

Lecture Notes for **Machine Learning in Python**

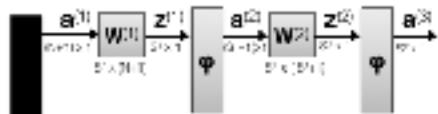
Revisiting CV A “Way too Early” History of Deep Learning

Logistics and Agenda

- Logistics
 - Next time: Wide and Deep Networks
- Agenda
 - Cross Validation
 - Town Hall
 - “Deep Learning” History
 - Remaining Topics
 - TensorFlow from 10,000 feet
 - Wide and Deep Networks
 - Convolution Neural Networks
 - Recurrent Neural Networks

Last time:

Back propagation summary



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

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

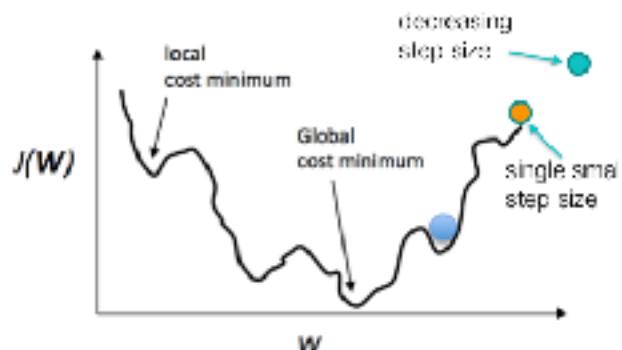
1. Forward propagate to get \mathbf{z}, \mathbf{a} for all layers
 2. Get final layer gradient
 3. Update back propagation variables
 4. Update each $\mathbf{W}^{(l)}$
- for each $y^{(l)}$
- $$\frac{\partial J(\mathbf{W})}{\partial a^{(l)}} = -2(y^{(l)} - a^{(l)}) \times a^{(l)} + (1 - a^{(l)})$$
- $$\frac{\partial J(\mathbf{W})}{\partial z^{(l)}} = \text{diag}[a^{(l+1)} * (1 - a^{(l+1)})] \cdot W^{(l+1)} \frac{\partial J(\mathbf{W})}{\partial z^{(l+1)}}$$
- $$W^{(l)} \leftarrow W^{(l)} - \eta \frac{\partial J(\mathbf{W}^{(l)})}{\partial z^{(l)}} \cdot a^{(l)}$$

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



2

Practical Implementation of Architectures

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

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

speeds up initial training

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

vectorized backpropagation
signal = (A2 - Y_enc) # <- this is only line
signal2 = (W2.T @ signal) * A2 * (1 - A2)

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

grad1 = signal2[1:,1] @ A1
grad2 = signal3 @ A2.T

new update

vectorized backpropagation
signal = -2*(Y_enc - A2) * A2 * (1 - A2)
signal2 = (W2.T @ signal) * A2 * (1 - A2)

grad1 = signal2[1:,1] @ A1
grad2 = signal3 @ A2.T

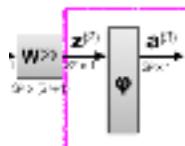
$$\frac{\partial J(\mathbf{W})}{\partial z^{(2)}} = -2(y^{(2)} - a^{(2)}) \times a^{(2)} + (1 - a^{(2)})$$

old update

bp-5
62

Practical Implementation of Architectures

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



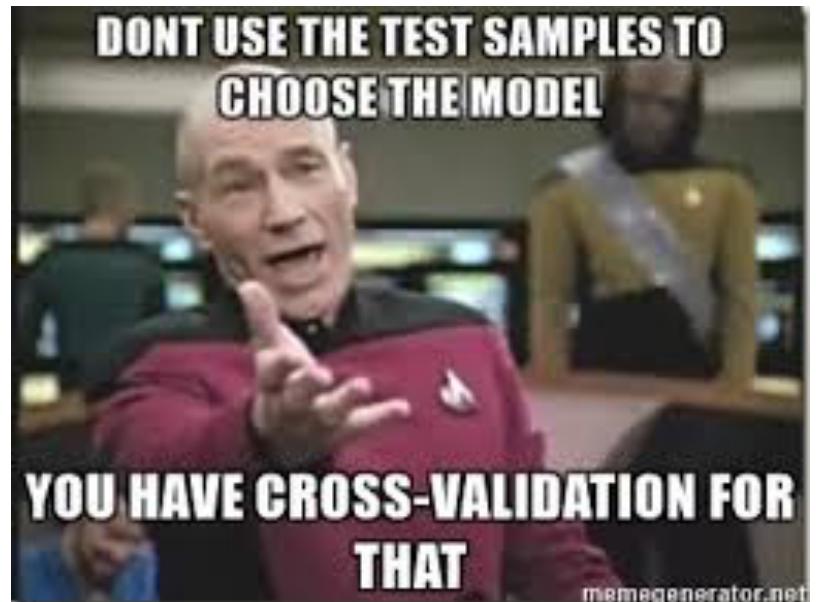
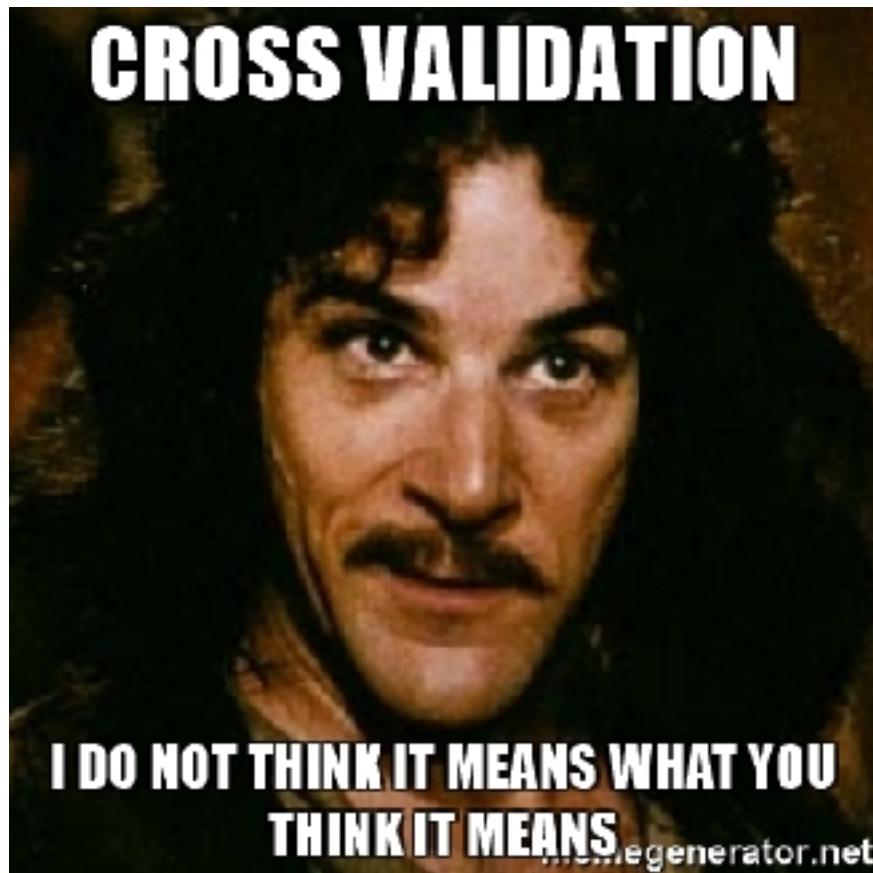
$$\phi(z^{(l)}) = \begin{cases} z^{(l)}, & \text{if } z^{(l)} > 0 \\ 0, & \text{else} \end{cases}$$

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

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

3

Revisiting Cross Validation



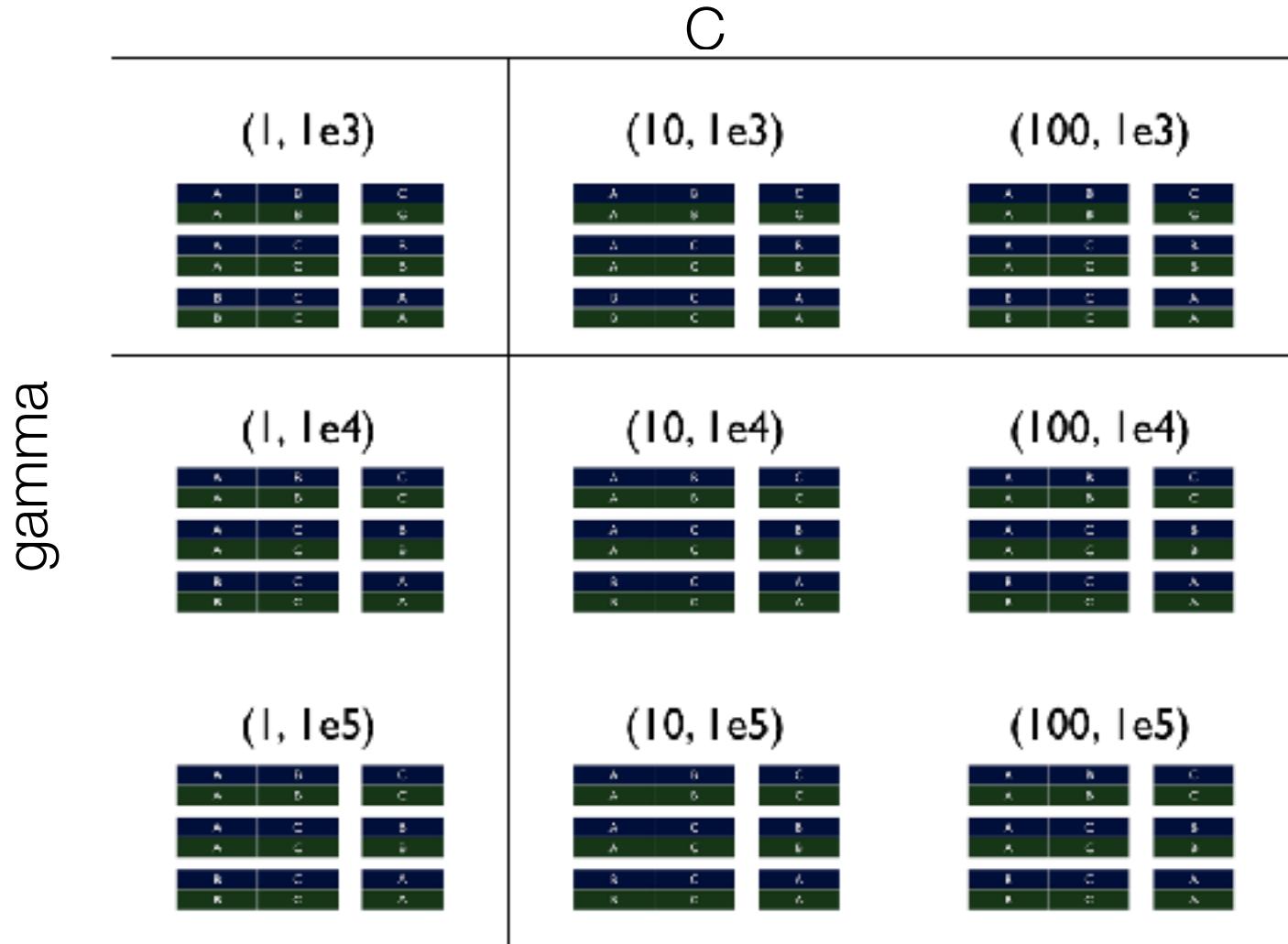
Grid Searching

- Trying to find the best parameters
 - SVM: $C=[1, 10, 100]$ $\gamma=[1e3, 1e4, 1e5]$

| | | C | | |
|--|--|----------|-----------|------------|
| | | (1, 1e3) | (10, 1e3) | (100, 1e3) |
| | | (1, 1e4) | (10, 1e4) | (100, 1e4) |
| | | (1, 1e5) | (10, 1e5) | (100, 1e5) |

