Neural Networks-Part 2

Vidya Samadi, PhD. , M.ASCE

Clemson University

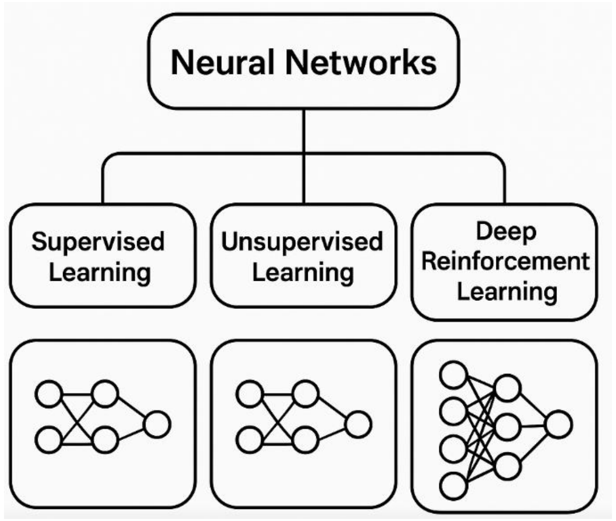
July 31, 2025

Relevant reading

- Intro to Autoencoders
- Hopfield Nets and Boltzmann Machines
- Neural machine translation with a Transformer and Keras
- Introduction to RL and Deep Q Networks
- Understanding masking & padding

Major Types of Neural Networks

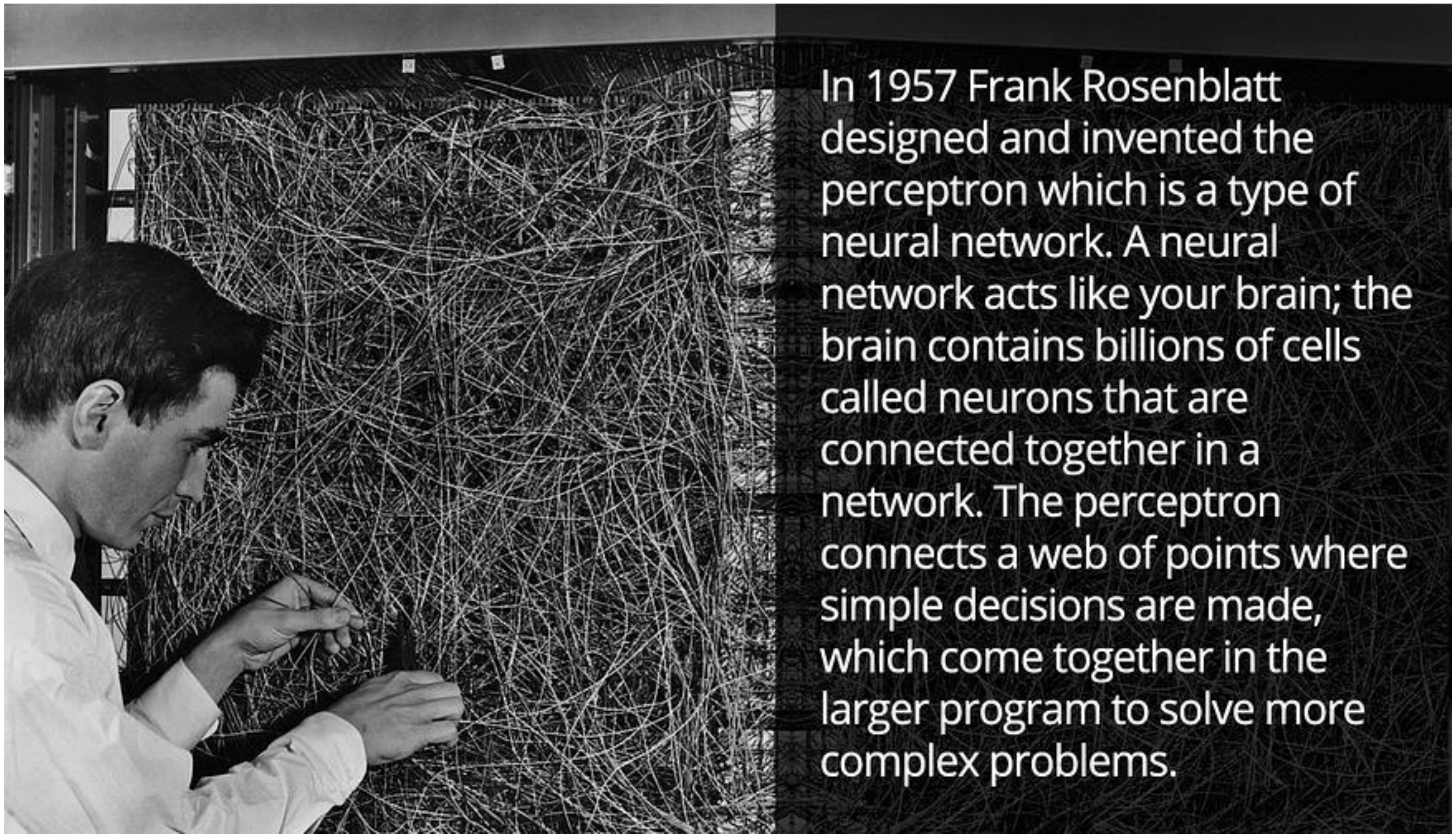
- Observe set of example: training data
- Infer the process that generated that data
- Use inference to make predictions about previously unseen data: test data
- Three major types



