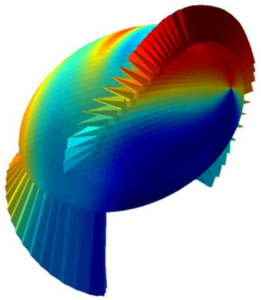
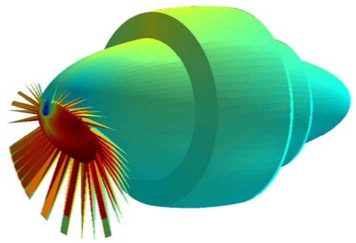
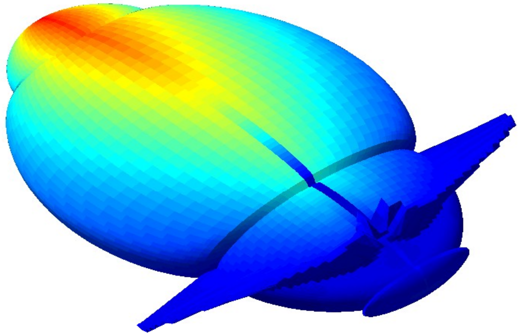
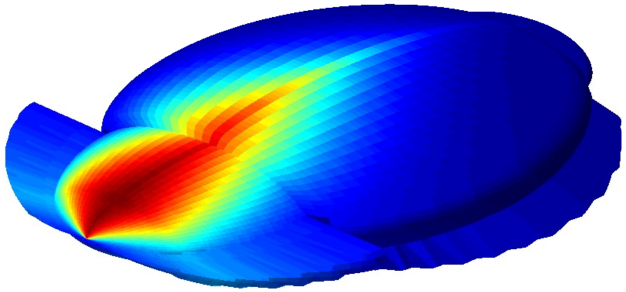
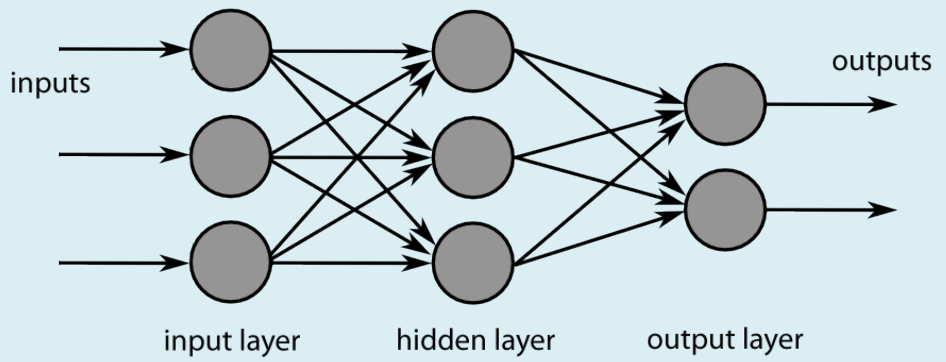



Lecture 13
Deep Learning I

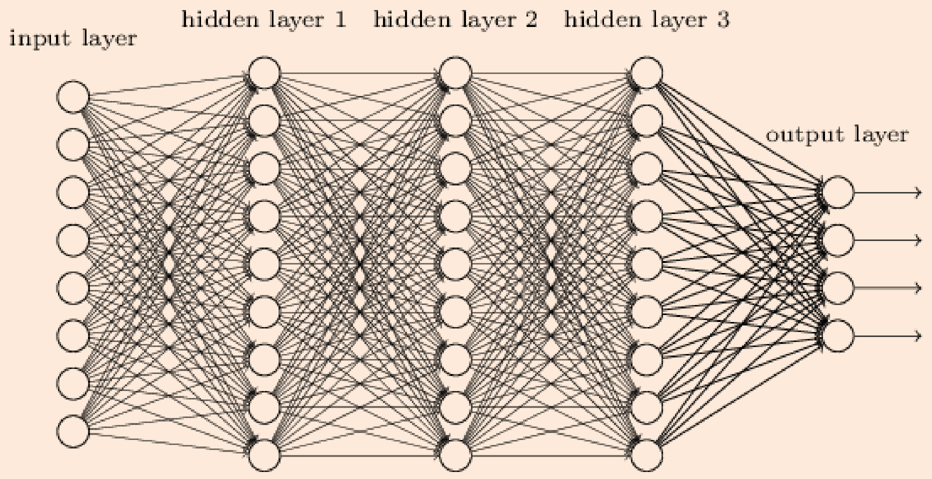
What is deep about "deep learning"?



https://blog.paralleldots.com/data-science/challenges-in-deep-learning/

Shallow Neural Network

- Can be feed-forward or recurrent.
- 1 or 2 layers between input and output.
- Typically, all layers except Input have similar functionality.

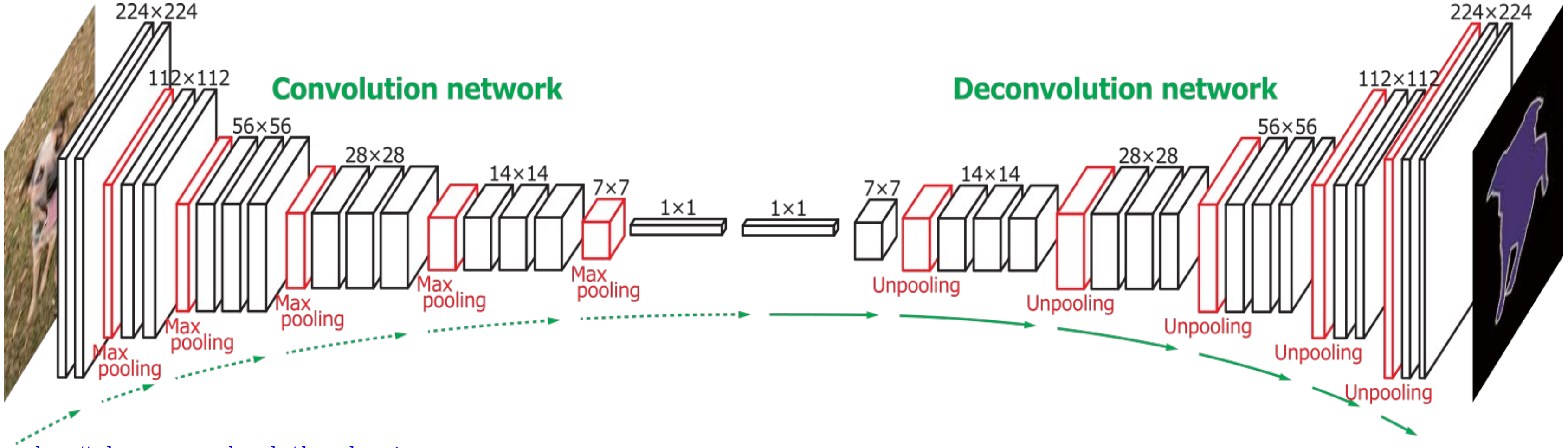


http://neuralnetworksanddeeplearning.com/chap6.html

Deep Neural Network

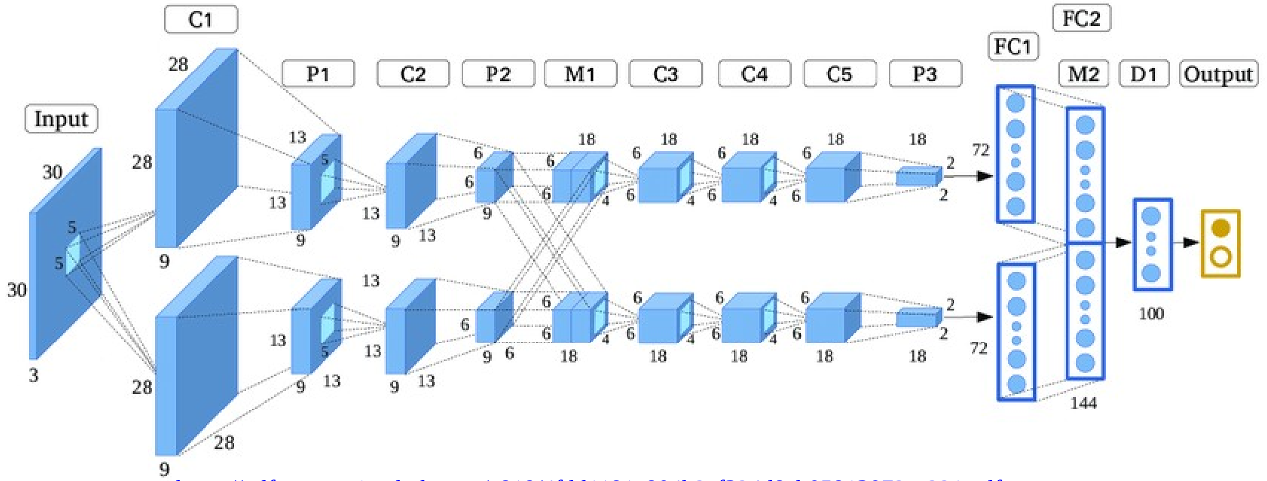
- Can be feed-forward or recurrent.
- 3 or more layers between input and output.
- Different layers may have very different functionality.

Sometimes, the networks can get very deep …..



http://mlcenter.postech.ac.kr/deep_learning

….. or have multiple streams.



https://pdfs.semanticscholar.org/e819/4fdd1191c804b2af324d8ab05913072ce221.pdf

Why are deep networks useful?

A layer of neurons can provide various types of transformations:

- Re-mapping to a new feature space.
- Compression to lower dimensional feature space.
- Expansion to higher dimensional feature space.
- Filtering of the input data.
- Discretization of the input data.
- Clustering of input data.
- Classification of the input data.
- Integration of current and previous inputs/outputs (memory).
- Noise removal.
- Pattern completion.

Deep networks allow us to combine layers with different functions to accomplish complicated tasks.

Why has deep learning become popular only recently?

- We now have much faster computers.
- We now have a lot of data (images, video, texts, etc.) that requires more complicated processing.
- Better methods have recently been developed to design and train deep networks.
- Several powerful new deep network models have been developed recently.

Classic Deep Learning Models

Feed-Forward

Deep feed-forward networks
(Classification, prediction, pattern recognition, etc.)

Auto-encoders
(Dimensionality reduction, feature extraction, generative models)

Convolutional networks (CNN)
(Image and video analysis, language analysis, etc.)

Recurrent

Deep recurrent networks
(Time-series prediction, sequence analysis, natural language processing, etc.)

Long short-term memory (LSTM)
(Sequence analysis, natural language processing, etc.)

