Lab3 : a LSTM Cell for Image Captioning
補充

Department of Computer Science, NCTU

TA Ziv(鍾嘉峻)

Useful Link

General

- Ask anything you want in this repo~~

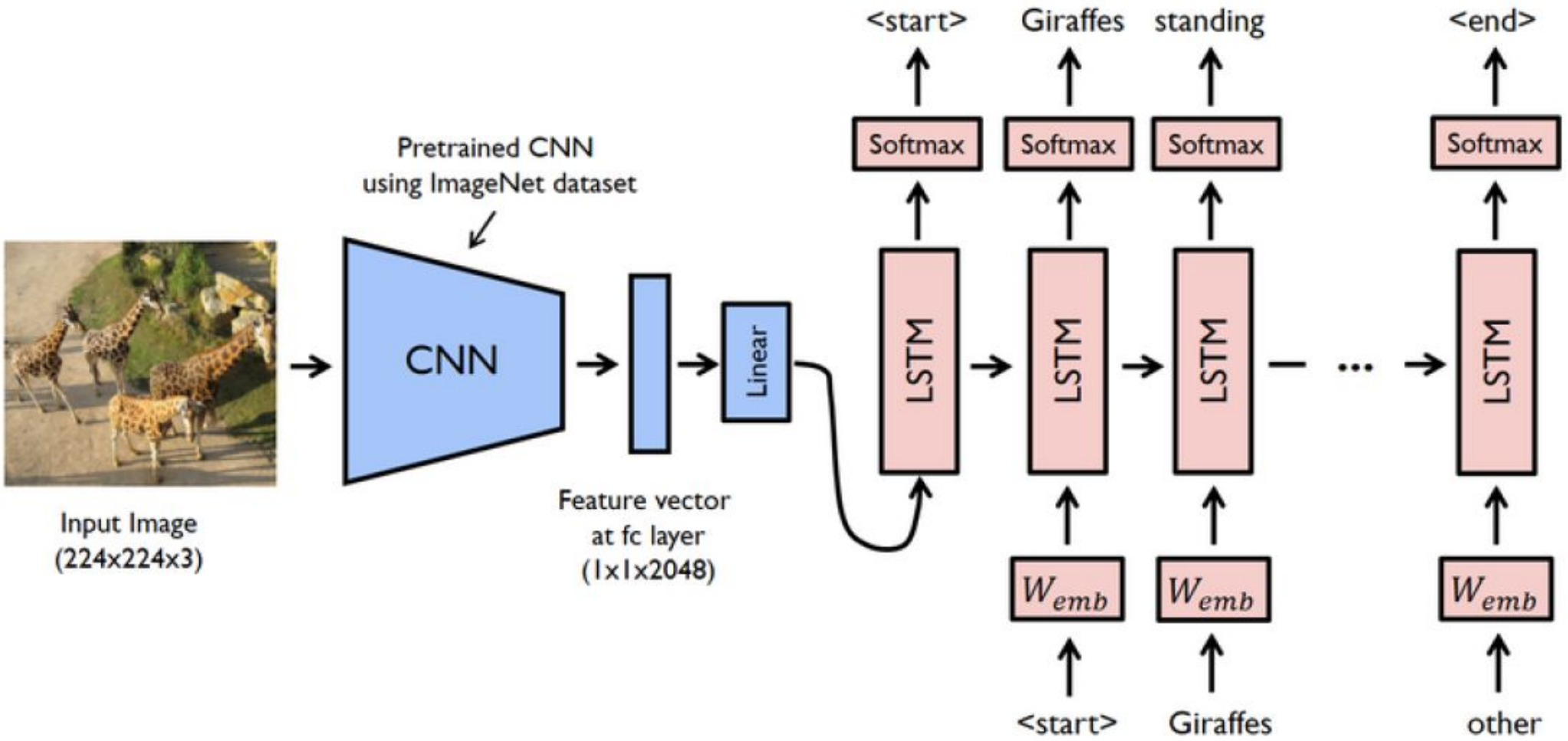
NCTU DL Final Project Demo

- 2018 Spring
- 2018 Summer
- 2019 Spring

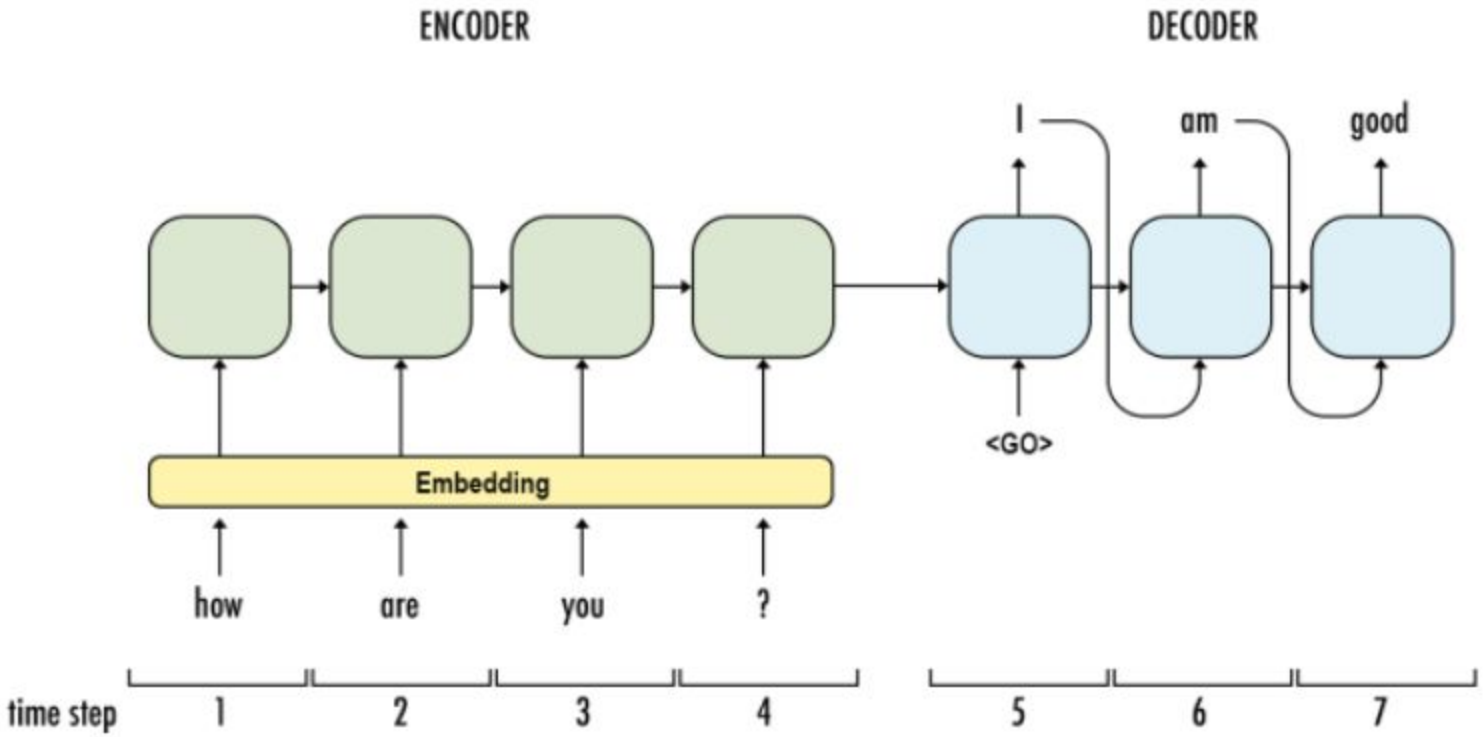
Useful Link

- awesome-AI-books
- 3D ML
- CNN training skill

