Neural networks:
Deep neural networks and deep learning

Prof. Dr. Sc. Sven Loncaric
Prof. Dr. Sc. Marko Subasic

Faculty of Electrical Engineering and Computing
University of Zagreb

http://www.fer.hr/predmet/neunet_a

Why LARGER networks?

- Greater network capacity can be achieved by increasing the number of neurons
- So it is in the living world
  - Simple organisms are several hundred neurons in the brain
  - Man 86 billion neurons
  - Elephant 257 billion neurons - not all in the number of neurons
- Network architecture is also important
- Research on NM architectures is still very current (with learning algorithms)

Why "deep" networks?

- It can be shown to be more effective if the increase is achieved by using more layers with fewer neurons versus fewer layers with more neurons
- The human brain has a similar structure
- We can’t talk about feed forward layers because there are feedback connections as well
- Human cognitive processes are often hierarchically organized and "deep"

Is that a new idea?

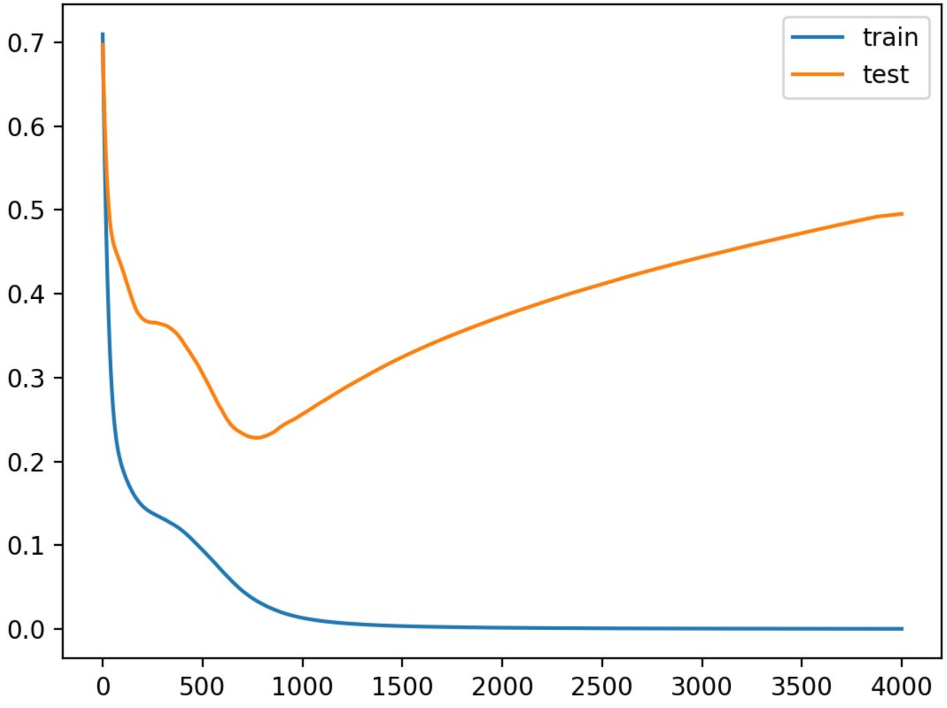
- Multilayer networks have existed before
  - MLP
- The idea of deep networks dates back to the 1980s
  - Training was too slow
    - In the meantime, the reasons have been identified
    - Part of the reason was reversed by hardware advances
    - Part of the reason was eliminated by new algorithms

Is this a new idea?

- Supervised learning and error backpropagation were mainly used
  - Like today
- Networks with more hidden layers have been shown to perform worse in both training and testing
  - Contrary to expectations
  - What is the reason?

Old problems I

- Overfitting - The network learns a random error or noise instead of essential input data relationships



Old problems I

- Weak generalization
  - It will not work well on new data
- The network is too complex for a given training set
  - Greater network complexity potentially leads to greater accuracy
  - A compromise is sought between network complexity (accuracy) and overfitting

Old problems I

- Some tricks to avoid overfitting have also been devised

1) Early stopping
   - Not desirable
2) Reducing network complexity
   - Not desirable
3) Increasing the amount of input data
   - It is not always possible

Old problems I

4) Training set augmentation

- Artificial increase of variability in the input set
- Adding <u>noise</u>

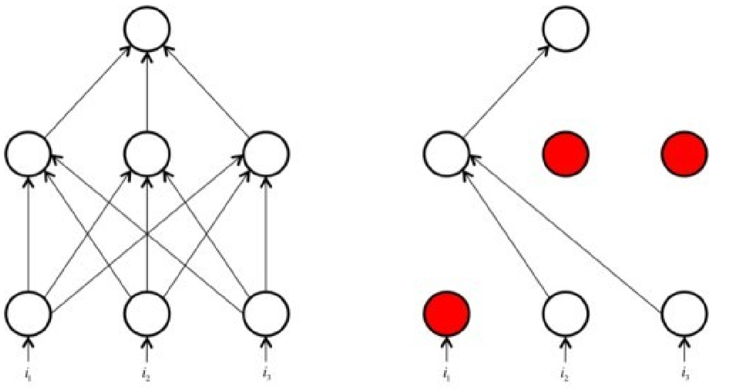
5) Regularization

- Additional member in the goal function
- It effectively reduces network complexity - hopefully in a good way
- Eg. L2 regularization - punishes heavy weights
  - Intuitive - encourages the network to use all inputs equally
  - Weights tend toward zero

Old problems I

6)Dropout

- Extinguishing individual neurons
- Adding noise



7)Choosing another architecture

- Use other people's experiences and intuition

- Exaggerating with tricks usually leads to poor network performance
  - How much is enough?

Old problems II

- Backpropagation is based on gradients
  - they are key to training
  - Propagation of gradients into initial/lower layers

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial \nu_j(n)} \frac{\partial \nu_j(n)}{\partial w_{ji}(n)}$$

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)$$

$$\delta_j(n) = \varphi_j'(\nu_j(n)) e_j(n)$$

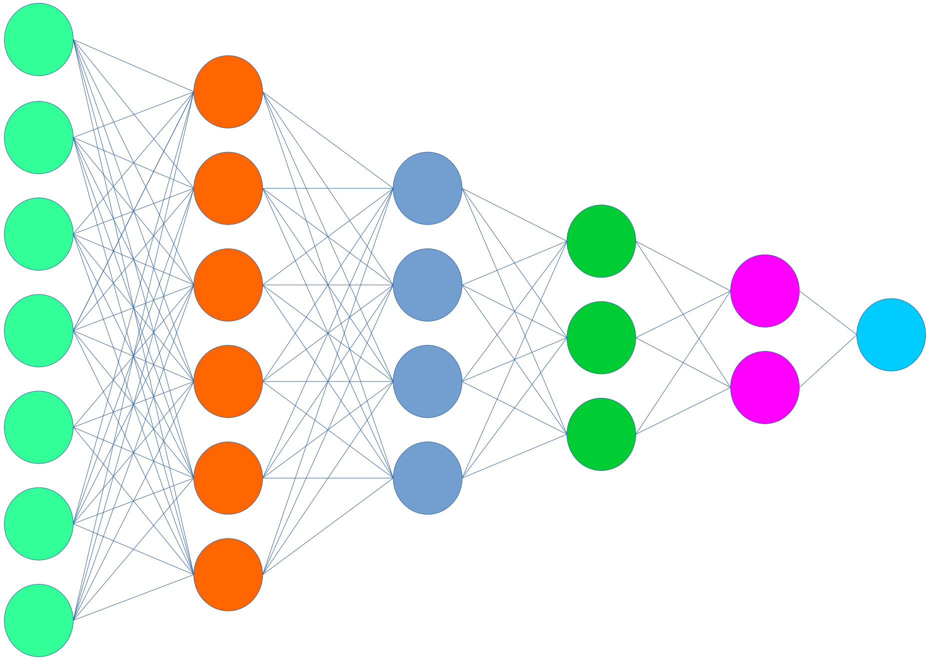
$$\delta_j(n) = \varphi_j'(\nu_j(n)) \sum_k \delta_k(n) w_{kj}(n)$$

Old problems II

- What do we want at the end of training?
  - That the gradients for each network parameter fall to 0
    - Then the network converged and further changes in network parameters do not minimize the cost function
    - We want to stop when the network has learned what it needed to - the end of training
- What do we not want at the beginning (or middle) of training?
  - That the gradients for each network parameter fall to 0
    - Effectively stops training
  - That the gradients for each network parameter be very large
    - Instability

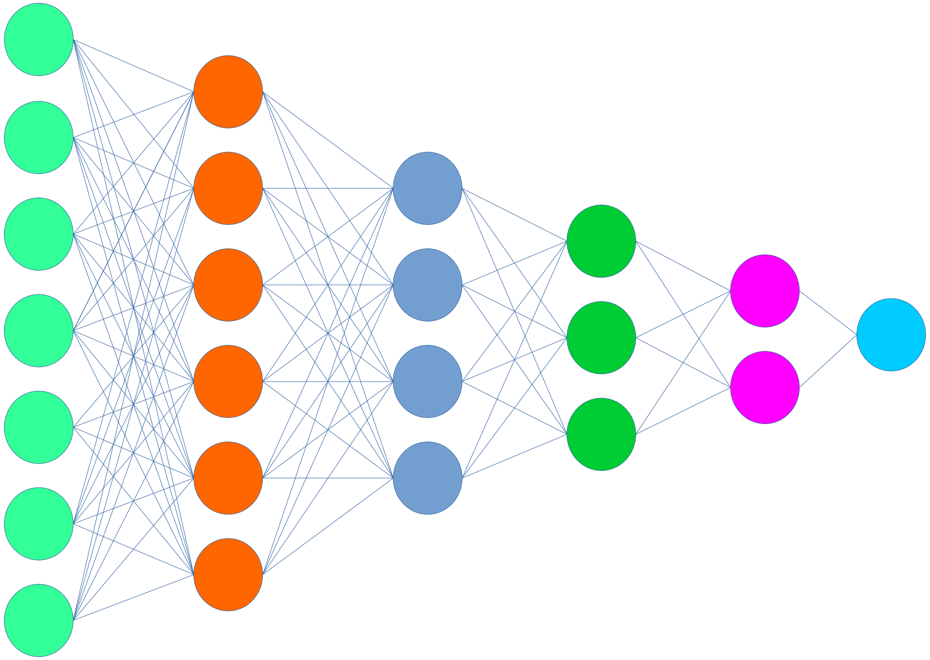
Old problems II

- Vanishing gradients - as the number of layers increases, the error backpropagation becomes worse in passing information to lower layers
- The deeper layers near the entrance are difficult to learn
  - Gradients are not propagated into the initial layers
- The problem is significant when random initialization is used
- It does not necessarily mean the end of training, but training is very long



