ISSCC, San Francisco, 2019-02-18

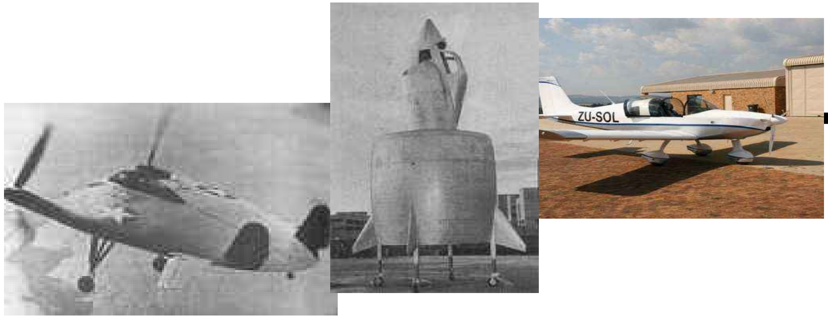
Deep Learning Hardware: Past, Present, & Future

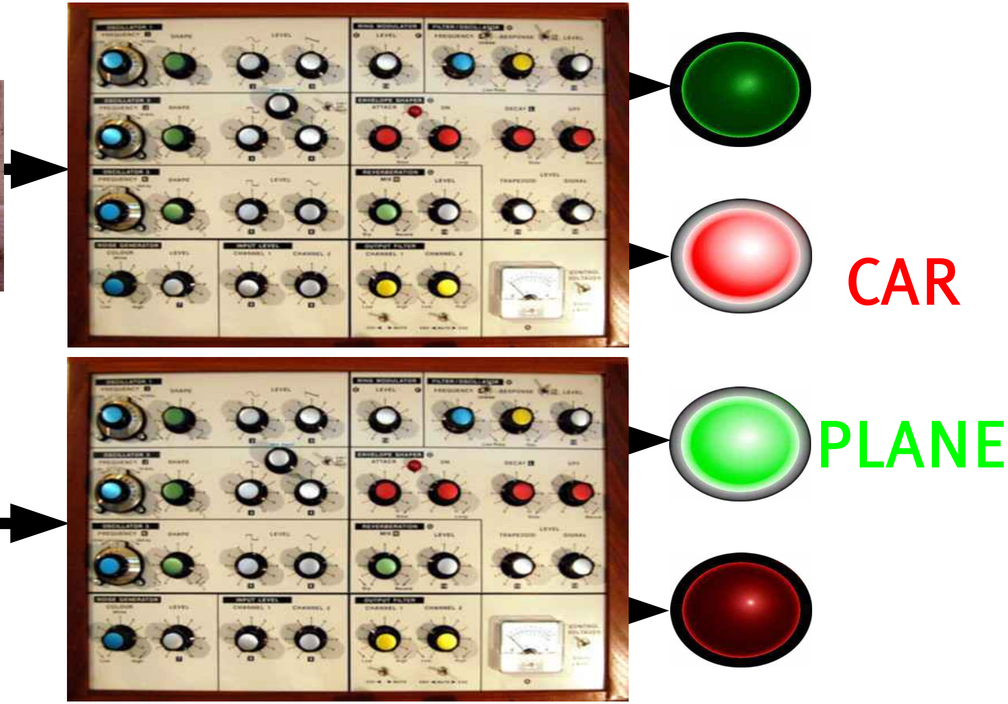
Yann LeCun
Facebook AI Research
New York University
http://yann.lecun.com

facebook
Artificial Intelligence Research

AI today is mostly supervised learning

- **Training a machine by showing examples instead of programming it**
- **When the output is wrong, tweak the parameters of the machine**
- **Works well for:**
  - Speech → words
  - Image → categories
  - Portrait → name
  - Photo → caption
  - Text → topic
  - ….

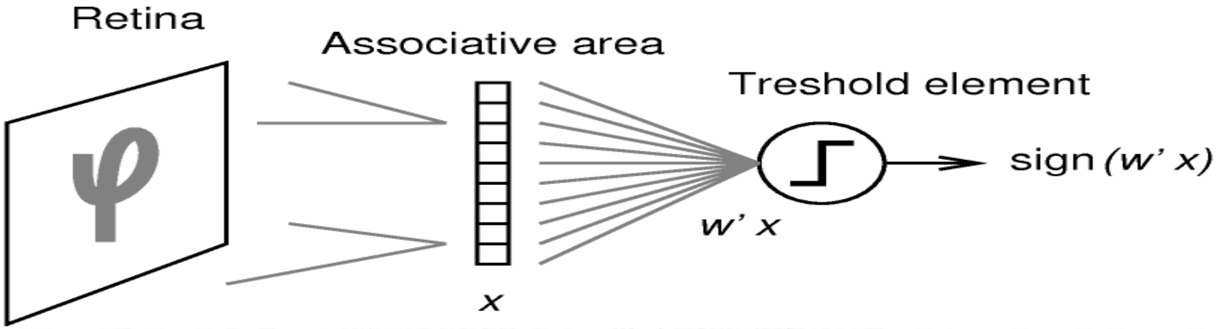




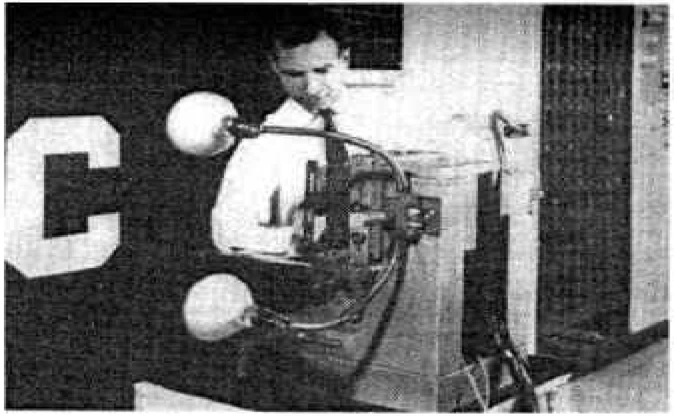
The History of Neural Nets is Inextricable from Hardware

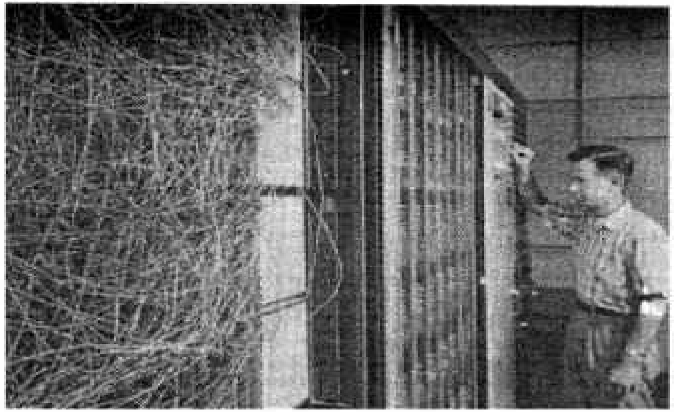
- **The McCulloch-Pitts Binar Neuron**
  - Perceptron: weights are motorized potentiometers
  - Adaline: Weights are electrochemical “memistors”

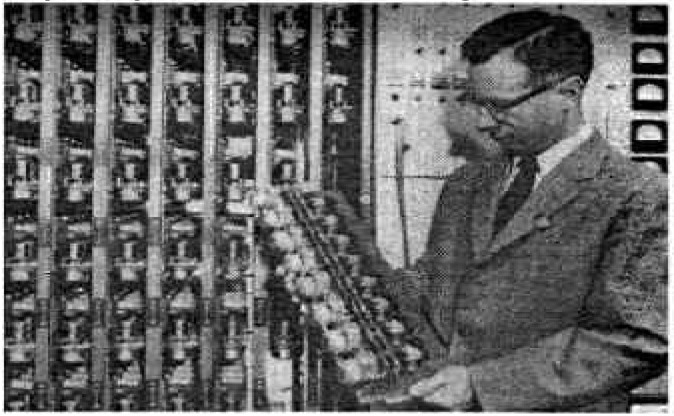
$$y = sign\left(\sum_{i=1}^{N} W_i X_i + b\right)$$



https://youtu.be/X1G2g3SiCwU

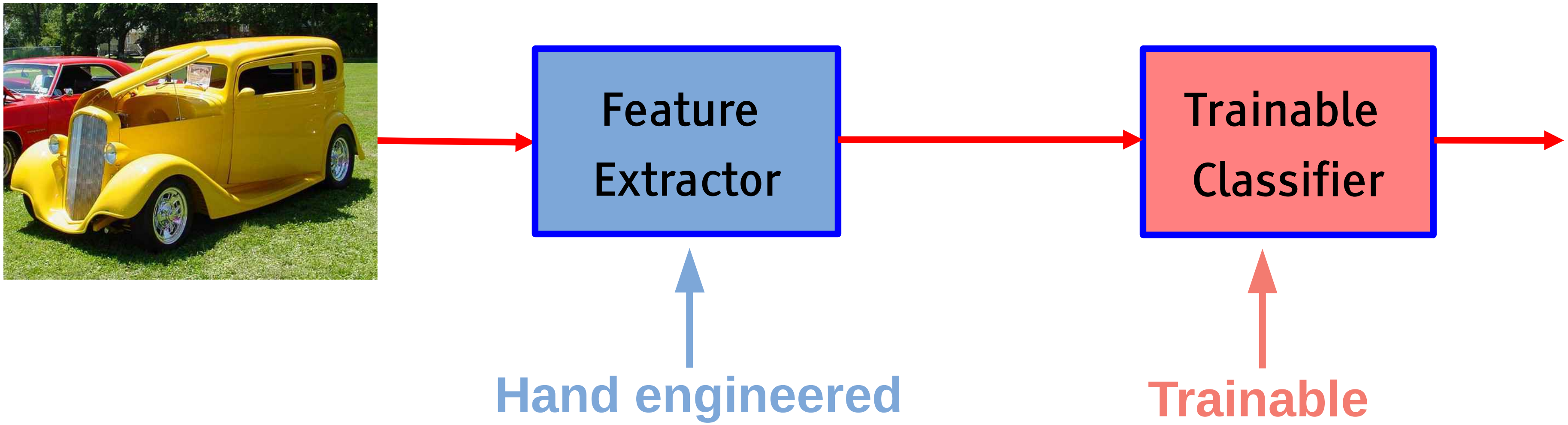






The Standard Paradigm of Pattern Recognition

- **...and “traditional” Machine Learning**



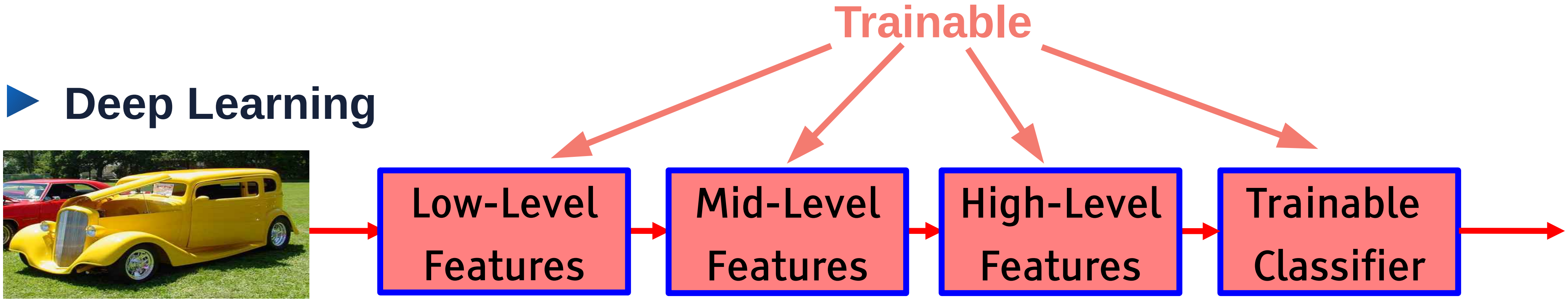
1969→1985: Neural Net Winter

- **No learning for multilayer nets, why?**
  - People used the wrong "neuron": the McCulloch & Pitts binary neuron
  - Binary neurons are easier to implement: No multiplication necessary!
  - Binary neurons prevented people from thinking about gradient-based methods for multi-layer nets
- **Early 1980s: The second wave of neural nets**
  - 1982: Hopfield nets: fully-connected recurrent binary networks
  - 1983: Boltzmann Machines: binary stochastic networks with hidden units
- **1985/86: Backprop! Q: Why only then? A: sigmoid neurons!**
  - Sigmoid neurons were enabled by "fast" floating point (Sun Workstations)

Multilayer Neural Nets and Deep Learning

- **Traditional Machine Learning**

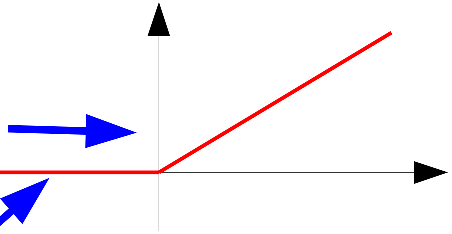
- **Deep Learning**



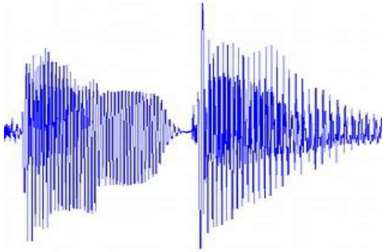
Multi-Layer Neural Nets

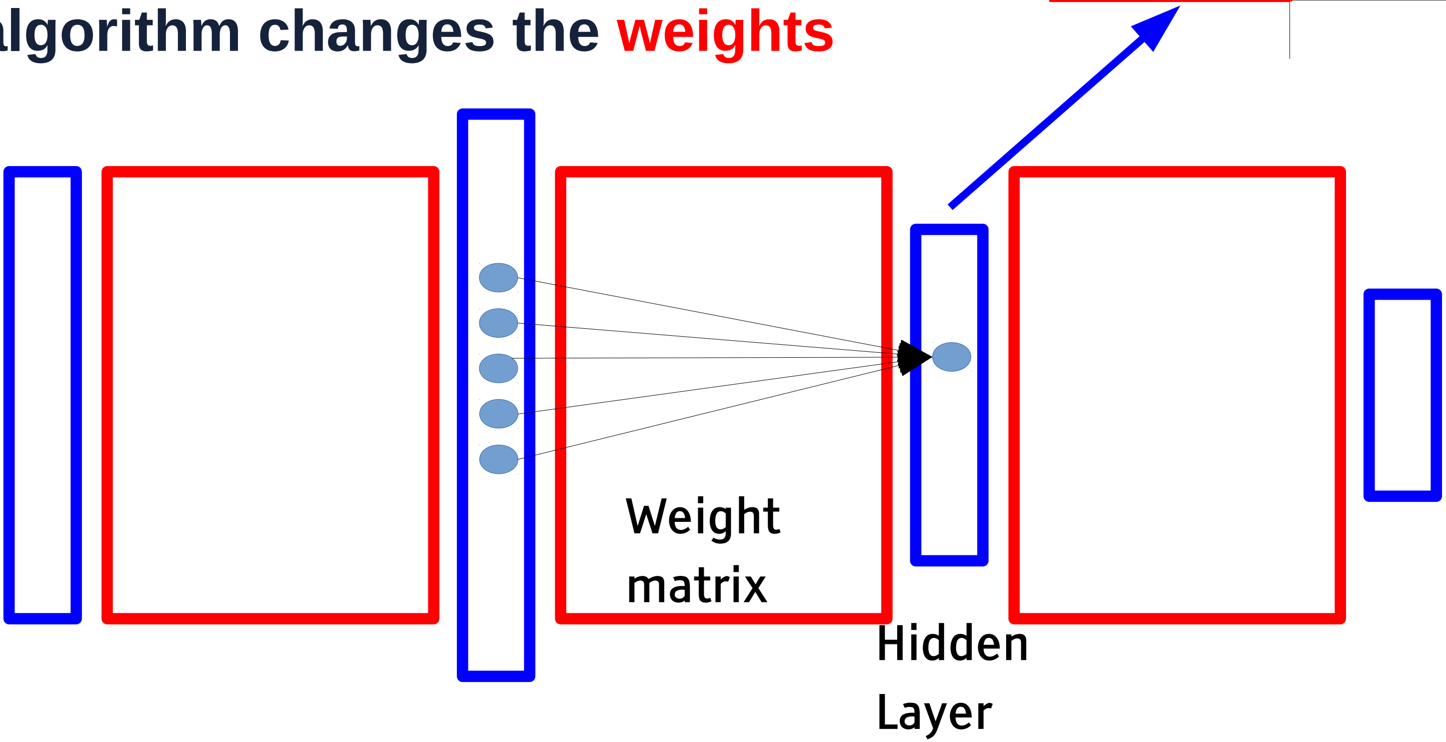
- Multiple Layers of **simple units**
- Each units computes a **weighted sum** of its inputs
- Weighted sum is passed through a **non-linear** function
- The learning algorithm changes the **weights**

$$ReLU(x) = max(x, 0)$$

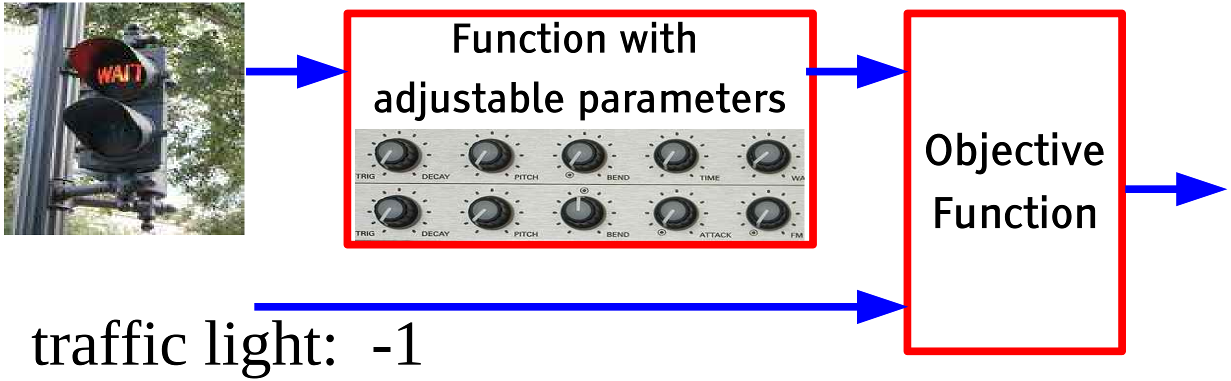


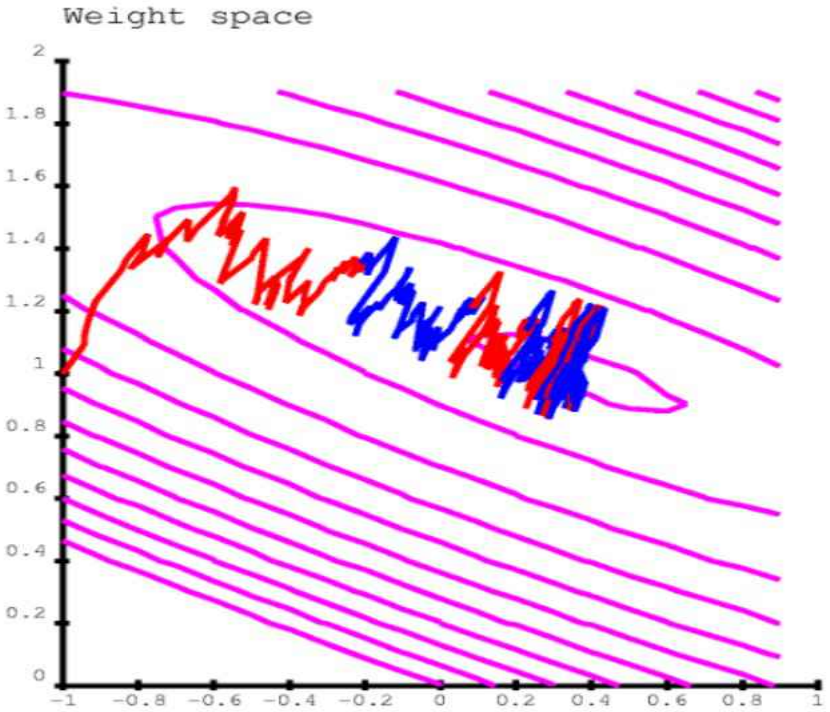
Ceci est une voiture





Supervised Machine Learning = Function Optimization



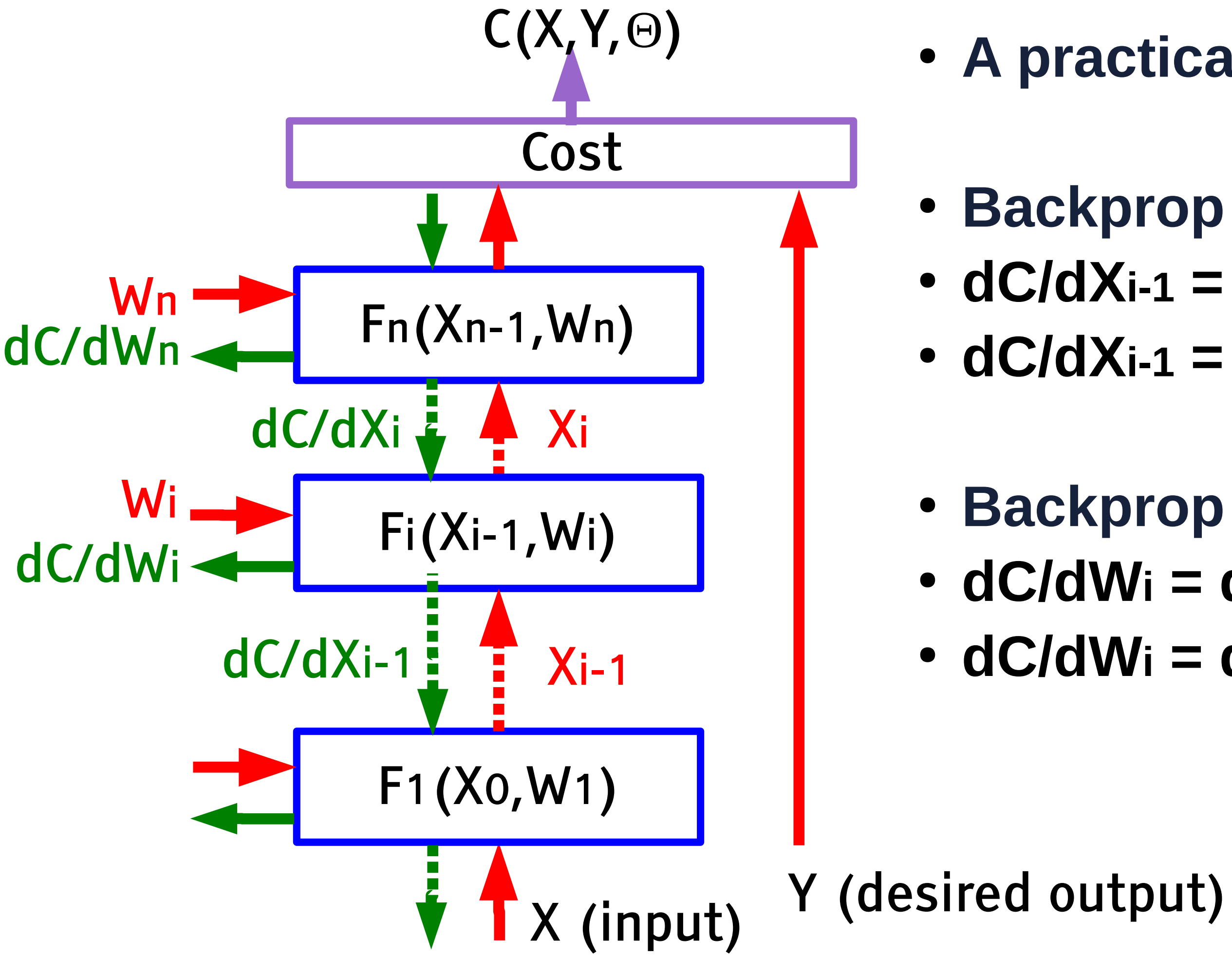


- **It's like walking in the mountains in a fog and following the direction of steepest descent to reach the village in the valley**
- **But each sample gives us a noisy estimate of the direction. So our path is a bit random.**

