AI VIET NAM
@aivietnam.edu.vn

Deep Architectures for POS Tagging

Quang-Vinh Dinh
Ph.D. in Computer Science

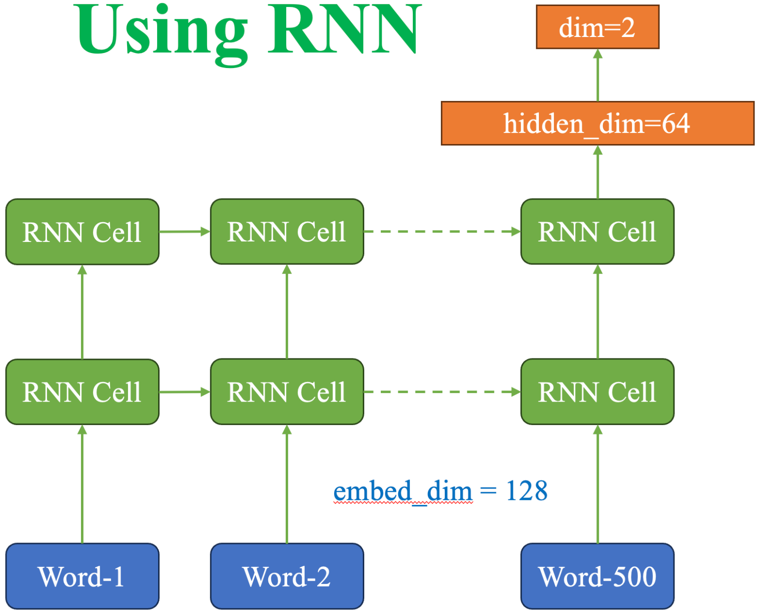
Year 2025

code&data

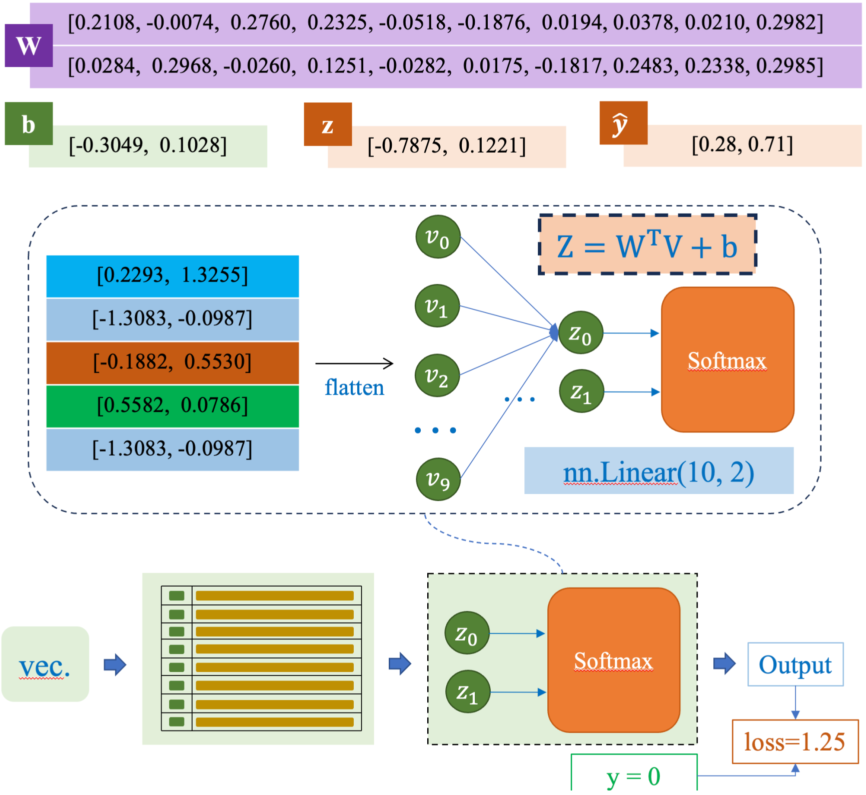
Objectives

Review

Using RNN



Text Classification



W
[0.2108, -0.0074, 0.2760, 0.2325, -0.0518, -0.1876, 0.0194, 0.0378, 0.0210, 0.2982]
[0.0284, 0.2968, -0.0260, 0.1251, -0.0282, 0.0175, -0.1817, 0.2483, 0.2338, 0.2985]

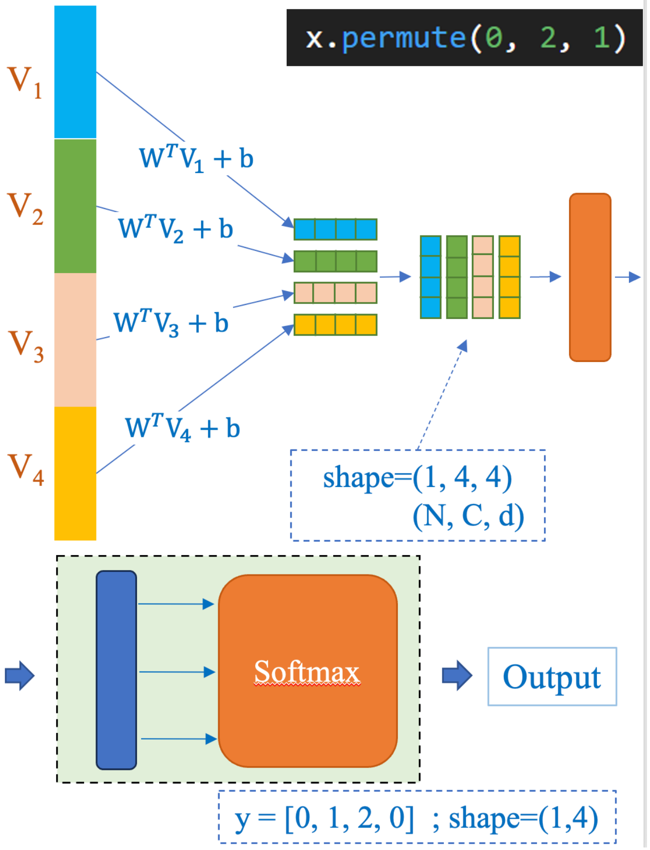
b [-0.3049, 0.1028]

z [-0.7875, 0.1221]

$\hat{y}$ [0.28, 0.71]

[0.2293, 1.3255]
[-1.3083, -0.0987]
[-0.1882, 0.5530]
[0.5582, 0.0786]
[-1.3083, -0.0987]

POS Tagging



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

Outline

SECTION 1

Review

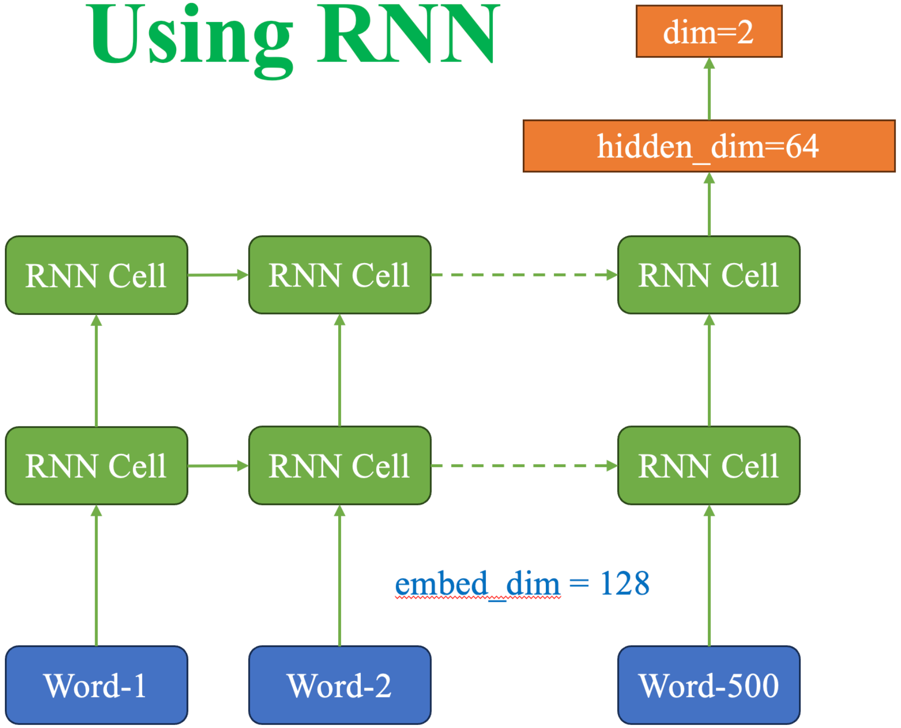
SECTION 2

Text Classification

SECTION 3

POS Tagging

Using RNN



Text Classification

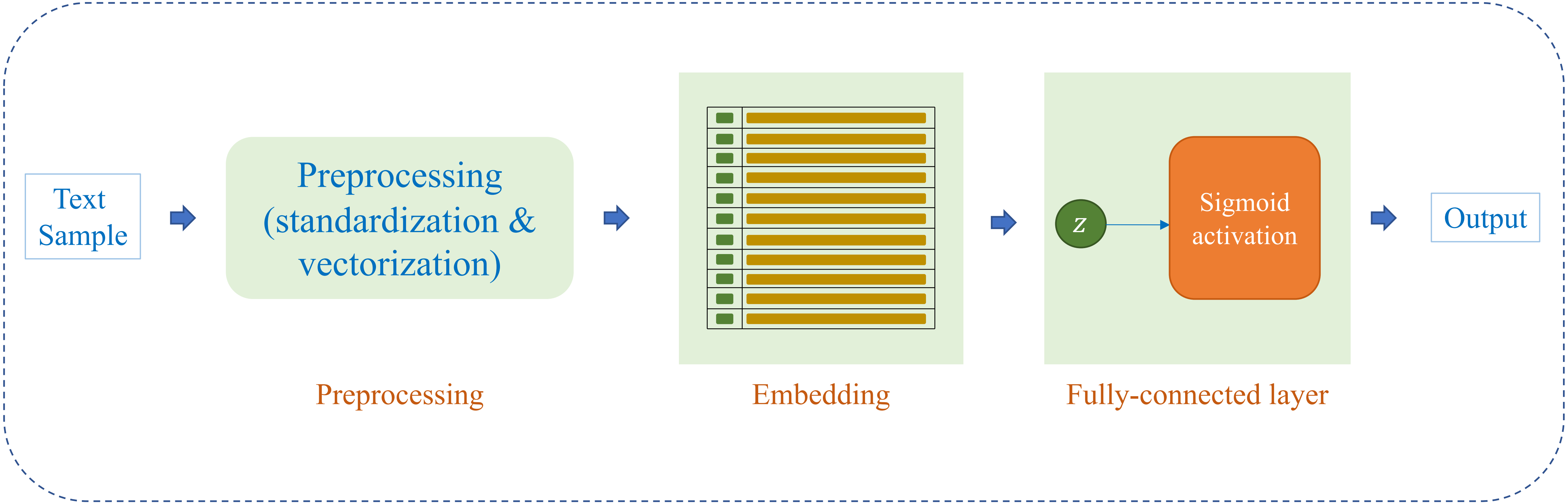
❖ IMDB dataset

- 50,000 movie review for sentiment analysis
- Consist of: + 25,000 movie review for training
  + 25,000 movie review for testing
- Label: positive – negative

“A wonderful little production. <br /><br />The filming technique is very unassuming- very old-time-BBC fashion and gives a comforting, and sometimes discomforting, sense of realism to the entire piece…..”	positive
“This show was an amazing, fresh & innovative idea in the 70's when it first aired. The first 7 or 8 years were brilliant, but things dropped off after that. By 1990, the show was not really funny anymore, and it's continued its decline further to the complete waste of time it is today….”	negative
“I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air conditioned theater and watching a light-hearted comedy. The plot is simplistic, but the dialogue is witty and the characters are likable (even the well bread suspected serial killer)….”	positive
“BTW Carver gets a very annoying sidekick who makes you wanna shoot him the first three minutes he's on screen.”	negative

Text Classification

❖ Simple approach



Text Sample

Preprocessing (standardization & vectorization)

Z

Sigmoid activation

Output

Preprocessing

Embedding

Fully-connected layer

Embedding

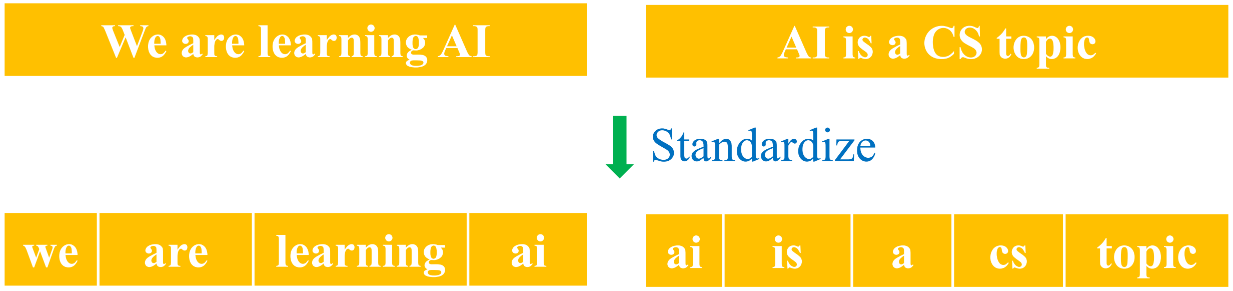
index	0	1	2	3	4	5	6	7
word	[UNK]	pad	ai	a	are	cs	is	learning

- Example corpus

  sample1: ‘We are learning AI’

  sample2: ‘AI is a CS topic’

(1) Build vocabulary from corpus



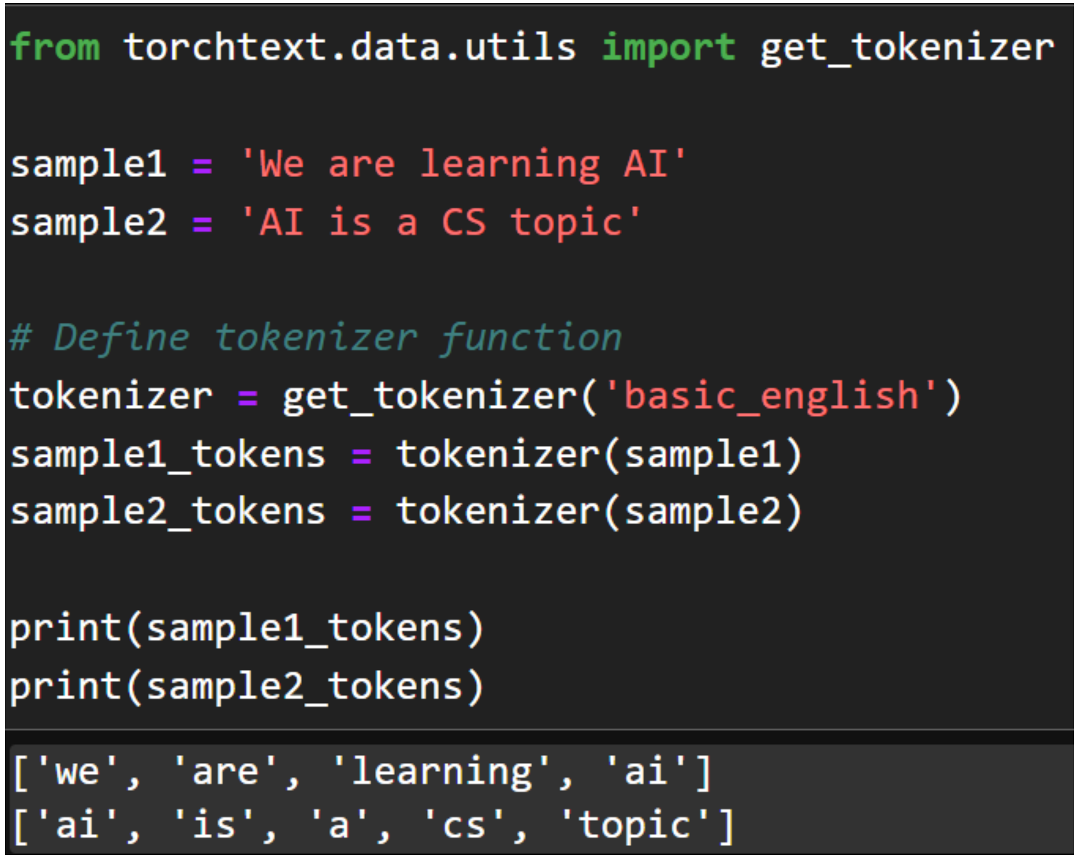
We are learning AI

AI is a CS topic

Standardize

we | are | learning | ai

ai | is | a | cs | topic



```
from torchtext.data.utils import get_tokenizer

sample1 = 'We are learning AI'
sample2 = 'AI is a CS topic'

# Define tokenizer function
tokenizer = get_tokenizer('basic_english')
sample1_tokens = tokenizer(sample1)
sample2_tokens = tokenizer(sample2)

print(sample1_tokens)
print(sample2_tokens)
```

```
['we', 'are', 'learning', 'ai']
['ai', 'is', 'a', 'cs', 'topic']
```

Embedding

index	0	1	2	3	4	5	6	7
word	[UNK]	pad	ai	a	are	cs	is	learning

- Example corpus

  sample1: ‘We are learning AI’

  sample2: ‘AI is a CS topic’

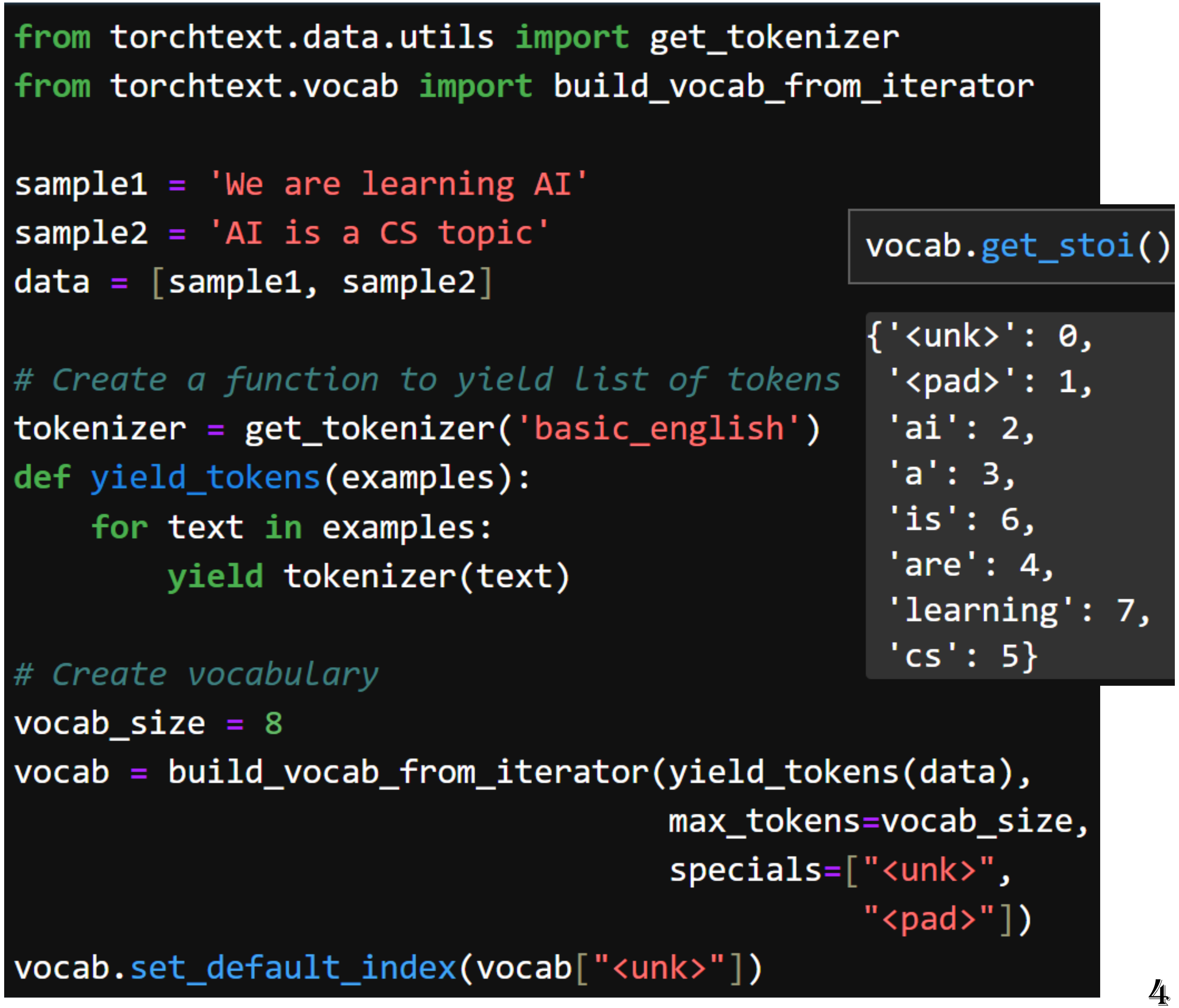
(1) Build vocabulary from corpus

#different words are enormous

How to represent ‘text’ effectively?

➡ Use a limited number of words

➡ Get data sample-by-sample



```
from torchtext.data.utils import get_tokenizer
from torchtext.vocab import build_vocab_from_iterator

sample1 = 'We are learning AI'
sample2 = 'AI is a CS topic'
data = [sample1, sample2]

# Create a function to yield list of tokens
tokenizer = get_tokenizer('basic_english')
def yield_tokens(examples):
    for text in examples:
        yield tokenizer(text)

# Create vocabulary
vocab_size = 8
vocab = build_vocab_from_iterator(yield_tokens(data),
                                  max_tokens=vocab_size,
                                  specials=["<unk>",
                                            "<pad>"])
vocab.set_default_index(vocab["<unk>"])
```

```
vocab.get_stoi()
```

```
{'<unk>': 0,
 '<pad>': 1,
 'ai': 2,
 'a': 3,
 'is': 6,
 'are': 4,
 'learning': 7,
 'cs': 5}
```

Embedding

index	0	1	2	3	4	5	6	7
word	[UNK]	pad	ai	a	are	cs	is	learning

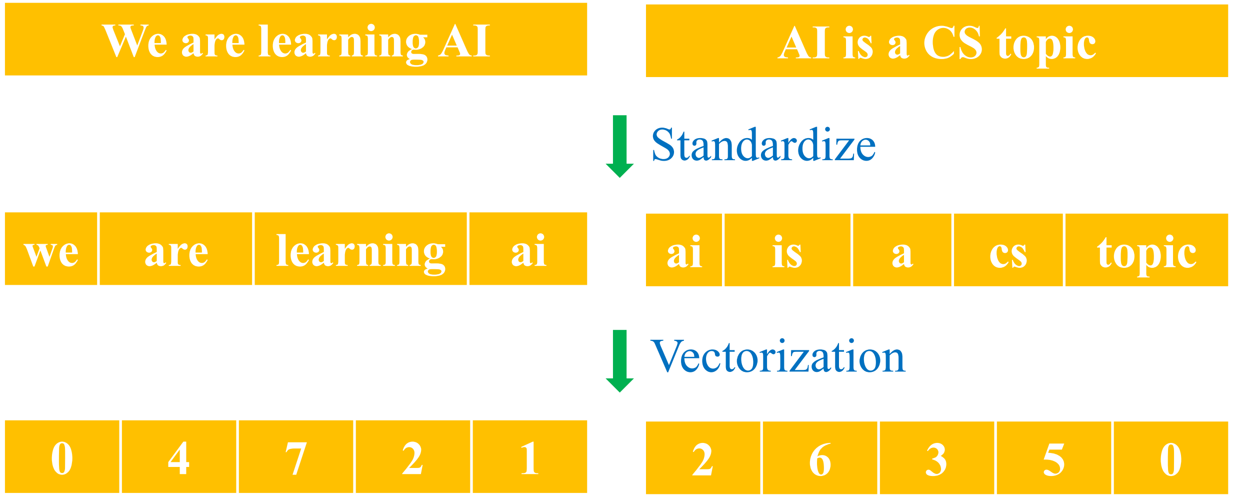
- Example corpus

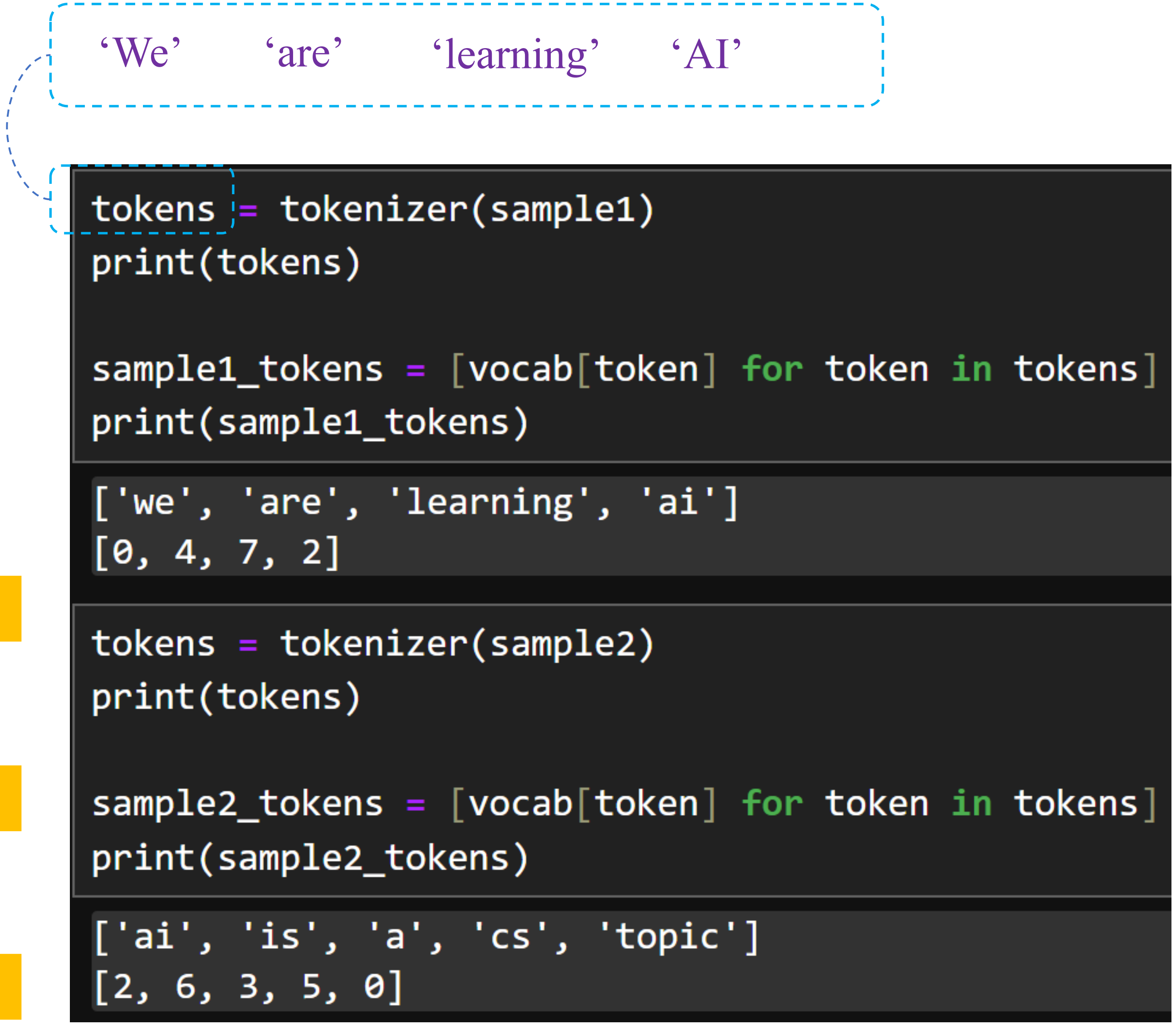
  sample1: ‘We are learning AI’

  sample2: ‘AI is a CS topic’

(1) Build vocabulary from corpus

(2) Transform text into features





```
tokens = tokenizer(sample1)
print(tokens)

sample1_tokens = [vocab[token] for token in tokens]
print(sample1_tokens)
```

```
['we', 'are', 'learning', 'ai']
[0, 4, 7, 2]
```

```
tokens = tokenizer(sample2)
print(tokens)

sample2_tokens = [vocab[token] for token in tokens]
print(sample2_tokens)
```

```
['ai', 'is', 'a', 'cs', 'topic']
[2, 6, 3, 5, 0]
```

Embedding

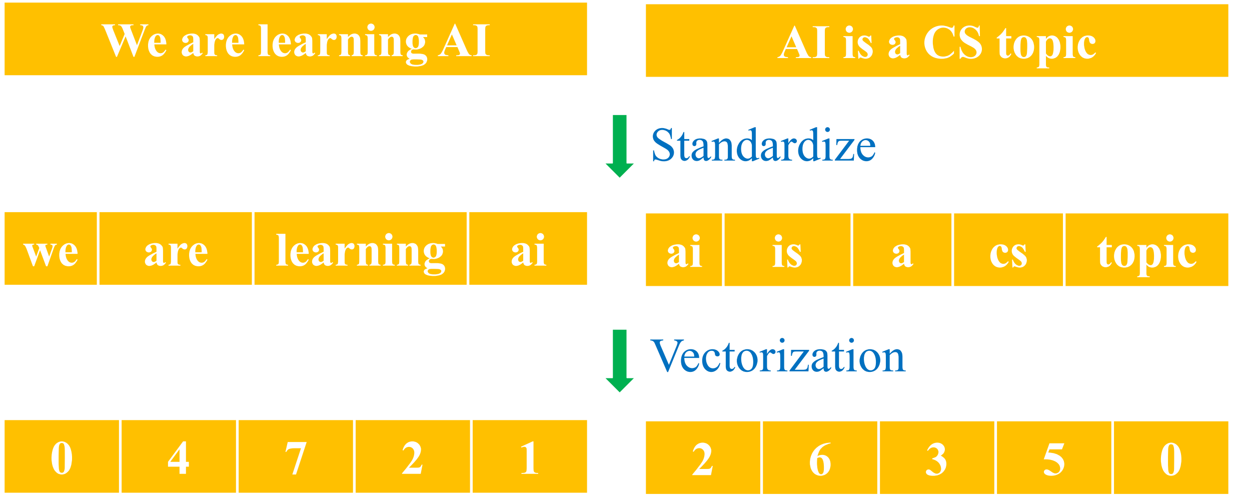
- Example corpus

  sample1: ‘We are learning AI’

  sample2: ‘AI is a CS topic’

(1) Build vocabulary from corpus

(2) Transform text into features



We are learning AI

AI is a CS topic

Standardize

we | are | learning | ai

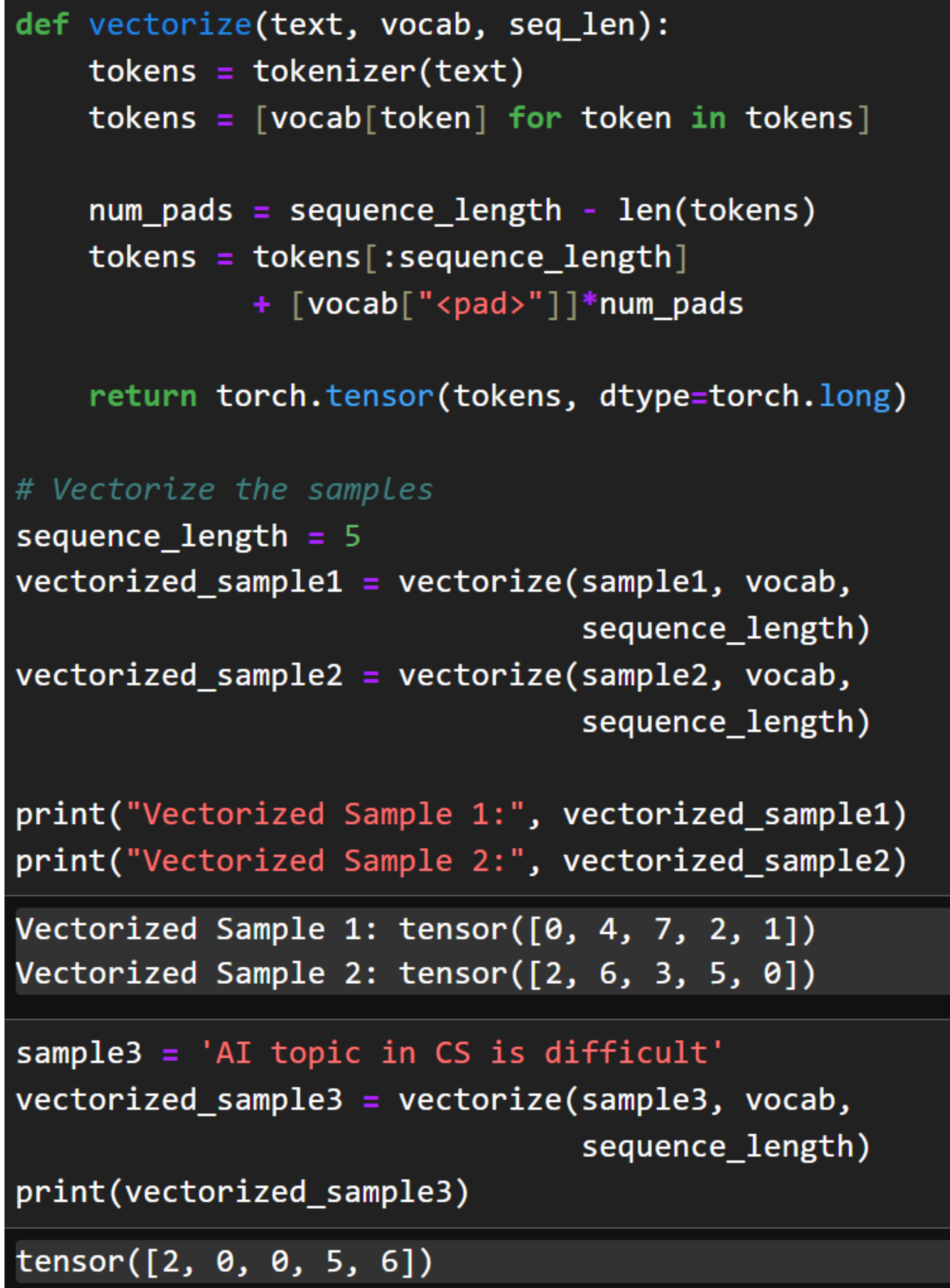
ai | is | a | cs | topic

Vectorization

0 | 4 | 7 | 2 | 1

2 | 6 | 3 | 5 | 0

index	0	1	2	3	4	5	6	7
word	[UNK]	pad	ai	a	are	cs	is	learning



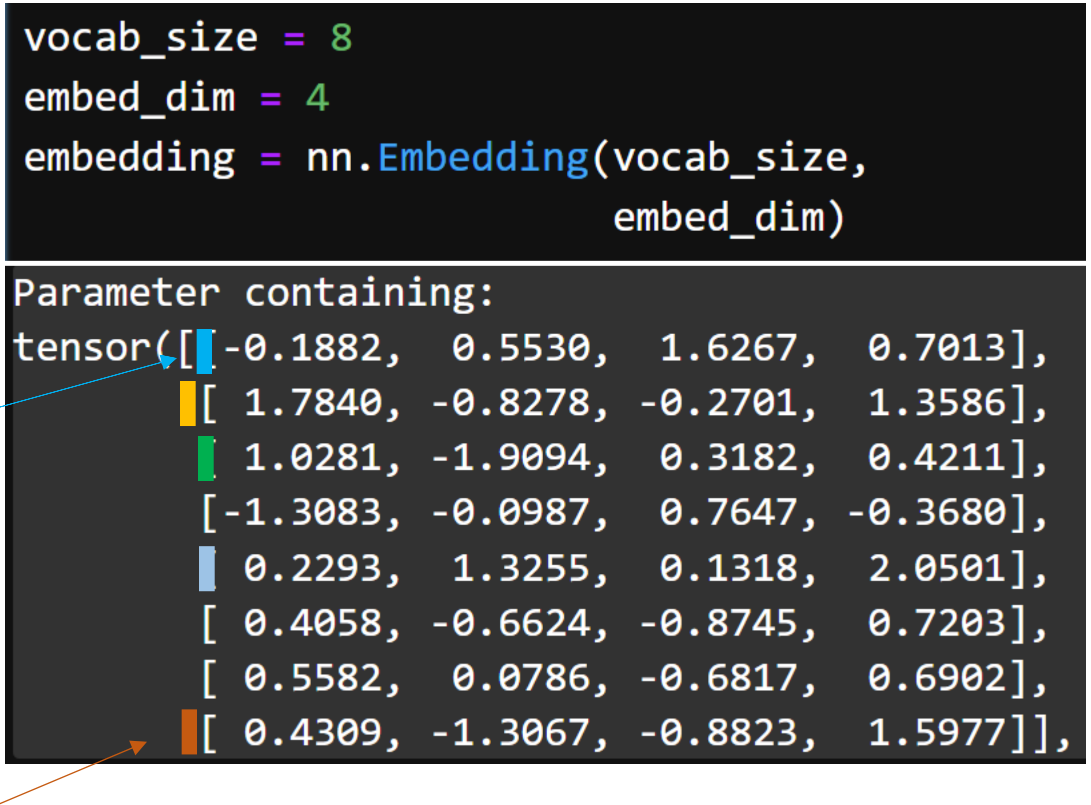
```
def vectorize(text, vocab, seq_len):
    tokens = tokenizer(text)
    tokens = [vocab[token] for token in tokens]

    num_pads = sequence_length - len(tokens)
    tokens = tokens[:sequence_length]
             + [vocab["<pad>"]]*num_pads

    return torch.tensor(tokens, dtype=torch.long)

# Vectorize the samples
sequence_length = 5
vectorized_sample1 = vectorize(sample1, vocab,
                               sequence_length)
vectorized_sample2 = vectorize(sample2, vocab,
                               sequence_length)

print("Vectorized Sample 1:", vectorized_sample1)
print("Vectorized Sample 2:", vectorized_sample2)
```

```
Vectorized Sample 1: tensor([0, 4, 7, 2, 1])
Vectorized Sample 2: tensor([2, 6, 3, 5, 0])
```

```
sample3 = 'AI topic in CS is difficult'
vectorized_sample3 = vectorize(sample3, vocab,
                               sequence_length)
print(vectorized_sample3)
```

```
tensor([2, 0, 0, 5, 6])
```

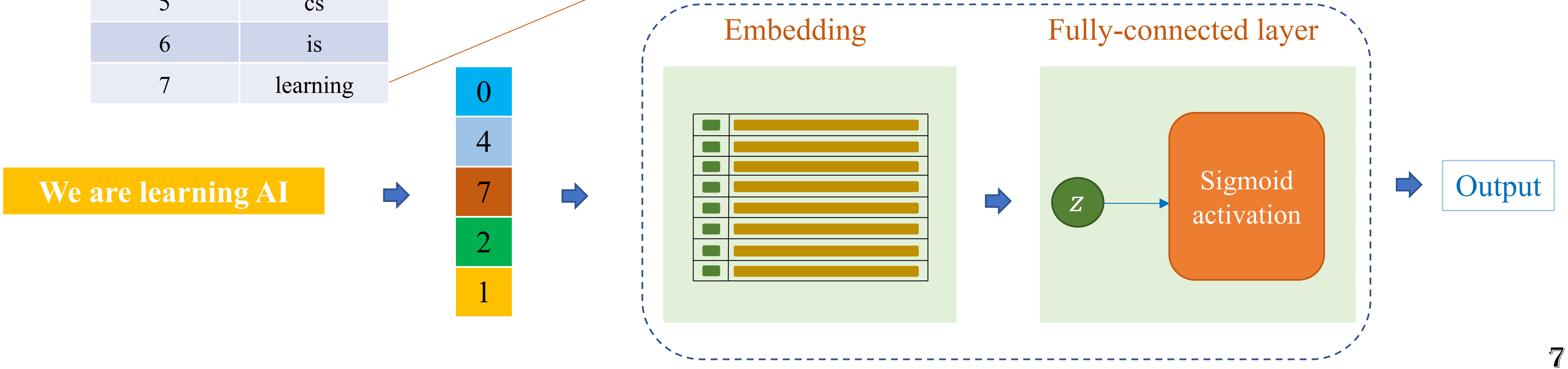
Embedding Layer



```
vocab_size = 8
embed_dim = 4
embedding = nn.Embedding(vocab_size,
                         embed_dim)
```

```
Parameter containing:
tensor([[-0.1882,  0.5530,  1.6267,  0.7013],
        [ 1.7840, -0.8278, -0.2701,  1.3586],
        [ 1.0281, -1.9094,  0.3182,  0.4211],
        [-1.3083, -0.0987,  0.7647, -0.3680],
        [ 0.2293,  1.3255,  0.1318,  2.0501],
        [ 0.4058, -0.6624, -0.8745,  0.7203],
        [ 0.5582,  0.0786, -0.6817,  0.6902],
        [ 0.4309, -1.3067, -0.8823,  1.5977]],
```

