LSTMs

Learning to forget

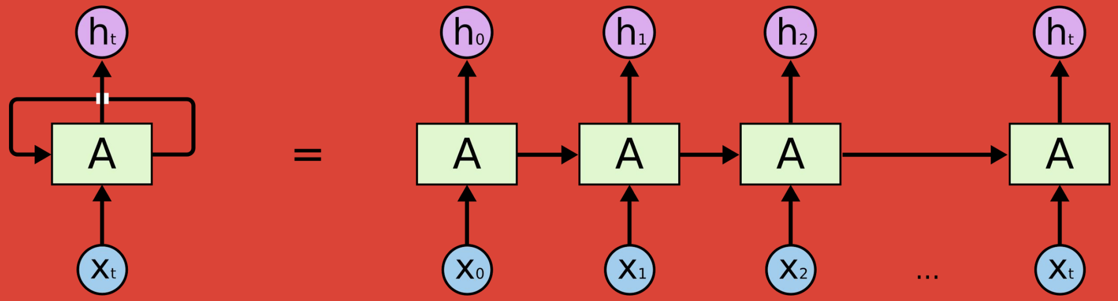
Devansh Kukreja and Eric Nie

Background/Motivation

- State-of-the-art in sequence learning
- Address the issue of remembering information
- Useful for analyzing sequential data
- Cells within RNNs and LSTMs maintain an internal state

Recurrent Neural Networks (A Brief Introduction)

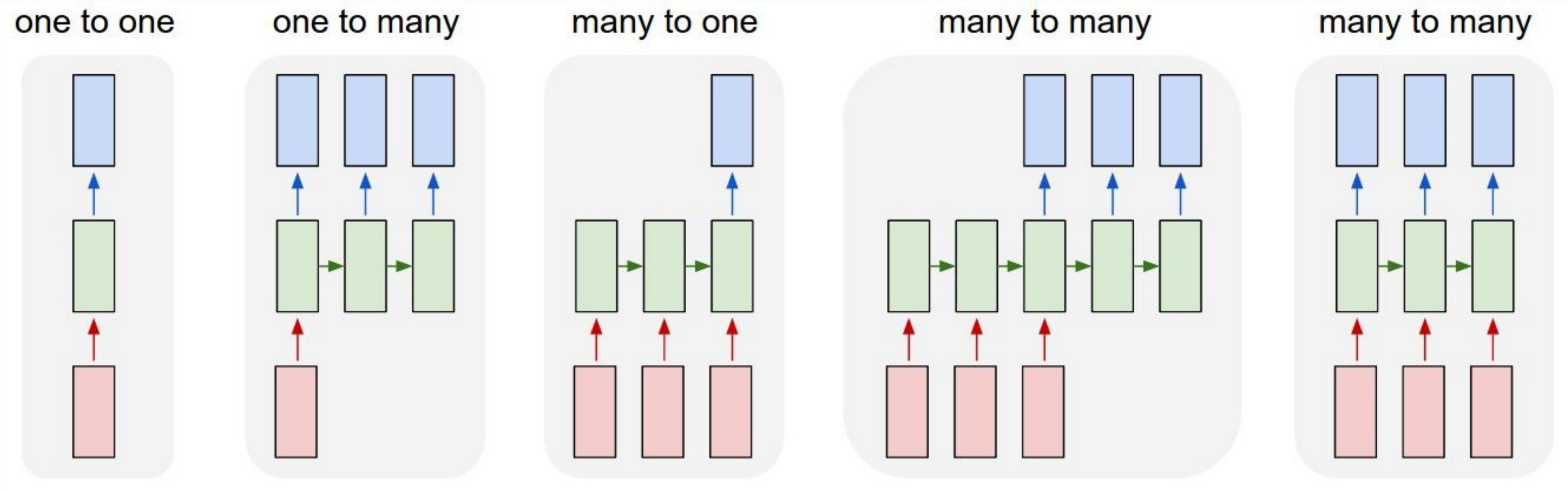
Intro



Recurrent Neural Networks

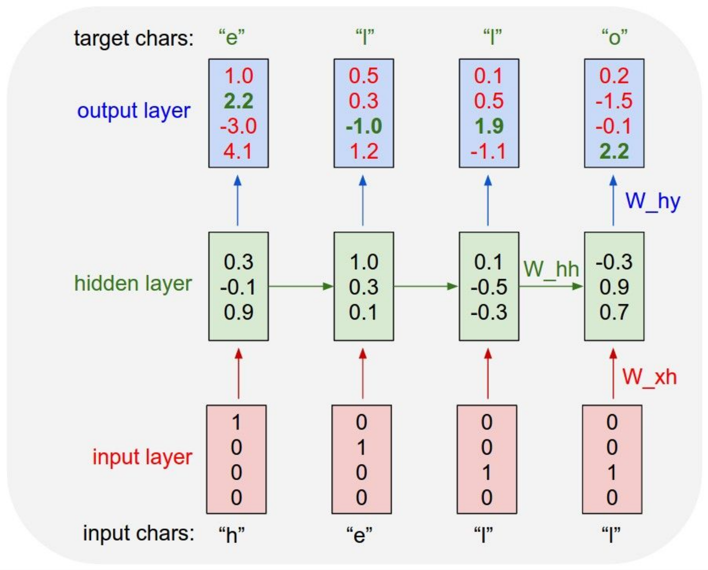
- Internal states
- Sequences of inputs and outputs
- Share weights across timesteps

More RNNs



https://karpathy.github.io/2015/05/21/rnn-effectiveness/

Character-level RNN



https://karpathy.github.io/2015/05/21/rnn-effectiveness/

- Predict the next character given input
- One-hot encoding

What can an RNN do?

