TOP MACHINE LEARNING ARCHITECTURES EXPLAINED

Different Machine Learning architectures are needed for different purposes. A car is a motor vehicle that gets you to work and to do road trips, a tractor tugs a plough, an 18-wheeler transports lots of merchandise.

Each machine learning model is used for different purposes. One is used to classify images, one is good for predicting the next item in a sequence, and one is good for sorting data into groups. Some are good for multiple purposes, and some are good for just one.

In this article, we'll look at the most common ML architectures and their use cases, including:

- CNNs
- RNNs
- Sorting/Clustering
- GANs

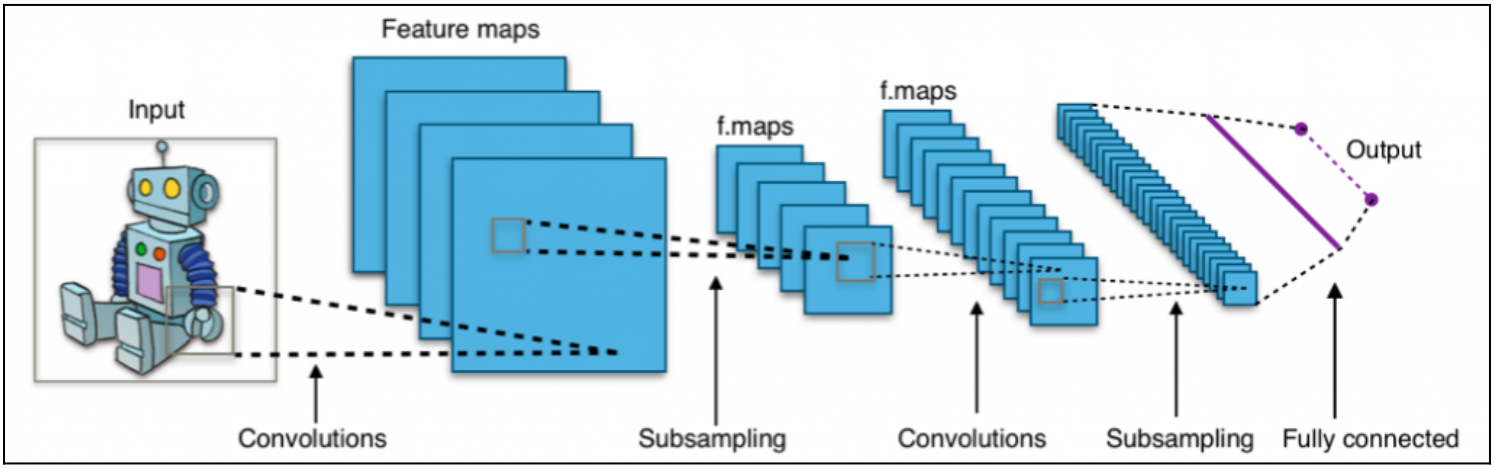
Convolutional neural networks (CNNs)

Used often in:

