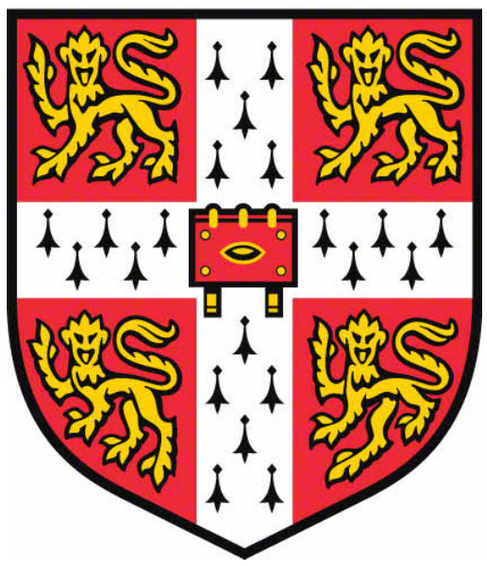
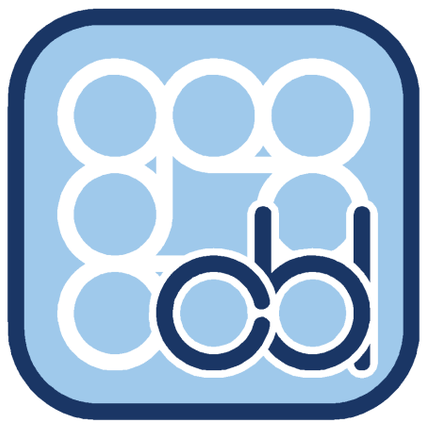
Structured Shrinkage Priors for Neural Networks

Eric Nalisnick

University of Cambridge





Computational and
Biological Learning
University of Cambridge

Outline

1 **Background:** Deep Learning and Modern Neural Networks

2 **Background:** Dropout Regularization

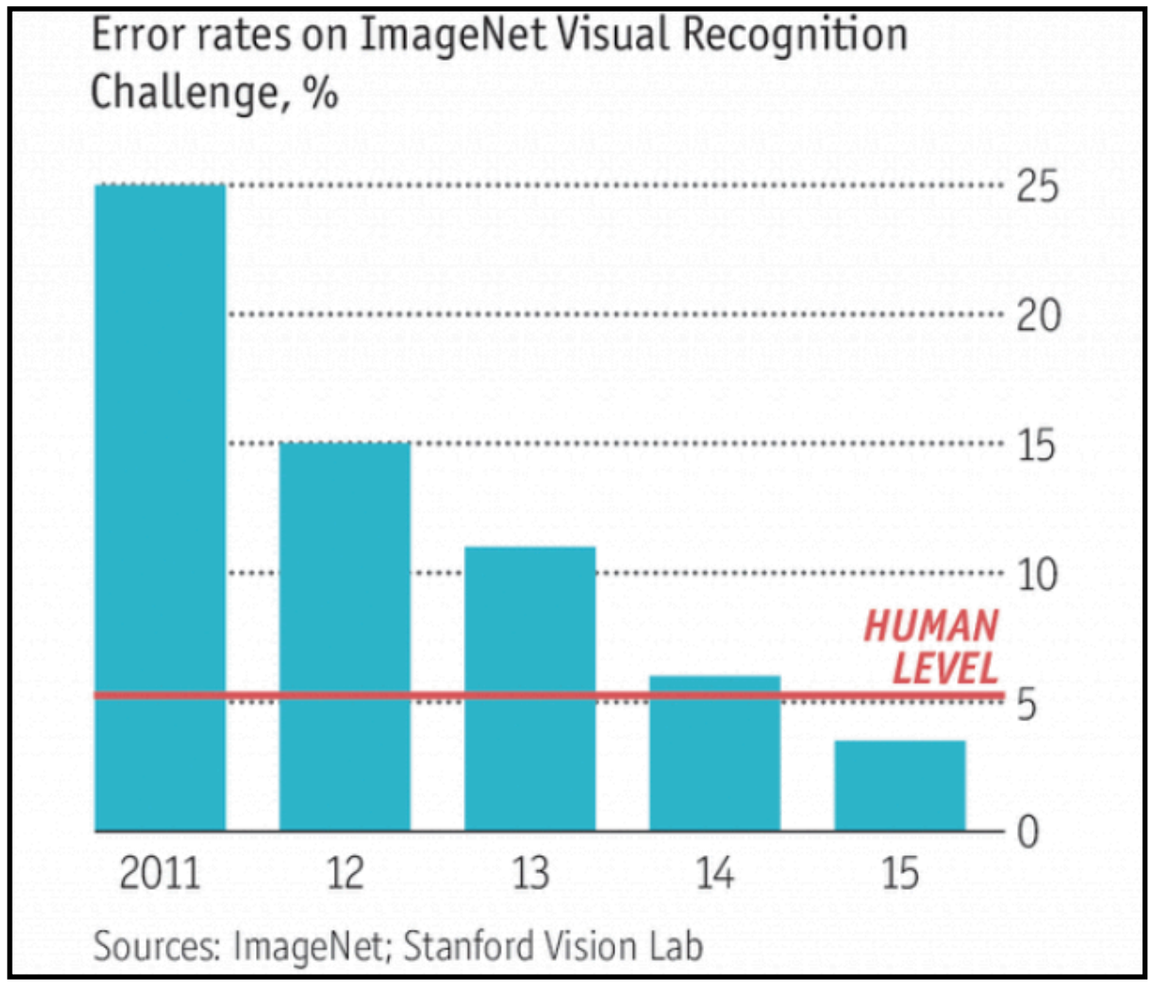
3 **Contribution:** Dropout as Bayesian Shrinkage

4 **Discussion:** Statistics and Deep Learning

Deep Learning and Neural Networks

Deep Learning Results

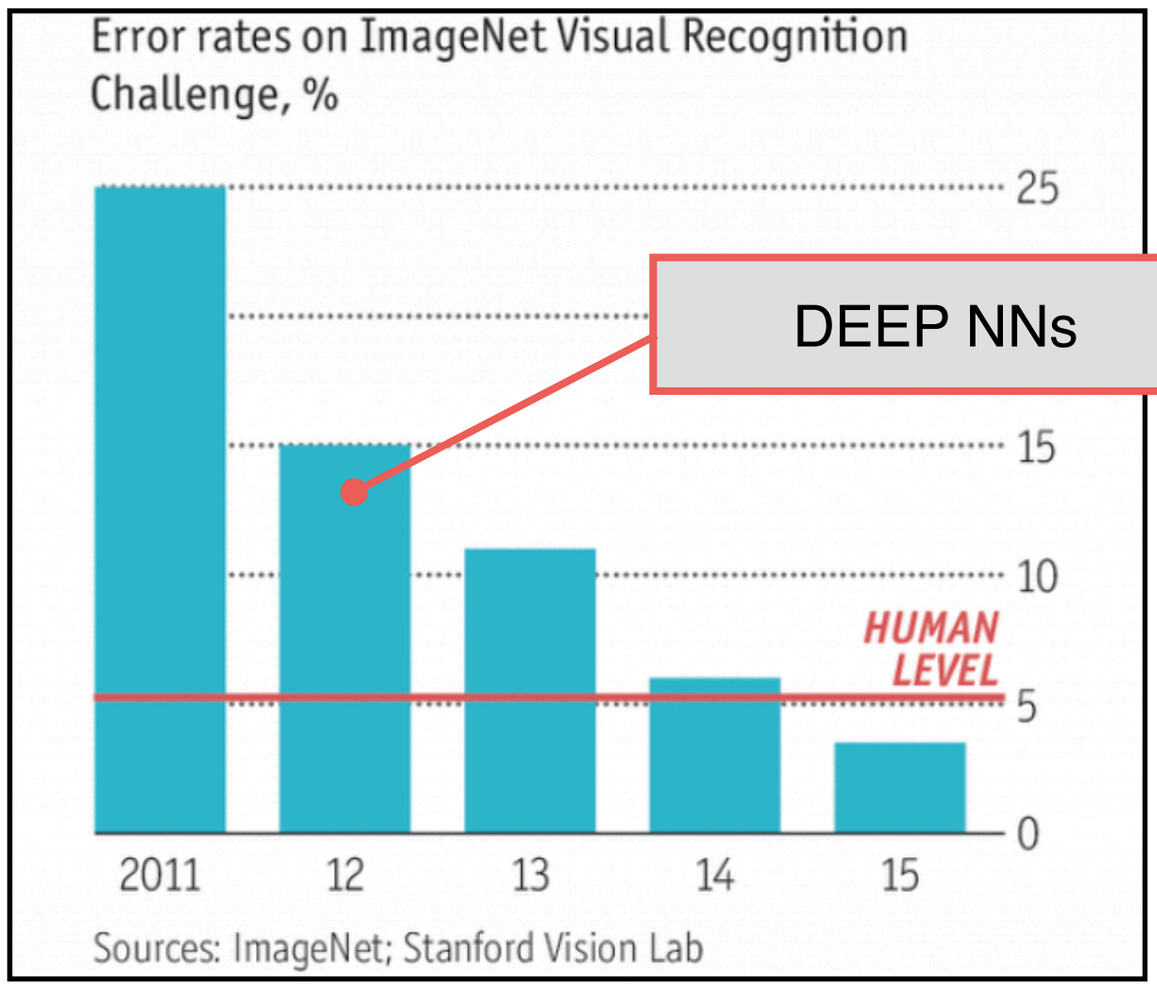
- **Computer Vision**: Results on ImageNet object classification dataset.



https://www.economist.com/news/special-report/21700756-artificial-intelligence-boom-based-old-idea-modern-twist-not

Deep Learning Results

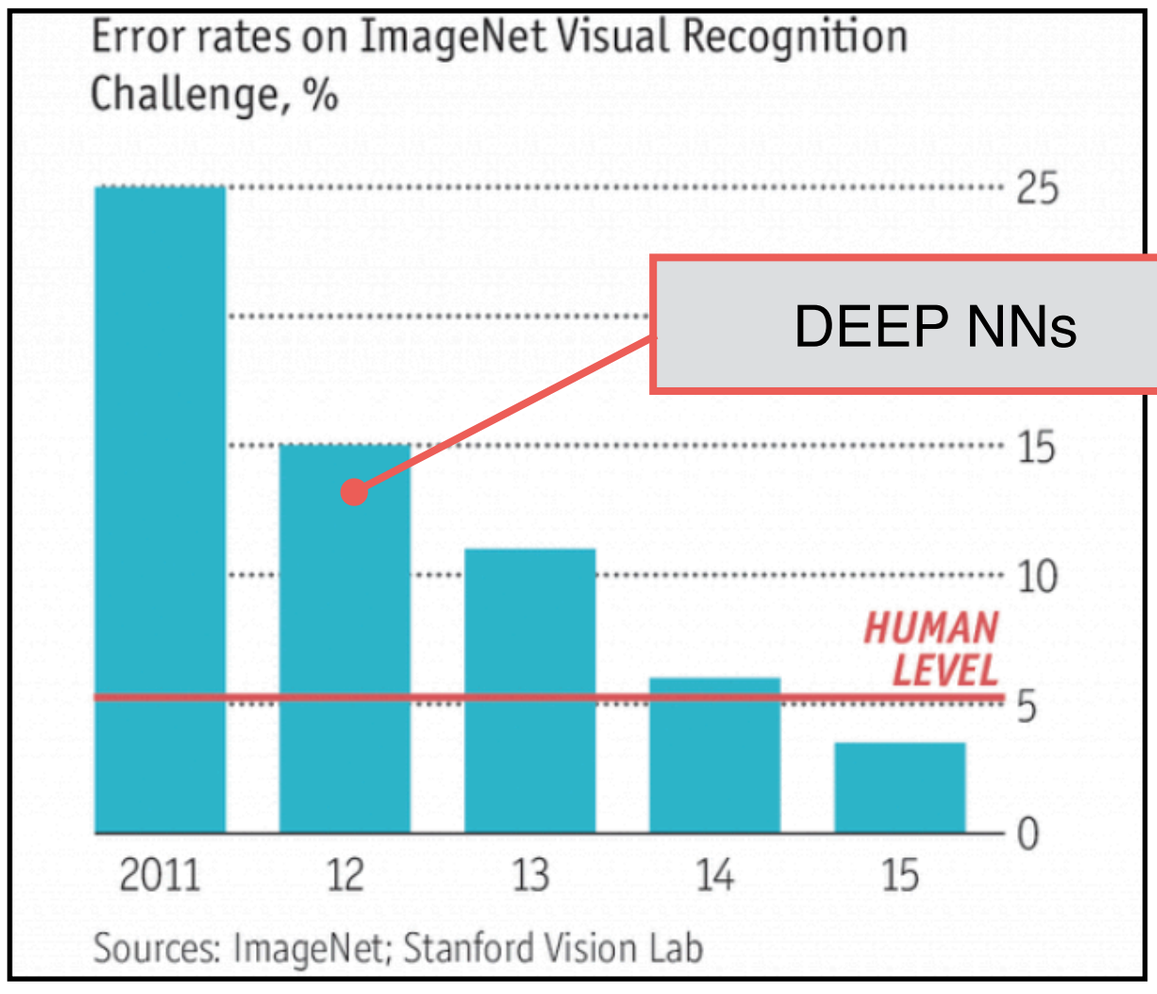
- **Computer Vision**: Results on ImageNet object classification dataset.

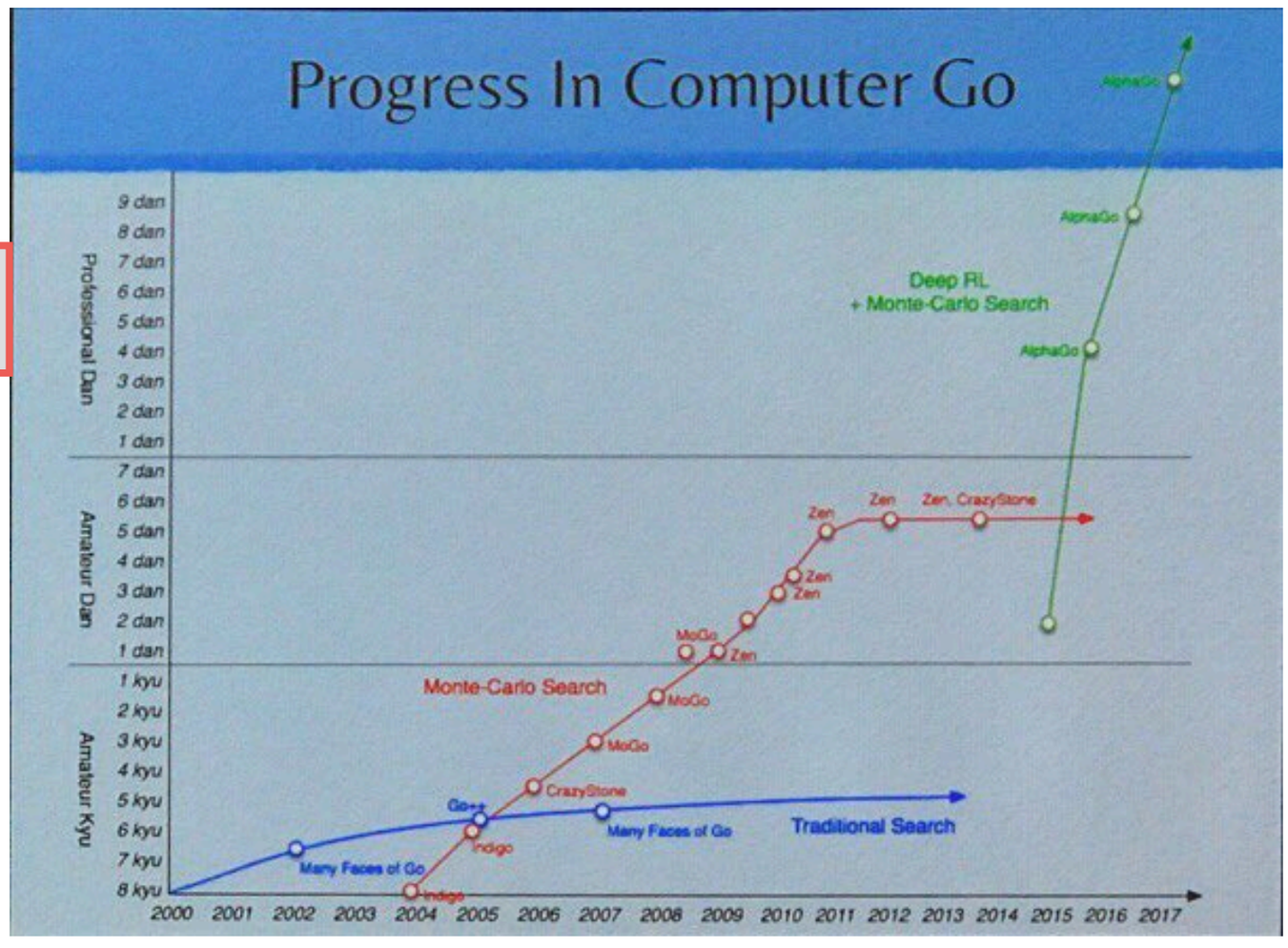


https://www.economist.com/news/special-report/21700756-artificial-intelligence-boom-based-old-idea-modern-twist-not

Deep Learning Results

- **Computer Vision**: Results on ImageNet object classification dataset.
- **Reinforcement Learning**: Results in playing Go.

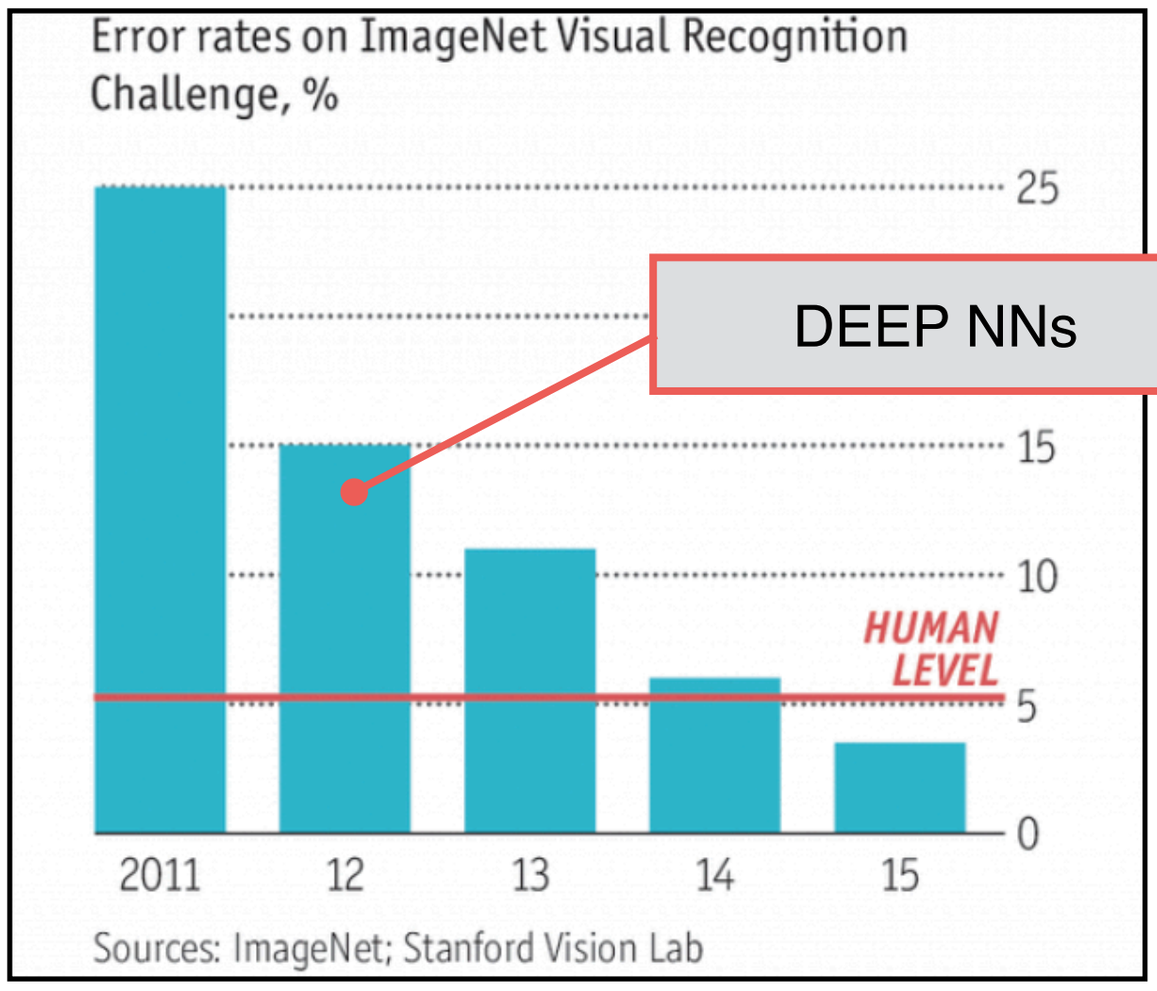




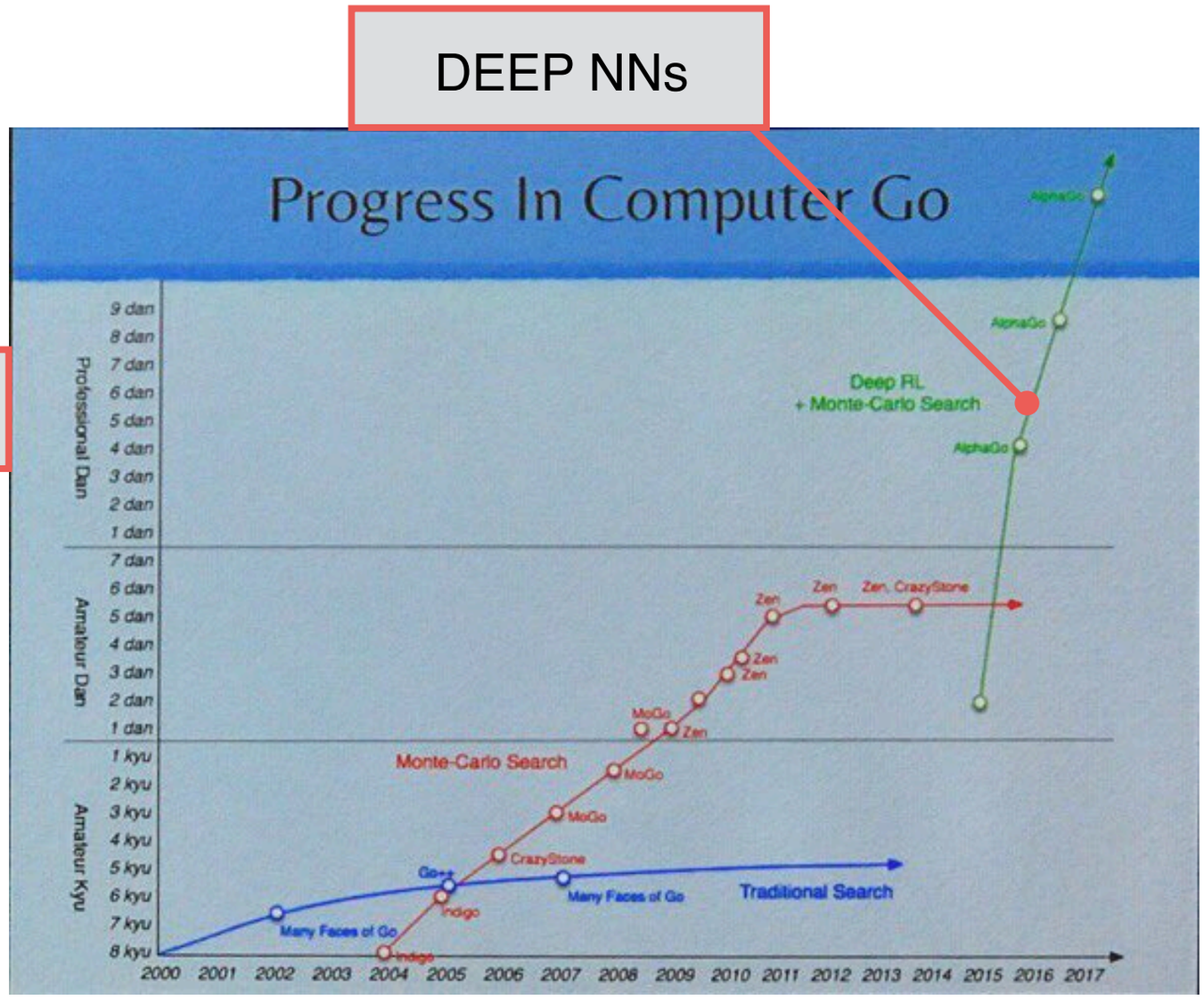
https://www.economist.com/news/special-report/21700756-artificial-intelligence-boom-based-old-idea-modern-twist-not

Deep Learning Results

- **Computer Vision**: Results on ImageNet object classification dataset.



- **Reinforcement Learning**: Results in playing Go.



https://www.economist.com/news/special-report/21700756-artificial-intelligence-boom-based-old-idea-modern-twist-not

Generalized Linear Models (GLMs)

$$\mathbf{x}_i \in \mathbb{R}^d, \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$$

Link Function

$$g^{-1}(\mathbf{x}_i^T \mathbf{w} + b) = \mathbb{E}[\mathbf{y}_i | \mathbf{x}_i]$$

Generalized Linear Models (GLMs)

$$\mathbf{x}_i \in \mathbb{R}^d, \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$$

Link Function

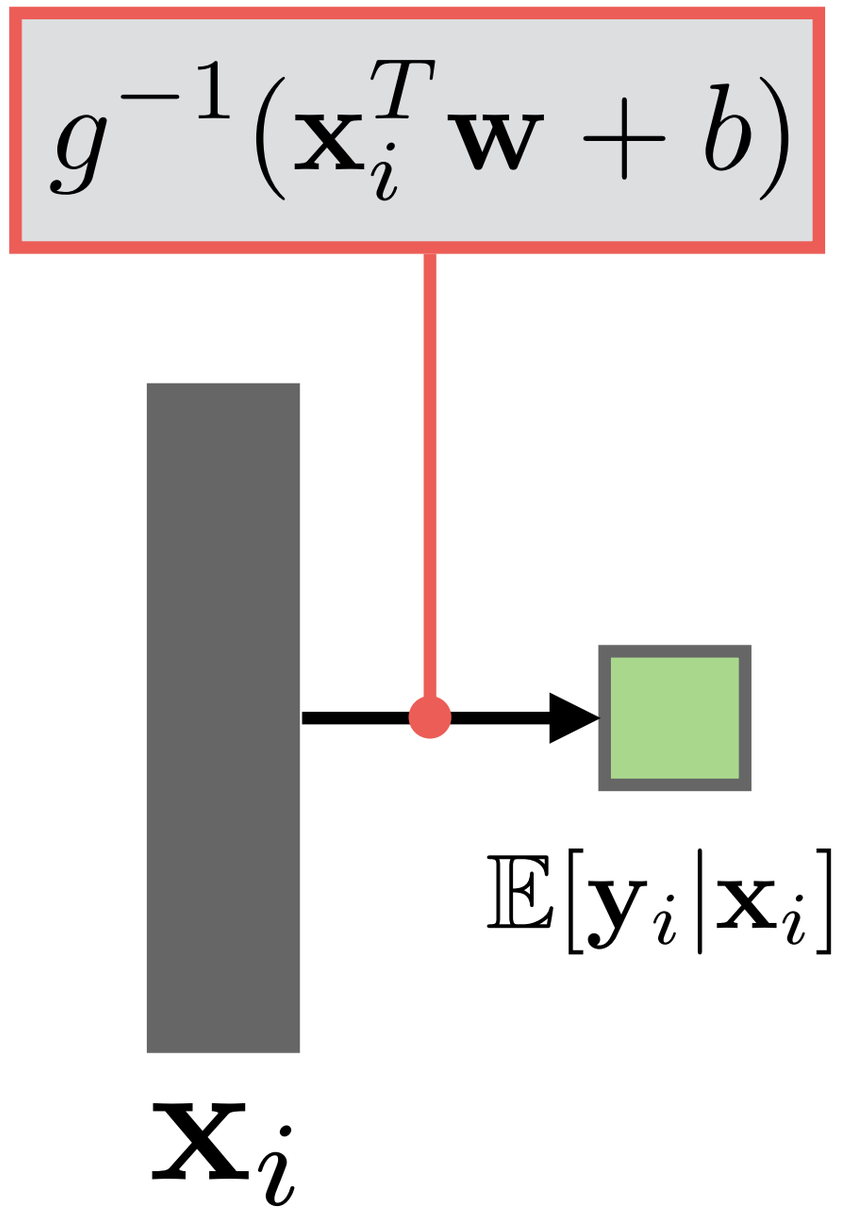
$$g^{-1}(\mathbf{x}_i^T \mathbf{w} + b) = \mathbb{E}[\mathbf{y}_i | \mathbf{x}_i]$$

Example: Logistic Regression

$$\mathbf{y}_i \in \{0, 1\} \qquad g^{-1}(z) = \frac{1}{1 + e^{-z}}$$

THE LOGISTIC FUNCTION

Generalized Linear Models (GLMs)



Recursive GLMs

Define a latent feature via a GLM:

$$g_1^{-1}(\mathbf{x}_i^T \mathbf{w}_1 + b_1) = h_i$$

Recursive GLMs

Define a latent feature via a GLM:

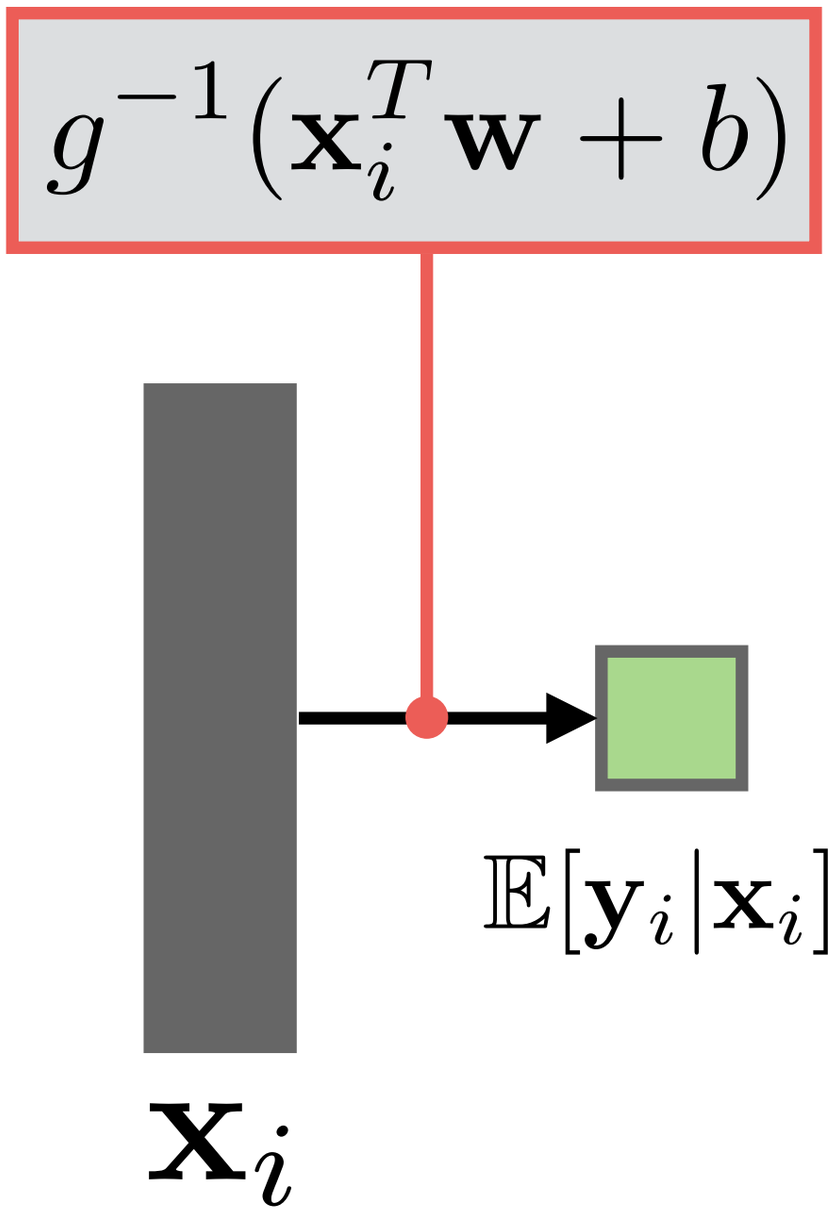
$$g_1^{-1}(\mathbf{x}_i^T \mathbf{w}_1 + b_1) = h_i$$