$$W_i \leftarrow W_i - \eta \frac{\partial L(W,X)}{\partial W_i}$$

- **Stochastic Gradient Descent (SGD)**

Computing Gradients by Back-Propagation



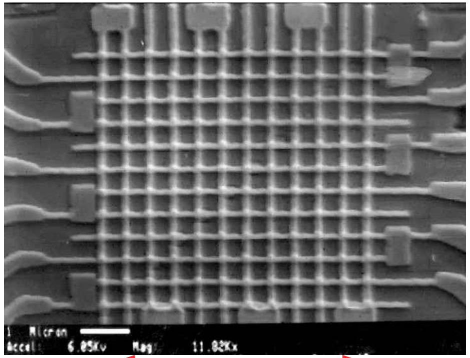
- **A practical Application of Chain Rule**

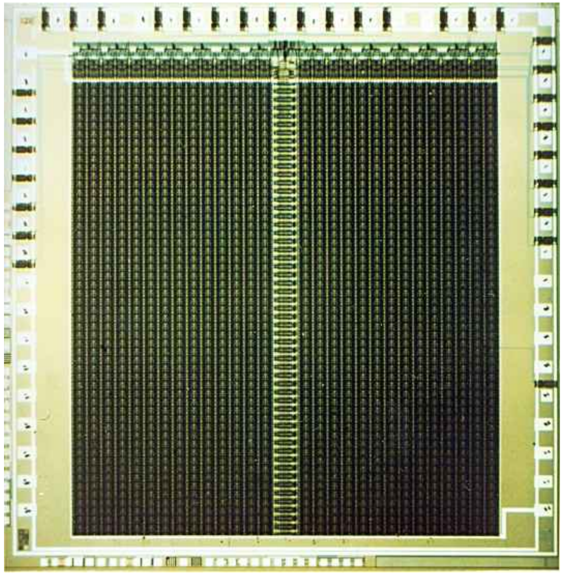
- **Backprop for the state gradients:**
- **$dC/dX_{i-1} = dC/dX_i \,.\, dX_i/dX_{i-1}$**
- **$dC/dX_{i-1} = dC/dX_i \,.\, dF_i(X_{i-1},W_i)/dX_{i-1}$**

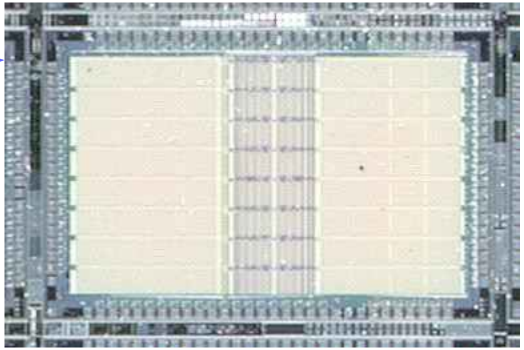
- **Backprop for the weight gradients:**
- **$dC/dW_i = dC/dX_i \,.\, dX_i/dW_i$**
- **$dC/dW_i = dC/dX_i \,.\, dF_i(X_{i-1},W_i)/dW_i$**

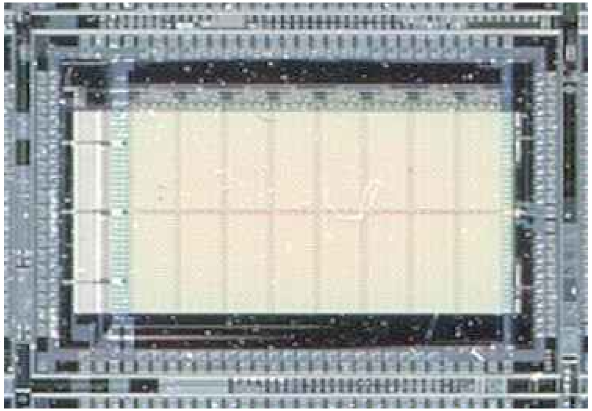
1986-1996 Neural Net Hardware at Bell Labs, Holmdel

- **1986: 12x12 resistor array**
  - Fixed resistor values
  - E-beam lithography: 6x6microns
- **1988: 54x54 neural net**
  - Programmable ternary weights
  - On-chip amplifiers and I/O
- **1991: Net32k: 256x128 net**
  - Programmable ternary weights
  - 320GOPS, 1-bit convolver.
- **1992: ANNA: 64x64 net**
  - ConvNet accelerator: 4GOPS
  - 6-bit weights, 3-bit activations

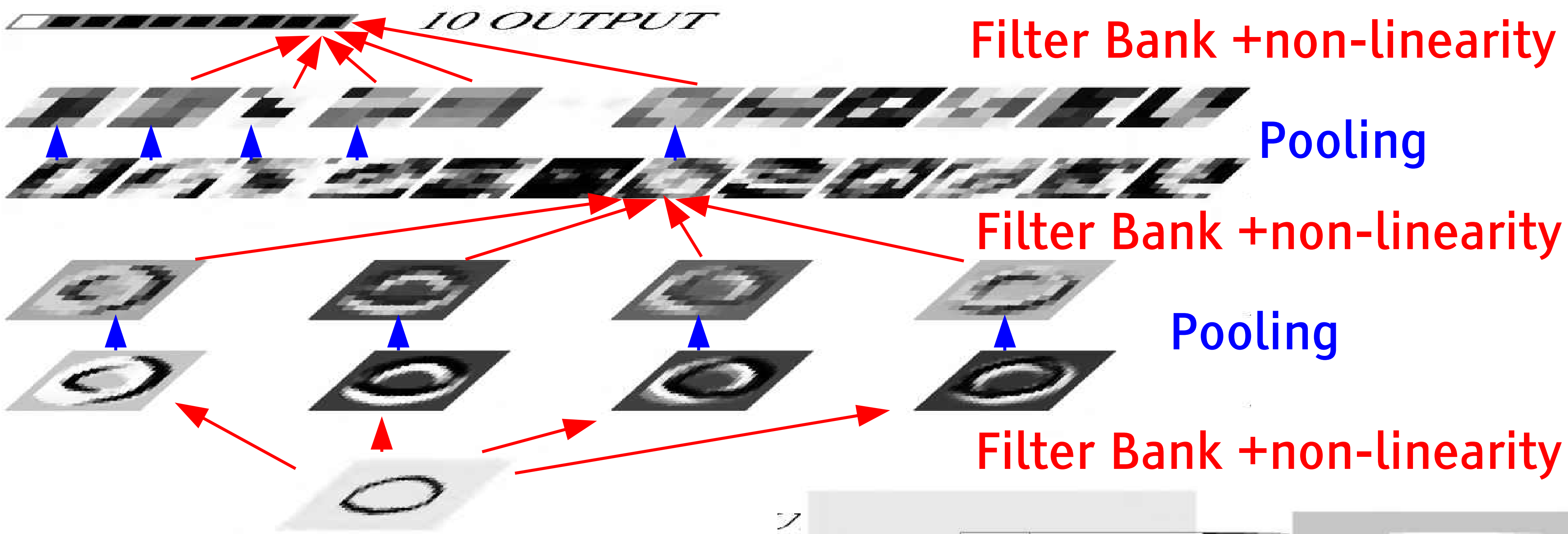




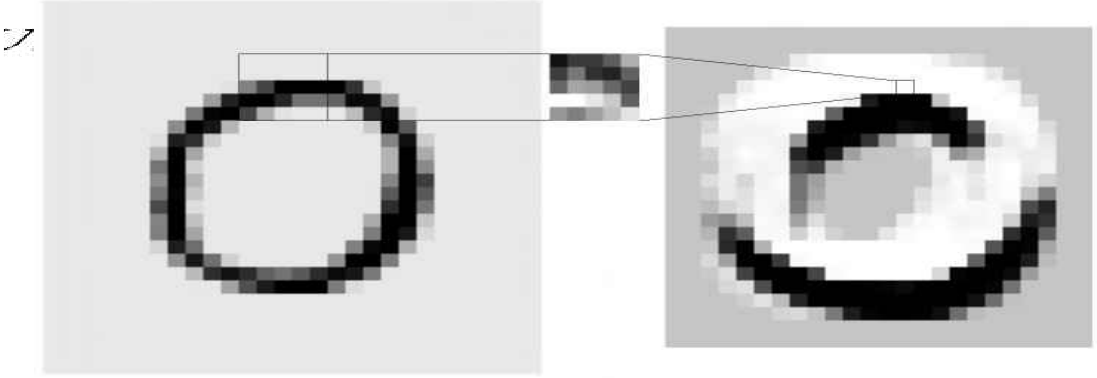




Convolutional Network Architecture [LeCun et al. NIPS 1989]



- **Inspired by [Hubel & Wiesel 1962] & [Fukushima 1982] (Neocognitron):**
  - simple cells detect local features
  - complex cells “pool” the outputs of simple cells within a retinotopic neighborhood.



LeNet character recognition demo 1992

- Running on an AT&T DSP32C (floating-point DSP, 20 MFLOPS)

Convolutional Network (LeNet5, vintage 1990)

- **Filters-tanh → pooling → filters-tanh → pooling → filters-tanh**

ConvNets can recognize multiple objects

- All layers are convolutional
- Networks performs simultaneous segmentation and recognition

Check Reader (AT&T 1995)

- **Check amount reader**
- **ConvNet+Language Model trained at the sequence level.**
- **50% percent correct, 49% reject, 1% error (detectable later in the process).**
- **Fielded in 1996, used in many banks in the US and Europe.**
- **Processed an estimated 10% to 20% of all the checks written in the US in the early 2000s.**
- **[LeCun, Bottou, Bengio ICASSP1997]**
  [LeCun, Bottou, Bengio, Haffner 1998]

1996 → 2006: 2nd NN Winter! Few teams could train large NNs

- **Hardware was slow for floating point computation**
  - Training a character recognizer took 2 weeks on a Sun or SGI workstation
  - A very small ConvNet by today's standard (500,000 connections)
- **Data was scarce and NN were data hungry**
  - No large datasets besides character and speech recognition
- **Interactive software tools had to be built from scratch**
  - We wrote a NN simulator with a custom Lisp interpreter/compiler
    - SN [Bottou & LeCun 1988] → SN2 [1992] → **Lush** (open sourced in 2002).
- **Open sourcing wasn't common in the pre-Internet days**
  - The "black art" of NN training could not be communicated easily
- **SN/SN2/Lush gave us superpowers: tools shape research directions**

Lessons learned #1

- ***1.1: It's hard to succeed with exotic hardware***
  - *Hardwired analog → programmable hybrid → digital*
- ***1.2: Hardware limitations influence research directions***
  - *It constrains what algorithm designers will let themselves imagine*
- ***1.3: Good software tools shape research and give superpowers***
  - *But require a significant investment*
  - *Common tools for Research and Development facilitates productization*
- ***1.4: Hardware performance matters***
  - *Fast turn-around is important for R&D*
  - *But high-end production models always take 2-3 weeks to train*
- ***1.5: When hardware is too slow, software is not readily available, or experiments are not easily reproducible, good ideas can be abandoned.***

The 2nd Neural Net Winter (1995-2005) & Spring (2006-2012)

The Lunatic Fringe and the Deep Learning Conspircy

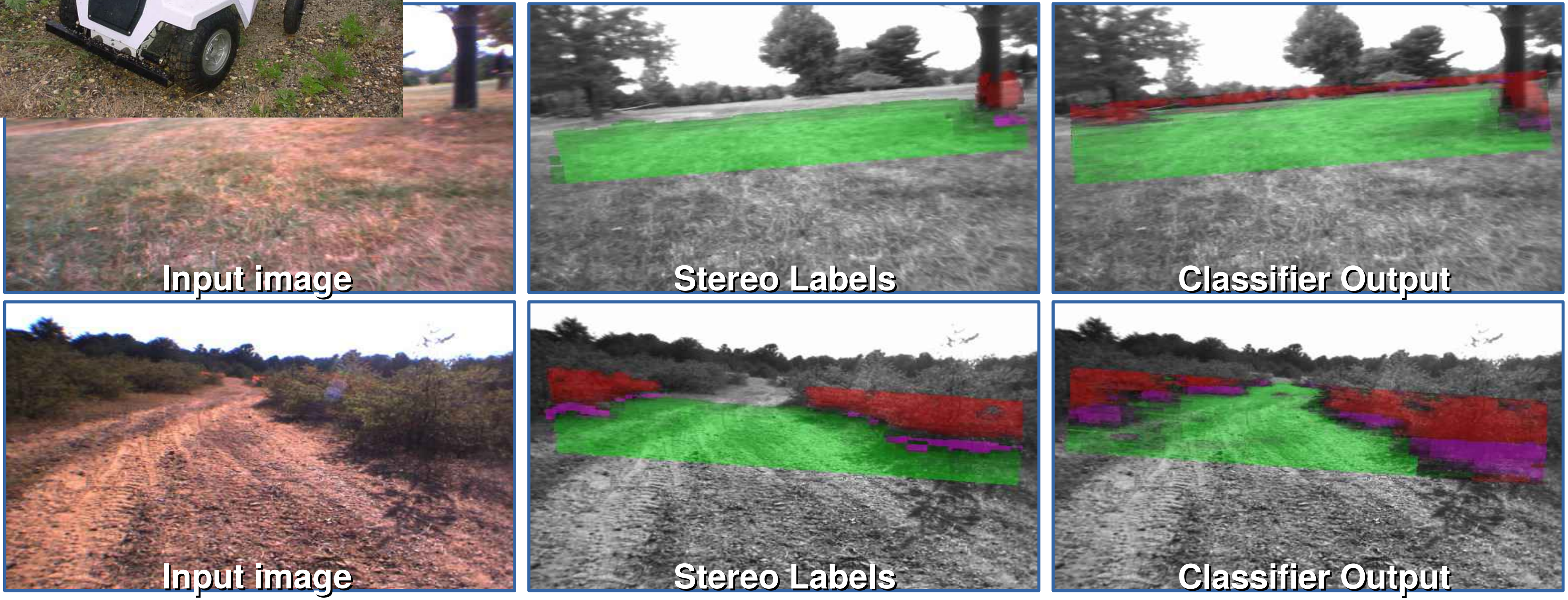
facebook
Artificial Intelligence Research

Semantic Segmentation with ConvNet for off-Road Driving

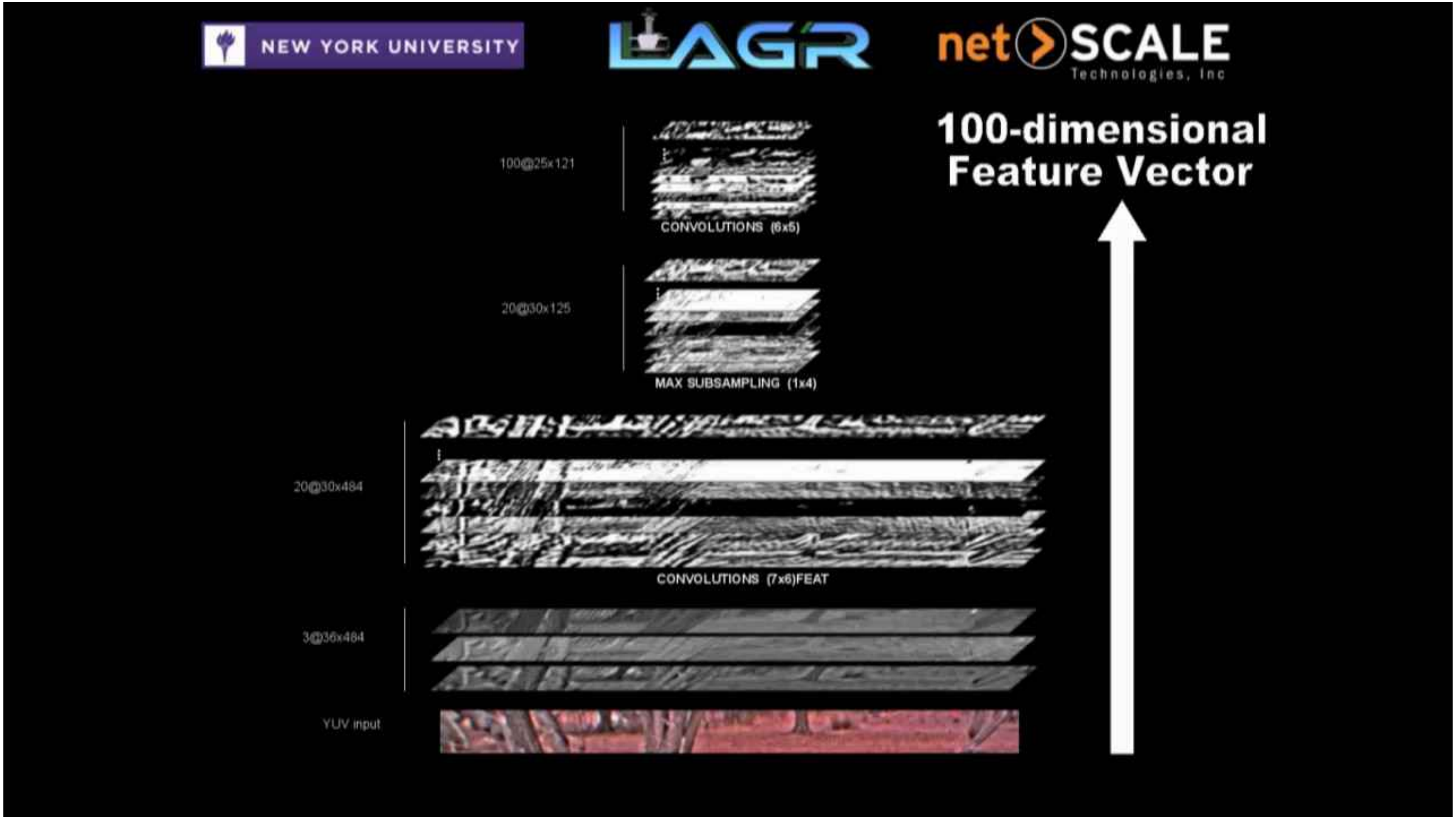
DARPA LAGR program 2005-2009

[Hadsell et al., J. of Field Robotics 2009]

[Sermanet et al., J. of Field Robotics 2009]