Generate Text: Linux source code

```c
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
  rw->name = "Getjbbregs";
  bprm_self_clearl(&iv->version);
  regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
  return segtable;
}
```

Produce Research Papers:

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ over U compatible with the complex

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S. Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) *f is locally of finite type. Since $S = \mathrm{Spec}(R)$ and $Y = \mathrm{Spec}(R)$.*

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\ldots,0}$.

Lemma 0.2. *Let X be a locally Noetherian scheme over S, $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \overline{A}_2$ works.*

Lemma 0.3. *In Situation ??. Hence we may assume $\mathfrak{q}' = 0$.*

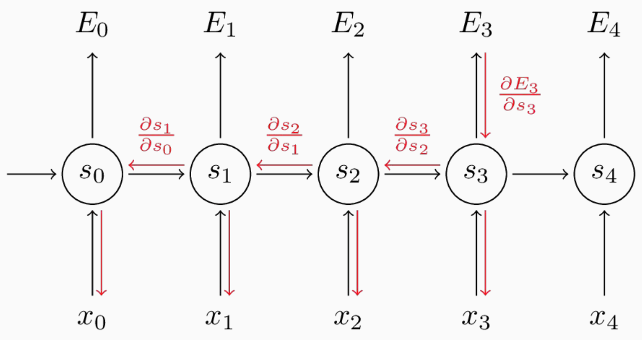
Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

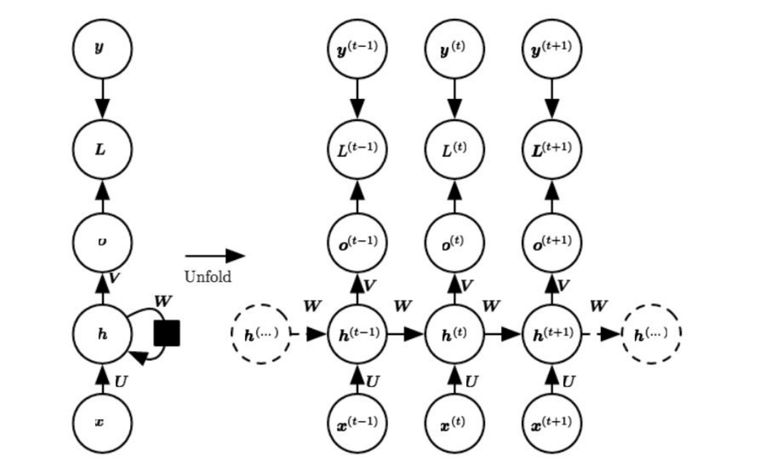
where K is an F-algebra where δ_{n+1} is a scheme over S. □

Training a RNN

Backpropagation through time:



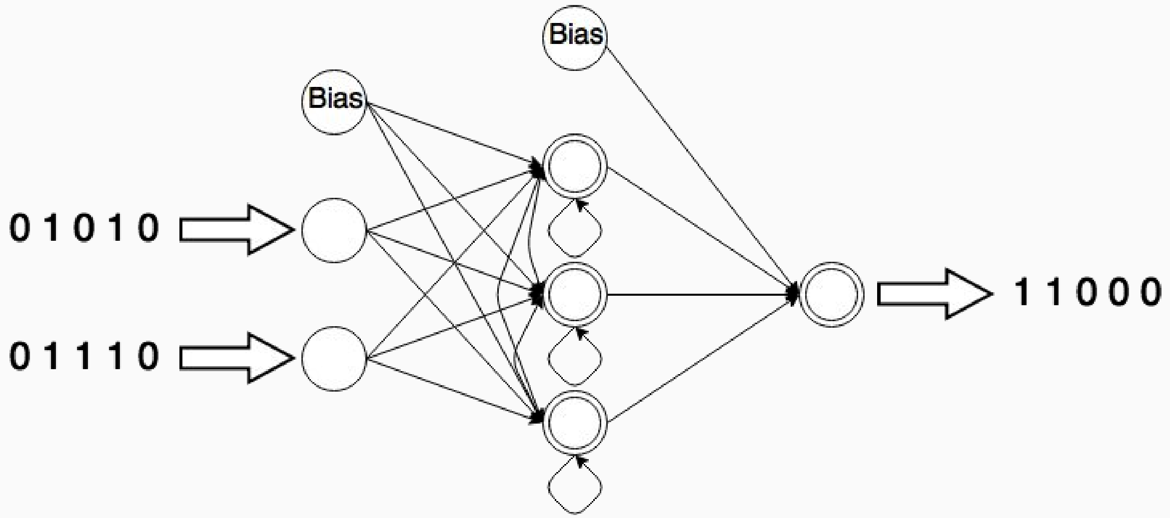
- Minibatches of subsequences
- Update weights after some number of inputs



$$\begin{aligned} \boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} & (10.8) \\ \boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) & (10.9) \\ \boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} & (10.10) \\ \hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)}) & (10.11) \end{aligned}$$

