ID2223 Scalable Machine Learning and Deep Learning

Examiner: Jim Dowling
Associate Prof @ KTH
jdowling@kth.se

Course Assistants
Alex Ormenisan aaor@kth.se
Kamal Hakimzadeh kamal2@kth.se

Climbing Deep Learning Mountain

Machine Translation

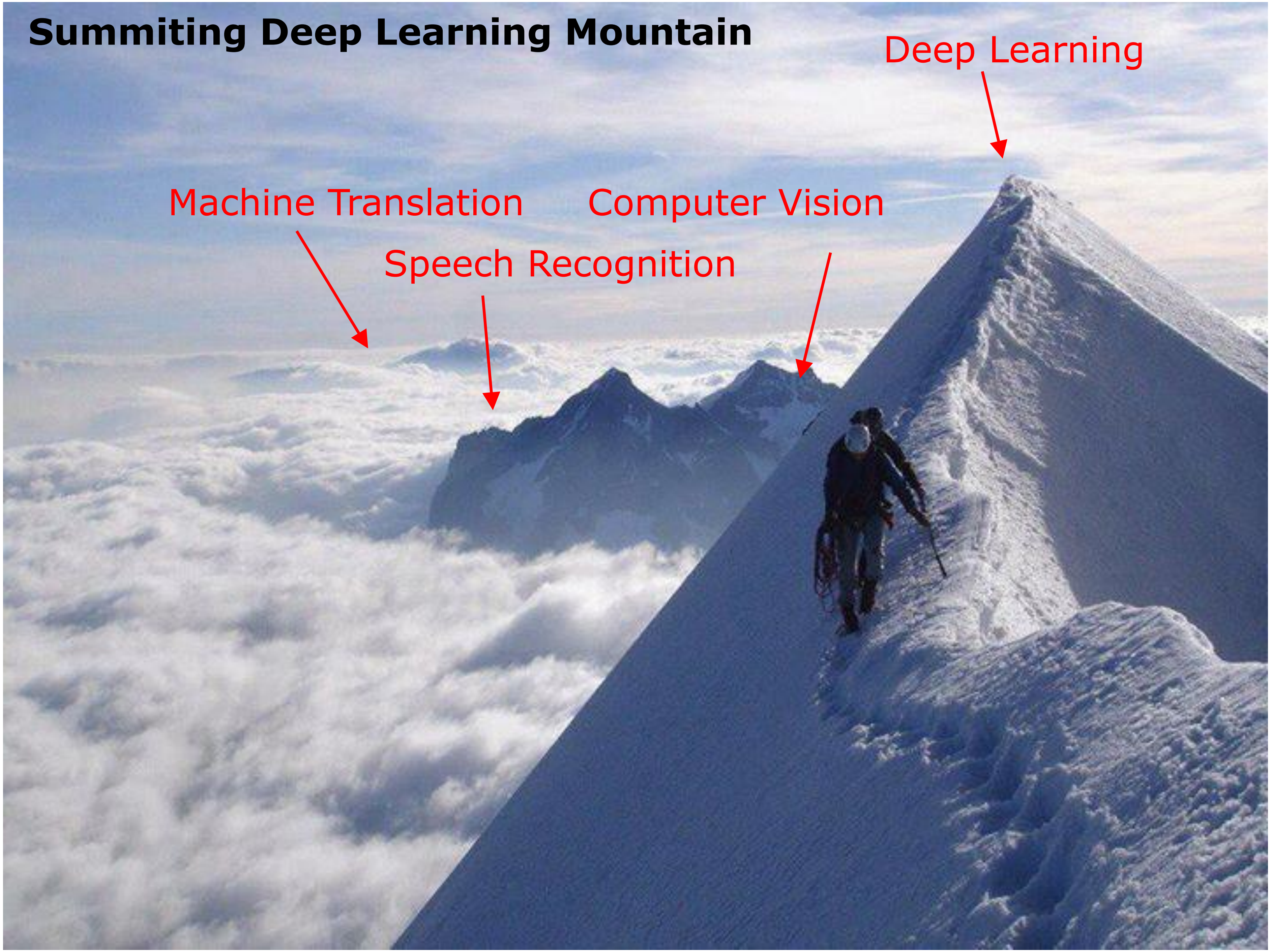
Speech Recognition

Computer Vision

Anomoly Detection

[https://www.pinterest.se/pin/328199891571564768/]

Summiting Deep Learning Mountain



ID2223 Scalable Machine Learning

- Distributed Machine Learning Algorithms
  - Linear Regression, Logistic Regression
  - Spark ML
- Deep Learning
  - Stochastic Gradient Descent
  - Training/Regularization/Optimization
  - Convolutional Neural Networks
  - Recurrent Neural Networks
- Reinforcement Learning
  - Deep Reinforcement Learning

Learning Objectives

- Be able to re-implement a classical machine learning algorithm as a scalable machine learning algorithm
- Be able to design and train a layered neural network system
- Apply a trained layered neural network system to make useful predictions or classifications in an application area
- Be able to elaborate the performance tradeoffs when parallelizing machine learning algorithms as well as the limitations in different network environments
- Be able to identify appropriate distributed machine learning algorithms to efficiently solve classification and pattern recognition problems.

Course Book

- Deep Learning, Yoshua Bengio, Ian Goodfellow and Aaron Courville, MIT Press.
  - Pre-print available on course homepage

- Other course material (large-scale ML, SparkML) gleamed from various sources

Examination

- LAB1 - Programming Assignments, 3.0
  - Lab 1
    - 20 % of coursework grade.
    - Grading will happen on 20th November at “redovisning” time.
  - Lab 2
    - 20 % of coursework grade.
    - Grading will happen on 29th November at “redovisning” time.
  - Project
    - 60% of coursework grade. Grading will happen in early January.
- LAB1 passing grade: 50% or more from any combination of labs and the project
- Examination, 4.5, grade scale: A, B, C, D, E, FX, F

Labs/Project

- Self-selected Groups of 2 (group of 1 ok) for labs.
- Self-selected Groups of 2-4 for the project.

- Labs will include Scala/Python programming
  - Spark ML – Graded on 18th November at lab
  - Tensorflow Python – Graded on 30th November at lab

- Project
  - Selection of a large dataset and Method (Deep Learning):
    - Dec 12th – project discussion session.
    - Dec 15th – project description due.
    - Early/mid January – demonstrated as a demo and short report delivered to Canvas.

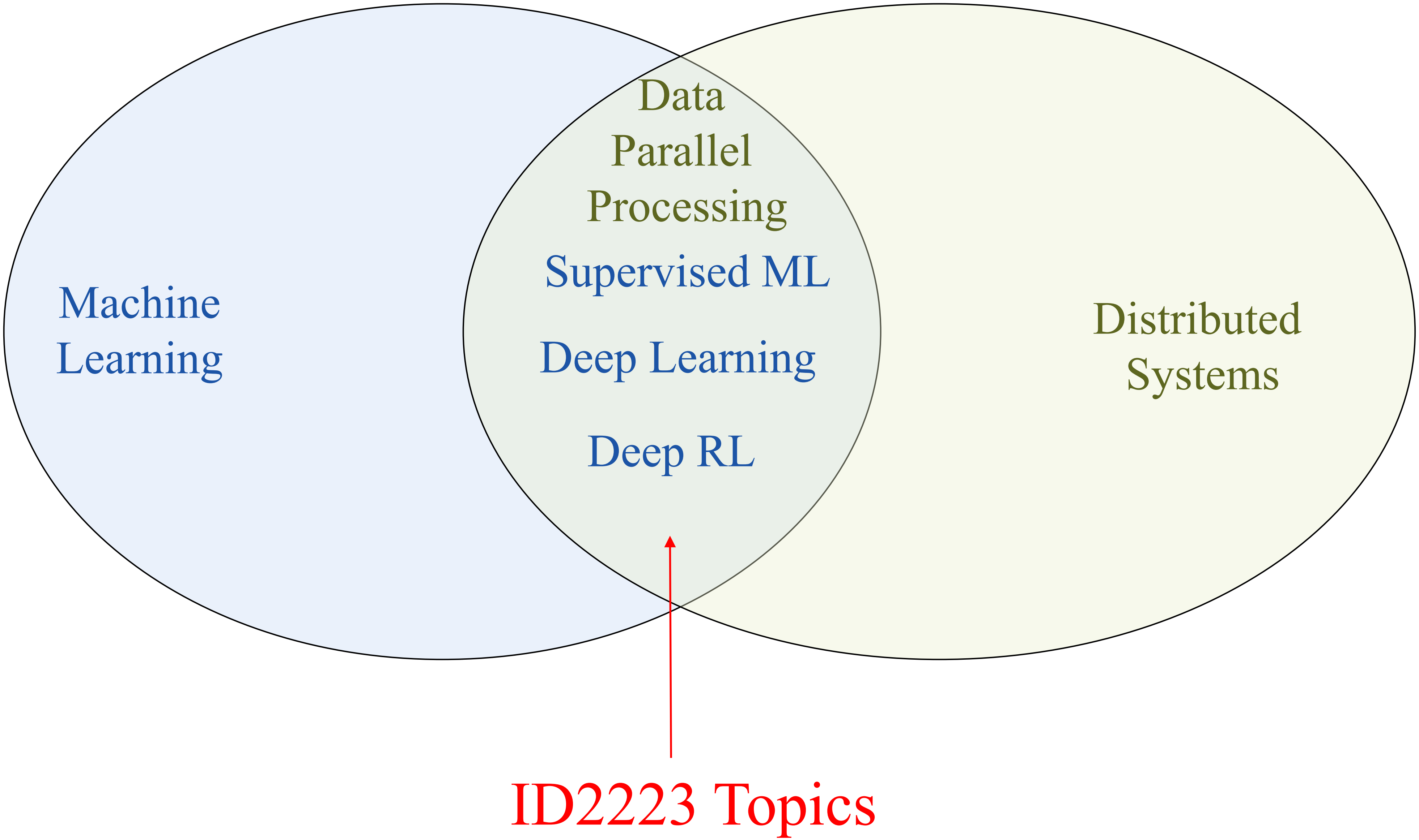
Large Data Sets Available on SICS ICE

- SICS ICE
  - 36 node Hadoop Cluster
  - Nvidia GTX 1080

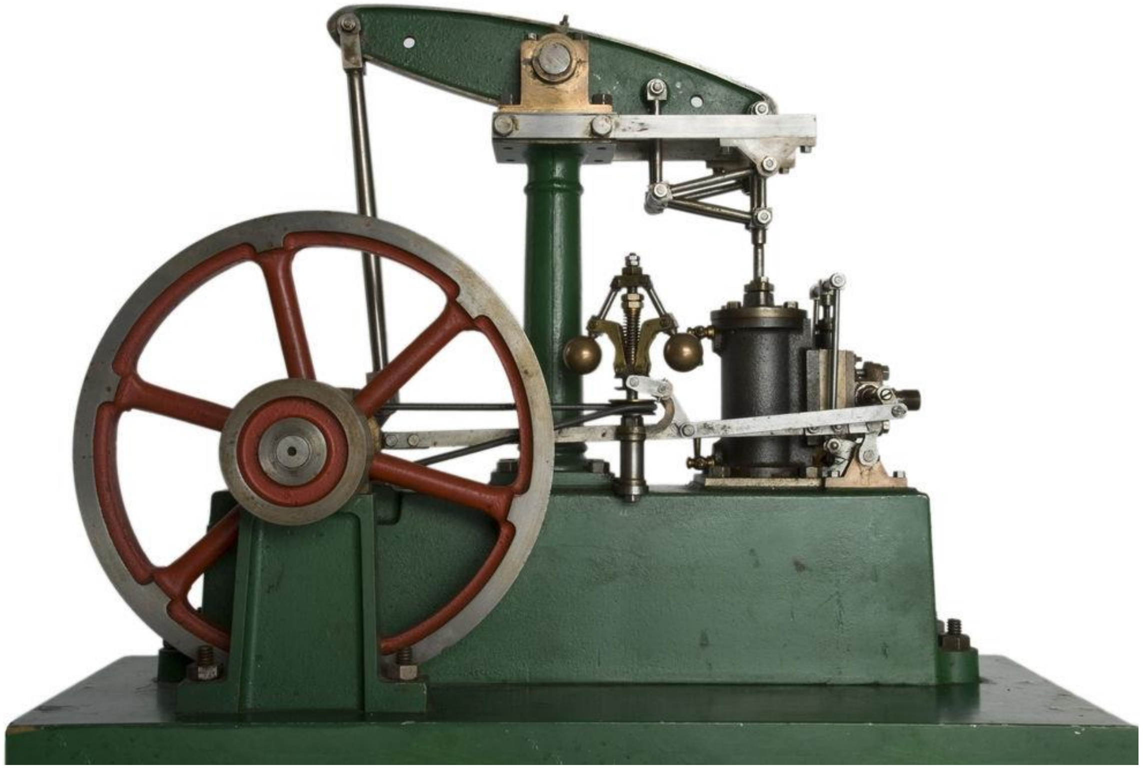
- www.hops.site
  - Spark
  - Tensorflow
  - Large Data Sets
  - Notebooks:
    - Zeppelin and Jupyter

SICS ICE: A datacenter research and test environment

Scalable Machine Learning



Deep Learning is the new Steam Engine

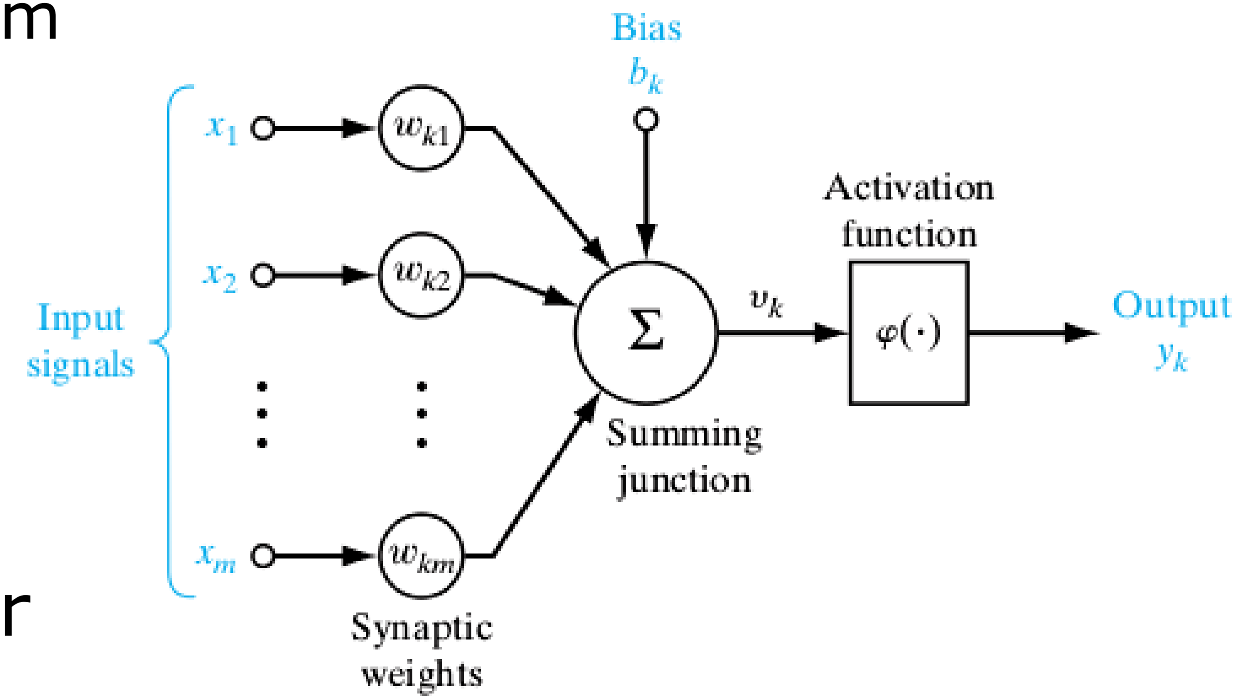


- 1765 Water Pump
- 1819 Steamship
- 1825 Locomotive
- 1852 Airship

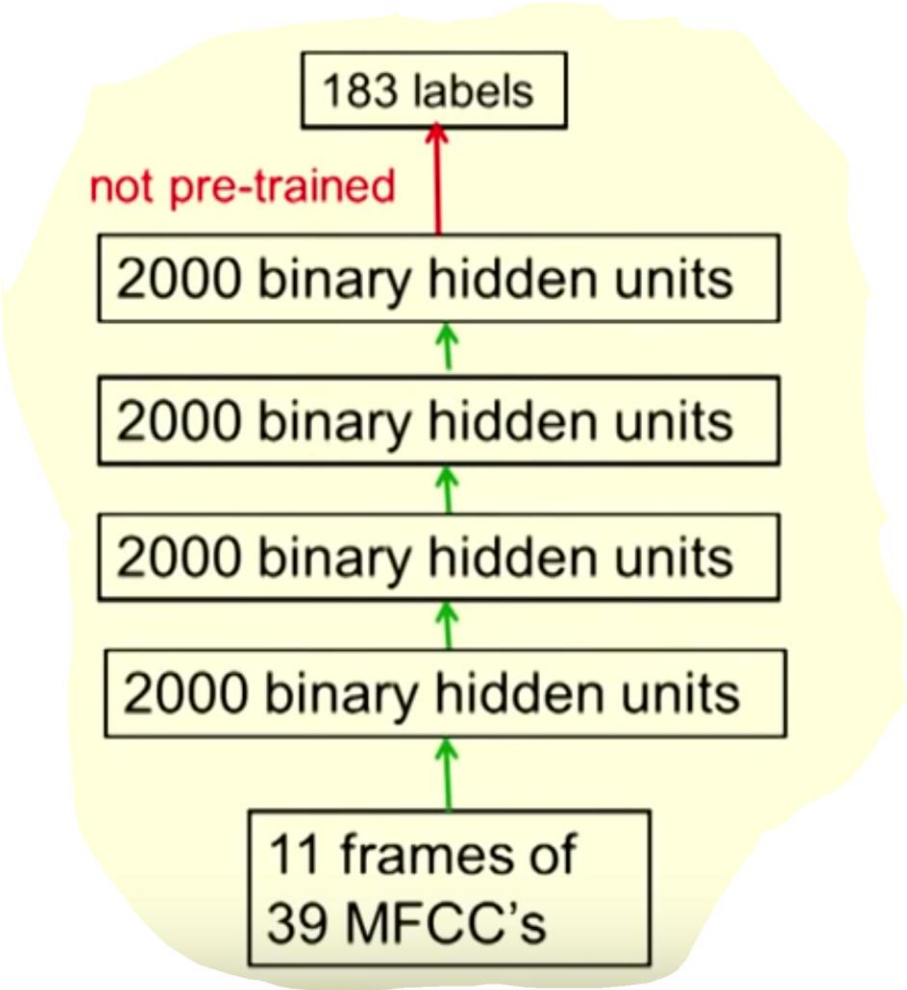
Brief History of Deep Learning

- 1950s/94s
  - Perceptron, XOR Problem
- 1980s
  - Backpropagation
- 1990s
  - Le Cun’s LENET-5

- Then the 2nd AI Winter until….