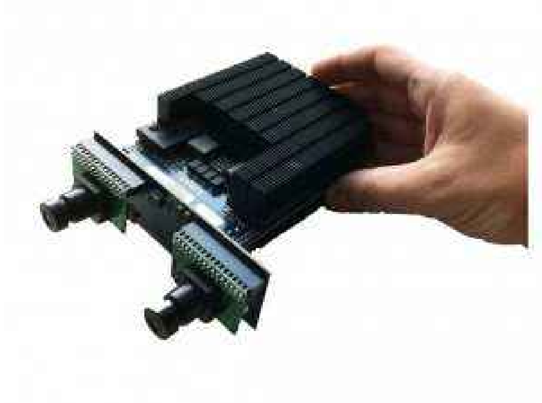
LAGR Video

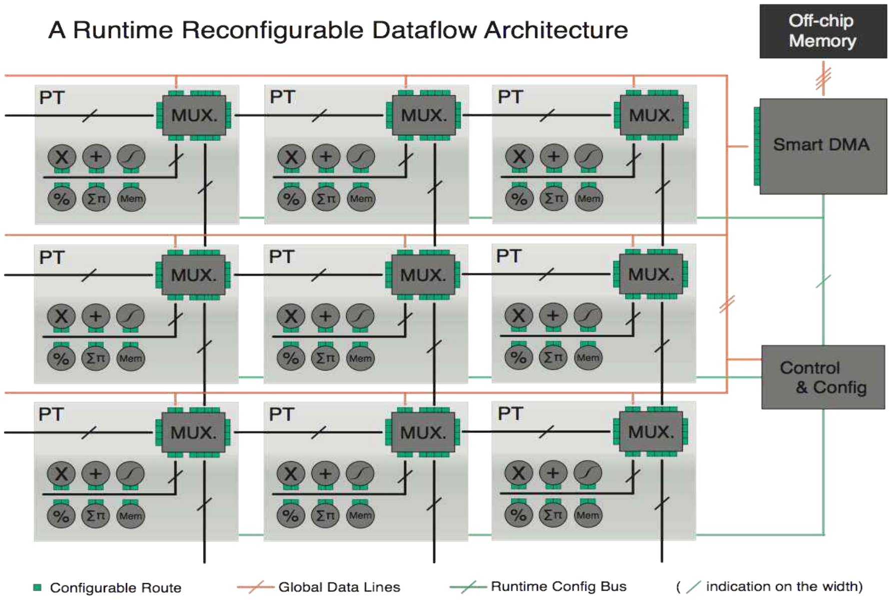

NEW YORK UNIVERSITY
LAGR
net SCALE
Technologies, Inc
100-dimensional
Feature Vector
100@25x121
CONVOLUTIONS (6x5)
20@30x125
MAX SUBSAMPLING (1x4)
20@30x484
CONVOLUTIONS (7x6)FEAT
3@36x484
YUV input

Semantic Segmentation with ConvNets (33 categories)

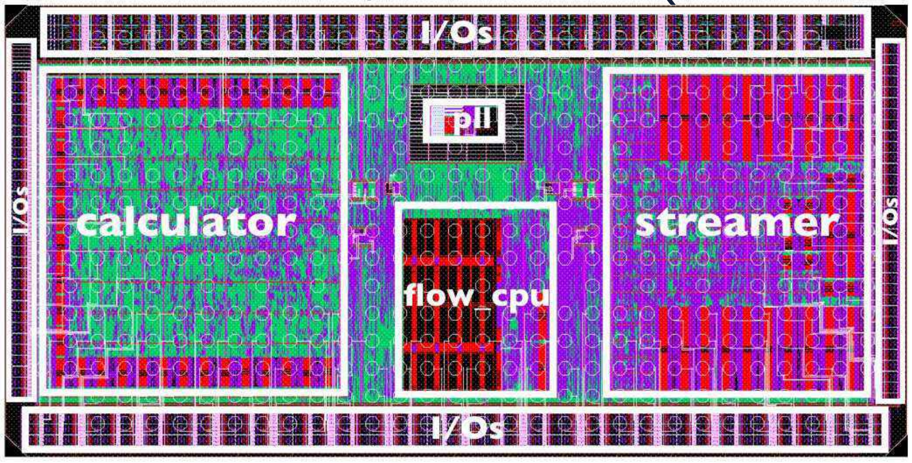
FPGA ConvNet Accelerator: NewFlow [Farabet 2011]

- **NeuFlow: Reconfigurable Dataflow architecture**
  - Implemented on Xilinx Virtex6 FPGA
  - 20 configurable tiles. 150GOPS, 10 Watts
  - Semantic Segmentation: 20 frames/sec at 320x240
  - **Exploits the structure of convolutions**



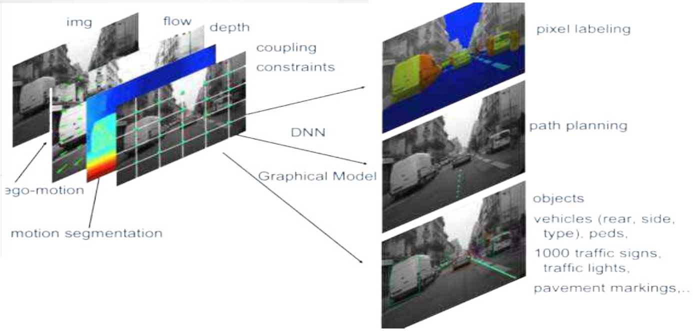


- **NeuFlow ASIC** [Pham 2012]
  - 150GOPS, 0.5 Watts (simulated)

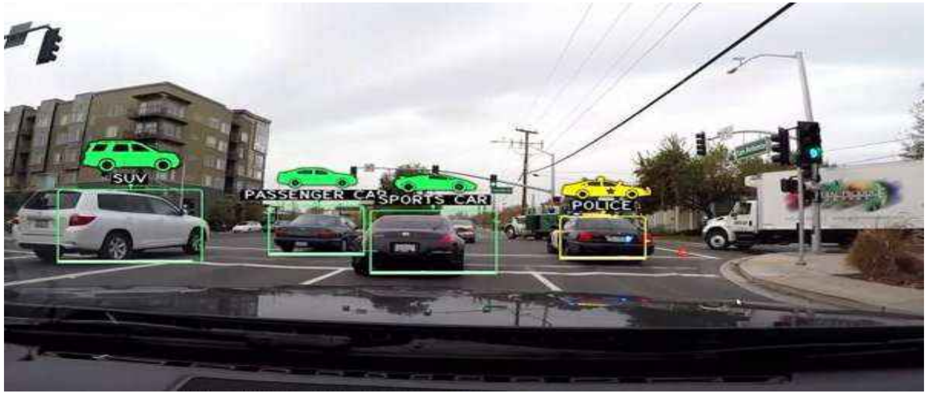


Driving Cars with Convolutional Nets

- **MobilEye**



- **NVIDIA**



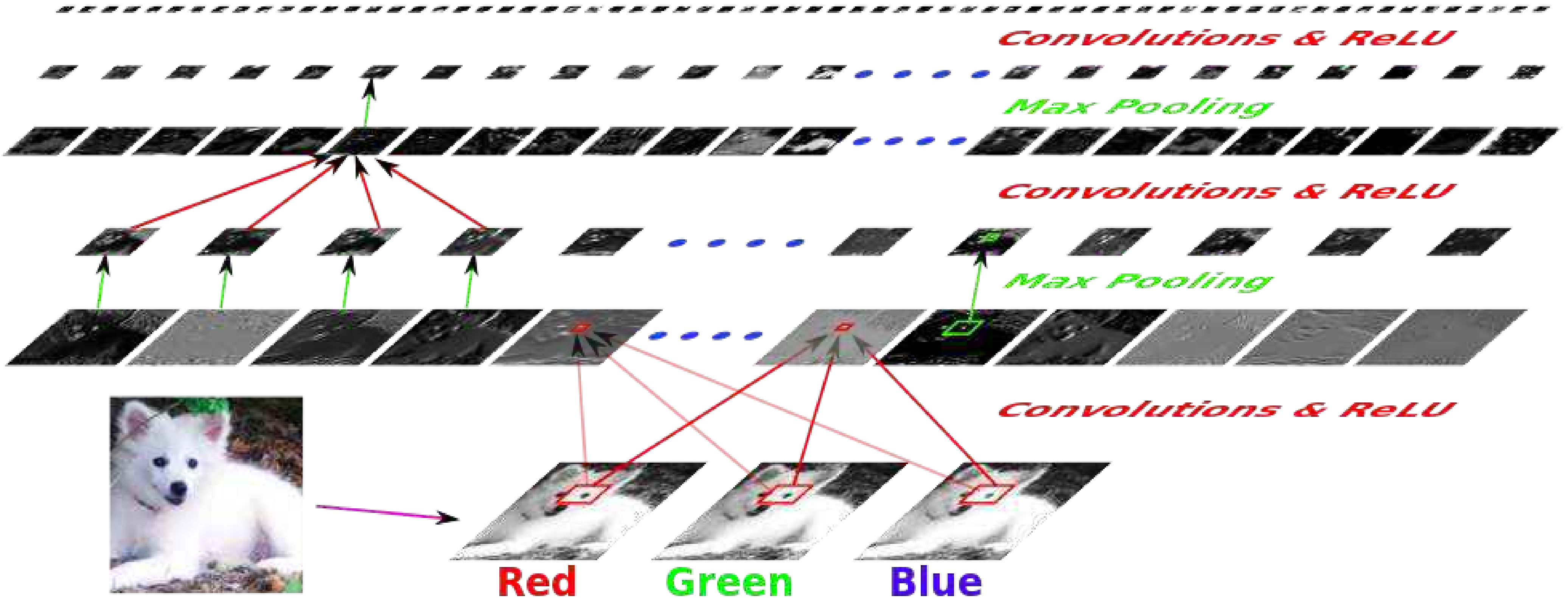
The Deep Learning Revolution

State of the Art

facebook
Artificial Intelligence Research

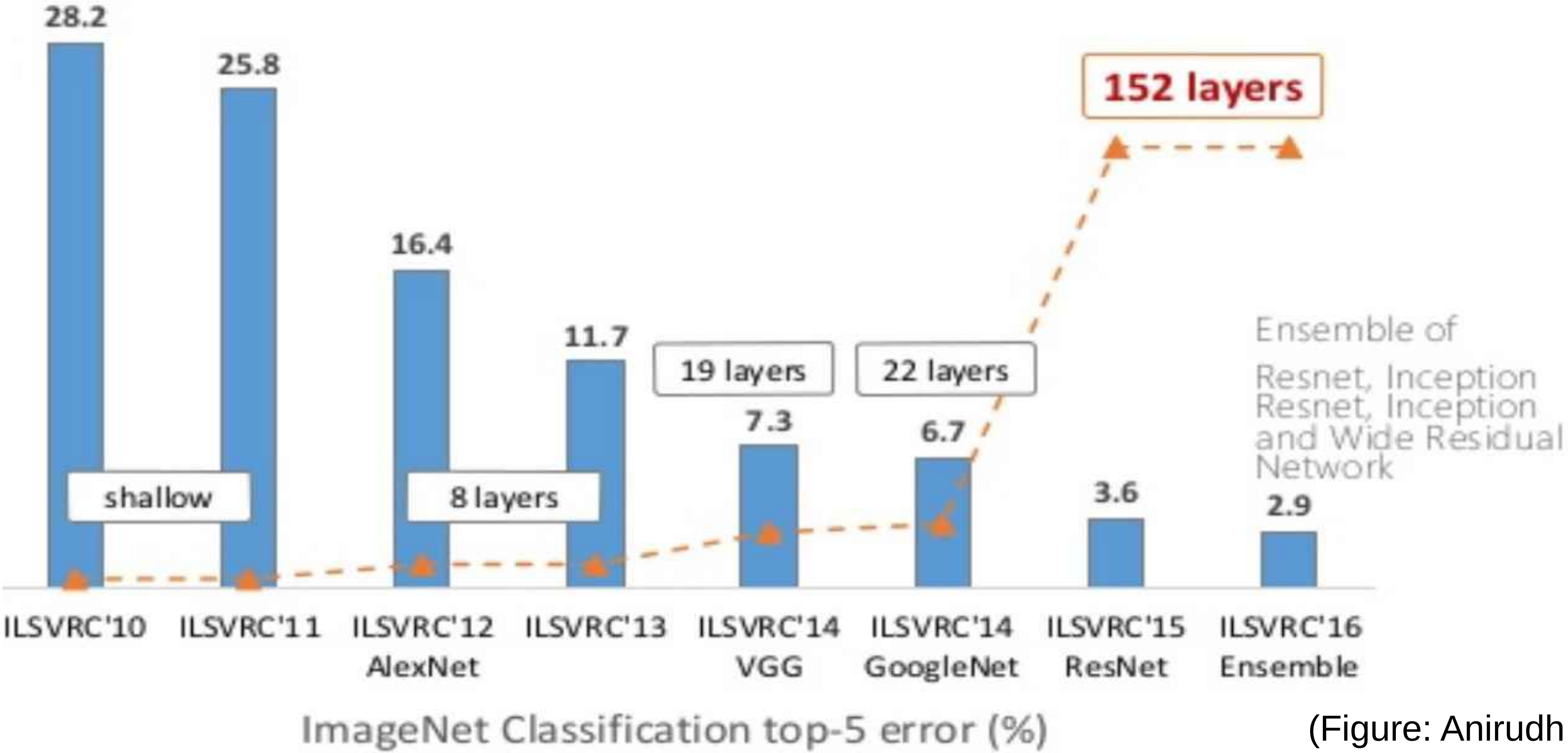
Deep ConvNets for Object Recognition (on GPU)

- AlexNet [Krizhevsky et al. NIPS 2012], OverFeat [Sermanet et al. 2013]
- 1 to 10 billion connections, 10 million to 1 billion parameters, 8 to 20 layers.



Error Rate on ImageNet

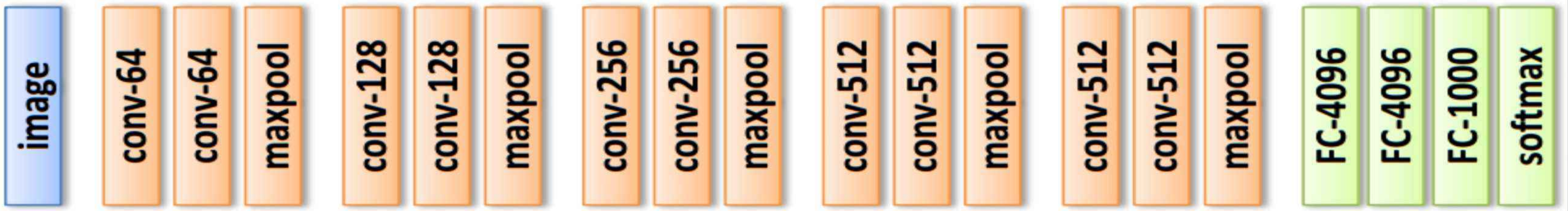
- **Depth inflation**



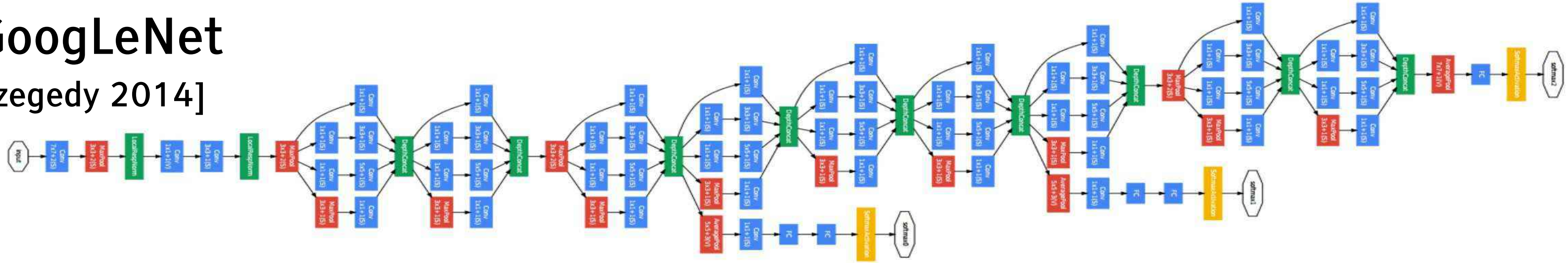
(Figure: Anirudh Koul)

Deep ConvNets (depth inflation)

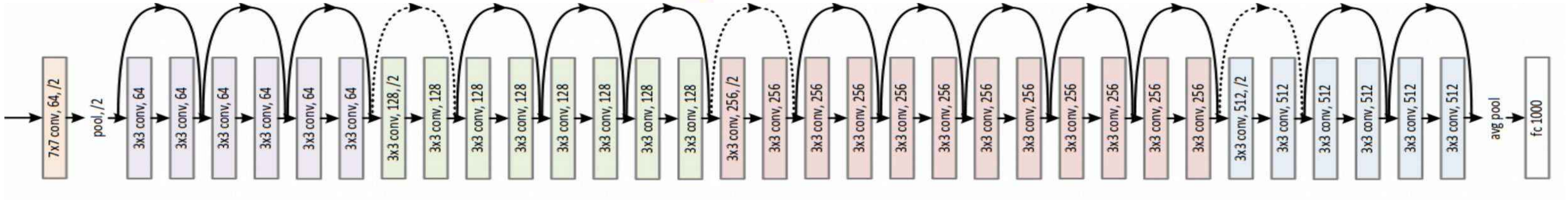
VGG
[Simonyan 2013]



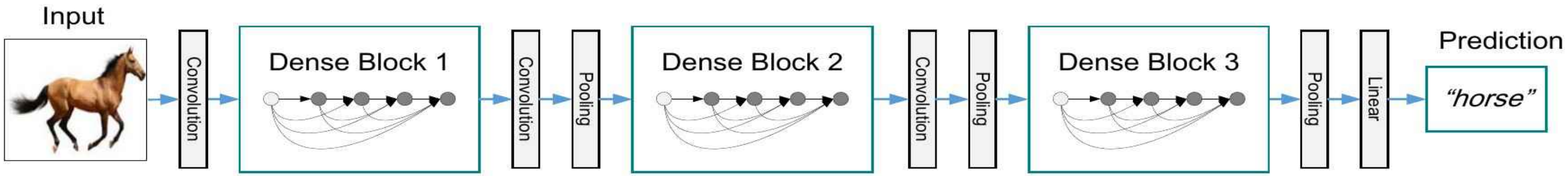
GoogLeNet
Szegedy 2014]



ResNet
[He et al. 2015]

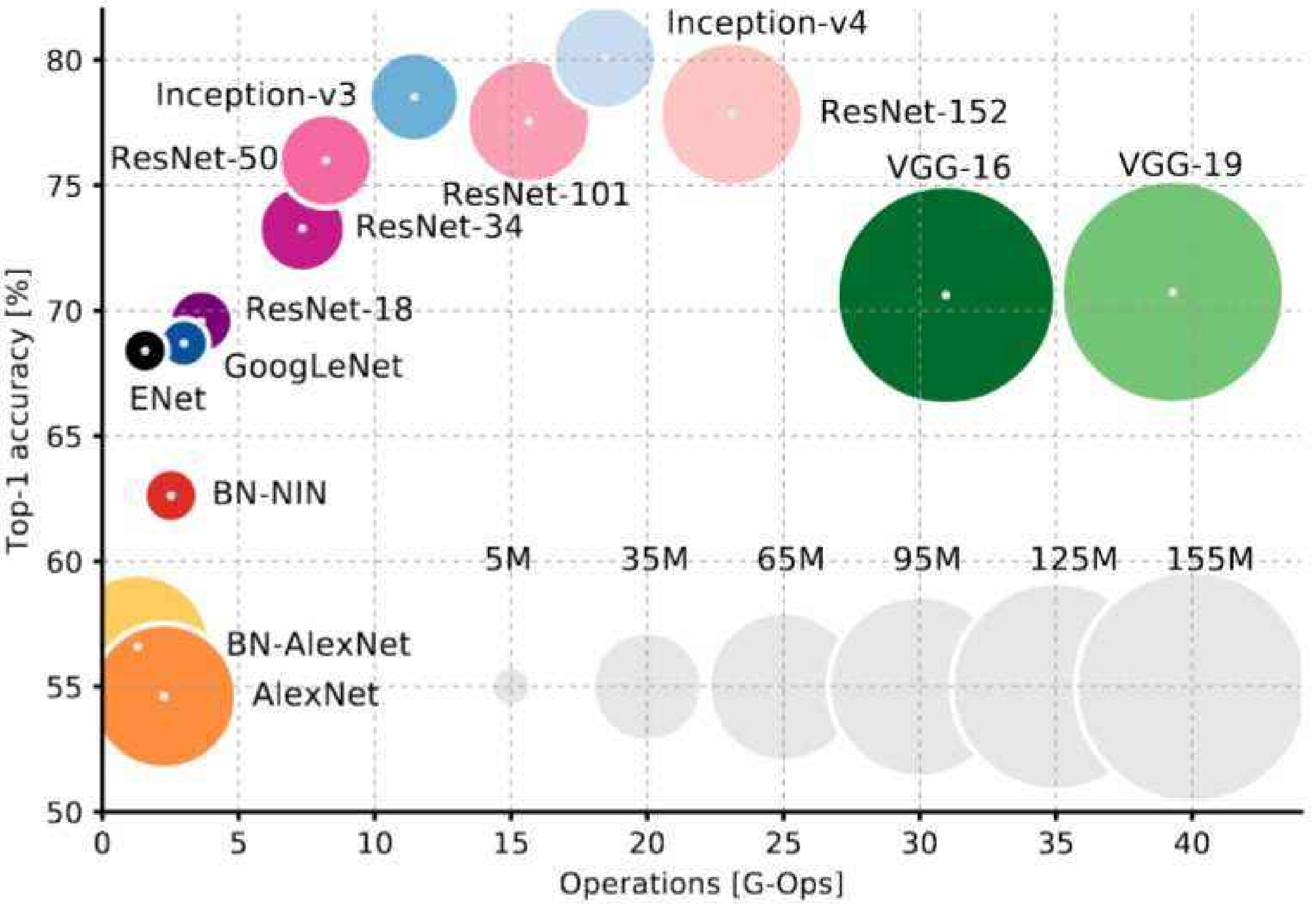


DenseNet
[Huang et al 2017]



