

# Deep Learning Adventures

## TensorFlow In Practice -

# Presentation 4

---

A quick overview of Coursera's Tensorflow in Practice specialization course

Robert Kraig, David Patton, George Zoto

<https://www.meetup.com/Deep-Learning-Adventures>

# In the beginning...

Meetup

The screenshot shows the homepage of the Deep Learning Adventures Meetup group. At the top, there's a photo of two men sitting on a blue couch. Below the photo, the group's name "Deep Learning Adventures" is displayed in large, bold letters. To the right of the name are links for "Start a new group", "Log in", and "Sign up". Under the group name, it says "Washington, DC", "377 members · Public group", and "Organized by George Z. and 2 others". There are sharing options (Facebook, Twitter, LinkedIn) and a "Join this group" button. The main content area includes sections for "What we're about", "Upcoming events (2)", and "Members (377)".

**What we're about**

Deep Learning Adventures is a welcoming group for anyone interested in learning more about deep learning, its foundations, its strengths and weaknesses and ever growing applications that best serve humanity and help those in need throughout the world. After participating in hundreds of meetups in the area, we have taken many lessons learned and incorporated them into this group. This group is also startup oriented in the sense that we are open minded and ready to pivot to new directions as our community and needs around the world guide us....

[Read more](#)

**Upcoming events (2)**

FRI, JUL 3, 7:30 PM EDT

**A chat with Laurence Moroney, AI Lead at Google**

Online event

Join us for a fun conversation with Laurence Moroney, AI Lead at Google (<https://www.linkedin.com/in/laurence-moroney/>) and developer of our TensorFlow in Practice and TensorFlow: Data and Deployment Specializations. We plan to...

**Members (377)**

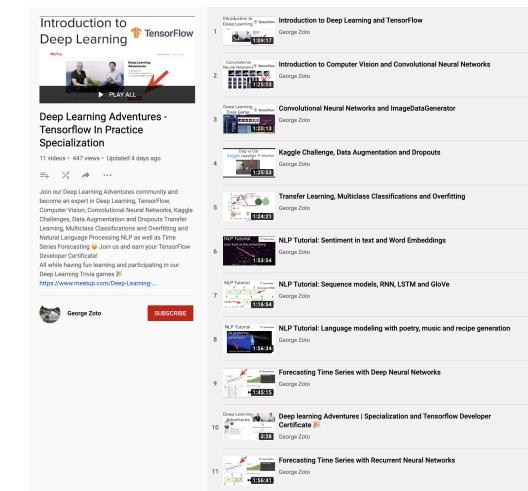
See all

**Organizers**

George Z. and 2 others

[Message](#)

- **Meetup Link**  
<https://www.meetup.com/Deep-Learning-Adventures>
- **GitHub repository**  
<https://github.com/georgezoto/Deep-Learning-Adventures>  
<https://github.com/georgezoto/TensorFlow-in-Practice>
- **Join us on Slack**  
[https://join.slack.com/t/deeplearninga-nmk8930/shared\\_invite/zt-d52h9mm9-h~Q0ZXw5PXsTDzPIINIVog](https://join.slack.com/t/deeplearninga-nmk8930/shared_invite/zt-d52h9mm9-h~Q0ZXw5PXsTDzPIINIVog)
- **Need a refresher or new to Deep Learning?**
- **YouTube recordings of all our Meetups 😊**  
<https://bit.ly/deep-learning-tf>



# Not a typical Meetup... Get ready for a fun game on 6/5



To play this game

1. Use any device to open  
**joinmyquiz.com**
2. Enter game code  
**775667**

or share via...

**START**

This is a screenshot of a Quizizz game setup screen. It shows instructions for playing the game, including the website URL and the game code. A large green "START" button at the bottom is intended to be pressed to begin the game.

Your Quizizz name is...

 Enter your name i 

**Start game**

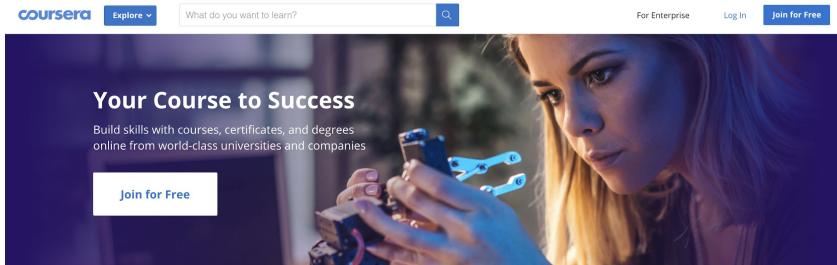
Game settings

 Music   Sound effects

 Read aloud

This is a screenshot of the Quizizz game lobby screen. It asks for the player's name, features a prominent green "Start game" button, and includes game settings for music, sound effects, and read-aloud options. The settings show that music and sound effects are currently enabled.

# Attribution to Coursera and deeplearning.ai



A screenshot of the deeplearning.ai website homepage. The header features the deeplearning.ai logo and a navigation menu with links for "Courses", "Workers", "The Batch", "Events", "Forums", "Blog", and "Company". The main section is titled "Break Into AI" with the subtext "Whether you want to build algorithms or build a company, deeplearning.ai's courses will teach you key concepts and applications of AI.". A red button at the bottom says "Take the Deep Learning Specialization". To the right, there's a graphic of a laptop screen showing two images: a "Content Image" of the Golden Gate Bridge and a "Generated Image" of the same bridge in a painterly style, labeled "Art Generation with Deep Learning".

Source:

<https://www.coursera.org/about/terms>  
<https://www.coursera.org/>  
<https://www.deeplearning.ai/>

# Chapter 1 - TensorFlow in Practice Specialization

## About this Specialization

199,621 recent views

Discover the tools software developers use to build scalable AI-powered algorithms in TensorFlow, a popular open-source machine learning framework.

In this four-course Specialization, you'll explore exciting opportunities for AI applications. Begin by developing an understanding of how to build and train neural networks. Improve a network's performance using convolutions as you train it to identify real-world images. You'll teach machines to understand, analyze, and respond to human speech with natural language processing systems. Learn to process text, represent sentences as vectors, and input data to a neural network. You'll even train an AI to create original poetry!

AI is already transforming industries across the world. After finishing this Specialization, you'll be able to apply your new TensorFlow skills to a wide range of problems and projects.

Looking for more advanced TensorFlow content? Check out the new [TensorFlow: Data and Deployment Specialization](#).

# Chapter 1 - TensorFlow in Practice Specialization

There are 4 Courses in this Specialization

COURSE

1

## Introduction to TensorFlow for Artificial Intelligence, Machine Learning, and Deep Learning

★★★★★ 4.7 6,196 ratings • 1,282 reviews

If you are a software developer who wants to build scalable AI-powered algorithms, you need to understand how to use the tools to build them. This course is part of the upcoming Machine Learning in Tensorflow Specialization and will teach you best practices for using TensorFlow, a popular open-source framework for machine learning.

[SHOW ALL](#)



6 hours to complete

### A New Programming Paradigm

Welcome to this course on going from Basics to Mastery of TensorFlow. We're excited you're here! In week 1 you'll get a soft introduction to what Machine Learning and Deep Learning are, and how they offer you a new programming paradigm, giving you a new set of tools to open previously unexplored scenarios. All you need to know is some very basic SHOW ALL



4 videos (Total 16 min), 5 readings, 3 quizzes [SEE ALL](#)

COURSE

2

## Convolutional Neural Networks in TensorFlow

★★★★★ 4.7 2,751 ratings • 410 reviews

If you are a software developer who wants to build scalable AI-powered algorithms, you need to understand how to use the tools to build them. This course is part of the upcoming Machine Learning in Tensorflow Specialization and will teach you best practices for using TensorFlow, a popular open-source framework for machine learning.

[SHOW ALL](#)



7 hours to complete

### Introduction to Computer Vision

Welcome to week 2 of the course! In week 1 you learned all about how Machine Learning and Deep Learning is a new programming paradigm. This week you're going to take that to the next level by beginning to solve problems of computer vision with just a few lines of code! SHOW ALL



7 videos (Total 15 min), 6 readings, 3 quizzes [SEE ALL](#)

COURSE

3

## Natural Language Processing in TensorFlow

★★★★★ 4.6 2,037 ratings • 277 reviews

If you are a software developer who wants to build scalable AI-powered algorithms, you need to understand how to use the tools to build them. This Specialization will teach you best practices for using TensorFlow, a popular open-source framework for machine learning.

[SHOW ALL](#)



8 hours to complete

### Enhancing Vision with Convolutional Neural Networks

Welcome to week 3! In week 2 you saw a basic Neural Network for Computer Vision. It did the job nicely, but it was a little naive in its approach. This week we'll see how to make it better, as discussed by Laurence and Andrew here. SHOW ALL



6 videos (Total 19 min), 6 readings, 3 quizzes [SEE ALL](#)

COURSE

4

## Sequences, Time Series and Prediction

★★★★★ 4.6 1,374 ratings • 223 reviews

If you are a software developer who wants to build scalable AI-powered algorithms, you need to understand how to use the tools to build them. This Specialization will teach you best practices for using TensorFlow, a popular open-source framework for machine learning.

[SHOW ALL](#)



9 hours to complete

### Using Real-world Images

Last week you saw how to improve the results from your deep neural network using convolutions. It was a good start, but the data you used was very basic. What happens when your images are larger, or if the features aren't always in the same place? Andrew and Laurence discuss this to prepare you for what you'll learn this week: handling complex images! SHOW ALL

Source: <https://www.coursera.org/specializations/tensorflow-in-practice>

# Setup



Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. You can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser.

<https://colab.research.google.com>

---

# **Course 3: Natural Language Processing in TensorFlow**

**Week 1: Sentiment in text**

**Week 2: Word Embeddings**

**Week 3: Sequence models**

**Week 4: Sequence models and literature**

**Sentiment can be tricky.** (That DJ killed me last night vs  
That DJ killed it last night)

- **Problem: Models need numbers for training**
- **Solution: Tokenization**

**Convert text (input and output) into numbers via a Tokenizer**

1. **Characters**
2. **Words**
3. **Subwords**

# Sentiment in text

1  83 073 076 069 078 084

S I L E N T

I Love my dog

001 002 003 004

076 073 083 084 069 078

L I S T E N

I Love my cat

001 002 003 004 005

 deplearning.ai

 deplearning.ai

2 

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer
```

7 

```
{'i': 3, 'my': 2, 'you': 6, 'love': 1, 'cat': 5, 'dog': 4}
```

3 

```
sentences = [
    'I love my dog',
    'I love my cat'
]
```

4 

```
tokenizer = Tokenizer(num_words = 100)
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index
print(word_index)
```

5 

```
{'i': 1, 'my': 3, 'dog': 4, 'cat': 5, 'love': 2}
```

6 

```
sentences = [
    'I love my dog',
    'I love my cat',
    'You love my dog!'
]
```

# Sentiment in text

```
from tensorflow.keras.preprocessing.text import Tokenizer

sentences = [
    'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
]

tokenizer = Tokenizer(num_words = 100)
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index

sequences = tokenizer.texts_to_sequences(sentences)

print(word_index)          {'amazing': 10, 'dog': 3, 'you': 5, 'cat': 6,
print(sequences)           'think': 8, 'i': 4, 'is': 9, 'my': 1, 'do': 7,
                           'love': 2}
                           [[4, 2, 1, 3], [4, 2, 1, 6], [5, 2, 1, 3], [7, 5,
                           8, 1, 3, 9, 10]]]
```

Source: <https://www.coursera.org/learn/natural-language-processing-tensorflow>

# Sentiment in text

```
test_data = [  
    'i really love my dog',  
    'my dog loves my manatee'  
]  
  
test_seq = tokenizer.texts_to_sequences(test_data)  
print(test_seq)  
  
[[4, 2, 1, 3], [1, 3, 1]]  
  
{'think': 8, 'amazing': 10, 'my': 1, 'love': 2, 'dog': 3, 'is': 9, 'you': 5, 'do': 7,  
'cat': 6, 'i': 4}
```



```
tokenizer = Tokenizer(num_words = 100, oov_token=<OOV>)  
{'think': 9, 'amazing': 11, 'dog': 4, 'do': 8, 'i': 5, 'cat': 7,  
 'you': 6, 'love': 3, '<OOV>': 1, 'my': 2, 'is': 10}  
[[5, 1, 3, 2, 4], [2, 4, 1, 2, 1]]
```

Source: <https://www.coursera.org/learn/natural-language-processing-tensorflow>

# Sentiment in text

```
from tensorflow.keras.preprocessing.sequence import pad_sequences

sequences = tokenizer.texts_to_sequences(sentences)

padded = pad_sequences(sequences)
```

```
sentences = [
    'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
]

{'do': 8, 'you': 6, 'love': 3, 'i': 5, 'amazing': 11, 'my': 2, 'is': 10, 'think': 9,
'dog': 4, '<OOV>': 1, 'cat': 7}

[[5, 3, 2, 4], [5, 3, 2, 7], [6, 3, 2, 4], [8, 6, 9, 2, 4, 10, 11]]

[[ 0  0  0  5  3  2  4]
 [ 0  0  0  5  3  2  7]
 [ 0  0  0  6  3  2  4]
 [ 8  6  9  2  4 10 11]]
```

```
padded = pad_sequences(sequences, padding='post',
                           truncating='post', maxlen=5)
```

Source: <https://www.coursera.org/learn/natural-language-processing-tensorflow>

# Sentiment in text

Dataset

News Headlines Dataset For Sarcasm Detection

High quality dataset for the task of Sarcasm Detection

 BAZINGA!  
The Big Bang Theory

Rishabh Misra • updated 10 months ago (Version 2)

Data Tasks Kernels (79) Discussion (3) Activity Metadata

Download (11 MB) New Notebook

More



Each record consists of three attributes:

- `is_sarcastic` : 1 if the record is sarcastic otherwise 0
- `headline` : the headline of the news article
- `article_link` : link to the original news article. Useful in collecting supplementary data

# **Course 3: Natural Language Processing in TensorFlow**

**Week 1:** Sentiment in text

**Week 2:** Word Embeddings

**Week 3:** Sequence models

**Week 4:** Sequence models and literature

# Word Embeddings

| audio                     | "fashion_mnist"          | text                  |
|---------------------------|--------------------------|-----------------------|
|                           | "horses_or_humans"       |                       |
|                           | "image_label_folder"     | "cnn_dailymail"       |
|                           | "imagenet2012"           | "glue"                |
| image                     | "imagenet2012_corrupted" | "imdb_reviews"        |
|                           | "kmnist"                 | "lm1b"                |
|                           | "lsun"                   | "multi_nli"           |
| "abstract_reasoning"      | "mnist"                  | "squad"               |
| "caltech101"              | "omniglot"               | "wikipedia"           |
| "cats_vs_dogs"            | "open_images_v4"         | "xnli"                |
| "celeb_a"                 | "oxford_iit_pet"         |                       |
| "celeb_a_hq"              | "quickdraw_bitmap"       | translate             |
| "cifar10"                 | "rock_paper_scissors"    |                       |
| "cifar100"                | "shapes3d"               | "flores"              |
| "cifar10_corrupted"       | "smallnorb"              | "para_crawl"          |
| "coco2014"                | "sun397"                 | "ted_hrlr_translate"  |
| "colorectal_histology"    | "svhn_cropped"           | "ted_multi_translate" |
| "cycle_gan"               | "tf_flowers"             | "wmt15_translate"     |
| "diabetic_retinopathy..." |                          | "wmt16_translate"     |
| "dsprites"                |                          | "wmt17_translate"     |
| "dtd"                     |                          | "wmt18_translate"     |
| "emnist"                  |                          | "wmt19_translate"     |
|                           | "higgs"                  |                       |
|                           | "iris"                   |                       |
|                           | "titanic"                |                       |
|                           |                          | structured            |

Source:

<https://www.tensorflow.org/datasets/catalog/overview>

<https://github.com/tensorflow/datasets>

[https://www.tensorflow.org/datasets/catalog/imdb\\_reviews](https://www.tensorflow.org/datasets/catalog/imdb_reviews)

## TensorFlow Datasets

|      |                             |
|------|-----------------------------|
| Text | blimp                       |
|      | c4 (manual)                 |
|      | cfq                         |
|      | civil_comments              |
|      | cos_e                       |
|      | definite_pronoun_resolution |
|      | eraser_multi_rc             |
|      | esnli                       |
|      | gap                         |
|      | glue                        |
|      | imdb_reviews                |
|      | librispeech_lm              |
|      | lm1b                        |
|      | math_dataset                |
|      | movie rationales            |
|      | multi_nli                   |
|      | multi_nli_mismatch          |
|      | natural_questions           |
|      | qa4mre                      |
|      | scan                        |
|      | scicite                     |
|      | snli                        |
|      | squad                       |
|      | super_glue                  |
|      | tiny_shakespeare            |
|      | trivia_qa                   |
|      | web_questions               |
|      | wiki40b                     |
|      | wikipedia                   |
|      | xnli                        |
|      | yelp_polarity_reviews       |

# Word Embeddings

## IMDB Reviews: Large Movie Review Dataset

This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. We provide a set of 25,000 highly polar movie reviews for training, and 25,000 for testing. There is additional unlabeled data for use as well. Raw text and already processed bag of words formats are provided. See the README file contained in the release for more details.

[Large Movie Review Dataset v1.0](#)

When using this dataset, please cite our ACL 2011 paper [\[bib\]](#).

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). [Learning Word Vectors for Sentiment Analysis, 2011](#).

In [0]: `# !pip install -q tensorflow-datasets`

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

```
training_sentences[:10]
```

```
'b"This was an absolutely terrible movie. Don\'t be lured in by Christopher Walken or Michael Ironside.  
'b'I have been known to fall asleep during films, but this is usually due to a combination of things in  
'b'Mann photographs the Alberta Rocky Mountains in a superb fashion, and Jimmy Stewart and Walter Brenn  
'b'This is the kind of film for a snowy Sunday afternoon when the rest of the world can go ahead with i  
'b'As others have mentioned, all the women that go nude in this film are mostly absolutely gorgeous. I  
'b'This is a film which should be seen by anybody interested in, effected by, or suffering from an eati  
'b'Okay, you have:<br /><br />Penelope Keith as Miss Herringbone-Tweed, B.B.E. (Backbone of England.)  
'b'The film is based on a genuine 1950s novel.<br /><br />Journalist Colin McInnes wrote a set of thre  
'b'I really love the sexy action and sci-fi films of the sixties and its because of the actress\\\'s t  
'b'Sure, this one isn\\\'t really a blockbuster, nor does it target such a position. "Dieter" is the f
```

In [1]:

```
import sys
print(sys.version)
import tensorflow as tf
print(tf.__version__)

# !pip install -q tensorflow-datasets
```

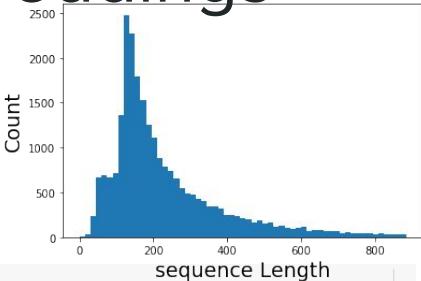
```
3.7.7 (default, Mar 23 2020, 23:19:08)
2.1.0
```

```
training_labels_final[:10]
```

```
array([0, 0, 0, 1, 1, 1, 0, 0, 0, 0])
```

# Word Embeddings

```
1 vocab_size = 10000
embedding_dim = 16
max_length = 120
trunc_type='post'
oov_tok = "<OOV>"
```



```
3 el = 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()
```

Model: "sequential"

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

```
4 num_epochs = 10
model.fit(padded, training_labels_final, epochs=num_epochs,
           validation_data=(testing_padded, testing_labels_final))
```

```
2 from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

tokenizer = Tokenizer(num_words = vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(training_sentences)
word_index = tokenizer.word_index
sequences = tokenizer.texts_to_sequences(training_sentences)
padded = pad_sequences(sequences,maxlen=max_length, truncating=trunc_type)