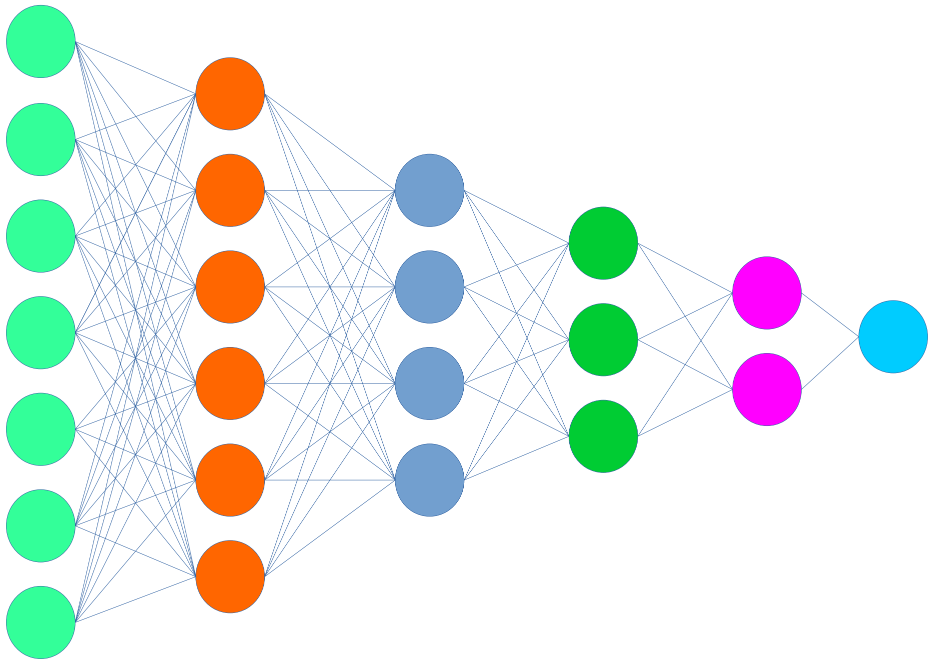
Old problems II

- Exploding gradients - the correction is too large so it does not reduce the error
- Similar effect to vanishing gradients
- It is important that the initial layers work well



Old problems II

- It is possible that some of the last hidden layers of the deep network have enough capacity to model a given problem
- The previous layers are then not needed and add noise to the input data
- A deep network works similarly to a shallow network

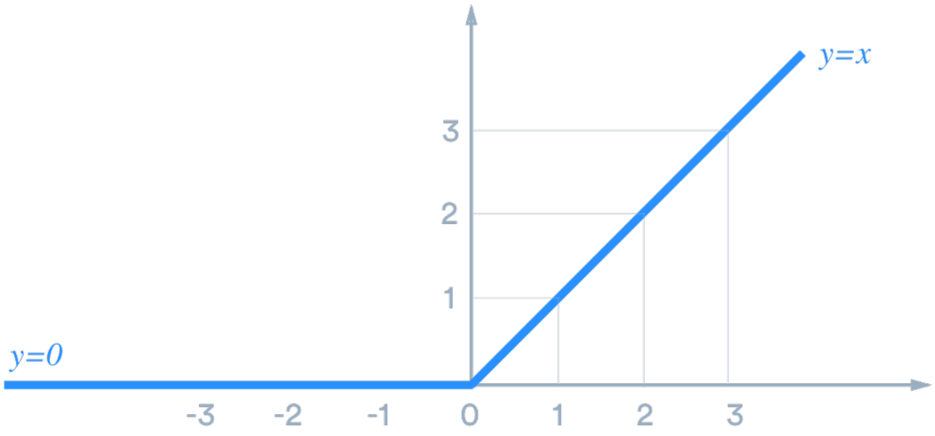


Old problems II

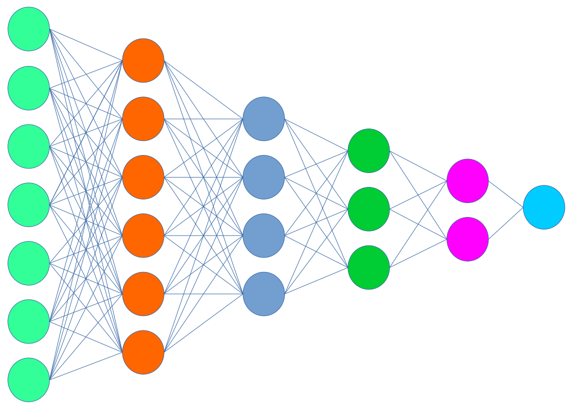
- Tricks designed to avoid vanishing gradients:
  - Better random initialization of network parameters
    - Targeting avoidance of small gradients
  - Using pre-trained layers
    - Pre-training on another problem
      - Eg. unsupervised learning (auto encoders)

Old problems II

- Training layer by layer
- Different activation functions
  - ReLU



- The "source" of the training gradients closer to the input layer
  - Eg. at the autoencoder

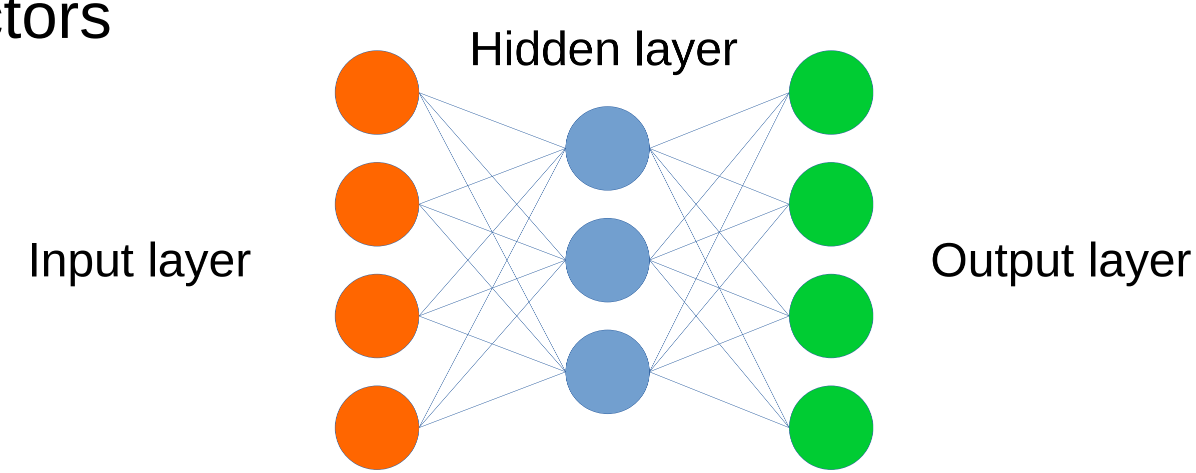


What else helped

- Cheap and affordable computing power from GPUs
- Increasing number of situations where large amounts of raw data need to be efficiently analyzed

Autoencoders

- Feedforward network that learns how to compress (encode) input vectors



- Present the information present in the input vector with a small number of elements (hidden layer nodes) from which a perfect reconstruction of all elements of the input vector is possible
- It needs to be taught to encode vectors from the training set

Autoencoders

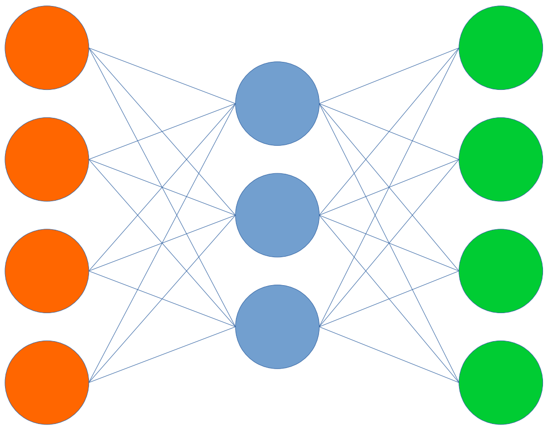
- The network needs to learn the internal structure of the data and the essential features - the essence of the data at the input
- The hidden layer is called the feature detector
- The number of neurons in the hidden layer is less than the number of neurons in the input layer
- The network is forced to find only essential features to achieve dimensionality reduction
- A good approach to achieving generalization

Autoencoders

- Supervised training - eg backpropagation
- This is not "real" supervision
  - An unknown function is not learned
  - It is checked whether the output is the same as the input
- In the literature, such training is considered unsupervised

Autoencoders

- There are numerous variations of the basic autoencoder
- They focus on the central hidden layer and the extraction of essential features in it
  - Mechanisms of regularization
  - Sparsity, robustness to noise, missing inputs, restrictions on derivations
  - The size of the layer is then less important