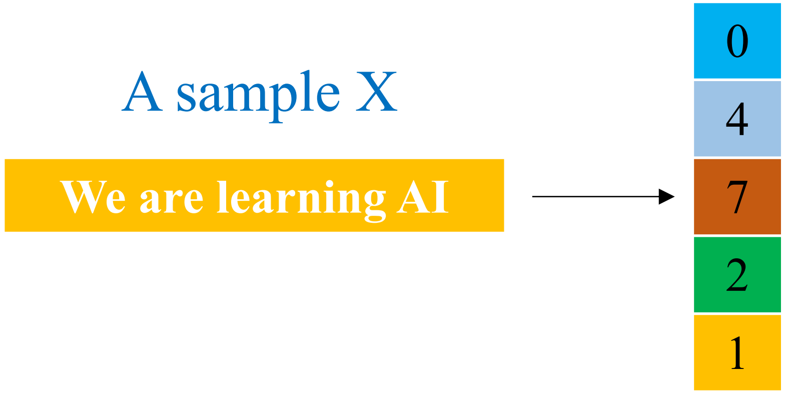
index	word
0	[UNK]
1	[pad]
2	ai
3	a
4	are
5	cs
6	is
7	learning



Revisit input x

Convert from text to numbers

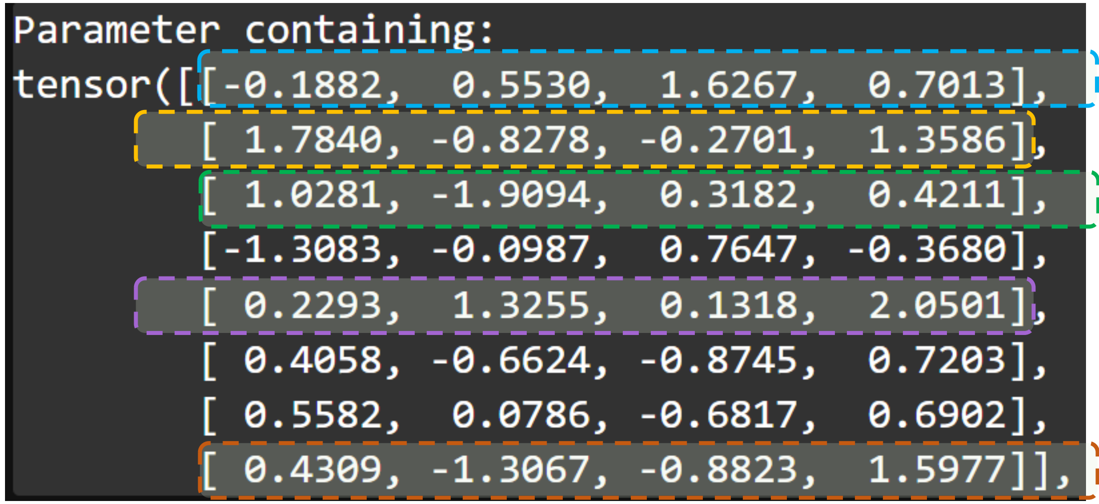
index	word
0	[UNK]
1	[pad]
2	ai
3	a
4	are
5	cs
6	is
7	learning



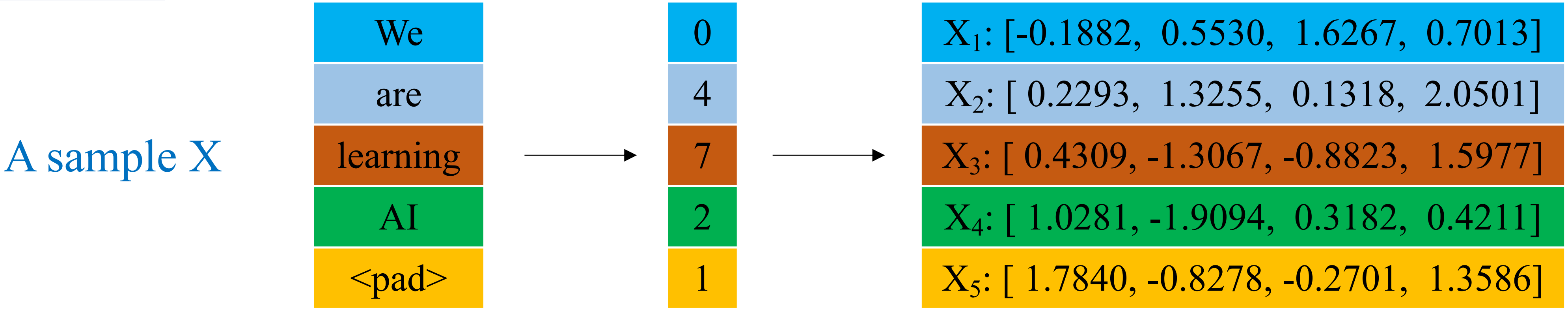
A sample X

We are learning AI

0
4
7
2
1



```
Parameter containing:
tensor([[-0.1882,  0.5530,  1.6267,  0.7013],
        [ 1.7840, -0.8278, -0.2701,  1.3586],
        [ 1.0281, -1.9094,  0.3182,  0.4211],
        [-1.3083, -0.0987,  0.7647, -0.3680],
        [ 0.2293,  1.3255,  0.1318,  2.0501],
        [ 0.4058, -0.6624, -0.8745,  0.7203],
        [ 0.5582,  0.0786, -0.6817,  0.6902],
        [ 0.4309, -1.3067, -0.8823,  1.5977]],
```

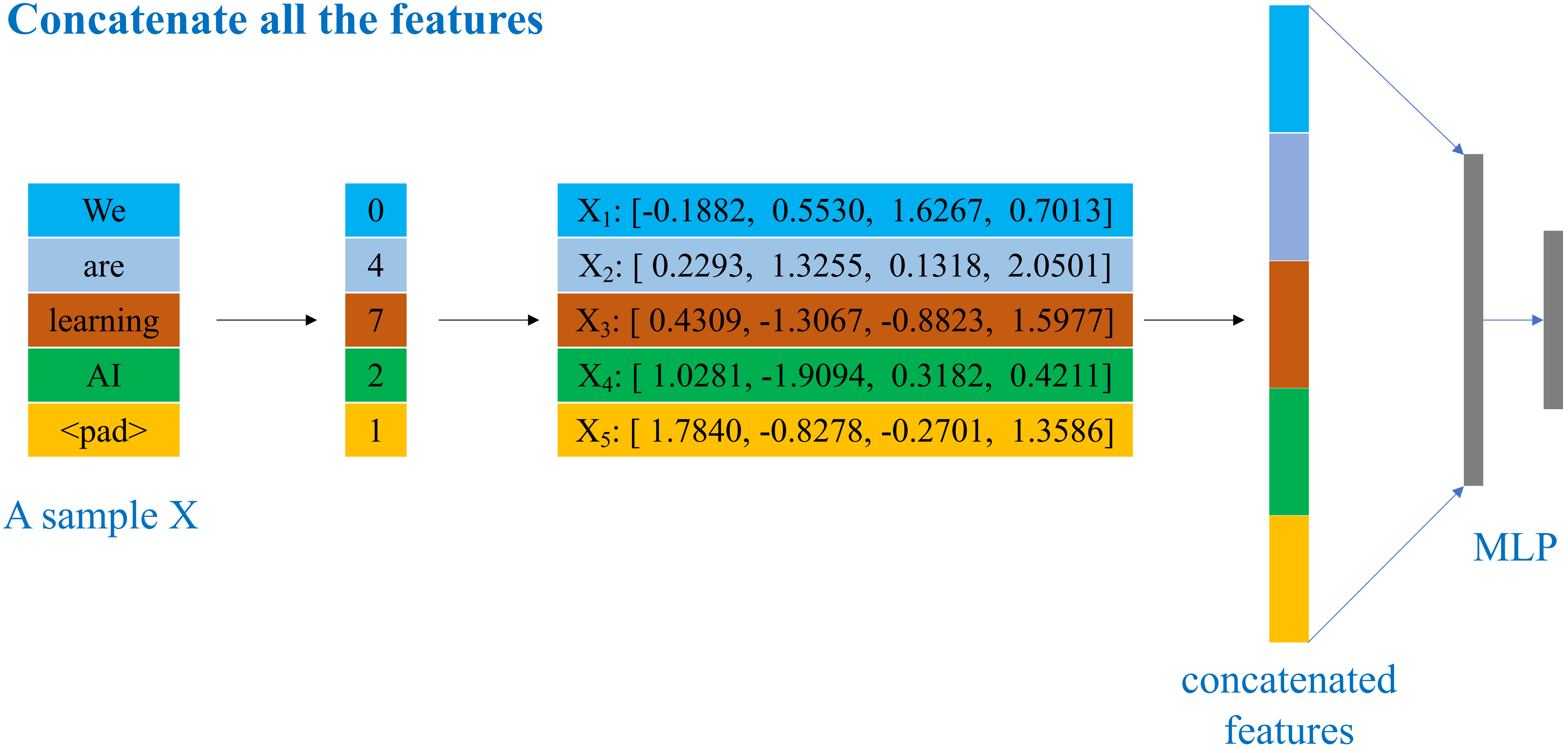


A sample X

We	0	X_1: [-0.1882, 0.5530, 1.6267, 0.7013]
are	4	X_2: [0.2293, 1.3255, 0.1318, 2.0501]
learning	7	X_3: [0.4309, -1.3067, -0.8823, 1.5977]
AI	2	X_4: [1.0281, -1.9094, 0.3182, 0.4211]
<pad>	1	X_5: [1.7840, -0.8278, -0.2701, 1.3586]

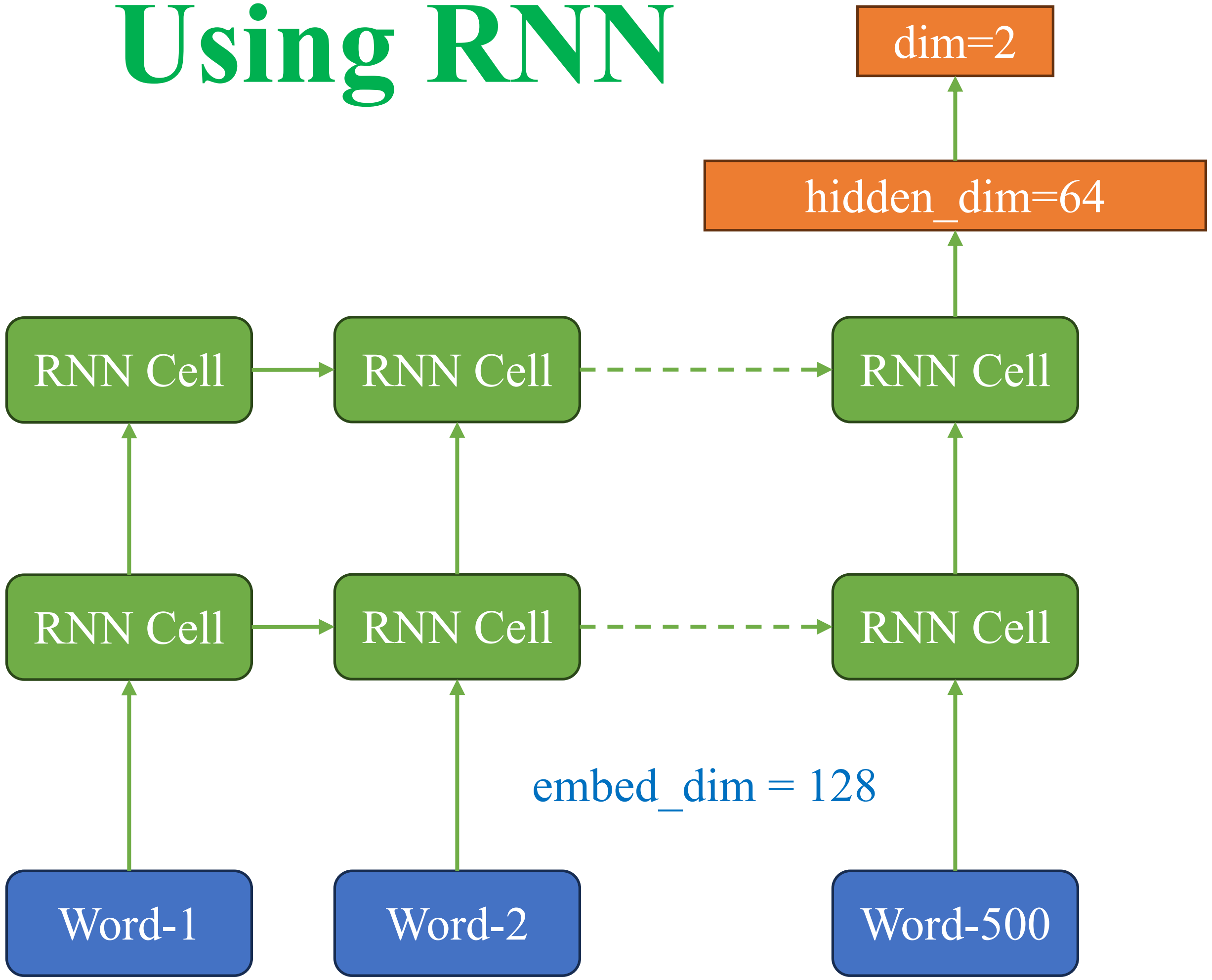
How to deal with this input?

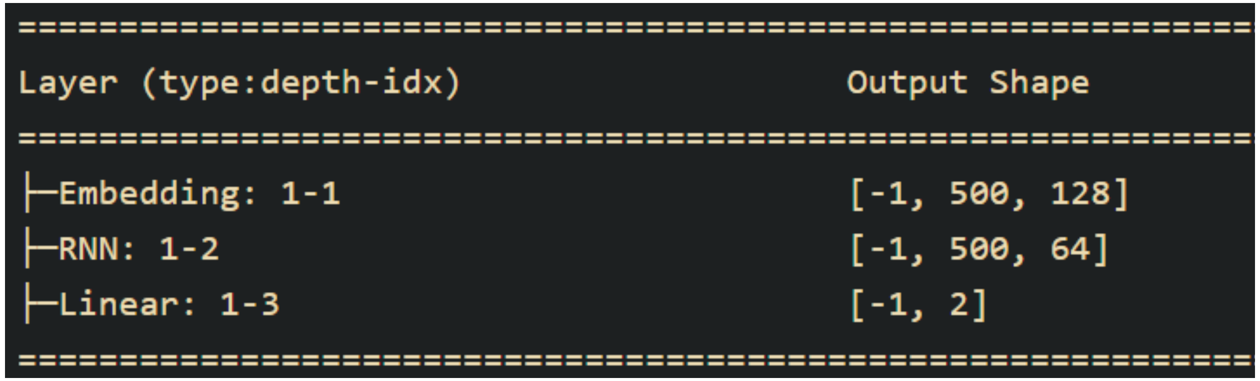
- ❖ **Simplest idea: Based on MLP**
- ❖ **Concatenate all the features**



We	0	X_1: [-0.1882, 0.5530, 1.6267, 0.7013]
are	4	X_2: [0.2293, 1.3255, 0.1318, 2.0501]
learning	7	X_3: [0.4309, -1.3067, -0.8823, 1.5977]
AI	2	X_4: [1.0281, -1.9094, 0.3182, 0.4211]
<pad>	1	X_5: [1.7840, -0.8278, -0.2701, 1.3586]

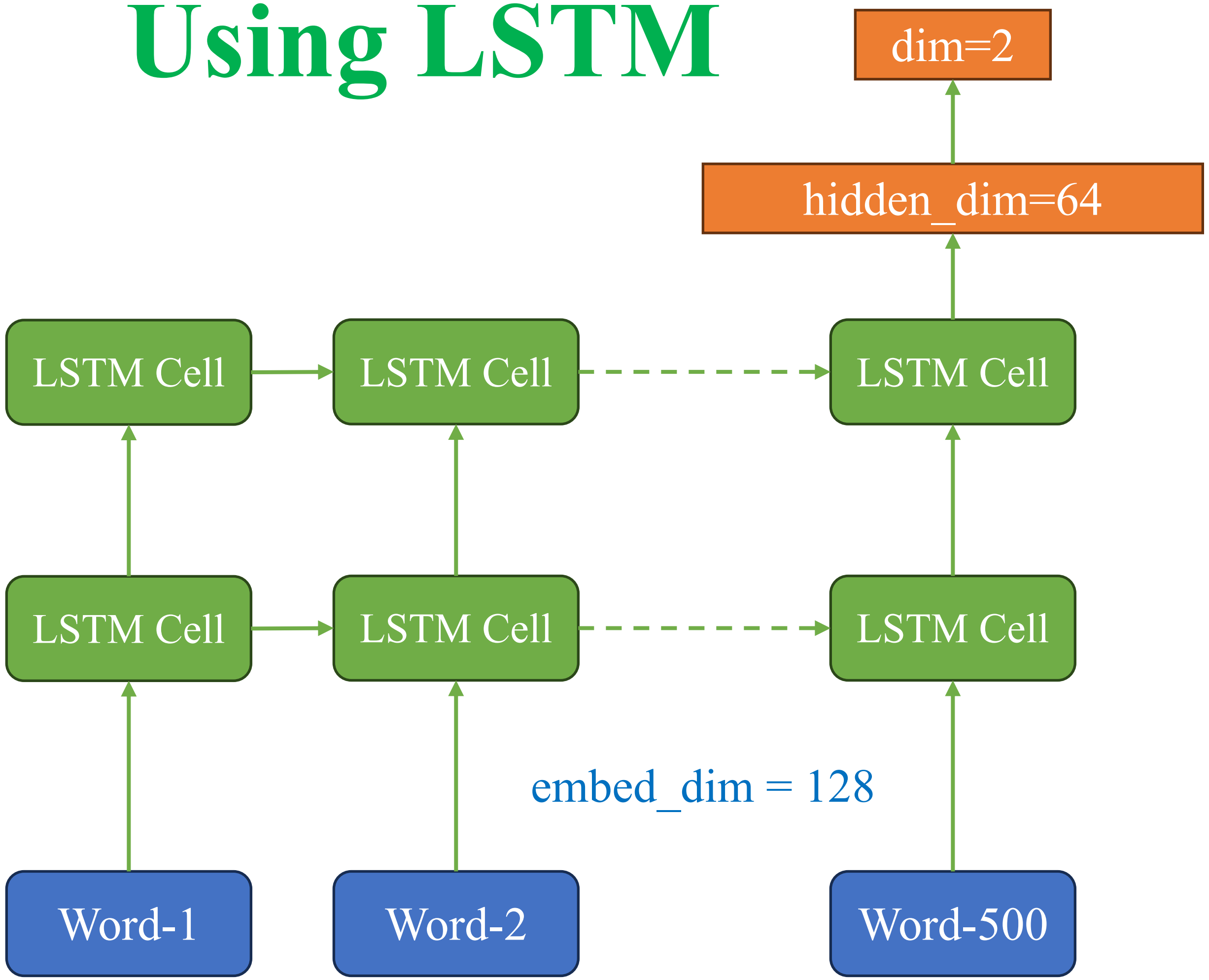
Using RNN



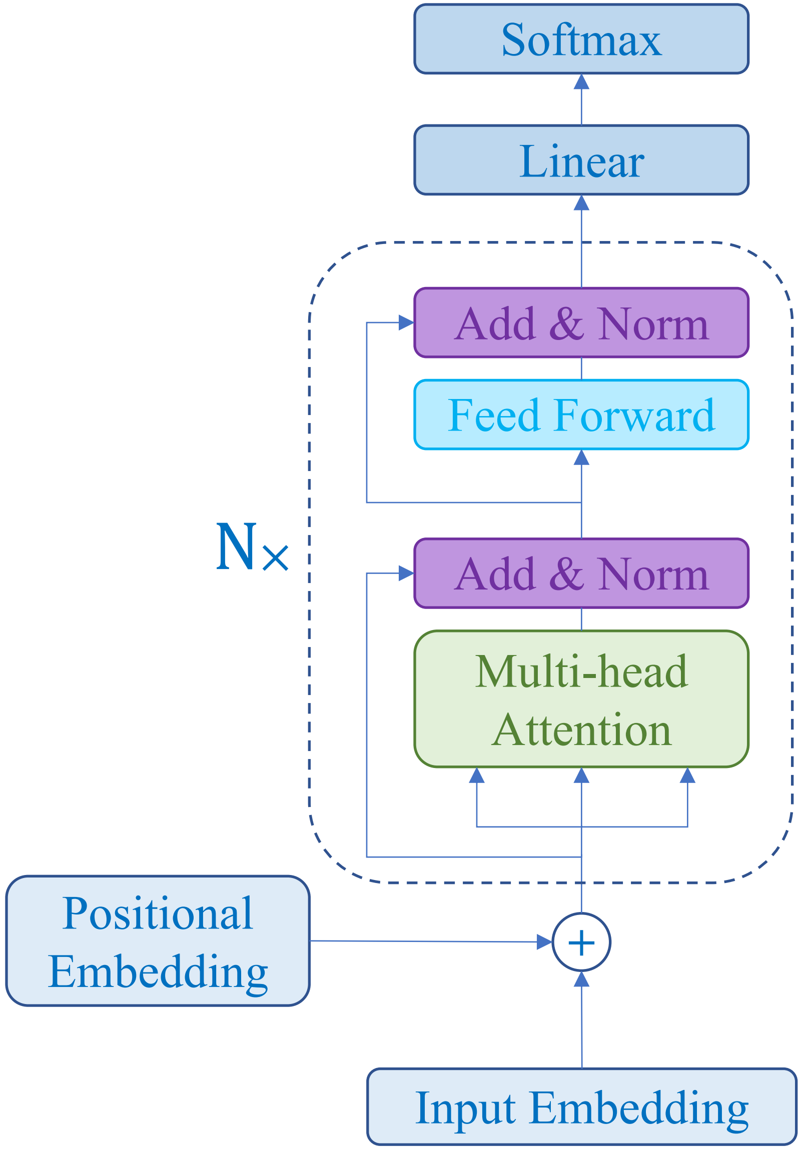


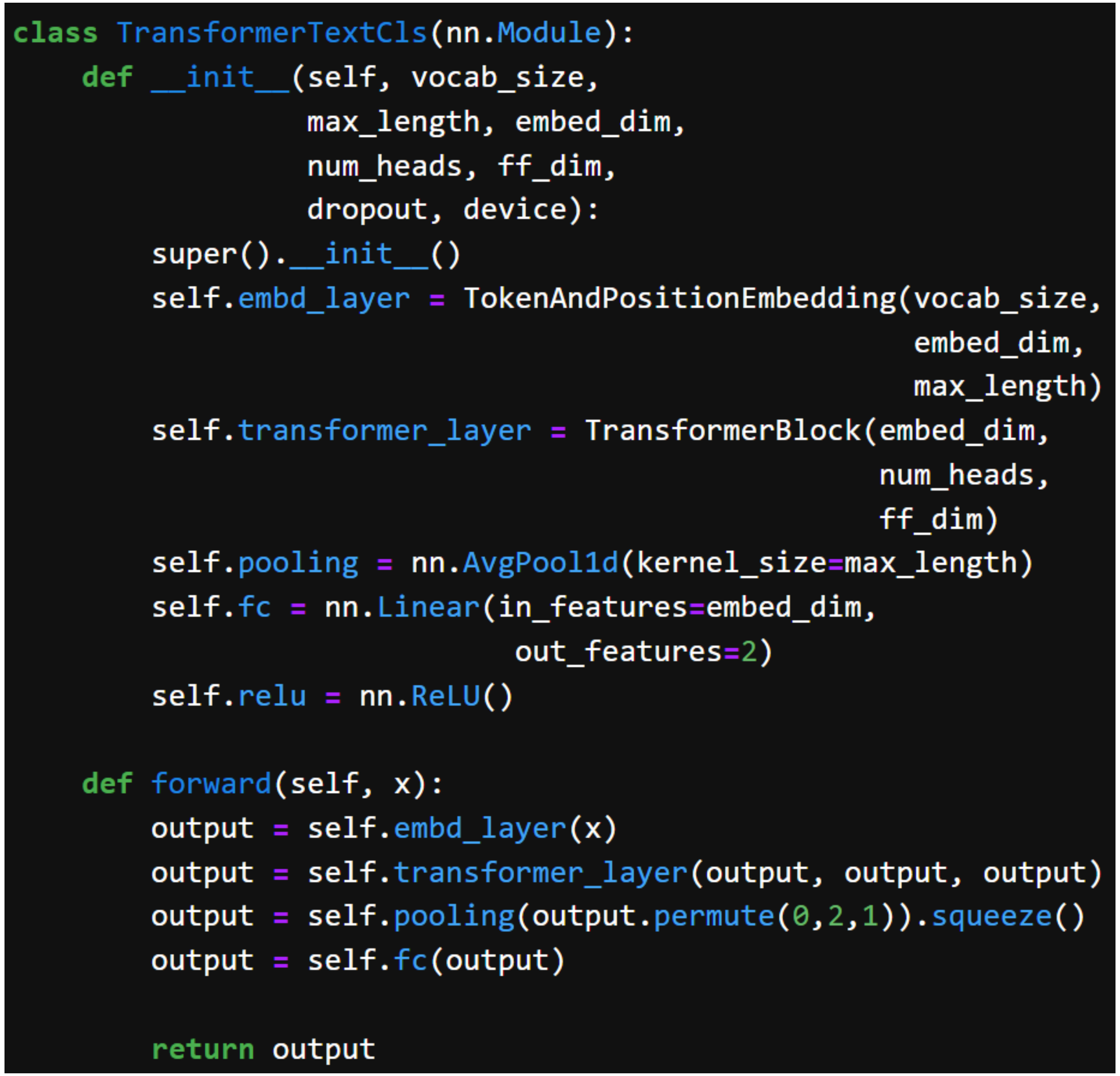
```
================================================================
Layout (type:depth-idx)                        Output Shape
================================================================
├─Embedding: 1-1                               [-1, 500, 128]
├─RNN: 1-2                                     [-1, 500, 64]
├─Linear: 1-3                                  [-1, 2]
================================================================
```

Using LSTM



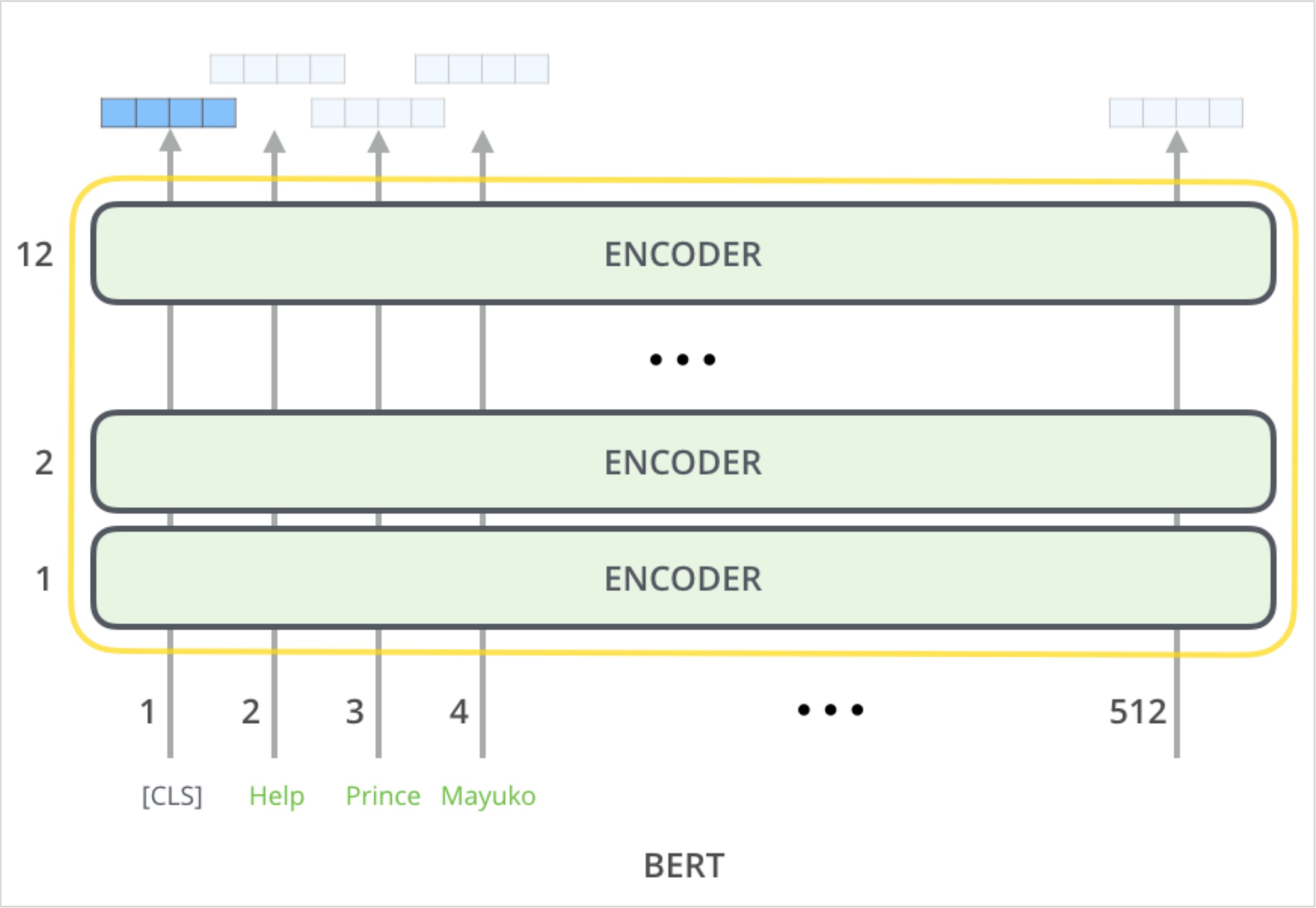
Transformer Models for Text Classification

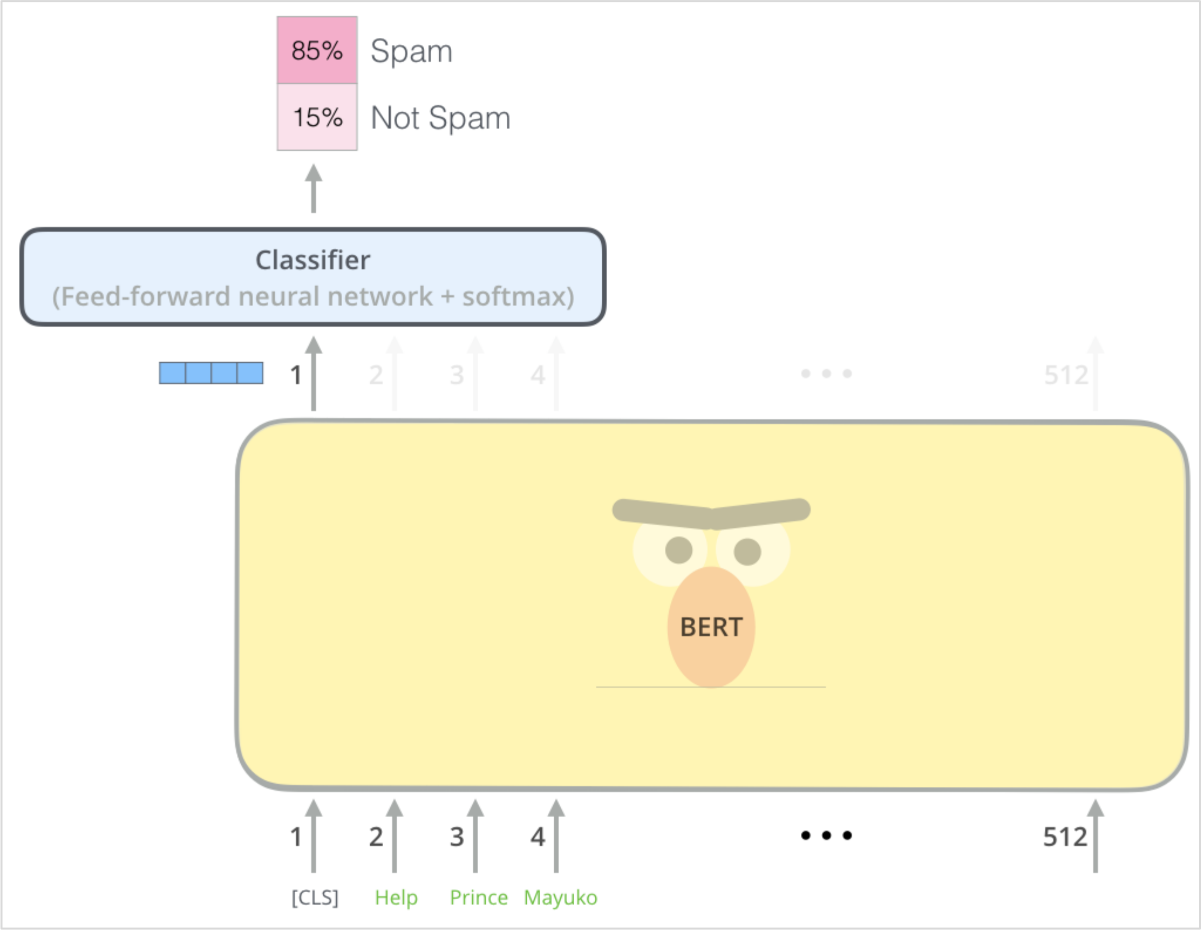




```
class TransformerTextCls(nn.Module):
    def __init__(self, vocab_size,
                 max_length, embed_dim,
                 num_heads, ff_dim,
                 dropout, device):
        super().__init__()
        self.embd_layer = TokenAndPositionEmbedding(vocab_size,
                                                    embed_dim,
                                                    max_length)
        self.transformer_layer = TransformerBlock(embed_dim,
                                                  num_heads,
                                                  ff_dim)
        self.pooling = nn.AvgPool1d(kernel_size=max_length)
        self.fc = nn.Linear(in_features=embed_dim,
                             out_features=2)
        self.relu = nn.ReLU()

    def forward(self, x):
        output = self.embd_layer(x)
        output = self.transformer_layer(output, output, output)
        output = self.pooling(output.permute(0,2,1)).squeeze()
        output = self.fc(output)

        return output
```

Bidirectional Encoder Representations from Transformers





Figures from the BERT paper

Text Classification: Example 1

Step-by-Step Example: Text Classification

Doc	Label
gậy ông đập lưng ông	0
có làm mới có ăn	1

Training data

- negative (0)
- positive (1)

building dictionary

index	word
0	[UNK]
1	[pad]
2	có
3	ông
4	gập
5	làm
6	lưng
7	mới

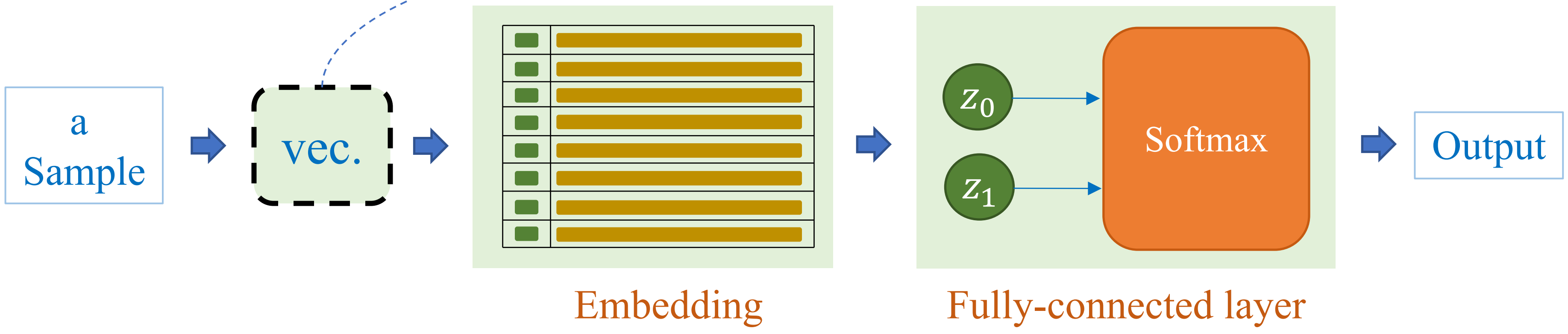
vocab size = 8
sequence length = 5

sample 1

gậy	4
ông	3
đập	0
lưng	6
ông	3

sample 2

có	2
làm	5
mới	7
có	2
ăn	0



Step-by-Step Example: Text Classification

Doc	Label
gậy ông đập lưng ông	0
có làm mới có ăn	1

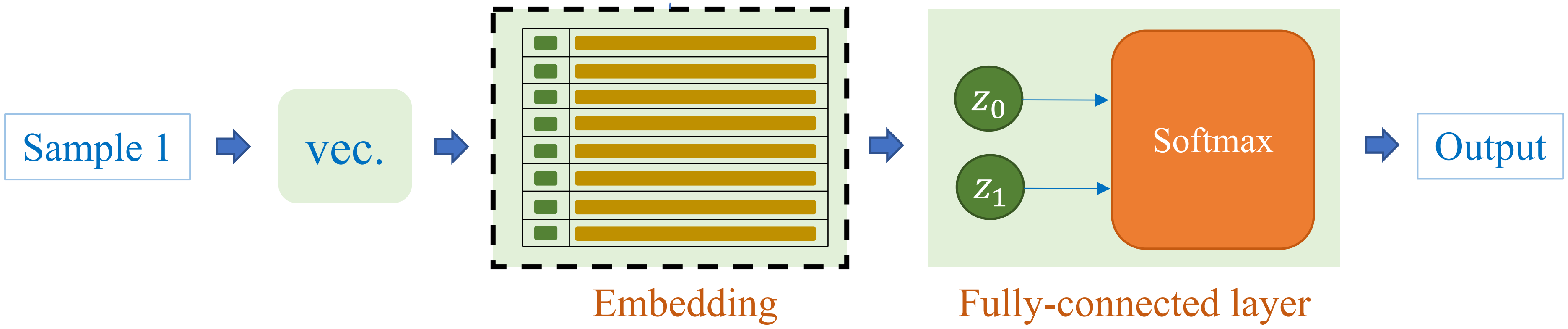
0	[-0.1882, 0.5530]
1	[1.7840, -0.8278]
2	[1.0281, -1.9094]
3	[-1.3083, -0.0987]
4	[0.2293, 1.3255]
5	[0.4058, -0.6624]
6	[0.5582, 0.0786]
7	[0.4309, -1.3067]

Embedding 8x2
(Random initialization)

parameter

gậy	4	[0.2293, 1.3255]
ông	3	[-1.3083, -0.0987]
đập	0	[-0.1882, 0.5530]
lưng	6	[0.5582, 0.0786]
ông	3	[-1.3083, -0.0987]

sample 1

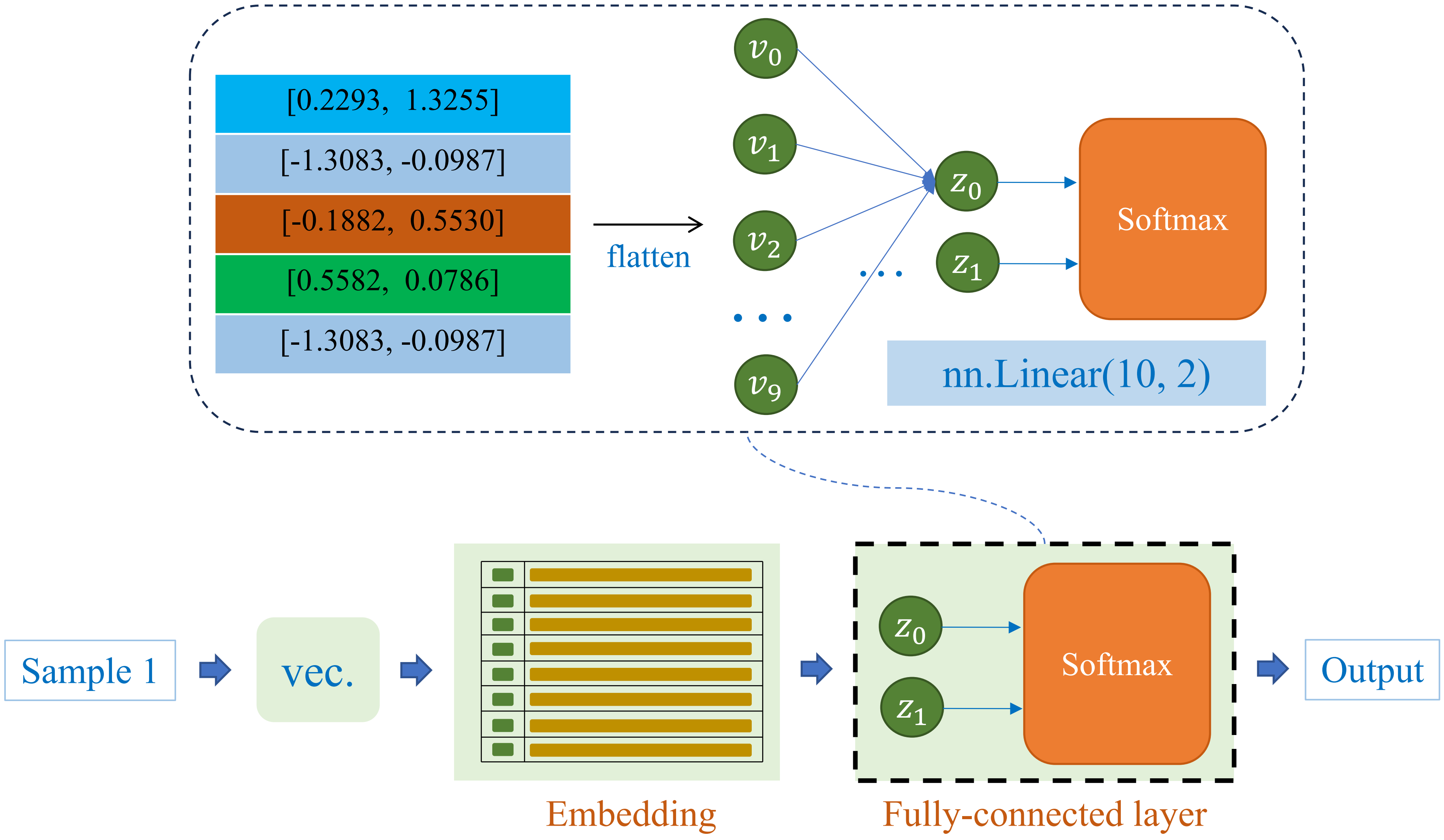


Step-by-Step Example: Text Classification

Doc	Label
gậy ông đập lưng ông	0
có làm mới có ăn	1

0	[-0.1882, 0.5530]
1	[1.7840, -0.8278]
2	[1.0281, -1.9094]
3	[-1.3083, -0.0987]
4	[0.2293, 1.3255]
5	[0.4058, -0.6624]
6	[0.5582, 0.0786]
7	[0.4309, -1.3067]

