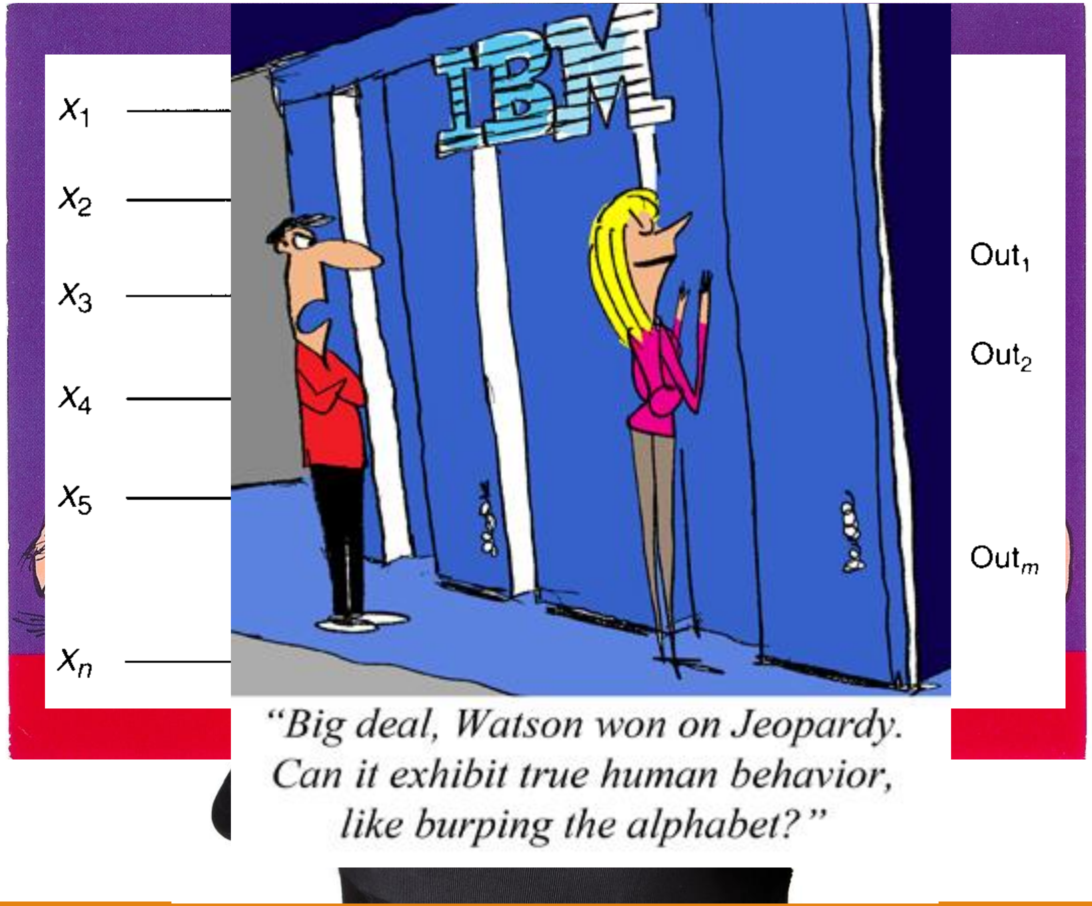
NFBI Summer School

AN INTRODUCTION TO DEEP LEARNING

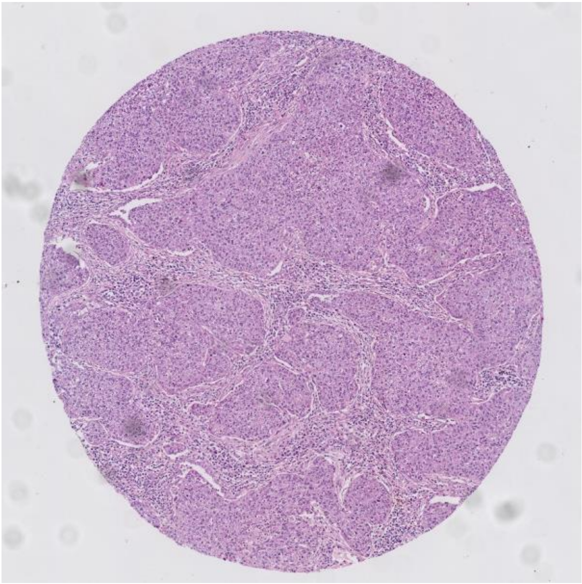
What is ‘deep learning’?

- Multi-layer neural networks?
- Trying to model the human brain?
- Something a supercomputer does?
- Magic?
- All of the above?