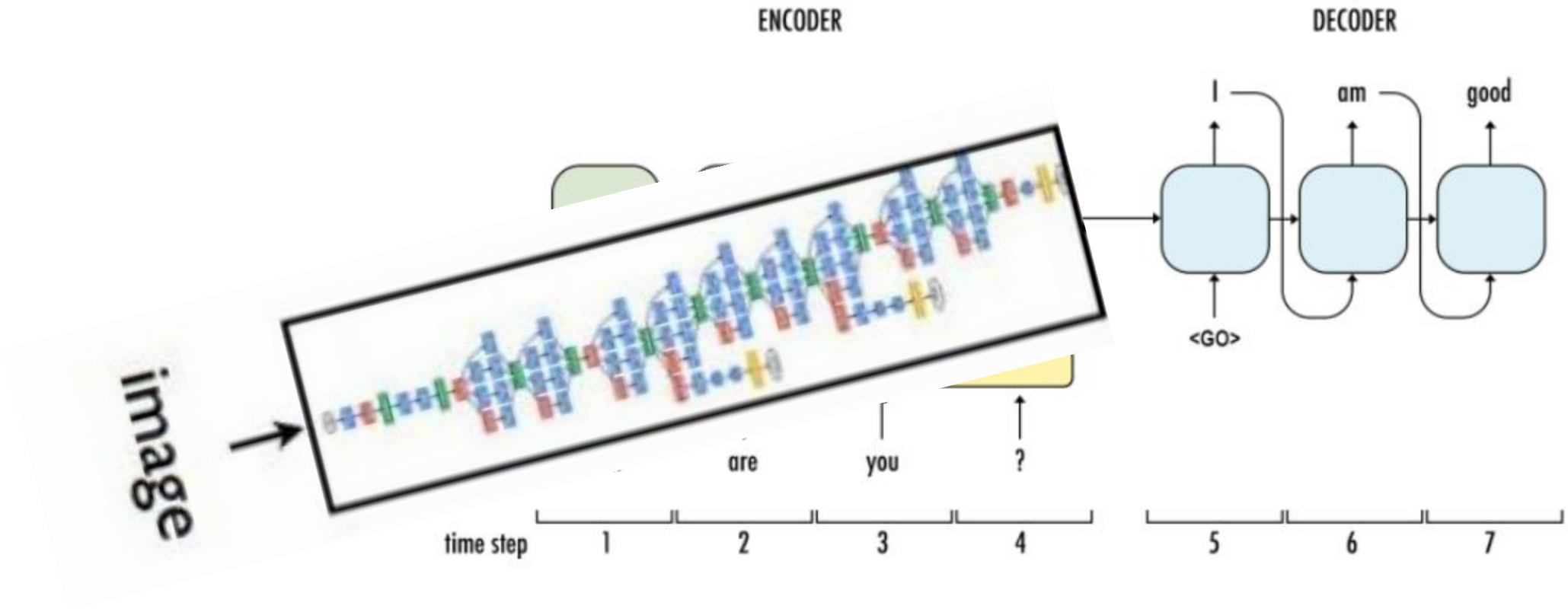
Image-Caption



Encoder-Decoder



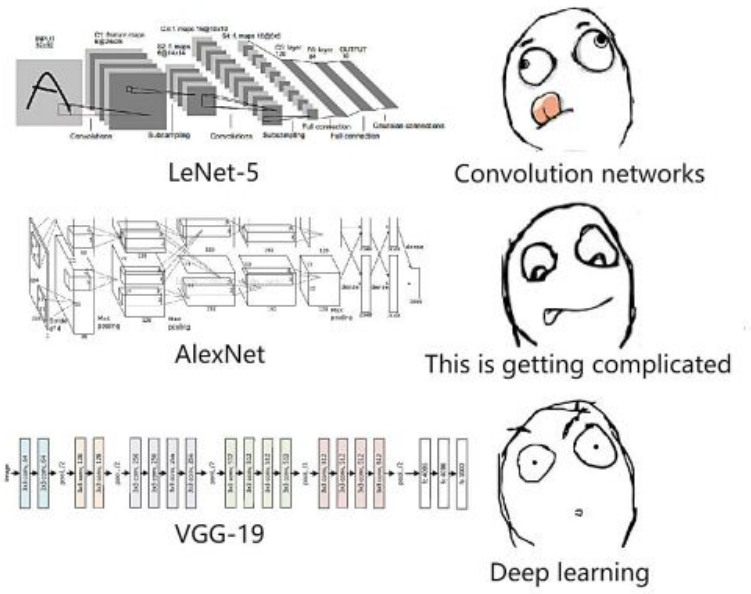
Encoder-Decoder

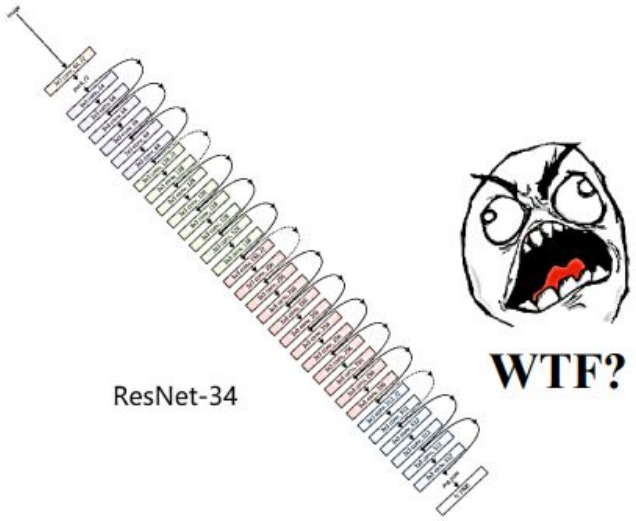


Show and Tell: A Neural Image Caption Generator

Encoder

- Using pretrain models to extract feature vector from a given input image
  - Using pretrained ResNet-152
  - From Torchvision





ResNet-152

Pretrained ResNet-152

- Pretrained on the ILSVRC-2012-CLS

```
import torchvision.models as models
resnet = models.resnet152(pretrained=True)
```

- Delete the last fc layer, use NEW linear layer to transform feature vector to have the same dimension as the input dimension of the LSTM network

```
self.linear = nn.Linear(resnet.fc.in_features, embed_size)
```