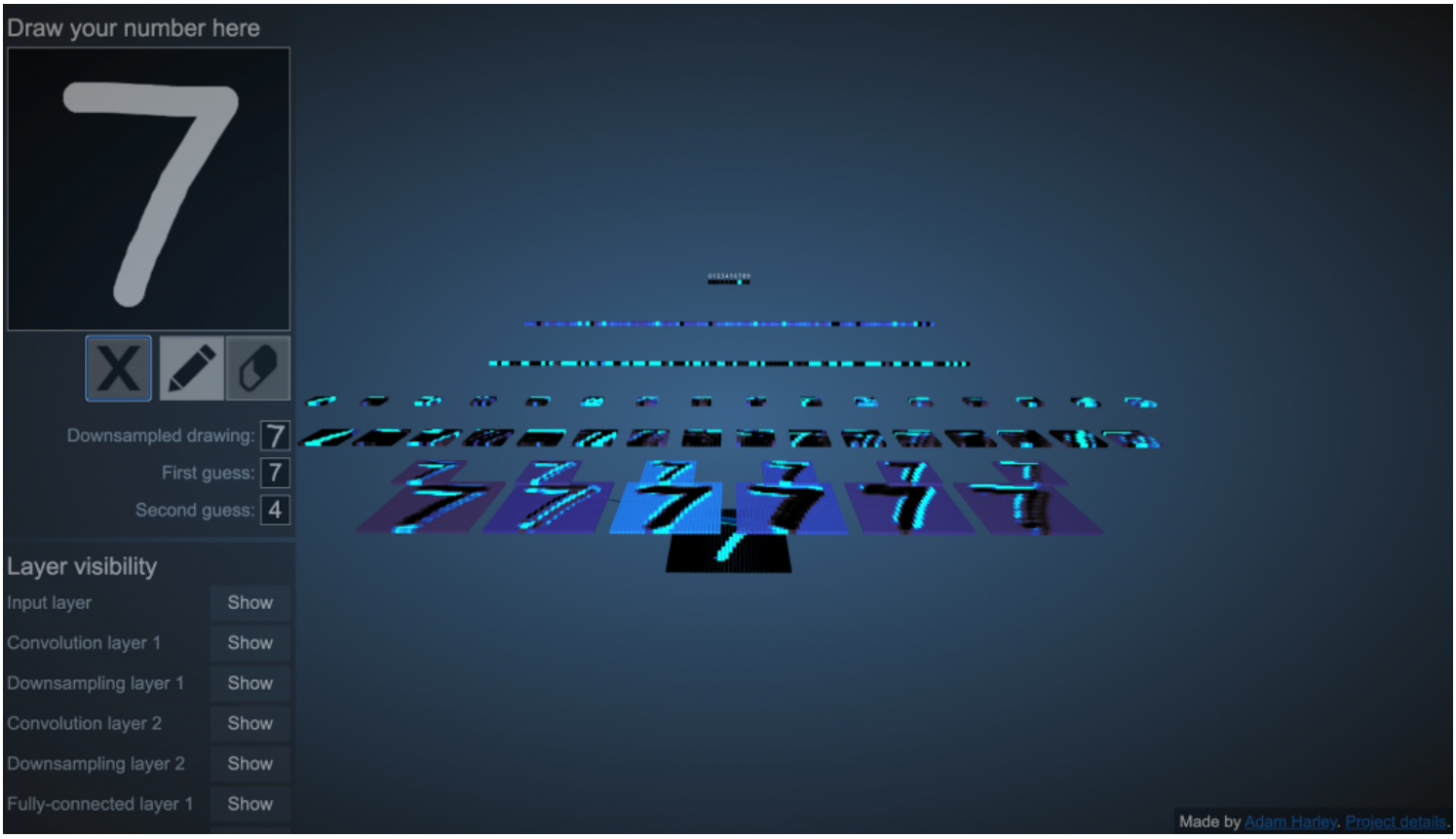
- Image classification
- Object detection
- Recommender systems

The essential component of the CNN is its convolutional layer—hence the name. The convolutional layer is a filter between the input and output. It creates a feature map of the inputs which summarizes the detected features. The convolutional layer can break an image down into important features, then predict its label based on those features.

After creating the feature map, the following layers are pooling layers. Pooling layers simplify the computation by reducing the dimensionality of the data. To do this, it combines the outputs of one layer before proceeding to the next layer. Pooling can happen locally or globally. The difference is whether the pooling happens in one convolutional neuron or across all the convolutional neurons.

