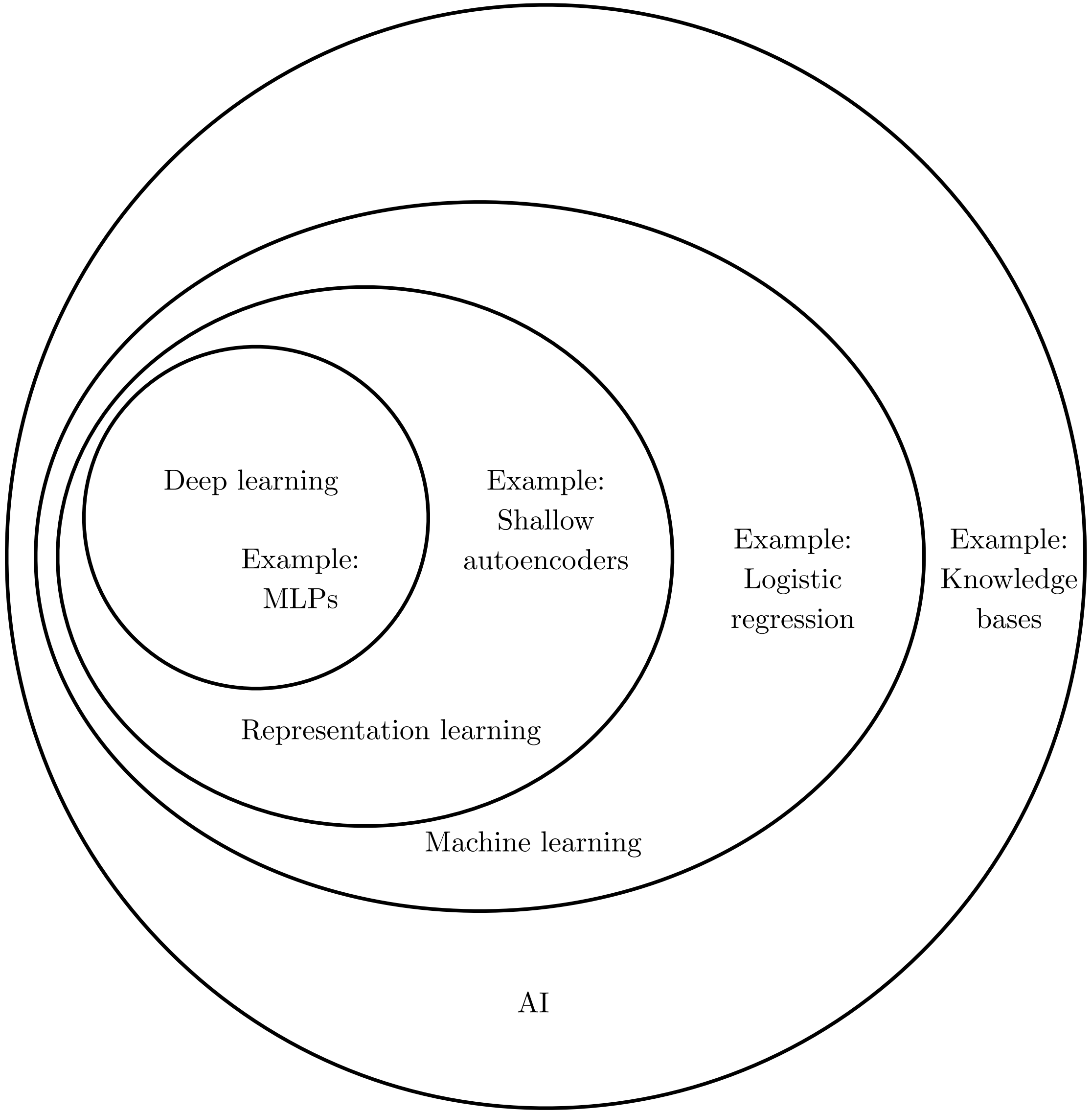
CS 466/566
Introduction to Deep Learning

