TTIC 31230, Fundamentals of Deep Learning

David McAllester

mcallester@ttic.edu

TAs: Hai Wang and Jialei Wang

Winter 2018

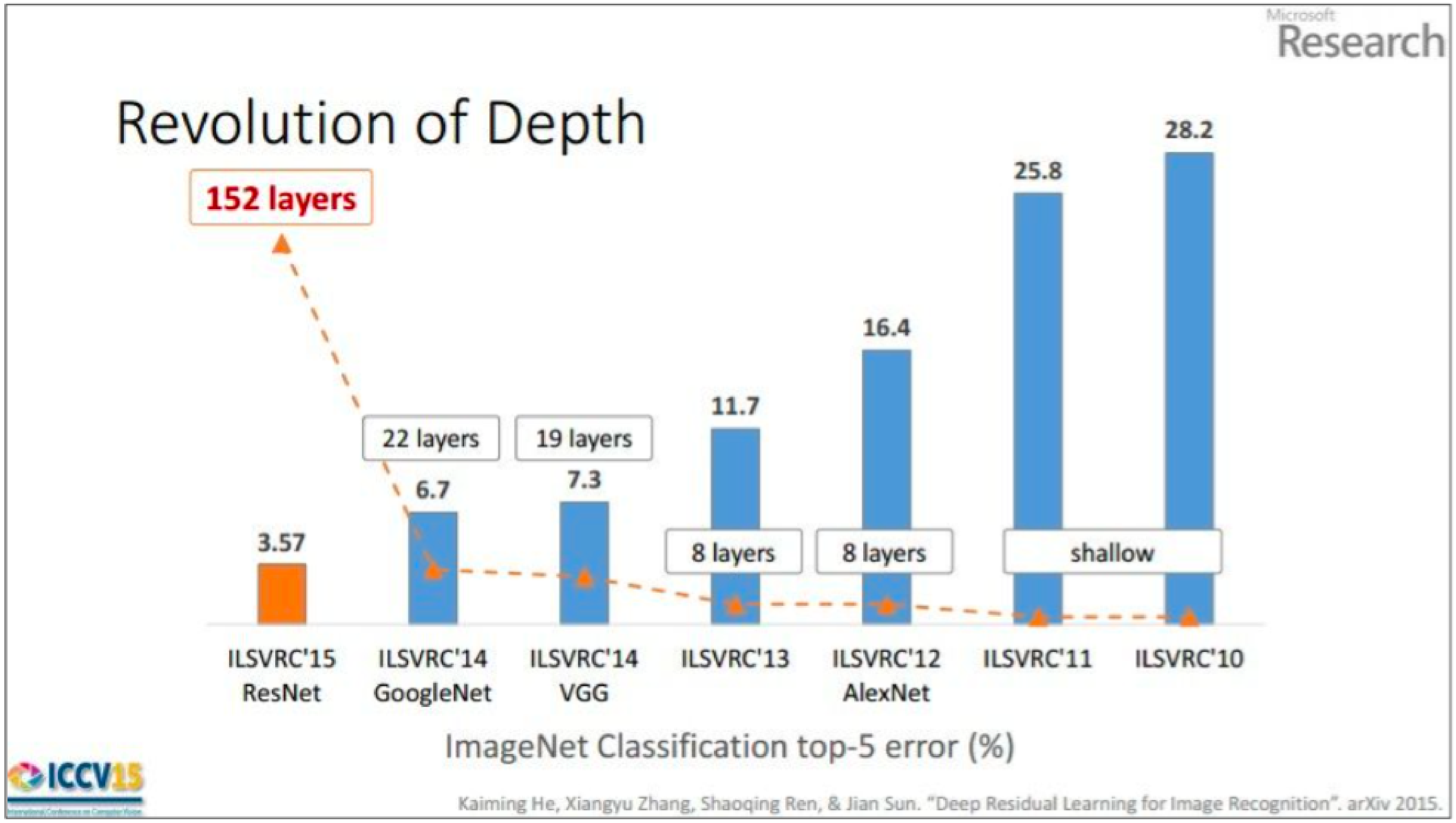
Deep Learning: A Moore's Law of AI?

