Design Philosophy of Optimization for Deep Learning

Ian Goodfellow
Senior Research Scientist

Stanford, California
2016-03-07

High-Level Lessons

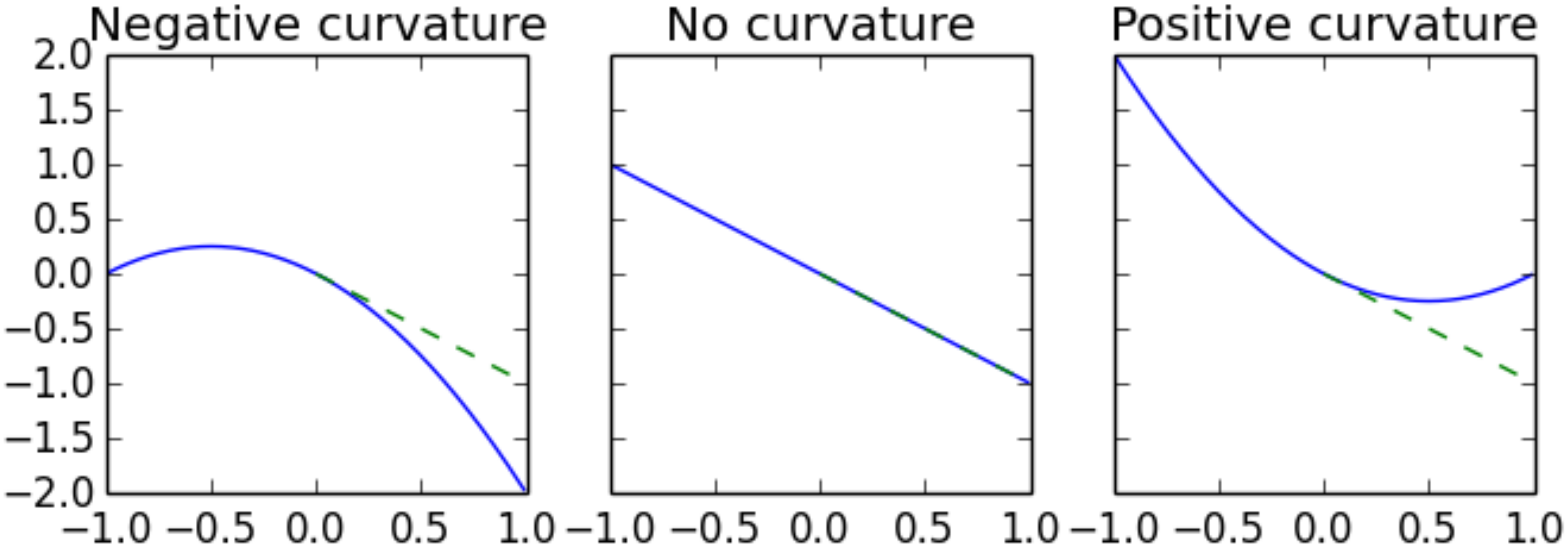
- Strive for *success*, not *perfection*
- Simple optimization methods are successful
- A little model redesign goes farther than a lot of optimization algorithm redesign

Terminology

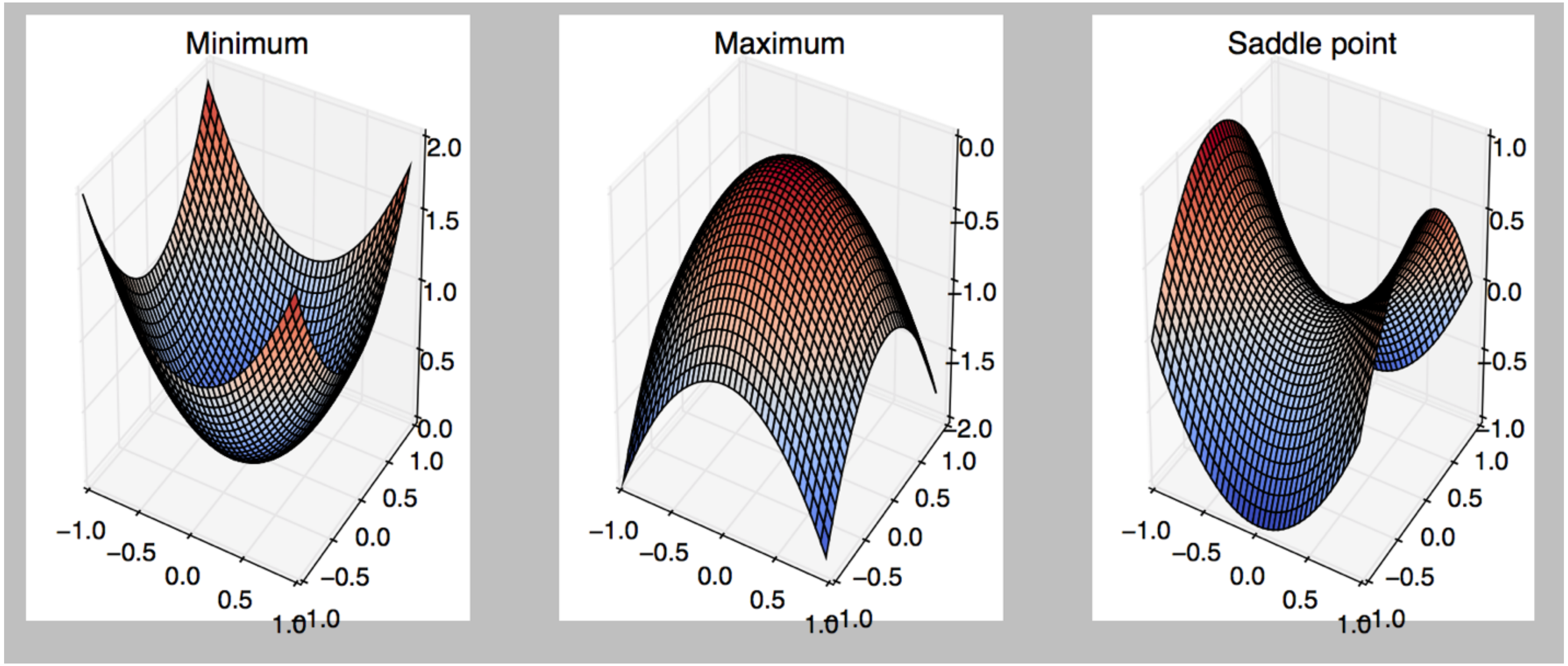
- Cost function
- Gradient
- Hessian
- Curvature
- Critical points: minima, maxima, saddle points

Derivatives and Second Derivatives

Cost function	Gradient	Hessian
$J(\boldsymbol{\theta})$	$\boldsymbol{g} = \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$	$\boldsymbol{H}$
	$g_i = \frac{\partial}{\partial \theta_i} J(\boldsymbol{\theta})$	$H_{i,j} = \frac{\partial}{\partial \theta_j} g_i$



