Stacking and Pre-training

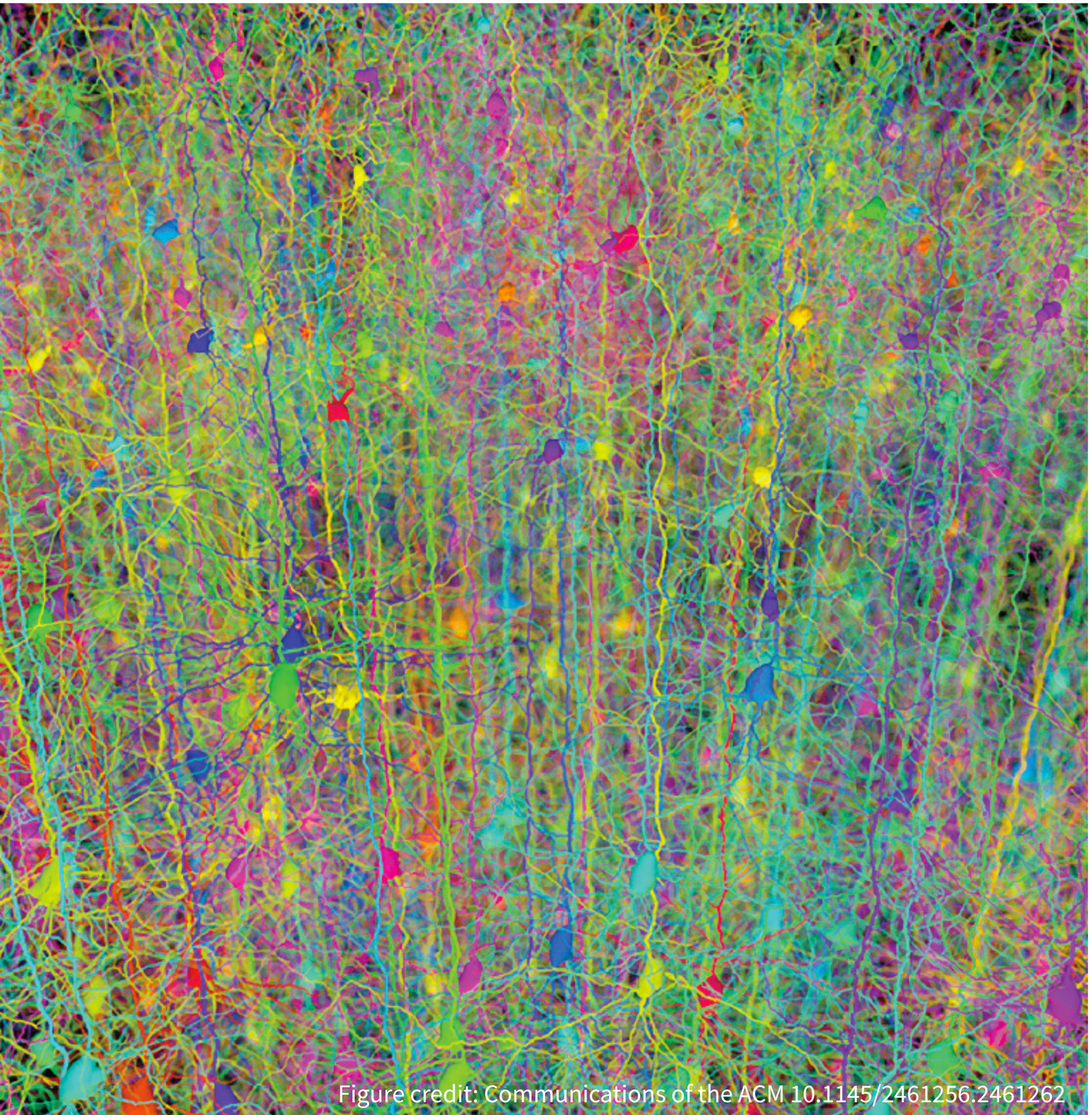


Figure credit: Communications of the ACM 10.1145/2461256.2461262

GRAHAM TAYLOR

VECTOR INSTITUTE

SCHOOL OF ENGINEERING
UNIVERSITY OF GUELPH

CANADIAN INSTITUTE
FOR ADVANCED RESEARCH

UNIVERSITY of GUELPH
CHANGING LIVES
IMPROVING LIFE

CIFAR
CANADIAN
INSTITUTE
FOR
ADVANCED
RESEARCH

Stacking to Build Deep Models

- Greedy layer-wise training can be used to build deep models
- It is most popular to use RBMs, but other architectures (regularized autoencoders, ICA, even k-means) can be stacked

Unsupervised Pre-Training

Initialize hidden layers using unsupervised learning

- forces the network to represent the latent structure of the input distribution
- encourages hidden layers to encode that structure
- it is a much harder task than supervised learning; hence we expect less overfitting
- extremely popular practice 2006-2012 (now supervised pre-training more popular); however still a valuable technique

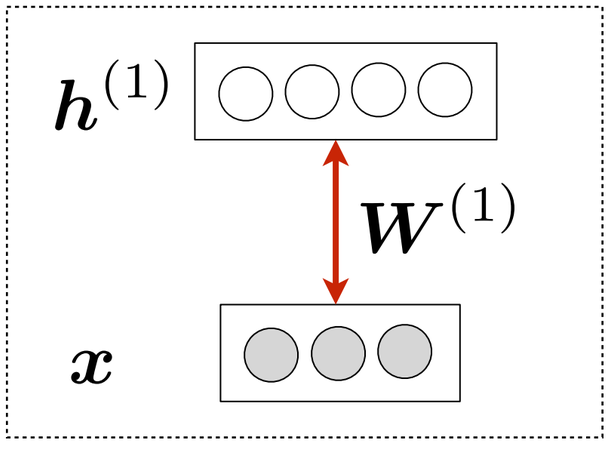
Greedy Layer-wise Procedure

- Train one layer at a time, from first to last, with unsupervised criterion
- Fix the parameters of previous hidden layers
- Previous layers viewed as feature extraction

Reproduced from Hugo Larochelle's slides

Greedy Layer-wise Procedure

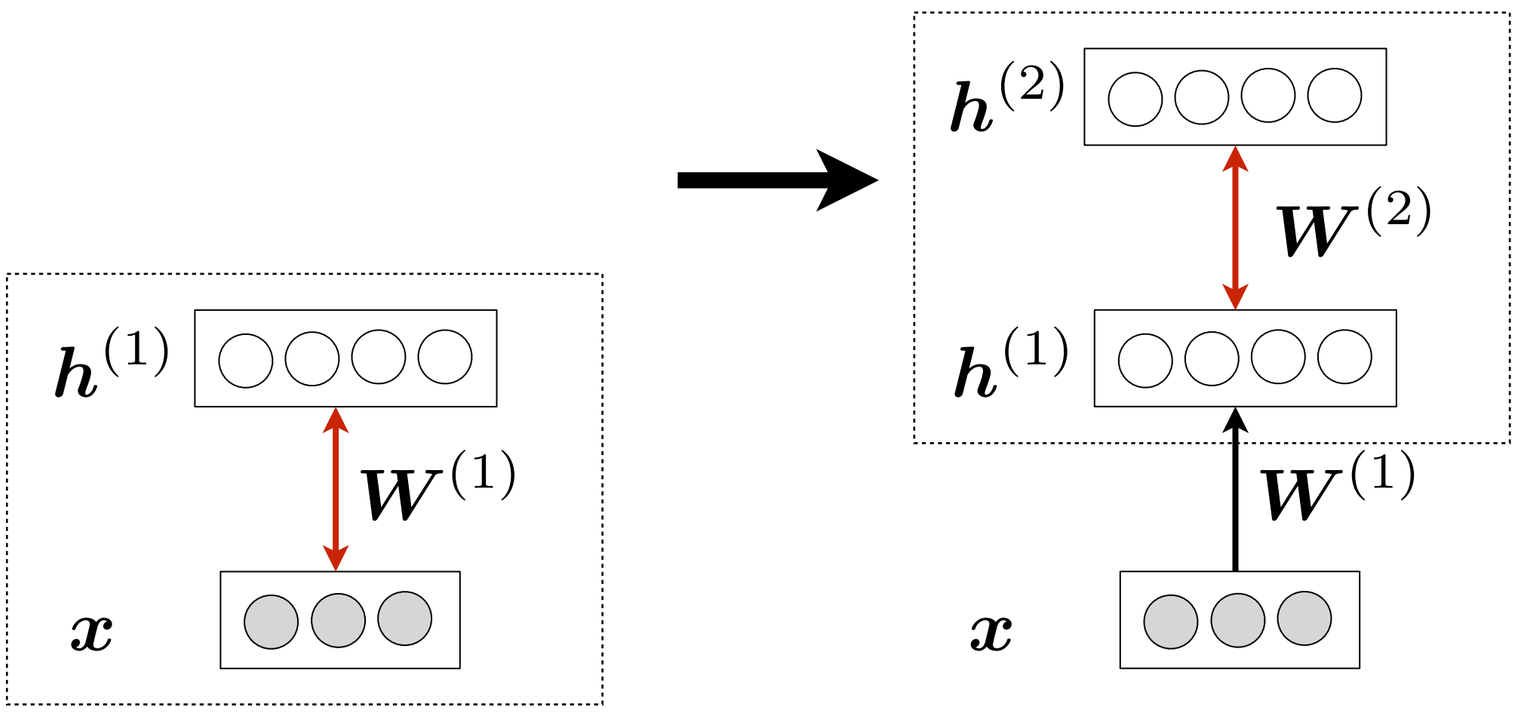
- Train one layer at a time, from first to last, with unsupervised criterion
- Fix the parameters of previous hidden layers
- Previous layers viewed as feature extraction



