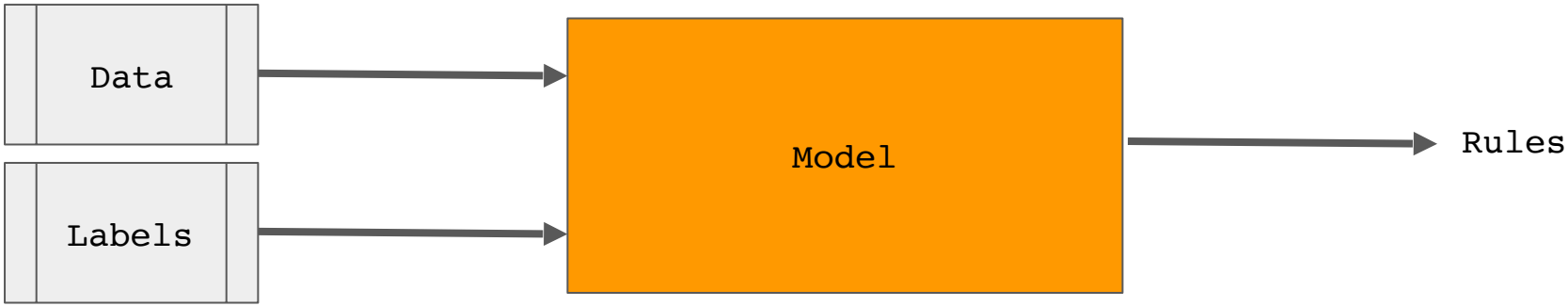
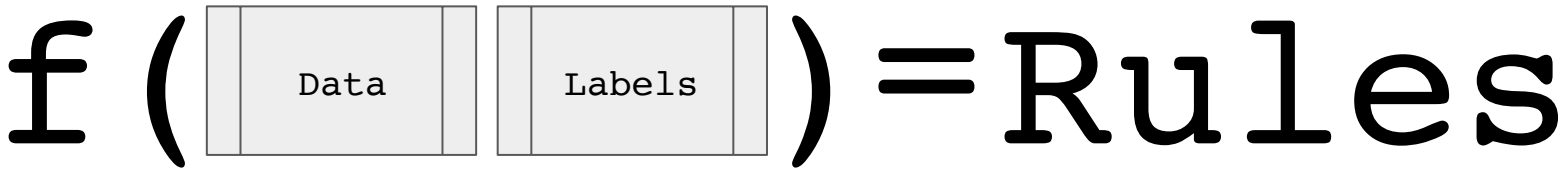
Copyright Notice

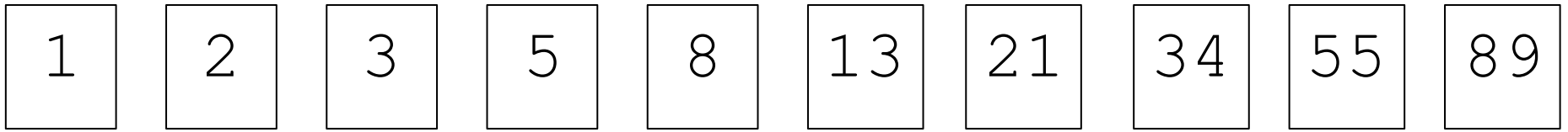
These slides are distributed under the Creative Commons License.

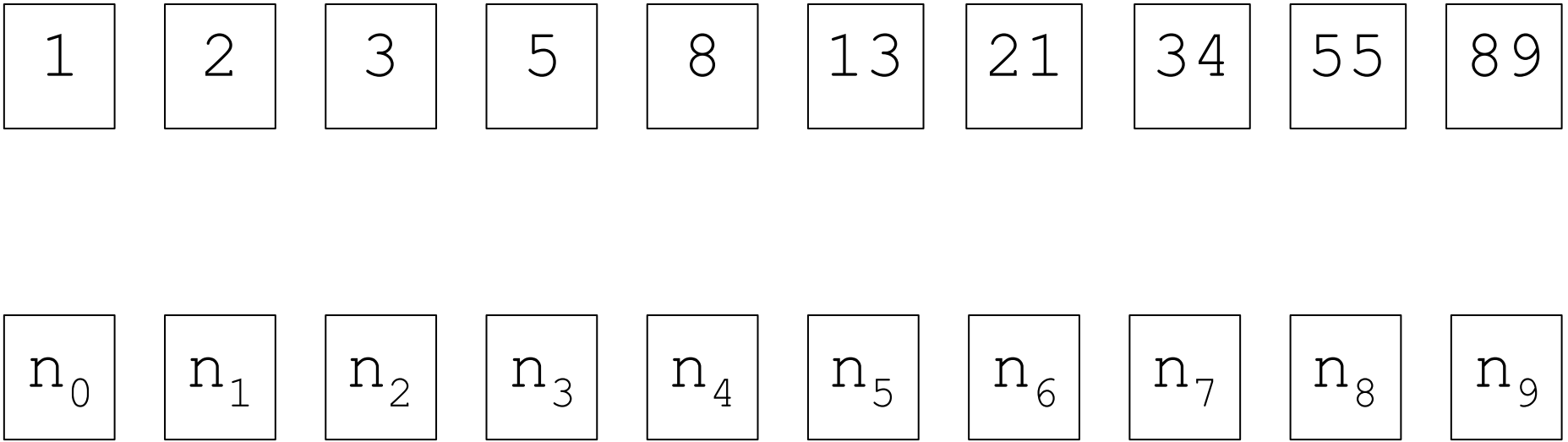
DeepLearning.AI makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite DeepLearning.AI as the source of the slides.

For the rest of the details of the license, see https://creativecommons.org/licenses/by-sa/2.0/legalcode


