Recurrent Neural Nets

David Barber

Gradient Decay/Explosion

- Consider a network with linear transfer function $f(x) = x$.
- For simplicity, assume each layer has the same width (number of neurons).
- For input $\mathbf{x}$ the final layer has value

$$\mathbf{W}_L \mathbf{W}_{L-1} \dots \mathbf{W}_2 \mathbf{x}$$

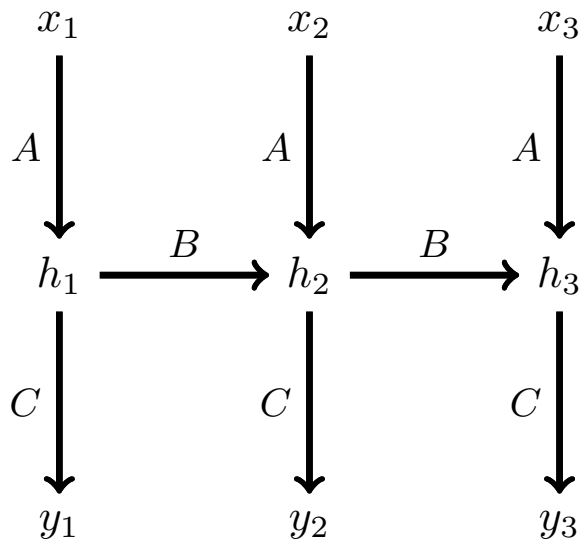
If all the weight matrices are the same $\mathbf{W}_l = \mathbf{W}$, we would have final layer

$$\mathbf{W}^{L-1}\mathbf{x} = \mathbf{E}\Lambda^{L-1}\mathbf{E}^{-1}\mathbf{x}$$

where Λ is the diagonal matrix of eigenvalues of $\mathbf{W}$ and $\mathbf{E}$ is the matrix of eigenvectors.

- Each element λ_i^{L-1} will be exponentially large (if $\lambda_i > 1$) or small (if $\lambda_i < 1$). Only elements with $\lambda_i = 1$ will remain well scaled.
- This suggests that, unless initialised very carefully, gradients may become either zero of infinite, leading to significant difficulties in training.
- This is particularly a problem is very deep or recurrent nets.

Recurrent Nets



- RNNs are used in timeseries applications
- The basic idea is that the hidden units at time h_t (and possibly output y_t) depend on the previous state of the network $h_{t-1}, x_{t-1}, y_{t-1}$ for inputs x_t and outputs y_t.
- In the above network, I 'unrolled the net through time' to give a standard NN diagram.
- I omitted the potential links from x_{t-1}, y_{t-1} to h_t.

Mitigating Memory Decay

- Someone shows you a picture of a dog and asks you to commit this to memory.
- They then show you a sequence of other images.
- If a new image is a dog, you should show the previous image of the dog at the output and store the new dog image to memory.
- This would be difficult to do with a traditional recurrent net since typically information is lost from timestep to timestep.
- Here you need to also explicitly store some information and retrieve it at a potentially much later timepoint.
- Thus we have the question of how to commit things to memory and potentially retrieve them later.

Structured RNNs

- It's worth noting that everything we do here will be in the context of RNNs.
- Each model (simple memory model, LSTM) is a *specially constrained* RNN.
- Theoretically, we can do all of this with a conventional unconstrained RNN. In practice, however, training such models to store long term dependencies has proven difficult. Life is much easier if we use specially constrained models that are biased towards solving long-term memory storage.

Mitigating Memory Decay

- Let's consider how we might solve the previous 'dog' problem.
- One way to solve this is to have units that represent a 'running memory' $m(t)$.
- First we need to decide if the current image $x(t)$ is a 'dog' and should therefore be stored. Using a matrix W_{in} that is a row vector and scalar bias B_{in} the logistic sigmoid outputs a value $\alpha(t) \in [0, 1]$ that represents whether the image $x(t)$ is a dog and should therefore be committed to memory. We can therefore define a 'gate':

$$\alpha(t) = \sigma\left(W_{in}x(t) + B_{in}\right)$$

- We can now update the memory vector $m(t)$ using

$$m(t) = \alpha(t)x(t) + (1 - \alpha(t))m(t-1)$$

  If $\alpha(t)$ is close to 1, we essentially replace the memory with $x(t)$. For $\alpha(t)$ close to zero, we ignore $x(t)$ and essentially copy the previous value of the memory $m(t-1)$ to the current memory $m(t)$.
- We can then simply output