Critical Points



All positive eigenvalues

All negative eigenvalues

Some positive
and some negative

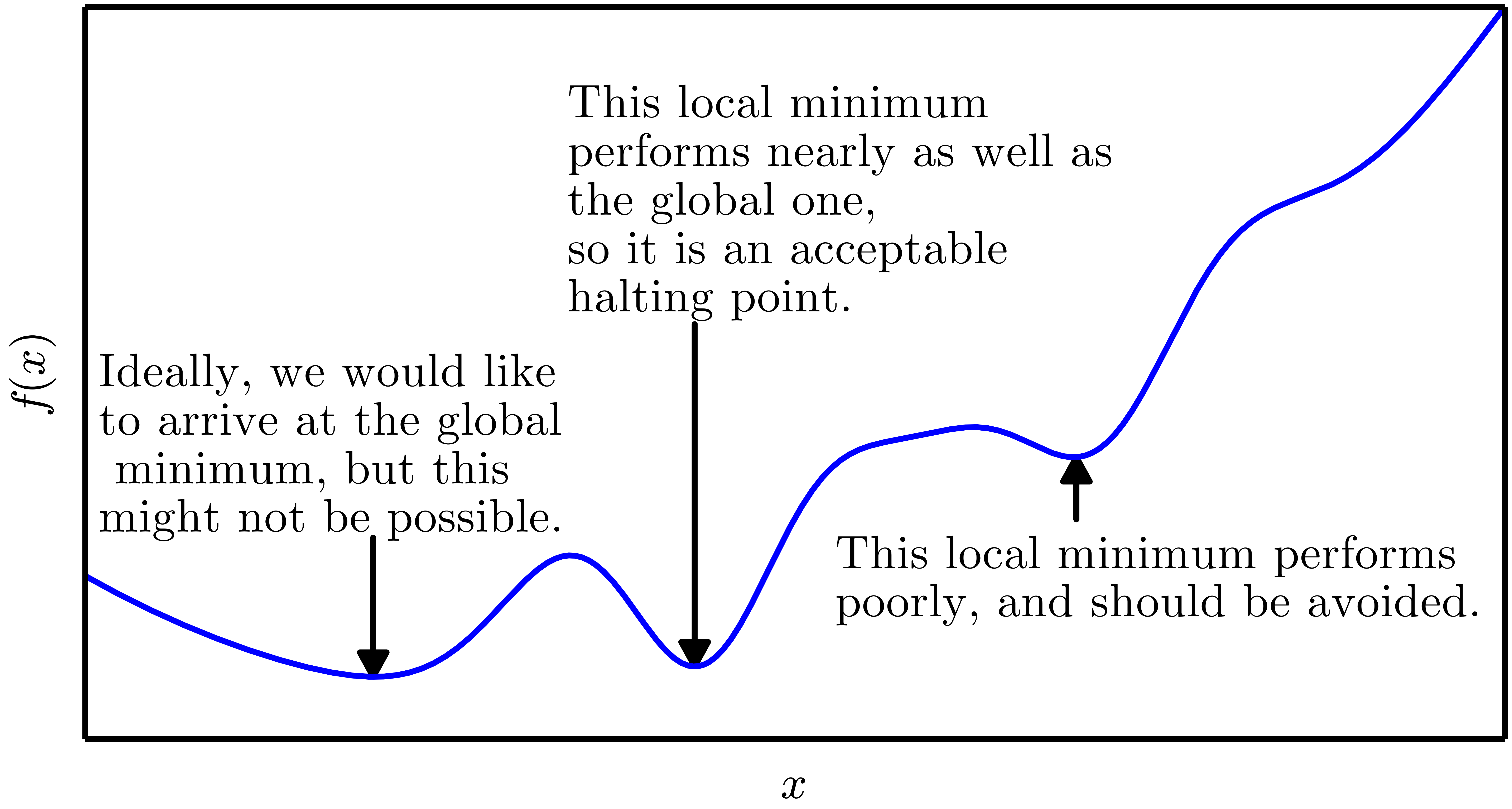
High-Level Lessons

- Strive for *success*, not *perfection*

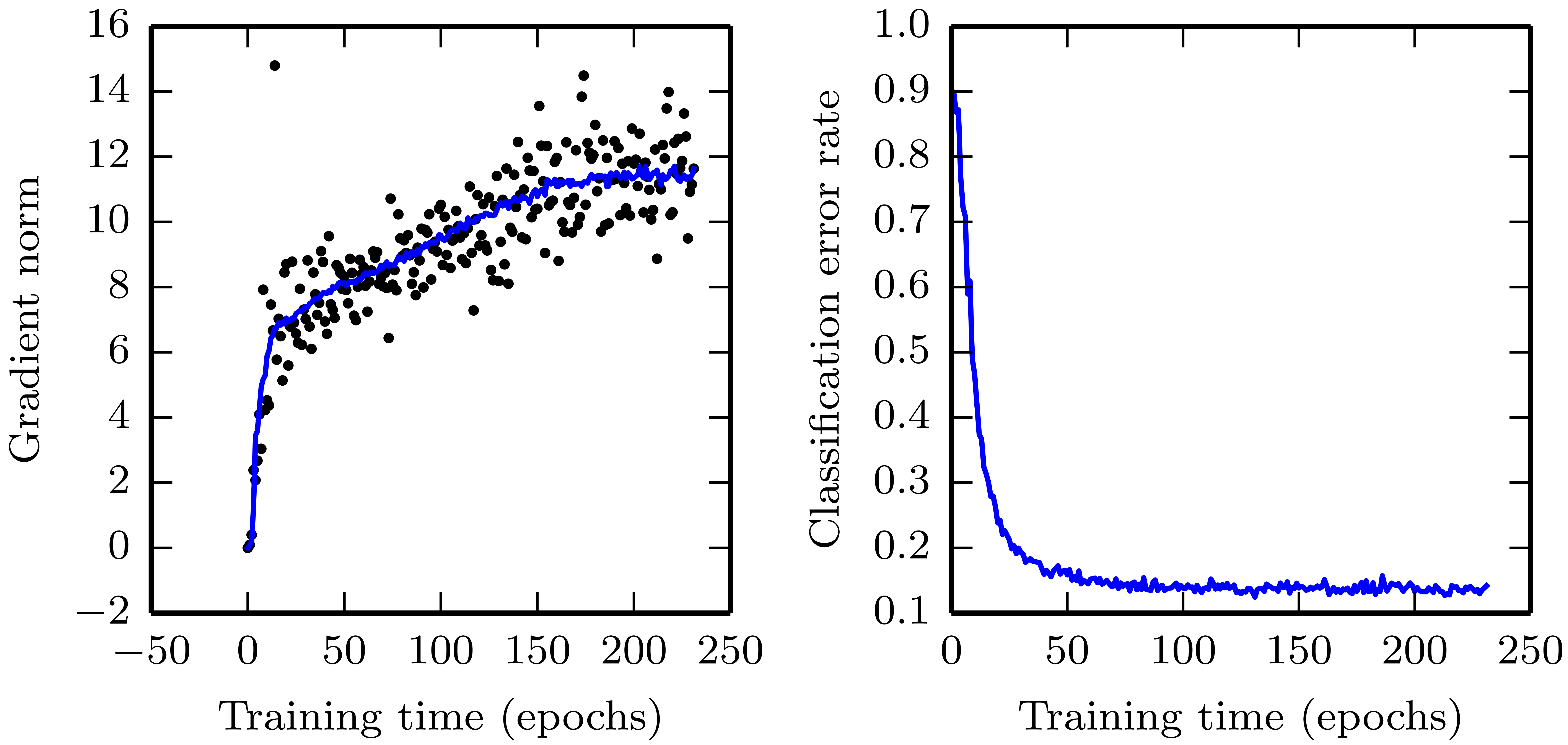
- Simple optimization methods are successful

- A little model redesign goes farther than a lot of optimization algorithm redesign

Approximate minimization



No Critical Point



High-Level Lessons

- Strive for *success*, not *perfection*
- Simple optimization methods are successful
- A little model redesign goes farther than a lot of optimization algorithm redesign

The old myth of SGD failure

- SGD usually moves downhill
- SGD eventually encounters a critical point
- Usually this is a minimum
- However, it is a *local minimum*
- The cost function is high at this point
- Some *global minimum* is the real target, and has much lower cost

The new myth of SGD failure

- SGD usually moves downhill
- SGD eventually encounters a critical point
- Usually this is a saddle point
- SGD is stuck, and the main reason it is stuck is that it fails to exploit negative curvature

Gradient descent flees saddle points

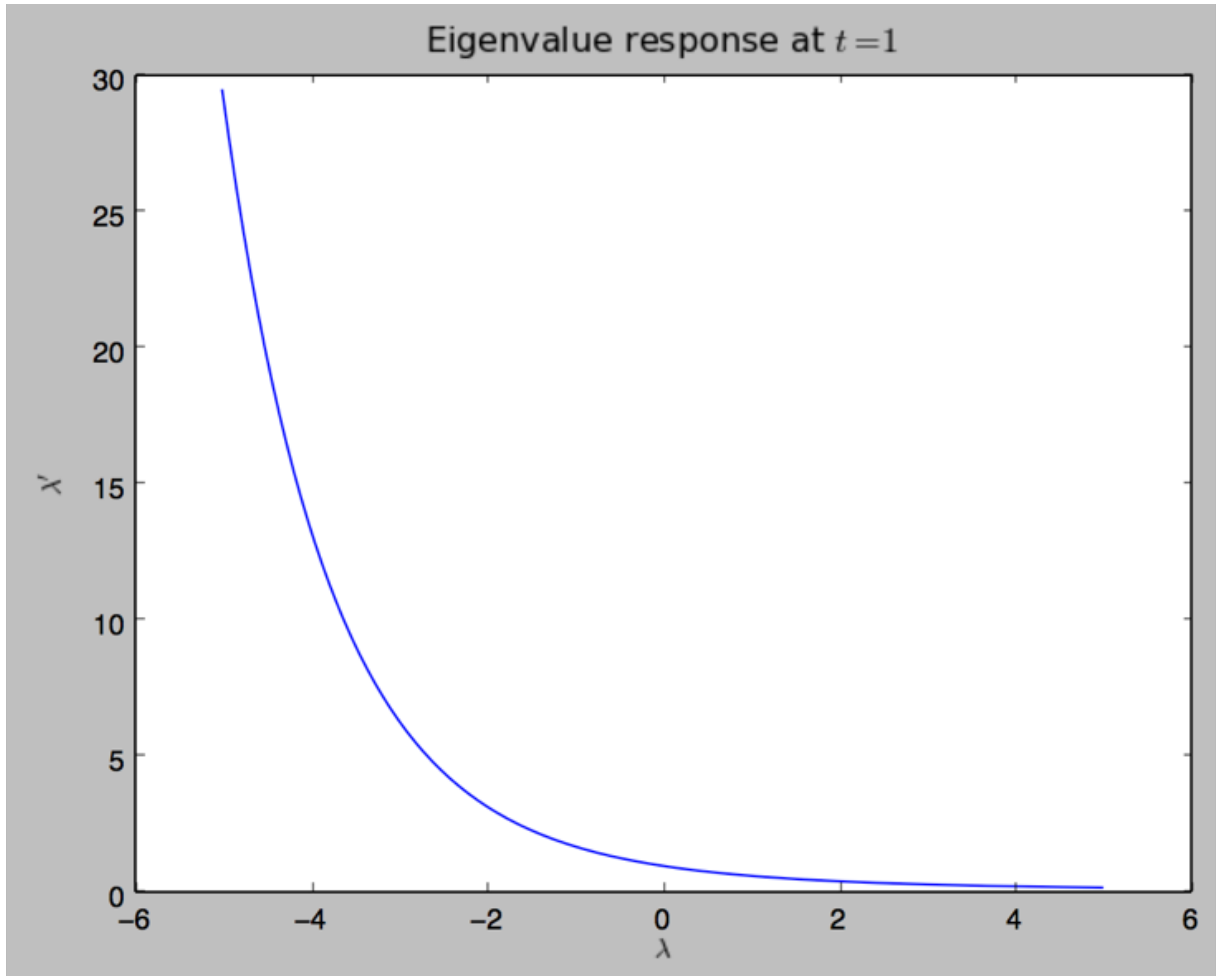
$$\frac{d}{dt}\boldsymbol{\theta}(t) = -\boldsymbol{g} - \boldsymbol{H}\left(\boldsymbol{\theta}(t) - \boldsymbol{\theta}(0)\right)$$

$$\Rightarrow$$

$$\boldsymbol{\theta}(t) = \boldsymbol{\theta}(0) - \boldsymbol{Q}\Lambda'(t)\boldsymbol{Q}^T\boldsymbol{g}$$

where

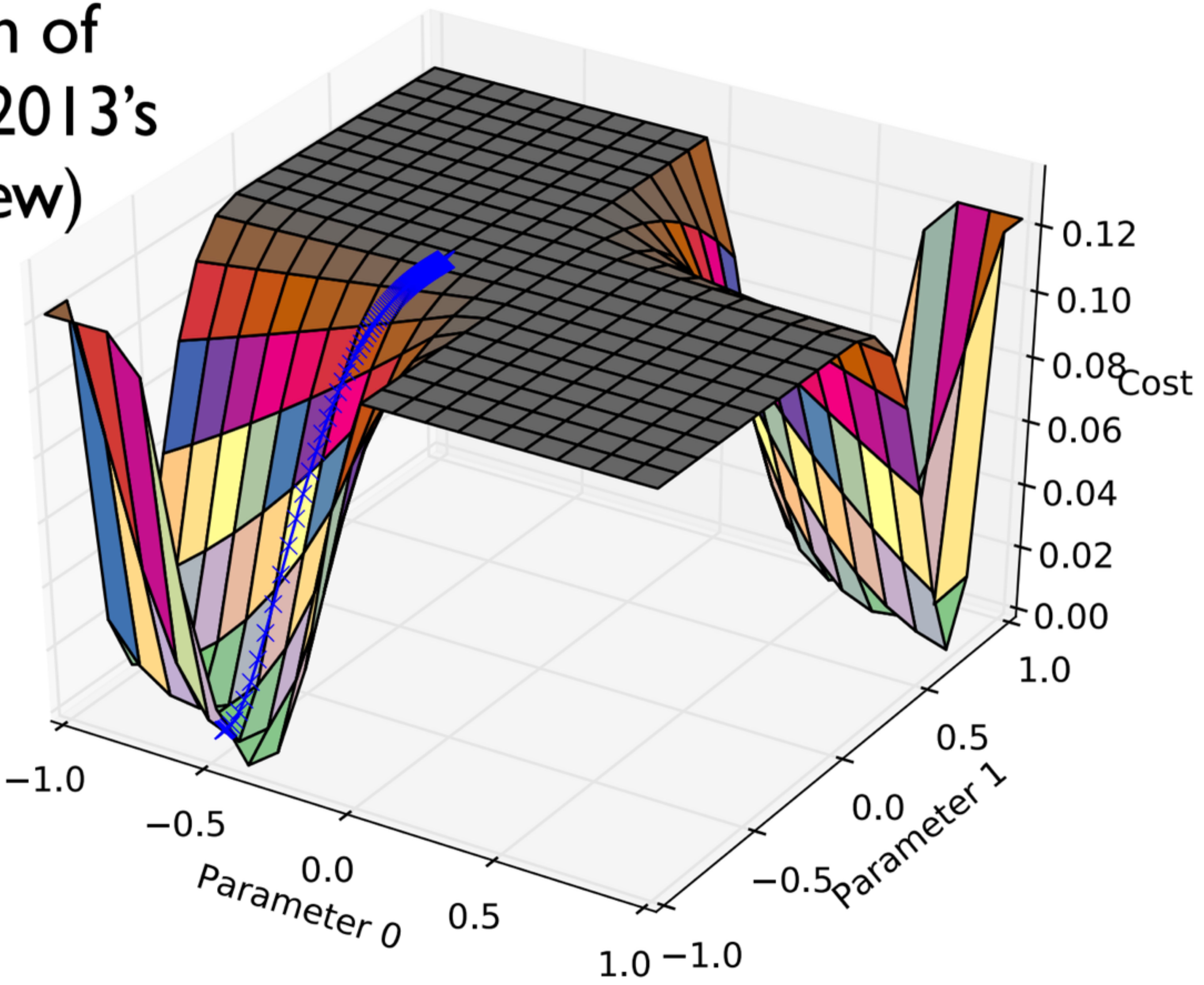
$$\lambda'(t) = \frac{1 - \exp(-\lambda t)}{\lambda}.$$

