

DEEP LEARNING

Philipp Krähenbühl

ADMINISTRATIVE

- Presentation list (soon) online

THE DEEP LEARNING CONSPIRACY

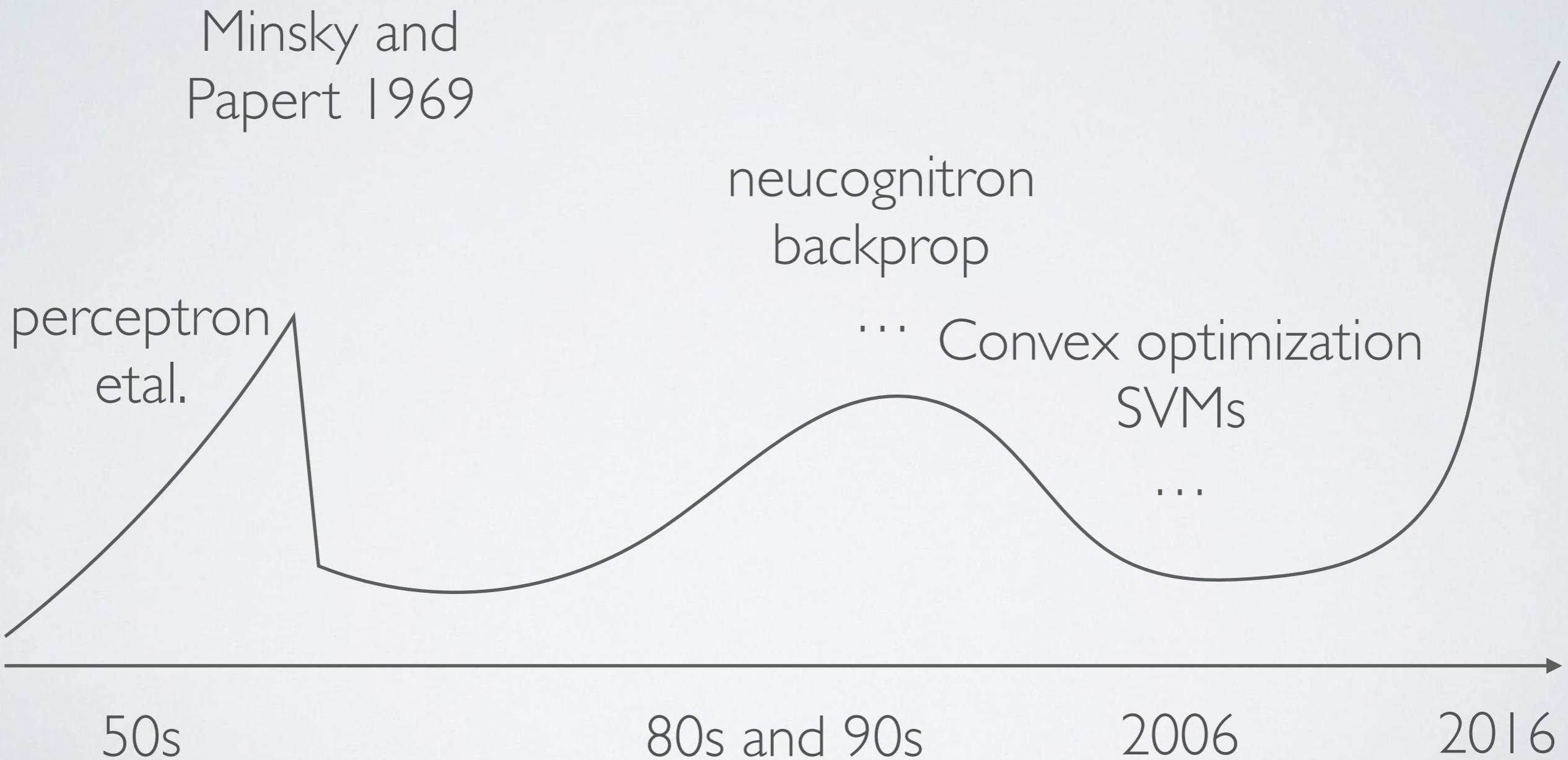
REVIEW

doi:10.1038/nature14539

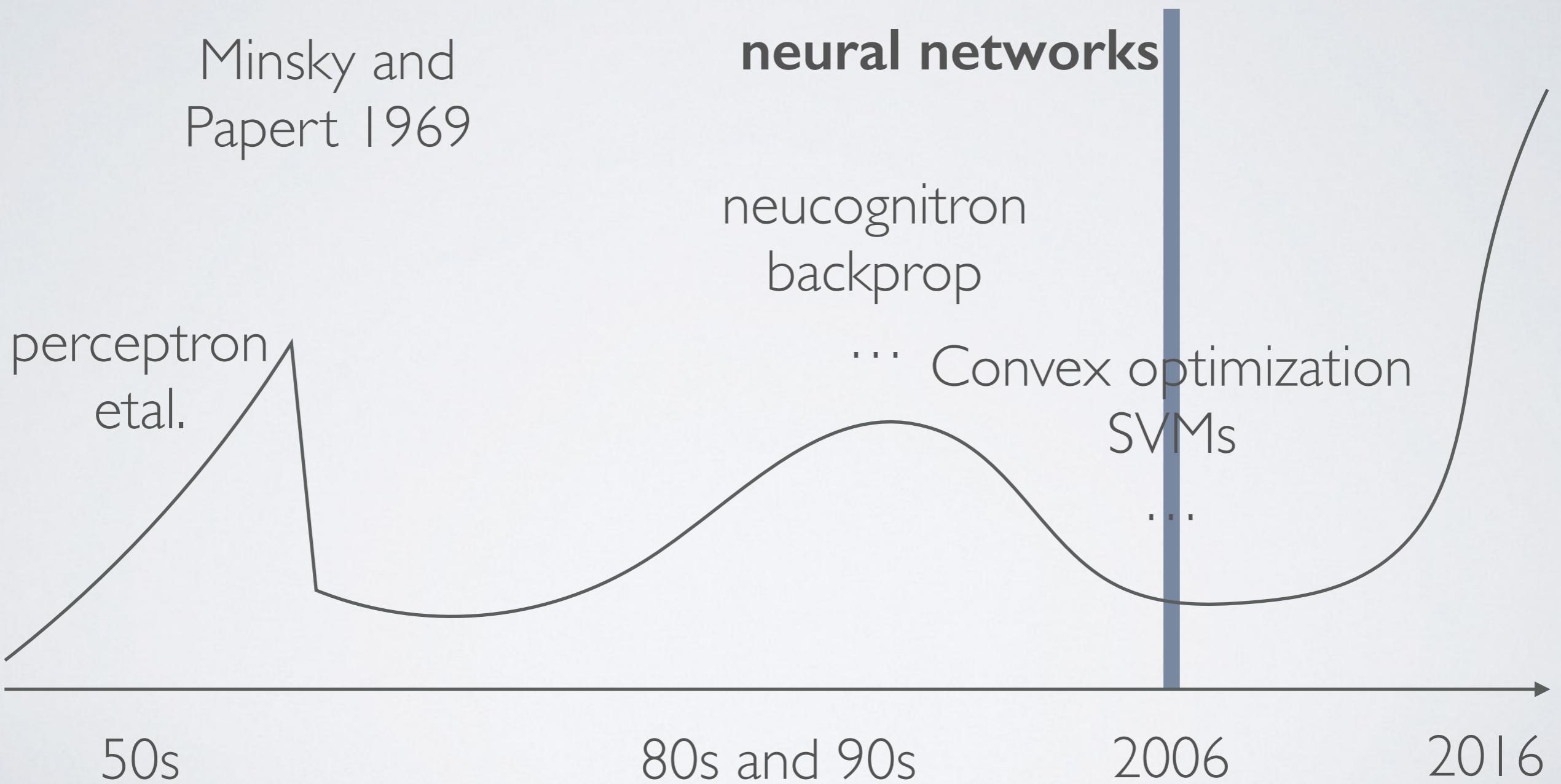
Deep learning

Yann LeCun^{1,2}, Yoshua Bengio³ & Geoffrey Hinton^{4,5}

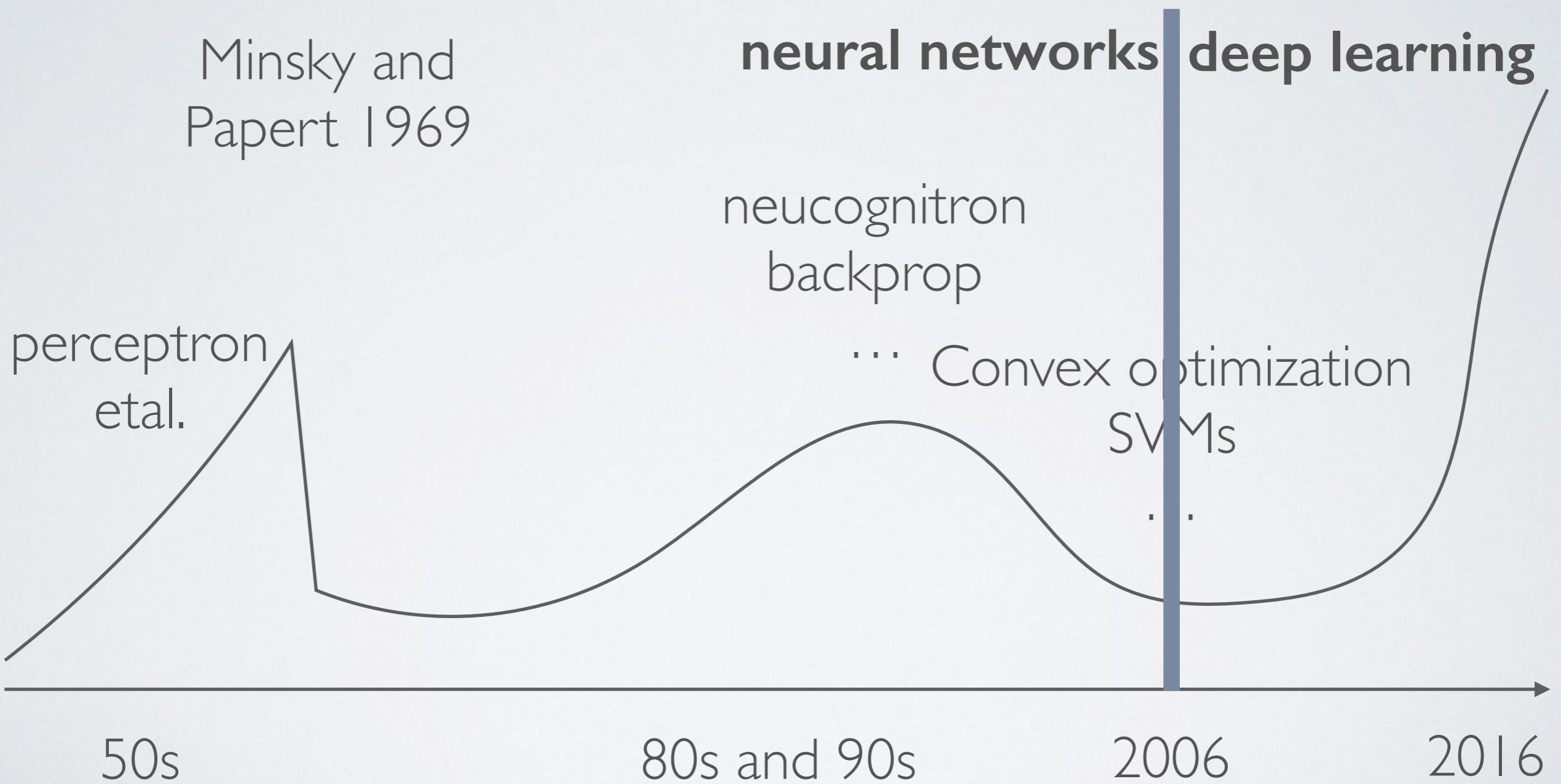
THE DEEP LEARNING CONSPIRACY



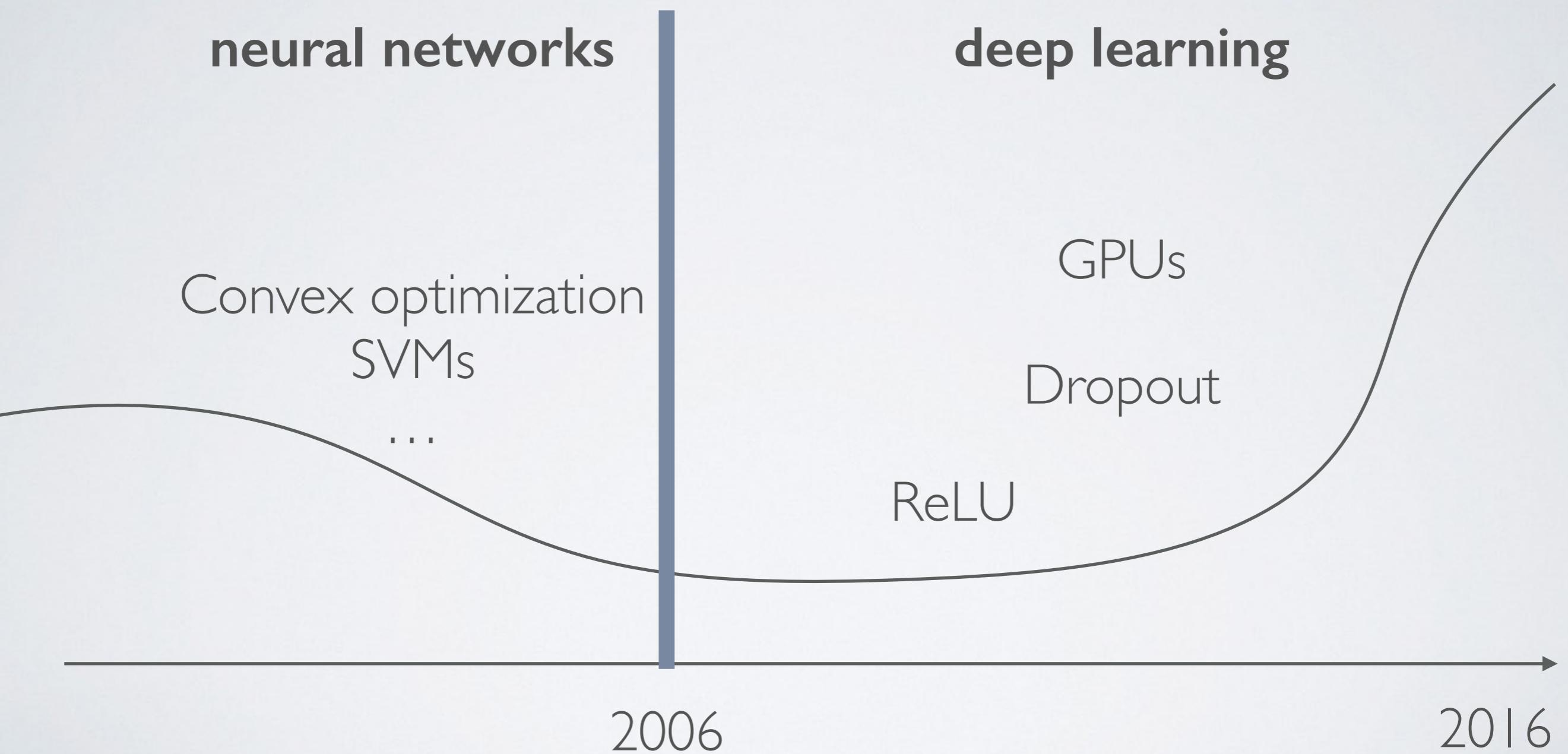
THE DEEP LEARNING CONSPIRACY



THE DEEP LEARNING CONSPIRACY



THE DEEP LEARNING CONSPIRACY



THE DEEP LEARNING CONSPIRACY

30 days of = x days of



THE DEEP LEARNING CONSPIRACY

30 c

ars of



THE DEEP LEARNING CONSPIRACY

30 days of = x years of



THE DEEP LEARNING CONSPIRACY

30 days of = 26 years of



WHY DEEP LEARNING?

traditional models

learned

(linear) classification

hand designed

Feature extraction

deep models

learned jointly

learned jointly



DESIGNING FEATURES IS HARD



SHALLOW OR LINEAR CLASSIFIERS NOT ENOUGH



A linear classifier, or any other ‘shallow’ classifier operating on raw pixels could not possibly distinguish the latter two, while putting the former two in the same category.