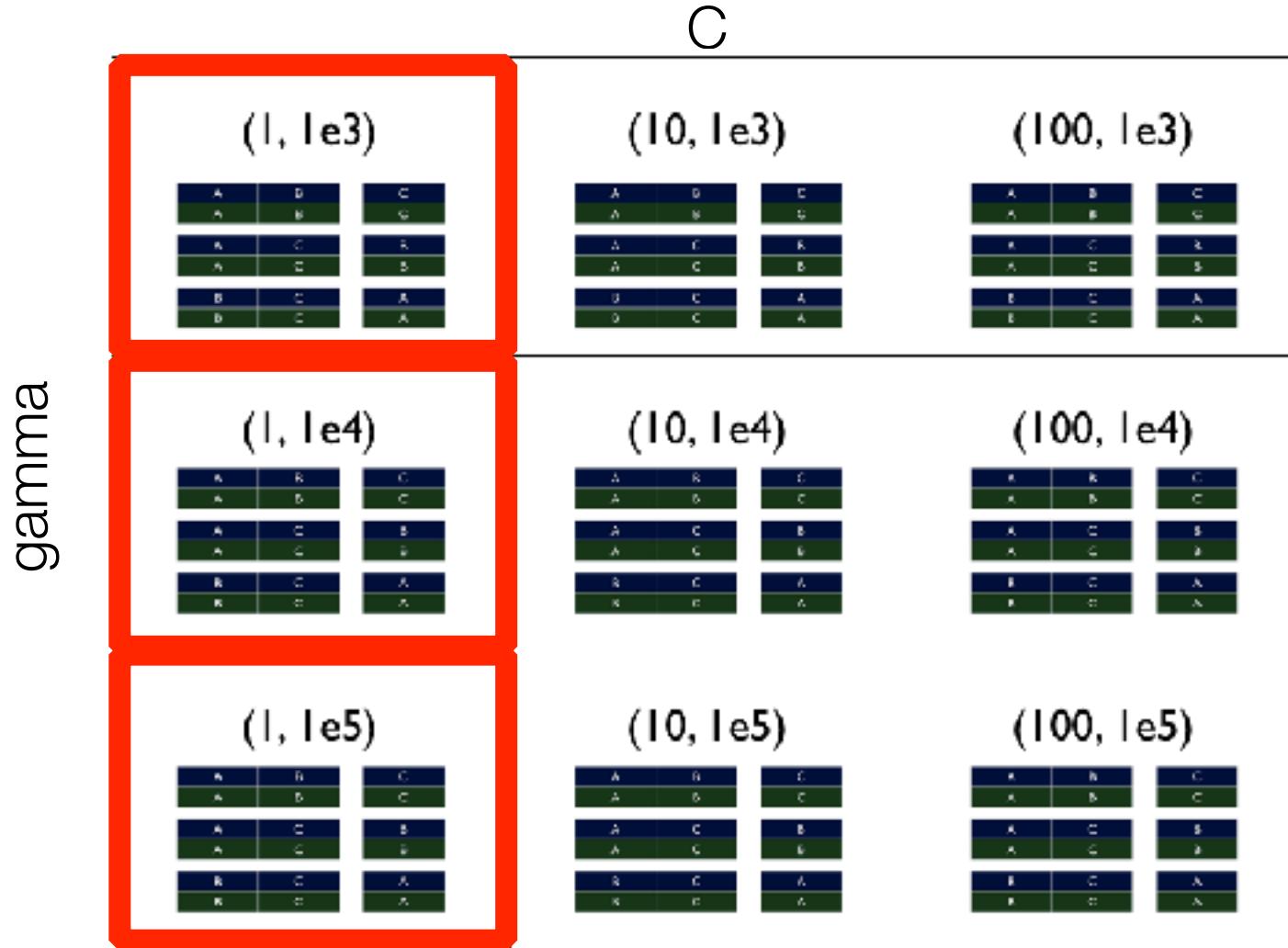
Grid Searching

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



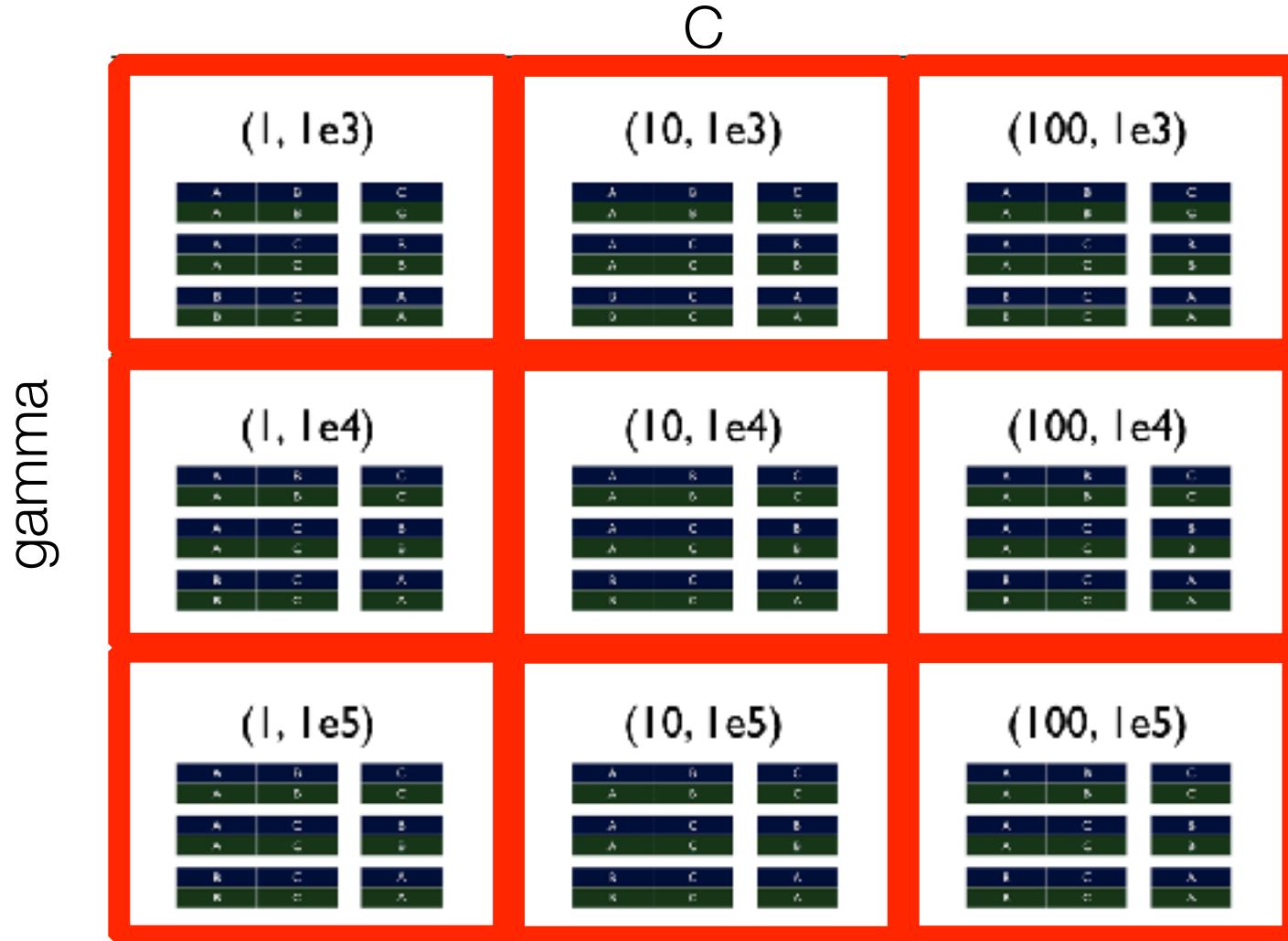
Grid Searching

- Could perform iteratively

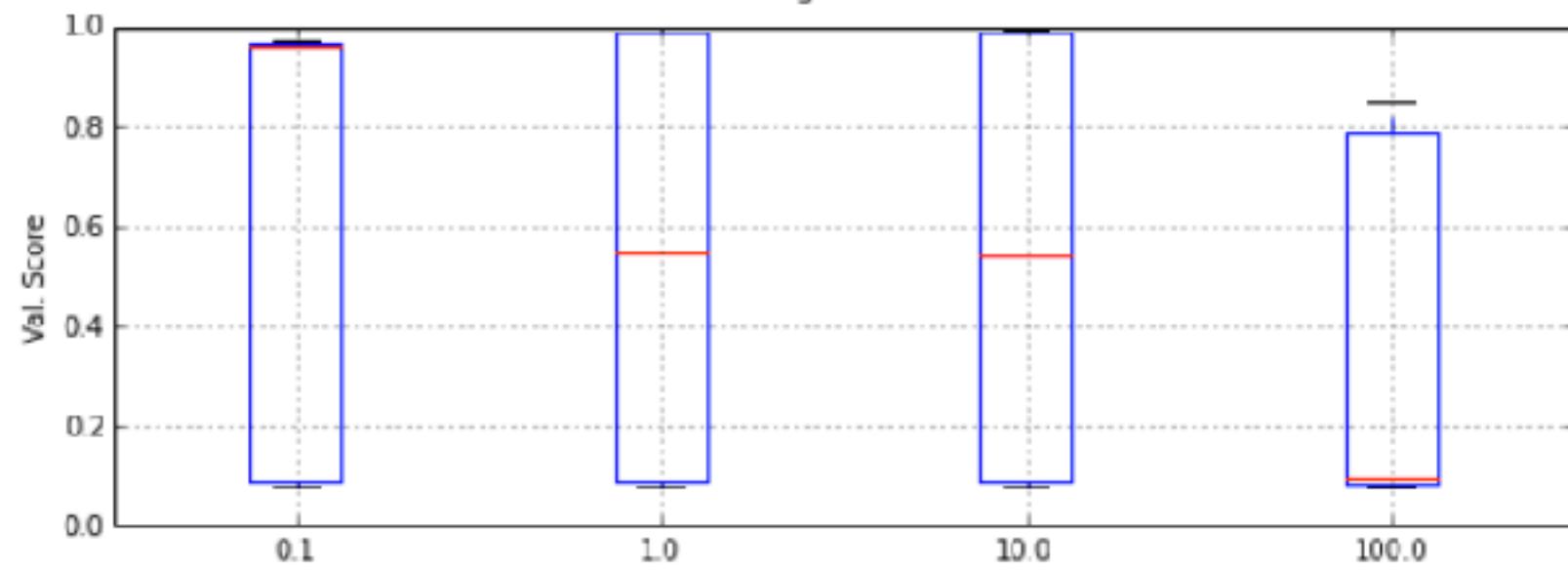
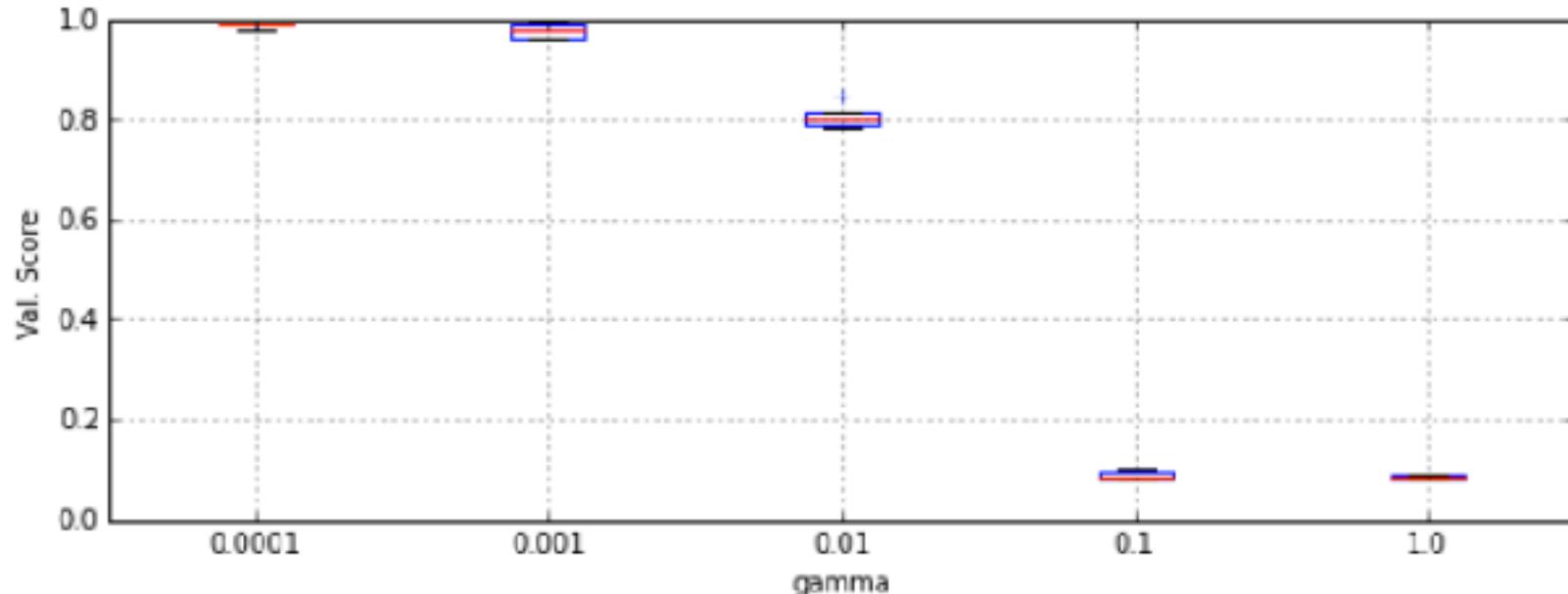


Grid Searching

- or at random...



```
print(search.report())
search.boxplot_parameters(display_train=False)
```