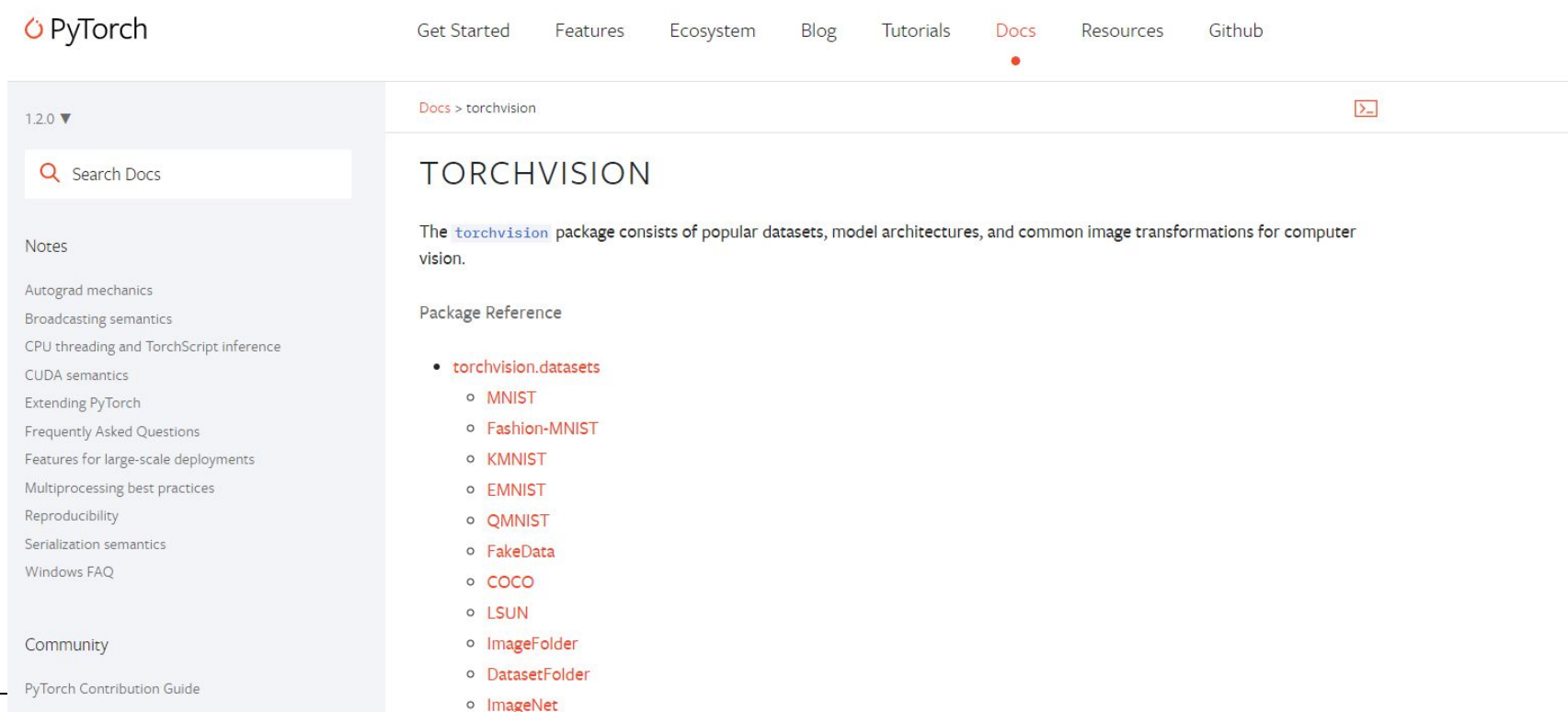
Parameters Update

- In train.py Line 45 ~ 46
  - ResNet part parameters won't update

```
43      # Loss and optimizer
44      criterion = nn.CrossEntropyLoss()
45      params = list(decoder.parameters()) + list(encoder.linear.parameters()) + list(encoder.bn.parameters())
46      optimizer = torch.optim.Adam(params, lr=args.learning_rate)
```

Torchvision

- Pytorch official package consists of
  - popular datasets
  - model architectures
  - common image transformations for computer vision.



Decoder

- Noticed that you will use your model in model.py line 34
  - You can use nn.LSTM to check your environment is OK or not

```
27  class DecoderRNN(nn.Module):
28          def __init__(self, embed_size, hidden_size, vocab_size, num_layers, max_seq_length=20):
29                  """Set the hyper-parameters and build the layers."""
30                  super(DecoderRNN, self).__init__()
31                  self.embed = nn.Embedding(vocab_size, embed_size)
32                  # uncomment this line to use the default setting
33                  #self.lstm = nn.LSTM(embed_size, hidden_size, num_layers, batch_first=True)