SHALLOW OR LINEAR CLASSIFIERS NOT ENOUGH

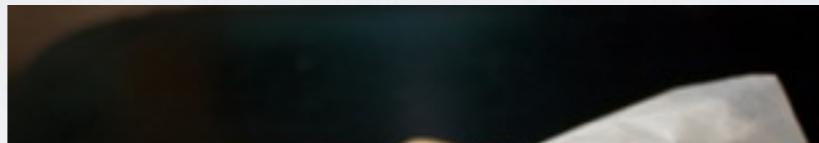
```
from sklearn import svm
c = svm.LinearSVC()

N,D = 1000, 2000
for it in range(10):
    # Generate some random features
    f = np.random.normal(size=(2*N,D))
    l = np.arange(2*N) < N
    # How many are linearly seperable
    print( c.fit(f,l).score(f,l) )
```

SHALLOW OR LINEAR CLASSIFIERS NOT ENOUGH

1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0

SHALLOW OR LINEAR CLASSIFIERS NOT ENOUGH

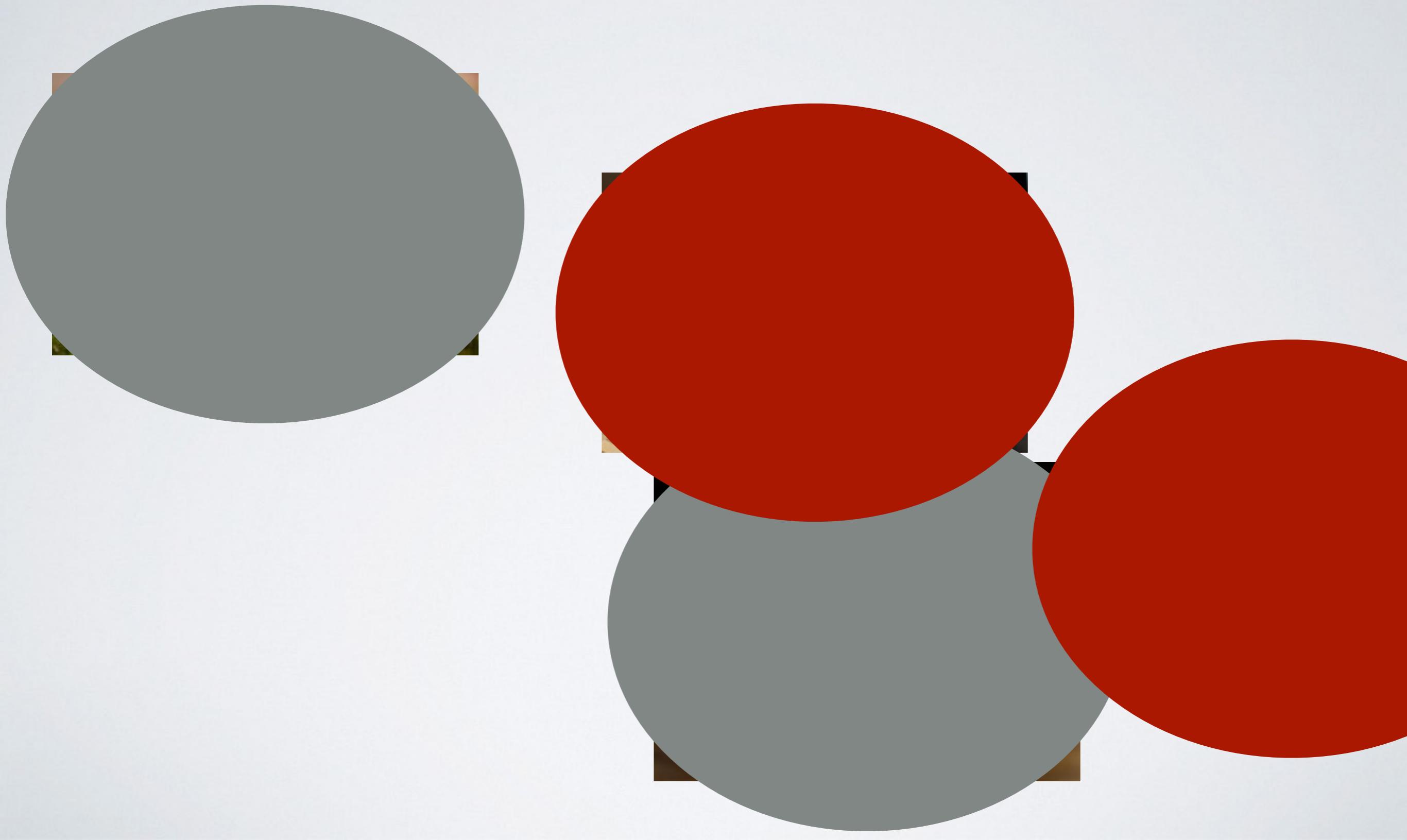


**might not hold in high dimensional
spaces such as images**

A linear classifier, or any other ‘shallow’ classifier operating on raw pixels could not possibly distinguish the latter two, while putting the former two in the same category.



KERNELS



JOINTLY LEARNING THE FEATURE SPACE



WHY DEEP LEARNING?

traditional models

learned

(linear) classification

hand designed

Feature extraction

deep models

learned jointly

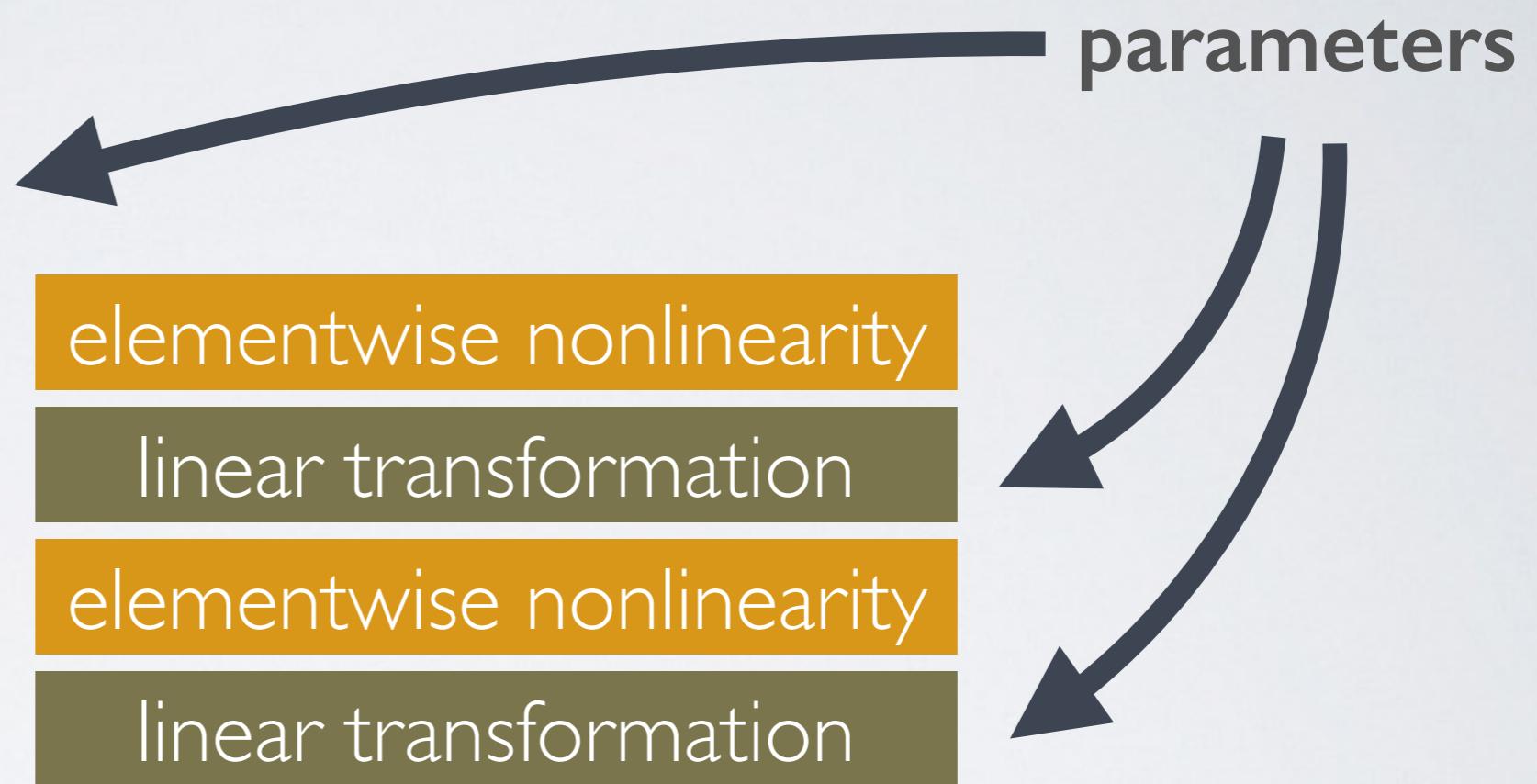
learned jointly



TRAINING

(linear) classification

Feature extraction



TRAINING

(linear) classification



Feature extraction

loss $l(y)$

$$y = A_3 z_4 + b_3$$

elementwise nonlinearity

linear transformation

elementwise nonlinearity

linear transformation

$$z_4 = \rho(z_3)$$

$$z_3 = A_2 z_2 + b_2$$

$$z_2 = \rho(z_1)$$

$$z_1 = A_1 x + b_1$$

$$\frac{\partial l}{\partial b_3} = \frac{\partial l}{\partial y} \cdot \frac{\partial y}{\partial b_3}$$