PASCAL VOC Object Detection

	bicycle	bus	car	motorbike	person	20 class average
2007	36.9	23.2	34.6	27.6	21.3	17.1
2008	42.0	23.2	32.0	38.6	42.0	22.9
2009	46.8	43.8	37.2	42.0	41.5	27.9
2010	54.3	54.2	49.1	51.6	47.5	36.8
2011	58.1	57.6	54.4	58.3	51.6	40.9
2012	54.5	57.1	49.3	59.4	46.1	41.1
2013 DNN	56.3	51.4	48.7	59.8	44.4	43.2
2014 DNN						63.8
2015 ResNet	88.4	86.3	87.8	89.6	90.9	83.8
2016 ResNet						86

Imagenet Classification

1000 kinds of objects.



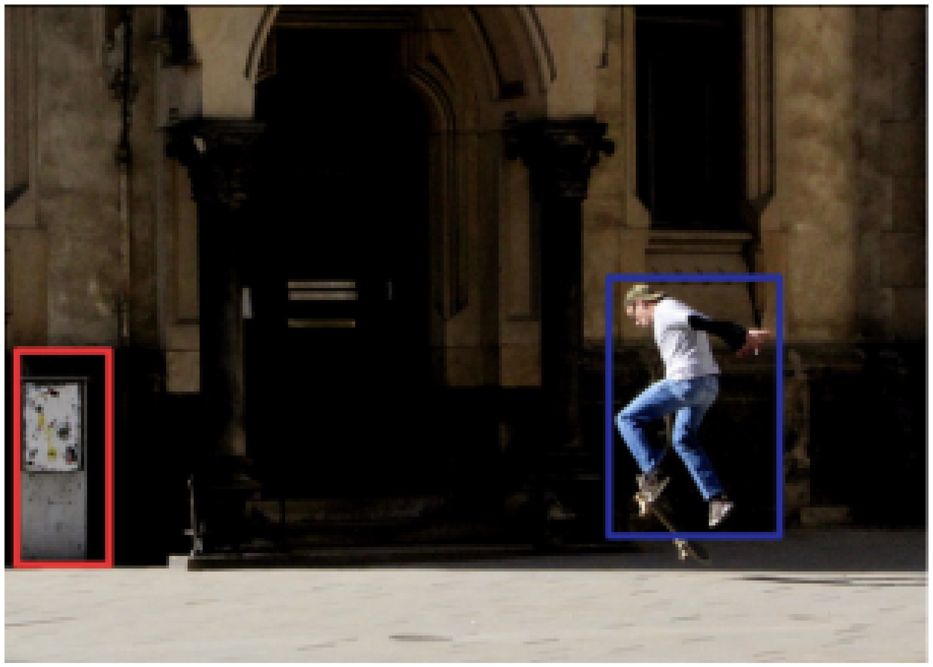
(slide from Kaiming He's recent presentation)

