Deep Learning基礎講座

Natural Language Processing

Day 2

講義シラバス

Day 1:　前処理、Word2Vec

Day 2:　Encoder-Decoderモデル

Day 3:　Transformerモデル

NLPのカテゴリ例

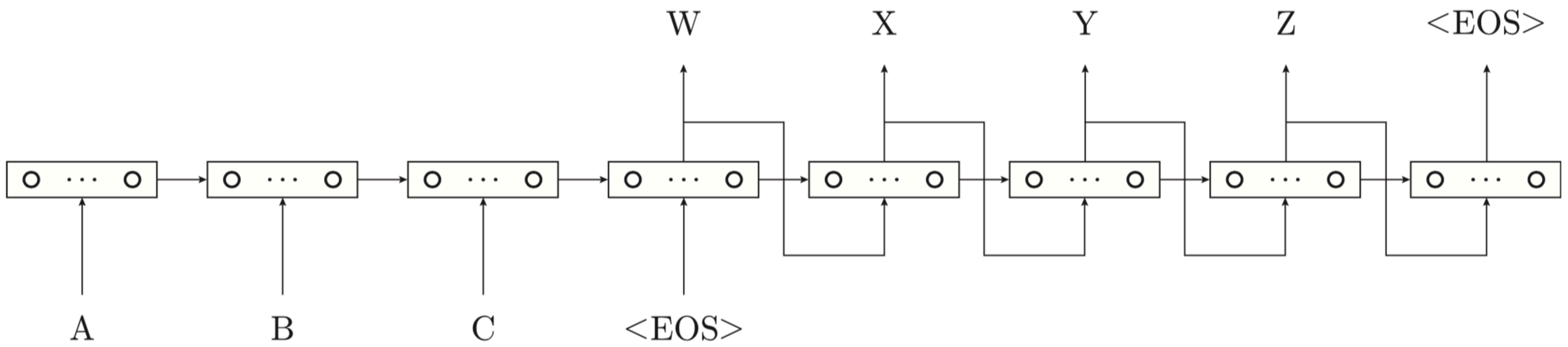
基礎技術：　品詞タグ付け、単語分割、構文解析、固有表現抽出

応用技術：　**機械翻訳**、文書分類、Q&A、対話

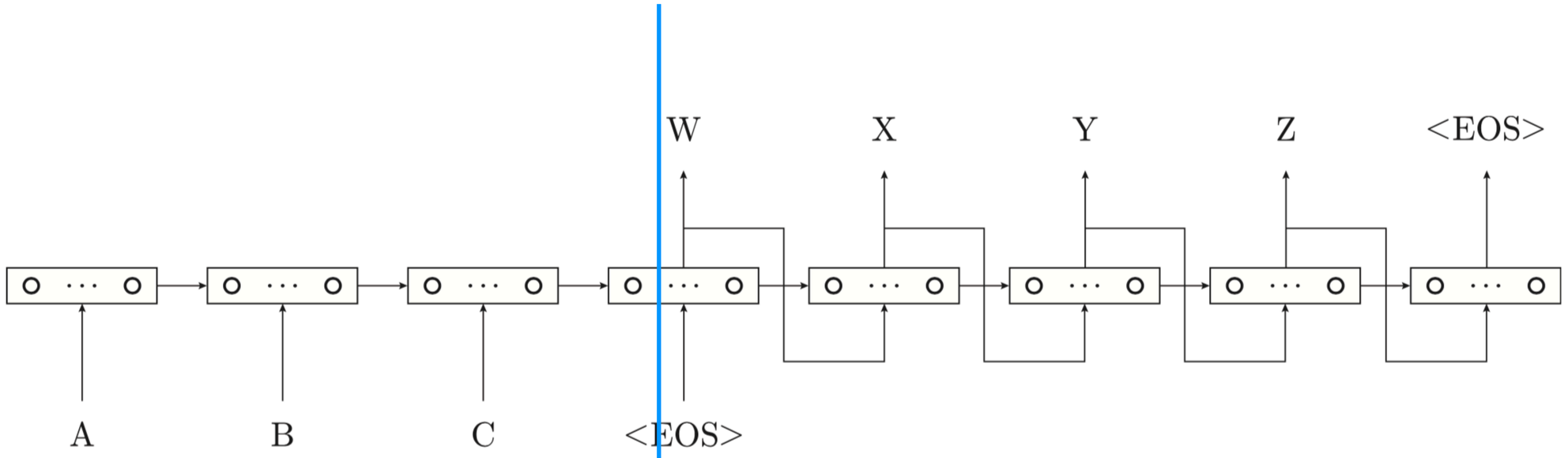
▼

入力も出力も時系列の形式

Encoder-Decoder



