AI VIETNAM
All-in-One Course

Deep Architectures for POS Tagging and NER

Quang-Vinh Dinh
Ph.D. in Computer Science

Year 2024

Outline

- Quick Review
- POS Tagging Using Different Models
- Named Entity Recognition
- Step-by-step Examples
- PyTorch Implementation

Quiz 1

❖ Choose the correct code segment?

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class MyModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(5, 4)
        self.fc2 = nn.Linear(4, 3)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x

model = MyModel()
```

```
from torchinfo import  summary

input_x = torch.randn((32, 5))
model = MyModel()

summary(model, input_data=input_x)
```

```
from torchinfo import  summary

input_x = torch.randn((32, 8, 5))
model = MyModel()

summary(model, input_data=input_x)
```

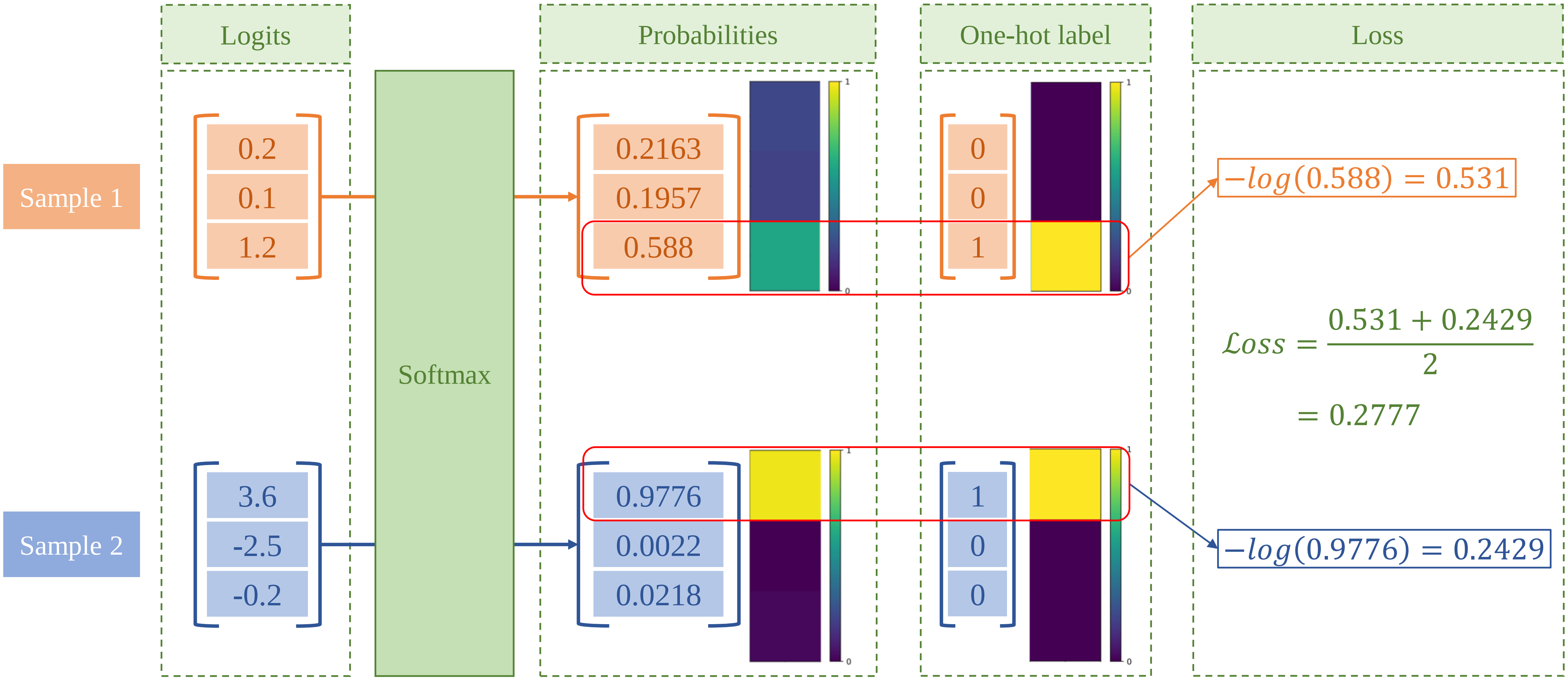
```
==========================================================
Layer (type:depth-idx)                      Output Shape
==========================================================
MyModel                                     [32, 3]
├─Linear: 1-1                               [32, 4]
├─Linear: 1-2                               [32, 3]
==========================================================
Total params: 39
```

```
==========================================================
Layer (type:depth-idx)                      Output Shape
==========================================================
MyModel                                     [32, 8, 3]
├─Linear: 1-1                               [32, 8, 4]
├─Linear: 1-2                               [32, 8, 3]
==========================================================
Total params: 39
```

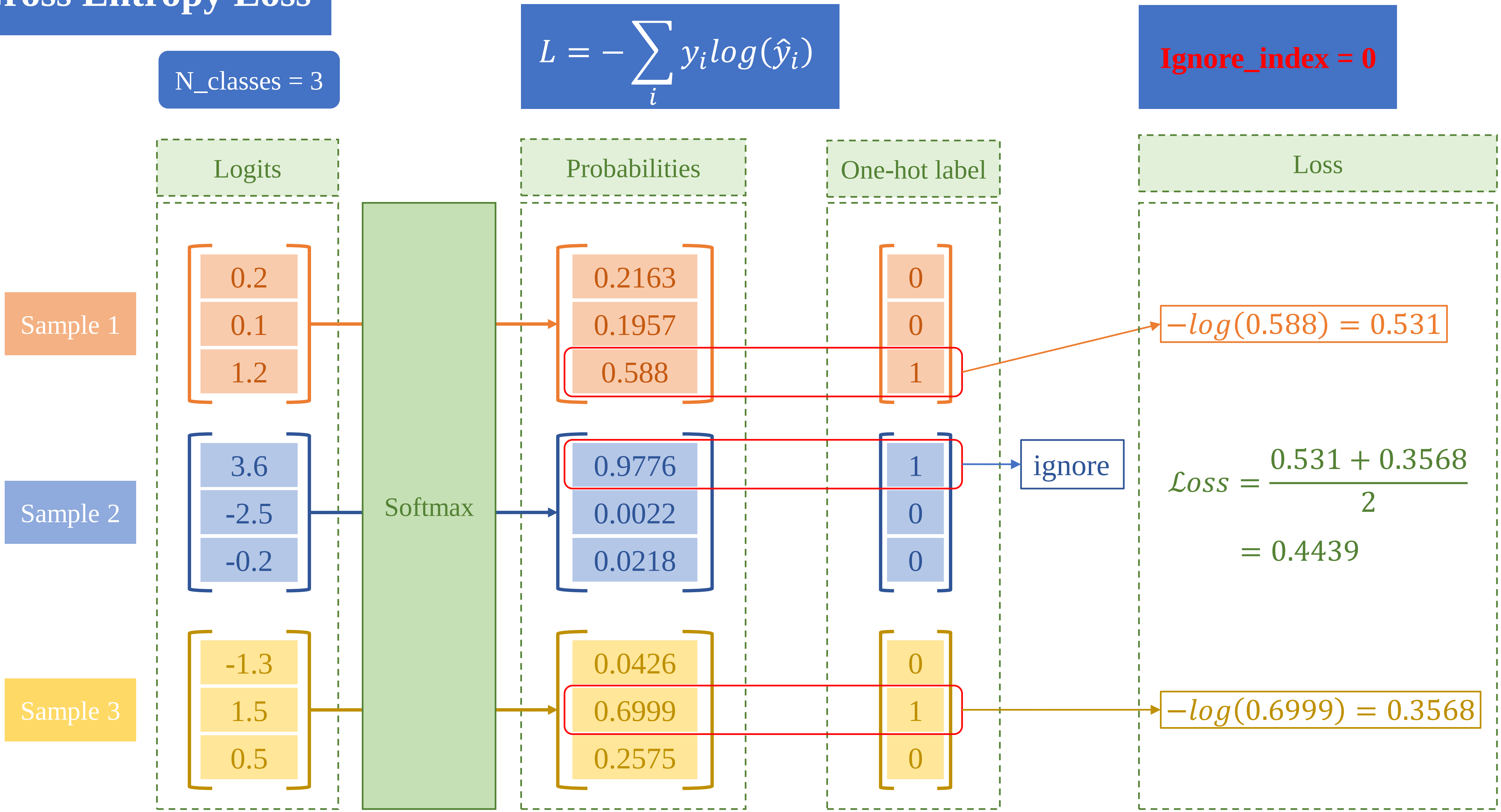
Cross Entropy Loss

N_classes = 3

$$L = -\sum_i y_i log(\hat{y}_i)$$

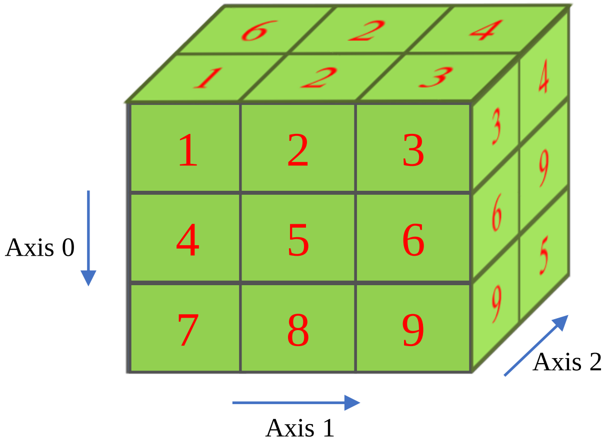


Cross Entropy Loss

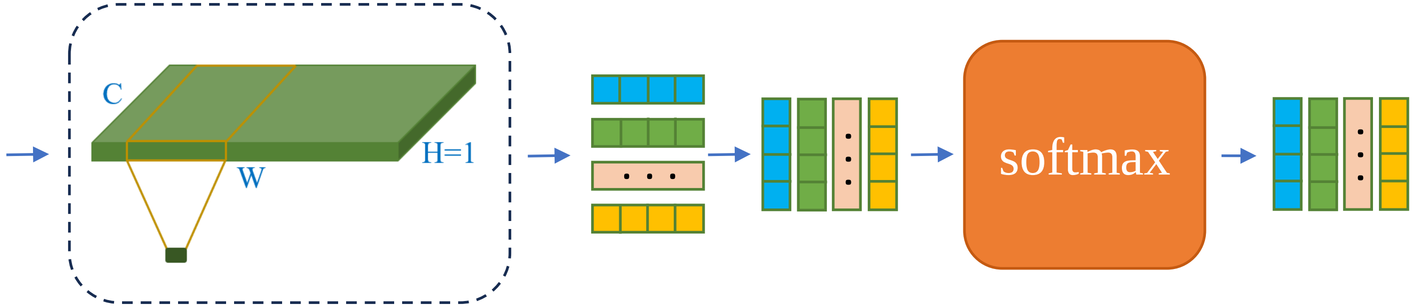


Quiz 2

❖ Loss function



Three dimensions includes
- batch_size N
- sequence_length L
- num_classes C



PyTorch

