Deep Learning II – Recurrent Neural Network

Dr Paul Yoo

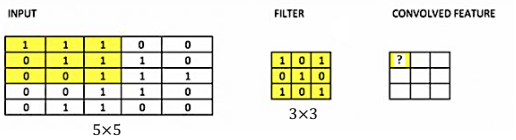
CSIS

Date

1

Quiz

What value would be in place of question mark?



Here we see a convolutional function being applied to input.

A. 3
B. 4
C. 5
D. 6

2

Quiz

Which of the following gives non-linearity to a CNN?

A. Stochastic Gradient Descent
B. Rectified Linear Unit
C. Convolution function
D. None of the above

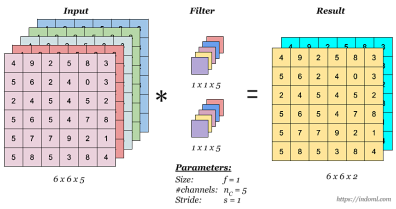
Rectified Linear unit is a non-linear activation function.

3

Quiz

Which of the following statements is true when you use 1×1 convolutions in a CNN?

A. It can help in dimensionality reduction
B. It can be used for feature pooling
C. It suffers less overfitting
D. All of the above



4

Quiz

The input image has been converted into a matrix of size 28×28 and a kernel/filter of size 7×7 with a stride of 1. What will be the size of the convoluted matrix?

1. 22 X 22
2. 21 X 21
3. 28 X 28
4. 7 X 7

The size of the convoluted matrix is given by $C = ((I - F + 2P)/S) + 1$,
where
C is the size of the Convoluted matrix,
I is the size of the input matrix,
F the size of the filter matrix and
P the padding applied to the input matrix.

Here P=0, I=28, F=7 and S=1. There the answer is 22.

5

Quiz

Given below is an input matrix named I, kernel F and Convoluted matrix named C. Which of the following is the correct option for matrix C with stride = 2 ?

I

1	0	0	1	1	0	1
0	0	1	1	1	0	1
1	1	1	0	1	0	1
1	1	0	1	0	0	0
1	0	1	0	1	1	0
0	1	1	0	0	1	1
0	1	1	1	0	1	1

F

1	0	0
0	1	1
1	1	0

A)

C

4	4	3	3	3
4	2	3	3	2
3	3	3	1	3
3	4	2	3	2
4	3	3	2	4

B)

C

4	4	3	3	3
4	2	3	2	2
3	2	3	3	3
3	4	2	3	2
4	3	2	2	4

C)

C

4	3	3
3	3	3
4	3	4

D)

C

4	3	3
3	2	2
3	3	4

1 and 2 are automatically eliminated since they do not conform to the output size for a stride of 2. Upon calculation option 3 is the correct answer.

6

Quiz

Which of following activation function can't be used at output layer to classify an image ?

1. Sigmoid
2. Tanh
3. ReLU
4. If(x>5,1,0)
5. None of the above

ReLU gives continuous output in range 0 to infinity. But in output layer, we want a finite range of values. So option C is correct.

7

Quiz

Which of following is the main contribution of VGG16?

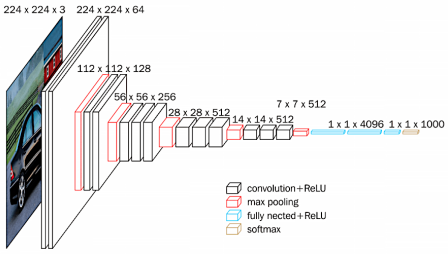
1. Four building blocks of CNN (Convolution, Non-linearity, Pooling and classification)
2. Use of ReLU as non-linearities and Dropout to avoid overfitting
3. First use of GPU
4. Depth of the network is a critical component for good performance.
5. None of the above

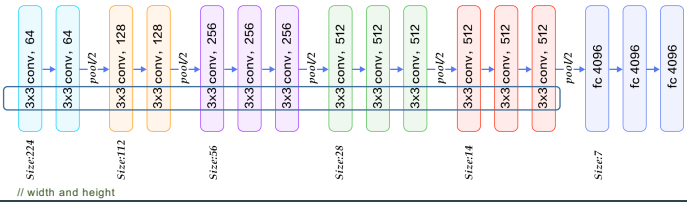
The first one is LeNet-5, the 2nd and 3rd are AlexNet and the 4th is VGG16

8

VGG16

- Its main contribution was in showing that the depth of the network (# of layers) is a critical component for good performance.
- Simplified Architecture
  - $Conv = 3\times3\ filter,$ $s = 1, same\ padding$
  - Max-Pool = 2×2 , $s = 1$
- 138M parameters





9

Training a single AI model can emit as much carbon as five cars in their lifetimes

Deep learning has a terrible carbon footprint.

by **Karen Hao** Jun 6, 2019

The artificial-intelligence industry is often compared to the oil industry: once mined and refined, data, like oil, can be a highly lucrative commodity. Now it seems the metaphor may extend even further. Like its fossil-fuel counterpart, the process of deep learning has an outsize environmental impact.

In a new paper, researchers at the University of Massachusetts, Amherst, performed a life cycle assessment for training several common large AI models. They found that the process can emit more than 626,000 pounds of carbon dioxide equivalent—nearly five times the lifetime emissions of the average American car (and that includes manufacture of the car itself).

Common carbon footprint benchmarks

in lbs of CO2 equivalent

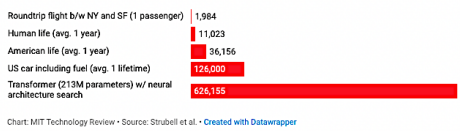


Chart: MIT Technology Review • Source: Strubell et al. • Created with Datawrapper

10

Forbes Billionaires Innovation Leadership Money Consumer Industry Lifestyle

5,551 views | Apr 19, 2018, 02:09pm

Google Has Reached 100% Renewable Energy, So I'm Issuing A New Challenge

Navigant Research Contributor
Navigant Research Contributor Group
Energy

by Roberto Rodriguez Labastida

Photo Courtesy: iStock iStock

In 2010, Google started a journey to replace the electricity it uses with renewable sources by signing its first power purchase agreement (PPA) with a 114 MW wind farm in Iowa.

To ensure that its purchases have a meaningful impact on the environment, Google has followed the concept of additionality, which means that all the electricity it buys is funding new renewable energy projects.

In 2017—2.6 GW over 20 projects and 7 years later—Google announced that it reached its 100% renewables target. This is a massive achievement, especially considering that Google began these plans when grid parity was little more than a dream for wind, and solar energy was a technology that only rich Californians and Germans put on their roofs.

CNN BUSINESS Markets Tech Media Success Perspectives Video

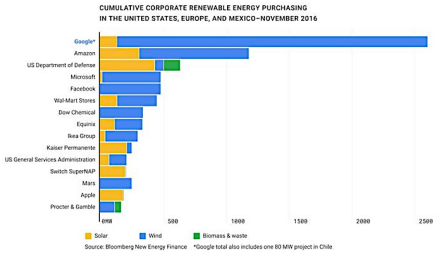
Work Transformed

Apple is now completely powered by clean energy

by Kaya Yurieff @kyurieff
April 10, 2018: 2:13 PM ET

Apple has made good on its promise to go green.

On Monday, the tech giant announced that all of its retail stores, data centers and corporate offices now run on 100% clean energy.

