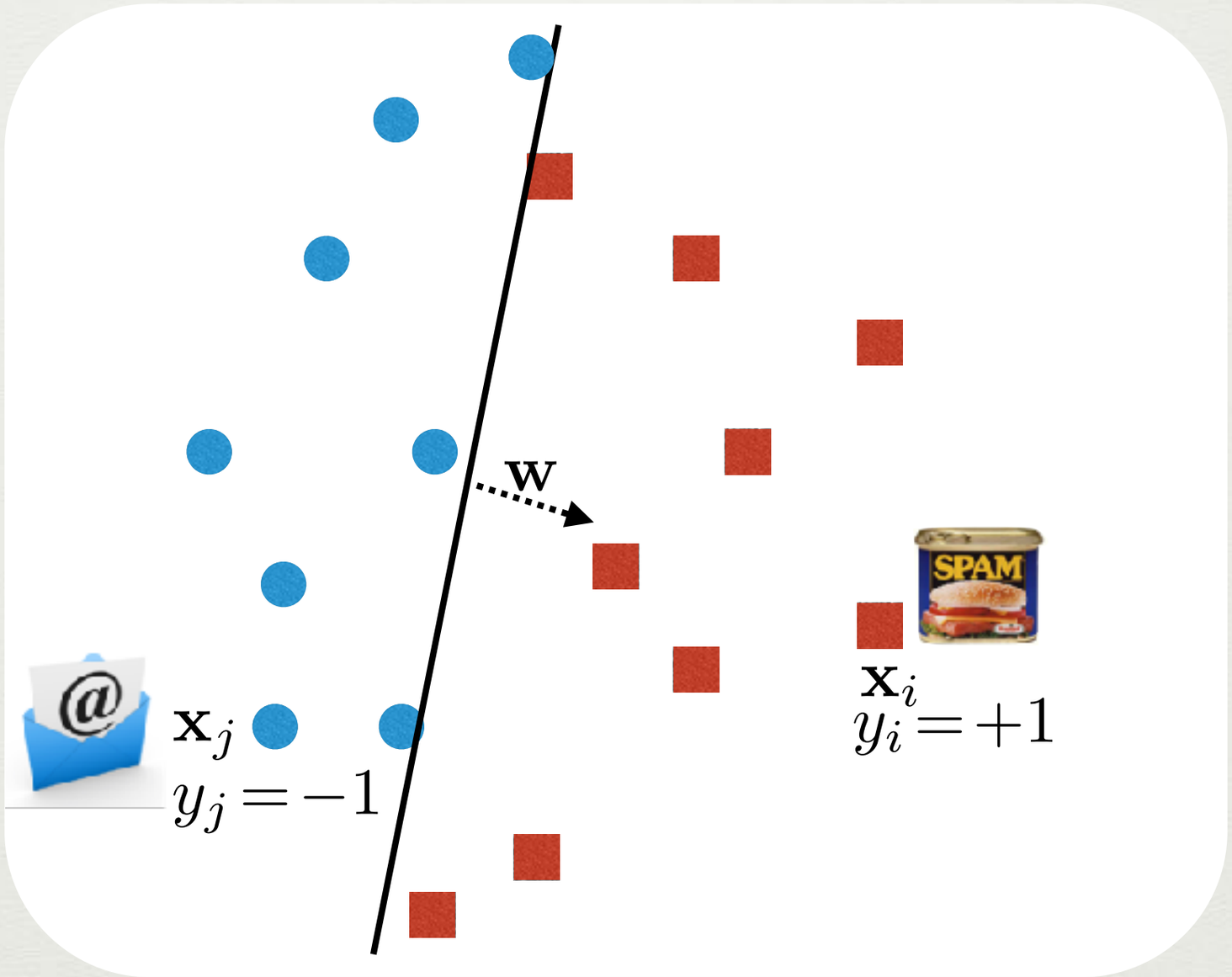
DEEP NETWORKS WITH DENSE CONNECTIVITY

Kilian Q. Weinberger
Cornell University

A 3 Minutes Introduction to "Deep Learning"

Perceptron

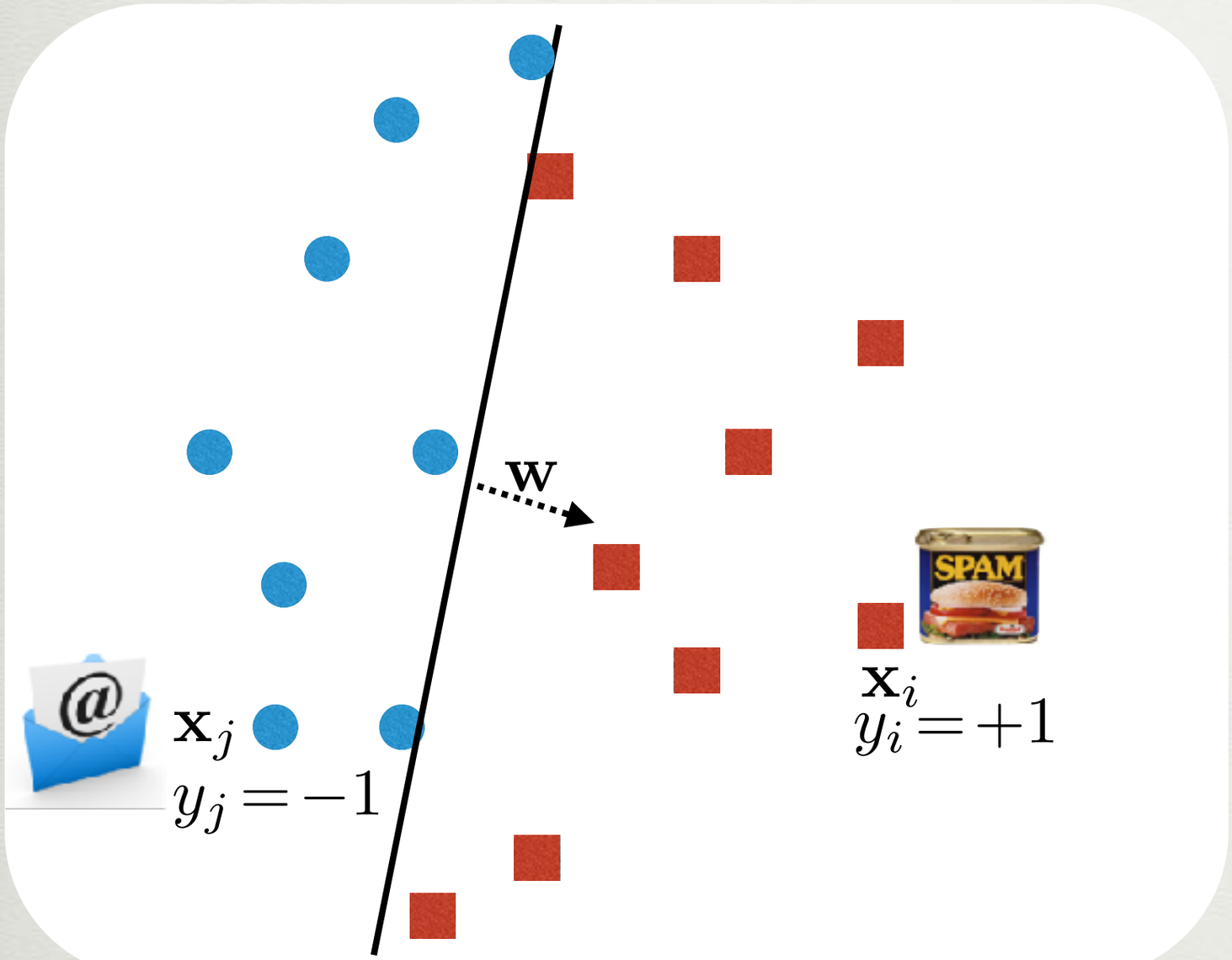
[Rosenblatt 1957]

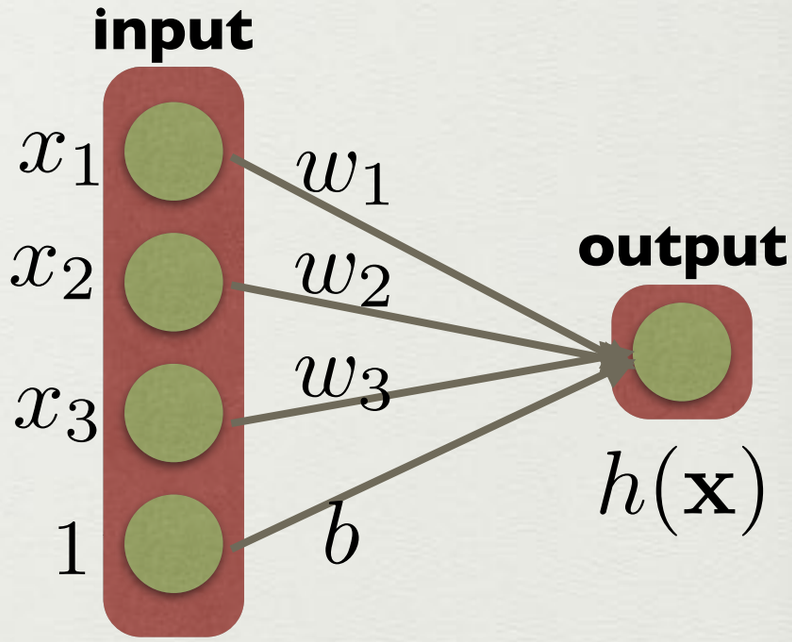


$$h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

Perceptron

[Rosenblatt 1957]

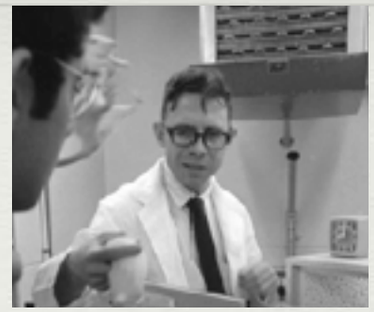




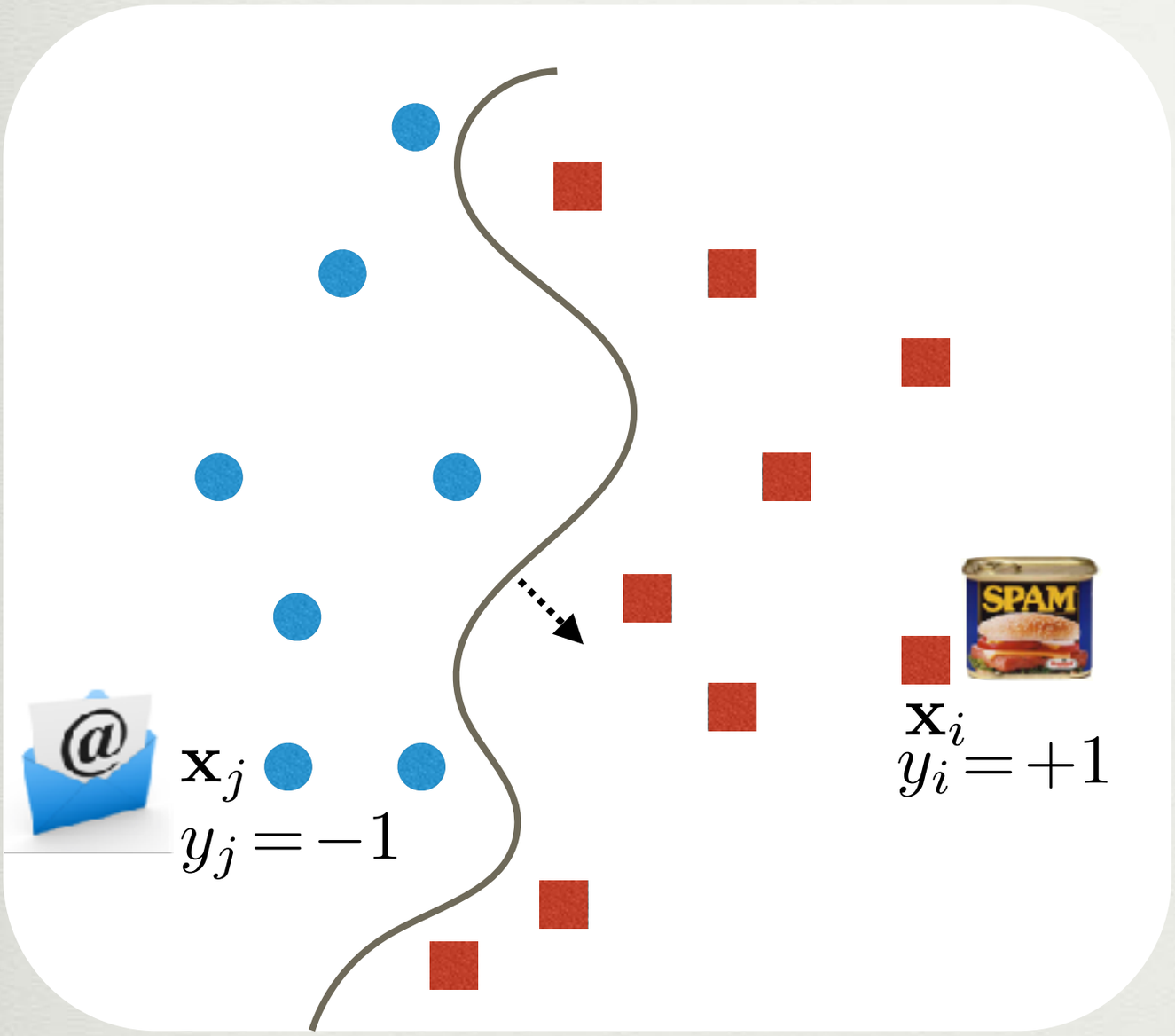
$$h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