(linear) classification

Feature extraction



loss $l(y)$ $\frac{\partial l}{\partial y}$

$y = A_3 z_4 + b_3$ $\frac{\partial l}{\partial z_4}$

elementwise nonlinearity

$z_4 = \rho(z_3)$

linear transformation

$z_3 = A_2 z_2 + b_2$

elementwise nonlinearity

$z_2 = \rho(z_1)$

linear transformation

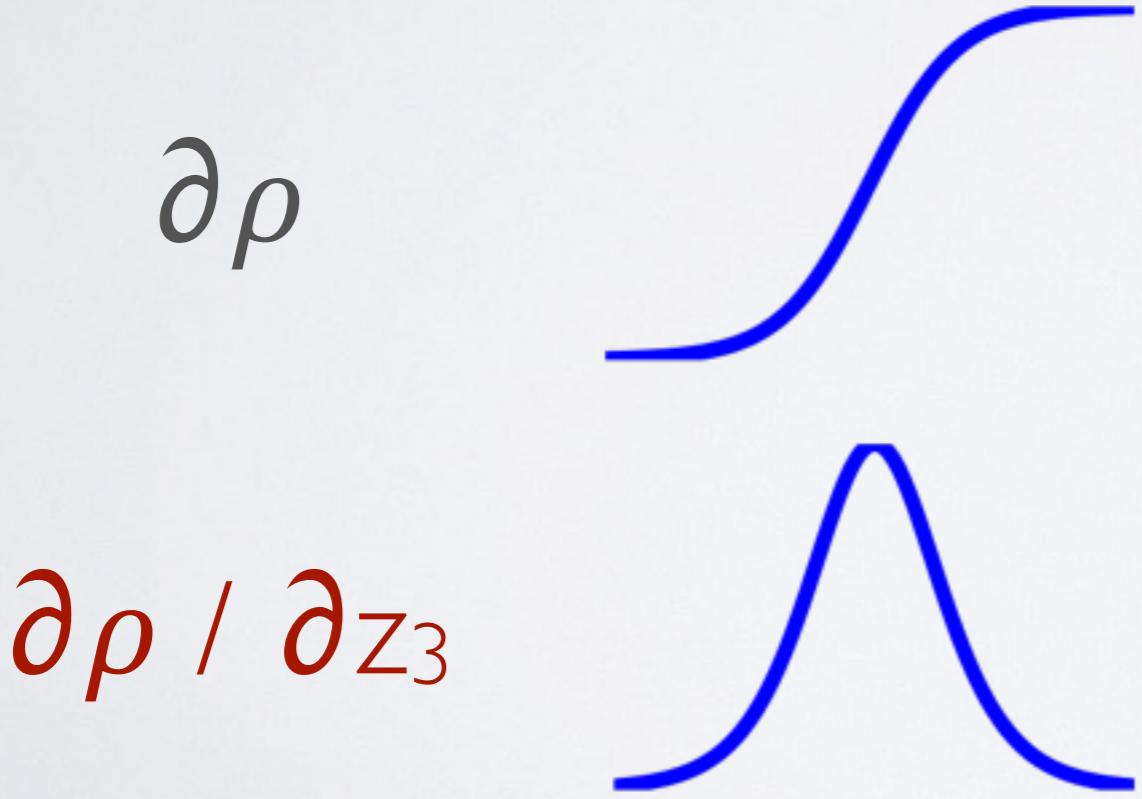
$z_1 = A_1 x + b_1$

ReLU

$$z_4 = \rho(z_3)$$

$$\partial l / \partial z_3 = \partial l / \partial z_4 \cdot \partial \rho / \partial z_3$$

sigmoid



ReLU



CONVNETS

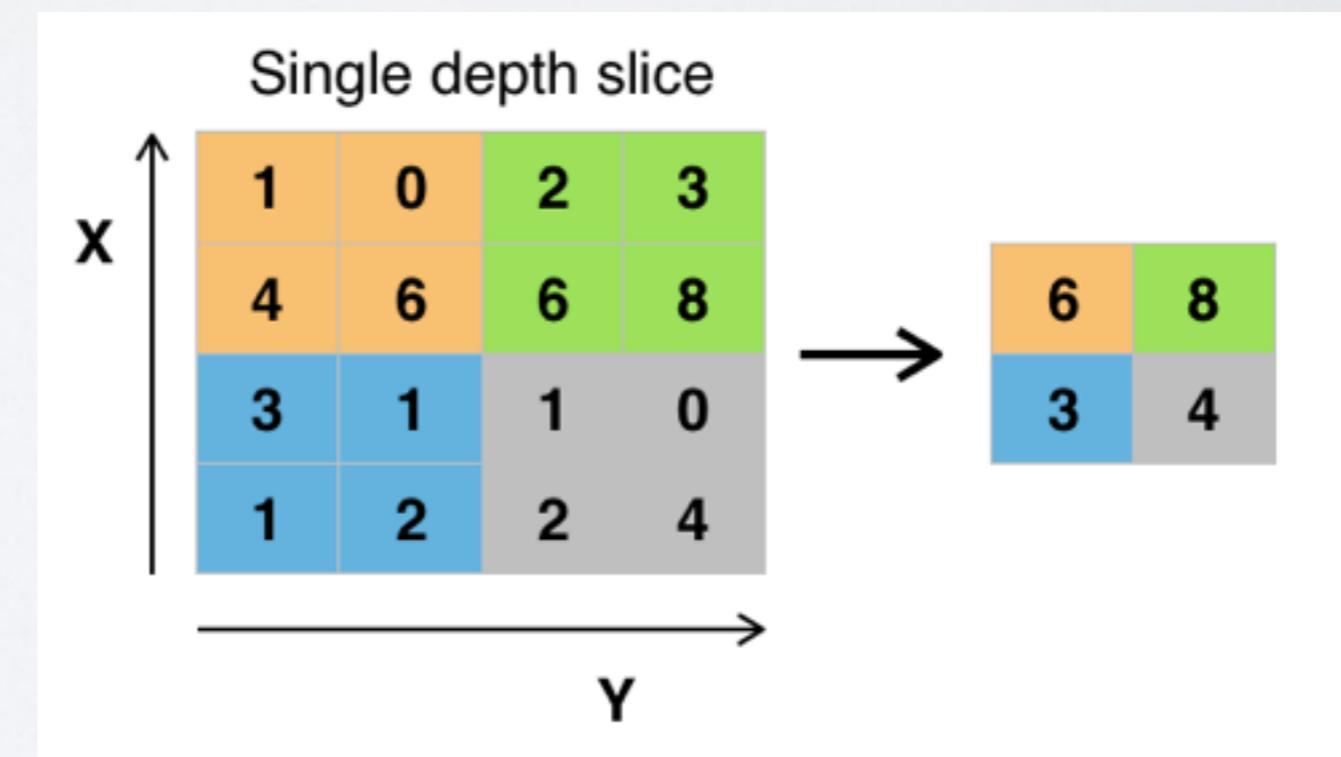
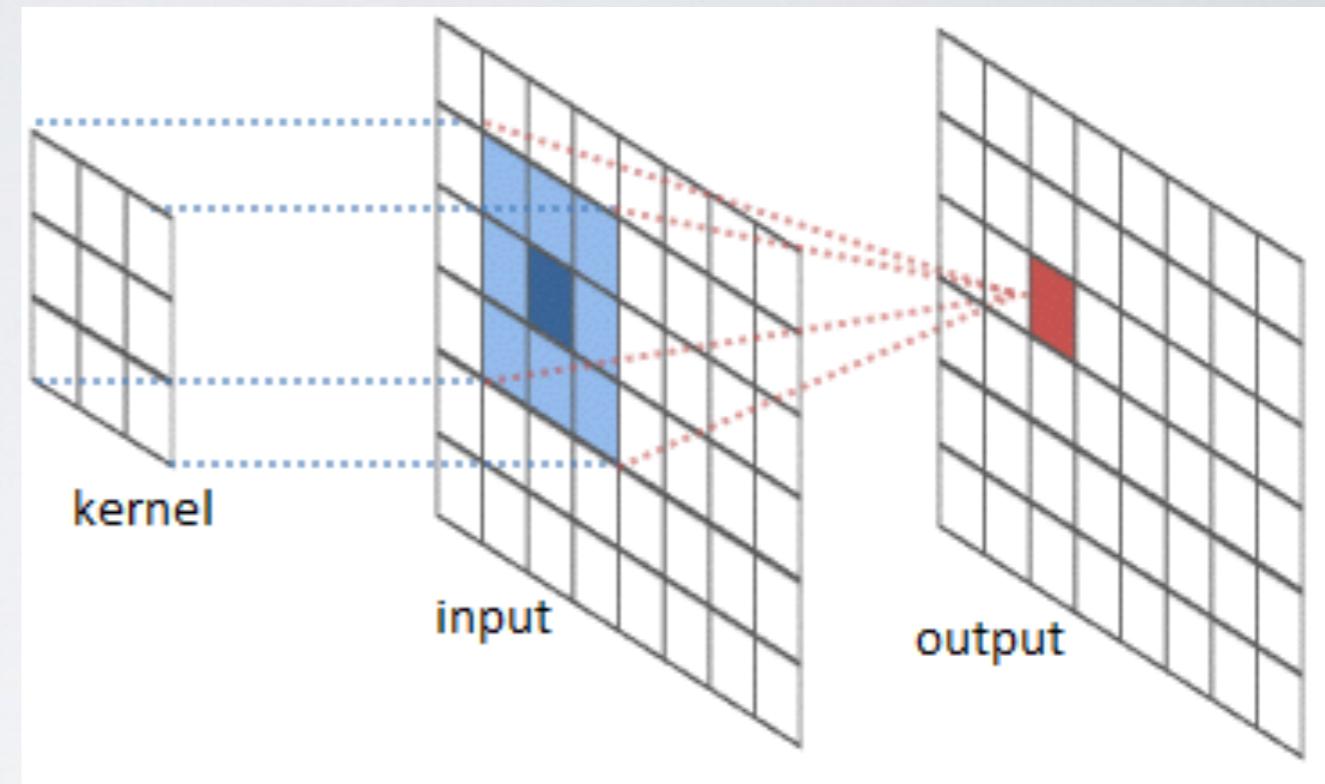
(linear) classification

