

Width vs. Depth

Dr. Fayyaz ul Amir Afsar Minhas

PIEAS Biomedical Informatics Research Lab

Department of Computer and Information Sciences

Pakistan Institute of Engineering & Applied Sciences

PO Nilore, Islamabad, Pakistan

<http://faculty.pieas.edu.pk/fayyaz/>

Universal Function Approximation

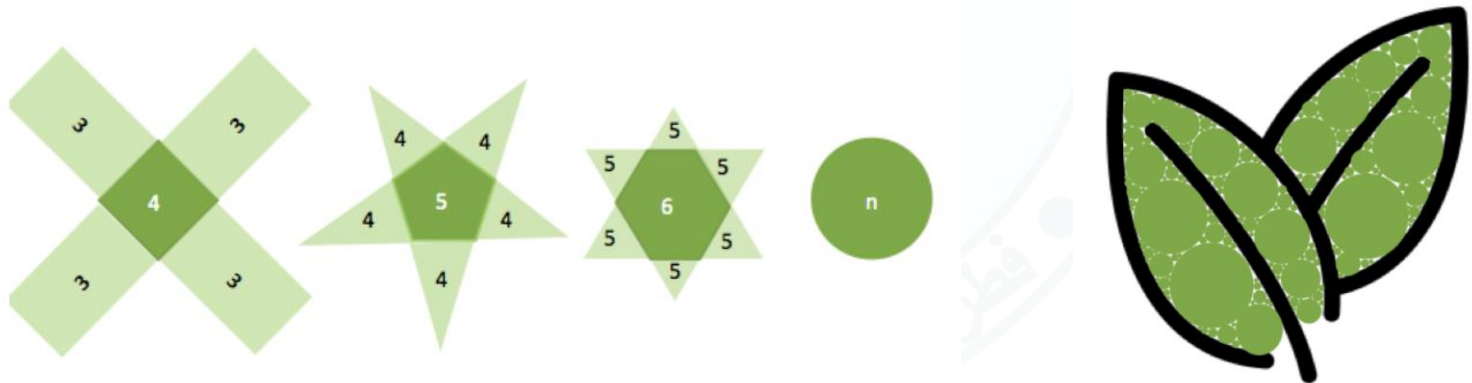
- Universal Approximation
 - Any function $f(\mathbf{x})$ over m inputs can be represented as follows:

$$F(\mathbf{x}) = \sum_{i=1}^N v_i \varphi(\mathbf{w}_i^T \mathbf{x} + b_i)$$

- $\varphi(\cdot)$ is a non-constant, bounded and monotonically-increasing continuous “basis” function
 - N is the number of functions
 - $F(\mathbf{x})$ is an approximation of $f(\mathbf{x})$, i.e., $|f(\mathbf{x}) - F(\mathbf{x})| < \epsilon$
- A neural network with a single hidden layer is a universal approximator
- A single hidden layer neuron with randomly initialized weights is a universal approximator
 - Extreme Learning Machine
 - Random Fourier Features as Kernel Approximators

Universal Function Approximation

- A neural network with one hidden layer can be used to approximate any shape
 - However, the approximation might require exponentially many neurons
 - How can we reduce the number of computations?



Wang, Haohan, and Bhiksha Raj. "On the Origin of Deep Learning." *arXiv:1702.07800 [Cs, Stat]*, February 24, 2017.
<http://arxiv.org/abs/1702.07800>.