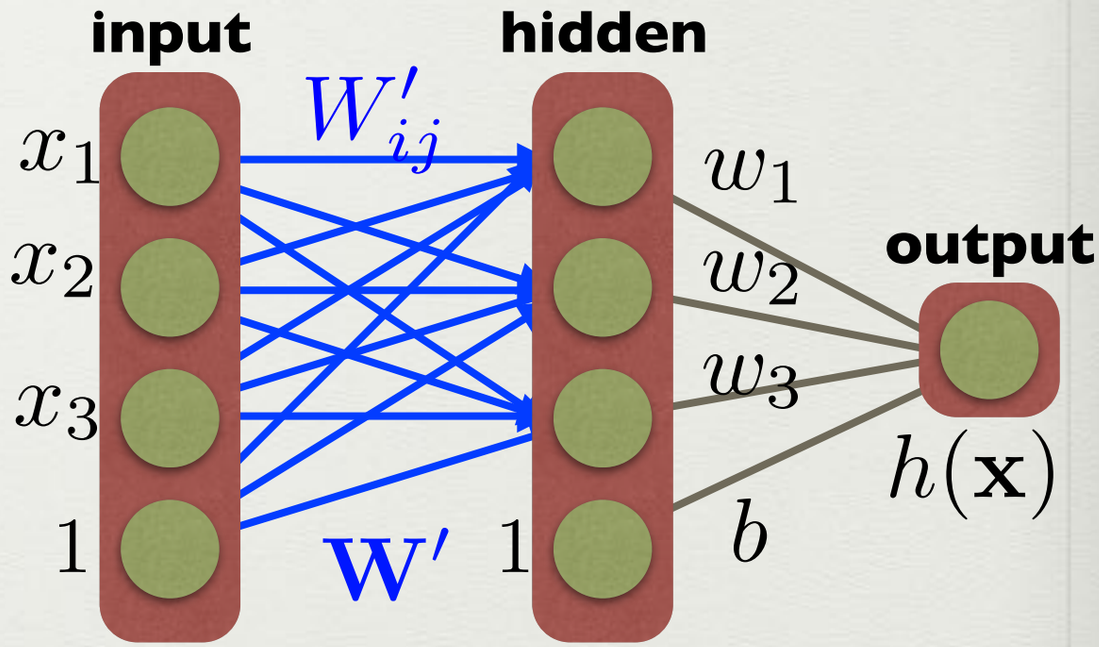
Multi-Layer Perceptron

(a.k.a. Neural Networks)



[Rosenblatt 1961]





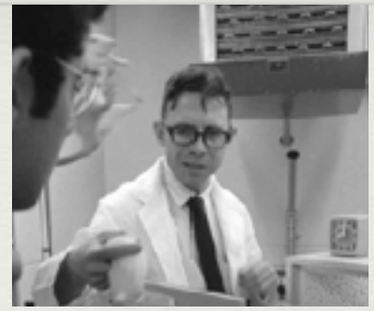
$$\sigma(a) = \max(a, 0)$$

Rectified Linear Units

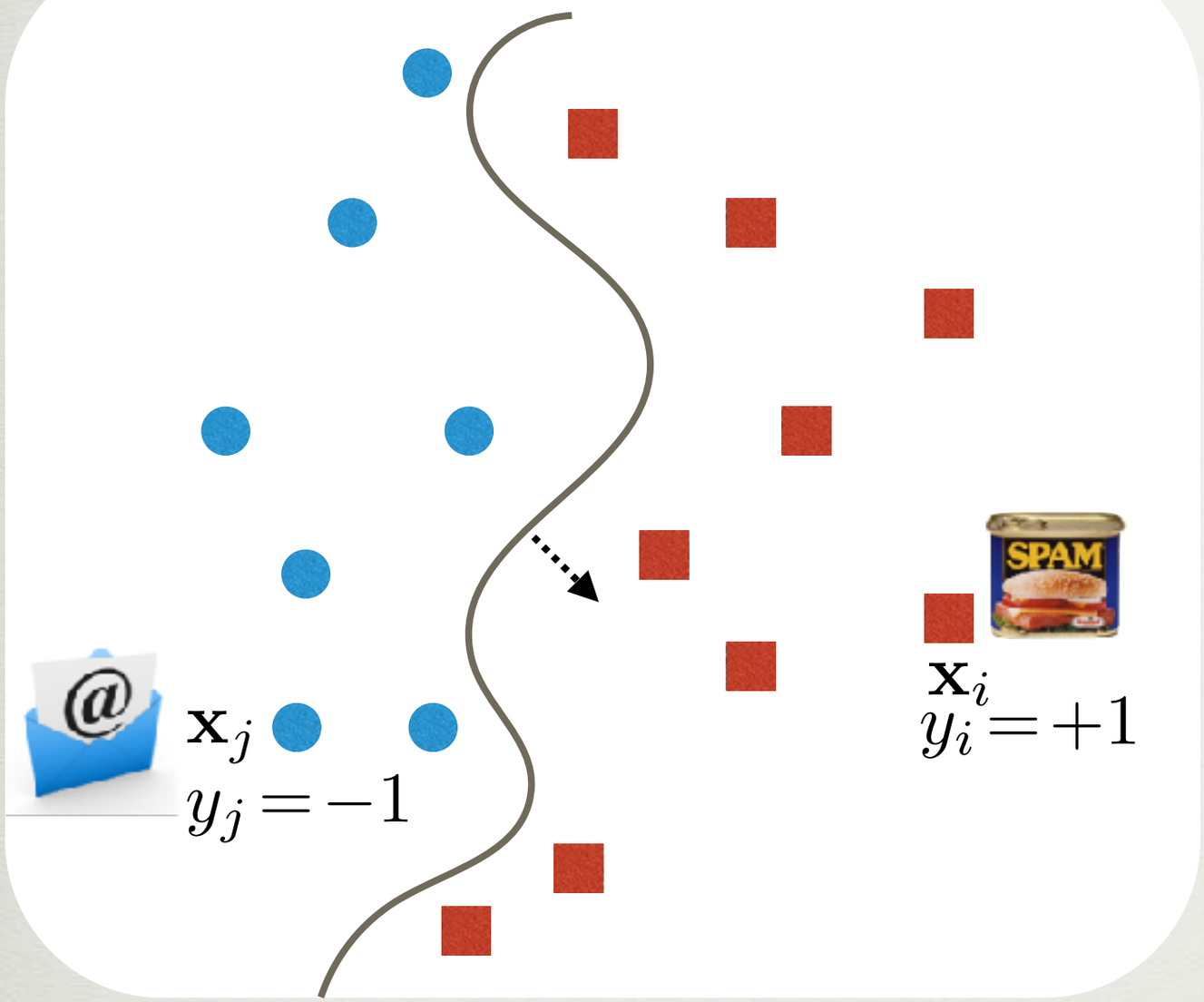
$$h(\mathbf{x}) = \mathbf{w}^\top \sigma\left(\mathbf{W}'\mathbf{x} + \mathbf{c}\right) + b$$

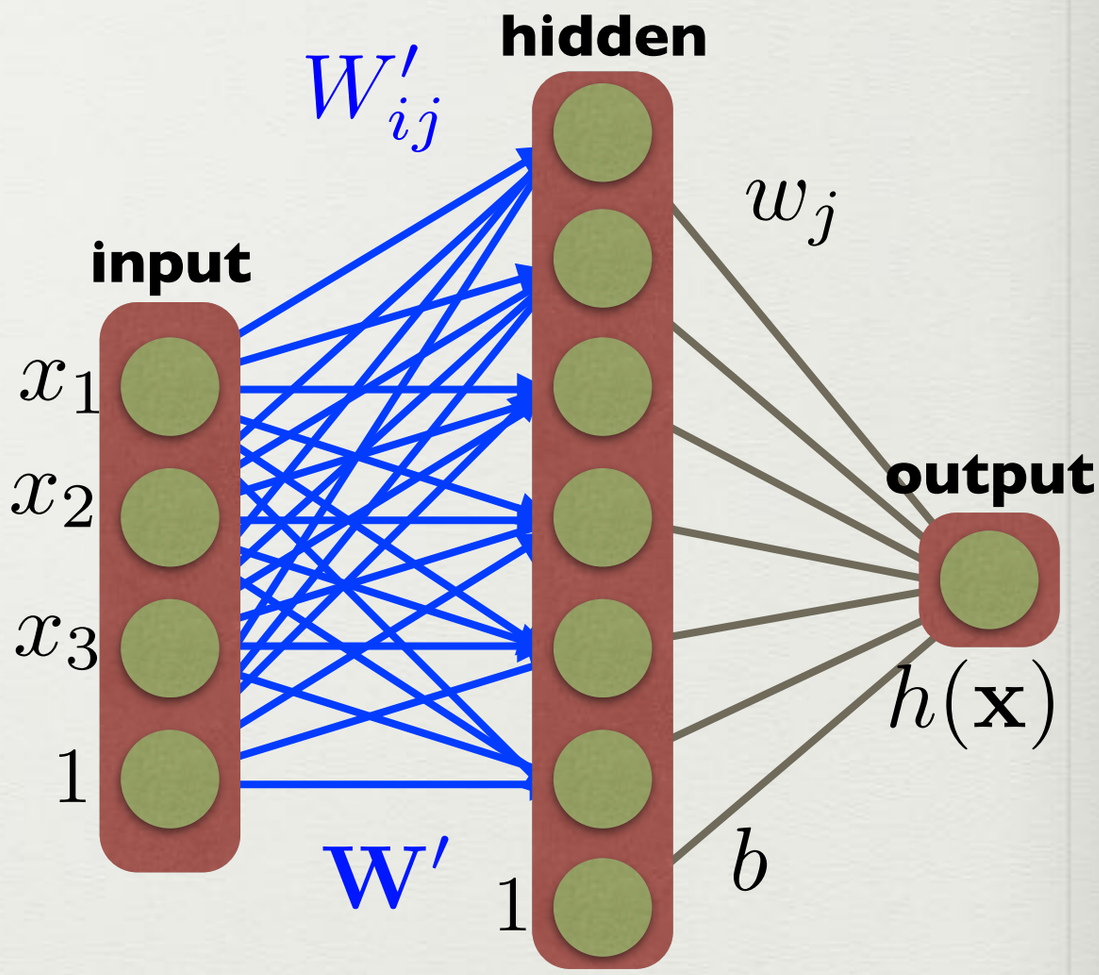
Multi-Layer Perceptron

(a.k.a. Neural Networks)



[Rosenblatt 1961]





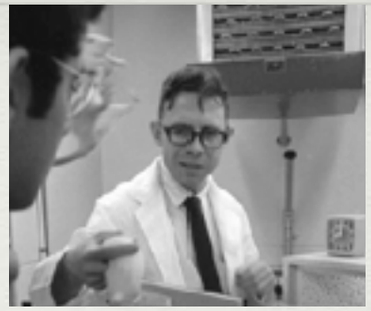
$$\sigma(a) = \max(a, 0)$$

Rectified Linear Units

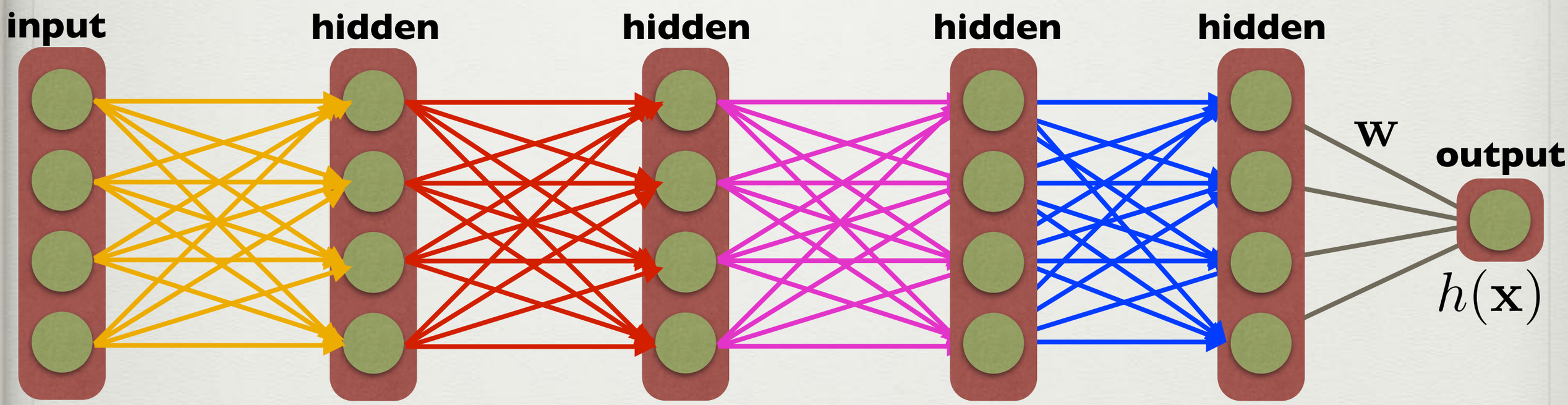
$$h(\mathbf{x}) = \mathbf{w}^\top \sigma\left(\mathbf{W}'\mathbf{x} + \mathbf{c}\right) + b$$

Multi-Layer Perceptron

(a.k.a. Neural Networks, Deep Learning)



[Rosenblatt 1961]



$$h(\mathbf{x}) = \mathbf{w}^\top \sigma(\mathbf{W}' \sigma(\mathbf{W}^2 \sigma(\mathbf{W}^3 \sigma(\mathbf{W}^4 \mathbf{x}))))$$

DEEP LEARNING WORKS

SCIENCE

Scientists See Promise in Deep-Learning Programs

By JOHN MARKOFF NOV. 23, 2012

Topic: Innovation

Google's DeepMind artificial intelligence aces Atari gamin challenge

Summary: DeepMind has published a paper detailing how its AI tech not only *a host of Atari games, but went on to succeed in a number of them.*

By Liam Tung | February 26, 2015 — 10:42 GMT (02:42 PST)

Google's DeepMind artificial intelligence unit has shown that, given little more than play with, its algorithm can not only learn how to play computer games from scratc ace them after a few hours of practice.

1 Artificial Inte

DEEP LEARNING IMAGENET

TAG AlphaGo , Deep Learning , Artificial Intelligence

AlphaGo Beats Go Human Champ: Godfather Of Deep Learning Tells Us Do Not Be Afraid Of AI

By Aaron Mamiit, Tech Times | March 21, 10:16 AM

Like Follow Share Tweet Reddit 0 Comments ••• SUBSCRIBE

Last week, Google's artificial intelligence program AlphaGo dominated its match with South Korean world Go champion, Lee Sedol. Geoffrey Hinton, called the godfather of deep learning, explained the win's importance and why we should not fear artificial intelligence.
(Photo : Google Handout | Getty Images)

Last week, Google's artificial intelligence program AlphaGo dominated its match with South Korean world Go champion Lee Sedol, winning with a 4-1 score.

The achievement stunned artificial intelligence experts, who previously thought that Google's computer program would need at least 10 more years before developing enough to be able to beat a human world champion.

What could be scary regarding the computer program is that Google DeepMind CEO Demis Hassabis said that AlphaGo could still improve its performance, as the match with Sedol was able to expose some of its weaknesses.