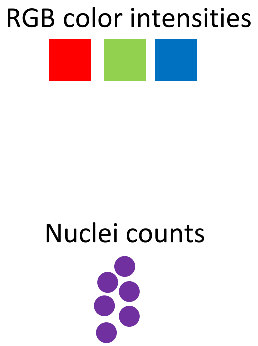


‘Traditional’ machine learning

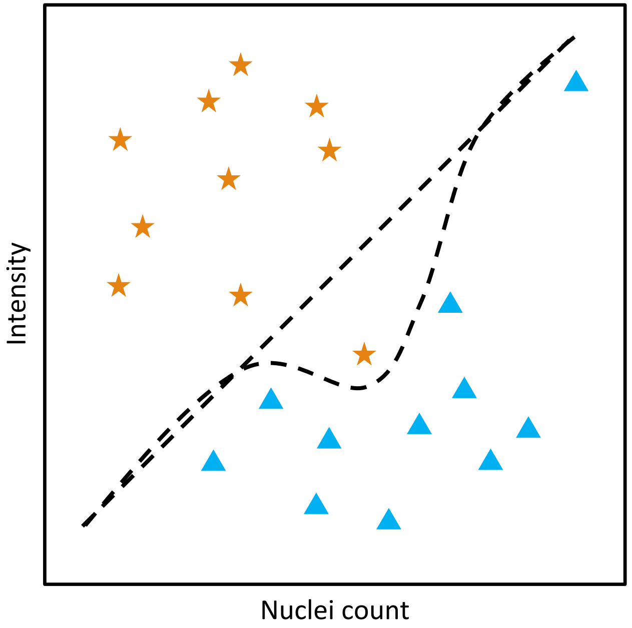
Input data



Feature design

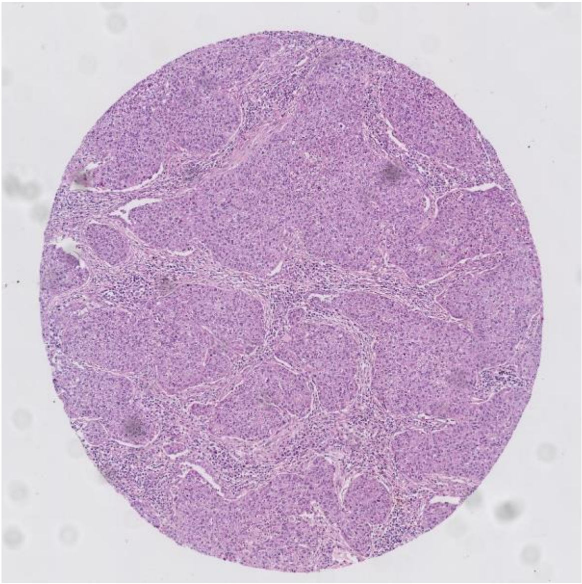


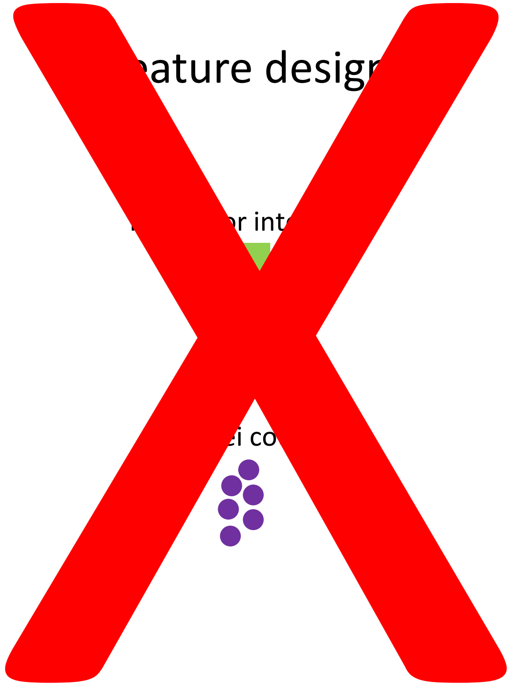
Classification



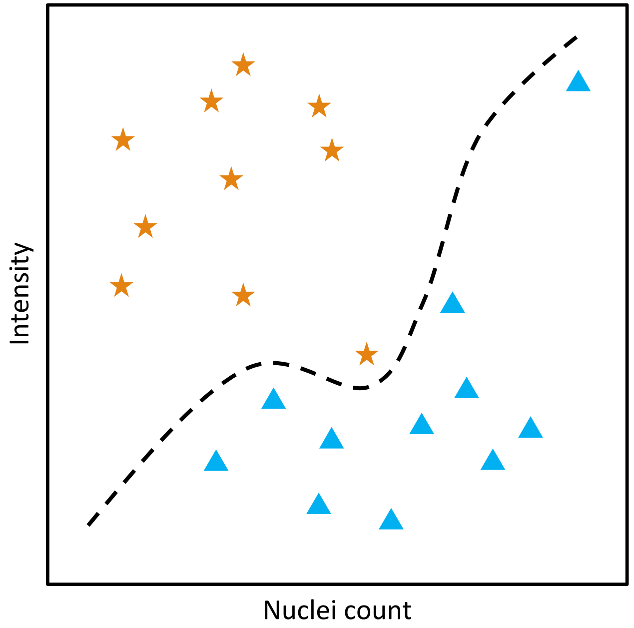
Representation learning

Input data





Classification



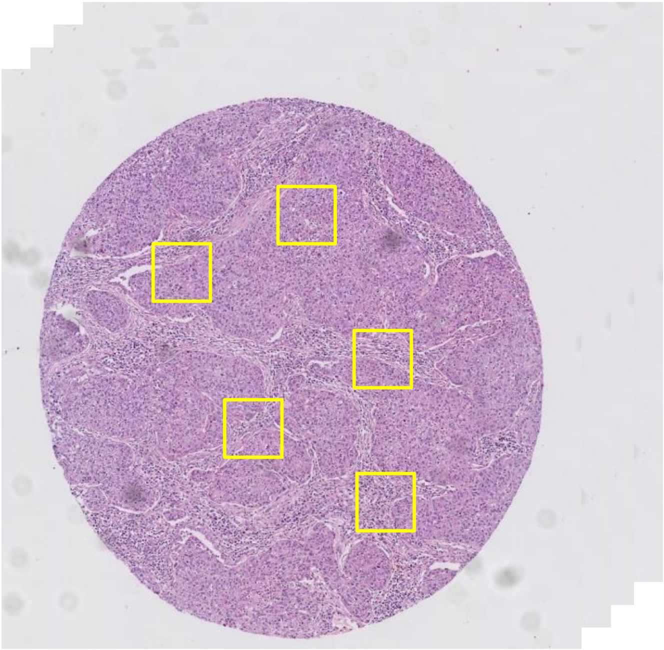
Representation learning

Several techniques

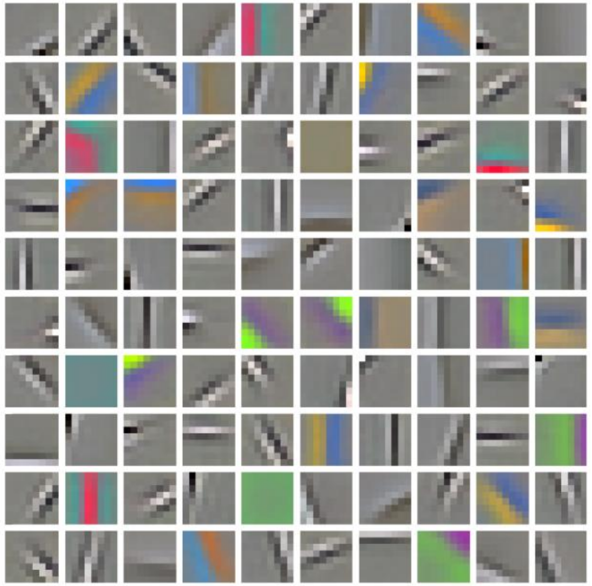
- K-means-based dictionary learning
- Sparse coding
- Auto-encoder neural networks

Dictionary learning: K-means

- Extract patches of size W from a set of images
- Apply K-mean clustering to find K clusters



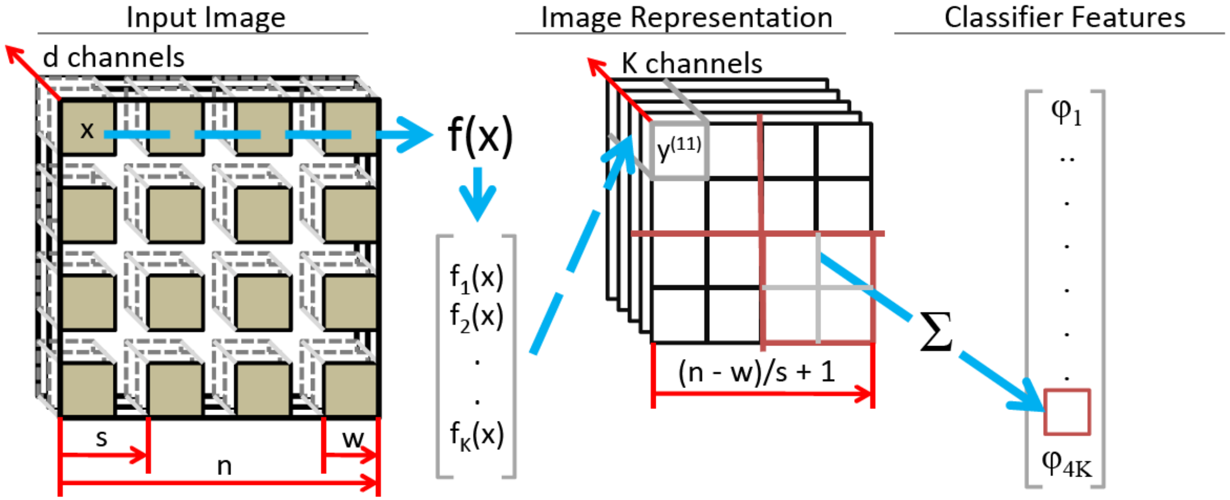
K-means clustering



Dictionary learning: K-means

To classify an image

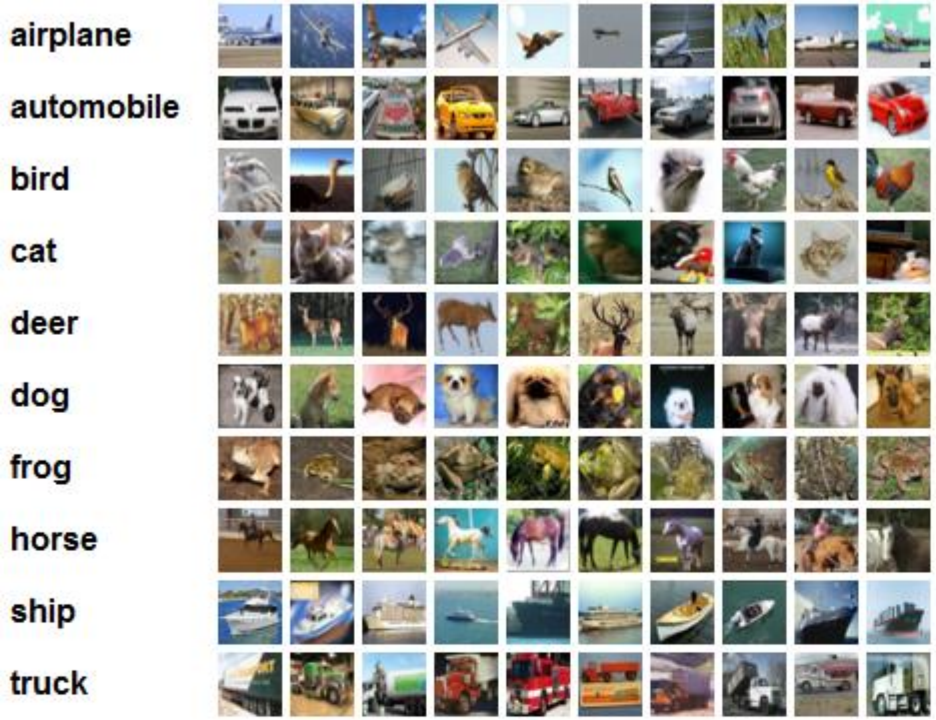
- W-sized patches are extracted with a stride S
- A feature vector is defined for each patch by assessing to which cluster(s) it belongs
- A feature vector for the image is obtained by combining patch feature vectors
- A classifier can then be trained and applied to new images



Dictionary learning: K-means

This relatively simple approach was state of the art on the CIFAR-10 dataset for quite some time

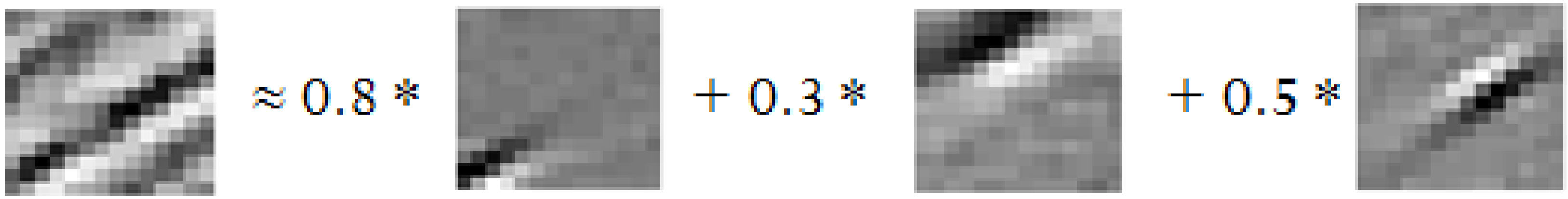
- K-means (4000 features, 'soft' feature vectors) – 80% accuracy
- Deeply supervised convolutional networks – 92% accuracy
- Human level performance – 94% accuracy



Dictionary learning: Sparse coding

General idea: an image is a linear combination of a sparse set of ‘basis’ images (a.k.a. features)

$$x = \sum_{i=0}^{k} a_i \varphi_i$$

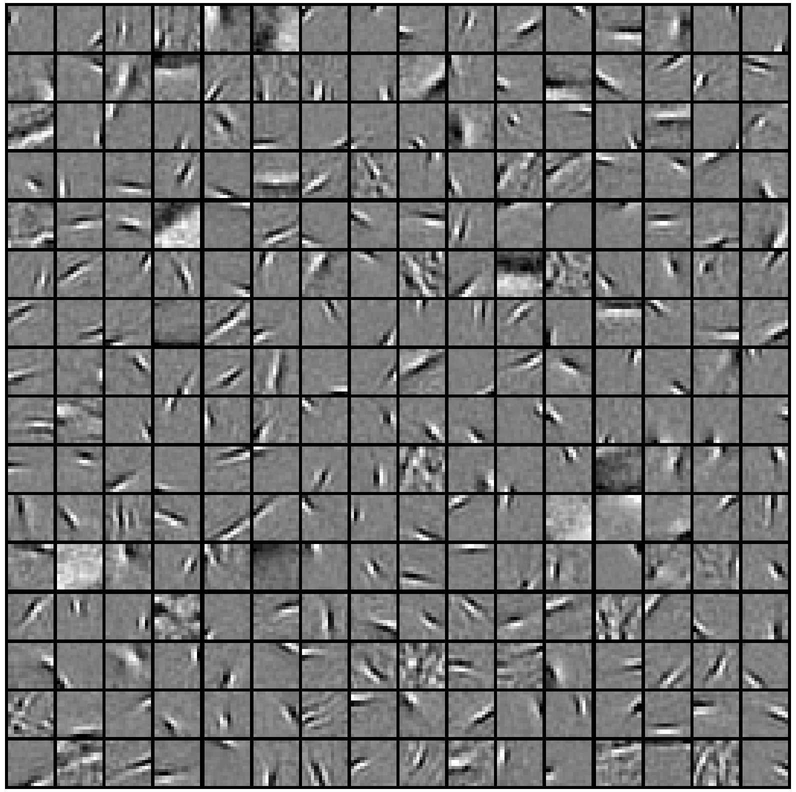


Dictionary learning: Sparse coding

To obtain the sparse code book, one iteratively optimizes:

$$\text{minimize}_{a_i^{(j)},\phi_i} \sum_{j=1}^{m} \left\| \mathbf{x}^{(j)} - \sum_{i=1}^{k} a_i^{(j)} \phi_i \right\|^2 + \lambda \sum_{i=1}^{k} S(a_i^{(j)})$$

