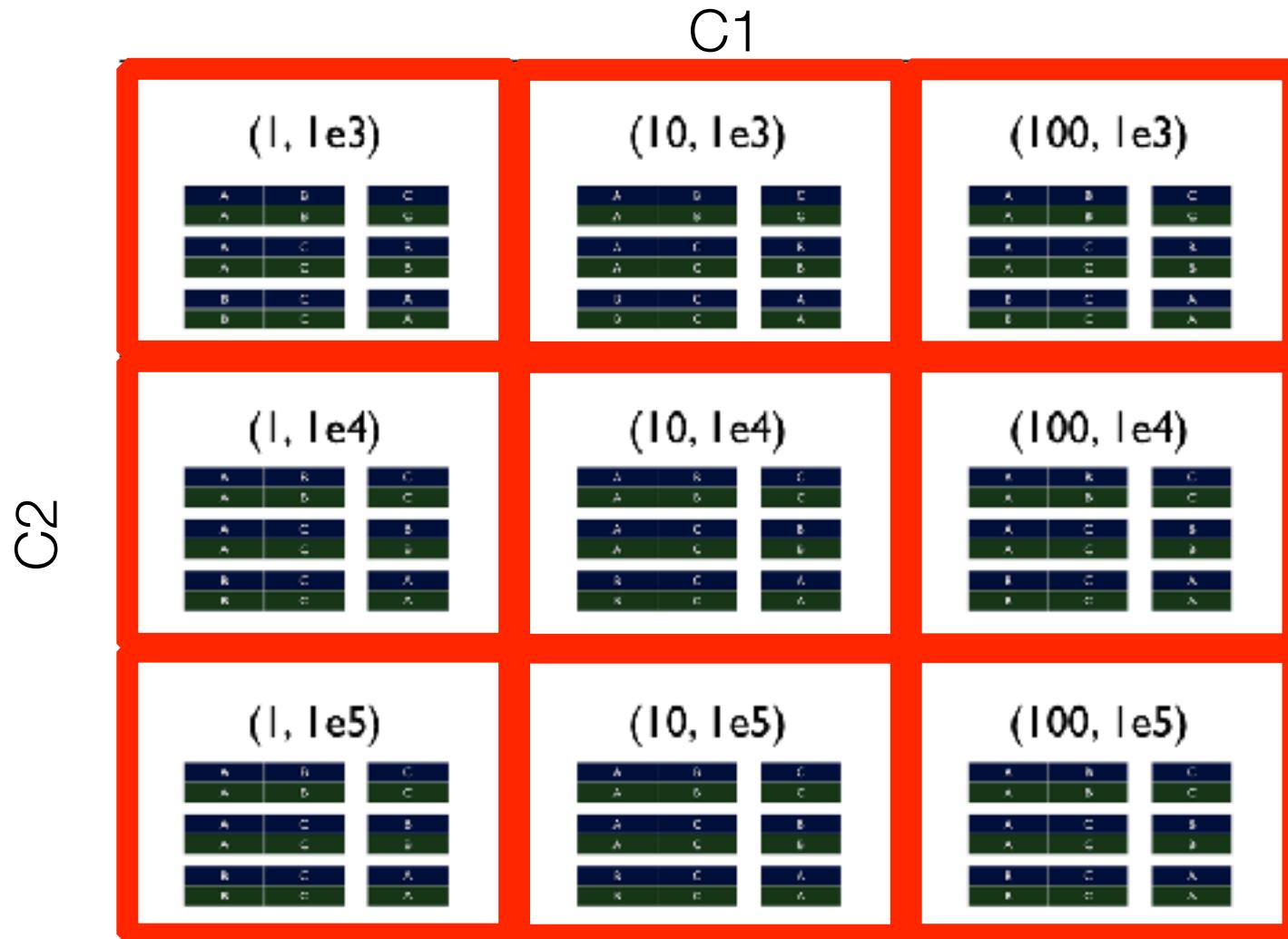
A	B	C
A	B	C
A	C	B
A	C	B
B	C	A
B	C	A

(100, 1e5)

A	B	C
A	B	C
A	C	B
A	C	B
B	C	A
B	C	A

Review: Grid Searching

or at random...



Review: Grid Searches in Scikit-learn

```
>>> from sklearn import svm, datasets
>>> from sklearn.model_selection import GridSearchCV
>>> iris = datasets.load_iris()
>>> parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
>>> svc = svm.SVC()
>>> clf = GridSearchCV(svc, parameters)
>>> clf.fit(iris.data, iris.target)
GridSearchCV(estimator=SVC(),
             param_grid={'C': [1, 10], 'kernel': ('linear', 'rbf')})
>>> sorted(clf.cv_results_.keys())
['mean_fit_time', 'mean_score_time', 'mean_test_score',...
 'param_C', 'param_kernel', 'params',...
 'rank_test_score', 'split0_test_score',...
 'split2_test_score', ...
 'std_fit_time', 'std_score_time', 'std_test_score']
```

Parameters:

- Fixed values
- Distribution of values, sampled

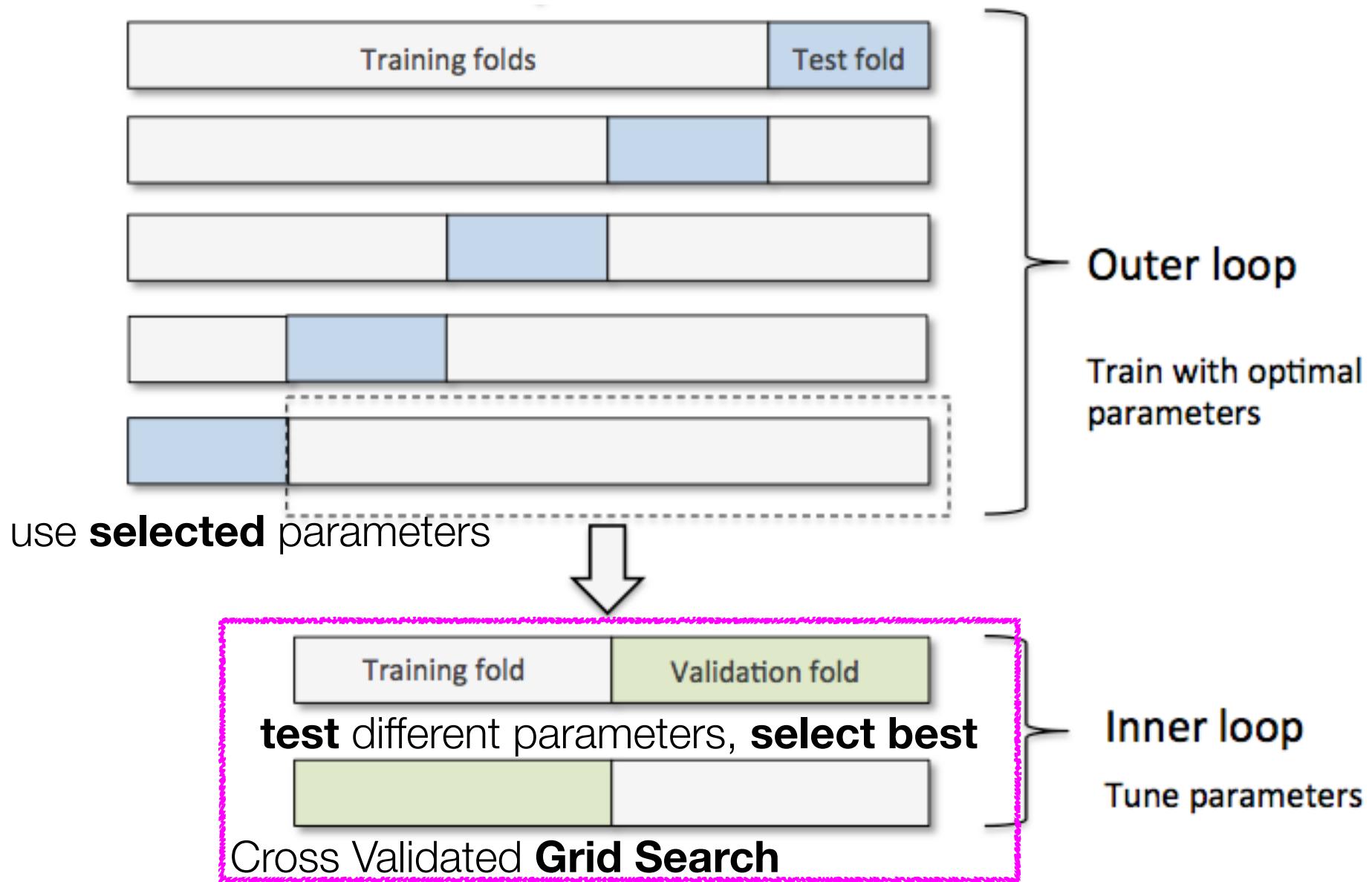
```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> from sklearn.model_selection import RandomizedSearchCV
>>> from scipy.stats import uniform
>>> iris = load_iris()
>>> logistic = LogisticRegression(solver='saga', tol=1e-2, max_iter=200,
                                  random_state=0)
...
>>> distributions = dict(C=uniform(loc=0, scale=4),
...                       penalty=['l2', 'l1'])
>>> clf = RandomizedSearchCV(logistic, distributions, random_state=0)
>>> search = clf.fit(iris.data, iris.target)
>>> search.best_params_
{'C': 2..., 'penalty': 'l1'}
```

Review: Data Snooping

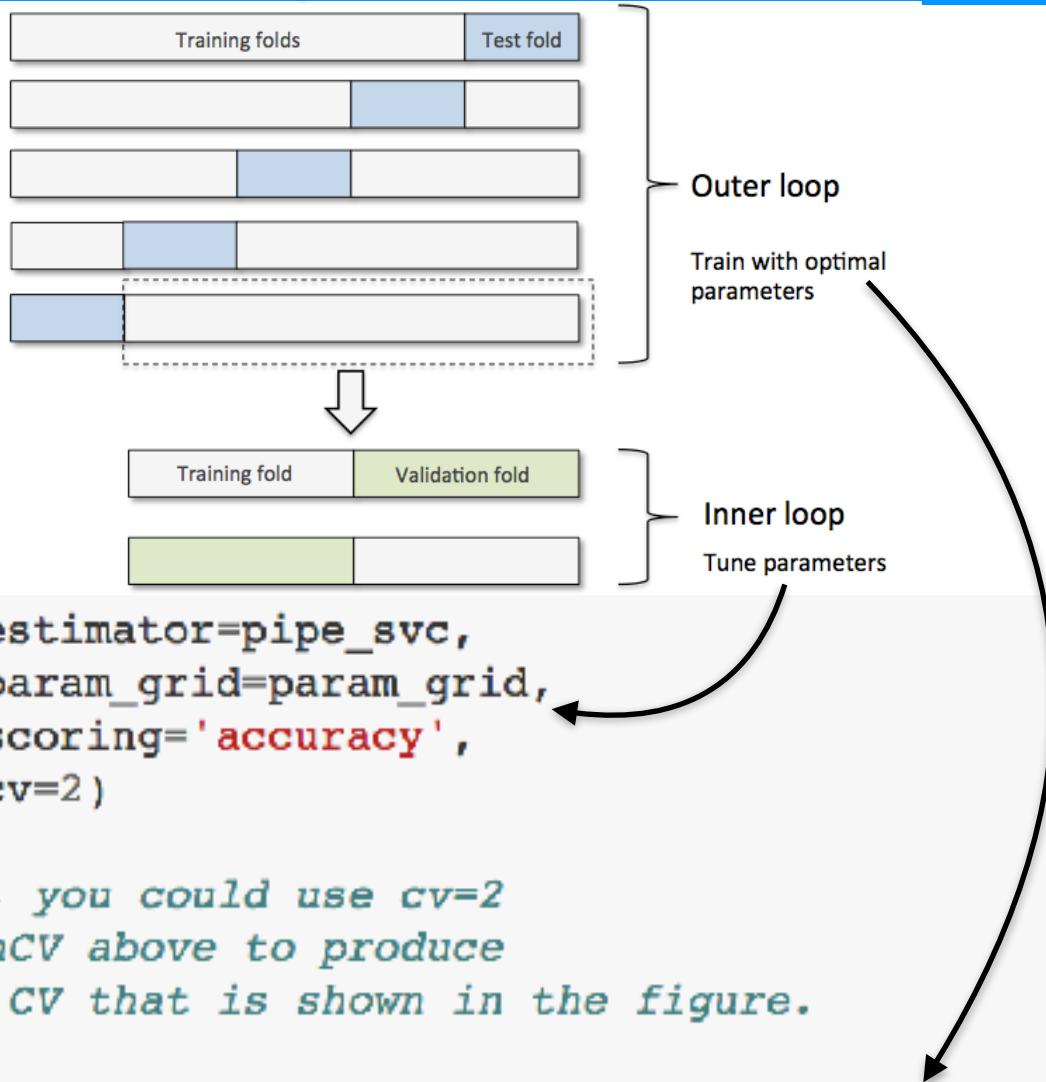
- Using the grid search parameters and testing on the same set...
 - the **performance on the dataset** could now be **biased**
 - cannot determine the **expected performance** on **new data**
 - this is **data snooping**



Review: Solution: Nested Cross Validation



Review: Nested Cross Validation: Hyper-parameters



```
scores = cross_val_score(gs, X_train, y_train, scoring='accuracy', cv=5)
print('CV accuracy: %.3f +/- %.3f' % (np.mean(scores), np.std(scores)))
```

Self Test

- **What is the end goal of nested cross-validation?**
 - A. To determine hyper parameters
 - B. To estimate generalization performance
 - C. To estimate generalization performance when performing hyper parameter tuning
 - D. To estimate the variation in tuned hyper parameters

McNemar Testing for Comparing Models

Few assumptions, **Null hypothesis**: these models are not different!