Practical Issues in Universal Approximation

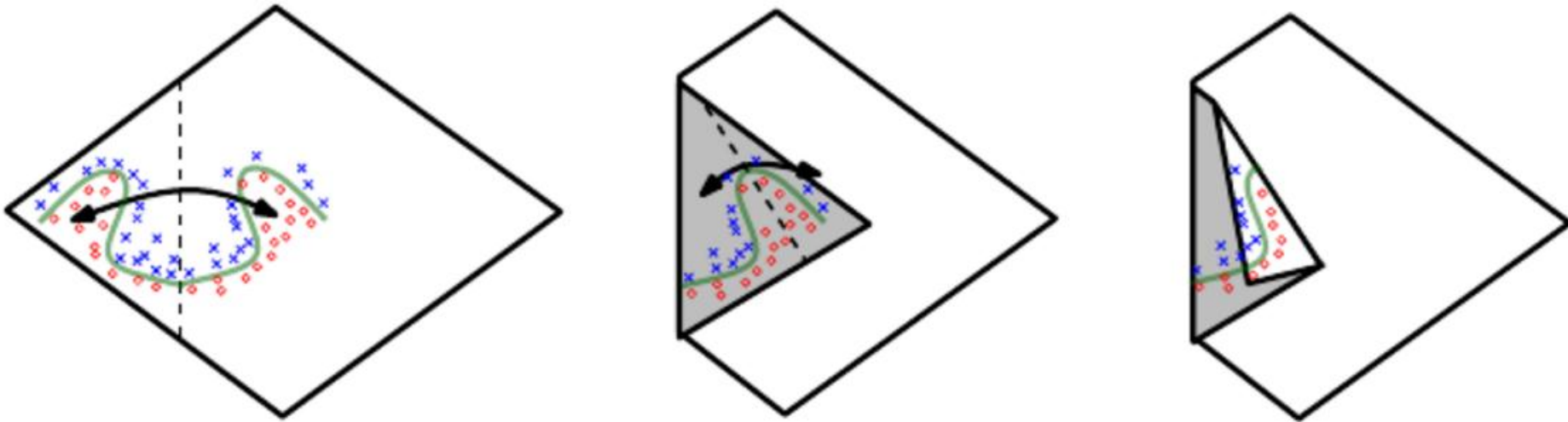
- The universal approximation theorem means that regardless of what function we are trying to learn, we know that a large MLP will be able to represent this function.
- However, we are not guaranteed that the training algorithm will be able to “learn” that function.
 - Optimization can fail
 - Learning is different from optimization
 - The primary requirement for learning is generalization

Width vs. Depth

- An MLP with a single hidden layer is sufficient to represent any function
 - But the layer may be infeasibly large
 - May fail to learn and generalize correctly
- Using a deeper model can reduce the number of units required to represent the desired function and can reduce the amount of generalization error
- A function that could be expressed with $O(n)$ neurons on a network of depth k required at least $O(2^{v^n})$ and $O((n-1)^k)$ neurons on a two-layer neural network: Delalleau and Bengio (2011)
- Functions representable with a deep rectifier net can require an exponential number of hidden units with a shallow (one hidden layer) network: Montufar (2014)
- For a shallow network, the representation power can only grow polynomially with respect to the number of neurons, but for deep architecture, the representation can grow exponentially with respect to the number of neurons: Bianchini and Scarselli (2014)
- Depth of a neural network is exponentially more valuable than the width of a neural network, for a standard MLP with any popular activation functions: Eldan and Shamir (2015)

Width vs. Depth

- Exponential advantage of deeper networks



Width vs. Depth

- Empirical results for some data showed that depth increases generalization performance in a variety of applications

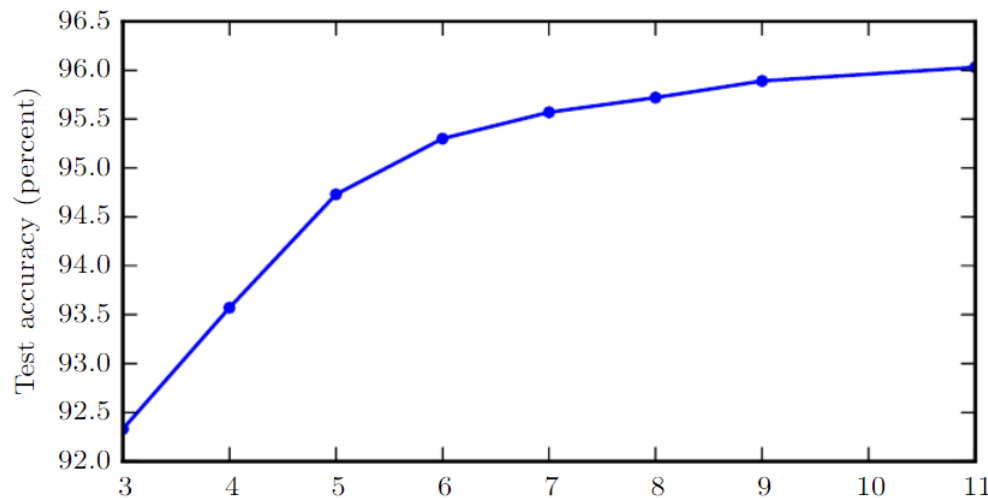


Figure 6.6: Empirical results showing that deeper networks generalize better when used to transcribe multi-digit numbers from photographs of addresses. Data from [Goodfellow et al. \(2014d\)](#). The test set accuracy consistently increases with increasing depth. See figure 6.7 for a control experiment demonstrating that other increases to the model size do not yield the same effect.

