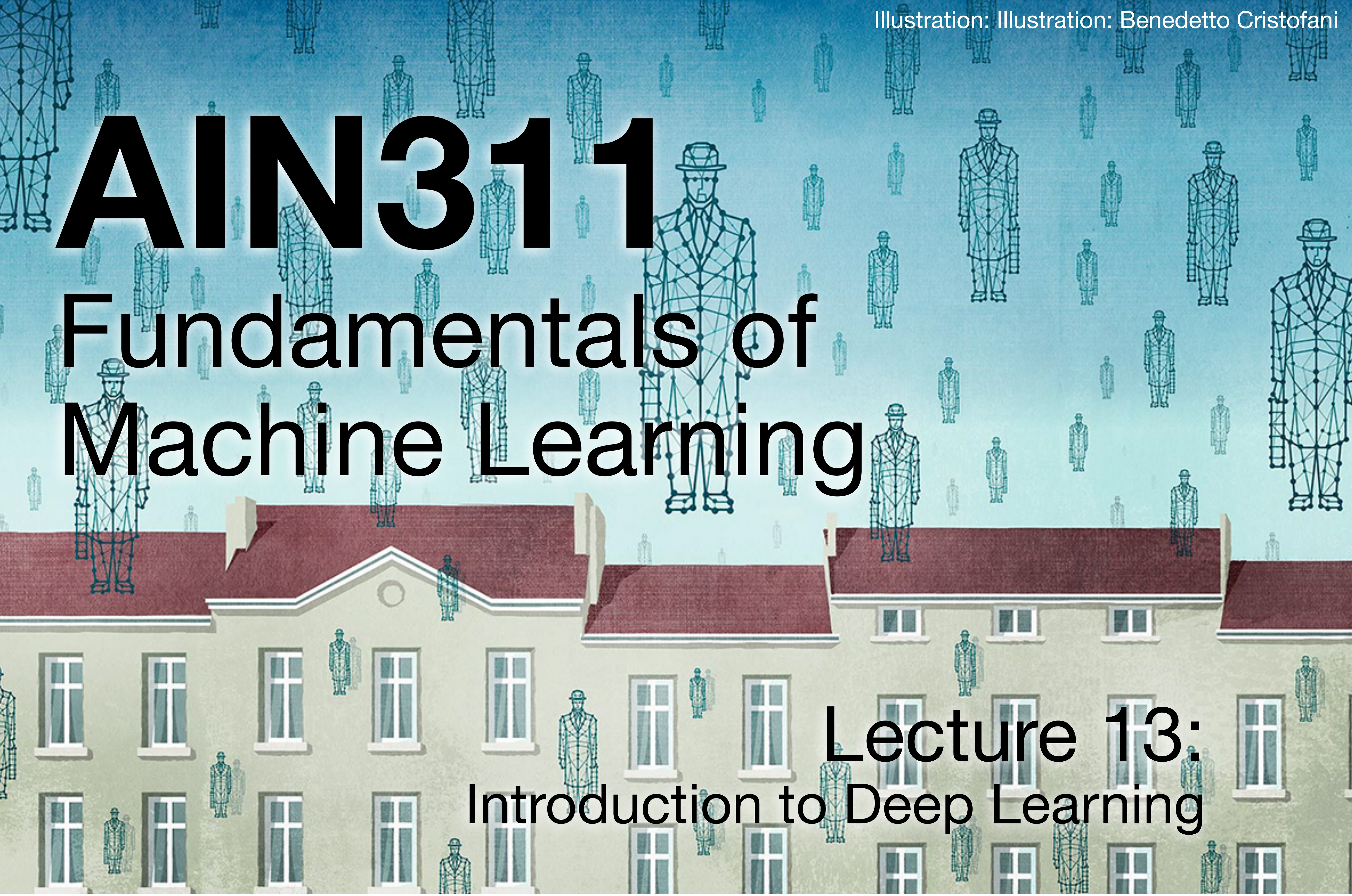


Illustration: Illustration: Benedetto Cristofani

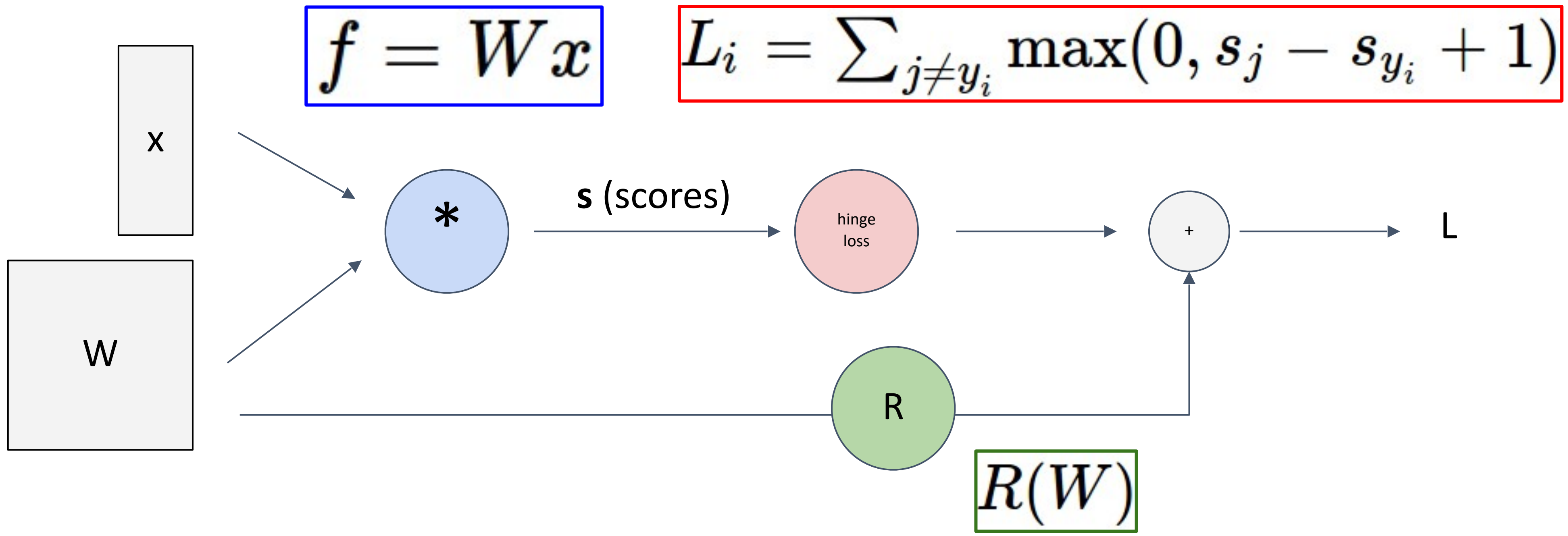
AIN311
Fundamentals of Machine Learning

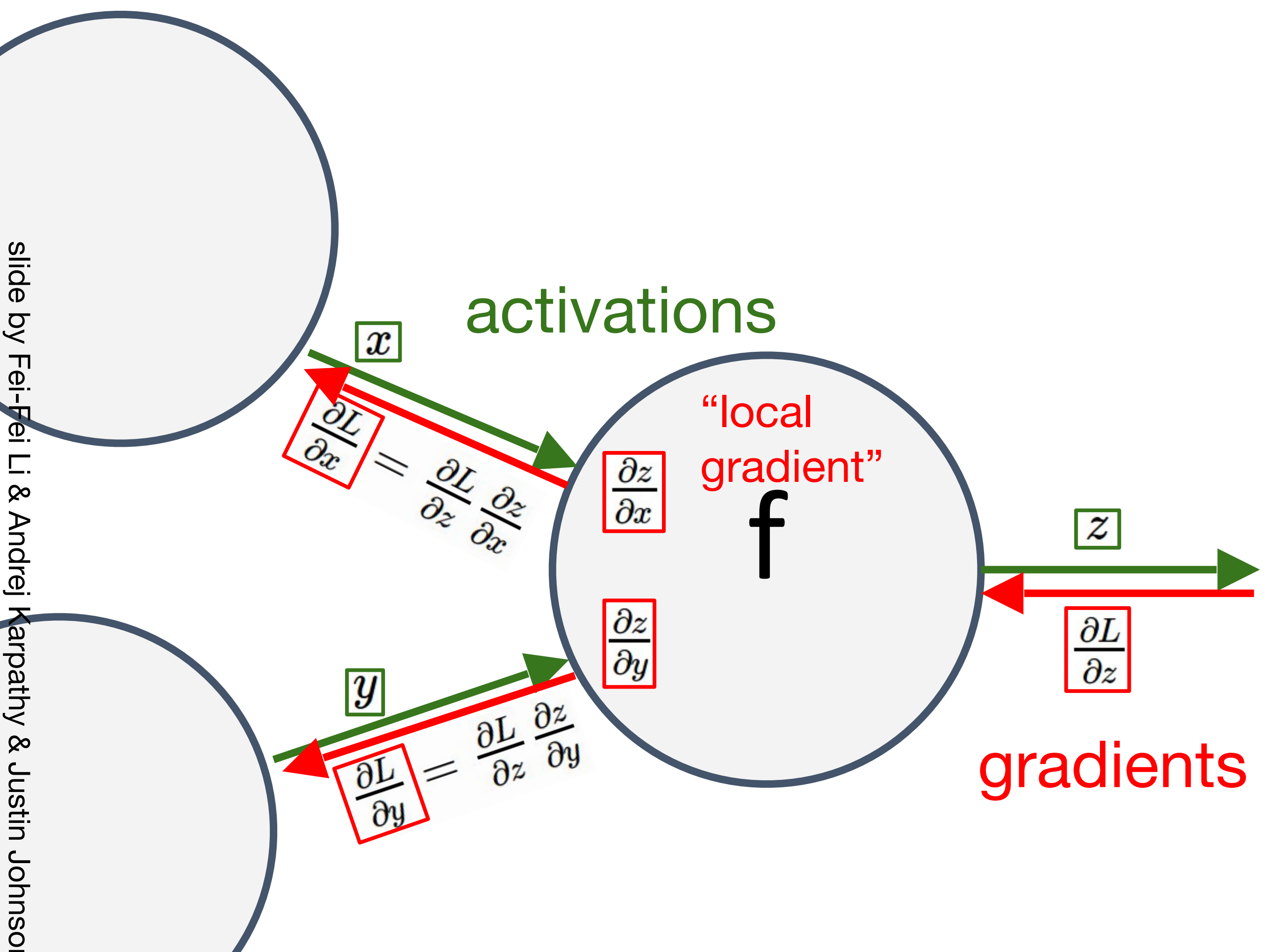
Lecture 13:
Introduction to Deep Learning

HACETTEPE UNIVERSITY COMPUTER VISION LAB

Erkut Erdem // Hacettepe University // Fall 2023

Last time.. **Computational Graph**






```python
class ComputationalGraph(object):
    #...
    def forward(inputs):
        # 1. [pass inputs to input gates...]
        # 2. forward the computational graph:
        for gate in self.graph.nodes_topologically_sorted():
            gate.forward()
        return loss # the final gate in the graph outputs the loss
    def backward():
        for gate in reversed(self.graph.nodes_topologically_sorted()):
            gate.backward() # little piece of backprop (chain rule applied)
        return inputs_gradients
```

Last time... **Training Neural Networks**

Mini-batch SGD

Loop:

1. **Sample** a batch of data
2. **Forward** prop it through the graph, get loss
3. **Backprop** to calculate the gradients
4. **Update** the parameters using the gradient

This week

- Introduction to Deep Learning
- Deep Convolutional Neural Networks

What is deep learning?

Y. LeCun, Y. Bengio, G. Hinton, "**Deep Learning**", Nature, Vol. 521, 28 May 2015

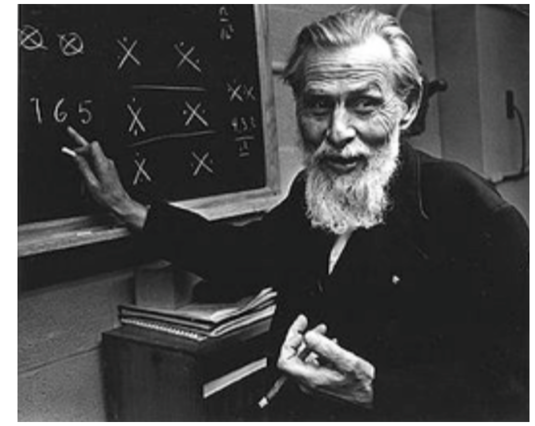
"Deep learning allows computational models that are composed of **multiple processing layers to learn representations of data with multiple levels of abstraction.**"

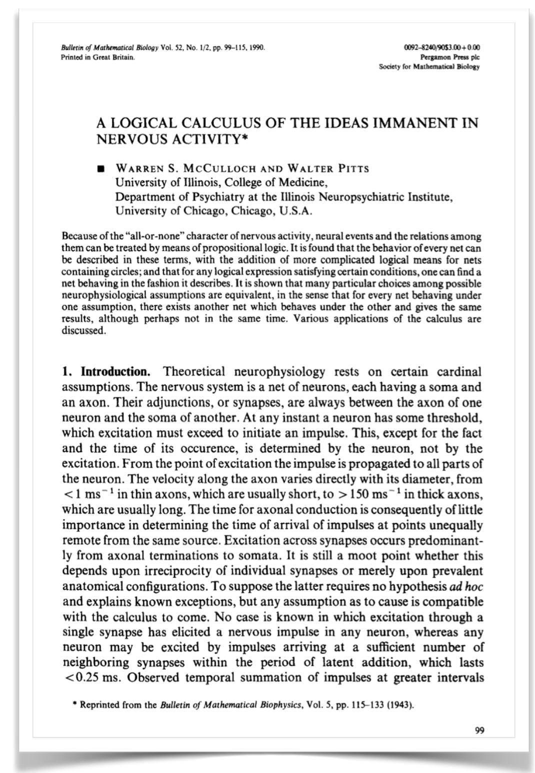
– Yann LeCun, Yoshua Bengio and Geoff Hinton

1943 – 2006:
A Prehistory of Deep Learning

1943: Warren McCulloch and Walter Pitts

- First computational model
- Neurons as logic gates (AND, OR, NOT)
- A neuron model that sums binary inputs and outputs 1 if the sum exceeds a certain threshold value, and otherwise outputs 0





Bulletin of Mathematical Biology Vol. 52, No. 1/2, pp. 99–115, 1990.
Printed in Great Britain.

