9. Sequential Neural Models

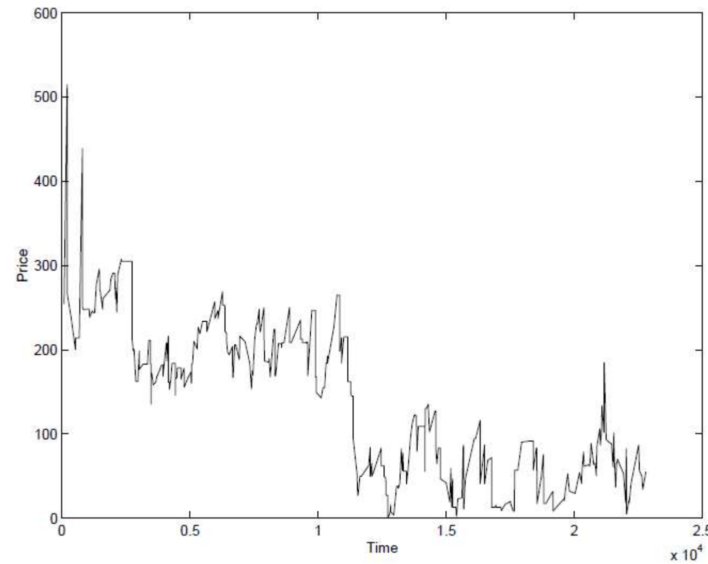
CS 519 Deep Learning, Winter 2018

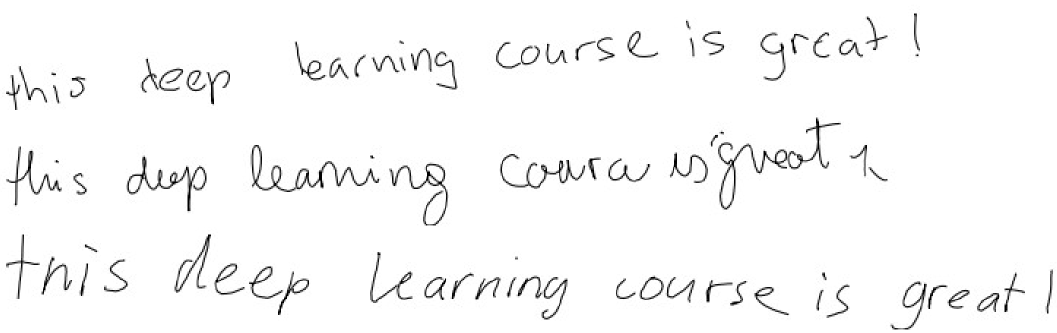
Fuxin Li

With materials from Andrej Karpathy, Bo Xie, Zsolt Kira

Sequential and Temporal Data

- Many applications exhibited by dynamically changing states
  - Language (e.g. sentences)
  - Temporal data
    - Speech
    - Stock Market





AAGTCAAGCTGCTCTGTGGGCTGTGATCTGCCTCAAACCCACAGCCTGGGTAGCAGG
AGGACCTTGATGCTCCTGGCACAGATGAGGAGAATCTCTCTTTTCTCCTGCTTGAAG
GACAGACATGACTTTGGATTTCCCCAGGAGGAGTTTGGCAACCAGTTCCAAAAGGCT
GAAACCATCCCTGTCCTCCATGAGATGATCCAGCAGATCTTCAATCTCTTCAGCACA
AAGGACTCATCTGCTGCTTGGGATGAGACCCTCCTAGACAAATTCTACACTGAACTC
TACCAGCAGCTGAATGACCTGGAAGCCTGTGTGATACAGGGGGTGGGGGTGACAGAG
ACTCCCCTGATGAAGGAGGACTCCATTCTGGCTGTGAGGAAATACTTCCAAAGAATC
ACTCTCTATCTGAAAGAGAAGAAATACAGCCCTTGTGCCTGGGAGGTTGTCAGAGCA
GAAATCATGAGATCTTTTTCTTTGTCAACAAACTTGCAAGAAAGTTTAAGAAGTAAG
GAATGA, TGTGATCTGCCTCAAACCCACAGCCTGGGTAGCAGGAGGACCTTGATGC
TCCTGGCACAGATGAGGAGAATCTCTCTTTTCTCCTGCTTGAAGGACAGACATGACT
TTGGATTTCCCCAGGAGGAGTTTGGCAACCAGTTCCAAAAGGCTGAAACCATCCCTG
TCCTCCATGAGATGATCCAGCAGATCTTCAATCTCTTCAGCACAAAGGACTCATCTG
CTGCTTGGGATGAGACCCTCCTAGACAAATTCTACACTGAACTCTACCAGCAGCTGA
ATGACCTGGAAGCCTGTGTGATACAGGGGGTGGGGGTGACAGAGACTCCCCTGATGA
AGGAGGACTCCATTCTGGCTGTGAGGAAATACTTCCAAAGAATCACTCTCTATCTGA
AAGAGAAGAAATACAGCCCTTGTGCCTGGGAGGTTGTCAGAGCAGAAATCATGAGAT
CTTTTTCTTTGTCAACAAACTTGCAAGAAAGTTTAAGAAGTAAGGAATGA and

Image Captioning

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

"girl in pink dress is jumping in air."

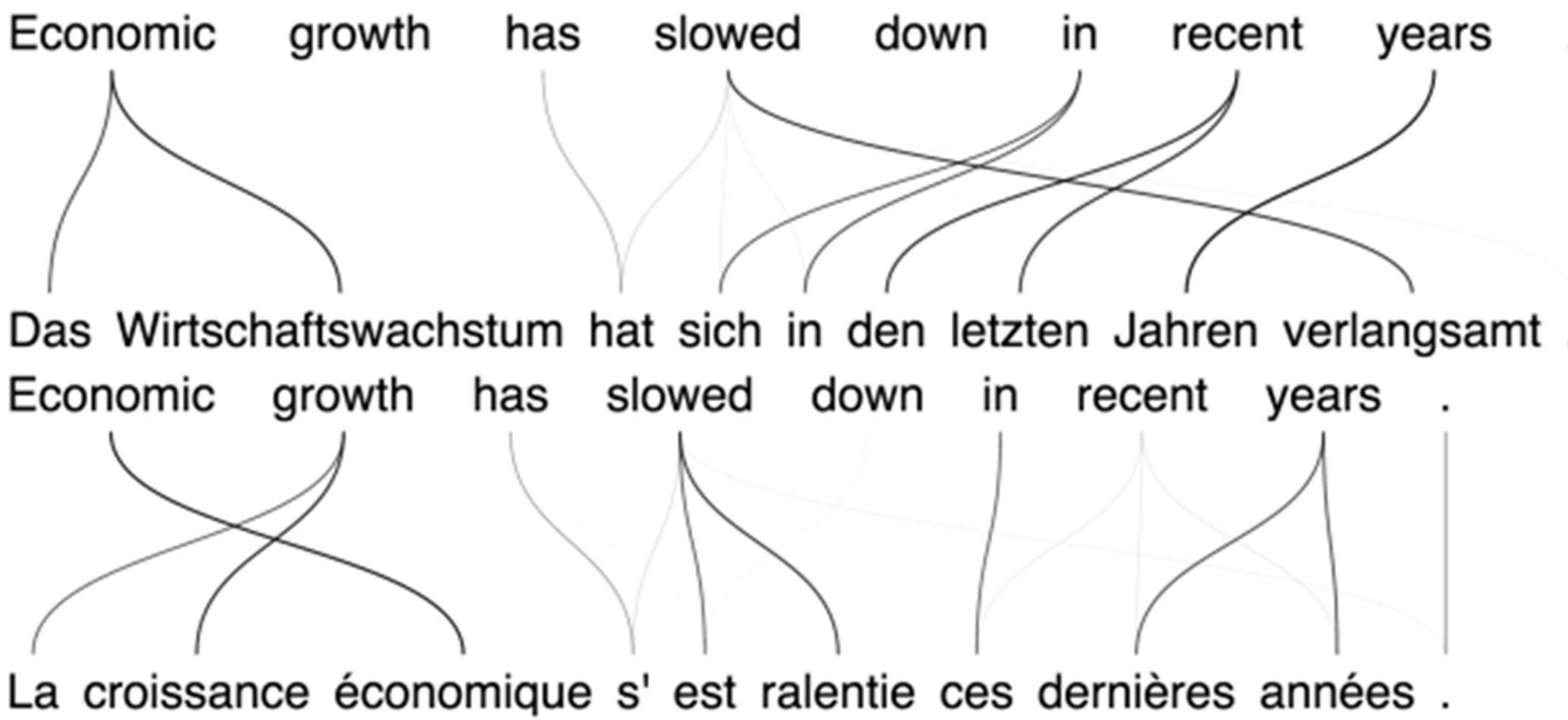
"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

"man in blue wetsuit is surfing on wave."

Machine Translation

- Have to look at the entire sentence (or, many sentences)



Sequence Data

- Many data are sequences and have different inputs/outputs

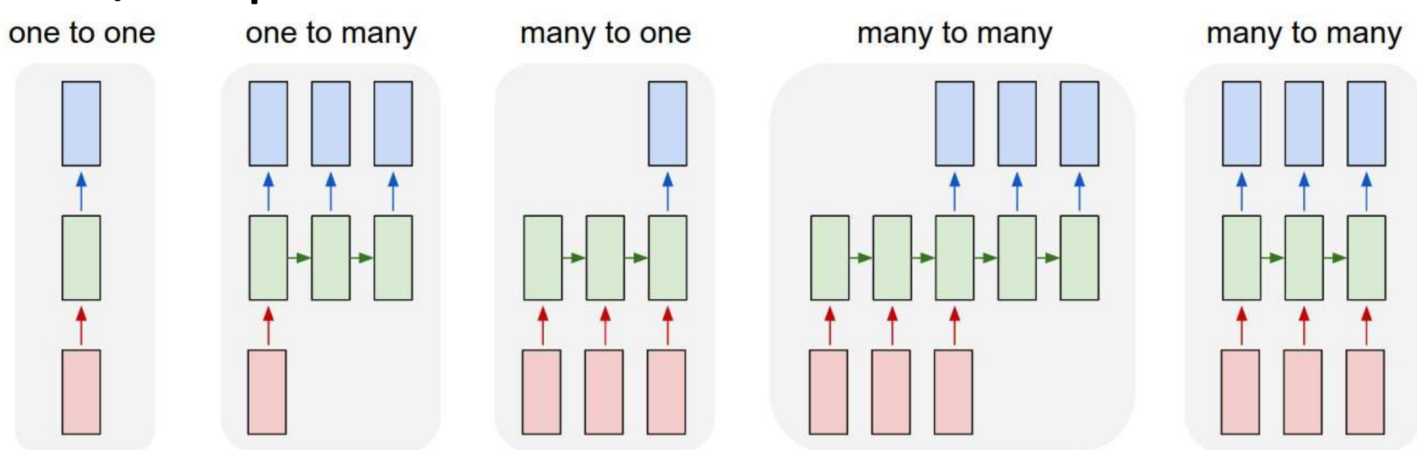
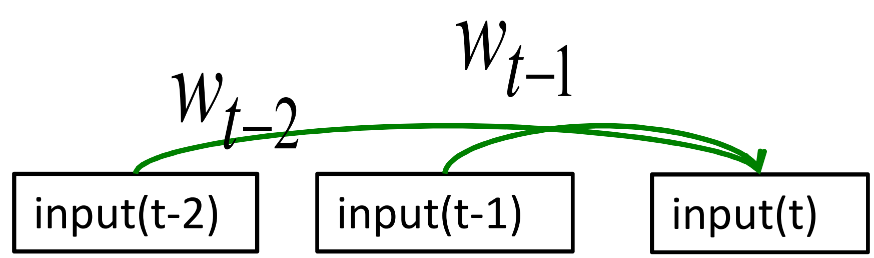


Image classification	Image captioning	Sentiment Analysis	Machine Translation	Video Classification

(cf. Andrej Karpathy blog)

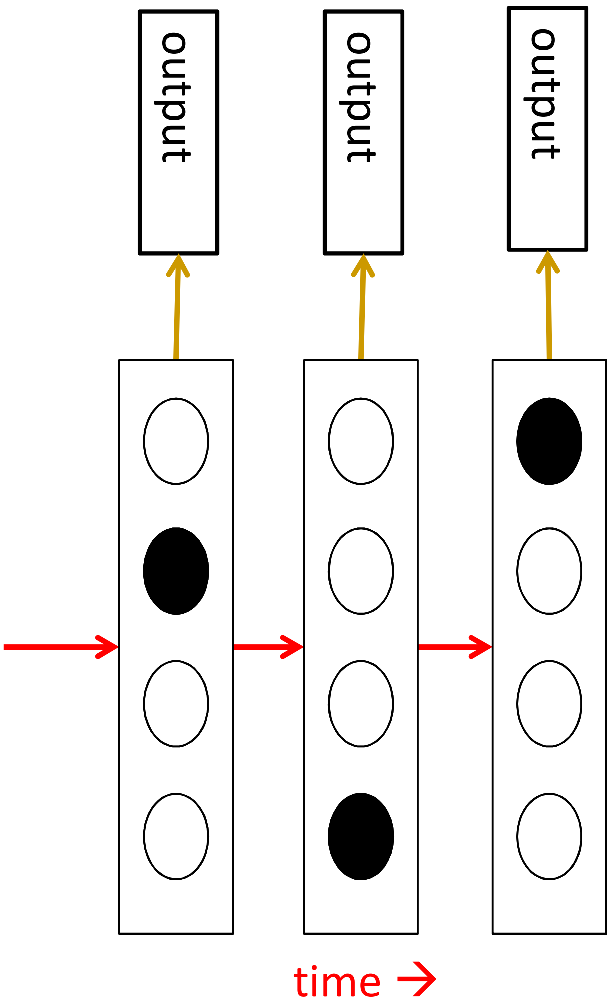
Previous: Autoregressive Models

- Autoregressive models
  - Predict the next term in a sequence from a fixed number of previous terms using “delay taps”.
- Neural Autoregressive models
  - Use neural net to do so



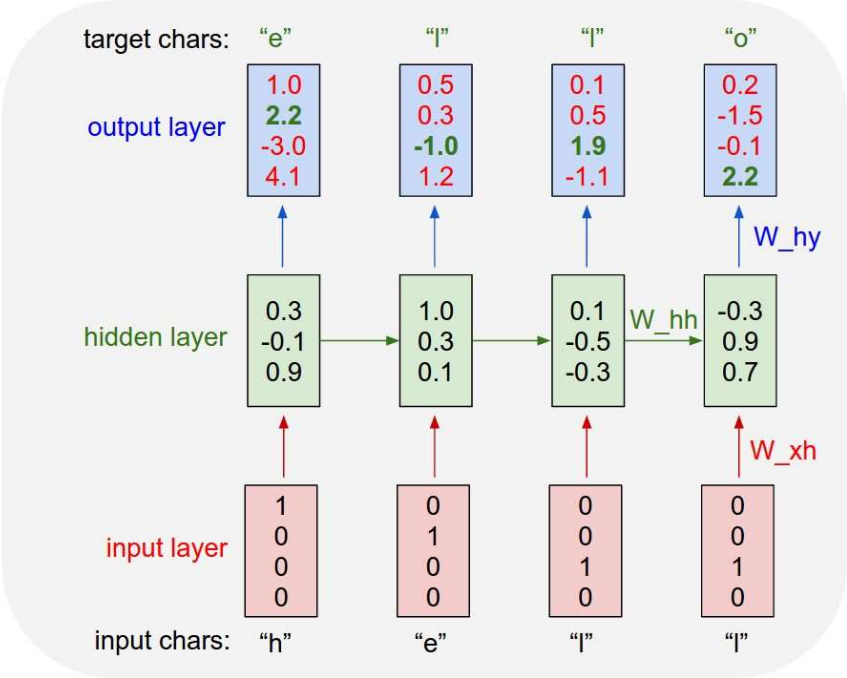
Previous: Hidden Markov Models

- Hidden states
- Outputs are generated from hidden states
  - Does not accept additional inputs
  - Discrete state-space
    - Need to learn all discrete transition probabilities!



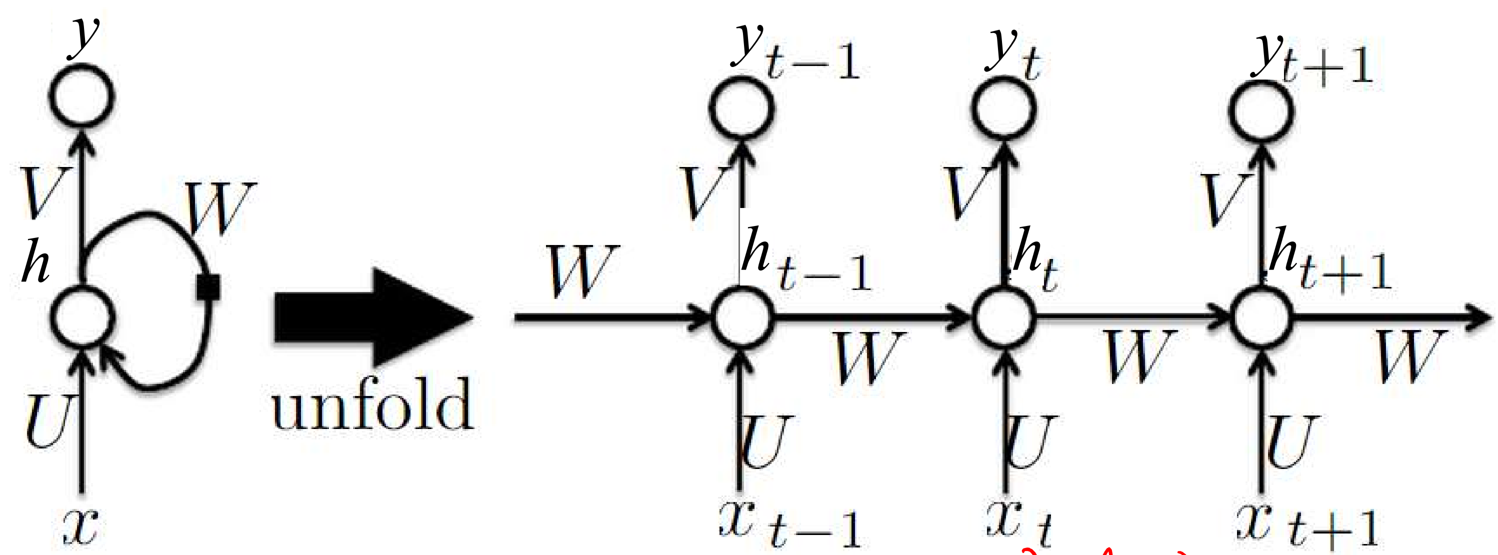
Recurrent Neural Networks

- Similar to
  - Linear Dynamic Systems
    - E.g. Kalman filters
  - Hidden Markov Models
  - But not generative
- “Turing-complete”



(cf. Andrej Karpathy blog)

Vanilla RNN Flow Graph



$$\boldsymbol{a}_t = \boldsymbol{b} + \mathbf{W}\boldsymbol{h}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t$$
$$\boldsymbol{h}_t = \tanh(\boldsymbol{a}_t)$$
$$\boldsymbol{y}_t = \boldsymbol{c} + \mathbf{V}\boldsymbol{h}_t$$

U – input to hidden

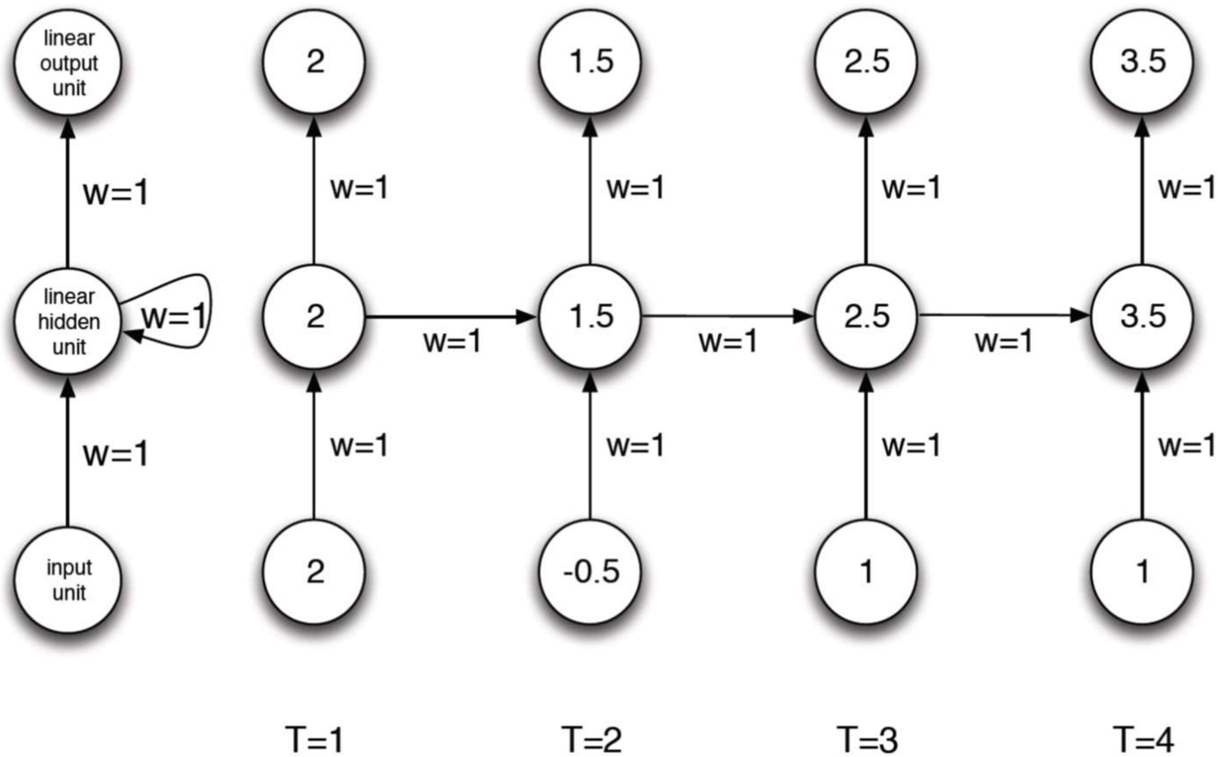
V – hidden to output

W – hidden to hidden

Examples

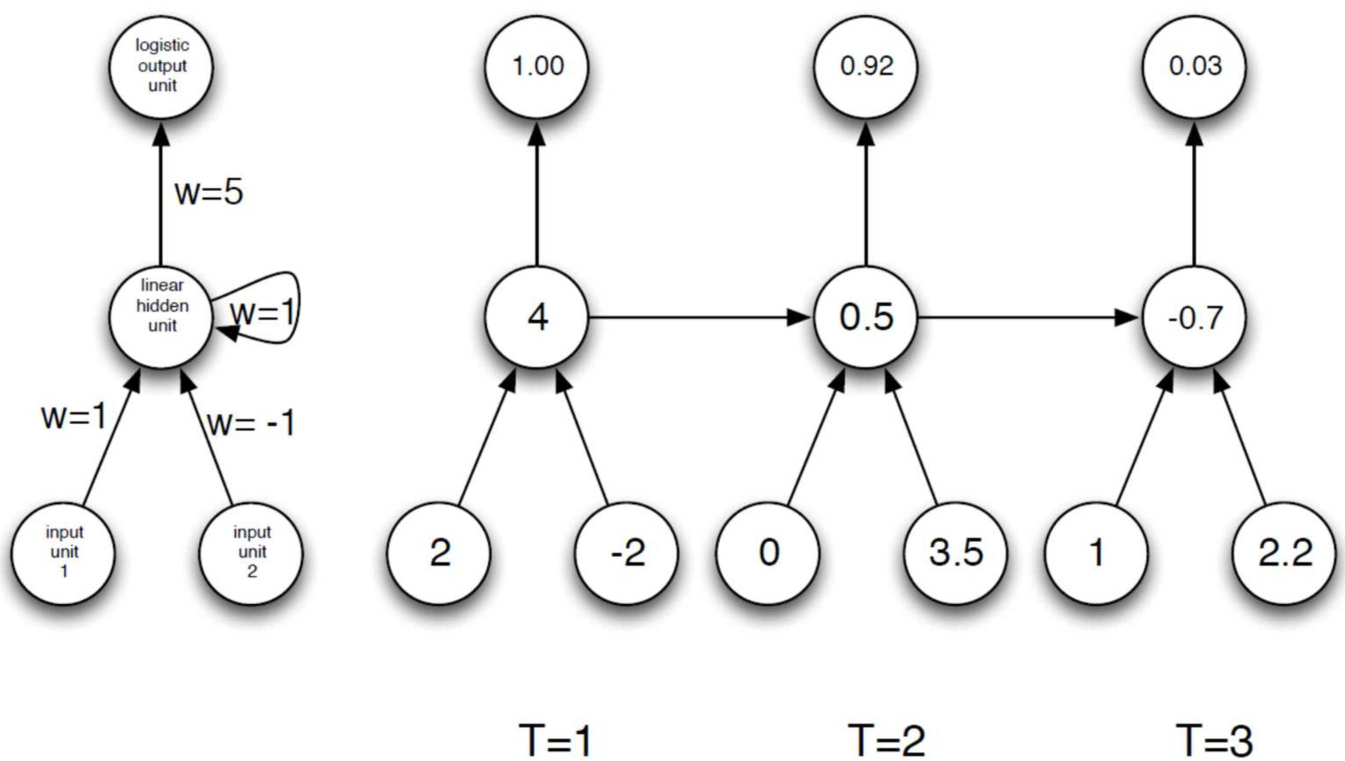
Now let's look at some simple examples of RNNs.