$$y_{out}(t) = \alpha(t)m(t-1)$$

Mitigating Memory Decay

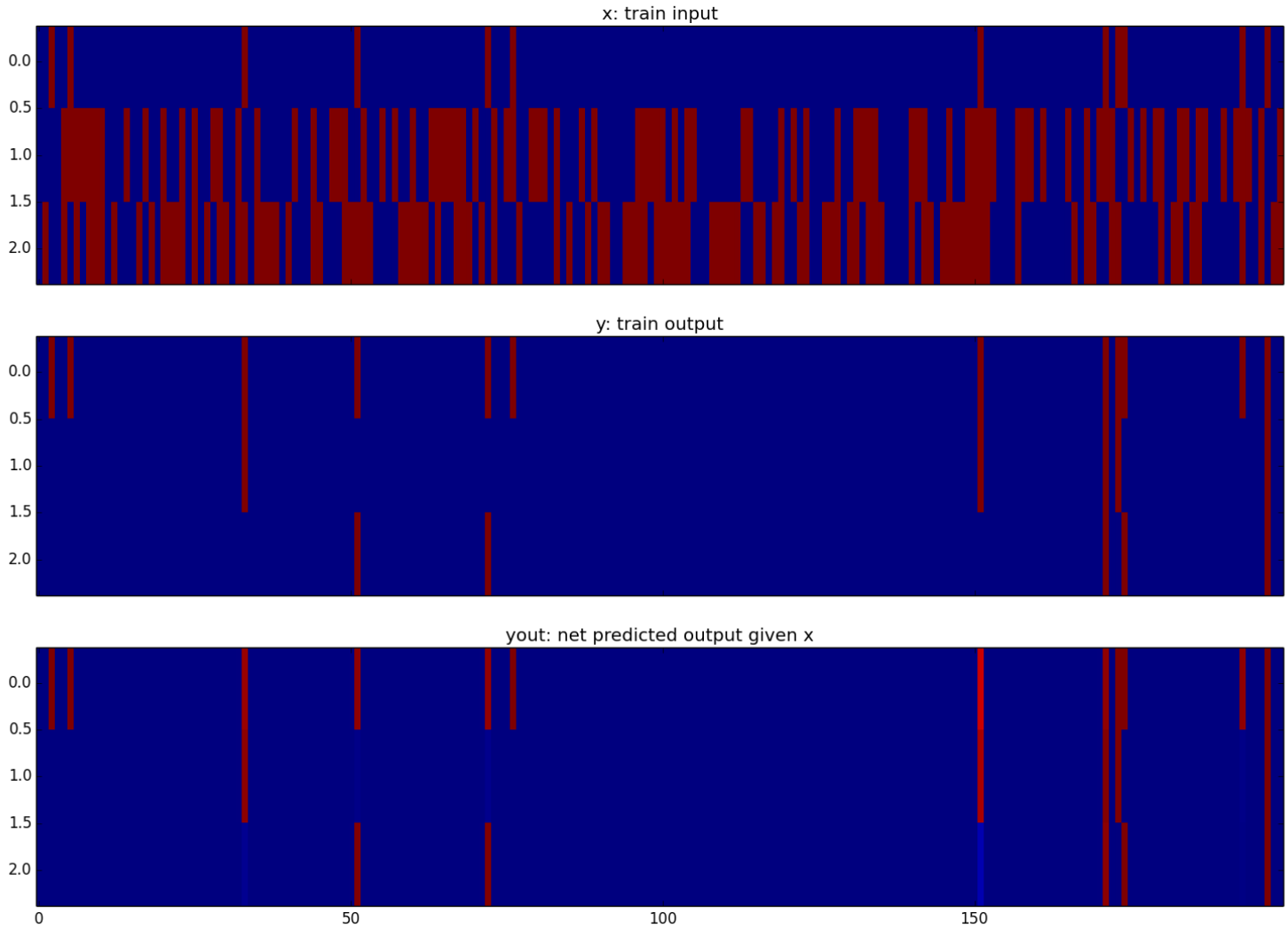
- The key idea is that we have made a structured form of RNN in which units in the network have specific roles. In this case the hidden units at time t have the role of a gate and a memory store:

$$h(t) = (\alpha(t), m(t))$$

- Critically, the update $h(t) \rightarrow h(t+1)$ contains a copying mechanism.
- Provided the $\alpha(t)$ are very close to 1, then the memory will be retained across many timepoints.
- Note, however, that eventually the memory would still decay (hence it is still 'short-term', just longer than rapid Markovian memory decay. However, this mechanism is a way to construct an architecture that is well suited to storing patterns over longer timescales.
- Technically, the reason this works is that for $\alpha(t)$ close to 1, then the gradient does not decay significantly over time.