0092-8240/90$3.00+0.00
Pergamon Press plc
Society for Mathematical Biology

A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY*

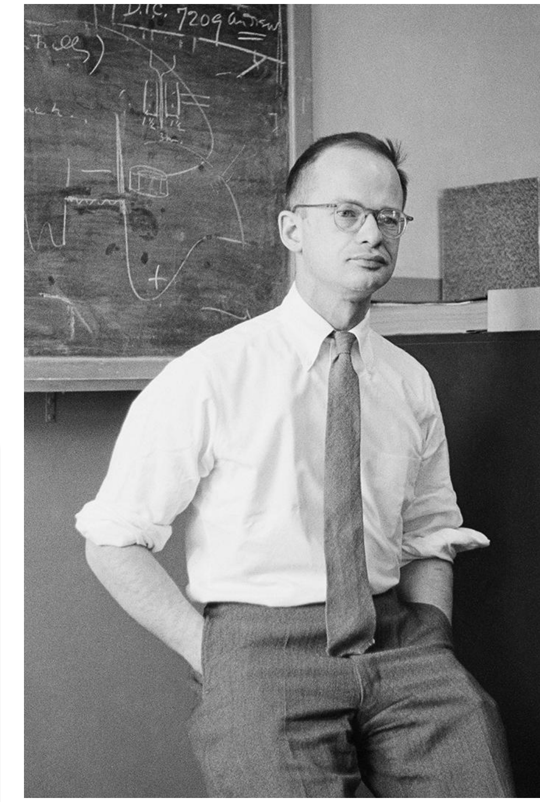
■ Warren S. McCulloch and Walter Pitts
University of Illinois, College of Medicine,
Department of Psychiatry at the Illinois Neuropsychiatric Institute,
University of Chicago, Chicago, U.S.A.

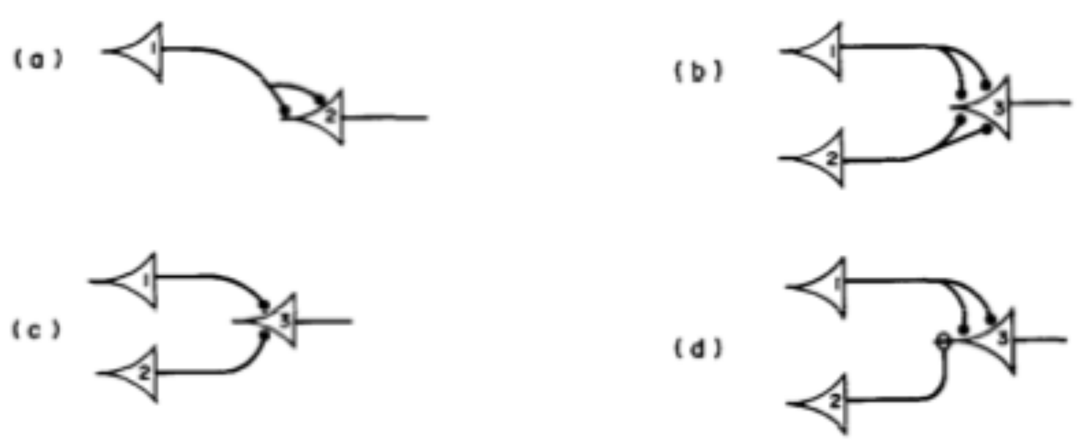
Because of the "all-or-none" character of nervous activity, neural events and the relations among them can be treated by means of propositional logic. It is found that the behavior of every net can be described in these terms, with the addition of more complicated logical means for nets containing circles; and that for any logical expression satisfying certain conditions, one can find a net behaving in the fashion it describes. It is shown that many particular choices among possible neurophysiological assumptions are equivalent, in the sense that for every net behaving under one assumption, there exists another net which behaves under the other and gives the same results, although perhaps not in the same time. Various applications of the calculus are discussed.

1. Introduction. Theoretical neurophysiology rests on certain cardinal assumptions. The nervous system is a net of neurons, each having a soma and an axon. Their adjunctions, or synapses, are always between the axon of one neuron and the soma of another. At any instant a neuron has some threshold, which excitation must exceed to initiate an impulse. This, except for the fact and the time of its occurence, is determined by the neuron, not by the excitation. From the point of excitation the impulse is propagated to all parts of the neuron. The velocity along the axon varies directly with its diameter, from $<1\ ms^{-1}$ in thin axons, which are usually short, to $>150\ ms^{-1}$ in thick axons, which are usually long. The time for axonal conduction is consequently of little importance in determining the time of arrival of impulses at points unequally remote from the same source. Excitation across synapses occurs predominantly from axonal terminations to somata. It is still a moot point whether this depends upon irreciprocity of individual synapses or merely upon prevalent anatomical configurations. To suppose the latter requires no hypothesis *ad hoc* and explains known exceptions, but any assumption as to cause is compatible with the calculus to come. No case is known in which excitation through a single synapse has elicited a nervous impulse in any neuron, whereas any neuron may be excited by impulses arriving at a sufficient number of neighboring synapses within the period of latent addition, which lasts <0.25 ms. Observed temporal summation of impulses at greater intervals

* Reprinted from the *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115–133 (1943).

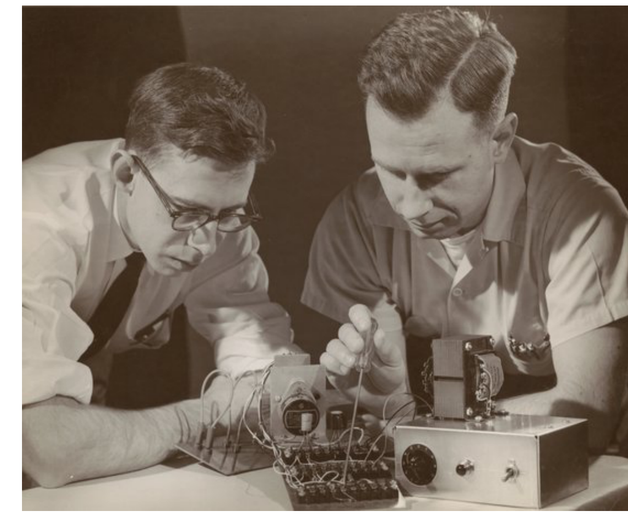
99

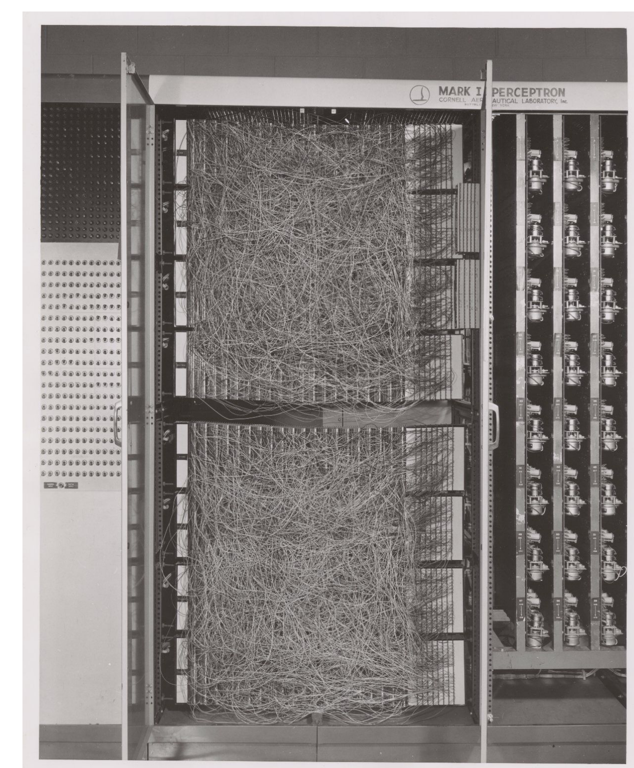


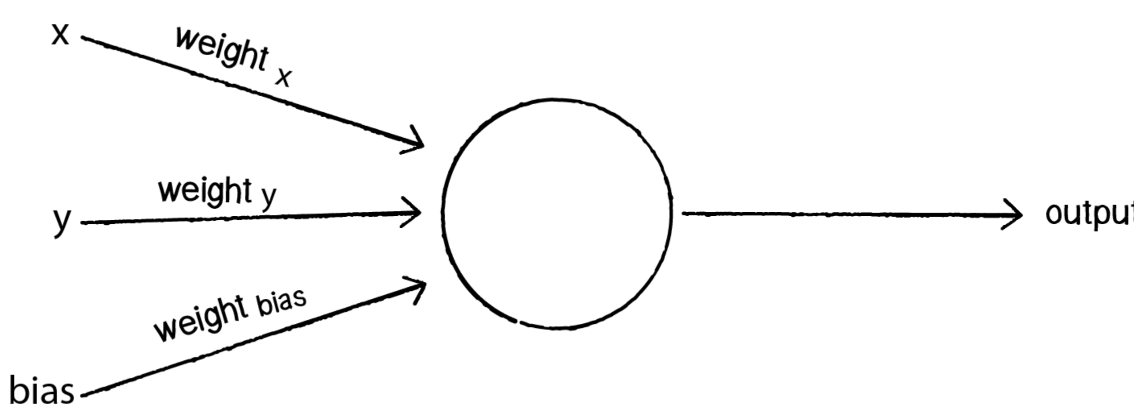


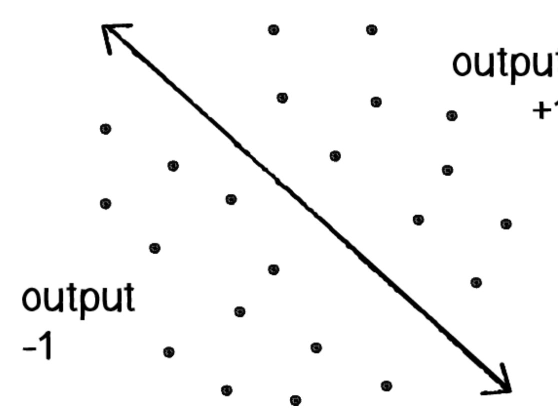
1958: Frank Rosenblatt's Perceptron

- A computational model of a **single neuron**
- Solves a **binary classification problem**
- Simple training algorithm
- Built using specialized hardware







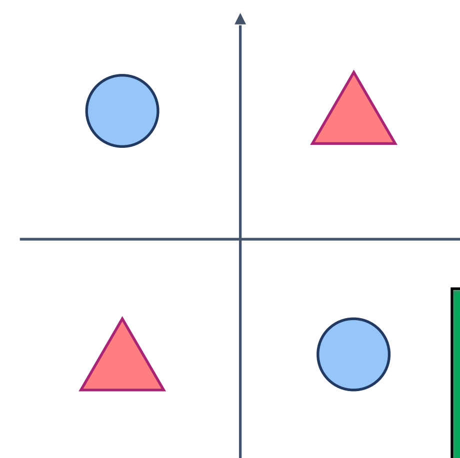


F. Rosenblatt, **"The perceptron: A probabilistic model for information storage and organization in the brain"**, Psychological Review, Vol. 65, 1958

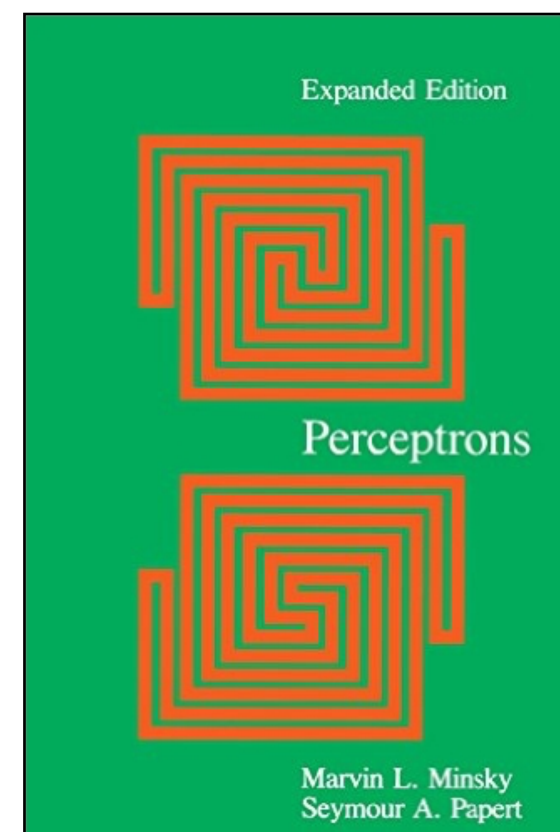
1969: Marvin Minsky and Seymour Papert

"No machine can learn to recognize X unless it possesses, at least potentially, some scheme for representing X." (p. xiii)

- Perceptrons can only represent linearly separable functions.
  - such as **XOR** Problem

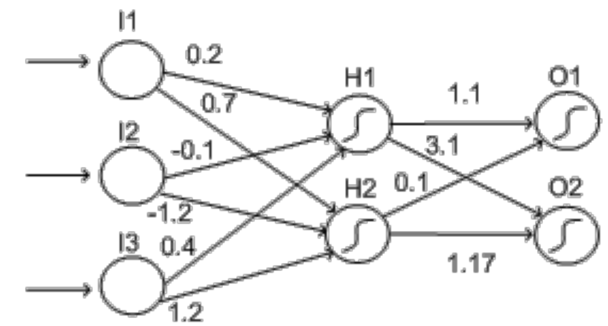


- Wrongly attributed as the reason behind the **AI winter**, a period of reduced funding and interest in AI research



1990s

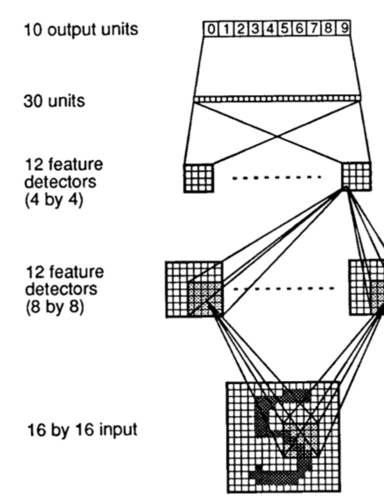
- **Multi-layer perceptrons** can theoretically learn any function (Cybenko, 1989; Hornik, 1991)