Data
Labels
Model
Rules

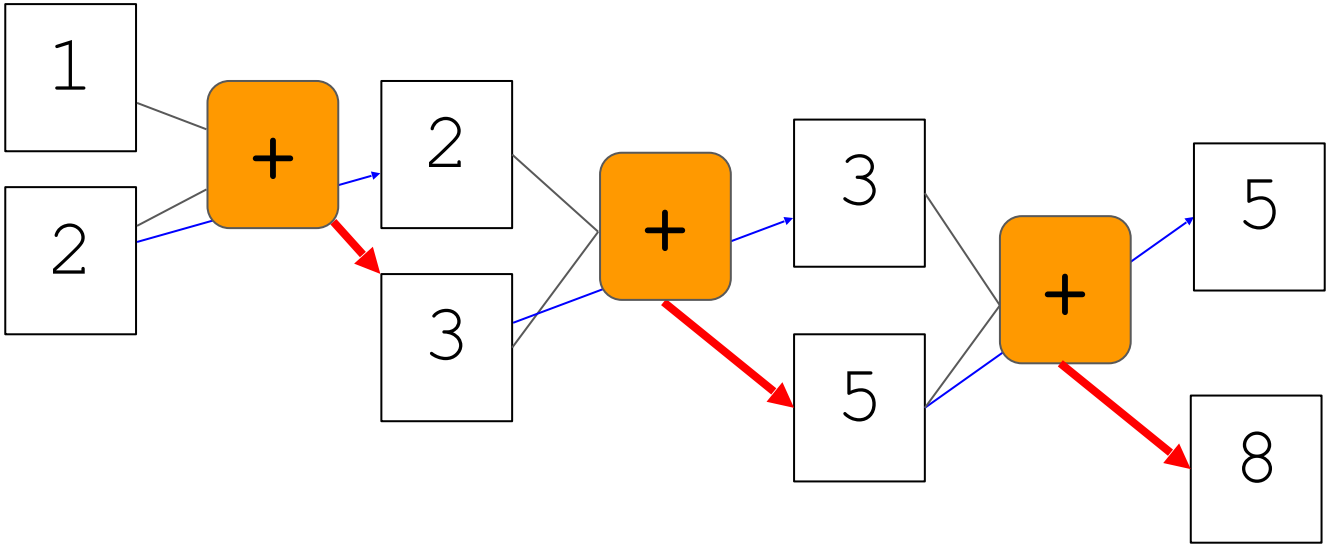

f(
Data
Labels
)=Rules

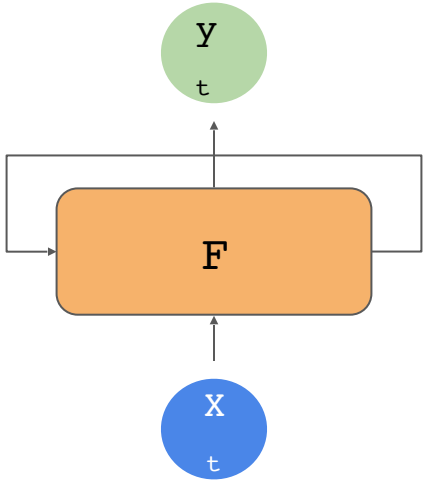

1
2
3
5
8
13
21
34
55
89

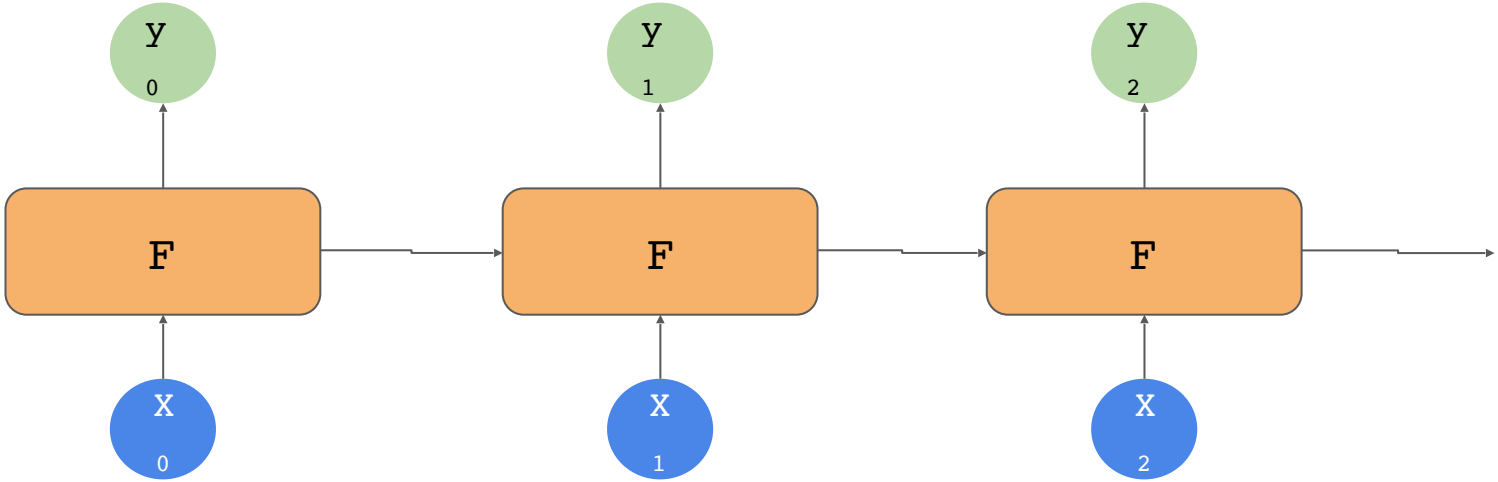

1
2
3
5
8
13
21
34
55
89
n_0
n_1
n_2
n_3
n_4
n_5
n_6
n_7
n_8
n_9

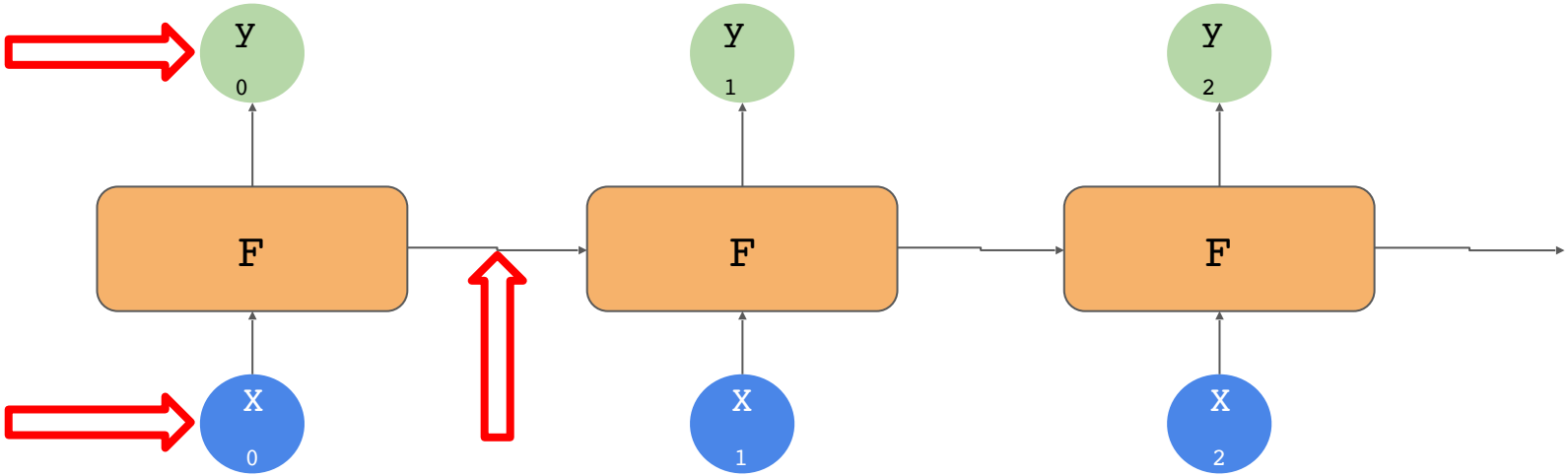
1	2	3	5	8	13	21	34	55	89
n_0	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9