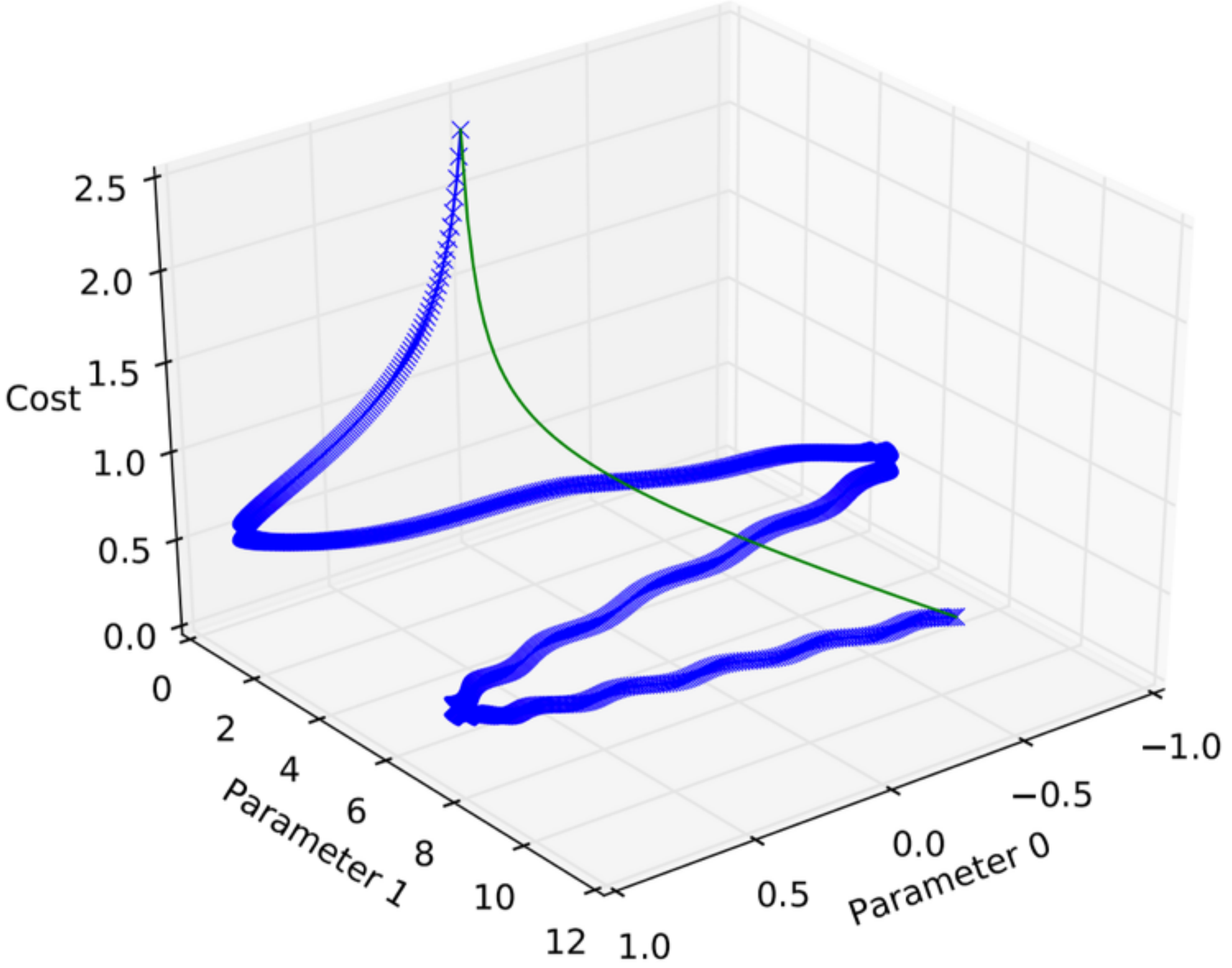


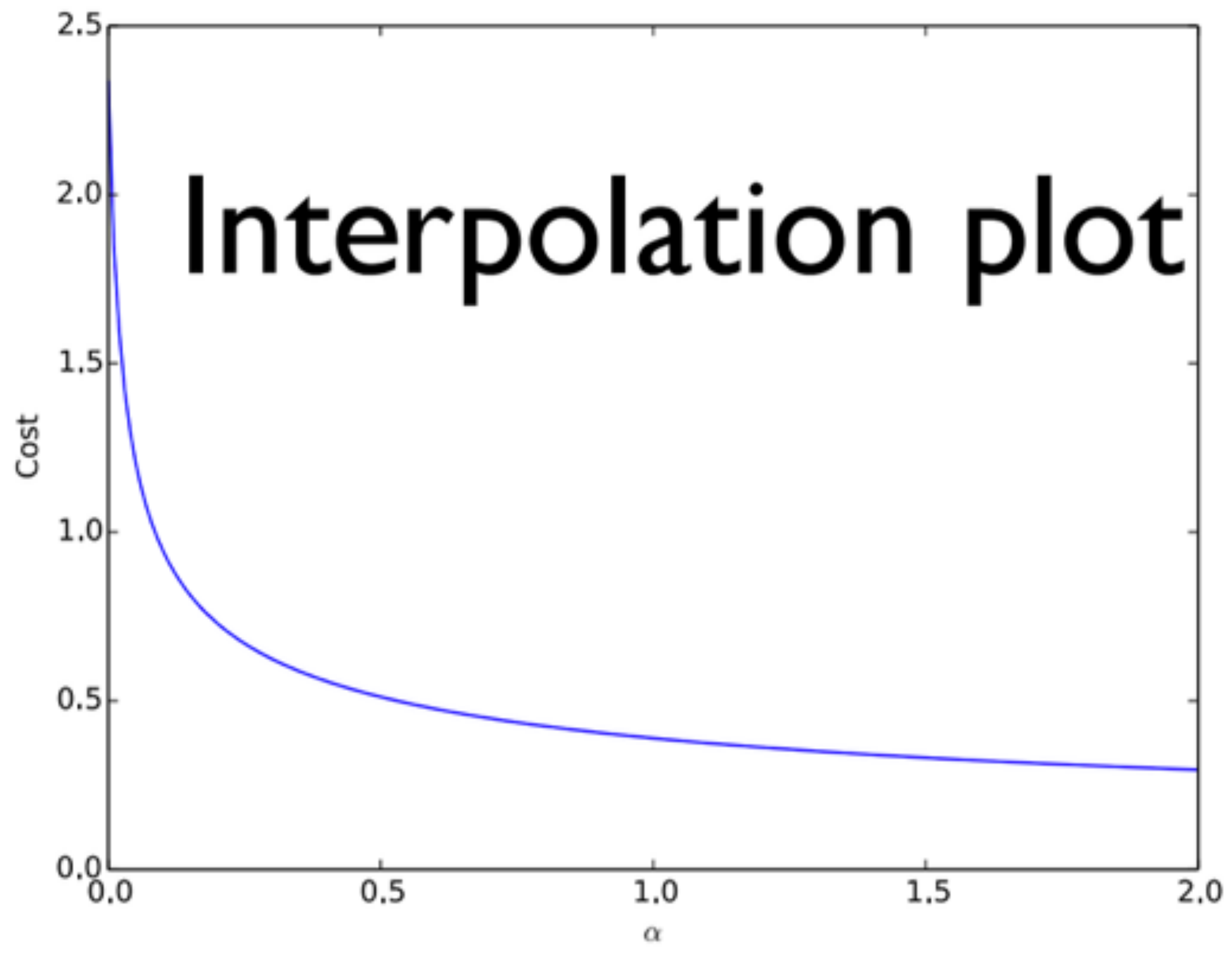
Saddle points are a problem.... **for Newton's method, not SGD.**

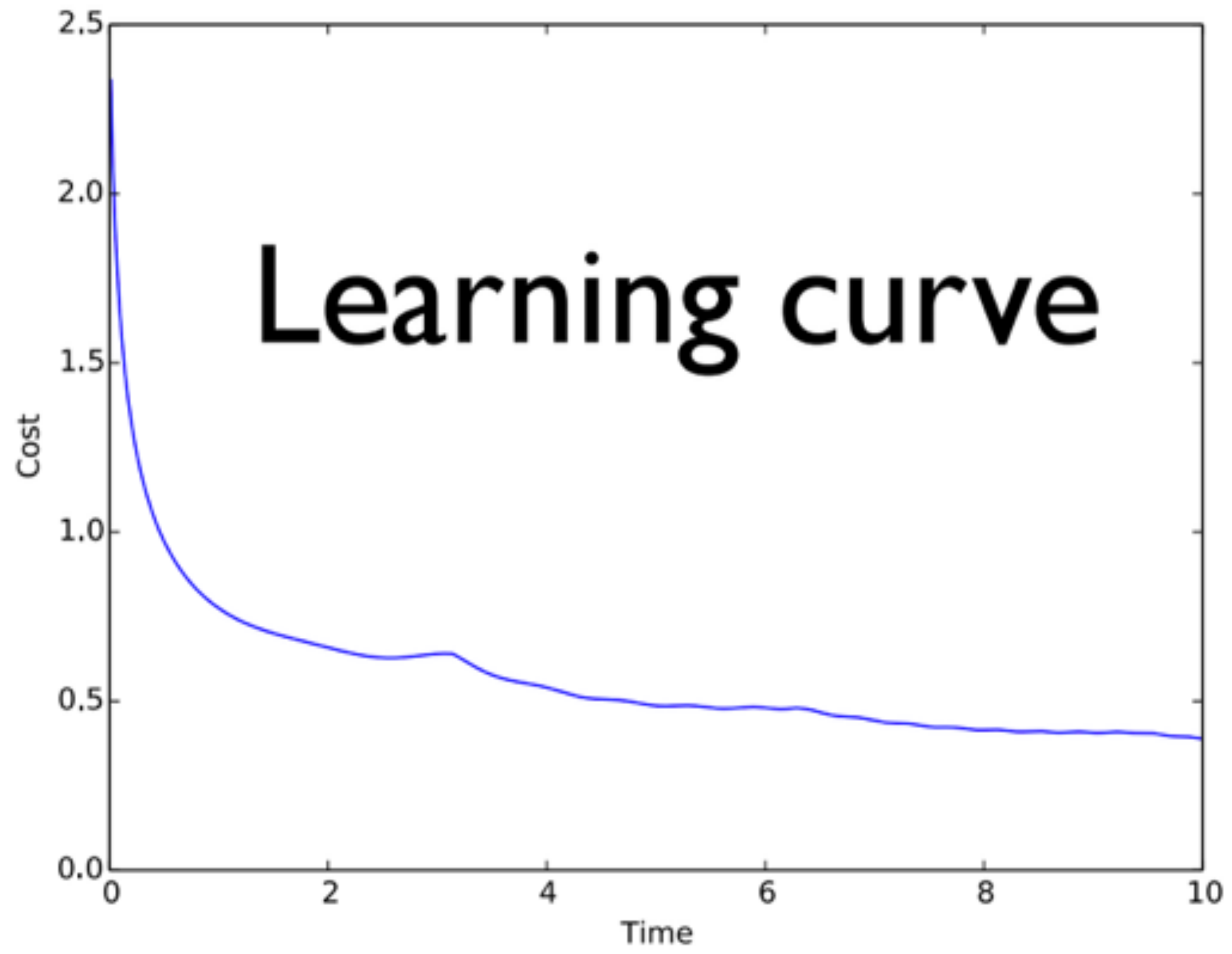
"Qualitatively Characterizing Neural Network Optimization Problems,"
Goodfellow, Vinyals and Saxe, ICLR 2015

(Cartoon of Saxe et al 2013's worldview)