	Model 2 correct	Model 2 wrong
Model 1 correct	A	B
Model 1 wrong	C	D

One caveat: Statistical power depends upon $B+C$, which might be small, even with lots of test data.

McNemar and Edwards, 1948

$$\chi^2 = \frac{(|B - C| - 1)^2}{B + C}$$

χ squared statistic, one DOF

Steps:

1. Compare each model's performance on the same test data (2x2 matrix)
2. Calculate χ statistic
3. Look up *critical value* associated with χ statistic for given confidence
4. Can you reject the null hypothesis that the models are the same ($p < 0.05$)?

McNemar Example

Model 1	Model 2	Label	Matrix
T-shirt	T-shirt	T-shirt	A
Sneaker	T-shirt	Sneaker	B
T-shirt	Pullover	Pullover	C
Sneaker	Sneaker	Sneaker	A
T-shirt	Sneaker	Sneaker	C
Pullover	Pullover	T-shirt	D
Pullover	T-shirt	Pullover	B
Sneaker	Sneaker	Sneaker	A
Sneaker	Sneaker	Sneaker	A

		Model 2 correct	Model 2 wrong
Model 1 correct	4	A	
	2	B	
Model 1 wrong	2	C	
	1	D	

McNemar and Edwards, 1948

$$\chi^2 = \frac{(|B - C| - 1)^2}{B + C}$$

$$\chi^2 = \frac{(|2 - 2| - 1)^2}{2 + 2} = 0.25$$

Confidence	0.90	0.95	0.99
1 DOF, Critical Value	2.706	3.841	6.635

<https://www.itl.nist.gov/div898/handbook/eda/section3/eda3674.htm>

Since $0.25 < 3.841$, we cannot reject the null hypothesis.

This means **we cannot say the models are different** based on the evidence.

Some History of Deep Learning

When you move on to
Deep Learning



Neural Networks: Where we left it

- Before 1986: AI Winter
- 1986: *Rumelhart, Hinton, and Williams* popularize gradient calculation for multi-layer network
 - *technically* introduced by Werbos in 1982
- **difference:** Rumelhart *et al.* validated ideas with a computer
- until this point no one could train a multiple layer network consistently
- algorithm is popularly called **Back-Propagation**
- wins pattern recognition prize in 1993, becomes de-facto machine learning algorithm in the 90's

David Rumelhart



Geoffrey Hinton



Machine Learning Timeline (Neural Nets)

- Up to this point: back propagation saved AI winter
- 80's, 90's, 2000's: neural networks for image processing start to get deeper
 - but back propagation no longer efficient for training
 - Back propagation gradient **stagnates** research—can't train **deeper** networks



1949, Hebb's Law
Close neuron fire together



1960, Widrow & Hoff
Adaline Network



1969, Minsky & Papert
Linear Models are Doomed



1986, Rumelhart & Hinton
Back-propagation



2003, Vapnik
Kernel SVMs



1943, McCulloch & Pitts
Logic Gates of The Mind

1957, Rosenblatt
Perceptron

1969, Minsky & Papert
Linear Models are Doomed

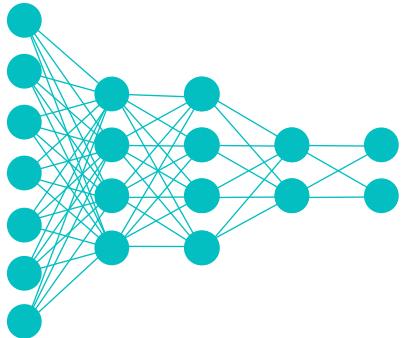
2001, Breiman
Random Forests

Read this: <http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/>

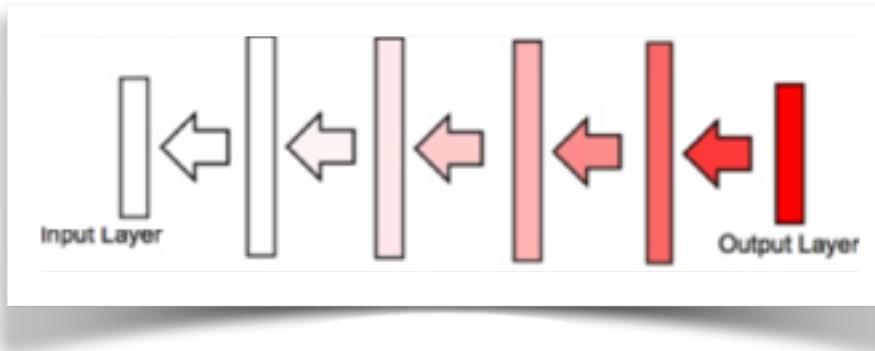
www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/

History of Deep Learning: Winter

- AI Winter is coming:



Easy to train, performs on par with other methods



Hard to train, performs worse than other methods
~chance (untrainable)

Researcher have difficulty reconciling expressiveness with performance



Read this: <http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/>

Machine Learning Timeline (Neural Nets)

- 2004: Hinton secures funding from CIFAR based on his reputation
 - *eventually*: Canada would be savior for neural networks
 - Hinton rebrands: **Deep Learning**
- 2006: Hinton publishes paper on using pre-training and Restricted Boltzmann Machines
- 2007: Another paper: Deep networks are more efficient when pre-trained
 - RBMs not really the important part



1949, Hebb's Law
Close neuron fire together



1960, Widrow & Hoff
Adaline Network



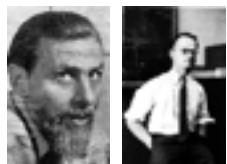
1986, Rumelhart & Hinton
Back-propagation



2003, Vapnik
Kernel SVMs



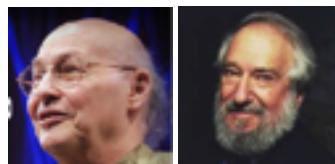
1943, McCulloch & Pitts
Logic Gates of The Mind