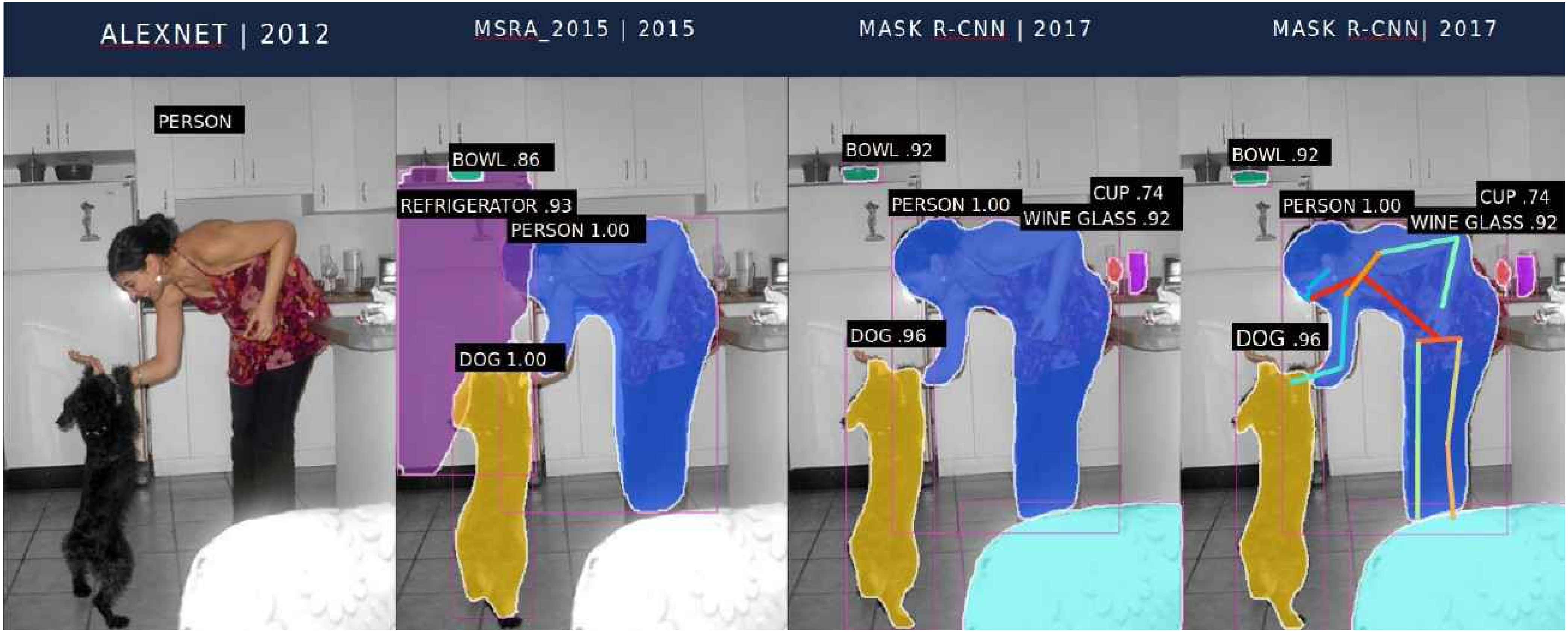
GOPS vs Accuracy on ImageNet vs #Parameters

- [Canziani 2016]
- ResNet50 and ResNet100 are used routinely in production.
- Each of the few billions photos uploaded on Facebook every day goes through a handful of ConvNets within 2 seconds.



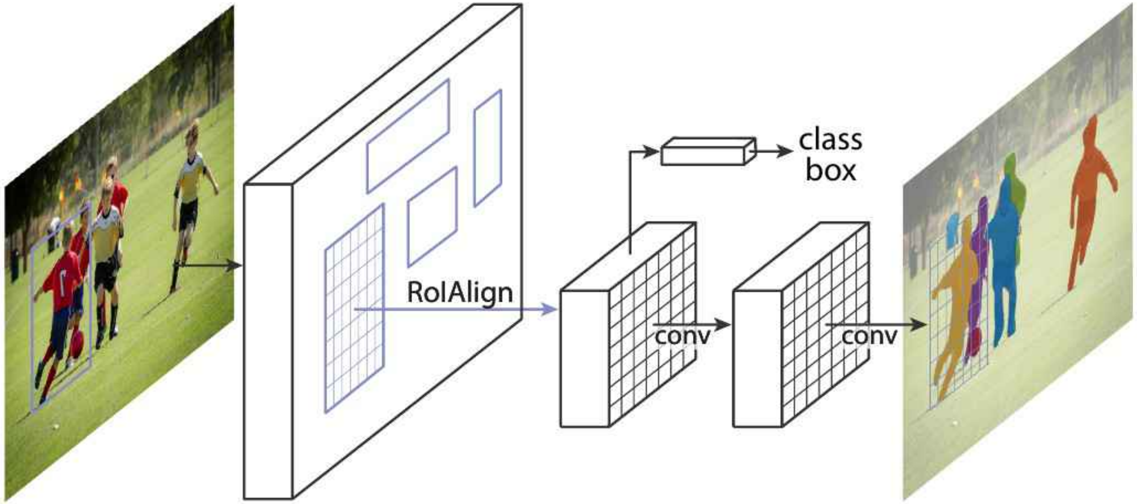
Progress in Computer Vision

- **[He 2017]**



Mask R-CNN: instance segmentation

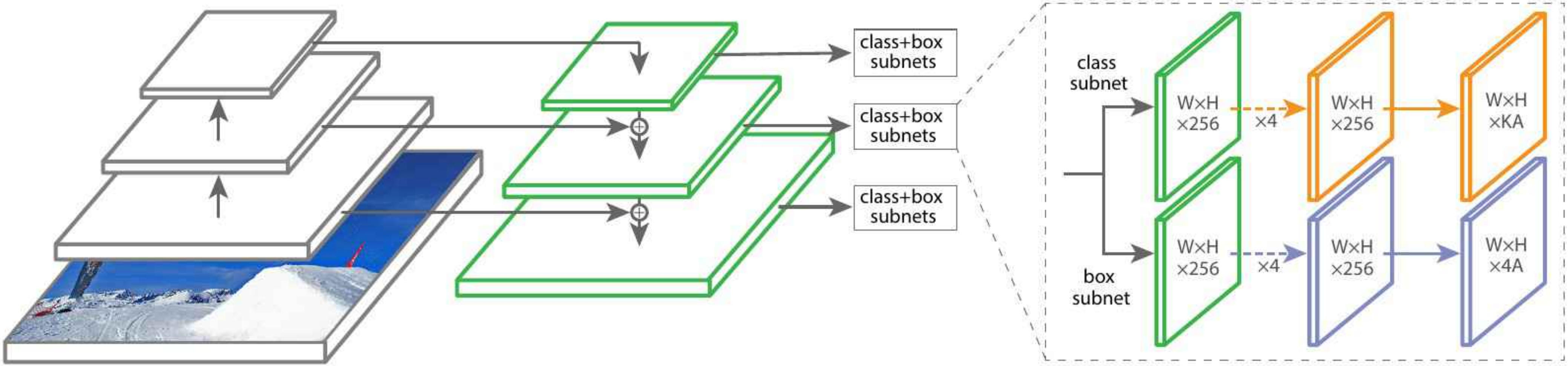
- [He, Gkioxari, Dollar, Girshick arXiv:1703.06870]
- ConvNet produces an object mask for each region of interest
- Combined ventral and dorsal pathways



	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [7]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [20] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [20] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

RetinaNet, feature pyramid network

- **One-pass object detection**
- **[Lin et al. ArXiv:1708.02002]**



(a) ResNet (b) feature pyramid net (c) class subnet (top) (d) box subnet (bottom)

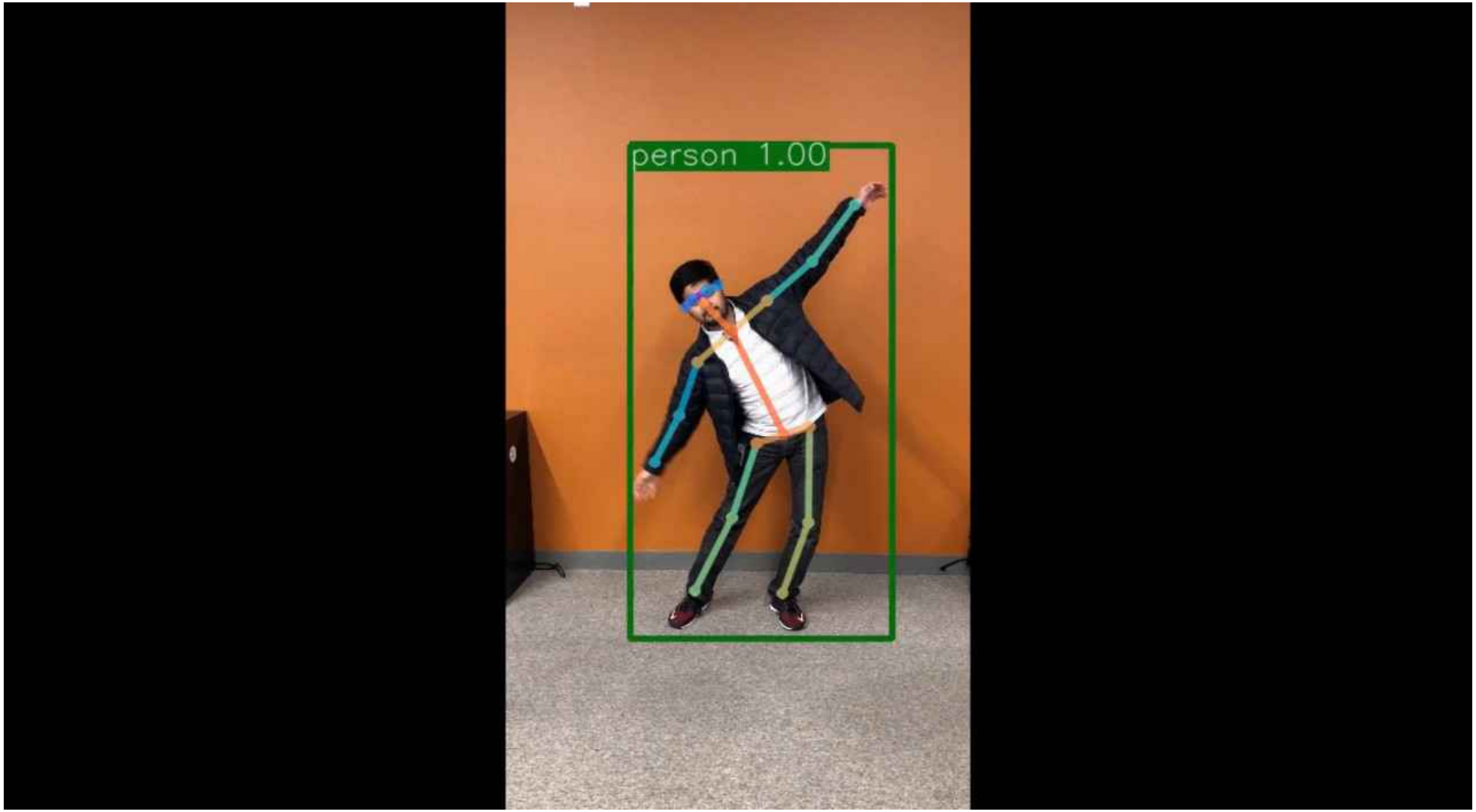
Mask-RCNN Results on COCO dataset

- **Individual objects are segmented.**

Mask R-CNN Results on COCO test set

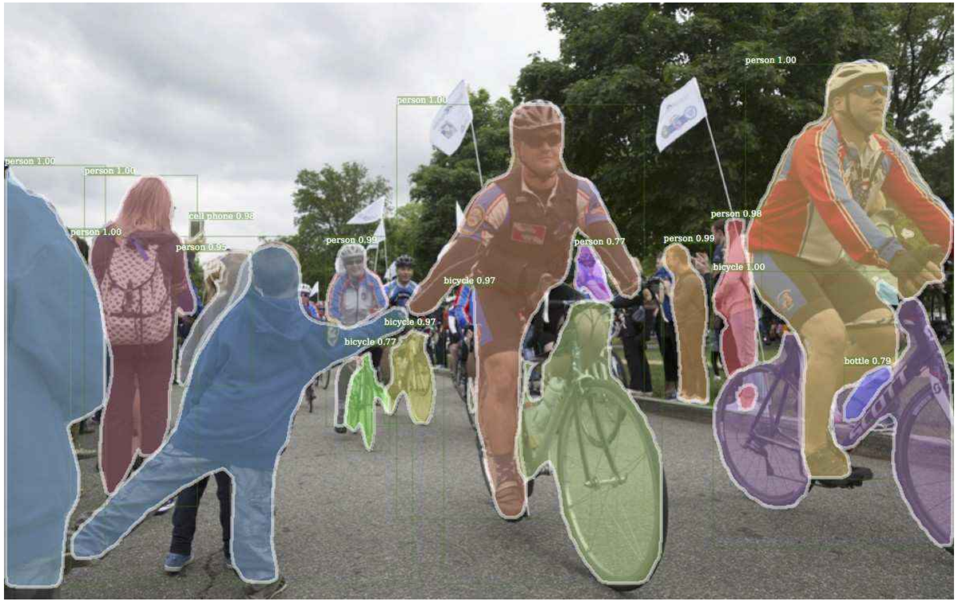
Real-Time Pose Estimation on Mobile Devices

- **Maks R-CNN running on Caffe2Go**



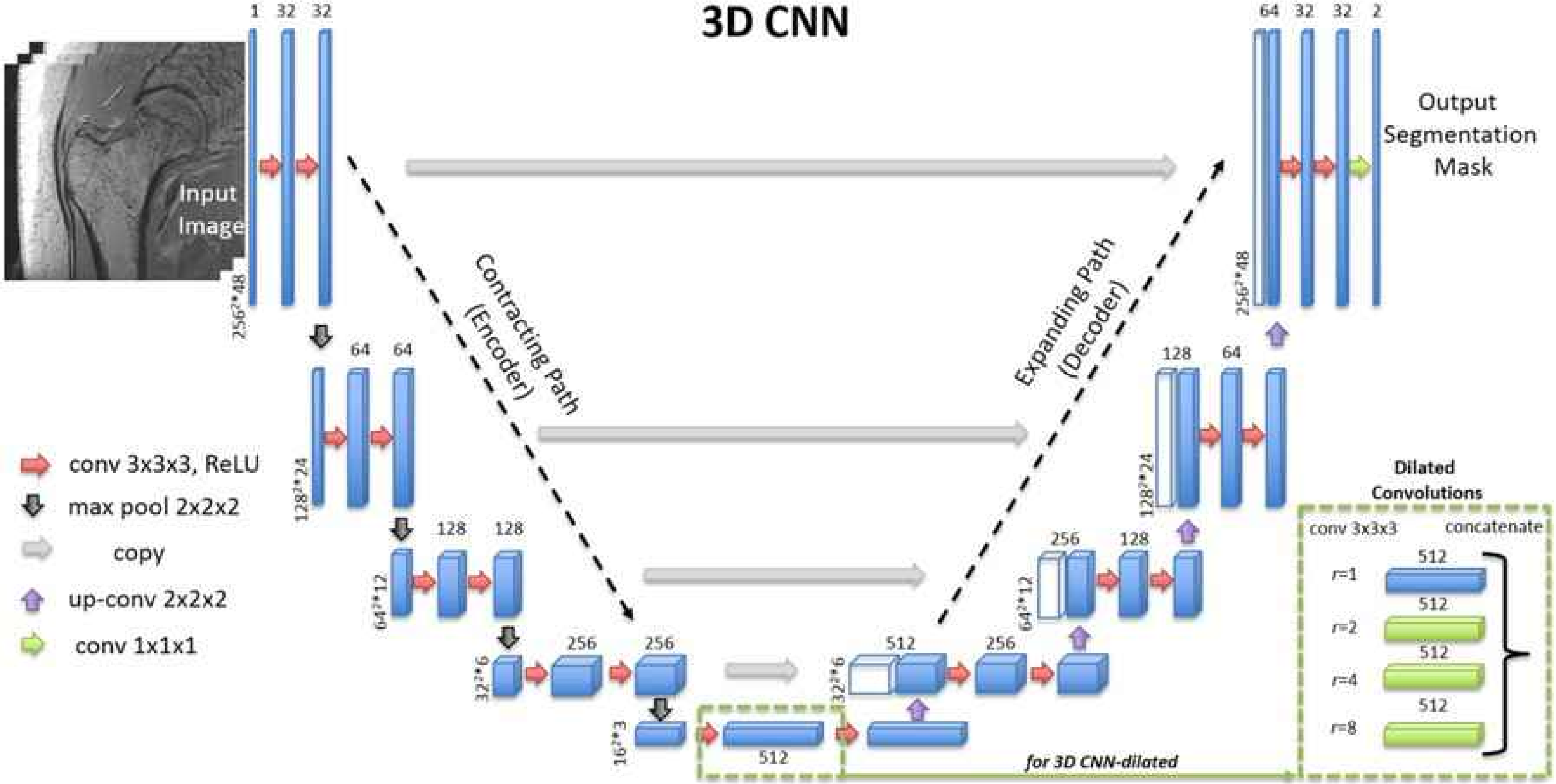
Detectron: open source vision in PyTorch

https://github.com/facebookresearch/maskrcnn-benchmark

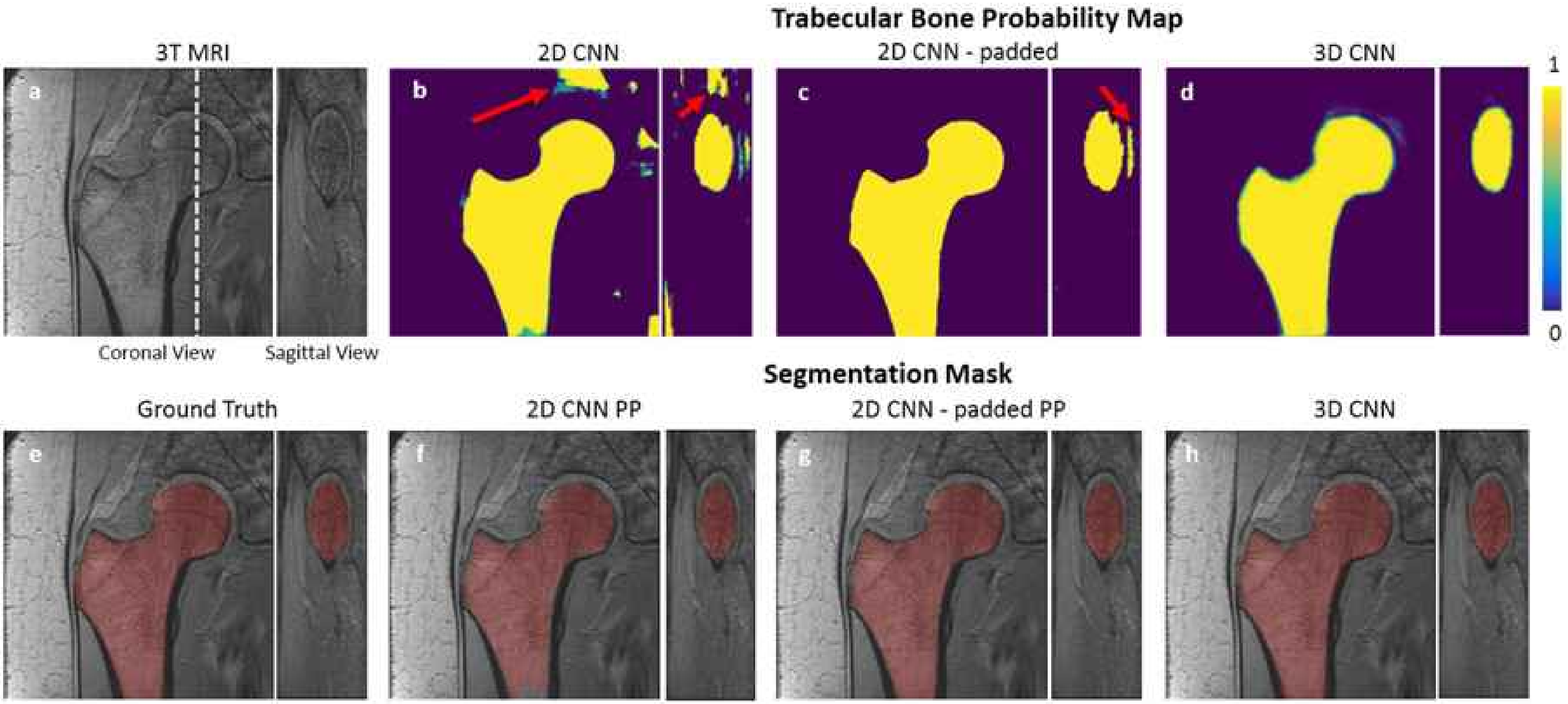


3D ConvNet for Medical Image Analysis

- **Segmentation Femur from MR Images**
- **[Deniz et al. Nature 2018]**

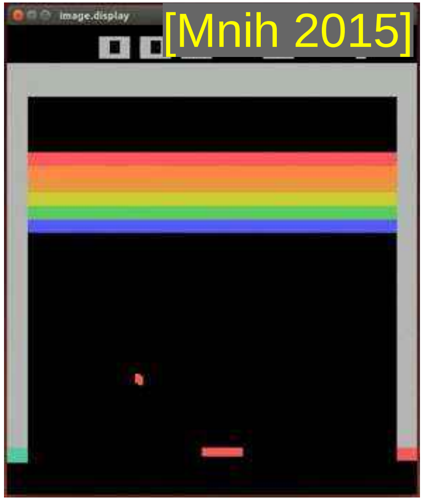


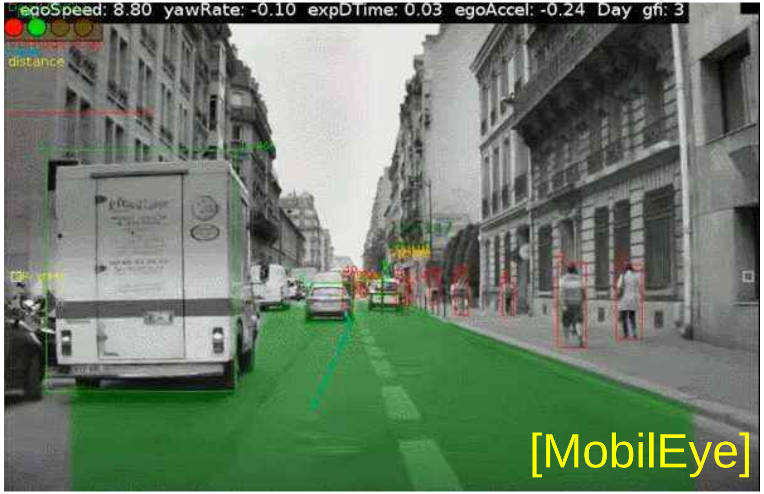
3D ConvNet for Medical Image Analysis

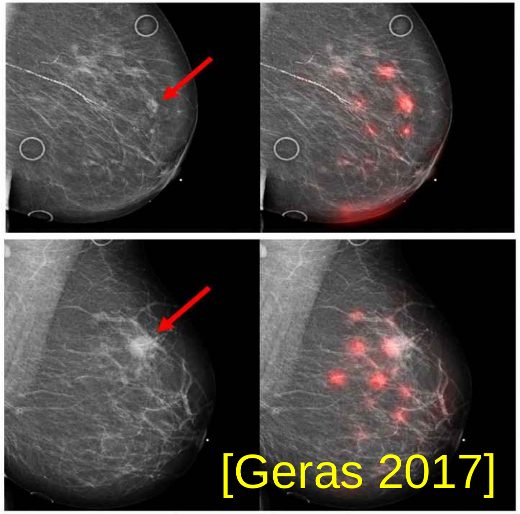


Applications of Deep Learning

- **Medical image analysis**
- **Self-driving cars**
- **Accessibility**
- **Face recognition**
- **Language translation**
- **Virtual assistants***
- **Content Understanding for:**
  - Filtering
  - Selection/ranking
  - Search
- **Games**
- **Security, anomaly detection**
- **Diagnosis, prediction**
- **Science!**







