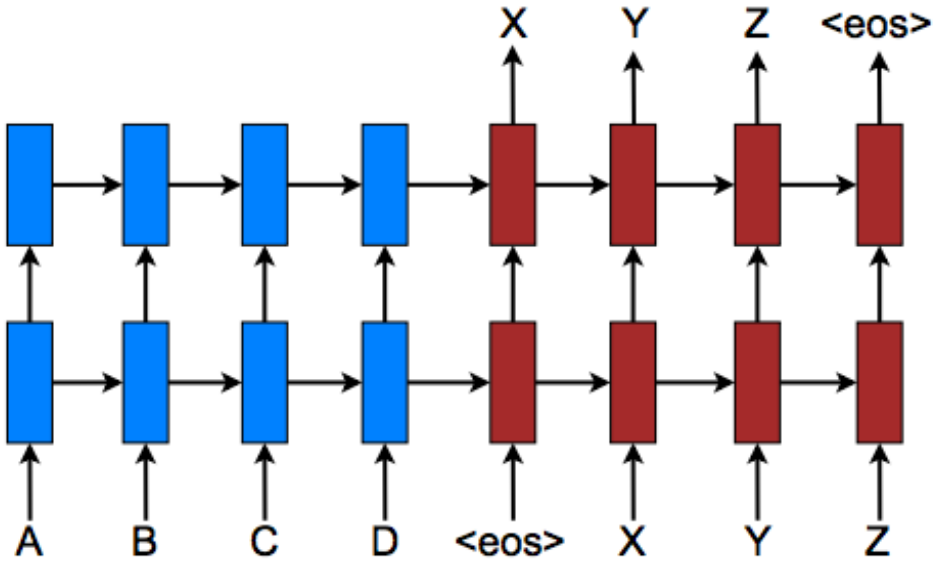
TTIC 31230, Fundamentals of Deep Learning

David McAllester, April 2017

Sequence to Sequence Models and Attention

Encode-Decode Architectures for Machine Translation

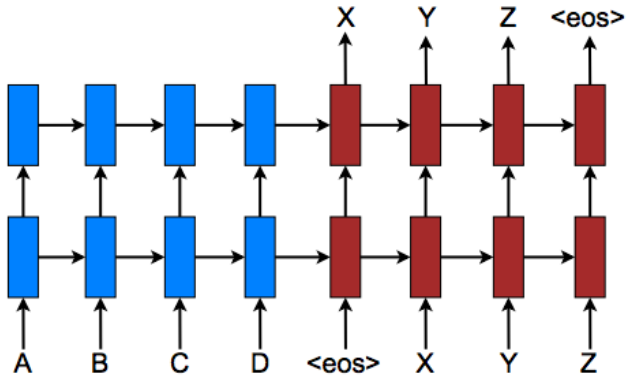


[Figure from Luong et al.]

In Sutskever et al. (2014) the LSTMs are layered 4 deep.

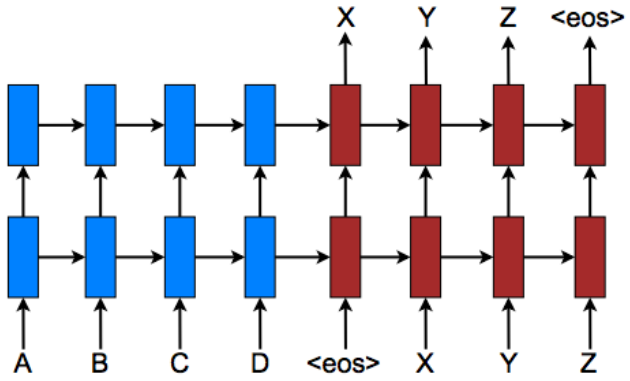
The input is reversed — "DCBA" is translated to "XYZ".

Encode-Decode Architecture for Machine Translation




```
H[0,t] = x[t]
H[l+1][t+1] = GRU(H[l+1][t],H[l][t],Theta[l+1])
y[0] = <eos>
S[l,0] = H[l,T]
S[l+1][t+1] = GRU(S[l+1][t],S[l][t],Psi[l+1])
P(y[t+1]) = Softmax W [e(y[t]),S[0,t],...,s[L,t]]
```

