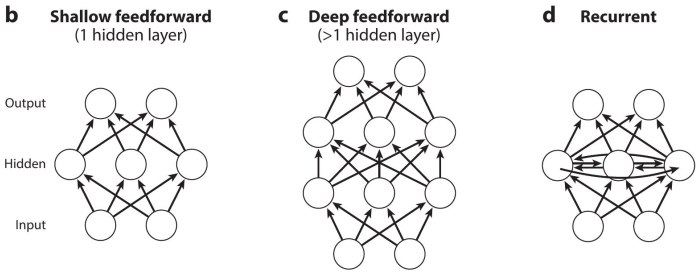
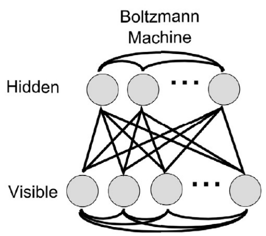
Shallow vs. deep networks



- **Shallow**: one hidden layer
  - Features can be learned more-or-less independently
  - Arbitrary function approximator (with enough hidden units)
- **Deep**: two or more hidden layers
  - Upper hidden units **reuse** lower-level features to compute more complex, general functions
  - Learning is **slow**: Learning high-level features is not independent of learning low-level features
- **Recurrent**: form of deep network that reuses features over time

Boltzmann Machine learning: Unsupervised version

- Visible units clamped to external "input" in positive phase
  - analogous to *outputs* in standard formulation
- Network "free-runs" in negative phase (nothing clamped)
- Network learns to make its free-running behavior look like its behavior when receiving input (i.e., learns to **generate** input patterns)

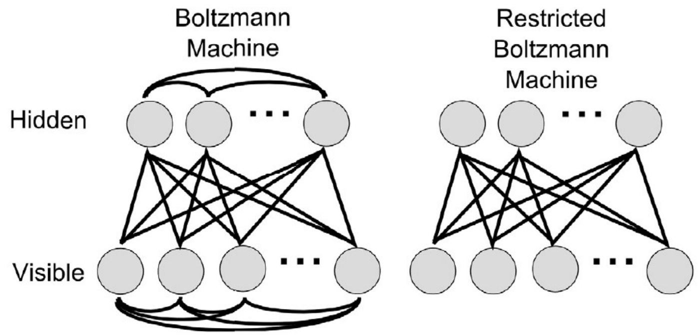


Objective function (unsupervised)

$$G = \sum_{\alpha} p^{+}(V_\alpha) \log \frac{p^{+}(V_\alpha)}{p^{-}(V_\alpha)} \qquad \left[G = \sum_{\alpha,\beta} p^{+}(I_\alpha, O_\beta) \log \frac{p^{+}(O_\beta | I_\alpha)}{p^{-}(O_\beta | I_\alpha)} \right]$$

V_α visible units in pattern α
p^{+} probabilities in *positive* phase [outputs (= "inputs") clamped]
p^{-} probabilities in *negative* phase [*nothing* clamped]