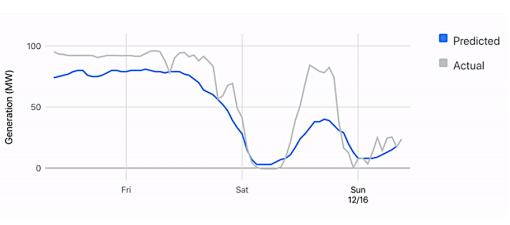


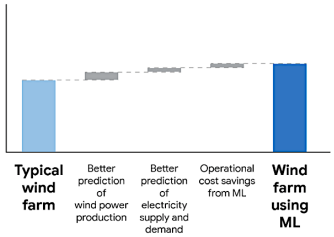
11

DeepMind Carbon Free Energy Program

ML boosts the values of wind energy.

- Successfully predict wind power output 36 hours ahead of actual generation.
  - A machine learning algorithm trained on widely available *weather forecasts* and *historical turbine data* – *which ML algorithm??*
- Recommend how to make optimal hourly delivery commitments to the power grid a full day in advance
- Energy source that can be scheduled are often more valuable to the grid.

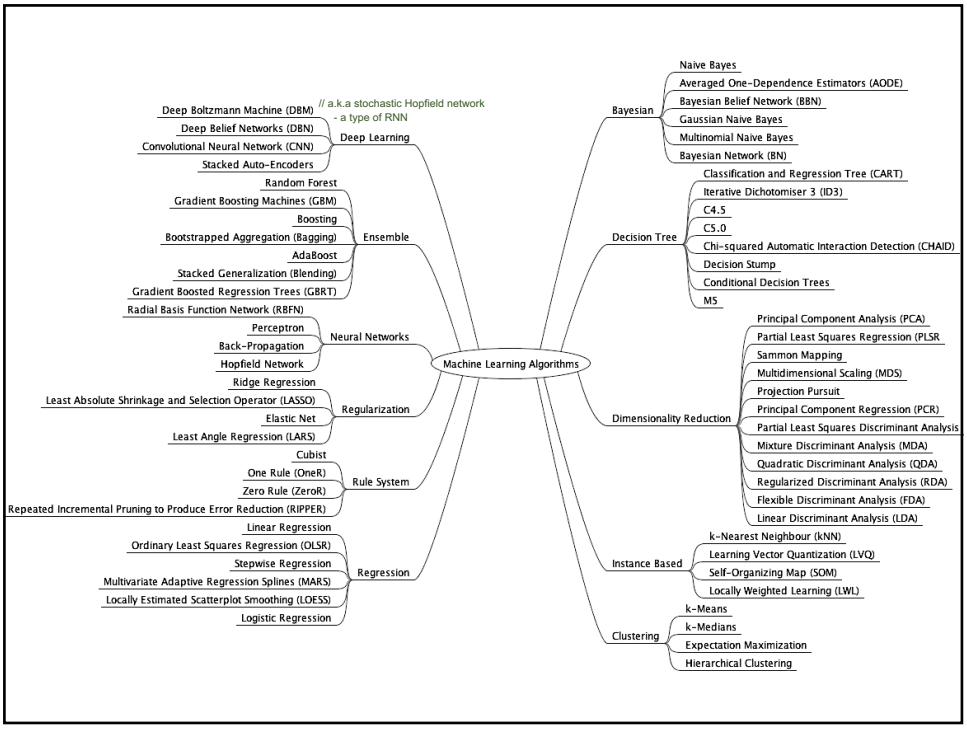




Illustrated results from 2018 Google/DeepMind field study

[Source] https://www.blog.google/outreach-initiatives/environment/meeting-our-match-buying-100-percent-renewable-energy/

12



13

Overview

We covered:

- Convolutional Neural Network
  - Convolution in 2D and 3D
  - Non-linearity
  - Pooling
  - Classification
  - Other CNNs (Alexnet, VGG16)

We will cover:

- Vanilla RNN
- Gated RNN (LSTM)
- BRNN
- Attention

14

Discussion

What is sequential data? Give a few examples

In small groups discuss what you think sequential data is and their importance is.

You have **3 minutes** and then we will discuss your answers.

15

Regression

$$\hat{y} = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2$$

($\hat{y}$: prediction estimate; $\hat{w}_0$: intercept estimate; $\hat{w}_1$: parameter estimate; x_1: input measurement)

Choose intercept and parameter estimates to *minimize*.

squared error function

$$\sum_{\text{training data}} (y_i - \hat{y}_i)^2$$

$$\text{logit}(\hat{p}) = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2$$

$$\hat{p} = \frac{1}{1 + e^{-\text{logit}(\hat{p})}}$$

Find parameter estimates by *maximizing*

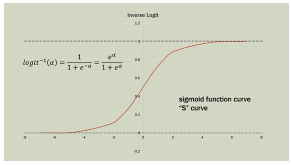
$$\sum \log(\hat{p}_i) + \sum \log(1 - \hat{p}_i)$$

primary outcome training cases; secondary outcome training cases

log-likelihood function

To obtain prediction estimates, the logit equation is solved for *p*.

The logistic function is simply the inverse of the logit function.



16

Neural Network

Neural networks can be seen as natural extension of our initial linear regression example.

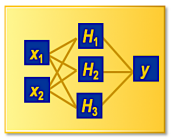
$$\log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{w}_{00} + \hat{w}_{01} H_1 + \hat{w}_{02} H_2 + \hat{w}_{03} H_3$$

Weight estimates found by *maximizing*:

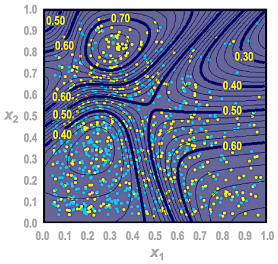
$$\sum \log(\hat{p}_i) + \sum \log(1-\hat{p}_i)$$

primary outcome training cases — secondary outcome training cases

Log-likelihood Function



$$H_1 = \tanh(\hat{w}_{10} + \hat{w}_{11} x_1 + \hat{w}_{12} x_2)$$
$$H_2 = \tanh(\hat{w}_{20} + \hat{w}_{21} x_1 + \hat{w}_{22} x_2)$$
$$H_3 = \tanh(\hat{w}_{30} + \hat{w}_{31} x_1 + \hat{w}_{32} x_2)$$



17

Proc. Natl. Acad. Sci. USA
Vol. 79, pp. 2554–2558, April 1982
Biophysics

Neural networks and physical systems with emergent collective computational abilities

(associative memory/parallel processing/categorization/content-addressable memory/fail-soft devices)

J. J. HOPFIELD

Division of Chemistry and Biology, California Institute of Technology, Pasadena, California 91125; and Bell Laboratories, Murray Hill, New Jersey 07974

Contributed by John J. Hopfield, January 15, 1982

ABSTRACT **Computational properties of use to biological organisms or to the construction of computers can emerge as collective properties of systems having a large number of simple equivalent components (or neurons). The physical meaning of content-addressable memory is described by an appropriate phase space flow of the state of a system. A model of such a system is given, based on aspects of neurobiology but readily adapted to integrated circuits. The collective properties of this model produce a content-addressable memory which correctly yields an entire memory from any subpart of sufficient size. The algorithm for the time evolution of the state of the system is based on asynchronous parallel processing. Additional emergent collective properties include some capacity for generalization, familiarity recognition, categorization, error correction, and time sequence retention. The collective properties are only weakly sensitive to details of the modeling or the failure of individual devices.**

Given the dynamical electrochemical properties of neurons and their interconnections (synapses), we readily understand schemes that use a few neurons to obtain elementary useful biological behavior (1–3). Our understanding of such simple circuits in electronics allows us to plan larger and more complex circuits which are essential to large computers. Because evolution has no such plan, it becomes relevant to ask whether the ability of large collections of neurons to perform "computational" tasks may in part be a spontaneous collective consequence of having a large number of interacting simple neurons.

In physical systems made from a large number of simple elements, interactions among large numbers of elementary components yield collective phenomena such as the stable magnetic orientations and domains in a magnetic system or the vortex patterns in fluid flow. Do analogous collective phenomena in a system of simple interacting neurons have useful "computational" correlates? For example, are the stability of memories, the construction of categories of generalization, or time-secalized content-addressable memory or categorizer using extensive asynchronous parallel processing.