Define GLM on latent feature:

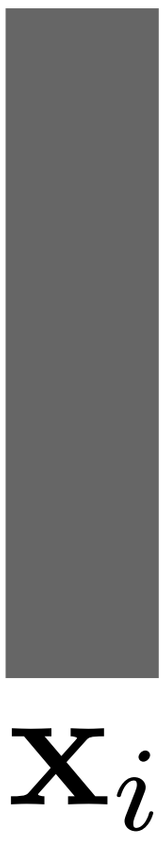
$$g_2^{-1}(h_i w_2 + b_2) = \mathbb{E}[\mathbf{y}_i | \mathbf{x}_i]$$

$$w_2 \in \mathbb{R}, b_2 \in \mathbb{R}$$

Recursive GLMs



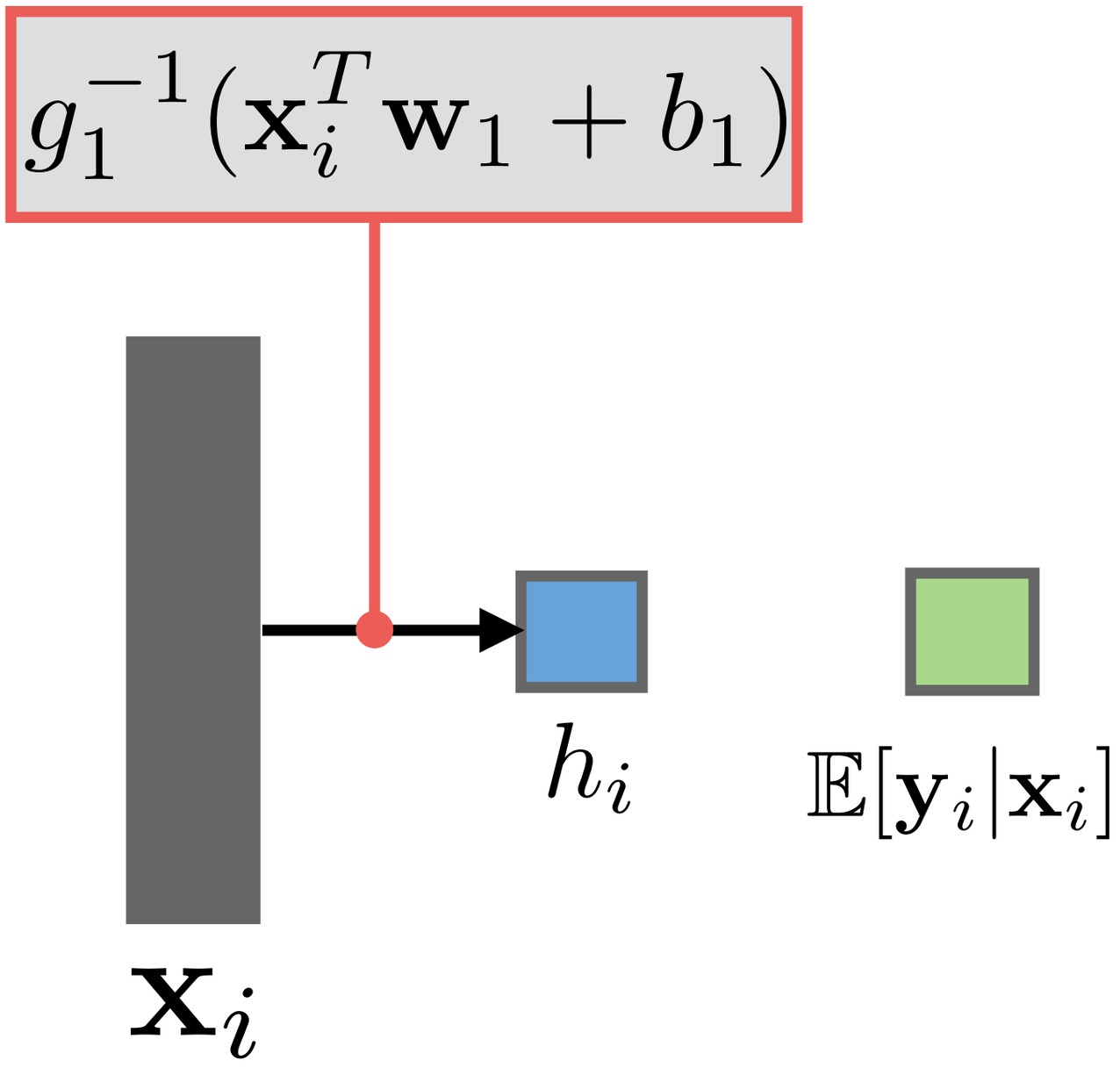
Recursive GLMs



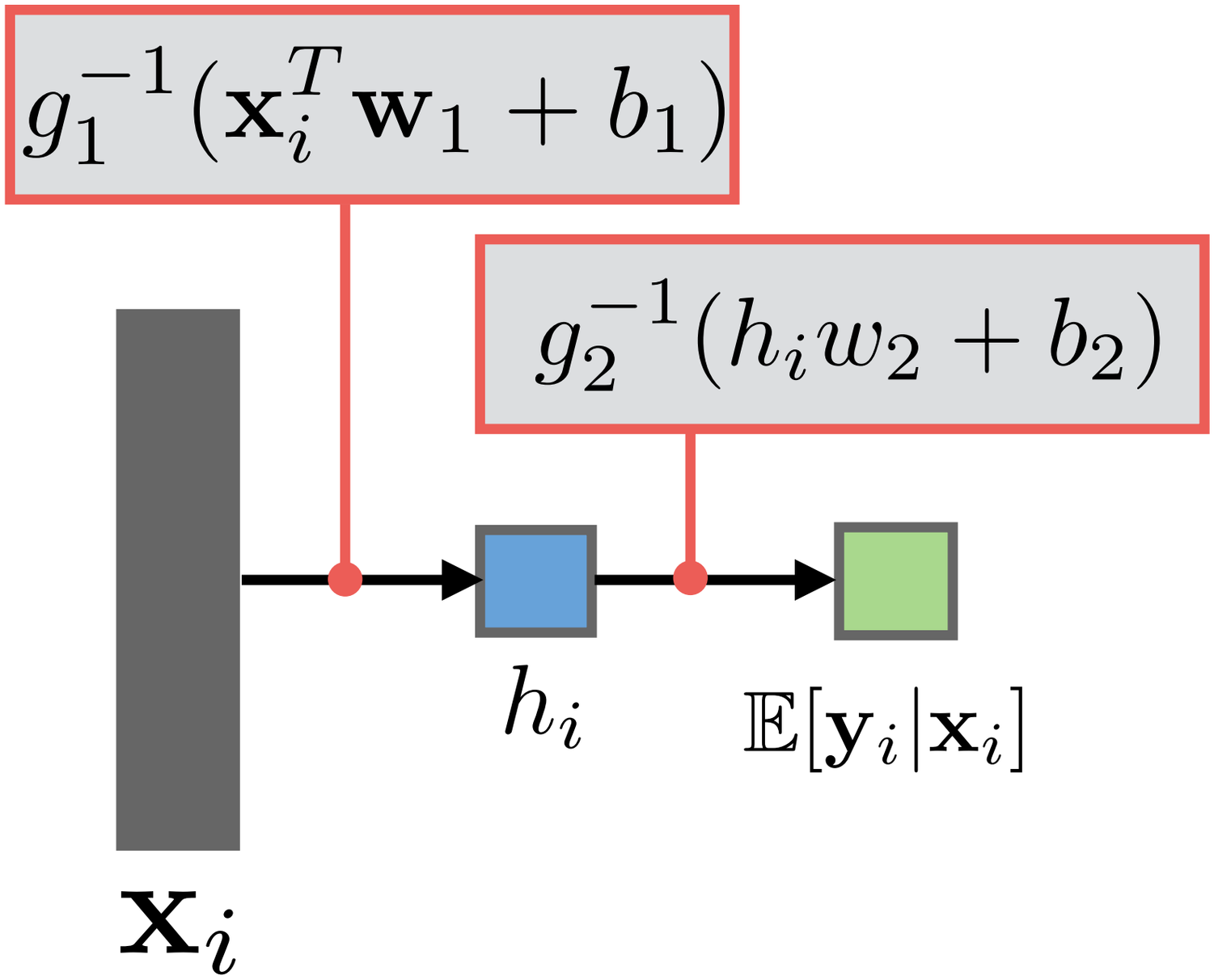
$\mathbf{x}_i$

$\mathbb{E}[\mathbf{y}_i|\mathbf{x}_i]$

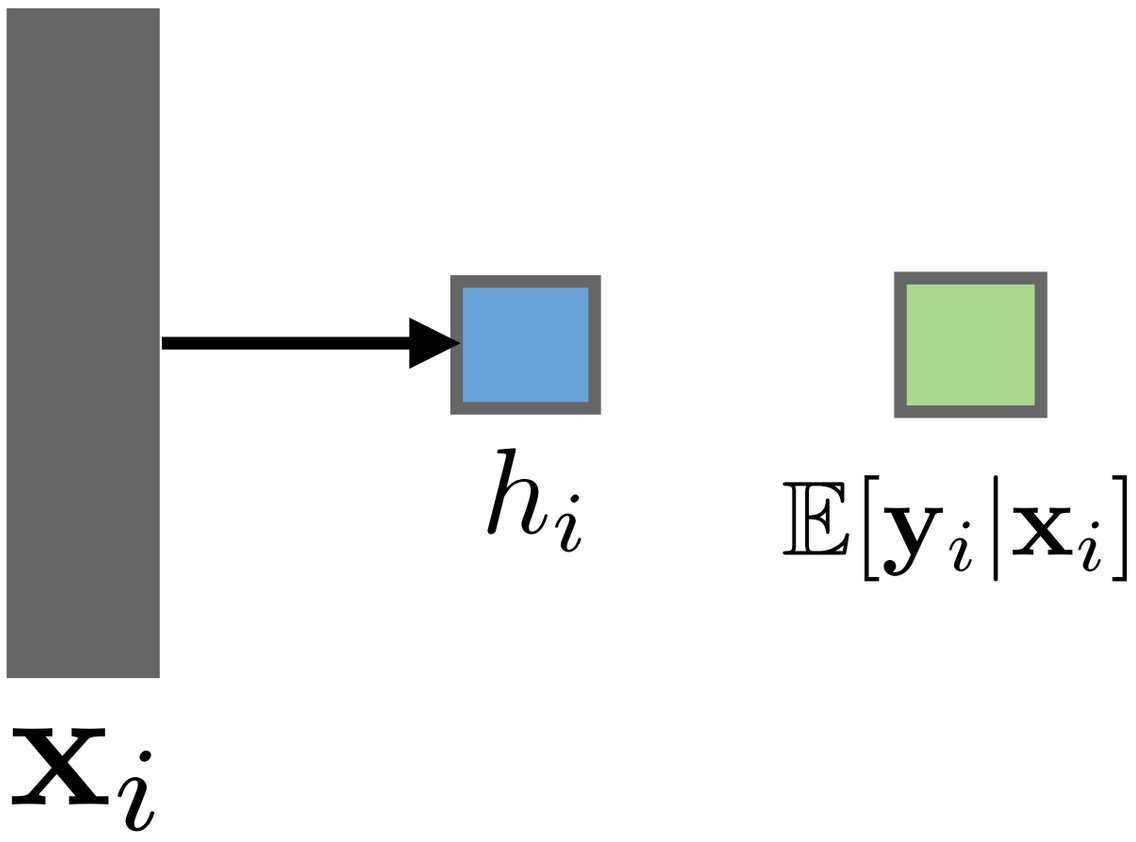
Recursive GLMs



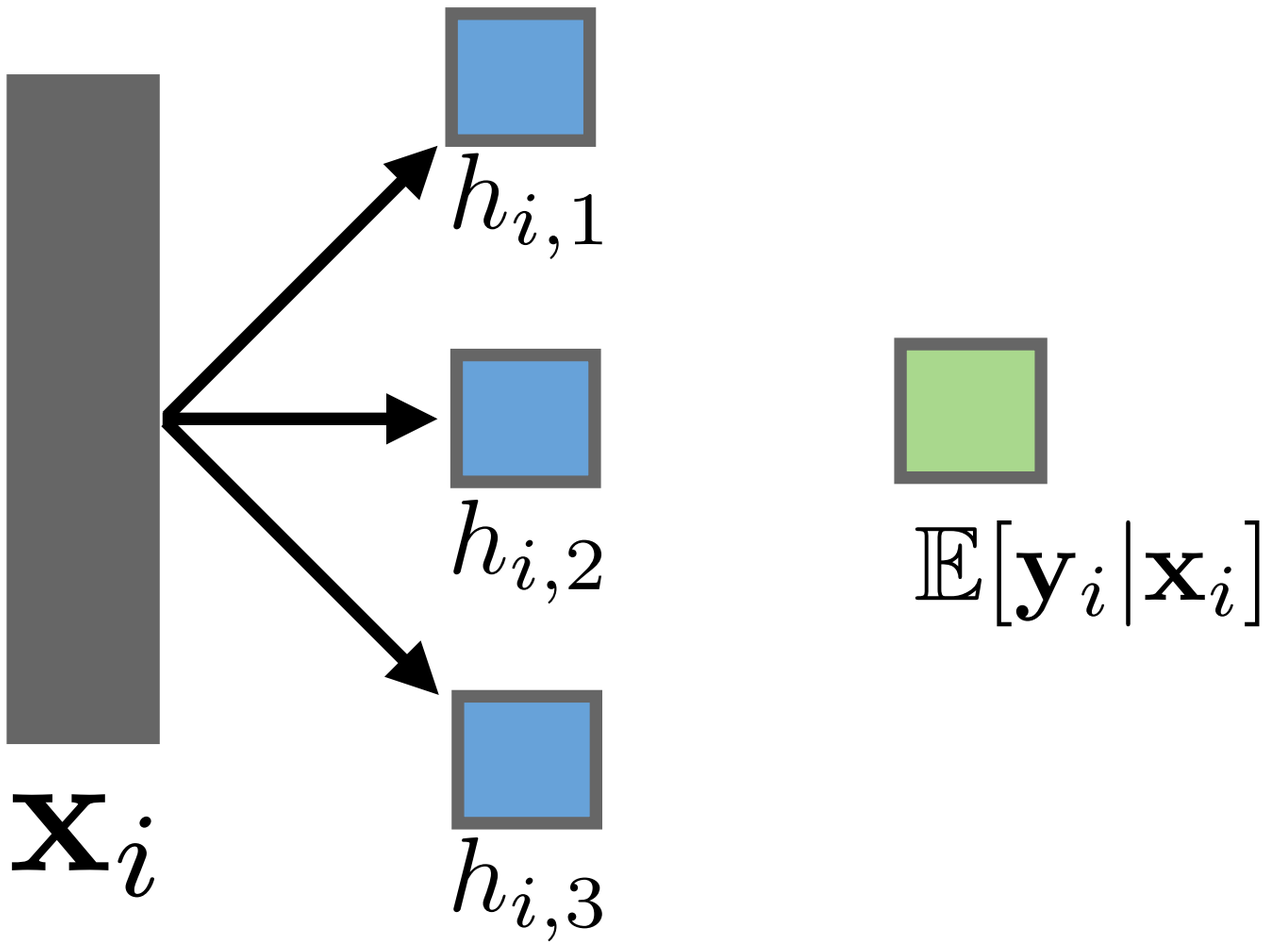
Recursive GLMs



Building Neural Nets from Recursive GLMs

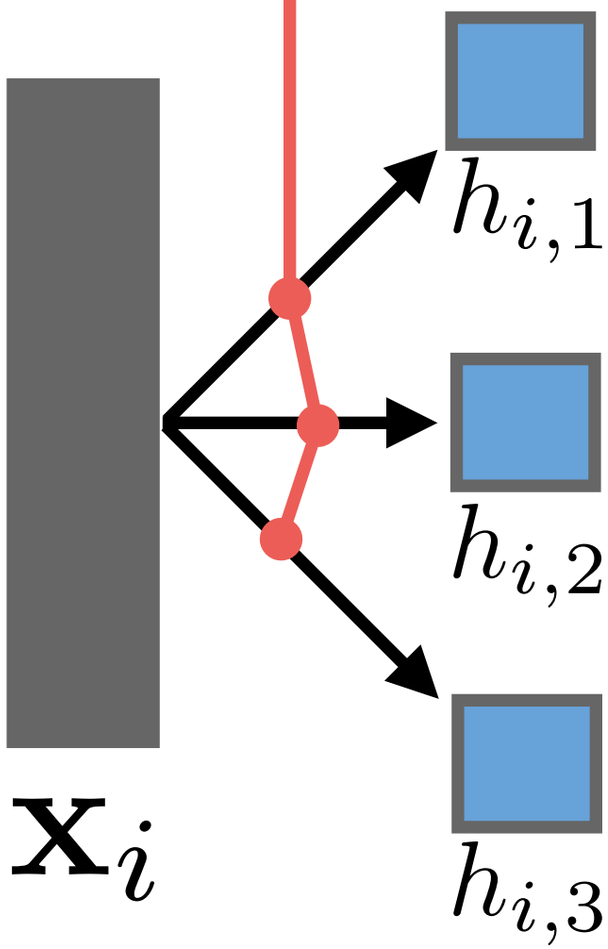


Building Neural Nets from Recursive GLMs



Building Neural Nets from Recursive GLMs

$$h_{i,j} = g_{1,j}^{-1}(\mathbf{x}_i^T \mathbf{w}_{1,j} + b_j)$$

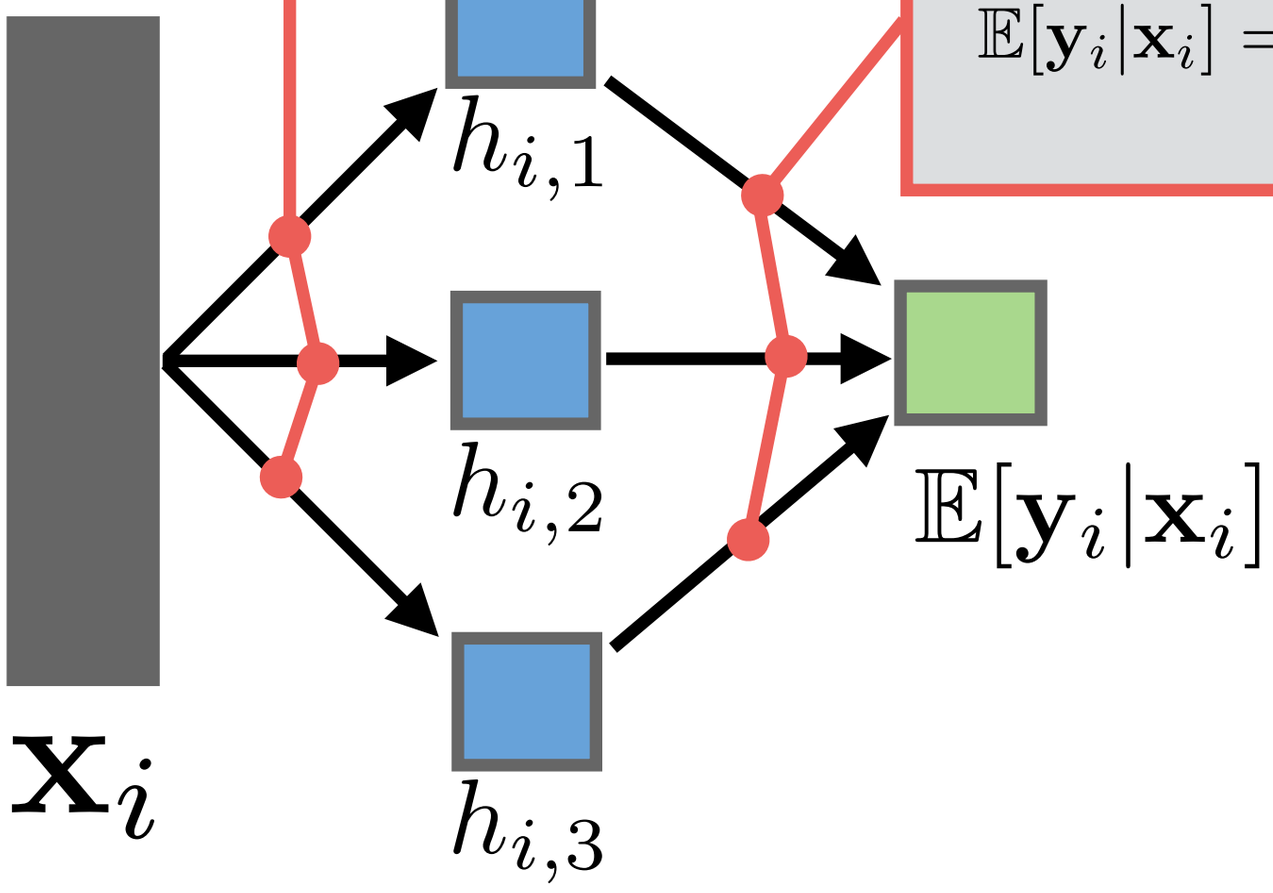


$\mathbb{E}[\mathbf{y}_i|\mathbf{x}_i]$

Building Neural Nets from Recursive GLMs

$$h_{i,j} = g_{1,j}^{-1}(\mathbf{x}_i^T \mathbf{w}_{1,j} + b_j)$$

$$\mathbb{E}[\mathbf{y}_i|\mathbf{x}_i] = g_2^{-1}\left(\sum_{j=1}^{H} h_{i,j} w_{2,j} + b_2\right)$$

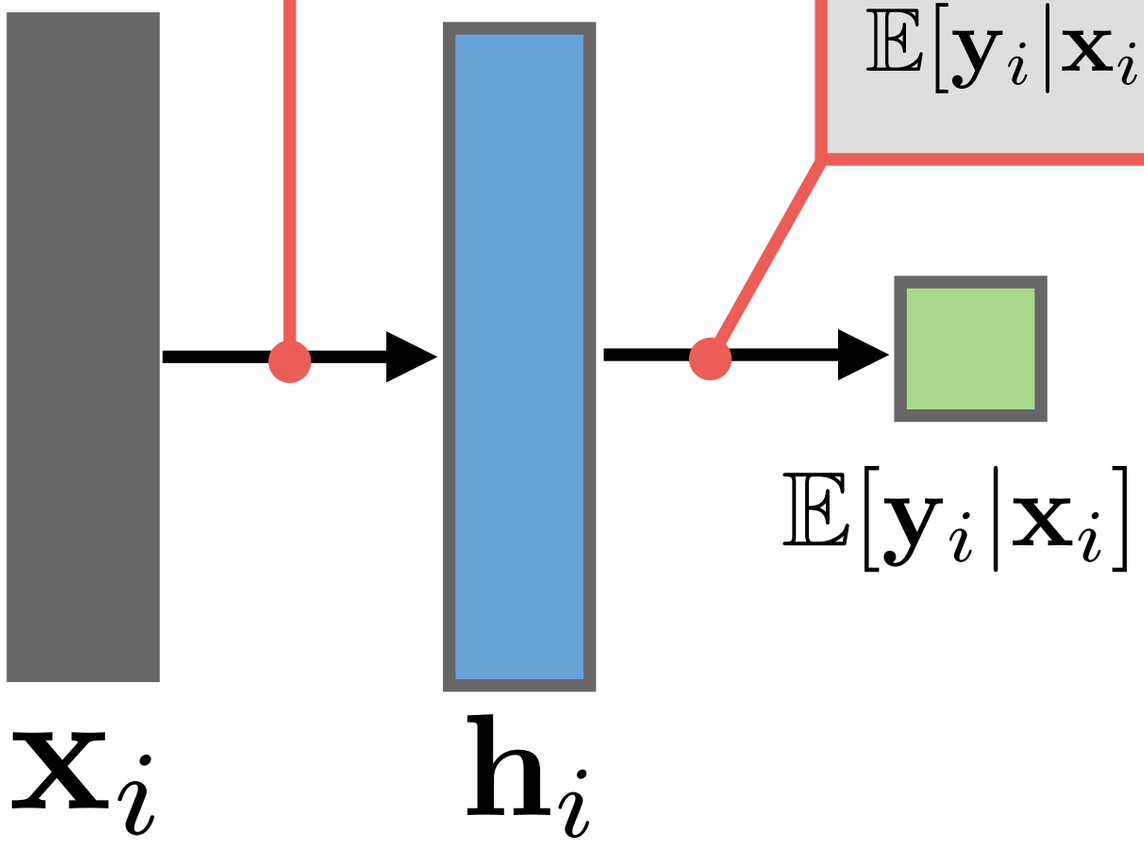


Building Neural Nets from Recursive GLMs

$\mathbf{h}_i \in \mathbb{R}^H, \mathbf{W}_1 \in \mathbb{R}^{d \times H}, \mathbf{b}_1 \in \mathbb{R}^H$

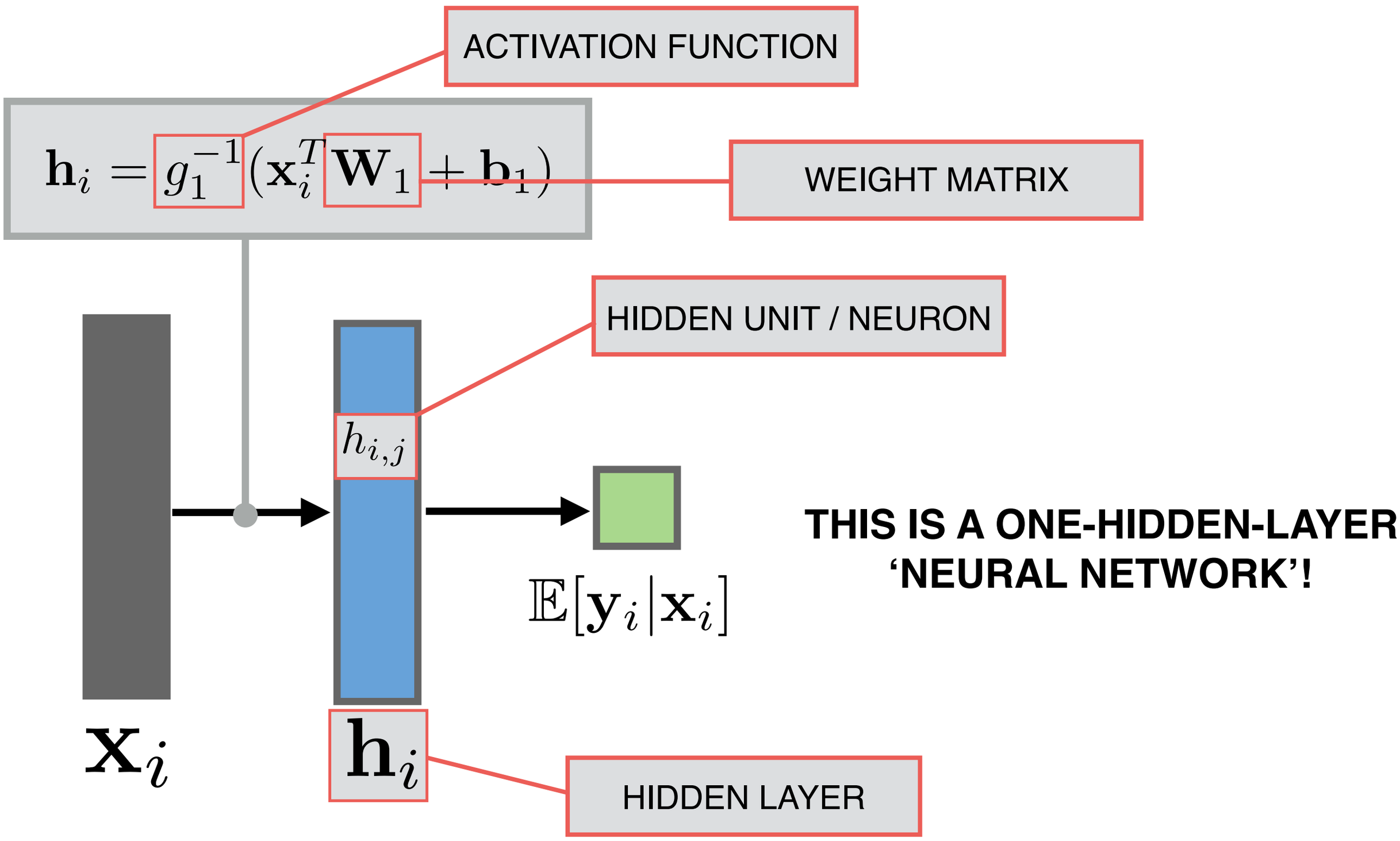
$$\mathbf{h}_i = g_1^{-1}(\mathbf{x}_i^T \mathbf{W}_1 + \mathbf{b}_1)$$

$$\mathbb{E}[\mathbf{y}_i|\mathbf{x}_i] = g_2^{-1}(\mathbf{h}_i \mathbf{w}_2 + b_2)$$



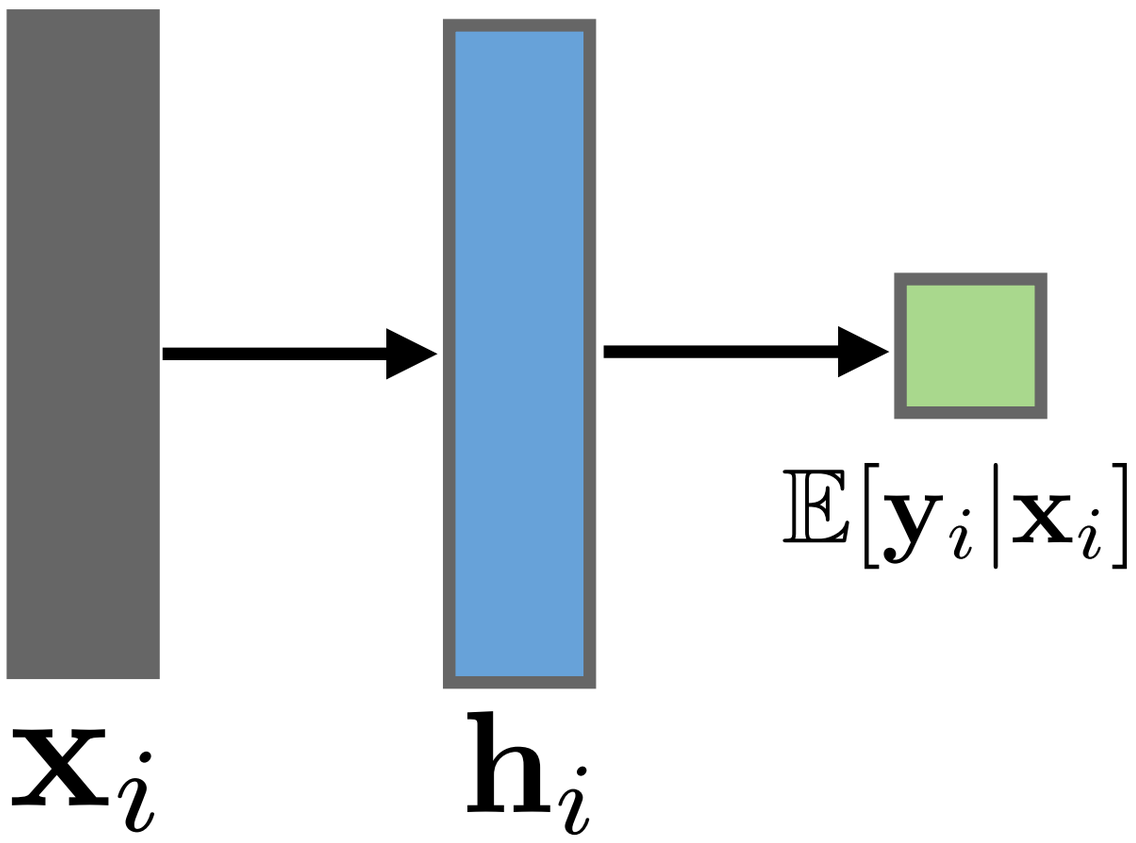
THIS IS A ONE-HIDDEN-LAYER ‘NEURAL NETWORK’!

