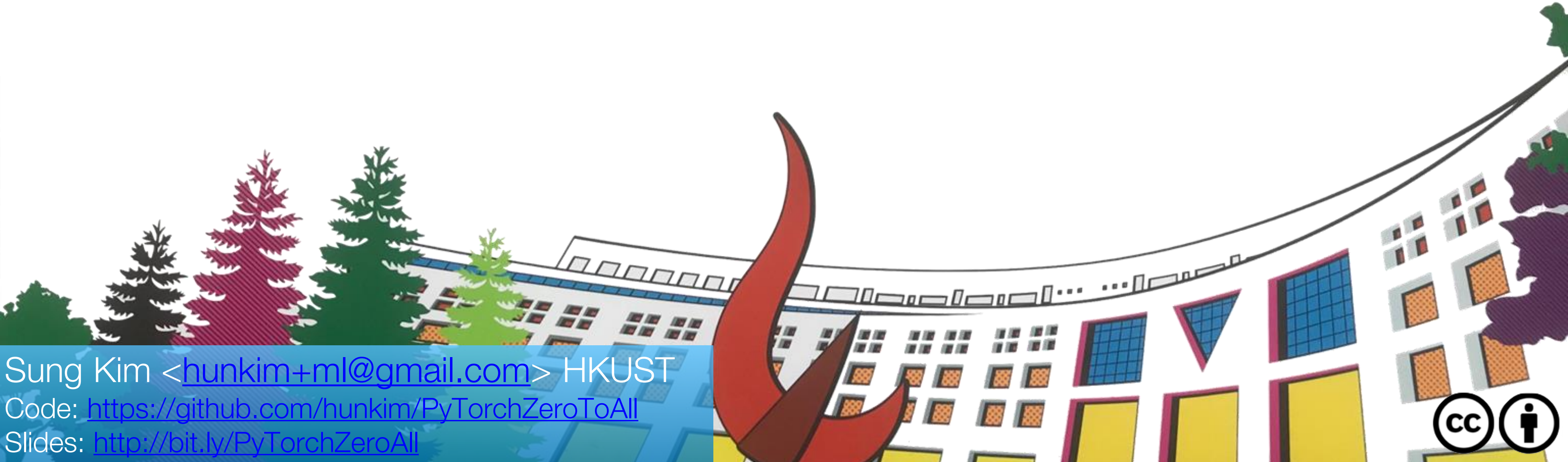
ML/DL for Everyone with PYTORCH

Lecture 14: Sequence to Sequence



Sung Kim <hunkim+ml@gmail.com> HKUST
Code: https://github.com/hunkim/PyTorchZeroToAll
Slides: http://bit.ly/PyTorchZeroAll