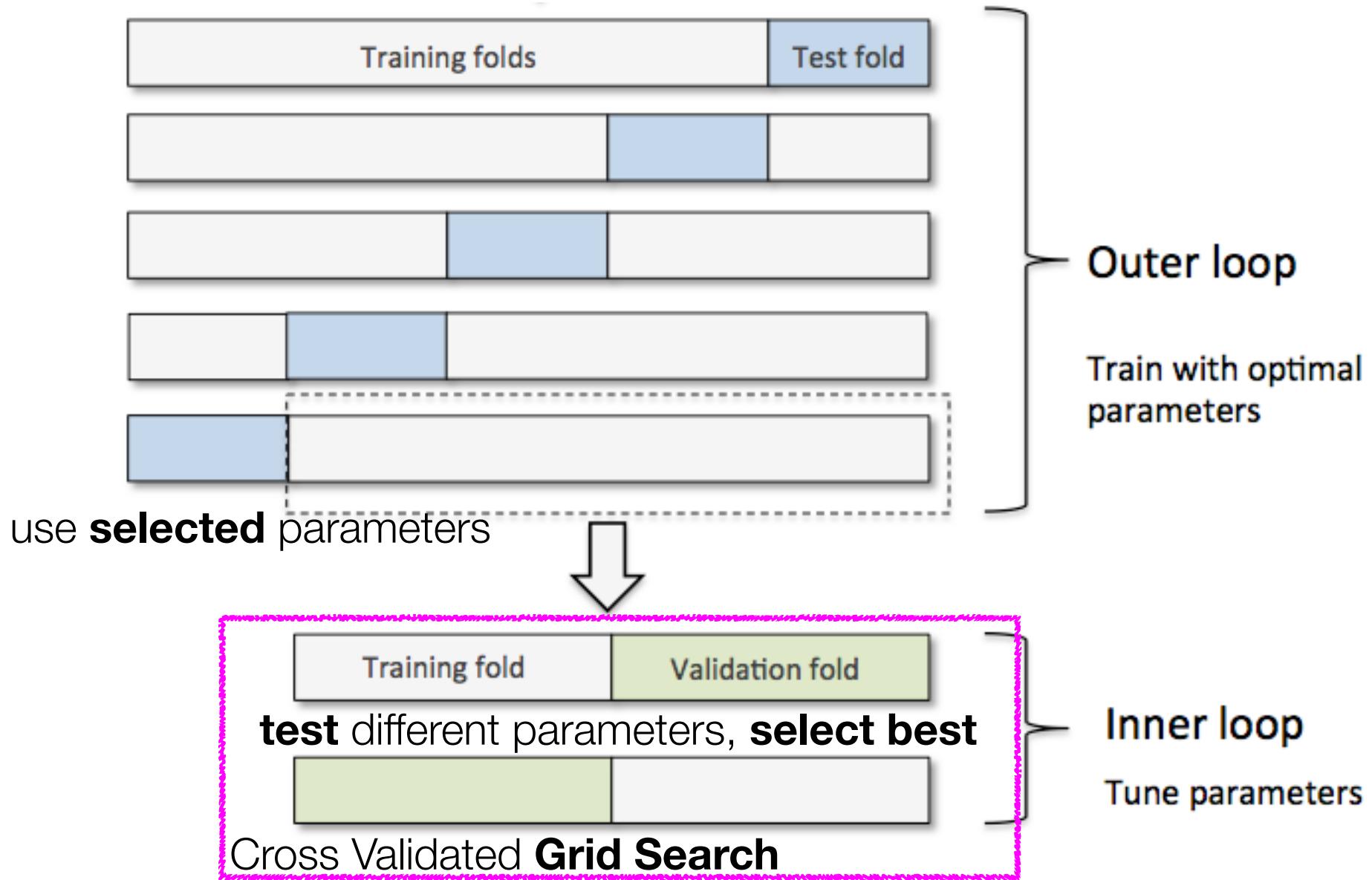


The Problem

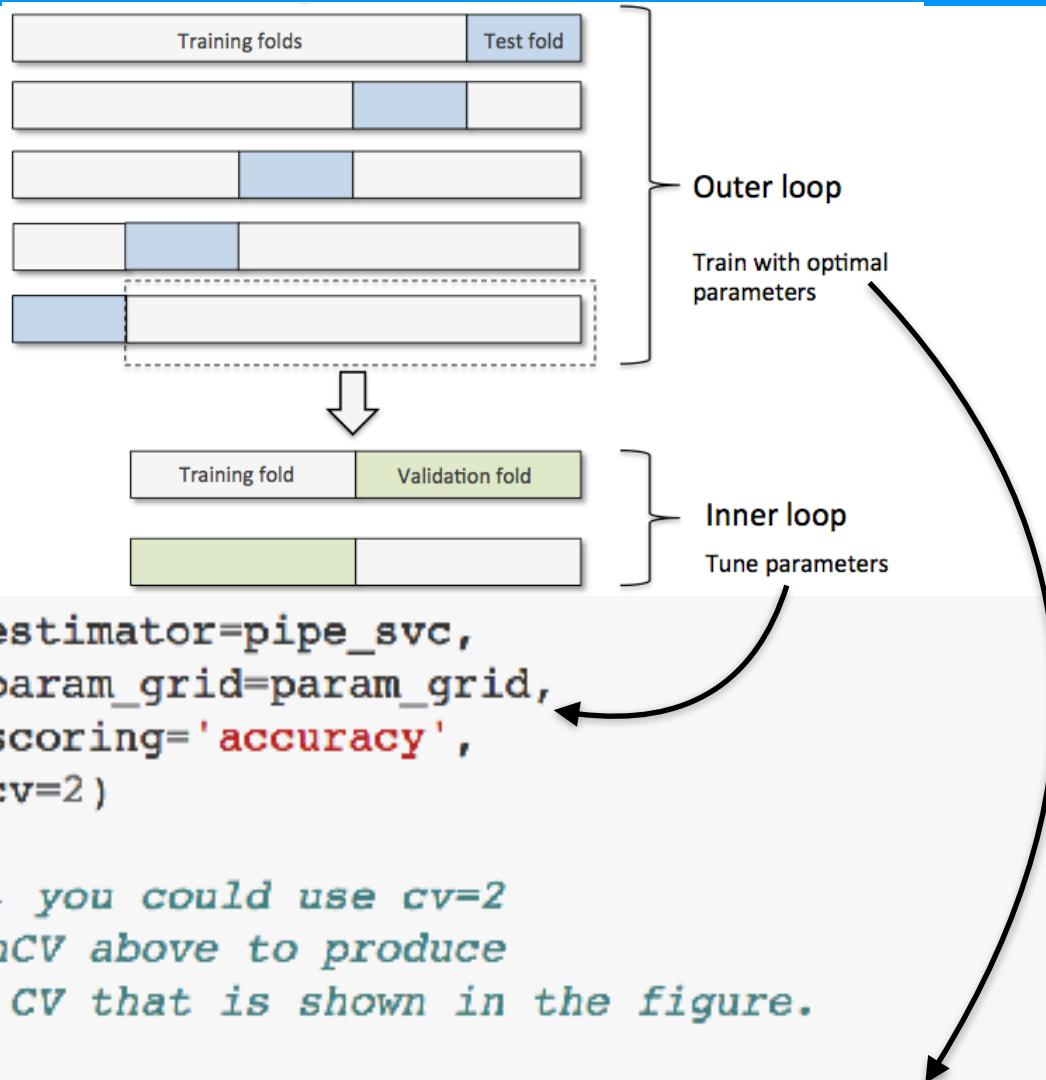
- Using the grid search parameters and testing on the same set...
 - the **performance on the dataset** could now be **biased**
 - cannot realistically determine the **expected performance** on new data



Solution: Nested Cross Validation



Nested Cross Validation: Hyper-parameters



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

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

McNemar and Edwards, 1948

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

chi squared statistic

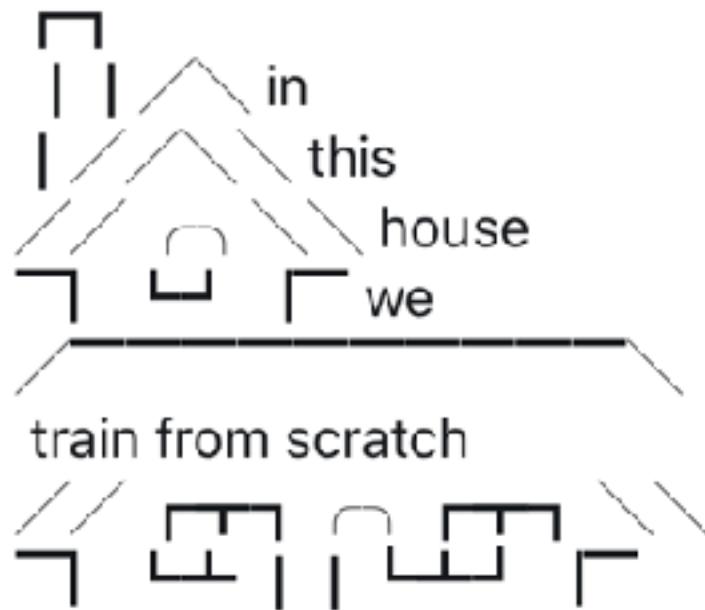
used to judge null hypothesis

Null: models have the same output

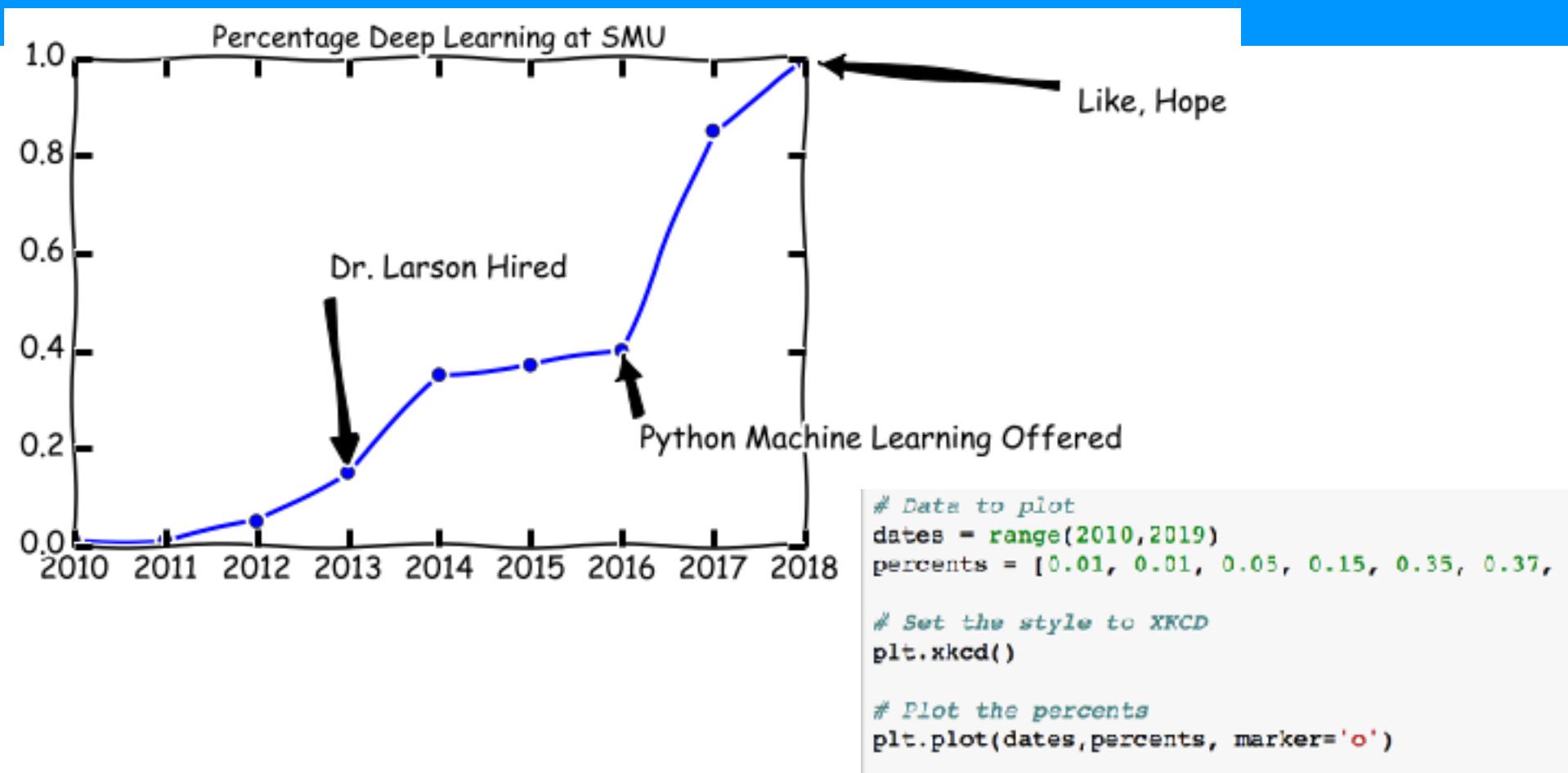
Town Hall



AI Memes for Artificially Intelligent Agents



Some History of 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



1942-2011

Geoffrey Hinton



History of Deep Learning: Winter

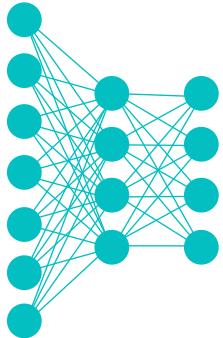
- Up to this point: back propagation saved AI winter for NN (Hinton and others!)
- 80's, 90's, 2000's: neural networks for image processing start to get deeper
 - but back propagation no longer efficient for training
 - Back propagation gradient **stagnates** research—can't train **deeper** networks
- 2001: SVMs and Random Forests gain traction...
 - The second AI winter begins, research in NN plummets
 - Funding for and accepted papers that incorporate Neural Networks asymptotically approaches zero

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

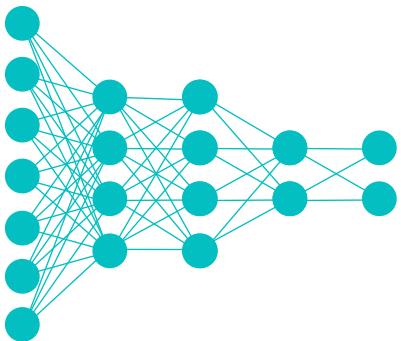
History of Deep Learning: Winter

BRACE YOURSELF

- AI Winter is coming:



Easy to train, performs on par with other methods



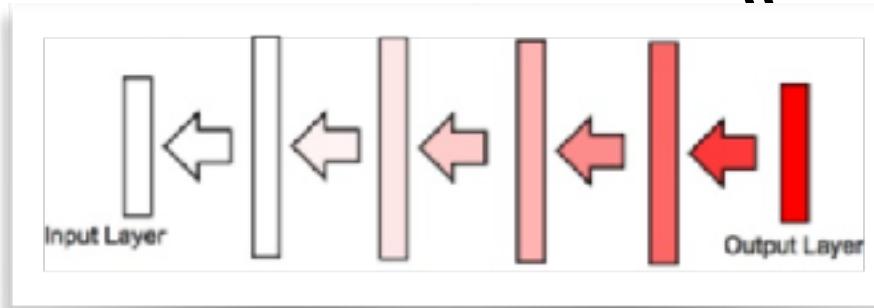
Hard to train, performs worse than other methods

Researchers have difficulty reconciling expressiveness with performance

~chance (untrainable)

AI WINTER IS
COMING

memorization



generalization

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

History of Deep Learning

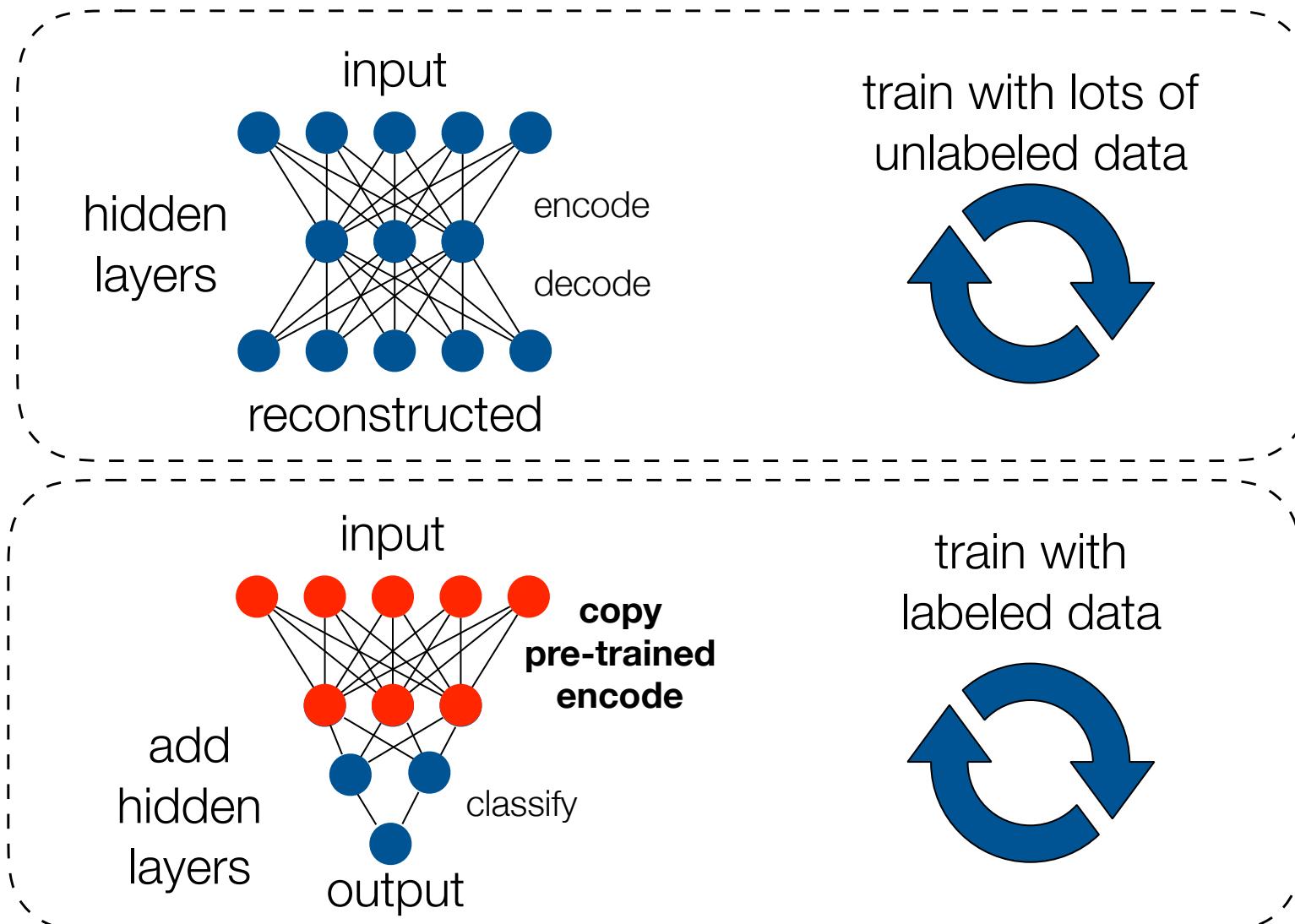
- 2004: Hinton secures funding from CIFAR based on his reputation
 - *eventually*: Canada would be savior for NN
 - Hinton rebrands: Deep Learning
- 2006: Hinton publishes paper on using pre-training and Restricted Boltzmann Machines
- 2007: Another paper: Deep networks are more efficient when pre-trained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