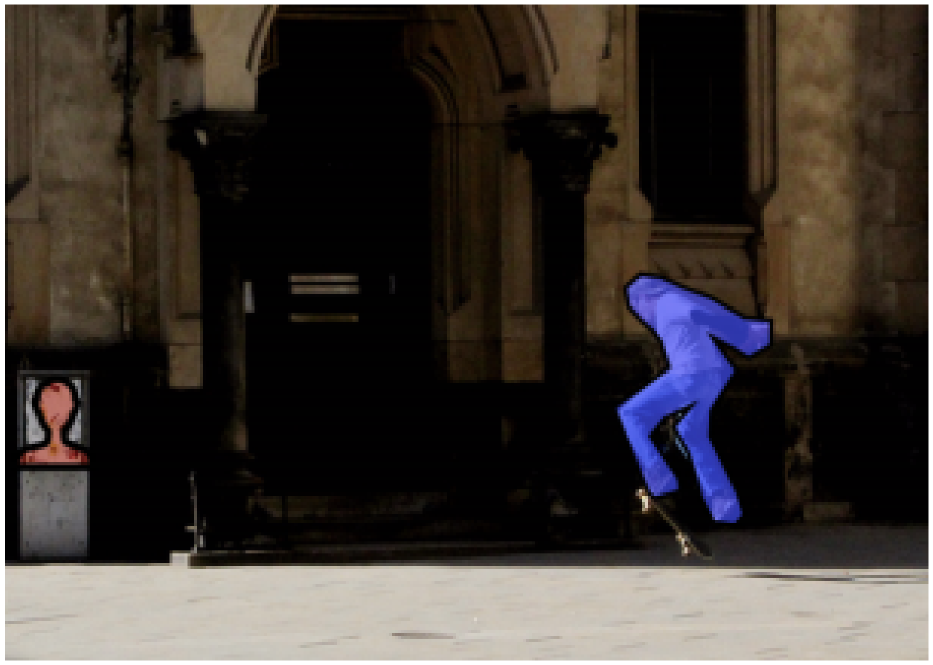
2016 error rate is 3.0%   2017 error rate is 2.25%

