Deep Learning in NLP

2023-04-11

Outline

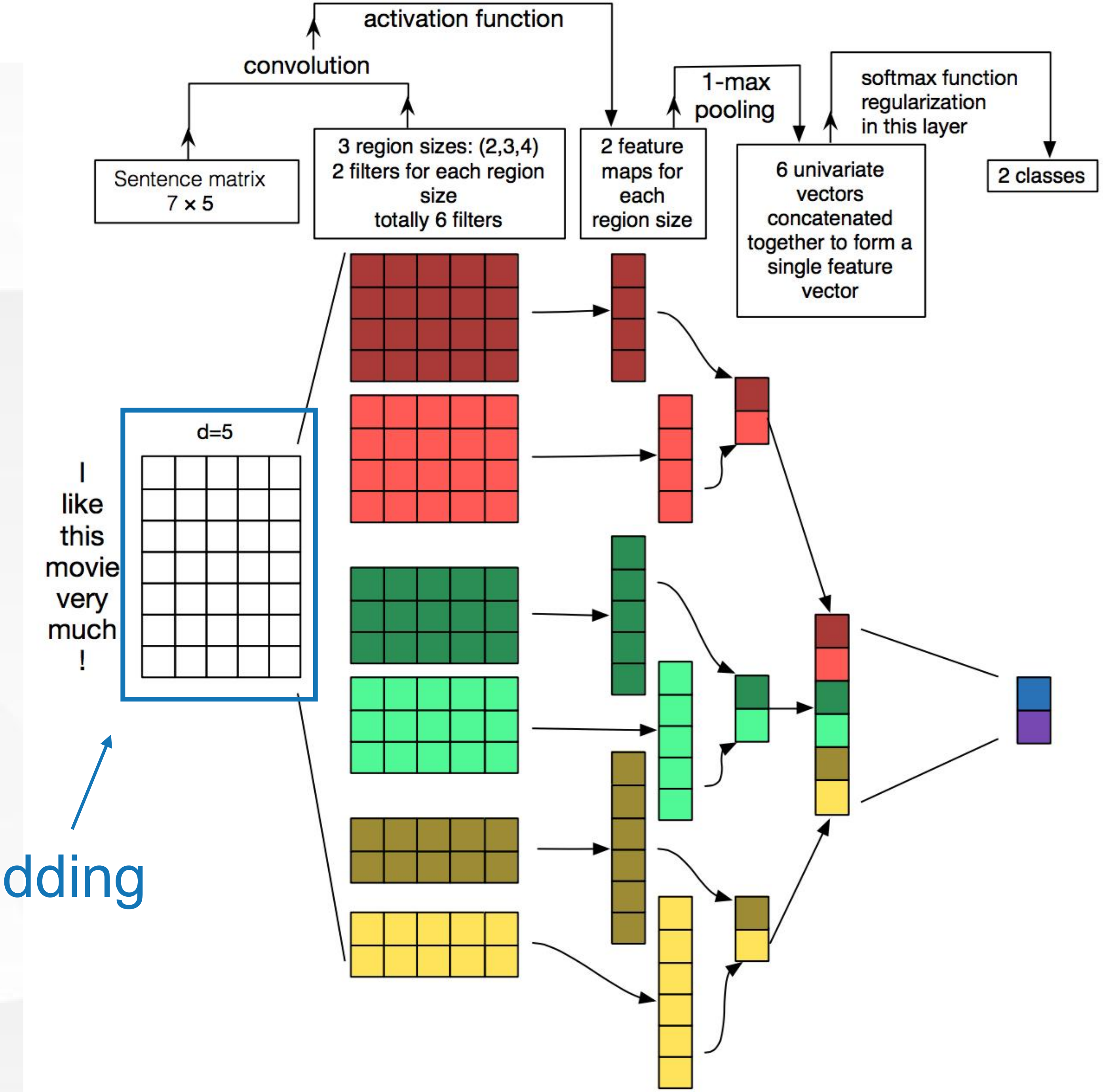
- Algorithm Introduction
  - CNN
  - RNN
- Implementation Details
- Experiment Result

Out

Outline

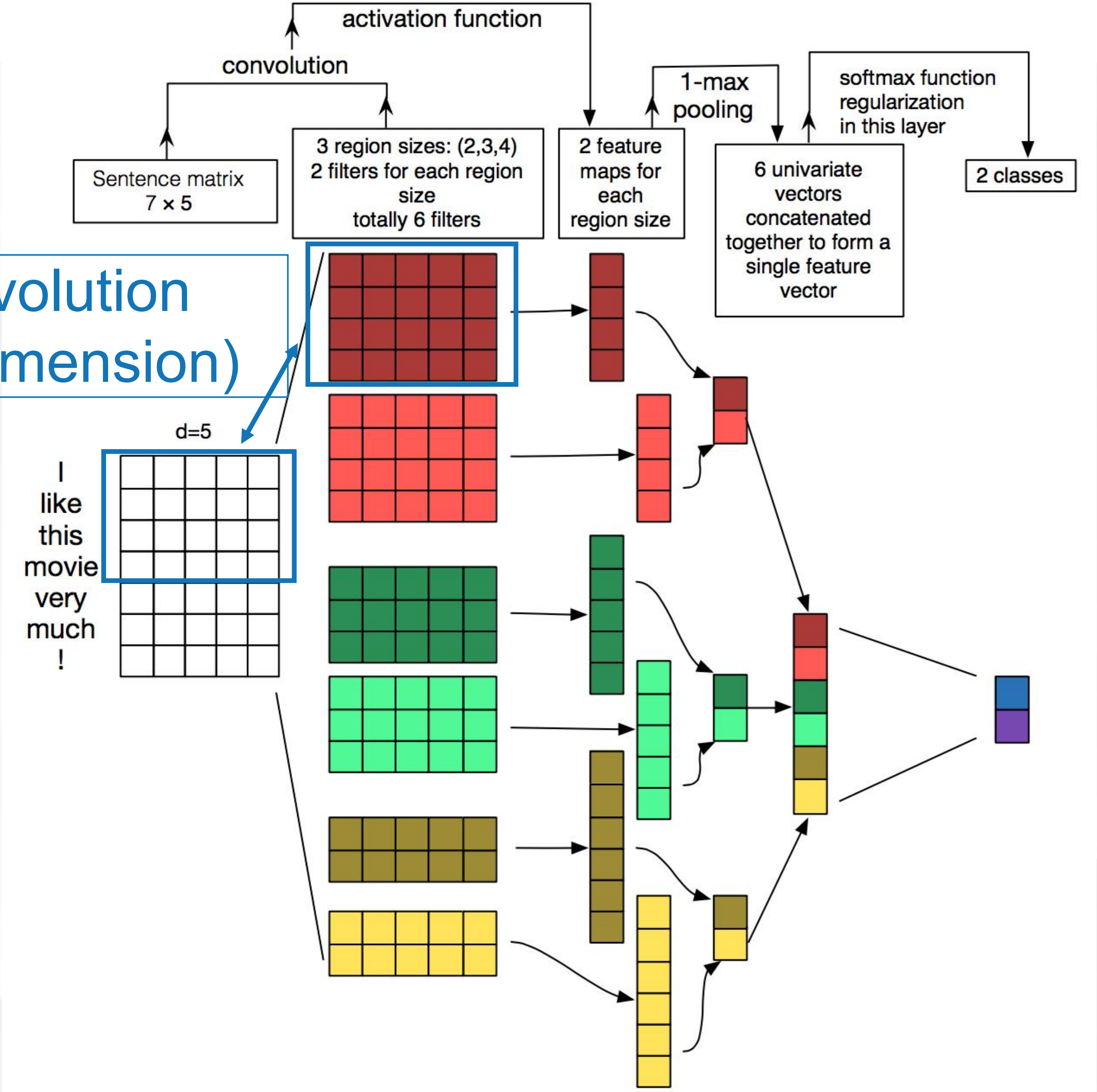
- Algorithm Introduction
  - CNN
  - RNN
- Implementation Details
- Experiment Result