This one sums its inputs:



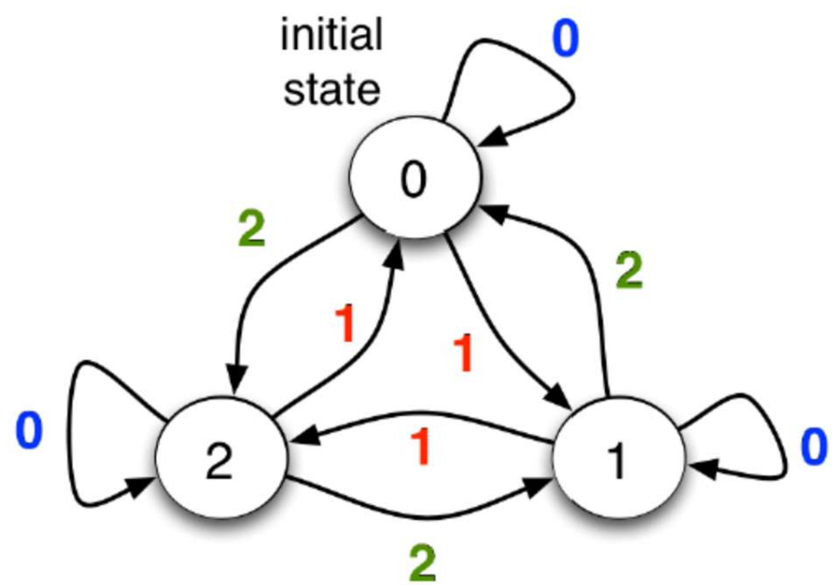
Examples

This one determines if the total values of the first or second input are larger:



Finite State Machines

- Each node denotes a state
- Reads input symbols one at a time
- After reading, transition to some other state
  - e.g. DFA, NFA
- States = hidden units



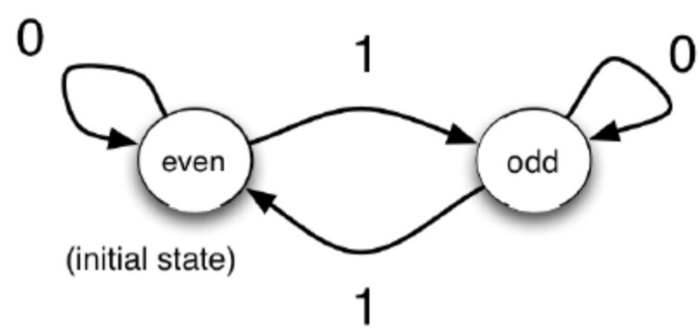
The parity Example

Assume we have a sequence of binary inputs. We'll consider how to determine the parity, i.e. whether the number of 1's is even or odd.

We can compute parity incrementally by keeping track of the parity of the input so far:

Parity bits: 0 1 1 0 1 1 $\longrightarrow$
Input: 0 1 0 1 1 0 1 0 1 1

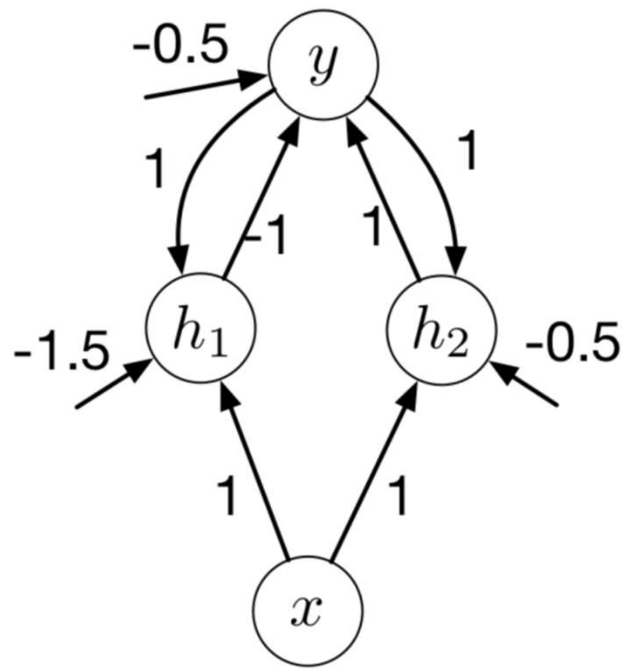
Each parity bit is the XOR of the input and the previous parity bit.



RNN Parity

- At each time step, compute parity between input vs. previous parity bit

$y^{(t-1)}$	$x^{(t)}$	$h_1^{(t)}$	$h_2^{(t)}$	$y^{(t)}$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0



RNN Universality

- RNN can simulate any finite state machines
  - is Turing complete with infinite hidden nodes (Siegelmann and Sontag, 1995)
  - e.g., a computer (Zaremba and Sutskever 2014)

Training data:

Input:
```
j=8584
for x in range(8):
  j+=920
b=(1500+j)
print((b+7567))
```
Target: 25011.

Input:
```
i=8827
c=(i-5347)
print((c+8704) if 2641<8500 else 5308)
```
Target: 12184.

RNN Universality

- Testing programs

Input:

```
d=8640;
print((7135 if 6710>((d+7080)*14) else 7200)).
```

Target:	7200.
"Baseline" prediction:	7200.
"Naive" prediction:	7200.
"Mix" prediction:	7200.
"Combined" prediction:	7200.

Input:

```
print((((841 if 2076<7326 else 1869)*10) if 7827<317 else 7192)).
```

Target:	7192.
"Baseline" prediction:	7192.
"Naive" prediction:	7192.
"Mix" prediction:	7192.
"Combined" prediction:	7192.

RNN Universality
(if only you can train it!)

Input:

```
print((4*7054)).
```

Target:	28216.
"Baseline" prediction:	28216.
"Naive" prediction:	28116.
"Mix" prediction:	28216.
"Combined" prediction:	28216.

Input:

```
print((4635-5257)).
```

Target:	-622.
"Baseline" prediction:	-688.
"Naive" prediction:	-628.
"Mix" prediction:	-692.
"Combined" prediction:	-632.

Input:

```
e=1079
for x in range(10):e+=4729
print(e).
```

Target:	48369.
"Baseline" prediction:	48017.
"Naive" prediction:	48011.
"Mix" prediction:	48101.
"Combined" prediction:	48009.

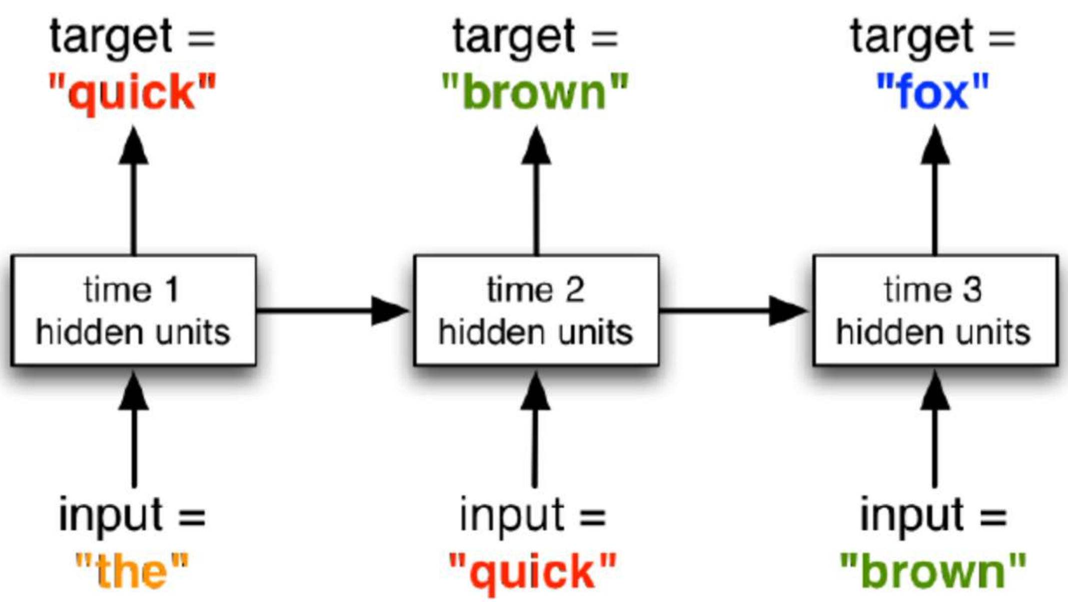
Input:

```
print((8*(5051-648))).
```

Target:	35224.
"Baseline" prediction:	34044.
"Naive" prediction:	32180.
"Mix" prediction:	33284.
"Combined" prediction:	33004.

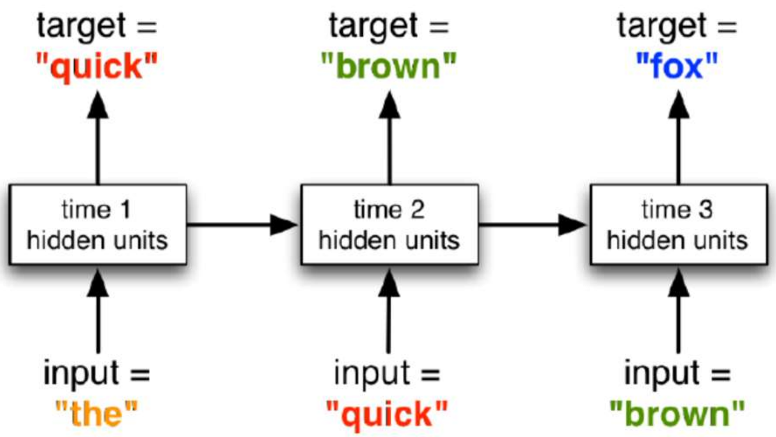
RNN Text Model

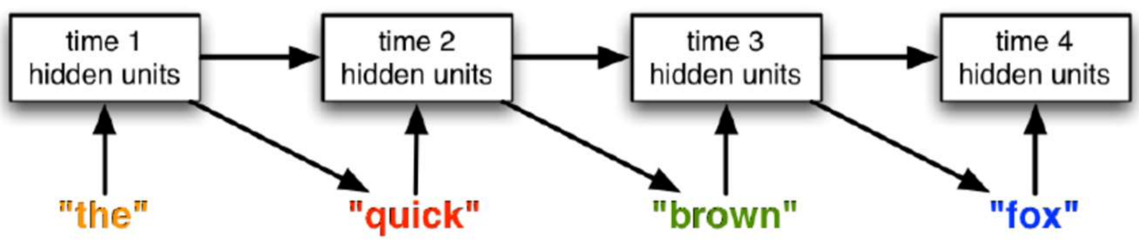
One way to use RNNs to model text:



Generate Text from RNN

One way to use RNNs to model text:





RNN Sentence Model

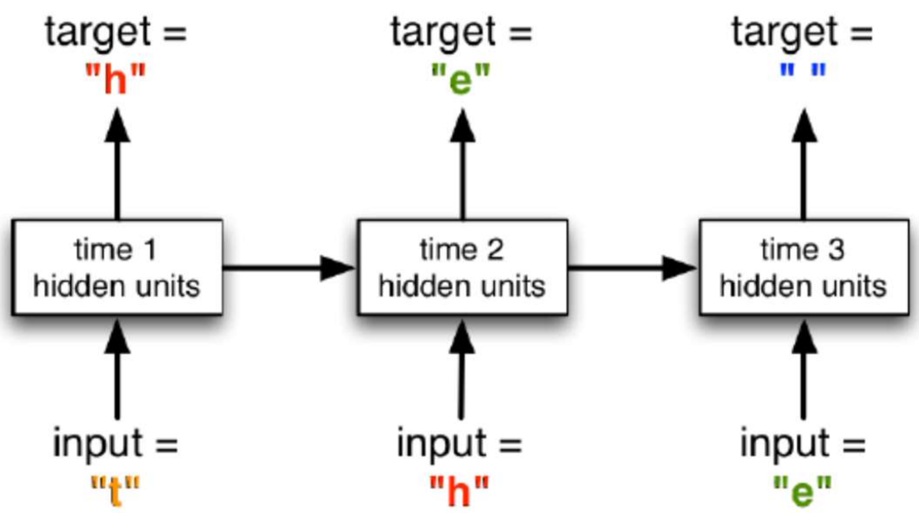
- Hypothetical: Different hidden units for:
  - Subject
  - Verb
  - Object (different type)

Realistic Ones

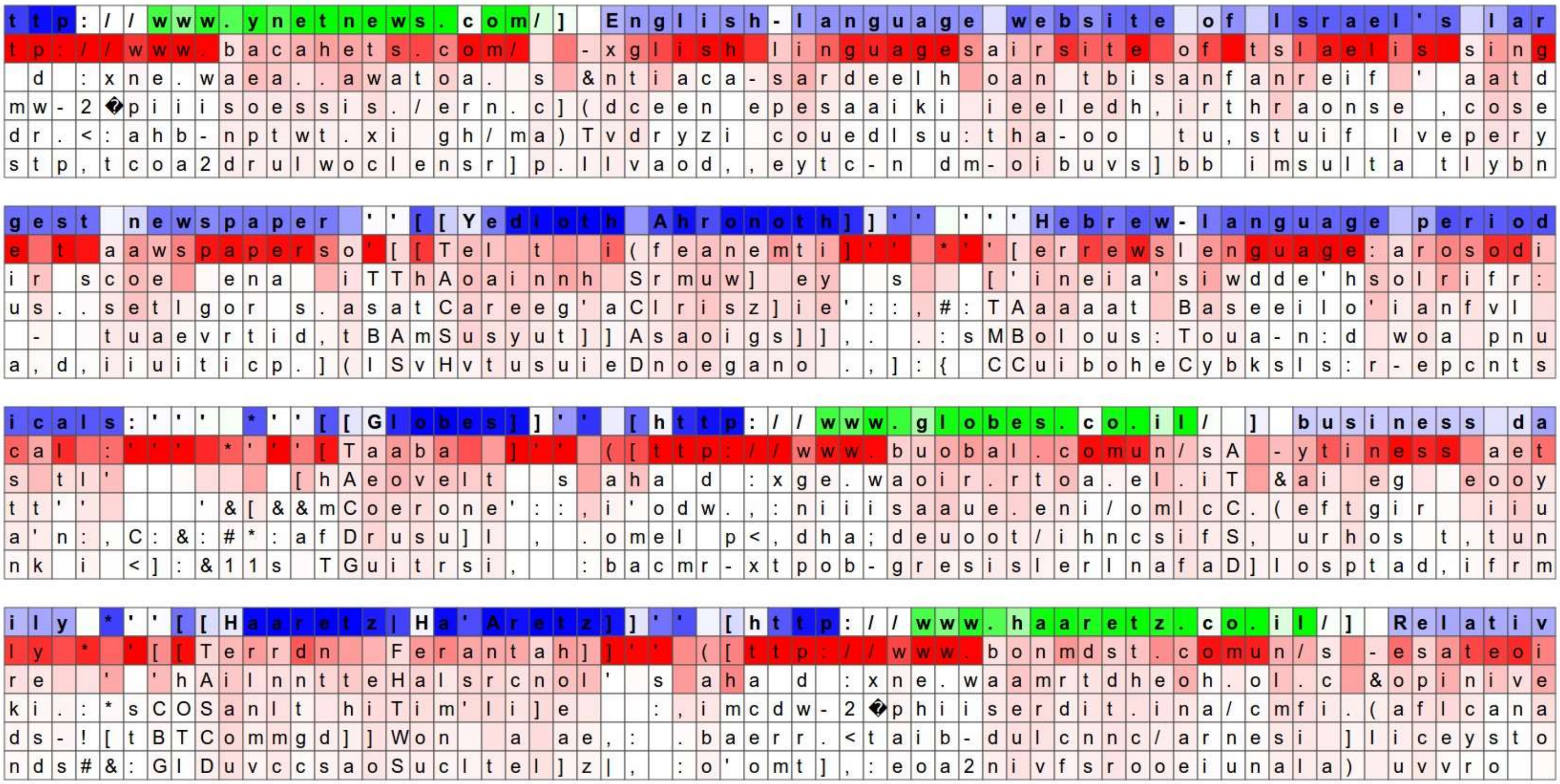
Query	cell 25	cell 26	cell 27	cell 30
al yo yo sauce				
atkins diet lasagna				diet
blender recipes		recipes		
cake bakery edinburgh		bakery		
canning corn beef hash		corn, beef		
torre de pizza		pizza		
famous desserts				
fried chicken		chicken		
smoked turkey recipes		recipes		
italian sausage hoagies		sausage		
do you get allergy				
much pain will after total knee replacement				
how to make whiter teeth		whiter		
illini community hospital				hospital
implant infection	infection			
introductory psychology				
narcotics during pregnancy side effects				
fight sinus infections	infections			
health insurance high blood pressure				insurance,high
all antidepressant medications	antidepressant			medications

RNN Character Model

Another approach is to model text *one character at a time*!

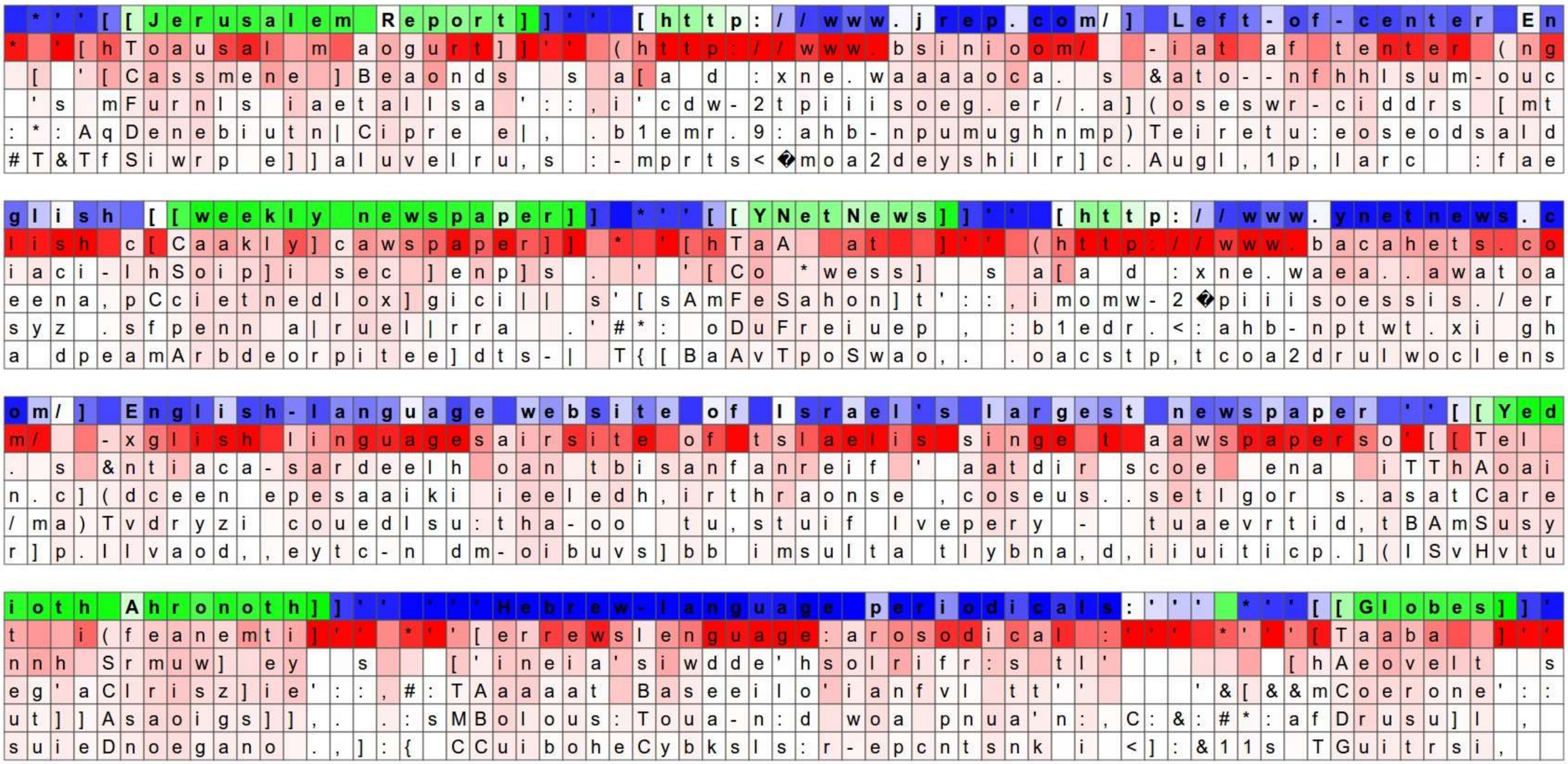


Realistic Wiki Hidden Unit



First row: Green for excited, blue for not excited
Next 5 rows: top-5 guesses for the next character

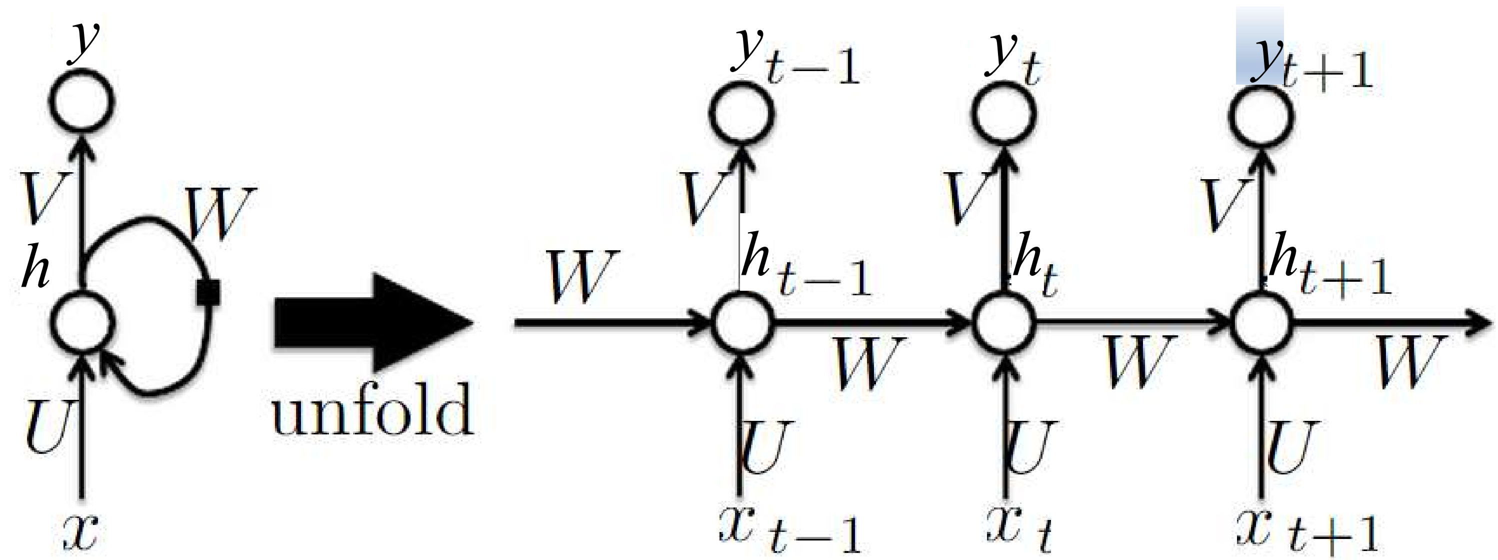
Realistic Wiki Hidden Unit



Above: Green for excited, blue for not excited

Below: top-5 guesses for the next character

Vanilla RNN Flow Graph



$$\boldsymbol{a}_t = \boldsymbol{b} + \mathbf{W}\boldsymbol{h}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t$$
$$\boldsymbol{h}_t = \tanh(\boldsymbol{a}_t)$$
$$\boldsymbol{y}_t = \boldsymbol{c} + \mathbf{V}\boldsymbol{h}_t$$

