Recurrent Neural Networks

HALMSTAD
UNIVERSITY

hh.se
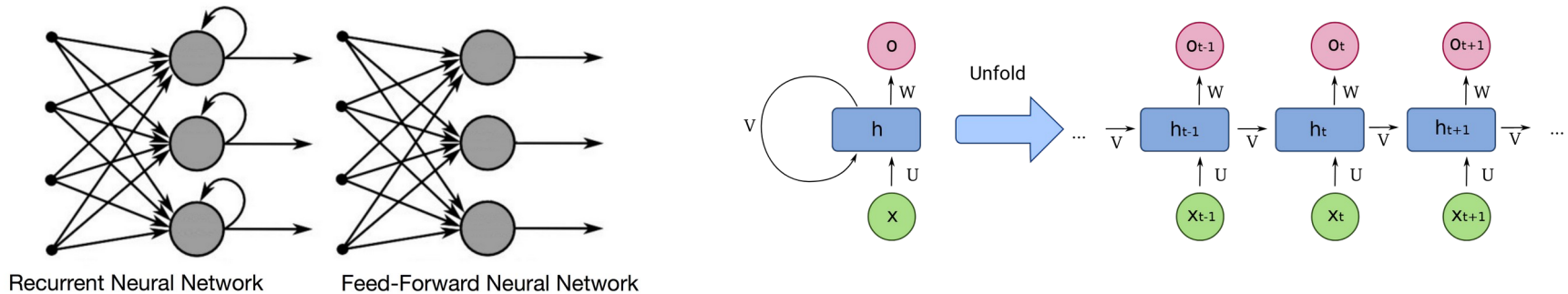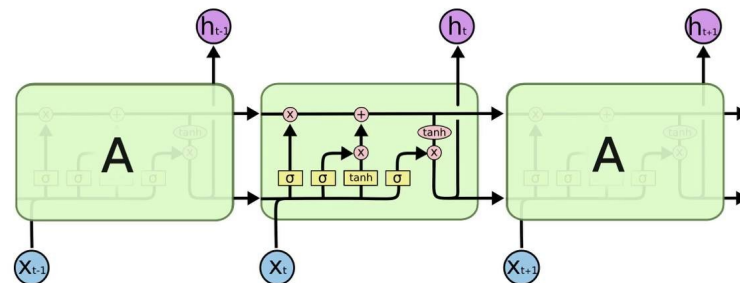
Content of this lecture

1. Time delay network
2. Recurrent networks, the Elman type
3. Recurrent networks, the general type
4. Recurrent networks, LSTM and GRU types
5. LSTM architectures …
6. Combination of architectures
7. Some examples in between ...

Recurrent Neural Networks

(Common google images when searching for RNN)

Recurrent Neural Network   Feed-Forward Neural Network
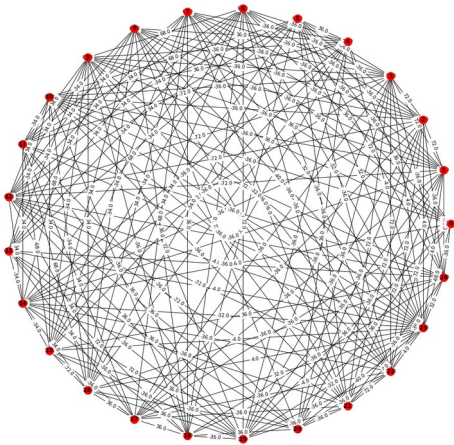
Unfold

Long-Short Term Memory module: LSTM

long-short term memory modules used in an RNN

Neural Network Layer   Pointwise Operation   Vector Transfer   Concatenate   Copy

http://colah.github.io/posts/2015-08-Understanding-LSTMs/   Eugenio Culurciello © 2016

hh.se

HALMSTAD UNIVERSITY

But these are also recurrent networks

Hopfield model

Echo state network

HALMSTAD
UNIVERSITY
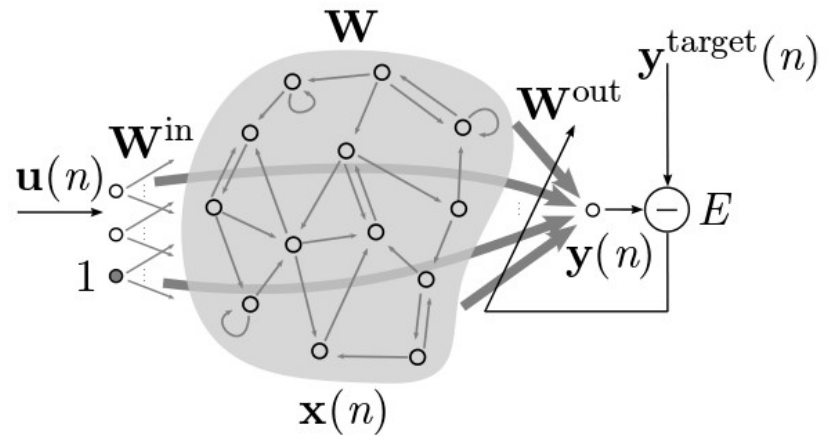
Recurrent Neural Networks

- Common for all neural network models we have studied so far is the lack of feedback connections. We will now study networks with such connections!

- Recurrent networks are typically used when we are dealing with sequence data. It can be text data, speech data, image data or numerical times series data coming from eg. sensors or stock markets. And combinations of all these!

- The feedback connections are used to capture the short and long term temporal dependencies in the data.
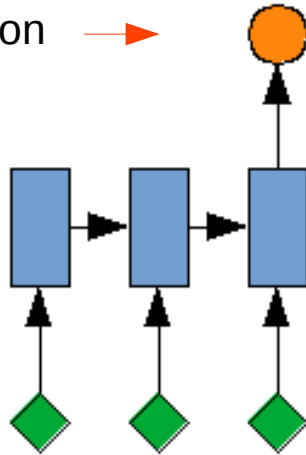
What is it that recurrent networks offer?

HALMSTAD
UNIVERSITY

Sofar, one-to-one problems!

classification
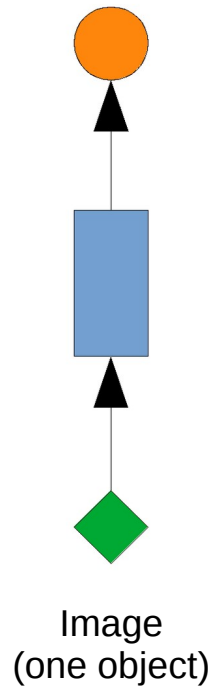
Image
(one object)

With recurrent networks

many-to-one

one-to-many

eg. classification

eg. caption

eg. text (sequence of words)

eg. image

HALMSTAD
UNIVERSITY
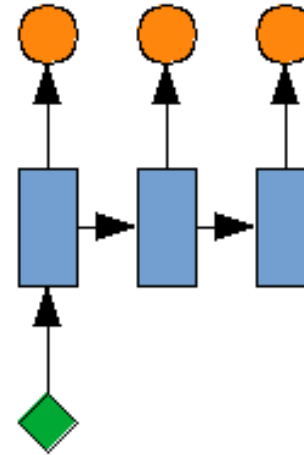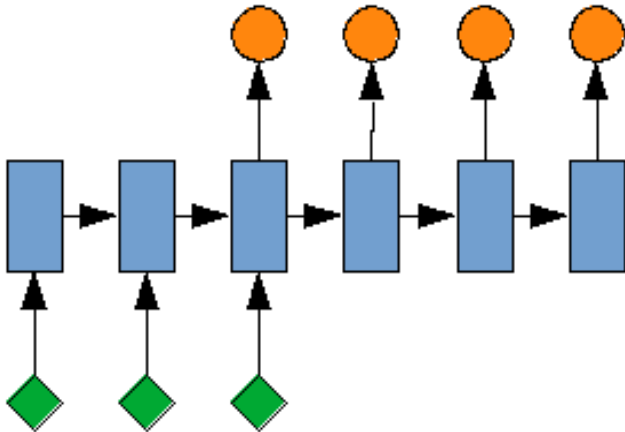
many-to-many

many-to-many

eg. another
sequence

eg. text

eg. text

eg. sequence

HALMSTAD
UNIVERSITY
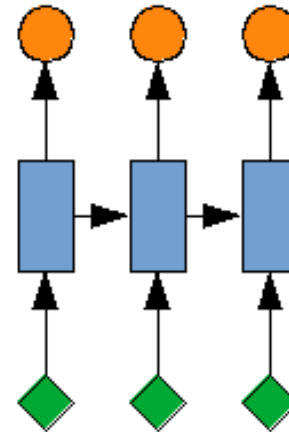
We will however start simple, by
using an ordinary MLP for a specific
task using sequence data.
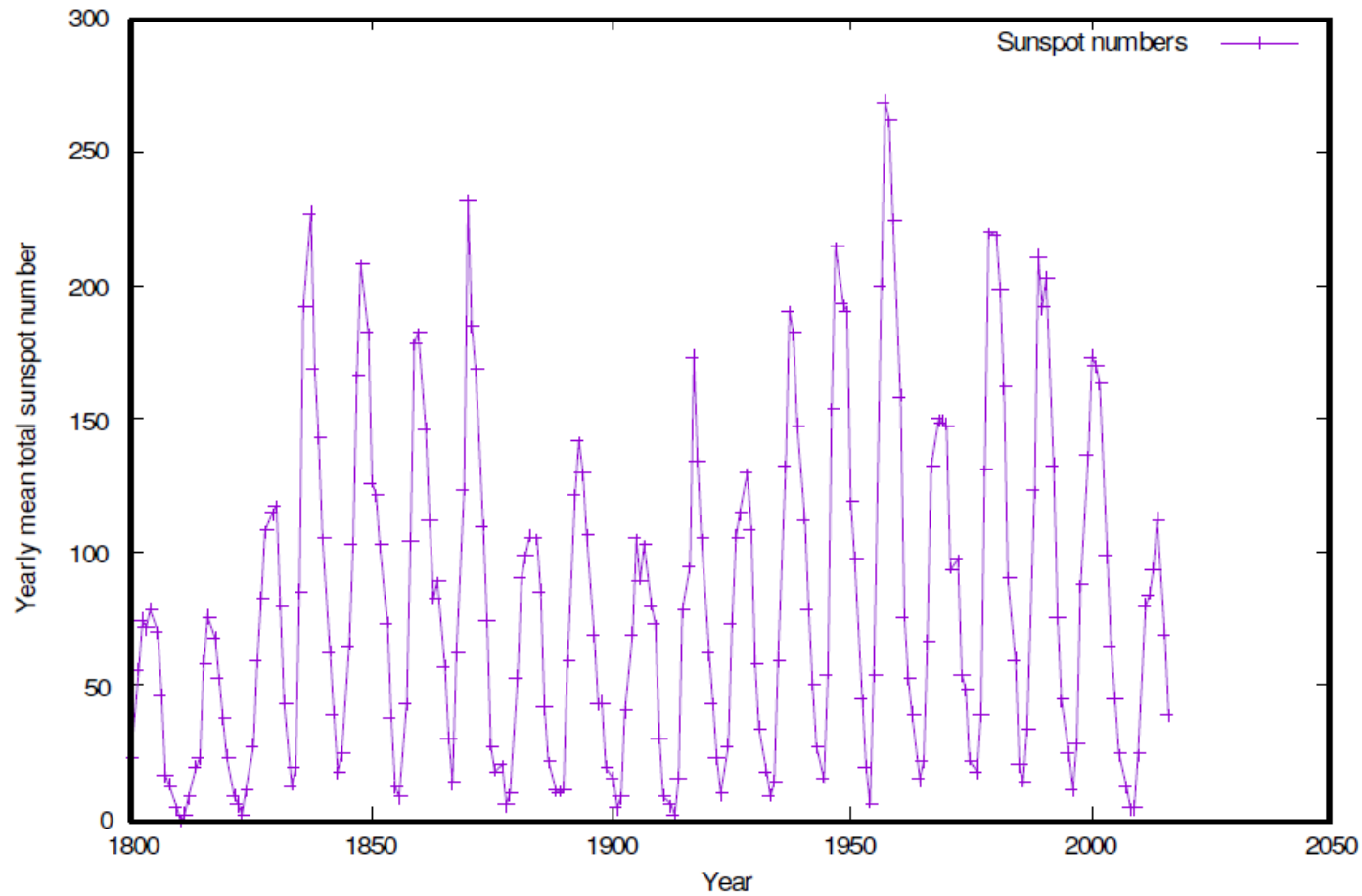
Time delay networks (auto regressive model)

Suppose we have a time series:

$$x(0), x(1), x(2), ..., x(T)$$

Task:

Predict $x(T + 1)$

HALMSTAD
UNIVERSITY

An example: The yearly sunspot number!

HALMSTAD UNIVERSITY

The approach is to use a fixed number of
"previous" values of *x(t)* in order to predict *x(t+1)*

As an example we can create the following
input-output dataset for the sunspot data
using 5 history data points:

Data no	Input	Target
1	$x(T-1), x(T-2), x(T-3), x(T-6), x(T-12)$	$x(T)$
2	$x(T-2), x(T-3), x(T-4), x(T-7), x(T-13)$	$x(T-1)$
3	$x(T-3), x(T-4), x(T-5), x(T-8), x(T-14)$	$x(T-2)$
...	...	...

HALMSTAD UNIVERSITY

Data no	Input	Target
1	$x(T-1), x(T-2), x(T-3), x(T-6), x(T-12)$	$x(T)$
2	$x(T-2), x(T-3), x(T-4), x(T-7), x(T-13)$	$x(T-1)$
3	$x(T-3), x(T-4), x(T-5), x(T-8), x(T-14)$	$x(T-2)$
...	...	...

