Which machine learning model to use?

State your problem and follow this article to know which model to use.

Maher

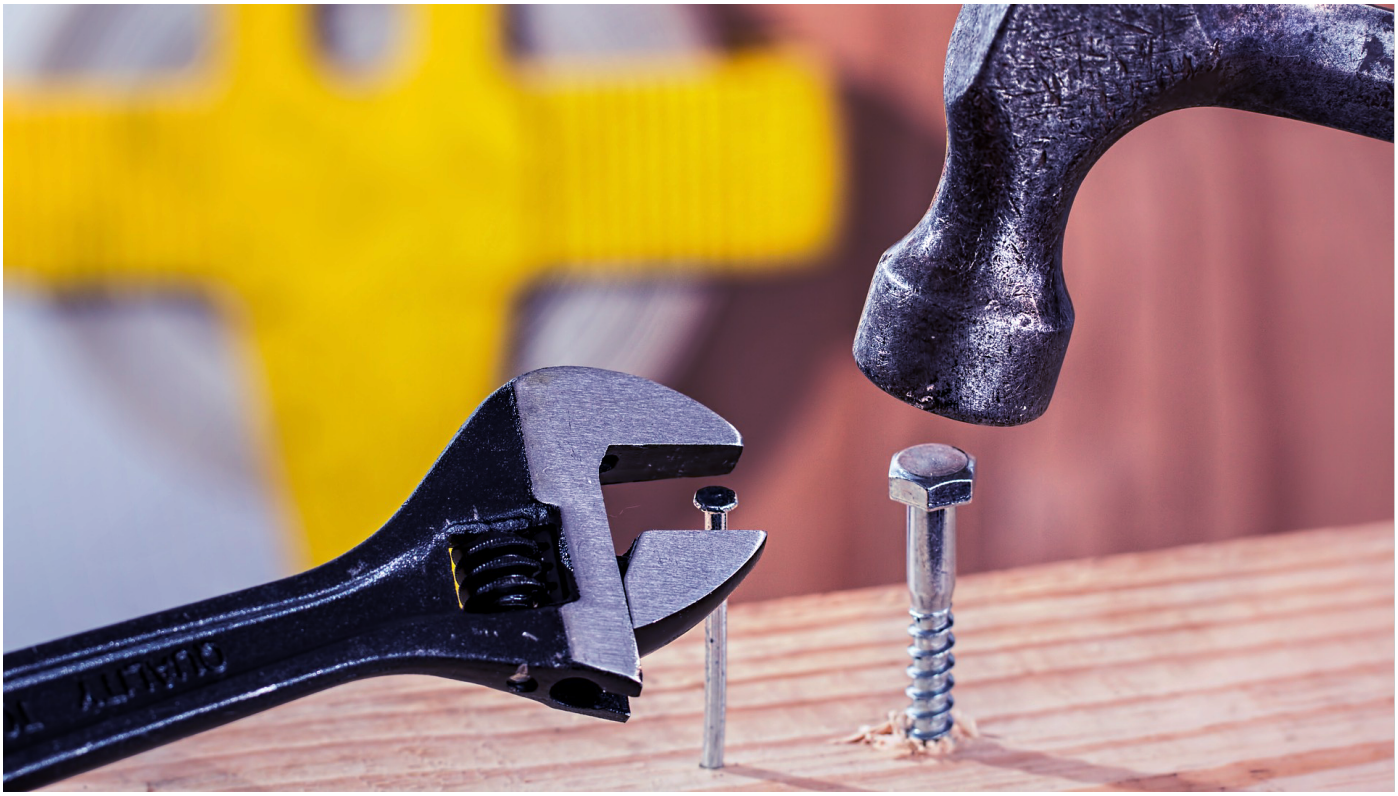
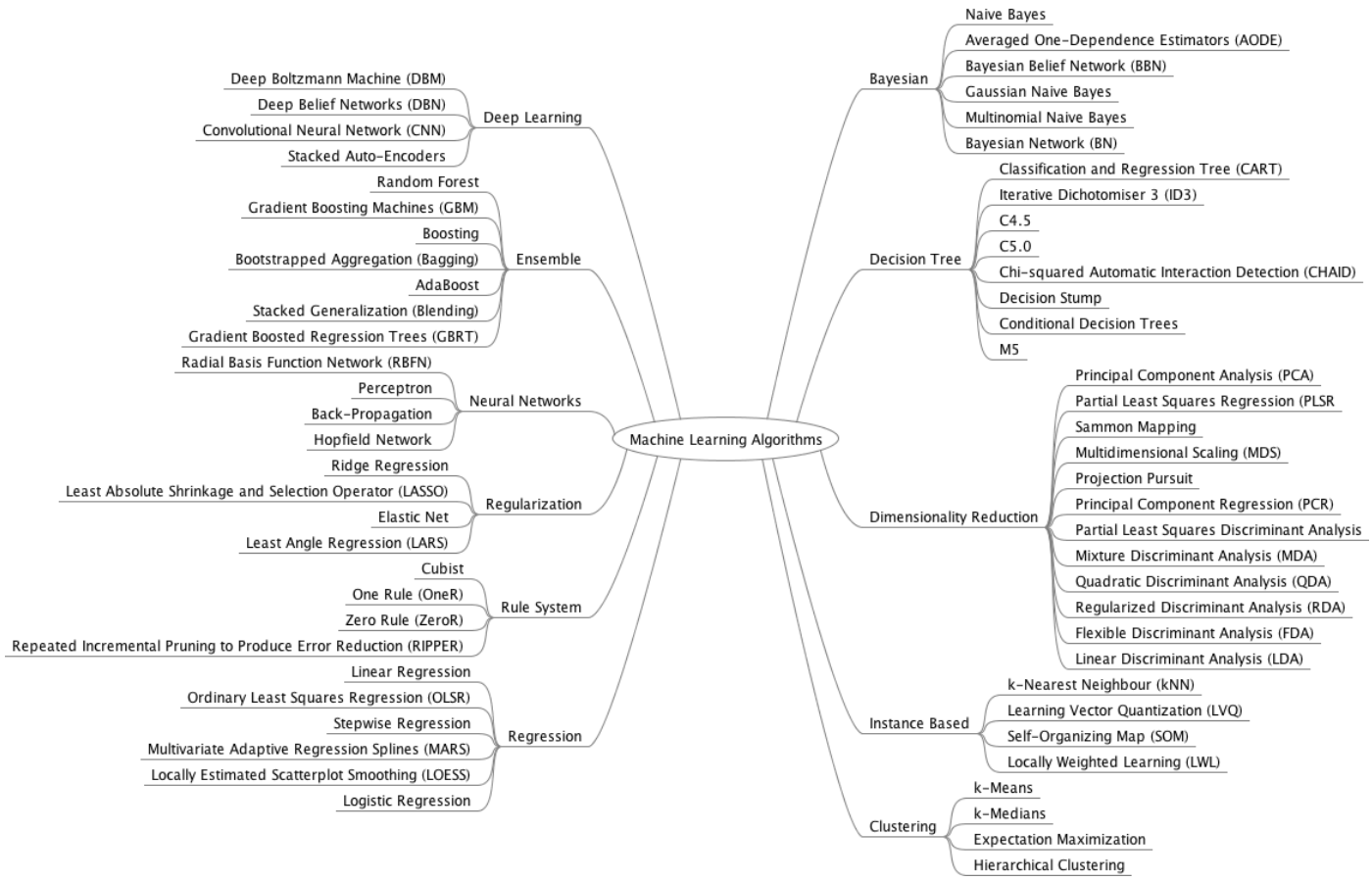