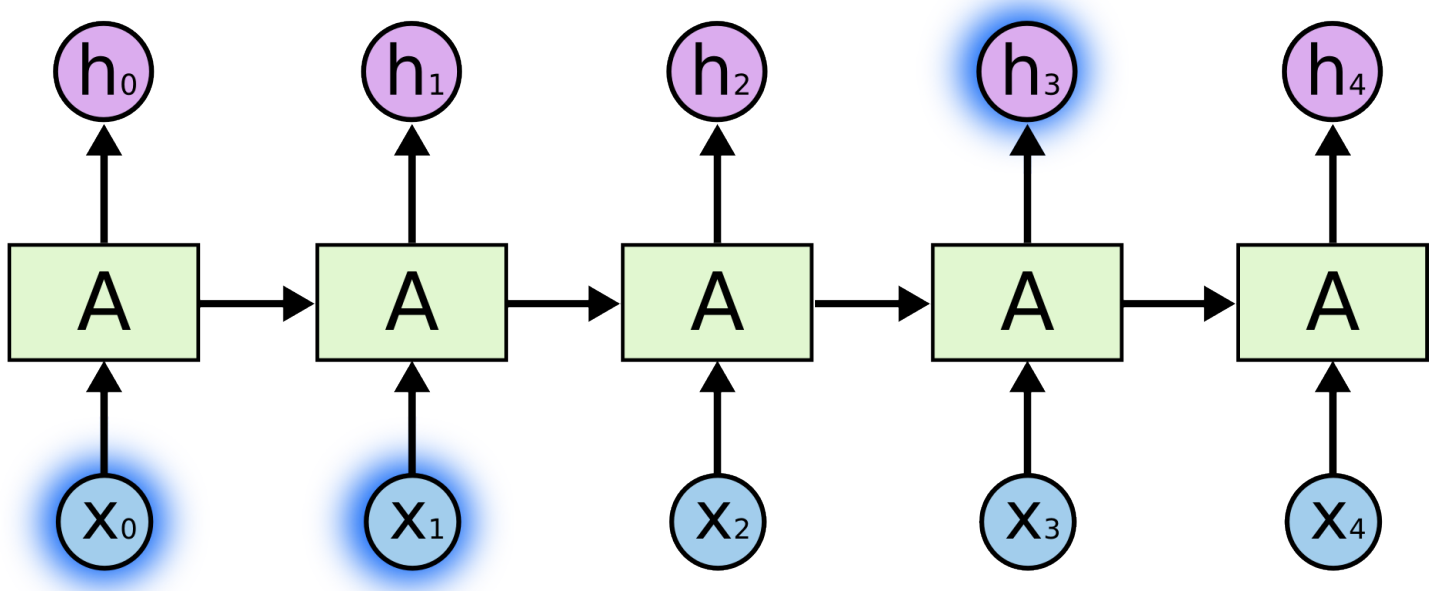
Simple Memory Storage

The task: If either x_1 or x_2 is 1 (or both), then store the current vector $x(t)$ to the memory and output the previous memory `DemoSimpleMemory.jl`:



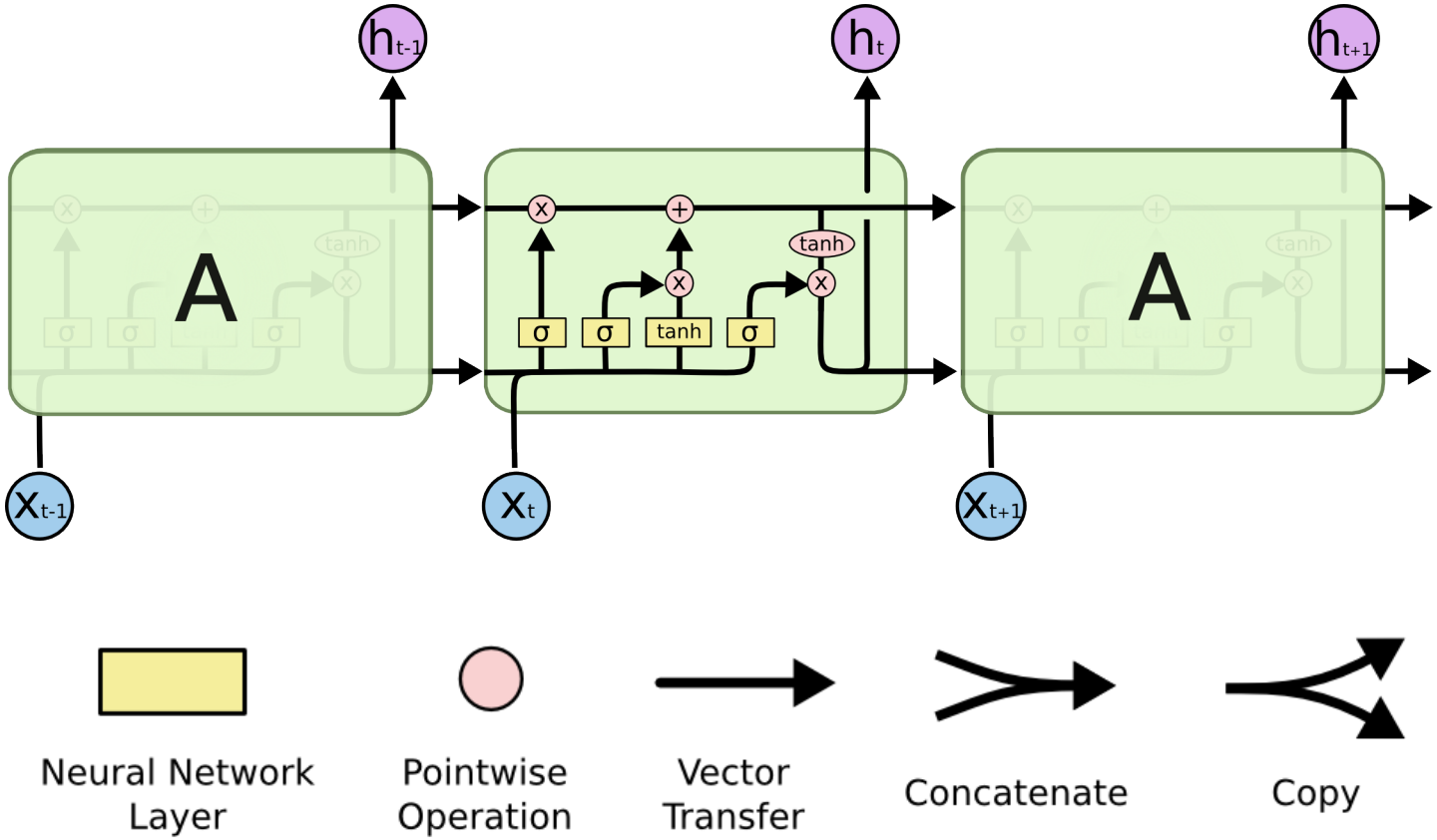
Long Short Term Memory

- LSTM is an extension of the previous simple memory model
- `http://colah.github.io/posts/2015-08-Understanding-LSTMs/` gives a nice tutorial, from where I've taken some of the images.



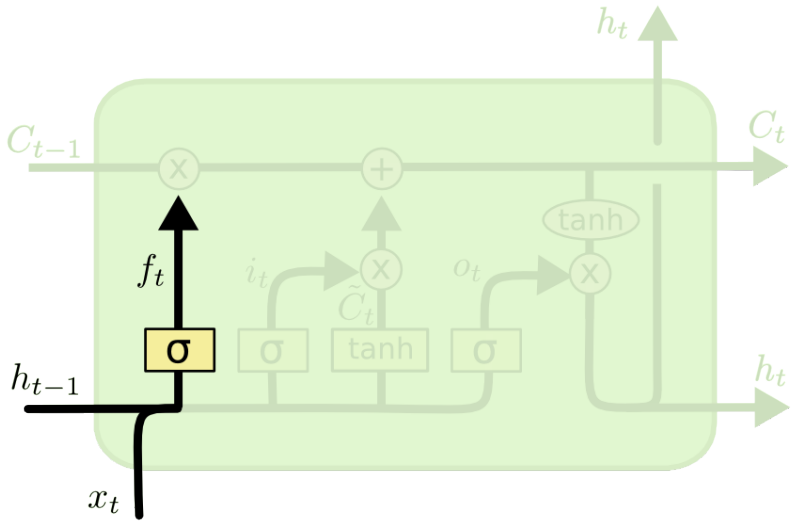
- In this notation, x_t is an input at time t, A is the state of the hidden nodes at each time and h_t is the output.
- We can train an RNN to predict the next word in the sequence. For a sentence like "I grew up in France I speak fluent *French*", predicting that the final word is French requires to store the knowledge "France", encountered many timesteps before.

Long Short Term Memory



- The forget gate decides whether to keep the old memory, or replace it.
- The input gate decides whether the current input should be added to the memory C (the upper horizontal line).
- The output gate decides whether the the current memory should be sent to the output.