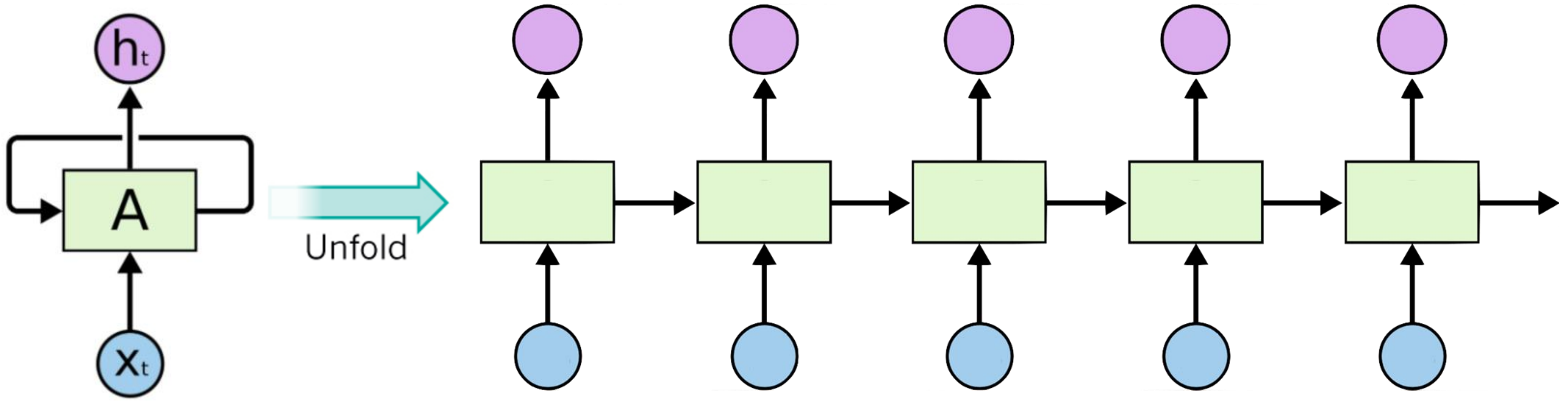
Call for Comments

Please feel free to add comments directly on these slides.

Other slides: http://bit.ly/PyTorchZeroAll

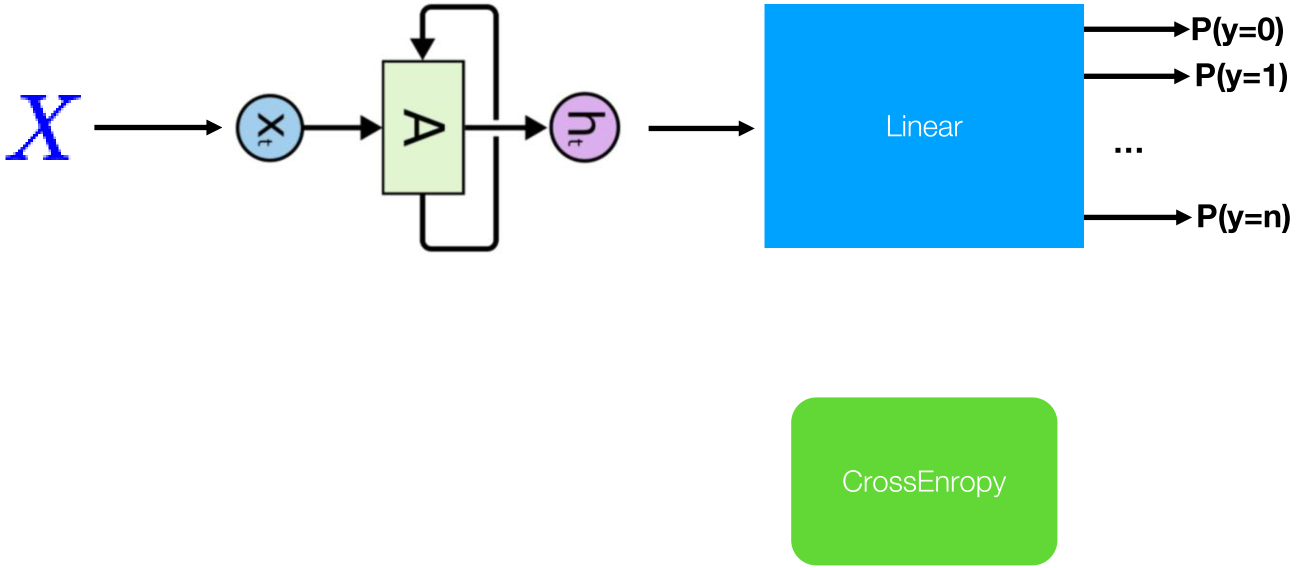
Picture from http://www.tssablog.org/archives/3280

