Deep Learning
Lecture 1 – Introduction & Convnets

Machine Learning Summer School – Madrid 2018

Rob Fergus

New York University
Facebook AI Research

Part 1 Schedule

- Deep Supervised Learning
    - Overview [30 mins]
    - Convolutional Networks [30 mins]
    - Residual Networks [20 mins]
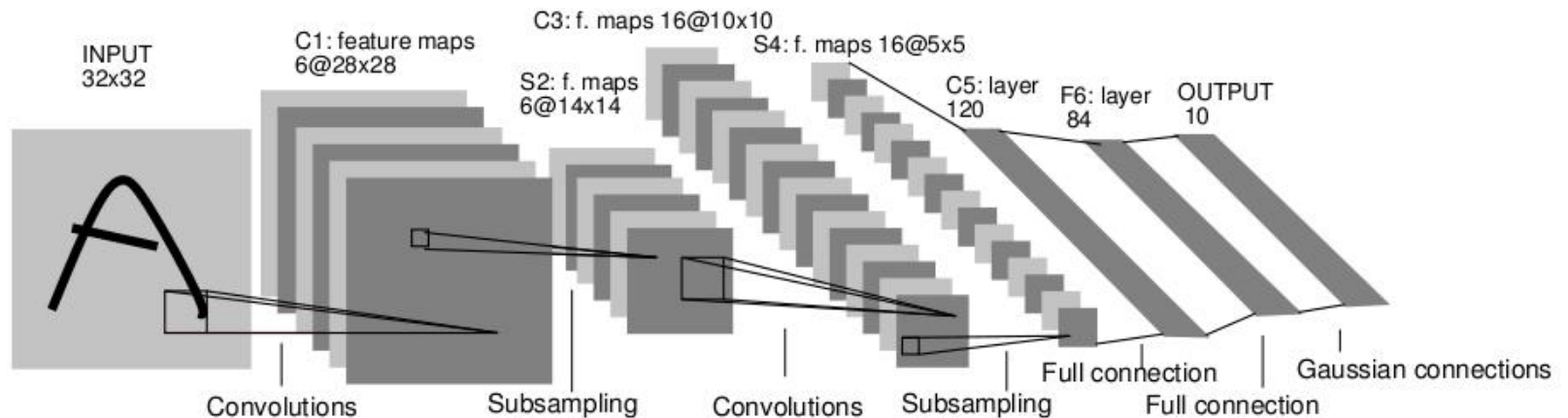   - Large scale training [5 mins]

History of Neural Nets

- 1st era (1940's-1960's): Invention
  - Connectionism [Hebb 1940's]: complex behaviors arise from interconnected networks of simple units
  - Artificial neurons [Hebb, McCulloch & Pitts 40's & 50's]
  - Perceptrons [Rosenblatt 50's]: Single layer with simple learning rule
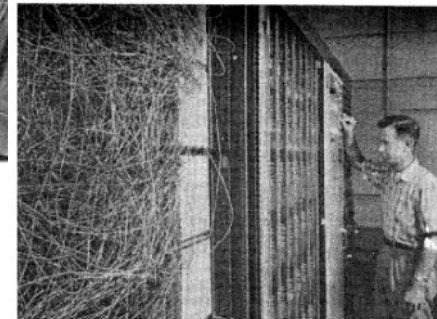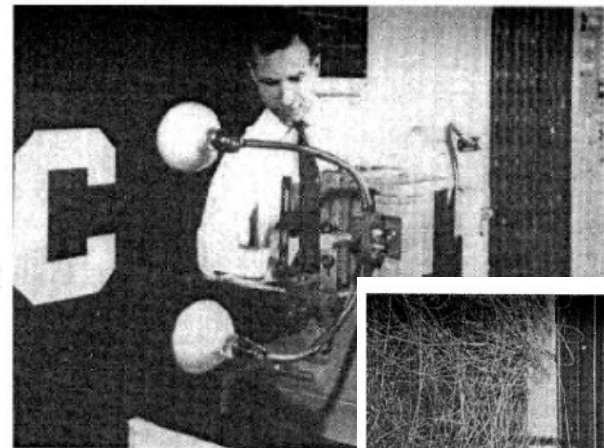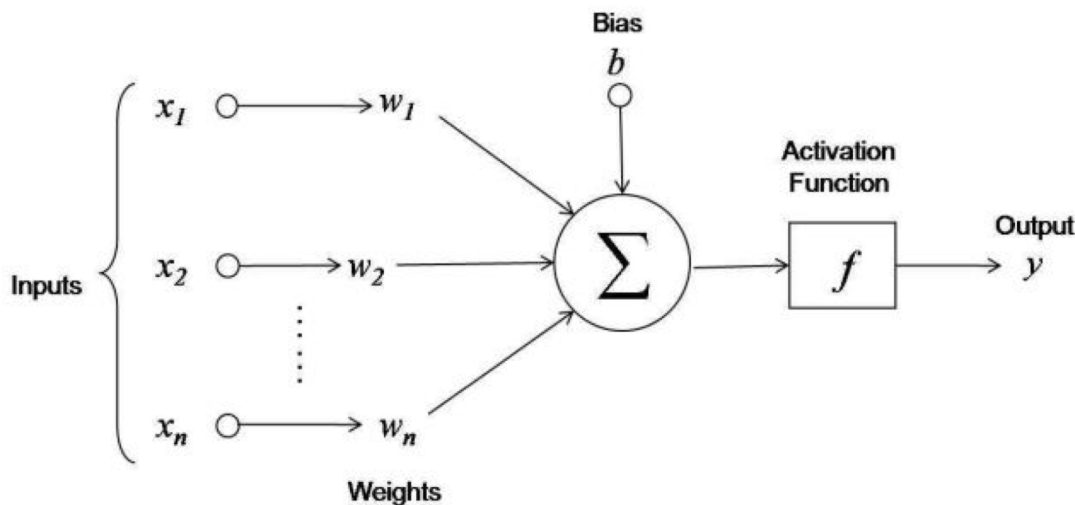
History of Neural Nets

- 2nd era (1980's-1990's): Multi-layered networks
  - Back-propagation [Rumelhart, Hinton & Williams 1986 + others]: effective way to train multi-layered networks
  - Convolutional networks [LeCun et al. 1989]: architecture adapted for images

INPUT
32x32

C1: feature maps
6@28x28

C3: f. maps 16@10x10

S2: f. maps
6@14x14

S4: f. maps 16@5x5

C5: layer
120

F6: layer
84

OUTPUT
10

Convolutions   Subsampling   Convolutions   Subsampling   Full connection   Gaussian connections
Full connection

The Deep Learning era [2011-present]

- Big gains in performance on perceptual tasks:
  - Vision
  - Speech understanding
  - Natural language processing
    - E.g. Translation between languages

- Three ingredients:
  1. Deep neural network models (supervised training)
     - Closely related to models from 1980's but much bigger
  2. Fast GPU computation
  3. Big labeled datasets

- Exciting progress on other AI tasks (e.g. playing Go)

Big Annotated Image Datasets
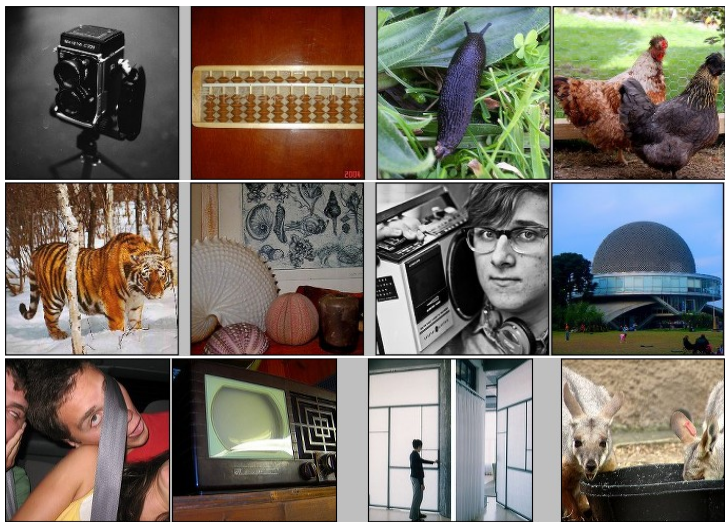
- Stanford Vision group [Deng et al. 2009]
- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon Turk

- Microsoft + universities [2014]
- 2 million objects in natural settings
- Human labels via Amazon Turk

Powerful Hardware

- Deep neural nets highly amenable to implementation on Graphics Processing Units (GPUs)
  - Matrix multiply
  - 2D convolution

- Latest generation nVidia GPUs (Pascal) deliver 10 TFlops
  - Faster than fastest computer in world in 2000
  - 10 million times faster than 1980's Sun workstation

Deep Neural Network for Vision

- Krizhevsky, Sutskever & Hinton [NIPS2012]
  - 8 layer Convolutional network model [LeCun et al. '89]
  - 7 hidden layers, 650,000 neurons, ~60,000,000 parameters
  - Trained on 1.2 million ImageNet images (with labels)
  - GPU implementation (50x speedup over CPU)
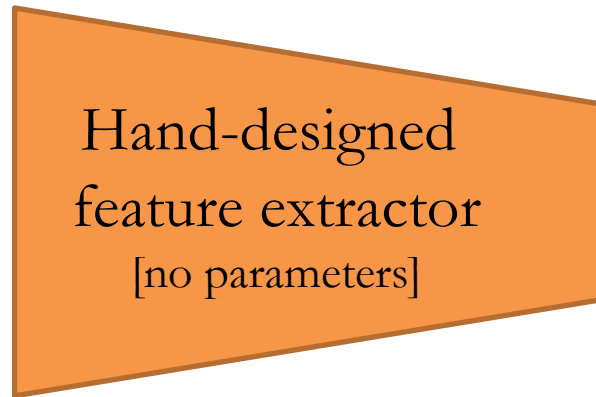 - Training time: 1 week on pair of GPUs
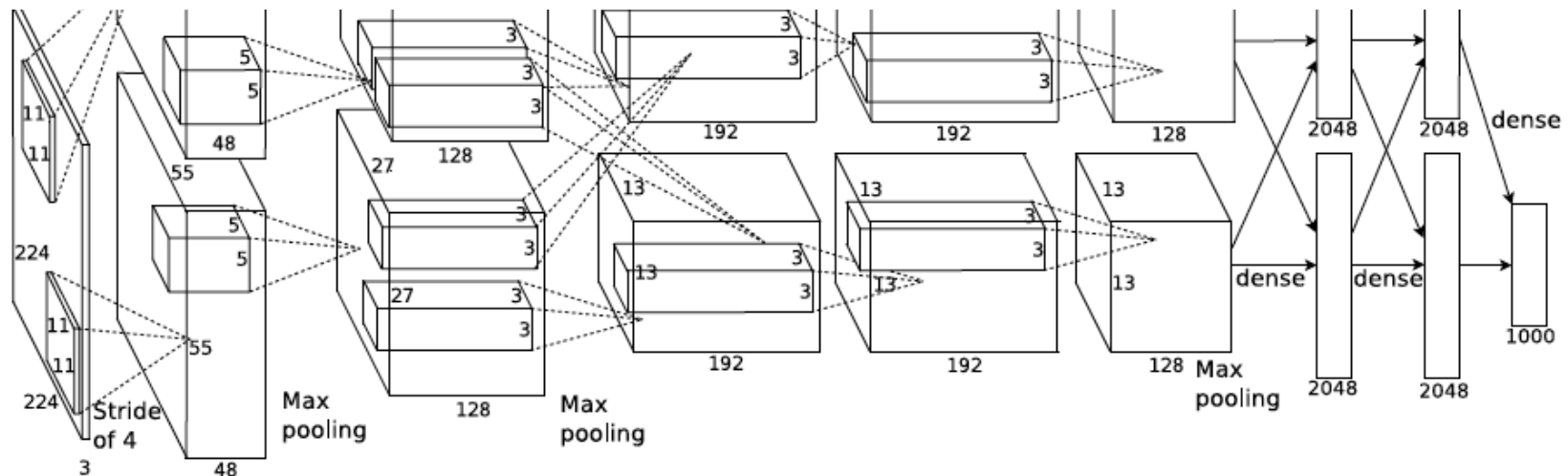
Input Image

Class Prediction

Deep Learning vs Traditional Approaches

- ## Traditional Approach

Input Image

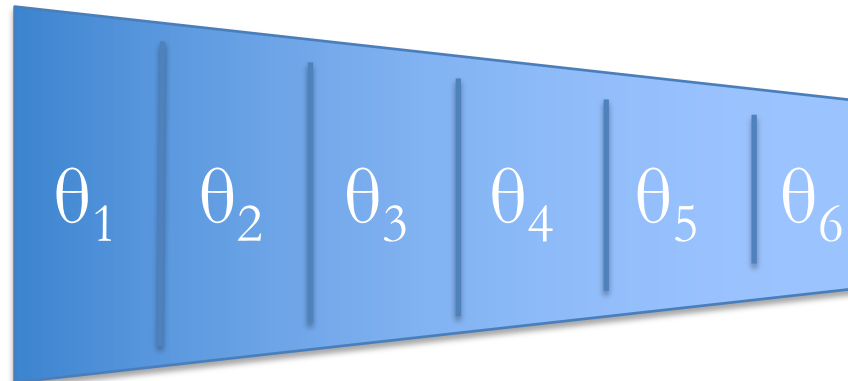Hand-designed feature extractor [no parameters]

Simple Classifier θ

Only part that is learnt

Predicted label: Abacus

- ## Deep Neural Network

Input Image

θ_1 θ_2 θ_3 θ_4 θ_5 θ_6
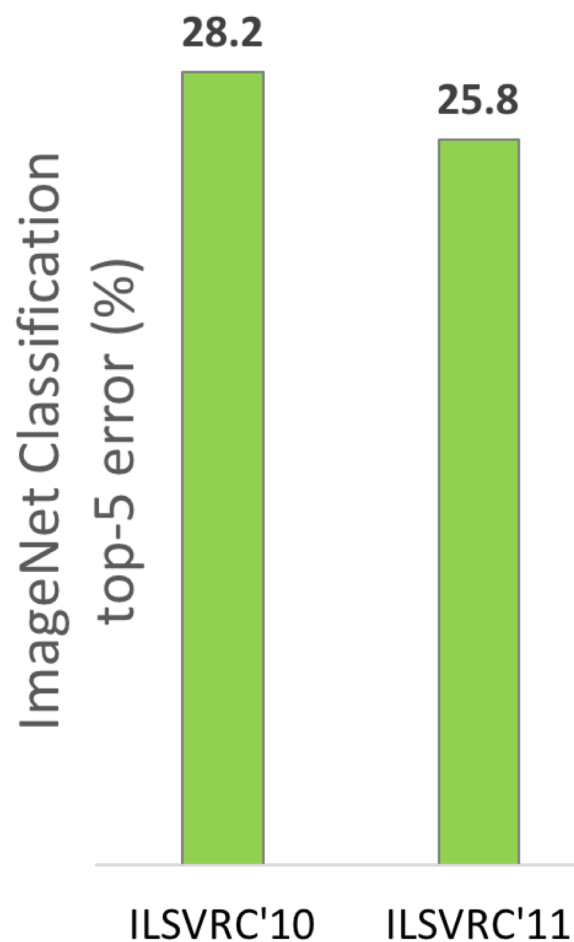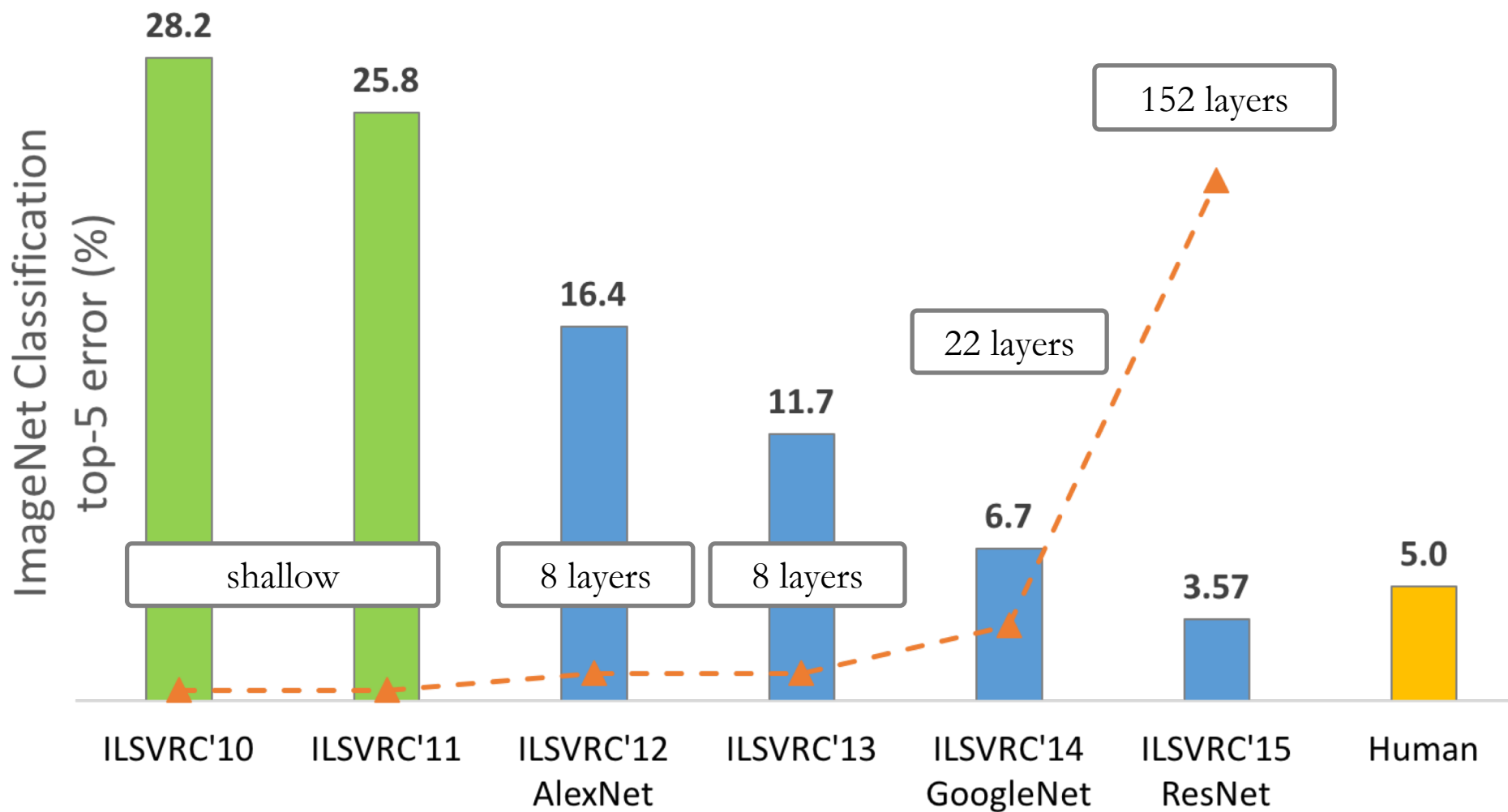
End-to-end Learning