The Translation Probability Distribution



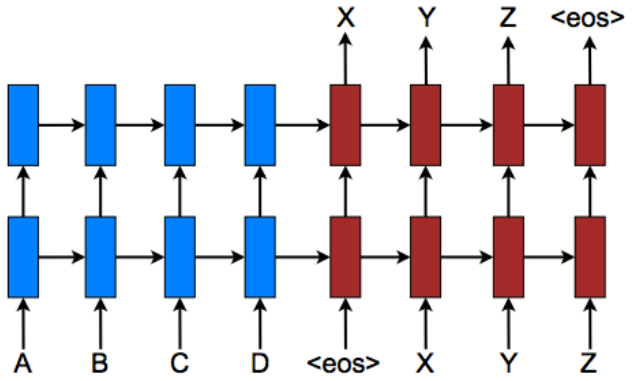
`P(y[t+1]) = Softmax W [e(y[t]),S[0,t],...,S[L,t]]`

$$P(XYZ|DCBA) = P(X|DCBA)\ P(Y|DCBDA; X)\ P(Z|DCBA; XY)$$

For the training pair $DCBA \Rightarrow XYX$ the training loss is

$$\log\ 1/P(XYZ \mid DCBA)$$

Encode-Decode Architecture for Machine Translation




```
H[0,t] = x[t]
H[l+1][t+1] = GRU(H[l+1][t],H[l][t],Theta[l+1])
y[0] = <eos>
S[l,0] = H[l,T]
S[l+1][t+1] = GRU(S[l+1][t],S[l][t],Psi[l+1])
P(y[t+1]) = Softmax W [e(y[t]),S[0,t],...,s[L,t]]
```

Greedy Decoding vs. Beam Search

We would like

$$y^* = \underset{y}{\operatorname{argmax}}\, P(y|x)$$

A **greedy algorithm** may do well

$$y_{t+1} = \underset{y}{\operatorname{argmax}}\ P(y \mid \mathbf{x},\ y_1, \ldots, y_t)$$

However a **beam search** will typically do somewhat better,

$$Y_{t+1} = \underset{y}{\operatorname{kbest}}\ P(y \mid \mathbf{x},\ Y_1, \ldots, Y_t)$$

Each $y \in Y_t$ is paired with a state vector (Viterbi Algorithm).

Training Details (Sutskever et a. (2014))

We used deep LSTMs with 4 layers, with 1000 cells at each layer and 1000 dimensional word embeddings, with an input vocabulary of 160,000 and an output vocabulary of 80,000.

We used a naive softmax over 80,000 words at each output.

We used batches of 128 sequences for the gradient and divided it by the size of the batch [128].

We used stochastic gradient descent without momentum, with a fixed learning rate of 0.7.

After 5 epochs, we begun halving the learning rate every half epoch. We trained our models for a total of 7.5 epochs.

Training Details

[We clipped gradients] by scaling [the gradient] when its norm exceeded a threshold. For each training batch, we compute $s = ||g||^2$ and for $s > 5$ we set $g = 5g/s$.

Different sentences have different lengths. ... To address this problem, we made sure that all sentences within a minibatch were roughly of the same length. [This gave] a 2x speedup.

Training Details

We parallelized our [C++] model using an 8-GPU machine.

Each layer of the LSTM was executed on a different GPU and communicated its activations to the next GPU (or layer) as soon as they were computed.

Our models have 4 layers of LSTMs, each of which resides on a separate GPU.

The remaining 4 GPUs were used to parallelize the softmax, so each GPU was responsible for multiplying by a 1000 20,000 matrix. [20,000 is 1/4 of the output vocabulary]

Training took about ten days with this implementation [on a training set of 348M French words and 304M English words].

Attention-Based Translation

BiGRUs

$$\vec{h}_{t+1} = \mathrm{GRU}(\vec{h}_t, x_t, \Theta)$$

$$\overleftarrow{h}_{t-1} = \mathrm{GRU}(\overleftarrow{h}_t, x_t, \Theta)$$

$$\overleftrightarrow{h}_t = \left[\vec{h}_t, \overleftarrow{h}_t\right] \qquad ([x, y] \text{ denotes vector concatenation})$$

Basic Sequence to Sequence Model

$$
\begin{aligned}
s_0 &= \vec{h}_T \\
y_0 &= \text{<eos>}
\end{aligned}
$$

$$
\begin{aligned}
P(\cdot \,|\mathbf{x},\ y_1, \ldots, y_i) &= \text{softmax}\ W_y\,[s_i,\ e(y_i)] \\
s_{i+1} &= \text{GRU}(s_i,\ e(y_i),\ \Theta_s)
\end{aligned}
$$