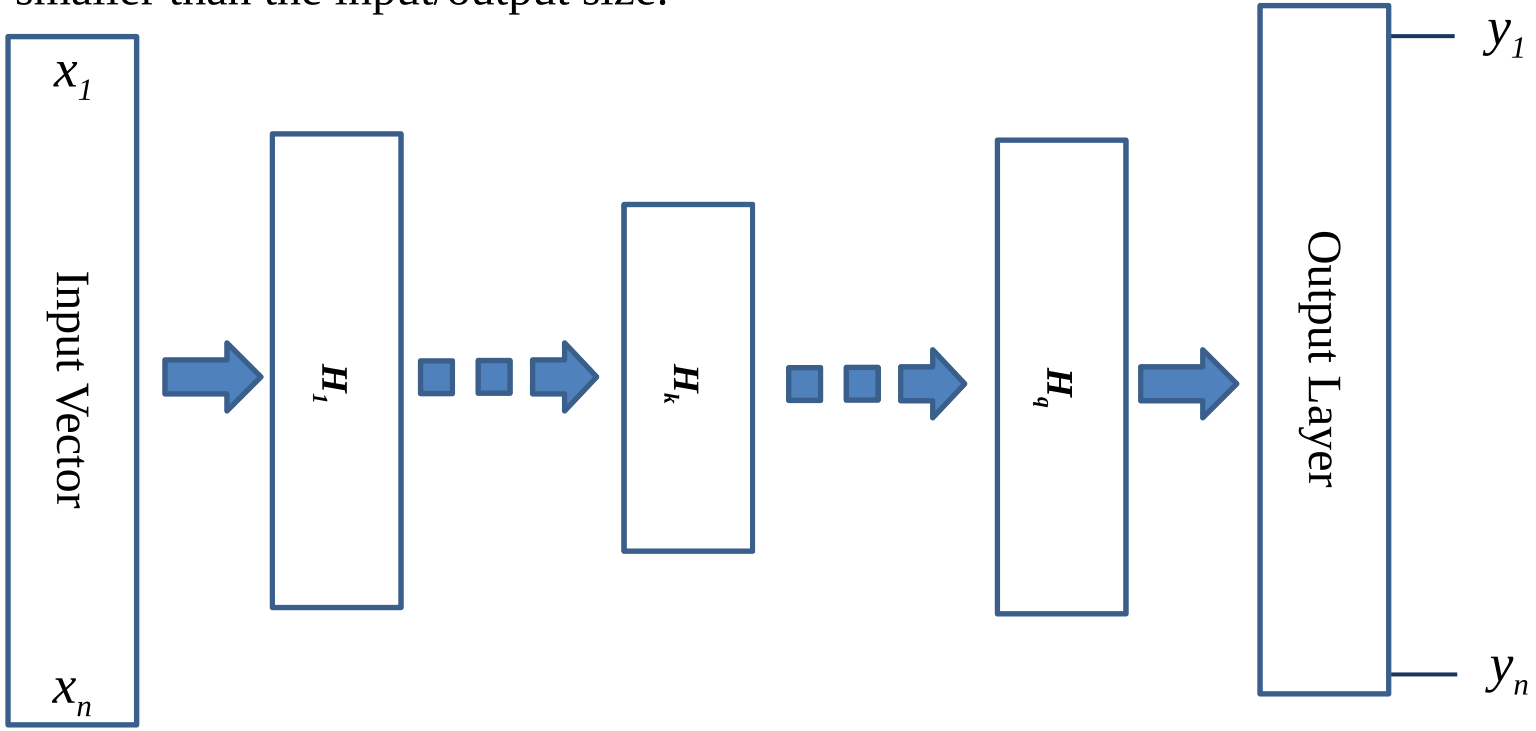
Deep belief networks (DBN)
(Classification, prediction, pattern recognition, feature extraction, language analysis, etc.)

A useful tutorial: https://lilianweng.github.io/lil-log/2017/06/21/an-overview-of-deep-learning.html#recurrent-neural-network

Autoencoder

A multi-layer feed-forward neural network that is trained to reproduce its input vector as output.

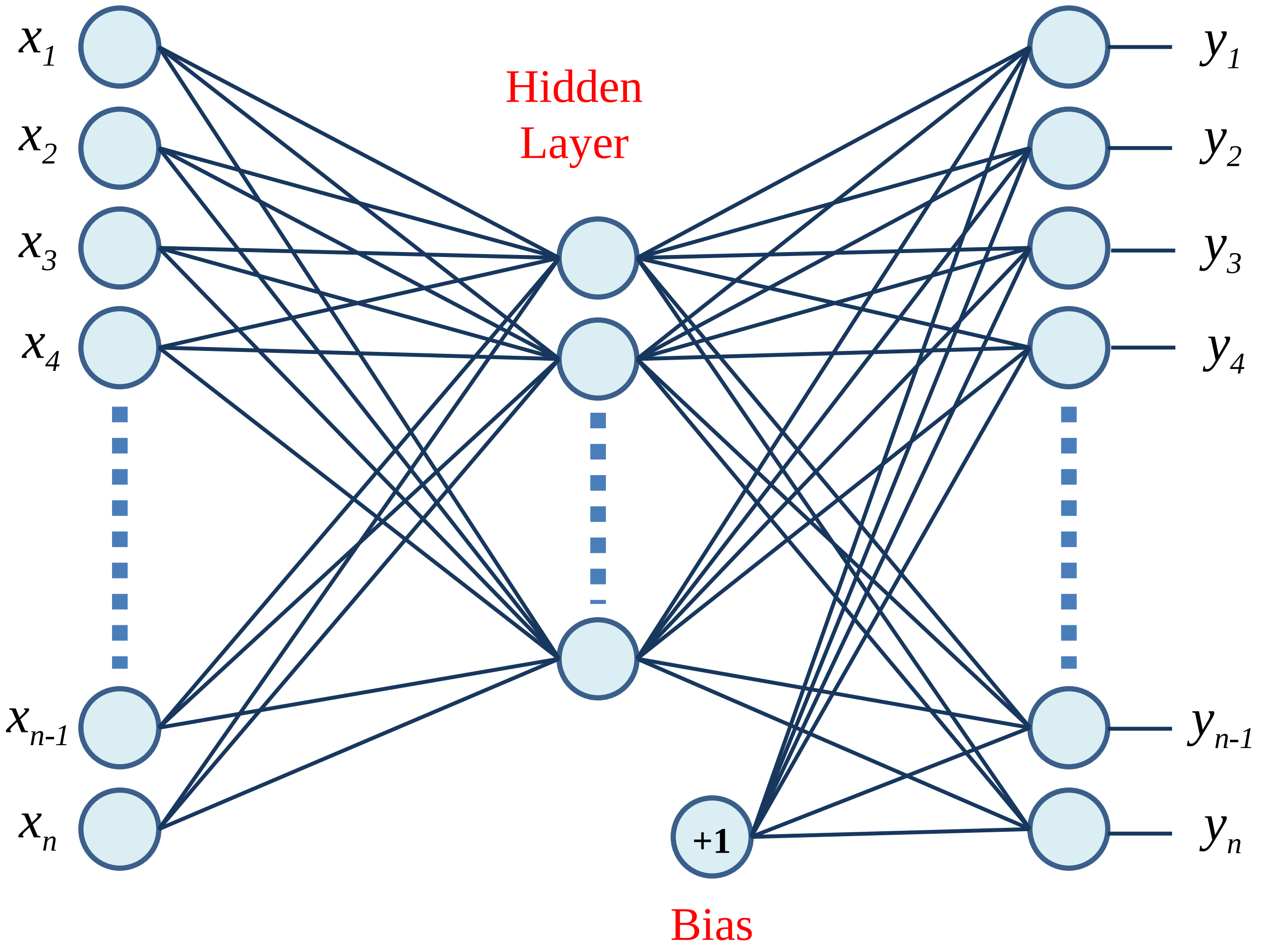
Autoencoders typically – but not always – use hidden layers that are smaller than the input/output size.



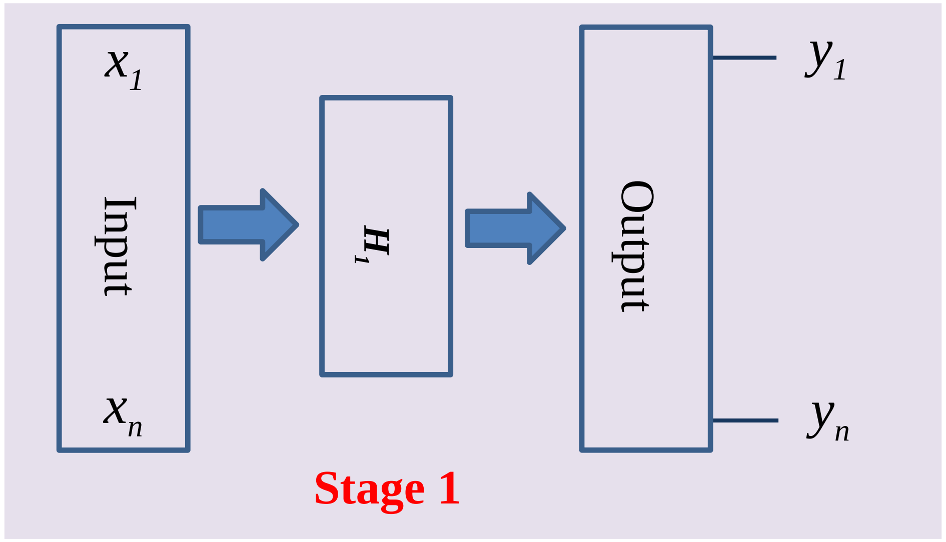
For a nice introduction, see:

http://www.subsubroutine.com/sub-subroutine/2015/7/15/how-neural-net-autoencoders-can-automatically-abstract-visual-features-from-handwriting

