Deep Learning

Lecture 1

Yannic Kilcher

Institute for Machine Learning
ETH Zurich – ml.inf.ethz.ch

September 24, 2018

Introduction

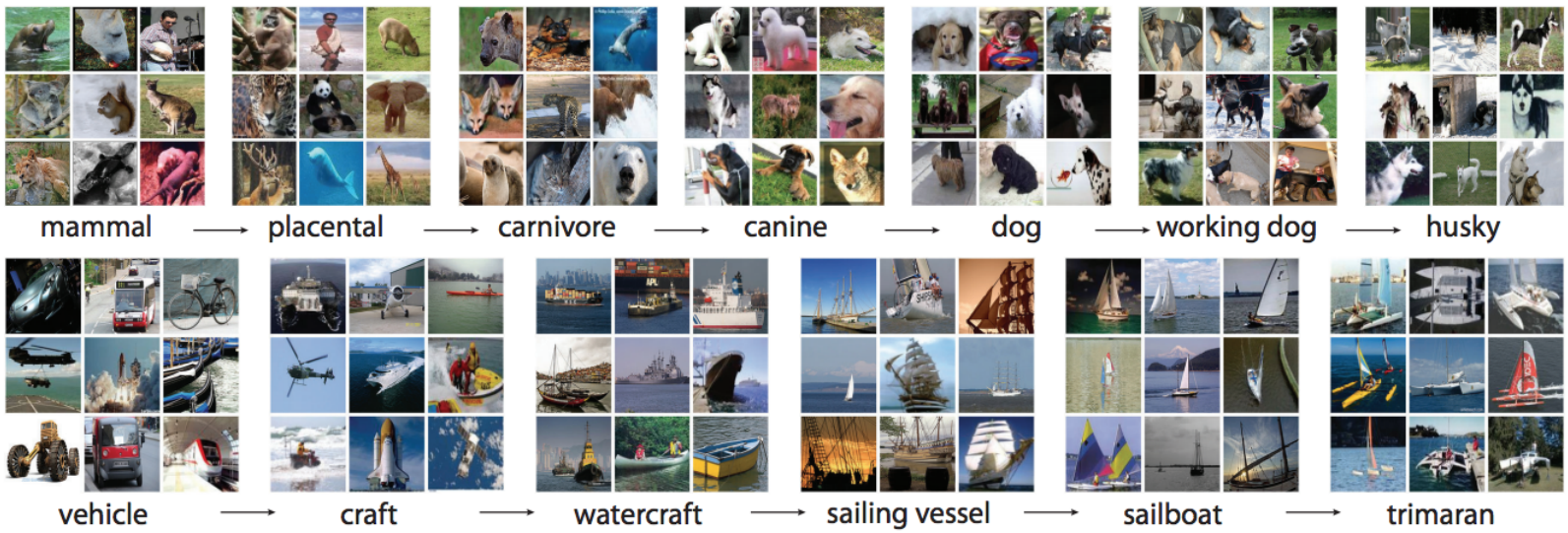
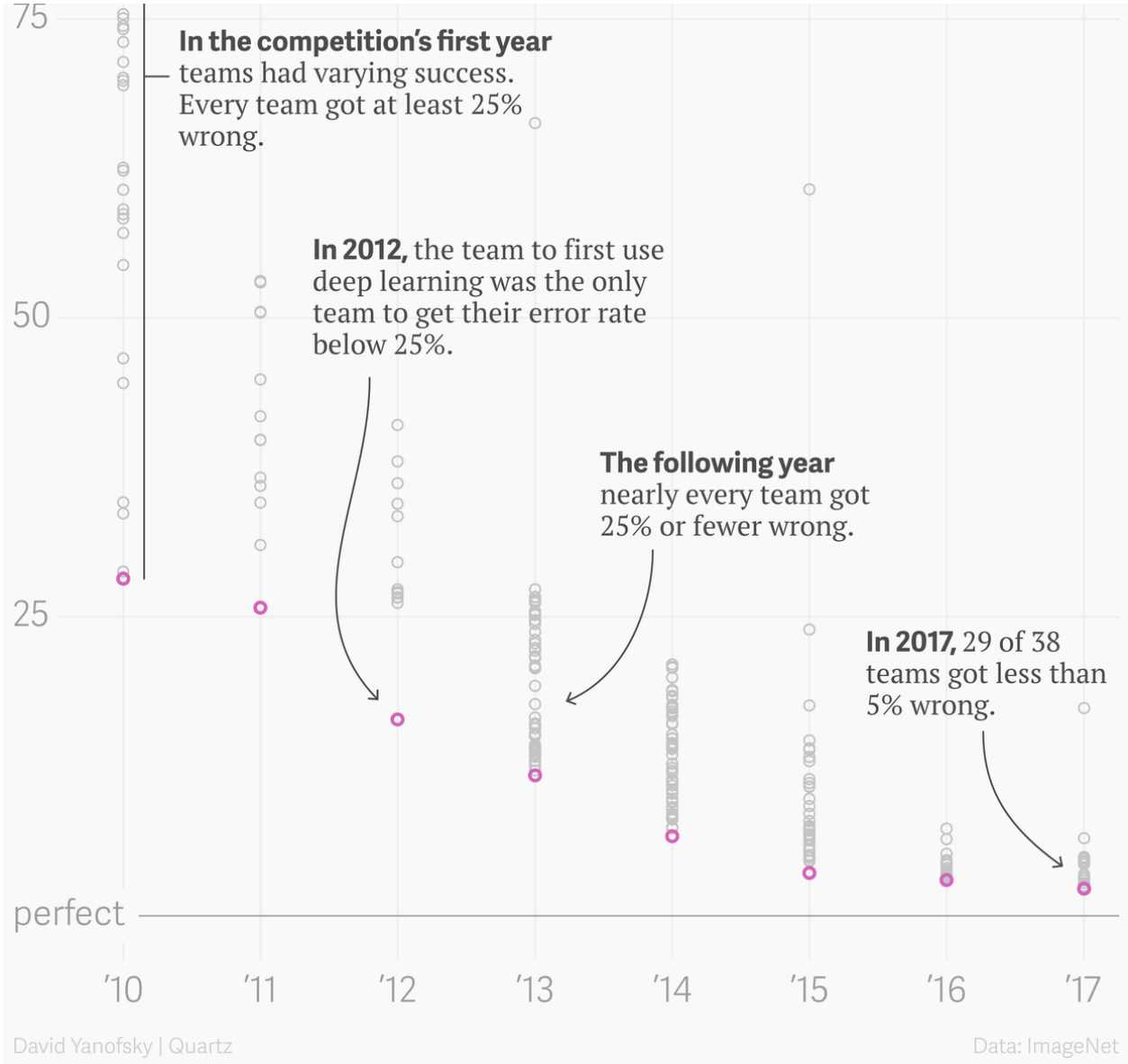


Figure 1: A snapshot of two root-to-leaf branches of ImageNet: the **top** row is from the mammal subtree; the **bottom** row is from the vehicle subtree. For each synset, 9 randomly sampled images are presented.

Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. Ieee, 2009.