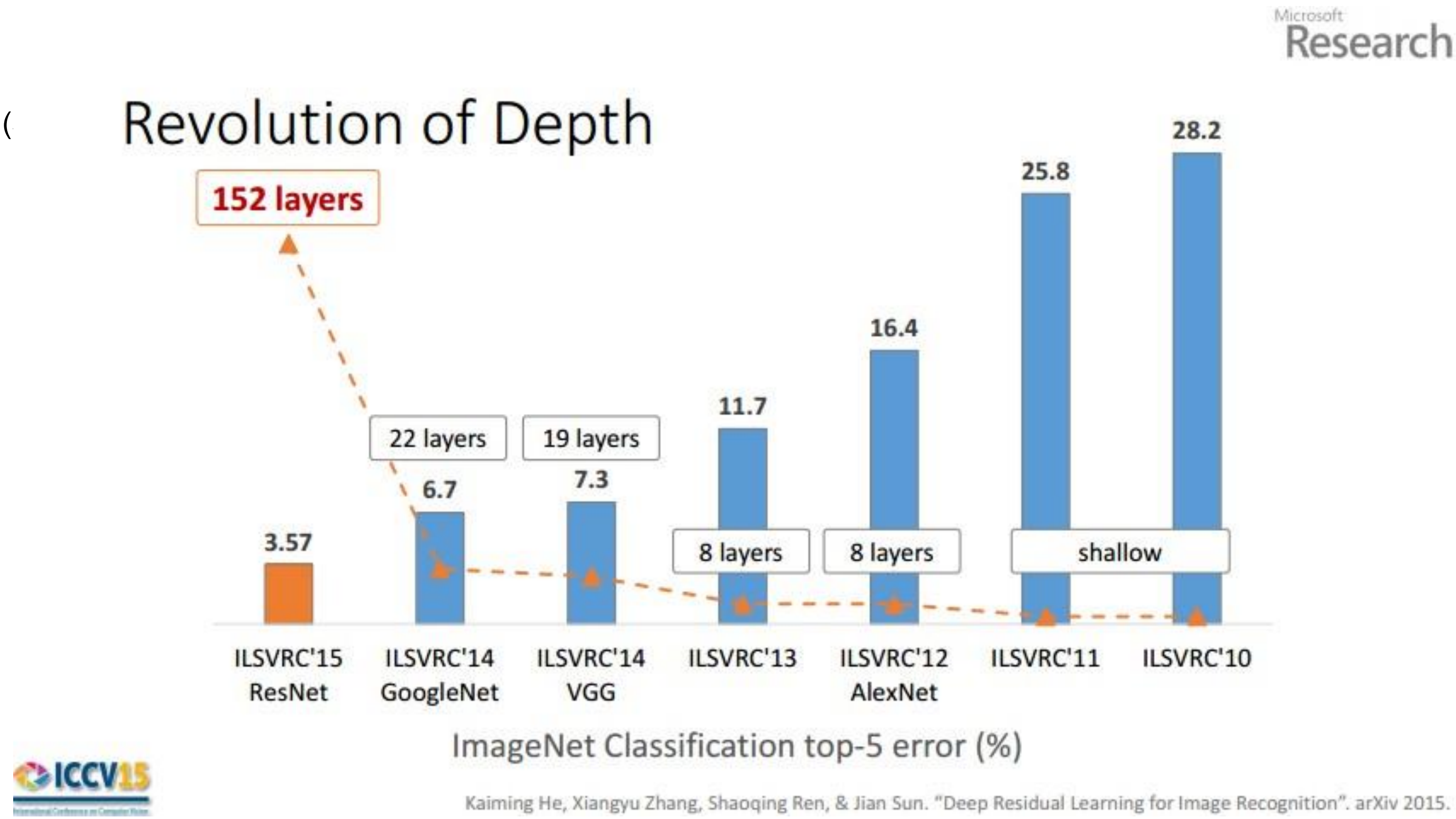


2009 Speech Recognition



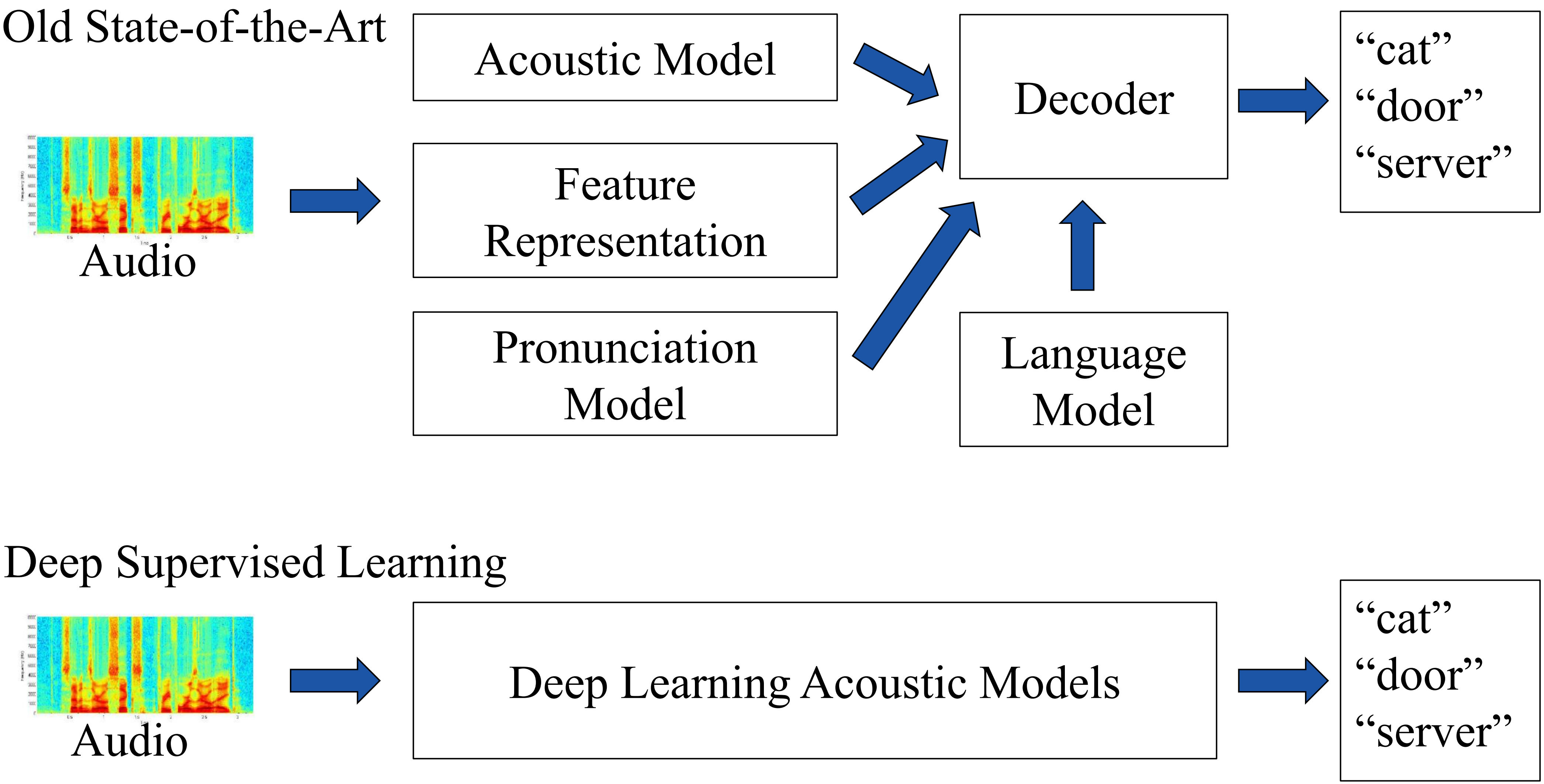
- Acoustic modelling with a pre-trained deep neural net (Mohamed, Dahl, and Hinton, 2009)
- 23% phone error rate vs previous best of 24.4% on TIMIT
- By 2012, Android's acoustic model was a DL network

2012 Image Recognition



[slide from Kaiming He]

End-to-End Deep Learning (Speech)



End-to-End Deep Learning (Vision)

State-of-the-Art 2012

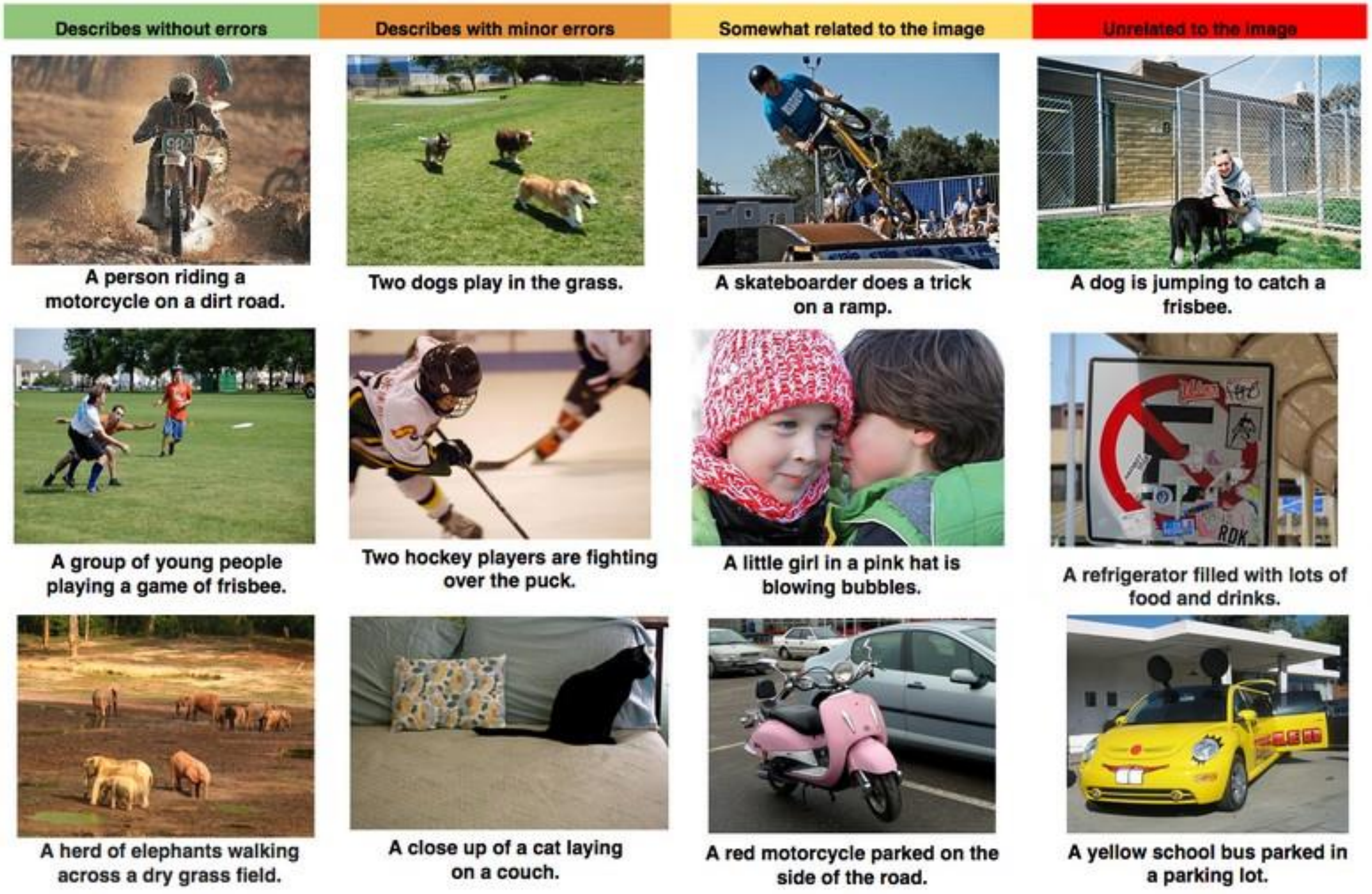
→ Hand Engineered Features → Support Vector Machine → "cat" "door" "server"

AlexNet 2012: Deep Supervised Learning

→ 8-layer NN with 60 million params → "cat" "door" "server"

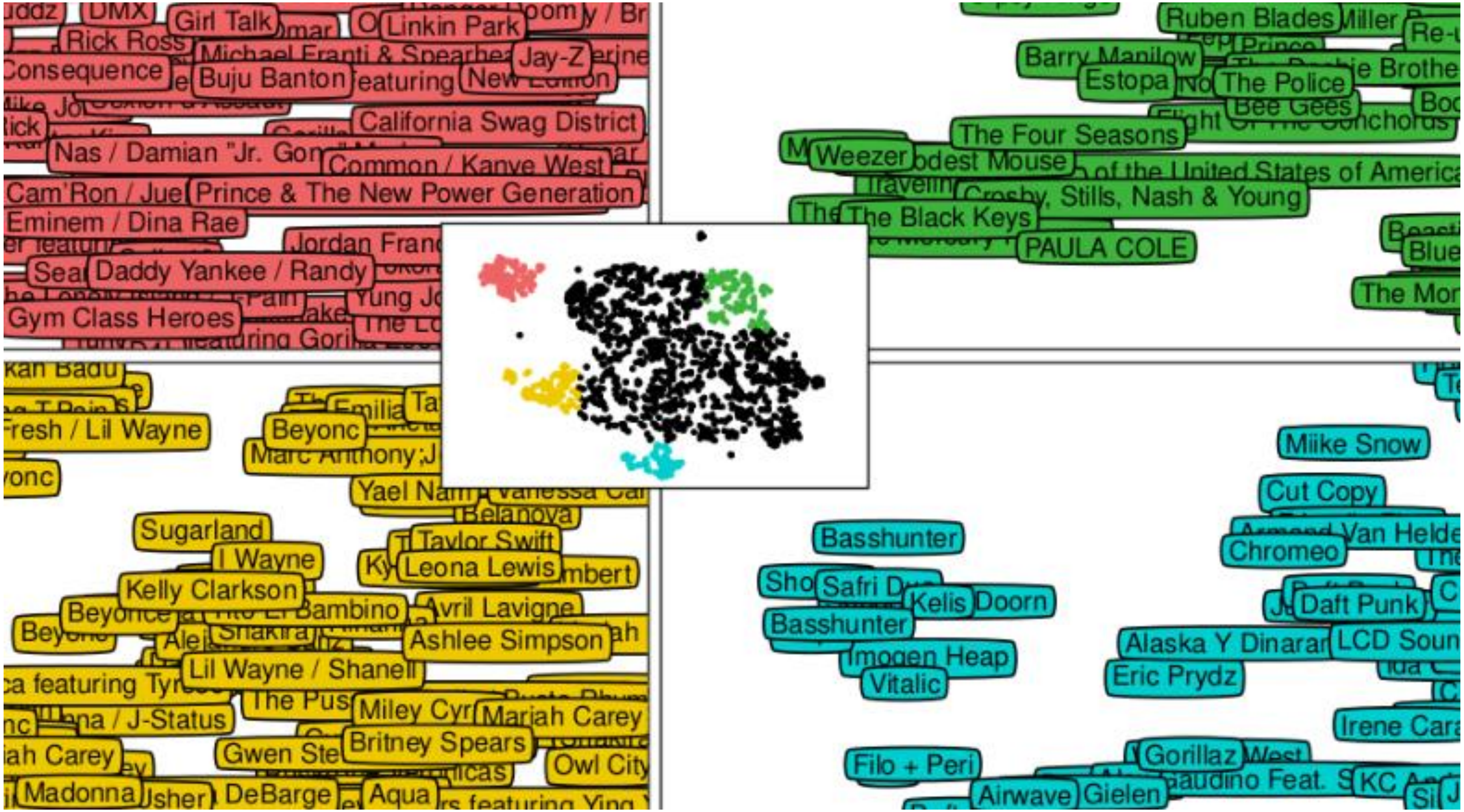
1.2 million training images from ImageNet

