AI VIETNAM
All-in-One Course

From Text Generation to Machine Translation

Quang-Vinh Dinh
Ph.D. in Computer Science

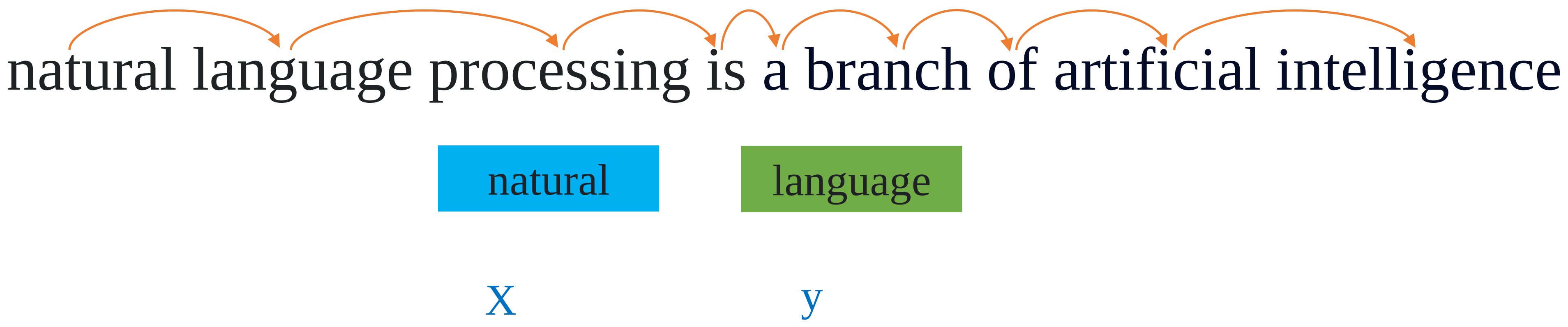
Year 2024

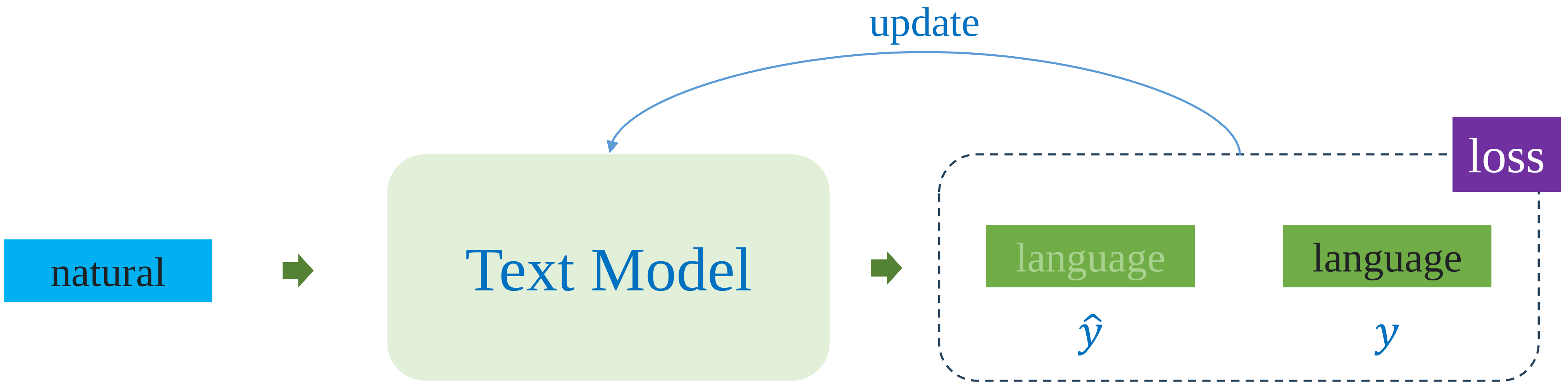
Outline

- Text Generation Using Transformer
- An improved Approach to Text Generation
- Machine Translation Using RNN
- Machine Translation Using Transformer

Self-Supervision Using Text Data

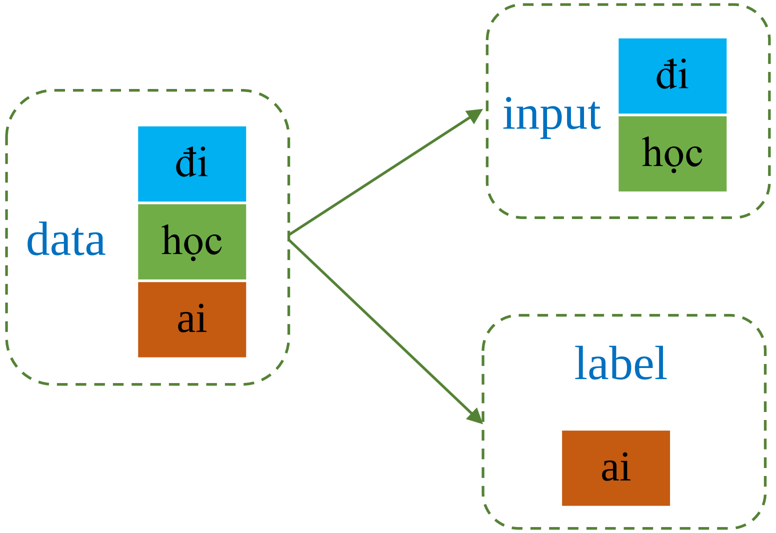
❖ Encode the sequential relationship

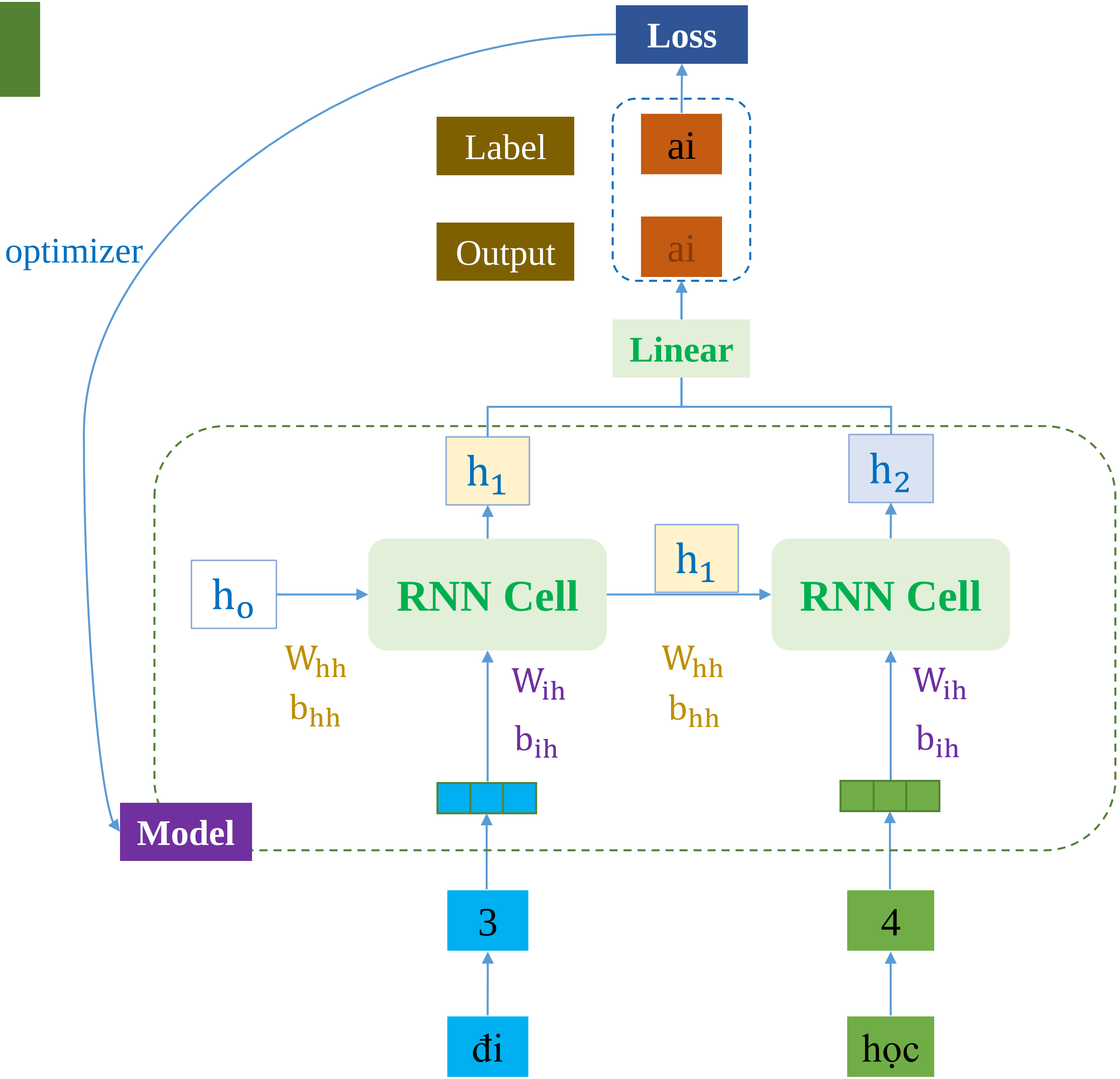

natural language processing is a branch of artificial intelligence



Implementation Using RNN

Using all the features

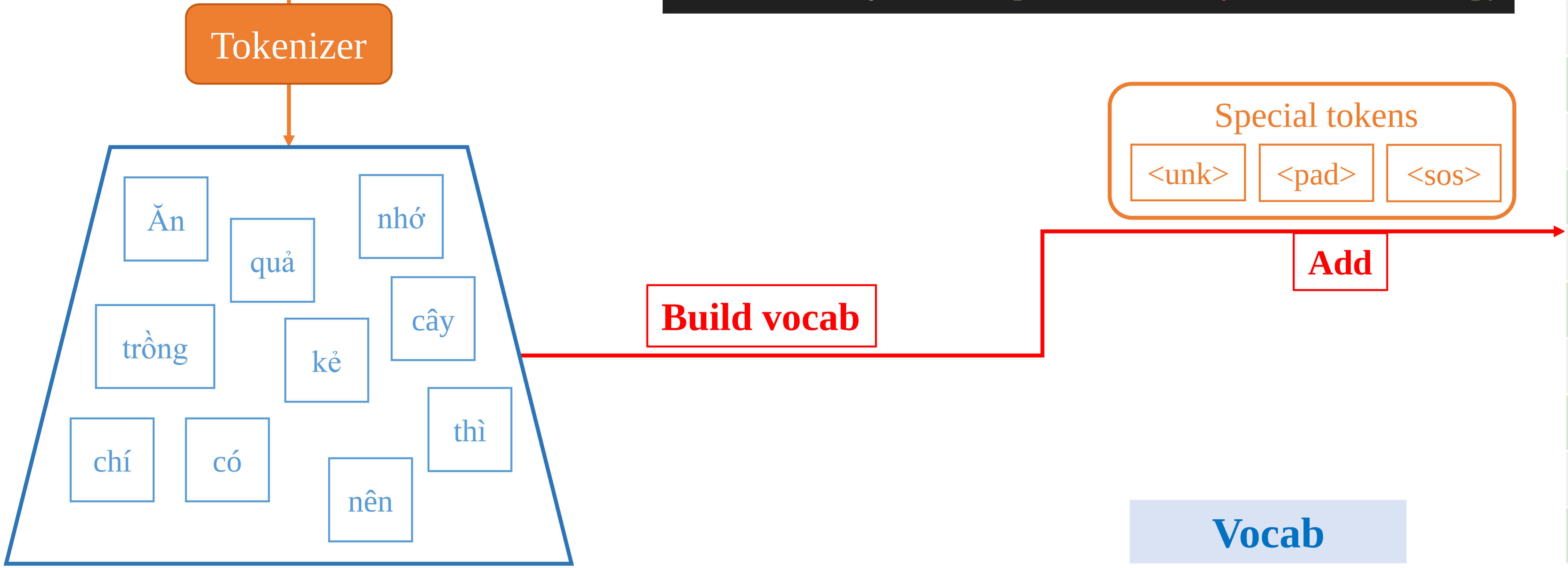


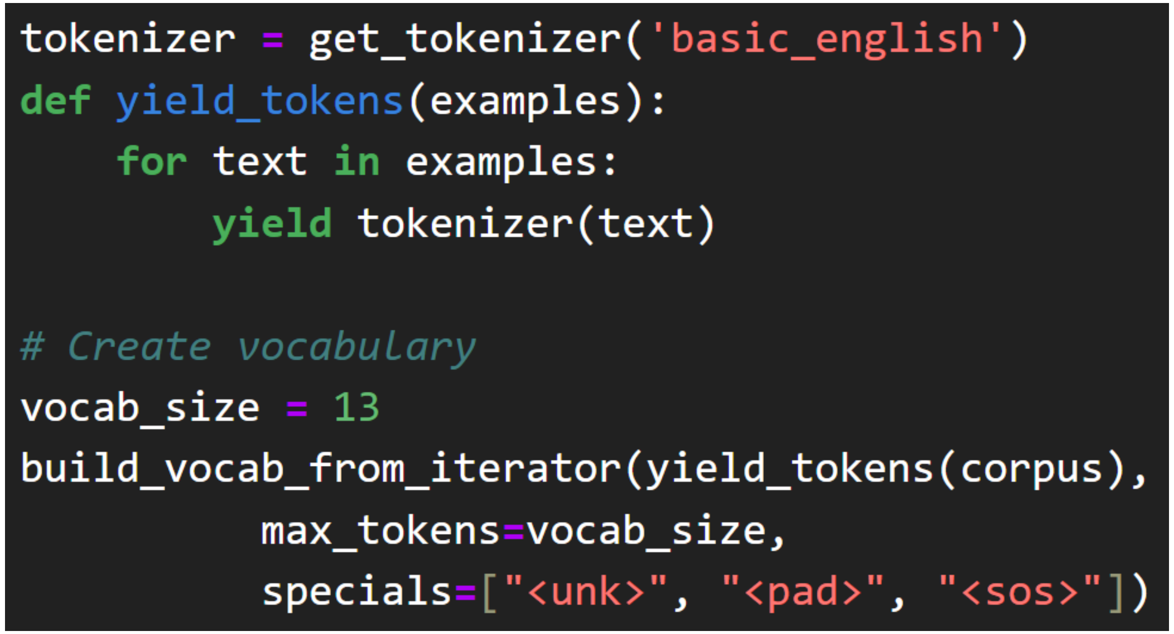


Example

Copus

Id	Text
0	Ăn quả nhớ kẻ trồng cây
1	Có chí thì nên





```
tokenizer = get_tokenizer('basic_english')
def yield_tokens(examples):
    for text in examples:
        yield tokenizer(text)

# Create vocabulary
vocab_size = 13
build_vocab_from_iterator(yield_tokens(corpus),
          max_tokens=vocab_size,
          specials=["<unk>", "<pad>", "<sos>"])
```

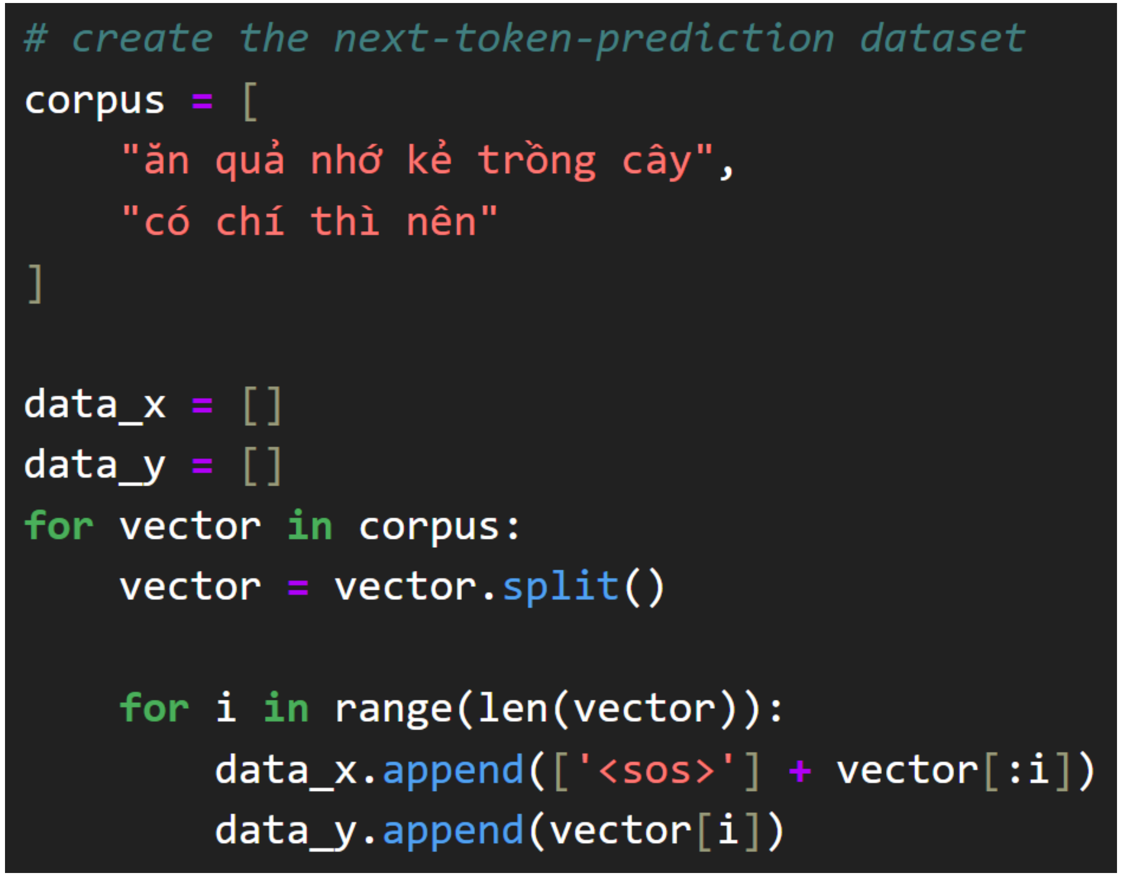
vocab_size = 13

Token	Id
<unk>	0
<pad>	1
<sos>	2
chí	3
cây	4
có	5
kẻ	6
nhớ	7
nên	8
quả	9
thì	10
trồng	11
ăn	12

Id	Text
0	Ăn quả nhớ kẻ trồng cây
1	Có chí thì nên

Input tokens	Target token
<sos>	Ăn
<sos> Ăn	quả
<sos> Ăn quả	nhớ
<sos> Ăn quả nhớ	kẻ
<sos> Ăn quả nhớ kẻ	trồng
<sos> Ăn quả nhớ kẻ trồng	cây
<sos>	Có
<sos> Có	chí
<sos> Có chí	thì
<sos> Có chí thì	nên

Next token prediction dataset



```
# create the next-token-prediction dataset
corpus = [
    "ăn quả nhớ kẻ trồng cây",
    "có chí thì nên"
]

data_x = []
data_y = []
for vector in corpus:
    vector = vector.split()

    for i in range(len(vector)):
        data_x.append(['<sos>'] + vector[:i])
        data_y.append(vector[i])
```

Token	Id
<unk>	0
<pad>	1
<sos>	2
chí	3
cây	4
có	5
kẻ	6
nhớ	7
nên	8
quả	9
thì	10
trồng	11
ăn	12

vocab_size = 13

Example

Id	Text
0	Ăn quả nhớ kẻ trồng cây
1	Có chí thì nên

Input tokens	Target token
<sos>	Ăn
<sos> Ăn	quả
<sos> Ăn quả	nhớ
<sos> Ăn quả nhớ	kẻ
<sos> Ăn quả nhớ kẻ	trồng
<sos> Ăn quả nhớ kẻ trồng	cây
<sos>	Có
<sos> Có	chí
<sos> Có chí	thì
<sos> Có chí thì	nên

Next token prediction dataset

Token	Id	kẻ	6
<unk>	0	nhớ	7
<pad>	1	nên	8
<sos>	2	quả	9
chí	3	thì	10
cây	4	trồng	11
có	5	ăn	12

Vocab

sequence_length = 6

padding

data_x_ids	data_y_ids
[2, 1, 1, 1, 1, 1]	12
[2, 12, 1, 1, 1, 1]	9
[2, 12, 9, 1, 1, 1]	7
[2, 12, 9, 7, 1, 1]	6
[2, 12, 9, 7, 6, 1]	11
[2, 12, 9, 7, 6, 11]	4
[2, 1, 1, 1, 1, 1]	5
[2, 5, 1, 1, 1, 1]	3
[2, 5, 3, 1, 1, 1]	10
[2, 5, 3, 10, 1, 1]	8

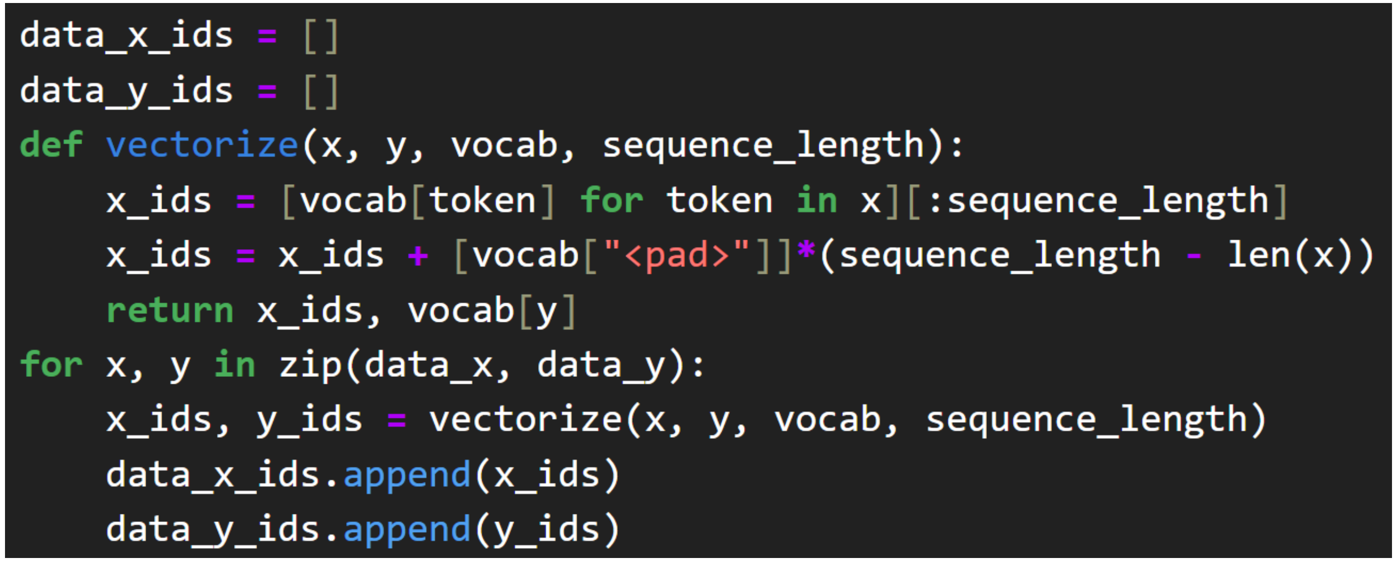
Training data

Example

Id	Text
0	Ăn quả nhớ kẻ trồng cây
1	Có chí thì nên

Input tokens	Target token
<sos>	Ăn
<sos> Ăn	quả
<sos> Ăn quả	nhớ
<sos> Ăn quả nhớ	kẻ
<sos> Ăn quả nhớ kẻ	trồng
<sos> Ăn quả nhớ kẻ trồng	cây
<sos>	Có
<sos> Có	chí
<sos> Có chí	thì
<sos> Có chí thì	nên