Coco Challenge 17

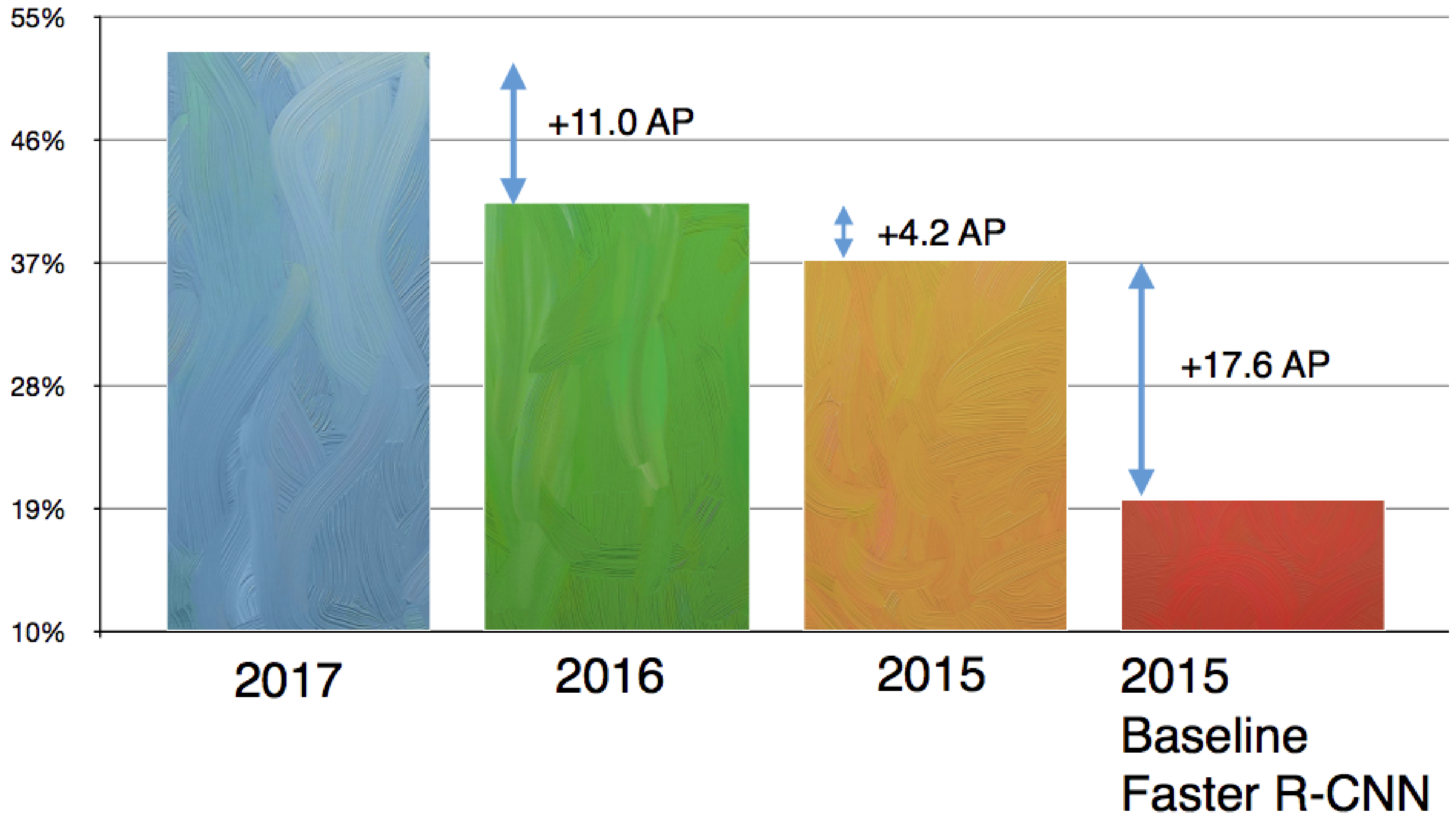
Detection

Segmentation



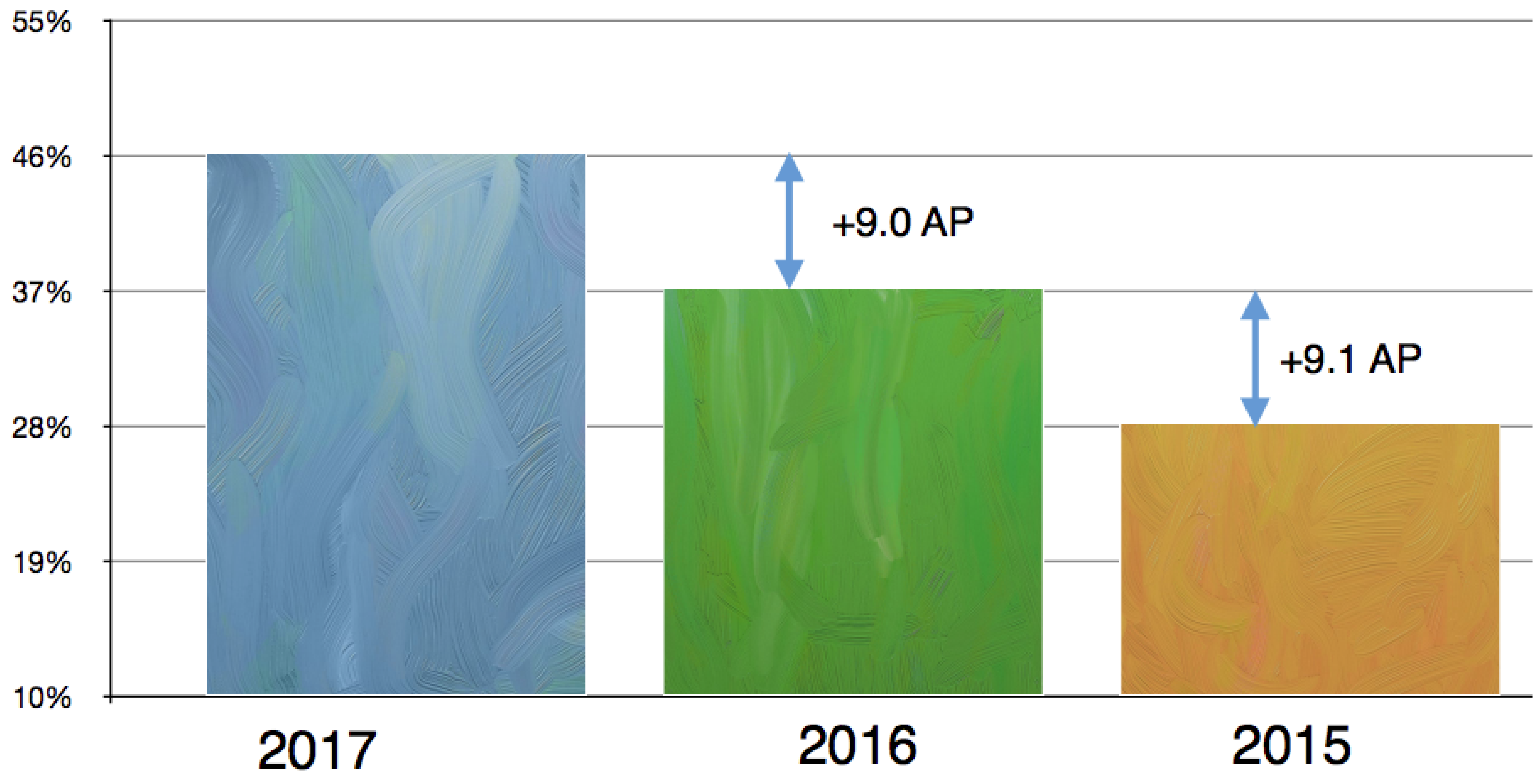


Detection



55%
46%
37%
28%
19%
10%
+11.0 AP
+4.2 AP
+17.6 AP
2017
2016
2015
2015 Baseline Faster R-CNN

Segmentation



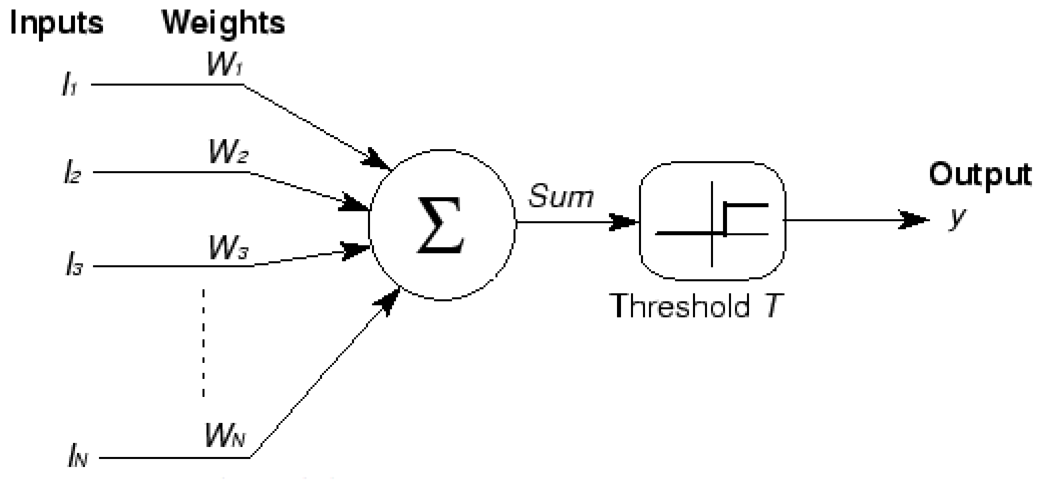
nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last — a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS GO

The Deep Revolution is Everywhere

- Computer Vision
- Speech Recognition
- Machine Translation
- Computer Games
- Information Retrieval (Google Search)
- Computational Chemistry
- ...

Some History of Deep Learning

McCullock and Pitts 1943 — introduced the linear threshold "neuron".



Rosenblatt 1962 — Applied a "Hebbian" learning rule.

Novikoff 1962 — proved the perceptron convergence theorem.

