Neural Networks

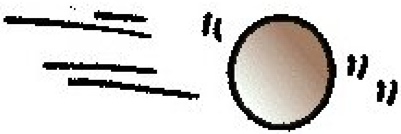
Recurrent and Deep Neural Networks

Dr. Petr Musilek

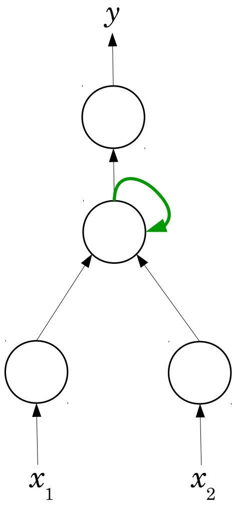
Department of Electrical and Computer Engineering
University of Alberta

Fall 2019

Dynamic Networks



Networks with Feedback



The difference this added connection makes:

- activation is now given by
  - current input pattern
  - and previous state of the unit
- States of hidden and output neurons are now functions of everything the network has seen so far (i.e. it has a “sense of history”)

RNN Topology

Adding feedback connections – topology becomes very free (any unit to any unit)

Two basic assumptions (so far) are violated

- for computing activations - knowing activations of all posterior units
- for computing errors – knowing error of all anterior units

Need a new approach for training and operation

FFNNs vs. RNNs

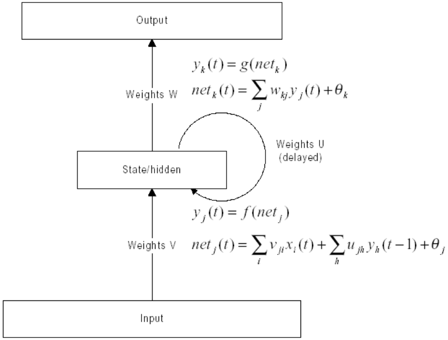
- Feed-Forward Neural Networks (FFNN) can *learn function mappings* (FIR filter)
- Recurrent Networks (RNN) use hidden layer as memory store to *learn sequences* (IIR filter) => RNNs can exhibit virtually unlimited temporal dynamics

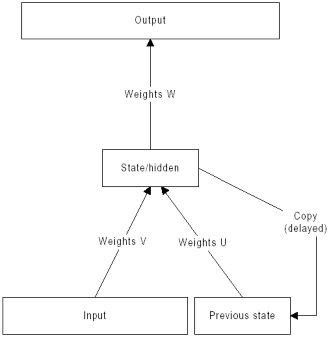
RNN Applications

- Grammar induction (recognition of legal strings of symbols)
- Speech recognition (RNN predicts phonemes, predictions fed into HMM for decoding)
- Filtering
- System Control, Identification
- Music Composition

Simple Recurrent NN (Elman, 1990)

Hidden layer activations copied into a context/copy layer


Output
$y_k(t) = g(net_k)$
$net_k(t) = \sum_j w_{kj} y_j(t) + \theta_k$
Weights W
Weights U (delayed)
State/hidden
$y_j(t) = f(net_j)$
$net_j(t) = \sum_i v_{ji} x_i(t) + \sum_h u_{jh} y_h(t-1) + \theta_j$
Weights V
Input



Elman Network

Operation:

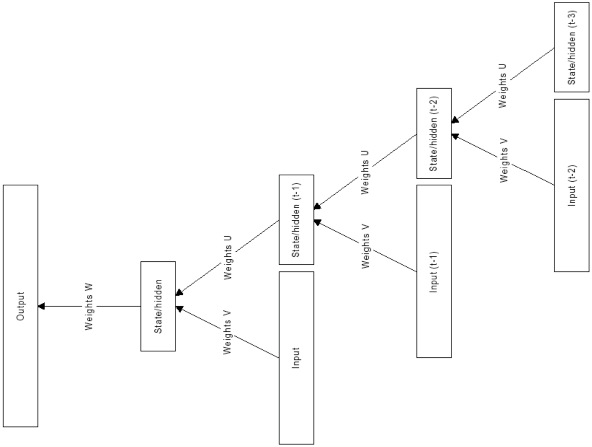
- Copy inputs for time t to the input units
- Compute hidden activations using net input from input units and from copy layer
- Copy new hidden unit activations to copy layer
- Compute output unit activations as usual
- Cycles are eliminated: activations and errors are available according to the original requirements (as for FFNN)

