More on RNNs

Variants, Regularization, Tricks and More

(Oct 26, 2016)

YANG Jiancheng

Outline

- I. Variants
- II. Regularization
- III. Tricks
  - Keras dropout
  - Forget Gate Bias
  - Critical Components and Learning Rate Search
- IV. More
  - Note on Vanishing Gradients
  - Note on Memory Cells and Hidden States
  - Missing data
  - Seq2Seq
  - Clockwork RNN

• I. Variants

- **Main Conclusion**

a) Vanilla LSTM (most common, with peephole) works well on various datasets

b) LSTM without peephole and GRU simplify the network, **not hurting the performance**, or even making some improvements.

c) **GRU outperformed** the LSTM in most cases

d) **Unable to find an architecture** that consistently beat LSTM and GRU in all experimental conditions **with extensive evolutionary architecture search**

"Nonetheless, the fact a reasonable search procedure failed to dramatically improve over the LSTM suggests that, at the very least, if there are architectures that are much better than the LSTM, then they are not trivial to find." (Jozefowicz et al.)

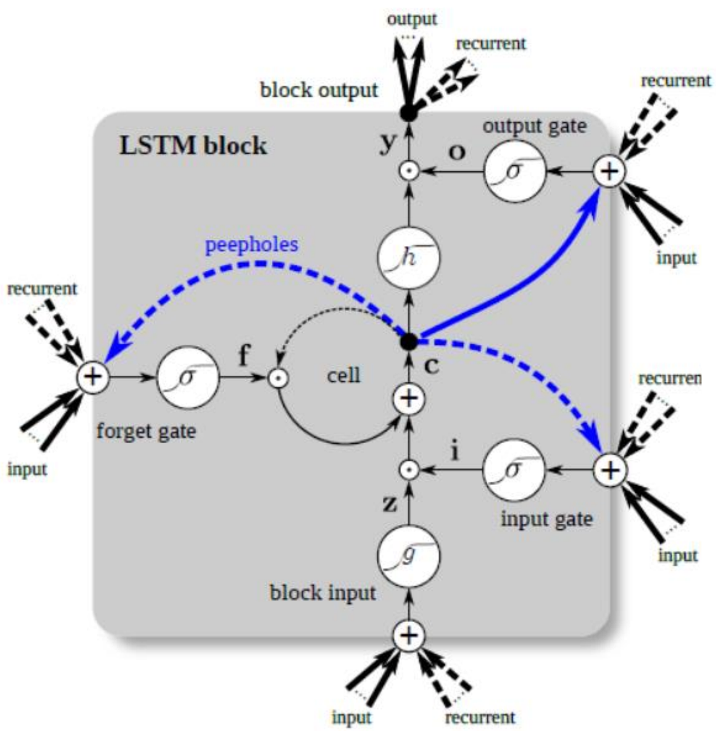
- **I. Variants**
  - **Safe Choice of Variants**

a) **Vanilla LSTM (with peephole)**
b) **LSTM without peephole**
c) **GRU**

• I. Variants

- **Safe Choice of Variants: LSTM with / without peephole**

a) Vanilla LSTM (with peephole)
b) LSTM without peephole
c) GRU



$$\mathbf{z}^t = g(\mathbf{W}_z\mathbf{x}^t + \mathbf{R}_z\mathbf{y}^{t-1} + \mathbf{b}_z) \quad \textit{block input}$$

