

# 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

---

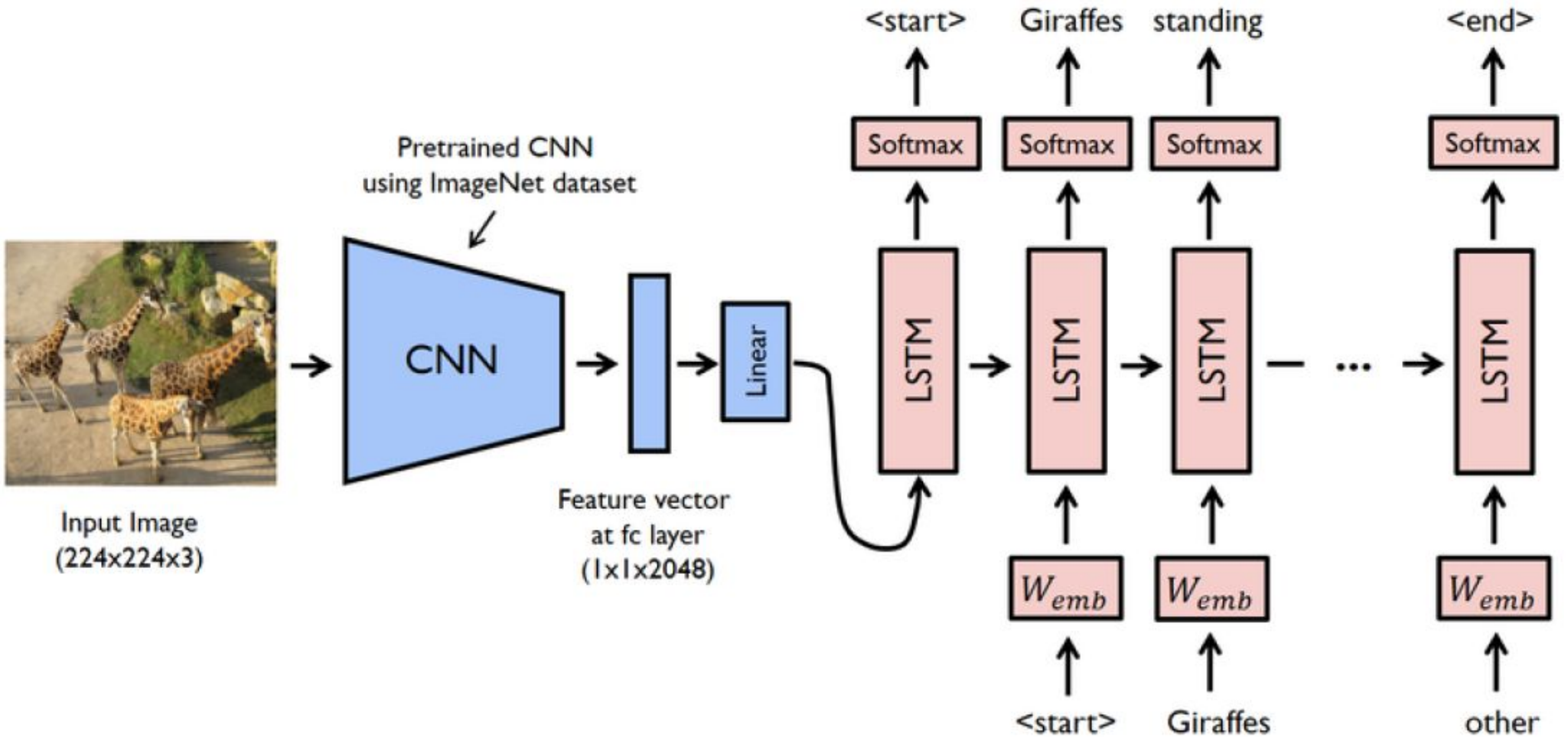
- [2018 Spring](#)
- [2018 Summer](#)
- [2019 Spring](#)