Computers have long been winning against skilled humans in

TECH

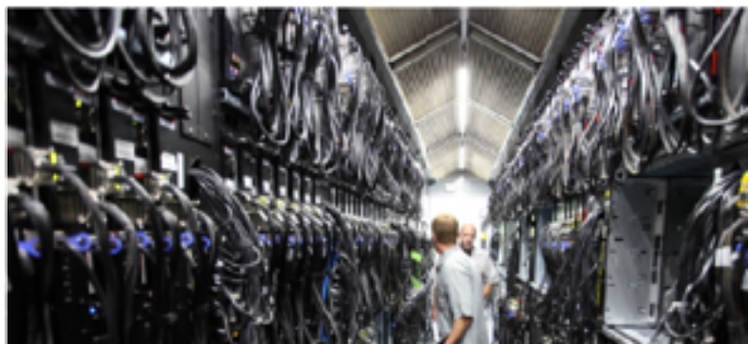
Microsoft's Deep Learning Project Outperforms Humans In Image Recognit

+ Comment Now + Follow Comments

Technology is blanketed in dishonesty. Computer phones are smart, software automations become intelligence, and coerced financialization becomes sharing. Because of the deceptive language surrounding these instruments it's difficult to talk about how they're used, and at what cost. Instead we're forced into false debates about sharing versus not sharing, intelligence versus inefficiency, progress versus everything.

s big a fraud as any of these endeavors, an scure discipline built around the claim that nimic human neuronal function and thus learn han humans. This week, Microsoft ved its newest deep learning project had outperformed humans jects in digital images. Researchers noted their scores shouldn't t computer image identification in general was better than g many general case instances where humans were better able

Microsoft researchers say their newest learning system beats humans — and Google



Part I

Problems when networks get really deep.

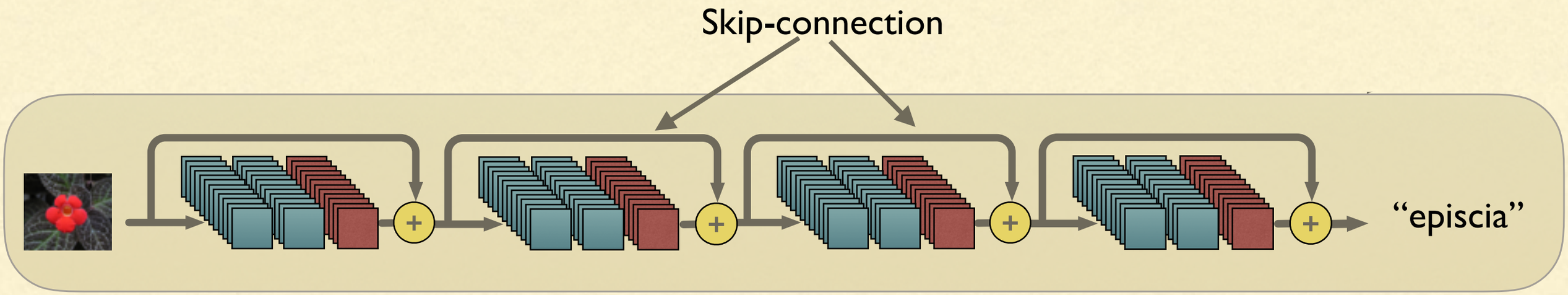
~~Stille Post~~

[ICLR 2018]

VERY DEEP NETWORKS

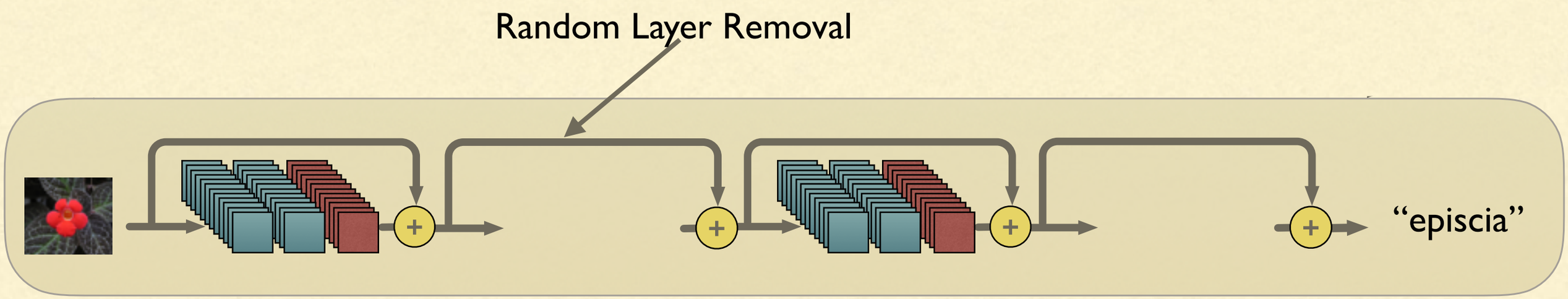
input

RESIDUAL NETWORKS



ResNet Architecture: [He, Zhang, Ren, Sun, CVPR'16]

STOCHASTIC DEPTH



[Huang et al., ECCV'16]

TELEPHONE

(input)

training

It's a car!!
(output)

TELEPHONE

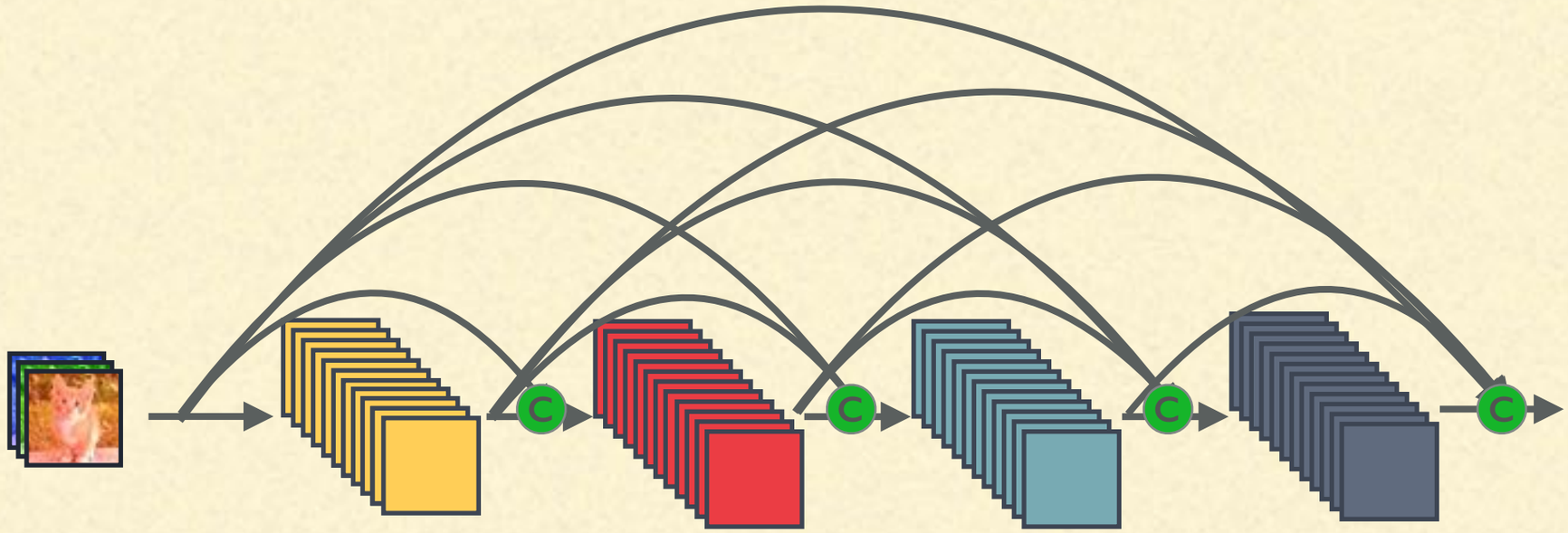
(input)

testing

It's a bar!!
(output)

(input)

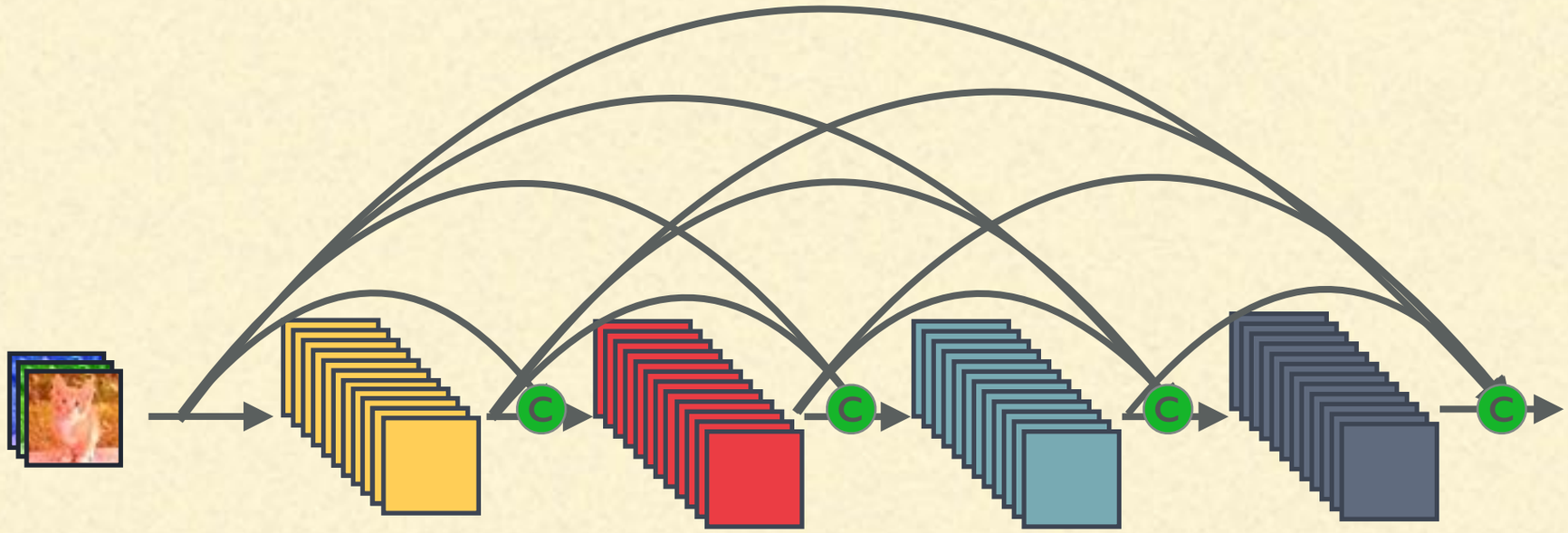
DENSE CONNECTIVITY



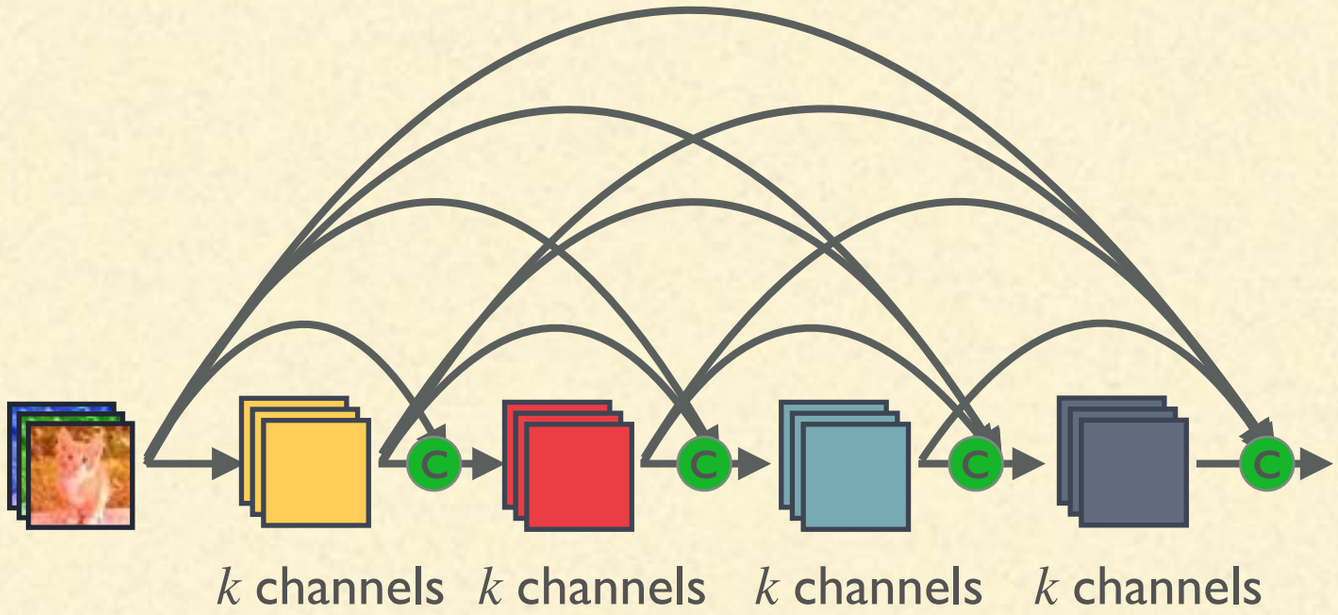
- C : Channel-wise concatenation

[CVPR 2017]

DENSE AND SLIM

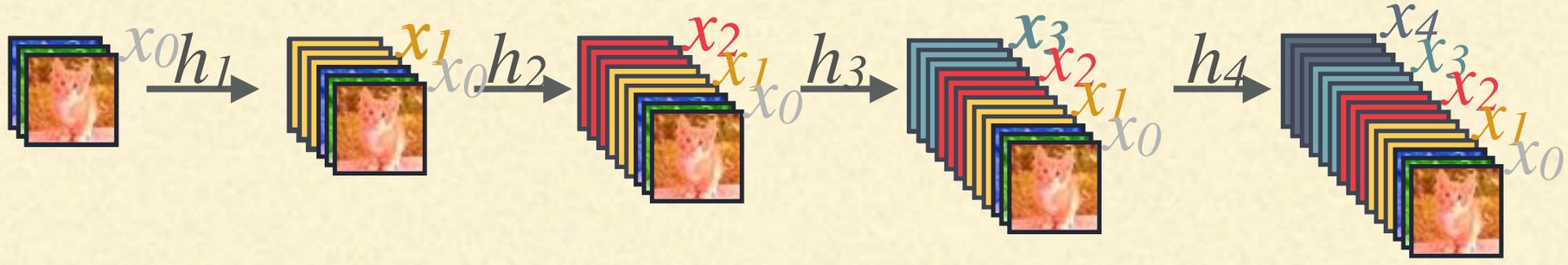


DENSE AND SLIM

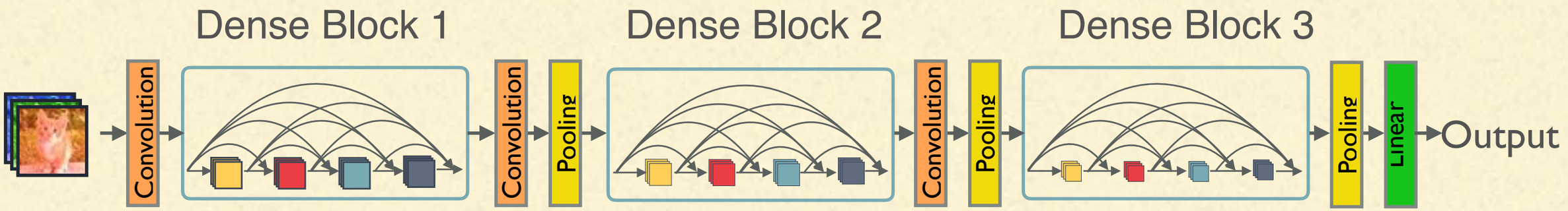


k : Growth Rate

FORWARD PROPAGATION



DENSENET


Dense Block 1
Dense Block 2
Dense Block 3
Convolution
Convolution
Pooling
Convolution
Pooling
Pooling
Linear
Output