Lessons learned #2

- ***2.1: Good results are not enough***
  - *Making them easily reproducible also makes them credible.*
- ***2.2: Hardware progress enables new breakthroughs***
  - *General-Purpose GPUs should have come 10 years earlier!*
  - *But can we please have hardware that doesn't require batching?*
- ***2.3: Open-source software platforms disseminate ideas***
  - *But making platforms that are good for research and production is hard.*
- ***2.4: Convolutional Nets will soon be everywhere***
  - *Hardware should exploit the properties of convolutions better*
  - *There is a need for low-cost, low-power ConvNet accelerators*
  - *Cars, cameras, vacuum cleaners, lawn mowers, toys, maintenance robots...*

New DL Architectures

With different hardware/software requirements:

Memory-Augmented Networks

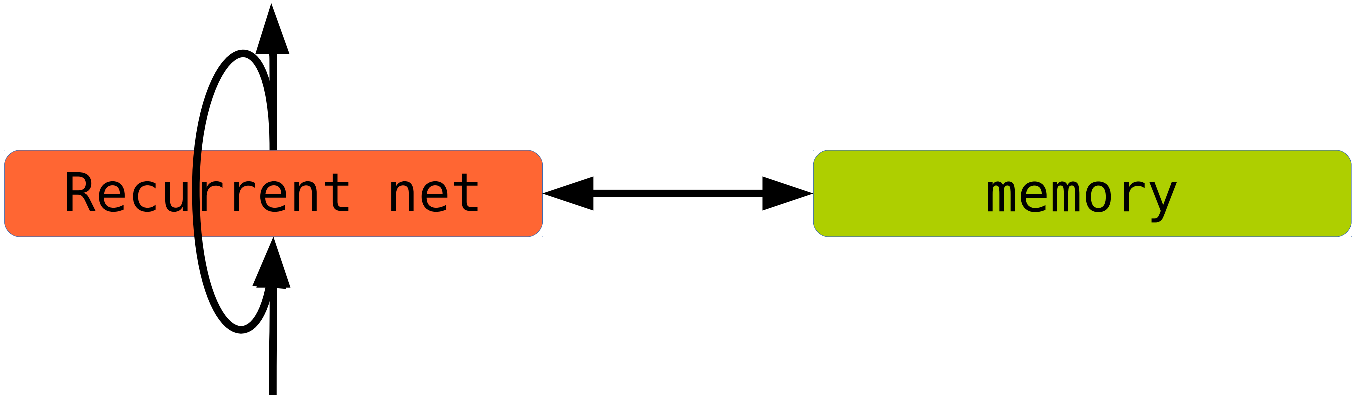
Dynamic Networks

Graph Convolutional Nets

Networks with Sparse Activations

Augmenting Neural Nets with a Memory Module

- **Recurrent networks cannot remember things for very long**
  - The cortex only remember things for 20 seconds
- **We need a "hippocampus" (a separate memory module)**
  - LSTM [Hochreiter 1997], registers
  - **Memory networks** [Weston et 2014] (FAIR), associative memory
  - **Stacked-Augmented Recurrent Neural Net** [Joulin & Mikolov 2014] (FAIR)
  - **Neural Turing Machine** [Graves 2014],
  - **Differentiable Neural Computer** [Graves 2016]

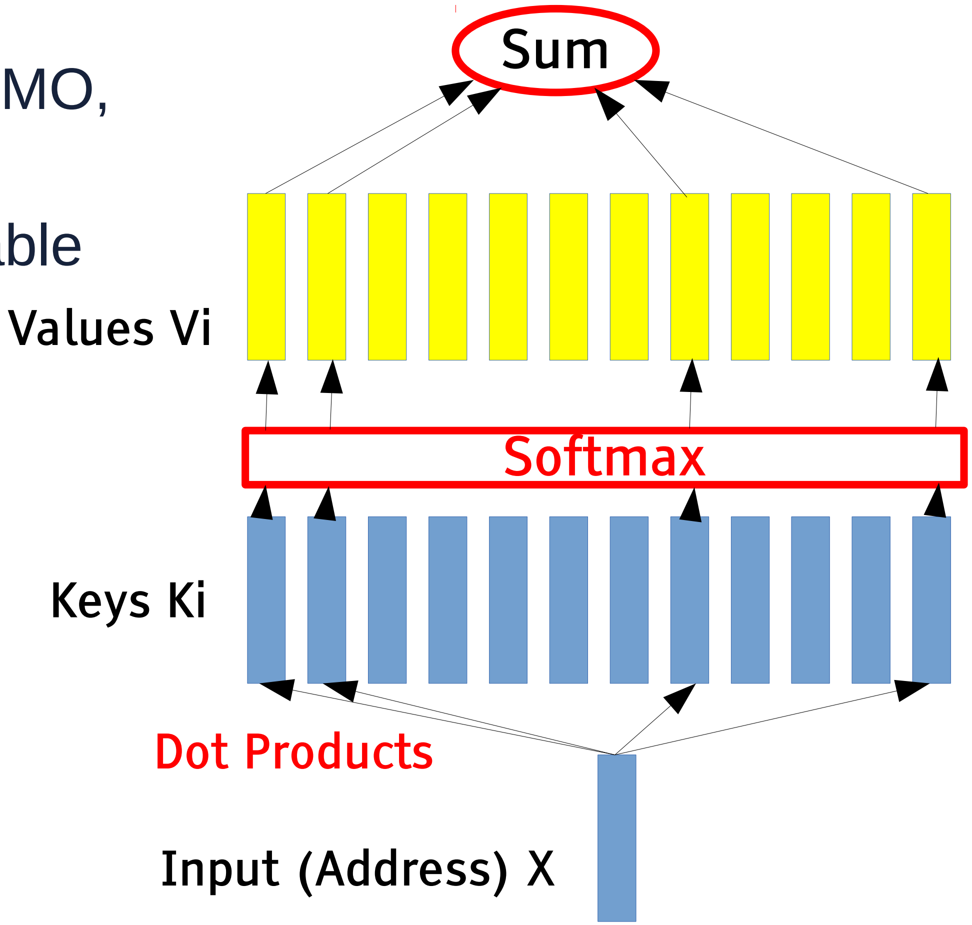


Differentiable Associative Memory

- Used very widely in NLP
- MemNN, Transformer Network, ELMO, GPT, BERT, GPT2, GLoMO
- Essentially a “soft” RAM or hash table

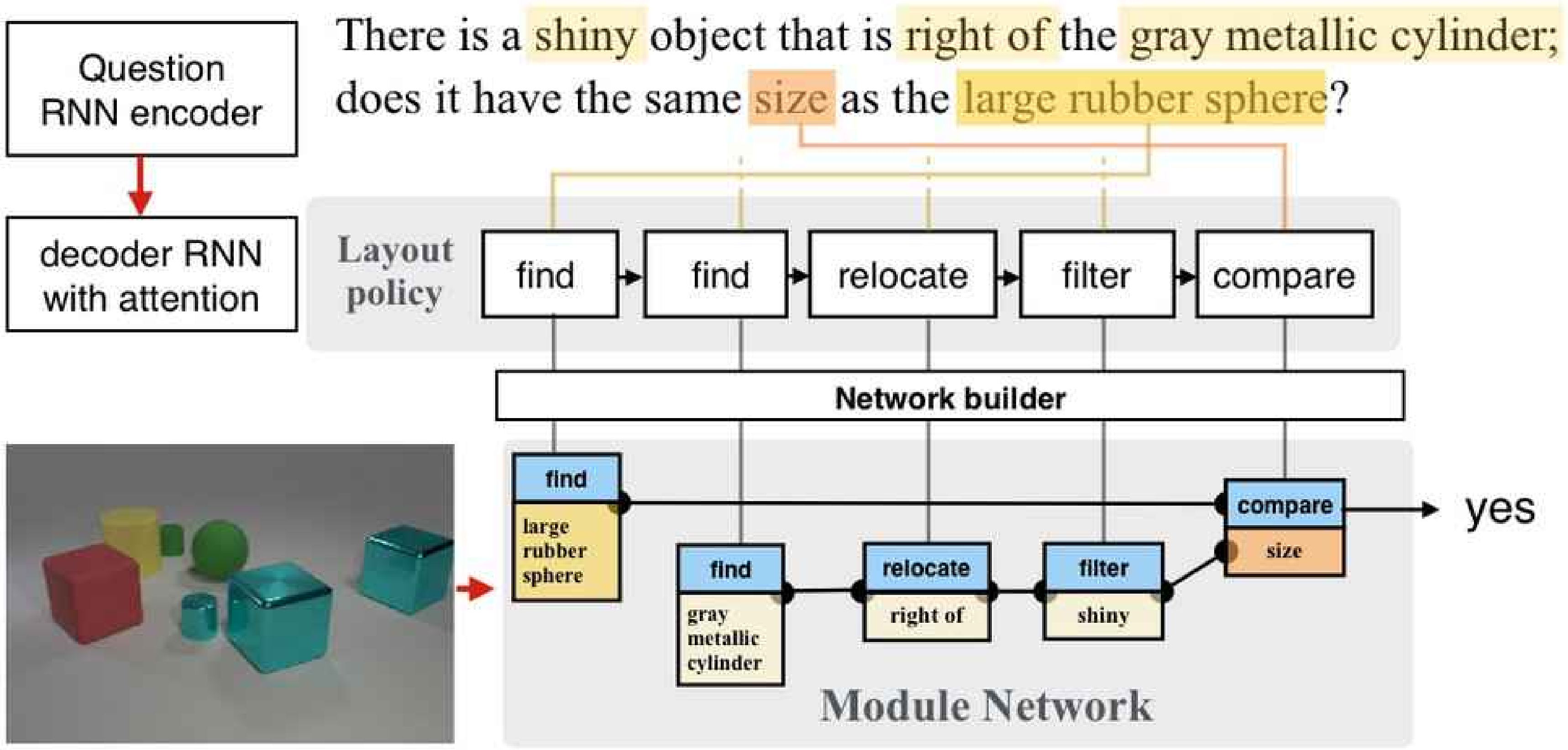
$$C_i = \frac{e^{K_i^T X}}{\sum_j e^{K_j^T X}}$$

$$Y = \sum_i C_i V_i$$



Learning to synthesize neural programs for visual reasoning

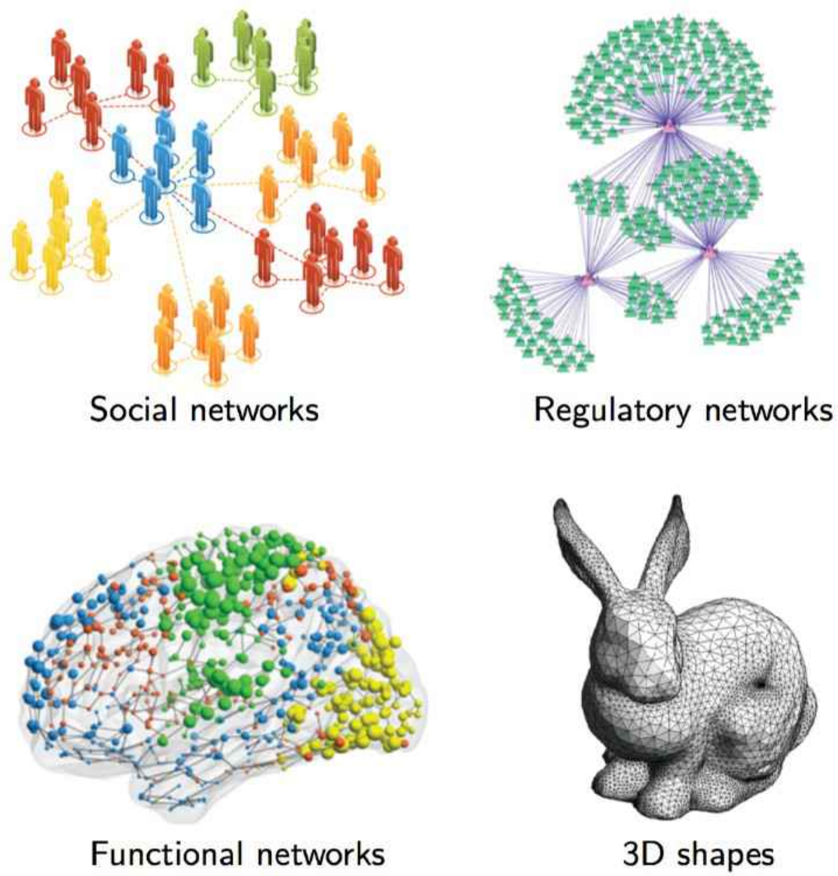
https://research.fb.com/visual-reasoning-and-dialog-towards-natural-language-conversations-about-visual-data/



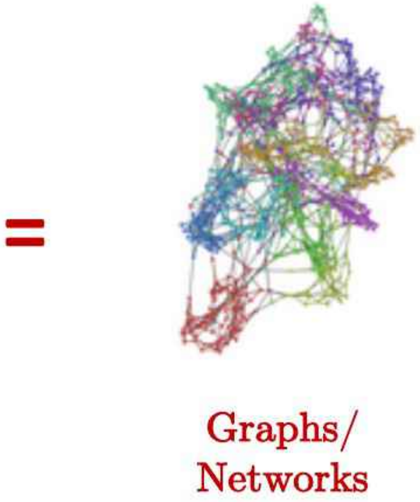
PyTorch: differentiable programming

- **Software 2.0:**
  - The operations in a program are only partially specified
  - They are trainable parameterized modules.
  - The precise operations are learned from data, only the general structure of the program is designed.
- **Dynamic computational graph**
  - Automatic differentiation by recording a “tape” of operations and rolling it backwards with the Jacobian of each operator.
  - Implemented in PyTorch1.0, Chainer…
  - Easy if the front-end language is dynamic and interpreted (e.g Python)
  - Not so easy if we want to run without a Python runtime...

ConvNets on Graphs (fixed and data-dependent)



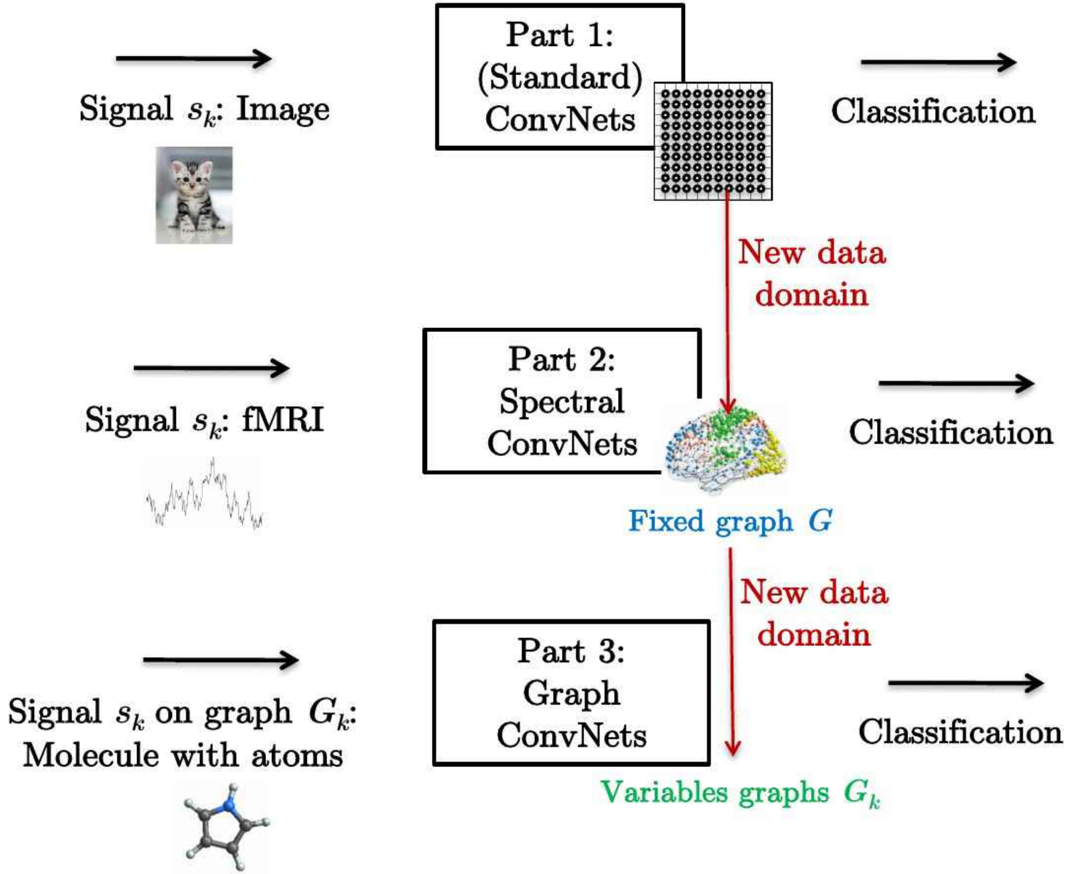
- **Graphs can represent: Natural language, social networks, chemistry, physics, communication networks...**



- **Review paper: "Geometric deep learning: going beyond euclidean data", MM Bronstein, J Bruna, Y LeCun, A Szlam, P Vandergheynst, IEEE Signal Processing Magazine 34 (4), 18-42, 2017 [ArXiv:1611.08097]**

Spectral ConvNets / Graph ConvNets

- **Regular grid graph**
  - Standard ConvNet
- **Fixed irregular graph**
  - Spectral ConvNet
- **Dynamic irregular graph**
  - Graph ConvNet

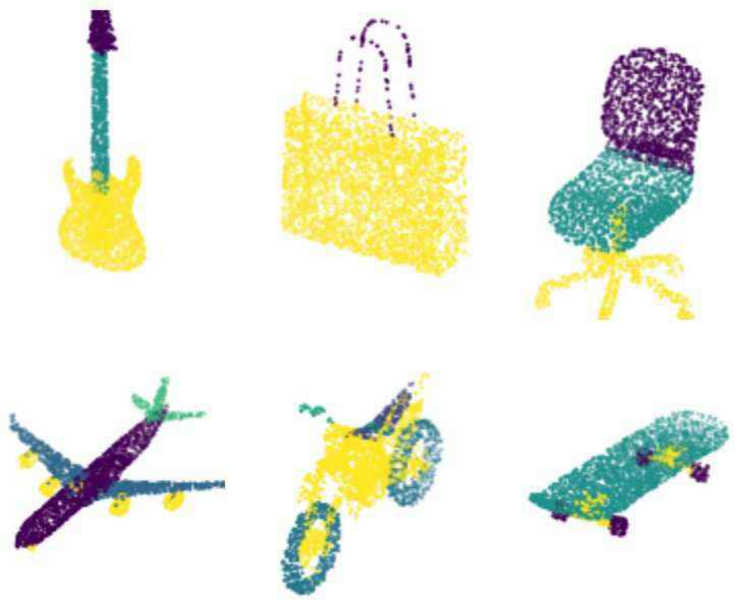


IPAM workshop:

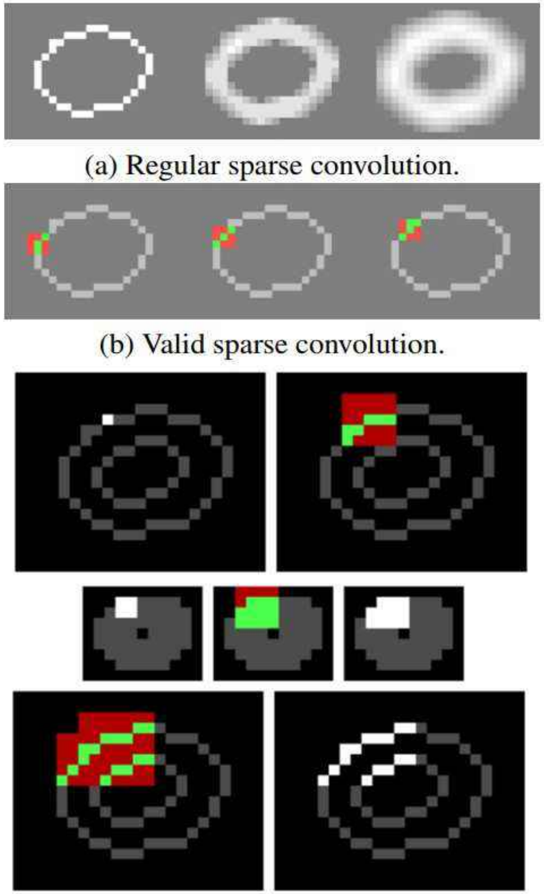
http://www.ipam.ucla.edu/programs/workshops/new-deep-learning-techniques/

Sparse ConvNets: for sparse voxel-based 3D data

- **ShapeNet competition results ArXiv:1710.06104]**
- **Winner: Submanifold Sparse ConvNet**
  - [Graham & van der Maaten arXiv 1706.01307]
  - PyTorch: https://github.com/facebookresearch/SparseConvNet



method	mean
SSCN	**86.00**
PdNet	85.49
DCPN	84.32
PCNN	82.29
PtAdLoss	77.96
KDTNet	65.80
DeepPool	42.79
NN	77.57
[19]	84.74



(a) Regular sparse convolution.