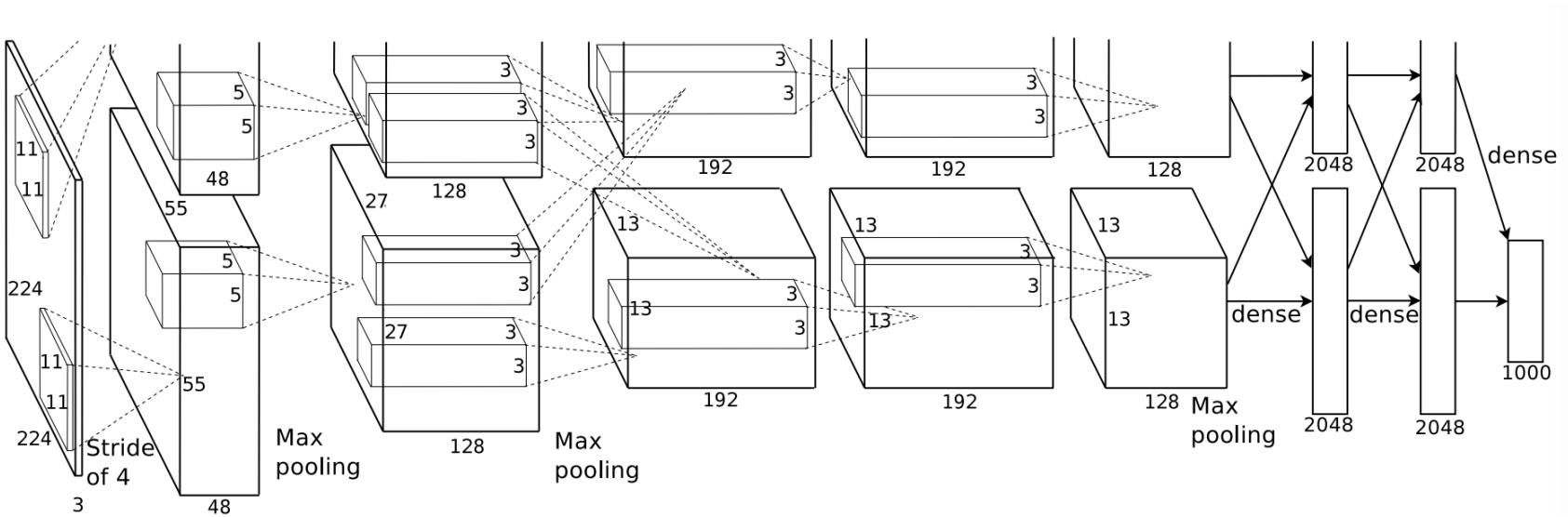
Introduction



Introduction

Introduction

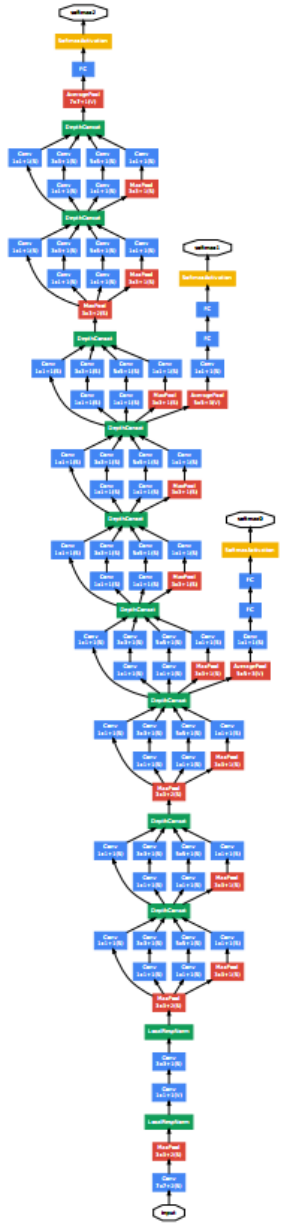
Introduction



Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

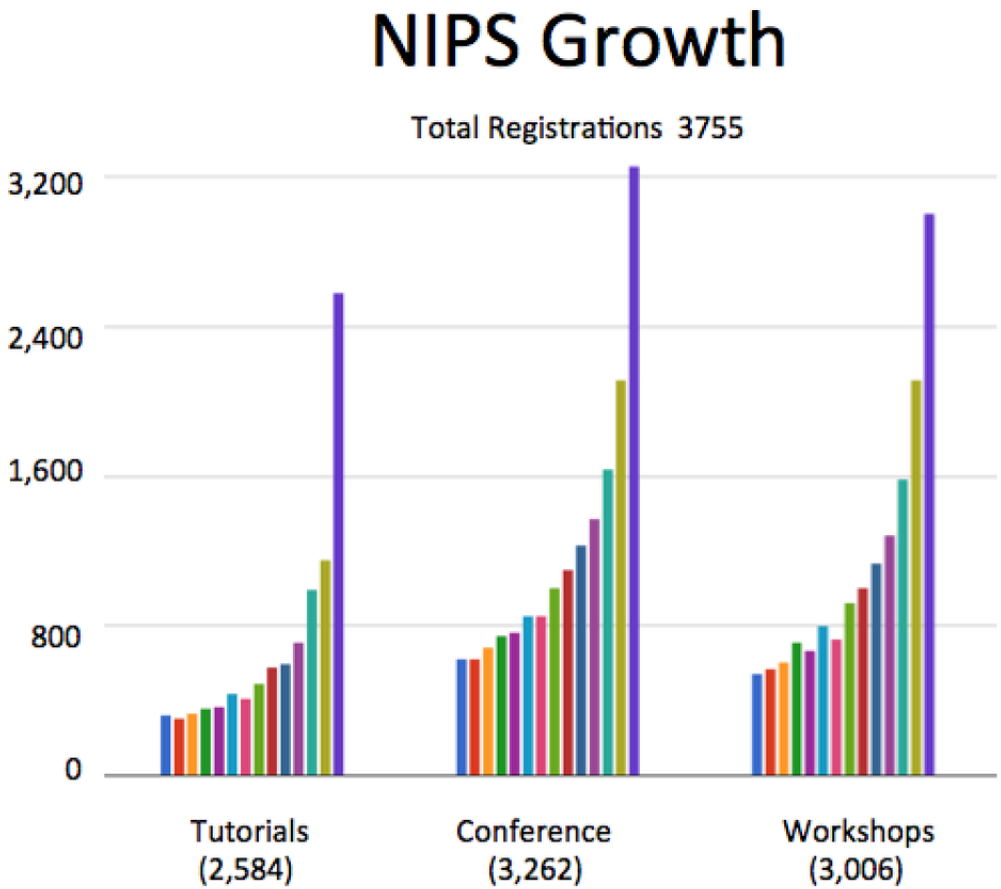
Introduction

Introduction

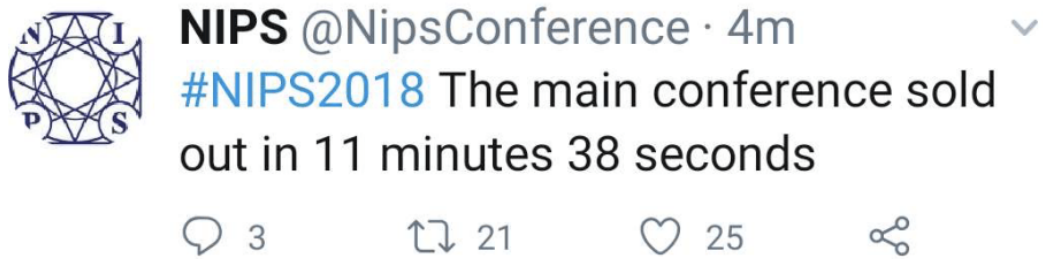


Introduction

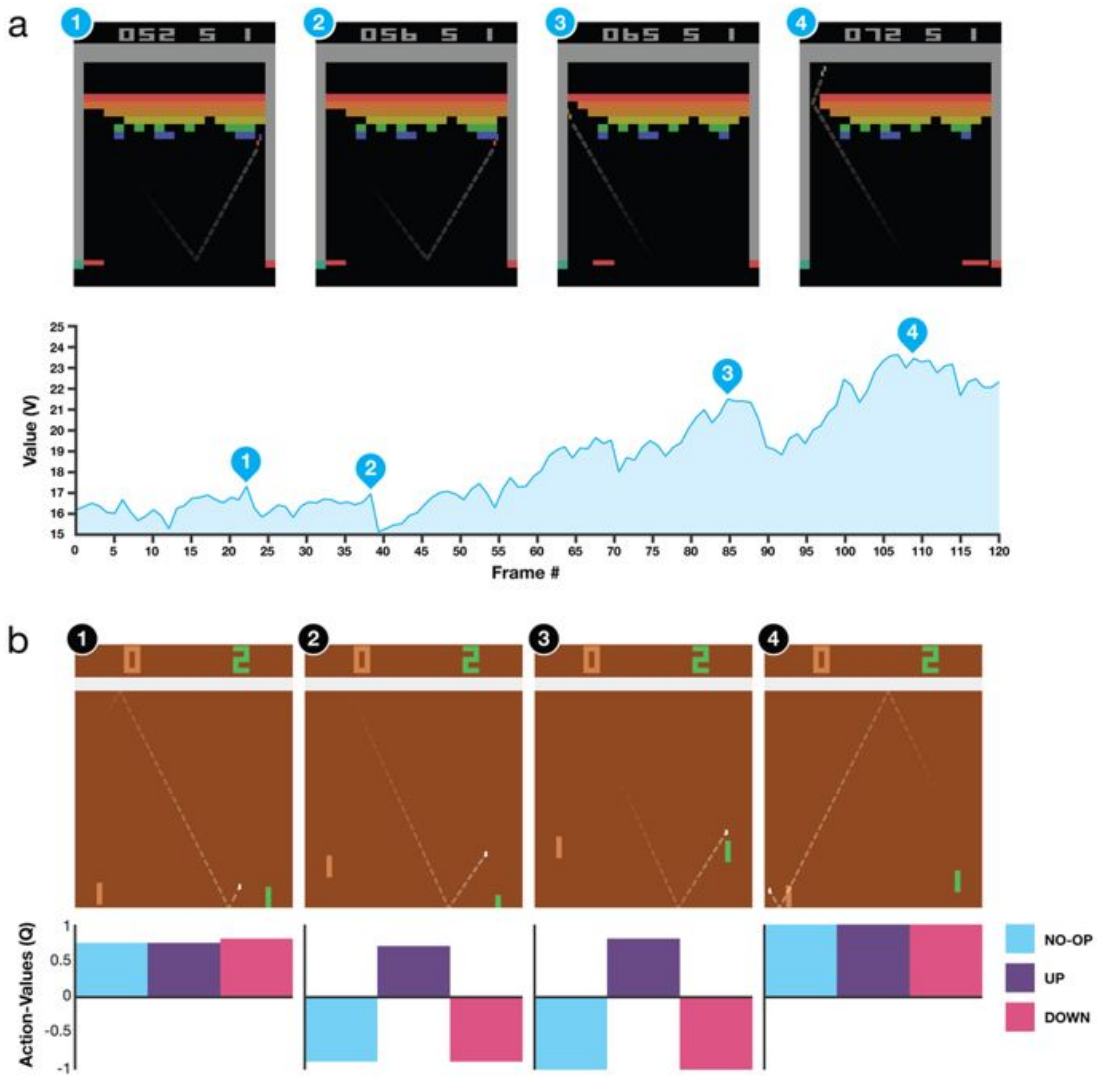
Introduction



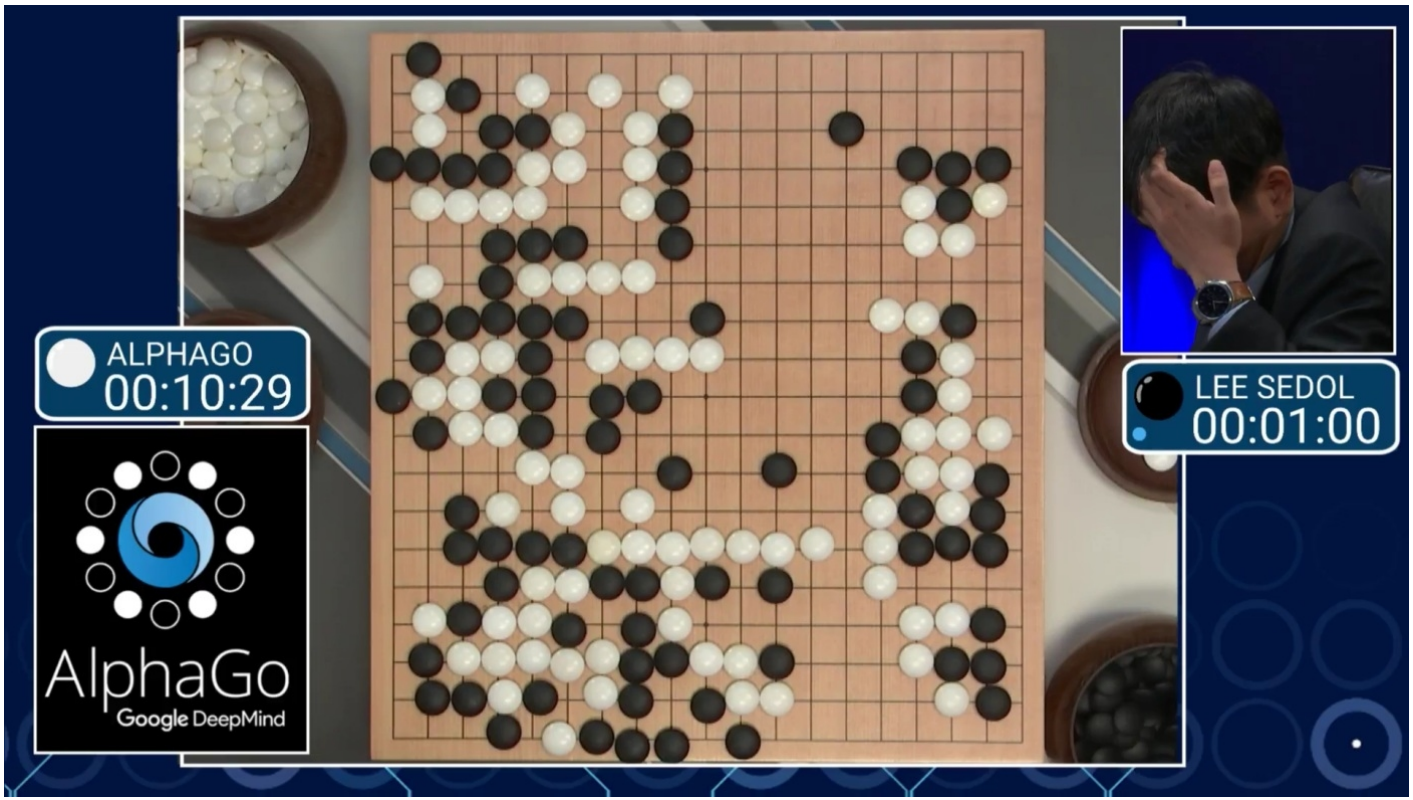
Introduction



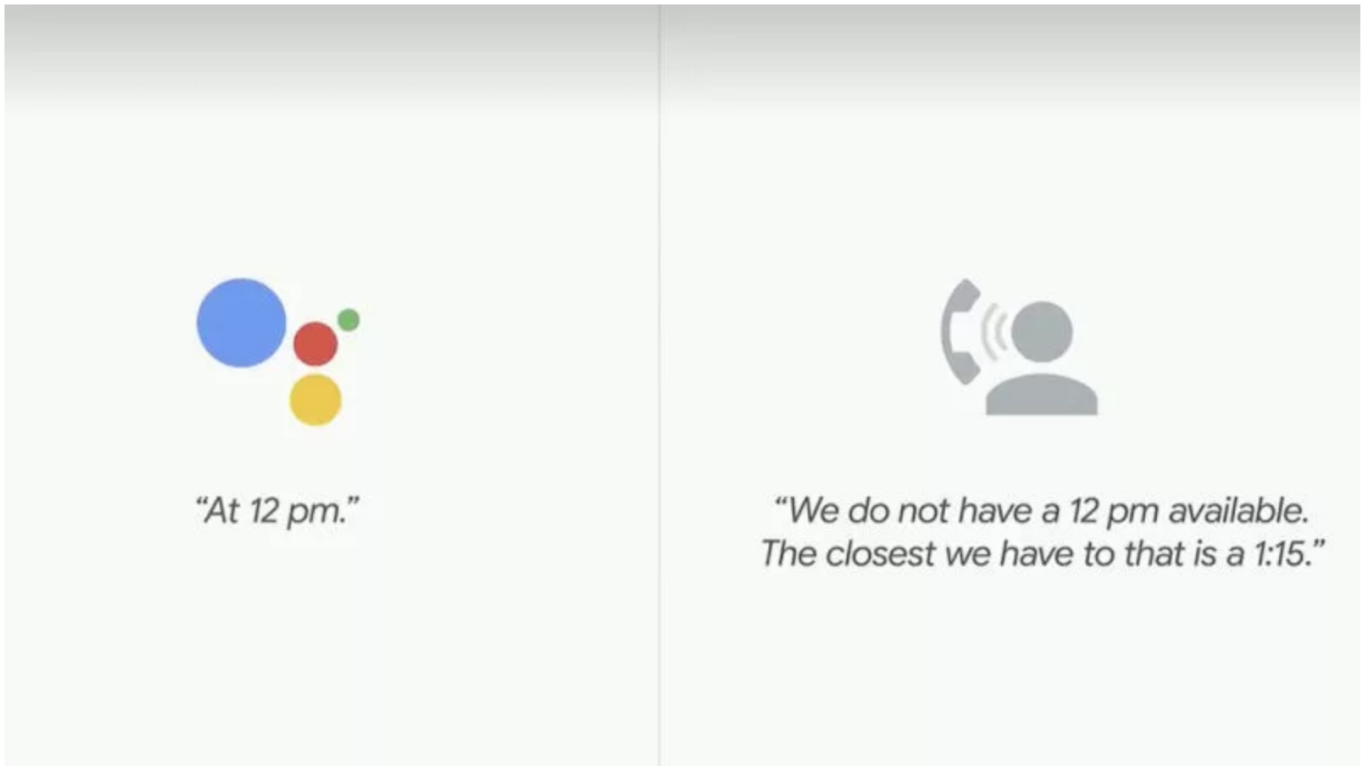
Introduction



Introduction



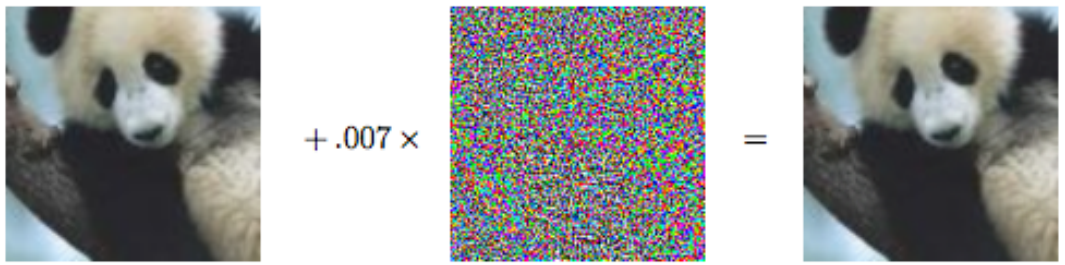
Introduction



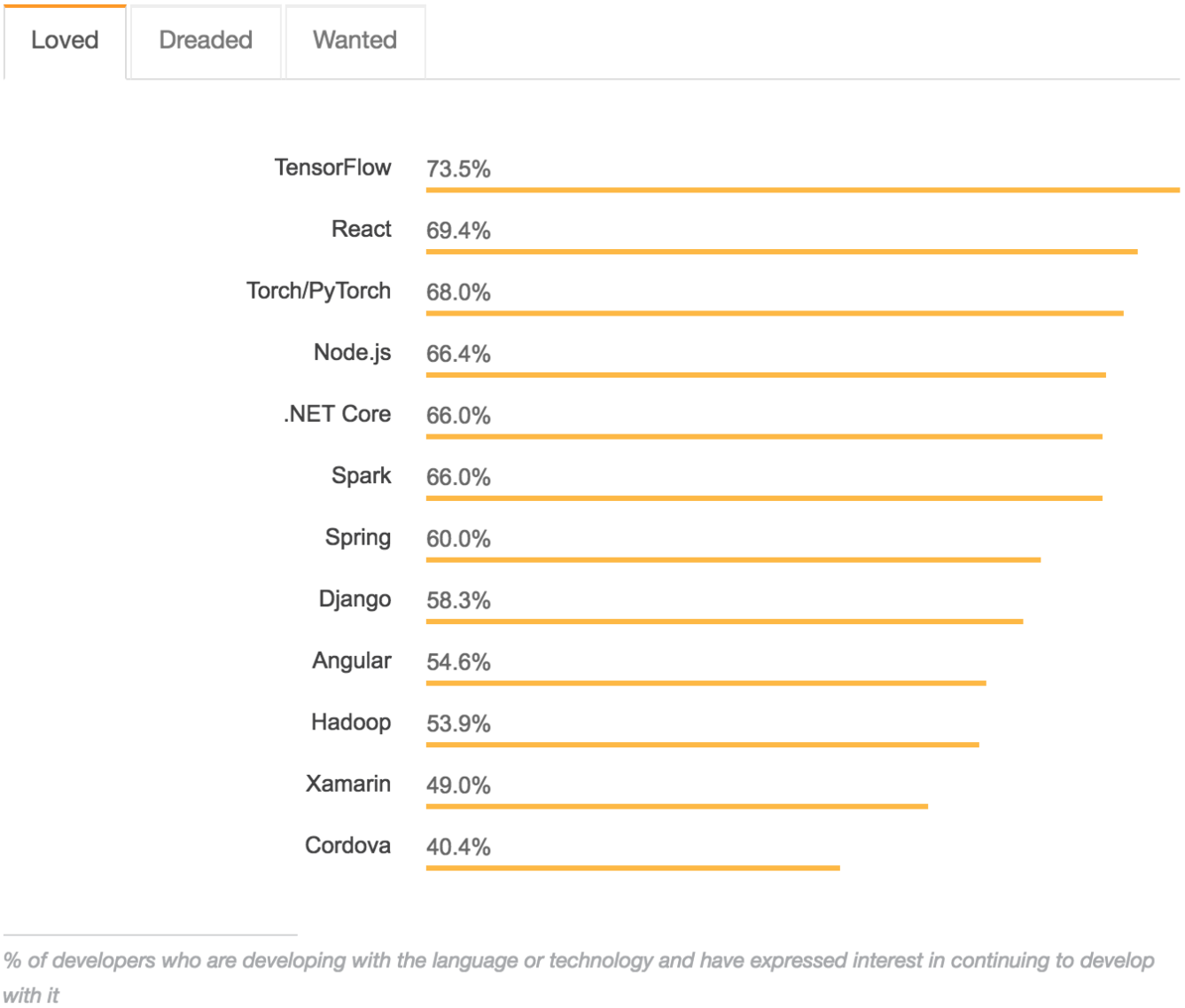
Screenshot: Andrew Liszewski (Gizmodo)

Introduction

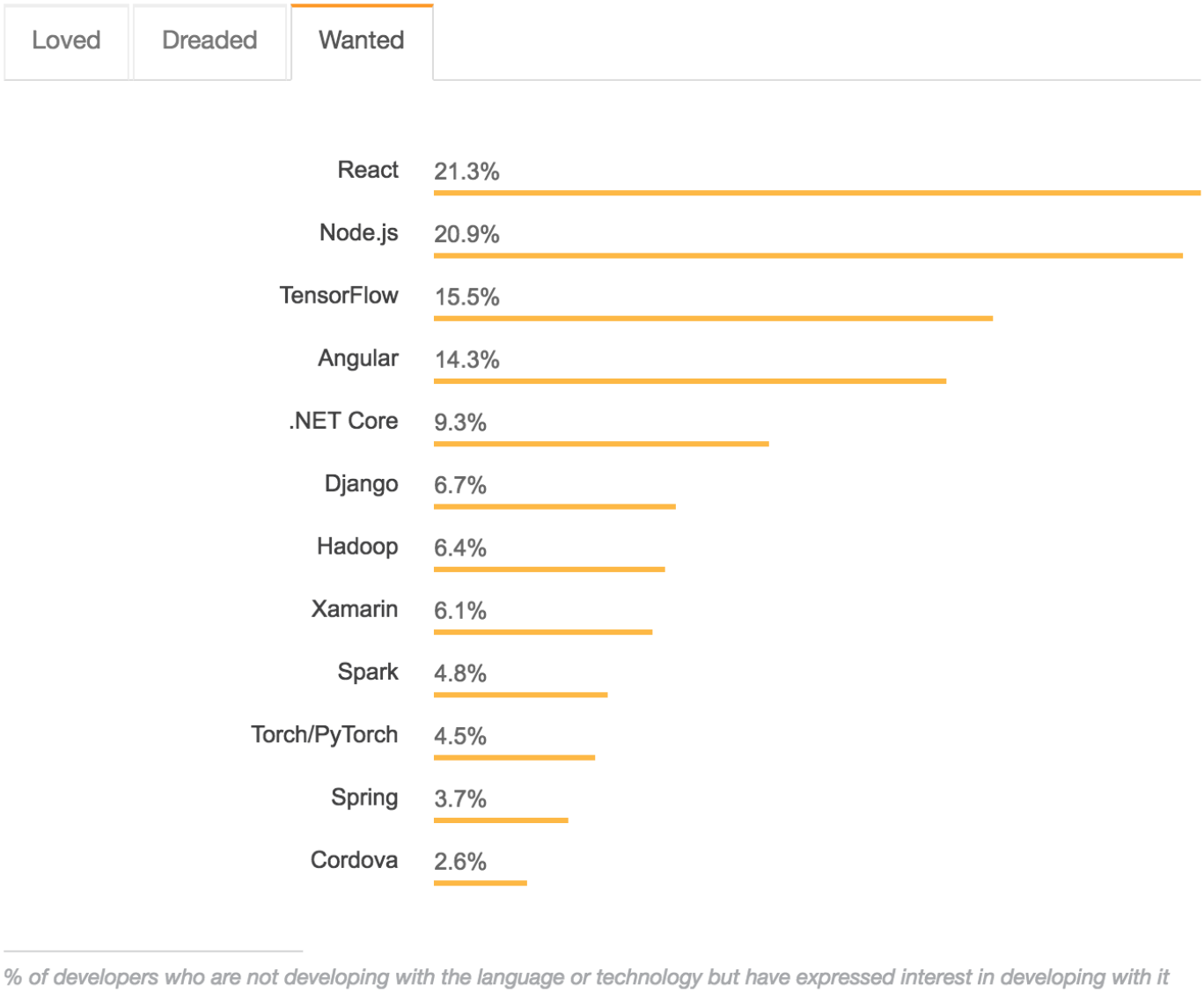
Introduction



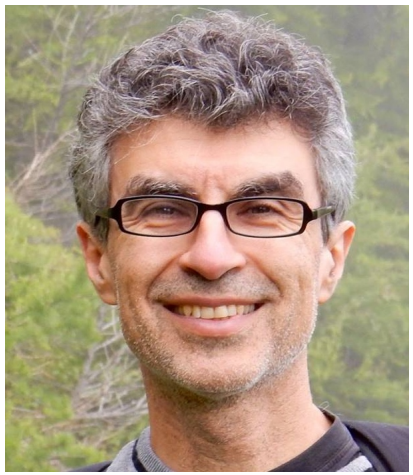
Introduction



Introduction



Introduction



Introduction

Section 1

Welcome

What you can expect

- Introduction to neural networks and deep learning
- Foundations and principles (mathematics!)
- Major model architectures and learning algorithms
- Exemplary use cases, getting intuitions
- Machine learning for machine intelligence
- Hands-on experience (exercise sessions / projects)
- Getting addicted to deep learning :)

What you should not expect

- Learning magic tricks
- Learning machine learning basics (pre-requisite)