Next token prediction dataset



```
data_x_ids = []
data_y_ids = []
def vectorize(x, y, vocab, sequence_length):
    x_ids = [vocab[token] for token in x][:sequence_length]
    x_ids = x_ids + [vocab["<pad>"]]*(sequence_length - len(x))
    return x_ids, vocab[y]
for x, y in zip(data_x, data_y):
    x_ids, y_ids = vectorize(x, y, vocab, sequence_length)
    data_x_ids.append(x_ids)
    data_y_ids.append(y_ids)
```

Training data

sequence_length = 6

Vocab

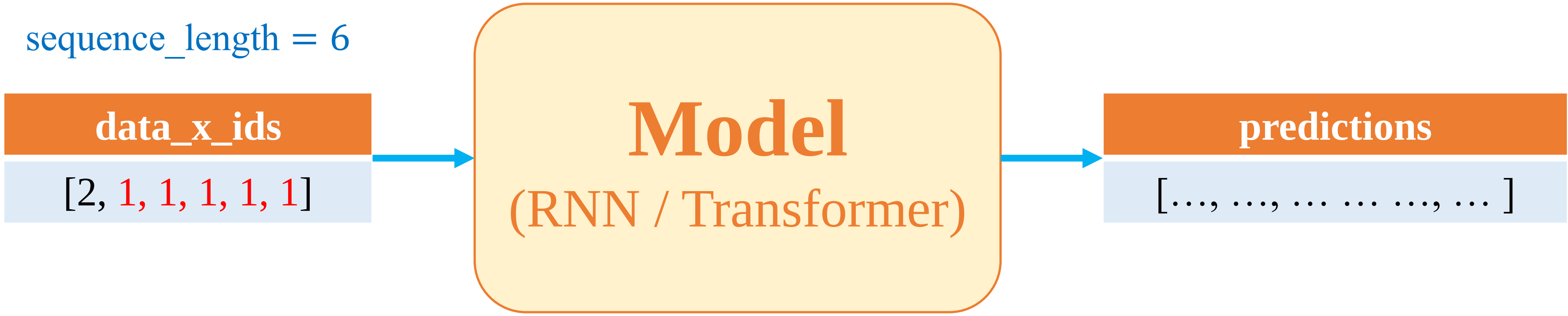
padding

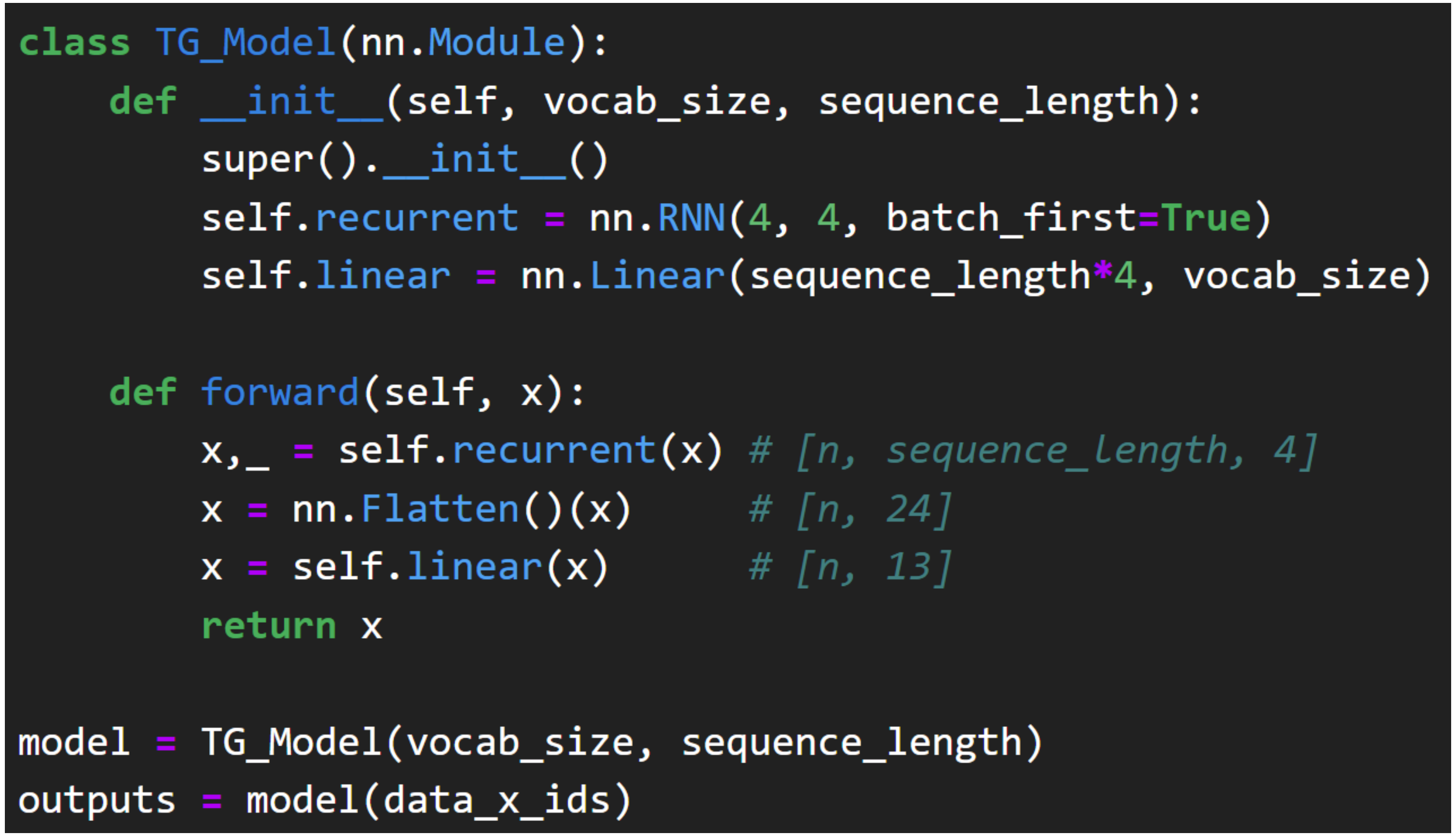
data_x_ids	data_y_ids
[2, 1, 1, 1, 1, 1]	12
[2, 12, 1, 1, 1, 1]	9
[2, 12, 9, 1, 1, 1]	7
[2, 12, 9, 7, 1, 1]	6
[2, 12, 9, 7, 6, 1]	11
[2, 12, 9, 7, 6, 11]	4
[2, 1, 1, 1, 1, 1]	5
[2, 5, 1, 1, 1, 1]	3
[2, 5, 3, 1, 1, 1]	10
[2, 5, 3, 10, 1, 1]	8

Example

Token	Id
<unk>	0
<pad>	1
<sos>	2
chí	3
cây	4
có	5
kẻ	6
nhớ	7
nên	8
quả	9
thì	10
trồng	11
ăn	12

vocab_size = 13





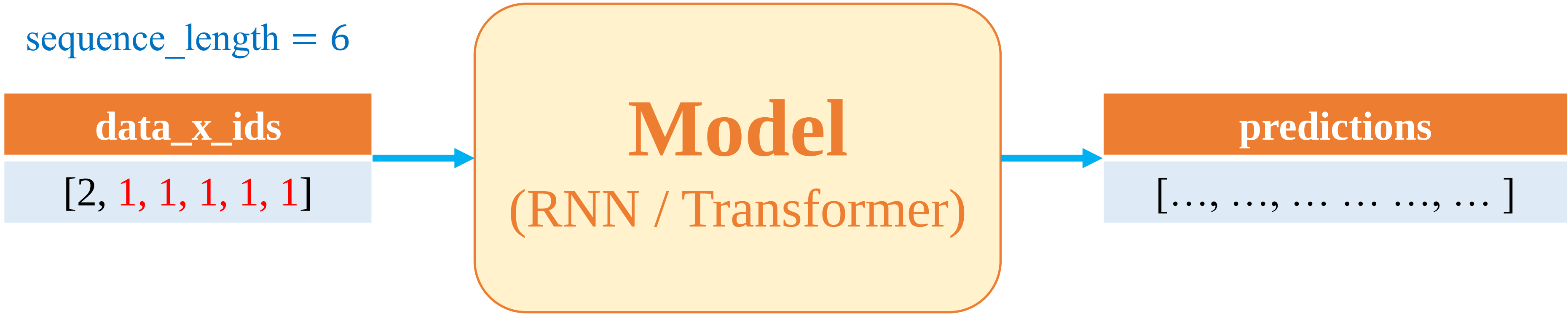
```
class TG_Model(nn.Module):
    def __init__(self, vocab_size, sequence_length):
        super().__init__()
        self.recurrent = nn.RNN(4, 4, batch_first=True)
        self.linear = nn.Linear(sequence_length*4, vocab_size)

    def forward(self, x):
        x,_ = self.recurrent(x) # [n, sequence_length, 4]
        x = nn.Flatten()(x)     # [n, 24]
        x = self.linear(x)      # [n, 13]
        return x

model = TG_Model(vocab_size, sequence_length)
outputs = model(data_x_ids)
```

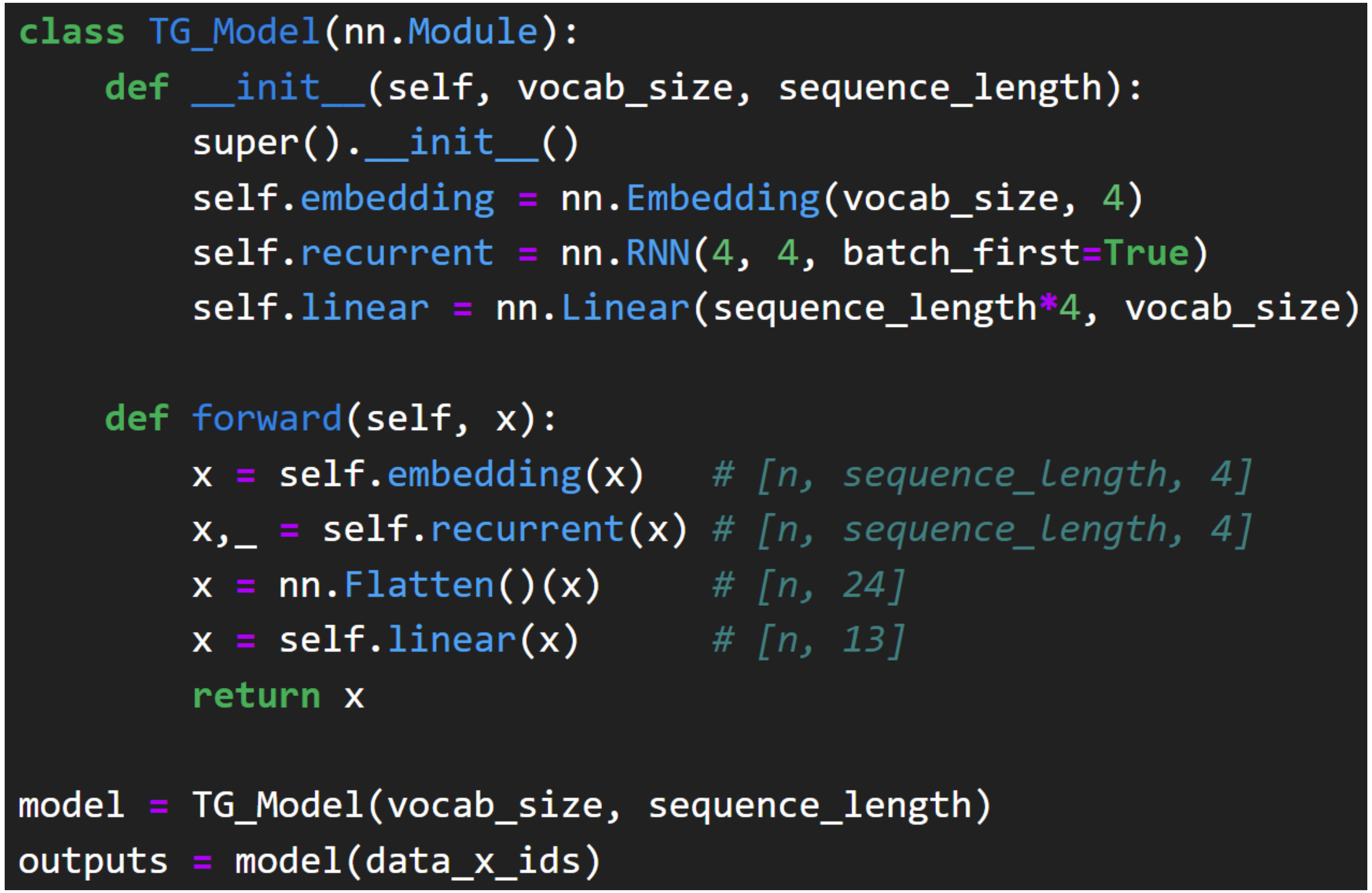
?

Example

Token	Id
<unk>	0
<pad>	1
<sos>	2
chí	3
cây	4
có	5
kẻ	6
nhớ	7
nên	8
quả	9
thì	10
trồng	11
ăn	12

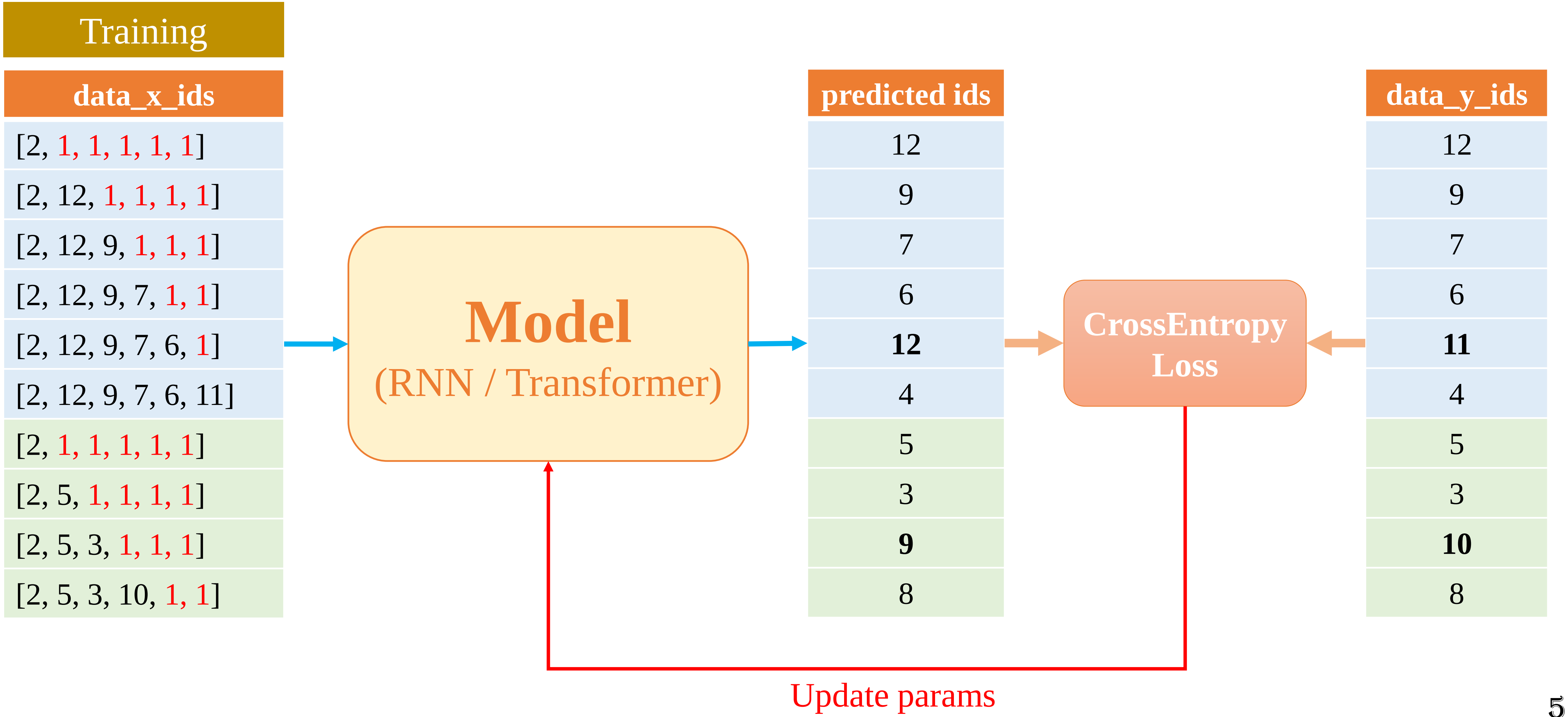
vocab_size = 13





```
class TG_Model(nn.Module):
    def __init__(self, vocab_size, sequence_length):
        super().__init__()
        self.embedding = nn.Embedding(vocab_size, 4)
        self.recurrent = nn.RNN(4, 4, batch_first=True)
        self.linear = nn.Linear(sequence_length*4, vocab_size)

    def forward(self, x):
        x = self.embedding(x)   # [n, sequence_length, 4]
        x,_ = self.recurrent(x) # [n, sequence_length, 4]
        x = nn.Flatten()(x)     # [n, 24]
        x = self.linear(x)      # [n, 13]
        return x

model = TG_Model(vocab_size, sequence_length)
outputs = model(data_x_ids)
```

Example



Training

data_x_ids	predicted ids	data_y_ids
[2, 1, 1, 1, 1, 1]	12	12
[2, 12, 1, 1, 1, 1]	9	9
[2, 12, 9, 1, 1, 1]	7	7
[2, 12, 9, 7, 1, 1]	6	6
[2, 12, 9, 7, 6, 1]	**12**	**11**
[2, 12, 9, 7, 6, 11]	4	4
[2, 1, 1, 1, 1, 1]	5	5
[2, 5, 1, 1, 1, 1]	3	3
[2, 5, 3, 1, 1, 1]	**9**	**10**
[2, 5, 3, 10, 1, 1]	8	8

Feed Forward

$$\text{sequence\_length} = 4$$
$$\text{hidden\_dim} = 4$$

Input tokens	Target token
<sos> Có chí thì	nên

Input ids	Target ids
[2, 5, 3, 10]	[8]

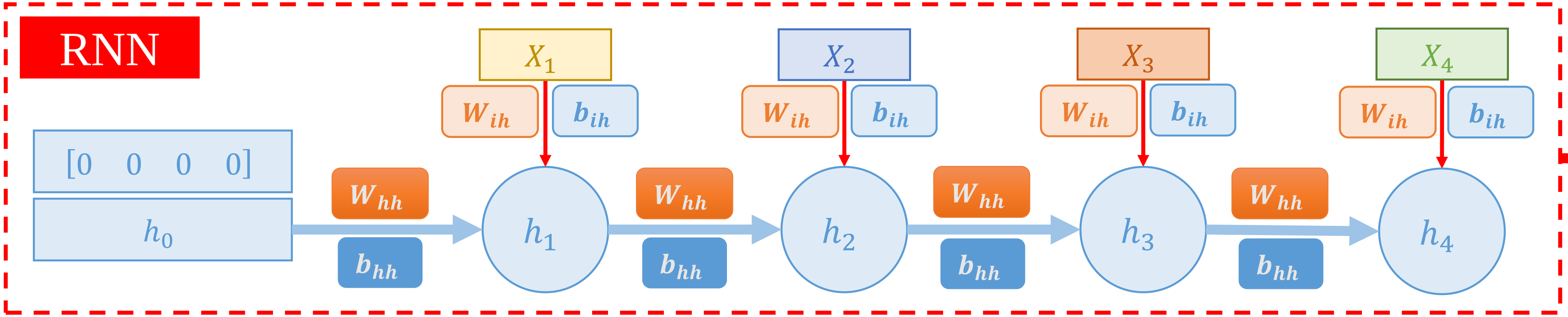
$[2 \quad 5 \quad 3 \quad 10]$ → Embedding layer →