Lecture 6 – Techniques for Training Better

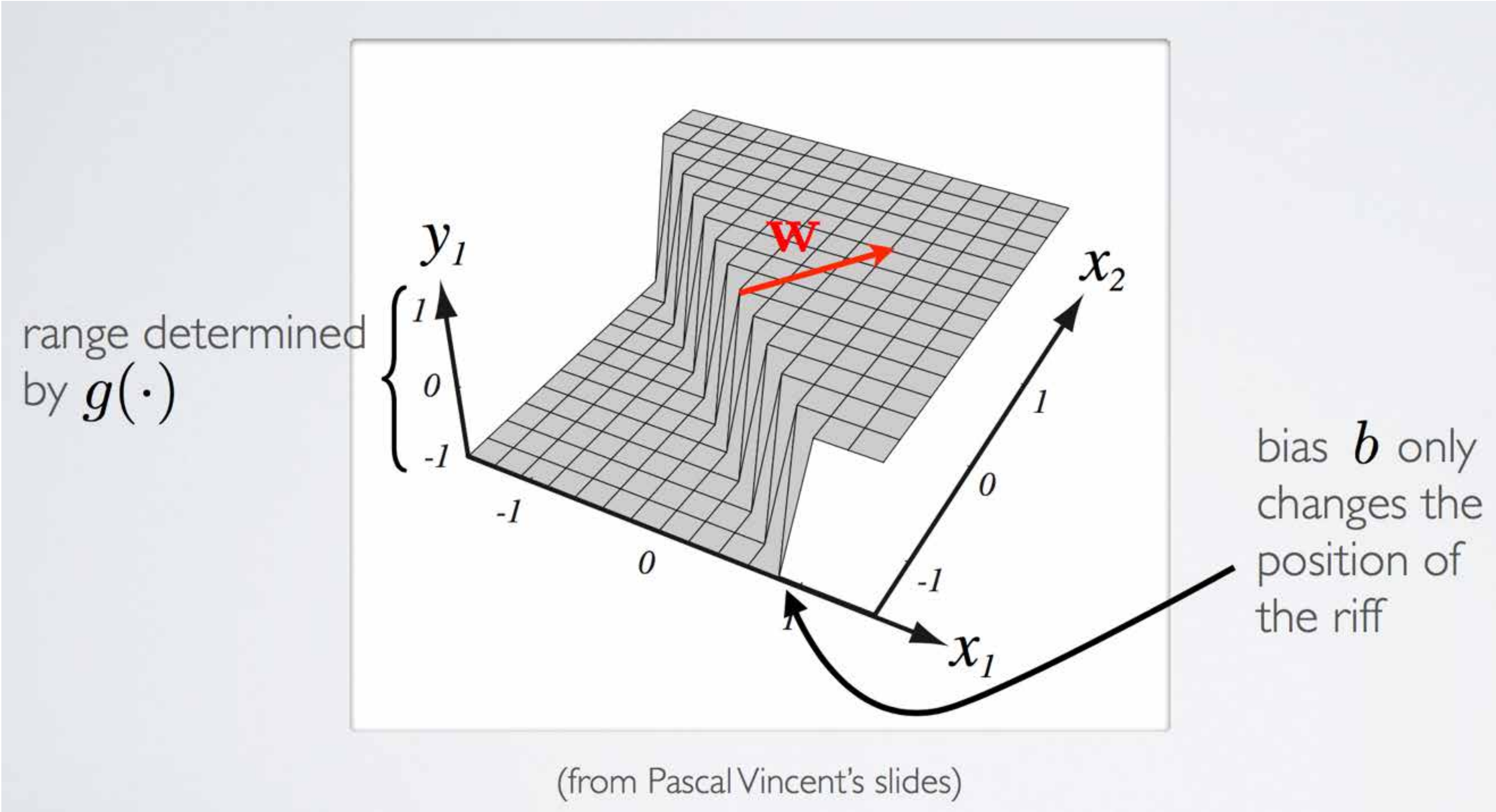
Compiled from various resources

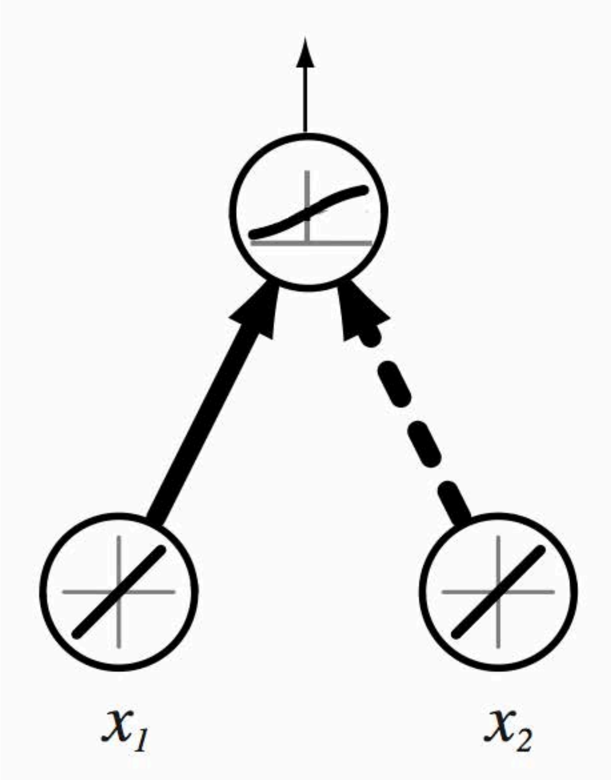
Recall: ML and AI vs Deep learning



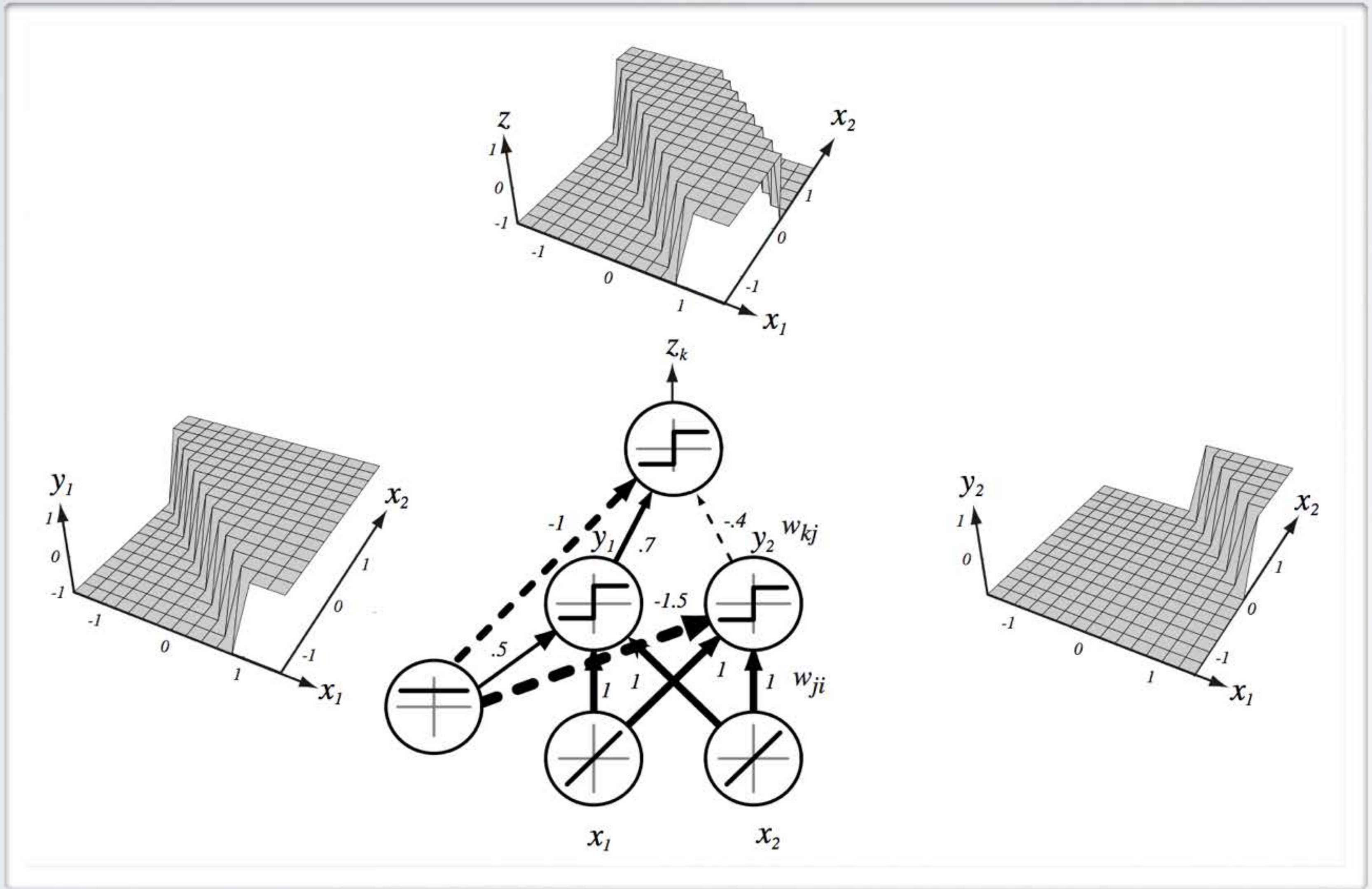
Goodfellow, 2016

Recall: Capacity of a Neuron



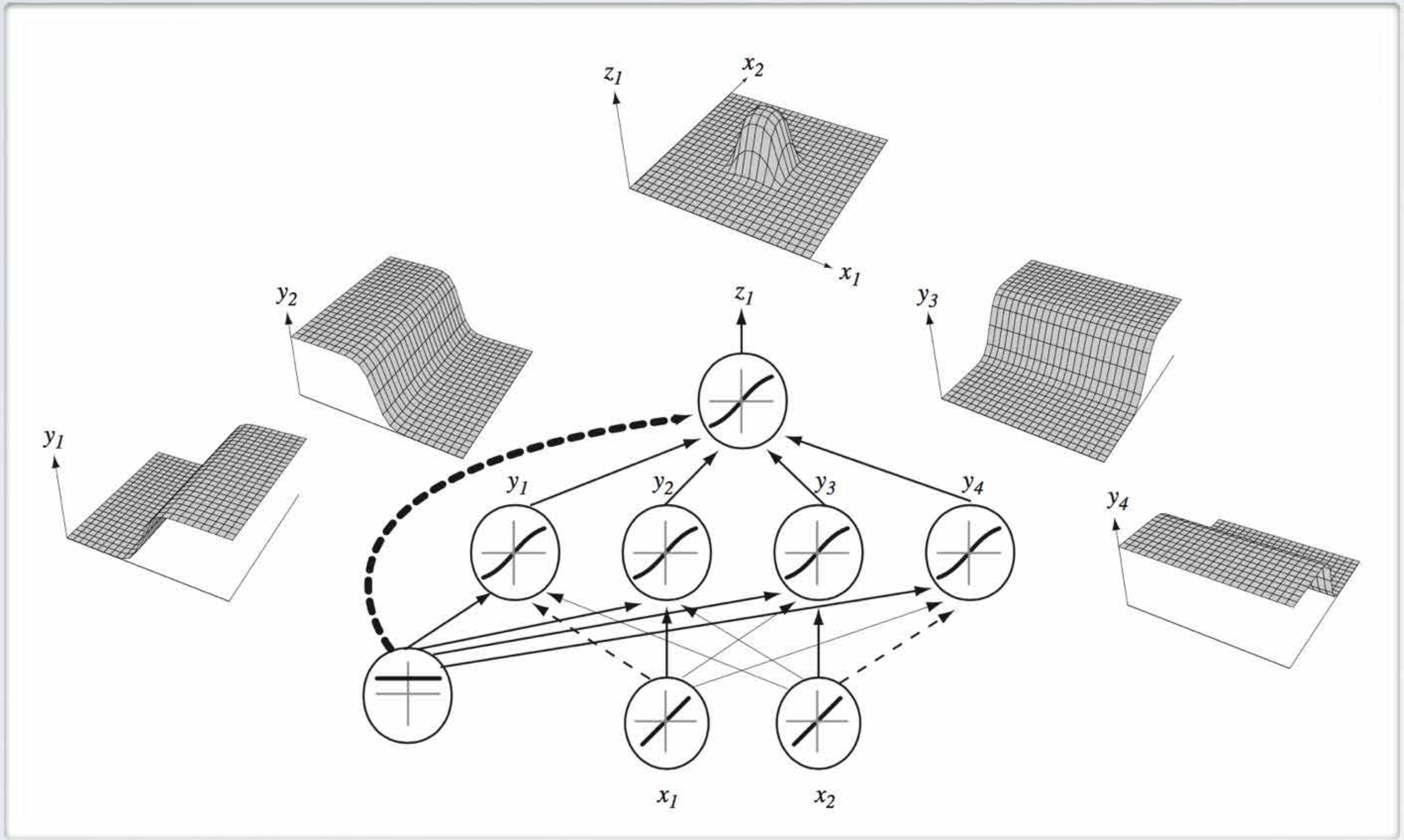


Recall: Capacity of Neural Network



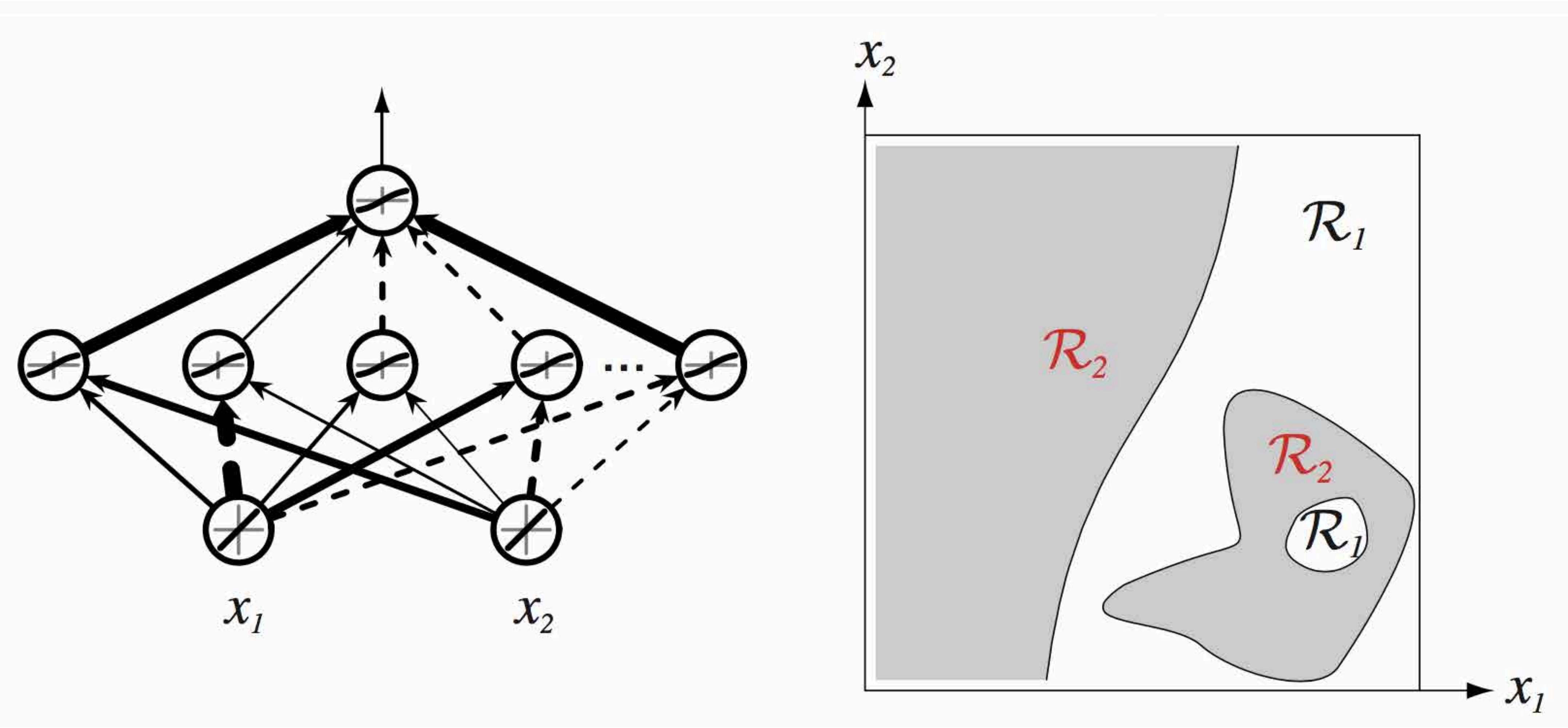
(from Pascal Vincent's slides)