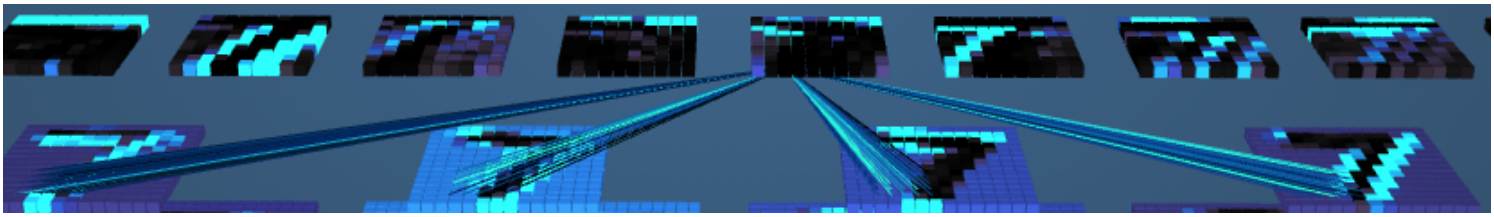


Example of pooling from feature maps



Feature map visualization (Source: Adam W. Harley)

In this interactive visualization, shown above, you can see how the first layer is 6 feature maps of the input layer (found at the bottom of the stack). These 6 feature maps are then downsampled, where they are reduced from 25x25 images to 14x14-sized images while their major shape stays intact. Again, a feature layer is created from these new images consisting of 16 feature maps.



One pixel is pooled from sections of four features

Finally, after looking at major features of the drawn image, the CNN puts the inputs through two fully connected layers and predicts what the label should be.

Recurrent/recursive neural networks (RNNs)

Used for:

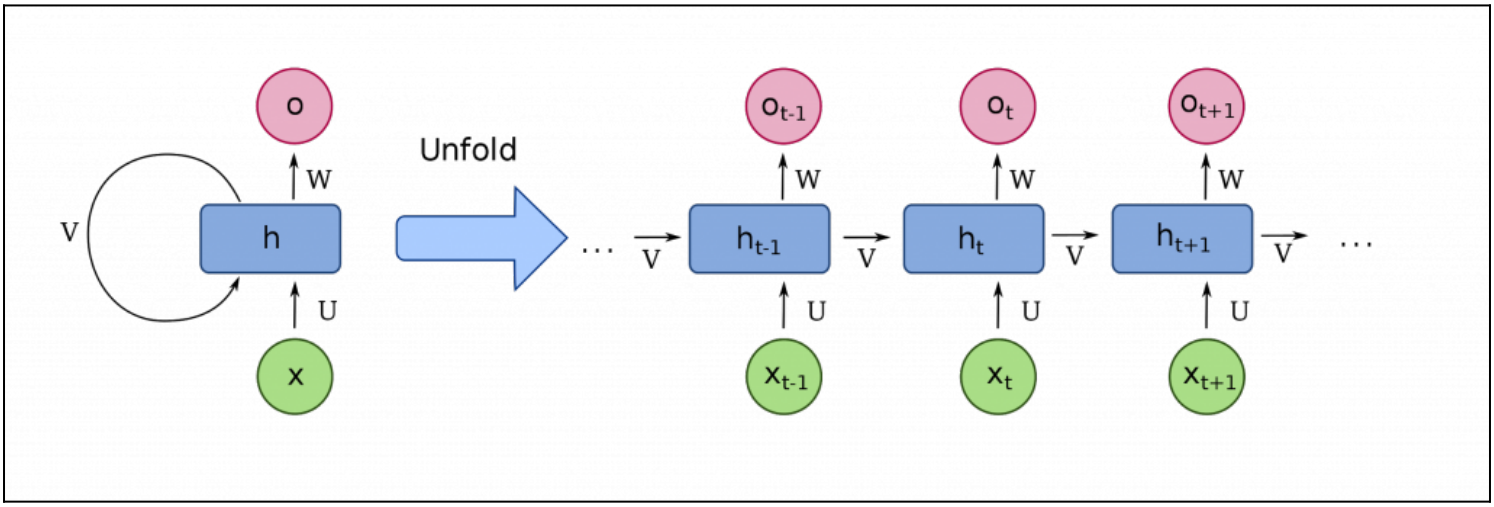
- Sequence modelling
- Next word prediction
- Translating sounds to words
- Human language translation

Recurrent neural networks are a basic architecture with many variations. An RNN is used largely to solve sequence-to-sequence problems where the input is a sequence and the output is another sequence. These problems typically include:

- Language translation (English to Chinese)
- Generation (predicting the next item in a sequence)

There are many variants. Of these, the LSTM and the Transformer tend to be the most important. The latter of which is responsible for the OpenAI GPT-3 hype. The LSTM and Transformer are both ways to add and control memory to the model. Here's how they work: The RNN is a sequence of nodes that can be represented as a directed graph. As information passes through each node, the node learns to predict what comes next.

