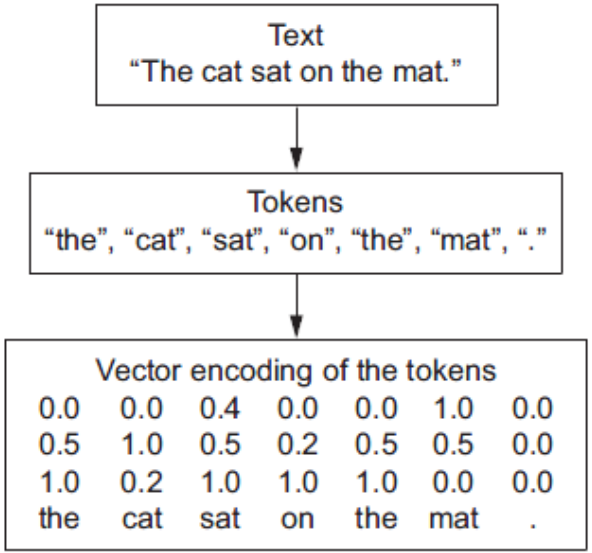
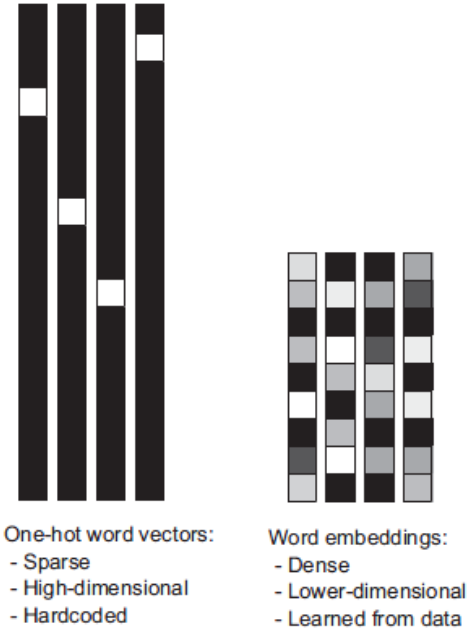
TF2-09.

RNNs

Dong Kook Kim

Text Processing
- One-hot encoding and Embedding





Word Embedding

- Approaches

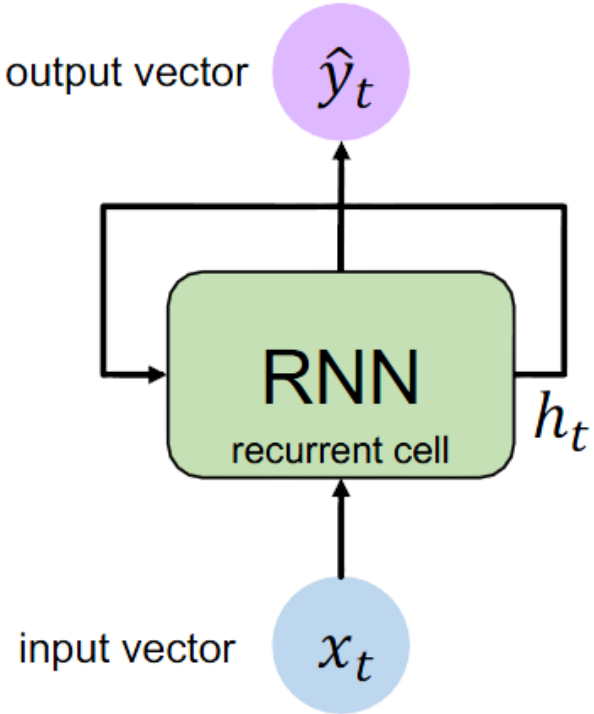
  1. Lear word embeddings jointly with the main task
     (such as document classification or sentiment prediction)

  2. Load into your model embeddings from pretrained word embeddings
     - Word2vec, GloVe(global Vectors for Word Representation)

- Keras
  - Embedding layer

```
model = Sequential()
model.add(Embedding(1000, 64, input_length=10))
```