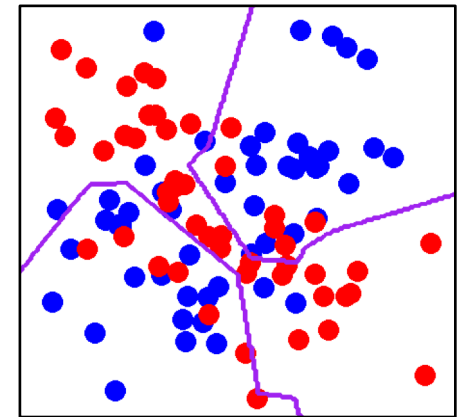Predicted label: Abacus

ImageNet Performance over time

Growth in Model Depth

Depth is Key
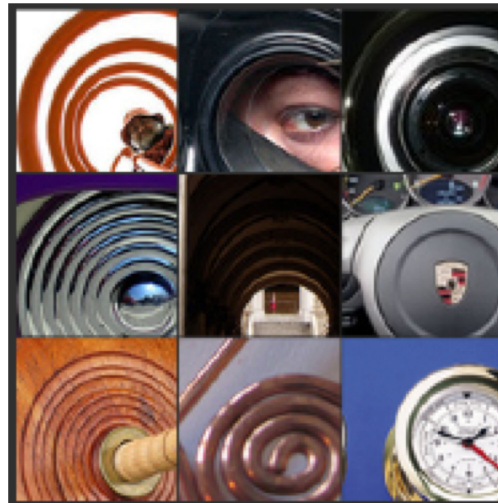
- Each layer is simple non-linear function

- Composition of them yields complex decision surfaces
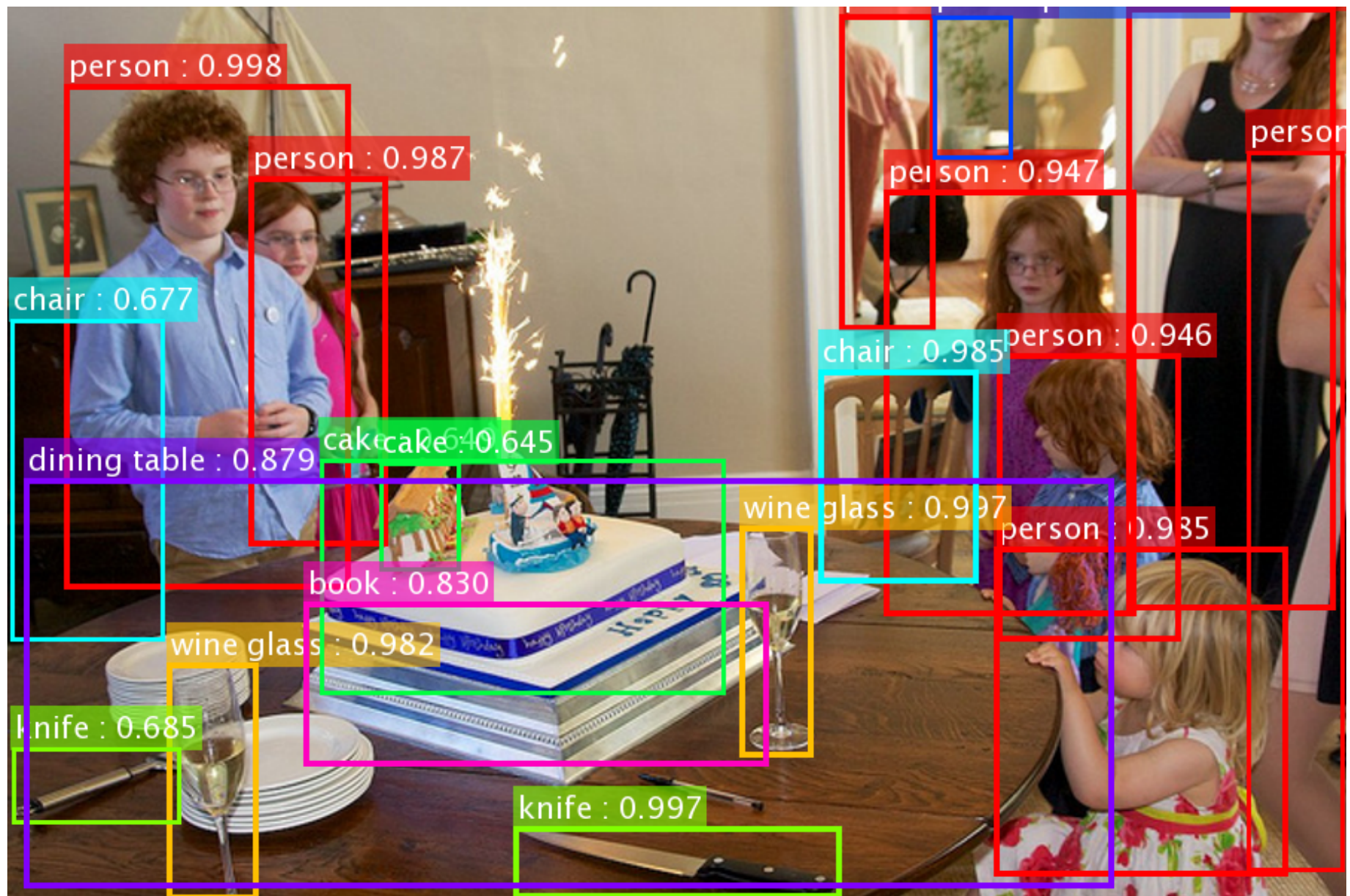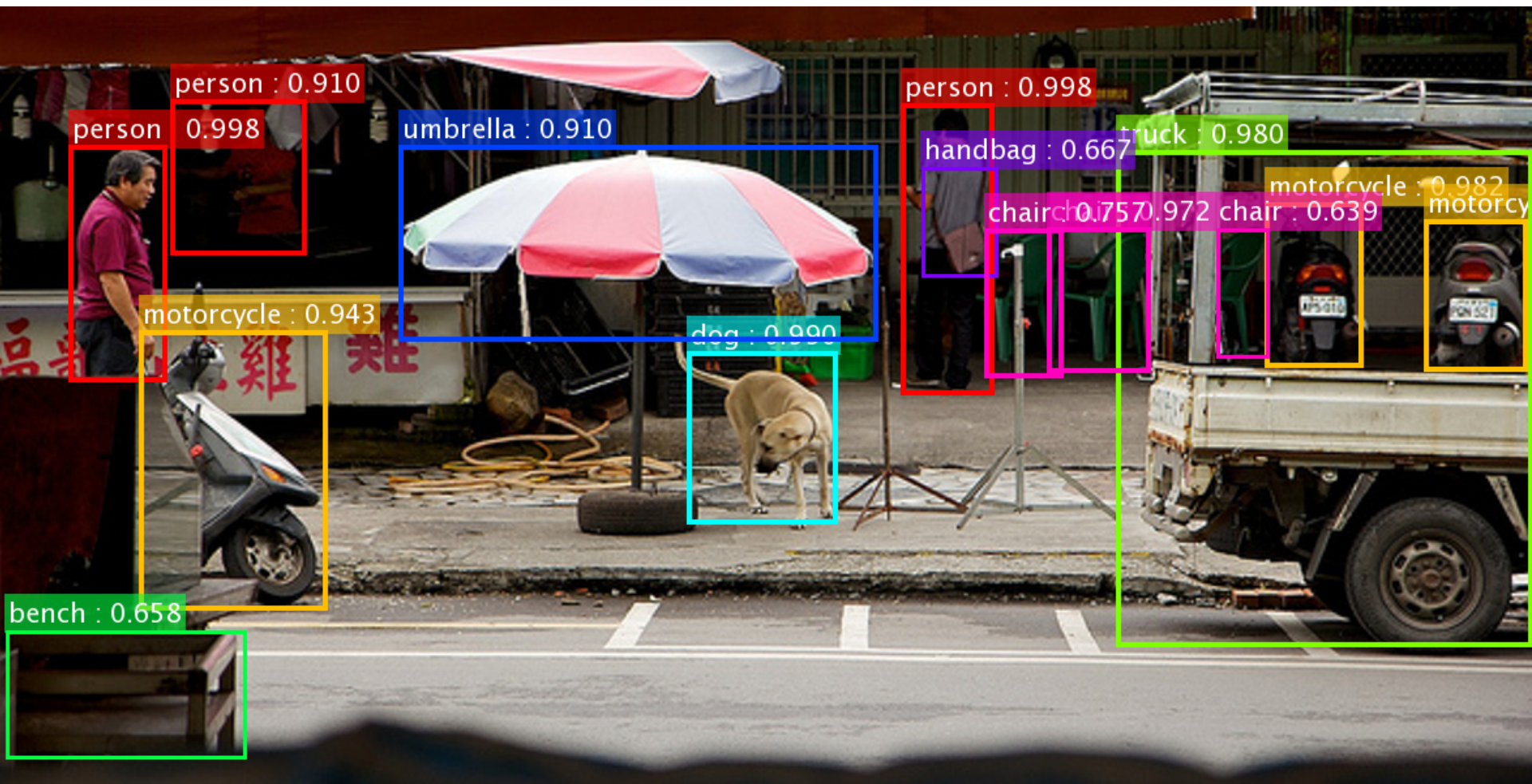
- Can learn very complex invariances

[http://www.kdnuggets.com]

- Visualization of features in trained model

[Zeiler & Fergus, ECCV 2014]

He, Zhang, Ren, & Sun. "Deep Residual Learning for Image Recognition". ICCV 2015.

He, Zhang, Ren, & Sun. "Deep Residual Learning for Image Recognition". ICCV 2015.
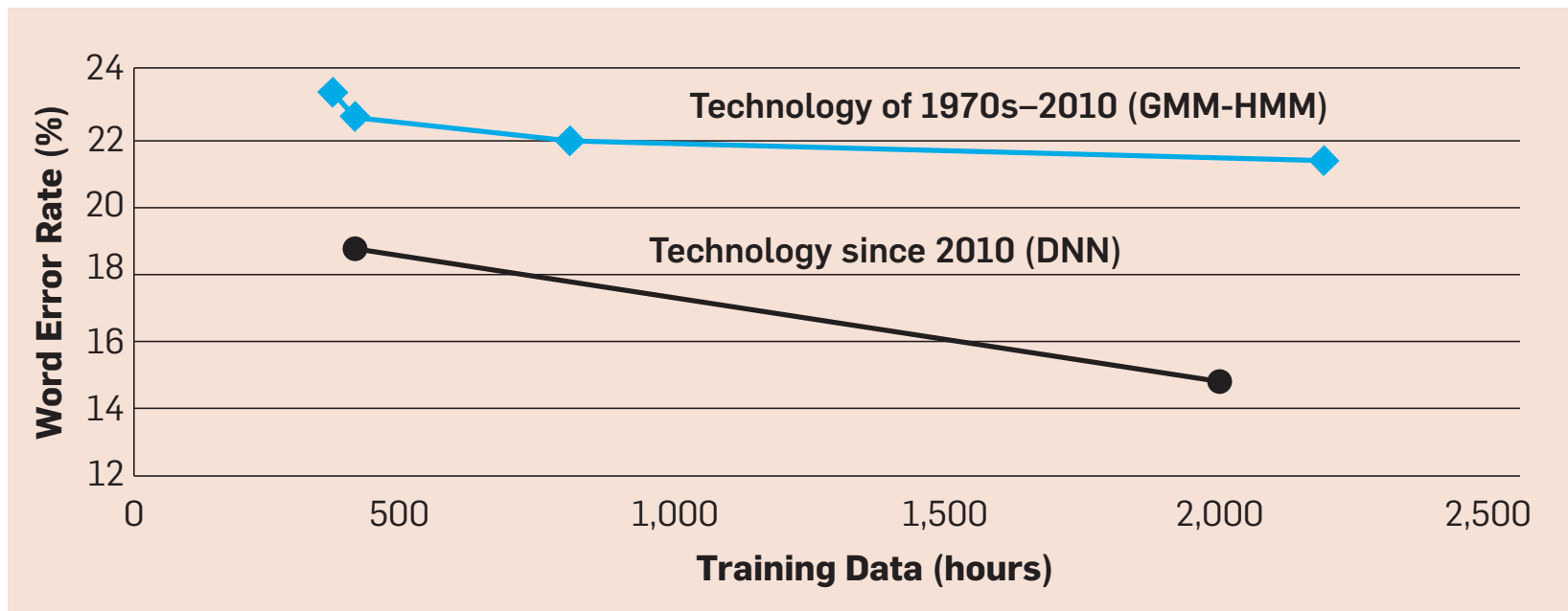
Mask R-CNN [He et al. 2017]

Mask R-CNN [He et al. 2017]

Speech Recognition

- Similar jump in performance seen with adaption of deep learning approaches

[A Historical Perspective of Speech Recognition, Huang, Baker & Reddy, CACM 2014]

Speech Recognition

- E.g. Baidu's Deep Speech 2 system [2015]
  - Input: spoken speech. Output: text
  - 100 million parameters; 11-layer Recurrent Neural Network model
  - English training set: 11,940 hours of labeled speech data containing 8 million utterances
  - Beats humans on 3 of 4 evaluation sets

Read Speech			
Test set	DS1	DS2	Human
WSJ eval'92	4.94	3.60	5.03
WSJ eval'93	6.94	4.98	8.08
LibriSpeech test-clean	7.89	5.33	5.83
LibriSpeech test-other	21.74	13.25	12.69

Table 13: Comparison of WER for two speech systems and human level performance on read speech.

[http://arxiv.org/pdf/1512.02595v1.pdf]

Natural Language Processing

- Recurrent Neural Networks
  - [Werbos 1988, Hochreiter & Schmidhuber 1997]
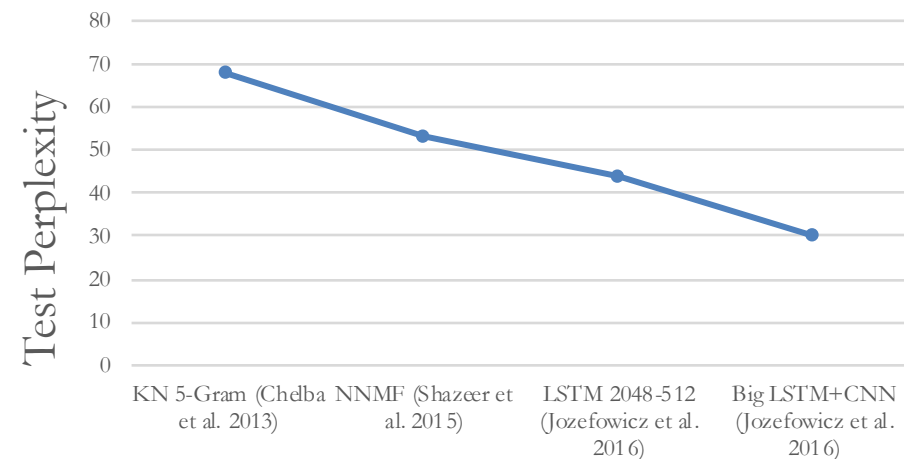- E.g. Machine Translation
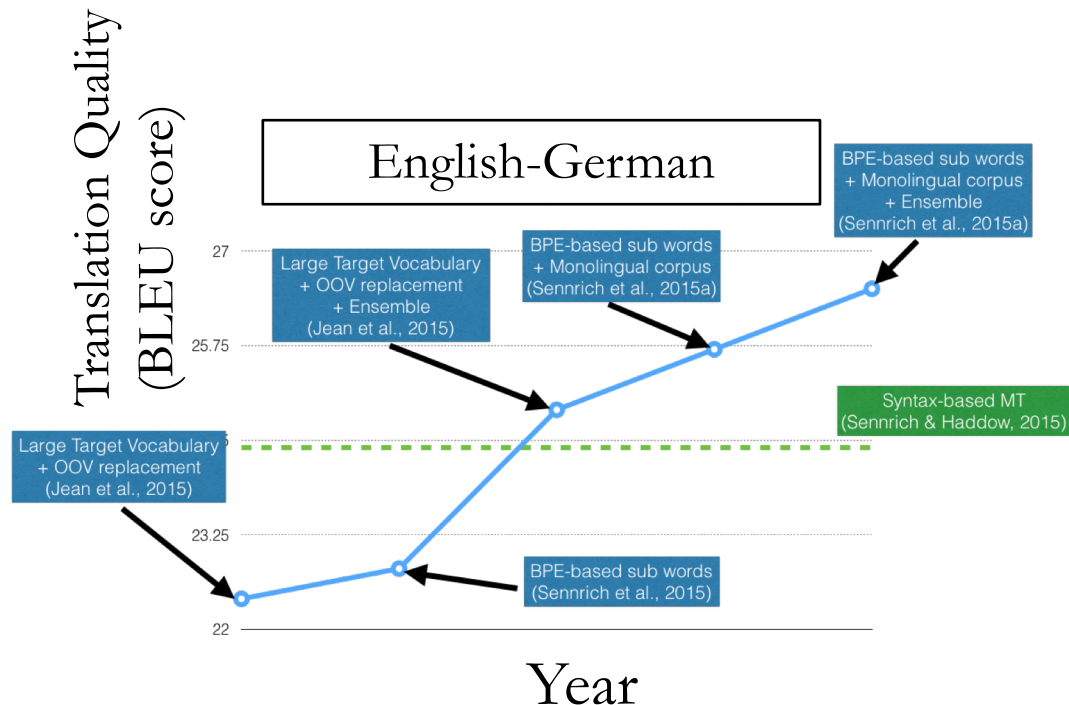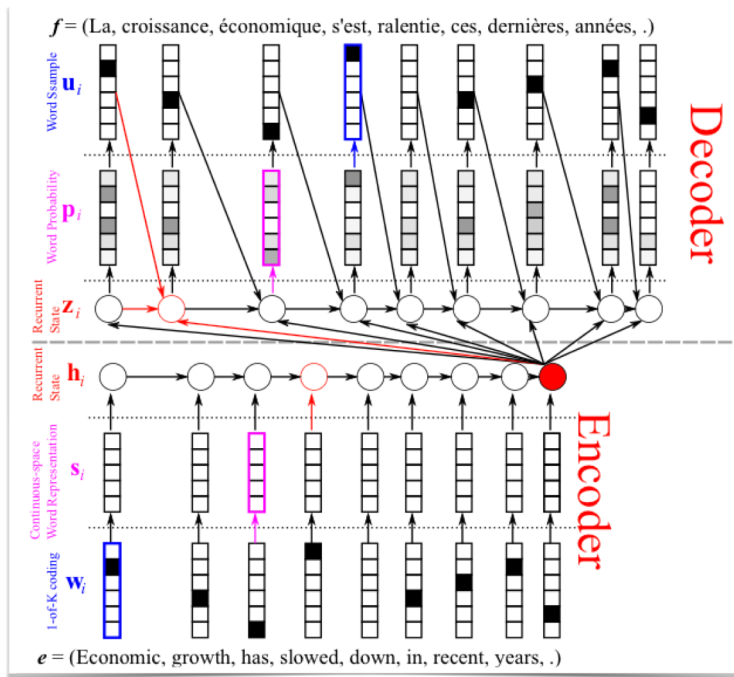
[Sutskever et al. 2014,
Cho et al. 2014, & others]

Natural Language Processing

- ## Recurrent Neural Networks
  - [Werbos 1988, Hochreiter & Schmidhuber 1997]

- ## E.g. Language modeling
  - Synthesize realistic text

Google 1B word dataset

Test Perplexity vs. method (KN 5-Gram (Chelba et al. 2013), NNMF (Shazeer et al. 2015), LSTM 2048-512 (Jozefowicz et al. 2016), Big LSTM+CNN (Jozefowicz et al. 2016))

Samples from [Jozefowicz et al. 2016]:

< S > With even more new technologies coming onto the market quickly during the past three years , an increasing number of companies now must tackle the ever-changing and ever-changing environmental challenges online . < S > Check back for updates on this breaking news story . < S > About 800 people gathered at Hever Castle on Long Beach from noon to 2pm , three to four times that of the funeral cortege .