$X_1 = [-0.7521 \quad 1.6487 \quad -0.3925 \quad -1.4036]$

$X_2 = [-0.7581 \quad 1.0783 \quad 0.8008 \quad 1.6806]$

$X_3 = [-0.7279 \quad -0.5594 \quad -0.7688 \quad 0.7624]$

$X_4 = [-0.8371 \quad -0.9224 \quad 1.8113 \quad 0.1606]$



$$\boldsymbol{W_{hh}}$$
$$\begin{bmatrix} 0.3035 & -0.1187 & 0.2860 & -0.3885 \\ -0.2523 & 0.1524 & 0.1057 & -0.1275 \\ 0.2980 & 0.3399 & -0.3626 & -0.2669 \\ 0.4578 & -0.1687 & -0.1773 & -0.4838 \end{bmatrix}$$

$$\boldsymbol{W_{ih}}$$
$$\begin{bmatrix} -0.2982 & -0.2982 & 0.4497 & 0.1666 \\ 0.4811 & -0.4126 & -0.4959 & -0.3912 \\ -0.3363 & 0.2025 & 0.1790 & 0.4155 \\ -0.2582 & -0.3409 & 0.2653 & -0.2021 \end{bmatrix}$$

$$b_{hh}$$
$$[0.0117 \quad -0.3415 \quad -0.4242 \quad -0.2753\ ]$$

$$b_{ih}$$
$$[-0.2863 \quad 0.1249 \quad -0.0660 \quad -0.3629]$$

$h_1 = [-0.7409 \quad -0.4739 \quad -0.5055 \quad -0.6786]$

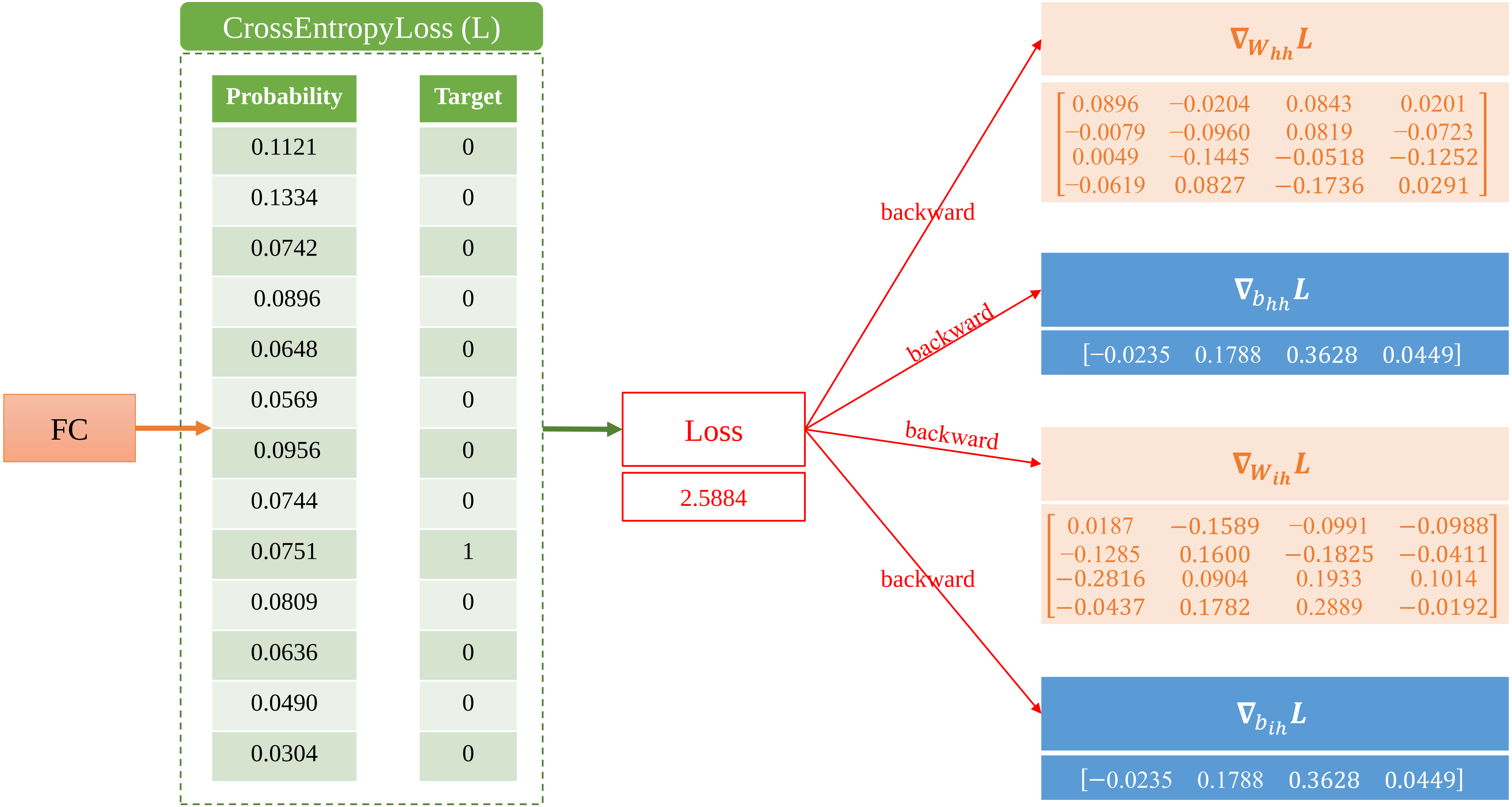
$h_2 = [0.2170 \quad -0.9590 \quad 0.6681 \quad -0.6519]$

$h_3 = [0.4735 \quad -0.2915 \quad -0.4692 \quad -0.1583]$

$h_4 = [0.8327 \quad -0.8839 \quad 0.2449 \quad 0.6446]$

→ Flatten → Linear (16, 13)

Back-Propagation



CrossEntropyLoss (L)

Probability	Target
0.1121	0
0.1334	0
0.0742	0
0.0896	0
0.0648	0
0.0569	0
0.0956	0
0.0744	0
0.0751	1
0.0809	0
0.0636	0
0.0490	0
0.0304	0

Loss

2.5884

$\nabla_{W_{hh}} L$

$$\begin{bmatrix} 0.0896 & -0.0204 & 0.0843 & 0.0201 \\ -0.0079 & -0.0960 & 0.0819 & -0.0723 \\ 0.0049 & -0.1445 & -0.0518 & -0.1252 \\ -0.0619 & 0.0827 & -0.1736 & 0.0291 \end{bmatrix}$$

$\nabla_{b_{hh}} L$

$[-0.0235 \quad 0.1788 \quad 0.3628 \quad 0.0449]$

$\nabla_{W_{ih}} L$

$$\begin{bmatrix} 0.0187 & -0.1589 & -0.0991 & -0.0988 \\ -0.1285 & 0.1600 & -0.1825 & -0.0411 \\ -0.2816 & 0.0904 & 0.1933 & 0.1014 \\ -0.0437 & 0.1782 & 0.2889 & -0.0192 \end{bmatrix}$$

$\nabla_{b_{ih}} L$

$[-0.0235 \quad 0.1788 \quad 0.3628 \quad 0.0449]$

Example

data_x_ids

[2, 1, 1, 1, 1, 1]

sequence_length = 6

Model

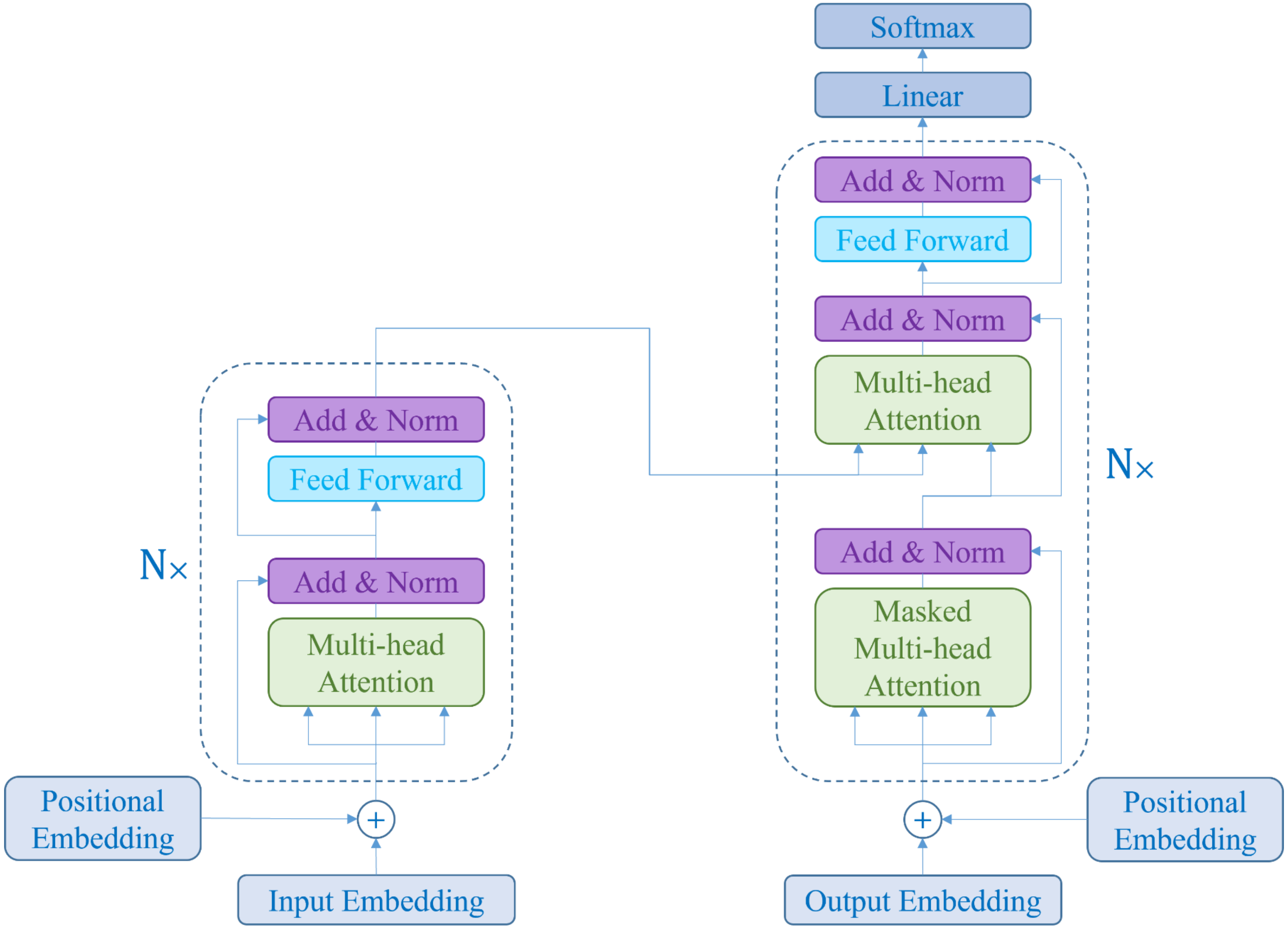
(Transformer)

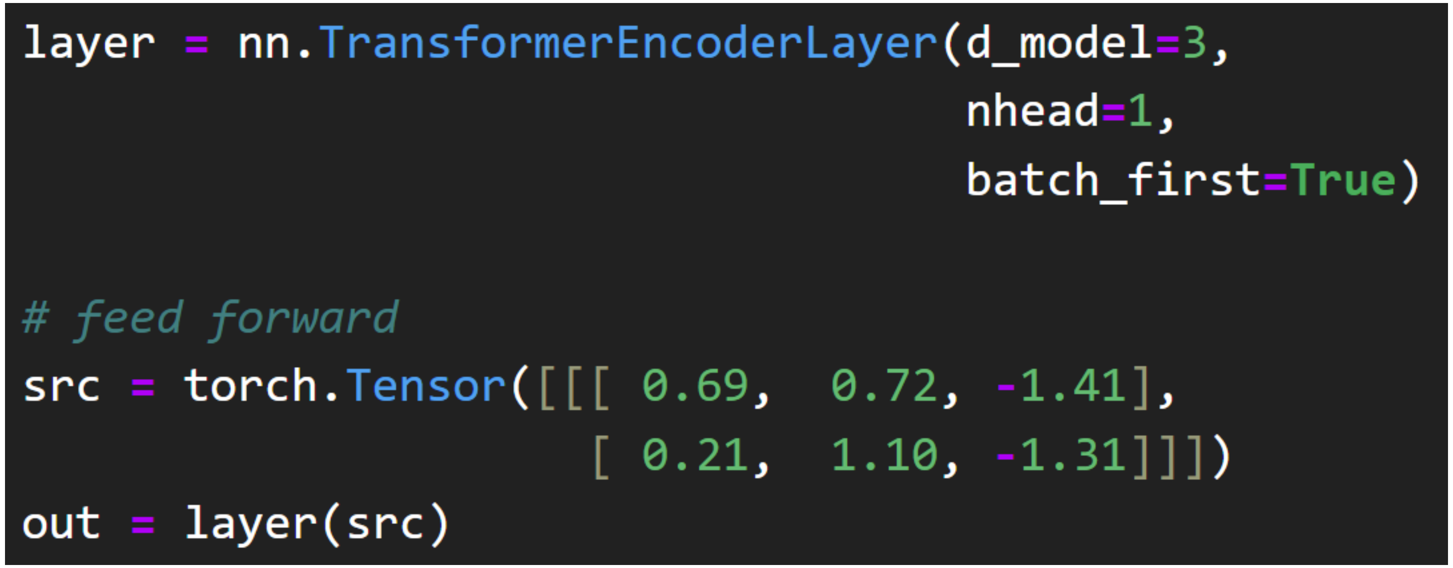
predictions

[…, …, … … …, …]

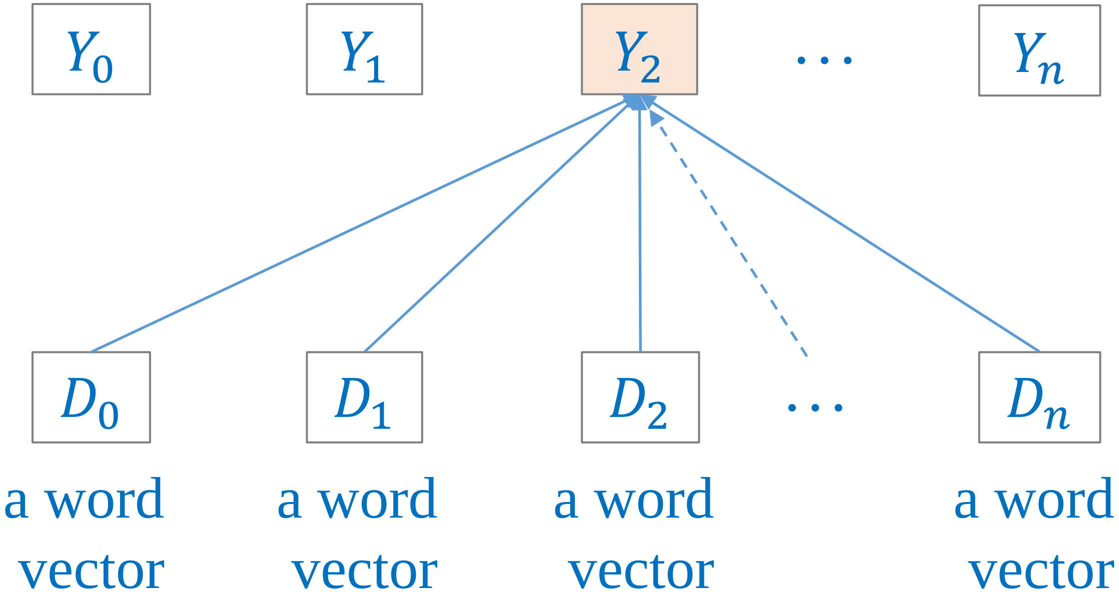
Token	Id
<unk>	0
<pad>	1
<sos>	2
chí	3
cây	4
có	5
kẻ	6
nhớ	7
nên	8
quả	9
thì	10
trồng	11
ăn	12

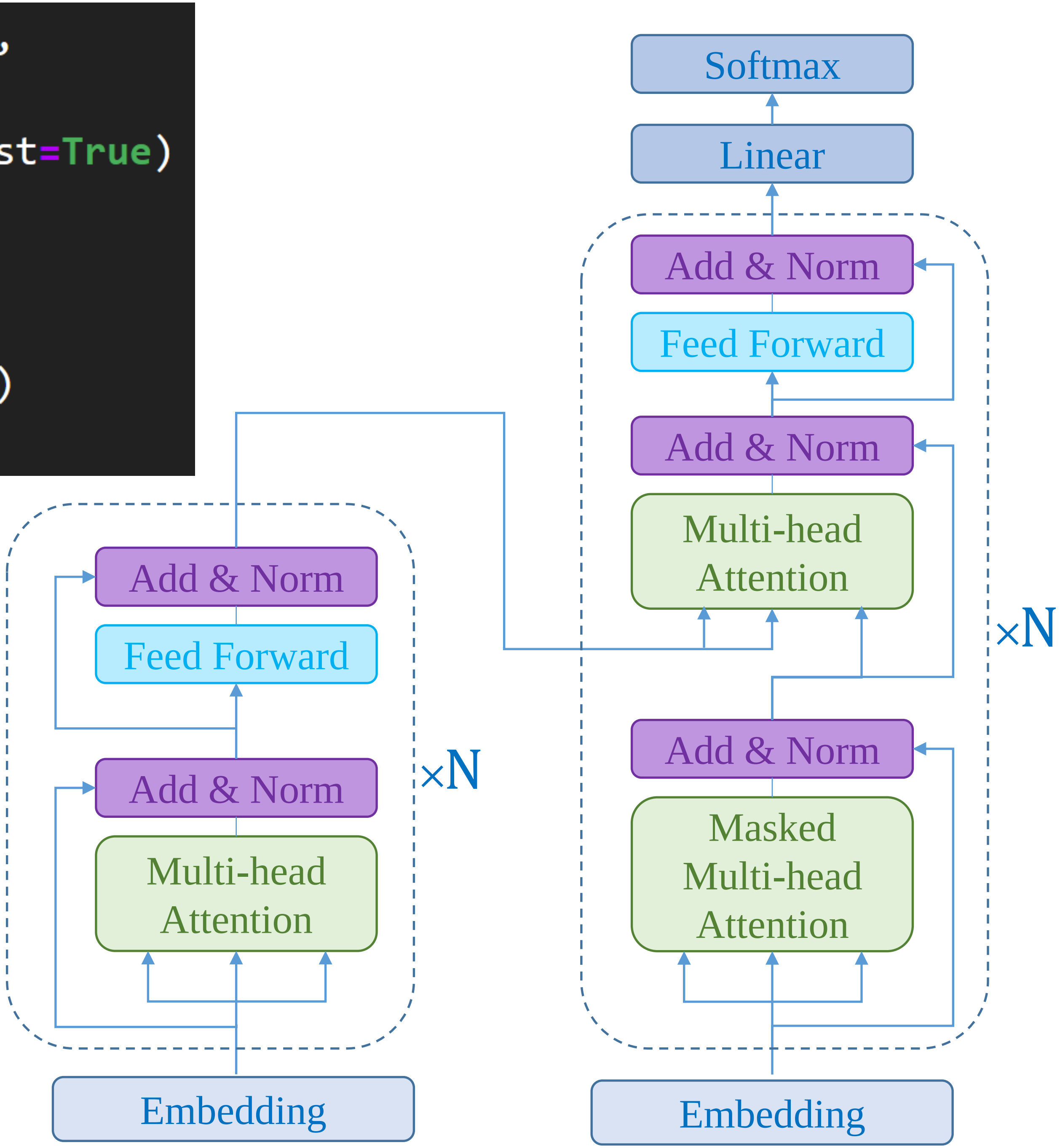
vocab_size = 13





```
layer = nn.TransformerEncoderLayer(d_model=3,
                                   nhead=1,
                                   batch_first=True)


# feed forward
src = torch.Tensor([[[ 0.69,  0.72, -1.41],
                     [ 0.21,  1.10, -1.31]]])
out = layer(src)
```

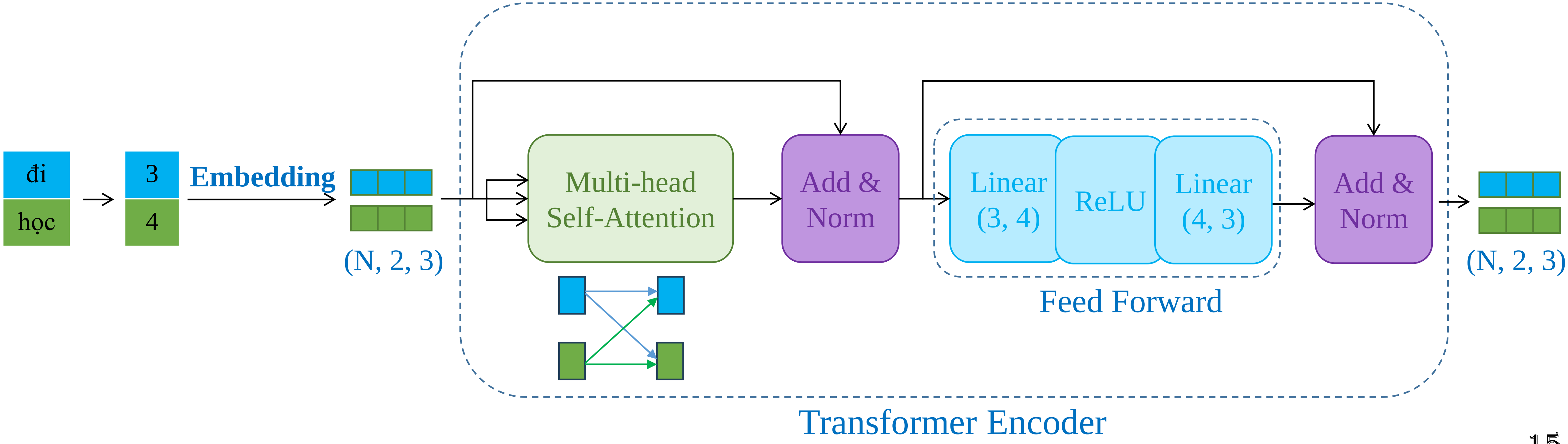




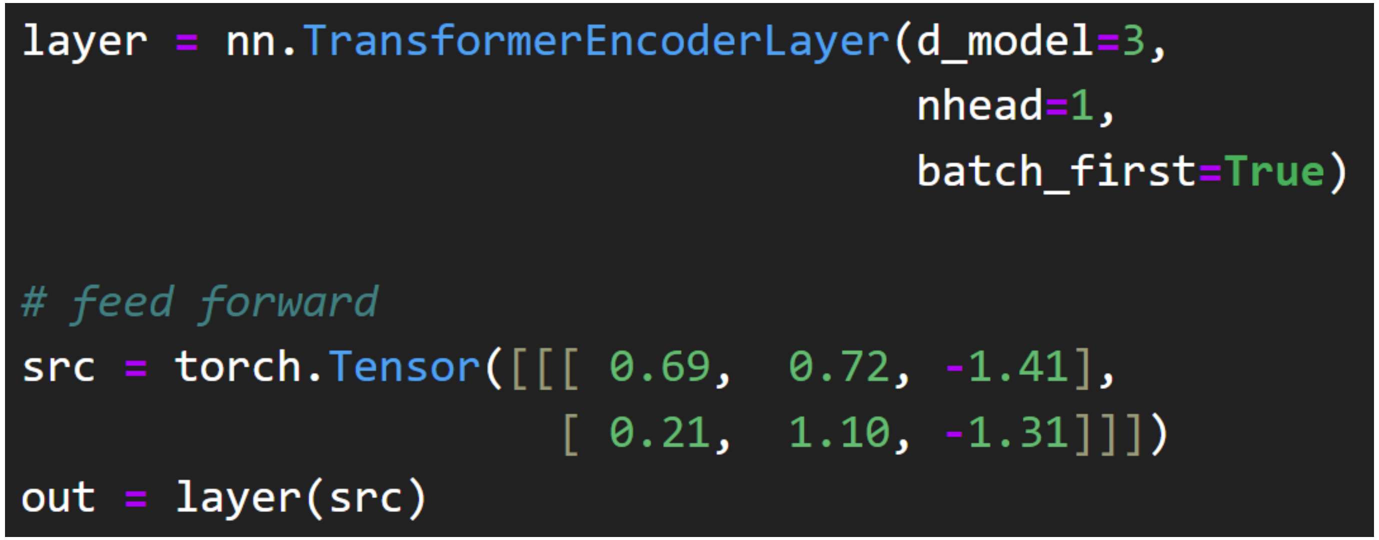
Encoder in Pytorch

index	word
0	[UNK]
1	[pad]
2	ai
3	đi
4	học
…	…

index	Embedding
0	[-0.188, …, 0.7013]
1	[1.7840…, 1.3586]
2	[1.0281, …, 0.4211]
3	[-1.308, …, -0.3680]
4	[0.2293, …, 2.0501]
…	…