Reproduced from Hugo Larochelle's slides

Greedy Layer-wise Procedure

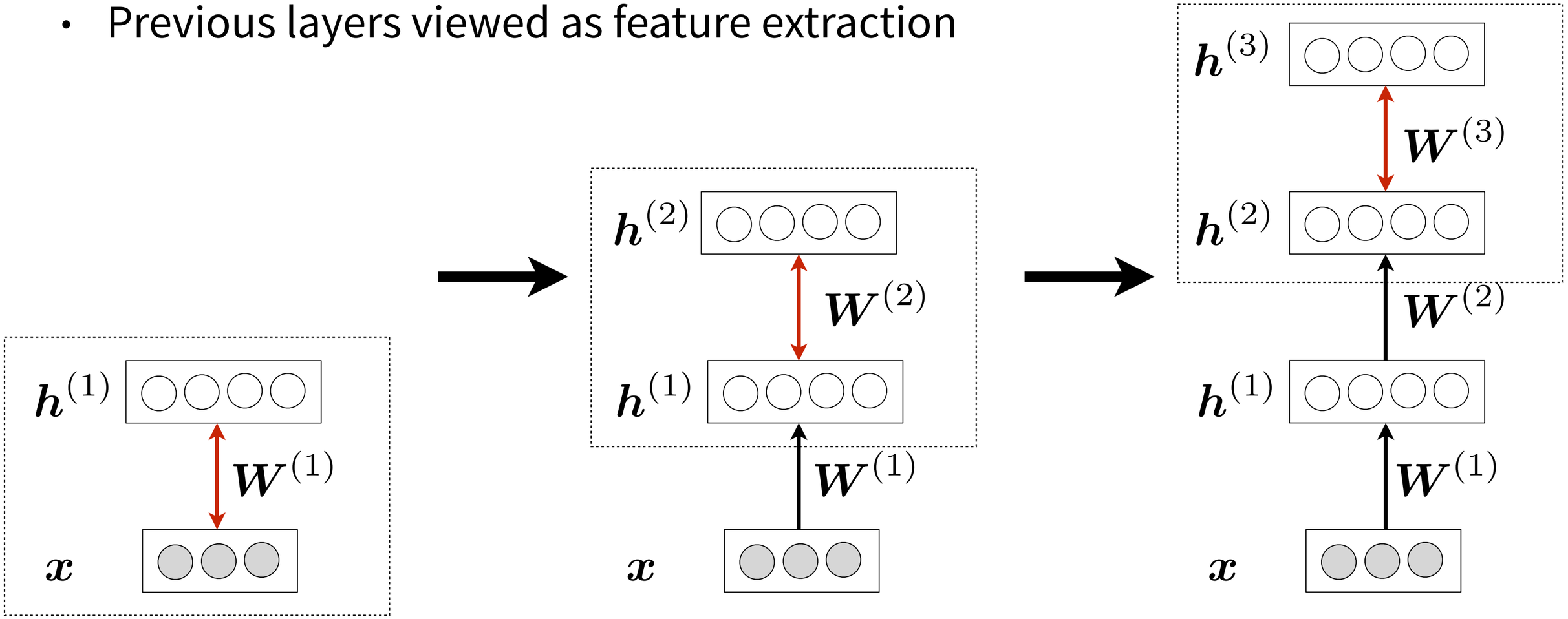
- Train one layer at a time, from first to last, with unsupervised criterion
- Fix the parameters of previous hidden layers
- Previous layers viewed as feature extraction



Reproduced from Hugo Larochelle's slides

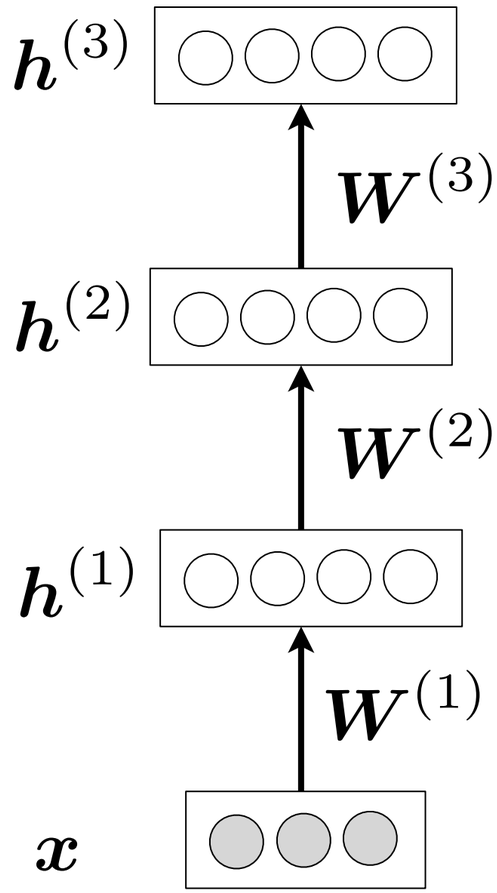
Greedy Layer-wise Procedure

- Train one layer at a time, from first to last, with unsupervised criterion
- Fix the parameters of previous hidden layers
- Previous layers viewed as feature extraction