1957, Rosenblatt
Perceptron



1969, Minsky & Papert
Linear Models are Doomed



2001, Breiman
Random Forests

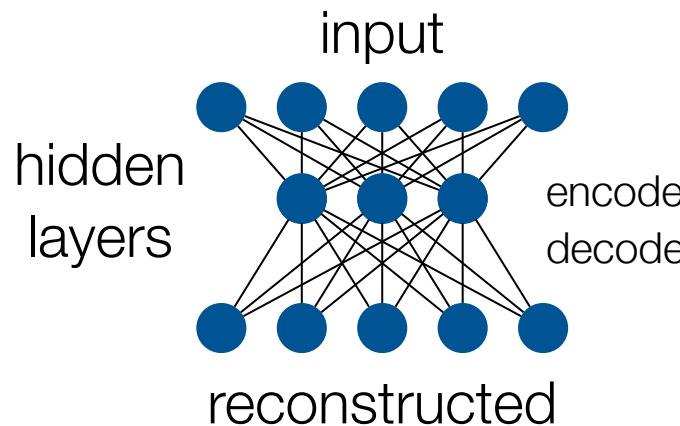


Read this: <http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/>

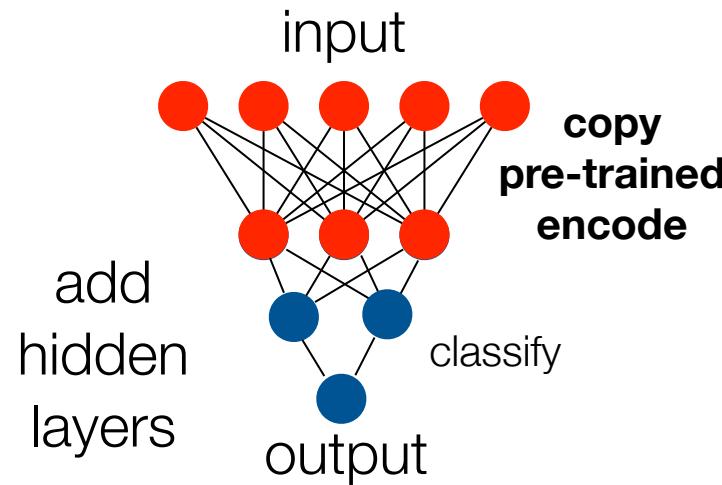
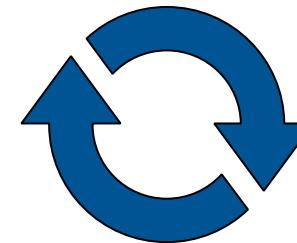
<http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/>

Pre-training: still in the long winter

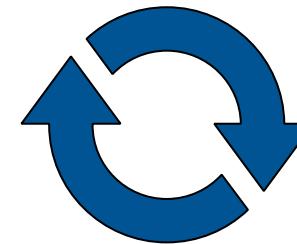
- auto-encoding (a form of pre-training)