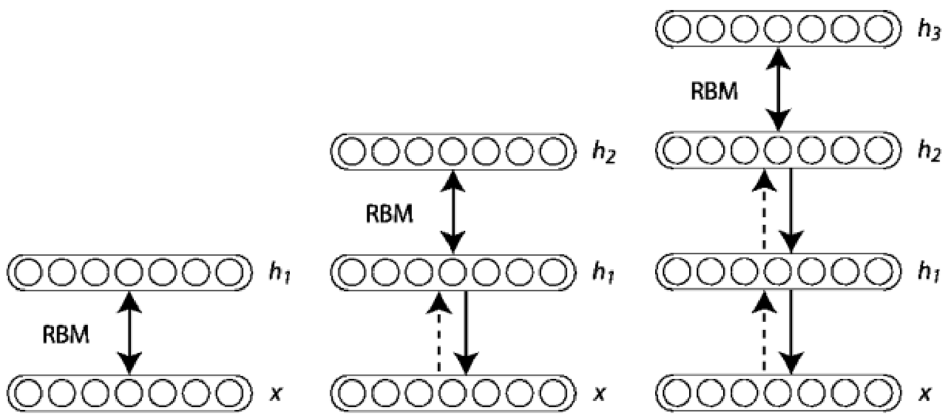
Restricted Boltzmann Machines



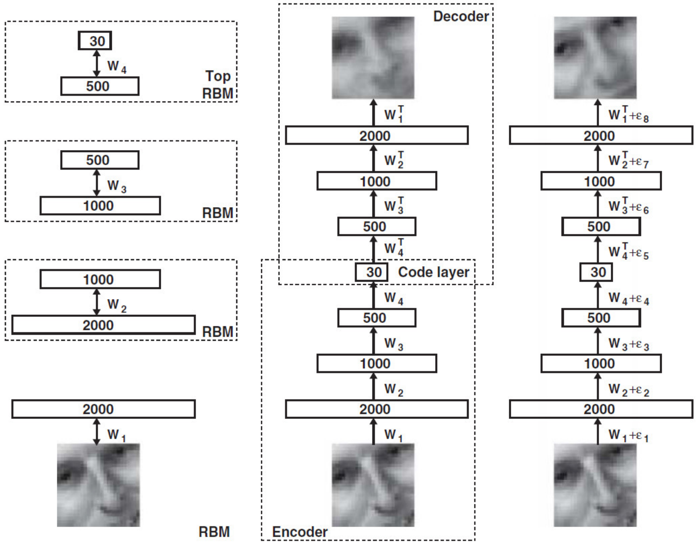
- No connections among units within a layer; allows fast settling
- Fast/efficient learning procedure
- Can be stacked; successive hidden layers can be **learned incrementally** (starting closest to the input) (Hinton)

Stacked RBMs

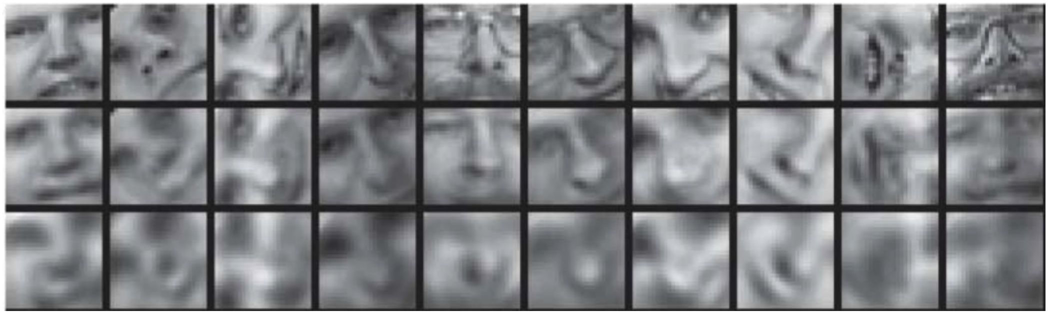
- Train iteratively; only use top-down (generative) weights in lower-level RBMs.



Deep autoencoder (Hinton & Salakhutdinov, 2006, *Science*)

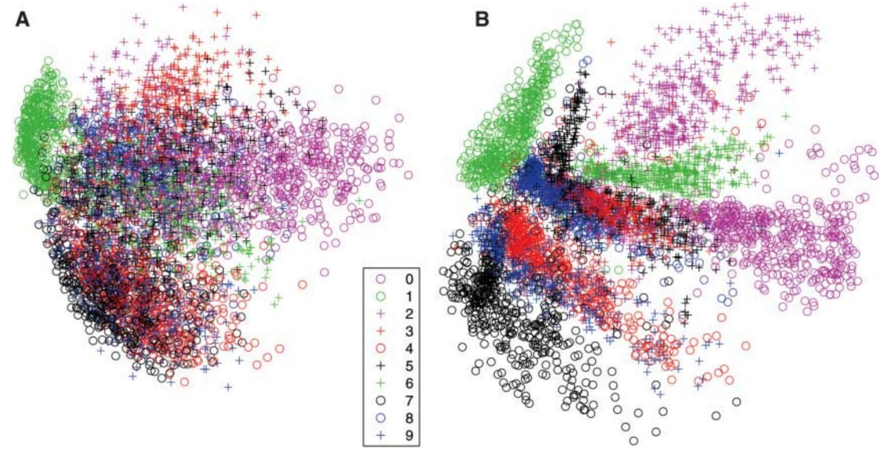


Face reconstructions (Hinton & Salakhutdinov, 2006)



Top: Original images in **test set**
Middle: Network reconstructions (30-unit bottleneck)
Bottom: PCA reconstructions (30 components)