Backpropagation Through Time: What It Does and How to Do It

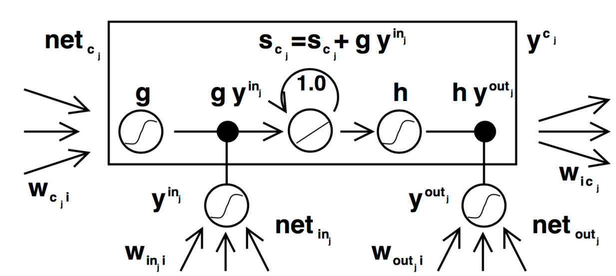
PAUL J. WERBOS

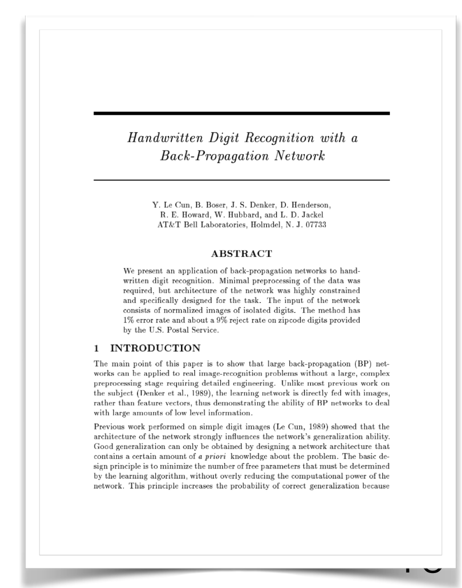
- Training multi-layer perceptrons
  - **Back-propagation** (Rumelhart, Hinton, Williams, 1986)
  - **Back-propagation through time** (BPTT) (Werbos, 1988)




Learning representations by back-propagating errors

- New neural architectures
  - **Convolutional neural nets** (LeCun et al., 1989)
  - **Long-short term memory networks** (LSTM) (Schmidhuber, 1997)




Handwritten Digit Recognition with a Back-Propagation Network

Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel
AT&T Bell Laboratories, Holmdel, N. J. 07733

ABSTRACT

1 INTRODUCTION

Why it failed then

- Too many parameters to learn from few labeled examples.
- “I know my features are better for this task”.
- Non-convex optimization? No, thanks.
- Black-box model, no interpretability.

- Very slow and inefficient
- Overshadowed by the success of SVMs (Cortes and Vapnik, 1995)

Adapted from Joan Bruna

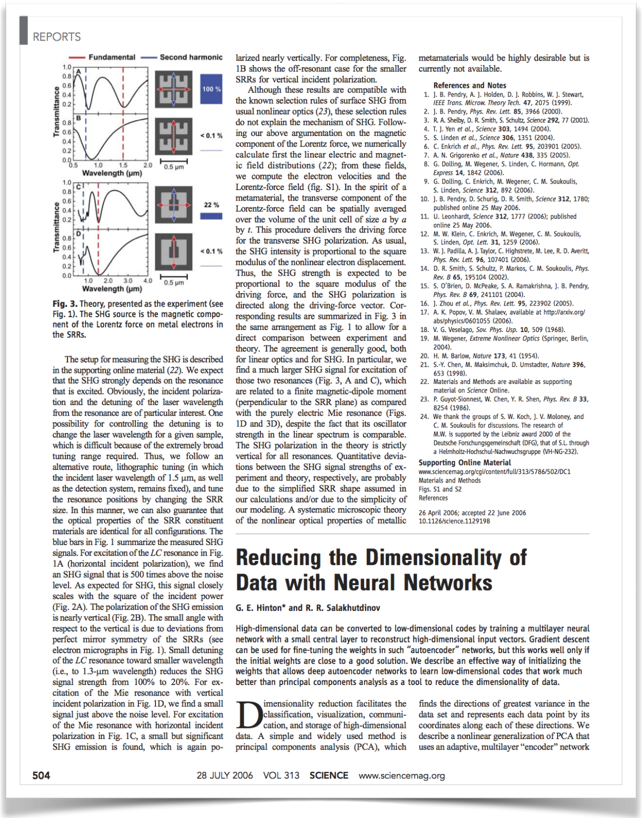
A major breakthrough in 2006

2006 Breakthrough: Hinton and Salakhutdinov

Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton* and R. R. Salakhutdinov

High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in such "autoencoder" networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.


REPORTS

Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton* and R. R. Salakhutdinov

504 28 JULY 2006 VOL 313 SCIENCE www.sciencemag.org

- The first solution to the **vanishing gradient problem**.
- Build the model in a layer-by-layer fashion using unsupervised learning
  - The features in early layers are already initialized or "pretrained" with some suitable features (weights).
  - Pretrained features in early layers only need to be adjusted slightly during supervised learning to achieve good results.

G. E. Hinton and R. R. Salakhutdinov, **"Reducing the dimensionality of data with neural networks"**, Science, Vol. 313, 28 July 2006.

The 2012 revolution

ImageNet Challenge

- IMAGENET Large Scale Visual Recognition Challenge (ILSVRC)
  - **1.2M** training images with **1K** categories
  - Measure top-5 classification error

Output
Scale
T-shirt
Steel drum
Drumstick
Mud turtle

Output
Scale
T-shirt
Giant panda
Drumstick
Mud turtle

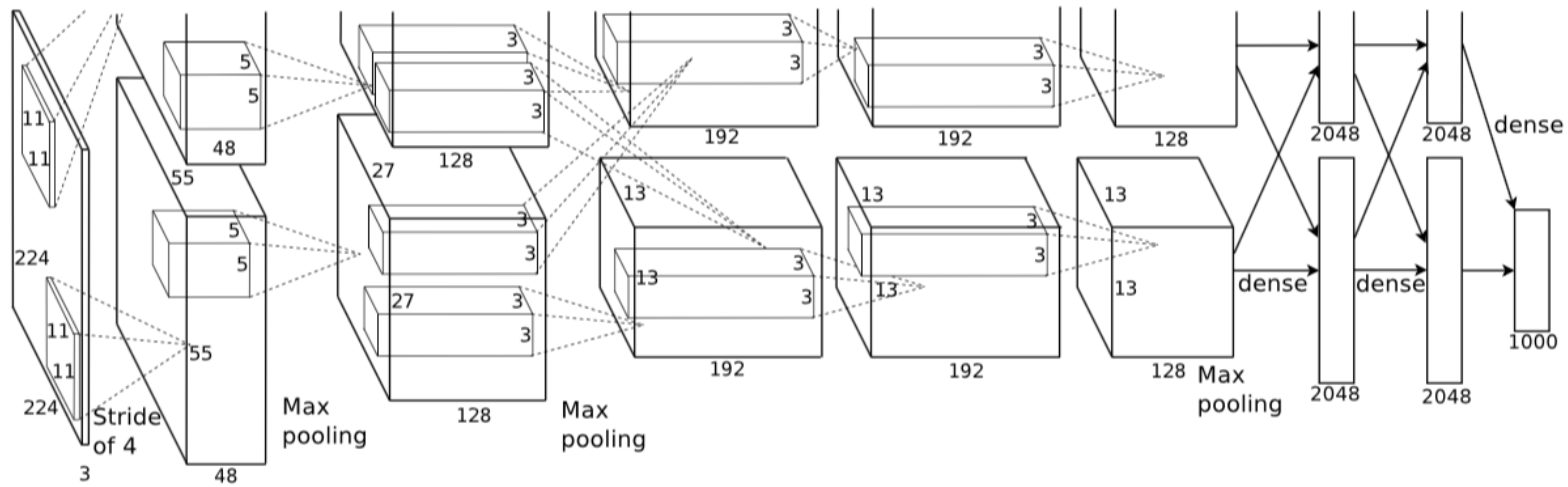
Image classification

J. Deng, Wei Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei , **“ImageNet: A Large-Scale Hierarchical Image Database”**, CVPR 2009.
O. Russakovsky et al., **“ImageNet Large Scale Visual Recognition Challenge”**, Int. J. Comput. Vis.,, Vol. 115, Issue 3, pp 211-252, 2015.

ILSVRC 2012 Competition

2012 Teams	%Error
Supervision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE/INRIA	27.0
UvA (Amsterdam)	29.6
INRIA/LEAR	33.4

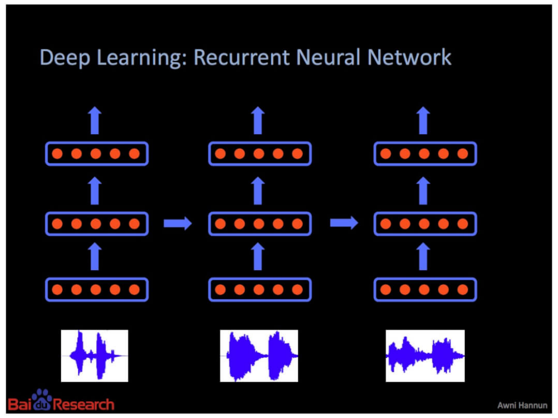
CNN based, non-CNN based



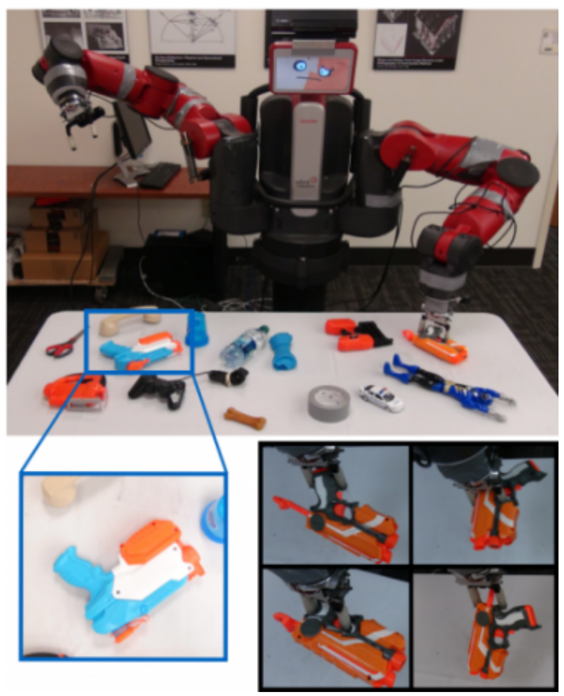
- The success of AlexNet, a deep convolutional network
  - 7 hidden layers (not counting some max pooling layers)
  - 60M parameters
- Combined several tricks
  - ReLU activation function, data augmentation, dropout

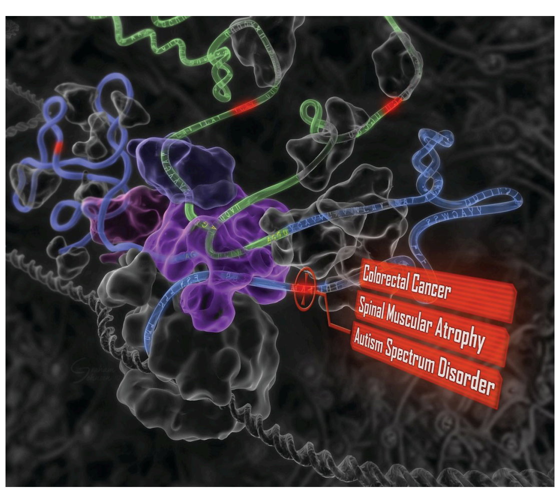
A. Krizhevsky, I. Sutskever, G.E. Hinton **"ImageNet Classification with Deep Convolutional Neural Networks"**, NIPS 2012

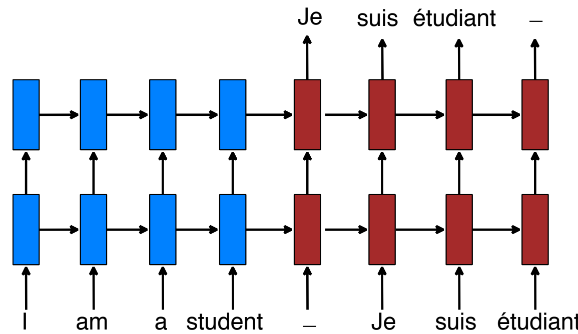
2012 – now
Deep Learning Era



Speech recognition


Robotics


Genomics



Machine Translation

Game Playing

Audio Generation


Self-Driving Cars

Amodei et al., **"Deep Speech 2: End-to-End Speech Recognition in English and Mandarin"**, In CoRR 2015

M.-T. Luong et al., **"Effective Approaches to Attention-based Neural Machine Translation"**, EMNLP 2015