Automated Image Captioning



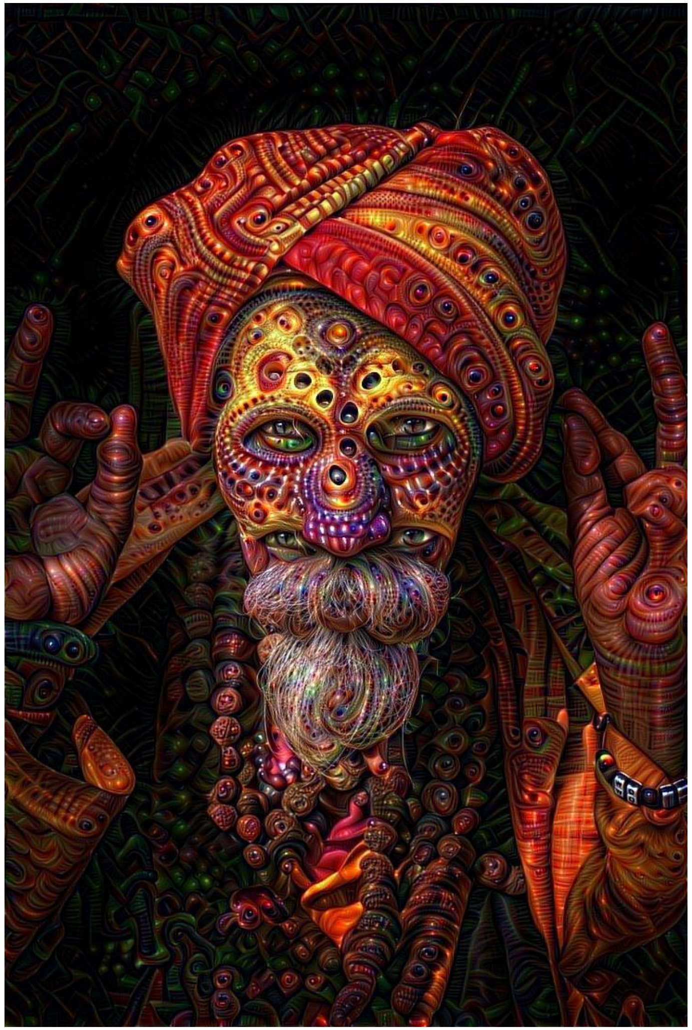
[Image captioning, Vinyals et al. 2015]

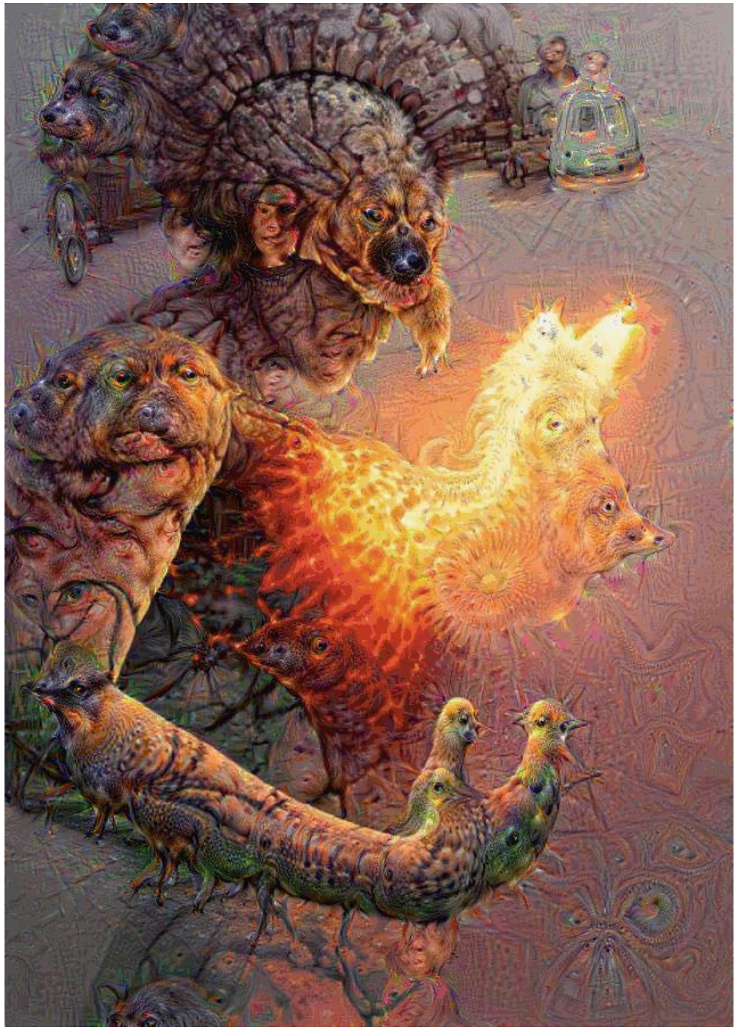
Convnets for Music Recommendation



[Recommending Music on Spotify with Deep Learning. Sander Dieleman]

Convnets for Art

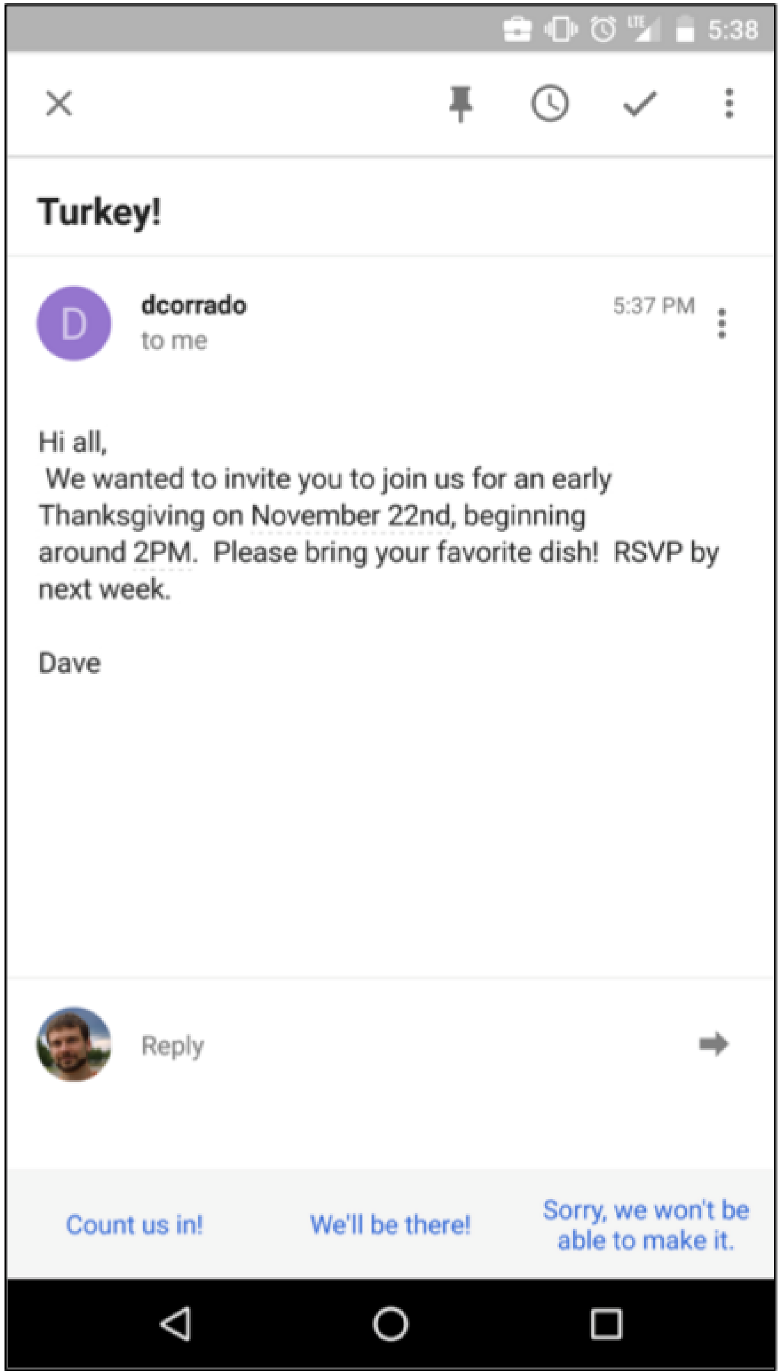


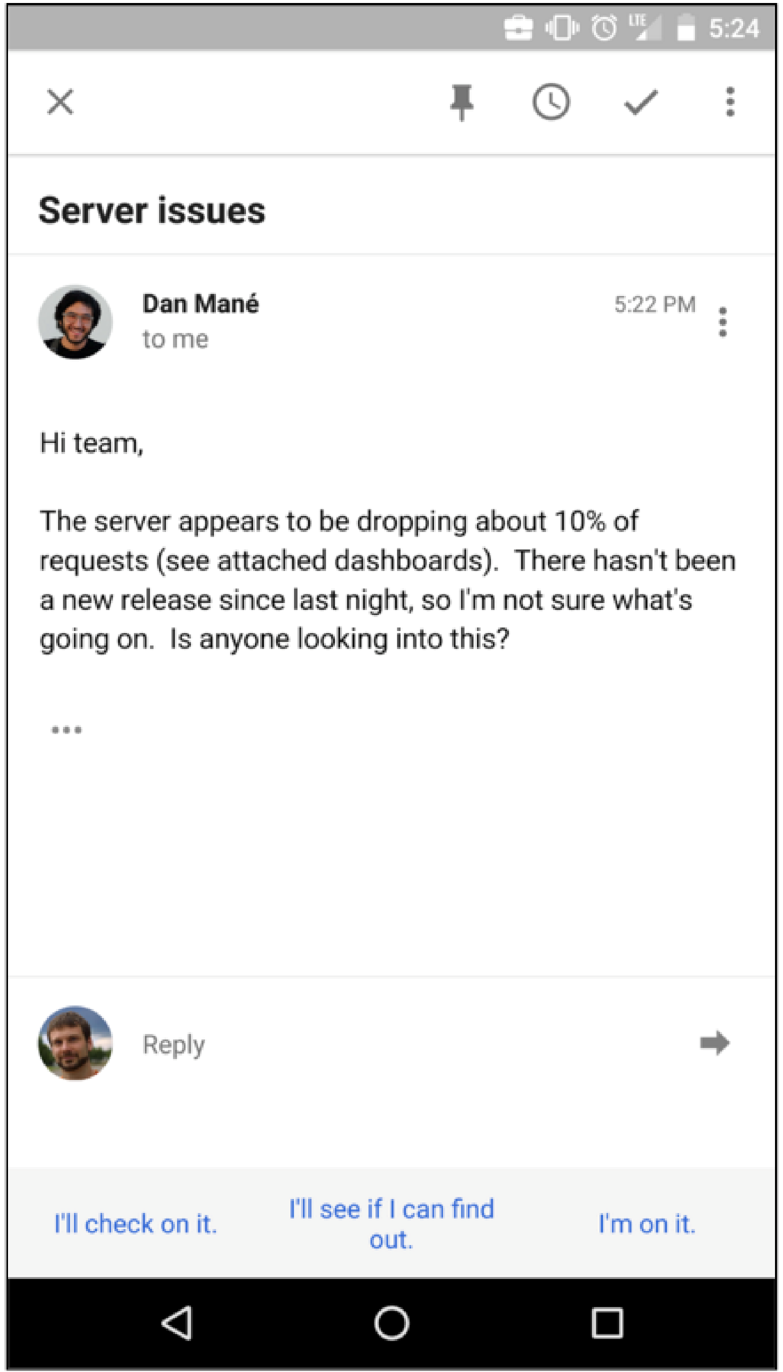


DeepDream *reddit.com/r/deepdream*

NeuralStyle, Gatys et al. 2015
deepart.io, Prisma, etc.

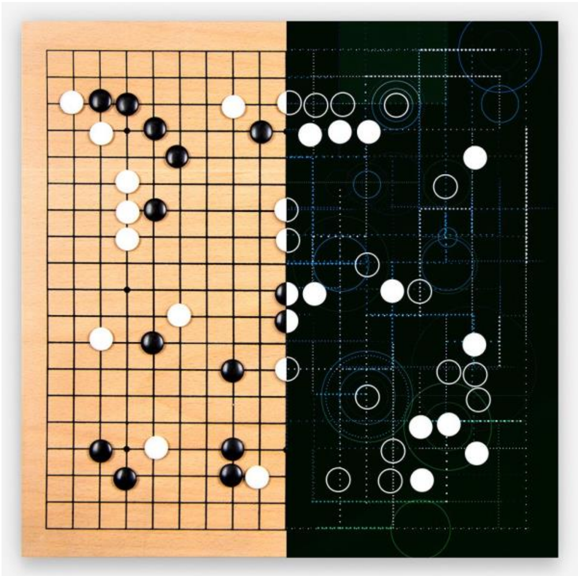
Email Smart Reply with RNNs





DeepRL for Playing Games

ATARI game playing, Mnih 2013



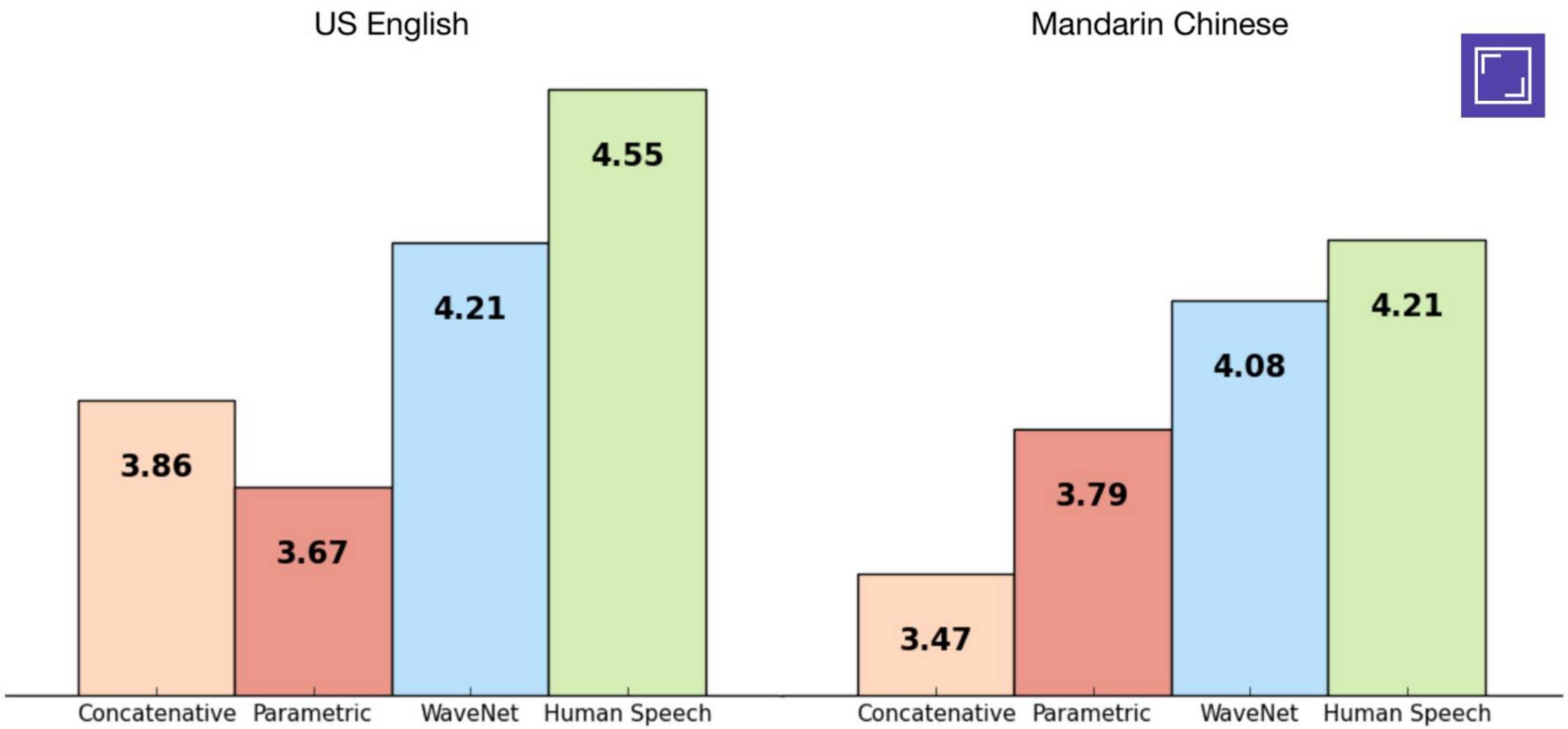
AlphaGo

AlphaGo, Silver et al 2016

Self-Driving Cars

DeepMind WaveNet

- A deep generative model of raw audio waveforms
  - Has to be heard to be believed



https://deepmind.com/blog/wavenet-generative-model-raw-audio/

Learning Large-Scale ML

Large-Scale Machine Learning at Google

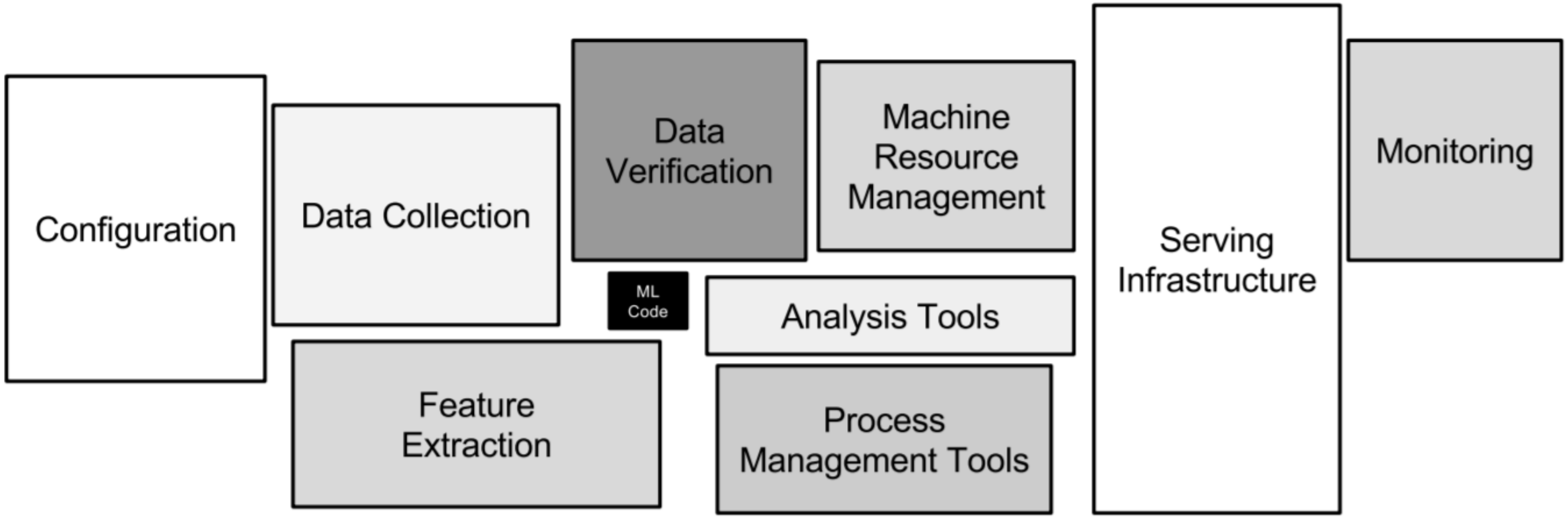


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

[Hidden Technical Debt in Machine Learning Systems, Schulley et Al, NIPS 2015]

Large Scale Machine Learning in Industry

- Machine learning is key to every part of our business, from image recognition, to advertising targeting, to search rankings, to abuse detection, to personalization…
- Instead of just using a “click” as the basic unit of engagement, machine learning enables us to track exactly how long a person spends reading an article, or if they are reading related stories…..