RESULTS

DATA SETS

CIFAR-10
10 classes
50K training images

Krizhevsky et al. 2009

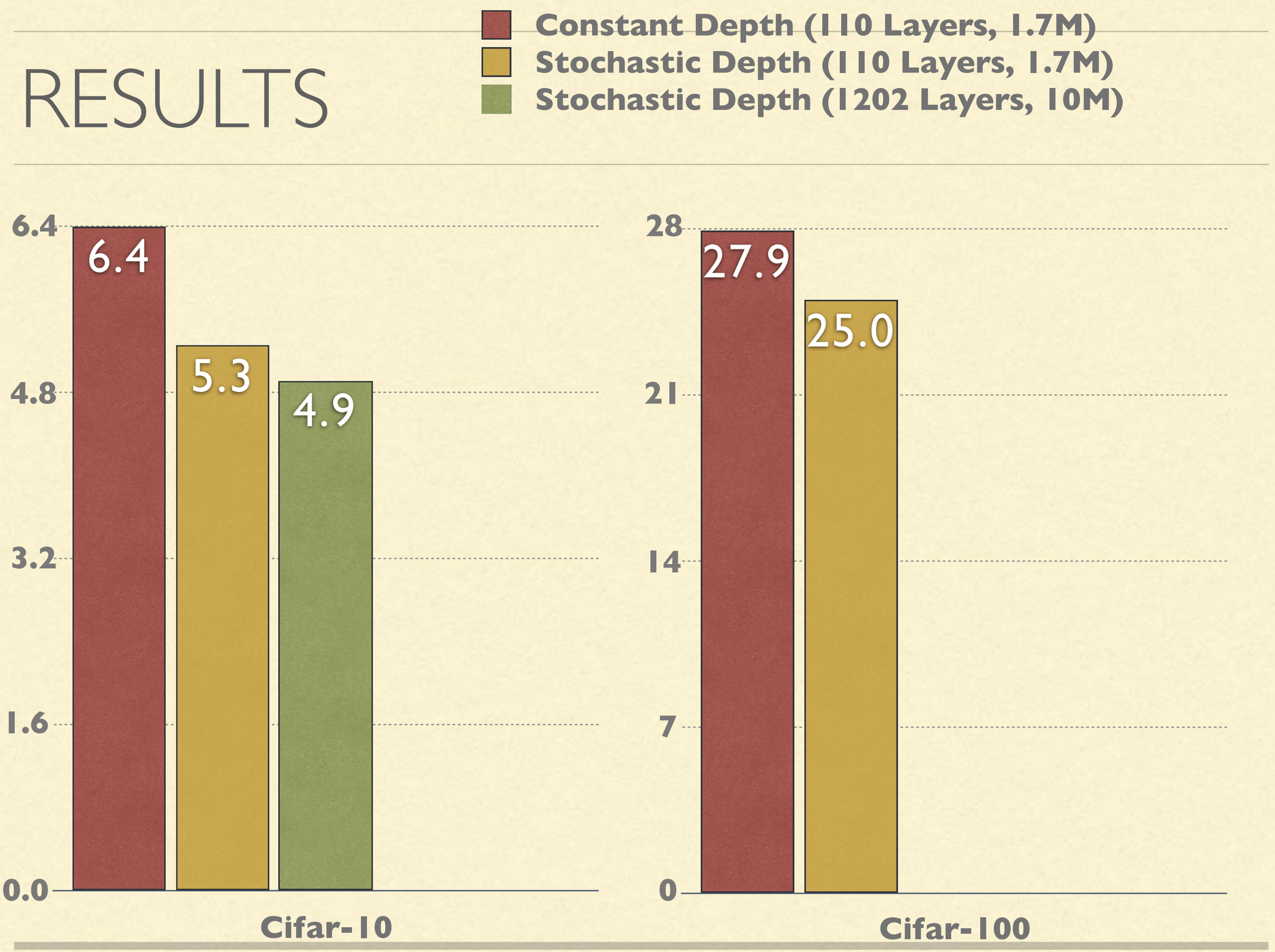
CIFAR-100
100 classes
50K training images

Krizhevsky et al. 2009

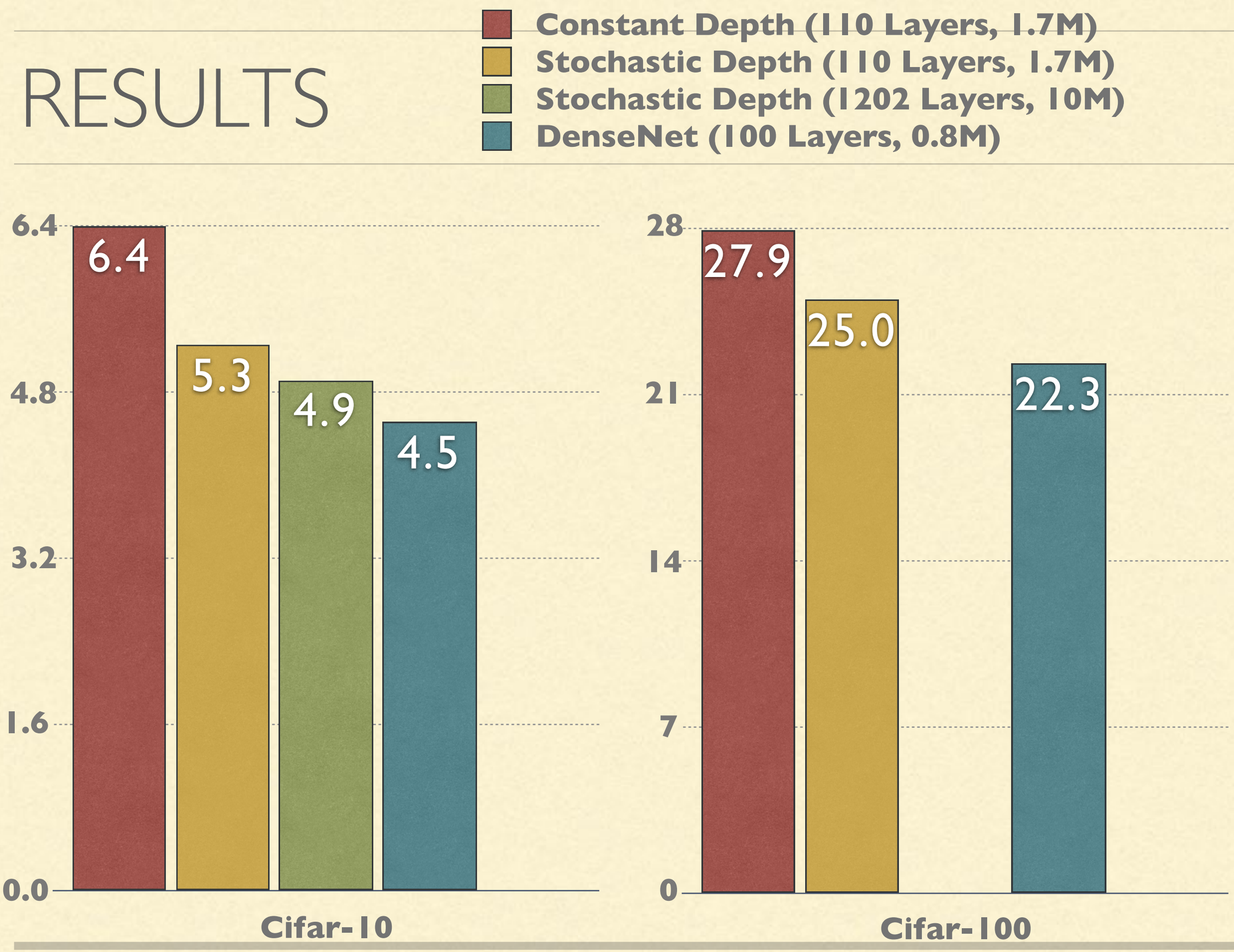
ImageNet
1000 classes
1.2M training images

Krizhevsky et al. 2009

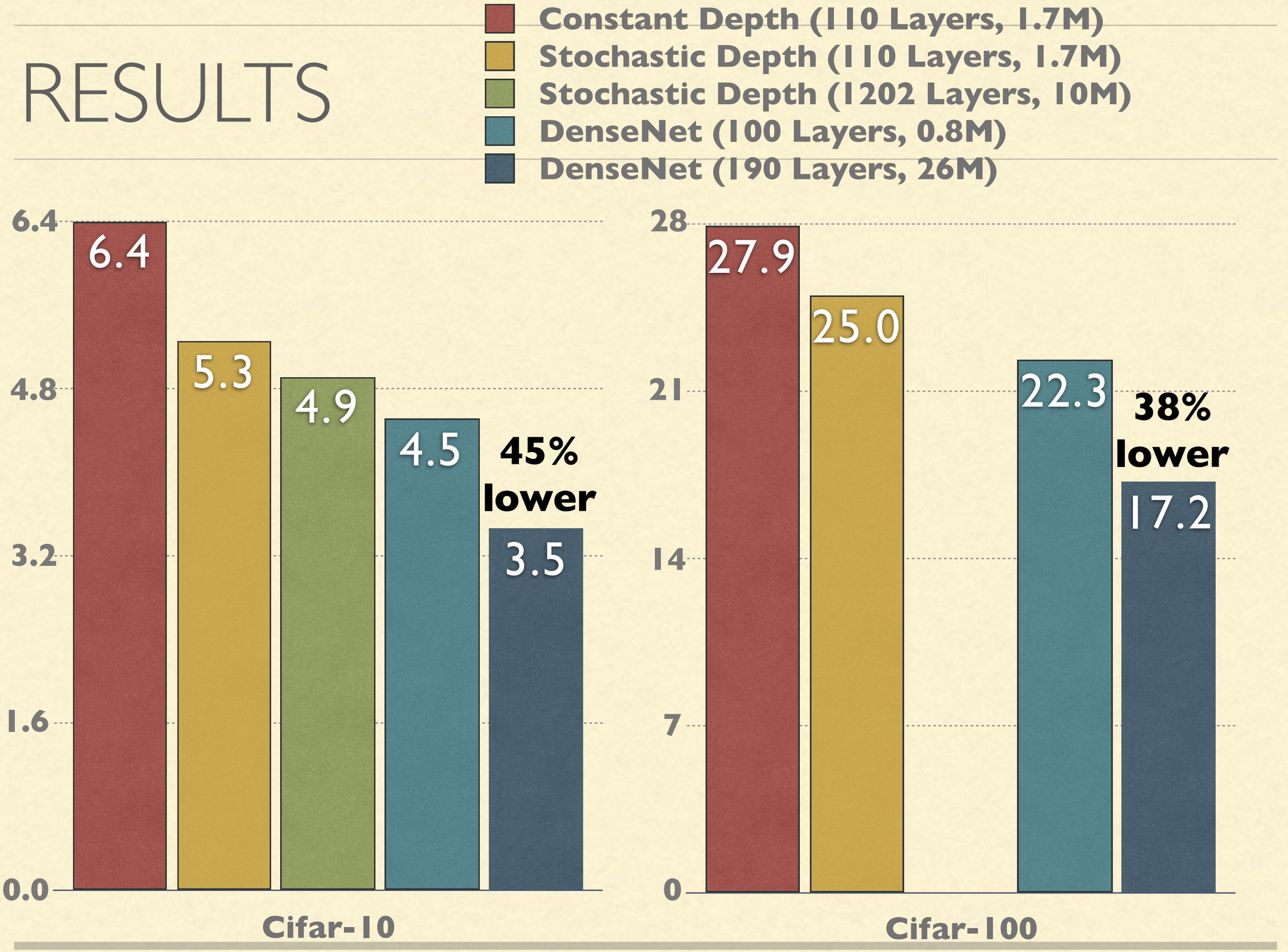
RESULTS


Constant Depth (110 Layers, 1.7M)
Stochastic Depth (110 Layers, 1.7M)
Stochastic Depth (1202 Layers, 10M)
6.4
4.8
3.2
1.6
0.0
6.4
5.3
4.9
Cifar-10
28
21
14
7
0
27.9
25.0
Cifar-100