We can train an ordinary MLP
for this regression problem
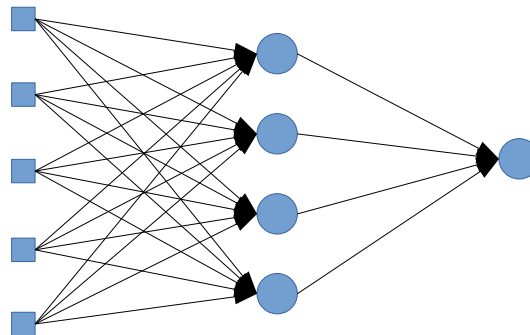
MLP: 5 inputs, 4 hidden nodes, a single linear output node.
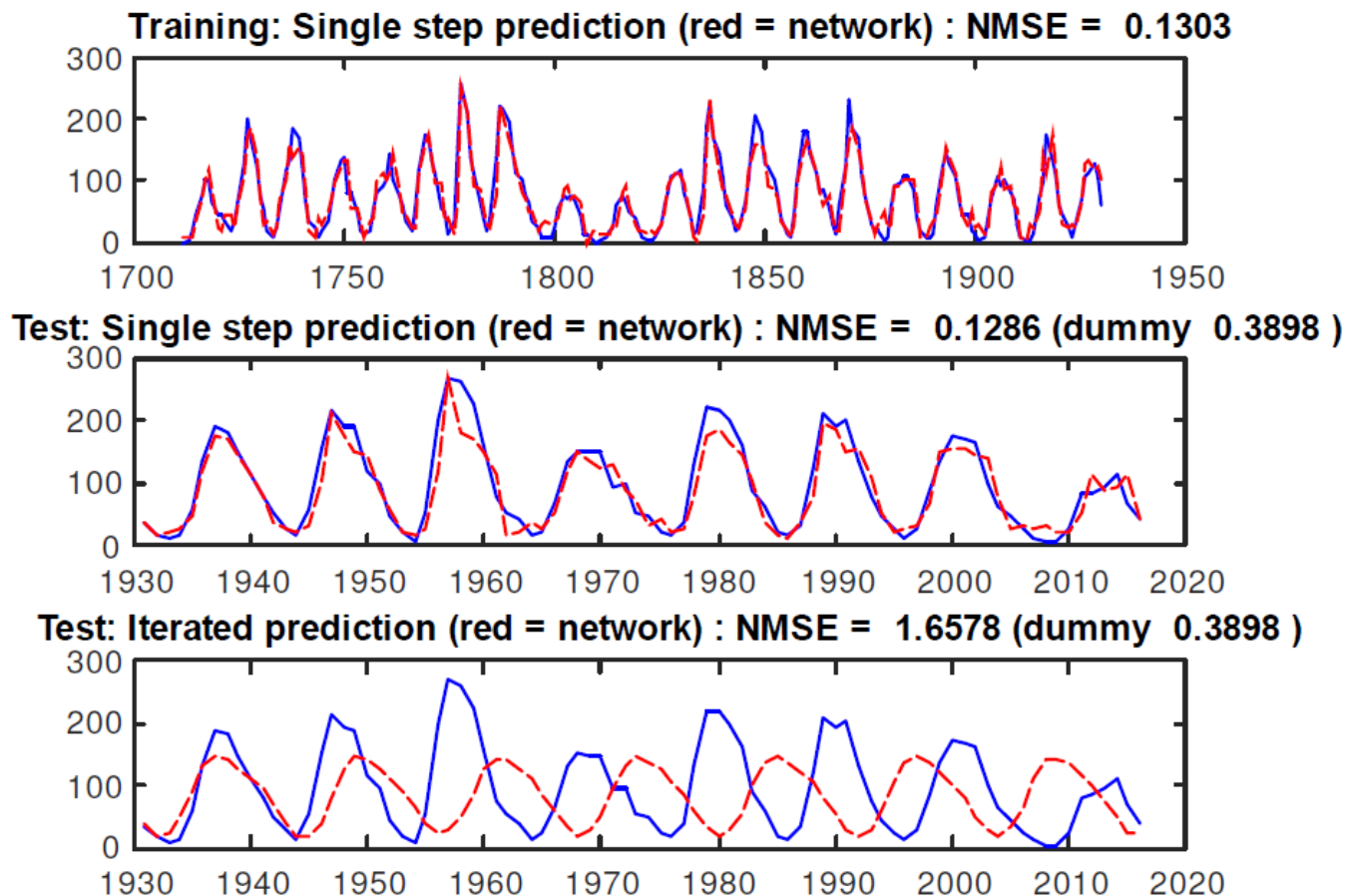
NMSE = Normalized mean squared error

HALMSTAD
UNIVERSITY

Data no	Input	Target
1	$x(T-1), x(T-2), x(T-3), x(T-6), x(T-12)$	$x(T)$
2	$x(T-2), x(T-3), x(T-4), x(T-7), x(T-13)$	$x(T-1)$
3	$x(T-3), x(T-4), x(T-5), x(T-8), x(T-14)$	$x(T-2)$
...	...	...

Obvious drawback! Need to specify the number of history values to use!

Simple recurrent networks, Elman type
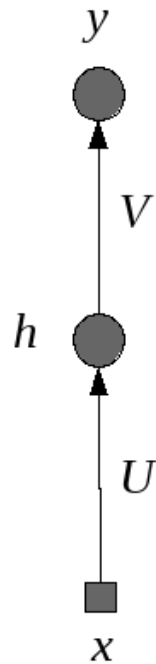
A simple 1-1-1 MLP
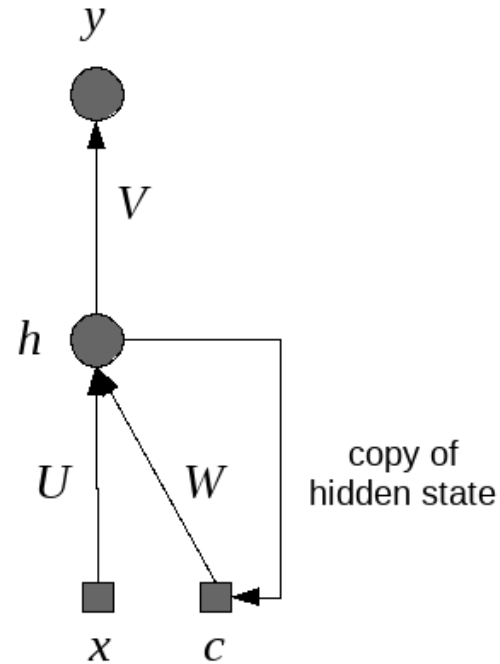
This network receives a copy of the hidden state and use that as an input.

So, how does the Elman network work?
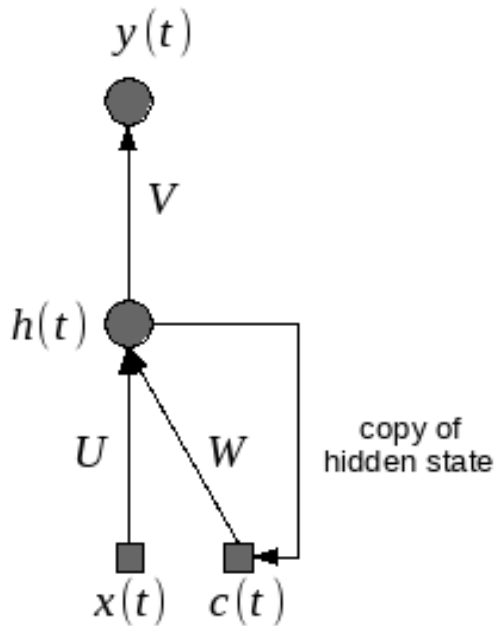

Given: $\quad x(0), x(1), ..., x(T)$

$$y(t) = g_{\mathrm{o}}\left(Vh(t)\right)$$
$$h(t) = g_{\mathrm{h}}\left(Ux(t) + Wc(t)\right)$$
$$c(t) = h(t-1)$$

HALMSTAD
UNIVERSITY

How do we train the Elman network?

$$E = \frac{1}{T} \sum_{t=0}^{T} E_t = \frac{1}{T} \sum_{t=0}^{T} (d(t) - y(t))^2$$

Target sequence

For gradient descent we need derivatives:

$$\frac{\partial y(t)}{\partial V} = g_o'(\cdot)h(t)$$

$$\frac{\partial y(t)}{\partial U} = g_o'(\cdot)Vg_h'(\cdot)x(t)$$

$$\frac{\partial y(t)}{\partial W} = g_o'(\cdot)Vg_h'(\cdot)c(t)$$

Correct?

HALMSTAD UNIVERSITY

How to treat this derivative?

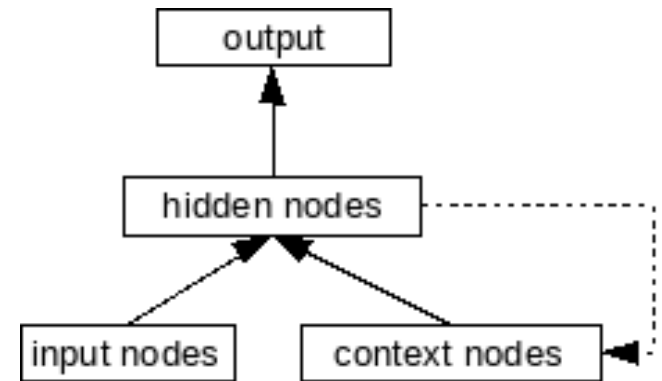$$\frac{\partial}{\partial W}(Wc(t))$$

Remember

$$c(t) = h(t-1)$$
$$h(t) = g_{\mathrm{h}}\left(Ux(t) + Wc(t)\right)$$

Elman network, simply ignore this dependence. This is
why we use the phrase ``copy of hidden state''

Elman, online
updating!

1. present an input $x(t)$

2. compute the hidden node value using $x(t)$ and the previous hidden node
   value $c(t)$. If $t = 0$ then $c(0)$ is typically set to zero

3. compute the output $y(t)$

4. perform all calculations (backpropagation) to compute all weight updates
   for this input "pattern"

5. update the weights V, U, W

6. go back to item 1, using the next input value in the sequence $(t \to t + 1)$

HALMSTAD
UNIVERSITY

Example using 5 hidden nodes (and a single input)!

Training: Single step prediction (red = network) : NMSE = 0.1395

Validation: Single step prediction (red = network) : NMSE = 0.1308

HALMSTAD UNIVERSITY
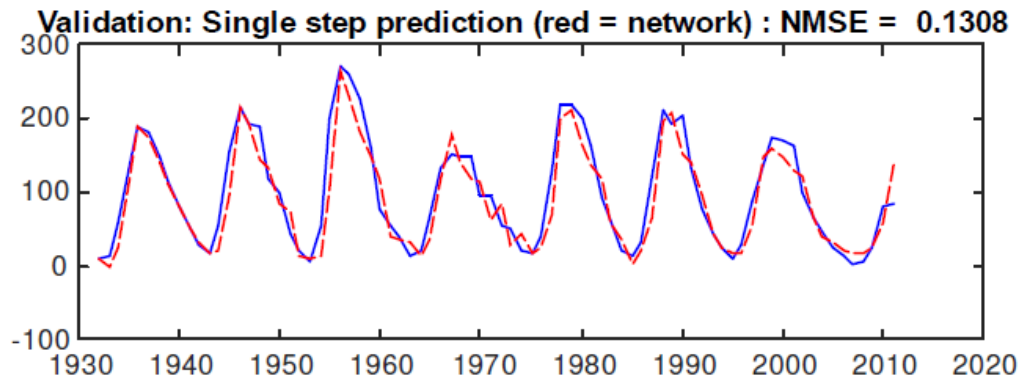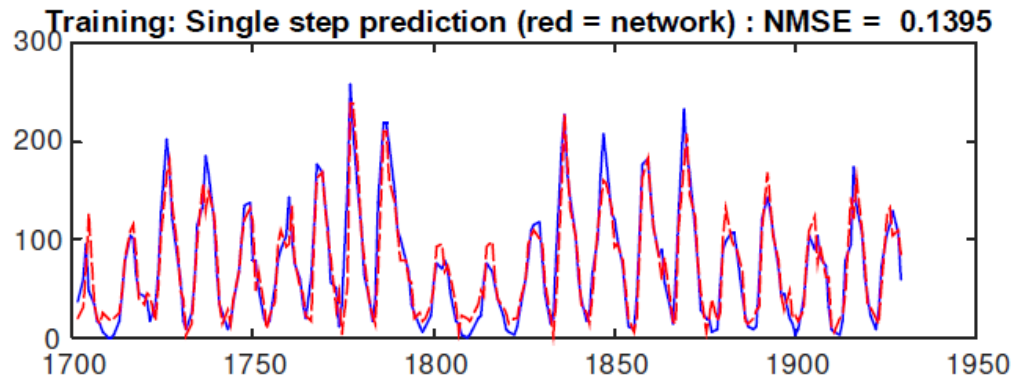
Simple recurrent networks, general type!
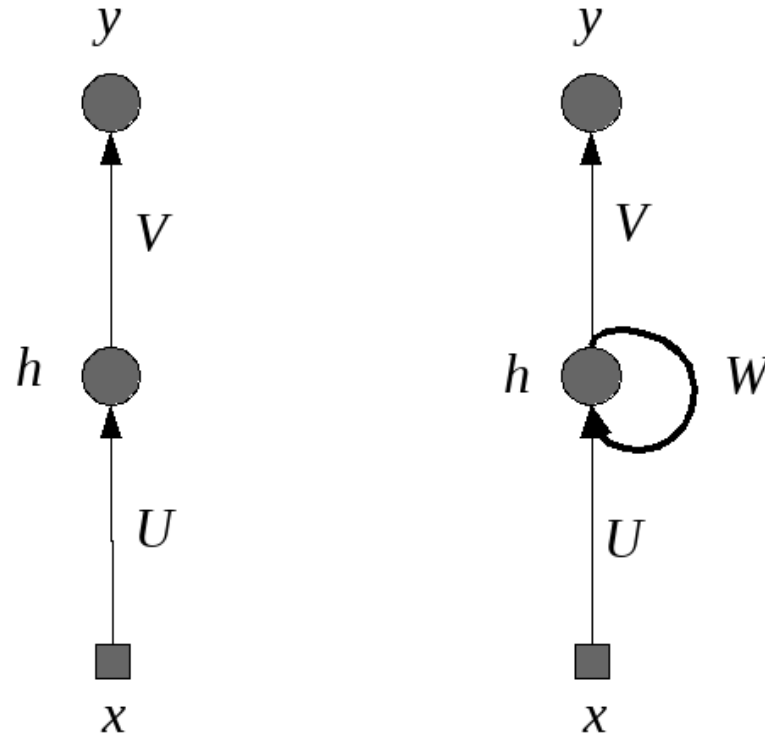
A simple 1-1-1 MLP

This network has a "true" feedback connection from the hidden output feeding into itself.