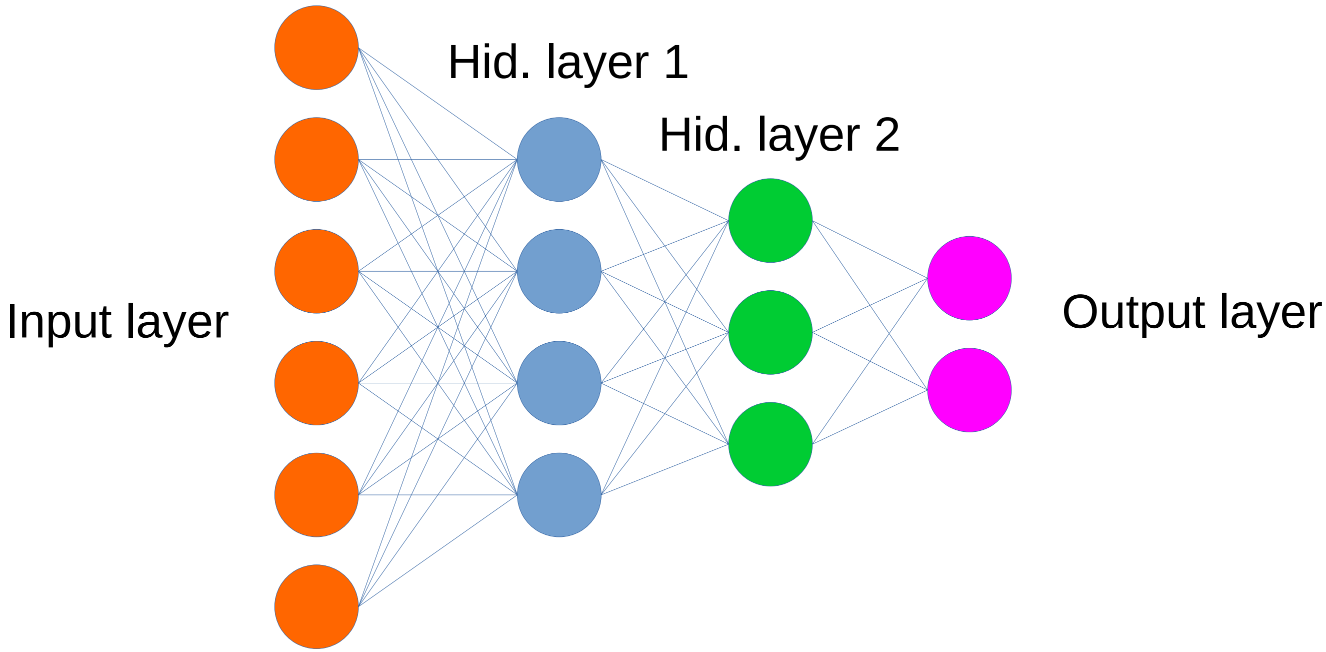
Autoencoders

- Variants:
  - Denoising autoencoder
    - Artificial <u>noise</u> is added to the input data
    - The autoencoder must learn to remove noise
- Contractive autoencoder
  - Avoiding changes in hidden layer for small input changes (small change = noise?)
  - An additional member of the derivation-based price function
  - Refers to the central hidden layer
    - Good for a fading gradient

Deep networks

- Autoencoders can function as feature detectors
- Detected features are "hidden" in the hidden layer - not directly usable
- such networks can be concatenated on top of each other
  - Greedy training layer by layer
  - The problem of vanishing gradients and overfitting is reduced

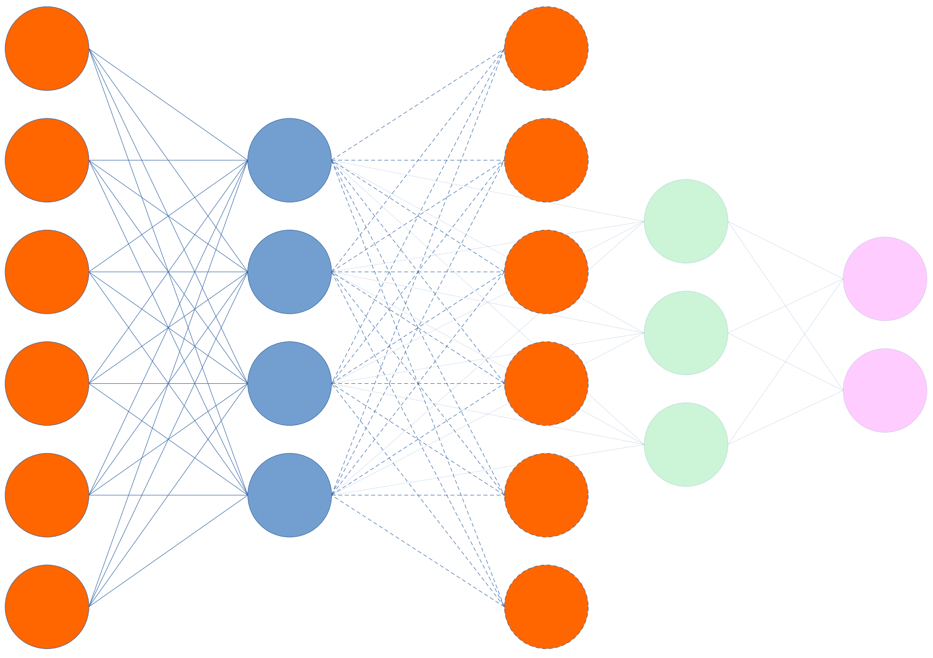
A series of autoencoders



- The input to the next autoencoder is the output from the previous one

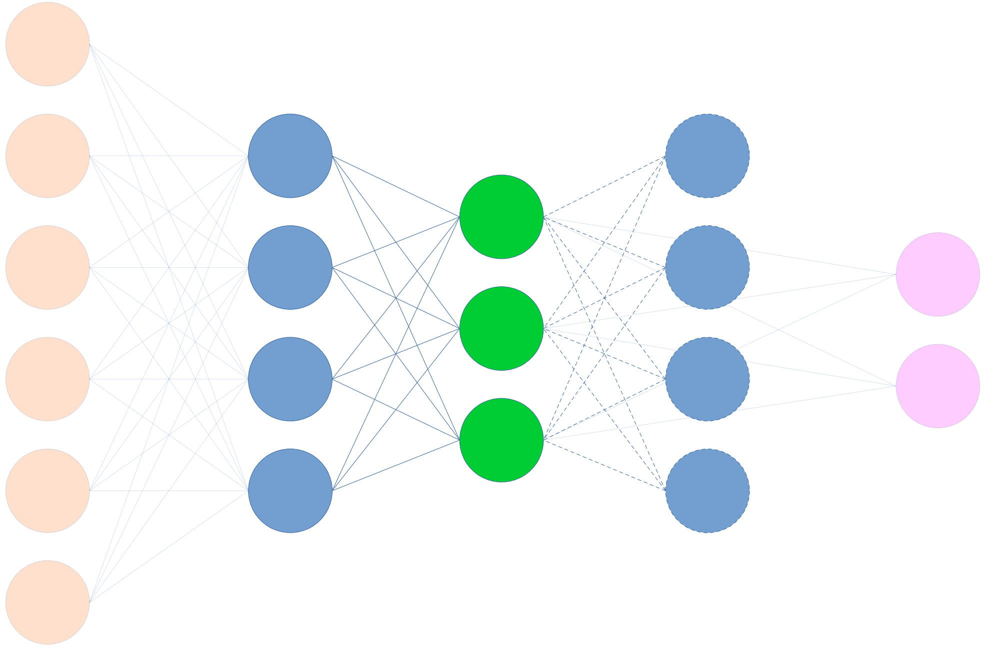
A series of autoencoders

- Training
  1) Training the first autoencoder
     - With temporary output layer
     - Using all the training data
     - Backpropagation algorithm



A series of autoencoders

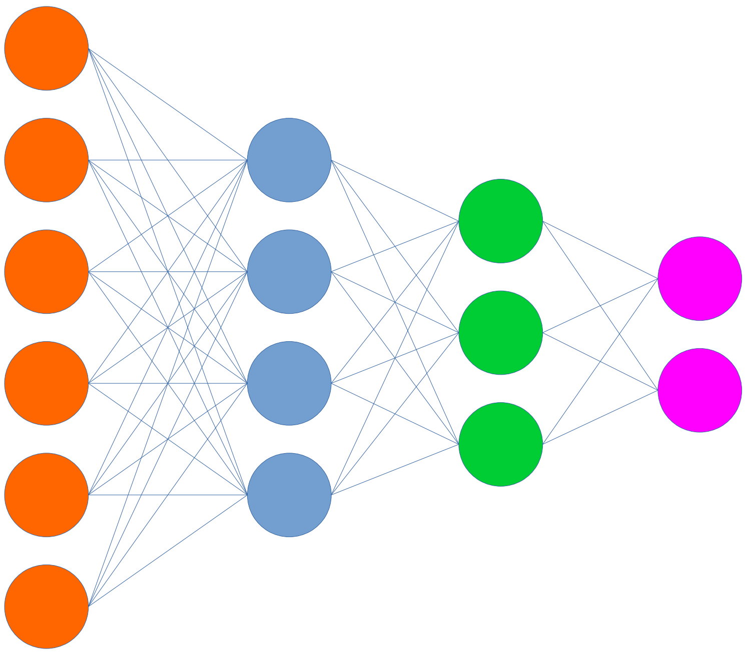
- Training
  2)Training of second autoencoder
  - With temporary output layer
  - Using all the training data
  - The inputs are the outputs of the previously trained autoencoder
  - Backpropagation algorithm