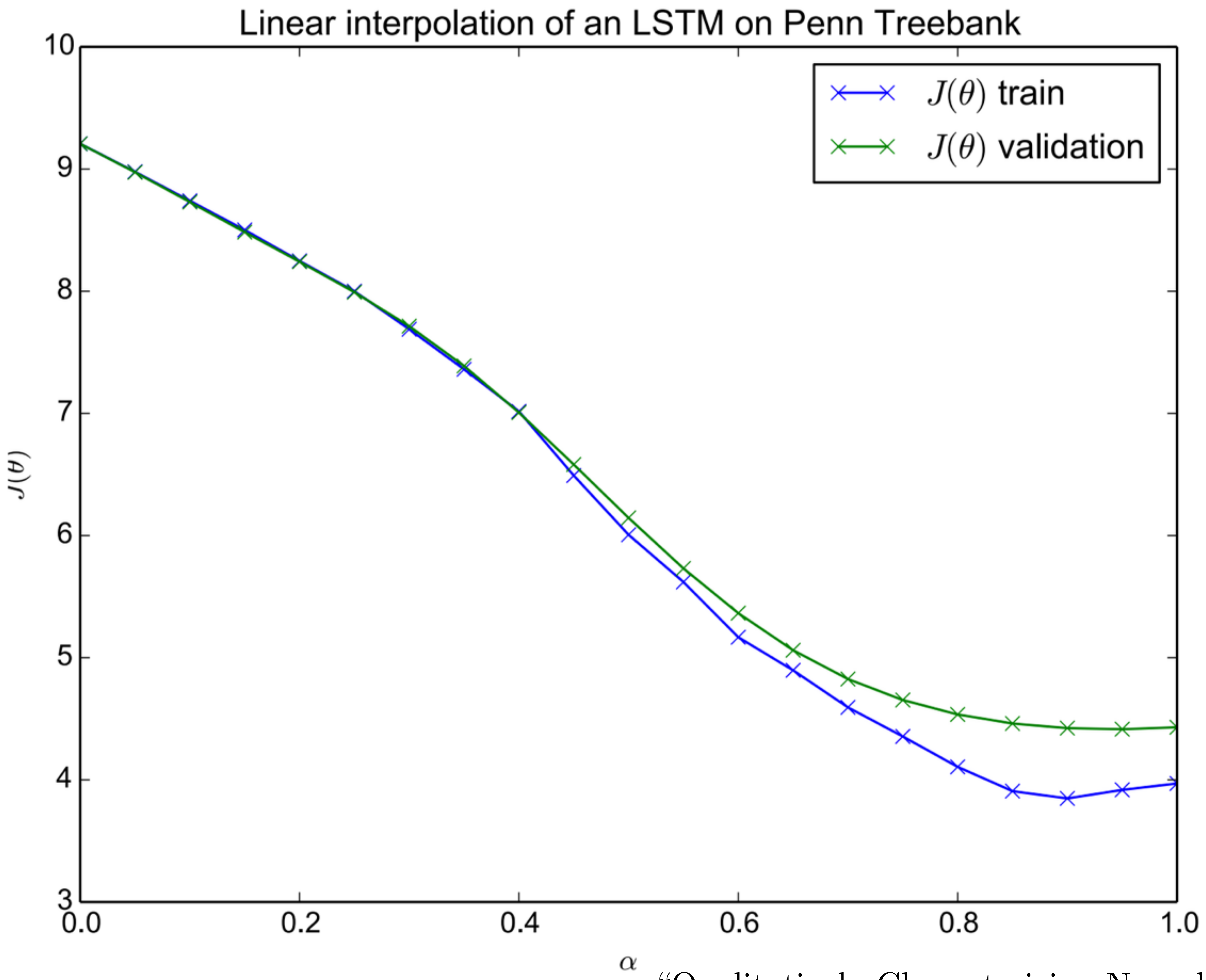




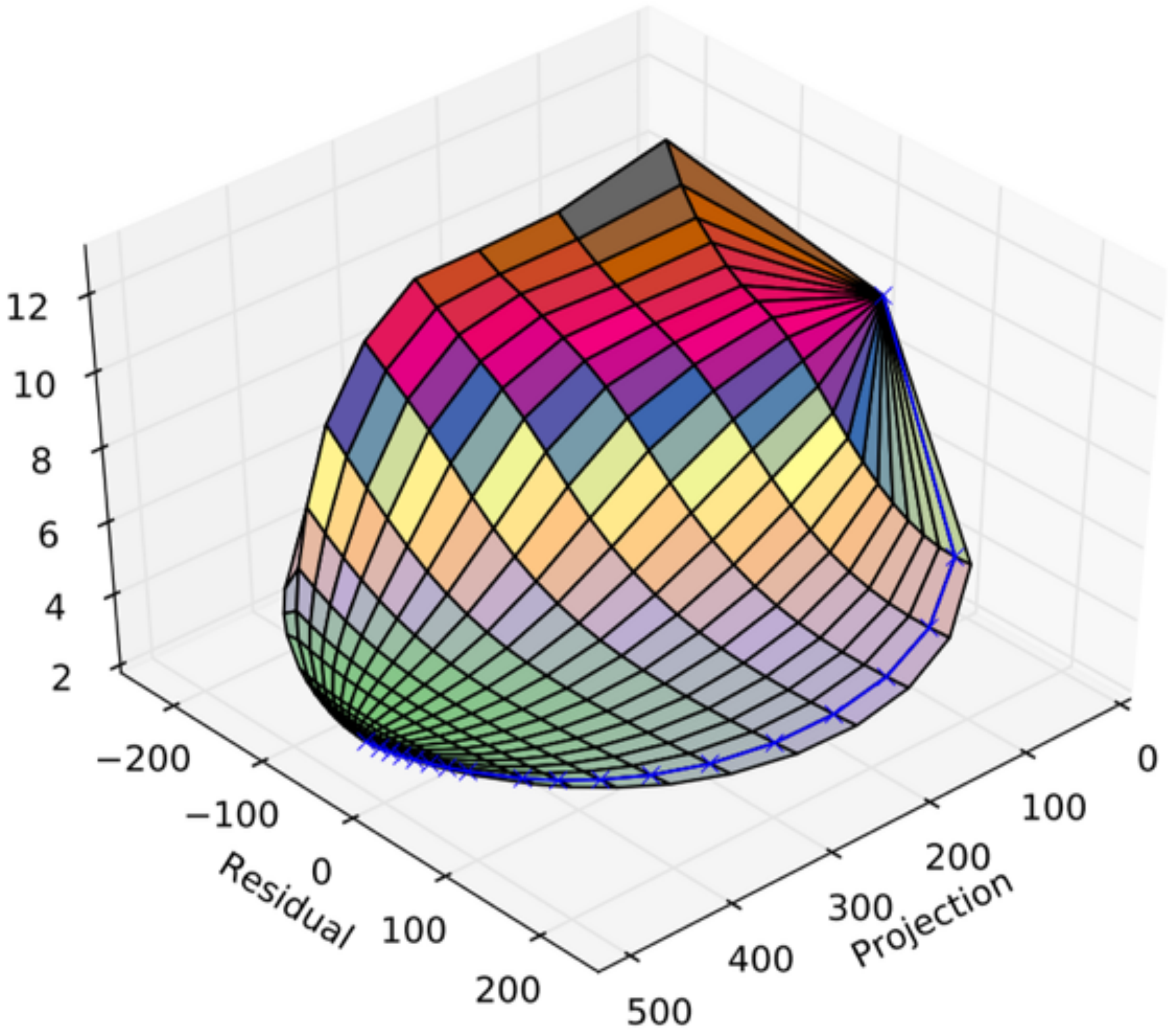


"Qualitatively Characterizing Neural Network Optimization Problems,"

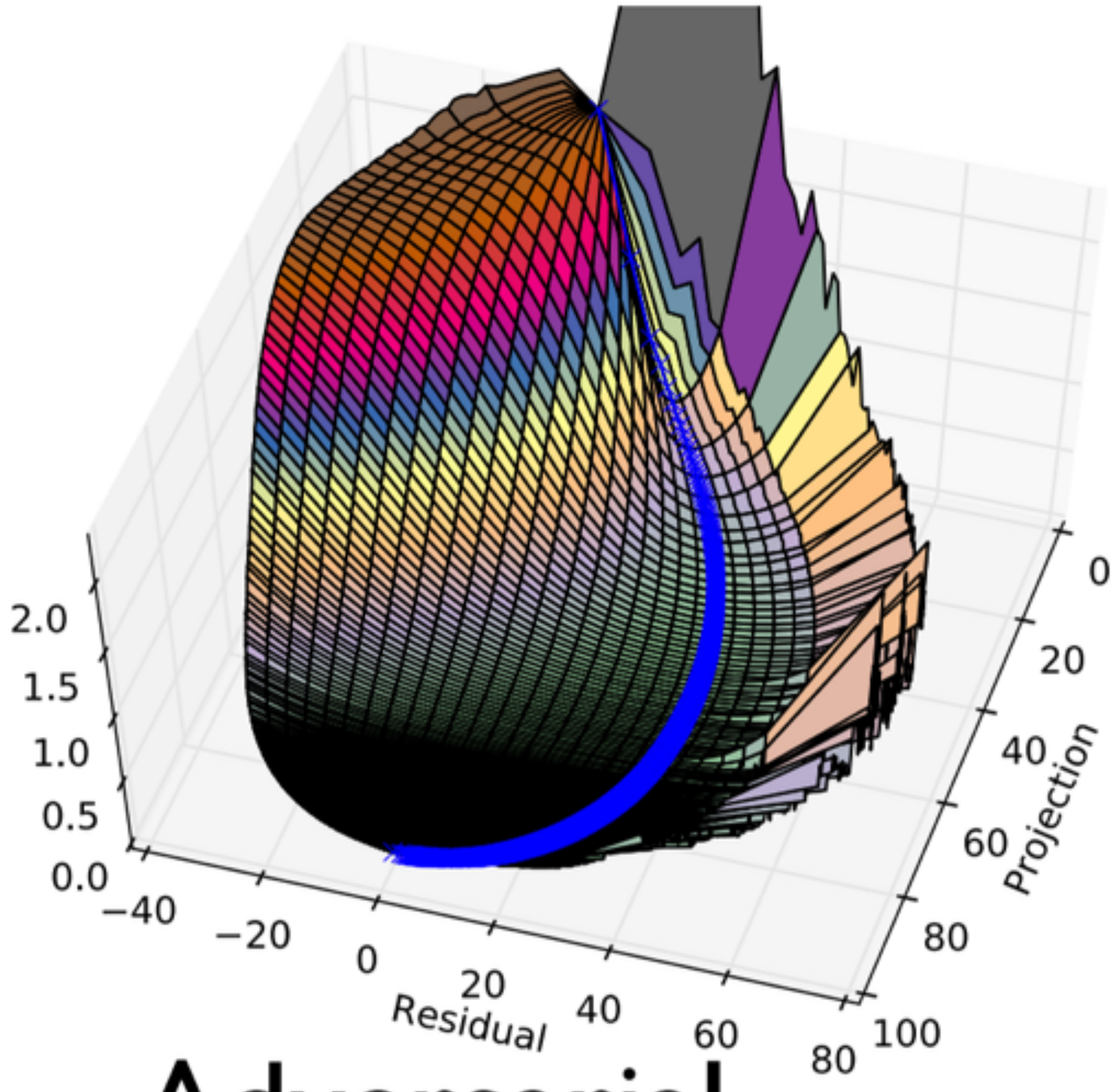
Goodfellow, Vinyals and Saxe, ICLR 2015



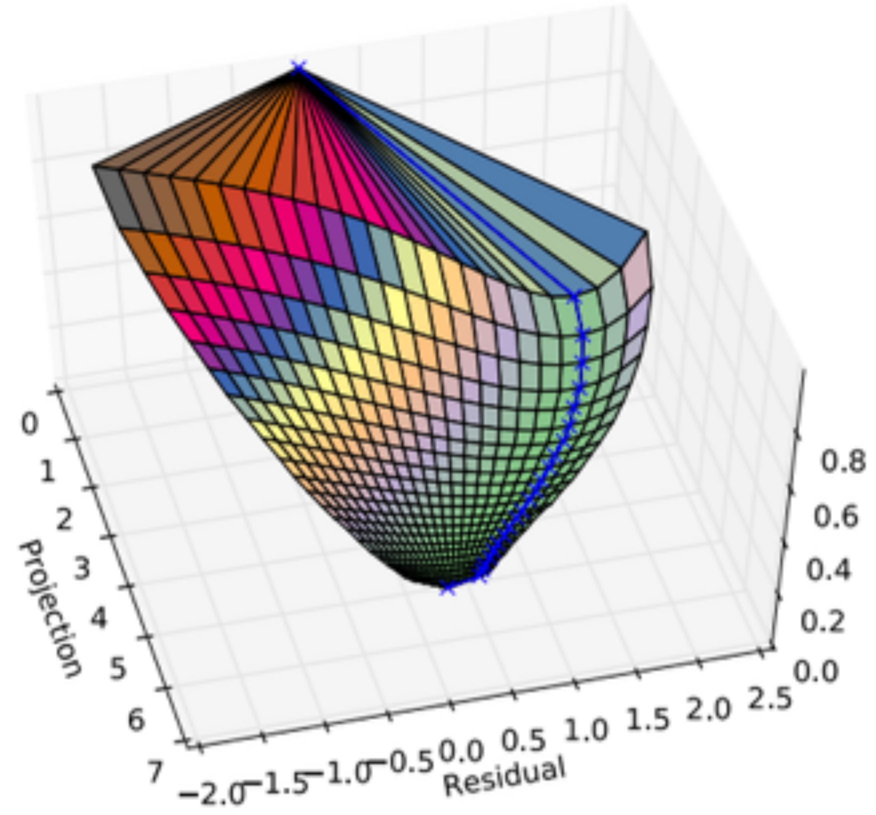
"Qualitatively Characterizing Neural Network Optimization Problems,"
Goodfellow, Vinyals and Saxe, ICLR 2015



LSTM



Adversarial ReLUs



Factored Linear

"Qualitatively Characterizing Neural Network Optimization Problems,"
Goodfellow, Vinyals and Saxe, ICLR 2015