Recall: Capacity of Neural Network



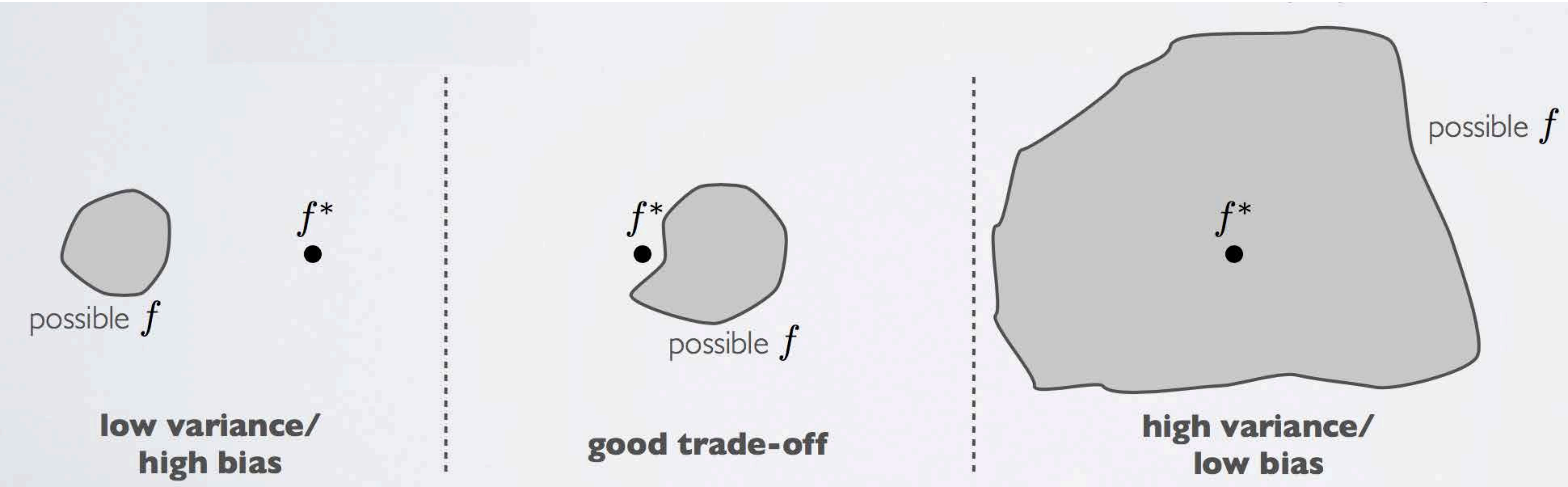
(from Pascal Vincent's slides)

Recall: Capacity of Neural Network



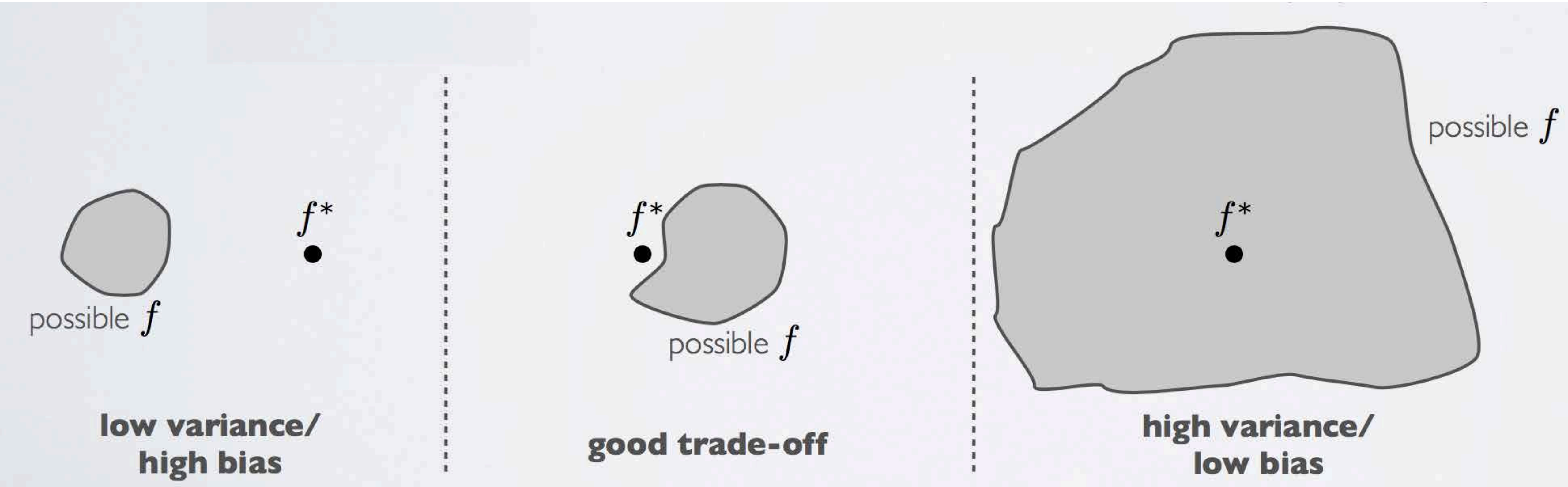
Bias – Variance Trade-off

- **Variance of trained model:** does it vary a lot if the training set changes?
- **Bias of trained model:** the average model close to the true solution
- **Generalization error:** (can be seen as) the sum of the (squared) bias and variance



Training Problem: Overfitting

- **Variance of trained model:** does it vary a lot if the training set changes?
- **Bias of trained model:** the average model close to the true solution
- **Generalization error:** (can be seen as) the sum of the (squared) bias and variance
    - **Might be in high variance/low bias situation**