elementwise nonlinearity

linear transformation

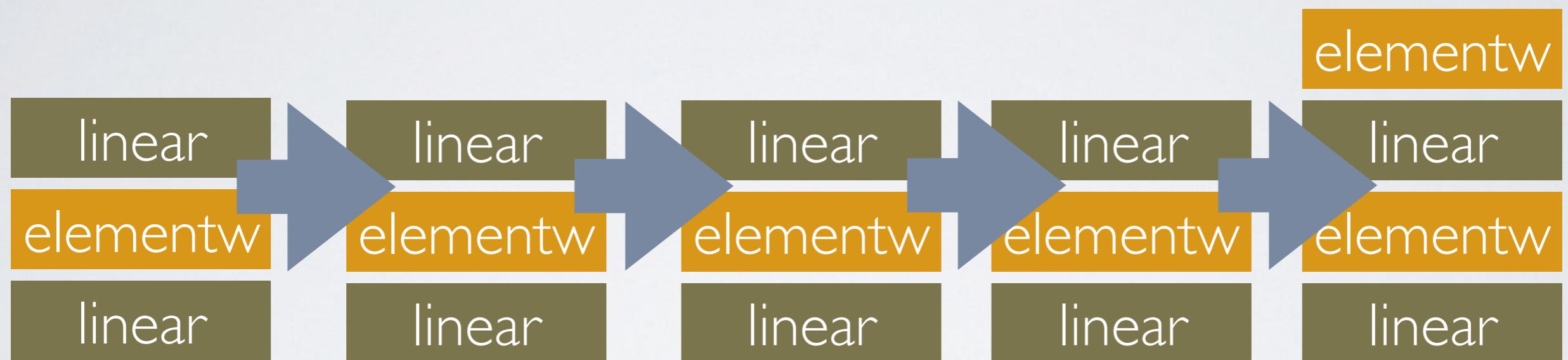
elementwise nonlinearity

linear transformation



RNNNS

(linear) classification



H

e

|

|

O

DISCUSSION

HOW TO PRESENT A PAPER

- Email your slides to Philipp one week in advance
- Read and review the paper carefully

is covariate shift important?

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
Sergey Ioffe, Christian Szegedy
Google Inc., szefi@google.com, Google Inc., szegedy@google.com

Abstract
Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities. We refer to this phenomenon as *internal covariate shift*. In this paper we propose a new technique to combat inputs. Our method draws its strength from making normalization a part of the model architecture and performing the normalization *for each training mini-batch*. Batch Normalization allows us to use much higher learning rates and be less careful about initialization. It also acts as a regularizer, in some cases eliminating the need for Dropout.

Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities. We refer to this phenomenon as *internal covariate shift*. In this paper we propose a new technique to combat inputs. Our method draws its strength from making normalization a part of the model architecture and performing the normalization *for each training mini-batch*. Batch Normalization allows us to use much higher learning rates and be less careful about initialization. It also acts as a regularizer, in some cases eliminating the need for Dropout.

Batch Normalization achieves the same accuracy with 14 times fewer training steps, and beats the original model by a wide margin. Using an equivalent number of training steps, we improve over the best published result on ImageNet classification, reaching 4.9% top-5 error (and 4.8% test error), exceeding the accuracy of human raters.

1 Introduction
Deep learning has dramatically advanced the state-of-the-art in vision, speech, and many other areas. Stochastic gradient descent (SGD) has proved to be an effective way of training deep networks, and SGD variants such as momentum (Sutskever et al., 2013) and Adagrad (Duchi et al., 2011) have been used to achieve state-of-the-art performance. SGD optimizes the parameters Θ of the network, so as to minimize the loss

$$\Theta = \arg \min_{\Theta} \frac{1}{N} \sum_{i=1}^N l(x_i, \Theta)$$

where $x_{1..N}$ is the training data set. With SGD, the training proceeds in steps, and at each step we consider a *mini-batch* $x_{1..m}$ of size m . The mini-batch is used to approximate the gradient of the loss function with respect to the parameters, by computing

$$\frac{1}{m} \frac{\partial l(x_i, \Theta)}{\partial \Theta}$$

(for batch size m and learning rate α) is exactly equivalent to that for a stand-alone network F_2 with input x . Therefore, if input distributions change, then the training will be effected by having the same distribution between the training and test data – apply to training the sub-network as well. As such it is advantageous for the distribution of x to remain fixed over time. Then, Θ_2 does

how about this baseline?

SHALLOW OR LINEAR CLASSIFIERS NOT ENOUGH

```
from sklearn import svm
c = svm.LinearSVC()

N,D = 1000, 2000
for it in range(10):
    # Generate some random features
    f = np.random.normal(size=(2*N,D))
    l = np.arange(2*N) < N
    # How many are linearly seperable
    print( c.fit(f,l).score(f,l) )
```

HAVE A STORY

- Motivate well
- Provide context

REVIEW

doi:10.1038/nature14539

Deep learning

Yann LeCun^{1,2}, Yoshua Bengio³ & Geoffrey Hinton^{4,5}

BE VISUAL

every slide should have a picture



A picture is worth a thousand words.