testing_sequences = tokenizer.texts_to_sequences(testing_sentences)
testing_padded = pad_sequences(testing_sequences,maxlen=max_length)
```

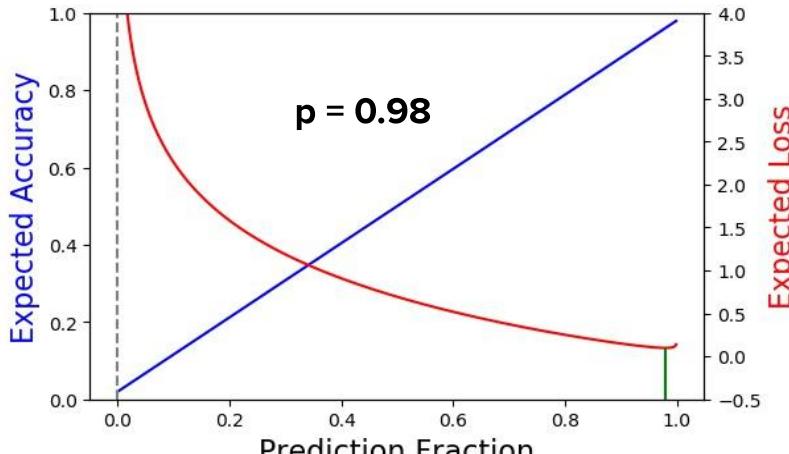
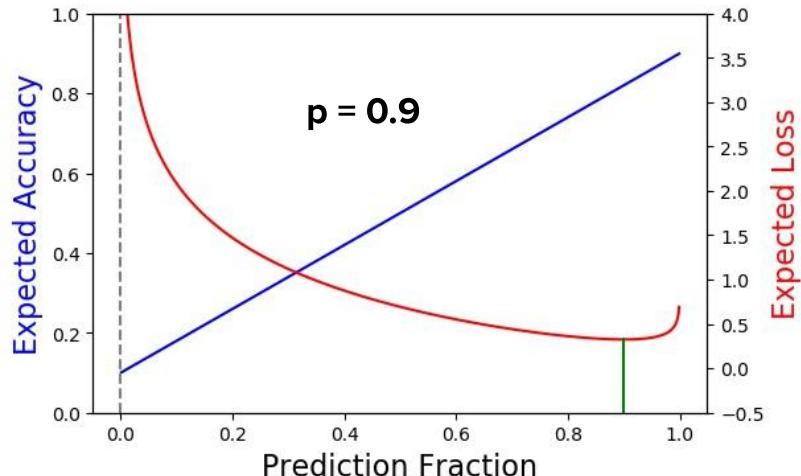
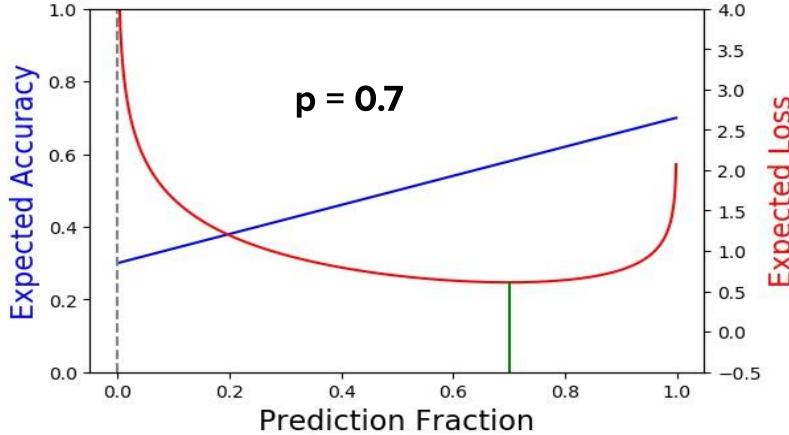
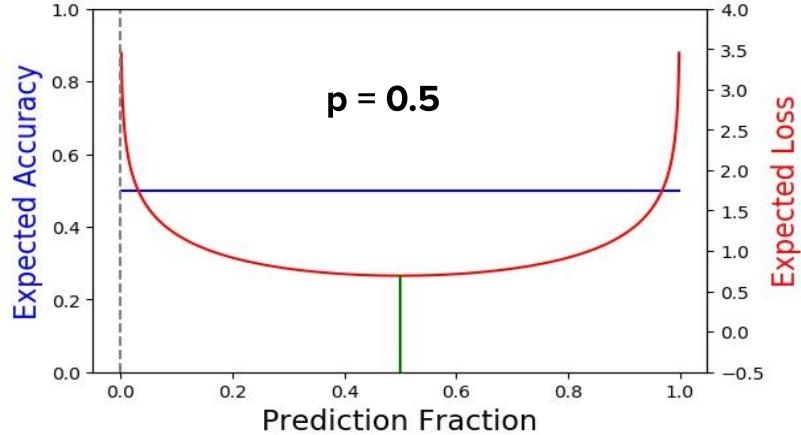
```
5 Epoch 1/10
782/782 [=====] - 5s 7ms/step - loss: 0.4831 - accuracy: 0.7515 - val_loss: 0.3459 - val_accuracy: 0.8464
Epoch 2/10
782/782 [=====] - 5s 6ms/step - loss: 0.2349 - accuracy: 0.9095 - val_loss: 0.3742 - val_accuracy: 0.8369
Epoch 3/10
782/782 [=====] - 5s 7ms/step - loss: 0.0888 - accuracy: 0.9771 - val_loss: 0.4547 - val_accuracy: 0.8264
Epoch 4/10
782/782 [=====] - 5s 7ms/step - loss: 0.0212 - accuracy: 0.9972 - val_loss: 0.5409 - val_accuracy: 0.8222
Epoch 5/10
782/782 [=====] - 5s 6ms/step - loss: 0.0051 - accuracy: 0.9998 - val_loss: 0.5965 - val_accuracy: 0.8241
Epoch 6/10
782/782 [=====] - 5s 6ms/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.6408 - val_accuracy: 0.8271
Epoch 7/10
782/782 [=====] - 5s 6ms/step - loss: 8.2845e-04 - accuracy: 1.0000 - val_loss: 0.6797 - val_accuracy: 0.8280
Epoch 8/10
782/782 [=====] - 5s 6ms/step - loss: 4.3873e-04 - accuracy: 1.0000 - val_loss: 0.7181 - val_accuracy: 0.8284
Epoch 9/10
782/782 [=====] - 5s 6ms/step - loss: 2.5901e-04 - accuracy: 1.0000 - val_loss: 0.7565 - val_accuracy: 0.8286
Epoch 10/10
782/782 [=====] - 5s 6ms/step - loss: 1.5668e-04 - accuracy: 1.0000 - val_loss: 0.7907 - val_accuracy: 0.8286
<tensorflow.python.keras.callbacks.History at 0x7f4f93c85c18>
```

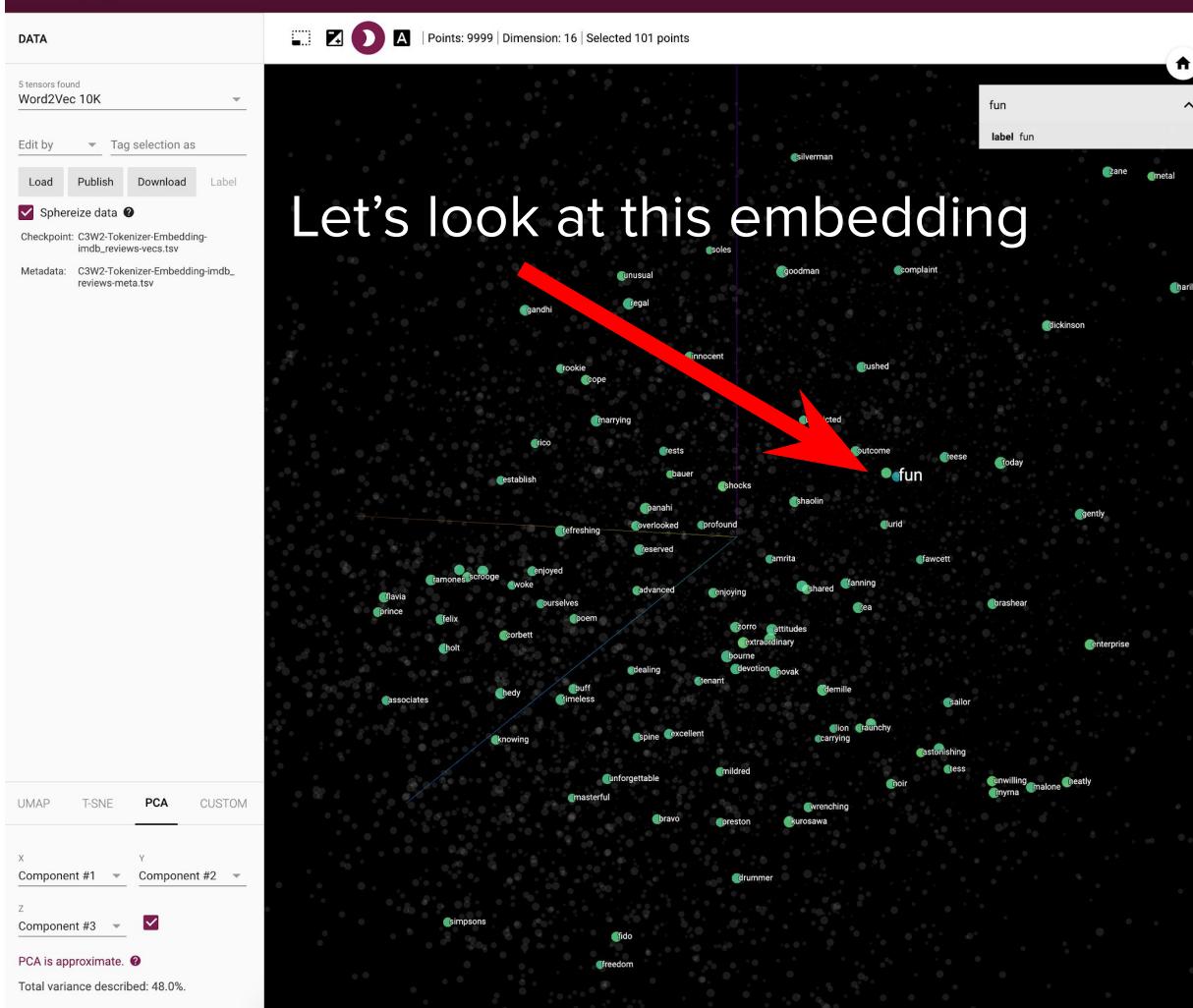
```
6 e = model.layers[0]
weights = e.get_weights()[0]
print(weights.shape) # shape: (vocab_size, embedding_dim)

(10000, 16)
```

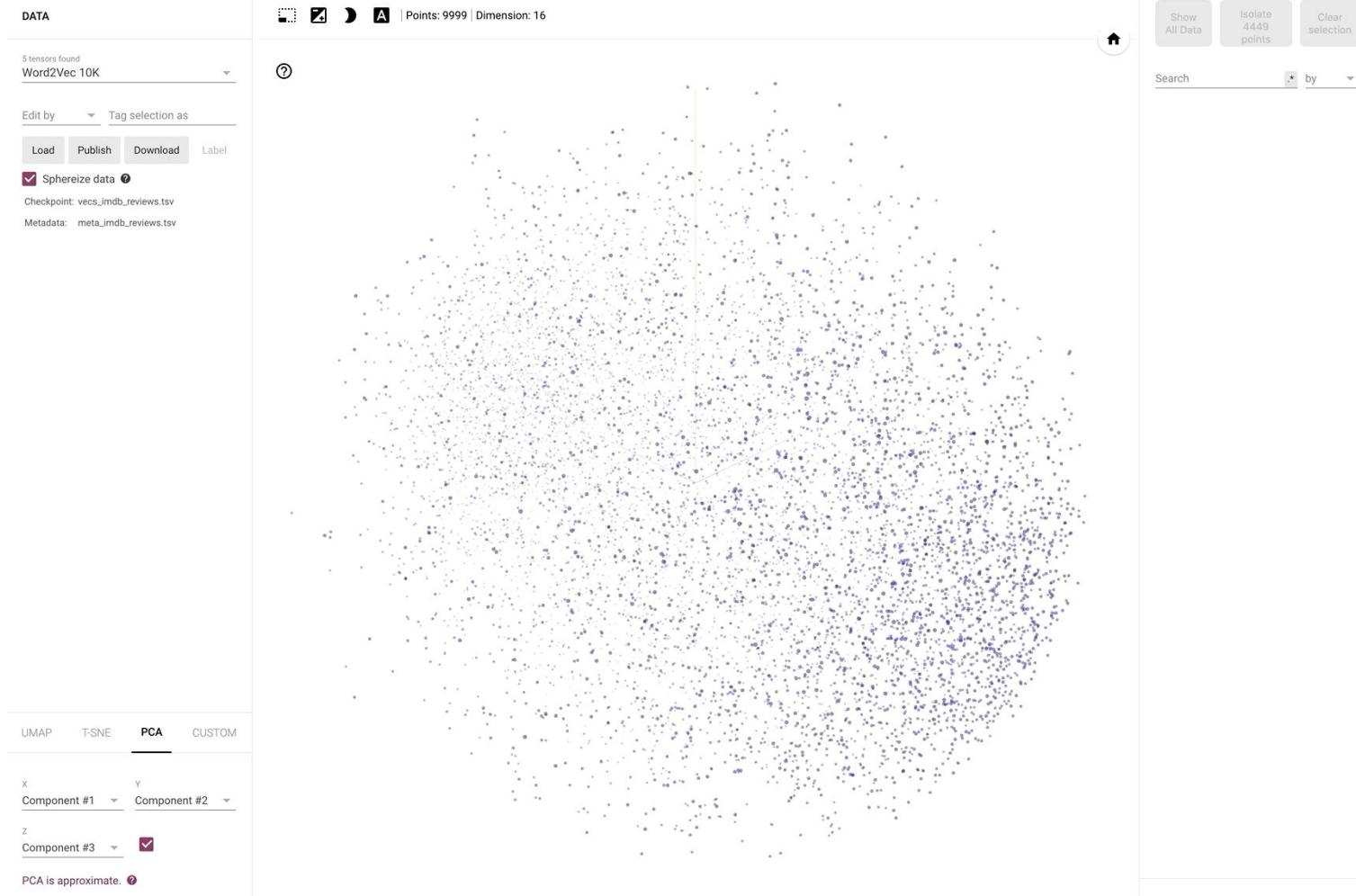
```
7 files.download('vecs.tsv')
files.download('meta.tsv')
```

# Word Embeddings: Revisit Loss vs. Accuracy





Source:  
<http://projector.tensorflow.org/>



# Word Embeddings

`imdb_reviews` is configured with `tfds.text.imdb.IMDBReviewsConfig` and has the following configurations predefined (defaults to the first one):

- "plain\_text" ( v0.0.1 ) ( Size: 80.23 MiB ): Plain text
- "bytes" ( v0.0.1 ) ( Size: 80.23 MiB ): Uses byte-level text encoding with `tfds.features.text.ByteTextEncoder`
- "subwords8k" ( v0.0.1 ) ( Size: 80.23 MiB ): Uses `tfds.features.text.SubwordTextEncoder` with 8k vocab size
- "subwords32k" ( v0.0.1 ) ( Size: 80.23 MiB ): Uses `tfds.features.text.SubwordTextEncoder` with 32k vocab size

```
print(tokenizer.subwords)

['the_', ' ', '.', ' ', 'a_', 'and_', 'of_', 'to_', 's_', 'is_',
'br', 'in_', 'I_', 'that_', 'this_', 'it_', ... ]
```

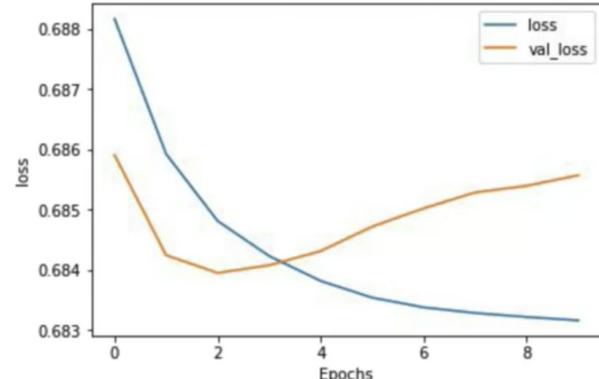
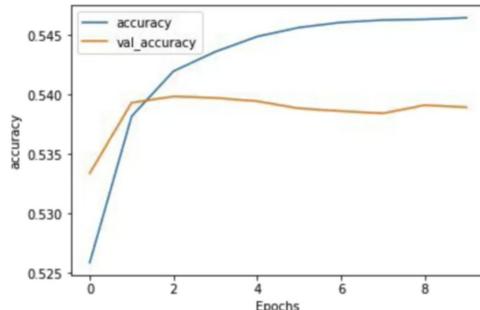
“...sub-word meanings are often nonsensical and it's only when we put them together in sequences that they have meaningful semantics.”

Source:

[https://www.tensorflow.org/datasets/catalog/imdb\\_reviews](https://www.tensorflow.org/datasets/catalog/imdb_reviews)

[https://www.tensorflow.org/datasets/api\\_docs/python/tfds/features/text/SubwordTextEncoder](https://www.tensorflow.org/datasets/api_docs/python/tfds/features/text/SubwordTextEncoder)

6307 ----> Ten  
2327 ----> sor  
4043 ----> Fl  
2120 ----> ow  
2 ----> ,  
48 ----> from  
4249 ----> basi  
4429 ----> cs  
7 ----> to  
2652 ----> master  
8050 ----> y



# Let's continue our NLP adventure 😊

FRI, MAY 22, 7:30 PM EDT

## TensorFlow in Practice - Course 3 Week 1,2 - Sentiment in text & Wor...

Online event



Join us for our 6th adventure in Deep Learning! Just bring your curiosity and be ready to meet our growing community 😊 We are taking Course 3 of TensorFlow in Practice Specialization available at:...



31 attendees

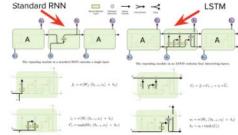


FRI, MAY 29, 7:30 PM EDT

## TensorFlow in Practice - Course 3 Week 3 - Sequence models

Online event

Join us for our 7th adventure in Deep Learning! Just bring your curiosity and be ready to meet our growing community 😊 We are taking Course 3 of TensorFlow in Practice Specialization available at:...



Attend

FRI, JUN 5, 7:30 PM EDT

## TensorFlow in Practice - C3 Week 4 - Sequence models & literature +...

Online event



Join us for our 8th adventure in Deep Learning! Just bring your curiosity and be ready to meet our growing community 😊 We are taking Course 3 of TensorFlow in Practice Specialization available at:...



13 attendees

Attend

FRI, JUN 12, 7:30 PM EDT

## Deep Learning Adventures - Happy Hour 🍻 🍹 🍹

Online event

Join us for a fun conversation 🍻 🍹 🍹 No slides or recording this time, just getting to know each other better and forge a stronger community 😊 We are taking a break from our TensorFlow in Practice Specialization available at:...



Attend

# **Course 3: Natural Language Processing in TensorFlow**

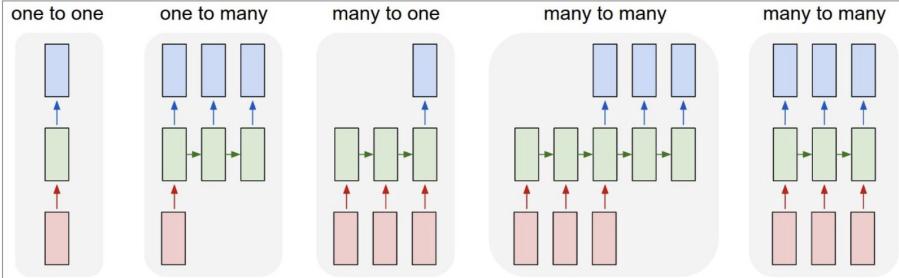
**Week 1:** Sentiment in text

**Week 2:** Word Embeddings

**Week 3:** Sequence models

**Week 4:** Sequence models and literature

# Sequence models



Each rectangle is a vector and arrows represent functions (e.g. matrix multiply). Input vectors are in red, output vectors are in blue and green vectors hold the RNN's state (more on this soon). From left to right: (1) Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification). (2) Sequence output (e.g. image captioning takes an image and outputs a sentence of words). (3) Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment). (4) Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French). (5) Synced sequence input and output (e.g. video classification where we wish to label each frame of the video). Notice that in every case are no pre-specified constraints on the lengths of sequences because the recurrent transformation (green) is fixed and can be applied as many times as we like.

```
rnn = RNN()  
y = rnn.step(x) # x is an input vector, y is the RNN's output vector
```

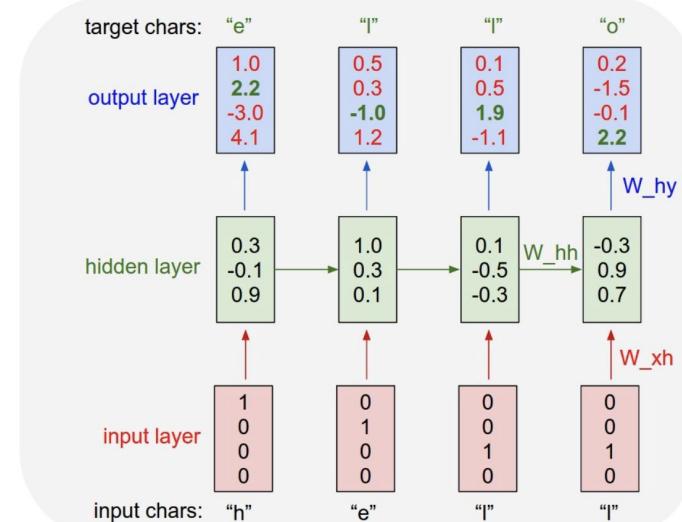
We can form a 2-layer recurrent network as follows:

```
y1 = rnn1.step(x)  
y = rnn2.step(y1)
```

```
self.h = ...
```

```
class RNN:  
    # ...  
    def step(self, x):  
        # update the hidden state  
        self.h = np.tanh(np.dot(self.W_hh, self.h) + np.dot(self.W_xh, x))  
        # compute the output vector  
        y = np.dot(self.W_hy, self.h)  
        return y
```

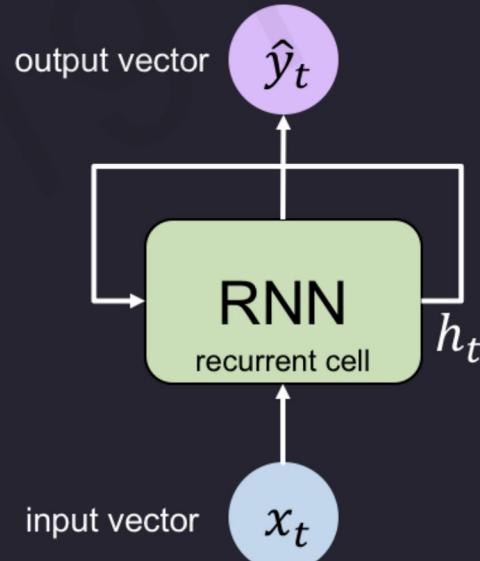
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t), \text{ where } \tanh \text{ is applied elementwise.}$$



# Sequence models

## RNN Intuition

```
my_rnn = RNN()  
hidden_state = [0, 0, 0, 0]  
  
sentence = ["I", "love", "recurrent", "neural"]  
  
for word in sentence:  
    prediction, hidden_state = my_rnn(word, hidden_state)  
  
    next_word_prediction = prediction  
    # >>> "networks!"
```



# Sequence models

## RNNs from Scratch

```
class MyRNNCell(tf.keras.layers.Layer):
    def __init__(self, rnn_units, input_dim, output_dim):
        super(MyRNNCell, self).__init__()

        # Initialize weight matrices
        self.W_xh = self.add_weight([rnn_units, input_dim])
        self.W_hh = self.add_weight([rnn_units, rnn_units])
        self.W_hy = self.add_weight([output_dim, rnn_units])

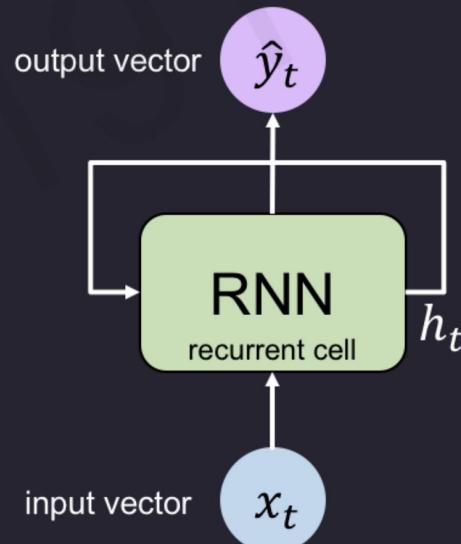
        # Initialize hidden state to zeros
        self.h = tf.zeros([rnn_units, 1])

    def call(self, x):
        # Update the hidden state
        self.h = tf.math.tanh( self.W_hh * self.h + self.W_xh * x )

