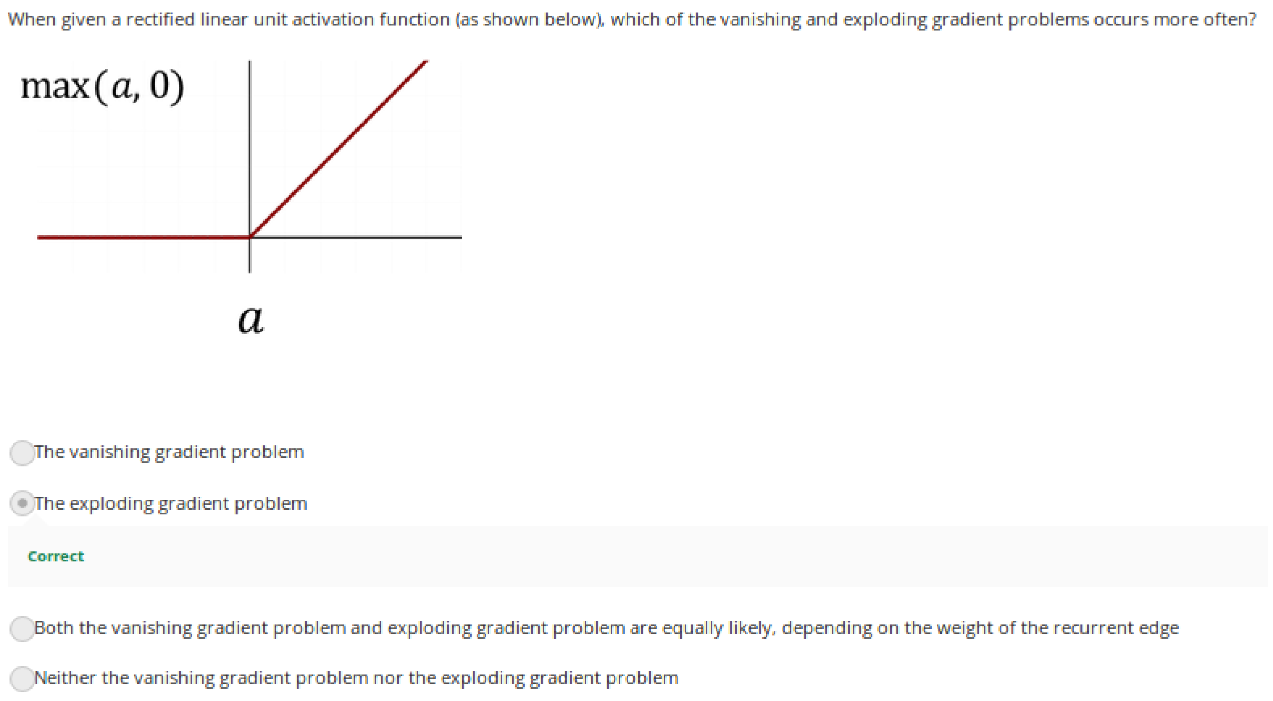
# Practical Deep Learning — Intel Course (WEEK — 03) — RNN

medium.com/@aakashgoel12/practical-deep-learning-intel-course-week-03-rnn-804800a4d12c

July 13, 2020

> Why RNN ?

- Feed fwd. NN assume training example is independent but some time temporal & context dependency need to explicitly captured by model
- Feed fwd. NN assume input vector of fixed length