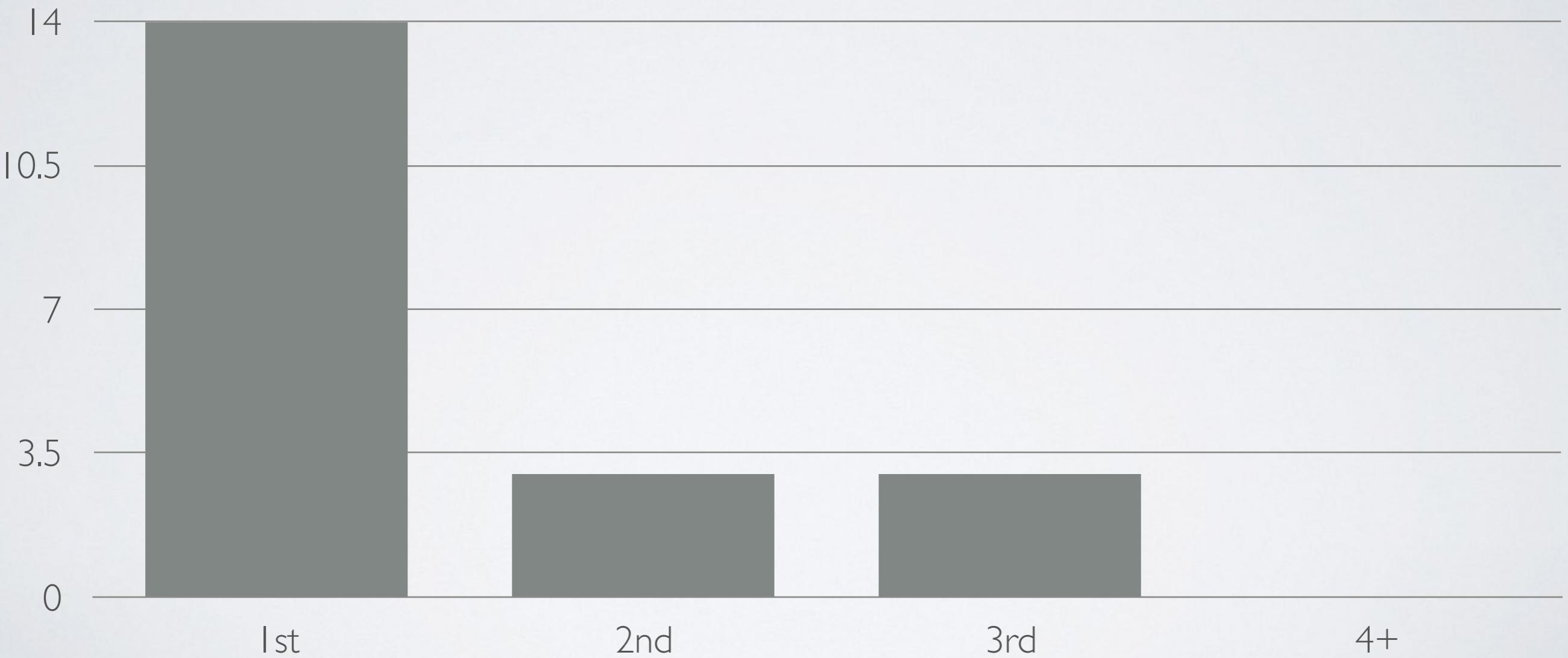
DESIGNING FEATURES IS HARD



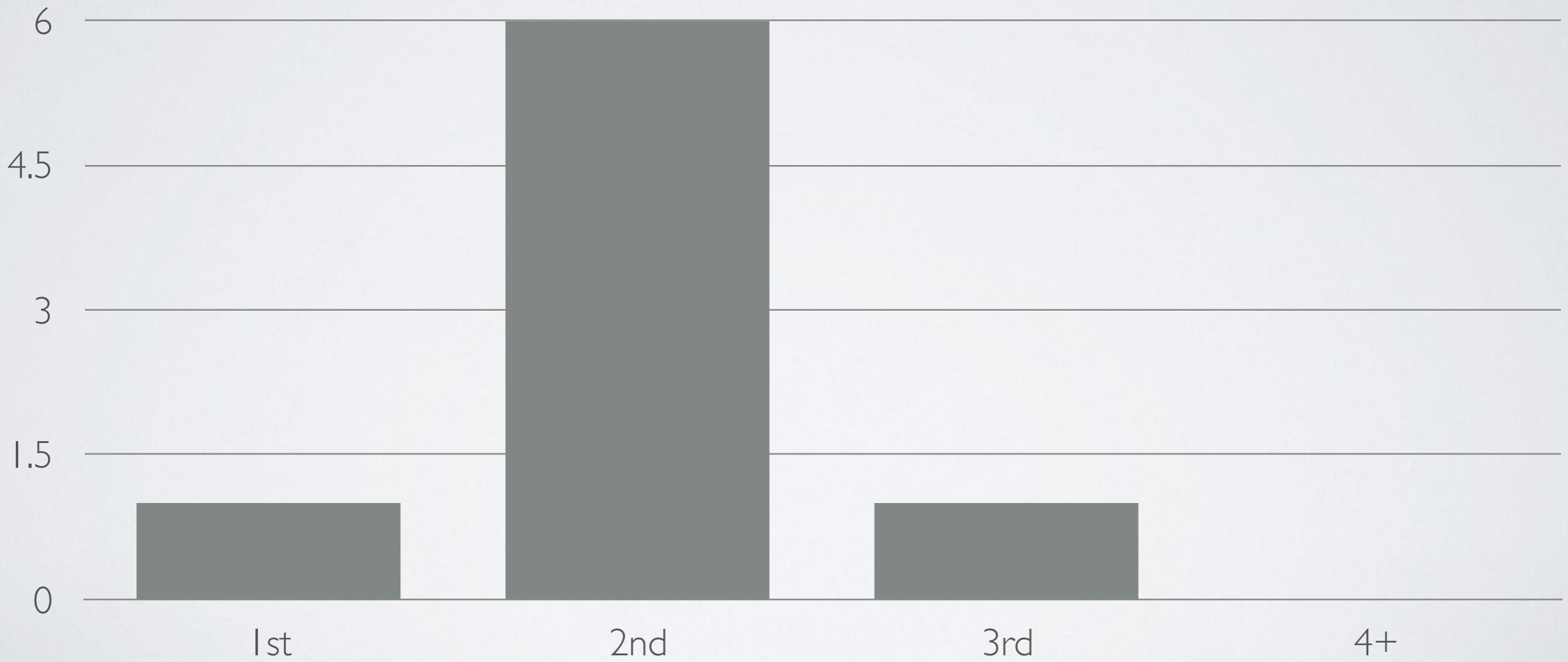
BE INTERACTIVE



PAPER ASSIGNMENT



PAPER ASSIGNMENT



PAPER ASSIGNMENT

Posted online later today

SADDLE POINTS

Identifying and attacking the saddle point problem in high-dimensional non-convex optimization

Yann N. Dauphin Razvan Pascanu Caglar Gulcehre Kyunghyun Cho

Université de Montréal

dauphiya@iro.umontreal.ca, r.pascanu@gmail.com,
gulcehrc@iro.umontreal.ca, kyunghyun.cho@umontreal.ca

Surya Ganguli

Stanford University

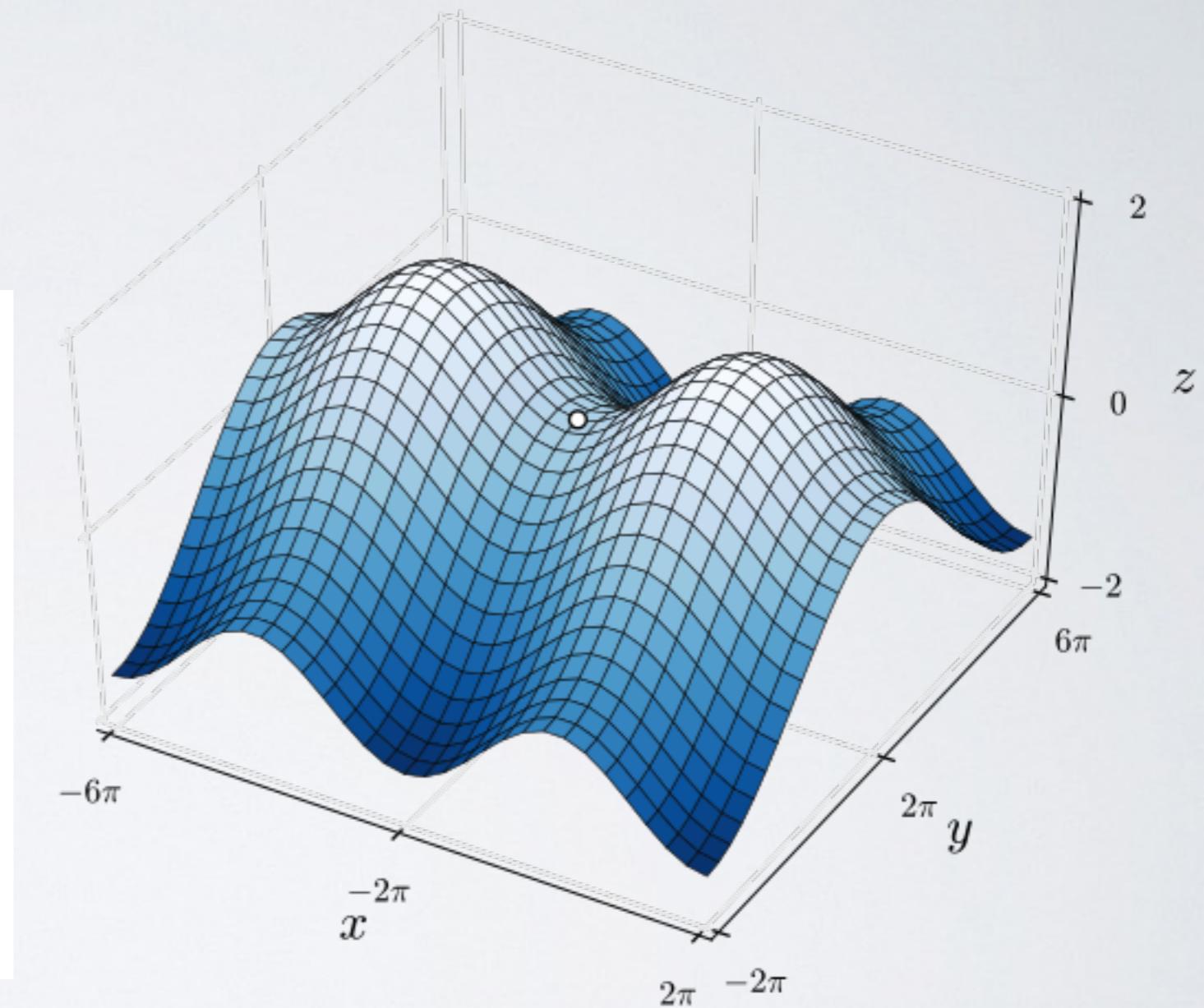
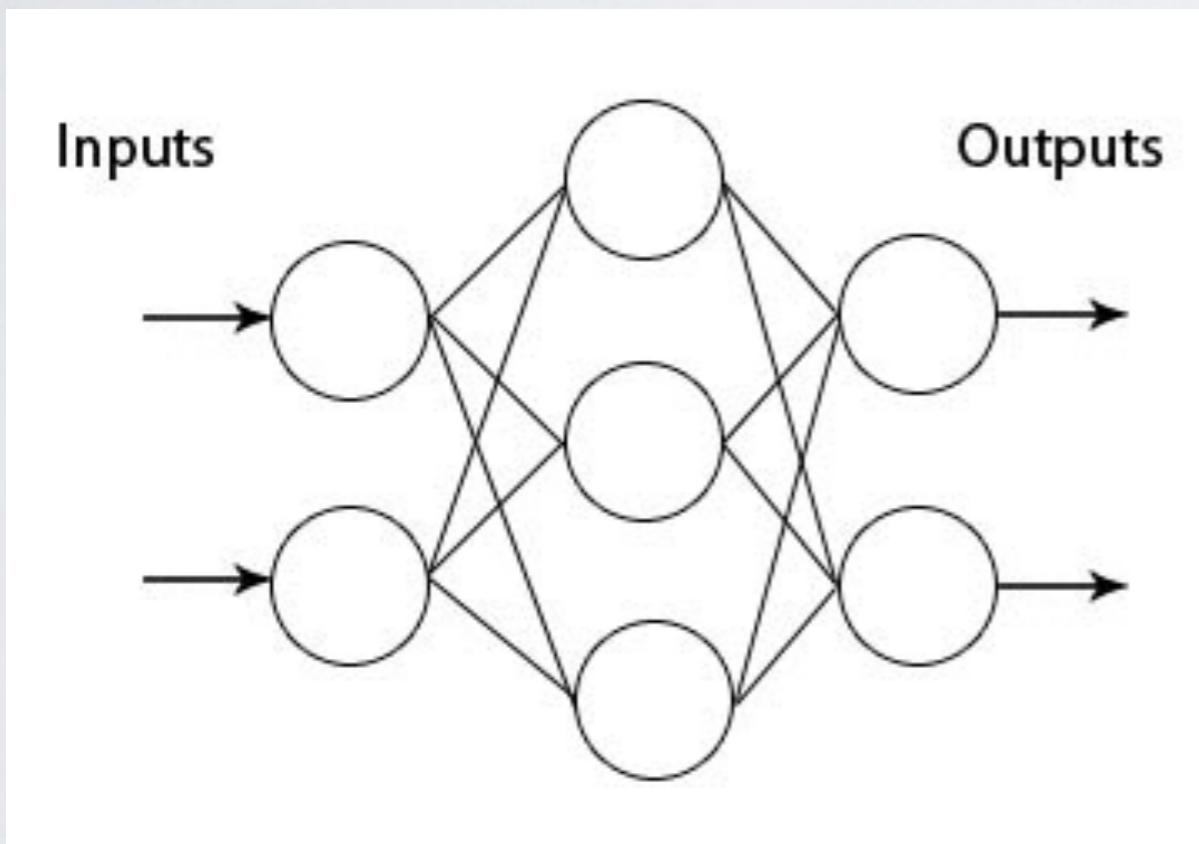
sganguli@standford.edu