Reproduced from Hugo Larochelle's slides

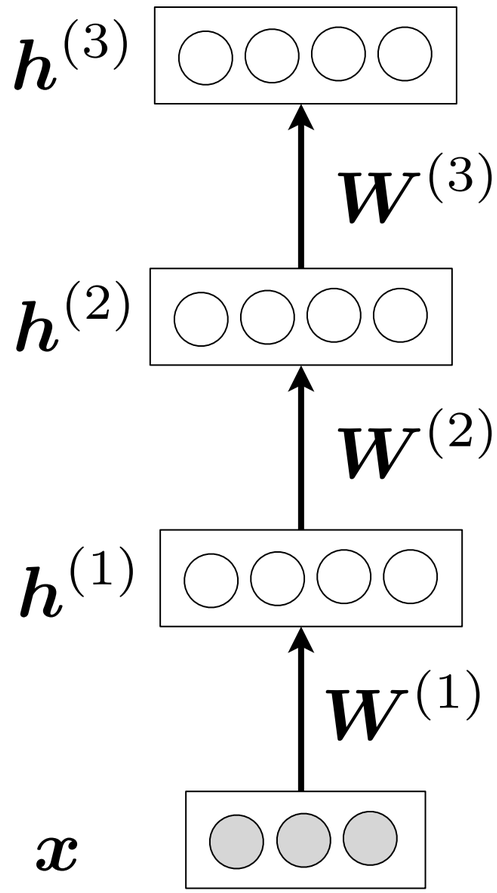
Fine-Tuning



Reproduced from Hugo Larochelle's slides

Fine-Tuning

Once all layers are pre-trained

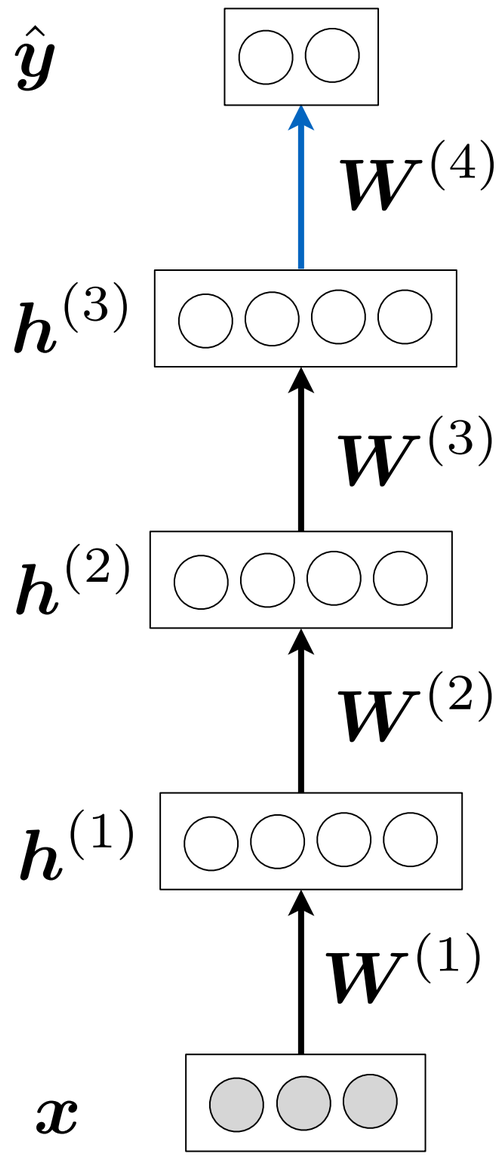


Reproduced from Hugo Larochelle's slides

Fine-Tuning

Once all layers are pre-trained

- add an output layer

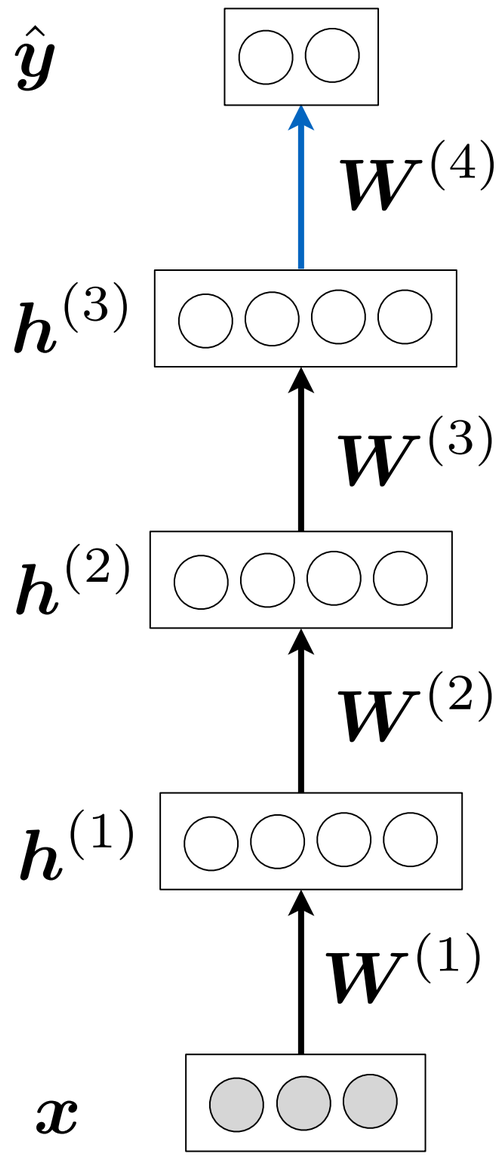


Reproduced from Hugo Larochelle's slides

Fine-Tuning

Once all layers are pre-trained

- add an output layer
- train the whole network using supervised learning



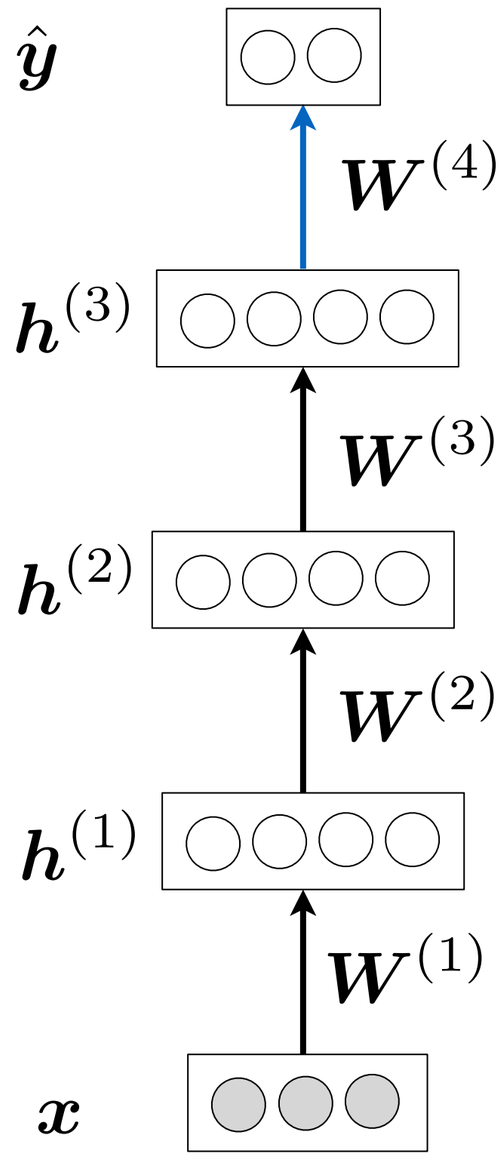
Reproduced from Hugo Larochelle's slides

Fine-Tuning

Once all layers are pre-trained

- add an output layer
- train the whole network using supervised learning

Supervised learning is performed as a regular feed-forward network



Reproduced from Hugo Larochelle's slides

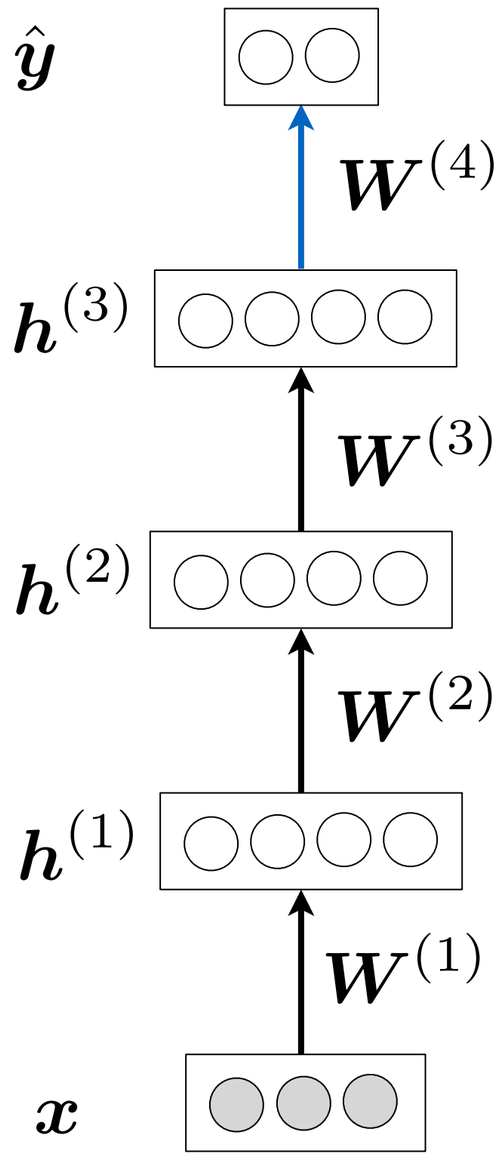
Fine-Tuning

Once all layers are pre-trained

- add an output layer
- train the whole network using supervised learning

Supervised learning is performed as a regular feed-forward network

We call this last phase fine-tuning



Reproduced from Hugo Larochelle's slides

Fine-Tuning

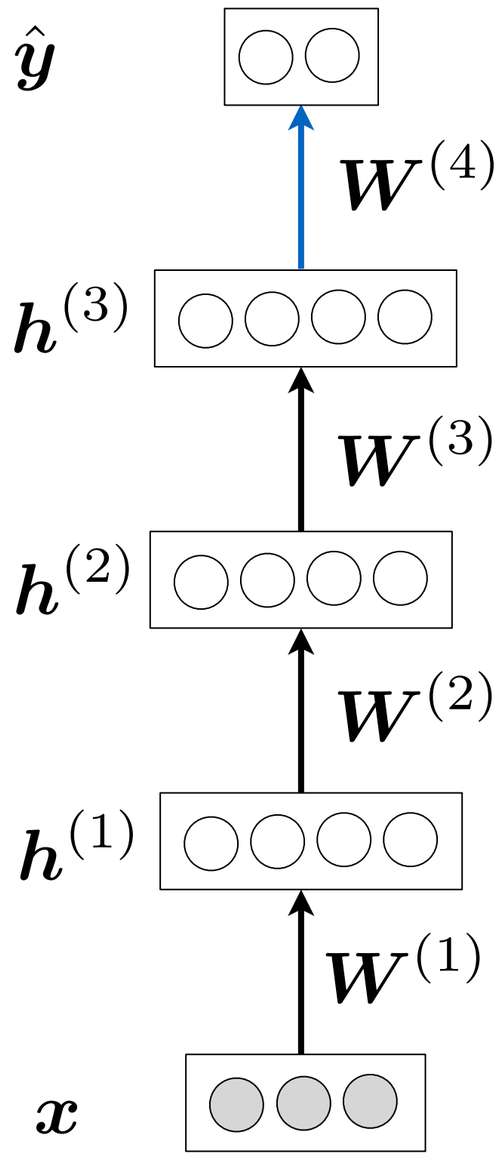
Once all layers are pre-trained

- add an output layer
- train the whole network using supervised learning

Supervised learning is performed as a regular feed-forward network

We call this last phase fine-tuning

- all parameters tuned for the task at hand



Reproduced from Hugo Larochelle's slides

Fine-Tuning

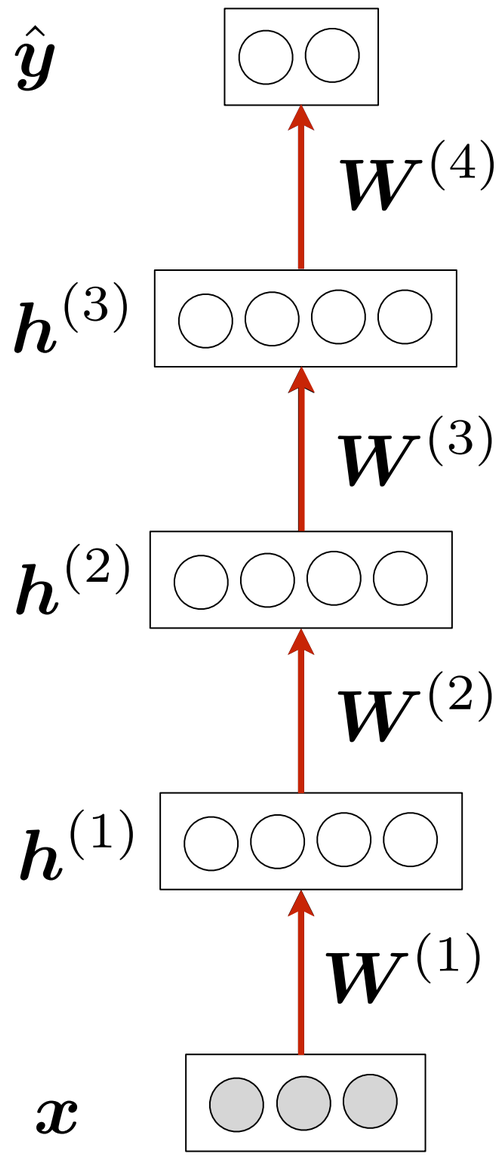
Once all layers are pre-trained

- add an output layer
- train the whole network using supervised learning

Supervised learning is performed as a regular feed-forward network

We call this last phase fine-tuning

- all parameters tuned for the task at hand
- representation adjusted to be more discriminative