A series of autoencoders

- Training

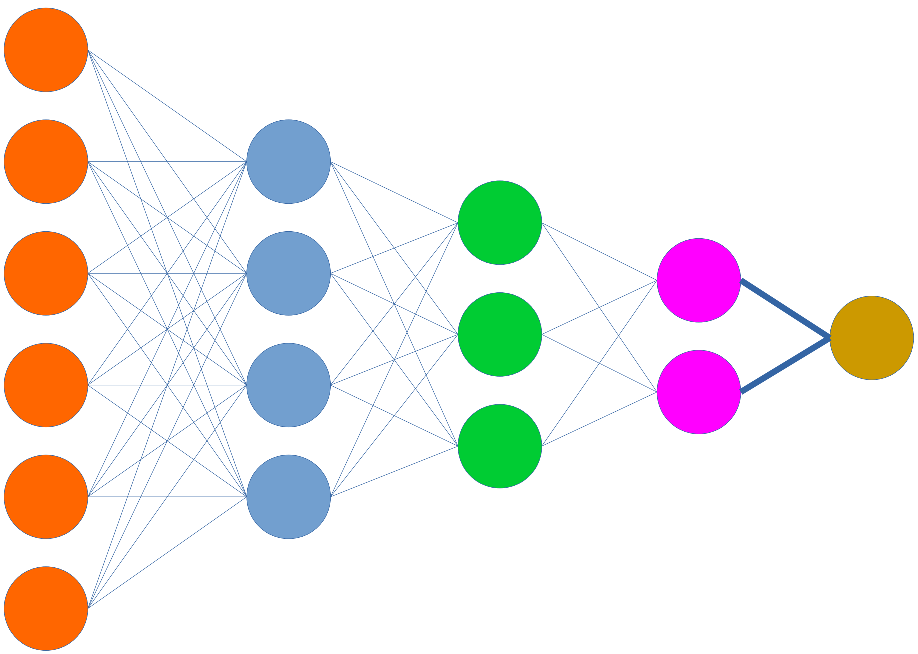
3)Repeat step 2) for all remaining hidden layers / autoencoders



A series of autoencoders

- Training

    4) Add one or more top layers as needed

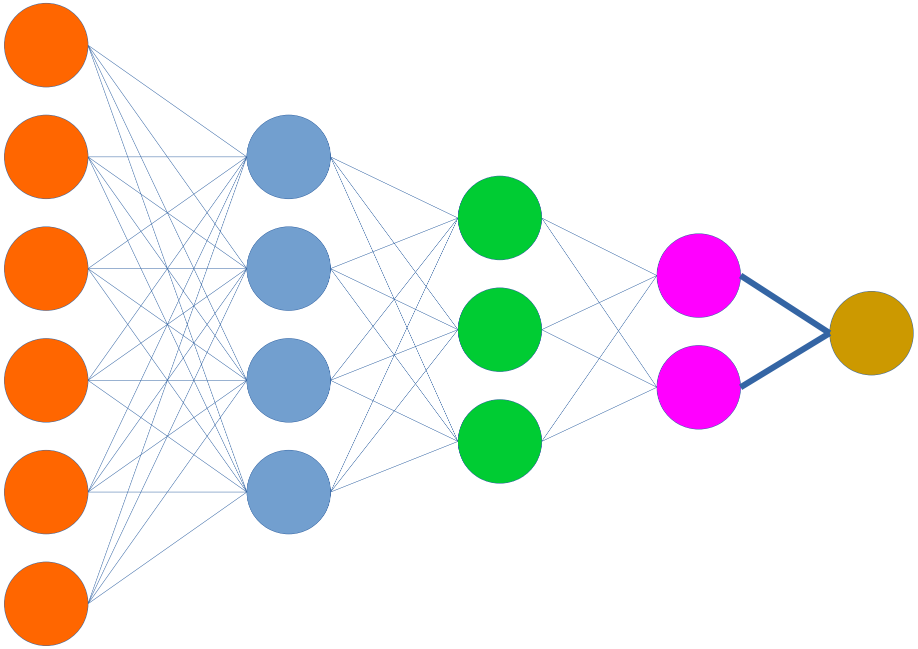


A series of autoencoders

- Training

    5)Train the final MLP using the backpropagation algorithm

    - This step involves modifying all the weights in the network
    - All weights obtained in the previous steps become the initial weights for the final MLP
    - A necessary step to connect the features learned in the previous steps to the desired network output
    - Fine-tuning
    - Steps 1-3 are "pre-training"

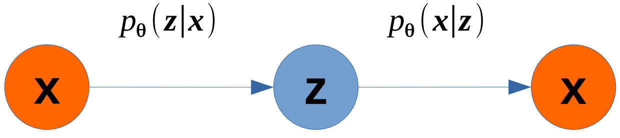


A series of autoencoders

- The well-known fact (problem) is exploited that the final result of backpropagation training depends on initialization
- The first layer teaches low-level features
  - Eg. edges in the figure
- The second layer teaches more complex features
  - Eg. combinations of low-level features
- Higher layers - highr-level features

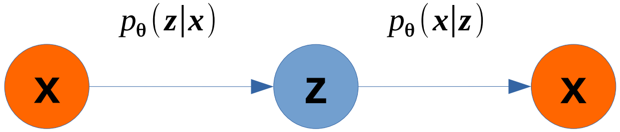
Variational autoencoder

- A more complex idea
- The aim is to learn the distribution of data from the training set p(x)
- Samples can then be generated according to that distribution
- It can be deep
- Although the network is stochastic, back-prop is used for training



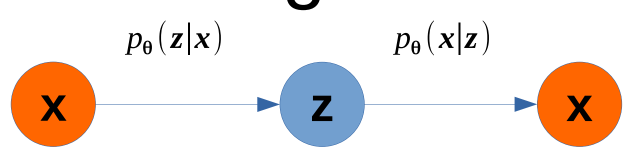
Variational autoencoder

- The hidden layer forms a random vector of a selected distribution into which training patterns are mapped
- From the hidden layer, the distributions of the output elements are determined, on the basis of which the output samples are generated
- Formally, the goal is to maximize the probabilities of training samples



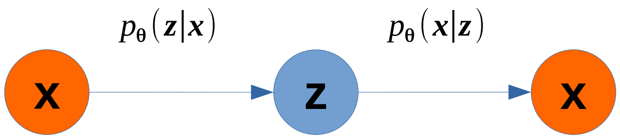
Variational autoencoder

- The goal function has a regularization member that refers to the hidden layer
  - Similar to a contractive autoencoder
  - It enables more efficient training and avoiding the vanishing gradient
  - Minimization of KL divergence
- The hidden layer is stochastic
  - Adding <u>noise</u> to the network
- Much like a denoising autoencoder



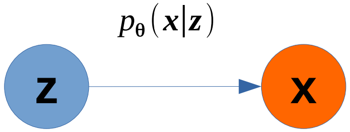
Variational autoencoder

- The algorithm has no hyperparameters
- Although the network is stochastic, back-prop is used successfully



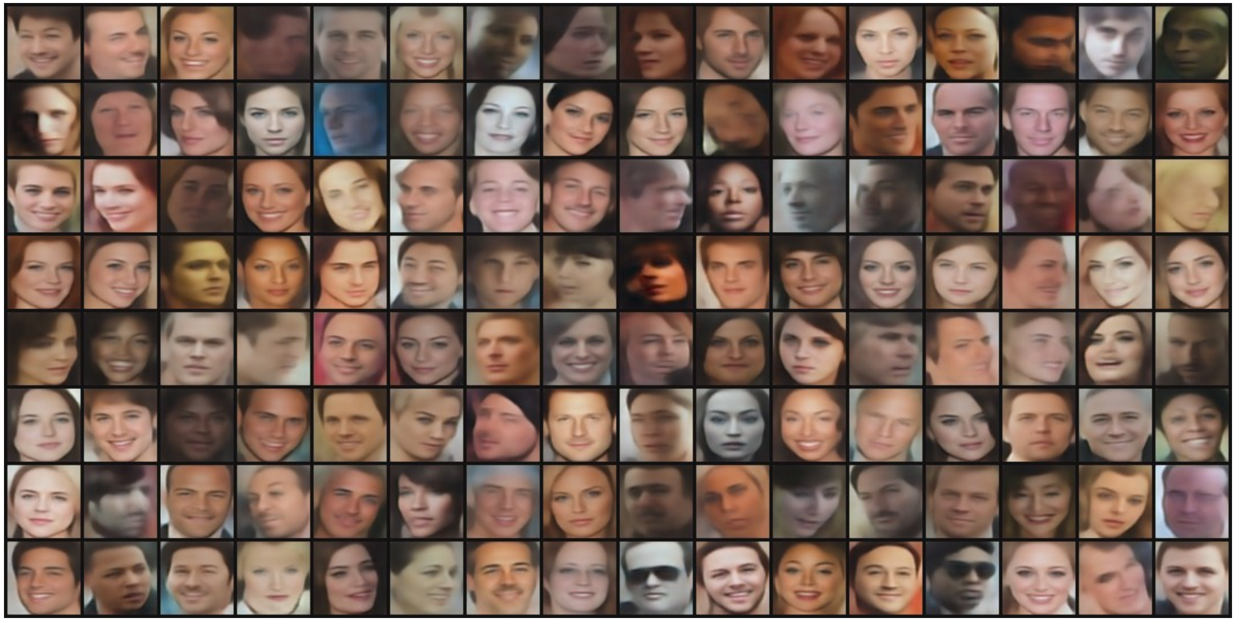
Variational autoencoder

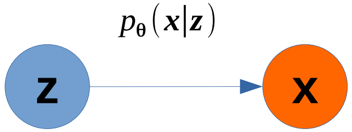
- Such a system allows the generation of new, never-before-seen patterns
  - Generative model
  - The input of the decoder part is a random sample from a selected distribution



Variational autoencoder

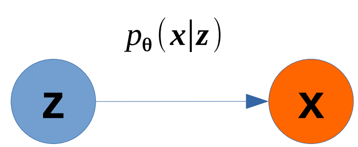
- Examples





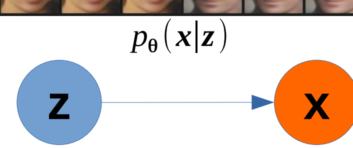
Variational autoencoder

- Examples



Variational autoencoder

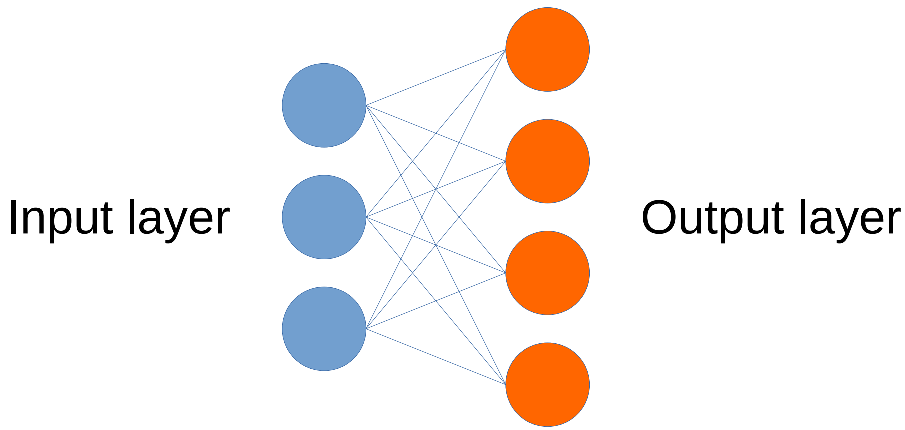
- Individual elements from the hidden layer can represent a single output characteristic
- Variation of this parameter modulates the presence of this characteristic



Restricted Boltzmann Machine

- Restricted Boltzmann Machine (RBM)
- A neural network that can learn the probability distribution in a learning set
- The network is stochastic!