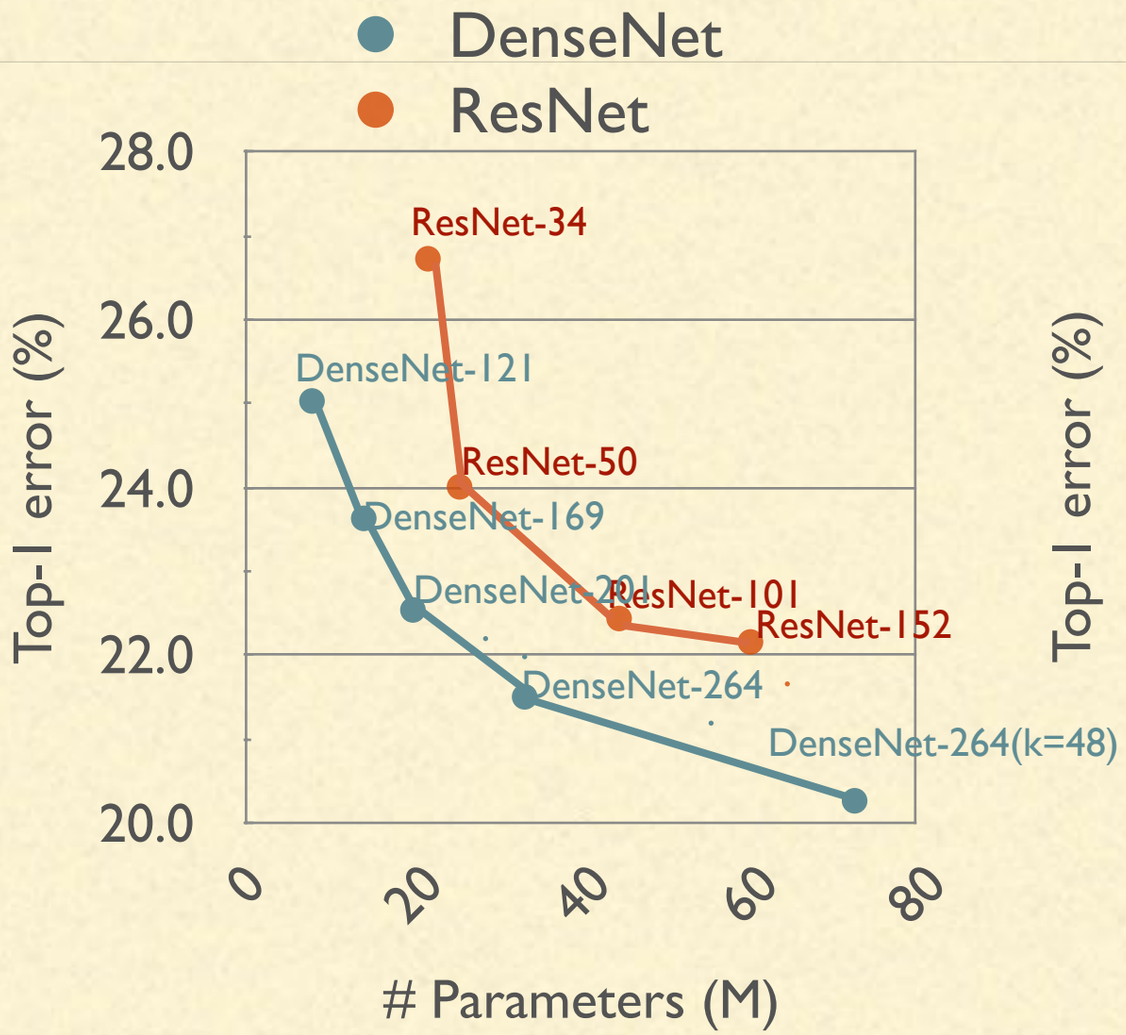
RESULTS

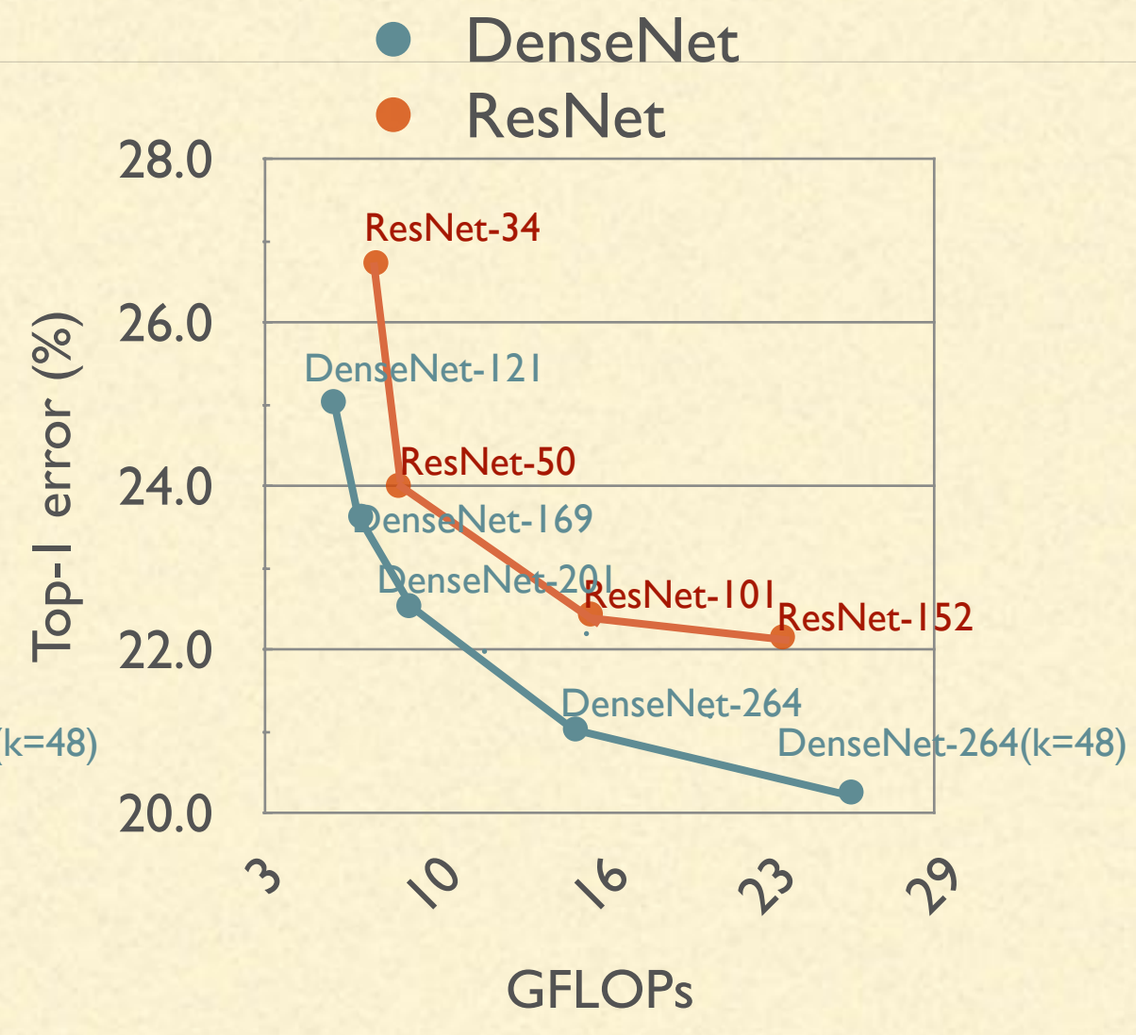

Constant Depth (110 Layers, 1.7M)
Stochastic Depth (110 Layers, 1.7M)
Stochastic Depth (1202 Layers, 10M)
DenseNet (100 Layers, 0.8M)
6.4
4.8
3.2
1.6
0.0
6.4
5.3
4.9
4.5
Cifar-10
28
21
14
7
0
27.9
25.0
22.3
Cifar-100

RESULTS


Constant Depth (110 Layers, 1.7M)
Stochastic Depth (110 Layers, 1.7M)
Stochastic Depth (1202 Layers, 10M)
DenseNet (100 Layers, 0.8M)
DenseNet (190 Layers, 26M)
6.4
4.8
3.2
1.6
0.0
6.4
5.3
4.9
4.5
45% lower
3.5
Cifar-10
28
21
14
7
0
27.9
25.0
22.3
38% lower
17.2
Cifar-100

RESULTS ON **IMAGENET**





Part II

Why do we need huge models?

[ICLR 2018]

EASY & HARD SAMPLES

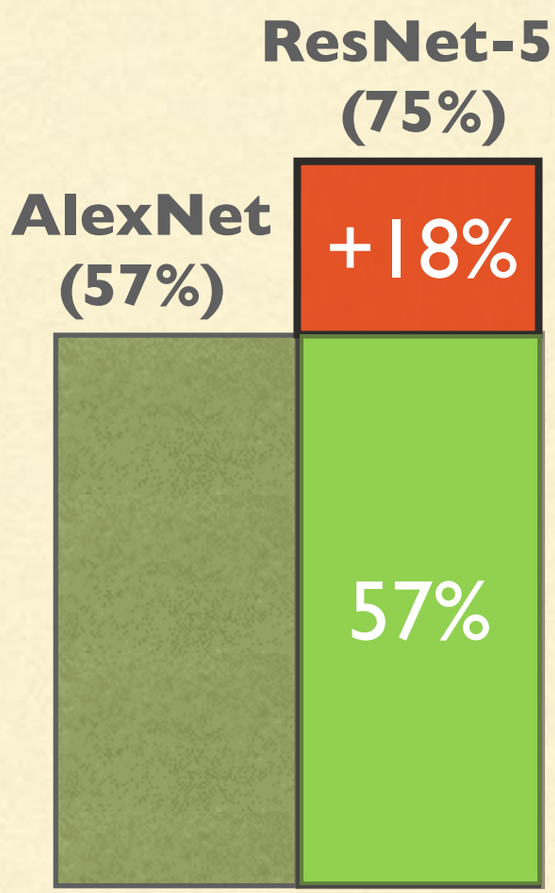
Some of the images are **easy**,
others are **hard.**

"easy" dog

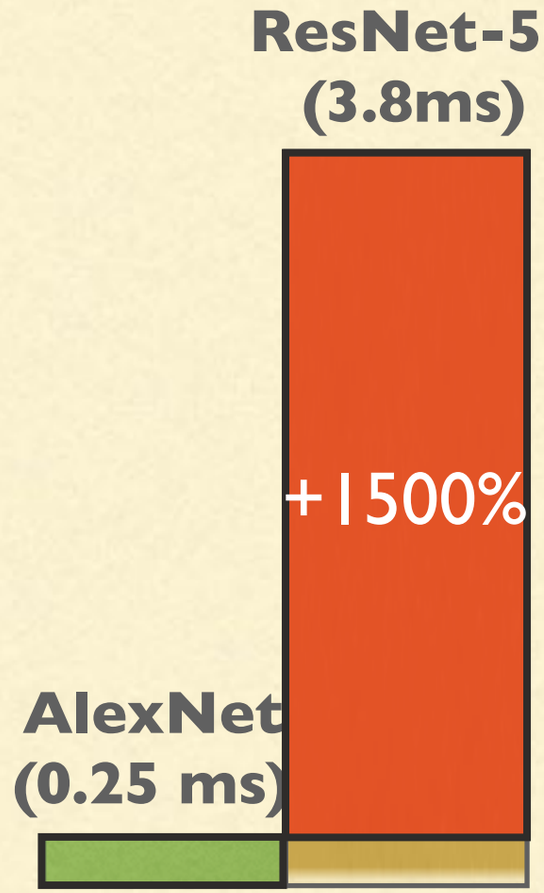
"hard" dog

ACCURACY & SPEED

(on ImageNet)