Deep Winter I: late 60s through early 80s

Robinson 1965 — introduces resolution theorem proving.

Minsky 1969 — wins Turing Award for “promoting AI”.

McCarthy and Hayes 1968 — introduced the situation calculus.

Minsky and Papert 1969 — published the book *Perceptrons*. They proved that many properties of images could not be determined by (single layer) perceptrons. Caused a decline of activity in neural network research.

McCarthy, 1971 — wins Turing Award.

Minsky 1974 — wrote “A Framework for Representing Knowledge”.

McCarthy 1980 — introduces “non-monotonic logic”.

Deep Resurgence I, late 80s

Fukushima 1980 — introduced the neocognitron (a form of CNN)

Hinton and Sejnowski 1985 — introduce the Boltzman machine

Rummelhart, Hinton and Williams 1986 — demonstrated empirical success with backpropagation (itself dating back to 1961).

Deep Winter II: Late 90s' and 00's

Valiant 1984 — introduces the formal definition of PAC learnability. Credited with starting learning theory as a branch of computer science. Turing Award, 2010.

Pearl 1995 — publishes *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Credited with driving the "statistical revolution" in AI. Turing Award, 2011.

Convex optimization and convex relaxations (the marginal polytope of a graphical model)

Nonparametric Bayesian inference (Dirichlet processes).

Submodular optimization

Deep Learning in Winter II

Schmidhuber et al. 1997 — introduces LSTMs

LeCun 1998 — introduces convolutional neural networks (CNNs) (LeNet).

Bengio 2003 — introduced neural language modeling

Deep Learning Explodes in 2012

Alexnet dominates the 2012 Imagenet challenge.

Google speech recognition converts to deep learning.

Both developments were driven by Hinton's group at the University of Toronto.

The four Horsemen of Deep Learning

Geoff Hinton, 70, H index = 136

Yann LeCun, 57, H index = 98

Yoshua Bengio, 53, H index = 105

Juergen Schmidhuber, 54, H index = 80

Prerequisites

- Linear Algebra, e.g. eigenvectors and eigenvalues of a matrix.
- Probability Theory, e.g., covariance matrices and multivariate Gaussians.
- Vector Calculus, e.g, Gradient, Jacobian and Hessian.
- Python (preferably knowledge of NumPy)

Prerequisites

I will be giving pointers into related sections of *Deep Learning* by Goodfellow, Begio and Courville.

We will be skipping over the first five chapters:

1. Introduction

2. Linear Algebra

3. Probability and Information Theory (we will cover information theory in lecture 8).

4. Numerical Computation

5. Machine Learning Basics

Lecture Plan

1. Introduction

2. Multiclass Logistic Regression and Multi Layer Perceptrons (MLPs). Introduction to Stochastic Gradient Descent.

3. Computation Graphs and Backpropagation. Minibatching. The educational framework EDF. **PS1 out: Residual MLP for MNIST in EDF.**

4. Convolutional Neural Networks (CNNs).

5. Recurrent Neural Networks (RNNs). Introduction to PyTorch. **PS1 due, PS2out: CNN for CIFAR10 in PyTorch.**

Lecture Plan

6. Stochastic Gradient Descent (SGD) : Gradient Estimation and Gradient Change. Asymptotic convergence. SGD vs. MCMC. Second order considerations and conditioning. Vector Space Treatment of Gradient Descent. Standard variants: vanilla, momentum, Adagrad, RMSProp, Adam, learning rate scaling with batch size.

7. Regularization. Weight Norm Regularization. Ensembles. Regularaization Dropout. PAC-Bayesian Generalization Guarantees. **PS2 due, PS3 out: SGD and regularization experiments on CFAR10**

8. Information Theory. entropy, cross entropy, KL-divergence, mutual information. Self-Supervised and Predictive Learning.