        # Compute the output
        output = self.W_hy * self.h

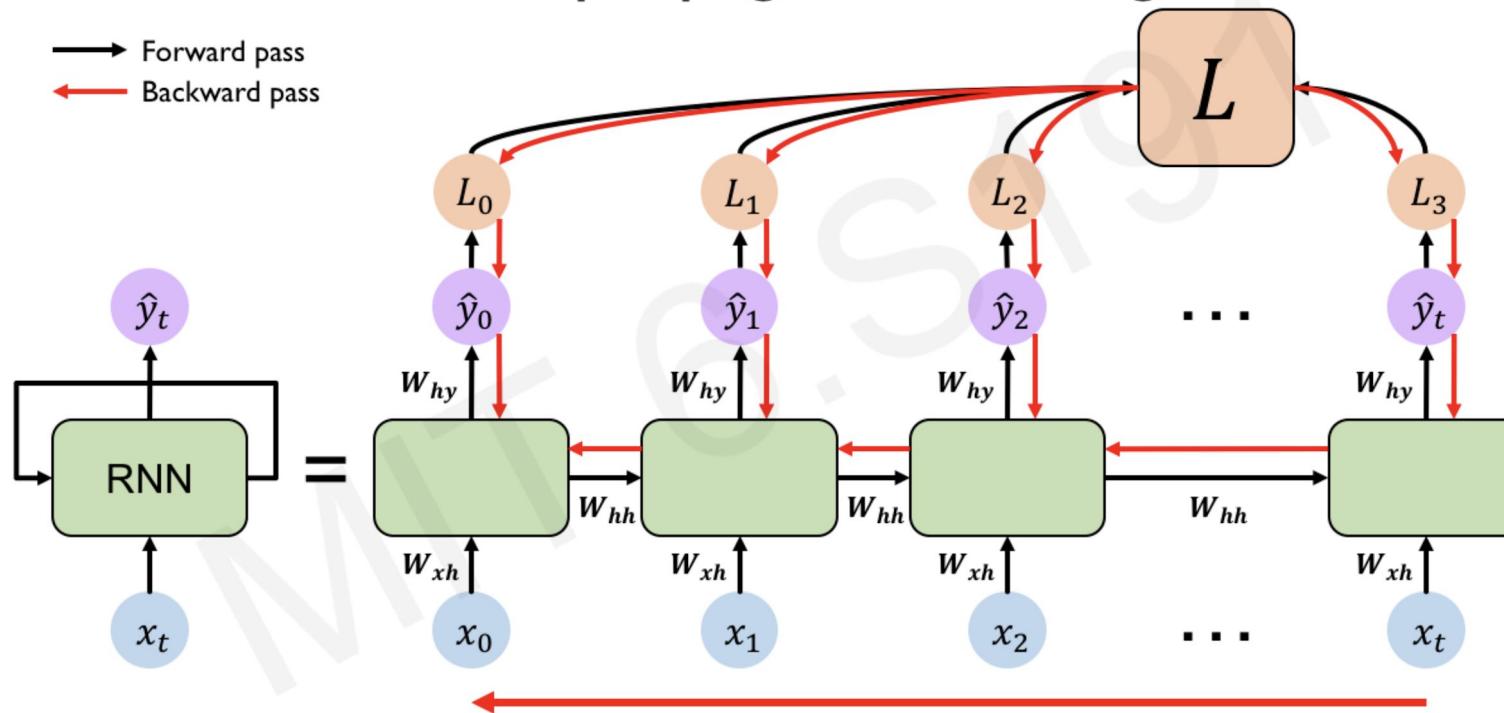
        # Return the current output and hidden state
        return output, self.h
```

`tf.keras.layers.SimpleRNN(rnn_units)`



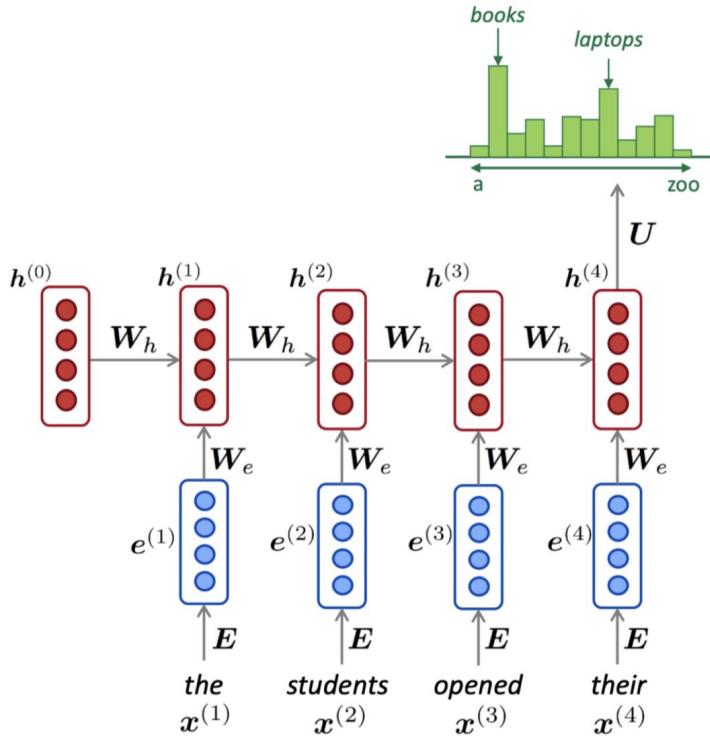
# Sequence models

## RNNs: Backpropagation Through Time

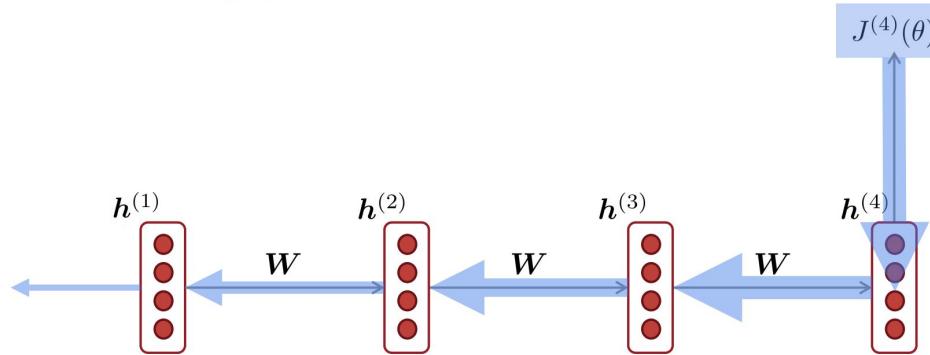


# Sequence models - new

$$\hat{y}^{(4)} = P(\mathbf{x}^{(5)} | \text{the students opened their})$$



## Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \times \frac{\partial h^{(3)}}{\partial h^{(2)}} \times \frac{\partial h^{(4)}}{\partial h^{(3)}} \times \frac{\partial J^{(4)}}{\partial h^{(4)}}$$

What happens if these are small?

Vanishing gradient problem:  
When these are small, the gradient signal gets smaller and smaller as it backpropagates further

# Sequence models - new

$$h_t = \sigma(W^{(hh)}h_{t-1} + W^{(hx)}x_{[t]}) \quad (5)$$

$$\hat{y}_t = \text{softmax}(W^{(S)}h_t) \quad (6)$$

The loss function used in RNNs is often the cross entropy error introduced in earlier notes. Equation 7 shows this function as the sum over the entire vocabulary at time-step  $t$ .

$$J^{(t)}(\theta) = -\sum_{j=1}^{|V|} y_{t,j} \times \log(\hat{y}_{t,j}) \quad (7)$$

The cross entropy error over a corpus of size  $T$  is:

$$J = \frac{1}{T} \sum_{t=1}^T J^{(t)}(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{j=1}^{|V|} y_{t,j} \times \log(\hat{y}_{t,j}) \quad (8)$$

Equation 9 is called the *perplexity* relationship; it is basically 2 to the power of the negative log probability of the cross entropy error function shown in Equation 8. Perplexity is a measure of confusion where lower values imply more confidence in predicting the next word in the sequence (compared to the ground truth outcome).

$$\text{Perplexity} = 2^J$$

Source: [\(9\)](http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture07-fancy-rnn.pdf)

- $x_1, \dots, x_{t-1}, x_t, x_{t+1}, \dots, x_T$ : the word vectors corresponding to a corpus with  $T$  words.
- $h_t = \sigma(W^{(hh)}h_{t-1} + W^{(hx)}x_t)$ : the relationship to compute the hidden layer output features at each time-step  $t$ 
  - $x_t \in \mathbb{R}^d$ : input word vector at time  $t$ .
  - $W^{hx} \in \mathbb{R}^{D_h \times d}$ : weights matrix used to condition the input word vector,  $x_t$
  - $W^{hh} \in \mathbb{R}^{D_h \times D_h}$ : weights matrix used to condition the output of the previous time-step,  $h_{t-1}$
  - $h_{t-1} \in \mathbb{R}^{D_h}$ : output of the non-linear function at the previous time-step,  $t - 1$ .  $h_0 \in \mathbb{R}^{D_h}$  is an initialization vector for the hidden layer at time-step  $t = 0$ .
  - $\sigma()$ : the non-linearity function (sigmoid here)
- $\hat{y}_t = \text{softmax}(W^{(S)}h_t)$ : the output probability distribution over the vocabulary at each time-step  $t$ . Essentially,  $\hat{y}_t$  is the next predicted word given the document context score so far (i.e.  $h_{t-1}$ ) and the last observed word vector  $x^{(t)}$ . Here,  $W^{(S)} \in \mathbb{R}^{|V| \times D_h}$  and  $\hat{y} \in \mathbb{R}^{|V|}$  where  $|V|$  is the vocabulary.

# Sequence models - new

Consider Equations 5 and 6 at a time-step  $t$ ; to compute the RNN error,  $dE/dW$ , we sum the error at each time-step. That is,  $dE_t/dW$  for every time-step,  $t$ , is computed and accumulated.

$$\frac{\partial E}{\partial W} = \sum_{t=1}^T \frac{\partial E_t}{\partial W} \quad (10)$$

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^t \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W} \quad (11)$$

$$\frac{\partial E}{\partial W} = \sum_{t=1}^T \sum_{k=1}^t \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \left( \prod_{j=k+1}^t \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial W} \quad (14)$$

```

 $\hat{g} \leftarrow \frac{\partial E}{\partial W}$ 
if  $\|\hat{g}\| \geq \text{threshold}$  then
     $\hat{g} \leftarrow \frac{\text{threshold}}{\|\hat{g}\|} \hat{g}$ 
end if