Embedding



[0.2293, 1.3255]

[-1.3083, -0.0987]

[-0.1882, 0.5530]

[0.5582, 0.0786]

[-1.3083, -0.0987]

flatten

nn.Linear(10, 2)

Embedding

Fully-connected layer

Example
Text Classification

Doc	Label
gậy ông đập lưng ông	0
có làm mới có ăn	1

W

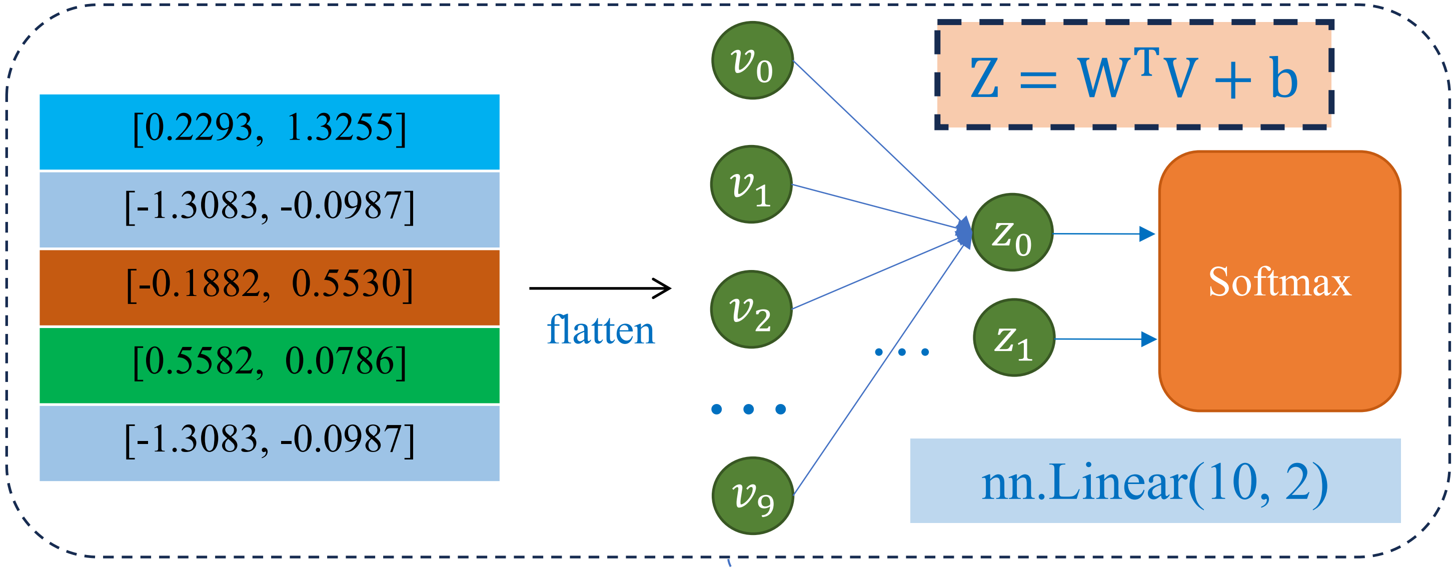
[0.2108, -0.0074, 0.2760, 0.2325, -0.0518, -0.1876, 0.0194, 0.0378, 0.0210, 0.2982]

[0.0284, 0.2968, -0.0260, 0.1251, -0.0282, 0.0175, -0.1817, 0.2483, 0.2338, 0.2985]

b [-0.3049, 0.1028]

z [-0.7875, 0.1221]

$\hat{y}$ [0.28, 0.71]



[0.2293, 1.3255]

[-1.3083, -0.0987]

[-0.1882, 0.5530]

[0.5582, 0.0786]

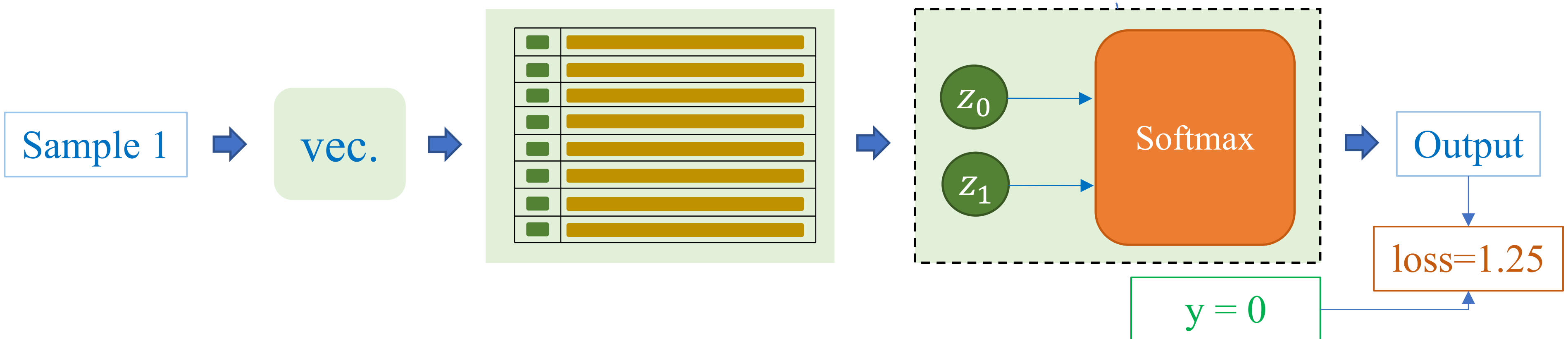
[-1.3083, -0.0987]

$Z = W^T V + b$

nn.Linear(10, 2)

0	[-0.1882, 0.5530]
1	[1.7840, -0.8278]
2	[1.0281, -1.9094]
3	[-1.3083, -0.0987]
4	[0.2293, 1.3255]
5	[0.4058, -0.6624]
6	[0.5582, 0.0786]
7	[0.4309, -1.3067]

Embedding



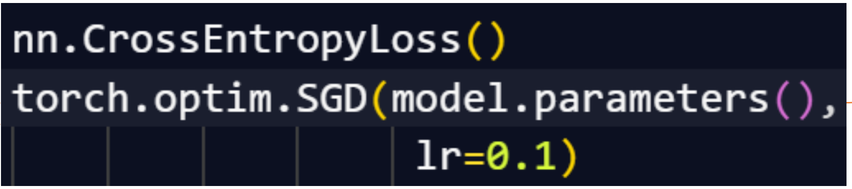
Example
Text Classification

Doc	Label
gậy ông đập lưng ông	0
có làm mới có ăn	1

W

[0.3108, 0.0926, 0.1760, 0.1325, -0.1518, -0.0876, 0.1194, 0.1378, -0.0790, 0.1982]

[-0.0716, 0.1968, 0.0740, 0.2251, 0.0718, -0.0825, -0.2817, 0.1483, 0.3338, 0.3985]



```
nn.CrossEntropyLoss()
torch.optim.SGD(model.parameters(),
                lr=0.1)
```

b

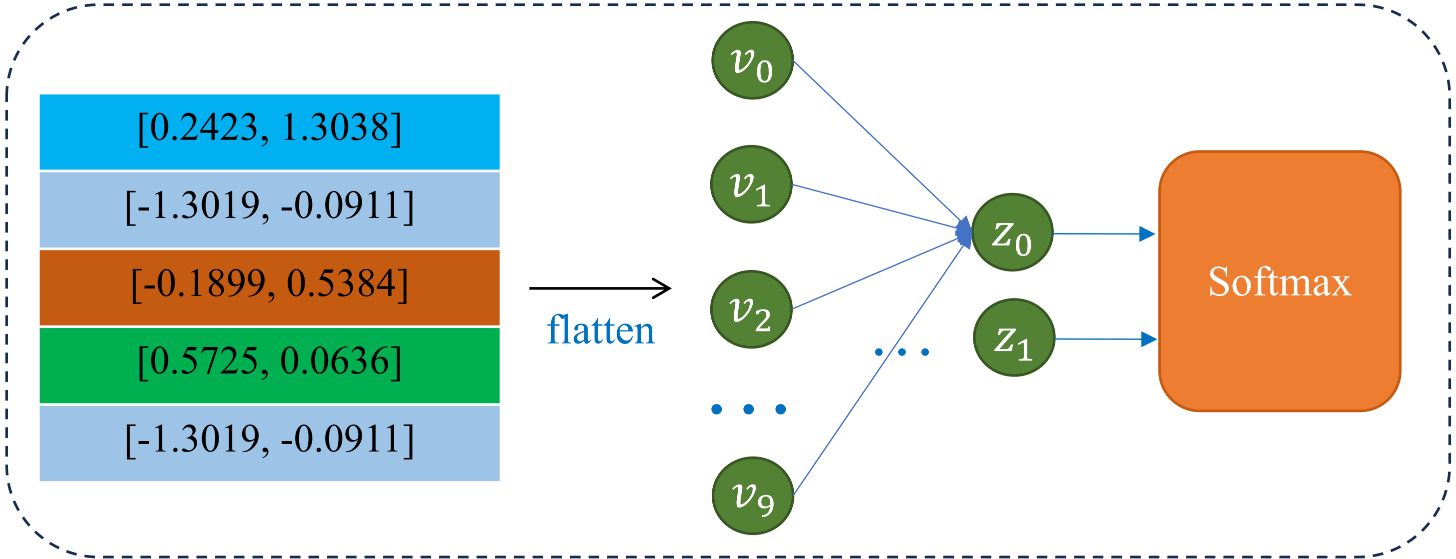
[-0.2049, 0.0028]

$$\theta_t = \theta_{t-1} - \eta \nabla_\theta L$$

update

0	[-0.1899, 0.5384]
1	[1.7840, -0.8278]
2	[1.0281, -1.9094]
3	[-1.3019, -0.0911]
4	[0.2423, 1.3038]
5	[0.4058, -0.6624]
6	[0.5725, 0.0636]
7	[0.4309, -1.3067]

Embedding



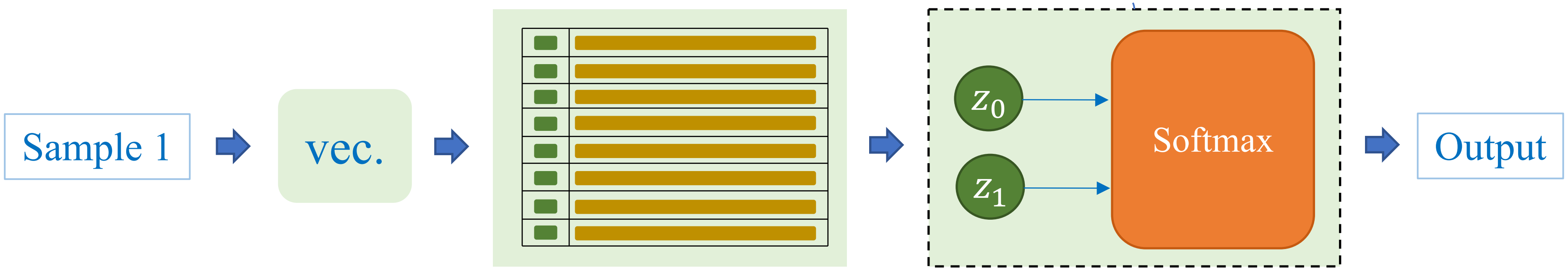
[0.2423, 1.3038]

[-1.3019, -0.0911]

[-0.1899, 0.5384]

[0.5725, 0.0636]

[-1.3019, -0.0911]



Example
Text Classification

Doc	Label
gậy ông đập lưng ông	0
có làm mới có ăn	1

W

[0.3108, 0.0926, 0.1760, 0.1325, -0.1518, -0.0876, 0.1194, 0.1378, -0.0790, 0.1982]

[-0.0716, 0.1968, 0.0740, 0.2251, 0.0718, -0.0825, -0.2817, 0.1483, 0.3338, 0.3985]

b [-0.2049, 0.0028]

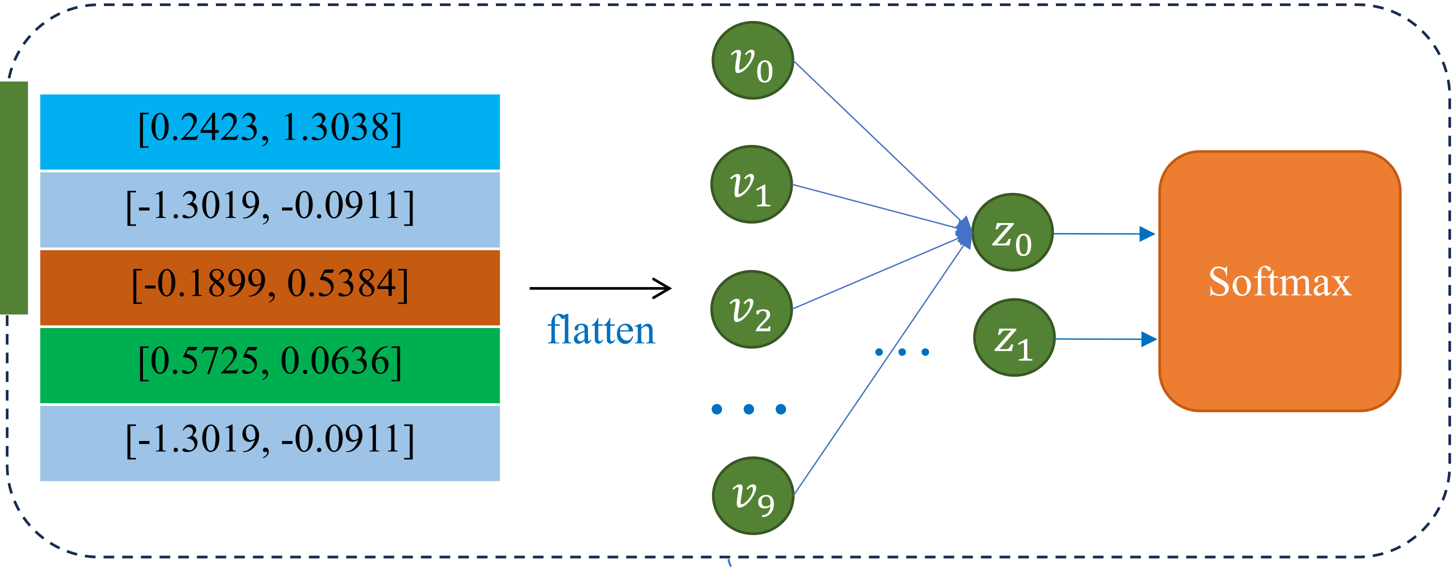
z [-0.0261, -0.5182]

$\hat{y}$ ↑[0.63, 0.37]↓

- Loss reduces
- $\hat{y}_0$ increases
- $\hat{y}_1$ reduces

0	[-0.1899, 0.5384]
1	[1.7840, -0.8278]
2	[1.0281, -1.9094]
3	[-1.3019, -0.0911]
4	[0.2423, 1.3038]
5	[0.4058, -0.6624]
6	[0.5725, 0.0636]
7	[0.4309, -1.3067]

Embedding



[0.2423, 1.3038]

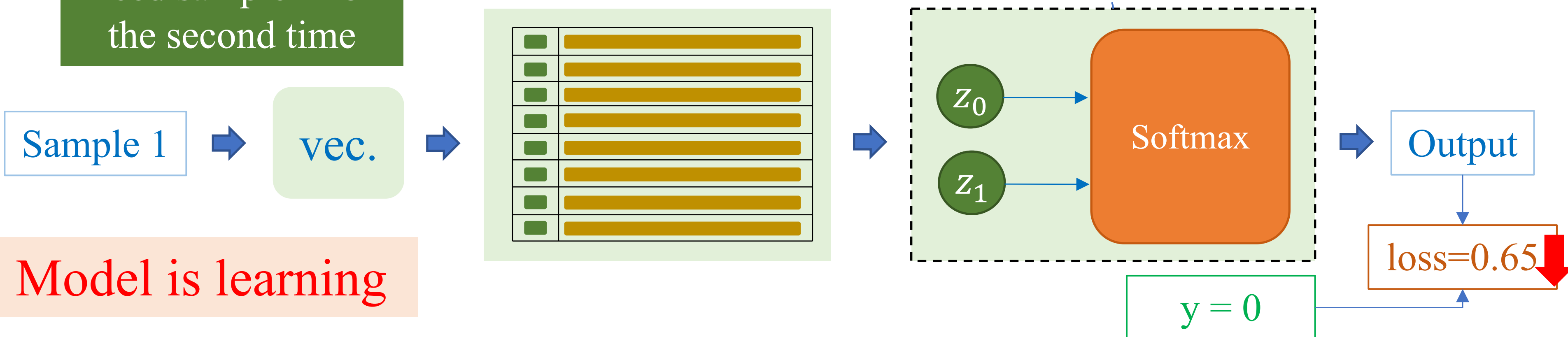
[-1.3019, -0.0911]

[-0.1899, 0.5384]

[0.5725, 0.0636]

[-1.3019, -0.0911]

Feed sample 1 for the second time



Model is learning

Text Classification: Example 2

Doc	Label
gậy ông đập lưng ông	0
có làm mới có ăn	1

building dictionary →

vocab size = 8
sequence length = 5

index	word
0	[UNK]
1	[pad]
2	có
3	ông
4	gậy
5	làm
6	lưng
7	mới

Dictionary

0	[-0.1, 0.5]
1	[1.7, -0.8]
2	[1.0, -1.9]
3	[-1.3, -0.1]
4	[0.2, 1.3]
5	[0.4, -0.6]
6	[0.5, 0.1]
7	[0.4, -1.3]

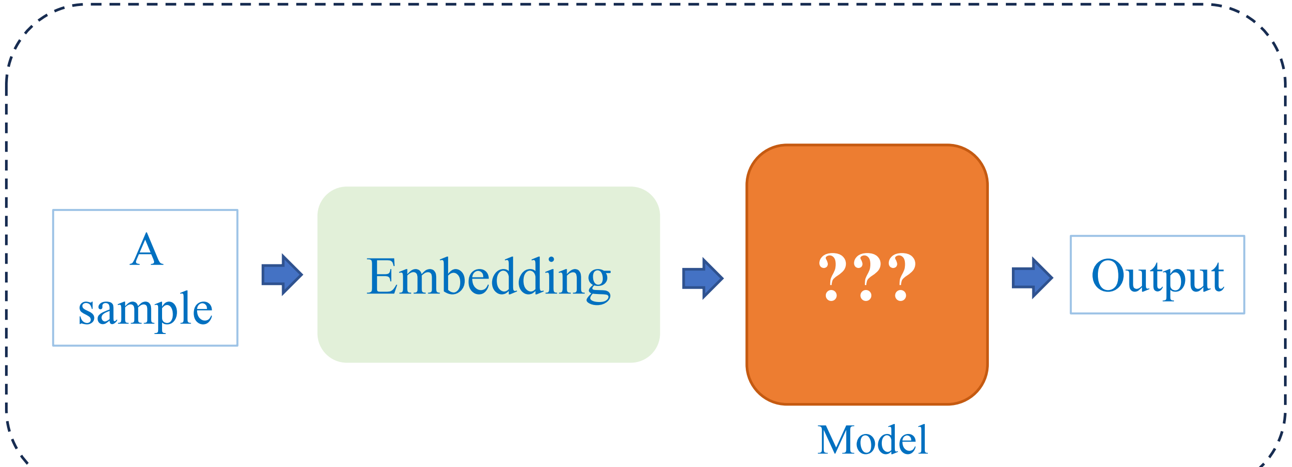
Embedding

gậy		4		[0.2, 1.3]
ông		3		[-1.3, -0.1]
đập	→	0	→	[-0.1, 0.5]
lưng		6		[0.5, 0.1]
ông		3		[-1.3, -0.1]

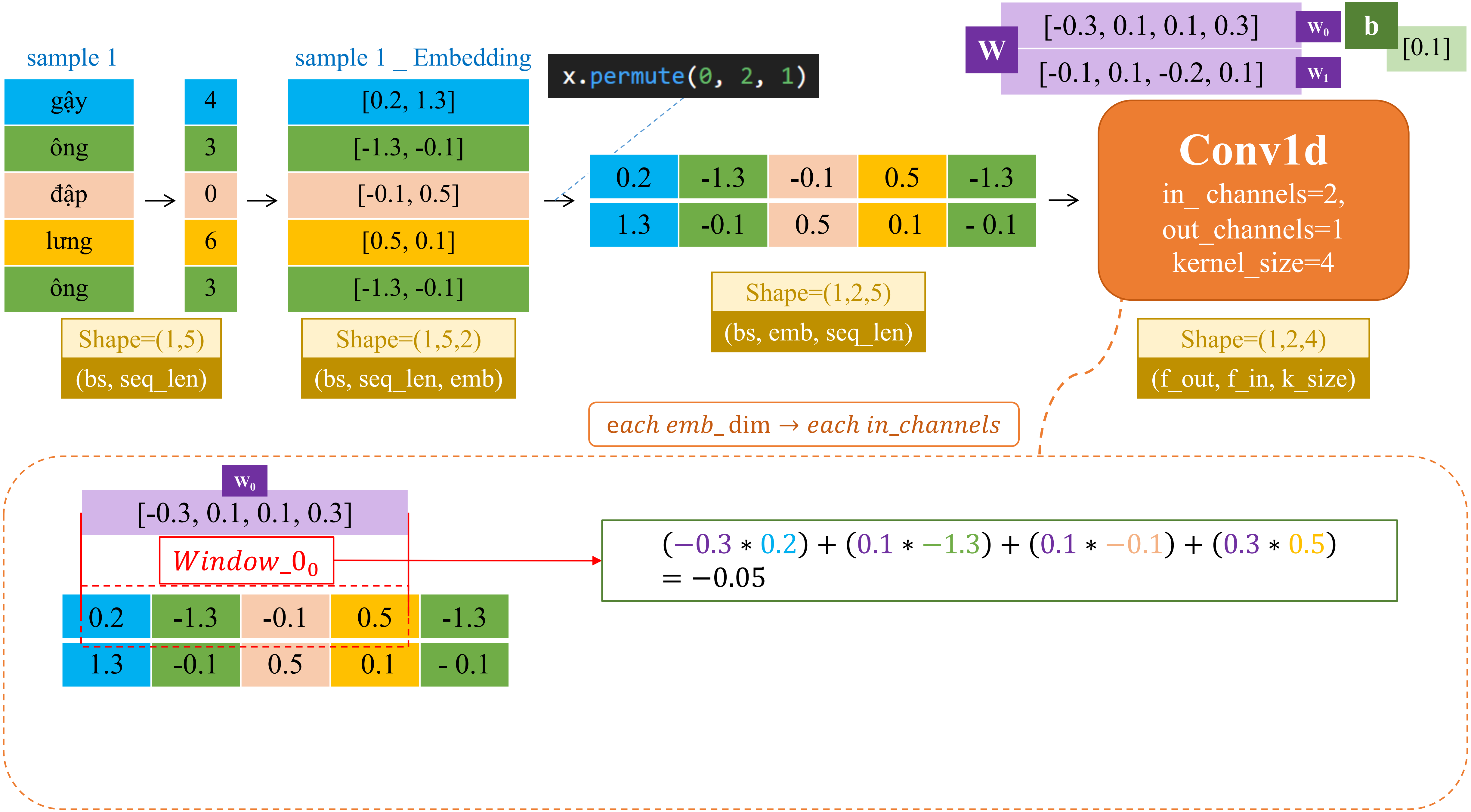
sample 1

sample 1 _ Embedding

Vectorization and Embedding



Model Pipeline



sample 1		sample 1 _ Embedding
gậy	4	[0.2, 1.3]
ông	3	[-1.3, -0.1]
đập	0	[-0.1, 0.5]
lưng	6	[0.5, 0.1]
ông	3	[-1.3, -0.1]

Shape=(1,5)
(bs, seq_len)

Shape=(1,5,2)
(bs, seq_len, emb)

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

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

Shape=(1,2,5)
(bs, emb, seq_len)

W		
	[-0.3, 0.1, 0.1, 0.3]	w_0
	[-0.1, 0.1, -0.2, 0.1]	w_1

b
[0.1]

Conv1d
in_ channels=2,
out_channels=1
kernel_size=4

Shape=(1,2,4)
(f_out, f_in, k_size)

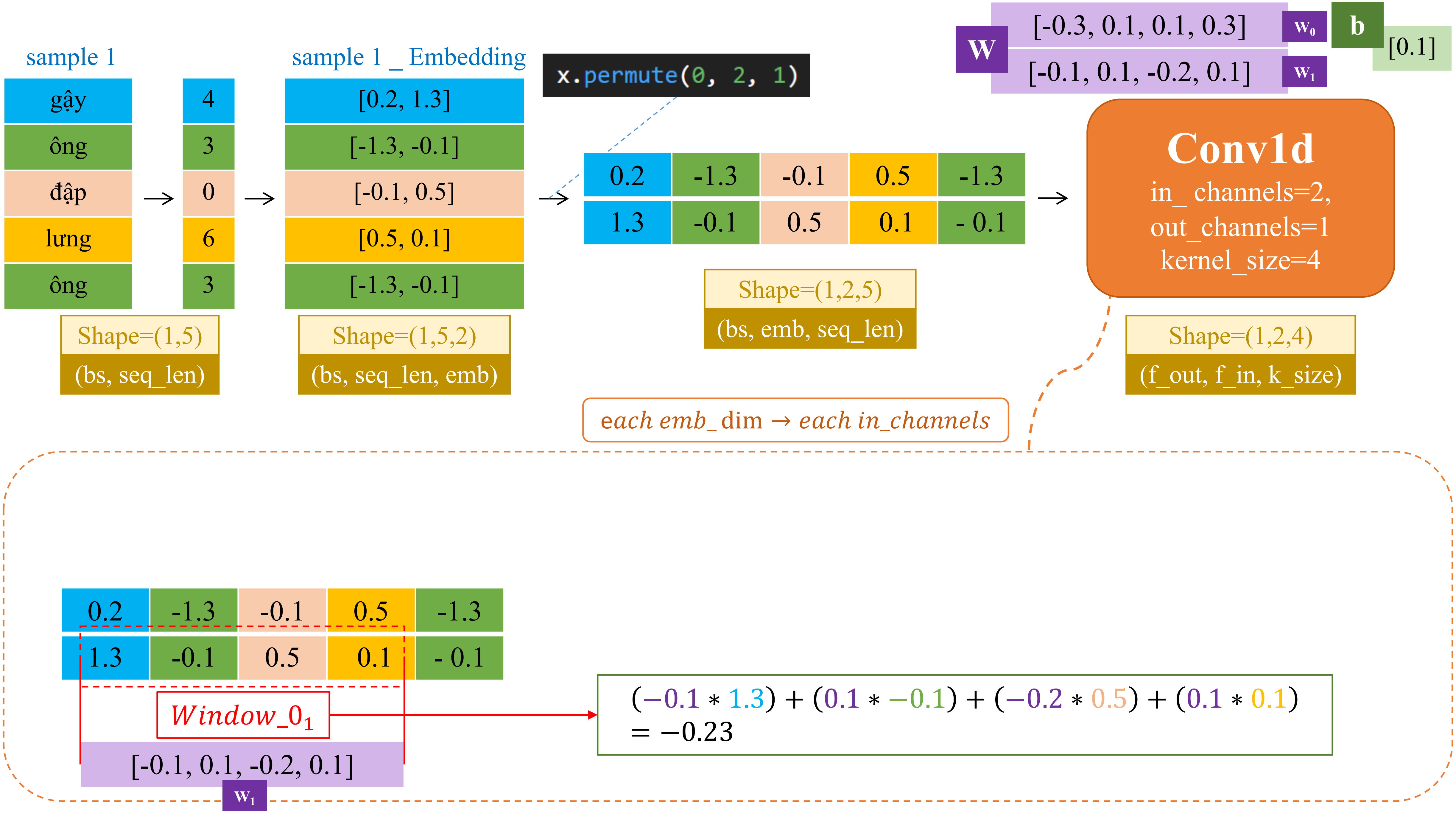
each emb_dim → each in_channels

w_0
[-0.3, 0.1, 0.1, 0.3]

$Window\_0_0$

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

$$(-0.3 * 0.2) + (0.1 * -1.3) + (0.1 * -0.1) + (0.3 * 0.5)$$
$$= -0.05$$



sample 1

gậy	→	4
ông		3
đập		0
lưng		6
ông		3

Shape=(1,5)
(bs, seq_len)

sample 1 _ Embedding

[0.2, 1.3]
[-1.3, -0.1]
[-0.1, 0.5]
[0.5, 0.1]
[-1.3, -0.1]

Shape=(1,5,2)
(bs, seq_len, emb)

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

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

Shape=(1,2,5)
(bs, emb, seq_len)

each emb_ dim → each in_channels

W	[-0.3, 0.1, 0.1, 0.3]	w_0
	[-0.1, 0.1, -0.2, 0.1]	w_1

b [0.1]

Conv1d
in_ channels=2,
out_channels=1
kernel_size=4

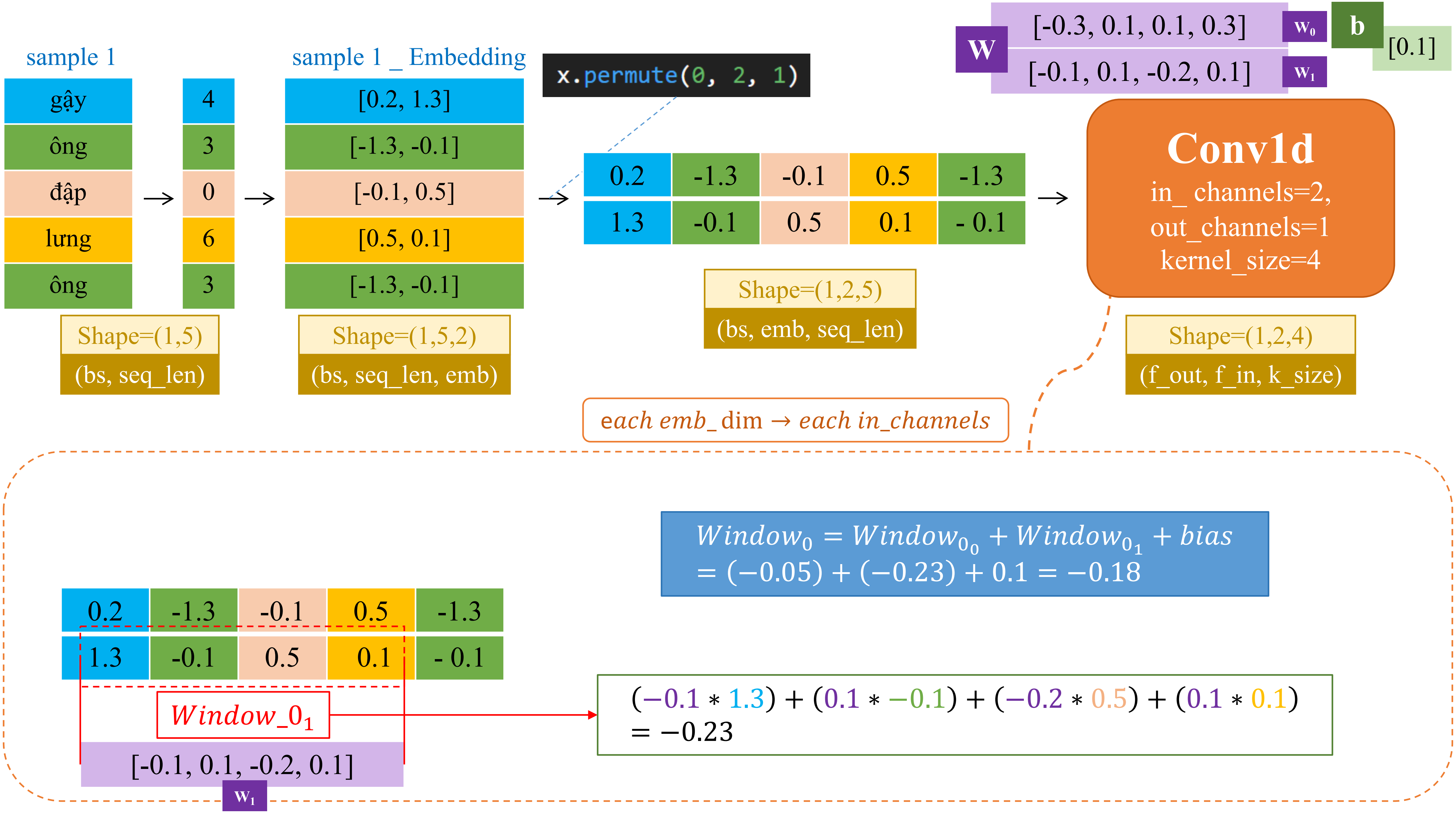
Shape=(1,2,4)
(f_out, f_in, k_size)

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

$Window\_0_1$

[-0.1, 0.1, -0.2, 0.1]
w_1

$$(-0.1 * 1.3) + (0.1 * -0.1) + (-0.2 * 0.5) + (0.1 * 0.1) = -0.23$$



sample 1		sample 1 _ Embedding
gậy	4	[0.2, 1.3]
ông	3	[-1.3, -0.1]
đập	0	[-0.1, 0.5]
lưng	6	[0.5, 0.1]
ông	3	[-1.3, -0.1]

Shape=(1,5)
(bs, seq_len)

Shape=(1,5,2)
(bs, seq_len, emb)

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

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

Shape=(1,2,5)
(bs, emb, seq_len)

each emb_ dim → each in_channels

W	[-0.3, 0.1, 0.1, 0.3]	w_0	b [0.1]
	[-0.1, 0.1, -0.2, 0.1]	w_1	

Conv1d
in_ channels=2,
out_channels=1
kernel_size=4

Shape=(1,2,4)
(f_out, f_in, k_size)

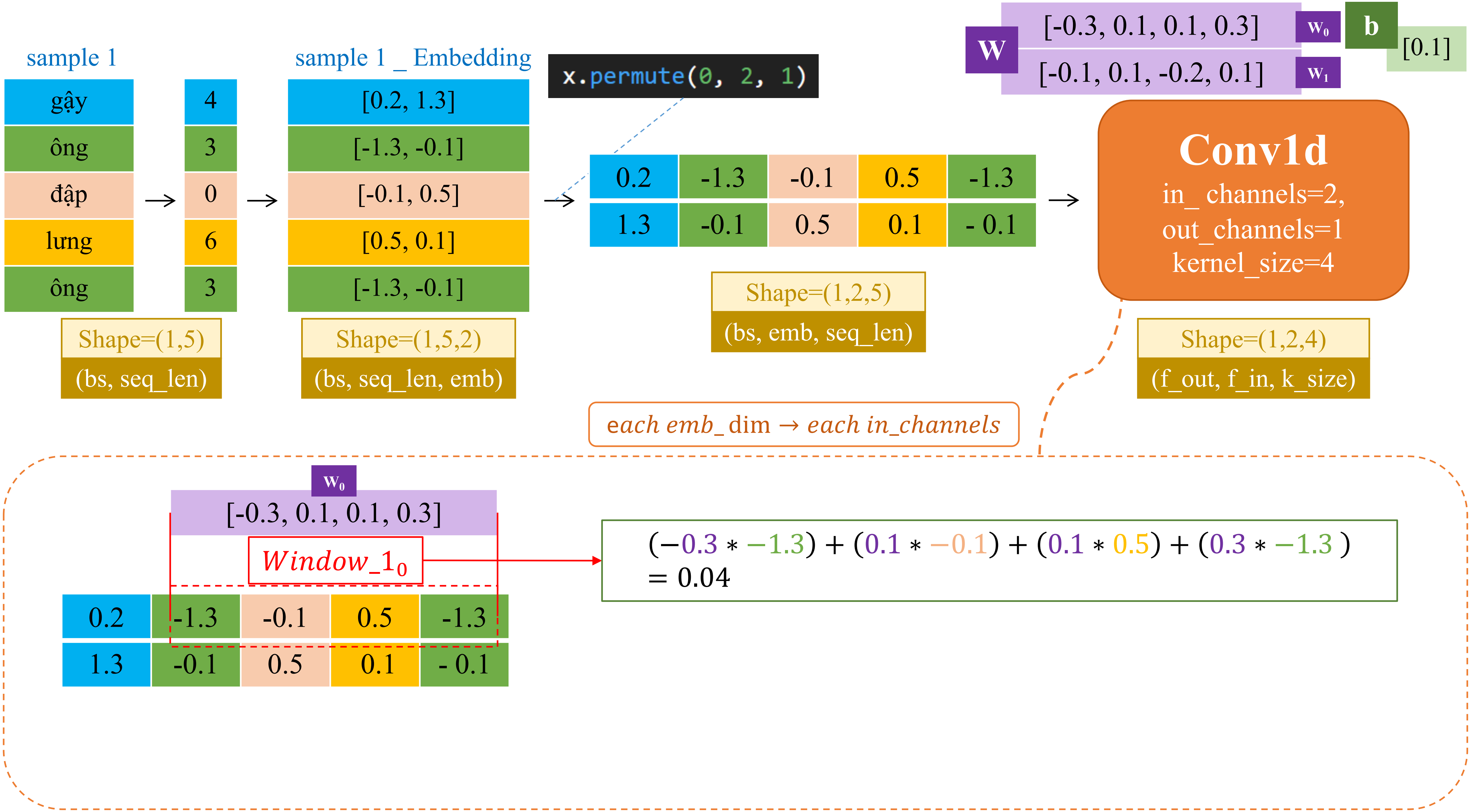
0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

$Window\_0_1$

[-0.1, 0.1, -0.2, 0.1] w_1

$$(-0.1 * 1.3) + (0.1 * -0.1) + (-0.2 * 0.5) + (0.1 * 0.1)$$
$$= -0.23$$

$$Window_0 = Window_{0_0} + Window_{0_1} + bias$$
$$= (-0.05) + (-0.23) + 0.1 = -0.18$$



sample 1		sample 1 _ Embedding
gây	4	[0.2, 1.3]
ông	3	[-1.3, -0.1]
đập	0	[-0.1, 0.5]
lưng	6	[0.5, 0.1]
ông	3	[-1.3, -0.1]

Shape=(1,5)
(bs, seq_len)

Shape=(1,5,2)
(bs, seq_len, emb)

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

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

Shape=(1,2,5)
(bs, emb, seq_len)

W			b
	[-0.3, 0.1, 0.1, 0.3]	w_0	[0.1]
	[-0.1, 0.1, -0.2, 0.1]	w_1	

Conv1d
in_ channels=2,
out_channels=1
kernel_size=4

Shape=(1,2,4)
(f_out, f_in, k_size)