(b) Valid sparse convolution.

:) Block with a strided, a valid, and a de-convolution.

Lessons learned #3

- ***3.1: Dynamic networks are gaining in popularity (e.g. for NLP)***
  - *Dynamicity breaks many assumptions of current hardware*
  - *Can't optimize the compute graph distribution at compile time.*
  - *Can't do batching easily!*
- ***3.2: Large-Scale Memory-Augmented Networks...***
  - *...Will require efficient associative memory/nearest-neighbor search*
- ***3.3: Graph ConvNets are very promising for many applications***
  - *Say goodbye to matrix multiplications?*
  - *Say goodbye to tensors?*
- ***3.4: Large Neural Nets may have sparse activity***
  - *How to exploit sparsity in hardware?*

What About (Deep) Reinforcement Learning?

It works great …
…for games and virtual environments

Reinforcement Learning works fine for games

- **RL works well for games**
  - Playing Atari games [Mnih 2013], Go [Silver 2016, Tian 2018], Doom [Tian 2017], StarCraft...
  - RL requires too many trials.
  - 100 hours to reach the performance that a human can reach in 15 minutes on Atari games [Hessel ArXiv:1710.02298]
  - RL often doesn’t really work in the real world
- **FAIR open Source go player: OpenGo**

  https://github.com/pytorch/elf

Pure RL is hard to use in the real world

- **Pure RL requires too many trials to learn anything**
  - it's OK in a game
  - it's not OK in the real world
- **RL works in simple virtual world that you can run faster than real-time on many machines in parallel.**

- **Anything you do in the real world can kill you**
- **You can't run the real world faster than real time**

What are we missing to get to “real” AI?

- **What we can have**
  - Safer cars, autonomous cars
  - Better medical image analysis
  - Personalized medicine
  - Adequate language translation
  - Useful but stupid chatbots
  - Information search, retrieval, filtering
  - Numerous applications in energy, finance, manufacturing, environmental protection, commerce, law, artistic creation, games,…..

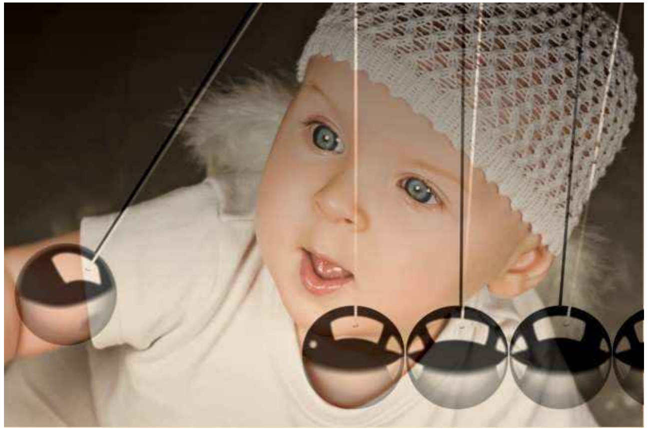
- **What we cannot have (yet)**
  - Machines with common sense
  - Intelligent personal assistants
  - “Smart” chatbots”
  - Household robots
  - Agile and dexterous robots
  - Artificial General Intelligence (AGI)

How do Humans and Animal Learn?

So quickly

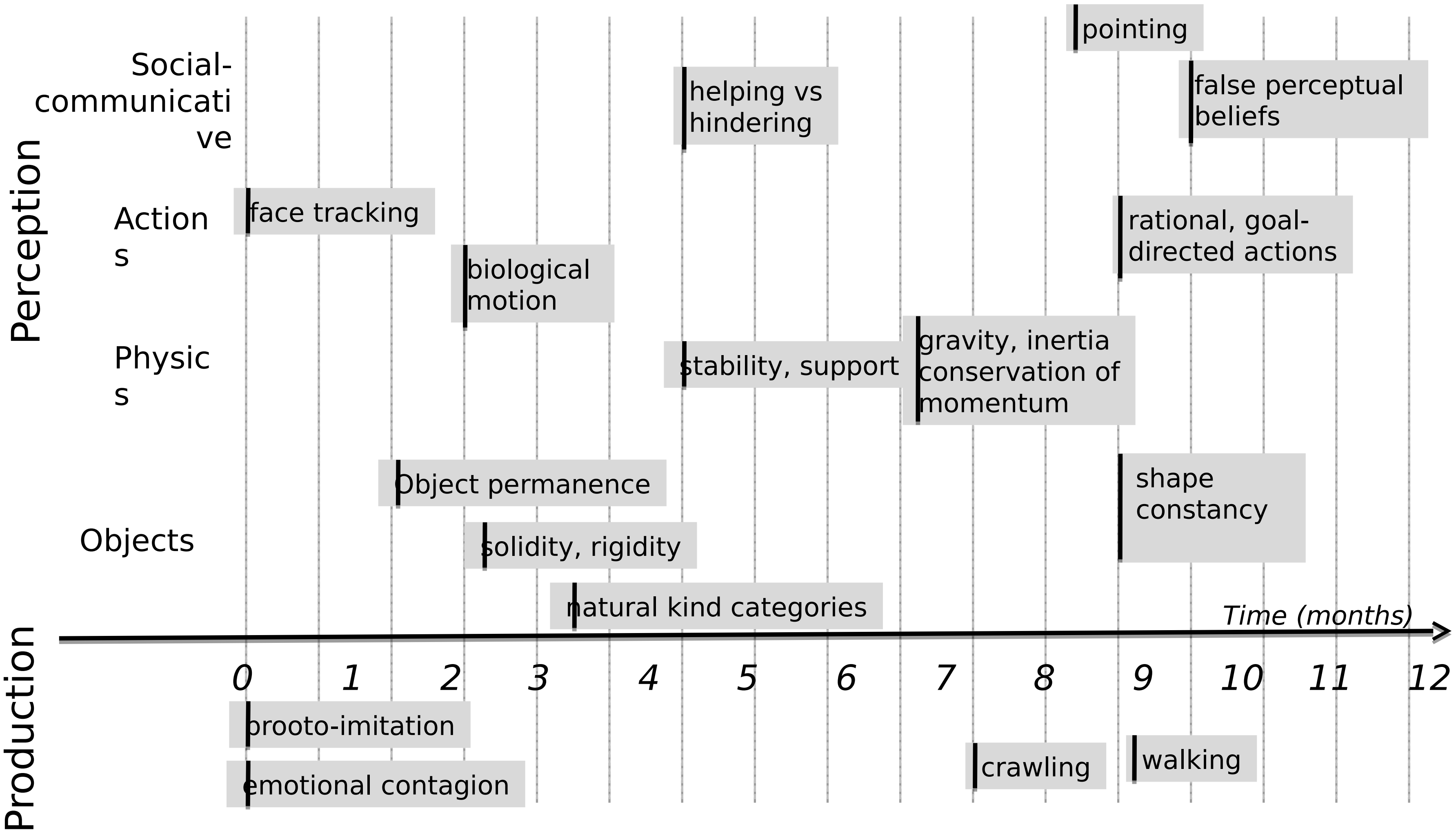
Babies learn how the world works by observation

- **Largely by observation, with remarkably little interaction.**



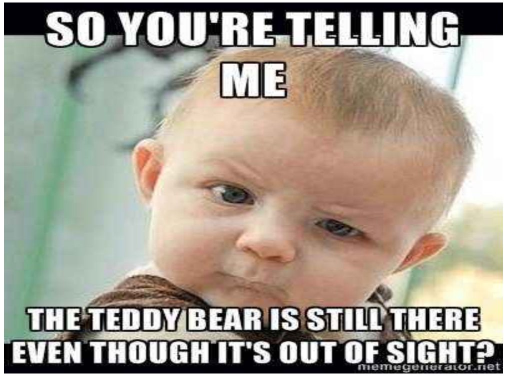
Photos courtesy of Emmanuel Dupoux

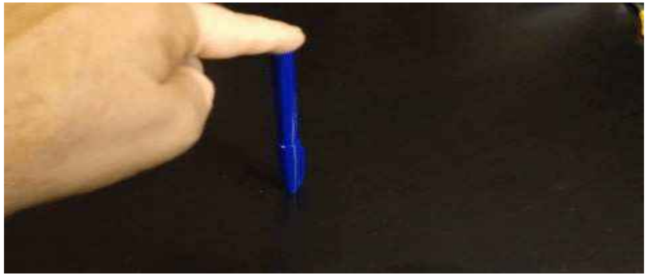
Early Conceptual Acquisition in Infants [from Emmanuel Dupoux]

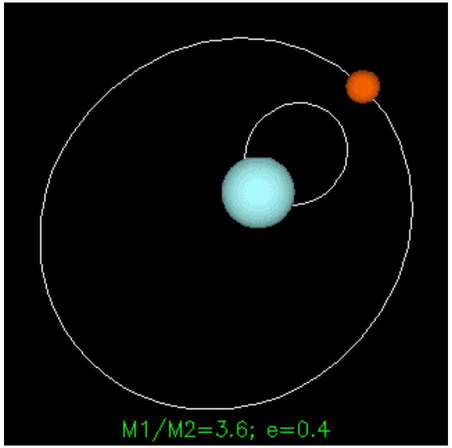


Prediction is the essence of Intelligence

- We learn models of the world by predicting





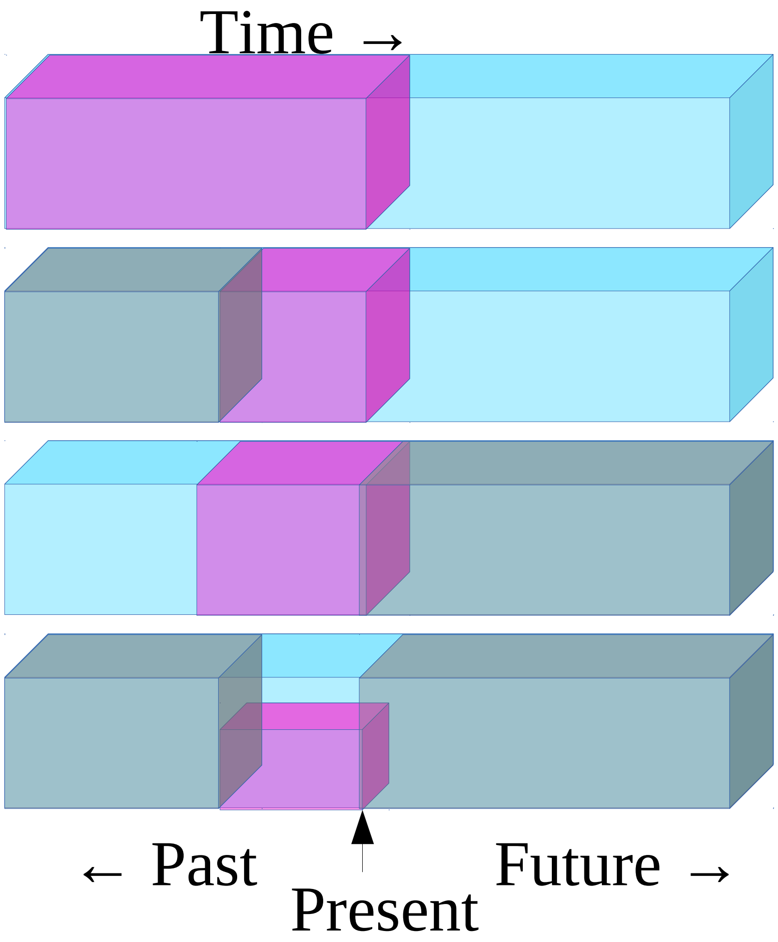


The Future: Self-Supervised Learning

With massive amounts of data and very large networks

Self-Supervised Learning

- **Predict any part of the input from any other part.**
- **Predict the future from the past.**
- **Predict the future from the recent past.**
- **Predict the past from the present.**
- **Predict the top from the bottom.**
- **Predict the occluded from the visible**
- **Pretend there is a part of the input you don't know and predict that.**



How Much Information is the Machine Given during Learning?

- **“Pure” Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

Y. LeCun

Self-Supervised Learning: Filling in the Blanks

Self-Supervised Learning works well for text

- **Word2vec**
  - [Mikolov 2013]
- **FastText**
  - [Joulin 2016]
- **BERT**
  - Bidirectional Encoder Representations from Transformers
  - [Devlin 2018]

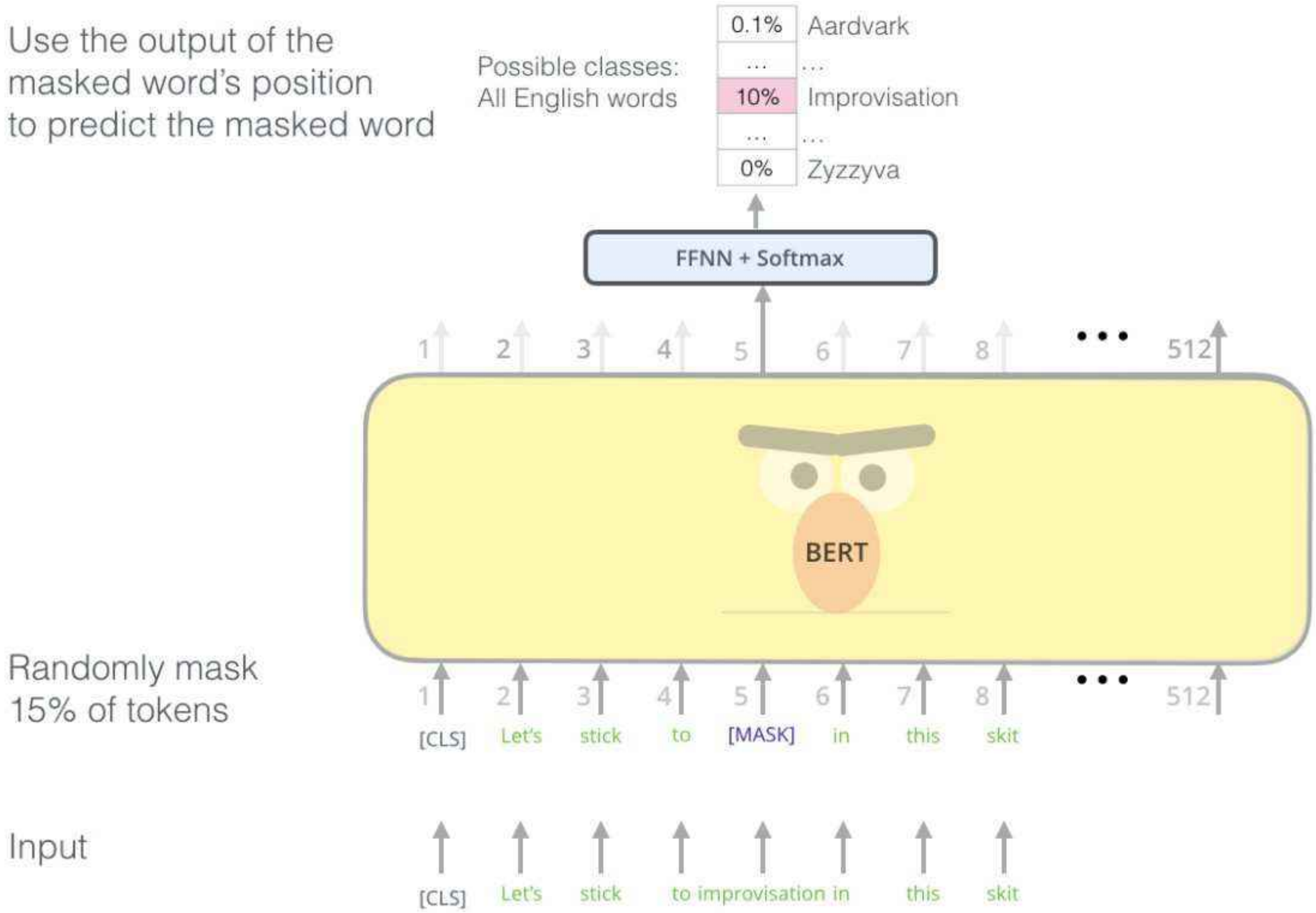
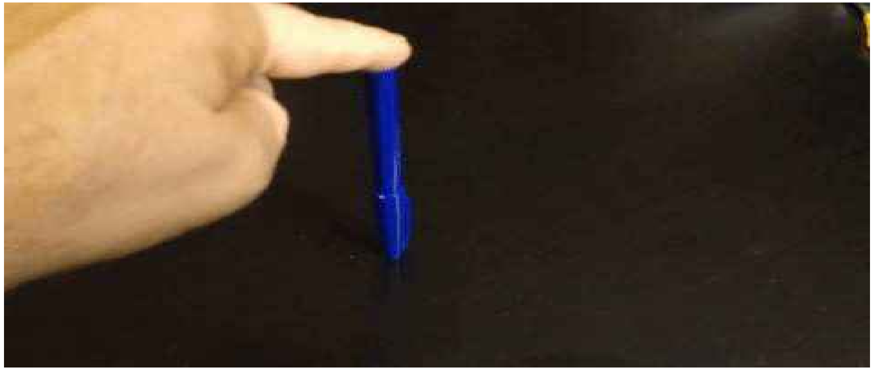


Figure credit: Jay Alammar http://jalammar.github.io/illustrated-bert/

But it doesn't really work for high-dim continuous signals



- **Video prediction:**
  - Multiple futures are possible.
  - Training a system to make a single prediction results in "blurry" results
  - the average of all the possible futures

The Next AI Revolution

With thanks
To
Alyosha Efros

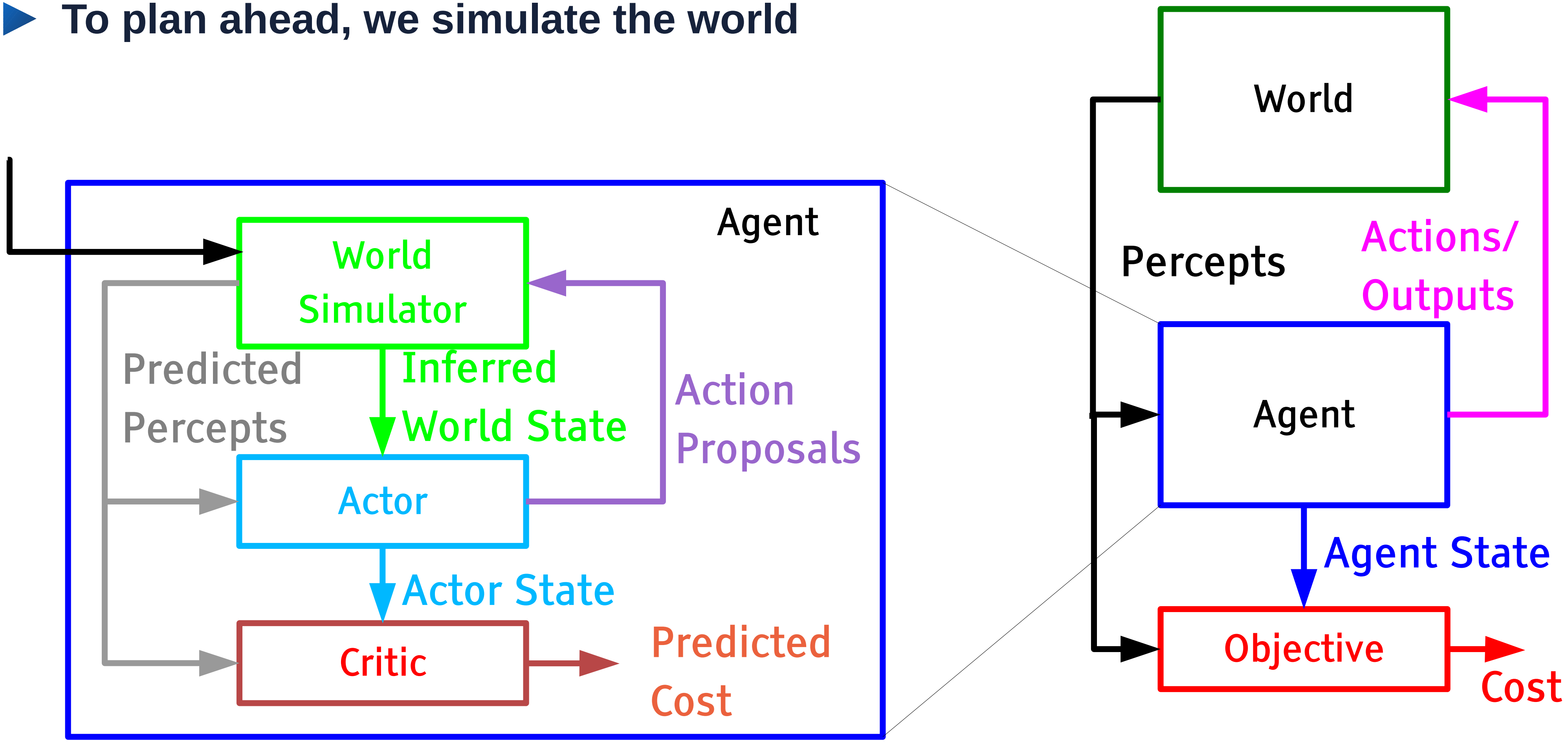
Learning Predictive Models of the World

Learning to predict, reason, and plan,
Learning Common Sense.