- **The LSTM** is an addition to the classical RNN that enables the RNN to have some control over its memory, defining data that occurred more recently to be more important and data that occurred further in the past as being less important.
- **The transformer** is another addition which also grants the RNN more memory control by not only mapping distances in the past as important but creating a population distribution of past events and stating, of those, which is most important to the present.



Architecture of a basic RNN (Source)

Sorting/Clustering Architectures

Used for:

- Anomaly detection
- Pattern recognition

Sorting and clustering algorithms are used to look at the distribution of a population, and possibly discover something unknown in the data. Thus, they're great models to use in unsupervised modelling scenarios and on unstructured data. The models can be used to:

- **Recognize patterns** and place a population in classes, like groups of people who like country, rock, or hip-hop.
- **Detection anomalies** by identifying outliers in a data set, like individuals who like none of those genres.

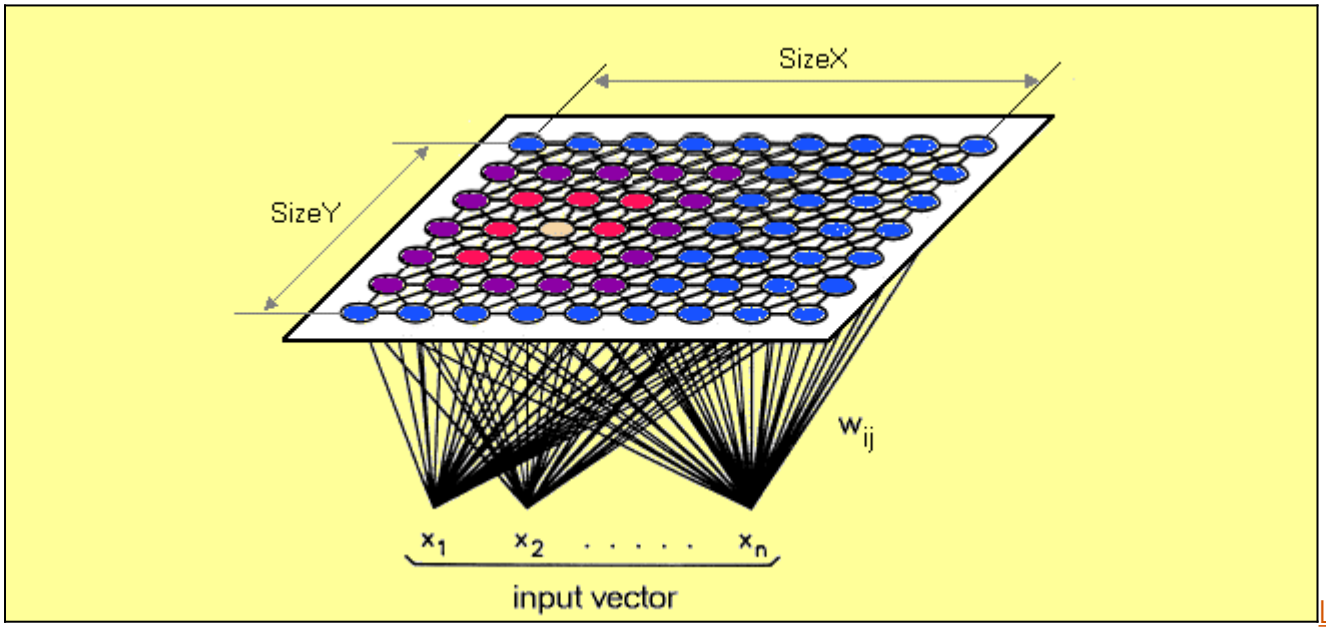
The Self-Organizing Map is a common tool in RNNs. The SOM has two layers, an input and an output. The output layer of the self-organizing map is a feature map. Very similar to the first layer of the CNN: you can see the way the feature map is formed can vary greatly model to model. But the output layer remains an essential step for transforming data points into something meaningful to work with in an ML architecture.