```python
criterion = nn.CrossEntropyLoss()
loss = criterion(Z, y)
```

if the y shape is (N,)

the Z shape is (?)

if the y shape is (N, C)

the Z shape is (?)

if the Z shape is (N, L, C)

the y shape is (?)

Outline

- Quick Review
- POS Tagging Using Different Models
- Named Entity Recognition
- Step-by-step Examples
- PyTorch Implementation

Designing a Model for POS Tagging

Doc	Label
i want a dog	[0, 1, 2, 0]
books are expensive	[0, 1, 2]

Label	Meaning
0	Noun/Pronoun
1	Verb
2	Others

building dictionary

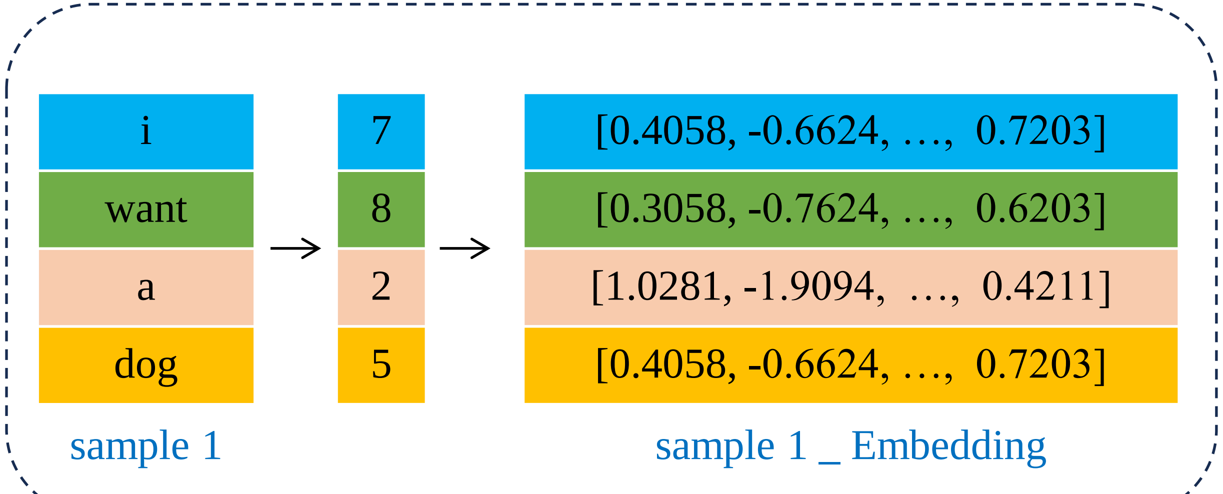
vocab size = 9
sequence length = 4

index	word
0	[UNK]
1	[pad]
2	a
3	are
4	books
5	dog
6	expensive
7	i
8	want

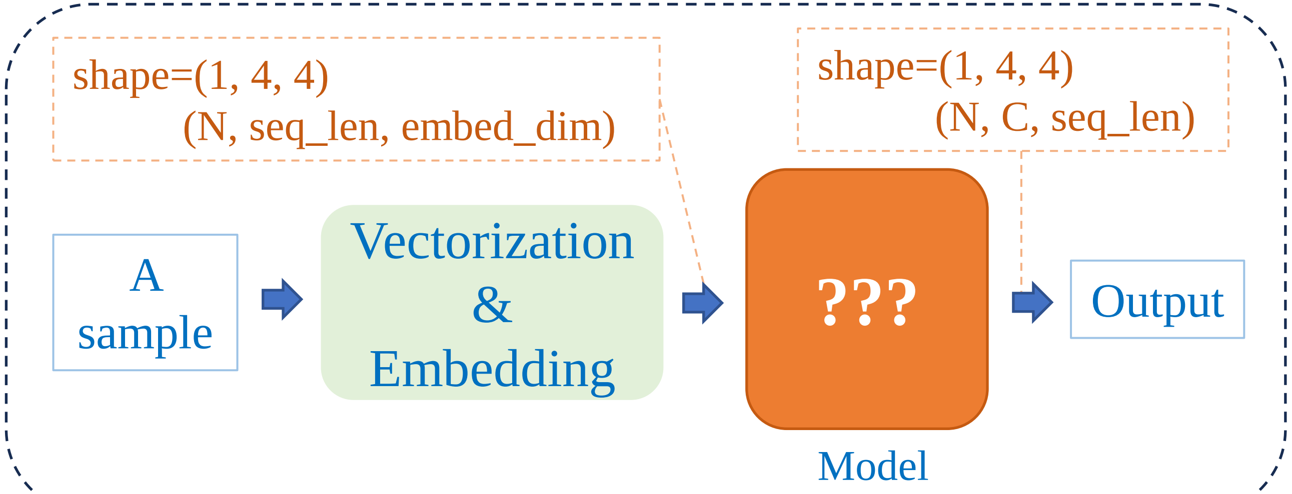
Dictionary

0	[-0.1882, 0.5530, …, 0.7013]
1	[1.7840, -0.8278, …, 1.3586]
2	[1.0281, -1.9094, …, 0.4211]
3	[-1.3083, -0.0987, …, -0.3680]
4	[0.2293, 1.3255, …, 2.0501]
5	[0.4058, -0.6624, …, 0.7203]
6	[0.5582, 0.0786, …, 0.6902]
7	[0.4309, -1.3067, …, 1.5977]
8	[0.3058, -0.7624, …, 0.6203]

Embedding

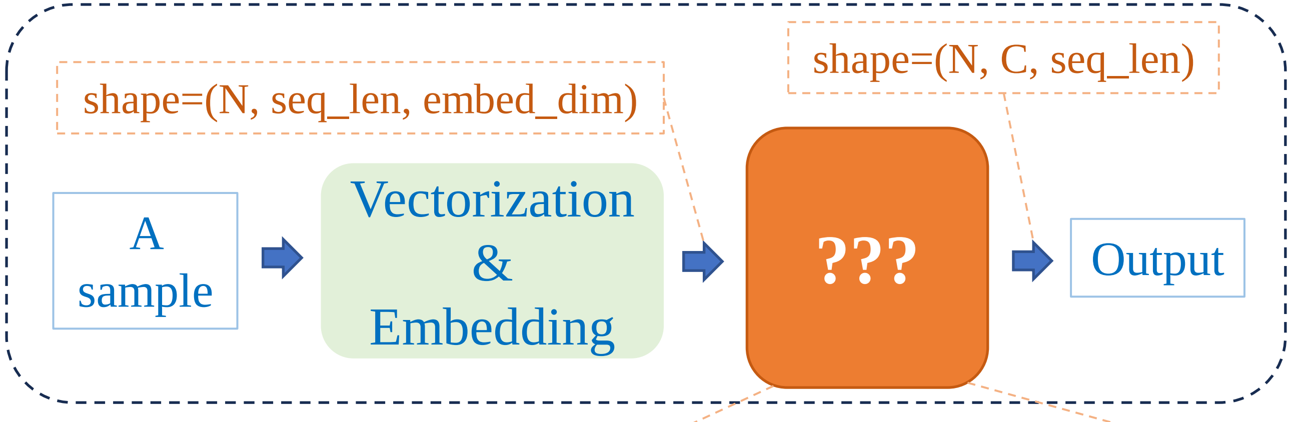


Vectorization and Embedding



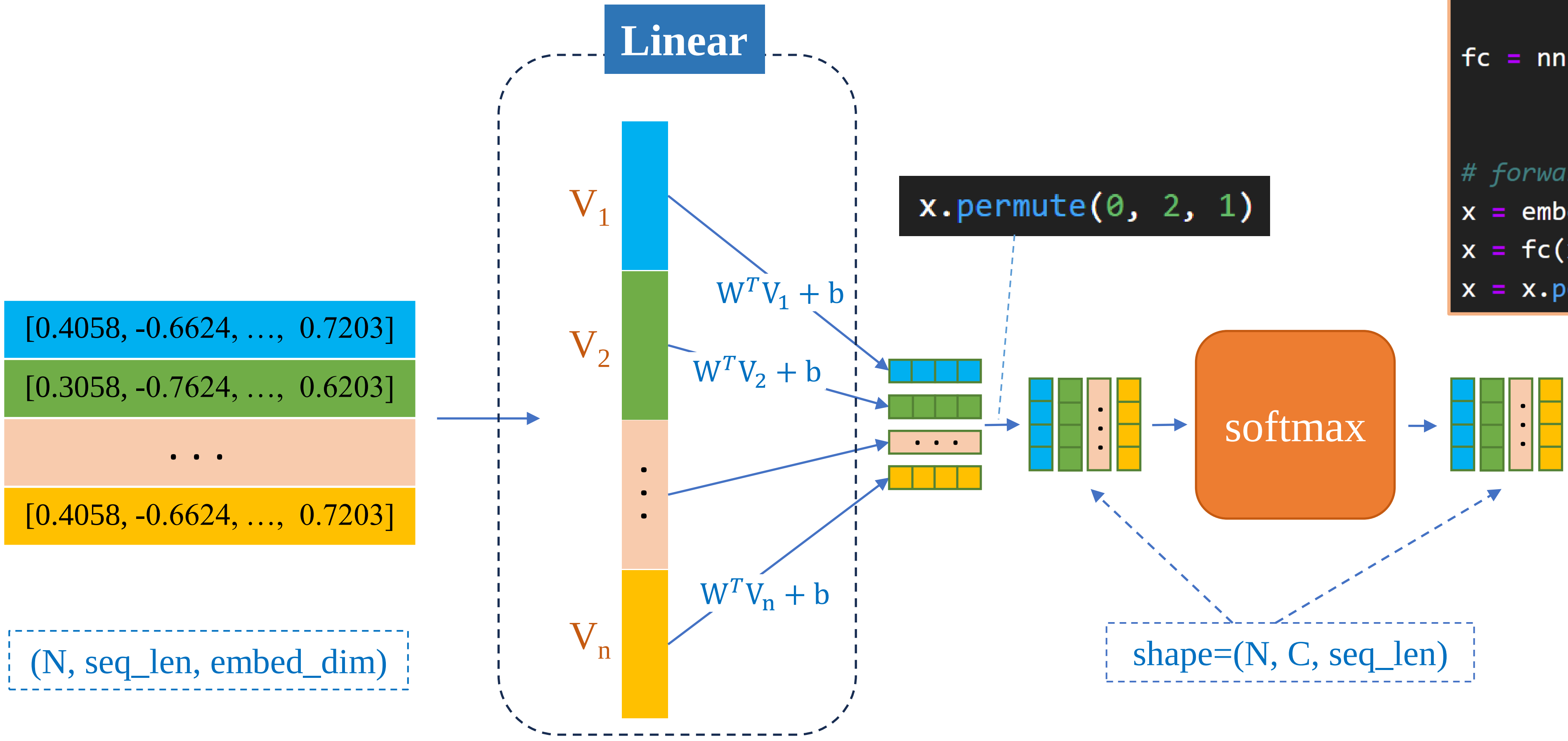
Model Pipeline

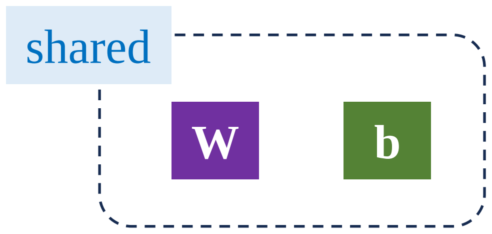
Designing a Model for POS Tagging



Using MLP

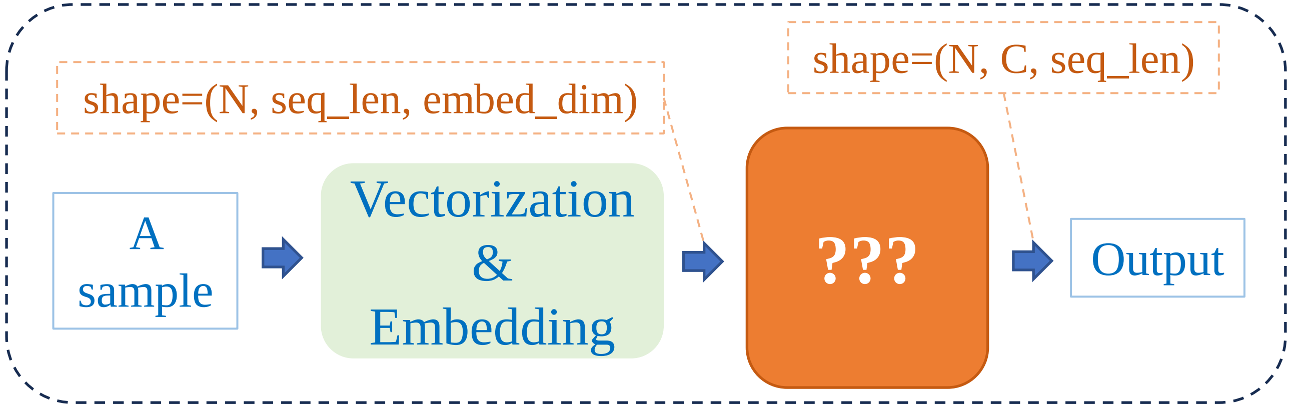
```
embedding = nn.Embedding(vocab_size,
                         embed_dim)
fc = nn.Linear(embed_dim,
               num_classes)