Practical Applications

- Real-world machine translation

- Facebook serves 2B translations/day
  - 40 different languages
  - All using deep nets

Industrial Applications

- Internet Companies
  - Facebook, Google, Amazon etc..

- E.g. Facebook
  - 1B+ images/day uploaded
  - Each passed through 2 deep nets
    - Object recognition / offensive content
    - Face recognition
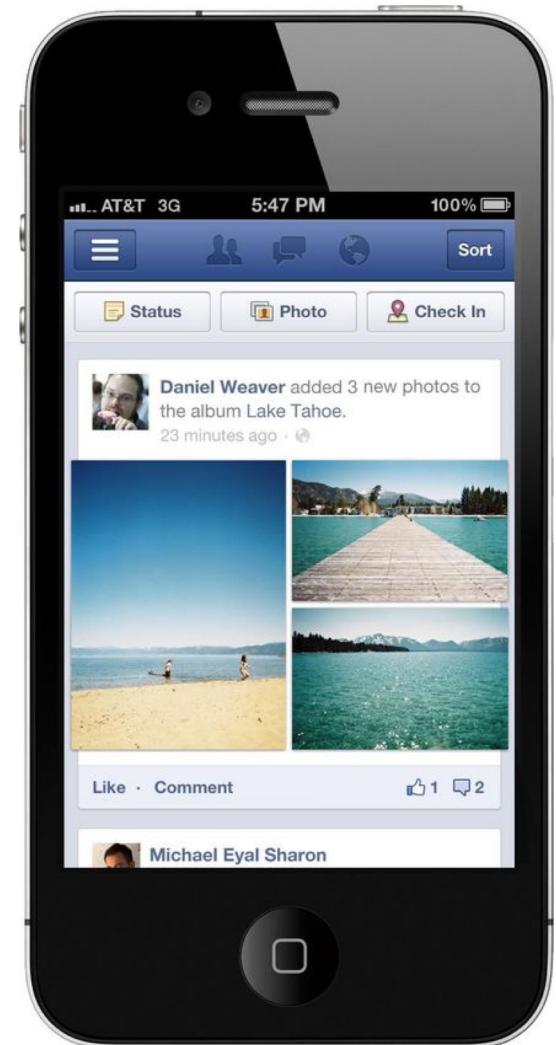
Practical Applications

- Speech recognition on your smartphone

Practical Applications

- Self-driving cars

Scientific Applications
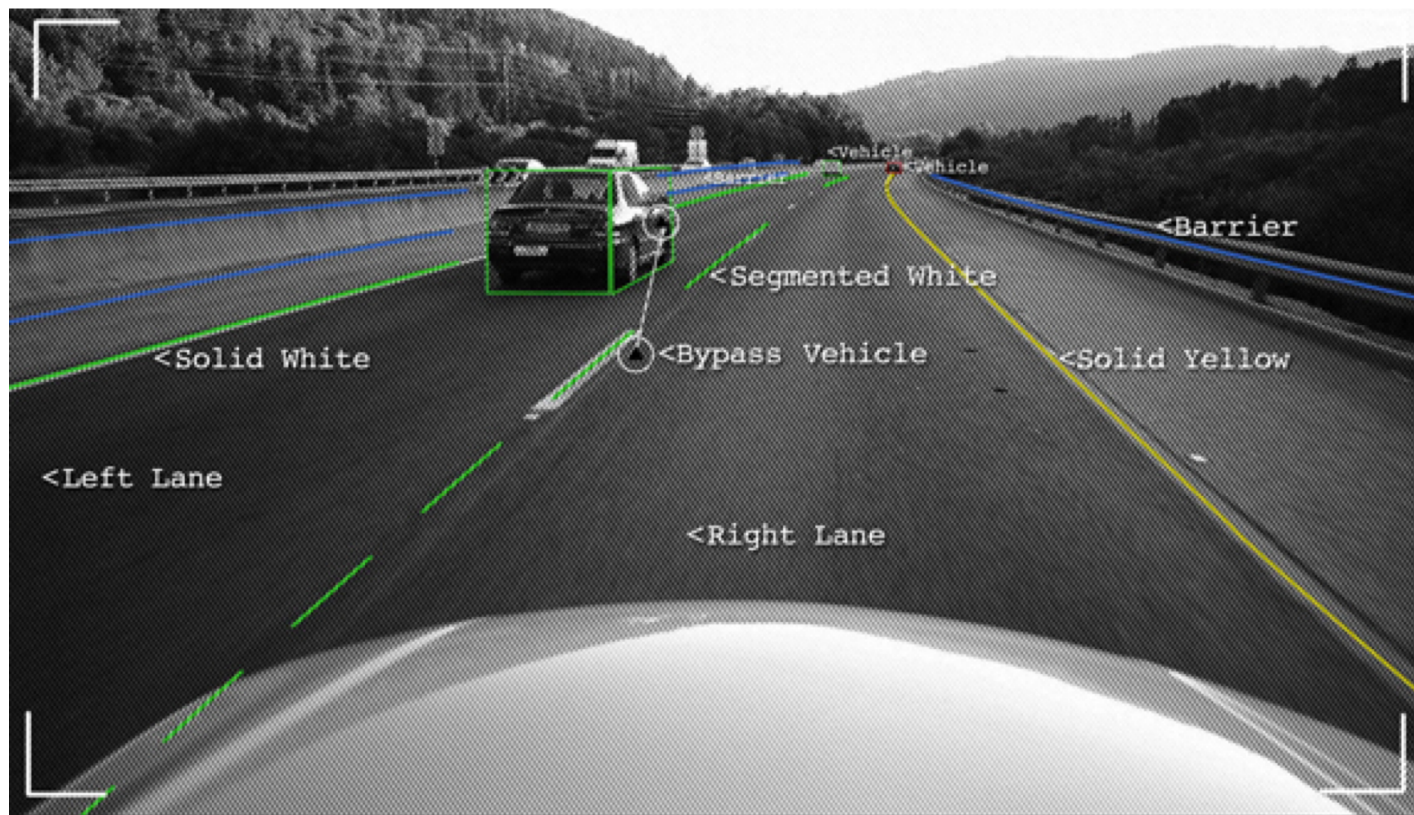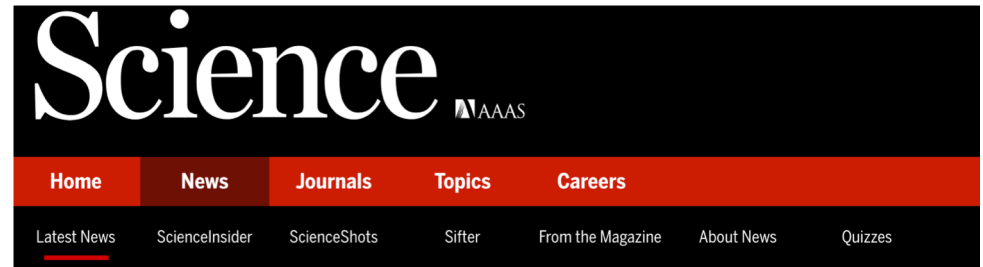
- Particle Physics

- Astronomy

- Chemistry

- Genomics

- Medicine

AI is changing how we do science. Get a glimpse

By Science News Staff | Jul. 5, 2017 , 11:00 AM

AI's early proving ground: the hunt for new particles

Particle physicists began fiddling with artificial intelligence (AI) in the late 1980s, just as the term "neural network" captured the public's imagination. Their field lends itself to AI and machine-learning algorithms because nearly every experiment centers on finding subtle spatial patterns in the countless, similar readouts of complex particle detectors—just the sort of thing at which AI excels. "It took us several years to convince people that this is not just some magic, hocus-pocus, black box stuff," says Boaz Klima, of Fermi National Accelerator Laboratory (Fermilab) in Batavia, Illinois, one of the first physicists to embrace the techniques. Now, AI techniques number among physicists' standard tools.

Particle physicists strive to understand the inner workings of the universe by smashing subatomic particles together with enormous energies to blast out exotic new bits of matter. In 2012, for example, teams working with the world's largest proton collider, the Large Hadron Collider (LHC) in Switzerland, discovered the long-predicted Higgs boson, the fleeting particle that is the linchpin to physicists' explanation of how all other fundamental particles get their mass.
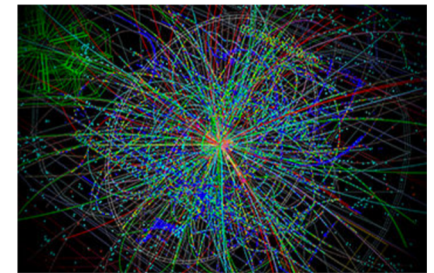
Neural networks search for fingerprints of new particles in the debris of collisions at

© 2012 CERN, FOR THE BENEFIT

Skin Cancer Classification

[Dermatologist-level classification of skin cancer with deep neural networks, Esteva, A. et al., Nature 2017]

Diabetic Retinopathy

[Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photography, Gulshan, V. et al. JAMA 2016]
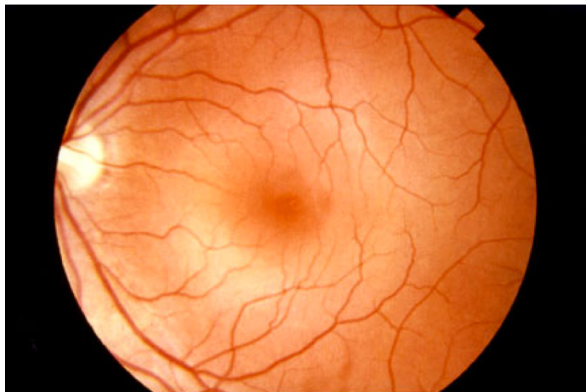
NORMAL MACULA
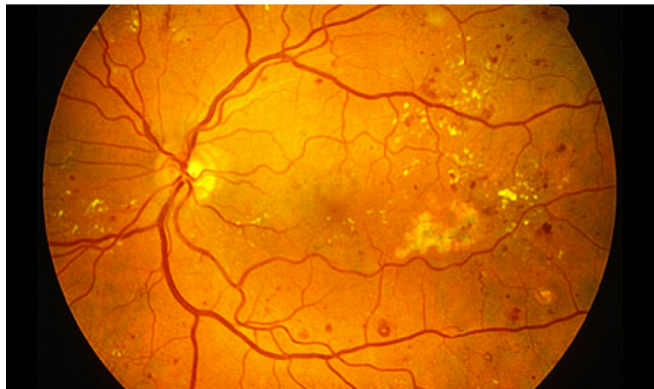
NPDR WITH RETINAL HEMORRHAGES
AND HARD EXUDATES

Figure 2. Validation Set Performance for Referable Diabetic Retinopathy

A EyePACS-1: AUC, 99.1%; 95% CI, 98.8%-99.3%
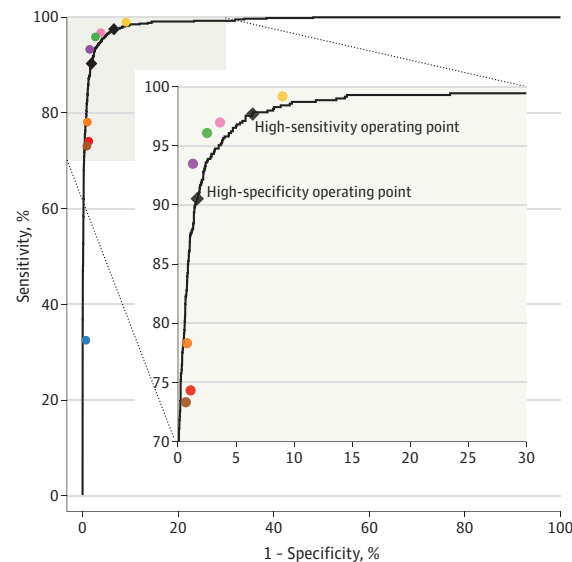
B Messidor-2: AUC, 99.0%; 95% CI, 98.6%-99.5%
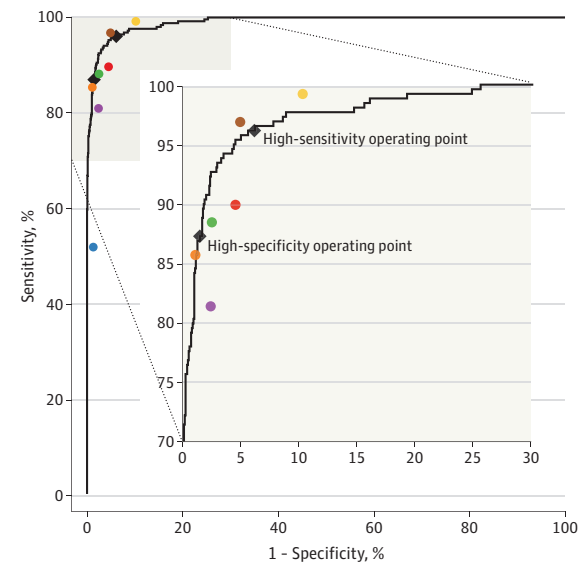
Some Issues with Deep Learning

- No good theoretical understanding or performance guarantees
  - Hard to analyze: very high dimensional, highly non-convex

- Difficult to inspect models
  - Cannot understand why certain output was produced
  - Cf. Probabilistic graphical models
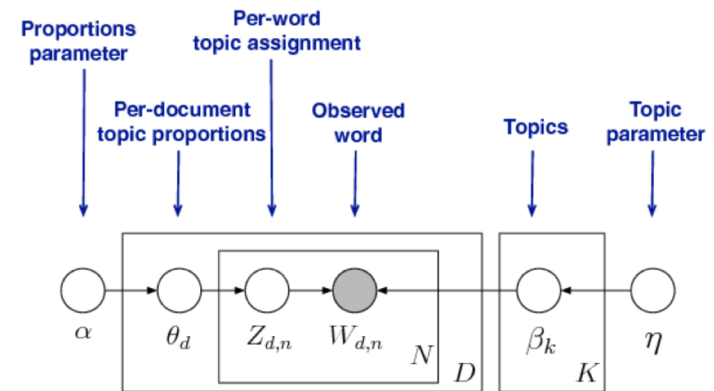
[Blei et al. 2003]

- Need lots of labeled data
  - Not always possible to obtain

Importance of Model Architecture

- Previously: Hand-design the feature representation
- Deep Learning: Learn the features but still need to hand-design the model architecture
  - Attempts to meta-learn it, e.g. [Neural architecture search with reinforcement learning, Zoph & Le, arXiv 1611.01578, 2016].

- Deep nets with generic structure (i.e. fully connected) do not work
- Architecture of network has to be appropriate to domain
  - E.g. for images, exploit 2D grid, local dependencies etc.