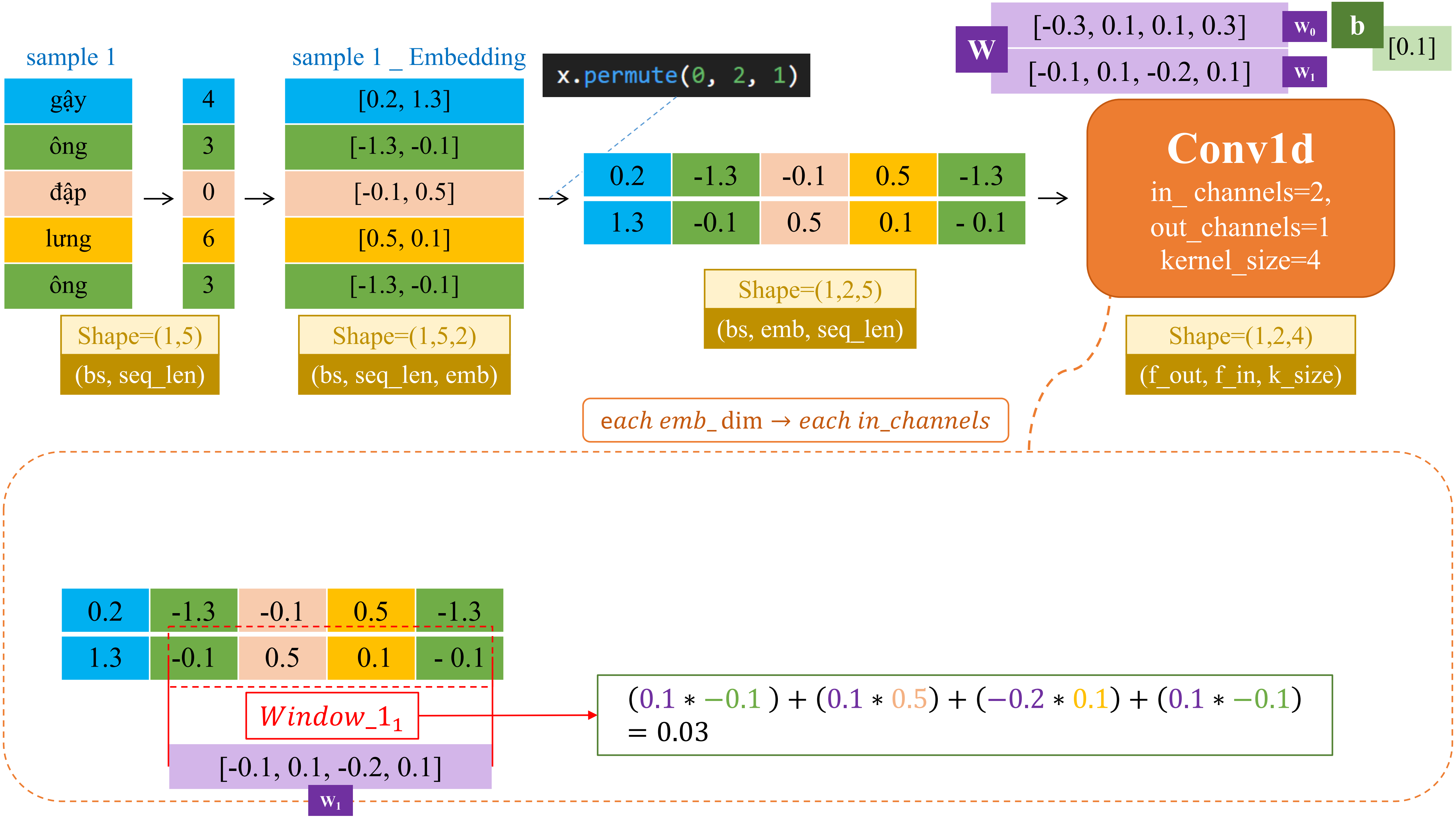
each emb_ dim → each in_channels

w_0
[-0.3, 0.1, 0.1, 0.3]

$Window\_1_0$

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

$$(-0.3 * -1.3) + (0.1 * -0.1) + (0.1 * 0.5) + (0.3 * -1.3\ ) = 0.04$$



sample 1

gậy	4
ông	3
đập	0
lưng	6
ông	3

Shape=(1,5)
(bs, seq_len)

sample 1 _ Embedding

[0.2, 1.3]
[-1.3, -0.1]
[-0.1, 0.5]
[0.5, 0.1]
[-1.3, -0.1]

Shape=(1,5,2)
(bs, seq_len, emb)

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

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

Shape=(1,2,5)
(bs, emb, seq_len)

each emb_dim → each in_channels

W

w₀ [-0.3, 0.1, 0.1, 0.3]
w₁ [-0.1, 0.1, -0.2, 0.1]

b [0.1]

Conv1d
in_ channels=2,
out_channels=1
kernel_size=4

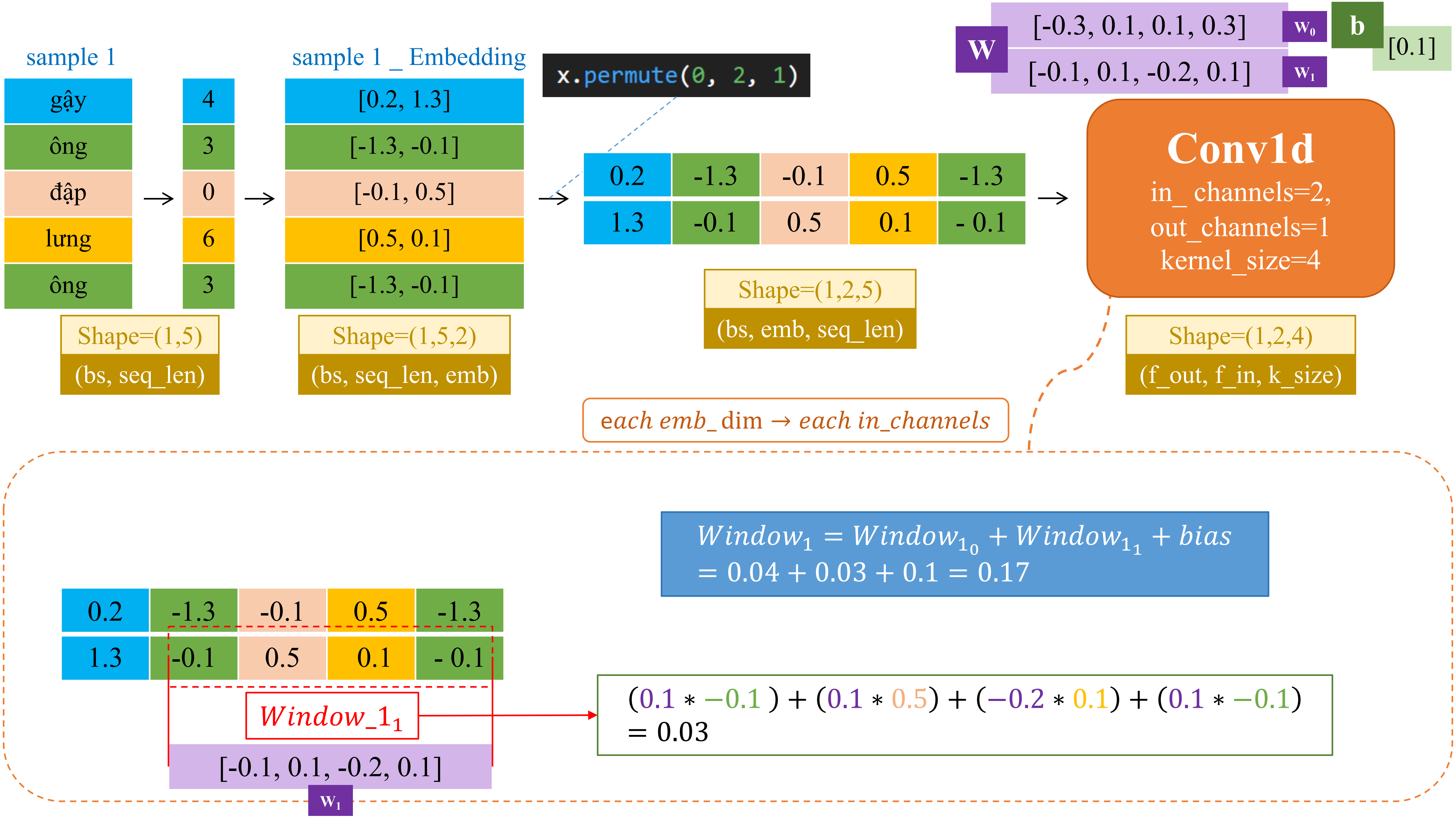
Shape=(1,2,4)
(f_out, f_in, k_size)

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

$Window\_1_1$

[-0.1, 0.1, -0.2, 0.1]
w₁

$$(0.1 * -0.1\ ) + (0.1 * 0.5) + (-0.2 * 0.1) + (0.1 * -0.1)$$
$$= 0.03$$



sample 1		sample 1 _ Embedding
gậy	4	[0.2, 1.3]
ông	3	[-1.3, -0.1]
đập	0	[-0.1, 0.5]
lưng	6	[0.5, 0.1]
ông	3	[-1.3, -0.1]
Shape=(1,5) (bs, seq_len)		Shape=(1,5,2) (bs, seq_len, emb)

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

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

Shape=(1,2,5)
(bs, emb, seq_len)

each emb_ dim → each in_channels

Conv1d
in_ channels=2,
out_channels=1
kernel_size=4

W		b
[-0.3, 0.1, 0.1, 0.3]	w₀	[0.1]
[-0.1, 0.1, -0.2, 0.1]	w₁	

Shape=(1,2,4)
(f_out, f_in, k_size)

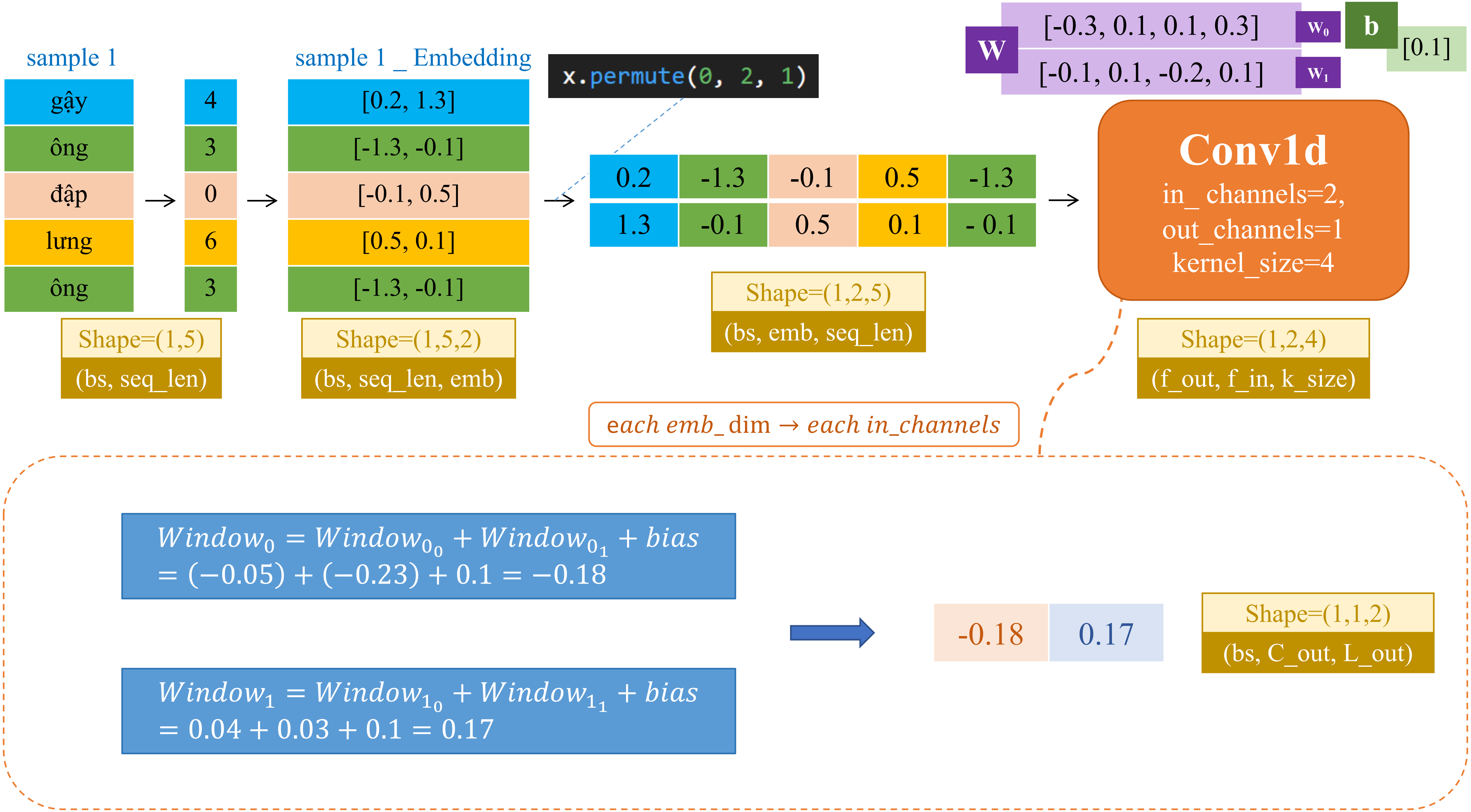
0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

$Window\_1_1$

[-0.1, 0.1, -0.2, 0.1]
w₁

$$(0.1 * -0.1\ ) + (0.1 * 0.5) + (-0.2 * 0.1) + (0.1 * -0.1) = 0.03$$

$$Window_1 = Window_{1_0} + Window_{1_1} + bias = 0.04 + 0.03 + 0.1 = 0.17$$



sample 1		sample 1 _ Embedding
gậy	4	[0.2, 1.3]
ông	3	[-1.3, -0.1]
đập	0	[-0.1, 0.5]
lưng	6	[0.5, 0.1]
ông	3	[-1.3, -0.1]
Shape=(1,5) (bs, seq_len)		Shape=(1,5,2) (bs, seq_len, emb)

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

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

Shape=(1,2,5)
(bs, emb, seq_len)

W		b
[-0.3, 0.1, 0.1, 0.3]	w_0	[0.1]
[-0.1, 0.1, -0.2, 0.1]	w_1	

Conv1d
in_ channels=2,
out_channels=1
kernel_size=4

Shape=(1,2,4)
(f_out, f_in, k_size)

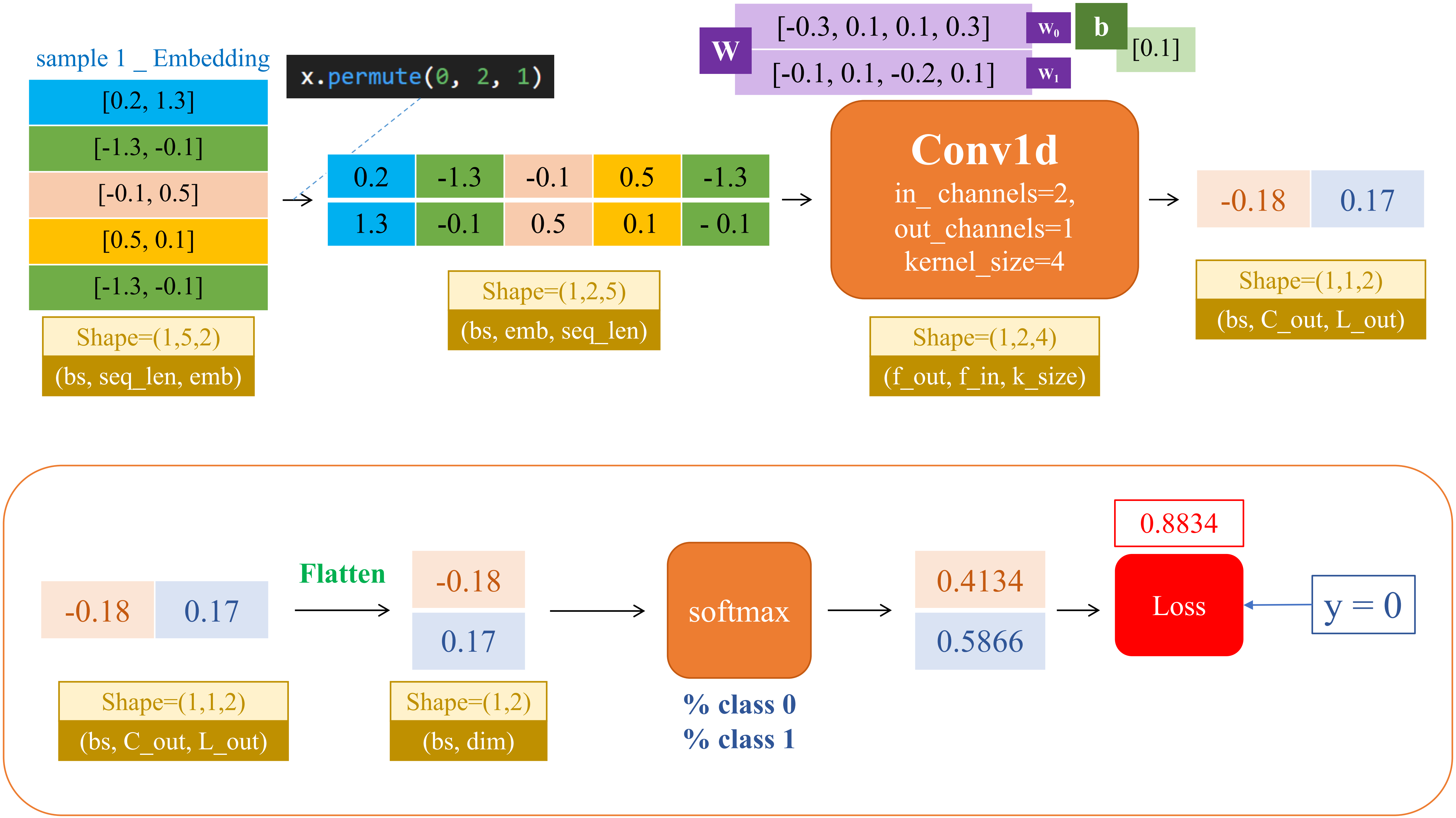
each emb_ dim → each in_channels

$$Window_0 = Window_{0_0} + Window_{0_1} + bias$$
$$= (-0.05) + (-0.23) + 0.1 = -0.18$$

$$Window_1 = Window_{1_0} + Window_{1_1} + bias$$
$$= 0.04 + 0.03 + 0.1 = 0.17$$

-0.18	0.17

Shape=(1,1,2)
(bs, C_out, L_out)



sample 1 _ Embedding

[0.2, 1.3]
[-1.3, -0.1]
[-0.1, 0.5]
[0.5, 0.1]
[-1.3, -0.1]

Shape=(1,5,2)

(bs, seq_len, emb)

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

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

Shape=(1,2,5)

(bs, emb, seq_len)

W

[-0.3, 0.1, 0.1, 0.3] w_0

[-0.1, 0.1, -0.2, 0.1] w_1

b [0.1]

Conv1d

in_ channels=2,
out_channels=1
kernel_size=4

Shape=(1,2,4)

(f_out, f_in, k_size)

-0.18	0.17

Shape=(1,1,2)

(bs, C_out, L_out)

-0.18	0.17

Shape=(1,1,2)

(bs, C_out, L_out)

Flatten

-0.18
0.17

Shape=(1,2)

(bs, dim)

softmax

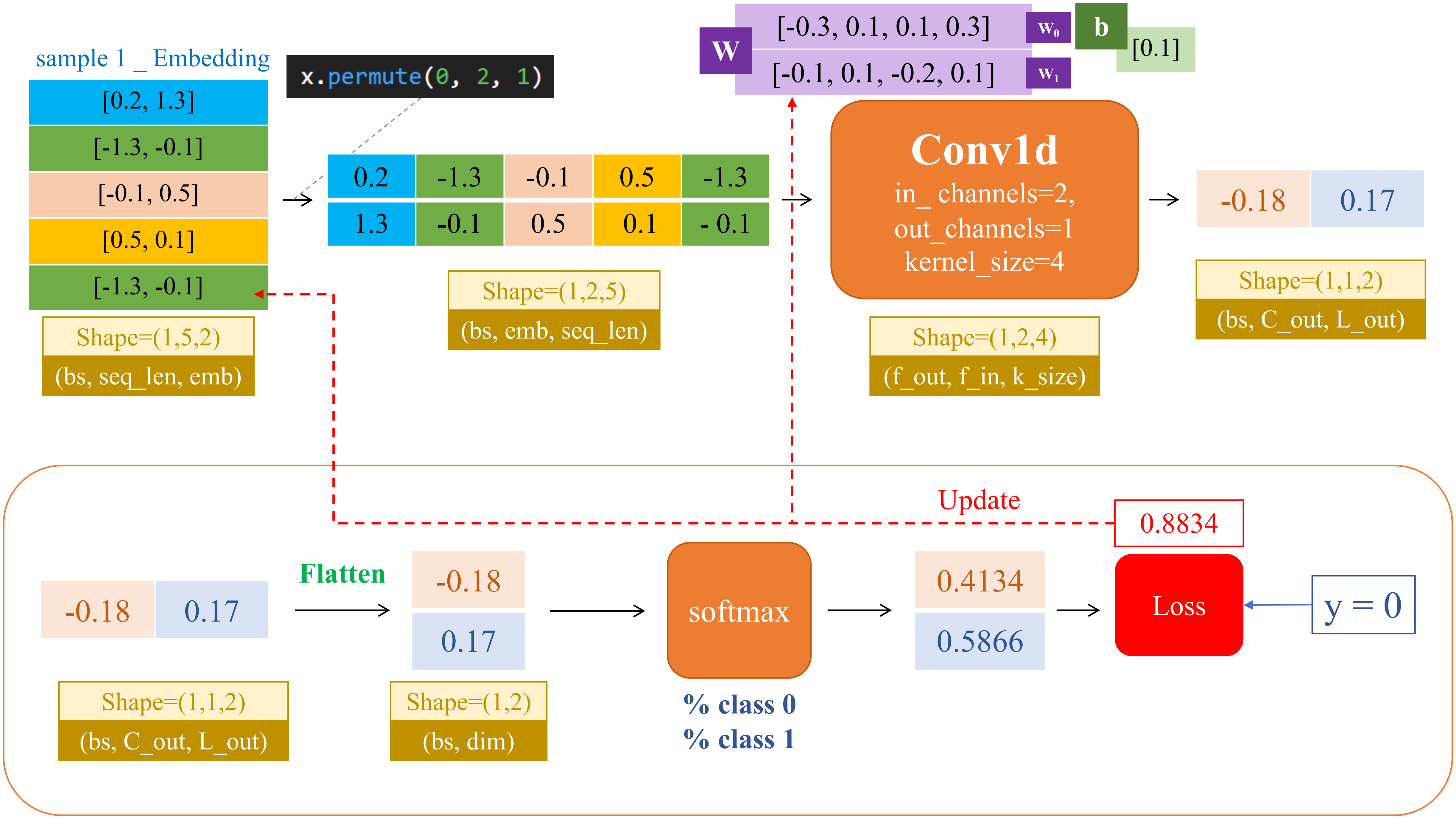
% class 0
% class 1

0.4134
0.5866

0.8834

Loss

y = 0



sample 1 _ Embedding

[0.2, 1.3]

[-1.3, -0.1]

[-0.1, 0.5]

[0.5, 0.1]

[-1.3, -0.1]

Shape=(1,5,2)

(bs, seq_len, emb)

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

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

Shape=(1,2,5)

(bs, emb, seq_len)

W

[-0.3, 0.1, 0.1, 0.3] w_0

[-0.1, 0.1, -0.2, 0.1] w_1

b

[0.1]

Conv1d

in_ channels=2,

out_channels=1

kernel_size=4

Shape=(1,2,4)

(f_out, f_in, k_size)

-0.18	0.17

Shape=(1,1,2)

(bs, C_out, L_out)

Update

0.8834

-0.18	0.17

Shape=(1,1,2)

(bs, C_out, L_out)

Flatten

-0.18

0.17

Shape=(1,2)

(bs, dim)

softmax

% class 0

% class 1

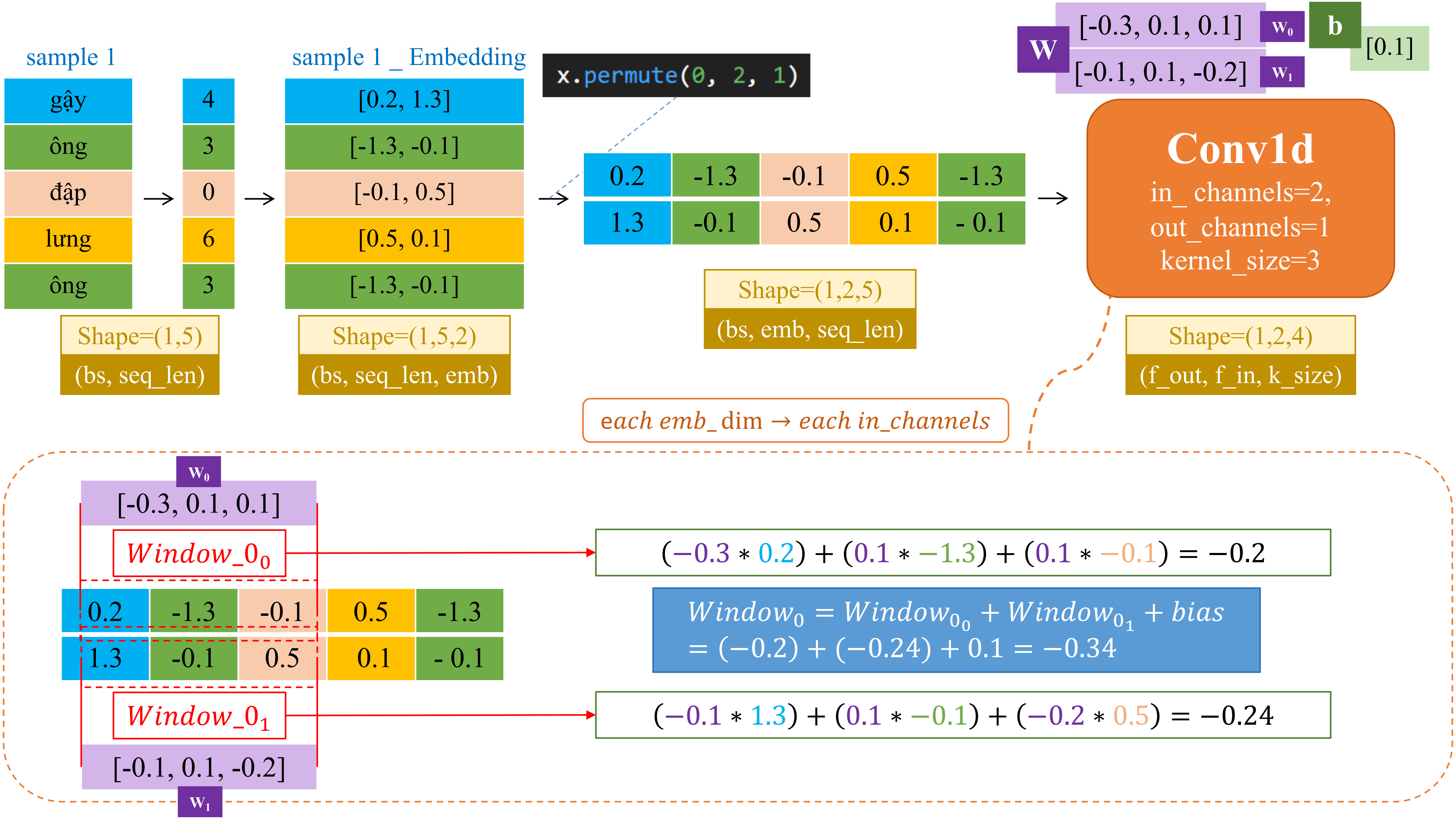
0.4134

0.5866

Loss

y = 0

Text Classification: Example 3



sample 1

gậy
ông
đập
lưng
ông

→

4
3
0
6
3

Shape=(1,5)

(bs, seq_len)

→

sample 1 _ Embedding

[0.2, 1.3]
[-1.3, -0.1]
[-0.1, 0.5]
[0.5, 0.1]
[-1.3, -0.1]

Shape=(1,5,2)

(bs, seq_len, emb)

→

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

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

Shape=(1,2,5)

(bs, emb, seq_len)

→

Conv1d

in_ channels=2,

out_channels=1

kernel_size=3

W

[-0.3, 0.1, 0.1] w_0

[-0.1, 0.1, -0.2] w_1

b [0.1]

Shape=(1,2,4)

(f_out, f_in, k_size)

*each emb_*dim → *each in_channels*

w_0

[-0.3, 0.1, 0.1]

$Window\_0_0$

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

$Window\_0_1$

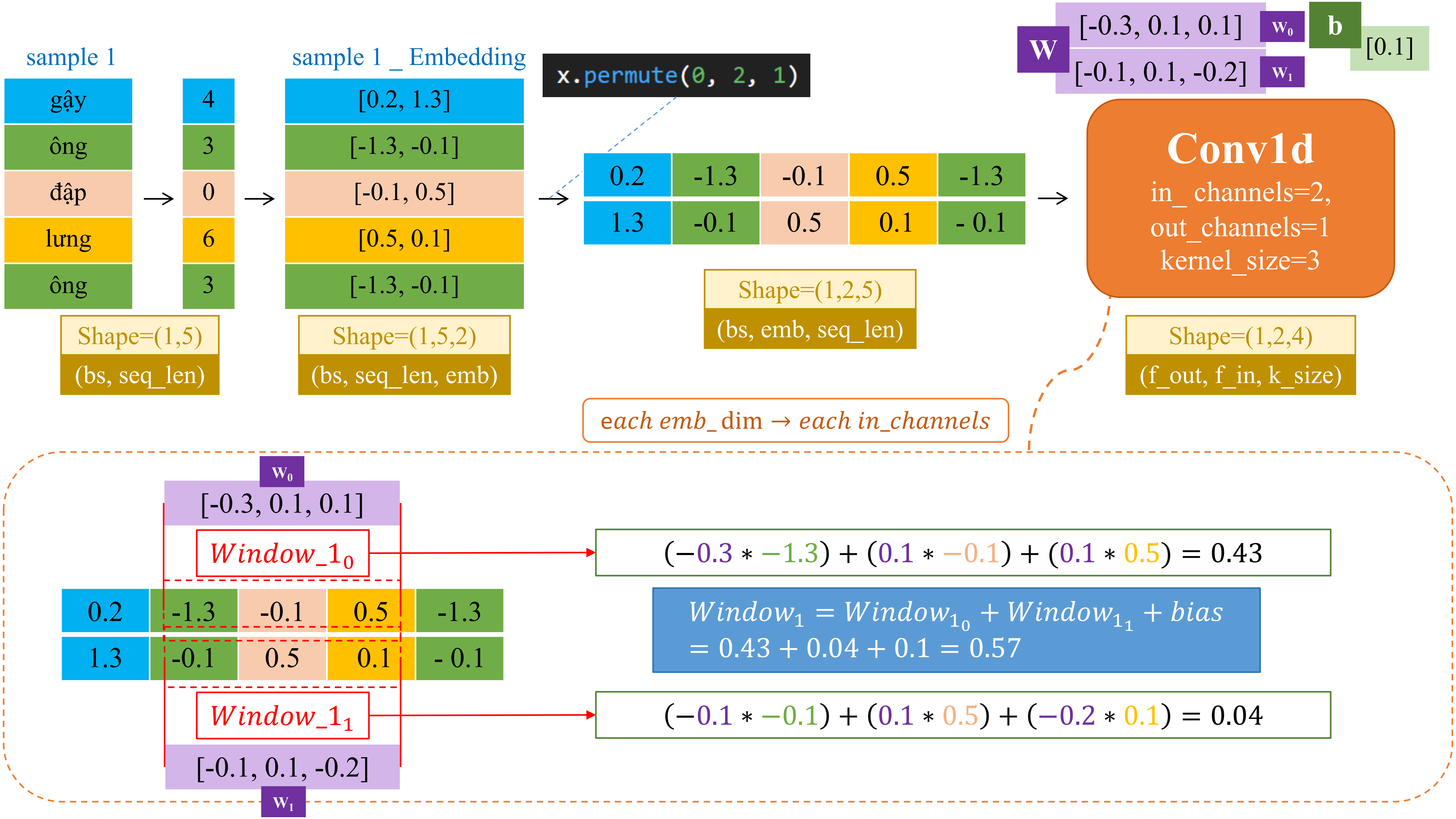
[-0.1, 0.1, -0.2]

w_1

$$(-0.3 * 0.2) + (0.1 * -1.3) + (0.1 * -0.1) = -0.2$$

$$Window_0 = Window_{0_0} + Window_{0_1} + bias$$
$$= (-0.2) + (-0.24) + 0.1 = -0.34$$

$$(-0.1 * 1.3) + (0.1 * -0.1) + (-0.2 * 0.5) = -0.24$$



sample 1

sample 1	
gậy	4
ông	3
đập	0
lưng	6
ông	3

Shape=(1,5)

(bs, seq_len)

sample 1 _ Embedding

sample 1 _ Embedding
[0.2, 1.3]
[-1.3, -0.1]
[-0.1, 0.5]
[0.5, 0.1]
[-1.3, -0.1]

Shape=(1,5,2)

(bs, seq_len, emb)

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

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

Shape=(1,2,5)

(bs, emb, seq_len)

W

[-0.3, 0.1, 0.1] w_0

[-0.1, 0.1, -0.2] w_1

b [0.1]

Conv1d

in_ channels=2,
out_channels=1
kernel_size=3

Shape=(1,2,4)

(f_out, f_in, k_size)

each emb_dim → *each in_channels*

w_0

[-0.3, 0.1, 0.1]

$Window\_1_0$

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

$Window\_1_1$

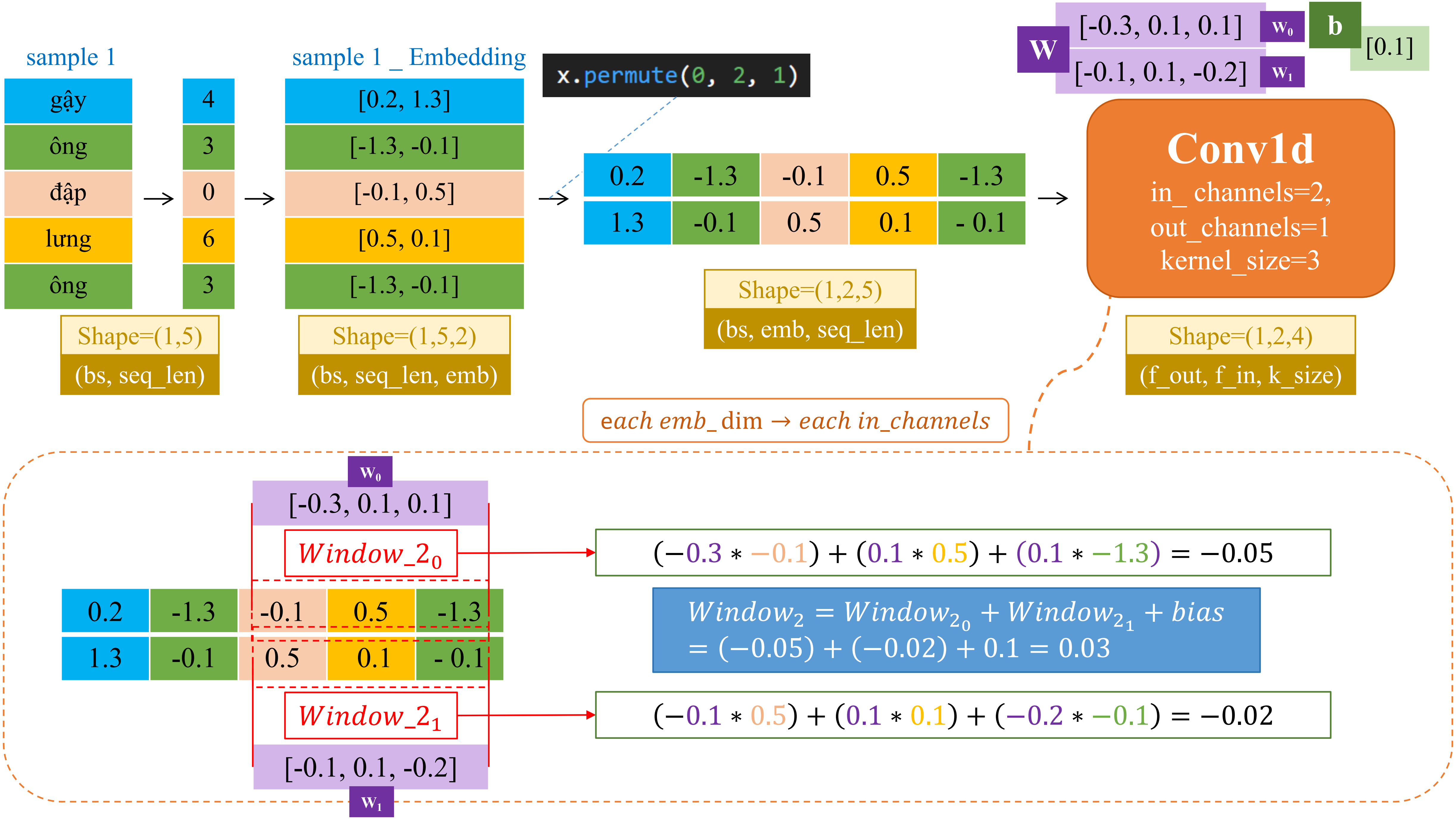
[-0.1, 0.1, -0.2]

w_1

$$(-0.3 * -1.3) + (0.1 * -0.1) + (0.1 * 0.5) = 0.43$$

$$Window_1 = Window_{1_0} + Window_{1_1} + bias$$
$$= 0.43 + 0.04 + 0.1 = 0.57$$

$$(-0.1 * -0.1) + (0.1 * 0.5) + (-0.2 * 0.1) = 0.04$$



sample 1

gậy
ông
đập
lưng
ông

→

4
3
0
6
3

Shape=(1,5)
(bs, seq_len)

→

sample 1 _ Embedding

[0.2, 1.3]
[-1.3, -0.1]
[-0.1, 0.5]
[0.5, 0.1]
[-1.3, -0.1]

Shape=(1,5,2)
(bs, seq_len, emb)

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

→

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

Shape=(1,2,5)
(bs, emb, seq_len)

→

Conv1d
in_ channels=2,
out_channels=1
kernel_size=3

W
[-0.3, 0.1, 0.1] w_0
[-0.1, 0.1, -0.2] w_1

b
[0.1]

Shape=(1,2,4)
(f_out, f_in, k_size)

*each emb_*dim → *each in_channels*

w_0
[-0.3, 0.1, 0.1]

$Window\_2_0$

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

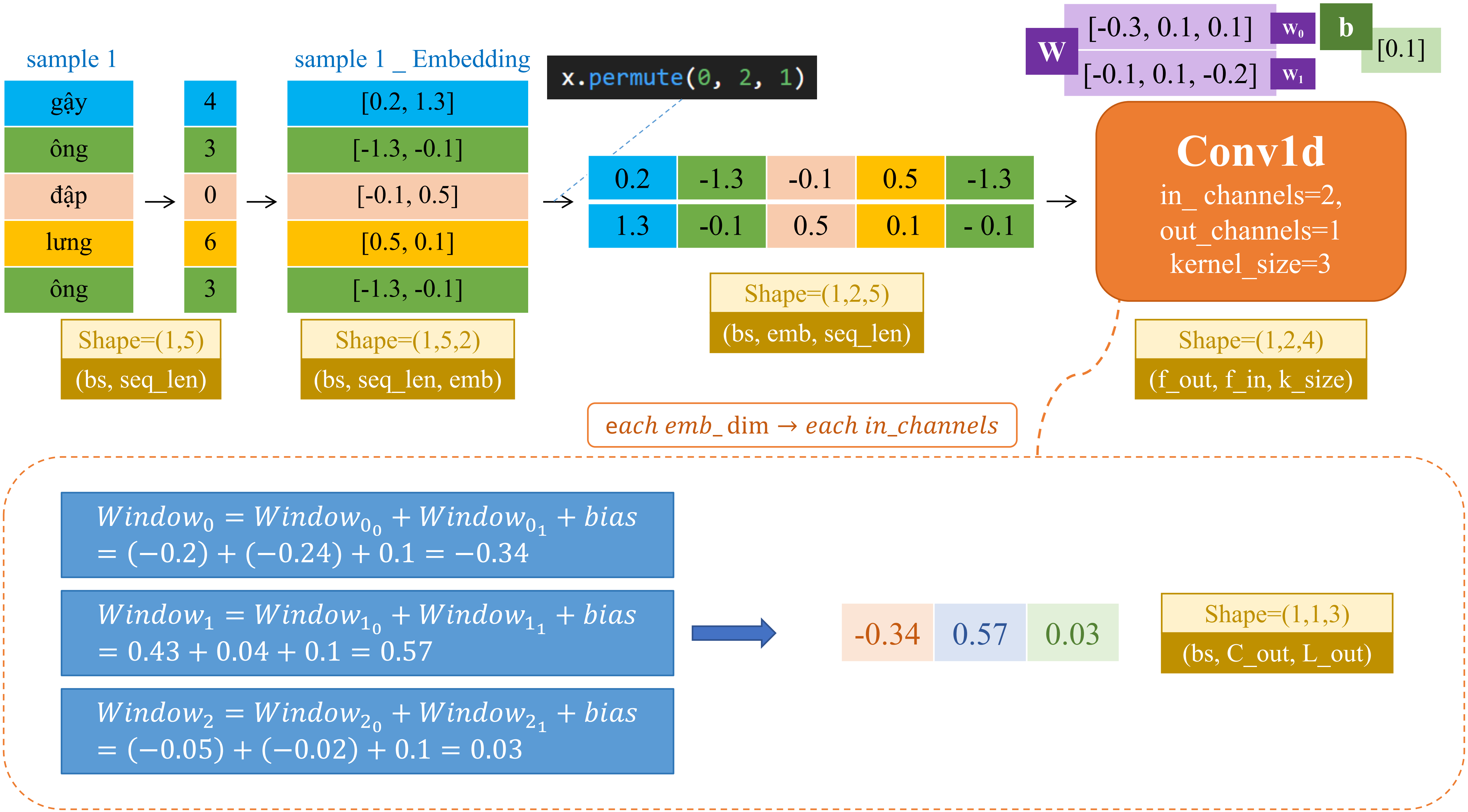
$Window\_2_1$

[-0.1, 0.1, -0.2]
w_1

$$(-0.3 * -0.1) + (0.1 * 0.5) + (0.1 * -1.3) = -0.05$$

$$Window_2 = Window_{2_0} + Window_{2_1} + bias$$
$$= (-0.05) + (-0.02) + 0.1 = 0.03$$

$$(-0.1 * 0.5) + (0.1 * 0.1) + (-0.2 * -0.1) = -0.02$$



sample 1		sample 1 _ Embedding
gậy	4	[0.2, 1.3]
ông	3	[-1.3, -0.1]
đập	0	[-0.1, 0.5]
lưng	6	[0.5, 0.1]
ông	3	[-1.3, -0.1]