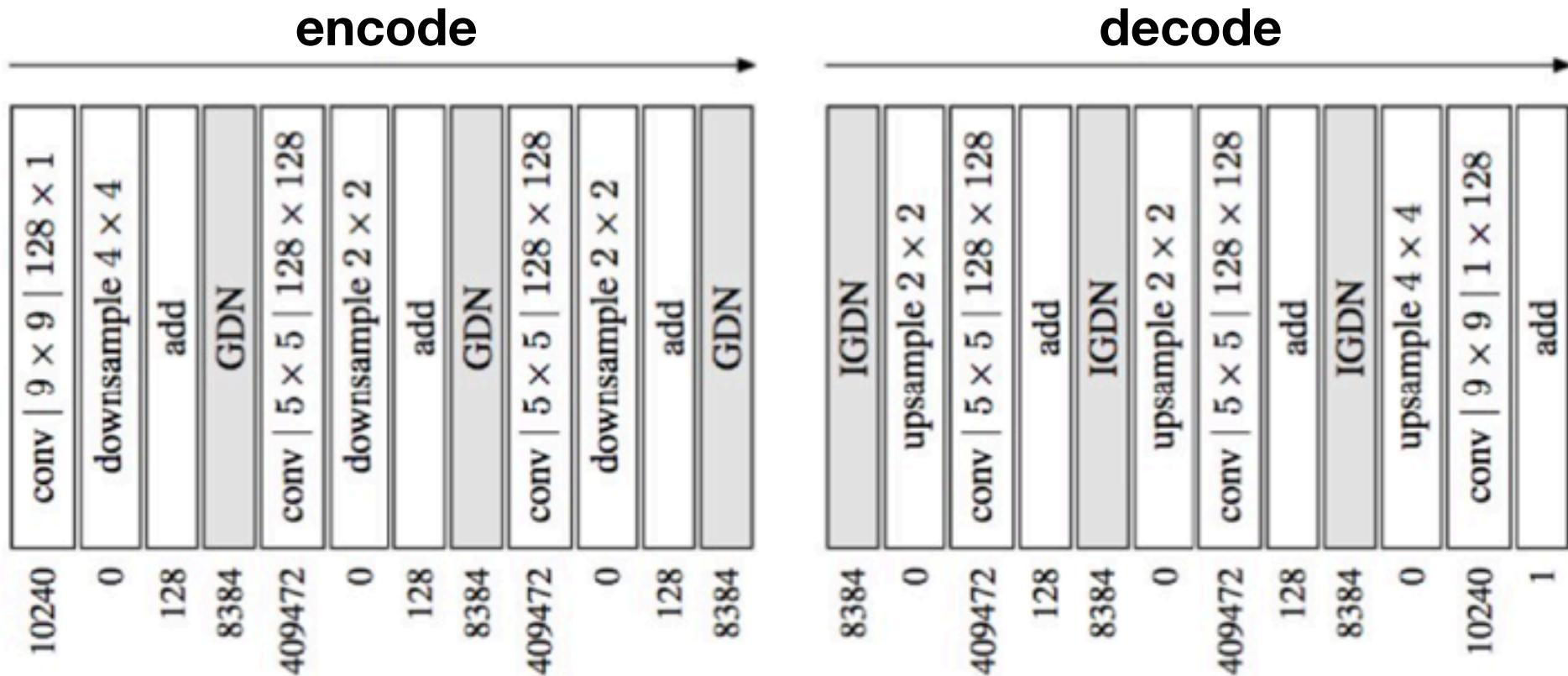
train with lots of
unlabeled data



train with
labeled data



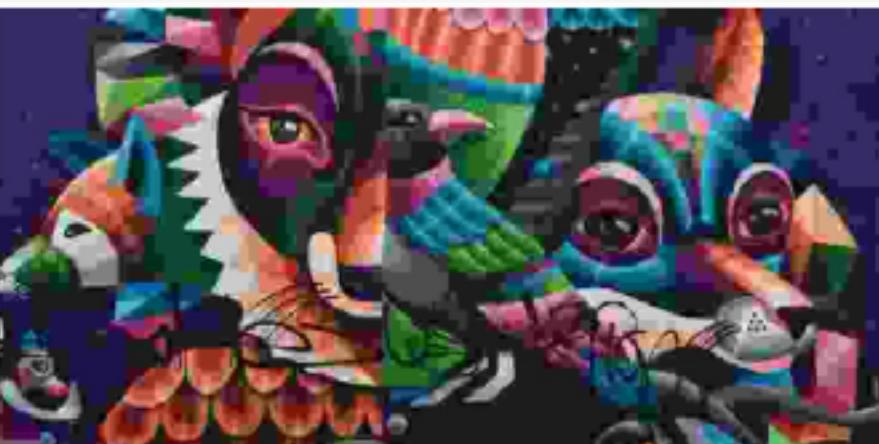
Pre-training: modern example



<https://arxiv.org/abs/1611.01704>



JPEG, 6006 bytes (0.170 bit/px), RMSE: 19.75



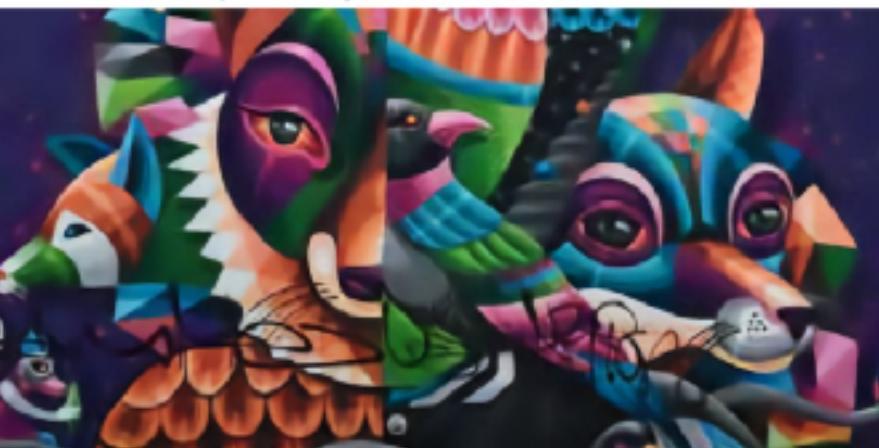
JPEG, 5928 bytes (0.168 bit/px), RMSE: 15.44/12.40, PSNR: 24.36 dB/26.26 dB



RMSE: 11.07/10.60, PSNR: 27.25 dB/27.63 dB



Proposed method, 5910 bytes (0.167 bit/px), RMSE



Proposed method, 5685 bytes (0.161 bit/px), RMSE: 10.41/5.98, PSNR: 27.78 dB/32.60 dB



bit/px), RMSE: 6.10/5.09, PSNR: 32.43 dB/34.00 dB



JPEG 2000, 5918 bytes (0.167 bit/px), RMSE: 11



JPEG 2000, 5724 bytes (0.162 bit/px), RMSE: 13.75/7.00, PSNR: 25.36 dB/31.20 dB



bit/px), RMSE: 8.56/5.71, PSNR: 29.49 dB/32.99 dB

Still in the Long Winter

- 2009: Hinton's lab starts using GPUs, Also Andrew Ng
 - GPUs decrease training time by 70 fold...
- 2010: Hinton's and Ng's students go to internships with Microsoft, Google, IBM, and Facebook



Navdeep Jaitly
Google Brain Team



George Dahl
Google Brain Team

Abdel-rahman Mohamed

Microsoft Research
Redmond, Washington | Computer Software

Current Microsoft
Previous University of Toronto, IBM, Microsoft
Education University of Toronto

- Xbox Voice
- Android Speech Recognition
- IBM Watson
- DeepFace
- All of Baidu



1949, Hebb's Law
Close neuron fire together



1960, Widrow & Hoff
Adaline Network



1986, Rumelhart & Hinton
Back-propagation



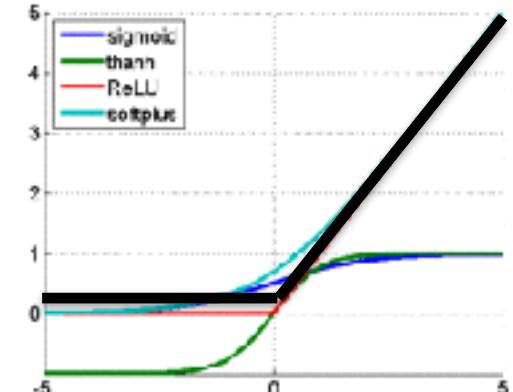
2003, Vapnik
Kernel SVMs



Read this: <http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/>

Still in the long Winter

- 2011: Glorot and Bengio investigate more systematic methods for why past deep architectures did not work
 - **discover some interesting, simple fixes:** the type of neurons chosen and the selection of initial weights
 - do not require pre-training to get deep networks properly trained, just sparser representations and less complicated derivatives



ReLU: $f(x) = \max(0, x)$
 $f'(x) = 1 \text{ if } x > 0 \text{ else } 0$



1949, Hebb's Law
Close neuron fire together



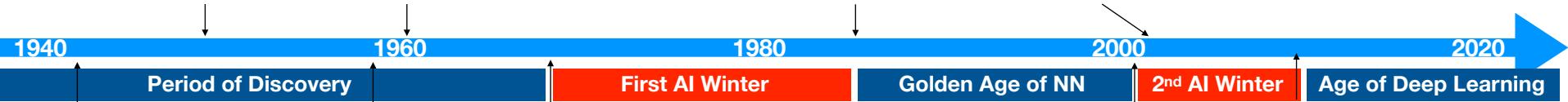
1960, Widrow & Hoff
Adaline Network



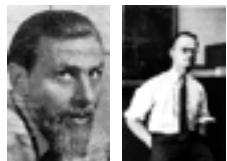
1986, Rumelhart & Hinton
Back-propagation



2003, Vapnik
Kernel SVMs



1943, McCulloch & Pitts
Logic Gates of The Mind



1957, Rosenblatt
Perceptron



1969, Minsky & Papert
Linear Models are Doomed



2001, Breiman
Random Forests



2011, Bengio
Init and ReLU



Read this: <http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/>

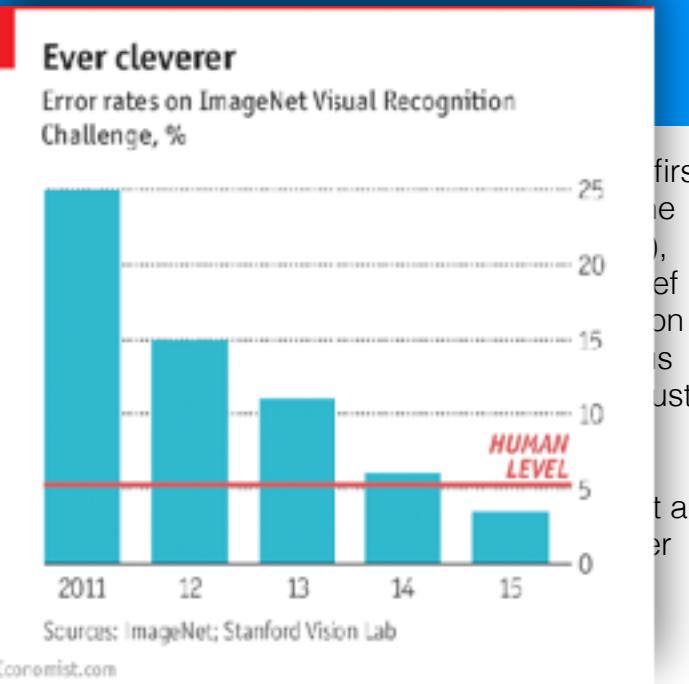
Machine Learning Timeline (1940-2020)