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

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

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

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

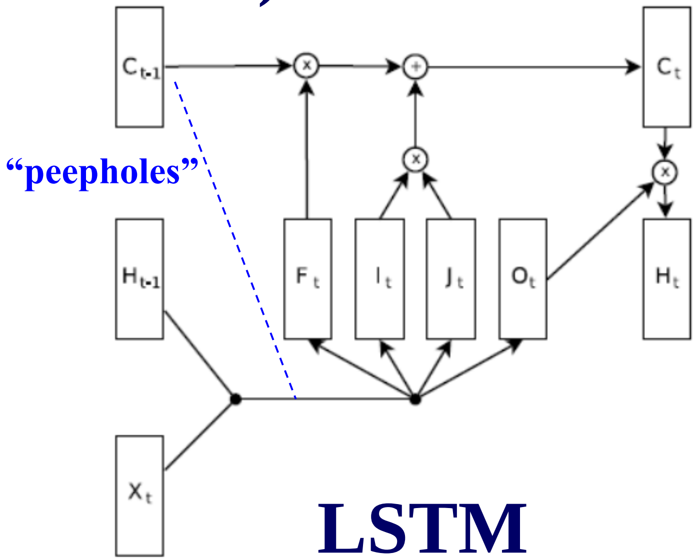
$$\mathbf{y}^t = \mathbf{o}^t \odot h(\mathbf{c}^t) \quad \textit{block output}$$

“peepholes”

• I. Variants

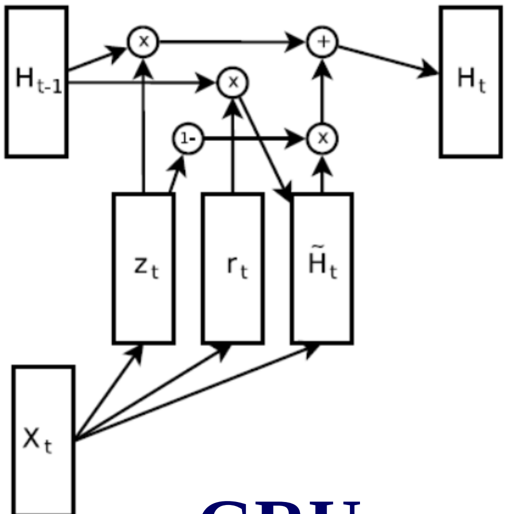
- **Safe Choice of Variants: LSTM and GRU**

a) Vanilla LSTM (with peephole)
b) LSTM without peephole
c) GRU



LSTM

$$
\begin{aligned}
i_t &= \tanh(W_{\mathrm{xi}} x_t + W_{\mathrm{hi}} h_{t-1} + b_{\mathrm{i}}) \\
j_t &= \mathrm{sigm}(W_{\mathrm{xj}} x_t + W_{\mathrm{hj}} h_{t-1} + b_{\mathrm{j}}) \\
f_t &= \mathrm{sigm}(W_{\mathrm{xf}} x_t + W_{\mathrm{hf}} h_{t-1} + b_{\mathrm{f}}) \\
o_t &= \tanh(W_{\mathrm{xo}} x_t + W_{\mathrm{ho}} h_{t-1} + b_{\mathrm{o}}) \\
c_t &= c_{t-1} \odot f_t + i_t \odot j_t \\
h_t &= \tanh(c_t) \odot o_t
\end{aligned}
$$