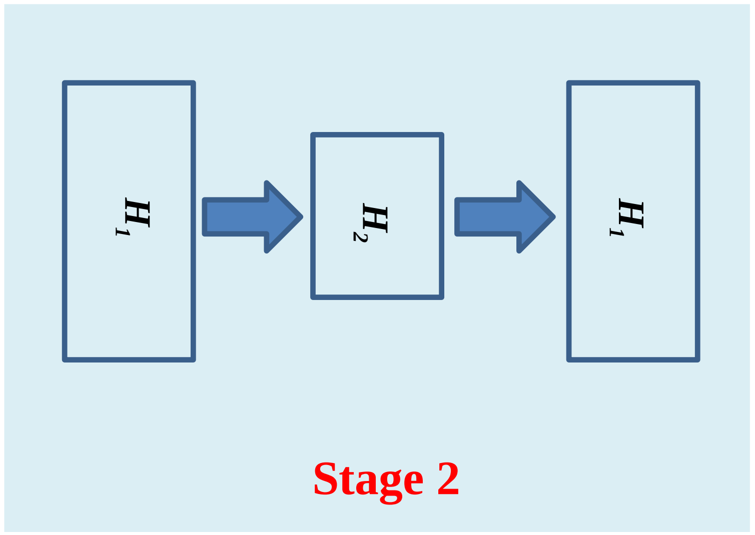
Shallow Autoencoder



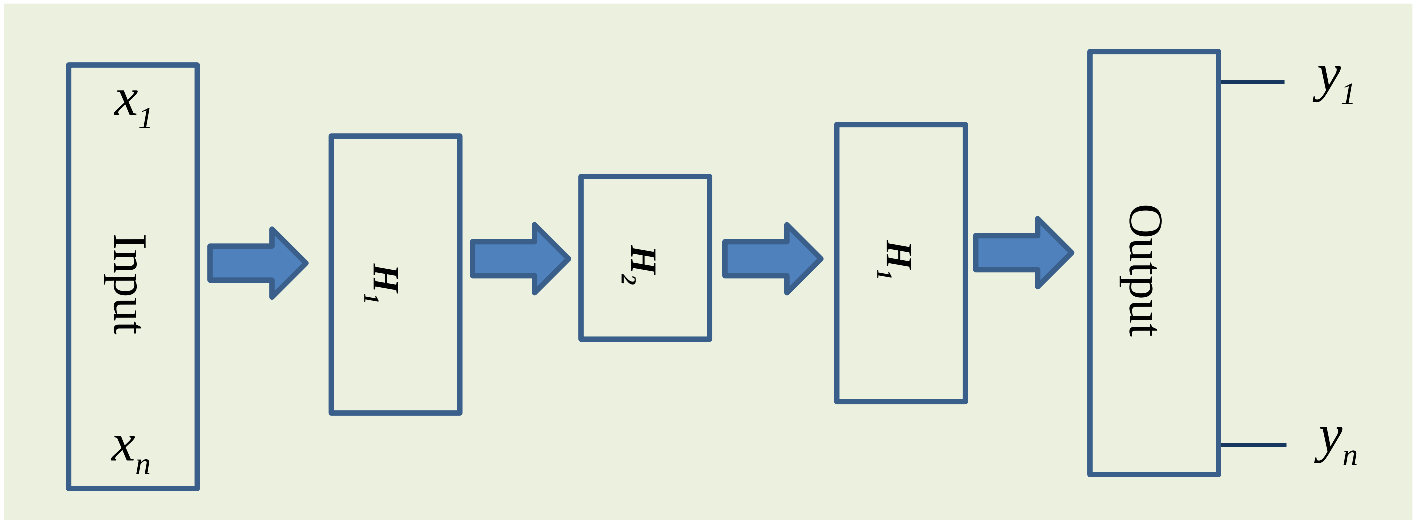
Building Multi-Stage Autoencoders



Stage 1

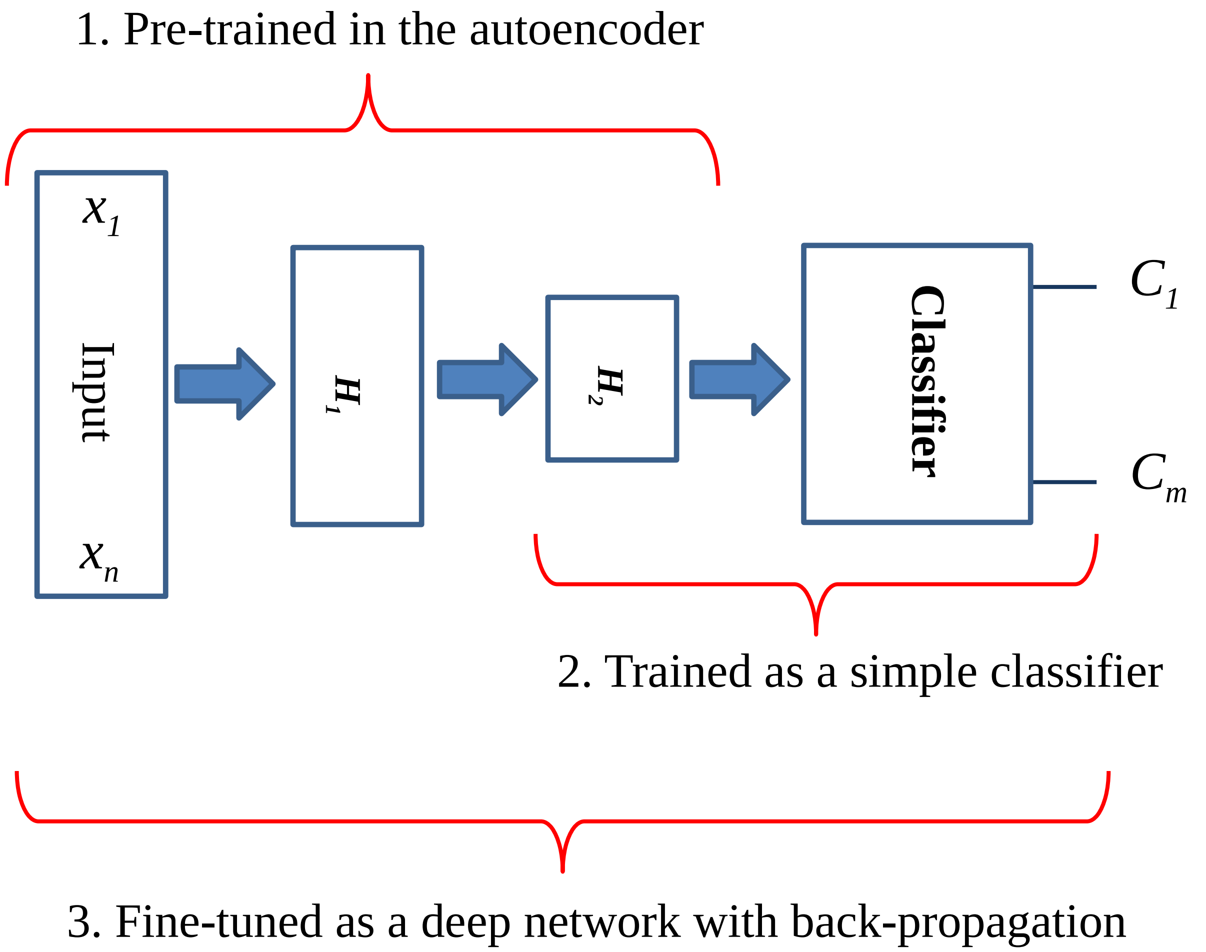


Stage 2



Final Network – possibly with some further end-to-end BP training

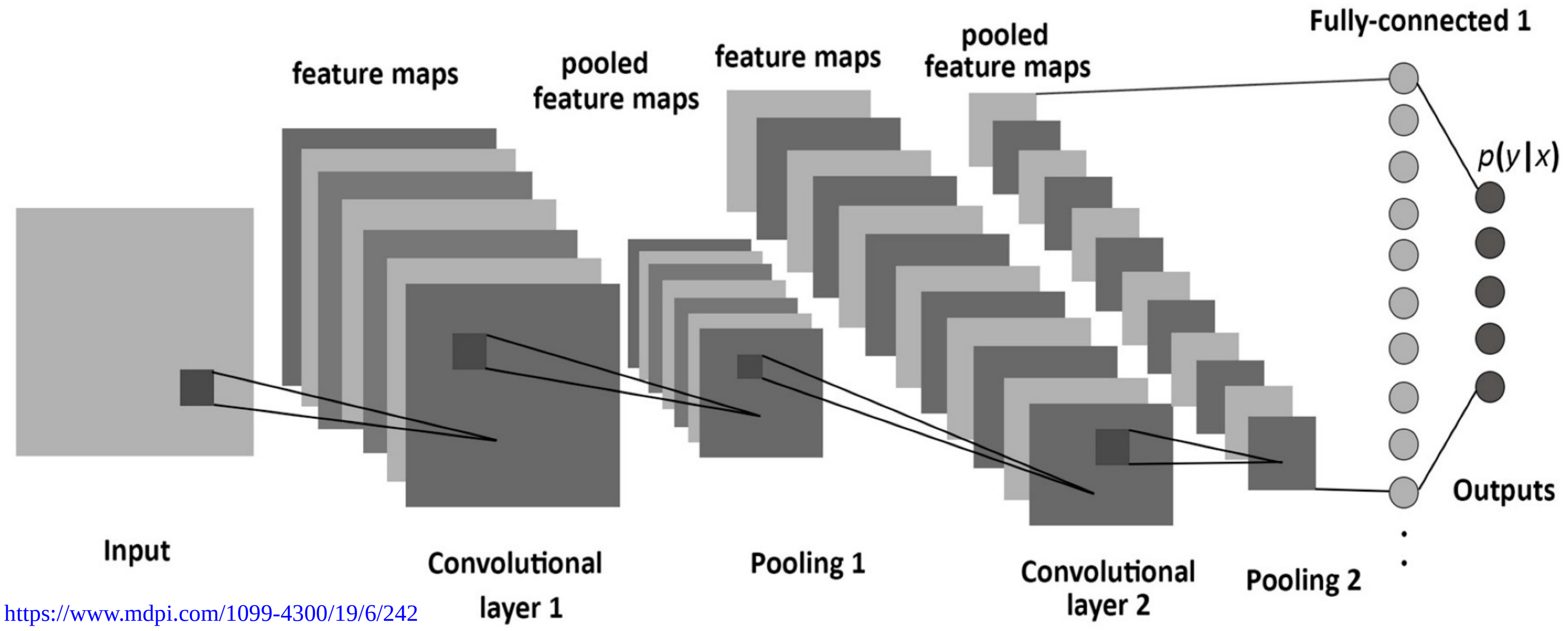
Autoencoder-Based Deep Classifier


1. Pre-trained in the autoencoder
x_1
Input
x_n
H_1
H_2
Classifier
C_1
C_m
2. Trained as a simple classifier
3. Fine-tuned as a deep network with back-propagation

The classifier could be a layer of neurons (LMS), a multi-layer network, a softmax classifier, etc.

Convolutional Neural Networks (CNN)

CNNs use filter convolution to extract features from images and other data, compress the dimension, and use the results for classification, object detection, etc.

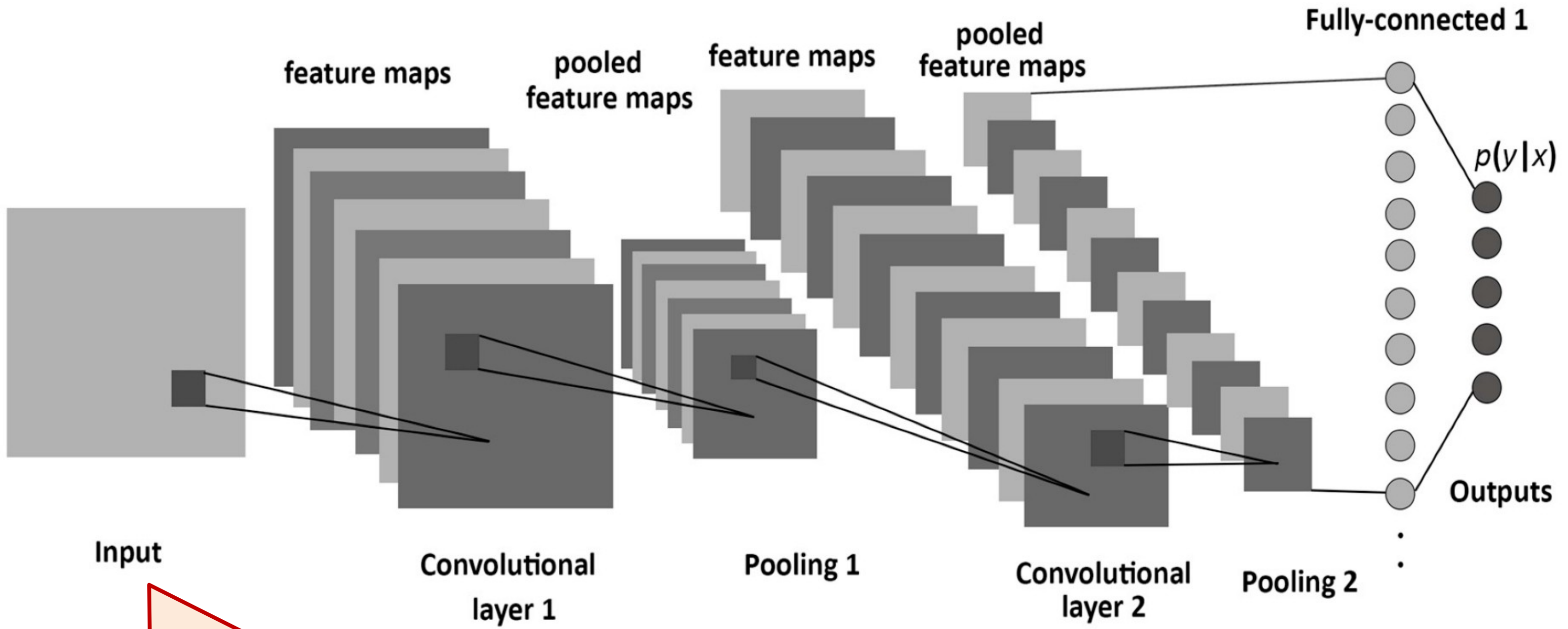


From: https://www.mdpi.com/1099-4300/19/6/242

Tutorials:

https://www.youtube.com/watch?v=oI2rvjbzVmI
http://cs231n.github.io/convolutional-networks/