DNN, CNN, RNN

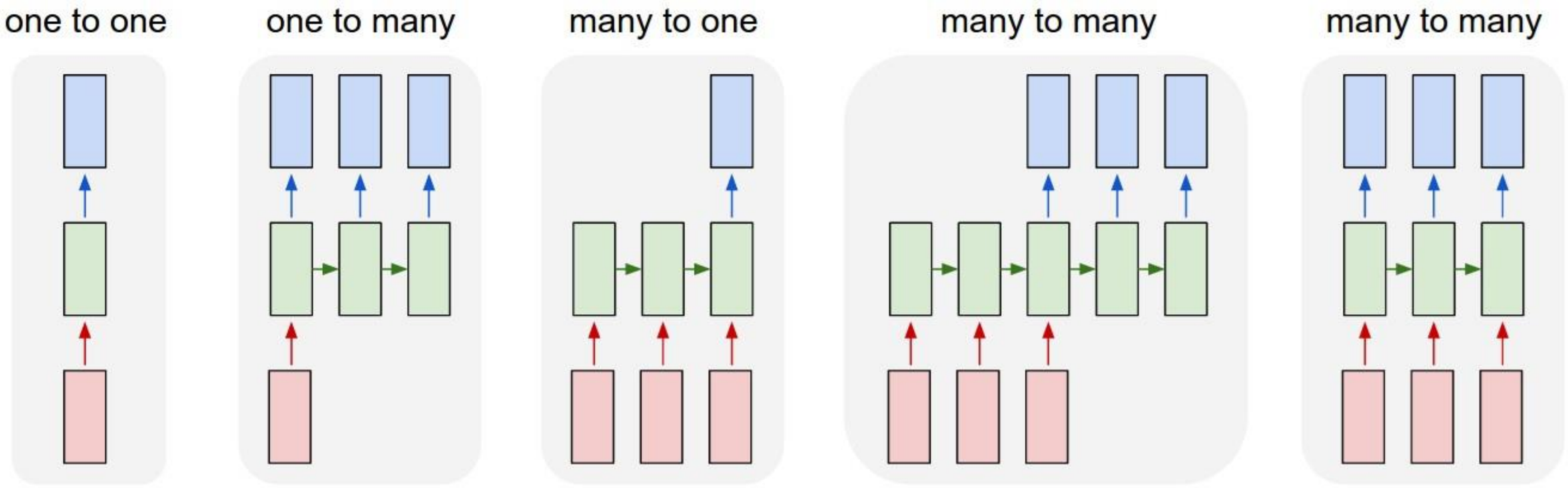


http://practicalquant.blogspot.hk/2013/10/deep-learning-oral-traditions.html

RNN

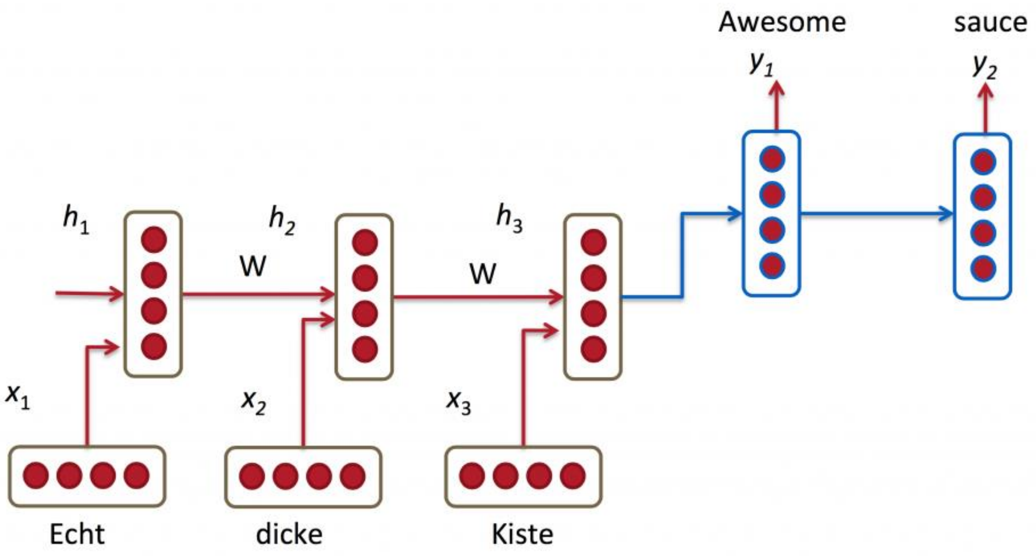


RNN Applications



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Sequence to Sequence



http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/

Lecture (TBA)