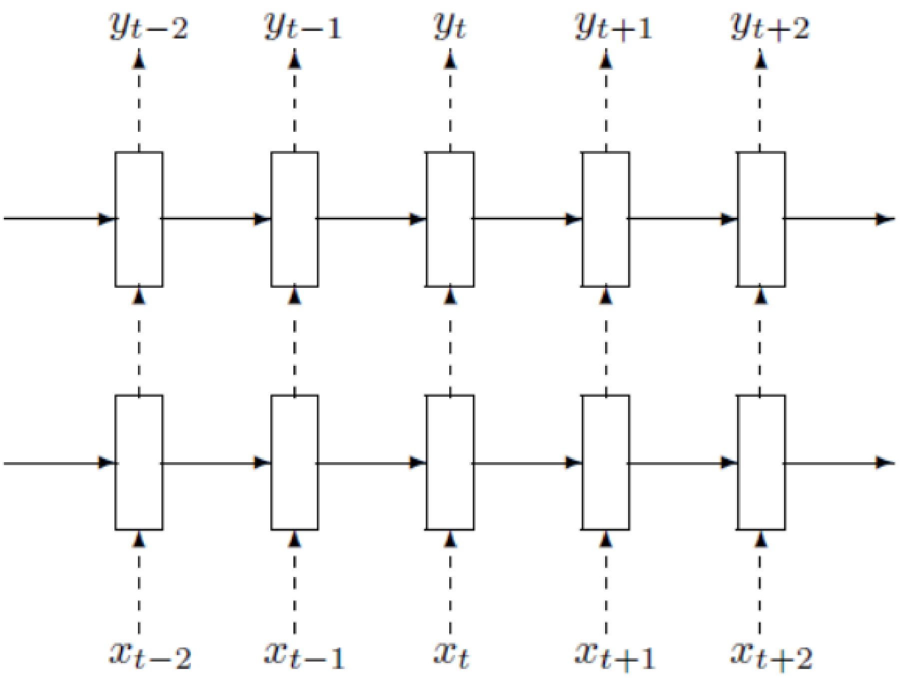


GRU

$$
\begin{aligned}
r_t &= \mathrm{sigm}\left(W_{\mathrm{xr}} x_t + W_{\mathrm{hr}} h_{t-1} + b_{\mathrm{r}}\right) \\
z_t &= \mathrm{sigm}(W_{\mathrm{xz}} x_t + W_{\mathrm{hz}} h_{t-1} + b_{\mathrm{z}}) \\
\tilde{h}_t &= \tanh(W_{\mathrm{xh}} x_t + W_{\mathrm{hh}}(r_t \odot h_{t-1}) + b_{\mathrm{h}}) \\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t
\end{aligned}
$$

• II. Regularization

- **First Proposed**



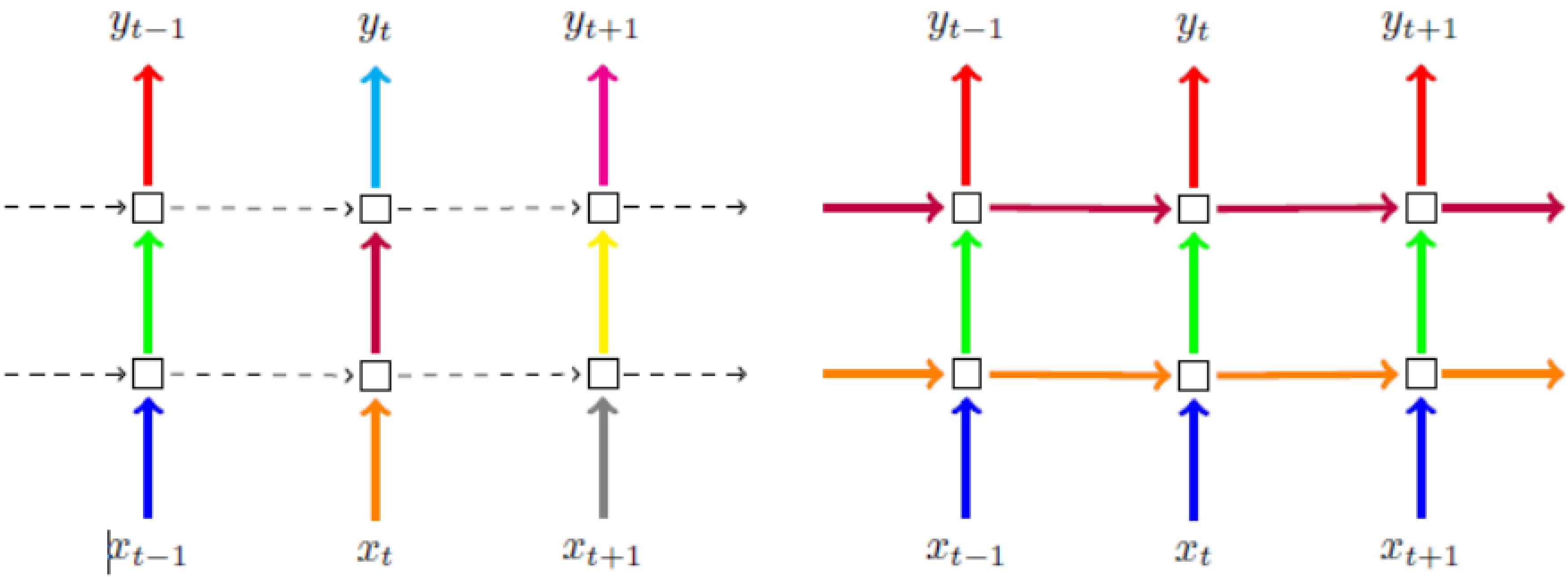
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} T_{2n,4n} \begin{pmatrix} \mathbf{D}(h_t^{l-1}) \\ h_{t-1}^l \end{pmatrix}$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