Reproduced from Hugo Larochelle's slides

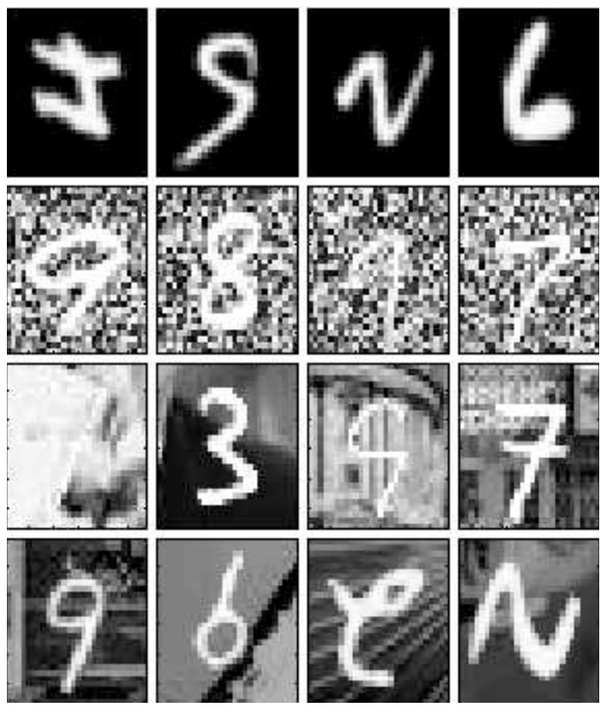
Example: Pre-Training (Data)

MNIST-rotation

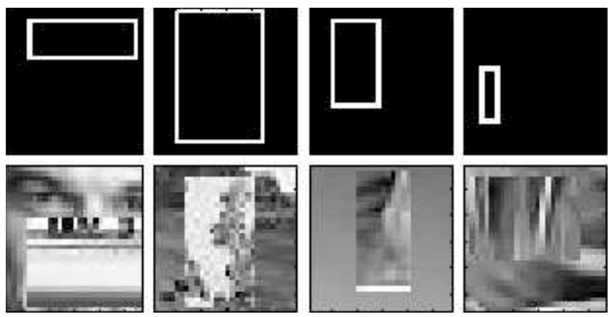
MNIST-random-background

MNIST-image-background

MNIST-background-rotation



Variations on MNIST



Tall or wide?

Convex shape or not?

An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation
Larochelle, Erhan, Courville, Bergstra and Bengio, 2007

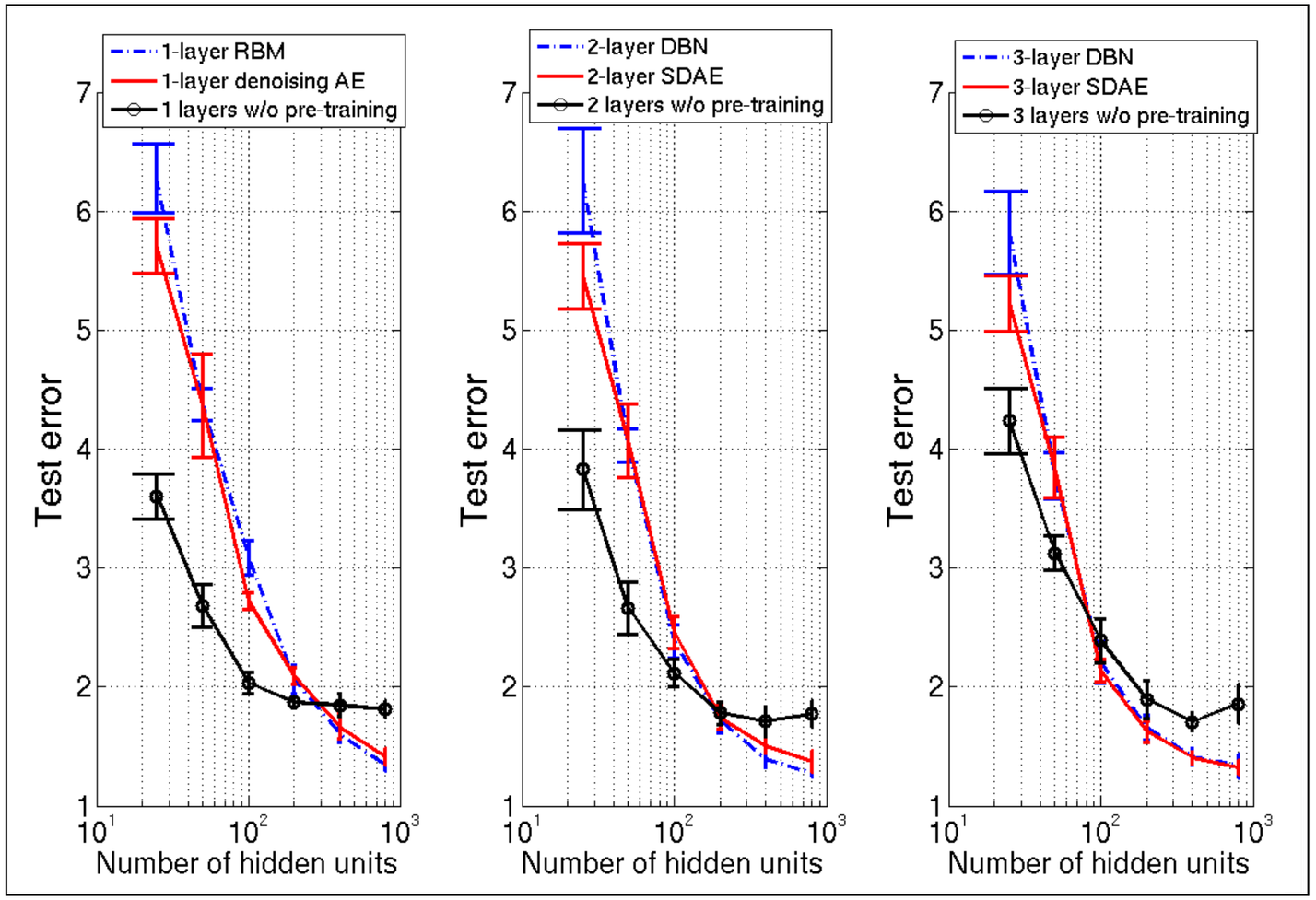
Reproduced from Hugo Larochelle's slides

Impact of Initialization (1)

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

Reproduced from Hugo Larochelle's slides

Impact of Initialization (2)



Pre-training acts as a regularizer:

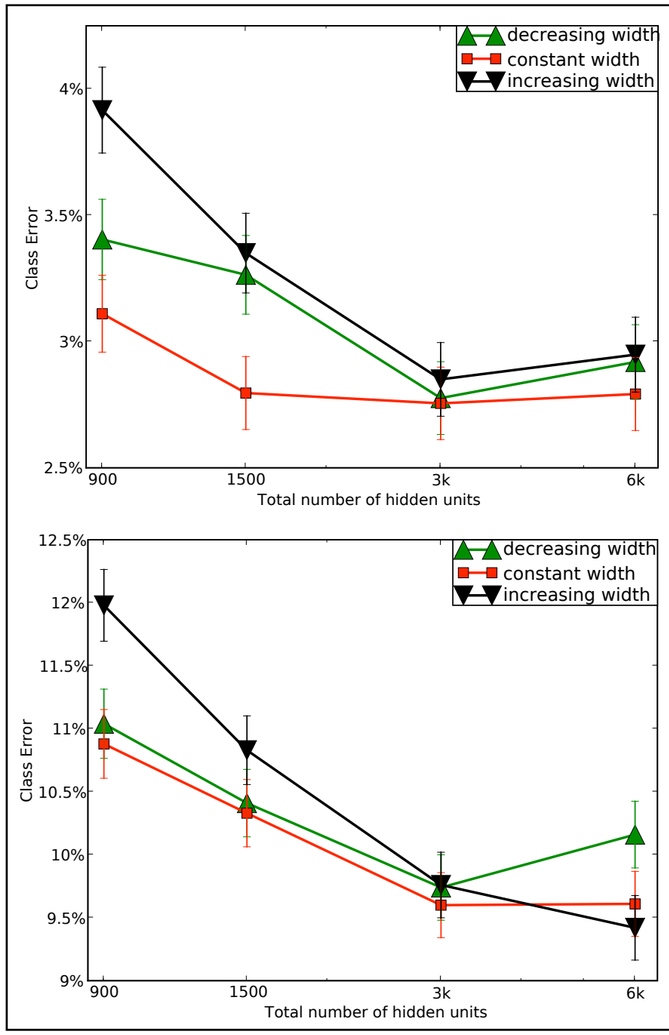
- lessens overfitting at high capacity
- underfits at low capacity

Why Does Unsupervised Pre-training Help Deep Learning?
Erhan, Bengio, Courville, Manzagol, Vincent and Bengio, 2007

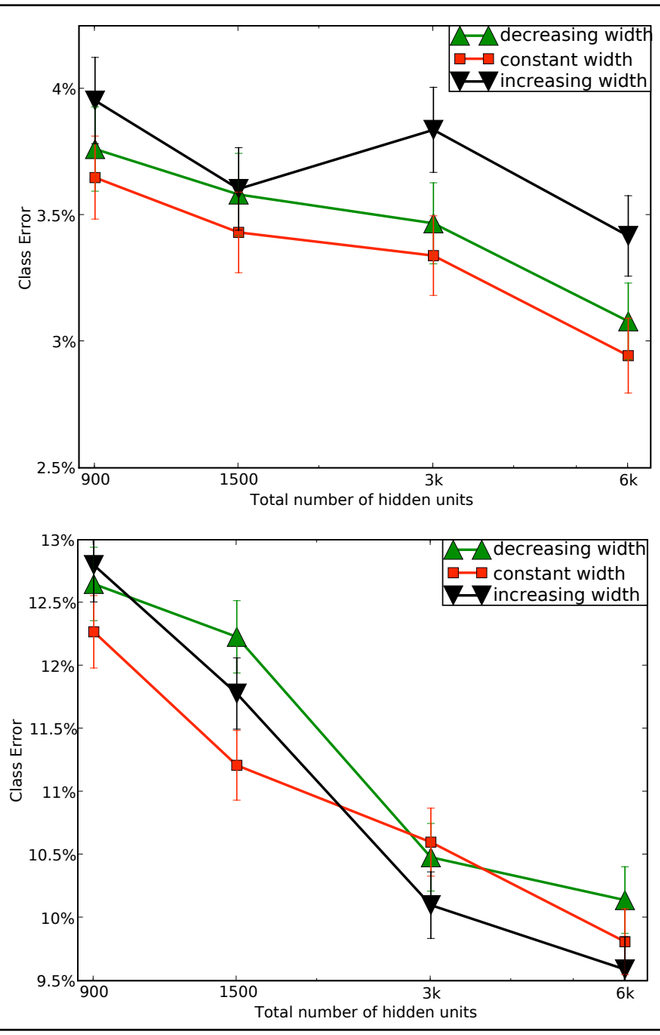
Reproduced from Hugo Larochelle's slides