# forward
x = embedding(x)
x = fc(x)
x = x.permute(0, 2, 1)
```





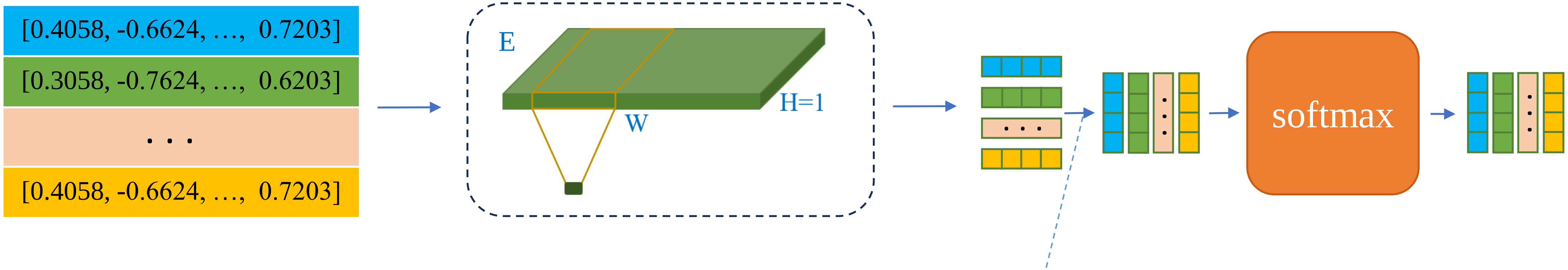
Designing a Model for POS Tagging

Using CNN



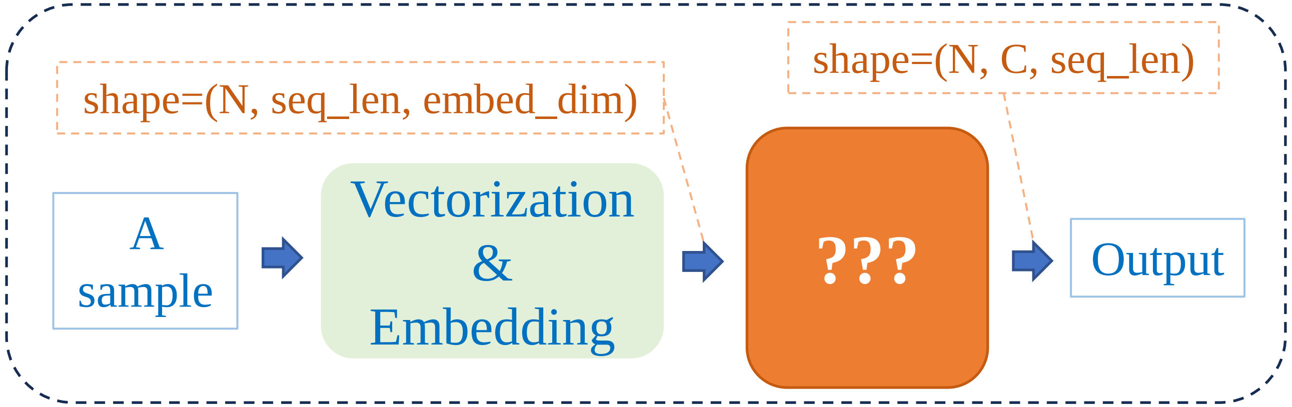
This pipeline is wrong.
Let's find out!

```
embedding = nn.Embedding(vocab_size, 3)
conv1d = nn.Conv1d(3, num_classes,
                   kernel_size=2, padding='same')
# forward
x = self.embedding(x)
x = self.conv1d(x)
x = x.permute(0, 2, 1)
```




```
x.permute(0, 2, 1)
```

Designing a Model for POS Tagging



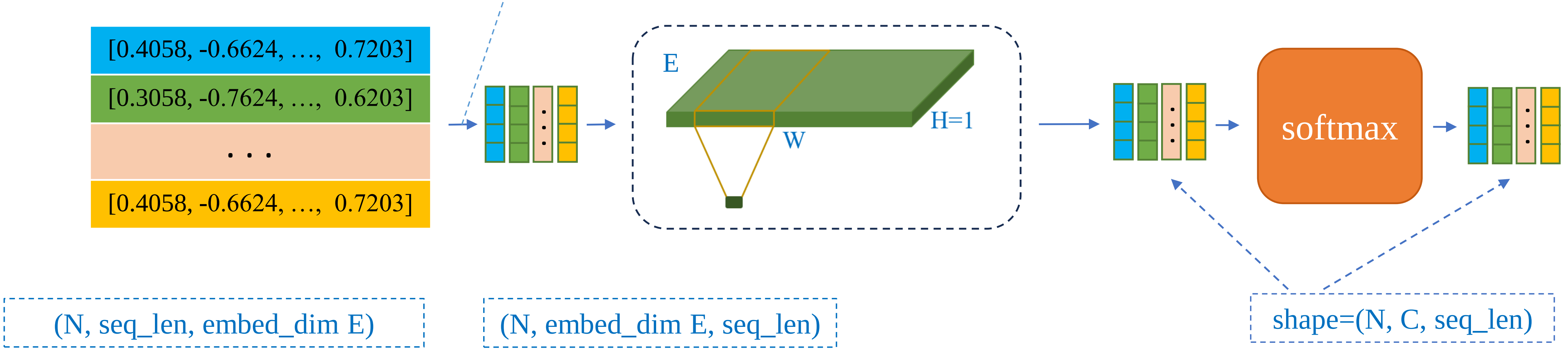
Using CNN

input size (N, C_{in}, L) and output $(N, C_{\text{out}}, L_{\text{out}})$

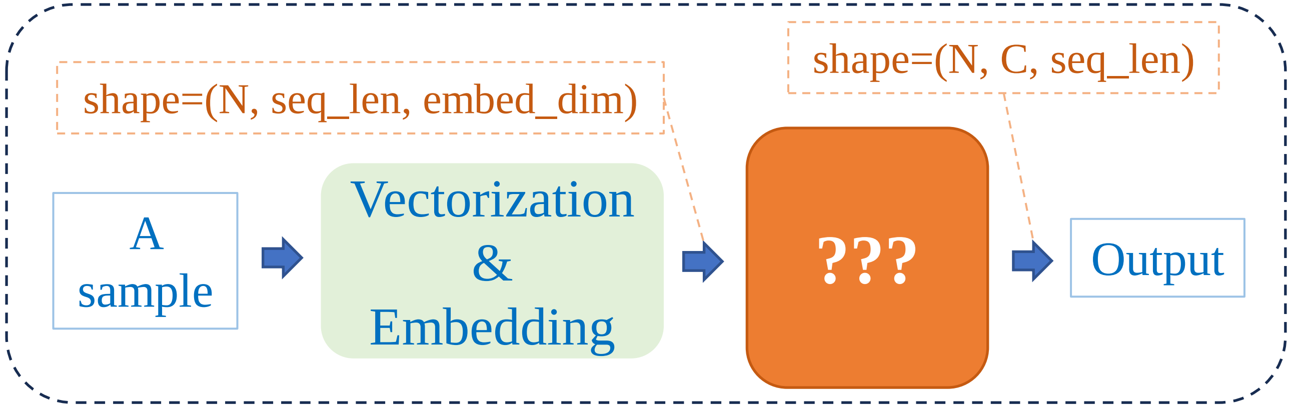
https://pytorch.org/docs/stable/generated/torch.nn.Conv1d.html