2.1 CNN



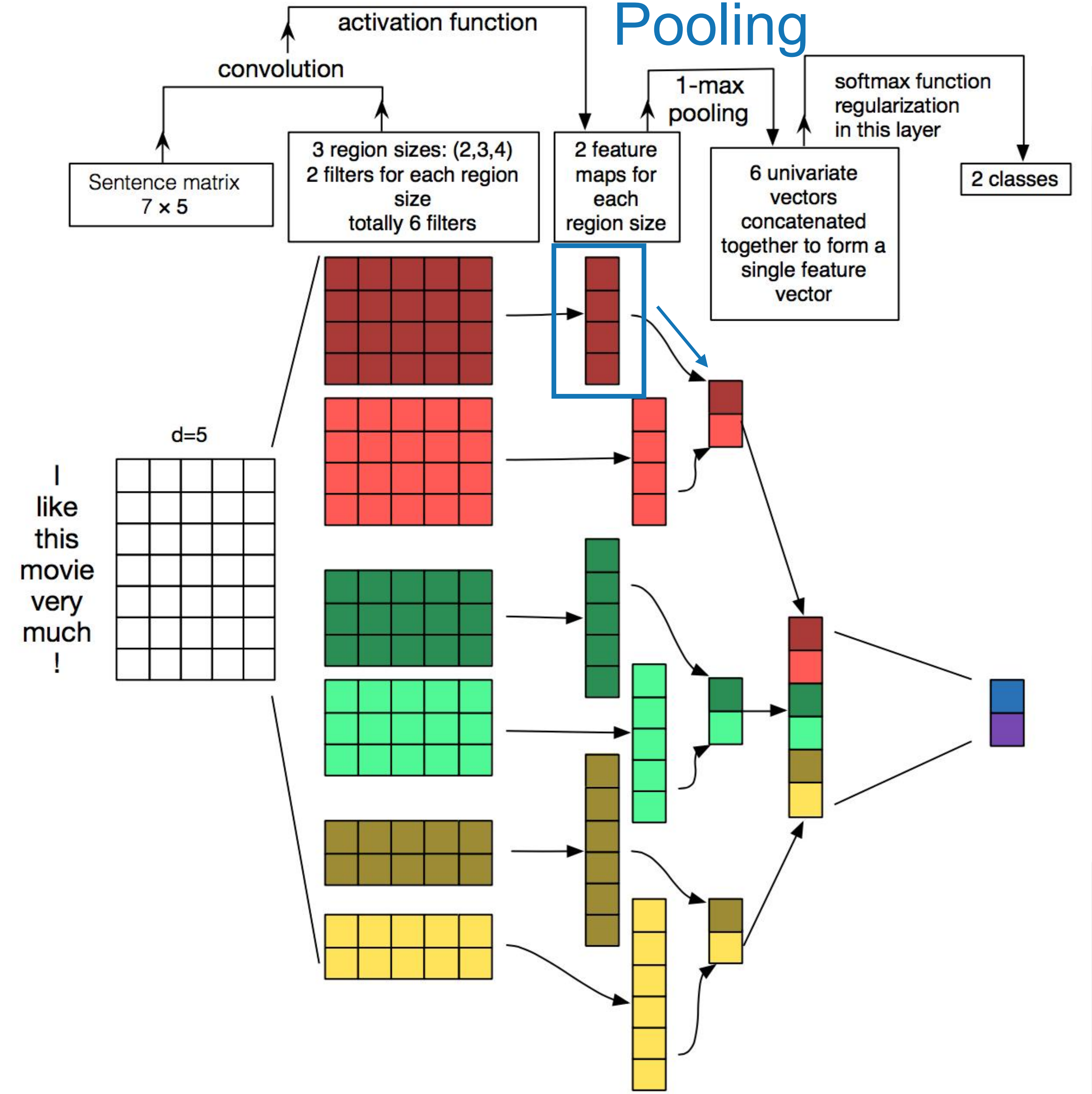
Word Embedding

2.1 CNN



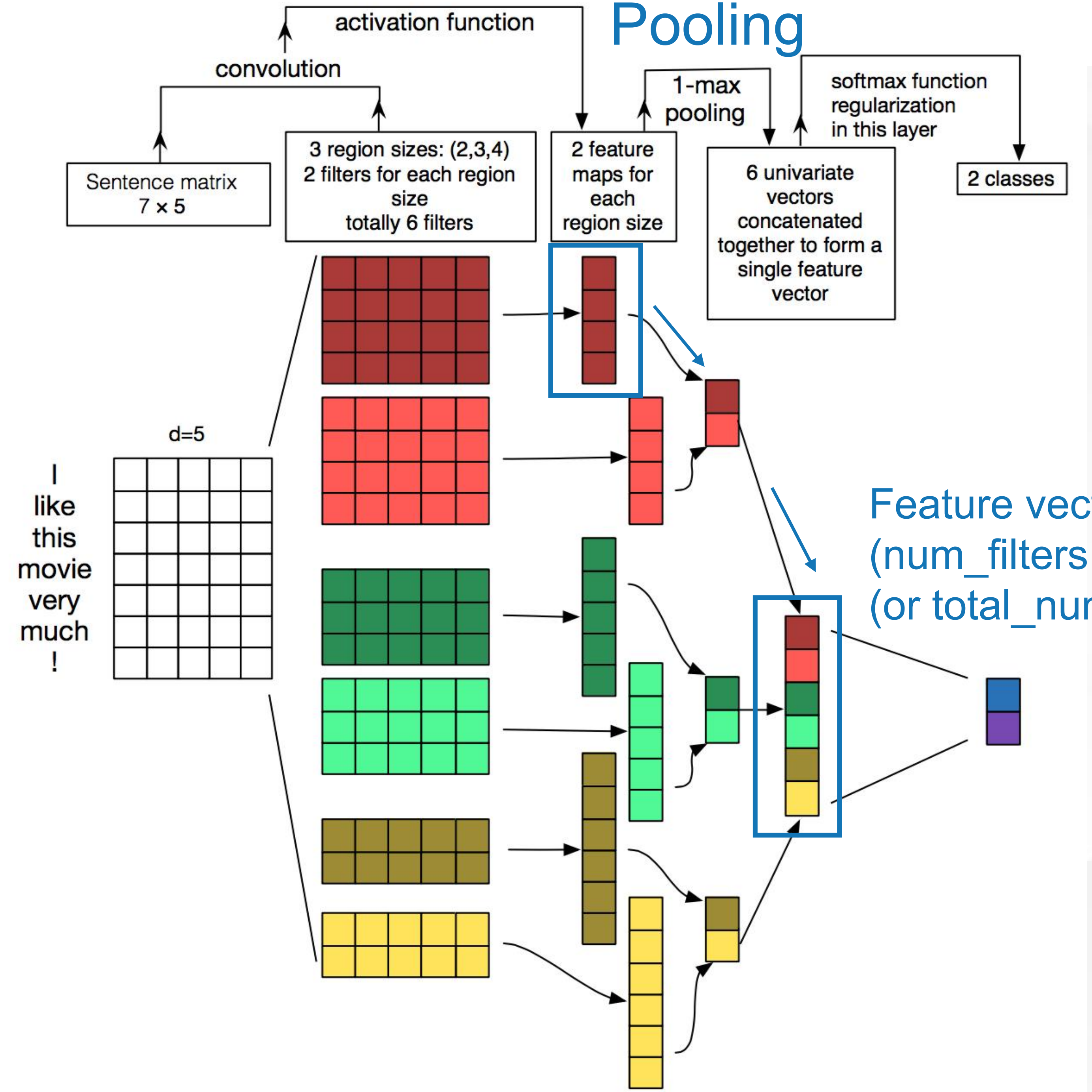
activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
Convolution (1-dimension)
d=5
I
like
this
movie
very
much
!