Equations for forward pass

Adding Binary Numbers

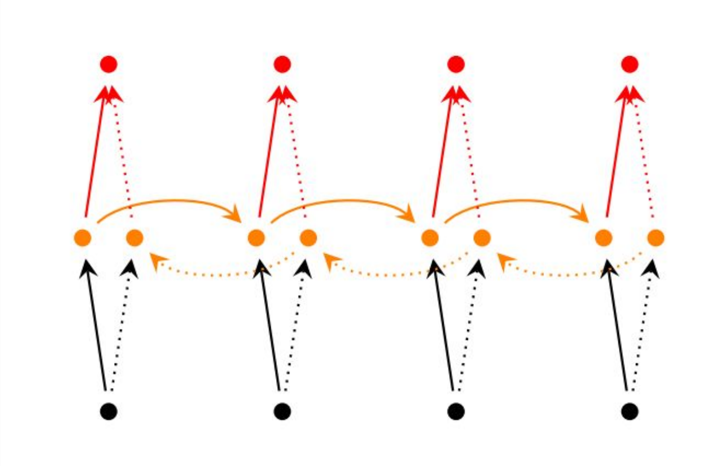


Issues with RNNs

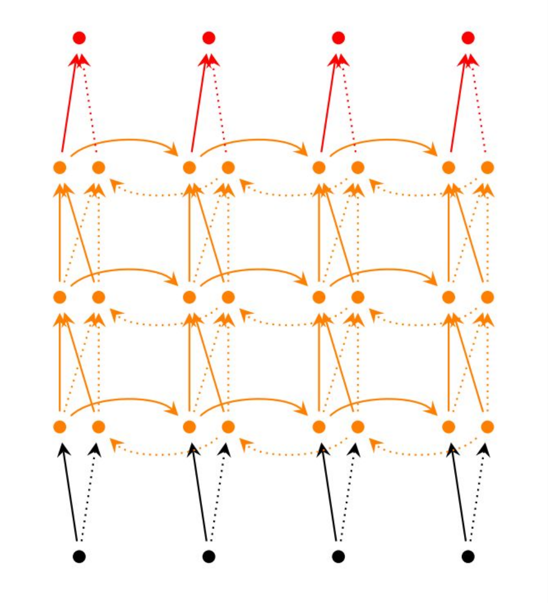
- Training RNNs uses backpropagation through time
- Exploding and vanishing gradient problem similar to that of deep neural networks