Implement Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation: https://arxiv.org/abs/1406.1078

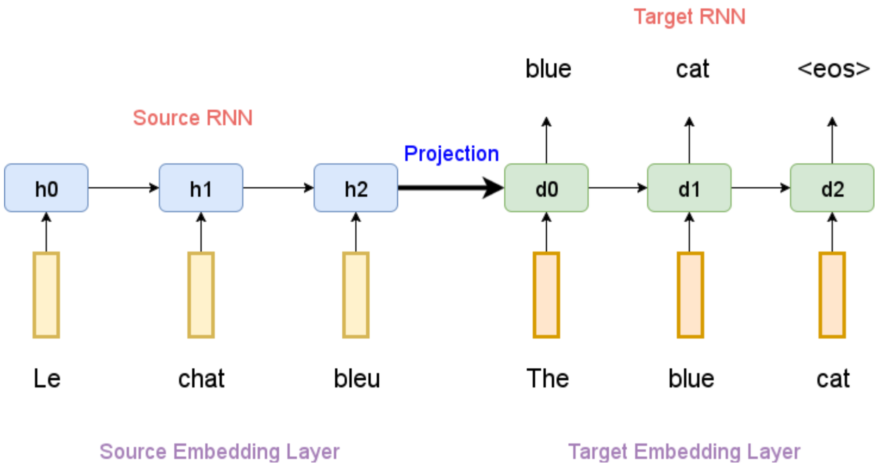
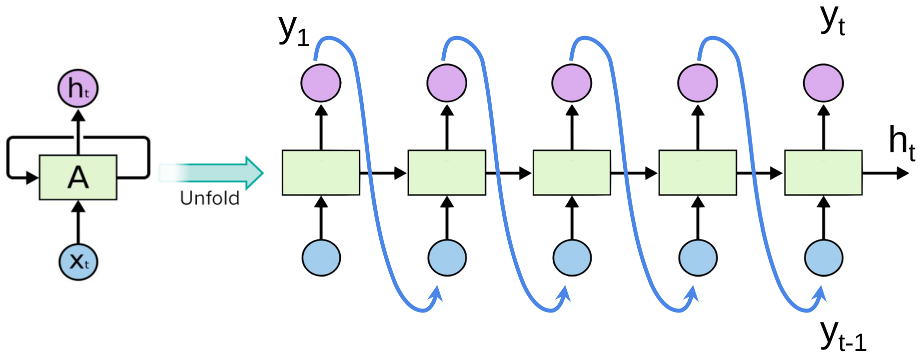


Image source: https://github.com/MaximumEntropy/Seq2Seq-PyTorch

Summary: RNN (WIP)