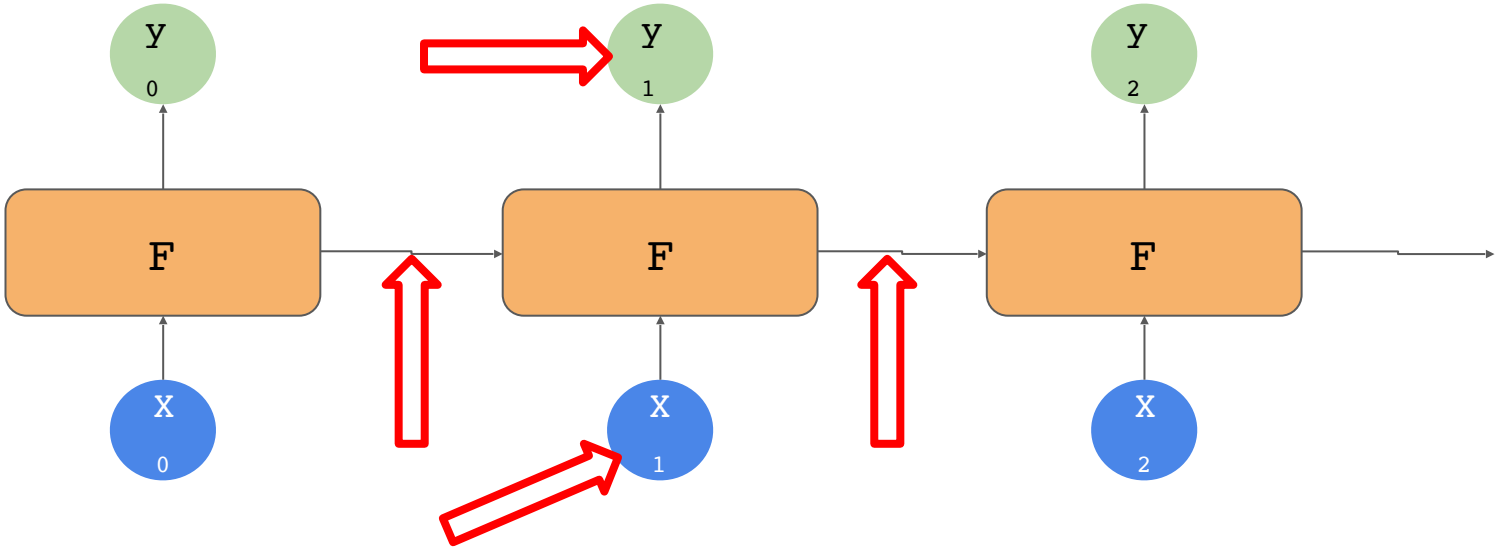
$$n_x = n_{x-1} + n_{x-2}$$

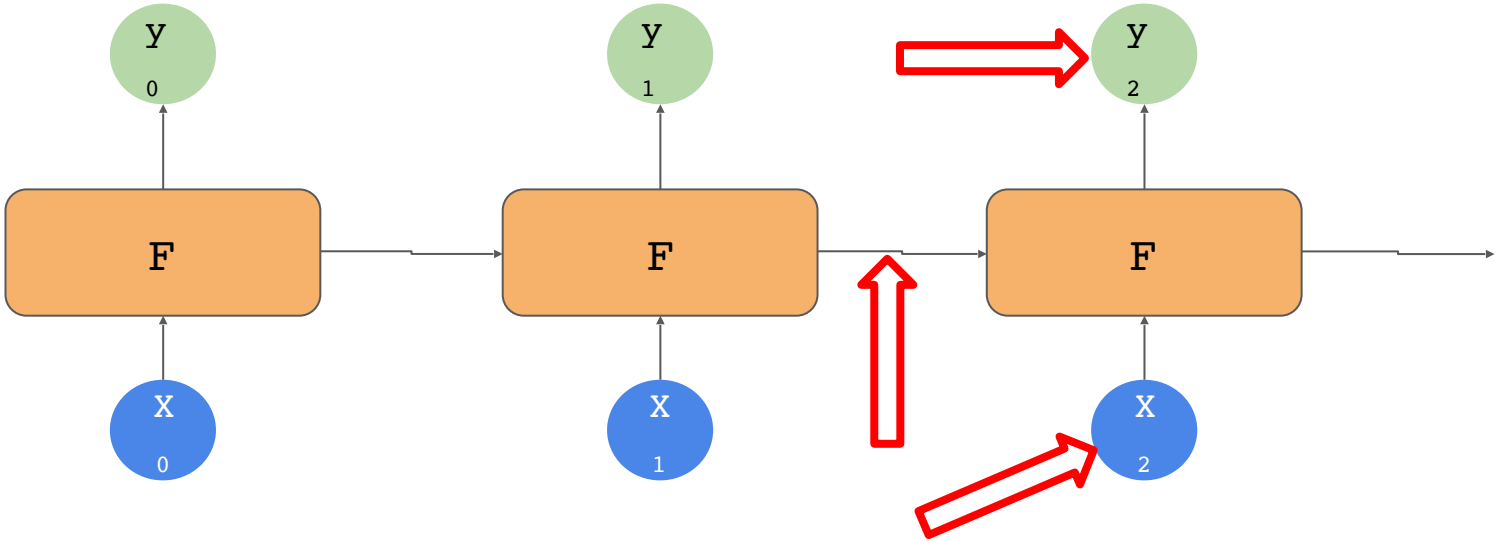

1
2
+
2
3
+
3
5
+
5
8

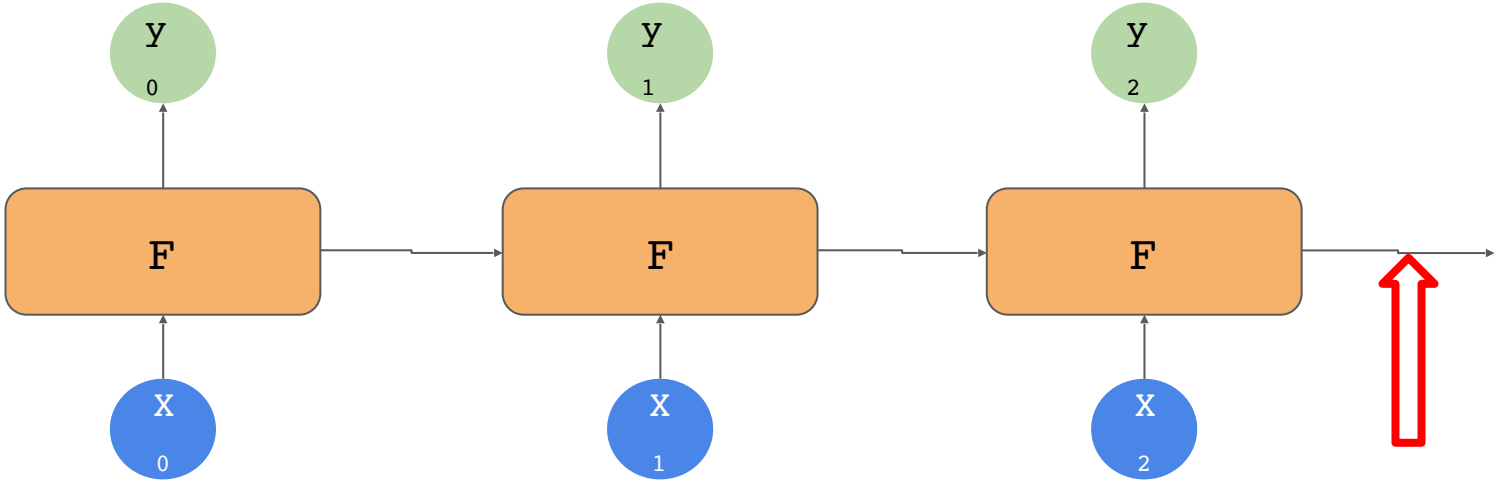

y
t
F
x
t


y
0
y
1
y
2
F
F
F
x
0
x
1
x
2


y
0
y
1
y
2
F
F
F
x
0
x
1
x
2

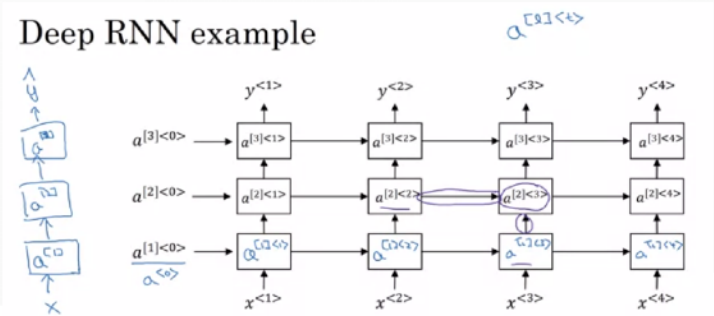

y
0
y
1
y
2
F
F
F
x
0
x
1
x
2


y
0
y
1
y
2
F
F
F
x
0
x
1
x
2


y
0
y
1
y
2
F
F
F
x
0
x
1
x
2

https://www.coursera.org/lecture/nlp-sequence-models/deep-rnns-ehs0S

Deep RNNs



From the course by deeplearning.ai

Sequence Models

13393 ratings

Try the Course for Free

This Course Video Transcript

deeplearning.ai

Sequence Models

13393 ratings

Course 5 of 5 in the Specialization Deep Learning

This course will teach you how to build models for natural language, audio, and other sequence data. Thanks to deep learning, sequence algorithms are working far better than just two years ago, and this is enabling numerous exciting applications in speech recognition, music synthesis, chatbots, machine translation, natural language understanding, and many others. You will: - Understand how to build and train Recurrent Neural Networks (RNNs), and commonly-