facebook
Artificial Intelligence Research

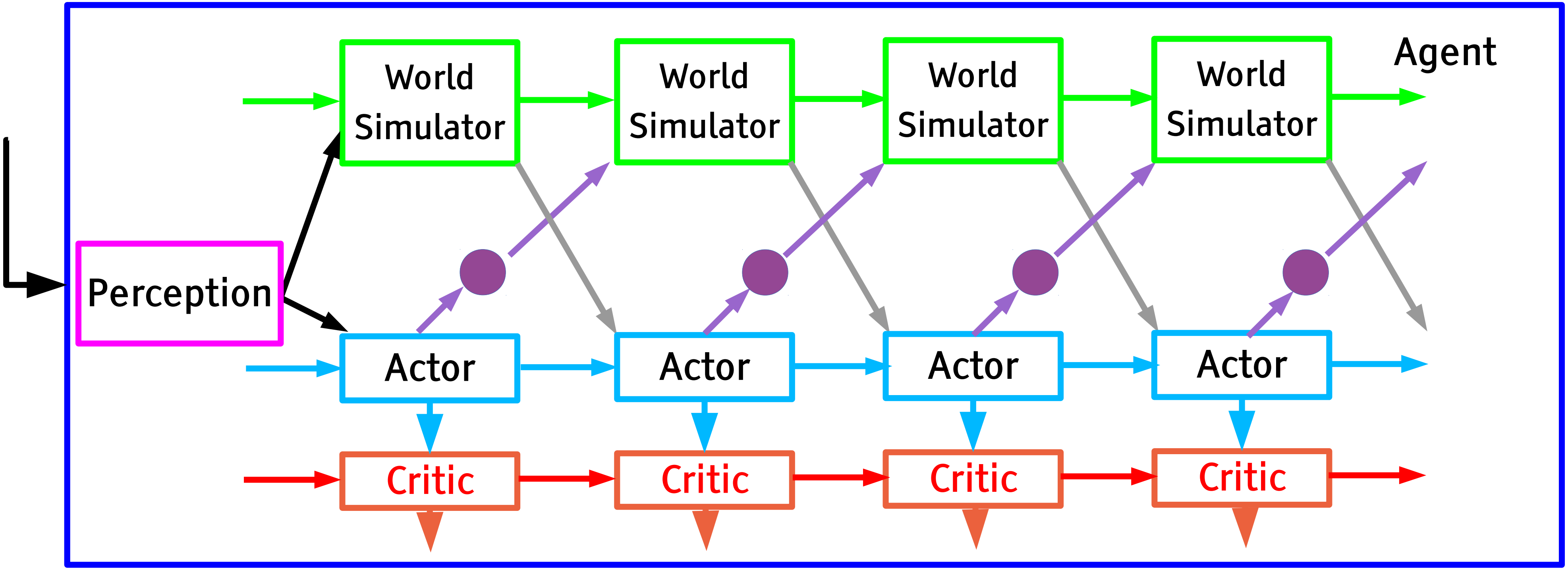
Planning Requires Prediction

- **To plan ahead, we simulate the world**



Training the Actor with Optimized Action Sequences

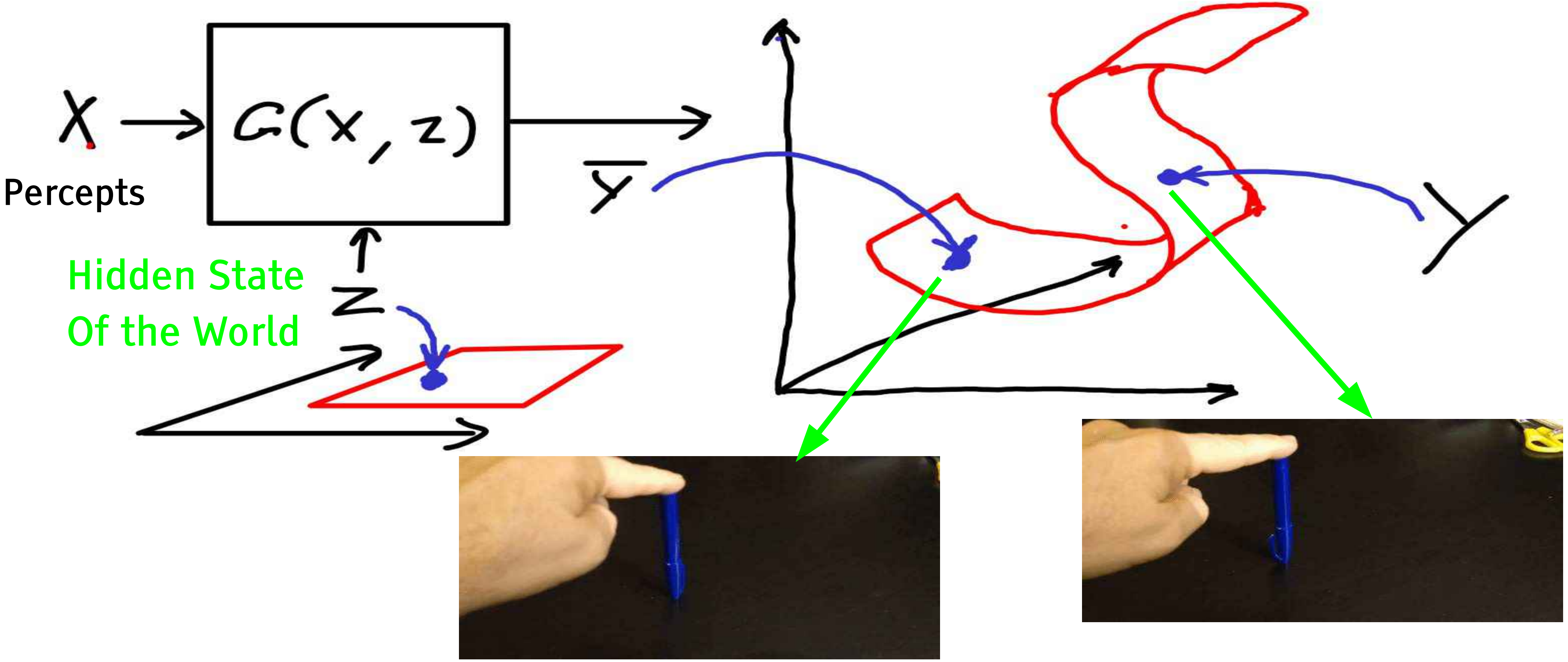
- **1. Find action sequence through optimization**
- **2. Use sequence as target to train the actor**
  - Over time we get a compact policy that requires no run-time optimization



Y. LeCun

The Hard Part: Prediction Under Uncertainty

- **Invariant prediction: The training samples are merely representatives of a whole set of possible outputs (e.g. a manifold of outputs).**



Faces "invented" by a GAN (Generative Adversarial Network)

- **Random vector → Generator Network → output image [Goodfellow NIPS 2014] [Karras et al. ICLR 2018] (from NVIDIA)**

Generative Adversarial Networks for Creation

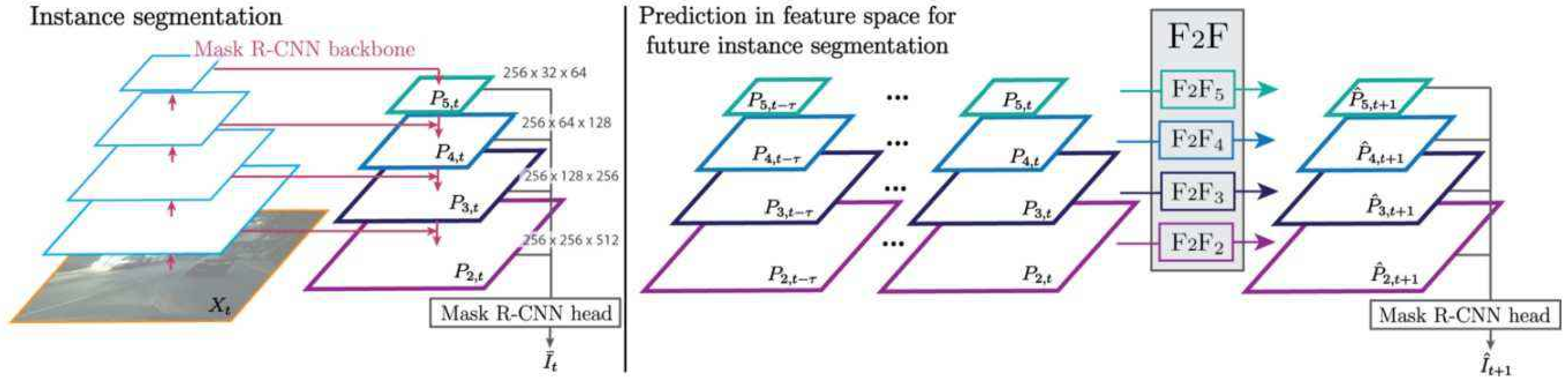
- [Sbai 2017]

Self-supervised Adversarial Learning for Video Prediction

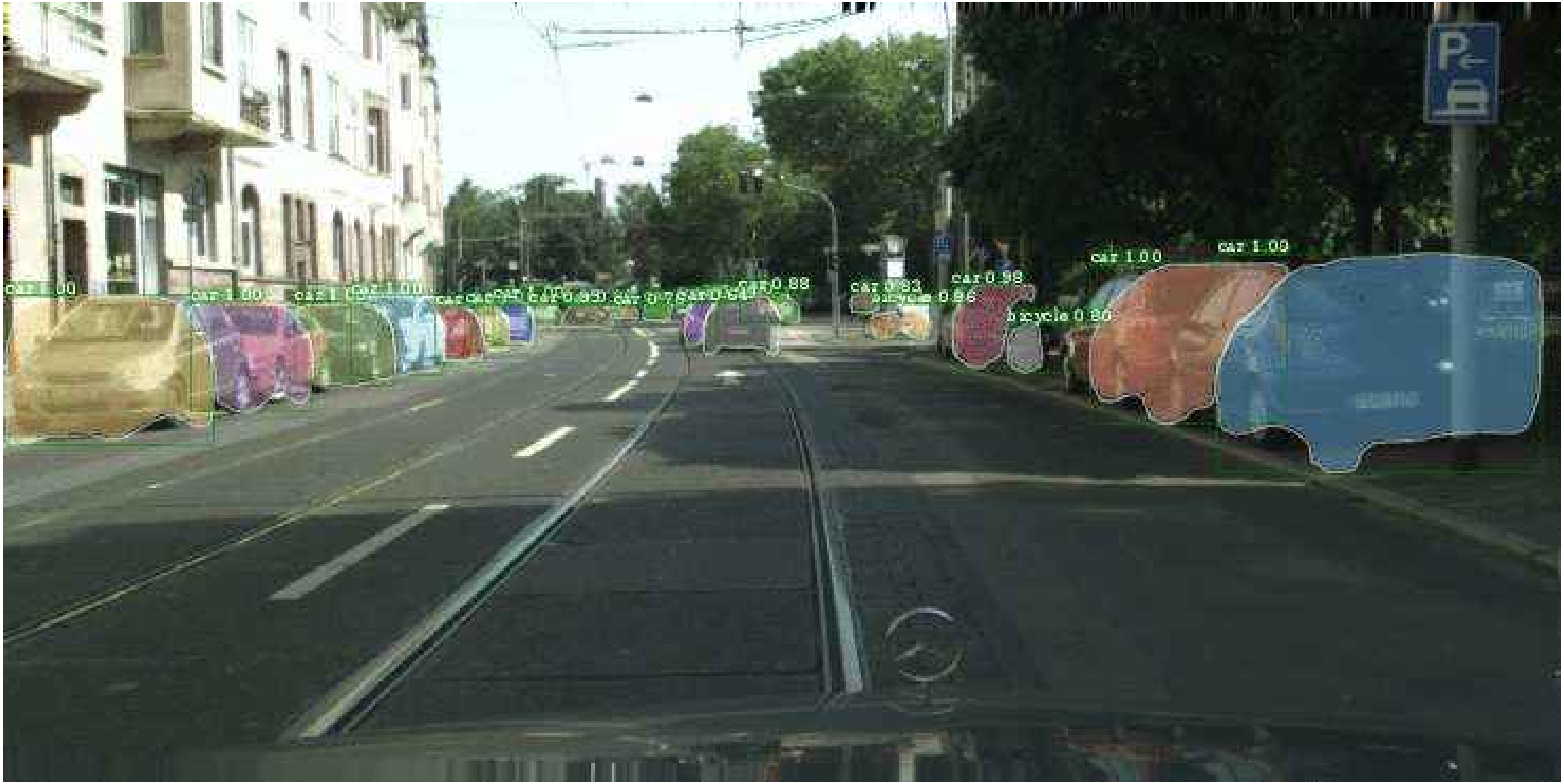
- **Our brains are "prediction machines"**
- **Can we train machines to predict the future?**
- **Some success with "adversarial training"**
  - [Mathieu, Couprie, LeCun arXiv:1511:05440]
- **But we are far from a complete solution.**

Predicting Instance Segmentation Maps

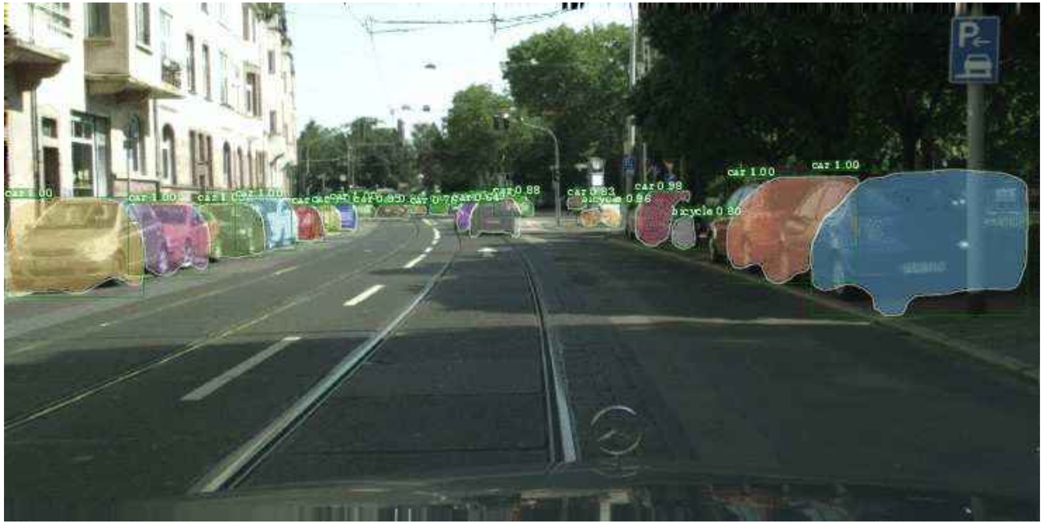
- [Luc, Couprie, LeCun, Verbeek ECCV 2018]
- Mask R-CNN Feature Pyramid Network backbone
- Trained for instance segmentation on COCO
- Separate predictors for each feature level

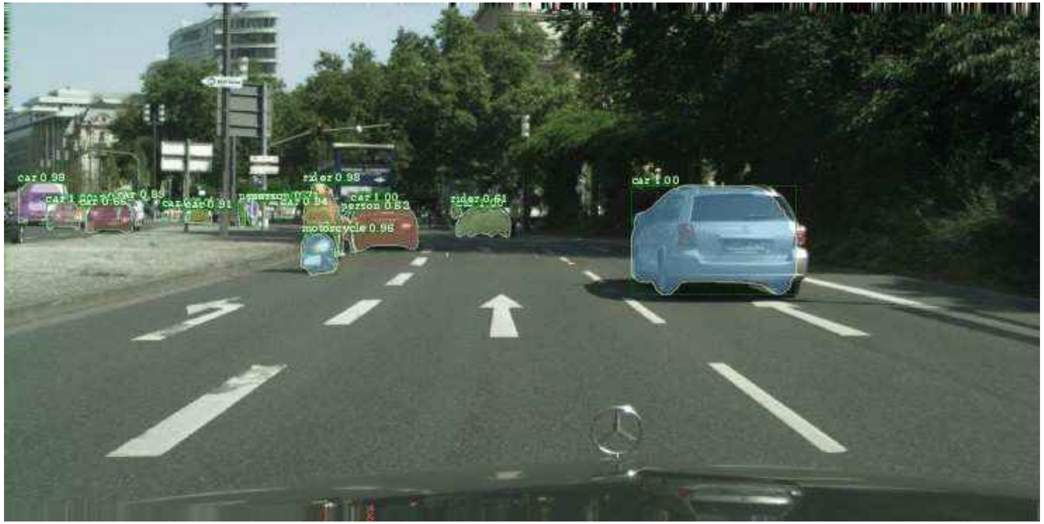


Predictions



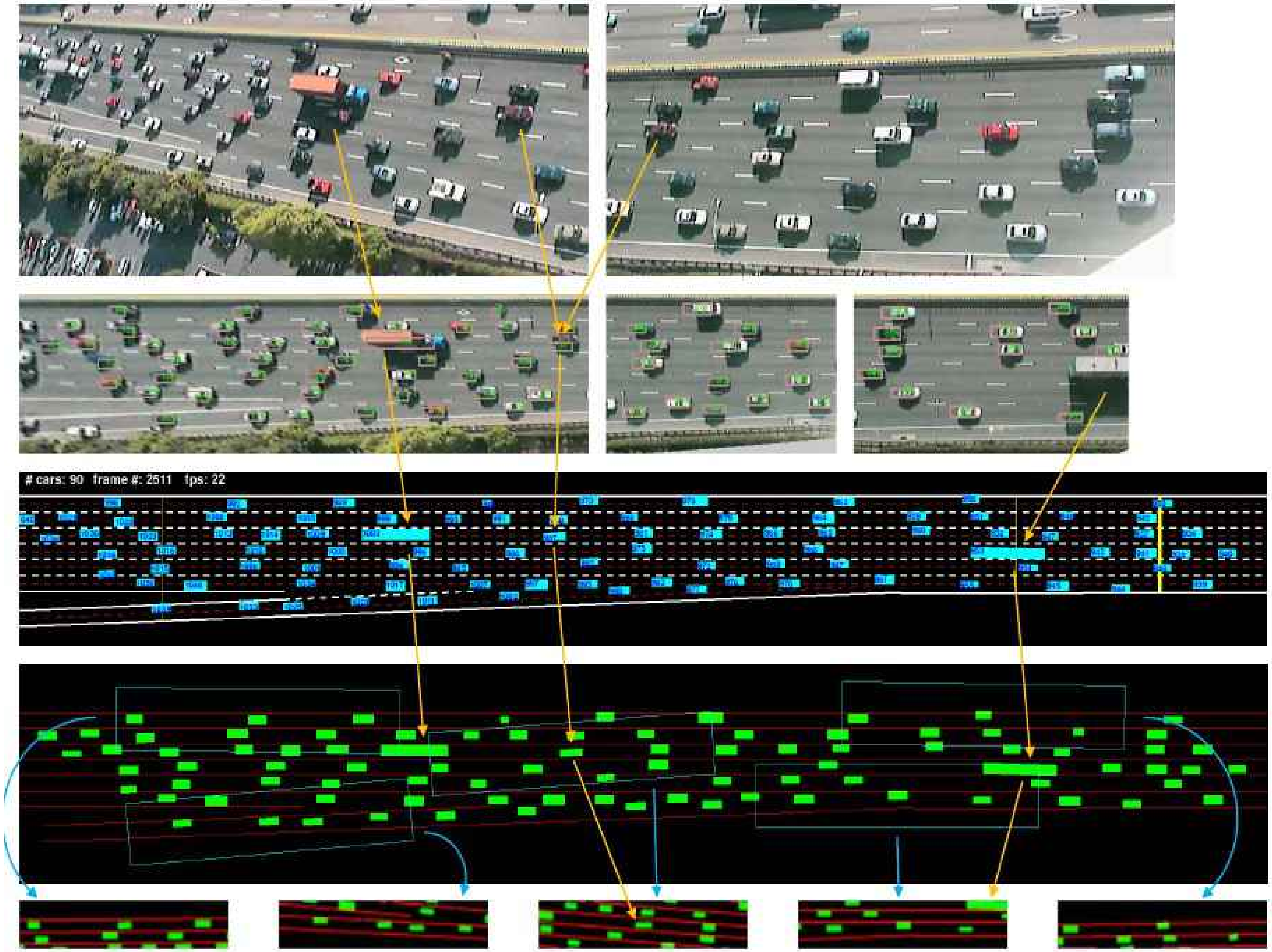
Long-term predictions (10 frames, 1.8 seconds)