The general content-addressable memory of a physical system

Suppose that an item stored in memory is "H. A. Kramers & G. H. Wannier *Phys. Rev.* **60**, 252 (1941)." A general content-addressable memory would be capable of retrieving this entire memory item on the basis of sufficient partial information. The input "& Wannier, (1941)" might suffice. An ideal memory could deal with errors and retrieve this reference even from the input "Vannier, (1941)". In computers, only relatively simple forms of content-addressable memory have been made in hardware (10, 11). Sophisticated ideas like error correction in accessing information are usually introduced as software (10).

There are classes of physical systems whose spontaneous behavior can be used as a form of general (and error-correcting) content-addressable memory. Consider the time evolution of a physical system that can be described by a set of general coordinates. A point in state space then represents the instantaneous condition of the system. This state space may be either continuous or discrete (as in the case of N Ising spins).

The equations of motion of the system describe a flow in state space. Various classes of flow patterns are possible, but the systems of use for memory particularly include those that flow toward locally stable points from anywhere within regions around those points. A particle with frictional damping moving in a potential well with two minima exemplifies such a dynamics.

If the flow is not completely deterministic, the description is more complicated. In the two-well problems above, if the frictional force is characterized by a temperature, it must also produce a random driving force. The limit points become small limiting regions, and the stability becomes not absolute. But as long as the stochastic effects are small, the essence of local stable points remains.

18

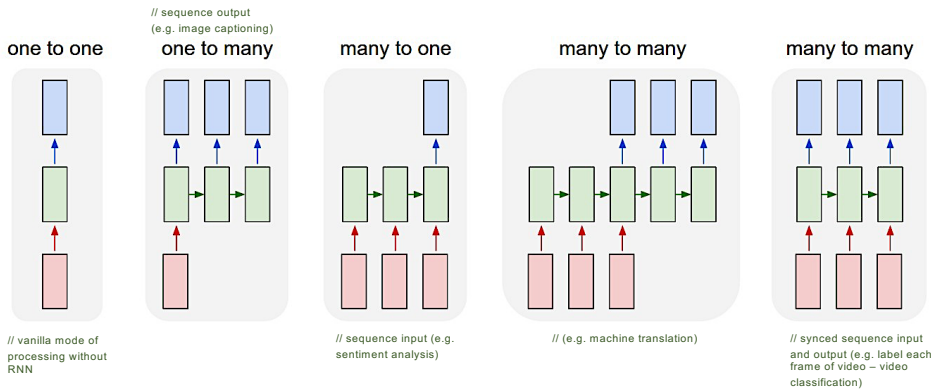
RNNs

In the last few years, there have been incredible success applying RNNs to a variety of problems:

- speech recognition
- language modelling
- translation
- image captioning
- energy systems
- biology
- finance
- etc

19

RNN Flexibility

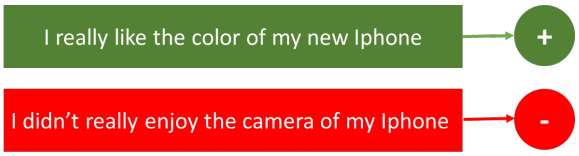


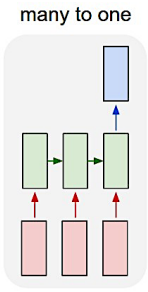
20

Sentiment Classification

A task of simply classifying tweets into positive and negative sentiment – useful for big data

- Input : a tweet of varying lengths
- Output : a fixed type and size



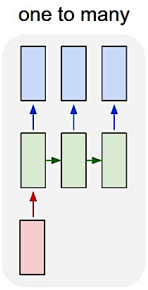


21

Image Captioning

An image for which we need a textual description.

- A single input – the image (a fixed size)
- Multiple output – a series or sequence of words (a description of varying lengths)

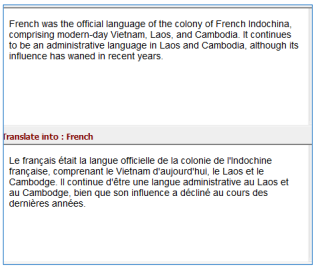


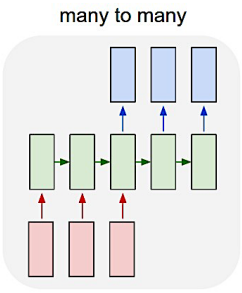
22

Language Translation

Translate English to French

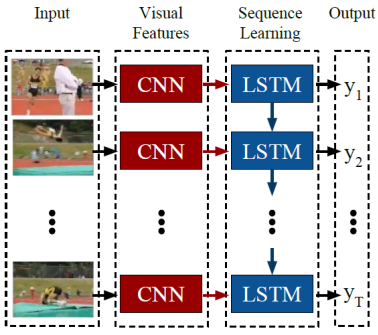
- Each language has it's own semantics and would have varying lengths for the same sentence.
- Both inputs and outputs are of varying lengths.

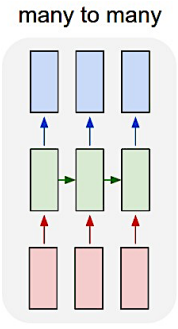




Video Classification

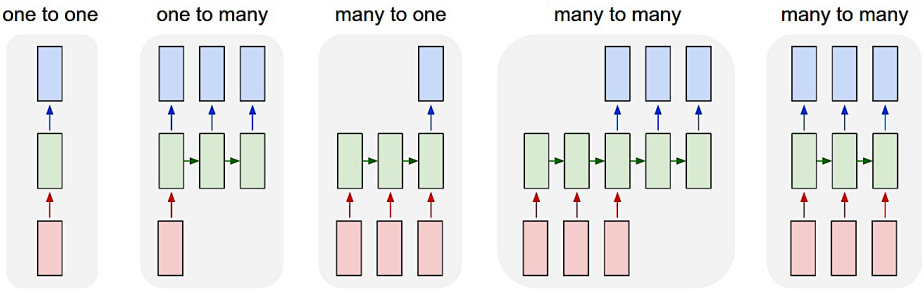
Label each frame of a video (motion)





Quiz

What RNN architecture is suitable for the prediction of wind power output next 3 days using *weather forecasts* and *historical turbine data*?



25

RNNs

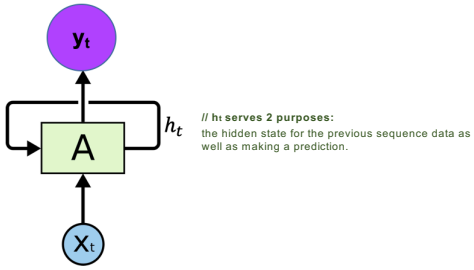
Predict the next word given a sequence of the N previous words.

- Your thoughts have persistence.
- Traditional neural networks can't do this, and it seems like a major shortcoming.
- Recurrent neural networks address this issue.
- They are networks with loops in them, allowing information to persist.

26

RNN

RNNs have loops.



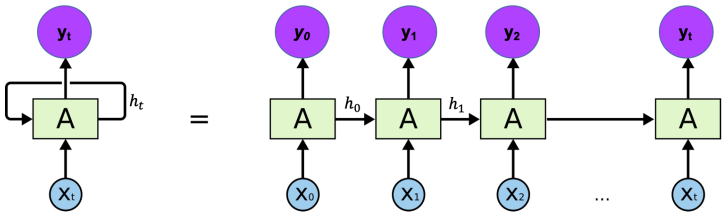
- A chunk of neural network, A, looks at some input x_t and outputs a value h_t (hidden state / internal state / memory).
- A loop allows information to be passed from one step of the network to the next.