- Avoiding mathematics "gets you through"
- Becoming an expert "framework" user (not prevented)
- Solving strong AI in 2019
- Immediate wealth :)

Course Information

- Lecture
  - Mondays, 13-15
  - ETF C 1
- Exercises
  - Mondays, 15-16, CAB G 51
  - Mondays, 16-17, ML F 36
  - Identical sessions
  - Discussion of the exercise sheets
  - No hand-in of homework

Course Information

- Enrollment limit: 300
  - size of largest lecture halls available
  - limited TA resources
  - requirements: see course catalogue
  - **Please drop out early**
- Lectures & material
  - course homepage:
    `www.da.inf.ethz.ch/teaching/2018/DeepLearning`
  - handouts: linked on course homepage
  - communication via piazza
  - textbook: Goodfellow, Bengio, Courville: Deep Learning, 2016

Grading

- Mixed grading scheme
  - 70% Exam
  - 30% Group project

Exam

- During exam session
- Written
- Closed-book
- Scope
  - All lecture content (written & oral)
  - All exercise content (written & oral)
  - Textbook
  - Referenced papers
  - Wikipedia
  - Everything else

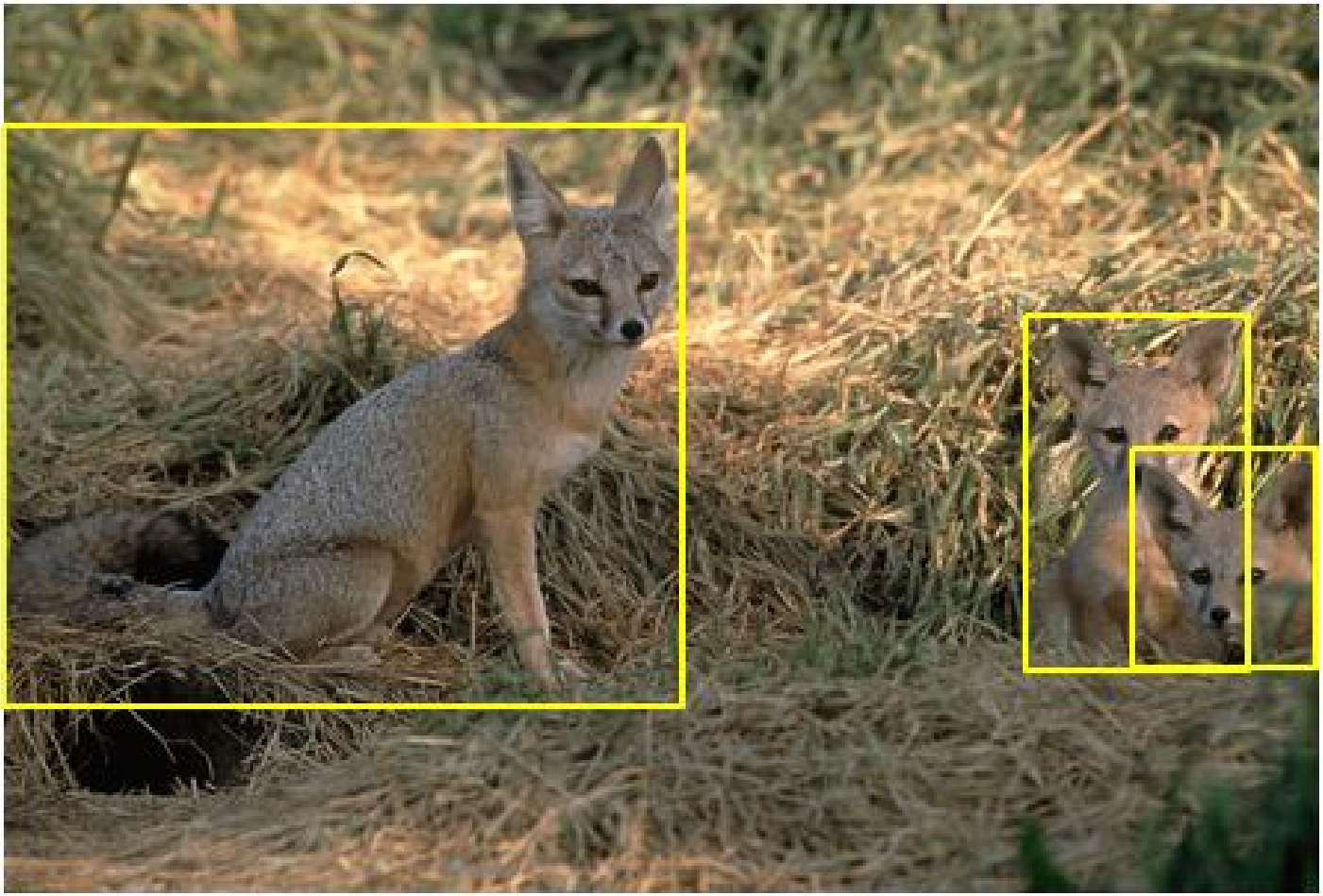
Project

- Hands-on, 40 hours
- Groups of 3-4 people
- You choose your topic
- 1-page proposal due before Christmas
- Report & code due January 19th
- Frameworks: Tensorflow, Pytorch, Keras