More

Today has a beautiful blue <...>

Today has a beautiful blue <...>

Today has a beautiful blue sky

Today has a beautiful blue <...>

Today has a beautiful blue sky

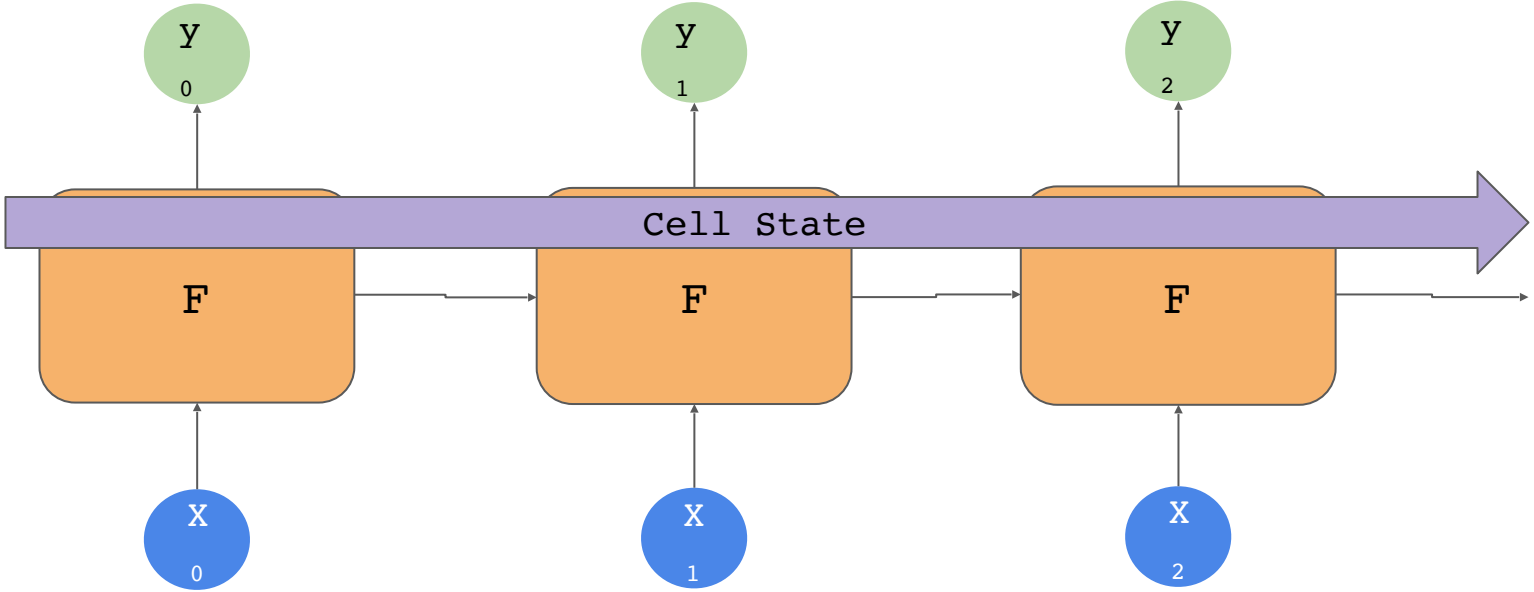
I lived in Ireland, so at school they made me learn how to speak <...>

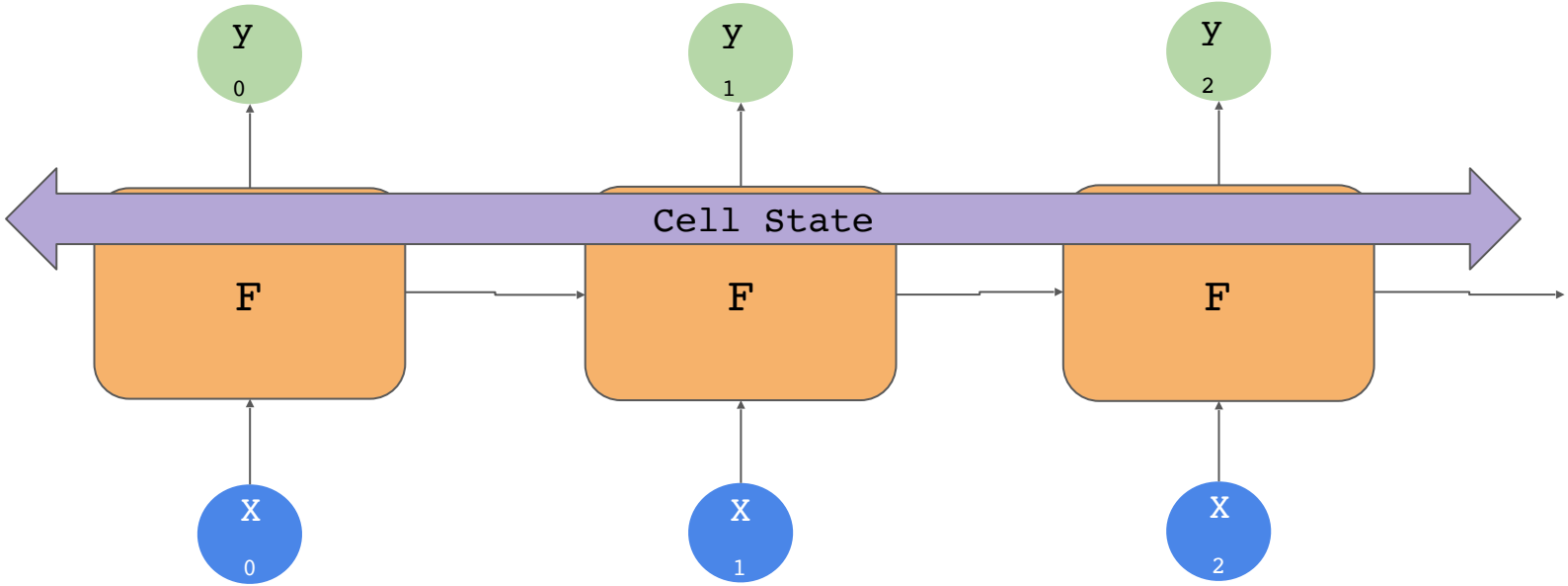
I lived in Ireland, so at school they made me learn how to speak <...>

I lived in Ireland, so at school they made me learn how to speak Gaelic

I lived in Ireland, so at school they made me learn how to speak <...>

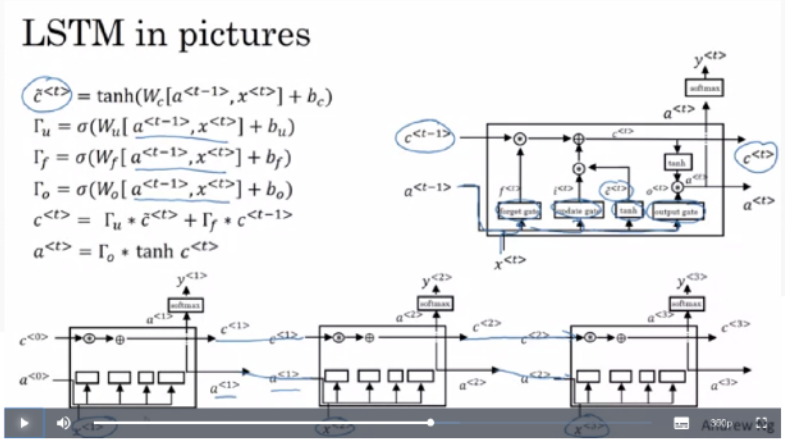
I lived in Ireland so at school they made me learn how to speak Gaelic


y
0
y
1
y
2
Cell State
F
F
F
X
0
X
1
X
2


y
0
y
1
y
2
Cell State
F
F
F
X
0
X
1
X
2

https://www.coursera.org/lecture/nlp-sequence-models/long-short-term-memory-lstm-KXoay

Long Short Term Memory (LSTM)



From the course by deeplearning.ai

Sequence Models

4.8 ★★★★★ 13393 ratings

Try the Course for Free

This Course | Video Transcript

deeplearning.ai

Sequence Models

4.8 ★★★★★ 13393 ratings

Course 5 of 5 in the Specialization Deep Learning

This course will teach you how to build models for natural language, audio, and other sequence data. Thanks to deep learning, sequence algorithms are working far better than just two years ago, and this is enabling numerous exciting applications in speech recognition, music synthesis, chatbots, machine translation, natural language understanding, and many others. You will: - Understand how to build and train Recurrent Neural Networks (RNNs), and commonly-

More

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(tokenizer.vocab_size, 64),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(tokenizer.vocab_size, 64),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(tokenizer.vocab_size, 64),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, None, 64)          523840
_________________________________________________________________
bidirectional_1 (Bidirection (None, 128)               66048
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 65
=================================================================
Total params: 598,209
Trainable params: 598,209
Non-trainable params: 0
```

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, None, 64)	523840
bidirectional_1 (Bidirection	(None, 128)	66048
dense_4 (Dense)	(None, 64)	8256
dense_5 (Dense)	(None, 1)	65

Total params: 598,209
Trainable params: 598,209
Non-trainable params: 0

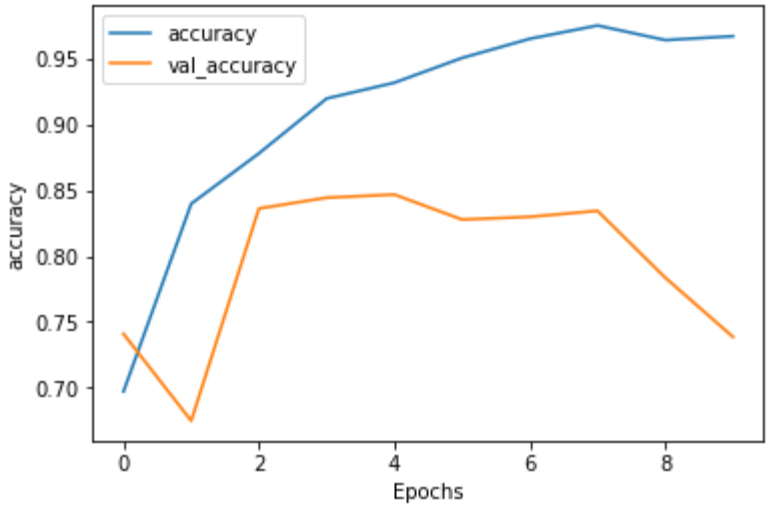
```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(tokenizer.vocab_size, 64),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(tokenizer.vocab_size, 64),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