Extension to RNNs

Bidirectional RNNs

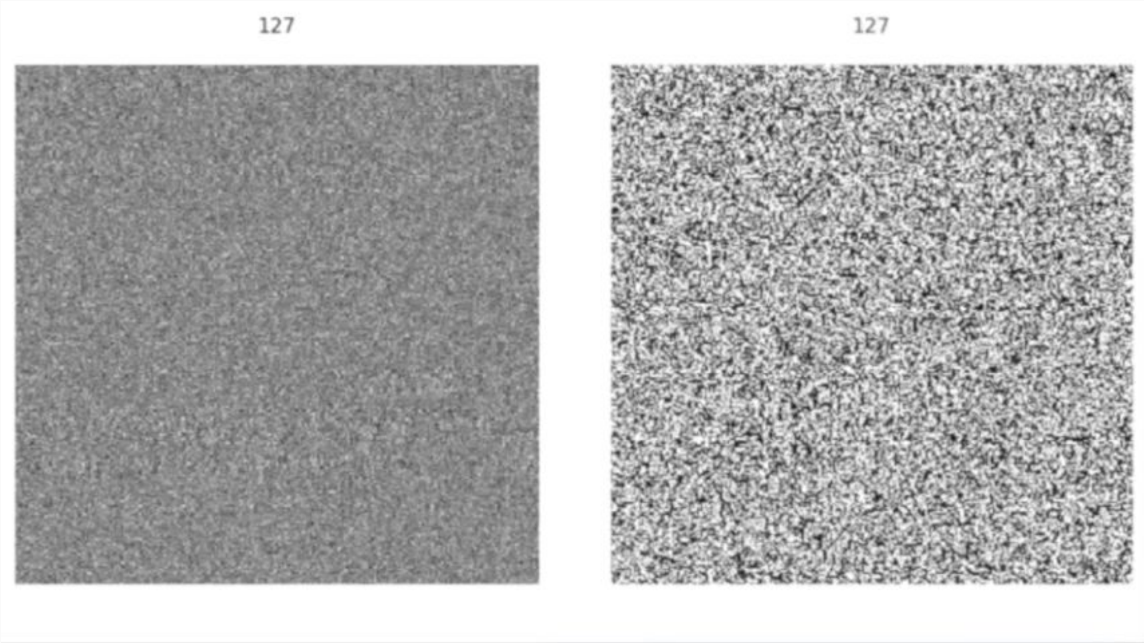


Deep Bidirectional RNNs



Final Word on RNNs

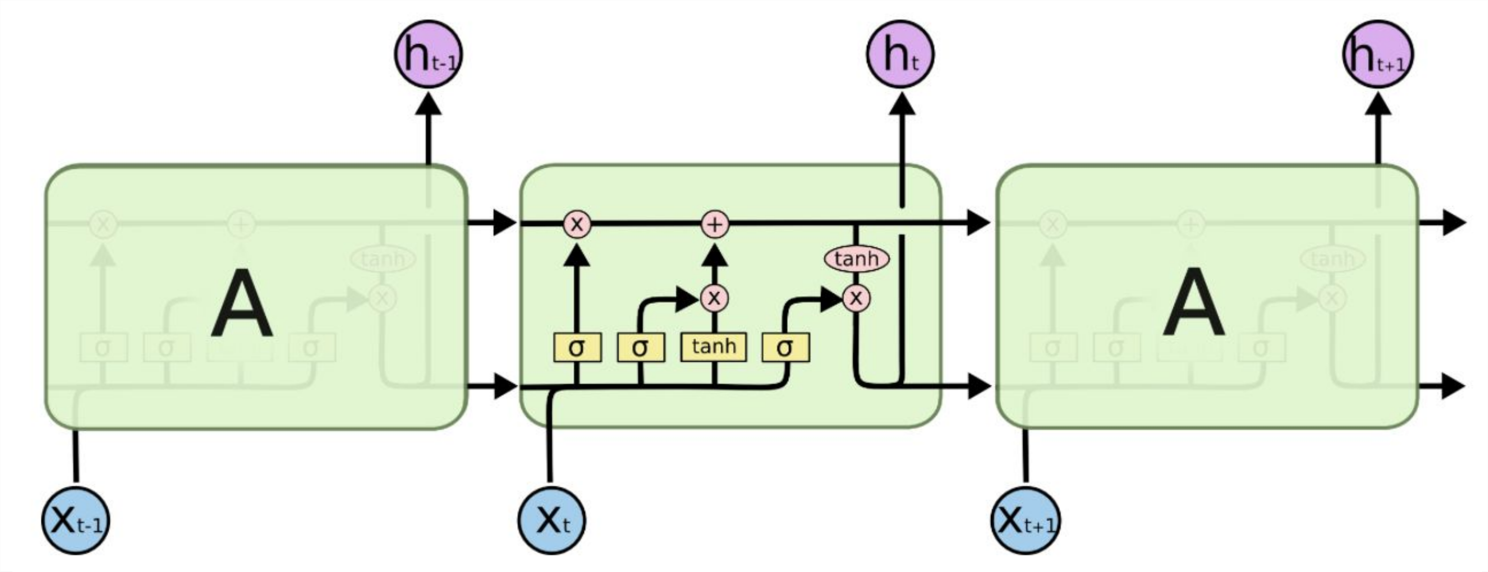
- Turing complete
- Useful for learning sequential data
- Trouble capturing long-term dependencies in practice



LSTMs are a variant RNN that uses “**gating units**” to control information flow adaptively to combat “**gradient vanishing**” *

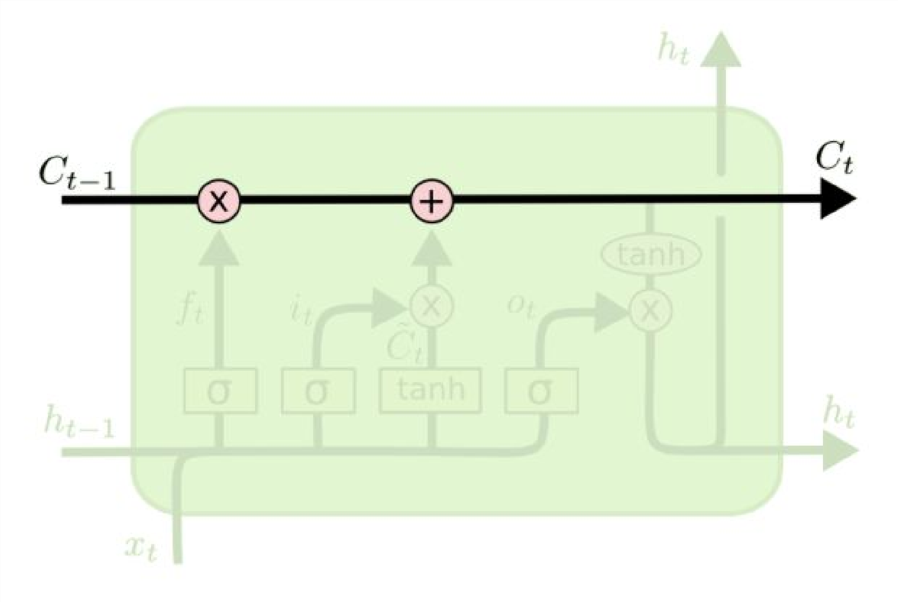
* Pulled from a whiteboard; likely written by a CMU Grad Student.