Comparing Hidden Layer Widths

RBM



Autoencoder



Reproduced from Hugo Larochelle's slides

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%)

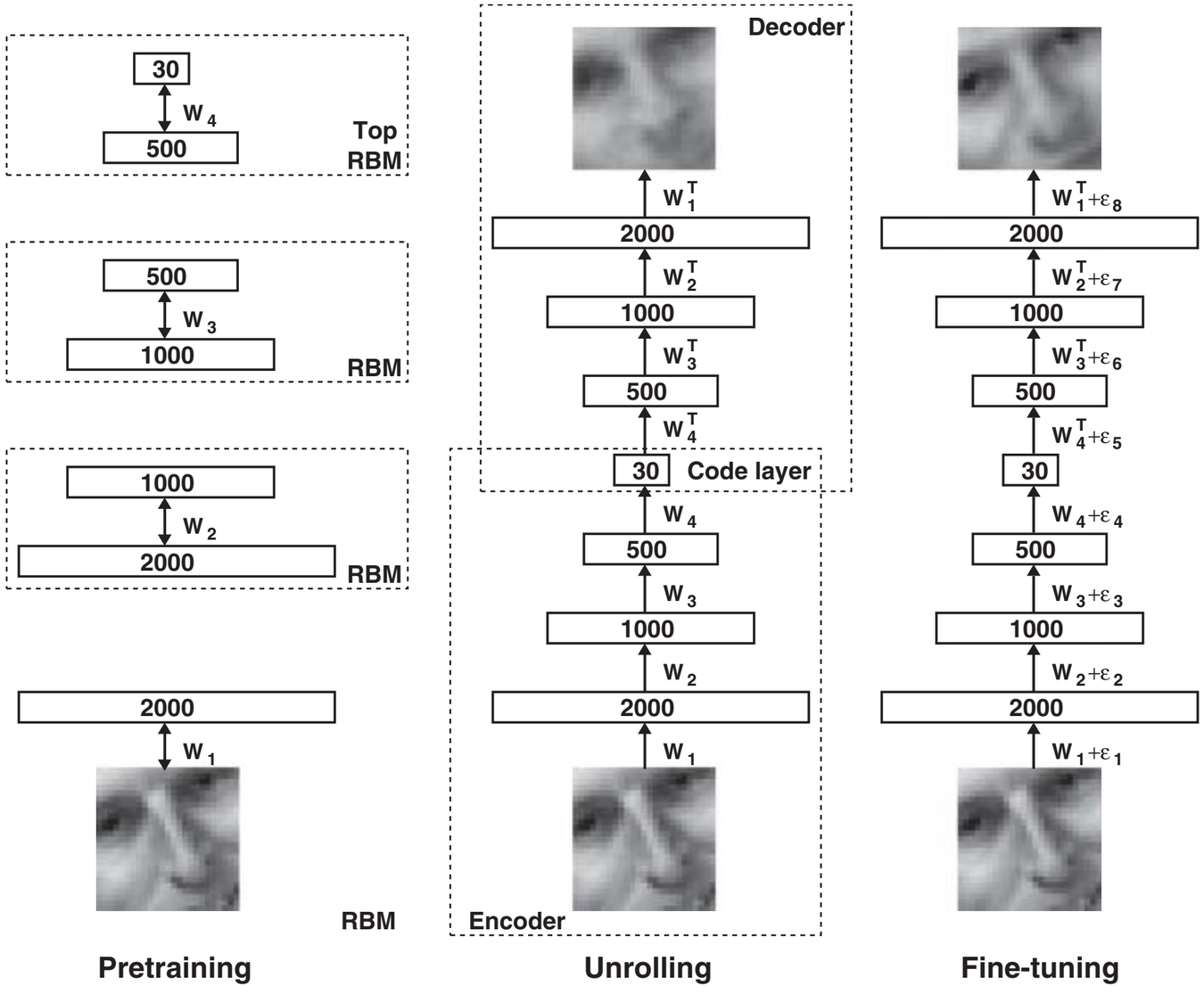
Extracting and Composing Robust Features with Denoising Autoencoders
Vincent, Larochelle, Bengio and Manzagol, 2008

Reproduced from Hugo Larochelle's slides

Deep Autoencoder (Hinton and Salakhutdinov, 2006)

Pre-training can be used to initialize a deep autoencoder

- This is an example of a situation where underfitting is an issue
  - perhaps surprisingly, pre-training initializes the optimization problem in a region with better local optima of training objective
- Each RBM used to initialize parameters both in encoder and decoder ("unrolling")



Reproduced from Hugo Larochelle's slides

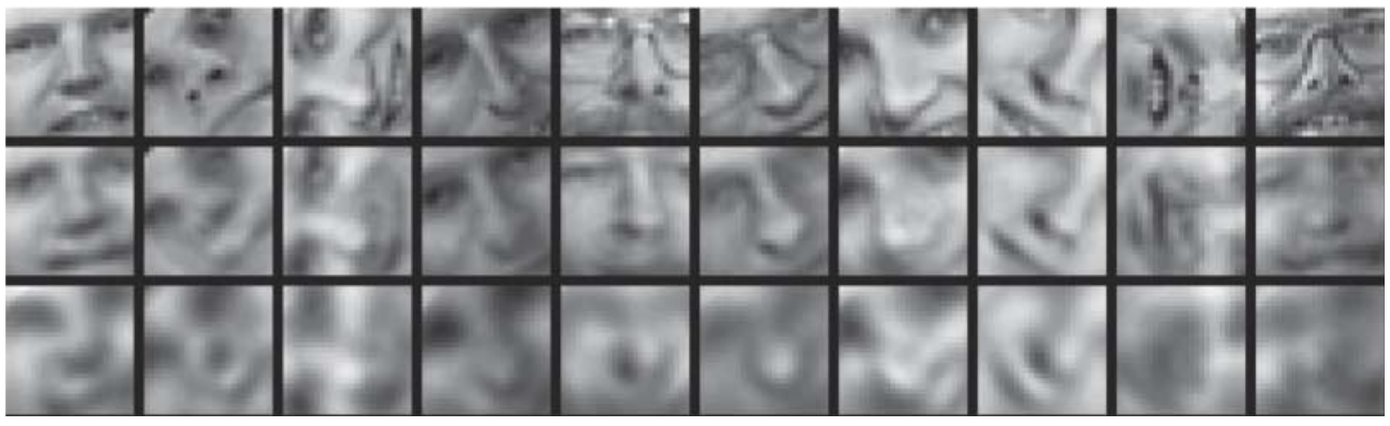
Deep Autoencoder- Application

- Can be used to reduce the dimensionality of the data
- Has better reconstruction than a single layer network (PCA)

Original data

Deep autoencoder reconstruction

PCA reconstruction



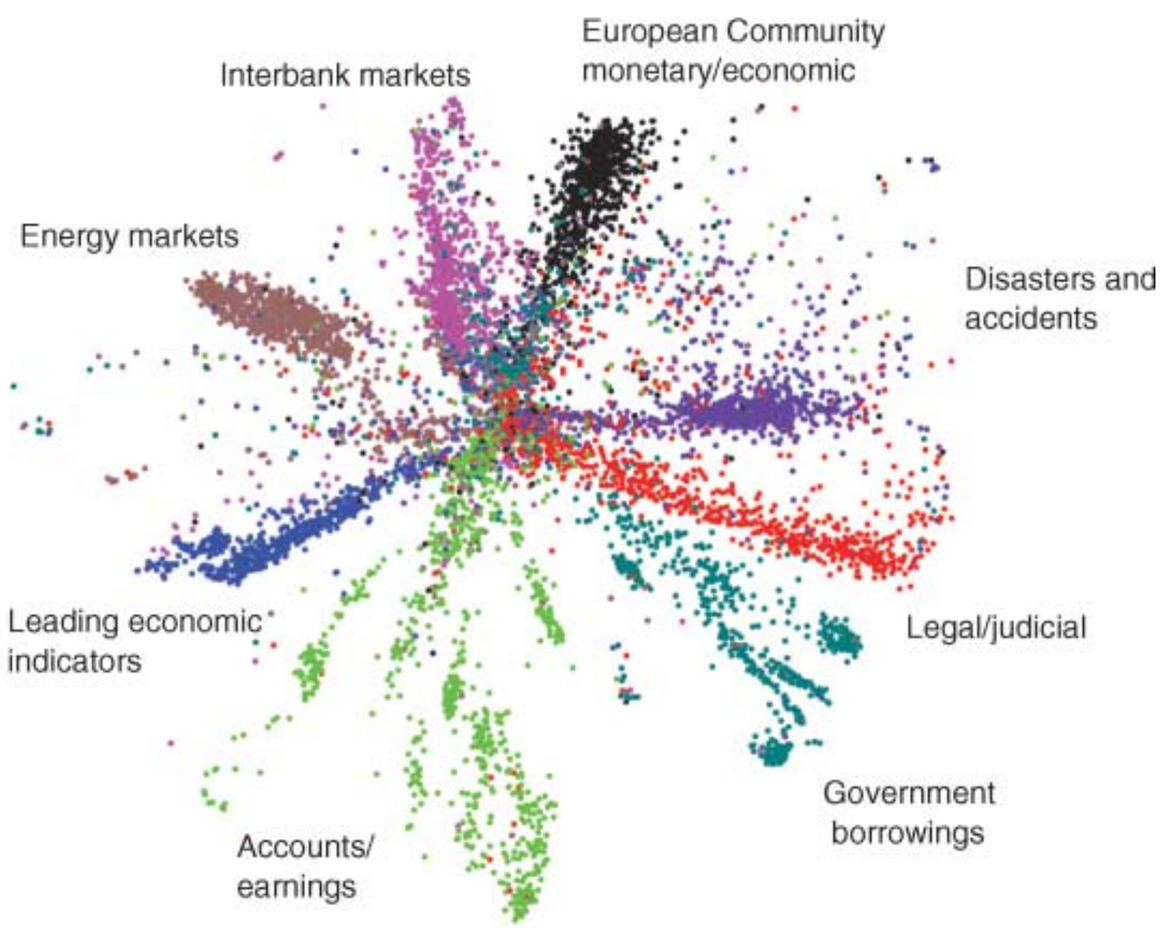
Reproduced from Hugo Larochelle's slides