- Peter Cnudde, VP of Engineering, Yahoo

https://www.infoq.com/articles/peter-cnudde-yahoo-big-data

Large Scale Machine Learning in Industry

- We developed a distributed word embedding algorithm to match user queries against ads with similar semantic vectors, instead of traditional syntactic matching….
- Deep learning powers Flickr's scene detection, object recognition, and computational aesthetics…..
- With Esports, we detect game highlights automatically…

- Peter Cnudde, VP of Engineering, Yahoo

https://www.infoq.com/articles/peter-cnudde-yahoo-big-data

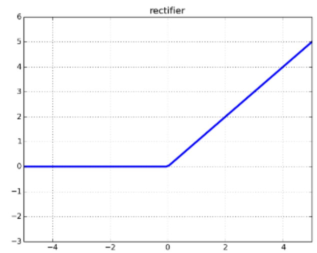
What Changed?

What changed?

Data

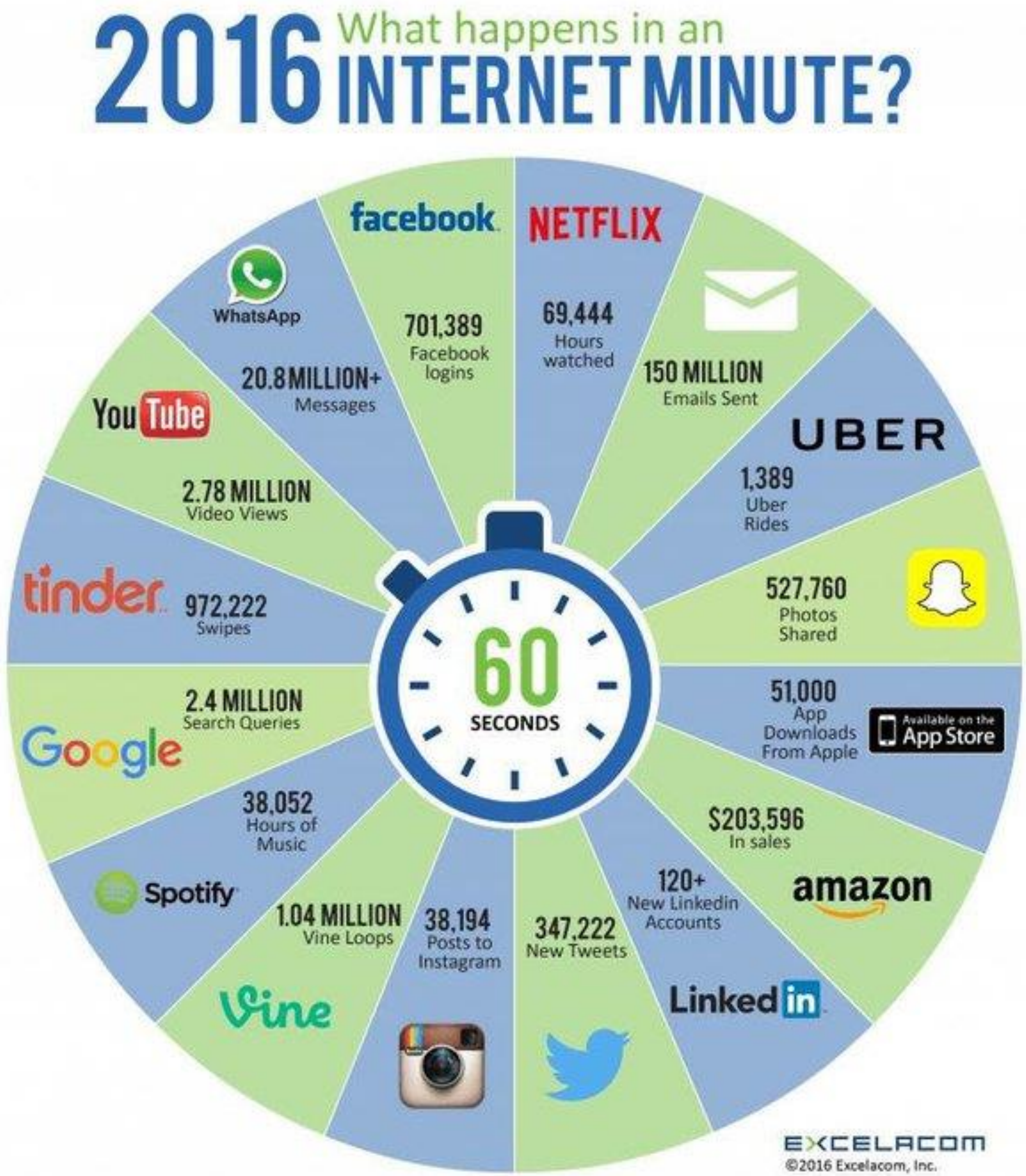
GPUs

Weight Initialization

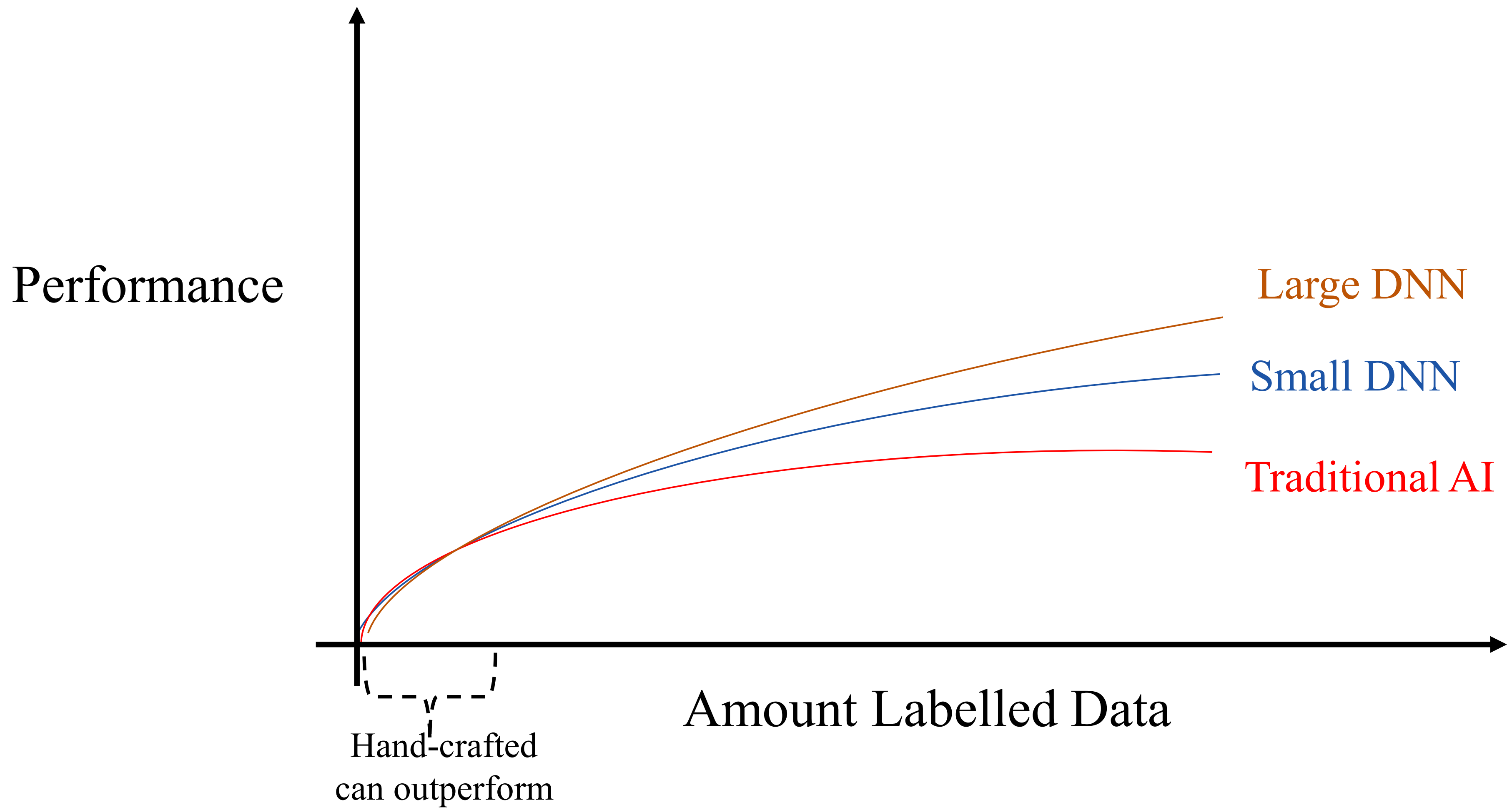


Non-Linearity

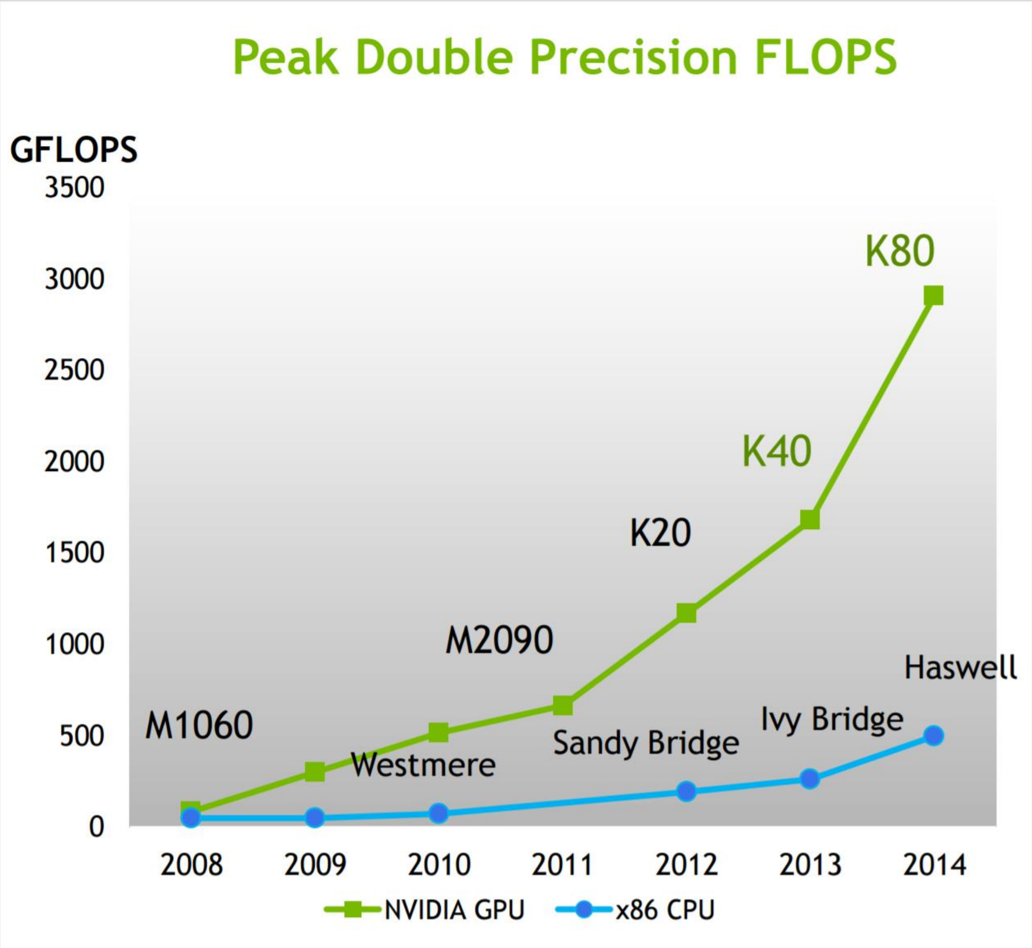
Increasing Data Volumes

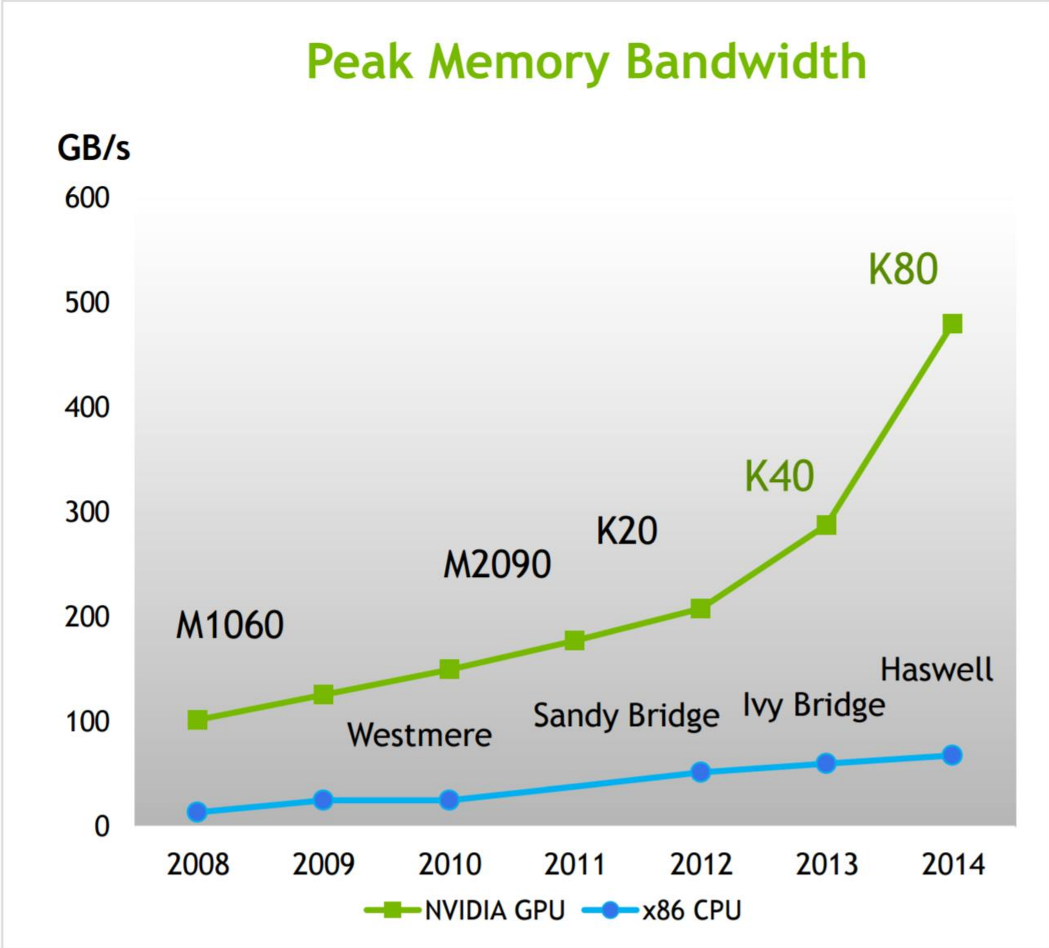


More data means Bigger DNN Models



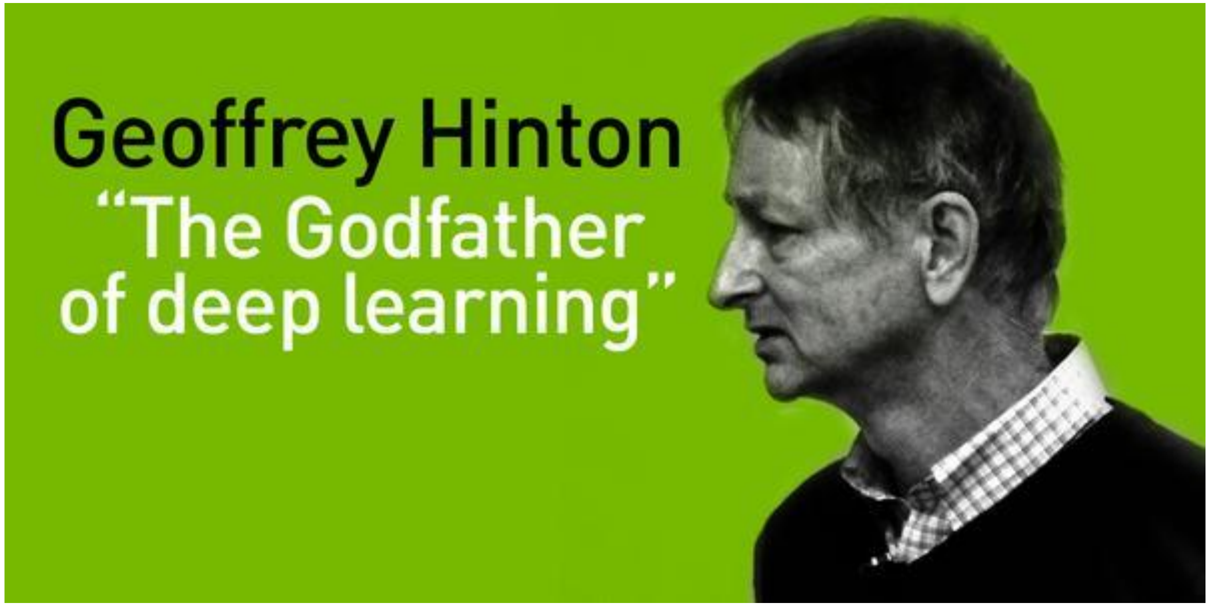
Graphical Processing Units (GPUs)

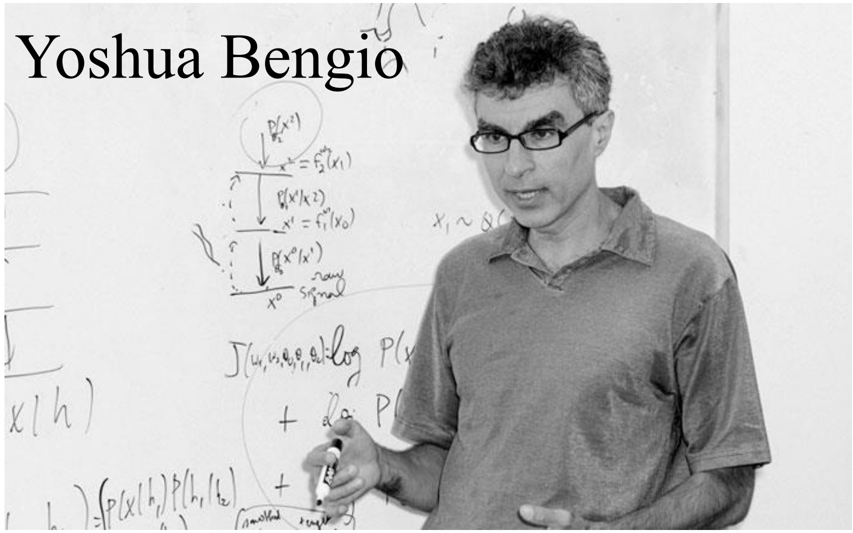




[nvidia.com]

What changed?