Yoshua Bengio

Université de Montréal, CIFAR Fellow

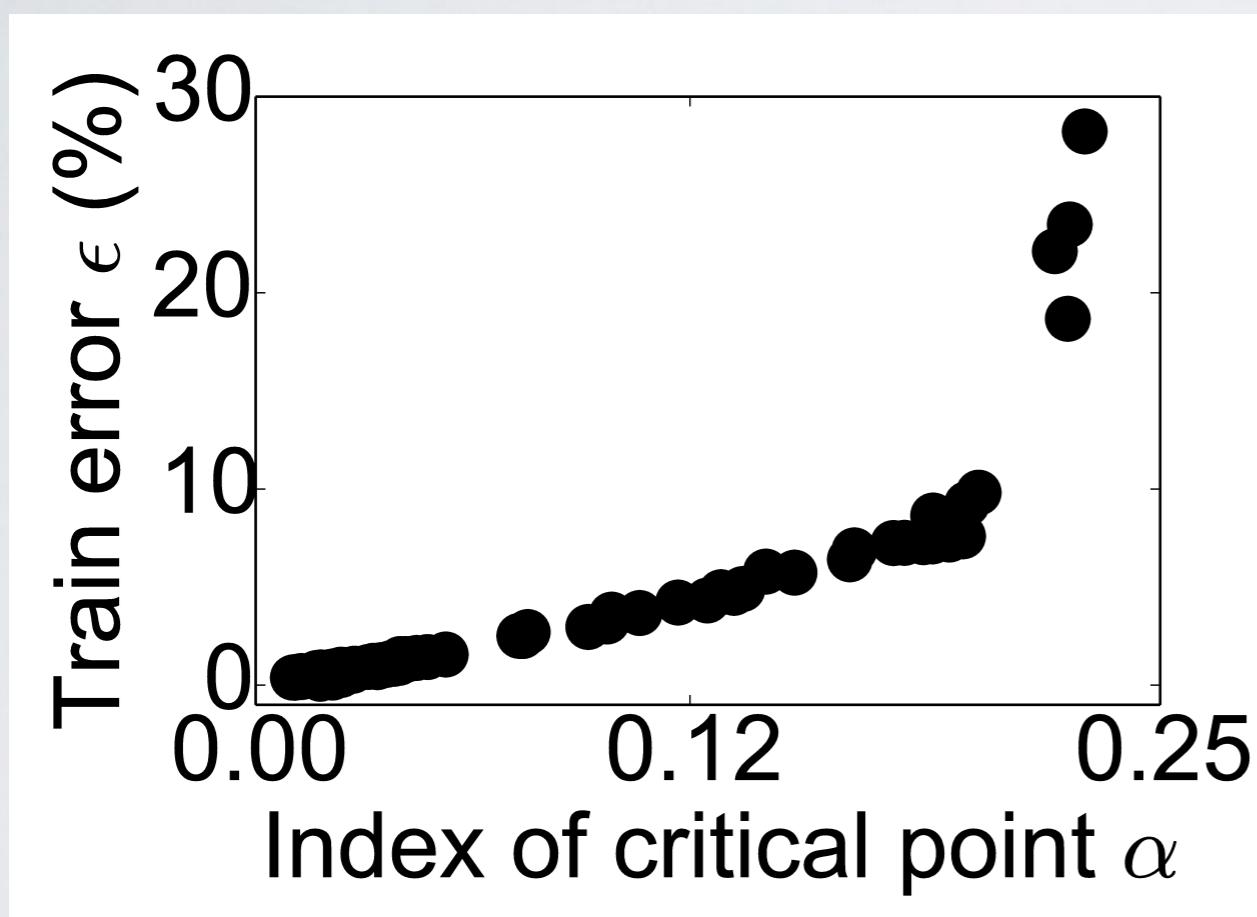
yoshua.bengio@umontreal.ca

SADDLE POINTS

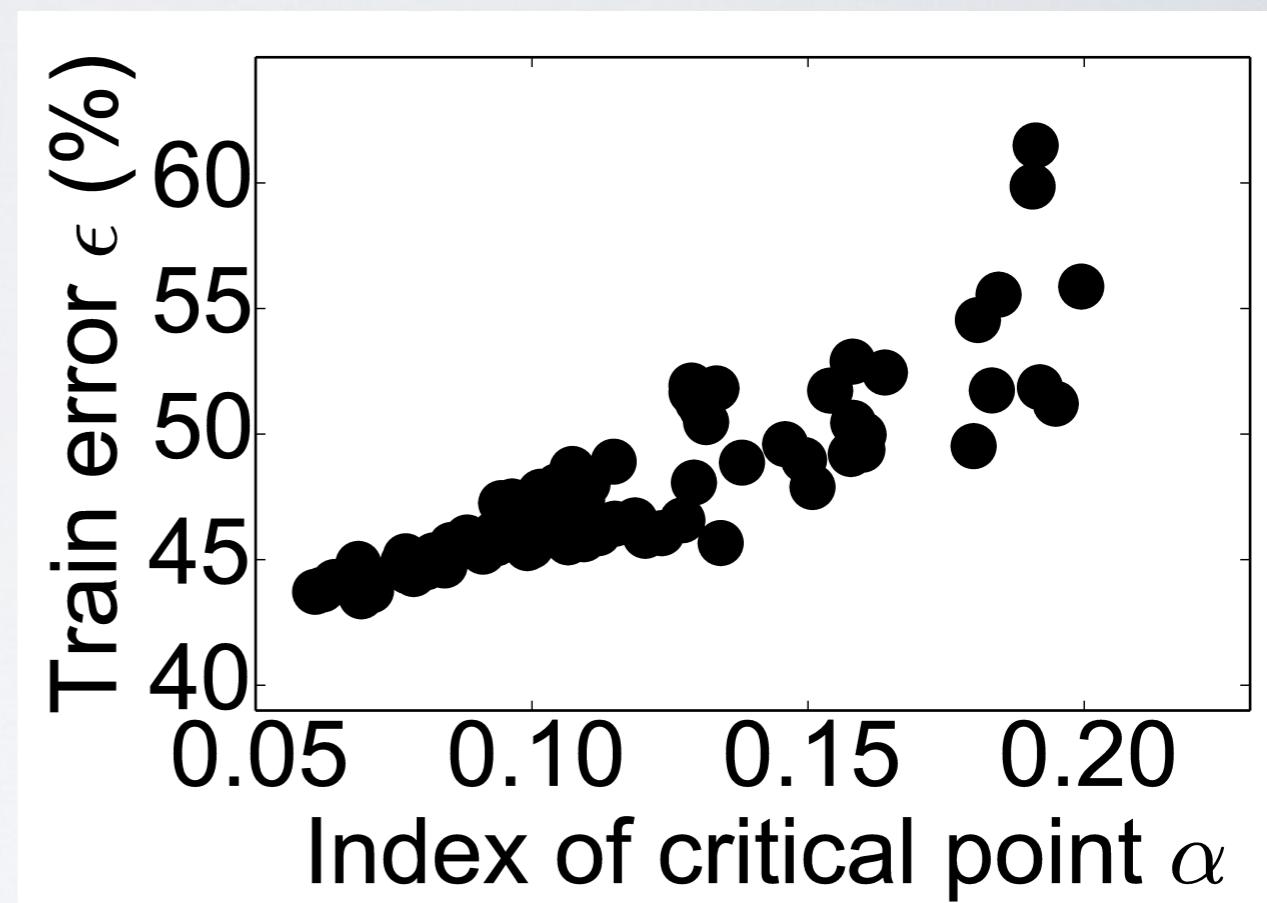


SADDLE POINTS

mnist

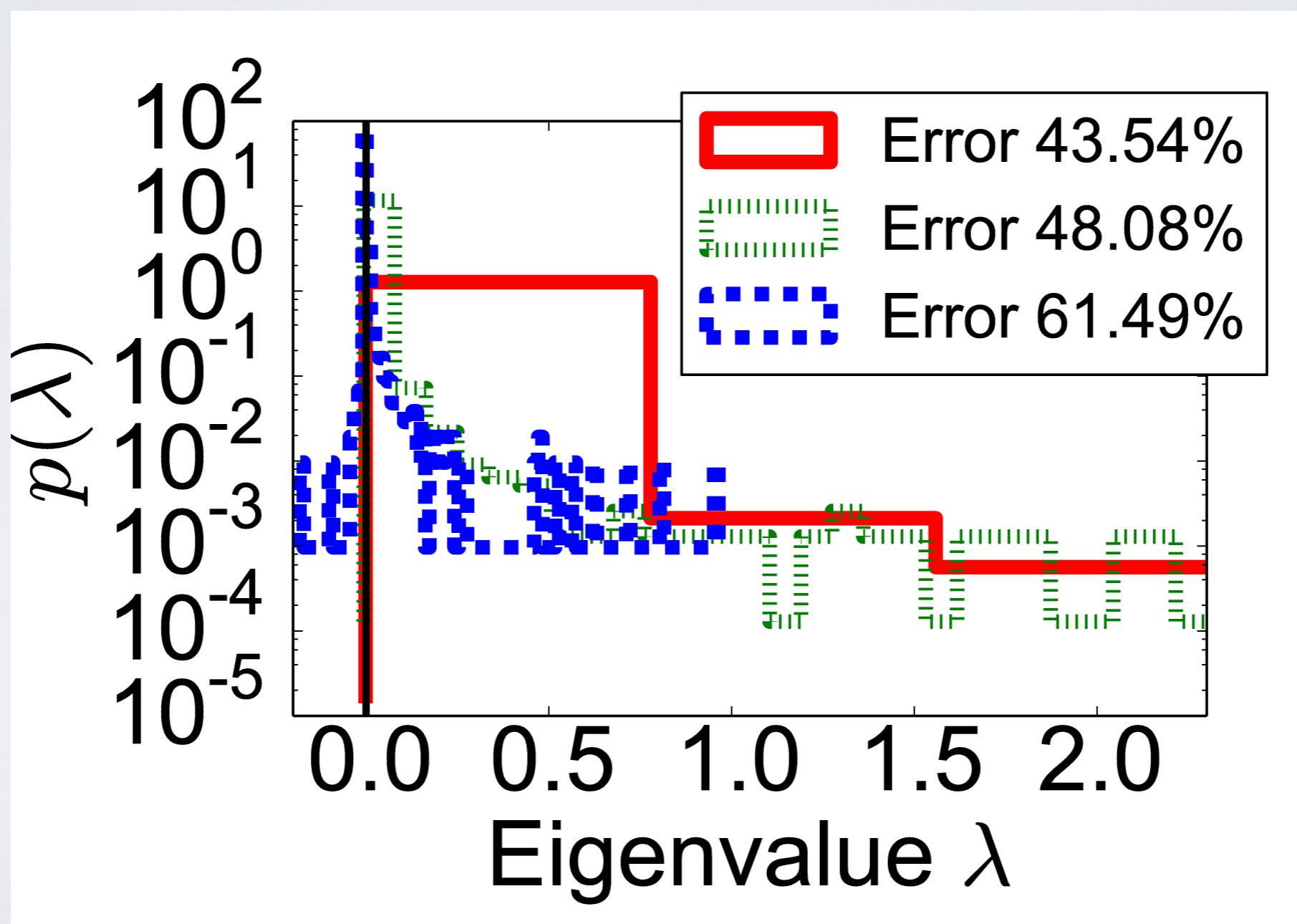