2.1 CNN


activation function
convolution
Pooling
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

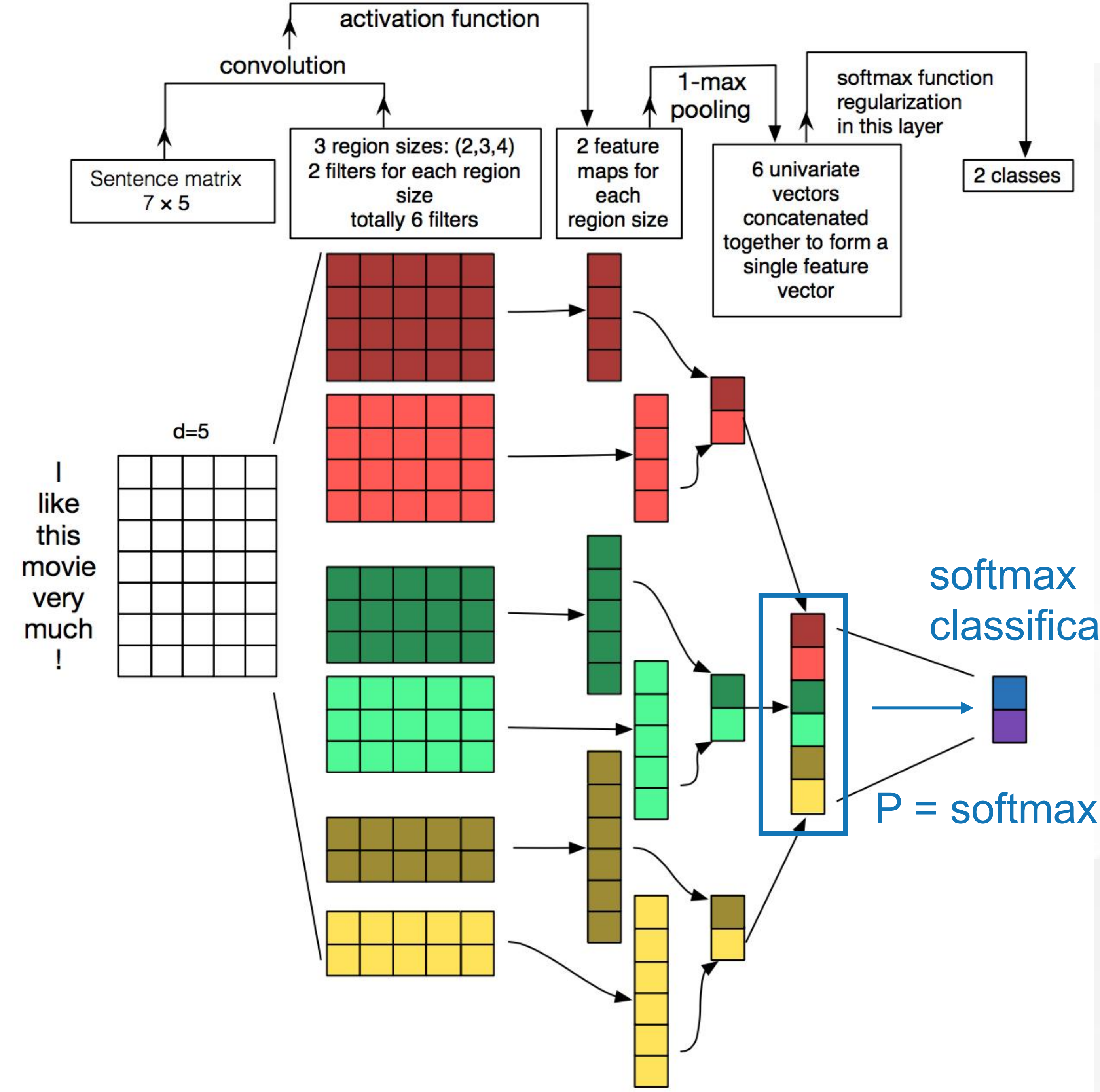
2.1 CNN

Pooling


activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

Feature vector
(num_filters * filter_size)
(or total_num_filters)

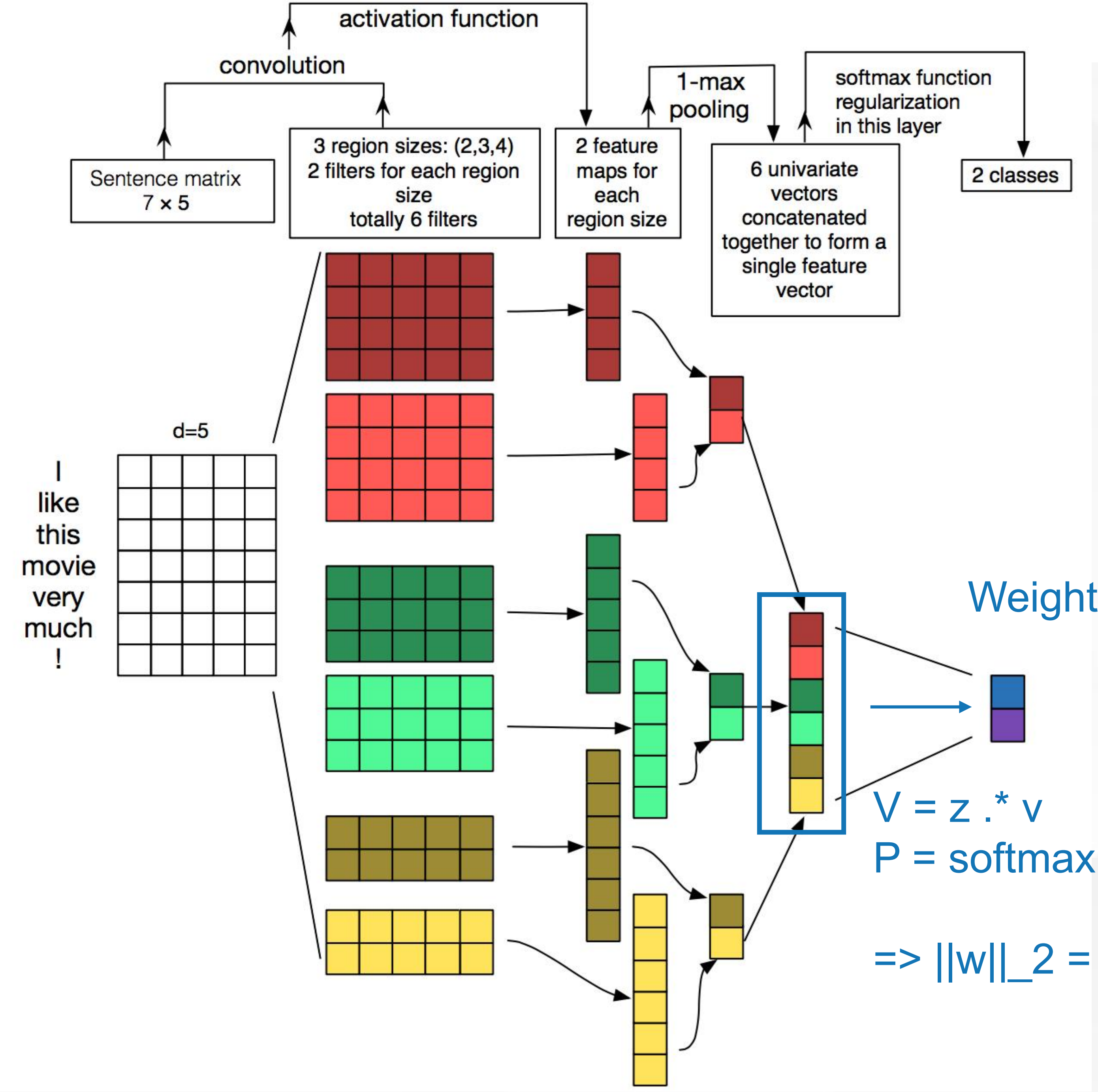
2.1 CNN


activation function
convolution
Sentence matrix
7 × 5
3 region sizes: (2,3,4)
2 filters for each region size
totally 6 filters
2 feature maps for each region size
1-max pooling
6 univariate vectors concatenated together to form a single feature vector
softmax function regularization in this layer
2 classes
d=5
I
like
this
movie
very
much
!