- **Supervised Learning:** given a set of feature/label pairs, find a rule that predicts the label associated with previously unseen input.
- **Unsupervised Learning:** given a set of feature vectors (without labels) group them into "clusters" (or create labels for groups).
- **Reinforcement Learning:** uses observations gathered from the interaction with its environment to take actions that would maximize the reward function. The reinforcement learning algorithm (called the agent) continuously learns from its environment using iteration. Example: computers reaching a super-human state and beating humans on computer games.

What is Perceptron?

The Perceptron is a foundational algorithm in neural networks.



In 1957 Frank Rosenblatt designed and invented the perceptron which is a type of neural network. A neural network acts like your brain; the brain contains billions of cells called neurons that are connected together in a network. The perceptron connects a web of points where simple decisions are made, which come together in the larger program to solve more complex problems.

The perceptron by Frank Rosenblatt (source: Machine Learning Department at Carnegie Mellon University)

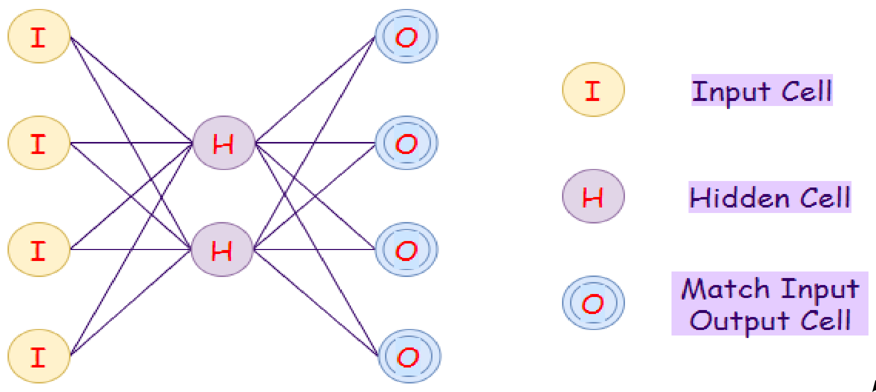
Auto Encoder (AE)

- In an autoencoder, the number of hidden cells is smaller than the input cells.
- The number of input cells in autoencoders equals to the number of output cells. On an AE network, we train it to display the output, which is as close as the fed input, which forces AEs to find common patterns and generalize the data.
- The algorithm is relatively simple as it requires output to be the same as the input.

Applications: Classification, clustering, & feature compression.

Encoder: Convert input data in lower dimensions.
Decoder: Reconstruct the compressed data.

Auto Encoder (AE)

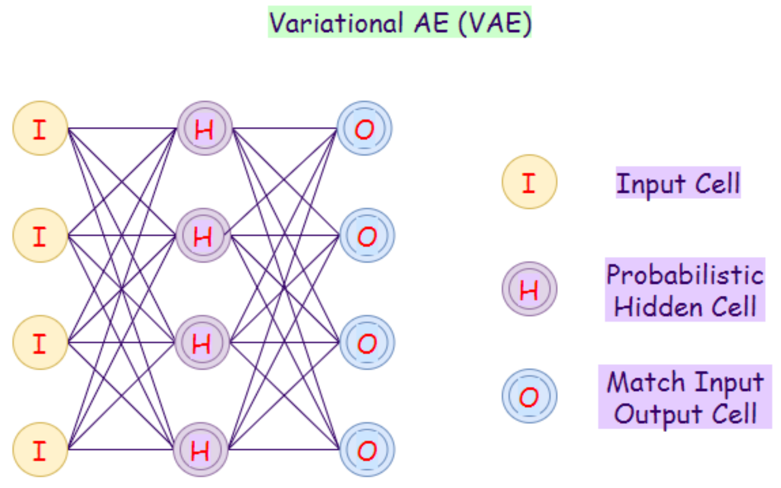


Variational Autoencoder (VAE)

- VAE uses a probabilistic approach for describing observations. It shows the probability distribution for each attribute in a feature set.
- It is a powerful generative model that learn the latent representations of input data as random variables.

Applications: Interpolation, automatic image generation, and time series data augmentation.

Encoder: Convert input data in lower dimensions.
Decoder: Reconstruct the compressed data.