image by stevepb

— if you don't know what is an ML model, take a look at this article.



summary of ML models, Source

Taking machine learning courses and reading articles about it doesn't necessarily tell you which machine learning model to use. They just give you an intuition on how these models work which may leave you in the hassle of choosing the suitable model for your problem.

At the beginning of my journey with ML, on solving a problem, I would try many ML models and use what works best, and I still do that now but I follow some best practices — on how to choose a machine learning model — that I learned from experience, intuition and colleagues, these best practices make things easier, here is what I had collected.

I'll tell you which machine learning model to use according to the nature of your problem, and I'll try to explain some concepts.

Pexels

Classification

First, if you have a classification problem “which is predicting the class of a given input”.

Keep in mind how many classes you’ll classify your inputs to, as some of the classifiers don’t support multiclass prediction, they only support 2 class prediction.

- Slow but accurate

- Non-linear SVM Look at the **NOTE** at the end of classification’s section for more information about the use of SVM.

- Random Forest
- Neural Network (needs a lot of data points)
- Gradient Boosting Tree (similar to Random Forest, but easier to overfit)

- Fast

- Explainable models: **Decision Tree** *and* **Logistic Regression**
- Non-explainable Models: **Linear SVM** *and* **Naive Bayes**

NOTE: SVM kernel uses (From Andrew NG's course)

- Use the linear kernel when the number of features is larger than the number of observations.
- Use the Gaussian kernel when the number of observations is larger than the number of features.
- If the number of observations is larger than 50k, speed could be an issue when using the Gaussian kernel; hence, one might want to use the linear kernel.

Regression

If you have a regression problem "which is predicting a continuous value like predicting prices of a house given the features of the house like size, number of rooms, etc".

- Accurate but slow

- Random Forest
- Neural Network (needs a lot of data points)

- Gradient Boosting Tree (similar to Random Forest, but easier to overfit)

- Fast

- Decision Tree
- Linear Regression

Clustering

If you have a clustering problem "which is dividing the data into k groups according to their features such that objects in the same group have some degree of similarity".

Hierarchical clustering (also called **hierarchical cluster analysis** or **HCA**) is a method of cluster analysis which seeks to build a hierarchy of clusters. strategies for hierarchical clustering generally fall into two types:

- **Agglomerative**: This is a "bottom-up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- **Divisive**: This is a "top-down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

Nonhierarchical Clustering:

- DBSCAN (you don't need to specify the value of the k, which is the number of clusters)
- k-means
- Gaussian Mixture Model