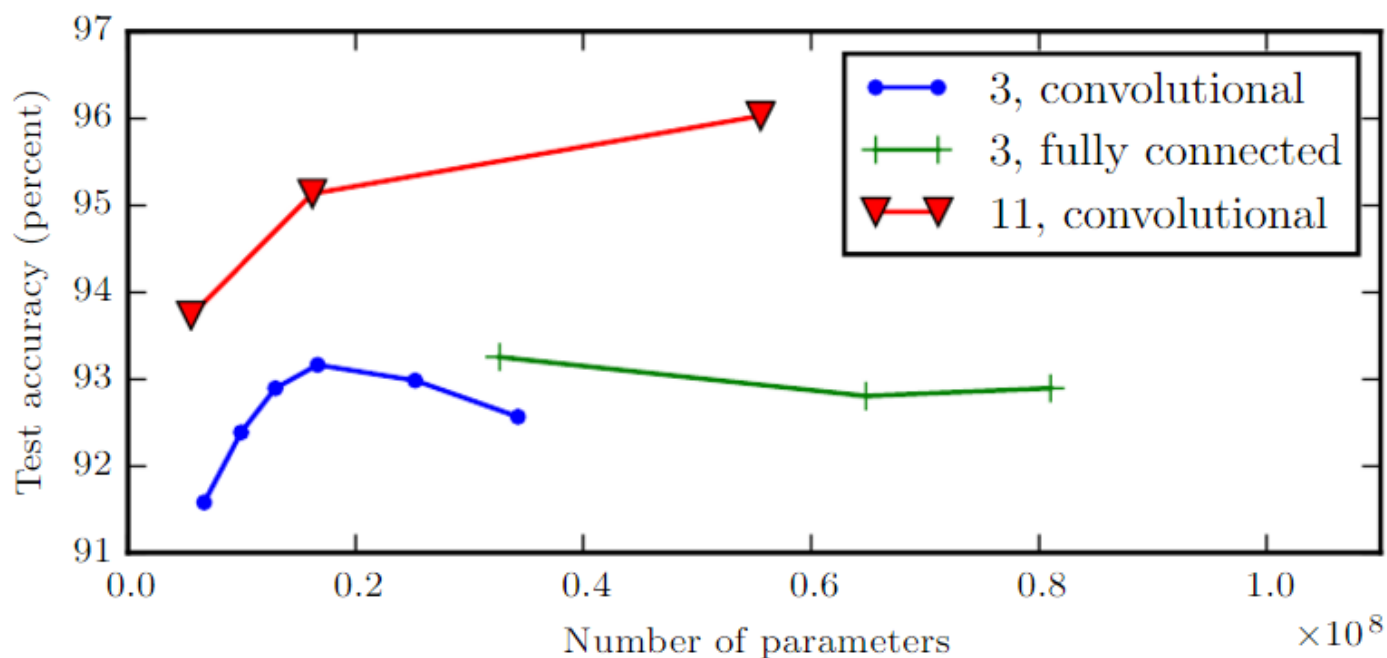
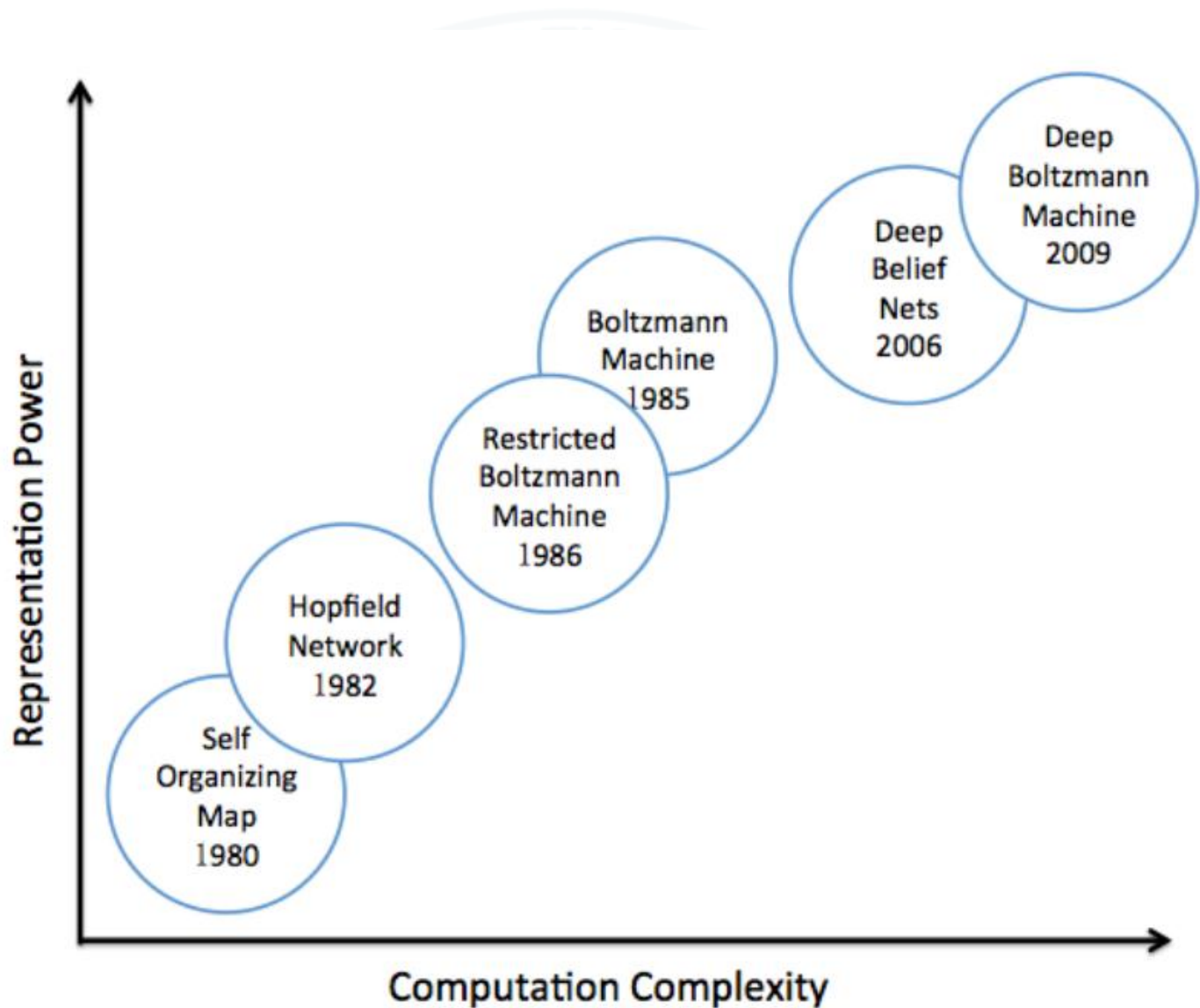