Building Neural Nets from Recursive GLMs

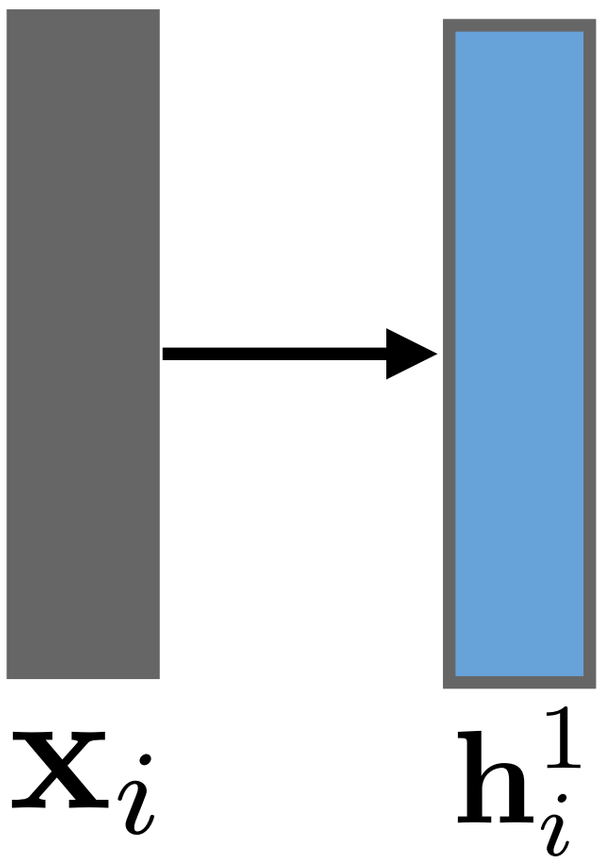


THIS IS A ONE-HIDDEN-LAYER ‘NEURAL NETWORK’!

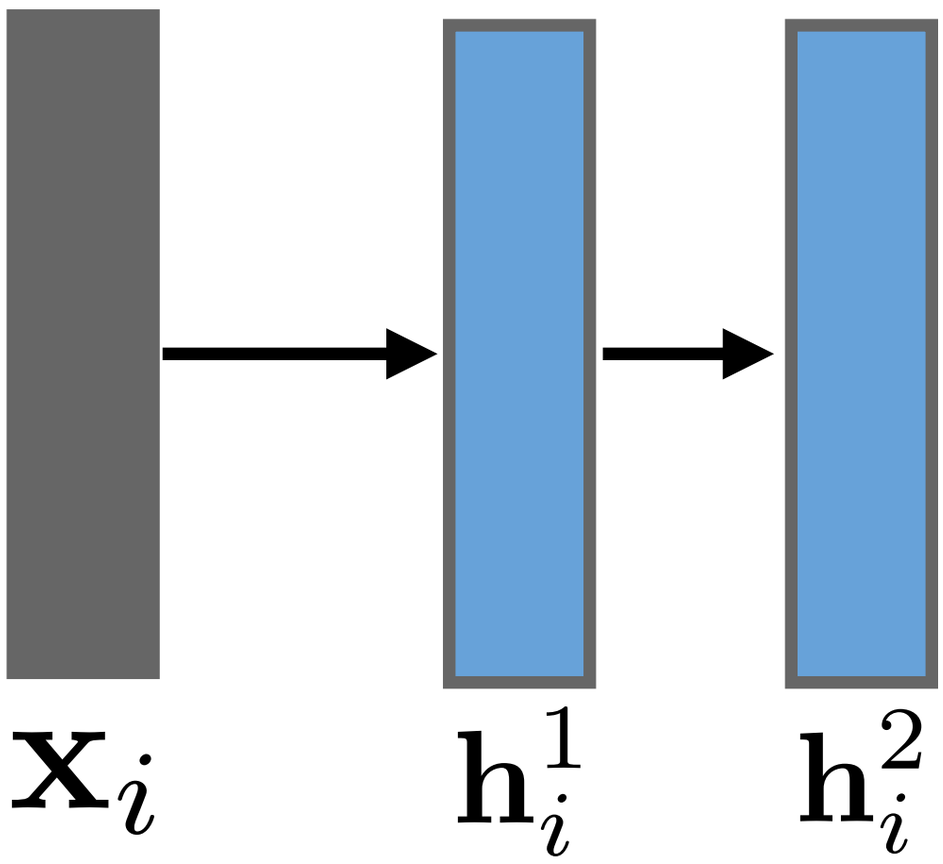
Deep Neural Networks

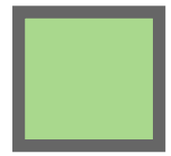


Deep Neural Networks



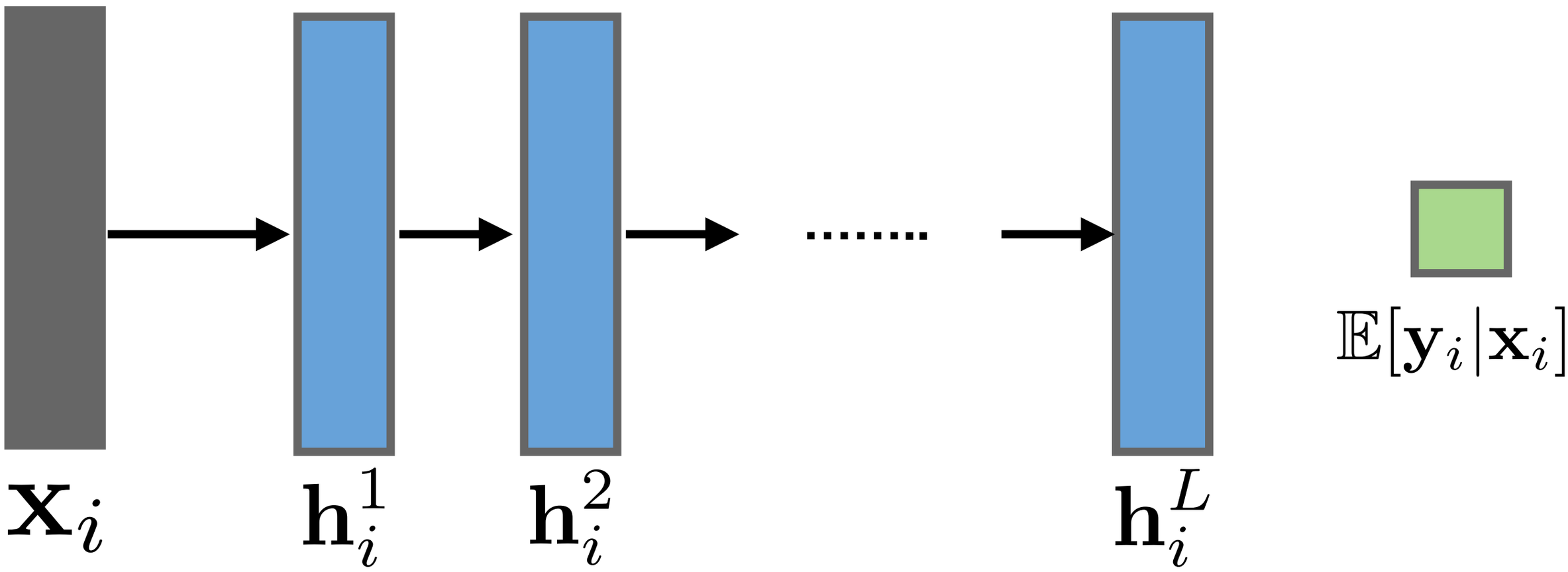
Deep Neural Networks





$\mathbb{E}[\mathbf{y}_i|\mathbf{x}_i]$

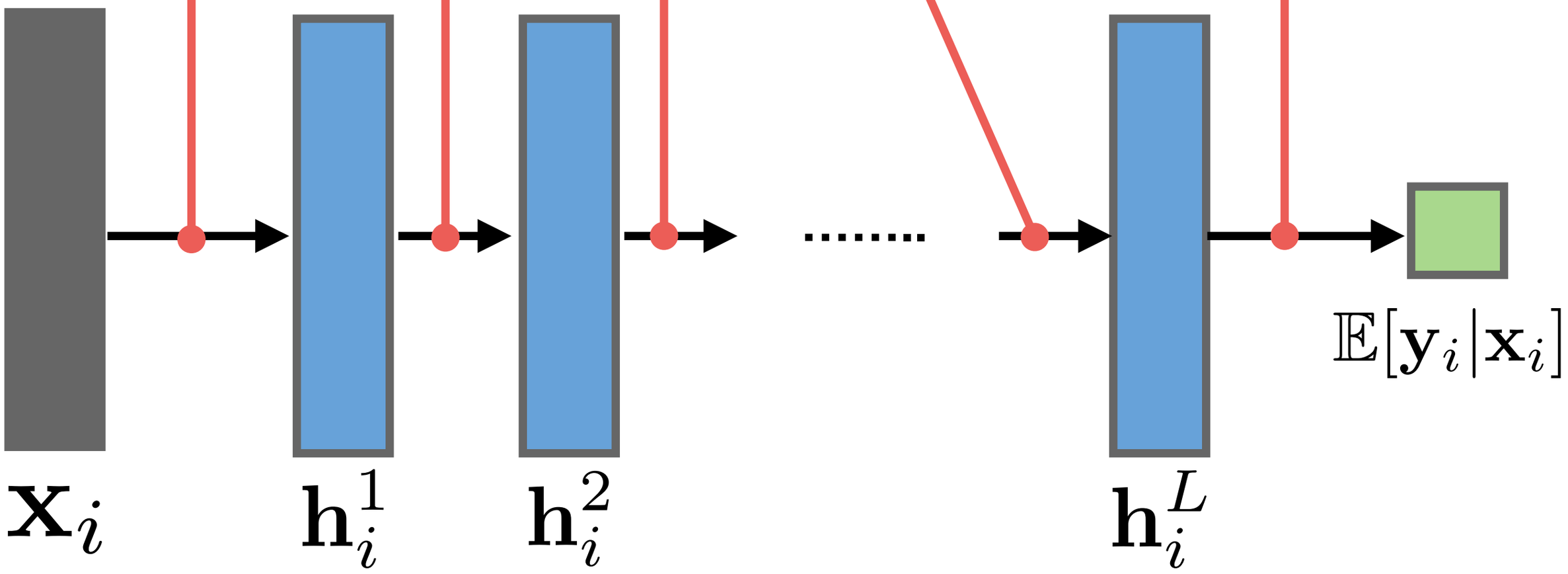
Deep Neural Networks



Deep Neural Networks

$$\mathbf{h}_i^l = f_l(\mathbf{h}_i^{l-1}\mathbf{W}_l + \mathbf{b}_l), \quad \mathbf{h}_i^0 = \mathbf{x}_i$$

$$\mathbb{E}[\mathbf{y}_i|\mathbf{x}_i] = g^{-1}(\mathbf{h}_i^L\mathbf{w}_o + b_o)$$

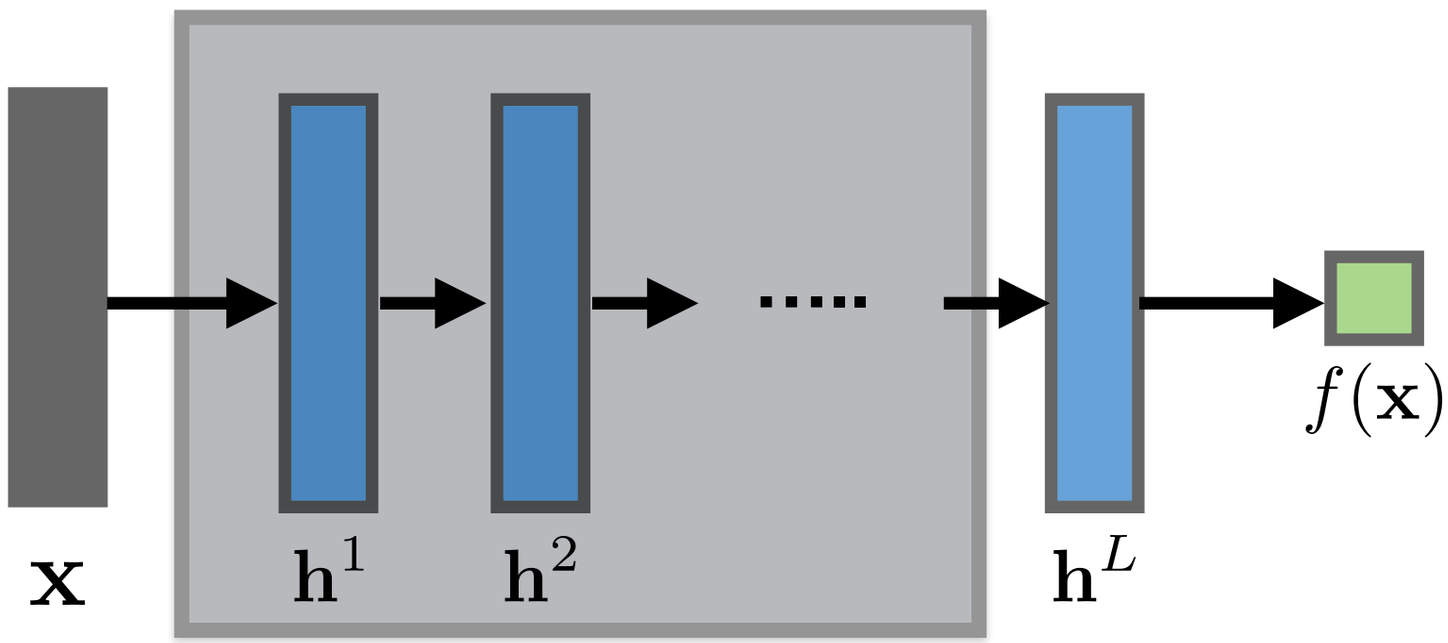


Deep Neural Networks

Adaptive Basis Function Regression: Neural networks are just performing regression with adaptive basis functions.

$$\mathbb{E}[\mathbf{y}_i|\mathbf{x}_i] = g^{-1}(\mathbf{h}_i^L \mathbf{w}_o + b_o) = g^{-1}(\mathbf{h}(\mathbf{x}_i)\mathbf{w}_o + b_o)$$

NEW REPRESENTATION
OF FEATURES COMPUTED BY NN LAYERS



Model Fitting

Parameter Estimation: All parameters are estimated via gradient ascent on the log likelihood.

LOG LIKELIHOOD

$$\mathcal{L}(\mathbf{X}, \mathbf{Y}, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1}) = \sum_{i=1}^{N} \log p(\mathbf{y}_i | \mathbf{x}_i, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1})$$

Model Fitting

Parameter Estimation: All parameters are estimated via gradient ascent on the log likelihood.

LOG LIKELIHOOD

$$\mathcal{L}(\mathbf{X}, \mathbf{Y}, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1}) = \sum_{i=1}^{N} \log p(\mathbf{y}_i | \mathbf{x}_i, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1})$$

PARAMETER UPDATE

$$\mathbf{W}_l^{t+1} = \mathbf{W}_l^t + \alpha \, \nabla_{\mathbf{W}_l} \mathcal{L}(\mathbf{X}, \mathbf{Y}, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1})$$

Model Fitting

Parameter Estimation: All parameters are estimated via gradient ascent on the log likelihood.

LOG LIKELIHOOD

$$\mathcal{L}(\mathbf{X}, \mathbf{Y}, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1}) = \sum_{i=1}^{N} \log p(\mathbf{y}_i | \mathbf{x}_i, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1})$$

PARAMETER UPDATE

$$\mathbf{W}_l^{t+1} = \mathbf{W}_l^t + \alpha \nabla_{\mathbf{W}_l} \mathcal{L}(\mathbf{X}, \mathbf{Y}, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1})$$

learning rate / step size.

Model Fitting

Parameter Estimation: All parameters are estimated via gradient ascent on the log likelihood.

LOG LIKELIHOOD

$$\mathcal{L}(\mathbf{X}, \mathbf{Y}, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1}) = \sum_{i=1}^{N} \log p(\mathbf{y}_i | \mathbf{x}_i, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1})$$

PARAMETER UPDATE

$$\mathbf{W}_l^{t+1} = \mathbf{W}_l^t + \alpha \nabla_{\mathbf{W}_l} \mathcal{L}(\mathbf{X}, \mathbf{Y}, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1})$$

learning rate / step size.

Usually estimated with a subsample (i.e. 'stochastic gradient')

$$\nabla_{\mathbf{W}_l} \mathcal{L}(\mathbf{X}, \mathbf{Y}, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1}) \approx \nabla_{\mathbf{W}_l} \sum_{k=1}^{K} \log p(\mathbf{y}_k | \mathbf{x}_k, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1})$$

for $K << N$

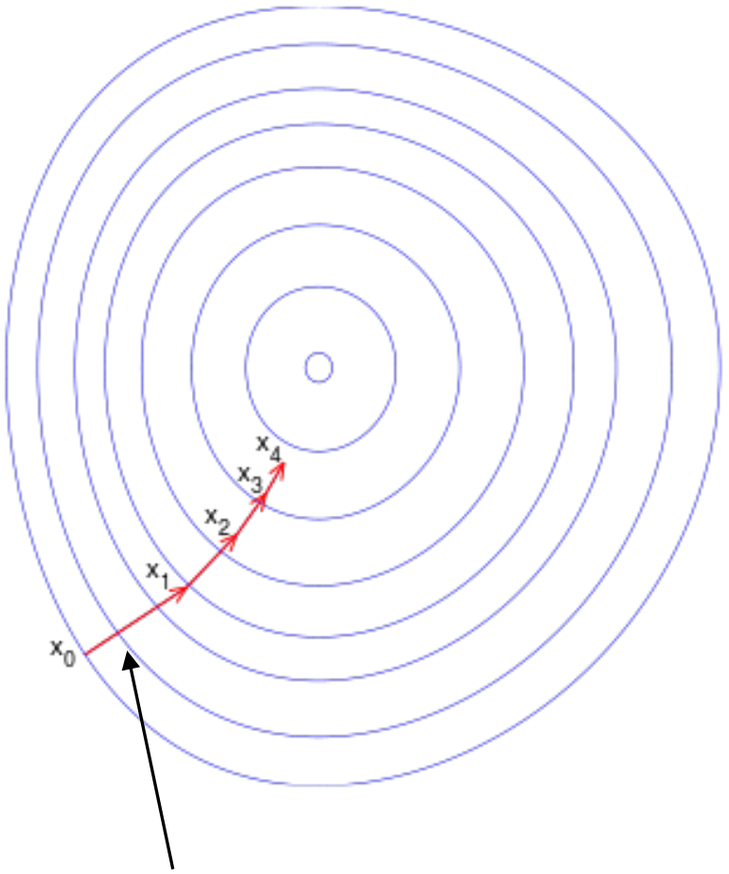
Model Fitting

PARAMETER UPDATE

$$\mathbf{W}_l^{t+1} = \mathbf{W}_l^t + \alpha \, \nabla_{\mathbf{W}_l} \mathcal{L}(\mathbf{X}, \mathbf{Y}, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1})$$

$$\nabla_{\mathbf{W}_l} \mathcal{L}(\mathbf{X}, \mathbf{Y}, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1}) \approx \nabla_{\mathbf{W}_l} \sum_{k=1}^{K} \log p(\mathbf{y}_k | \mathbf{x}_k, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1})$$

for $K << N$



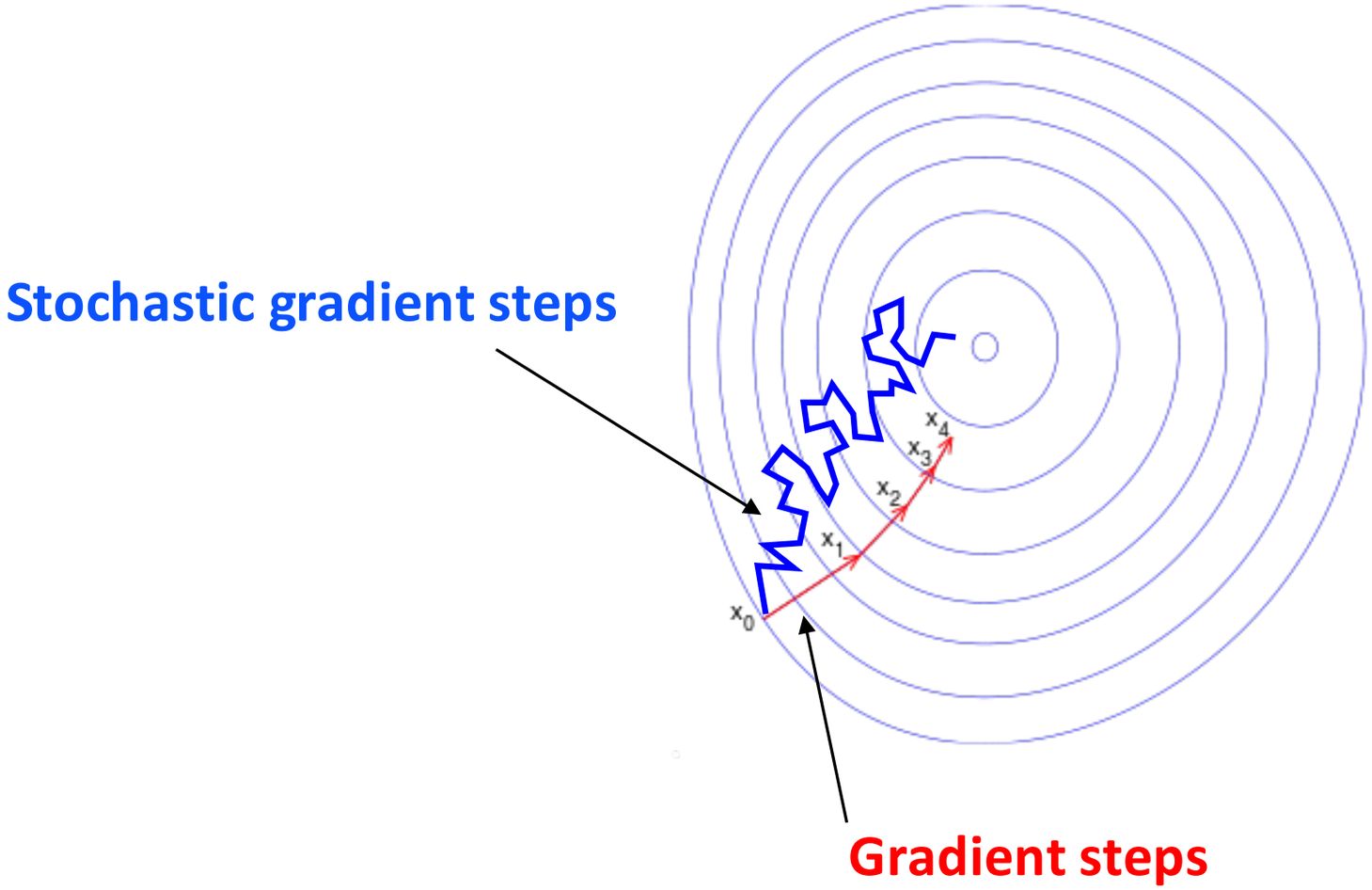
Model Fitting

PARAMETER UPDATE

$$\mathbf{W}_l^{t+1} = \mathbf{W}_l^t + \alpha \, \nabla_{\mathbf{W}_l} \mathcal{L}(\mathbf{X}, \mathbf{Y}, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1})$$

$$\nabla_{\mathbf{W}_l} \mathcal{L}(\mathbf{X}, \mathbf{Y}, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1}) \approx \nabla_{\mathbf{W}_l} \sum_{k=1}^{K} \log p(\mathbf{y}_k | \mathbf{x}_k, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1})$$

for $K << N$



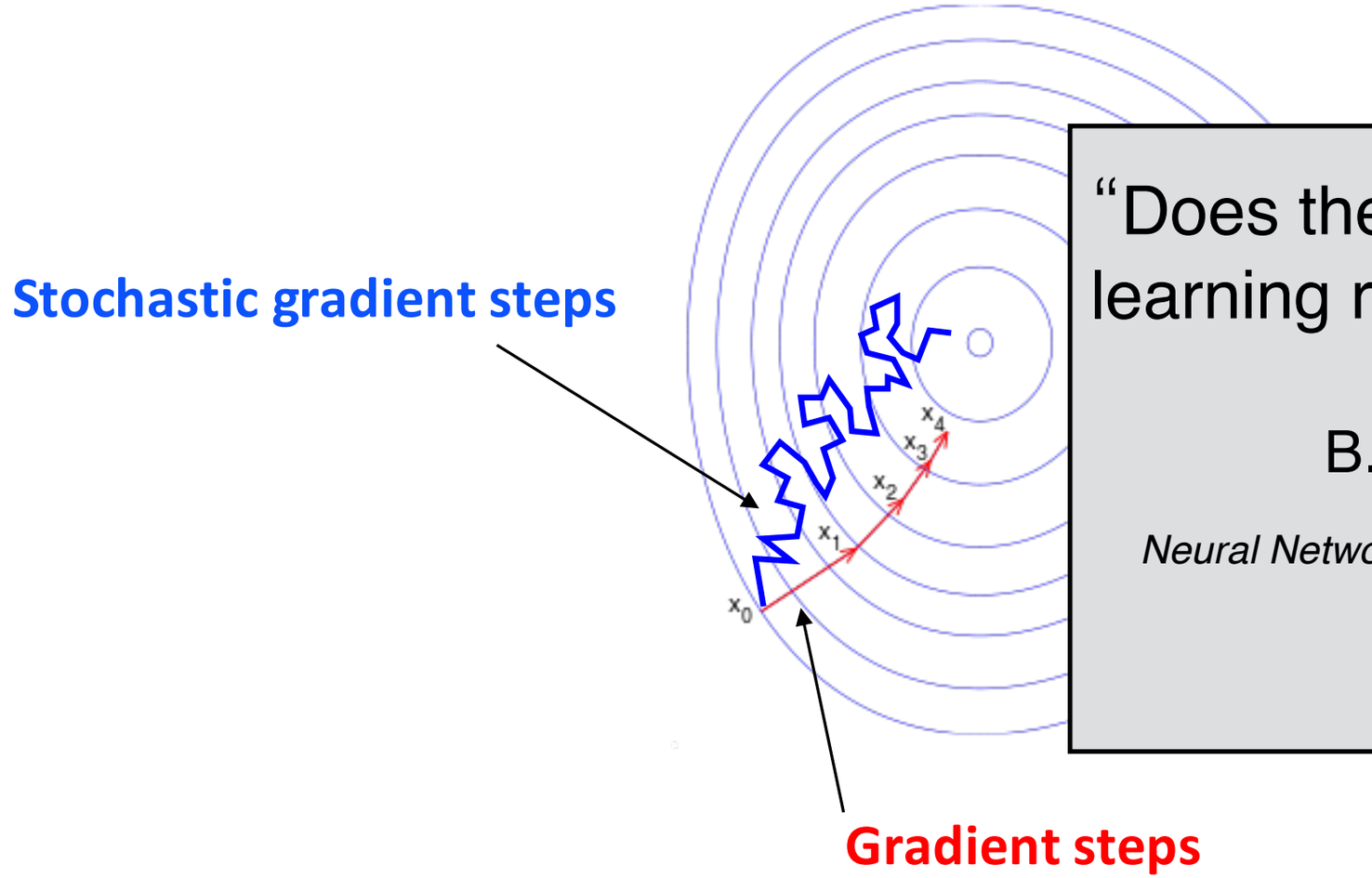
Model Fitting

PARAMETER UPDATE

$$\mathbf{W}_l^{t+1} = \mathbf{W}_l^t + \alpha \, \nabla_{\mathbf{W}_l} \mathcal{L}(\mathbf{X}, \mathbf{Y}, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1})$$

$$\nabla_{\mathbf{W}_l} \mathcal{L}(\mathbf{X}, \mathbf{Y}, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1}) \approx \nabla_{\mathbf{W}_l} \sum_{k=1}^{K} \log p(\mathbf{y}_k | \mathbf{x}_k, \{\mathbf{W}_l\}_1^{L+1}, \{\mathbf{b}_l\}_1^{L+1})$$

for $K << N$



"Does the error-backpropagation learning rule…still have a place?"

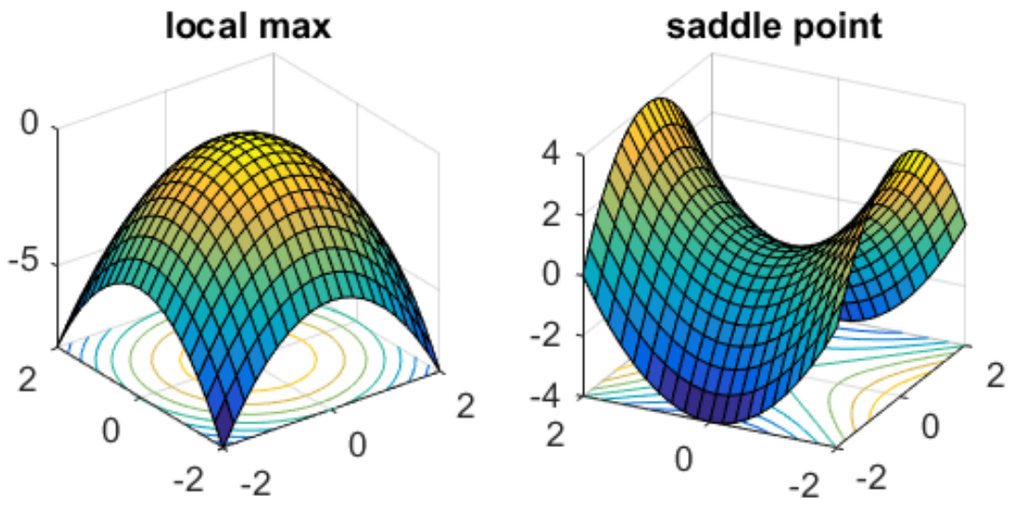
B. Cheng and D. M. Titterington

Neural Networks: A Review from a Statistical Perspective

(1994)

Optimization Landscape

- Previously, optimization of deep NNs was thought to be hopelessly non-convex. However, many of the critical points are not maxima but rather ***saddle points***. [Dauphin et al., 2014]

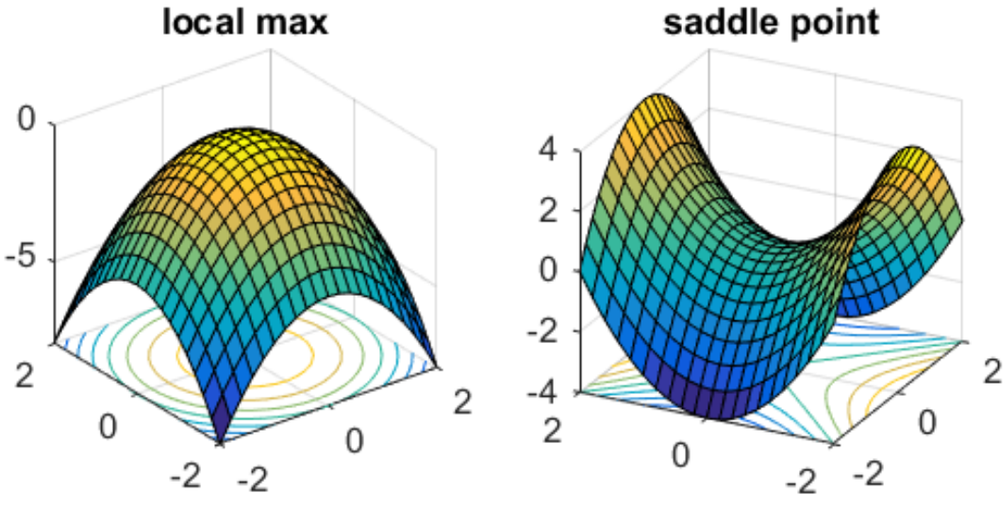


http://www.offconvex.org/2016/03/22/

Optimization Landscape

- Previously, optimization of deep NNs was thought to be hopelessly non-convex. However, many of the critical points are not maxima but rather ***saddle points***. [Dauphin et al., 2014]

- **Intuition:** It's hard to build a high-dimensional max / min because the surface must be going in the same direction in all dimensions.