27

Unrolled RNN

A RNN can be thought of as multiple copies of the same network, each passing a message to a successor.

- Consider what happens if we unroll the loop:



- This chain-like nature reveals that RNNs are intimately related to sequences and lists.
- They're the natural architecture of neural network to use for such data.

28

RNN and Sequential Data

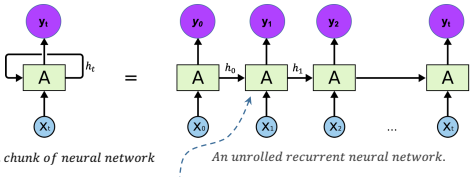
Unlike neural networks, RNNs can use their (hidden) internal state (memory) to process sequences of inputs.

A sequence of vectors x is processed by applying a recurrent formula at every time step.

$$h_t = f_W(h_{t-1}, x_t)$$

new state — old state — input vector at some time step

some function with parameters W



- The h_0 and x_1 is the input for the next step.
- Similarly, h_1 from the next is the input with x_2 for the next step and so on.
- This way, it keeps remembering the context while training.

29

(Vanilla) RNN

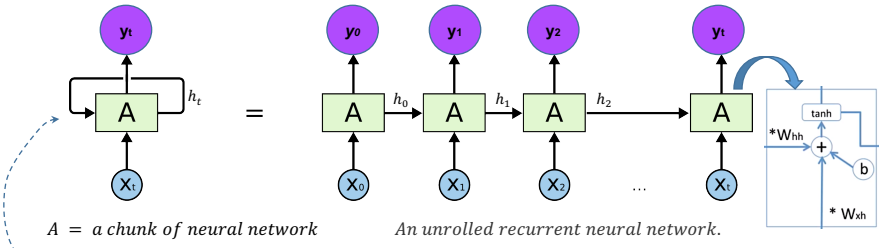
RNNs can use their (hidden) internal state (memory) to process sequences of inputs.

// h_t serves 2 purposes:
the hidden state for the previous sequence data as well as making a prediction.

$$h_t = f_w(h_{t-1}, x_t) \Rightarrow h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$WX$$

$$y_t = W_{hy}h_t$$



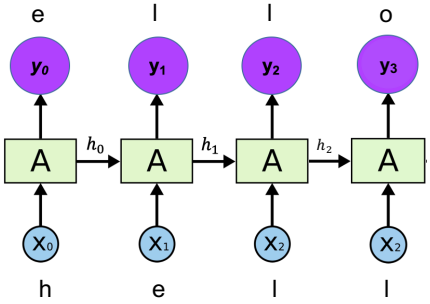
** the same function and the same set of parameters $(e.g.\ W_{hh}, W_{xh}, W_{hy})$ are used at every time step

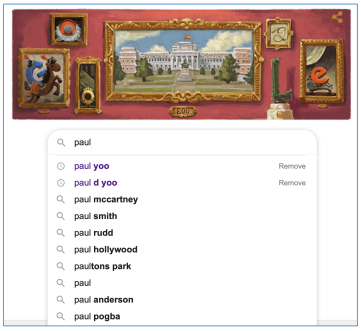
30

Character-level Language Model Example

Vocabulary: [h, e, l, o]

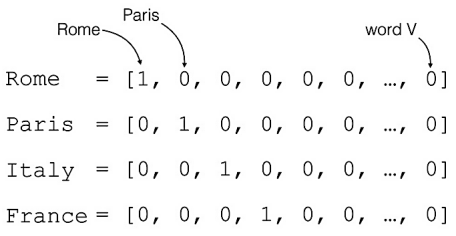
- Example training sequence : “hello”



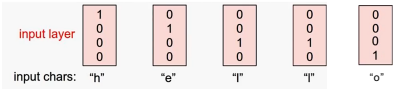


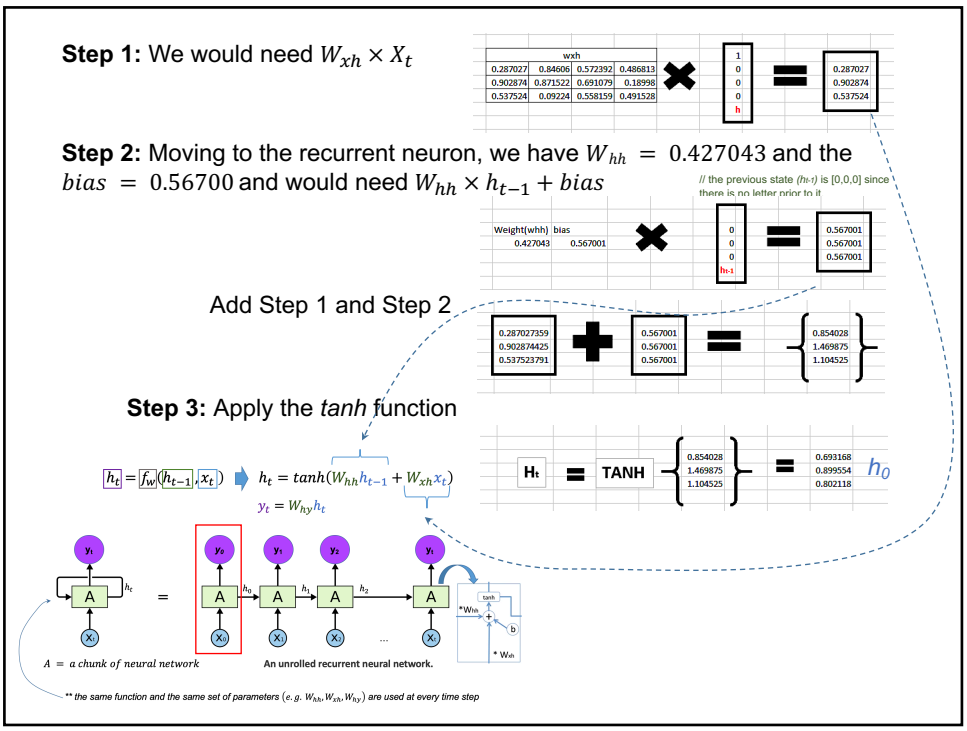
One Hot Encoding

A process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction.