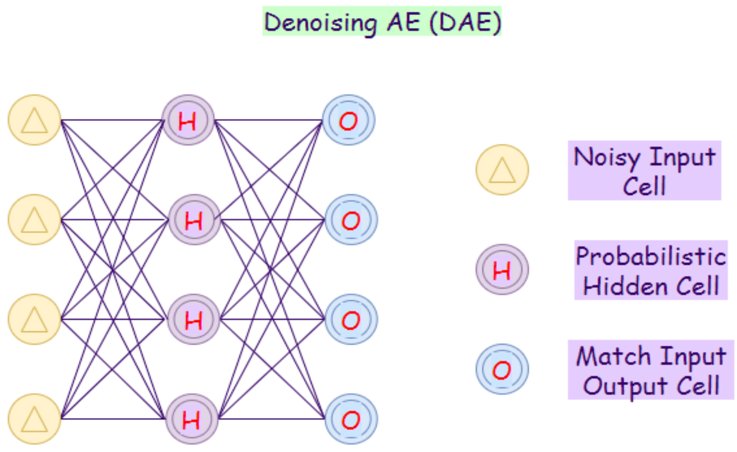


Denoising Autoencoder (DAE)

- DAE is one type of autoencoder that we can use to reduce the noise in input data .
- The algorithm intentionally introduces noises into the input during training and forces the hidden layer to learn more robust features so that the output is a more refined version of the noisy input.
- learns to reconstruct clean data from corrupted or noisy versions of the same data.

Applications: Feature extraction & data dimensionality reduction.

Encoder: Convert input data in lower dimensions.
Decoder: Reconstruct the compressed data.

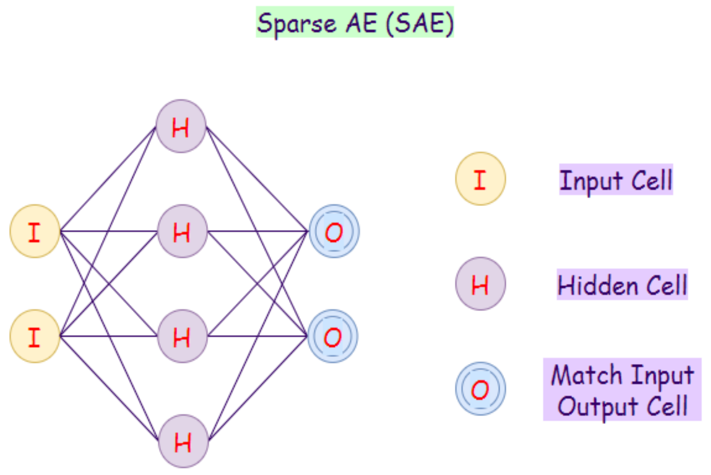


Sparse Autoencoder (SAE)

- SAE encourages sparsity in its hidden layer activations. This means that during training, the autoencoder learns to represent data using only a small subset of its hidden neurons, leading to more interpretable and efficient representations. Reduce noise in data.
- Some studies showed that the sparse autoencoder actually learns better representation than autoencoder.

Applications: Feature extraction & data dimensionality reduction.

Encoder: Convert input data in lower dimensions.
Decoder: Reconstruct the compressed data.

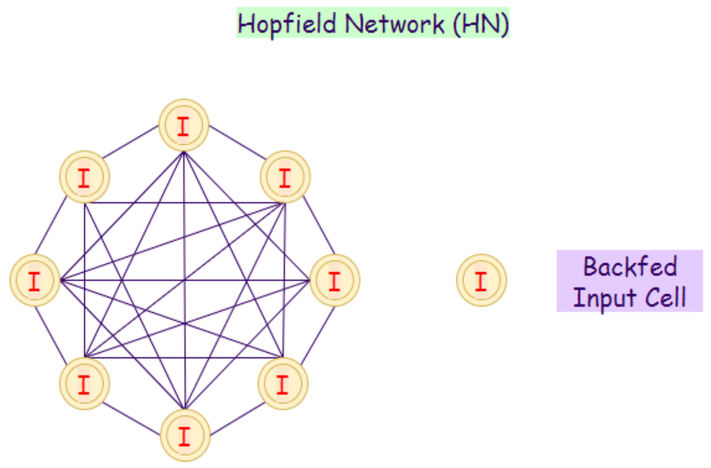


Hopfield Neural Network (HNN)

- HNN is a recurrent neural networks capable of storing and retrieving multiple memories.
- Every neuron is connected with other neurons directly.
- In this network, a neuron is either ON or OFF. The state of the neurons can change by receiving inputs from other neurons.
- When we train a neural network on a set of patterns, it can then recognize the pattern even if it is somewhat distorted or incomplete(returns the best guess to complete the pattern).

Applications: Optimization problems, image detection & recognition.

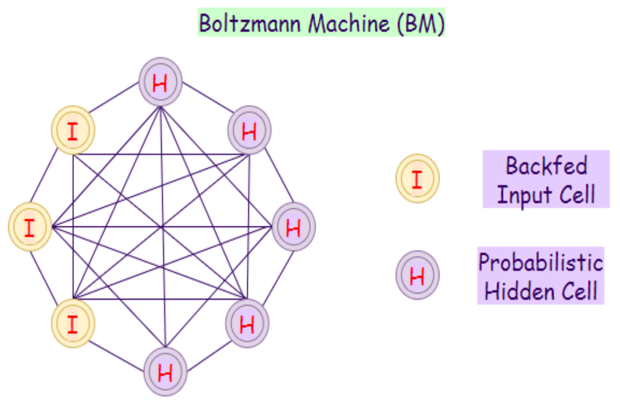
Can we combine HNN with LSTM?