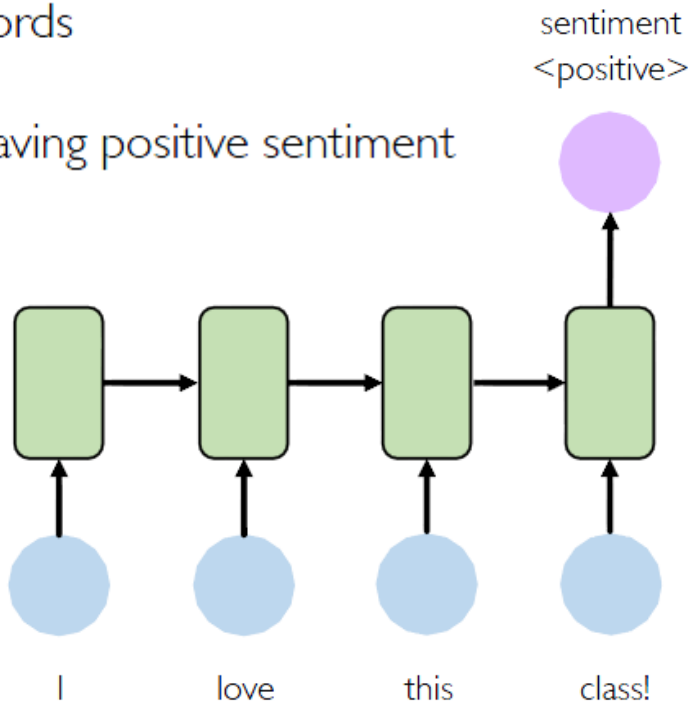
Simple RNN



IMDB : Sentiment classification

Input: sequence of words

Output: probability of having positive sentiment



Exercise 09-1.

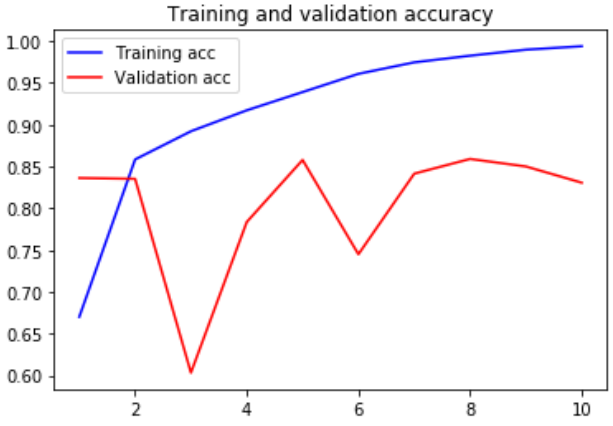
tf2-09-1-imdb_simple_rnn.py

Simple RNN

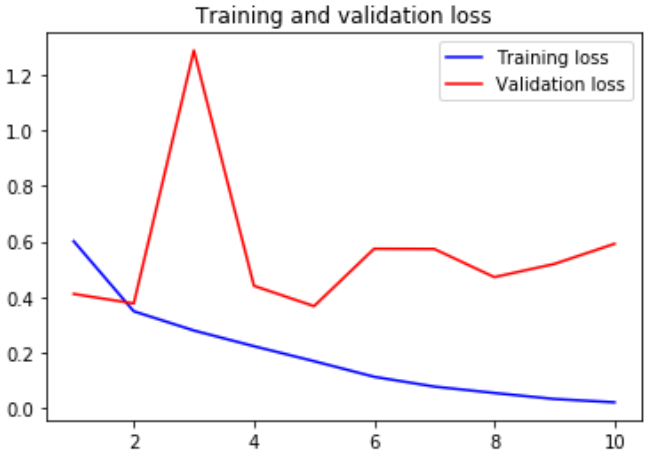
```
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))
```