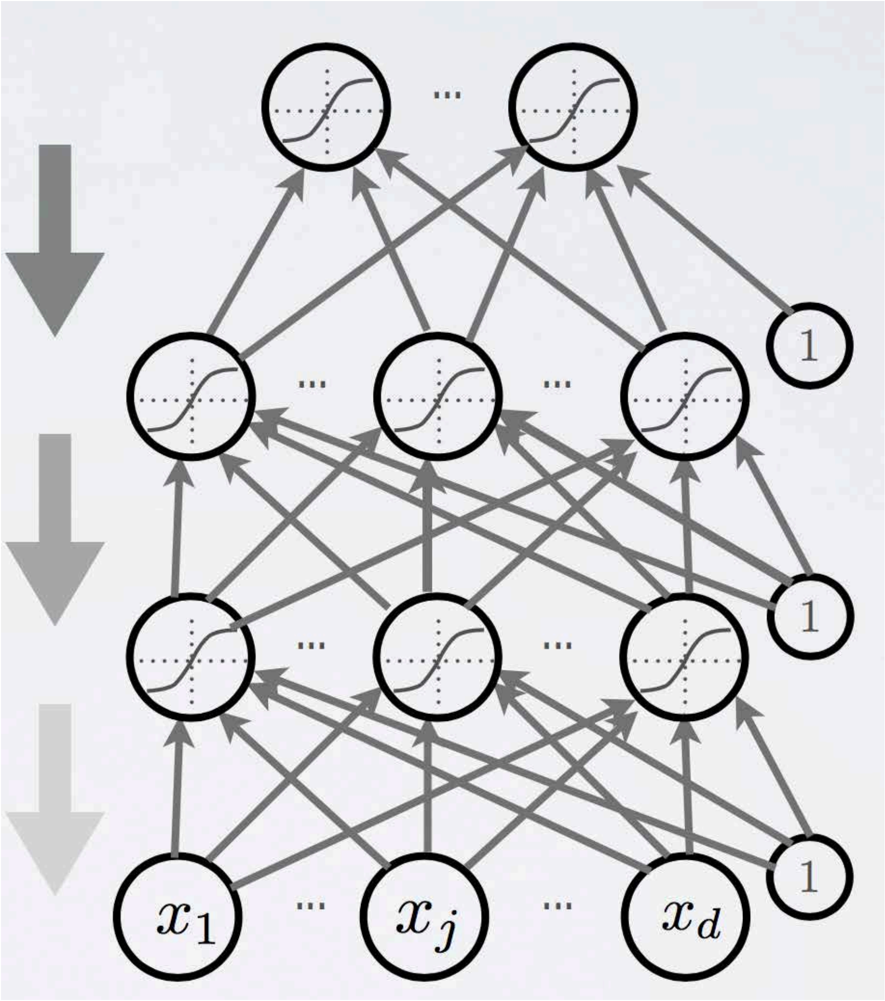
Standard Neural Network Training Demo

- http://cs.stanford.edu/people/karpathy/svmjs/demo/demonn.html

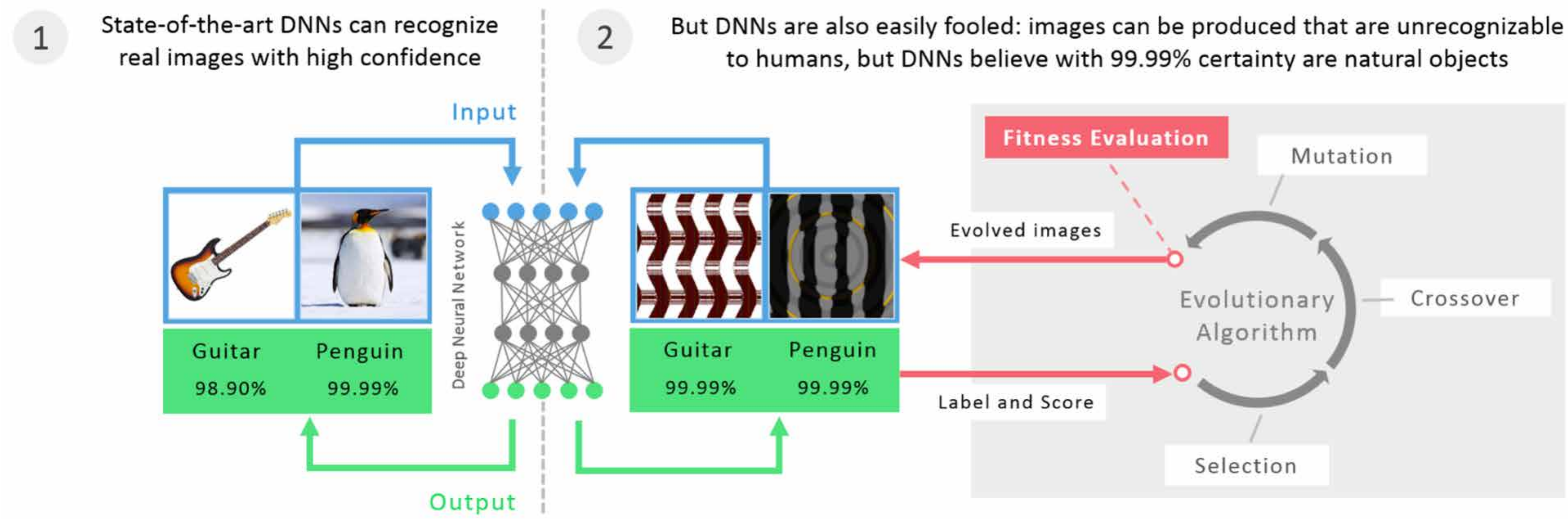
- We have lots of problems to tackle with during NN training
- Some of them are:
  - Vanishing gradient
  - Adversarial Samples
  - High variance / low bias networks (generally related to overfitting)

Training Problem: Vanishing Gradient Problem

- Saturated units block gradient propagation.
- This is especially true for recurrent neural networks. Why?
- Solution?



Problem: Adversarial Examples

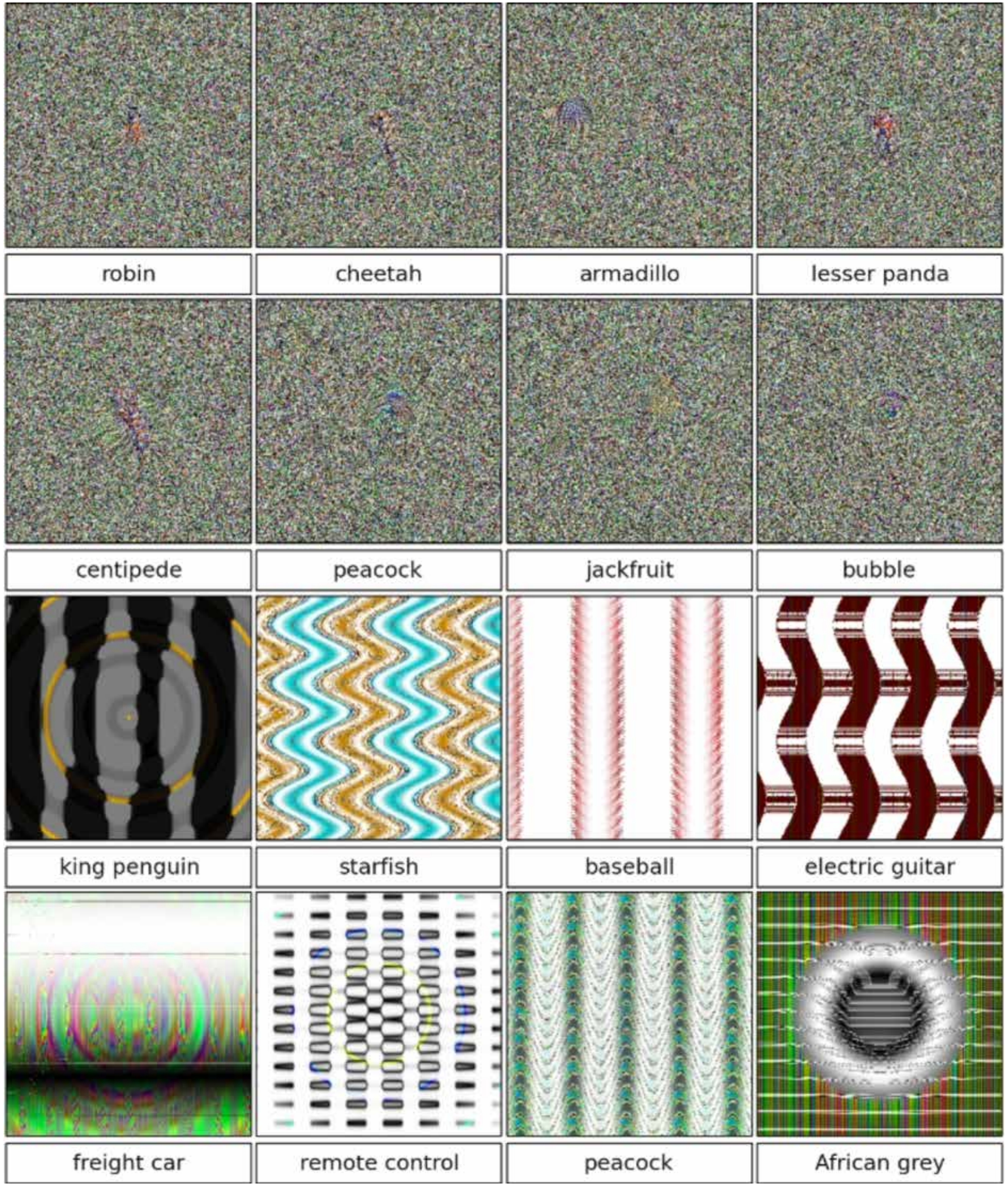


- Although state-of-the-art DNNs can increasingly recognize natural images, they also are easily fooled into declaring with near-certainty that unrecognizable images are familiar objects.
- Images that fool DNNs are produced by evolutionary algorithms that optimize images to generate high-confidence DNN predictions for each class in the dataset the DNN is trained on (here, ImageNet).

Problem: Adversarial Examples

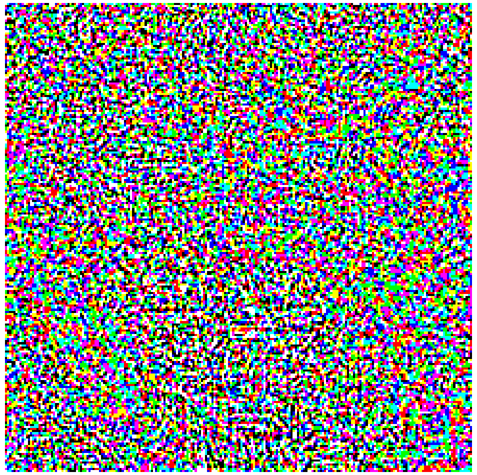
- Evolved images that are unrecognizable to humans but that state-of-the-art DNNs trained on ImageNet believe with ≥ 99.6% certainty to be a familiar object.

- This result highlights differences between how DNNs and humans recognize objects.