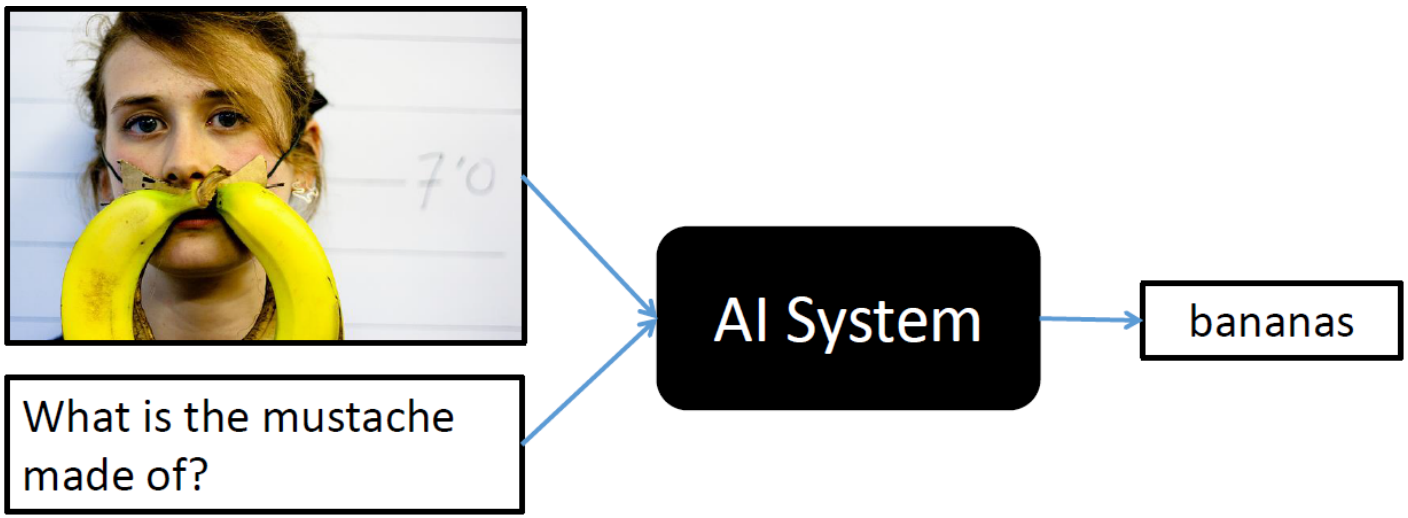
Project Grading

- Quality of report
- Cleanliness of execution
- Soundness of evaluation (comparison to baselines, etc.)
- Presentation and interpretation of results
- Creativity

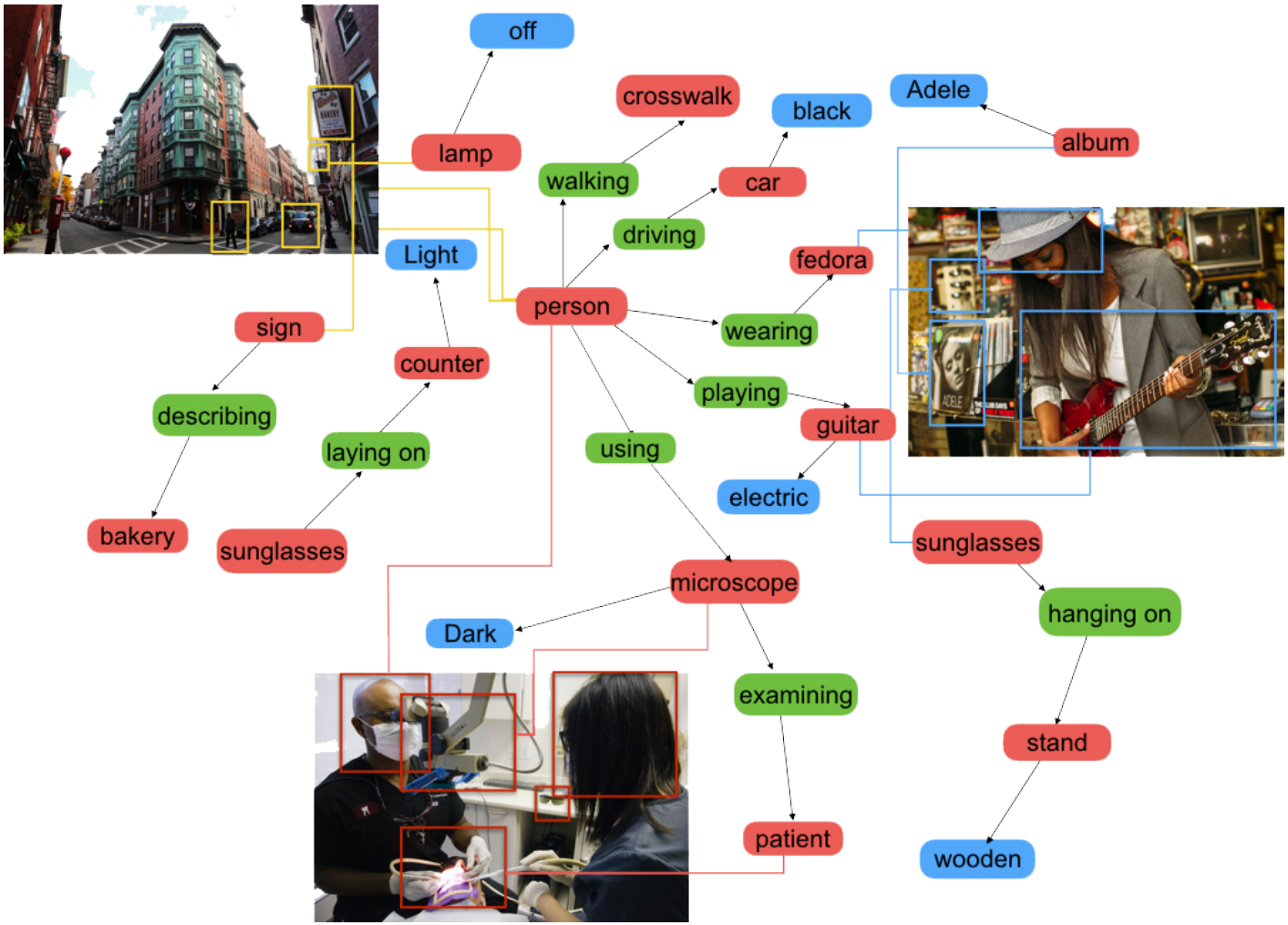
Project Suggestions



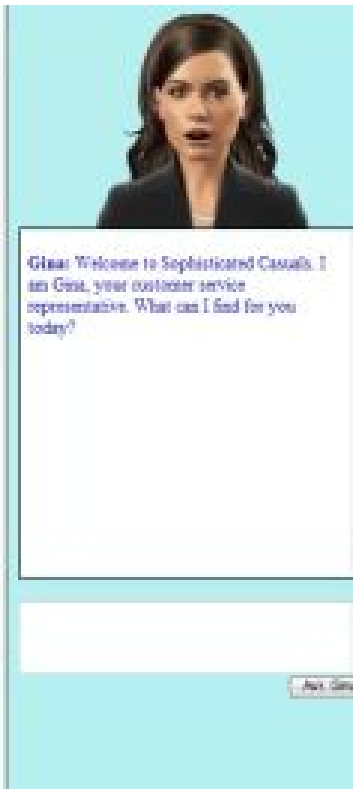
Project Suggestions



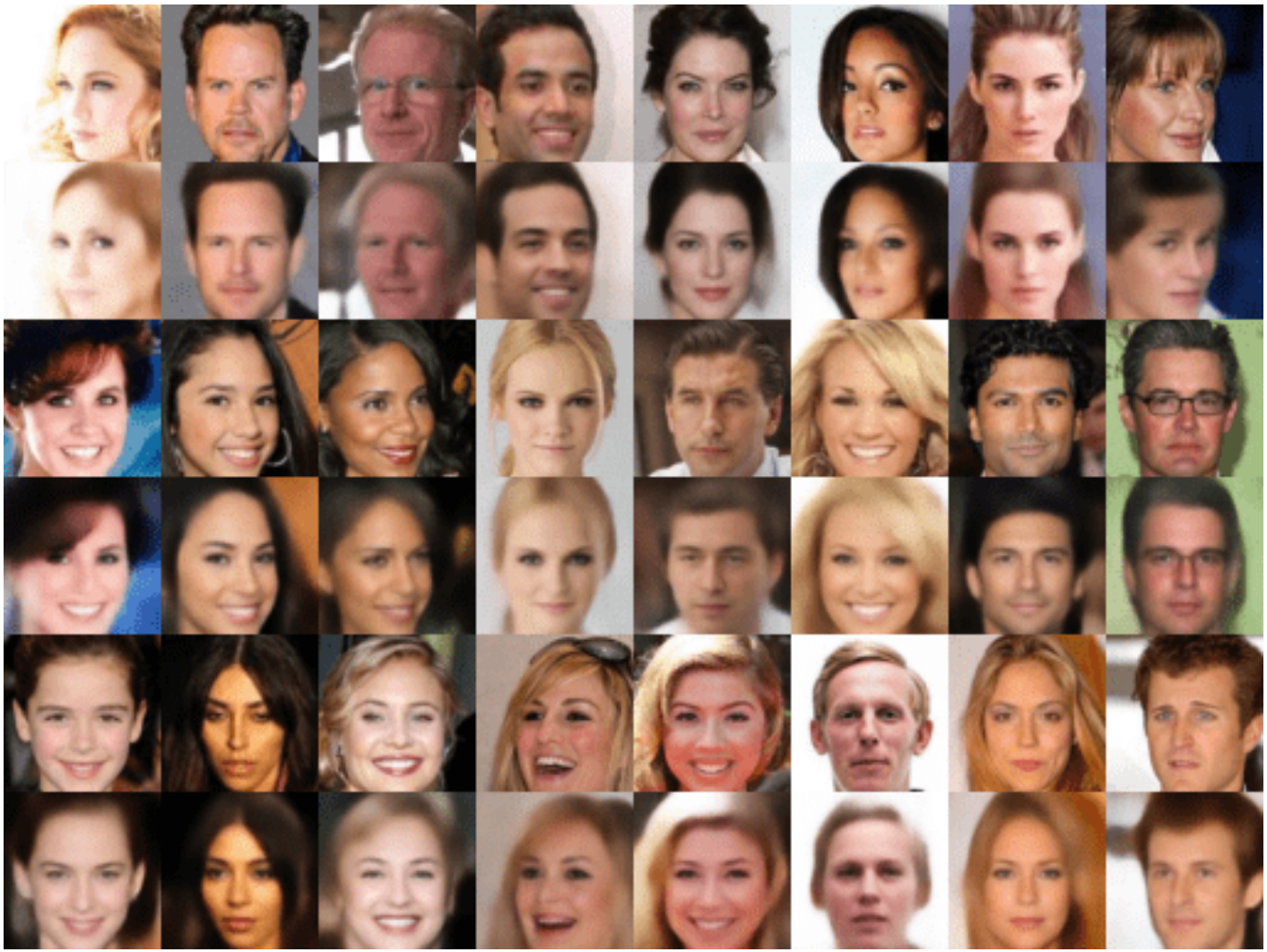
Project Suggestions



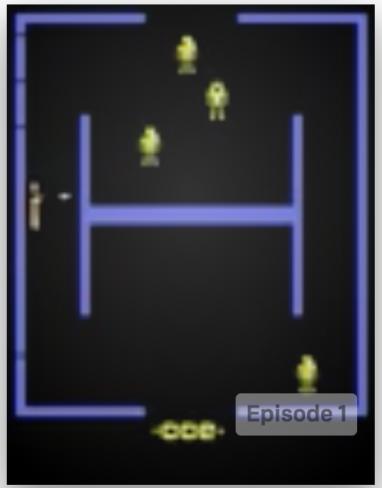
Project Suggestions



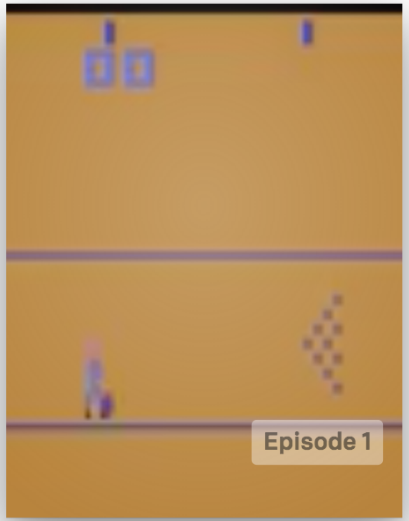
Project Suggestions



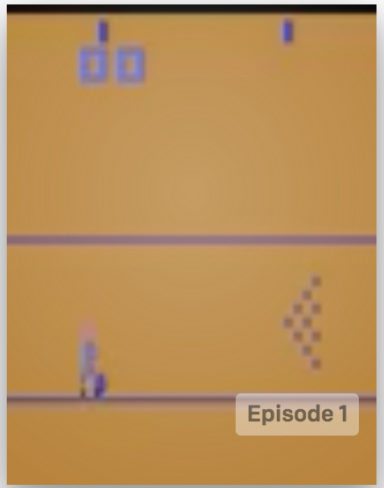
Project Suggestions



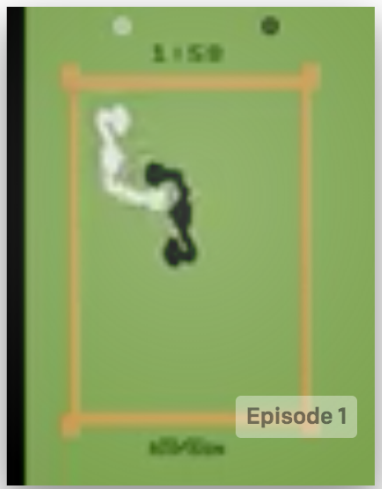
Berzerk-v0
Maximize score in the game Berzerk, with screen images as input