Long Short Term Memory Networks



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

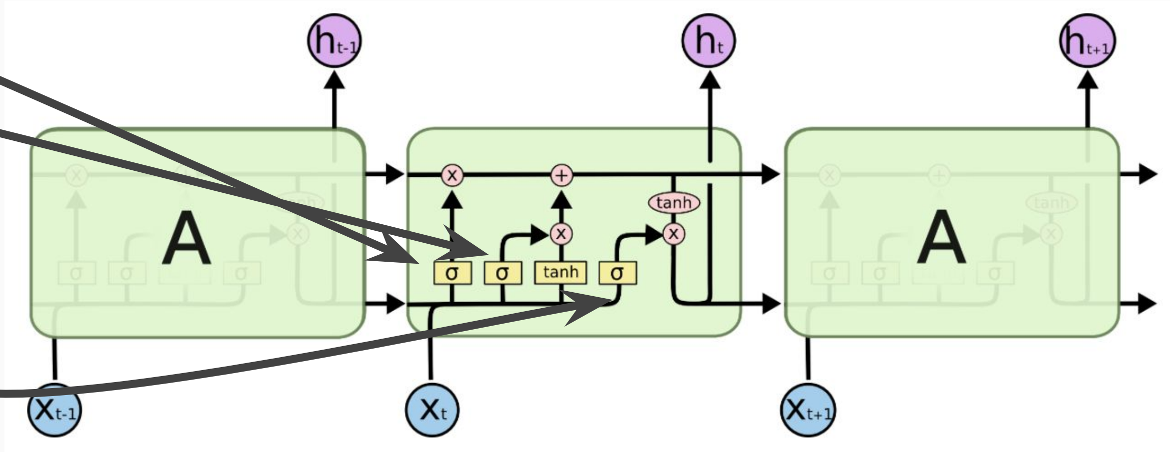
Long Short Term Memory Networks



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory Networks

1. **Forget** Gate
2. **Input** Gate
3. **Output** Gate



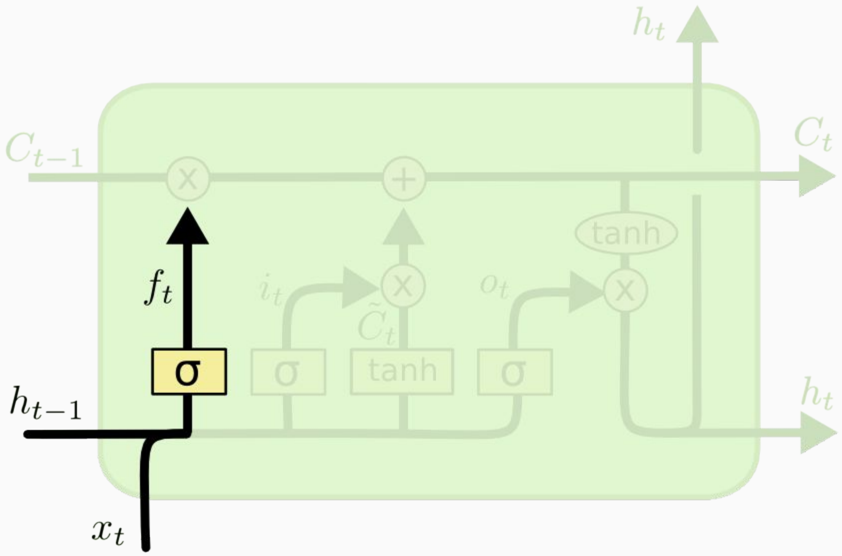
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Forget Gate