Deep Boltzmann Machine (DBM)

- The original BM is invented by Hinton and Sejnowski in 1985 as an Unsupervised Deep Learning. DBM is introduced in 2009.
- This model is based on Boltzmann Distribution (also known as Gibbs Distribution) which is an integral part of Statistical Mechanics and helps us to understand the impacts of algorithmic parameters on simulation models
- In DBMs, there are input nodes and hidden nodes, as soon as all our hidden nodes change its state, our input nodes transform into output nodes.
- DBMs use RBMs (Restricted Boltzmann Machines) as building blocks, which were also developed from the original Boltzmann Machines to improve learning efficiency.

It is also known as Generative Deep Learning model.

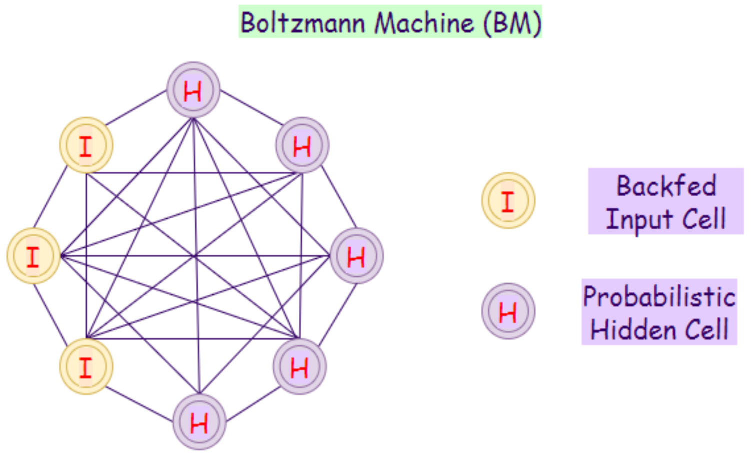


Deep Boltzmann Machine (DBM)-cont.

- Deep Boltzmann Machines are primarily divided into two categories: Energy-based Models (EBMs) and Restricted Boltzmann Machines (RBM)– when these RBMs are stacked on top of each other, they are known as Deep Belief Networks.

- Deep Belief Networks are further sub-divided into Greedy Layer-Wise Training and Wake-Sleep Algorithm– these are known as complex Neural Networks models.

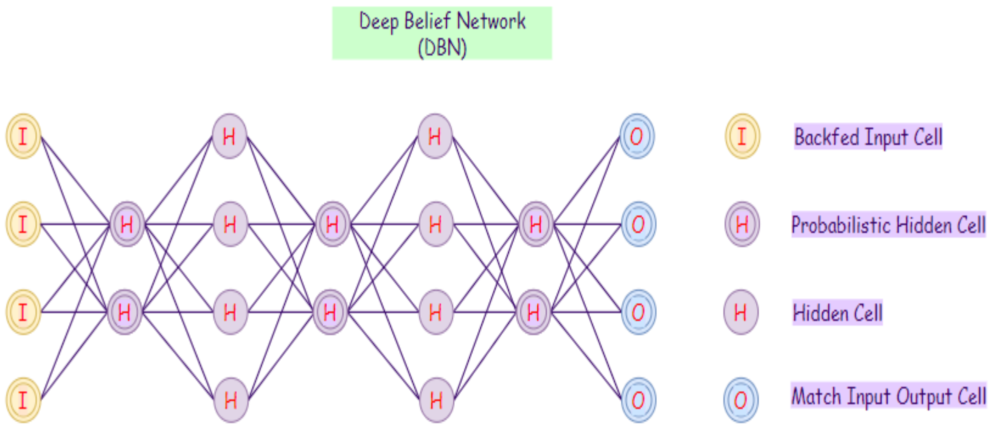
Applications: Dimensionality reduction, classification, regression and feature learning.



Deep Belief Network (DBN)

- DBNs contain many hidden layers– these hidden layers help DBN to learn without any supervision – we call DBNs an unsupervised algorithm!
- After unsupervised training, we can train our model with supervision methods to perform regression.
- DBNs represent as a composition of Restricted Boltzmann Machines and Autoencoders.
- DBNs use a probabilistic approach toward its results.

Applications: Retrieval of images, non-linear dimensionality reduction.



Many hidden layers-- act as a feature detector.