In case you are clustering a **categorical data** use

- k-modes

Dimensionality reduction

Use The **Principal Component Analysis (PCA)**

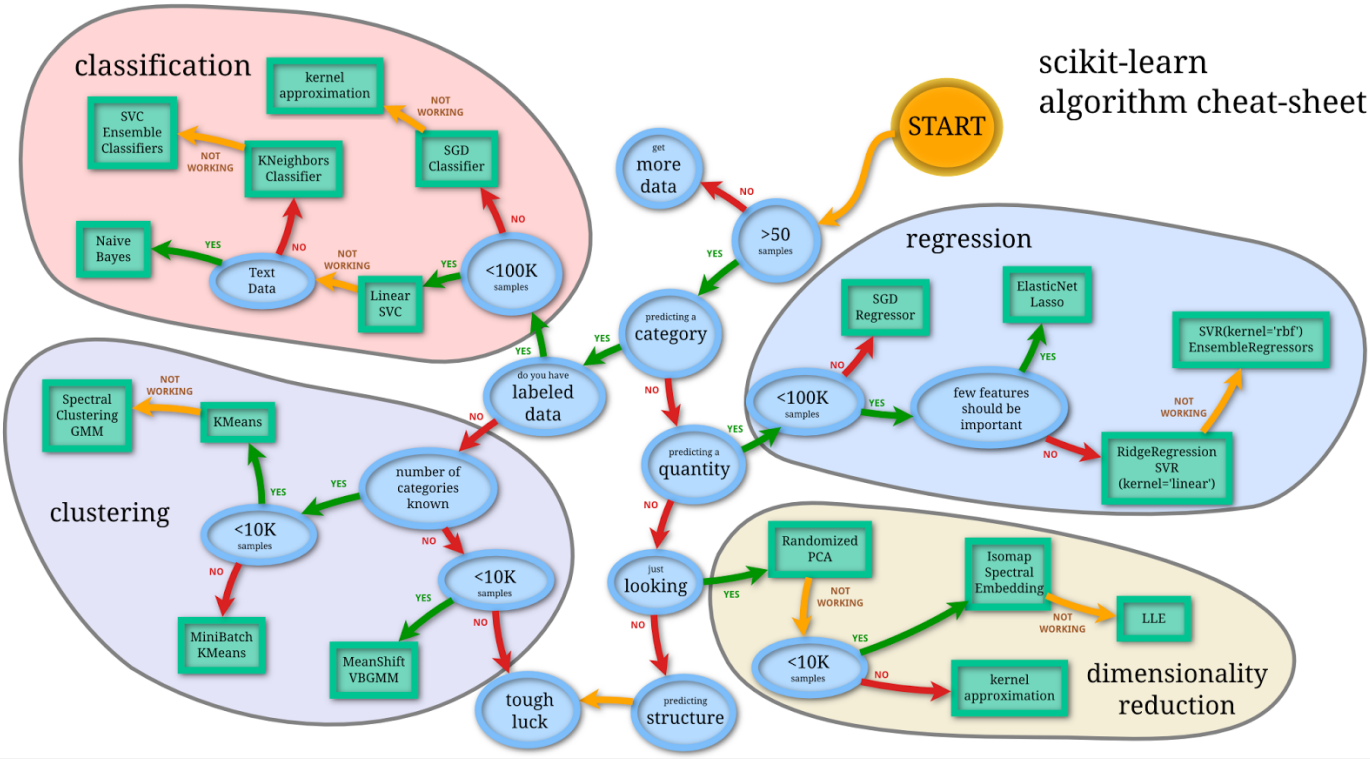
PCA can be thought of as fitting an *n*-dimensional ellipsoid to the data, where each axis of the ellipsoid represents a principal component. If some axis of the ellipsoid is small, then the variance along that axis is also small, and by omitting that axis and its corresponding principal component from our representation of the dataset, we lose only a commensurately small amount of information.

In case you want to make **topic modeling** (explanation below) you use **Singular Value Decomposition** (**SVD**) or **Latent Dirichlet Analysis** (**LDA**), and use **LDA** in case of probabilistic topic modeling.