```
embedding = nn.Embedding(vocab_size, 3)
conv1d = nn.Conv1d(3, num_classes,
                   kernel_size=2, padding='same')
# forward
x = self.embedding(x)
x = x.permute(0, 2, 1)
x = self.conv1d(x)
```

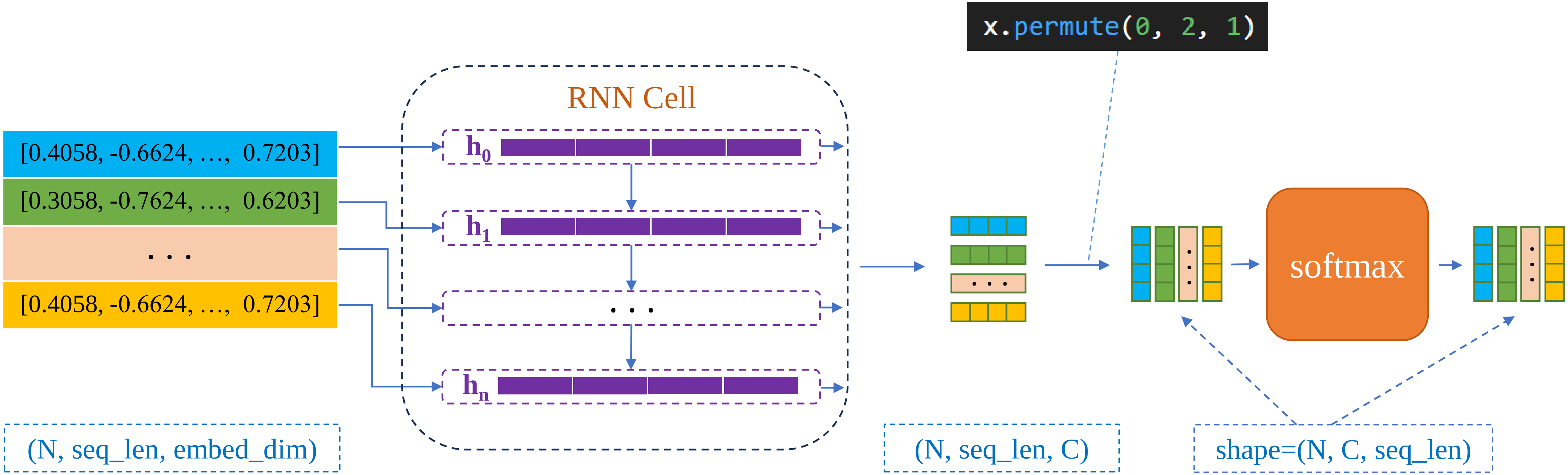
```
x.permute(0, 2, 1)
```



Designing a Model for POS Tagging



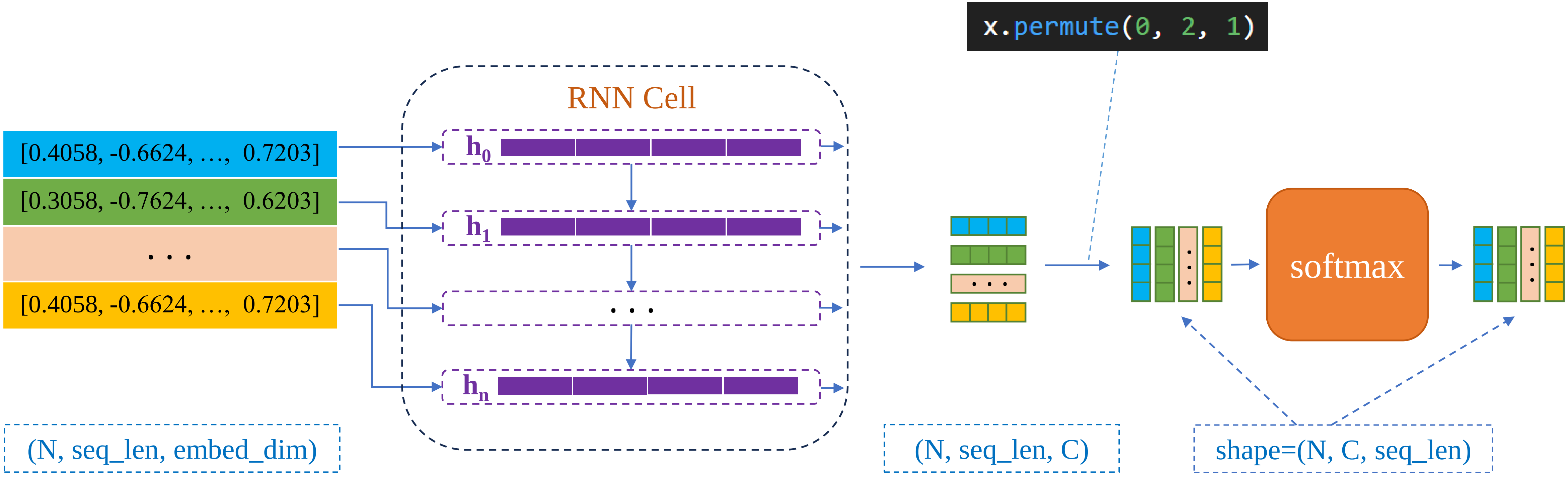
Using RNN



Designing a Model for POS Tagging

Using RNN: Implementation

```
embedding = nn.Embedding(vocab_size, emb_dim)
recurrent = nn.RNN(emb_dim, num_classes, batch_first=True)

# forward
x = embedding(x)
output, _ = recurrent(x)
x = output.permute(0, 2, 1)
```

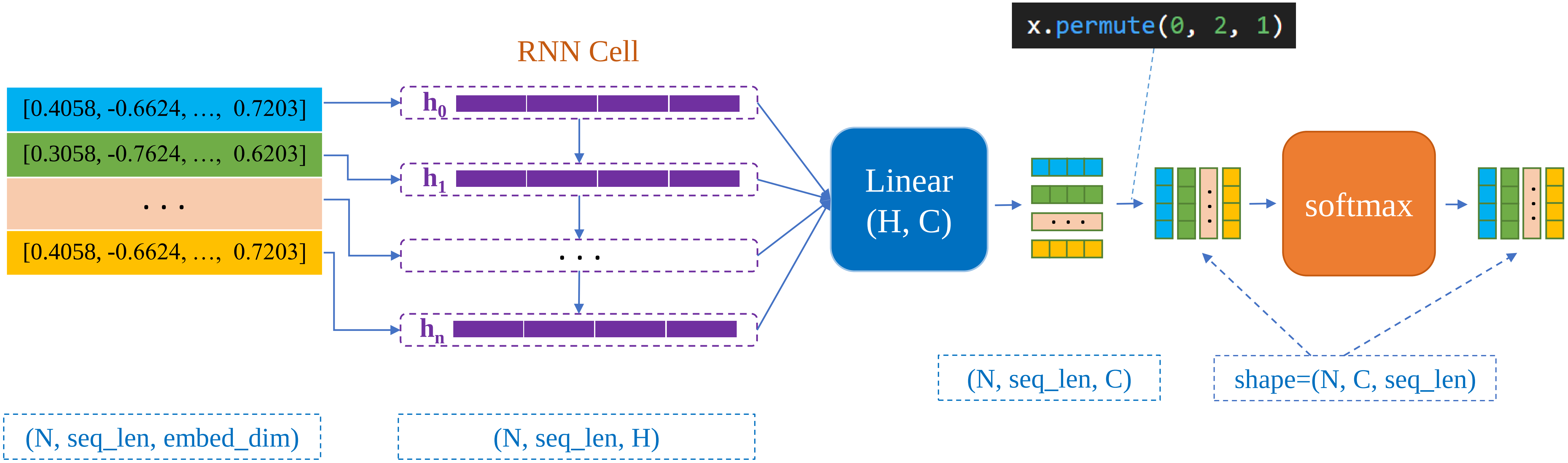


Designing a Model for POS Tagging

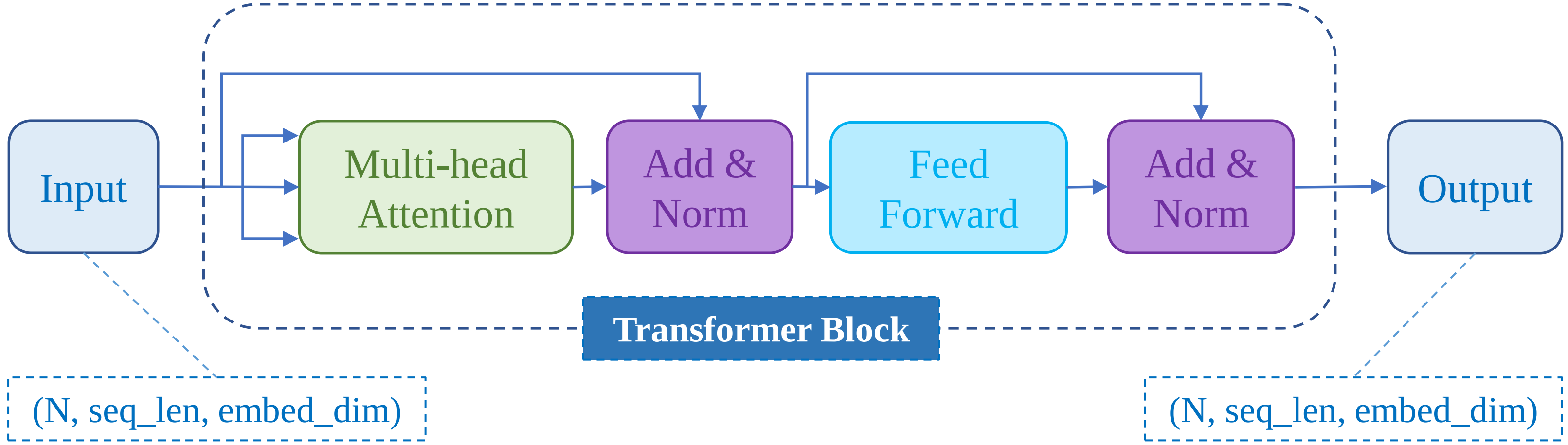
Using RNN + Linear

Similar to LSTM/GRU

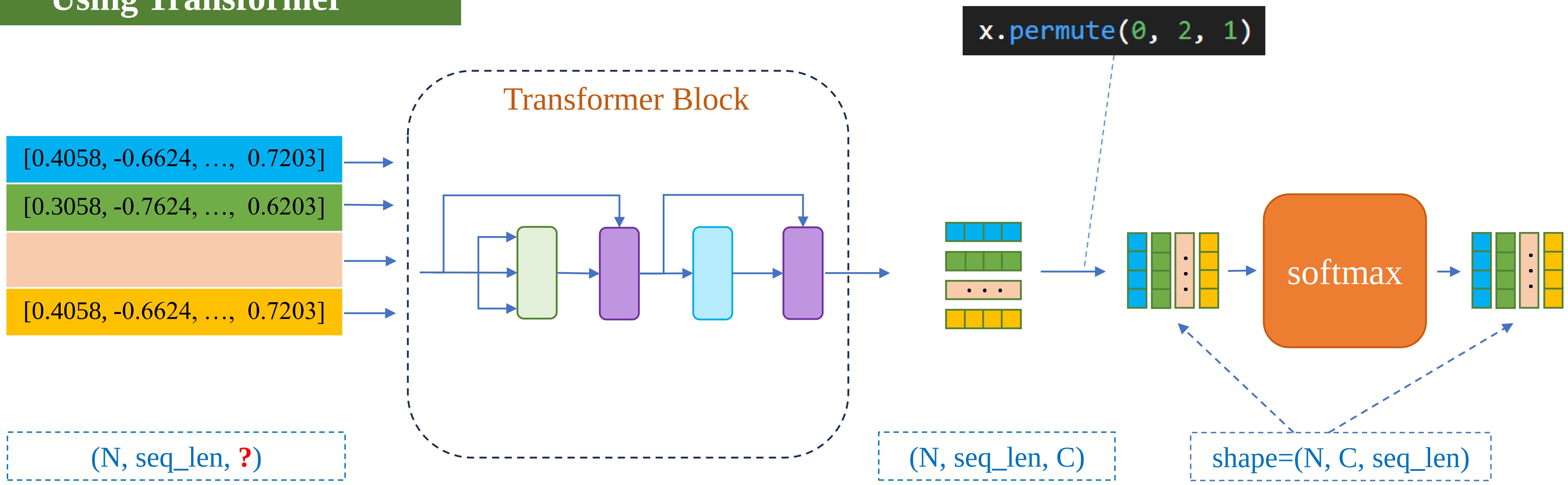
```python
embedding = nn.Embedding(vocab_size, emb_dim)
recurrent = nn.RNN(emb_dim, hidden_size, batch_first=True)
fc = nn.Linear(hidden_size, num_classes)