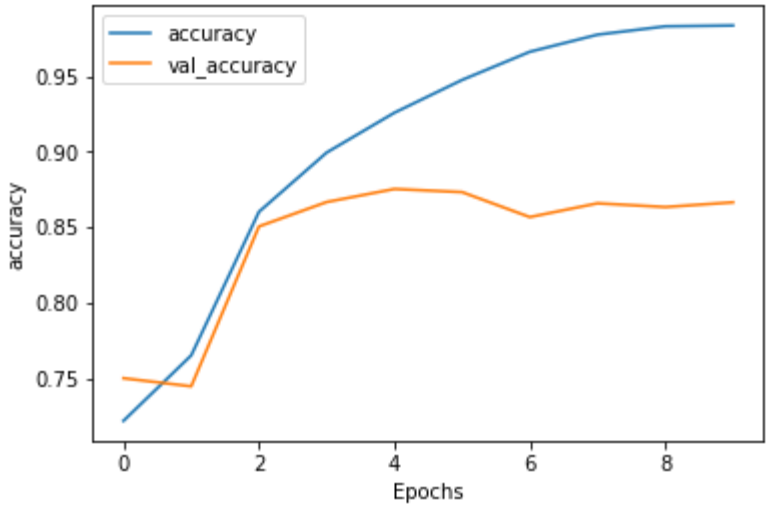
Layer (type)	Output Shape	Param #
embedding_3 (Embedding)	(None, None, 64)	523840
bidirectional_2 (Bidirection	(None, None, 128)	66048
bidirectional_3 (Bidirection	(None, 64)	41216
dense_6 (Dense)	(None, 64)	4160
dense_7 (Dense)	(None, 1)	65