Here S is the sparsity penalty, often the L1-norm



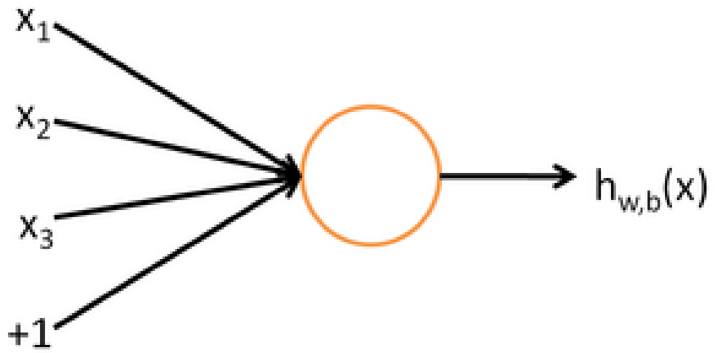
One disadvantage of sparse coding is that to obtain the coefficient alpha for a new image the following optimization has to be performed, making it computationally intensive even at test time:

$$\min_{a} \sum_{i=1}^{m} \left\| x_i - \sum_{j=1}^{k} a_{i,j} \phi_j \right\|^2 + \lambda \sum_{i=1}^{m} \sum_{j=1}^{k} |a_{i,j}|$$

(Auto-encoder) neural networks

A neural network is a network of neurons (duh)

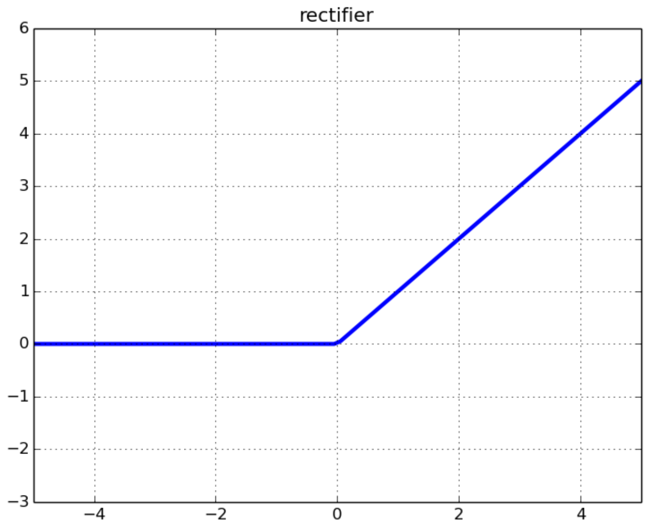
- A single neuron consists of a weight vector W, a bias b, and a non-linearity f
- A neuron is mathematically defined as: $h_{W,b}(x) = f(W^T x + b)$ with x the input vector
- And schematically as:



(Auto-encoder) neural networks

Several choices for the non-linearity are possible

- Threshold
- Sigmoid
- Hyperbolic tangent
- Rectifier

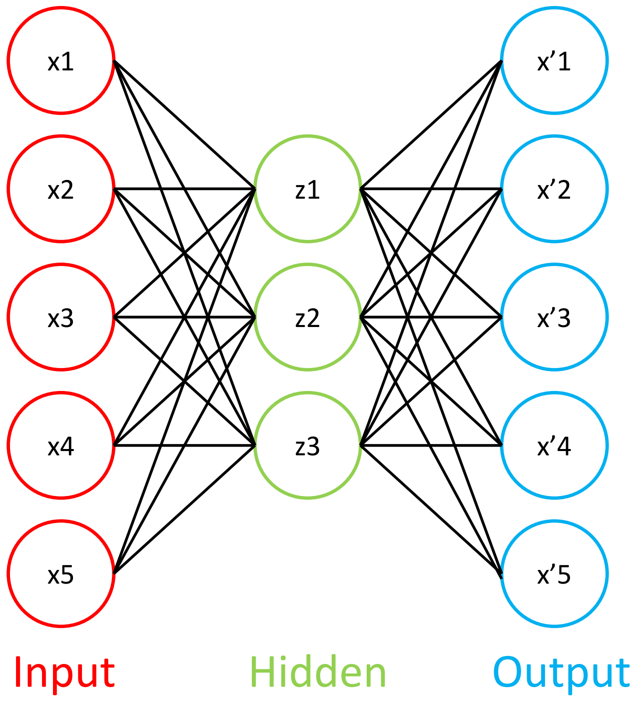


A single node with a sigmoid linearity can be used to represent logistic regression (perceptron)

(Auto-encoder) neural networks

When nodes are stacked they form networks

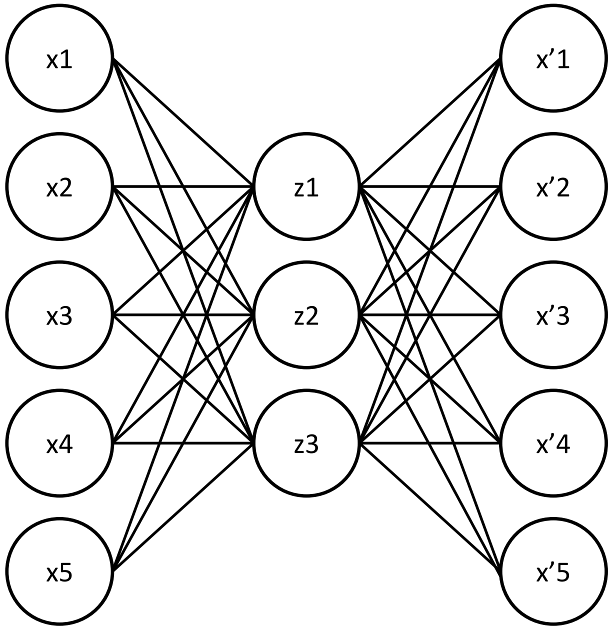
- Feed-forward neural network
- Multi-layered perceptron



Auto-encoder neural networks

A network which reconstructs the input X through a compacter representation Z

- Z can be considered the ‘feature vector’ of X
- After learning the features, one could replace the decoder (from Z to X’) with a classifier



Auto-encoder neural network

The weight-vectors for the nodes in the auto-encoder network can be learned by minimizing the cost across a large number of example images. For an auto-encoder, the cost is defined as:

$$C(X) = \|X' - X\|^2$$

Where X' is the model reconstruction of X. The weights can be updated using gradient descent:

$\frac{dC}{dW_{ij}} = \frac{\|X'-X\|^2}{dW_{ij}}$ and subsequently $W_{ij} = W_{ij} - \alpha \frac{dC}{dW_{ij}}$

for each edge weight W connecting node i from layer L – 1 to node j from layer L. α is the learning rate

As each node is only dependent on its direct inputs, the derivatives can be efficiently computed using the chain rule and backpropagation.

Auto-encoder neural network

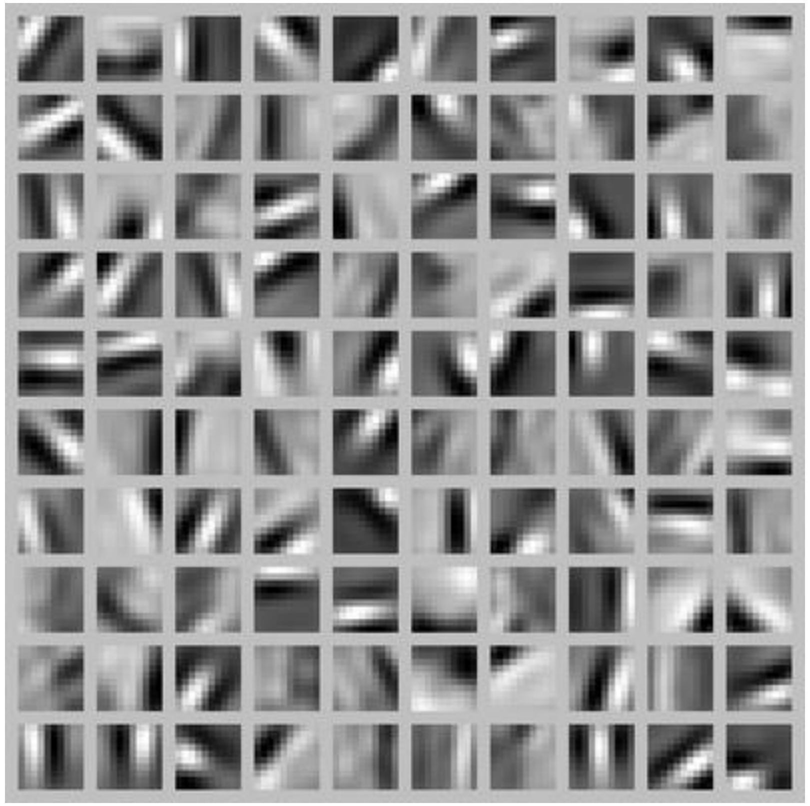
The hidden representation Z of the auto-encoder can be interpreted as the feature space. The features learned by the system can also be visualized.