# forward
x = embedding(x)
output, _ = recurrent(x)
x = fc(output)
x = x.permute(0, 2, 1)
```



Designing a Model for POS Tagging



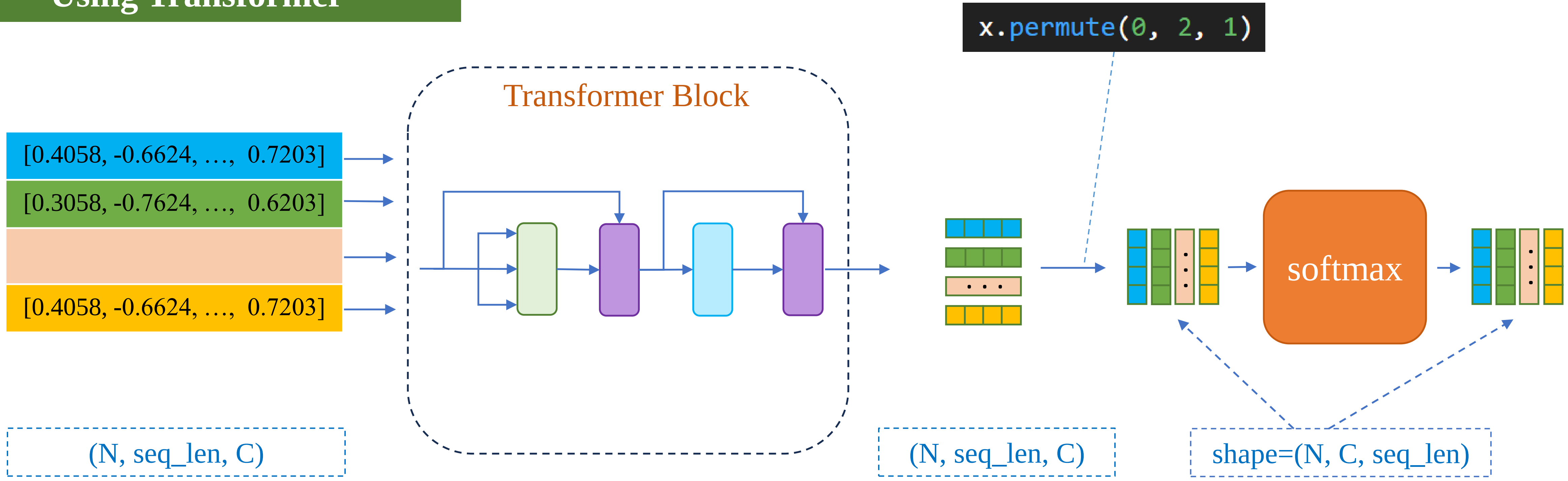
Using Transformer



Designing a Model for POS Tagging

```python
embedding = nn.Embedding(vocab_size, 4)
transformer = TransformerBlock(4, 1, 4)
# embed_dim, num_heads, ff_dim

# forward
x = self.embedding(x)
x = self.transformer(x, x, x)
x =  x.permute(0, 2, 1)
```

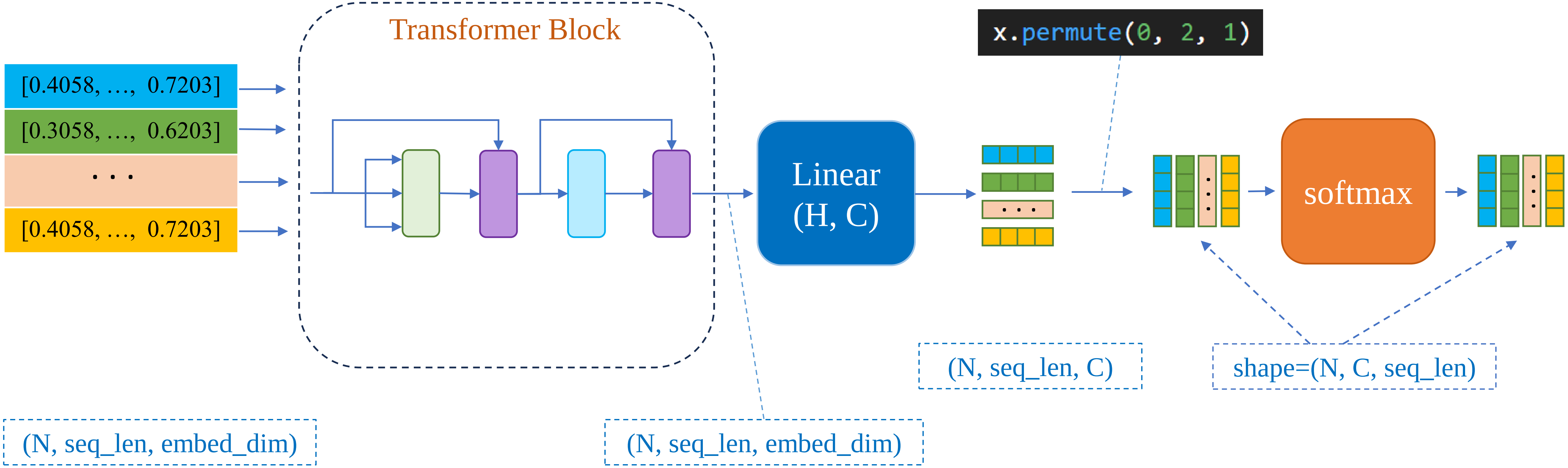
Using Transformer

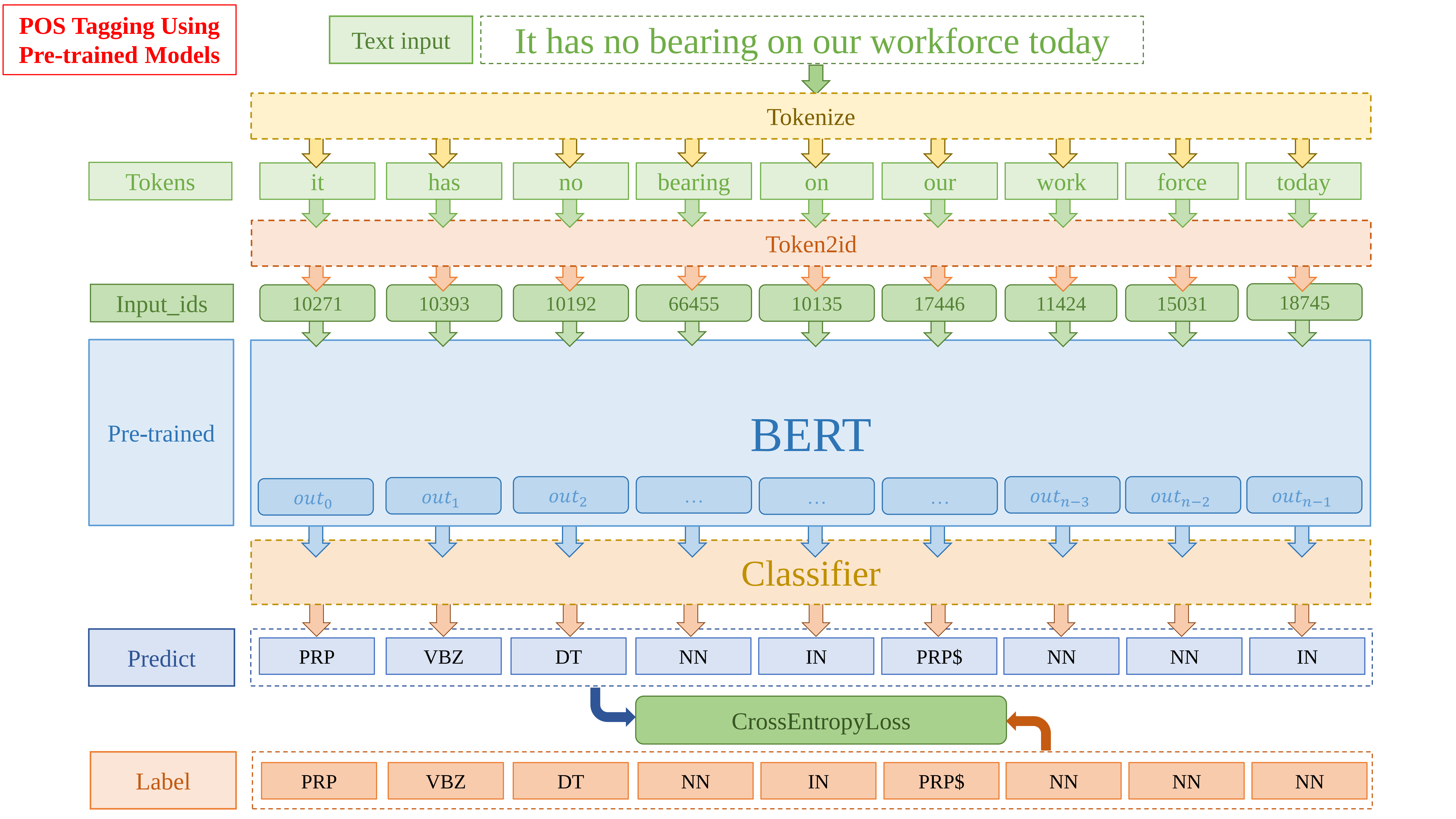


Designing a Model for POS Tagging

Using Transformer + Linear