softmax classification

P = softmax(W·v+b)

2.1 CNN


activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

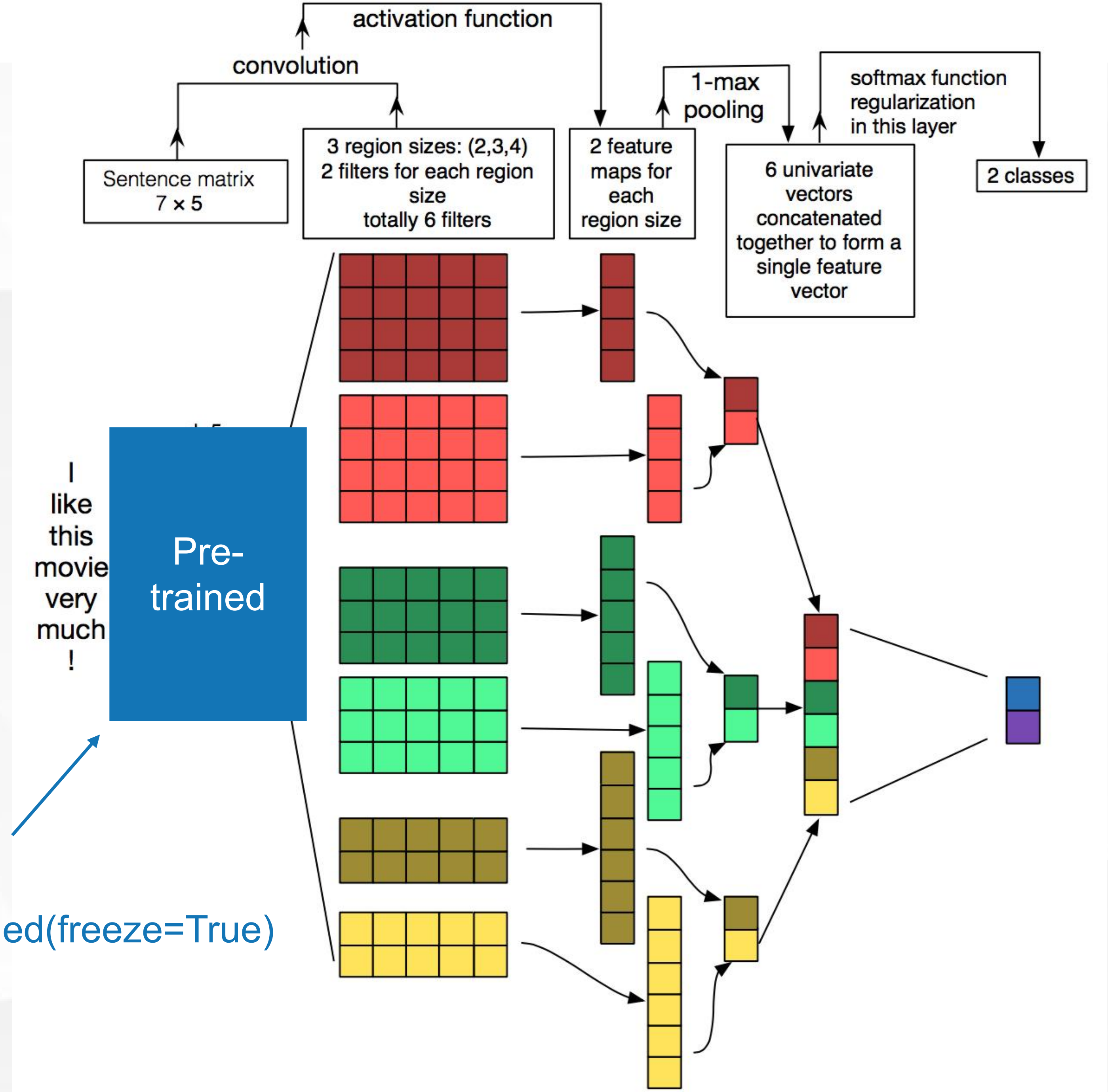
Weight Rescaling

$V = z .* v$

$P = \text{softmax}(W \cdot V + b)$

$\Rightarrow ||w||\_2 = s$

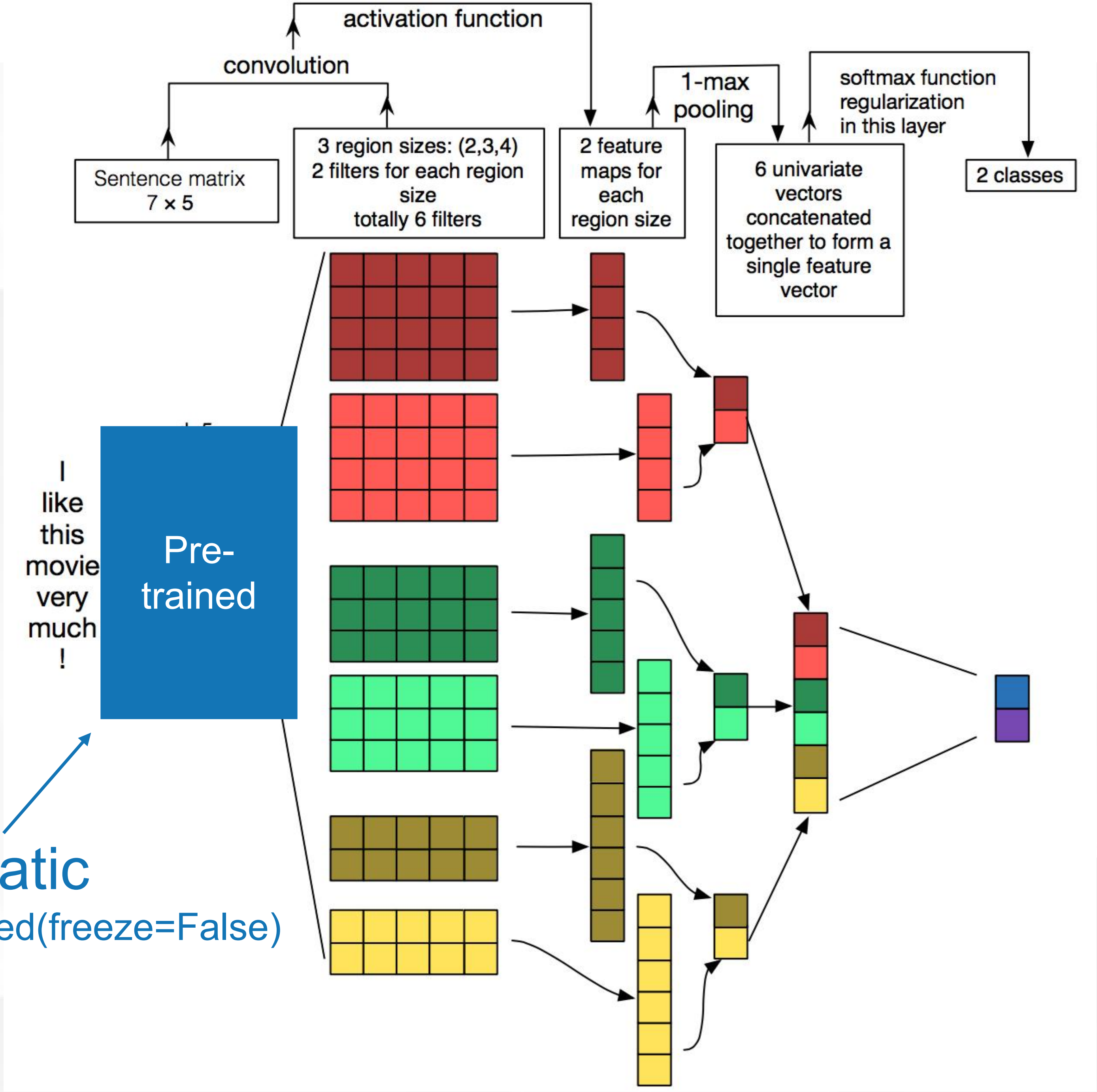
2.1 CNN


activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
I
like
this
movie
very
much
!
Pre-trained