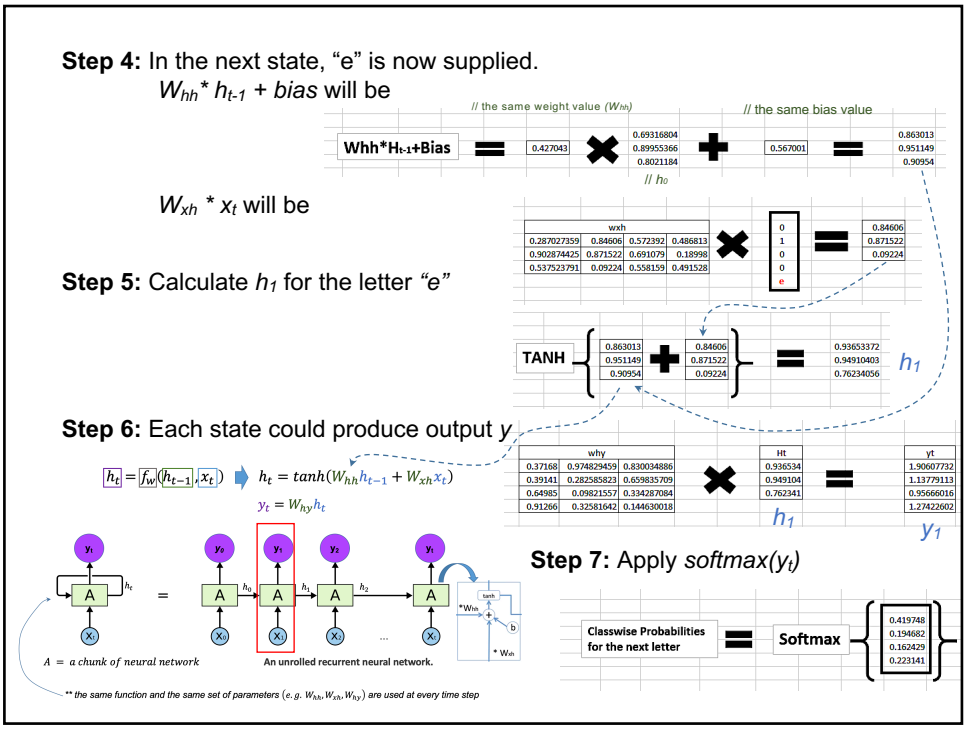


Our input




Step 1: We would need $W_{xh} \times X_t$
Step 2: Moving to the recurrent neuron, we have $W_{hh} = 0.427043$ and the $bias = 0.56700$ and would need $W_{hh} \times h_{t-1} + bias$
// the previous state (ht-1) is [0,0,0] since there is no letter prior to it
Add Step 1 and Step 2
Step 3: Apply the tanh function
$h_t = f_w(h_{t-1}, x_t) \Rightarrow h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t)$
$y_t = W_{hy}h_t$
Ht = TANH
h_0
A = a chunk of neural network
An unrolled recurrent neural network.
** the same function and the same set of parameters (e.g. W_{hh}, W_{xh}, W_{hy}) are used at every time step

33


Step 4: In the next state, "e" is now supplied.
$W_{hh} * h_{t-1} + bias$ will be
// the same weight value (W_{hh})
// the same bias value
Whh*Ht-1+Bias
// h_0
$W_{xh} * x_t$ will be
Step 5: Calculate h_1 for the letter "e"
TANH
h_1
Step 6: Each state could produce output y
$h_t = f_w(h_{t-1}, x_t) \Rightarrow h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t)$
$y_t = W_{hy}h_t$
Step 7: Apply softmax(y_t)
Classwise Probabilities for the next letter
Softmax
A = a chunk of neural network
An unrolled recurrent neural network.
** the same function and the same set of parameters (e.g. W_{hh}, W_{xh}, W_{hy}) are used at every time step

34

Training

1. The cross entropy error is first computed using the current output and the actual output
   - Remember that the network is unrolled for all the time steps
2. The gradient is calculated for each time step with respect to the weight parameter
3. Now that the weight is the same for all the time steps the gradients can be combined together for all time steps
4. The weights are then updated for both recurrent neuron and the dense layers

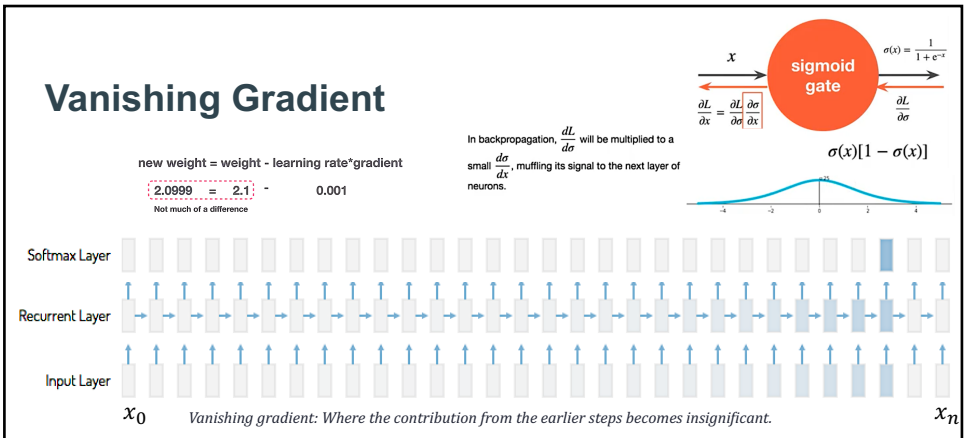
**The unrolled network looks much like a regular neural network.

**And the back propagation algorithm is similar to a regular neural network, just that we combine the gradients of the error for all time steps.

**If there are 100s of time steps – this would basically take really long for the network to converge since after unrolling the network becomes really huge.

35

Vanishing Gradient



- This arrow means that *long-term information* has to sequentially travel through all cells before getting to the current processing cell.
- This means it can be easily corrupted by being multiplied many time by small numbers < 0. This is the cause of vanishing gradients.

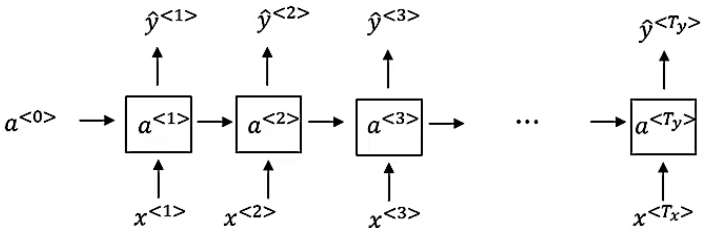
36

Vanishing Gradients

As we go back to the lower layers gradient often get smaller. eventually causing weights to never change at lower layers.

The ***cats***, which already ate, ……***were*** full.

The ***cat***, which already ate, ……***was*** full.



Basic RNN model has many ***local influences*** because of RNNs as the earlier information. Meaning the output is mostly affected by the value close to it.

37

Cho, K., Van Merriënboer, B., Bahdanau, D. and Bengio, Y., 2014. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259.

On the Properties of Neural Machine Translation: Encoder–Decoder Approaches

Kyunghyun Cho **Bart van Merriënboer** **Dzmitry Bahdanau***
Université de Montréal Jacobs University, Germany

Yoshua Bengio
Université de Montréal, CIFAR Senior Fellow

Abstract

Neural machine translation is a relatively new approach to statistical machine translation based purely on neural networks. The neural machine translation models often consist of an encoder and a decoder. The encoder extracts a fixed-length representation from a variable-length input sentence, and the decoder generates a correct translation from this representation. In this paper, we focus on analyzing the properties of the neural machine translation using two models; RNN Encoder–Decoder and a newly proposed gated recursive convolutional neural network. We show that the neural machine translation performs relatively well on short sentences without unknown words, but its performance degrades rapidly as the length of the sentence and the number of unknown words increase. Furthermore, we find that the proposed gated recursive convolutional network learns a grammatical structure of a sentence automatically.

generates a correct, variable-length target translation.

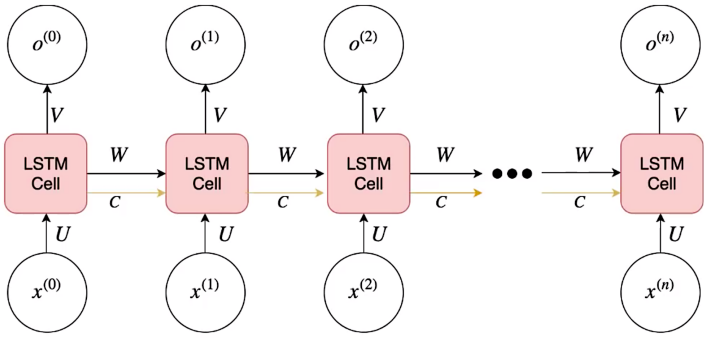
The emergence of the neural machine translation is highly significant, both practically and theoretically. Neural machine translation models require only a fraction of the memory needed by traditional statistical machine translation (SMT) models. The models we trained for this paper require only 500MB of memory in total. This stands in stark contrast with existing SMT systems, which often require tens of gigabytes of memory. This makes the neural machine translation appealing in practice. Furthermore, unlike conventional translation systems, each and every component of the neural translation model is trained jointly to maximize the translation performance.