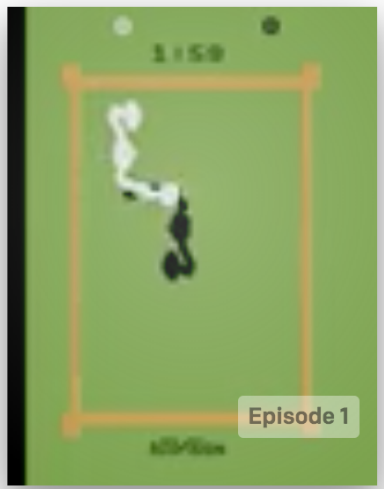


Bowling-ram-v0
Maximize score in the game Bowling, with RAM as input

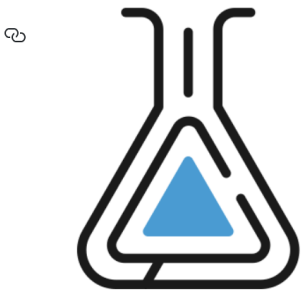


Bowling-v0
Maximize score in the game Bowling, with screen images as input



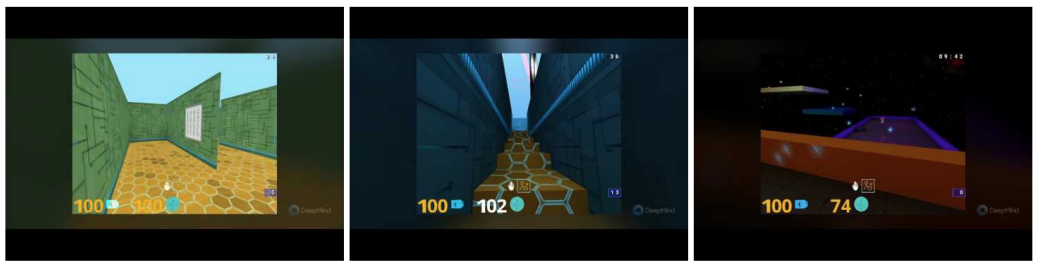


Project Suggestions



DeepMind Lab

DeepMind Lab is a 3D learning environment based on id Software's Quake III Arena via ioquake3 and other open source software.



Project Suggestions

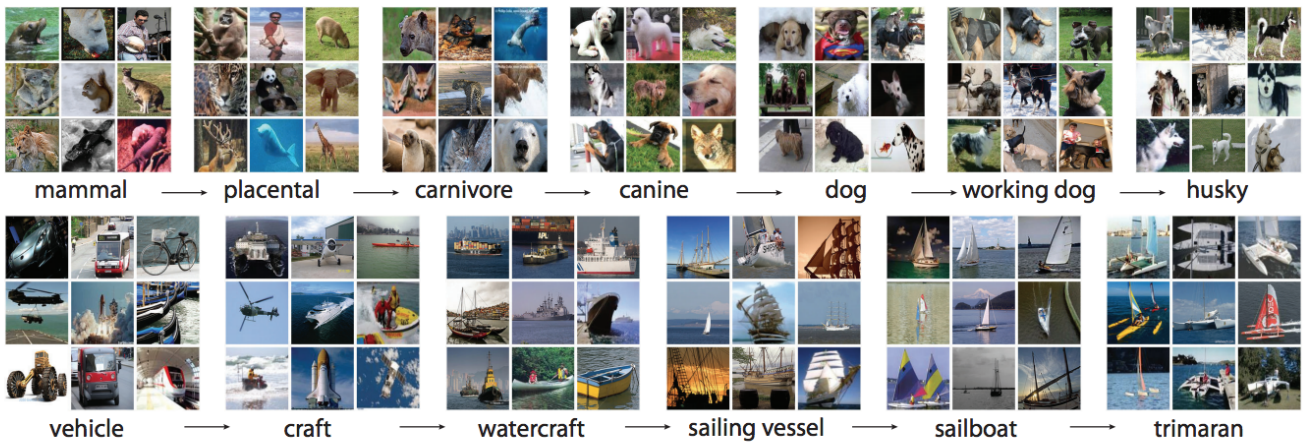


Figure 1: A snapshot of two root-to-leaf branches of ImageNet: the **top** row is from the mammal subtree; the **bottom** row is from the vehicle subtree. For each synset, 9 randomly sampled images are presented.

Project Suggestions

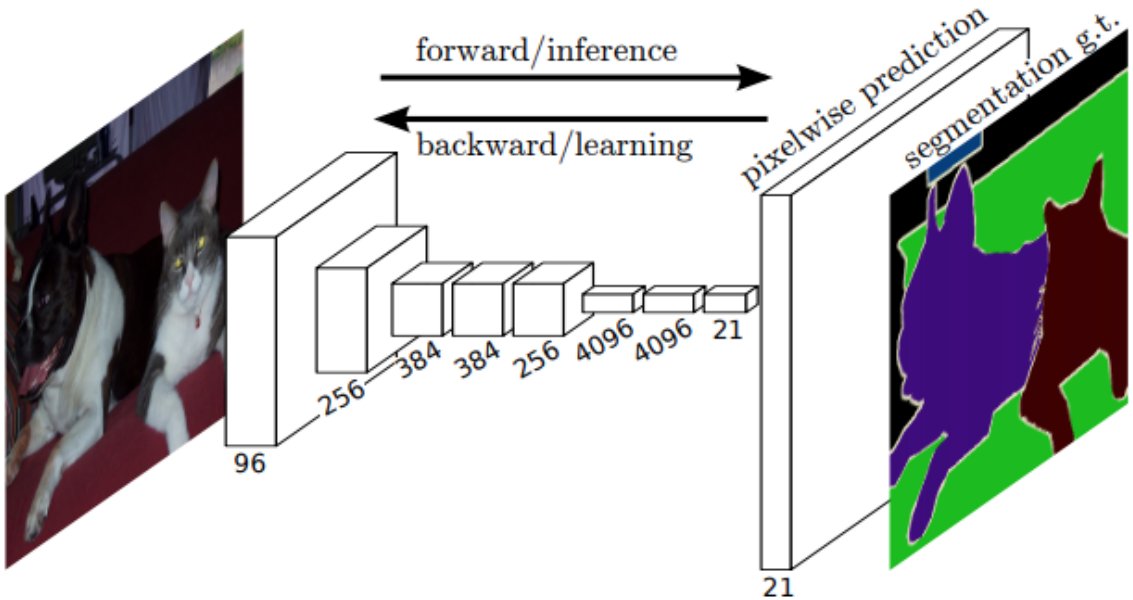
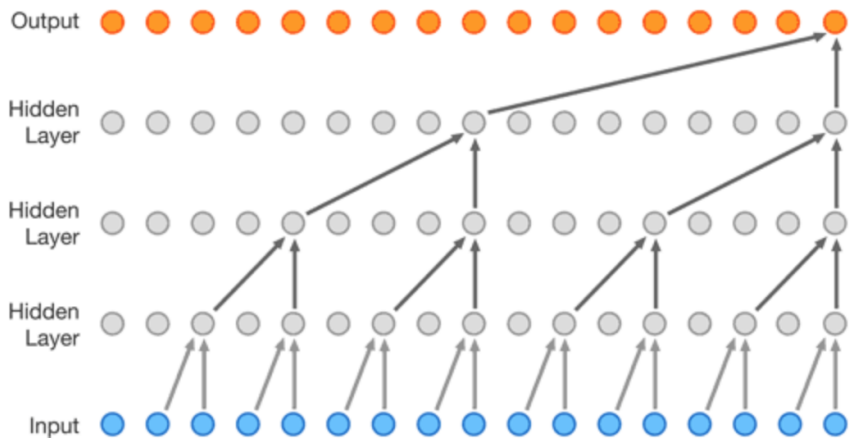