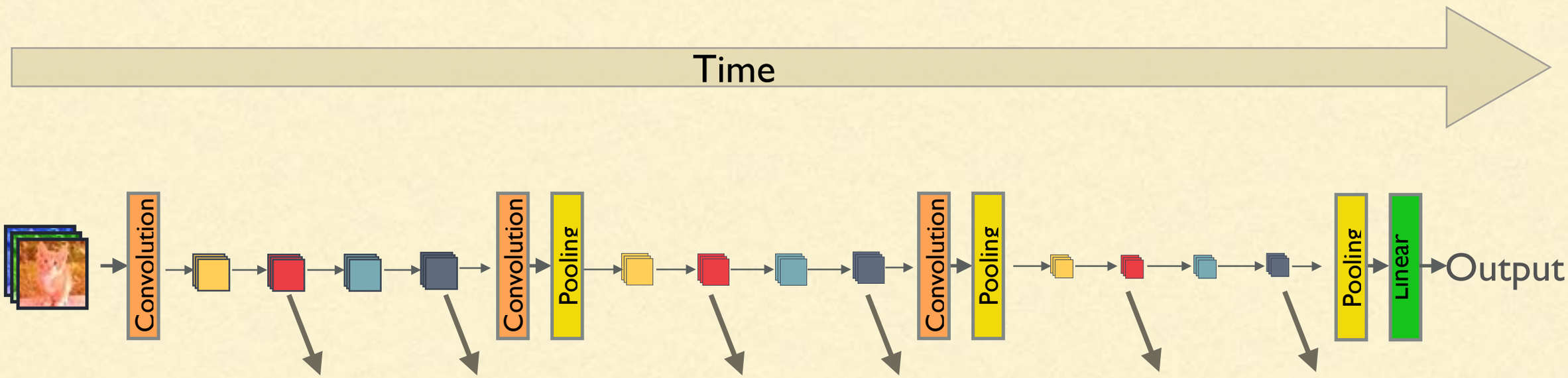


Accuracy

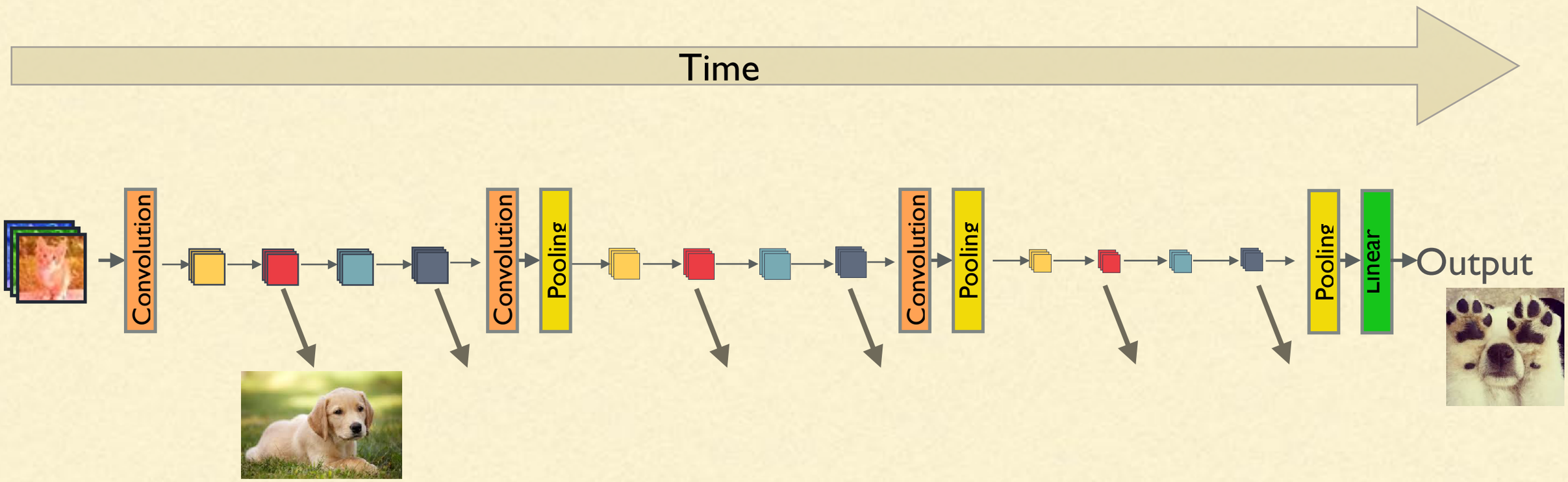


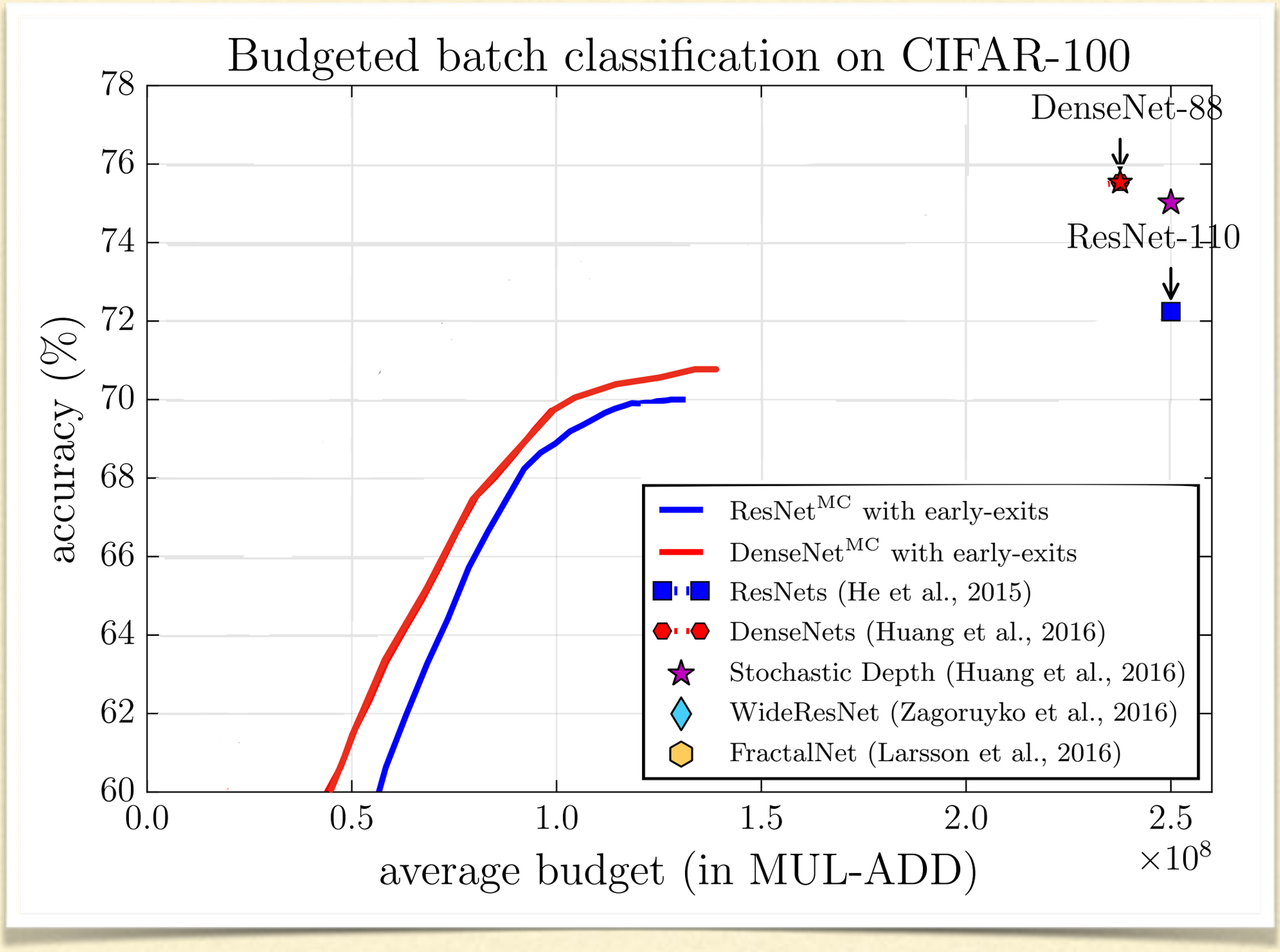
Inference Time

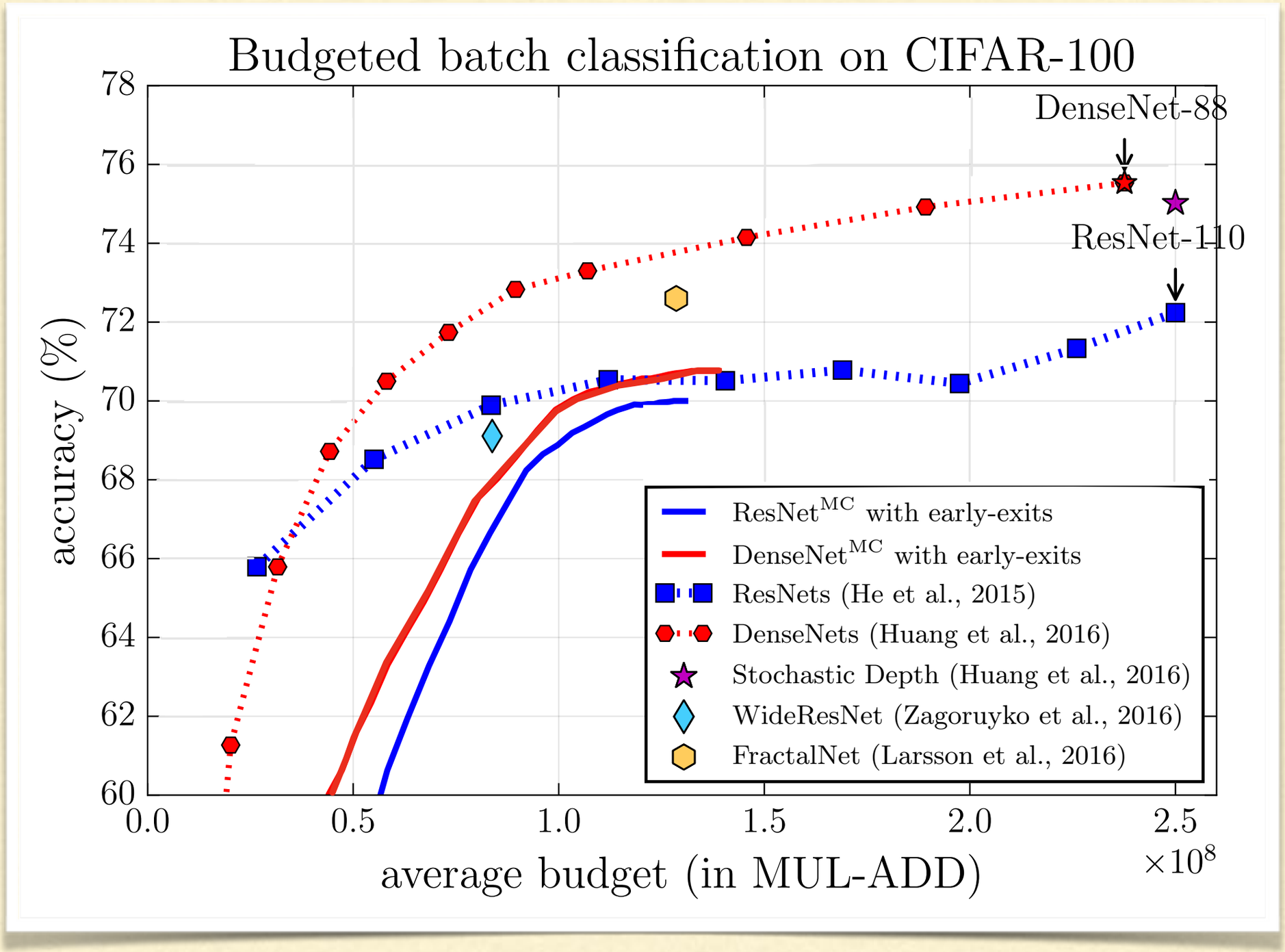
EARLY EXITS


Time
Convolution
Pooling
Convolution
Pooling
Convolution
Pooling
Pooling
Linear
Output

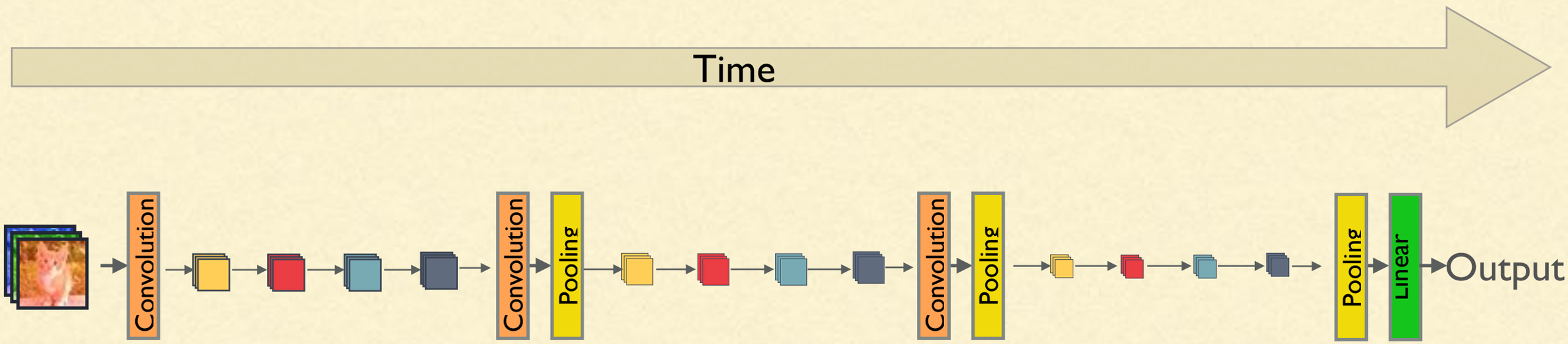
EARLY EXITS


Time
Convolution
Pooling
Convolution
Pooling
Convolution
Pooling
Pooling
Linear
Output


Budgeted batch classification on CIFAR-100
78
76
74
72
70
68
66
64
62
60
accuracy (%)
0.0
0.5
1.0
1.5
2.0
2.5
$\times 10^8$
average budget (in MUL-ADD)
DenseNet-88
ResNet-110
ResNet^{MC} with early-exits
DenseNet^{MC} with early-exits
ResNets (He et al., 2015)
DenseNets (Huang et al., 2016)
Stochastic Depth (Huang et al., 2016)
WideResNet (Zagoruyko et al., 2016)
FractalNet (Larsson et al., 2016)

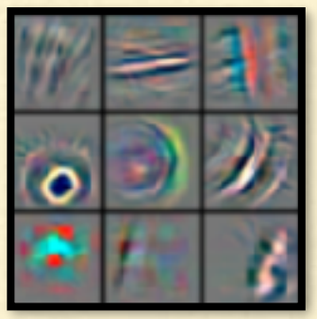

Budgeted batch classification on CIFAR-100
78
76
74
72
70
68
66
64
62
60
accuracy (%)
0.0
0.5
1.0
1.5
2.0
2.5
×10^8
average budget (in MUL-ADD)
DenseNet-88
ResNet-110
ResNet^MC with early-exits
DenseNet^MC with early-exits
ResNets (He et al., 2015)
DenseNets (Huang et al., 2016)
Stochastic Depth (Huang et al., 2016)
WideResNet (Zagoruyko et al., 2016)
FractalNet (Larsson et al., 2016)