Restricted Boltzmann Machine



- One input and one hidden layer
- Connecting neurons in the same layer is not allowed as with the Boltzmann machine - a restriction
- Full connectivity between the layers
- The connections between neurons are bidirectional and symmetrical

Restricted Boltzmann Machine

- Binary neurons
- Mode of operation:

1) Determine the activation energy

$$a_i = \sum_j w_{ij} u_j$$

2) Set yourself in 1 with probability

$$p_{1i} = \frac{1}{1 + e^{-a_i}}$$

3) or 0 with probability

$$p_{0i} = 1 - p_{1i}$$

Contrastive divergence

- Positive phase
  - Place the sample at the entrance *u*
  - Determine the response of the hidden layer *s*
- Negative phase
  - Determine the response of the input layer *u*' with respect to the response of the hidden layer *s*
  - Determine the response of the hidden layer *s*' with respect to the input *u*'
- Weight correction

$$\Delta w_{ij} = \eta \left(u_i s_j^T - u'_i s'^T_j \right)$$

Contrastive divergence

- Weight modification

$$\Delta w = \eta \left(u_i s_j^T - u'_i s'^T_j \right)$$

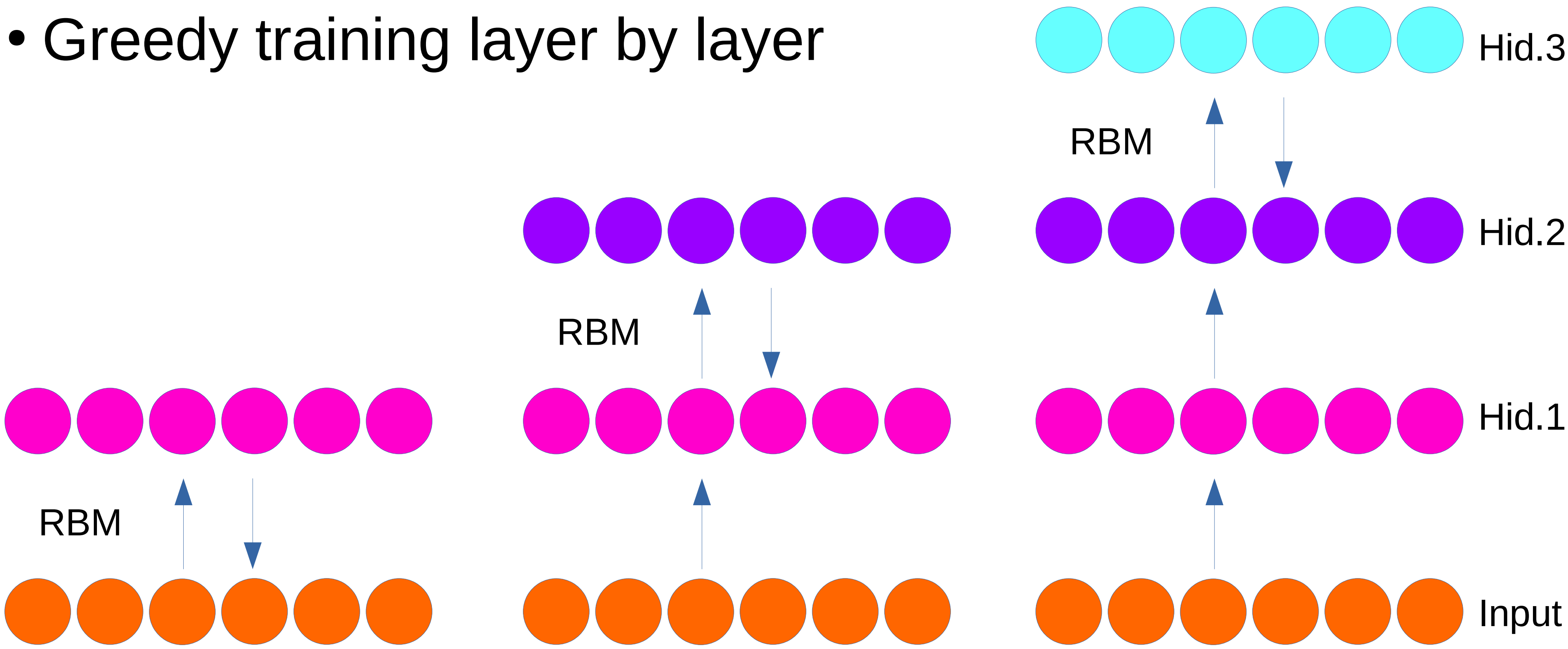
- In the positive phase, the network determines its representation (*s*) of the input data (*u*)
- In the negative phase, the network determines the reconstruction of the input data (*u*') based on its representation (*s*)
- The aim is to achieve similarity between *u* and *u*'
- The network is modified until it reaches the goal

Deep networks

- RBMs can function as feature detectors
- Detected features are "hidden" in the hidden layer - not directly usable
- Such networks can be concatenated on top of each other
  - Greedy training layer by layer
  - The problem of vanishing gradients and overfitting is reduced

Deep Belief Networks

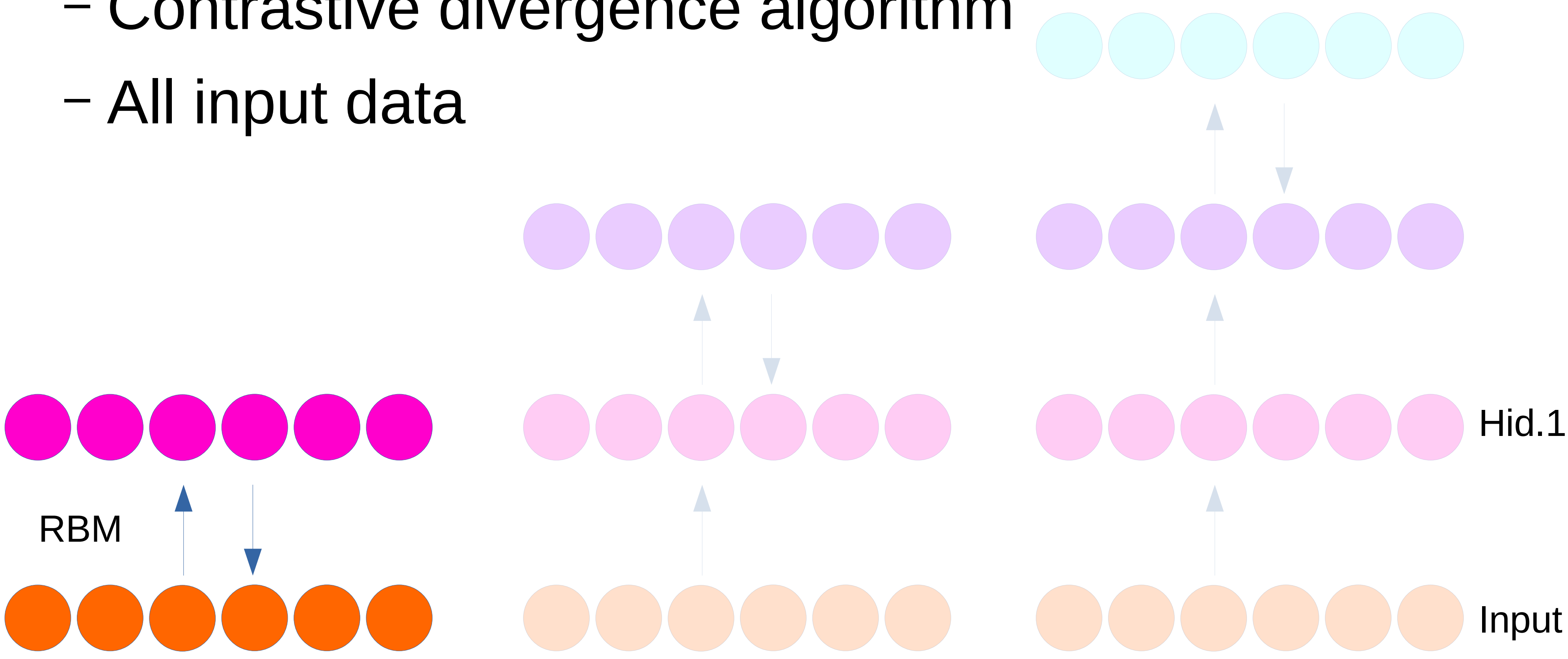
- A series of Boltzman machines
- The hidden layer of the previous RBM becomes the input layer for the RBM in the next layer
- Greedy training layer by layer