Training:

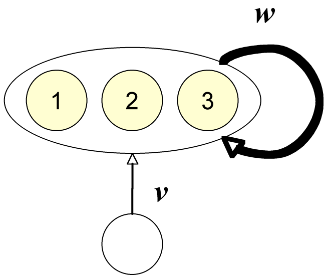
- Need to consider ∇E determined by the activations at the *present* and the *previous* time steps
- Otherwise, standard backpropagation algorithm is retained

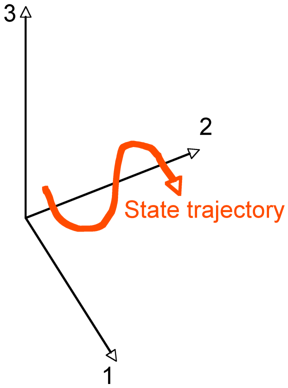
Back-Propagation Through Time (BPTT)

Considering more time steps (practical limit 10-12)



Dynamics of RNN





Long-Term and Short-Term Memory

RNN stores information in two different ways:

- Activations (states) – short term
- Weights – long term

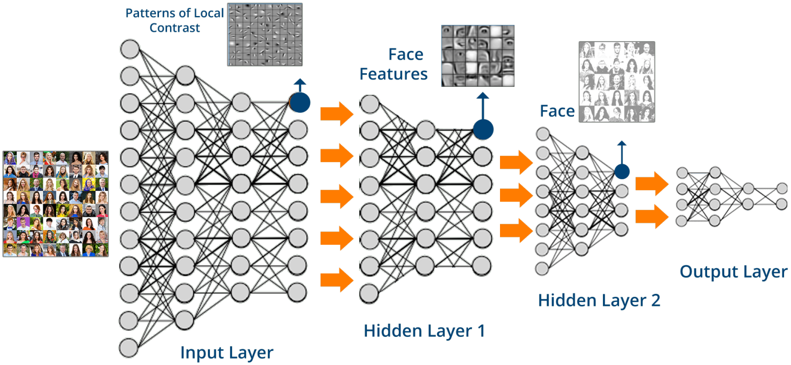
. . . each has a different time scale.

LSTM network - attempt to retain information in unit activations (states) for longer period of time (LSTM application for music composition is described at `http://www.iro.umontreal.ca/~eckdoug/blues/index.html`)

LSTM Memory Block

$$
\begin{aligned}
y_j^c(t) &= y_j^{out}(t) h(s_{c_j}(t)) \\
s_{c_j}(0) &= 0 \\
s_{c_j}(t) &= s_{c_j}(t-1) + y_j^{in}(t) g(net_{c_j}(t))
\end{aligned}
$$

Deep Neural Networks



Introduction

- In short, a Deep Neural Network (DNN) is a Multi-Layer Perceptron (MLP) with many hidden layers.
- However the field of DNN has developments on its own to improve processing and learning abilities.

Neural Networks and Deep Learning by Michael Nielsen

John Smart on Medium

Early DNNs

Relied on the back propagation algorithm to update each neuron weight individually.

Problems:

- Vanishing gradient error,
- Costly computationally and memory-wise,
- Slow learning,
- And required a large number of training data samples.

Early DNNs - solutions

Solutions to vanishing gradient error:

- Amplify error for each layer by a factor,
- Layer-by-layer learning,
- Use of Rectified Linear Unit in the hidden layers.

Amplification speeds up learning, but the amplification constant is selected manually and its optimal value is unknown. Learning layer by layer also speeds up learning to some extend, however, stopping learning for each layer and adding new layers is done *manually* and there is no target for the hidden layers.

Solutions for other problems:

- None.