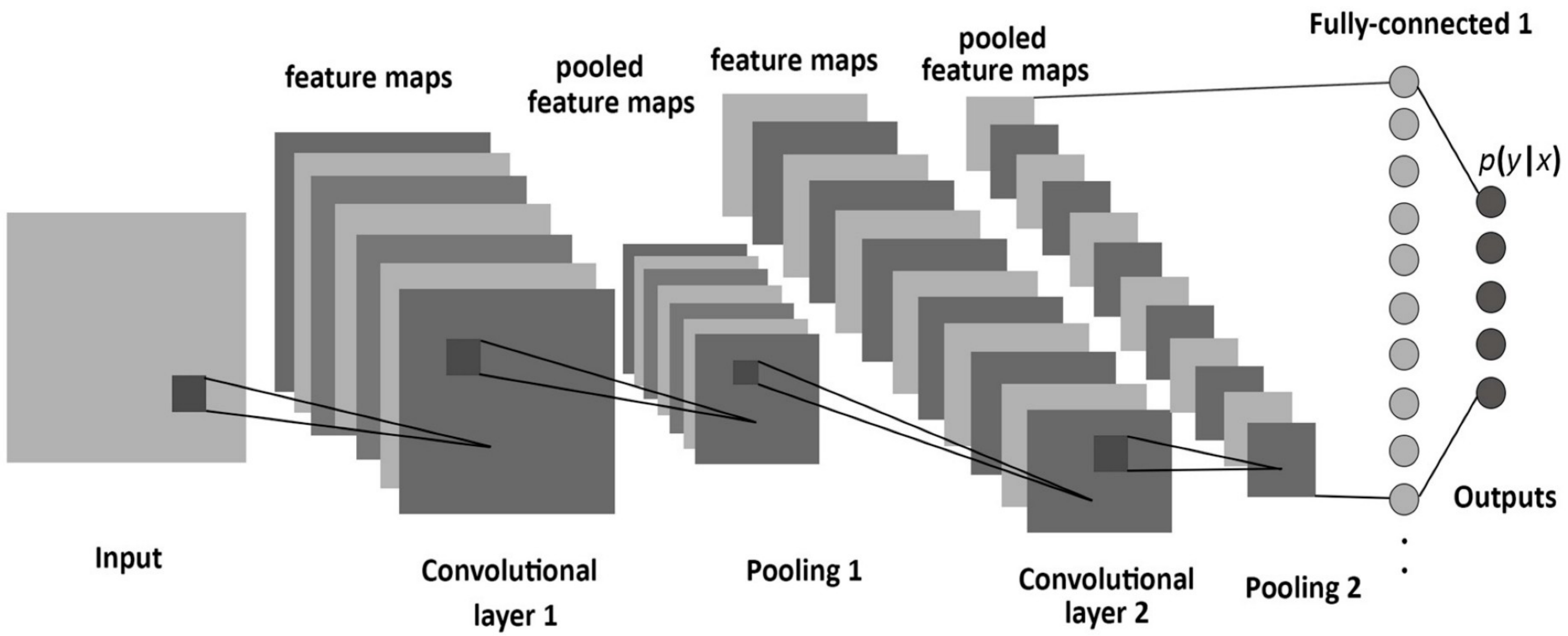
CNN Components



The **Input Layer** provides the data to be processed (usually image, video frame, or text)

From: https://www.mdpi.com/1099-4300/19/6/242

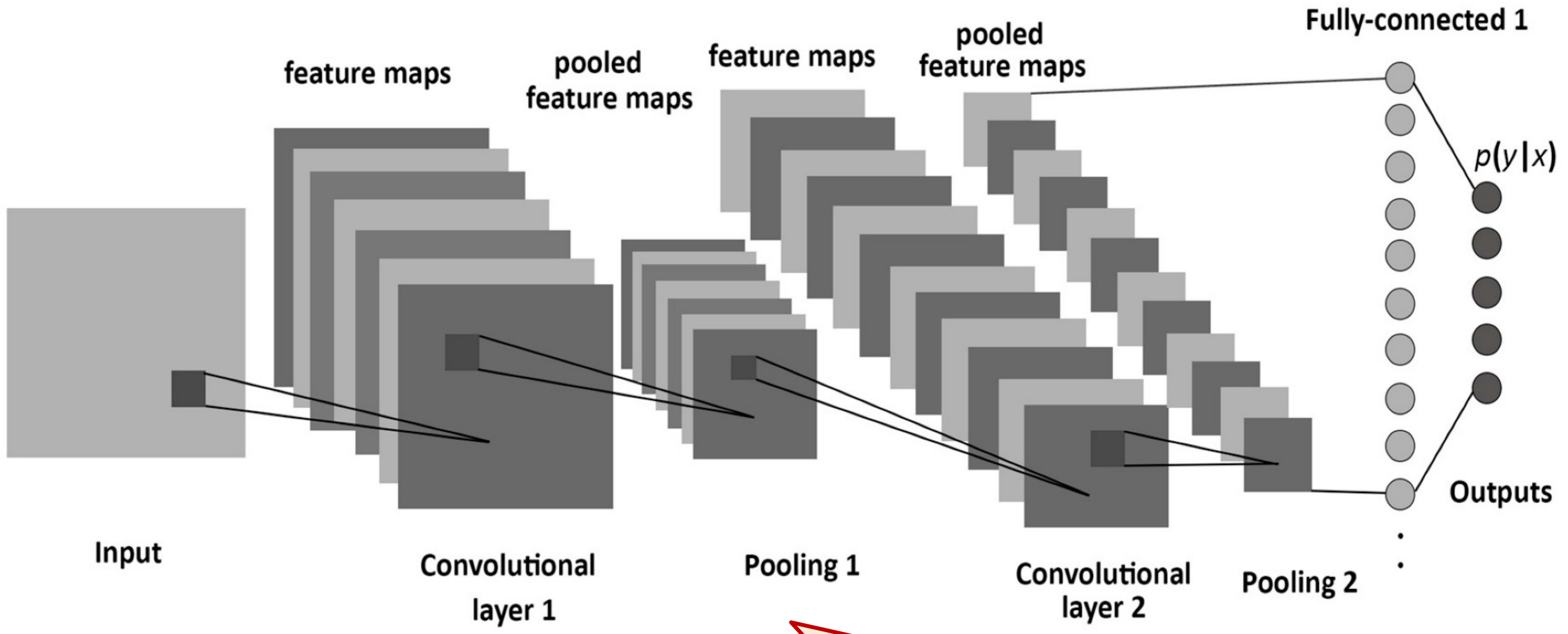
CNN Components



- A **Convolutional Layer** has several sub-layers, each representing a ***filter mask*** convolved with the entire input image in parallel.
- Each filter mask is typically much smaller than the input image.
- Each sub-layer applies the mask at regularly spaced patches on the image, and returns a resulting image.

From: https://www.mdpi.com/1099-4300/19/6/242

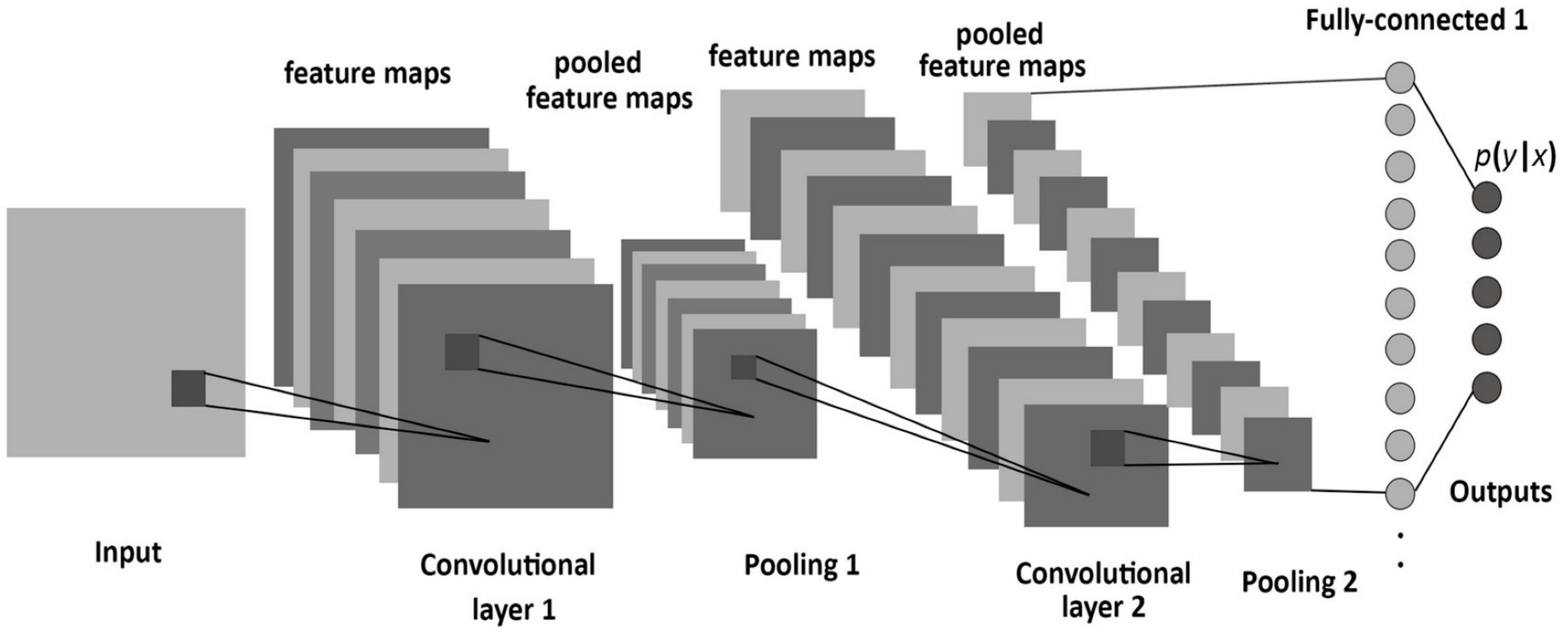
CNN Components



- A **Pooling Layer** samples $k \times k$ patches of each preceding convolutional sub-layer, and produces a single number for each patch (e.g., maximum value in the patch = ***max-pooling***).
- The pooling process is a type of ***sub-sampling***.

From: https://www.mdpi.com/1099-4300/19/6/242

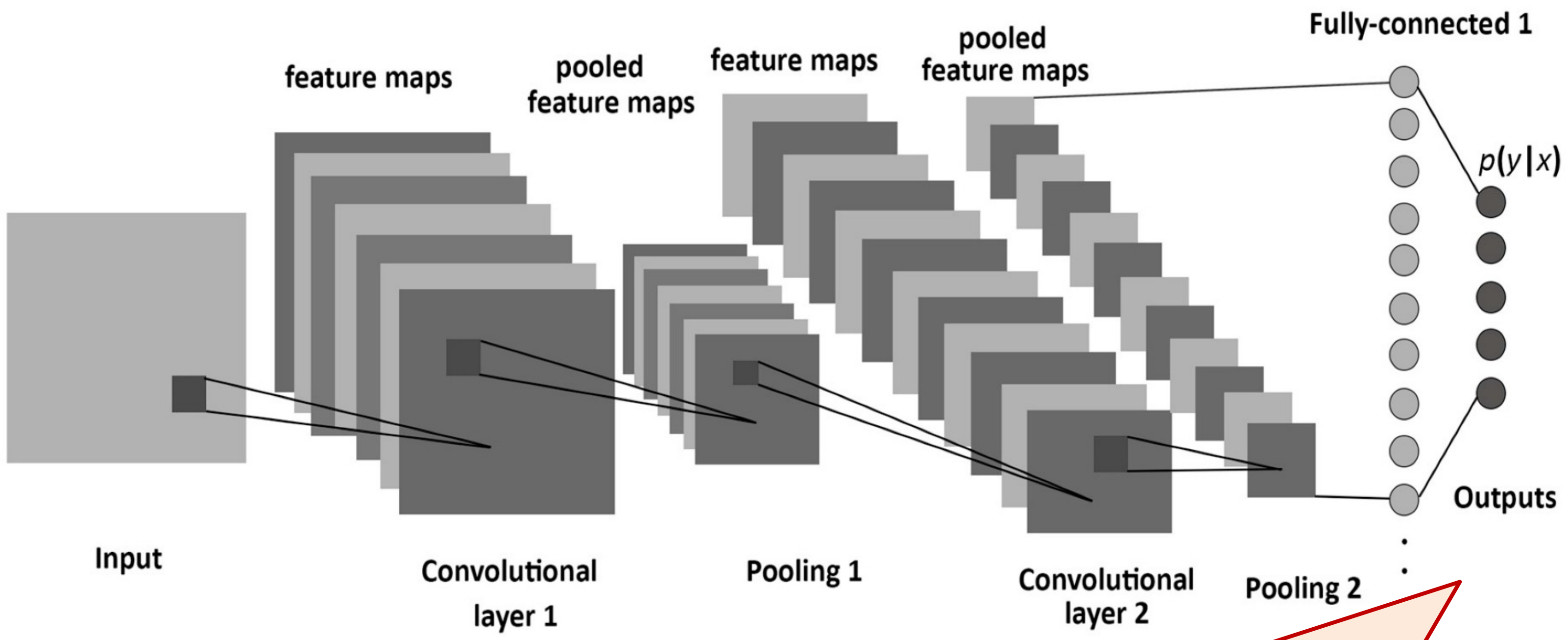
CNN Components



- Convolution layers and pooling layers typically alternate in the network.
- Each convolutional layer extracts higher order features from the previous layer (as in the visual system).
- This process can be seen as “squeezing out” significant information from the original image.