```
embedding = nn.Embedding(vocab_size, embed_dim)
transformer = TransformerBlock(embed_dim, 1, embed_dim)
fc = nn.Linear(embed_dim, num_classes)

# forward
x = self.embedding(x)
x = self.transformer(x, x, x)
x = self.fc(x)
x = x.permute(0, 2, 1)
```





POS Tagging Using Pre-trained Models
Text input
It has no bearing on our workforce today
Tokenize
Tokens
it
has
no
bearing
on
our
work
force
today
Token2id
Input_ids
10271
10393
10192
66455
10135
17446
11424
15031
18745
Pre-trained
BERT
out_0
out_1
out_2
...
...
...
out_{n-3}
out_{n-2}
out_{n-1}
Classifier
Predict
PRP
VBZ
DT
NN
IN
PRP$
NN
NN
IN
CrossEntropyLoss
Label
PRP
VBZ
DT
NN
IN
PRP$
NN
NN
NN

Outline

- Quick Review
- POS Tagging Using Different Models
- Named Entity Recognition
- Step-by-step Examples
- PyTorch Implementation

Conll2003 Dataset for Part-of-Speed Tagging

Num_classes = 47

Train	Val	Test
14041	3250	3453

0	“	Quotation mask
1		space
2	#	Hash
3	$	Dolla
4	(	Opening parenthesis
5	)	Closing parenthesis
6	,	Comma
7	.	Dot
8	:	Colon
9	``	Apostrophe

10	CC	Coordinating conjunction
11	CD	Cardinal number
12	DT	Determiner
13	EX	Existential *there*
14	FW	Foreign word
15	IN	Preposition or subordinating conjunction
16	JJ	Adjective
17	JJR	Adjective, comparative
18	JJS	Adjective, superlative
19	LS	List item marker

20	MD	Modal
21	NN	Noun, singular or mass
22	NNP	Proper noun, singular
23	NNPS	Proper noun, plural
24	NNS	Noun, plural
25	NN|SYM	Noun or Symbol
26	PDT	Predeterminer
27	POS	Possessive ending
28	PRP	Personal pronoun
29	PRP$	Possessive pronoun

Conll2003 Dataset for Part-of-Speed Tagging

Num_classes = 47

Example

Input tokens

["Cup", "qualifying", "round", ",", "second", "leg", "soccer", "matches", "on", "Thursday"]

Label

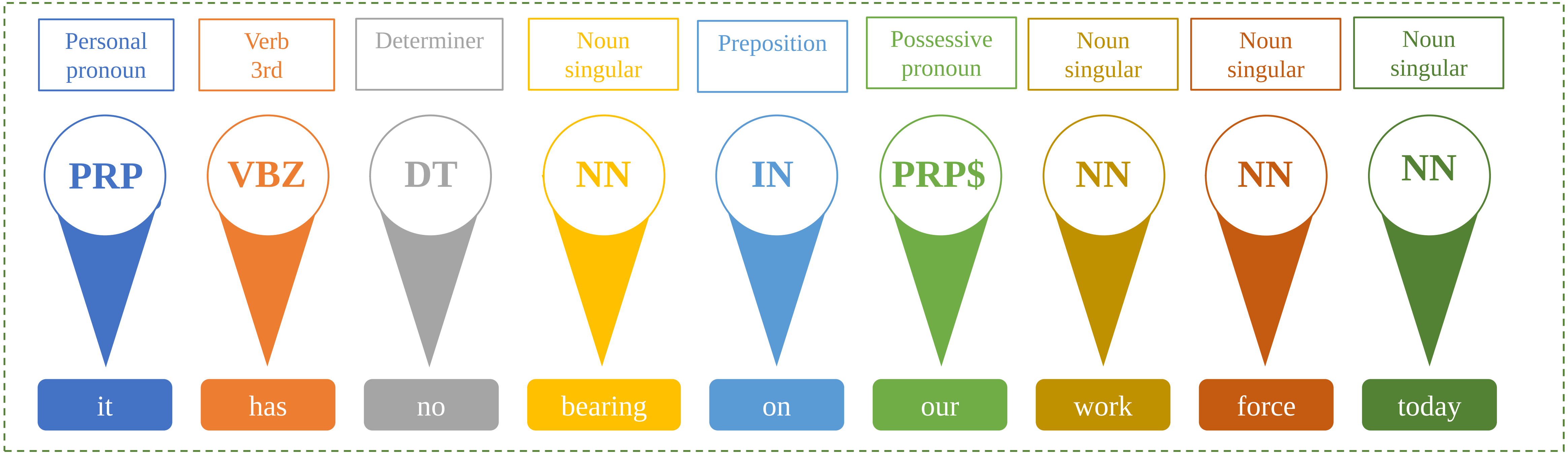
[”NNP", ”VBG", ”RB", ",", ”JJ", ”NN", ”NN", ”NNS", ”IN", ”NNP"]

Label-encoded

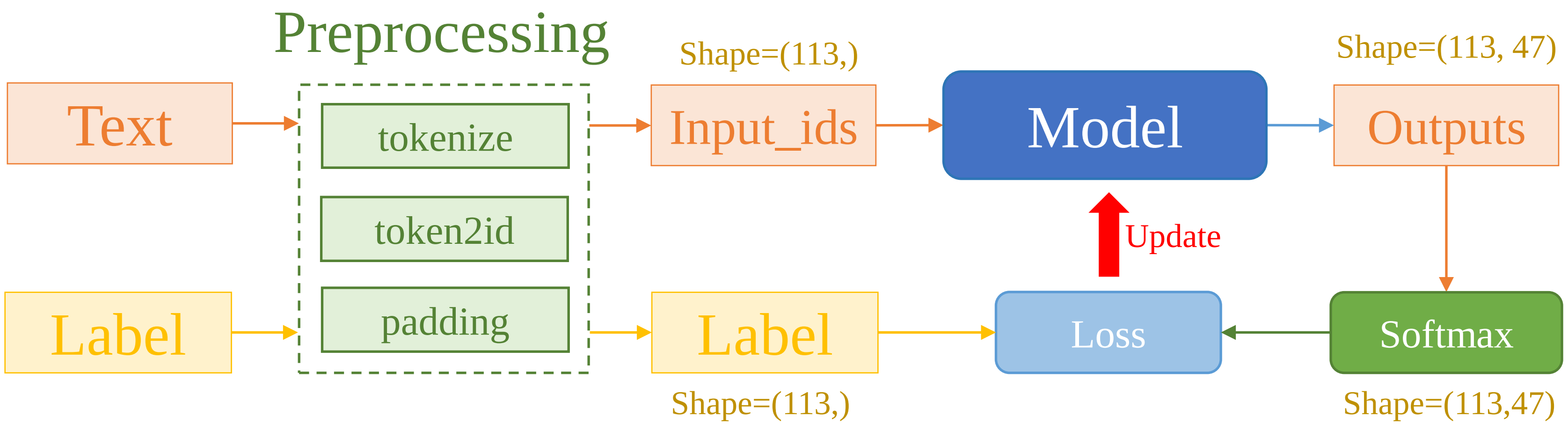
[22, 39, 30, 6, 16, 21, 21, 24, 15, 22,]

30	RB	Adverb
31	RBR	Adverb, comparative
32	RBS	Adverb, superlative
33	RP	Particle
34	SYM	Symbol
35	TO	to
36	UH	Interjection
37	VB	Verb, base form
38	VBD	Verb, past tense
39	VBG	Verb, gerund or present participle
40	VBN	Verb, past participle
41	VBP	Verb, non-3rd person singular present
42	VBZ	Verb, 3rd person singular present
43	WDT	Wh-determiner
44	WP	Wh-pronoun
45	WP$	Possessive wh-pronoun
46	WRB	Wh-adverb

Part-of-speed Tagging



Index	Label
0	<unk>
1	NN
2	IN
3	NNP
…	…
43	LS
44	FW
45	UH
46	SYM



Custom Dataset in Pytorch

Create a Custom Dataset

⇩

__init__(self, …) function:
Khởi tạo các thuộc tính/biến

⇩

__len__(self) function:
Trả về độ dài của dataset

⇩

__getitem__(selfm idx) function:
Xử lý một sample và trả về x và y