HALMSTAD UNIVERSITY

Given: $x(0), x(1), ..., x(T)$

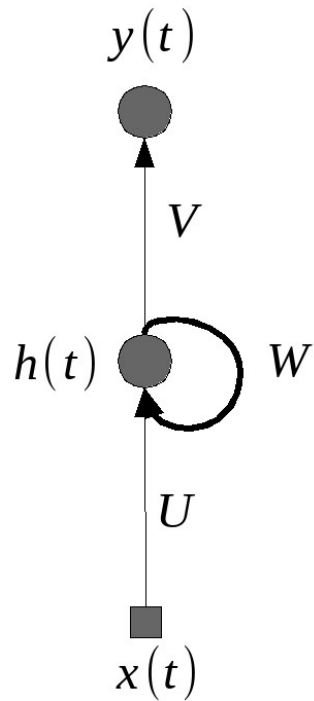$$y(t) = g_{\mathrm{o}}\left(Vh(t)\right)$$
$$h(t) = g_{\mathrm{h}}\left(Ux(t) + Wh(t-1)\right)$$

HALMSTAD
UNIVERSITY

Let's be explicit for a few time steps!

to avoid clutter: $\quad x(t) \to x_t, \;\; h(t) \to h_t$

$$y(0) = g_\mathrm{o}\big[V h_0\big] = g_\mathrm{o}\big[V g_\mathrm{h}\big(U x_0\big)\big] \qquad \text{(using the initial condition } h_{-1} = 0)$$

$$y(1) = g_\mathrm{o}\big[V h_1\big] = g_\mathrm{o}\big[V g_\mathrm{h}\big(U x_1 + W h_0\big)\big] = g_\mathrm{o}\big[V g_\mathrm{h}\big(U x_1 + W g_\mathrm{h}(U x_0)\big)\big]$$

$$y(2) = g_\mathrm{o}\big[V h_2\big] = g_\mathrm{o}\big[V g_\mathrm{h}\big(U x_2 + W h_1\big)\big] =$$

$$= g_\mathrm{o}\big[V g_\mathrm{h}\big(U x_2 + W g_\mathrm{h}(U x_1 + W h_0)\big)\big] =$$

$$= g_\mathrm{o}\Big[V g_\mathrm{h}\Big(U x_2 + W g_\mathrm{h}\big(U x_1 + W g_\mathrm{h}(U x_0)\big)\Big)\Big]$$

Do we recognize this structure?

HALMSTAD UNIVERSITY
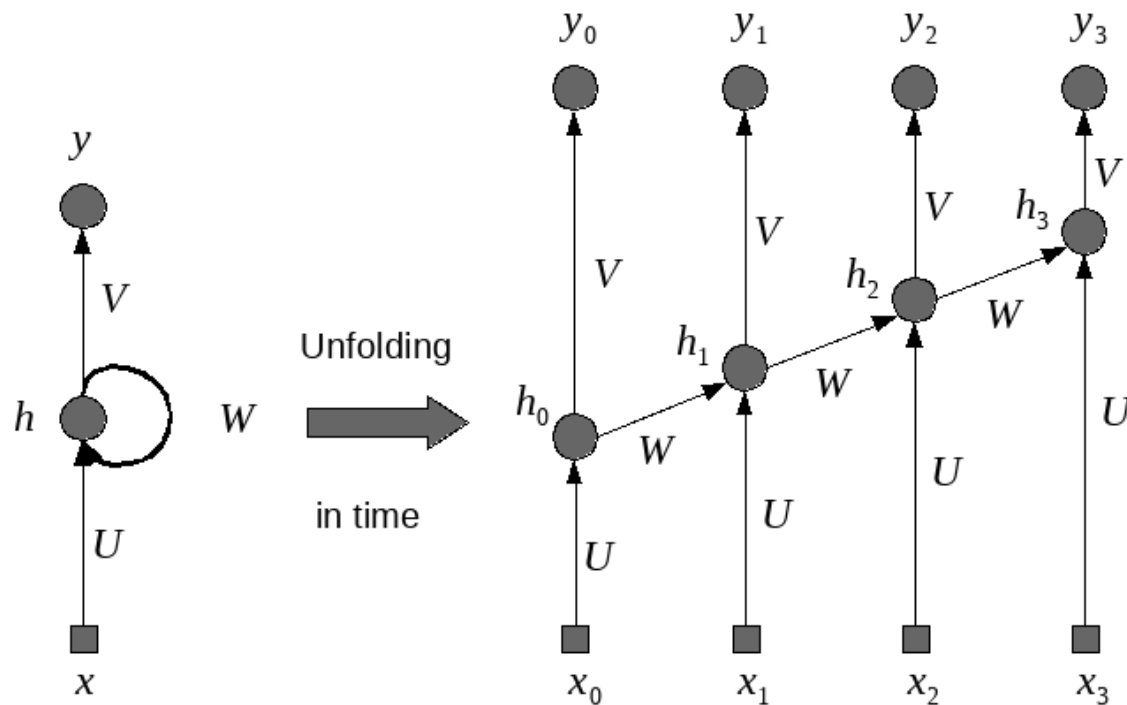
$$y(2) = g_{\text{o}} \left[V g_{\text{h}} \Big(U x_2 + W g_{\text{h}} \big(U x_1 + W g_{\text{h}} \big(U x_0 \big) \big) \Big) \right]$$

Unfolding in time!

The unfolded network is an MLP with 4 hidden layers,
sparsely connected and shared weights.

HALMSTAD
UNIVERSITY

Backpropagation through time (BPTT)
(= basically training the unfolded MLP)
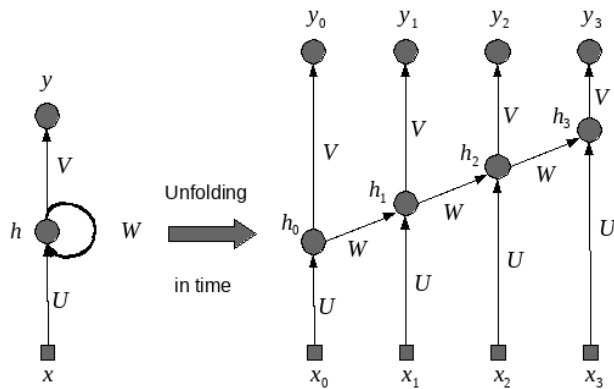
Assume an error of this kind!

$$E(U, W, V) = \sum_{t=0}^{T} E_t(U, W, V)$$

Can measure the difference between
a target sequence and an input sequence

But many other possibilities, eg. classification of sequences.

HALMSTAD
UNIVERSITY

To do gradient descent we need to compute derivatives!

The network

$$\frac{\partial E}{\partial V} = \sum_t \frac{\partial E_t}{\partial V}$$

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$

$$\frac{\partial E}{\partial U} = \sum_t \frac{\partial E_t}{\partial U}$$

HALMSTAD UNIVERSITY

The V weight is rather simple!

$$\frac{\partial E_t}{\partial V} = \frac{\partial E_t}{\partial y_t}\frac{\partial y_t}{\partial V} = \frac{\partial E_t}{\partial y_t}g'_{\mathrm{o}}(Vh_t)h_t$$

(Remember)

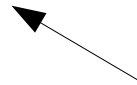$$y_t = g_{\mathrm{o}}\left(Vh_t\right)$$
$$h_t = g_{\mathrm{h}}\left(Ux_t + Wh_{t-1}\right)$$

The other two are more complicated! For the W weight we get:

$$\frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial W}$$

Now,

$$h_t = g_{\mathrm{h}}(Ux_t + Wh_{t-1})$$

Depends on W

The correct expression:

$$\frac{\partial E_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$$

HALMSTAD
UNIVERSITY

Is this a problem?

$$\frac{\partial E_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$$

This is also a chainrule

For example $\quad \dfrac{\partial h_3}{\partial h_1} = \dfrac{\partial h_3}{\partial h_2} \dfrac{\partial h_2}{\partial h_1}$

Now each $\quad \dfrac{\partial h_{t+1}}{\partial h_t} = g'_h(\cdot)W$

HALMSTAD UNIVERSITY

To conclude: Training a simple recurrent network like this, for long sequences, involves multiplication of many terms like,

$$g'_h(\cdot)W$$
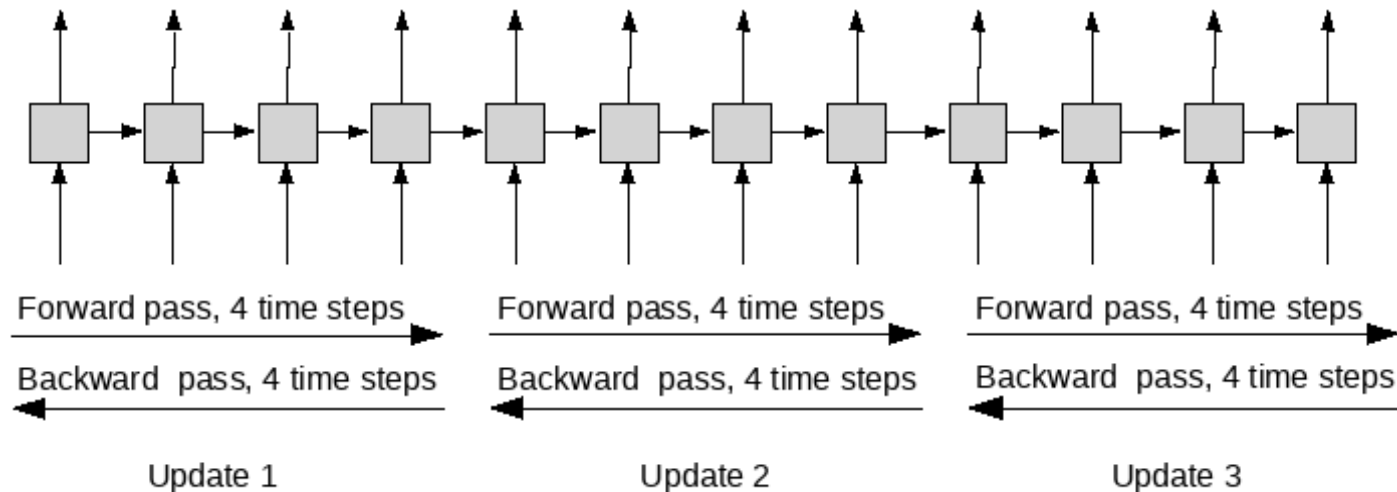
Two numerical problems can occur:

- Vanishing gradients
- Exploding gradients

Where vanishing gradients are probably most common!

As a result it is difficult to learn long term dependencies!

HALMSTAD UNIVERSITY

How do we solve this?

First approach: Truncated backpropagation through time!

Forward pass, 4 time steps

Backward pass, 4 time steps

Update 1

Forward pass, 4 time steps

Backward pass, 4 time steps

Update 2

Forward pass, 4 time steps

Backward pass, 4 time steps

Update 3

Here it is important to preserve the state between updates!