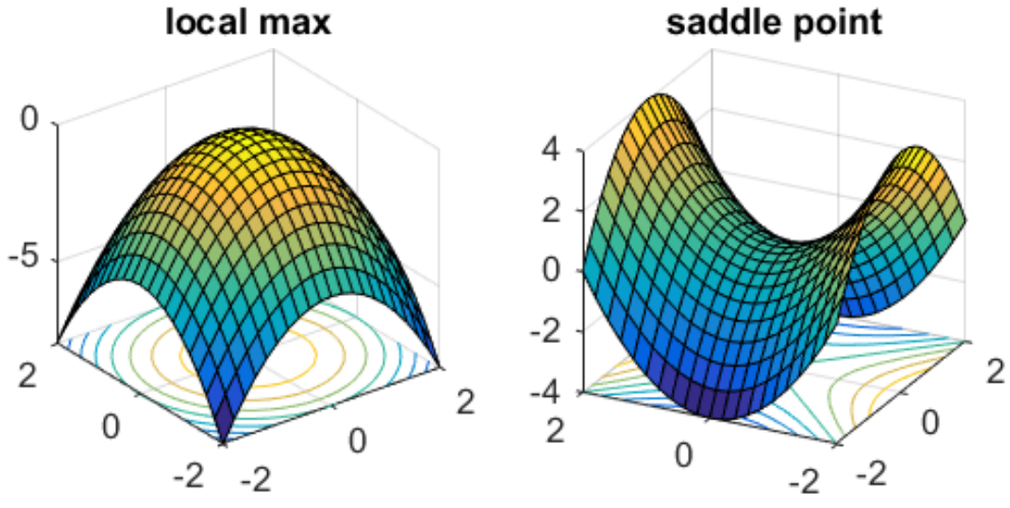
http://www.offconvex.org/2016/03/22/

Optimization Landscape

- Previously, optimization of deep NNs was thought to be hopelessly non-convex. However, many of the critical points are not maxima but rather ***saddle points***. [Dauphin et al., 2014]

- **Intuition:** It's hard to build a high-dimensional max / min because the surface must be going in the same direction in all dimensions.

- **Stochastic Gradient:** [Hardt et al., 2016] (+ others) suggest the noise from stochastic gradient regularizes, improving model generalization.



http://www.offconvex.org/2016/03/22/

Model Fitting

- **Backpropagation**: The *backpropagation* algorithm [Rumelhart et al., 1986] is just an efficient way to compute the chain rule through the network.

$$\frac{\partial}{\partial\ \mathbf{W}_l}\mathcal{L} = \frac{\partial}{\partial\ \mathbf{h}_i^L}\log p(\mathbf{y}_i|\mathbf{h}_i^L, \mathbf{w}_o, b_o)\ \frac{\partial\ \mathbf{h}_i^L}{\partial\ \mathbf{h}_i^{L-1}}\cdots\frac{\partial\ \mathbf{h}_i^{l+1}}{\partial\ \mathbf{h}_i^l}\ \frac{\partial}{\partial\ \mathbf{W}_l}f_l(\mathbf{h}_i^{l-1}\mathbf{W}_l + \mathbf{b}_l)$$

Model Fitting

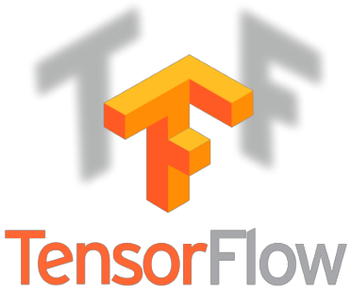
- **Backpropagation**: The *backpropagation* algorithm [Rumelhart et al., 1986] is just an efficient way to compute the chain rule through the network.

$$\frac{\partial}{\partial\ \mathbf{W}_l}\mathcal{L} = \frac{\partial}{\partial\ \mathbf{h}_i^L}\log p(\mathbf{y}_i|\mathbf{h}_i^L, \mathbf{w}_o, b_o)\ \frac{\partial\ \mathbf{h}_i^L}{\partial\ \mathbf{h}_i^{L-1}} \cdots \frac{\partial\ \mathbf{h}_i^{l+1}}{\partial\ \mathbf{h}_i^l}\ \frac{\partial}{\partial\ \mathbf{W}_l} f_l(\mathbf{h}_i^{l-1}\mathbf{W}_l + \mathbf{b}_l)$$

- **Automatic Differentiation**: Various 'autodiff' software libraries can compute these gradients for you, just needing definition of the forward model.

RONAN COLLOBERT (2000)

UNIVERSITY OF MONTREAL (2011)



GOOGLE BRAIN (2009/2015)

FACEBOOK AI (2016)

Model Fitting

- **Backpropagation**: The *backpropagation* algorithm [Rumelhart et al., 1986] is just an efficient way to compute the chain rule through the network.

$$\frac{\partial}{\partial\ \mathbf{W}_l}\mathcal{L} = \frac{\partial}{\partial\ \mathbf{h}_i^L}\log p(\mathbf{y}_i|\mathbf{h}_i^L, \mathbf{w}_o, b_o)\ \frac{\partial\ \mathbf{h}_i^L}{\partial\ \mathbf{h}_i^{L-1}} \cdots \frac{\partial\ \mathbf{h}_i^{l+1}}{\partial\ \mathbf{h}_i^l}\ \frac{\partial}{\partial\ \mathbf{W}_l} f_l(\mathbf{h}_i^{l-1}\mathbf{W}_l + \mathbf{b}_l)$$

- **Automatic Differentiation**: Various 'autodiff' software libraries can compute these gradients for you, just needing definition of the forward model.

MOST POPULAR TODAY

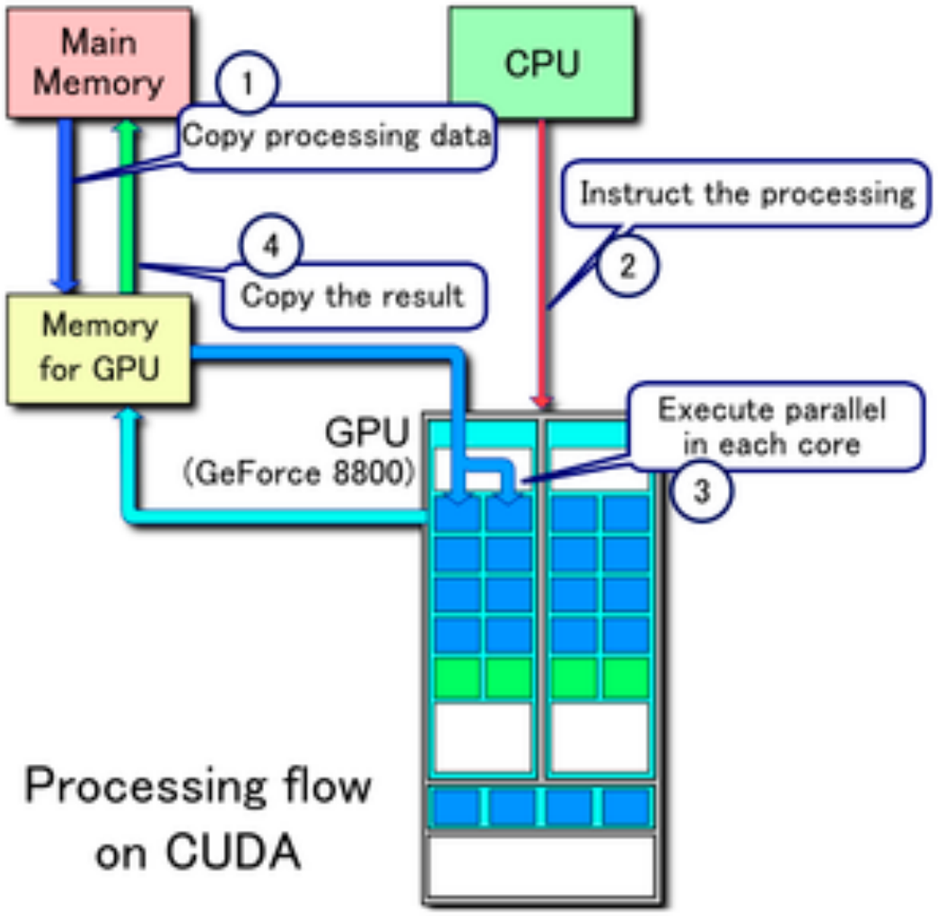
torch — RONAN COLLOBERT (2000)

theano — UNIVERSITY OF MONTREAL (2011)

TensorFlow — GOOGLE BRAIN (2009/2015)

PYTORCH — FACEBOOK AI (2016)

Graphics Processing Unit (GPU)



- GPU cards have been essential to scaling deep learning models.
- Perform fast matrix multiplications via parallel, but low precision, computation

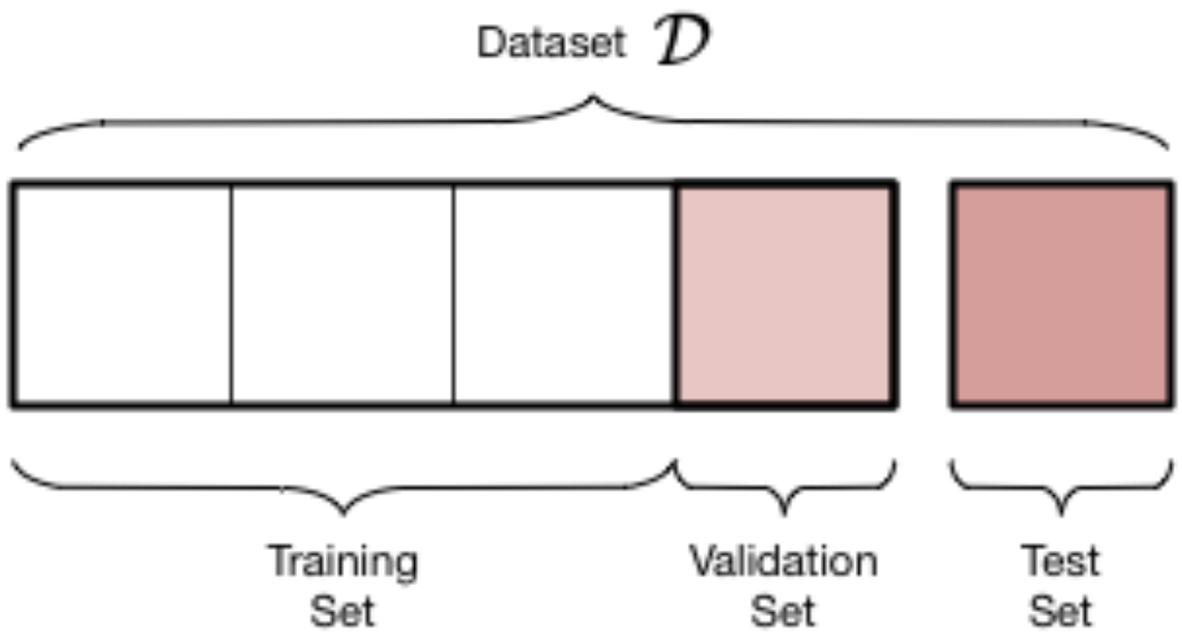
Hyperparameter Tuning

- [ ] Tuning the network architecture (e.g. number of hidden units, layers) and optimization parameters (e.g. step size) is needed for best performance.

Hyperparameter Tuning

- [ ] Tuning the network architecture (e.g. number of hidden units, layers) and optimization parameters (e.g. step size) is needed for best performance.

- [ ] Usually, a **random or grid search** over hyper parameters is performed, and **performance on a validation set** is used to make the final selection.

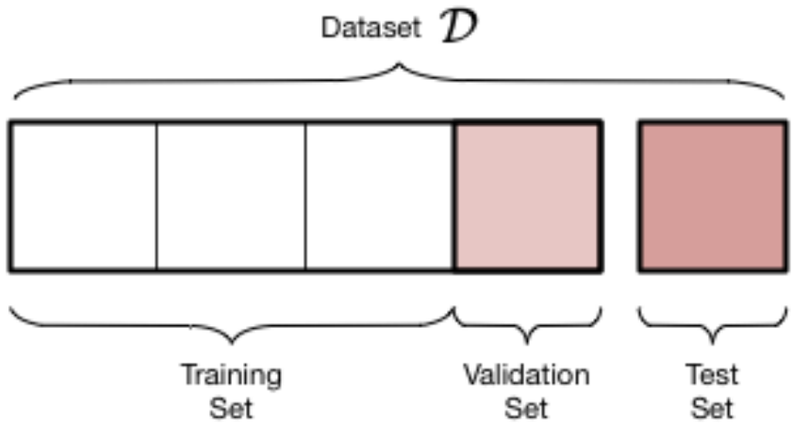


Hyperparameter Tuning

- Tuning the network architecture (e.g. number of hidden units, layers) and optimization parameters (e.g. step size) is needed for best performance.

- Usually, a **random or grid search** over hyper parameters is performed, and **performance on a validation set** is used to make the final selection.

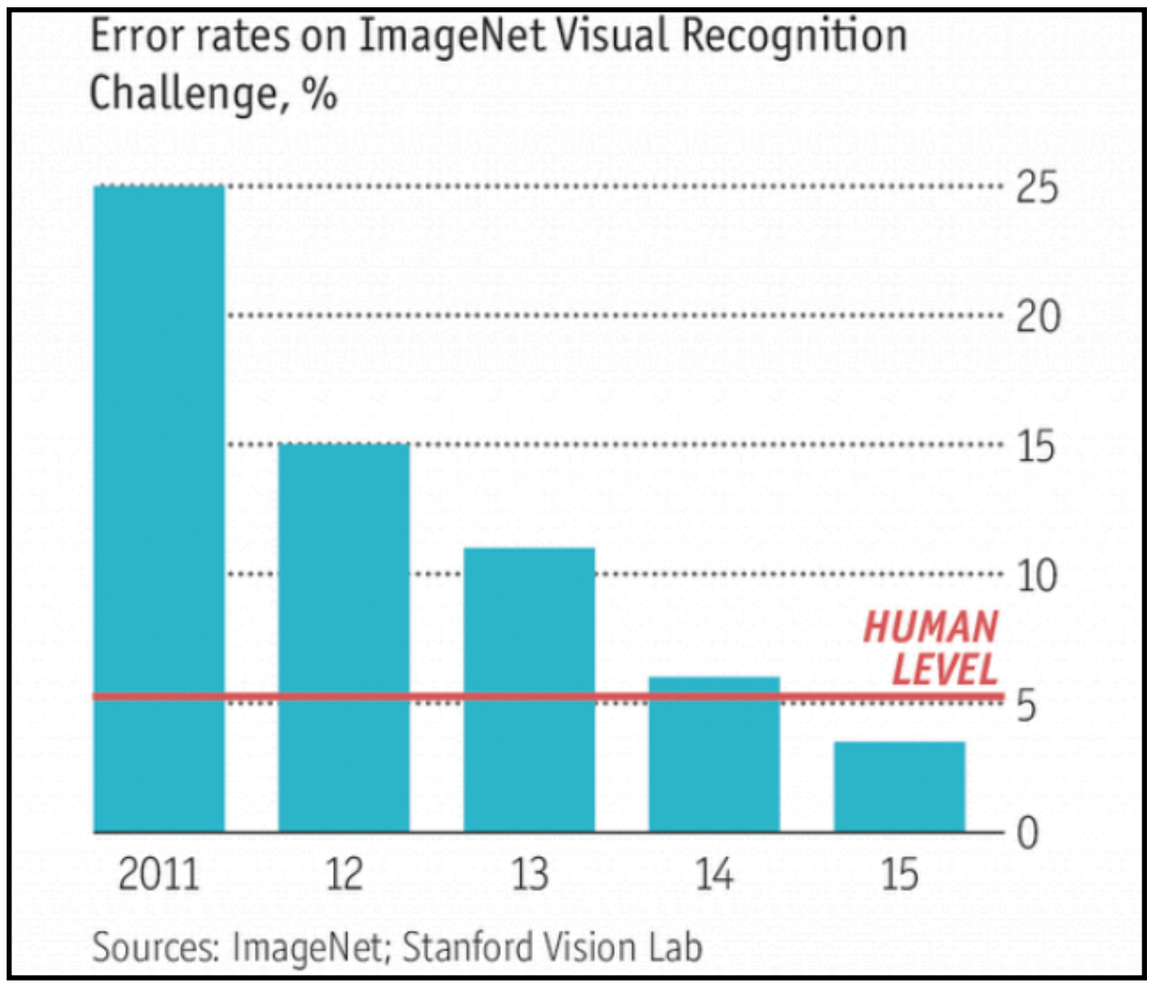
- Improving the model selection process is an on-going area of research. Bayesian optimization, bandit algorithms, and reinforcement learning methods have been proposed.



Dropout Regularization

Deep Learning Results

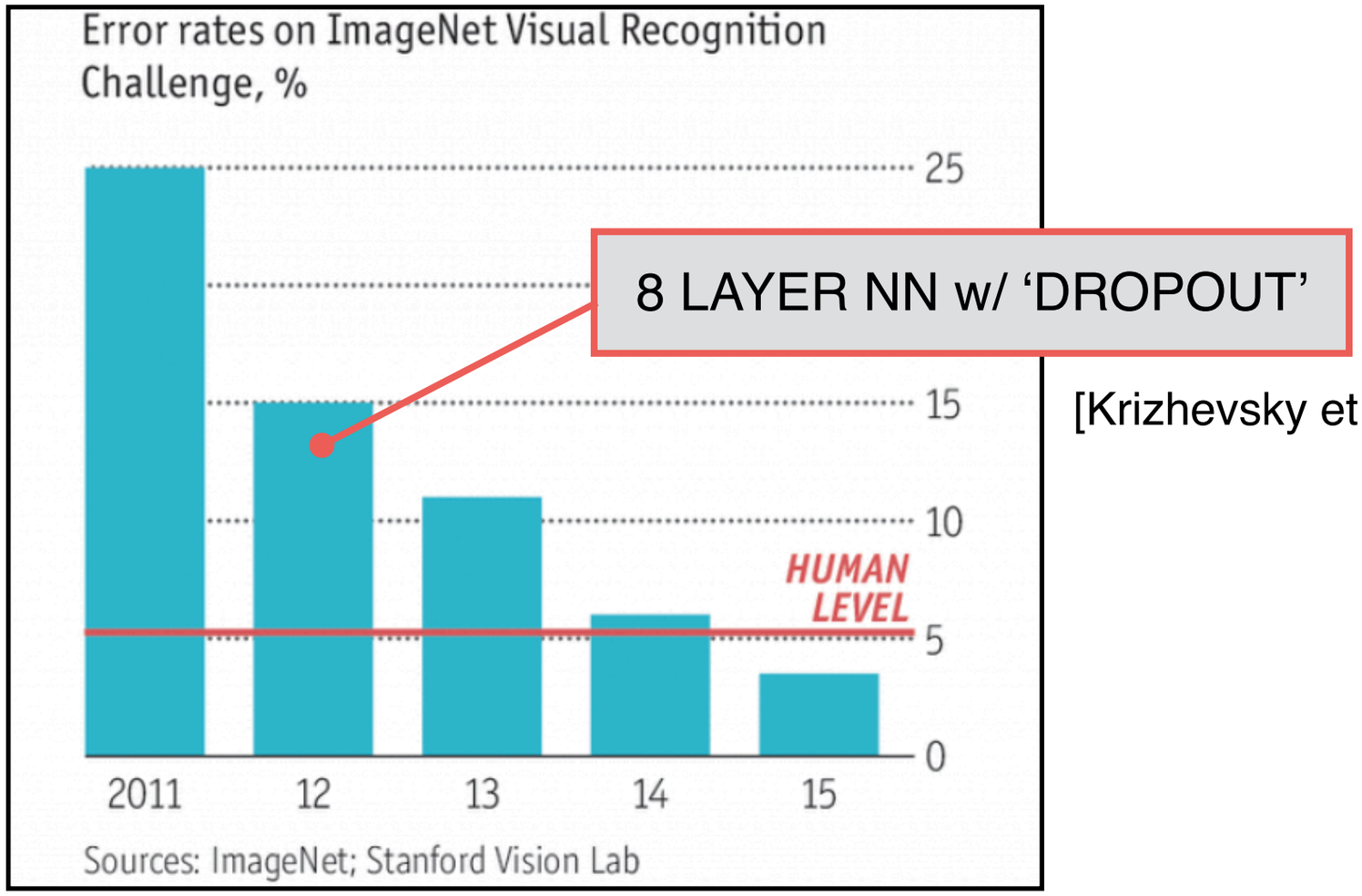
Computer Vision: Results on ImageNet object classification dataset.



https://www.economist.com/news/special-report/21700756-artificial-intelligence-boom-based-old-idea-modern-twist-not

Deep Learning Results

Computer Vision: Results on ImageNet object classification dataset.



[Krizhevsky et al., 2012]

https://www.economist.com/news/special-report/21700756-artificial-intelligence-boom-based-old-idea-modern-twist-not

Dropout Regularization

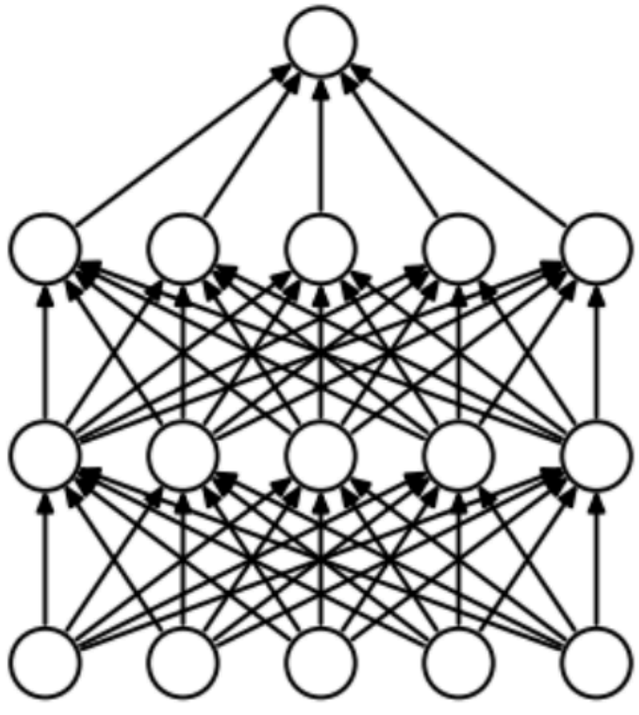
Improving neural networks by preventing co-adaptation of feature detectors (2012)

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada

Dropout Regularization

Improving neural networks by preventing co-adaptation of feature detectors (2012)

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada



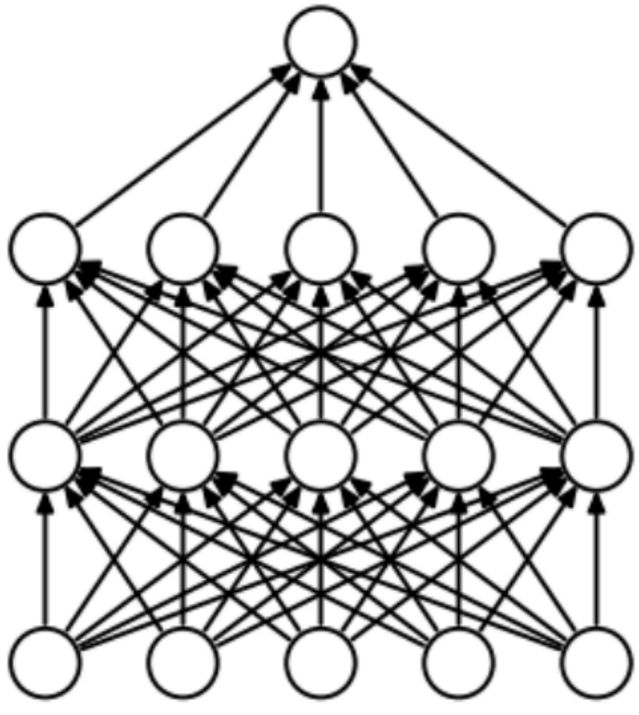
(a) Standard Neural Net

(b) After applying dropout.

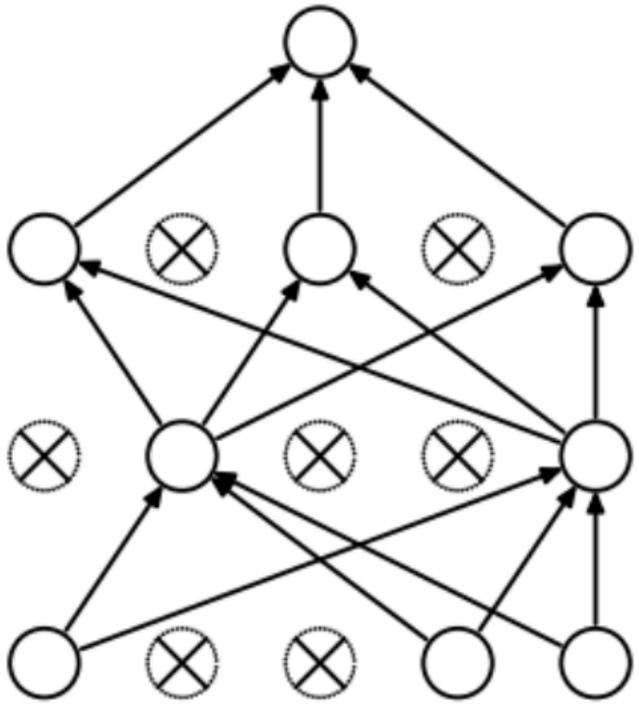
Dropout Regularization

Improving neural networks by preventing co-adaptation of feature detectors (2012)

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada



(a) Standard Neural Net



(b) After applying dropout.

Dropout Regularization

Improving neural networks by preventing co-adaptation of feature detectors (2012)

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada

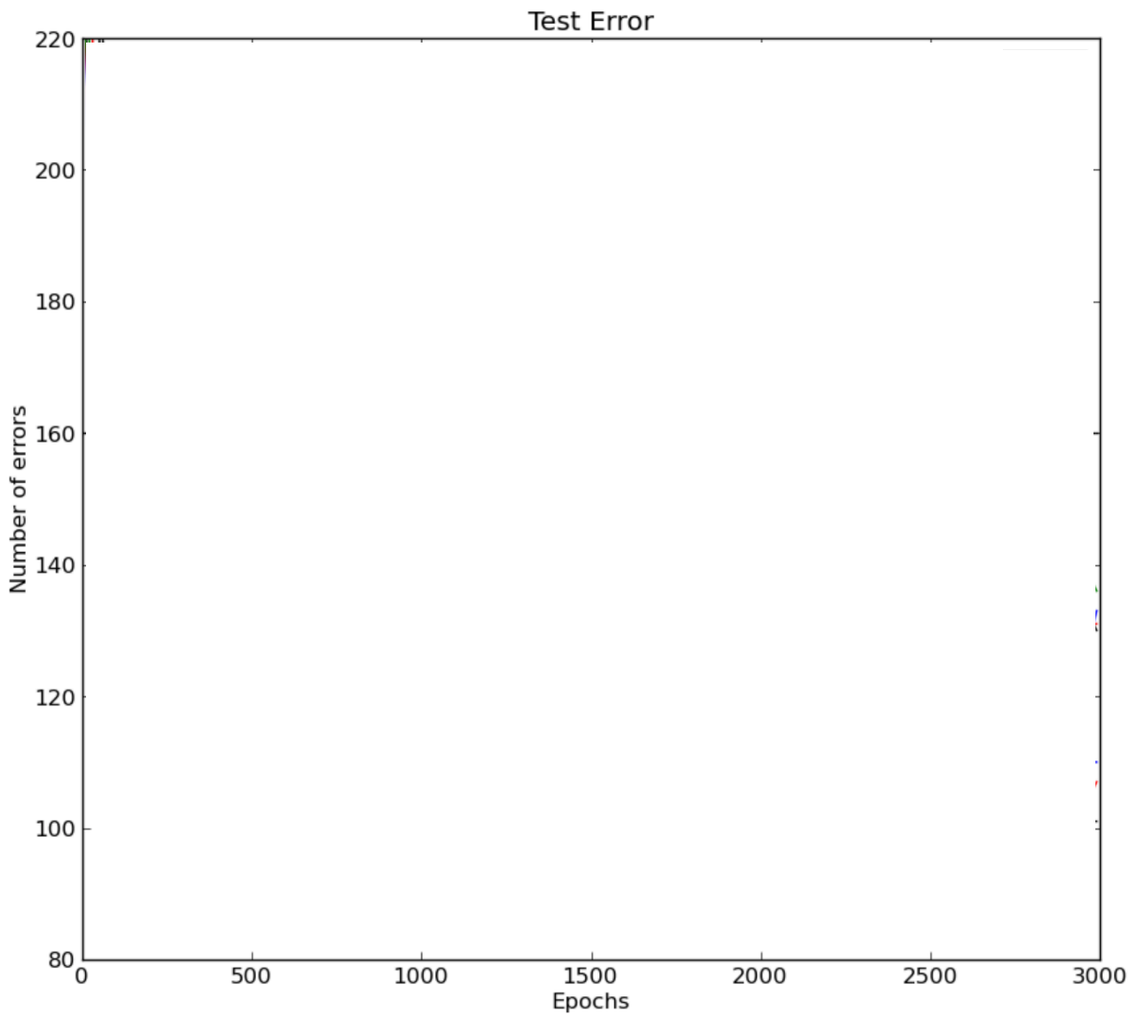


Image Classification (MNIST digits)

Dropout Regularization

Improving neural networks by preventing co-adaptation of feature detectors (2012)

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada

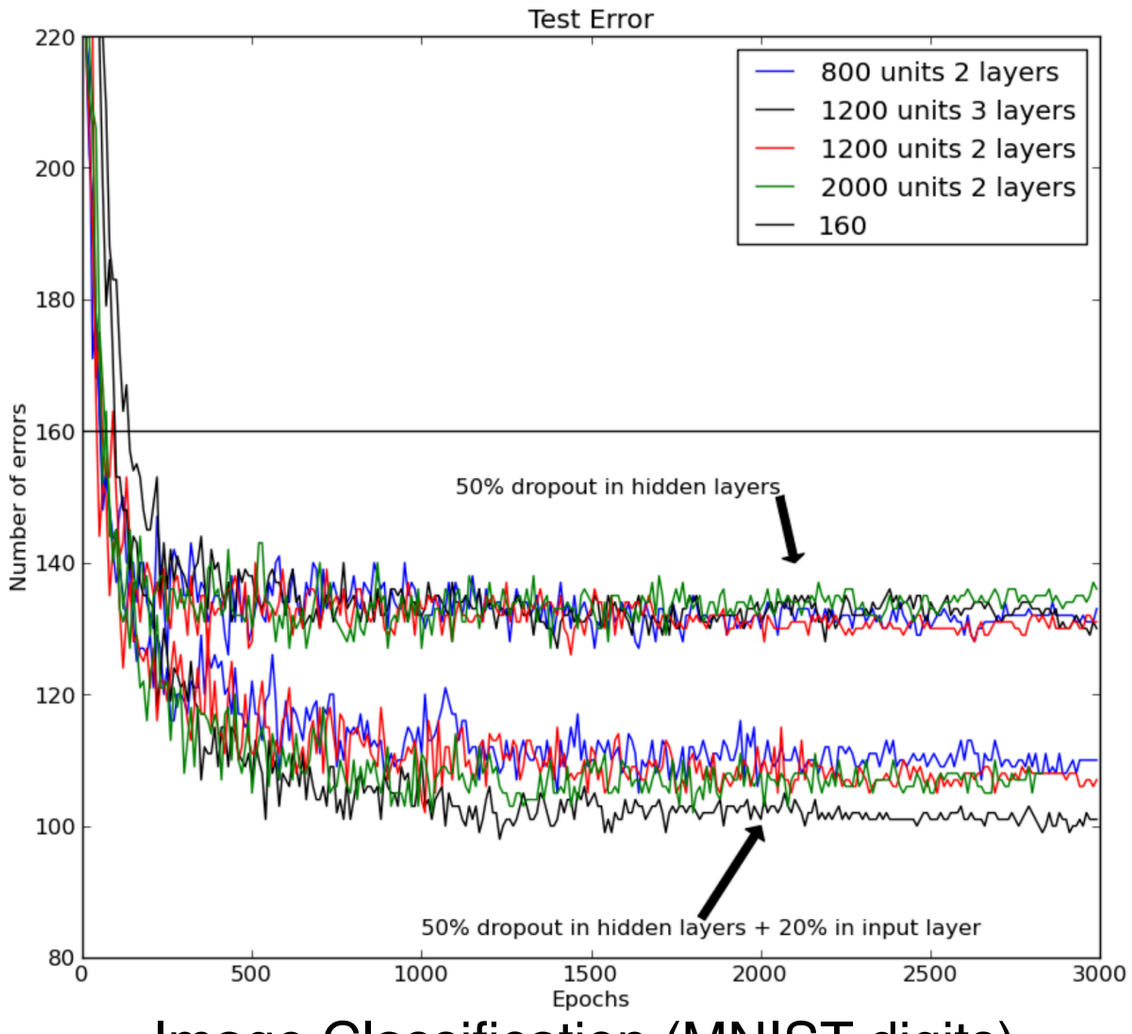


Image Classification (MNIST digits)