M. Bojarski et al., **“End to End Learning for Self-Driving Cars”**, In CoRR 2016

D. Silver et al., **"Mastering the game of Go with deep neural networks and tree search"**, Nature 529, 2016

L. Pinto and A. Gupta, **“Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours”** ICRA 2015

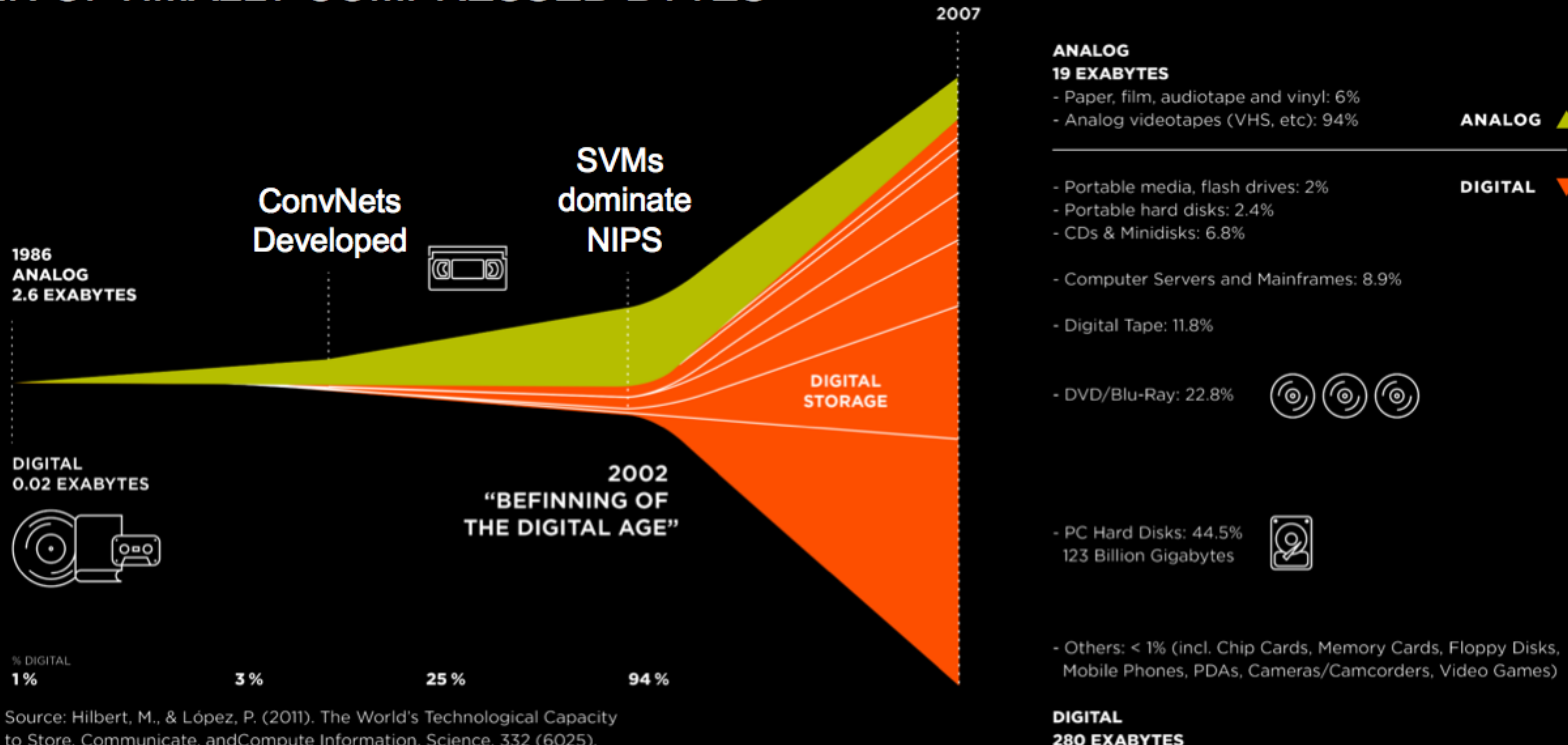
H. Y. Xiong et al., **"The human splicing code reveals new insights into the genetic determinants of disease"**, Science 347, 2015

M. Ramona et al., **"Capturing a Musician's Groove: Generation of Realistic Accompaniments from Single Song Recordings"**, In IJCAI 2015

And many more…

Why now?

GLOBAL INFORMATION STORAGE CAPACITY
IN OPTIMALLY COMPRESSED BYTES



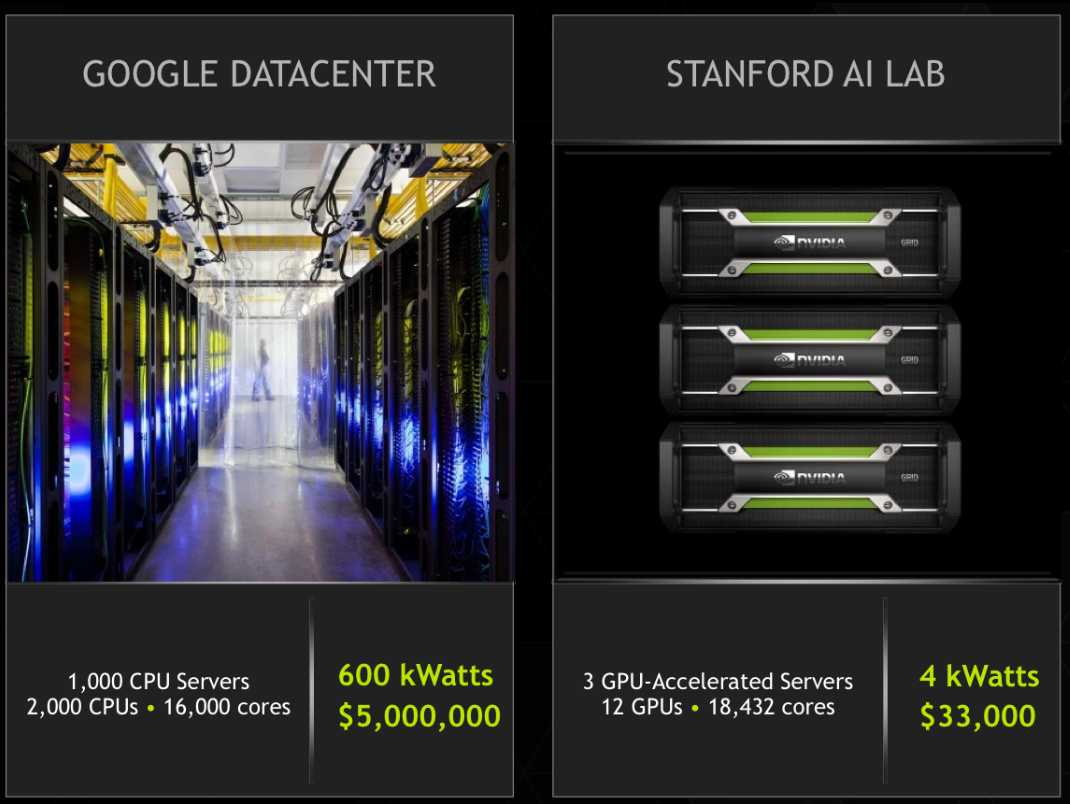
Source: Hilbert, M., & López, P. (2011). The World's Technological Capacity to Store, Communicate, andCompute Information. Science, 332 (6025), 60-65. martinhilbert.net/worldinfocapacity.html

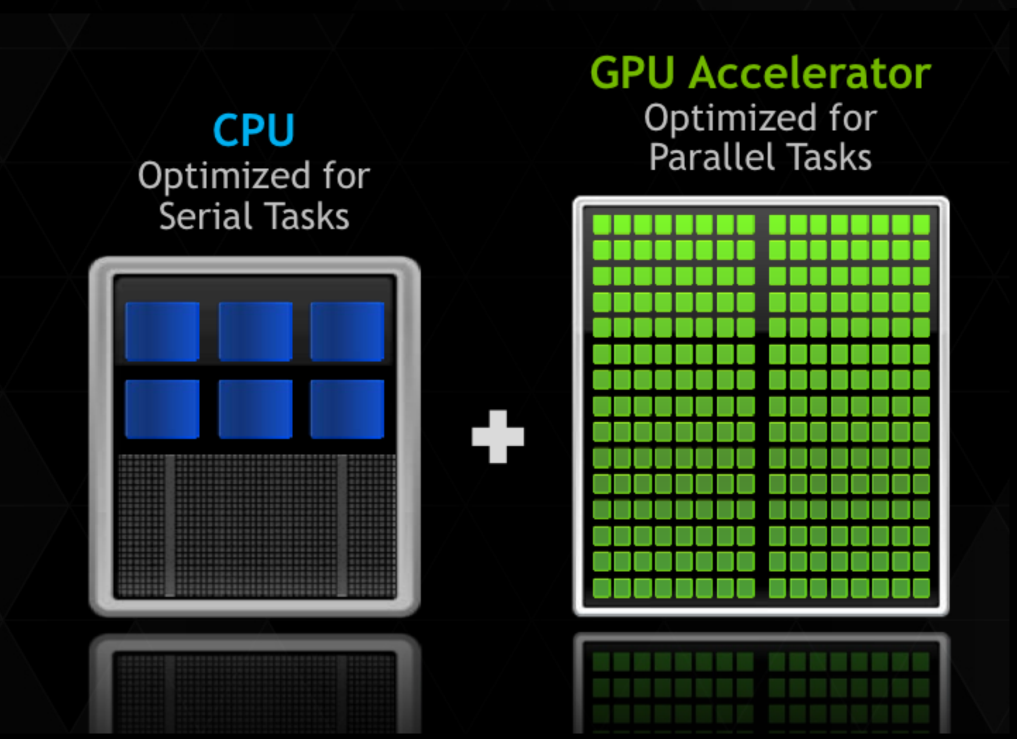
Datasets vs. Algorithms

Year	Breakthroughs in AI	Datasets (First Available)	Algorithms (First Proposed)
1994	Human-level spontaneous speech recognition	Spoken Wall Street Journal articles and other texts (1991)	Hidden Markov Model (1984)
1997	IBM Deep Blue defeated Garry Kasparov	700,000 Grandmaster chess games, aka “The Extended Book” (1991)	Negascout planning algorithm (1983)
2005	Google’s Arabic-and Chinese-to-English translation	1.8 trillion tokens from Google Web and News pages (collected in 2005)	Statistical machine translation algorithm (1988)
2011	IBM Watson became the world Jeopardy! champion	8.6 million documents from Wikipedia, Wiktionary, and Project Gutenberg (updated in 2010)	Mixture-of-Experts (1991)
2014	Google’s GoogLeNet object classification at near-human performance	ImageNet corpus of 1.5 million labeled images and 1,000 object categories (2010)	Convolutional Neural Networks (1989)
2015	Google’s DeepMind achieved human parity in playing 29 Atari games by learning general control from video	Arcade Learning Environment dataset of over 50 Atari games (2013)	Q-learning (1992)
Average No. of Years to Breakthrough:		**3 years**	**18 years**

Table credit: Quant Quanto

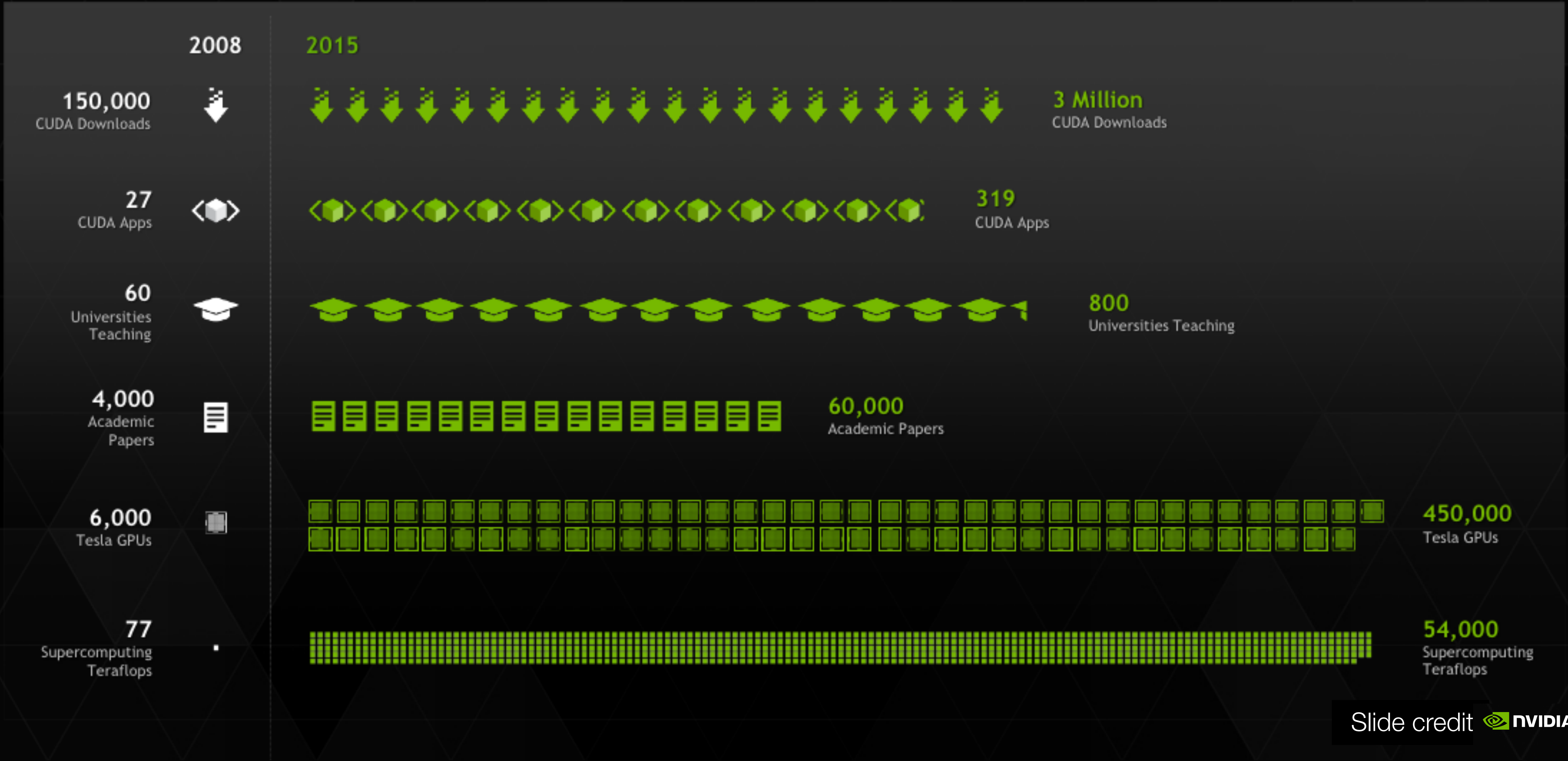
Powerful Hardware





Slide credit NVIDIA

10X GROWTH IN GPU COMPUTING



Slide credit NVIDIA

Working ideas on how to train deep architectures

Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Nitish Srivastava NITISH@CS.TORONTO.EDU
Geoffrey Hinton HINTON@CS.TORONTO.EDU
Alex Krizhevsky KRIZ@CS.TORONTO.EDU
Ilya Sutskever ILYA@CS.TORONTO.EDU
Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU

Abstract

Deep neural nets with a large number of parameters are very powerful machine learning systems. However, overfitting is a serious problem in such networks. Large networks are also slow to use, making it difficult to deal with overfitting by combining the predictions of many different large neural nets at test time. Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different "thinned" networks. At test time,

Journal of Machine Learning Research 15 (2014) 1929-1958 Submitted 11/13; Published 6/14

Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Nitish Srivastava NITISH@CS.TORONTO.EDU
Geoffrey Hinton HINTON@CS.TORONTO.EDU
Alex Krizhevsky KRIZ@CS.TORONTO.EDU
Ilya Sutskever ILYA@CS.TORONTO.EDU
Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
10 Kings College Road, Rm 3302
Toronto, Ontario, M5S 3G4, Canada.