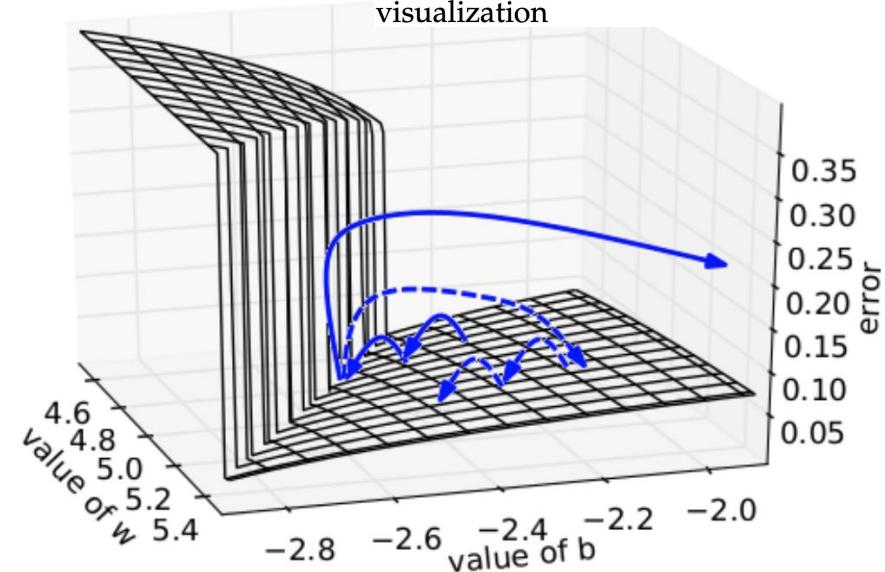
```

Algorithm 1: Pseudo-code for norm clipping in the gradients whenever they explode

$$\left\| \frac{\partial h_t}{\partial h_k} \right\| = \left\| \prod_{j=k+1}^t \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq (\beta_W \beta_h)^{t-k} \quad (16)$$

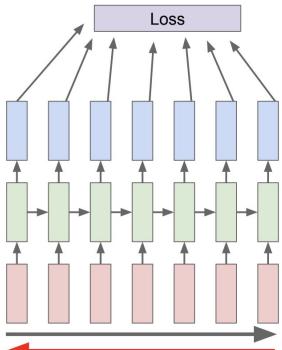
The exponential term  $(\beta_W \beta_h)^{t-k}$  can easily become a very small or large number when  $\beta_W \beta_h$  is much smaller or larger than 1 and  $t - k$  is sufficiently large. Recall a large  $t - k$  evaluates the cross entropy error due to faraway words. The contribution of faraway words to predicting the next word at time-step  $t$  diminishes when the gradient vanishes early on.

Figure 7: Gradient explosion clipping visualization



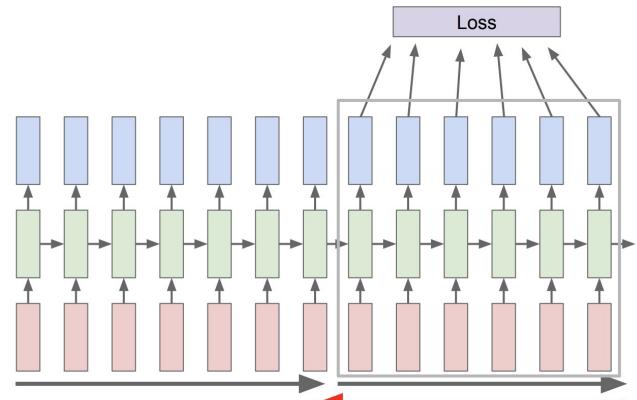
# Sequence models - new

## Truncated Backpropagation through time



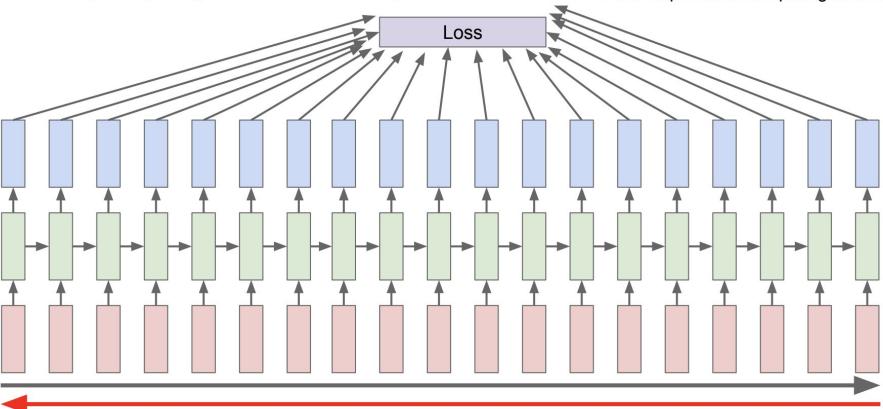
Run forward and backward through chunks of the sequence instead of whole sequence

## Truncated Backpropagation through time



Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

## Backpropagation through time



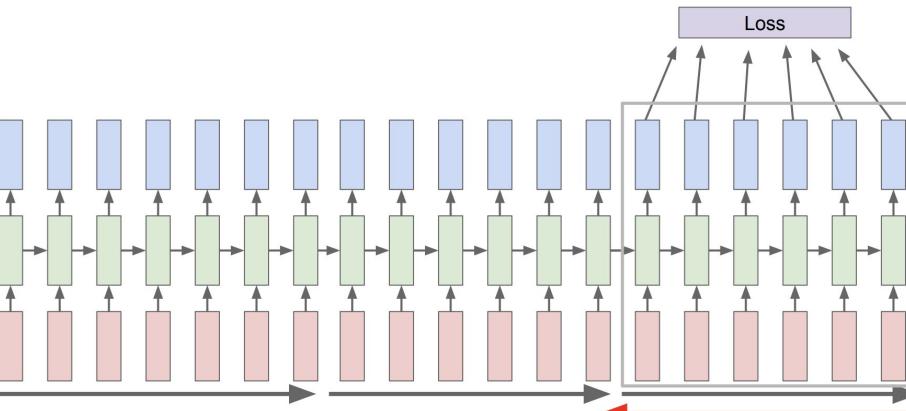
Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient

Fei-Fei Li, Ranjay Krishna, Danfei Xu

Lecture 10 - 47

May 7, 2020

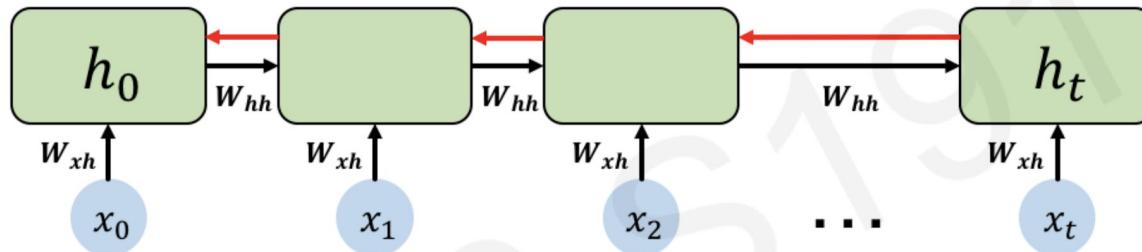
## Truncated Backpropagation through time



Source: [http://cs231n.stanford.edu/slides/2020/lecture\\_10.pdf](http://cs231n.stanford.edu/slides/2020/lecture_10.pdf)

# Sequence models

## Standard RNN Gradient Flow: Exploding Gradients



Computing the gradient wrt  $h_0$  involves **many factors of  $W_{hh}$**  + repeated gradient computation!

Many values  $> 1$ :  
**exploding gradients**

Gradient clipping to  
scale big gradients

Many values  $< 1$ :  
**vanishing gradients**

1. Activation function
2. Weight initialization
3. Network architecture

# Sequence models - new

## The Problem of Long-Term Dependencies

### A Simple Way to Initialize Recurrent Networks of Rectified Linear Units

Quoc V. Le, Navdeep Jaitly, Geoffrey E. Hinton  
Google

### 2 The initialization trick

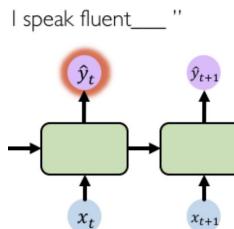
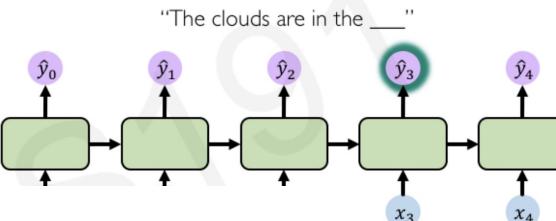
In this paper, we demonstrate that, with the right initialization of the weights, RNNs composed of rectified linear units are relatively easy to train and are good at modeling long-range dependencies. The RNNs are trained by using backpropagation through time to get error-derivatives for the weights and by updating the weights after each small mini-batch of sequences. Their performance on test data is comparable with LSTMs, both for toy problems involving very long-range temporal structures and for real tasks like predicting the next word in a very large corpus of text.

We initialize the recurrent weight matrix to be the **identity** matrix and biases to be zero. This means that each new hidden state vector is obtained by simply copying the previous hidden vector then adding on the effect of the current inputs and replacing all negative states by zero. In the absence of input, an RNN that is composed of ReLUs and initialized with the identity matrix (which we call an IRNN) just stays in the same state indefinitely. The identity initialization has the very desirable property that when the error derivatives for the hidden units are backpropagated through time they remain constant provided no extra error-derivatives are added. This is the same behavior as LSTMs when their forget gates are set so that there is no decay and it makes it easy to learn very long-range temporal dependencies.

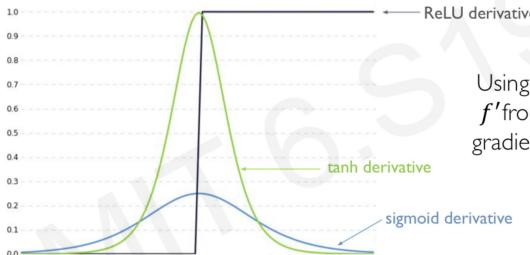
We also find that for tasks that exhibit less long range dependencies, scaling the identity matrix by a small scalar is an effective mechanism to forget long range effects. This is the same behavior as LSTMs when their forget gates are set so that the memory decays fast.

Source: [http://introtodeeplearning.com/slides/6S191\\_MIT\\_DeepLearning\\_L2.pdf](http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L2.pdf)  
<https://arxiv.org/pdf/1504.00941.pdf>

### Trick #1: Activation Functions



H. Suresh, 6.S191 2018. 1/27/20



Using ReLU prevents  $f'$  from shrinking the gradients when  $x > 0$

### Trick #2: Parameter Initialization

Initialize **weights** to identity matrix

$$I_n = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

Initialize **biases** to zero

This helps prevent the weights from shrinking to zero.

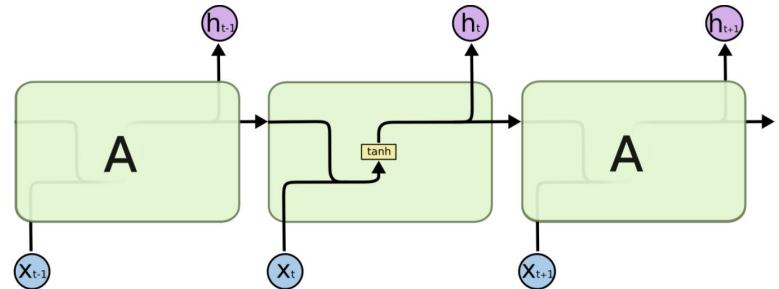
### Solution #3: Gated Cells

Idea: use a more **complex recurrent unit with gates** to control what information is passed through

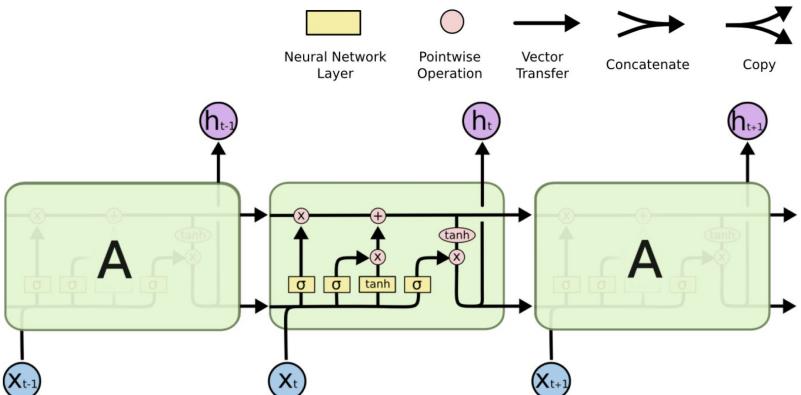
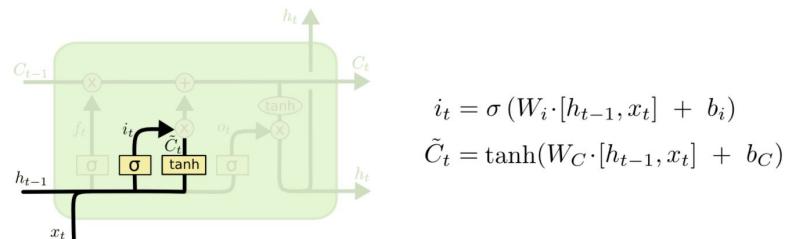
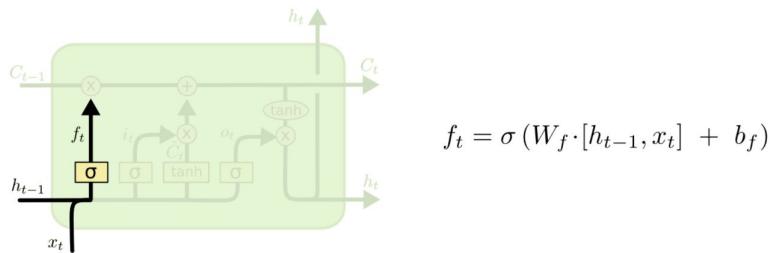
**gated cell**  
LSTM, GRU, etc.

Long Short Term Memory (LSTMs) networks rely on a gated cell to track information throughout many time steps.

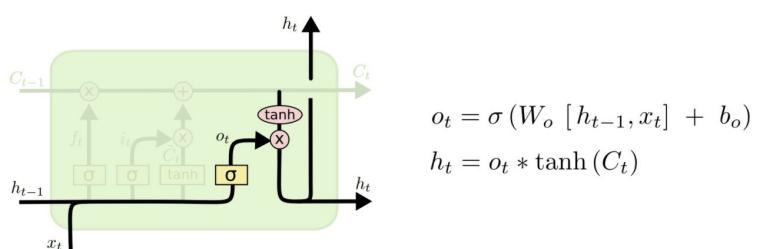
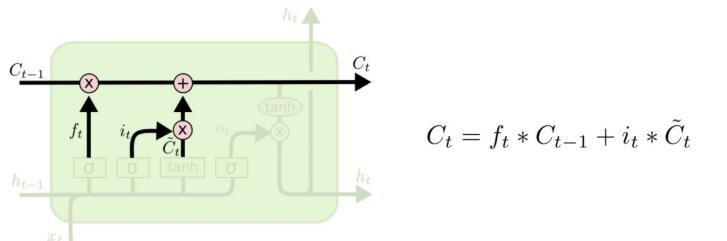
# Sequence models



The repeating module in a standard RNN contains a single layer.



The repeating module in an LSTM contains four interacting layers.



Source: <https://colah.github.io/posts/2015-08-Understanding-LSTMs>

# Sequence models - new

## Long Short-Term Memory (LSTM)

We have a sequence of inputs  $x^{(t)}$ , and we will compute a sequence of hidden states  $h^{(t)}$  and cell states  $c^{(t)}$ . On timestep  $t$ :

**Forget gate:** controls what is kept vs forgotten, from previous cell state

**Input gate:** controls what parts of the new cell content are written to cell

**Output gate:** controls what parts of cell are output to hidden state

**New cell content:** this is the new content to be written to the cell

**Cell state:** erase ("forget") some content from last cell state, and write ("input") some new cell content

**Hidden state:** read ("output") some content from the cell

**Sigmoid function:** all gate values are between 0 and 1

$$f^{(t)} = \sigma(W_f h^{(t-1)} + U_f x^{(t)} + b_f)$$

$$i^{(t)} = \sigma(W_i h^{(t-1)} + U_i x^{(t)} + b_i)$$

$$o^{(t)} = \sigma(W_o h^{(t-1)} + U_o x^{(t)} + b_o)$$

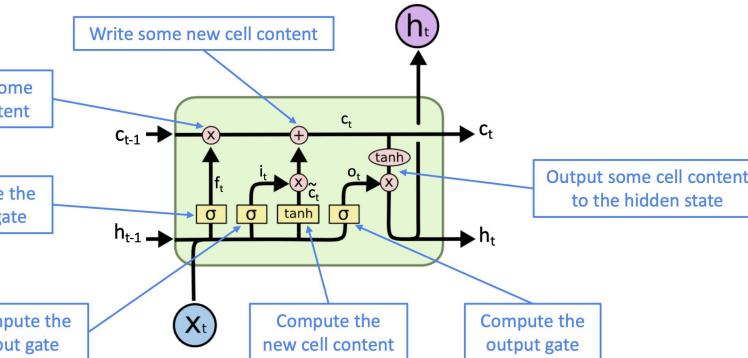
$$\tilde{c}^{(t)} = \tanh(W_c h^{(t-1)} + U_c x^{(t)} + b_c)$$

$$c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)}$$

$$h^{(t)} = o^{(t)} \circ \tanh c^{(t)}$$

Gates are applied using element-wise product

All these are vectors of same length  $n$



# Sequence models - new

## Gated Recurrent Units (GRU)

- Proposed by Cho et al. in 2014 as a simpler alternative to the LSTM.
- On each timestep  $t$  we have input  $x^{(t)}$  and hidden state  $h^{(t)}$  (no cell state).

Update gate: controls what parts of hidden state are updated vs preserved

$$u^{(t)} = \sigma(W_u h^{(t-1)} + U_u x^{(t)} + b_u)$$

Reset gate: controls what parts of previous hidden state are used to compute new content

$$r^{(t)} = \sigma(W_r h^{(t-1)} + U_r x^{(t)} + b_r)$$

New hidden state content: reset gate selects useful parts of prev hidden state. Use this and current input to compute new hidden content.

$$\tilde{h}^{(t)} = \tanh(W_h(r^{(t)} \circ h^{(t-1)}) + U_h x^{(t)} + b_h)$$

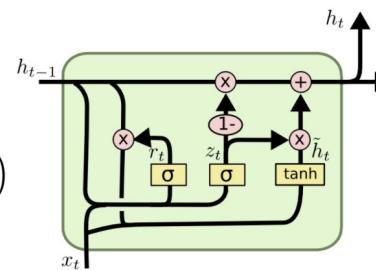
Hidden state: update gate simultaneously controls what is kept from previous hidden state, and what is updated to new hidden state content

$$h^{(t)} = (1 - u^{(t)}) \circ h^{(t-1)} + u^{(t)} \circ \tilde{h}^{(t)}$$

**How does this solve vanishing gradient?**

Like LSTM, GRU makes it easier to retain info long-term (e.g. by setting update gate to 0)

A slightly more dramatic variation on the LSTM is the Gated Recurrent Unit, or GRU, introduced by Cho, et al. (2014). It combines the forget and input gates into a single “update gate.” It also merges the cell state and hidden state, and makes some other changes. The resulting model is simpler than standard LSTM models, and has been growing increasingly popular.



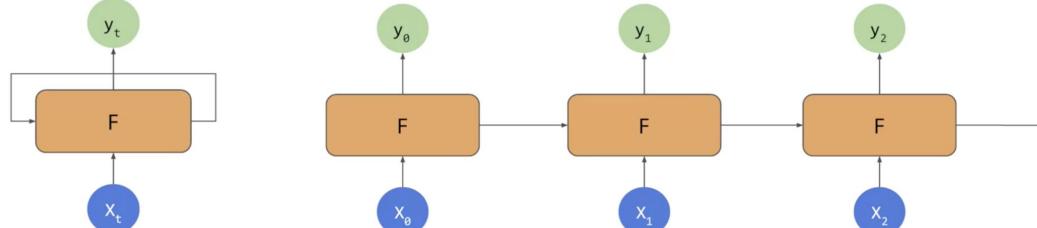
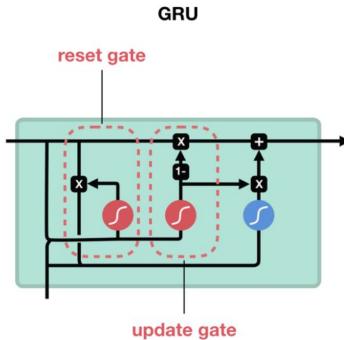
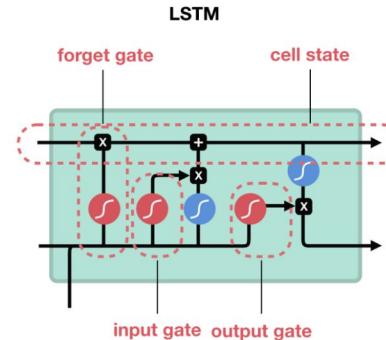
$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

# Sequence models

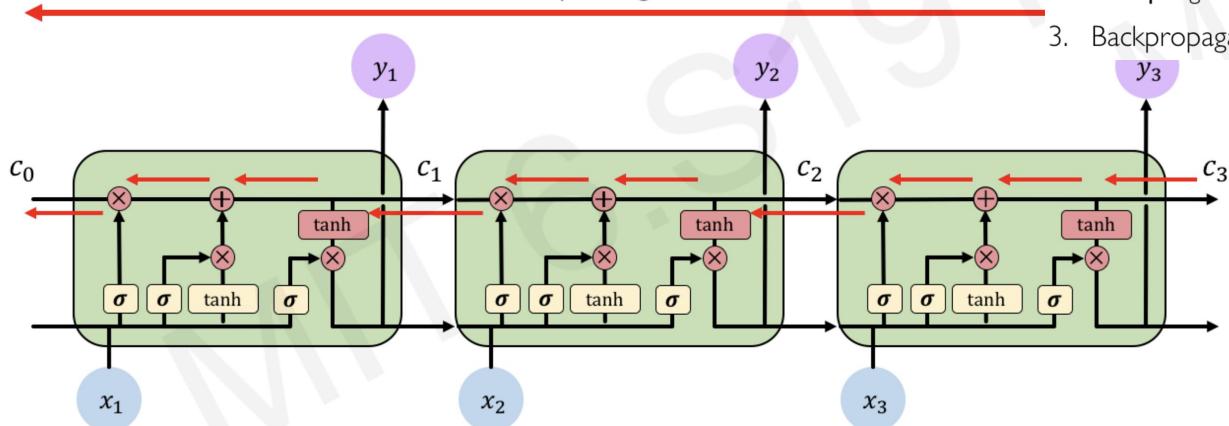


Source: <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>  
<https://www.coursera.org/learn/natural-language-processing-tensorflow/lecture/09WN5/introduction>

# Sequence models

## LSTM Gradient Flow

Uninterrupted gradient flow!



1. Maintain a **separate cell state** from what is outputted

2. Use **gates** to control the **flow of information**

- **Forget** gate gets rid of irrelevant information

- **Store** relevant information from current input

- Selectively **update** cell state

- **Output** gate returns a filtered version of the cell state

3. Backpropagation through time with **uninterrupted gradient flow**

# Sequence models - new

## Is vanishing/exploding gradient just a RNN problem?

- No! It can be a problem for all neural architectures (including feed-forward and convolutional), especially deep ones.
  - Due to chain rule / choice of nonlinearity function, gradient can become vanishingly small as it backpropagates
  - Thus lower layers are learnt very slowly (hard to train)
  - Solution: lots of new deep feedforward/convolutional architectures that add more direct connections (thus allowing the gradient to flow)



For example: For example: For example:

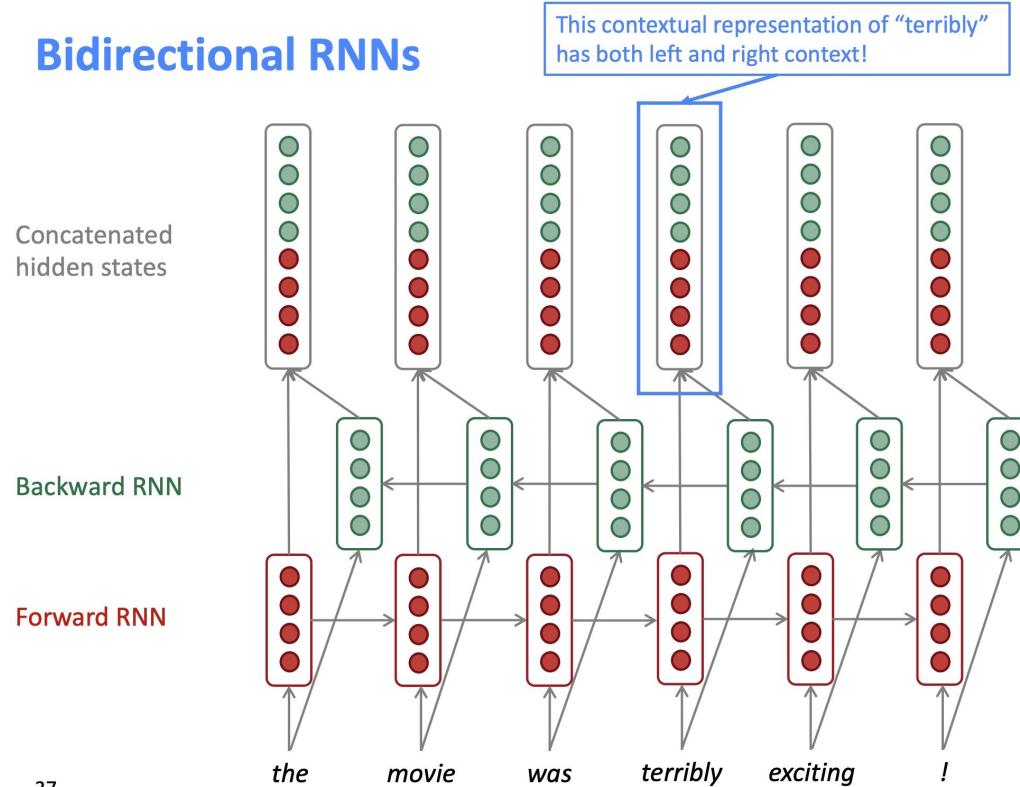
- Residual cc
- Dense con
- Highway co
- Also knowi
- Directly cc
- Similar to re
- The identit
- to all futu
- preserves i
- Inspired by
- This makes
- easier to tr

- Conclusion: Though vanishing/exploding gradients are a general problem, RNNs are particularly unstable due to the repeated multiplication by the same weight matrix [Bengio et al, 1994]

"Learning Long-Term Dependencies with Gradient Descent is Difficult", Bengio et al. 1994, <http://ai.dinfo.unifi.it/paolo//ps/tnn-94-gradient.pdf>

# Sequence models - new

## Bidirectional RNNs



On timestep  $t$ :

This is a general notation to mean “compute one forward step of the RNN” – it could be a vanilla, LSTM or GRU computation.

$$\begin{aligned}\vec{h}^{(t)} &= \text{RNN}_{\text{FW}}(\vec{h}^{(t-1)}, \mathbf{x}^{(t)}) \\ \overleftarrow{h}^{(t)} &= \text{RNN}_{\text{BW}}(\overleftarrow{h}^{(t+1)}, \mathbf{x}^{(t)})\end{aligned}$$

Generally, these two RNNs have separate weights

Concatenated hidden states

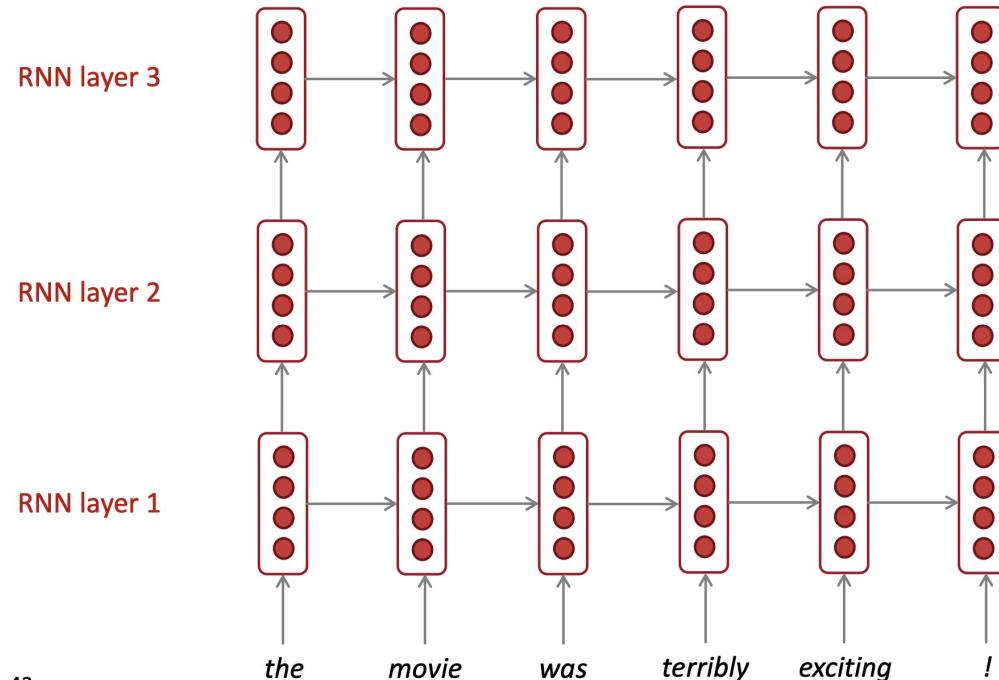
$$\mathbf{h}^{(t)} = [\vec{h}^{(t)}; \overleftarrow{h}^{(t)}]$$

We regard this as “the hidden state” of a bidirectional RNN. This is what we pass on to the next parts of the network.

- Note: bidirectional RNNs are only applicable if you have access to the **entire input sequence**.
  - They are **not** applicable to Language Modeling, because in LM you *only* have left context available.
- If you do have entire input sequence (e.g. any kind of encoding), **bidirectionality is powerful** (you should use it by default).
- For example, **BERT** (**Bidirectional** Encoder Representations from Transformers) is a powerful pretrained contextual representation system **built on bidirectionality**.

# Sequence models - new

## Multi-layer RNNs



42

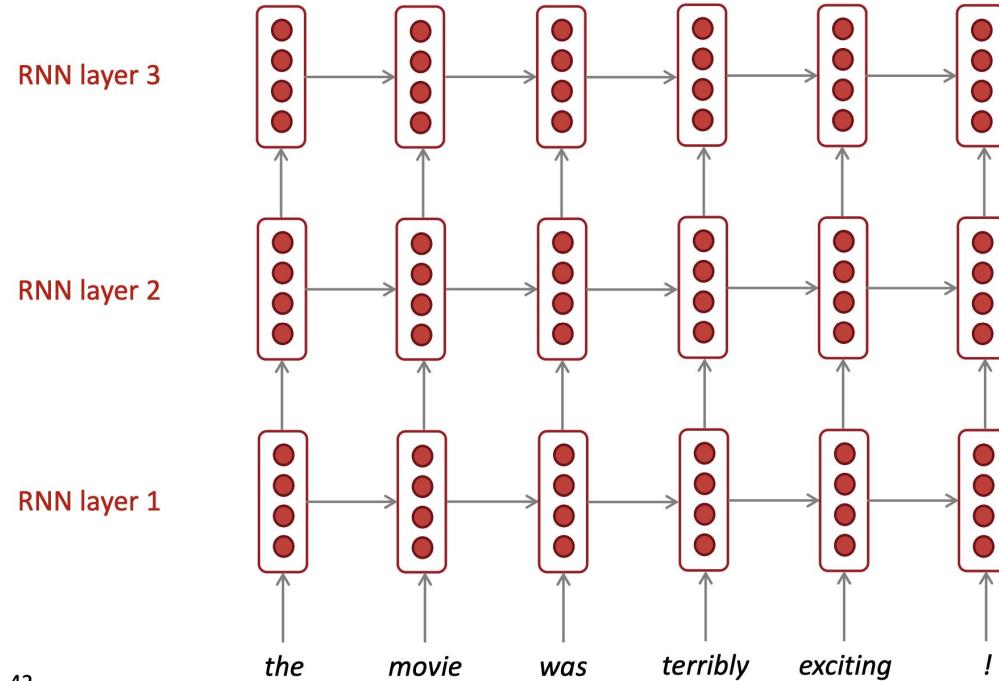
Source: <http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture07-fancy-rnn.pdf>  
<http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture06-rnnlm.pdf>

- RNNs are already “deep” on one dimension (they unroll over many timesteps)
- We can also make them “deep” in another dimension by applying multiple RNNs – this is a multi-layer RNN.
- This allows the network to compute **more complex representations**
  - The **lower RNNs** should compute **lower-level features** and the **higher RNNs** should compute **higher-level features**.
- Multi-layer RNNs are also called *stacked RNNs*.
- High-performing RNNs are often multi-layer (but aren’t as deep as convolutional or feed-forward networks)
- For example: In a 2017 paper, Britz et al find that for Neural Machine Translation, **2 to 4 layers** is best for the encoder RNN, and **4 layers** is best for the decoder RNN
  - However, **skip-connections/dense-connections** are needed to train deeper RNNs (e.g. 8 layers)
- **Transformer-based** networks (e.g. BERT) are frequently deeper, like **12 or 24 layers**.
  - You will learn about Transformers later; they have a lot of skipping-like connections

# Sequence models - new

## Multi-layer RNNs

The hidden states from RNN layer  $i$   
are the inputs to RNN layer  $i+1$



### A note on terminology

The RNN described in this lecture = simple/vanilla/Elman RNN

**Next lecture:** You will learn about other RNN flavors

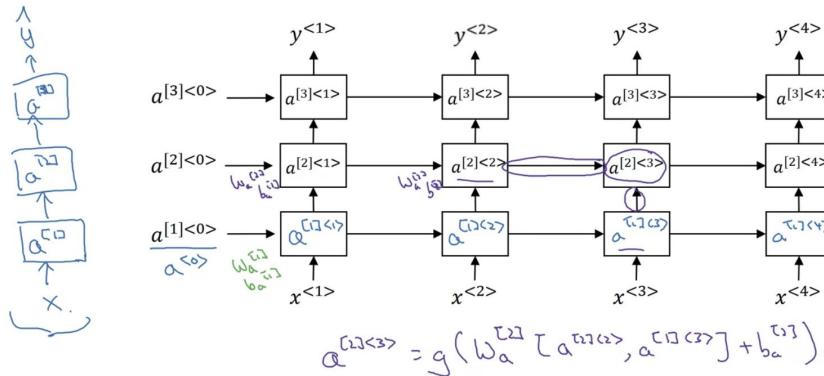
like GRU and LSTM and multi-layer RNNs



**By the end of the course:** You will understand phrases like  
"stacked bidirectional LSTM with residual connections and self-attention"

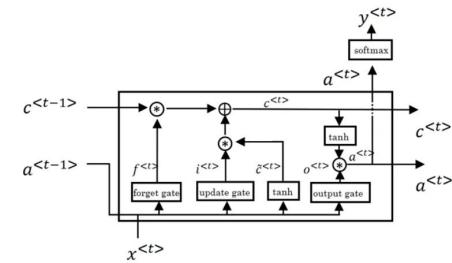
# Sequence models

## Deep RNN example



## LSTM in pictures

$$\begin{aligned} \tilde{c}^{<t>} &= \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c) \\ \Gamma_u &= \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \\ \Gamma_f &= \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \\ \Gamma_o &= \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \\ c^{<t>} &= \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>} \\ a^{<t>} &= \Gamma_o * \tanh c^{<t>} \end{aligned}$$



**Sequence Models**  
deeplearning.ai  
4.8 (20,428 ratings) | 180K Students Enrolled

Enroll for Free



**Sequence Models**  
deeplearning.ai  
4.8 (20,446 ratings) | 180K Students Enrolled

Enroll for Free

# Sequence models - new

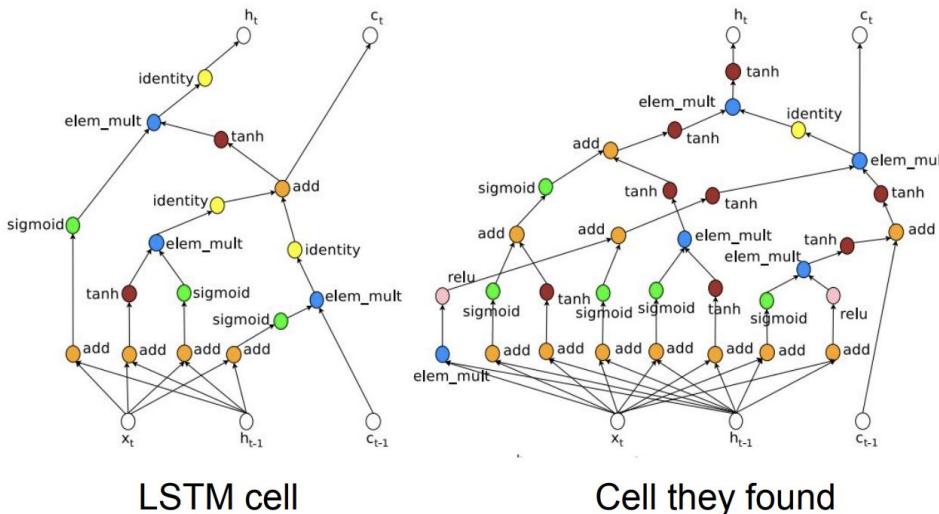
LSTM

Neural Architecture Search for RNN architectures

ion of  
lectures

Klaus Greff, Rupesh

This paper re  
variants of the  
most commonly  
performs reasona  
investigated mod  
However, certain  
forget gates (CI)  
simplified LSTM  
decreasing perfo  
because they r  
computational co



Zoph et Le, "Neural Architecture Search with Reinforcement Learning", ICLR 2017  
Figures copyright Zoph et al. 2017. Reproduced with permission.

Fei-Fei Li, Ranjay Krishna, Danfei Xu

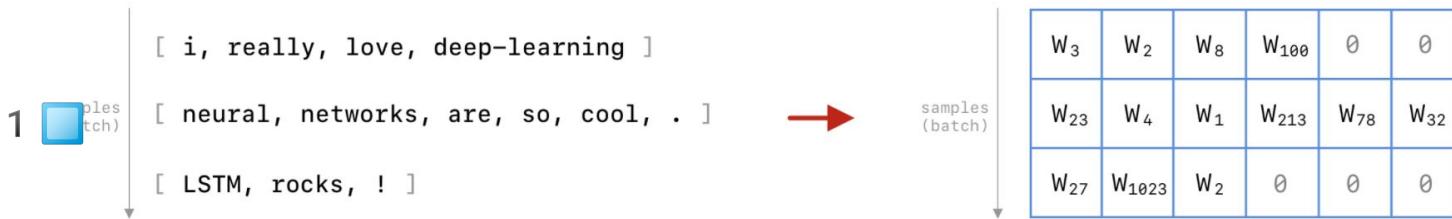
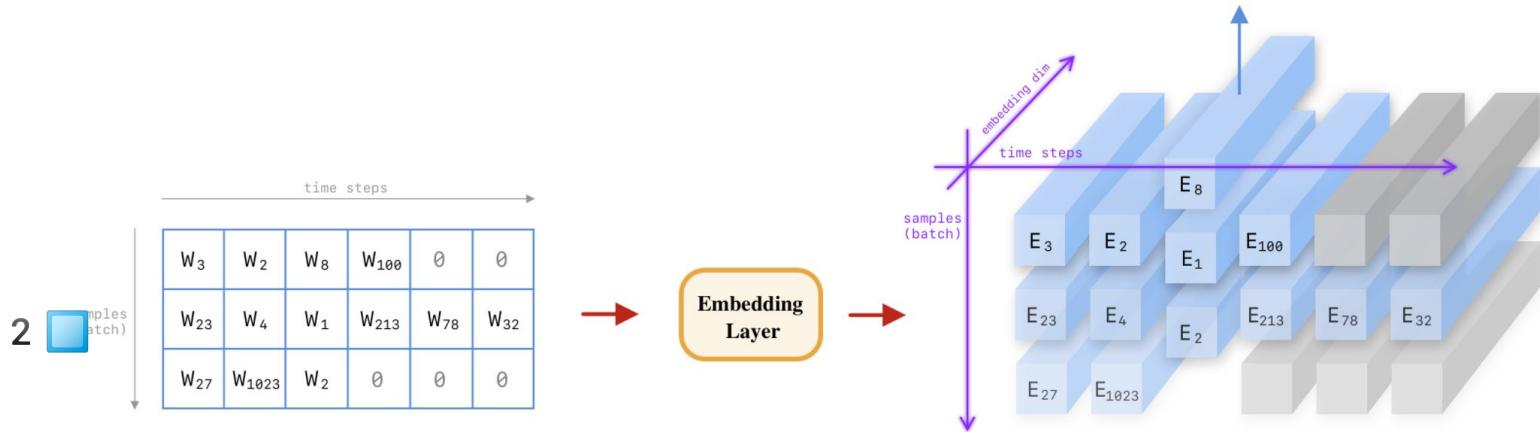
Source: <https://www.youtube.com/watch?v=6niqTuYFZLQ>  
<https://arxiv.org/abs/1503.04069>  
<http://proceedings.mlr.press/v37/jozefowicz15.pdf>  
[http://cs231n.stanford.edu/slides/2020/lecture\\_10.pdf](http://cs231n.stanford.edu/slides/2020/lecture_10.pdf)

Lecture 10 - 123

May 7, 2020

ture failed to dramatically improve over the LSTM suggests that, at the very least, if there are architectures that are much better than the LSTM, then they are not trivial to find.

# Sequence models - new



Source: [https://iust-deep-learning.github.io/972/static\\_files/assignments/assignment\\_05\\_preview.html](https://iust-deep-learning.github.io/972/static_files/assignments/assignment_05_preview.html)

# Sequence models - new