U – input to hidden

V – hidden to output

W – hidden to hidden

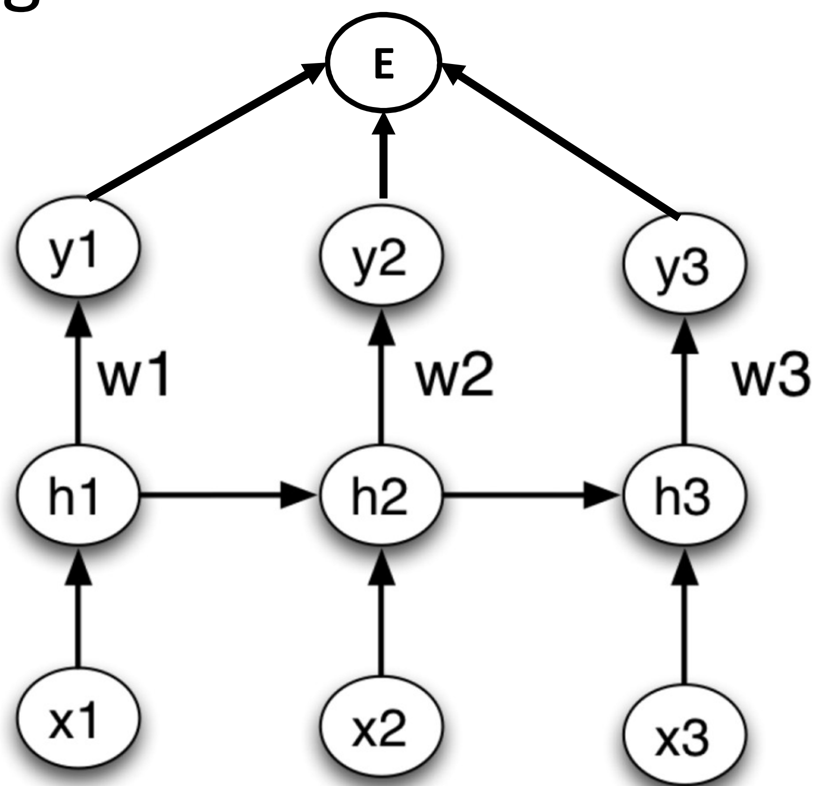
Training RNN

- "Backpropagation through time"

  = Backpropagation

- What to do with this if

  $$\boldsymbol{w} = \boldsymbol{w}_1 = \boldsymbol{w}_2 = \boldsymbol{w}_3?$$



$$\frac{\partial E}{\partial \boldsymbol{w}} = \frac{\partial E}{\partial \boldsymbol{y}_1}\frac{\partial \boldsymbol{y}_1}{\partial \boldsymbol{w}} + \frac{\partial E}{\partial \boldsymbol{y}_2}\frac{\partial \boldsymbol{y}_2}{\partial \boldsymbol{w}} + \frac{\partial E}{\partial \boldsymbol{y}_3}\frac{\partial \boldsymbol{y}_3}{\partial \boldsymbol{w}}$$

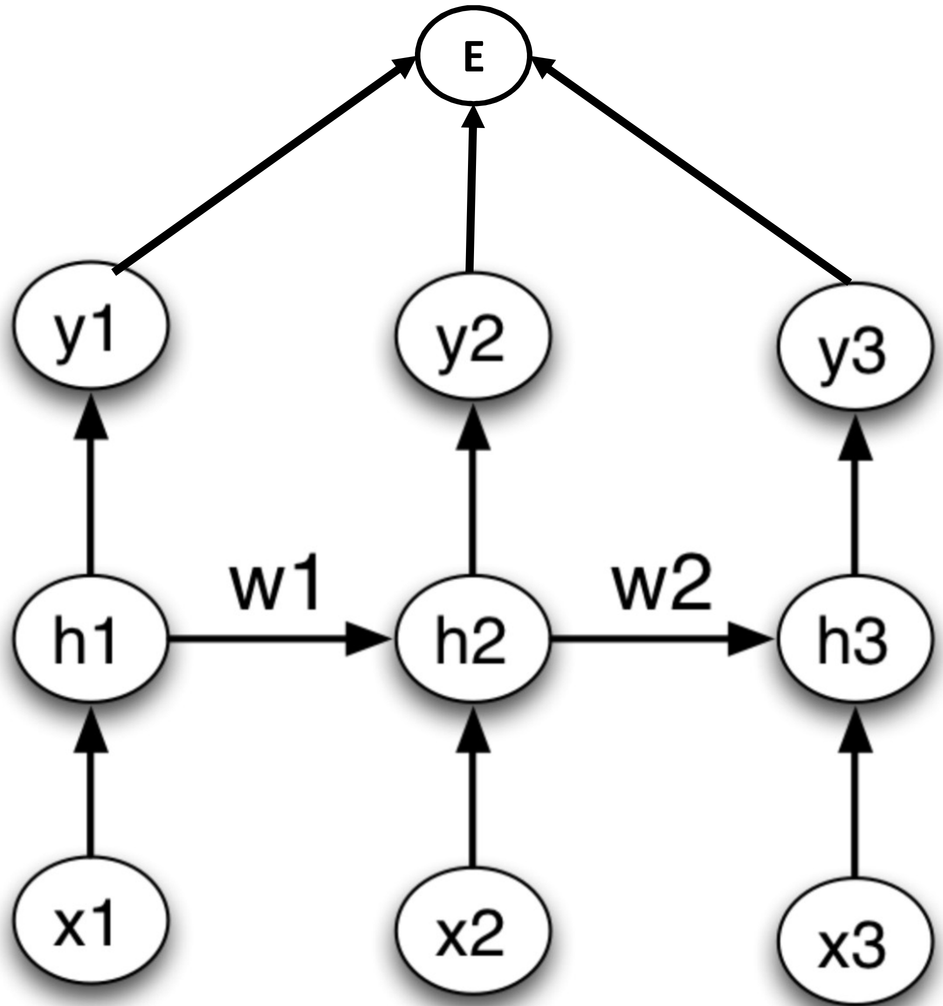
Training RNN

- Again, assume

$$\boldsymbol{w} = \boldsymbol{w}_1 = \boldsymbol{w}_2$$

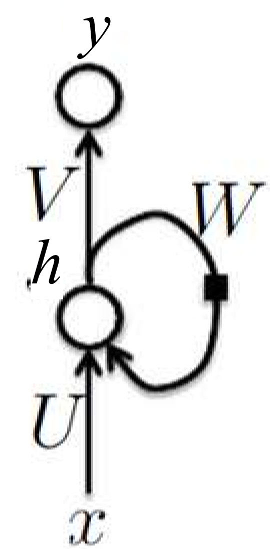
$$\frac{\partial E}{\partial \boldsymbol{w}} = \frac{\partial E}{\partial \boldsymbol{h}_2}\frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{w}} + \frac{\partial E}{\partial \boldsymbol{h}_3}\frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{w}}$$

$$\frac{\partial E}{\partial \boldsymbol{h}_2} = \frac{\partial E}{\partial \boldsymbol{y}_2}\frac{\partial \boldsymbol{y}_2}{\partial \boldsymbol{h}_2} + \frac{\partial E}{\partial \boldsymbol{h}_3}\frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{h}_2}$$



k timesteps?

$$\frac{\partial E}{\partial \boldsymbol{h}_2} = \frac{\partial E}{\partial \boldsymbol{y}_k}\frac{\partial \boldsymbol{y}_k}{\partial \boldsymbol{h}_k}\frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{h}_{k-1}} \dots \frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{h}_2} + \frac{\partial E}{\partial \boldsymbol{y}_{k-1}}\frac{\partial \boldsymbol{y}_{k-1}}{\partial \boldsymbol{h}_{k-1}} \dots \frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{h}_2} + \cdots$$



- What's the problem?

$$\frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{h}_{k-1}} = \frac{\partial \boldsymbol{h}_{k-1}}{\partial \boldsymbol{h}_{k-2}} = \cdots = \tanh(\boldsymbol{a}_t)' \, \mathbf{W}$$

- There are terms like $\mathbf{W}^k$ in the gradient

$$\boldsymbol{a}_t = \boldsymbol{b} + \mathbf{W}\boldsymbol{h}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t$$
$$\boldsymbol{h}_t = \tanh(\boldsymbol{a_t})$$
$$\boldsymbol{y}_t = \boldsymbol{c} + \mathbf{V}\boldsymbol{h_t}$$

What's wrong with $\mathbf{W}^k$?

- Suppose $\mathbf{W}$ is diagonlizable for simplicity

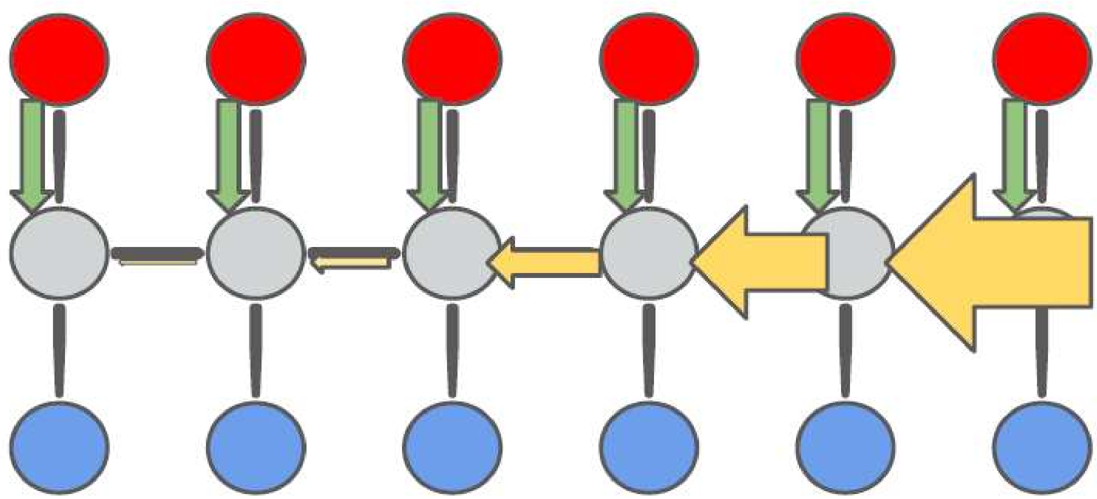
$$\mathbf{W} = \mathbf{U}\mathbf{D}\mathbf{U}^\top$$
$$\mathbf{W}^k = \mathbf{U}\mathbf{D}^k\mathbf{U}^\top$$

- What if,
  - W has an eigenvalue of 4?
  - W has an eigenvalue of 0.25?
  - Both?

Cannot train it with backprop

Vanishing gradients

- Vanishing long term gradient (g^t is very small if $g < 1$)
- Strong short term gradient



Do we need long-term gradients?

- Long-term dependency is one main reason we want temporal models
  - Example:

Rob Ford told the flabbergasted reporters assembled at the press conference that __________.

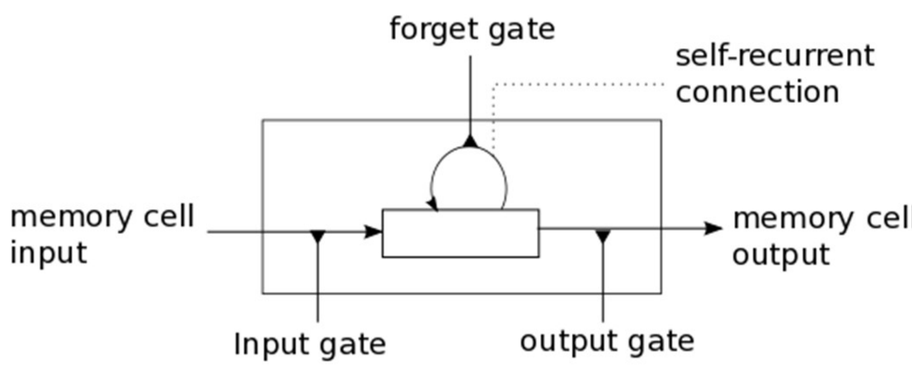
German for "travel"

Die Koffer waren gepackt, und er ***reiste***, nachdem er seine Mutter und seine Schwestern geküsst und noch ein letztes Mal sein angebetetes Gretchen an sich gedrückt hatte, das, in schlichten weißen Musselin gekleidet und mit einer einzelnen Nachthyazinthe im üppigen braunen Haar, kraftlos die Treppe herabgetaumelt war, immer noch blass von dem Entsetzen und der Aufregung des vorangegangenen Abends, aber voller Sehnsucht, ihren armen schmerzenden Kopf noch einmal an die Brust des Mannes zu legen, den sie mehr als ihr eigenes Leben liebte, ***ab***."

Only now we are sure the travel started, not ended (reiste an)

LSTM: Long short-term Memory

- Need memory!
  - Vanilla RNN has volatile memory (automatically transformed every time-step)
  - More “fixed” memory stores info longer so errors don’t need to be propagated very far
- Complex architecture with memory



LSTM Starting point

- Instead of using volatile state transition

$$\boldsymbol{h}_t = \tanh(\mathbf{W}\boldsymbol{h}_{t-1} + \mathbf{U}\boldsymbol{x}_{\boldsymbol{t}} + b)$$

- Use fixed transition and learn the difference

$$\boldsymbol{c}_t = \boldsymbol{c}_{t-1} + \tanh(\mathbf{W}\boldsymbol{y}_{t-1} + \mathbf{U}\boldsymbol{x}_t + b)$$

  – Now we can truncate the BPTT safely after several timesteps

- However, this has the drawback of $\boldsymbol{c}_t$ being stored for too long
  – Add a weight? (subject to vanishing as well)
  – Add an "adaptive weight"

Forget Gate

- Decide how much of c_{t-1} should we forget

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{z}_t$$
$$\boldsymbol{z}_t = \tanh(\mathbf{W}\boldsymbol{y}_{t-1} + \mathbf{U}\boldsymbol{x}_t + b)$$

- Forget neurons also trained

$$f_t = \sigma(\mathbf{W}_f \boldsymbol{x}_t + \mathbf{R}_f \boldsymbol{y}_{t-1} + \boldsymbol{p}_f \odot \boldsymbol{c}_{t-1} + b_f)$$

- How much we forget is dependent on:
  - Previous output
  - Current input
  - Previous memory

Input Modulation

- Memory is supposed to be "persistent"
- Some input might be corrupt and should not affect our memory
- We may want to decide which input affects our memory
- Input Gate:

$$\mathbf{i}^t = \sigma(\mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i) \quad \textit{input gate}$$

- Final memory update:

$$\mathbf{c}^t = \mathbf{i}^t \odot \mathbf{z}^t + \mathbf{f}^t \odot \mathbf{c}^{t-1}$$

Output Modulation

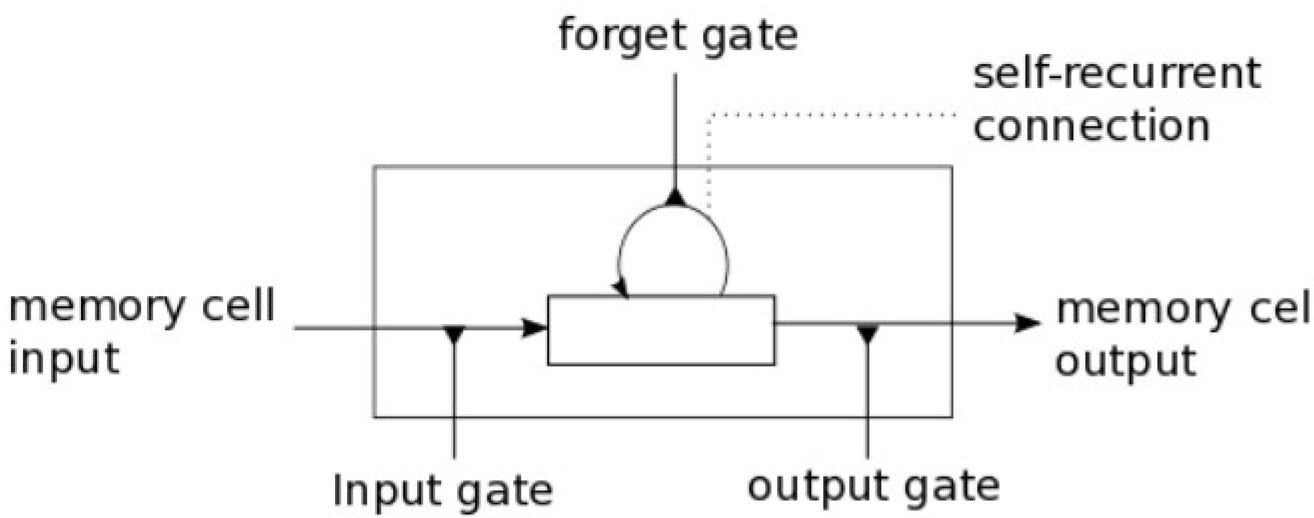
- Do not always “tell” what we remembered

$$\mathbf{o}^t = \sigma(\mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^t + \mathbf{b}_o) \quad \textit{output gate}$$
$$\mathbf{y}^t = \mathbf{o}^t \odot h(\mathbf{c}^t) \quad \textit{block output}$$

- Only output if we “feel like it”
- The output part can vary a lot depending on applications

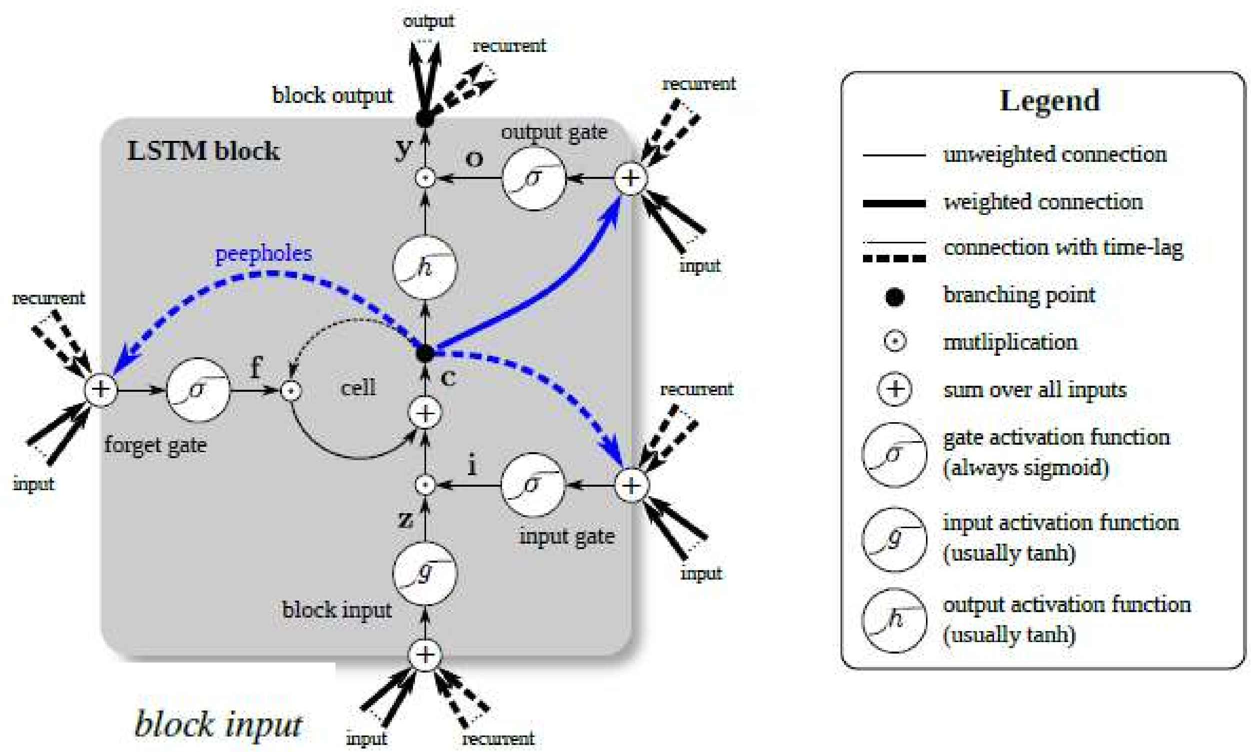
LSTM

- Hochreiter & Schmidhuber (1997)
- Use gates to remember things for a long period of time
- Use gates to modulate input and output



LSTM Architecture

- “Official version” with a lot of peepholes

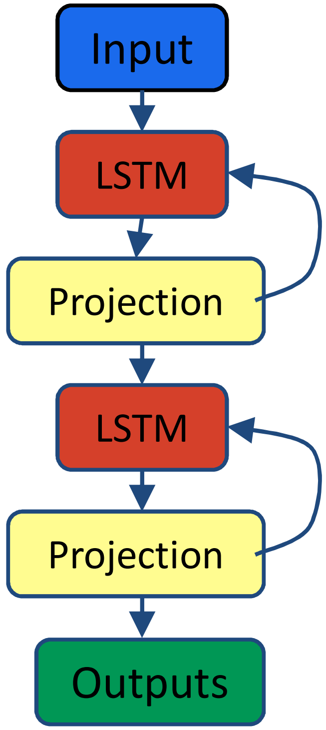


$$\mathbf{z}^t = g(\mathbf{W}_z \mathbf{x}^t + \mathbf{R}_z \mathbf{y}^{t-1} + \mathbf{b}_z) \qquad \textit{block input}$$
$$\mathbf{i}^t = \sigma(\mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i) \qquad \textit{input gate}$$
$$\mathbf{f}^t = \sigma(\mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{y}^{t-1} + \mathbf{p}_f \odot \mathbf{c}^{t-1} + \mathbf{b}_f) \qquad \textit{forget gate}$$
$$\mathbf{c}^t = \mathbf{i}^t \odot \mathbf{z}^t + \mathbf{f}^t \odot \mathbf{c}^{t-1} \qquad \textit{cell state}$$
$$\mathbf{o}^t = \sigma(\mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^{t} + \mathbf{b}_o) \qquad \textit{output gate}$$
$$\mathbf{y}^t = \mathbf{o}^t \odot h(\mathbf{c}^t) \qquad \textit{block output}$$

Cf. LSTM: a search space odyssey

Google Speech recognition

- Task:
  - Google Now/Voice search / mobile dictation
  - Streaming, real-time recognition in 50 languages
- Model:
  - Deep Projection Long-Short Term Memory Recurrent Neural networks
  - Distributed training with asynchronous gradient descent across hundreds of machines.
  - Cross-entropy objective (truncated backpropagation through time) followed by sequence discriminative training (sMBR).
  - 40-dimensional filterbank energy inputs
  - Predict 14,000 acoustic state posteriors



Slide provided by Andrew Senior, Vincent Vanhoucke, Hasim Sak (June 2014)

LSTM Large vocabulary speech recognition

Google

Models	Parameters	Cross-Entropy	sMBR sequence training
ReLU DNN	85M	11.3	10.4
Deep Projection LSTM RNN (2 layer)	13M	10.7	9.7

- ***Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling*** **H. Sak, A. Senior, F. Beaufays** to appear in Interspeech 2014
- ***Sequence Discriminative Distributed Training of Long Short-Term Memory Recurrent Neural Networks*** **H. Sak, O. Vinyals, G. Heigold A. Senior, E. McDermott, R. Monga, M. Mao** to appear in Interspeech 2014

Voice search task; Training data: 3M utterances (1900 hrs); models trained on CPU clusters

Slide provided by Andrew Senior, Vincent Vanhoucke, Hasim Sak (June 2014)

Bidirectional LSTM

Both forward and backward paths
Still DAG!