Deep Autoencoder - Visualization

Reducing to 2D, we can use the deep autoencoder to visualize data (here, a collection of documents)

Latent Semantic Analysis

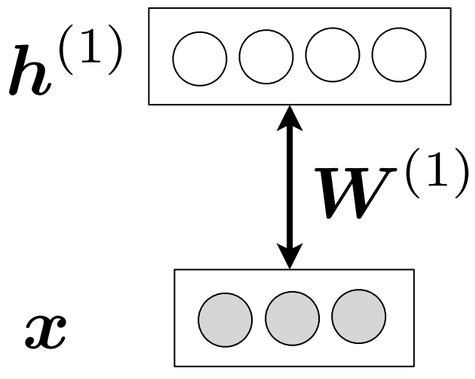
Deep Autoencoder



Reproduced from Hugo Larochelle's slides

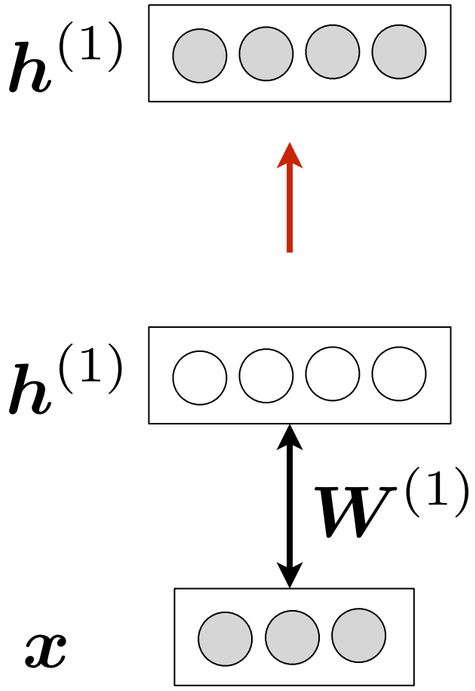
From RBMs to DBNs: Procedure

From RBMs to DBNs: Procedure



① Train an RBM

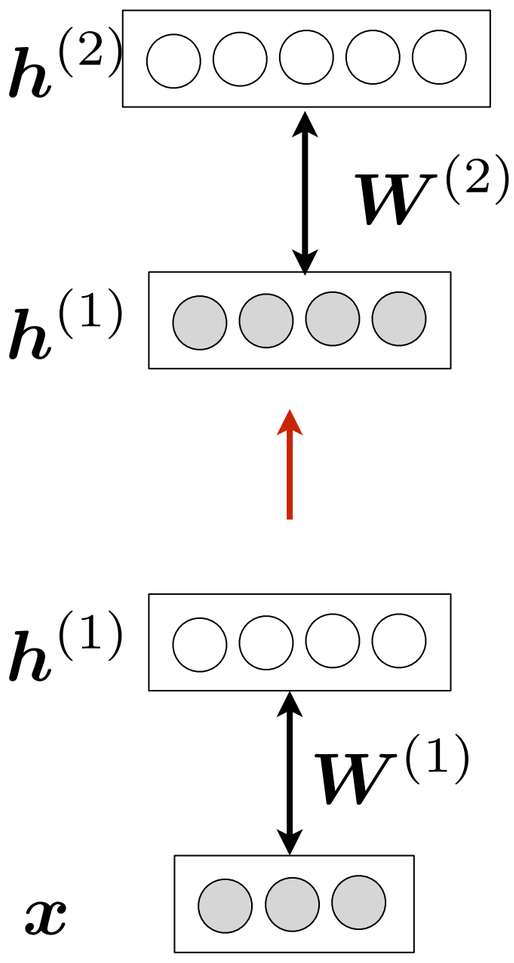
From RBMs to DBNs: Procedure



② Run your data through the model to generate a dataset of hidden activations

① Train an RBM

From RBMs to DBNs: Procedure

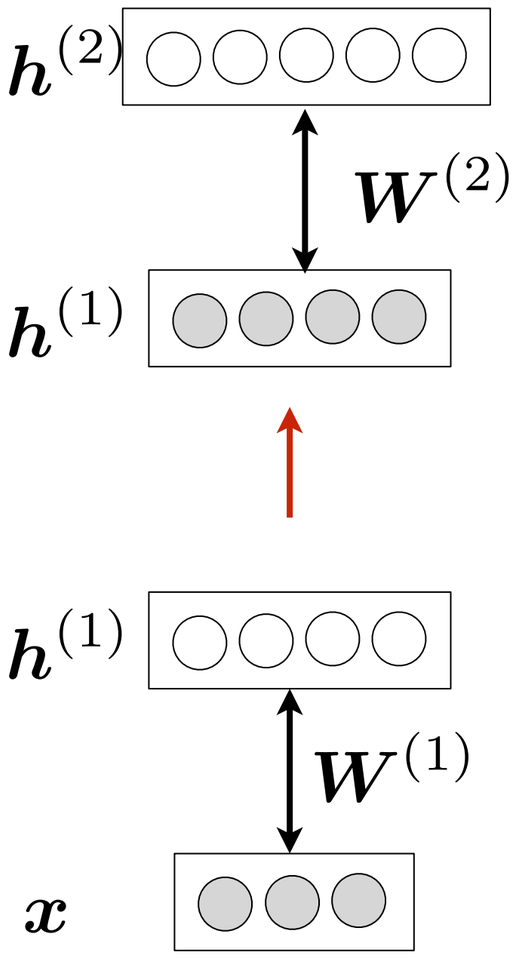


③ Treat the hiddens like data, train another RBM

② Run your data through the model to generate a dataset of hidden activations

① Train an RBM

From RBMs to DBNs: Procedure

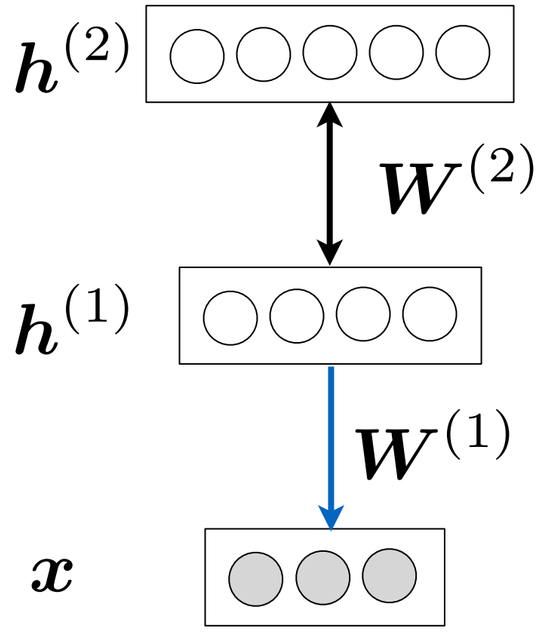


③ Treat the hiddens like data, train another RBM

② Run your data through the model to generate a dataset of hidden activations

① Train an RBM

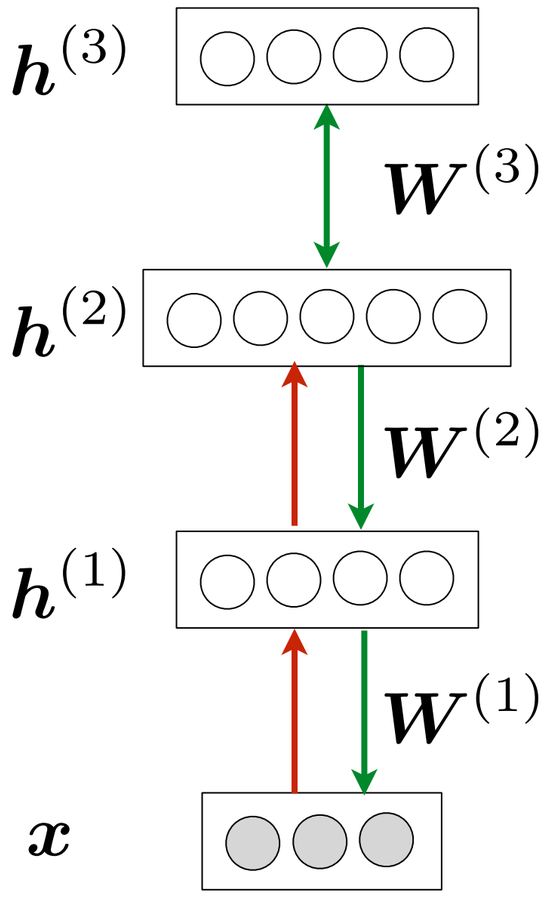
④ Compose the two models



(Hinton et al. 2006)