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

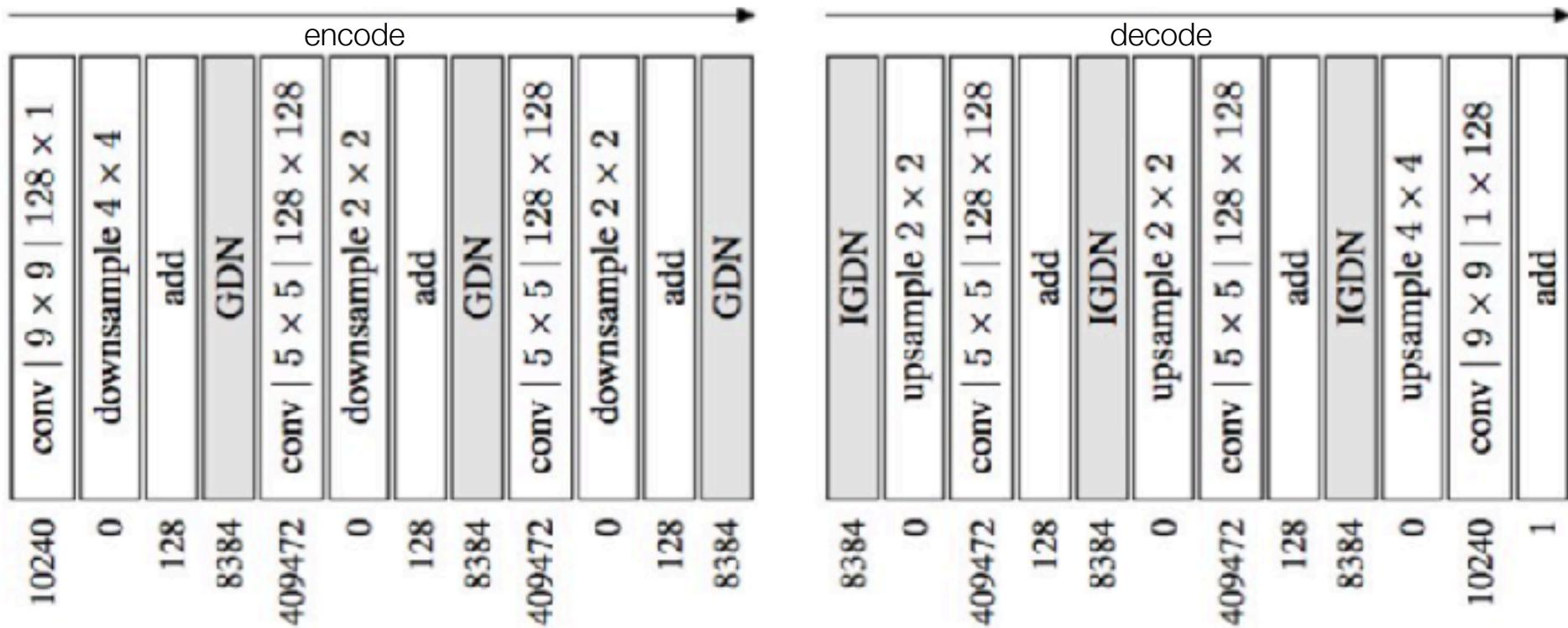
20

Pre-training: still in the long winter

- auto-encoding



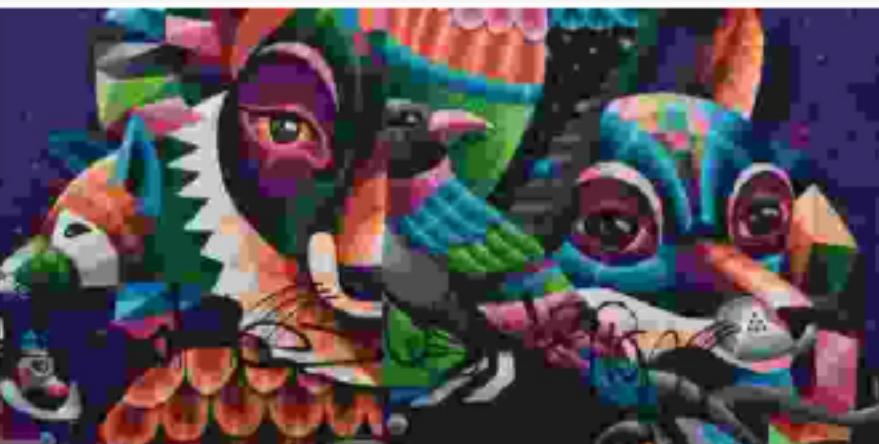
Pre-training: modern example



<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



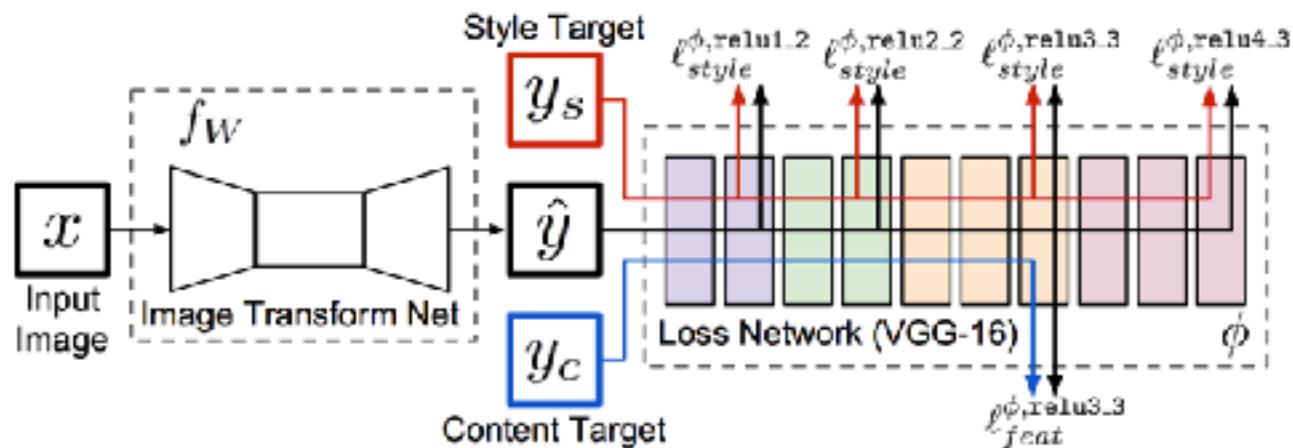
JPEG 2000, 5724 bytes (0.162 bit/px), RMSE: 13.75/7.00, PSNR: 25.36 dB/31.20 dB



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

Getting to basic descriptors of images

- 2004: Hinton secures funding from CIFAR based on his reputation
 - eventually: Canada would be savior for NN
 - 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



Wood, Ledford, Pop 2017

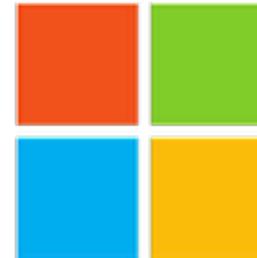
History of Deep Learning

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



Navdeep Jaitly
Google Brain Team

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



George Dahl
Google Brain Team



Abdel-rahman Mohamed

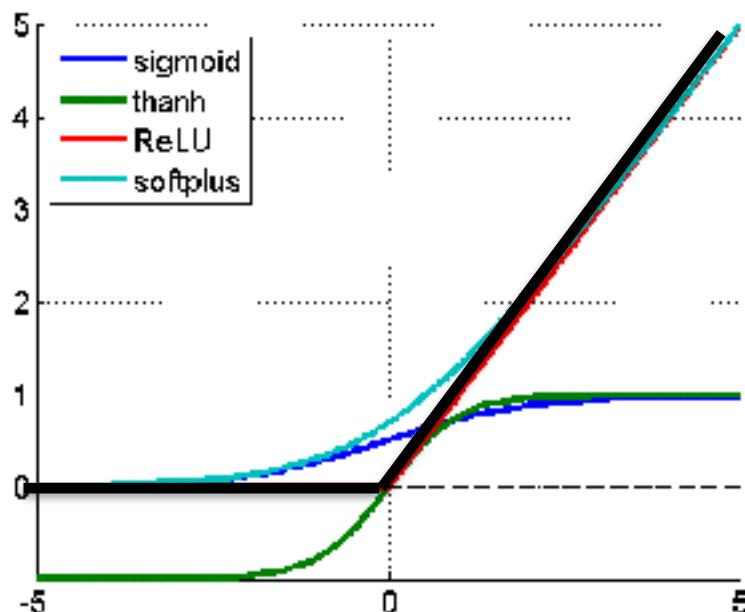
Microsoft Research
Redmond, Washington | Computer Software

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

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

History of Deep Learning

- 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



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

that's a really easy gradient to compute!
...and it makes the weights more sparse
...and helps to solve the vanishing gradient problem
...and its inspired by biological vision community

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

History of Deep Learning

- ReLU not the only way to do it!

2 SWISH SiLU

From Ramachandran P, Zoph B, Le QV. Swish: a Self-Gated Activation Function. arXiv preprint arXiv: 1710.05941. 2017 Oct 16

We propose a new activation function that we name *Swish*:

$$f(x) = x \cdot \sigma(x) \quad (1)$$

where $\sigma(x) = (1 + \exp(-x))^{-1}$ is the sigmoid function. See Figure 1 for the graph of Swish.

*Work done as a member of the Google Brain Residency program (g.co/brainresidency).

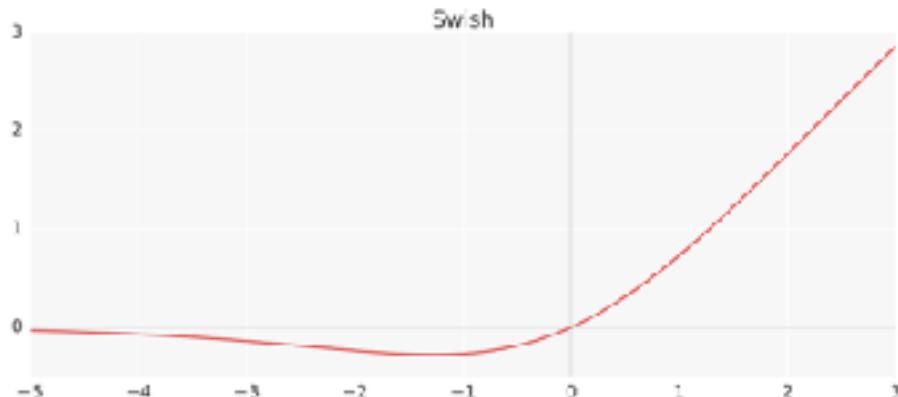
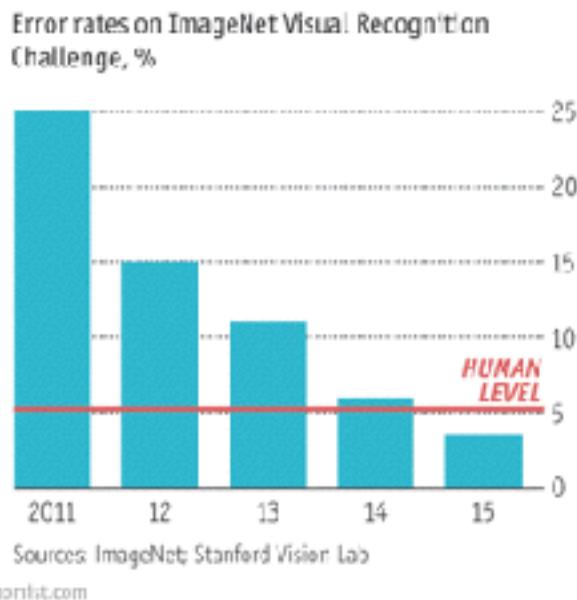


Figure 1: The Swish activation function.

$$\begin{aligned}f'(x) &= \sigma(x) + x \cdot \sigma(x)(1 - \sigma(x)) \\&= \sigma(x) + x \cdot \sigma(x) - x \cdot \sigma(x)^2 \\&= x \cdot \sigma(x) + \sigma(x)(1 - x \cdot \sigma(x)) \\&= f(x) + \sigma(x)(1 - f(x))\end{aligned}$$

History of Deep Learning

- 2012: 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
Former
Director of Stanford's
AI Lab

I happened to witness this critical juncture in time first hand because the ImageNet challenge was over the last few years organized by Fei-Fei Li's lab (my lab), so I remember when my labmate gasped in disbelief as she noticed the (very strong) ConvNet submission come up in the submission logs. And I remember us pacing around the room trying to digest what had just happened. In the next few months ConvNets went from obscure models that were shrouded in skepticism to rockstars of Computer Vision, present as a core building block in almost every new Computer Vision paper.

History of Deep Learning

- 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)

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

History of Deep Learning

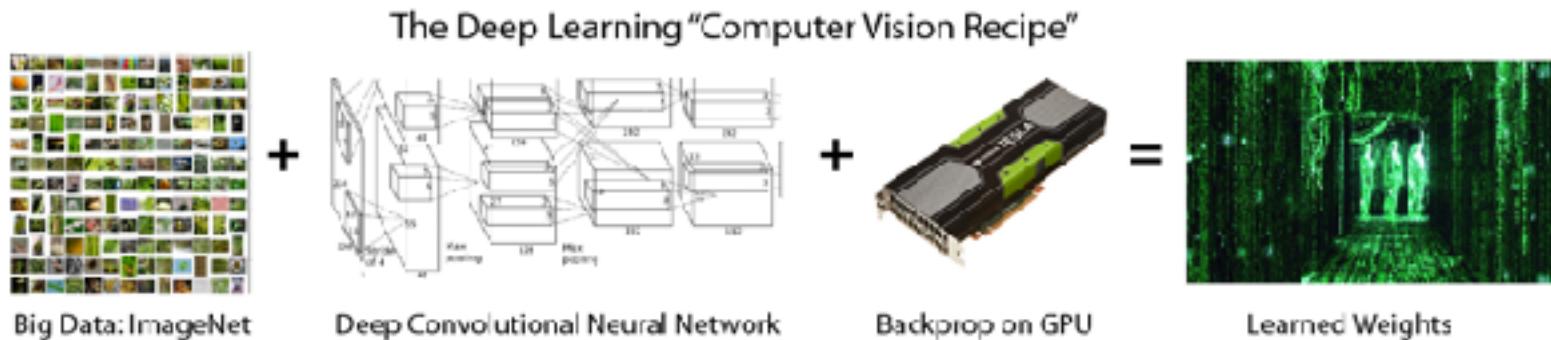
- 2013: Ng and Google (founded BrainTeam)
 - run unsupervised feature creation on YouTube videos (becomes computer vision benchmark)

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.