34                  self.lstm = my_LSTM(embed_size, hidden_size, num_layers, batch_first=True)
35                  self.linear = nn.Linear(hidden_size, vocab_size)
36                  self.max_seg_length = max_seq_length
37
```

LSTM Recall

- At professor slide “RecurrentNeuralNetworks.pdf”
  - Memory state: $\boldsymbol{s}^{(t)}$
  - Input gate: $\boldsymbol{g}^{(t)} = \sigma(\boldsymbol{U}^g \boldsymbol{x}^{(t)} + \boldsymbol{W}^g \boldsymbol{h}^{(t-1)})$
  - Output gate: $\boldsymbol{q}^{(t)} = \sigma(\boldsymbol{U}^o \boldsymbol{x}^{(t)} + \boldsymbol{W}^o \boldsymbol{h}^{(t-1)})$
  - Forget gate: $\boldsymbol{f}^{(t)} = \sigma(\boldsymbol{U}^f \boldsymbol{x}^{(t)} + \boldsymbol{W}^f \boldsymbol{h}^{(t-1)})$
  - New content: $\boldsymbol{a}^{(t)} = \boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{W}\boldsymbol{h}^{(t-1)}$
  - Memory update: $\boldsymbol{s}^{(t)} = \boldsymbol{f}^{(t)} \odot \boldsymbol{s}^{(t-1)} + \boldsymbol{g}^{(t)} \odot \tanh(\boldsymbol{a}^{(t)})$
  - Hidden unit update: $\boldsymbol{h}^{(t)} = \boldsymbol{q}^{(t)} \odot \tanh(\boldsymbol{s}^{(t)})$
  - Output unit update: $\boldsymbol{o}^{(t)} = \boldsymbol{V}\boldsymbol{h}^{(t)}$