From: https://www.mdpi.com/1099-4300/19/6/242

CNN Components

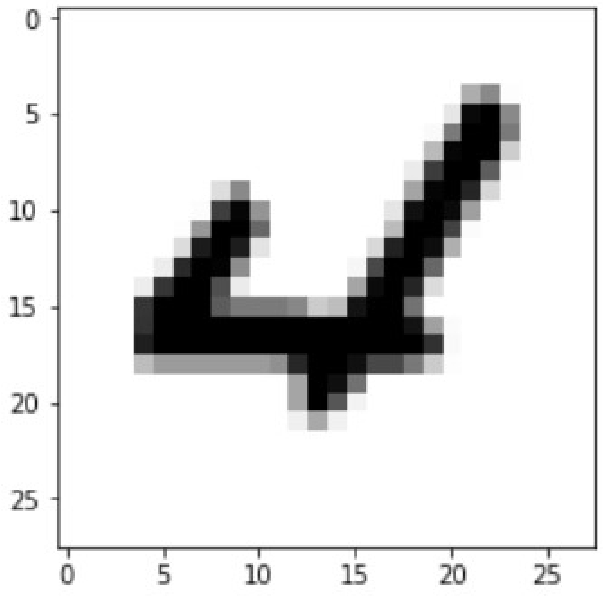


- Typically, the final pooling layer output is input to a classifier or some other functional module.
- The CNN allows this function to operate on a smaller set of relevant features rather than the raw image/data.

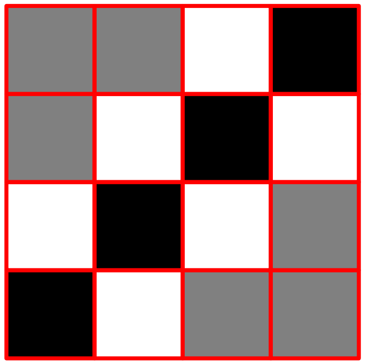
From: https://www.mdpi.com/1099-4300/19/6/242

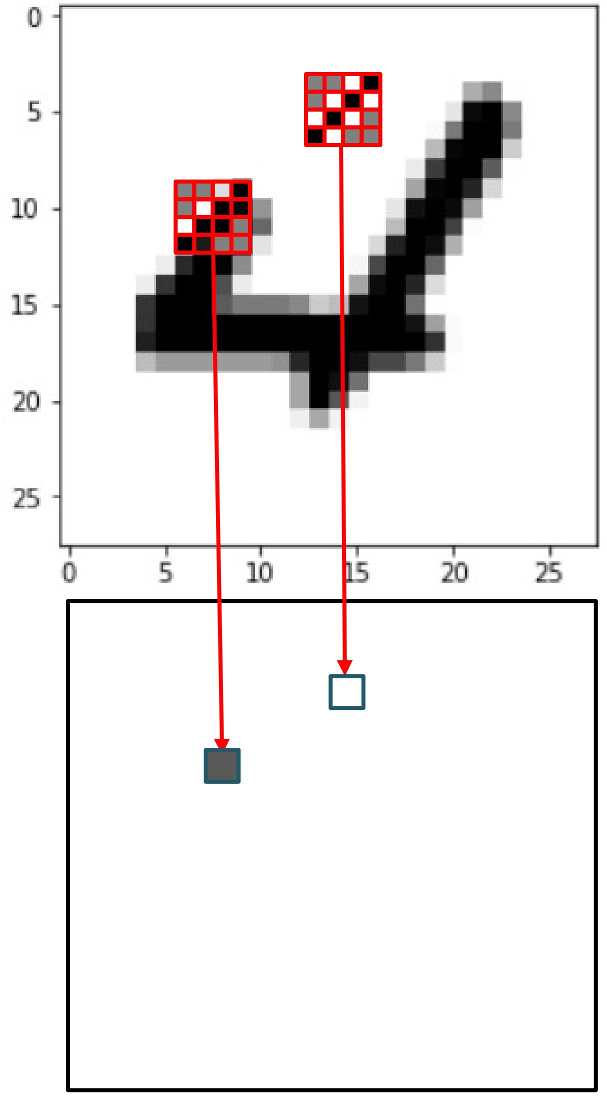
Convolution

28×28 image



4×4 filter





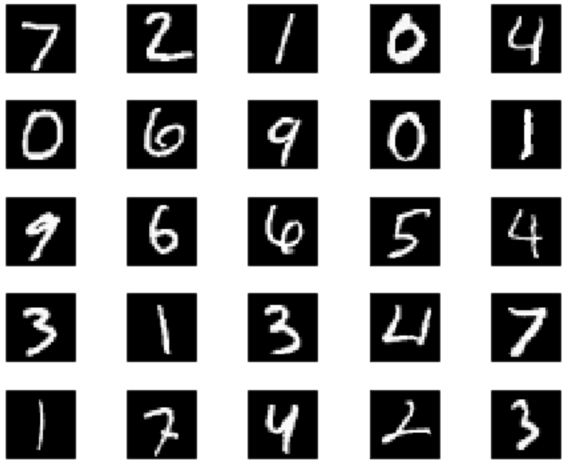
The filter is applied to patches of the image in a grid.

Each patch feeds a neuron in the convolution sub-layer for that filter.