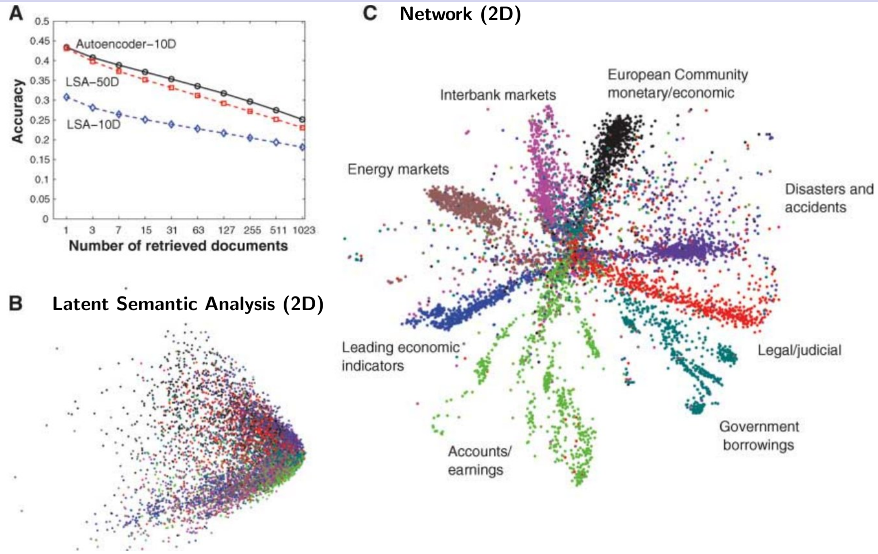
Digit reconstructions (Hinton & Salakhutdinov, 2006)



PCA reconstructions
(2 components)

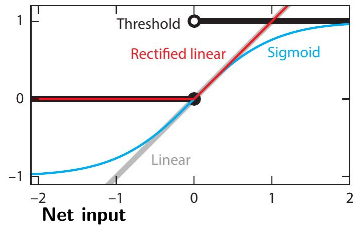
Network reconstructions
(2-unit bottleneck)

Document retrieval (Hinton & Salakhutdinov, 2006)

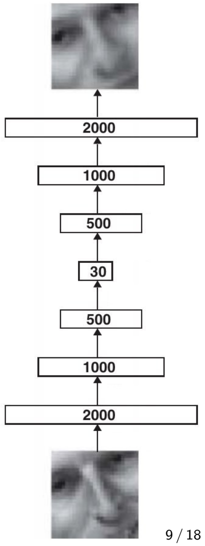


Deep learning with back-propagation

- Sigmoid function leads to extremely small derivatives for early layers (due to asympototes)
- Linear units preserve derivatives but cannot alter similarity structure
- **Rectified** linear units (ReLUs) preserve derivatives but impose (limited) non-linearity



- Often applied with **dropout**: On any given trial, only a random subset of units (e.g., half) actually work (i.e., produce output if input > 0).

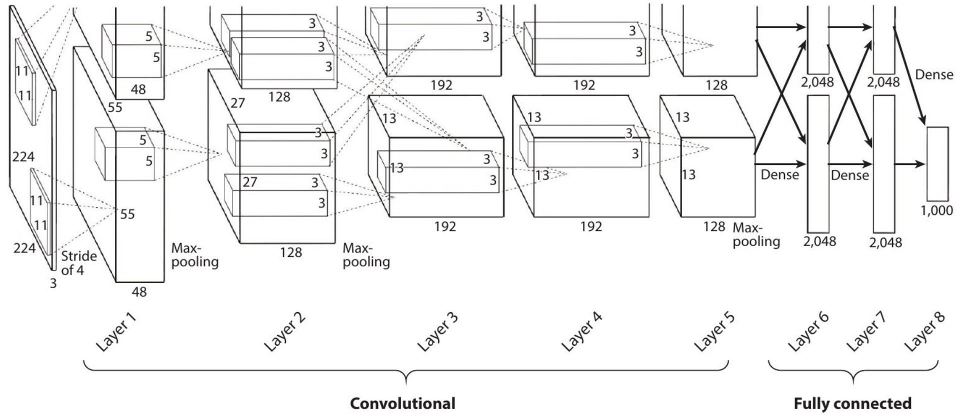


Online simulator

playground.tensorflow.org

Deep learning with back-propagation: Technical advances

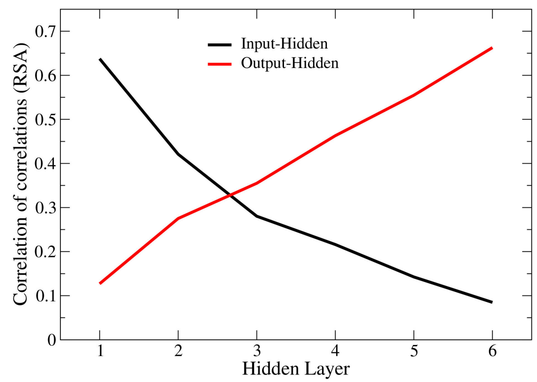
- Huge datasets available via the internet ("big data")
- Application of GPUs (Graphics Processing Units) for very efficient 2D image processing



Krizhevsky, Sutskever, and Hinton (2012, NIPS)

What does a deep network learn?