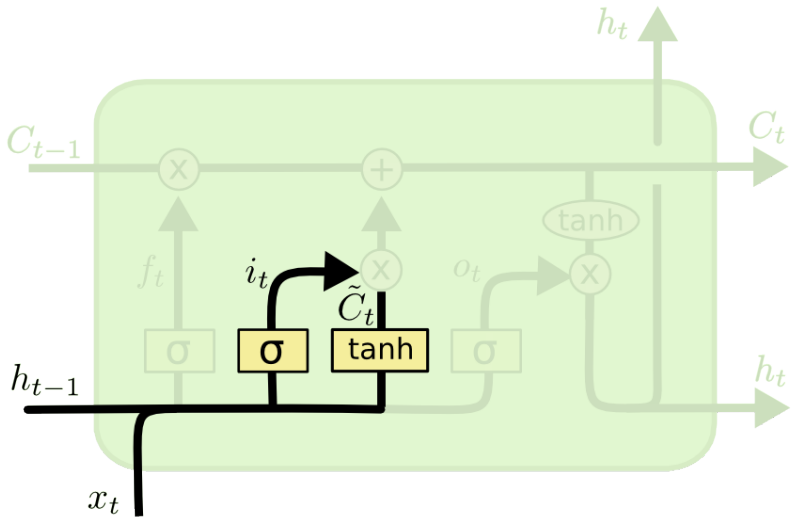
Long Short Term Memory: Forget Gate



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

- We first concatenate the previous output and current input.
- The value of $f_t \in [0, 1]$ then decides whether to forget ($f_t = 1$) or keep ($f_t = 0$) the current memory C_{t-1}.

Long Short Term Memory: Input Gate

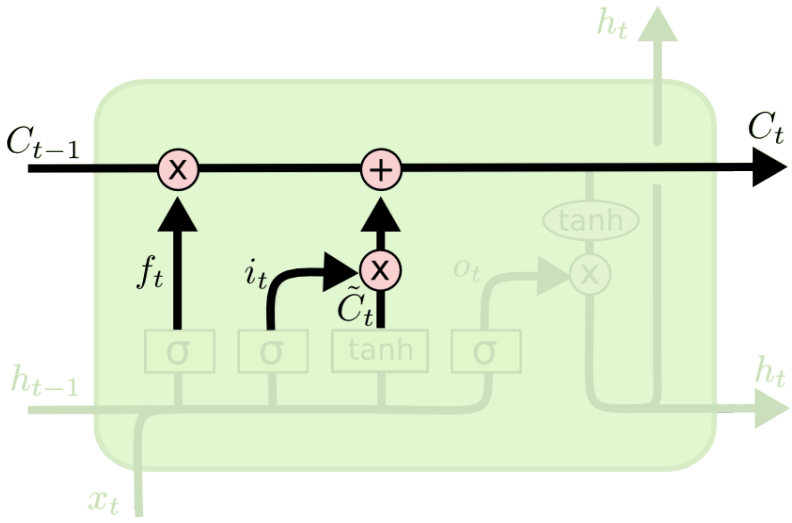


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

- The input gate decides (on the basis of x_t and the previous output h_{t-1} whether something should be stored in the memory at this timestep.
- $\tilde{C}_t$ is what we will store.

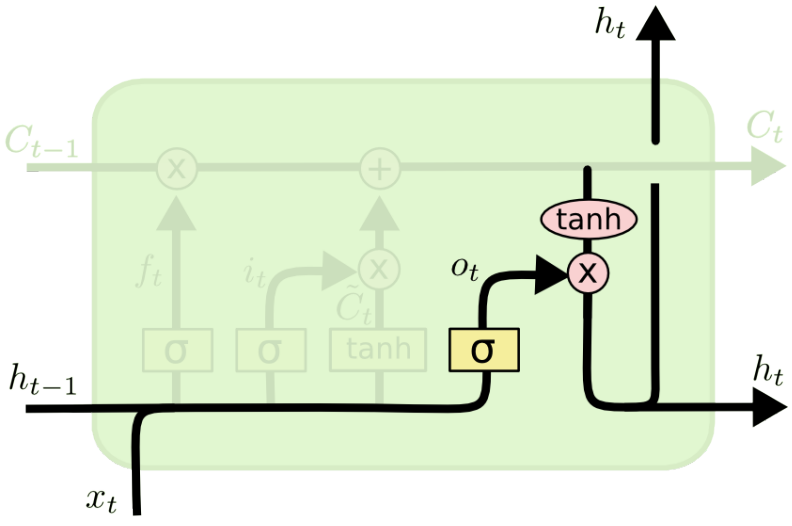
Long Short Term Memory: Memory Cell



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

- We then update the memory cell.
- We forget previous values and commit new values to the memory.