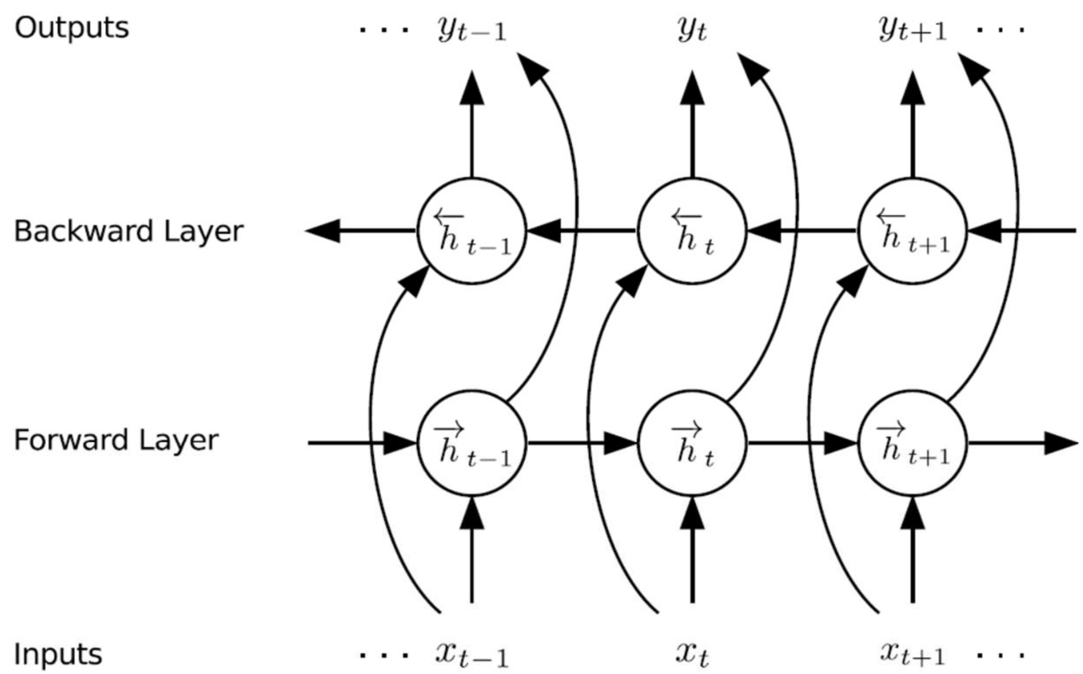
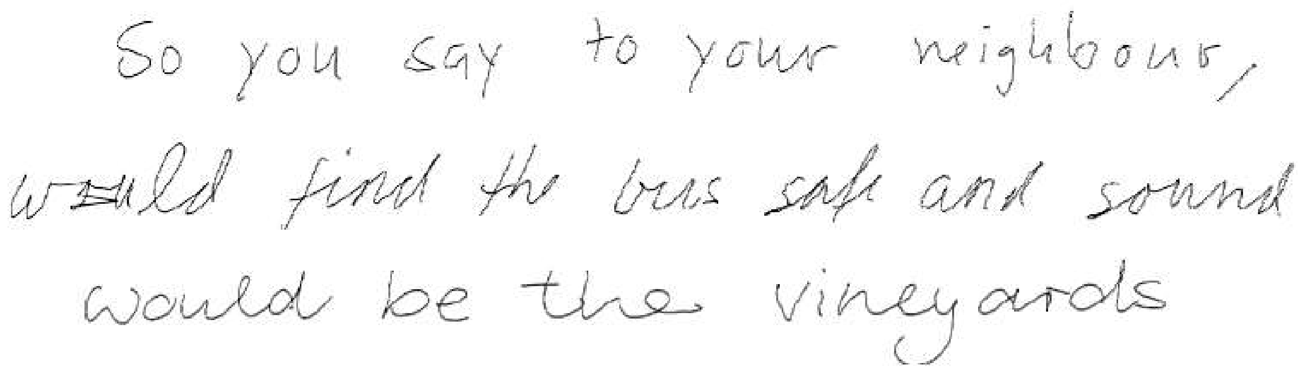


Fig. 2. Bidirectional Recurrent Neural Network

Pen trajectories

- Task: generate pen trajectories by predicting one (x,y) point at a time
- Data: IAM online handwriting, 10K training sequences, many writers, unconstrained style, captured from a whiteboard



- First problem: what to use for the density model?

Recurrent Mixture Density Networks

- Network outputs parameterise a mixture distribution (usually Gaussian)
- Every prediction conditioned on all inputs so far

$$\Pr(x_{t+1}|x_{1:t}) = \sum_k w_k(x_{1:t})\mathcal{N}(x_{t+1}|\mu_k(x_{1:t}), \Sigma_k(x_{1:t}))$$

- Number of components is number of *choices* for what comes next

- M. Schuster, "Better Generative Models for Sequential Data Problems: Bidirectional Recurrent Mixture Density Networks", NIPS 1999

Network details

$$x_t \in \mathbb{R} \times \mathbb{R} \times \{0,1\}$$
$$y_t = \left(e_t, \{\pi_t^j, \mu_t^j, \sigma_t^j, \rho_t^j\}_{j=1}^M\right)$$

$$\hat{y}_t = \left(\hat{e}_t, \{\hat{w}_t^j, \hat{\mu}_t^j, \hat{\sigma}_t^j, \hat{\rho}_t^j\}_{j=1}^M\right) = b_y + \sum_{n=1}^{N} W_{h^n y} h_t^n$$

$$e_t = \frac{1}{1 + \exp\left(\hat{e}_t\right)} \qquad \Longrightarrow e_t \in (0,1)$$

$$\pi_t^j = \frac{\exp\left(\hat{\pi}_t^j\right)}{\sum_{j'=1}^M \exp\left(\hat{\pi}_t^{j'}\right)} \qquad \Longrightarrow \pi_t^j \in (0,1), \quad \sum_j \pi_t^j = 1$$

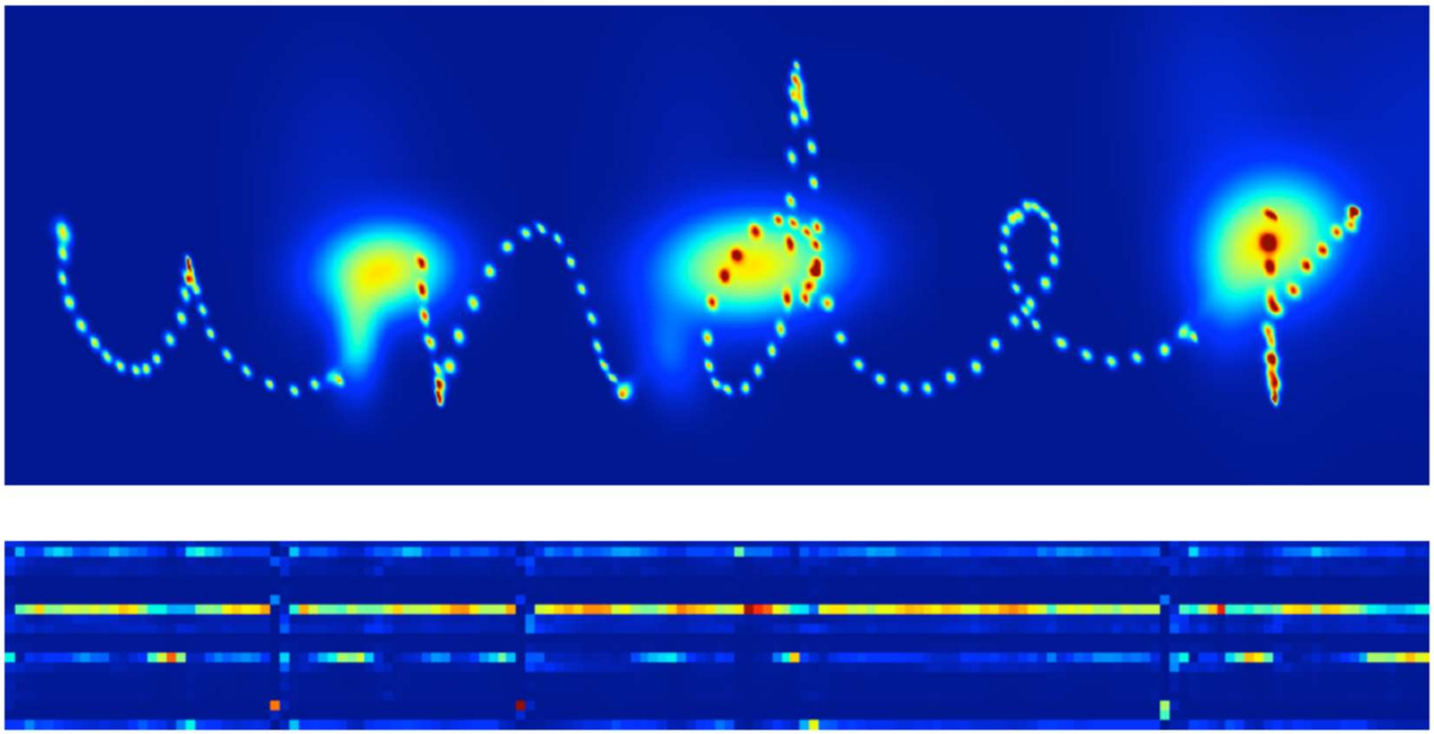
$$\mu_t^j = \hat{\mu}_t^j \qquad \Longrightarrow \mu_t^j \in \mathbb{R}$$

$$\sigma_t^j = \exp\left(\hat{\sigma}_t^j\right) \qquad \Longrightarrow \sigma_t^j > 0$$

$$\rho_t^j = tanh(\hat{\rho}_t^j) \qquad \Longrightarrow \rho_t^j \in (-1,1)$$

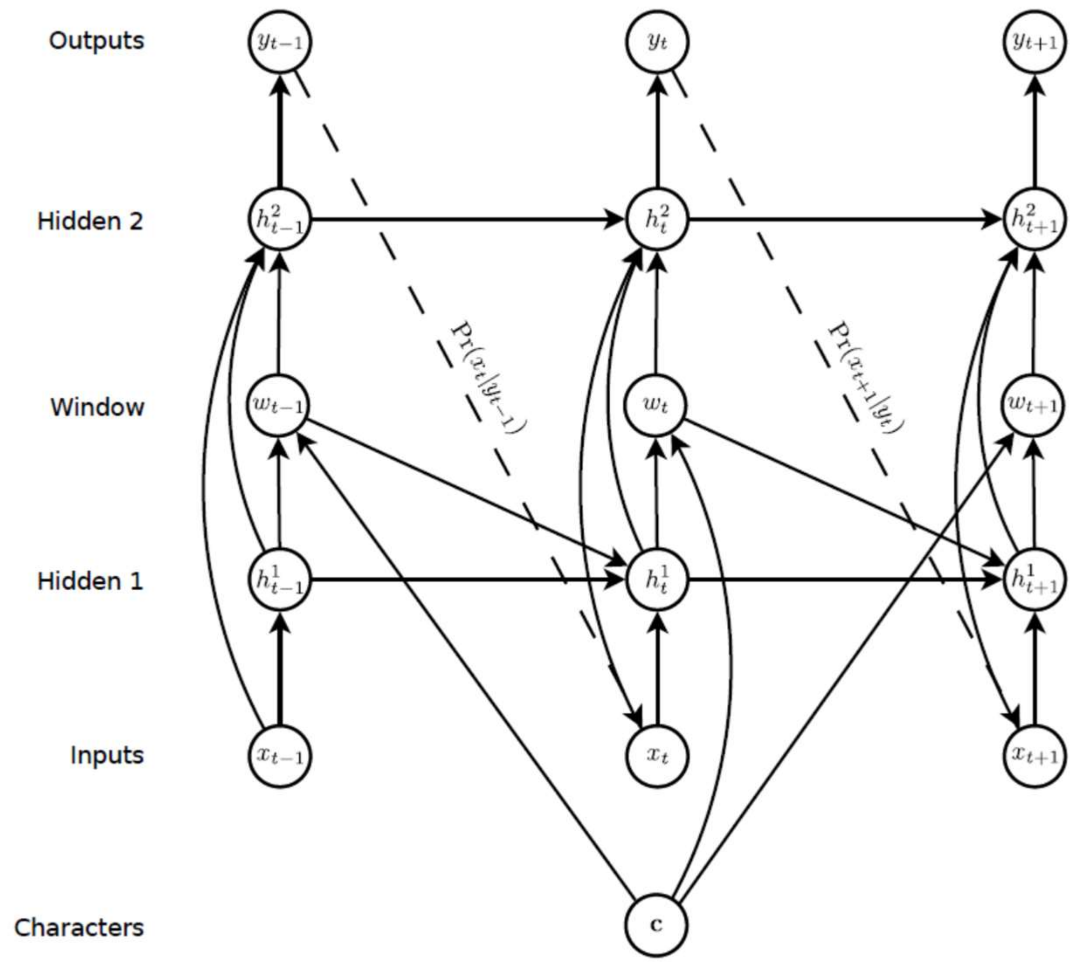
A. Graves, “Generating Sequences with Recurrent Neural Networks,
arXiv:1308.0850v5

Illustration of mixture density



Synthesis

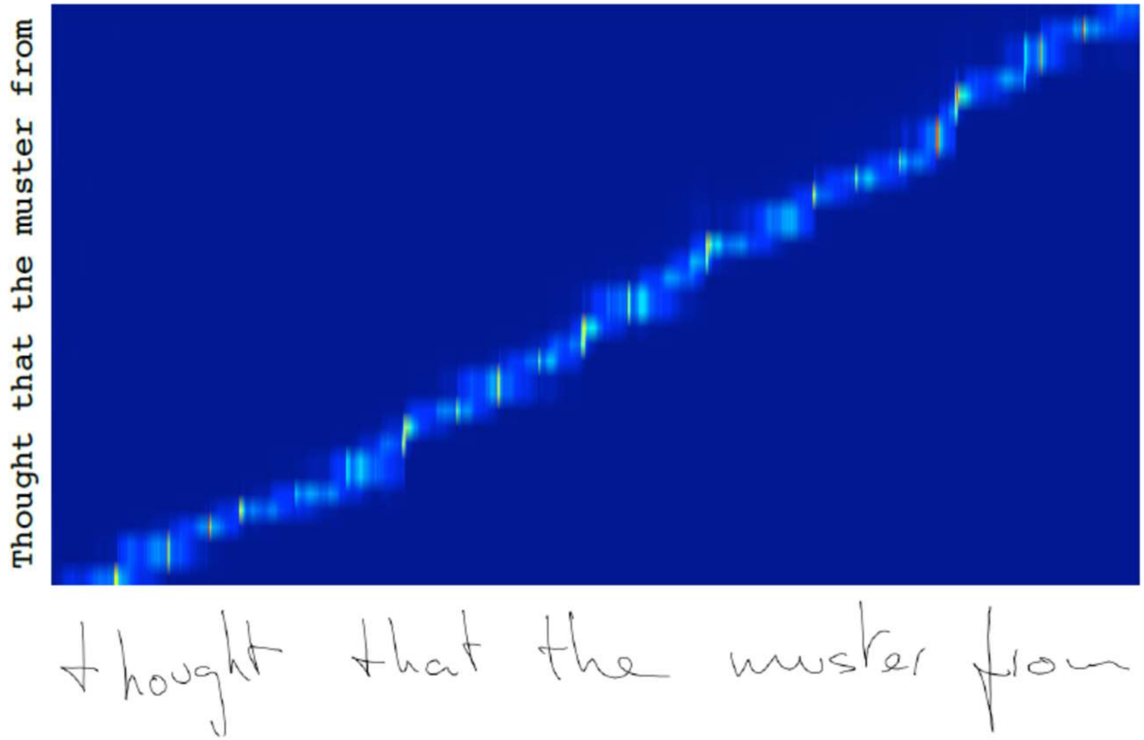
- Adding text input



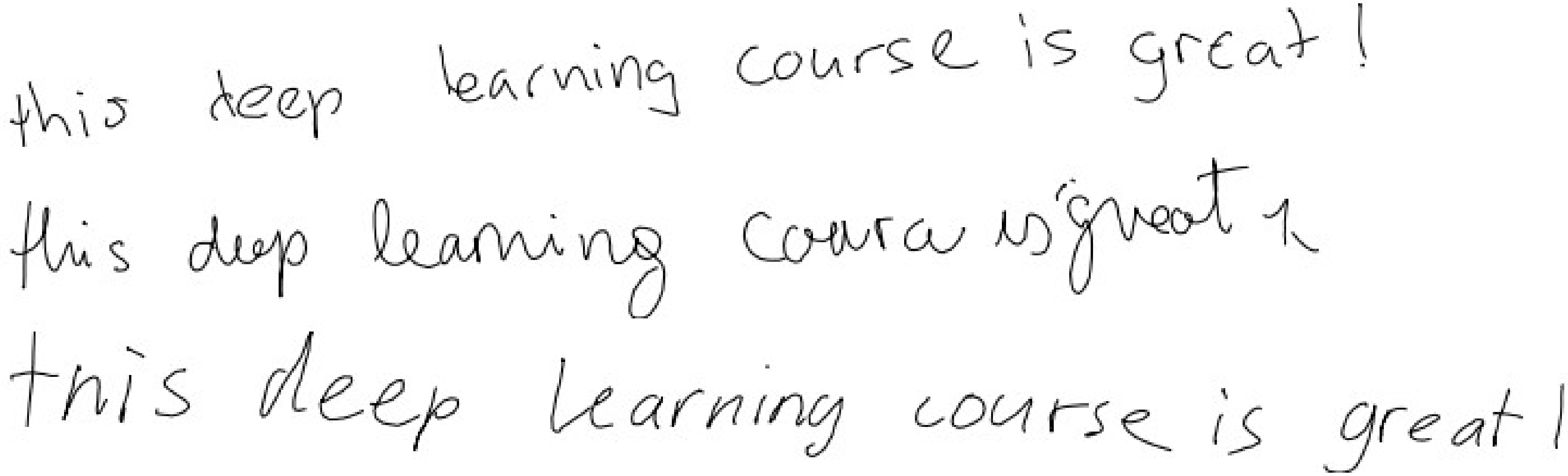
Learning text windows

$$\phi(t,u) = \sum_{k=1}^{K} \alpha_t^k \exp\left(-\beta_t^k \left(\kappa_t^k - u\right)^2\right)$$

$$w_t = \sum_{u=1}^{U} \phi(t,u) c_u$$



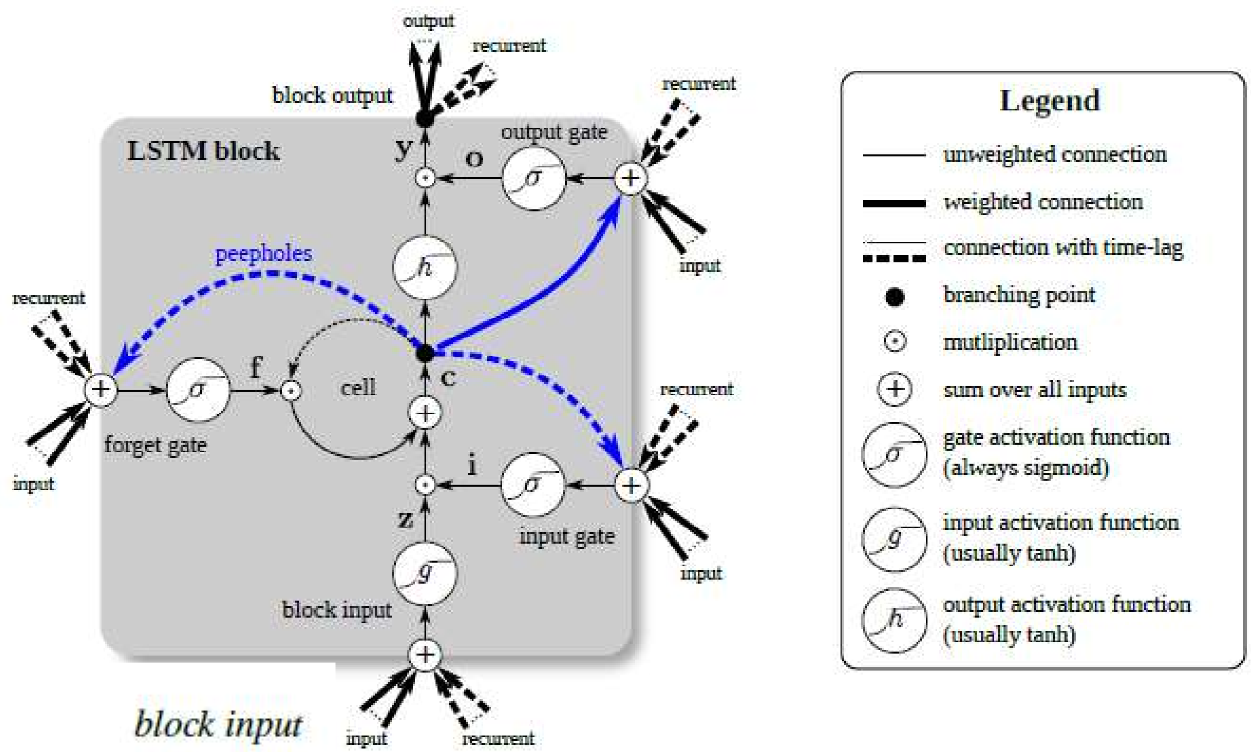
A demonstration of online handwriting recognition by an RNN with Long Short Term Memory (from Alex Graves)



http://www.cs.toronto.edu/~graves/handwriting.html

LSTM Architecture Explorations

- "Official version" with a lot of peepholes



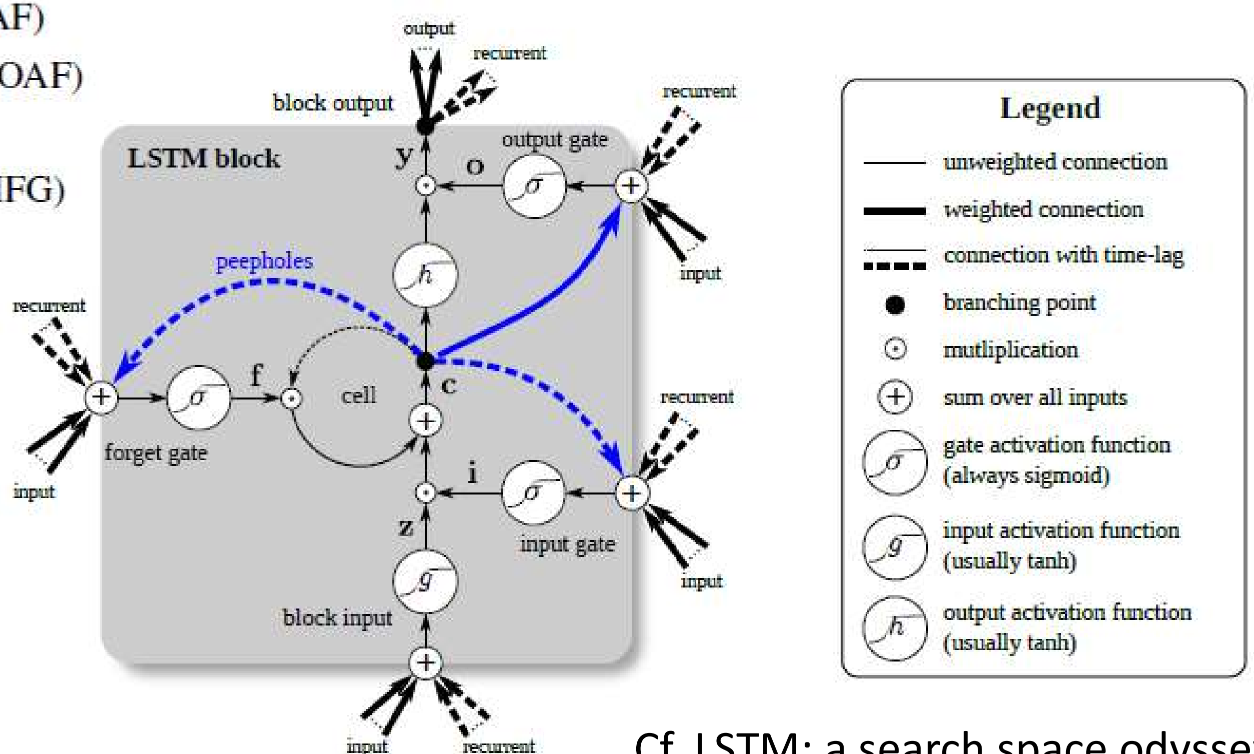
$$
\begin{aligned}
\mathbf{z}^t &= g(\mathbf{W}_z\mathbf{x}^t + \mathbf{R}_z\mathbf{y}^{t-1} + \mathbf{b}_z) && \textit{block input}\\
\mathbf{i}^t &= \sigma(\mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i) && \textit{input gate}\\
\mathbf{f}^t &= \sigma(\mathbf{W}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{y}^{t-1} + \mathbf{p}_f \odot \mathbf{c}^{t-1} + \mathbf{b}_f) && \textit{forget gate}\\
\mathbf{c}^t &= \mathbf{i}^t \odot \mathbf{z}^t + \mathbf{f}^t \odot \mathbf{c}^{t-1} && \textit{cell state}\\
\mathbf{o}^t &= \sigma(\mathbf{W}_o\mathbf{x}^t + \mathbf{R}_o\mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^{t} + \mathbf{b}_o) && \textit{output gate}\\
\mathbf{y}^t &= \mathbf{o}^t \odot h(\mathbf{c}^t) && \textit{block output}
\end{aligned}
$$

Cf. LSTM: a search space odyssey

A search space odyssey

- What if we remove some parts of this?