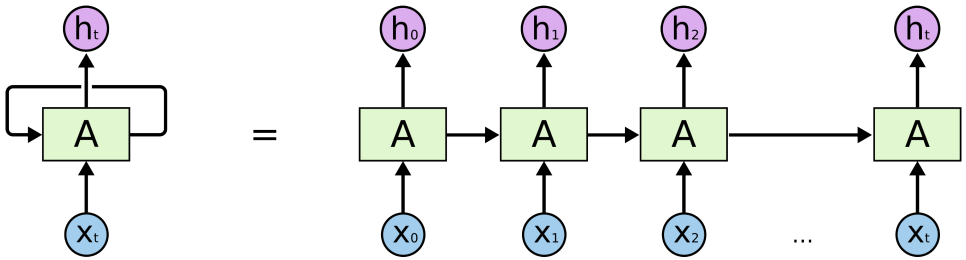
- Feedfoward network: 40 inputs to 40 outputs via 6 hidden layers (of size 40)
- Random input patterns map to random output patterns (n = 100)
- Compute pairwise similarities of representations at each hidden layer and compare to (correlate with) pairwise similarities of inputs and of outputs (⇒ *Representational Similarity Analysis*)
- Network gradually transforms from input similarity to output similarity



Promoting generalization

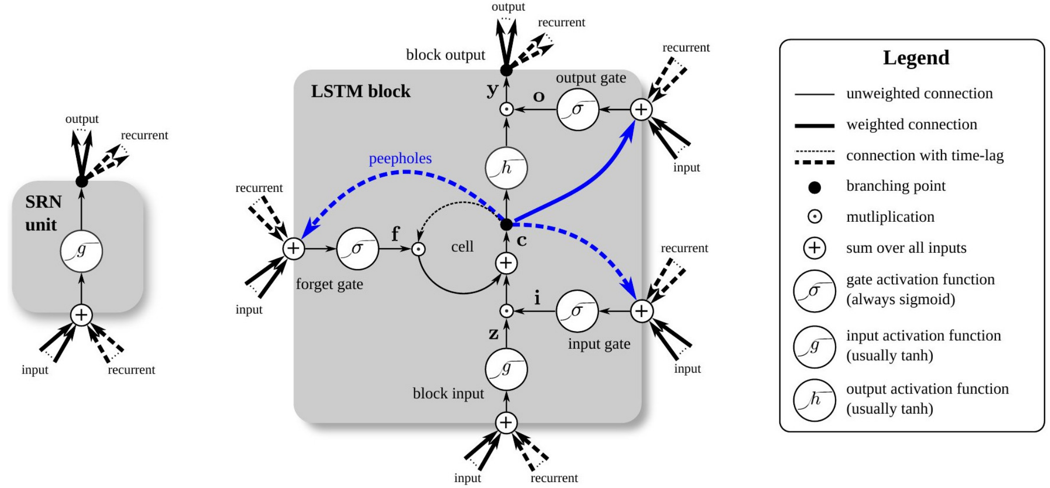
- Prevent overfitting by constraining network in a general way
  - weight decay, cross-validation
- Train on so much data that it's not possible to overfit
  - Including **fabricating new data** by transforming existing data in a way that you know the network must generalize over (e.g., viewpoint, color, lighting transformations)
  - Can also train an **adversarial** network to generate examples that produce high error
- Constrain structure of network in a way that forces a specific type of generalization
  - **Temporal invariance**
    Long short-term memory networks (LSTMs)
    Time-delay neural networks (TDNNs)
  - **Position invariance**
    Convolutional neural networks (CNNs)

Long short-term memory networks (LSTMs)