HALMSTAD UNIVERSITY

Second approach: Change the behavior of the hidden node
as to have a more or less constant value of the recursive derivative.
→ **LSTMs**

Before that let's just conclude by noting that the simple
recurrent network can be more complicated than what we have
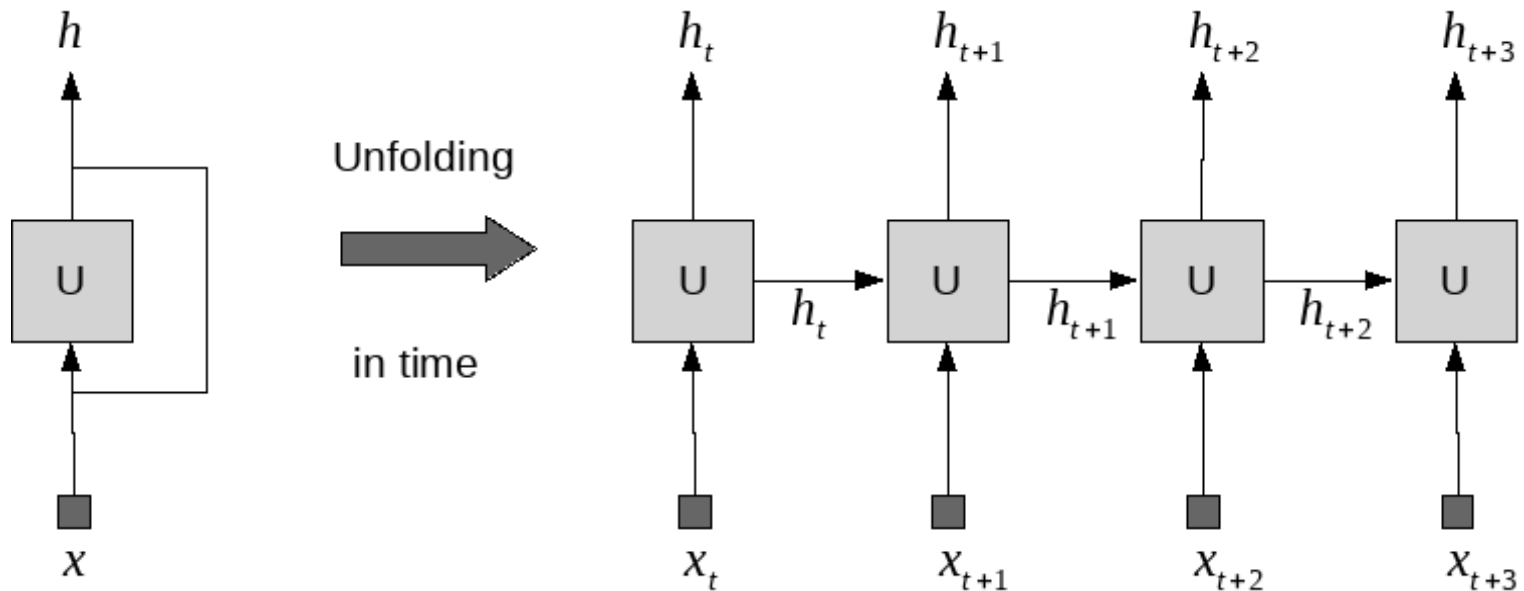showed here. As an example:
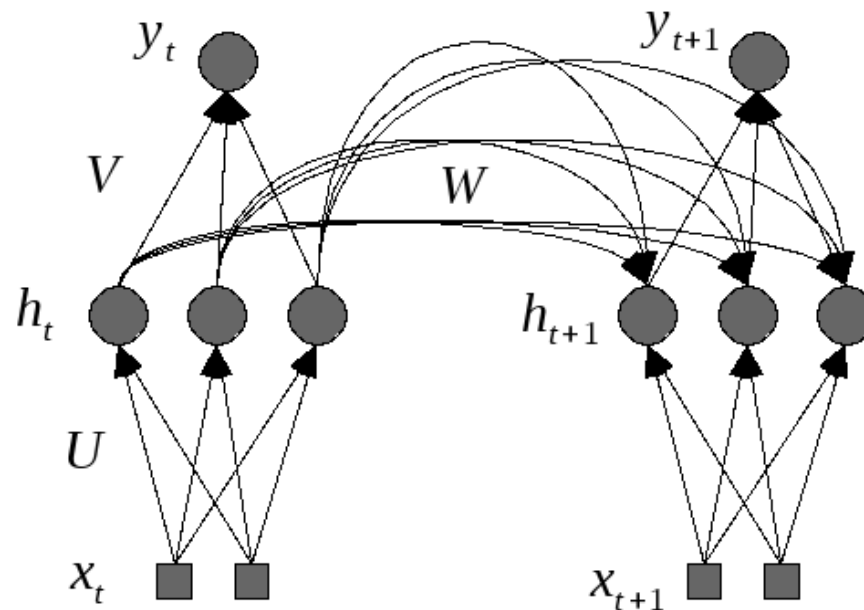
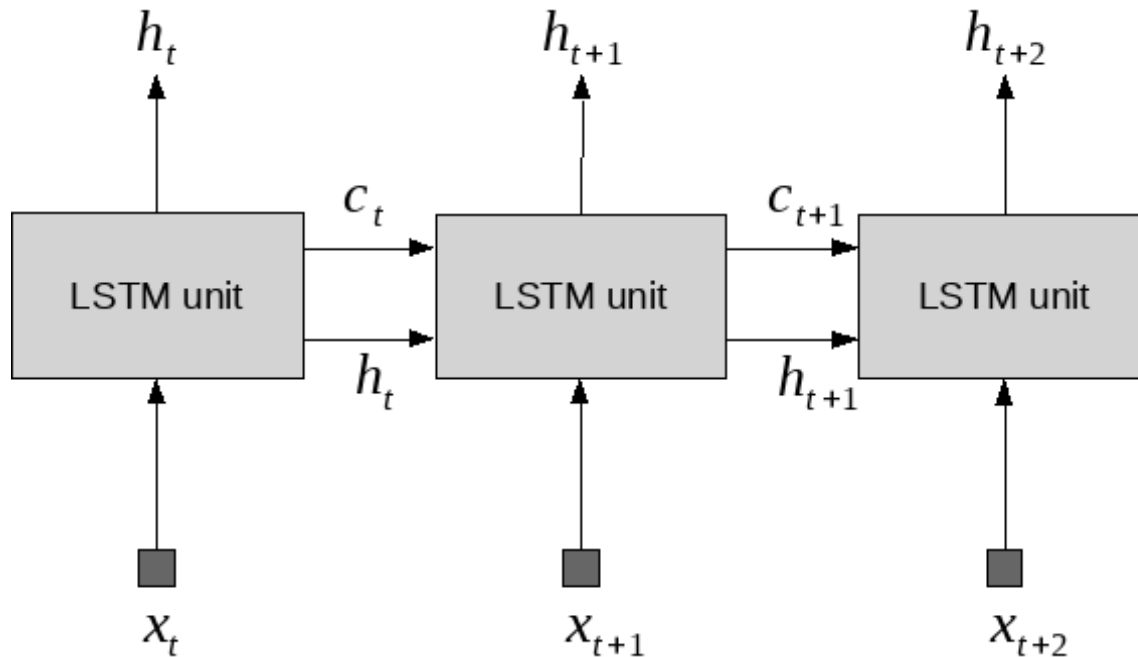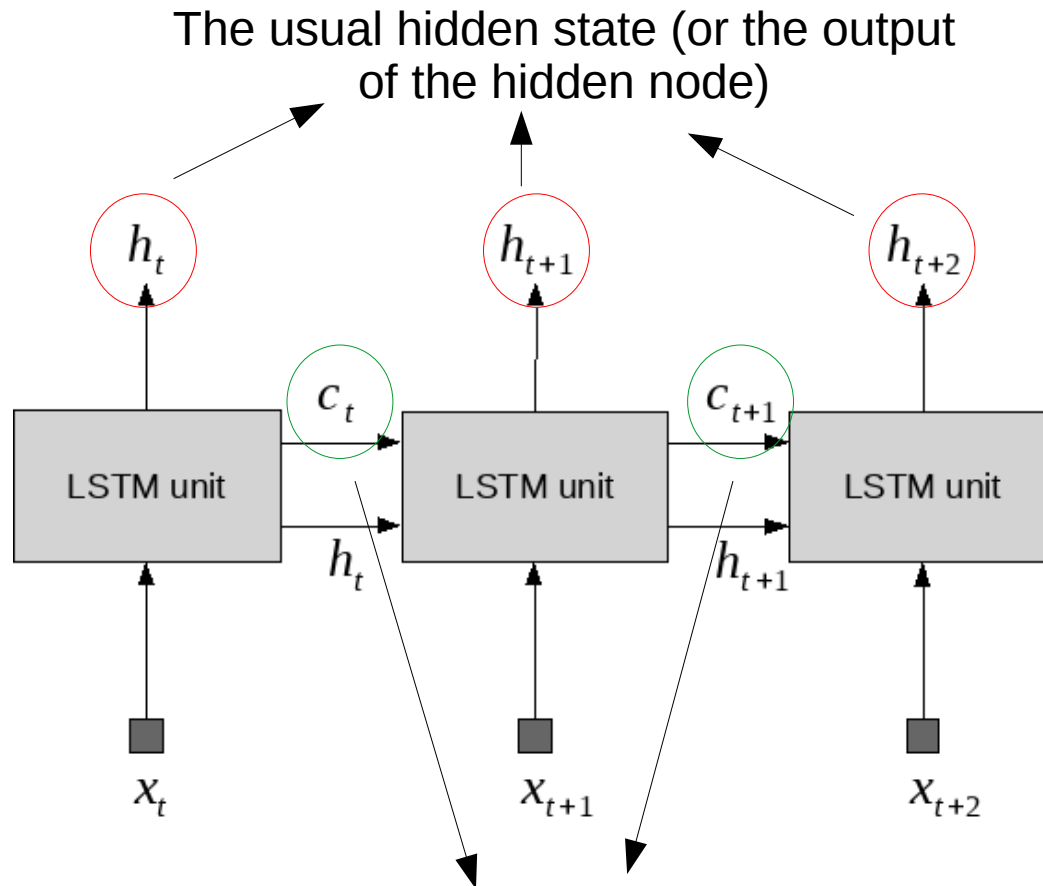Two input, three hidden nodes. Unfolded one time step!

HALMSTAD
UNIVERSITY

Long short-term memories (LSTM)

Previously we had networks like this, where the "U" box represents the simple recurrent unit.

HALMSTAD UNIVERSITY

For a LSTM network we simply replace the "U" box with a LSTM box and add a new connection,

$$h_t \qquad\qquad h_{t+1} \qquad\qquad h_{t+2}$$

So in a sense it is similar to the previous network (apart from the "c" value. However the LSTM box is more complicated than the "U" box!

HALMSTAD UNIVERSITY

The usual hidden state (or the output of the hidden node)

h_t

h_{t+1}

h_{t+2}

c_t

c_{t+1}

LSTM unit

LSTM unit

LSTM unit

h_t

h_{t+1}

x_t

x_{t+1}

x_{t+2}

Internal memory of the LSTM node. Is not "exported"!

HALMSTAD UNIVERSITY

Details of the LSTM node!

In addition to the internal memory c_t we have so called gates

$$
\begin{aligned}
\text{input gate } i &= \sigma(x_t U^i + h_{t-1} W^i + b_i) \\
\text{forget gate } f &= \sigma(x_t U^f + h_{t-1} W^f + b_f) \\
\text{output gate } o &= \sigma(x_t U^o + h_{t-1} W^o + b_o)
\end{aligned}
$$

They are all number between 0 and 1, and are used to
"filter" new values for the memory and hidden state.

$$(U^i, U^f, U^o), (W^i, W^f, W^o), (b_i, b_f, b_o)$$ **= new weights!**

HALMSTAD
UNIVERSITY

The internal calculation is done as follows:

1. New candidate value for the internal memory.

$$\tilde{c}_t = \tanh(x_t U^c + h_{t-1} W^c + b_c)$$

2. New internal memory

$$c_t = c_{t-1} f + \tilde{c}_t i$$
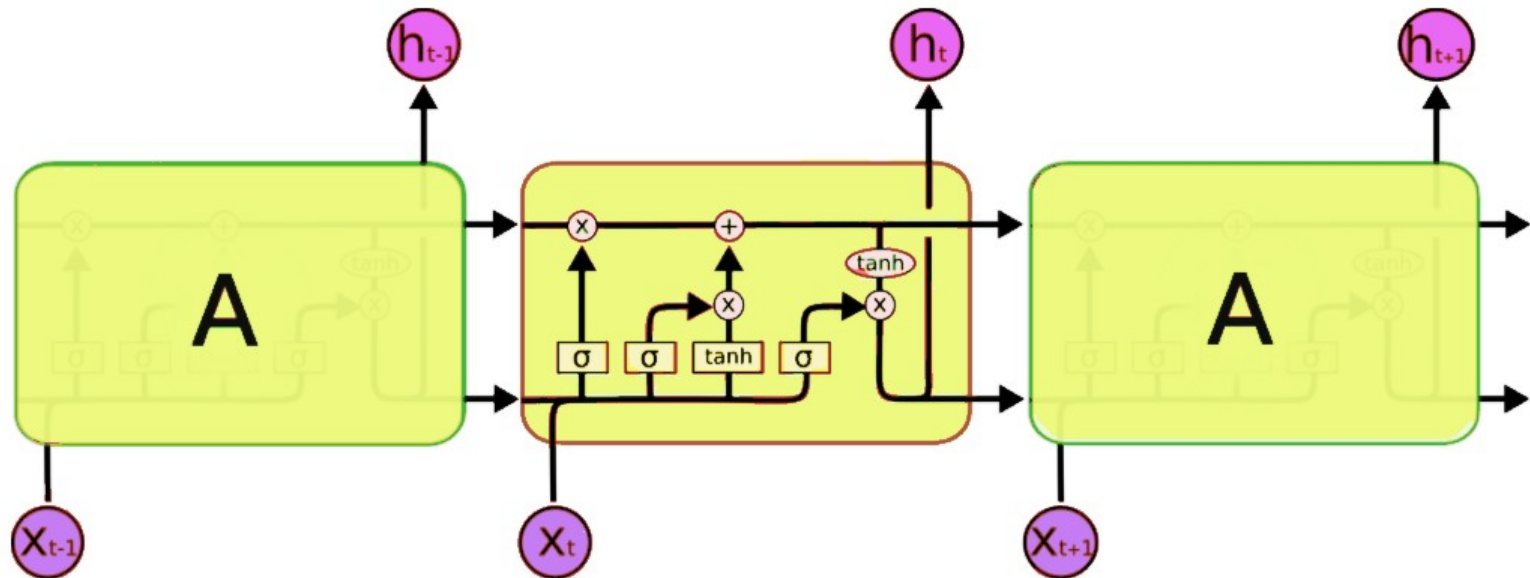
3. New hidden value

$$h_t = \tanh(c_t) o$$

The new memory value is a combination of the previous value filtered by the forget gate and the candidate value filtered by the input gate.