Problem: Adversarial Examples

$+\ .007 \times$



$=$

$\boldsymbol{x}$

$y =$"panda"
w/ 57.7%
confidence

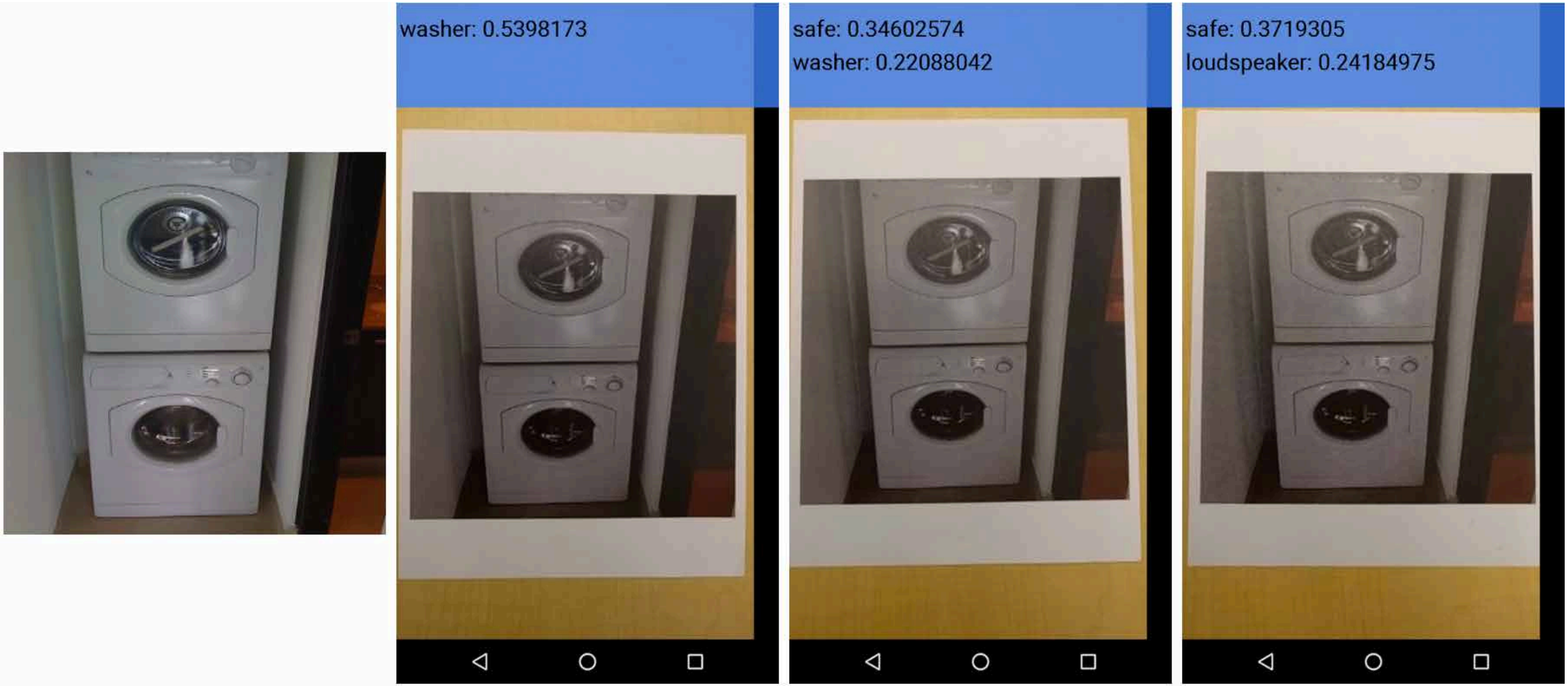
$\mathrm{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
w/ 8.2%
confidence

$\boldsymbol{x} + \epsilon\, \mathrm{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
w/ 99.3 %
confidence

Problem: Adversarial Examples in Real World



(a) Image from dataset (b) Clean image (c) Adv. image, $\epsilon = 4$ (d) Adv. image, $\epsilon = 8$

Overfitting

- To overcome overfitting:
  - Pre-training (unsupervised learning):
    Auto-encoders, Stacked auto-encoders, Restricted Boltzmann Machines, etc…
  - We can use local receptive fields and shared weights, which makes the neural network a Convolutional Neural Network.
  - We can corrupt inputs (Denoising Auto-encoders)
  - We can use regularization (L1, L2, dropout, augmentation)

Regularization

- Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.

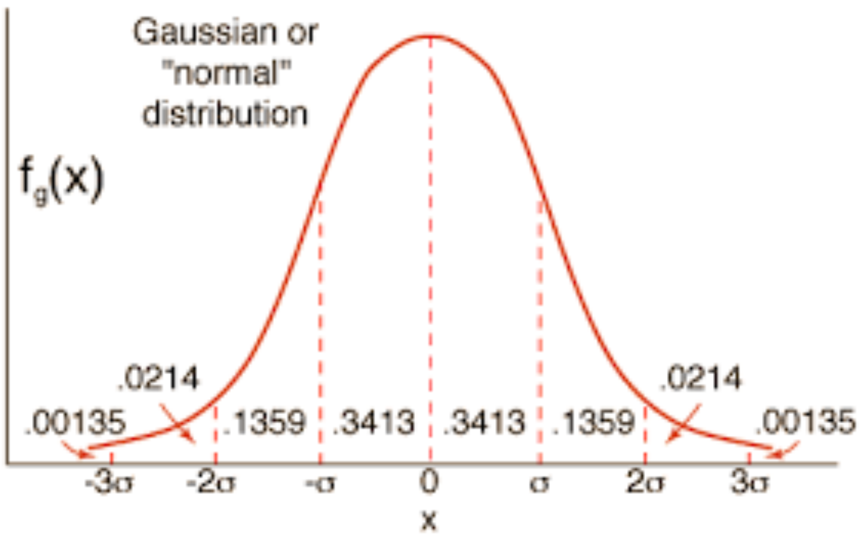
- L2 regularization
- L1 regularization
- Dropout
- Data augmentation

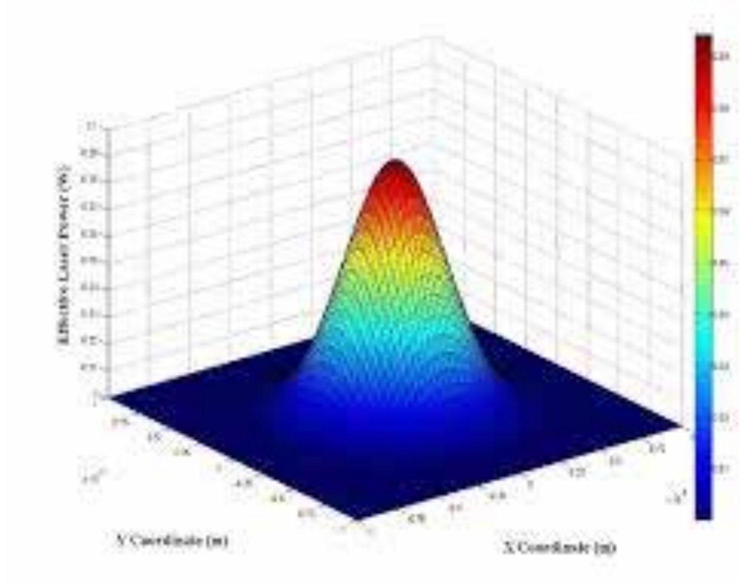
Regularization: L2 Regularization

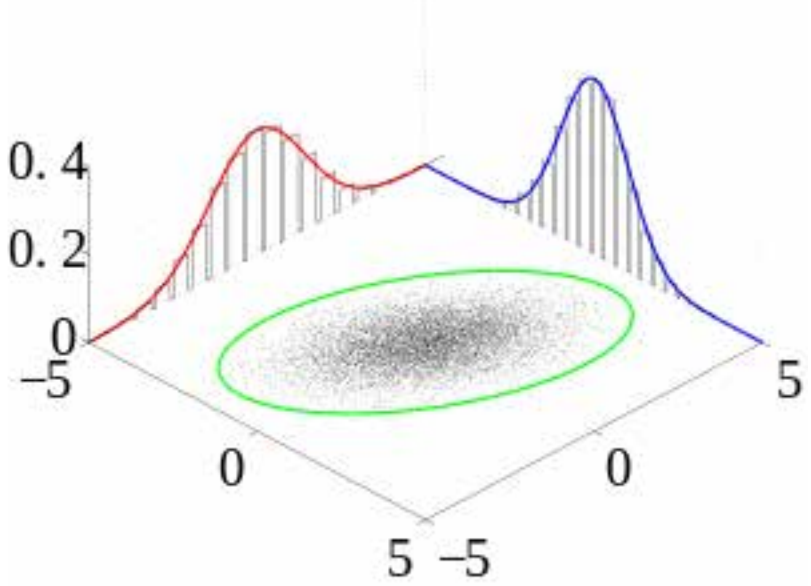
- Penalizes the gradient term in back-prop algorithm:

$$\Omega(\boldsymbol{\theta}) = \sum_k \sum_i \sum_j \left(W_{i,j}^{(k)}\right)^2 = \sum_k ||\mathbf{W}^{(k)}||_F^2$$

- Gradient: $\nabla_{\mathbf{W}^{(k)}} \Omega(\boldsymbol{\theta}) = 2\mathbf{W}^{(k)}$
    - Only applied on weights, not biases
    - Can be interpreted as having a Gaussian Prior over weight values.