```python
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
sequence_length = 5

sample1 = 'We are learning AI'
sample2 = 'AI is a CS topic'
sentences = [sample1, sample2]
labels = [0, 1]
```

```python
from torch.utils.data import Dataset

class MyDataset(Dataset):
    def __init__(self, sentences, labels, tokenizer, max_len):
        super().__init__()
        #...

    def __len__(self):
        return len(self.sentences)

    def __getitem__(self, idx):
        #...

        return tokens, sentence_label
```

Outline

- Quick Review
- POS Tagging Using Different Models
- Named Entity Recognition
- Step-by-step Examples
- PyTorch Implementation

Named Entity Recognition

❖ Introduction

Conll2003 dataset for Named-Entity Recognition

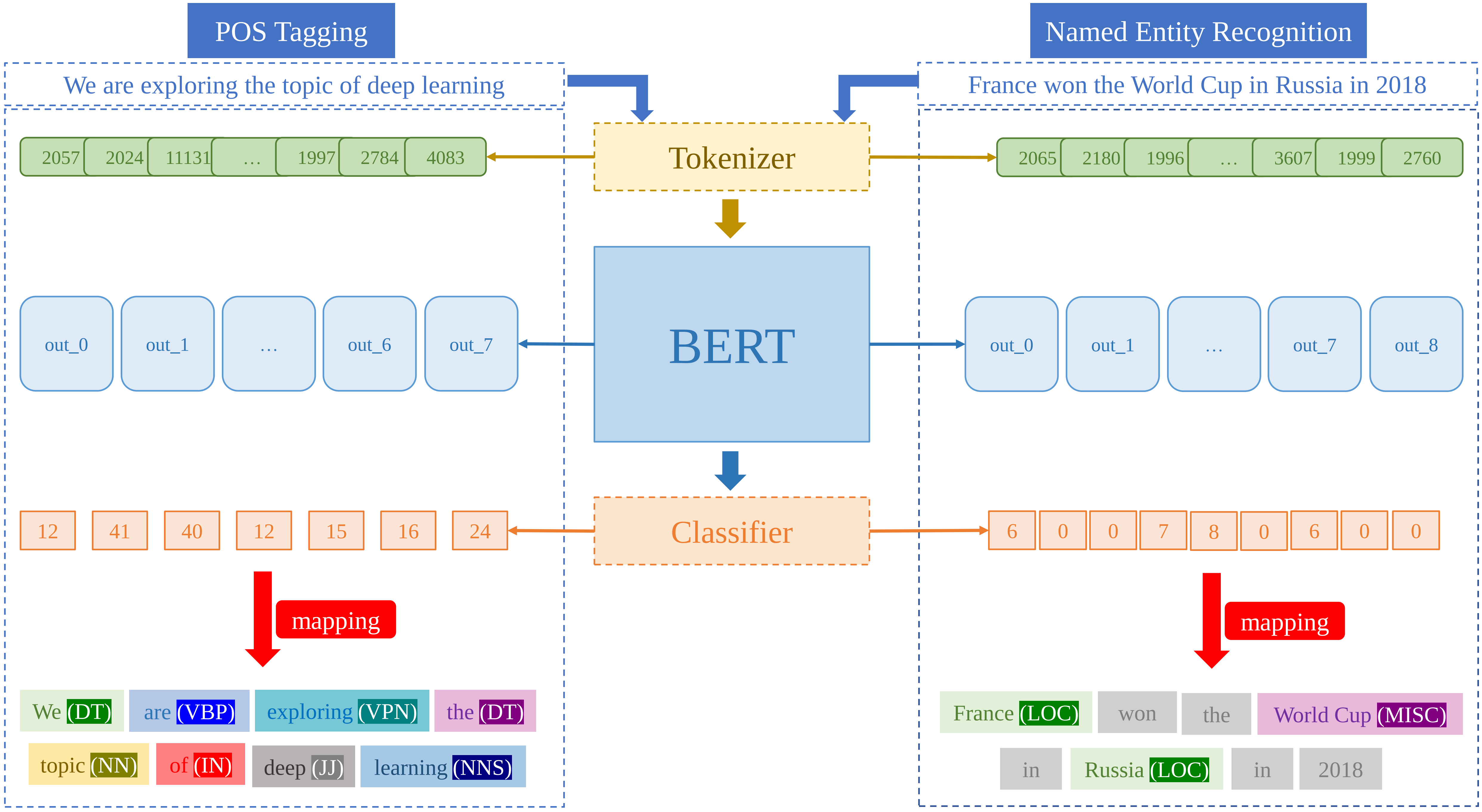
Num_classes = 9

Train	Val	Test
14041	3250	3453

0	O	Out-of-class
1	B-PER	Begin-Person
2	I-PER	In-Person
3	B-ORG	Begin-Organization
4	I-ORG	In-Organization
5	B-LOC	Begin-Location
6	I-LOC	In-Location
7	B-MISC	Begin-Miscellaneous
8	I-MISC	In-Miscellaneous

Example

Input tokens	["BCH", "in", "the", "hive", "of", "Chilean", "pensions"]
Label	[”B-ORG", ”O", ”O", ”O", ”O", ”B-MISC", ”O"]
Label-encoded	[3, 0, 0, 0, 0, 7, 0]



POS Tagging
Named Entity Recognition
We are exploring the topic of deep learning
France won the World Cup in Russia in 2018
Tokenizer
2057
2024
11131
…
1997
2784
4083
2065
2180
1996
…
3607
1999
2760
BERT
out_0
out_1
…
out_6
out_7
out_0
out_1
…
out_7
out_8
Classifier
12
41
40
12
15
16
24
6
0
0
7
8
0
6
0
0
mapping
mapping
We (DT)
are (VBP)
exploring (VPN)
the (DT)
topic (NN)
of (IN)
deep (JJ)
learning (NNS)
France (LOC)
won
the
World Cup (MISC)
in
Russia (LOC)
in
2018

Step-by-step Examples

Named Entity Recognition

Doc	Label
karpathy is working in openai	[0, 4, 4, 4, 2]
geoffrey hinton is from canada	[0, 1, 4, 4, 2]

karpathy	8	[0.7109, -1.2178, -1.5470, -1.2587]
is	2	[0.5303, 0.7931, -1.1894, 0.1906]
working	10	[-0.2059, 1.3111, -1.2398, -1.0455]
in	7	[-0.1117, 1.2757, -0.3398, 0.5976]
openai	9	[-0.4392, 0.5843, -0.7790, 0.2032]

Sample 1 → Sample 1 _ Embedding

building dictionary

index	word
0	[UNK]
1	[pad]
2	is
3	canada
4	from
5	geoffrey
6	hinton
7	in
8	karpathy
9	openai
10	working

Dictionary

0	[-1.5755, 0.0146, 0.2361, 0.3852]
1	[0.2267, -1.1683, 0.0791, -1.3988]
2	[0.5303, 0.7931, -1.1894, 0.1906]
3	[0.0649, -0.0649, 2.3004, 0.3508]
4	[0.4401, -0.1977, 1.1706, -0.4241]
5	[-0.9880, 1.1651, -0.7740, -0.5781]
6	[-0.1220, 0.3313, 0.6327, -0.3742]
7	[-0.1117, 1.2757, -0.3398, 0.5976]
8	[0.7109, -1.2178, -1.5470, -1.2587]