- Assume the input is norm-constrained to 1
- The node *i* in Z is then maximally activated when X is

$$x_j = \frac{W_{ij}^{(1)}}{\sqrt{\sum_{j=1}^{100}(W_{ij}^{(1)})^2}}.$$

  for all pixels *j* in X

- The obtained X is then the visual representation of the feature node *i* is looking for



Multi-layer representation (‘deep’) learning

Previous examples have a single layer of abstraction

These can already represent any function (just add enough nodes)

Multi-layer (‘deep’) architecture can model the same function with exponentially less nodes

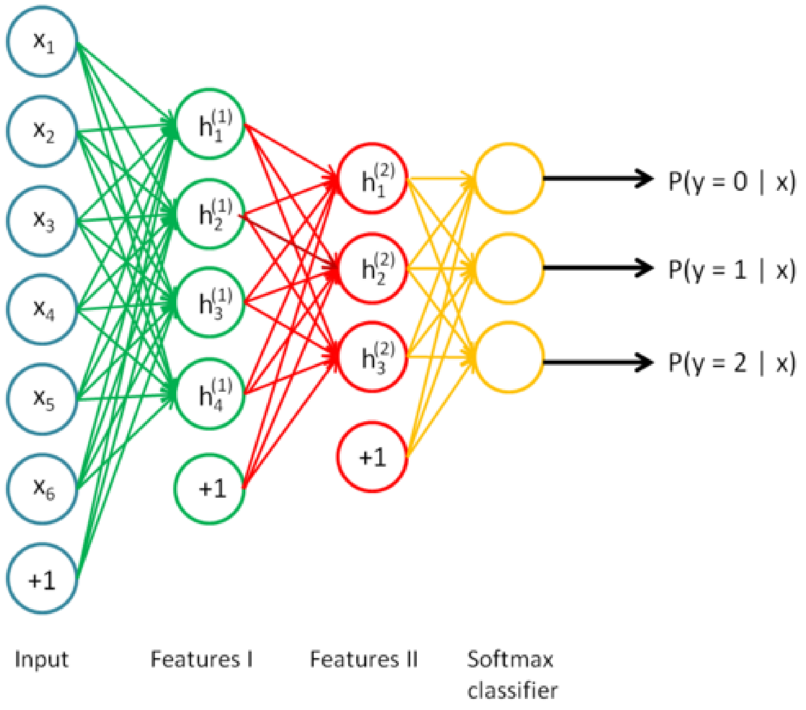
Multi-layer representation ('deep') learning

Multi-layer ('deep') auto-encoders/stacked auto-encoders

Strangely, training multi-layer networks did not improve results

- Early layers do not learn compared to later layers (vanishing gradient problem)

Declined interest related to multi-layer representation learning



Multi-layer representation (‘deep’) learning

In 2006 three landmark papers changed this:

- A fast learning algorithm for deep belief nets (Hinton et al.)
- Greedy Layer-Wise Training of Deep Networks (Bengio et al.)
- Efficient Learning of Sparse Representations with an Energy-Based Model (LeCun et al.)

Multi-layer representation (‘deep’) learning

Worked around the vanishing gradient problem

- Each layer is pre-trained using unsupervised pre-training
- After pre-training, layers are connected and supervised fine-tuning is performed

Another important factor in re-igniting ‘deep learning’ research interest was the release of CUDA (NVIDIA) and OpenCL (Kronos) in 2007 and 2009

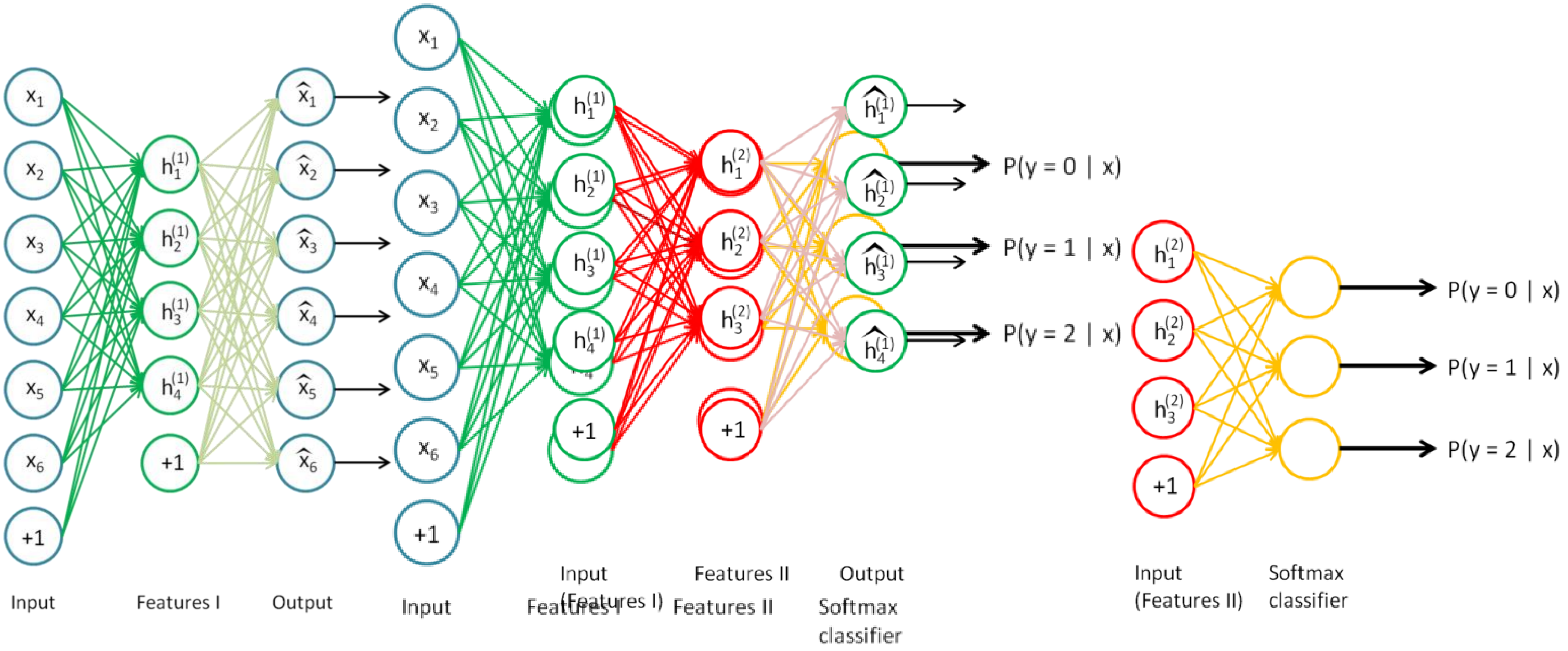
- Allowed efficient computation of deep architectures

Differing model types

A large diversity in neural network types has been developed

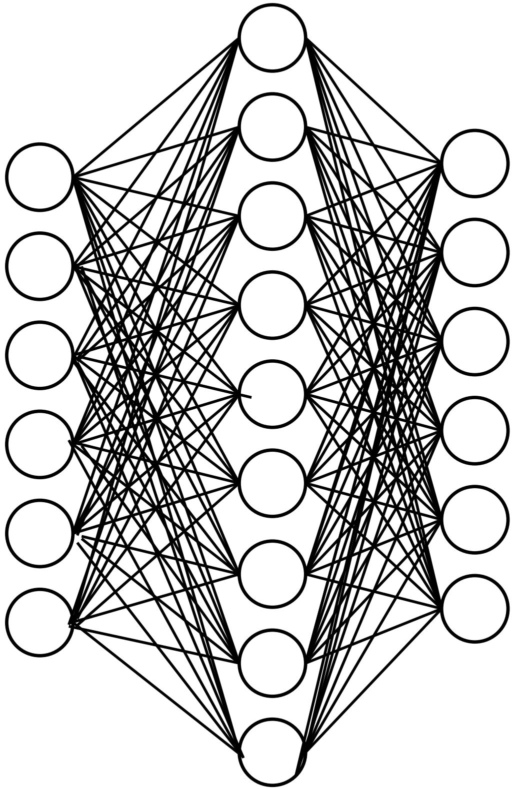
- Stacked auto-encoders
- Restricted Boltzmann machines
- Recurrent neural networks
- Convolutional neural networks