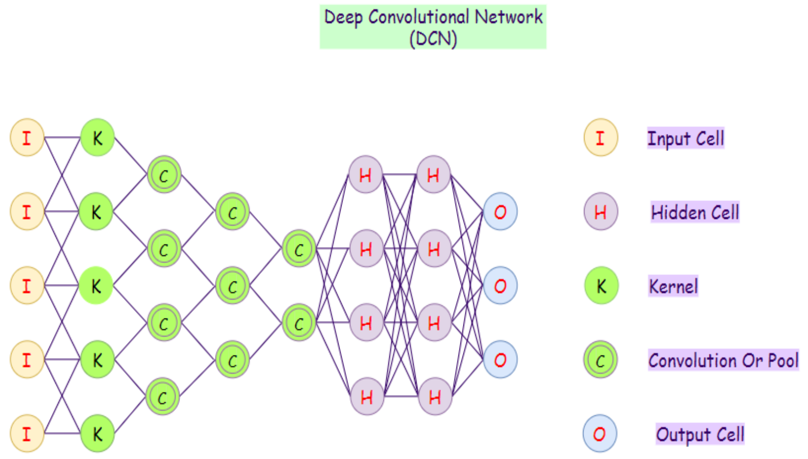
Deep Convolutional Network (DCN)

- DCNs are neural networks used primarily for classification of images, clustering of images and object recognition.
- DNNs enable unsupervised construction of hierarchical image representations.
- Remote sensing data, satellite data, SMAP data for soil moisture, etc.

Applications: Image recognition, video analysis, Natural Language Processing (NLP), & time series forecasting.

A **pooling layer** merges the features collected from one **convolution** layer before passing them to the next, thereby increasing computation efficiency!

A **kernel** is a matrix, which is slid across the image and multiplied with the input such that the output is enhanced in a certain desirable manner.

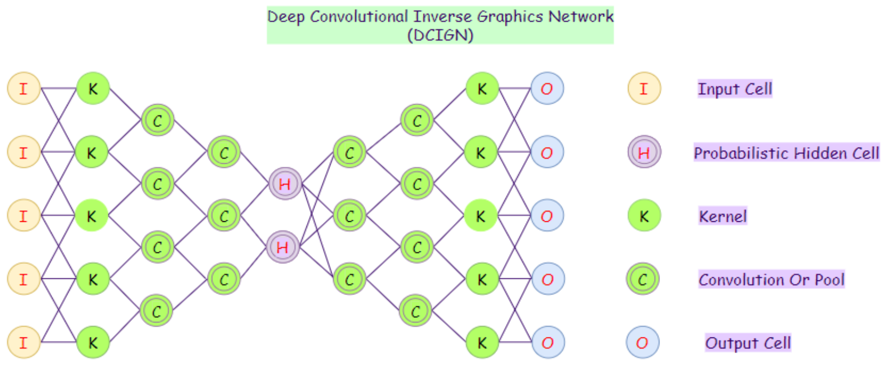


Deep Convolutional Inverse Graphics Network (DC-IGN)

- DC-IGN aim at relating graphics representations to images.
- The model uses elements like lighting, object location, texture, and other aspects of image design for image processing. It uses various layers to process input and output.
- DC-IGN uses initial layers to encode through various convolutions, utilizing max pooling, and then uses subsequent layers to decode with unspooling.

Applications: Manipulation of images– can be used for RS images, river cam images, etc.

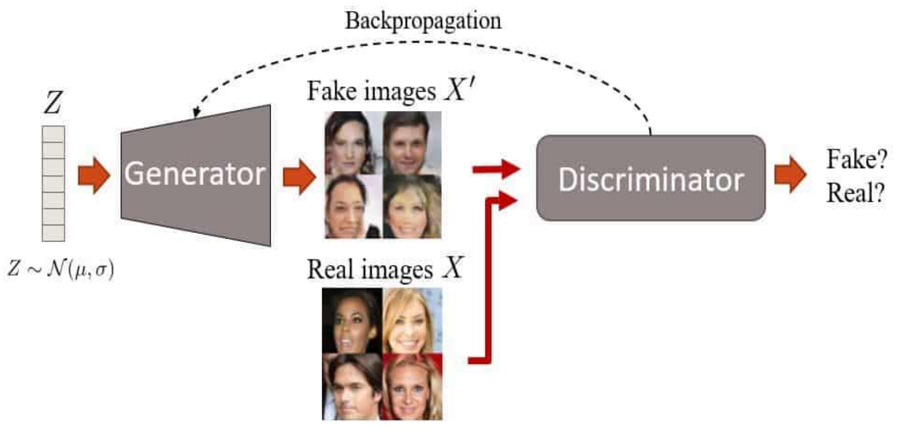
A model that learns an interpretable representation of images.

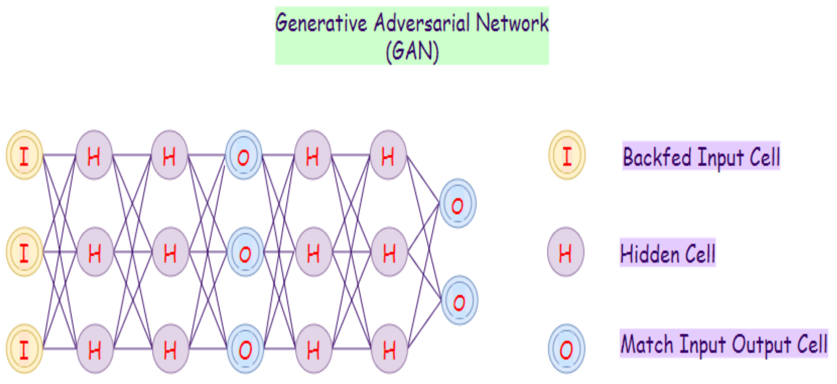

Deep Convolutional Inverse Graphics Network (DCIGN)
I
K
C
H
O
Input Cell
Probabilistic Hidden Cell
Kernel
Convolution Or Pool
Output Cell