High-Level Lessons

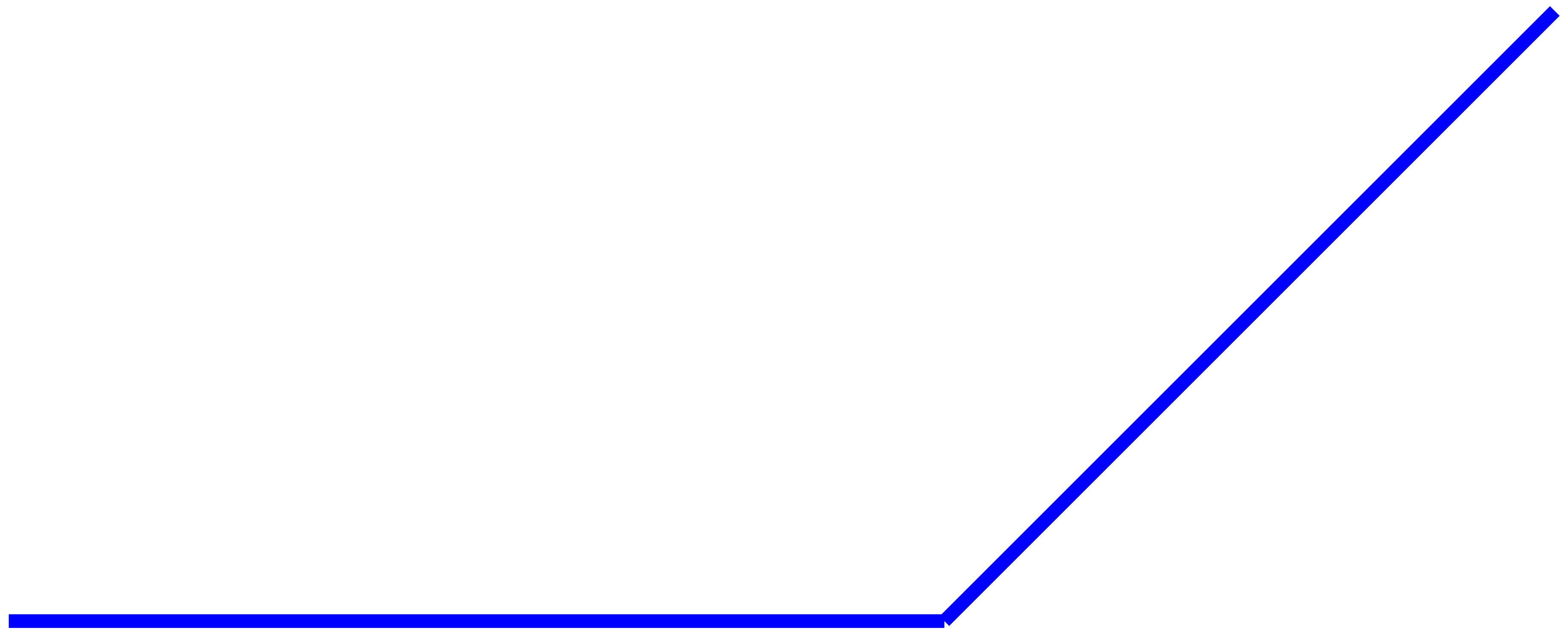
- Strive for *success*, not *perfection*
- Simple optimization methods are successful
- A little model redesign goes farther than a lot of optimization algorithm redesign

Two Extreme Positions

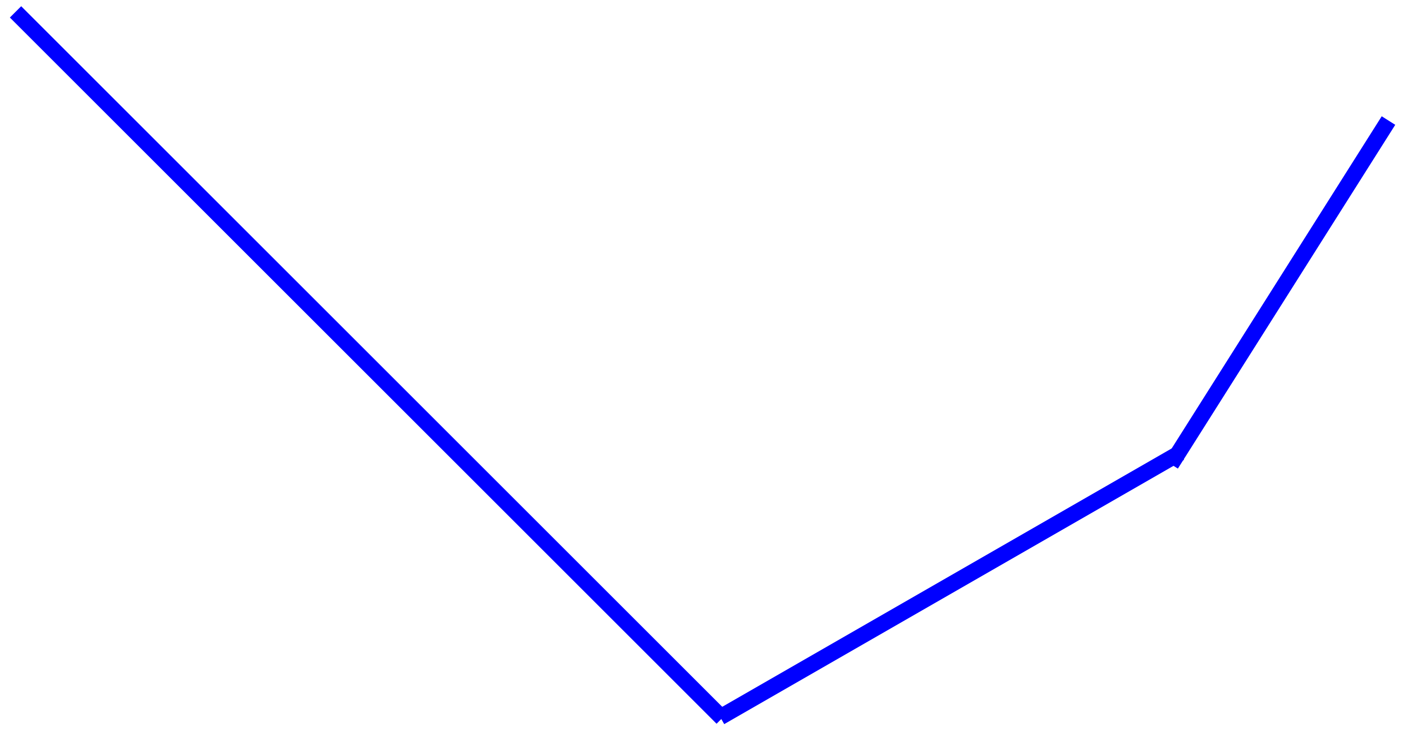
- Convex optimization: Design model within a set of formal constraints, such that efficient and perfect optimization is guaranteed

- Fully general optimization: Write down whatever model seems most intuitive, hope you can optimize it

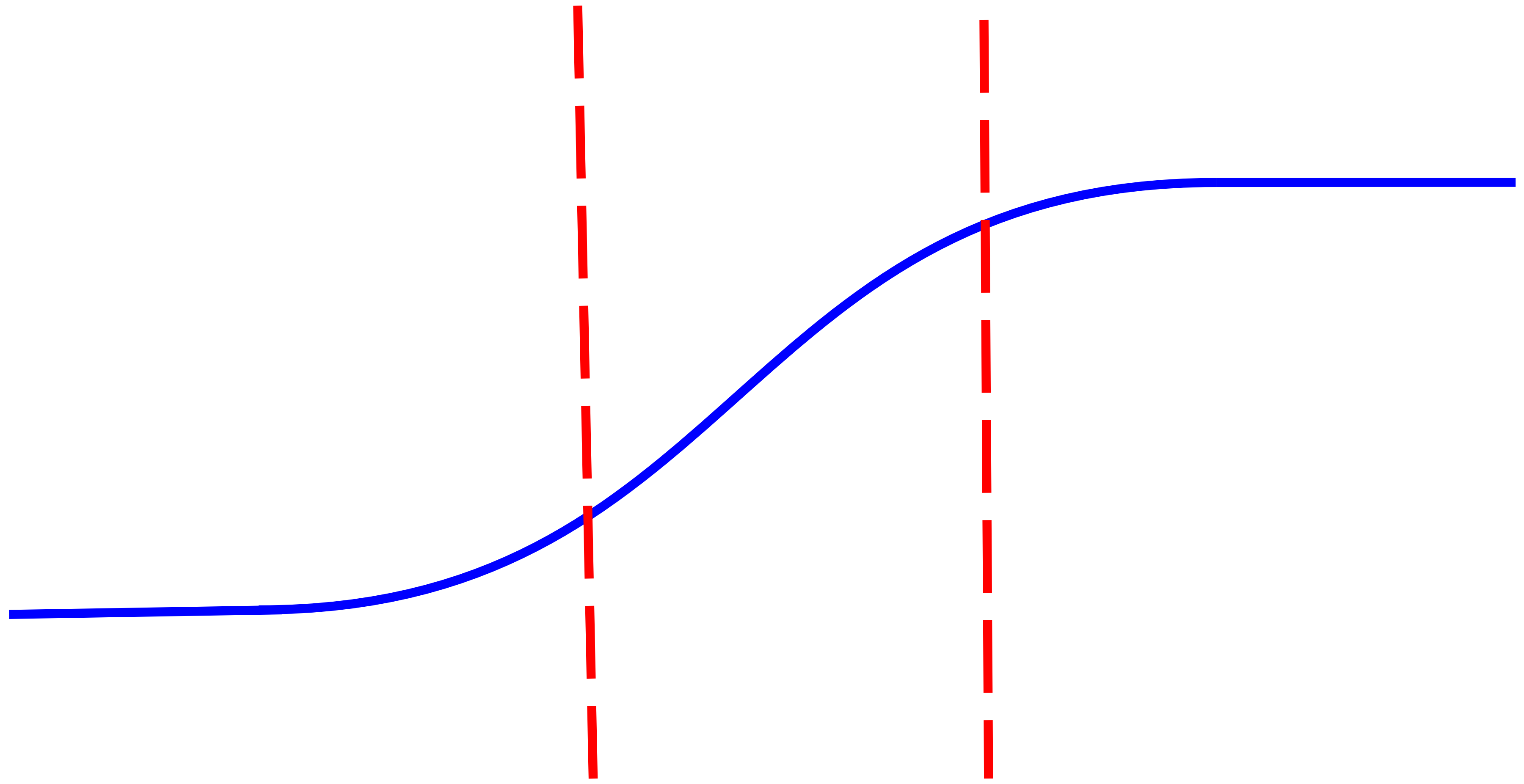
Modern Deep Nets are very (Piece-Wise) Linear