Fig. 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

Project Suggestions

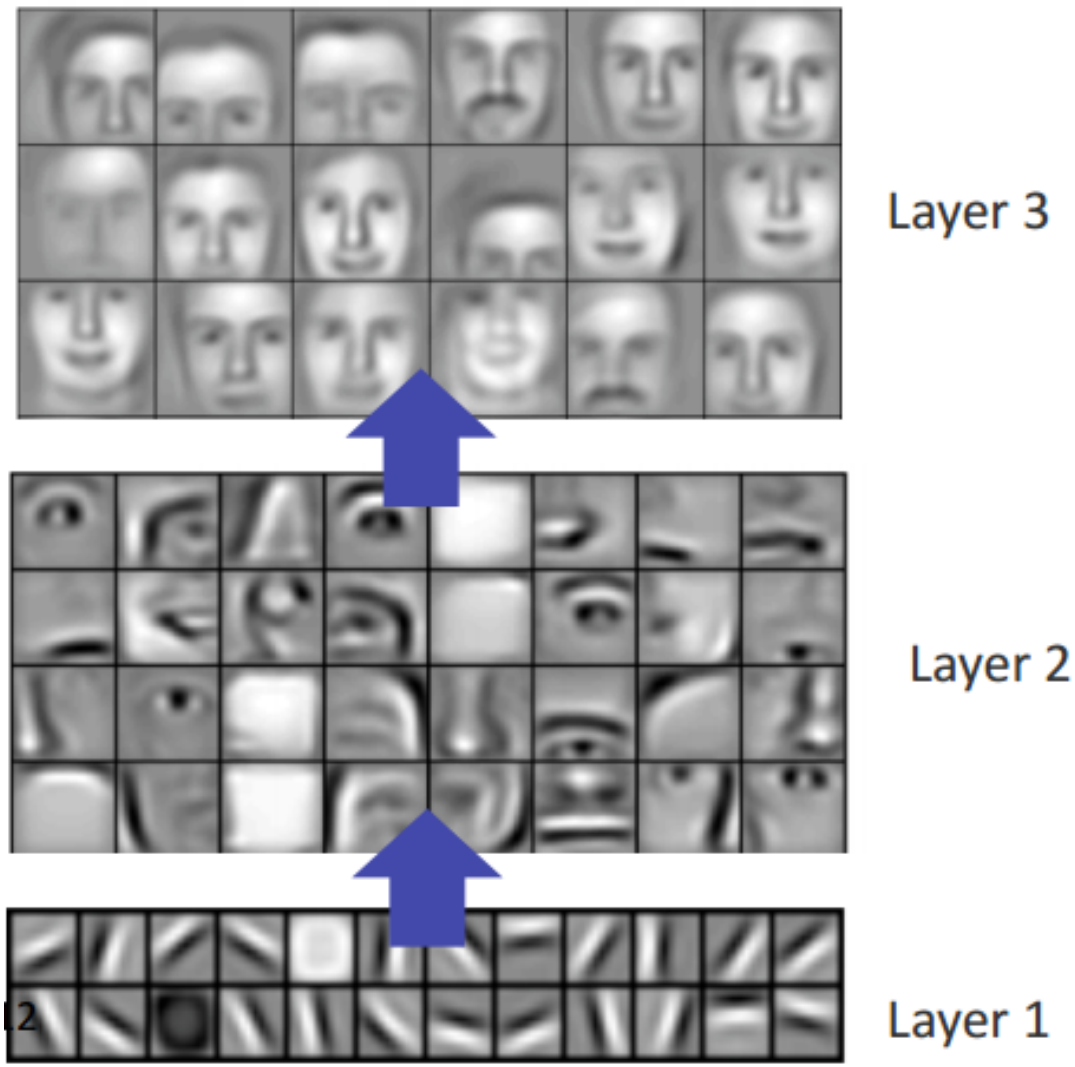


Questions

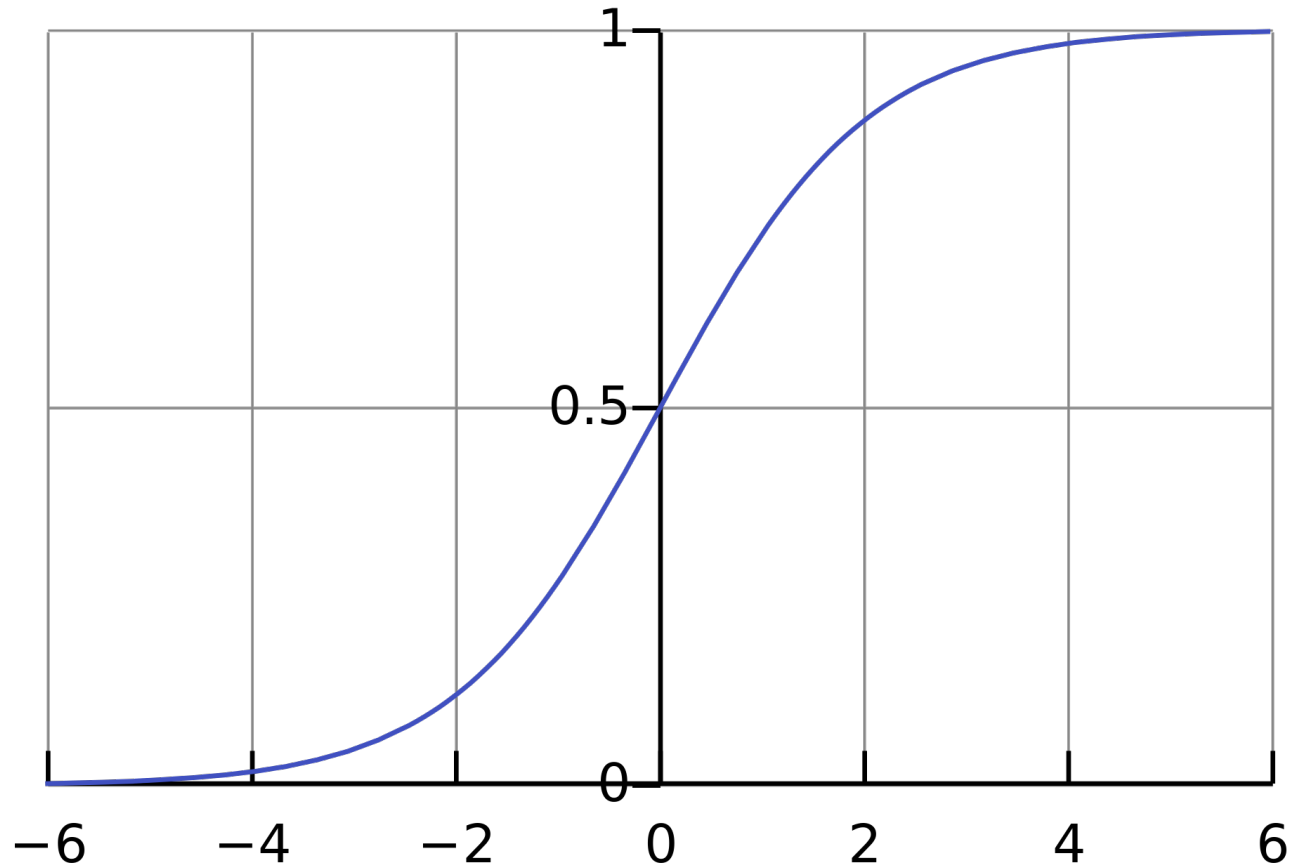
Section 2

Course Overview

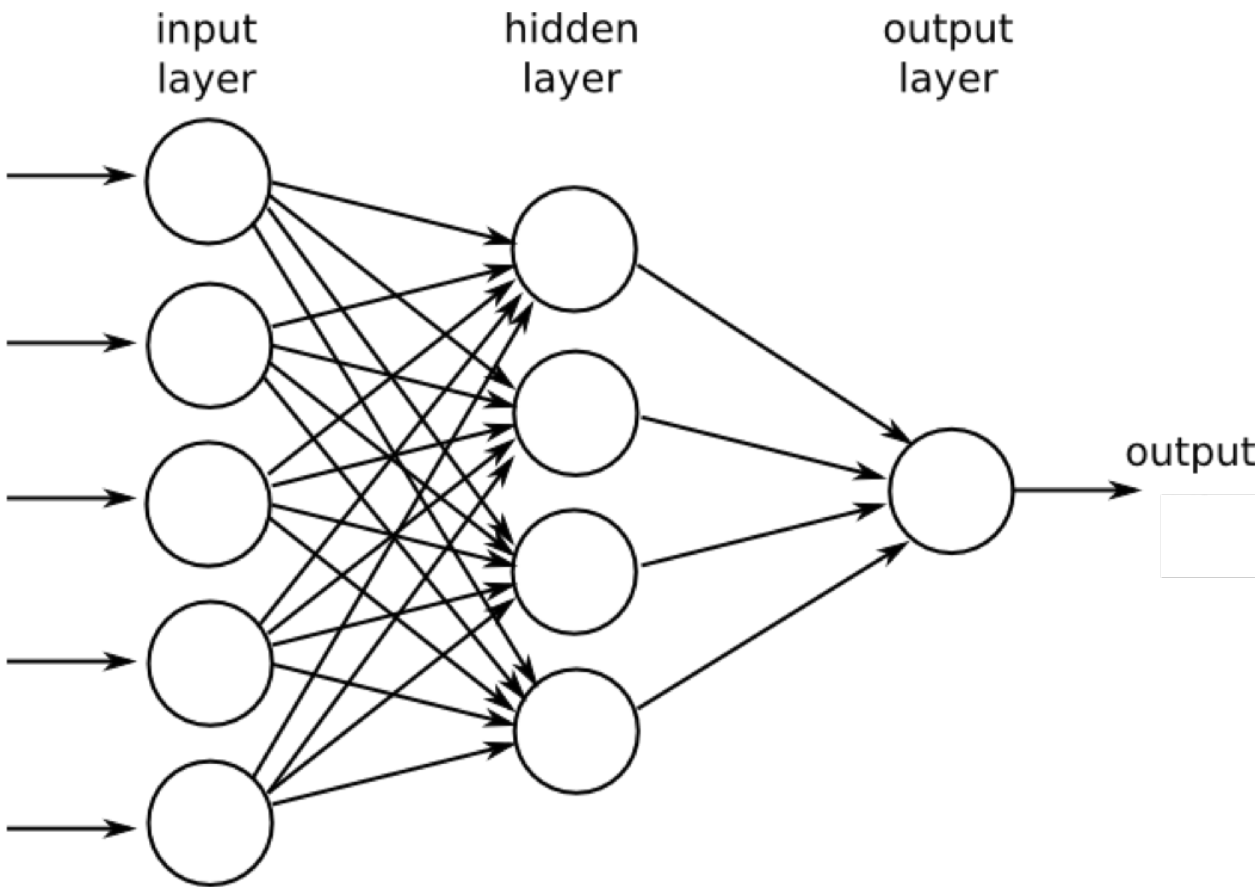
Lecture 1: Compositional Models



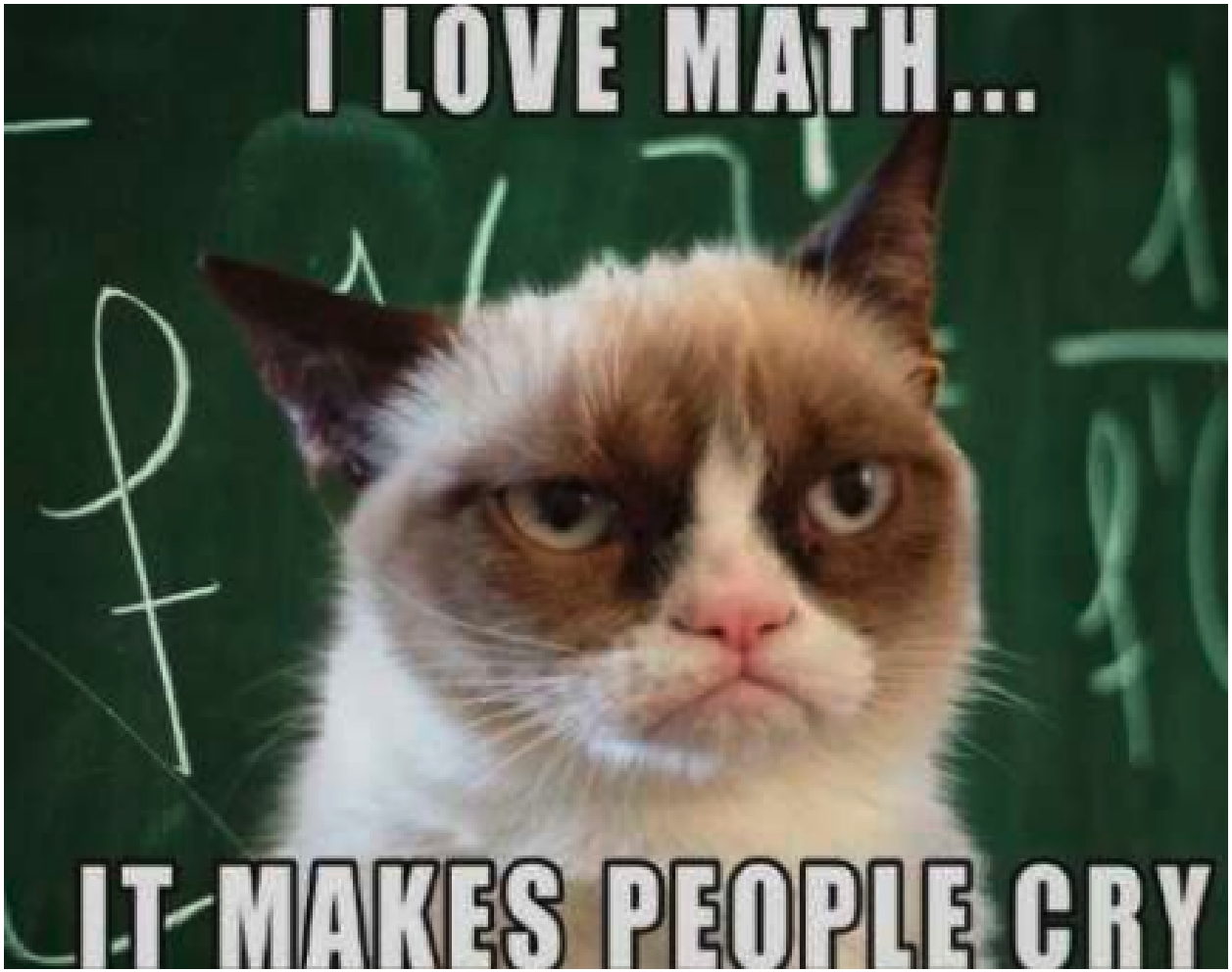
Lecture 2: Approximation theory, Rectification & Sigmoid networks