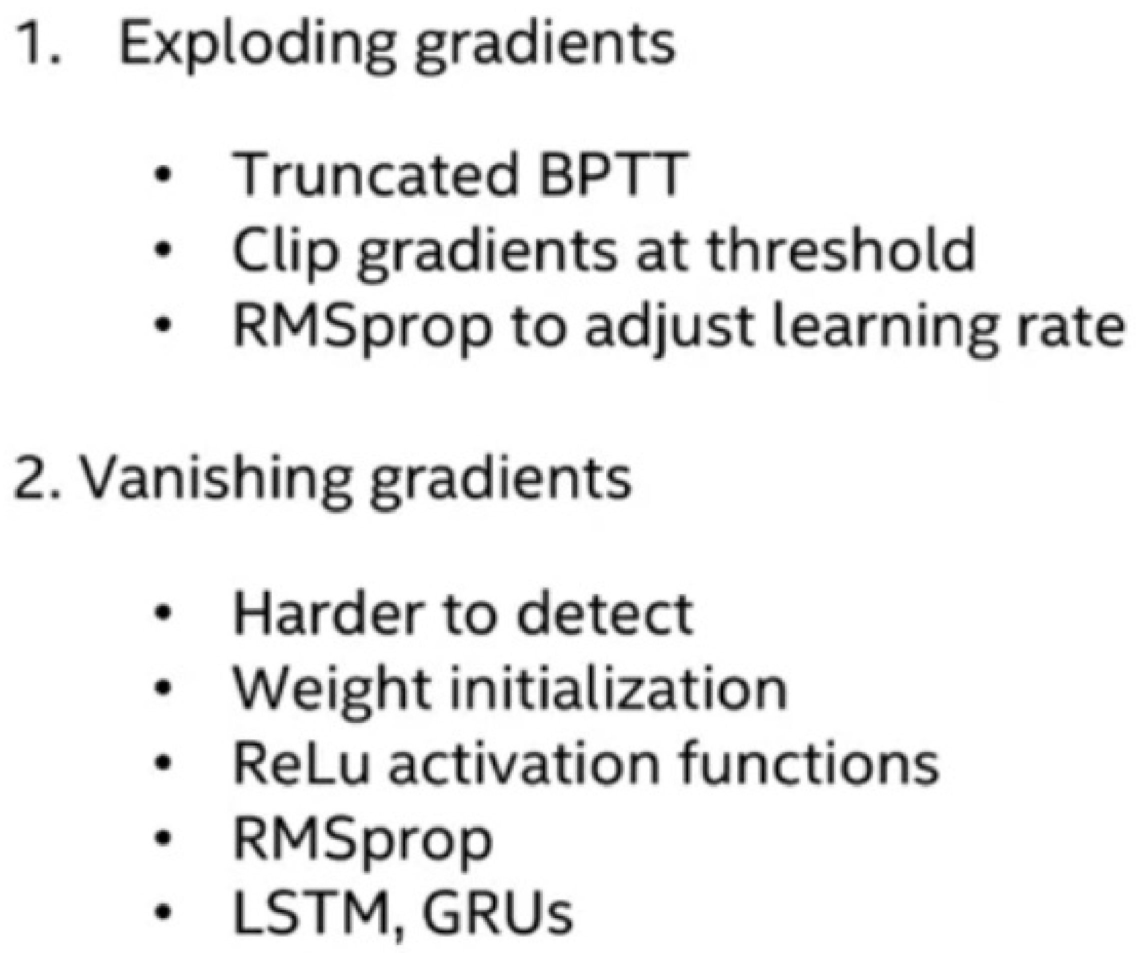
When given a rectified linear unit activation function (as shown below), which of the vanishing and exploding gradient problems occurs more often?

- The vanishing gradient problem
- [✓ The exploding gradient problem]

Correct

- Both the vanishing gradient problem and exploding gradient problem are equally likely, depending on the weight of the recurrent edge
- Neither the vanishing gradient problem nor the exploding gradient problem

Control Exploding Gradient — Clip gradients @ threshold, RMSprop adapt learning rate depend on size of gradient itself