The new output of the LSTM node (h_t) is the memory "squashed" by the tanh() and filtered by the output gate.

It is the gating mechanism that allows the LSTM network to model long-term dependencies.

HALMSTAD UNIVERSITY

There are many illustrations of the LSTM node, here is one:

From a good blog:
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

HALMSTAD UNIVERSITY

Can we understand why the LSTM handles the
vanishing gradient problem?

Previously we looked at $\dfrac{\partial h_{t+1}}{\partial h_t}$

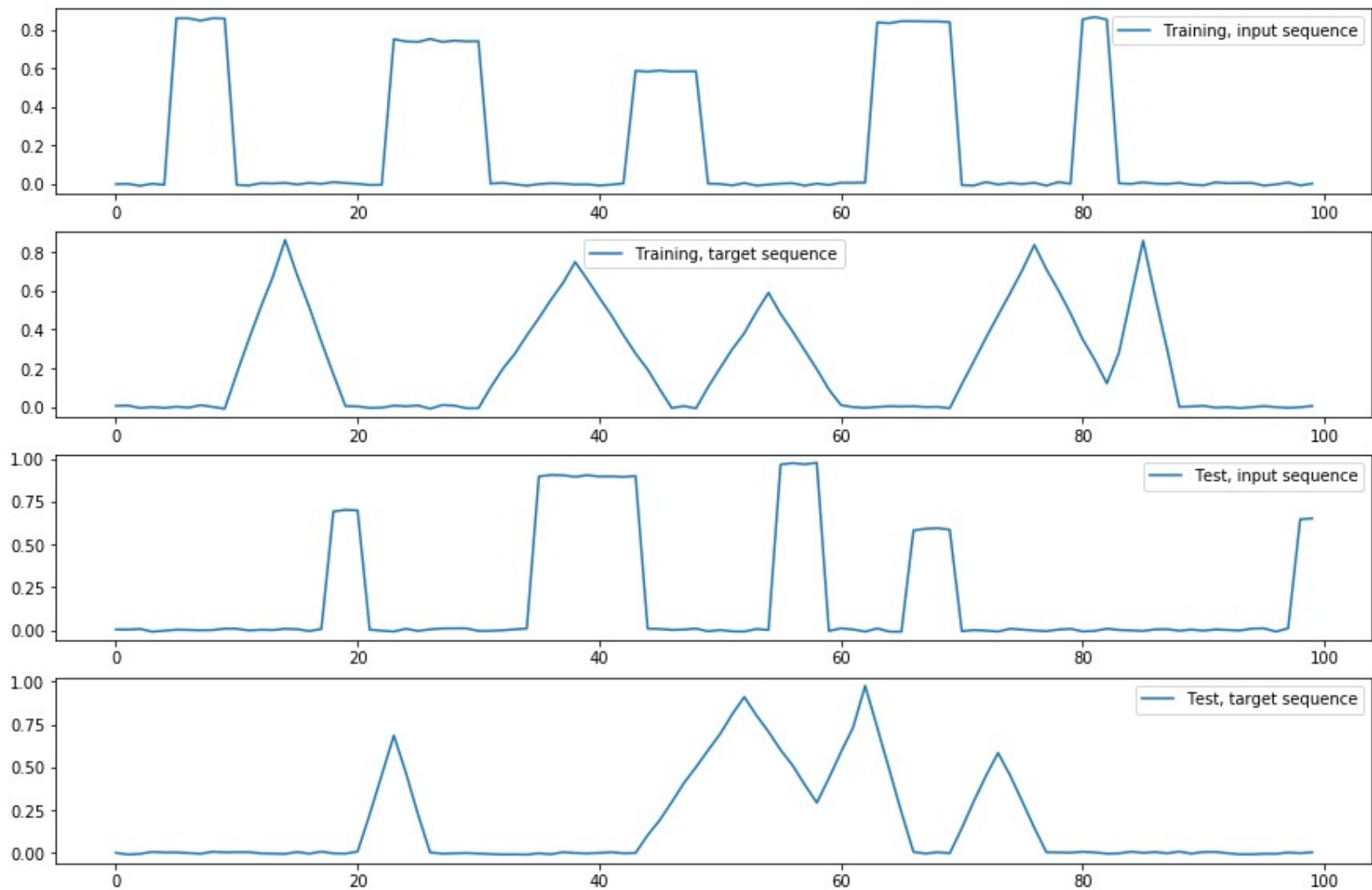A corresponding derivative is now: $\dfrac{\partial c_t}{\partial c_{t-1}}$

HALMSTAD
UNIVERSITY

$$\frac{\partial c_t}{\partial c_{t-1}} = c_{t-1}\sigma'(\cdot)W^f o_{t-1}\tanh'(c_{t-1})$$

$$+\tilde{c}_t\sigma'(\cdot)W^i o_{t-1}\tanh'(c_{t-1})$$

$$+i_t\tanh(\cdot)W^c o_{t-1}\tanh'(c_{t-1})$$

$$+f_t$$

The details are not important, but:

- We will multiply many such terms
- The terms can be both > 1 and < 1
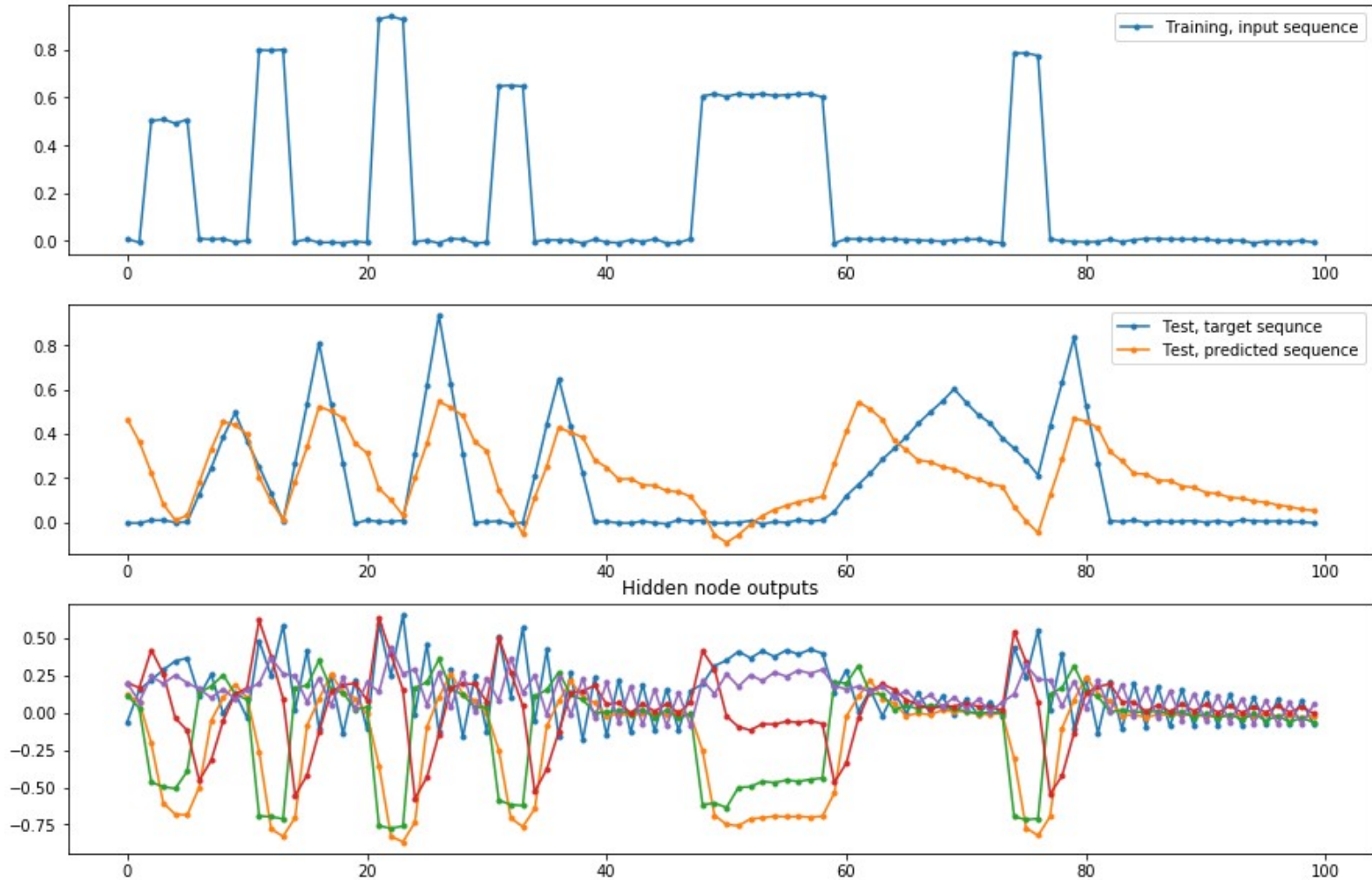- In a sense the network can *learn* when gradients should vanish!!
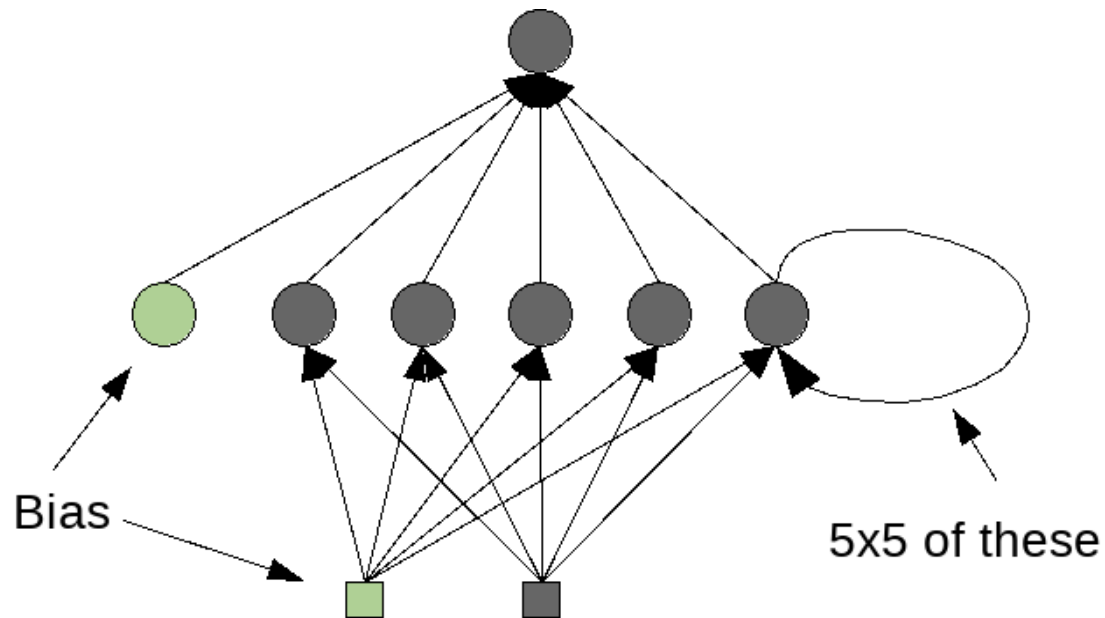
HALMSTAD
UNIVERSITY

Numerical example! RNN as a pulse converter

HALMSTAD UNIVERSITY

Simple recurrent network with 5 hidden nodes (no LSTM)

(41 trainable weights! How?)

Test error: 0.47

HALMSTAD UNIVERSITY

(41 trainable weights! How?)

Bias

5x5 of these

(5+1) + 5x5 + 5x(1+1) = 41

HALMSTAD UNIVERSITY

LSTM network with 5 hidden nodes!

(146 trainable weights!)