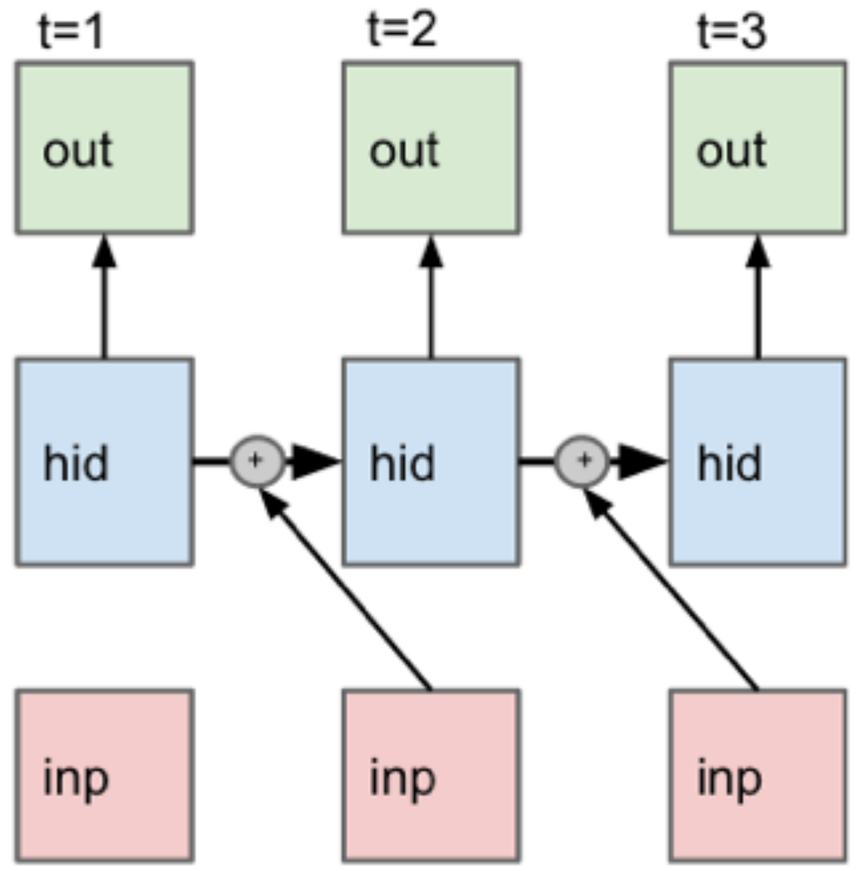
Rectified Linear Unit



Maxout



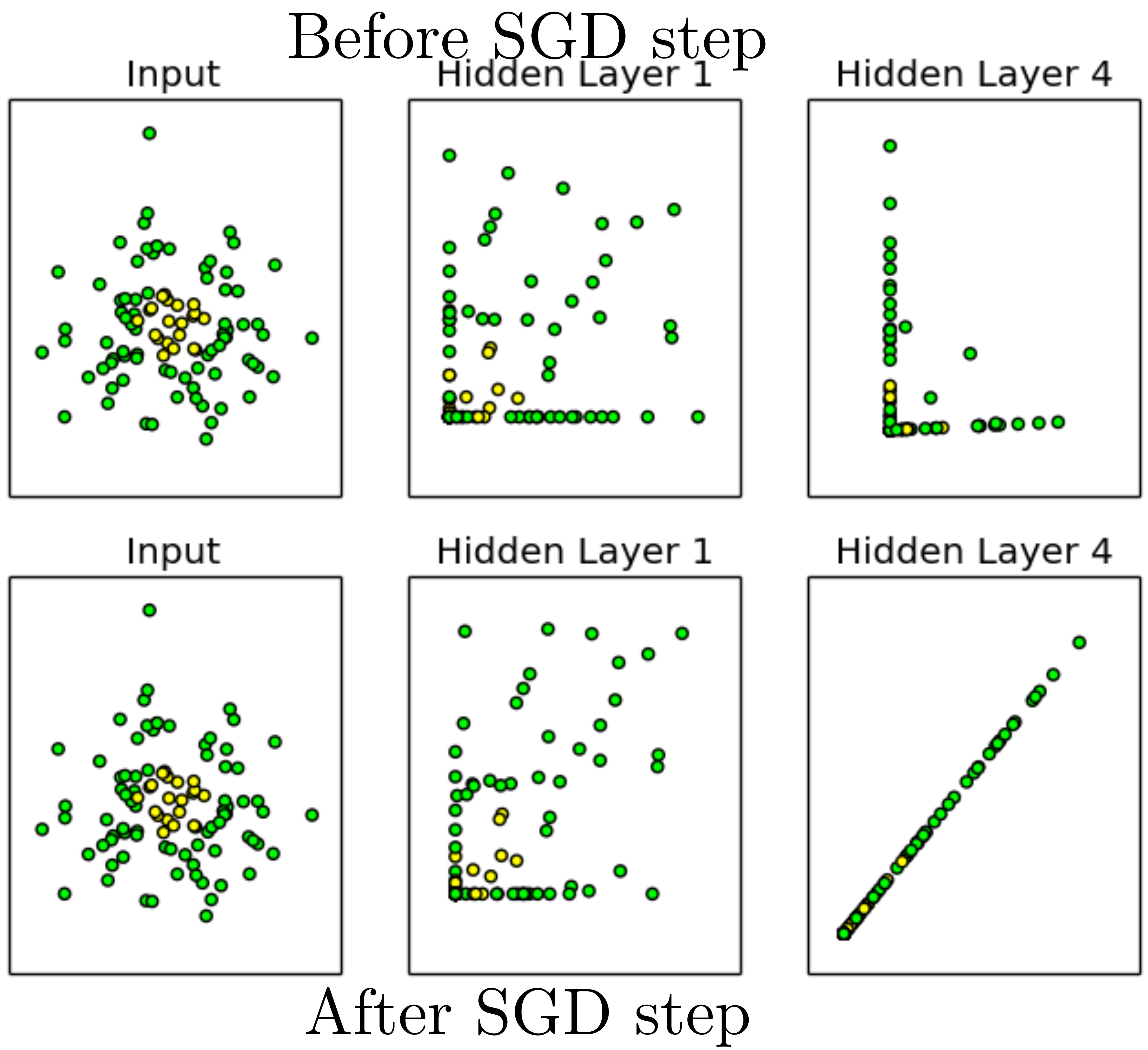
Carefully tuned sigmoid



LSTM (addition is linear)

Batch Normalization

- Consider a very deep net with
  - No nonlinearities
  - Only one unit per layer
- y = abcdef...x



"Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," Ioffe and Szegedy 2015

Batch Normalization

$$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{W}$$

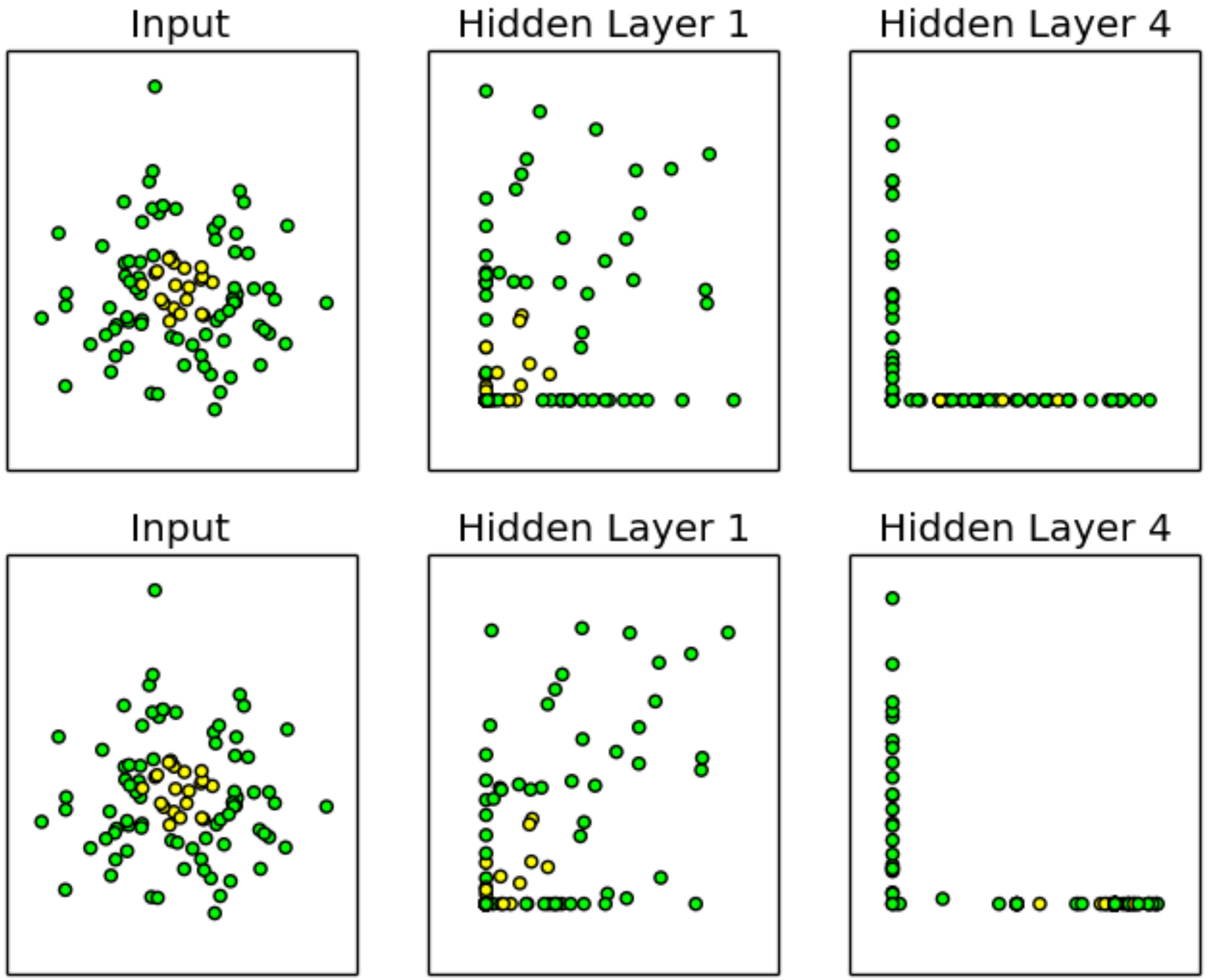
$$\tilde{\boldsymbol{Z}} = \boldsymbol{Z} - \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{Z}_{i,:}$$

$$\hat{\boldsymbol{Z}} = \frac{\tilde{\boldsymbol{Z}}}{\sqrt{\epsilon + \frac{1}{m}\sum_{i=1}^{m} \tilde{\boldsymbol{Z}}_{i,:}^2}}$$

$$\boldsymbol{H} = \max\{0, \boldsymbol{\gamma}\hat{Z} + \boldsymbol{\beta}\}$$

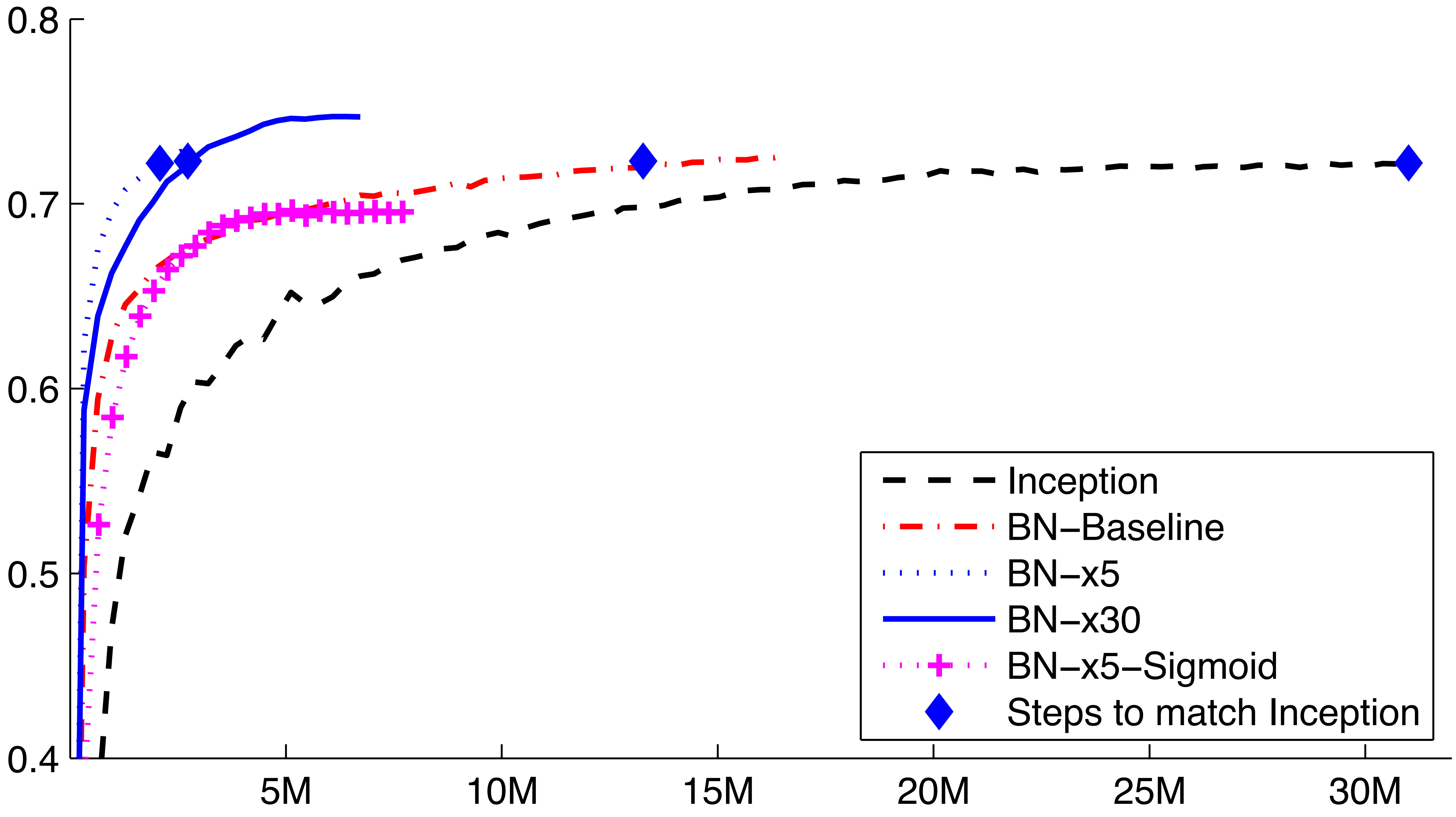
"Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," Ioffe and Szegedy 2015