CIFAR

Combining Systems and AI Research

- To build bigger models with more data, we need systems experts
- Jeff Dean: expert systems researcher led the development of DistBelief
- OpenAI, Baidu, Google, Microsoft, Facebook all organized with collaborating systems and AI teams

Machine Learning Papers have Changed

4.1. Image Features and Kernels

We selected or designed several state-of-art features that are potentially useful for scene classification. GIST features [21] are proposed specifically for scene recognition tasks. Dense SIFT features are also found to perform very well at the 15-category dataset [17]. We also evaluate sparse SIFTs as used in "Video Google" [27]. HOG features provide excellent performance for object and human recognition tasks [4, 9], so it is interesting to examine their utility for scene recognition. While SIFT is known to be very good at finding repeated image content, the self-similarity descriptor (SSIM) [26] relates images using their internal layout of local self-similarities. Unlike GIST, SIFT, and HOG, which are all local gradient-based approaches, SSIM may provide a distinct, complementary measure of scene layout that is somewhat appearance invariant. As a baseline, we also include Tiny Images [28], color histograms and straight line histograms. To make our color and texton histograms more invariant to scene layout, we also build histograms for specific geometric classes as determined by [13]. The geometric classification of a scene is then itself used as a feature, hopefully being invariant to appearance but responsive to layout.

GIST: The GIST descriptor [21] computes the output energy of a bank of 24 filters. The filters are Gabor-like filters tuned to 8 orientations at 4 different scales. The square output of each filter is then averaged on a 4×4 grid.

HOG2x2: First, histogram of oriented edges (HOG) descriptors [4] are densely extracted on a regular grid at steps of 8 pixels. HOG features are computed using the code available online provided by [9], which gives a 31-dimension descriptor for each node of the grid. Then, 2×2 neighboring HOG descriptors are stacked together to form a descriptor with 124 dimensions. The stacked descriptors spatially overlap. This 2×2 neighbor stacking is important because the higher feature dimensionality provides more descriptive power. The descriptors are quantized into 300 visual words by k-means. With this visual word representation, three-level spatial histograms are computed on grids of 1×1, 2×2 and 4×4. Histogram intersection[17] is used to define the similarity of two histograms at the same pyramid level for two images. The kernel matrices at the three levels are normalized by their respective means, and linearly combined together using equal weights.

Dense SIFT: As with HOG2x2, SIFT descriptors are densely extracted [17] using a flat rather than Gaussian window at two scales (4 and 8 pixel radii) on a regular grid at steps of 5 pixels. The three descriptors are stacked together for each HSV color channels, and quantized into 300 visual words by k-means, and spatial pyramid histograms are used as kernels[17].

LBP: Local Binary Patterns (LBP) [20] is a powerful texture feature based on occurrence histogram of local binary patterns. We can regard the scene recognition as a texture classification problem of 2D images, and therefore apply this model to our problem. We also try the rotation invariant extension version [2] of LBP to examine whether rotation invariance is suitable for scene recognition.

Sparse SIFT histograms: As in "Video Google" [27], we build SIFT features at Hessian-affine and MSER [19] interest points. We cluster each set of SIFTs, independently, into dictionaries of 1,000 visual words using k-means. An image is represented by two histograms counting the number of sparse SIFTs that fall into each bin. An image is represented by two 1,000 dimension histograms where each SIFT is soft-assigned, as in [22], to its nearest cluster centers. Kernels are computed with χ^2 distance.

SSIM: Self-similarity descriptors [26] are computed on a regular grid at steps of five pixels. Each descriptor is obtained by computing the correlation map of a patch of 5×5 in a window with radius equal to 40 pixels, then quantizing it in 3 radial bins and 10 angular bins, obtaining 30 dimensional descriptor vectors. The descriptors are then quantized into 300 visual words by k-means and we use χ^2 distance on spatial histograms for the kernels.

Tiny Images: The most trivial way to match scenes is to compare them directly in color image space. Reducing the image dimensions drastically makes this approach more computationally feasible and less sensitive to exact alignment. This method of image matching has been examined thoroughly by Torralba et al.[28] for the purpose of object recognition and scene classification.

Line Features: We detect straight lines from Canny edges using the method described in Video Compass [15]. For each image we build two histograms based on the statistics of detected lines– one with bins corresponding to line angles and one with bins corresponding to line lengths. We use an RBF kernel to compare these unnormalized histograms. This feature was used in [11].

Texton Histograms: We build a 512 entry universal texton dictionary [18] by clustering responses to a bank of filters with 8 orientations, 2 scales, and 2 elongations. For each image we then build a 512-dimensional histogram by assigning each pixel's set of filter responses to the nearest texton dictionary entry. We compute kernels from normalized χ^2 distances.

Color Histograms: We build joint histograms of color in CIE L*a*b* color space for each image. Our histograms have 4, 14, and 14 bins in L, a, and b respectively for a total of 784 dimensions. We compute distances between these histograms using χ^2 distance on the normalized histograms.

Geometric Probability Map: We compute the geometric class probabilities for image regions using the method of Hoiem et al. [13]. We use only the ground, vertical, porous, and sky classes because they are more reliably classified. We reduce the probability maps for each class to 8×8 and use an RBF kernel. This feature was used in [11].

Geometry Specific Histograms: Inspired by "Illumination Context" [16], we build color and texton histograms for each geometric class (ground, vertical, porous, and sky). Specifically, for each color and texture sample, we weight its contribution to each histogram by the probability that it belongs to that geometric class. These eight histograms are compared with χ^2 distance after normalization.

"Run the image through 20 layers of 3x3 convolutions and train the filters with SGD."*

* to the first order

[Karpathy, BayArea DL School, 16]

New frameworks for DL.

TensorFlow

```python
import tensorflow as tf
import numpy as np

N, D, H, C = 64, 1000, 100, 10

x = tf.placeholder(tf.float32, shape=[None, D])
y = tf.placeholder(tf.float32, shape=[None, C])

w1 = tf.Variable(1e-3 * np.random.randn(D, H).astype(np.float32))
w2 = tf.Variable(1e-3 * np.random.randn(H, C).astype(np.float32))

a = tf.matmul(x, w1)
a_relu = tf.nn.relu(a)
scores = tf.matmul(a_relu, w2)
probs = tf.nn.softmax(scores)
loss = -tf.reduce_sum(y * tf.log(probs))

learning_rate = 1e-2
train_step = tf.train.GradientDescentOptimizer(learning_rate).minimize(loss)

xx = np.random.randn(N, D).astype(np.float32)
yy = np.zeros((N, C)).astype(np.float32)
yy[np.arange(N), np.random.randint(C, size=N)] = 1

with tf.Session() as sess:
  sess.run(tf.initialize_all_variables())

  for t in xrange(100):
    _, loss_value = sess.run([train_step, loss],
                             feed_dict={x: xx, y: yy})
    print loss_value
```

Torch

```lua
require 'torch'
require 'nn'
require 'optim'

-- Batch size, input dim, hidden dim, num classes
local N, D, H, C = 100, 1000, 100, 10

-- Build a one-layer ReLU network
local net = nn.Sequential()
net:add(nn.Linear(D, H))
net:add(nn.ReLU())
net:add(nn.Linear(H, C))

-- Collect all weights and gradients in a single Tensor
local weights, grad_weights = net:getParameters()

-- Loss functions are called "criterions"
local crit = nn.CrossEntropyCriterion()  -- Softmax loss

-- Calback to interface with optim methods
local function f(w)
  assert(w == weights)

  -- Generate some random input data
  local x = torch.randn(N, D)
  local y = torch.Tensor(N):random(C)

  -- Forward pass: Compute scores and loss
  local scores = net:forward(x)
  local loss = crit:forward(scores, y)

  -- Backward pass: compute gradients
  grad_weights:zero()
  local dscores = crit:backward(scores, y)
  local dx = net:backward(x, dscores)

  return loss, grad_weights
end

-- Make a step using Adam
local state = {learningRate=1e-3}
optim.adam(f, weights, state)
```

Theano

```python
import theano
import theano.tensor as T

# Batch size, input dim, hidden dim, num classes
N, D, H, C = 64, 1000, 100, 10

x = T.matrix('x')
y = T.vector('y', dtype='int64')
w1 = T.matrix('w1')
w2 = T.matrix('w2')

# Forward pass: Compute scores
a = x.dot(w1)
a_relu = T.nnet.relu(a)
scores = a_relu.dot(w2)

# Forward pass: compute softmax loss
probs = T.nnet.softmax(scores)
loss = T.nnet.categorical_crossentropy(probs, y).mean()

# Backward pass: compute gradients
dw1, dw2 = T.grad(loss, [w1, w2])

f = theano.function(
      inputs=[x, y, w1, w2],
      outputs=[loss, scores, dw1, dw2],
    )
```

Caffe-on-Spark

No need to write code!

1. Convert data (run a script)
2. Define net (edit prototxt)
3. Define solver (edit prototxt)
4. Train (with pretrained weights)

Keras

```python
from keras.models import Sequential
from keras.layers.core import Dense, Activation
from keras.optimizers import SGD
from keras.utils import np_utils

D, H, C = 1000, 100, 10

model = Sequential()
model.add(Dense(input_dim=D, output_dim=H))
model.add(Activation('relu'))
model.add(Dense(input_dim=H, output_dim=C))
model.add(Activation('softmax'))

sgd = SGD(lr=1e-3, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd)

N, N_batch = 1000, 32
X = np.random.randn(N, D)
y = np.random.randint(C, size=N)
y = np_utils.to_categorical(y)

model.fit(X, y, nb_epoch=5, batch_size=N_batch, verbose=2)
```

Machine Learning Background

Scientific Method

example → mental model (hypothesis) → evaluate

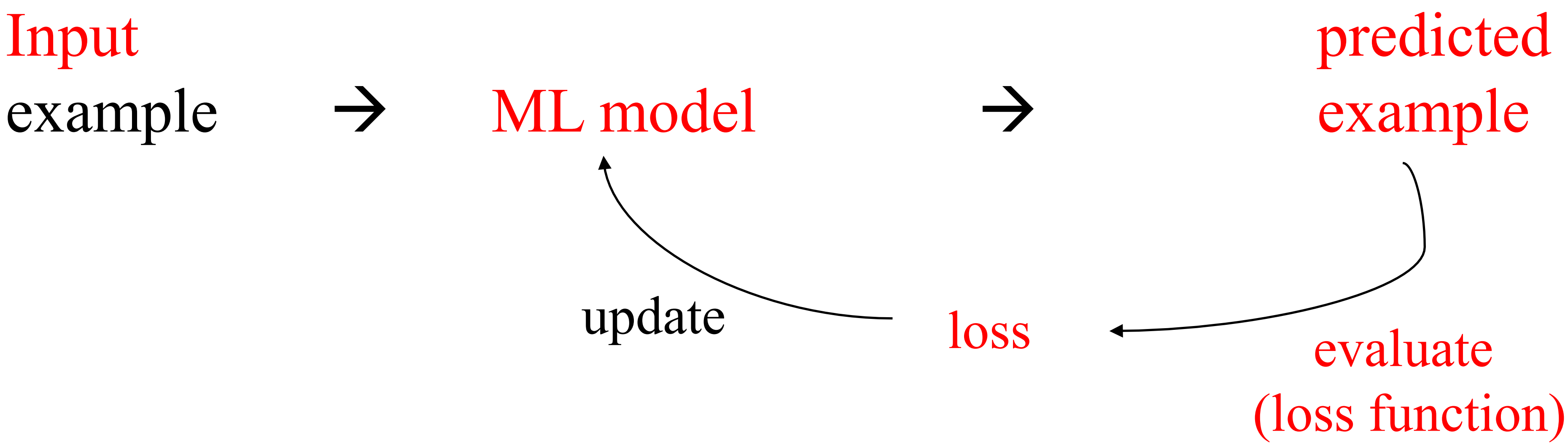
update

Machine Learning

Input example → ML model (hypothesis) → predicted example

evaluate + update

Machine Learning



Machine Learning

“A field of study that gives computers the ability to learn without being explicitly programmed”

- Arthur Samuel

- Machines take as input some data and attempt to identify patterns in the data

- Machines take as input some data and attempt to imitate patterns in the data, either directly or indirectly

Machine Learning Definition

- A computer program is said to **learn** from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*.

  [Mitchell, T., Machine Learning: An algorithmic perspective]

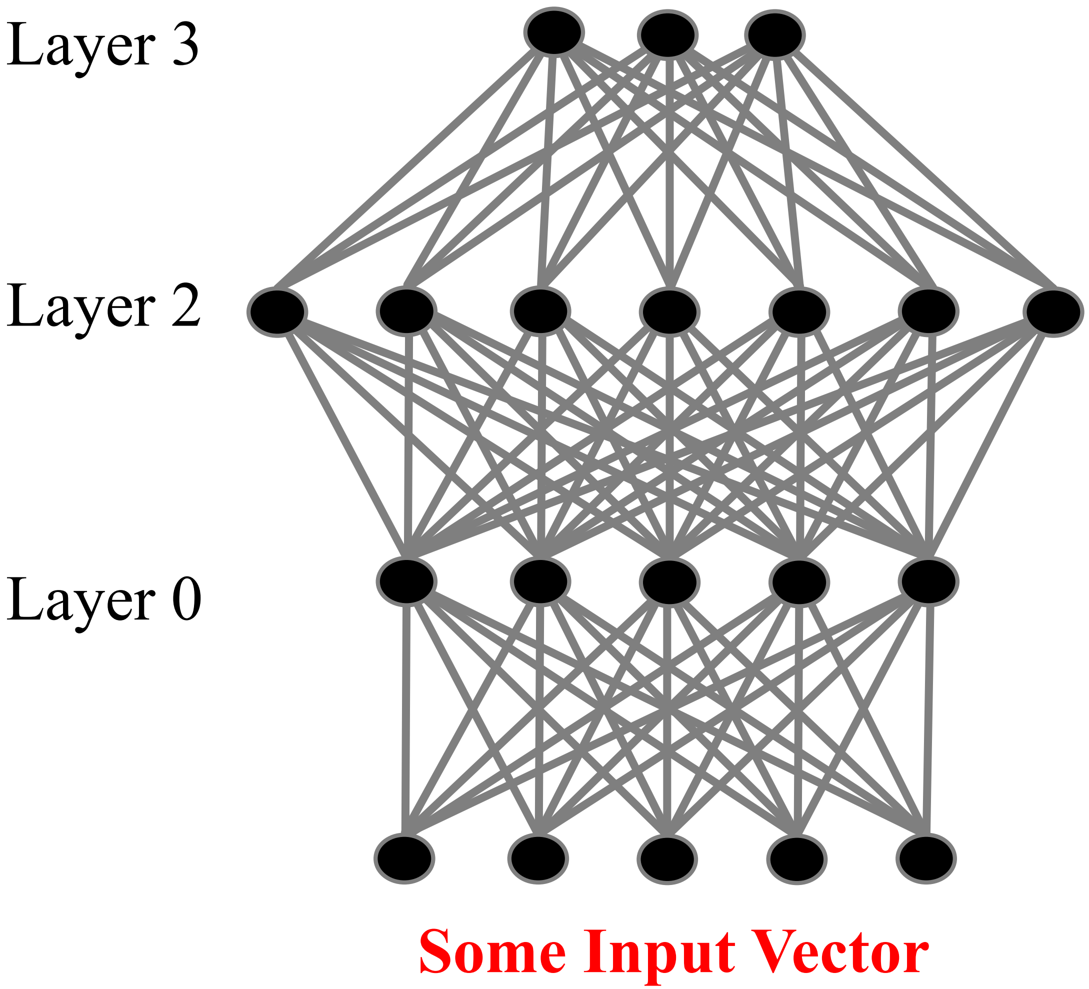
Study of Machine Learning

- Study of algorithms and systems that
  - improve their performance P
  - at some task T
  - with experience E