1. Takes in h_{t-1} concat x_t and outputs a number between 0 and 1 for each number in the cell state.

2. That output filters the cell state, selectively choosing how much of each node to forget.

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



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

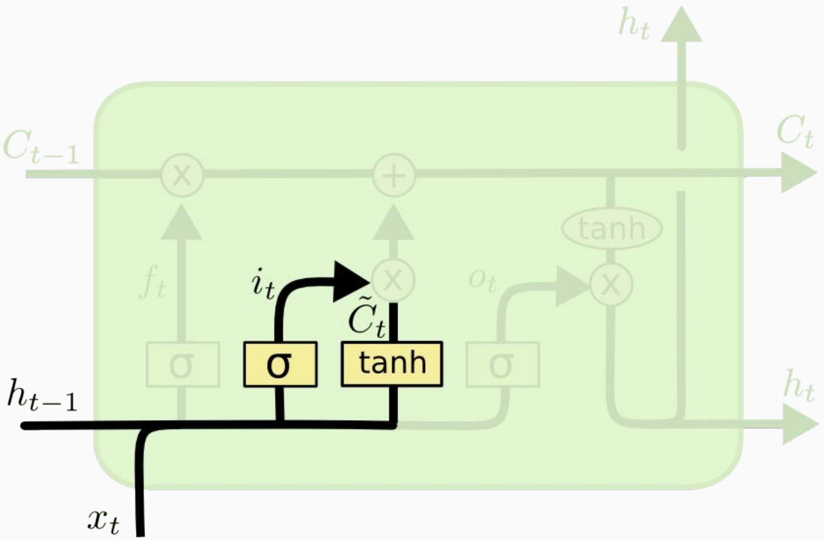
Input Gate

1. Sigmoid Input gate decides what new information can flow in.

2. Tanh layer creates a vector of new candidate values that could be added.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

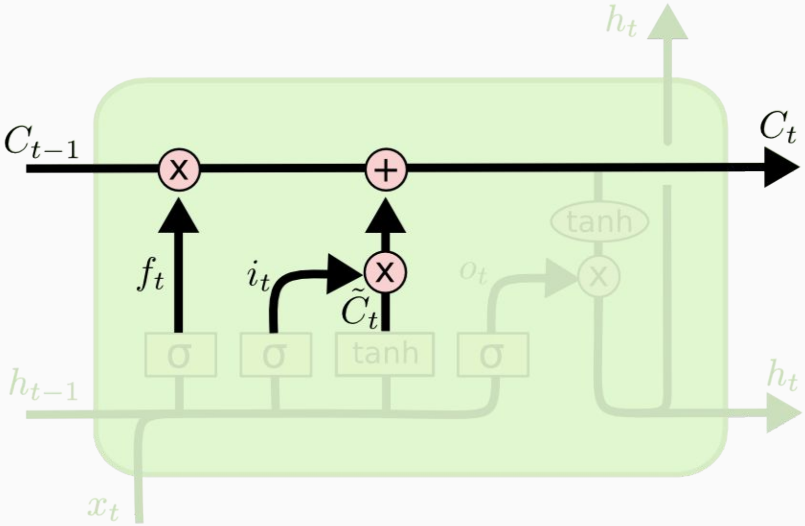
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Putting it together

1. Apply forget gate
2. Compute new candidate cell state
3. Filter in new cell state



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

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

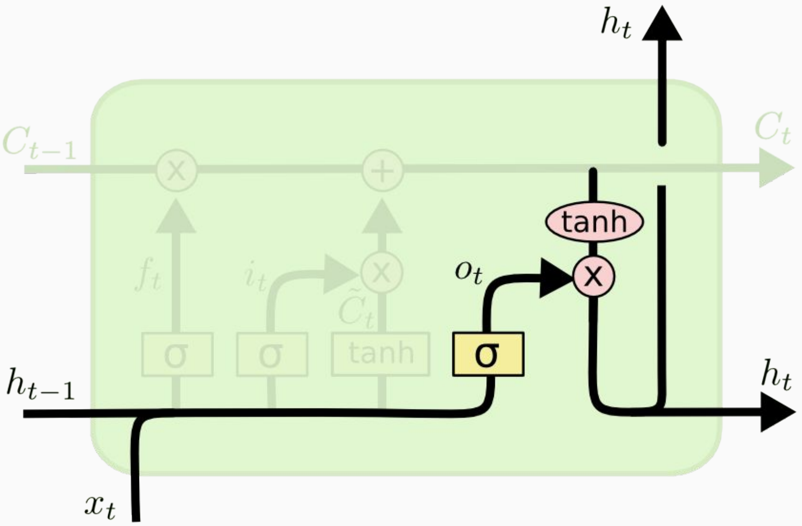
Output Gate

1. Run new cell state through tanh to put values between -1 and 1.

2. Multiply it by the output of the third sigmoid gate to decide what to output.

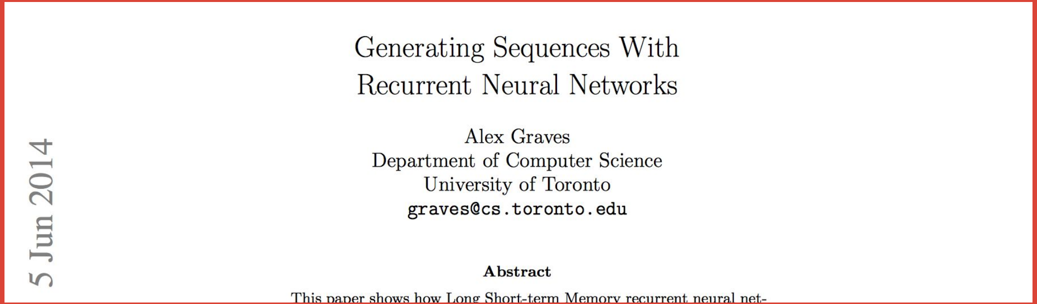
$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM Demo


5 Jun 2014

Generating Sequences With
Recurrent Neural Networks

Alex Graves
Department of Computer Science
University of Toronto
graves@cs.toronto.edu