Deep Belief Networks

- Training

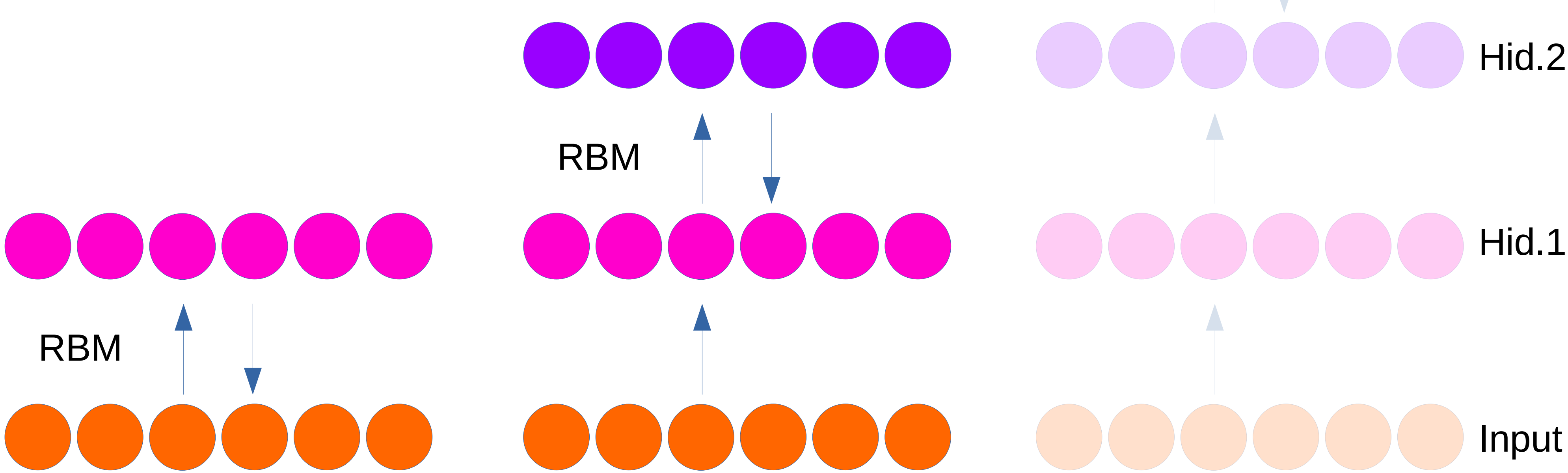
1) Train the first RBM
   - Contrastive divergence algorithm
   - All input data



Deep Belief Networks

- Training

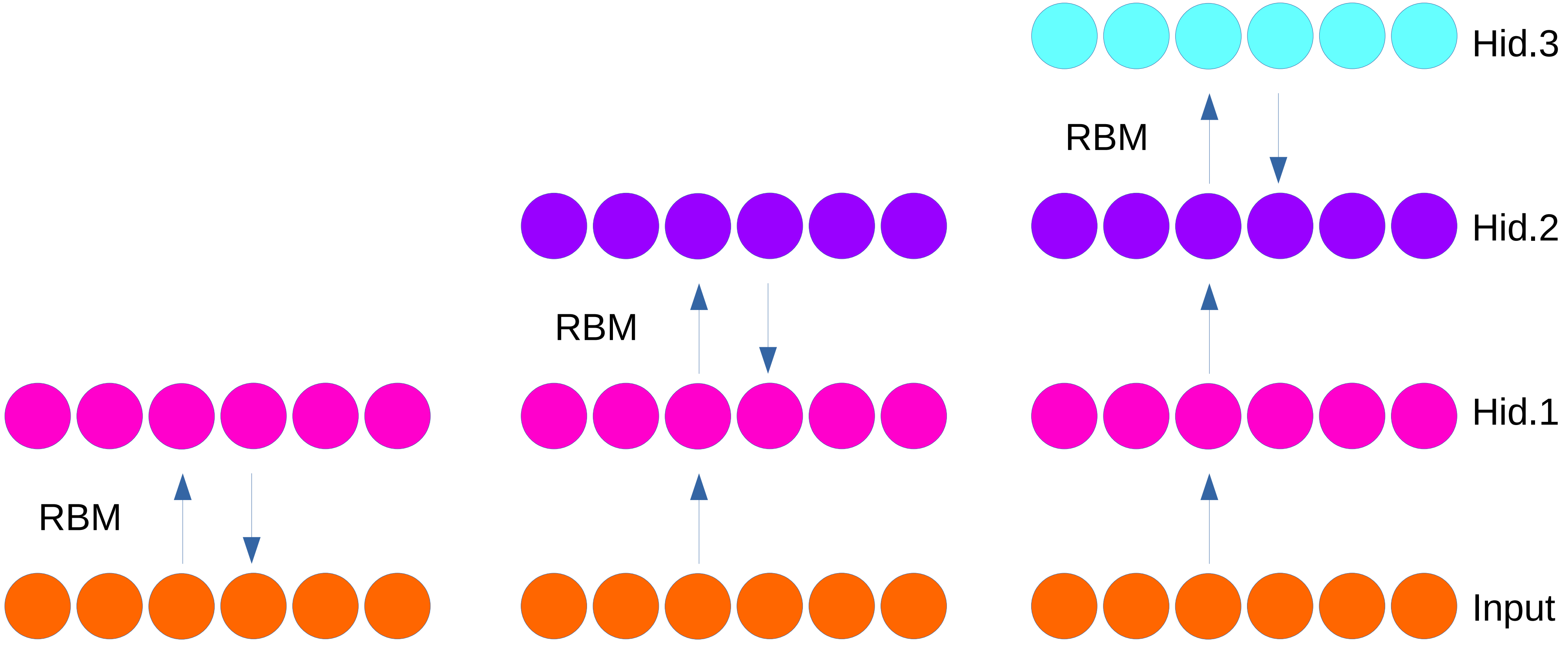
2) Train the second RBM
  - Contrastive divergence algorithm
  - input data from the output layer of the first RBM



Deep Belief Networks

- Training

3) Repeat the procedure for all remaining RBMs

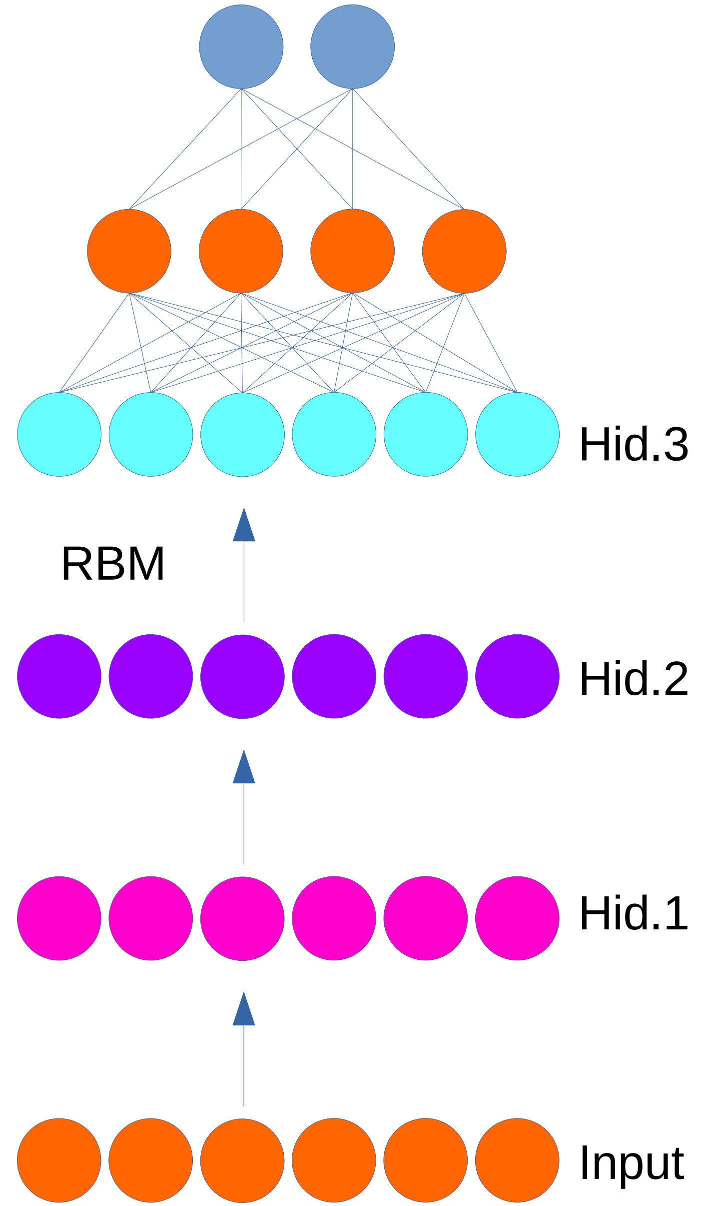


Deep Belief Networks

- Training

4) Add the MLP on the last RBM in the sequence

  - Training using backpropagation algorithm
  - Establishes a link between learned features and desired network outputs