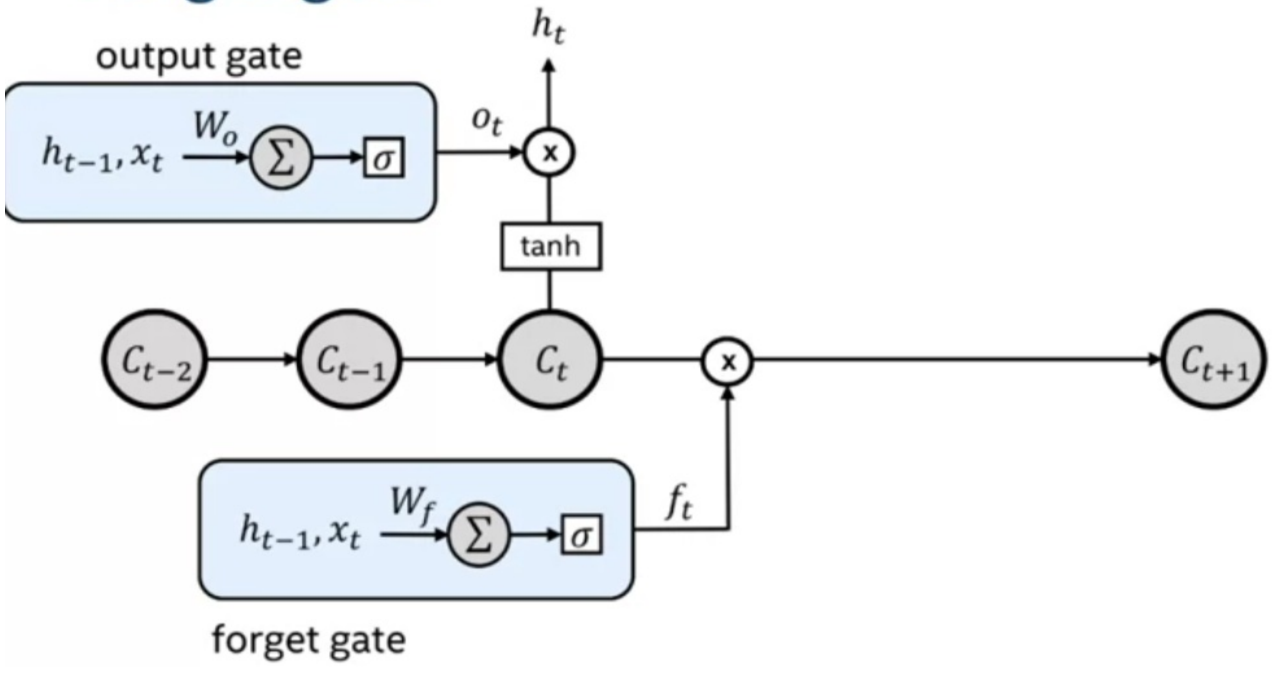
1. Exploding gradients
   - Truncated BPTT
   - Clip gradients at threshold
   - RMSprop to adjust learning rate

2. Vanishing gradients
   - Harder to detect
   - Weight initialization
   - ReLu activation functions
   - RMSprop
   - LSTM, GRUs

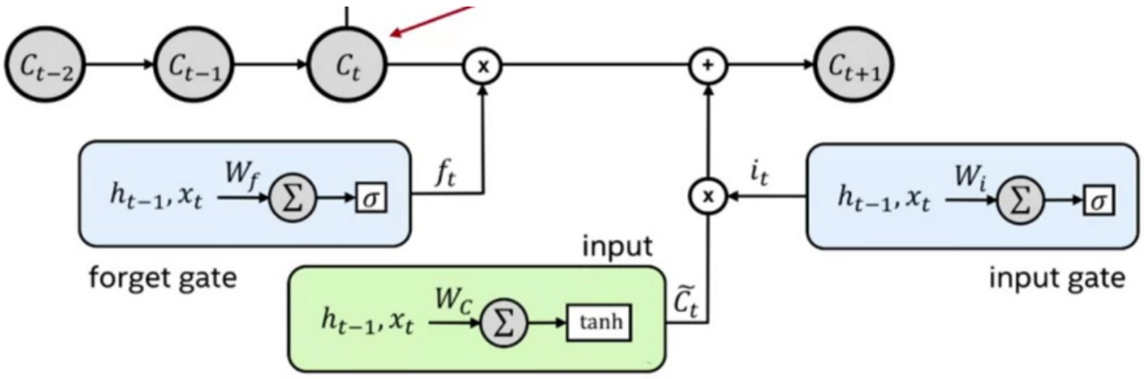
## LSTM

Forget Gate — Adding, the ability, to,flush the memory, by rewriting it. So, if memory values at C(t) are close to zero, its forgotten.