Deep Belief Networks

- The resulting model is called a Deep Belief Network
- Generate by alternating Gibbs sampling between the top two layers followed by a down-pass
- The lower level bottom-up connections are not part of the generative model, they are used only for inference



Stacking RBMs: Intuition

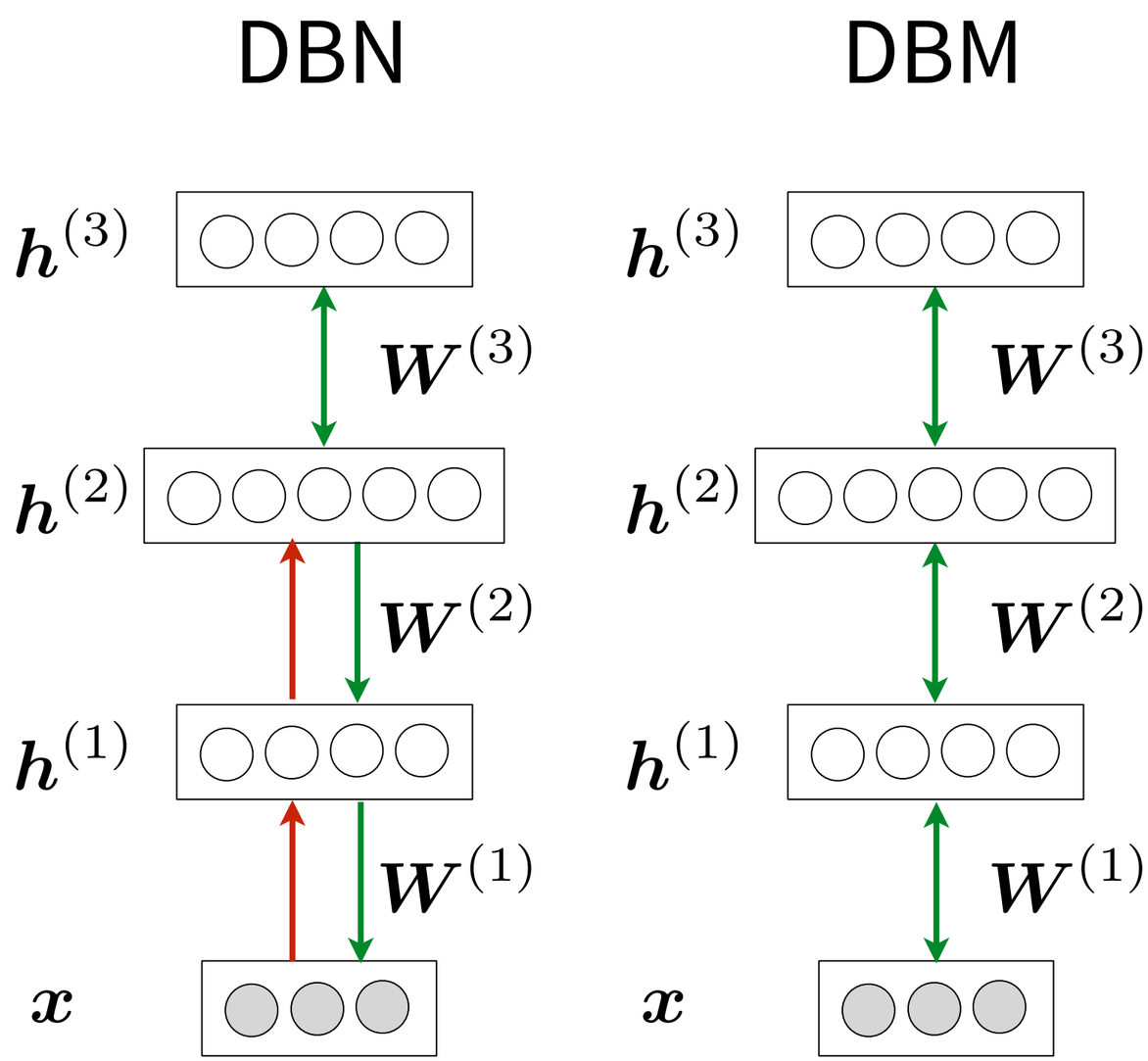
- The weights in the bottom-most RBM define many different distributions: $P(\boldsymbol{x}, \boldsymbol{h}), P(\boldsymbol{x}|\boldsymbol{h}), P(\boldsymbol{h}|\boldsymbol{x}), P(\boldsymbol{x}), P(\boldsymbol{h})$
- We can express the RBM as: $P(\boldsymbol{x}) = \sum_{\boldsymbol{h}} P(\boldsymbol{h}) P(\boldsymbol{x}|\boldsymbol{h})$
- If we leave $p(\mathbf{x}|\mathbf{h})$ as-is and improve $P(\boldsymbol{h})$, we improve $P(\boldsymbol{x})$
- To improve $P(\boldsymbol{h})$ we need it to be better than $P(\boldsymbol{h}; \boldsymbol{W}^{(1)})$ at modeling the aggregated posterior over hidden vectors produced by applying the RBM to the data

Credit: Geoff Hinton

(Salakhutdinov and Hinton 2009)

Deep Boltzmann Machines

- DBN is a hybrid directed graphical model
  - maintains a set of “feed-forward” connections for inference
- DBM is an undirected graphical model
  - feedback is important
- Both take different approaches to dealing with intractable $P(\boldsymbol{h}|\boldsymbol{x})$



Training DBMs

Training DBMs

- Standard DBM training procedure:

Training DBMs

- Standard DBM training procedure:
  - Greedy-wise pre-training of RBMs

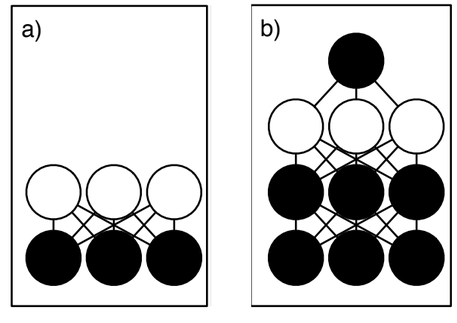


Image: (Goodfellow et al. 2013)

Training DBMs

- Standard DBM training procedure:
  - Greedy-wise pre-training of RBMs
  - Stitch the RBMs into a DBM and train with variational approximation to log-likelihood

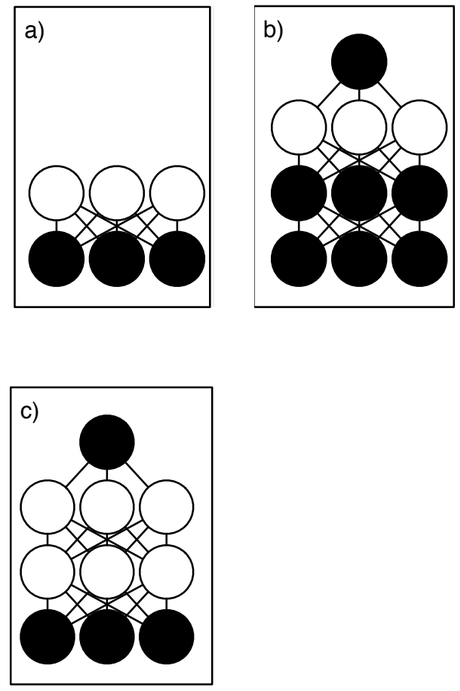


Image: (Goodfellow et al. 2013)

Training DBMs

- Standard DBM training procedure:
  - Greedy-wise pre-training of RBMs
  - Stitch the RBMs into a DBM and train with variational approximation to log-likelihood
  - Discriminative fine-tuning (DBM used as feature learner)

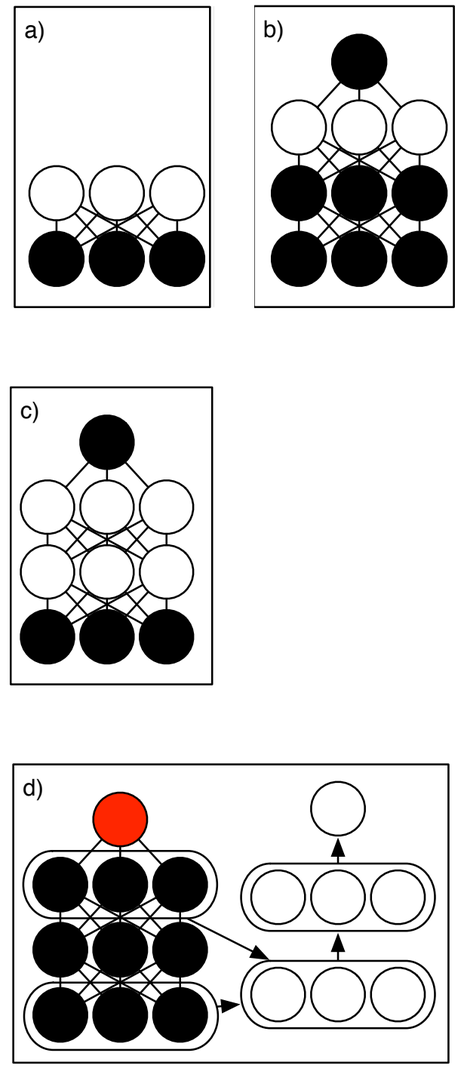


Image: (Goodfellow et al. 2013)