Figure 6.7: Deeper models tend to perform better. This is not merely because the model is larger. This experiment from [Goodfellow *et al.* \(2014d\)](#) shows that increasing the number of parameters in layers of convolutional networks without increasing their depth is not nearly as effective at increasing test set performance. The legend indicates the depth of network used to make each curve and whether the curve represents variation in the size of the convolutional or the fully connected layers. We observe that shallow models in this context overfit at around 20 million parameters while deep ones can benefit from having over 60 million. This suggests that using a deep model expresses a useful preference over the space of functions the model can learn. Specifically, it expresses a belief that the function should consist of many simpler functions composed together. This could result either in learning a representation that is composed in turn of simpler representations (e.g., corners defined in terms of edges) or in learning a program with sequentially dependent steps (e.g., first locate a set of objects, then segment them from each other, then recognize them).

Increasing representation power with depth



Issues with Depth

- Generalization
 - Large networks are large capacity machines
 - Remember: Learning requires generalization and goes beyond mere minimization of an objective function!
- Failure to Optimize
 - Random initialization leads to the network being stuck in poor solutions
 - Deeper networks are more prone to vanishing gradients and optimization failures
 - “Greedy Layer-Wise Training of Deep Networks” by Bengio et al., 2006
 - Uses unsupervised pre-training to initialize the weights of a network such that the optimization becomes easier
 - Since 2010, this has been replaced with Drop-out and batch-normalization schemes which improve the optimization performance
 - Rectified Linear Units get rid of the vanishing gradient problem
 - Drop-out improves generalization
 - Batch Normalization accelerates deep learning and improves generalization
 - » Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift by Ioffe and Sze, 2015.
- Large scale optimization is tricky in deep learning
 - Computationally demanding
 - Requires efficient methods
 - Stochastic gradient and sub-gradient methods














Issues with depth

- Handling variety of neural network architectures
 - How can we develop a framework of learning in which we can add layers, have a large diversity of layer connectivity, change objective functions and losses, layer connectivity, regularization, etc.?
 - And still solve the optimization problem in an efficient manner!
 - Symbolic Computation and Automatic Differentiation
 - GPU
 - Efficient matrix operations
 - Higher bandwidth