Before SGD step



After SGD step

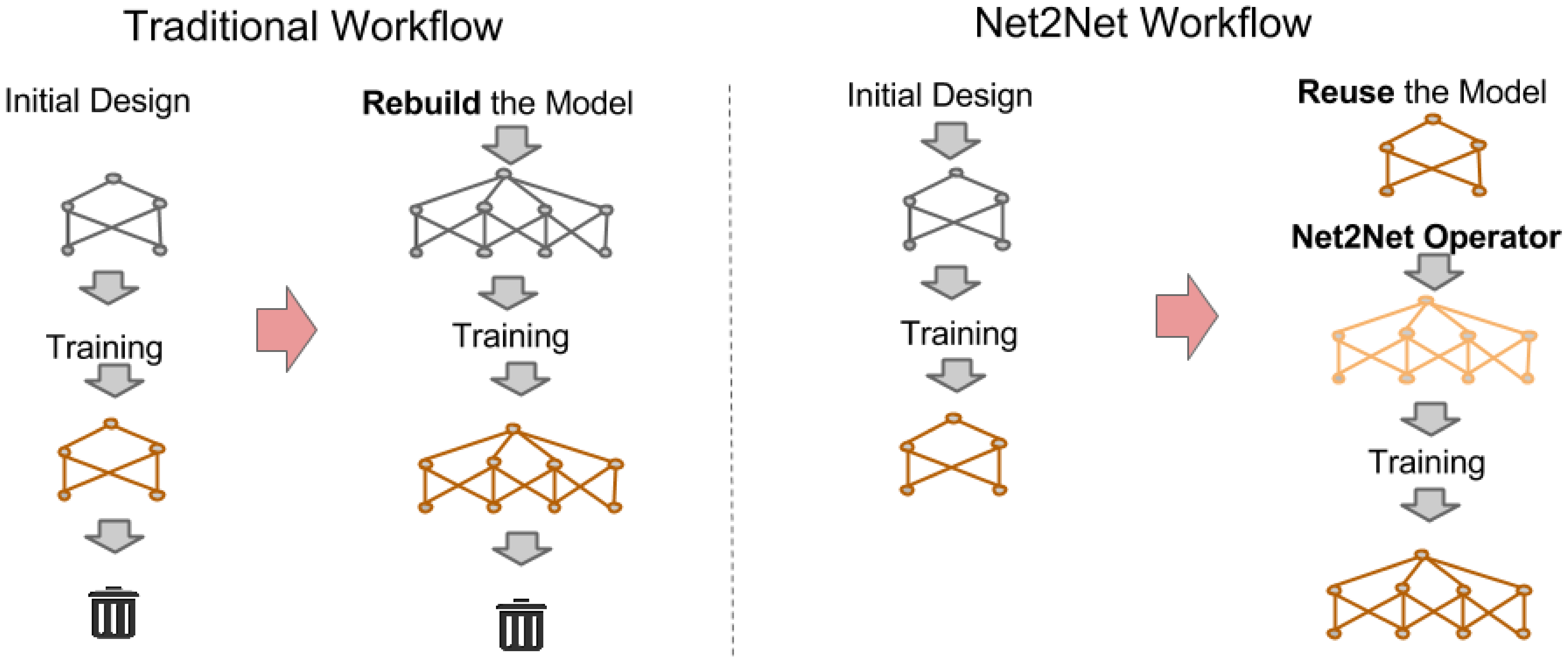
"Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," Ioffe and Szegedy 2015



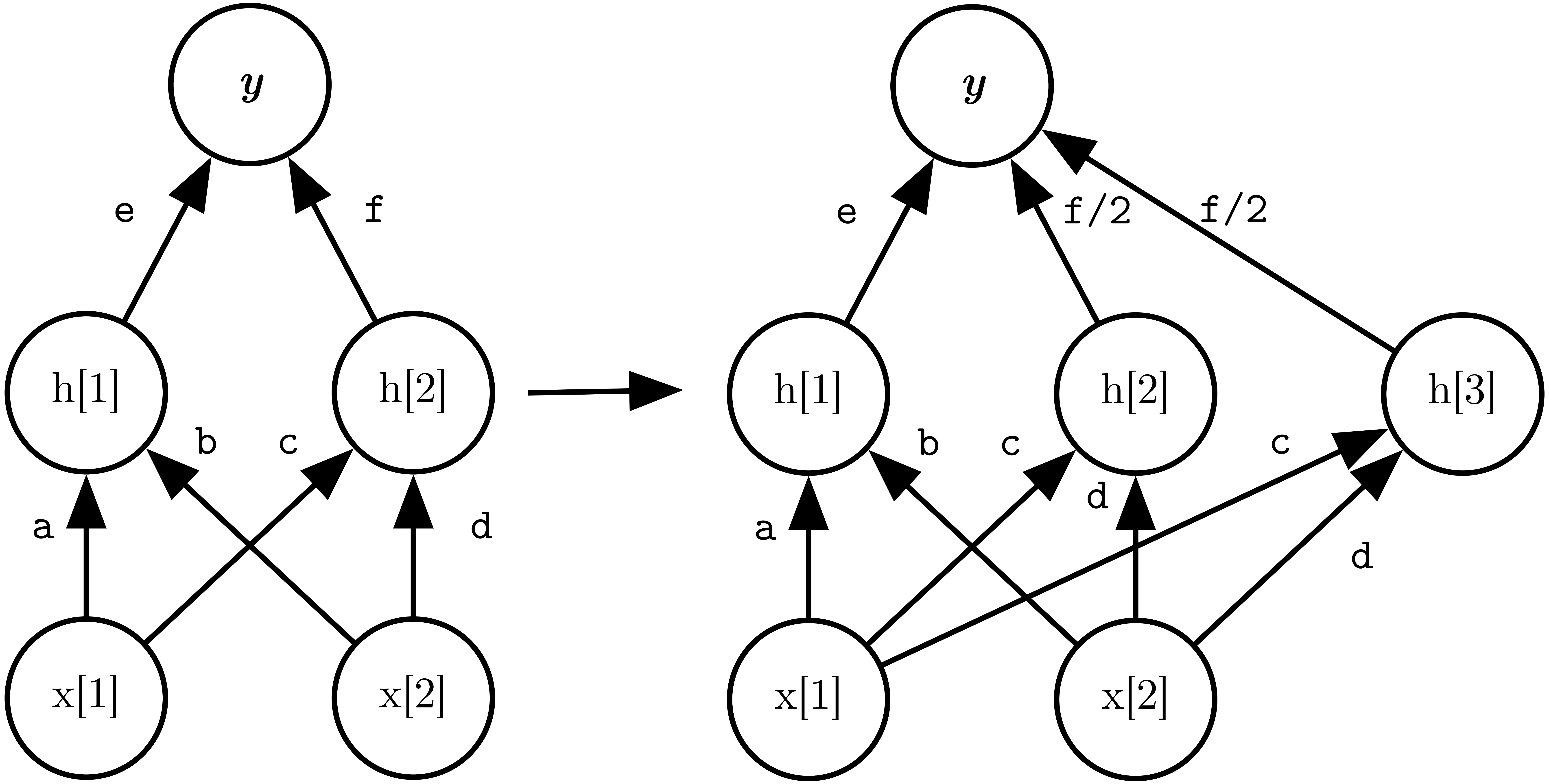
"Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," Ioffe and Szegedy 2015

Net2Net

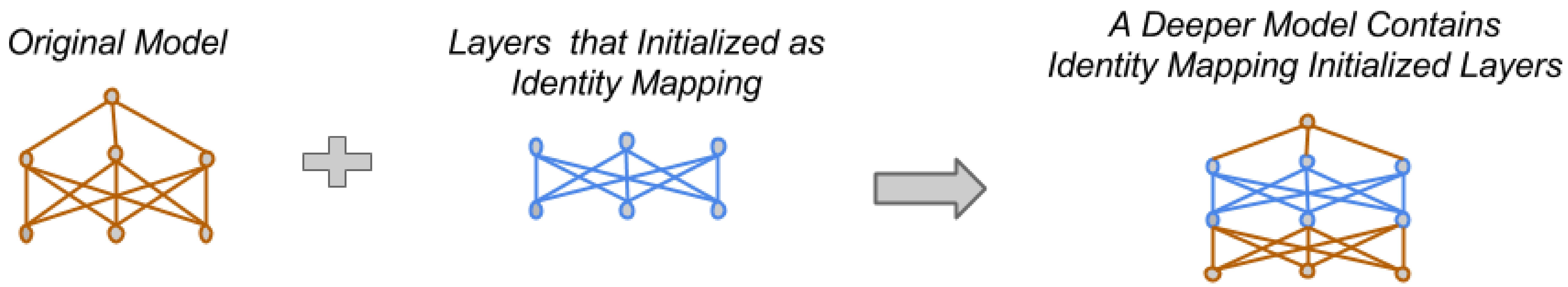
- Transferring knowledge between neural nets is hard
- Restrict the model architecture to make it easy

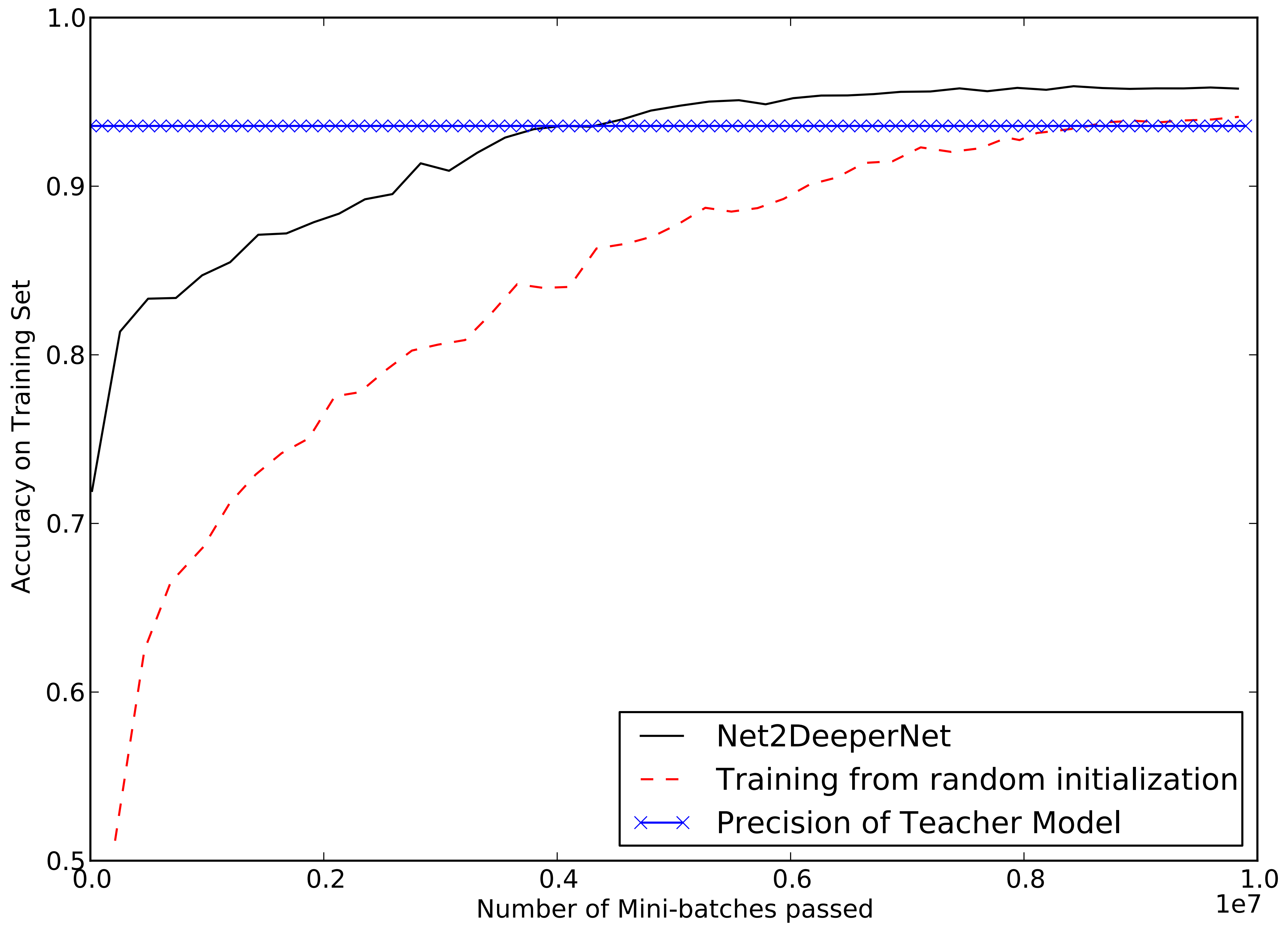


"Net2Net: Accelerating Learning via Knowledge Transfer." Chen, Goodfellow, and Shlens, submitted to ICLR 2016