Encoder in Pytorch



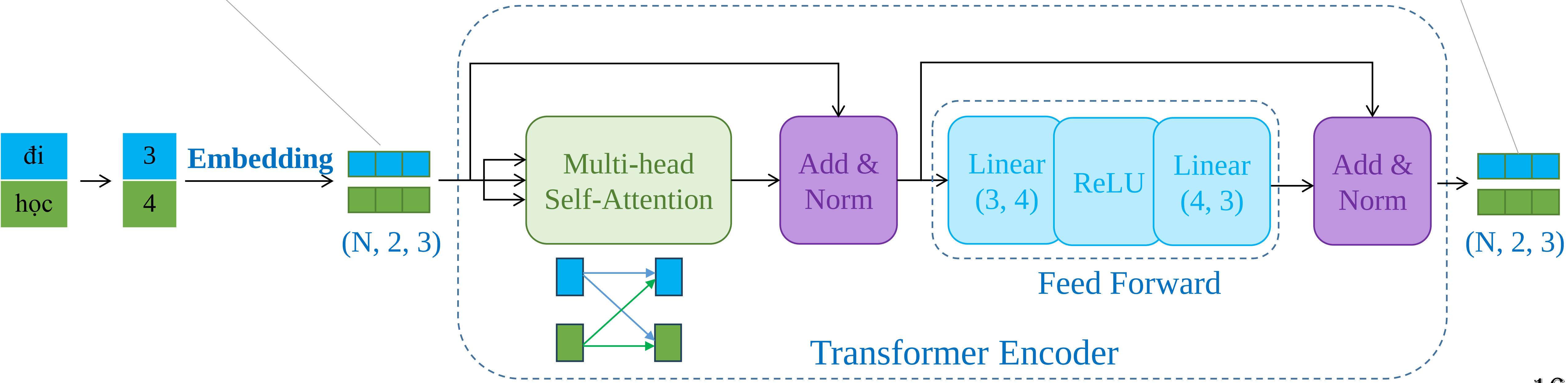
```
layer = nn.TransformerEncoderLayer(d_model=3,
                                   nhead=1,
                                   batch_first=True)


# feed forward
src = torch.Tensor([[[ 0.69,  0.72, -1.41],
                     [ 0.21,  1.10, -1.31]]])
out = layer(src)
```

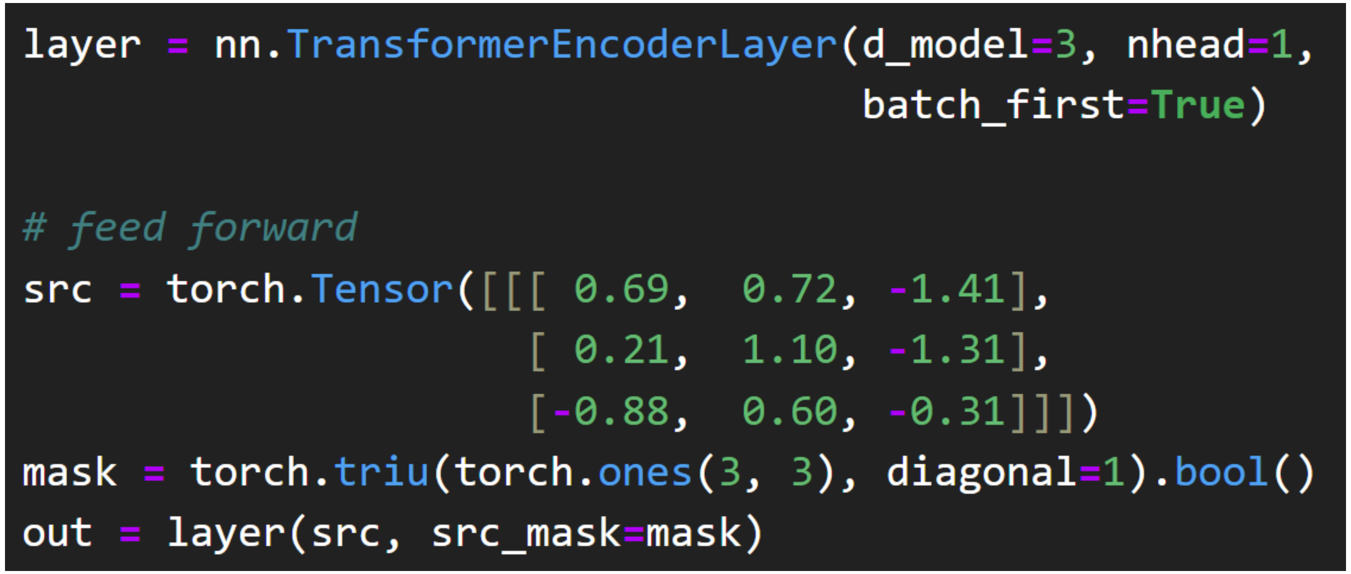
[0.69, 0.72, -1.41]

[0.21, 1.10, -1.31]

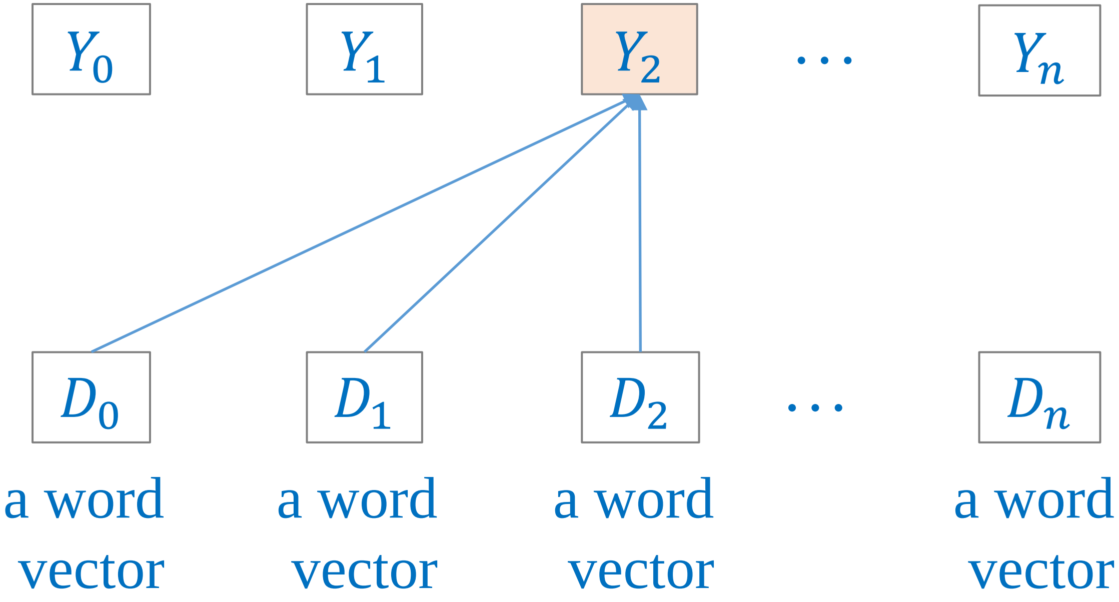
[0.97, 0.39, -1.37]

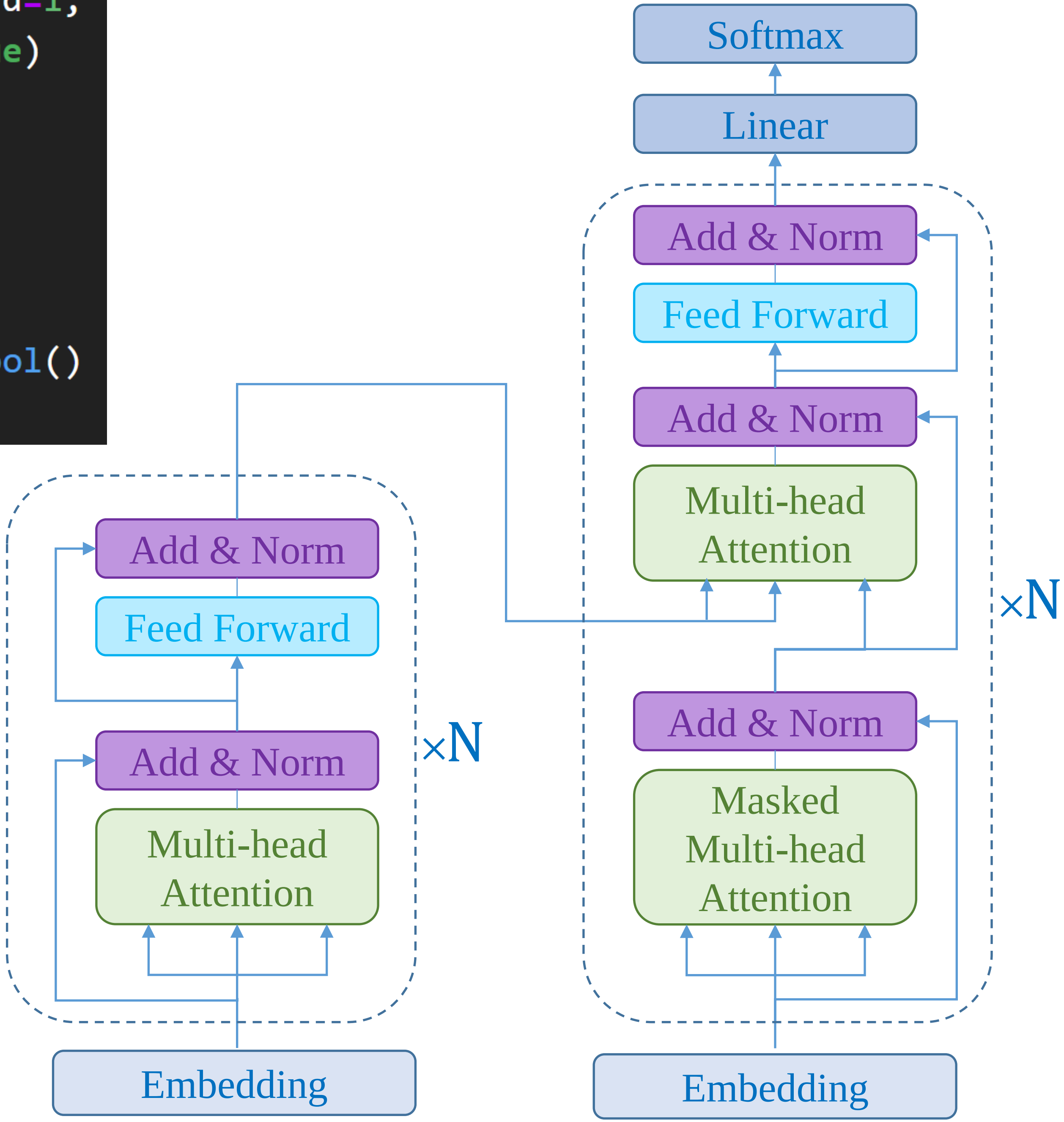
[0.58, 0.82, -1.40]





```
layer = nn.TransformerEncoderLayer(d_model=3, nhead=1,
                                   batch_first=True)

# feed forward
src = torch.Tensor([[[ 0.69,  0.72, -1.41],
                     [ 0.21,  1.10, -1.31],
                     [-0.88,  0.60, -0.31]]])
mask = torch.triu(torch.ones(3, 3), diagonal=1).bool()
out = layer(src, src_mask=mask)
```

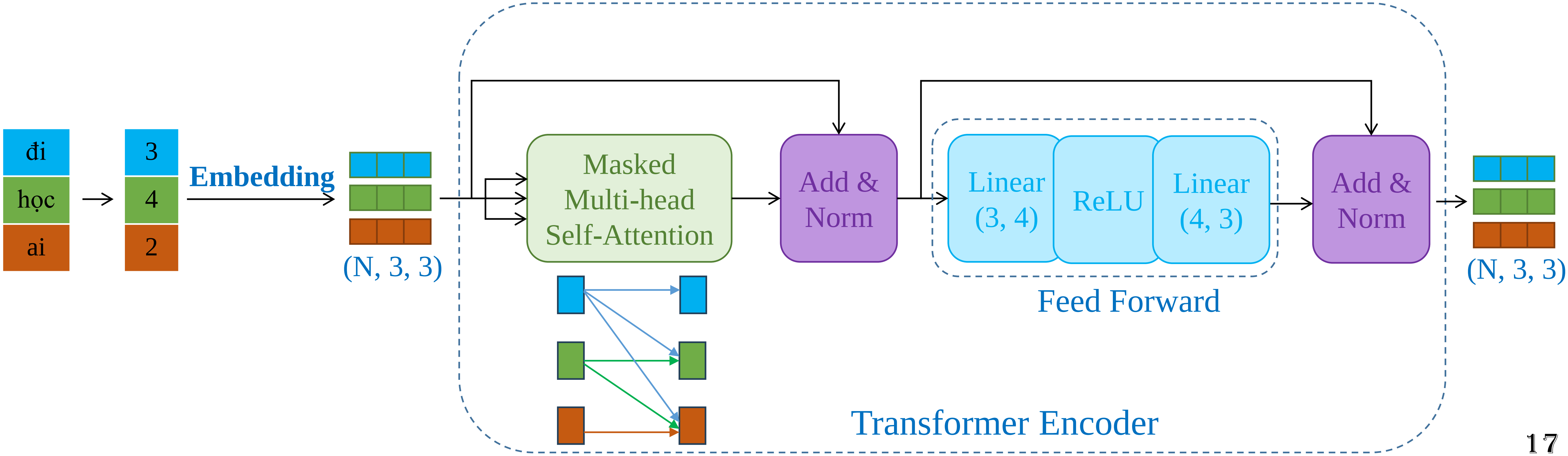




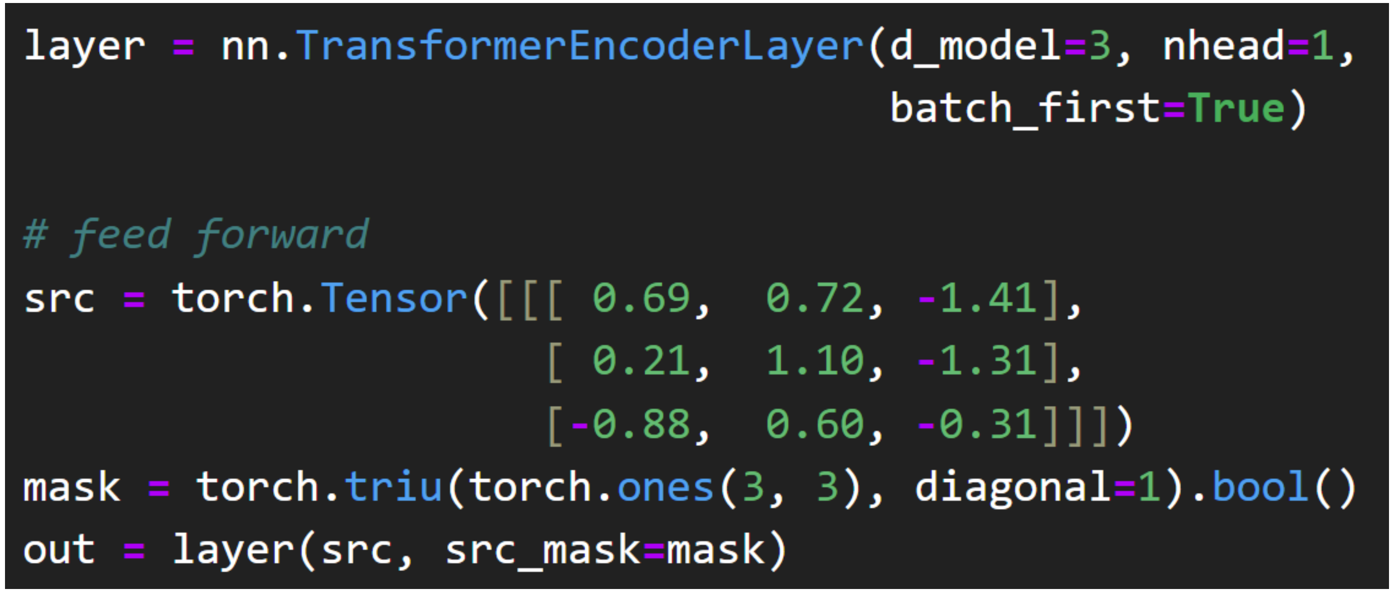
Masked Encoder in Pytorch

index	word
0	[UNK]
1	[pad]
2	ai
3	đi
4	học
…	…

index	Embedding
0	[-0.188, …, 0.7013]
1	[1.7840…, 1.3586]
2	[1.0281, …, 0.4211]
3	[-1.308, …, -0.3680]
4	[0.2293, …, 2.0501]
…	…



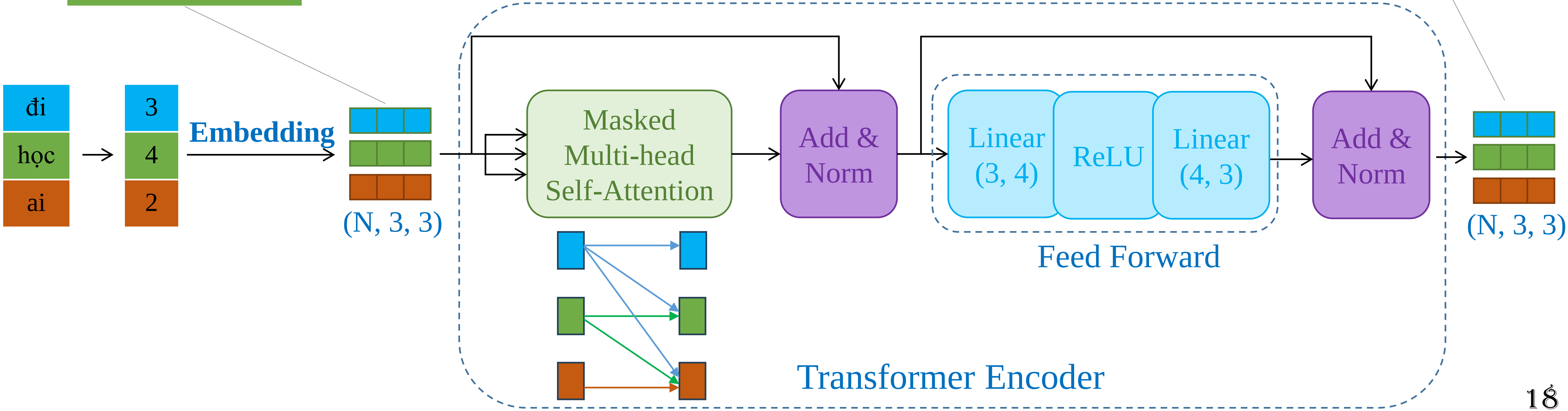
Masked Encoder in Pytorch



```
layer = nn.TransformerEncoderLayer(d_model=3, nhead=1,
                                   batch_first=True)

# feed forward
src = torch.Tensor([[[ 0.69,  0.72, -1.41],
                     [ 0.21,  1.10, -1.31],
                     [-0.88,  0.60, -0.31]]])
mask = torch.triu(torch.ones(3, 3), diagonal=1).bool()
out = layer(src, src_mask=mask)
```

[0.69, 0.72, -1.41]
[0.21, 1.10, -1.31]
[-0.88, 0.60, -0.31]

[0.97, 0.39, -1.37]
[0.58, 0.82, -1.40]
[-0.85, 1.40, -0.54]