Total params: 635,329
Trainable params: 635,329
Non-trainable params: 0

10 Epochs : Accuracy Measurement

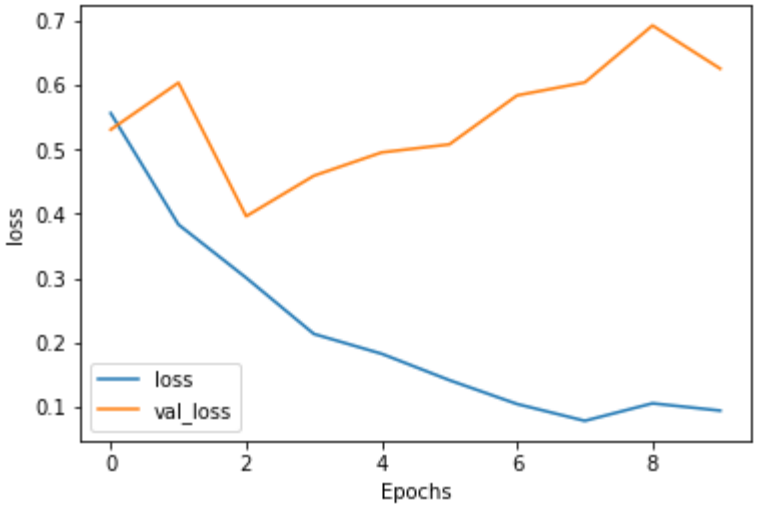


1 Layer
LSTM

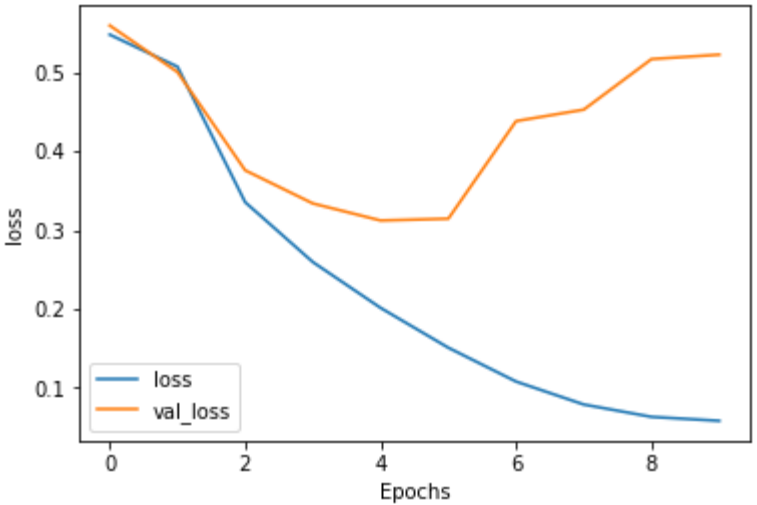


2 Layer
LSTM

10 Epochs : Loss Measurement

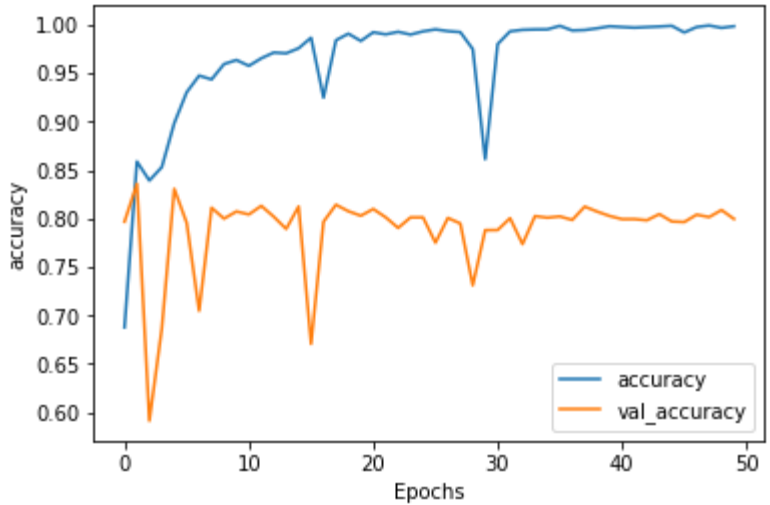


1 Layer
LSTM

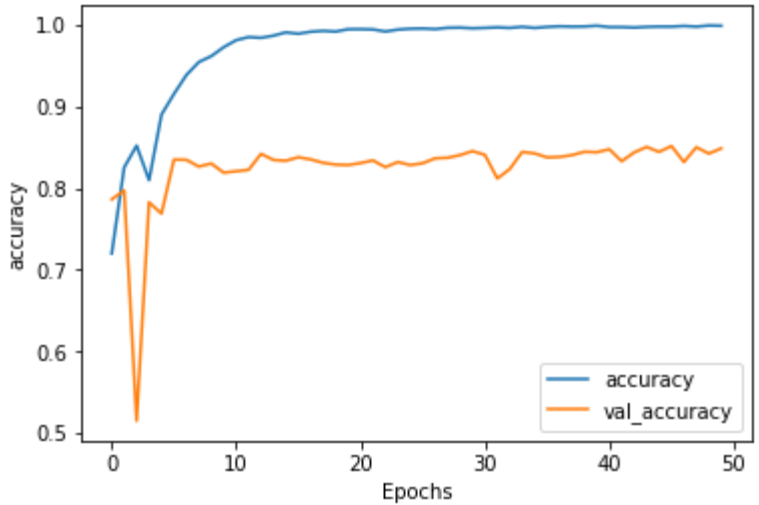


2 Layer
LSTM

50 Epochs : Accuracy Measurement

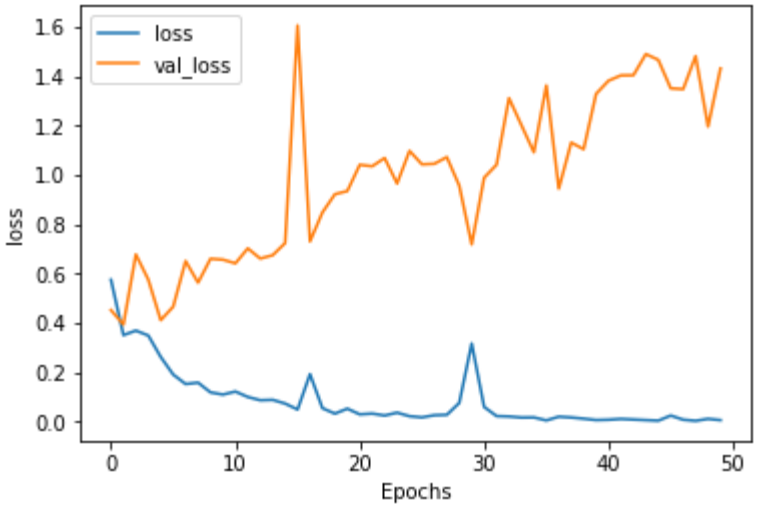


1 Layer
LSTM

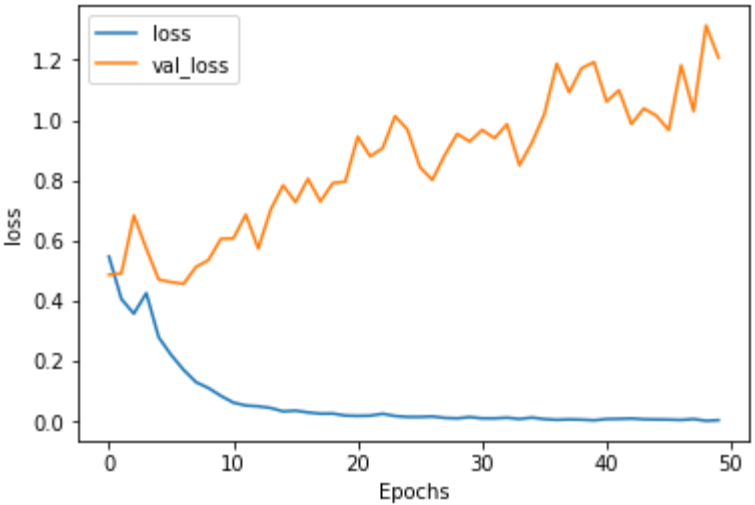


2 Layer
LSTM

50 Epochs : Loss Measurement



1 Layer
LSTM

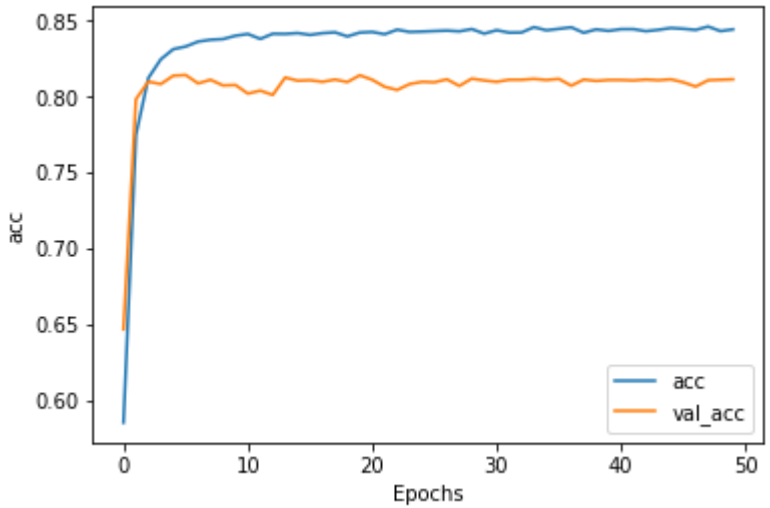


2 Layer
LSTM

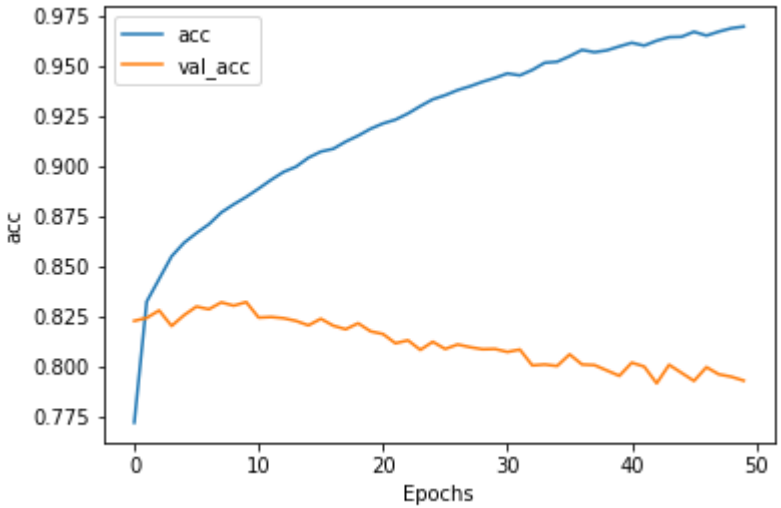
```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim,
                              input_length=max_length),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(24, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim,
                              input_length=max_length),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(24, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

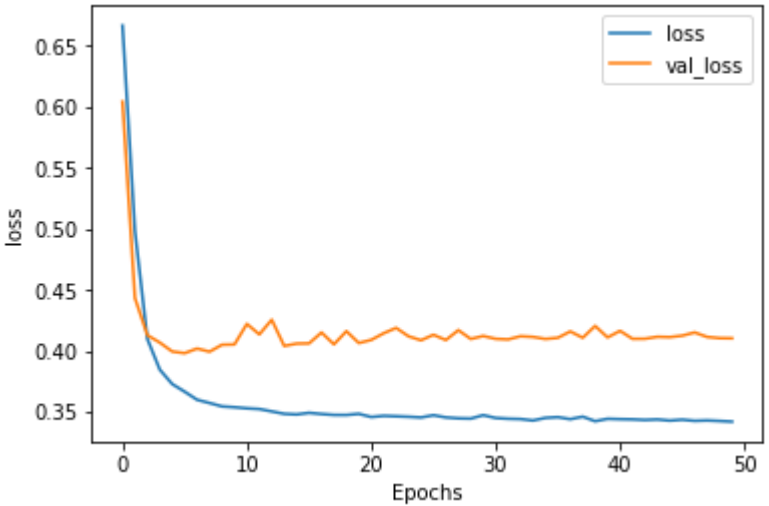
```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim,
                                                input_length=max_length),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
    tf.keras.layers.Dense(24, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```