Encoder-Decoder



Encoder

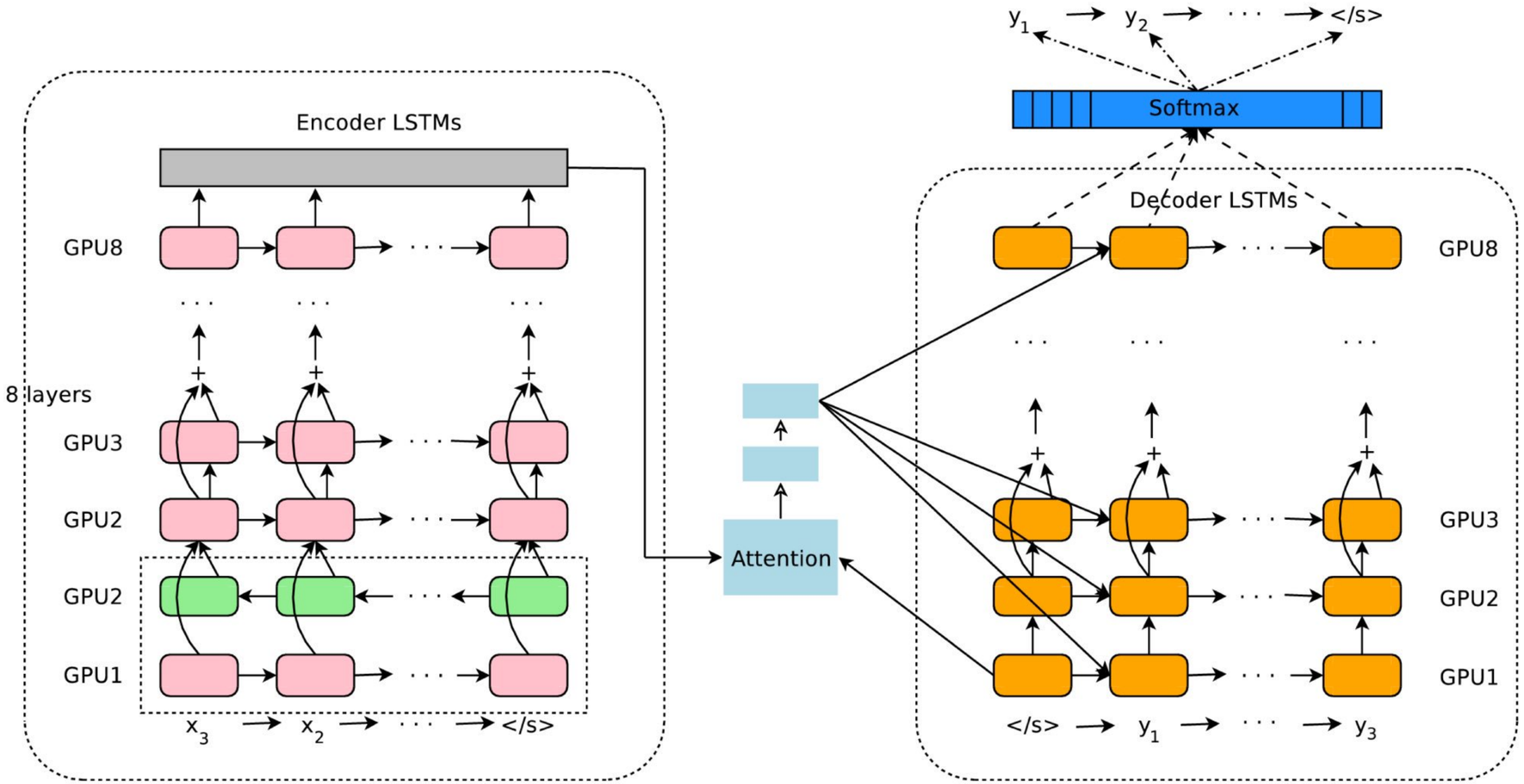
Decoder

Encoder-Decoderは２つのネットワークで構成されるモデル

- Encoder: 入力の時系列用のネットワーク

- Decoder: 出力の時系列用のネットワーク

cf. Google翻訳


Encoder LSTMs
GPU8
8 layers
GPU3
GPU2
GPU2
GPU1
x3
x2
</s>
Attention
y1
y2
</s>
Softmax
Decoder LSTMs
GPU8
GPU3
GPU2
GPU1
</s>
y1
y3