Abstract

This paper shows how Long Short-term Memory recurrent neural net-

Row 1: Characters as they're recognized

Row 2: States of some memory cells

Row 3: Writing as it's analyzed

Row 4: Gradient backpropagated to inputs from the most active character

https://arxiv.org/pdf/1308.0850.pdf

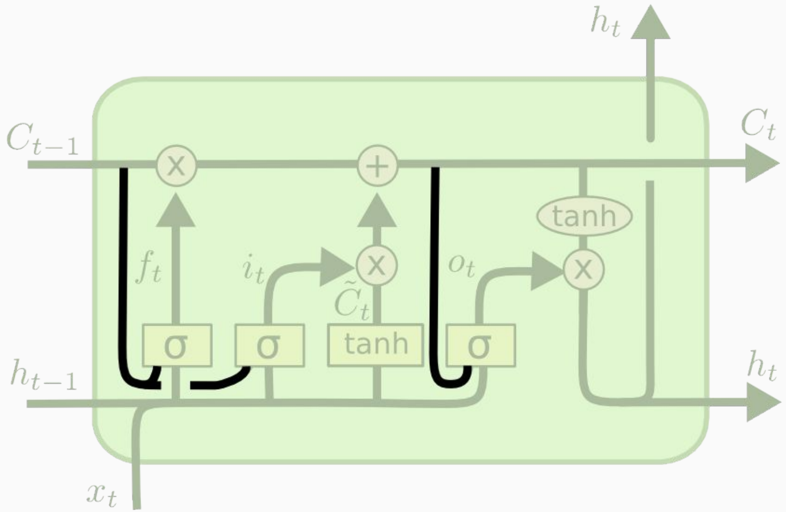
Variations on LSTM

Peephole Connections

1. Pass the gate layers the current cell state.

2. Helps the system learn from the size of the time lags.

$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \ + \ b_o\right)$$



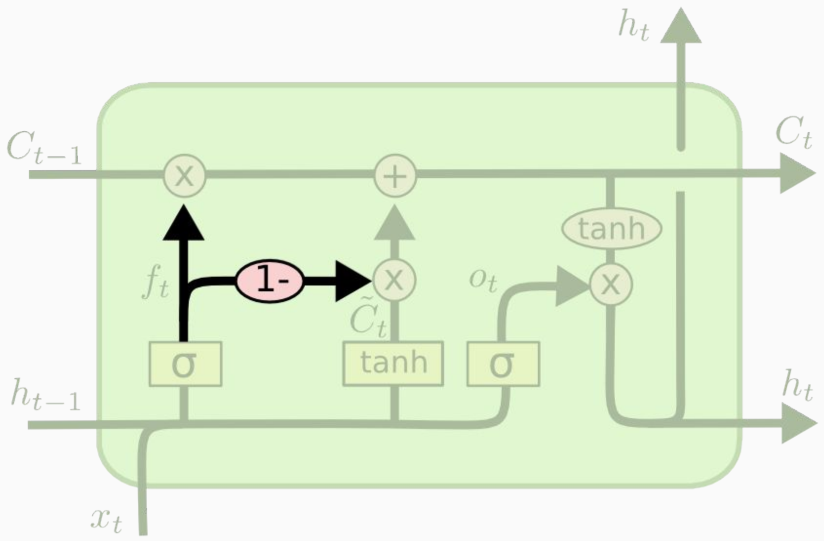
ftp://ftp.idsia.ch/pub/juergen/TimeCount-IJCNN2000.pdf

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Couple Forget/Input Gates

1. Pass the gate layers the current cell state.

2. Simplifies the cell but reduced performance slightly.



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

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated Recurrent Unit (GRU)

1. Combine forget and input gates into an **"update gate"**

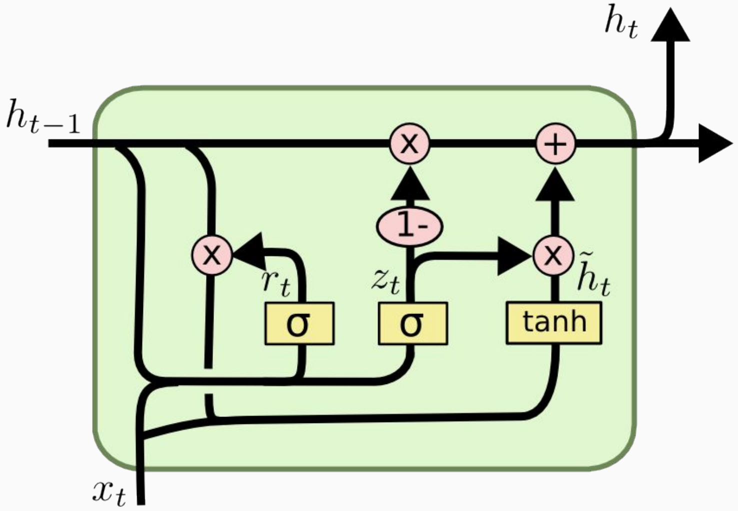
2. In general, equal to LSTMs in performance and faster in training