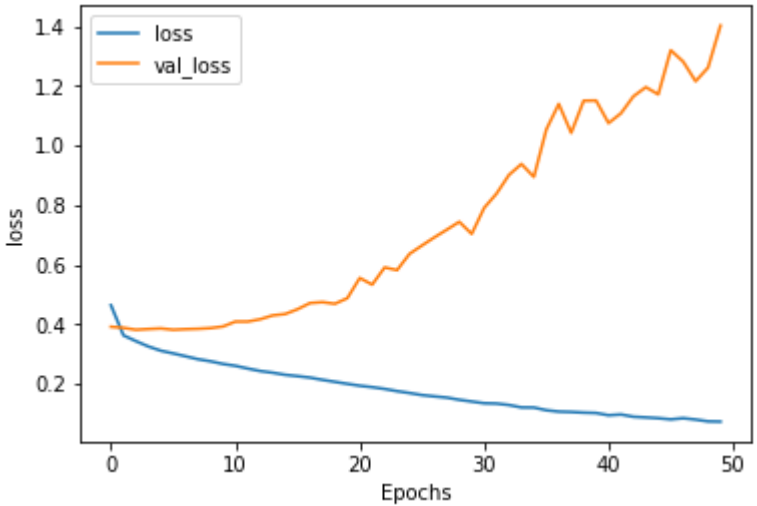
Without LSTM



With LSTM



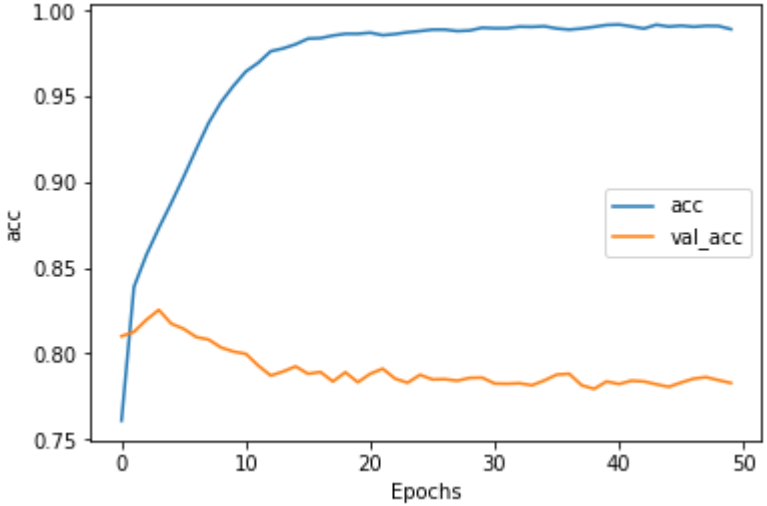
Without LSTM

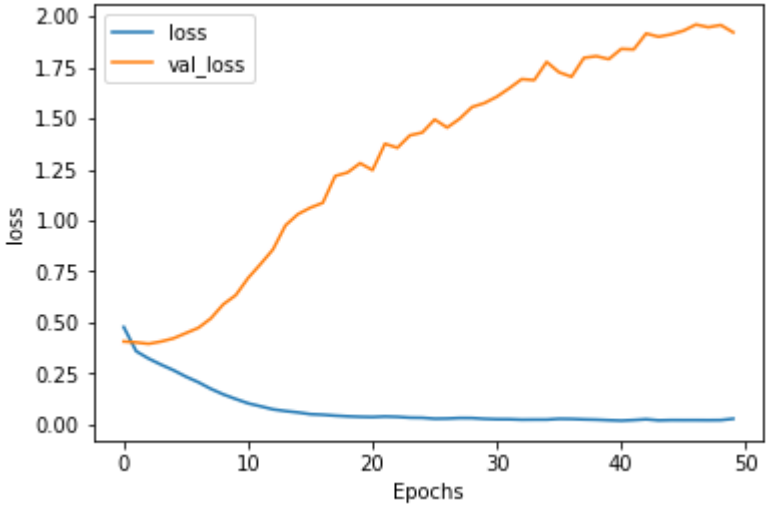


With LSTM

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim,
                              input_length=max_length),
    tf.keras.layers.Conv1D(128, 5, activation='relu'),
    tf.keras.layers.GlobalMaxPooling1D(),
    tf.keras.layers.Dense(24, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim,
                              input_length=max_length),
    tf.keras.layers.Conv1D(128, 5, activation='relu'),
    tf.keras.layers.GlobalMaxPooling1D(),
    tf.keras.layers.Dense(24, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```


acc
val_acc
Epochs
1.00
0.95
0.90
0.85
0.80
0.75
0
10
20
30
40
50


loss
val_loss
Epochs
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
10
20
30
40
50

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim,
                              input_length=max_length),
    tf.keras.layers.Conv1D(128, 5, activation='relu'),
    tf.keras.layers.GlobalMaxPooling1D(),
    tf.keras.layers.Dense(24, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
max_length = 120

tf.keras.layers.Conv1D(128, 5, activation='relu'),
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 120, 16)           16000
_________________________________________________________________
conv1d (Conv1D)              (None, 116, 128)          10368
_________________________________________________________________
global_max_pooling1d (Global (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 24)                3096
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 25
=================================================================
Total params: 29,489
Trainable params: 29,489
Non-trainable params: 0
```

```
max_length = 120

tf.keras.layers.Conv1D(128, 5, activation='relu'),
```

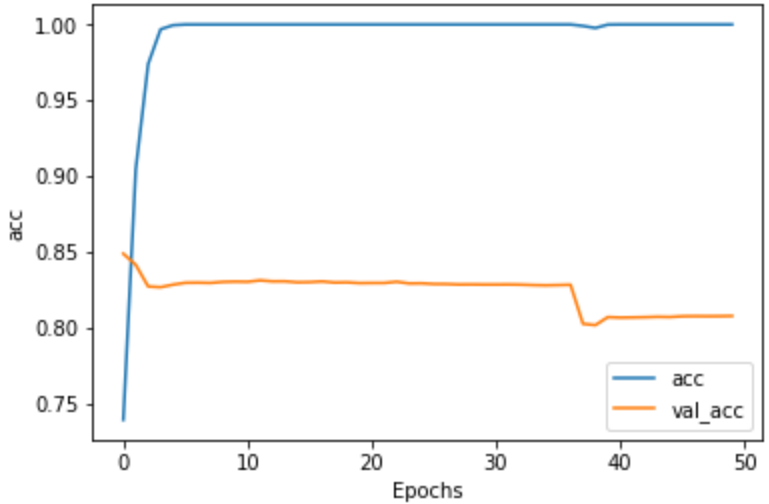
```
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 120, 16)           16000
_________________________________________________________________
conv1d (Conv1D)              (None, 116, 128)          10368
_________________________________________________________________
global_max_pooling1d (Global (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 24)                3096
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 25
=================================================================
Total params: 29,489
Trainable params: 29,489
Non-trainable params: 0
```

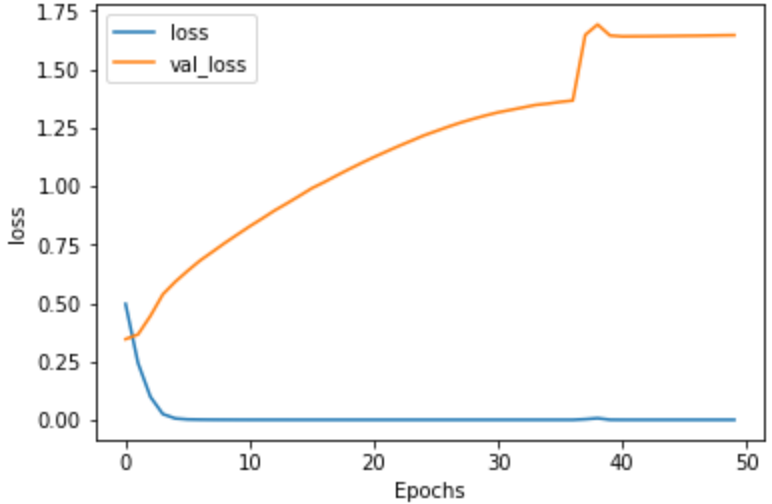
```
max_length = 120

tf.keras.layers.Conv1D(128, 5, activation='relu'),
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 120, 16)           16000
_________________________________________________________________
conv1d (Conv1D)              (None, 116, 128)          10368
_________________________________________________________________
global_max_pooling1d (Global (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 24)                3096
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 25
=================================================================
Total params: 29,489
Trainable params: 29,489
Non-trainable params: 0
```

```
imdb, info = tfds.load("imdb_reviews", with_info=True, as_supervised=True)