$$h_t^l = o \odot \tanh(c_t^l)$$

• II. Regularization

- **Variational RNN**



(a) Naive dropout RNN (b) Variational RNN

$$\begin{pmatrix} \underline{\mathrm{i}} \\ \underline{\mathrm{f}} \\ \underline{\mathrm{o}} \\ \underline{\mathrm{g}} \end{pmatrix} = \begin{pmatrix} \mathrm{sigm} \\ \mathrm{sigm} \\ \mathrm{sigm} \\ \mathrm{tanh} \end{pmatrix} \left(\begin{pmatrix} \mathbf{x}_t \circ \mathbf{z}_{\mathbf{x}}^t \\ \mathbf{h}_{t-1} \end{pmatrix} \cdot \mathbf{W} \right)$$

$$\begin{pmatrix} \underline{\mathrm{i}} \\ \underline{\mathrm{f}} \\ \underline{\mathrm{o}} \\ \underline{\mathrm{g}} \end{pmatrix} = \begin{pmatrix} \mathrm{sigm} \\ \mathrm{sigm} \\ \mathrm{sigm} \\ \mathrm{tanh} \end{pmatrix} \left(\begin{pmatrix} \mathbf{x}_t \circ \mathbf{z}_{\mathbf{x}} \\ \mathbf{h}_{t-1} \circ \mathbf{z}_{\mathbf{h}} \end{pmatrix} \cdot \mathbf{W} \right)$$

• II. Regularization

- **Other Keynotes in Gal and Ghahramani's work**

a) Word embedding dropout

As an example, the sentence
"the dog and the cat"
might become
"— dog and — cat" or "the — and the cat"
but never
"— dog and the cat"

b) MC dropout

[Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning]

- # III. Tricks

  - **Keras dropout**

a) **Dropout in Keras means the ratio to “drop”, not the ratio to “keep” (unlike in TensorFlow)**

b) **For RNN dropout,**

dropout_W

$$\begin{pmatrix} \underline{\mathbf{i}} \\ \underline{\mathbf{f}} \\ \underline{\mathbf{o}} \\ \underline{\mathbf{g}} \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} \left(\begin{pmatrix} \mathbf{x}_t \circ \mathbf{z_x} \\ \mathbf{h}_{t-1} \circ \mathbf{z_h} \end{pmatrix} \cdot \mathbf{W} \right)$$

dropout_U

c) **Word embedding dropout has already implemented in Keras**

• III. Tricks

- **Forget Gate Bias**

But most importantly, we determined that adding a positive bias to the forget gate greatly improves the performance of the LSTM. Given that this technique the simplest to implement, we recommend it for every LSTM implementation. Interestingly, this idea has already been stated in the paper that introduced the forget gate to the LSTM (Gers et al., 2000).

