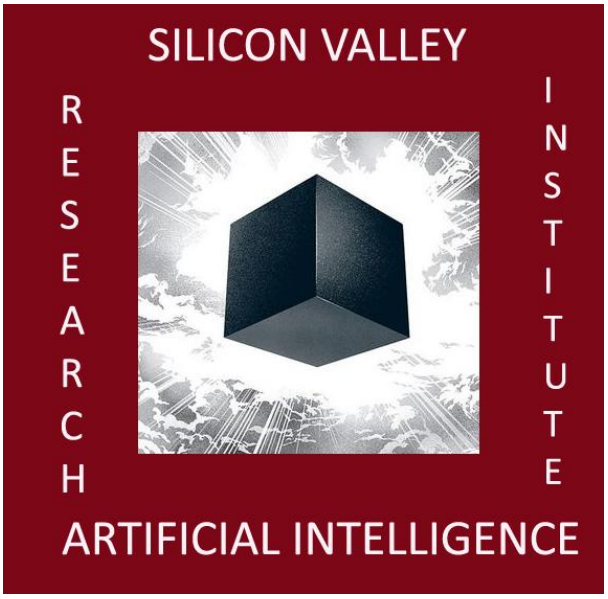


# Recurrent Neural Networks: a brief introduction

(Copyright © 2018 Piero Scaruffi - Silicon Valley Artificial Intelligence Research Institute)

Neural networks, up to this point, were good for recognizing patterns (e.g., that this particular object is an apple) but not for learning events in time. Neural networks, in principle, have no sense of time. Recurrent neural networks (and, in general, connectionist models) have a rudimentary sense of persistence, of "memory". The question soon arose of how to represent time in connectionist models, in particular related to natural language processing. Traditional approaches to representing time basically used a spatial metaphor for time: create a representation in space of the sequence of events. The difficulty in using neural networks for analyzing language is that the connectionist representation of a neural network has none of the compositionality of syntax. However, while the connectionist representation is not syntactically compositional, it can still be functionally compositional.

Michael Jordan extended recurrent neural networks to analyses of sequences by feeding the output into a "context" layer which is then fed into the middle layer; i.e. the network is fed with the input of a step and with the output of the previous step ("Attractor Dynamics and Parallelism in a Connectionist Sequential Machine", 1986). Following in his footsteps, Jeffrey Elman at UC San Diego invented what he called a "simple recurrent neural network" or SRNN, which is actually another sophisticated model for processing sequences and not just patterns ("Finding Structure in Time", 1990). If recurrent neural networks can operate on sequences, it means that they can model relationships in time, something crucial for processing natural language.

Hava Siegelmann of Bar-Ilan University in Israel and Eduardo Sontag of Rutgers University proved an encouraging theorem: that recurrent neural networks can implement Turing machines if the weights of the connections are rational numbers; and that recurrent neural networks are even more powerful than a universal Turing machine if the weights are real numbers, i.e. if the

network is analogic ("On the Computational Power of Neural Networks", written in 1992 but published only in 1995)

Kenichi Funahashi and Yuichi Nakamura at Toyohashi University of Technology in Japan proved that recurrent neural networks are universal approximators, i.e. they can approximate any dynamical system ("Approximation of Dynamical Systems by Continuous Time Recurrent Neural Networks", 1993). Christian Omlin and Lee Giles of NEC Research Institute in Princeton proved that recurrent neural networks can approximate arbitrary finite state machines ("Constructing Deterministic Finite-State Automata In Sparse Recurrent Neural Networks", 1994).

Typical applications of RNNs are: image captioning, that turns an image into a sequence of words ("sequence output"); sentence classification, that turns a sequence of words into a category ("sequence input"); and sentence translation (sequence input and sequence output). The innovation in RNNs is a hidden layer that connects two points in time. In the traditional feed-forward structure, each layer of a neural network feeds into the next layer. In RNNs there is a hidden layer that feeds not only into the next layer but also into itself at the next time step. This recursion or cycle adds a model of time to traditional backpropagation, and is therefore known as "backpropagation through time".

Neural networks were being designed to solve one specific problem at a time, but in 1990 Robert Jacobs at the University of Massachusetts introduced the "mixture-of-experts" architecture that trains different neural networks simultaneously and let them compete to learn, with the result that different networks end up learning different functions ("Task Decomposition Through Competition in a Modular Connectionist Architecture", 1990). Similarly, Tom Mitchell's student Rich Caruana at Carnegie Mellon University showed that multiple neural networks learning tasks in parallel could benefit from sharing what they learned ("Multitask Learning, a Knowledge-based Source of Inductive Bias", 1993). This would also be shown by Ronan Collobert's and Jason Weston's unified architecture for natural language processing of 2008.

Another elusive goal of machine learning was to equip machines with "transfer learning", i.e. the ability to transfer what they learn on a task to another task, and therefore "multitaks learning", the ability to learn more than just one task. In 1991 Satinder Singh of the University of Massachusetts published "Transfer of Learning by Composing Solutions of Elemental Sequential Tasks" and Lorien Pratt of the Colorado School of Mines published "Direct Transfer Of Learned Information Among Neural Networks". By then, Tom Mitchell's group at Carnegie Mellon University had become the world's center of excellence in transfer learning and multitask learning, as documented by Sebastian Thrun's "Learning One More Thing" (1994) and Rich Caruana's "Multitask Learning" (1997). But that school seemed to reach an unsurmountable peak just when Sebastian Thrun and Lorien Pratt curated the book "Learning to Learn" (1998).