## Useful Link

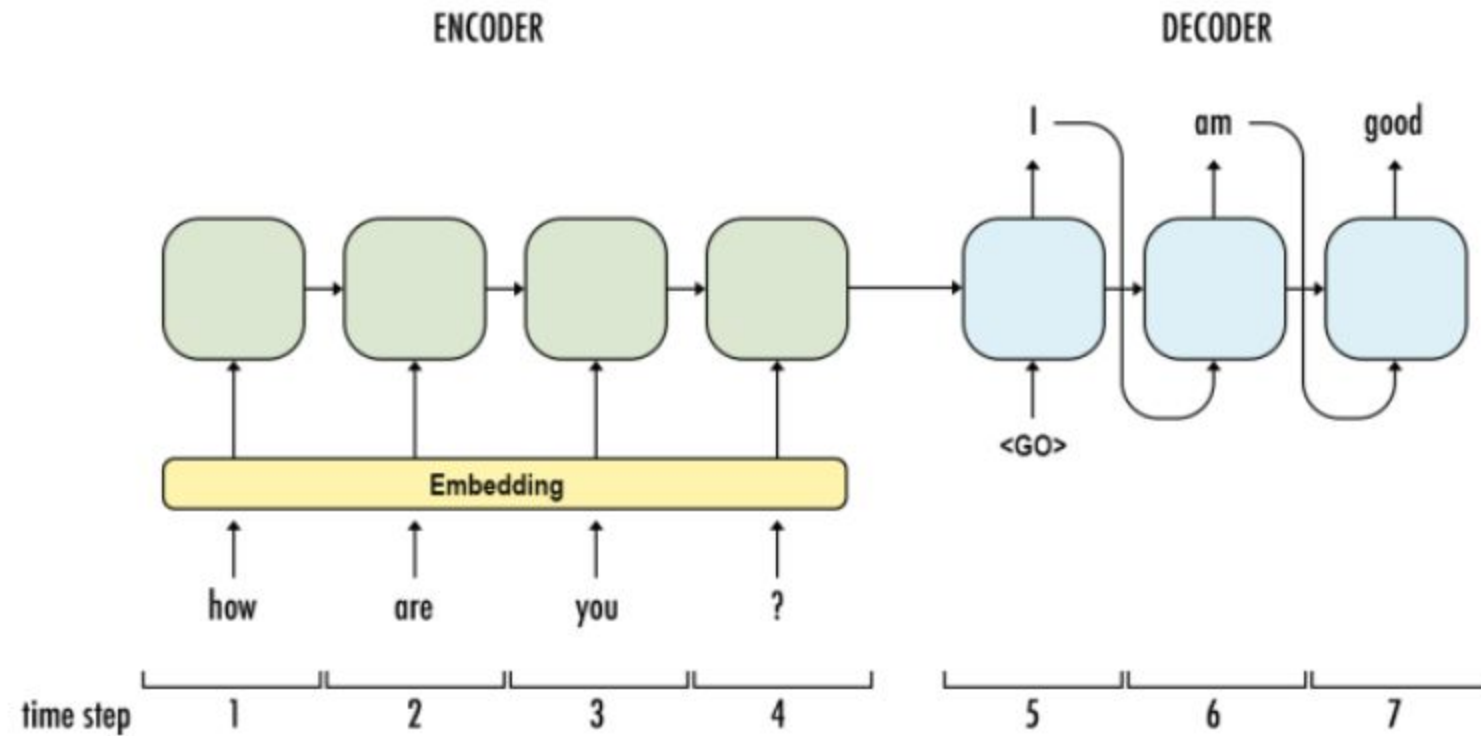
---

- [awesome-AI-books](#)
- [3D ML](#)
- [CNN training skill](#)