Dropout Regularization

Improving neural networks by preventing co-adaptation of feature detectors (2012)

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada

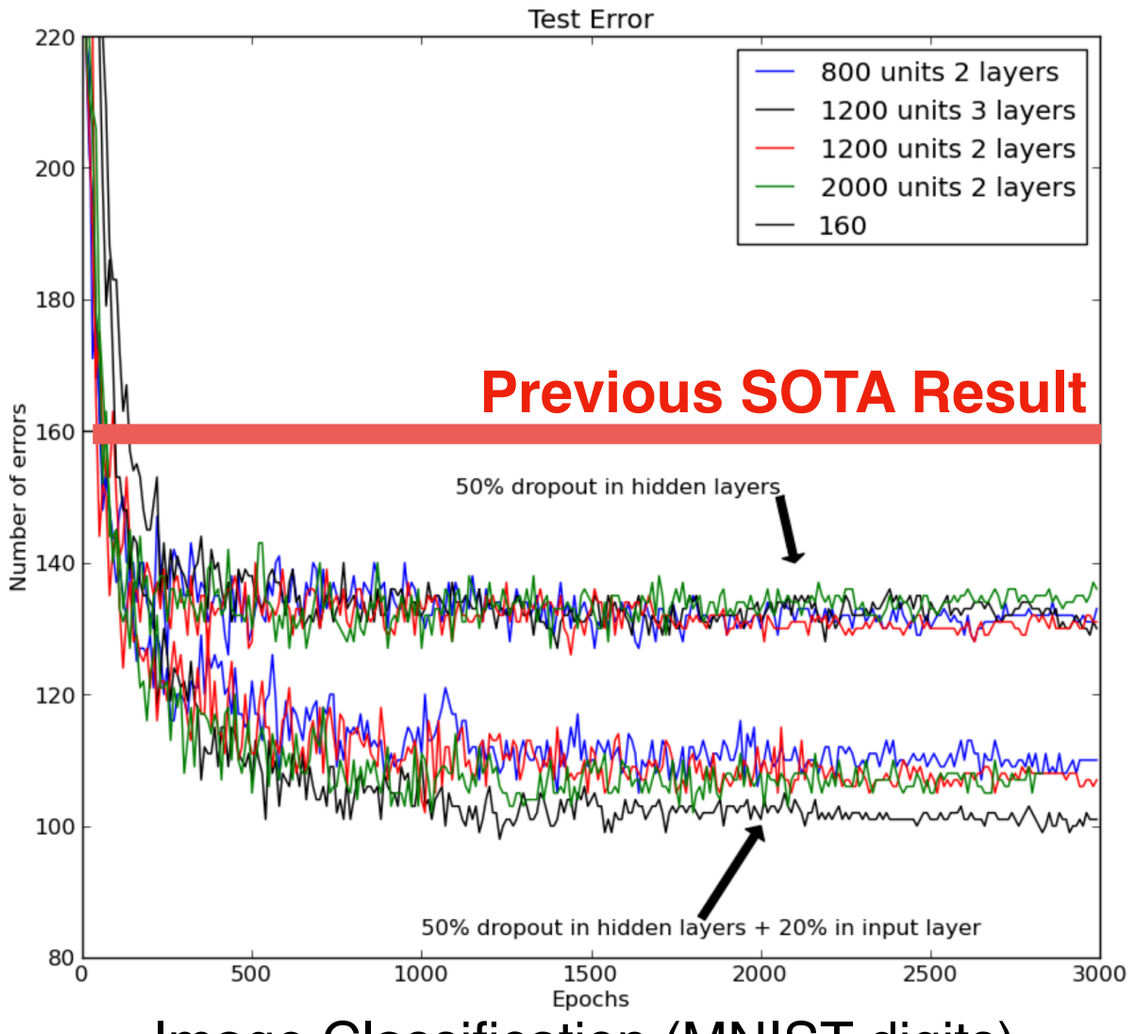


Image Classification (MNIST digits)

Dropout Regularization

Improving neural networks by preventing co-adaptation of feature detectors (2012)

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada

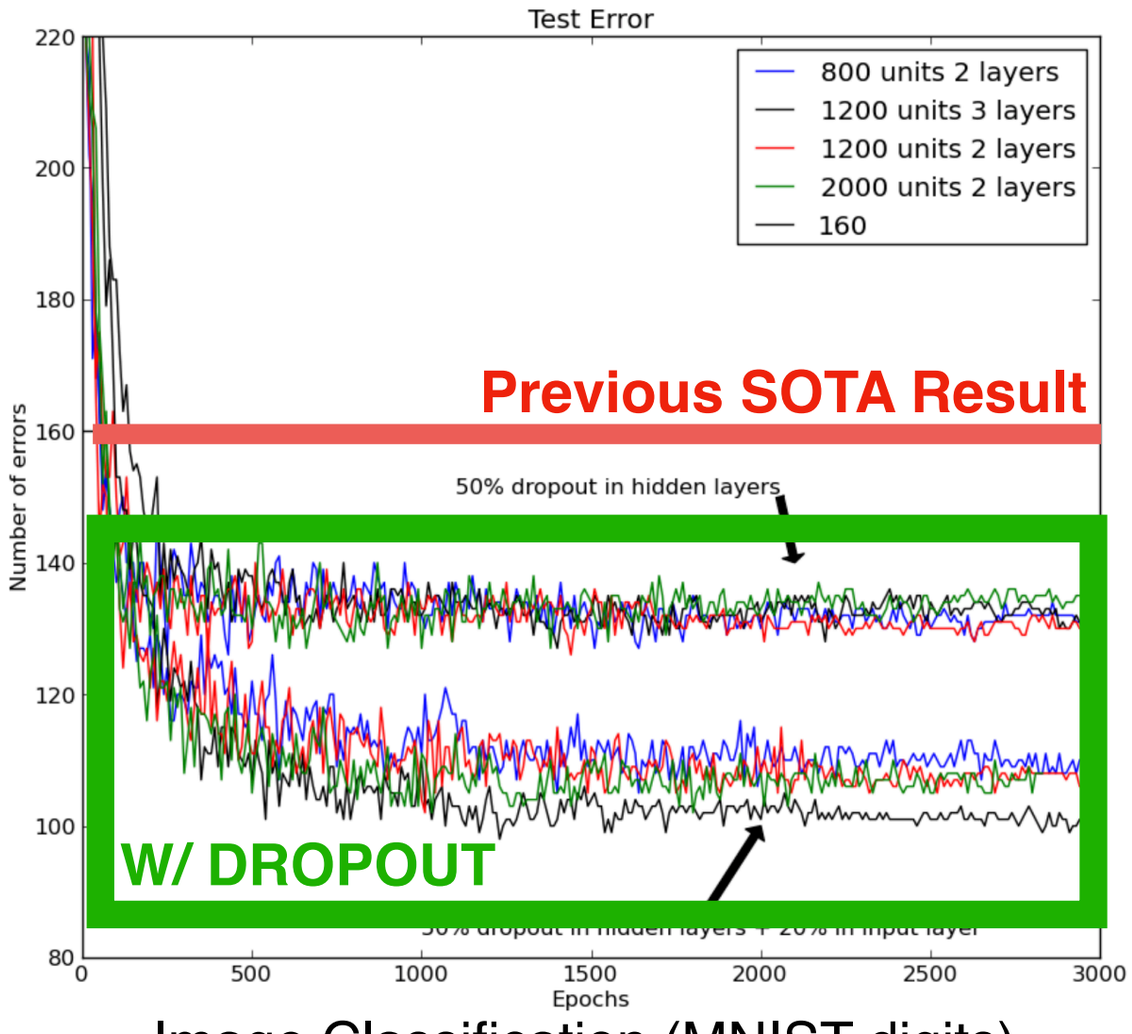


Image Classification (MNIST digits)

Dropout Regularization

Improving neural networks by preventing co-adaptation of feature detectors (2012)

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada

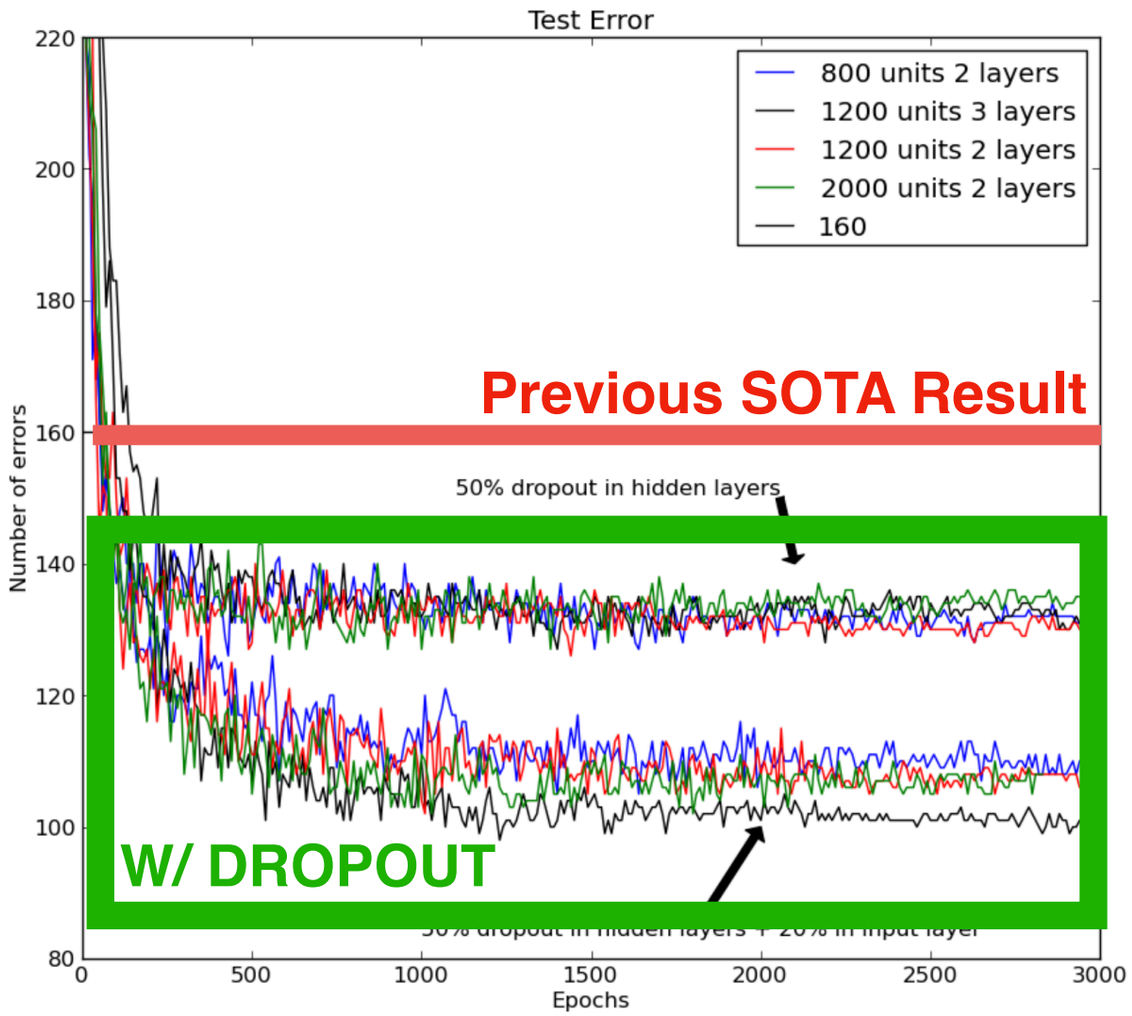
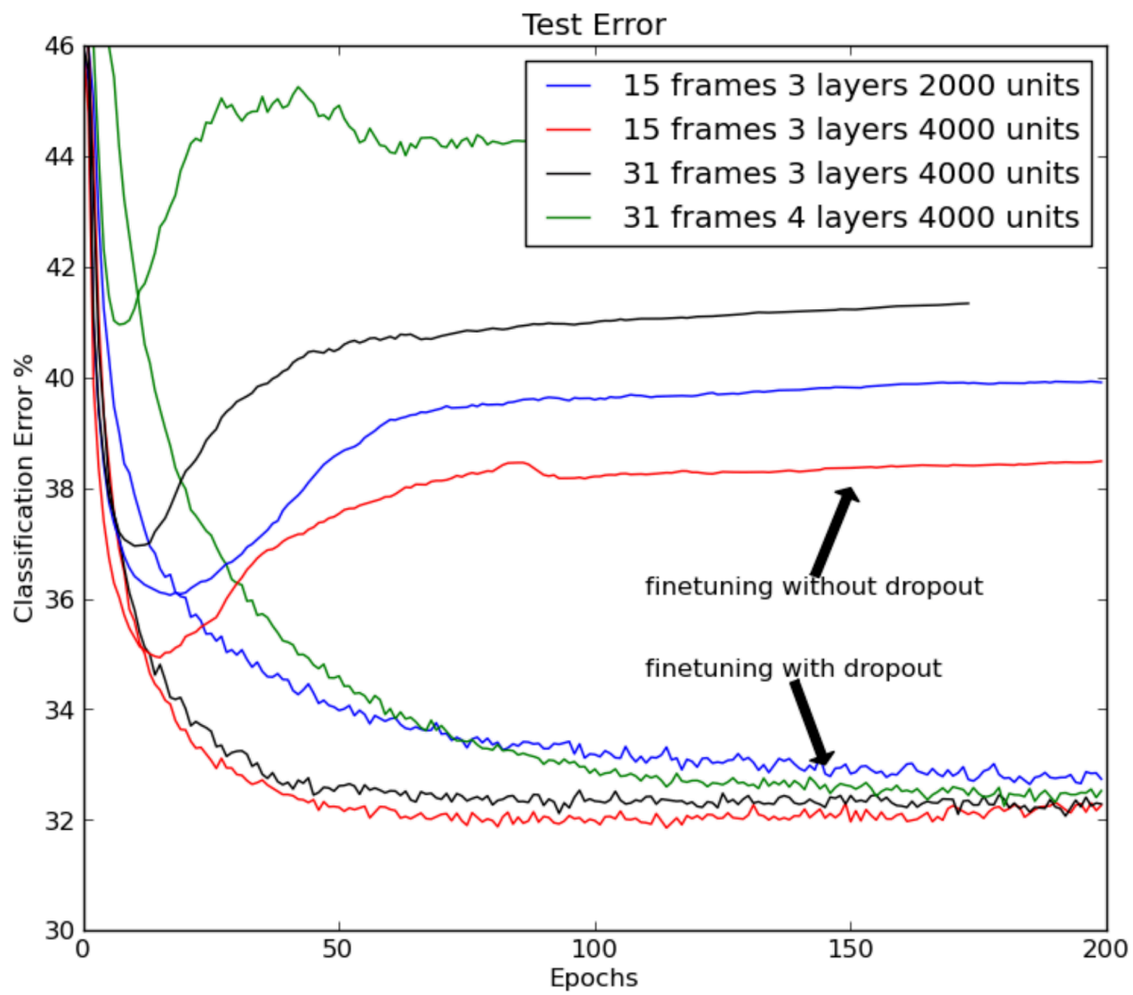


Image Classification (MNIST digits)



Phoneme Classification (TIMIT speeches)

Dropout Regularization

Improving neural networks by preventing co-adaptation of feature detectors (2012)

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada

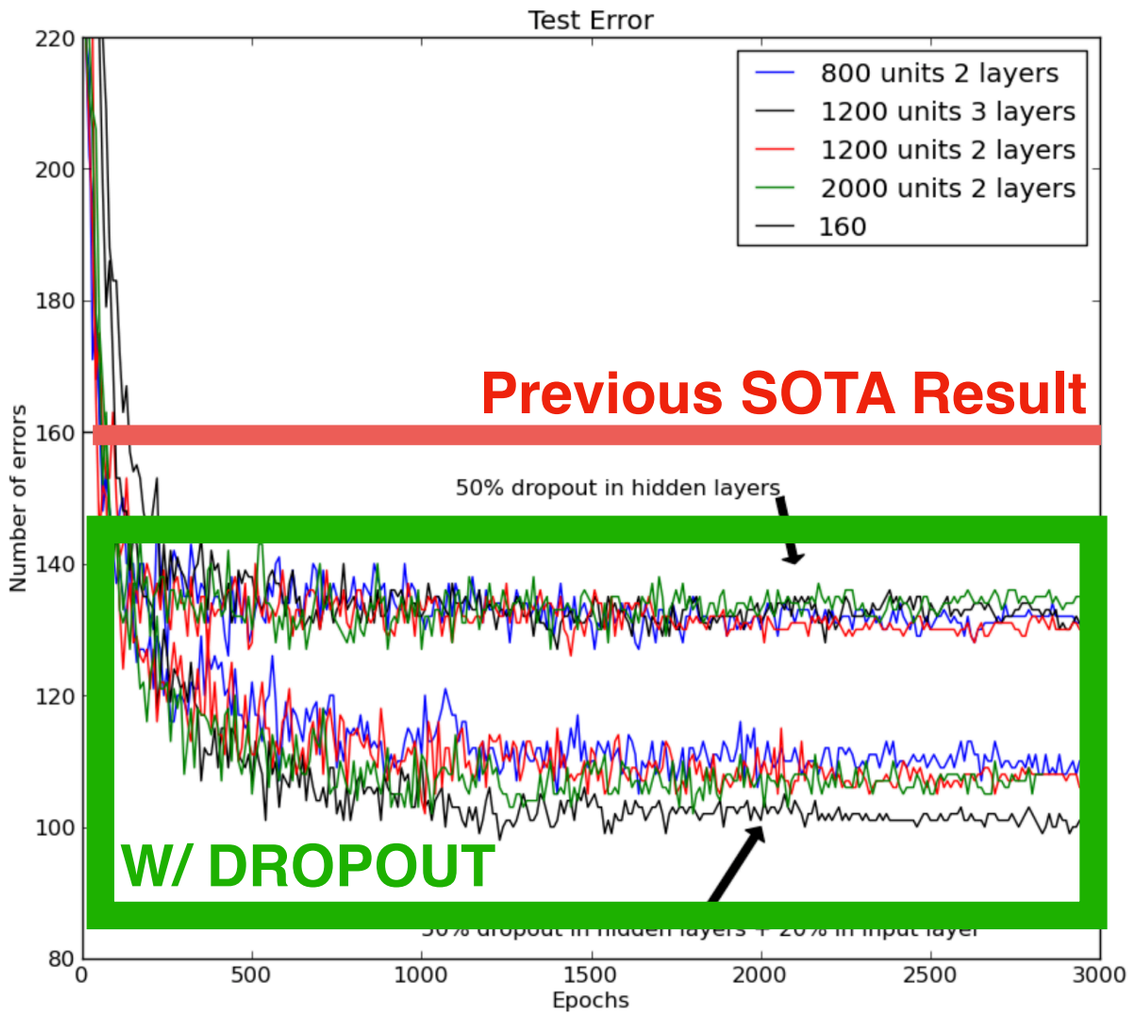
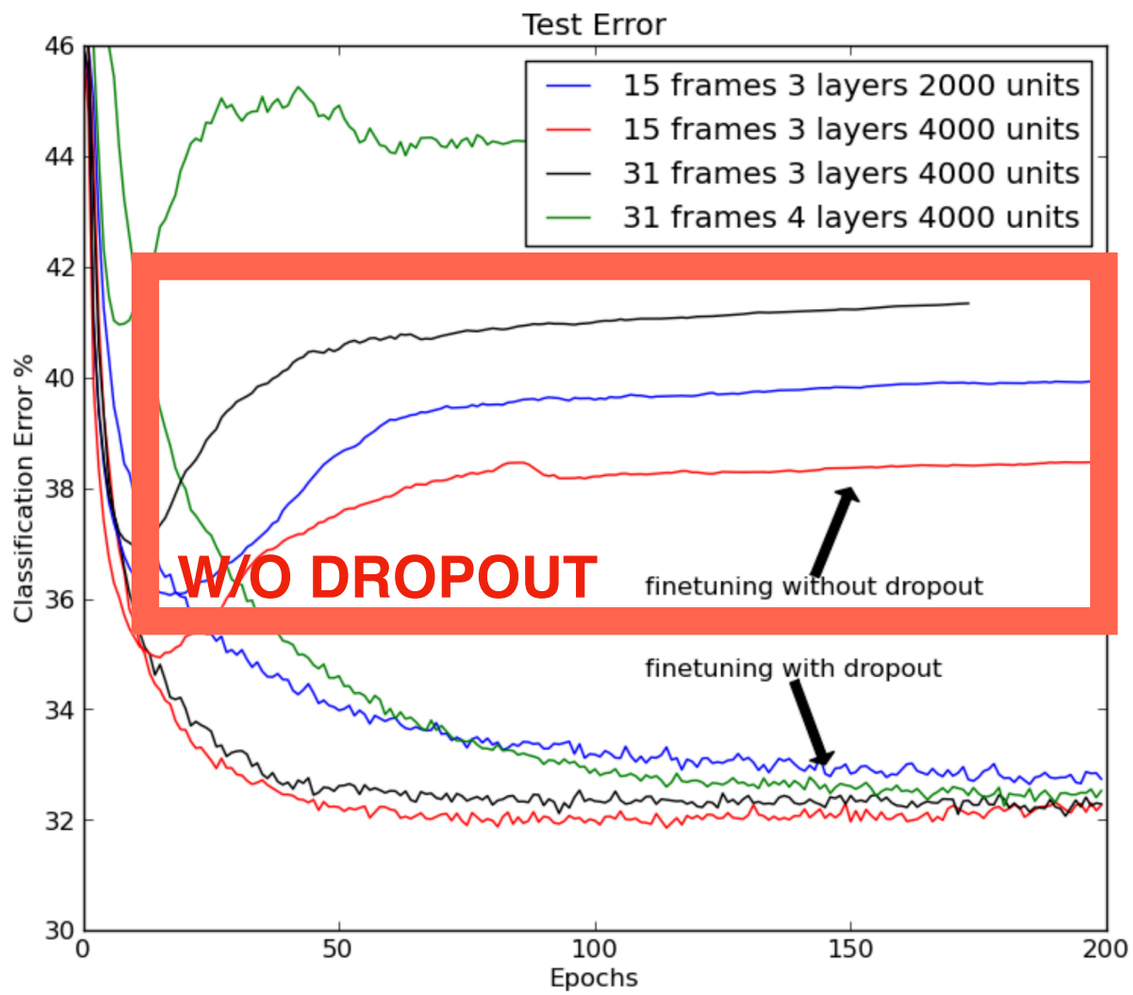


Image Classification (MNIST digits)



Phoneme Classification (TIMIT speeches)

Dropout Regularization

Improving neural networks by preventing co-adaptation of feature detectors (2012)

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada

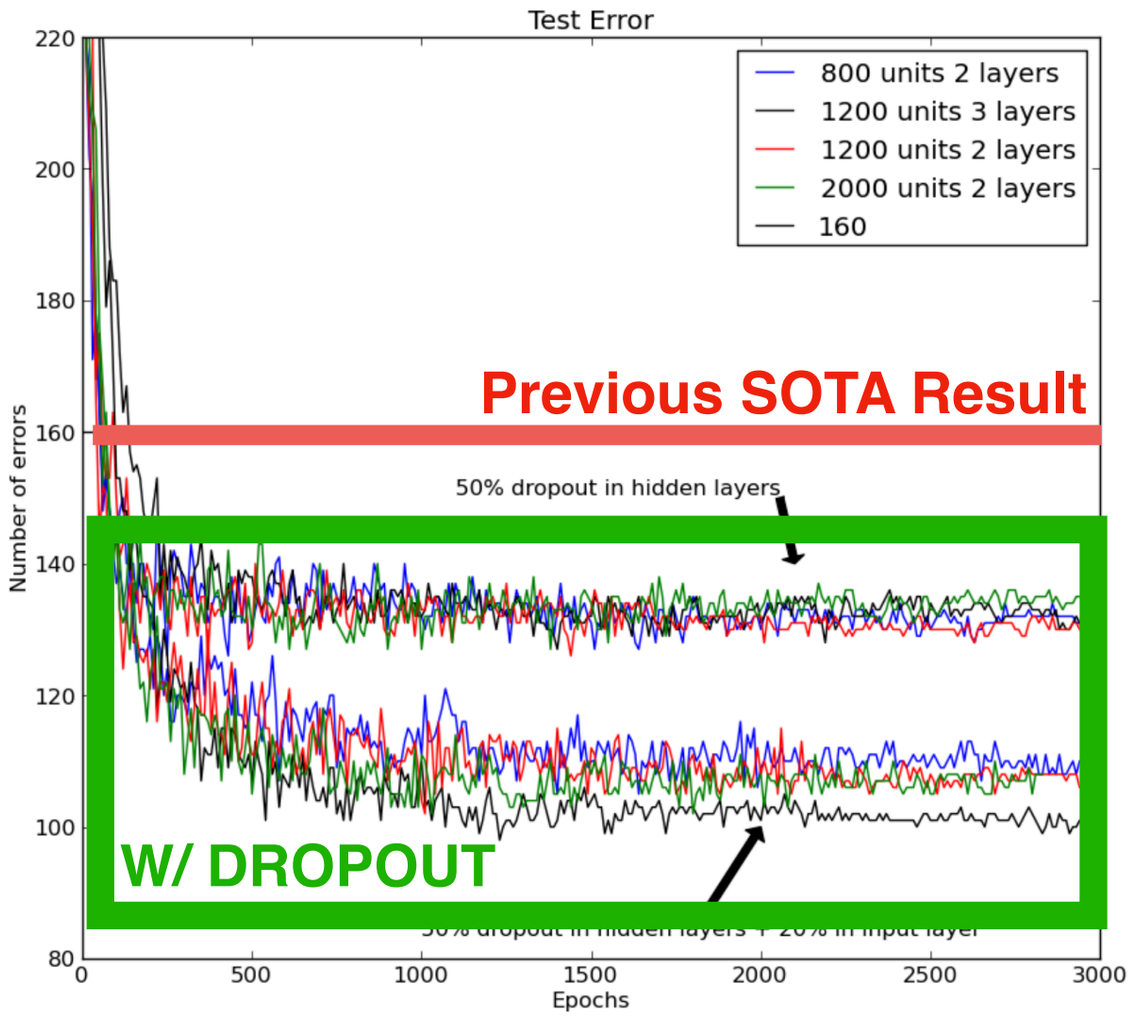
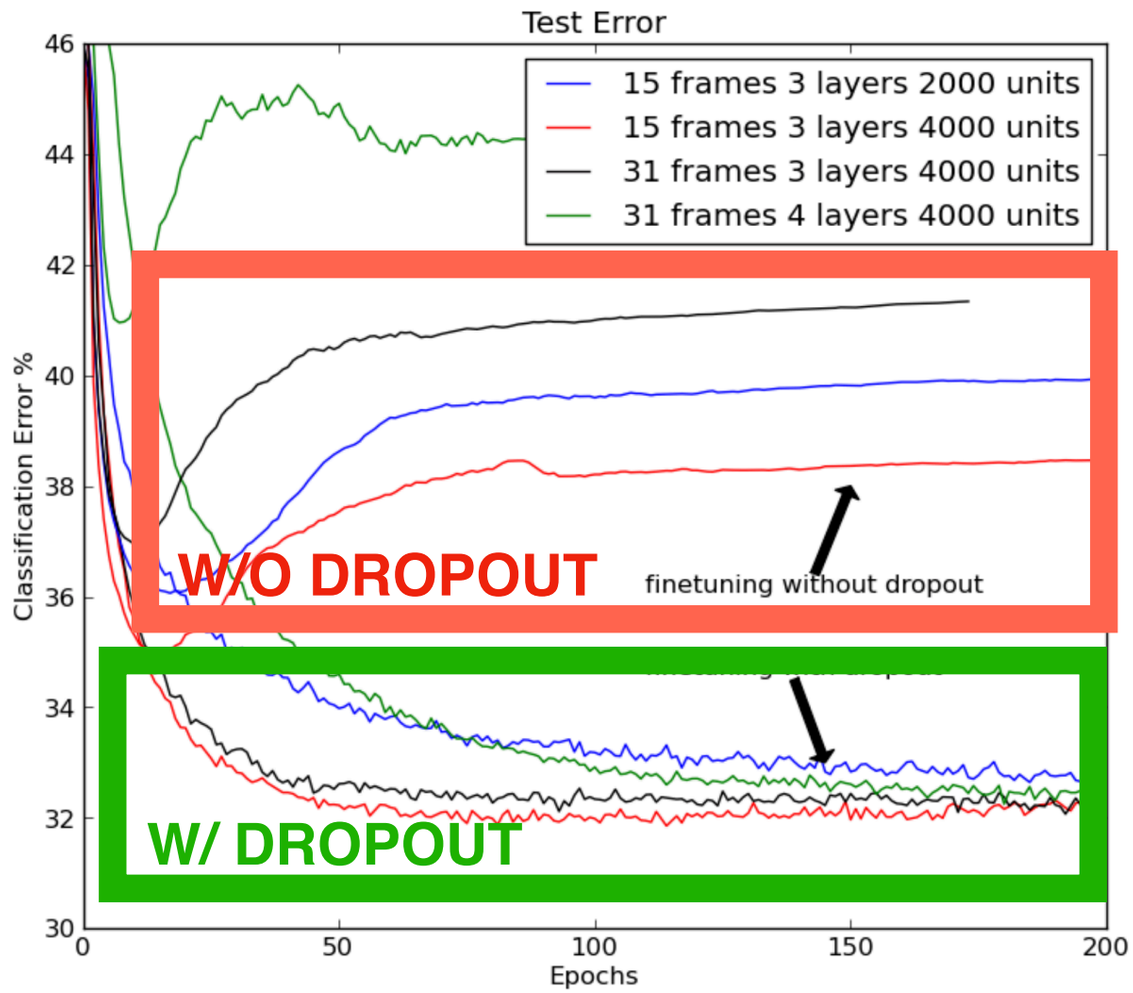


Image Classification (MNIST digits)

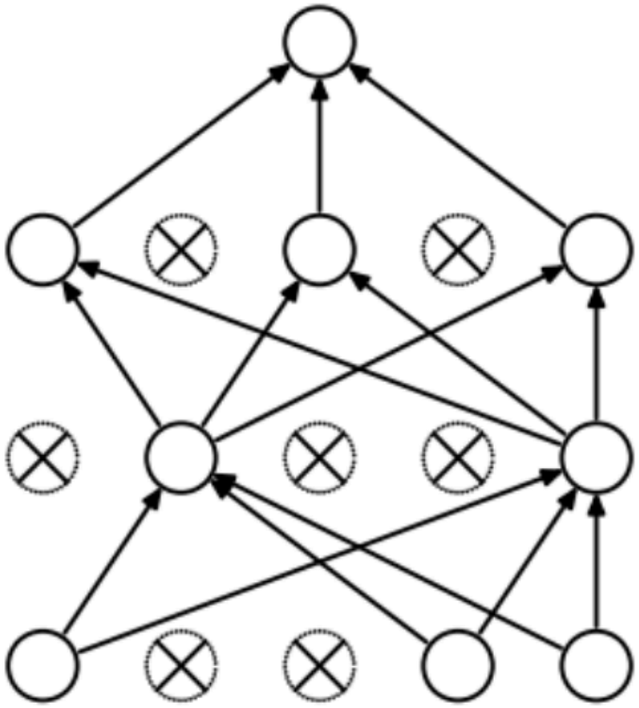


Phoneme Classification (TIMIT speeches)

Dropout Regularization

Improving neural networks by preventing co-adaptation of feature detectors (2012)

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada

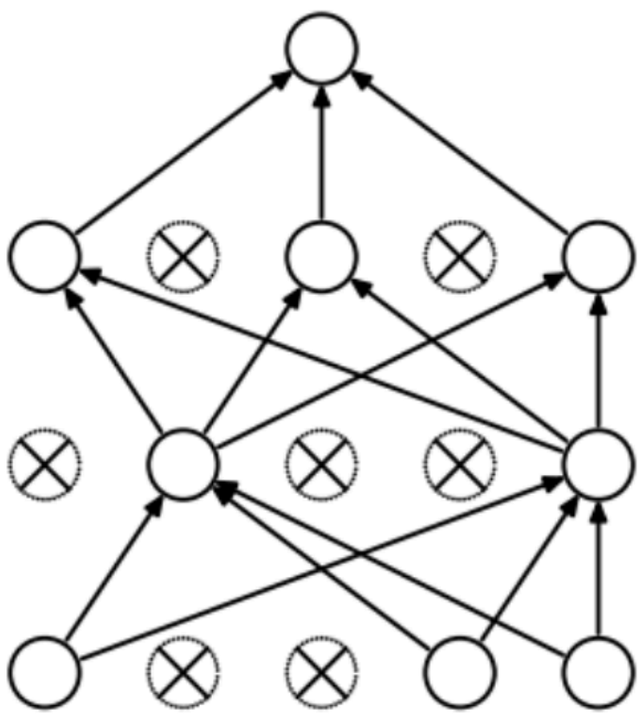


(b) After applying dropout.