Generative Adversarial Network (GAN)

- Given training data, GANs learn to generate new data with the same statistics as the training data.
- In GANs, there is a Generator and a Discriminator. The Generator generates fake samples of data (an image) and tries to fool the Discriminator. The Discriminator, on the other hand, tries to distinguish between the real and fake samples.
- The steps are repeated several times and in this, the Generator and Discriminator get better and better in their respective jobs after each repetition.

Applications: image generation.





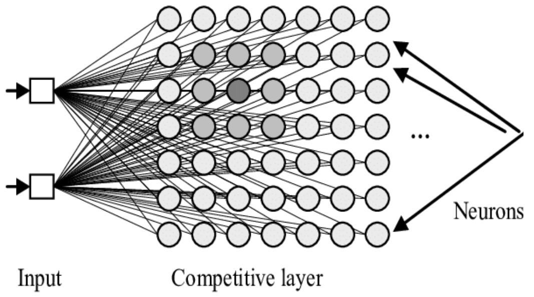
Developed by Goodfellow et al., 2014.

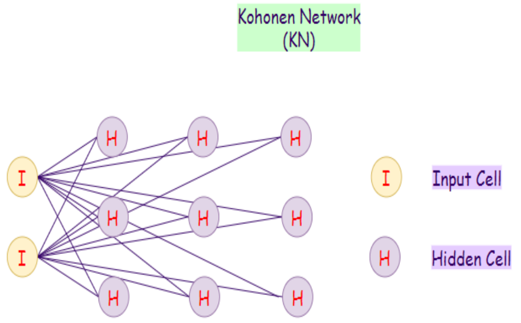
https://www.iangoodfellow.com/slides/

Kohonen Network (KN)

- The basic idea behind the KN is to set up a structure of interconnected processing units ("neurons") which compete for the signal-- they use competitive learning rather than error correction learning.
- KN is also known as self-organizing maps, which is very useful when dealing with the data that is scattered in many dimensions, and we want it in one or two dimensions only.
- Use for dimensionality reduction and visualization of high dimensional data.
- Use Rectangular Grid Topology and Hexagonal Grid Topology for dimensionality reduction. A Hexagonal Grid is a grid formed by a tessellation of regular hexagons.

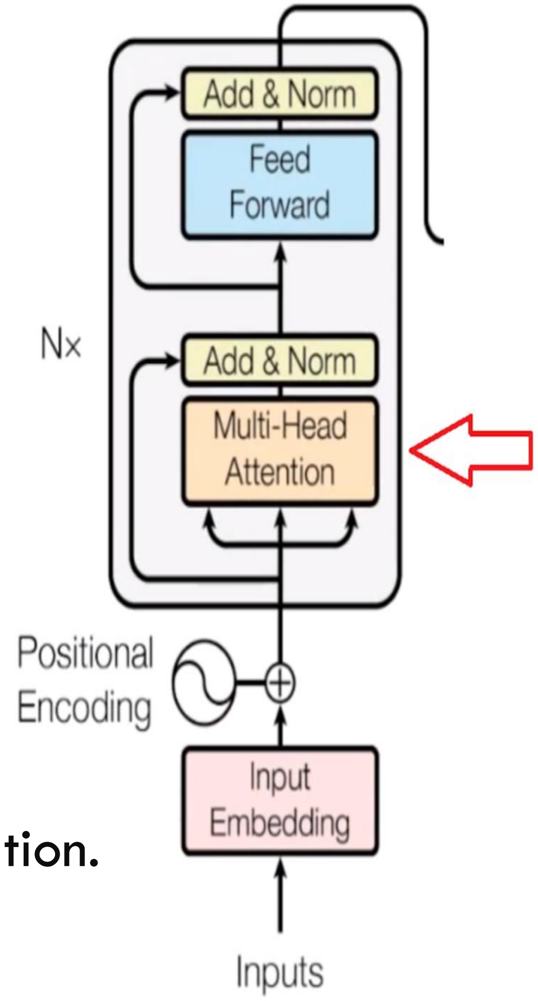
Applications: Dimensionality reduction, assessment and prediction of time series data particularly water quality, & coastal water management.





Transformer Neural Network (TNN)

- TNN is a novel architecture that aims to solve sequence-to-sequence tasks while handling long-range dependencies- very new NN algorithms.
- TNN replace CNNs and RNNs with self-attention. Self attention allows Transformers to easily transmit information.
- TNNs are the heart of pretty much everything exciting in AI right now including ChatGPT, Google Translate and many others.
- TNN can be used to predict what will happen next, or to analyze what is happening in a specific sequence.