注意：

Encoder, Decoder はRNNであるとは限らない

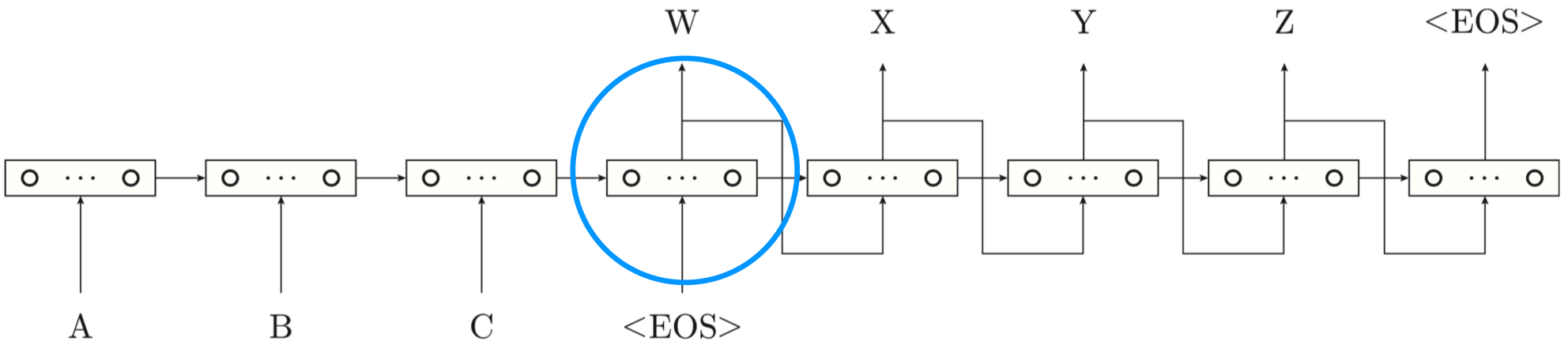
- 画像キャプションではEncoderはCNN

- 最近ではそもそも翻訳タスクにRNNを用いなくなっている
（詳しくはDay3にて）

Encoder-Decoder の概観

- Encoderの実装はネットワークそのままのケースがほとんど

- 逆に、Decoderではいくつか気をつけるべき点がある



EncoderとDecoderは別ネットワークだが、モデル全体としてはひとつなぎ

Encoderの（隠れ層の）状態は、Decoderの初期値として設定する必要がある

Decoderでは学習の仕方も２通りある

1. Decoderの各時刻の出力を次の入力として学習

2. Decoderへの入力は教師データをそのまま使って学習

(Teacher Forcing)

1. Decoderの各時刻の出力を次の入力として学習

2. Decoderへの入力は教師データをそのまま使って学習

1 だと誤差がどんどん大きくなり学習が不安定になる

2 だと学習は安定するが、評価時とはデータ分布が異なる

⇢ **Teacher Forcing を確率的に取り入れる**

Scheduled Sampling を用いる

生成文の評価について：BLEUスコア

- Bilingual Evaluation Understudy（BLUEではなくBLEU）

- 単なる予測誤差ではなく「どれくらい自然な翻訳ができているか」を評価

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} \frac{1}{N} \log p_n\right)$$

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} \frac{1}{N} \log p_n\right)$$

Brevity Penalty

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c \geq r \\ \exp(1 - r/c) & \text{if } c < r \end{cases}$$

c : 生成文の長さ（candidate）

r : 正解文の長さ（reference）

⇢ 短かすぎる文にはペナルティを与える

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} \frac{1}{N} \log p_n\right)$$

Modified n-gram precision

- Reference中の単語は一回しか使えないという制約下でprecisionを計算

例： Candidate: the the the the the the the

Reference: The cat is on the mat

⇢ Precision = 1, Modified = 1/7

（実際はこれをn-gramに対して計算する）

生成文の評価について：Greedy Search と Beam Search

Greedy:

各時刻で最も確率の高い単語を正解として採用し、次のステップでの入力として使う

Beam:

各時刻において一定の数のそれまでのスコアが高い文を保持しながら選択を行う

（演習で実装あり）

演習

(Define-by-RunのほうがNLPはかなり実装しやすい！)

Attentionについて

Encoder-Decoder は一定性能を誇るモデルだが、

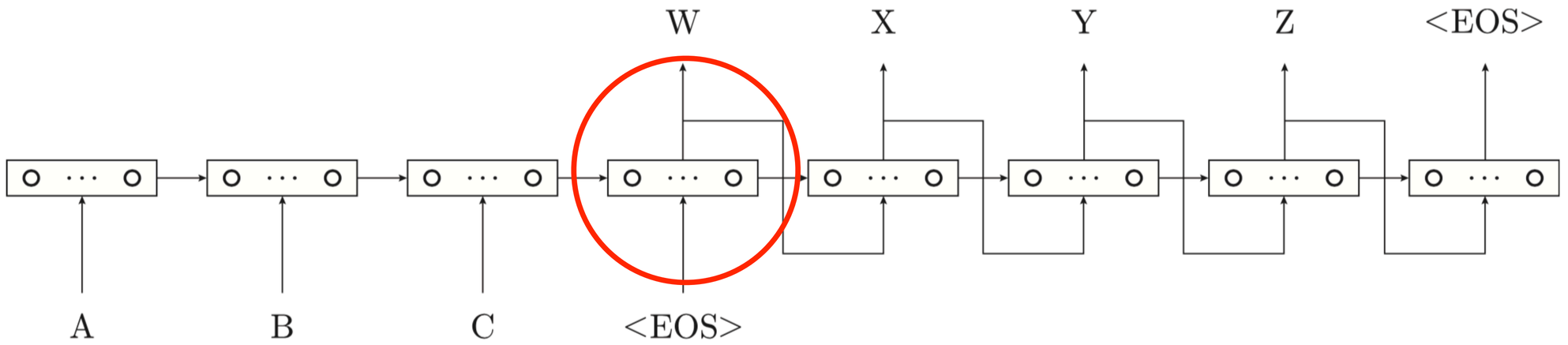
よくよく考えると、不要な処理を行っている点がある

入力系列が与えられたときの出力系列が得られる確率：

$$p(\boldsymbol{y}(1), \ldots, \boldsymbol{y}(T') \mid \boldsymbol{x}(1), \ldots, \boldsymbol{x}(T)) = \prod_{t=1}^{T'} p(\boldsymbol{y}(t) \mid \boldsymbol{y}(1), \ldots, \boldsymbol{y}(t-1), \boldsymbol{c})$$

入力系列が与えられたときの出力系列が得られる確率：

$$p(\boldsymbol{y}(1), \ldots, \boldsymbol{y}(T') \mid \boldsymbol{x}(1), \ldots, \boldsymbol{x}(T)) = \prod_{t=1}^{T'} p(\boldsymbol{y}(t) \mid \boldsymbol{y}(1), \ldots, \boldsymbol{y}(t-1), \boldsymbol{c})$$