Stacked auto-encoders

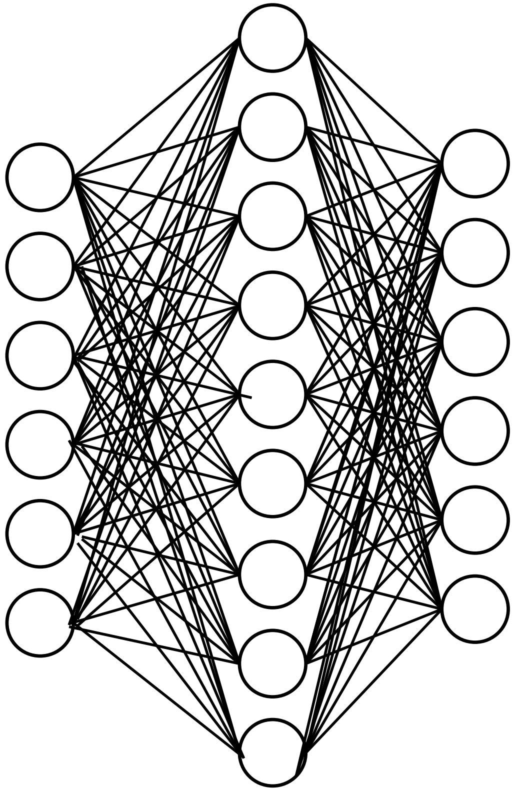


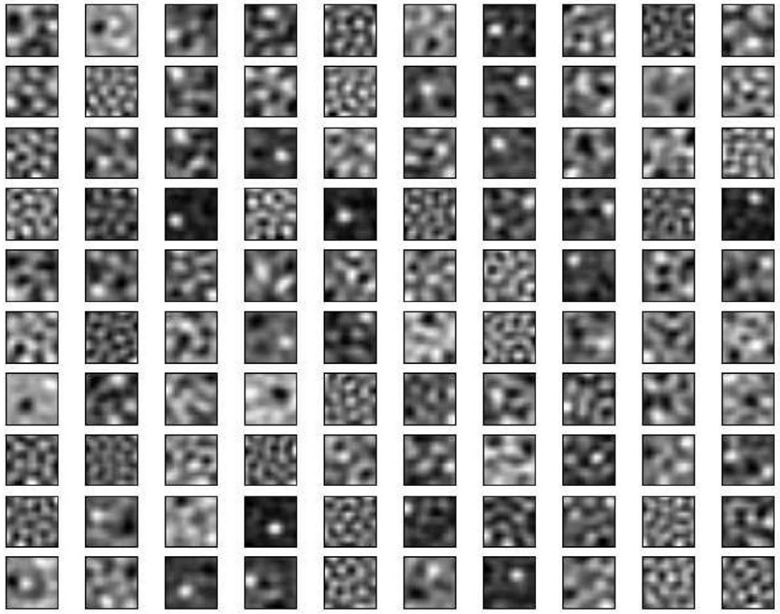
Stacked denoising auto-encoders

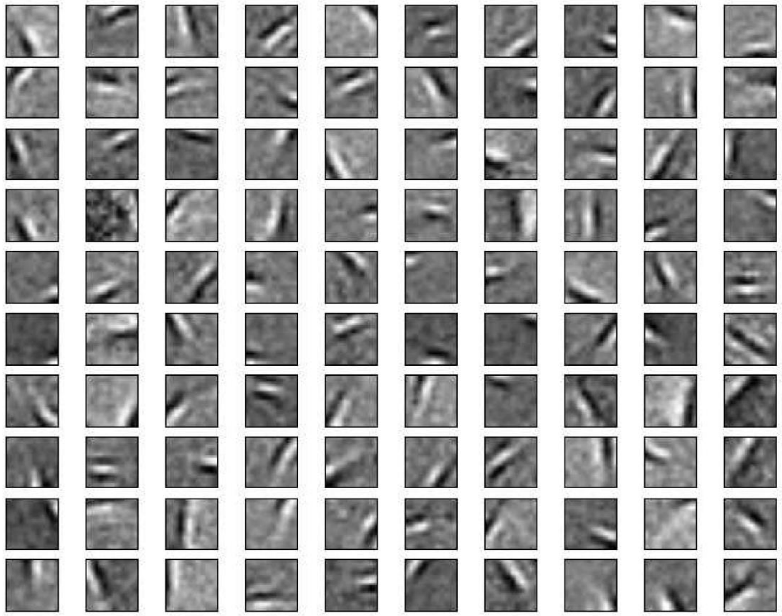
What to do when we need more features than input pixels?



Stacked denoising auto-encoders



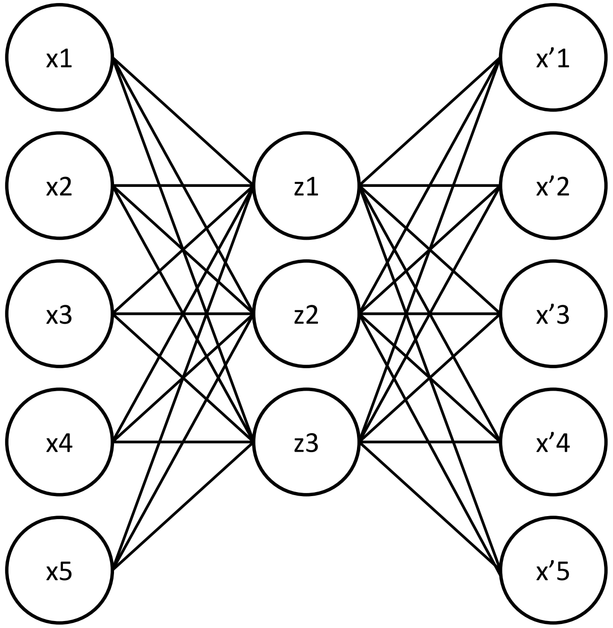




Variational auto-encoders

In auto-encoders, there are no guarantees on reasonable latent variables Z

- Kingma et al. added a Bayesian framework to auto-encoders resulting in the so-called variational auto-encoders

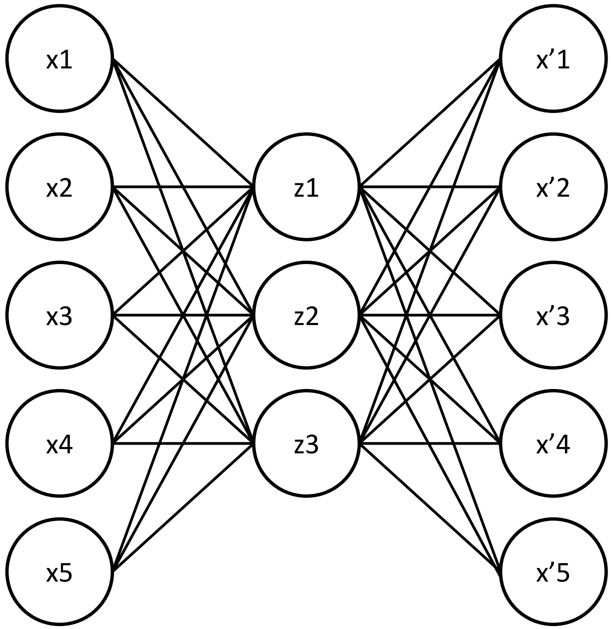


Variational auto-encoders

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

$$L = \|x - x'\|$$

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \boxed{\frac{1}{2}\sum_{j=1}^{J}\left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right)} + \boxed{\frac{1}{L}\sum_{l=1}^{L}\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})}$$



[Demo (faces)]

Restricted Boltzmann machines

Another category of probabilistic models are the so-called Boltzmann machines

- Energy-based model
- Stochastic neural network
- Bi-directional (from visible to hidden AND hidden to visible)
- Restricted refers to the fact that no interactions between nodes in the same layer is allowed

$$P(x) = \sum_h P(x,h) = \sum_h \frac{e^{-E(x,h)}}{Z}.$$

$$Z = \sum_x e^{-E(x)}$$

$$\mathcal{F}(x) = -\log \sum_h e^{-E(x,h)}$$

$$P(x) = \frac{e^{-\mathcal{F}(x)}}{Z} \text{ with } Z = \sum_x e^{-\mathcal{F}(x)}.$$

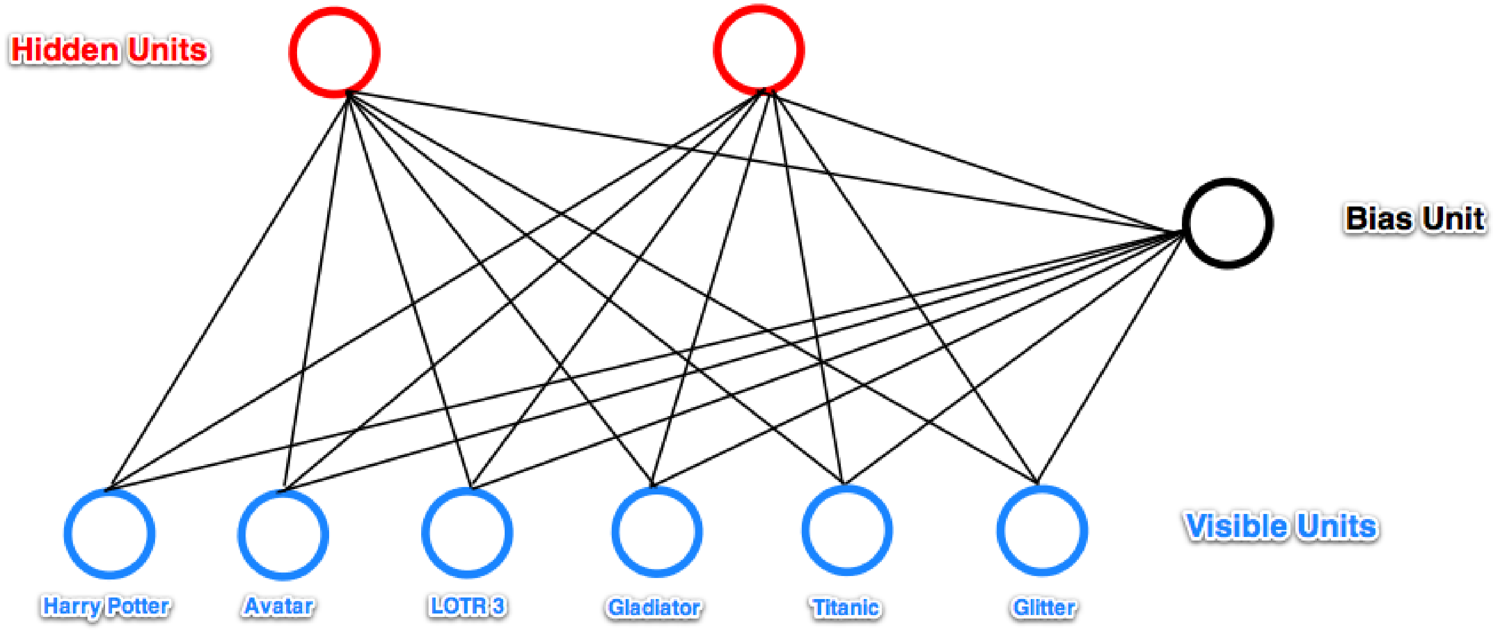
$$\mathcal{L}(\theta, \mathcal{D}) = \frac{1}{N} \sum_{x^{(i)} \in \mathcal{D}} \log\ p(x^{(i)})$$

$$\ell(\theta, \mathcal{D}) = -\mathcal{L}(\theta, \mathcal{D})$$

$$-\frac{\partial \log p(x)}{\partial \theta} = \frac{\partial \mathcal{F}(x)}{\partial \theta} - \boxed{\sum_{\tilde{x}} p(\tilde{x}) \ \frac{\partial \mathcal{F}(\tilde{x})}{\partial \theta}}.$$