Masked Multi-head Attention

head = 1

$$W_Q = \begin{bmatrix} -0.35 & 0.51 & 0.50 \\ 0.36 & -0.47 & -0.29 \\ -0.51 & -0.14 & -0.56 \end{bmatrix}$$

$$W_K = \begin{bmatrix} -0.49 & -0.68 & 0.18 \\ -0.44 & -0.46 & 0.18 \\ 0.07 & -0.10 & 0.44 \end{bmatrix}$$

$$W_V = \begin{bmatrix} -0.41 & 0.39 & -0.65 \\ -0.40 & -0.07 & -0.34 \\ -0.55 & -0.13 & -0.29 \end{bmatrix}$$

$$W_O = \begin{bmatrix} -0.36 & -0.08 & 0.32 \\ 0.27 & 0.05 & 0.15 \\ -0.05 & -0.28 & 0.05 \end{bmatrix}$$

$$X = \begin{bmatrix} -0.1 & 0.1 & 0.3 \\ 0.4 & -1.1 & -0.3 \end{bmatrix}$$

$$Q = XW_Q = \begin{bmatrix} -0.1 & 0.1 & 0.3 \\ 0.4 & -1.1 & -0.3 \end{bmatrix} \begin{bmatrix} -0.35 & 0.51 & 0.50 \\ 0.36 & -0.47 & -0.29 \\ -0.51 & -0.14 & -0.56 \end{bmatrix}$$

$$= \begin{bmatrix} -0.08 & -0.14 & -0.24 \\ -0.39 & 0.77 & 0.69 \end{bmatrix}$$

$$K = XW_K = \begin{bmatrix} -0.1 & 0.1 & 0.3 \\ 0.4 & -1.1 & -0.3 \end{bmatrix} \begin{bmatrix} -0.49 & -0.68 & 0.18 \\ -0.44 & -0.46 & 0.18 \\ 0.07 & -0.10 & 0.44 \end{bmatrix}$$

$$= \begin{bmatrix} 0.02 & -0.01 & 0.13 \\ 0.27 & 0.27 & -0.26 \end{bmatrix}$$

$$V = XW_V = \begin{bmatrix} -0.1 & 0.1 & 0.3 \\ 0.4 & -1.1 & -0.3 \end{bmatrix} \begin{bmatrix} -0.41 & 0.39 & -0.65 \\ -0.40 & -0.07 & -0.34 \\ -0.55 & -0.13 & -0.29 \end{bmatrix}$$

$$= \begin{bmatrix} -0.16 & -0.08 & -0.05 \\ -0.02 & -0.02 & 0.05 \end{bmatrix}$$

$$M = \begin{bmatrix} 0 & -\infty \\ 0 & 0 \end{bmatrix}$$

Masked Multi-head Attention

❖ **Example**

approximately

$$A = softmax\left(\frac{QK^T}{\sqrt{d}} + M\right)V$$

$$= softmax\left(\begin{bmatrix} -0.08 & -0.14 & -0.24 \\ -0.39 & 0.77 & 0.69 \end{bmatrix}\begin{bmatrix} 0.02 & 0.27 \\ -0.01 & 0.27 \\ 0.13 & -0.26 \end{bmatrix}\frac{1}{\sqrt{d}} + \begin{bmatrix} 0 & -\infty \\ 0 & 0 \end{bmatrix}\right)\begin{bmatrix} -0.16 & -0.08 & -0.05 \\ -0.02 & -0.02 & 0.05 \end{bmatrix}$$

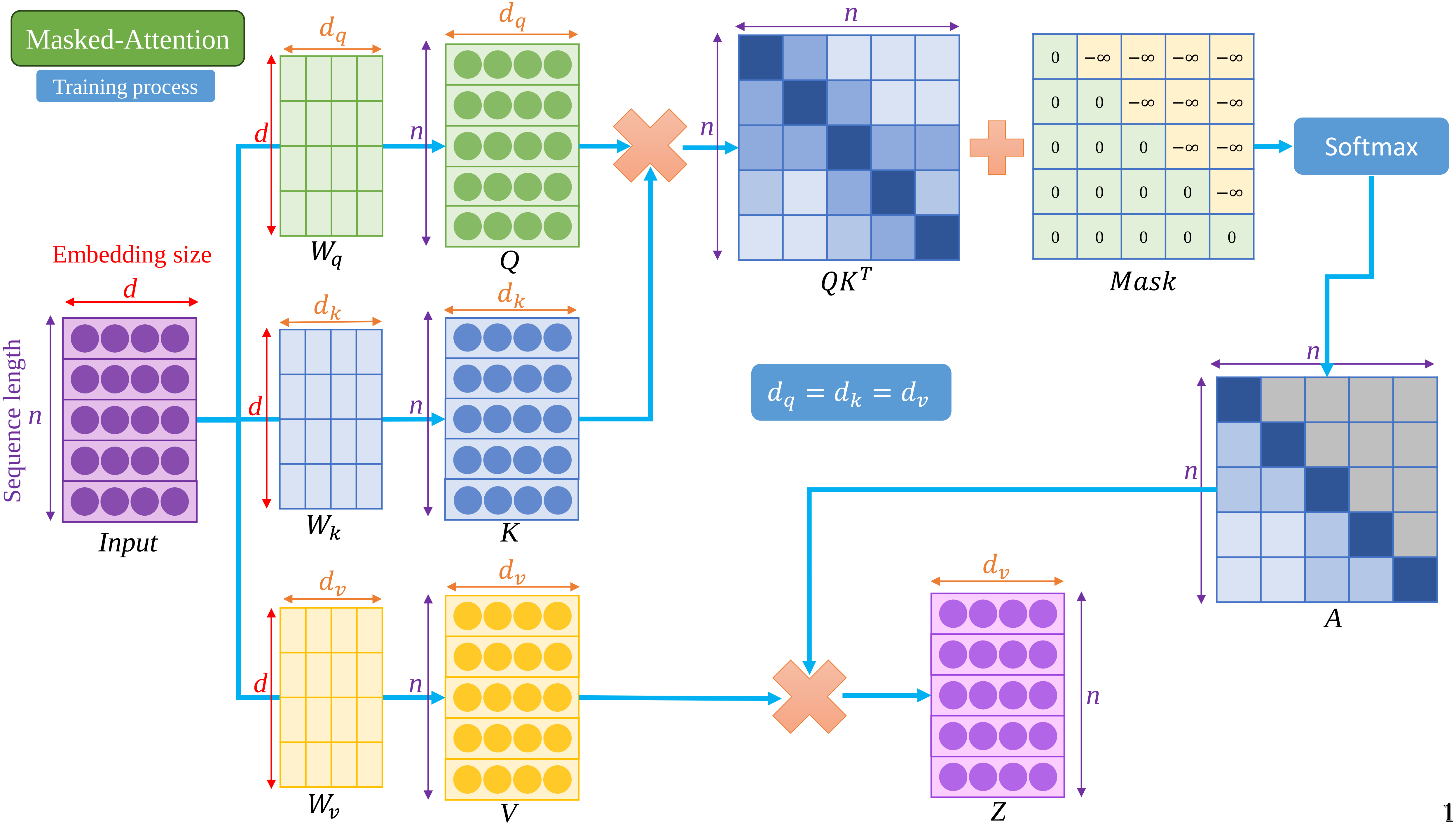
$$= softmax\left(\begin{bmatrix} -0.019 & 0.002 \\ 0.043 & -0.046 \end{bmatrix} + \begin{bmatrix} 0 & -\infty \\ 0 & 0 \end{bmatrix}\right)\begin{bmatrix} -0.16 & -0.08 & -0.05 \\ -0.02 & -0.02 & 0.05 \end{bmatrix}$$

$$= \begin{bmatrix} 1.0 & 0.0 \\ 0.52 & 0.48 \end{bmatrix}\begin{bmatrix} -0.16 & -0.08 & -0.05 \\ -0.02 & -0.02 & 0.05 \end{bmatrix} = \begin{bmatrix} -0.16 & -0.08 & -0.05 \\ 0.12 & 0.08 & 0.06 \end{bmatrix}$$

$$Y = AW_O = \begin{bmatrix} -0.16 & -0.08 & -0.05 \\ 0.12 & 0.08 & 0.06 \end{bmatrix}\begin{bmatrix} -0.36 & -0.08 & 0.32 \\ 0.27 & 0.05 & 0.15 \\ -0.05 & -0.28 & 0.05 \end{bmatrix} = \begin{bmatrix} 0.03 & 0.02 & -0.06 \\ -0.02 & -0.02 & 0.05 \end{bmatrix}$$

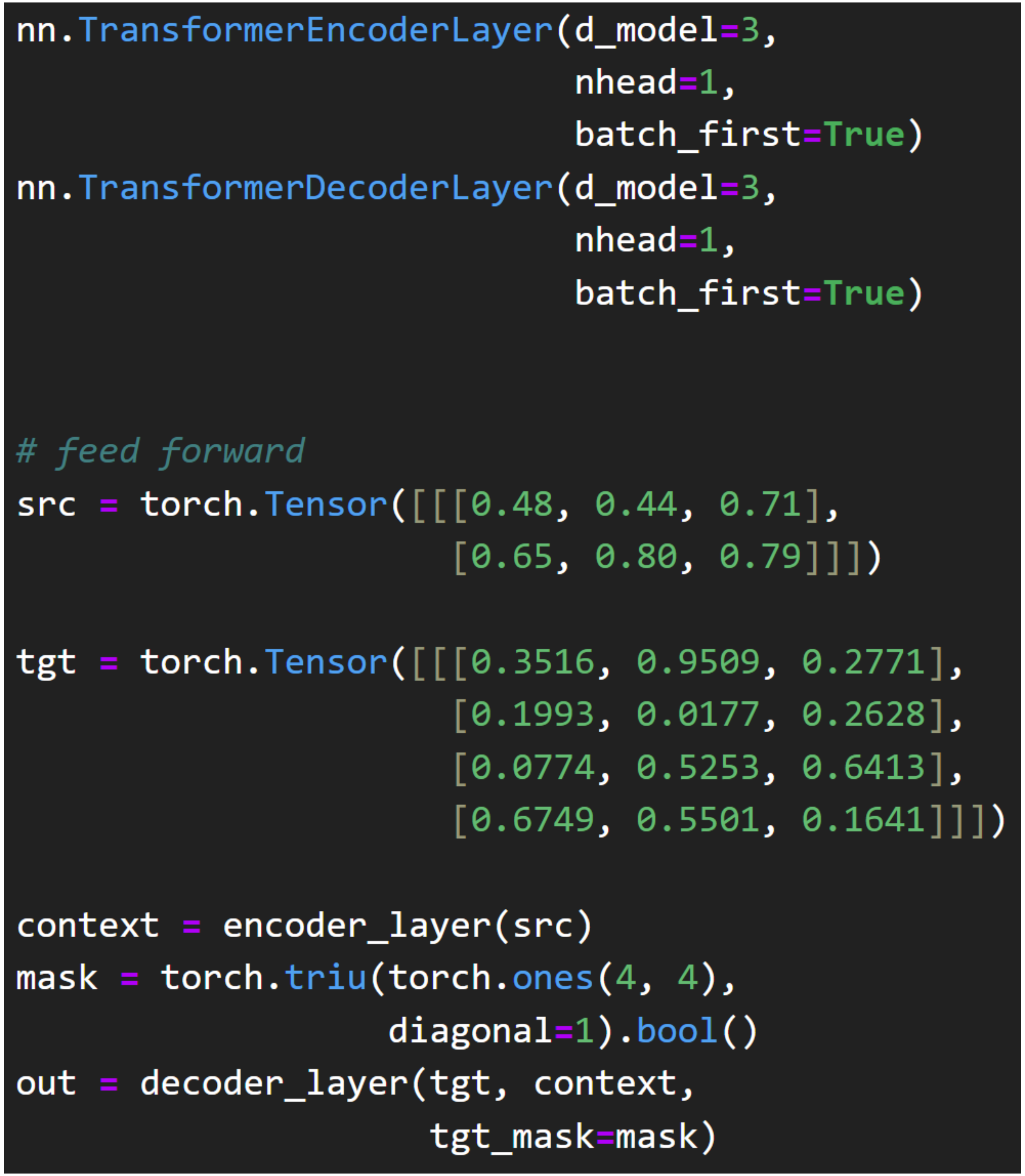
Masked-Attention

Training process

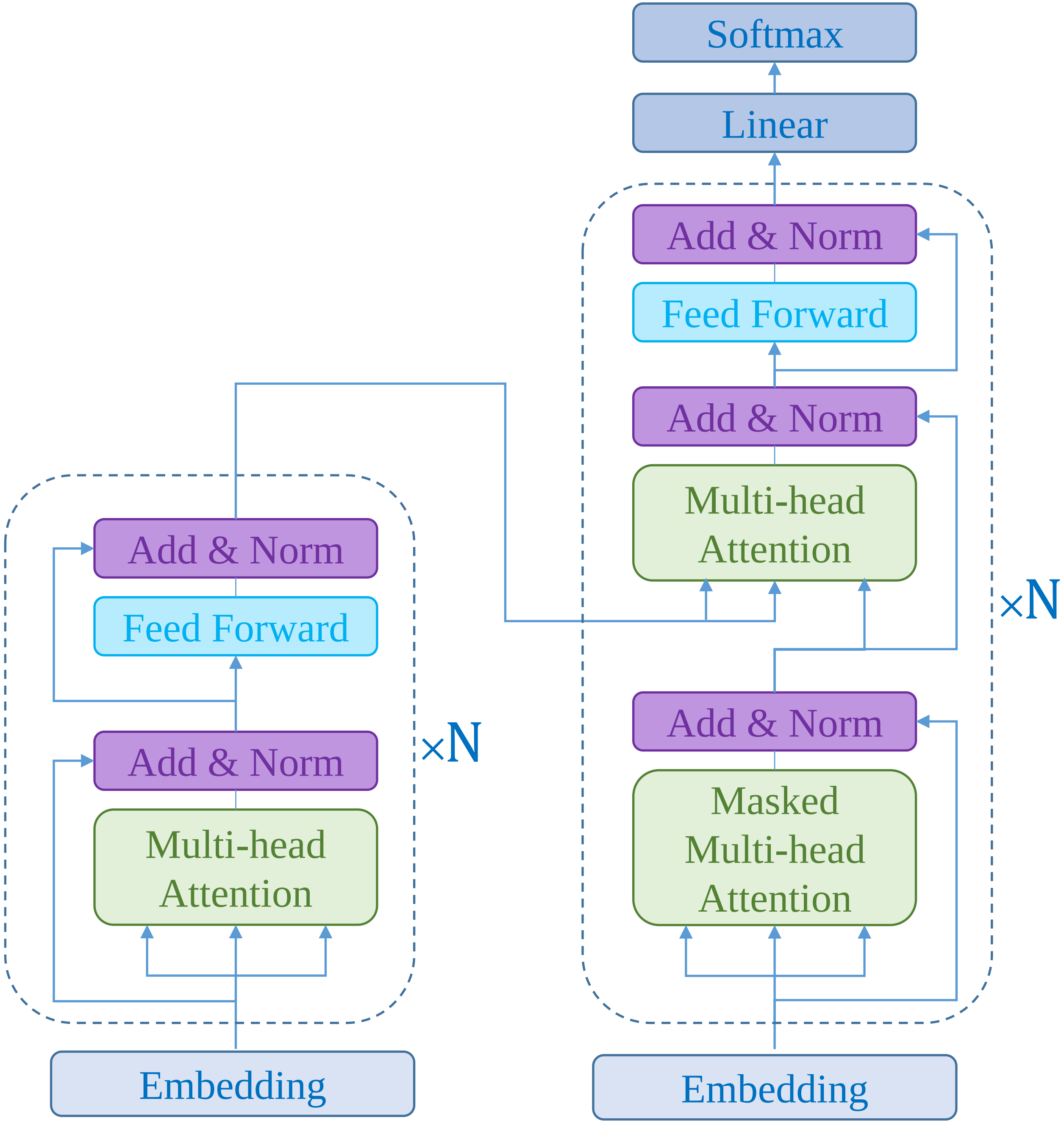


0	$-\infty$	$-\infty$	$-\infty$	$-\infty$
0	0	$-\infty$	$-\infty$	$-\infty$
0	0	0	$-\infty$	$-\infty$
0	0	0	0	$-\infty$
0	0	0	0	0

$$d_q = d_k = d_v$$

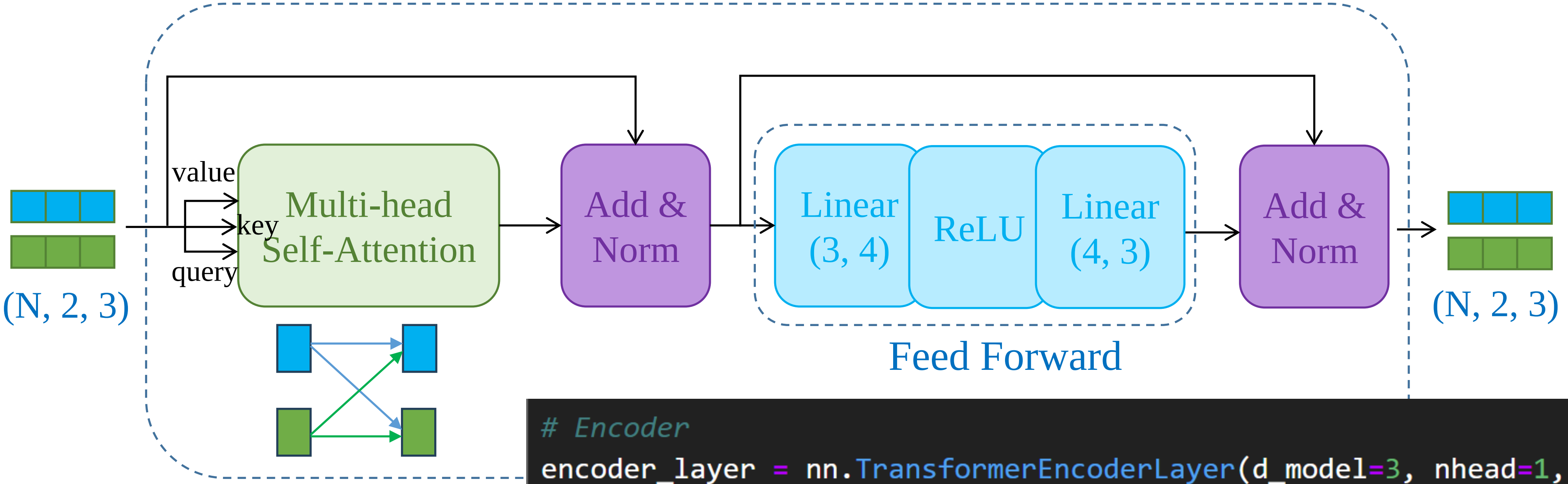


```
nn.TransformerEncoderLayer(d_model=3,
                           nhead=1,
                           batch_first=True)
nn.TransformerDecoderLayer(d_model=3,
                           nhead=1,
                           batch_first=True)


# feed forward
src = torch.Tensor([[[0.48, 0.44, 0.71],
                     [0.65, 0.80, 0.79]]])

tgt = torch.Tensor([[[0.3516, 0.9509, 0.2771],
                     [0.1993, 0.0177, 0.2628],
                     [0.0774, 0.5253, 0.6413],
                     [0.6749, 0.5501, 0.1641]]])

context = encoder_layer(src)
mask = torch.triu(torch.ones(4, 4),
                  diagonal=1).bool()
out = decoder_layer(tgt, context,
                    tgt_mask=mask)
```