Applications: Time series prediction, text classification.

Optuna

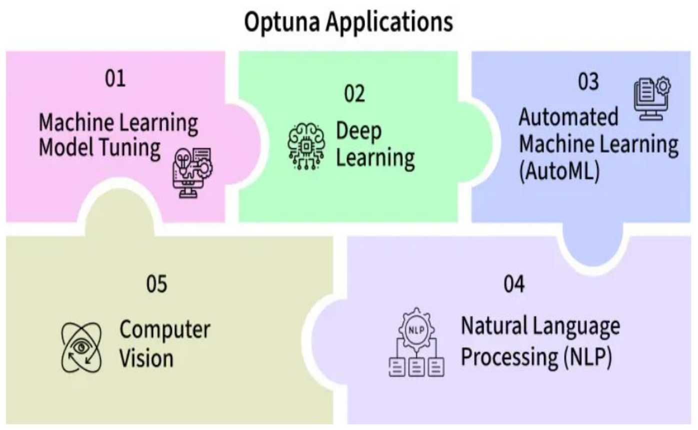
- Optuna is an open-source hyperparameter optimization framework that can be effectively used to tune neural networks.
- Advanced algorithms are employed to efficiently search for optimal hyperparameters, which are crucial for the performance of neural networks. These algorithms include

  - **Tree-structured Parzen Estimator:** A Bayesian optimization algorithm that builds a probabilistic model of the objective function to guide the search towards promising regions.

  - **Random Search:** A baseline method that randomly samples hyperparameters.

  - **Genetic Algorithms (e.g., Non-dominated Sorting Genetic Algorithm II; NSGA-II):** Evolutionary algorithms that evolve a population of hyperparameter sets to find optimal solutions.

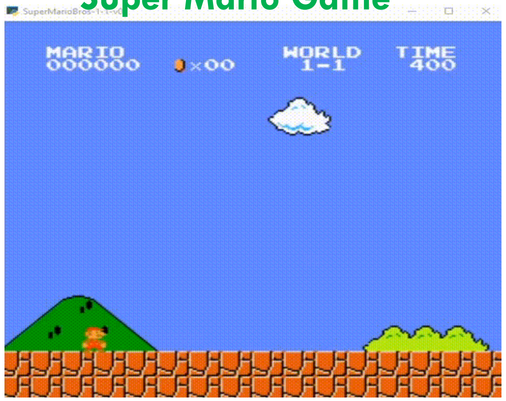
Applications: integration with popular deep learning frameworks like PyTorch and TensorFlow/Keras, allowing to define neural network training within Optuna's objective function.


Optuna Applications
01 Machine Learning Model Tuning
02 Deep Learning
03 Automated Machine Learning (AutoML)
04 Natural Language Processing (NLP)
05 Computer Vision

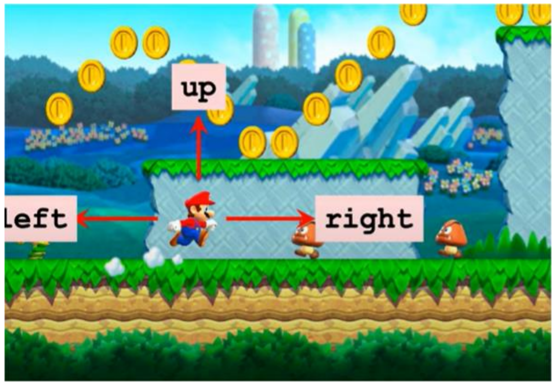
Deep Reinforcement Learning - Terminology

What is the policy? Policy means make a decision based on current state!

Super Mario Game



Action



Reward

π(a | s)= the probability density function of taking A=a given state s, for example

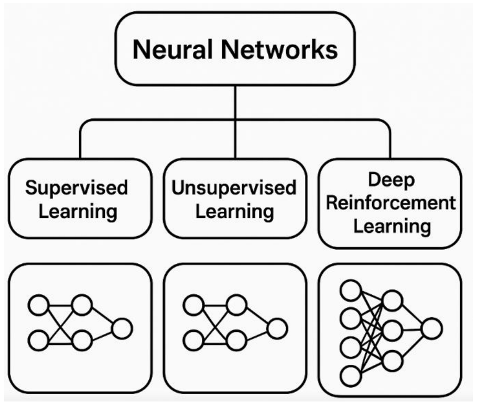
policy π

Upon observing state $S = s$, the agent's action A can be random.

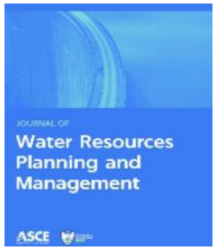
π(left | s)=0.2

π(right | s)=0.1

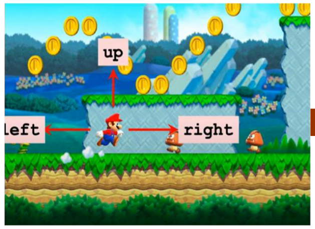
π(up | s)=0.7

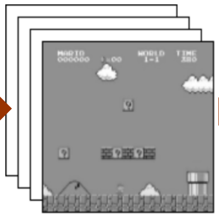


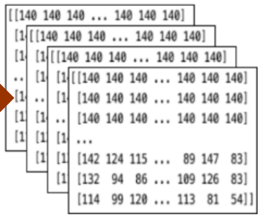
Deep Reinforcement Learning - Random or Deterministic Policy!