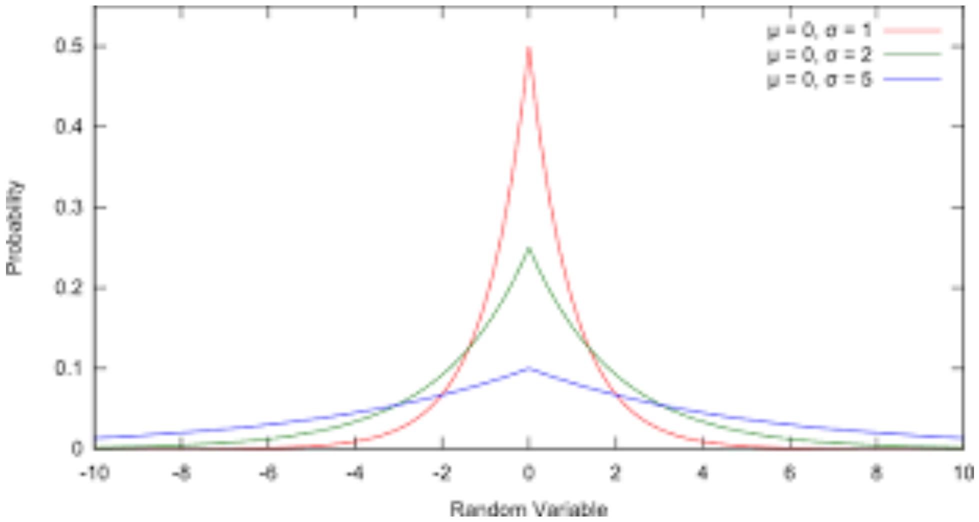


Regularization: L1 Regularization

- Penalizes the gradient term in back-prop algorithm:

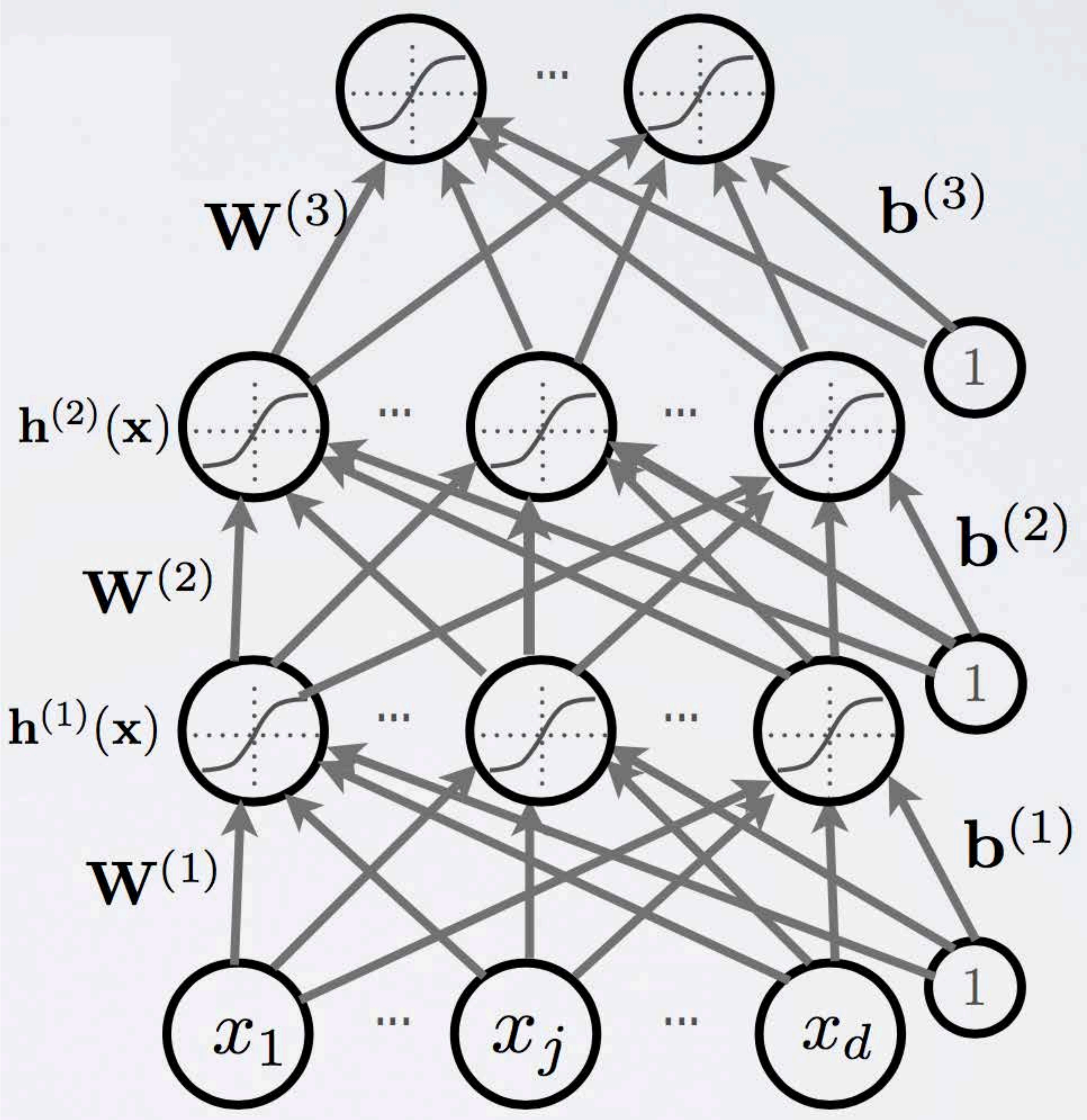
$$\Omega(\boldsymbol{\theta}) = \sum_k \sum_i \sum_j |W_{i,j}^{(k)}|$$

- Gradient: $\nabla_{\mathbf{W}^{(k)}} \Omega(\boldsymbol{\theta}) = \mathrm{sign}(\mathbf{W}^{(k)})$
  - Only applied on weights, not biases (again)
  - Unlike L2, L1 regularization will make some of the weights exactly 0.
  - Can be interpreted as having a Laplacian Prior over weight values.



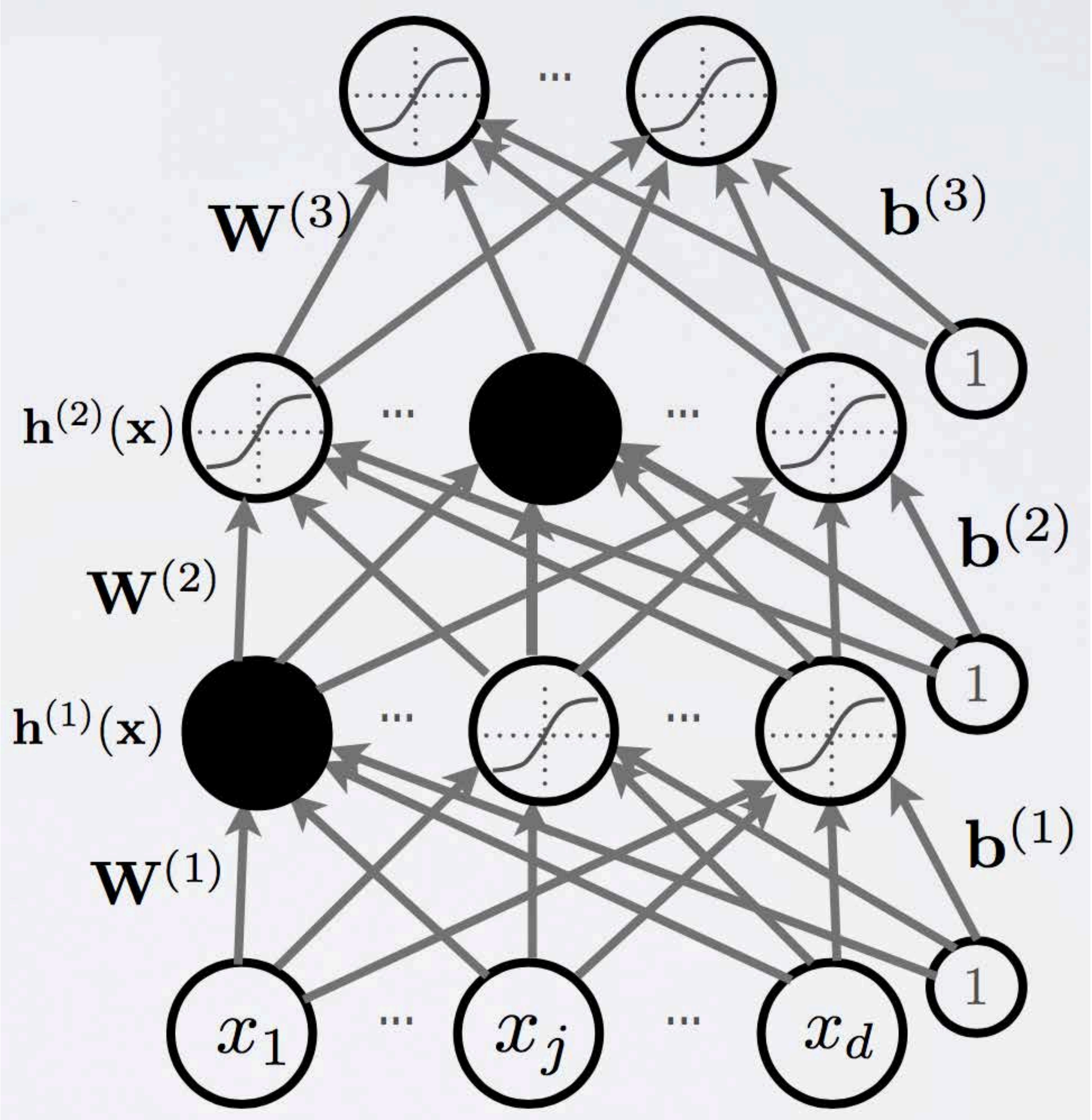
Regularization: Dropout

- Idea: Cripple neural network by removing hidden units randomly.
  - Each hidden unit's activation is set to 0 with a probability, independently.
  - Usually 0.5 works well.