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 Law:**
 - use a GPU when possible, init weights for consistent gradient magnitude, ReLU/BN/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/>

A summary of the Deep Learning people:



Yoshua
Bengio



Yann
LeCun



Geoffrey
Hinton



FeiFei
Li



Andrew
Ng



Daphne
Koller

Stayed at
University
Advises IBM

Heads
Facebook
AI Team

Google

Chief Scientist,
AI/ML
Google Cloud

Coursera
Baidu
Google

Stanford
Founded Coursera
MacArthur Genius

**Popularized CNNs, RNNs, and
Mitigated Gradient Instability**

**Made Deep Learning
Instruction Accessible**

Read this paper from 2015,
as it sums up advancements
nicely

And predicts the future of
deep learning

REVIEW

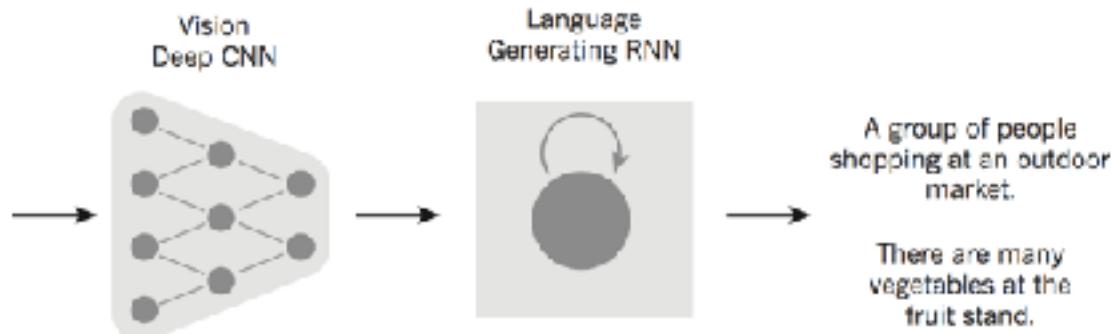
doi:10.1038/nature14539

Deep learning

Yann LeCun^{1,2}, Yoshua Bengio³ & Geoffrey Hinton^{4,5}

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and

Famous examples:



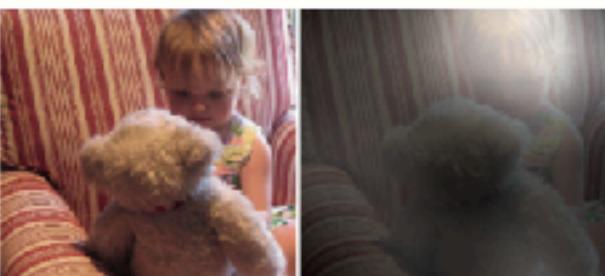
A woman is throwing a **frisbee** in a park.



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.



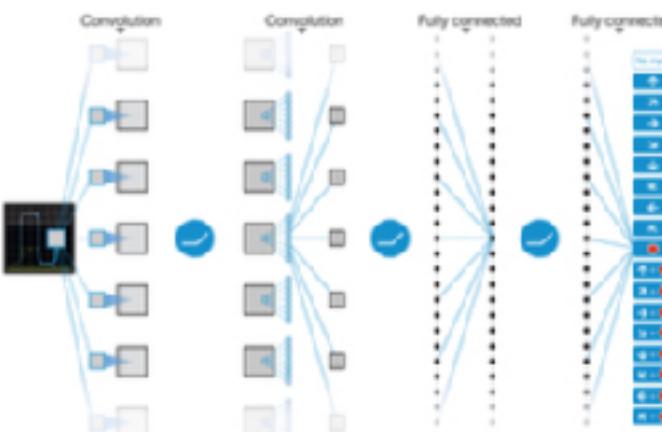
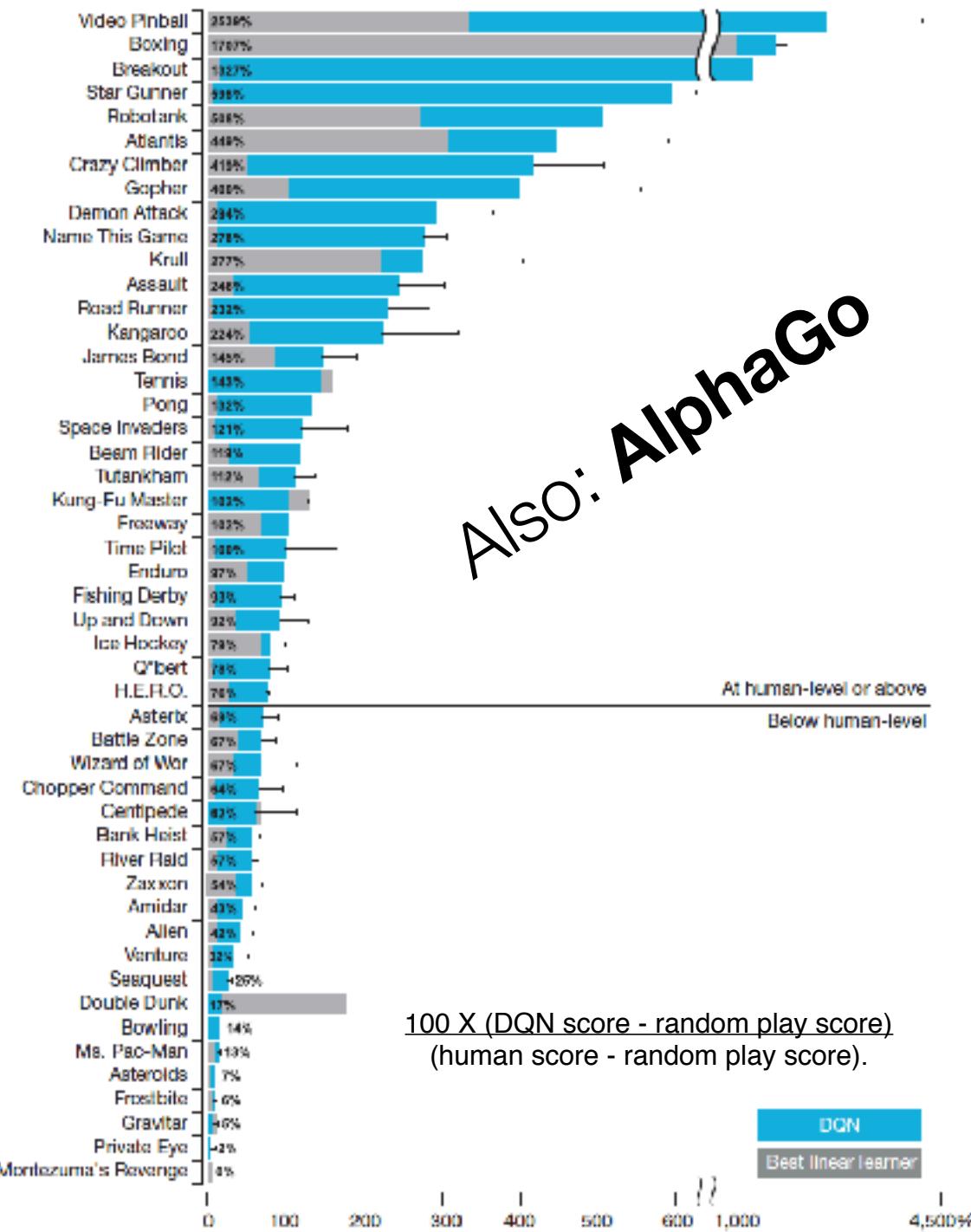
A group of people sitting on a boat in the water.



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

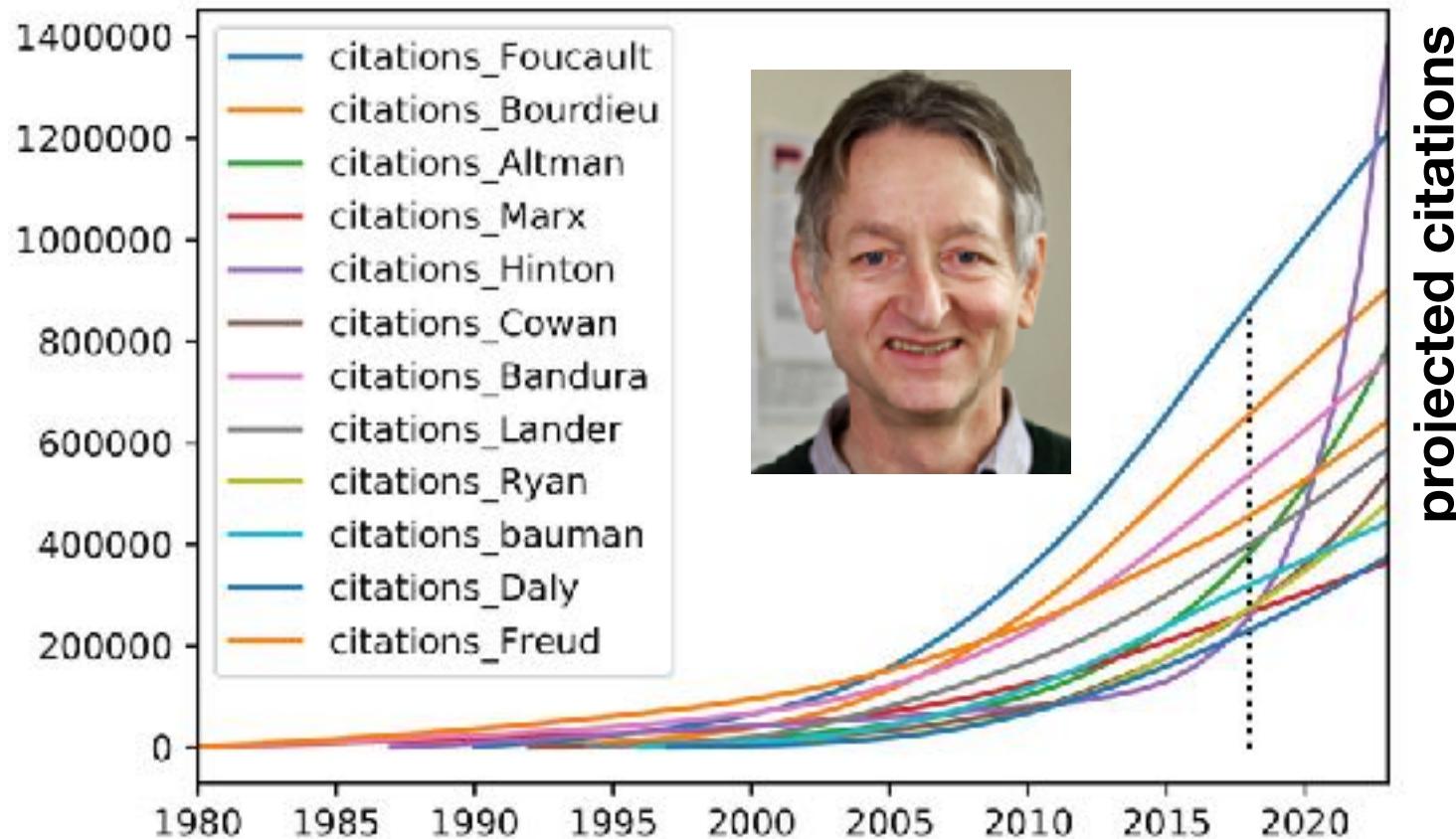
More Famous

Also: AlphaGo



End of Session

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



End of Session

- if time:
 - batch normalization

Backup: History of Deep Learning

- ReLU not the only way to do it!

Input: Values of x over a mini-batch: $B = \{x_1, \dots, x_m\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\begin{aligned}\mu_B &\leftarrow \frac{1}{m} \sum_{i=1}^m x_i && // \text{mini-batch mean} \\ \sigma_B^2 &\leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 && // \text{mini-batch variance} \\ \hat{x}_i &\leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} && // \text{normalize} \\ y_i &\leftarrow \gamma \hat{x}_i + \beta = \text{BN}_{\gamma, \beta}(x_i) && // \text{scale and shift}\end{aligned}$$

Batch Normalization

Normalize input layers per mini-batch and add control parameters, γ and β

- help reduce gradient instability
- differentiable normalization==gradient!
- can be applied to each layer input

$$\frac{\partial \ell}{\partial \hat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_B^2} = \sum_{i=1}^m \frac{\partial \ell}{\partial \hat{x}_i} \cdot (x_i - \mu_B) \cdot \frac{-1}{2} (\sigma_B^2 + \epsilon)^{-3/2}$$

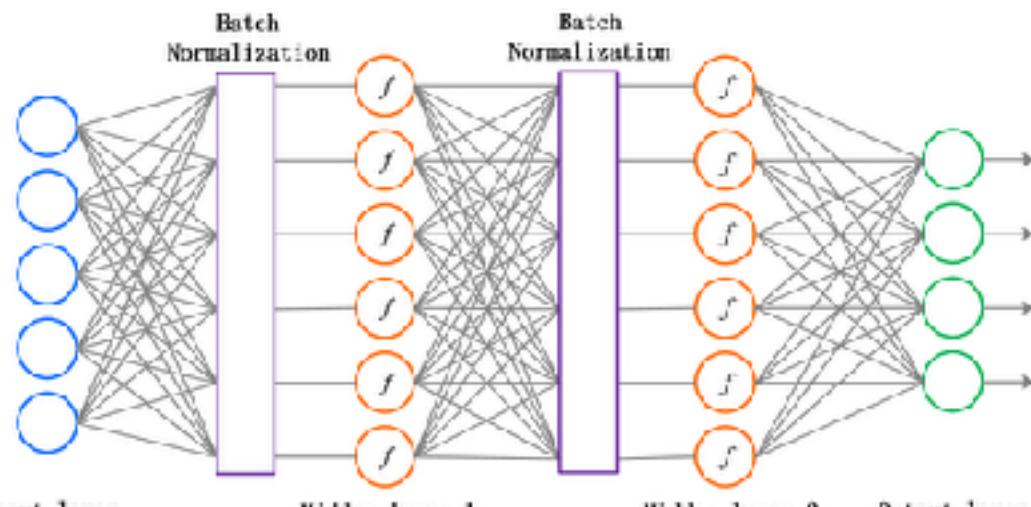
$$\frac{\partial \ell}{\partial \mu_B} = \left(\sum_{i=1}^m \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_B^2 + \epsilon}} \right) + \frac{\partial \ell}{\partial \sigma_B^2} \cdot \frac{\sum_{i=1}^m -2(x_i - \mu_B)}{m}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{1}{\sqrt{\sigma_B^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_B^2} \cdot \frac{2(x_i - \mu_B)}{m} + \frac{\partial \ell}{\partial \mu_B} \cdot \frac{1}{m}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^m \frac{\partial \ell}{\partial y_i} \cdot \hat{x}_i$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^m \frac{\partial \ell}{\partial y_i}$$

From Ioffe and Szegedy (2015), Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift:
<https://arxiv.org/pdf/1502.03167.pdf>



Lecture Notes for **Machine Learning in Python**

Professor Eric Larson
Wide and Deep Networks

Lecture Agenda

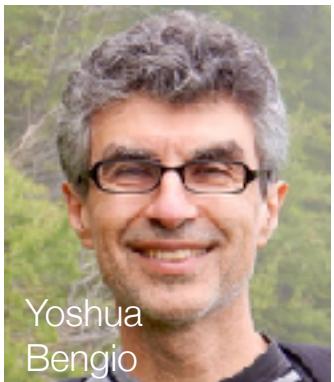
- Logistics: CS 8321 in Spring
- Introduction to TensorFlow
 - Tensors, Namespaces, Numerical methods
 - Deep APIs
- Wide and Deep Networks

Last Time!

- 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 (not called neural networks)
- 2006: Auto-encoding and Restricted Boltzmann Machines
- 2007: Deep networks are more efficient when pre-trained
 - RBMs not really the important part
- 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 skyrockets 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, BP, SiLU)

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

What else did we talk about last time?