- We need a well-defined learning task: <P,T,E>

Quick Hello to Deep Learning

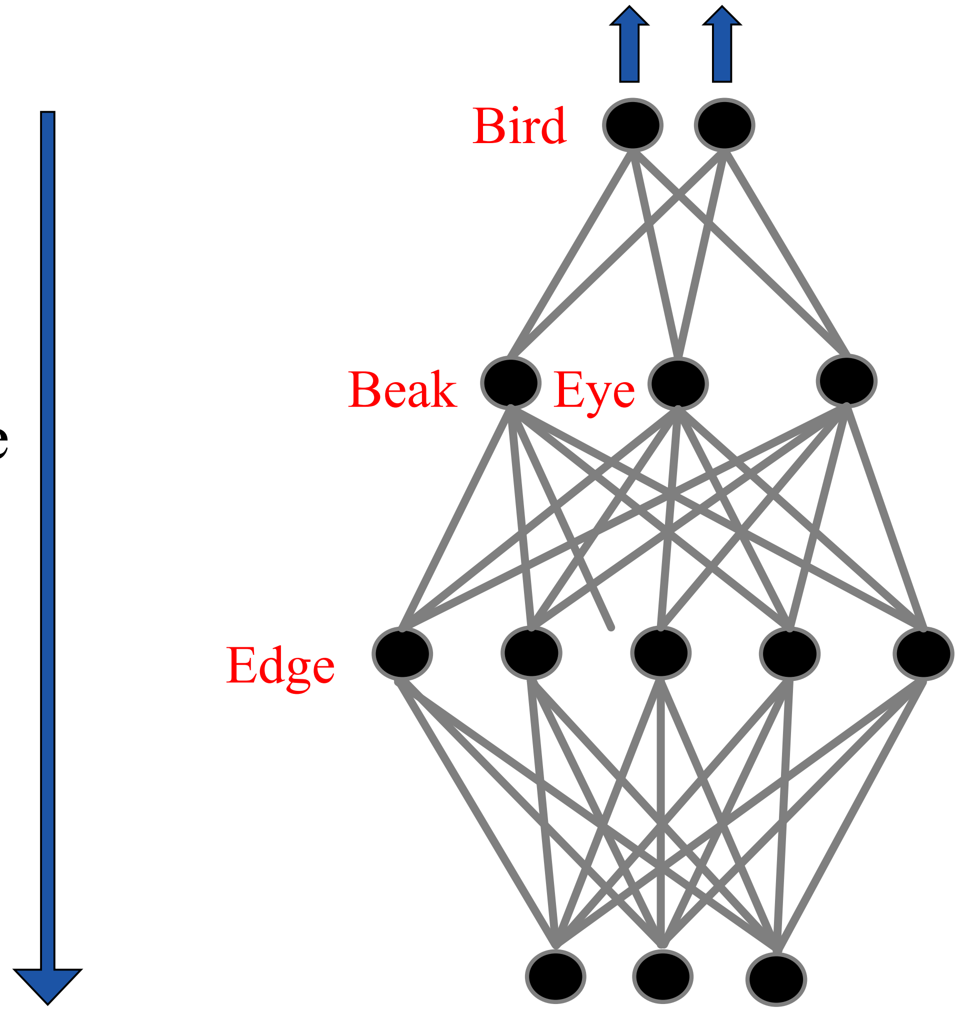
Deep Neural Nets have 3 Layers or more



Supervised ML with Back Propagation

Compare outputs with correct answer to get error signal

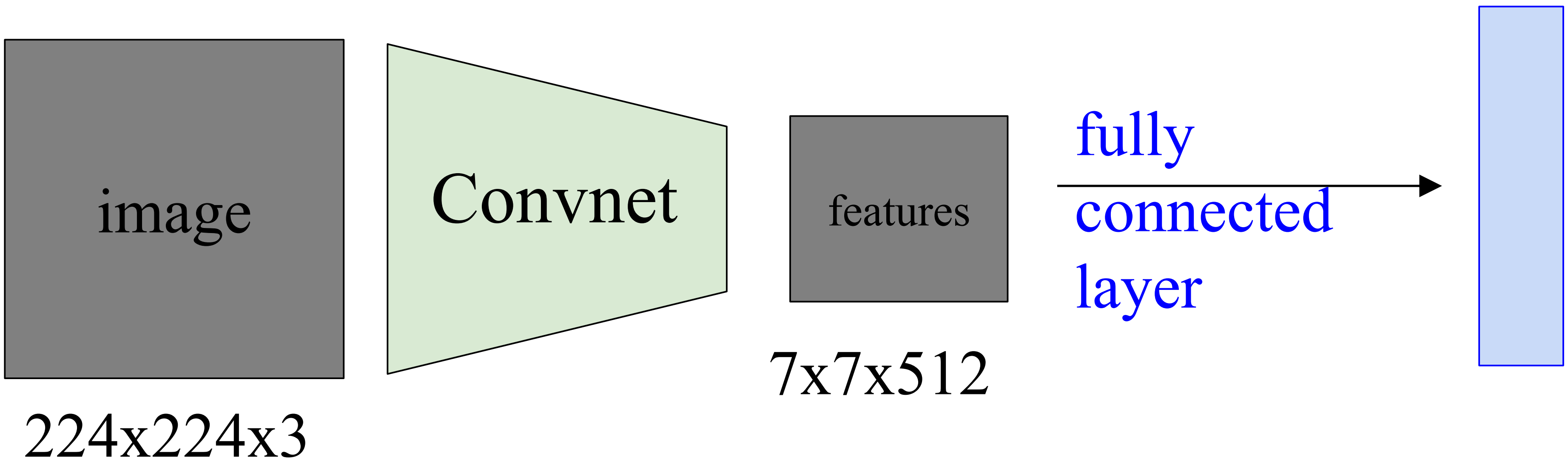
Back-propegate the gradient vector to change the weights.



Classes of Deep Learning Networks

Pattern Recognition Convnets (CNNs)	Sequence Models RNN/LSTM
General DNNs Feed-forward	Unsupervised DNNs Deep RL

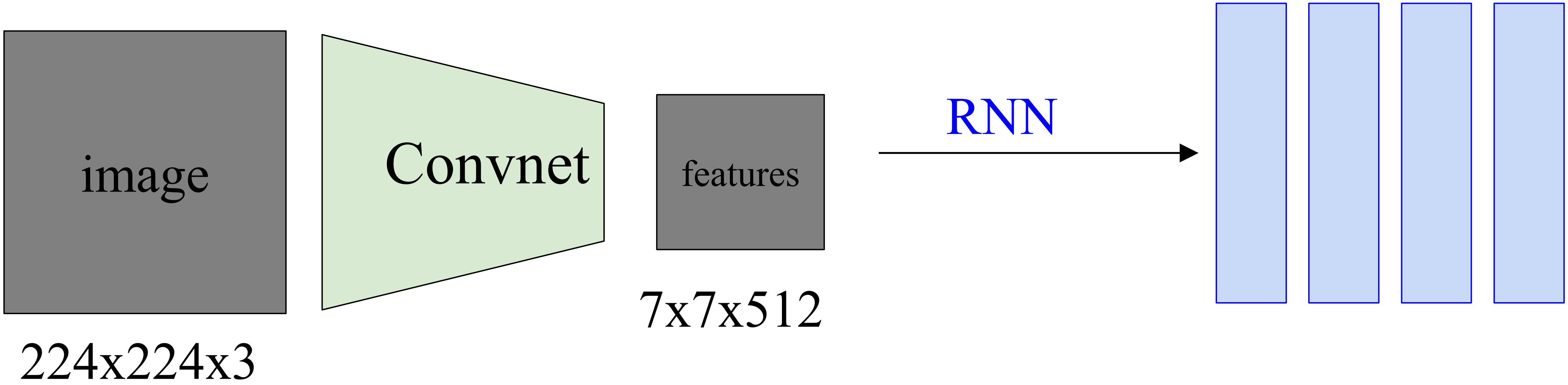
Image Classification



e.g. vector of 1000 numbers giving probabilities for different classes.

[Karpathy, BayArea DL School, 16]

Image Captioning

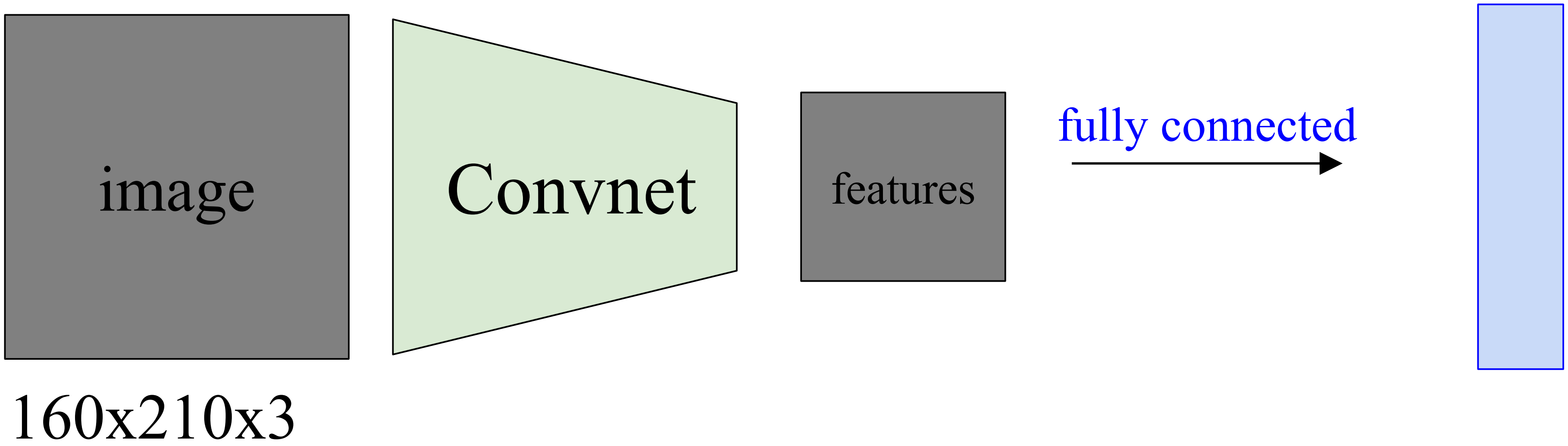


A sequence of 10,000-dimensional vectors giving probabilities of different words in the caption.

[Karpathy, BayArea DL School, 16]

Reinforcement Learning

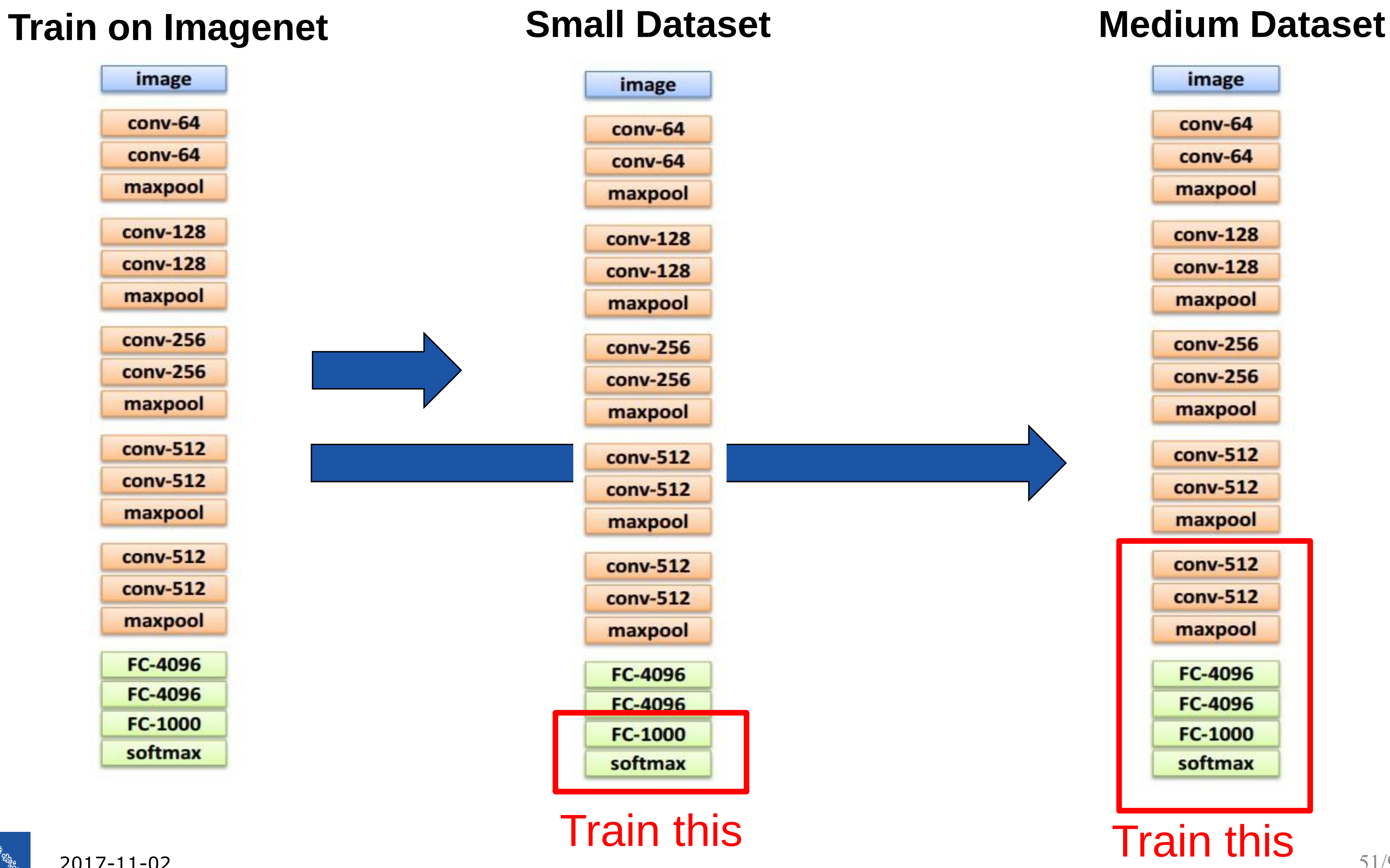
Mnih et al. 2015



e.g., vector of 8 numbers giving probability of wanting to take any of the 8 possible ATARI actions.

[Karpathy, BayArea DL School, 16]

Transfer Learning



[Karpathy, BayArea DL School, 16]

Understanding Deep Learning Systems

DL vs Human-Level Performance

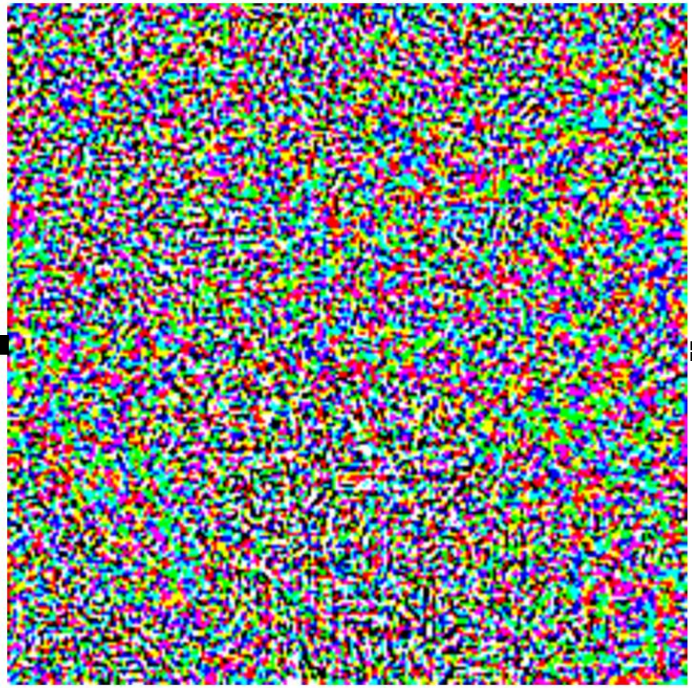
- In the old days we had to prove convergence of our ML algorithms
  - Limited to convex optimization problems

- Human-level accuracy is useful for evaluating the performance of deep-learning systems.

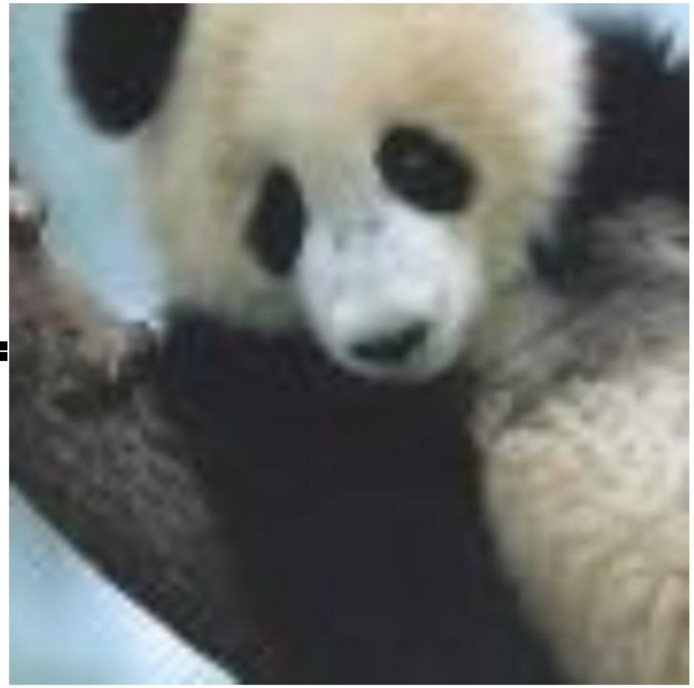
- How do we define human-level performance?
  - typical human
  - expert human
  - team of experts

Adverserial Deep Learning

Original image classified as a panda with 60% confidence.



Tiny adversarial perturbation.