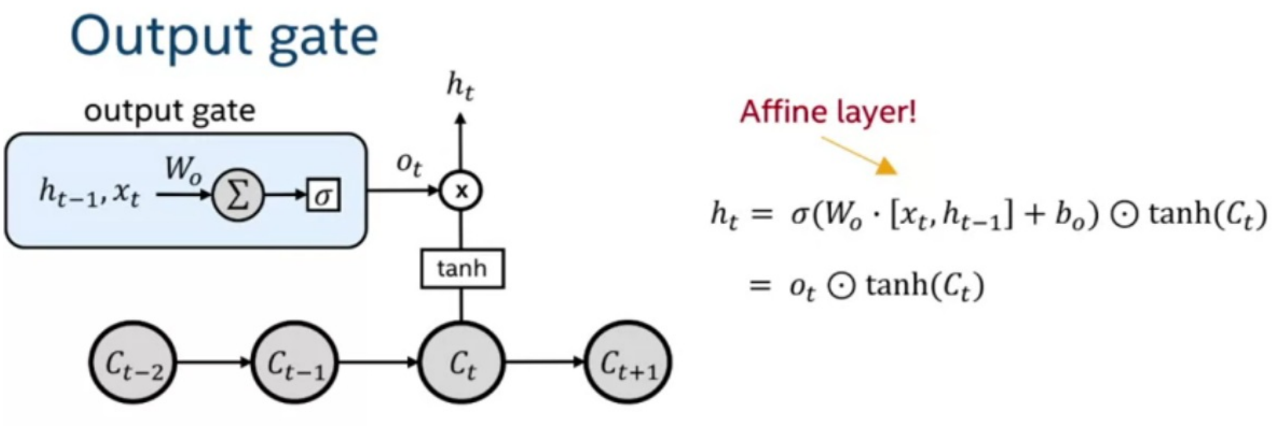
### Forget gate



Input Gate — The ability, to, add to the memory. So, we can think of the next, stage, of the LSTM, C of t, plus one, as, how much you want to, forget, from the previous time step, plus, a proposal, for the new time step input, multiply, by how much we want to accept, this new proposal.



Output Gate — and the ability, to read, from the memory.



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

## Additional resource

word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method: https://arxiv.org/abs/1402.3722

This note is an attempt to explain equation (4) (negative sampling) in "Distributed Representations of Words and Phrases and their Compositionality" by Tomas Mikolov, et al.

GloVe: Global Vectors for Word Representation: http://www.aclweb.org/anthology/D14-1162

We propose a new global log-bilinear regression model that combines the advantages of the two major model families in the literature: global matrix factorization and local context window methods. The model produces a vector space with meaningful sub-structure, as evidenced by its performance of 75 % on a recent word analogy task.

Generating Sequences With Recurrent Neural Networks: https://arxiv.org/abs/1308.0850

This paper shows how Long Short-term Memory recurrent neural networks can be used to generate complex sequences with long-range structure, simply by predicting one data point at a time. The approach is demonstrated for text (where the data are discrete) and online handwriting (where the data are real-valued).

> Sequence to sequence learning with neural networks: https://arxiv.org/abs/1409.3215

Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task from the WMT'14 dataset, the translations produced by the LSTM achieve a BLEU score of 34.8 on the entire test set

> An Empirical Exploration of Recurrent Network Architectures: http://proceedings.mlr.press/v37/jozefowicz15.pdf

We aim to determine whether the LSTM architecture is optimal or whether much better architectures exist. We conducted a thorough architecture search where we evaluated over ten thousand different RNN architectures, and identified an architecture that outperforms both the LSTM and the recently-introduced Gated Recurrent Unit (GRU) on some but not all tasks.