Shape=(1,5)
(bs, seq_len)

Shape=(1,5,2)
(bs, seq_len, emb)

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

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

Shape=(1,2,5)
(bs, emb, seq_len)

each emb_dim → *each in_channels*

W

[-0.3, 0.1, 0.1]	w_0
[-0.1, 0.1, -0.2]	w_1

b [0.1]

Conv1d
in_ channels=2,
out_channels=1
kernel_size=3

Shape=(1,2,4)
(f_out, f_in, k_size)

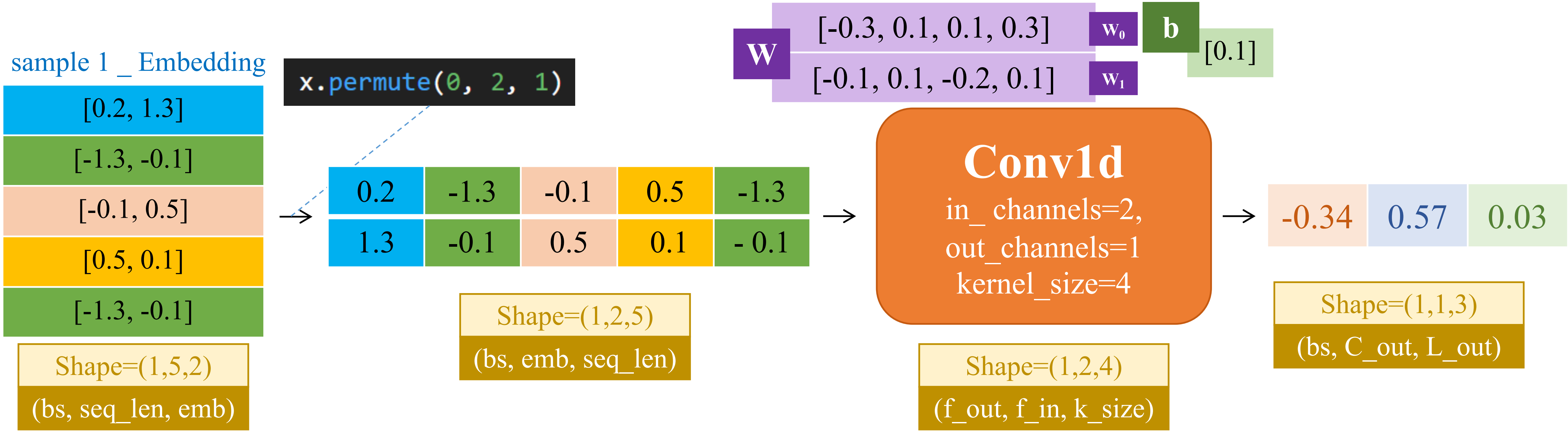
$$Window_0 = Window_{0_0} + Window_{0_1} + bias$$
$$= (-0.2) + (-0.24) + 0.1 = -0.34$$

$$Window_1 = Window_{1_0} + Window_{1_1} + bias$$
$$= 0.43 + 0.04 + 0.1 = 0.57$$

$$Window_2 = Window_{2_0} + Window_{2_1} + bias$$
$$= (-0.05) + (-0.02) + 0.1 = 0.03$$

-0.34	0.57	0.03

Shape=(1,1,3)
(bs, C_out, L_out)



sample 1 _ Embedding

[0.2, 1.3]
[-1.3, -0.1]
[-0.1, 0.5]
[0.5, 0.1]
[-1.3, -0.1]

Shape=(1,5,2)

(bs, seq_len, emb)

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

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

Shape=(1,2,5)

(bs, emb, seq_len)

W

[-0.3, 0.1, 0.1, 0.3] w_0

[-0.1, 0.1, -0.2, 0.1] w_1

b [0.1]

Conv1d

in_ channels=2,
out_channels=1
kernel_size=4

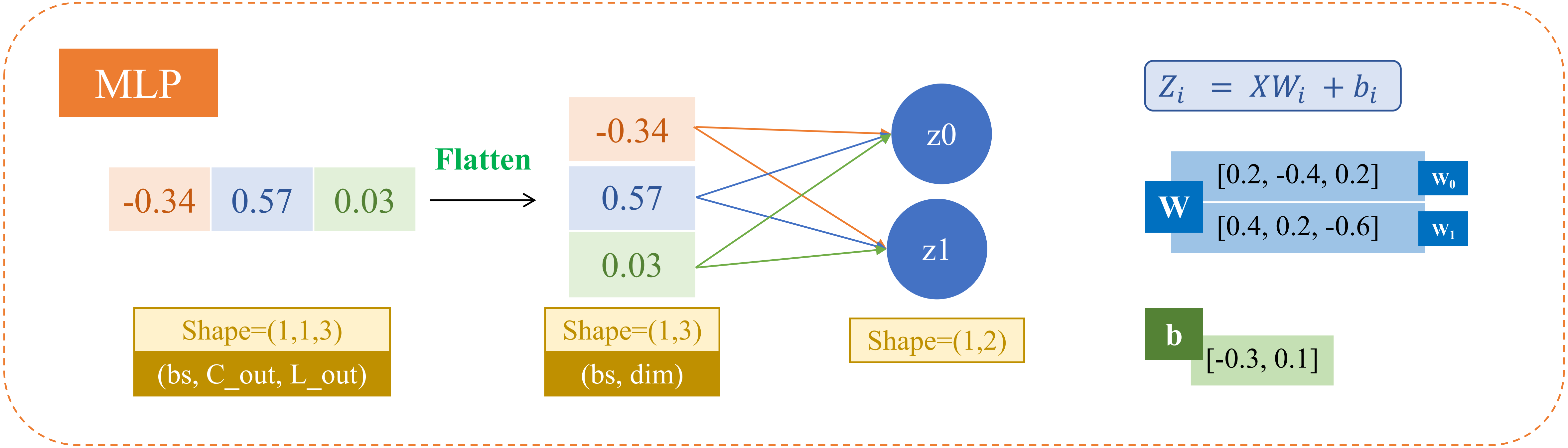
Shape=(1,2,4)

(f_out, f_in, k_size)

-0.34	0.57	0.03

Shape=(1,1,3)

(bs, C_out, L_out)



MLP

-0.34	0.57	0.03

Shape=(1,1,3)

(bs, C_out, L_out)

Flatten

-0.34
0.57
0.03

Shape=(1,3)

(bs, dim)

Shape=(1,2)

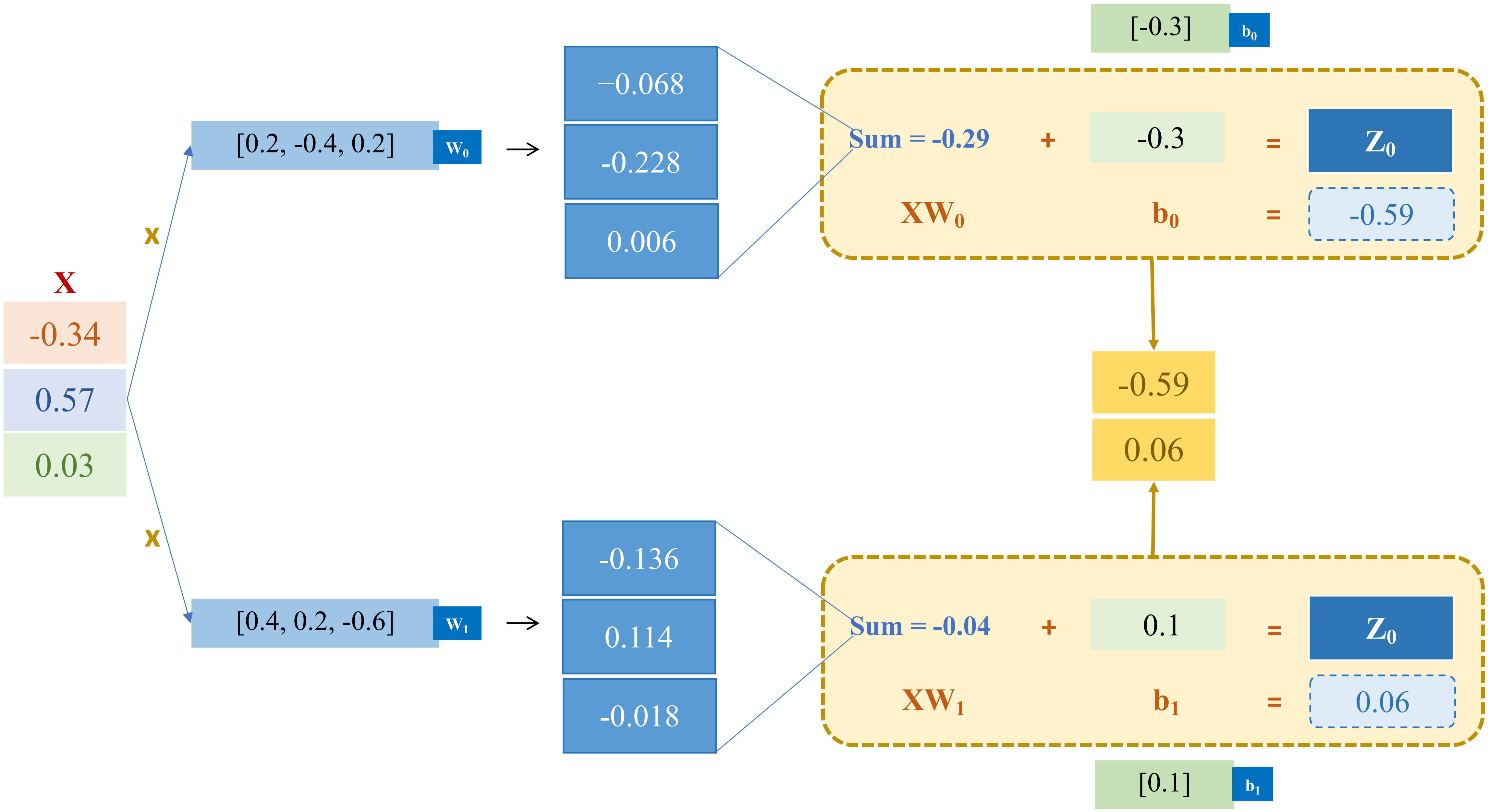
$$Z_i = XW_i + b_i$$

W

[0.2, -0.4, 0.2] w_0

[0.4, 0.2, -0.6] w_1

b [-0.3, 0.1]



X

-0.34

0.57

0.03

x

[0.2, -0.4, 0.2] w_0 →

−0.068

-0.228

0.006

[-0.3] b_0

Sum = -0.29 + -0.3 = Z_0

XW_0 b_0 = -0.59

-0.59

0.06

x

[0.4, 0.2, -0.6] w_1 →

-0.136

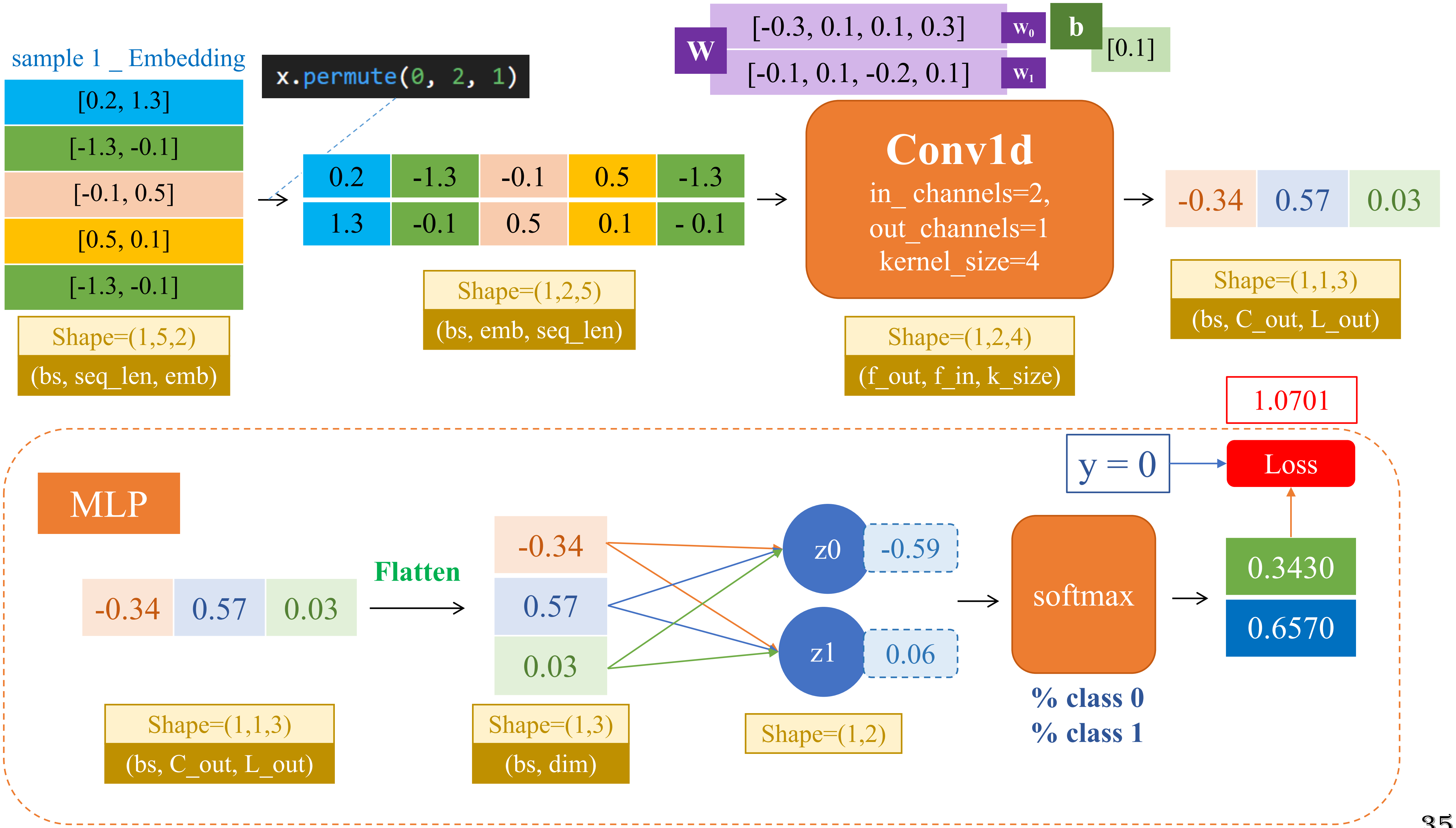
0.114

-0.018

Sum = -0.04 + 0.1 = Z_0

XW_1 b_1 = 0.06

[0.1] b_1



sample 1 _ Embedding

[0.2, 1.3]
[-1.3, -0.1]
[-0.1, 0.5]
[0.5, 0.1]
[-1.3, -0.1]

Shape=(1,5,2)

(bs, seq_len, emb)

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

0.2	-1.3	-0.1	0.5	-1.3
1.3	-0.1	0.5	0.1	- 0.1

Shape=(1,2,5)

(bs, emb, seq_len)

W

[-0.3, 0.1, 0.1, 0.3]	w_0
[-0.1, 0.1, -0.2, 0.1]	w_1

b

[0.1]

Conv1d

in_ channels=2,
out_channels=1
kernel_size=4

Shape=(1,2,4)

(f_out, f_in, k_size)

-0.34	0.57	0.03

Shape=(1,1,3)

(bs, C_out, L_out)

1.0701

y = 0

Loss

MLP

-0.34	0.57	0.03

Shape=(1,1,3)

(bs, C_out, L_out)

Flatten

-0.34
0.57
0.03

Shape=(1,3)

(bs, dim)

z0 -0.59

z1 0.06

Shape=(1,2)

softmax

% class 0
% class 1

0.3430
0.6570

Text Classification: Example 4

Doc	Label
gậy ông đập lưng ông	0
có làm mới có ăn	1

building dictionary →

vocab size = 8
sequence length = 5

index	word
0	[UNK]
1	[pad]
2	có
3	ông
4	gậy
5	làm
6	lưng
7	mới

Dictionary

0	[-0.1, 0.5]
1	[1.7, -0.8]
2	[1.0, -1.9]
3	[-1.3, -0.1]
4	[0.2, 1.3]
5	[0.4, -0.6]
6	[0.5, 0.1]
7	[0.4, -1.3]

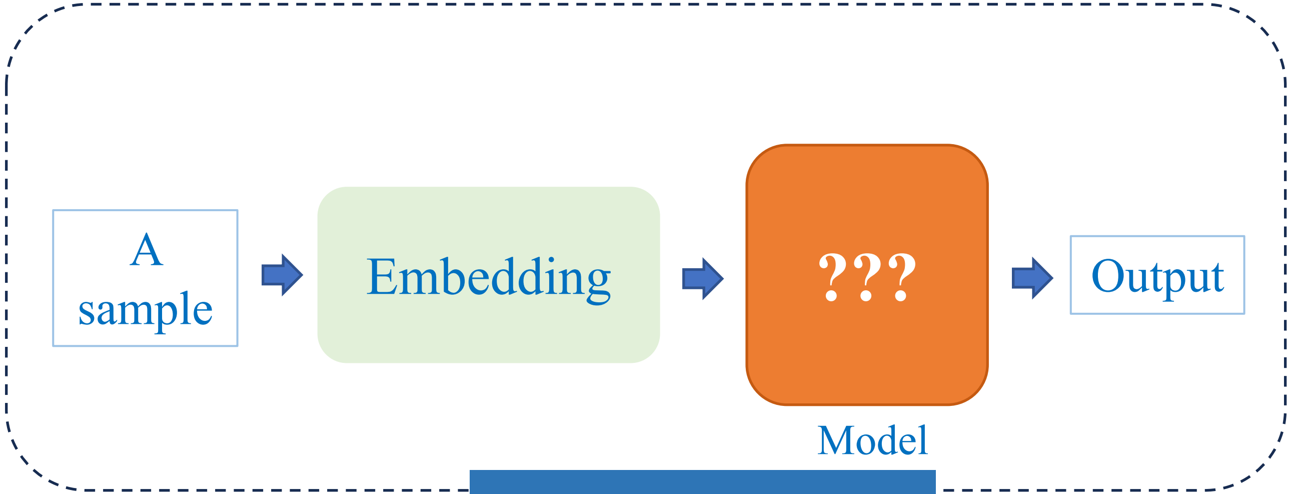
Embedding

gậy		4		[0.2, 1.3]
ông		3		[-1.3, -0.1]
đập	→	0	→	[-0.1, 0.5]
lưng		6		[0.5, 0.1]
ông		3		[-1.3, -0.1]

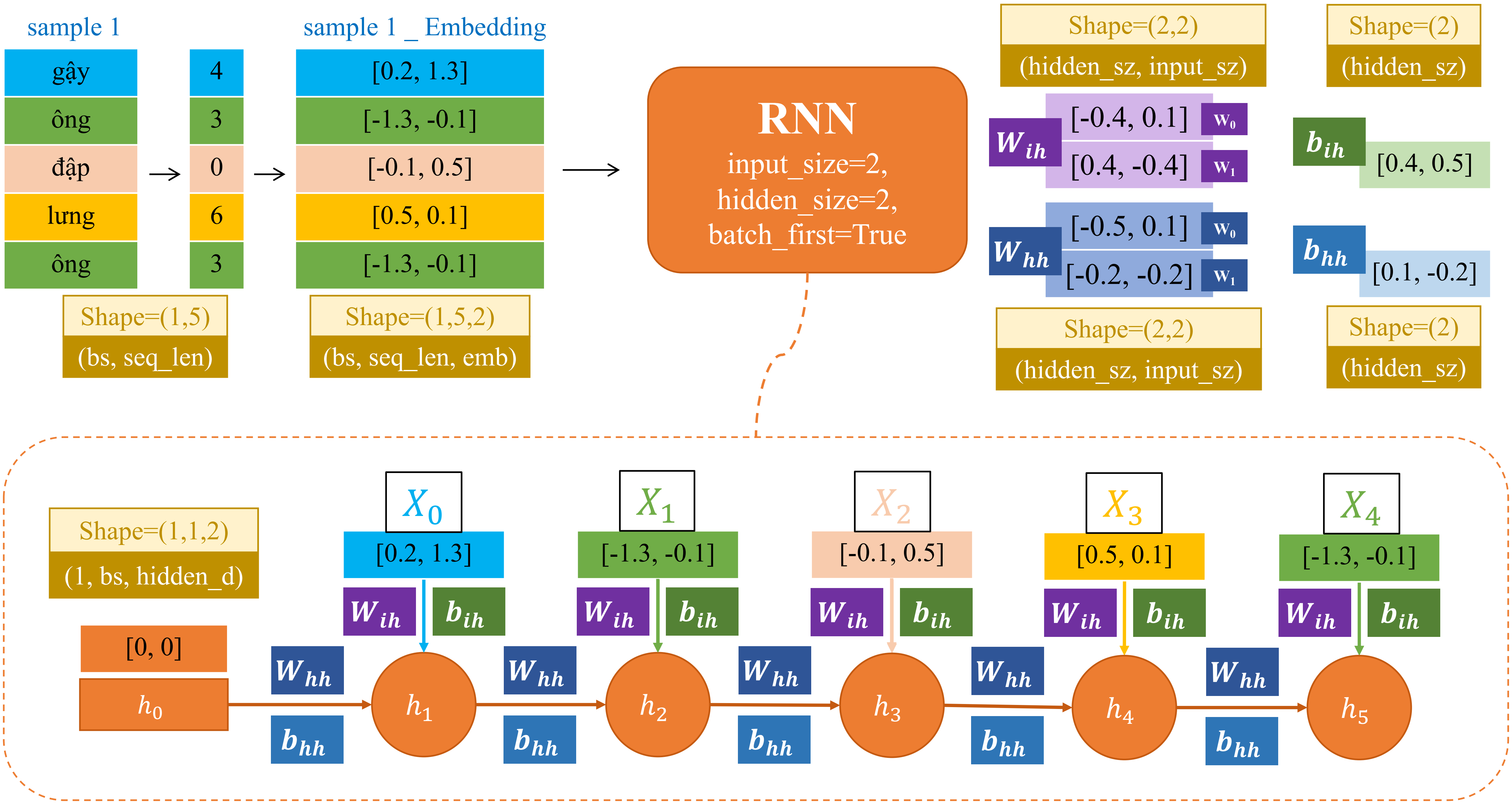
sample 1

sample 1 _ Embedding

Vectorization and Embedding



Model Pipeline



sample 1		sample 1 _ Embedding
gậy	4	[0.2, 1.3]
ông	3	[-1.3, -0.1]
đập	0	[-0.1, 0.5]
lưng	6	[0.5, 0.1]
ông	3	[-1.3, -0.1]
Shape=(1,5) (bs, seq_len)		Shape=(1,5,2) (bs, seq_len, emb)

RNN
input_size=2,
hidden_size=2,
batch_first=True

Shape=(2,2)
(hidden_sz, input_sz)

W_{ih}: [-0.4, 0.1] w_0; [0.4, -0.4] w_1

Shape=(2)
(hidden_sz)

b_{ih}: [0.4, 0.5]

W_{hh}: [-0.5, 0.1] w_0; [-0.2, -0.2] w_1

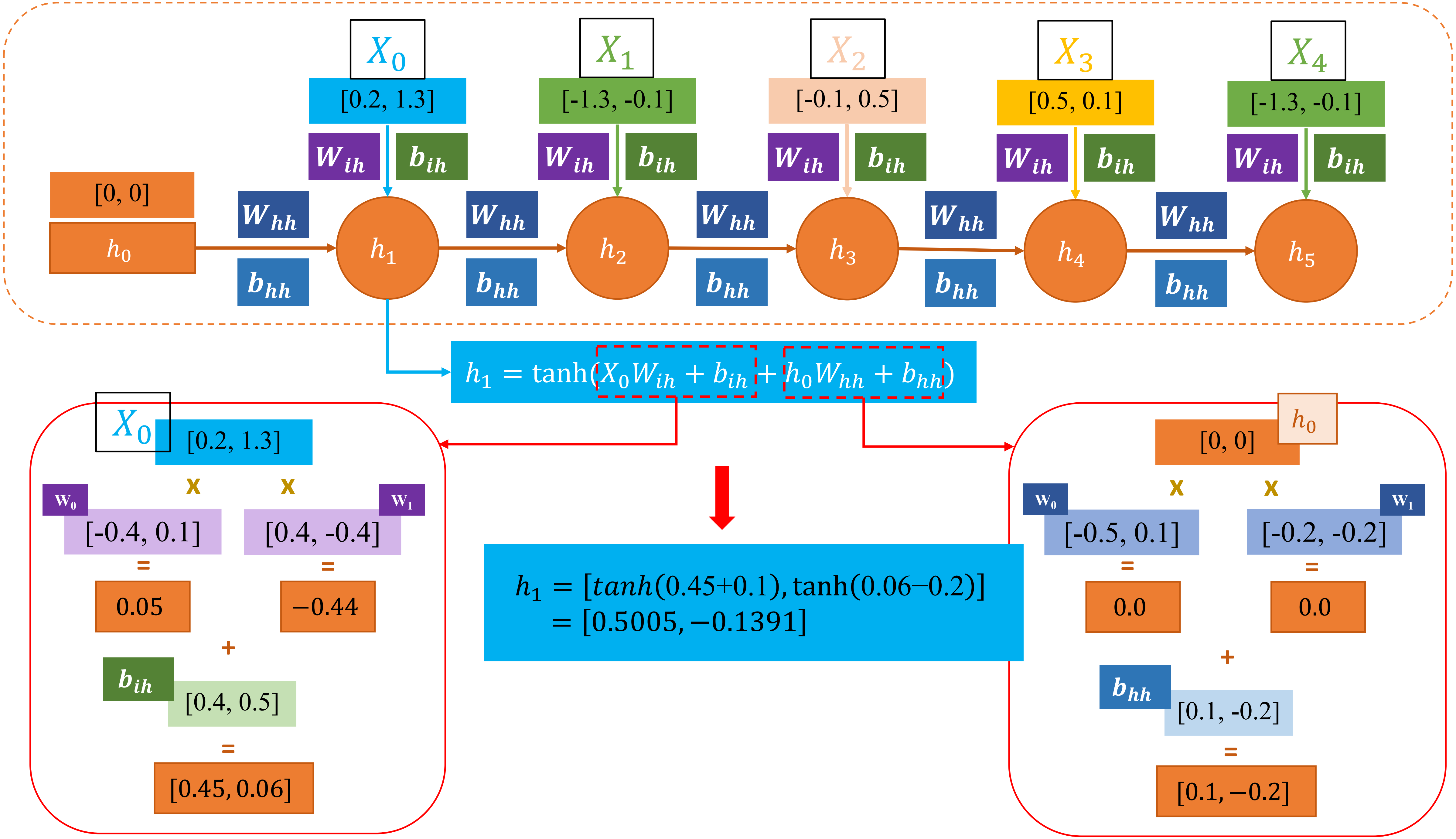
Shape=(2,2)
(hidden_sz, input_sz)

b_{hh}: [0.1, -0.2]

Shape=(2)
(hidden_sz)

Shape=(1,1,2)
(1, bs, hidden_d)

[0, 0]



X_0 [0.2, 1.3]

X_1 [-1.3, -0.1]

X_2 [-0.1, 0.5]

X_3 [0.5, 0.1]

X_4 [-1.3, -0.1]

W_{ih} b_{ih}

[0, 0] h_0

W_{hh} b_{hh}

h_1 h_2 h_3 h_4 h_5

$$h_1 = \tanh(X_0 W_{ih} + b_{ih} + h_0 W_{hh} + b_{hh})$$

X_0 [0.2, 1.3]

W_0 x [-0.4, 0.1] = 0.05

x W_1 [0.4, -0.4] = -0.44

+ b_{ih} [0.4, 0.5] = [0.45, 0.06]

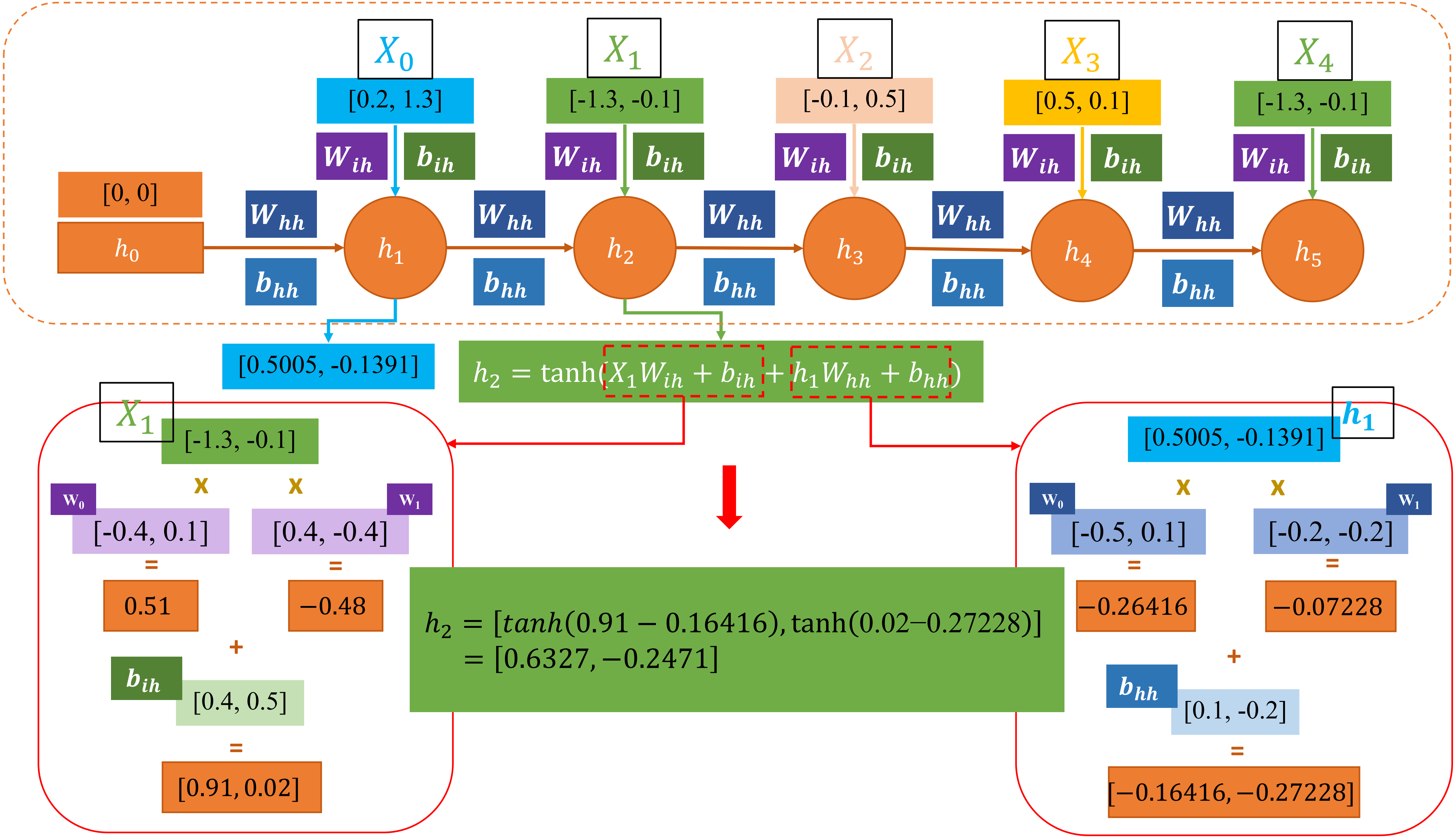
$$h_1 = [tanh(0.45+0.1), \tanh(0.06-0.2)]$$
$$= [0.5005, -0.1391]$$

h_0 [0, 0]

W_0 x [-0.5, 0.1] = 0.0

x W_1 [-0.2, -0.2] = 0.0

+ b_{hh} [0.1, -0.2] = [0.1, -0.2]



$$h_2 = \tanh(X_1 W_{ih} + b_{ih} + h_1 W_{hh} + b_{hh})$$

X_1 [-1.3, -0.1]

x W_0 [-0.4, 0.1] = 0.51

x W_1 [0.4, -0.4] = −0.48

+ b_{ih} [0.4, 0.5]

= [0.91, 0.02]

h_1 [0.5005, -0.1391]

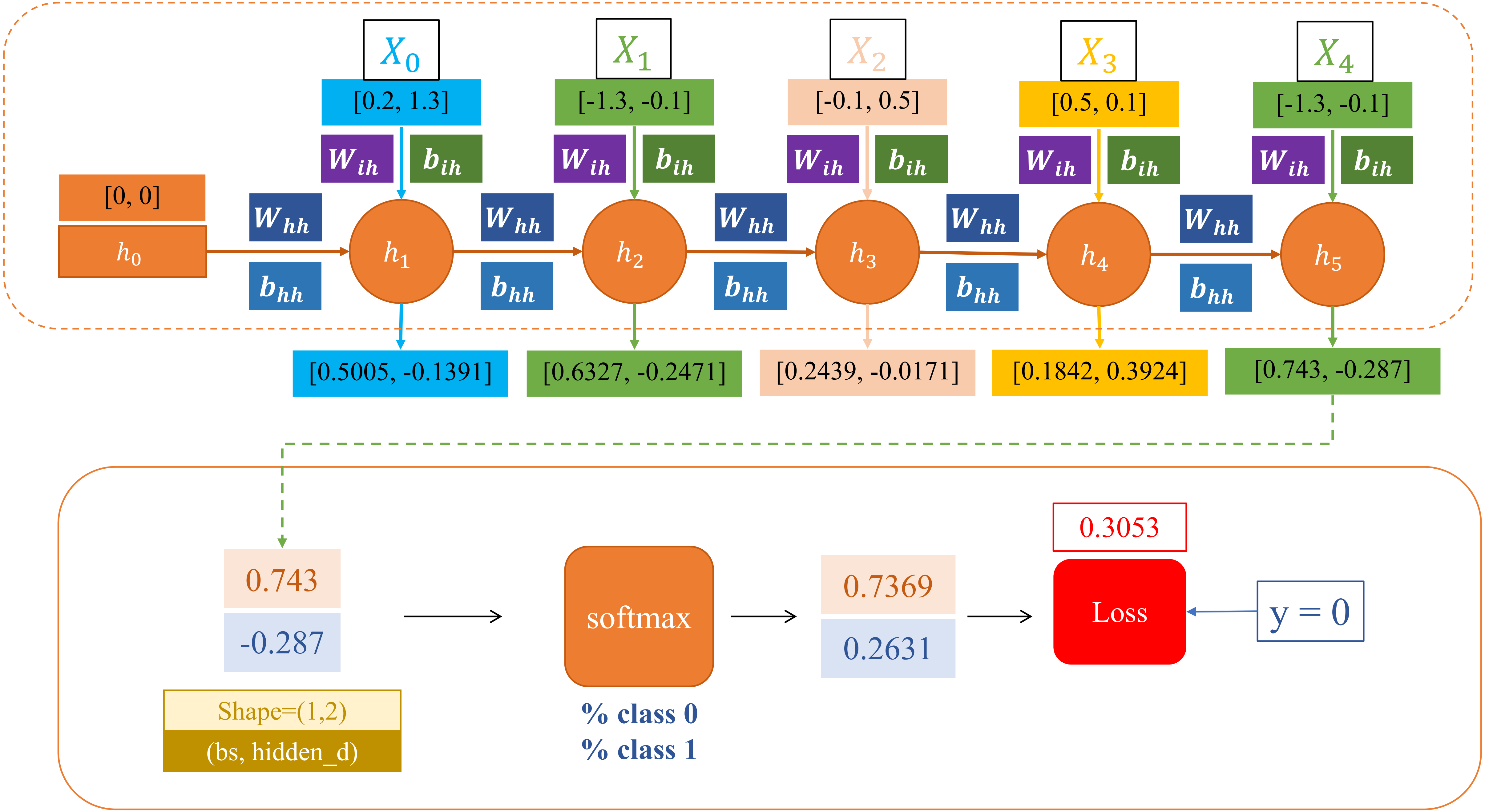
x W_0 [-0.5, 0.1] = −0.26416

x W_1 [-0.2, -0.2] = −0.07228

+ b_{hh} [0.1, -0.2]

= [−0.16416, −0.27228]

$$h_2 = [tanh(0.91 - 0.16416), \tanh(0.02 - 0.27228)] \\ = [0.6327, -0.2471]$$



X_0 [0.2, 1.3]

X_1 [-1.3, -0.1]

X_2 [-0.1, 0.5]

X_3 [0.5, 0.1]

X_4 [-1.3, -0.1]

W_{ih} b_{ih}

[0, 0] h_0

W_{hh} b_{hh}

h_1 h_2 h_3 h_4 h_5

[0.5005, -0.1391]

[0.6327, -0.2471]

[0.2439, -0.0171]

[0.1842, 0.3924]

[0.743, -0.287]

0.743

-0.287

Shape=(1,2)

(bs, hidden_d)

softmax

% class 0

% class 1

0.7369

0.2631

0.3053

Loss

y = 0

Text Classification: Example 5

Doc	Label
gậy ông đập lưng ông	0
có làm mới có ăn	1

building dictionary →

vocab size = 8
sequence length = 5

index	word
0	[UNK]
1	[pad]
2	có
3	ông
4	gậy
5	làm
6	lưng
7	mới

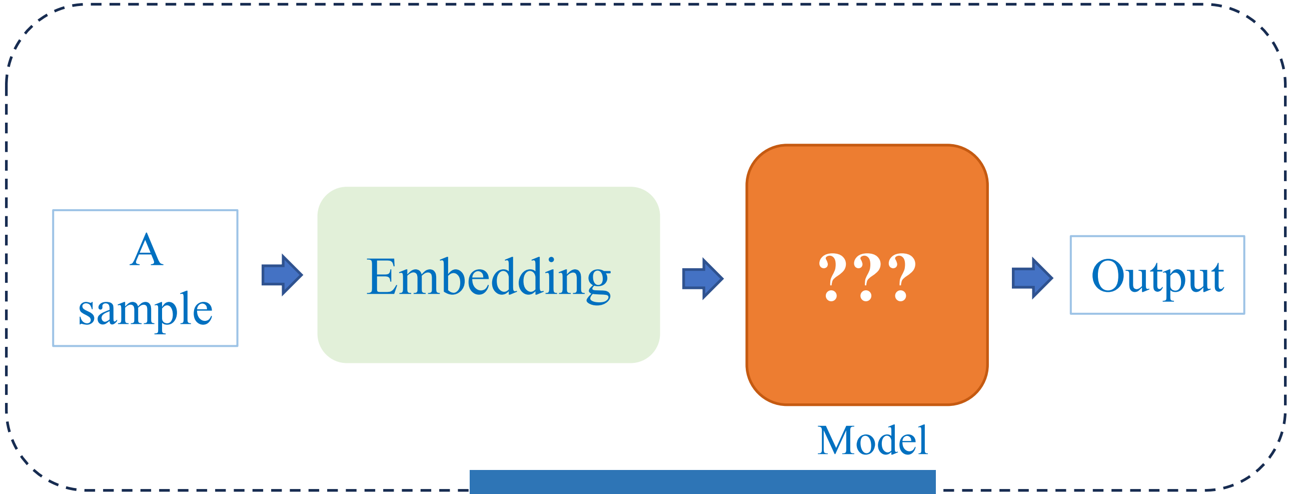
Dictionary

0	[-0.1, 0.5]
1	[1.7, -0.8]
2	[1.0, -1.9]
3	[-1.3, -0.1]
4	[0.2, 1.3]
5	[0.4, -0.6]
6	[0.5, 0.1]
7	[0.4, -1.3]