Editor: Yoshua Bengio

Abstract

Deep neural nets with a large number of parameters are very powerful machine learning systems. However, overfitting is a serious problem in such networks. Large networks are also slow to use, making it difficult to deal with overfitting by combining the predictions of many different large neural nets at test time. Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different "thinned" networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization methods. We show that dropout improves the performance of neural networks on supervised learning tasks in vision, speech recognition, document classification and computational biology, obtaining state-of-the-art results on many benchmark data sets.

Keywords: neural networks, regularization, model combination, deep learning

1. Introduction

Deep neural networks contain multiple non-linear hidden layers and this makes them very expressive models that can learn very complicated relationships between their inputs and outputs. With limited training data, however, many of these complicated relationships will be the result of sampling noise, so they will exist in the training set but not in real test data even if it is drawn from the same distribution. This leads to overfitting and many methods have been developed for reducing it. These include stopping the training as soon as performance on a validation set starts to get worse, introducing weight penalties of various kinds such as L1 and L2 regularization and soft weight sharing (Nowlan and Hinton, 1992).

With unlimited computation, the best way to "regularize" a fixed-sized model is to average the predictions of all possible settings of the parameters, weighting each setting by

©2014 Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov.

- Better Learning Regularization (e.g. **Dropout**)

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, **"Dropout: A Simple Way to Prevent Neural Networks from Overfitting"**,
JMLR Vol. 15, No. 1,

Working ideas on how to train deep architectures

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Sergey Ioffe
Google Inc., *sioffe@google.com*

Christian Szegedy
Google Inc., *szegedy@google.com*

Abstract

Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities. We refer to this phenomenon as *internal covariate shift*, and address the problem by normalizing layer inputs. Our method draws its strength from making normalization a part of the model architecture and performing the normalization *for each training mini-batch*. Batch Nor-

Using mini-batches of examples, as opposed to one example at a time, is helpful in several ways. First, the gradient of the loss over a mini-batch is an estimate of the gradient over the training set, whose quality improves as the batch size increases. Second, computation over a batch can be much more efficient than m computations for individual examples, due to the parallelism afforded by the modern computing platforms.

While stochastic gradient is simple and effective, it requires careful tuning of the model hyper-parameters, specifically the learning rate used in optimization, as well as the initial values for the model parameters. The training is complicated by the fact that the inputs to each layer

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Sergey Ioffe
Google Inc., *sioffe@google.com*

Christian Szegedy
Google Inc., *szegedy@google.com*

Abstract

Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities. We refer to this phenomenon as *internal covariate shift*, and address the problem by normalizing layer inputs. Our method draws its strength from making normalization a part of the model architecture and performing the normalization *for each training mini-batch*. Batch Nor-malization allows us to use much higher learning rates and be less careful about initialization. It also acts as a regularizer, in some cases eliminating the need for Dropout. Applied to a state-of-the-art image classification model, Batch Normalization achieves the same accuracy with 14 times fewer training steps, and beats the original model by a significant margin. Using an ensemble of batch-normalized networks, we improve upon the best published result on ImageNet classification: reaching 4.9% top-5 validation error (and 4.8% test error), exceeding the accuracy of human raters.

1 Introduction

Deep learning has dramatically advanced the state of the art in vision, speech, and many other areas. Stochastic gradient descent (SGD) has proved to be an effective way of training deep networks, and SGD variants such as momentum (Sutskever et al., 2013) and Adagrad (Duchi et al., 2011) have been used to achieve state of the art performance. SGD optimizes the parameters Θ of the network, so as to minimize the loss

$$\Theta = \arg\min_{\Theta} \frac{1}{N}\sum_{i=1}^{N} \ell(\mathrm{x}_i, \Theta)$$

where $\mathrm{x}_{1\ldots N}$ is the training data set. With SGD, the training proceeds in steps, and at each step we consider a *mini-batch* $\mathrm{x}_{1\ldots m}$ of size m. The mini-batch is used to approximate the gradient of the loss function with respect to the parameters, by computing

$$\frac{1}{m}\frac{\partial \ell(\mathrm{x}_i, \Theta)}{\partial \Theta}.$$

Using mini-batches of examples, as opposed to one example at a time, is helpful in several ways. First, the gradient of the loss over a mini-batch is an estimate of the gradient over the training set, whose quality improves as the batch size increases. Second, computation over a batch can be much more efficient than m computations for individual examples, due to the parallelism afforded by the modern computing platforms.

While stochastic gradient is simple and effective, it requires careful tuning of the model hyper-parameters, specifically the learning rate used in optimization, as well as the initial values for the model parameters. The training is complicated by the fact that the inputs to each layer are affected by the parameters of all preceding layers – so that small changes to the network parameters amplify as the network becomes deeper.

The change in the distributions of layers' inputs presents a problem because the layers need to continuously adapt to the new distribution. When the input distribution to a learning system changes, it is said to experience *covariate shift* (Shimodaira, 2000). This is typically handled via domain adaptation (Jiang, 2008). However, the notion of covariate shift can be extended beyond the learning system as a whole, to apply to its parts, such as a sub-network or a layer. Consider a network computing

$$\ell = F_2(F_1(\mathrm{u}, \Theta_1), \Theta_2)$$

where F_1 and F_2 are arbitrary transformations, and the parameters Θ_1, Θ_2 are to be learned so as to minimize the loss ℓ. Learning Θ_2 can be viewed as if the inputs $\mathrm{x} = F_1(\mathrm{u}, \Theta_1)$ are fed into the sub-network

$$\ell = F_2(\mathrm{x}, \Theta_2).$$

For example, a gradient descent step

$$\Theta_2 \leftarrow \Theta_2 - \frac{\alpha}{m}\sum_{i=1}^{m}\frac{\partial F_2(\mathrm{x}_i, \Theta_2)}{\partial \Theta_2}$$

(for batch size m and learning rate α) is exactly equivalent to that for a stand-alone network F_2 with input x. Therefore, the input distribution properties that make training more efficient – such as having the same distribution between the training and test data – apply to training the sub-network as well. As such it is advantageous for the distribution of x to remain fixed over time. Then, Θ_2 does

1

- Better Optimization Conditioning (e.g. **Batch Normalization**)

S. Ioffe, C. Szegedy, **"Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift"**, In ICML 2015

Working ideas on how to train deep architectures

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets [41] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error

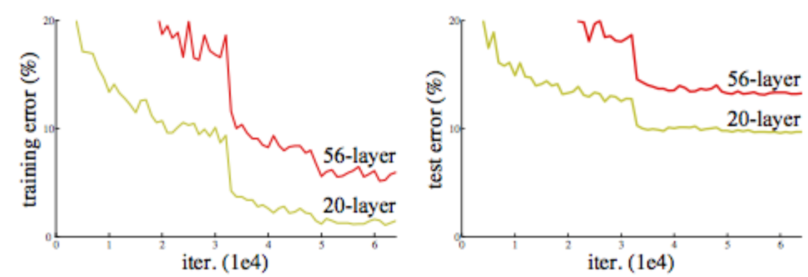
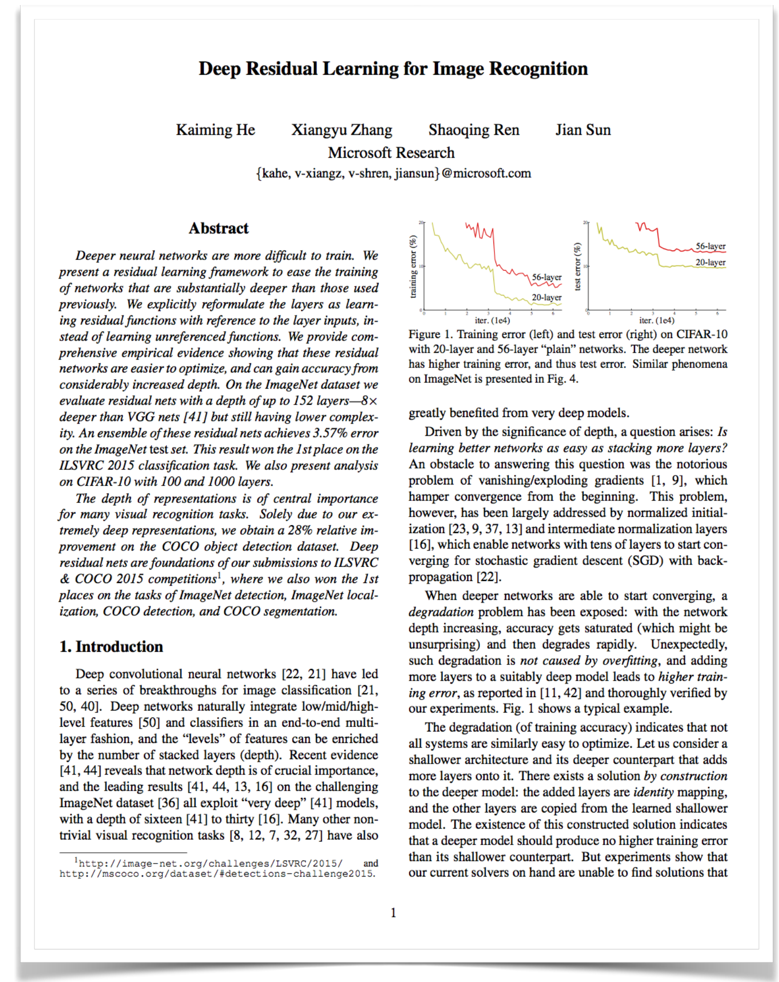


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

greatly benefited from very deep models.

Driven by the significance of depth, a question arises: *Is*



Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets [41] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. *This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers.*

The depth of representations is of central importance for many visual recognition tasks. Solely due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset. Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions[1]*, where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.*

1. Introduction

Deep convolutional neural networks [22, 21] have led to a series of breakthroughs for image classification [21, 50, 40]. Deep networks naturally integrate low/mid/high-level features [50] and classifiers in an end-to-end multi-layer fashion, and the "levels" of features can be enriched by the number of stacked layers (depth). Recent evidence [41, 44] reveals that network depth is of crucial importance, and the leading results [41, 44, 13, 16] on the challenging ImageNet dataset [36] all exploit "very deep" [41] models, with a depth of sixteen [41] to thirty [16]. Many other non-trivial visual recognition tasks [8, 12, 7, 32, 27] have also

[1] http://image-net.org/challenges/LSVRC/2015/ and http://mscoco.org/dataset/#detections-challenge2015.

Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

greatly benefited from very deep models.

Driven by the significance of depth, a question arises: *Is learning better networks as easy as stacking more layers?* An obstacle to answering this question was the notorious problem of vanishing/exploding gradients [1, 9], which hamper convergence from the beginning. This problem, however, has been largely addressed by normalized initialization [23, 9, 37, 13] and intermediate normalization layers [16], which enable networks with tens of layers to start converging for stochastic gradient descent (SGD) with back-propagation [22].

When deeper networks are able to start converging, a *degradation* problem has been exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. Unexpectedly, such degradation is *not caused by overfitting*, and adding more layers to a suitably deep model leads to *higher training error*, as reported in [11, 42] and thoroughly verified by our experiments. Fig. 1 shows a typical example.

The degradation (of training accuracy) indicates that not all systems are similarly easy to optimize. Let us consider a shallower architecture and its deeper counterpart that adds more layers onto it. There exists a solution *by construction* to the deeper model: the added layers are *identity* mapping, and the other layers are copied from the learned shallower model. The existence of this constructed solution indicates that a deeper model should produce no higher training error than its shallower counterpart. But experiments show that our current solvers on hand are unable to find solutions that

1

- Better neural achitectures (e.g. **Residual Nets**)

K. He, X. Zhang, S. Ren, J. Sun, **"Deep Residual Learning for Image Recognition"**, In CVPR 2016

So what is deep learning?

Three key ideas

- (Hierarchical) Compositionality
- End-to-End Learning
- Distributed Representations

slide by Dhruv Batra

Three key ideas

- **(Hierarchical) Compositionality**
  - Cascade of non-linear transformations
  - Multiple layers of representations
- End-to-End Learning
  - Learning (goal-driven) representations
  - Learning to feature extract
- Distributed Representations
  - No single neuron “encodes” everything
  - Groups of neurons work together

slide by Dhruv Batra

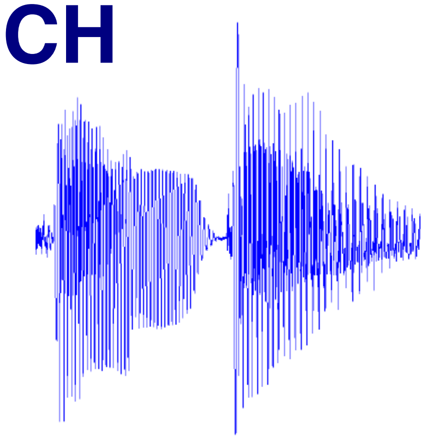
Traditional Machine Learning

VISION

hand-crafted features SIFT/HOG → your favorite classifier → "car"

fixed | learned

SPEECH



hand-crafted features MFCC → your favorite classifier → \'d ē p\

fixed | learned

NLP

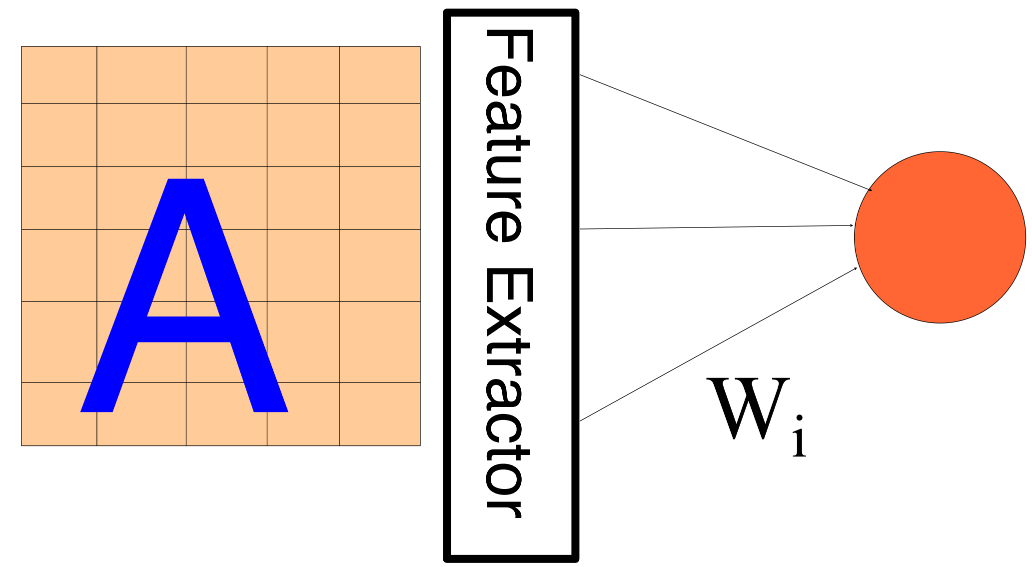
This burrito place is yummy and fun! → hand-crafted features Bag-of-words → your favorite classifier → "+"