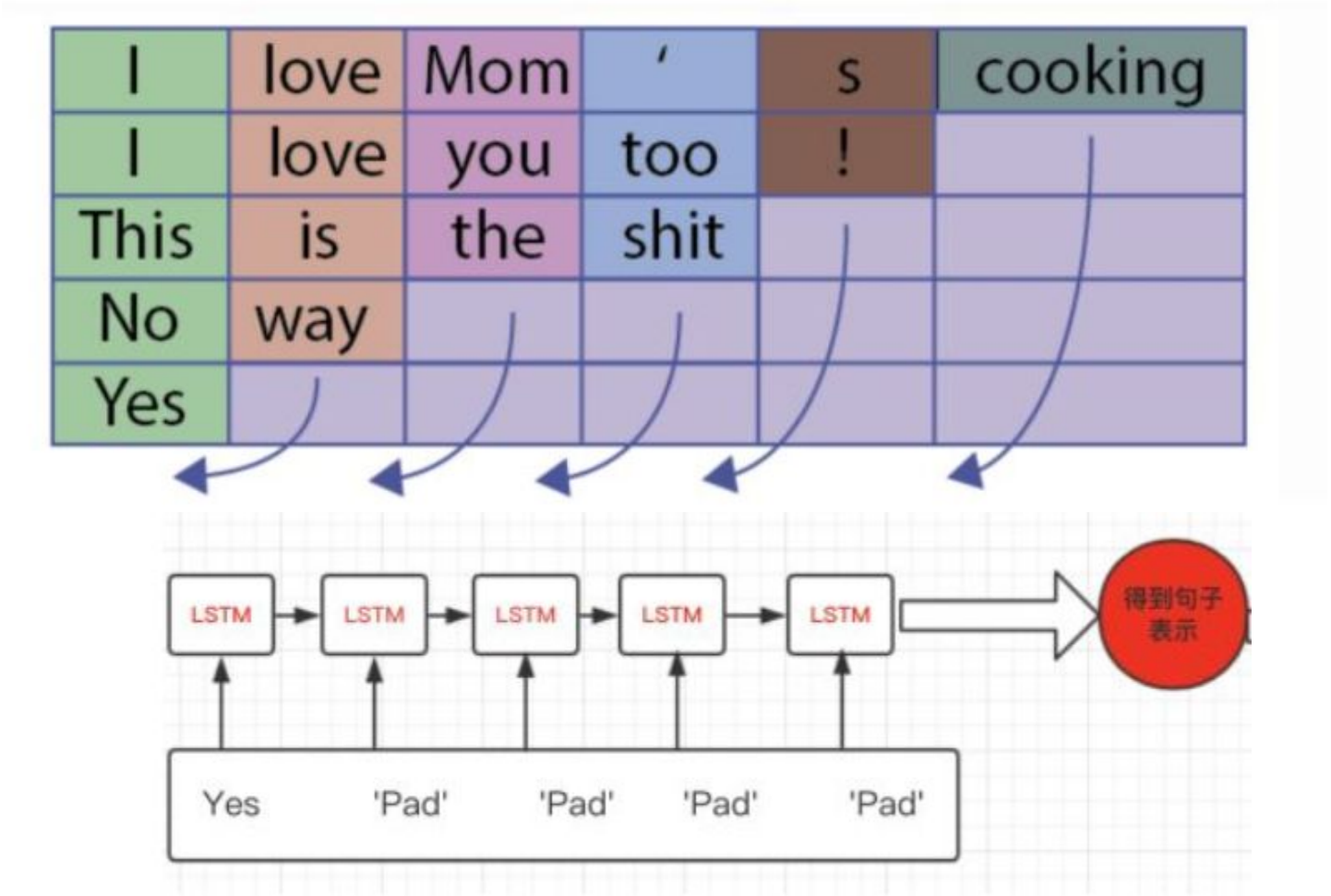
Lstm Implement Hint

- You can use nn.Linear to build your lstm

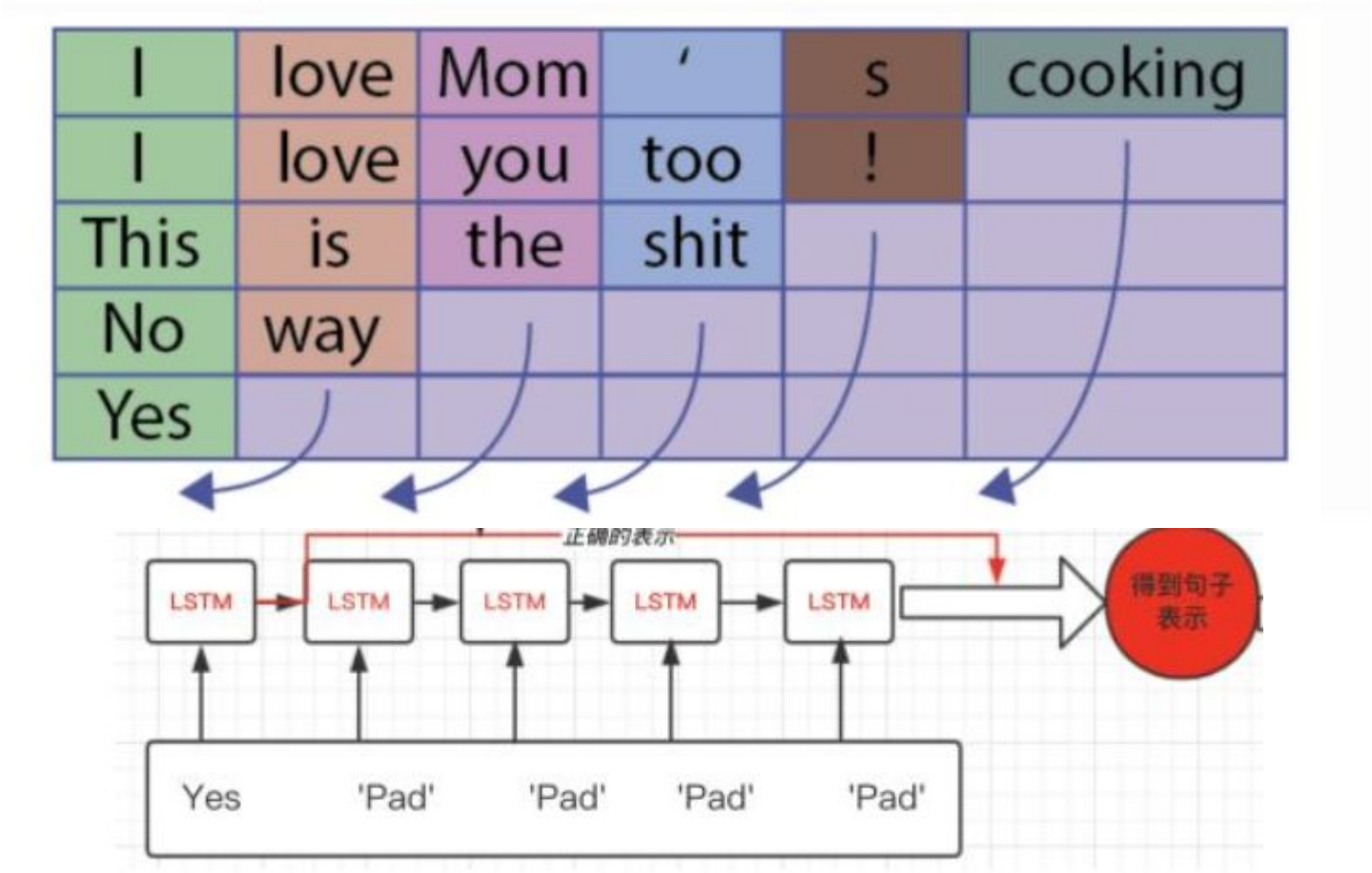
```
self.fc_ho = nn.Linear(hidden_size, hidden_size, bias=if_bias)
```

- RNN Example

pack_padded_sequence



pack_padded_sequence



pack_padded_sequence