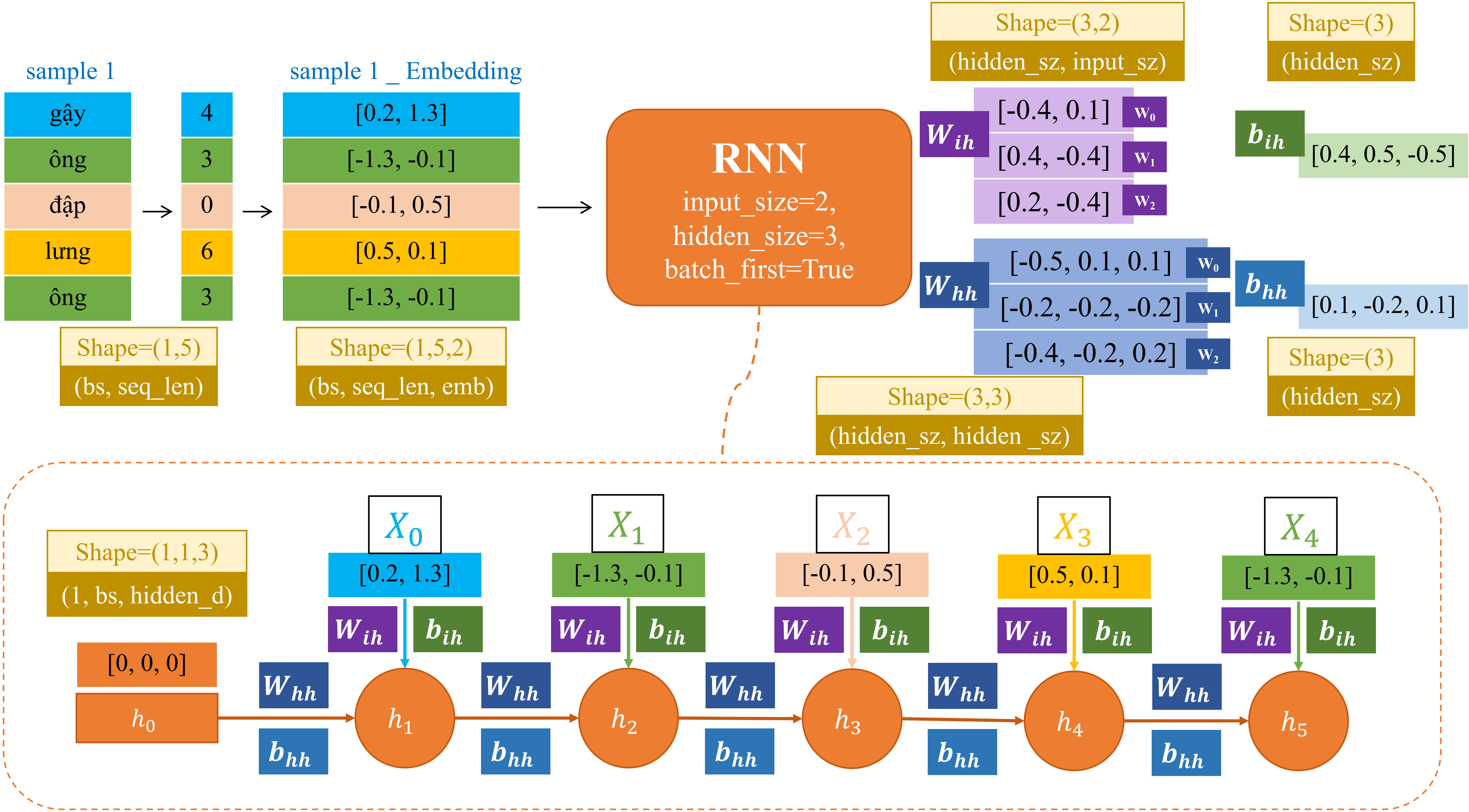
Embedding

gậy		4		[0.2, 1.3]
ông		3		[-1.3, -0.1]
đập	→	0	→	[-0.1, 0.5]
lưng		6		[0.5, 0.1]
ông		3		[-1.3, -0.1]
sample 1				sample 1 _ Embedding

Vectorization and Embedding



Model Pipeline



sample 1		sample 1 _ Embedding
gậy	4	[0.2, 1.3]
ông	3	[-1.3, -0.1]
đập	0	[-0.1, 0.5]
lưng	6	[0.5, 0.1]
ông	3	[-1.3, -0.1]

Shape=(1,5)
(bs, seq_len)

Shape=(1,5,2)
(bs, seq_len, emb)

RNN
input_size=2,
hidden_size=3,
batch_first=True

Shape=(3,2)
(hidden_sz, input_sz)

W_{ih}

[-0.4, 0.1]	w_0
[0.4, -0.4]	w_1
[0.2, -0.4]	w_2

Shape=(3)
(hidden_sz)

b_{ih} [0.4, 0.5, -0.5]

W_{hh}

[-0.5, 0.1, 0.1]	w_0
[-0.2, -0.2, -0.2]	w_1
[-0.4, -0.2, 0.2]	w_2

Shape=(3,3)
(hidden_sz, hidden _sz)

b_{hh} [0.1, -0.2, 0.1]

Shape=(3)
(hidden_sz)

Shape=(1,1,3)
(1, bs, hidden_d)

[0, 0, 0]
h_0

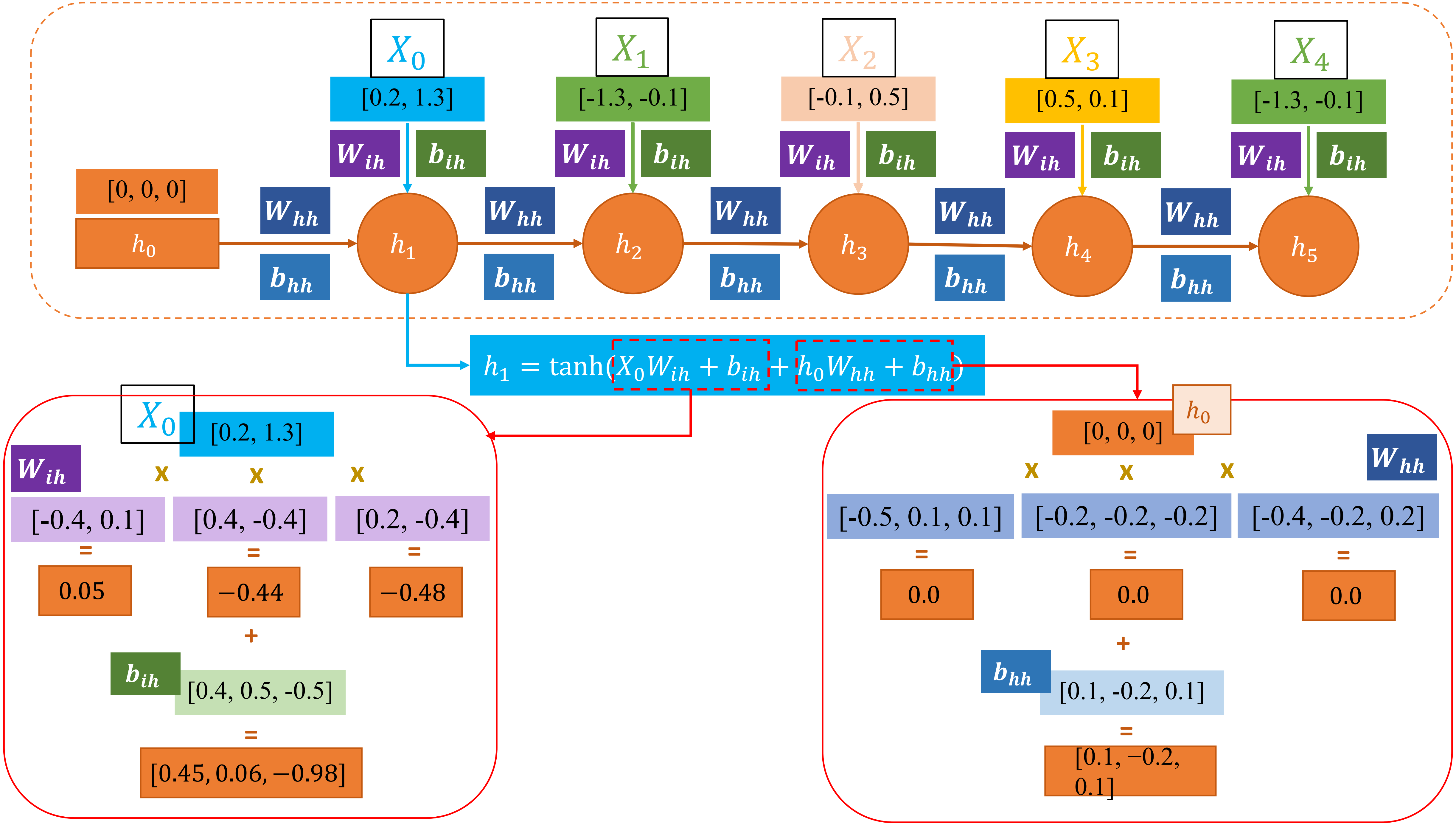
X_0 [0.2, 1.3] W_{ih} b_{ih} W_{hh} b_{hh} h_1

X_1 [-1.3, -0.1] W_{ih} b_{ih} W_{hh} b_{hh} h_2

X_2 [-0.1, 0.5] W_{ih} b_{ih} W_{hh} b_{hh} h_3

X_3 [0.5, 0.1] W_{ih} b_{ih} W_{hh} b_{hh} h_4

X_4 [-1.3, -0.1] W_{ih} b_{ih} W_{hh} b_{hh} h_5



X_0 [0.2, 1.3] W_{ih} b_{ih}

X_1 [-1.3, -0.1] W_{ih} b_{ih}

X_2 [-0.1, 0.5] W_{ih} b_{ih}

X_3 [0.5, 0.1] W_{ih} b_{ih}

X_4 [-1.3, -0.1] W_{ih} b_{ih}

[0, 0, 0] h_0

W_{hh} b_{hh} h_1 W_{hh} b_{hh} h_2 W_{hh} b_{hh} h_3 W_{hh} b_{hh} h_4 W_{hh} b_{hh} h_5

$$h_1 = \tanh(X_0 W_{ih} + b_{ih} + h_0 W_{hh} + b_{hh})$$

X_0 [0.2, 1.3]

W_{ih}

x x x

[-0.4, 0.1] [0.4, -0.4] [0.2, -0.4]

= = =

0.05 −0.44 −0.48

+

b_{ih} [0.4, 0.5, -0.5]

=

[0.45, 0.06, −0.98]

h_0 [0, 0, 0]

W_{hh}

x x x

[-0.5, 0.1, 0.1] [-0.2, -0.2, -0.2] [-0.4, -0.2, 0.2]

= = =

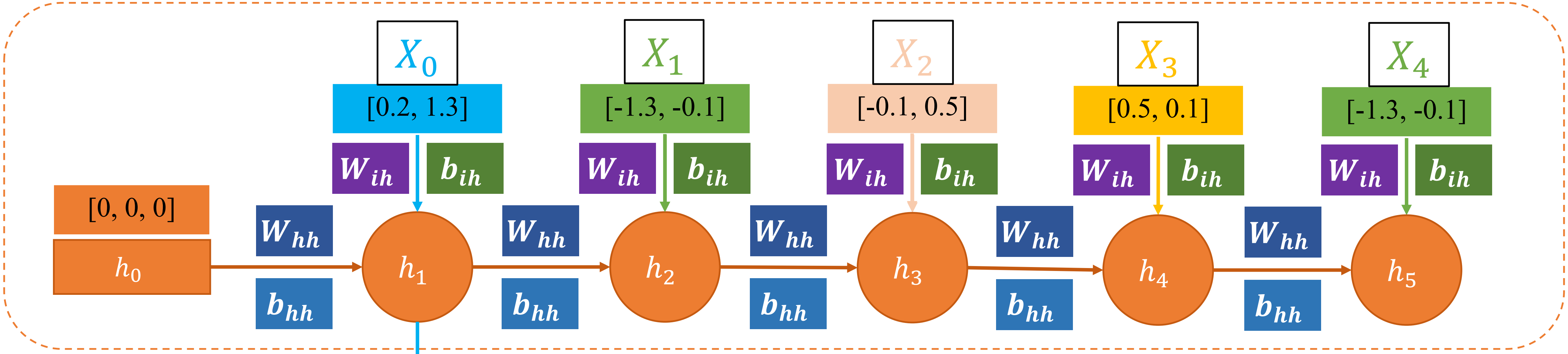
0.0 0.0 0.0

+

b_{hh} [0.1, -0.2, 0.1]

=

[0.1, −0.2, 0.1]



X_0 [0.2, 1.3]

X_1 [-1.3, -0.1]

X_2 [-0.1, 0.5]

X_3 [0.5, 0.1]

X_4 [-1.3, -0.1]

W_{ih} b_{ih}

W_{hh} b_{hh}

[0, 0, 0] h_0

h_1 h_2 h_3 h_4 h_5

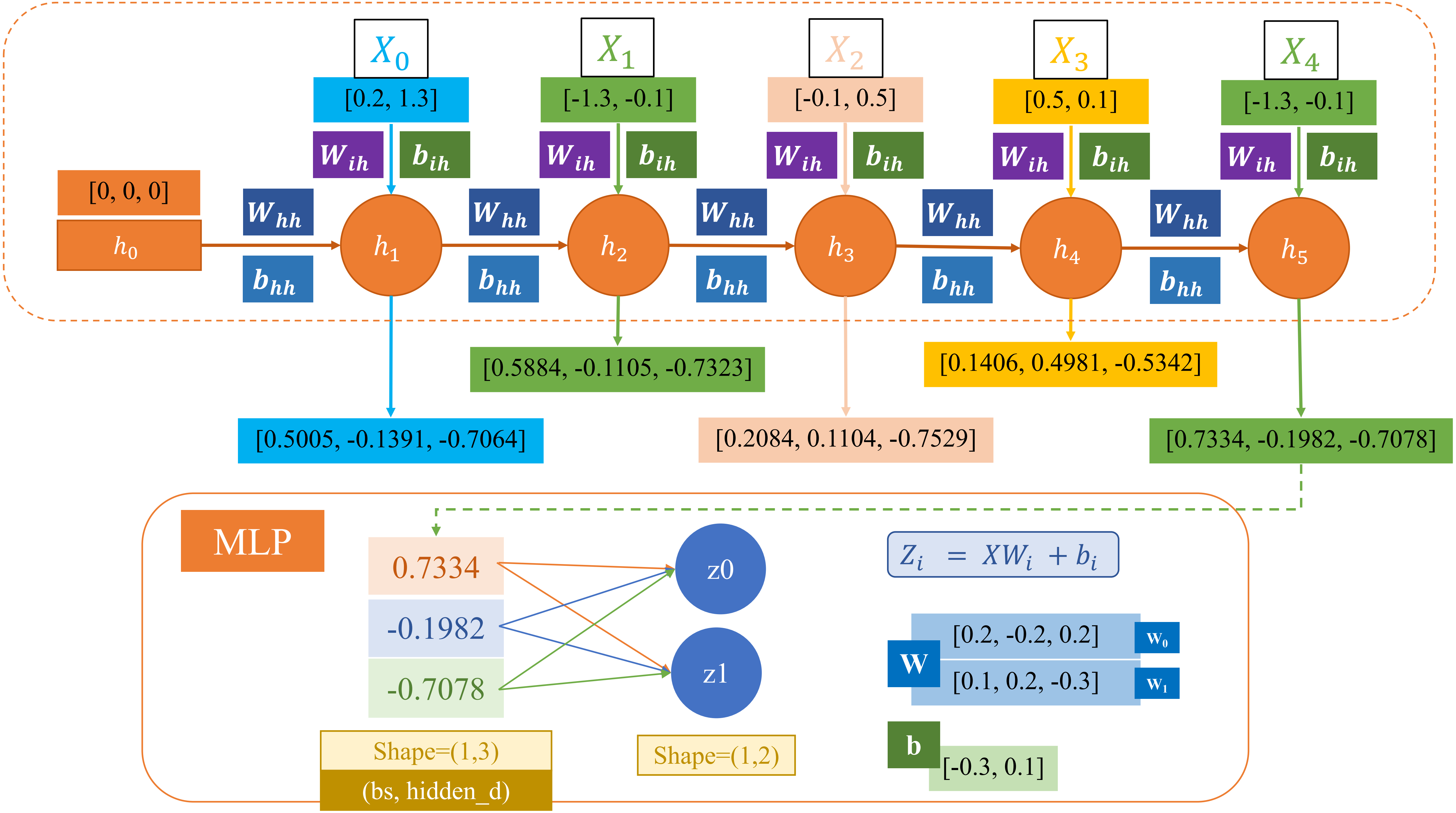
$$h_1 = \tanh(\boxed{X_0 W_{ih} + b_{ih}} + \boxed{h_0 W_{hh} + b_{hh}})$$

[0.45, 0.06, −0.98]

[0.1, −0.2, 0.1]

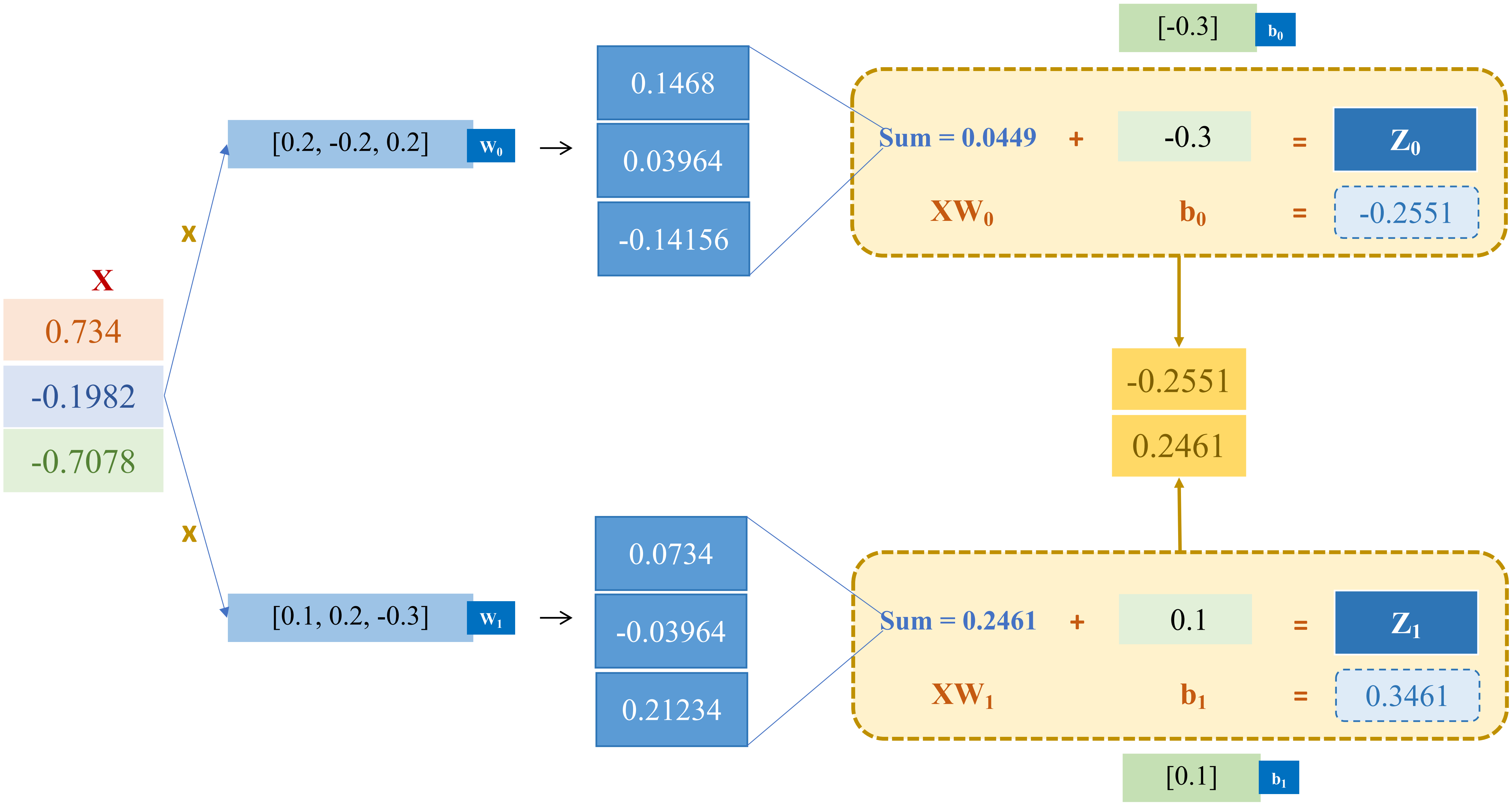
$$h_1 = [tanh(0.45{+}0.1), \tanh(0.06{-}0.2), \tanh(-0.98 + 0.1)]$$

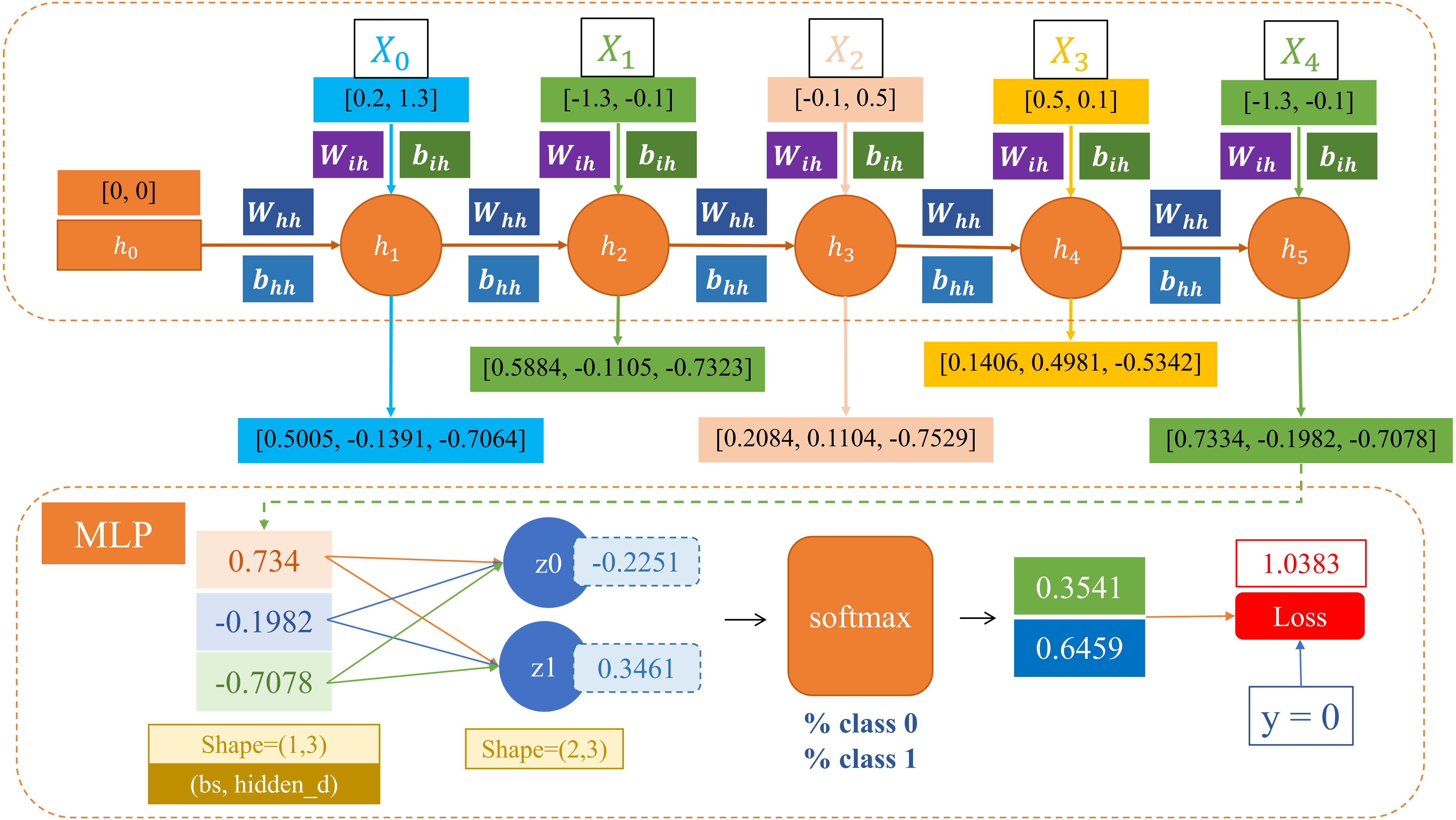
$$h_1 = [0.5005, -0.1391, -0.7064]$$



X_0
[0.2, 1.3]
X_1
[-1.3, -0.1]
X_2
[-0.1, 0.5]
X_3
[0.5, 0.1]
X_4
[-1.3, -0.1]
W_{ih}
b_{ih}
[0, 0, 0]
h_0
W_{hh}
b_{hh}
h_1
h_2
h_3
h_4
h_5
[0.5884, -0.1105, -0.7323]
[0.1406, 0.4981, -0.5342]
[0.5005, -0.1391, -0.7064]
[0.2084, 0.1104, -0.7529]
[0.7334, -0.1982, -0.7078]
MLP
0.7334
-0.1982
-0.7078
z0
z1
Shape=(1,3)
(bs, hidden_d)
Shape=(1,2)
W
[0.2, -0.2, 0.2]
w_0
[0.1, 0.2, -0.3]
w_1
b
[-0.3, 0.1]

$$Z_i = XW_i + b_i$$


X
0.734
-0.1982
-0.7078
x
x
[0.2, -0.2, 0.2]
W_0
→
0.1468
0.03964
-0.14156
[0.1, 0.2, -0.3]
W_1
→
0.0734
-0.03964
0.21234
[-0.3]
b_0
Sum = 0.0449
+
-0.3
=
Z_0
XW_0
b_0
=
-0.2551
-0.2551
0.2461
Sum = 0.2461
+
0.1
=
Z_1
XW_1
b_1
=
0.3461
[0.1]
b_1


X_0
[0.2, 1.3]
X_1
[-1.3, -0.1]
X_2
[-0.1, 0.5]
X_3
[0.5, 0.1]
X_4
[-1.3, -0.1]
W_{ih}
b_{ih}
[0, 0]
h_0
W_{hh}
b_{hh}
h_1
h_2
h_3
h_4
h_5
[0.5005, -0.1391, -0.7064]
[0.5884, -0.1105, -0.7323]
[0.2084, 0.1104, -0.7529]
[0.1406, 0.4981, -0.5342]
[0.7334, -0.1982, -0.7078]
MLP
0.734
-0.1982
-0.7078
Shape=(1,3)
(bs, hidden_d)
z0
-0.2251
z1
0.3461
Shape=(2,3)
softmax
% class 0
% class 1
0.3541
0.6459
1.0383
Loss
y = 0

Outline

SECTION 1

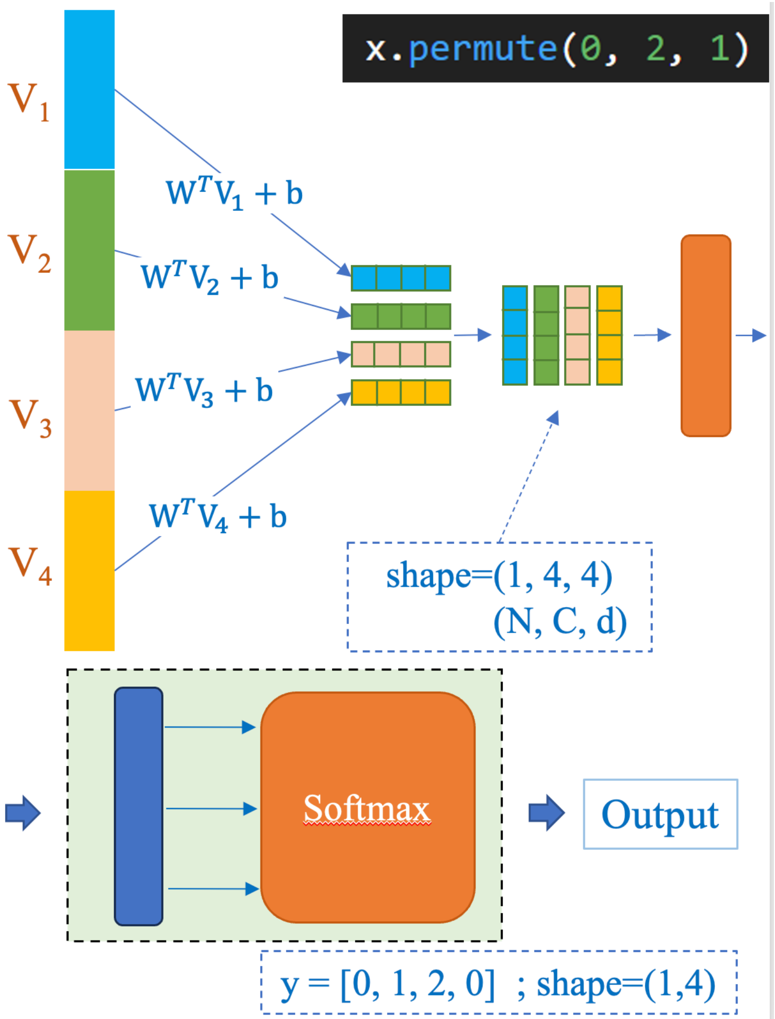
Review

SECTION 2

Text Classification

SECTION 3

POS Tagging



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

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

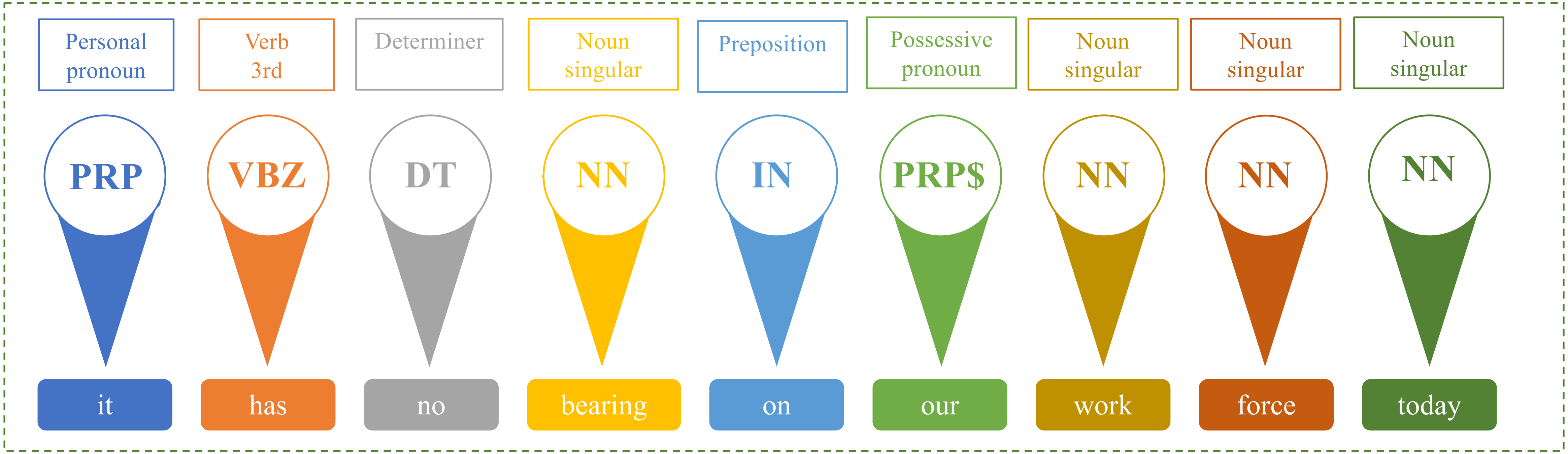
[”NNP", ”VBG", ”RB", ",", ”JJ", ”NN", ”NN", ”NNS", ”IN", ”NNP"]

Label-encoded

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

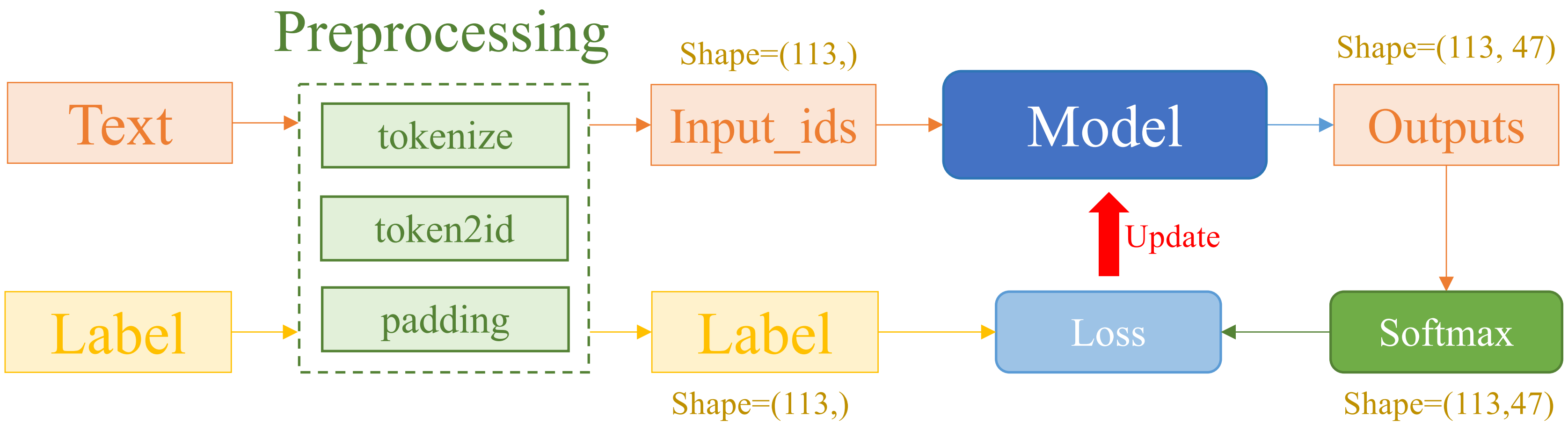
30	RB	Adverb
31	RBR	Adverb, comparative
32	RBS	Adverb, superlative
33	RP	Particle
34	SYM	Symbol
35	TO	to
36	UH	Interjection
37	VB	Verb, base form
38	VBD	Verb, past tense
39	VBG	Verb, gerund or present participle
40	VBN	Verb, past participle
41	VBP	Verb, non-3rd person singular present
42	VBZ	Verb, 3rd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

Part-of-speed Tagging



Personal pronoun	Verb 3rd	Determiner	Noun singular	Preposition	Possessive pronoun	Noun singular	Noun singular	Noun singular
PRP	VBZ	DT	NN	IN	PRP$	NN	NN	NN
it	has	no	bearing	on	our	work	force	today

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



Preprocessing

Text

tokenize

token2id

padding

Label

Shape=(113,)

Input_ids

Model

Shape=(113, 47)

Outputs

Update

Label

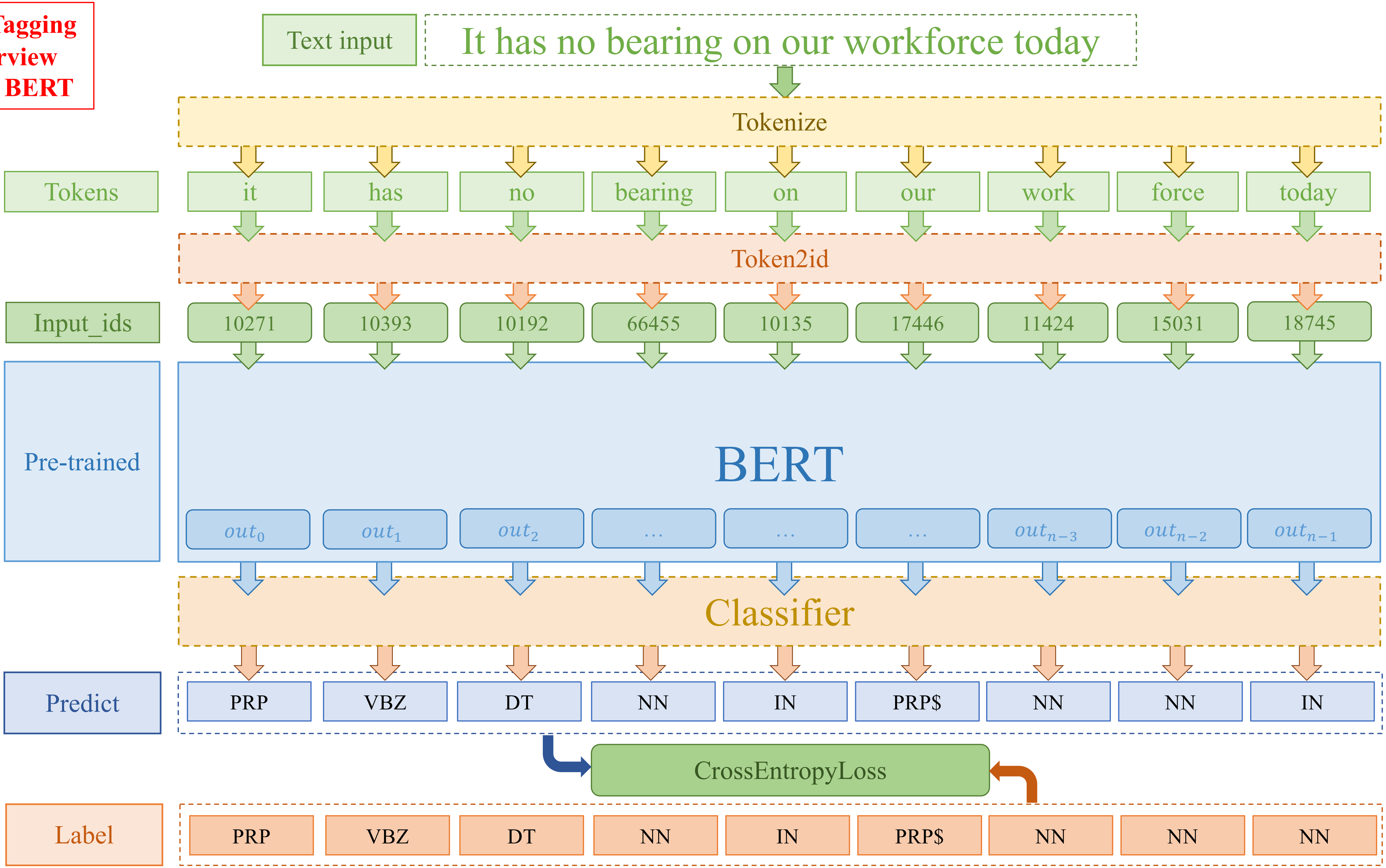
Shape=(113,)

Loss

Softmax

Shape=(113,47)

POS Tagging Overview Using BERT



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 | $\dots$ | $\dots$ | $\dots$ | 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