1. No Input Gate (NIG)
2. No Forget Gate (NFG)
3. No Output Gate (NOG)
4. No Input Activation Function (NIAF)
5. No Output Activation Function (NOAF)
6. No Peepholes (NP)
7. Coupled Input and Forget Gate (CIFG)
8. Full Gate Recurrence (FGR)

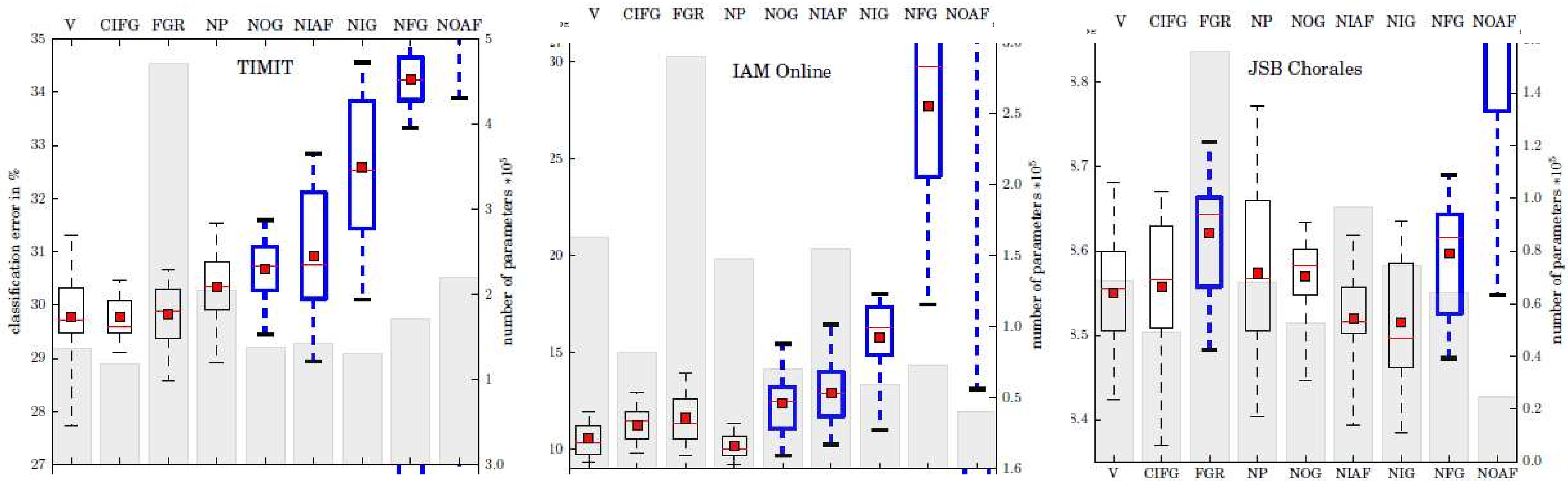


Cf. LSTM: a search space odyssey

Datasets

- TIMIT
  - Speech data
  - Framewise classification
  - 3696 sequences, 304 frames per sequence
- IAM
  - Handwriting stroke data
  - Map handwriting strokes to characters
  - 5535 sequences, 334 frames per sequence
- JSB
  - Music Modeling
  - Predict next note
  - 229 sequences, 61 frames per sequence

Results



1. No Input Gate (NIG)
2. No Forget Gate (NFG)
3. No Output Gate (NOG)
4. No Input Activation Function (NIAF)
5. No Output Activation Function (NOAF)
6. No Peepholes (NP)
7. Coupled Input and Forget Gate (CIFG)
8. Full Gate Recurrence (FGR)

Cf. LSTM: a search space odyssey

Impact of Parameters

- Analysis method: fANOVA (Hutter et al. 2011, 2014)
- (Random) Decision forests trained on the parameter space to partition the parameter space and find the best parameter
- Given trained (random) decision forest, can go to each leave node and count the impact of missing one predictor

Impact of Parameters

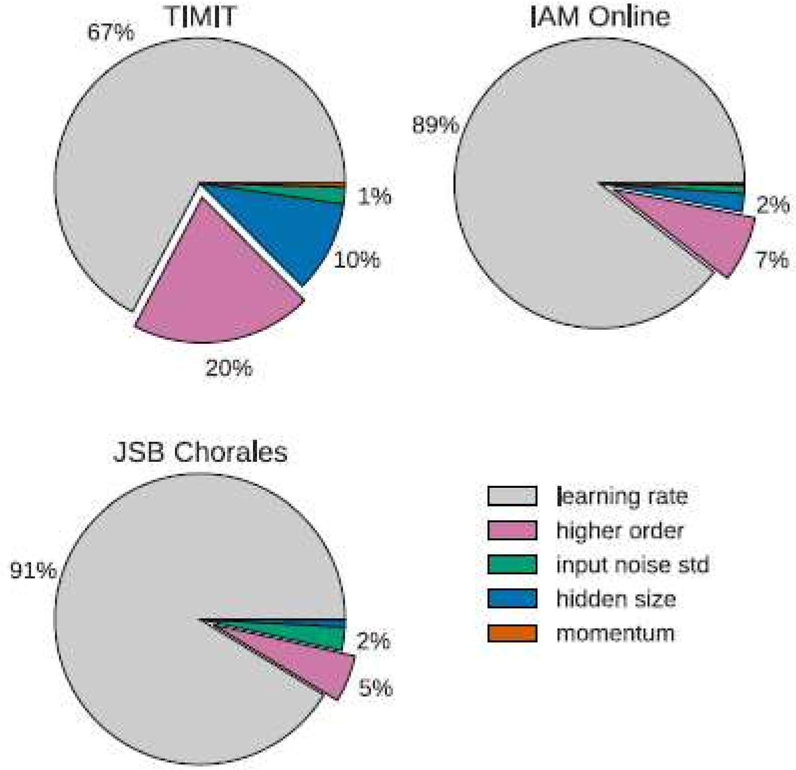


Figure 3. Pie charts showing which fraction of variance of the test set performance can be attributed to each of the hyperparameters. The percentage of variance that is due to interactions between multiple parameters is indicated as "higher order."

Cf. LSTM: a search space odyssey

Impact of Parameters

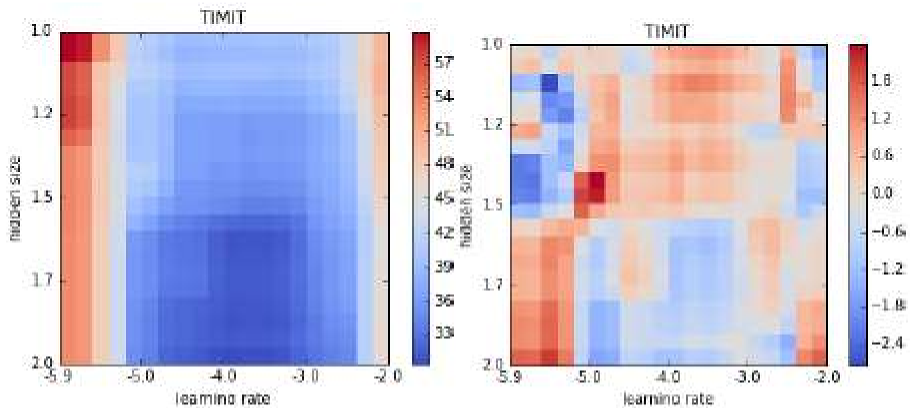


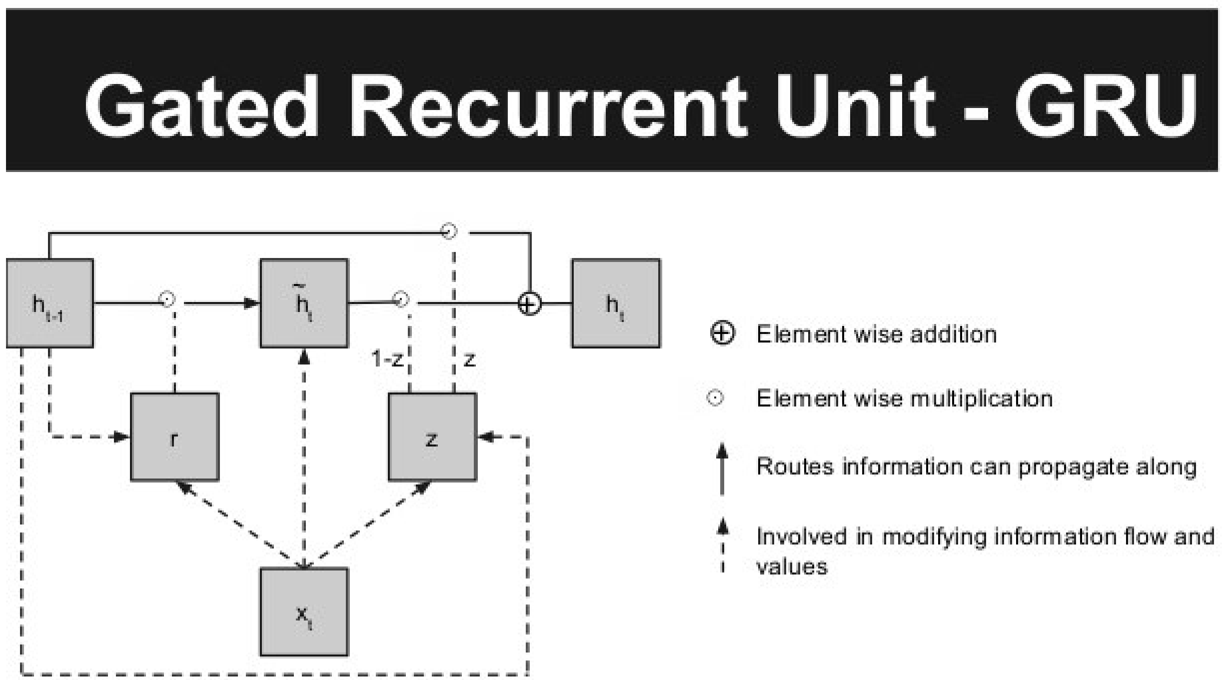
Figure 5. Left: The predicted marginal error for combinations of learning rate and hidden size. Right: The component that is solely due to the interaction of the two and cannot be attributet to changes in one of them alone. In other words the difference to the case of them being perfectly independent. (Blue is better than red.)

$$\begin{aligned} \text{learning rate} \times \text{hidden size} &= 6.7\% \\ \text{learning rate} \times \text{input noise} &= 4.4\% \\ \text{hidden size} \times \text{input noise} &= 2.0\% \\ \text{learning rate} \times \text{momentum} &= 1.5\% \\ \text{momentum} \times \text{hidden size} &= 0.6\% \\ \text{momentum} \times \text{input noise} &= 0.4\% \end{aligned}$$

Cf. LSTM: a search space odyssey

GRU: Gated Recurrence Unit

- Much simpler than LSTM
  - No output gate
  - Coupled input and forget gate



Cf. slideshare.net

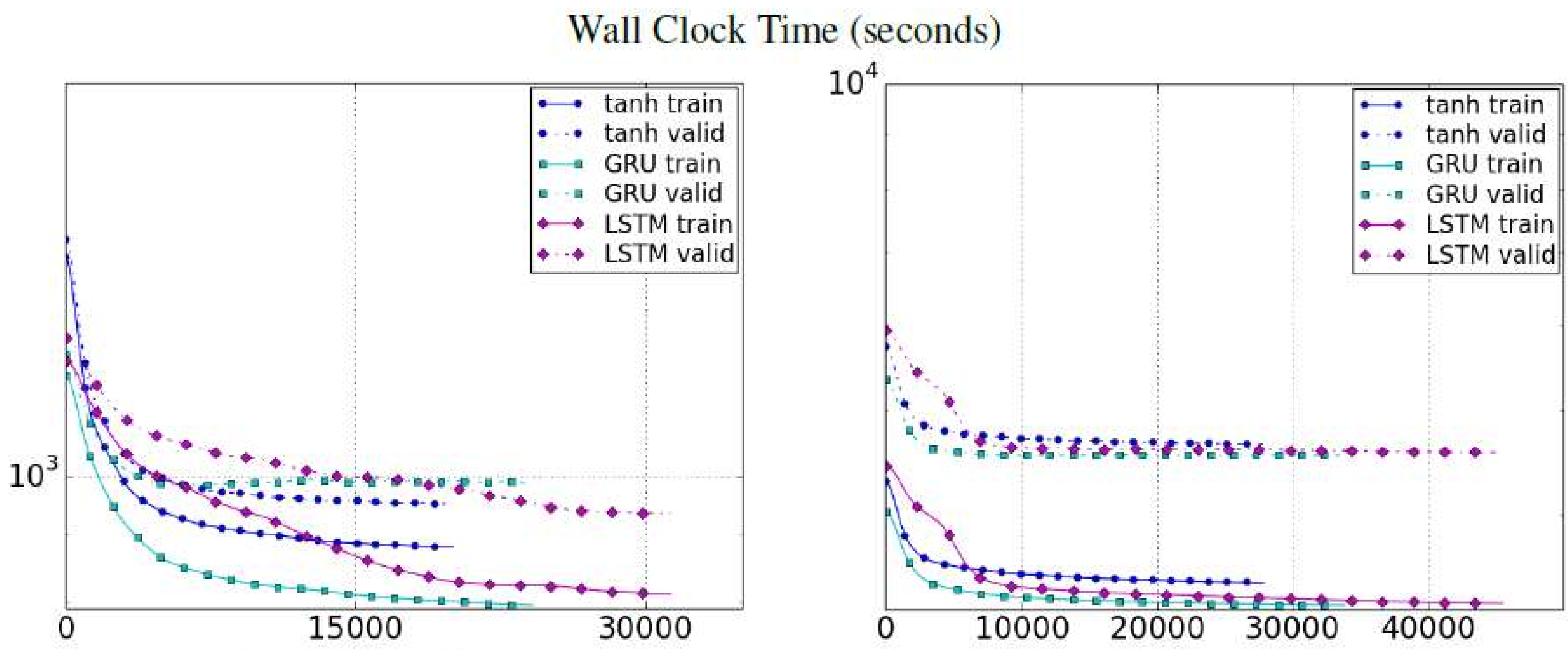
Data

- Music Datasets:
  - Nottingham, 1200 sequences
  - MuseData, 881 sequences
  - JSB, 382 sequences
- Ubisoft Data A
  - Speech, 7230 sequences, length 500
- Ubisoft Data B
  - Speech, 800 sequences, length 8000

Results

Nottingham
Music, 1200 sequences

MuseData
Music, 881 sequences



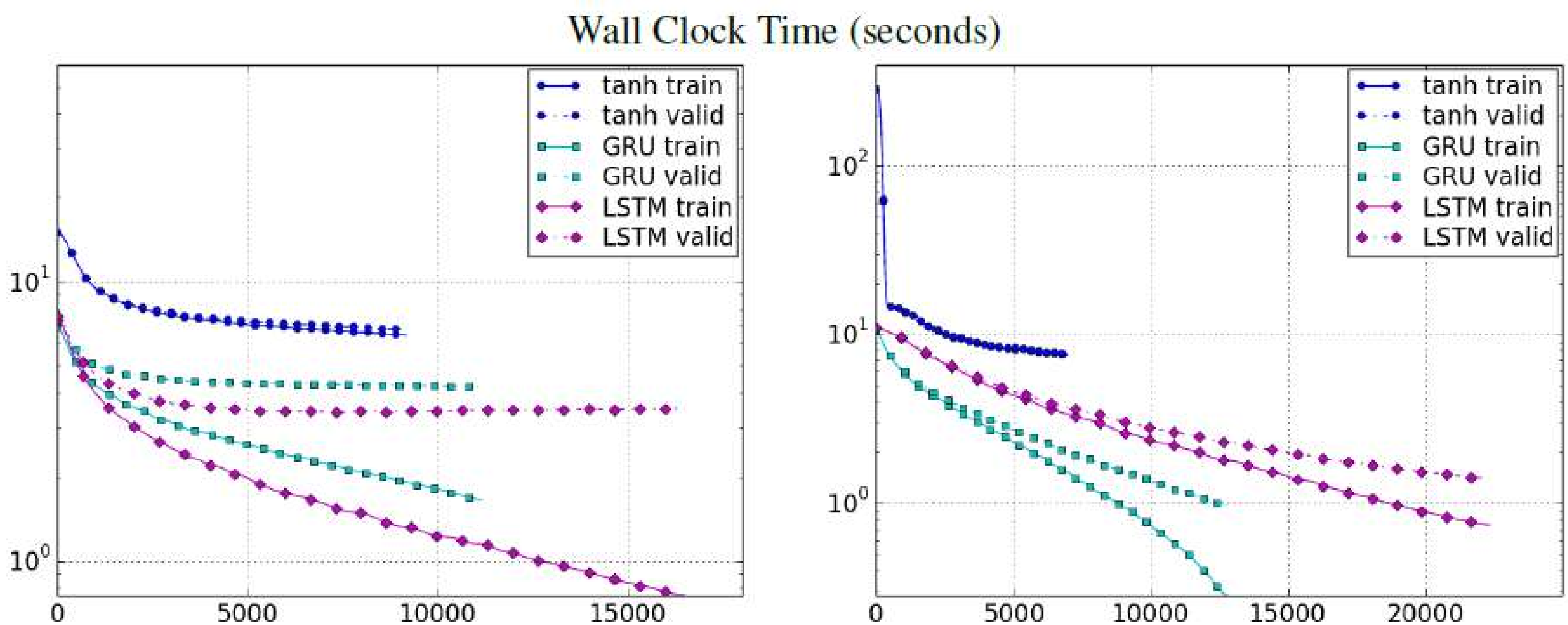
(a) Nottingham Dataset (b) MuseData Dataset

Cf. Empirical Evaluation of Gated Recurrent Neural Network Modeling

Results

Ubisoft Data A
Speech, 7230 sequences, length 500

Ubisoft Data B
Speech, 800 sequences, length 8000

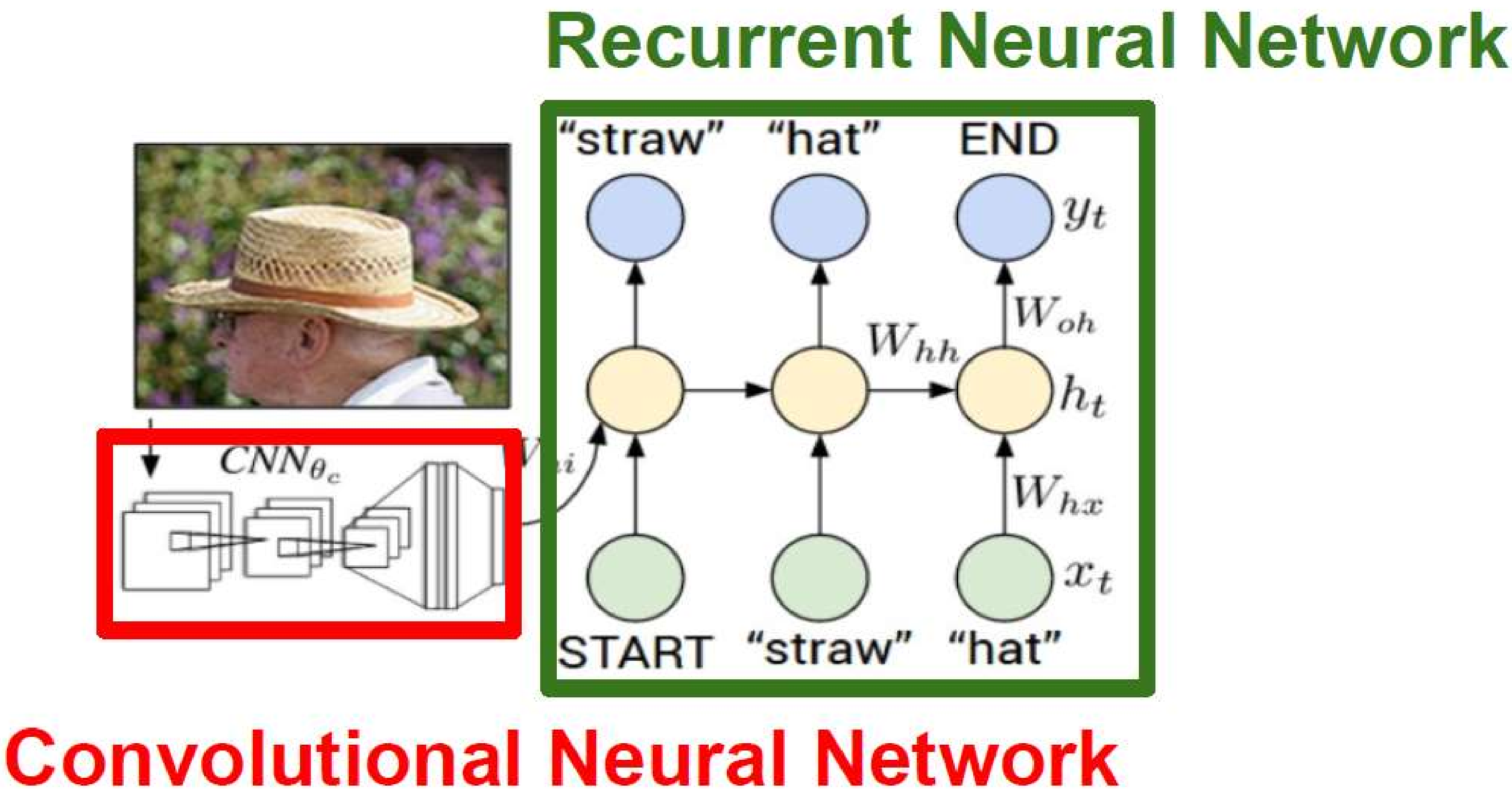


(a) Ubisoft Dataset A

(b) Ubisoft Dataset B

Cf. Empirical Eva

CNN+RNN Example



“straw hat”

training example

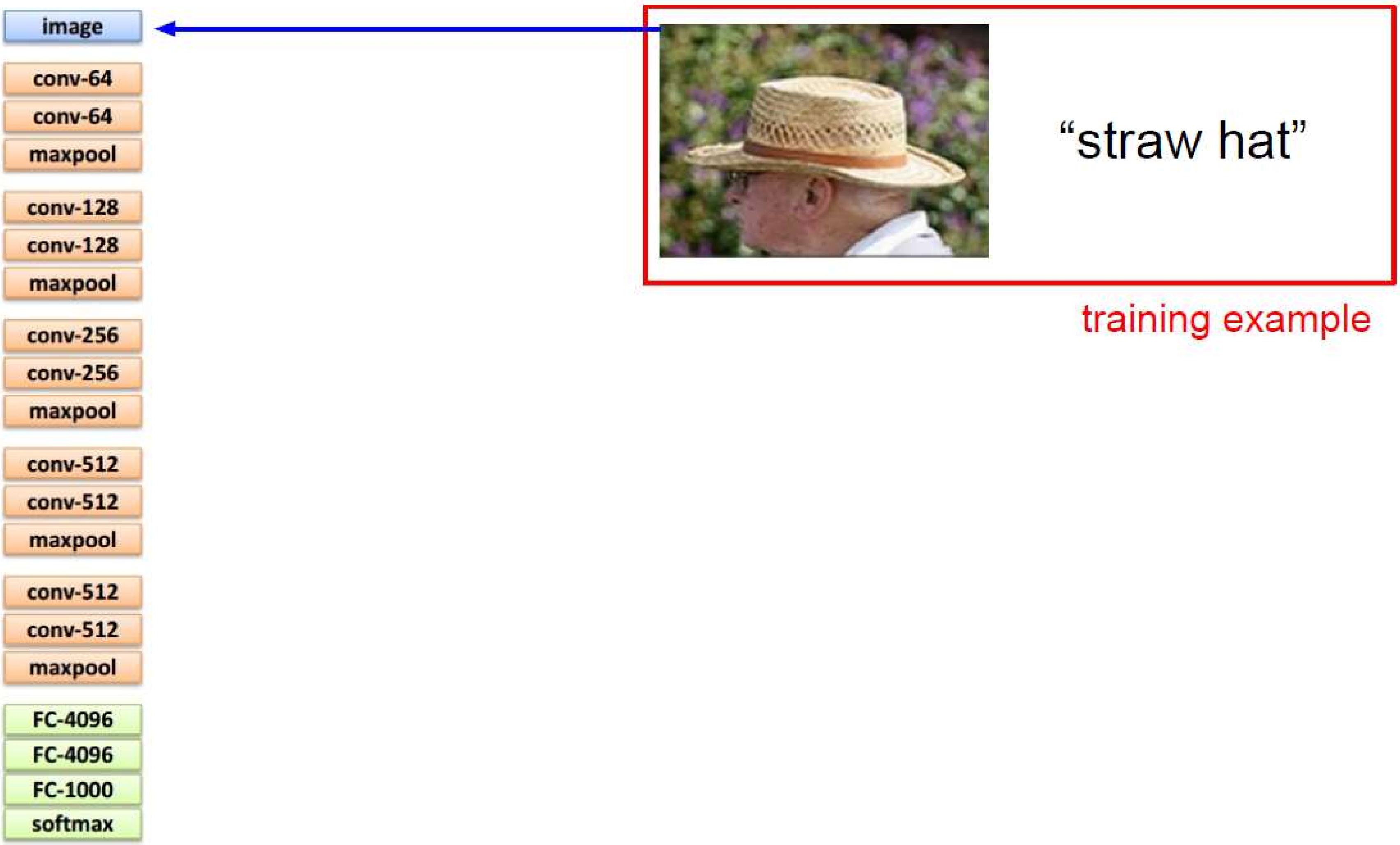


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
“straw hat”
training example