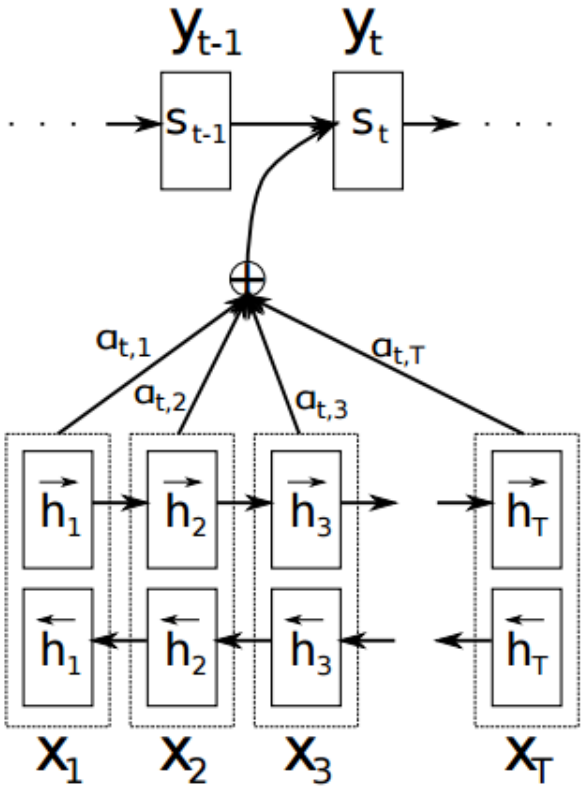
Adding Attention

$$
\begin{aligned}
c_0 &= \overleftrightarrow{h}_T \\
s_0 &= \vec{h}_T \\
y_0 &= \text{<eos>}
\end{aligned}
$$

$$
\begin{aligned}
P(\cdot \,|\mathbf{x},\ y_1, \ldots, y_i) &= \text{softmax}\ W_y\,[s_i,\ e(y_i),\ c_i] \\
s_{i+1} &= \text{GRU}(s_i,\ [e(y_i),\ c_i],\ \Theta_s)
\end{aligned}
$$

$$
\begin{aligned}
c_{i+1} &= \sum_t \alpha_{i,t}\ \overleftrightarrow{h}_t \\
\alpha_{i+1,t} &= \underset{t}{\text{softmax}}\ \tanh(W_a\,[s_i,\ \overleftrightarrow{h}_t])
\end{aligned}
$$

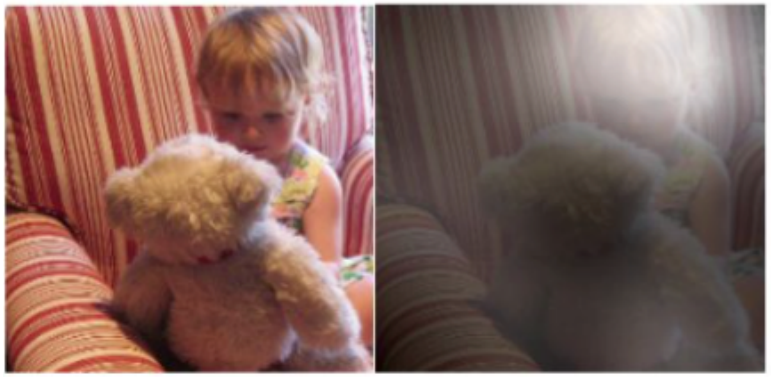
Attention



[Bahdanau, Cho, Bengio (2014)]

Attention in Image Captioning

A woman is throwing a frisbee in a park.



A little girl sitting on a bed with a teddy bear.

Xu et al. ICML 2015

Attention in Image Captioning

A dog is standing on a hardwood floor.

A group of people sitting on a boat in the water.

Xu et al. ICML 2015

Attention in Image Captioning

A stop sign is on a road with a mountain in the background.

A giraffe standing in a forest with trees in the background.

Xu et al. ICML 2015

Phrase Based Statistical Machine Translation (SMT)

Phrased Based SMT

Step I: Learn a phrase table — a set of triples (p, q, s) where

- p is a (short) sequence of source words.
- q is a (short) sequence of target words.
- s is a score.

("au", "to the", .5) ("au banque", "the the bank", .01)

For a phrase P we will write P.source for the source phrase, P.target for the target phrase, and P.score for the score.

Derivations

Consider an input sentence x of length T.

We will write $x[s:t]$ for the substring $x[s], \ldots, x[t-1]$.

A derivation d from x is a sequence $(P_1, s_1, t_1,), \ldots, (P_K, s_K, t_K)$ where P_k.source $= x[s_k : t_k]$.

The substrings $x[s_k : t_k]$ should be disjoint and "cover" x.

For $d = [(P_1, s_1, t_1,), \ldots, (P_L, s_K, t_K)]$ we define

$$y(d) \equiv P_1.\text{target} \ \cdots P_K.\text{target}$$

We let $D(x)$ be the set of derivations from x.

Scoring

For $d \in D(x)$ we define a score $s(d)$

$$s(d) = \alpha \ln P_{\mathrm{LM}}(y(d)) + \beta \sum_k P_k.\mathrm{score} + \gamma\ \mathrm{distortion}(d)$$

where $P_{\mathrm{LM}}(y)$ is the probability assigned to string y under a language model for the target language

and distortion(d) is a measure of consistency of word ordering between source and target strings as defined by the indeces $(s_1, t_1), \ldots, (s_K, t_K)$.

Translation

$$y(x) = y(d^*(x))$$

$$d^*(x) = \underset{d \in D(x)}{\text{argmax}} \; s(d)$$

END