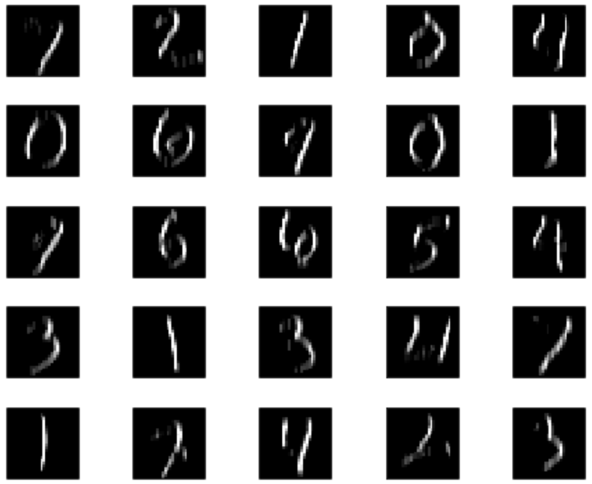
Feature map

Filters

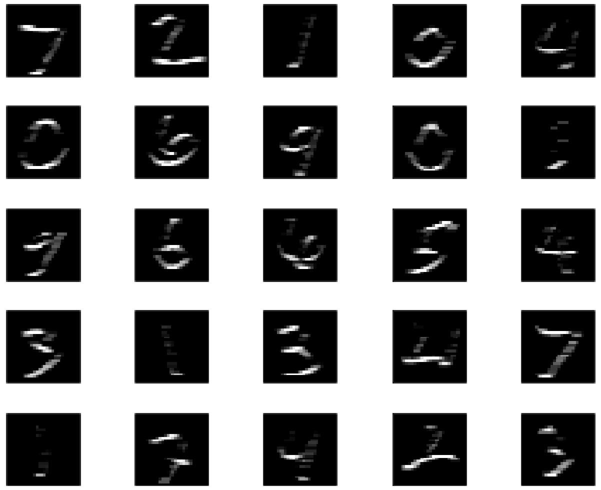
Images



0	-1	1	0
0	-1	1	0
0	-1	1	0
0	-1	1	0

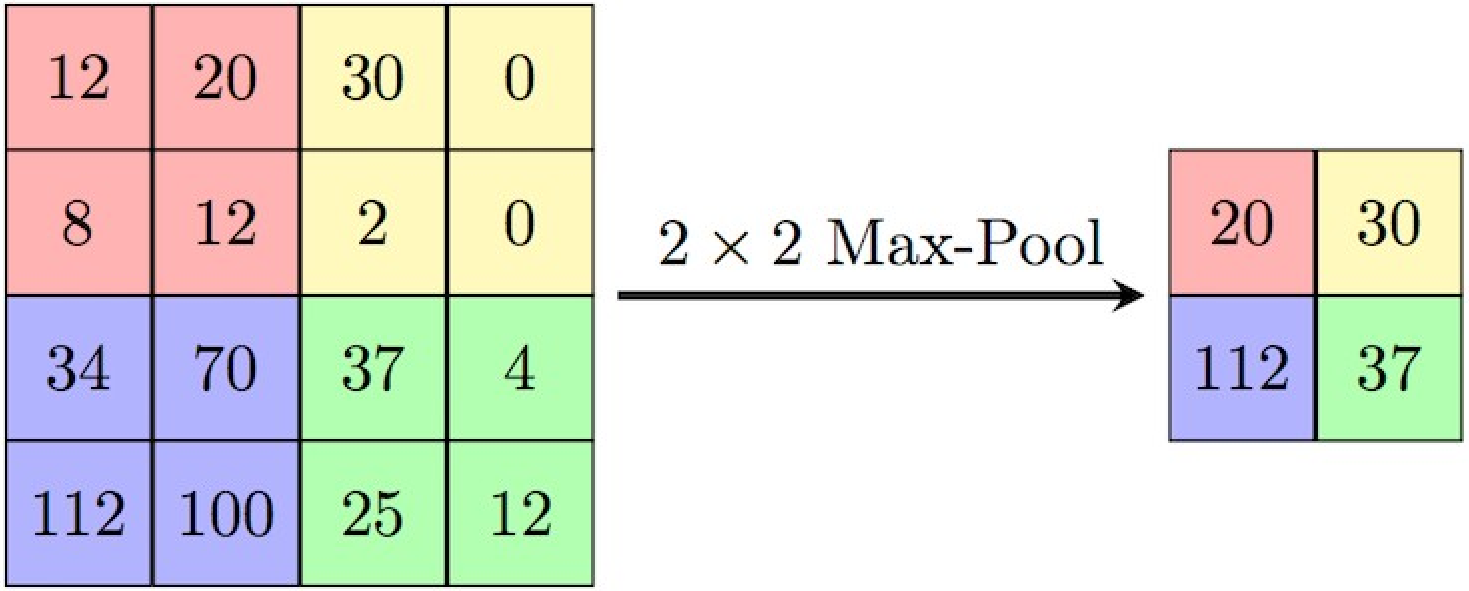


0	0	0	0
-1	-1	-1	-1
1	1	1	1
0	0	0	0

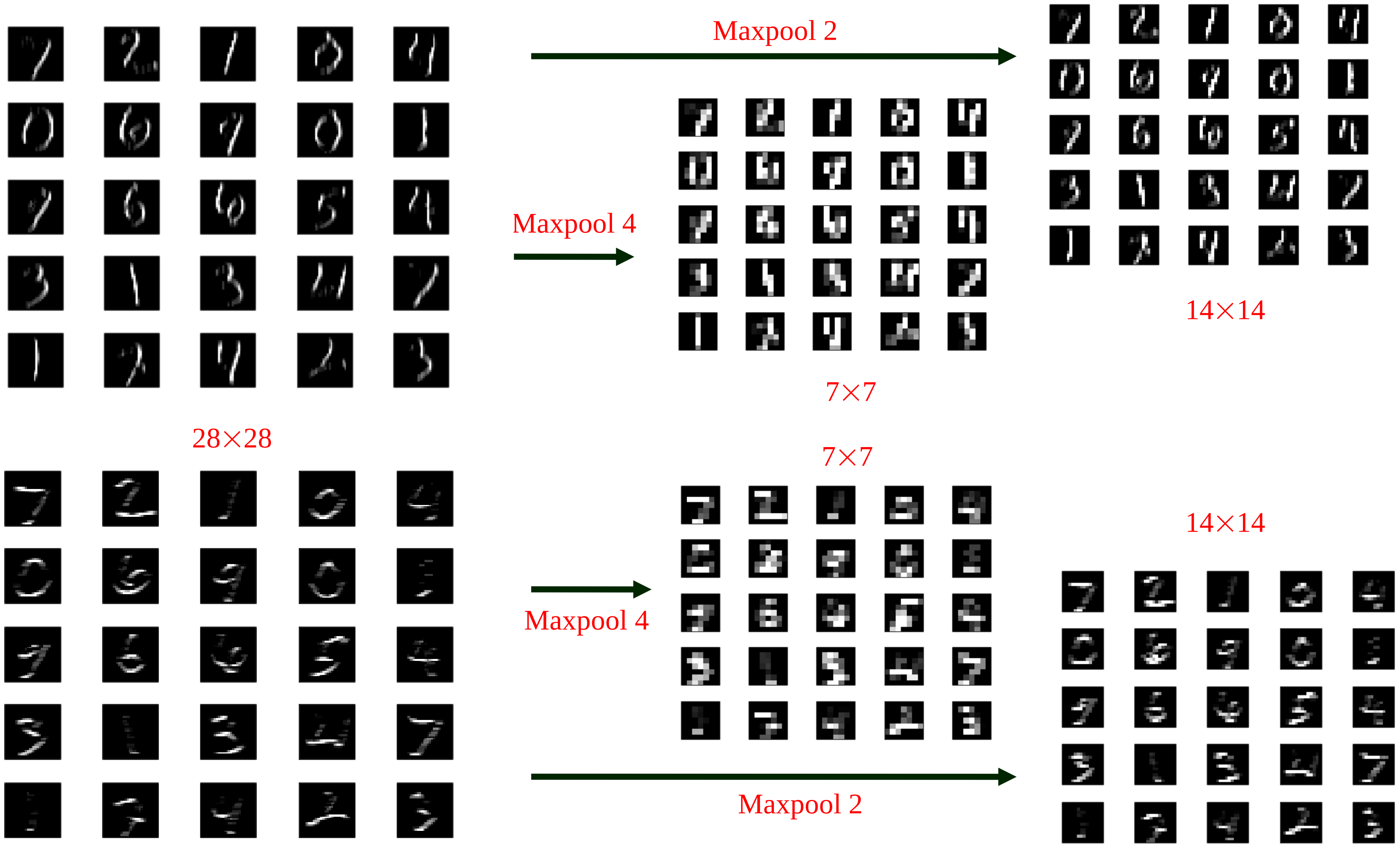


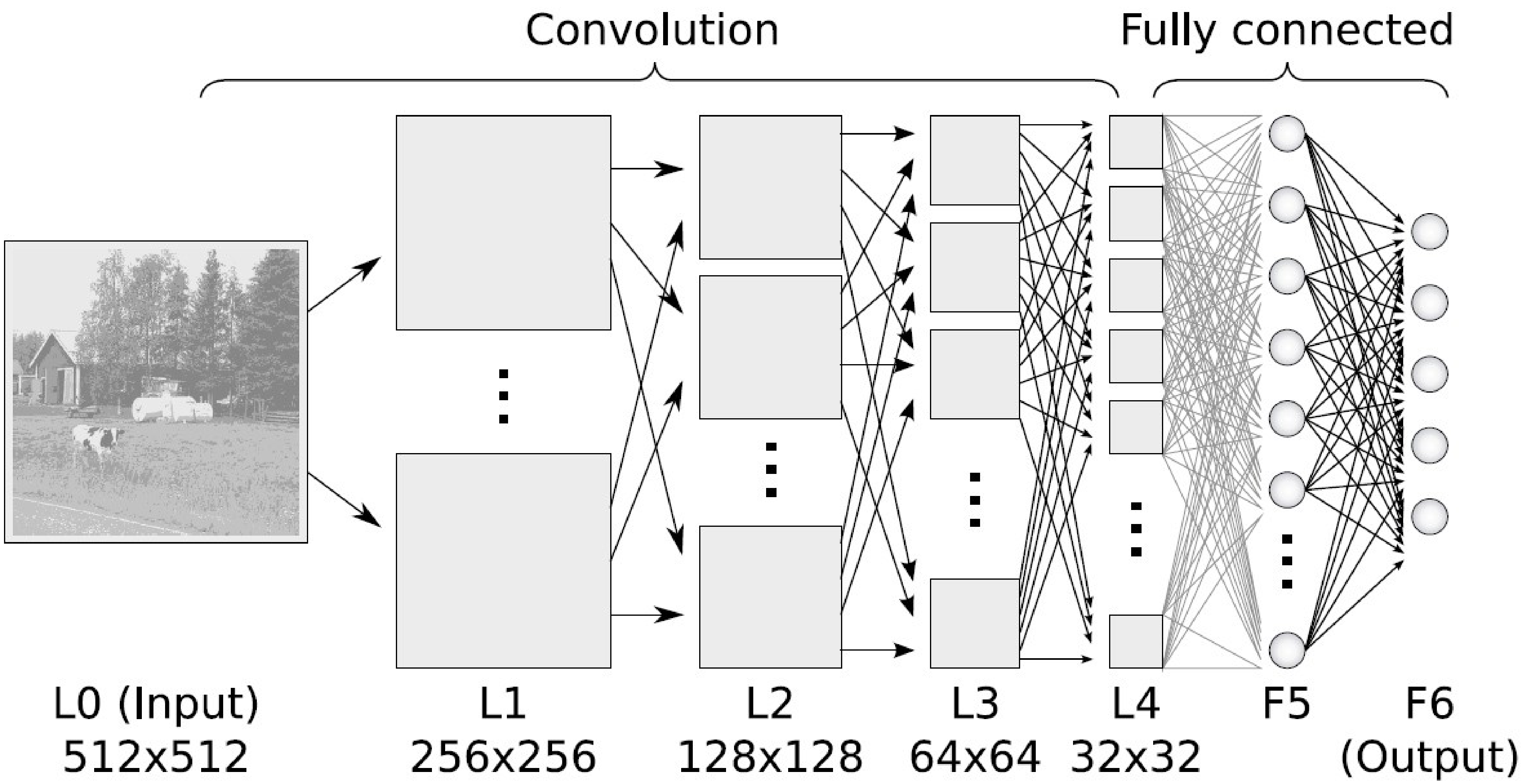
Max Pooling

- Sample *k*×*k* patches of the image in a grid.
- Find the maximum pixel value in each patch.
- Output a smaller image where each patch is replaced by a single pixel with the max value from the patch.



Source: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling


Maxpool 2
Maxpool 4
14×14
7×7
28×28
7×7
14×14
Maxpool 4
Maxpool 2



https://www.ais.uni-bonn.de/deep_learning/images/Convolutional_NN.jpg

Belief Networks

The Inference Problem:
Inferring the state of the hidden variables

The Learning Problem:
Finding a set of interactions (weights) among the variables that make the observed variable states most likely.

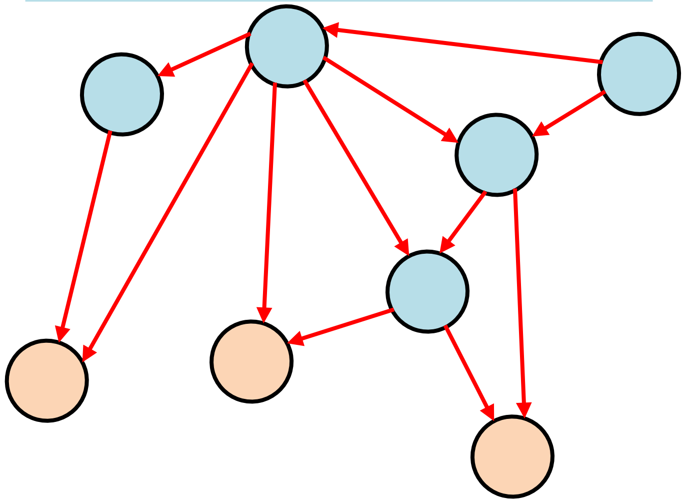
Ideally, we want to infer the joint distribution $P\,(\mathbf{v}, \boldsymbol{h})$

For example, we could try maximizing

$$\log P(\mathbf{v},\boldsymbol{h}) = \log\left(P(\boldsymbol{h}\,|\,\mathbf{v})P(\mathbf{v})\right)$$

but, while estimating $P(\mathbf{v})$ is easy, estimating $P(\boldsymbol{h}|\mathbf{v})$ is not.

Stochastic hidden variables ($\boldsymbol{h}$ = causes)



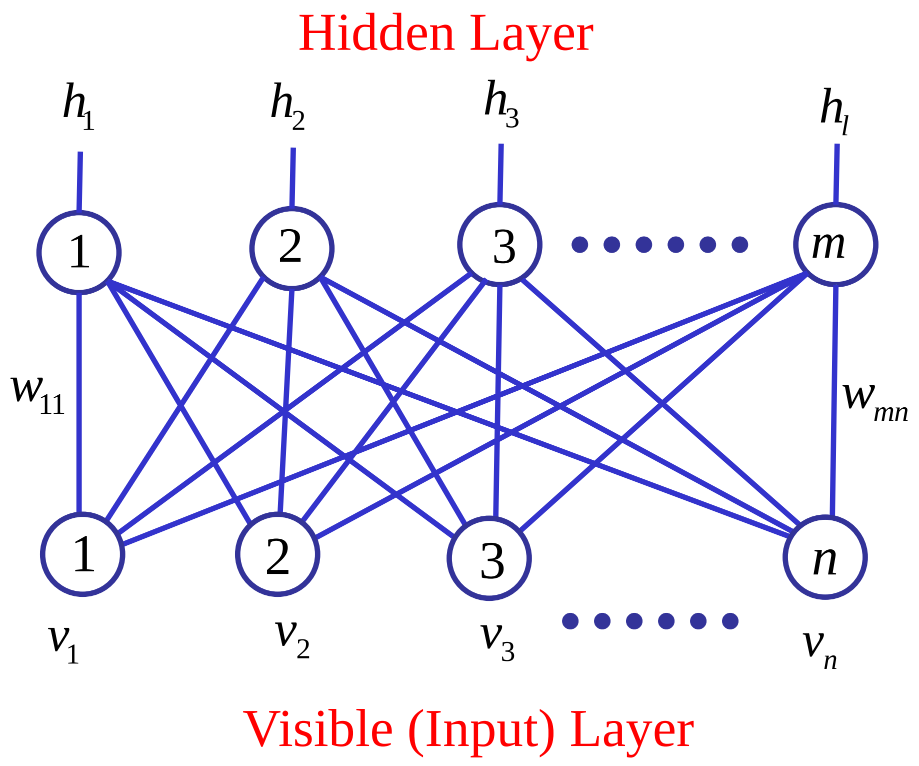
Observed variables ($\mathbf{v}$ = effects)