As this approach is relatively new, there has not been much work on analyzing the properties and behavior of these models. For instance: What are the properties of sentences on which this approach performs better? How does the choice of source/target vocabulary affect the performance? In which cases does the neural machine translation fail?

38

Gated RNN (LSTM)

LSTM is the most commonly used Gated RNN, capable of learning long-term dependencies.



- α regulates the amount of information the network remembers over time.
- LSTM adds another connection from every cell called C, the cell state.

39

Chung, J., Gulcehre, C., Cho, K. and Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling

Junyoung Chung **Caglar Gulcehre** **KyungHyun Cho** **Yoshua Bengio**
Université de Montréal Université de Montréal
CIFAR Senior Fellow

Abstract

In this paper we compare different types of recurrent units in recurrent neural networks (RNNs). Especially, we focus on more sophisticated units that implement a gating mechanism, such as a long short-term memory (LSTM) unit and a recently proposed gated recurrent unit (GRU). We evaluate these recurrent units on the tasks of polyphonic music modeling and speech signal modeling. Our experiments revealed that these advanced recurrent units are indeed better than more traditional recurrent units such as tanh units. Also, we found GRU to be comparable to LSTM.

1 Introduction

Recurrent neural networks have recently shown promising results in many machine learning tasks, especially when input and/or output are of variable length [see, e.g., Graves, 2012]. More recently, Sutskever et al. [2014] and Bahdanau et al. [2014] reported that recurrent neural networks are able to perform as well as the existing, well-developed systems on a challenging task of machine translation.

One interesting observation, we make from these recent successes is that almost none of these successes were achieved with a vanilla recurrent neural network. Rather, it was a recurrent neural network with sophisticated recurrent hidden units, such as long short-term memory units [Hochreiter and Schmidhuber, 1997], that was used in those successful applications.

40

Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), pp.1735-1780.

LONG SHORT-TERM MEMORY

NEURAL COMPUTATION 9(8):1735–1780, 1997

Sepp Hochreiter
Fakultät für Informatik
Technische Universität München
80290 München, Germany
hochreit@informatik.tu-muenchen.de
http://www7.informatik.tu-muenchen.de/~hochreit

Jürgen Schmidhuber
IDSIA
Corso Elvezia 36
6900 Lugano, Switzerland
juergen@idsia.ch
http://www.idsia.ch/~juergen

Abstract

Learning to store information over extended time intervals via recurrent backpropagation takes a very long time, mostly due to insufficient, decaying error back flow. We briefly review Hochreiter's 1991 analysis of this problem, then address it by introducing a novel, efficient, gradient-based method called "Long Short-Term Memory" (LSTM). Truncating the gradient where this does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete time steps by enforcing *constant* error flow through "constant error carrousels" within special units. Multiplicative gate units learn to open and close access to the constant error flow. LSTM is local in space and time; its computational complexity per time step and weight is $O(1)$. Our experiments with artificial data involve local, distributed, real-valued, and noisy pattern representations. In comparisons with RTRL, BPTT, Recurrent Cascade-Correlation, Elman nets, and Neural Sequence Chunking, LSTM leads to many more successful runs, and learns much faster. LSTM also solves complex, artificial long time lag tasks that have never been solved by previous recurrent network algorithms.

1 INTRODUCTION

Recurrent networks can in principle use their feedback connections to store representations of recent input events in form of activations ("short-term memory", as opposed to "long-term memory" embodied by slowly changing weights). This is potentially significant for many applications, including speech processing, non-Markovian control, and music composition (e.g., Mozer 1992). The most widely used algorithms for learning *what* to put in short-term memory, however, take too much time or do not work well at all, especially when minimal time lags between inputs and corresponding teacher signals are long. Although theoretically fascinating, existing methods do not provide clear *practical* advantages over, say, backprop in feedforward nets with limited time windows. This paper will review an analysis of the problem and suggest a remedy.

41

LSTM

- When you read the review, your brain subconsciously only remembers important keywords.
- You pick up words like "amazing" and "perfectly balanced breakfast".

Customers Review 2,491

Thanos

September 2018

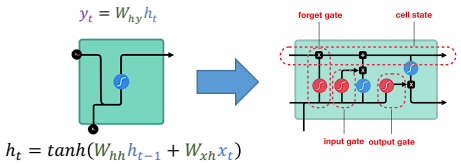
Verified Purchase

Amazing! This box of cereal gave me a perfectly balanced breakfast, as all things should be. I only ate half of it but will definitely be buying again!

A Box of Cereal
$3.99

42

Core Concept



The core concept of LSTM's are the cell state, and it's various gates.

The cell state

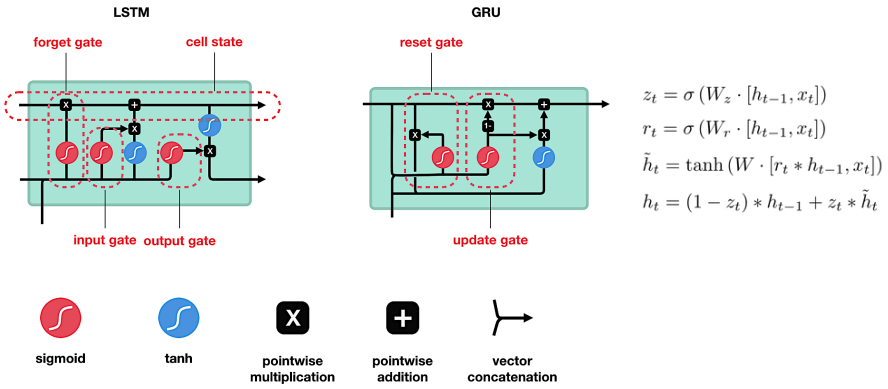
- a transport highway that transfers relative information all the way down the sequence chain – the "memory" of the network.
- information gets added or removed to the cell state via gates.

// reducing the effects of short-term memory

The gates

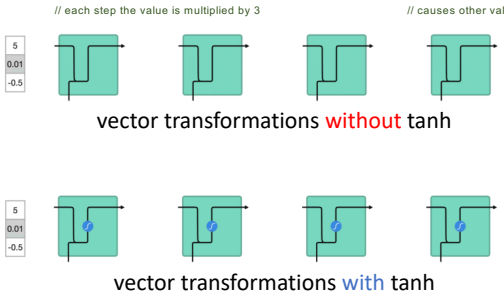
- different neural networks that decide which information is allowed on the cell state.
- can learn what information is relevant to keep or forget during training.

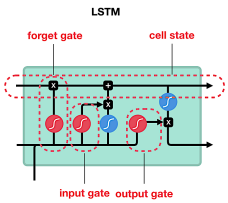
Sigmoid Vs Tanh



Tanh

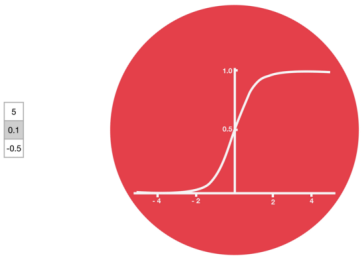
The tanh is used to regulate the values flowing through the network as it squishes values to always be between -1 and 1.

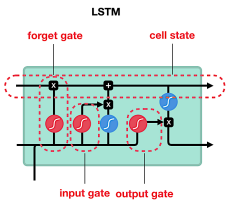




Sigmoid

- Used for Gates
- Any number getting multiplied by 0 is 0, causing values to disappears or be “forgotten”.
- Any number multiplied by 1 is the same value therefore that value stay’s the same or is “kept”.