Results

Training/Validation Loss



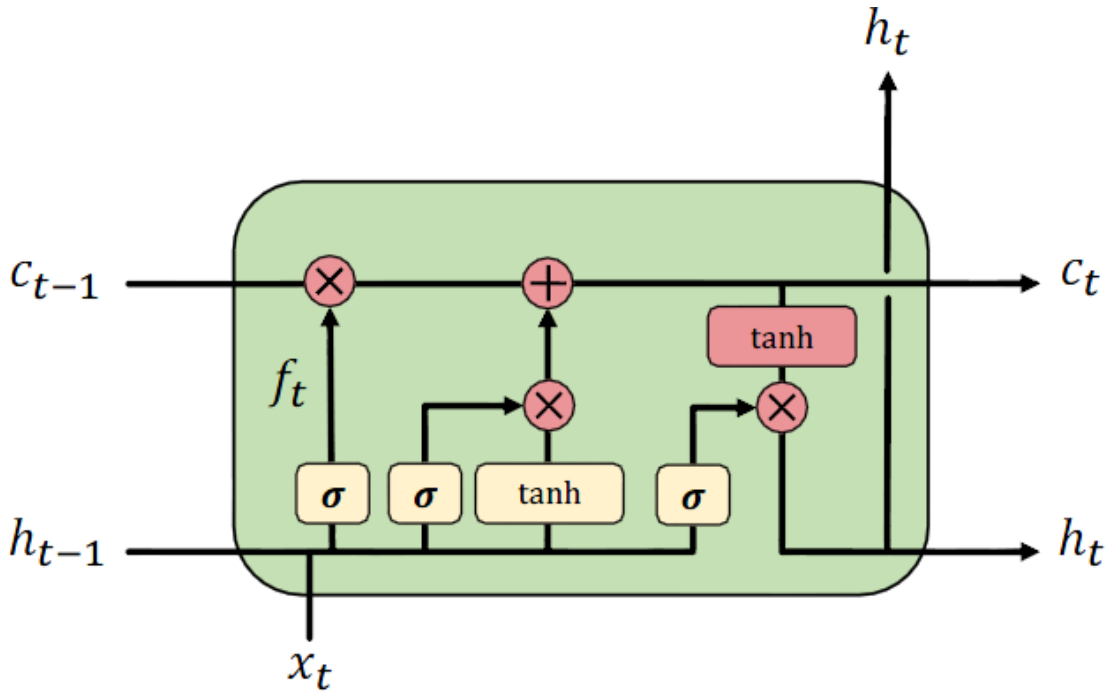
Training/Validataion Accuracy



Test Accuracy

83.06 %

LSTM

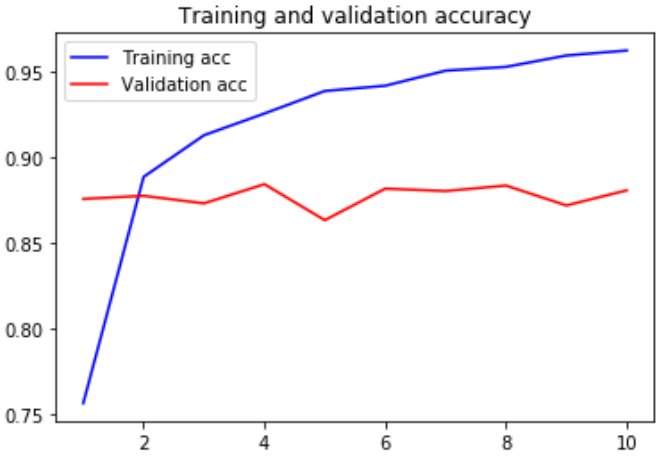


Exercise 09-2.