> Deep Visual-Semantic Alignments for Generating Image Descriptions: https://arxiv.org/abs/1412.2306

We present a model that generates natural language descriptions of images and their regions. Our approach leverages datasets of images and their sentence descriptions to learn about the inter-modal correspondences between language and visual data. Our alignment model is based on a novel combination of Convolutional Neural Networks over image regions, bidirectional Recurrent Neural Networks over sentences, and a structured objective that aligns the two modalities through a multimodal embedding. We demonstrate that our alignment model produces state of the art results in retrieval experiments on Flickr8K, Flickr30K and MSCOCO datasets.

> Show, Attend and Tell: Neural Image Caption Generation with Visual Attention: https://arxiv.org/abs/1502.03044

We introduce an attention based model that automatically learns to describe the content of images. We also show through visualization how the model is able to automatically learn to fix its gaze on salient objects while generating the corresponding words in the output sequence. We validate the use of attention with state-of-the-art performance on three benchmark datasets: Flickr8k, Flickr30k and MS COCO.

> Speech Recognition with Deep Recurrent Neural Networks: https://arxiv.org/pdf/1303.5778.pdf

This paper investigates deep recurrent neural networks. When trained end-to-end with suitable regularisation, we find that deep Long Short-term Memory RNNs achieve a test set error of 17.7% on the TIMIT phoneme recognition benchmark, which to our knowledge is the best recorded score.

> Towards End-To-End Speech Recognition with Recurrent Neural Networks: http://proceedings.mlr.press/v32/graves14.pdf

This paper presents a speech recognition system that directly transcribes audio data with text, without requiring an intermediate phonetic representation. The system is based on a combination of the deep bidirectional LSTM recurrent neural network architecture and the Connectionist Temporal Classification objective function. The system achieves a word error rate of 27.3% on the Wall Street Journal corpus with no prior linguistic information, 21.9% with only a lexicon of allowed words, and 8.2% with a trigram language model. Combining the network with a baseline system further reduces the error rate to 6.7%.

> Deep Speech 2: End-to-End Speech Recognition in English and Mandarin: https://arxiv.org/abs/1512.02595

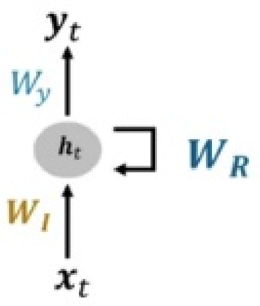
We show that an end-to-end deep learning approach can be used to recognize either English or Mandarin Chinese speech — two vastly different languages. Key to our approach is our application of HPC techniques, resulting in a 7x speedup over our previous system.

> Achieving Human Parity in Conversational Speech Recognition: https://arxiv.org/abs/1610.05256

Our latest automated system has reached human parity on the widely used NIST 2000 test set. The key to our system's performance is the use of various convolutional and LSTM acoustic model architectures, combined with a novel spatial smoothing method and lattice-free MMI acoustic training, multiple recurrent neural network language modeling approaches, and a systematic use of system combination.

> Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation: https://arxiv.org/abs/1609.08144

Neural Machine Translation (NMT) is an end-to-end learning approach for automated translation. Unfortunately, NMT systems are known to be computationally expensive both in training and in translation inference. Also, most NMT systems have difficulty with rare words. These issues have hindered NMT's use in practical deployments and services, where both accuracy and speed are essential. In this work, we present GNMT, Google's Neural Machine Translation system, which attempts to address many of these issues. Our model consists of a deep LSTM network with 8 encoder and 8 decoder layers using attention and residual connections. To accelerate the final translation speed, we employ low-precision arithmetic during inference computations. On the WMT'14 English-to-French and English-to-German benchmarks, GNMT achieves competitive results to state-of-the-art. Using a human side-by-side evaluation on a set of isolated simple sentences, it reduces translation errors by an average of 60% compared to Google's phrase-based production system.