```
dataset, info = tfds.load('imdb_reviews/subwords8k', with_info=True, as_supervised=True)
train_dataset, test_dataset = dataset['train'], dataset['test']
```

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),

    tf.keras.layers.Embedding(
        input_dim, output_dim, embeddings_initializer='uniform',
        embeddings_regularizer=None, activity_regularizer=None,
        embeddings_constraint=None, mask_zero=False, input_length=None, **kwargs
    )

model = Sequential()
model.add(Embedding(1000, 64, input_length=10))
# the model will take as input an integer matrix of size (batch,
# input_length).
# the largest integer (i.e. word index) in the input should be no larger
# than 999 (vocabulary size).
# now model.output_shape == (None, 10, 64), where None is the batch
# dimension.

input_array = np.random.randint(1000, size=(32, 10))

model.compile('rmsprop', 'mse')
output_array = model.predict(input_array)
assert output_array.shape == (32, 10, 64)
```

TensorFlow > API > TensorFlow Core v2.2.0 > Python

## tf.keras.layers.Embedding

Turns positive integers (indexes) into dense vectors of fixed size.

Inherits From: [Layer](#)

**input\_dim** int > 0. Size of the vocabulary, i.e. maximum integer index + 1.

**output\_dim** int  $\geq 0$ . Dimension of the dense embedding.

**Input shape:**

2D tensor with shape: `(batch_size, input_length)`.

**Output shape:**

3D tensor with shape: `(batch_size, input_length, output_dim)`.

Source: [https://www.tensorflow.org/api\\_docs/python/tf/keras/layers/Embedding](https://www.tensorflow.org/api_docs/python/tf/keras/layers/Embedding)

# Sequence models - new

## tf.keras.layers.Flatten

Inherits From: [Layer](#)

```
dataset, info = tfds.load('imdb_reviews/subwords8k', with_info=True, as_supervised=True)
train_dataset, test_dataset = dataset['train'], dataset['test']
```

```
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()
```

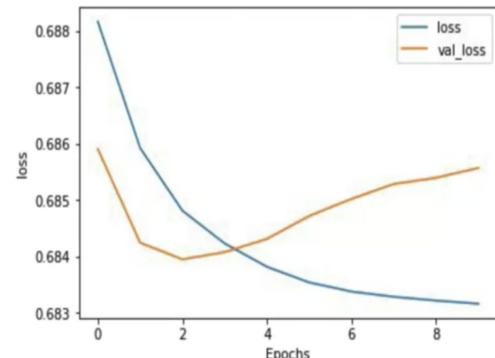
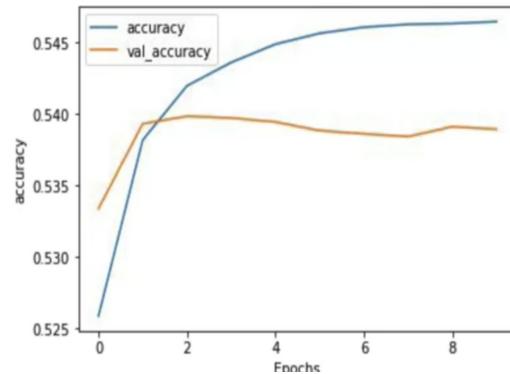
Model: "sequential"

| Layer (type)          | Output Shape    | Param # |
|-----------------------|-----------------|---------|
| <hr/>                 |                 |         |
| embedding (Embedding) | (None, 120, 16) | 160000  |
| flatten (Flatten)     | (None, 1920)    | 0       |
| dense (Dense)         | (None, 6)       | 11526   |
| dense_1 (Dense)       | (None, 1)       | 7       |
| <hr/>                 |                 |         |

Total params: 171,533

Trainable params: 171,533

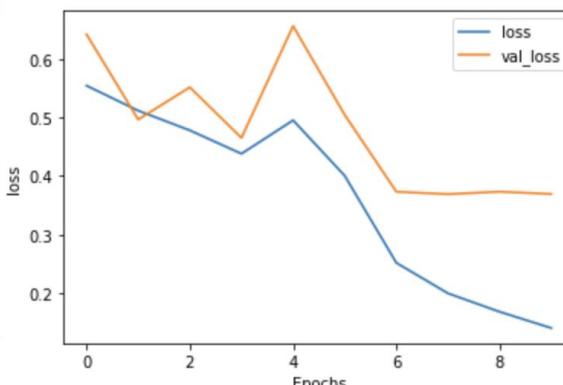
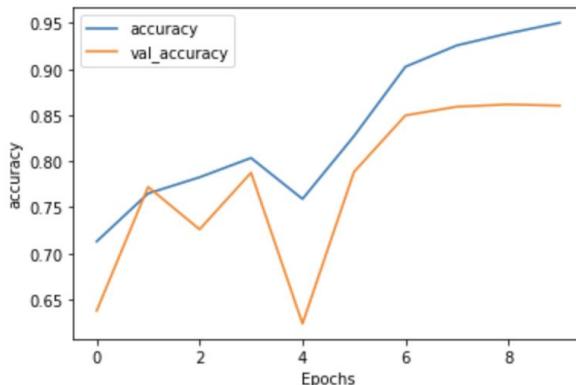
Non-trainable params: 0



# Sequence models

```
dataset, info = tfds.load('imdb_reviews/subwords8k', with_info=True, as_supervised=True)
train_dataset, test_dataset = dataset['train'], dataset['test']
```

```
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.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
NUM_EPOCHS = 10
history = model.fit(train_dataset, epochs=NUM_EPOCHS, validation_data=test_dataset)
```



Model: "sequential"

| Layer (type)                  | Output Shape     | Param # |
|-------------------------------|------------------|---------|
| embedding (Embedding)         | (None, None, 64) | 523840  |
| bidirectional (Bidirectional) | (None, 128)      | 66048   |
| dense (Dense)                 | (None, 64)       | 8256    |
| dense_1 (Dense)               | (None, 1)        | 65      |

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

TensorFlow > API > TensorFlow Core v2.2.0 > Python

## tf.keras.layers.Bidirectional

```
tf.keras.layers.Bidirectional(  
    layer, merge_mode='concat', weights=None, backward_layer=None, **kwargs  
)
```

### merge\_mode

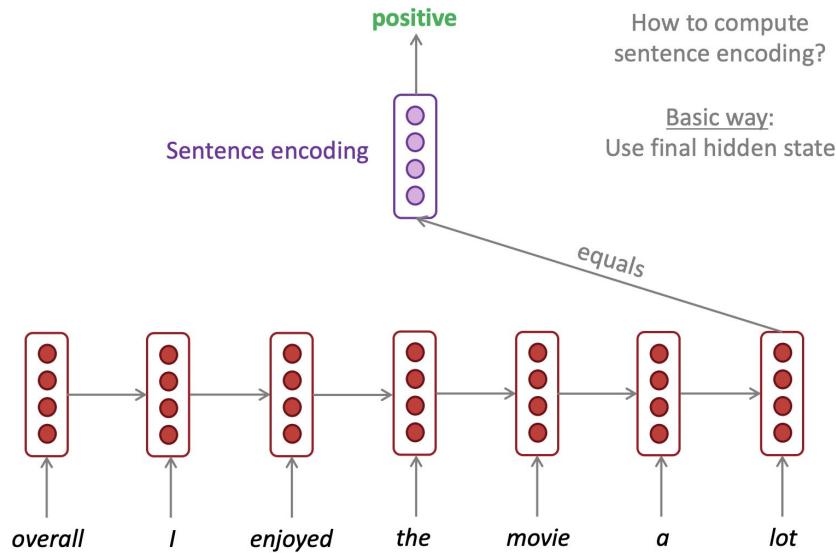
Mode by which outputs of the forward and backward RNNs will be combined. One of ('sum', 'mul', 'concat', 'ave', None). If None, the outputs will not be combined, they will be returned as a list. Default value is 'concat'.

# Sequence models - new

```
dataset, info = tfds.load('imdb_reviews/subwords8k', with_info=True, as_supervised=True)
train_dataset, test_dataset = dataset['train'], dataset['test']
```

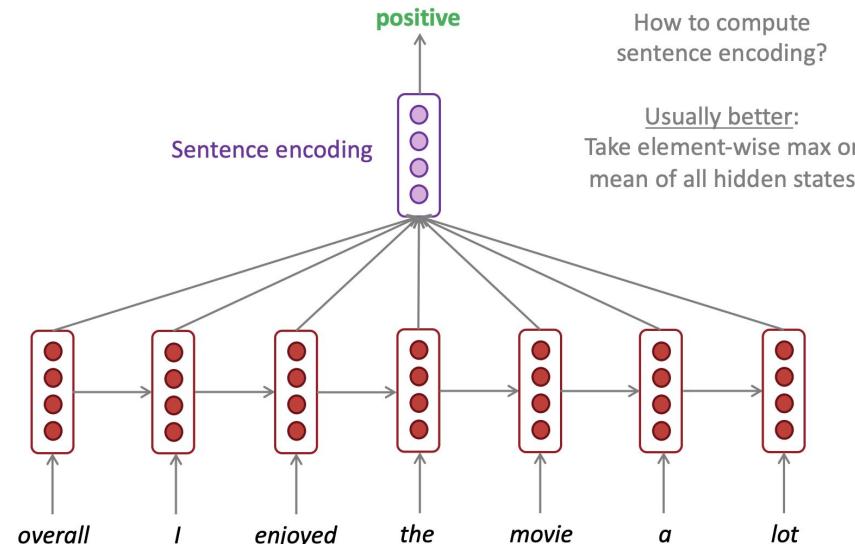
## RNNs can be used for sentence classification

e.g. sentiment classification



## RNNs can be used for sentence classification

e.g. sentiment classification



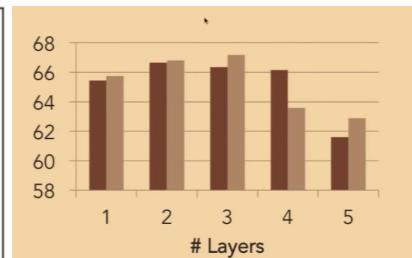
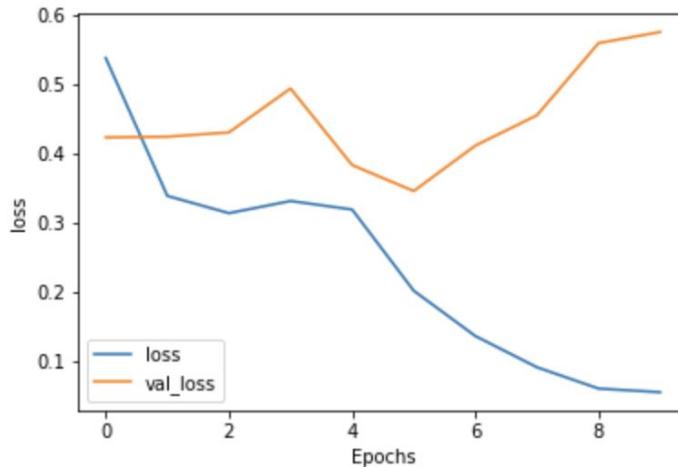
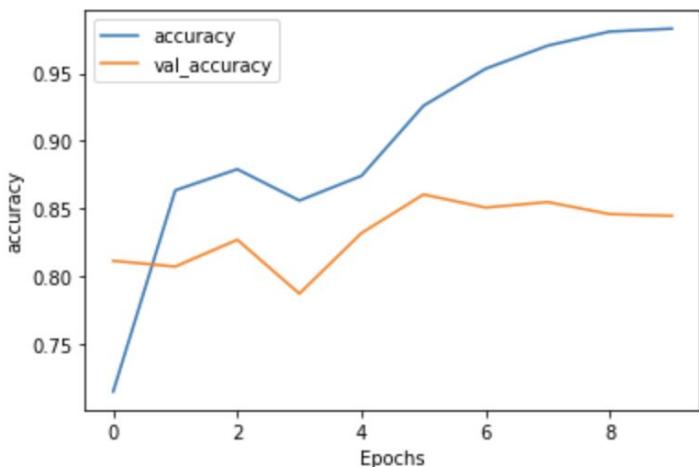
# Sequence models

```
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: "sequential\_1"

| Layer (type)                 | Output Shape      | Param # |
|------------------------------|-------------------|---------|
| embedding_1 (Embedding)      | (None, None, 64)  | 523840  |
| bidirectional_1 (Bidirection | (None, None, 128) | 66048   |
| bidirectional_2 (Bidirection | (None, 64)        | 41216   |
| dense_2 (Dense)              | (None, 64)        | 4160    |
| dense_3 (Dense)              | (None, 1)         | 65      |

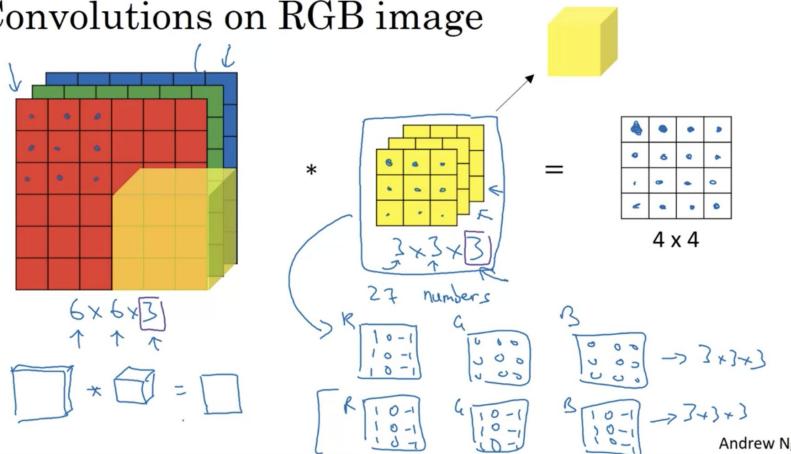
Total params: 635,329  
Trainable params: 635,329  
Non-trainable params: 0



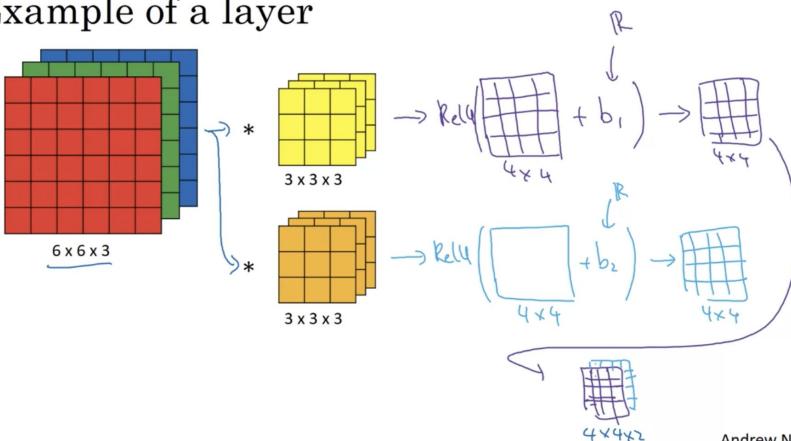
Source:  
[https://www.youtube.com/watch?v=Keqep\\_PKrY8](https://www.youtube.com/watch?v=Keqep_PKrY8)

# Reminder from Presentation-3 on CNNs

## Convolutions on RGB image



## Example of a layer



Andrew Ng

| Input Volume (+pad 1) (7x7x3)                                     | Filter W0 (3x3x3)   | Filter W1 (3x3x3)  | Output Volume (3x3x2)  |
|---|---|--|--|
| $x[:, :, 0]$  | $w0[:, :, 0]$   | $w1[:, :, 0]$  | $o[:, :, 0]$   |
| $\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{matrix}$ | $\begin{matrix} -1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & -1 & 1 \end{matrix}$ | $\begin{matrix} 0 & 1 & -1 \\ 0 & -1 & 0 \\ 0 & -1 & 1 \end{matrix}$ | $\begin{matrix} 2 & 3 & 3 \\ 3 & 7 & 3 \\ 8 & 10 & -3 \end{matrix}$    |
| $\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 0 \end{matrix}$ | $\begin{matrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 1 \end{matrix}$   | $\begin{matrix} -1 & 0 & 1 \\ 1 & -1 & 1 \\ 0 & 1 & 0 \end{matrix}$  | $\begin{matrix} o[:, :, 1] \\ -8 & -8 & -3 \\ 1 & -1 & 0 \end{matrix}$ |
| $\begin{matrix} 0 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 1 \end{matrix}$ | $\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}$   | $\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & -1 & 0 \end{matrix}$   | $\begin{matrix} -3 & 1 & 0 \\ -3 & -8 & -5 \end{matrix}$               |
| $\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}$ | $\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}$   | $\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & -1 & 0 \end{matrix}$   | $\begin{matrix} Bias b1 (1x1x1) \\ b1[:, :, 0] \\ 0 \end{matrix}$      |
| $x[:, :, 1]$  | $w0[:, :, 1]$   | $w1[:, :, 1]$  |  |
| $x[:, :, 2]$  | $w0[:, :, 2]$   | $w1[:, :, 2]$  |  |
| $x[:, :, 3]$  | $b0[:, :, 0]$   |  |  |

toggle movement

Source: <https://www.coursera.org/learn/convolutional-neural-networks>  
<https://images.app.goo.gl/Z4xDvji3Gcerpm27>

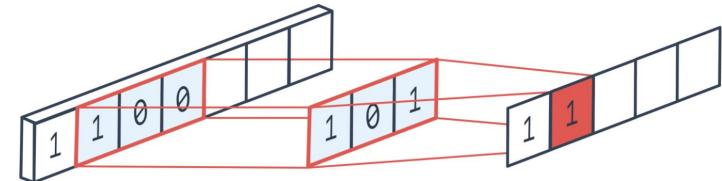
# Sequence models - new

TensorFlow > API > TensorFlow Core v2.2.0 > Python

## tf.keras.layers.Conv1D

1D convolution layer (e.g. temporal convolution).