Test error: 0.08

HALMSTAD UNIVERSITY
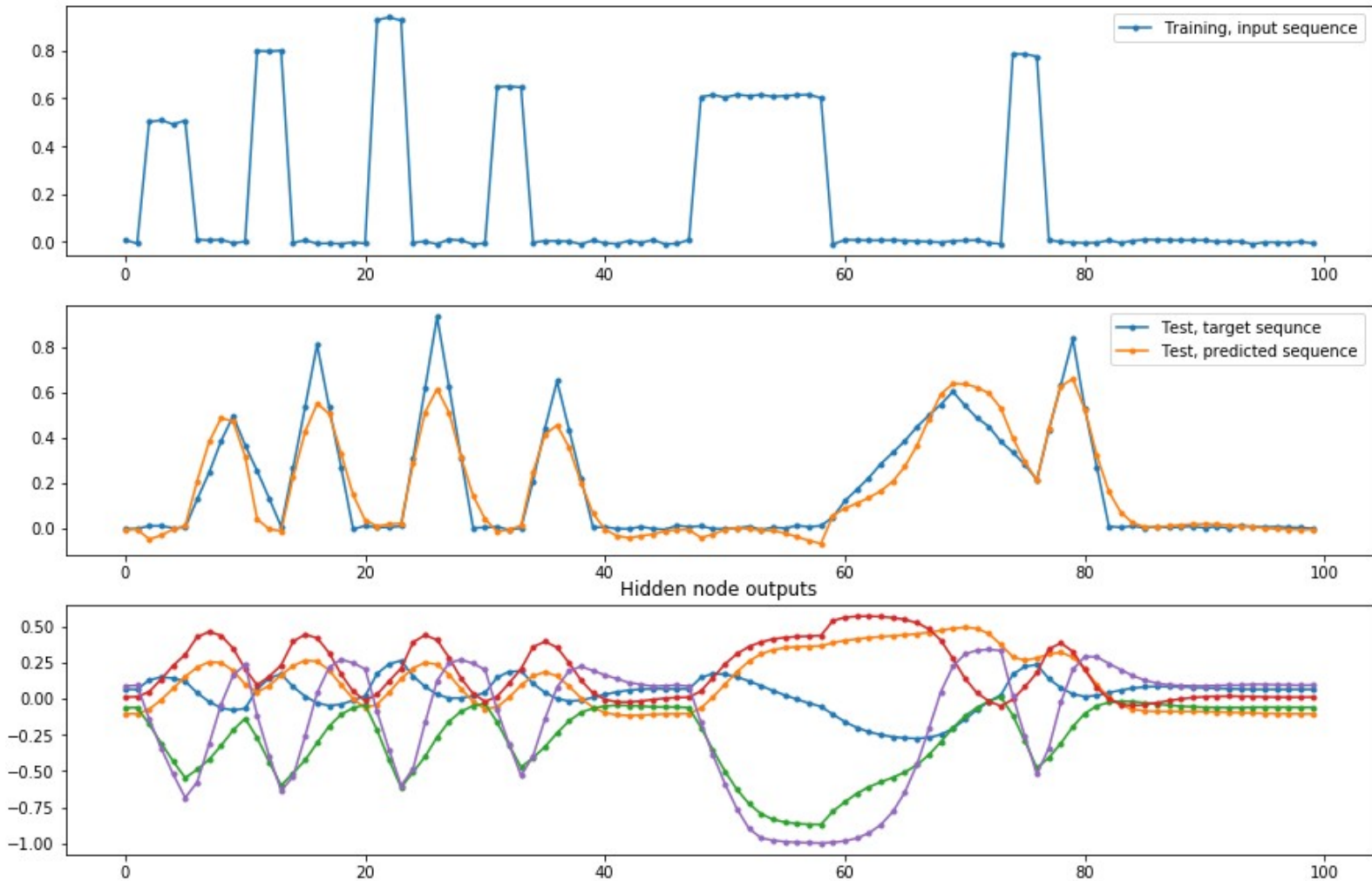
Simple recurrent network with 10 hidden nodes!

(131 trainable weights)

Test error: 0.27

HALMSTAD UNIVERSITY

Multilayer LSTMs

HALMSTAD
UNIVERSITY
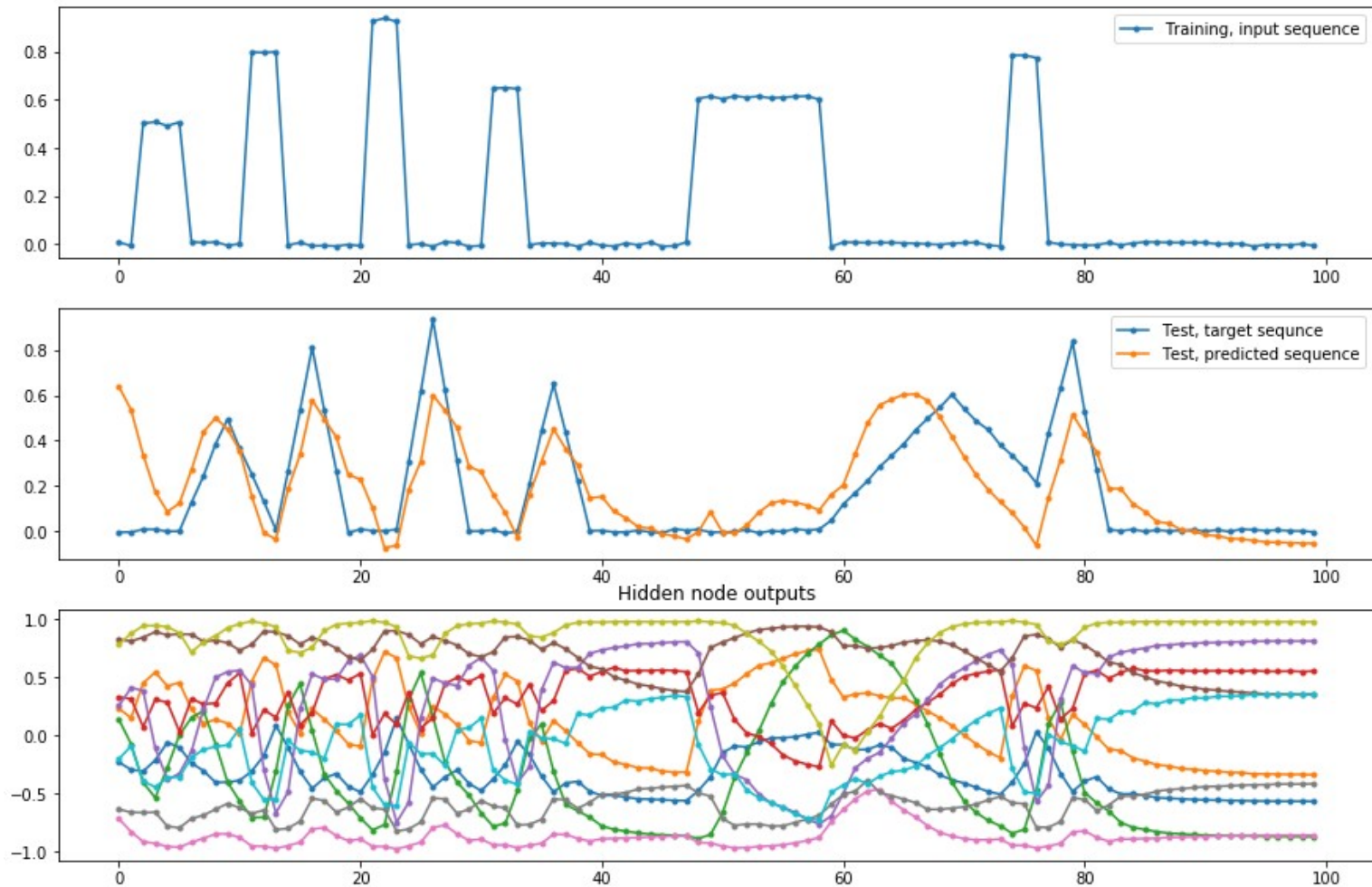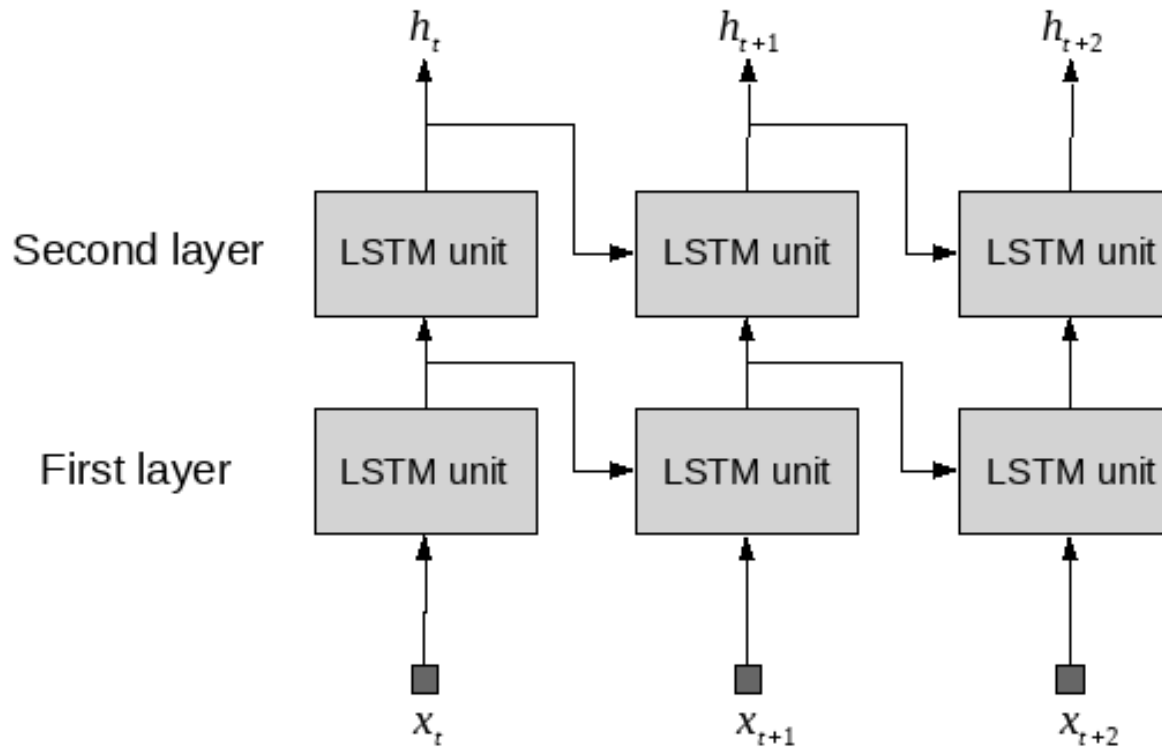
LSTM network with 5 – 3 nodes!

(252 trainable weights!)

Test error: 0.04

HALMSTAD UNIVERSITY
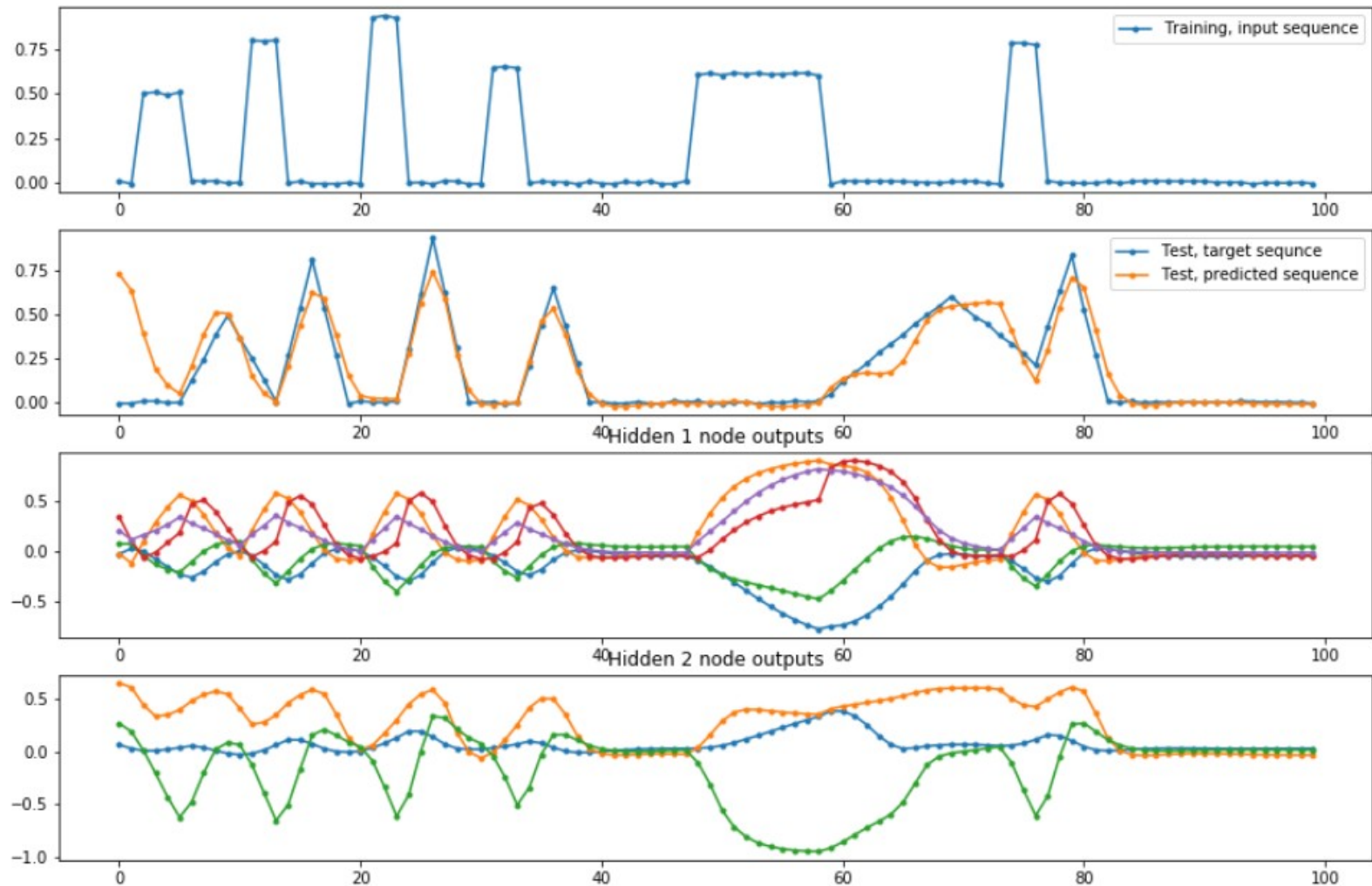
The inverse problem is more difficult!