Samoyed (16); Papillon (5.7); Pomeranian (2.7); Arctic fox (1.0); Eskimo dog (0.6); white wolf (0.4); Siberian husky (0.4)

Convolutions and ReLU

Max pooling

Convolutions and ReLU

Max pooling

Convolutions and ReLU

Red   Green   Blue
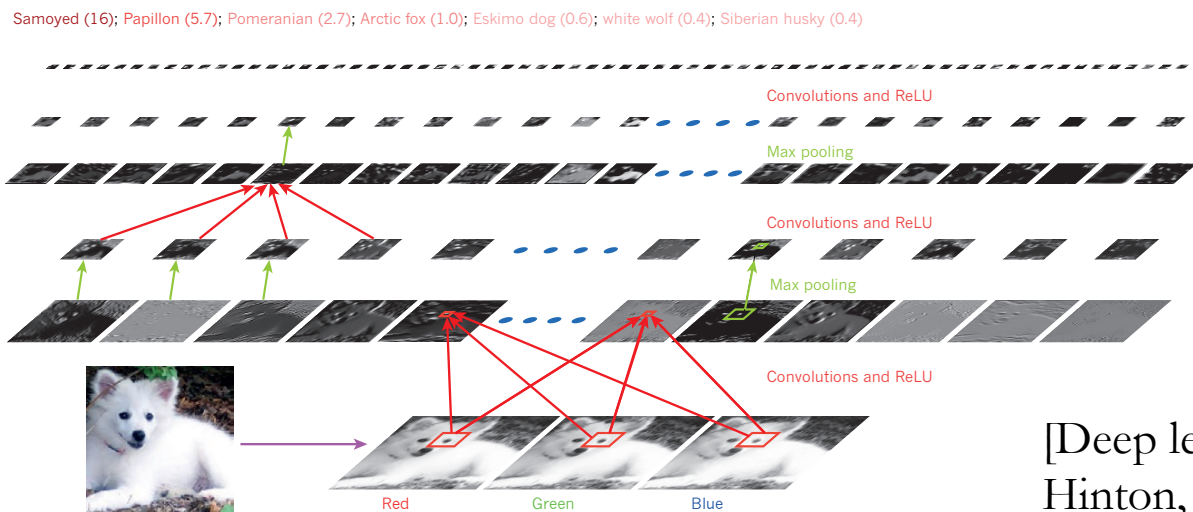
[Deep learning, LeCun, Bengio, Hinton, Nature 2015]

Convolutional Neural Networks

- LeCun et al. 1989
- Neural network with specialized connectivity structure
- Can view as a multi-scale Hubel-Wiesel architecture
  - Alternating layer of: simple cells (filtering) and complex cells (averaging)
  - Higher stages compute more global, more invariant features
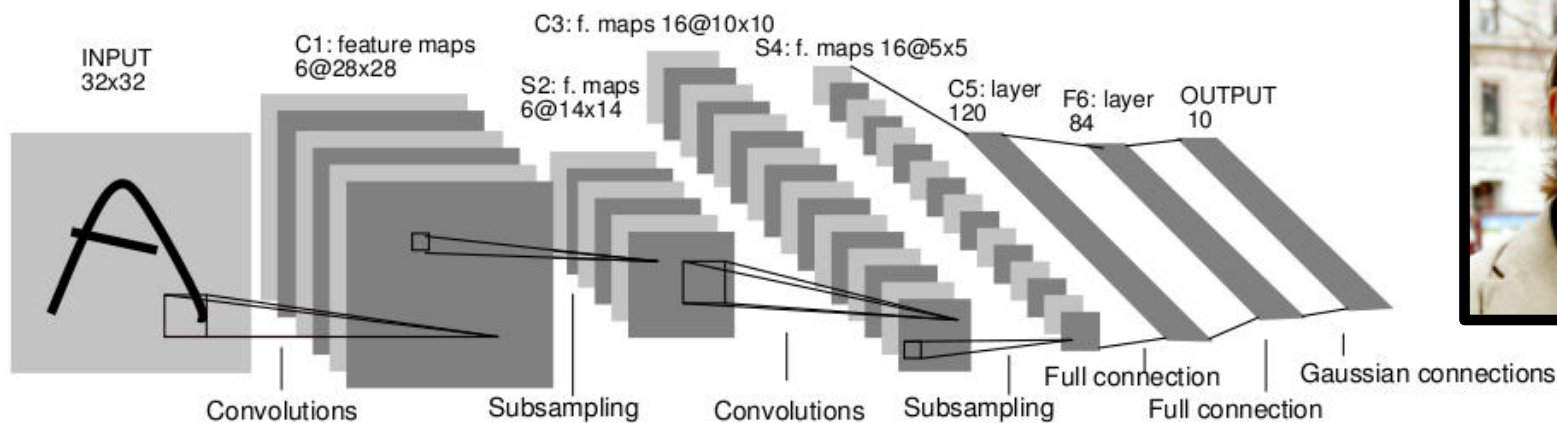
ConvNet Architecture

- Exploits two properties of images:

- 1. Dependencies are local
  - No need to have each unit connect to every pixel
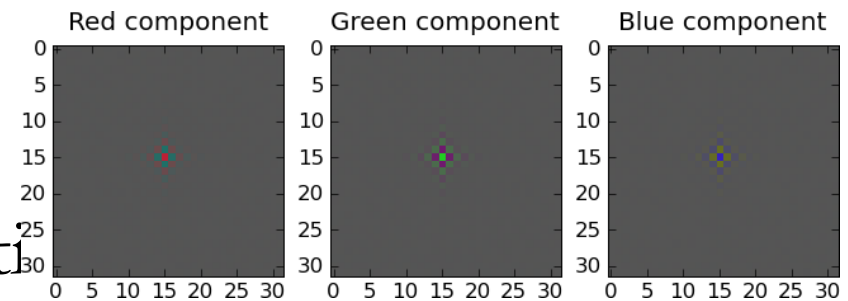
- 2. Spatially stationary statistics
  - Translation invariant dependencies
  - Only approximately true

Multistage Hubel-Wiesel Architecture

- Stack multiple stages of simple cells / complex cells layers
- Higher stages compute more global, more invariant features
- Classification layer on top

History:

- Neocognitron [Fukushima 1971-1982]
- Convolutional Nets [LeCun 1988-200
- HMAX [Poggio 2002-2006]
- Many others….

Components of Each Layer

Pixels / Features →

Filter with learned dictionary

↓
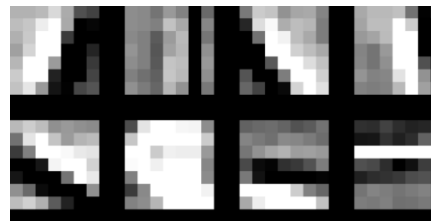
Non-linearity

↓
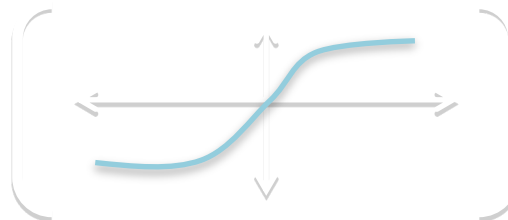
[Optional] Spatial local pooling

$p_{1,1}$ $p_{2,1}$

$z_{1,1}$ $z_{4,1}$

→
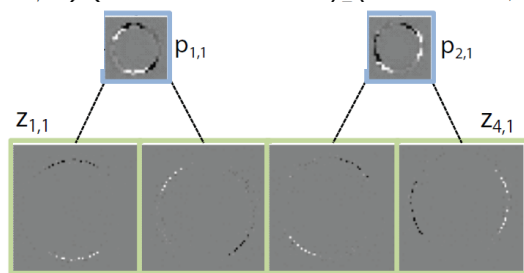
[Optional] Normalization across data/features

→ Output Features

Filtering

- Convolution
  - Filter is learned during training
  - Same filter at each location
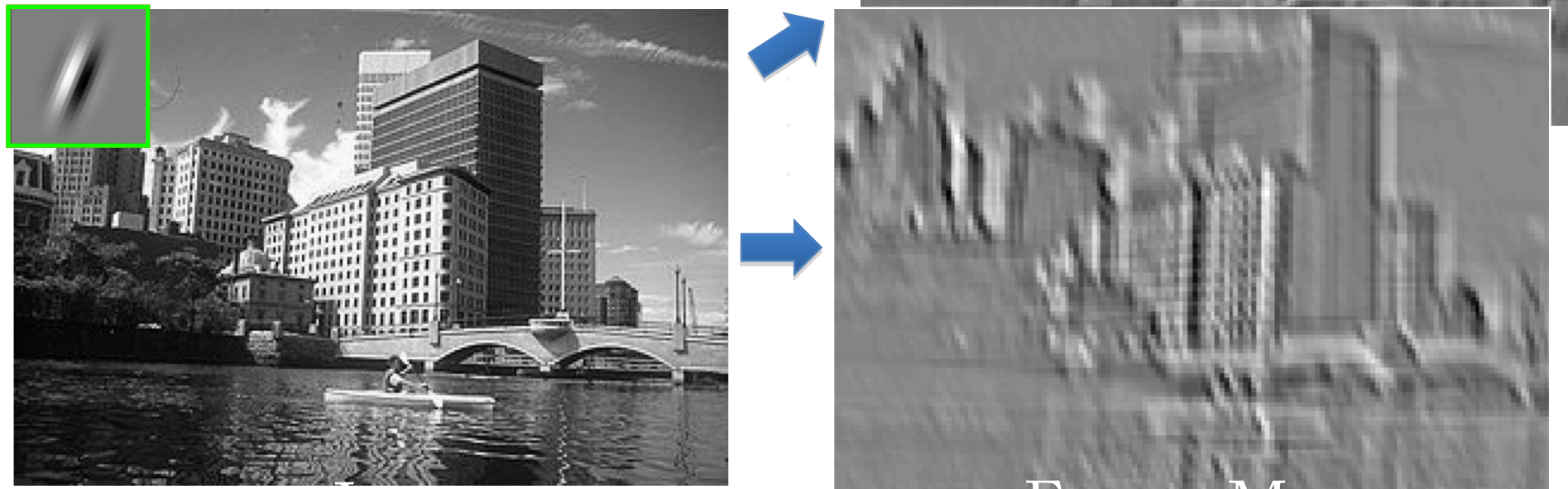
Non-Linearity

- Rectified linear function
  - Applied per-pixel
  - output = max(0,input)

Black = negative; white = positive values

Only non-negative values
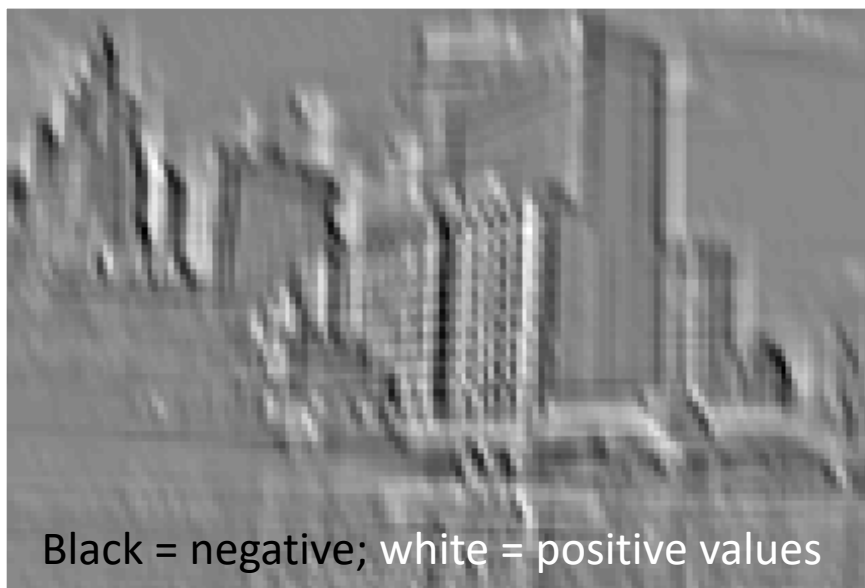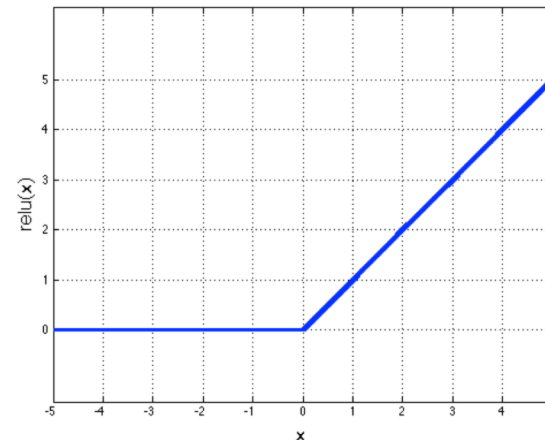
Non-Linearity

- Traditional options:
  - Tanh
  - Sigmoid: 1/(1+exp(-x))
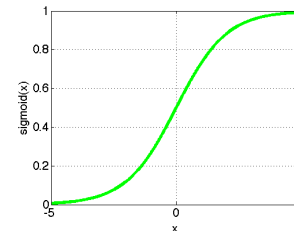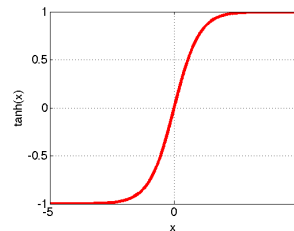
- More recent ones:
  - Leaky ReLU, ELU, SELU, PReLU

$$f(y_i) = \begin{cases} y_i, & \text{if } y_i > 0 \\ a_i y_i, & \text{if } y_i \leq 0 \end{cases}.$$

$$\text{selu}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leqslant 0 \end{cases}.$$
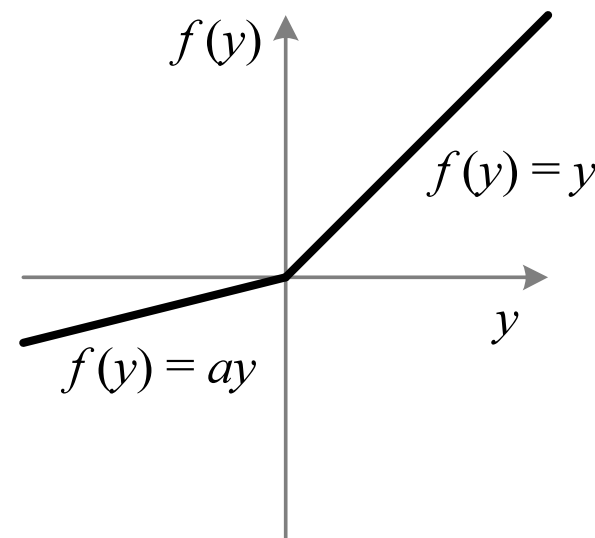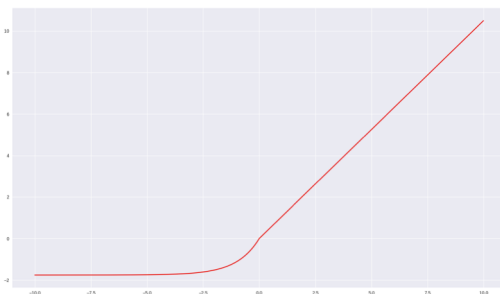
[https://towardsdatascience.com/selu-make-fnns-great-again-snn-8d61526802a9]
[Self-Normalizing Neural Networks, Klambauer et al. arXiv:1706.02515.pdf, Sept 2017]

[Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, Kaiming He et al. arXiv:1502.01852v1.pdf, Feb 2015]

Pooling

- Spatial Pooling
  - Non-overlapping / overlapping regions
  - Sum or max
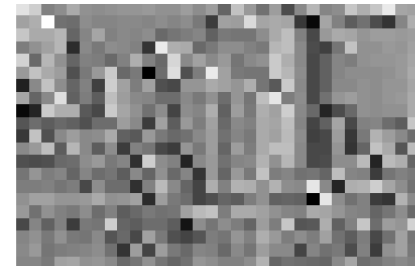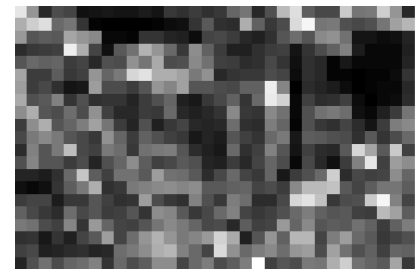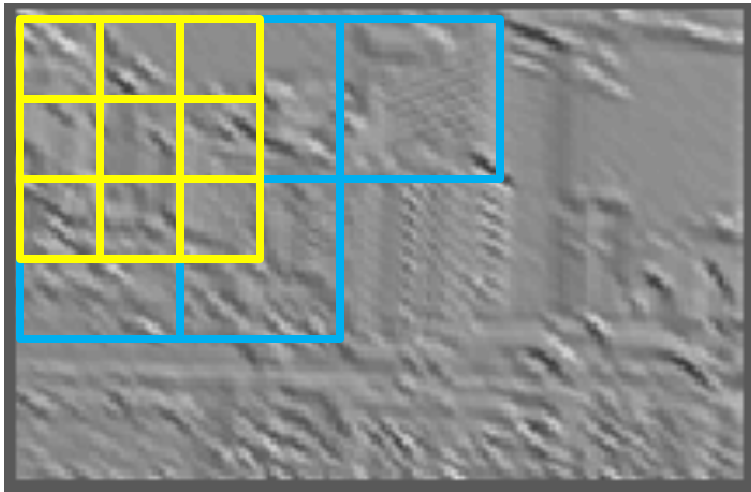
Batch Normalization (BN)

- Recap: Normalizing image input (LeCun et al 1998 "Efficient Backprop")

- BN: data-driven normalization, for each layer, for each mini-batch
  - Greatly accelerate training
  - Less sensitive to initialization
  - Improve regularization

Ioffe & Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". ICML 2015

Batch Normalization (BN)

$$x \quad \blacktriangleright \quad \hat{x} = \frac{x - \mu}{\sigma} \quad \blacktriangleright \quad y = \gamma \hat{x} + \beta$$

- μ: mean of x in mini-batch
- σ: std of x in mini-batch
- γ: scale
- β: shift

- μ, σ: functions of x, analogous to responses
- γ, β: parameters to be learned, analogous to weights

Ioffe & Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". ICML 2015

Batch Normalization (BN)

$$x \ \blacktriangleright \ \hat{x} = \frac{x - \mu}{\sigma} \ \blacktriangleright \ y = \gamma \hat{x} + \beta$$

2 modes of BN:
- Train mode:
  - μ, σ are functions of a batch of x
- Test mode:
  - μ, σ are pre-computed on training set

Caution: make sure your BN usage is correct!
(this causes many of my bugs in my research experience!)