Dropout Regularization

Improving neural networks by preventing co-adaptation of feature detectors (2012)

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada



(b) After applying dropout.

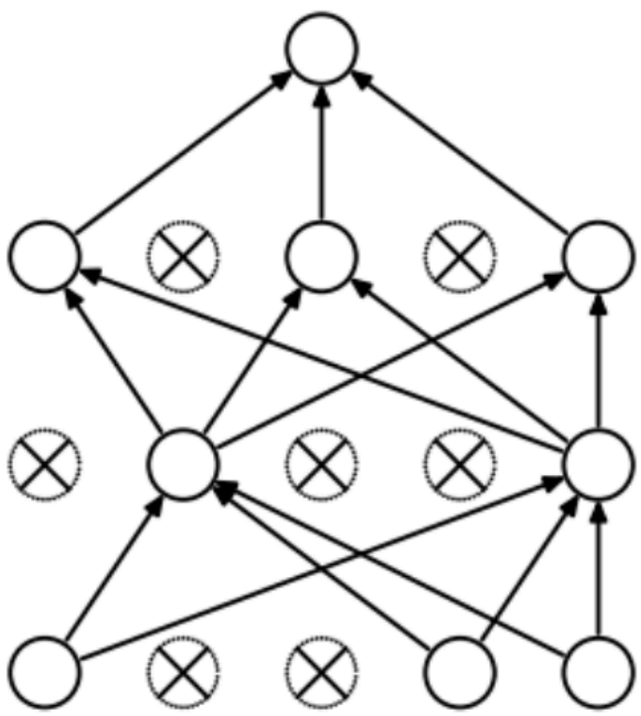
$$\mathbf{h}_{n,l} = f_l(\mathbf{h}_{n,l-1}\mathbf{\Lambda}_l\mathbf{W}_l)$$

Diagonal matrix of Bernoulli random variables.

Dropout Regularization

Improving neural networks by preventing co-adaptation of feature detectors (2012)

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada



(b) After applying dropout.

$$\mathbf{h}_{n,l} = f_l(\mathbf{h}_{n,l-1}\mathbf{\Lambda}_l\mathbf{W}_l)$$

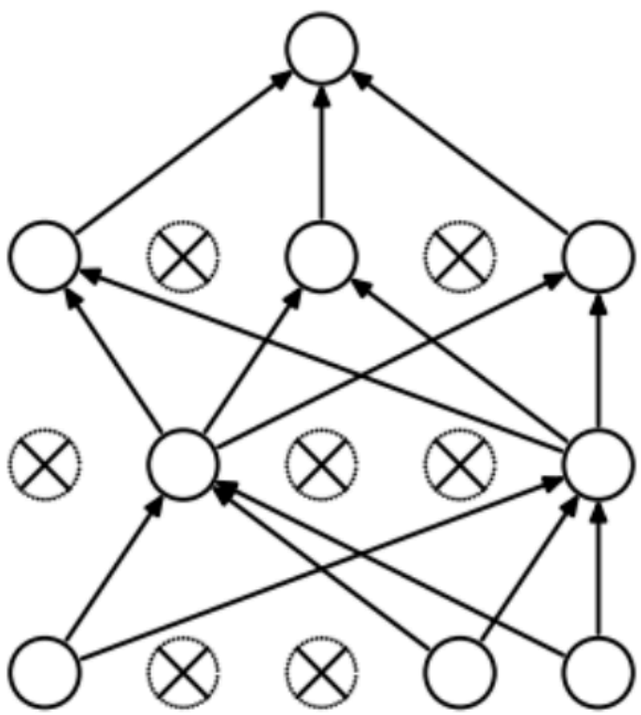
Diagonal matrix of Bernoulli random variables.

$$\mathcal{L}_{\mathrm{MN}}(\{\mathbf{W}_l\}_{l=1}^{L+1}) = \mathbb{E}_{p(\lambda)}[\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{\Lambda}_l\}_{l=1}^{L})]$$

Dropout Regularization

Improving neural networks by preventing co-adaptation of feature detectors (2012)

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada



(b) After applying dropout.

$$\mathbf{h}_{n,l} = f_l(\mathbf{h}_{n,l-1}\mathbf{\Lambda}_l\mathbf{W}_l)$$

Diagonal matrix of Bernoulli random variables.

$$\begin{aligned}
&\mathcal{L}_{\mathrm{MN}}(\{\mathbf{W}_l\}_{l=1}^{L+1}) \\
&\quad = \mathbb{E}_{p(\lambda)}[\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{\Lambda}_l\}_{l=1}^{L})] \\
&\quad \approx \frac{1}{S}\sum_{s=1}^{S} \log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\hat{\mathbf{\Lambda}}_{l,s}\}_{l=1}^{L})
\end{aligned}$$

Understanding Dropout

- **How should we interpret dropout?**
  How does it work?

Improving neural networks by preventing co-adaptation of feature detectors (2012)

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada

Understanding Dropout

- **How should we interpret dropout?**
  How does it work?

Improving neural networks by preventing co-adaptation of feature detectors (2012)

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada

"I went to my bank. The tellers kept changing and…this made me realize that ***randomly removing a different subset of neurons on each example would prevent conspiracies and thus reduce overfitting****."*

~ Geoffrey Hinton

Understanding Dropout

- How should we interpret dropout?
  How does it work?

Dropout Training as Adaptive Regularization

Stefan Wager*, **Sida Wang**†, and **Percy Liang**†
Departments of Statistics* and Computer Science†
Stanford University, Stanford, CA-94305
swager@stanford.edu, {sidaw, pliang}@cs.stanford.edu

(2013)

$$\mathcal{L}_{\mathrm{MN}}(\{\mathbf{W}_l\}_{l=1}^{L+1}) = \mathbb{E}_{p(\lambda)}[\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{\Lambda}_l\}_{l=1}^{L})]$$

Understanding Dropout

- How should we interpret dropout?
  How does it work?

Dropout Training as Adaptive Regularization

Stefan Wager*, **Sida Wang**†, and **Percy Liang**†
Departments of Statistics* and Computer Science†
Stanford University, Stanford, CA-94305
swager@stanford.edu, {sidaw, pliang}@cs.stanford.edu

(2013)

$$\mathcal{L}_{\text{MN}}(\{\mathbf{W}_l\}_{l=1}^{L+1}) = \mathbb{E}_{p(\lambda)}[\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{\Lambda}_l\}_{l=1}^{L})]$$

$$\approx \log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\boldsymbol{\mu}_l^{\Lambda}\}_{l=1}^{L}) + \frac{1}{2}\text{Var}[\lambda]\text{Tr}\left\{\nabla^2_{\boldsymbol{\mu}^{\Lambda}} \log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\boldsymbol{\mu}_l^{\Lambda}\}_{l=1}^{L})\right\}$$

Log-Likelihood

Regularization Penalty

Understanding Dropout

- How should we interpret dropout?
  How does it work?

Dropout Training as Adaptive Regularization

Stefan Wager*, **Sida Wang**†, and **Percy Liang**†
Departments of Statistics* and Computer Science†
Stanford University, Stanford, CA-94305
swager@stanford.edu, {sidaw, pliang}@cs.stanford.edu
(2013)

For logistic regression:

$$\frac{1}{2}\mathrm{Var}[\lambda]\mathrm{Tr}\left\{\nabla^2_{\boldsymbol{\mu}^\Lambda}\log p(\mathbf{y}|\mathbf{X},\{\mathbf{W}_l\}_{l=1}^{L+1},\{\boldsymbol{\mu}_l^\Lambda\}_{l=1}^{L})\right\} =$$

$$-\frac{1}{2}\pi(1-\pi)\sum_{n=1}^{N}\sum_{d=1}^{D} f(\boldsymbol{\beta}^T\mathbf{x}_n)(1-f(\boldsymbol{\beta}^T\mathbf{x}_n))x_{n,d}^2\beta_d^2$$

Understanding Dropout

- **General case of deep neural networks thought to be hopeless due to intractability of expectation.**

$$\mathbb{E}_{p(\lambda)}[\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{\Lambda}_l\}_{l=1}^{L})]$$

Understanding Dropout

- **General case of deep neural networks thought to be hopeless due to intractability of expectation.**

$$\mathbb{E}_{p(\lambda)}[\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{\Lambda}_l\}_{l=1}^{L})]$$

- **Open Questions**
  - **Why Bernoulli noise?**: Other noise distributions shown to work just as well (e.g. Gaussian). What does the choice of distribution mean?

Understanding Dropout

- **General case of deep neural networks thought to be hopeless due to intractability of expectation.**

$$\mathbb{E}_{p(\lambda)}[\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{\Lambda}_l\}_{l=1}^{L})]$$

- **Open Questions**
  - **Why Bernoulli noise?**: Other noise distributions shown to work just as well (e.g. Gaussian). What does the choice of distribution mean?
  - **Why drop hidden units?**: Dropping hidden units (instead of weights) results in best performance—why?

Understanding Dropout

- **General case of deep neural networks thought to be hopeless due to intractability of expectation.**

$$\mathbb{E}_{p(\lambda)}[\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{\Lambda}_l\}_{l=1}^{L})]$$

- **Open Questions**
  - **Why Bernoulli noise?**: Other noise distributions shown to work just as well (e.g. Gaussian). What does the choice of distribution mean?
  - **Why drop hidden units?**: Dropping hidden units (instead of weights) results in best performance—why?
  - **Other architectures?**: How should dropout be implemented in other NN architectures?

Dropout as Structured Bayesian Shrinkage

A Bayesian Interpretation

Idea: Don't go through the expectation, go around it.

$$\mathbb{E}_{p(\lambda)}[\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{\Lambda}_l\}_{l=1}^{L})]$$

A Bayesian Interpretation

Idea: Don't go through the expectation, go around it.

$$\mathbb{E}_{p(\lambda)}[\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{\Lambda}_l\}_{l=1}^{L})]$$

Assume a Gaussian prior on a weight:

$$w \sim \mathrm{N}(0, \sigma_0^2)$$

A Bayesian Interpretation

Idea: Don't go through the expectation, go around it.

$$\mathbb{E}_{p(\lambda)}[\log p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}_l\}_{l=1}^{L+1}, \{\mathbf{\Lambda}_l\}_{l=1}^{L})]$$

Assume a Gaussian prior on a weight:

$$w \sim \mathrm{N}(0, \sigma_0^2)$$

Consider the product of the weight and noise:

$$\lambda \cdot w, \quad \lambda \sim p(\lambda)$$

What's the distribution of this product?

A Bayesian Interpretation

Scale Mixtures! [Beale & Mallows, 1959; Andrews & Mallows, 1974]:

$$\lambda \cdot w, \quad \lambda \sim p(\lambda), \quad w \sim \mathrm{N}(0, \sigma_0^2)$$

Definition of a Gaussian scale mixture

A Bayesian Interpretation

- **Scale Mixtures!** [Beale & Mallows, 1959; Andrews & Mallows, 1974]:

$$\lambda \cdot w, \quad \lambda \sim p(\lambda), \quad w \sim \mathrm{N}(0, \sigma_0^2)$$

Definition of a Gaussian scale mixture

- **Can switch to a hierarchical parametrization:**

$$w \sim \mathrm{N}(0, \lambda^2 \sigma_0^2), \;\; \lambda \sim p(\lambda)$$

Reparametrizing Deep Networks

Can apply the trick inside the deep network:

$$\mathbf{h}_{n,l} = f_l(\mathbf{h}_{n,l-1}\mathbf{\Lambda}_l\mathbf{W}_l)$$

Induced scale mixture

Reparametrizing Deep Networks

Can apply the trick inside the deep network:

$$\mathbf{h}_{n,l} = f_l(\mathbf{h}_{n,l-1}\boldsymbol{\Lambda}_l\mathbf{W}_l)$$

Induced scale mixture

Reparametrize

Noise moves from likelihood to prior

$$\mathbf{h}_{n,l} = f_l(\mathbf{h}_{n,l-1}\mathbf{W}_l), \quad w_{i,j} \sim \mathrm{N}(0, \lambda_i^2\sigma_0^2)$$

Works no matter how deep the NN or the type of non-linearity!

Reparametrizing Deep Networks

Can apply the trick inside the deep network:

$$\mathbf{h}_{n,l} = f_l(\mathbf{h}_{n,l-1}\boldsymbol{\Lambda}_l\mathbf{W}_l)$$

Induced scale mixture

Reparametrize

$$\mathbf{h}_{n,l} = f_l(\mathbf{h}_{n,l-1}\mathbf{W}_l), \quad w_{i,j} \sim \mathrm{N}(0, \lambda_i^2\sigma_0^2)$$

Noise moves from likelihood to prior

Structure?

Works no matter how deep the NN or the type of non-linearity!

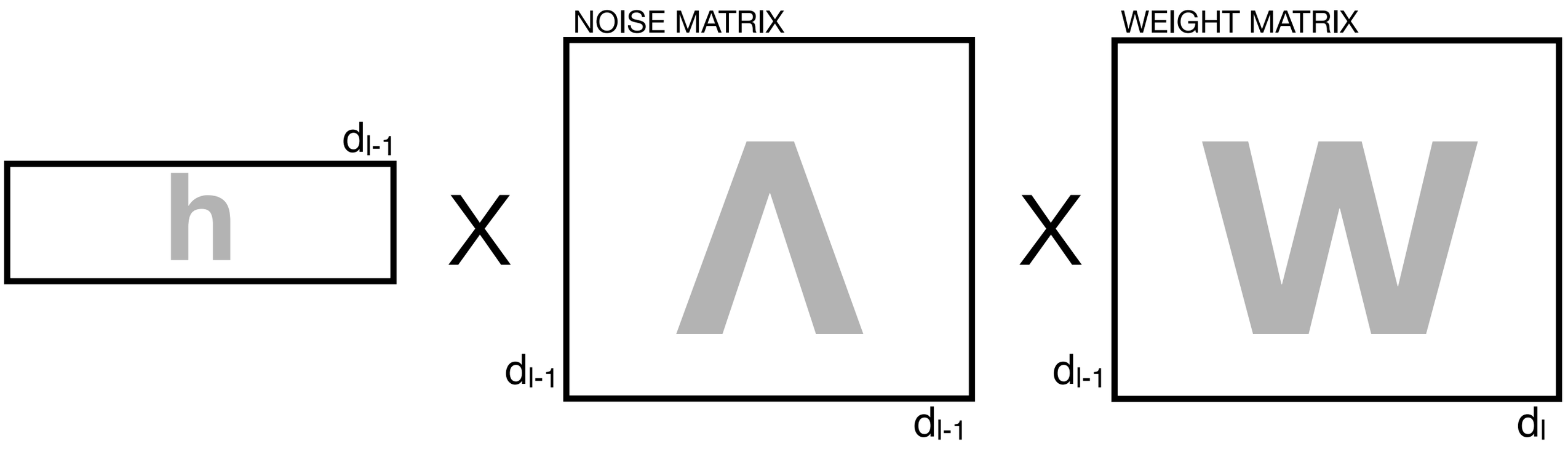
Reparametrizing Deep Networks

Sampling the noise on the hidden units induces structure:

$$\mathbf{h}_{n,l} = f_l(\mathbf{h}_{n,l-1}\mathbf{\Lambda}_l\mathbf{W}_l)$$

$$w_{i,j} \sim \mathrm{N}(0, \sigma_0^2)$$

$$\lambda_i \sim p(\lambda_i)$$



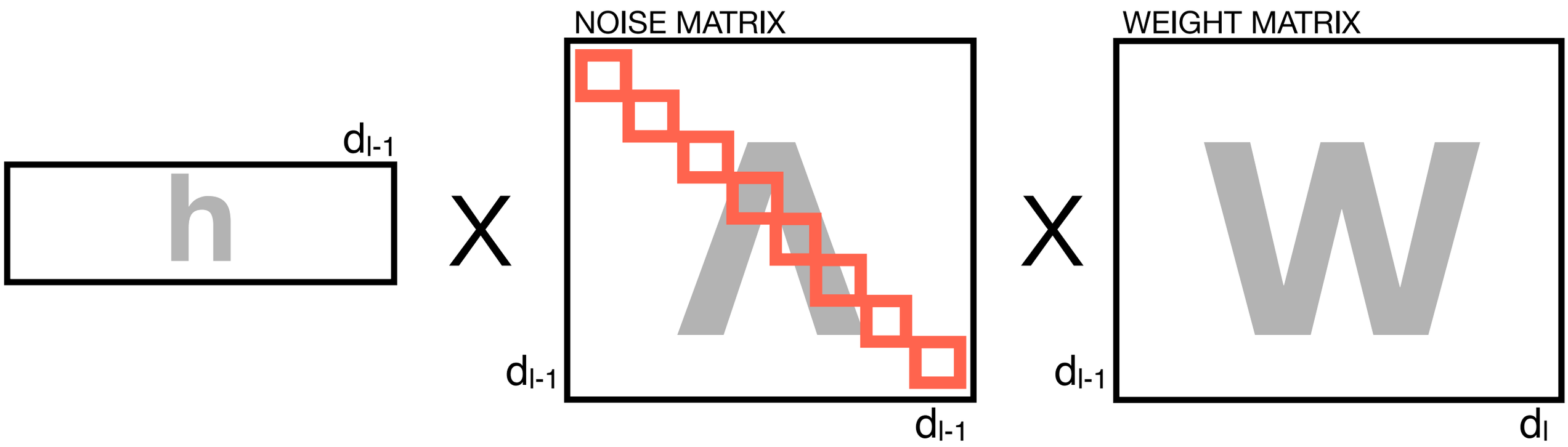
Reparametrizing Deep Networks

Sampling the noise on the hidden units induces structure:

$$\mathbf{h}_{n,l} = f_l(\mathbf{h}_{n,l-1}\mathbf{\Lambda}_l\mathbf{W}_l)$$

$$w_{i,j} \sim \mathrm{N}(0, \sigma_0^2)$$

$$\lambda_i \sim p(\lambda_i)$$



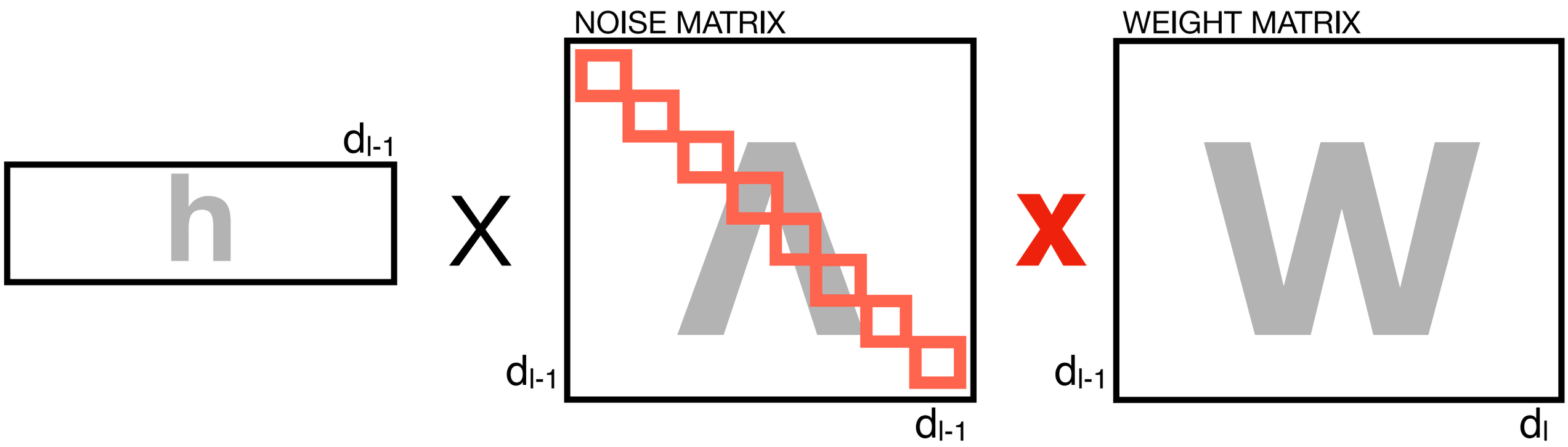
Reparametrizing Deep Networks

Sampling the noise on the hidden units induces structure:

$$\mathbf{h}_{n,l} = f_l(\mathbf{h}_{n,l-1}\mathbf{\Lambda}_l\mathbf{W}_l)$$

$$w_{i,j} \sim \mathrm{N}(0, \sigma_0^2)$$

$$\lambda_i \sim p(\lambda_i)$$



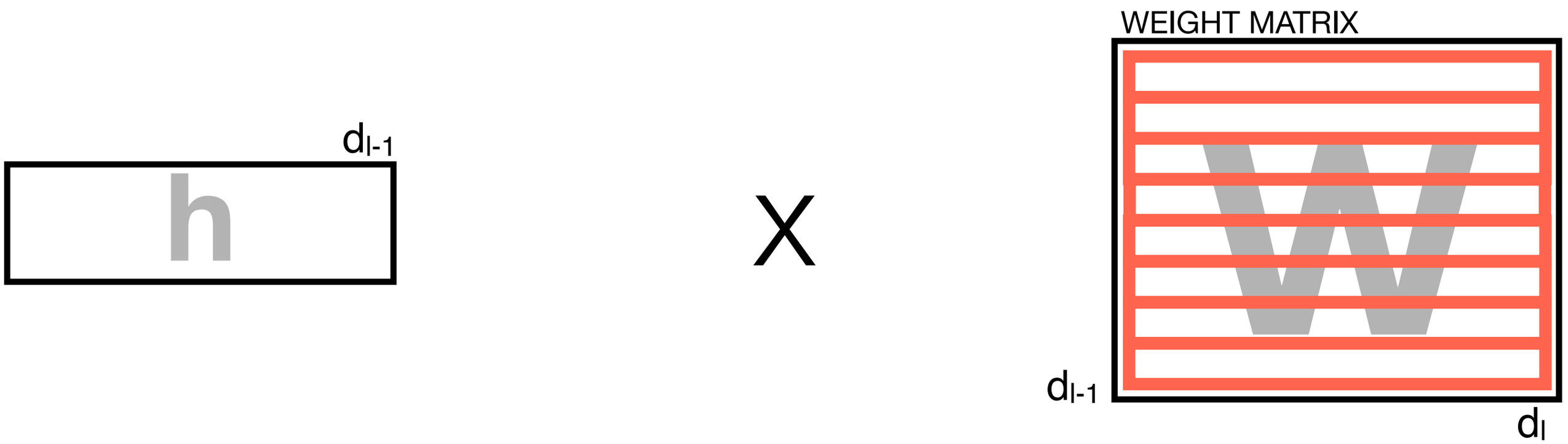
Reparametrizing Deep Networks

Sampling the noise on the hidden units induces structure:

$$\mathbf{h}_{n,l} = f_l(\mathbf{h}_{n,l-1}\mathbf{\Lambda}_l\mathbf{W}_l)$$

$$w_{i,j} \sim \mathrm{N}(0, \sigma_0^2)$$

$$\lambda_i \sim p(\lambda_i)$$



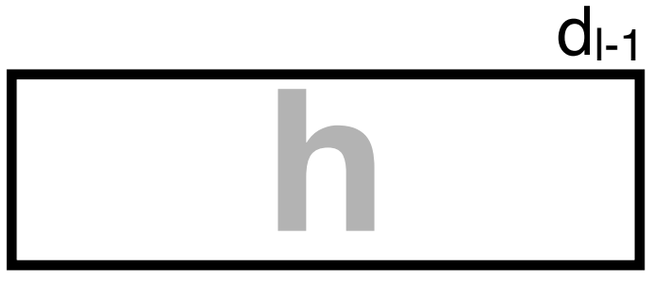
Reparametrizing Deep Networks

Sampling the noise on the hidden units induces structure:

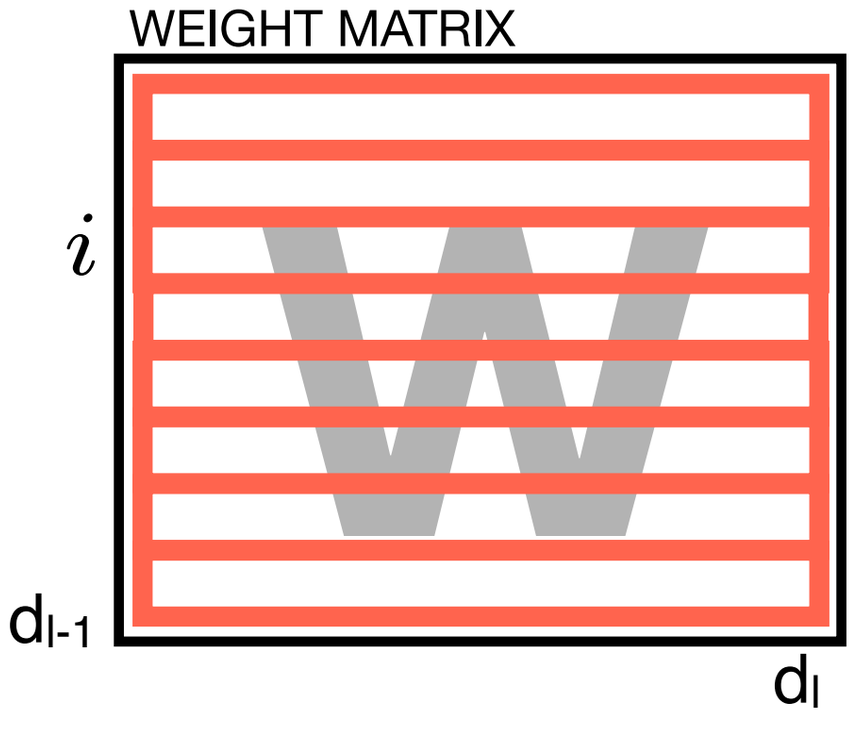
$$\mathbf{h}_{n,l} = f_l(\mathbf{h}_{n,l-1}\mathbf{W}_l), \quad w_{i,j} \sim \mathrm{N}(0, \lambda_i^2 \sigma_0^2)$$

$$\lambda_i \sim p(\lambda_i)$$

Selects for **number of hidden units** (i.e. layer width).



X



Reparametrizing Deep Networks

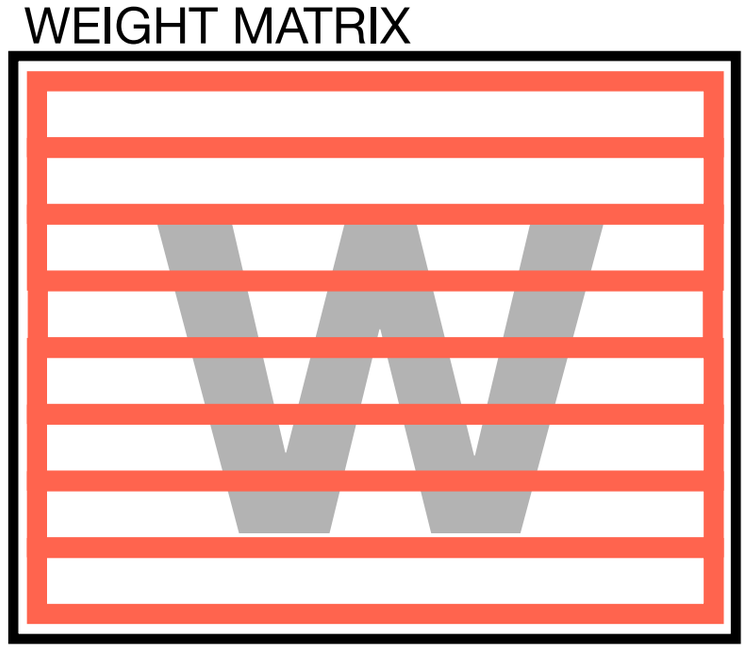
$$\mathbf{h}_{n,l} = f_l(\mathbf{h}_{n,l-1}\mathbf{W}_l), \quad w_{i,j} \sim \mathrm{N}(0, \lambda_i^2 \sigma_0^2)$$

i indexes rows

Exactly the *automatic relevance determination* (ARD) prior proposed by D. MacKay and R. Neal (1994)

MacKay

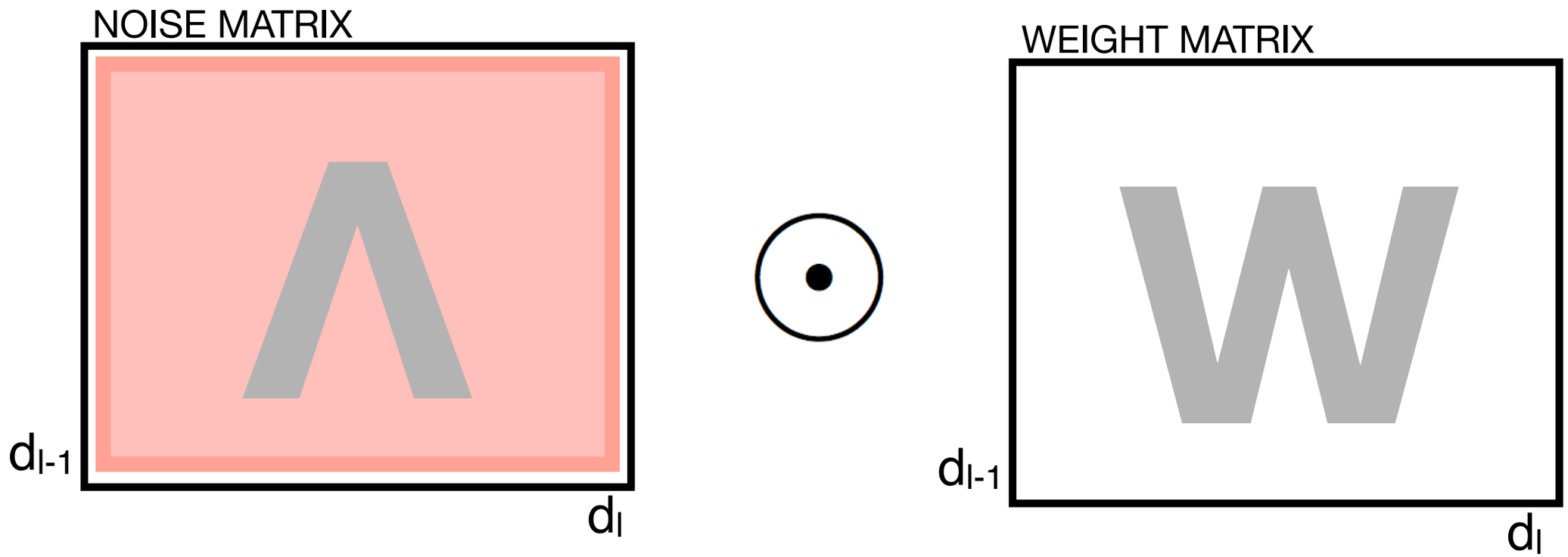
Neal



Shared scale

DropConnect

Previous work by Wan et al. (2013) proposed ***dropconnect***, which drops weights independently.



Has **not** gained wide adoption—likely because **ARD structure is broken.**

Understanding Dropout

- **Open Questions**
  - **Why Bernoulli noise?**: Other noise distributions shown to work just as well (e.g. Gaussian). What does the choice of distribution mean?
  - **Why drop hidden units?**: Dropping hidden units (instead of weights) results in best performance—why?
  - **Other architectures?**: How should dropout be implemented in other NN architectures?

Understanding Dropout

- **Open Questions**
  - **Why Bernoulli noise?**: Other noise distributions shown to work just as well (e.g. Gaussian). What does the choice of distribution mean?
  - **Why drop hidden units?**: Dropping hidden units (instead of weights) is crucial for performance—why?

    Because it induces ARD (scale sharing)
  - **Other architectures?**: How should dropout be implemented in other NN architectures?

Noise Priors

Is there anything special about **Bernoulli noise**?