1. static

from_pretrained(freeze=True)

2.1 CNN


activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
I
like
this
movie
very
much
!
Pre-trained

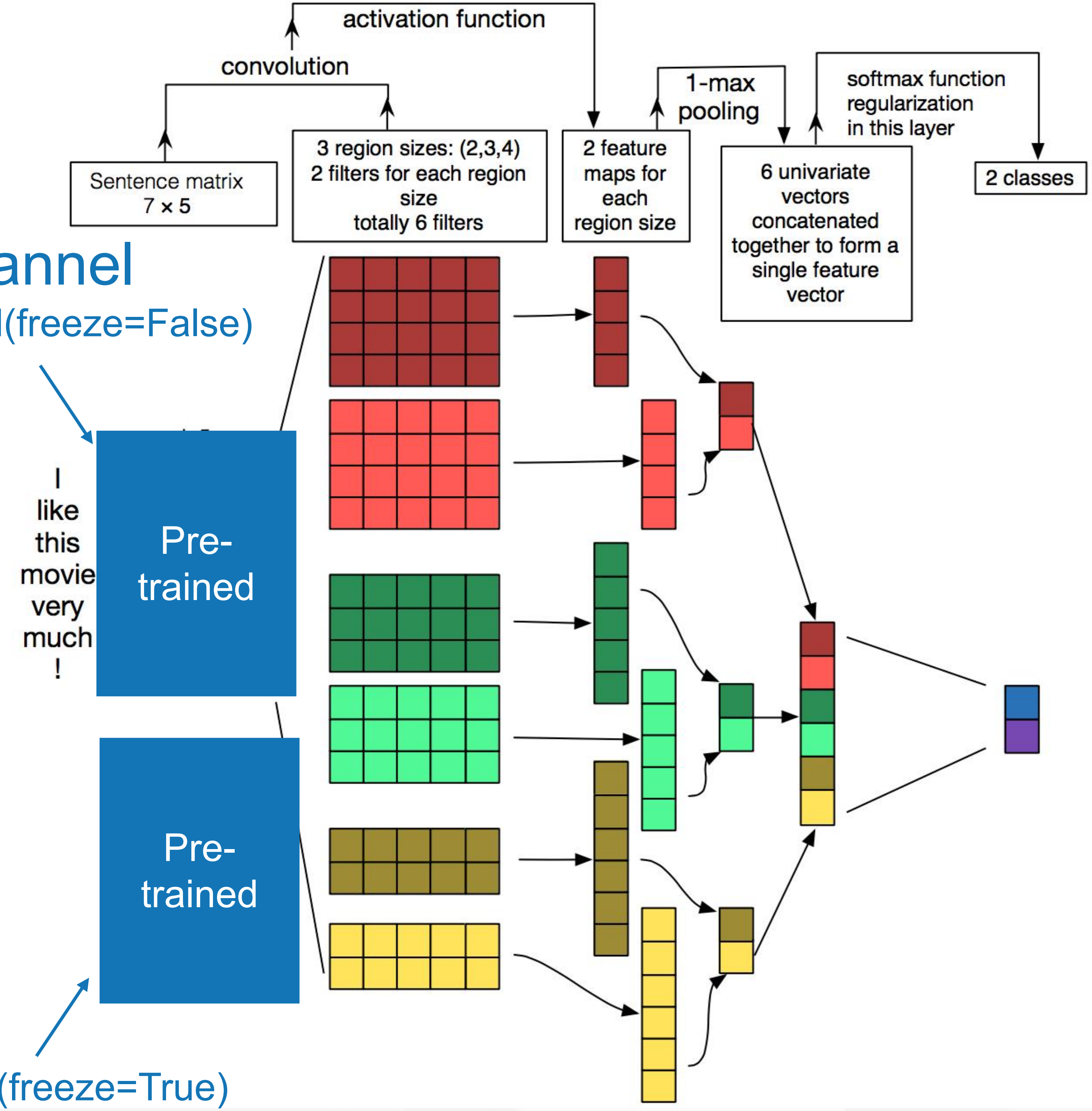
2. non-static

from_pretrained(freeze=False)

2.1 CNN

3. multichannel

from_pretrained(freeze=False)


activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
I
like
this
movie
very
much
!
Pre-trained
Pre-trained

from_pretrained(freeze=True)

Out

Outline

- Algorithm Introduction
  - CNN
  - RNN
- Implementation Details
- Experiment Result

2.2 RNN

output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \mathrm{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

tanh

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

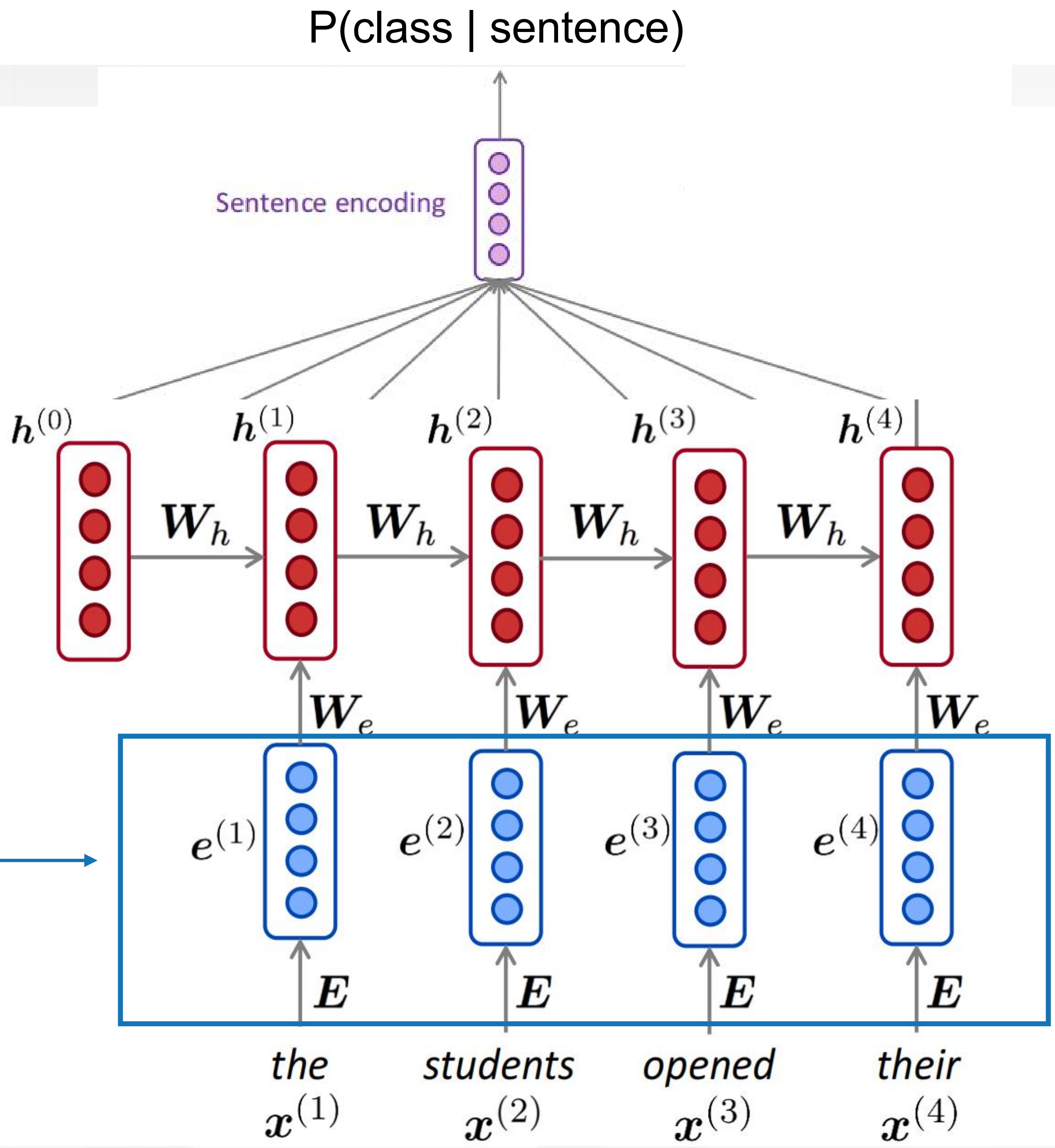
$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