cifar

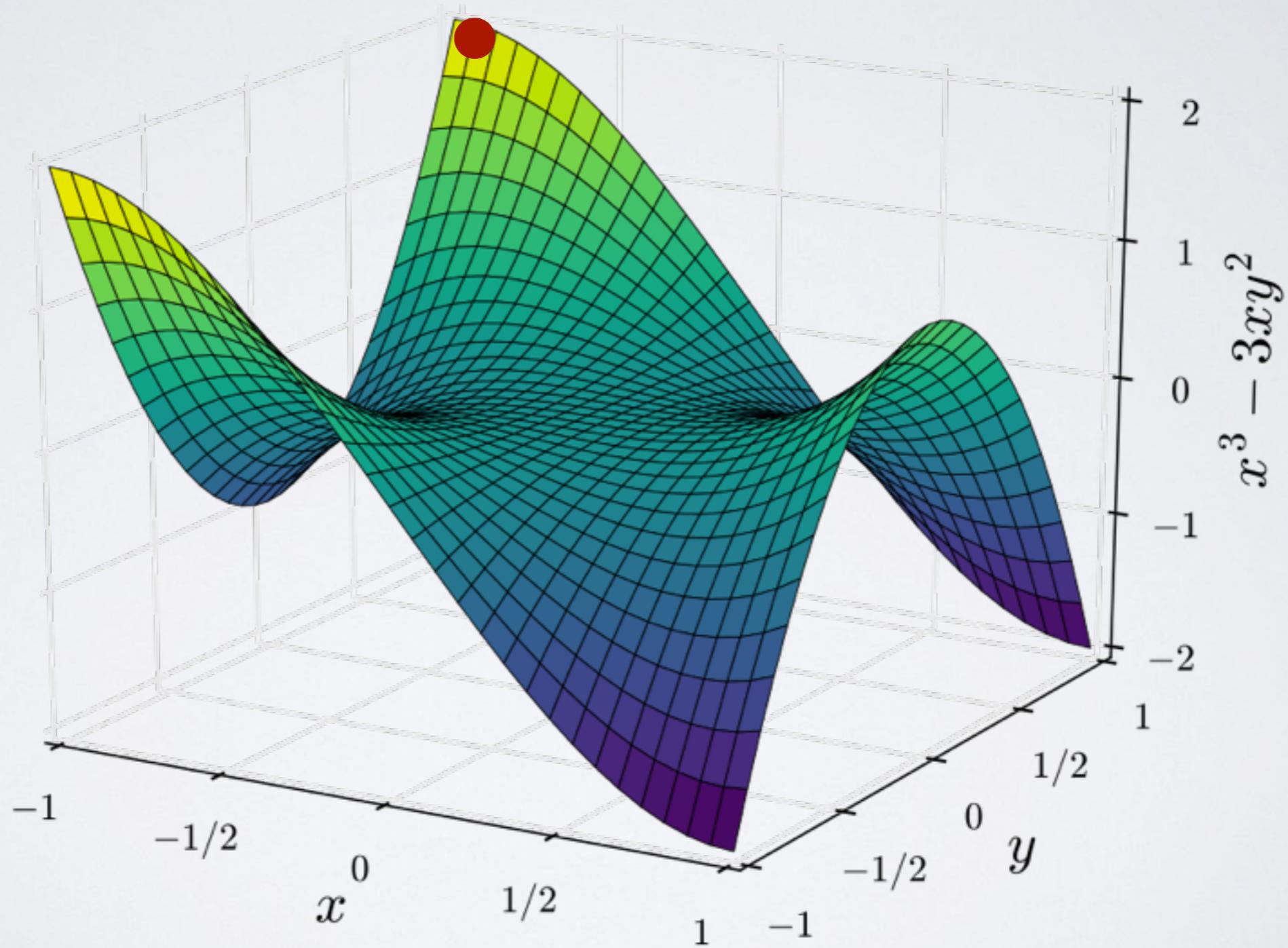


SADDLE POINTS

cifar



SADDLE POINTS



SADDLE POINTS

Escaping From Saddle Points –
Online Stochastic Gradient for Tensor Decomposition

Rong Ge*

Furong Huang †

Chi Jin ‡

Yang Yuan §