```
tf.keras.layers.Conv1D(  
    filters, kernel_size, strides=1, padding='valid', data_format='channels_last',  
    dilation_rate=1, activation=None, use_bias=True,  
    kernel_initializer='glorot_uniform', bias_initializer='zeros',  
    kernel_regularizer=None, bias_regularizer=None, activity_regularizer=None,  
    kernel_constraint=None, bias_constraint=None, **kwargs  
)  
  
>>> # The inputs are 128-length vectors with 10 timesteps, and the batch size  
>>> # is 4.  
>>> input_shape = (4, 10, 128)  
>>> x = tf.random.normal(input_shape)  
>>> y = tf.keras.layers.Conv1D(  
... 32, 3, activation='relu', input_shape=input_shape)(x)  
>>> print(y.shape)  
(4, 8, 32)
```



Conv1d-Input1d Example [Image [12]]

filters

kernel\_size

strides

Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).

An integer or tuple/list of a single integer, specifying the length of the 1D convolution window.

An integer or tuple/list of a single integer, specifying the stride length of the convolution. Specifying any stride value != 1 is incompatible with specifying any `dilation_rate` value != 1.

Input shape:

3D tensor with shape: (batch\_size, steps, input\_dim)

Output shape:

3D tensor with shape: (batch\_size, new\_steps, filters) `steps` value might have changed due to padding or strides.

Source: [https://www.tensorflow.org/api\\_docs/python/tf/keras/layers/Conv1D](https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv1D)

<https://towardsdatascience.com/pytorch-basics-how-to-train-your-neural-net-intro-to-cnn-26a14c2ea29>

# Sequence models - new

TensorFlow > API > TensorFlow Core v2.2.0 > Python

## tf.keras.layers.AveragePooling1D

Average pooling for temporal data.

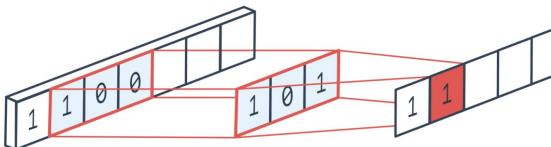
```
tf.keras.layers.AveragePooling1D(  
    pool_size=2, strides=None, padding='valid', data_format='channels_last',  
    **kwargs  
)
```

Input shape:

- If `data_format='channels_last'`: 3D tensor with shape `(batch_size, steps, features)`.
- If `data_format='channels_first'`: 3D tensor with shape `(batch_size, features, steps)`.

Output shape:

- If `data_format='channels_last'`: 3D tensor with shape `(batch_size, downsampled_steps, features)`.
- If `data_format='channels_first'`: 3D tensor with shape `(batch_size, features, downsampled_steps)`.



Source: [https://www.tensorflow.org/api\\_docs/python/tf/keras/layers/AveragePooling1D](https://www.tensorflow.org/api_docs/python/tf/keras/layers/AveragePooling1D)

[https://www.tensorflow.org/api\\_docs/python/tf/keras/layers/GlobalAveragePooling1D](https://www.tensorflow.org/api_docs/python/tf/keras/layers/GlobalAveragePooling1D)

<https://stackoverflow.com/questions/54493738/keras-difference-between-averagepooling1d-layer-and-globalaveragepooling1d-laye>

**GlobalAveragePooling1D** is same as **AveragePooling1D** with `pool_size=steps`

TensorFlow > API > TensorFlow Core v2.2.0 > Python

## tf.keras.layers.GlobalAveragePooling1D

Global average pooling operation for temporal data.

```
tf.keras.layers.GlobalAveragePooling1D(  
    data_format='channels_last', **kwargs  
)  
  
>>> input_shape = (2, 3, 4)  
>>> x = tf.random.normal(input_shape)  
>>> y = tf.keras.layers.GlobalAveragePooling1D()(x)  
>>> print(y.shape)  
(2, 4)
```

Input shape:

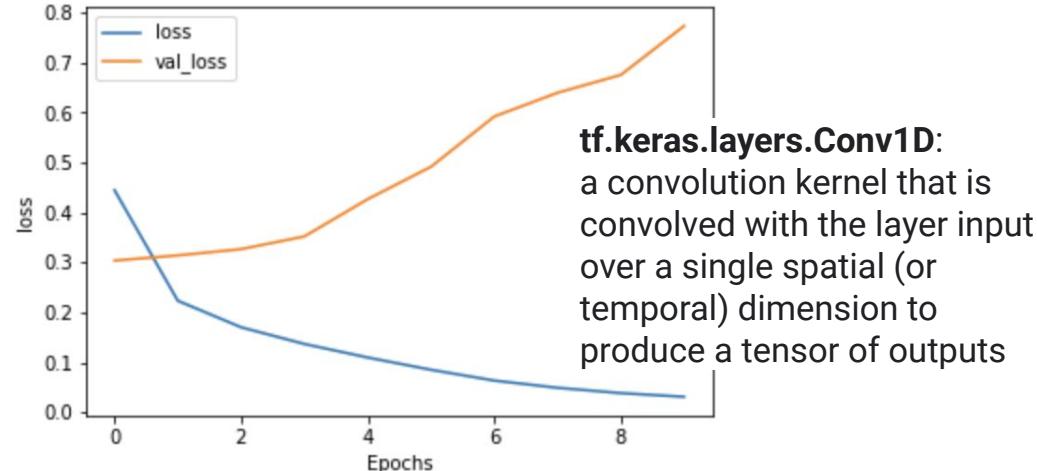
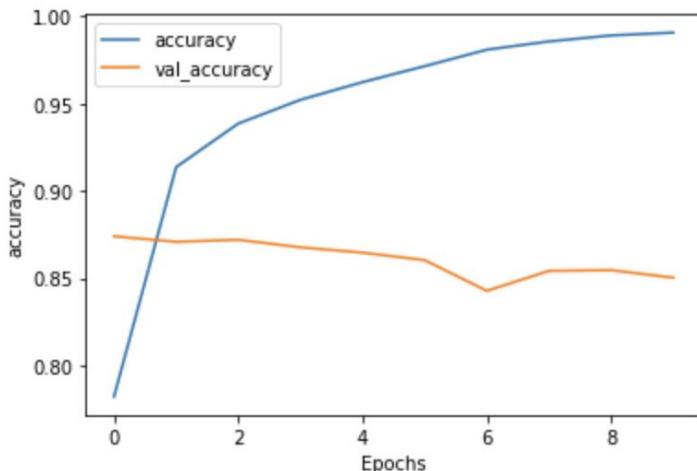
- If `data_format='channels_last'`: 3D tensor with shape: `(batch_size, steps, features)`
- If `data_format='channels_first'`: 3D tensor with shape: `(batch_size, features, steps)`

Output shape:

2D tensor with shape `(batch_size, features)`.

# Sequence models

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(tokenizer.vocab_size, 64),
    tf.keras.layers.Conv1D(128, 5, activation='relu'),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```



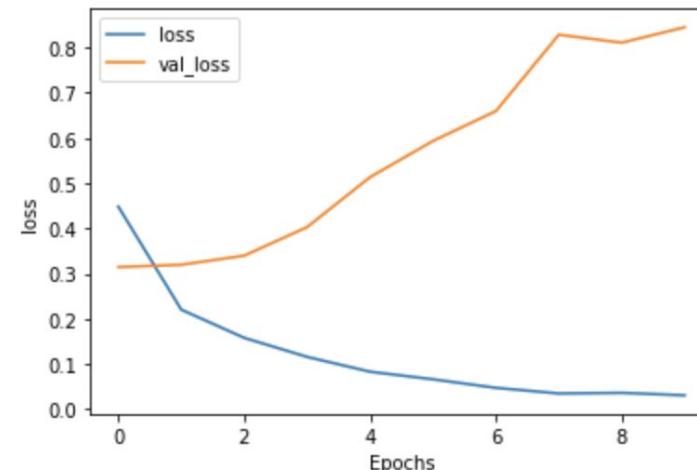
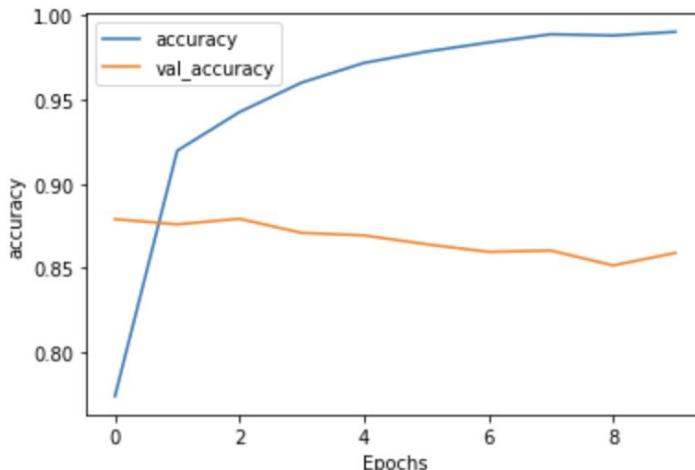
Model: "sequential\_2"

| Layer (type)                                      | Output Shape      | Param # |
|---|-------------------|---------|
| <hr/>   |                   |         |
| embedding_2 (Embedding)                           | (None, None, 64)  | 523840  |
| conv1d (Conv1D)                                   | (None, None, 128) | 41088   |
| global_average_pooling1d (GlobalAveragePooling1D) | (None, 128)       | 0       |
| dense_4 (Dense)                                   | (None, 64)        | 8256    |
| dense_5 (Dense)                                   | (None, 1)         | 65      |
| <hr/>   |                   |         |

Total params: 573,249  
Trainable params: 573,249  
Non-trainable params: 0

# Sequence models

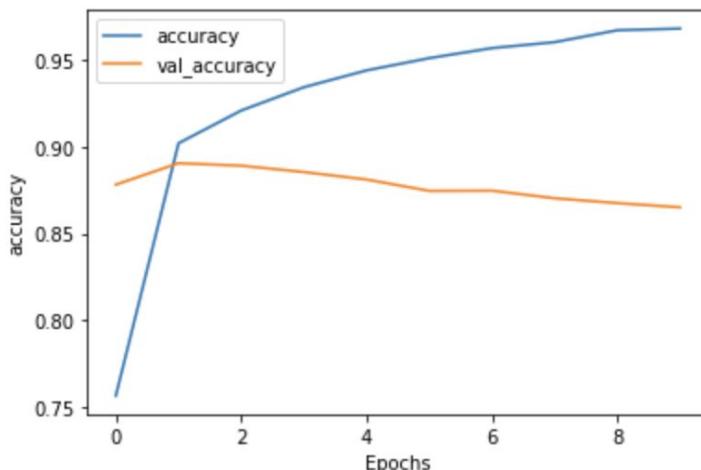
```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(tokenizer.vocab_size, 64),
    tf.keras.layers.Conv1D(128, 5, activation='relu'),
    tf.keras.layers.Conv1D(128, 3, activation='relu'),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```



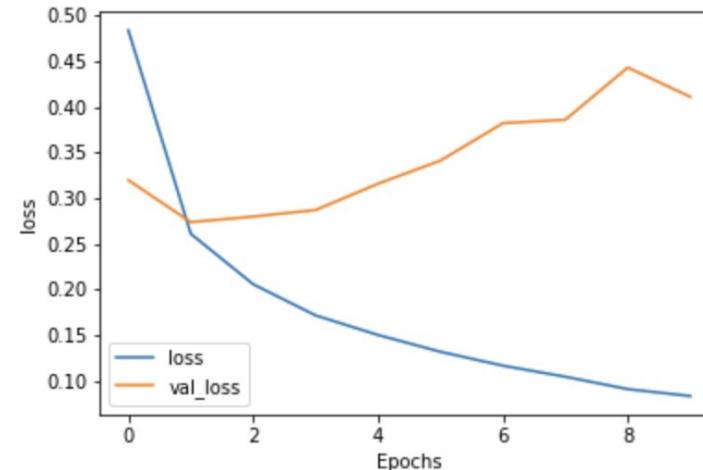
| Model: "sequential_3"   |                   |         |
|---|-------------------|---------|
| Layer (type)  | Output Shape      | Param # |
| embedding_3 (Embedding)   | (None, None, 64)  | 523840  |
| conv1d_1 (Conv1D)   | (None, None, 128) | 41088   |
| conv1d_2 (Conv1D)   | (None, None, 128) | 49280   |
| global_average_pooling1d_1 (GlobalAveragePooling1D)                           | (None, 128)       | 0       |
| dense_6 (Dense)   | (None, 64)        | 8256    |
| dense_7 (Dense)   | (None, 1)         | 65      |
| Total params: 622,529<br>Trainable params: 622,529<br>Non-trainable params: 0 |                   |         |

# Sequence models

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(tokenizer.vocab_size, 64),
    tf.keras.layers.Conv1D(128, 5, activation='relu'),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```



```
Model: "sequential_4"
Layer (type)          Output Shape         Param #
=====
embedding_4 (Embedding) (None, None, 64)      523840
conv1d_3 (Conv1D)       (None, None, 128)     41088
global_average_pooling1d_2 ( (None, 128)      0
dropout (Dropout)        (None, 128)      0
dense_8 (Dense)          (None, 64)       8256
dense_9 (Dense)          (None, 1)        65
=====
Total params: 573,249
Trainable params: 573,249
Non-trainable params: 0
```



# Sequence models

The screenshot shows the Kaggle interface for the Sentiment140 dataset. At the top, there's a banner with three large smiley faces (green, red, yellow) representing sentiment. Below the banner, the title "Sentiment140 dataset with 1.6 million tweets" is displayed, along with the subtitle "Sentiment analysis with tweets". On the left, there's a profile picture of a person and the text "Μαρίος Μιχαηλίδης Kazanova • updated 3 years ago (Version 2)". Below this, a navigation bar includes links for "Data", "Tasks", "Kernels (75)", "Discussion (7)", "Activity", and "Metadata". To the right of the navigation bar are buttons for "Download (228 MB)" and "New Notebook", followed by a vertical ellipsis menu. A blue arrow points from the right side of the slide towards the "New Notebook" button.

1. target: the polarity of the tweet ( $0 = \text{negative}$ ,  $2 = \text{neutral}$ ,  $4 = \text{positive}$ )
2. ids: The id of the tweet ( 2087)
3. date: the date of the tweet (Sat May 16 23:58:44 UTC 2009)
4. flag: The query (*llyx*). If there is no query, then this value is NO\_QUERY.
5. user: the user that tweeted (*robotickilldozr*)
6. text: the text of the tweet (*Lyx is cool*)

```
1600000
1600000
['0', "@alydesigns i was out most of the day so didn't get much done "]
['0', "one of my friend called me, and asked to meet with her at Mid Valley today...but i've no time *sigh* "]
['0', '@angry_barista I baked you a cake but I ated it ']
[[['0', "@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D"], ['0', "is upset that he can't upda
{0, 1}
138824
1
The Second Edition of the 20-volume Oxford English Dictionary, published in 1989, contains full entries for 171,476 words in current use
138825
```

# Sequence models

## GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning

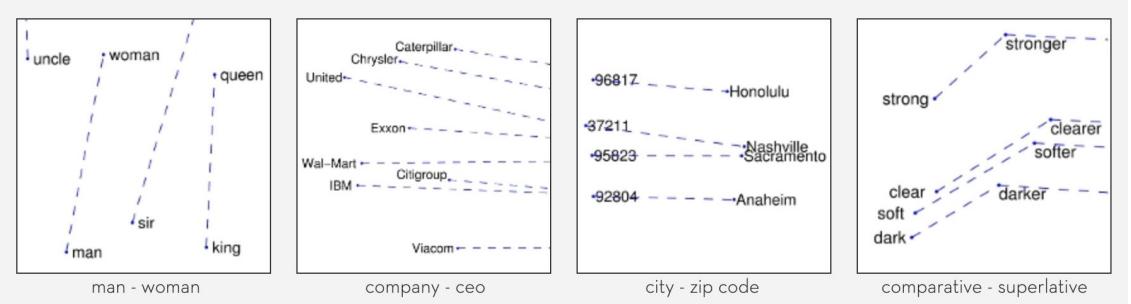
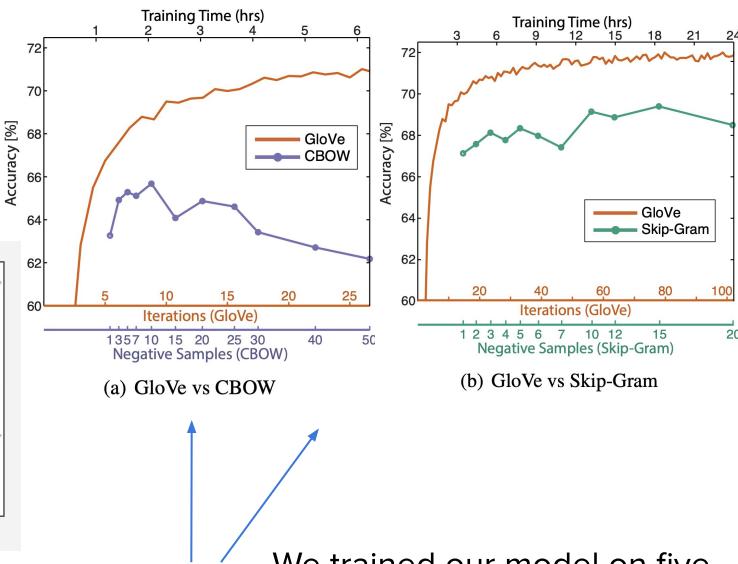


Figure 4: Overall accuracy on the word analogy task as a function of training time, which is governed by the number of iterations for GloVe and by the number of negative samples for CBOW (a) and skip-gram (b). In all cases, we train 300-dimensional vectors on the same 6B token corpus (Wikipedia 2014 + Gigaword 5) with the same 400,000 word vocabulary, and use a symmetric context window of size 10.

**GloVe** is an unsupervised learning algorithm for obtaining **vector** representations for **words**. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

Source: <https://nlp.stanford.edu/projects/glove>  
<https://nlp.stanford.edu/pubs/glove.pdf>

Tokenize,lowercase,Stanford tokenizer, build a vocabulary of the 400,000 most frequent words , and then construct a matrix of cooccurrence counts  $X$ .



We trained our model on five corpora of varying sizes: a 2010 Wikipedia dump with 1 billion tokens; a 2014 Wikipedia dump with 1.6 billion tokens; Gigaword 5 which has 4.3 billion tokens; the combination Gigaword5 + Wikipedia2014, which has 6 billion tokens; and on 42 billion tokens of web data, from Common Crawl.

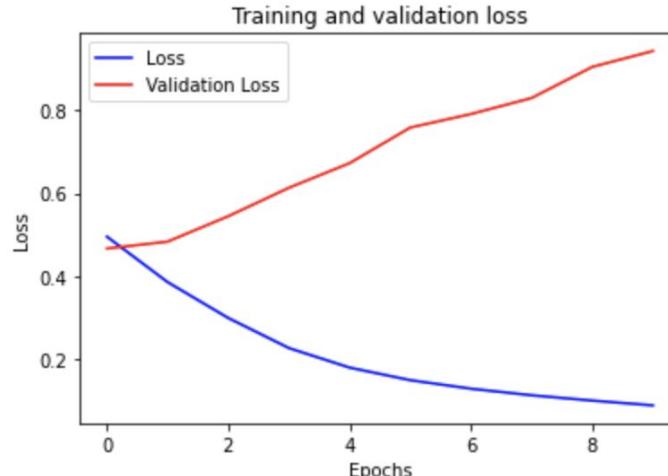
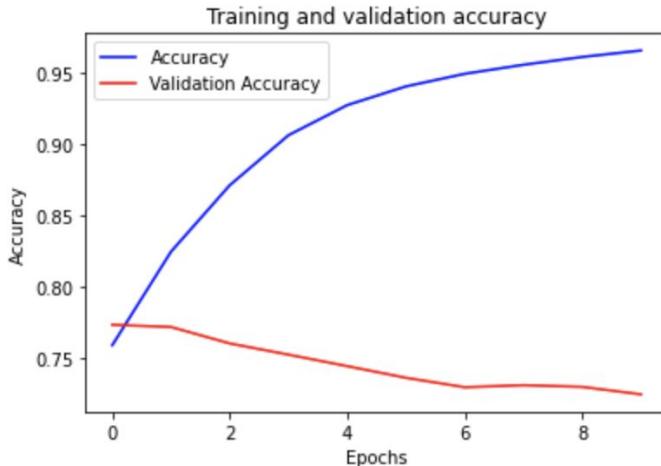
# Sequence models

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size+1, 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: "sequential"

| Layer (type)                                      | Output Shape    | Param #  |
|---|-----------------|----------|
| embedding (Embedding)                             | (None, 16, 100) | 13910600 |
| global_average_pooling1d (GlobalAveragePooling1D) | (None, 100)     | 0        |
| dense (Dense)                                     | (None, 24)      | 2424     |
| dense_1 (Dense)                                   | (None, 1)       | 25       |

Total params: 13,913,049  
Trainable params: 13,913,049  
Non-trainable params: 0



# Sequence models

```
with open('/tmp/glove.6B.100d.txt') as f:  
    for line in f:  
        values = line.split();  
        word = values[0];  
        coefs = np.asarray(values[1:], dtype='float32');  
        embeddings_index[word] = coefs;  
  
embeddings_matrix = np.zeros((vocab_size+1, embedding_dim));  
for word, i in word_index.items():  
    embedding_vector = embeddings_index.get(word);  
    if embedding_vector is not None:  
        embeddings_matrix[i] = embedding_vector;
```

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

Model: "sequential\_16"

| Layer (type)                     | Output Shape    | Param #  |
|----------------------------------|-----------------|----------|
| <hr/>                            |                 |          |
| embedding_16 (Embedding)         | (None, 16, 100) | 13870500 |
| <hr/>                            |                 |          |
| global_average_pooling1D_13      | (None, 100)     | 0        |
| <hr/>                            |                 |          |
| dense_32 (Dense)                 | (None, 24)      | 2424     |
| <hr/>                            |                 |          |
| dense_33 (Dense)                 | (None, 1)       | 25       |
| <hr/>                            |                 |          |
| Total params: 13,872,949         |                 |          |
| Trainable params: 2,449          |                 |          |
| Non-trainable params: 13,870,500 |                 |          |

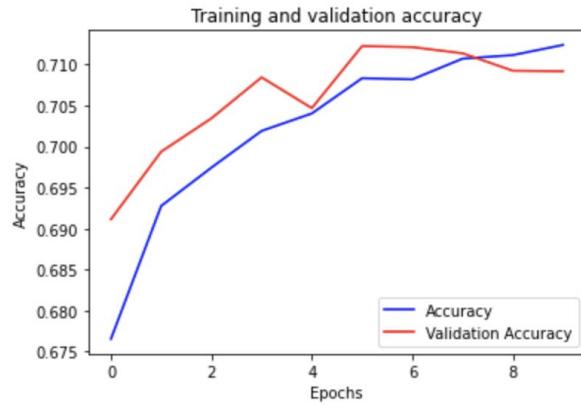
len(embeddings\_matrix)

138,705

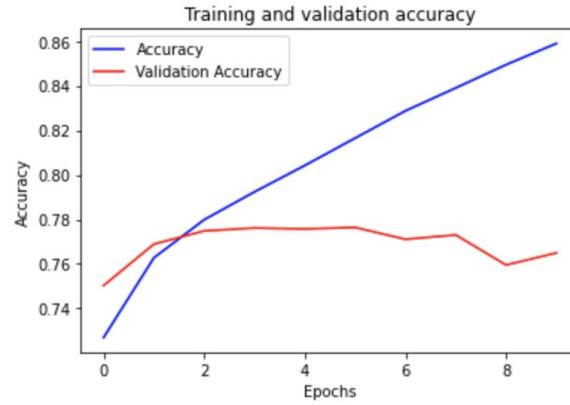
The Second Edition of the 20-volume Oxford English Dictionary, published in 1989, contains full entries for 171,476 words in current use