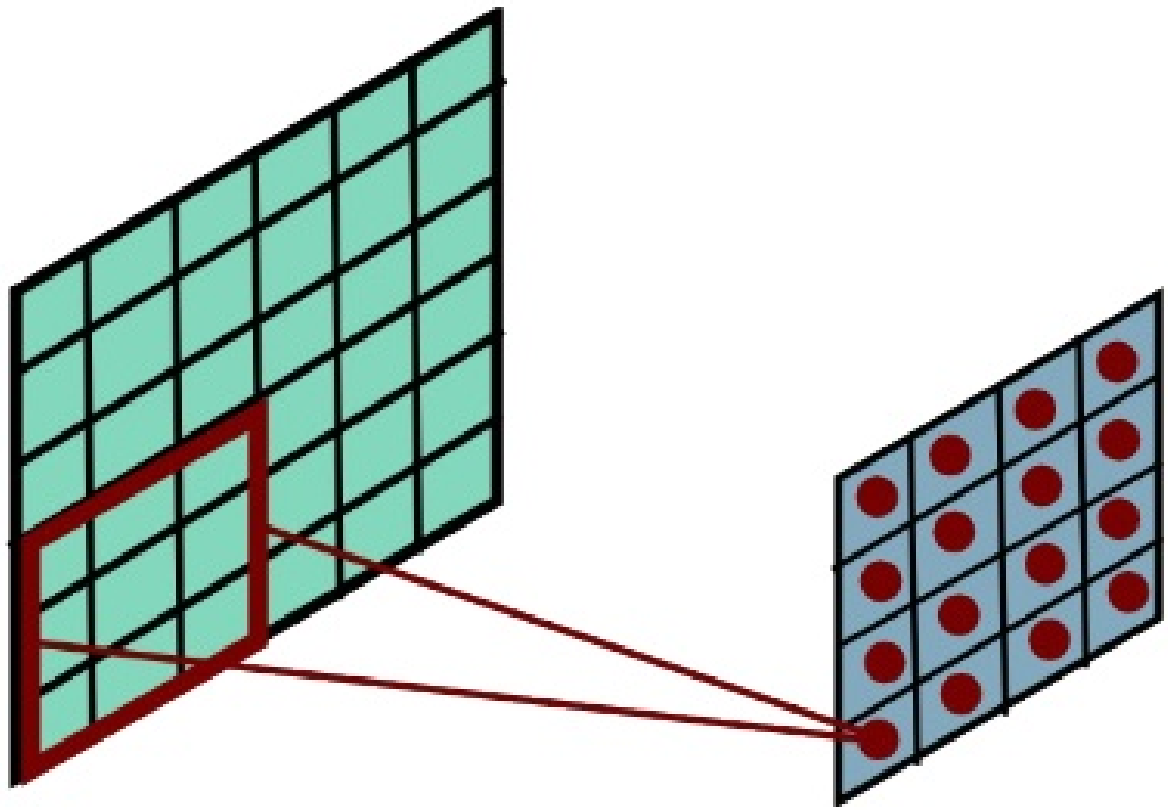
Lecture 3: Feedforward Networks



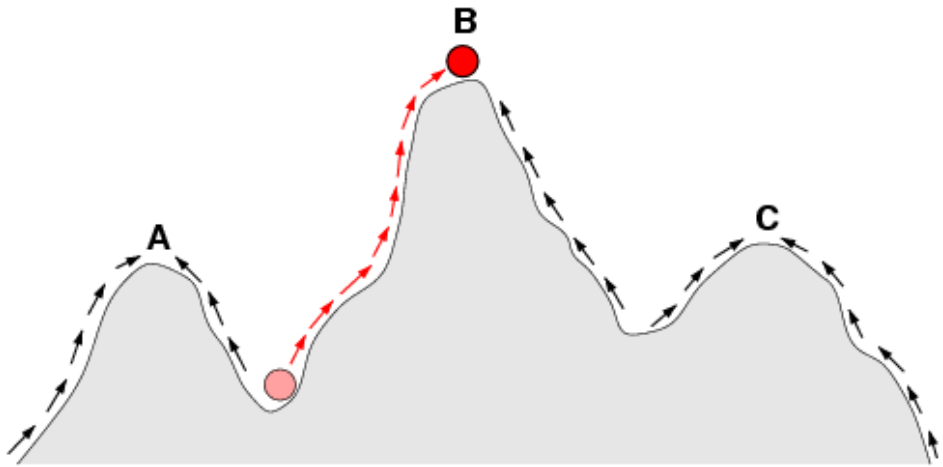
Lecture 4: Backpropagation



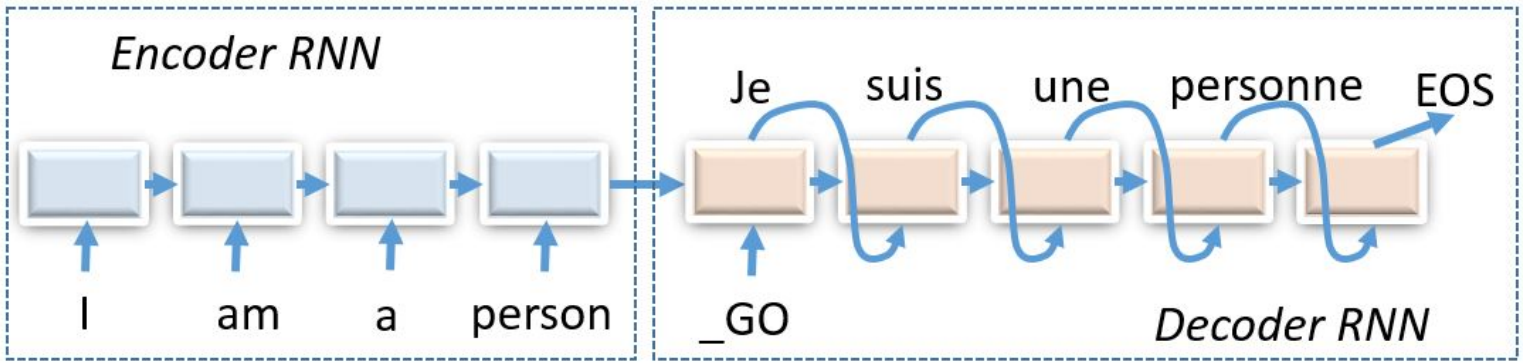
Lecture 5: Convolutional Neural Networks



Lectures 6 & 7: Optimization



Lecture 8: Natural Language Processing



Lecture 9: Memory & Attention

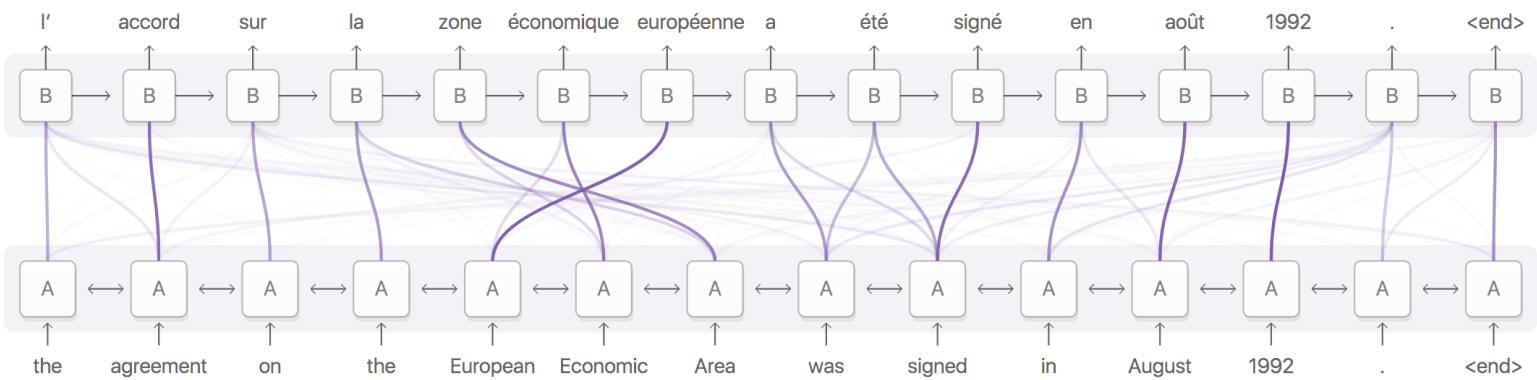
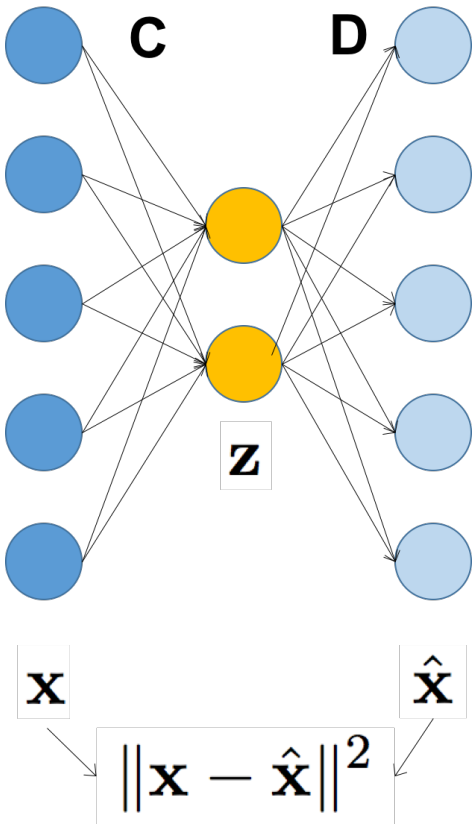


Diagram derived from Fig. 3 of Bahdanau, *et al.* 2014

Lecture 10: Autoencoders



Lecture 11: Density Estimation & Generative Adversarial Models

Section 3

End-to-End Learning

General Purpose Learning Machine

Desiderata:

1. **Flexibility**: diversity of functions (few prior assumptions)
2. **Adaptivity**: class of parametrized functions, generic learning algorithms
3. **Architecture**: modeling power vs. complexity trade-off (suitable design space/metaphors)

Learning with Feature Engineering

Linear & Kernel Methods ("classical"):

1. **Flexibility**: initial feature representation, dimension expansion via kernels (RKHS)

2. **Adaptivity**: linear functions (simple, lowest complexity), easy to understand (Hilbert space), convex optimization (works!)

3. **Architecture**:
   - feature engineering – domain expertise
   - choice of kernels – mostly generic, sometimes domain-specific
   - hyperparameter(s) – grid search, etc.

Conclusion: limitations w/ regard to desiderata

Deep Learning

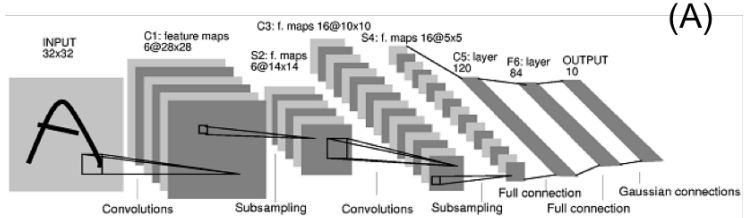
End-to-End Learning Paradigm (aka "Deep Learning"):

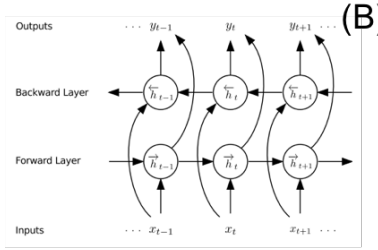
1. **Flexibility**: modularity, compositionally, toolbox principle
2. **Adaptivity**: restricted families of non-linear functions, easy to define, good statistical efficiency, non-convex optimization
3. **Architecture**:
   - layers of representation – width, depth, type: best practice, design patterns, informed exploration
   - re-use of representations – multi-task, pre-training, "AI" etc.
   - input representation – goal: less domain knowledge/feature craft, raw features (informative, but not necessarily explicit)

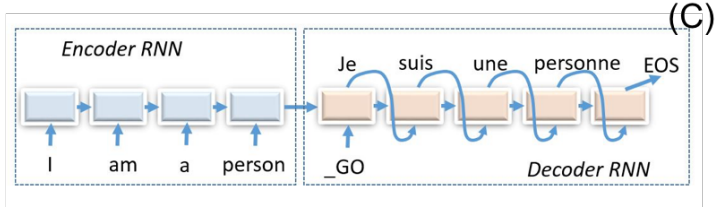
Flexibility

Examples

- Visual object recognition (A)
- Speech recognition (B)
- Natural language understanding (C)



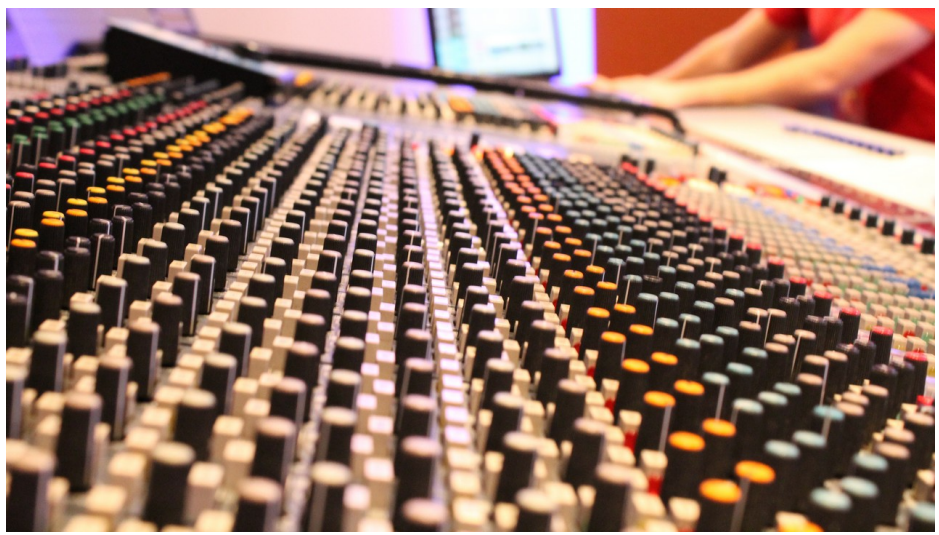




How can we use the same general model family to approximate functions across all/many use cases?

Adaptivity

- Example: visual object recognition - huge models
  - AlexNet: 61 million parameters
  - VGG-16: 138 million parameters
  - ResNet: < 2 million parameters (extremely deep)



Architecture

DL "Guru Jargon":

- DL frameworks, e.g. Tensorflow
- metaphors, e.g. "healthy" gradients, parameter sharing
- insights & intuitions, e.g. LSTM vs. recurrent vs. convolutional
- data augmentation "tricks", e.g. distant training, pre-training
- best practices, e.g. similar architectures across many tasks
- model re-use, e.g. word embeddings, CNNs
- optimization: batch normalization, ADAM, etc.
- vs. domain expert, who carefully crafts features and model

Art or Science?

- Demis Hassabis (CEO, DeepMind):
  - *It's almost like an art form to get the best out of these systems. There's only a few hundred people in the world that can do that really well.* (Wired Magazine, 05/2016)
- IMHO: not really and it will surely not stay that way
- Scientific challenge:
  - deep learning success has shown: theory of learning is vastly insufficient
  - need to gain deeper understanding of models, architectures, algorithms, optimization, etc.