Ioffe & Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". ICML 2015

Batch Normalization (BN)
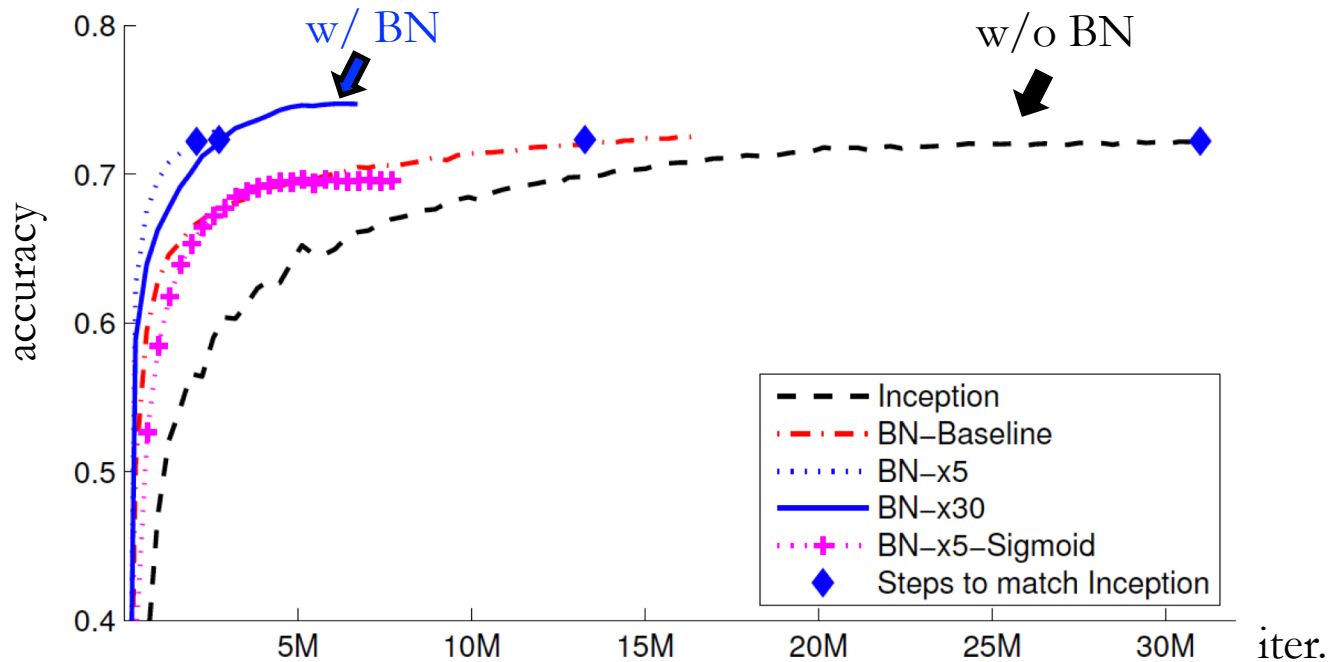
Figure credit: Ioffe & Szegedy

Ioffe & Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". ICML 2015

Summary of Convnet Model

- Feed-forward:
  - Convolve input
  - Non-linearity (rectified linear)
  - [Optional] Pooling (local max)
  - [Optional] Batch Normalization
- Fully-connected classifier layer at top
- Supervised loss function (uses image label)
- Train convolutional filters by back-propagating classification error

Feature maps

⬆

Pooling

⬆

Non-linearity

⬆

Convolution (Learned)

⬆

Input Image

INPUT 32x32   C1: feature maps 6@28x28   C3: f. maps 16@10x10   S4: f. maps 16@5x5   C5: layer 120   F6: layer 84   OUTPUT 10
S2: f. maps 6@14x14

Convolutions   Subsampling   Convolutions   Subsampling   Full connection   Gaussian connections
Full connection

LeCun et al. 1998
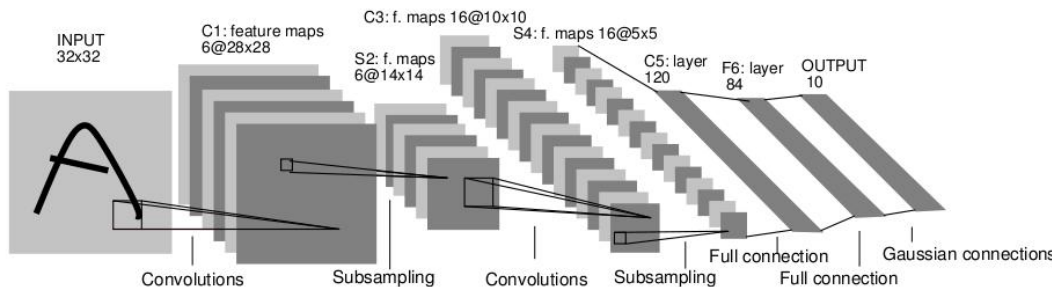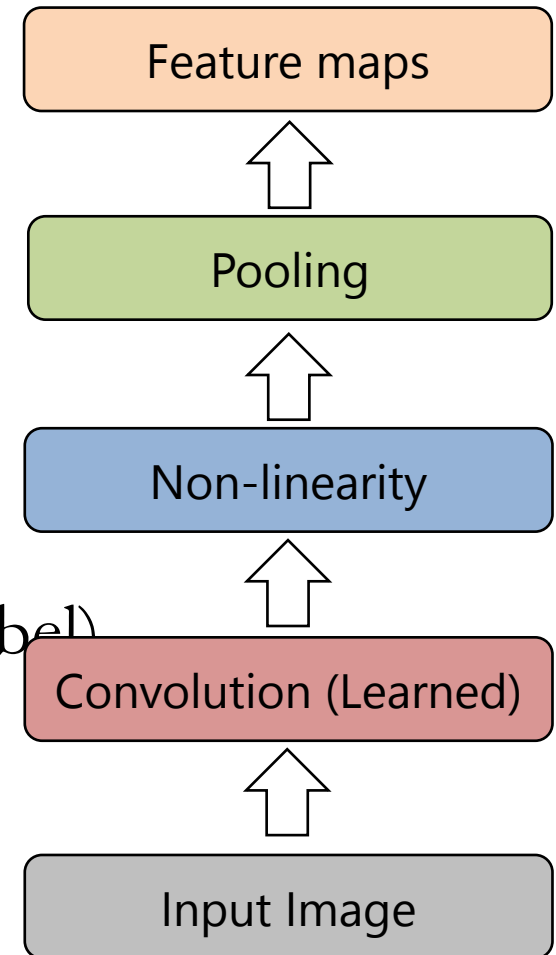
Training

- Many parameters: $O(10^6+)$
  - 2nd order methods not practical (Hessian too big)
- Big datasets: $O(10^6)$
  - Expensive to compute full objective, i.e. loss on all examples

- Use 1st order methods and update using subset of examples
  - Pick random batch at each iteration

Stochastic Gradient Descent (SGD)

$$\Delta_t = \mu \Delta_{t-1} - \eta \nabla L_t(\theta_t)$$
$$\theta_{t+1} = \theta_t + \Delta_t$$

- Fixed learning rate η
  - Large as possible without being unstable, e.g. 0.01
- Momentum term μ
  - Typically ~0.9
  - Smooths updates → helps convergence
  - Also Nesterov version: apply momentum before gradient

Annealing of Learning Rate

- Start large, slowly reduce when the training error stops decreasing

- Explore different scales of energy surface
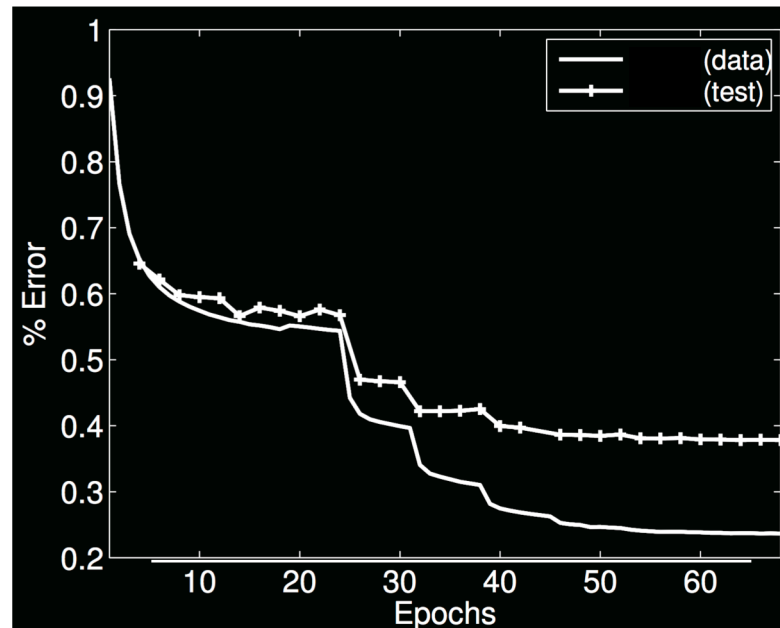
AdaGrad

- Learning rate now scaled per-dimension
- Decreased for dimensions with high variance
- Issue: learning rate monotonically decreases
  - Stop making progress after while

$$\theta_{t+1} = \theta_t - \eta \frac{\nabla L_t(\theta_t)}{\sqrt{\sum_{t'=1}^{t} \nabla L_{t'}(\theta_{t'})^2}}$$

[Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, Duchi et al., JMLR 2011]

RMSProp

- Similar to AdaGrad, but now with moving average
  - Small μ emphasizes recent gradients

$$\Delta_t = \mu \Delta_{t-1} + (1 - \mu) \nabla L_t(\theta_t)^2$$

$$\theta_{t+1} = \theta_t - \eta \frac{\nabla L_t(\theta_t)}{\sqrt{\Delta_t}}$$

ADAM

- ADApative Moment Estimation
- Combines AdaGrad and RMSProp

- Idea: maintain moving averages of gradient and gradient^2
- Update $\propto \dfrac{\text{Mean gradient}}{\sqrt{\text{Mean gradient}^2}}$

For more details, see:
https://moodle2.cs.huji.ac.il/nu15/pluginfile.php/316969/mod_resource/content/1/adam_pres.pdf

[Adam: A Method for Stochastic Optimization, Kingma & Ba, arXiv:1412.6980]

Batch-size

- [Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour, Goyal et al., arXiv 1706.02677, 2017]

- Scale learning rate with batch-size

- Large-batch size efficiently implemented via synchronous parallel training
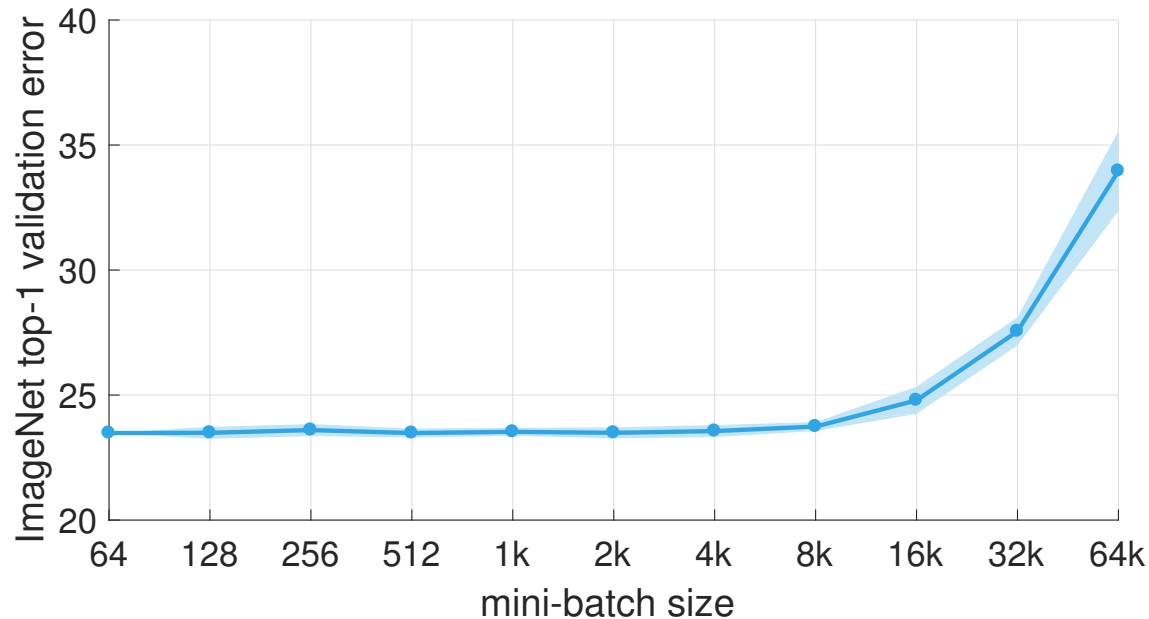
Some Practical Debugging Tips

- Train on small subset of data
    - Train error should → 0.
    - If not, check data (& pre-processing) and size of model.

- Training diverges:
    - Learning rate may be too large → decrease learning rate.
    - BPROP is buggy → numerical gradient checking.

- Parameters collapse / loss is minimized but train accuracy is low
    - Check loss function:
        - Is it appropriate for the task you want to solve?
        - Does it have degenerate solutions? Check "pull-up" term.

- Model is underperforming
    - Compute flops and nr. params. → if too small, make net larger
    - Visualize hidden units/params → fix optimization

- Model is too slow
    - Compute flops and nr. params. → GPU,distrib. framework, make net smaller

[M. Ranzato]

Convolutional Network Layer 1 Filters
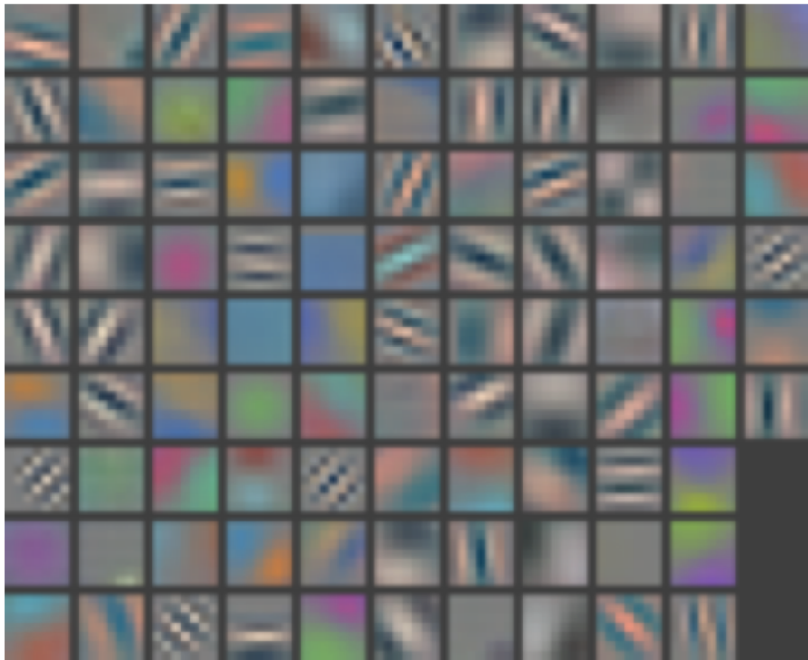
V1 physiology: orientation selectivity

Hubel & Wiesel, 1968
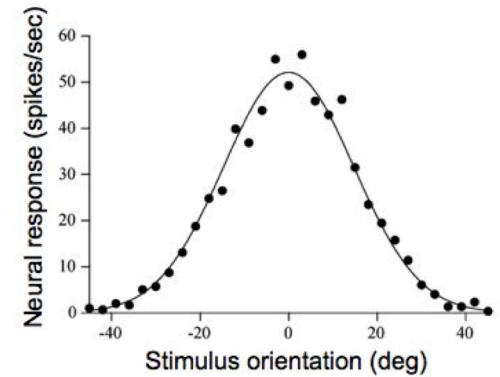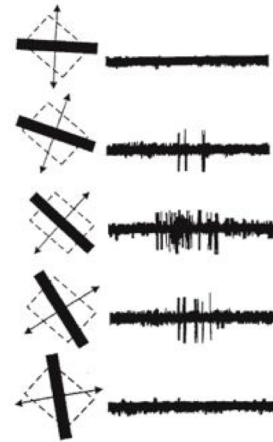
Layer 1: Patches that give largest activations

Layer 2

Layer 3

Layer 5
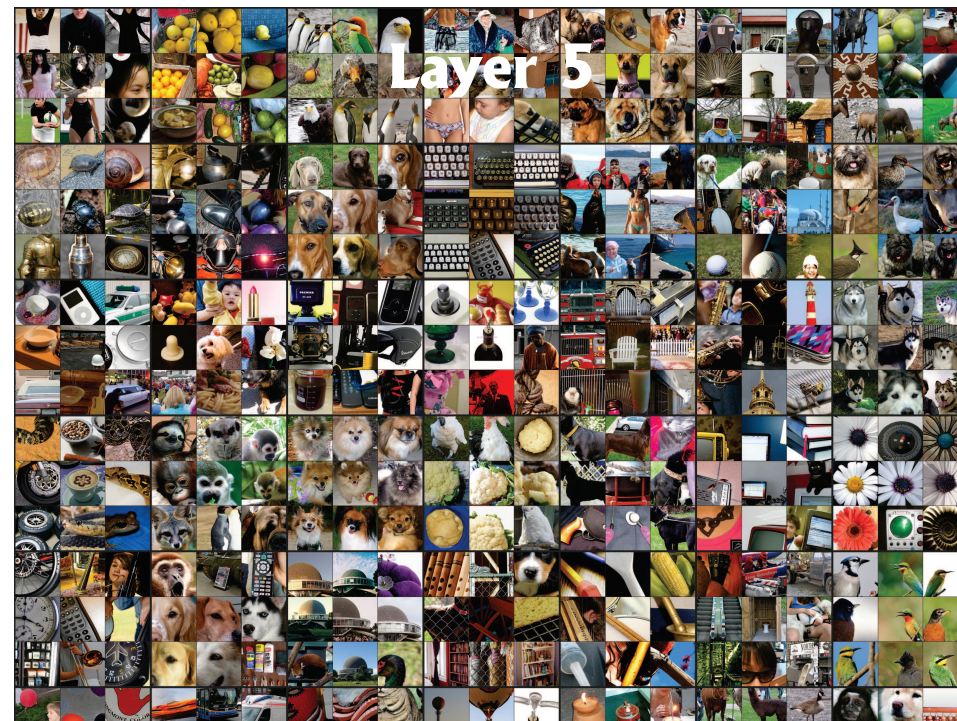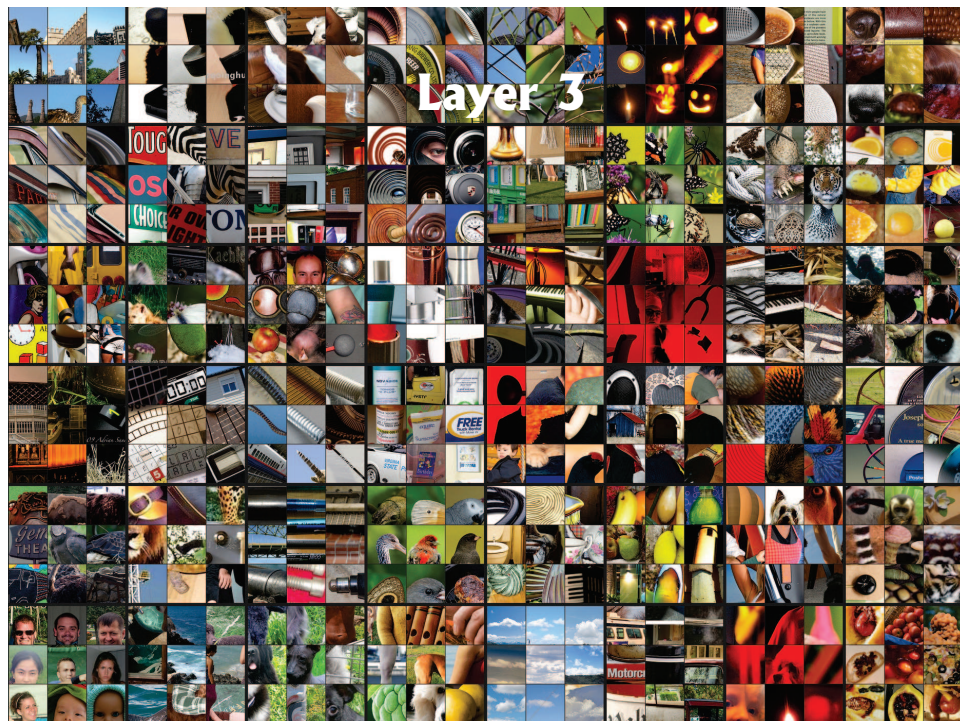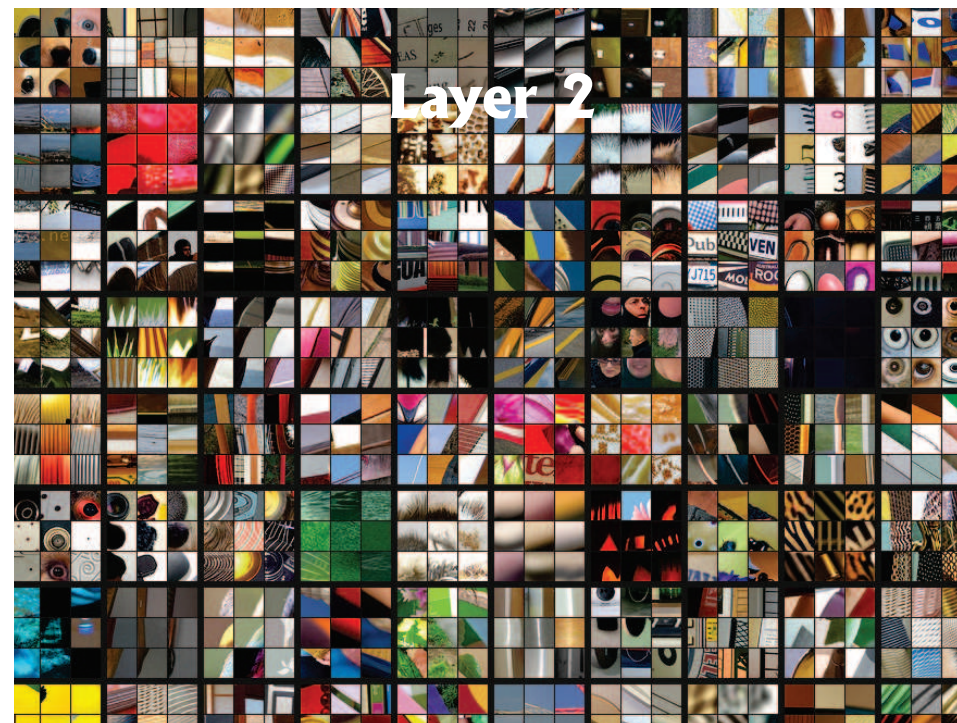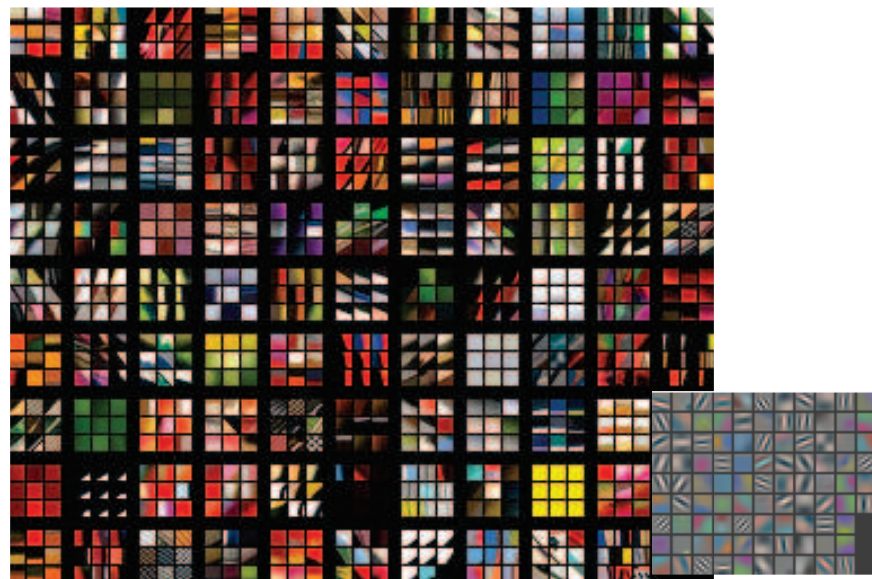
Initialization