Sadeghi & Samadi, 2024







```
[[140 140 140 ... 140 140 140]
 [140 140 140 ... 140 140 140]
 [140 140 140 ... 140 140 140]
 ...
 [142 124 115 ...  89 147  83]
 [132  94  86 ... 109 126  83]
 [114  99 120 ... 113  81  54]]
```

Rock Paper Scissors is a Random Policy!

State Transition

$$\textit{old state} \longrightarrow \textit{new state}$$

For example, "up" action leads to a new state

- State transition can be both **deterministic** or **random**—but we mostly use random and **Markov Chain** method!
- Randomness comes from the environment!

Reward R!

Collect a coin:	$R = +1$
Win the game:	*R=+10000*
Touch a Goomba (game over):	*R=-10000*
Nothing happens:	*R=0*

$$p\,(s' \mid s\,, a) = P\,(S'=s' \mid S=s,\, A=a)$$

Conditional probability density function

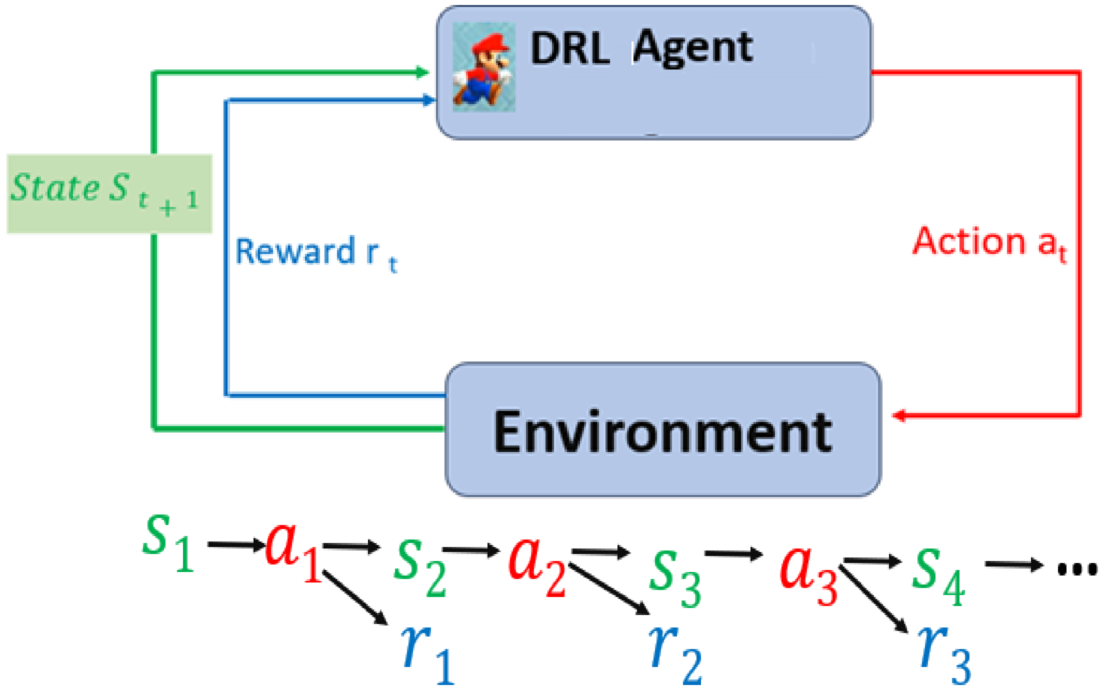
Current state

Current action

Probability of new state

Applications: water reservoir policy, environmental system, self driven cars, robotics, etc.

Types of Deep Reinforcement Learning


DRL Agent
State S_{t+1}
Reward r_t
Action a_t
Environment
$s_1 \rightarrow a_1 \rightarrow s_2 \rightarrow a_2 \rightarrow s_3 \rightarrow a_3 \rightarrow s_4 \rightarrow \dots$
r_1 r_2 r_3

- **DQN (Deep Q-Network)** uses a neural network to approximate Q-values, enabling it to learn optimal actions in discrete action spaces.
- **PPO (Proximal Policy Optimization)** is a policy gradient method that optimizes the policy directly, often used for continuous action spaces and known for its stability.
- …

Summary

- Some algorithms are well structured for regression, and some for classification—simple vs complex structures.
- The number of hidden layers in a neural network primarily determines its ability to learn complex, non-linear relationships in data.
- Among different NNs, Transformers are gaining momentum for time series and text analytics research.
- All these algorithms have their pros and cons, adapt them with care!
- We're hitting a tipping point with ChatGPT and many advanced algorithms for real world applications.

THANK YOU

The support from the National Science Foundation (award # 2320979) is acknowledged.