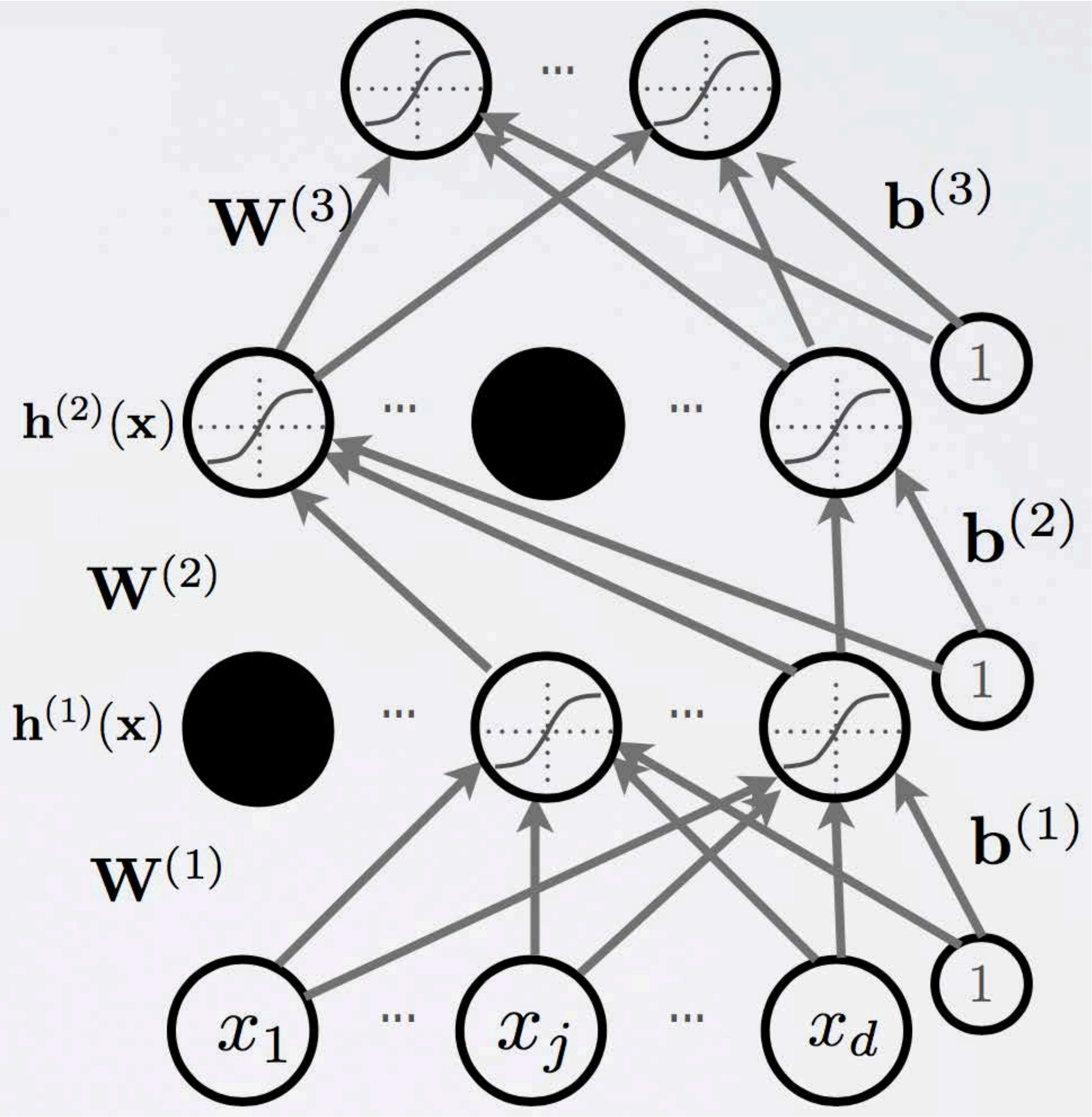
Regularization: Dropout

- Idea: Cripple neural network by removing hidden units randomly.
  - Each hidden unit's activation is set to 0 with a probability, independently.
  - Usually 0.5 works well.



Regularization: Dropout

- Idea: Cripple neural network by removing hidden units randomly.
  - Hidden units cannot cooperate with each other in a specific layer.
  - Therefore, hidden units must be more generally useful.

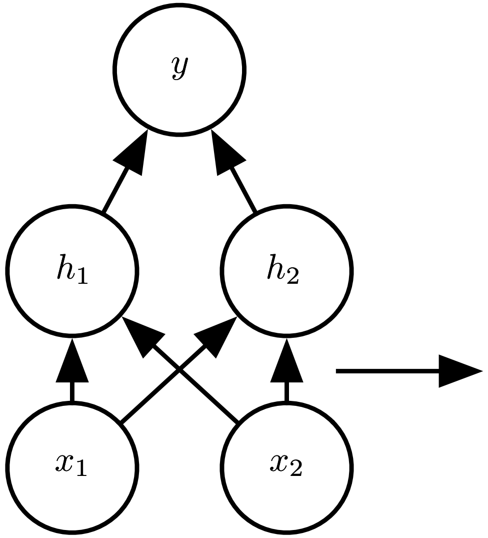


Regularization: Dropout

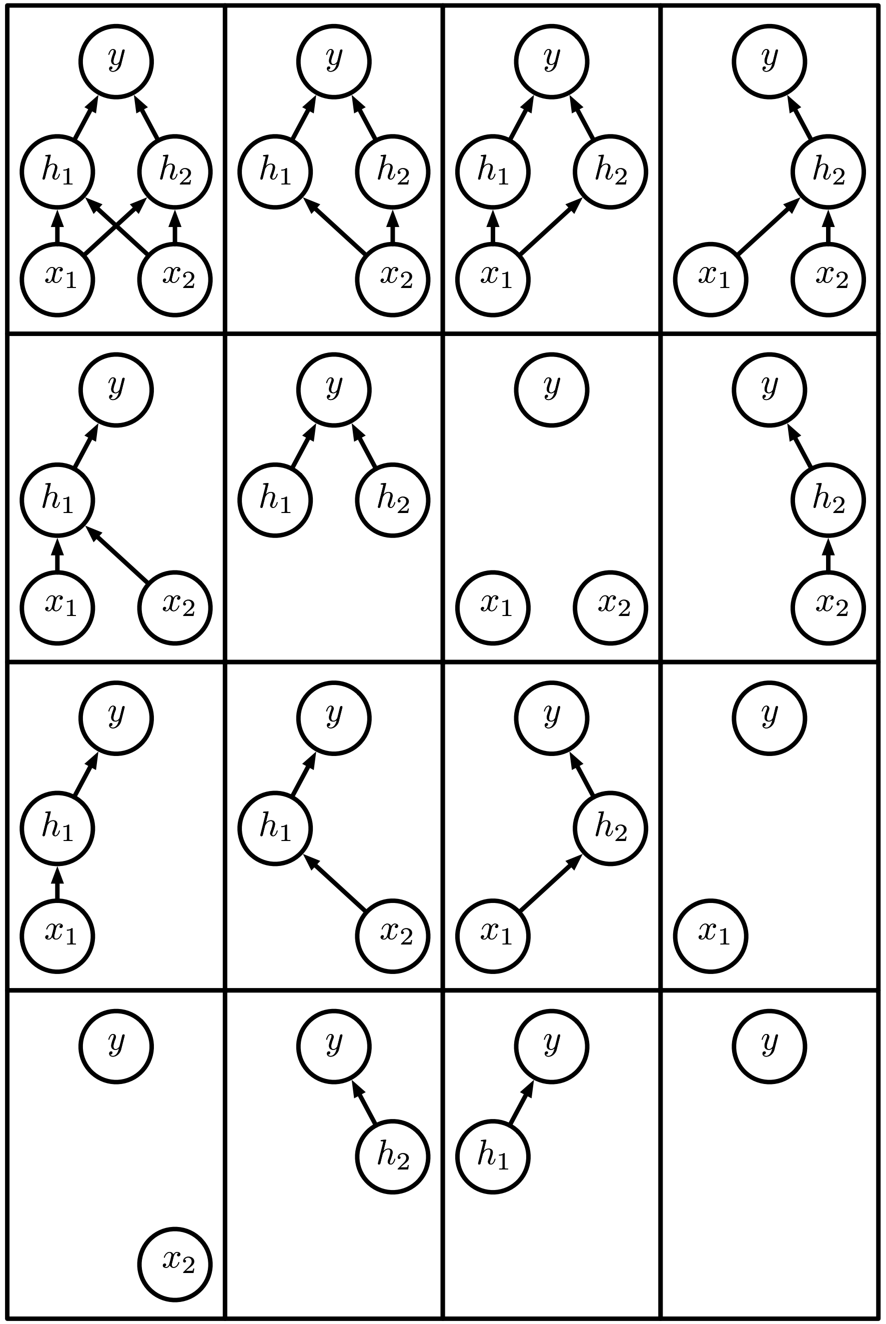
- At test time, we replace the masks by their expectation
  - this is simply the constant vector 0.5 if dropout probability is 0.5
  - for single hidden layer, can show this is equivalent to taking the geometric average of all neural networks, with all possible binary masks

- Can be combined with unsupervised pre-training
- Beats regular backpropagation on many datasets