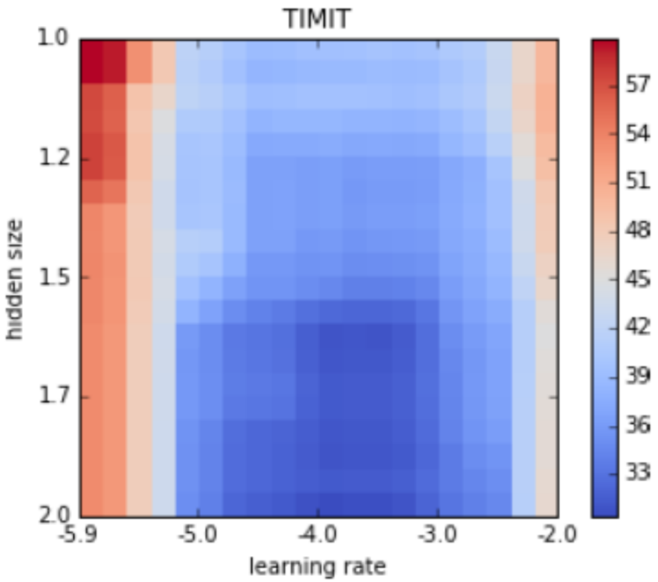
Already implemented in Keras

• III. Tricks

- **Critical components and Learning Rate Search**

a) **The forget gate and the output activation function are the critical components of the LSTM block.**

b) **Learning rate and network size are the most crucial tunable LSTM hyperparameters.**



• IV. More

- **Note on Vanishing Gradients**

a) **The LSTM addresses the vanishing gradient problem by reparameterizing the RNN**

b) **the LSTM directly computes S_t, which is then added to S_{t-1} to obtain S_t**

Given that $S_{1000} = \sum_{t=1}^{1000} \Delta S_t$, every single ΔS_t (including ΔS_1) will receive a sizeable contribution from the gradient at timestep 1000. This immediately implies that the gradient of the long-term dependencies cannot vanish. It may become "smeared", but it will never be negligibly small.

• IV. More

- **Note on Memory Cells and Hidden States**

"Thus the LSTM has two kinds of hidden states:

a "slow" state $\mathbf{C}_t$ *that fights the vanishing gradient problem, and*
a "fast" state $\mathbf{H}_t$ *that allows the LSTM to make complex decisions over short periods of time.*

It is notable that an LSTM with n memory cells has a hidden state of dimension 2n."

• IV. More

- **Missing data (with RNNs)**

X: Input time series (2 variables);
s: Timestamps for X;
M: Masking for X;
Δ: Time duration for X.

$$X = \begin{bmatrix} 47 & 49 & NA & 40 & NA & 43 & 55 \\ NA & 15 & 14 & NA & NA & NA & 15 \end{bmatrix}$$

$$s = [0 \quad 0.1 \quad 0.6 \quad 1.6 \quad 2.2 \quad 2.5 \quad 3.1]$$

$$M = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Delta = \begin{bmatrix} 0.0 & 0.1 & 0.5 & 1.5 & 0.6 & 0.9 & 0.6 \\ 0.0 & 0.1 & 0.5 & 1.0 & 1.6 & 1.9 & 2.5 \end{bmatrix}$$

$$\delta_t^d = \begin{cases} s_t - s_{t-1} + \delta_{t-1}^d, & t > 1, m_{t-1}^d = 0 \\ s_t - s_{t-1}, & t > 1, m_{t-1}^d = 1 \\ 0, & t = 1 \end{cases}$$

• IV. More

- **Missing data (with RNNs)**

Various Way to "fill" missing data (baseline):

a) with mean

$$x_t^d \leftarrow m_t^d x_t^d + (1 - m_t^d)\tilde{x}^d$$

b) with lagged

$$x_t^d \leftarrow m_t^d x_t^d + (1 - m_t^d)x_{t'}^d$$

c) mixed

$$x_t^{(n)} \leftarrow \left[x_t^{(n)}; m_t^{(n)}; \delta_t^{(n)}\right]$$

• IV. More

- **Missing data (with RNNs)**

Trainable decay models (proposed):

$$\gamma_t = \exp\{-\max(0, W_\gamma \delta_t + b_\gamma)\}$$

a) input decay

$$x_t^d \leftarrow m_t^d x_t^d + (1 - m_t^d)\gamma_t^d x_{t'}^d + (1 - m_t^d)(1 - \gamma_t^d)\tilde{x}^d$$