End-to-End Learning

Success Factors

- Availability of very large datasets
- Abundance of compute resources
- Leverage of representation learning
  (cross-task, distant, semi-/unsupervised, etc.)
- Complexity of the domain, difficulty of problem

Section 4

Compositional Models

Compositional Models

- Given (implicitly): function

$$F^* : \mathbb{R}^n \to \mathbb{R}^m$$

  - via noisy examples $(\mathbf{x}_i \mapsto \mathbf{y}_i)$
  - complicated, highly non-linear, (possibly stochastic)

- Learning: approximate F^* within class of functions

$$F : \mathbb{R}^n (\times \mathbb{R}^d) \to \mathbb{R}^m, \quad \mathcal{F} := \{F(\cdot, \theta)\}$$

  - parametrized via parameters of weights θ
  - d-dimensional family
  - model selection: chose suitable $\mathcal{F}$
  - model fitting: find best $F \in \mathcal{F}$ such that $F \approx F^*$

Compositional Models

- Composition (L-fold): $F = G_L \circ \cdots \circ G_2 \circ G_1$ such that

$$F(\mathbf{x}; \theta) = G_L(\ldots G_2(G_1(\mathbf{x}; \theta_1); \theta_2) \ldots; \theta_L)$$

  - restrict G_j to simple(r), fixed class of functions
  - each G_j is parametrized by θ_j (e.g. weight matrix, biases)
  - chain of mappings ($\mathbb{R}^*$: arbitrary dimensionality)

$$\mathbb{R}^n \xrightarrow{G_1} \mathbb{R}^* \xrightarrow{G_2} \mathbb{R}^* \xrightarrow{G_3} \ldots \xrightarrow{G_L} \mathbb{R}^m$$

- Intermediate representations = layers

Hierarchical Representations

- Goal: learn hierarchical data representations
  - cf. Bengio et al. 2013, Representation learning: A review and new perspectives

- "Depth" in deep learning

  = level of functional nesting or composition

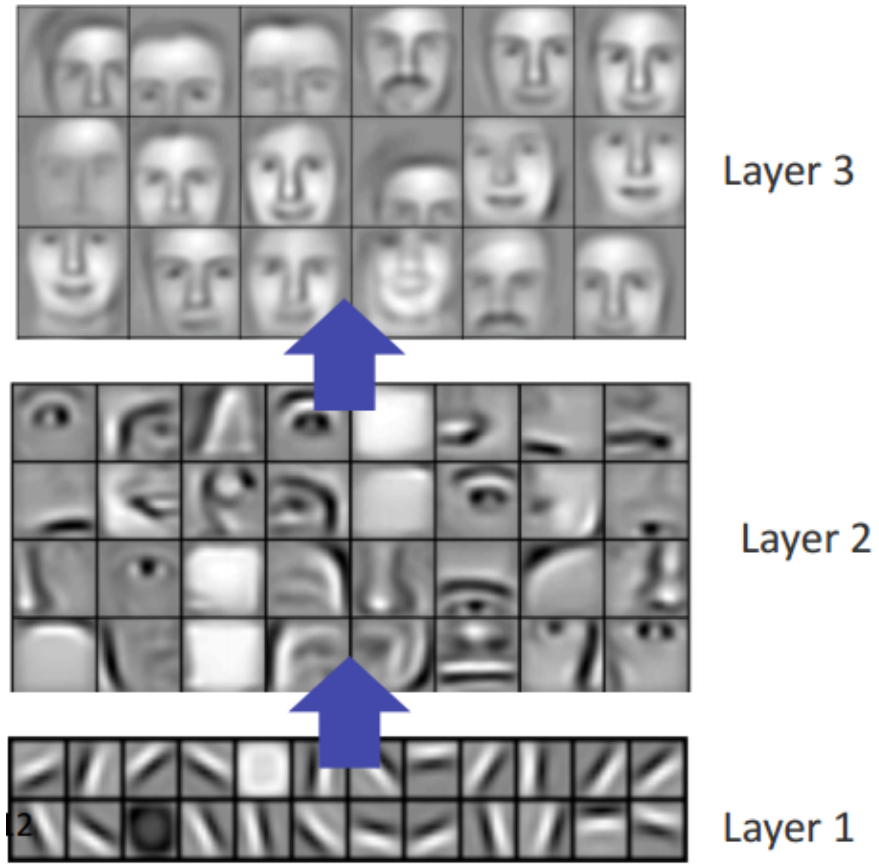
  = height of abstraction level (generative view)

- Deep Architectures
  - feature re-use and re-combination (at higher levels)
  - successively more abstract representations
  - successively more invariant, less noise-sensitive representations (but: what do we want to be invariant to?)

Visual Feature Hierarchy

Prototypical example: visual feature hierarchy

- Lee et al., Convolutional Deep Belief Networks, 2009
- from oriented edges to parts to objects



Deep vs. Shallow Networks

Not all successful networks are deep, e.g. in natural language processing.

Need to understand simpler, shallow networks, before we look at deeper nesting.

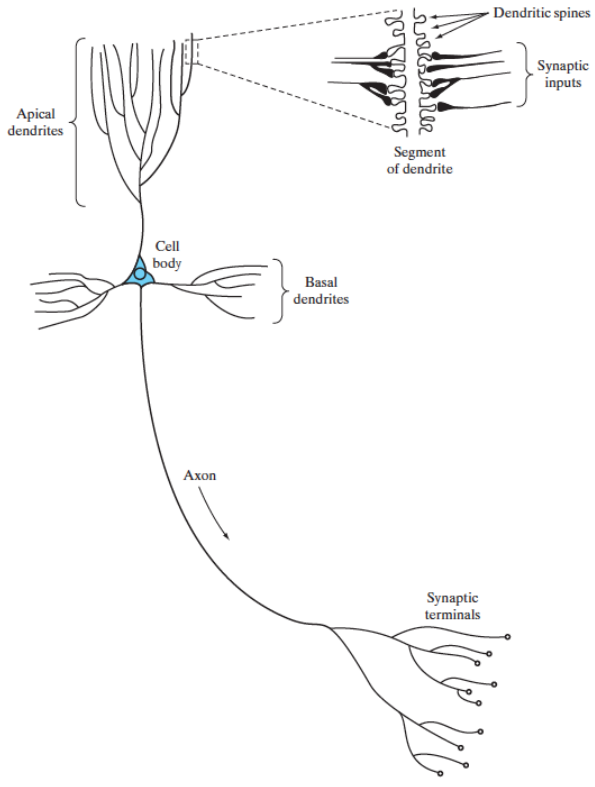
Section 5

Elements of Computation

Computational Units

Metaphor: computational unit = neuron
(signal-processing, biological cell)

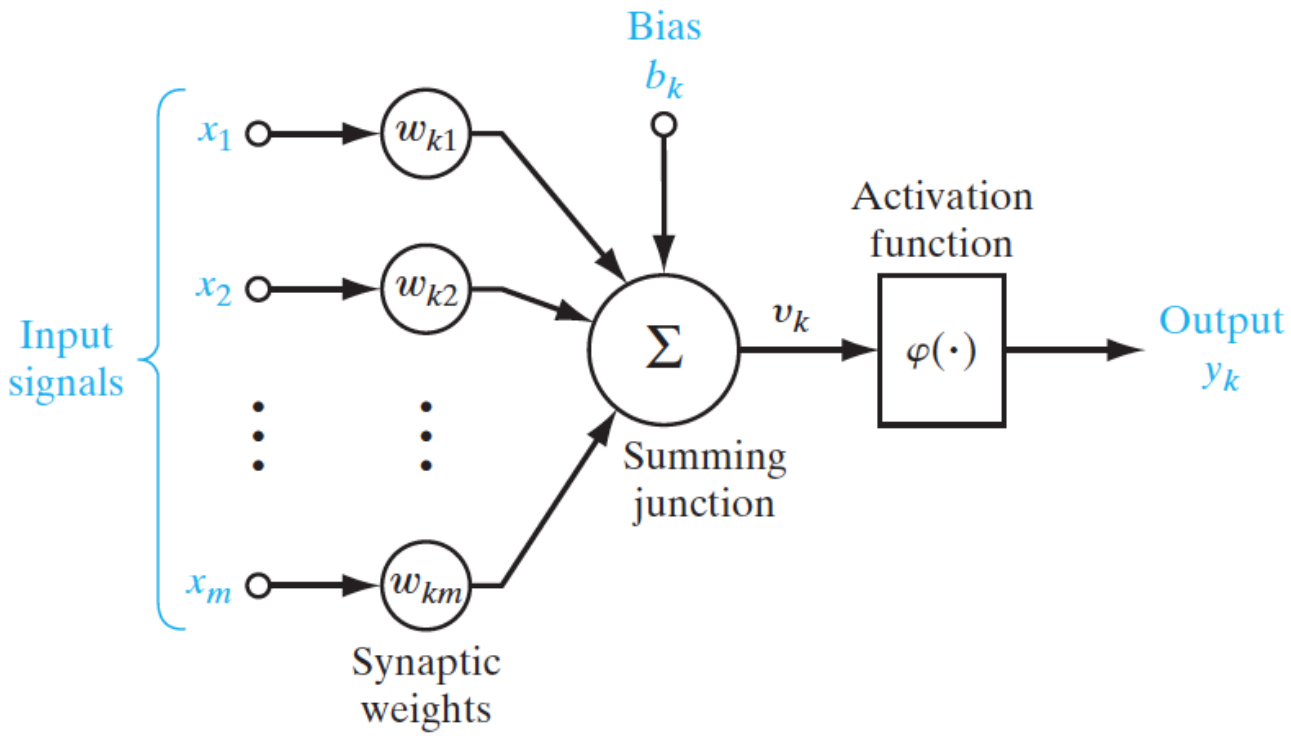
- dendritic tree: integrating incoming signals = inputs
- axon: action potentials, firing rate = output
- synapses: connection strengths = weights
- connectivity: e.g. cortex = layer structure



Artificial networks: "resemblance" with biological networks
(should not be over-interpreted)

Neuron: Schematic View

Mathematical abstraction of basic neuron



(from Haykin, Neural Networks and Learning Machines, 2009)

Linear Functions

Definition: Linear function

A function $f : \mathbb{R}^n \to \mathbb{R}$ is a linear function if the following holds:

$$f(\mathbf{x} + \mathbf{x}') = f(\mathbf{x}) + f(\mathbf{x}'), \quad (\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n)$$

$$f(\alpha \mathbf{x}) = \alpha f(\mathbf{x}), \quad (\forall \alpha \in \mathbb{R})$$

Linear Units

Linear functions = simplest non-trivial function

- weighted summing of inputs (intuitive and robust)

Proposition: f linear $\iff$ $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$ for some $\mathbf{w} \in \mathbb{R}^n$.

$\Longleftarrow$: By properties of the scalar product

$$f(\mathbf{x} + \mathbf{x}') = \mathbf{w}^\top(\mathbf{x} + \mathbf{x}') = \mathbf{w}^\top \mathbf{x} + \mathbf{w}^\top \mathbf{x}' = f(\mathbf{x}) + f(\mathbf{x}')$$
$$f(\alpha \mathbf{x}) = \mathbf{w}^\top(\alpha \mathbf{x}) = \alpha \mathbf{w}^\top \mathbf{x} = \alpha f(\mathbf{x})$$

$\Longrightarrow$: Write $\mathbf{x} = \sum_{i=1}^n x_i \mathbf{e}_i$ in canonical basis. Linearity implies

$$f(\mathbf{x}) = \sum_{i=1}^n x_i f(\mathbf{e}_i), \quad \text{identify} \quad w_i := f(\mathbf{e}_i)$$

Level Sets

Definition: Hyperplane

A hyperplane is an affine subspace of co-dimension 1

Definition: Level set

The level sets of a function $f : \mathbb{R}^n \to \mathbb{R}$ is a one-parametric family of sets defined as

$$L_f(c) := \{\mathbf{x} : f(\mathbf{x}) = c\} = f^{-1}(c) \subseteq \mathbb{R}^n$$

Proposition: Level sets of linear functions

Let $f : \mathbb{R}^n \to \mathbb{R}$ be linear, $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$, then

$$L_f(c) = \{\mathbf{x} : \mathbf{w}^\top \mathbf{x} = c - b\} = \text{hyperplane} \perp \mathbf{w}$$

Linear Maps

From linear functions to linear (affine) maps

- vectorized notation, $F : \mathbb{R}^n \to \mathbb{R}^m$, with

$$F(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \dots \\ f_m(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \mathbf{w}_1^\top \mathbf{x} + b_1 \\ \mathbf{w}_2^\top \mathbf{x} + b_2 \\ \dots \\ \mathbf{w}_m^\top \mathbf{x} + b_m \end{pmatrix} = \begin{pmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \dots \\ \mathbf{w}_m^\top \end{pmatrix} \mathbf{x} + \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_m \end{pmatrix}$$

$$=: \mathbf{W}\mathbf{x} + \mathbf{b}$$

- $\mathbf{W}$: weight matrix, "fully connected layer"

Composing Linear Units

Proposition: Composition of linear maps

Let $F_1, \ldots, F_L$ be linear maps, then $F = F_L \circ \cdots \circ F_1$ is also a linear map.

Proof: Represent F_l (affine coordinates) by its weight matrix W_l,

$$F(\mathbf{x}) = (\mathbf{W}_L \ldots (\mathbf{W}_2(\mathbf{W}_1\mathbf{x})) \ldots) = (\underbrace{\mathbf{W}_L \ldots \mathbf{W}_2\mathbf{W}_1}_{=:\mathbf{W}})\,\mathbf{x},$$

which follows from the definition of matrix multiplication. □

- every L-level hierarchy collapses to one level
- note that $\text{rank}(F) \leq \min_l \text{rank}(F_l)$, $\text{rank}(F) \equiv \dim(\text{im}(F))$

$\Longrightarrow$ need to move beyond linearity

Rectified Units

What is a "modest" generalization of linear maps?

- General idea: keep level set structure of linear function
- Variant #1: Piecewise linear functions
  - rectified activation functions
  - "simple pieces" = polyhedra, map is linear on each
  - continuous, but typically non-differentiable (at border faces)

Generalized Linear Units

What is a "modest" generalization of linear maps?

- General idea: keep level set structure of linear function
- Variant #2: Generalized linear functions
  - invertible non-linearity, e.g. sigmoid functions
  - typically C^∞ (smooth)

Next Step

Develop some basic results from approximation theory

$$\Longrightarrow$$

Guidance for choosing non-linearities