Convolutional Neural Networks (CNNs)

CNN use a shared weight distribution for each layer section, where each section learns a specific pattern/feature of the previous layer. Updates for each section are done based on the overall section error.

- Significantly reduce the memory and computational costs,
- and further accelerates learning.

Additionally, it becomes largely immune to:

- Linear translations,
- and uniform scaling.

Artificial Inteligence by Leonardo Araujo Santos

However, CNNs are still sensitive to rotations, and non-linear scaling and distortions.

Pooling

A lossy compression (subsampling) commonly executed by either averaging the pooled values or by taking the maximum value within the pool. This step is done between convolutional layers (*i.e.* I, C, P, C, P, ..., C, P, O).

Why Reduce the amount of information?

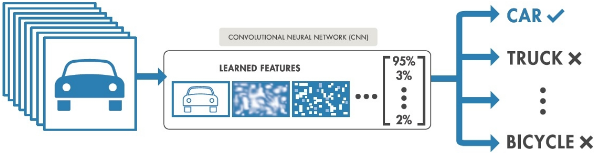
- Increase immunity to non-linear translations and distortions,
- and reduce the likelihood of over-fitting.

How is this different from a convolutional layer with $M \times m < n$?

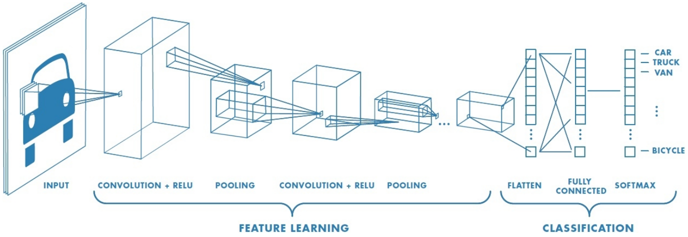
- No learning takes place,
- And the format of each Convolutional layer tends to be identical with some overlapping sources for each neuron. Pooling layers tend to also be identical to one another but do not have overlapping sources for each output.

Deep learning workflow

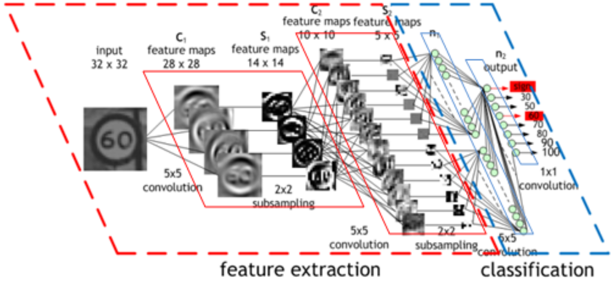
Images are passed to the CNN, which automatically learns features and classifies objects.



https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html



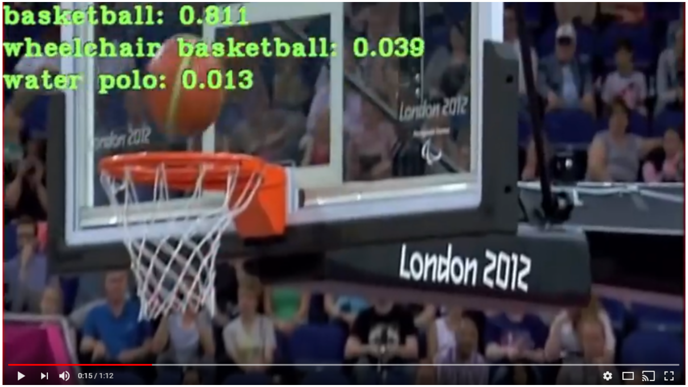
CNN for traffic sign classification



Deep Learning in a Nutshell: Core Concepts by NVIDIA

An image of a traffic sign is filtered by 4 5x5 convolutional kernels to create 4 feature maps subsampled by max pooling. The next layer applies 10 5x5 convolutional kernels to these subsampled images and again we pool the feature maps. The final layer is a fully connected layer where all generated features are combined and used in the classifier.

Example: Video Classification with CNNs



Large-scale Video Classification with Convolutional Neural Networks, CVPR 2014