b) hidden decay

$$h_{t-1} \leftarrow \gamma_t \odot h_{t-1}$$

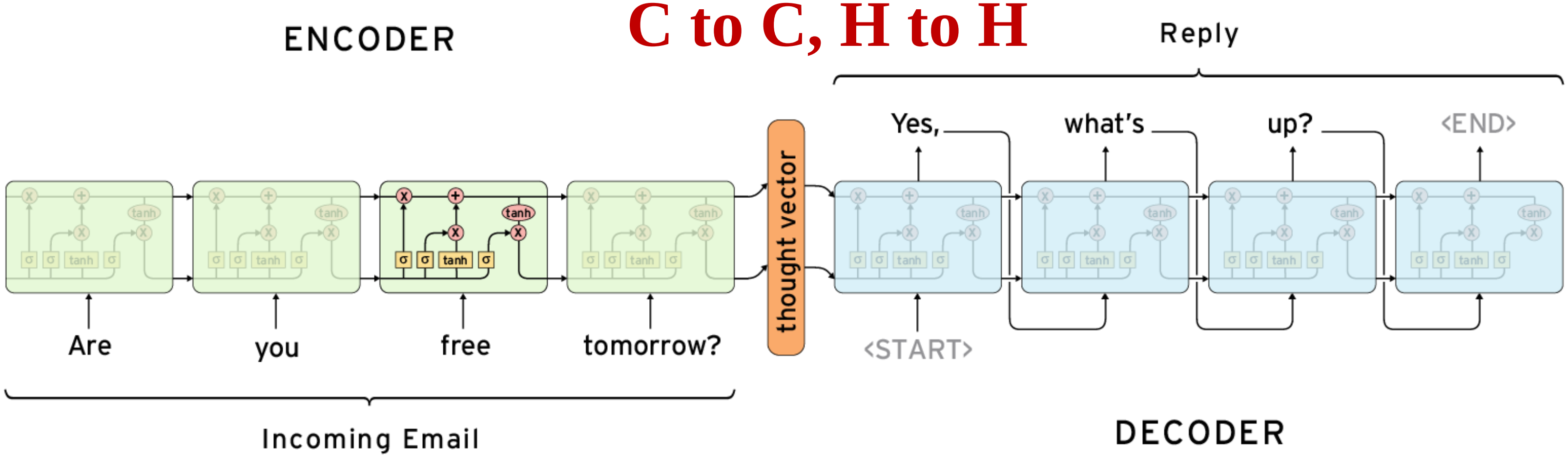
Predict missing data (proposed):

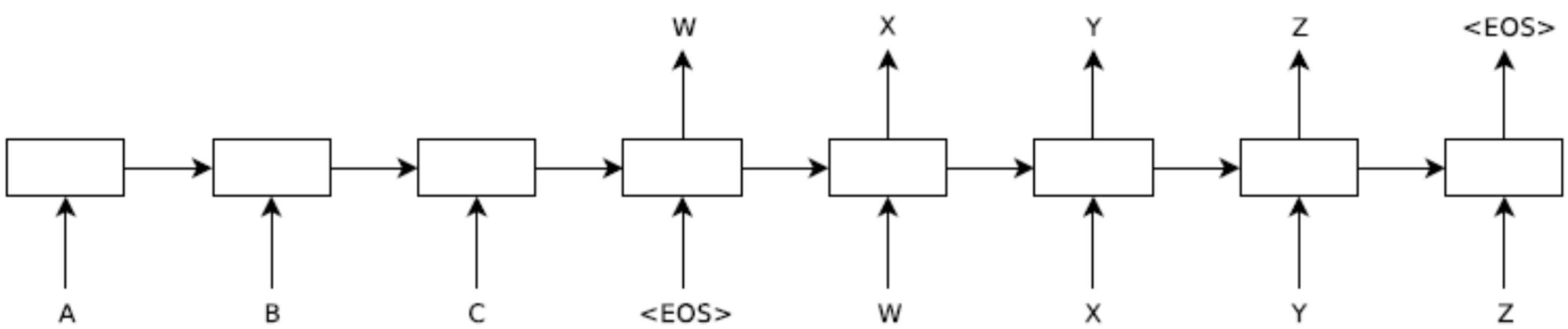
$$\ell = \ell_{log\_loss} + \lambda \frac{1}{N} \sum_{n=1}^{N} \frac{1}{T_n} \sum_{t=1}^{T_n} \frac{\sum_{d=1}^{D} m_t^d \cdot \log p(x_t^d | \mu_t^d, \sigma_t^d)}{\sum_{d=1}^{D} m_t^d}$$