入力が持つ「文脈」情報は、EncoderとDecoderの境界部分に集約されてしまう

- 時系列情報が、結局は固定長のベクトル $\boldsymbol{c}$ として表現されることになる

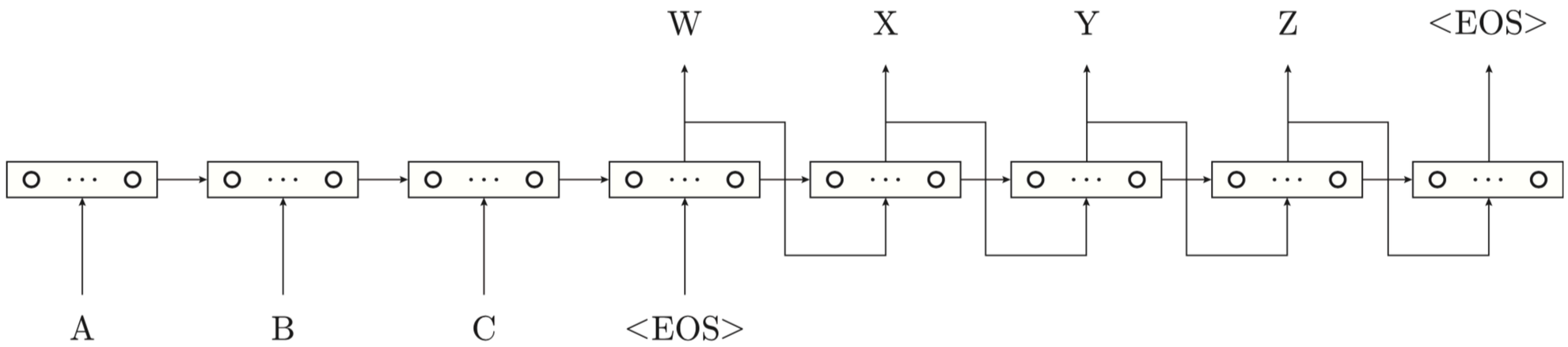
しかし、本来は各時刻によって過去のどの時刻を重視すべきなのかは異なるはず

入力系列がもつ情報を1つのベクトルに集約する必然性はない

時間の重みを考慮し、各時刻によって動的に変わるベクトルを考えれば、
よりよいモデルが得られるはず

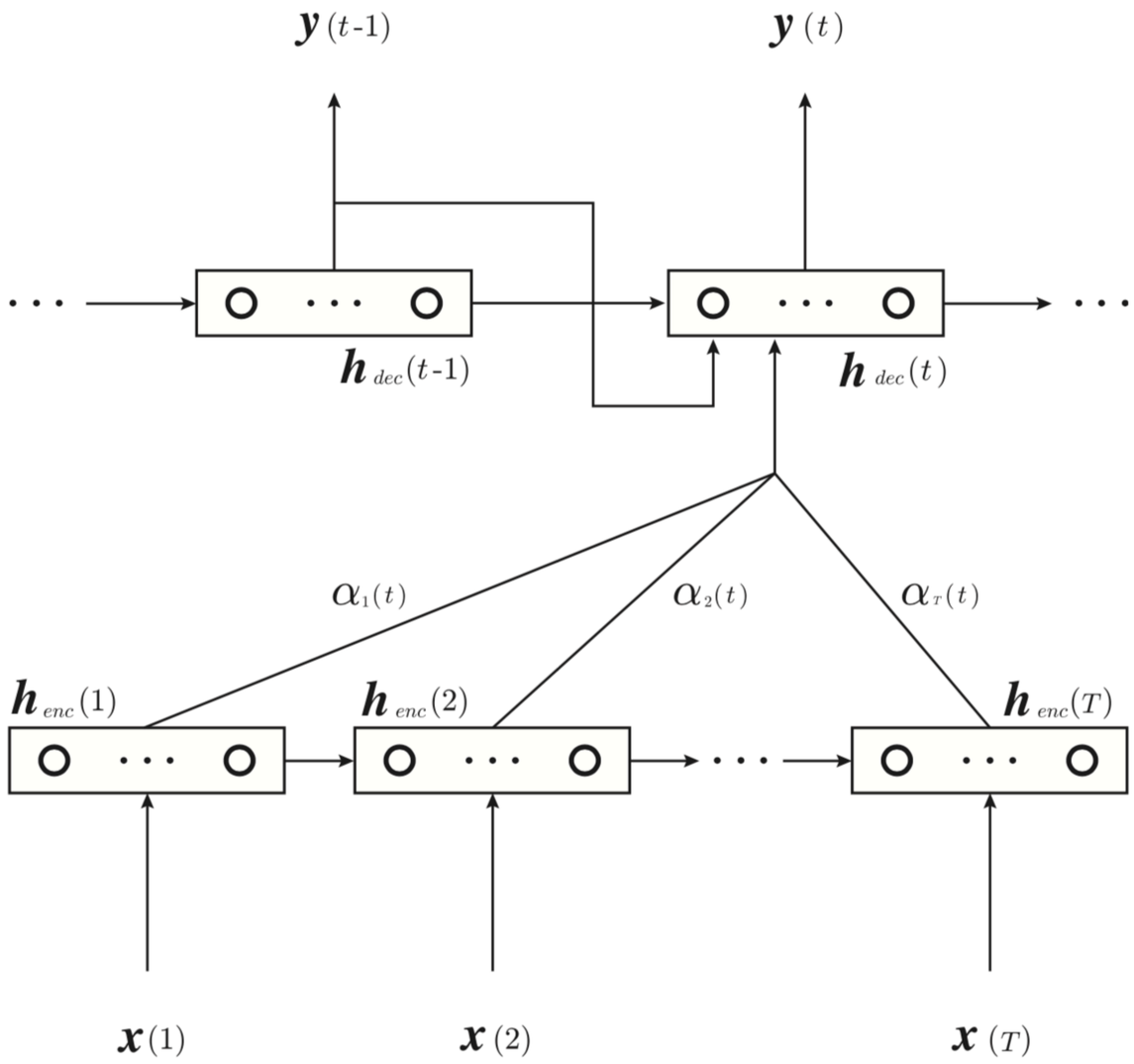
⇢ **Attention**

Encoder-Decoder



Attentionを取り入れた

RNN Encoder-Decoder



$$\boldsymbol{h}_{dec}(t) = f(\boldsymbol{h}_{dec}(t-1),\ \boldsymbol{y}(t-1),\ \boldsymbol{c})$$

$$\boldsymbol{h}_{dec}(t) = f(\boldsymbol{h}_{dec}(t-1),\ \boldsymbol{y}(t-1),\ \boldsymbol{c}(t))$$

時間を考慮