A mostly complete chart of Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

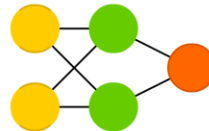
<http://www.asimovinstitute.org/neural-network-zoo/>

-  Backfed Input Cell
-  Input Cell
-  Noisy Input Cell
-  Hidden Cell
-  Probabilistic Hidden Cell
-  Spiking Hidden Cell
-  Output Cell
-  Match Input Output Cell
-  Recurrent Cell
-  Memory Cell
-  Different Memory Cell
-  Kernel
-  Convolution or Pool

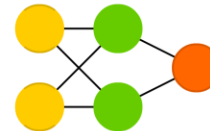
Perceptron (P)



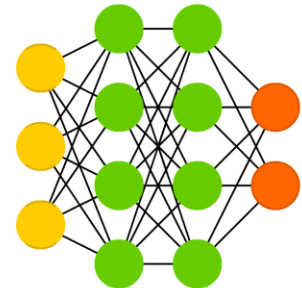
Feed Forward (FF)



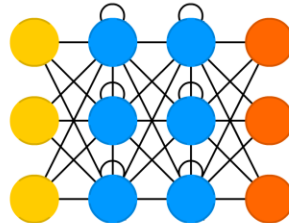
Radial Basis Network (RBF)



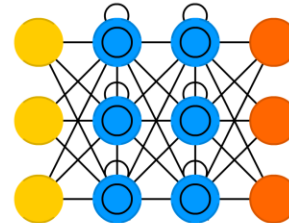
Deep Feed Forward (DFF)



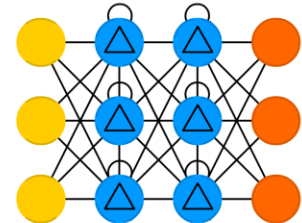
Recurrent Neural Network (RNN)



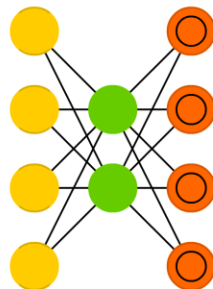
Long / Short Term Memory (LSTM)



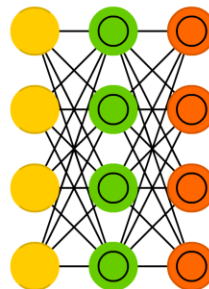
Gated Recurrent Unit (GRU)



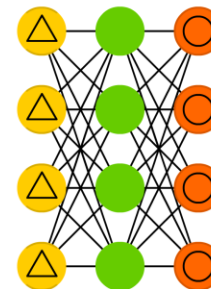
Auto Encoder (AE)



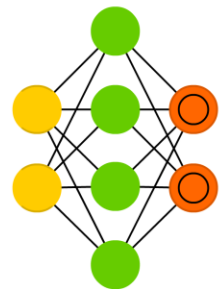
Variational AE (VAE)



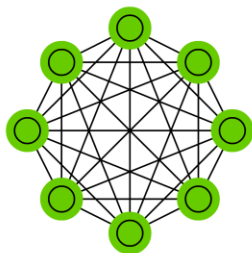
Denoising AE (DAE)



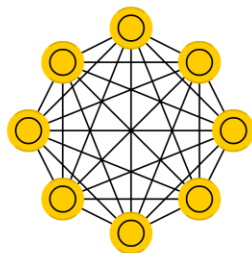
Sparse AE (SAE)



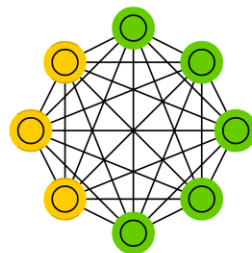
Markov Chain (MC)



Hopfield Network (HN)



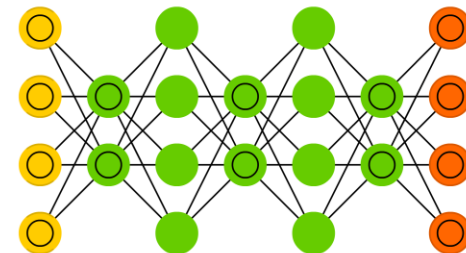
Boltzmann Machine (BM)