Both forward (response) and backward (gradient) signal can vanish/explode

Forward:

$$Var[y] = (\prod_d n_d^{in} Var[w_d])Var[x]$$

Backward:

$$Var\left[\frac{\partial}{\partial x}\right] = (\prod_d n_d^{out} Var[w_d])Var[\frac{\partial}{\partial y}]$$
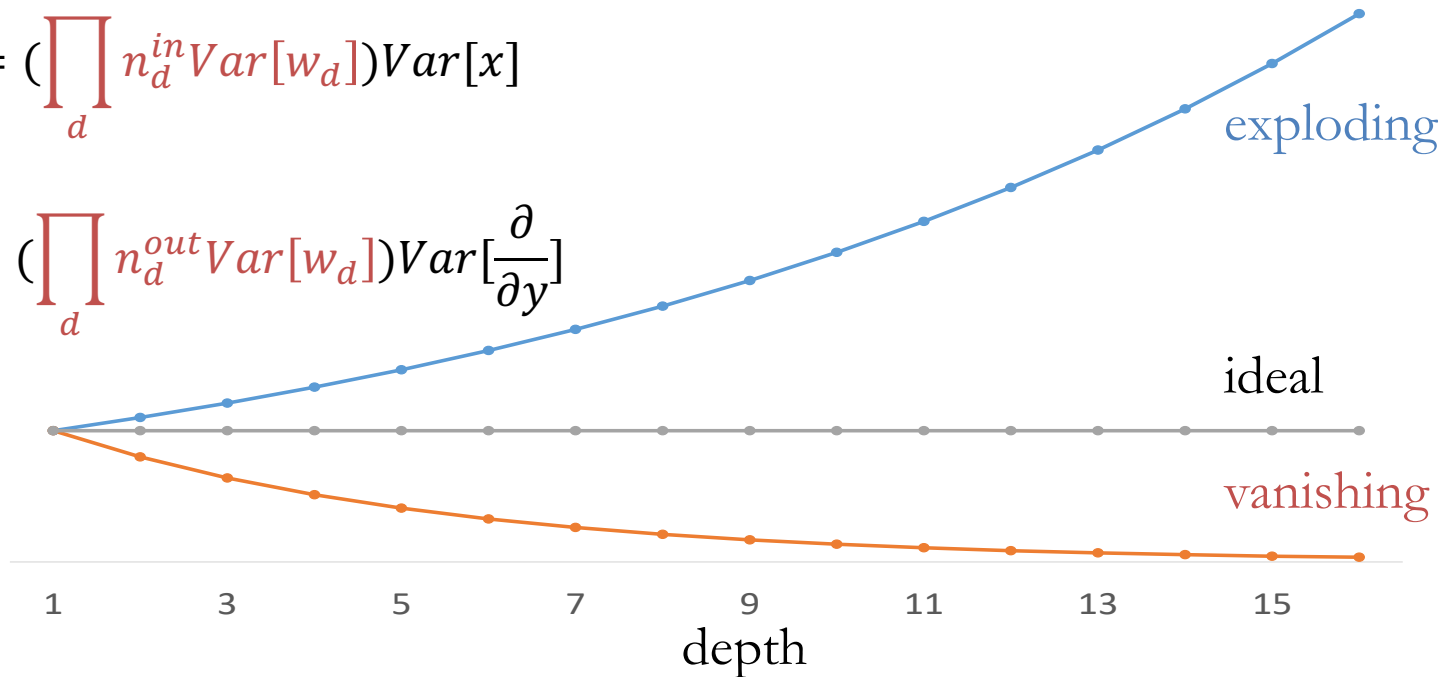
exploding

ideal

vanishing

depth

LeCun et al 1998 "Efficient Backprop"
Glorot & Bengio 2010 "Understanding the difficulty of training deep feedforward neural networks"

[Slide: Kaiming He]

Initialization

- Initialization under **linear** assumption

$$\prod_d n_d^{in} Var[w_d] = const_{\text{fw}} \text{ (healthy forward)}$$

$$\text{and}$$

$$\prod_d n_d^{out} Var[w_d] = const_{\text{bw}} \text{ (healthy backward)}$$

➡ $\boxed{\begin{array}{c} n_d^{in} Var[w_d] = 1 \\ \text{or*} \\ n_d^{out} Var[w_d] = 1 \end{array}}$

"*Xavier*" init in Caffe

*: $n_d^{out} = n_{d+1}^{in}$, so $\frac{const_{\text{bw}}}{const_{\text{fw}}} = \frac{n_{\text{last}}^{out}}{n_{\text{first}}^{in}} < \infty$.

It is sufficient to use either form.

LeCun et al 1998 "Efficient Backprop"
Glorot & Bengio 2010 "Understanding the difficulty of training deep feedforward neural networks"

[Slide: Kaiming He]

Initialization

- Initialization under **ReLU**

$$\prod_d \frac{1}{2} n_d^{in} Var[w_d] = const_{\text{fw}} \text{ (healthy forward)}$$

and

$$\prod_d \frac{1}{2} n_d^{out} Var[w_d] = const_{\text{bw}} \text{(healthy backward)}$$

➡

$$\frac{1}{2} n_d^{in} Var[w_d] = 1$$

or

$$\frac{1}{2} n_d^{out} Var[w_d] = 1$$

"*MSRA*" init in Caffe

With D layers, a factor of 2 per layer has exponential impact of 2^D

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". ICCV 2015.

[Slide: Kaiming He]

Initialization

22-layer ReLU net:
good init converges faster

30-layer ReLU net:
good init is able to converge
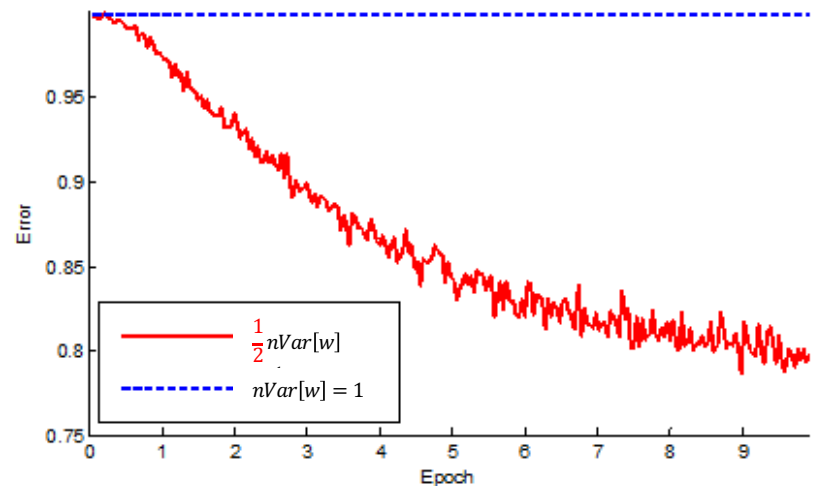
*Figures show the beginning of training
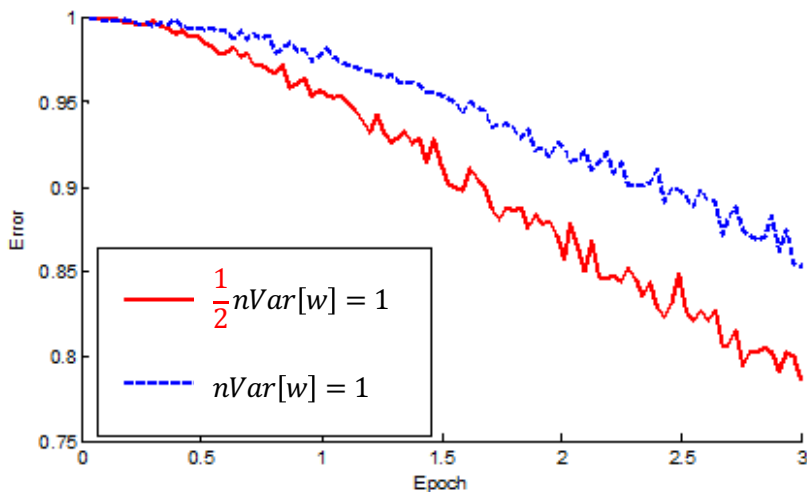
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". ICCV 2015.

[Slide: Kaiming He]

Simply stacking layers?

CIFAR-10
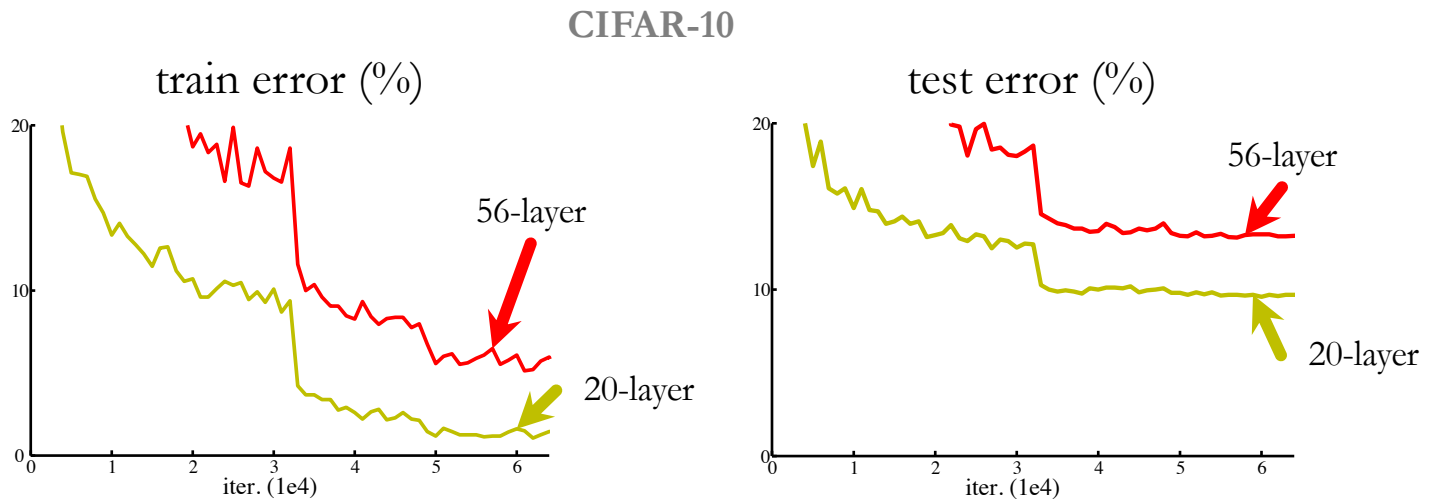
- *Plain* nets: stacking 3x3 conv layers…
- 56-layer net has **higher training error** and test error than 20-layer net

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

[Slide: Kaiming He]

Simply stacking layers?

CIFAR-10

56-layer
44-layer
32-layer
20-layer

plain-20
plain-32
plain-44
plain-56

solid: test/val
dashed: train

ImageNet-1000

34-layer

18-layer

plain-18
plain-34
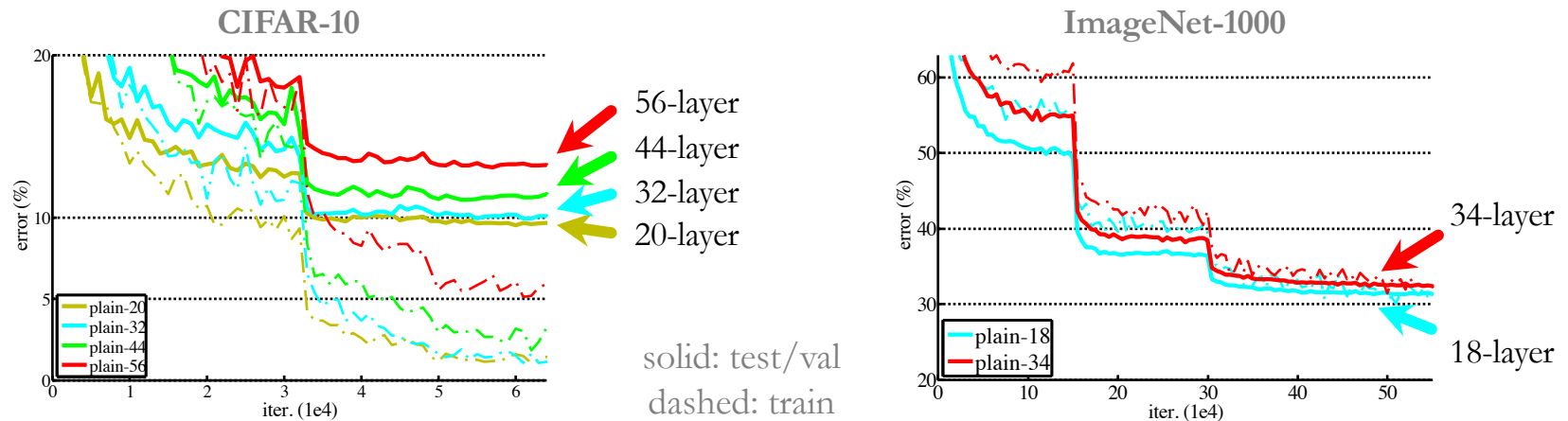
- "Overly deep" plain nets have **higher training error**
- A general phenomenon, observed in many datasets

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

[Slide: Kaiming He]

a shallower model (18 layers)

a deeper counterpart (34 layers)
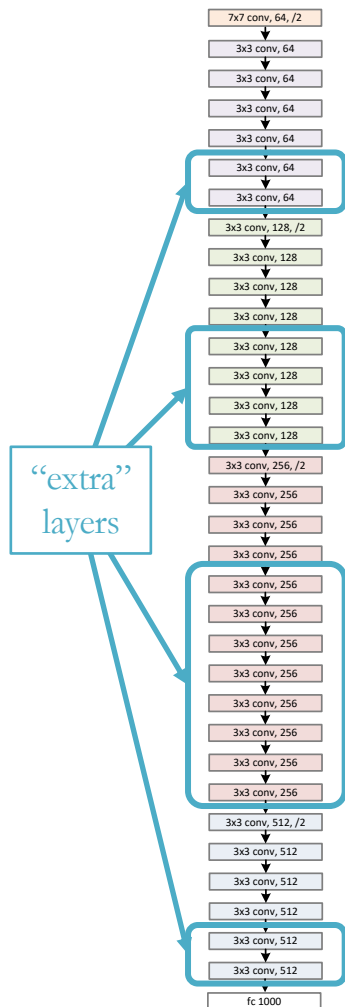
"extra" layers

- Richer solution space

- A deeper model should not have **higher training error**
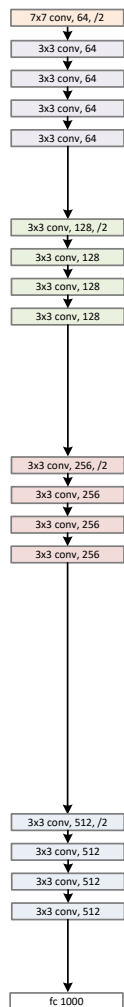
- A solution *by construction*:
  - original layers: copied from a learned shallower model
  - extra layers: set as identity
  - at least the same training error

- Optimization difficulties: solvers cannot find the solution when going deeper…

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

[Slide: Kaiming He]

Deep Residual Learning

- Plain net

x

weight layer

any small
subnet

relu

weight layer

relu

$H(x)$
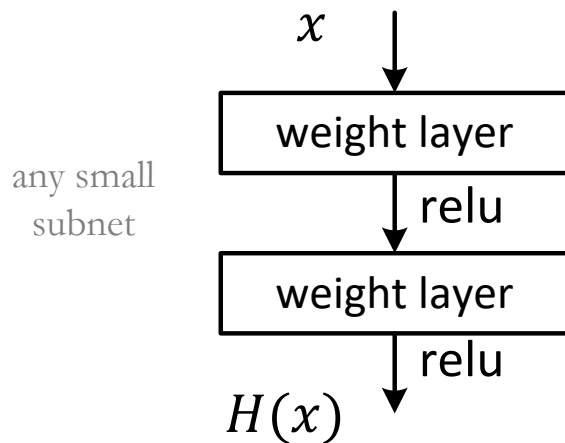
$H(x)$ is any desired mapping,

hope the small subnet fit $H(x)$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