A general problem of neural networks with many layers ("deep" neural networks), and of RNNs in particular, is the "vanishing gradient", already described in 1991 by Josef "Sepp" Hochreiter at the Technical University of Munich ("Investigations on Dynamic Neural Networks", 1991) and more famously in 1994 by Yoshua Bengio ("Learning Long-Term

Dependencies with Gradient Descent is Difficult"). The expression "vanishing gradient" refers to the fact that the computations for each new layer become less and less clear. It is a problem similar to calculating the probability of a chain of events: if you multiply a probability between 0 and 1 by another probability between 0 and 1 and so on many times, the result is always zero, even in the case in which all those numbers expressed probabilities of 99%. A network with many layers is difficult to train because the "weights" of the last layer end up being too weak.

New unsupervised learning algorithms included imax by Geoffrey Hinton's student Suzanna Becker at the University of Toronto ("A self-organising neural network that discovers surfaces in random dot stereograms", 1992) and Geoffrey Hinton's "wake-sleep" algorithm ("The Wake-sleep Algorithm for Unsupervised Neural Networks", 1995). Ralph Linsker's infomax was further simplified by Anthony Bell in Terrence Sejnowski's lab at the Salk Institute improved infomax ("A Non-Linear Information Maximization Algorithm that Performs Blind Separation", 1995). Then Shunichi Amari at Riken in Japan realized that the infomax algorithm could be improved by using the so-called "natural gradient", a gradient that considers the old and the new states of a neural network as two distributions of probabilities, and then uses the so-called Kullback-Leibler divergence to measure the distance between these two distributions ("Neural Learning in Structured Parameter Spaces", 1996, later expanded in "Natural Gradient Works Efficiently in Learning, 1998). This was the rare application of differential geometry to statistics because it deals with Riemannian geometry instead of Euclidean geometry, just like Albert Einstein's theory of general relativity.

Meanwhile in 1996 David Field and Bruno Olshausen at Cornell University had invented "sparse coding", an unsupervised technique for neural networks to learn the patterns inherent in a dataset. Sparse coding helps neural networks represent data in an efficient way that can be used by other neural networks. Autoencoders have a propensity to diverge but sparse autoencoders fixed that problem. The idea came from the very way in which the primary visual cortex works. When a stimulus (e.g. a sound or an image) causes the activation of only a small number of neurons, this pattern of activation represents a "sparse coding" of the stimulus, which turns out to be an efficient way to represent stimuli: minimal computation and still the ability to reconstruct the input image. Autoencoders trained with a sparsity constraint exhibit similar benefits: they are simple and allow stacking multiple layers. Their vogue began with successful implementations by Andrew Ng's group at Stanford ("Self-taught Learning - Transfer Learning from Unlabeled Data", 2007) and by LeCun's group at New York University ("Fast Inference in Sparse Coding Algorithms with Applications to Object Recognition", 2008).

Machine learning was doing just about fine without any need to model the time dimension: support vector machines, logistic regression, and traditional as well as convolutional neural networks can recognize patterns. There are, however, applications that require a sequential analysis, i.e. that need to model the time dimension. Examples are speech recognition, image captioning, language translation, and handwriting recognition. Hidden Markov models have been used to model time but they become computationally impractical for modeling long-range dependencies (when you have to remember not just a few earlier steps but many earlier steps).

In 1997 Sepp Hochreiter and his professor Juergen Schmidhuber (by now director of the Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, or IDSIA, in Switzerland) came up with

a solution: the Long Short Term Memory (LSTM) model. In this model, the unit of the neural network (the "neuron") is replaced by one or more memory cells. Each cell functions like a mini-Turing machine, performing simple operations of read, write, store and erase that are triggered by simple events. The big difference with Turing machines is that these are not binary decisions but "analog" decisions, represented by real numbers between 0 and 1, not just 0 and 1. For example, if the network is analyzing a text, a unit can store the information contained in a paragraph and apply this information to a subsequent paragraph. The reasoning behind the LSTM model is that a recurrent neural network contains two kinds of memory: there is a short-term memory about recent activity and there is a long-term memory which is the traditional "weights" of the connections that change based on this recent activity. The weights change very slowly as the network is being trained. The LSTM model tries to retain also information contained in the recent activity, that traditional network only use to fine-tune the weights and then discard.

LSTM recurrent neural nets represented a significant quantum leap in neural networks because they capture temporal dependencies, such as those necessary to understand video, text, speech and movement.

LSTMs were later improved for tasks such as handwriting recognition and speech recognition by Schmidhuber's student Alex Graves, who developed "connectionist temporal classification" or CTC ("Connectionist Temporal Classification", 2006).

At this point one is tempted to rewrite the history of neural networks as rediscovering the elements of programming languages. First there was the simple McCulloch-Pitts neuron, a computational unit that produces a sum of products, coupled with an activation function (such as the sigmoidal) that plays the role of the “IF” statement of programming languages. Hopfield’s recurrent networks introduced the equivalent of the “loop”. Finally, LSTMs added the equivalent of the variable of a programming language, a way to store a value.

LeCun's convolutional nets had solved the problem of how to train deep feedforward neural nets: Long Short Term Memory solved the problem of how to train deep recurrent neural nets.

At the same time that LSTM were introduced, in 1997 Mike Schuster and Kuldip Paliwal at the Advanced Telecommunications Research Institute in Japan discovered that "bidirectional recurrent neural networks" can achieve a similar feat of persistence.

In the 2010s several other kinds of neural networks coupled with some long-term memories, similar to LSTMs, have been proposed: "Neural Turing Machines" (2014, by Alex Graves at Google's DeepMind in Britain) and "Memory Networks" (2014, by James Weston at Facebook's labs in New York). Just like LSTMs, these are neural networks that can perform complex inference. This seems to be a progressive shift towards a form of hybrid computing that may reconcile neural networks and the old knowledge-based school.

(Copyright © 2018 Piero Scaruffi - Silicon Valley Artificial Intelligence Research Institute)