• IV. More

- **Seq2Seq**

C to C, H to H



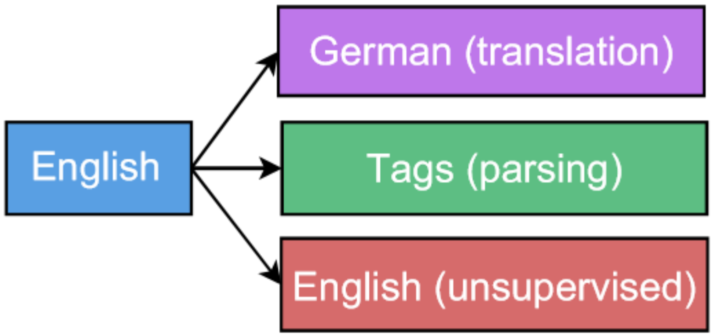


Note: Deep LSTM outperform Wide LSTM.

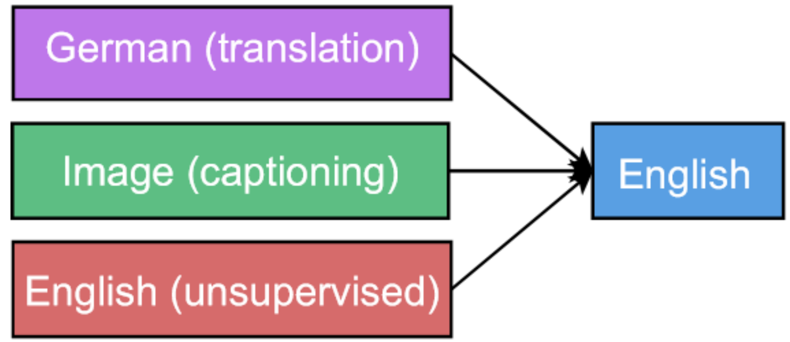
• IV. More

- **Seq2Seq (Multi-task)**

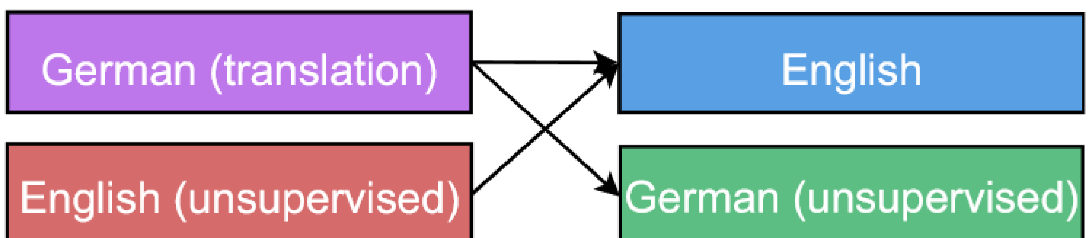
Train separately. With a ratio to switch the tasks.