HALMSTAD UNIVERSITY
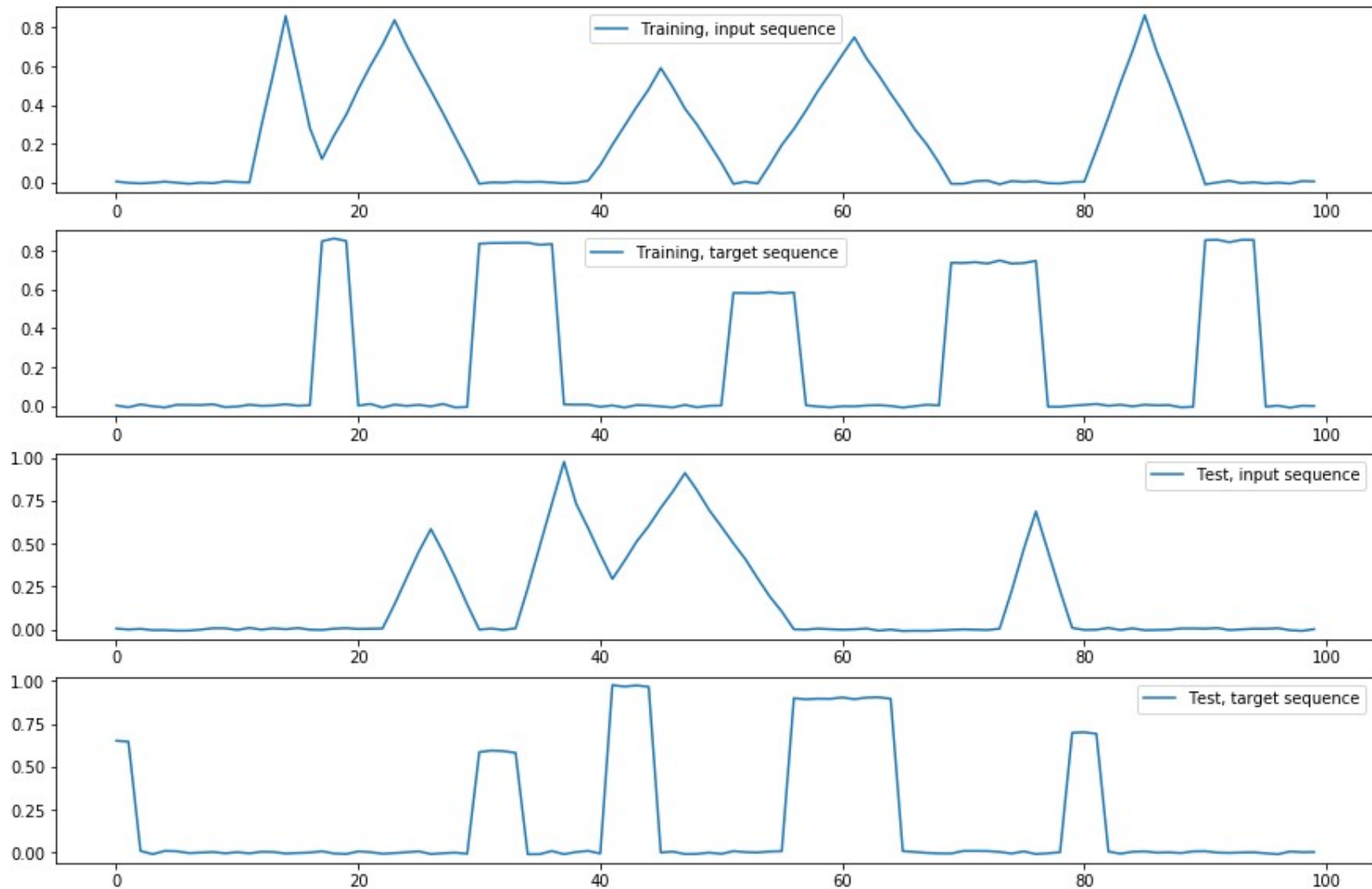
LSTM network with 25 hidden nodes!

(2726 trainable weights!)

Test error: 0.07

HALMSTAD
UNIVERSITY
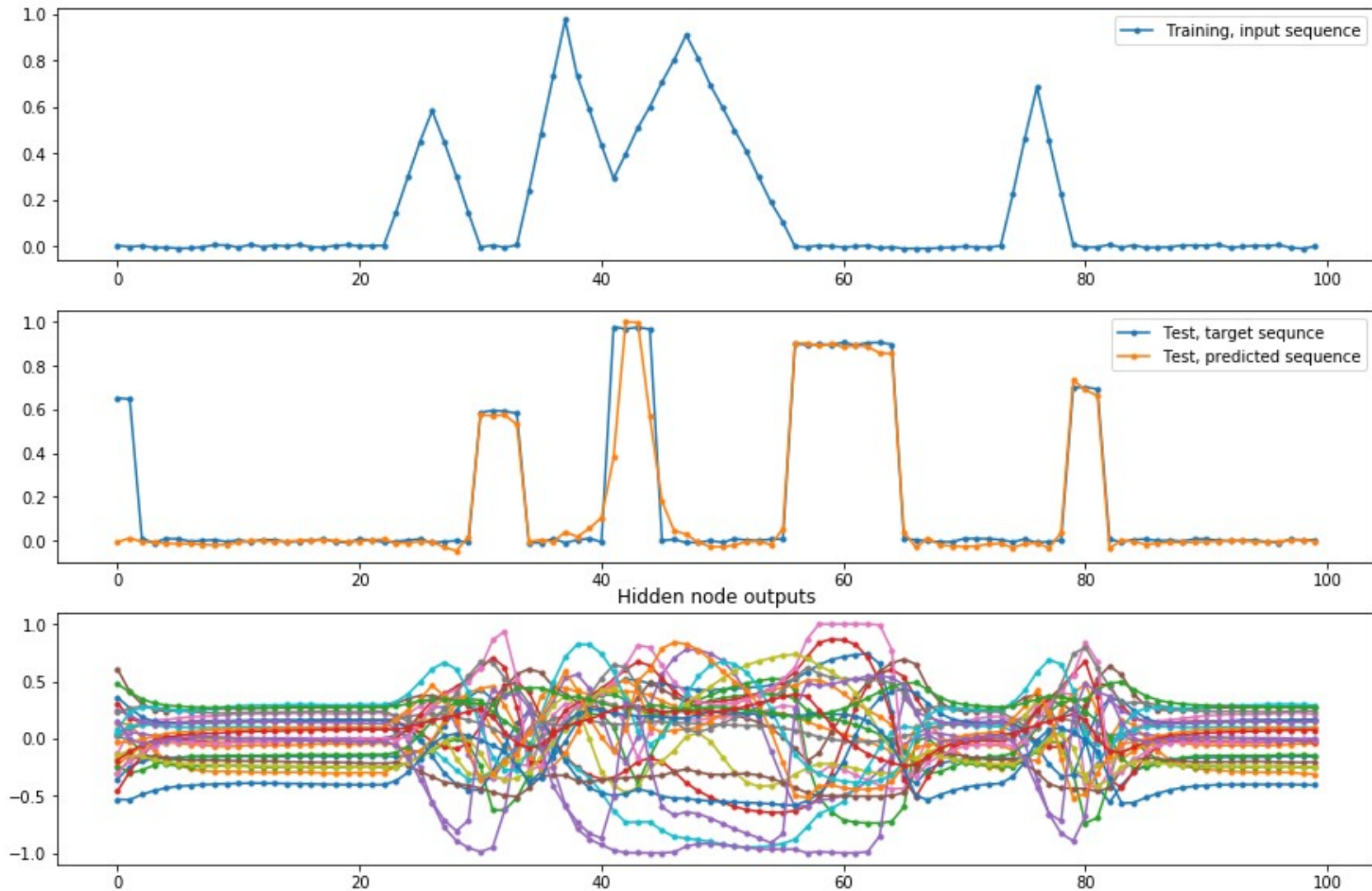
Even with LSTMs it is very common to
use truncated BPTT!

Forward pass, 4 time steps

Backward pass, 4 time steps

Update 1

Forward pass, 4 time steps

Backward pass, 4 time steps

Update 2

Forward pass, 4 time steps

Backward pass, 4 time steps
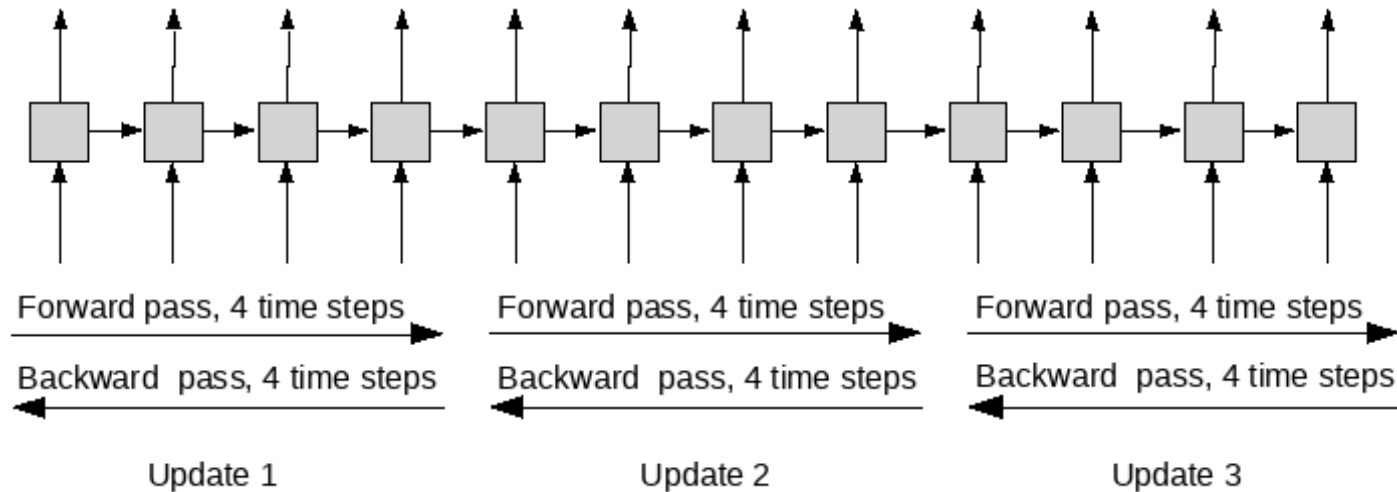
Update 3

But typically with more than 4 times steps!!!

HALMSTAD UNIVERSITY

LSTM variant, The Gated Recurrent Unit (GRU)

Simpler version of the LSTM unit. Introduced 2014!
The core idea is the same, but it is simpler.
No memory and only two gates.

Update gate $\quad z \quad = \quad \sigma(x_t U^z + h_{t-1} W^z + b_z)$

Reset gate $\quad r \quad = \quad \sigma(x_t U^r + h_{t-1} W^r + b_r)$

Candidate output $\quad \tilde{h} \quad = \quad \tanh(x_t U^h + (h_{t-1} r) W^h + b_h)$

Output $\quad h_t \quad = \quad (1-z)\tilde{h} + z h_{t-1}$
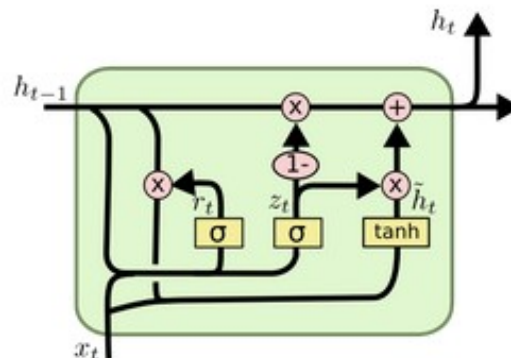
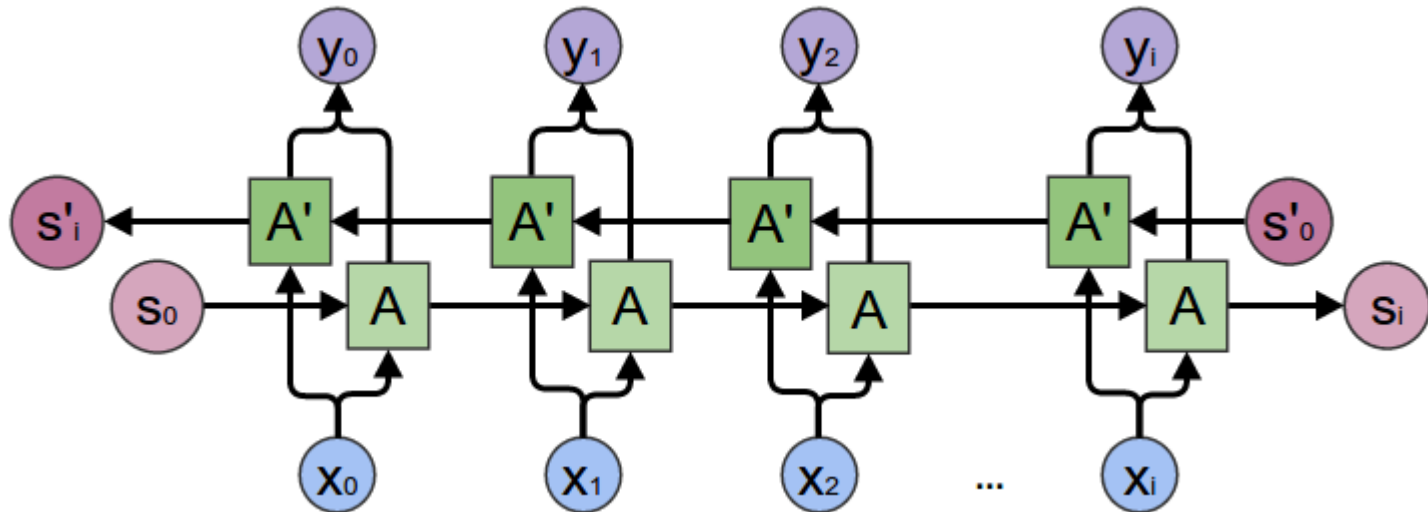# GRU network with 5 – 3 nodes!