- **ImageNet competition occurs**
- **Second place:** 26.2% error rate
- **First place:**
 - From Hinton's lab, uses convolutional network with ReLU and dropout
 - 15.2% error rate
- Computer vision adopts deep learning with convolutional neural networks en masse



Fei Fei Li
Director of Stanford's
AI Lab (Former)
HAI Founder

"I have had a hand in last few years so I must say as she comes along pacifying people happens from skeptics to a cool Vision



1949, Hebb's Law
Close neuron fire together



1960, Widrow & Hoff
Adaline Network



1986, Rumelhart & Hinton
Back-propagation



2003, Vapnik
Kernel SVMs



2012, Hinton, Fei-Fei Li
CNNs win ImageNet



1943, McCulloch & Pitts
Logic Gates of The Mind



1957, Rosenblatt
Perceptron



1969, Minsky & Papert
Linear Models are Doomed



2001, Breiman
Random Forests



2011, Bengio
Init and ReLU



Read this: <http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/>

Machine Learning Timeline (Neural Nets)

- 2012: Hinton Lab, Google, IBM, and Microsoft jointly publish paper, popularity for deep learning methods increases

Deep Neural Networks for Acoustic Modeling in Speech Recognition

[The shared views of four research groups]

Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury

[https://www.cs.toronto.edu/~gdahl/papers/
deepSpeechReviewSPM2012.pdf](https://www.cs.toronto.edu/~gdahl/papers/deepSpeechReviewSPM2012.pdf)



1949, Hebb's Law
Close neuron fire together



1960, Widrow & Hoff
Adaline Network



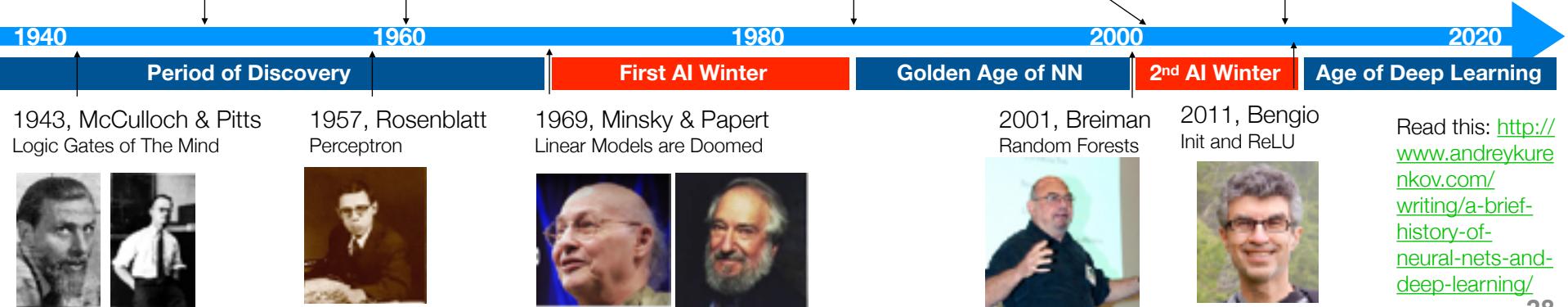
1986, Rumelhart & Hinton
Back-propagation



2003, Vapnik
Kernel SVMs



2012, Hinton, Fei-Fei Li
CNNs win ImageNet



Machine Learning Timeline (Neural Nets)

- 2013: Andrew Ng and Google (BrainTeam)
 - run unsupervised feature creation on YouTube videos (becomes computer vision benchmark)

The work resulted in unsupervised neural net learning of an unprecedented scale - 16,000 CPU cores powering the learning of a whopping 1 billion weights. The neural net was trained on YouTube videos, entirely without labels, and learned to recognize the most common objects in those videos.



1949, Hebb's Law
Close neuron fire together



1960, Widrow & Hoff
Adaline Network



1986, Rumelhart & Hinton
Back-propagation



2003, Vapnik
Kernel SVMs



2012, Hinton, Fei-Fei Li
CNNs win ImageNet



Period of Discovery

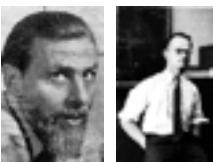
First AI Winter

Golden Age of NN

2nd AI Winter

Age of Deep Learning

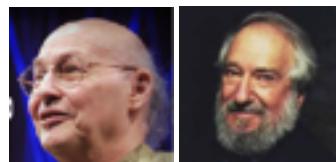
1943, McCulloch & Pitts
Logic Gates of The Mind



1957, Rosenblatt
Perceptron



1969, Minsky & Papert
Linear Models are Doomed



2001, Breiman
Random Forests

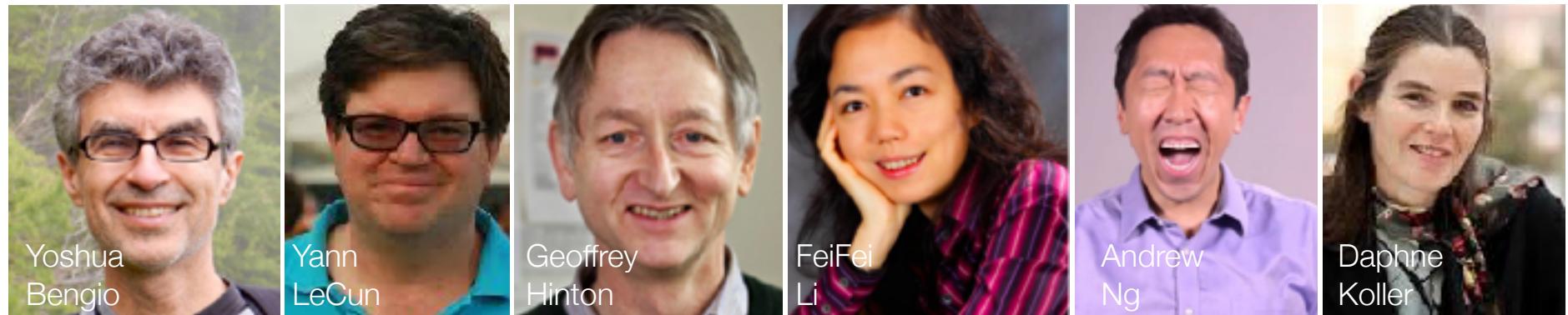


2011, Bengio
Init and ReLU



Read this: <http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/>

A summary of the Deep Learning people:



Yoshua
Bengio

Stayed at Univ.
Montreal
Advises IBM

Yann
LeCun

Heads
Facebook
AI Team

Geoffrey
Hinton

Univ. Toronto
Google

FeiFei
Li

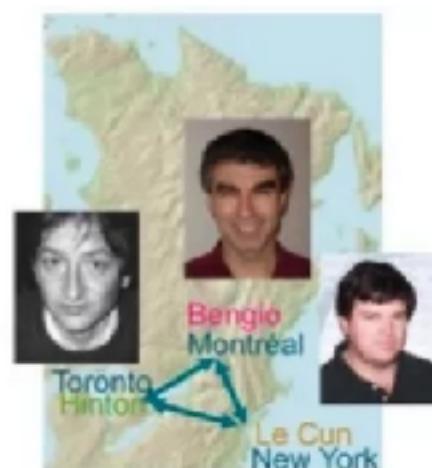
Stanford (HAI)
Former Chief Scien.,
AI/ML Google Cloud

Andrew
Ng

Coursera
Baidu
Google

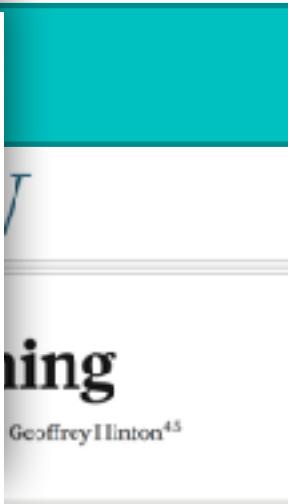
Daphne
Koller

Stanford
Founded Coursera
MacArthur Genius



deep learning

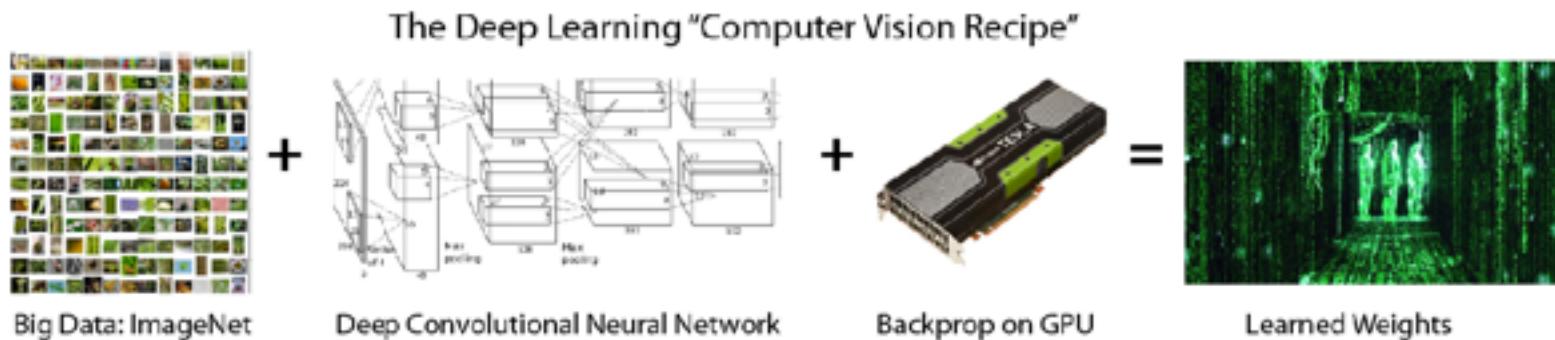
- Hinton: Restricted Boltzmann Machine, Deep autoencoder
- Bengio: neural language modeling.
- LeCun: Convolutional Neural Network
- NIPS, ICML, CVPR, ACL
- Google Brain, Deep Mind.
- Facebook AI.



Made Deep Learning
Instruction Accessible

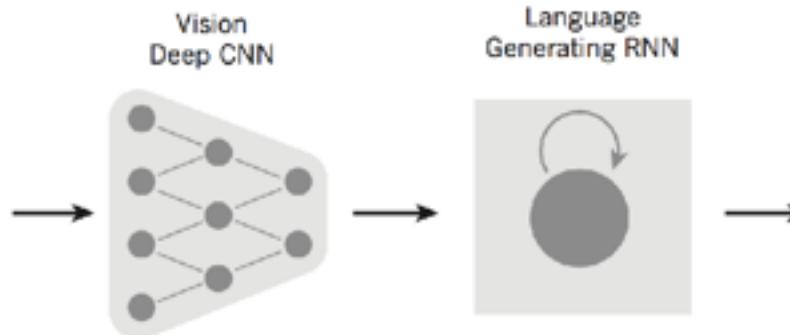
History of Deep Learning

- **Hinton** summarized what we learned in deep learning from the 2006 to present. Where we went wrong before present day:
 - labeled dataset were 1000s of times too small
 - computers were millions of times too slow
 - weights were initialized in stupid ways
 - we used the wrong non-linearities
- Or **Larson's Laws:**
 - use a GPU when possible, init weights for consistent gradient magnitude, ReLU/SiLU where it makes sense (like in early feedforward layers), and lots of dropout in the final layers that tend to learn more quickly!



Read this: <http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/>

Famous examples:



A group of people shopping at an outdoor market.

There are many vegetables at the fruit stand.



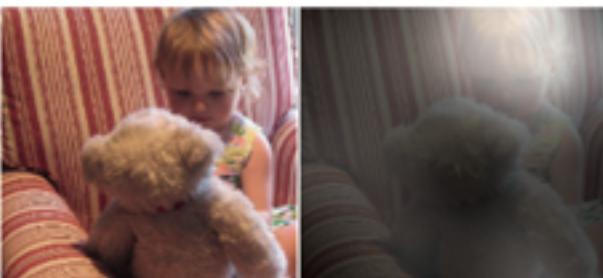
A woman is throwing a **frisbee** in a park.



A **dog** is standing on a hardwood floor.



A **stop** sign is on a road with a **mountain** in the background.



A little girl sitting on a bed with a **teddy bear**.