fixed | learned

slide by Marc'Aurelio Ranzato, Yann LeCun

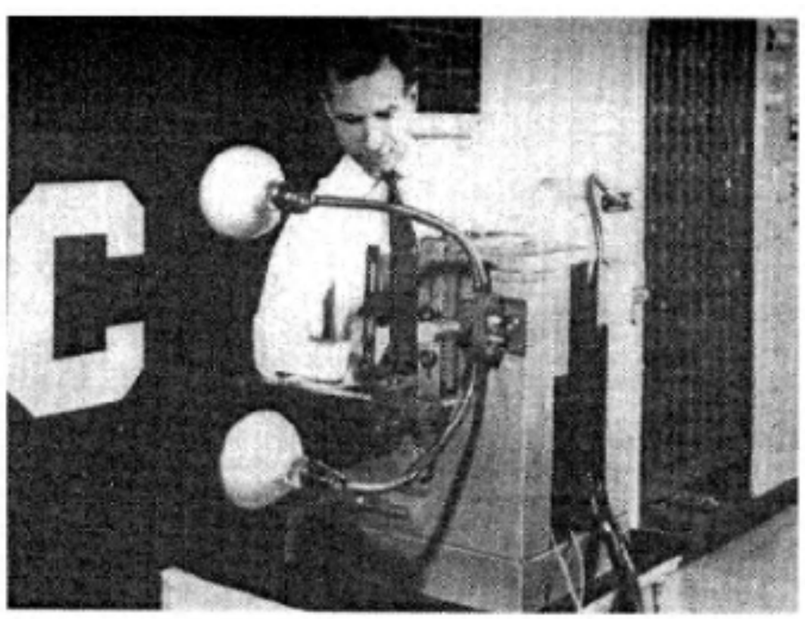
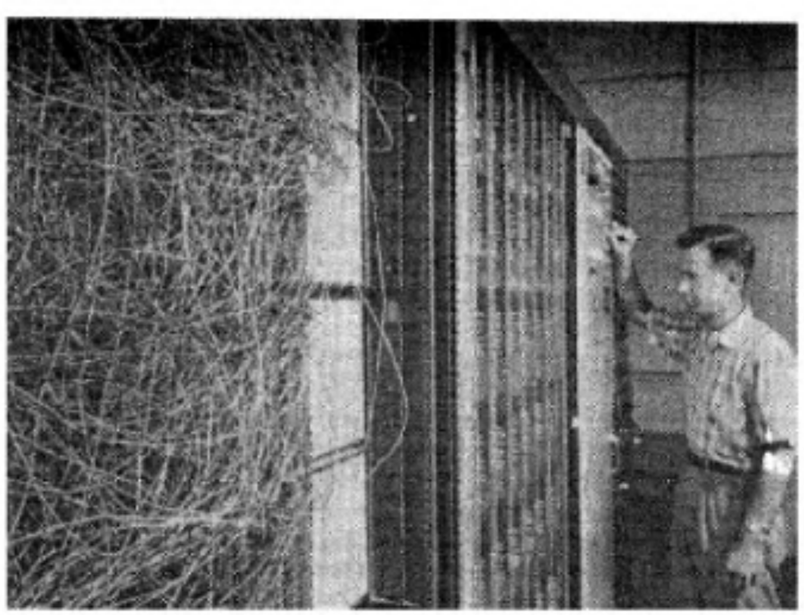
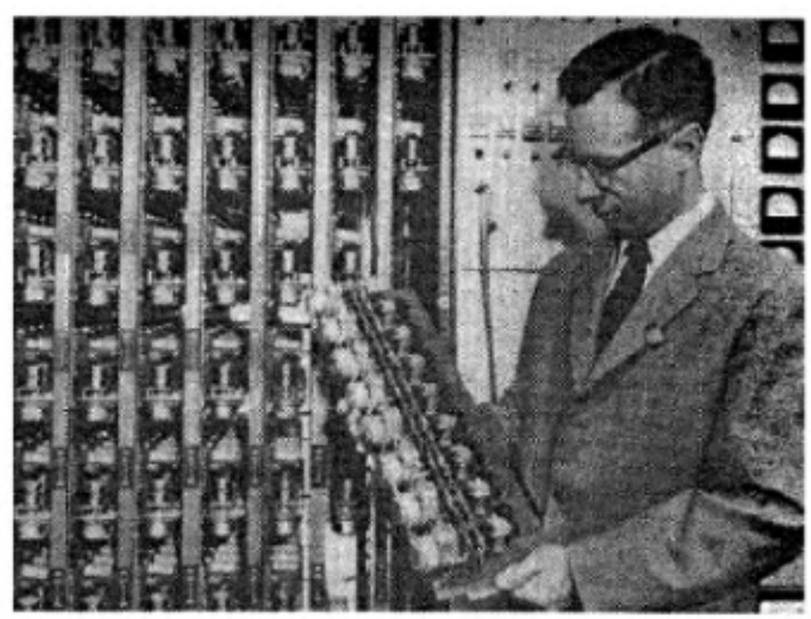
It's an old paradigm

- The first learning machine: the Perceptron
  - Built at Cornell in 1960
- The Perceptron was a linear classifier on top of a simple feature extractor



$$y = sign\left(\sum_{i=1}^{N} W_i F_i(X) + b\right)$$

- The vast majority of practical applications of ML today use glorified linear classifiers or glorified template matching.
- Designing a feature extractor requires considerable efforts by experts.

slide by Marc'Aurelio Ranzato, Yann LeCun

Hierarchical Compositionality

VISION

pixels → edge → texton → motif → part → object

SPEECH

sample → spectral band → formant → motif → phone → word

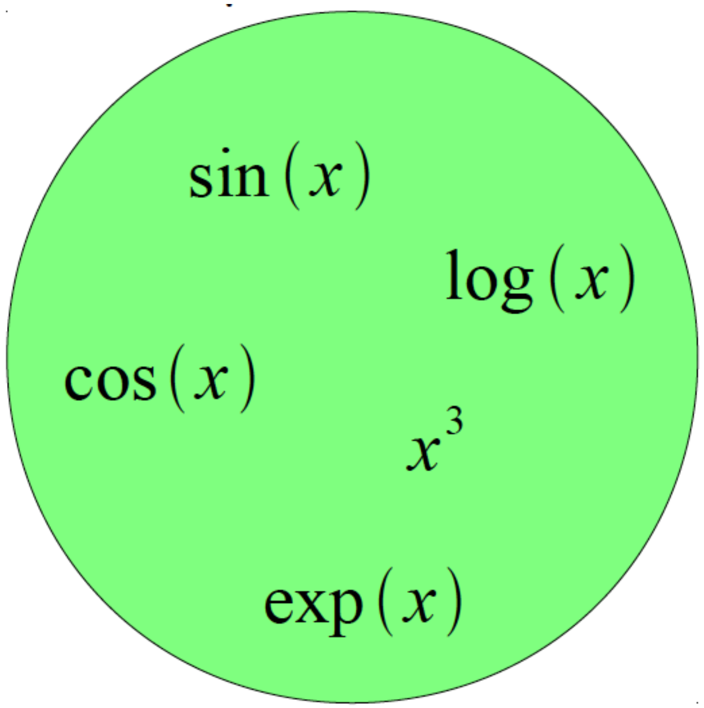
NLP

character → word → NP/VP/.. → clause → sentence → story

slide by Marc'Aurelio Ranzato, Yann LeCun

Building A Complicated Function

Given a library of simple functions



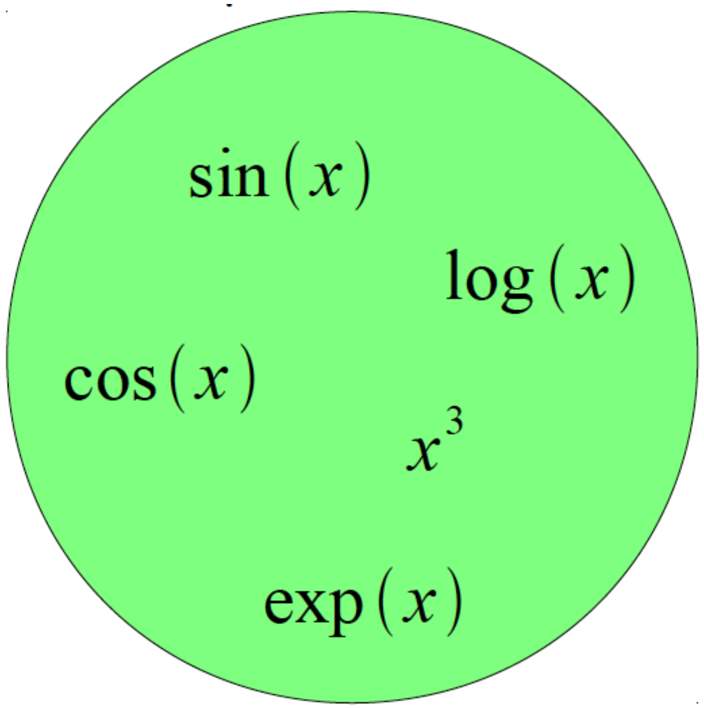
Compose into a

complicate function

slide by Marc'Aurelio Ranzato, Yann LeCun

Building A Complicated Function

Given a library of simple functions

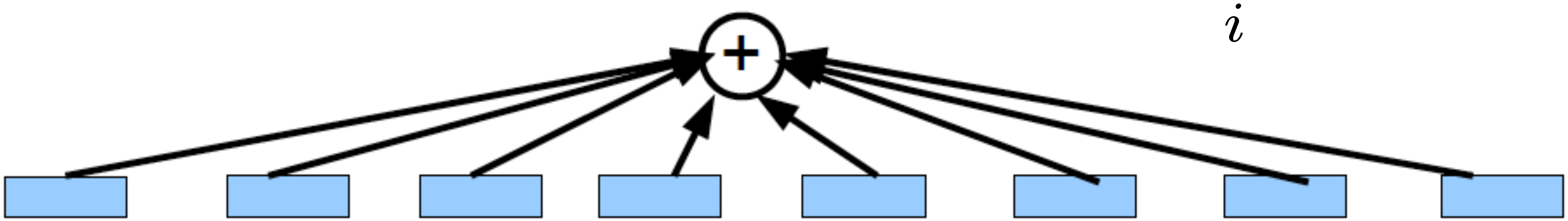


Compose into a complicate function

Idea 1: Linear Combinations

- Boosting
- Kernels
- $\cdots$

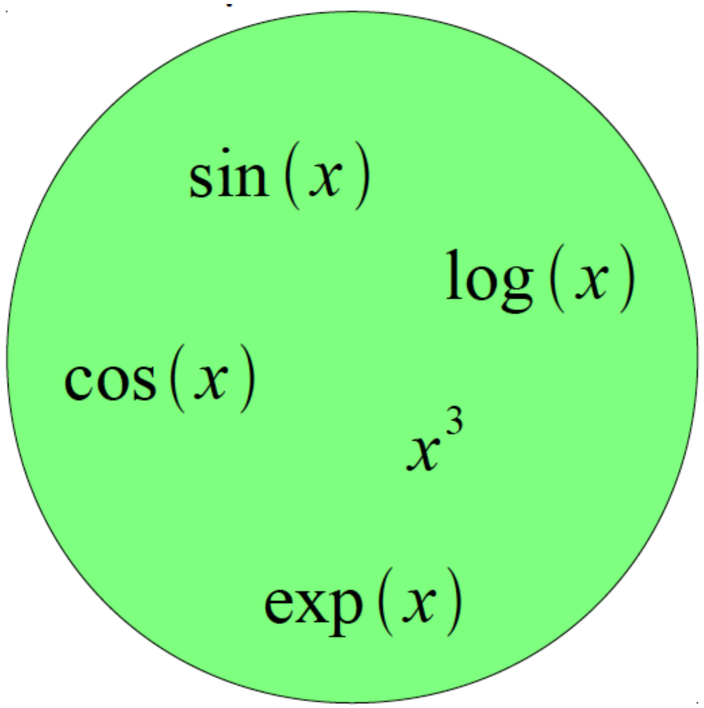
$$f(x) = \sum_i \alpha_i g_i(x)$$



slide by Marc'Aurelio Ranzato, Yann LeCun

Building A Complicated Function

Given a library of simple functions



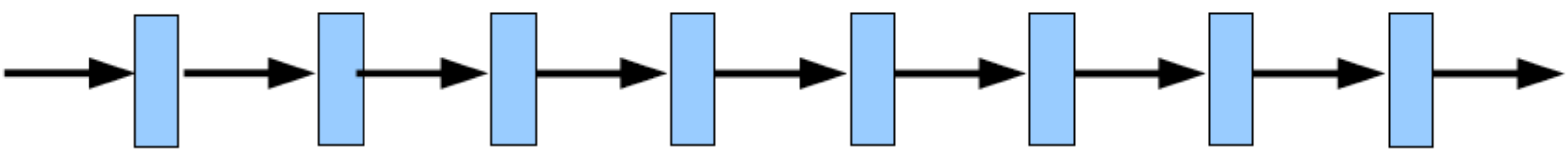
Compose into a

complicate function

Idea 2: Compositions

- Deep Learning
- Grammar models
- Scattering transforms…

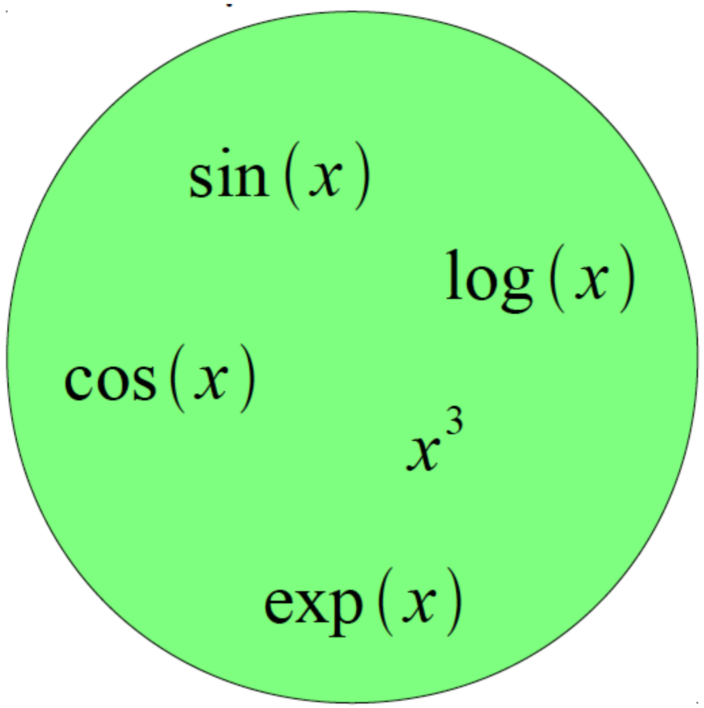
$$f(x) = g_1(g_2(\ldots(g_n(x)\ldots))$$