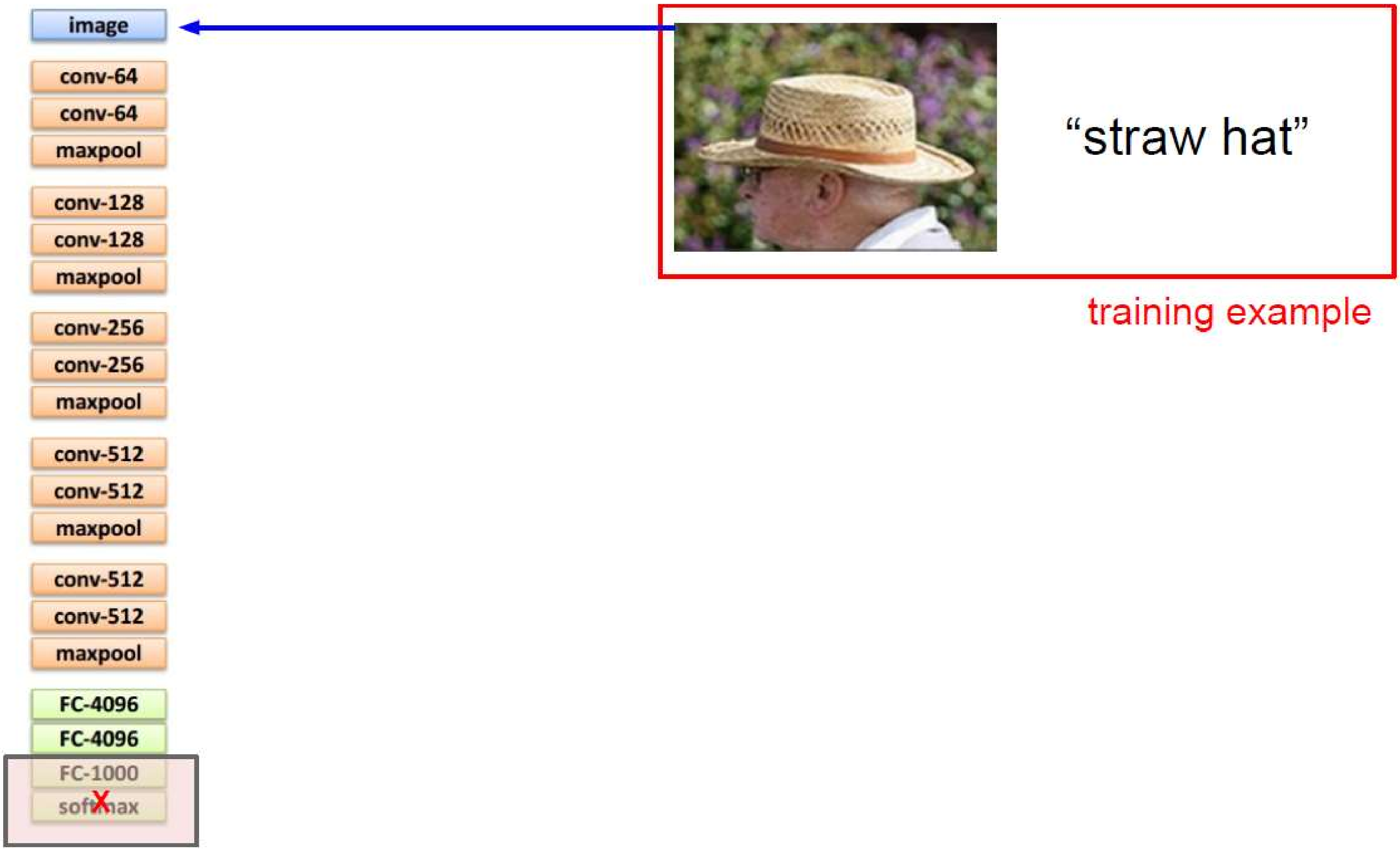


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
"straw hat"
training example

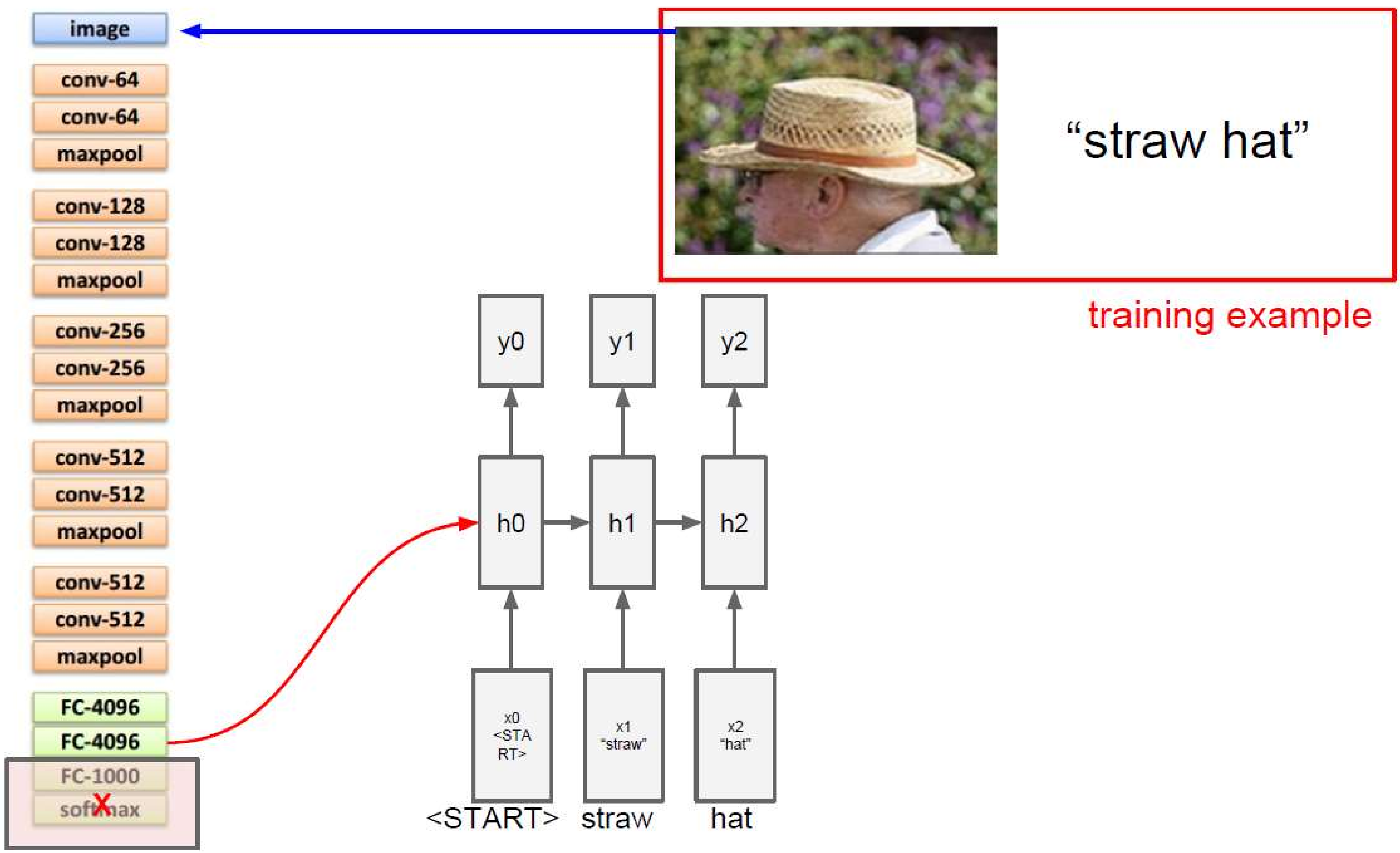
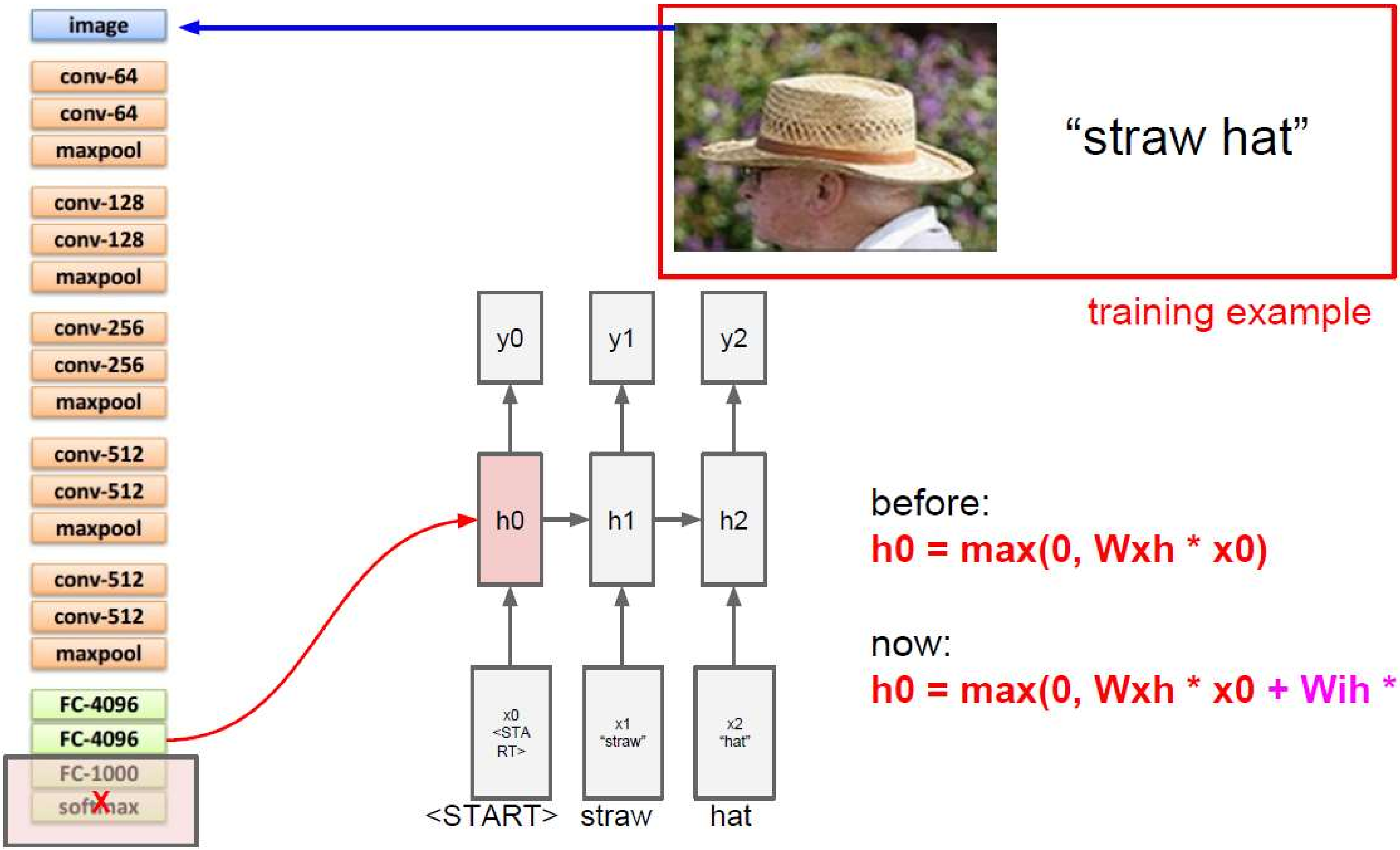


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
X
"straw hat"
training example
y0
y1
y2
h0
h1
h2
x0 <START>
x1 "straw"
x2 "hat"
<START>
straw
hat



before:

h0 = max(0, Wxh * x0)

now:

h0 = max(0, Wxh * x0 + Wih * v)

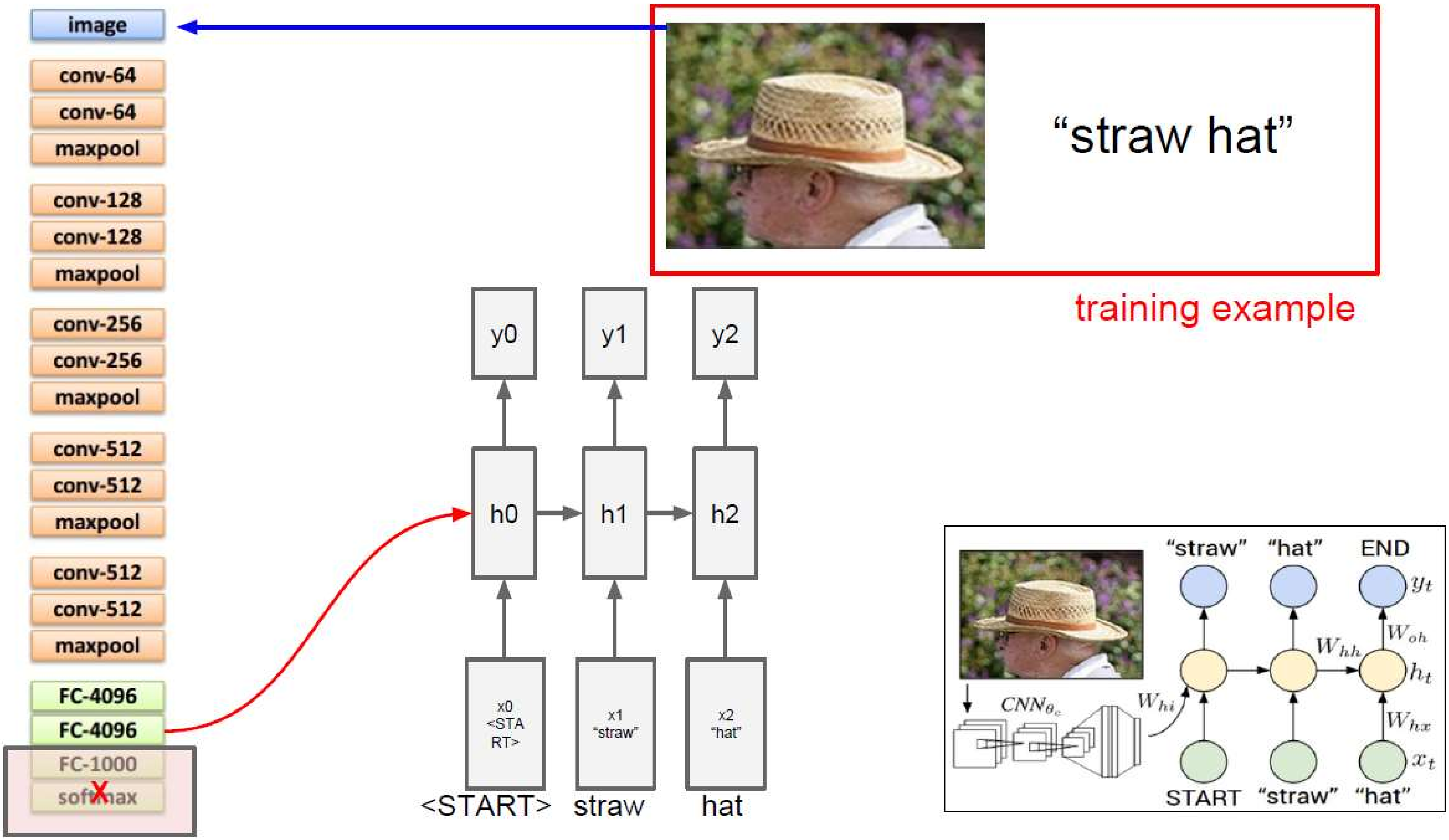


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
“straw hat”
training example
y0
y1
y2
h0
h1
h2
x0 <START>
x1 “straw”
x2 “hat”
<START>
straw
hat
“straw”
“hat”
END
y_t
W_{oh}
W_{hh}
h_t
W_{hi}
CNN_{θ_c}
W_{hx}
x_t
START
“straw”
“hat”

test image

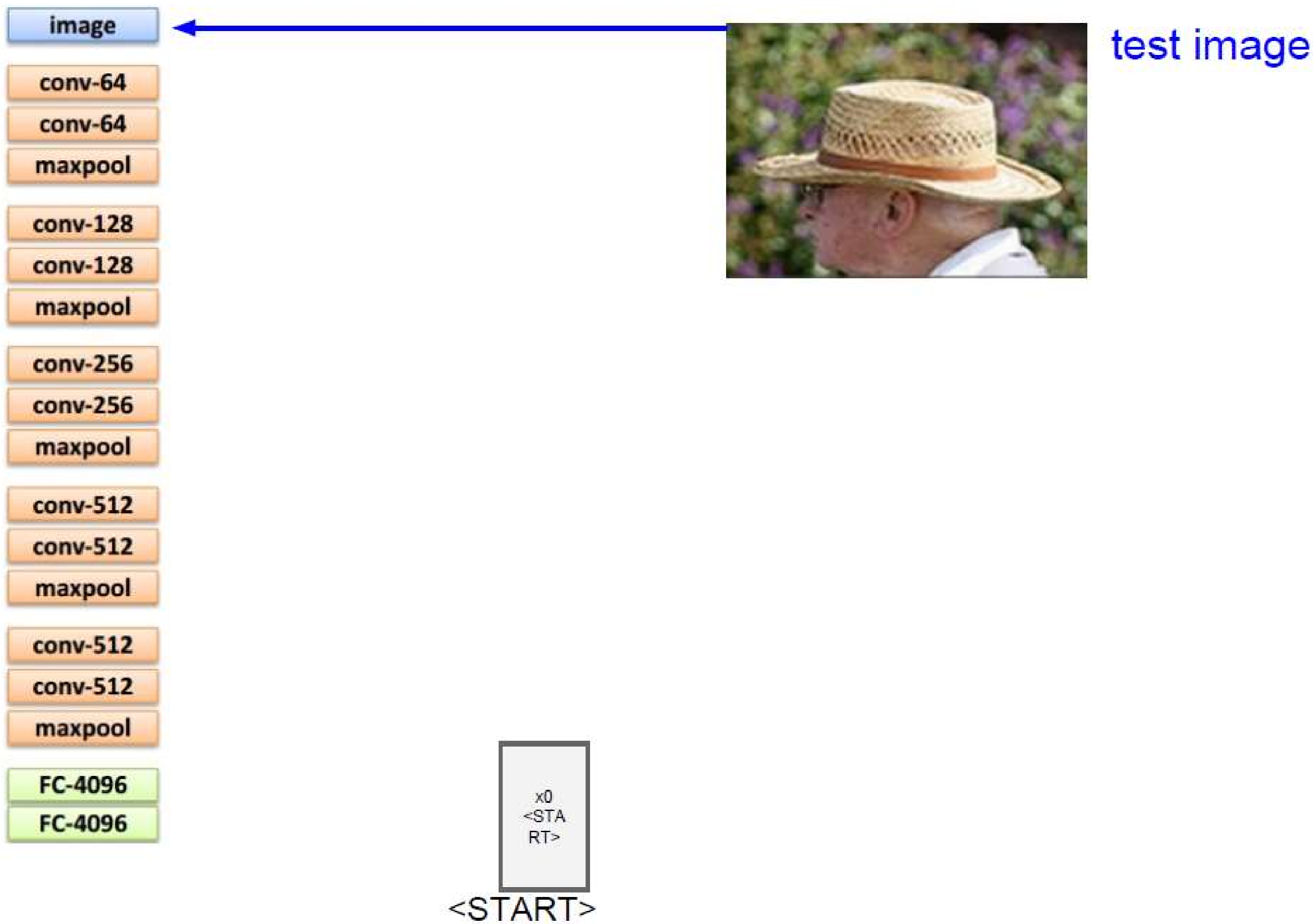


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
x0
<STA
RT>
<START>

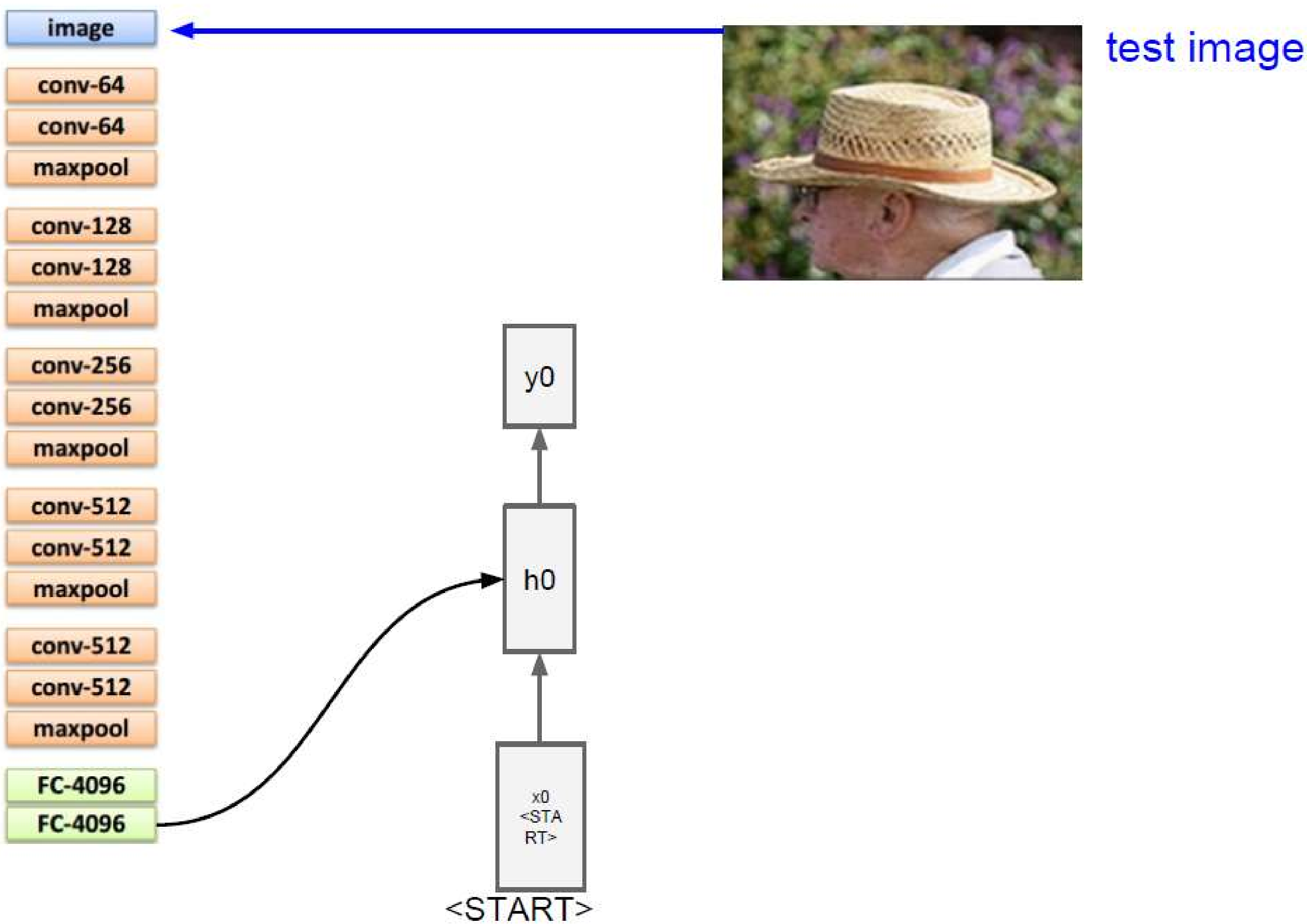


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
h0
x0
<STA
RT>
<START>

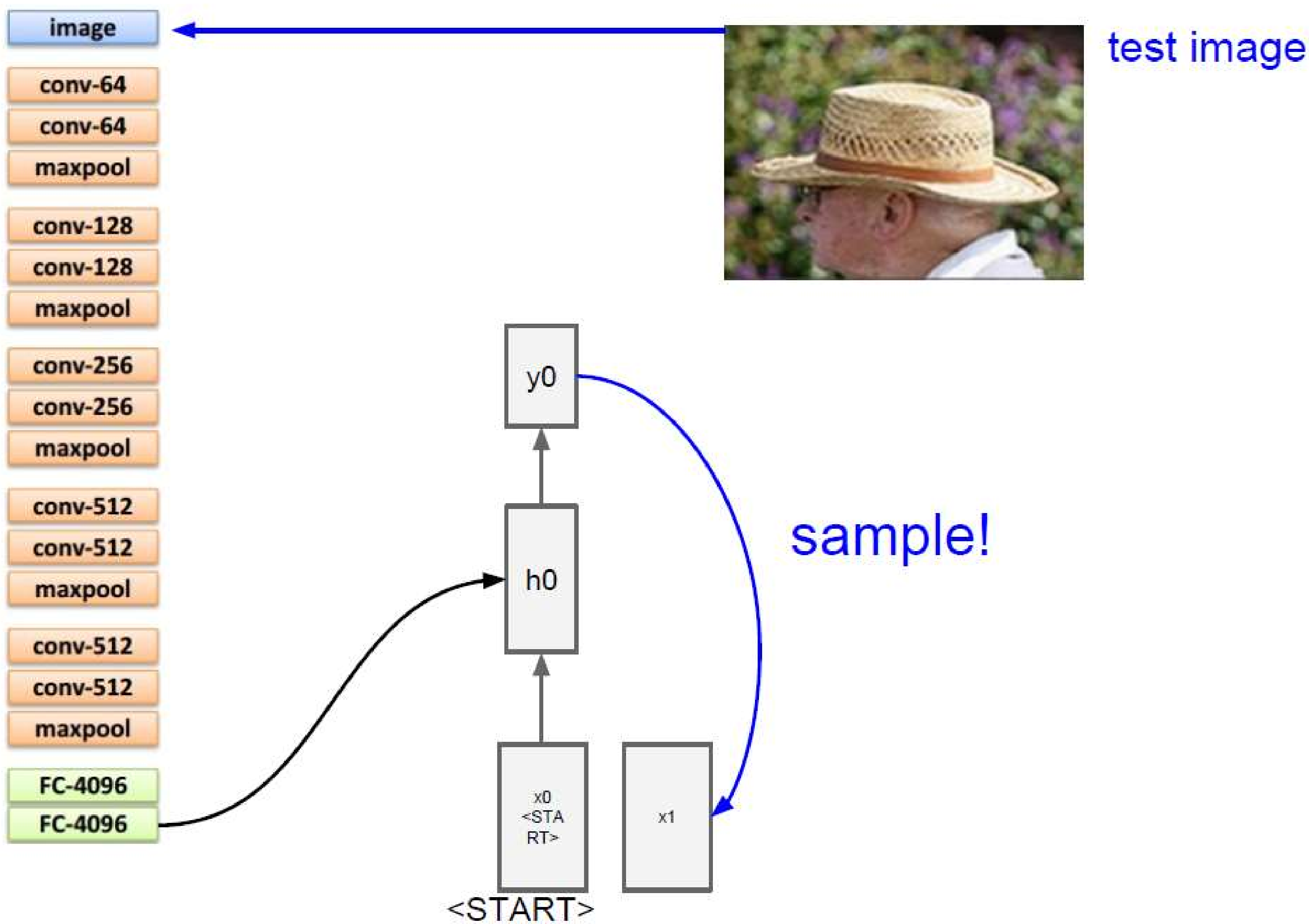


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
h0
sample!
x0 <START>
x1
<START>