Restricted Boltzmann machines

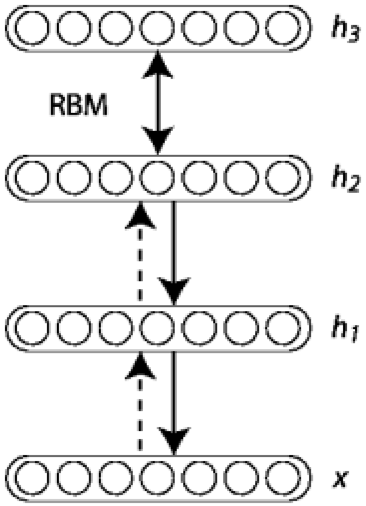
Can be approximated using contrastive divergence



[Music generation demo]

Deep belief networks

Stacked restricted Boltzmann machines



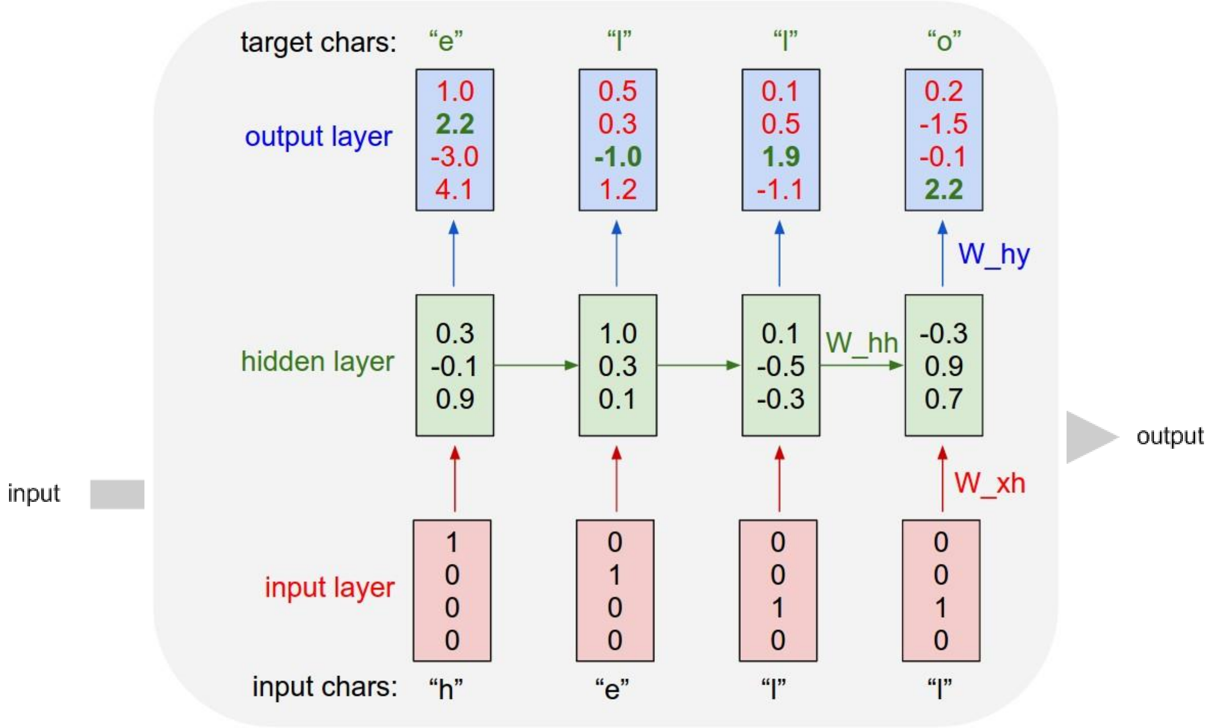
Recurrent neural networks

All networks discussed up till now take a static input and produce a static output

- How to apply networks to variable length data?
- How to obtain variable length output?

Recurrent neural networks offer a solution

Recurrent neural networks



For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ over U compatible with the complex

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S. Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) *f is locally of finite type. Since $S = \mathrm{Spec}(R)$ and $Y = \mathrm{Spec}(R)$.*

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\ldots,0}$.

Lemma 0.2. *Let X be a locally Noetherian scheme over S, $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \overline{A}_2$ works.*

Lemma 0.3. *In Situation ??. Hence we may assume $\mathfrak{q}' = 0$.*

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

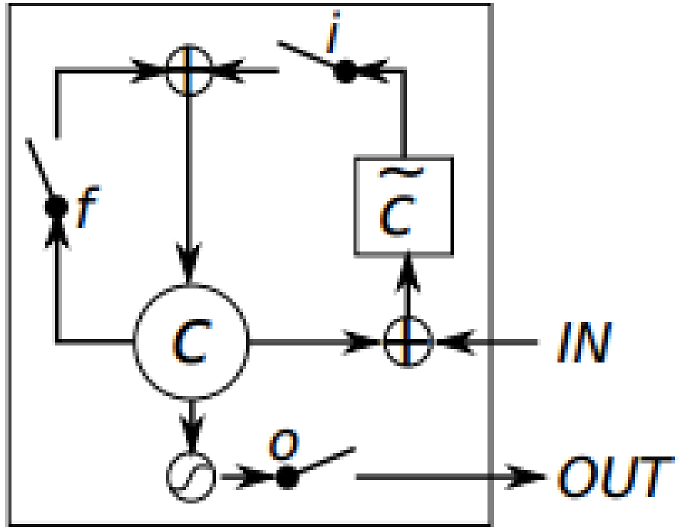
where K is an F-algebra where δ_{n+1} is a scheme over S. □

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Long short term memory

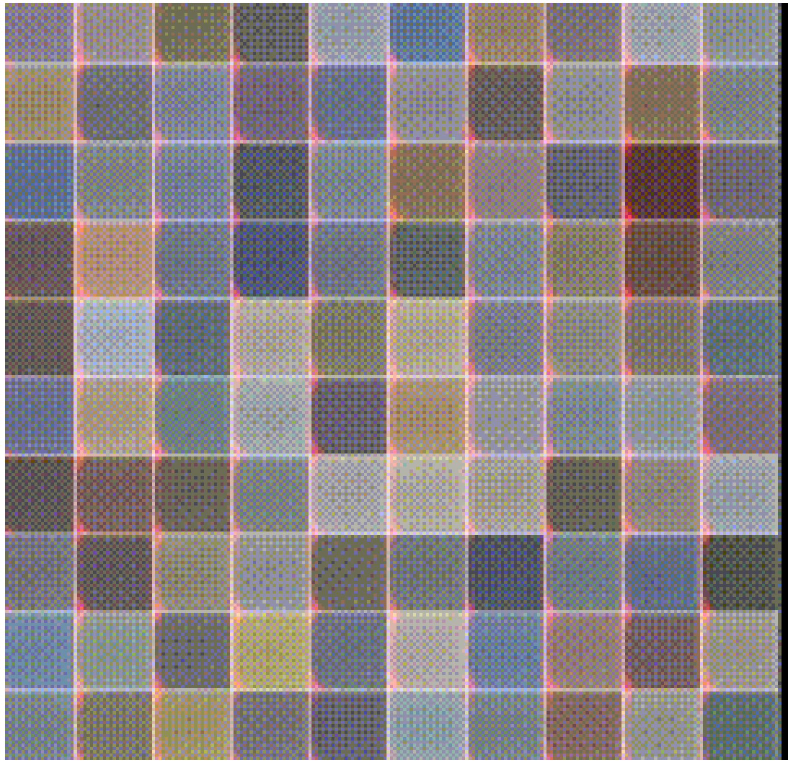
One issues with regular RNNs, how long should you allow sequences to become?

- Longer sequences reduce the importance of new time steps t
- Not every time step carries important information

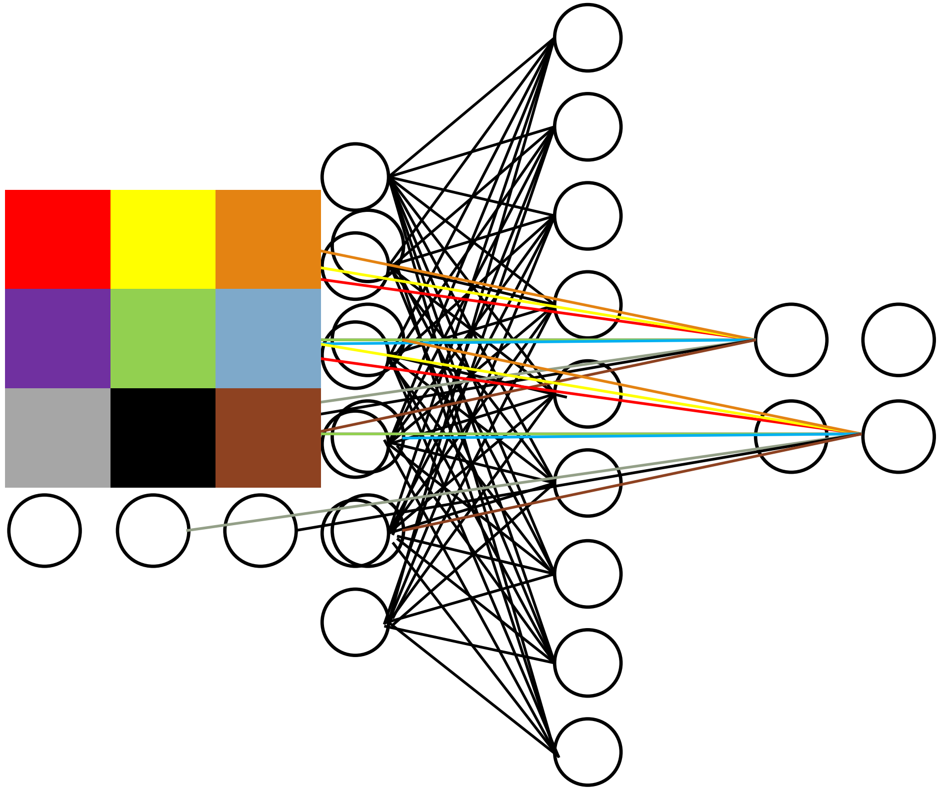


Recurrent neural networks

If you combine variational auto-encoders with recurrent LSTM networks, you can even have networks ‘draw’ images