Long Short Term Memory: Output Gate

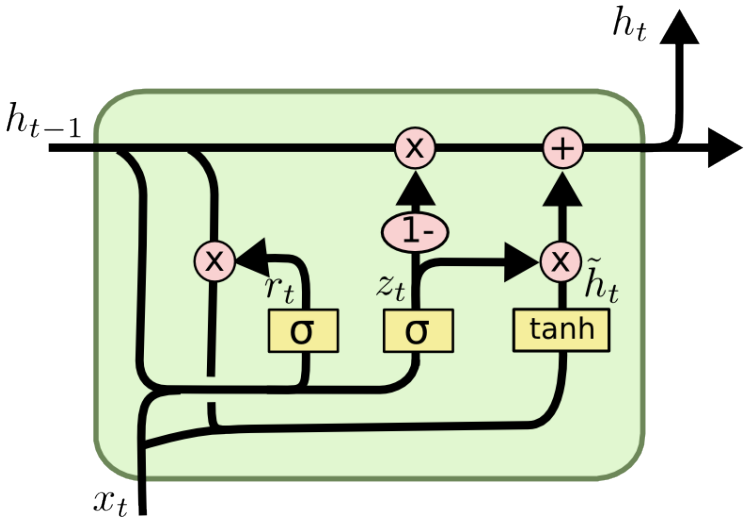


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

- The output gate decides whether to output the memory cell at this timestep.
- h_t is what we output. One could easily modify this output if the $\tanh$ constraint is not appropriate.

Gated Recurrent Net



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

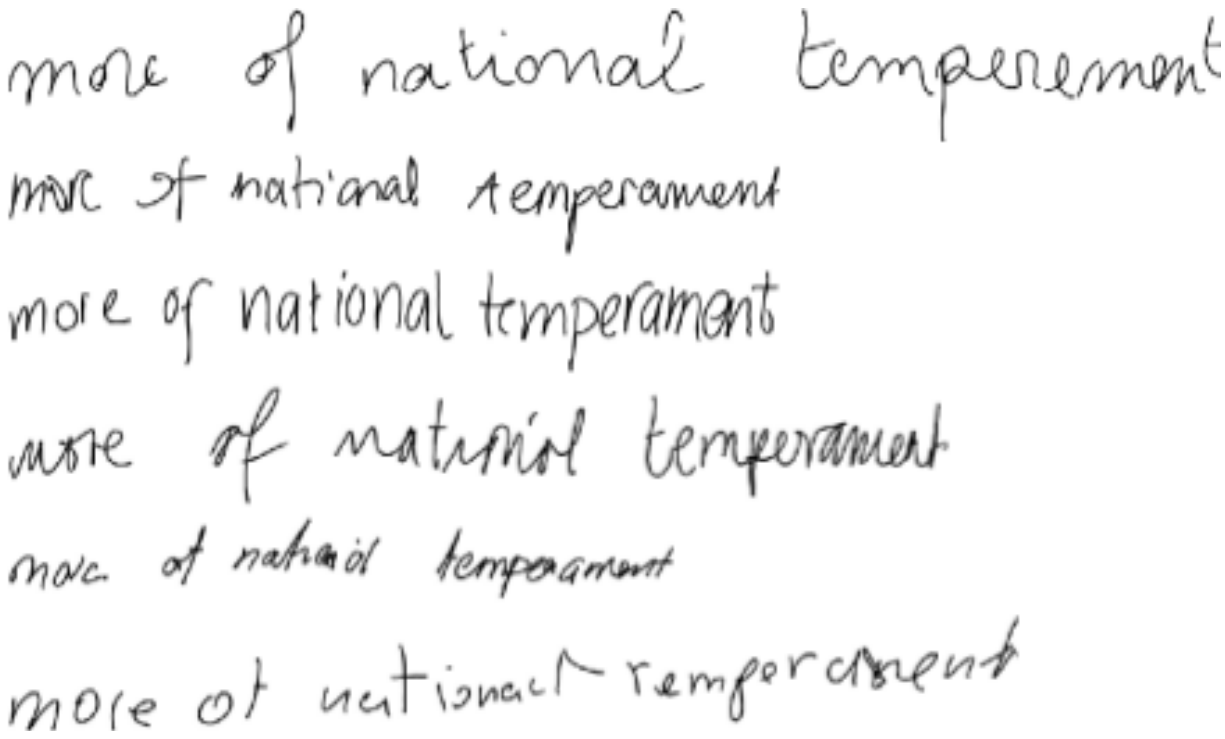
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

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

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

- The GRU is a simpler architecture that also works well.
- Here h_t plays the role of the memory.
- z_t is a factor that determines whether to forget the old and commit a new memory $\tilde{h}_t$.
- What you store $\tilde{h}_t$ depends on the current input and possibly an amount (gated by r_t) of the previous memory h_{t-1}.
- Note that we can also use multiple LSTM or GRU units in a net. Infinitely many architectures for this – eg Grid LSTM, stacked LSTM, *etc.*