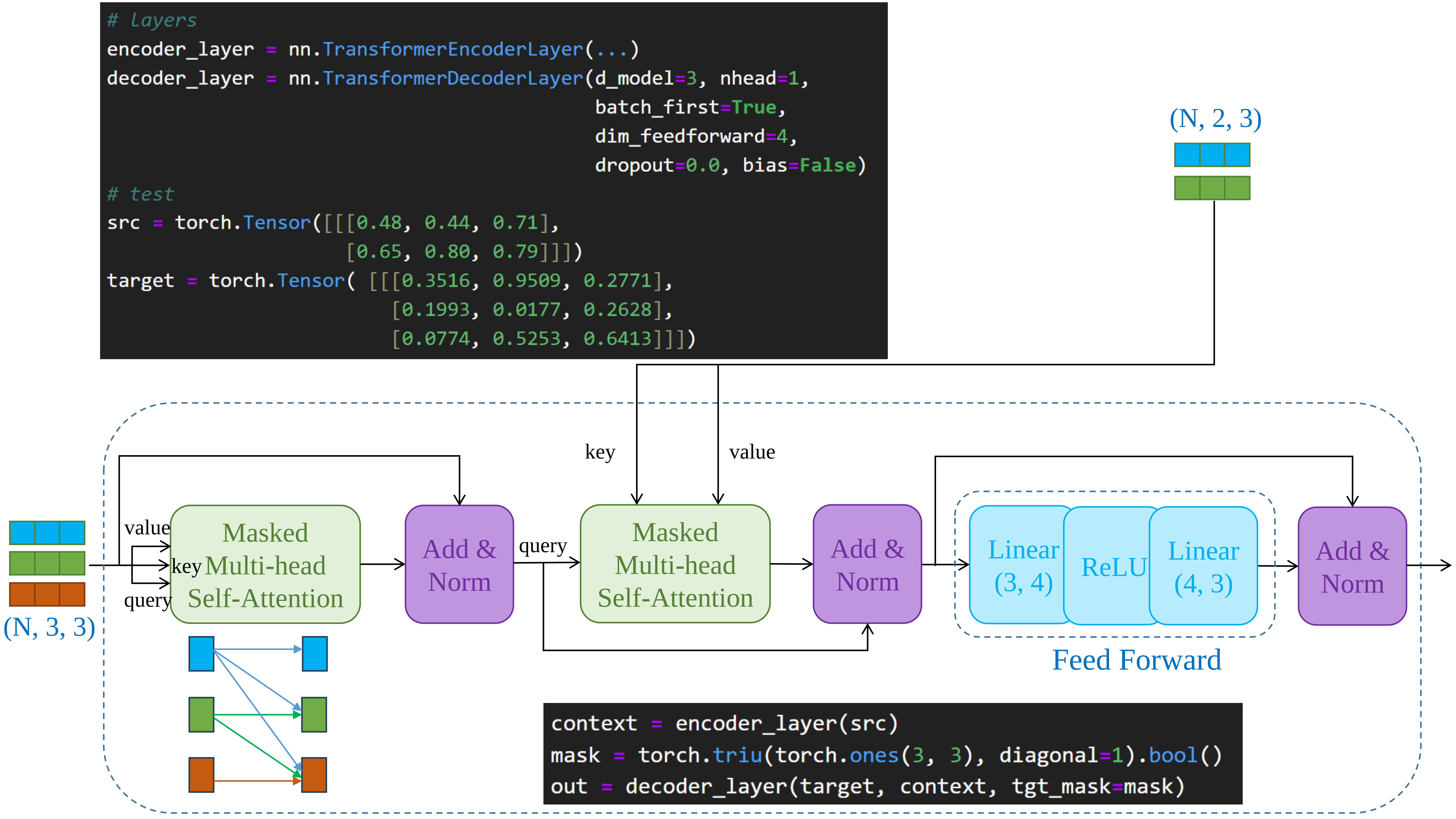


Transformer in PyTorch

❖ Transformer Encoder

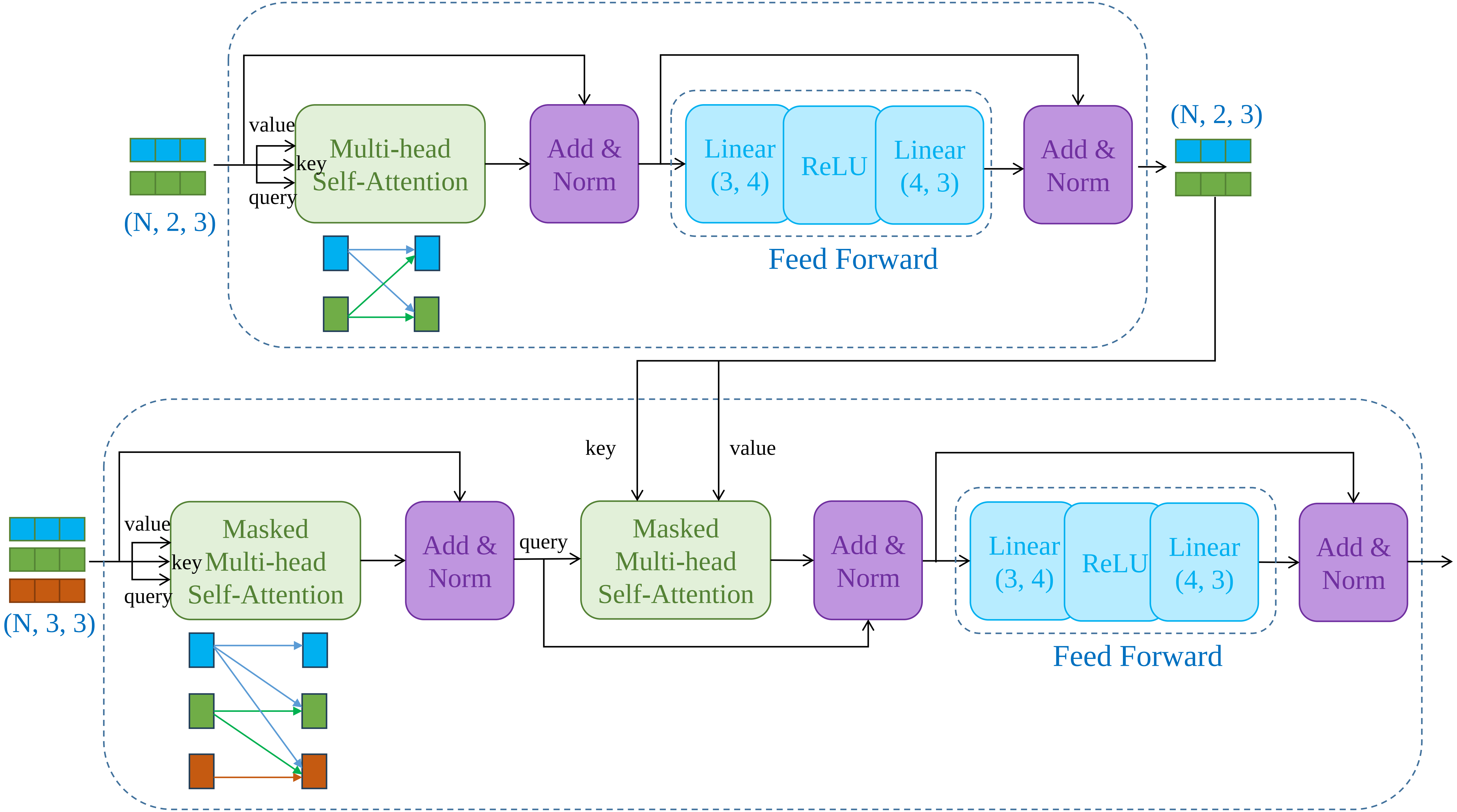


```python
# Encoder
encoder_layer = nn.TransformerEncoderLayer(d_model=3, nhead=1,
                                           batch_first=True,
                                           dim_feedforward=4,
                                           dropout=0.0, bias=False)

# test
src = torch.Tensor([[[0.48, 0.44, 0.71],
                     [0.65, 0.80, 0.79]]])
context = encoder_layer(src)
```



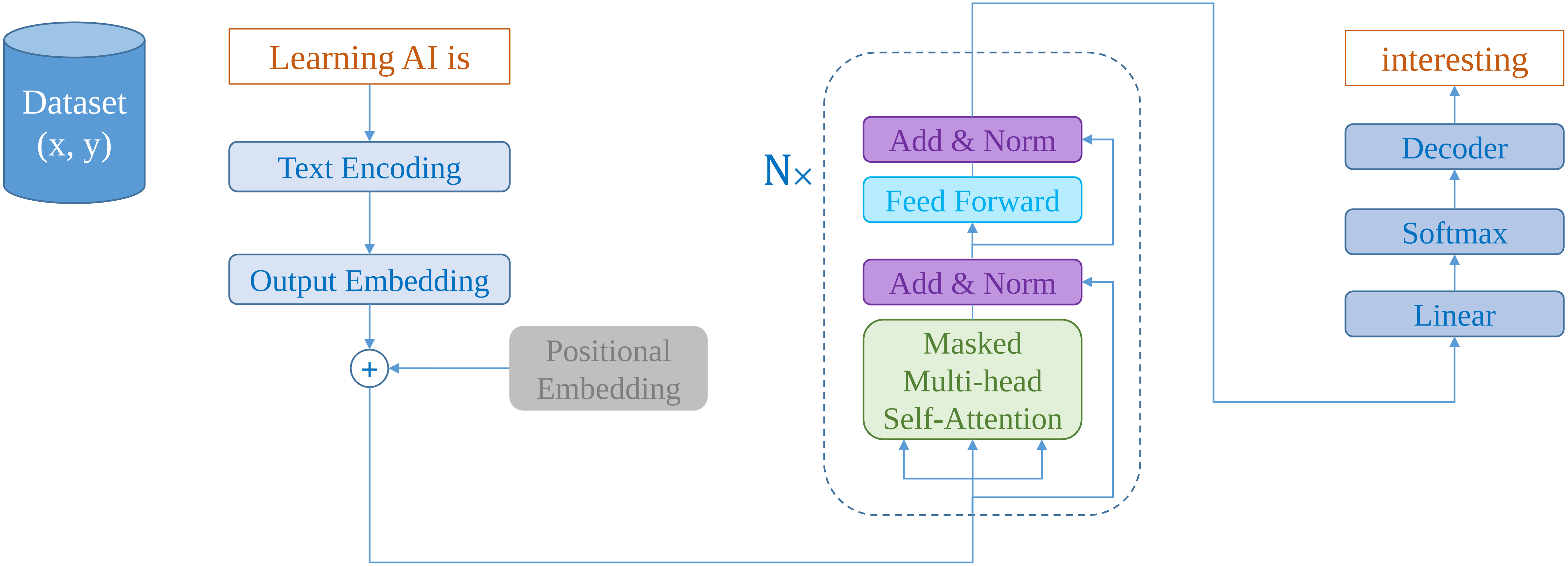
```python
# layers
encoder_layer = nn.TransformerEncoderLayer(...)
decoder_layer = nn.TransformerDecoderLayer(d_model=3, nhead=1,
                                           batch_first=True,
                                           dim_feedforward=4,
                                           dropout=0.0, bias=False)
# test
src = torch.Tensor([[[0.48, 0.44, 0.71],
                     [0.65, 0.80, 0.79]]])
target = torch.Tensor( [[[0.3516, 0.9509, 0.2771],
                         [0.1993, 0.0177, 0.2628],
                         [0.0774, 0.5253, 0.6413]]])
```

```python
context = encoder_layer(src)
mask = torch.triu(torch.ones(3, 3), diagonal=1).bool()
out = decoder_layer(target, context, tgt_mask=mask)
```


(N, 2, 3)
value
key
query
Multi-head Self-Attention
Add & Norm
Linear (3, 4)
ReLU
Linear (4, 3)
Add & Norm
Feed Forward
(N, 2, 3)
key
value
(N, 3, 3)
value
key
query
Masked Multi-head Self-Attention
Add & Norm
query
Masked Multi-head Self-Attention
Add & Norm
Linear (3, 4)
ReLU
Linear (4, 3)
Add & Norm
Feed Forward

Text Generation

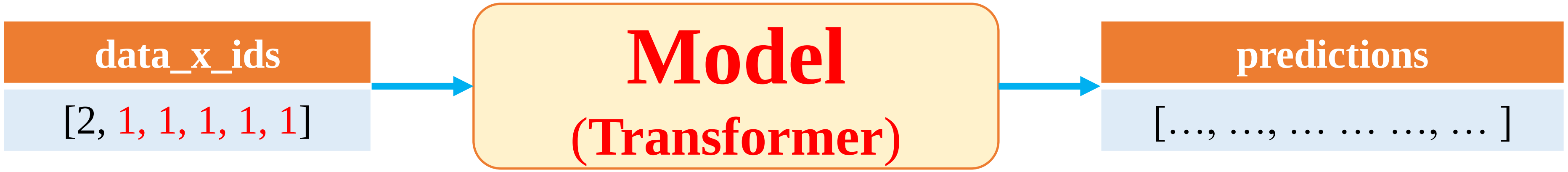
❖ Architecture



Example

Token	Id
<unk>	0
<pad>	1
<sos>	2
chí	3
cây	4
có	5
kẻ	6
nhớ	7
nên	8
quả	9
thì	10
trồng	11
ăn	12

vocab_size = 13



data_x_ids

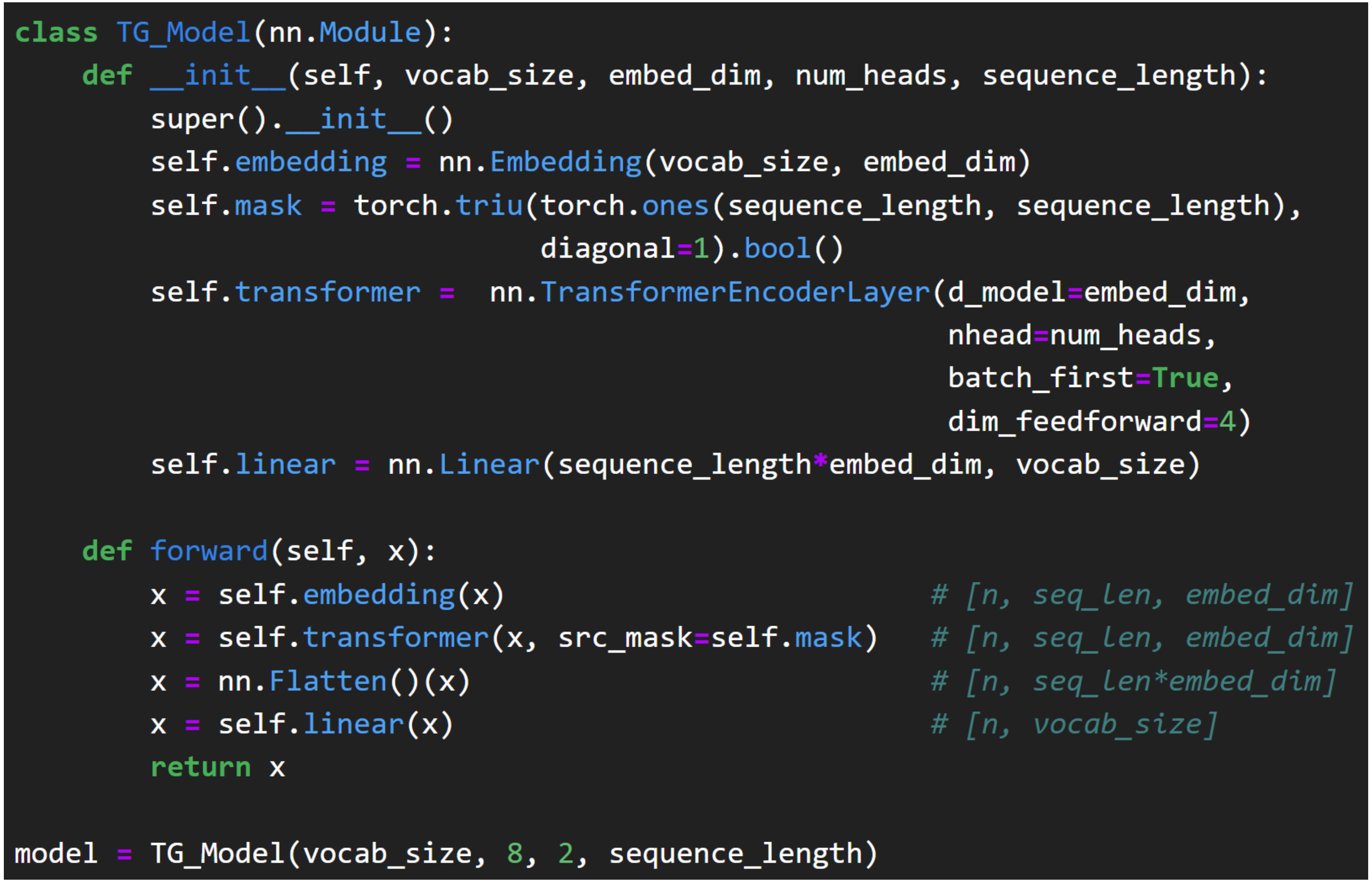
[2, 1, 1, 1, 1, 1]

Model
(Transformer)

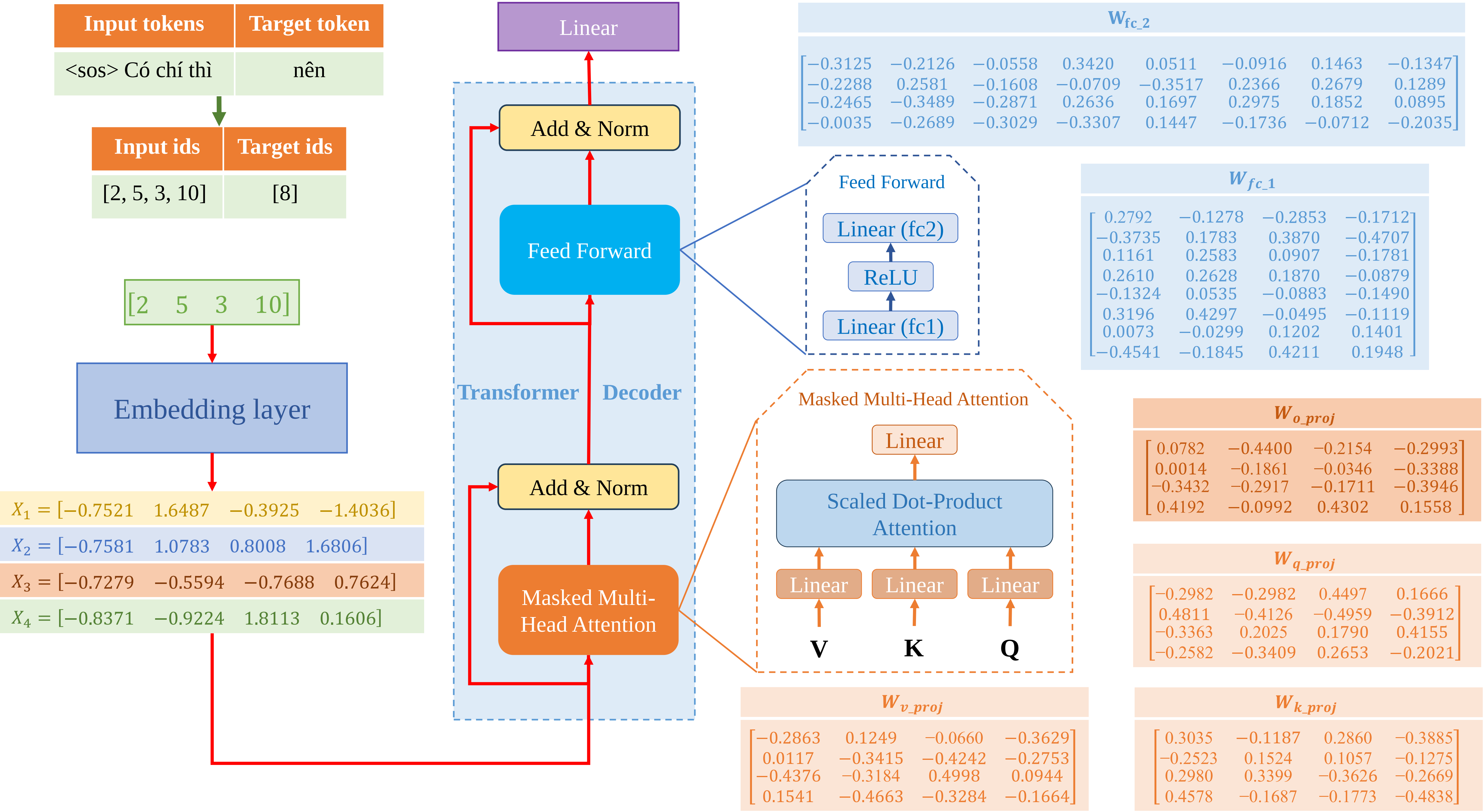
predictions

[…, …, … … …, …]

sequence_length = 6



```
class TG_Model(nn.Module):
    def __init__(self, vocab_size, embed_dim, num_heads, sequence_length):
        super().__init__()
        self.embedding = nn.Embedding(vocab_size, embed_dim)
        self.mask = torch.triu(torch.ones(sequence_length, sequence_length),
                               diagonal=1).bool()
        self.transformer =  nn.TransformerEncoderLayer(d_model=embed_dim,
                                                       nhead=num_heads,
                                                       batch_first=True,
                                                       dim_feedforward=4)
        self.linear = nn.Linear(sequence_length*embed_dim, vocab_size)

    def forward(self, x):
        x = self.embedding(x)                              # [n, seq_len, embed_dim]
        x = self.transformer(x, src_mask=self.mask)        # [n, seq_len, embed_dim]
        x = nn.Flatten()(x)                                # [n, seq_len*embed_dim]
        x = self.linear(x)                                 # [n, vocab_size]
        return x

model = TG_Model(vocab_size, 8, 2, sequence_length)
```



Input tokens	Target token
<sos> Có chí thì	nên

Input ids	Target ids
[2, 5, 3, 10]	[8]

$$X_1 = [-0.7521 \quad 1.6487 \quad -0.3925 \quad -1.4036]$$

$$X_2 = [-0.7581 \quad 1.0783 \quad 0.8008 \quad 1.6806]$$

$$X_3 = [-0.7279 \quad -0.5594 \quad -0.7688 \quad 0.7624]$$

$$X_4 = [-0.8371 \quad -0.9224 \quad 1.8113 \quad 0.1606]$$

$\mathbf{W_{fc\_2}}$

$$\begin{bmatrix} -0.3125 & -0.2126 & -0.0558 & 0.3420 & 0.0511 & -0.0916 & 0.1463 & -0.1347 \\ -0.2288 & 0.2581 & -0.1608 & -0.0709 & -0.3517 & 0.2366 & 0.2679 & 0.1289 \\ -0.2465 & -0.3489 & -0.2871 & 0.2636 & 0.1697 & 0.2975 & 0.1852 & 0.0895 \\ -0.0035 & -0.2689 & -0.3029 & -0.3307 & 0.1447 & -0.1736 & -0.0712 & -0.2035 \end{bmatrix}$$

$\boldsymbol{W_{fc\_1}}$

$$\begin{bmatrix} 0.2792 & -0.1278 & -0.2853 & -0.1712 \\ -0.3735 & 0.1783 & 0.3870 & -0.4707 \\ 0.1161 & 0.2583 & 0.0907 & -0.1781 \\ 0.2610 & 0.2628 & 0.1870 & -0.0879 \\ -0.1324 & 0.0535 & -0.0883 & -0.1490 \\ 0.3196 & 0.4297 & -0.0495 & -0.1119 \\ 0.0073 & -0.0299 & 0.1202 & 0.1401 \\ -0.4541 & -0.1845 & 0.4211 & 0.1948 \end{bmatrix}$$

$\boldsymbol{W_{o\_proj}}$

$$\begin{bmatrix} 0.0782 & -0.4400 & -0.2154 & -0.2993 \\ 0.0014 & -0.1861 & -0.0346 & -0.3388 \\ -0.3432 & -0.2917 & -0.1711 & -0.3946 \\ 0.4192 & -0.0992 & 0.4302 & 0.1558 \end{bmatrix}$$

$\boldsymbol{W_{q\_proj}}$

$$\begin{bmatrix} -0.2982 & -0.2982 & 0.4497 & 0.1666 \\ 0.4811 & -0.4126 & -0.4959 & -0.3912 \\ -0.3363 & 0.2025 & 0.1790 & 0.4155 \\ -0.2582 & -0.3409 & 0.2653 & -0.2021 \end{bmatrix}$$

$\boldsymbol{W_{v\_proj}}$

$$\begin{bmatrix} -0.2863 & 0.1249 & -0.0660 & -0.3629 \\ 0.0117 & -0.3415 & -0.4242 & -0.2753 \\ -0.4376 & -0.3184 & 0.4998 & 0.0944 \\ 0.1541 & -0.4663 & -0.3284 & -0.1664 \end{bmatrix}$$

$\boldsymbol{W_{k\_proj}}$

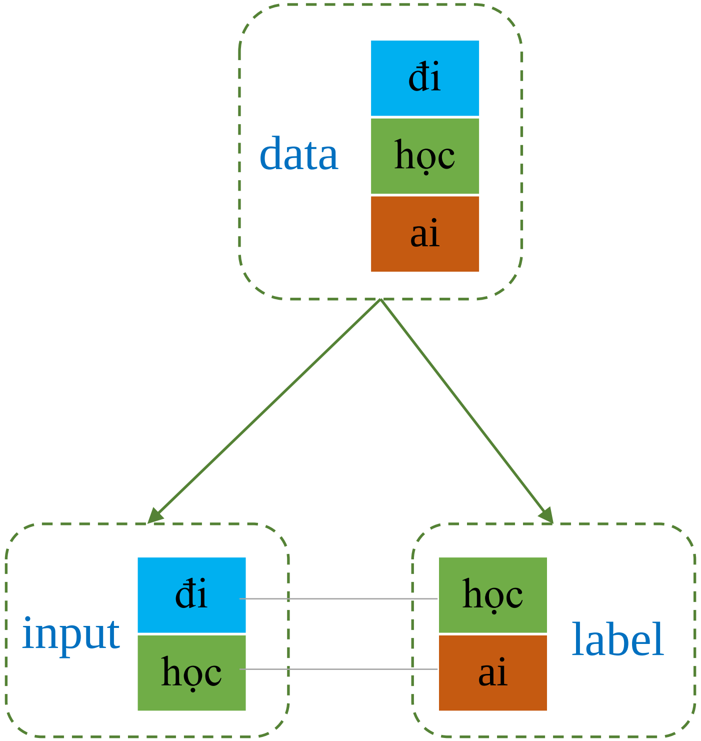
$$\begin{bmatrix} 0.3035 & -0.1187 & 0.2860 & -0.3885 \\ -0.2523 & 0.1524 & 0.1057 & -0.1275 \\ 0.2980 & 0.3399 & -0.3626 & -0.2669 \\ 0.4578 & -0.1687 & -0.1773 & -0.4838 \end{bmatrix}$$