EARLY EXITS

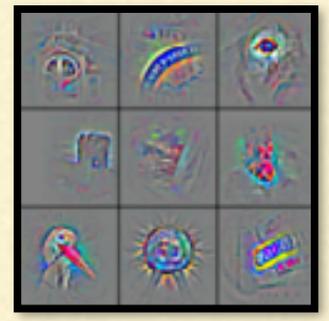


Low-level features

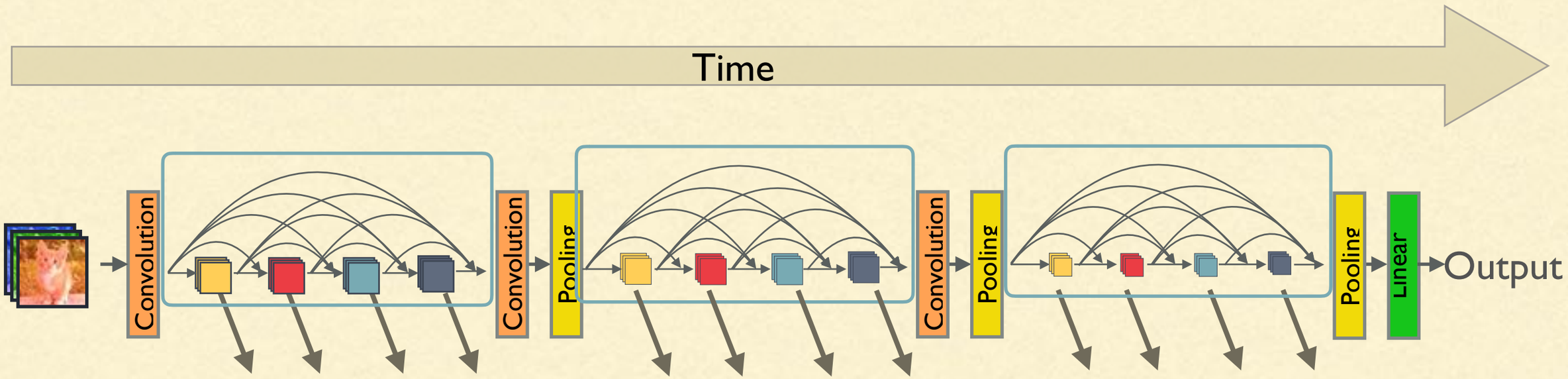
Mid-level features



High-level features

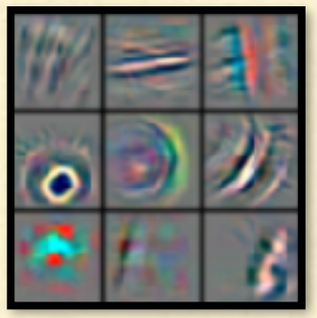


EARLY EXITS

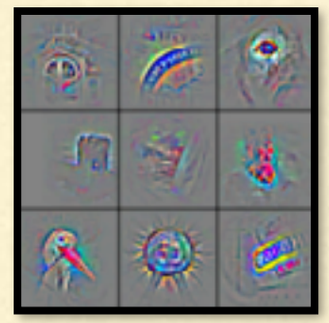


Low-level features

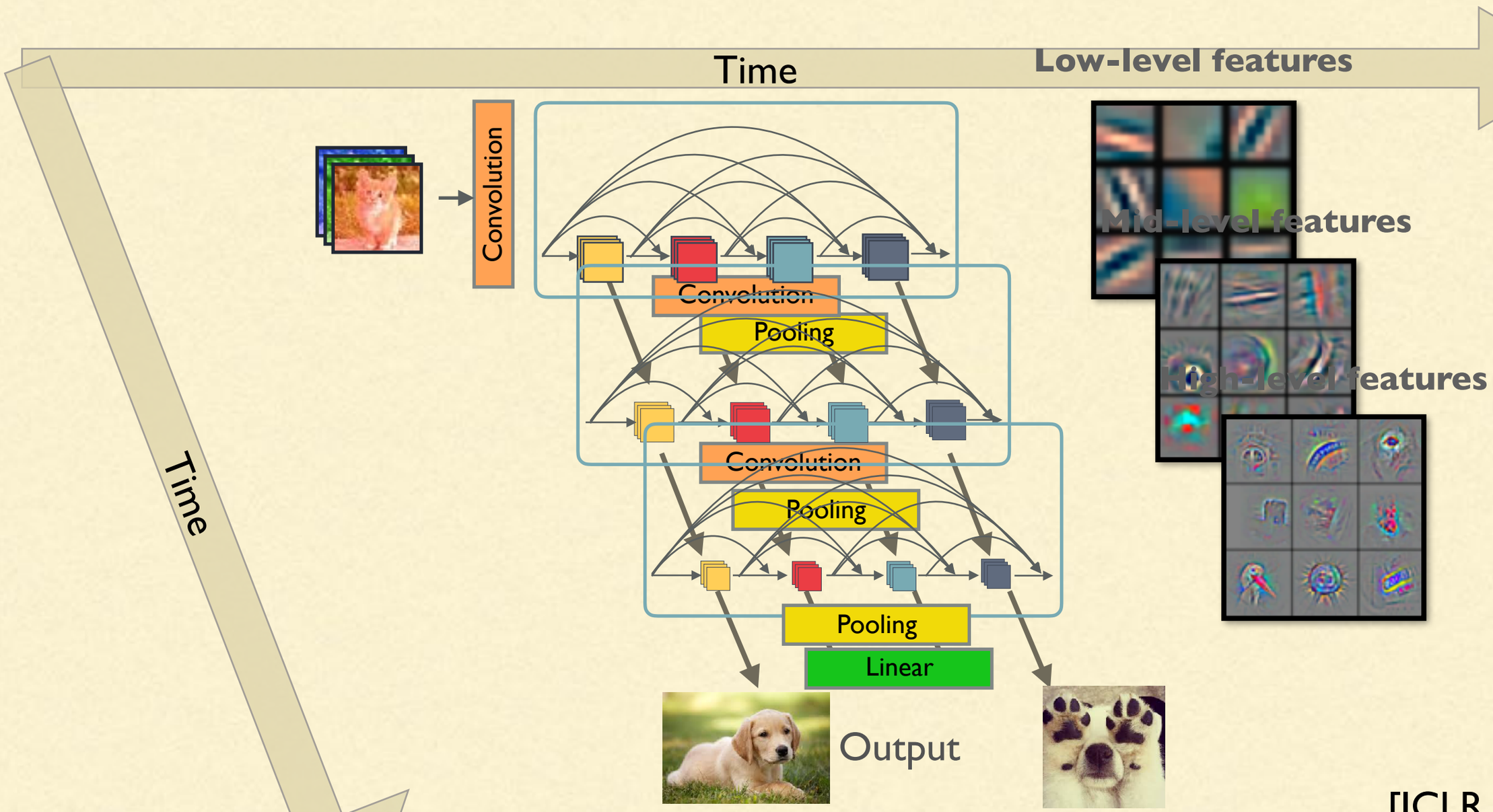
Mid-level features



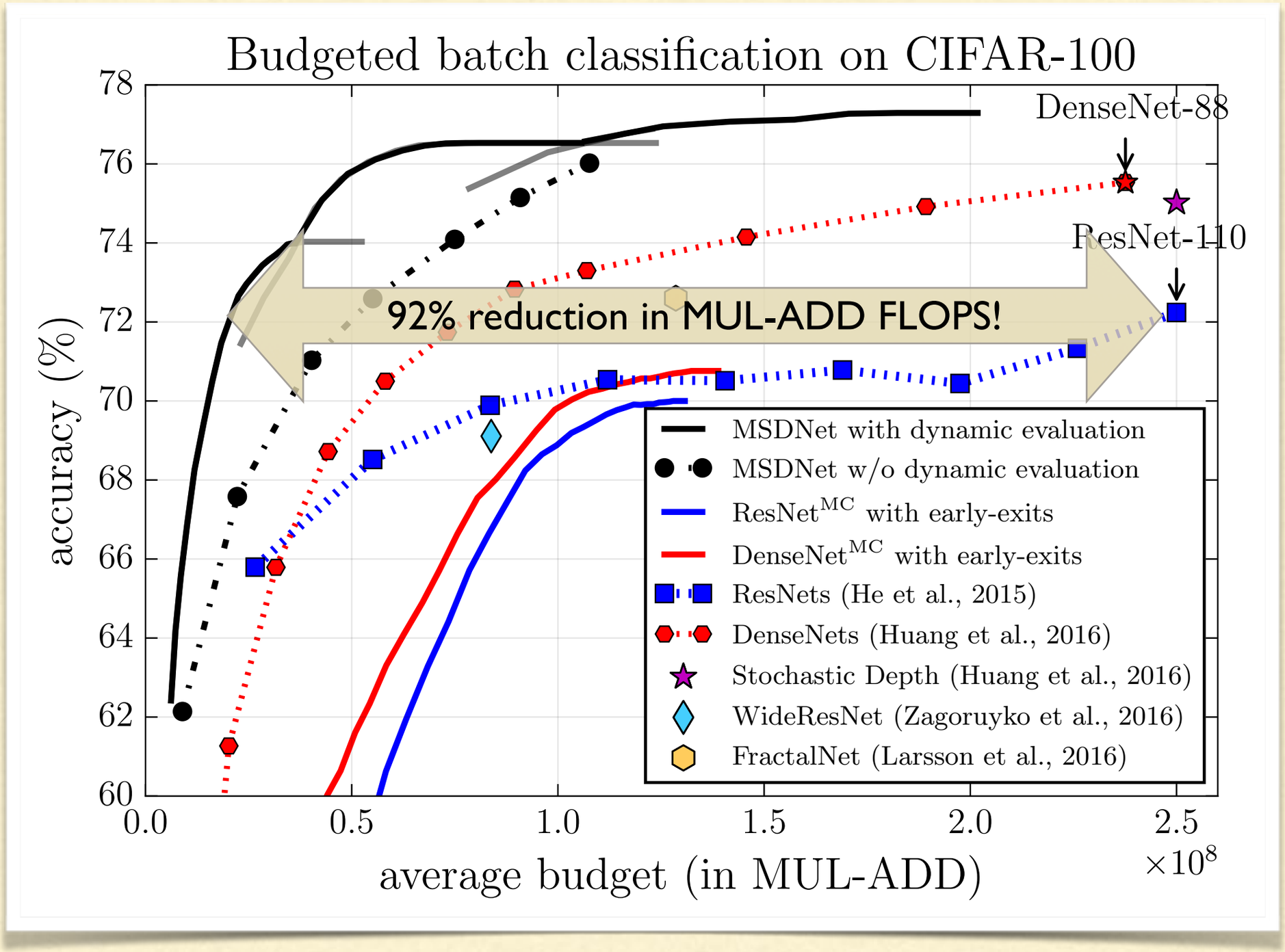
High-level features

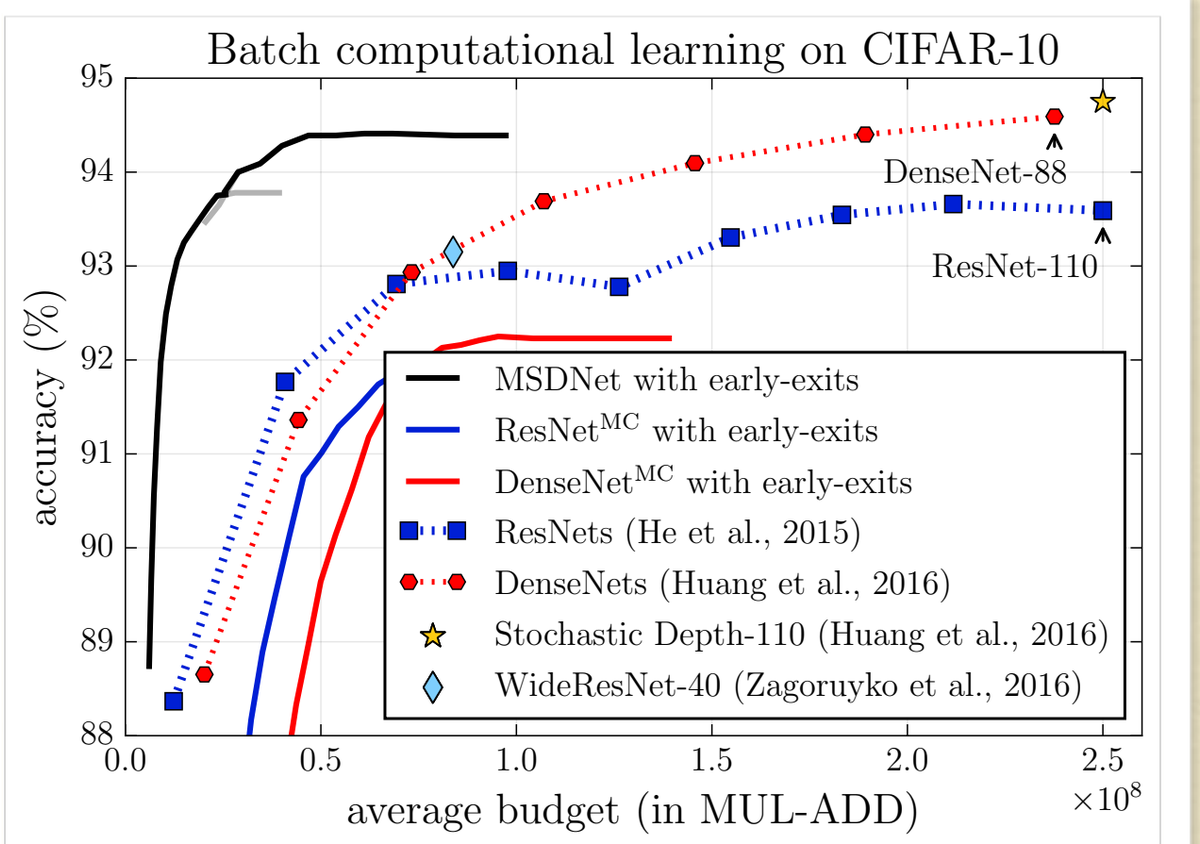


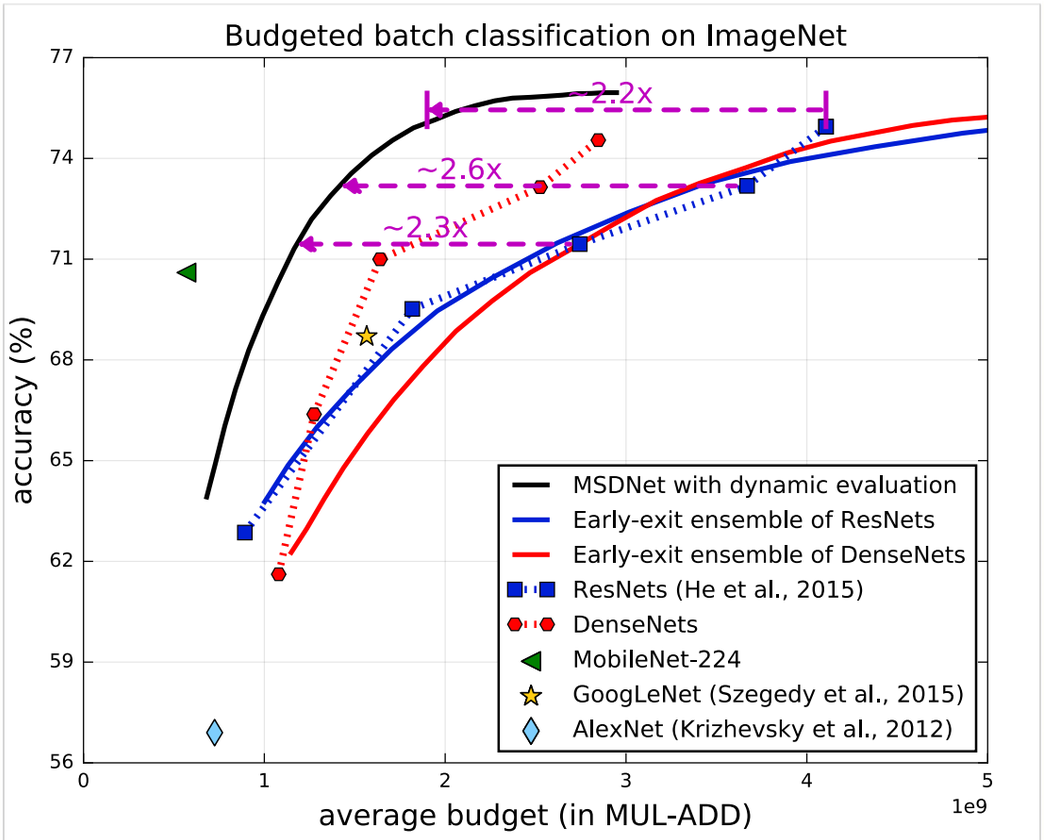
MULTI-SCALE DENSENETS



[ICLR 2018]


Budgeted batch classification on CIFAR-100
DenseNet-88
ResNet-110
92% reduction in MUL-ADD FLOPS!
MSDNet with dynamic evaluation
MSDNet w/o dynamic evaluation
ResNet^MC with early-exits
DenseNet^MC with early-exits
ResNets (He et al., 2015)
DenseNets (Huang et al., 2016)
Stochastic Depth (Huang et al., 2016)
WideResNet (Zagoruyko et al., 2016)
FractalNet (Larsson et al., 2016)
78
76
74
72
70
68
66
64
62
60
0.0
0.5
1.0
1.5
2.0
2.5
×10^8
average budget (in MUL-ADD)