slide by Marc'Aurelio Ranzato, Yann LeCun

Building A Complicated Function

Given a library of simple functions



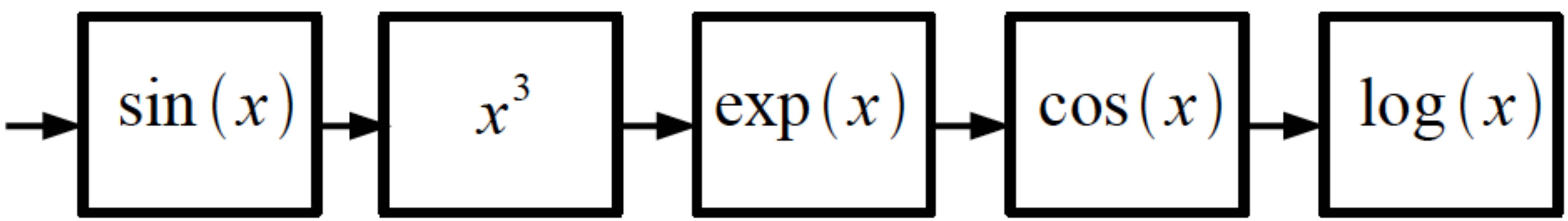
Compose into a

complicate function

Idea 2: Compositions

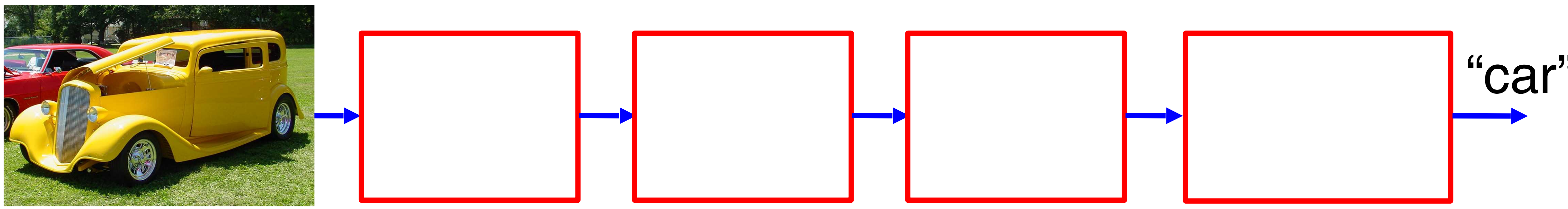
- Deep Learning
- Grammar models
- Scattering transforms…

$$f(x) = \log(\cos(\exp(\sin^3(x))))$$



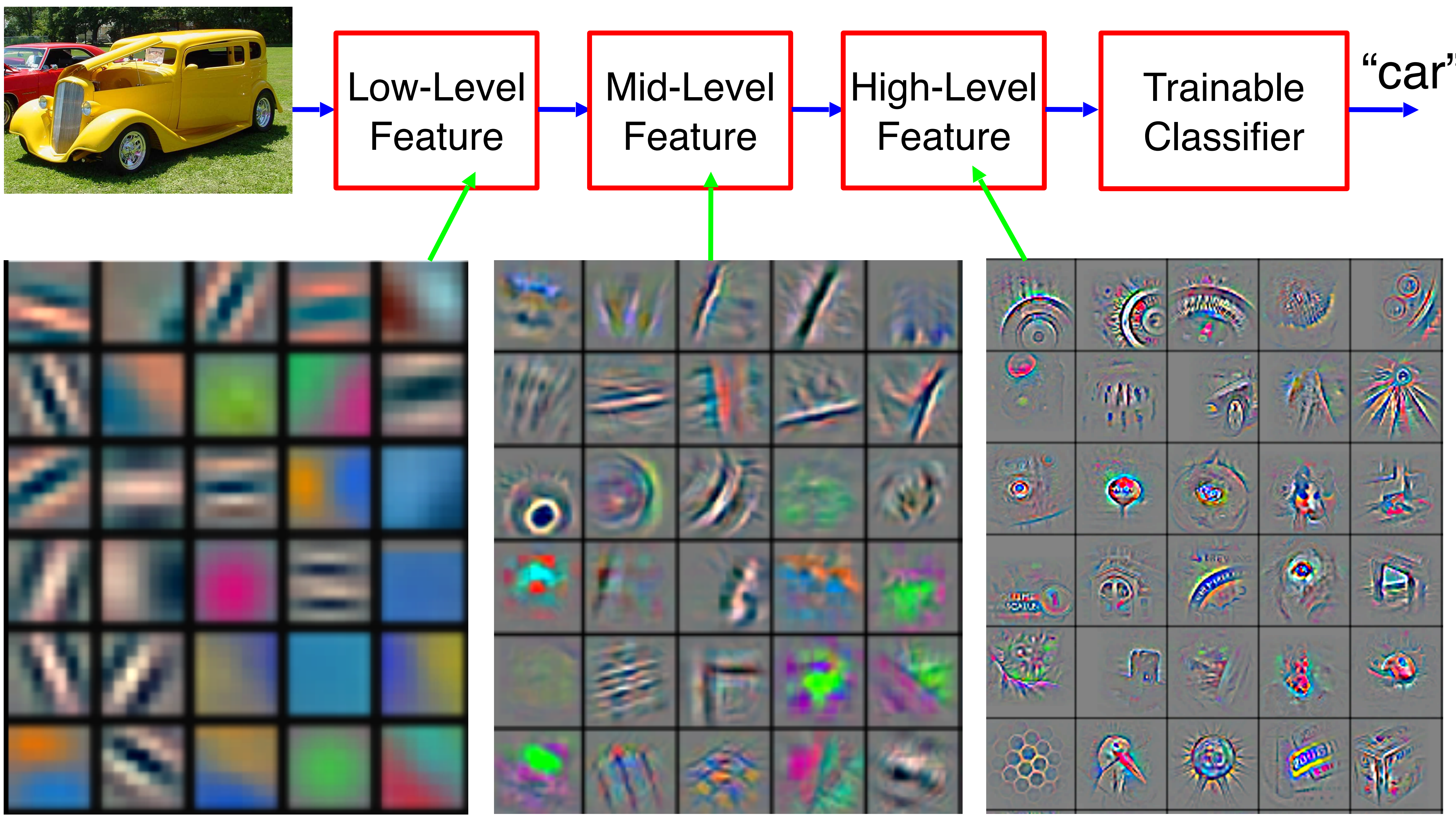
slide by Marc'Aurelio Ranzato, Yann LeCun

Deep Learning = Hierarchical Compositionality



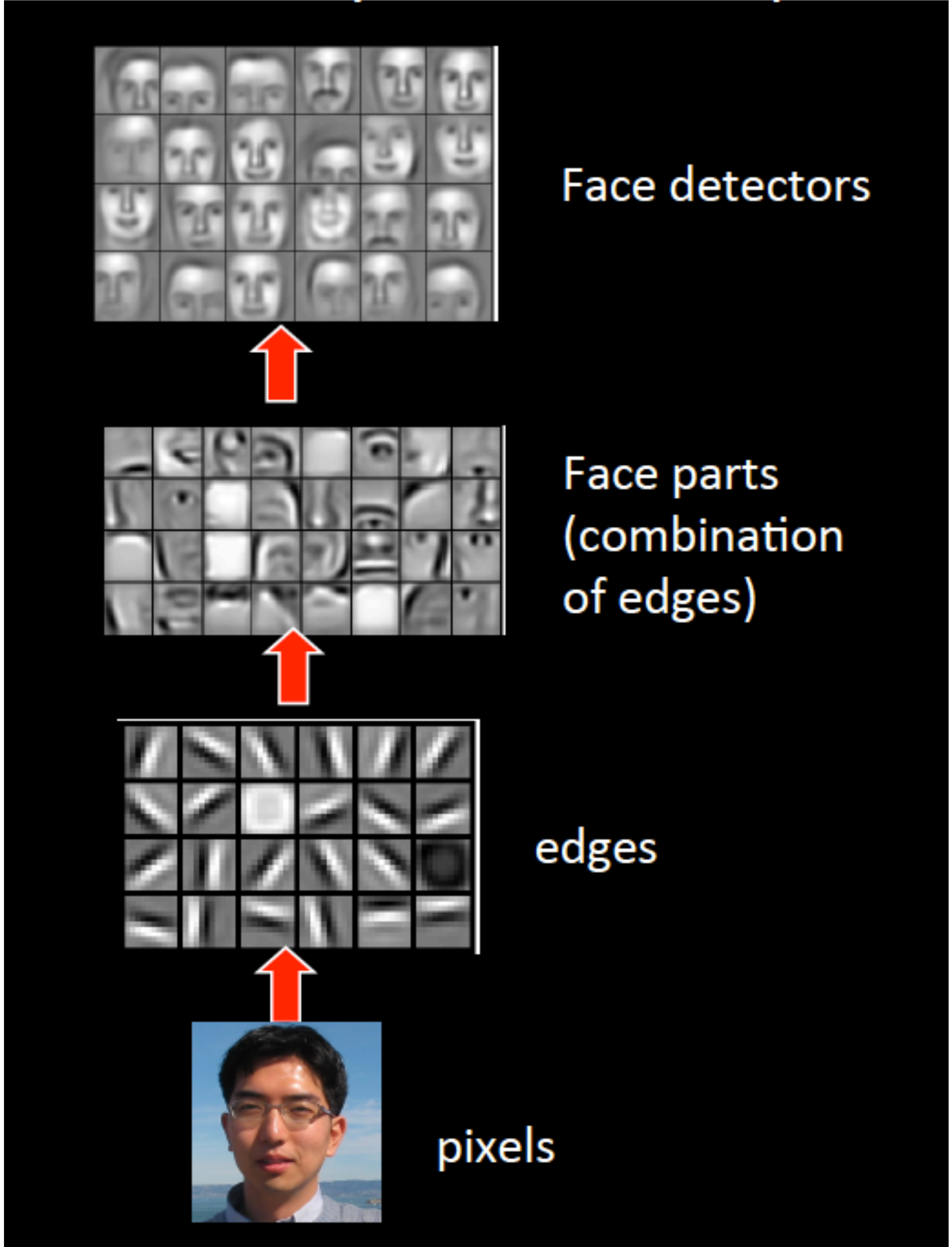
slide by Marc'Aurelio Ranzato, Yann LeCun

Deep Learning = Hierarchical Compositionality



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

slide by Marc’Aurelio Ranzato, Yann LeCun



Sparse DBNs
[Lee et al. ICML '09]
Figure courtesy: Quoc Le

slide by Dhruv Batra

Three key ideas

- (Hierarchical) Compositionality
  - Cascade of non-linear transformations
  - Multiple layers of representations
- **End-to-End Learning**
  - Learning (goal-driven) representations
  - Learning to feature extract
- Distributed Representations
  - No single neuron “encodes” everything
  - Groups of neurons work together

slide by Dhruv Batra

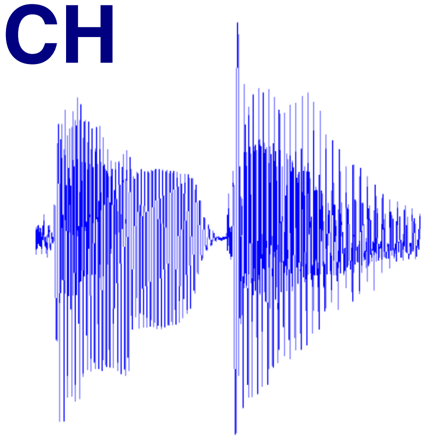
Traditional Machine Learning

VISION

hand-crafted features SIFT/HOG → **your favorite classifier** → “car”

fixed | learned

SPEECH



hand-crafted features MFCC → **your favorite classifier** → \'d ē p\

fixed | learned

NLP

This burrito place is yummy and fun! → **hand-crafted features Bag-of-words** → **your favorite classifier** → “+”

fixed | learned

slide by Marc’Aurelio Ranzato, Yann LeCun

Traditional Machine Learning (more accurately)

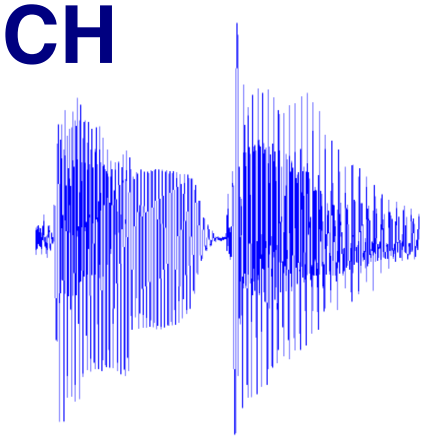
“Learned”