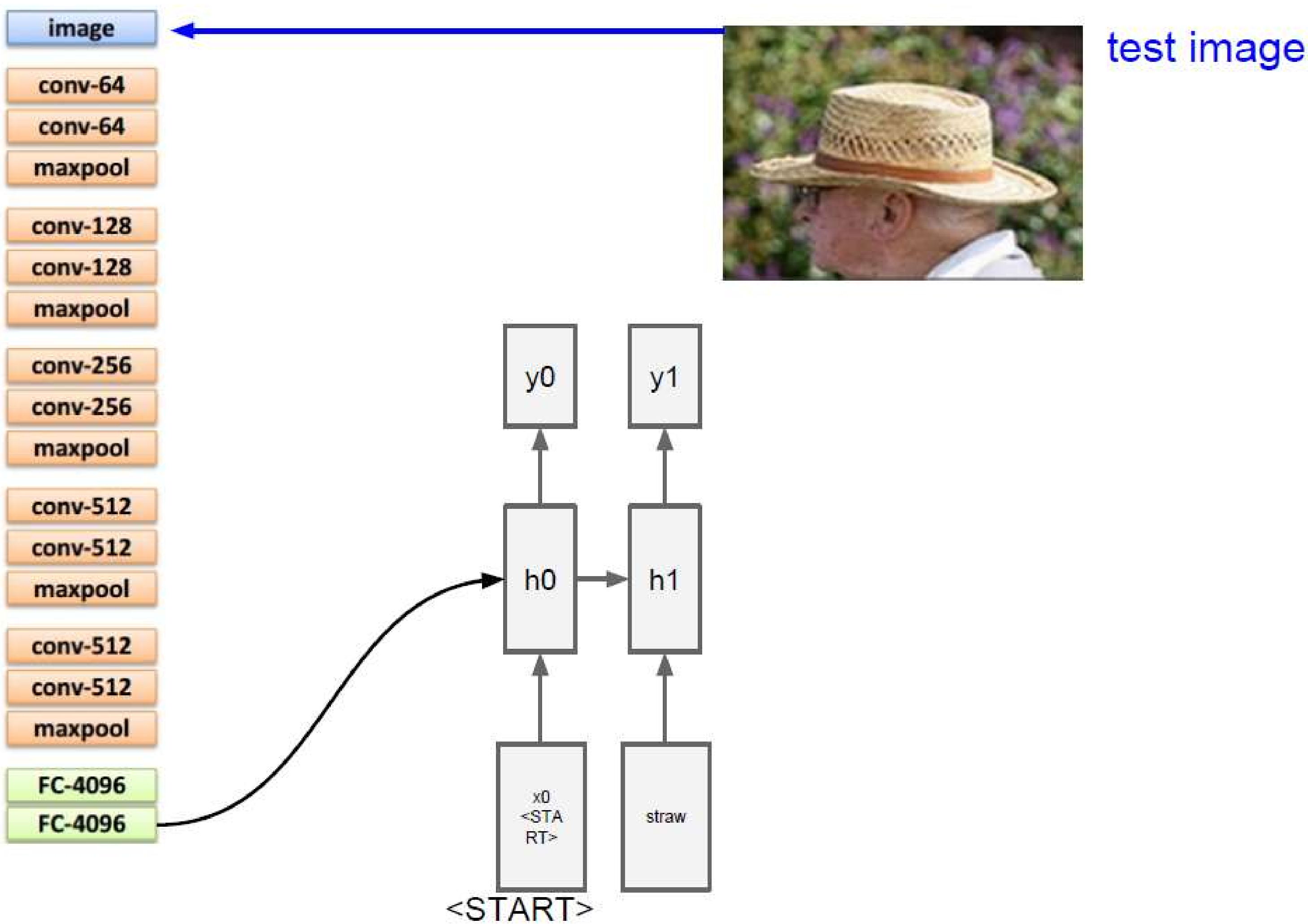


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
h0
h1
x0 <START>
straw
<START>

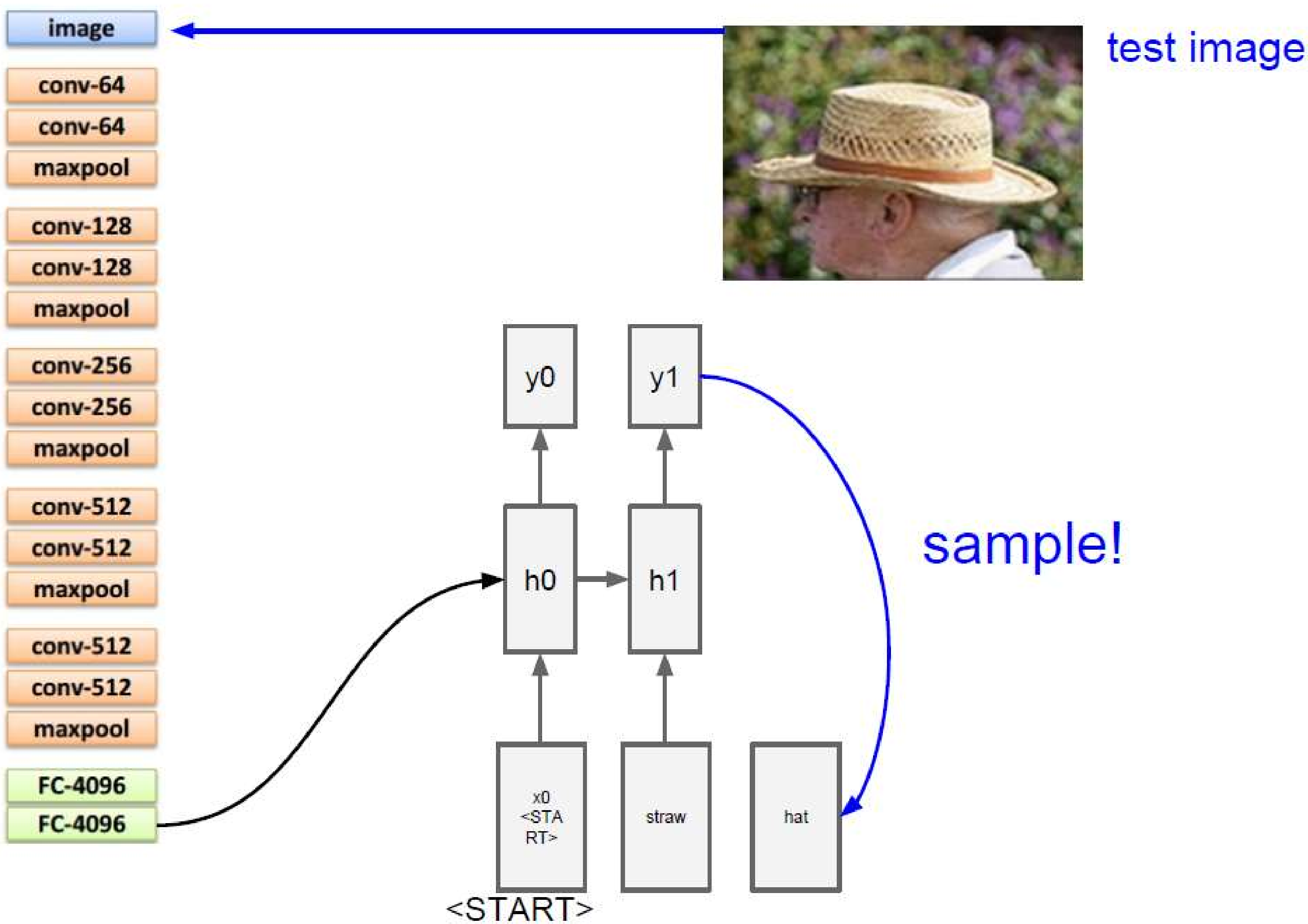


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
sample!
h0
h1
x0 <START>
straw
hat
<START>

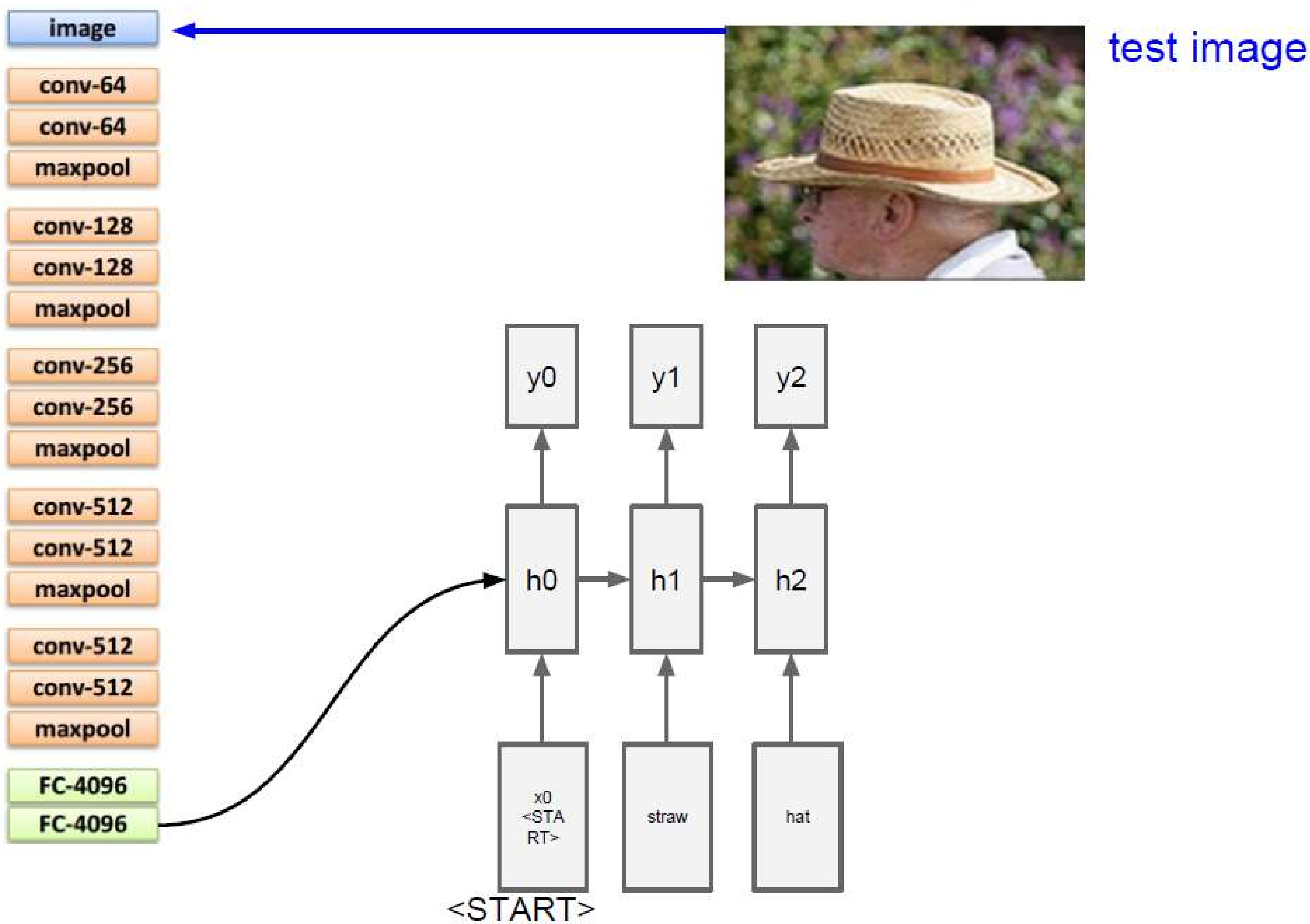


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
y2
h0
h1
h2
x0 <START>
straw
hat
<START>

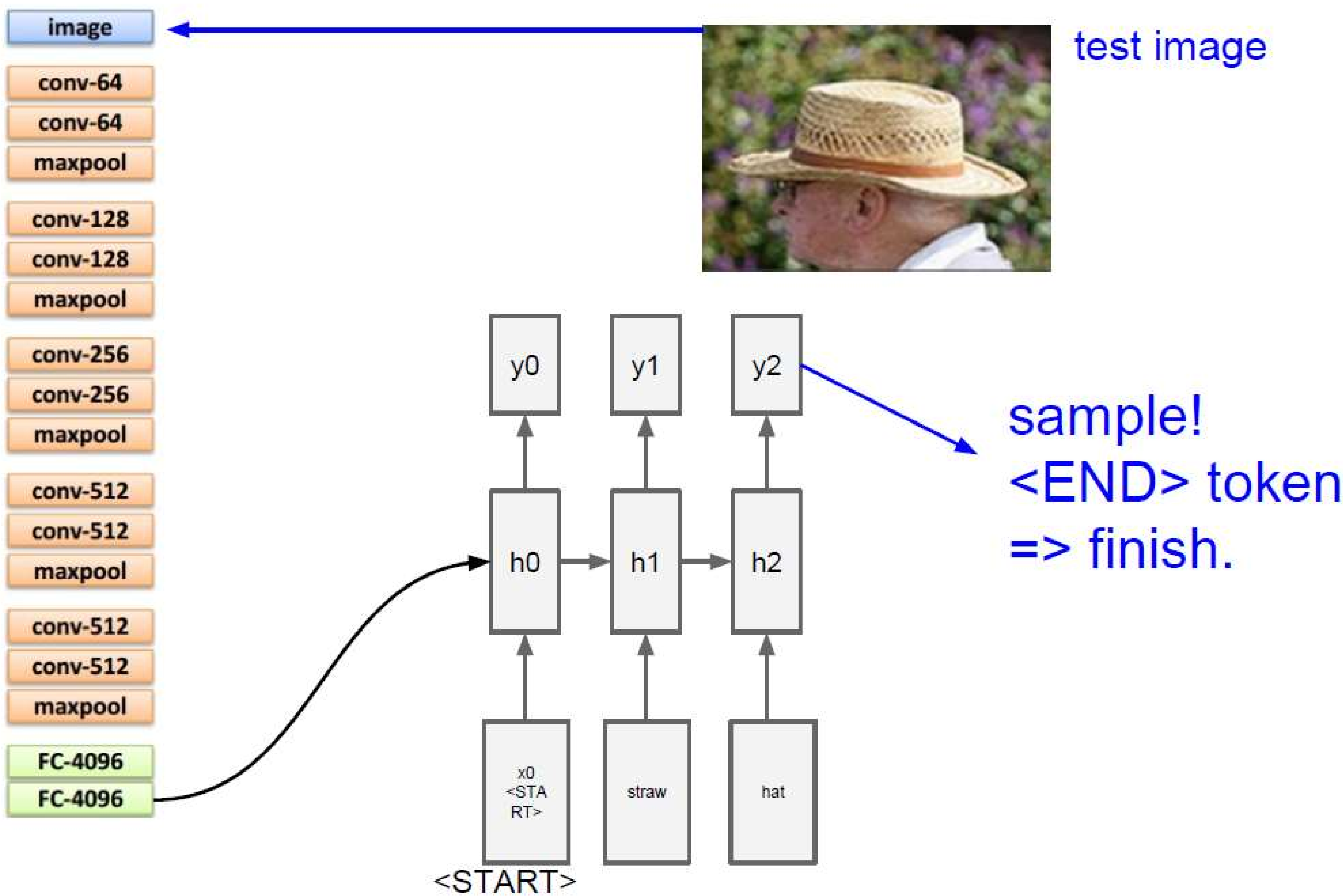
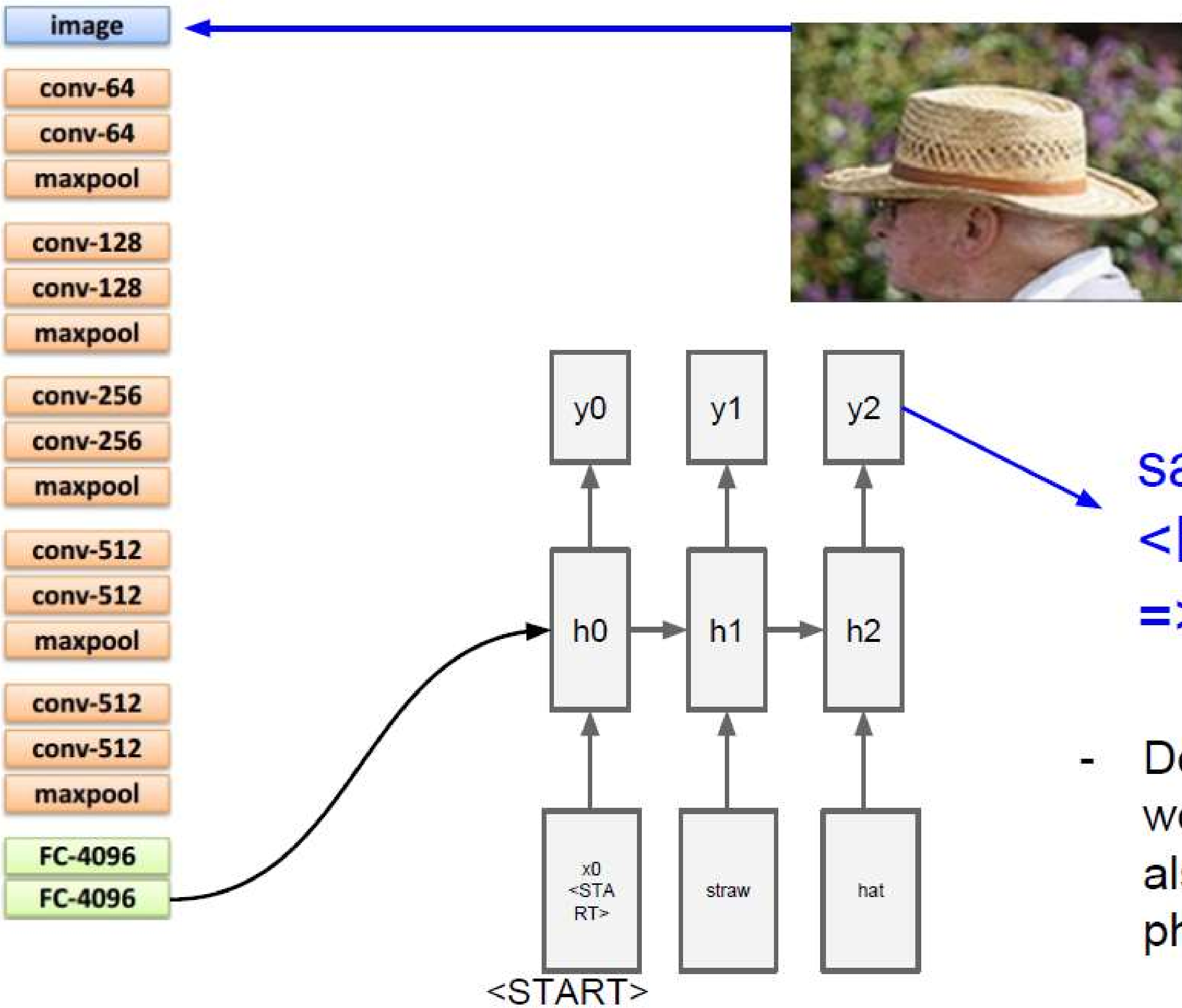


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
y2
sample!
<END> token
=> finish.
h0
h1
h2
x0
<STA
RT>
straw
hat
<START>

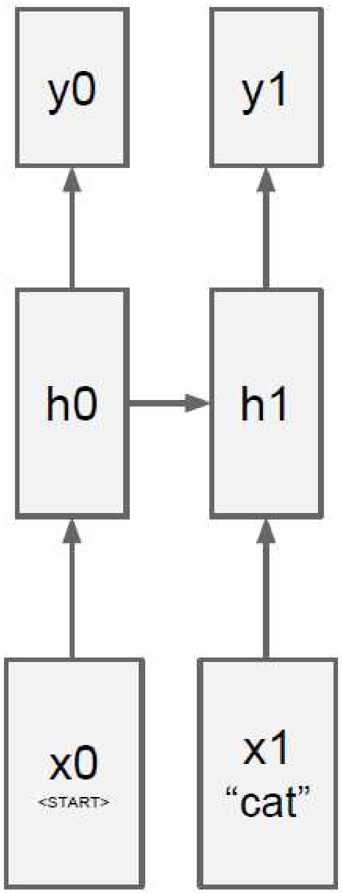


test image

sample!
<END> token
=> finish.