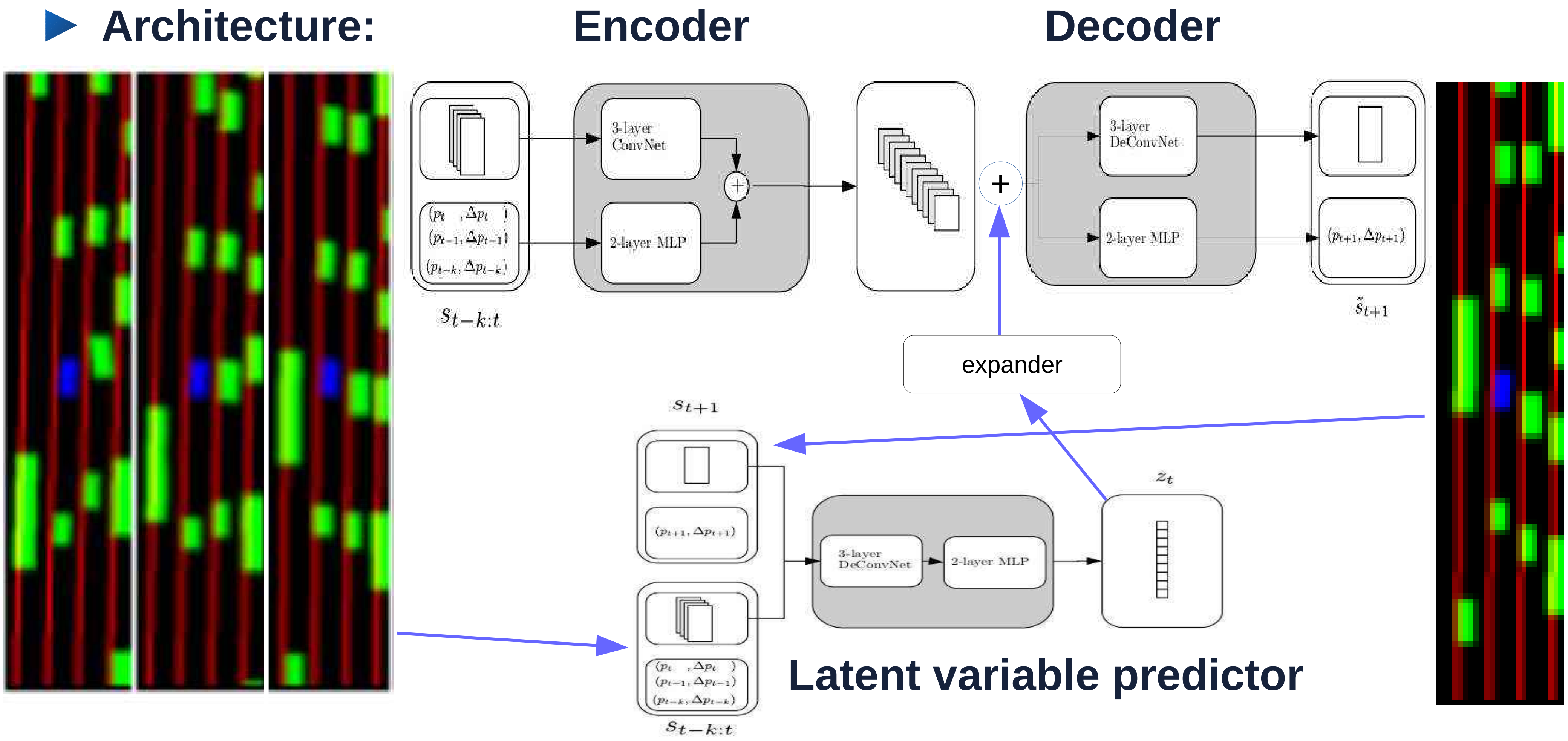
Using Forward Models to Plan (and to learn to drive)

- **Overhead camera on highway.**
  - Vehicles are tracked
- **A "state" is a pixel representation of a rectangular window centered around each car.**
- **Forward model is trained to predict how every car moves relative to the central car.**
  - steering and acceleration are computed

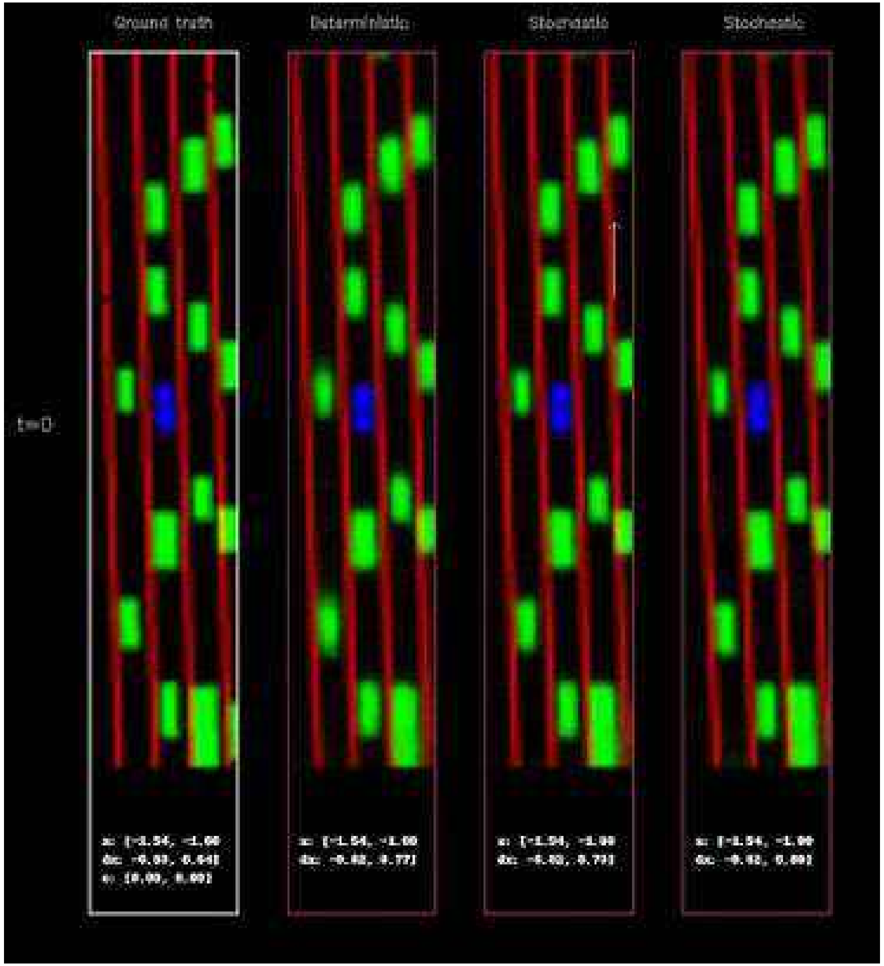


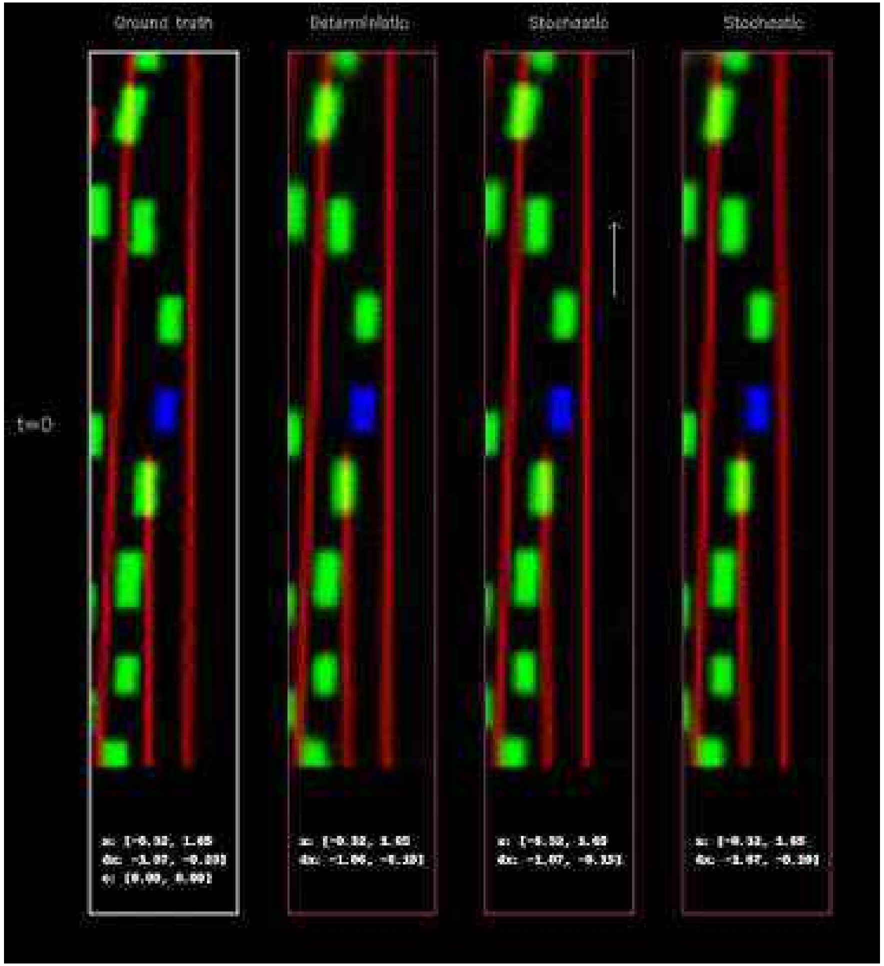
Forward Model Architecture

- **Architecture:**



Predictions

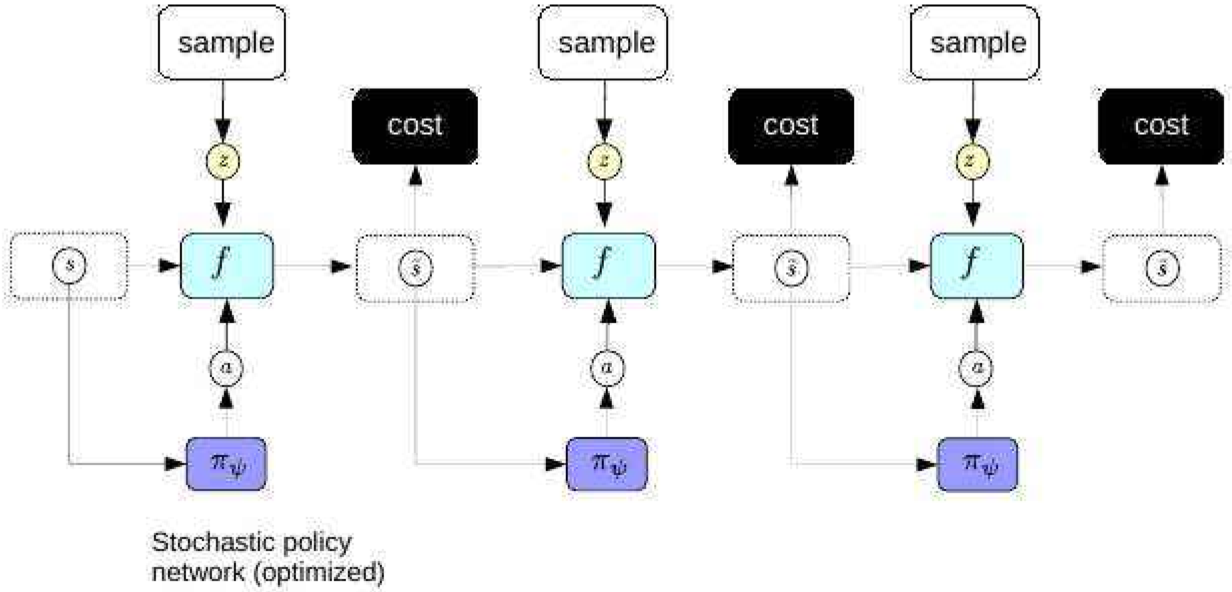




Learning to Drive by Simulating it in your Head

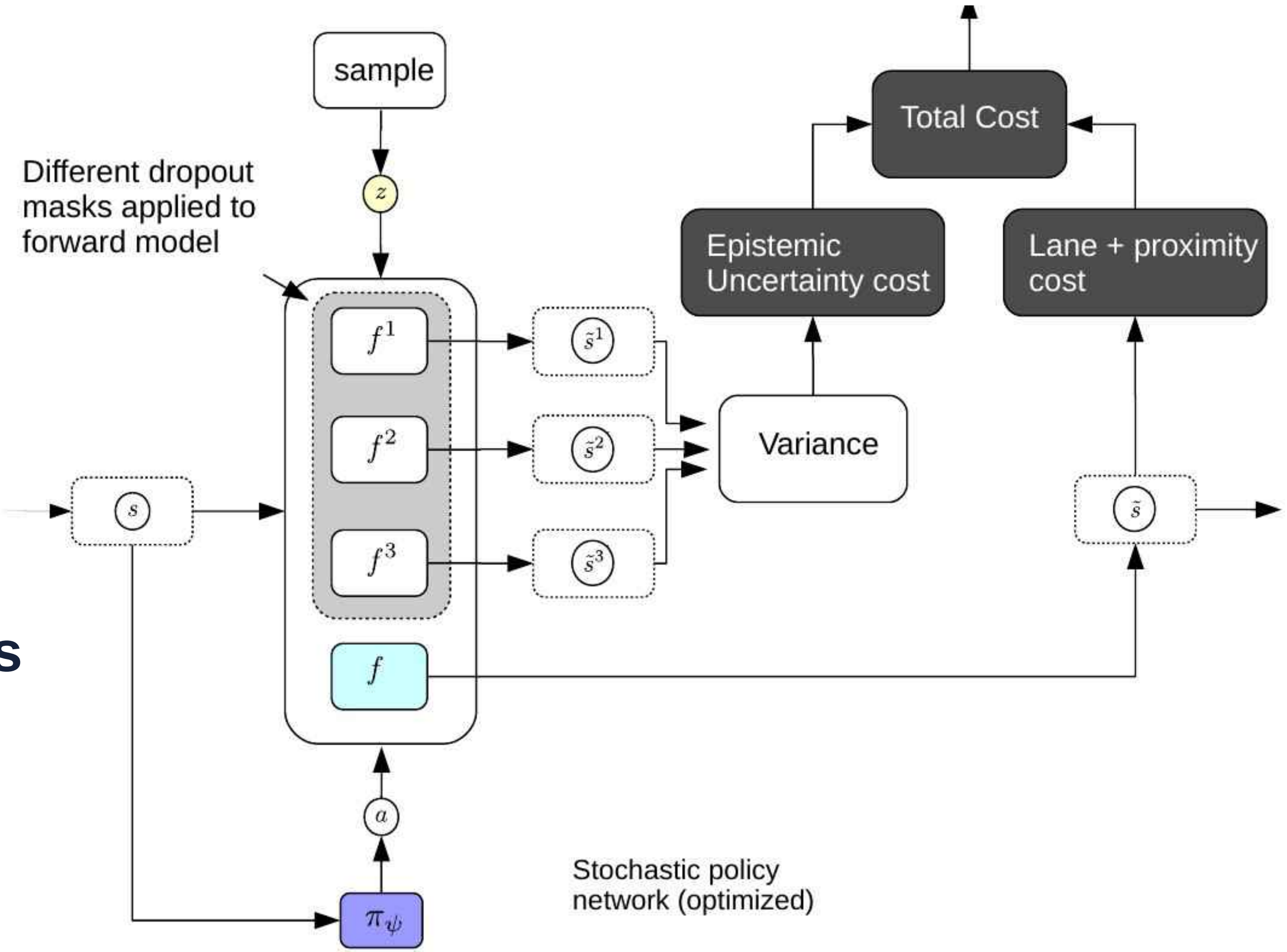
- **Feed initial state**
- **Sample latent variable sequences of length 20**
- **Run the forward model with these sequences**
- **Backpropagate gradient of cost to train a policy network.**
- **Iterate**

- **No need for planning at run time.**

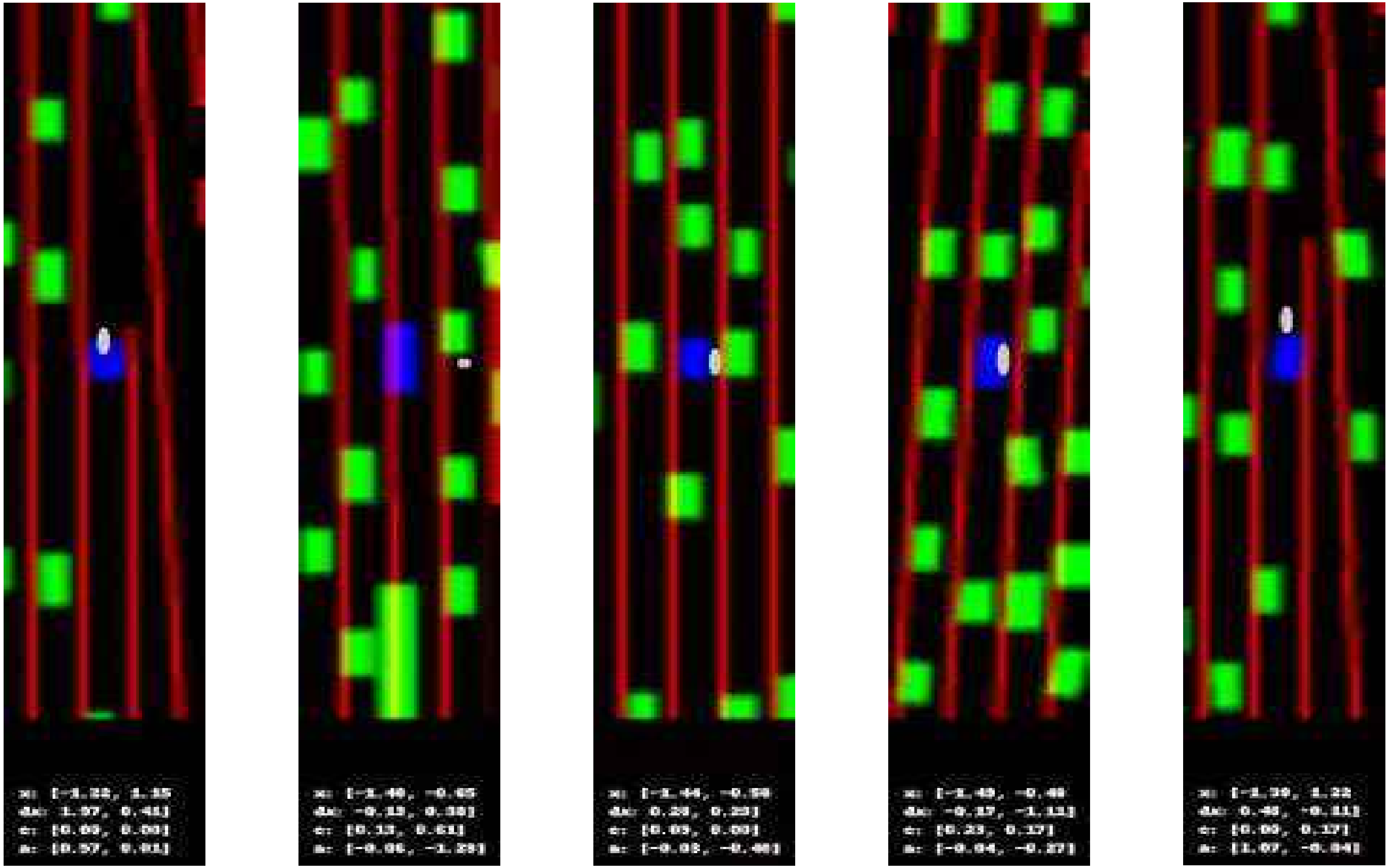


Adding an Uncertainty Cost (doesn't work without it)

- **Estimates epistemic uncertainty**
- **Samples multiple drop-puts in forward model**
- **Computes variance of predictions (differentiably)**
- **Train the policy network to minimize the lane&proximity cost plus the uncertainty cost.**
- **Avoids unpredictable outcomes**



Driving an Invisible Car in "Real" Traffic



Lessons learned #4

- ***4.1: Self-Supervised learning is the future***
  - *Networks will be much larger than today, perhaps sparse*
- ***4.2: Reasoning/inference through minimization***
- ***4.3: DL hardware use cases***
  - *A. DL R&D: 32-bit FP, high parallelism, fast inter-node communication, flexible hardware and software.*
  - *B. Routine training: 16-bit FP, some parallelism, moderate cost.*
  - *C. inference in data centers: 8 or 16-bit FP, low latency, low power consumption, standard interface.*
  - *D. inference on embedded devices: low cost, low power, exotic number systems?*
    - *AR/VR, consumer items, household robots, toys, manufacturing, monitoring,...*

Speculations

- ***Spiking Neural Nets, and neuromorphic architectures?***
  - *I’m skeptical…..*
  - *No spike-based NN comes close to state of the art on practical tasks*
  - *Why build chips for algorithms that don’t work?*

- ***Exotic technologies?***
  - *Resistor/Memristor matrices, and other analog implementations?*
    - *Conversion to and from digital kills us.*
    - *No possibility of hardware multiplexing*
  - *Spintronics?*
  - *Optical implementations?*

Thank you

facebook
Artificial Intelligence Research