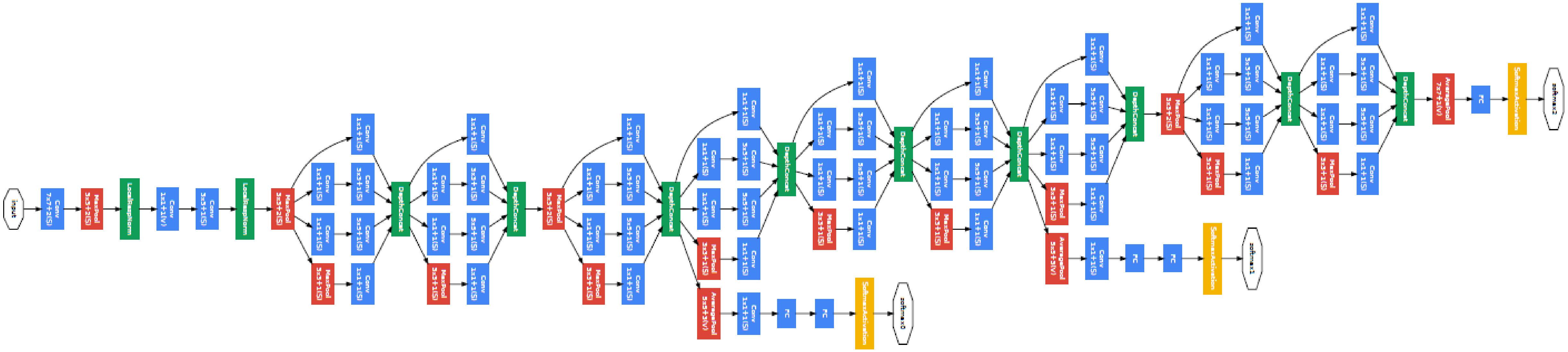


Convolutional neural networks



Convolutional neural networks

Convolutional layers can be stacked, together with other operations like pooling (downsampling) to form complex network architectures



Convolutional neural networks

Convolutional neural networks enforce sparse connectivity and weight sharing

- Not every node in layer L is connected to every node in layer L - 1
- Weights are structured as filters, replicated across every input position

Convolutional networks maintain positional information from one layer to the next

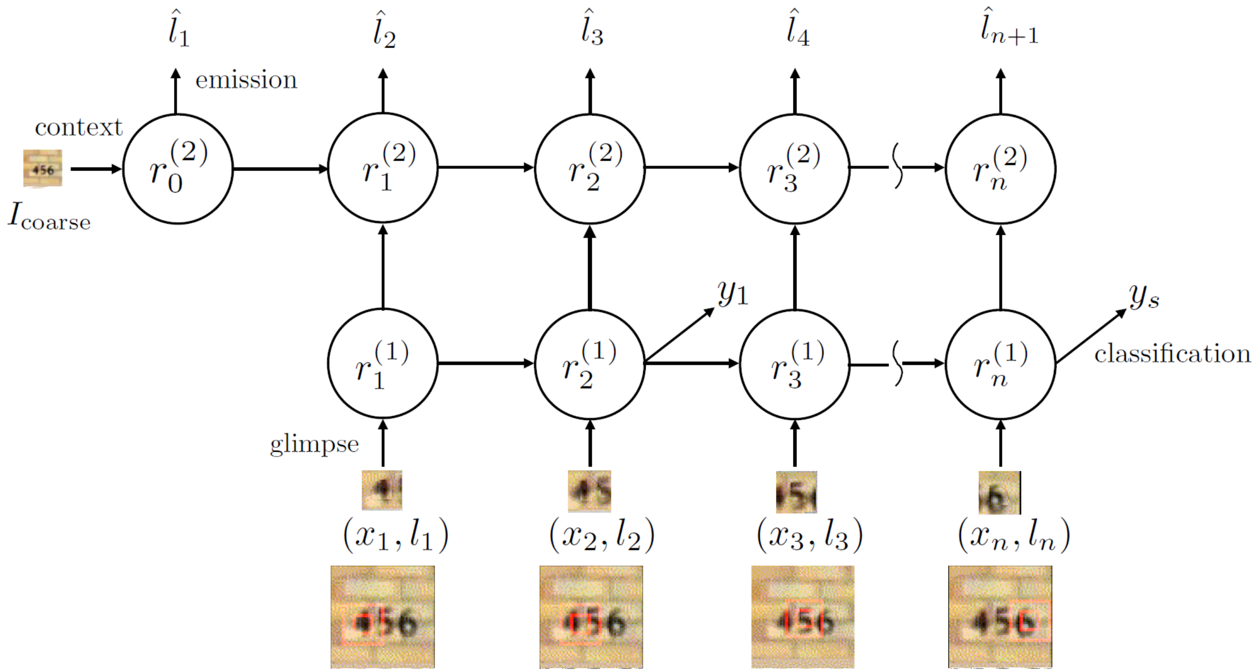
- Often highly important in image analysis as images have a lot of positional information

Convolutional networks are currently record holders in most generic computer vision challenges

- MNIST handwritten digit classification
- ImageNet natural image classification (1000 classes)
- Labelled Faces in the Wild

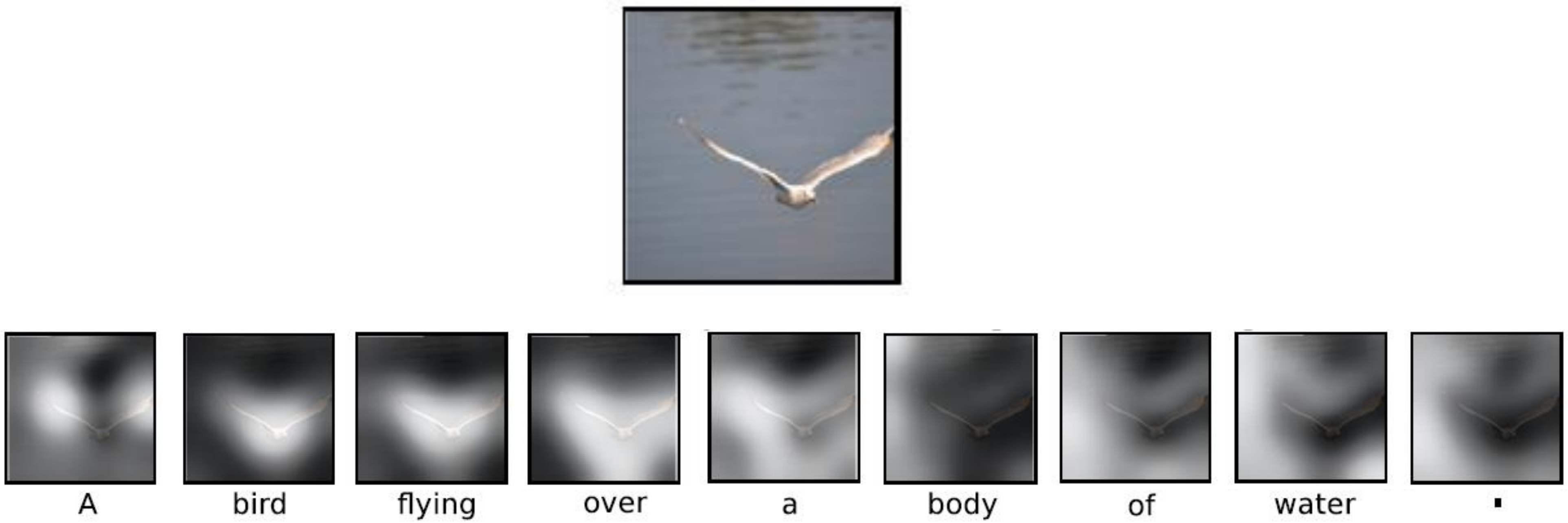
State-of-the-art: ‘visual attention’-networks