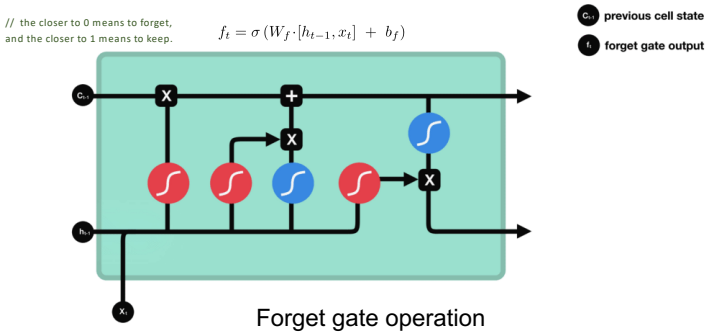
The Gates

- **The cell sates** regulates the amount of information the network remembers over time.
  - Inputs
    - **The forget gate** decides what is relevant to keep from prior steps.
    - **The input gate** decides what information is relevant to add from the current step.
- **The output gate** determines what the next hidden state should be.
  - Input
    - The cell state

47

Forget Gate

Forget gate decides what information should be thrown away or kept.

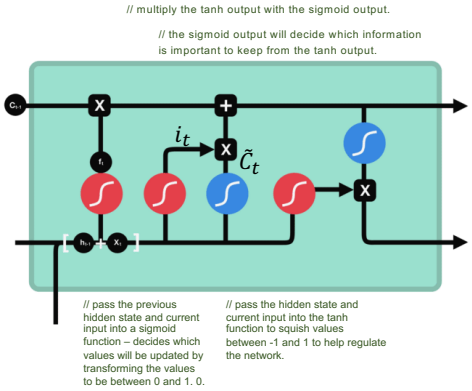


Forget gate operation

48

Input Gate

Input gate controls how much of input is used in the new cell state.

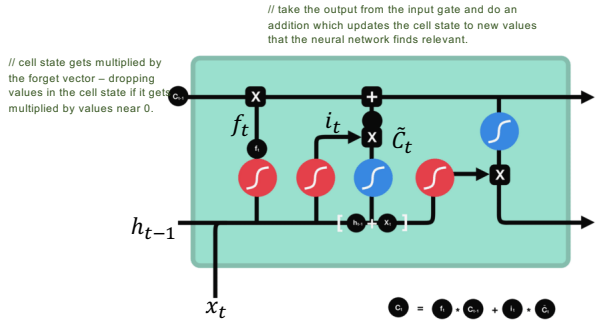


previous cell state
forget gate output
input gate output
candidate

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Cell State

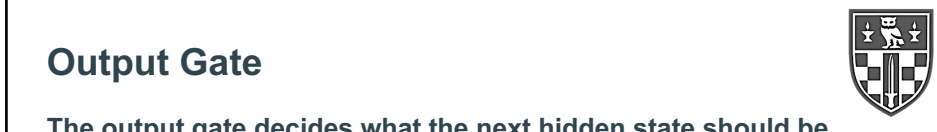
Update the old cell state, C_{t-1}, into the new cell state C_t.



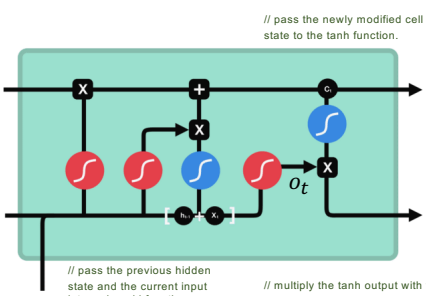
previous cell state
forget gate output
input gate output
candidate
new cell state

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Output Gate



The output gate decides what the next hidden state should be. Remember that the hidden state contains information on previous inputs.

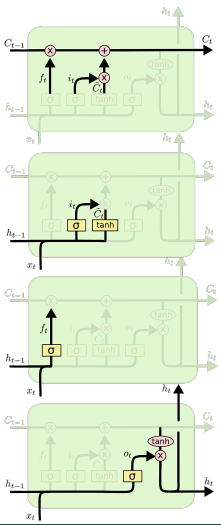


previous cell state
forget gate output
input gate output
candidate
new cell state
output gate output
hidden state

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

51

LSTM Gates



Update the old cell state, C_{t-1}, into the new cell state C_t.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Input Gate: to control how much of input is used in the new cell state.

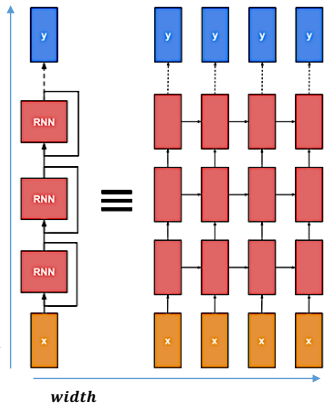
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Forget Gate: to decide what information we're going to throw away/reset from the cell state.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

Output Gate: decide whether the info. of the C_t is visible or not

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$



LSTM Parameters: $b_i, W_i, b_f, W_f, b_c, W_c, b_o, W_o$

52

LSTM Review

- **The cell sates** regulates the amount of information the network remembers over time.
- **The forget gate** decides what is relevant to keep from prior steps.
- **The input gate** decides what information is relevant to add from the current step.
- **The output gate** determines what the next hidden state should be.

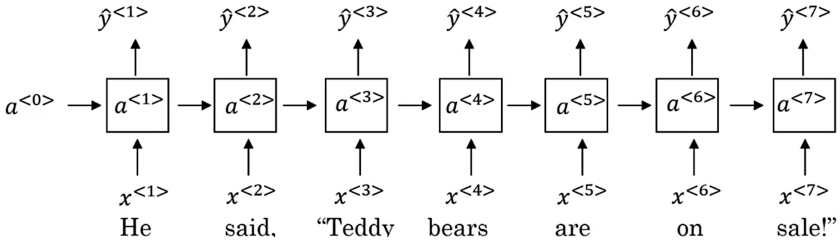
Bidirectional RNN

Getting information from future

// need more information than this

He said, "Teddy bears are on sale!"

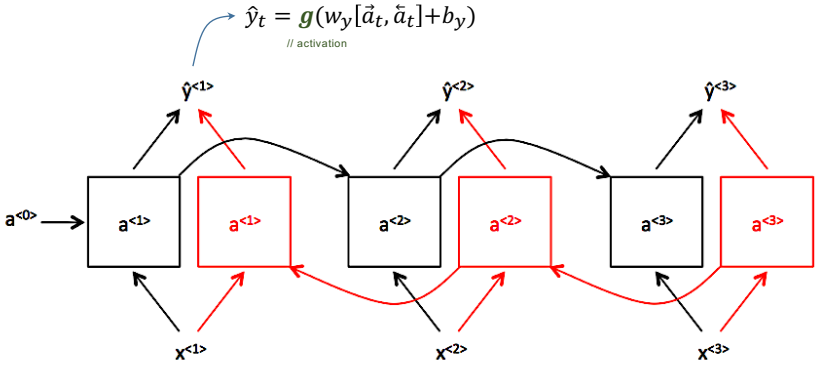
He said, "Teddy Roosevelt was a great President!"



BRNN

RNN Activation

$h_t = f_w(h_{t-1}, x_t) \Rightarrow h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t)$



55

IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 45, NO. 11, NOVEMBER 1997 2673

Bidirectional Recurrent Neural Networks

Mike Schuster and Kuldip K. Paliwal, *Member, IEEE*

***Abstract*— In the first part of this paper, a regular recurrent neural network (RNN) is extended to a bidirectional recurrent neural network (BRNN). The BRNN can be trained without the limitation of using input information just up to a preset future frame. This is accomplished by training it simultaneously in positive and negative time direction. Structure and training procedure of the proposed network are explained. In regression and classification experiments on artificial data, the proposed structure gives better results than other approaches. For real data, classification experiments for phonemes from the TIMIT database show the same tendency.**

In the second part of this paper, it is shown how the proposed bidirectional structure can be easily modified to allow efficient estimation of the conditional posterior probability of complete symbol sequences without making any explicit assumption about the shape of the distribution. For this part, experiments on real data are reported.