# Sequence models

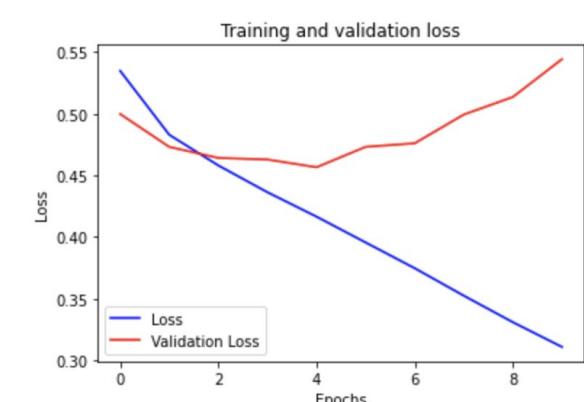
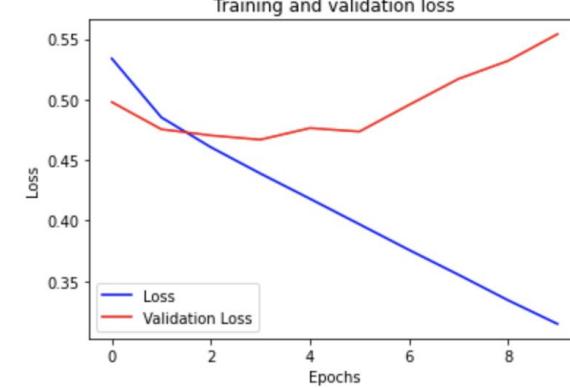
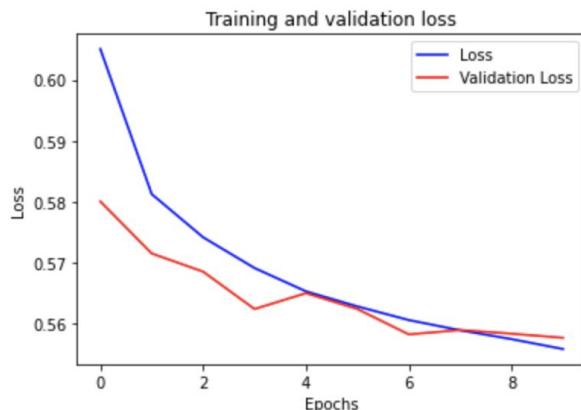
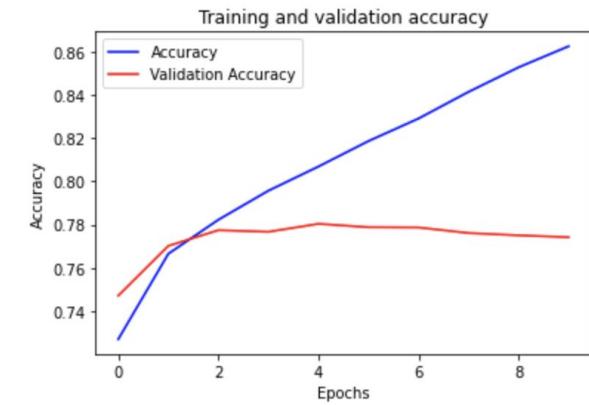
GlobalAveragePooling1D()



Bidirectional(LSTM(64))

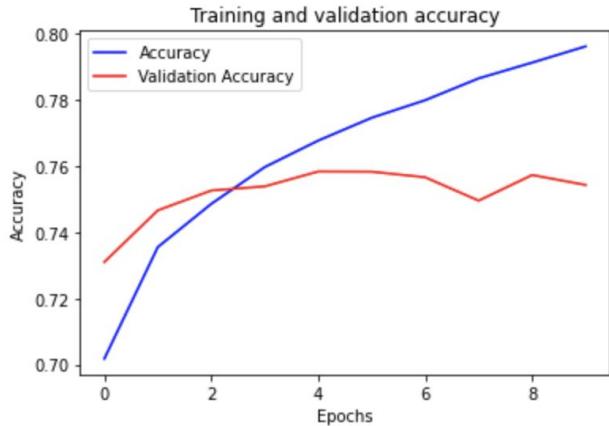


2 x Bidirectional(LSTM(64)), Dropout

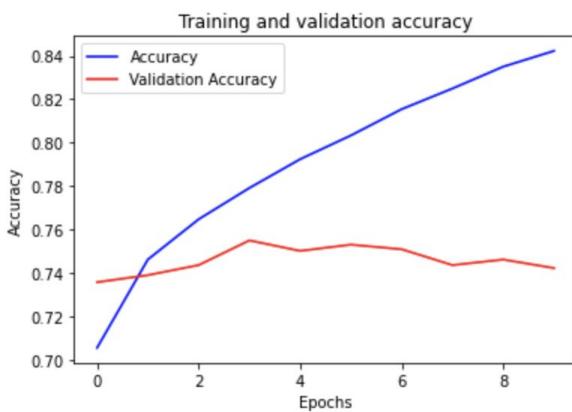


# Sequence models

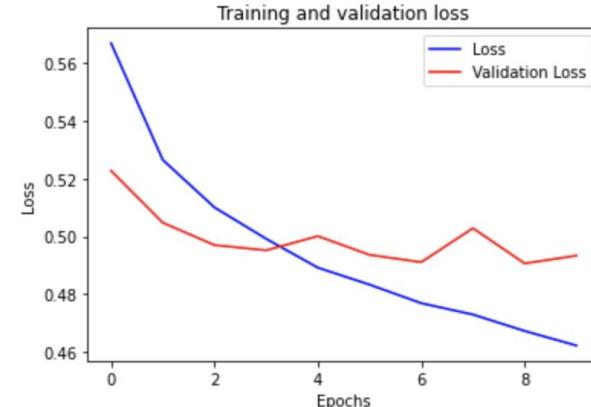
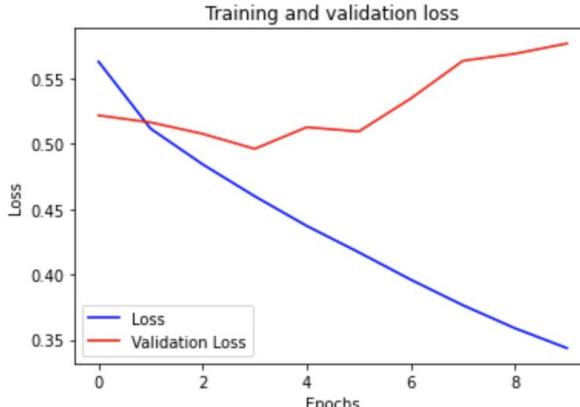
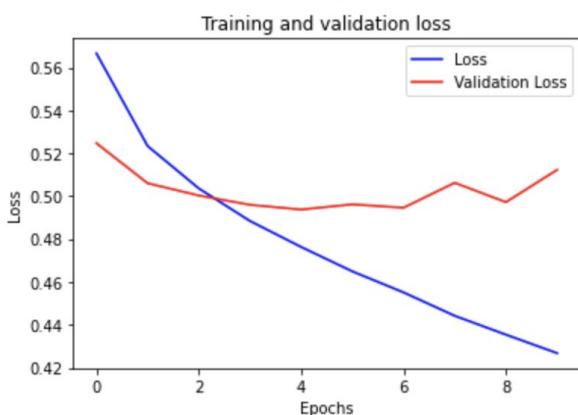
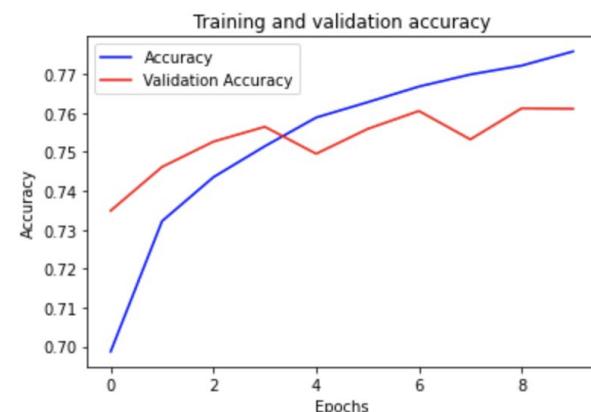
Conv1D, GlobalAveragePooling1D()



2 x Conv1D, GAP1D(), Dropout

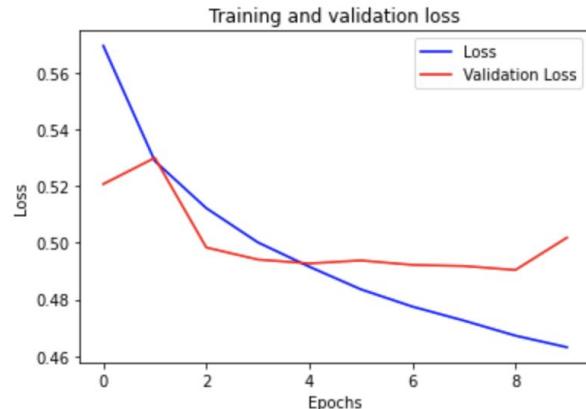
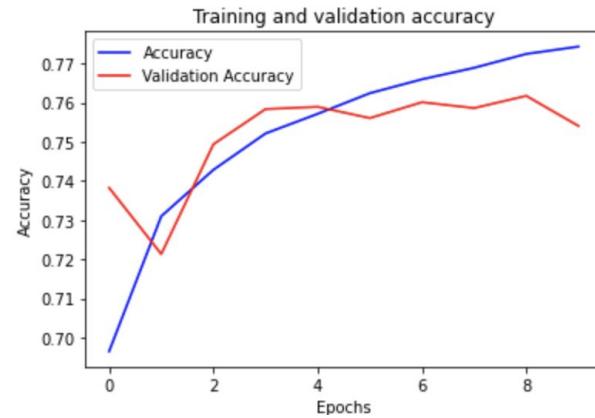


Dropout, Conv1D, MaxP1D(), LSTM(64)



# Sequence models

Dropout, Conv1D, MaxP1D(), 2 x Bidirectional(LSTM(64))



# Let's continue our NLP adventure 😊

FRI, MAY 22, 7:30 PM EDT

## TensorFlow in Practice - Course 3 Week 1,2 - Sentiment in text & Wor...

Online event



Join us for our 6th adventure in Deep Learning! Just bring your curiosity and be ready to meet our growing community 😊 We are taking Course 3 of TensorFlow in Practice Specialization available at:...



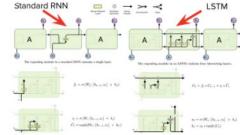
31 attendees



FRI, MAY 29, 7:30 PM EDT

## TensorFlow in Practice - Course 3 Week 3 - Sequence models

Online event



Join us for our 7th adventure in Deep Learning! Just bring your curiosity and be ready to meet our growing community 😊 We are taking Course 3 of TensorFlow in Practice Specialization available at:...

Attend

FRI, JUN 5, 7:30 PM EDT

## TensorFlow in Practice - C3 Week 4 - Sequence models & literature +...

Online event



Join us for our 8th adventure in Deep Learning! Just bring your curiosity and be ready to meet our growing community 😊 We are taking Course 3 of TensorFlow in Practice Specialization available at:...



13 attendees

Attend

FRI, JUN 12, 7:30 PM EDT

## Deep Learning Adventures - Happy Hour 🍻 🍹 🍹

Online event



Join us for a fun conversation 🍻 🍹 🍹 No slides or recording this time, just getting to know each other better and forge a stronger community 😊 We are taking a break from our TensorFlow in Practice Specialization available at:...

Attend

# **Course 3: Natural Language Processing in TensorFlow**

**Week 1:** Sentiment in text

**Week 2:** Word Embeddings

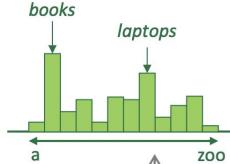
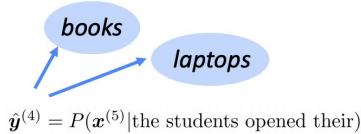
**Week 3:** Sequence models

**Week 4:** Sequence models and literature

# Sequence models and literature

- **Language Modeling** is the task of predicting what word comes next.

the students opened their



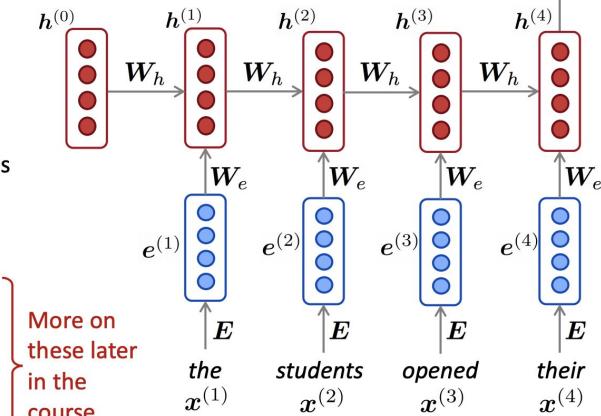
## RNN Language Models

### RNN Advantages:

- Can process **any length** input
- Computation for step  $t$  can (in theory) use information from **many steps back**
- Model size **doesn't increase** for longer input
- Same weights applied on every timestep, so there is **symmetry** in how inputs are processed.

### RNN Disadvantages:

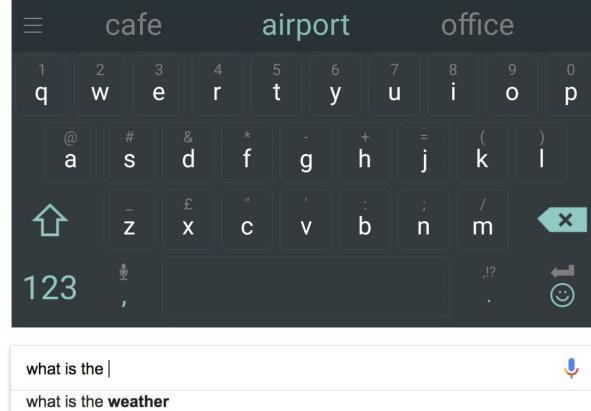
- Recurrent computation is **slow**
- In practice, difficult to access information from **many steps back**



Source: <http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture06-rnnlm.pdf>



I'll meet you at the



**Question:** How to learn a Language Model?

**Answer** (pre- Deep Learning): learn an ***n*-gram Language Model!**

**Definition:** A ***n*-gram** is a chunk of  $n$  consecutive words.

- **unigrams:** "the", "students", "opened", "their"
- **bigrams:** "the students", "students opened", "opened their"
- **trigrams:** "the students opened", "students opened their"
- **4-grams:** "the students opened their"

**Idea:** Collect statistics about how frequent different n-grams are, and use these to predict next word.

# Sequence models and literature

## Evaluating Language Models

- The standard evaluation metric for Language Models is **perplexity**.

$$\text{perplexity} = \prod_{t=1}^T \left( \frac{1}{P_{\text{LM}}(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(1)})} \right)^{1/T}$$

Normalized by number of words

Inverse probability of corpus, according to Language Model

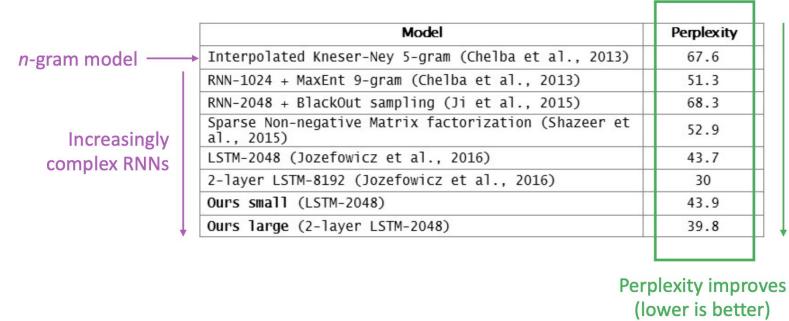
- This is equal to the exponential of the cross-entropy loss  $J(\theta)$ :

$$= \prod_{t=1}^T \left( \frac{1}{\hat{y}_{\mathbf{x}^{t+1}}^{(t)}} \right)^{1/T} = \exp \left( \frac{1}{T} \sum_{t=1}^T -\log \hat{y}_{\mathbf{x}^{t+1}}^{(t)} \right) = \exp(J(\theta))$$

**Lower perplexity is better!**

53

Source: <http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture06-rnnlm.pdf>



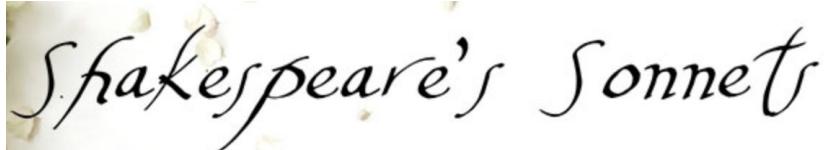
Source: <https://research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words/>

## Why should we care about Language Modeling?

- Language Modeling is a **benchmark task** that helps us **measure our progress** on understanding language
- Language Modeling is a **subcomponent** of many NLP tasks, especially those involving **generating text** or **estimating the probability of text**:

- Predictive typing
- Speech recognition
- Handwriting recognition
- Spelling/grammar correction
- Authorship identification
- Machine translation
- Summarization
- Dialogue
- etc.

# Sequence models and literature



```
data = open('/tmp/sonnets.txt').read()

corpus = data.lower().split("\n")
print(corpus[:3])

tokenizer.fit_on_texts(corpus)
total_words = len(tokenizer.word_index) + 1

# create input sequences using list of tokens
input_sequences = []
for line in corpus:
    #print([line])
    #print(tokenizer.texts_to_sequences([line]))
    #print(tokenizer.texts_to_sequences([line])[0])
    #break
    token_list = tokenizer.texts_to_sequences([line])[0]
    for i in range(1, len(token_list)):
        n_gram_sequence = token_list[:i+1]
        input_sequences.append(n_gram_sequence)

# pad sequences
max_sequence_len = max([len(x) for x in input_sequences])
input_sequences = np.array(pad_sequences(input_sequences, maxlen=max_sequence_len, padding='pre'))

# create predictors and label
predictors, label = input_sequences[:, :-1], input_sequences[:, -1]
```

Source: <http://www.shakespeares-sonnets.com/sonnet/1>

*From fairest creatures we desire increase,  
That thereby beauty's rose might never die,  
But as the riper should by time decease,  
His tender heir might bear his memory:  
But thou contracted to thine own bright eyes,  
Feed'st thy light's flame with self-substantial fuel,  
Making a famine where abundance lies,  
Thy self thy foe, to thy sweet self too cruel:  
Thou that art now the world's fresh ornament,  
And only herald to the gaudy spring,  
Within thine own bud buriest thy content,  
And, tender churl, mak'st waste in niggarding:  
Pity the world, or else this glutton be,  
To eat the world's due, by the grave and thee.*

```
#https://keras.io/api/utils/python_utils/#to_categorical-function
label = ku.to_categorical(label, num_classes=total_words)

label[0]
array([0., 0., 0., ..., 0., 0., 0.], dtype=float32)

len(label[0])
3211

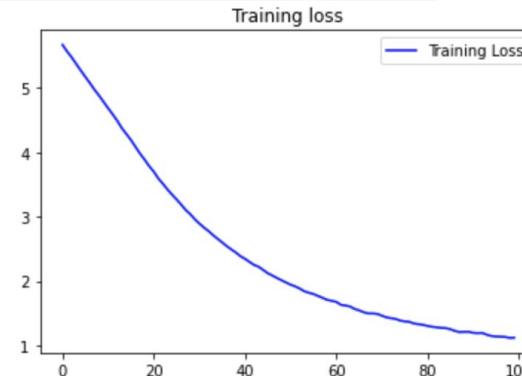
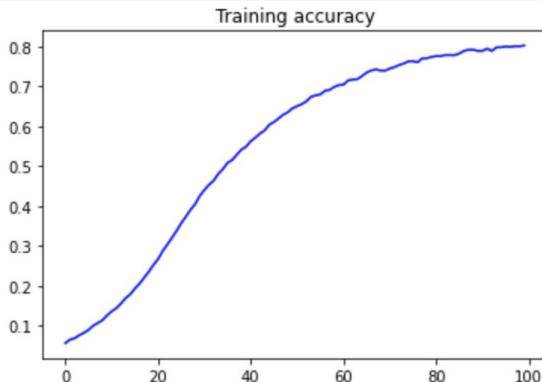
total_words
3211
```

# Sequence models and literature

## Shakespeare's Sonnets

```
model = Sequential()
model.add(Embedding(input_dim=total_words, output_dim=100, input_length=max_sequence_len-1))
model.add(Bidirectional(LSTM(150, return_sequences=True)))
model.add(Dropout(0.2))
model.add(LSTM(100))
model.add(Dense(total_words/2, activation='relu', kernel_regularizer=regularizers.l2(0.01)))
model.add(Dense(total_words, activation='softmax'))

my_optimizer = Adam(learning_rate=0.001)
model.compile(loss='categorical_crossentropy', optimizer=my_optimizer, metrics=['accuracy'])
history = model.fit(predictors, label, epochs=100, verbose=1)
```



| Model: "sequential_3"        |                 |         |  |
|------------------------------|-----------------|---------|--|
| Layer (type)                 | Output Shape    | Param # |  |
| embedding_3 (Embedding)      | (None, 10, 100) | 321100  |  |
| bidirectional_6 (Bidirection | (None, 10, 300) | 301200  |  |
| dropout_3 (Dropout)          | (None, 10, 300) | 0       |  |
| lstm_7 (LSTM)                | (None, 100)     | 160400  |  |
| dense_4 (Dense)              | (None, 1605)    | 162105  |  |
| dense_5 (Dense)              | (None, 3211)    | 5156866 |  |
| <hr/>                        |                 |         |  |
| Total params: 6,101,671      |                 |         |  |
| Trainable params: 6,101,671  |                 |         |  |
| Non-trainable params: 0      |                 |         |  |

Source: <http://www.shakespeares-sonnets.com/sonnet/1>

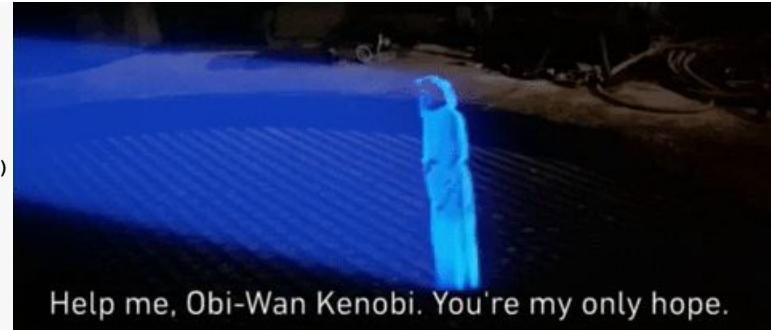
From fairest creatures we desire increase,  
That thereby beauty's rose might never die,  
But as the riper should by time decease,  
His tender heir might bear his memory:  
But thou contracted to thine own bright eyes,  
Feed'st thy light's flame with self-substantial fuel,  
Making a famine where abundance lies,  
Thy self thy foe, to thy sweet self too cruel:  
Thou that art now the world's fresh ornament,  
And only herald to the gaudy spring,  
Within thine own bud buriest thy content,  
And, tender churl, mak'st waste in niggarding:  
Pity the world, or else this glutton be,  
To eat the world's due, by the grave and thee.