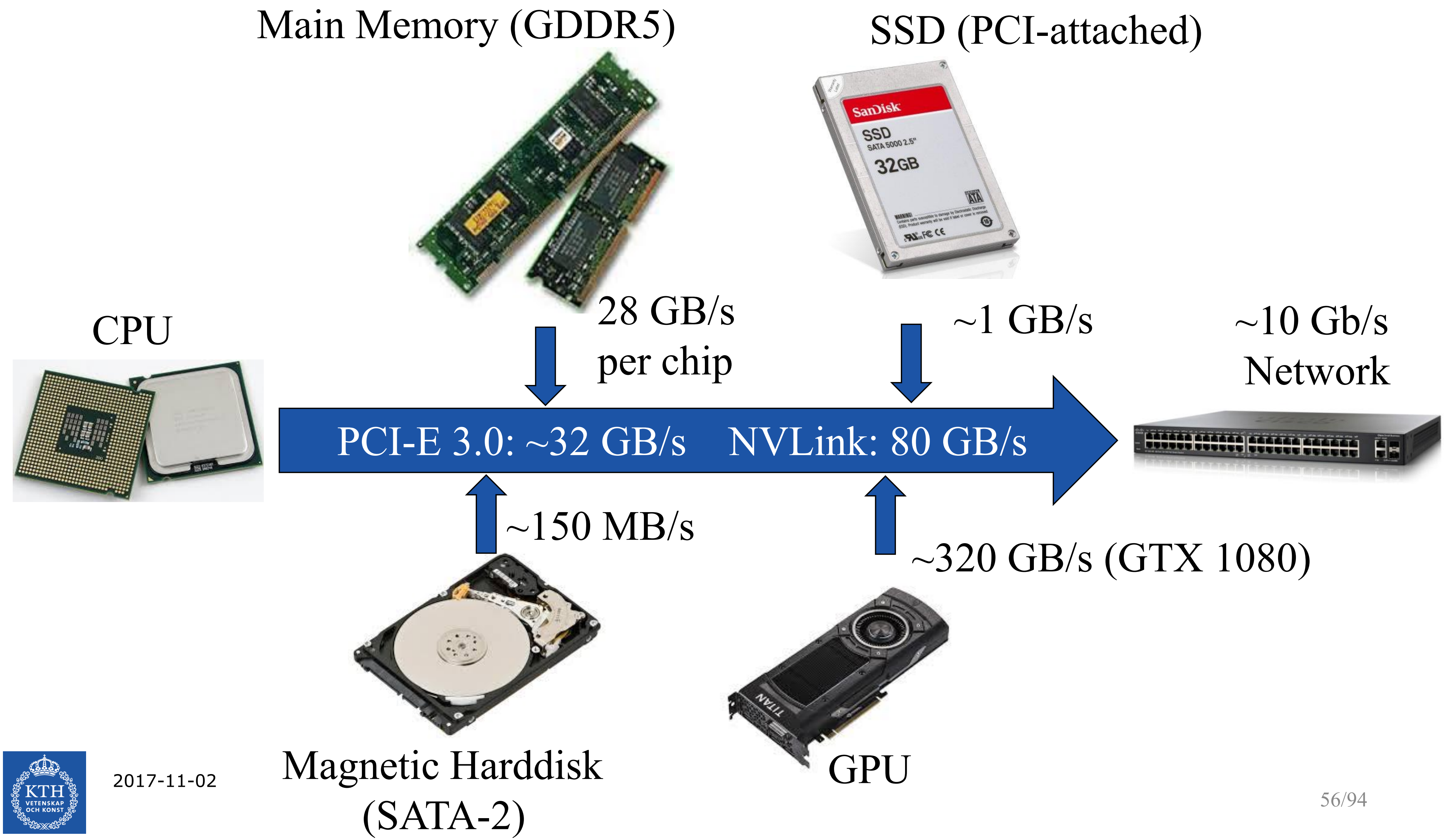


Imperceptibly modified image, classified as a gibbon with 99% confidence.

[http://www.kdnuggets.com/2015/07/deep-learning-adversarial-examples-misconceptions.html]

Hardware Numbers that you should know

Key (Network/Bus) Bandwidths



Linear Algebra Review

Matrices

A Matrix is a 2-d array

$$A = \left(a_{ij}\right) = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

- n = # of columns
- m = # of rows
- dimensions = $m \times$ n

- Notation:
  - Matrices are denoted by (bold) uppercase letters
  - A_{ij} denotes the entry in i^{th} row and j^{th} column
  - If **A** is $m \times n$, it has m rows and n columns
  - If **A** is $m \times n$, then $\boldsymbol{A} \in \mathbb{R}^{\boldsymbol{m \times n}}$

Vectors

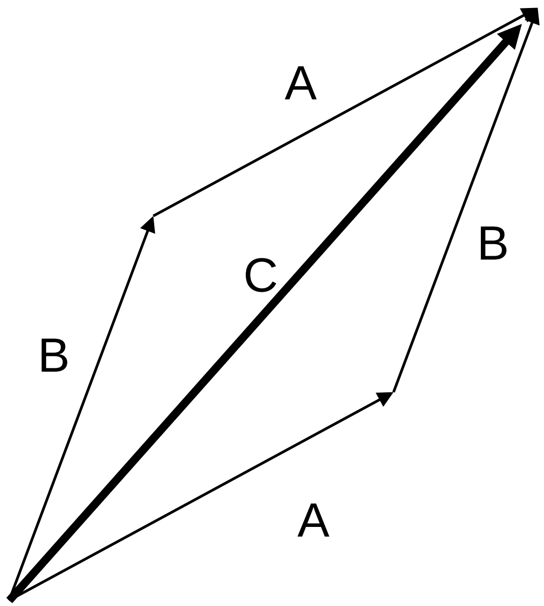
- A vector, **v**, of dimension n is an n × 1 matrix rectangular array of elements

$$\boldsymbol{v} = \begin{bmatrix} 1.1 \\ 0.5 \\ 9.4 \end{bmatrix}$$

- Notation:
  - Vectors are denoted by (bold) lowercase letters
  - $\mathbf{v}_i$ denotes the i^{th} entry
  - If **v** is *n* dimensional, then $\boldsymbol{v} \in \mathbb{R}^n$

Vector Addition

$$\mathbf{c{=}a{+}b{=}} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \end{bmatrix}$$



Matrix Addition/Subtraction

- Addition

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a+e & b+f \\ c+g & d+h \end{bmatrix}$$

Add the elements

- Subtraction

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a-e & b-f \\ c-g & d-h \end{bmatrix}$$

Subtract the elements

- For addition and subtraction, the matrices must have the same dimensions

Matrix Transpose

A_{12}

$$\begin{bmatrix} 1 & 4 \\ 6 & 1 \\ 3 & 5 \end{bmatrix} \Longrightarrow \begin{bmatrix} 1 & 6 & 3 \\ 4 & 1 & 5 \end{bmatrix} \qquad \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix} \Longrightarrow \begin{bmatrix} 3 & 4 & 1 \end{bmatrix}$$

3×2 $\quad$ 2×3 $\quad$ 3×1 $\quad$ 1×3

$(A^{\top})_{21}$

- Swap the rows and columns of a matrix
- Properties of matrix transposes:
  - $A_{ij} = (A^{\top})_{ji}$
  - If **A** is $m \times n$, then $\mathbf{A}^{\top}$ is $n \times m$

$$(A+B)^T = A^T + B^T$$

$$(AB)^T = B^T A^T$$

Inverse of a Matrix

- If A is a square matrix, the **inverse** of A, called A^{-1}, satisfies

$$AA^{-1} = \mathrm{I} \quad \text{and} \quad A^{-1}A = \mathrm{I},$$

- Where I, the **identity matrix**, is a diagonal matrix with all 1's on the diagonal.

$$I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrix Scalar Multiplication

- Multiply each matrix element by the scalar value

$$4 \times \begin{bmatrix} 1 & 6 & 3 \\ 4 & 9 & 6 \end{bmatrix} = \begin{bmatrix} 4 & 24 & 12 \\ 16 & 36 & 24 \end{bmatrix}$$

$$-0.5 \times \begin{bmatrix} 3 \\ 8 \\ 4 \end{bmatrix} = \begin{bmatrix} -1.5 \\ -4 \\ -2 \end{bmatrix}$$

Inner Product

- A function that maps two vectors to a scalar
  - Called the *dot* product or *inner* product

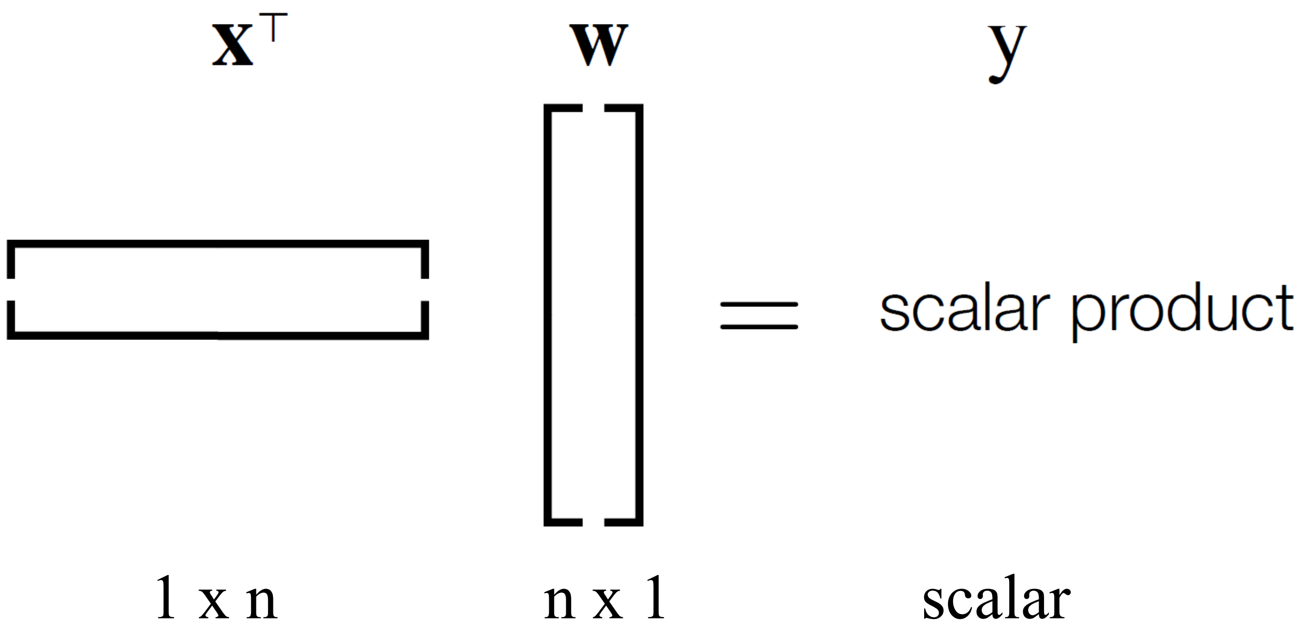
$$\begin{bmatrix} 3 \\ 8 \\ 4 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = 31$$

$$3 \times 1 + 8 \times 2 + 4 \times 3 = 31$$

- Multiplies the vector elements pairwise
- Both vectors must be the same dimension

Scalar Product

- Vectors assumed to be in column form
- Transposed vectors are row vectors



- Common notation for the scalar product: $\mathbf{x}^\top\mathbf{w}$

Matrix-Scalar Multiplication

- Involves repeated scalar products

$$\begin{bmatrix} 3 & 4 & 6 \\ 2 & 3 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 29 \\ 11 \end{bmatrix}$$

$$3 \times 1 + 4 \times 2 + 6 \times 3 = 29$$
$$2 \times 1 + 3 \times 2 + 1 \times 3 = 11$$

Matrix-Matrix Multiplication

- Involves repeated scalar products

$$\begin{bmatrix} 3 & 4 & 6 \\ 2 & 3 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 3 \\ 1 & -1 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 34 & 35 \\ 11 & 8 \end{bmatrix}$$

$$3 \times 2 + 4 \times 1 + 6 \times 4 = 34$$

Matrix-Matrix Multiplication

- Involves repeated scalar products

$$\begin{bmatrix} 3 & 4 & 6 \\ 2 & 3 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 3 \\ 1 & -1 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 34 & 35 \\ 11 & 8 \end{bmatrix}$$

$$3 \times 3 + 4 \times -1 + 6 \times 5 = 35$$

Matrix-Matrix Multiplication

- Involves repeated scalar products

$$\begin{bmatrix} 3 & 4 & 6 \\ 2 & 3 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 3 \\ 1 & -1 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 34 & 35 \\ 11 & 8 \end{bmatrix}$$

$$2 \times 2 + 3 \times 1 + 1 \times 4 = 11$$

Matrix-Matrix Multiplication

- Involves repeated scalar products

$$\begin{bmatrix} 3 & 4 & 6 \\ 2 & 3 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 3 \\ 1 & -1 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 34 & 35 \\ 11 & 8 \end{bmatrix}$$

$$2 \times 3 + 3 \times -1 + 1 \times 5 = 8$$

Matrix Product

Let $\boldsymbol{A} = (a_{ij})$ denote an $m \times n$ matrix and $\boldsymbol{B} = (b_{jl})$ denote an $n \times k$ matrix

Then the $m \times k$ matrix $\boldsymbol{C} = (c_{il})$ where

$$c_{il} = \sum_{j=1}^{m} a_{ij} b_{jl}$$

is called the **product** of $\boldsymbol{A}$ and $\boldsymbol{B}$ and is denoted by $\boldsymbol{A} \cdot \boldsymbol{B}$

KTH VETENSKAP OCH KONST

Matrix Multiplication Properties

- Associative

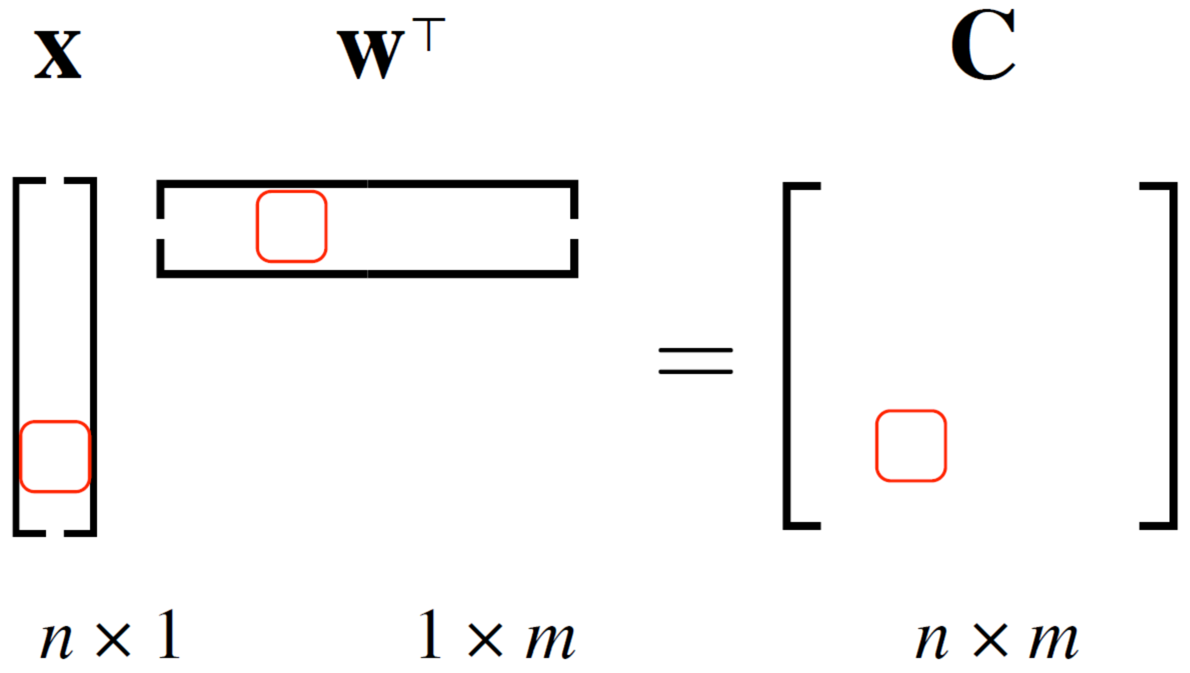
  $$(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$$

- Not commutative

  $$\mathbf{AB} \neq \mathbf{BA}$$

Outer Product

- Matrix-Matrix Multiplication involving two vectors
- C_{ij} is the *inner product* of i^{th} entry of **x** and j^{th} entry of **w**



L^p Norm for Vectors

- Norms are functions that measure how large a vector is
  - A scalar has a magnitude/length: its absolute value
- L1 Norm for $\boldsymbol{x} \in \mathbb{R}^n$

$$\|\boldsymbol{x}\|_1 = \sum_i |x_i|$$

- L2 Norm (Euclidean norm)

$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_i |x_i|^2} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

- L^p Norm

$$\|\boldsymbol{x}\|_p = \left(\sum_i |x_i|^p\right)^{1/p}$$

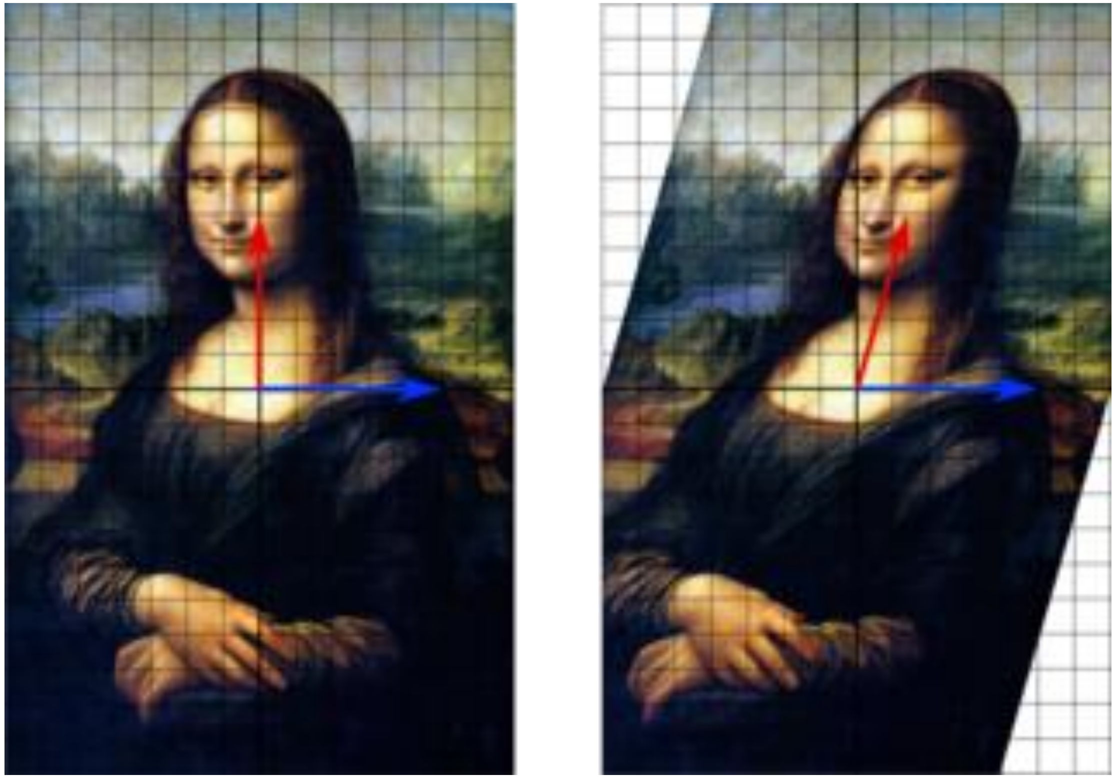
Special Matrices and Vectors

- A Unit vector has a magnitude of 1:
  - $\hat{\boldsymbol{x}} = \|\boldsymbol{x}\|_2 = 1$
- Symmetric Matrix:
  - $\mathbf{A}^{\mathsf{T}} = \mathbf{A}$
- Orthogonal Matrix:
  - $\mathbf{A}^{\mathsf{T}}\mathbf{A} = \mathbf{A}\mathbf{A}^{\mathsf{T}} = \mathbf{I}$