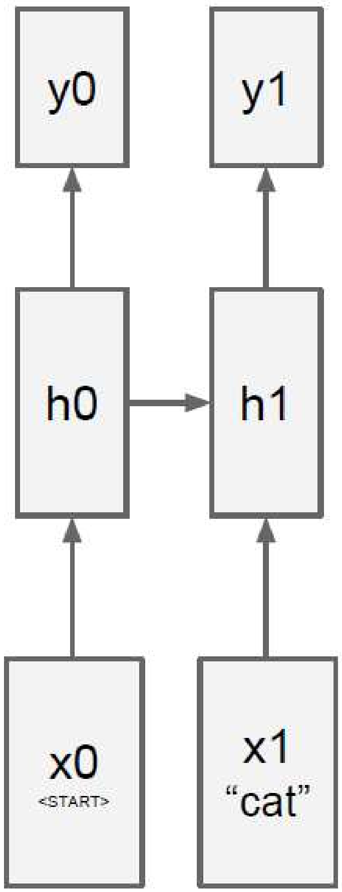
- Don't have to do greedy word-by-word sampling, can also search over longer phrases with **beam search**

RNN vs. LSTM



“hidden” representation
(e.g. 200 numbers)
h1 = max(0, Wxh * x1 + Whh * h0)

RNN vs. LSTM

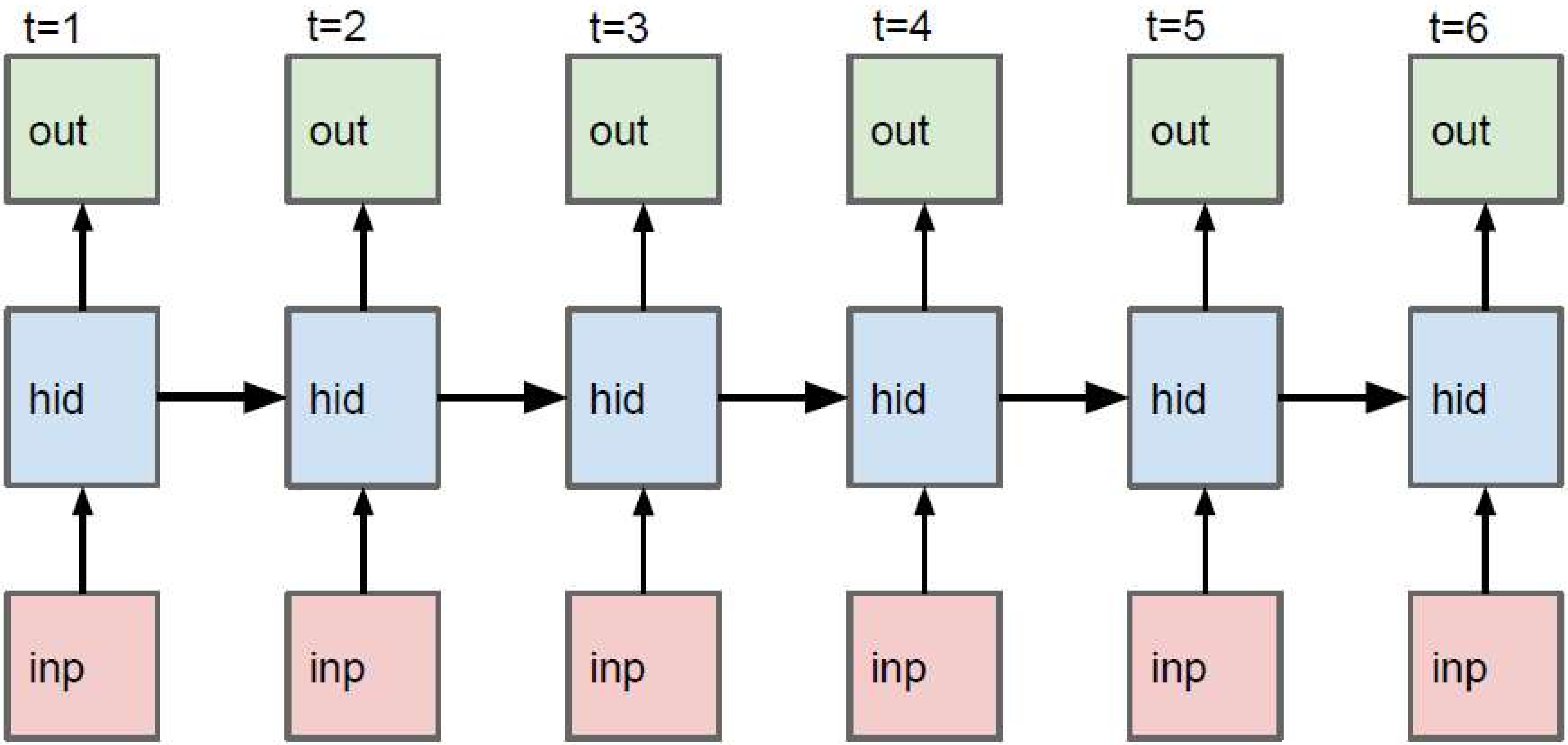


“hidden” representation
(e.g. 200 numbers)
h1 = max(0, Wxh * x1 + Whh * h0)

LSTM changes the form of the equation for **h1** such that**:**

1. more expressive multiplicative interactions
2. gradients flow nicer
3. network can explicitly decide to reset the hidden state

RNN



t=1
t=2
t=3
t=4
t=5
t=6
out
out
out
out
out
out
hid
hid
hid
hid
hid
hid
inp
inp
inp
inp
inp
inp

LSTM

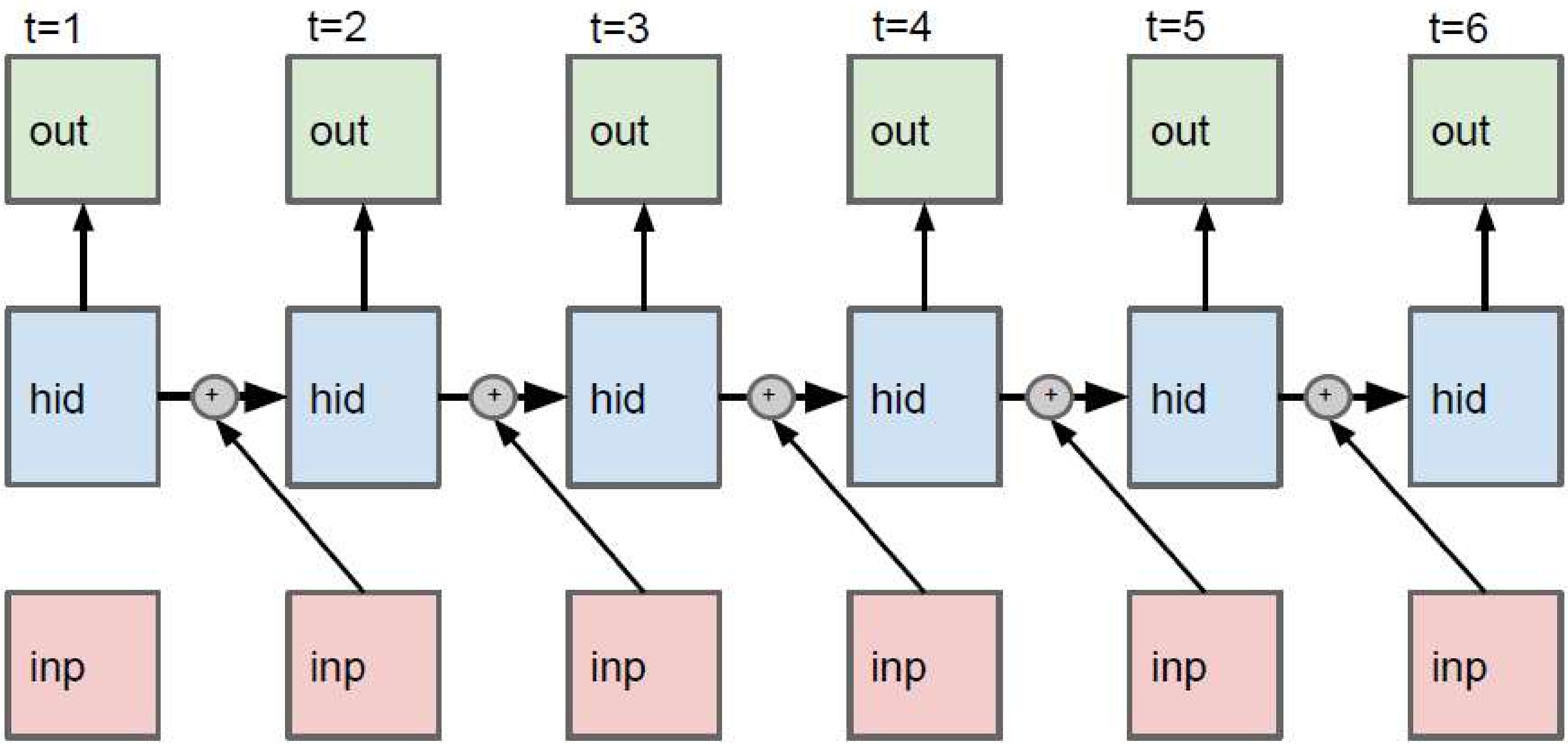

t=1
t=2
t=3
t=4
t=5
t=6
out
out
out
out
out
out
hid
hid
hid
hid
hid
hid
+
+
+
+
+
inp
inp
inp
inp
inp
inp

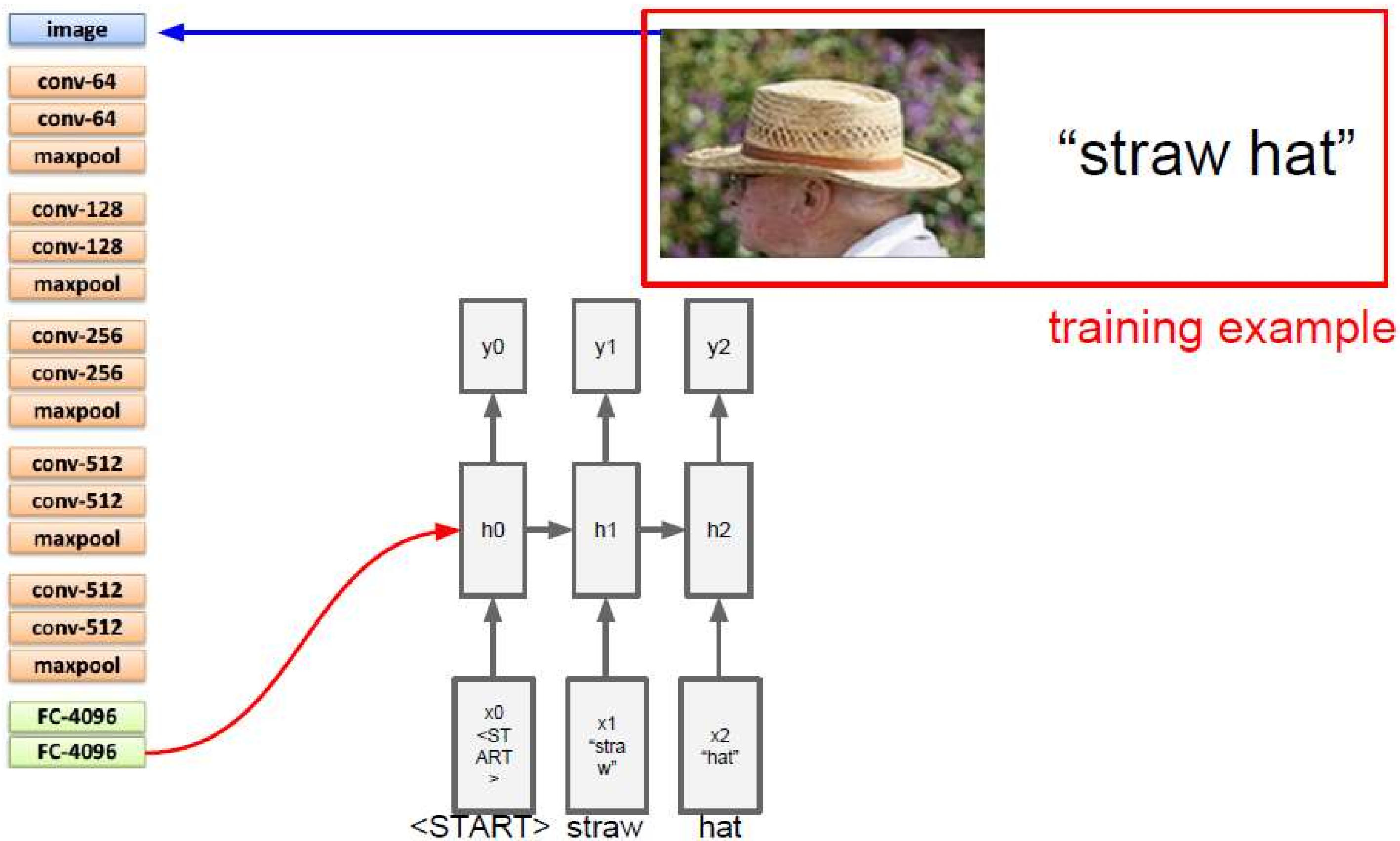
Image Sentence Datasets

a man riding a bike on a dirt path through a forest.
bicyclist raises his fist as he rides on desert dirt trail.
this dirt bike rider is smiling and raising his fist in triumph.
a man riding a bicycle while pumping his fist in the air.
a mountain biker pumps his fist in celebration.

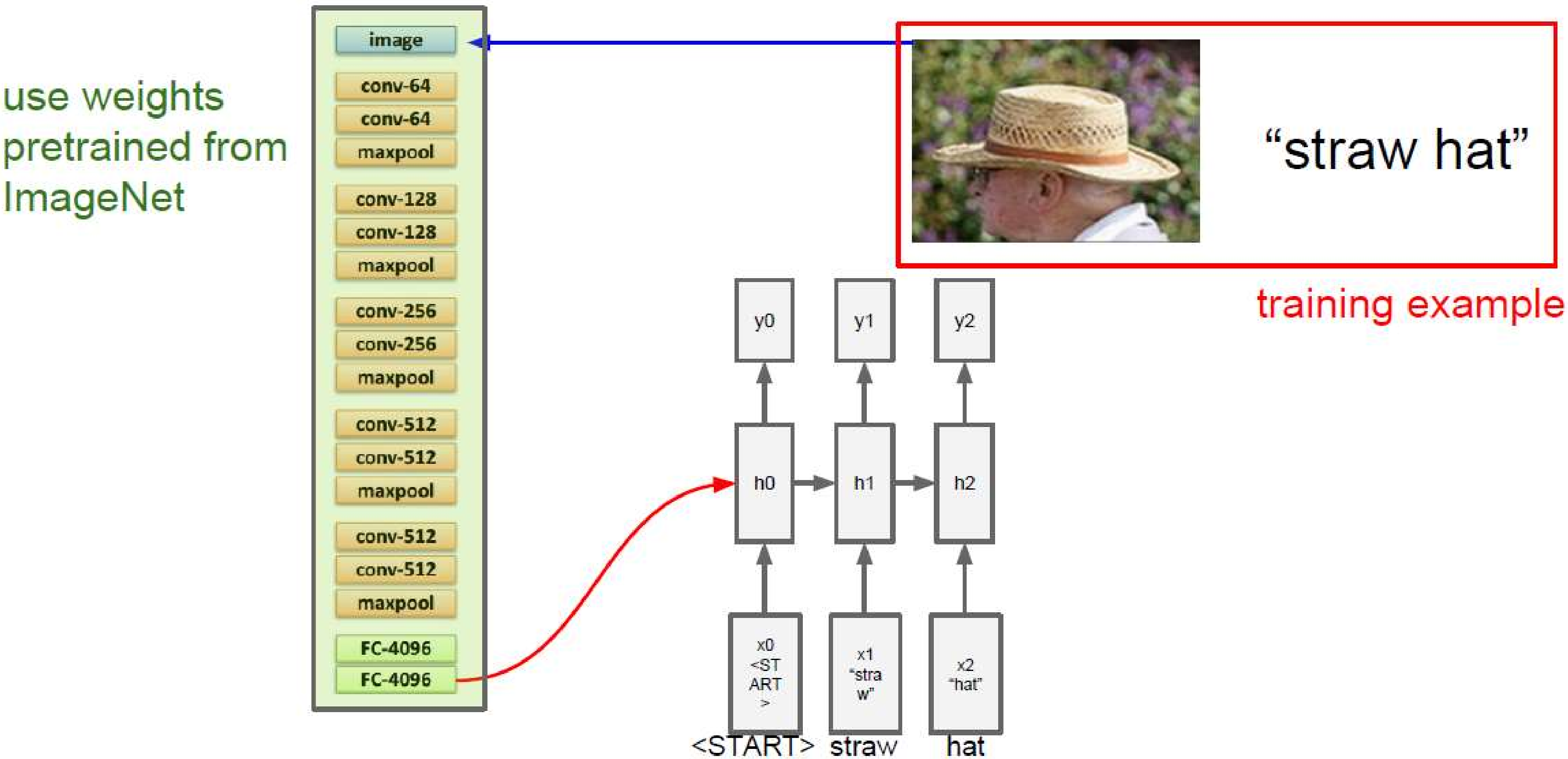
Microsoft COCO
[Tsung-Yi Lin et al. 2014]
mscoco.org

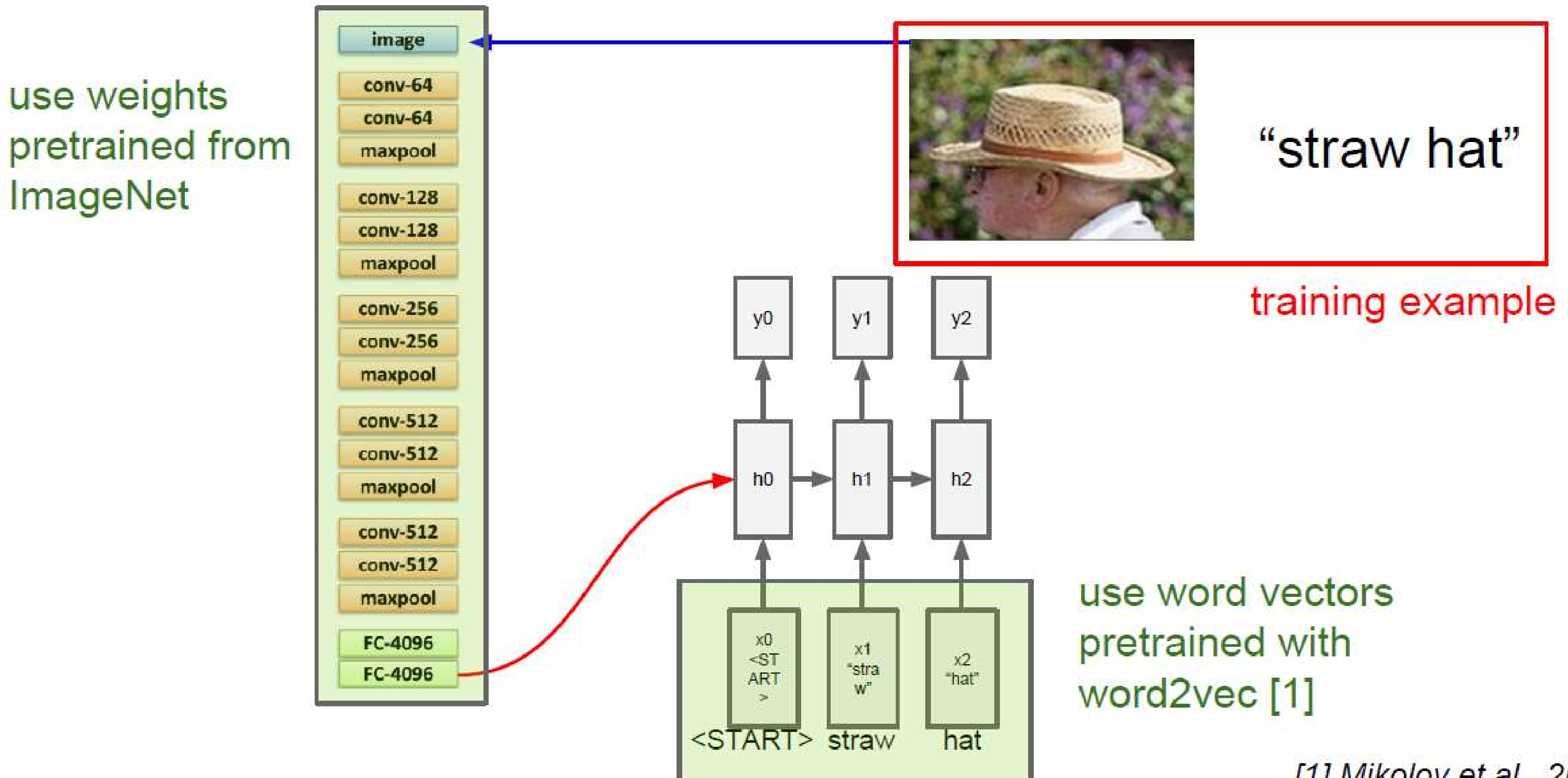
currently:
~120K images
~5 sentences each

+ Transfer Learning



Pre-training





[1] Mikolov et al., 2013

Summary of the approach

We wanted to describe images with sentences.

1. Define a single function from input -> output
2. Initialize parts of net from elsewhere if possible
3. Get some data
4. Train with SGD

Wow I can't believe that worked

a group of people standing around a room with remotes
`logprob: -9.17`

a young boy is holding a baseball bat
`logprob: -7.61`

a cow is standing in the middle of a street
`logprob: -8.84`

Wow I can't believe that worked

a cat is sitting on a toilet seat
`logprob: -7.79`

a display case filled with lots of different types of donuts
`logprob: -7.78`

a group of people sitting at a table with wine glasses
`logprob: -6.71`

Well, I can kind of see it

a man standing next to a clock on a wall
`logprob: -10.08`

a young boy is holding a baseball bat
`logprob: -7.65`

a cat is sitting on a couch with a remote control
`logprob: -12.45`

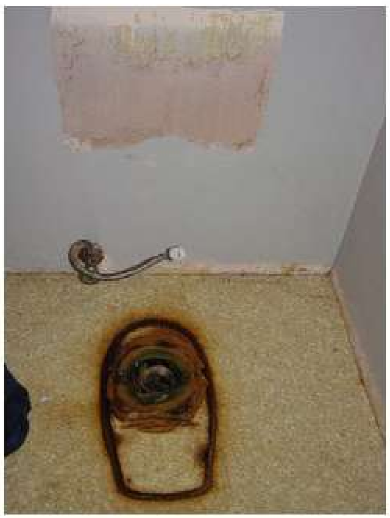
Well, I can kind of see it

a baby laying on a bed with a stuffed bear
`logprob: -8.66`

a table with a plate of food and a cup of coffee
`logprob: -9.93`

a young boy is playing frisbee in the park
`logprob: -9.52`

Not sure what happened there...



a toilet with a seat up in a bathroom
`logprob: -13.44`

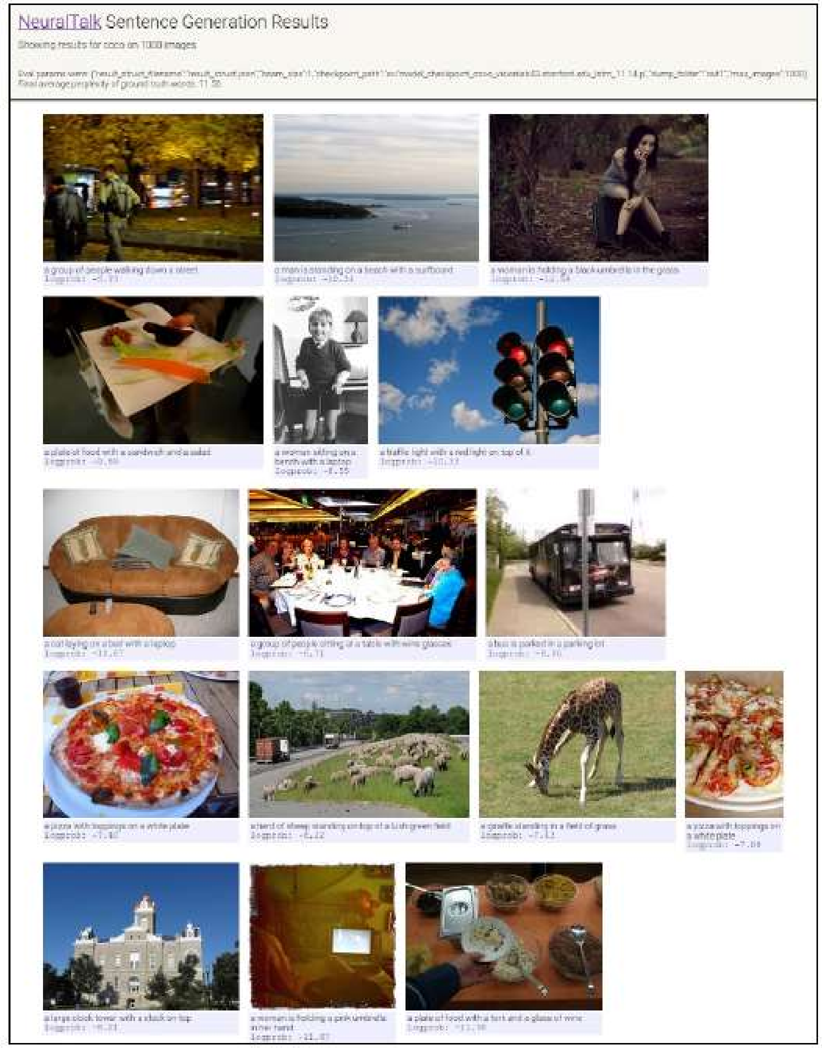
a woman holding a teddy bear in front of a mirror
`logprob: -9.65`

a horse is standing in the middle of a road
`logprob: -10.34`

See predictions on

1000 COCO images:

http://bit.ly/neuraltalkdemo



What this approach Doesn't do:

- There is no *reasoning*
- A single glance is taken at the image, no objects are detected, etc.
- We can't just describe any image