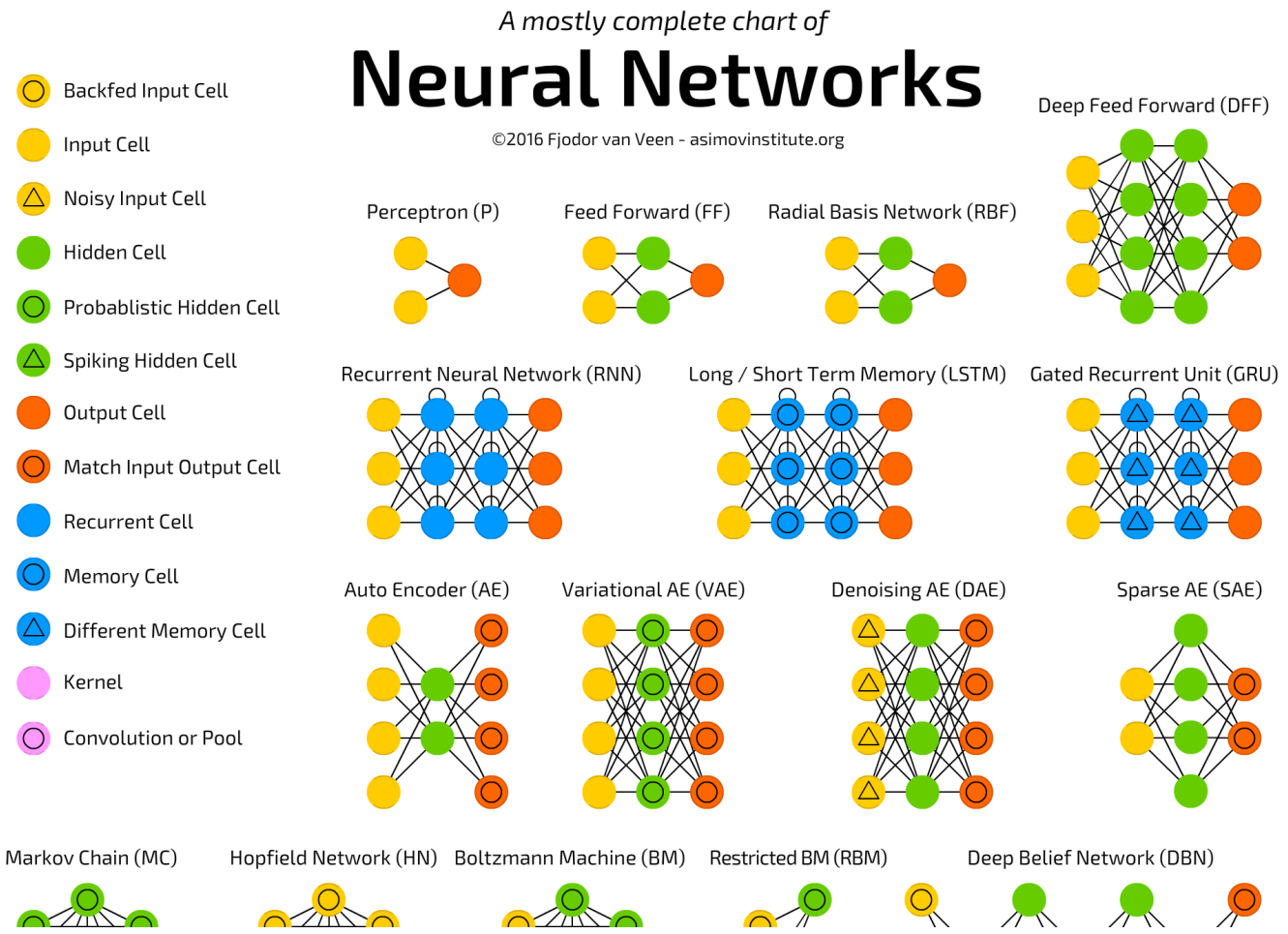
- **Topic modeling** is a type of statistical model for discovering the abstract "topics" that occur in a collection of documents. Topic modeling is a frequently used text-mining tool for the discovery of hidden semantic structures in a text body.