EigenVectors and Eigenvalues

- Eigenvector and eigenvalue:

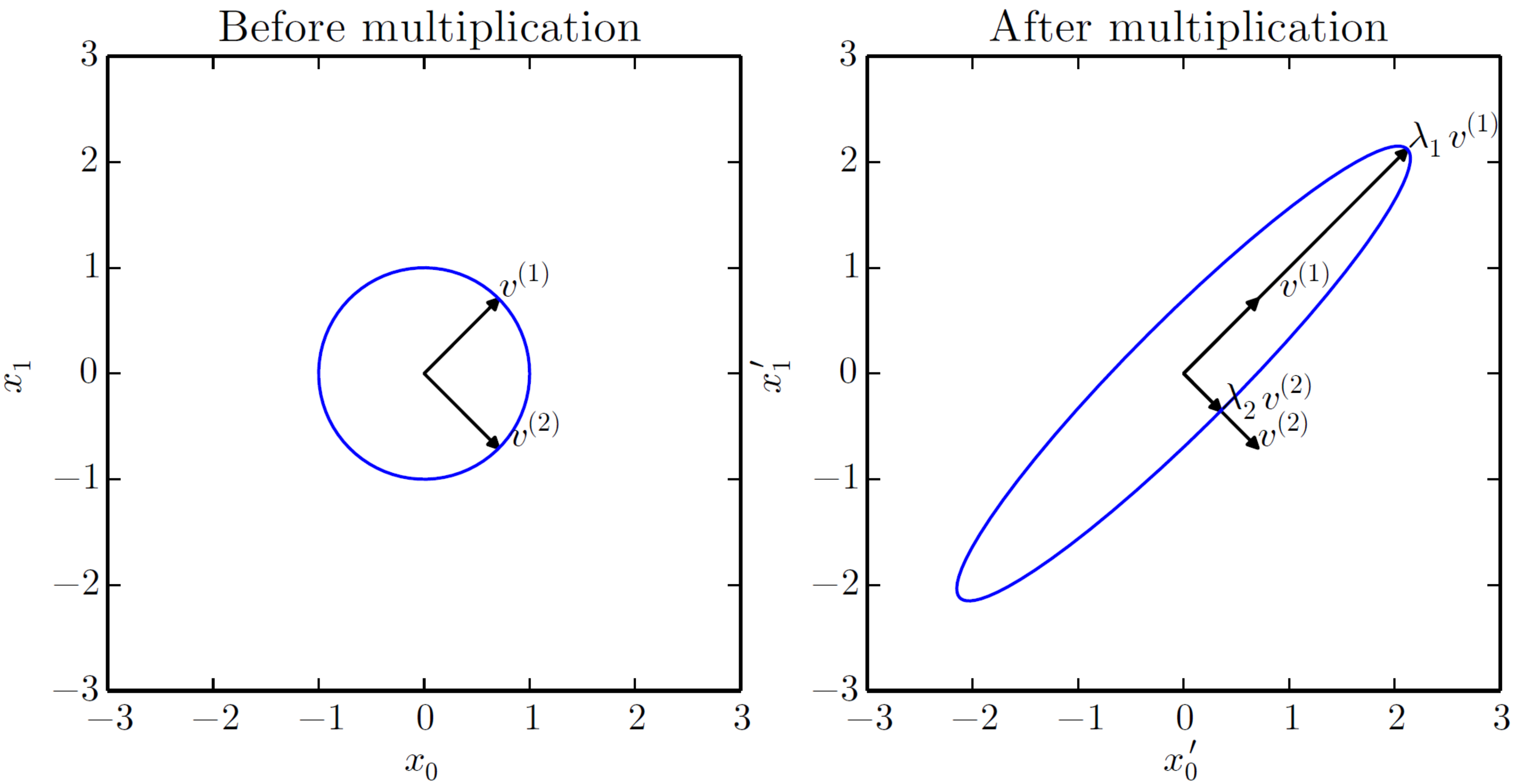
$$\boldsymbol{A}\boldsymbol{v} = \lambda\boldsymbol{v}$$



Any vector that points directly to the right or left with no vertical component is an eigenvector of this transformation (shear mapping) because the mapping does not change its direction.

[from Wikipedia]

Effect of Eigenvalues



[from Deep Learning BooK, Goodfellow et aL]

Tensors

- A tensor is an array of numbers, that may have
  - zero dimensions, and be a scalar
  - one dimension, and be a vector
  - two dimensions, and be a matrix
  - or more dimensions.

[from Deep Learning BooK, Goodfellow et aL]

Learning linear algebra

- Do a lot of practice problems
- Start out with lots of summation signs and indexing into individual entries
- Eventually you will be able to mostly use matrix and vector product notation quickly and easily

Probability Theory

[Slides Adapted from Deep Learning BooK, Goodfellow et aL]

Random Variable

- If a variable can take on any value between two specified values, it is called a **continuous variable**; otherwise, it is called a **discrete variable**.

- A random variable has a probability distribution, which specifies the probability that its value falls in any given interval.

Probability Mass Function (Discrete)

The domain of P must be the set of all possible states of x.

$\forall x \in \mathrm{x}, 0 \leq P(x) \leq 1$. An impossible event has probability 0 and no state can be less probable than that. Likewise, an event that is guaranteed to happen has probability 1, and no state can have a greater chance of occurring.

$\sum_{x \in \mathrm{x}} P(x) = 1$. We refer to this property as being **normalized**. Without this property, we could obtain probabilities greater than one by computing the probability of one of many events occurring.

Probability Density Function (Continuous)

The domain of p must be the set of all possible states of x.

$\forall x \in \mathrm{x}, p(x) \geq 0$. Note that we do not require $p(x) \leq 1$.

$\int p(x)dx = 1$.

Computing Marginal Probability with the Sum Rule

i/j	1	2	3	4	5	6	px(i)
1	1/36	1/36	1/36	1/36	1/36	1/36	1/6
2	1/36	1/36	1/36	1/36	1/36	1/36	1/6
3	1/36	1/36	1/36	1/36	1/36	1/36	1/6
4	1/36	1/36	1/36	1/36	1/36	1/36	1/6
5	1/36	1/36	1/36	1/36	1/36	1/36	1/6
6	1/36	1/36	1/36	1/36	1/36	1/36	1/6
py(j)	1/6	1/6	1/6	1/6	1/6	1/6	

$$\forall x \in \mathrm{x}, P(\mathrm{x} = x) = \sum_y P(\mathrm{x} = x, \mathrm{y} = y).$$

$$p(x) = \int p(x, y) dy.$$

Conditional Probability

- If 60% of the class passed both labs and 80% of the class passed the first test. What percent of those who passed the first test also passed the second test?

$$P(\mathrm{y} = y \mid \mathrm{x} = x) = \frac{P(\mathrm{y} = y, \mathrm{x} = x)}{P(\mathrm{x} = x)}.$$

Chain Rule of Probability

- Natural Language Processing.
  "Play it again ____ "- Humphrey Bogart

- Probability of the next word (assuming a 4-Gram)?
  P(w4 | w1, w2, w3)

$$P(\mathrm{x}^{(1)}, \ldots, \mathrm{x}^{(n)}) = P(\mathrm{x}^{(1)}) \Pi_{i=2}^{n} P(\mathrm{x}^{(i)} \mid \mathrm{x}^{(1)}, \ldots, \mathrm{x}^{(i-1)}).$$

Independence

- The event of getting a 6 the first time a die is rolled and the event of getting a 6 the second time are *independent*.
- The event of getting a 6 the first time a die is rolled and the event that the sum of the numbers seen on the first and second trials is 8 are *not* independent.

$$\forall x \in \mathrm{x}, y \in \mathrm{y},\ p(\mathrm{x} = x, \mathrm{y} = y) = p(\mathrm{x} = x)p(\mathrm{y} = y).$$

Expectation

- Expected value of a dice 1-6 is 3.5 (weighted mean)

$$\mathbb{E}_{\mathrm{x}\sim P}[f(x)] = \sum_x P(x)f(x),$$

$$\mathbb{E}_{\mathrm{x}\sim p}[f(x)] = \int p(x)f(x)dx.$$

linearity of expectations:

$$\mathbb{E}_{\mathrm{x}}[\alpha f(x) + \beta g(x)] = \alpha\mathbb{E}_{\mathrm{x}}[f(x)] + \beta\mathbb{E}_{\mathrm{x}}[g(x)]$$

Variance and Covariance

- The covariance between two Random Variables X and Y measures the degree to which X and Y are linearly related.

$$\mathrm{Var}(f(x)) = \mathbb{E}\left[(f(x) - \mathbb{E}[f(x)])^2\right].$$

$$\mathrm{Cov}(f(x), g(y)) = \mathbb{E}\left[(f(x) - \mathbb{E}\left[f(x)\right])\left(g(y) - \mathbb{E}\left[g(y)\right]\right)\right].$$

Functions of Interest

Logistic Sigmoid

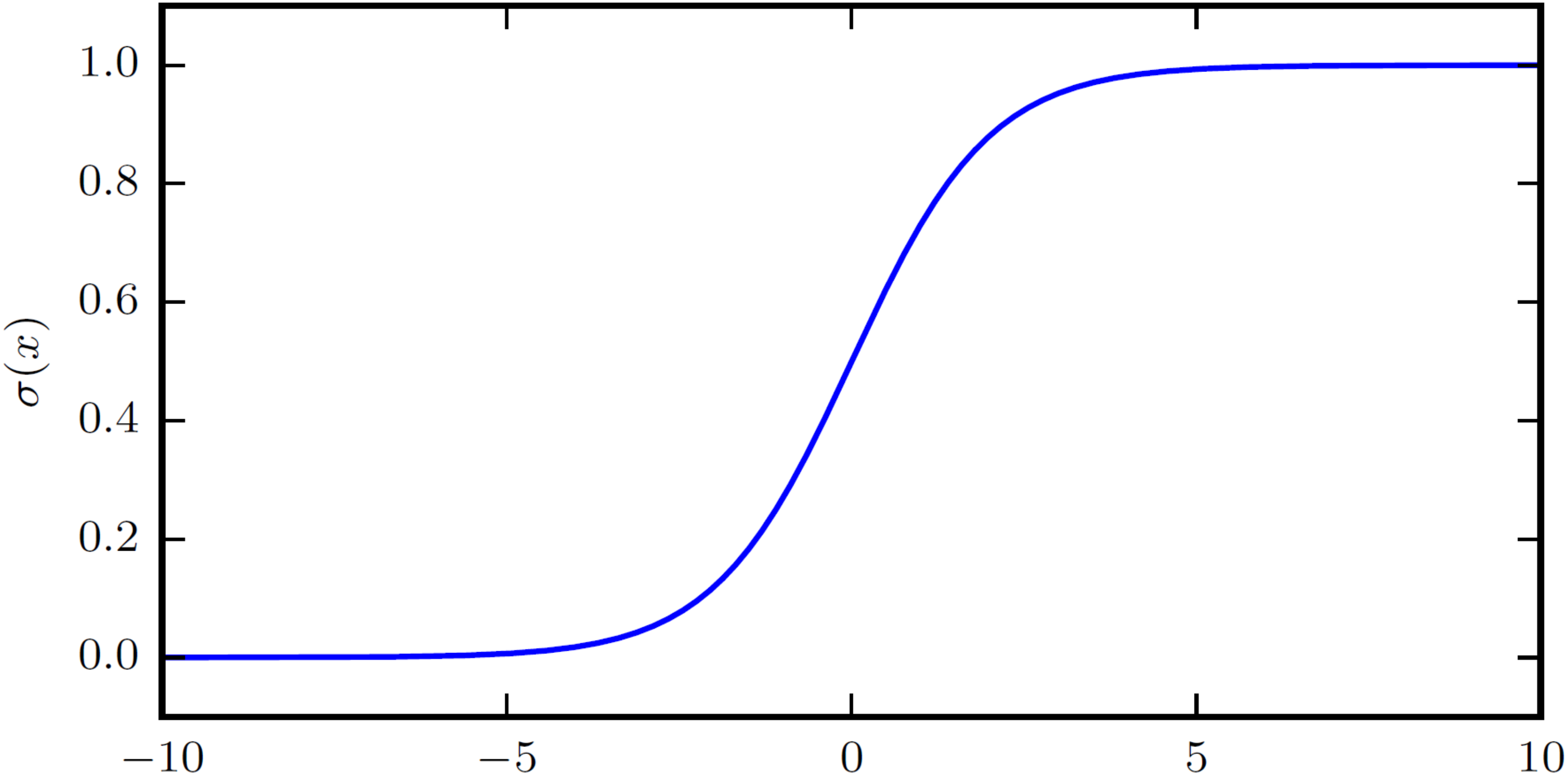


Figure 3.3: The logistic sigmoid function.

The SoftPlus Function

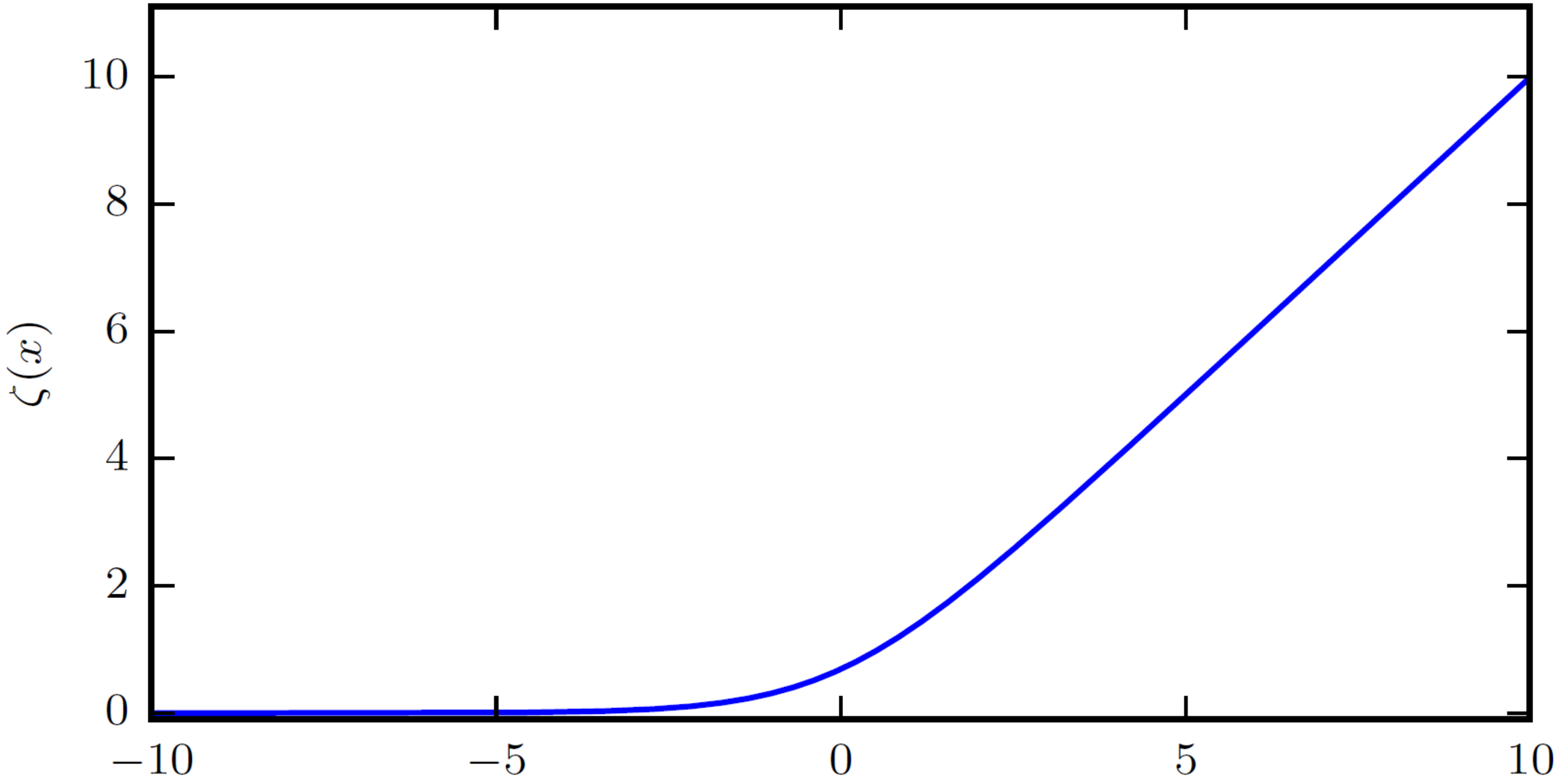


Figure 3.4: The softplus function.

Reading Instructions

- Chapter 1
- Chapter 2.1-2.7
- Chapter 3

- References
  - Linear Algebra, Gilbert Strang.
  - Probability Notes: http://web.mit.edu/13.42/www/handouts/reading-probability.pdf