model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(6, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 120, 16)           160000
_________________________________________________________________
flatten (Flatten)            (None, 1920)              0
_________________________________________________________________
dense (Dense)                (None, 6)                 11526
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 7
=================================================================
Total params: 171,533
Trainable params: 171,533
Non-trainable params: 0
```

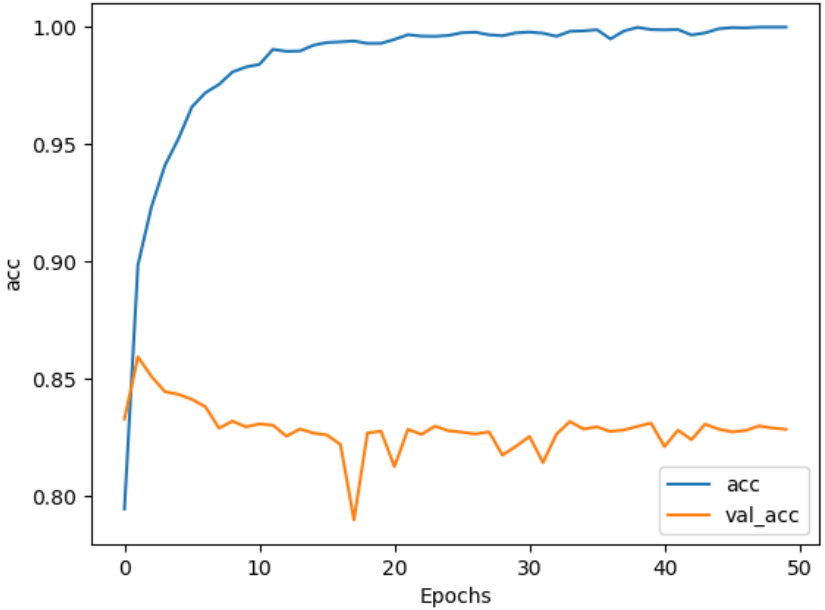


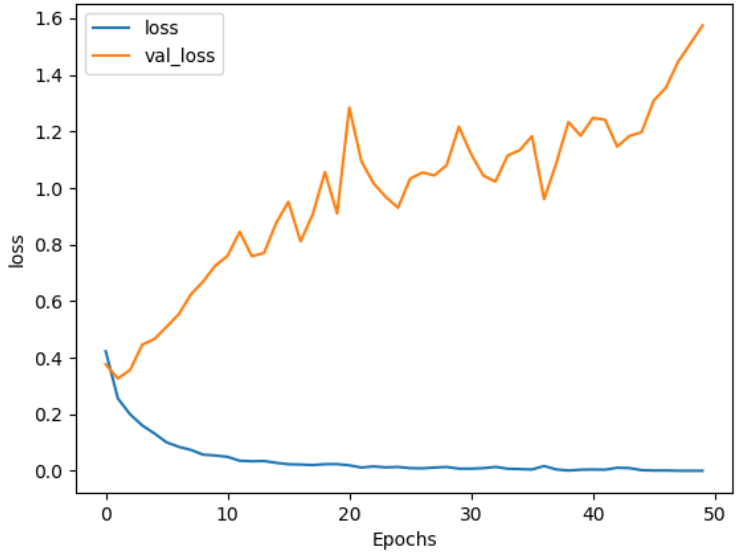


IMDB with Embedding-only : ~ 5s per epoch

```
imdb, info = tfds.load("imdb_reviews", with_info=True, as_supervised=True)



# Model Definition with LSTM
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
    tf.keras.layers.Dense(6, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_7 (Embedding)      (None, 120, 16)           160000
_________________________________________________________________
bidirectional_7 (Bidirection (None, 64)                12544
_________________________________________________________________
dense_14 (Dense)             (None, 6)                 390
_________________________________________________________________
dense_15 (Dense)             (None, 1)                 7
=================================================================
Total params: 173,941
Trainable params: 172,941
Non-trainable params: 0
```

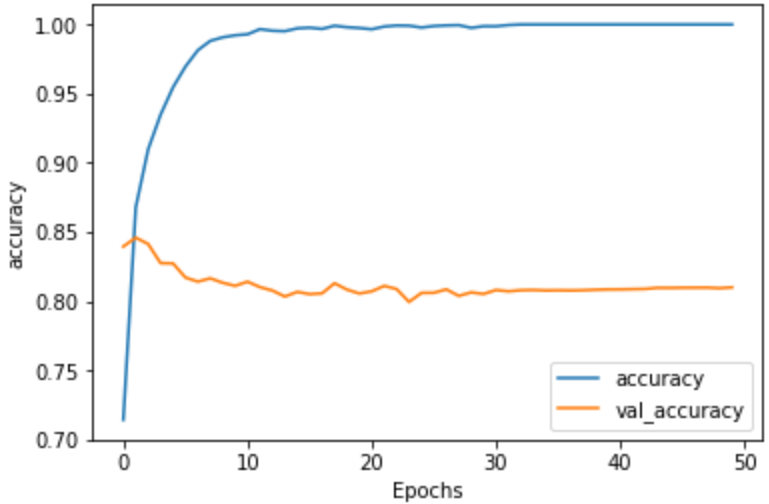


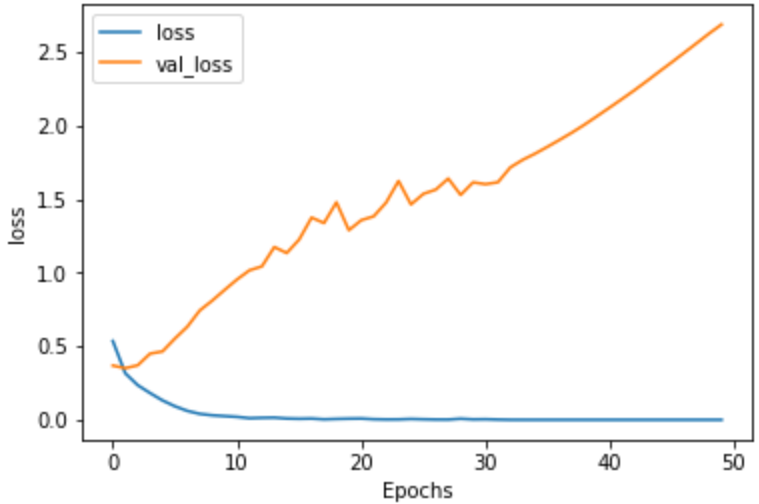


IMDB with LSTM ~43s per epoch

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.Bidirectional(tf.keras.layers.GRU(32)),
    tf.keras.layers.Dense(6, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 120, 16)           160000
_________________________________________________________________
bidirectional_1 (Bidirection (None, 64)                9600
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 390
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 7
=================================================================
Total params: 169,997
Trainable params: 169,997
Non-trainable params: 0
```

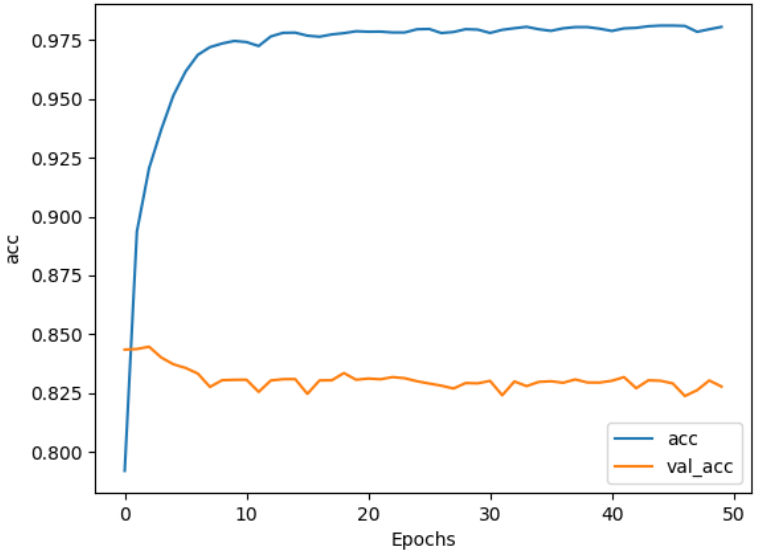


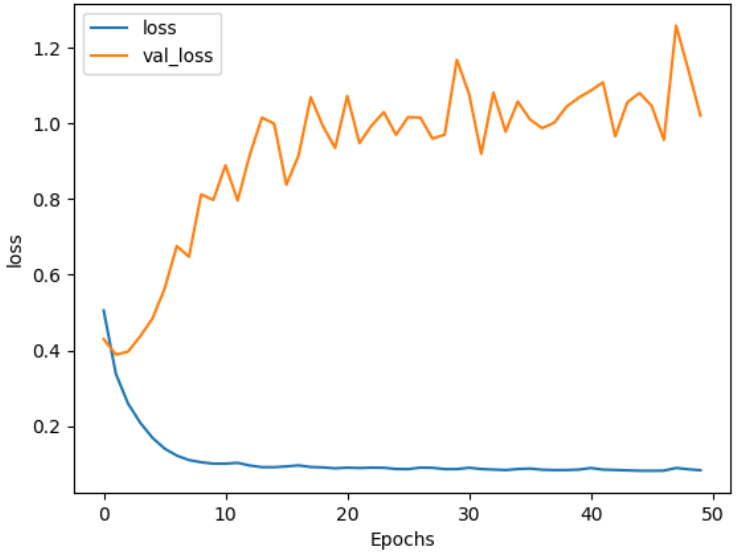


IMDB with GRU : ~ 20s per epoch

```
# Model Definition with Conv1D
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.Conv1D(128, 5, activation='relu'),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(6, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 120, 16)           160000
_________________________________________________________________
conv1d (Conv1D)              (None, 116, 128)          10368
_________________________________________________________________
global_average_pooling1d (Gl (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 6)                 774
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 7
=================================================================
Total params: 171,149
Trainable params: 171,149
Non-trainable params: 0
_________________________________________________________________
```





IMDB with CNN : ~ 6s per epoch