Modeling: $p(y)$

$$\log p(y) = \sum \log p(y_t | y_{<t})$$

$$h_t = f(h_{t-1}, y_{t-1})$$

$$p(y_t | y_{<t}) = \mathrm{softmax}(g(h_t))$$

Algorithm 1 RNN training loop

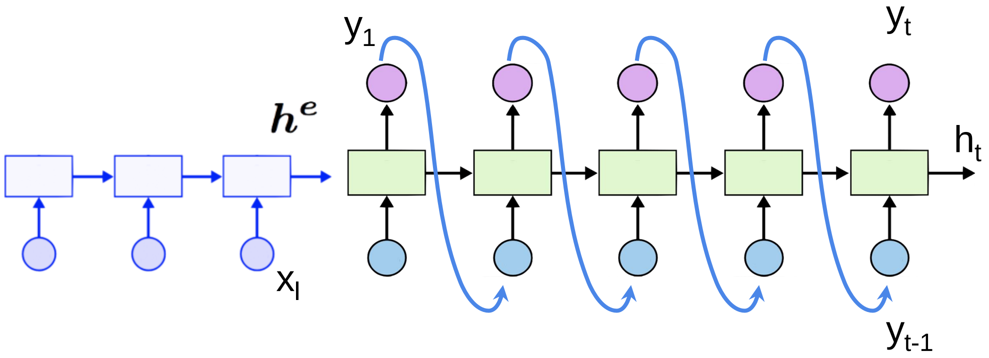
procedure TRAINING LOOP
    $h_0 = 0$
    $in_0 = 0$
    $l = 0$
    for each t from 1 to T **do**
        $h_t, out_t = \text{RNN-Module}(h_{t-1}, in_{t-1})$
        $l = l + \text{loss}(out_t, y_t)$
        $in_t = out_t$
    end for
end procedure

```
loss = 0
hidden = model.init_hidden()
Input = one_hot(labels[0])

for label in labels:
    hidden, output = model(hidden, input)
    loss += criterion(output, label)
    input = one_hot(output.max(1))

loss.backward()
optimizer.step()
```

Summary: S2S (WIP)



$$\text{Modeling: } p(y|x)$$

$$\boldsymbol{h^e} = \text{encoder}(x)$$

$$\log p(y|x) = \sum \log p(y_t|y_{<t}, \boldsymbol{h^e})$$

$$h_t = f(h_{t-1}, y_{t-1}, \boldsymbol{h^e})$$

$$p(y_t|y_{<t}, \boldsymbol{h^e}) = \text{softmax}(g(h_t))$$

Algorithm 2 Sequence to Sequence training loop

procedure Training Loop
- $h^e = \text{encoder}(x)$
- $h_0 = h^e_l$
- $in_0 = \text{SOS}$
- $l = 0$
- **for** each t from 1 to T **do**
  - $h_t, out_t = \text{decorder}(h_{t-1}, in_{t-1})$
  - $l = l + \text{loss}(out_t, y_t)$
  - $in_t = out_t$
- **end for**

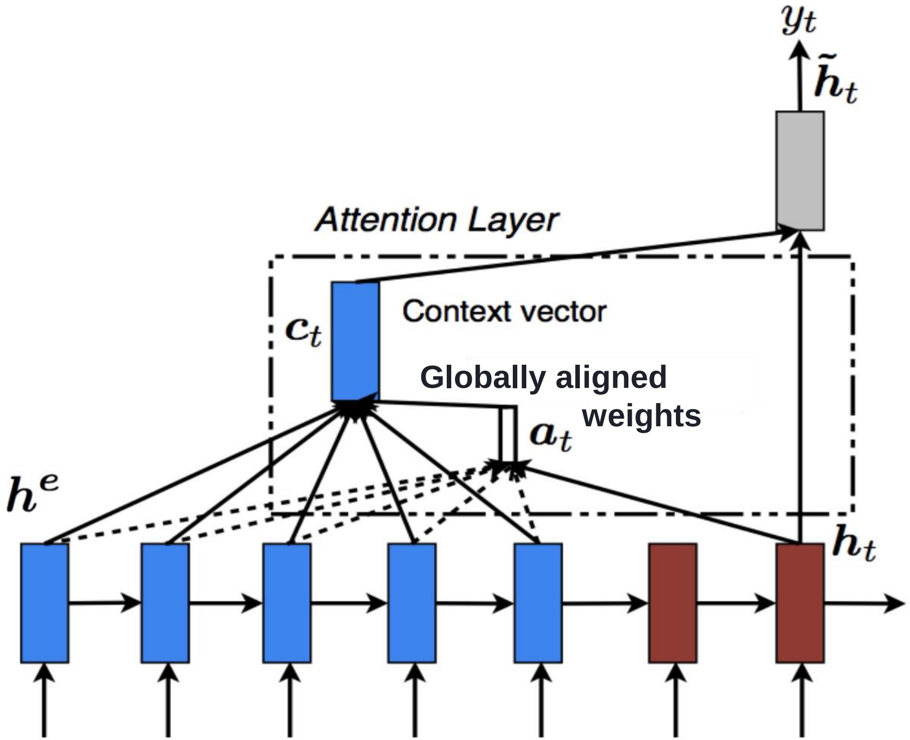
end procedure

```
loss = 0
hidden = encoder(x)
input = SOS

for label in labels:
    hidden, output = model(hidden, input)
    loss += criterion(output, label)
    input = one_hot(output.max(1))

loss.backward()
optimizer.step()
```