$$\boldsymbol{h}^{(t)} = g_h(W_I\boldsymbol{x}^{(t)} + W_R\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_h)$$

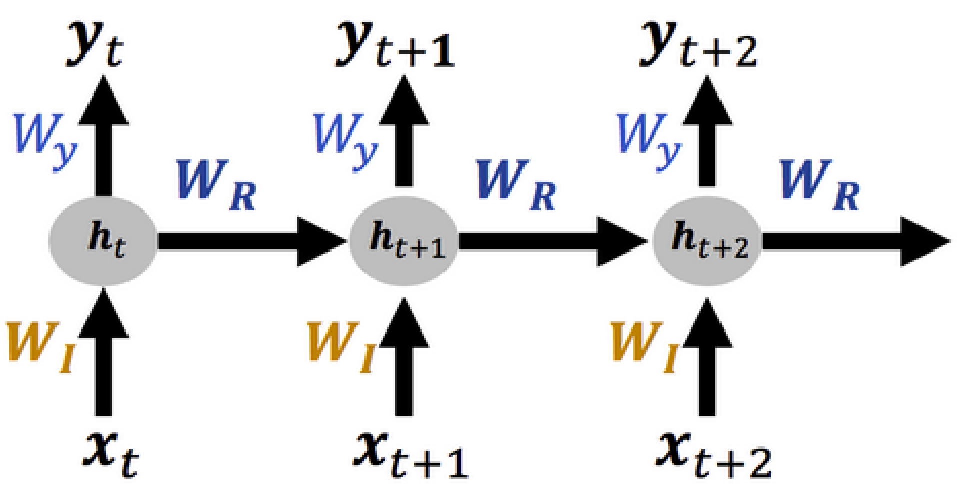
$$\boldsymbol{y}^{(t)} = g_y(W_y\boldsymbol{h}^{(t)} + \boldsymbol{b}_y)$$

## QUIZ

1. Which of the following statements are true regarding feed-forward and recurrent neural networks?

- [✓] Feed-forward networks assume independence in the training examples
- [ ] Feed-forward networks model temporal dependence and contextual dependence, unlike recurrent neural networks
- [ ] Recurrent neural networks assume an input vector of fixed length
- [✓] Recurrent neural networks, in comparison to feed-forward networks, are more useful for text or speech inputs which can vary greatly in size

2. You are given the following unrolled RNN:



If you know the following values of the inputs and weights, if the ReLU activation function is used throughout the network, what is the value for $h_t$?

| Parameter/state/input | Value |
|---|---|
| $h_{t-1}$ | 0 |
| $W_I$ | 0.1 |
| $x_t$ | 0.4 |
| $b_h$ (bias) | 0.3 |

0.34

H(t) = F(W(i) * X(t) + W(r) * H(t-1) + b)

= RELU (0.04 + 0 + 0.3)
= RELU (0.34) = 0.34

3. Given the same RNN and values as above, now assume that the activation function is the hyperbolic tangent (tanh) throughout the network as opposed to ReLU. What is the new value for the hidden state at time $t$?