VISION

SIFT/HOG → **K-Means/ pooling** → **classifier** → “car”

fixed | unsupervised | supervised

SPEECH



MFCC → **Mixture of Gaussians** → **classifier** → \ˈd ē p\

fixed | unsupervised | supervised

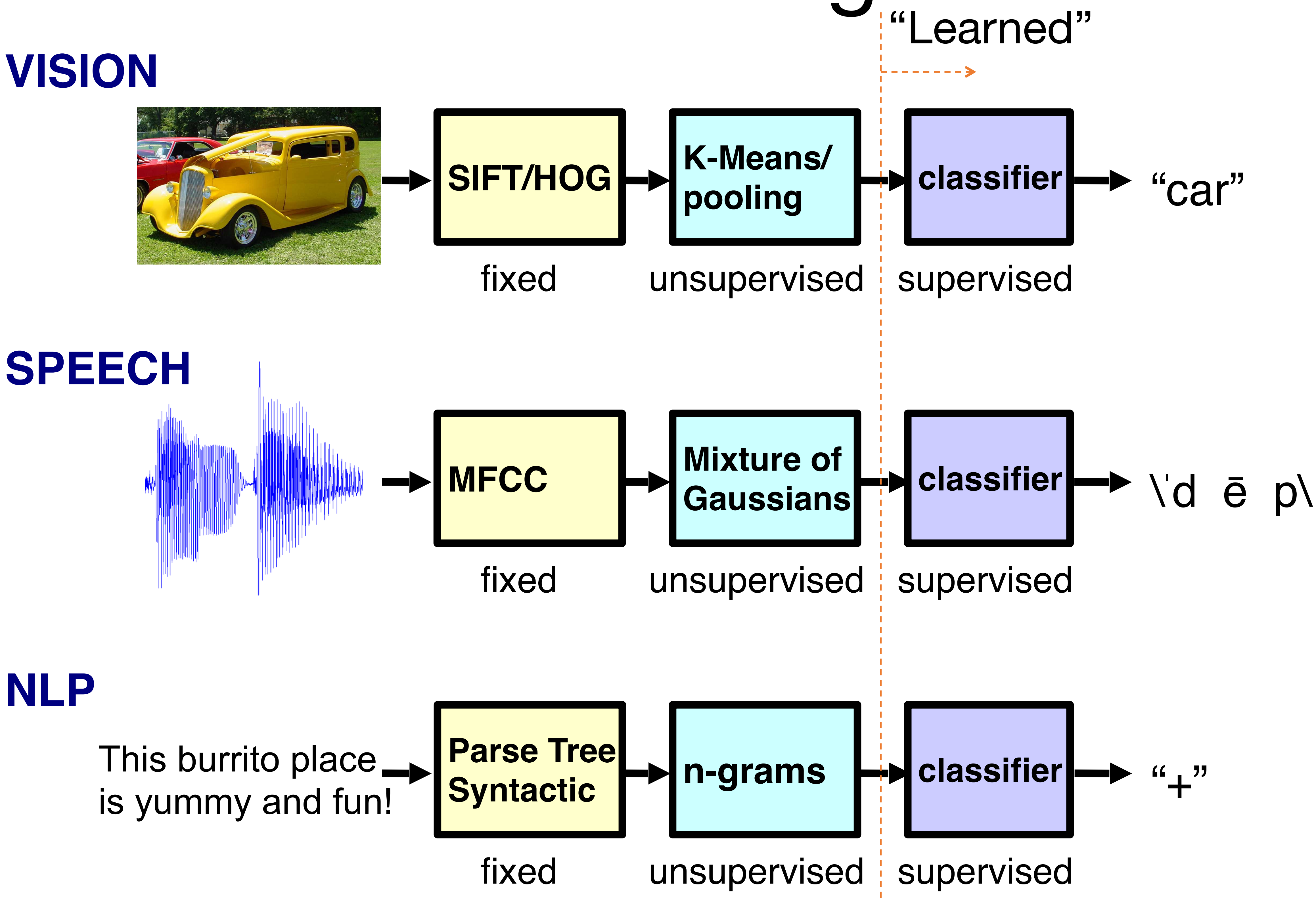
NLP

This burrito place is yummy and fun! → **Parse Tree Syntactic** → **n-grams** → **classifier** → “+”

fixed | unsupervised | supervised

slide by Marc’Aurelio Ranzato, Yann LeCun

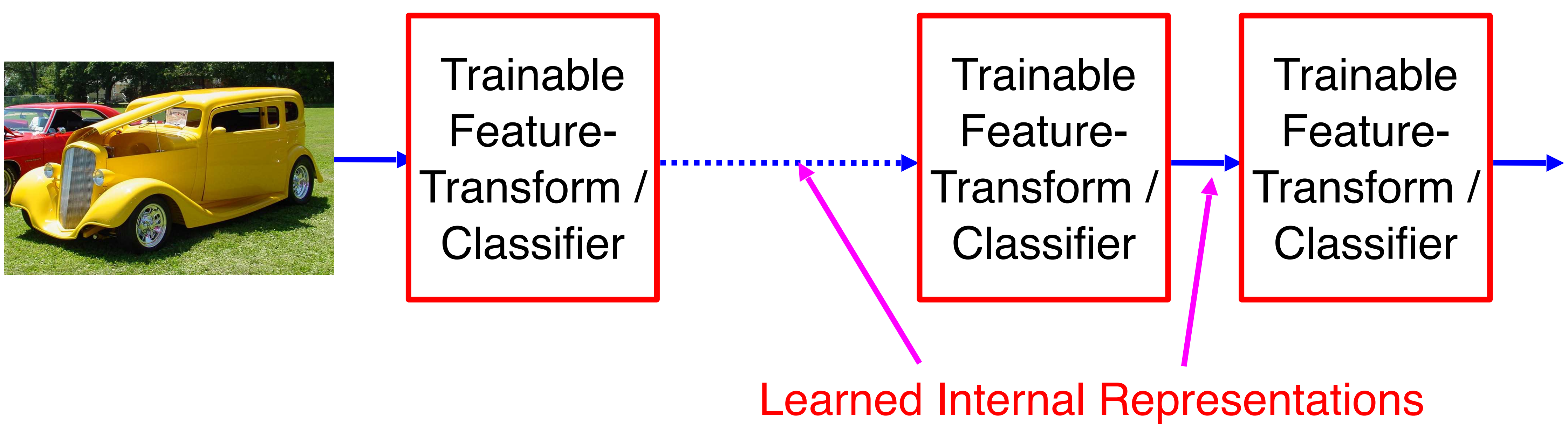
Deep Learning = End-to-End Learning



slide by Marc'Aurelio Ranzato, Yann LeCun

Deep Learning = End-to-End Learning

- A hierarchy of trainable feature transforms
  - Each module transforms its input representation into a higher-level one.
  - High-level features are more global and more invariant
  - Low-level features are shared among categories



slide by Marc'Aurelio Ranzato, Yann LeCun

"Shallow" vs Deep Learning

- "Shallow" models

hand-crafted Feature Extractor

fixed

"Simple" Trainable Classifier

learned

- Deep models

Trainable Feature-Transform / Classifier

Trainable Feature-Transform / Classifier

Trainable Feature-Transform / Classifier

Learned Internal Representations

slide by Marc'Aurelio Ranzato, Yann LeCun

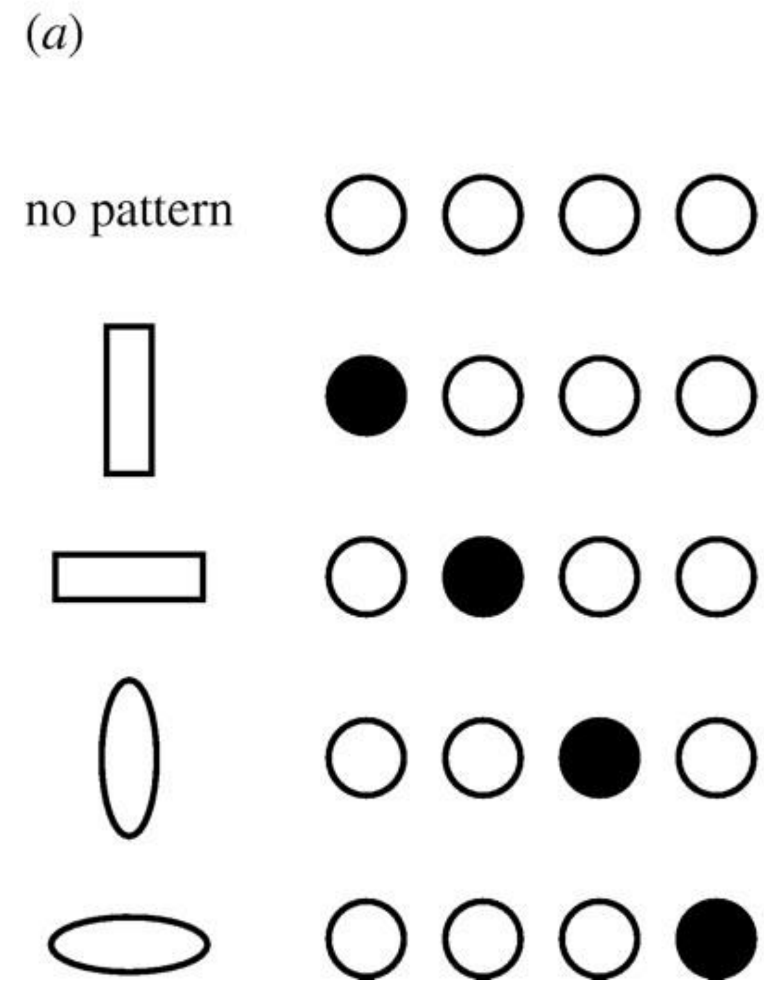
Three key ideas

- (Hierarchical) Compositionality
  - Cascade of non-linear transformations
  - Multiple layers of representations
- End-to-End Learning
  - Learning (goal-driven) representations
  - Learning to feature extract
- **Distributed Representations**
  - No single neuron “encodes” everything
  - Groups of neurons work together

slide by Dhruv Batra

Localist representations

- The simplest way to represent things with neural networks is to **dedicate one neuron to each thing**.
  - Easy to understand.
  - Easy to code by hand
    - Often used to represent inputs to a net
  - Easy to learn
    - This is what mixture models do.
    - Each cluster corresponds to one neuron
  - Easy to associate with other representations or responses.
- But localist models are very inefficient whenever the data has componential structure.

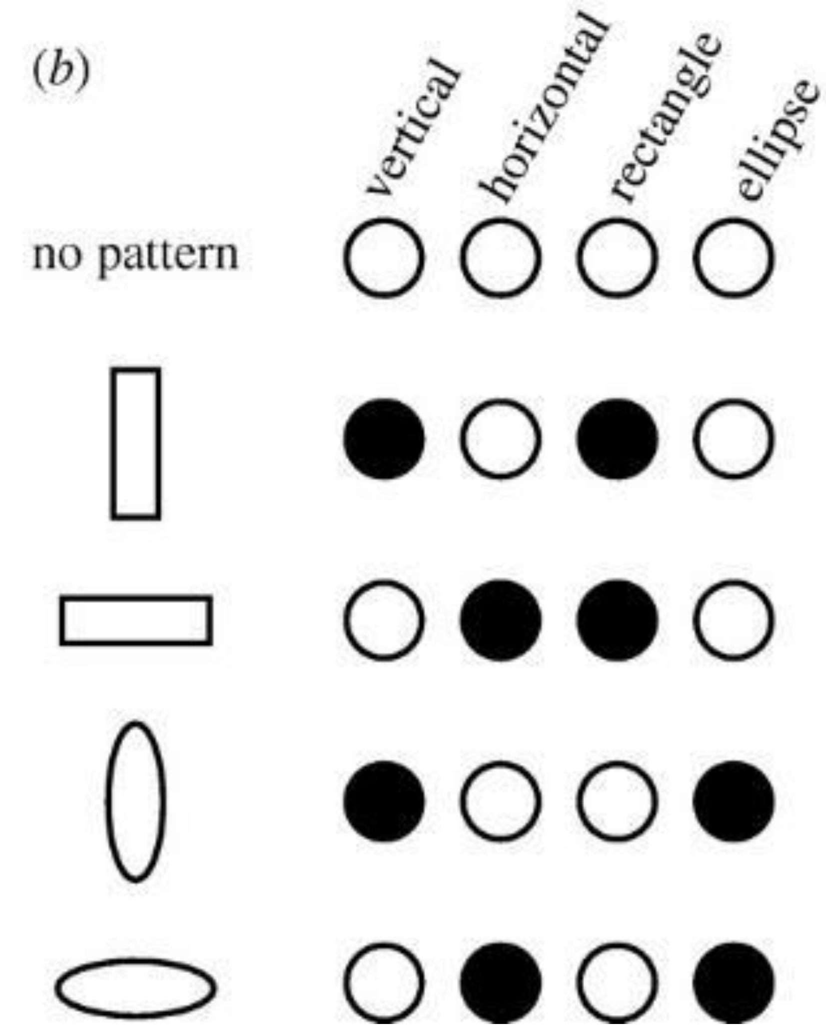


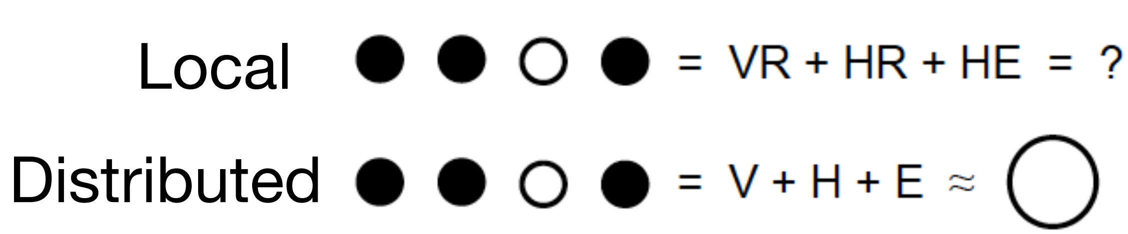
slide by Geoff Hinton

Image credit: Moontae Lee

Distributed Representations

- Each neuron must represent something, so this must be a local representation.
- **Distributed representation** means a many-to-many relationship between two types of representation (such as concepts and neurons).
  - Each concept is represented by many neurons
  - Each neuron participates in the representation of many concepts





slide by Geoff Hinton

Image credit: Moontae Lee

Power of distributed representations!

Scene Classification

bedroom

mountain

- Possible internal representations:
  - Objects
  - Scene attributes
  - Object parts
  - Textures

B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba **“Object Detectors Emerge in Deep Scene CNNs”**, ICLR 2015

slide by Bolei Zhou

Next Lecture:

Convolutional Neural Networks