[Slide: Kaiming He]

Deep Residual Learning

- Residual net

x

weight layer

relu

weight layer

$F(x)$

identity

x

$H(x) = F(x) + x$ ⊕

relu
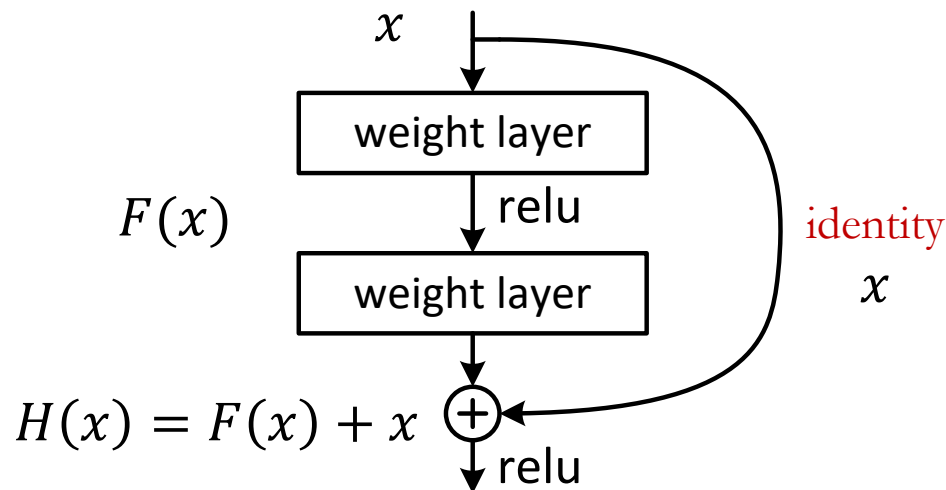
$H(x)$ is any desired mapping,

~~hope the small subnet fit $H(x)$~~

hope the small subnet fit $F(x)$

let $H(x) = F(x) + x$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

[Slide: Kaiming He]

Deep Residual Learning

- $F(x)$ is a residual mapping w.r.t. identity

- If identity were optimal, easy to set weights as 0

- If optimal mapping is closer to identity, easier to find small fluctuations

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.
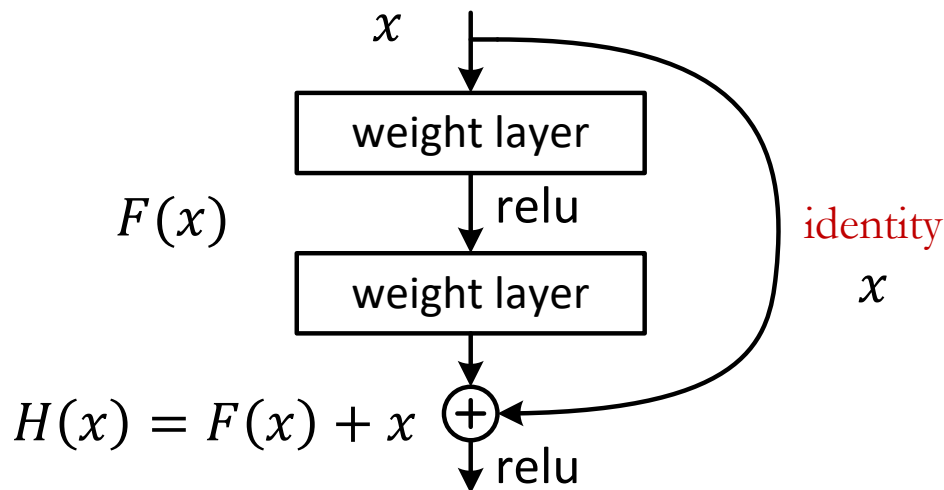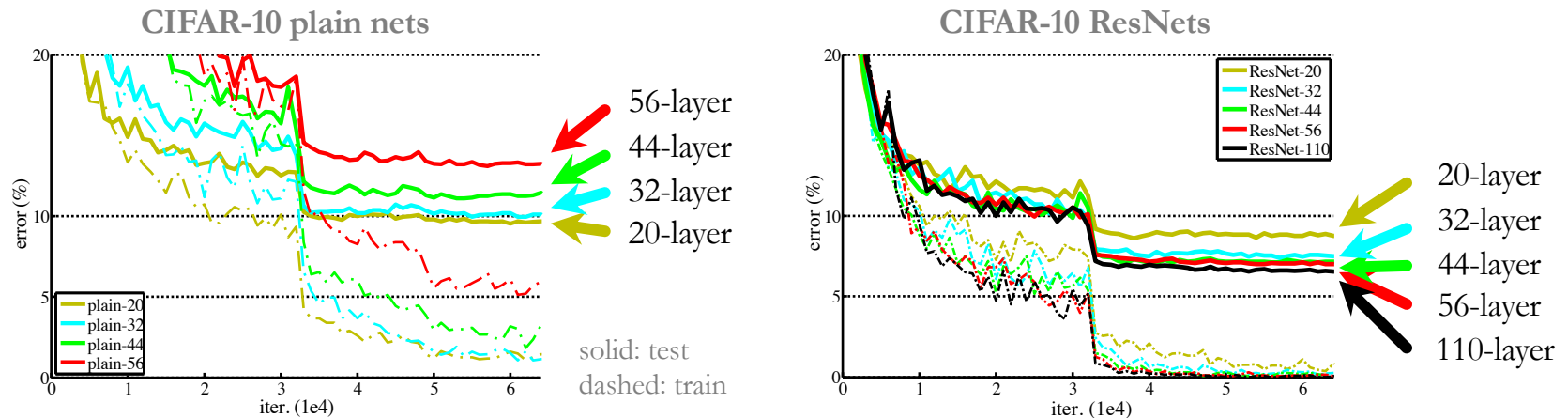
[Slide: Kaiming He]

CIFAR-10 experiments

CIFAR-10 plain nets

error (%)

20-layer
32-layer
44-layer
56-layer

plain-20
plain-32
plain-44
plain-56

solid: test
dashed: train

iter. (1e4)

CIFAR-10 ResNets

error (%)

ResNet-20
ResNet-32
ResNet-44
ResNet-56
ResNet-110

20-layer
32-layer
44-layer
56-layer
110-layer

iter. (1e4)

- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

[Slide: Kaiming He]
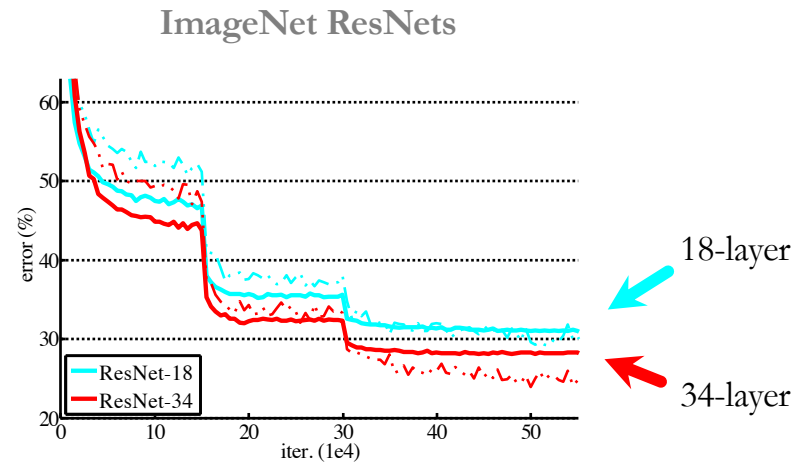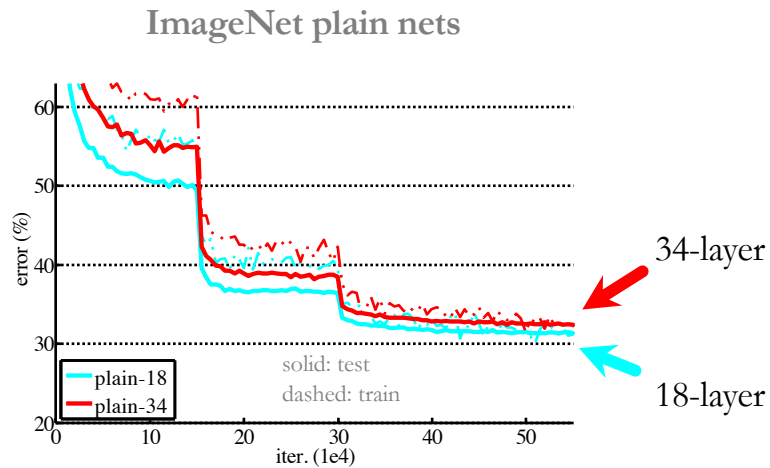
ImageNet experiments

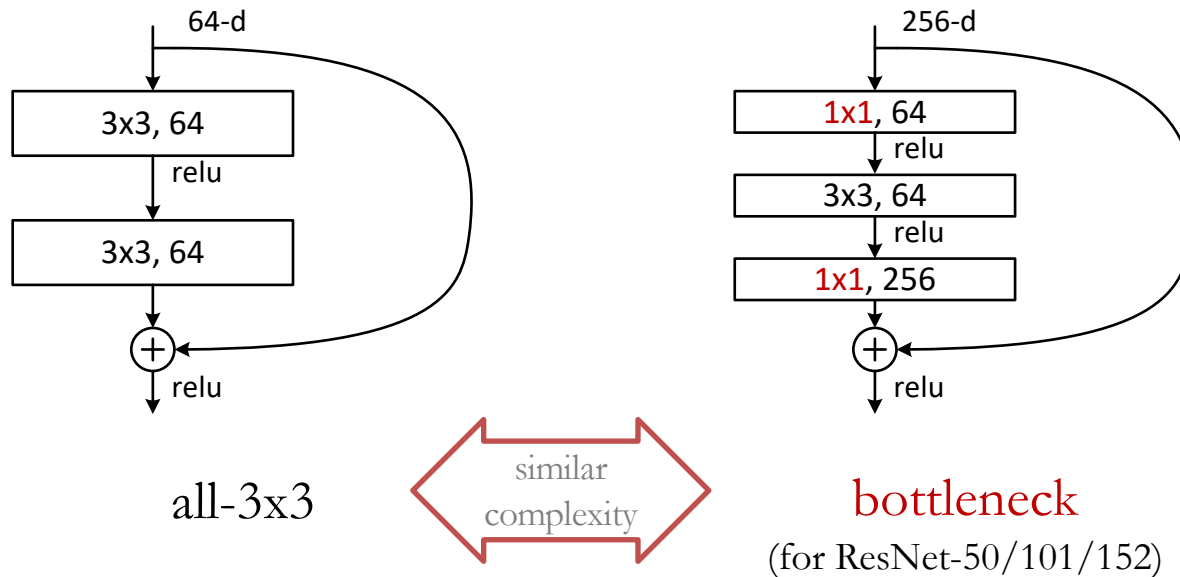

ImageNet plain nets

ImageNet ResNets

- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

[Slide: Kaiming He]

ImageNet experiments

- A practical design of going deeper

all-3x3 ⟷ similar complexity ⟷ **bottleneck**
(for ResNet-50/101/152)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

[Slide: Kaiming He]

ImageNet experiments

this model has **lower time complexity** than VGG-16/19

- Deeper ResNets have lower error

5.7 — ResNet-152
6.1 — ResNet-101
6.7 — ResNet-50
7.4 — ResNet-34

10-crop testing, top-5 val error (%)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

[Slide: Kaiming He]
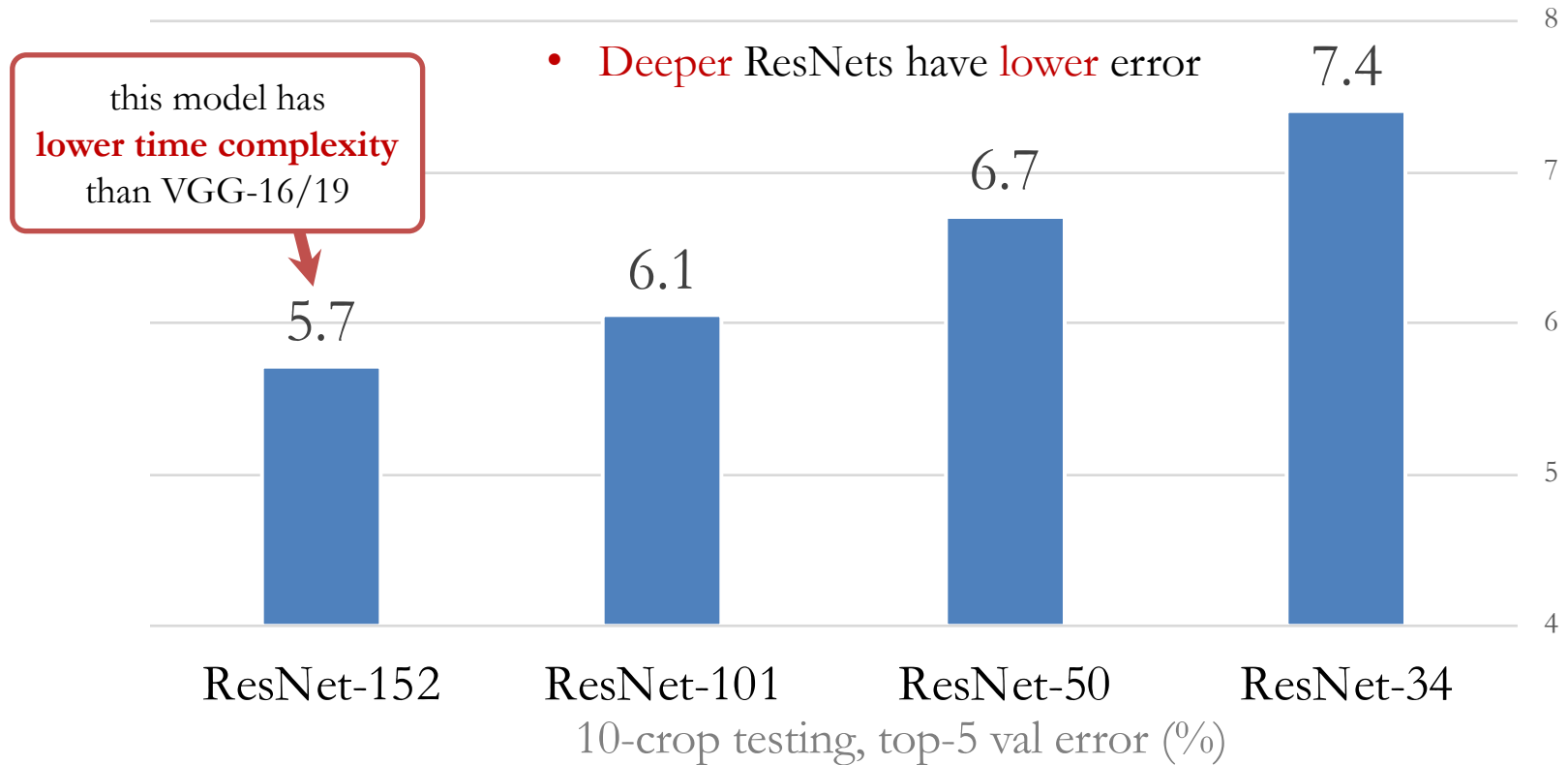
Issues on learning deep models

- **Representation** ability

  - Ability of model to fit training data, if optimum could be found
  - If model A's solution space is a superset of B's, A should be better.

- **Optimization** ability

  - Feasibility of finding an optimum
  - Not all models are equally easy to optimize

- **Generalization** ability

  - Once training data is fit, how good is the test performance

[The Tradeoffs of Large Scale Learning, Bottou & Bousquet, 2011]

[Slide: Kaiming He]

How do ResNets address these issues?