Handwriting Generation using an LSTM RNN

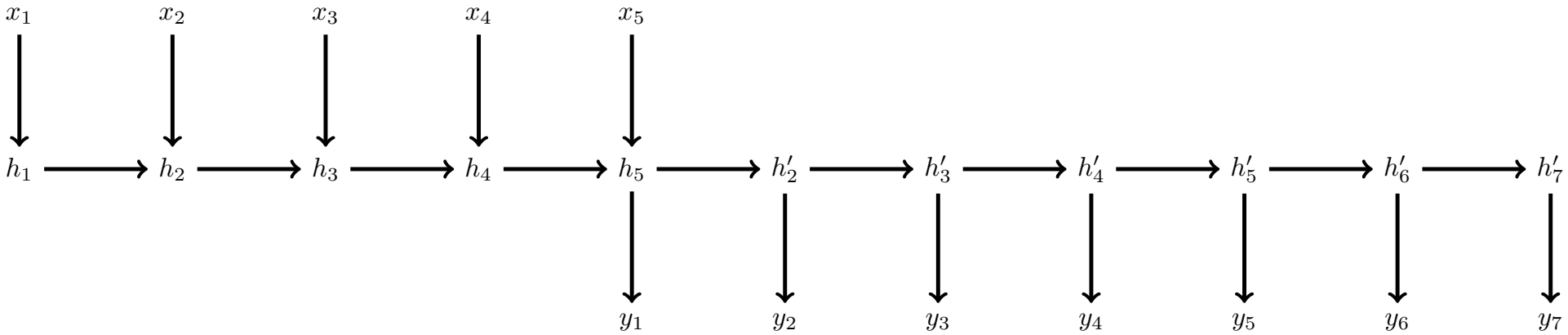


- Top line is real handwriting.
- Some generated examples (from Alex Graves' work).
- The LSTM enables long term consistency in the generated examples.

Sequence to Sequence models

- Seq2Seq models take an input sequence $x_1, \ldots, x_M$ and output a sequence $y_1, \ldots, y_N$.
- An important consideration is that the input and output sequences *are potentially of different length*.
- An example is machine translation:
  $x_{1:4}$: `Stop tickling me!`
  $y_{1:5}$: `Hör auf mich zu kitzeln!`
- There are various approaches based on using a RNN to convert the input sequence to a fixed length vector.
- This vector is then used by another RNN to generate the output sequence. See Sutskever etal "Sequence to Sequence Learning using Neural Networks".

A Sequence to Sequence Model



$x_1 = \texttt{stop}, x_2 = \texttt{tickling}, x_3 = \texttt{me}, x_4 = \texttt{!}, x_5 = \texttt{EndOfSentence}$

$y_1 = \texttt{Hör}, y_2 = \texttt{auf}, y_3 = \texttt{mich}, y_4 = \texttt{zu}, y_5 = \texttt{kitzeln}, y_6 = \texttt{!}, y_7 = \texttt{EndOfSentence}$

- In this model the hidden node h_5 contains all the relevant information in the source sentence.
- We then use a different net to generate the translation sequence.
- One can also add inputs to recursively generate the translation, so that y_{t-1} is an additional input to h'_t.
- In practice it is also useful to reverse the input sequence order.
- Sutskever etal used a 4-layer LSTM in which the outputs of each LSTM cell is used as the input of another.