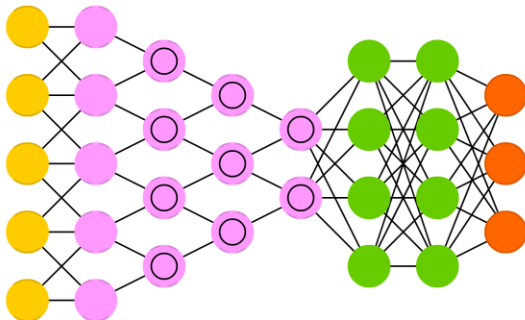
Restricted BM (RBM)



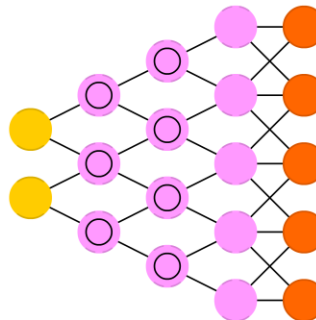
Deep Belief Network (DBN)



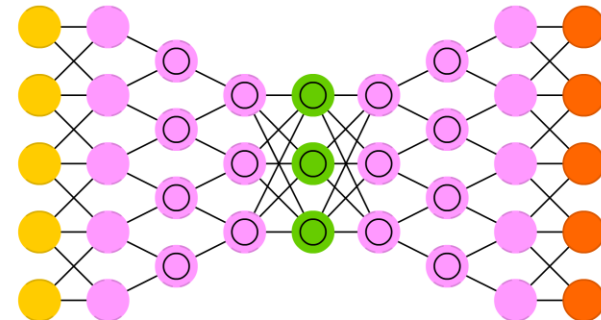
Deep Convolutional Network (DCN)



Deconvolutional Network (DN)



Deep Convolutional Inverse Graphics Network (DCIGN)



Backfed Input Cell

Input Cell

Noisy Input Cell

Hidden Cell

Probabilistic Hidden Cell

Spiking Hidden Cell

Output Cell

Match Input Output Cell

Recurrent Cell

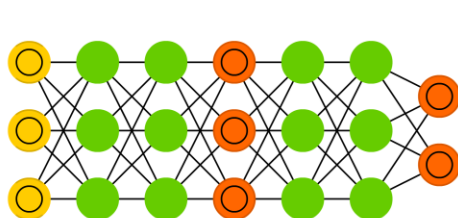
Memory Cell

Different Memory Cell

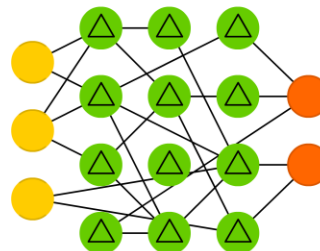
Kernel

Convolution or Pool

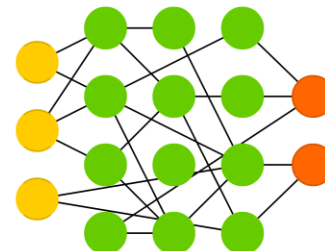
Generative Adversarial Network (GAN)



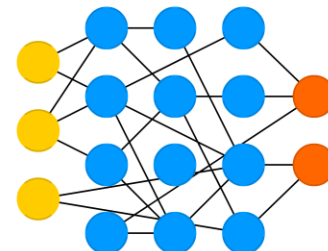
Liquid State Machine (LSM)



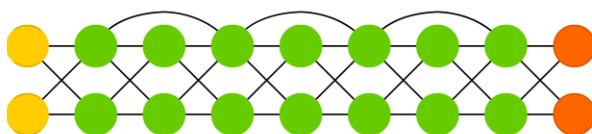
Extreme Learning Machine (ELM)



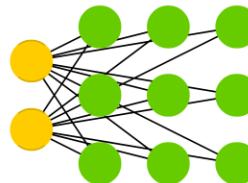
Echo State Network (ESN)



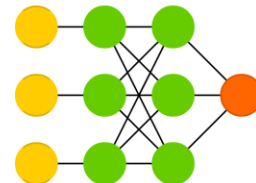
Deep Residual Network (DRN)



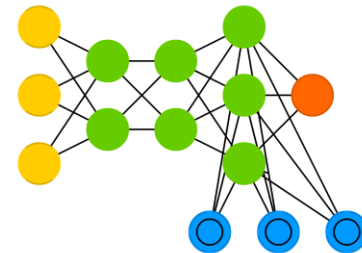
Kohonen Network (KN)



Support Vector Machine (SVM)



Neural Turing Machine (NTM)





End of Lecture-1

We want to make a machine that will be
proud of us.

- Danny Hillis