(190 trainable weights!)

Test error: 0.03

HALMSTAD UNIVERSITY
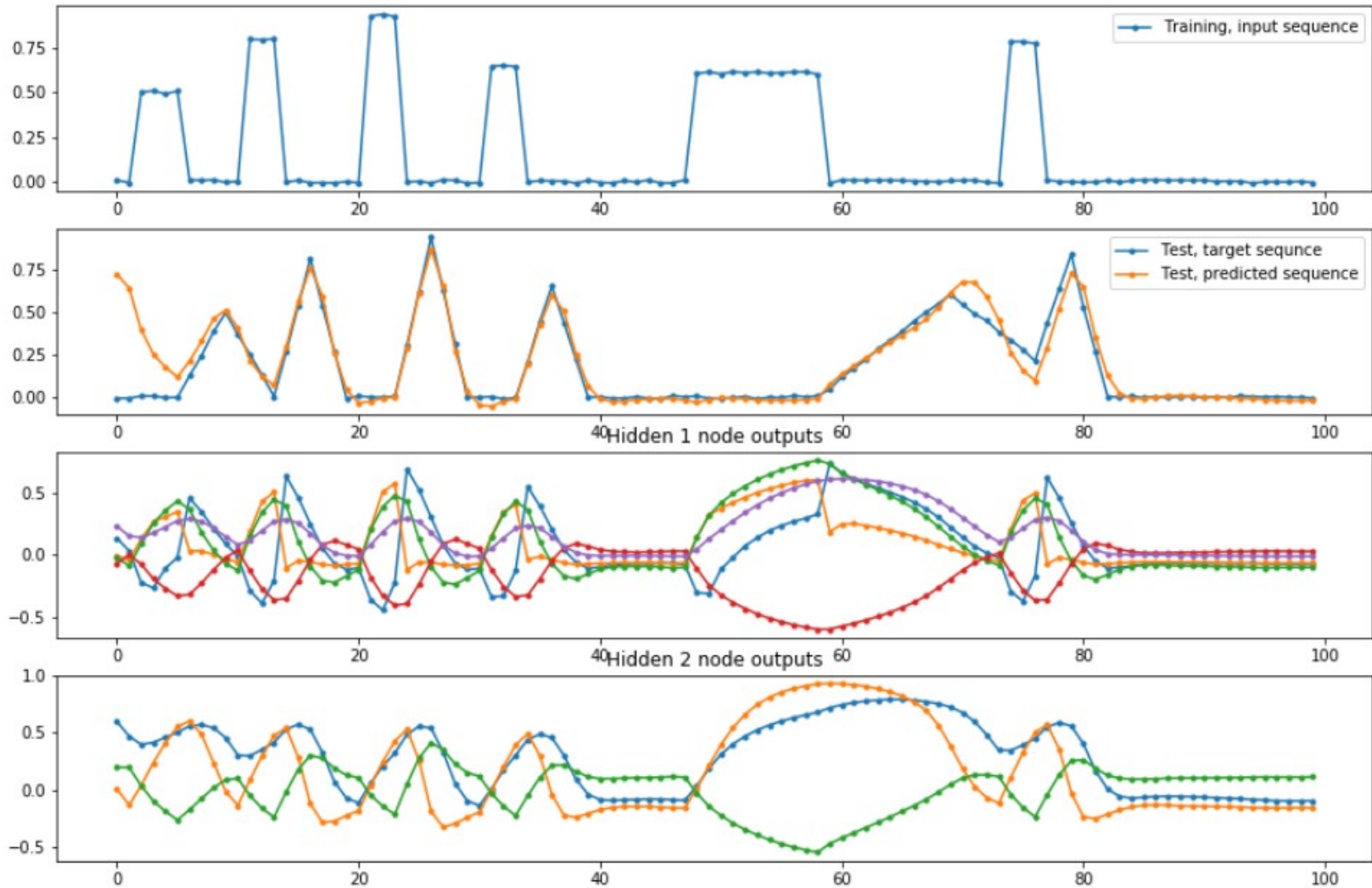
Bidirectional LSTMs

Putting two independent LSTMs together, one for the forward sequence and one for the reverse sequence.

Useful when the context of the input is needed.

CNN LSTM architectures

These architectures combine the CNN for dealing with images and LSTM for the sequnce part. Examples:

Activity recognition: image sequence to classification
Image description: image to text sequence (caption)
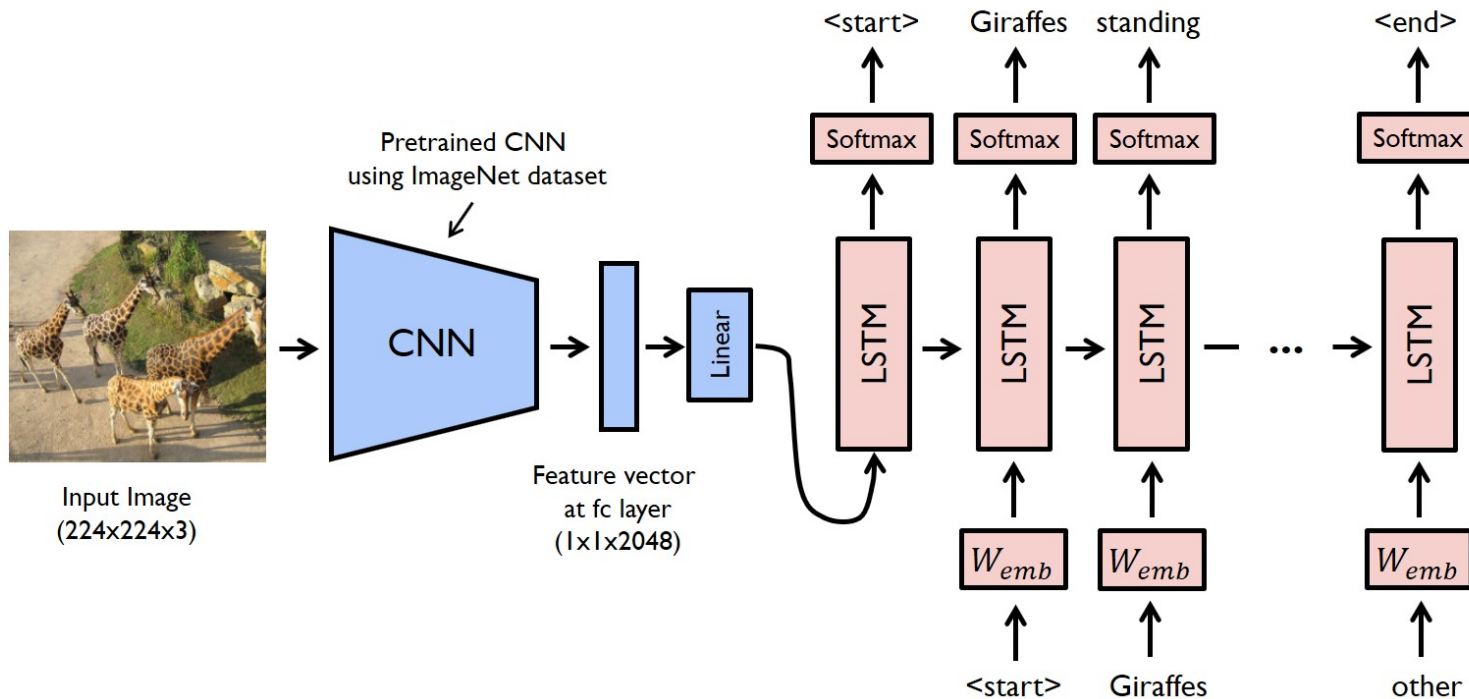**Video description:** video sequence to sequence (description)

Two "early" papers:

Show and Tell: A Neural Image Caption Generator (Link)

Long-term Recurrent Convolutional Networks for
Visual Recognition and Description (Link)

HALMSTAD UNIVERSITY

Example of a caption generator:

HALMSTAD UNIVERSITY

A last example: Sampling from a character model

Task: given a sequence of characters, predict the next character!
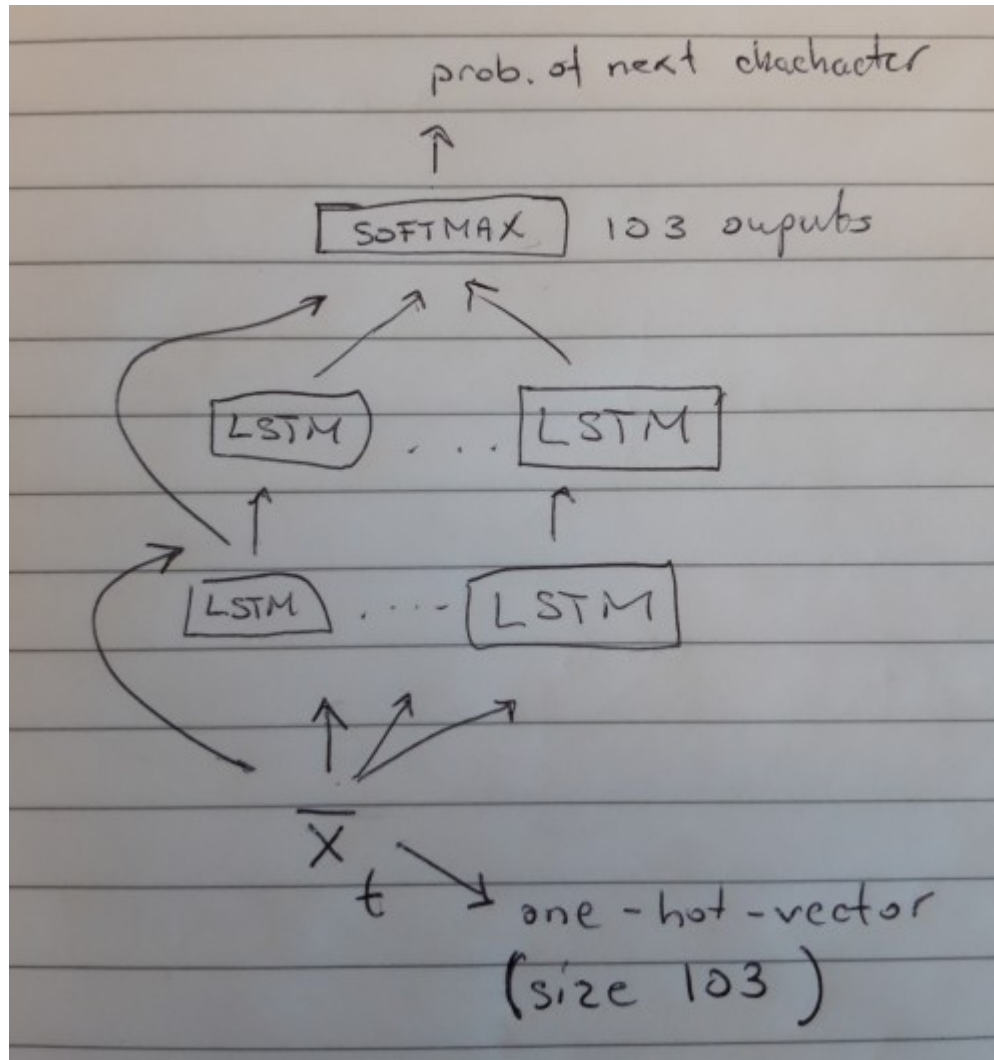
One, two, thre?

My name i?

In this specific example the training sequence
is the source code of (some earlier version) **Tensorflow**.
This is C++ code!

About 14 million characters in the sequence!

HALMSTAD
UNIVERSITY

The model:

103 different characters in the code

Skip layer connections

Two layers of 1024 LSTMs each

About 13 million parameters

HALMSTAD UNIVERSITY

Since the output are probabilities for the 103 different characters, we can sample from the model:

Five characters (A, B, C, D, E)

As a simple example   [0.1, 0.3, 0.5, 0.1, 0.0]

The probability for 'C' is 50%, for 'B' 30% and so on.

The sampled character is then used as the
new input to the (trained) model.

HALMSTAD
UNIVERSITY

Seed: "void "

```
void AddShapeMemoryTranspose(const Key& key, const AllocatorAttributes attr) {
  VLOG(2) << "Instanding worker threads with dues to set cuSolveDDe gradient for: " << variant->devices[0]->env_;
  def.label_fills.push_back(TestMultipleWrites());
  std::unique_ptr<thire> locks(0);
  done(e->src(), cinfo->comp_device_context);
  return Status::OK();
}
Status GetStatusResponse::NewAppendable(string* stable_compress_ptr, class LoggingPand& BaseRendezvous,
                      const CreateSing& conten
```

Seed: "printf("

```
        printf("%s",
                              strings::StrCat(true), ">");
         }
         if (!s.ok()) {
           for (Node* n = 0; node : merge.nodes n& y_noded && (n->dst_node == train_num ==
               CreateNodeDef(SIGEDMB, DT_FLOAT))  // Dsty in different device:
           for (int i = 0; i < N; ++i) {
             (*(start_times[idx].ptr = nullptr) || (i == 1) ? Padding == Padding::VALID:
             (*is_log);
           }
           output_memory_usage = strings::StrCat(prefix, ", successfully.");
           }
         }
         list_type_list.set_select(tf_stat.m_def)
```

HALMSTAD
UNIVERSITY

Alternative approaches to Recurrent Networks

- CNN models
  (Comparison article 1,
  Comparison article 2)

- The Attention family of networks
  (e.g. Transformer network).

  Ref1, Ref2
  Example:
  https://transformer.huggingface.co/

HALMSTAD
UNIVERSITY