$$\boldsymbol{c}(t) = \sum_{\tau=1}^{T} \alpha_\tau(t)\boldsymbol{h}_{enc}(\tau)$$

各エンコーダの値をどれだけデコーダに伝えるかの割合（＝ 重み）

$\alpha_\tau(t)$　は和が１になるので、

$w_\tau(t) := f(\boldsymbol{h}_{dec}(t-1), \boldsymbol{h}_{enc}(\tau))$　とおくと（＝最適化すべき重み）

$$\alpha_\tau(t) = \frac{\exp(w_\tau(t))}{\sum_{\rho=1}^{T} \exp(w_\rho(t))} = \mathrm{softmax}\,(w_\tau(t))$$

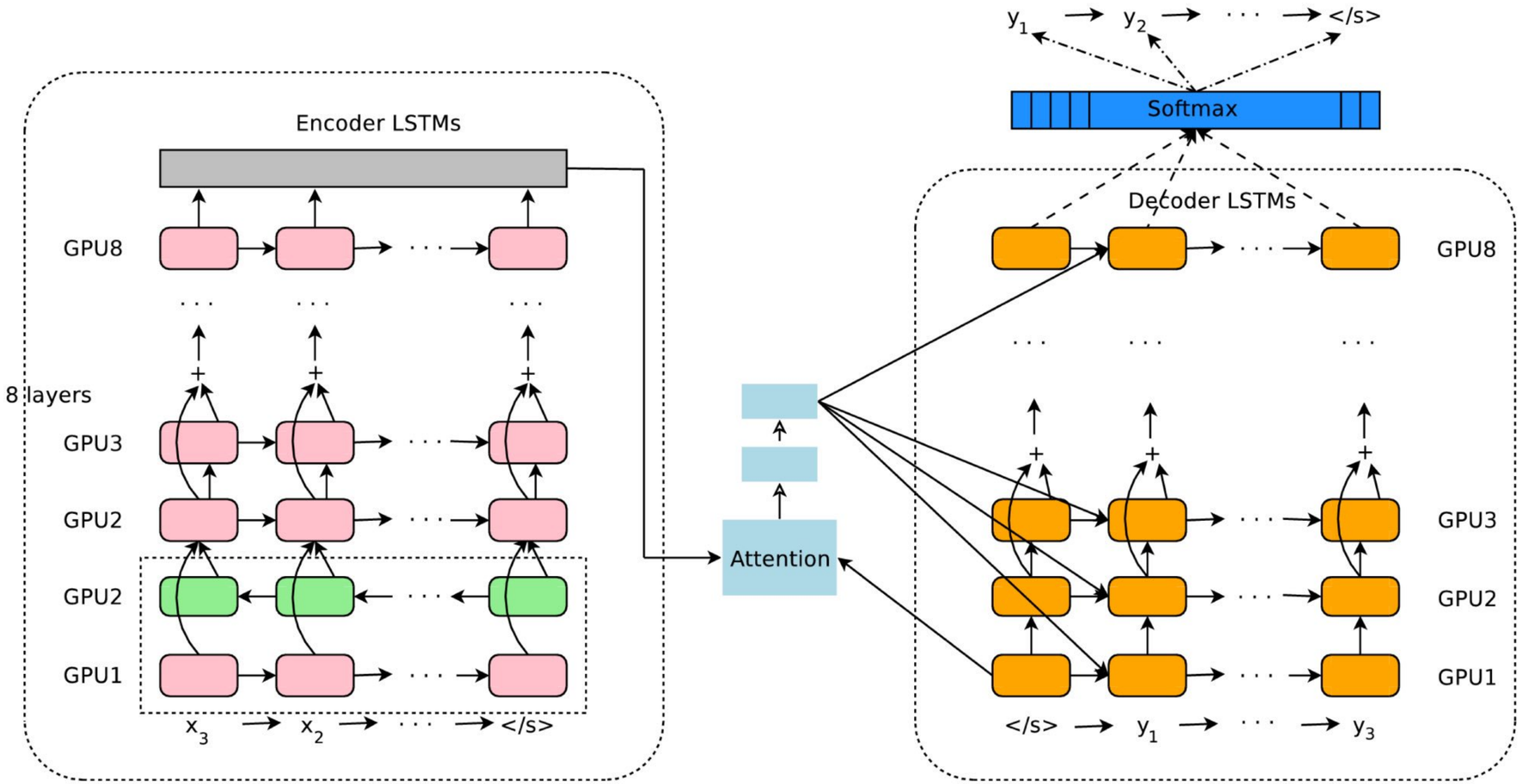
$$
\begin{aligned}
w_\tau(t) &= f(\boldsymbol{h}_{dec}(t-1),\, \boldsymbol{h}_{enc}(\tau)) \\
&= \boldsymbol{v}_a^T \tanh\left(W_a\, \boldsymbol{h}_{dec}(t-1) + U_a\, \boldsymbol{h}_{enc}(\tau)\right)
\end{aligned}
$$