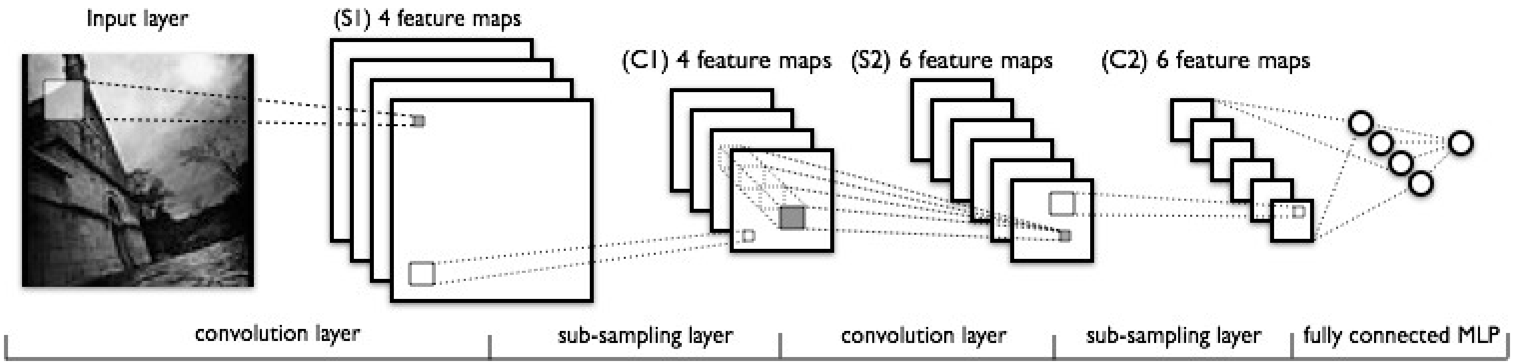
Deep Belief Networks

- Features are learned from feature through the layers

Greedy training

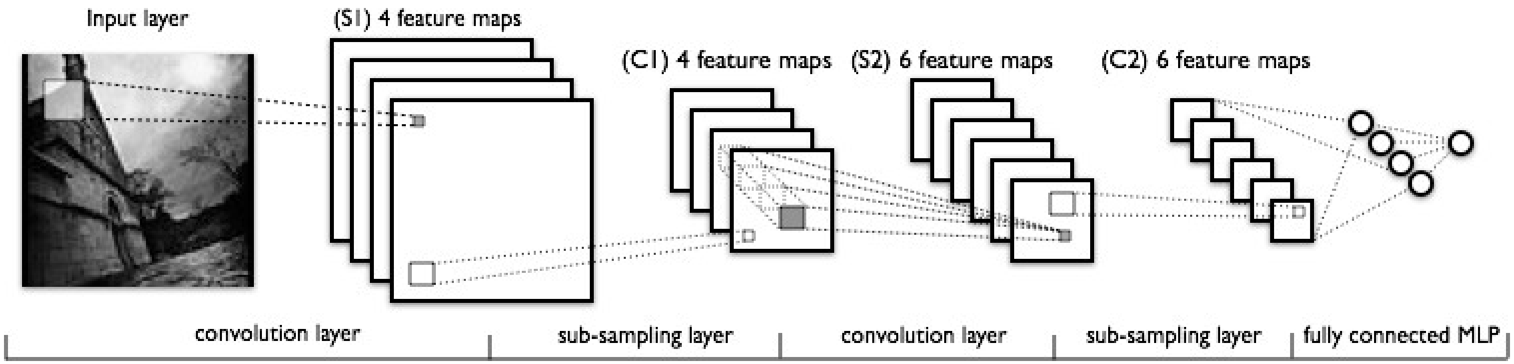
- Layer by layer
- It can be supervised or <u>unsupervised</u>
- Unsupervised -> exploitation of unalbeled data
  - They are more accessible
  - Combining a larger amount of unlabeled initialization data with a smaller amount of labeled data for fine-tuning
- Better for finding the global minimum
  - Thanks to good initialization

Convolutional networks



- A variant of MLP
- It is based on image filtering - convolution
- All weights are trained by a custom backpropagation algorithm

Convolutional networks



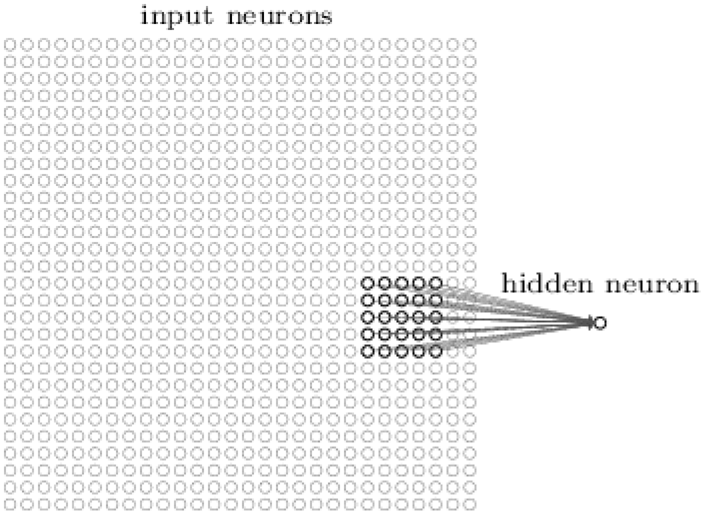
- The network is deterministic
- The end of the network can be a classic MLP (no convolution)
  - Establishes a link between learned feature maps and desired outputs

Convolutional networks

- Suitable for image analysis - object detection
  - The images are stationary
  - A feature that appears at one location in an image is just as likely at other locations
  - Insensitivity to translation in the image
  - It makes sense to aggregate information about such features

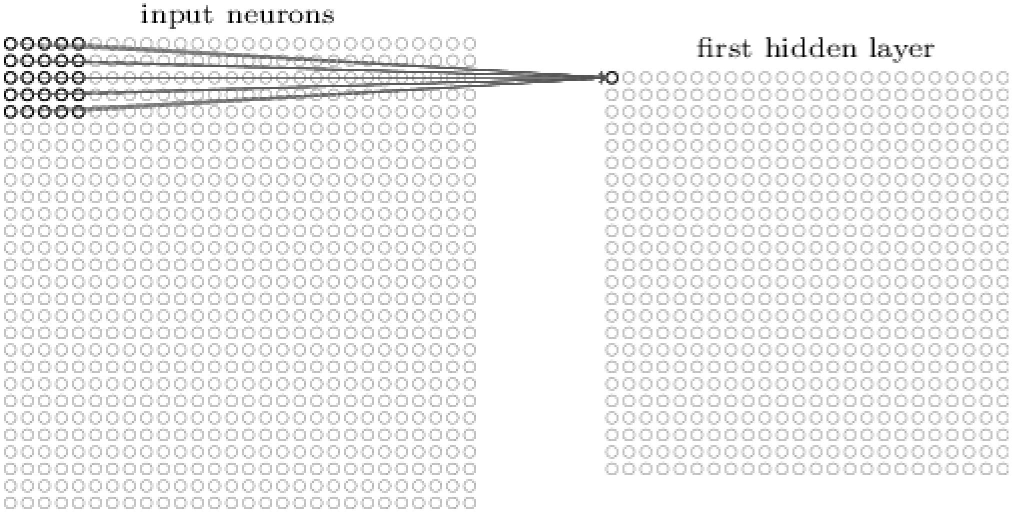
Convolutional networks

- Convolutional layers
  - A number of filters - neurons - are applied
  - Allows the search for different features
    - Each filter is in charge of one "feature"
  - Local receptive field



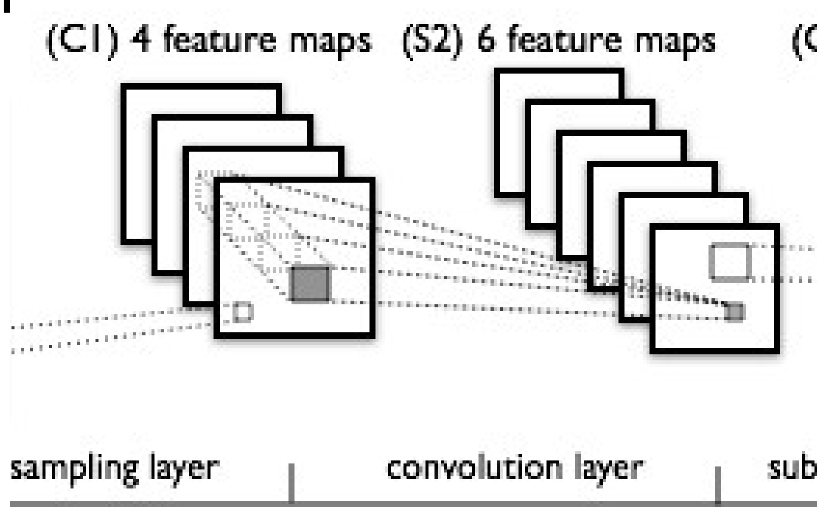
Convolutional networks

- Convolutional layers
  - The filtering result of each filter is called a feature map.



Convolutional networks

- Convolutional layers
  - Each filter is applied to all feature maps of the previous layer but with the same / similar weights
    - Allows to find features no matter which feature map they are i

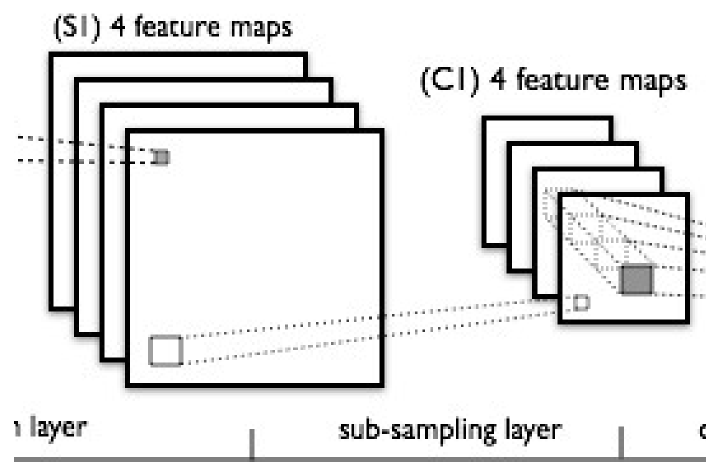


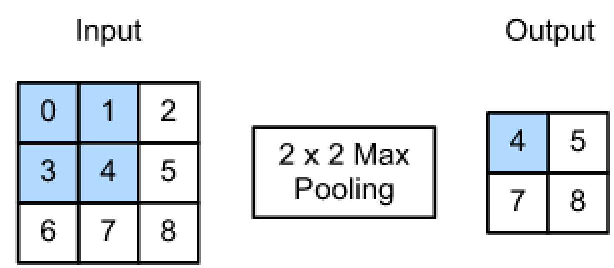
Convolutional networks

- Convolutional layers
  - The position of the feature is not important - the mutual spatial arrangement of the features is important
    - Translation independence
    - Limited robustness to rotation