Summary: Attention (WIP)



Effective Approaches to Attention-based Neural Machine Translation (emnlp15)

$$\text{Modeling } p(y|x)$$
$$\boldsymbol{h^e} = \text{encoder}(x)$$
$$\log p(y|x) = \sum \log p(y_t|y_{<t}, \boldsymbol{h^e})$$

$$h_t = f(h_{t-1}, y_{t-1}, \boldsymbol{h^e})$$
$$\tilde{\boldsymbol{h}}_{\boldsymbol{t}} = tanh(\boldsymbol{W_c}[\mathbf{c}_t; \mathbf{h}_t])$$
$$p(y_t|y_{<t}, \boldsymbol{h^e}) = \text{softmax}(g(\tilde{\boldsymbol{h}}_{\boldsymbol{t}}))$$

$$\mathbf{c}_t = \boldsymbol{h^e a_t}$$
$$\boldsymbol{a_t} = \text{aligh}(\boldsymbol{h_t}, \boldsymbol{h^e}) = \frac{\exp(\text{score}(\boldsymbol{h_t}, \boldsymbol{h^e}))}{\sum_i \exp(score(\boldsymbol{h_t}, \boldsymbol{h^e_i}))}$$
$$\text{score}(\boldsymbol{h_t}, \boldsymbol{h^e}) = \boldsymbol{h_t^\intercal W_a h^e}$$

Algorithm 3 Attention training loop

```
procedure TRAINING LOOP
    h^e = encoder(x)
    h_0 = h^e_l
    in_0 = SOS
    l = 0
    for each t from 1 to T do
        s_t = h_t^T W_a h^e
        a_t = softmax(s_t)
        c_t = h^e a_t
        h_t, out_t = decorder(h_{t-1}, in_{t-1}, c_t)
        l = l + loss(out_t, y_t)
        in_t = out_t
    end for
end procedure
```

References

- Sequence to Sequence
  - Sequence to Sequence models: https://github.com/MaximumEntropy/Seq2Seq-PyTorch
  - Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation: https://arxiv.org/abs/1406.1078
- Attention Models
  - Attention and Augmented Recurrent Neural Networks https://distill.pub/2016/augmented-rnns/
  - Neural Machine Translation by Jointly Learning to Align and Translate: https://arxiv.org/abs/1409.0473
  - Effective Approaches to Attention-based Neural Machine Translation: https://arxiv.org/abs/1508.04025

Exercise 13-1

Implement Neural Machine Translation by Jointly Learning to Align and Translate: https://arxiv.org/abs/1409.0473