ただし、重みに他の関数を用いるケースや、

Attentionの作り方にも別パターンがある

（再掲）

cf. Google翻訳



参考：LSTMにおけるAttention

LSTMブロックの出力

$$h(t) = f\left(h(t-1), x(t)\right)$$

⇢ この式も、直前の時刻 t - 1 にこれまでのすべての時系列情報が集約されている形になっている

セル（CEC）自体に、時間の重みをもたせられないか？

⇢ 覗き穴結合なしのLSTMで考えてみる

$$\begin{pmatrix} \boldsymbol{i}(t) \\ \boldsymbol{o}(t) \\ \boldsymbol{f}(t) \\ \boldsymbol{a}(t) \end{pmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \cdot \begin{pmatrix} W_i & U_i & \boldsymbol{b}_i \\ W_o & U_o & \boldsymbol{b}_o \\ W_f & U_f & \boldsymbol{b}_f \\ W_a & U_a & \boldsymbol{b}_a \end{pmatrix} \begin{pmatrix} \boldsymbol{x}(t) \\ \boldsymbol{h}(t-1) \\ 1 \end{pmatrix}$$

$$\boldsymbol{c}(t) = \boldsymbol{i}(t) \odot \boldsymbol{a}(t) + \boldsymbol{f}(t) \odot \boldsymbol{c}(t-1)$$

それぞれ、時間の重み

を考慮した項へ変更

$$\begin{pmatrix} \tilde{\boldsymbol{h}}(t) \\ \tilde{\boldsymbol{c}}(t) \end{pmatrix} = \sum_{\tau=1}^{t-1} \alpha_\tau(t) \begin{pmatrix} \boldsymbol{h}(\tau) \\ \boldsymbol{c}(\tau) \end{pmatrix}$$

$$\alpha_\tau(t) = \text{softmax}\left(w_\tau(t)\right)$$

$$\begin{aligned} w_\tau(t) &:= g\left(\boldsymbol{x}(t),\ \tilde{\boldsymbol{h}}(t-1),\ \boldsymbol{h}(\tau)\right) \\ &= \boldsymbol{v}^T \tanh\left(W_x\,\boldsymbol{x}(t) + W_{\tilde{\boldsymbol{h}}}\,\tilde{\boldsymbol{h}}(t-1) + W_h\,\boldsymbol{h}(\tau)\right) \end{aligned}$$