Combination of recurrent and convolutional networks



State-of-the-art: ‘visual attention’-networks

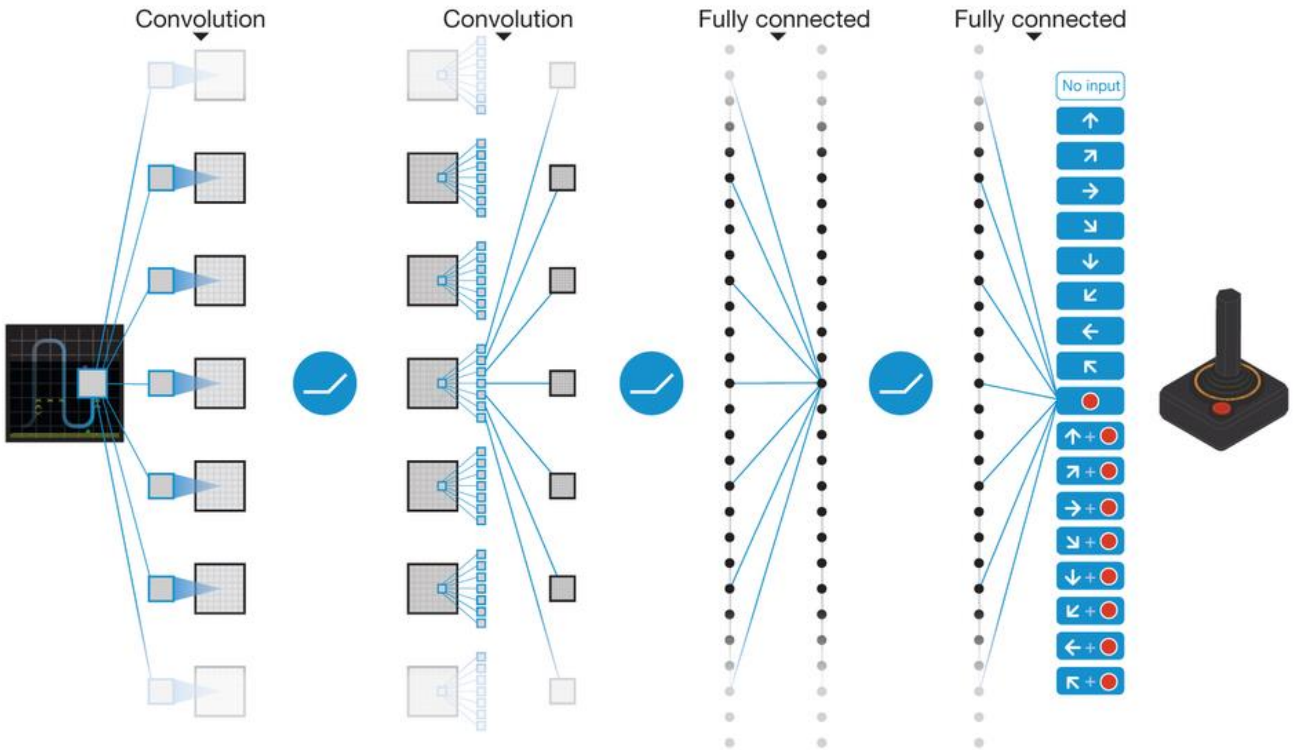
- Can be used to caption images



State-of-the-art: reinforcement learning

- Playing video games

Algorithm 1: deep Q-learning with experience replay.
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \mathrm{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^{-}\right) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

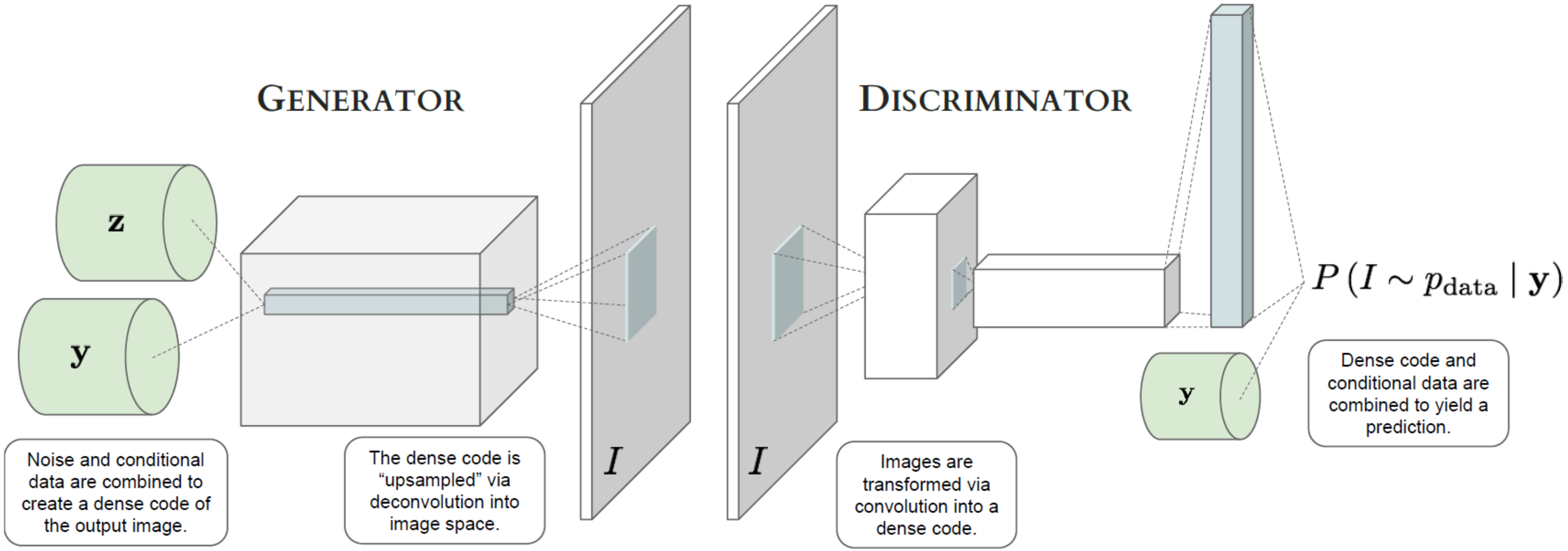


State-of-the-art: reinforcement learning

- Playing video games

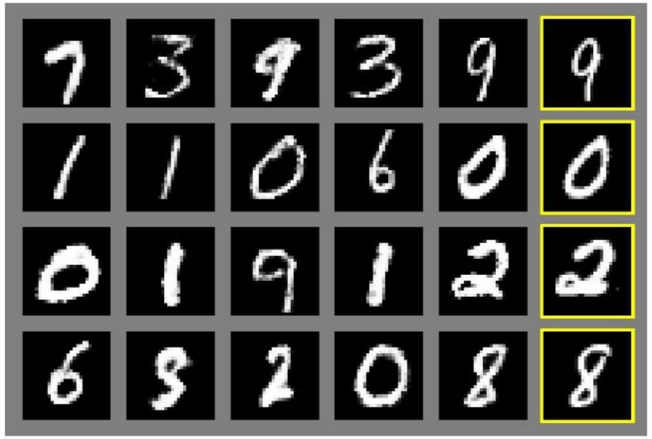
State-of-the-art: adversarial generative models

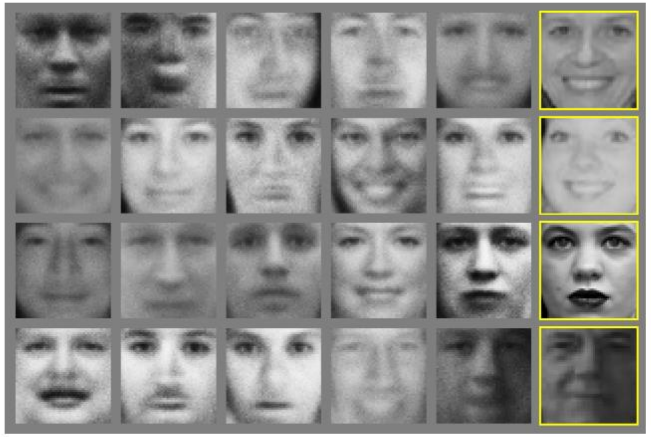
- Generative models and adversarial images

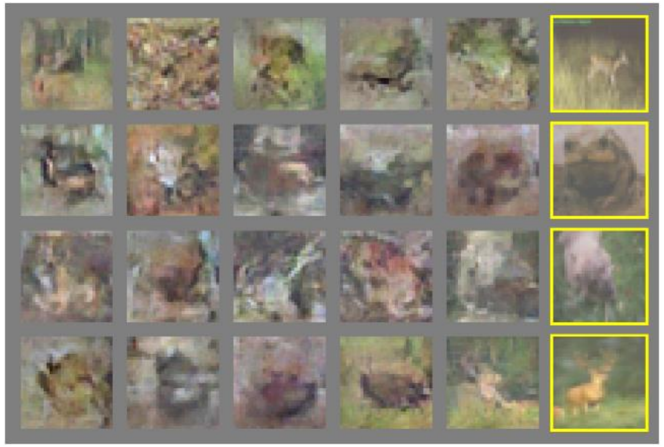


State-of-the-art: adversarial generative models

- Generative models and adversarial images







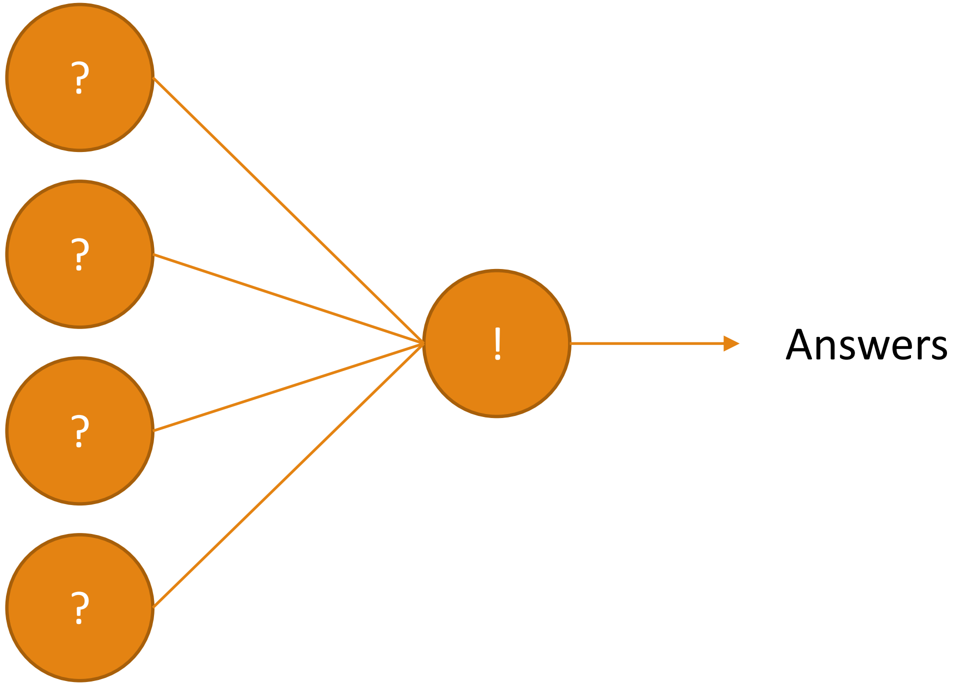
Future directions

The future is unsupervised

- Most human learning is unsupervised
- Way more unsupervised than supervised data available

Machine reasoning

- E.g. moving from classification/recognition to understanding


?
?
?
?
!
Answers