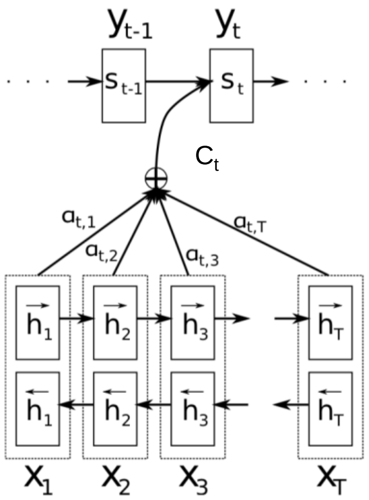


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

$$e_{ij} = a(s_{i-1}, h_j)$$

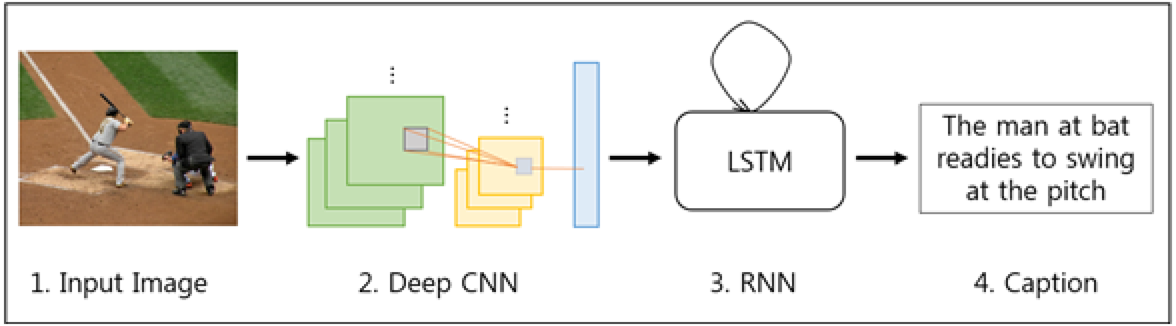
alignment model a as a feedforward neural network

https://arxiv.org/abs/1409.0473

Exercise 13-2

Implement A Neural Image Caption Generator:

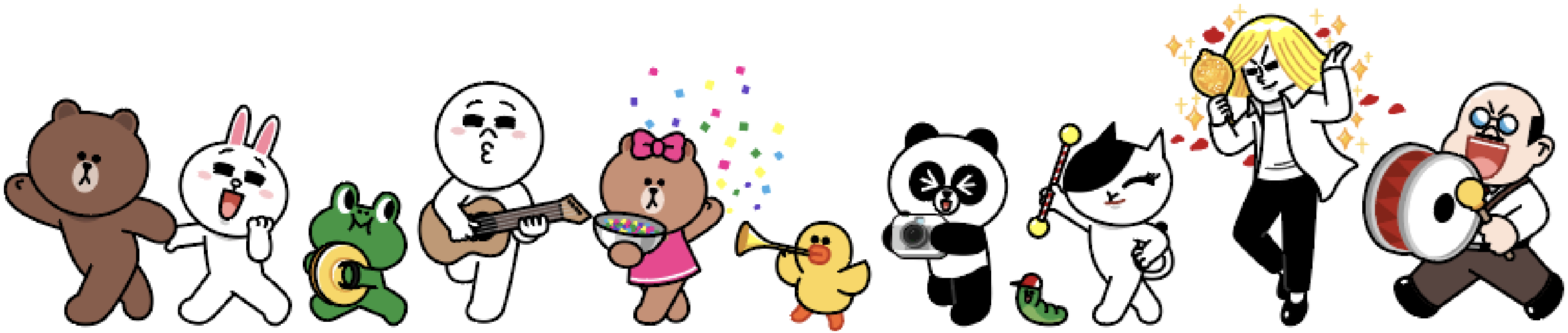
https://arxiv.org/abs/1411.4555



http://brain.kaist.ac.kr/research.html

If you've got this far, you did Good job! Congratulations!!

Interested in DL/ML related PHD, Postdoc at HKUST
and/or internship, residency, research fellows at LINE/NAVER?
Please email your exercises (Lectures 10 to 13) and CV to
hunkim+jobs@gmail.com.



WHAT NEXT

Lecture 14:
NSML,
Smartest ML Platform