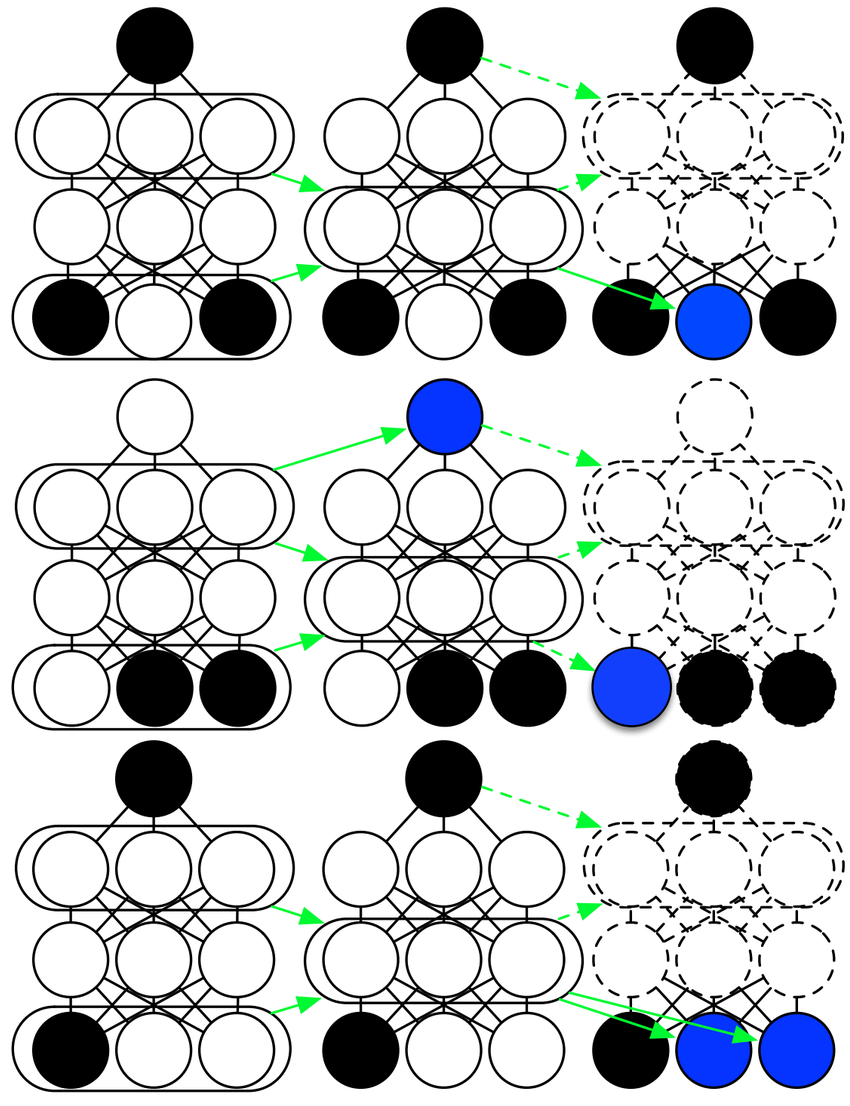
Multi-prediction DBMs

(Goodfellow et al. 2013)

Multi-prediction DBMs

(Goodfellow et al. 2013)

Multi-prediction training for classification



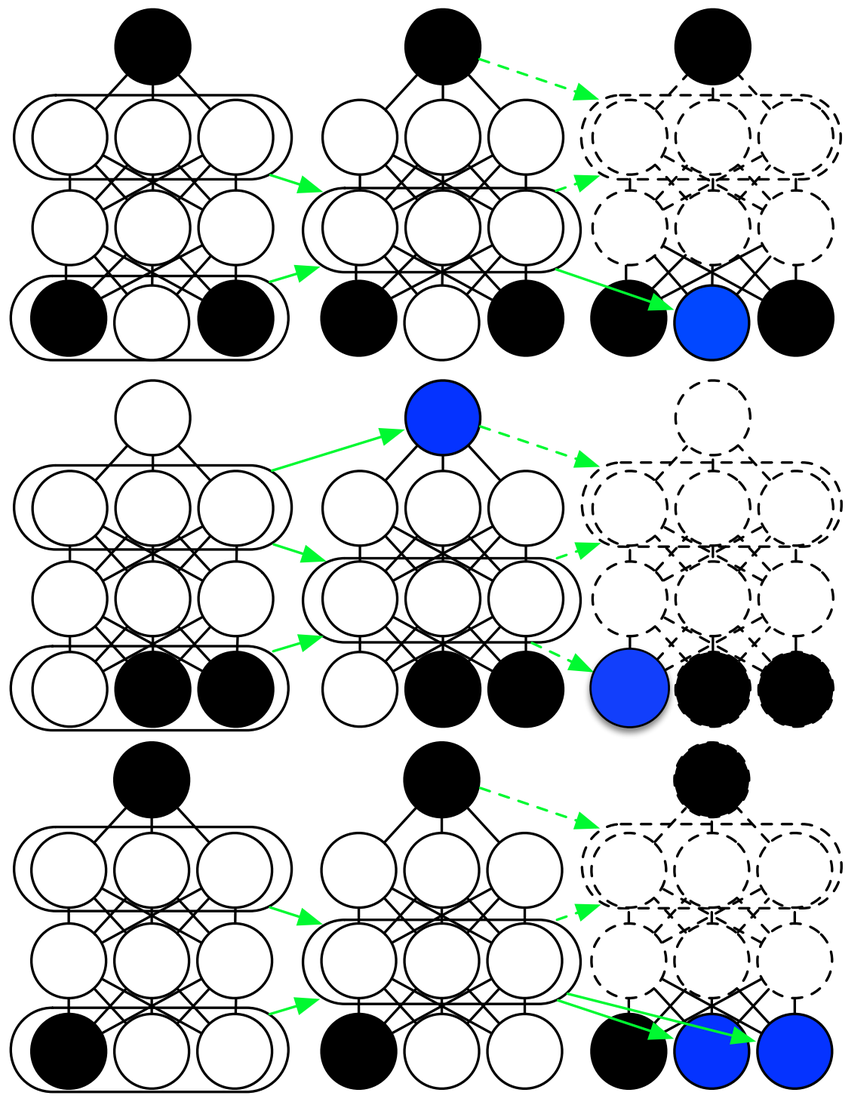
Black - variables net is allowed to observe
Blue - prediction targets

Multi-prediction DBMs

(Goodfellow et al. 2013)

- Greedy pre-training is suboptimal
  - training procedure for each layer should account for the influence of deeper layers
  - one model for all tasks can use inference for arbitrary queries
  - needing to implement multiple models and stages makes DBMs cumbersome

Multi-prediction training for classification



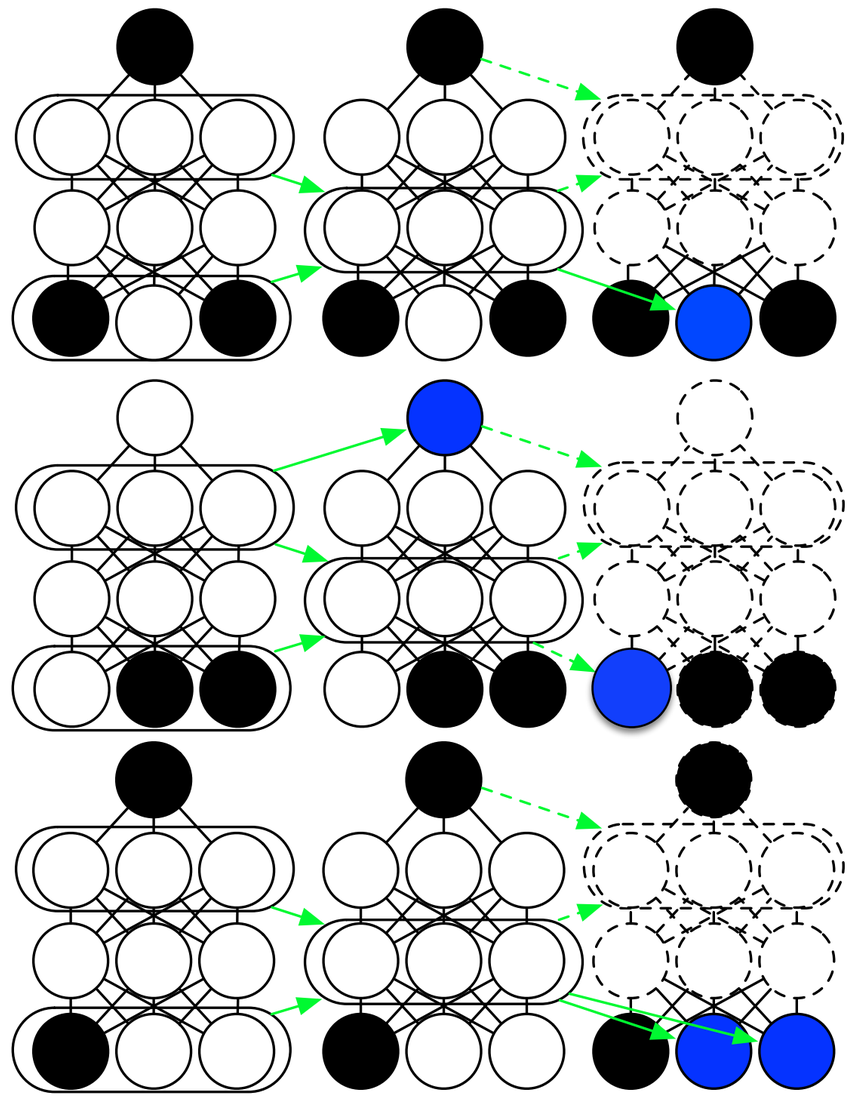
Black - variables net is allowed to observe
Blue - prediction targets

Multi-prediction DBMs

(Goodfellow et al. 2013)

- Greedy pre-training is suboptimal
  - training procedure for each layer should account for the influence of deeper layers
  - one model for all tasks can use inference for arbitrary queries
  - needing to implement multiple models and stages makes DBMs cumbersome
- Joint “multi-prediction” training (Goodfellow et al. 2013)
  - Train DBM to predict any subset of vars given the complement of that subset

Multi-prediction training for classification



Black - variables net is allowed to observe
Blue - prediction targets