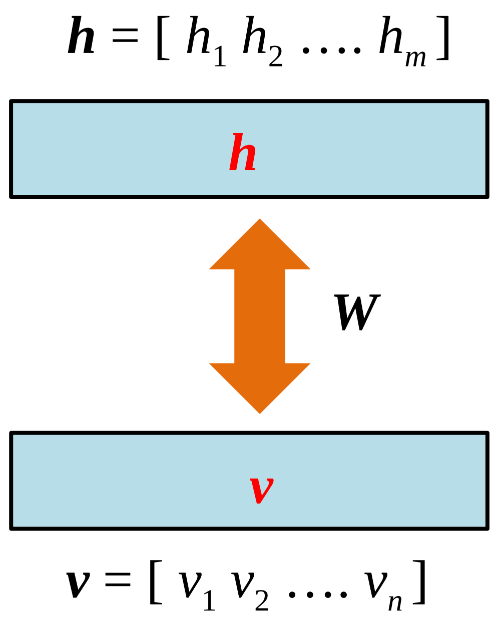
Restricted Boltzmann Machines

An excellent but complex tutorial: https://www.cs.toronto.edu/~hinton/nipstutorial/nipstut3.pdf

Restricted Boltzmann Machine

The Restricted Boltzmann Machine (RBM) is a stochastic 2-layer bipartite recurrent network with symmetric weights: $w_{ij} = w_{ji}$

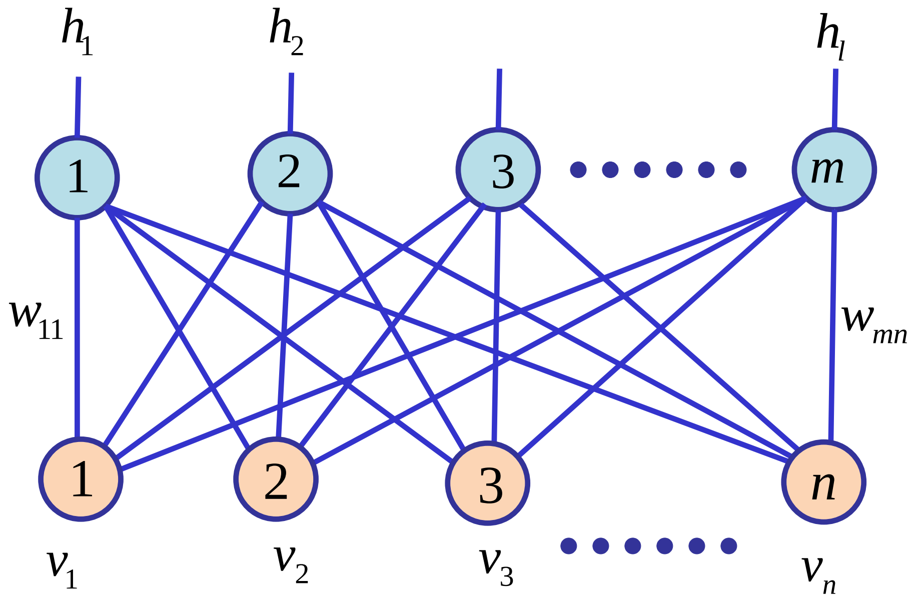



$\mathbf{h} = [\ h_1\ h_2\ \ldots.\ h_m\ ]$
$\boldsymbol{h}$
$\mathbf{W}$
$\boldsymbol{v}$
$\mathbf{v} = [\ v_1\ v_2\ \ldots.\ v_n\ ]$

RBMs are used to infer latent (hidden) variables

For a nice introduction, see:
https://towardsdatascience.com/deep-learning-meets-physics-restricted-boltzmann-machines-part-i-6df5c4918c15

Binary (0/1) RBM



Hidden neuron *i*:

$$s_i^h(t) = \sum_{j=1}^{n} w_{ij} v_j(t) + b_i$$ net input to *i* at step *t*

$$p_i^h(t) = \frac{1}{1 + \exp\left(-s_i^h(t)\right)}$$ probability of *i* being active at step *t*

$$h_i(t) = \begin{cases} 1 & \text{with probability } p_i^h(t) \\ 0 & \text{else} \end{cases}$$

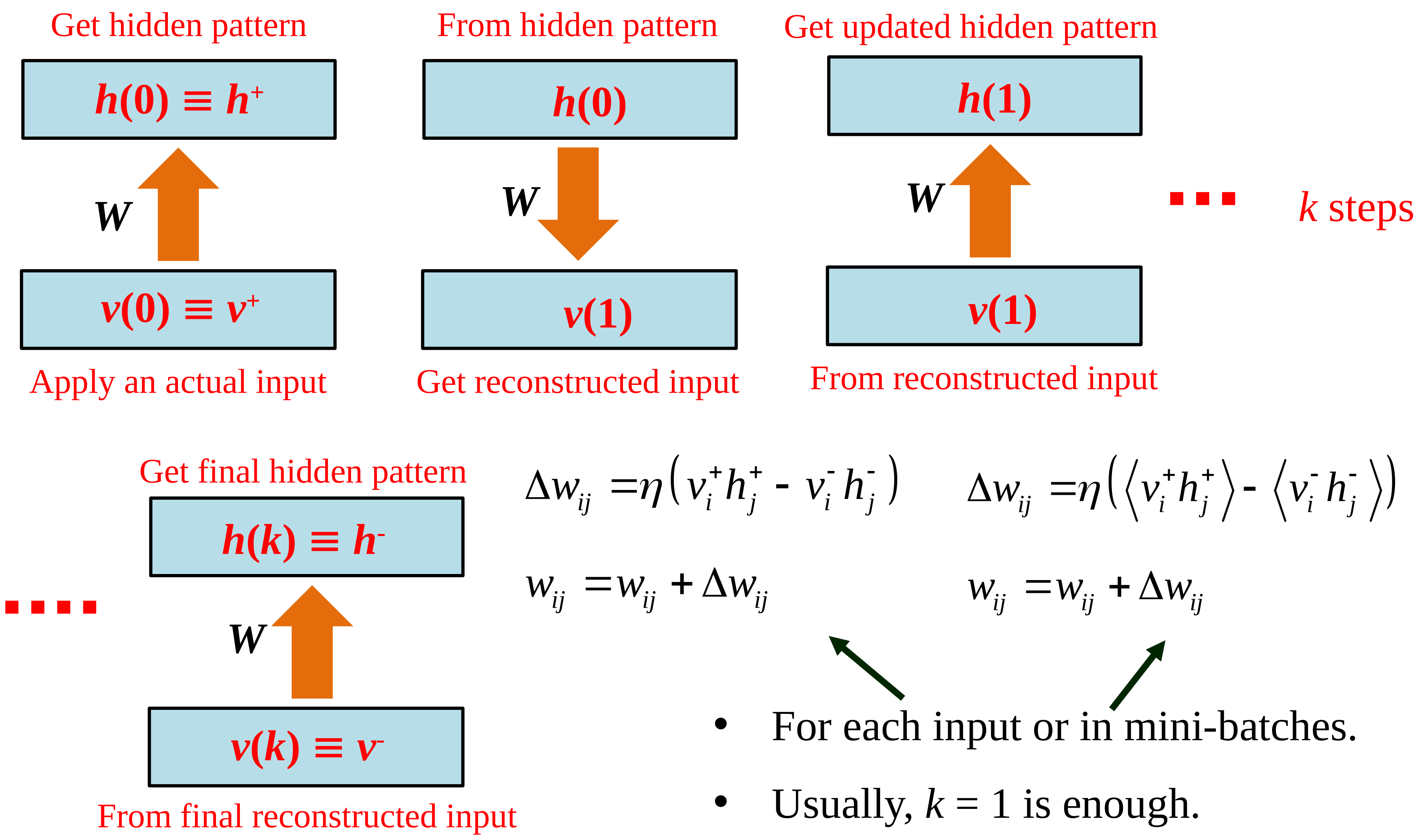
Visible neuron *j*:

$$s_j^v(t) = \sum_{i=1}^{m} w_{ji} h_i(t-1) + c_j$$ net input to *j* at step *t*

$$p_j^v(t) = \frac{1}{1 + \exp\left(-s_j^v(t)\right)}$$ probability of *j* being active at step *t*

$$v_i(t) = \begin{cases} 1 & \text{with probability } p_j^v(t) \\ 0 & \text{else} \end{cases}$$

Training with Contrastive Divergence



$$\Delta w_{ij} = \eta\left(v_i^+ h_j^+ - v_i^- h_j^-\right)$$

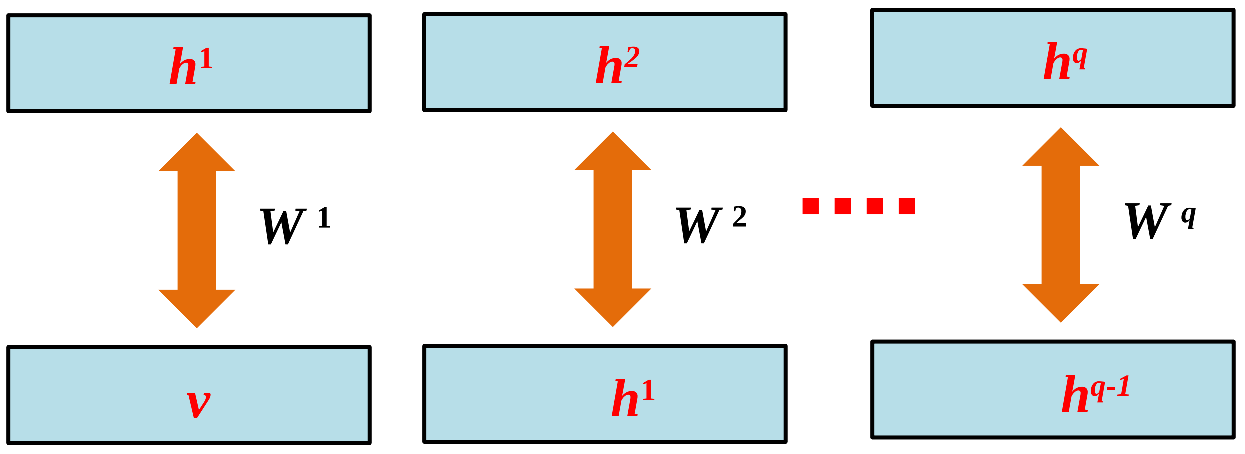
$$w_{ij} = w_{ij} + \Delta w_{ij}$$

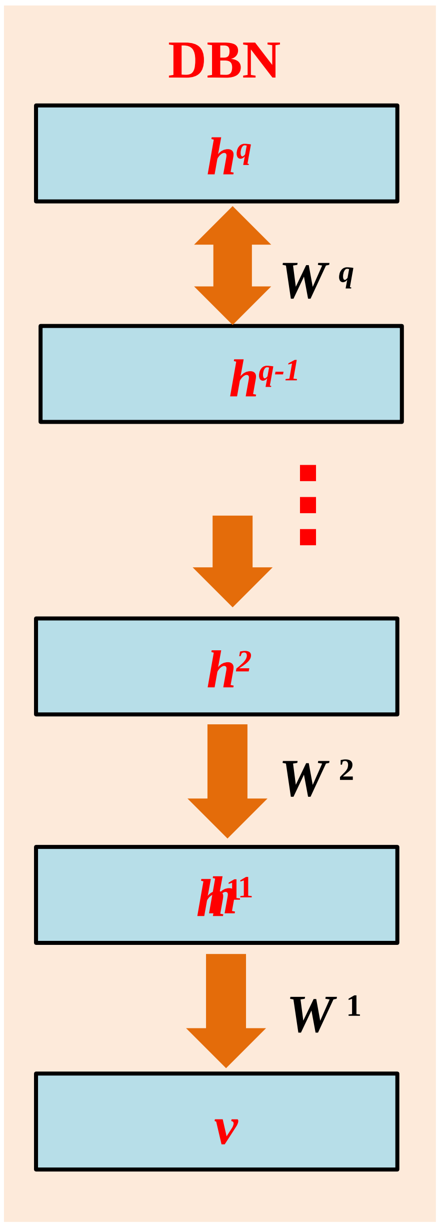
$$\Delta w_{ij} = \eta\left(\langle v_i^+ h_j^+ \rangle - \langle v_i^- h_j^- \rangle\right)$$

$$w_{ij} = w_{ij} + \Delta w_{ij}$$

- For each input or in mini-batches.
- Usually, $k = 1$ is enough.