Noise Priors

Is there anything special about **Bernoulli noise**?

Pushing through to the hierarchical parametrization,
we can see dropout's corresponding **marginal prior**:

$$
\begin{aligned}
p(w) &= \sum_{\lambda \in \{0,1\}} \texttt{Bernoulli}(\lambda; \pi)\ \mathrm{N}(0, \lambda^2 \sigma_0^2) \\
&= \pi\ \mathrm{N}(w; 0, \sigma_0^2) + (1 - \pi)\ \delta[w]
\end{aligned}
$$

Noise Priors

Is there anything special about **Bernoulli noise**?

Pushing through to the hierarchical parametrization, we can see dropout's corresponding **marginal prior**:

$$p(w) = \sum_{\lambda \in \{0,1\}} \texttt{Bernoulli}(\lambda; \pi)\ \mathrm{N}(0, \lambda^2 \sigma_0^2)$$

$$= \pi\ \mathrm{N}(w; 0, \sigma_0^2) + (1 - \pi)\ \delta[w]$$

Spike and slab prior commonly used for Bayesian variable selection. The **expanded parametrization** (i.e. likelihood noise) was used at least as early as Kuo and Mallick (1998).

Noise Priors

Other scale mixture priors as noise:

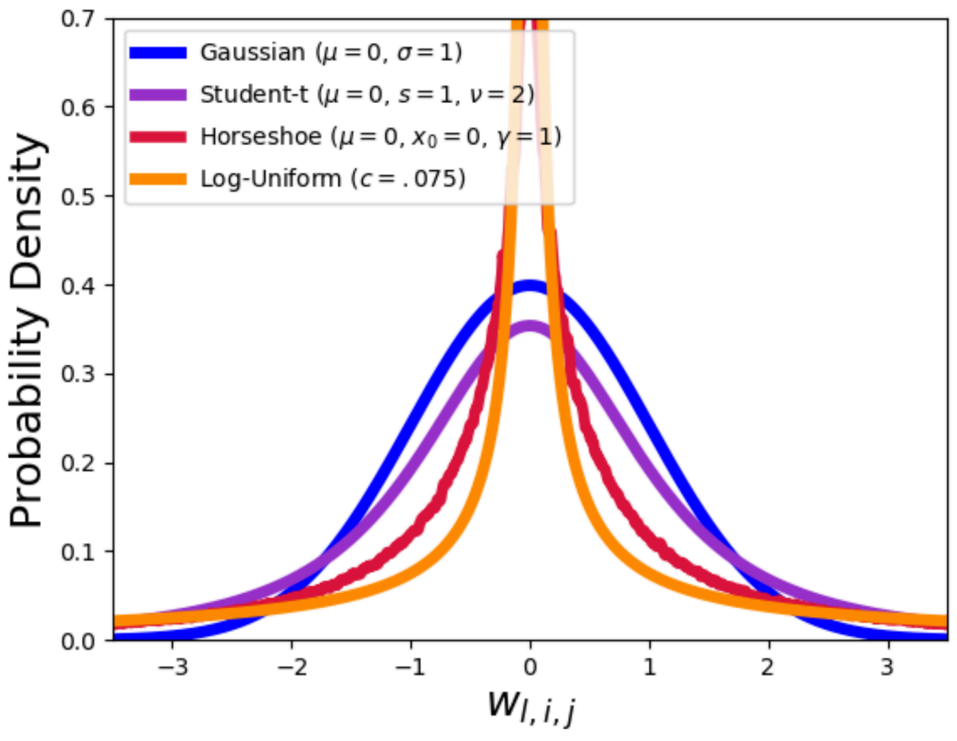
Noise Model $p(\lambda)$	Variance Prior $p(\lambda^2)$	Marginal Prior $p(w)$
Bernoulli	Bernoulli	Spike-and-Slab
Gaussian	χ^2	Generalized Hyperbolic
Rayleigh	Exponential	Laplace
Inverse Nakagami	Γ^{-1}	Student-t
Half-Cauchy	Unnamed	Horseshoe

Noise Priors

Other scale mixture priors as noise:

Noise Model $p(\lambda)$	Variance Prior $p(\lambda^2)$	Marginal Prior $p(w)$
Bernoulli	Bernoulli	Spike-and-Slab
Gaussian	χ^2	Generalized Hyperbolic
Rayleigh	Exponential	Laplace
Inverse Nakagami	Γ^{-1}	Student-t
Half-Cauchy	Unnamed	Horseshoe

Bernoulli noise is not essential and lots of other noise models / marginal priors work well, e.g. Horseshoe [Ghosh et al., 2018].

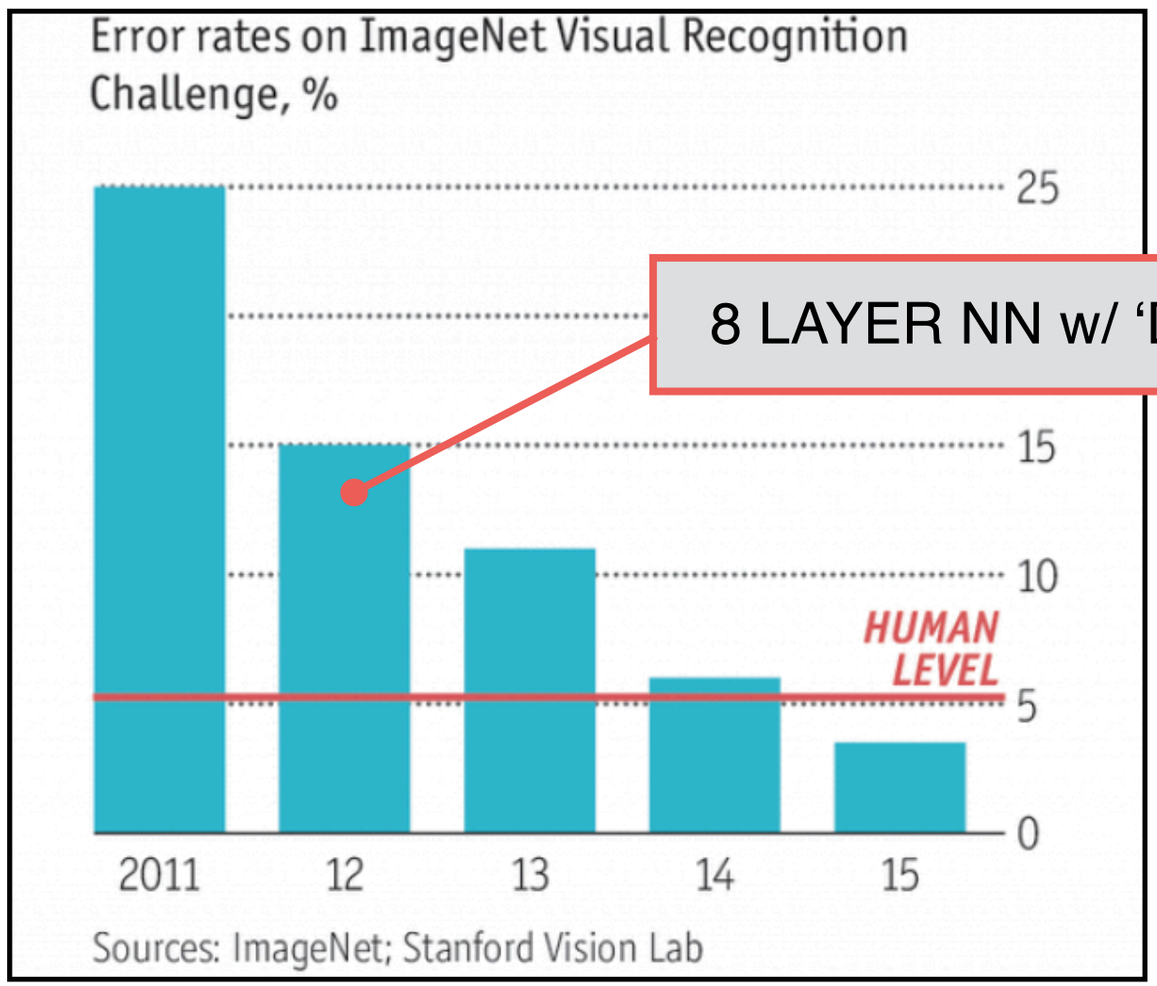


Understanding Dropout

- **Open Questions**
  - **Why Bernoulli noise?**: Other noise distributions shown to work just as well (e.g. Gaussian). What does the choice of distribution mean?
    Bernoulli is Spike and Slab. But not essential.
  - **Why drop hidden units?**: Dropping hidden units (instead of weights) is crucial for performance—why?
    Because it induces ARD (scale sharing)
  - **Other architectures?**: How should dropout be implemented in other NN architectures?

Residual Networks and Skip Connections

Computer Vision: Results on ImageNet object classification dataset.

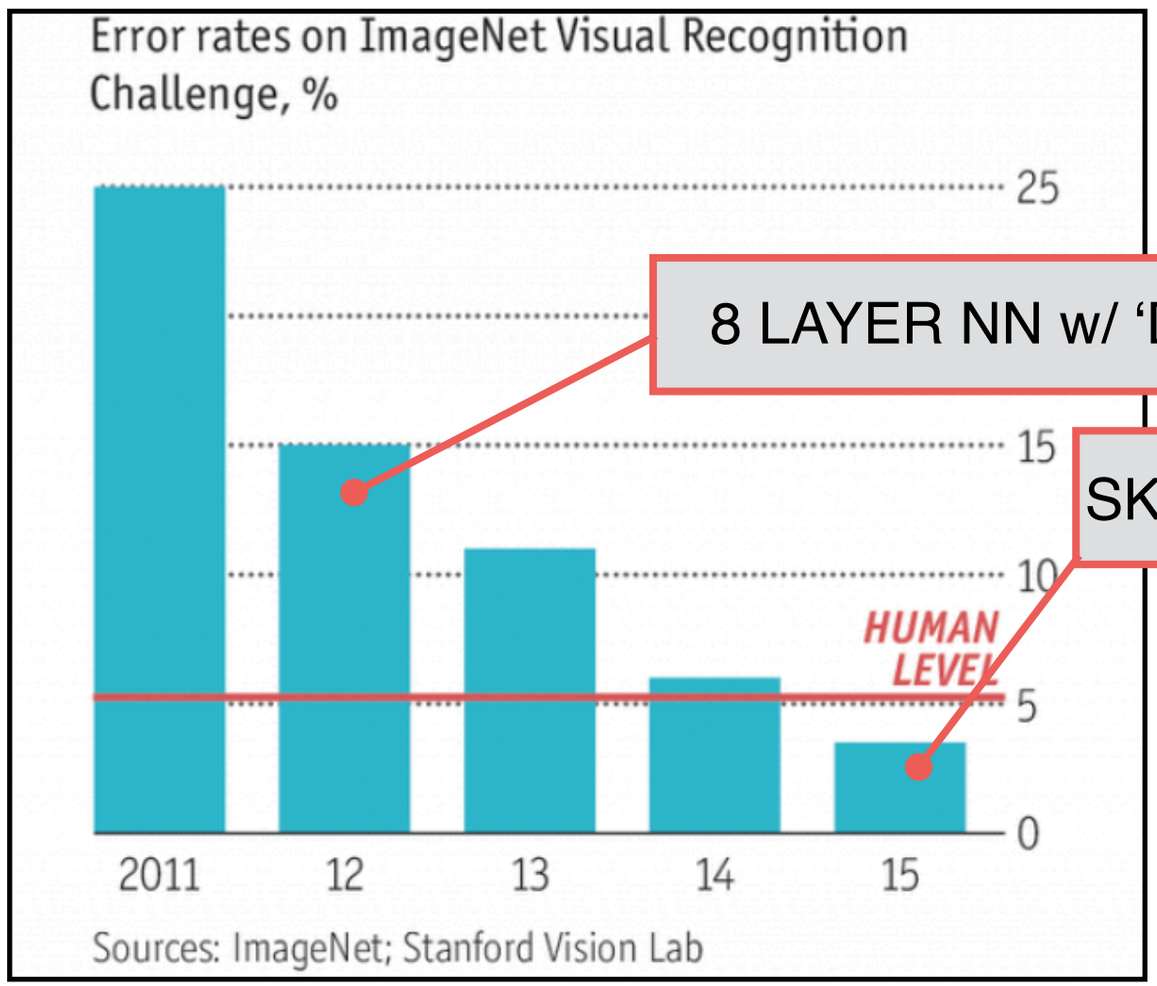


[Krizhevsky et al., 2012]

https://www.economist.com/news/special-report/21700756-artificial-intelligence-boom-based-old-idea-modern-twist-not

Residual Networks and Skip Connections

Computer Vision: Results on ImageNet object classification dataset.



[Krizhevsky et al., 2012]

[He et al., 2012]

https://www.economist.com/news/special-report/21700756-artificial-intelligence-boom-based-old-idea-modern-twist-not

Residual Networks and Skip Connections

Skip Connections: Identity connections to previous layer [He et al., 2015].

$$\mathbf{h}_i^l = f_l(\mathbf{h}_i^{l-1}\mathbf{W}_l + \mathbf{b}_l) \quad \text{vs} \quad \mathbf{h}_i^l = f_l(\mathbf{h}_i^{l-1}\mathbf{W}_l + \mathbf{b}_l) + \mathbf{h}_i^{l-1}$$

NONLINEAR TRANSFORM — SKIP CONNECTION

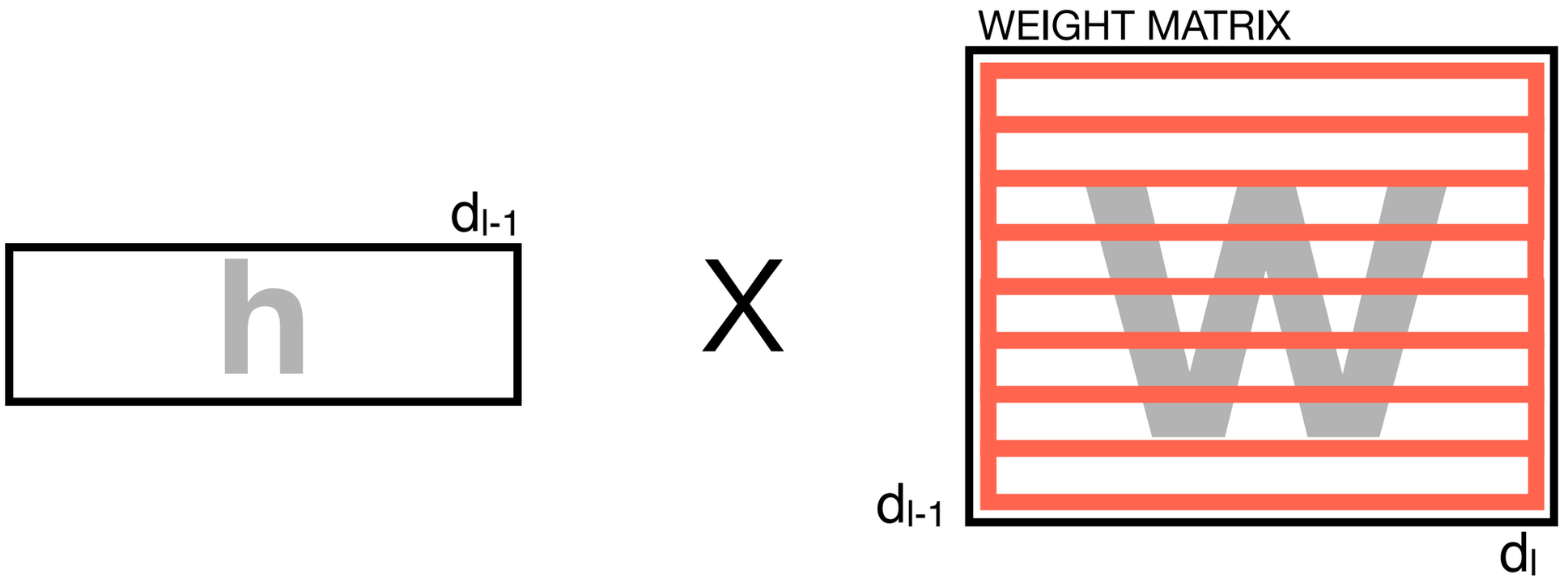
OLD — NEW

Can think of skip-connections as modeling the residual transformation:

$$\mathbf{h}_i^l - \mathbf{h}_i^{l-1} = f_l(\mathbf{h}_i^{l-1}\mathbf{W}_l + \mathbf{b}_l)$$

Residual Networks and Skip Connections

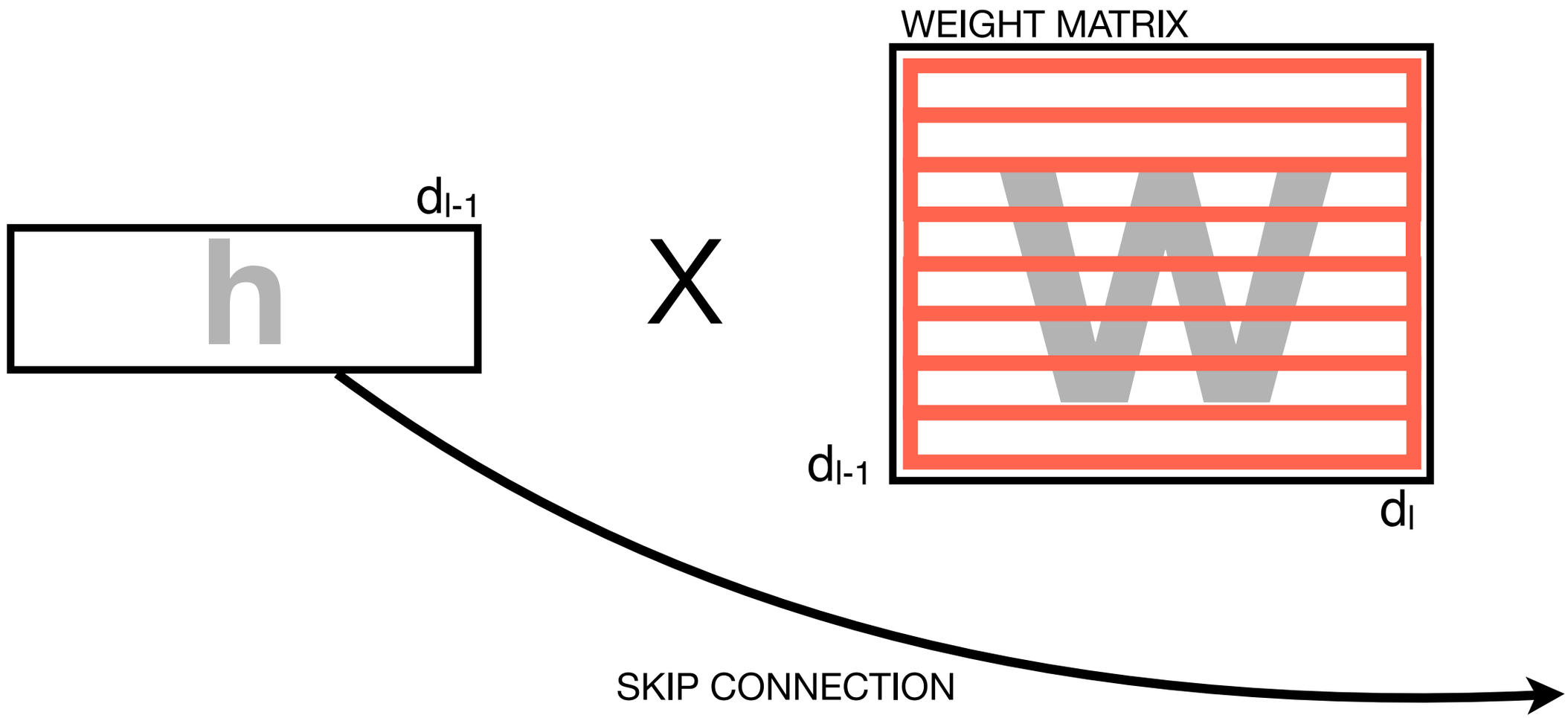
How should we implement dropout / multiplicative noise with skip connections?



$$\mathbf{h}_i^l = f_l(\mathbf{h}_i^{l-1}\mathbf{W}_l + \mathbf{b}_l)$$

Residual Networks and Skip Connections

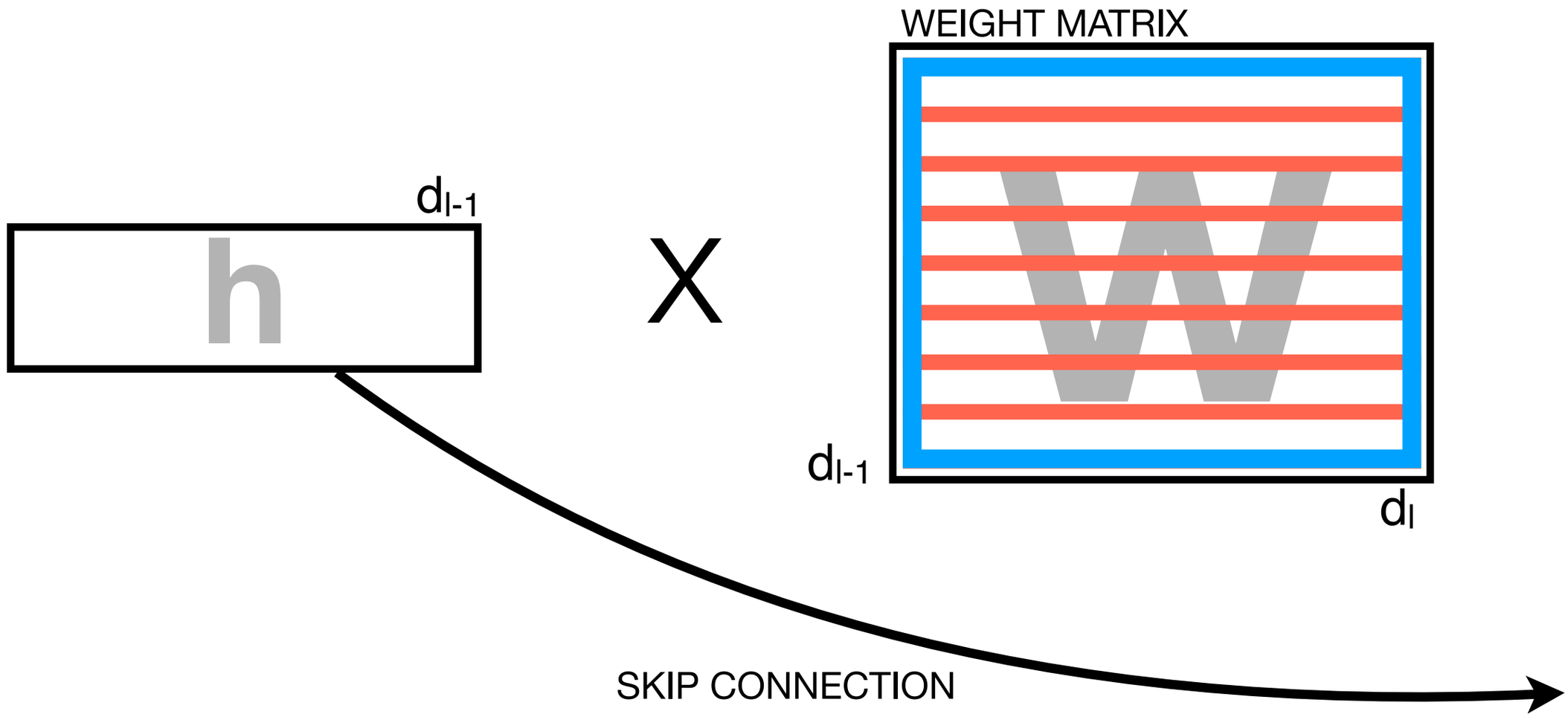
How should we implement dropout / multiplicative noise with skip connections?



$$\mathbf{h}_i^l = f_l(\mathbf{h}_i^{l-1}\mathbf{W}_l + \mathbf{b}_l) + \mathbf{h}_i^{l-1}$$

Residual Networks and Skip Connections

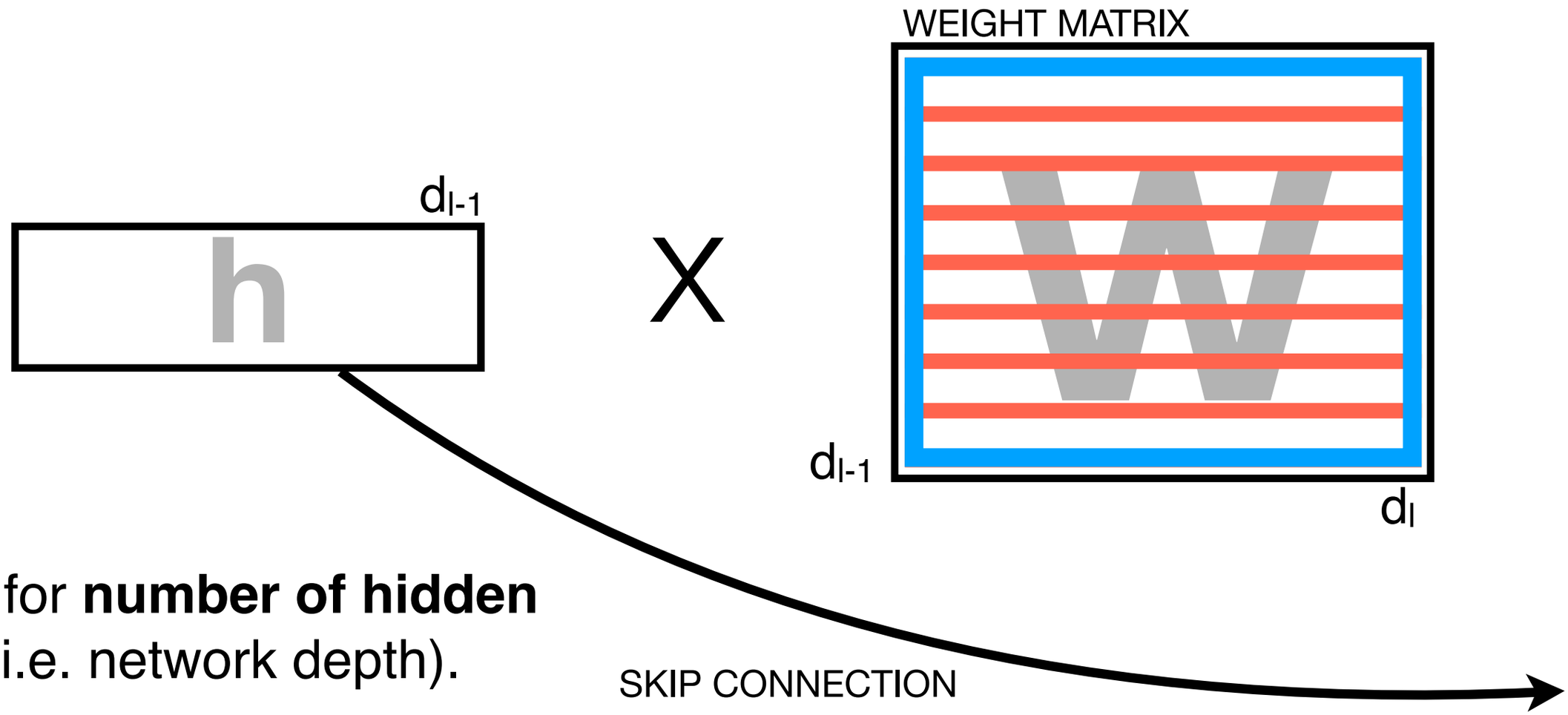
How should we implement dropout / multiplicative noise with skip connections?



$$\mathbf{h}_i^l = f_l(\mathbf{h}_i^{l-1}\mathbf{W}_l + \mathbf{b}_l) + \mathbf{h}_i^{l-1}$$

Residual Networks and Skip Connections

How should we implement dropout / multiplicative noise with skip connections?



Selects for **number of hidden layers** (i.e. network depth).

$$\mathbf{h}_i^l = f_l(\mathbf{h}_i^{l-1}\mathbf{W}_l + \mathbf{b}_l) + \mathbf{h}_i^{l-1}$$

Automatic Depth Determination

We can extend ARD to layer with the following prior, which we call **automatic depth determination** (ADD).

$$w_{l,i,j} \sim \mathbf{N}(w_{l,i,j}; 0, \sigma_0^2 \tau_{l,.,.}), \quad \tau_{l,.,.} \sim p(\tau_l)$$

Can reparametrize to interpret as multiplicative noise:

$$\mathbf{h}_{n,l} = f_l(\mathbf{h}_{n,l-1}\mathbf{W}_l) + \mathbf{h}_{n,l-1}$$

$$\xrightarrow{\text{reparametrization}} \quad f_l(\tau_l \mathbf{h}_{n,l-1}\mathbf{W}_l) + \mathbf{h}_{n,l-1}$$

Structured Priors

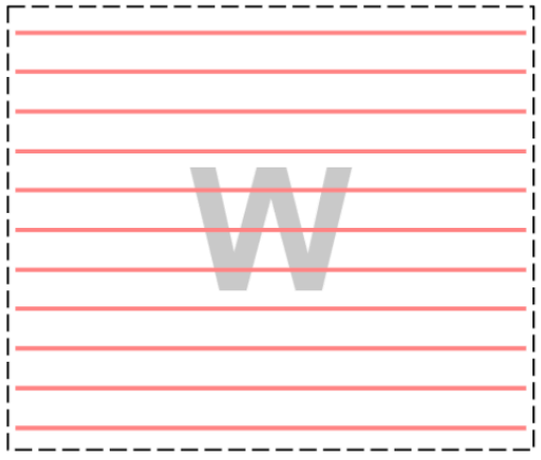
(a) Unstructured (DropConnect)

(b) ARD (Dropout)

(c) ADD (Stochastic Depth)

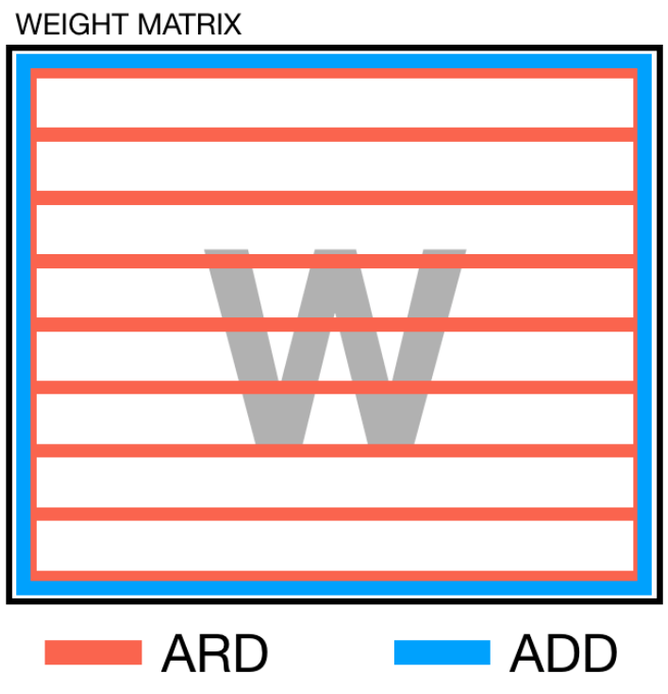
Structured Priors

(a) Unstructured (DropConnect)



(b) ARD (Dropout)

(c) ADD (Stochastic Depth)



Combine

$$w_{l,i,j} \sim \mathbf{N}(w_{l,i,j}; 0, \sigma_0^2 \lambda_{l,i,.} \tau_{l,.,.}),$$

$$\lambda_{l,i,.} \sim p(\lambda_{l,i}), \quad \tau_{l,.,.} \sim p(\tau_l)$$

Understanding Dropout

- **Open Questions**
  - **Why Bernoulli noise?**: Other noise distributions shown to work just as well (e.g. Gaussian). What does the choice of distribution mean?
    Bernoulli is Spike and Slab. But not essential.
  - **Why drop hidden units?**: Dropping hidden units (instead of weights) is crucial for performance—why?
    Because it induces ARD (scale sharing)
  - **Other architectures?**: How should dropout be implemented in other NN architectures?