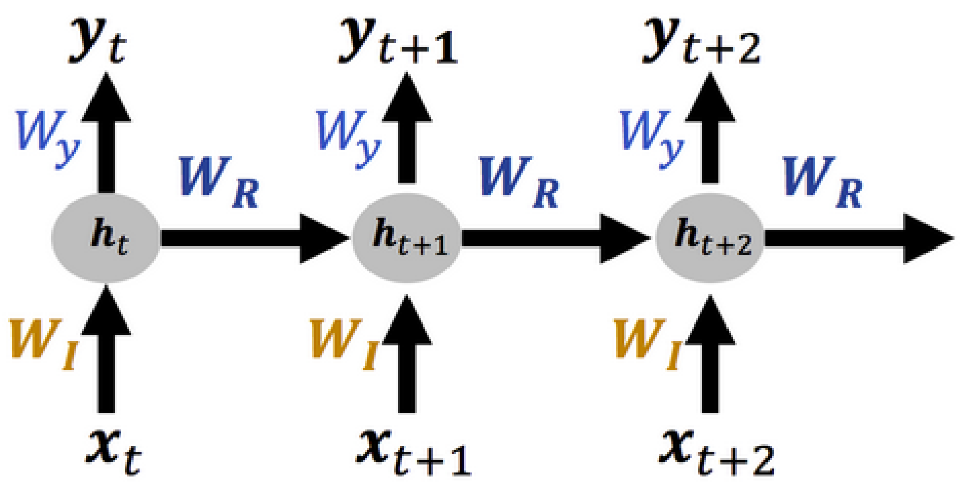
0.32

TanH (0.04 + 0 + 0.3) = 0.3274

shared 0.33 → incorrect

shared 0.327 →correct

4. You are given the following unrolled RNN:



Using the given values provided earlier and your previous calculations, assuming that the ReLU activation function is used throughout network, what is the value of $y_t$?

The following additional values are also provided:

| Parameter/state/input | Value |
|---|---|
| $W_y$ | 2.25 |
| $b_y$ (output bias) | -1.0 |

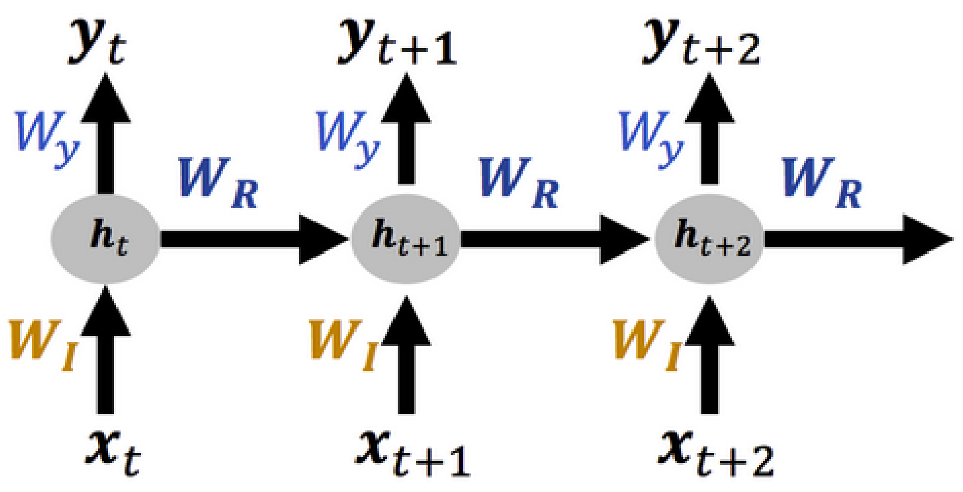
Preview
$0$

0

Y(t) = F(H(t)*W(y) + b)

= RELU(0.34*2.25–1)
= RELU(-0.235) = 0

5. You are given the following unrolled RNN:



Using the given values provided earlier and your previous calculations, assuming that the ReLU activation function is used throughout network, what is the value of $y_{t+1}$?

The following additional values are also provided:

| Parameter/state/input | Value |
|---|---|
| $W_R$ | -1.3 |
| $x_{t+1}$ | 7.7 |

0.41

**Y(t+1) = F(H(t+1)*W(y) + b)**

**H(t+1) = F(W(i) * X(t+1) + W(r) * H(t) + b)**

H(t+1) = RELU(0.1 * 7.7 + (-1.3) * 0.34 + 0.3)
= RELU(0.77–0.442 + 0.3)
= RELU (0.6279999) = 0.62799999

Y(t+1) = RELU(0.628 * 2.25–1)
= RELU(0.413) = 0.412999

shared 0.41 → incorrect

shared 0.413 → correct