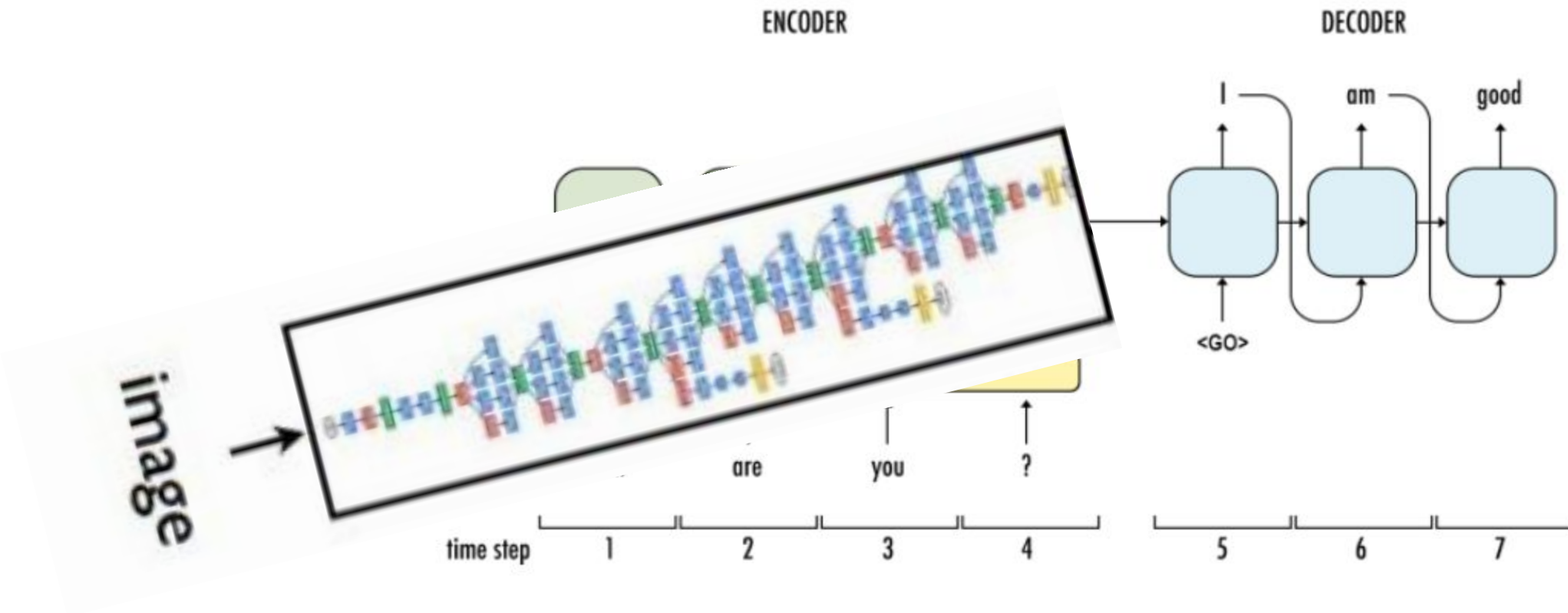
# Image-Caption



# Encoder-Decoder

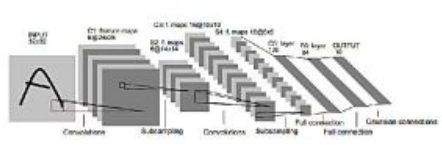


# Encoder-Decoder



# Encoder

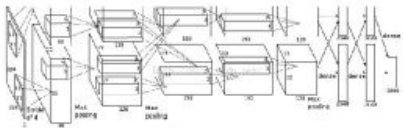
- Using pretrain models to extract feature vector from a given input image
  - Using pretrained ResNet-152
  - From Torchvision



LeNet-5



Convolution networks



AlexNet



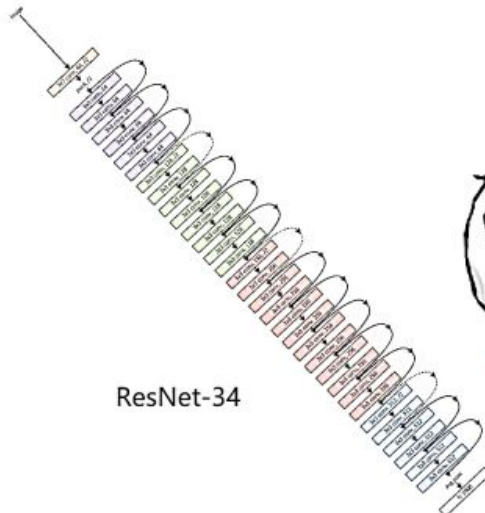
This is getting complicated



VGG-19



Deep learning



ResNet-34



WTF?

ResNet-152



WTF! ! !