One-to-many



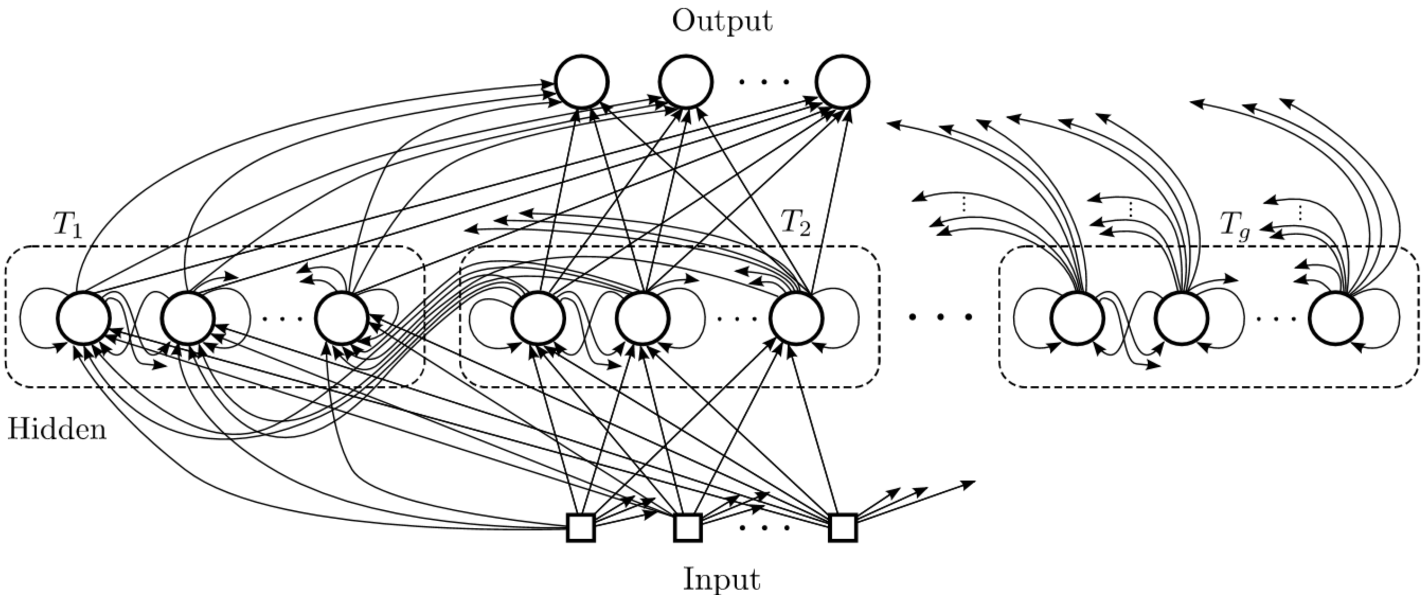
Many-to-one

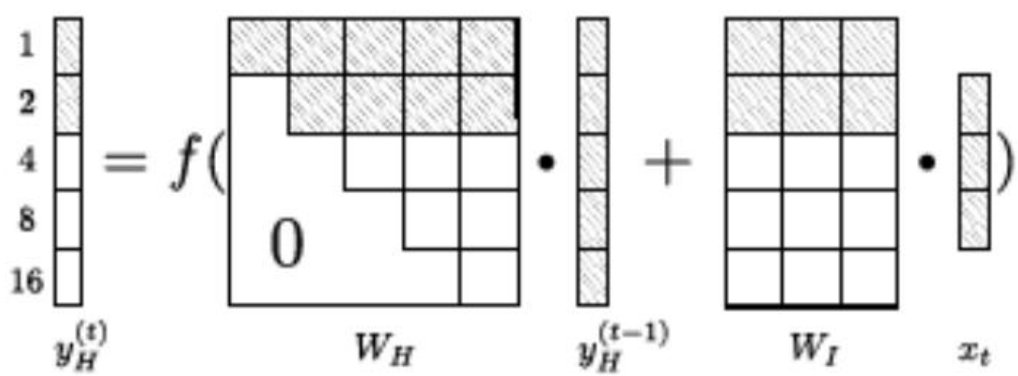


Many-to-many

• IV. More

- **Clockwalk RNN**





T=6

Bibliography

- LSTM: A Search Space Odyssey
- Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling
- A Clockwork RNN
- Sequence to Sequence Learning with Neural Networks
- Multi-task Sequence to Sequence Learning

Bibliography

- Recurrent Neural Networks for Multivariate Time Series with Missing Values
- An Empirical Exploration of Recurrent Network Architectures
- A Theoretically Grounded Application of Dropout in Recurrent Neural Networks
- Recurrent Neural Network Regularization
- Keras: Deep Learning library for Theano and TensorFlow

Thanks for listening!