Yoshua
Bengio

Stayed at
University, Montreal
Advises IBM



Yann
LeCun

Heads
Facebook
AI Team,
NYU



Geoffrey
Hinton

Toronto
GoogleBrain



FeiFei
Li

Chief Scientist, AI/ML
Google Cloud
Director for Human AI at Stanford



Daphne
Koller

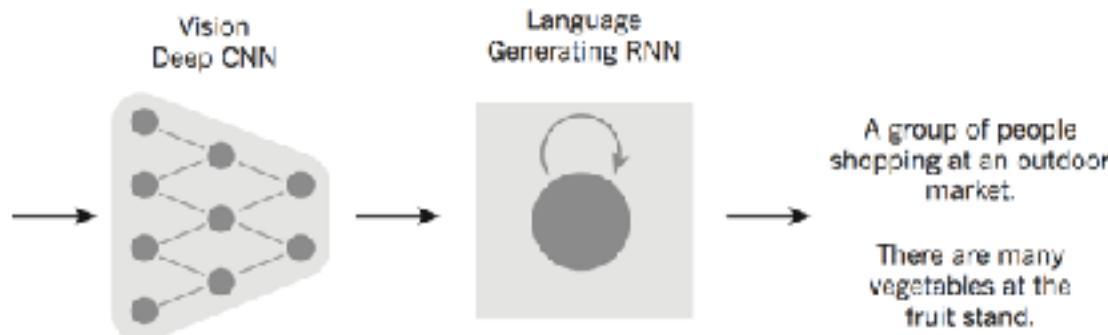
Stanford
Founded Coursera
MacArthur Genius



Andrew
Ng

Stanford
Co-Founded Coursera
Baidu, GoogleBrain

Famous examples:



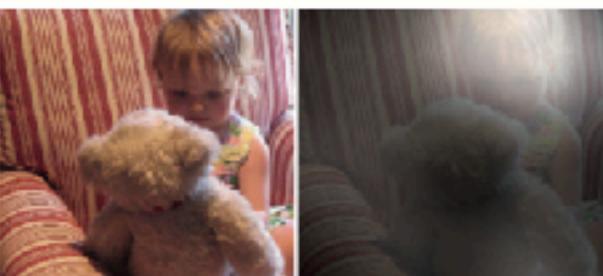
A woman is throwing a **frisbee** in a park.



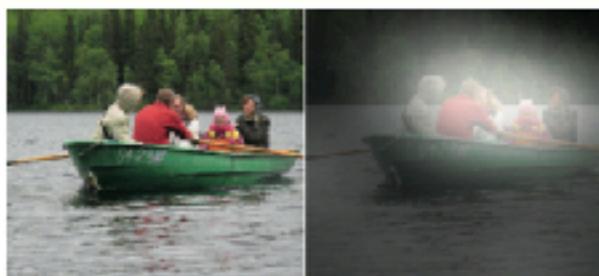
A dog is standing on a hardwood floor.



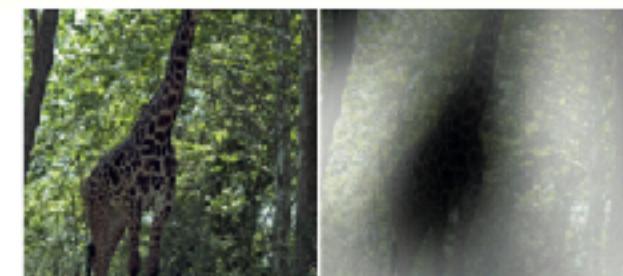
A **stop** sign is on a road with a mountain in the **background**.



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



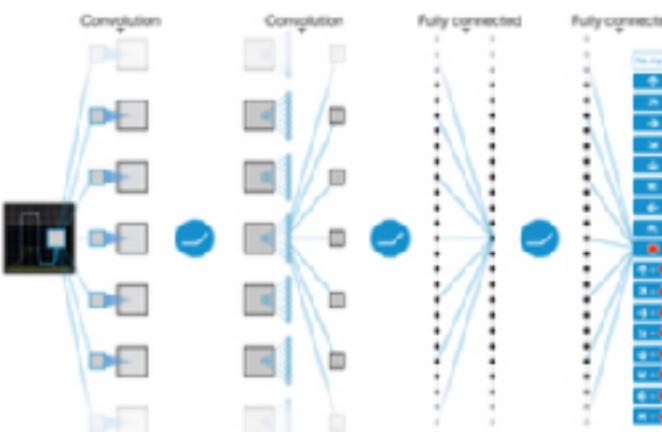
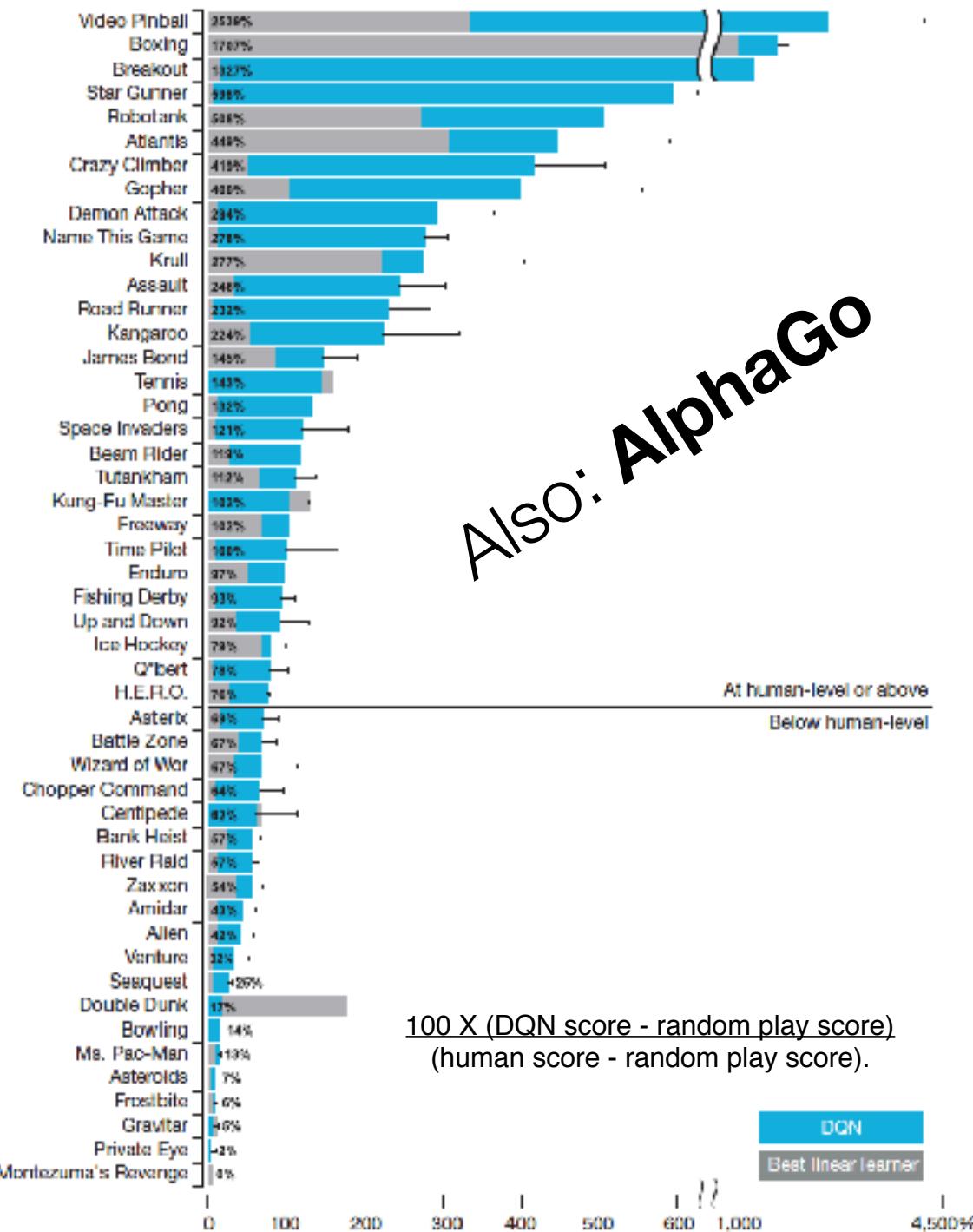
A group of people sitting on a boat in the water.



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

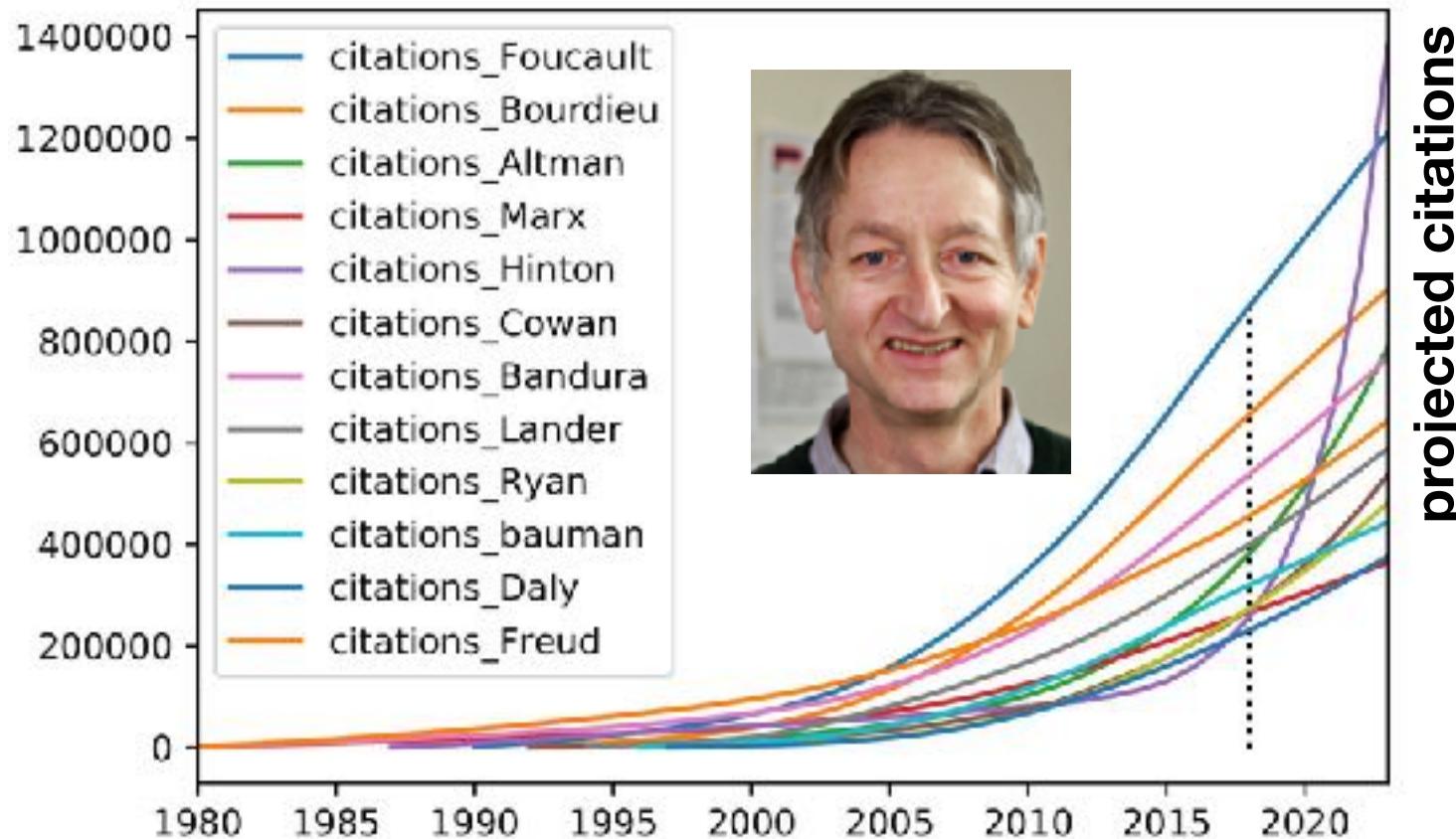
More Famous

Also: AlphaGo



End of Session

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



TensorFlow

"Further discussion of it merely incumbers the literature and befogs the mind of fellow students."

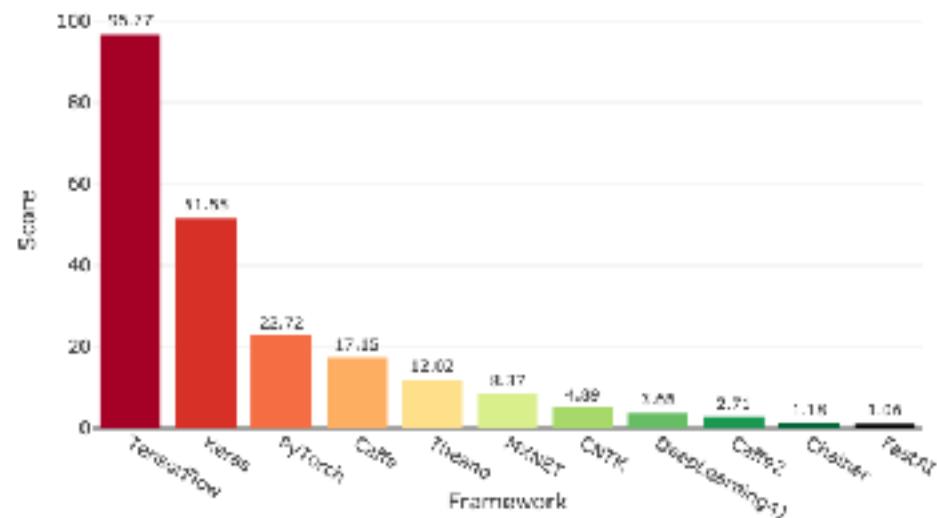
- 2007: NIPS program committee rejects a paper on deep learning by *al. et.* Hinton because they already accepted a paper on deep learning and two papers on the same topic would be excessive.
- ~2009: A reviewer tells Yoshua Bengio that papers about neural nets have no place in ICML.
- ~2010: A CVPR reviewer rejects Yann LeCun's paper even though it beats the state-of-the-art. The reviewer says that it tells us nothing about computer vision because everything is learned.