# 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.

The screenshot shows the PyTorch torchvision documentation page. The top navigation bar includes links for 'Get Started', 'Features', 'Ecosystem', 'Blog', 'Tutorials', 'Docs' (which is highlighted with a red dot), 'Resources', and 'Github'. Below the navigation bar, the left sidebar shows the version '1.2.0' and a search bar labeled 'Search Docs'. The sidebar also lists various topics under 'Notes' and 'Community'. The main content area is titled 'TORCHVISION' and contains a description: 'The torchvision package consists of popular datasets, model architectures, and common image transformations for computer vision.' Below this, there is a 'Package Reference' section listing various datasets and models, including MNIST, Fashion-MNIST, KMNIST, EMNIST, QMNIST, FakeData, COCO, LSUN, ImageFolder, DatasetFolder, and ImageNet.

PyTorch

Get Started Features Ecosystem Blog Tutorials Docs Resources Github

1.2.0 ▼

Search Docs

Notes

- Autograd mechanics
- Broadcasting semantics
- CPU threading and TorchScript inference
- CUDA semantics
- Extending PyTorch
- Frequently Asked Questions
- Features for large-scale deployments
- Multiprocessing best practices
- Reproducibility
- Serialization semantics
- Windows FAQ

Community

PyTorch Contribution Guide

Docs > torchvision

## TORCHVISION

The `torchvision` package consists of popular datasets, model architectures, and common image transformations for computer vision.

### Package Reference

- `torchvision.datasets`
  - MNIST
  - Fashion-MNIST
  - KMNIST
  - EMNIST
  - QMNIST
  - FakeData
  - COCO
  - LSUN
  - ImageFolder
  - DatasetFolder
  - ImageNet

# 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:  $\mathbf{s}^{(t)}$
- Input gate:  $\mathbf{g}^{(t)} = \sigma(\mathbf{U}^g \mathbf{x}^{(t)} + \mathbf{W}^g \mathbf{h}^{(t-1)})$
- Output gate:  $\mathbf{q}^{(t)} = \sigma(\mathbf{U}^o \mathbf{x}^{(t)} + \mathbf{W}^o \mathbf{h}^{(t-1)})$
- Forget gate:  $\mathbf{f}^{(t)} = \sigma(\mathbf{U}^f \mathbf{x}^{(t)} + \mathbf{W}^f \mathbf{h}^{(t-1)})$
- New content:  $\mathbf{a}^{(t)} = \mathbf{U} \mathbf{x}^{(t)} + \mathbf{W} \mathbf{h}^{(t-1)}$
- Memory update:  $\mathbf{s}^{(t)} = \mathbf{f}^{(t)} \odot \mathbf{s}^{(t-1)} + \mathbf{g}^{(t)} \odot \tanh(\mathbf{a}^{(t)})$
- Hidden unit update:  $\mathbf{h}^{(t)} = \mathbf{q}^{(t)} \odot \tanh(\mathbf{s}^{(t)})$
- Output unit update:  $\mathbf{o}^{(t)} = \mathbf{V} \mathbf{h}^{(t)}$

# Lstm Implement Hint

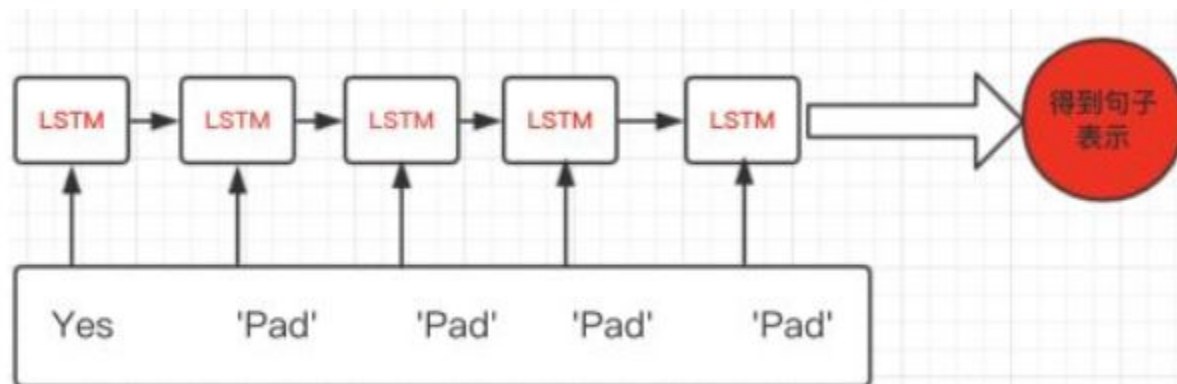
- You can use nn.Linear to build your lstm