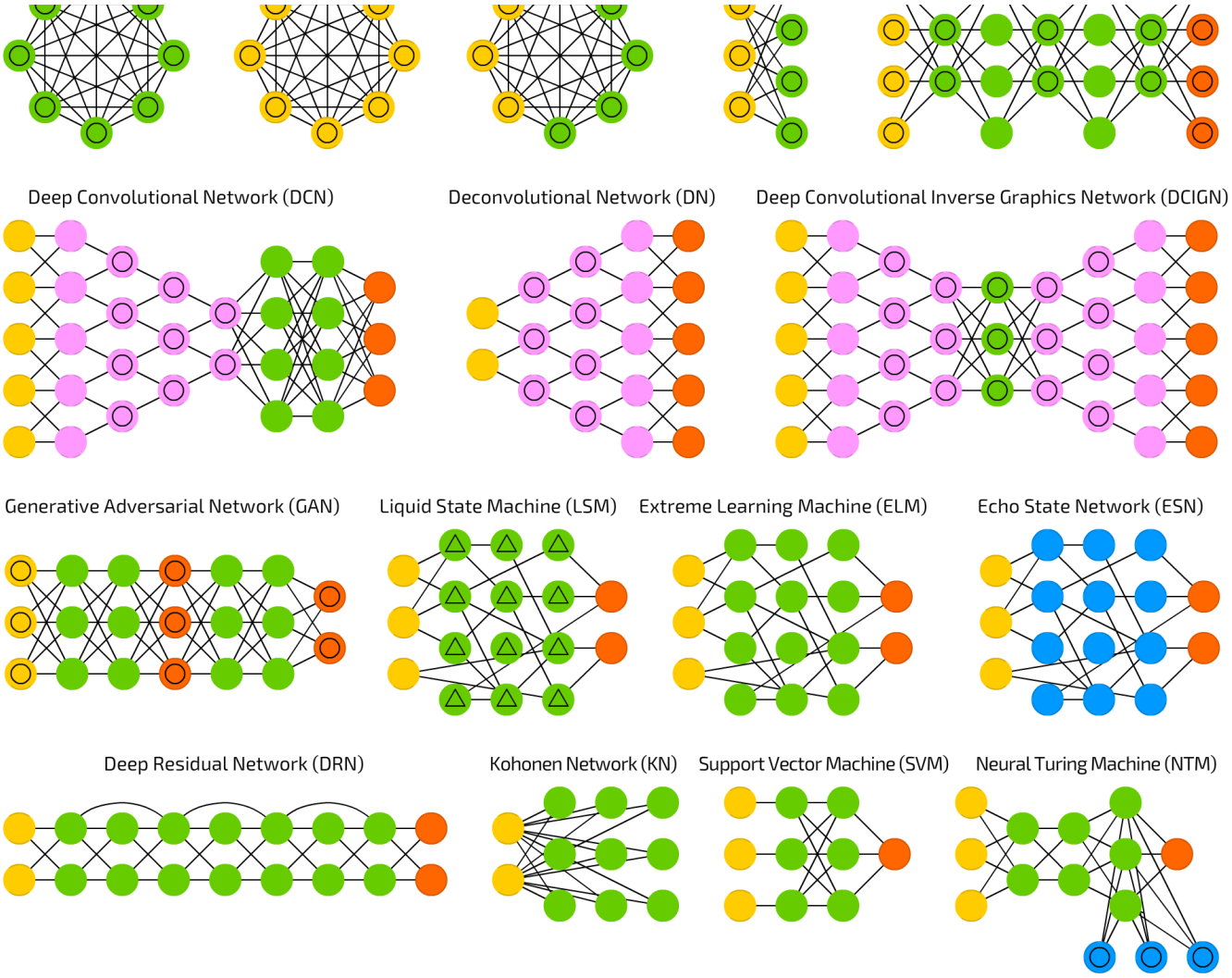
I hope things are easier for you now, I'll update the article with the information I get from your feedback and your experiments.

I'll leave you with these 2 awesome summaries.


scikit-learn
algorithm cheat-sheet
START
get more data
>50 samples
predicting a category
do you have labeled data
predicting a quantity
just looking
predicting structure
tough luck
classification
kernel approximation
SVC Ensemble Classifiers
KNeighbors Classifier
SGD Classifier
Naive Bayes
Text Data
Linear SVC
<100K samples
regression
SGD Regressor
ElasticNet Lasso
SVR(kernel='rbf') EnsembleRegressors
<100K samples
few features should be important
RidgeRegression SVR (kernel='linear')
clustering
Spectral Clustering GMM
KMeans
number of categories known
<10K samples
MiniBatch KMeans
<10K samples
MeanShift VBGMM
dimensionality reduction
Randomized PCA
Isomap Spectral Embedding
LLE
<10K samples
kernel approximation
YES
NO
NOT WORKING

source


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic 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)
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)

[Source]