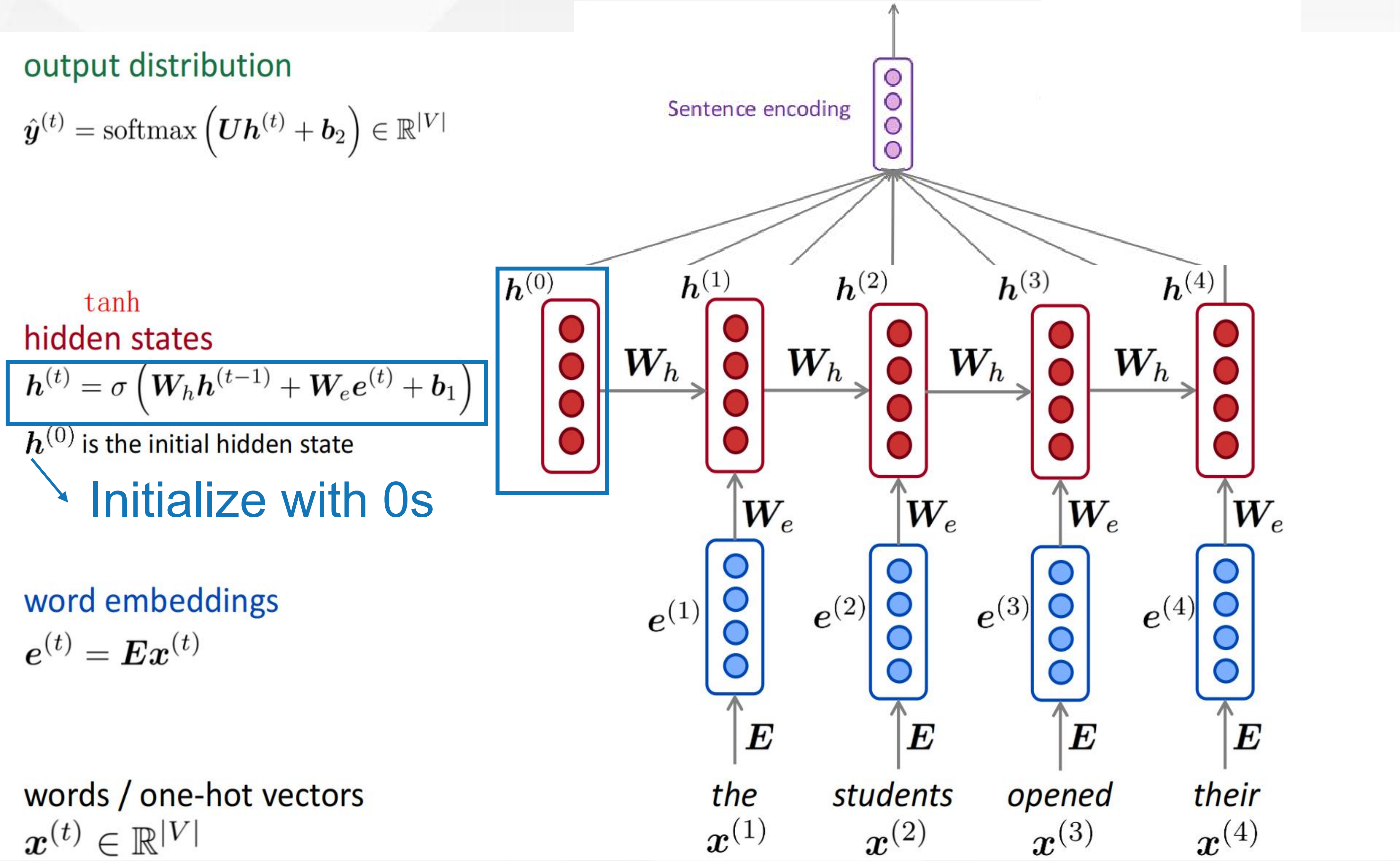
words / one-hot vectors

$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$



2.2 RNN

P(class | sentence)



output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \mathrm{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

tanh

hidden states

$$\boxed{\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)}$$

$\boldsymbol{h}^{(0)}$ is the initial hidden state

Initialize with 0s

word embeddings

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors

$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$

2.2 RNN

P(class | sentence)

output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \mathrm{softmax}\left(\boldsymbol{U}\boxed{\boldsymbol{h}^{(t)}} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

tanh

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e \boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