***Index Terms*— Recurrent neural networks.**

I. Introduction

A. General

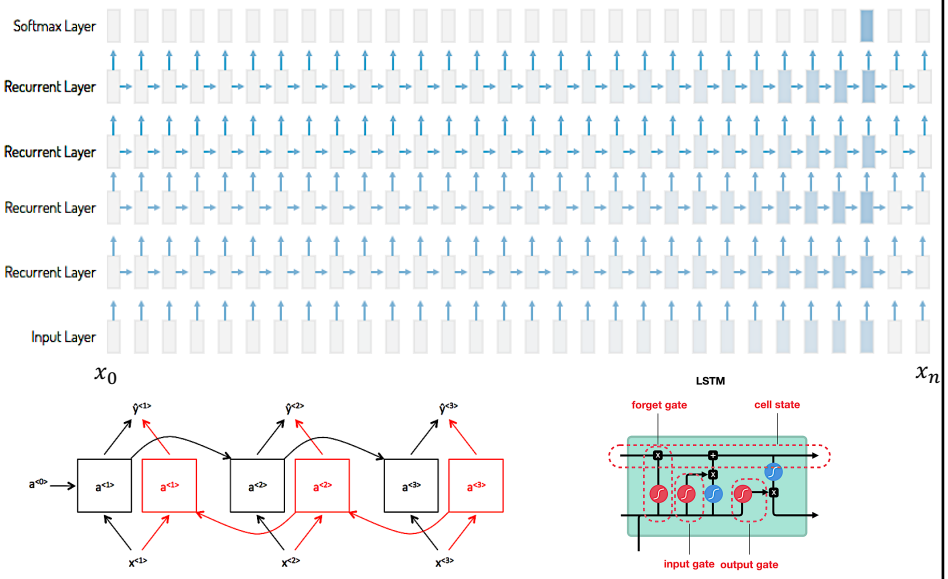
MANY classification and regression problems of engineering interest are currently solved with statistical approaches using the principle of "learning from examples" that, at least theoretically, is able to use all available input information to predict a point in the output space.

Many ANN structures have been proposed in the literature to deal with time varying patterns. Multilayer perceptrons (MLP's) have the limitation that they can only deal with static data patterns (i.e., input patterns of a predefined dimensionality), which requires definition of the size of the input window in advance. Waibel *et al.* [16] have pursued time delay neural networks (TDNN's), which have proven to be a useful improvement over regular MLP's in many applications. The basic idea of a TDNN is to tie certain parameters in a regular MLP structure without restricting the learning capability of the ANN too much. Recurrent neural networks (RNN's) [5], [8], [12], [13], [15] provide another alternative for incorporating temporal dynamics and are discussed in more detail in a later section.

In this paper, we investigate different ANN structures for incorporating temporal dynamics. We conduct a number of experiments using both artificial and real-world data. We show the superiority of RNN's over the other structures. We then point out some of the limitations of RNN's and propose a modified version of an RNN called a bidirectional recurrent neural network, which overcomes these limitations.

56

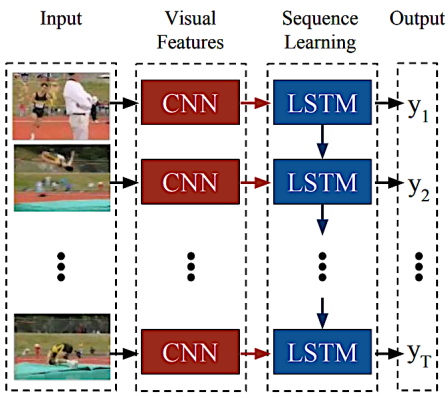
Deep RNN



57

CNN + Deep RNN

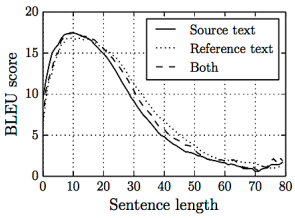
Activity Recognition



58

Attention

// an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another.



The problem of long sequences

- Difficult to memories the whole long sentence.
- Translate part by part

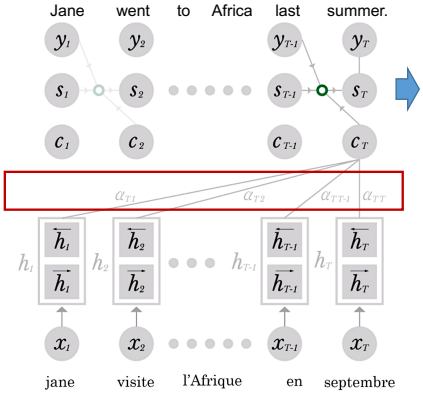
Jane s'est rendue en Afrique en septembre dernier, a apprécié la culture et a rencontré beaucoup de gens merveilleux; elle est revenue en parlant comment son voyage était merveilleux, et elle me tente d'y aller aussi.

Jane went to Africa last September, and enjoyed the culture and met many wonderful people; she came back raving about how wonderful her trip was, and is tempting me to go too.

59

Attention

Attention (combined in RNN) focuses on certain parts of the input sequence when predicting a certain part of the output sequence.



y_T: *Output* *Decoder* $p(y_T \mid y_{T-1}, \cdots, y_{T-1}, X) = p(y_{T-1}, s_T, c_T)$

$s_T = f(s_{T-1}, y_{T-1}, c_T)$ // activation = s

$c_T = \sum_{j=1}^{T} \alpha_{Tj} h_T$

$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T} exp(e_{ik})}$ $\quad e_{ij} = a(s_{i-1}, h_i)$ *where a: alignment model (parameterized)*

$h_j = [\overrightarrow{h_j}^\top; \overleftarrow{h_j}^\top]^\top$ *concatenating the forward/backward hidden state*

x_T : *Input*

Computation of attention weights at time step T, notice how this needs to be computed separately on every time step since the computation at time step T involves S_{T-1}, the state form time step $T-1$.

60

Bahdanau, D., Cho, K. and Bengio, Y., 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

61

Lab

Task 1

Given a year and a month, the task is to predict the number of international airline passengers

- the data ranges from January 1949 to December 1960 or 12 years, with 144 observations.

Task 2

Can you use SMS spam dataset to build a LSTM prediction model that will accurately classify which texts are spam?

- The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research.
  - It contains one set of SMS messages in English of 5,574 messages, tagged being ham (legitimate) or spam.
  - Download from the VLE
  - Ham 87% and Spam 13%
  - 97.6% Accuracy
- Keras + Tensorflow
- Use Anaconda

65

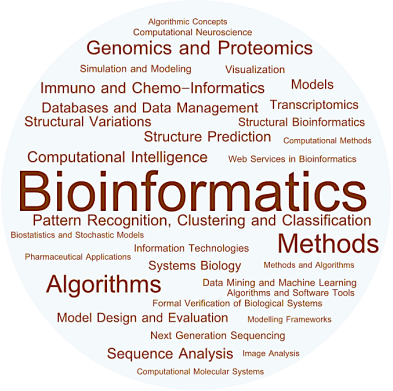
Bioinformatics

Automated Protein Function Prediction with Machine Learning

Dr Wen Cen

Abstract

As one of the major challenges in bioinformatics, accurate protein function prediction is crucial to understand the roles of protein in complex biological systems. Machine learning has been widely used in this research area and obtains significant progress in improving accuracy. Here I will be discussing the recent development of automated protein function prediction methods using the state-of-the-art machine (deep) learning techniques.



66

Questions?

paul@dcs.bbk.ac.uk

67