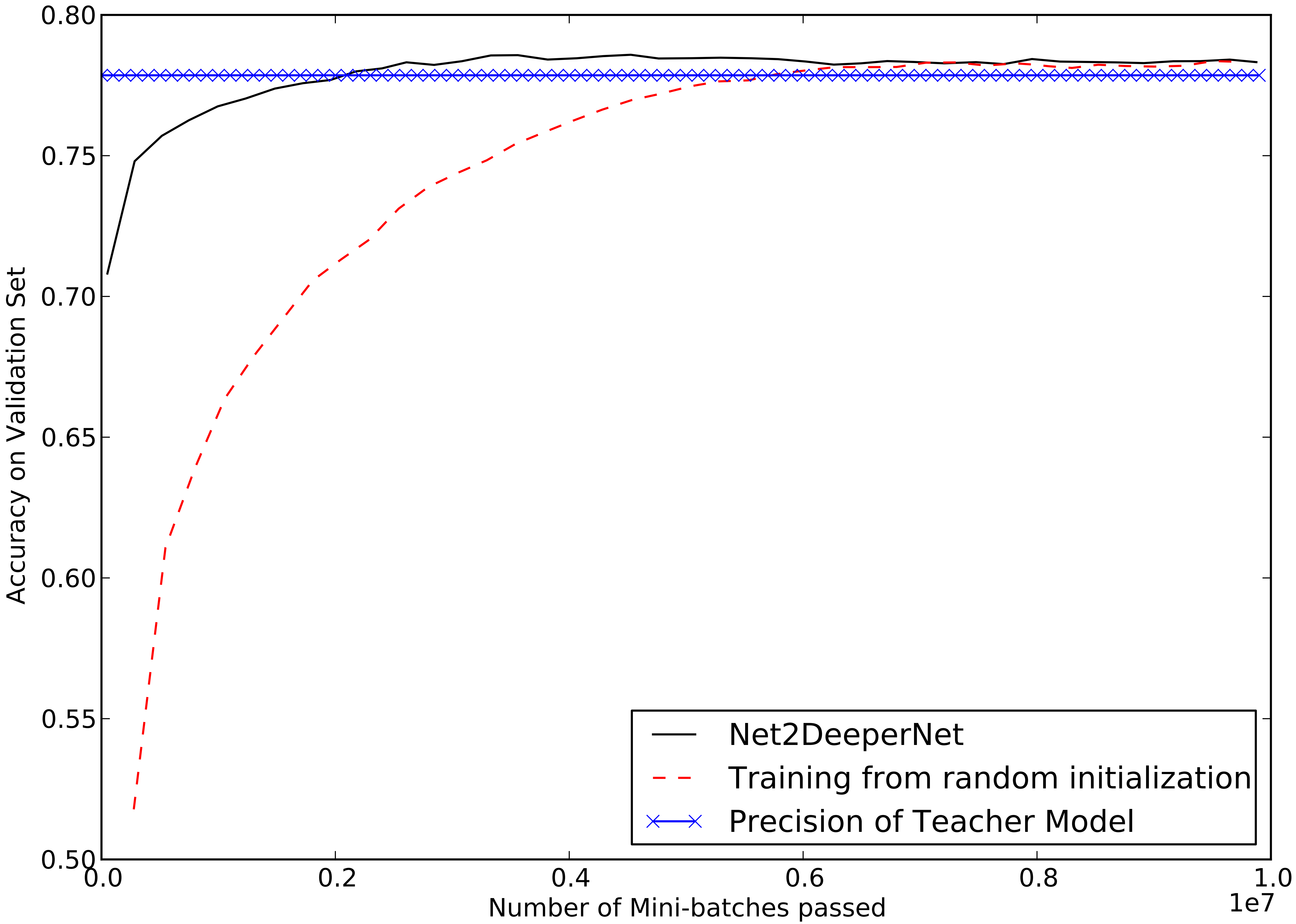


"Net2Net: Accelerating Learning via Knowledge Transfer." Chen, Goodfellow, and Shlens, submitted to ICLR 2016



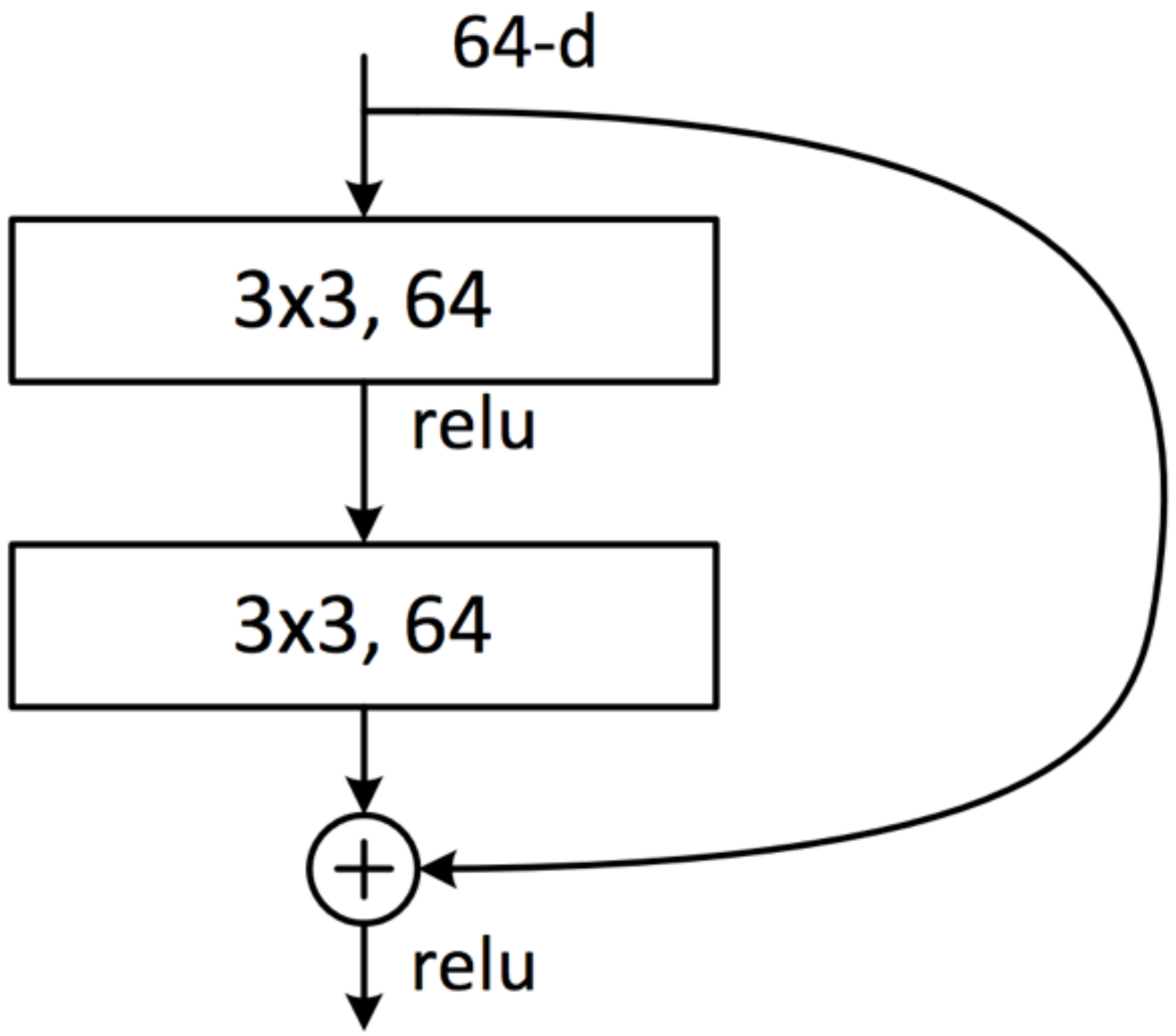


"Net2Net: Accelerating Learning via Knowledge Transfer." Chen, Goodfellow, and Shlens, submitted to ICLR 2016



"Net2Net: Accelerating Learning via Knowledge Transfer." Chen, Goodfellow, and Shlens, submitted to ICLR 2016

Residual Nets



He et al, 2015

- Similar to much older skip connections strategies
- Add much shorter paths from input to output while retaining depth
- Multi-step program initialized to sequence of no-ops

High-Level Lessons

- Strive for *success*, not *perfection*
- Simple optimization methods are successful
- A little model redesign goes farther than a lot of optimization algorithm redesign

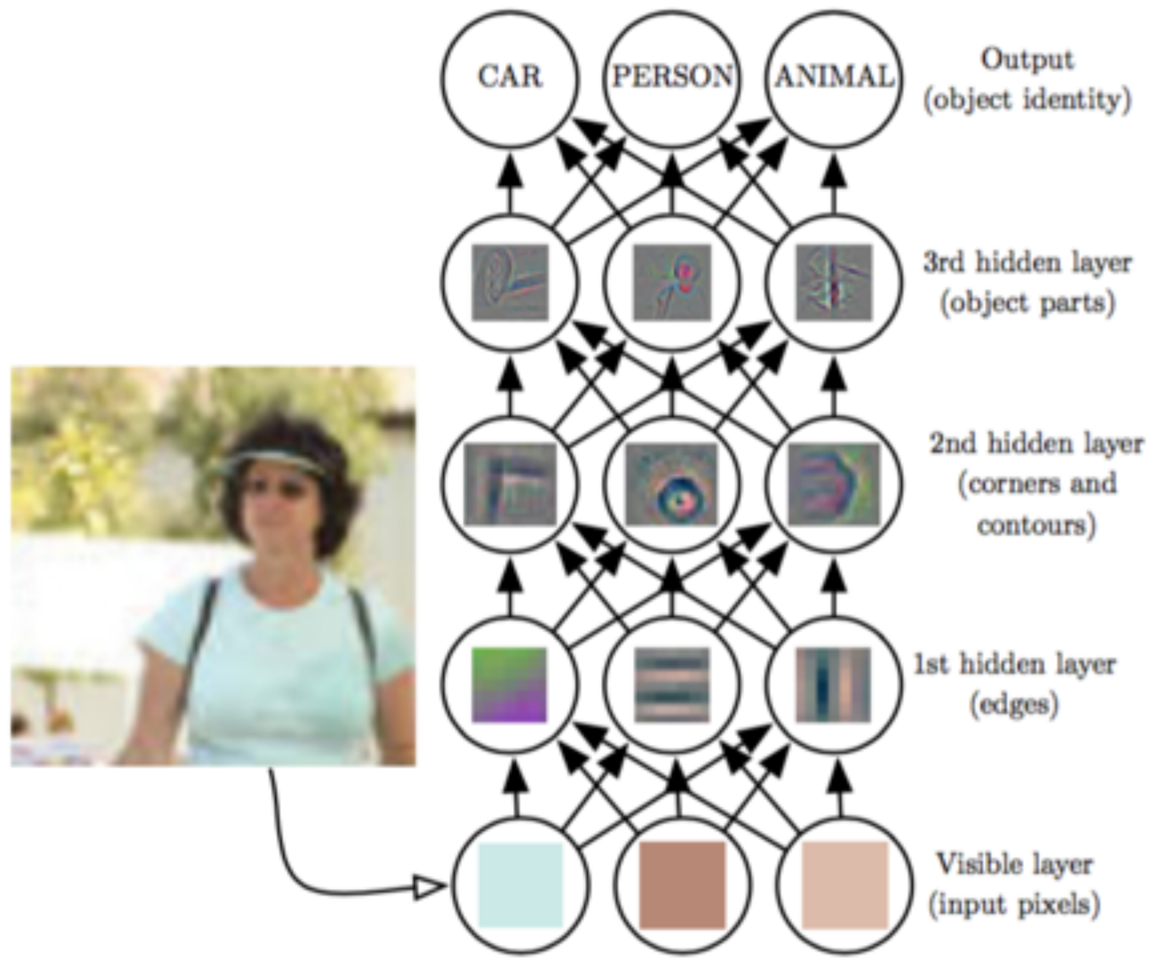


Figure 1.2: Illustration of a deep learning model. It is difficult for a computer to understand the meaning of raw sensory input data, such as this image represented as a collection of pixel values. The function mapping from a set of pixels to an object identity is very complicated. Learning or evaluating this mapping seems insurmountable if tackled directly. Deep learning resolves this difficulty by breaking the desired complicated mapping into a series of nested simple mappings, each described by a different layer of the model. The input is presented at the *visible layer*, so named because it contains the variables that we are able to observe. Then a series of *hidden layers* extracts increasingly abstract features from the image. These layers are called "hidden" because their values are not given in the data; instead the model must determine which concepts are useful for explaining the relationships in the observed data. The images here are visualizations of the kind of feature represented by each hidden unit. Given the pixels, the first layer can easily identify edges, by comparing the brightness of neighboring pixels. Given the first hidden layer's description of the edges, the second hidden layer can easily search for corners and extended contours, which are recognizable as collections of edges. Given the second hidden layer's description of the image in terms of corners and contours, the third hidden layer can detect entire parts of specific objects, by finding specific collections of contours and corners. Finally, this description of the image in terms of the object parts it contains can be used to recognize the objects present in the image. Images reproduced with permission from Zeiler and Fergus (2014).

Deep Learning

Goodfellow, Bengio, and Courville

www.deeplearningbook.org

In preparation for MIT Press