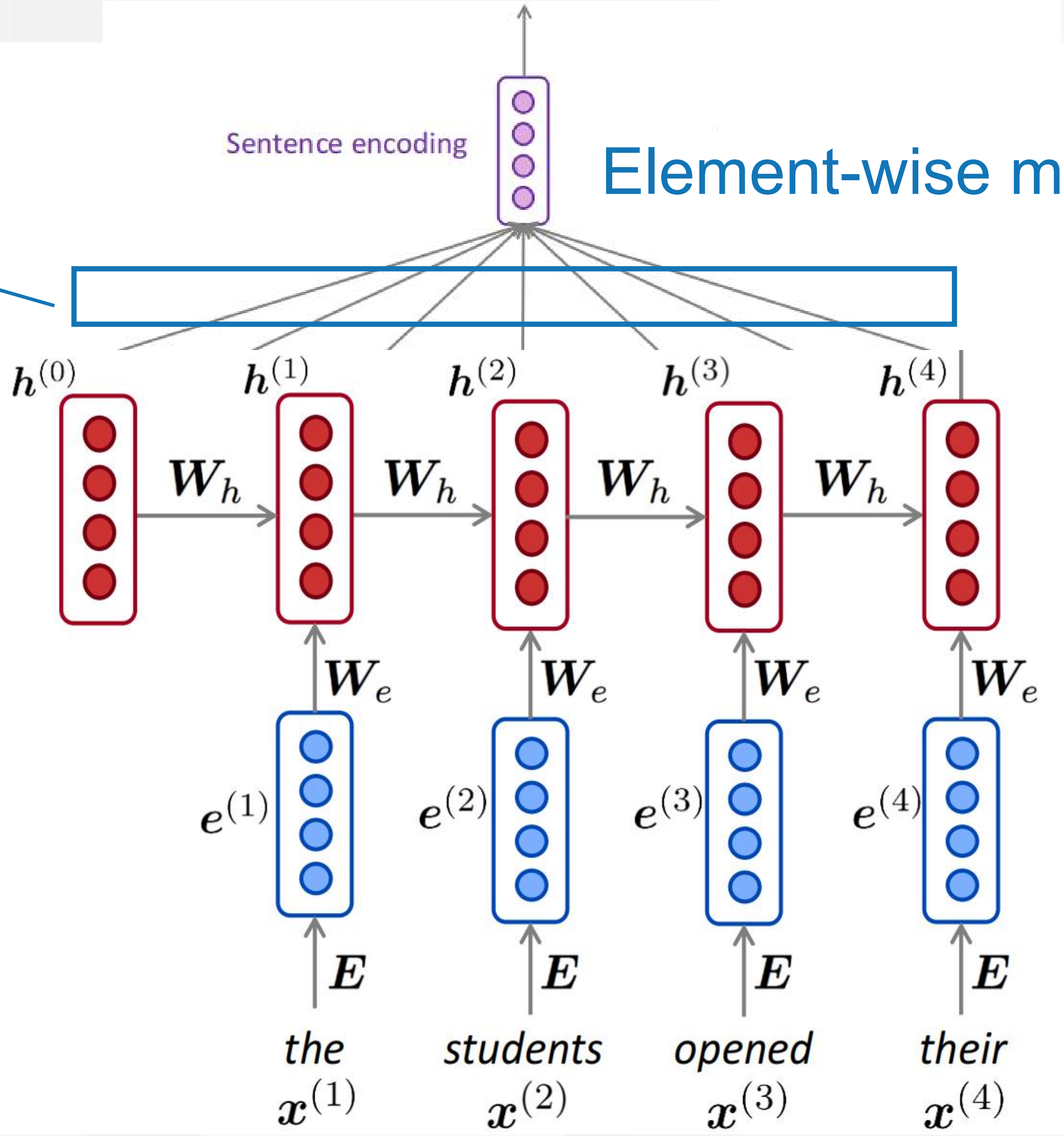
$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors

$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$



Element-wise max/mean

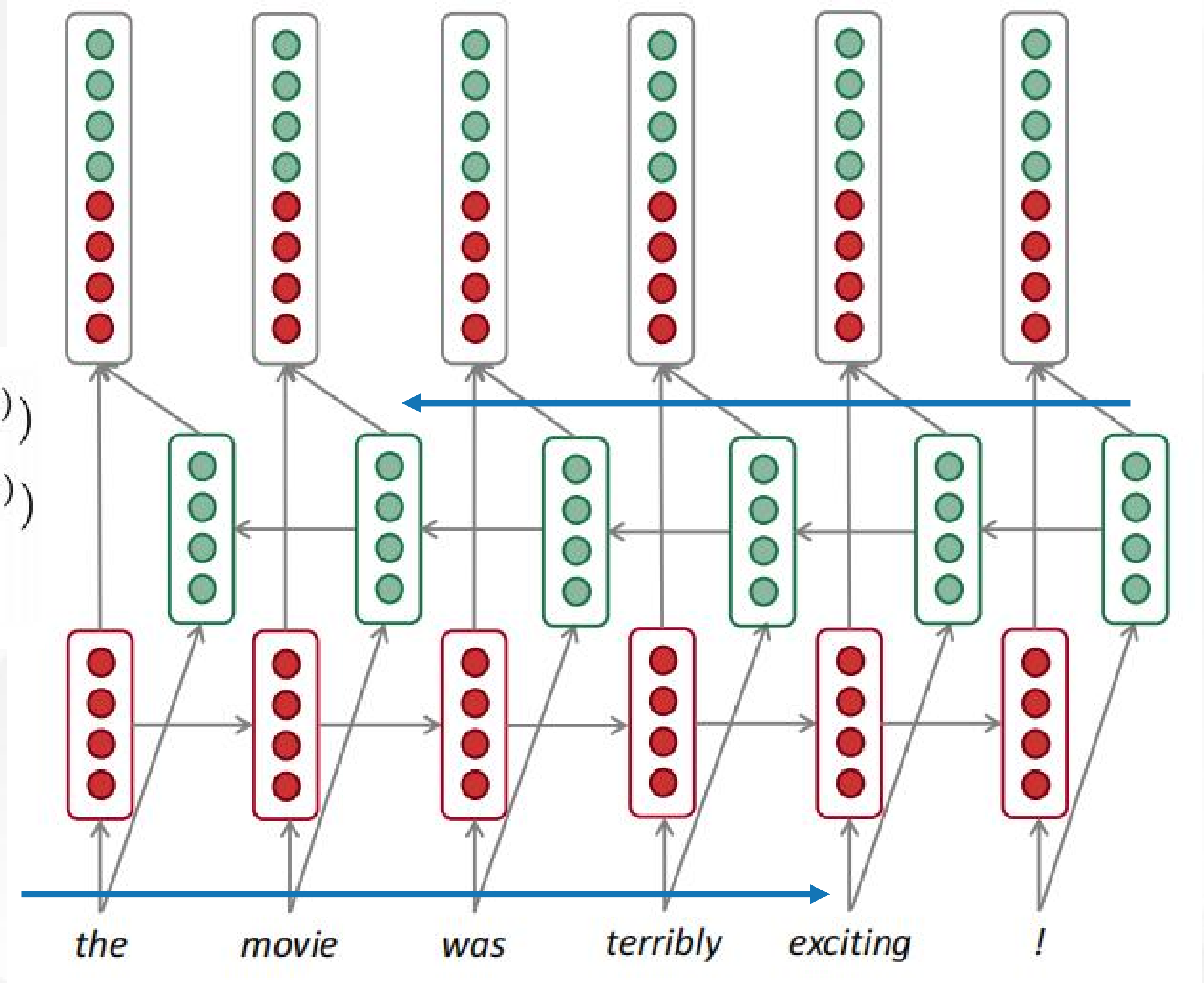
2.2 RNN: Bidirectional

Forward RNN $\overrightarrow{\boldsymbol{h}}^{(t)} = \mathrm{RNN}_{\mathrm{FW}}(\overrightarrow{\boldsymbol{h}}^{(t-1)}, \boldsymbol{x}^{(t)})$

Backward RNN $\overleftarrow{\boldsymbol{h}}^{(t)} = \mathrm{RNN}_{\mathrm{BW}}(\overleftarrow{\boldsymbol{h}}^{(t+1)}, \boldsymbol{x}^{(t)})$