Convolutional networks

- Subsampling layer
  - Reduces input size
  - Reduces the impact of feature shifts in space and other distortions
  - Different procedures
  - Max pooling
  - Averaging
  - Stochastic sampling
  - Trained neuron





Convolutional networks - backprop

- Other variations and additions are possible
- It is only necessary that the error back propagation works
  - Calculation of partial derivatives of the output with respect to each layer’s input
- In convolutional layers, we can talk about shared weights that are then corrected at once
  - Average correction

$$\Delta w_i^{uk} = \frac{1}{N} \sum_n \Delta w_i^n$$

Convolutional networks - backprop

- Usually training is carried out in mini groups (mini batch)
  - It is done in order to increase parallelism and speed up training
  - The main limitation is the amount of GPU memory

$$\Delta w_i^b = \frac{1}{N} \sum_m \Delta w_i^{uk}(m)$$

Convolutional networks - a problem

- Lots of convolutional layers - deep network
- The problem of the vanishing gradient still exists
- The problem of overfitting still exists

Convolutional networks - tricks

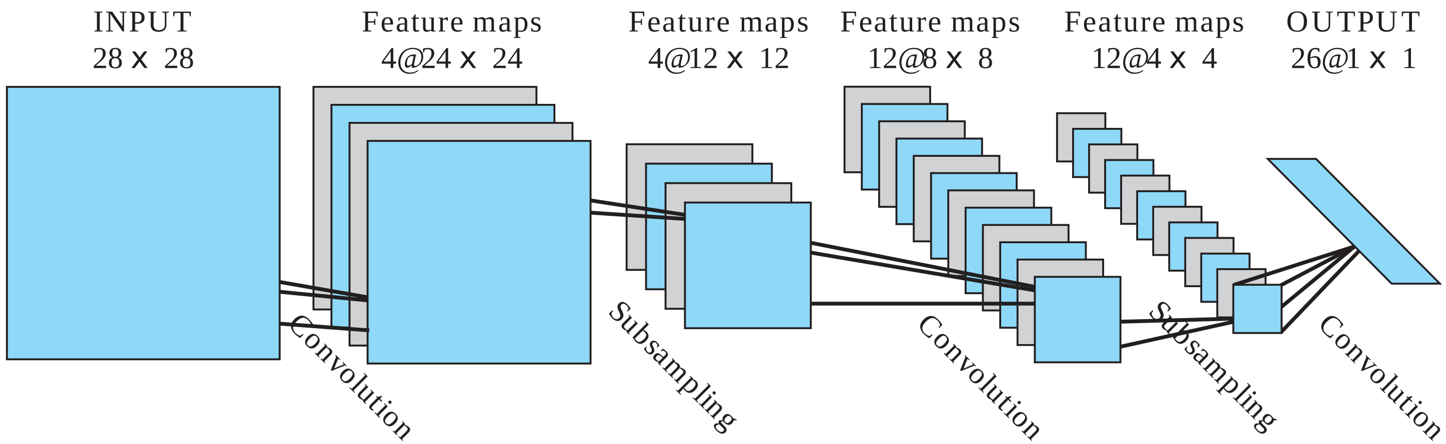
- One solution is to use pre-trained layers
  - Transfer learning
  - There are popular pre-trained convolutional layers
    - Trained on a large amount of images for some image analysis problem - most often image classification
  - They can also train for a specific problem if they wish, but due to the vanishing gradient, the change will be small
    - Fine-tuning

Convolutional networks - tricks

- Another solution is dropout
  - Accidental exclusion of individual neurons
  - Used only for training
  - It makes the network redundant
  - Stochastics are introduced into the deterministic training process

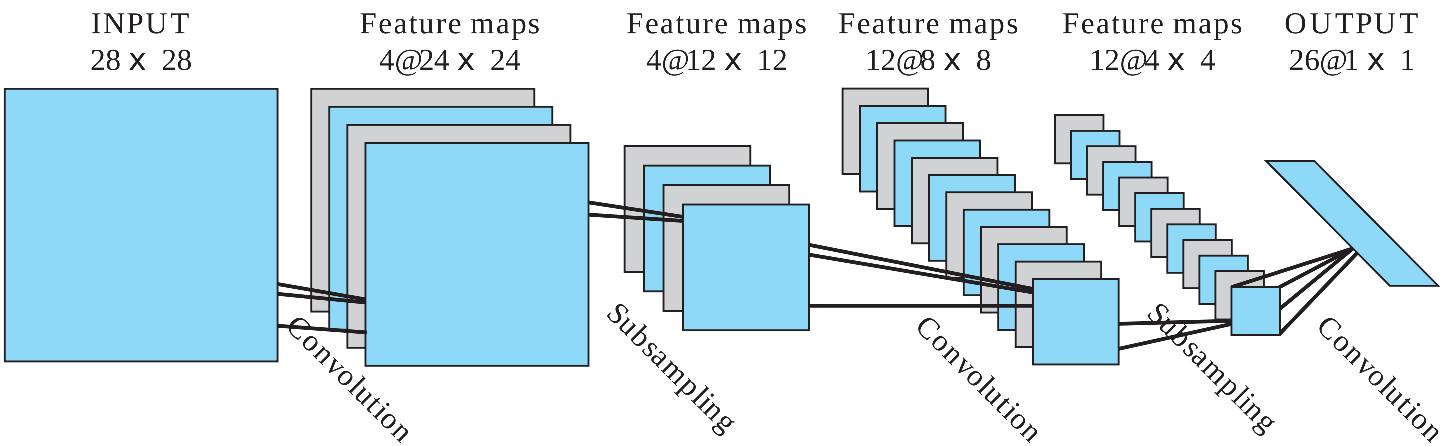
An example of a convolutional network

- Application - character recognition



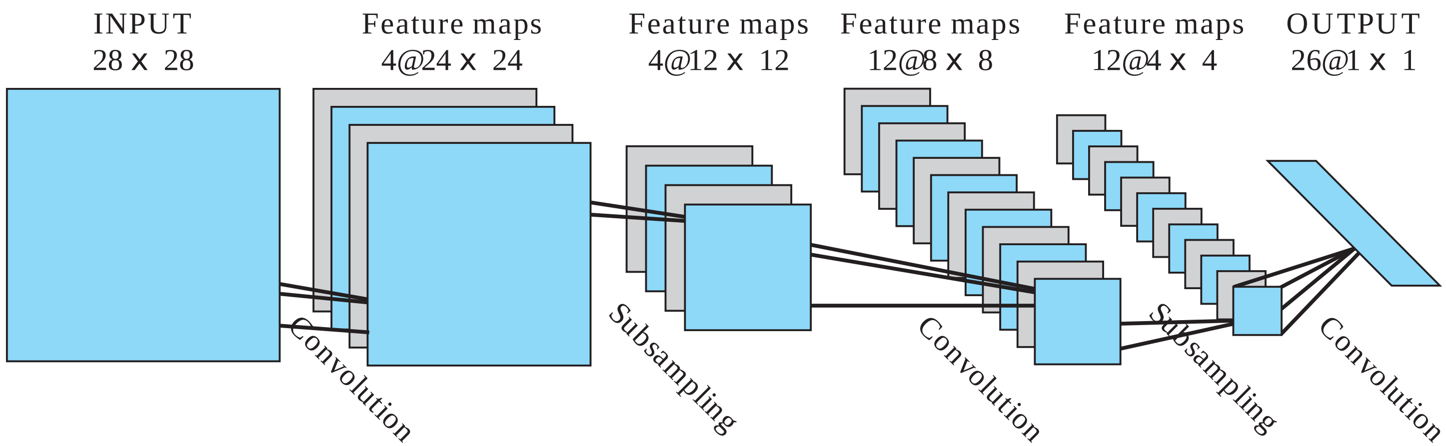
1) the hidden layer performs a convolution
   - The result is four feature maps with 24x24 neurons
   - Each neuron has a 5x5 receptive field
2) the hidden layer performs subsampling and local averaging
   - Feature maps 12x12
   - Neuron receptive fields 2x2
   - Each neuron has weight(s), a shift, and a nonlinear activation function

An example of a convolutional network



3) the hidden layer performs a convolution
   - The result is four feature maps with 8x8 neurons
   - Each neuron is weight-wise connected to all feature maps from the previous layer
4) the hidden layer performs subsampling and local averaging
   - 4x4 feature maps
   - Receptive neuron fields 2x2

An example of a convolutional network



5) output layer
   - has 26 neurons – one for each character
   - Receptive field of each neuron is a 4x4 after all maps features from the previous layer

- Spatial resolution is reduced as the number of feature maps increases
  - Inspired by the visual cortex of cat
- Each element of a feature map is one neuron but all neurons in the feature map share the same set of weights that are trained
  - Reduction of the number of parameters to train
    - Limiting the network capacity
    - Better generalization
  - Parallelization!

Convolutional networks - tricks

- Using only convolutional layers
  - The last feature maps becomes the solution map
  - The final solution is obtained by averaging
  - Average pooling
  - Advantages
    - Greater robustness to shift
    - It does not depend on the size of the image

Convolutional networks

- Significantly fewer parameters compared to MLP
- Number of filters in the convolutional layer
  - Significantly affects the time of calculation and training
  - Fewer filters in layers closer to the input layer
    - Feature maps are larger
  - Recipe: uniform number of calculation operations between layers - the product of the number of pixels/neurons and filters in the layer is constant

Convolutional networks - choices

- Filter size
  - Depending on the data set
- Subsampling
  - Scale
  - Procedure (if not trained)
- Learning rate
  - Reduce them with each epoch?
- Preprocessing
  - PCA?

Convolutional networks

- The end result is an MLP whose size is not excessive
- Local connection of neurons
  - Full network connectivity is avoided
- The design of the network is adapted to the problem being sought

Summary

- Deep networks - Multiple layers with nonlinear neurons (eg larger than 10)
- Learning with or without supervision
- Hierarchical structure: from simpler to more complex features
- More layers - higher level of abstraction
- In some tasks they reached the accuracy of human experts
- Frequent criticism - used as a black box