POS Tagging (1): One-Word Input

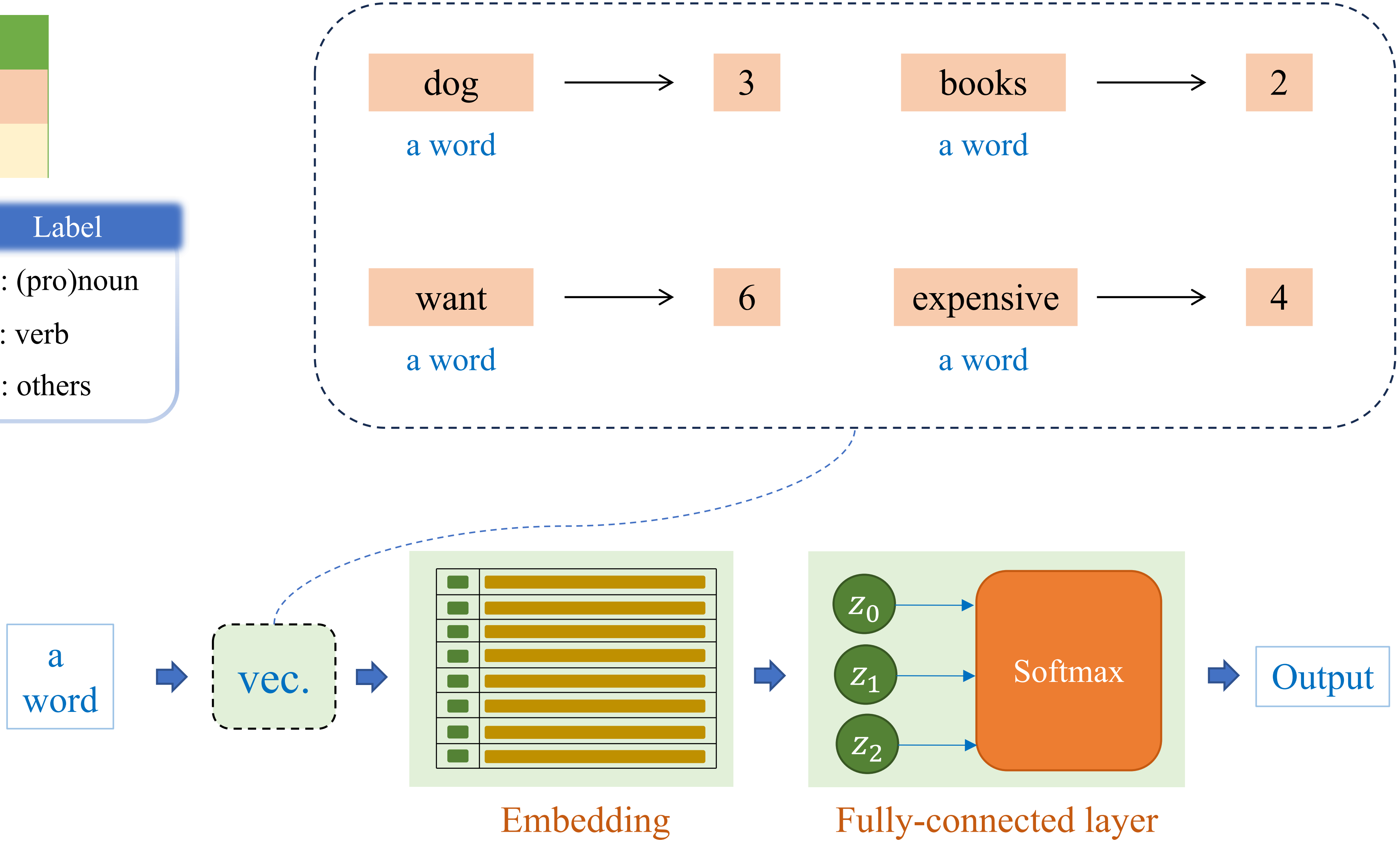
Doc	Label
i want a dog	[0, 1, 2, 0]
books are expensive	[0, 1, 2]

building dictionary

index	word
0	a
1	are
2	books
3	dog
4	expensive
5	i
6	want

vocab size = 7
word-based classification

Label
0: (pro)noun
1: verb
2: others



dog → 3
a word

books → 2
a word

want → 6
a word

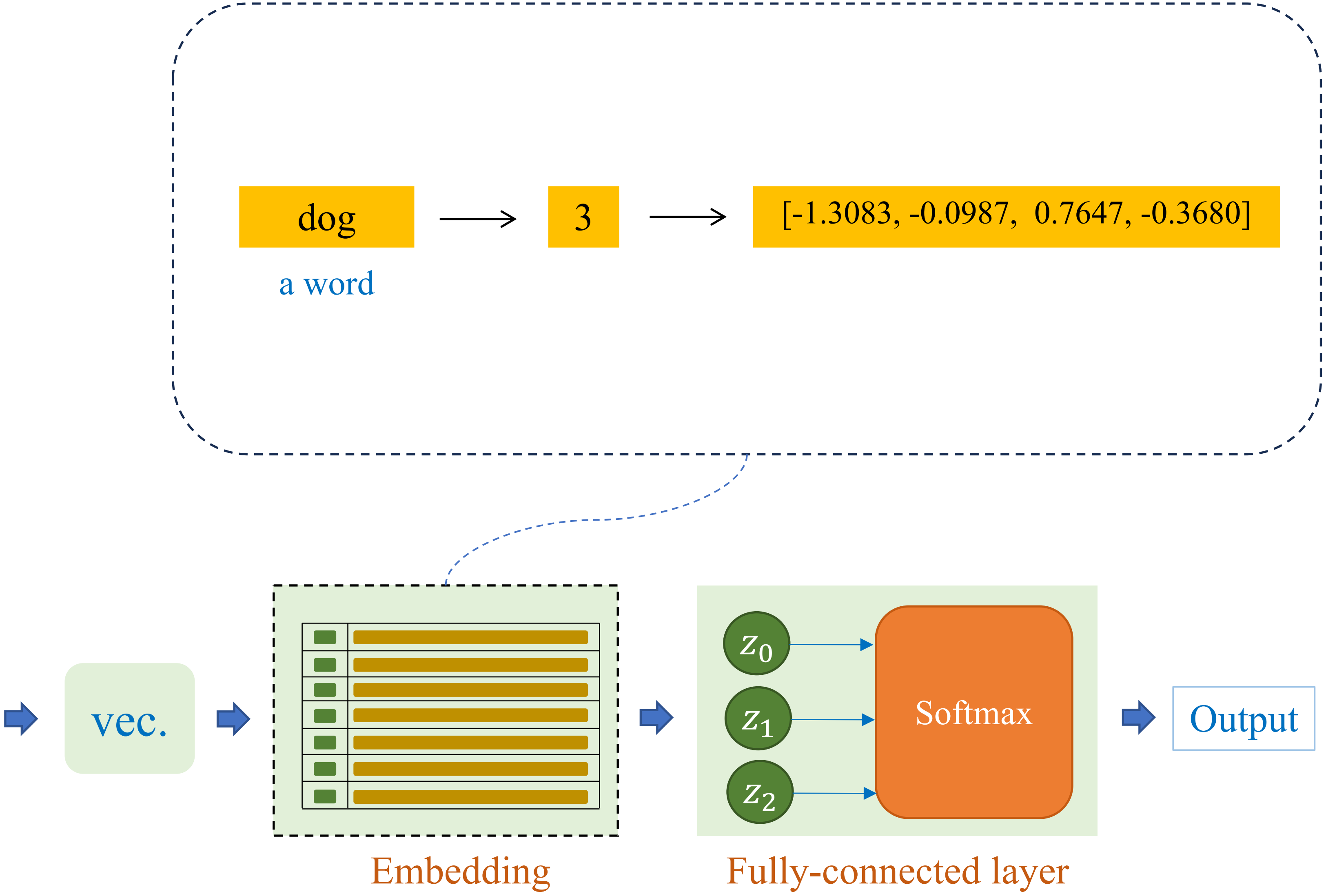
expensive → 4
a word

a word → vec. → Embedding → Fully-connected layer → Output

POS Tagging (1): One-Word Input

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

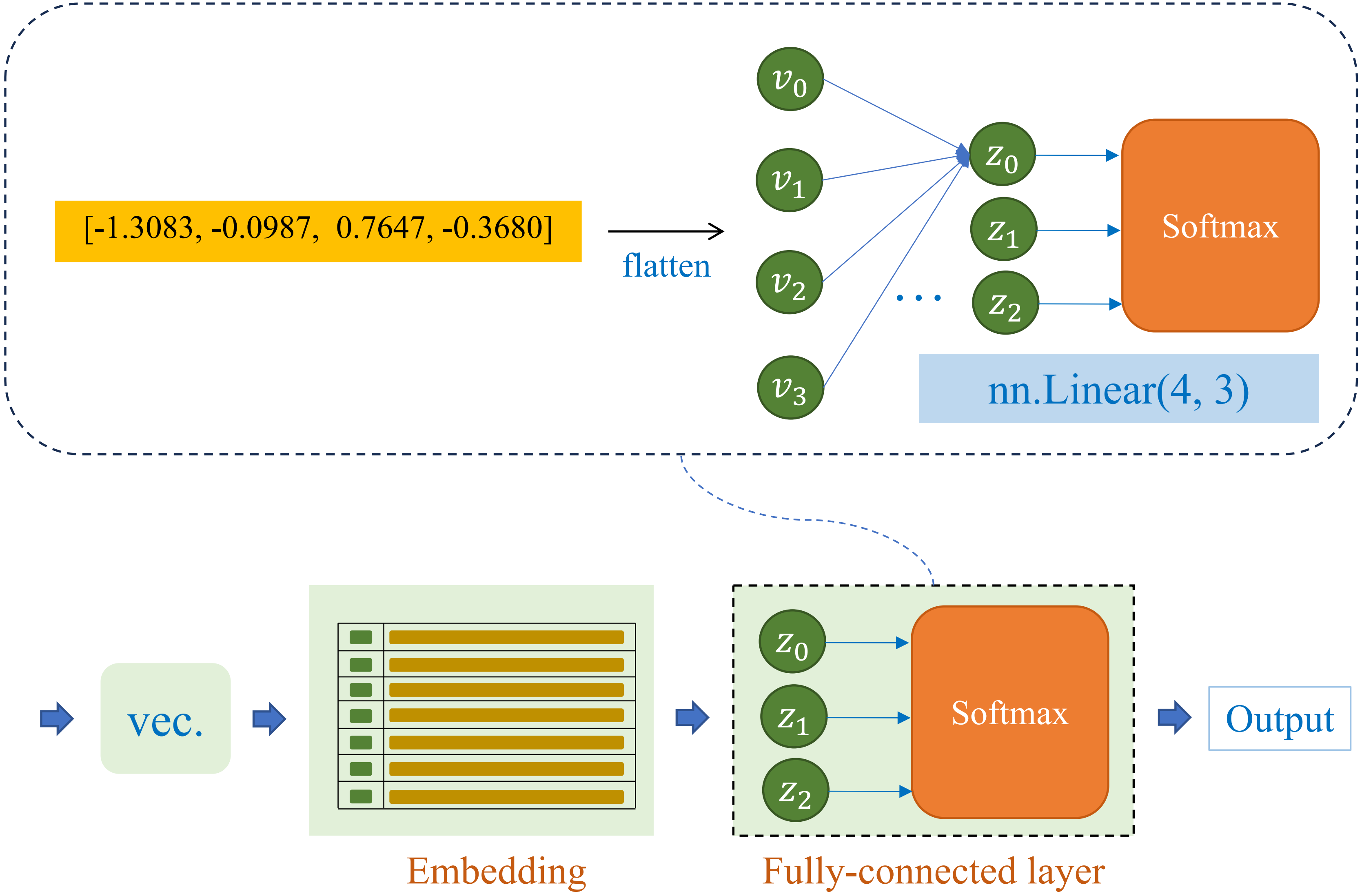
0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]



POS Tagging (1): One-Word Input

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

0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]



POS Tagging (1): One-Word Input

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

0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]

W

[0.3847, -0.4621, 0.1749, -0.0139]

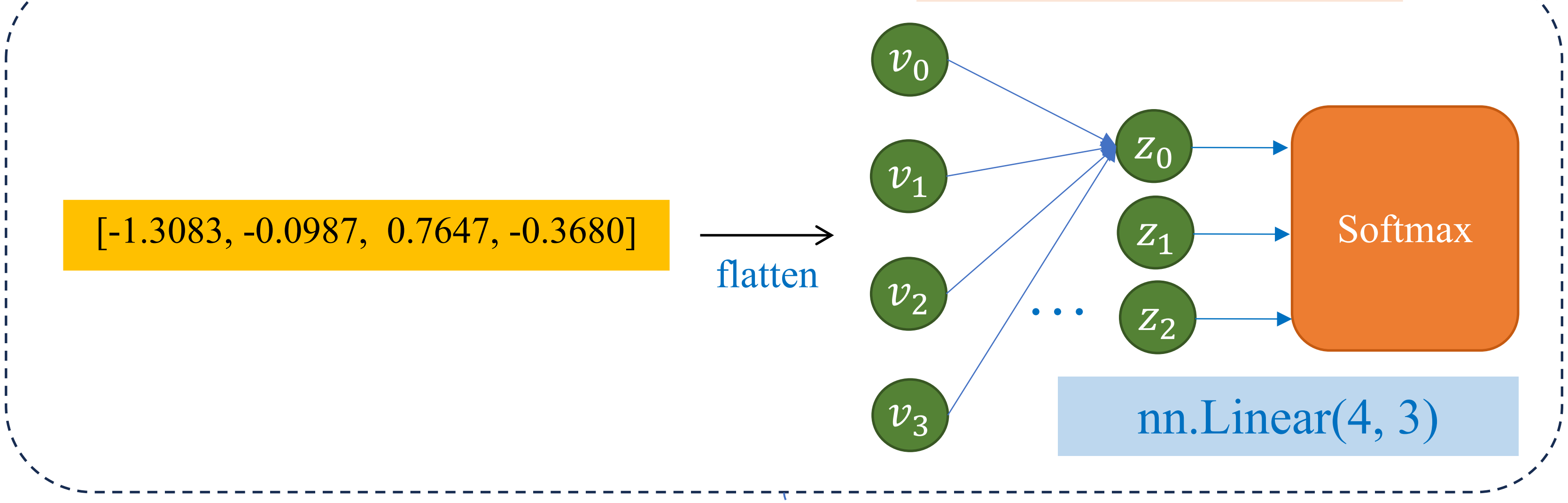
[-0.3024, -0.1529, 0.4329, 0.4254]

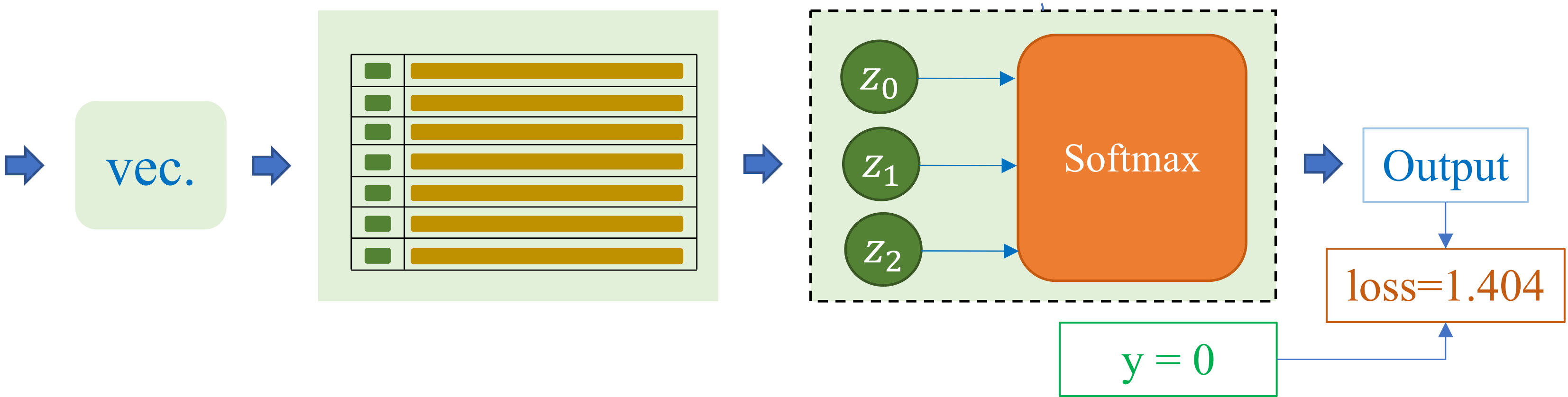
[-0.4441, 0.4113, 0.0054, -0.3220]

b [0.3548, -0.2819, -0.0579]

z [0.0360, 0.3033, 0.6051]

$\hat{y}$ [0.24, 0.32, 0.44]





POS Tagging (1): One-Word Input

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

W

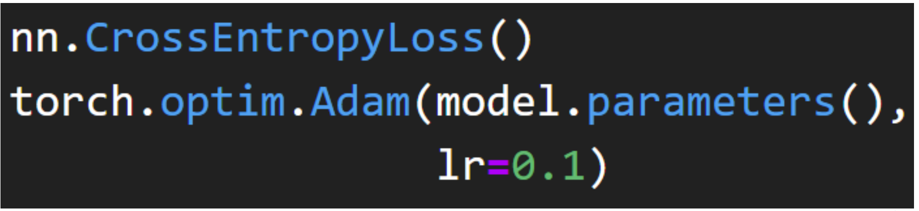
[0.2847, -0.5621, 0.2749, -0.1139]

[-0.2024, -0.0529, 0.3329, 0.5254]

[-0.3441, 0.5113, -0.0946, -0.2220]

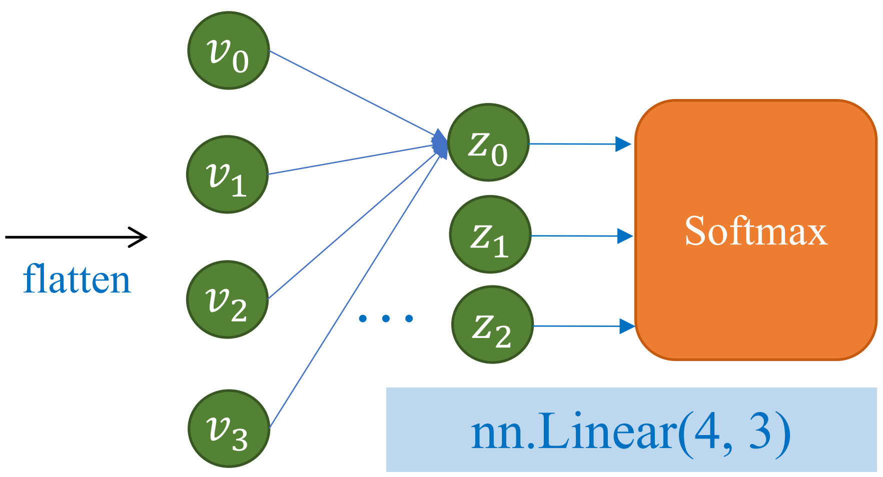
b

[0.4548, -0.3819, -0.1579]

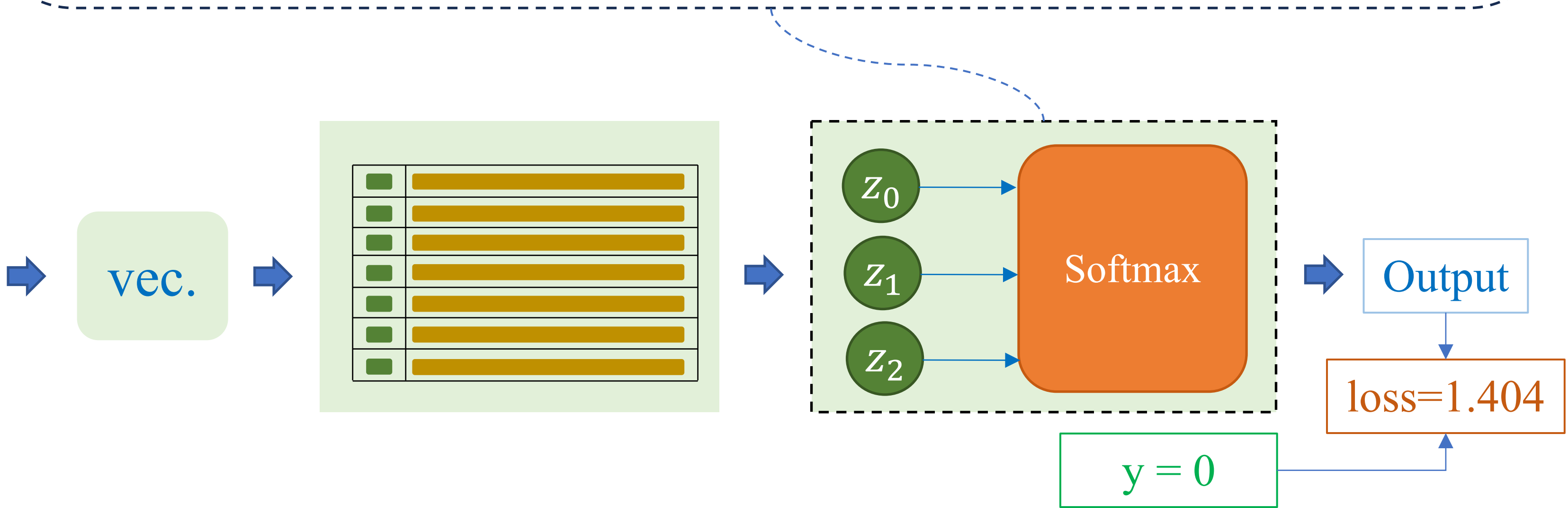


```
nn.CrossEntropyLoss()
torch.optim.Adam(model.parameters(),
                 lr=0.1)
```

[-1.2083, -0.1987, 0.6647, -0.4680]



0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.2083, -0.1987, 0.6647, -0.4680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]



POS Tagging (1): One-Word Input

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

0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]

W

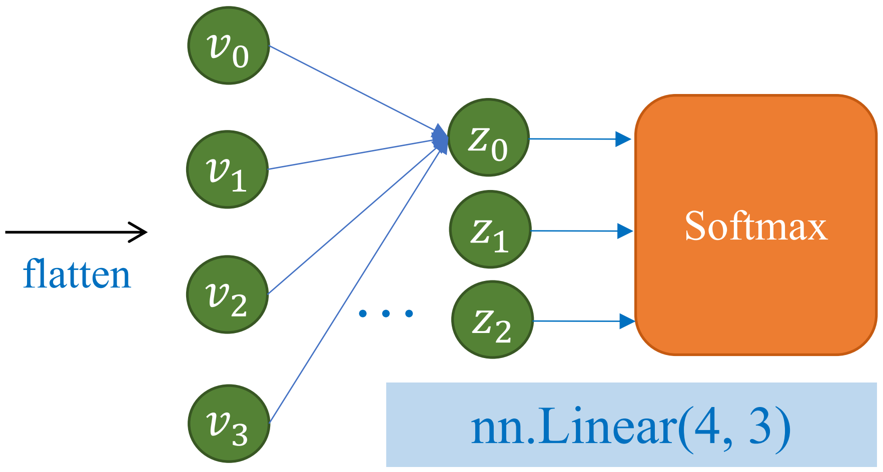
[0.2847, -0.5621, 0.2749, -0.1139]
[-0.2024, -0.0529, 0.3329, 0.5254]
[-0.3441, 0.5113, -0.0946, -0.2220]

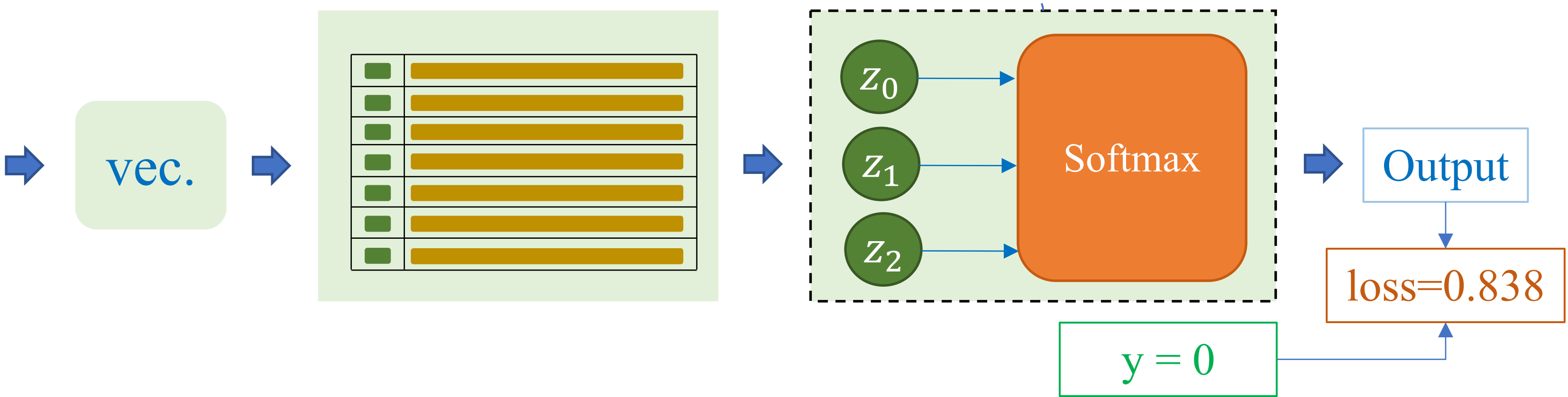
b [0.4548, -0.3819, -0.1579]

z [0.4585, -0.1514, 0.1973]

$\hat{y}$ [0.43, 0.23, 0.34]

[-1.3083, -0.0987, 0.7647, -0.3680]





Problem?

POS Tagging (2): Sentence + MLP

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

building dictionary

index	word
0	a
1	are
2	books
3	dog
4	expensive
5	i
6	quite
7	want

vocab size = 8
sequence length = 4

Label

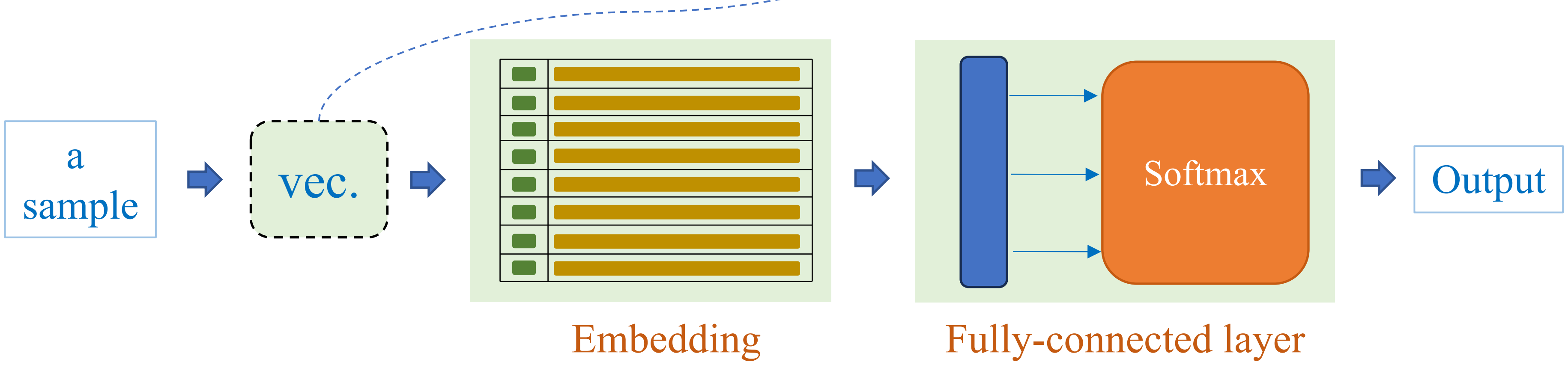
0: (pro)noun

1: verb

2: others

sample 1	→
i	5
want	7
a	0
dog	3

sample 2	→
books	2
are	1
quite	6
expensive	4

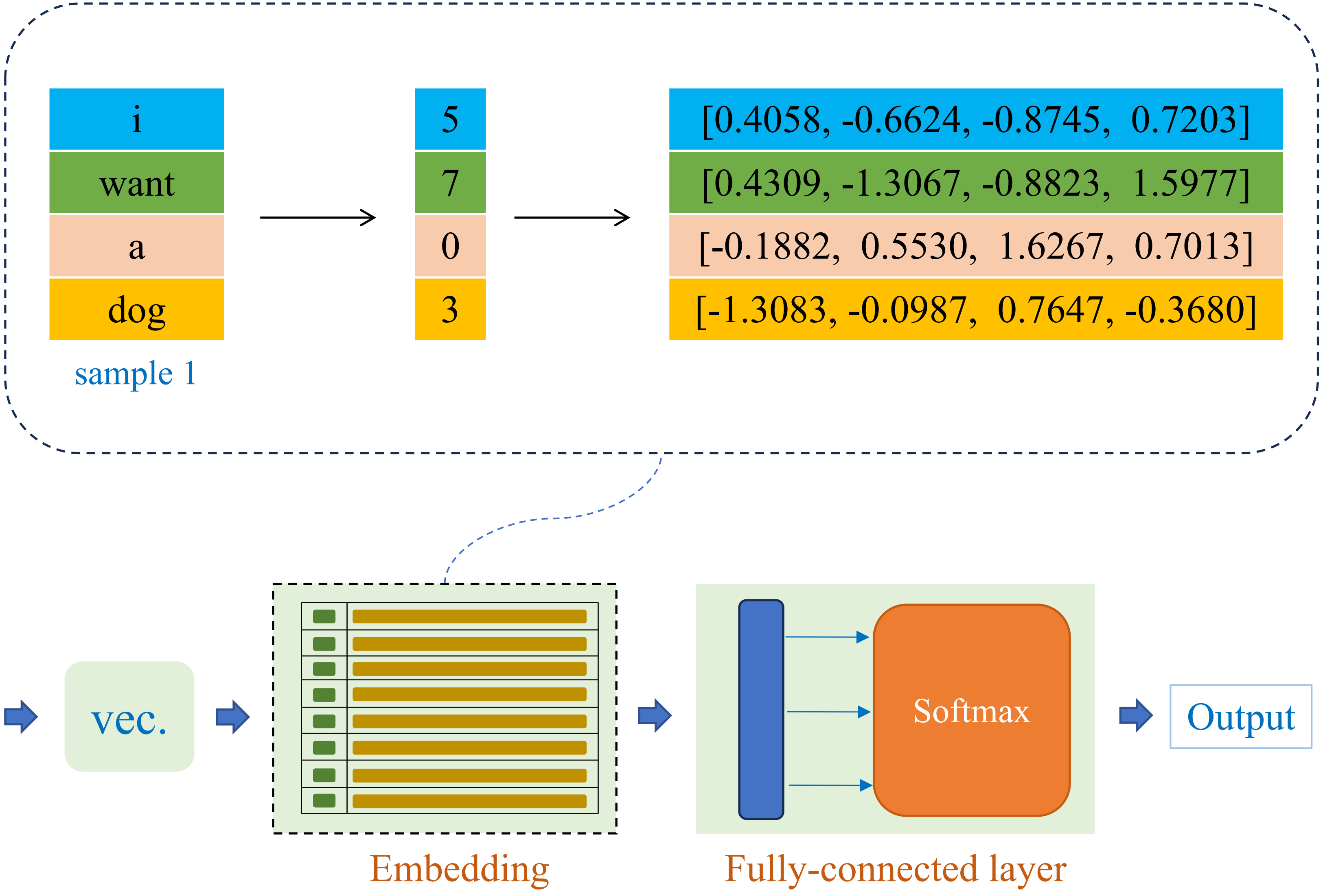


POS Tagging (2): Sentence + MLP

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

0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]
7	[0.4309, -1.3067, -0.8823, 1.5977]

vocab size = 8
sequence length = 4



i	5	[0.4058, -0.6624, -0.8745, 0.7203]
want	7	[0.4309, -1.3067, -0.8823, 1.5977]
a	0	[-0.1882, 0.5530, 1.6267, 0.7013]
dog	3	[-1.3083, -0.0987, 0.7647, -0.3680]

W

[0.3847, -0.4621, 0.1749, -0.0139]
[-0.3024, -0.1529, 0.4329, 0.4254]
[-0.4441, 0.4113, 0.0054, -0.3220]

b

[0.3548, -0.2819, -0.0579]

$\hat{y}$

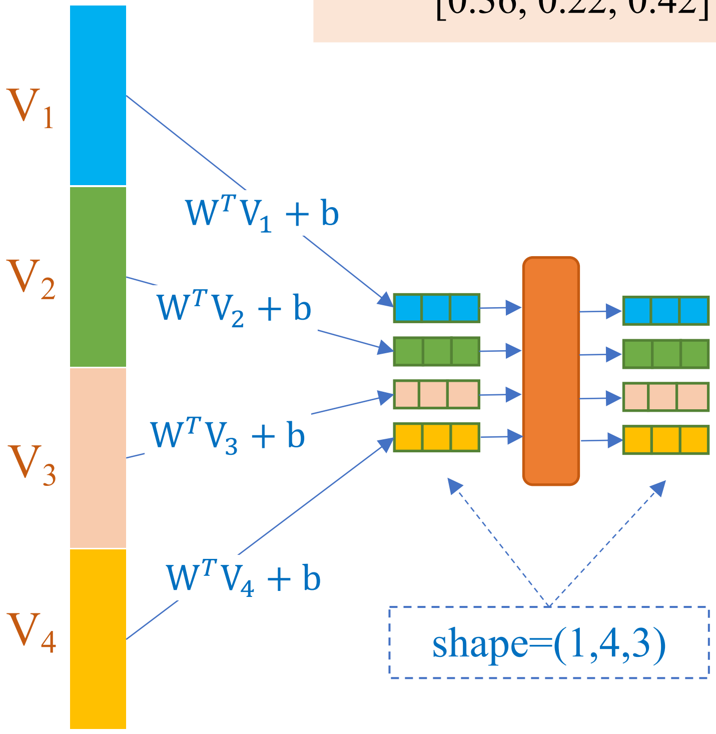
[0.58, 0.26, 0.16]
[0.47, 0.44, 0.09]
[0.35, 0.29, 0.36]
[0.36, 0.22, 0.42]

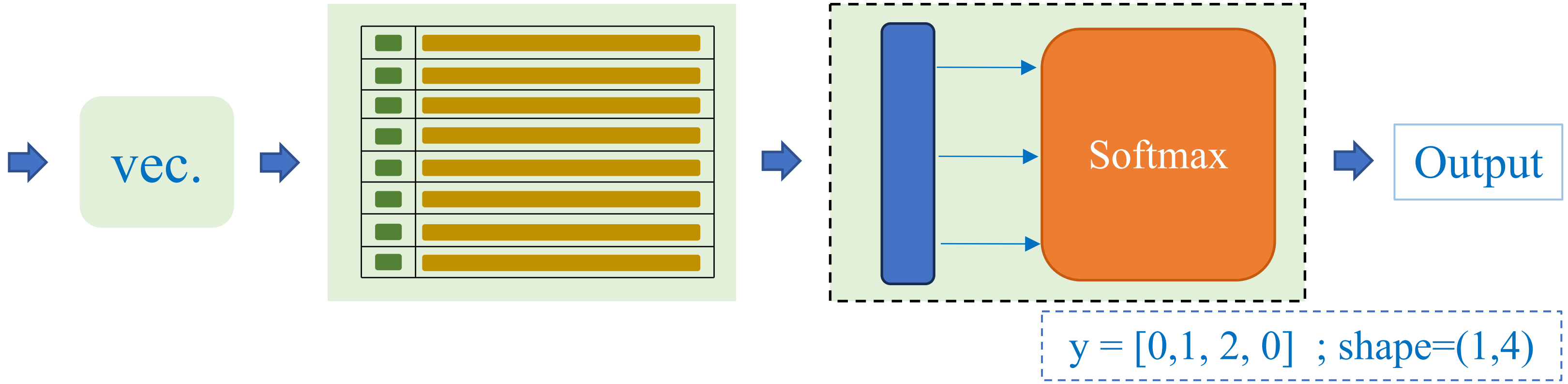
Doc	Label
i want a dog	[0, 1, 2, 0]
books are quite expensive	[0, 1, 2, 2]

0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]
7	[0.4309, -1.3067, -0.8823, 1.5977]

vocab size = 8
sequence length = 4

[0.4058, -0.6624, -0.8745, 0.7203]
[0.4309, -1.3067, -0.8823, 1.5977]
[-0.1882, 0.5530, 1.6267, 0.7013]
[-1.3083, -0.0987, 0.7647, -0.3680]





W

[0.3847, -0.4621, 0.1749, -0.0139]
[-0.3024, -0.1529, 0.4329, 0.4254]
[-0.4441, 0.4113, 0.0054, -0.3220]

b

[0.3548, -0.2819, -0.0579]

$\hat{y}$

[0.58, 0.47, 0.35, 0.36]
[0.26, 0.44, 0.29, 0.22]
[0.16, 0.22, 0.36, 0.42]

Shape of logits = (N, C, d)
Shape of target = (N, d)

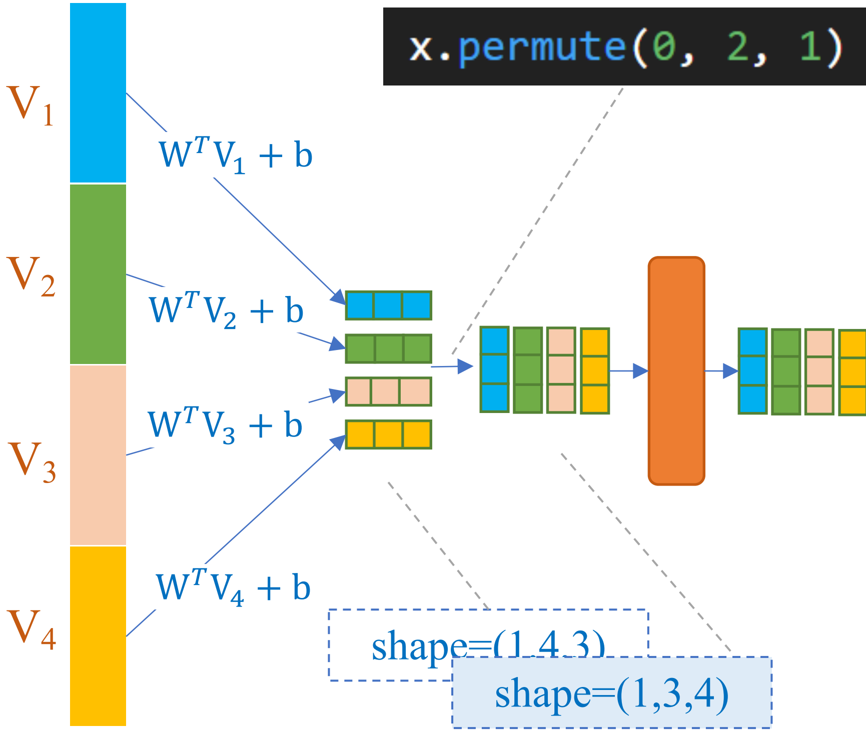
pytorch requirement

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

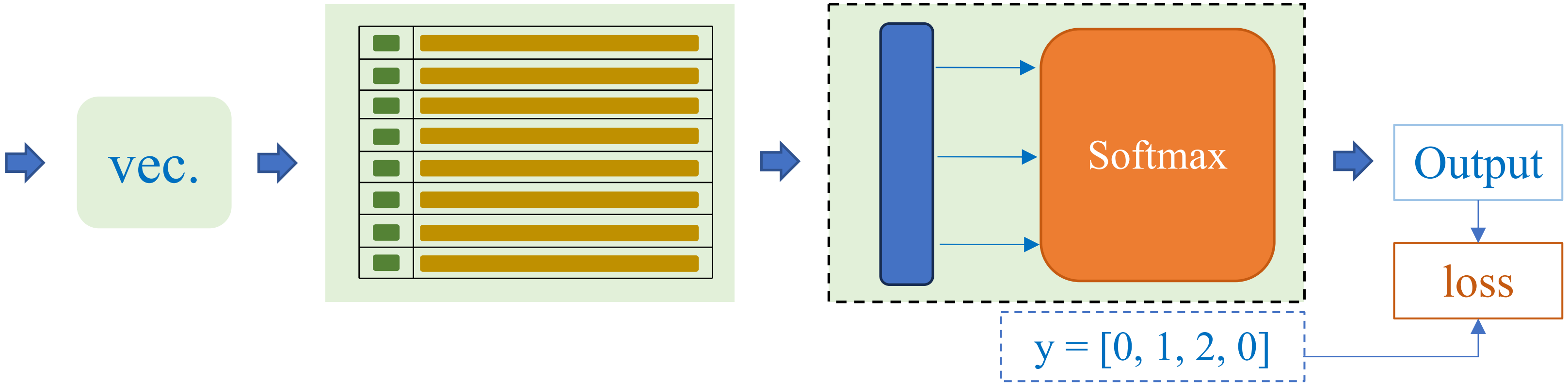
0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]
7	[0.4309, -1.3067, -0.8823, 1.5977]

vocab size = 8
sequence length = 4

[0.4058, -0.6624, -0.8745, 0.7203]
[0.4309, -1.3067, -0.8823, 1.5977]
[-0.1882, 0.5530, 1.6267, 0.7013]
[-1.3083, -0.0987, 0.7647, -0.3680]



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



W

[0.3847, -0.4621, 0.1749, -0.0139]

[-0.3024, -0.1529, 0.4329, 0.4254]

[-0.4441, 0.4113, 0.0054, -0.3220]

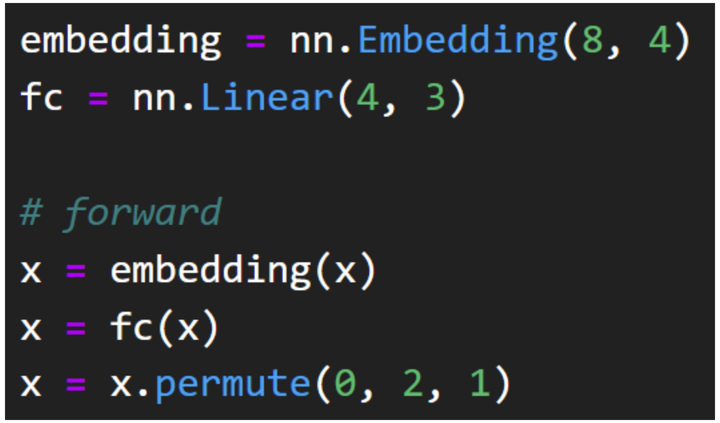
b

[0.3548, -0.2819, -0.0579]

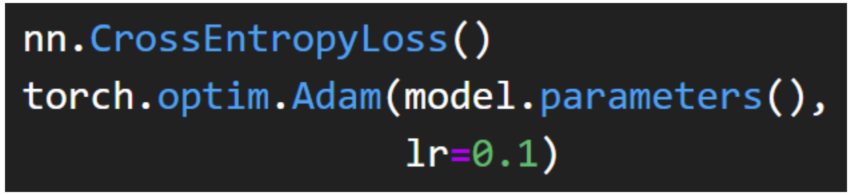
Doc	Label
i want a dog	[0, 1, 2, 0]
books are quite expensive	[0, 1, 2, 2]