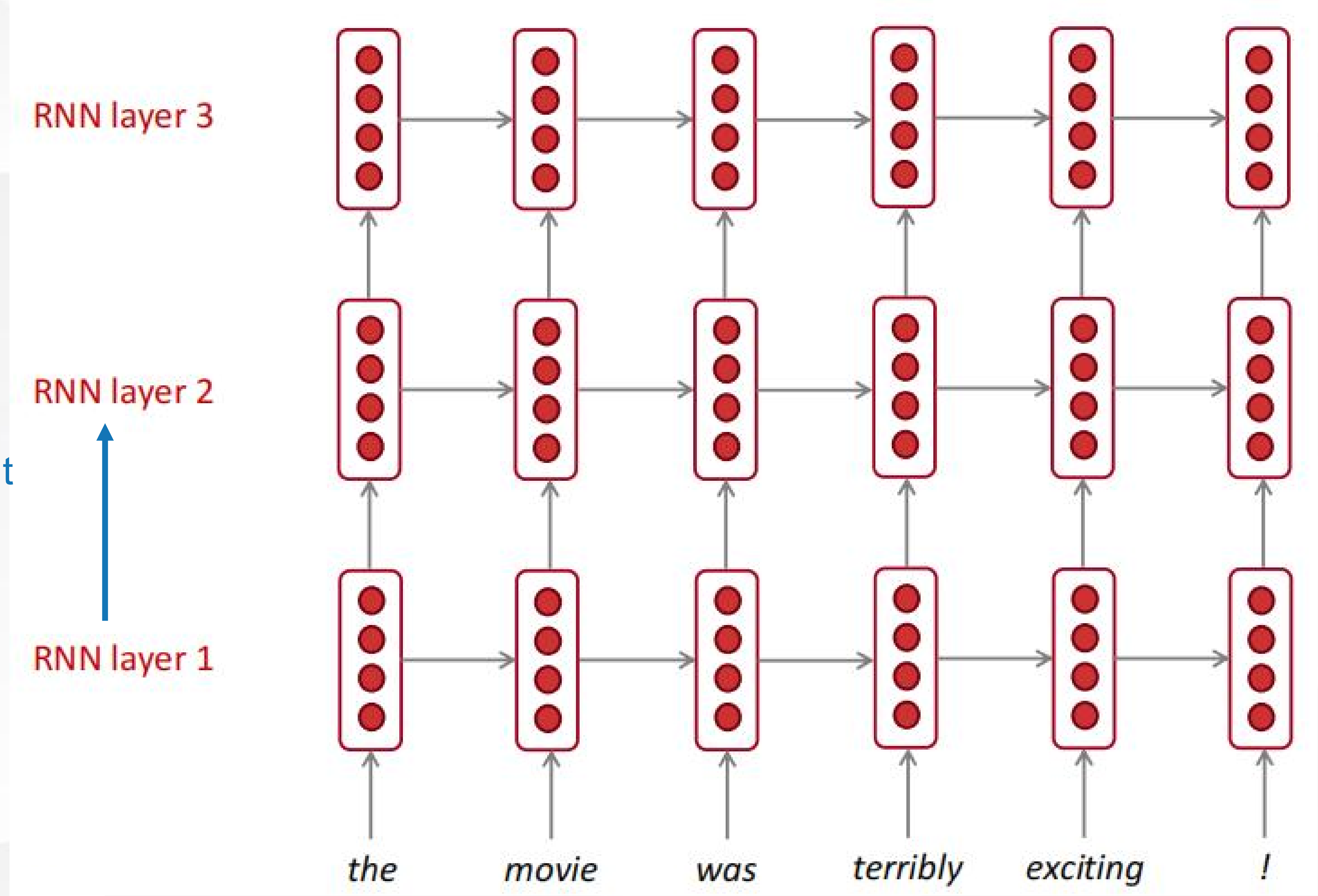
$$\boldsymbol{h}^{(t)} = [\overrightarrow{\boldsymbol{h}}^{(t)}; \overleftarrow{\boldsymbol{h}}^{(t)}]$$

Contain both left and right



2.2 RNN: Multi-layer

Layer 1 output
Becomes
Layer 2 input



Outline

- Algorithm Introduction
  - CNN
  - RNN
- Implementation Details
- Experiment Result

3.1 Implementation: Preprocessing

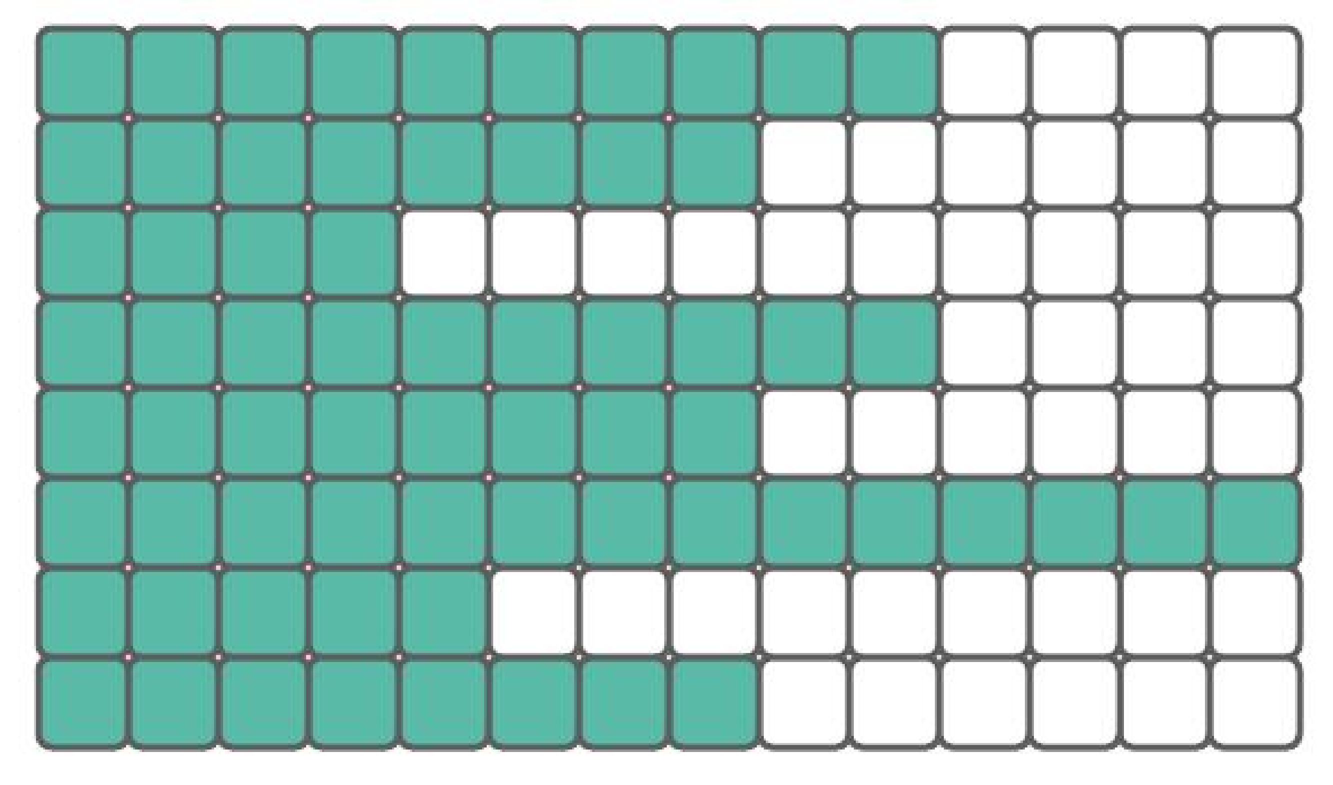
Remove tags, punctuations, stop words, short words…

Stemming?

3.2 Implementation: Padding

Why padding?

1. **For CNN: different sentence length**
2. **For mini-batch: assuming batches in the same length..**



Out

Outline

- Algorithm Introduction
  - CNN
  - RNN
- Implementation Details
- Experiment Result

4.1 Experiment Results

Bidirectional-RNN
(pre-trained with 300d vectors)

Model	Stem?	Test Acc
RNN_random	F	54.853
RNN_random	T	54.565
RNN_CBOW	F	58.013
RNN_GloVe	F	57.595

RNN (with GloVe)

Model	layers	Train Acc	Test Acc
RNN	1	65.967	57.654
Bi-RNN	1	67.556	58.100
Bi-RNN	2	67.797	58.717

4.1 Experiment Results

CNN (with 100 filters per window size)

Model	Sizes	Stem?	Test Acc
CNN_random	[3,4,5]	F	56.805
CNN_random	[3,4,5]	T	57.654
CNN_random	[2,3,4]	T	57.161
CNN_random	[4,5,6]	T	57.237
CNN_CBOW	[3,4,5]	F	57.755
CNN_GloVe	[3,4,5]	F	56.591

Thanks