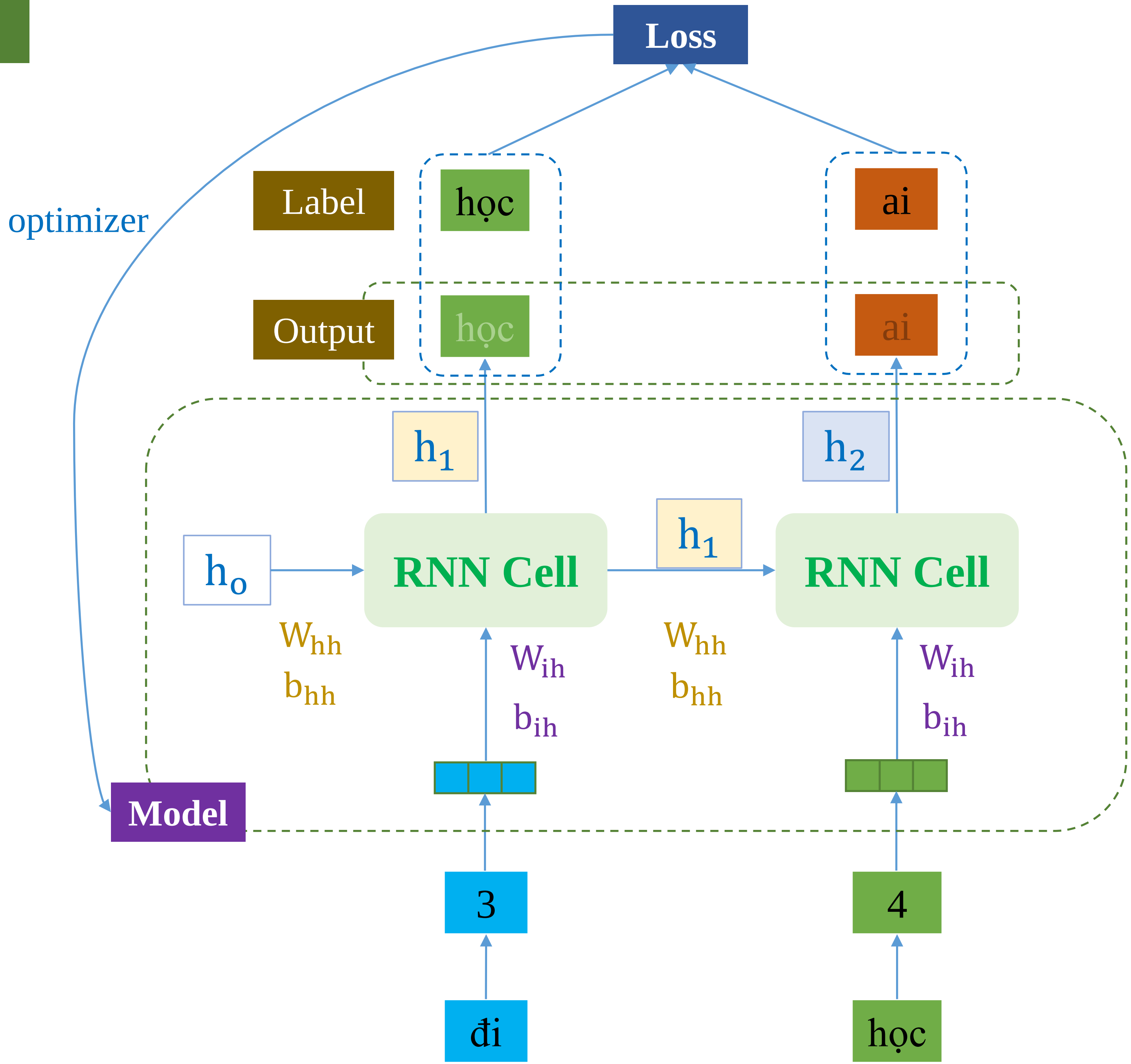
Outline

- ➢ Text Generation Using Transformer
- ➢ An improved Approach to Text Generation
- ➢ Machine Translation Using RNN
- ➢ Machine Translation Using Transformer

An improved approach

*Linear can be added (where?)





An improved approach

hidden_dim = vocab_size = 4

[-0.88, 0.81, 0.77]
[-1.27, 0.84, 0.04]

Embedded Input

W_{ih}

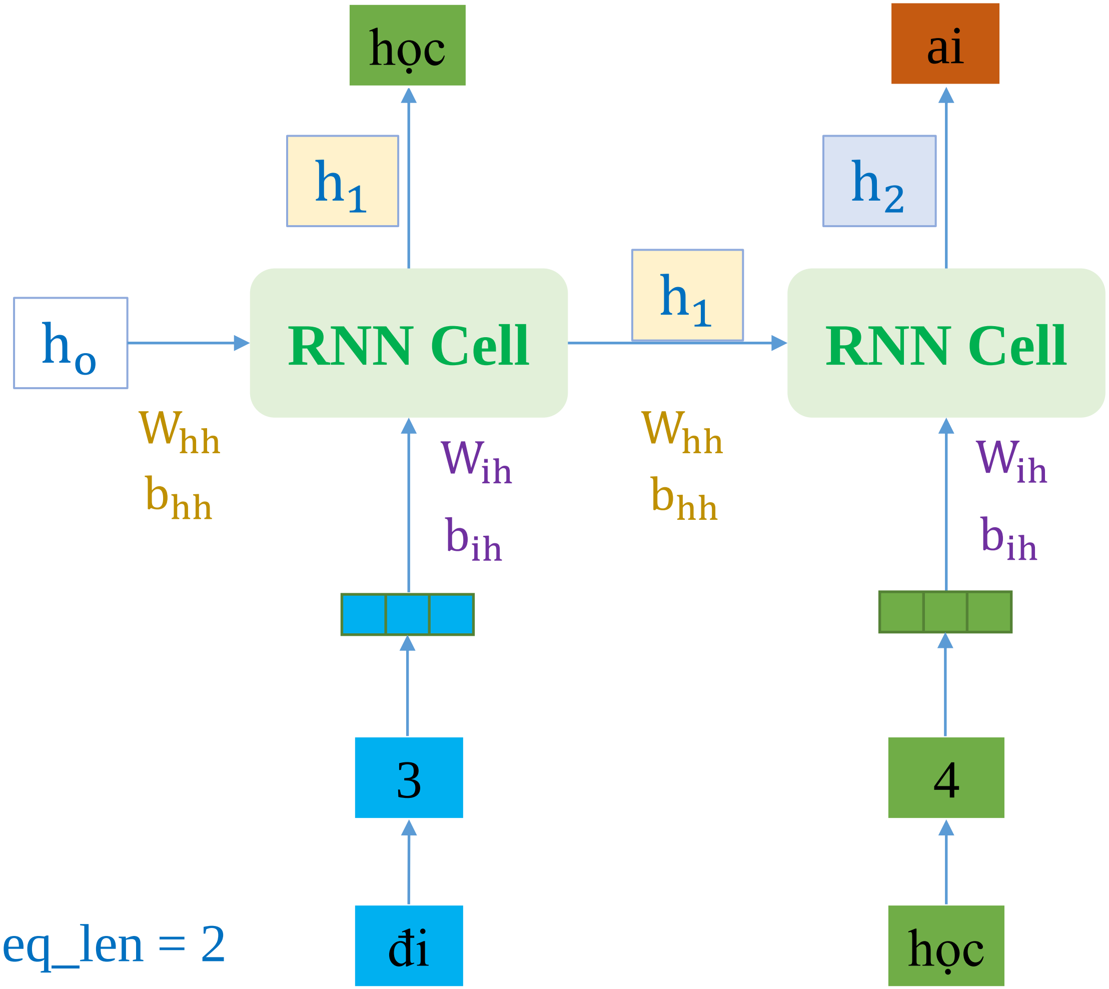
[0.25,   0.05, 0.17]
[-0.11, 0.35, 0.22]
[0.44, -0.22, -0.42]
[0.38, 0.36, -0.01]

b_{ih} [-0.47, 0.13, 0.46, -0.15]

W_{hh}

[0.38, -0.06, 0.37, -0.19]
[-0.16, 0.42, 0.08, -0.02]
[-0.36, -0.01, 0.17, 0.39]
[-0.43, 0.09, 0.33, 0.03]

b_{hh} [0.43, 0.16, 0.34, -0.39]



seq_len = 2

*Linear can be added

Output

[-0.08, 0.68, -0.08, -0.53]
[-0.30, 0.77, -0.15, -0.58]

Hidden [-0.30, 0.77, -0.15, -0.58]

Training

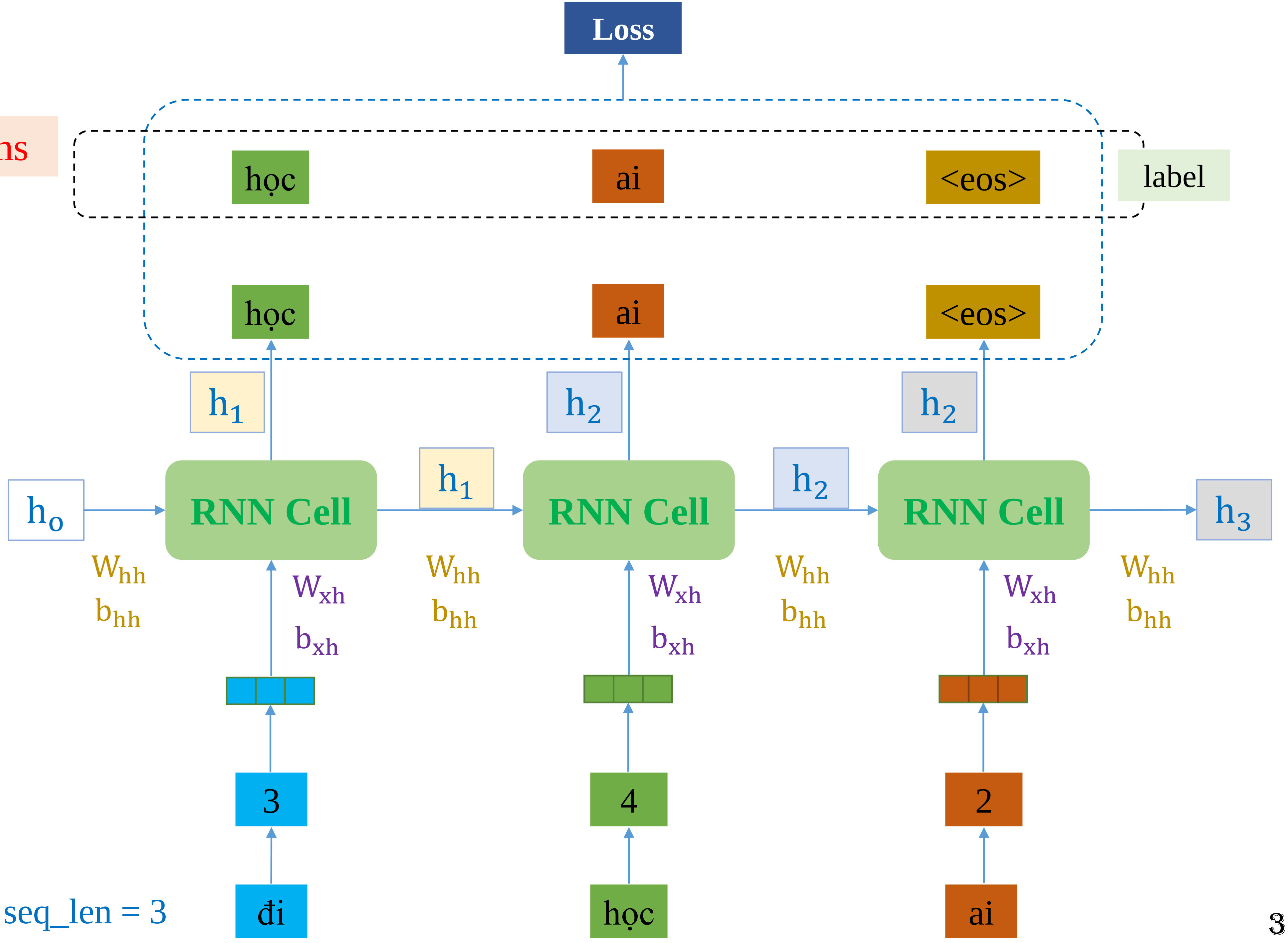
Add more special tokens

đi

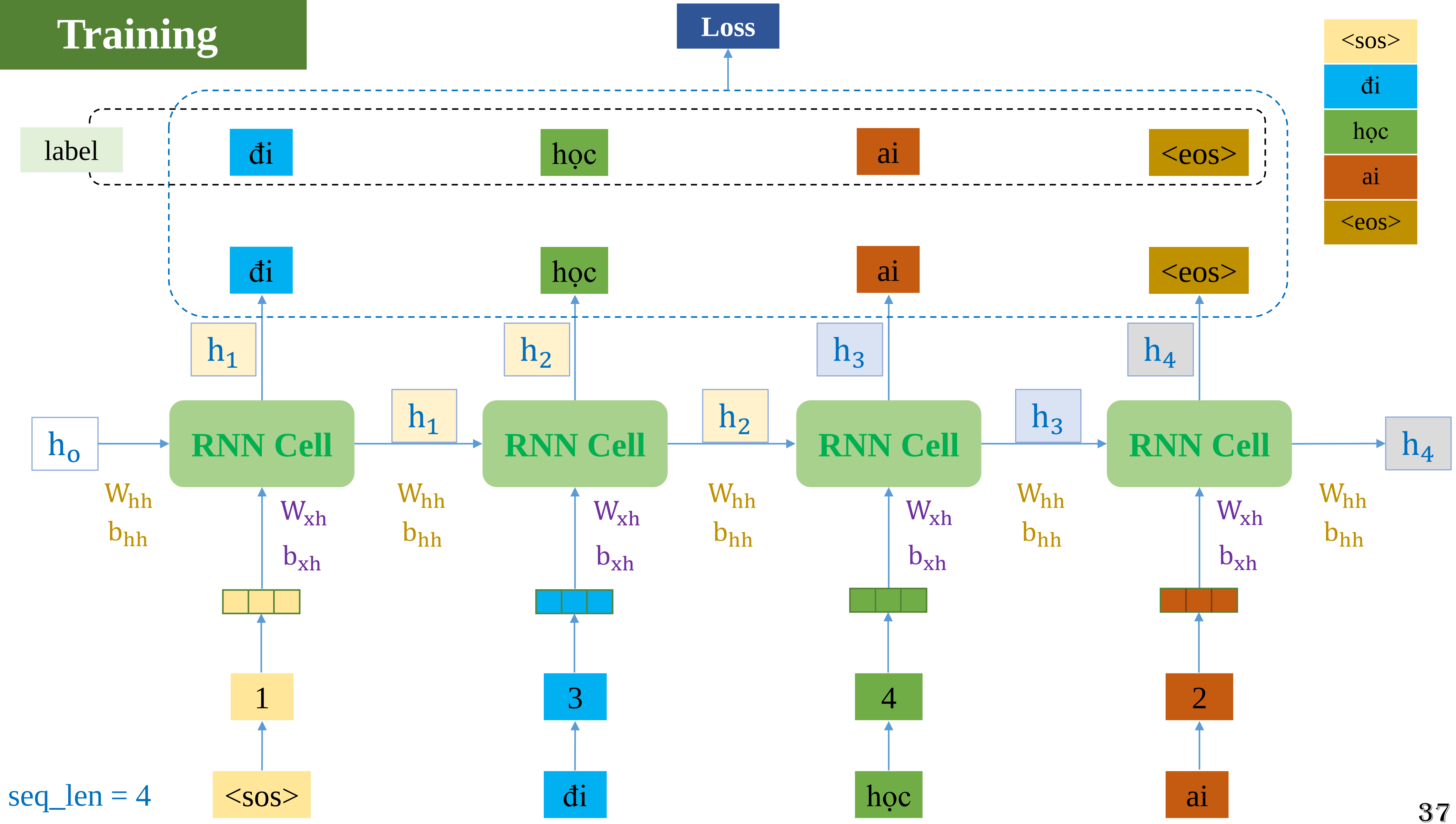
học

ai

<eos>



Training

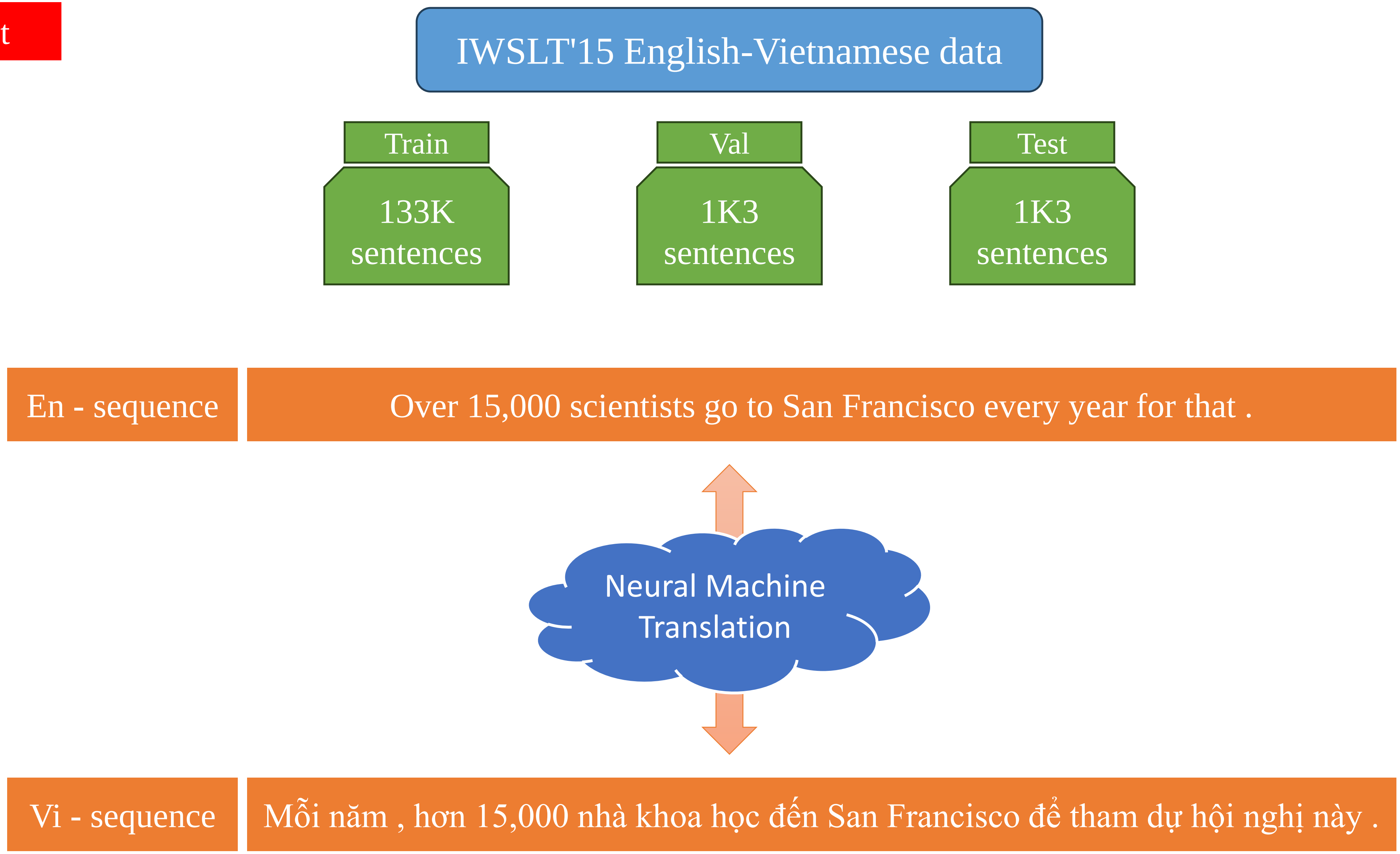


seq_len = 4

Outline

- Text Generation Using Transformer
- An improved Approach to Text Generation
- Machine Translation Using RNN
- Machine Translation Using Transformer

Dataset



IWSLT'15 English-Vietnamese data

Train

133K sentences

Val

1K3 sentences

Test

1K3 sentences

En - sequence

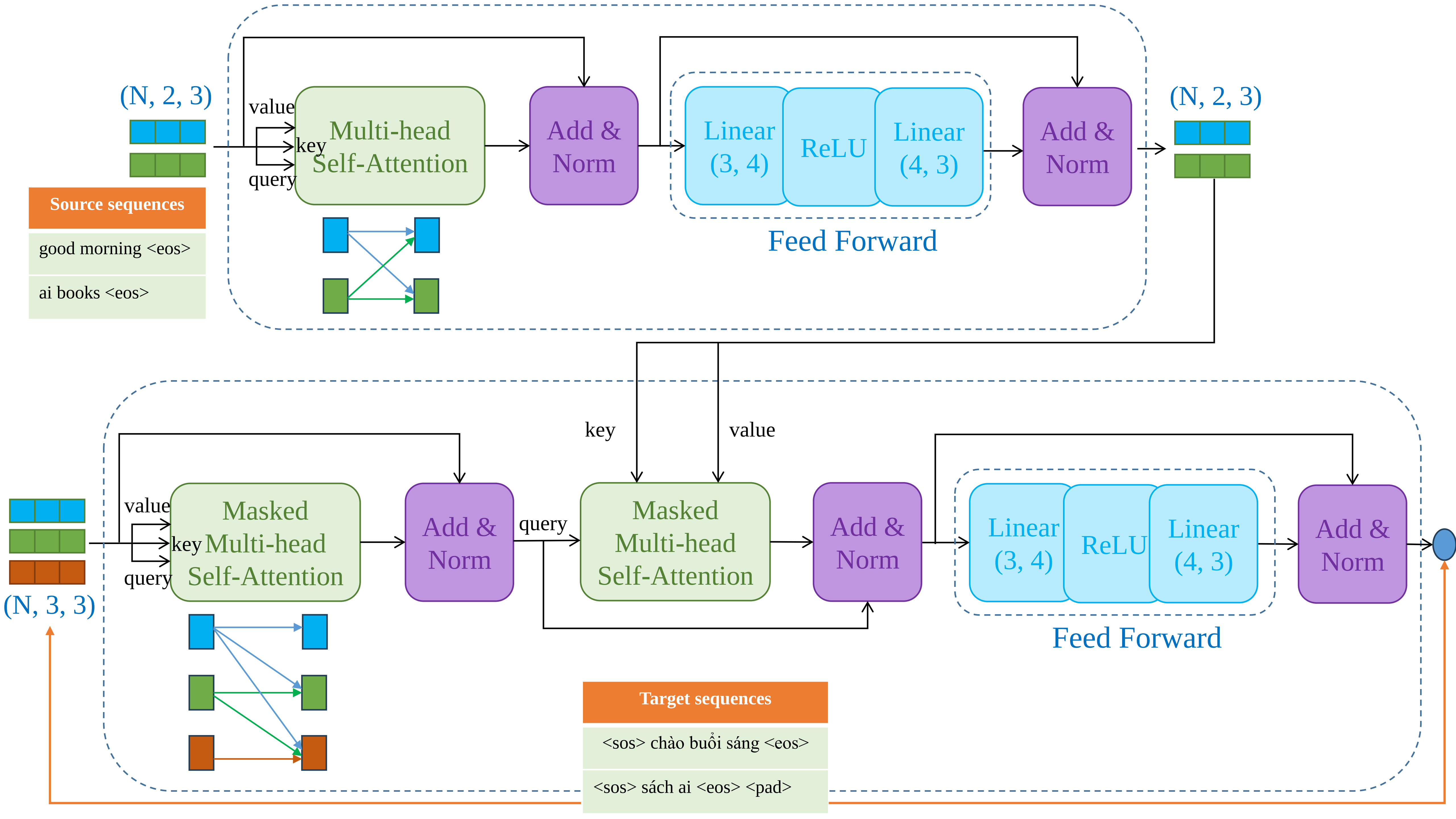
Over 15,000 scientists go to San Francisco every year for that .