9	[-0.4392, 0.5843, -0.7790, 0.2032]
10	[-0.2059, 1.3111, -1.2398, -1.0455]

Embedding

vocab size = 11
sequence length = 5
num of classes = 5+1

ID	Meaning
0	B-Person
1	I-Person
2	B-Org./Location
3	I-Org./Location
4	Others
5	<padding>

Label Codes

Named Entity Recognition

Doc	Label
karpathy is working in openai	[0, 4, 4, 4, 2]
geoffrey hinton is from canada	[0, 1, 4, 4, 2]

vocab size = 12
sequence length = 5
num of classes = 5+1

ID	Meaning
0	B-Person
1	I-Person
2	B-Org./Location
3	I-Org./Location
4	Others
5	<padding>

Label Codes

Sample 1		Sample 1 _ Embedding (N, seq_len, embed_dim)
karpathy	8	[0.7109, -1.2178, -1.5470, -1.2587]
is	2	[0.5303, 0.7931, -1.1894, 0.1906]
working	10	[-0.2059, 1.3111, -1.2398, -1.0455]
in	7	[-0.1117, 1.2757, -0.3398, 0.5976]
openai	9	[-0.4392, 0.5843, -0.7790, 0.2032]

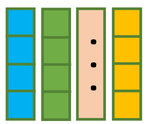
→ → →

Model

Let's design a concrete model!

(CNN, RNN, Transformer, …)

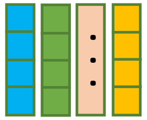
→



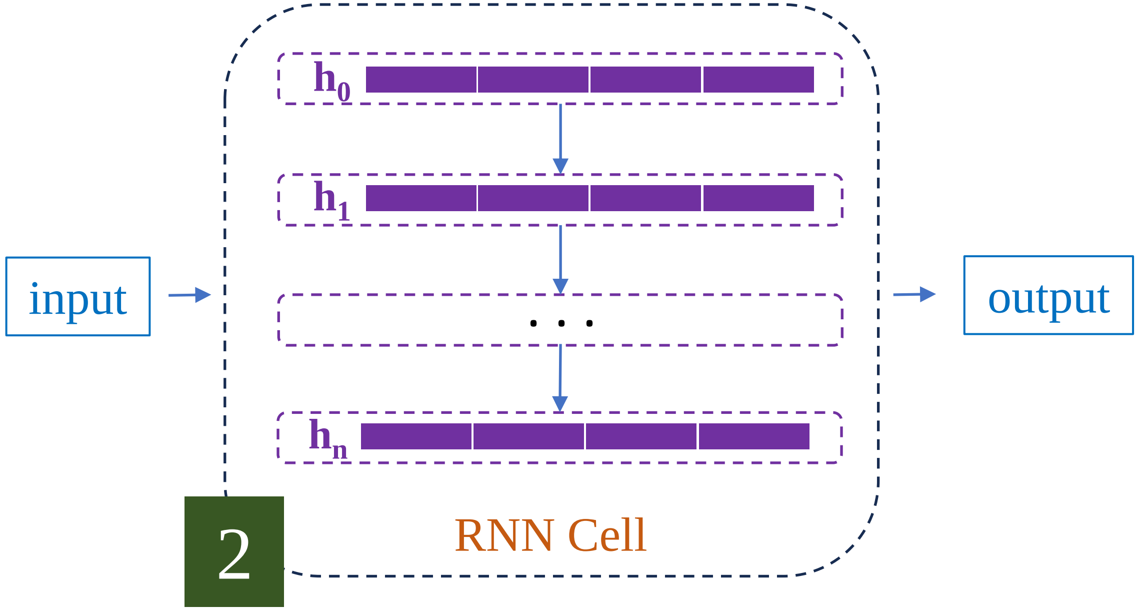
shape=(N, C, seq_len)

[0.7109, -1.2178, -1.5470, -1.2587]
[0.5303, 0.7931, -1.1894, 0.1906]
[-0.2059, 1.3111, -1.2398, -1.0455]
[-0.1117, 1.2757, -0.3398, 0.5976]
[-0.4392, 0.5843, -0.7790, 0.2032]

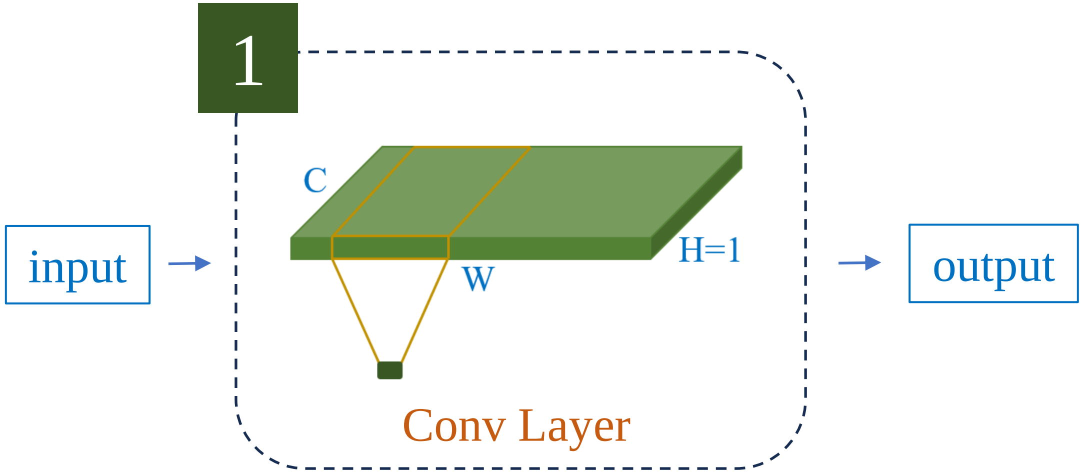
Input: Sample 1 _ Embedding
(N, seq_len, embed_dim)

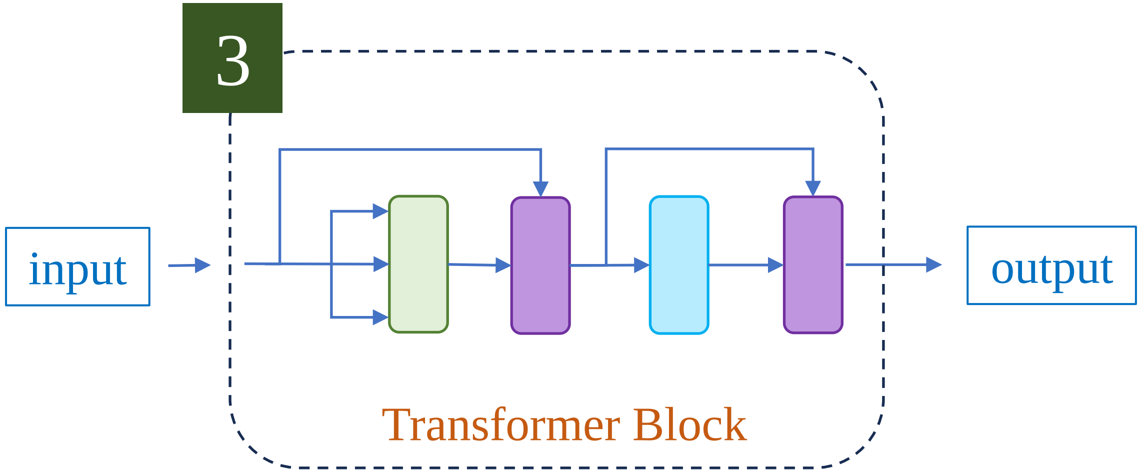


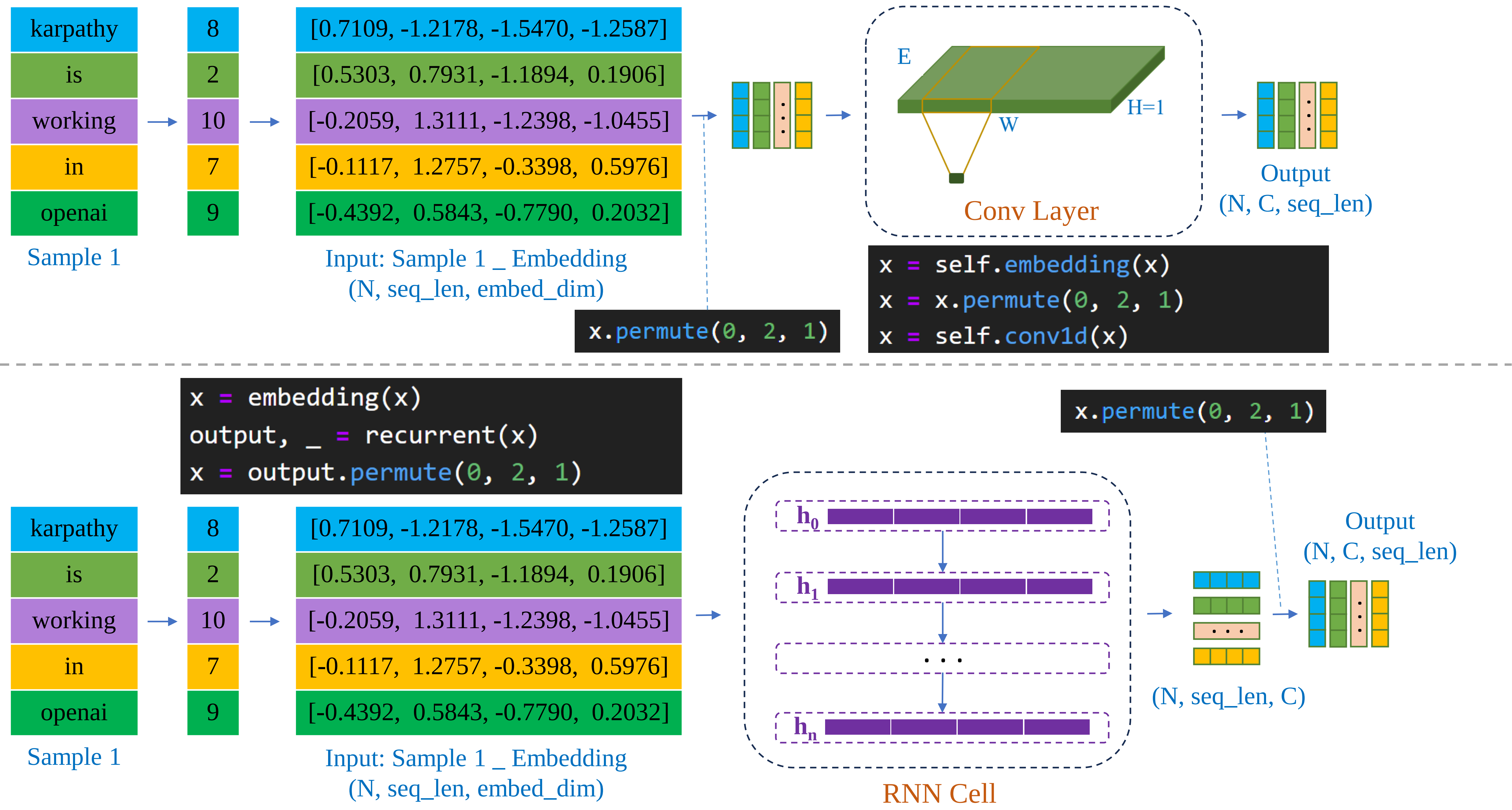
Output
shape=(N, C, seq_len)



Which ones are feasible?







karpathy
is
working
in
openai
Sample 1
8
2
10
7
9
[0.7109, -1.2178, -1.5470, -1.2587]
[0.5303, 0.7931, -1.1894, 0.1906]
[-0.2059, 1.3111, -1.2398, -1.0455]
[-0.1117, 1.2757, -0.3398, 0.5976]
[-0.4392, 0.5843, -0.7790, 0.2032]
Input: Sample 1 _ Embedding
(N, seq_len, embed_dim)
x.permute(0, 2, 1)
E
H=1
W
Conv Layer
Output
(N, C, seq_len)
x = self.embedding(x)
x = x.permute(0, 2, 1)
x = self.conv1d(x)
x = embedding(x)
output, _ = recurrent(x)
x = output.permute(0, 2, 1)
karpathy
is
working
in
openai
Sample 1
8
2
10
7
9
[0.7109, -1.2178, -1.5470, -1.2587]
[0.5303, 0.7931, -1.1894, 0.1906]
[-0.2059, 1.3111, -1.2398, -1.0455]
[-0.1117, 1.2757, -0.3398, 0.5976]
[-0.4392, 0.5843, -0.7790, 0.2032]
Input: Sample 1 _ Embedding
(N, seq_len, embed_dim)
h0
h1
hn
RNN Cell
(N, seq_len, C)
x.permute(0, 2, 1)
Output
(N, C, seq_len)

Designing a Model for POS Tagging

```
embedding = nn.Embedding(vocab_size, 4)
transformer = TransformerBlock(4, 1, 4)
# embed_dim, num_heads, ff_dim

# forward
x = self.embedding(x)
x = self.transformer(x, x, x)
x =  x.permute(0, 2, 1)
```

Using Transformer

```
x.permute(0, 2, 1)
```