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

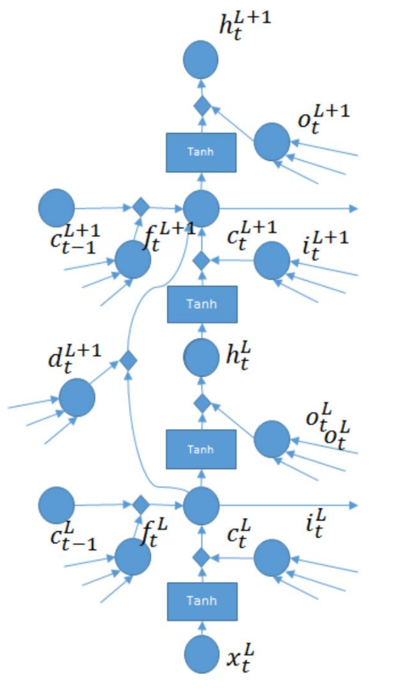
Depth Gated RNNs

1. Adds a fourth gate - the Depth Gate
2. Creates a gated linear connection between lower and upper layer memory cells
3. Relates to highway networks and Grid LSTM

Using the depth gate, a DGLSTM unit can be written as

$$
\begin{aligned}
i_t^{L+1} &= \sigma(W_{xi}^{L+1} x_t + W_{hi}^{L+1} h_{t-1}^{L+1} + W_{ci}^{L+1} c_{t-1}^{L+1}) && (13)\\
f_t^{L+1} &= \sigma(W_{xf}^{L+1} x_t + W_{hf}^{L+1} h_{t-1}^{L+1} + W_{cf} c_{t-1})^{L+1} && (14)\\
d_t^{L+1} &= \sigma(b_d^{L+1} + W_{xd}^{L+1} x_t^{L+1} + W_{cd}^{L+1} \odot c_{t-1}^{L+1} + W_{ld}^{L+1} \odot c_t^{L}) && (15)\\
c_t^{L+1} &= d_t^{L+1} c_t^{L} + f_t^{L+1} \odot c_{t-1} + i_t^{L+1} \odot tanh(W_{xc} x_t + W_{hc} h_{t-1}^{L+1}) && (16)\\
o_t^{L+1} &= \sigma(W_{xo}^{L+1} x_t + W_{ho}^{L+1} h_{t-1} + W_{co}^{L+1} c_t^{L+1}) && (17)\\
h_t^{L+1} &= o_t^{L+1} \odot tanh(c_t^{L+1}) && (18)
\end{aligned}
$$

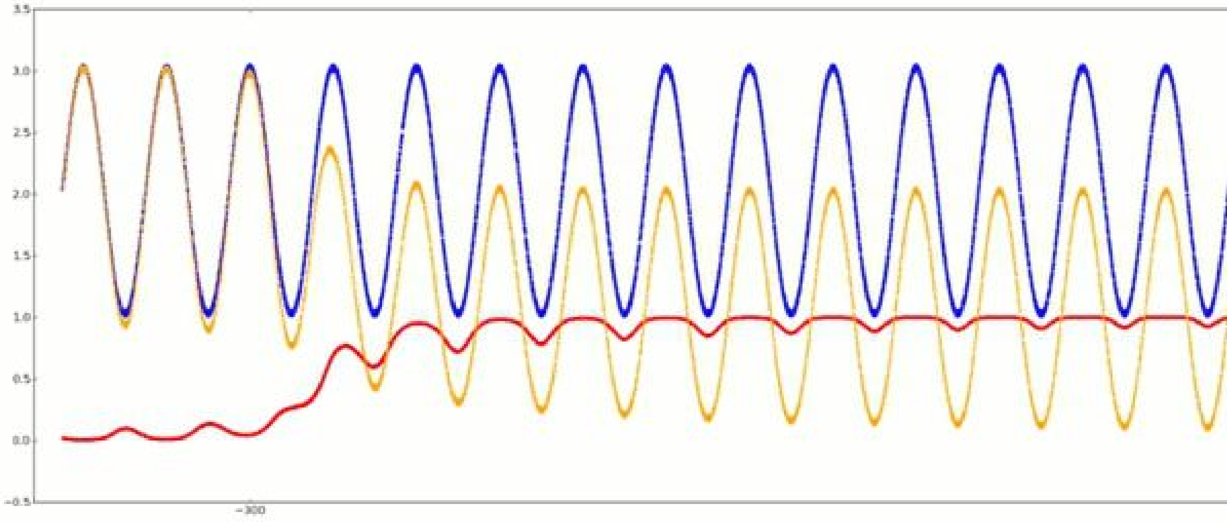
where i_t^{L+1}, f_t^{L+1} o_t^{L+1}, and d_t^{L+1} are the input gate, forget gate, output gate and the depth gate.



Questions

Eric Nie
Devansh Kukreja

LSTMs



LSTM learning to predict next noised sinus value.

Training Signal
LSTM Prediction
Error