Translation

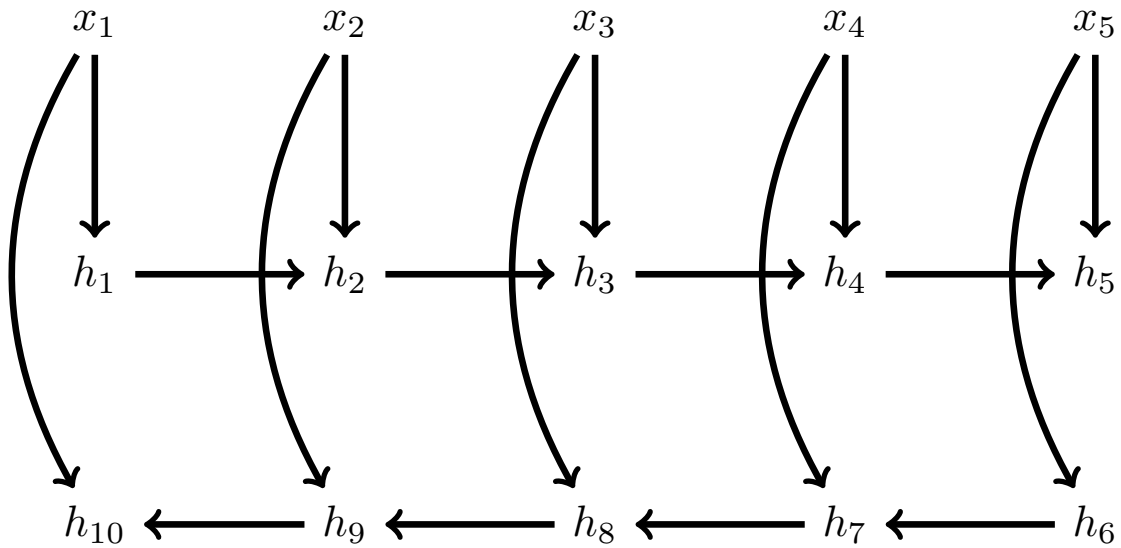
Type	Sentence
Our model	Ulrich UNK , membre du conseil d' administration du constructeur automobile Audi , affirme qu' il s' agit d' une pratique courante depuis des années pour que les téléphones portables puissent être collectés avant les réunions du conseil d' administration afin qu' ils ne soient pas utilisés comme appareils d' écoute à distance .
Truth	Ulrich Hackenberg , membre du conseil d' administration du constructeur automobile Audi , déclare que la collecte des téléphones portables avant les réunions du conseil , afin qu' ils ne puissent pas être utilisés comme appareils d' écoute à distance , est une pratique courante depuis des années .
Our model	" Les téléphones cellulaires , qui sont vraiment une question , non seulement parce qu' ils pourraient potentiellement causer des interférences avec les appareils de navigation , mais nous savons , selon la FCC , qu' ils pourraient interférer avec les tours de téléphone cellulaire lorsqu' ils sont dans l' air " , dit UNK .
Truth	" Les téléphones portables sont véritablement un problème , non seulement parce qu' ils pourraient éventuellement créer des interférences avec les instruments de navigation , mais parce que nous savons , d' après la FCC , qu' ils pourraient perturber les antennes-relais de téléphonie mobile s' ils sont utilisés à bord " , a déclaré Rosenker .
Our model	Avec la crémation , il y a un " sentiment de violence contre le corps d' un être cher " , qui sera " réduit à une pile de cendres " en très peu de temps au lieu d' un processus de décomposition " qui accompagnera les étapes du deuil " .
Truth	Il y a , avec la crémation , " une violence faite au corps aimé " , qui va être " réduit à un tas de cendres " en très peu de temps , et non après un processus de décomposition , qui " accompagnerait les phases du deuil " .

Table 3: A few examples of long translations produced by the LSTM alongside the ground truth translations. The reader can verify that the translations are sensible using Google translate.

- Some translations (original English sentence not shown).
- These are examples of long sentences to demonstrate the ability of the model.

Bidirectional RNNs

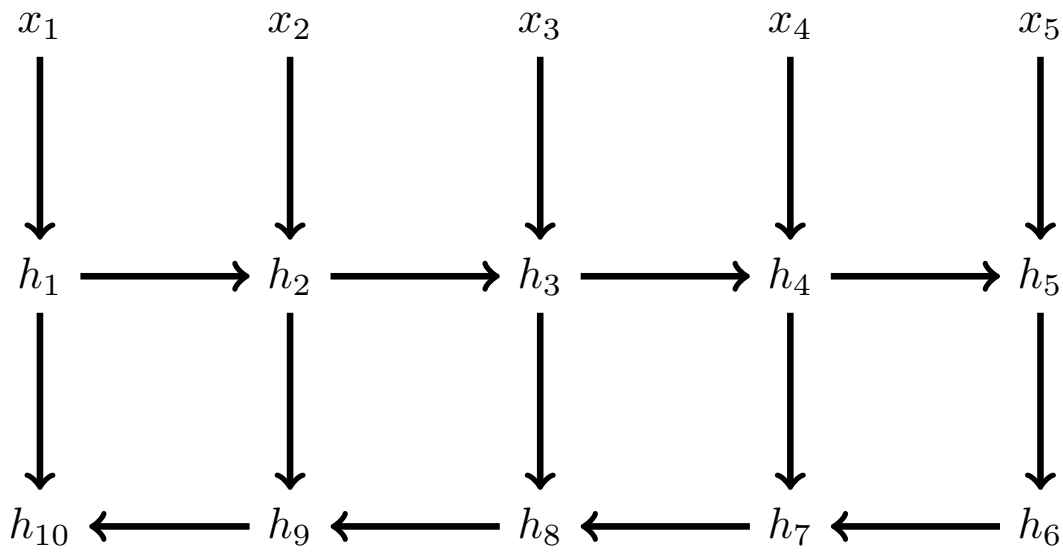
- A potential criticism of the previous Seq2Seq model is that is biased towards capturing the latter part of the sentence.
- In bidirectional models, the encoder model traverses the input sequence in both directions. One example architecture is:



- The joint state (h_5, h_{10}) in this case could be used to represent the input sequence $x_{1:5}$ and used as input to a decoder RNN.

Bidirectional RNNs

- An alternative bidirectional model:



- The state of h_{10} in this case could be used to represent the input sequence $x_{1:5}$ and used as input to a decoder RNN.