Deep Belief Networks

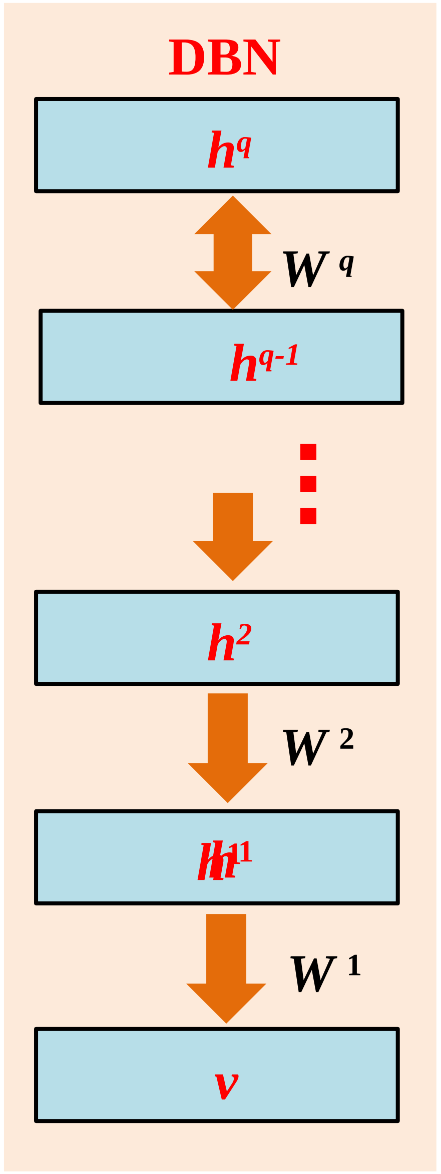




- The $\boldsymbol{h^q}$ pattern is a deep latent representation of the input.
- The DBN is a ***generative model*** – it generates new data that follows the distribution of the training data.

https://www.cs.toronto.edu/~hinton/nipstutorial/nipstut3.pdf

How does a DBN work?

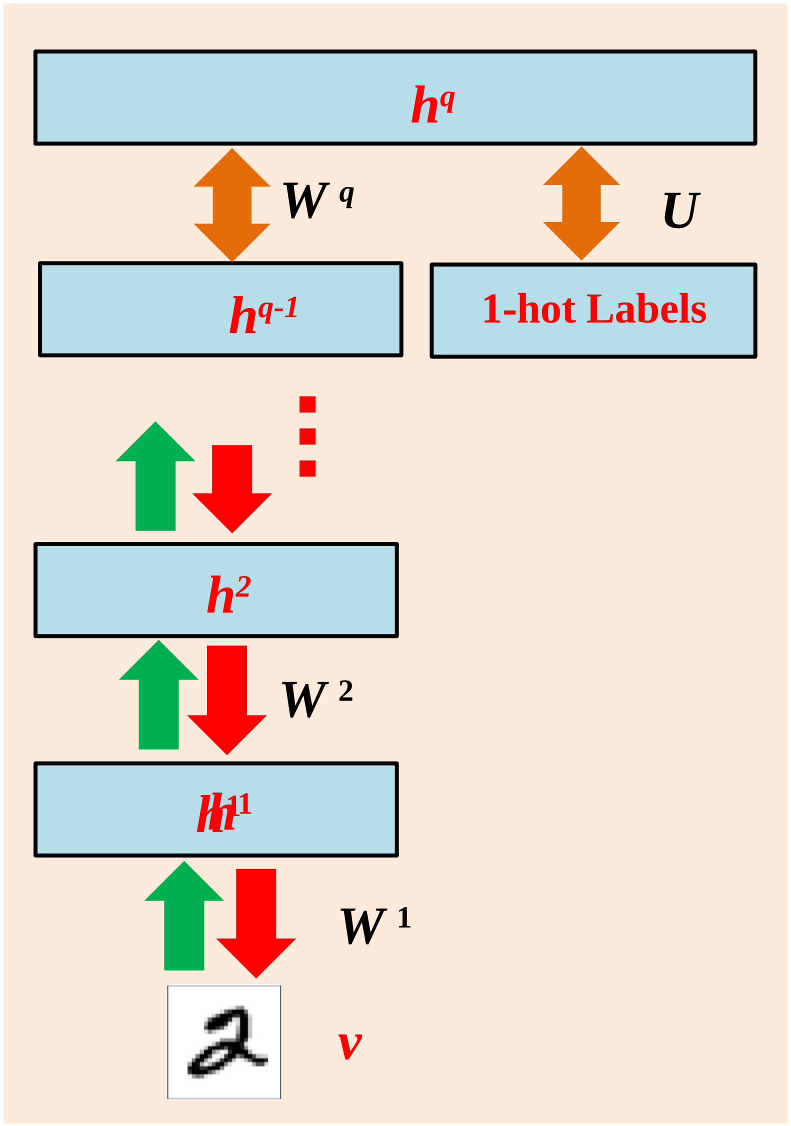


- Randomly initialize the top two layers $\boldsymbol{h}^q$ and $\boldsymbol{h}^{q-1}$
- Iterate these two layers until convergence (Gibbs sampling)
- Propagate the activity down from $\boldsymbol{h}^{q-1}$ to $\boldsymbol{v}$ to obtain a sample of the observable variables.

Why is this useful?

1. It explains where the observable variables come from.
2. It provides a deep set of latent causes for the observed data.
3. The latent variables can be used as features for further inference about the observable data.

An Example: MNIST Classifier



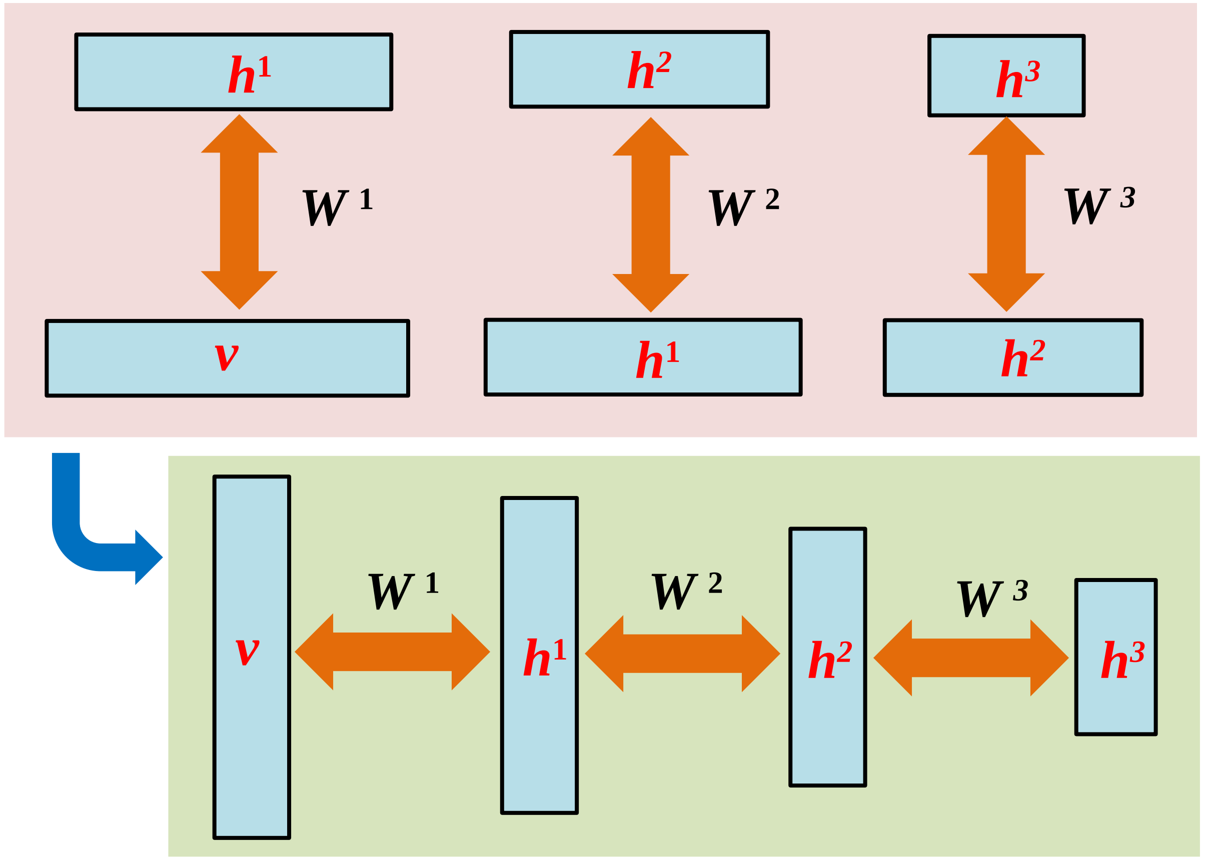
- Randomly initialize Labels layer.
- Present data input and propagate up to h^q
- Iterate the recurrent associative memory at the top until convergence.
- Read the inferred label from the Labels layer (label can be seen as a "corrected" part of the [h^{q-1} h^q Label] memory vector)

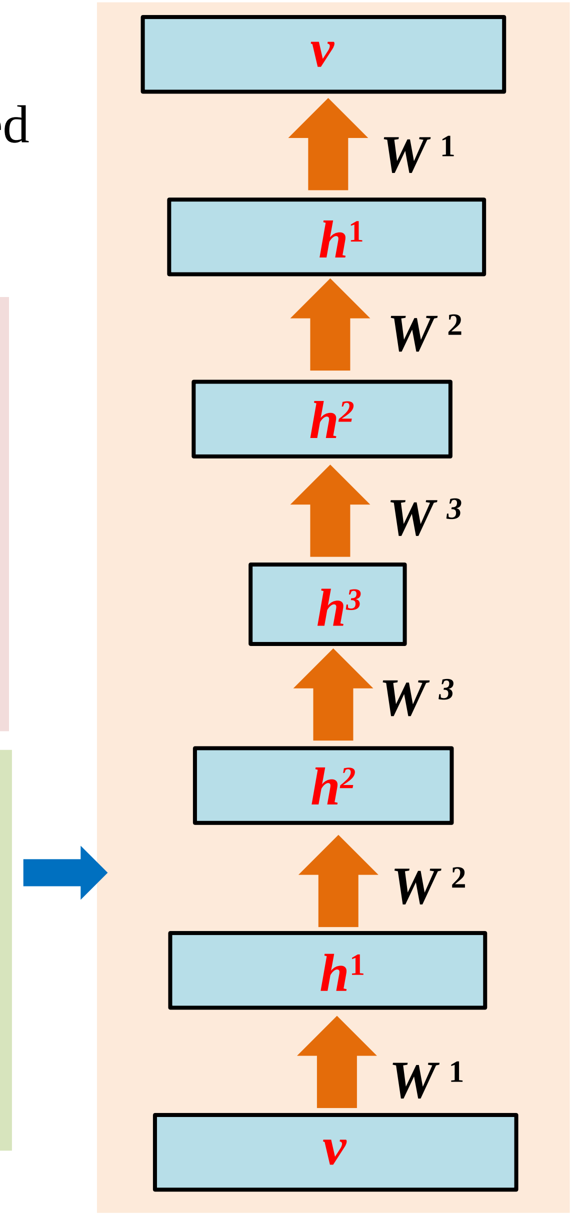
We can also "verify" the label by then running the DBN down from to see if the right class of digit is generated.

This allows us to generate novel samples of each digit, i.e., digits that the network "believes" are 2s or 7s or 0s.

Deep Autoencoders from DBNs

Deep autoencoders can be learned in an unsupervised way by training several RBMs, stacking them, and "unfolding" the stack.





Detailed tutorial:
https://www.cs.toronto.edu/~hinton/nipstutorial/nipstut3.pdf