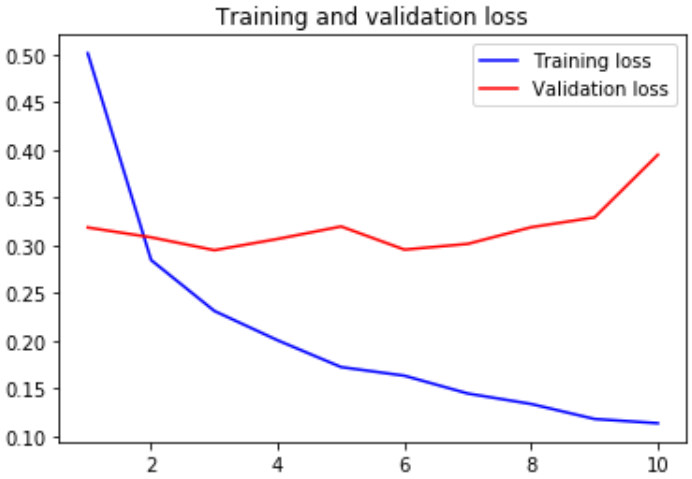
tf2-09-2-imdb_lstm.py

Results

Training/Validation Loss



Training/Validataion Accuracy



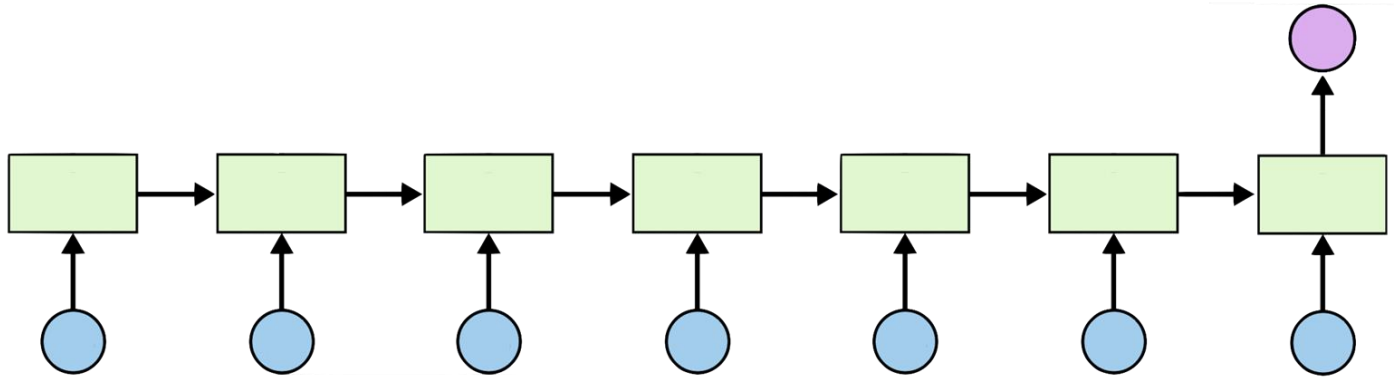
Test Accuracy

86.34 %

RNN with time series data (stock)

Open	High	Low	Volume	Close
828.659973	833.450012	828.349976	1247700	**831.659973**
823.02002	828.070007	821.655029	1597800	**828.070007**
819.929993	824.400024	818.97998	1281700	**824.159973**
819.359985	823	818.469971	1304000	**818.97998**
819	823	816	1053600	**820.450012**
816	820.958984	815.48999	1198100	**819.23999**
811.700012	815.25	809.780029	1129100	**813.669983**
809.51001	810.659973	804.539978	989700	**809.559998**
807	811.840027	803.190002	1155300	**808.380005**

'data-02-stock_daily.csv'



Open	High	Low	Volume	Close
828.659973	833.450012	828.349976	1247700	**831.659973**
823.02002	828.070007	821.655029	1597800	**828.070007**
819.929993	824.400024	818.97998	1281700	**824.159973**
819.359985	823	818.469971	1304000	**818.97998**
819	823	816	1053600	**820.450012**
816	820.958984	815.48999	1198100	**819.23999**
811.700012	815.25	809.780029	1129100	**813.669983**
809.51001	810.659973	804.539978	989700	**?**
807	811.840027	803.190002	1155300	**?**

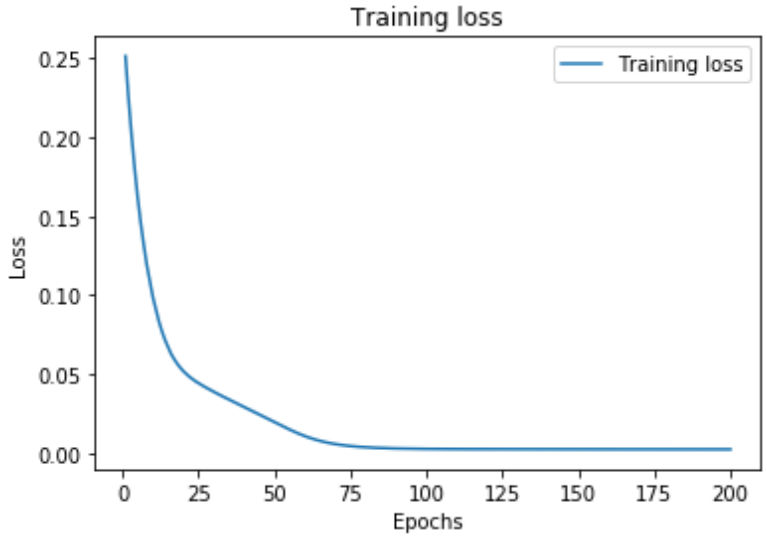
Many to one

Exercise 09-3.