0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]
7	[0.4309, -1.3067, -0.8823, 1.5977]

vocab size = 8
sequence length = 4



```
embedding = nn.Embedding(8, 4)
fc = nn.Linear(4, 3)

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



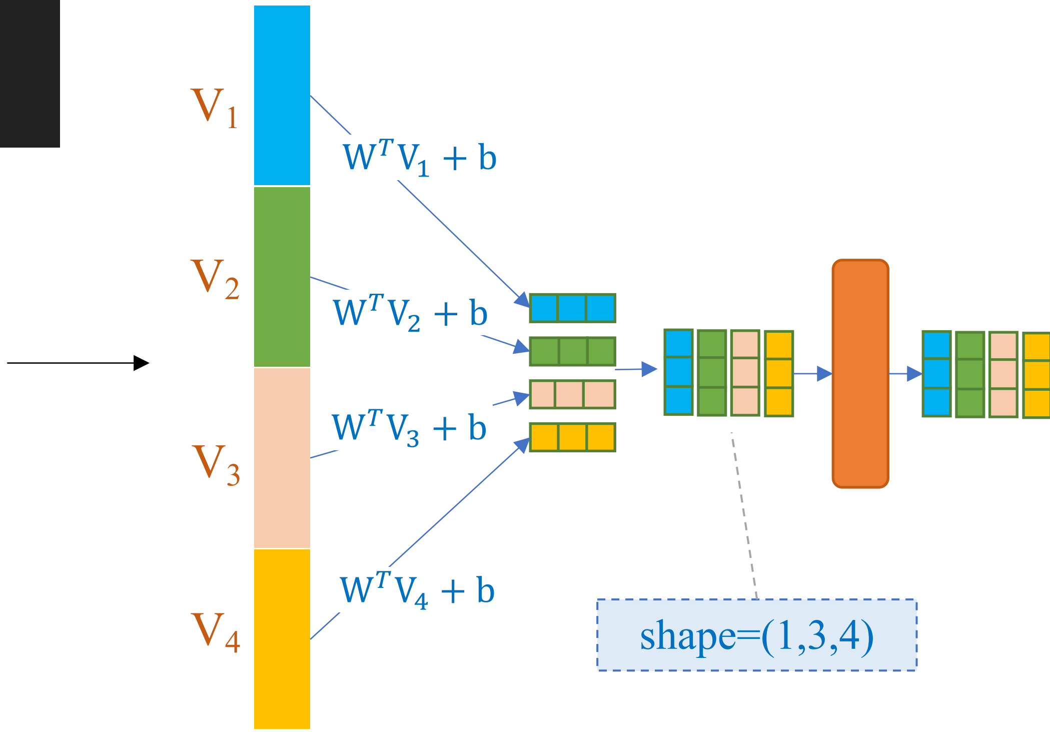
```
nn.CrossEntropyLoss()
torch.optim.Adam(model.parameters(),
                 lr=0.1)
```

[0.4058, -0.6624, -0.8745, 0.7203]

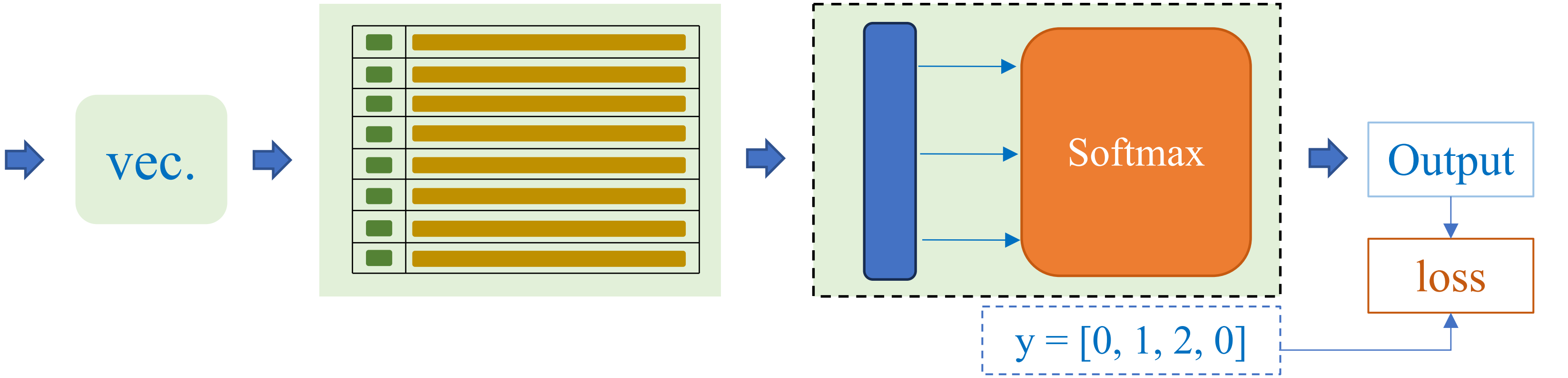
[0.4309, -1.3067, -0.8823, 1.5977]

[-0.1882, 0.5530, 1.6267, 0.7013]

[-1.3083, -0.0987, 0.7647, -0.3680]



Problem?



POS Tagging (3): Using Padding

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

building dictionary

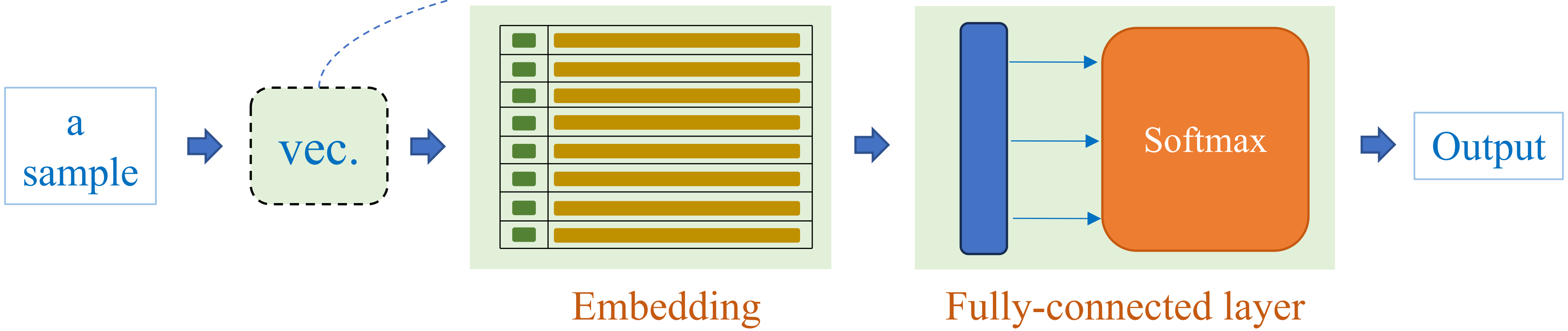
index	word
0	[UNK]
1	[pad]
2	a
3	are
4	books
5	dog
6	expensive
7	i
8	want

vocab size = 9
sequence length = 4

Label
0: (pro)noun
1: verb
2: others

sample 1		vec 1	label 1
i		7	0
want	→	8	1
a		2	2
dog		5	0

sample 2		vec 2	label 2
books		4	0
are	→	3	1
expensive		6	2
<pad>		1	?



POS Tagging (3): Using Padding

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

building dictionary

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

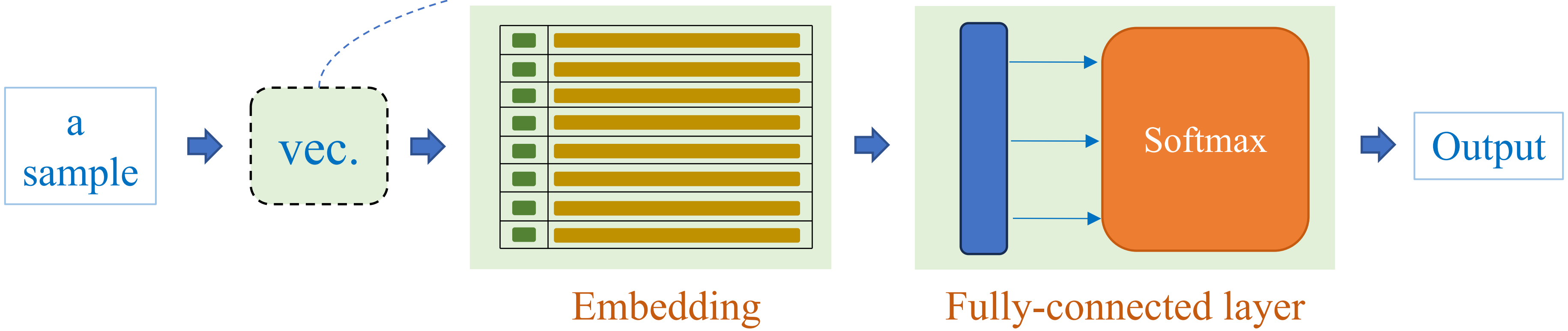
vocab size = 9
sequence length = 4

Label

0: (pro)noun
1: verb
2: others
3: <pad>

sample 1	→	vec 1	label 1
i		7	0
want		8	1
a		2	2
dog		5	0

sample 2	→	vec 2	label 2
books		4	0
are		3	1
expensive		6	2
<pad>		1	3



POS Tagging (3): Using Padding

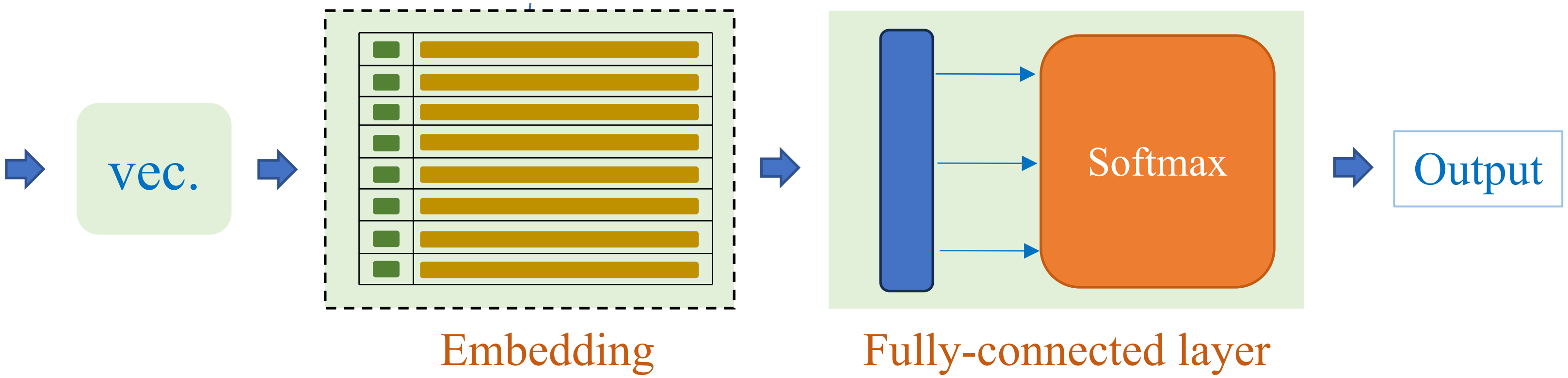
Doc	Label
i want a dog	[0, 1, 2, 0]
books are expensive	[0, 1, 2]

0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]
7	[0.4309, -1.3067, -0.8823, 1.5977]
8	[0.3058, -0.7624, -0.7745, 0.6203]

vocab size = 9
sequence length = 4

i	→	7	→	[0.4058, -0.6624, -0.8745, 0.7203]
want		8		[0.3058, -0.7624, -0.7745, 0.6203]
a		2		[1.0281, -1.9094, 0.3182, 0.4211]
dog		5		[0.4058, -0.6624, -0.8745, 0.7203]

sample 1



W

[-0.3875, -0.3519, -0.1275, -0.1719]

[0.4391, 0.0455, -0.1566, -0.2897]

[0.1777, -0.1178, -0.3101, -0.2451]

[0.3730, 0.0996, -0.3004, 0.2219]

b

[0.3548, -0.2819, -0.0579, 0.5113]

$\hat{y}$

[0.26, 0.09, 0.16, 0.49]

[0.27, 0.13, 0.19, 0.41]

[0.29, 0.15, 0.21, 0.35]

[0.24, 0.13, 0.19, 0.44]

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

Label

0: (pro)noun　　2: others

1: verb　　3: <pad>

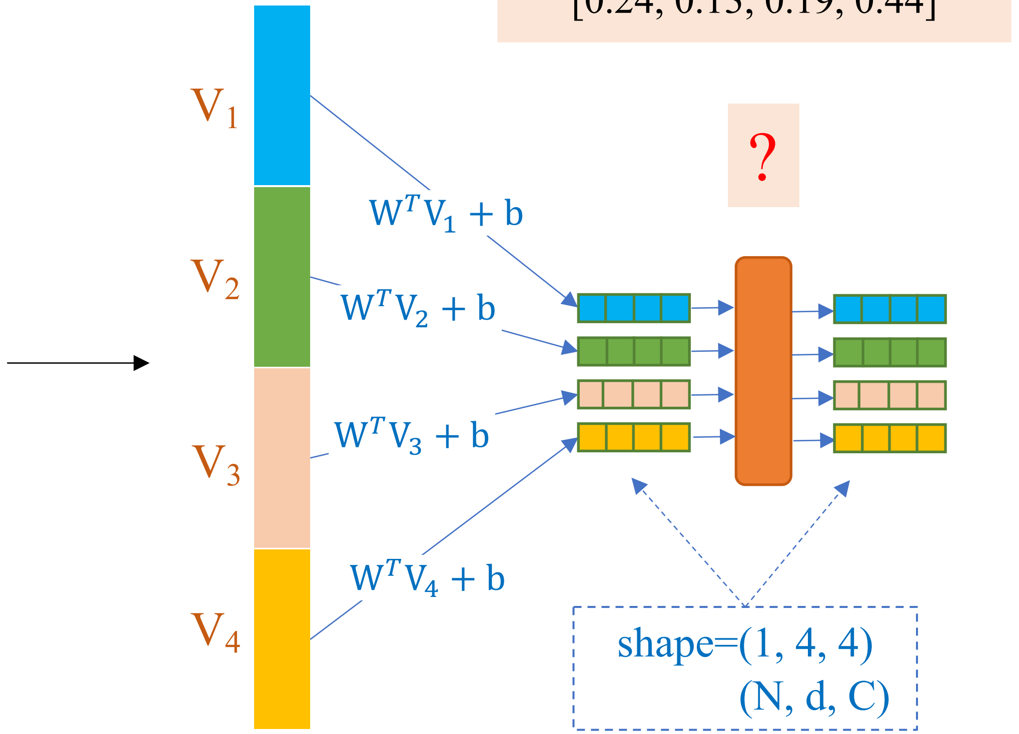
0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]
7	[0.4309, -1.3067, -0.8823, 1.5977]
8	[0.3058, -0.7624, -0.7745, 0.6203]

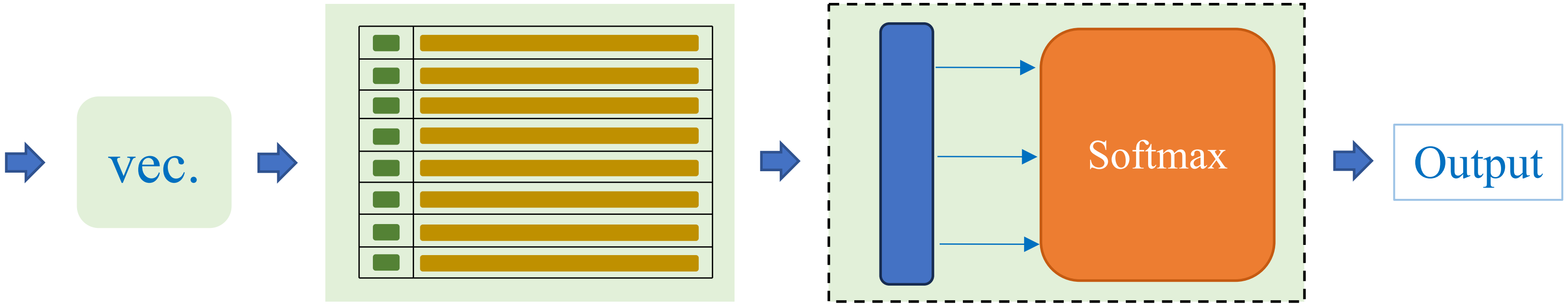
[0.4058, -0.6624, -0.8745, 0.7203]

[0.3058, -0.7624, -0.7745, 0.6203]

[1.0281, -1.9094, 0.3182, 0.4211]

[0.4058, -0.6624, -0.8745, 0.7203]





y = [0,1, 2, 0] ; shape=(1,4)

vocab size = 9
sequence length = 4

W

[-0.3875, -0.3519, -0.1275, -0.1719]

[0.4391, 0.0455, -0.1566, -0.2897]

[0.1777, -0.1178, -0.3101, -0.2451]

[0.3730, 0.0996, -0.3004, 0.2219]

b

[0.3548, -0.2819, -0.0579, 0.5113]

Shape of logits = (N, C, d)
Shape of target = (N, d)

pytorch requirement

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

Label

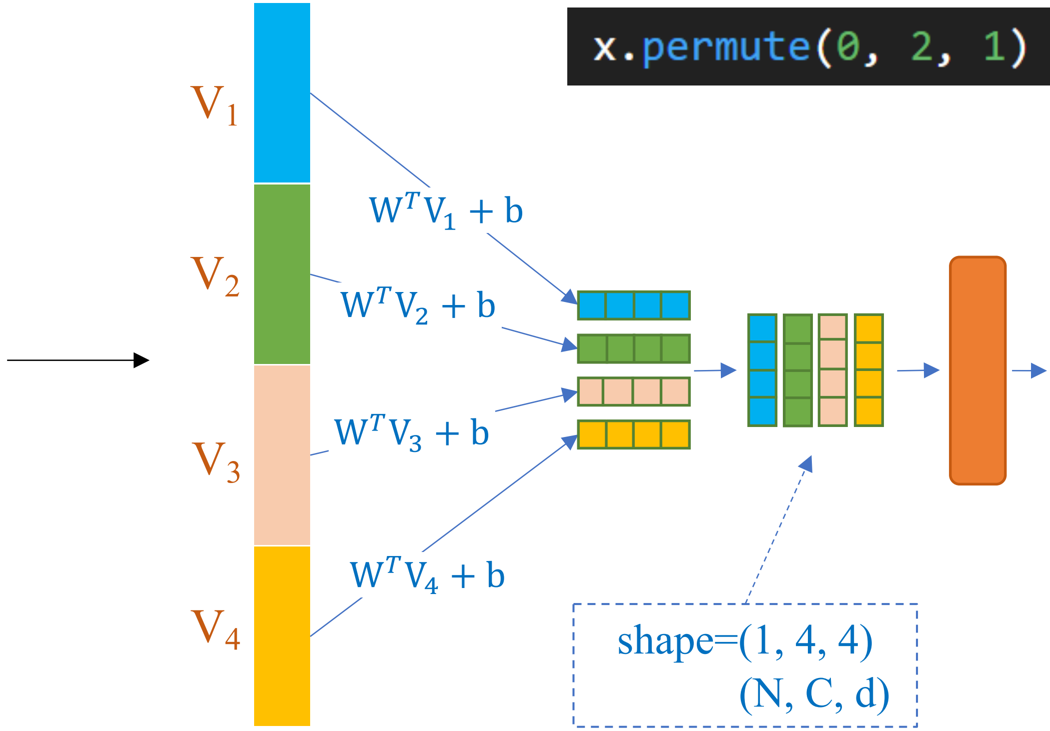
0: (pro)noun
1: verb
2: others
3: <pad>

[0.4058, -0.6624, -0.8745, 0.7203]

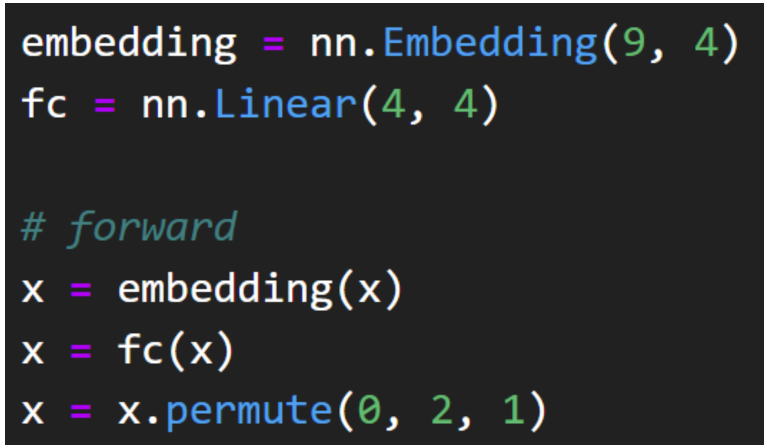
[0.3058, -0.7624, -0.7745, 0.6203]

[1.0281, -1.9094, 0.3182, 0.4211]

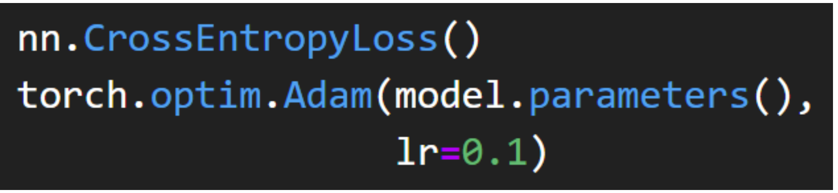
[0.4058, -0.6624, -0.8745, 0.7203]



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

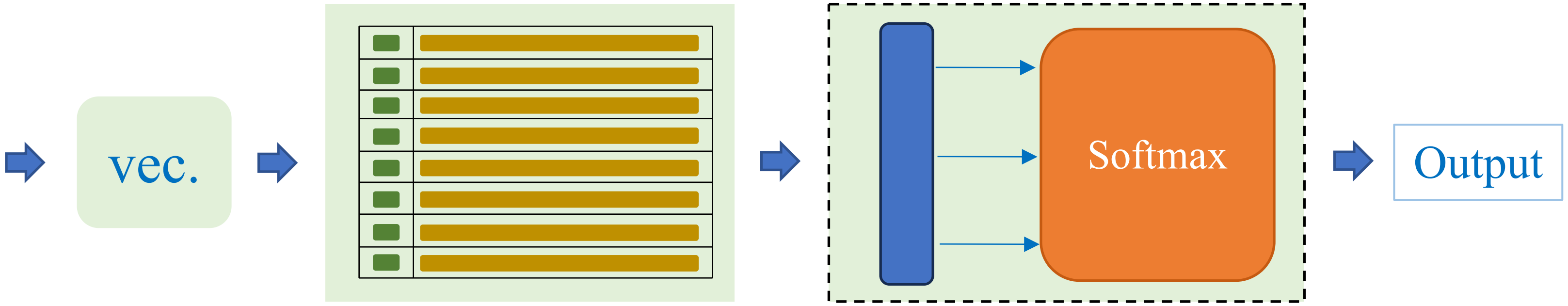


```
embedding = nn.Embedding(9, 4)
fc = nn.Linear(4, 4)

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



```
nn.CrossEntropyLoss()
torch.optim.Adam(model.parameters(),
                 lr=0.1)
```

problem?



y = [0, 1, 2, 0] ; shape=(1,4)

vocab size = 9
sequence length = 4

W

[-0.3875, -0.3519, -0.1275, -0.1719]
[0.4391, 0.0455, -0.1566, -0.2897]
[0.1777, -0.1178, -0.3101, -0.2451]
[0.3730, 0.0996, -0.3004, 0.2219]

b

[0.3548, -0.2819, -0.0579, 0.5113]

Shape of logits = (N, C, d)
Shape of target = (N, d)

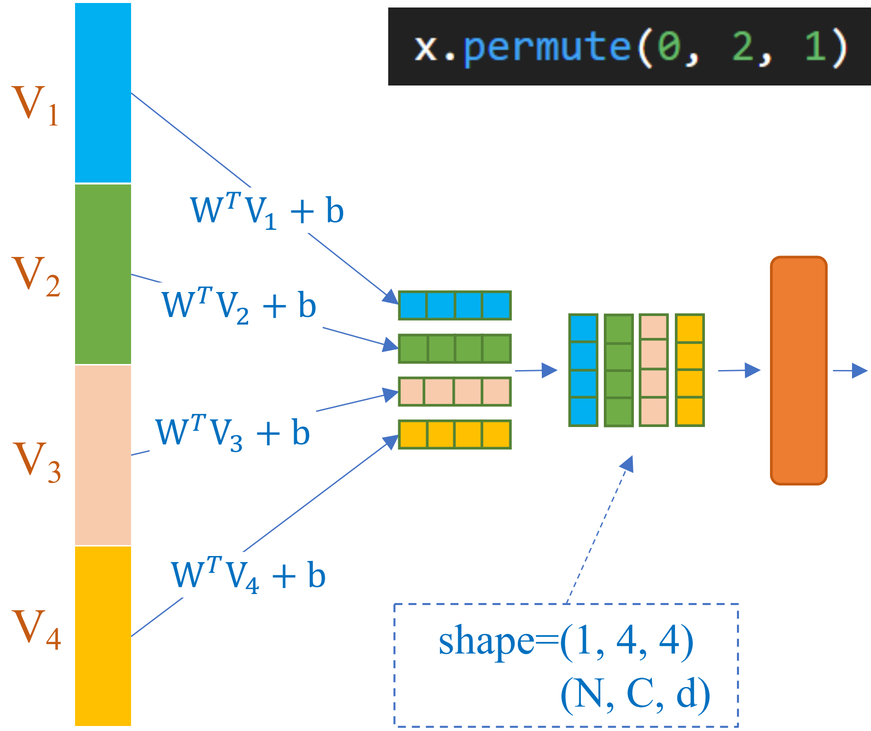
pytorch requirement

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

Label

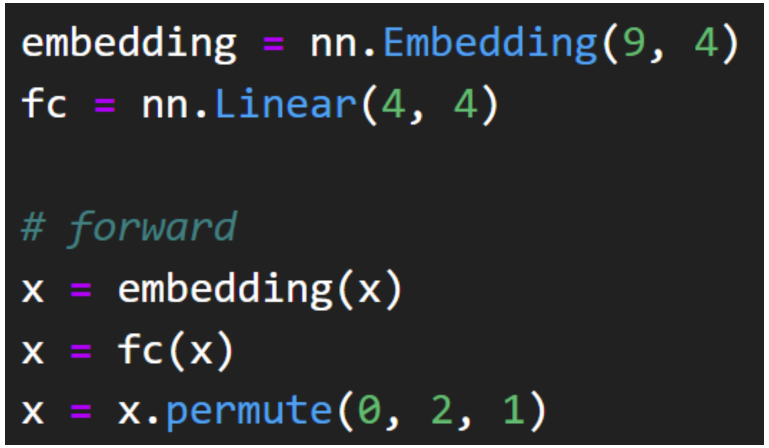
0: (pro)noun
1: verb
2: others
3: <pad>

[0.4058, -0.6624, -0.8745, 0.7203]
[0.3058, -0.7624, -0.7745, 0.6203]
[1.0281, -1.9094, 0.3182, 0.4211]
[0.4058, -0.6624, -0.8745, 0.7203]

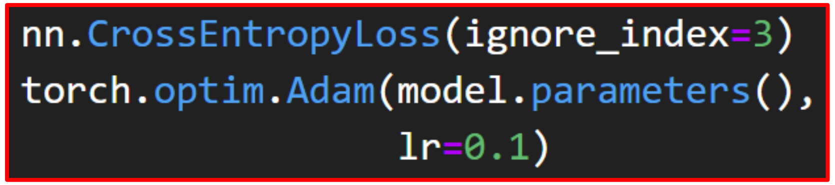


```
x.permute(0, 2, 1)
```

shape=(1, 4, 4)
(N, C, d)

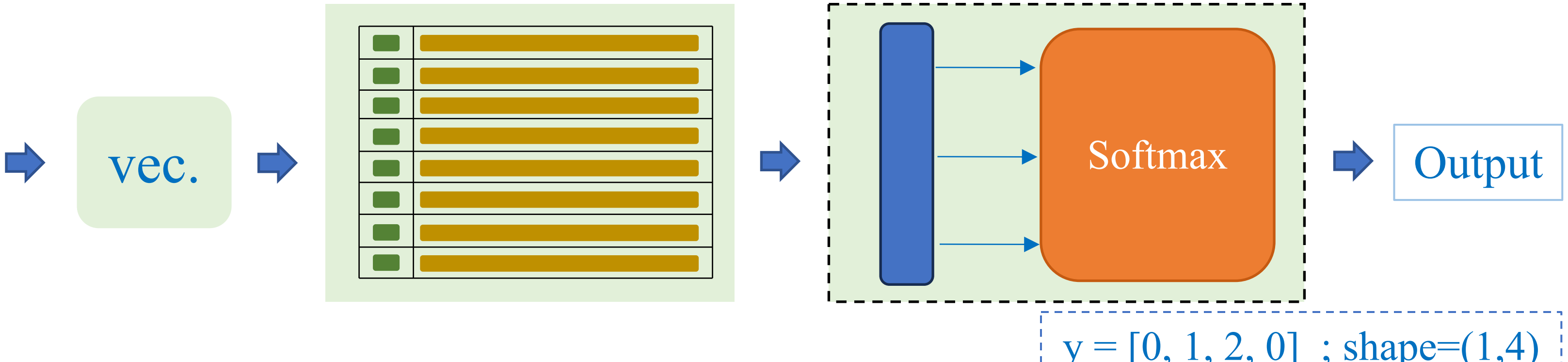


```
embedding = nn.Embedding(9, 4)
fc = nn.Linear(4, 4)

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



```
nn.CrossEntropyLoss(ignore_index=3)
torch.optim.Adam(model.parameters(),
                 lr=0.1)
```

vocab size = 9
sequence length = 4

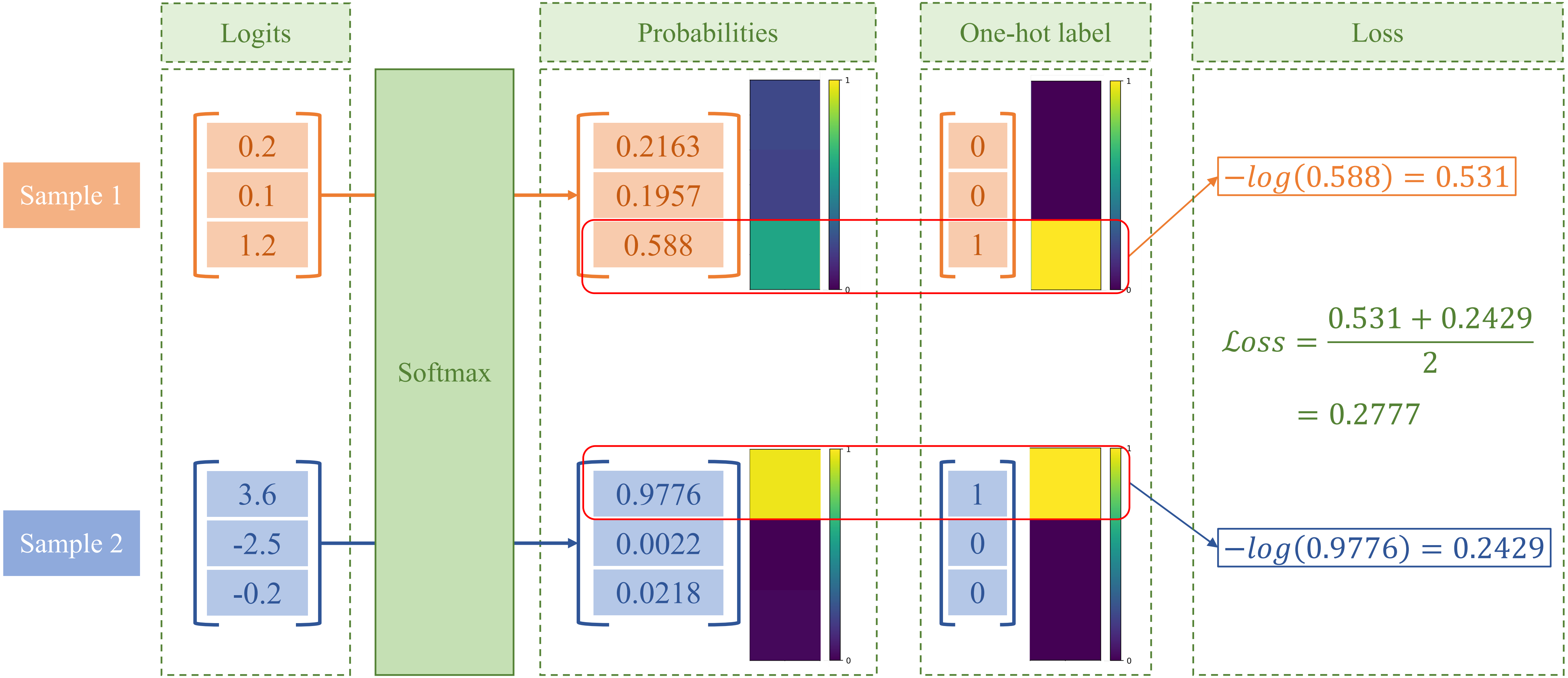


y = [0, 1, 2, 0] ; shape=(1,4)

Cross Entropy Loss

N_classes = 3

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



$$-log(0.588) = 0.531$$

$$\mathcal{L}oss = \frac{0.531 + 0.2429}{2}$$

$$= 0.2777$$

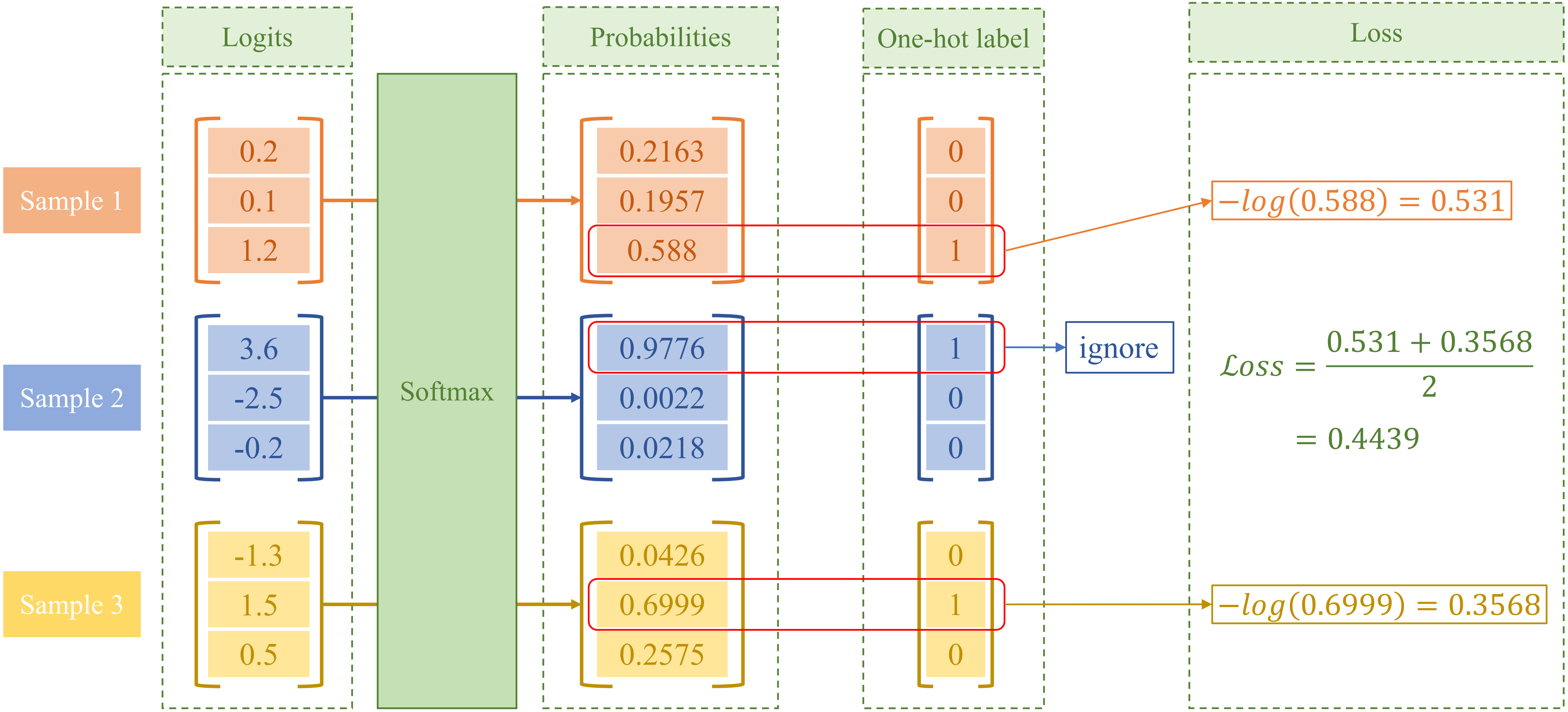
$$-log(0.9776) = 0.2429$$

Cross Entropy Loss

N_classes = 3

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

Ignore_index = 0



	Logits		Probabilities	One-hot label
Sample 1	0.2	Softmax	0.2163	0
	0.1		0.1957	0
	1.2		0.588	1
Sample 2	3.6		0.9776	1
	-2.5		0.0022	0
	-0.2		0.0218	0
Sample 3	-1.3		0.0426	0
	1.5		0.6999	1
	0.5		0.2575	0

ignore

Loss

$$-log(0.588) = 0.531$$

$$\mathcal{L}oss = \frac{0.531 + 0.3568}{2}$$

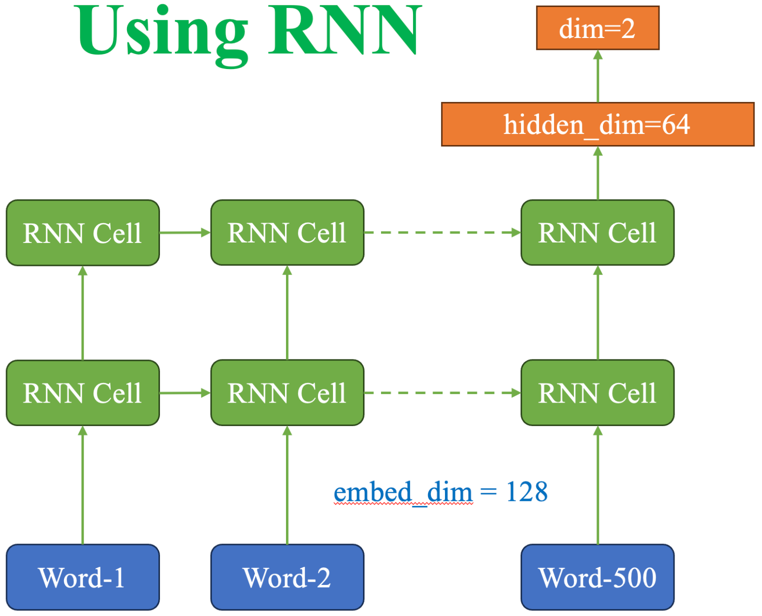
$$= 0.4439$$

$$-log(0.6999) = 0.3568$$

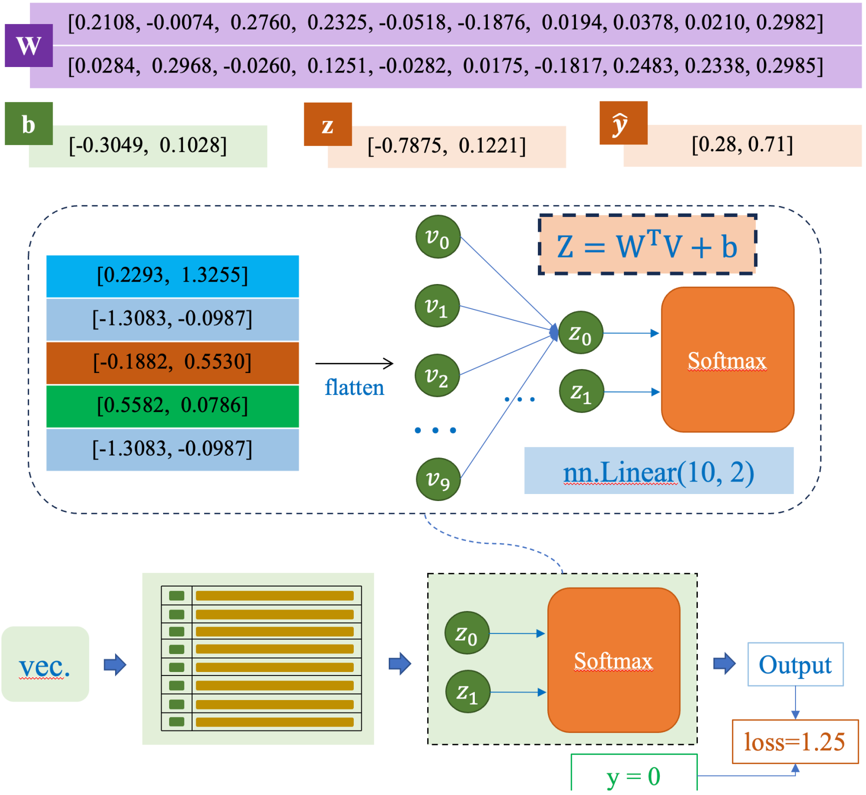
Summary

Review

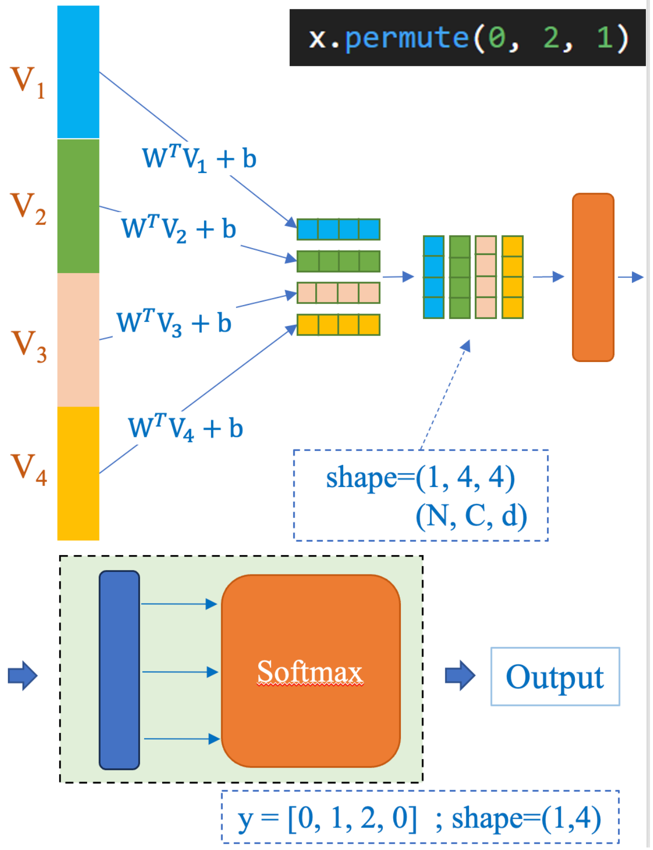
Using RNN



Text Classification



POS Tagging


x.permute(0, 2, 1)
V_1
V_2
V_3
V_4
$W^T V_1 + b$
$W^T V_2 + b$
$W^T V_3 + b$
$W^T V_4 + b$
shape=(1, 4, 4)
(N, C, d)
Softmax
Output
y = [0, 1, 2, 0] ; shape=(1,4)