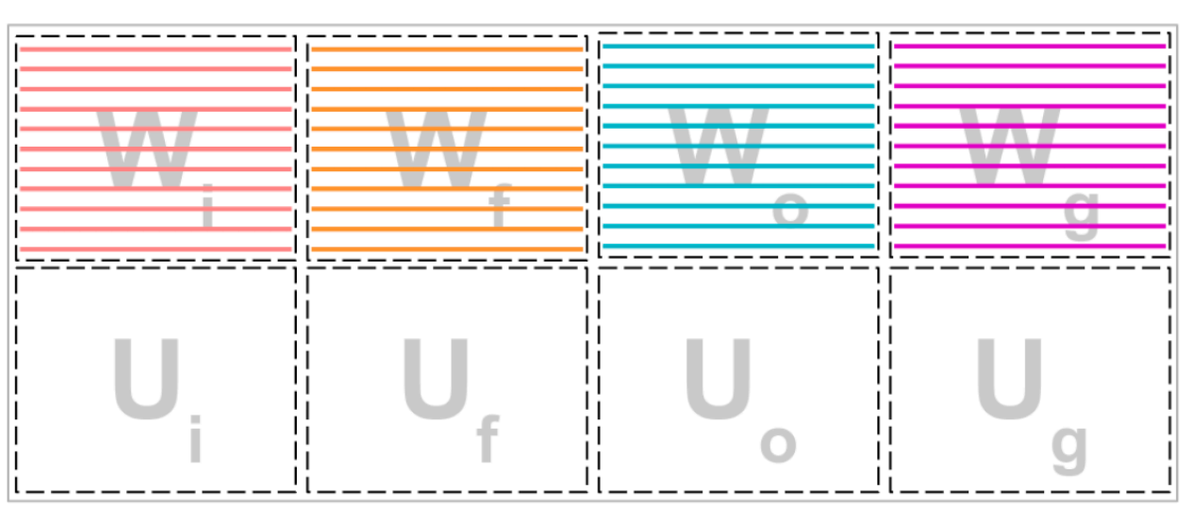
Understanding Dropout

- **Open Questions**
  - **Why Bernoulli noise?**: Other noise distributions shown to work just as well (e.g. Gaussian). What does the choice of distribution mean?
    Bernoulli is Spike and Slab. But not essential.
  - **Why drop hidden units?**: Dropping hidden units (instead of weights) is a bad idea. But it gives good performance—why?
    Because it induces ARD (scale sharing)
  - **Other architectures?**: How should dropout be implemented in other NN architectures?
    Think about params that should share scales (and be pruned together)

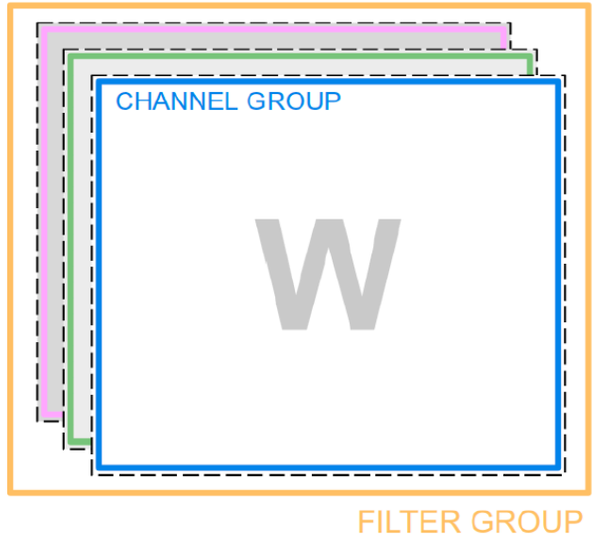
(a) Unstructured (DropConnect)

(b) ARD (Dropout)

(c) ADD (Stochastic Depth)



(d) LSTM ARD



(e) CNN ARD (SpatialDropout)

Understanding Dropout

Open Questions

- **Bernoulli is Spike and Slab. But not essential.**
- **Because it induces ARD (scale sharing)**
- **Think about params that should share scales (and be pruned together)**

Hold on: Aren't we sweeping all of this under the rug that is posterior inference?

Posterior Inference for Neural Networks

Hold on: Aren't we sweeping all of this under the rug that is posterior inference?

$$p(\{\mathbf{W}\}_{l=1}^{L+1}, \{\mathbf{\Lambda}_l\}_{l=1}^{L+1}, \{\tau_l\}_{l=1}^{L+1} | \mathbf{y}, \mathbf{X}) =$$

$$\frac{p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}\}_{l=1}^{L+1}) p(\{\mathbf{W}\}_{l=1}^{L+1} | \{\mathbf{\Lambda}_l\}_{l=1}^{L+1}, \{\tau_l\}_{l=1}^{L+1})\ p(\{\mathbf{\Lambda}_l\}_{l=1}^{L+1})\ p(\{\tau_l\}_{l=1}^{L+1})}{p(\mathbf{y}|\mathbf{X})}$$

where *L* can be 150+?!

Posterior Inference for Neural Networks

Hold on: Aren't we sweeping all of this under the rug that is posterior inference?

$$p(\{\mathbf{W}\}_{l=1}^{L+1}, \{\mathbf{\Lambda}_l\}_{l=1}^{L+1}, \{\tau_l\}_{l=1}^{L+1} | \mathbf{y}, \mathbf{X}) =$$

$$\frac{p(\mathbf{y}|\mathbf{X}, \{\mathbf{W}\}_{l=1}^{L+1}) p(\{\mathbf{W}\}_{l=1}^{L+1} | \{\mathbf{\Lambda}_l\}_{l=1}^{L+1}, \{\tau_l\}_{l=1}^{L+1})\ p(\{\mathbf{\Lambda}_l\}_{l=1}^{L+1})\ p(\{\tau_l\}_{l=1}^{L+1})}{p(\mathbf{y}|\mathbf{X})}$$

where *L* can be 150+?!

Posterior inference is an open problem. Especially with heavy-tailed priors. Previous work has needed truncated approximations [Ghosh et al., 2018], alternate parametrizations [Ingraham & Marks, 2017], and quasi-divergences [Hron et al., 2018] in addition to variational methods.

Light-Weight EM Algorithm

[Nalisnick & Hernández-Lobato, 2018]

Variational Inference for W: Perform backprop-style gradient updates for a Normal approx. of the weights.

$$\frac{\partial}{\partial \boldsymbol{\phi}} \mathcal{J}_{\text{ELBO}}(\boldsymbol{\phi}, \bar{\lambda}_i^*, \bar{\tau}_l^*) = \frac{\partial}{\partial \boldsymbol{\phi}} \mathbb{E}_{\text{N}(\boldsymbol{W};\boldsymbol{\phi})} \left[\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{W})\right] - \frac{\partial}{\partial \boldsymbol{\phi}} \text{KL}\left[\text{N}(\boldsymbol{W};\boldsymbol{\phi})||\text{N}(\boldsymbol{W}|\bar{\lambda}_i^*, \bar{\tau}_l^*)\right]$$

Light-Weight EM Algorithm

[Nalisnick & Hernández-Lobato, 2018]

Variational Inference for W: Perform backprop-style gradient updates for a Normal approx. of the weights.

$$\frac{\partial}{\partial \boldsymbol{\phi}} \mathcal{J}_{\text{ELBO}}(\boldsymbol{\phi}, \bar{\lambda}_i^*, \bar{\tau}_l^*) = \frac{\partial}{\partial \boldsymbol{\phi}} \mathbb{E}_{\text{N}(\boldsymbol{W};\boldsymbol{\phi})} \left[\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{W})\right] - \frac{\partial}{\partial \boldsymbol{\phi}} \text{KL}\left[\text{N}(\boldsymbol{W};\boldsymbol{\phi})||\text{N}(\boldsymbol{W}|\bar{\lambda}_i^*, \bar{\tau}_l^*)\right]$$

Closed-Form Updates for λ and τ: Update based on current estimates of **W**'s posterior parameters.

Light-Weight EM Algorithm

[Nalisnick & Hernández-Lobato, 2018]

Variational Inference for W: Perform backprop-style gradient updates for a Normal approx. of the weights.

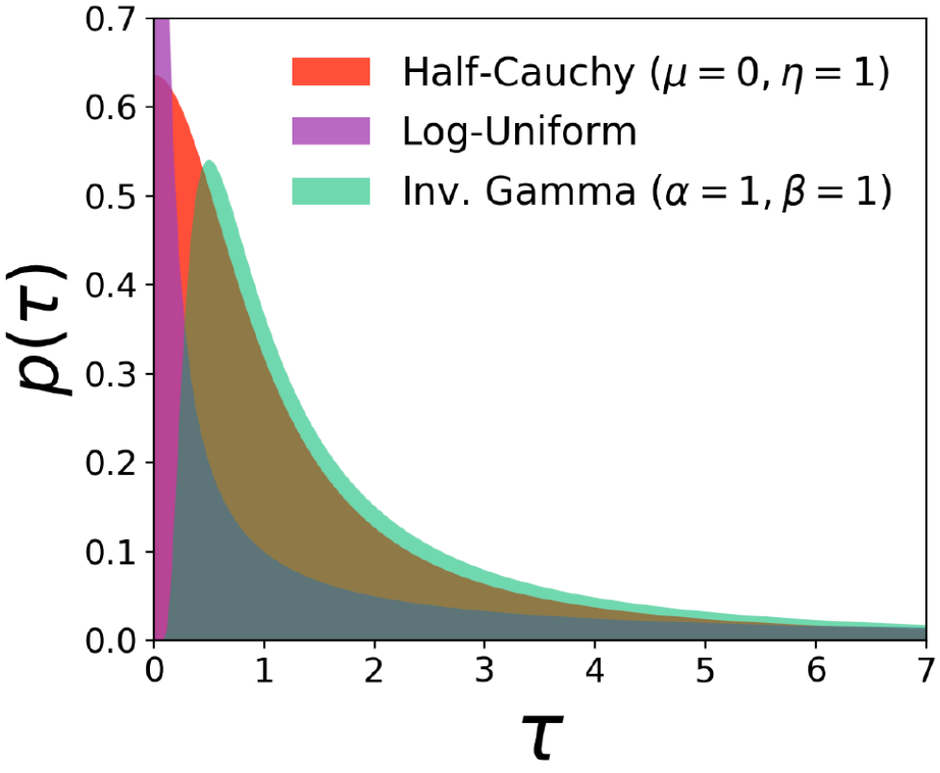
$$\frac{\partial}{\partial \boldsymbol{\phi}} \mathcal{J}_{\text{ELBO}}(\boldsymbol{\phi}, \bar{\lambda}_i^*, \bar{\tau}_l^*) = \frac{\partial}{\partial \boldsymbol{\phi}} \mathbb{E}_{\text{N}(\boldsymbol{W};\boldsymbol{\phi})} \left[\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{W})\right] - \frac{\partial}{\partial \boldsymbol{\phi}} \text{KL}\left[\text{N}(\boldsymbol{W};\boldsymbol{\phi})||\text{N}(\boldsymbol{W}|\bar{\lambda}_i^*, \bar{\tau}_l^*)\right]$$

Closed-Form Updates for λ and τ: Update based on current estimates of **W**'s posterior parameters.

Example, Inv. Gamma:

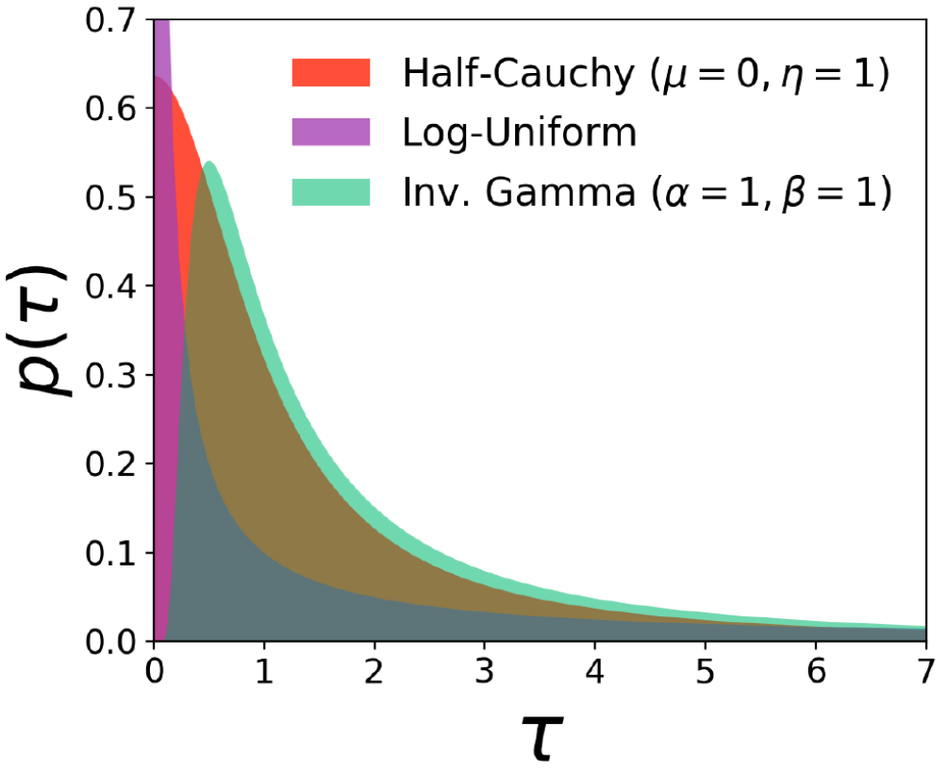
$$\bar{\lambda}_i^* = \frac{2\beta\bar{\tau}_l + \sum_j \sigma_{i,j}^2 + \mu_{i,j}^2}{\bar{\tau}_l(D_i + 2\alpha + 2)} \qquad \bar{\tau}_l^* = \frac{2\beta + \sum_i \bar{\lambda}_i^{-1} \sum_j \sigma_{i,j}^2 + \mu_{i,j}^2}{D + 2\alpha + 2}$$

EM Updates

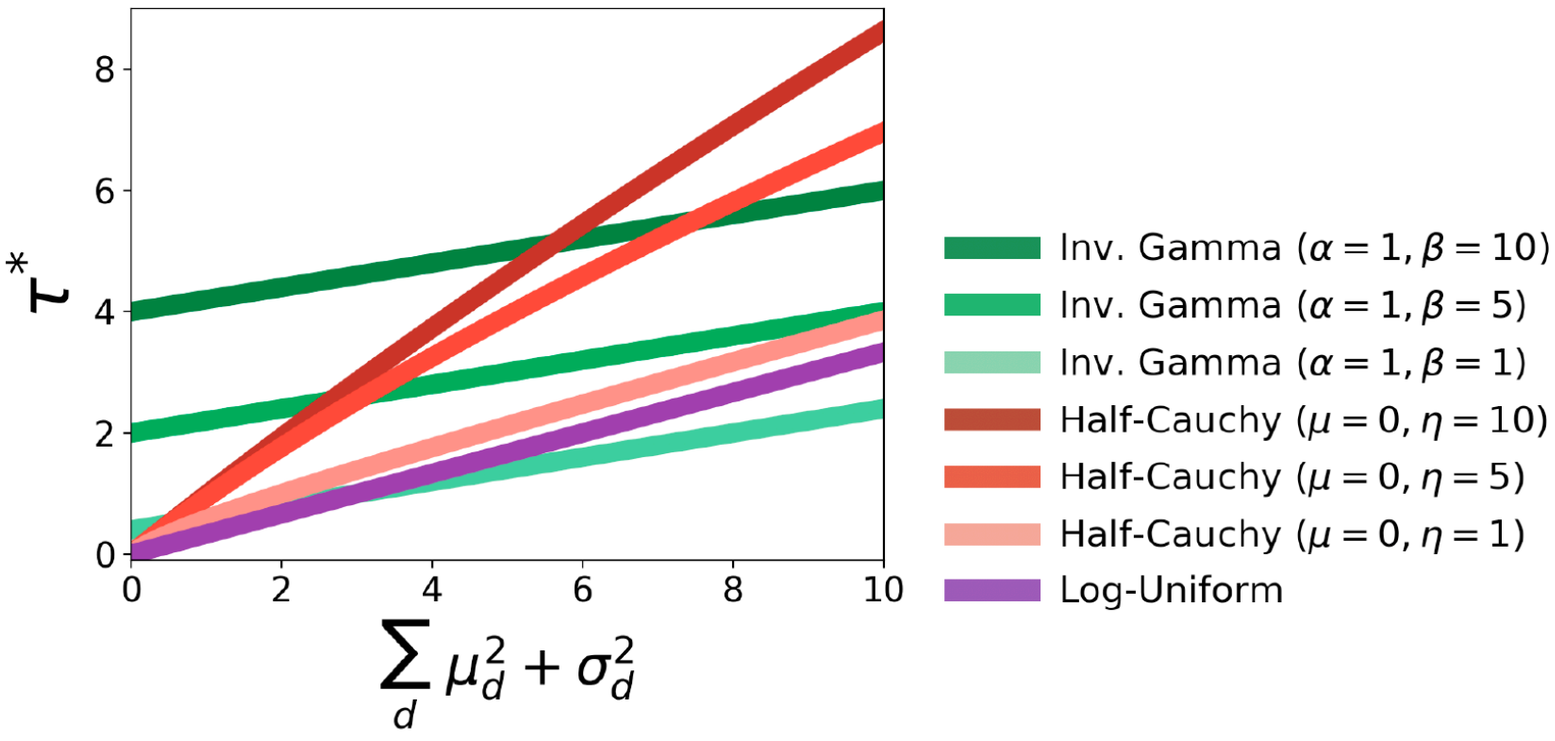


(a) Hyper-Priors

EM Updates



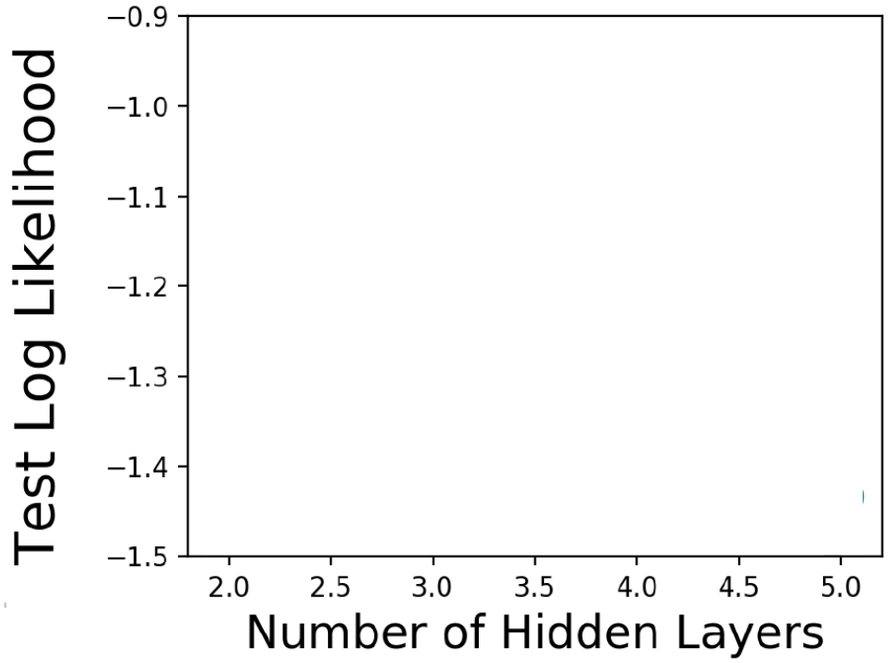
(a) Hyper-Priors



(b) EM Updates for Variance

Regression Results

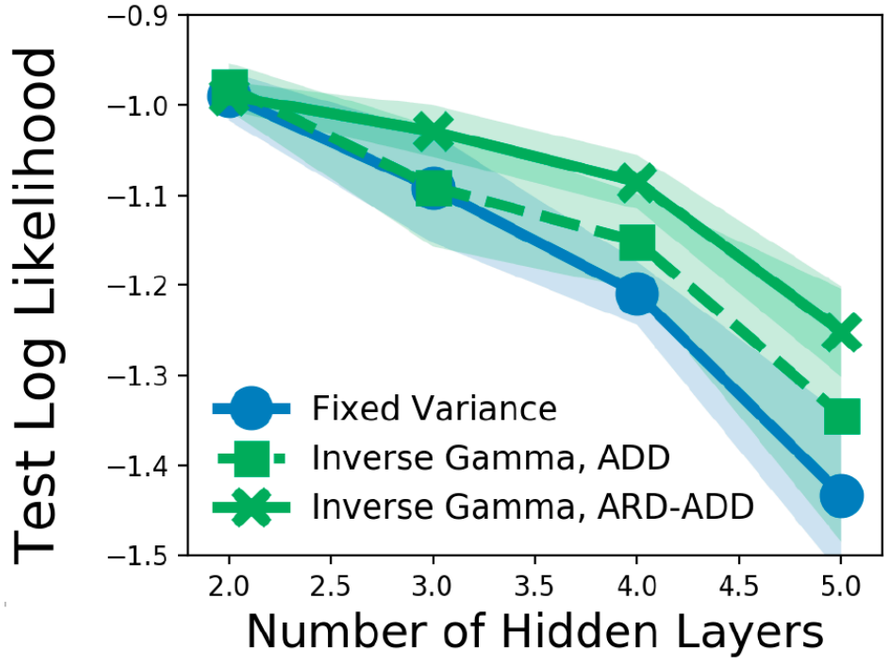
Regression: Test set performance on Yacht data set.



(a) Inverse Gamma $(\alpha = 1, \beta = 1)$

Regression Results

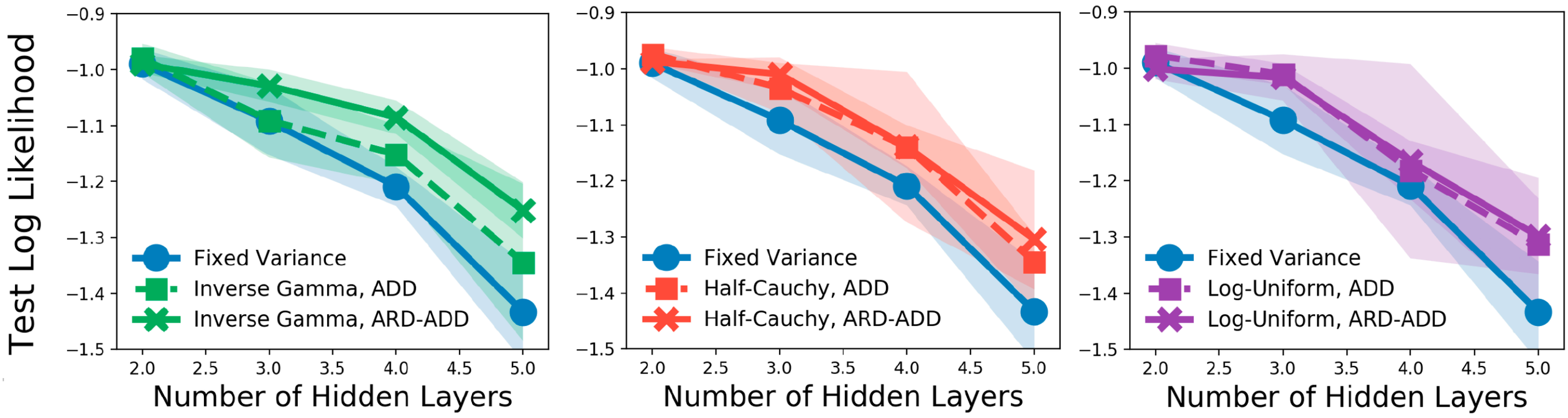
Regression: Test set performance on Yacht data set.



(a) Inverse Gamma $(\alpha = 1, \beta = 1)$

Regression Results

Regression: Test set performance on Yacht data set.



(a) Inverse Gamma $(\alpha = 1, \beta = 1)$ (b) Half-Cauchy $(\mu = 0, \eta = 2)$ (c) Log-Uniform

DISCUSSION

Statistics and Deep Learning

https://xkcd.com/1838/

We need principled approaches…

https://xkcd.com/1838/

Further Reading from a Statistical Perspective

- [ ] B. Cheng and D. M. Titterington
Neural Networks: A Review from a Statistical Perspective
Statistical Science, 1994

- [ ] B. Efron and T. Hastie
Chapter 18: Neural Networks and Deep Learning
Computer Age Statistical Inference
Cambridge University Press, 2016

- [ ] S. Mohamed
A Statistical View of Deep Learning
Blog post: http://blog.shakirm.com/wp-content/uploads/2015/07/SVDL.pdf, 2015

- [ ] E. Nalisnick, D. Tran, P. Smyth, and Y. W. Teh
Deep Learning: A Synthesis from Probabilistic Foundations
coming soon…

Further Reading from a Statistical Perspective

- [ ] B. Cheng and D. M. Titterington
Neural Networks: A Review from a Statistical Perspective
Statistical Science, 1994

- [ ] B. Efron and T. Hastie
Chapter 18: Neural Networks and Deep Learning
Computer Age Statistical Inference
Cambridge University Press, 2016

- [ ] S. Mohamed
A Statistical View of Deep Learning
Blog post: http://blog.shakirm.com/wp-content/uploads/2015/07/SVDL.pdf, 2015

- [ ] E. Nalisnick, D. Tran, P. Smyth, and Y. W. Teh
Deep Learning: A Synthesis from Probabilistic Foundations
coming soon…

Further Reading from a Statistical Perspective

B. Cheng and D. M. Titterington
Neural Networks: A Review from a Statistical Perspective
Statistical Science, 1994

The commentary is better yet…

Brian Ripley referring to some early NN experiments:
"…inadequately designed propaganda studies."

Further Reading from a Statistical Perspective

- [ ] B. Cheng and D. M. Titterington
Neural Networks: A Review from a Statistical Perspective
Statistical Science, 1994

Two lists from R. Tibshirani:

WHAT THE STATISTICIAN CAN LEARN FROM NEURAL NETWORK RESEARCHERS

1. We should worry less about statistical optimality and more about finding methods that work, especially with large data sets.
2. We should tackle difficult real data problems like some of those addressed by neural network researchers, like character and speech recognition and DNA structure prediction. As John Tukey has said, it is often better to get an approximate solution to a real problem than an exact solution to an oversimplified one.
3. Models with very large numbers of parameters can be useful for prediction, especially for large data sets and problems exhibiting high signal-to-noise ratios.
4. Modelling linear combinations of input variables can be a very effective approach because it provides both feature extraction and dimension reduction.
5. Iterative, nongreedy fitting algorithms (like steepest descent with a learning rate) can help to avoid overfitting in models with large numbers of parameters.
6. We (statisticians) should sell ourselves better.

WHAT THE NEURAL NETWORK RESEARCHER CAN LEARN FROM STATISTICIANS

1. They should worry more about statistical optimality or at least about the statistical properties of methods.
2. They should spend more effort comparing their methods to simpler statistical approaches. They will be surprised how often linear regression performs as well as a multilayered perceptron. They should not use a complicated model where a simple one will do.

Thank you. Questions?

In collaboration with…

Padhraic Smyth

José Miguel Hernández-Lobato

References

David F Andrews and Colin L Mallows. Scale Mixtures of Normal Distributions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 99–102, 1974.

EML Beale, CL Mallows, et al. Scale mixing of symmetric distributions with zero means. *The Annals of Mathematical Statistics*, 30(4):1145–1151, 1959.

Leo Breiman. Statistical modeling: The two cultures. *Statistical science*, 16(3):199–231, 2001.

Bing Cheng and D Michael Titterington. Neural Networks: A Review from a Statistical Perspective. *Statistical Science*, pages 2–30, 1994.

Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941, 2014.

J. Deng, K. Li, M. Do, H. Su, and L. Fei-Fei. Construction and Analysis of a Large Scale Image Ontology. Vision Sciences Society, 2009.

Bradley Efron and Trevor Hastie. *Computer age statistical inference*, volume 5. Cambridge University Press, 2016.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, pages 1050–1059, 2016.

Soumya Ghosh, Jiayu Yao, and Finale Doshi-Velez. Structured Variational Learning of Bayesian Neural Networks with Horseshoe Priors. *International Conference on Machine Learning*, 2018.

References

Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, pages 1225–1234, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Jiri Hron, Alexander Matthews, and Zoubin Ghahramani. Variational Bayesian Dropout: Pitfalls and Fixes. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 1–8, 2018.

John Ingraham and Debora Marks. Variational Inference for Sparse and Undirected Models. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1607–1616, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Lynn Kuo and Bani Mallick. Variable Selection for Regression Models. *Sankhyā: The Indian Journal of Statistics, Series B*, pages 65–81, 1998.

Yingzhen Li and Yarin Gal. Dropout inference in bayesian neural networks with alpha-divergences. In *International Conference on Machine Learning*, pages 2052–2061, 2017.

David JC MacKay. Bayesian non-linear modeling for the prediction competition. In *Maximum Entropy and Bayesian Methods*, pages 221–234. Springer, 1994.

Lars M. Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. *Proceedings of the 34th International Conference on Machine Learning*, 2017.

References

Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, pages 1791–1799, 2014.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, and Georg Ostrovski. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Shakir Mohamed. A statistical view of deep learning. *The Spectator: Shakir's Machine Learning Blog*, 2015.

Eric Nalisnick and Jos Miguel Hernndez-Lobato. Automatic Depth Determination for Bayesian ResNets. *In Submission*, 2018.

Eric Nalisnick and Padhraic Smyth. Unifying the Dropout Family Through Structured Shrinkage Priors. *ArXiv Preprint*, 2018.

Radford M Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1994.

John Ashworth Nelder and R Jacob Baker. *Generalized Linear Models*. Wiley Online Library, 1972.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533, 1986.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

References

Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient Object Localization Using Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 648–656, 2015.

Stefan Wager, Sida Wang, and Percy S Liang. Dropout Training as Adaptive Regularization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 351–359, 2013.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of Neural Networks Using DropConnect. In *International Conference on Machine Learning (ICML)*, pages 1058–1066, 2013.

Appendix

Dropout Objective as Marginal MAP

$$p(\{\mathbf{W}_l\}_{l=1}^{L+1}|\mathbf{y},\mathbf{X}) \propto p(\{\mathbf{W}_l\}_{l=1}^{L+1})\ \mathbb{E}_{p(\xi)}\left[p(\mathbf{y}|\mathbf{X},\{\mathbf{W}_l\}_{l=1}^{L+1},\{\mathbf{\Xi}_l\}_{l=1}^{L})\right]. \tag{9}$$

The marginal MAP estimate of the weights is then found by maximizing the logarithm of this expression:

$$\begin{aligned}
&\log p(\{\mathbf{W}_l\}_{l=1}^{L+1}|\mathbf{y},\mathbf{X}) \\
&\propto \log \mathbb{E}_{p(\xi)}\left[p(\mathbf{y}|\mathbf{X},\{\mathbf{W}_l\}_{l=1}^{L+1},\{\mathbf{\Xi}_l\}_{l=1}^{L})\right] \\
&\qquad + \frac{-1}{2\sigma_0^2}\sum_{l=1}^{L+1}\sum_{r=1}^{D_{l-1}}\sum_{j=1}^{D_l} w_{l,r,j}^2 \\
&= \mathcal{L}_{\text{MAP}}(\{\mathbf{W}_l\}_{l=1}^{L+1}).
\end{aligned} \tag{10}$$

Lastly, we perform the final two steps in the derivation: *(i)* use Jensen's inequality to lower-bound the first term, and *(ii)* assume the variance of the prior on the weights goes to infinity.

$$\begin{aligned}
&\mathcal{L}_{\text{MAP}}(\{\mathbf{W}_l\}_{l=1}^{L+1}) \\
&\geq \mathbb{E}_{p(\xi)}\left[\log p(\mathbf{y}|\mathbf{X},\{\mathbf{W}_l\}_{l=1}^{L+1},\{\mathbf{\Xi}_l\}_{l=1}^{L})\right] \\
&\qquad + \frac{-1}{2\sigma_0^2}\sum_{l=1}^{L+1}\sum_{r=1}^{D_{l-1}}\sum_{j=1}^{D_l} w_{l,r,j}^2 \\
&\to \mathbb{E}_{p(\xi)}\left[\log p(\mathbf{y}|\mathbf{X},\{\mathbf{W}_l\}_{l=1}^{L+1},\{\mathbf{\Xi}_l\}_{l=1}^{L})\right] \\
&= \mathcal{L}_{\text{MN}}(\{\mathbf{W}_l\}_{l=1}^{L+1}) \\
&\approx \frac{1}{S}\sum_{s=1}^{S}\log p(\mathbf{y}|\mathbf{X},\{\mathbf{W}_l\}_{l=1}^{L+1},\{\hat{\mathbf{\Xi}}_{l,s}\}_{l=1}^{L})
\end{aligned} \tag{11}$$