Regularization: Dropout

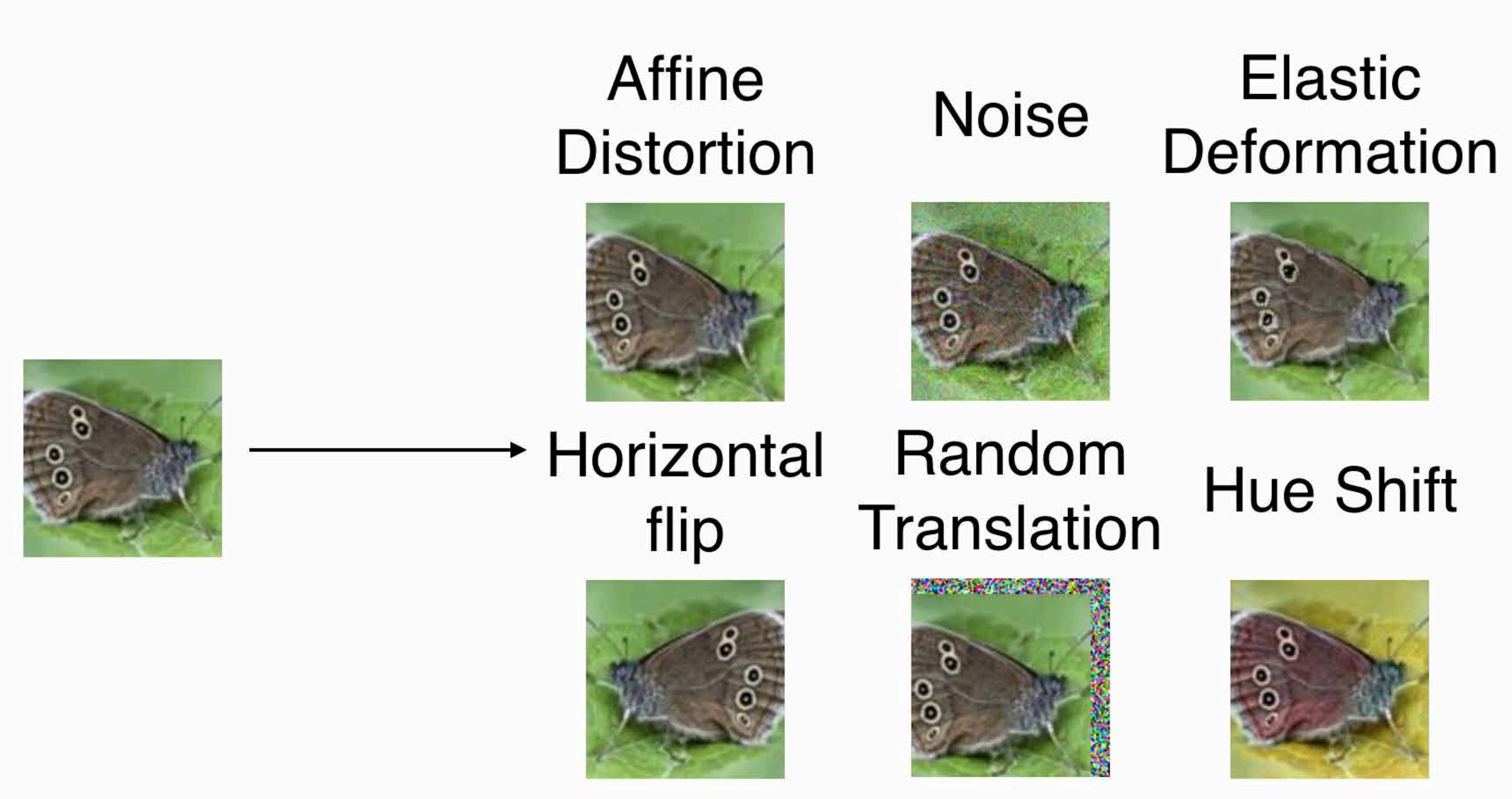


Base network

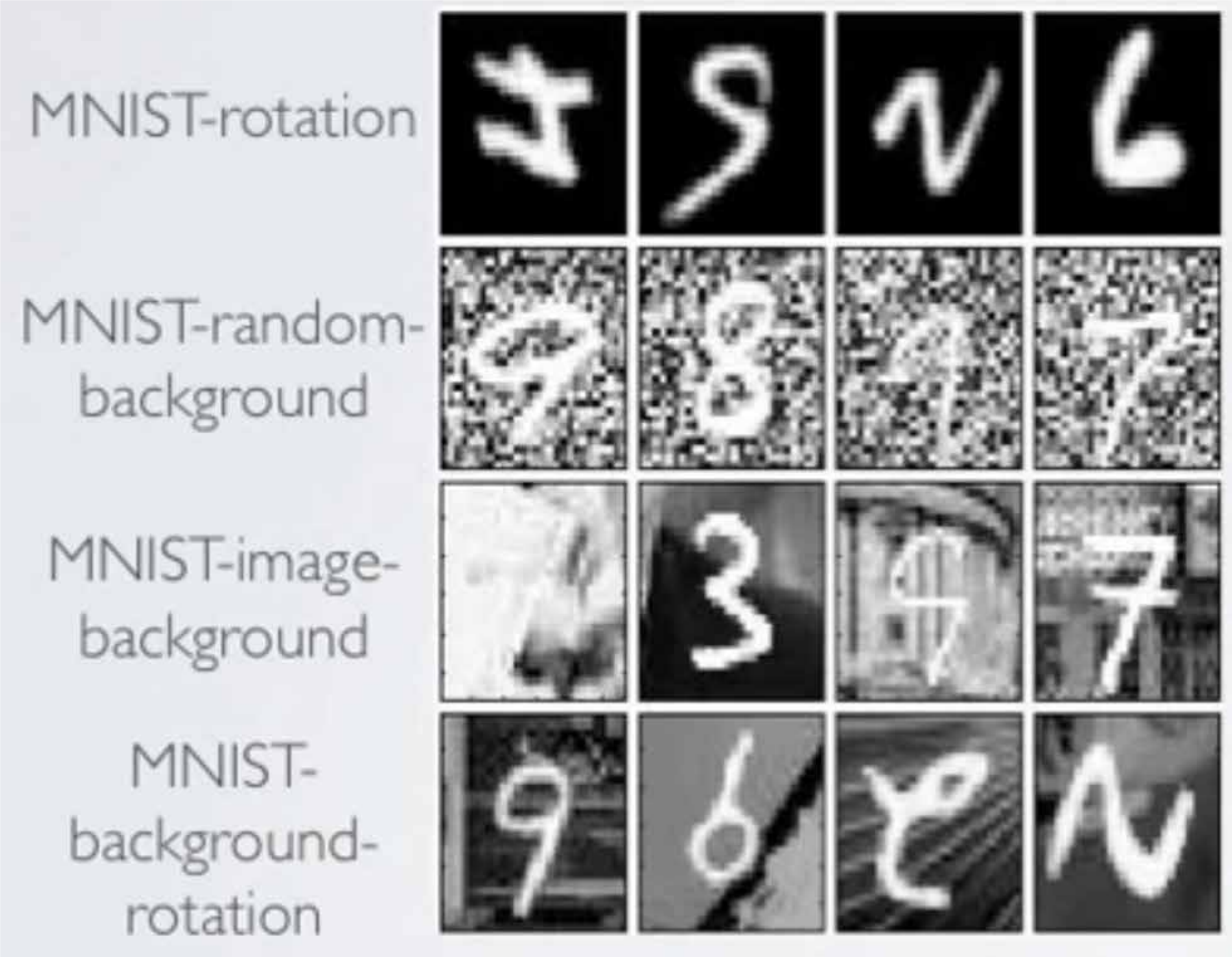


Ensemble of subnetworks

Regularization: Dataset Augmentation



Possible Variations on MNIST



Impact of Pre-training
(intentional underfitting scenario)

Network		MNIST-small classif. test error	MNIST-rotation classif. test error
Type	Depth		
Deep net	1	**4.14** % ± 0.17	15.22 % ± 0.31
	2	**4.03** % ± 0.17	**10.63** % ± 0.27
	3	**4.24** % ± 0.18	11.98 % ± 0.28
	4	4.47 % ± 0.18	11.73 % ± 0.29
Deep net + autoencoder	1	3.87 % ± 0.17	11.43% ± 0.28
	2	**3.38** % ± 0.16	9.88 % ± 0.26
	3	**3.37** % ± 0.16	**9.22** % ± 0.25
	4	**3.39** % ± 0.16	**9.20** % ± 0.25
Deep net + RBM	1	3.17 % ± 0.15	10.47 % ± 0.27
	2	**2.74** % ± 0.14	9.54 % ± 0.26
	3	**2.71** % ± 0.14	**8.80** % ± 0.25
	4	**2.72** % ± 0.14	**8.83** % ± 0.24

Performance on Different Datasets

		Stacked Autoencoders	Stacked RBMS	Stacked Denoising Autoencoders
Dataset	**SVM$_{rbf}$**	**SAA-3**	**DBN-3**	**SdA-3** (ν)
basic	**3.03±0.15**	3.46±0.16	3.11±0.15	**2.80±0.14** (10%)
rot	11.11±0.28	**10.30±0.27**	**10.30±0.27**	**10.29±0.27** (10%)
bg-rand	14.58±0.31	11.28±0.28	**6.73±0.22**	10.38±0.27 (40%)
bg-img	22.61±0.37	23.00±0.37	**16.31±0.32**	**16.68±0.33** (25%)
rot-bg-img	55.18±0.44	51.93±0.44	47.39±0.44	**44.49±0.44** (25%)
rect	**2.15±0.13**	2.41±0.13	2.60±0.14	**1.99±0.12** (10%)
rect-img	24.04±0.37	24.05±0.37	22.50±0.37	**21.59±0.36** (25%)
convex	19.13±0.34	**18.41±0.34**	**18.63±0.34**	**19.06±0.34** (10%)

Extracting and Composing Robust Features with Denoising Autoencoders, Vincent, Larochelle, Bengio and Manzagol, 2008.