The output of data could look similar to the graphic below, where:

- Each blue node (dot) could be people with low credit scores.
- The khaki dot in the middle could a person with a 800+ credit score.

The self-organizing map would not know those features ahead of time, of course. When it creates the map, the algorithm will compare the data inputs side by side, then:

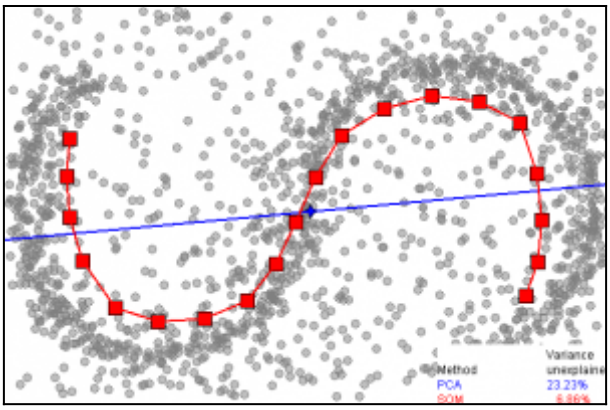
- If they are similar, it keeps them close.
- If they are not similar, it pushes the two farther apart.



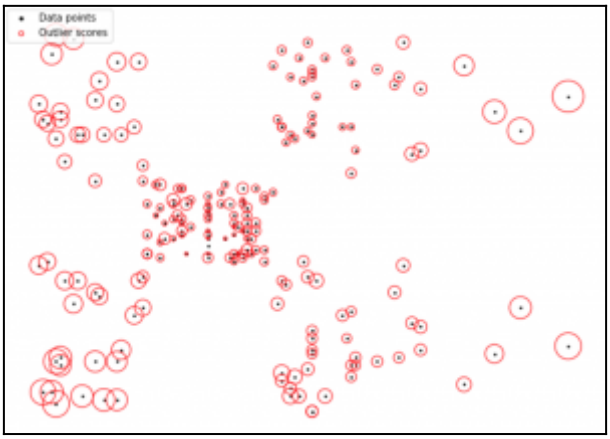
Language models usually use a similar sorting method to transform meaningless words into a feature map that can be used for computation.

Popular ML models used on unstructured data are:

- Self-organizing maps (SOM)
- K-means
- C-means
- Expectation-maximization meta-algorithm (EM)
- Adaptive resonance theory (ART)
- One-class support vector machine



SOM detection (Source)



K-means clustering (Source)

Generative models

Used for:

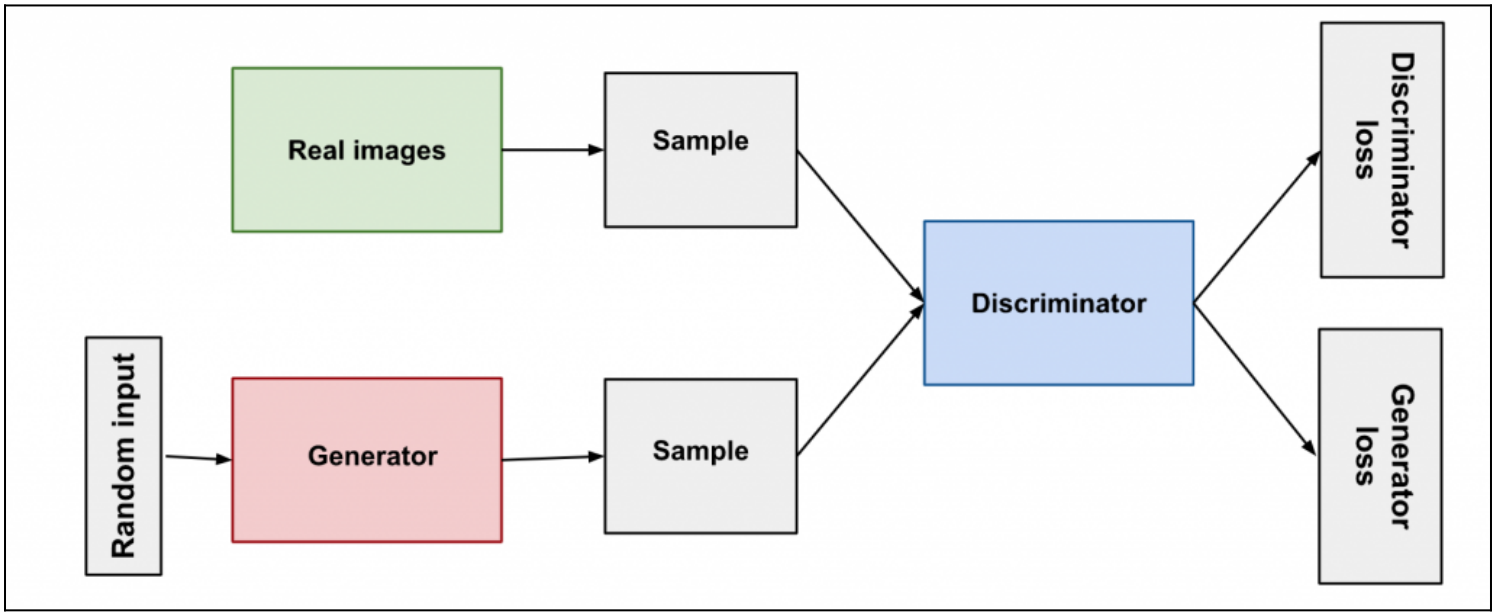
- Anomaly detection
- Pattern recognition
- Cyber-security
- Self-driving cars
- Reinforced learning

Generative models are a breed of ML model meant to generate data identical to the samples. They are used to say how likely a given example is. They can create faces that look like celebrities. They are used in cyber-security to say how likely an image might be a deep fake, or how likely an IP address might be an attacker. GANs can spot fakes.

In this architecture, an ML model trains by generating a random piece of data and testing it with a real piece of data:

- If a discriminator passes it off as real, it works; if not, it goes back and tries again.
- When a fake data point passes, the discriminator can get better at its job and it gets updated with the generated dataset of fakes.

The generator and discriminator play a game with one another to outperform the other.



Overview of GAN structure (Source)

For simulation games, the generator is similar to Tom Cruise in Edge of Tomorrow, who spawns (a generation) and tries to make it to the end to beat the war. Or, like Neo in The Matrix, who is a generated data point who always finds his way to the Great Maker, and when he does so, the system fails and has to start over.

Popular generative ML models are:

- Generative Adversarial Networks (GANs)

- Boltzmann Machines
- Hidden Markov Model
- Variational Autoencoder

Machine learning models vs architectures

Models and architecture aren't the same. Remember that your machine learning architecture is the bigger piece. Think of it as your overall approach to the problem you need to solve. The architecture provides the working parameters—such as the number, size, and type of layers in a neural network.

Models are one piece of your architecture; a specific instance that trains on a chosen set of data. For example, in a neural net, the trained weights of each node, per the architecture, comprise the model.

Additional resources

For more on this topic, explore these resources:

- BMC Machine Learning & Big Data Blog
- What's a Deep Neural Network? Deep Nets Explained
- Top Machine Learning Frameworks To Use
- Guide to ML with Keras & TensorFlow
- scikit-learn Guide
- Containerized Machine Learning: An Intro to ML in Containers
- How To Run Machine Learning Transforms in AWS Glue