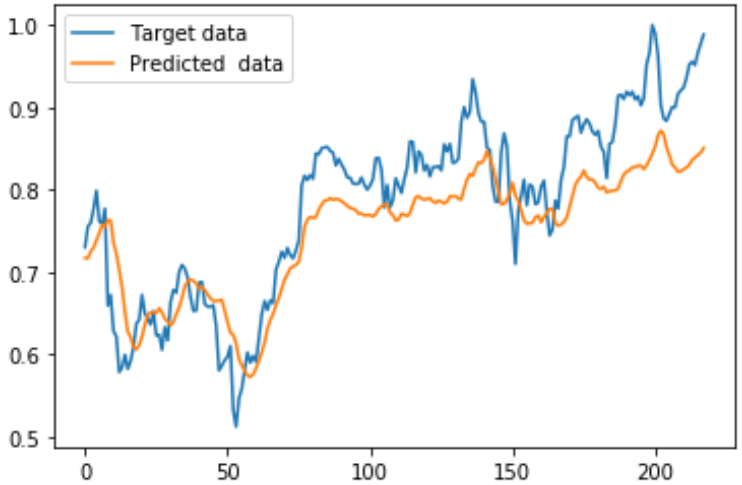
tf2-09-3-stock_gru.py

Results

Training Loss



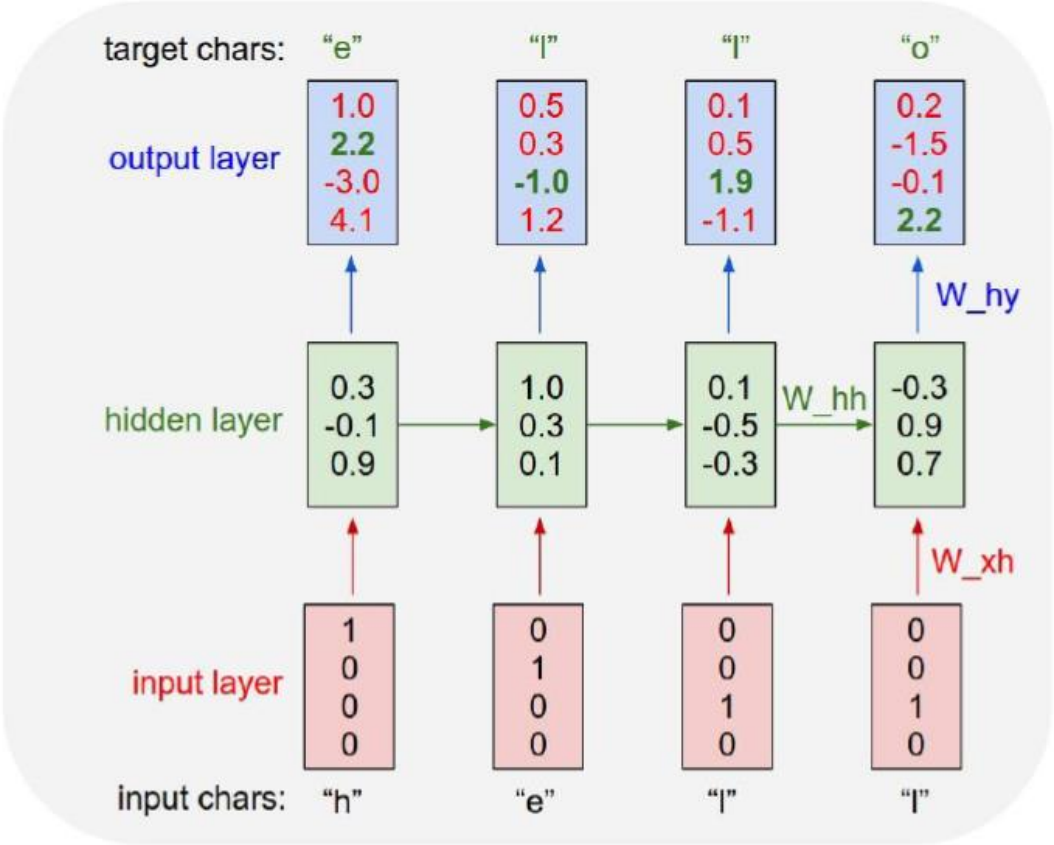
Prediction Results



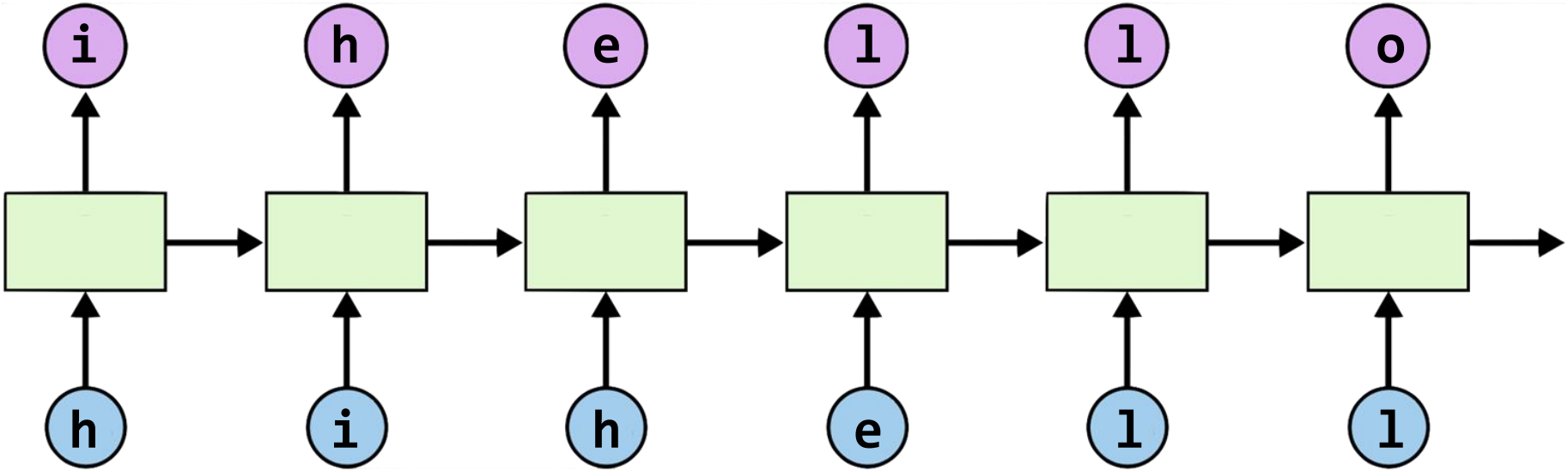
Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



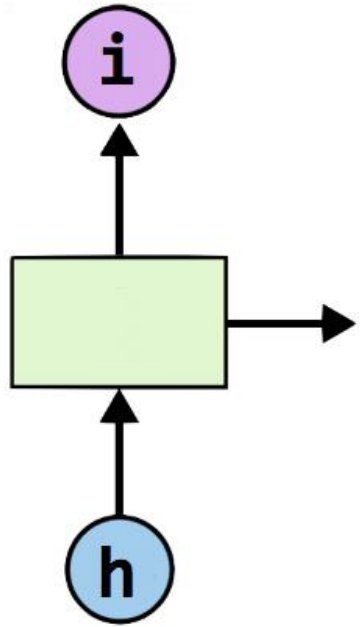
Teach RNN ‘hihello’



- text: ‘hihello’
- unique chars (vocabulary, voc):
  h, i, e, l, o
- voc index:
  h:0, i:1, e:2, l:3, o:4