Neural Machine Translation

Vi - sequence

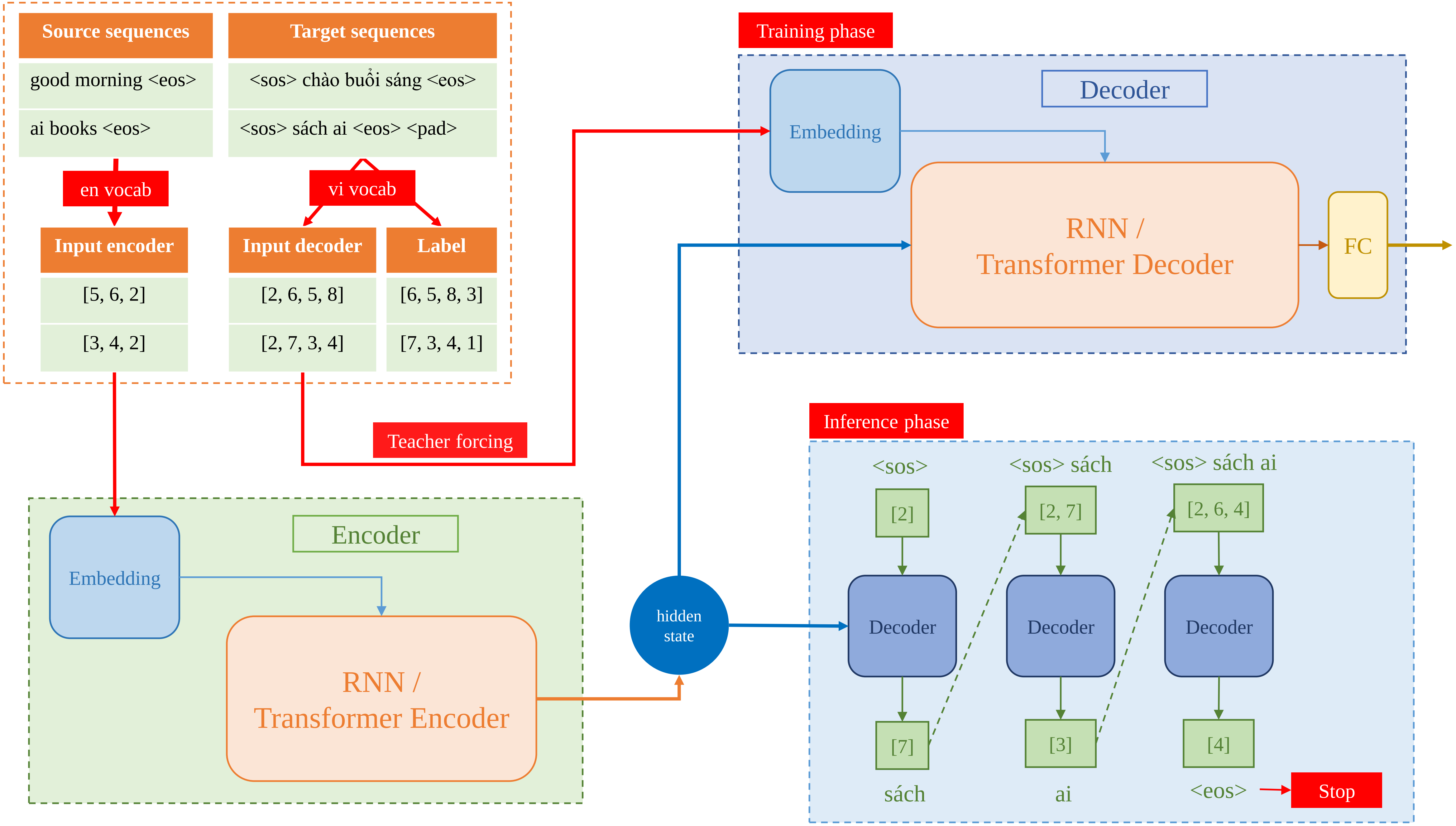
Mỗi năm , hơn 15,000 nhà khoa học đến San Francisco để tham dự hội nghị này .

Ref: https://nlp.stanford.edu/projects/nmt/



Source sequences
good morning <eos>
ai books <eos>

Target sequences
<sos> chào buổi sáng <eos>
<sos> sách ai <eos> <pad>



Source sequences	Target sequences
good morning <eos>	<sos> chào buổi sáng <eos>
ai books <eos>	<sos> sách ai <eos> <pad>

en vocab

vi vocab

Input encoder	Input decoder	Label
[5, 6, 2]	[2, 6, 5, 8]	[6, 5, 8, 3]
[3, 4, 2]	[2, 7, 3, 4]	[7, 3, 4, 1]

Teacher forcing

Training phase

Decoder

Embedding

RNN /
Transformer Decoder

FC

Encoder

Embedding

RNN /
Transformer Encoder

hidden
state

Inference phase

<sos>

[2]

Decoder

[7]

sách

<sos> sách

[2, 7]

Decoder

[3]

ai

<sos> sách ai

[2, 6, 4]

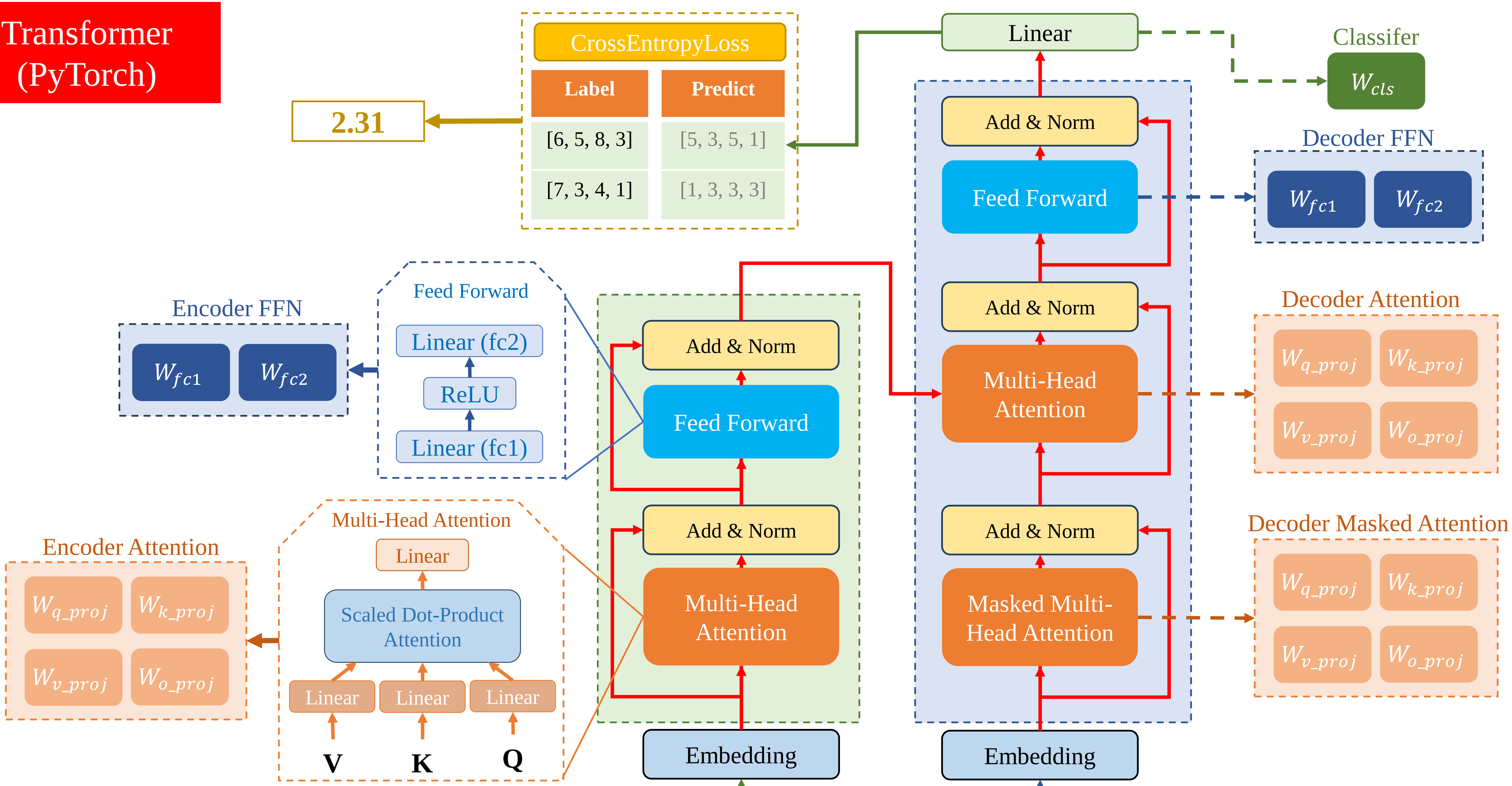
Decoder

[4]

<eos>

Stop

Transformer (PyTorch)



Label	Predict
[6, 5, 8, 3]	[5, 3, 5, 1]
[7, 3, 4, 1]	[1, 3, 3, 3]

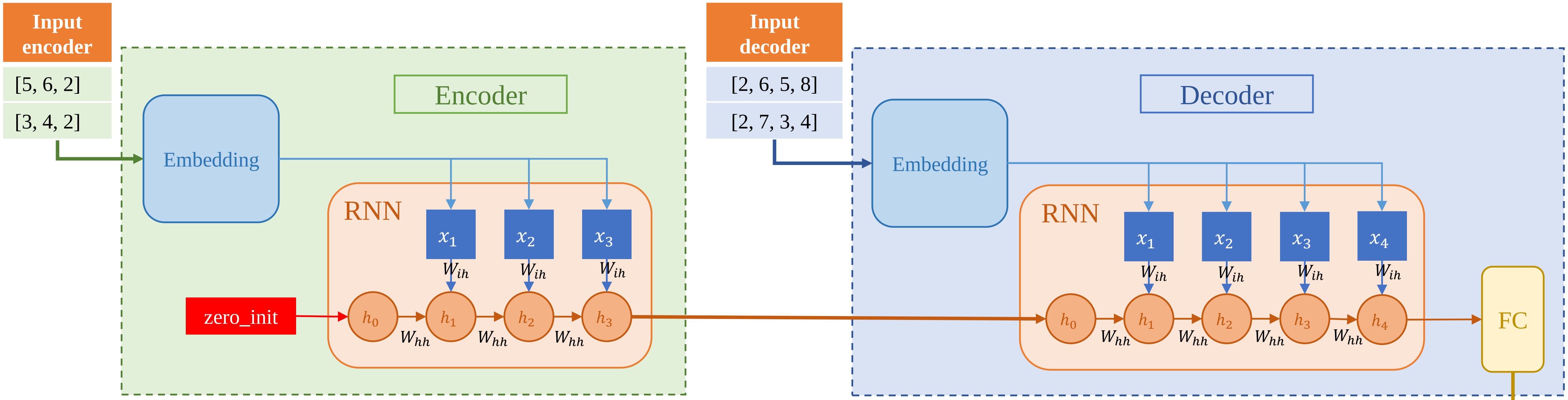
Source sequences
good morning <eos>
ai books <eos>

Input encoder
[5, 6, 2]
[3, 4, 2]

Input decoder
[2, 6, 5, 8]
[2, 7, 3, 4]

Target sequences
<sos> chào buổi sáng <eos>
<sos> sách ai <eos> <pad>

RNN



$\boldsymbol{W_{ih}}$

$$\begin{bmatrix} 0.3195 & -0.2901 & \dots & \dots & 0.1259 & -0.1406 \\ 0.1251 & -0.0851 & \dots & \cdots & -0.2435 & -0.2435 \\ 0.3672 & 0.1360 & \dots & \cdots & -0.4049 & -0.3194 \\ -0.2746 & 0.1654 & \dots & \dots & -0.2108 & -0.2783 \\ 0.2166 & -0.1650 & \dots & \dots & 0.2335 & -0.3172] \\ -0.2060 & 0.1245 & \dots & \dots & 0.2433 & 0.2775 \end{bmatrix}$$

$\boldsymbol{W_{hh}}$

$$\begin{bmatrix} -0.2961 & -0.2180 & \dots & \dots & -0.1447 & -0.3950 \\ -0.2338 & 0.1020 & \dots & \dots & 0.0096 & -0.2789 \\ -0.3464 & -0.2248 & \dots & \dots & 0.4081 & 0.0771 \\ 0.1258 & -0.3808 & \dots & \dots & 0.0638 & -0.3592 \\ -0.1759 & -0.2444 & \dots & \dots & -0.0283 & -0.2766 \\ -0.2802 & -0.2382 & \dots & \dots & 0.3423 & -0.0810 \end{bmatrix}$$

$\boldsymbol{W_{ih}}$

$$\begin{bmatrix} -0.3609 & -0.2455 & \dots & \dots & 0.0591 & -0.1057 \\ 0.1689 & -0.1555 & \dots & \cdots & -0.1856 & -0.0818 \\ -0.4061 & 0.2732 & \dots & \cdots & -0.2847 & -0.4029 \\ -0.3316 & 0.3044 & \dots & \dots & 0.2139 & 0.1033 \\ -0.0040 & -0.3105 & \dots & \dots & 0.1671 & -0.2004 \\ -0.0822 & -0.2349 & \dots & \dots & -0.2668 & 0.1354 \end{bmatrix}$$

$\boldsymbol{W_{hh}}$

$$\begin{bmatrix} -0.1213 & 0.2520 & \dots & \dots & -0.0720 & -0.1979 \\ -0.1249 & -0.3887 & \dots & \cdots & 0.2052 & 0.1853 \\ 0.2917 & -0.3131 & \dots & \cdots & 0.1515 & 0.3834 \\ -0.0576 & -0.0032 & \dots & \dots & 0.1959 & -0.4053 \\ 0.2534 & 0.3055 & \dots & \dots & -0.3354 & 0.0918 \\ 0.2255 & -0.4063 & \dots & \dots & -0.0357 & -0.2009 \end{bmatrix}$$

Label	Predict
[6, 5, 8, 3]	[1, 1, 1, 6]
[7, 3, 4, 1]	[1, 1, 1, 1]

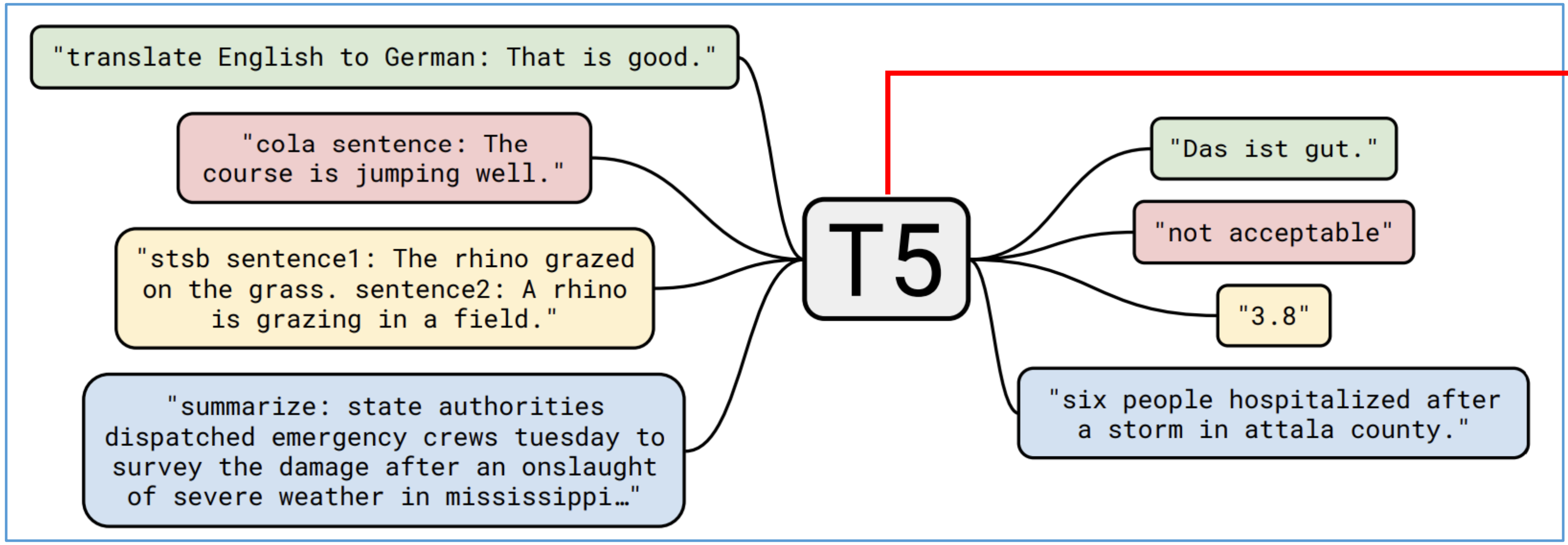
CrossEntropyLoss

2.33

Model

T5: Text-to-Text Transfer Transformer

Original transformer



"translate English to German: That is good."

"cola sentence: The course is jumping well."

"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."

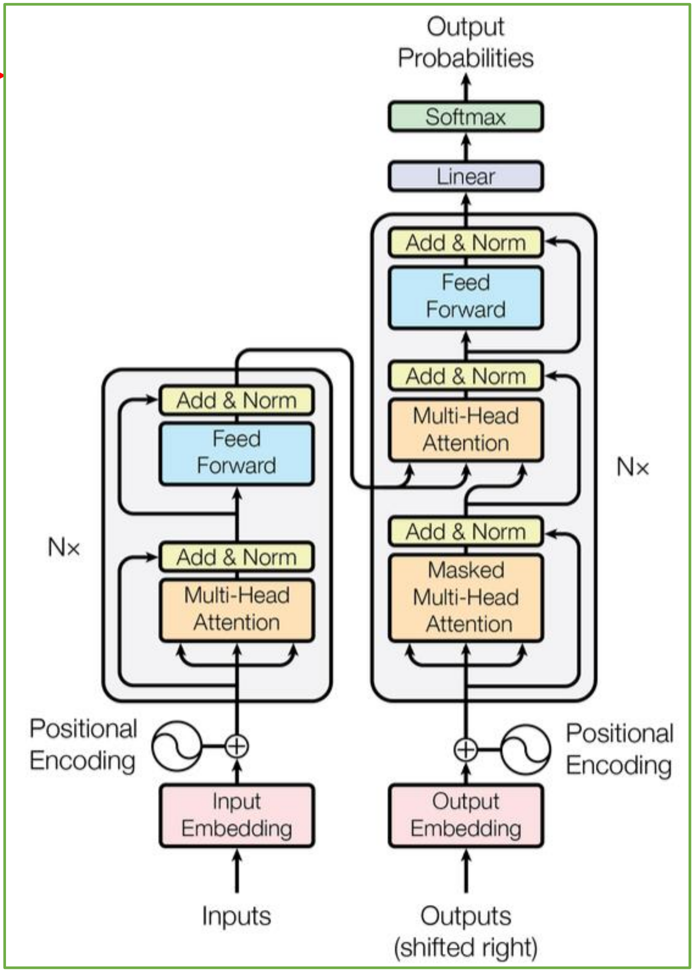
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"

"Das ist gut."

"not acceptable"

"3.8"

"six people hospitalized after a storm in attala county."



Encoder-Decoder model pre-trained on a **multi-task mixture** of **unsupervised** and **supervised** tasks

Unsupervised

Predict masked spans

Supervised

seq2seq
input-output mapping

Ref: https://arxiv.org/pdf/1910.10683.pdf