- **Representation** ability

  - No explicit advantage on representation (only re-parameterization), but
  - Allow models to go deeper

- **Optimization** ability

  - Enable very smooth forward/backward prop
  - Greatly ease optimizing deeper models

- **Generalization** ability
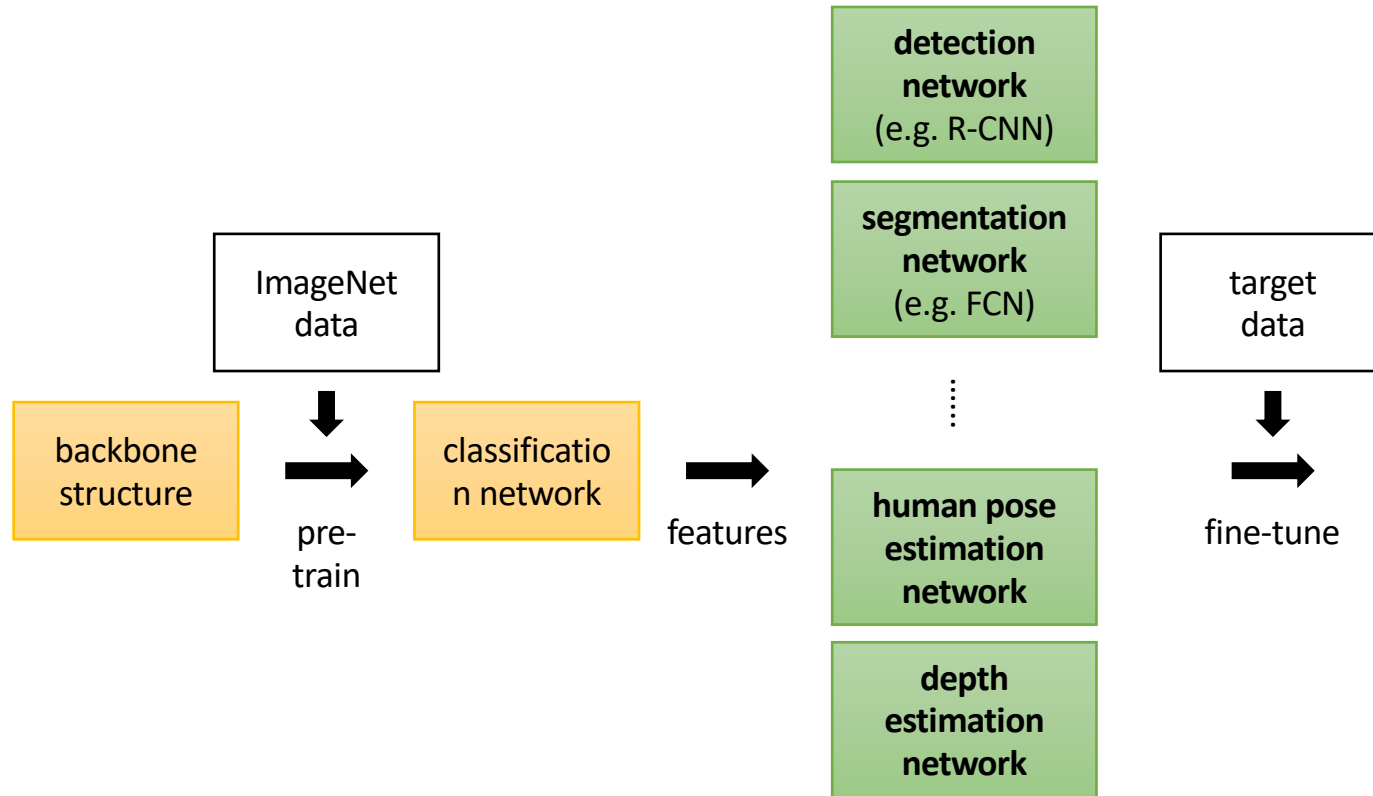
  - Not explicitly address generalization, but
  - Deeper+thinner is good generalization

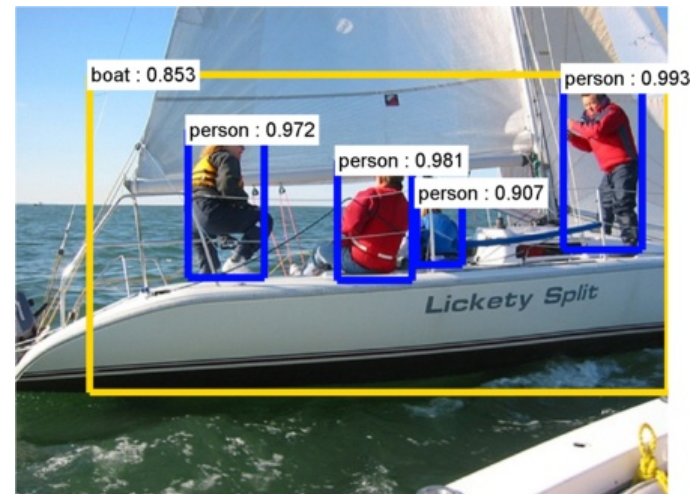[The Tradeoffs of Large Scale Learning, Bottou & Bousquet, 2011]

[Slide: Kaiming He]

Deep Learning for Computer Vision

[Slide: Kaiming He]

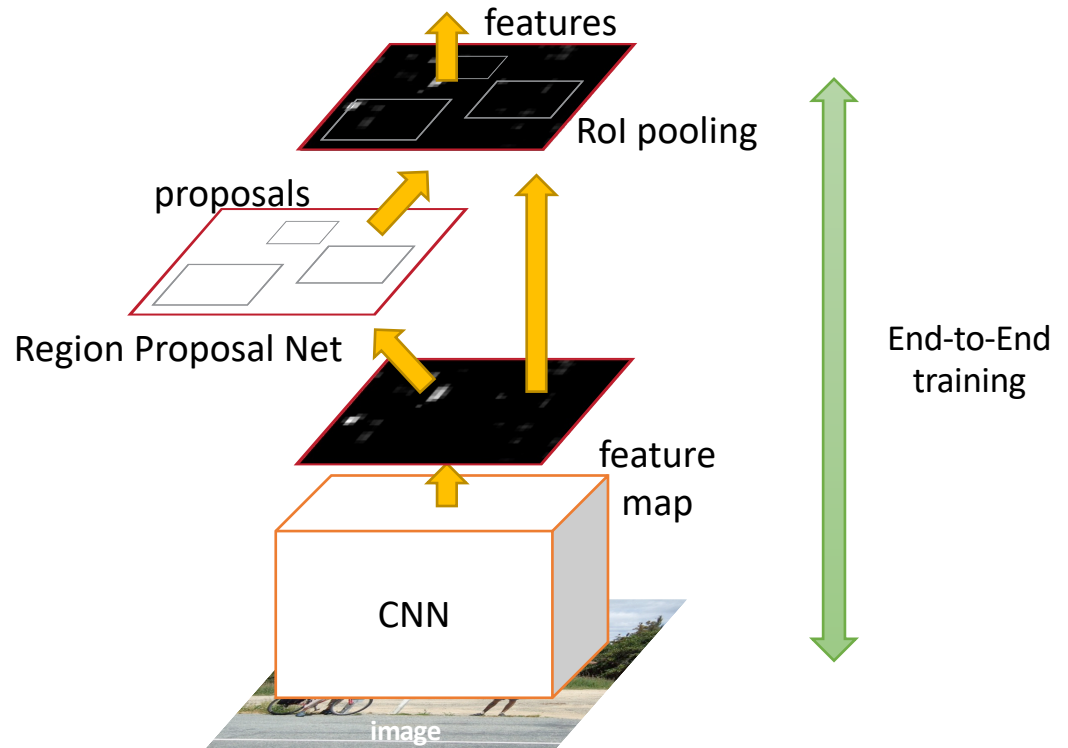# Example: Object Detection


Image Classification
(what?)

Object Detection
(what + where?)

Object Detection: Faster R-CNN

- Faster R-CNN
  - Solely based on CNN
  - No external modules
  - Each step is end-to-end

features

RoI pooling

proposals

Region Proposal Net

feature map

CNN

image

End-to-End training

Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". NIPS 2015.

[Slide: Kaiming He]

Mask R-CNN
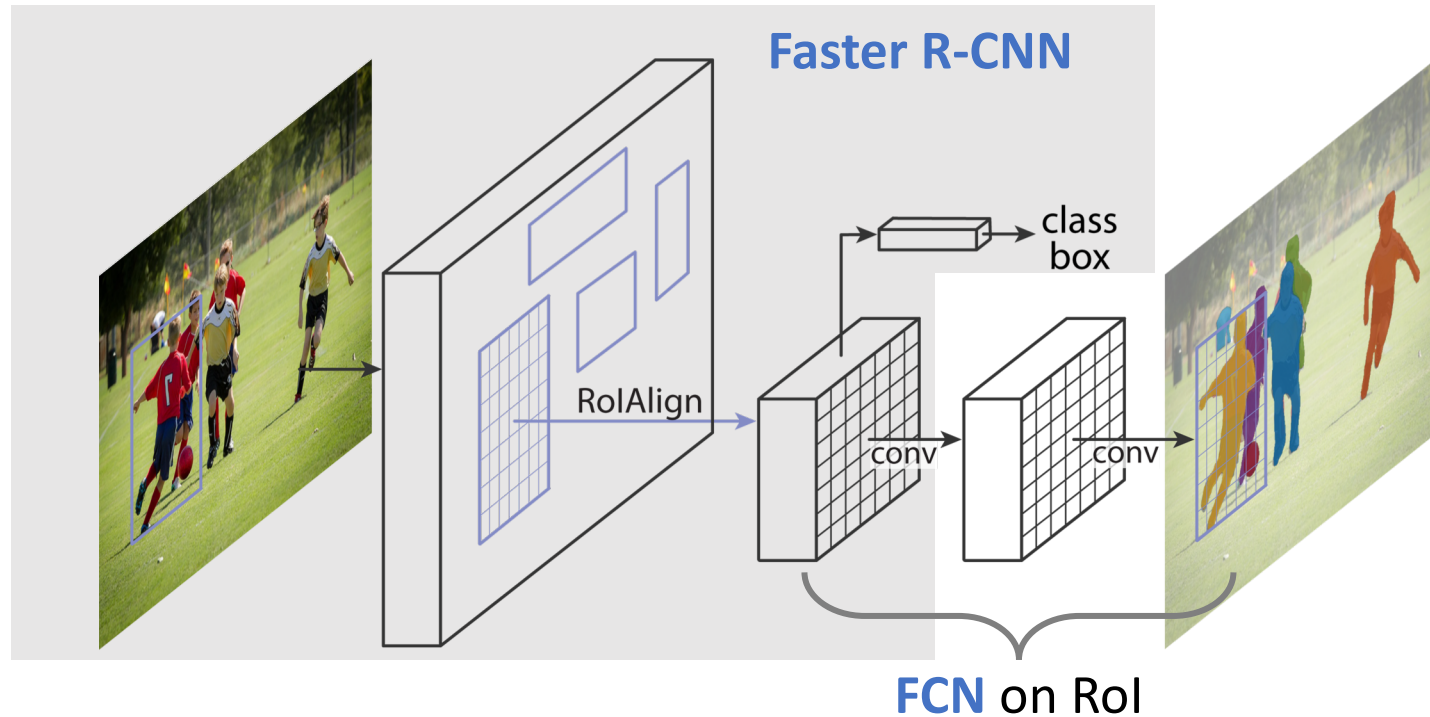
Jonathan Long, Evan Shelhamer, Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". CVPR 2015.
Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". NIPS 2015.
Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick. "Mask R-CNN". ICCV 2017.

[Slide: Kaiming He]

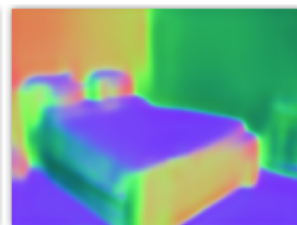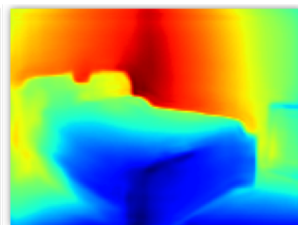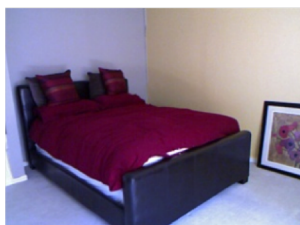Mask R-CNN frame-by-frame

pixels in, pixels out

colorization
Zhang et al.2016

monocular depth + normals Eigen & Fergus 2015

semantic
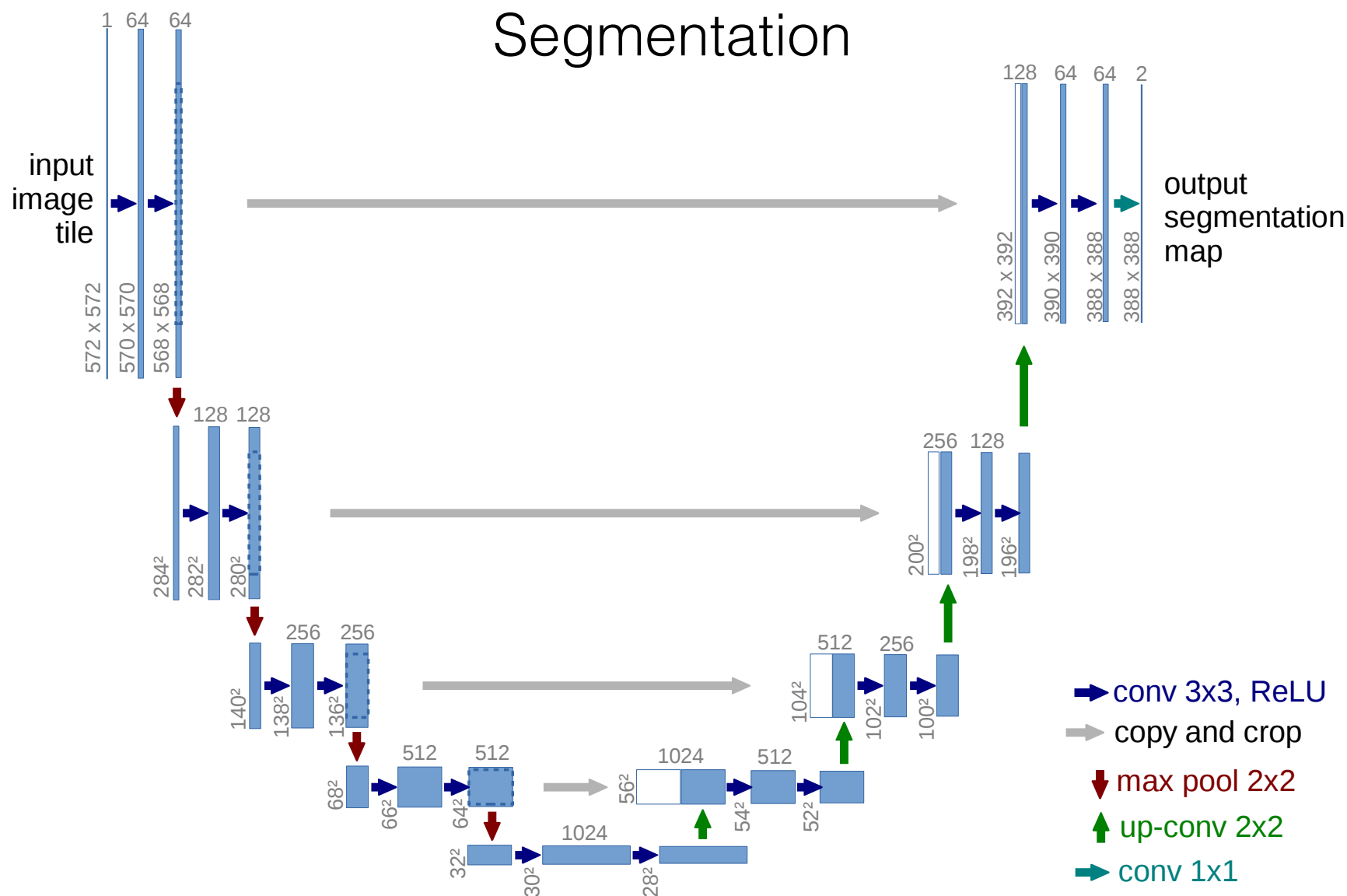segmentation

convolutional
network
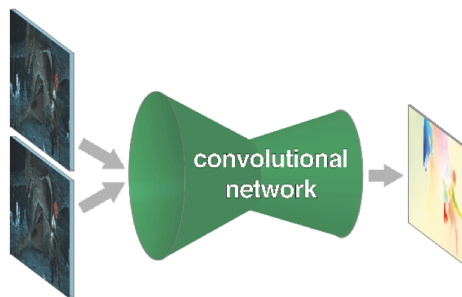
optical flow Fischer et al. 2015

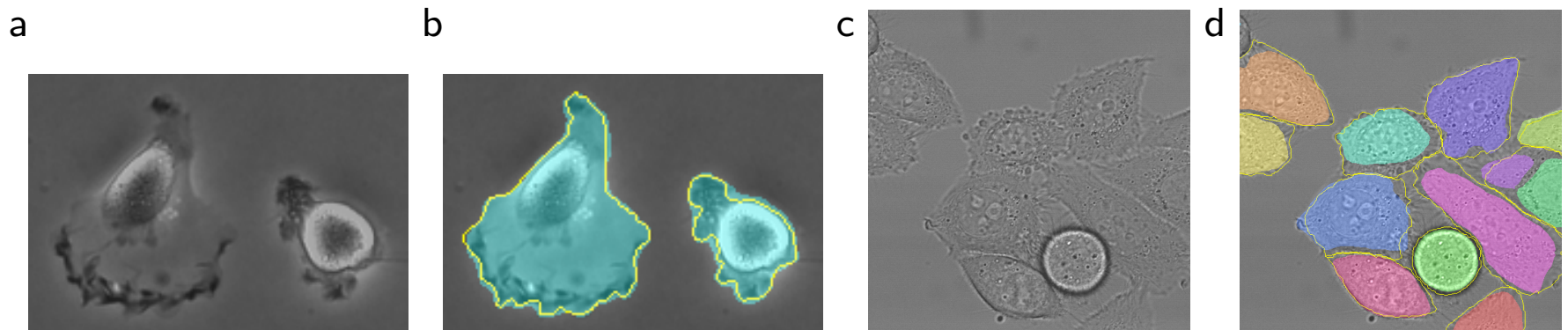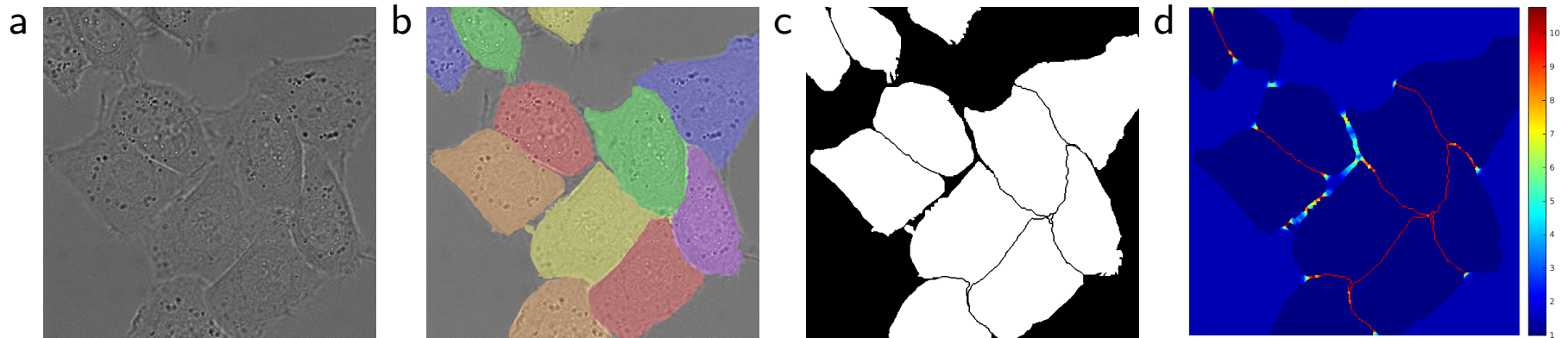boundary prediction Xie & Tu 2015

78

[Slide: Evan Shelhamer]

U-Net: Convolutional Network for Biomedical Image Segmentation

U-Net: Convolutional Network for Biomedical Image Segmentation

Exploring the Limits of Weakly Supervised Pretraining

[Mahajan et al., ECCV 2018]

Hashtag Supervision

- It is easy to get billions of public images and hashtags

- Hashtags are more structured than captions

- Hashtags were often assigned to make images "searchable"

#cheesecake #birthday

[Slide: Laurens van der Maaten]

Exploring the Limits of Weakly Supervised Pretraining

Hashtag Supervision

- But hashtags are not perfect supervision

- Some hashtags are not visually relevant

- Other hashtags are not in the photo

- And there are many false negatives

- Is this noise bias or variance? Is scaling up sufficient to reduce the variance?

#cat #travel #thailand #family
#building #fence #...

[Slide: Laurens van der Maaten]

Experiments

- Select a set of hashtags

- Download all public Instagram images that has at least one of these hashtags

- Use WordNet synsets to merge hashtags into canonical form (merge #brownbear and #ursusarctos)

- Final dataset has ~3.5 **billion** images

facebook
Artificial Intelligence Research

[Slide: Laurens van der Maaten]