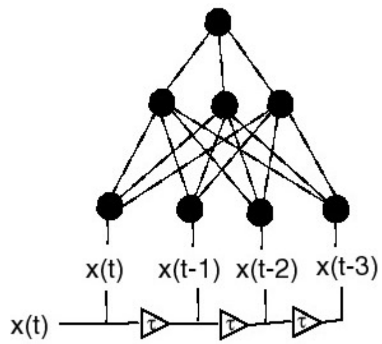


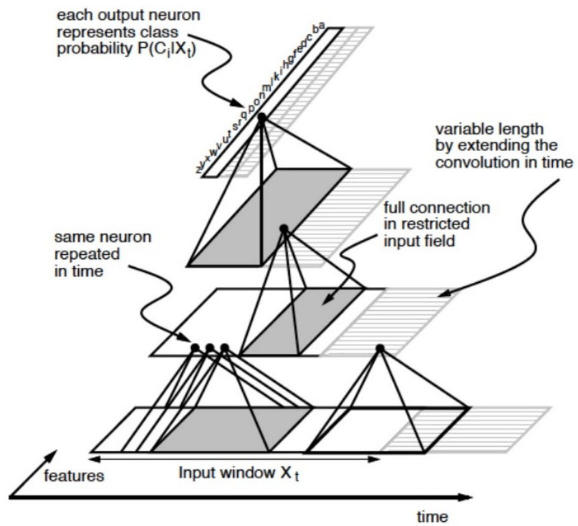
- Learning long-distance dependencies requires preserving information over multiple time steps
- Conventential networks (e.g., SRNs) must <u>learn</u> to do this
- LSTM networks use much more complex "units" that intrinsically preserve and manipulate information

Long short-term memory networks (LSTMs)

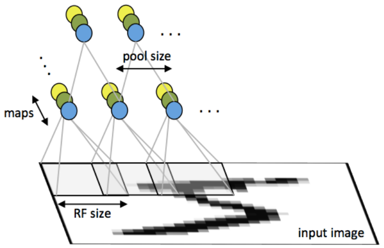


Time-delay neural networks (TDNNs)

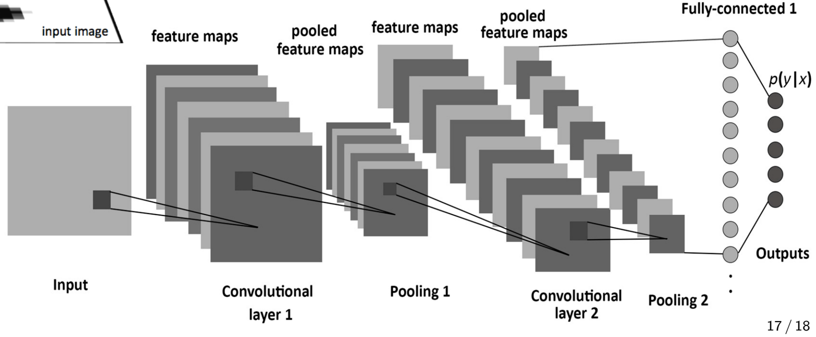




Convolutional neural networks (CNNs)

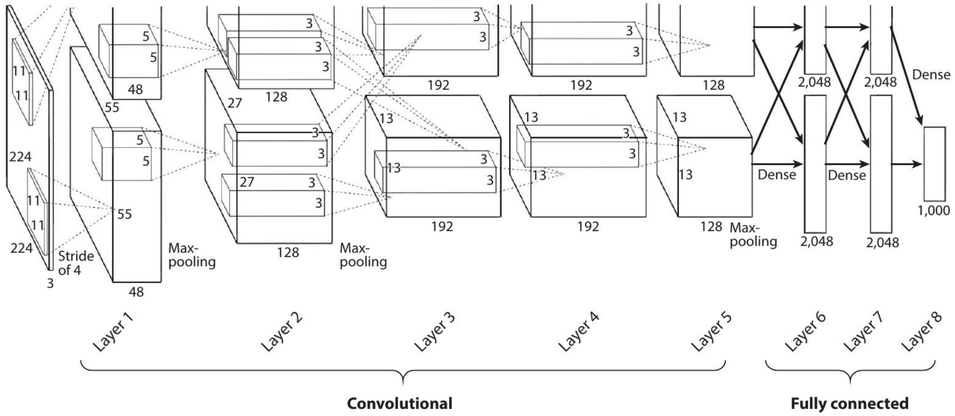


- Hidden units organized into feature maps (each using **weight sharing** to enforce identical receptive fields)
- Subsequent layer "pools" across features at similar locations (e.g., MAX function)



Deep learning with back-propagation: Technical advances

- Huge datasets available via the internet ("big data")
- Application of GPUs (Graphics Processing Units) for very efficient 2D image processing



Krizhevsky, Sutskever, and Hinton (2012, NIPS)