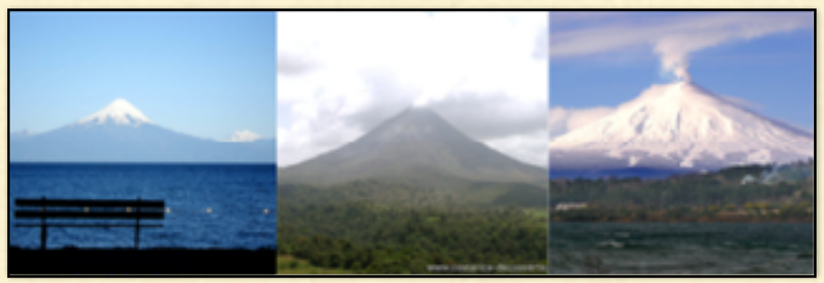

Batch computational learning on CIFAR-10
accuracy (%)
average budget (in MUL-ADD)
×10^8
DenseNet-88
ResNet-110
MSDNet with early-exits
ResNet^MC with early-exits
DenseNet^MC with early-exits
ResNets (He et al., 2015)
DenseNets (Huang et al., 2016)
Stochastic Depth-110 (Huang et al., 2016)
WideResNet-40 (Zagoruyko et al., 2016)


Budgeted batch classification on ImageNet
accuracy (%)
average budget (in MUL-ADD)
1e9
~2.2x
~2.6x
~2.3x
MSDNet with dynamic evaluation
Early-exit ensemble of ResNets
Early-exit ensemble of DenseNets
ResNets (He et al., 2015)
DenseNets
MobileNet-224
GoogLeNet (Szegedy et al., 2015)
AlexNet (Krizhevsky et al., 2012)

EASY AND HARD EXAMPLES

red wine

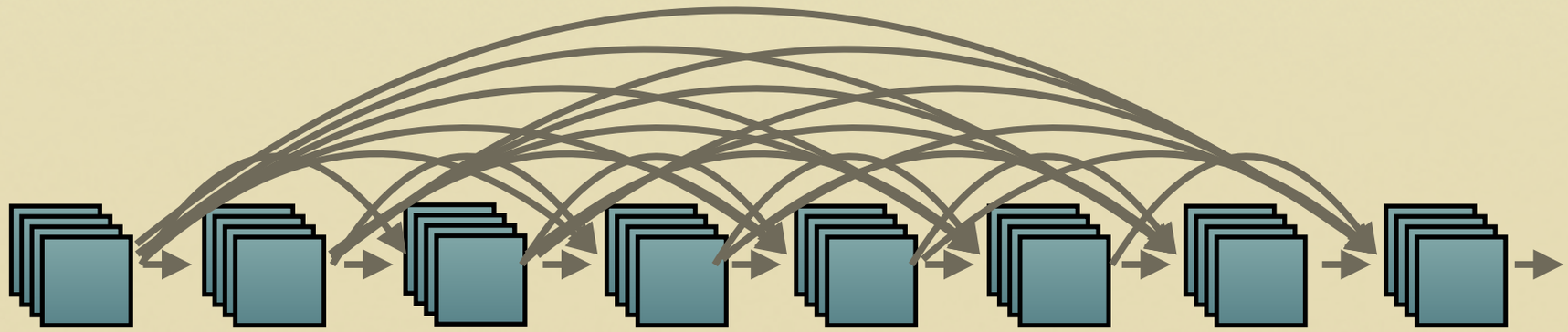
volcano



"*easy*"
(exit at **first** stage)

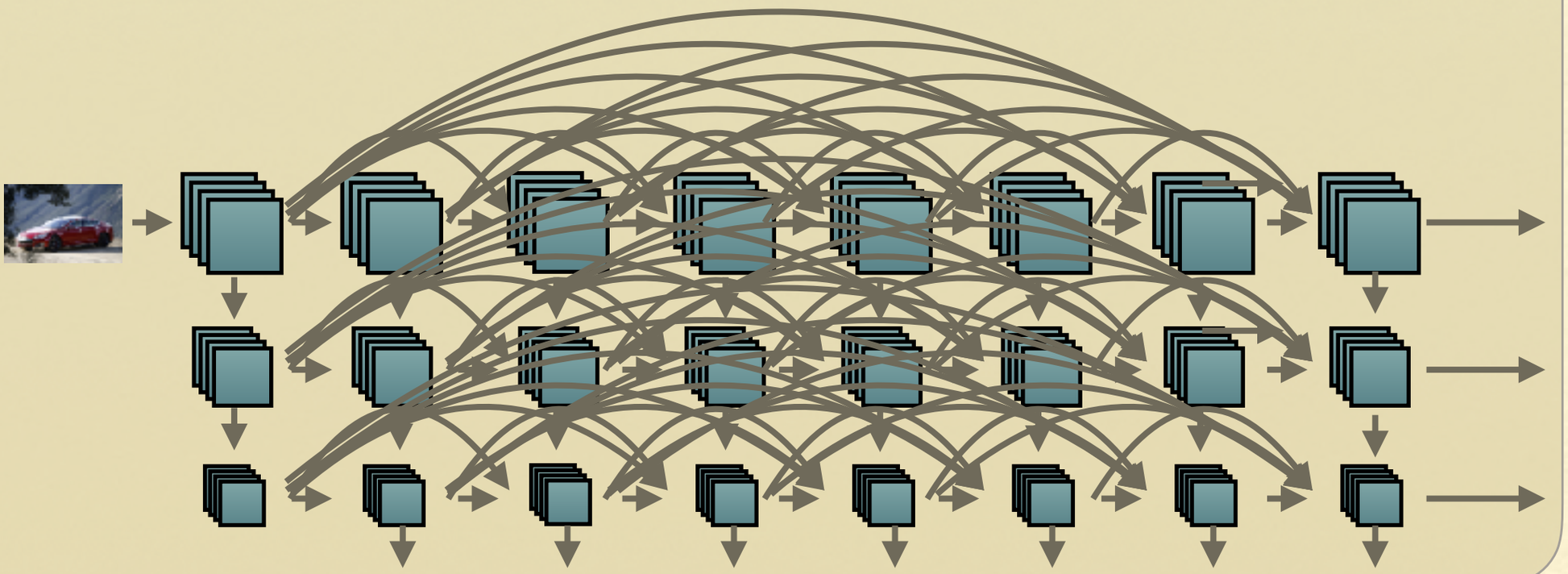
"*hard*"
(exit at **last** stage)

Try out Dense Connectivity!

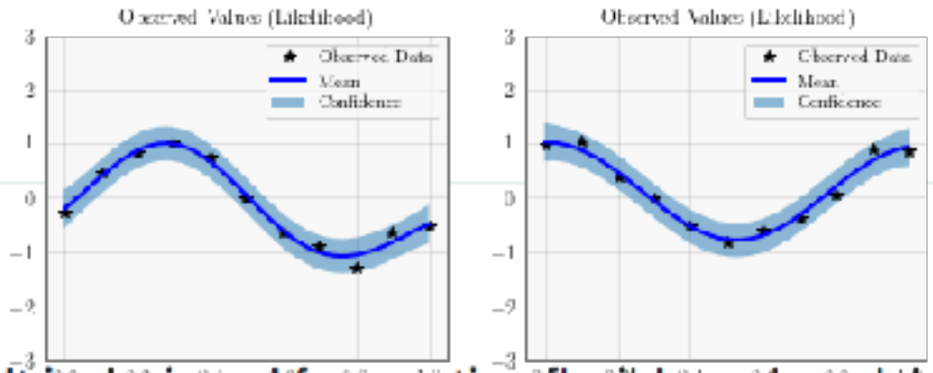


- **Explicit long term connections**
- **Best generalization performance**

Save the planet with MSD-Nets!



GPyTorch (Alpha Release)



build passing

GPyTorch is a Gaussian Process library, implemented using PyTorch. It is designed for creating flexible and modular Gaussian Process models with ease, so that you don't have to be an expert to use GPs.

This package is currently under development, and is likely to change. Some things you can do right now:

- Simple GP regression (example here)
- Simple GP classification (example here)
- Multitask GP regression (example here)
- Scalable GP regression using kernel interpolation (example here)
- Scalable GP classification using kernel interpolation (example here)
- Deep kernel learning (example here)
- And (more!)

If you use GPyTorch, please cite the following papers:

> Gardner, Jacob R., Geoff Pleiss, Ruihan Wu, Kilian Q. Weinberger, and Andrew Gordon Interpolation for Scalable Gaussian Processes." In *AISTATS* (2018).

Constant-Time Predictive Distributions for Gaussian Processes

Geoff Pleiss [1] Jacob R. Gardner [1] Kilian Q. Weinberger [1] Andrew Gordon Wilson [1]

Abstract

One of the most compelling features of Gaussian process (GP) regression is its ability to provide well calibrated posterior distributions. Recent advances in inducing point methods have drastically sped up marginal likelihood and posterior mean computations, leaving posterior covariance estimation and sampling as the remaining computational bottlenecks. In this paper we address this shortcoming by using the Lanczos decomposition algorithm to rapidly approximate the predictive covariance matrix. Our approach, which we refer to as LOVE (LanczOs Variance Estimates), substantially reduces the time and space complexity over any previous method. In practice, it can compute predictive covariances up to 2,000 times faster and draw samples 18,000 time faster than existing methods, all *without* sacrificing accuracy.

1. Introduction

Gaussian processes (GPs) are fully probabilistic models which can naturally estimate predictive uncertainty through posterior variances. These uncertainties play a pivotal role in many application domains. For example, uncertainty information is crucial when incorrect predictions could have catastrophic consequences, such as in medicine (Schulam & Saria, 2017) or large-scale robotics (Deisenroth et al., 2015); Bayesian optimization approaches typically incorporate model uncertainty when choosing actions (Snoek et al., 2012; Deisenroth & Rasmussen, 2011; Wang & Jegelka, 2017); and reliable uncertainty estimates are arguably useful for establishing trust in predictive models, especially when predictions would be otherwise difficult to interpret (Doshi-Velez & Kim, 2017; Zhou et al., 2017).

Although predictive uncertainties are a primary advantage of GP models, they have recently become their primary computational bottleneck. Historically, this has not always been the case. The use of GPs used to be limited to problems with small datasets, since learning and inference computations naïvely scale cubically with the number of data points (n). However, recent advances in *inducing point methods* have managed to scale up GPs to much larger datasets (Snelson & Ghahramani, 2006; Quiñonero-Candela & Rasmussen, 2005; Titsias, 2009). For example, *Kernel Interpolation for Scalable Structured GPs* (KISS-GP) scales to millions of data points (Wilson & Nickisch, 2015; Wilson et al., 2015). For a given test point $\mathbf{x}^*$, KISS-GP expresses the GP's predictive mean as $\mathbf{a}^\top \mathbf{w}(\mathbf{x}^*)$, where $\mathbf{a}$ is a pre-computed vector dependent only on training data, and $\mathbf{w}(\mathbf{x}^*)$ is a sparse interpolation vector. This particular formulation affords the ability to compute predictive means in *constant time*, independent of n.

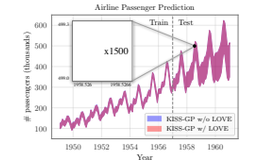


Figure 1. Comparison of predictive variances on airline passenger extrapolation. The variances predicted with LOVE are accurate within 10^{-4}, yet can be computed orders of magnitude faster.

However, these computational savings do not extend naturally to predictive uncertainties. With KISS-GP, computing the predictive covariance between two test points requires $O(n + m \log m)$ computations, where m is the number of inducing points used (see Table 1). While this asymptotic complexity is lower than that of standard Gaussian process inference, it quickly becomes prohibitive when n is large, or when we wish to make many repeated computations. Additionally, drawing samples from the predictive distributions – a necessary operation in many applications – is similarly expensive. Matching the reduced complexity of predictive mean inference has remained an open problem.

In this paper, we provide a solution based on the tridi-

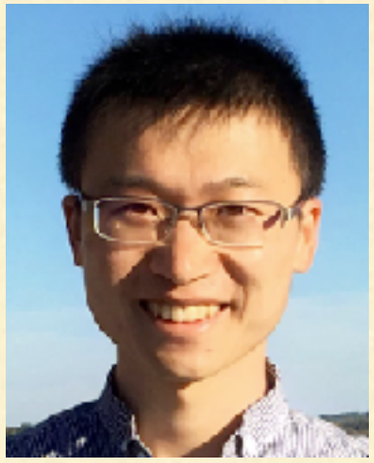
[1]Cornell University. Correspondence to: Geoff Pleiss <geoff@cs.cornell.edu>, Jacob R. Gardner <jrg365@cornell.edu>, Andrew Gordon Wilson <andrew@cornell.edu>.

[NIPS 2018?, ICML 2018, AISTATS 2017]

SPONSORS

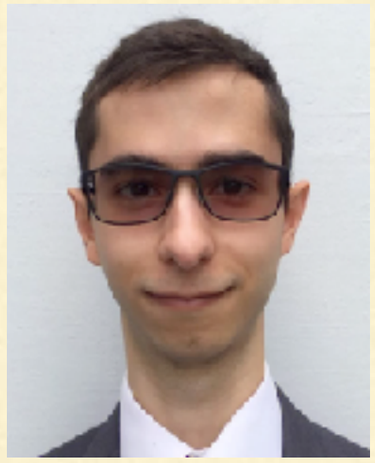
[If time click here]

THANKS TO …

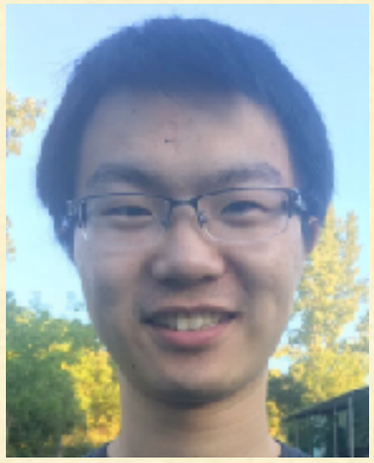


Gao Huang
Cornell

Yu Sun
U.C. Berkeley



Dan Sedra
Amazon



Zhuang Liu
U.C. Berkeley

Laurens v.d. Maaten
Facebook

Noah Snavely
Cornell

Geoff Pleiss
Cornell

Paul Upchurch
Cornell

Kavita Bala
Cornell

Robert Pless
G. W. U.

Jake Gardner
Cornell

… and Danlu Chen, Tianhong Li, Felix Wu