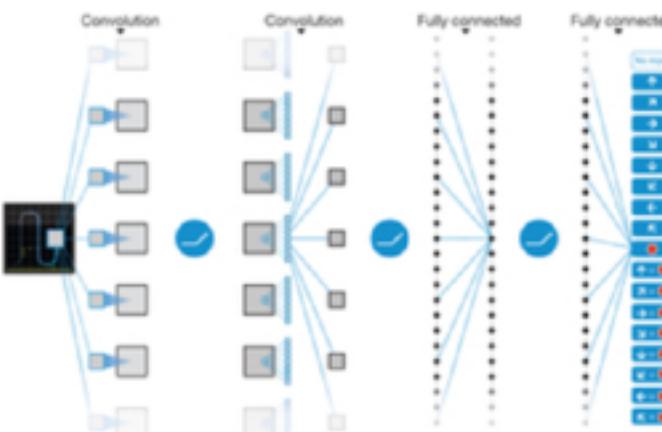
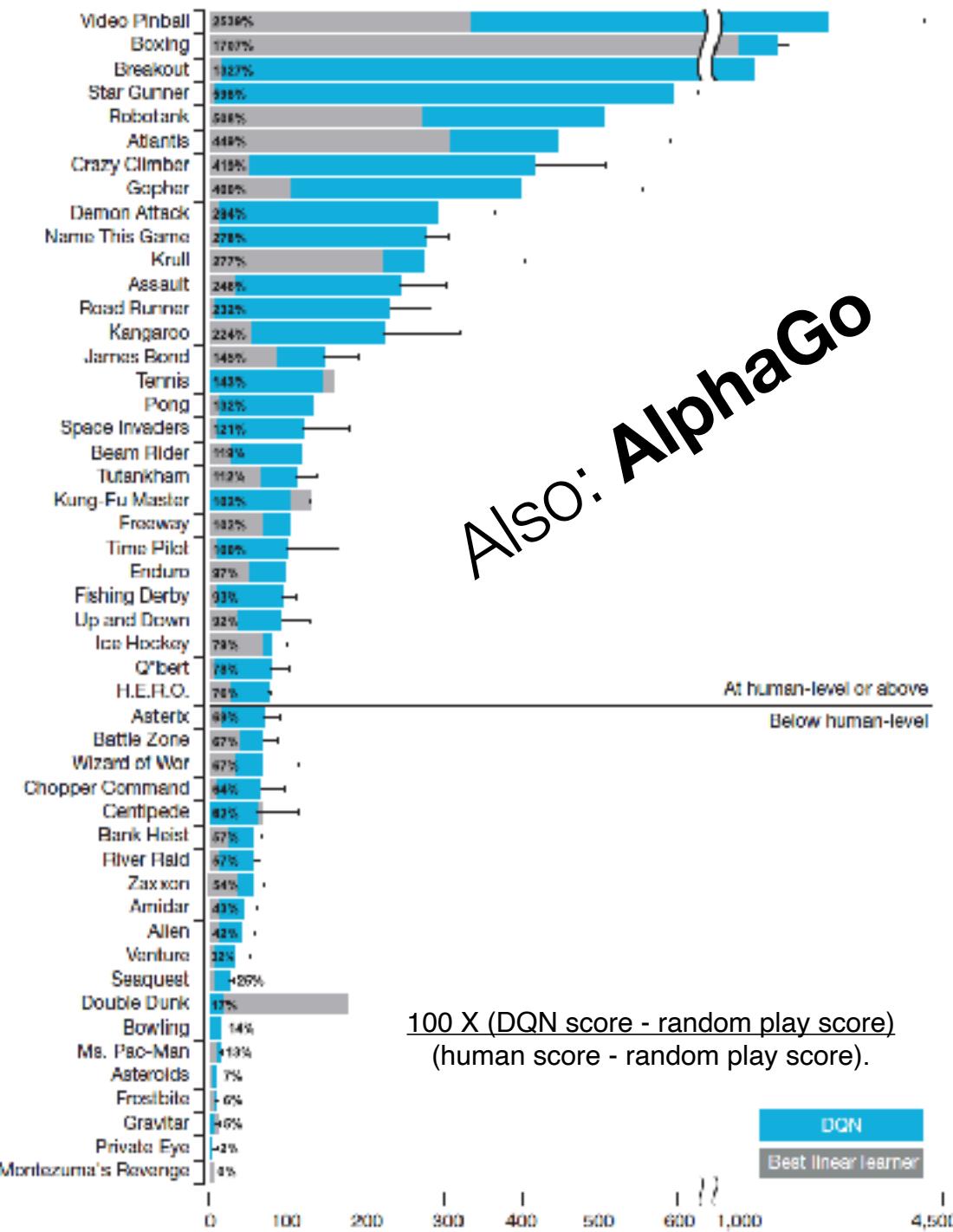
A group of **people** sitting on a boat in the water.



A **giraffe** standing in a forest with **trees** in the background.

More Famous

Also: AlphaGo



Credit for Deep Learning

Official ACM @TheOfficialACM

Yoshua Bengio, Geoffrey Hinton and Yann LeCun, the fathers of #DeepLearning, receive the 2018 #ACMTuringAward for conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing today. bit.ly/2HVJtdV



Yoshua Bengio

Geoffrey Hinton

Yann LeCun



Machine learning is the science of credit assignment. The machine learning community itself profits from proper credit assignment to its members. The inventor of an important method should get credit for inventing it. She may not always be the one who popularizes it. Then the popularizer should get credit for popularizing it (but not for inventing it). Relatively young research areas such

Review of Deep Learning History

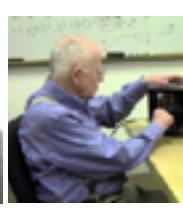
- Up to this point: back propagation saved AI winter for NN (Hinton and others!)
- 80's, 90's, 2000's: convolutional networks for image processing start to get deeper
 - but back propagation no longer does great job at training them
- SVMs and Random Forests gain traction...
 - The second AI winter begins, research in NN plummets
- 2004: Hinton secures funding from CIFAR in 2004 Hinton rebrands: Deep Learning
- 2006: Auto-encoding and Restricted Boltzmann Machines
- 2007: Deep networks are more efficient when pre-trained
- 2009: GPUs decrease training time by 70 fold...
- 2010: Hinton's students go to internships with Microsoft, Google, and IBM, making their speech recognition systems faster, more accurate and deployed in only 3 months...
- 2012: Hinton Lab, Google, IBM, and Microsoft jointly publish paper, popularity sky-rockets for deep learning methods
- 2011-2013: Ng and Google run unsupervised feature creation on YouTube videos (becomes computer vision benchmark)
- 2012+: Pre-training is not actually needed, just solutions for vanishing gradients (like ReLU, SiLU, initializations, more data, GPUs)



1949, Hebb's Law
Close neuron fire together



1960, Widrow & Hoff
Adaline Network



1986, Rumelhart & Hinton
Back-propagation



2003, Vapnik
Kernel SVMs



2012, Hinton, Fei-Fei Li
CNNs win ImageNet



1940

1960

1980

2000

2020

Period of Discovery

First AI Winter

Golden Age of NN

2nd AI Winter

Age of Deep Learning

1943, McCulloch & Pitts
Logic Gates of The Mind

1957, Rosenblatt
Perceptron

1969, Minsky & Papert
Linear Models are Doomed

2001, Breiman
Random Forests

2011, Bengio
Init and ReLU

2015, Google
Tensorflow Open Source



TensorFlow 35

End of Session

- Next Time:
 - Introduction to TensorFlow
 - Wide and Deep Networks