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

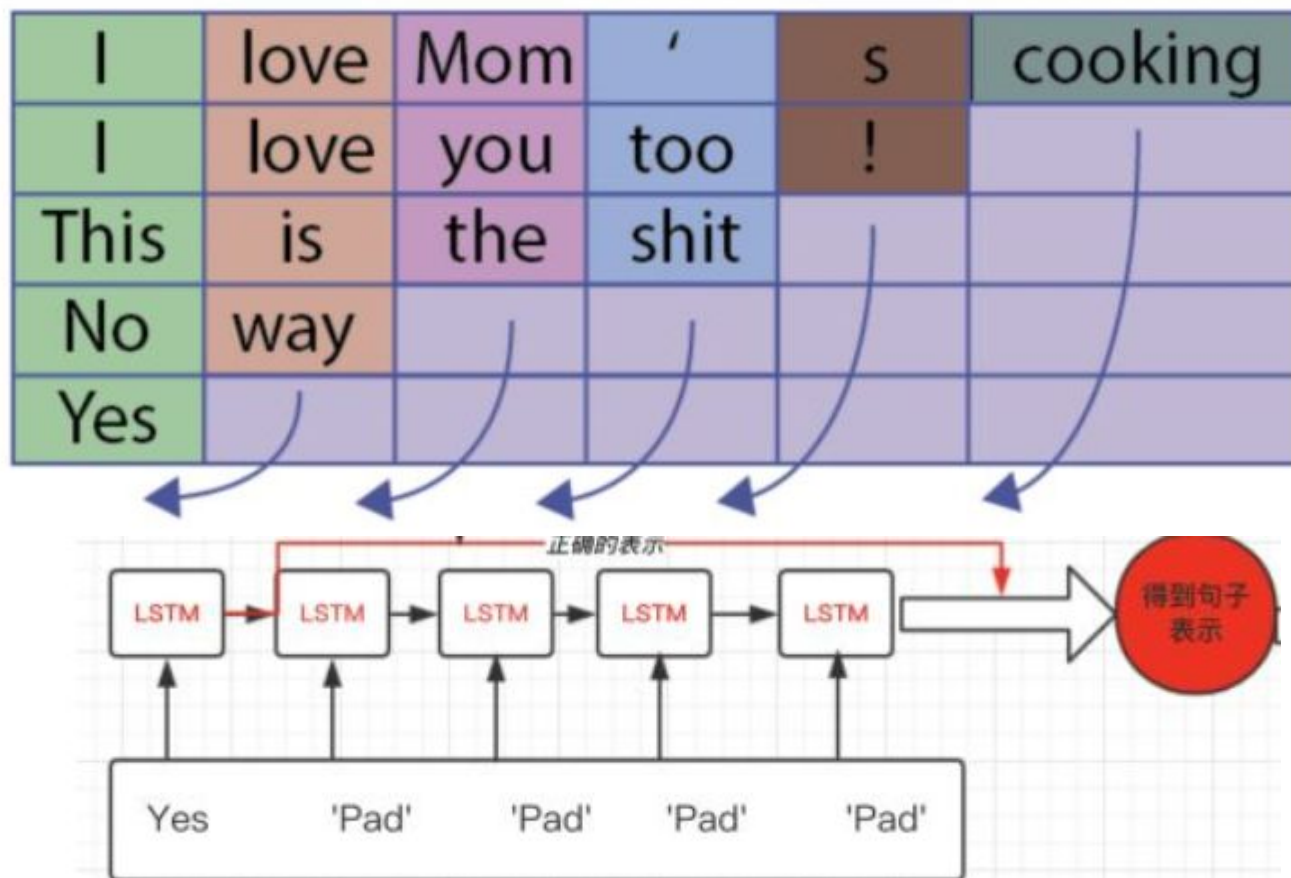
- [RNN Example](#)

# pack\_padded\_sequence

I	love	Mom	'	s	cooking
I	love	you	too	!	
This	is	the	shit		
No	way				
Yes					



# pack\_padded\_sequence



# pack\_padded\_sequence

I	love	Mom	'	s	cooking
I	love	you	too	!	
This	is	the	shit		
No	way				
Yes					

**data**



**batch\_sizes** [ 5, 4, 3, 3, 2, 1 ]