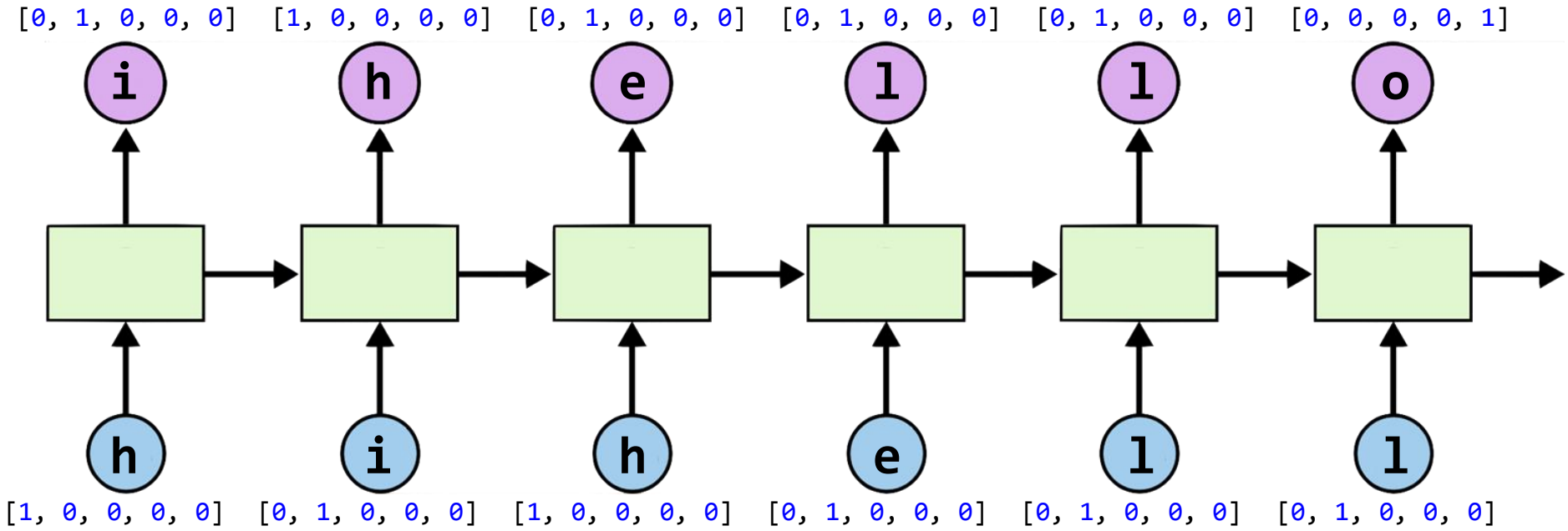
One-hot encoding

```
[1, 0, 0, 0, 0],   # h 0
[0, 1, 0, 0, 0],   # i 1
[0, 0, 1, 0, 0],   # e 2
[0, 0, 1, 0, 0],   # l 3
[0, 0, 0, 0, 1],   # o 4
```



Teach RNN ‘hihello’

```
[1, 0, 0, 0, 0],   # h 0
[0, 1, 0, 0, 0],   # i 1
[0, 0, 1, 0, 0],   # e 2
[0, 0, 0, 1, 0],   # l 3
[0, 0, 0, 0, 1],   # o 4
```

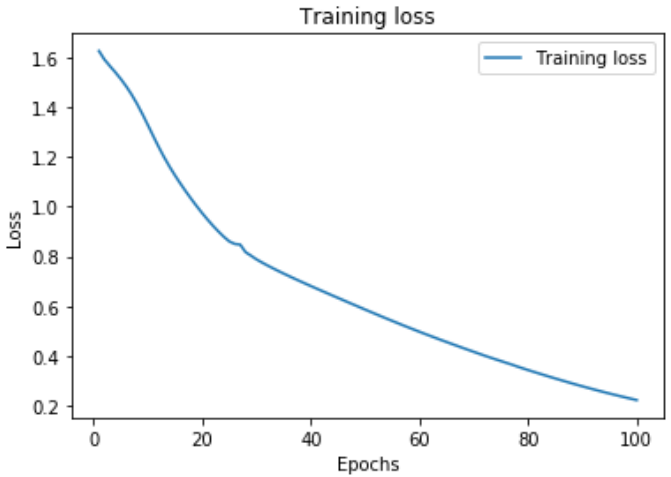


Exercise 09-4.

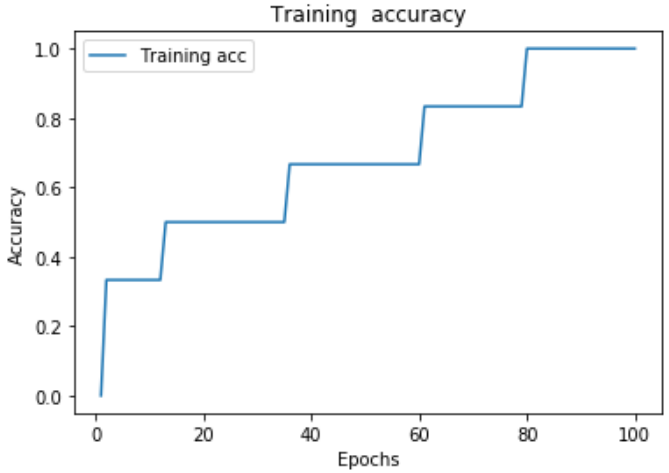
tf2-09-4-hihello_lstm.py

Results

Training Loss



Training Accuracy



Results

[2 1 2 3 0 0] hihell
[1 2 3 0 0 4] ihello