Options for Deep Learning Toolkits

- Caffe/Caffe2
- TensorFlow
- PyTorch
- MxNet
- CNTK
- [Theano]
- DL4J
- Fast.AI

Deep Learning Framework Power Scores 2018



| Framework | Online Job Listings | | | | | KDnuggets | Google | Medium | Amazon | ArXiv | GitHub Activity | | | | | |
|----------------|---------------------|---------|--------------|----------|------------|-----------|--------|--------|--------|-------|-----------------|-------|----------|-------|----------|-------|
| | Indeed | Monster | Simply Hired | LinkedIn | Angel List | | | | | | Articles | Books | Articles | Stars | Watchers | Forks |
| TensorFlow | 2,079 | 1,253 | 1,582 | 2,810 | 652 | 29.90% | 73 | 6,200 | 202 | 3,700 | 109,578 | 8,384 | 67,551 | 1,642 | | |
| Keras | 884 | 364 | 449 | 895 | 177 | 22.20% | 53 | 9,120 | 79 | 1,390 | 33,558 | 1,847 | 12,656 | 719 | | |
| PyTorch | 486 | 309 | 428 | 885 | 120 | 8.40% | 19 | 1,780 | 16 | 1,580 | 18,718 | 952 | 4,474 | 780 | | |
| Caffe | 607 | 399 | 615 | 866 | 123 | 1.50% | 4 | 815 | 14 | 1,360 | 25,604 | 2,218 | 15,633 | 270 | | |
| Theano | 356 | 316 | 279 | 508 | 95 | 4.90% | 0 | 428 | 17 | 652 | 8,477 | 585 | 2,447 | 328 | | |
| MXNET | 266 | 154 | 200 | 298 | 29 | 1.50% | 2 | 624 | 32 | 260 | 15,200 | 1,170 | 6,496 | 687 | | |
| CNTK | 126 | 96 | 97 | 160 | 12 | 3.00% | 0 | 223 | 1 | 88 | 15,106 | 1,368 | 4,029 | 189 | | |
| DeepLearning4J | 17 | 6 | 9 | 35 | 3 | 3.40% | 2 | 70 | 11 | 27 | 9,615 | 829 | 4,441 | 232 | | |
| Caffe2 | 55 | 51 | 49 | 109 | 12 | 1.20% | 2 | 335 | 2 | 67 | 8,284 | 577 | 2,102 | 193 | | |
| Chainer | 19 | 19 | 19 | 28 | 3 | 0.00% | 2 | 91 | 3 | 164 | 4,128 | 325 | 1,095 | 182 | | |
| FastAI | 0 | 0 | 0 | 0 | 0 | 0.00% | 0 | 858 | 0 | 11 | 7,268 | 432 | 2,647 | 196 | | |

Programmatic creation

- Most toolkits use python to build a computation graph of operations
 - Build up computations
 - Execute computations
- **Support:**
 - tensor creation
 - functions on tensors
 - automatic derivative computation

Tensorflow

- Open sourced library from Google
- Second generation release from Google Brain
 - supported for Linux, Unix, Windows
 - Also works on Android/iOS
- Released November 9th, 2015
 - (this class first offered January 2016)

Tensors

- Tensors are just multidimensional arrays
 - like in Numpy
 - typically, for NN,
 - scalars (biases and constants)
 - vectors (e.g., input arrays)
 - 2D matrices (e.g., images)
 - 3D matrices (e.g., color images)
 - 4D matrices (e.g., batches of color images)

Tensor neural network functions

- Easy to define operations on layers of networks
 - `relu(features, name=None)`
 - `bias_add(value, bias, data_format=None, name=None)`
 - `sigmoid(x, name=None)`
 - `tanh(x, name=None)`
 - `conv2d(input, filter, strides, padding)`
 - `conv1d(value, filters, stride, padding)`
 - `conv3d(input, filter, strides, padding)`
 - `conv3d_transpose(value, filter, output_shape, strides)`
 - `sigmoid_cross_entropy_with_logits(logits, targets)`
 - `softmax(logits, dim=-1)`
 - `log_softmax(logits, dim=-1)`
 - `softmax_cross_entropy_with_logits(logits, labels, dim=-1)`
- Each function creates layers easily, *knows its gradient*
- **Automatic Differentiation** is just **chain rule**
- But... lets start simple...

Tensor basic functions

- Easy to define operations on tensors

```
a = tf.constant(5.0)
b = tf.constant(6.0)
c = a * b
```

| Numpy | TensorFlow |
|---|---|
| a = np.zeros((2,2)); b = np.ones((2,2)) | a = tf.zeros((2,2)), b = tf.ones((2,2)) |
| np.sum(b, axis=1) | tf.reduce_sum(a, reduction_indices=[1]) |
| a.shape | a.get_shape() |
| np.reshape(a, (1,4)) | tf.reshape(a, (1,4)) |
| b * 5 + 1 | b * 5 + 1 |
| np.dot(a,b) | tf.matmul(a, b) |
| a[0,0], a[:,0], a[0,:] | a[0,0], a[:,0], a[0,:] |

- Also supports convolution: `tf.nn.conv2d`, `tf.nn.conv3D`

Tensor function evaluation

- Easy to define operations on tensors
- Nothing evaluated until you define a session and tell it to evaluate it
- Session defines configuration of execution
 - like GPU versus CPU

```
a = tf.constant(5.0)
```

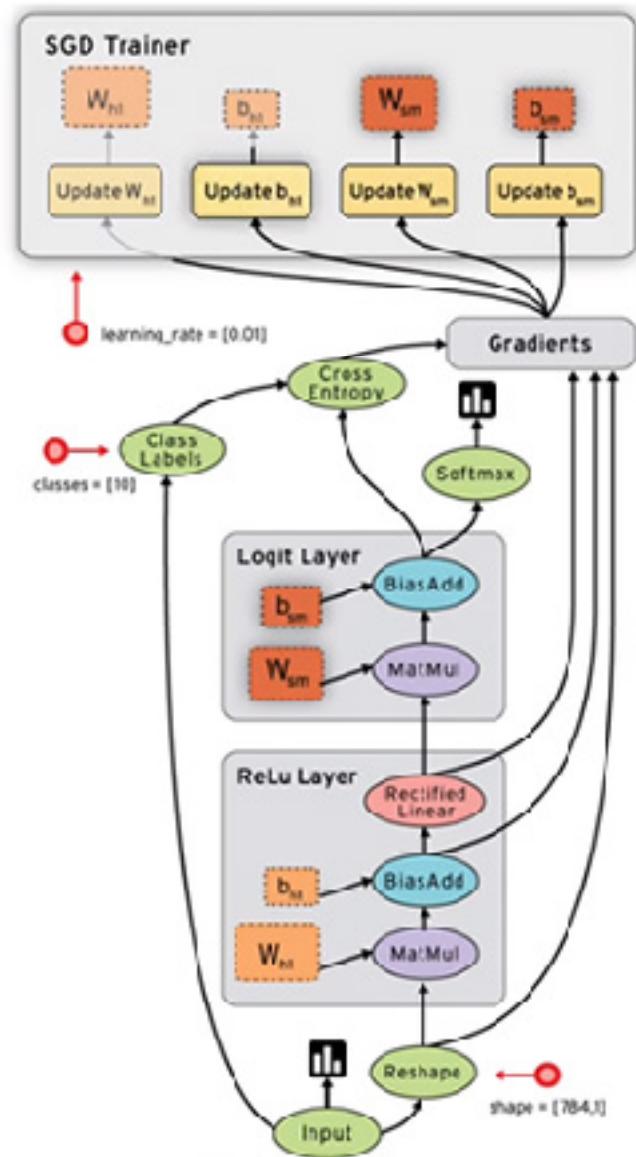
```
b = tf.constant(6.0)
```

```
c = a * b
```

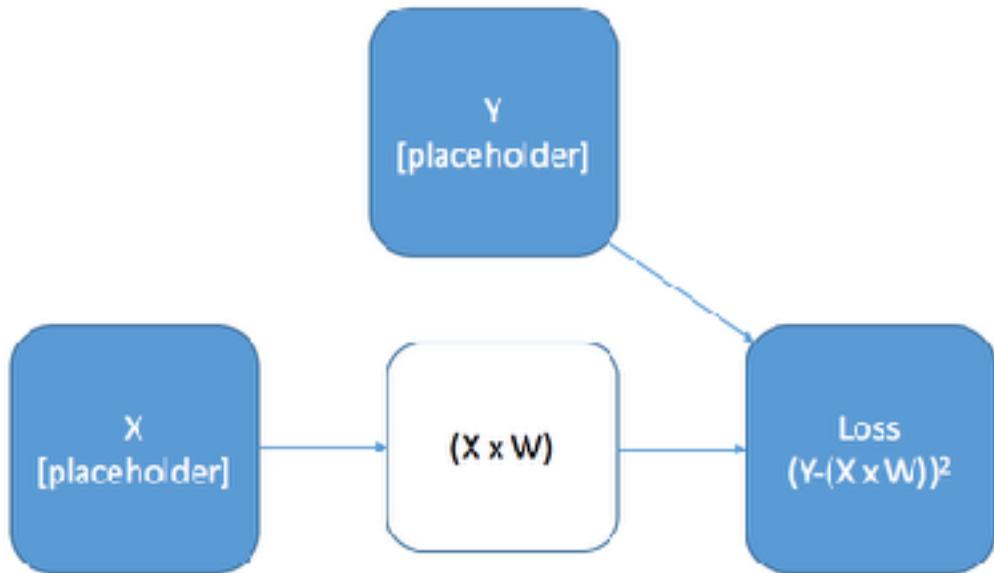
```
with tf.Session() as sess:  
    print(sess.run(c))  
    print(c.eval())
```

output = 30

Computation Graph



- Nothing evaluated until you define a session and tell it to evaluate it
- Session defines configuration of execution
 - like GPU versus CPU



<http://www.kdnuggets.com/2016/07/multi-task-learning-tensorflow-part-1.html>

<http://www.datasciencecentral.com/profiles/blogs/google-open-source-tensorflow>

Tensorflow with Linear Regression

- Simple Computation Graph

$$J(W, b) = \frac{1}{N} \sum_{i=1}^N (y_i - (Wx_i + b))^2$$

```
W = tf.get_variable("weights", (1, 1),
                     initializer=tf.random_normal_initializer())
b = tf.get_variable("bias", (1,)),
                     initializer=tf.constant_initializer(0.0))
y_pred = tf.matmul(X, W) + b
loss = tf.reduce_sum((y - y_pred)**2/n_samples)

opt = tf.train.AdamOptimizer()
opt_operation = opt.minimize(loss)

with tf.Session() as sess:
    sess.run(tf.initialize_all_variables())
    sess.run([opt_operation], feed_dict={X: X_data, y: y_data})
```

Tensor Mini-batching?

```
opt = tf.train.AdamOptimizer()
opt_operation = opt.minimize(loss)

with tf.Session() as sess:
    # Initialize Variables in graph
    sess.run(tf.initialize_all_variables())
    # Gradient descent loop for 500 steps
    for _ in range(500):
        # Select random minibatch
        indices = np.random.choice(n_samples, batch_size)
        X_batch, y_batch = X_data[indices], y_data[indices]
        # Do gradient descent step
        _, loss_val = sess.run([opt_operation, loss], feed_dict={X: X_batch, y: y_batch})
```

Tensor-flow Simplification

- **Self Test:** Can the syntax be simplified?
 - (A) **Yes**, we could write a generic mini-batch optimization computation graph, then use it for arbitrary inputs
 - (B) **Yes**, but we lose control over the optimization procedures
 - (C) **Yes**, but we lose control over the NN models that we can create via Tensorflow
 - (D) **Yes**, and Dr. Larson is going to make us write it ourselves

Keras Programming Interfaces

- Keras Sequential API
 - Great for very simple, single feed forward models
- Keras Functional API
 - build models through series of nested functions
 - each “function” represents an operation in the NN
- Keras Classes (Inheritance)
 - good for more advanced functionality

10. Keras Wide and Deep.ipynb

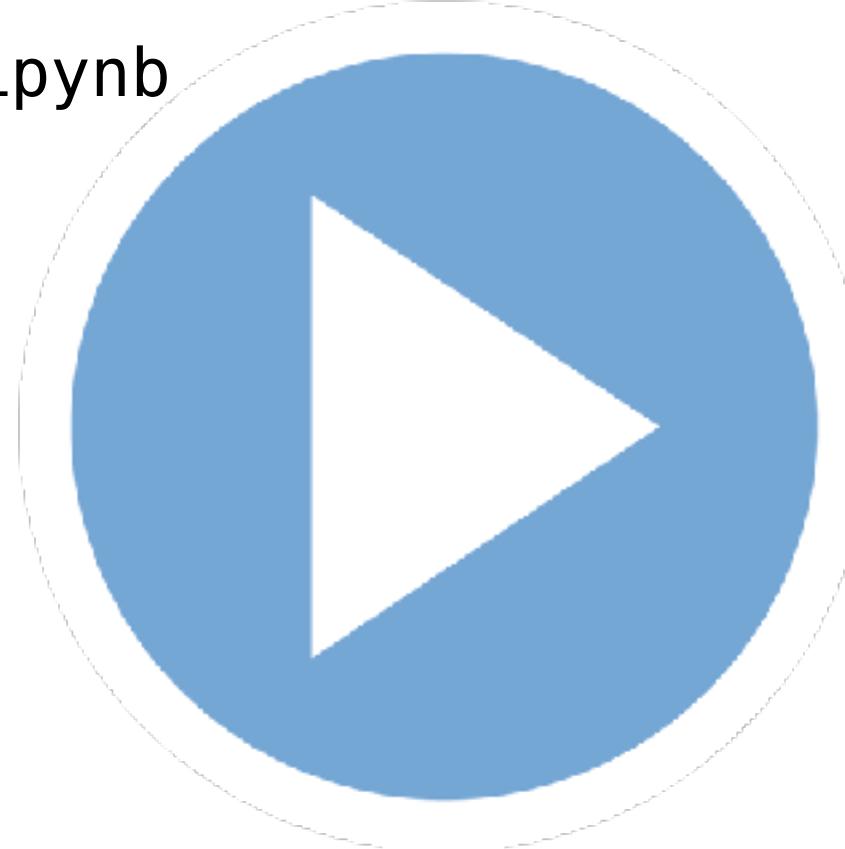
Reinventing the MLP
Wheel

Other tutorials:

<https://github.com/jtoy/awesome-tensorflow>

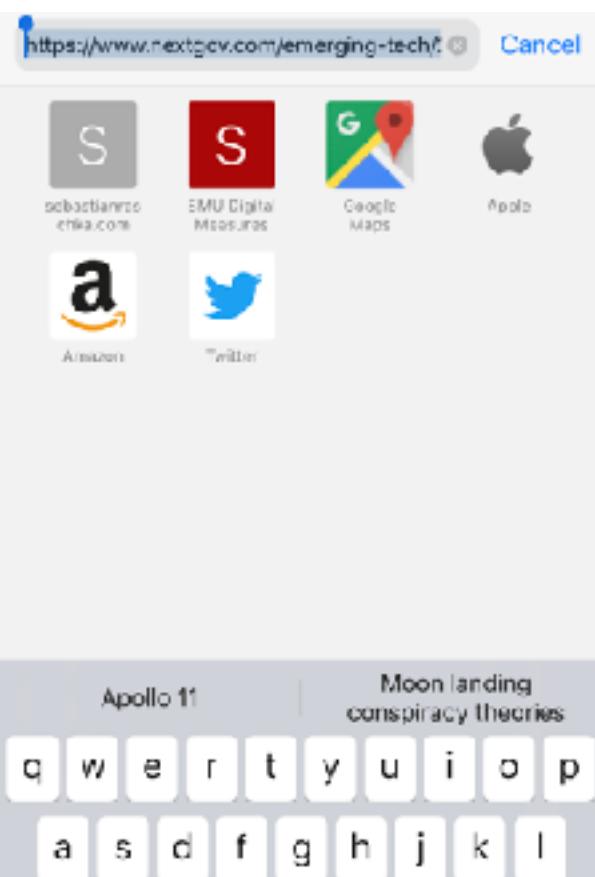
<https://elitedatascience.com/keras-tutorial-deep-learning-in-python>

Or do a Google search!!! They are everywhere!!!



Make me slow down if I go too fast!!

Wide and Deep Networks



Wide and Deep

Wide & Deep Learning for Recommender Systems

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra,
Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil,
Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, Hemal Shah
* Google Inc.