9. Deep Graphical Models. (Chapter 16 of Goodfellow et al.) **PS3 due. Ps4 Out: No coding, practice for midterm**

Lecture Plan

10. Deep Speech Recognition, Connections Temporal Classification (CTC), Expected Gradient (EG) and Expectation Maximization (EM).

11. Autoencoders: Rate-Distortion Autoencoders, Variational Autoencoders. **PS4 due**

12. **Midterm**

13. Generative Adversarial Networks. **Pareto Competition Base code out.**

14. Reinforcement Learning.

Lecture Plan

17. Alpha Zero **First Competiton Entries due**

18. Too Be Determined.

19. Interpretability **Final competition entries due.**

20. Universality.

21. Pereto compeition analysis and course review.

Problem Sets

Everyone should install **anaconda**. We will be using python 3.5 and pytorch 0.3.0.

Programming problem sets will be given in Jupyter notebooks and should be turned in as Jupyter notebooks.

Students are allowed to discuss the problem sets and projects with other students but each student should write their own code and problem solution and state what other students they discussed the problem with. The projects will be individual (we will not have group projects).

The Pareto Competition

- The competion will be on Penn Tree Bank (PTB) language modeling. PTB language modeling has become “the MNIST of RNNs”

- The competition will explore the training-time/test-performance Pareto curve.

- PTB performance is measured by perplexity where smaller is better.

- We will provide a baseline system that defines a single time/perplexity point.

- Each student will submit a pair (t, p) where t is the ratio of their running time to the baseline time on their machine and p is ratio of their perplexity to the baseline perplexity. The time ratio and the perplexity ratio must each be less than 2.

The Pareto Competition

- Each student can submit only one pair. A submission (t, p) dominates a submission (t', p') if both $t < t'$ and $p < p'$.
- A pair (t, p) that is not dominated by any other pair is called a Pareto point (a point on the Pareto curve).
- Every student entering a Pareto point will be considered a winner.
- The student dominating the most other students will be the overall winner.
- Students can freely discuss ideas but cannot share code.

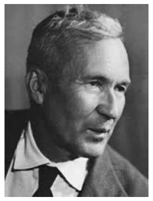
Grading

The problem sets, the midterm exam and the Pareto competition will each be worth 1/3 of the grade.

END

It seems we have time to debate Nature vs. Nurture

Noam Chomsky: Natural language grammar is unlearnable without with an innate linguistic capacity. This position is supported by the "no free lunch theorem".



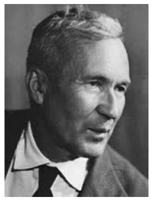
Andrey Kolmogorov, Geoff Hinton: Universal learning algorithms exist. This position is supported by the "free lunch theorem".

The No Free Lunch Theorem

Without prior knowledge, such as universal grammar, it is impossible to make a prediction for an input you have not seen in the training data.

Proof: Select a predictor h uniformly at random from all functions from $\mathcal{X}$ to $\mathcal{Y}$ and then take the data distribution to draw pairs $(x, h(x))$ where x is drawn uniformly from $\mathcal{X}$. No learning algorithm can predict $h(x)$ where x does not occur in the training data.

The Free Lunch Theorem



Universal (knowledge-free) learning algorithms exists.

Let h be a C++ procedure taking an input from $\mathcal{X}$ and returning a value in $\mathcal{Y}$, where h is written using calls to prodecures in an (arbitrarily large) code library L. Let $|h|$ be the number of bit in a standard compression algorithm applied to the source code for h. We are compressing only the "main" procedure h and not the library L.

Theorem: For the library L fixed before the draw of the sample, we have that with probability at least $1 - \delta$ over the draw of the sample the following holds *simultaneously* for all h and $\lambda > 1/2$.

$$\mathbf{Err}(h) \leq \frac{1}{1 - \frac{1}{2\lambda}} \left(\widehat{\mathbf{Err}}(h) + \frac{\lambda}{N} \left((\ln 2)|h| + \ln \frac{1}{\delta} \right) \right)$$

END