ABSTRACT

Generalized linear models with nonlinear feature transfor-

have never or rarely occurred in the past. Recommendations based on memorization are usually more topical and

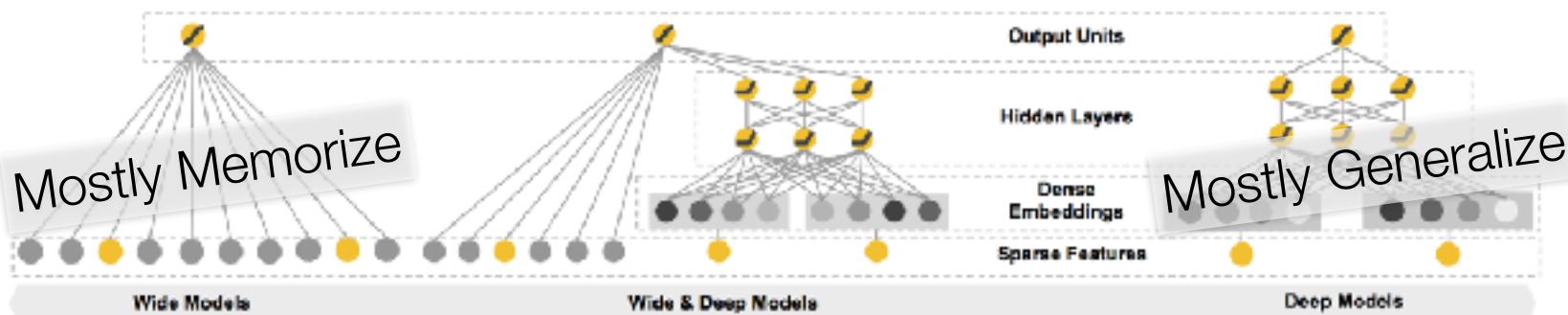
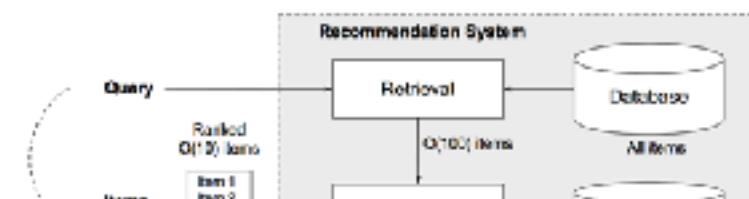


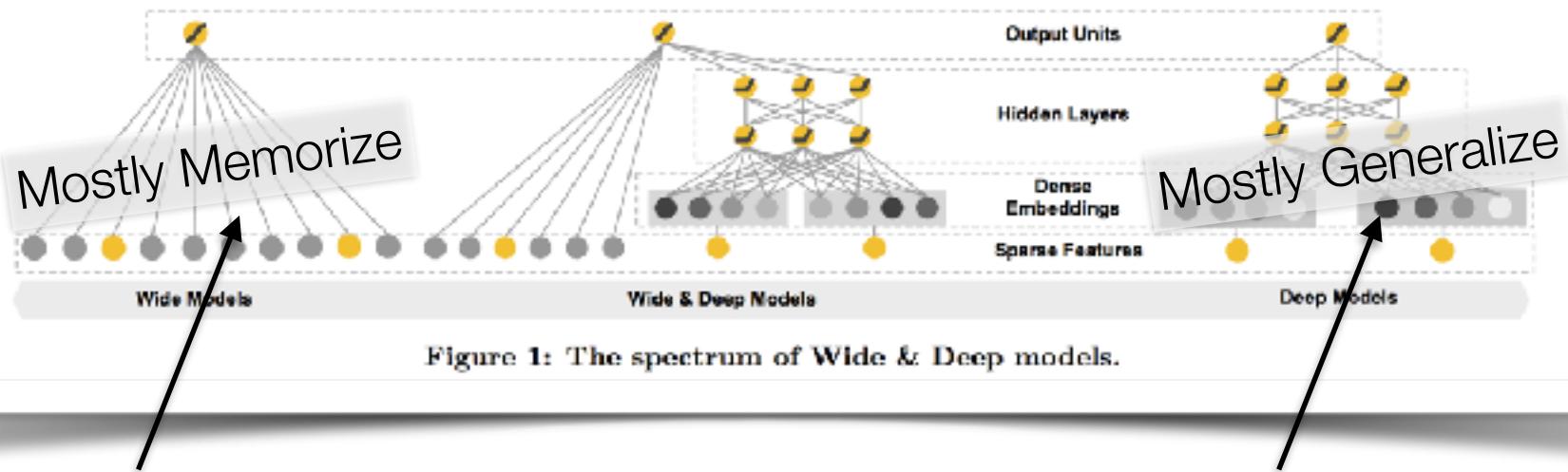
Figure 1: The spectrum of Wide & Deep models.

linear model with feature transformations for generic recommender systems with sparse inputs.

- The implementation and evaluation of the Wide & Deep recommender system productionized on Google



Why wide and deep?



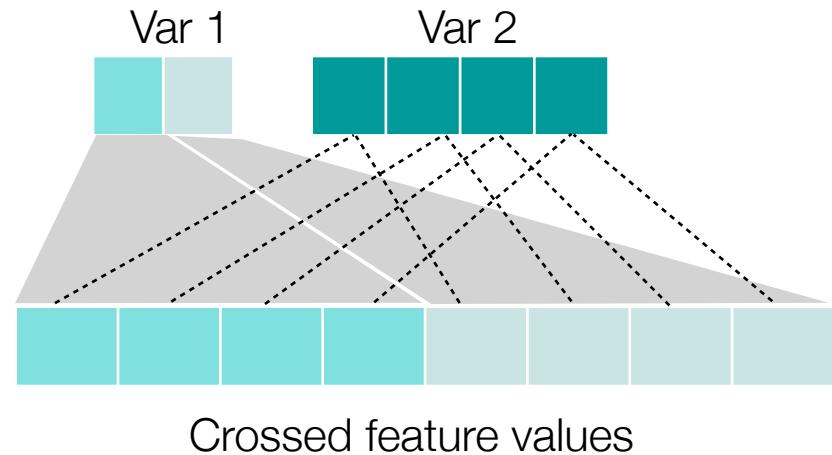
But why memorize?

Obvious!

- Categorical values have combinations that repeat!
 - so memorizing these values is not necessarily a bad strategy
 - let's make memorizing easy on one network

Wide networks (Memorize?)

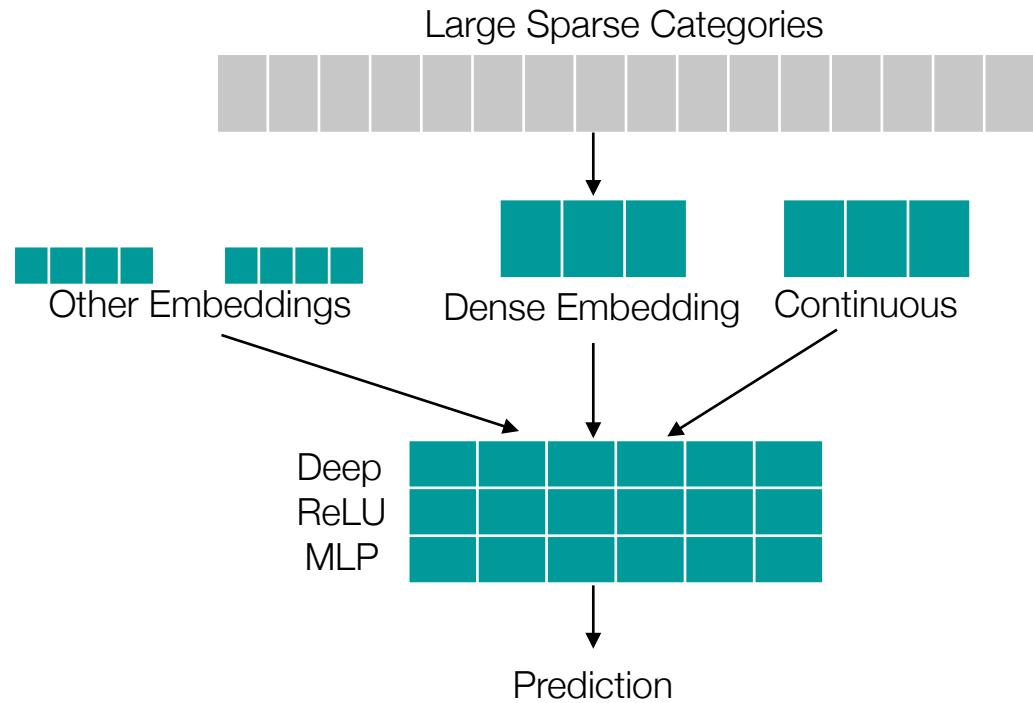
- Wide refers to the expansion of features set
- Crossed feature columns of categorical features
 - Movie Rating
 - G
 - PG
 - PG-13
 - R
 - Else
 - Movie Genre
 - Action
 - Drama
 - Comedy
 - Horror
 - Else
- Crossed feature “Rating-Genre”
 - G-Action, G-Drama, G-Comedy, G-Horror, G-else
 - PG-Action, PG-Drama, PG-Comedy, PG-Horror, G-else
 - and so on ... one hot encoded



Sparse Embedding and Deep MLP (Generalize?)

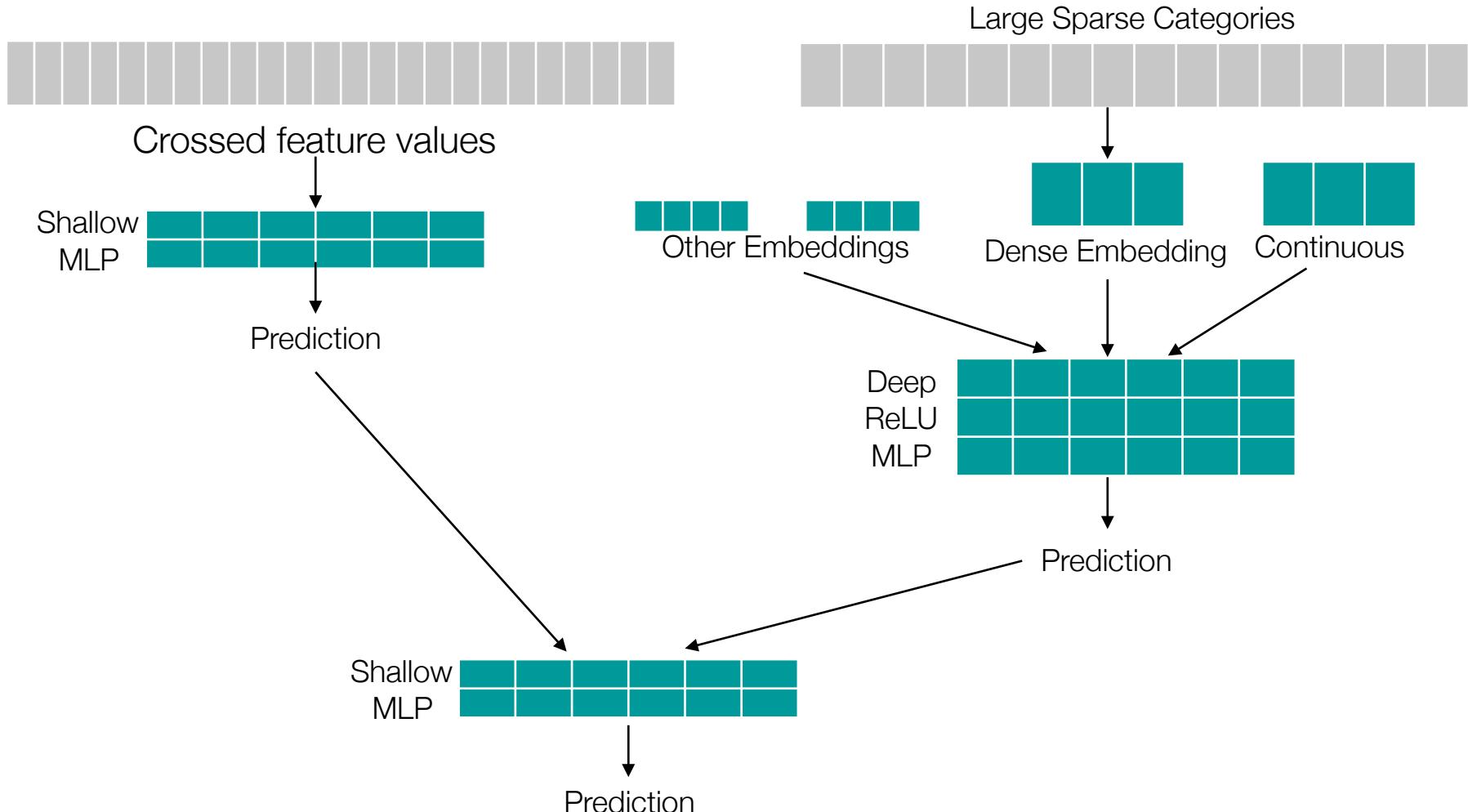
- Deep refers to increasingly smaller hidden layers
- Embed into sparse representations via ReLU

- Movie Actors
 - Armand Assante
 - Meryl Streep
 - Danny Trejo
 - Kevin Bacon
 - Audrey Hepburn
 - ...



Combining Memorization and Generalization

- Deep refers to increasingly smaller hidden layers
- Embed into sparse representations via ReLU

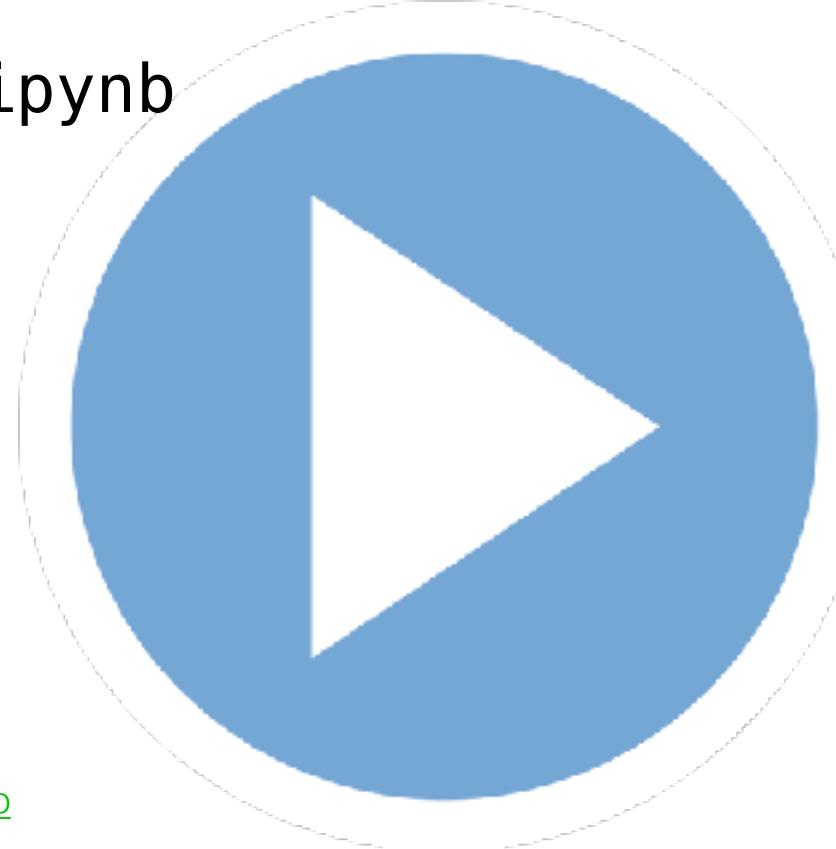


10. Keras Wide and Deep.ipynb

The awful dataset:
Toy Census Data Example

Other tutorials:

https://www.tensorflow.org/tutorials/wide_and_deep



End of Session

- Next Time:
 - Town Hall
 - **Convolutional Neural Networks**