# Sequence models and literature

```
seed_text = "Help me Obi Wan Kenobi, you're my only hope"
next_words = 100

for _ in range(next_words):
    token_list = tokenizer.texts_to_sequences([seed_text])[0]
    token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
    predicted = model.predict_classes(token_list, verbose=0)
    output_word = ""
    for word, index in tokenizer.word_index.items():
        if index == predicted:
            output_word = word
            break
    seed_text += " " + output_word
print(seed_text)

tf.keras.preprocessing.sequence.pad_sequences(
    sequences, maxlen=None, dtype='int32', padding='pre', truncating='pre',
    value=0.0
)
```



Help me, Obi-Wan Kenobi. You're my only hope.

shakespeare thyself away desired be cross afford twain

All Images News Shopping Videos More Settings Tools

About 1,530,000 results (0.85 seconds)

www.opensourceshakespeare.org › views › sonnets › sonnet\_view ▾

[View Shakespeare sonnets \(OpenSourceShakespeare.org\)](#)

SONNET I. From fairest creatures we desire increase, ... So thou, **thyself** out-going in thy noon,  
Unlook'd ... To him that bears the strong offence's **cross**. ... And that thou teachest how to make  
one **twain**, ... And found it in thy cheek; he can **afford**

Help me Obi Wan Kenobi, you're my only hope thyself away desired be cross afford twain date of men ' doth groan ' be free free live away away your days here brought of one one more must grow of free ride hits green thee burn pleasure pleasure me of days much brought of pleasure pleasure of pleasure green die burn'd hits dwells beard ward fall by days another rolling ' doth ride ride ride ride ' her view free grew to ride cherish plight grow thus days did tell away away night exchanged mad so meant free his senses can alive pleasure ride ride light decease translate must

# Sequence models and literature

```
seed_text = "Help me Obi Wan Kenobi, you're my only hope"
next_words = 100

for _ in range(next_words):
    token_list = tokenizer.texts_to_sequences([seed_text])[0]
    token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
    predicted = model.predict_classes(token_list, verbose=0)
    output_word = ""
    for word, index in tokenizer.word_index.items():
        if index == predicted:
            output_word = word
            break
    seed_text += " " + output_word
print(seed_text)
```



Help me, Obi-Wan Kenobi. You're my only hope.

shakespeare of every pen shows know confounds thine thine eye aside brig X | 🔍

All Videos Shopping Images News More Settings Tools

About 259,000 results (0.76 seconds)

www.shakespeares-sonnets.com › all ▾

**All sonnets - Shakespeare's Sonnets**

But thou contracted to thine own bright eyes, Feed'st thy ... winter meet, Leese but their show; their substance still lives sweet. ... They do but sweetly chide thee, who confounds ... And though they be outstripped by every pen, ... Thou dost love her, because thou know'st I love her; ... Why with the time do I not glance aside

Help me Obi Wan Kenobi, you're my only hope of every pen shows know confounds thine thine eye aside bright than gently cheeks express'd place in mind keep something new hate of die much date cheeks day was me away the rhyme ' may be bevel disdain erred light light stand bright ' write me flowers so look me doth grow ill old old glory still say thee but now come away a gainer his part torn torn away away ill young new date lies be express'd rare rare ill ill near feast so 'will ' hast me well seen live up memory prove keep invention spent kind check

# Sequence models and literature

TensorFlow > Learn > TensorFlow Core > Tutorials

☆☆☆☆☆

## Text generation with an RNN

 Run in Google Colab

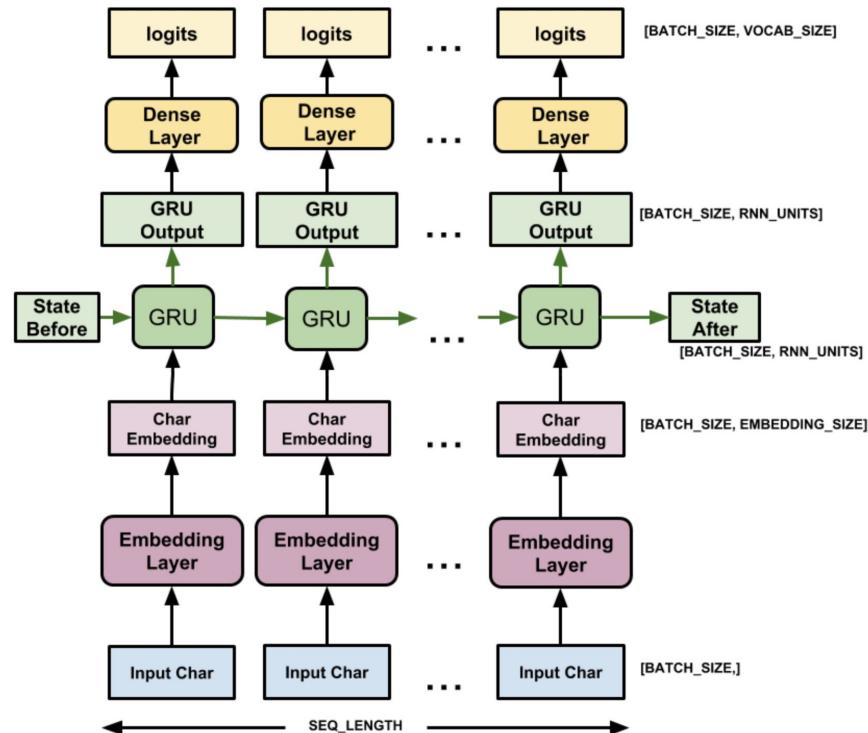
 View source on GitHub

 Download notebook

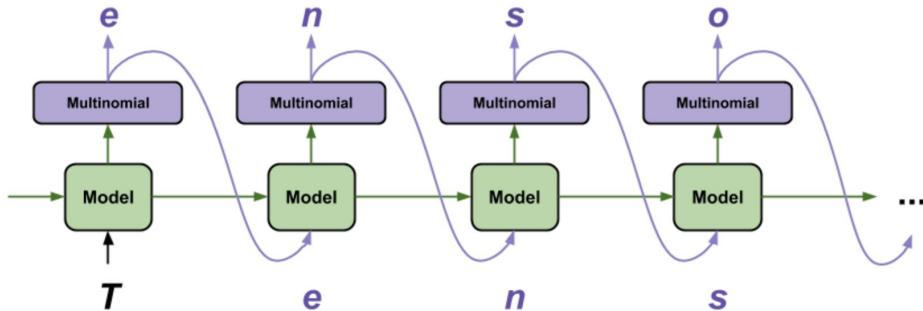
This tutorial demonstrates how to generate text using a character-based RNN. We will work with a dataset of Shakespeare's writing from Andrej Karpathy's [The Unreasonable Effectiveness of Recurrent Neural Networks](#). Given a sequence of characters from this data ("Shakespear"), train a model to predict the next character in the sequence ('e'). Longer sequences of text can be generated by calling the model repeatedly.

```
def build_model(vocab_size, embedding_dim, rnn_units, batch_size):
    model = tf.keras.Sequential([
        tf.keras.layers.Embedding(vocab_size, embedding_dim,
                                  batch_input_shape=[batch_size, None]),
        tf.keras.layers.GRU(rnn_units,
                           return_sequences=True,
                           stateful=True,
                           recurrent_initializer='glorot_uniform'),
        tf.keras.layers.Dense(vocab_size)
    ])
    return model
```

For each character the model looks up the embedding, runs the GRU one timestep with the embedding as input, and applies the dense layer to generate logits predicting the log-likelihood of the next character:



# Sequence models and literature



The prediction loop

The following code block generates the text:

- It Starts by choosing a start string, initializing the RNN state and setting the number of characters to generate.
- Get the prediction distribution of the next character using the start string and the RNN state.
- Then, use a categorical distribution to calculate the index of the predicted character. Use this predicted character as our next input to the model.
- The RNN state returned by the model is fed back into the model so that it now has more context, instead than only one character. After predicting the next character, the modified RNN states are again fed back into the model, which is how it learns as it gets more context from the previously predicted characters.

```
print(generate_text(model, start_string=u"ROMEO: "))
```

Source: [https://www.tensorflow.org/tutorials/text/text\\_generation](https://www.tensorflow.org/tutorials/text/text_generation)

ROMEO: I am unpluck, shalt to the  
Francies would women cluncime against it.

MENENIUS:

Pemprescorce that you shall not be thus; let's in  
justices and by  
have thoughts, to test the stars as great  
As open lack gawned raged  
Duke of Northumberland, this manner of his prince.

MARCIUS:

How do thou wast forced;  
The endempily east enough than whence, or, bear  
headed me aple, to-morrow why I rue,  
My own brothers on't, but stins abboon of  
so souris; or ghinf purnicy in base as as  
Two kings at my heart?

DUKE OF AUMERLE:

Which dost thou be in this person? peace: but whoreson  
hat a not flee  
Whose honour and athe stampet's presence in sorrow is  
there?

BIONDELLO:

What can you gone, and suck'd your prunishen.  
They, we beseech you, sister, whilst Margaret  
pleasant, gentle speed?  
About! following proclaim the end,  
And whom unto his rooate passion,  
Go apperial court.  
Now, sir, but 'tain for Richard, whom is that all.

PETRUCHIO:

Now, there, i' for the !  
With welt repared on Rome, to  
Grimy, sad I fought. Whe

# Sequence models applications

## Character level language model - Dinosaurs

Welcome to Dinosaur Island! 65 million years ago, dinosaurs had a task. Leading biology researchers are creating new breeds of dinosaurs. If a dinosaur does not like its name, it might growl.



Iteration: 30000, Loss: 22.532136

Oryxisaurus

Loca

Lustreoenliusauraes

Pacaisig

Xussancophydpis

Eg

Stochisaurus

Iteration: 32000, Loss: 22.298402

Nivusaaurus

Knacalosaurus

Lustrasaurus

Ndaairus

Xuspandon

Eg

Trohaliomumsaurus

Iteration: 34000, Loss: 22.429814

Onyxmigomalex

Kica

Lutrolomia

Ola

Xosalonghauglosaurus

Ehadon

Stlerdor

## 1 - Problem statement

You would like to create a jazz music piece specially for a friend's birthday. However, you don't know any instruments or music composition. Fortunately, you know deep learning and will solve this problem using an LSTM network.

You will train a network to generate novel jazz solos in a style representative of a body of performed work.



The Deep Learning Specialization is designed to prepare learners to participate in the development of cutting-edge AI technology, and to provide the skills required for the modern Data Scientist. This specialization consists of six courses. After completing all of the courses of this specialization, learners will have a thorough knowledge of deep learning and how it can be used to solve complex problems. This specialization includes courses on Convolutional Neural Networks, Recurrent Language Processing, Speech Processing, and Sequence Models.

Source: <https://www.youtube.com/watch?v=ggQ1y1UHOvc>

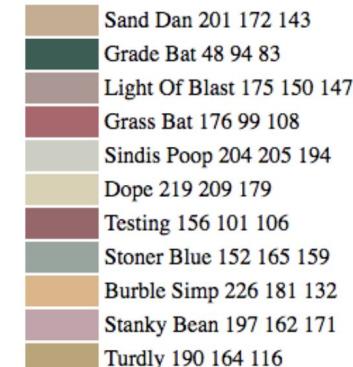
# Sequence models applications

## Gen Gene Generating text Generating text with a RNN Language Model

Let's! Let's have some fun!

- You can train a RNN in that style.
- RNN-LM trained on
- You can train a RNN-LM on any kind of text, then generate text in that style.
- RNN-LM trained on paint color names:

Title: CHOCOLATE RANCH |  
Categories: Game, Casseroles  
Yield: 6 Servings  
  
The  
peop  
and  
be ai  
  
“Sor  
they  
2 tb Parmesan cheese --  
1 c Coconut milk  
3 Eggs, beaten  
  
last Place each pasta over layer  
com until firm. Serve hot in bo  
  
spid Combine the cheese and salt  
and stir in the chocolate a



This is an example of a character-level RNN-LM (predicts what character comes next)

# Sequence models applications



Andrej Karpathy blog

About

Hacker's guide to Neural Networks

## The Unreasonable Effectiveness of Recurrent Neural Networks

May 21, 2015

There's something magical about Recurrent Neural Networks (RNNs). I still remember when I trained my first recurrent network for [Image Captioning](#). Within a few dozen minutes of training my first baby model (with rather arbitrarily-chosen hyperparameters) started to generate very nice looking descriptions of images that were on the edge of making sense. Sometimes the ratio of how simple your model is to the quality of the results you get out of it blows past your expectations, and this was one of those times. What made this result so shocking at the time was that the common wisdom was that RNNs were supposed to be difficult to train (with more experience I've in fact reached the opposite conclusion). Fast forward about a year: I'm training RNNs all the time and I've witnessed their power and robustness many times, and yet their magical outputs still find ways of amusing me. This post is about sharing some of that magic with you.

*We'll train RNNs to generate text character by character and ponder the question "how is that even possible?"*

Visualizing the predictions and the “neuron” firings in the RNN

Source: <http://karpathy.github.io/2015/05/21/rnn-effectiveness>

Paul Graham generator

Shakespeare

Wikipedia

Algebraic Geometry (Latex)

Linux Source Code

Generating Baby Names

# Sequence models and computer vision



Xu et al., "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Figure copyright Kelvin Xu, Jimmy Lei Ba, Jamie Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio, 2015. Reproduced with permission.

# **Course 3: Natural Language Processing in TensorFlow**

**Week 1:** Sentiment in text

**Week 2:** Word Embeddings

**Week 3:** Sequence models

**Week 4:** Sequence models and literature

# Check out these resources



## Get more from Colab

UPGRADE NOW

\$9.99/month

Recurring billing • Cancel anytime

[Restrictions apply, learn more here](#)



### Faster GPUs

Priority access to faster GPUs and TPUs means you spend less time waiting while code is running. [Learn more](#)



### Longer runtimes

Longer running notebooks and fewer idle timeouts mean you disconnect less often. [Learn more](#)



### More memory

More RAM means better performance, and less running out of memory. [Learn more](#)

[See what Colab Pro benefits would look like](#)



Install

Learn

API

Resources

Community

Why TensorFlow

Search

## TensorFlow Core

Overview

Tutorials

Guide

TF 1

### TensorFlow tutorials

Quickstart for beginners  
Quickstart for experts

#### BEGINNER

ML basics with Keras

Load and preprocess data

Estimator

#### ADVANCED

Customization

Distributed training

Images

Text

Structured data

## Text

### Word embeddings

Text classification with an RNN

Text generation with an RNN

Neural machine translation with attention

Image captioning

Transformer model for language understanding

The TensorFlow tutorials are written as Jupyter notebooks and run directly in Google Colab—a hosted notebook environment that requires no setup. Click the [Run in Google Colab](#) button.

## For beginners

The best place to start is with the user-friendly Keras sequential API. Build models by plugging together building blocks. After these tutorials, read the [Keras guide](#).

### Beginner quickstart

This "Hello, World!" notebook shows the Keras Sequential API and `model.fit`.

### Keras basics

This notebook collection demonstrates basic machine learning tasks using Keras.

### Load data

These tutorials use `tf.data` to load various data formats and build input pipelines.

## For experts

The Keras functional and subclassing APIs provide a define-by-run interface for customization and advanced research. Build your model, then write the forward and backward pass. Create custom layers, activations, and training loops.

### Advanced quickstart

This "Hello, World!" notebook uses the Keras subclassing API and a custom training loop.

### Customization

This notebook collection shows how to build custom layers and training loops in TensorFlow.

### Distributed training

Distribute your model training across multiple GPUs, multiple machines or TPUs.

The Advanced section has many instructive notebook examples, including [Neural machine translation](#), [Transformers](#), and [CycleGAN](#).

Source:

<https://colab.research.google.com/signup>

<https://www.tensorflow.org/tutorials/>

# Check out these AI for Healthcare resources



AI for Medicine Specialization

Enroll for Free  
Starts May 22

About How It Works Courses Instructors Enrollment Options FAQ

**There are 3 Courses in this Specialization**

COURSE

## AI for Medical Diagnosis

★★★★★ 4.6 410 ratings • 102 reviews

AI is transforming the practice of medicine. It's helping doctors diagnose patients more accurately, make predictions about patients' future health, and recommend better treatments. As an AI practitioner, you have the opportunity to join in this transformation of modern medicine. If you're already familiar with some of the math and coding behind AI

[SHOW ALL](#)

1

COURSE

## AI for Medical Prognosis

★★★★★ 4.6 113 ratings • 27 reviews

AI is transforming the practice of medicine. It's helping doctors diagnose patients more accurately, make predictions about patients' future health, and recommend better treatments. This Specialization will give you practical experience in applying machine learning to concrete problems in medicine.

[SHOW ALL](#)

2

COURSE

## AI For Medical Treatment

AI is transforming the practice of medicine. It's helping doctors diagnose patients more accurately, make predictions about patients' future health, and recommend better treatments. This Specialization will give you practical experience in applying machine learning to concrete problems in medicine.

[SHOW ALL](#)

3

## AI for Healthcare

Learn to build, evaluate, and integrate predictive models that have the power to transform patient outcomes. Begin by classifying and segmenting 2D and 3D medical images to augment diagnosis and then move on to modeling patient outcomes with electronic health records to optimize clinical trial testing decisions. Finally, build an algorithm that uses data collected from wearable devices to estimate the wearer's pulse rate in the presence of motion.

### PREREQUISITE KNOWLEDGE

Intermediate Python, and Experience with Machine Learning See detailed requirements.

[HIDE DETAILS](#)



### Applying AI to 2D Medical Imaging Data

Learn the fundamental skills needed to work with 2D medical imaging data and how to use AI to derive clinically-relevant insights from data gathered via different types of 2D medical imaging such as x-ray, mammography, and digital pathology. Extract 2D images from DICOM files and apply the appropriate tools to perform exploratory data analysis on them. Build different AI models for different clinical scenarios that involve 2D images and learn how to position AI tools for regulatory approval.

#### PNEUMONIA DETECTION FROM CHEST X-RAYS

### Applying AI to 3D Medical Imaging Data

Learn the fundamental skills needed to work with 3D medical imaging datasets and frame insights derived from the data in a clinically relevant context. Understand how these images are acquired, stored in clinical archives, and subsequently read and analyzed. Discover how clinicians use 3D medical images in practice and where AI holds most potential in their work with these images. Design and apply machine learning algorithms to solve the challenging problems in 3D medical imaging and how to integrate the algorithms into the clinical workflow.

#### HIPPOCAMPUS VOLUME QUANTIFICATION FOR ALZHEIMER'S PROGRESSION

### Applying AI to EHR Data

Learn the fundamental skills to work with EHR data and build and evaluate compliant, interpretable models. You will cover EHR data privacy and security standards, how to analyze EHR data and avoid common challenges, and cover key industry code sets. By the end of the course, you will have the skills to analyze an EHR dataset, transform it to the right level, build powerful features with TensorFlow, and model the uncertainty and bias with TensorFlow Probability and Aequitas.

#### PATIENT SELECTION FOR DIABETES DRUG TESTING

### Applying AI to Wearable Device Data

Learn how to build algorithms that process the data collected by wearable devices and surface insights about the wearer's health. Cover the sensors and signal processing foundation that are critical for success in this domain, including IMU, PPG, and ECG that are common to most wearable devices, and learn how to build three algorithms from real-world sensor data.

#### MOTION COMPENSATED PULSE RATE ESTIMATION

Source:

<https://www.coursera.org/specializations/ai-for-medicine>

<https://www.udacity.com/course/ai-for-healthcare-nanodegree--nd320>

# Check out this certification and books

TensorFlow Core

## Introducing the TensorFlow Developer Certificate!

March 12, 2020



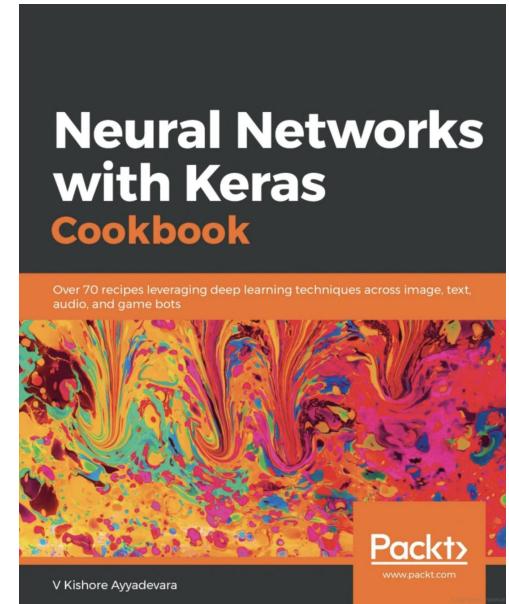
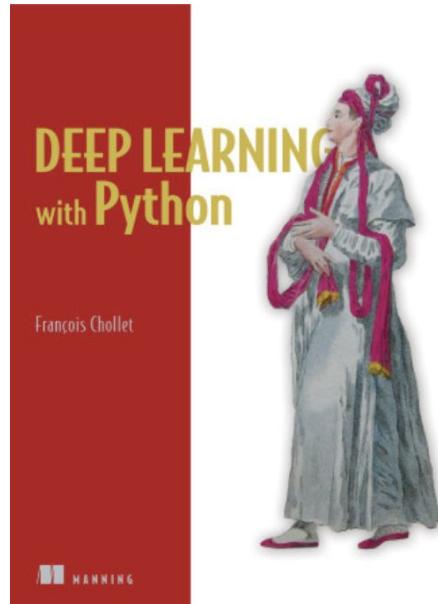
Posted by Alina Shinkarsky, on behalf of the TensorFlow Team

In the AI world today, more and more companies are looking to hire machine learning talent, and simultaneously, an increasing number of students and developers are looking for ways to gain and showcase their ML knowledge with formal recognition. In addition to the courses and learning resources available online, we want to help developers showcase their ML proficiency and help companies hire ML developers to solve challenging problems.



Source:

- <https://blog.tensorflow.org/2020/03/introducing-tensorflow-developer-certificate.html?m=1>
- <https://www.manning.com/books/deep-learning-with-python>
- [https://books.google.com/books?id=5quLDwAAQBAJ&printsec=frontcover&source=gbs\\_qe\\_summary\\_r&cad=0#v=onepage&q&f=false](https://books.google.com/books?id=5quLDwAAQBAJ&printsec=frontcover&source=gbs_qe_summary_r&cad=0#v=onepage&q&f=false)
- <https://github.com/PacktPublishing/Neural-Networks-with-Keras-Cookbook/tree/master/Chapter11>



# Let's continue our NLP adventure 😊

FRI, MAY 22, 7:30 PM EDT

## TensorFlow in Practice - Course 3 Week 1,2 - Sentiment in text & Wor...

Online event



Join us for our 6th adventure in Deep Learning! Just bring your curiosity and be ready to meet our growing community 😊 We are taking Course 3 of TensorFlow in Practice Specialization available at:...



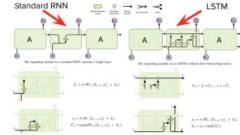
31 attendees



FRI, MAY 29, 7:30 PM EDT

## TensorFlow in Practice - Course 3 Week 3 - Sequence models

Online event



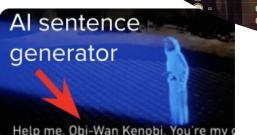
Join us for our 7th adventure in Deep Learning! Just bring your curiosity and be ready to meet our growing community 😊 We are taking Course 3 of TensorFlow in Practice Specialization available at:...

Attend

FRI, JUN 5, 7:30 PM EDT

## TensorFlow in Practice - C3 Week 4 - Sequence models & literature +...

Online event



Join us for our 8th adventure in Deep Learning! Just bring your curiosity and be ready to meet our growing community 😊 We are taking Course 3 of TensorFlow in Practice Specialization available at:...



13 attendees

Attend

FRI, JUN 12, 7:30 PM EDT

## Deep Learning Adventures - Happy Hour 🍻 🍹 🍹

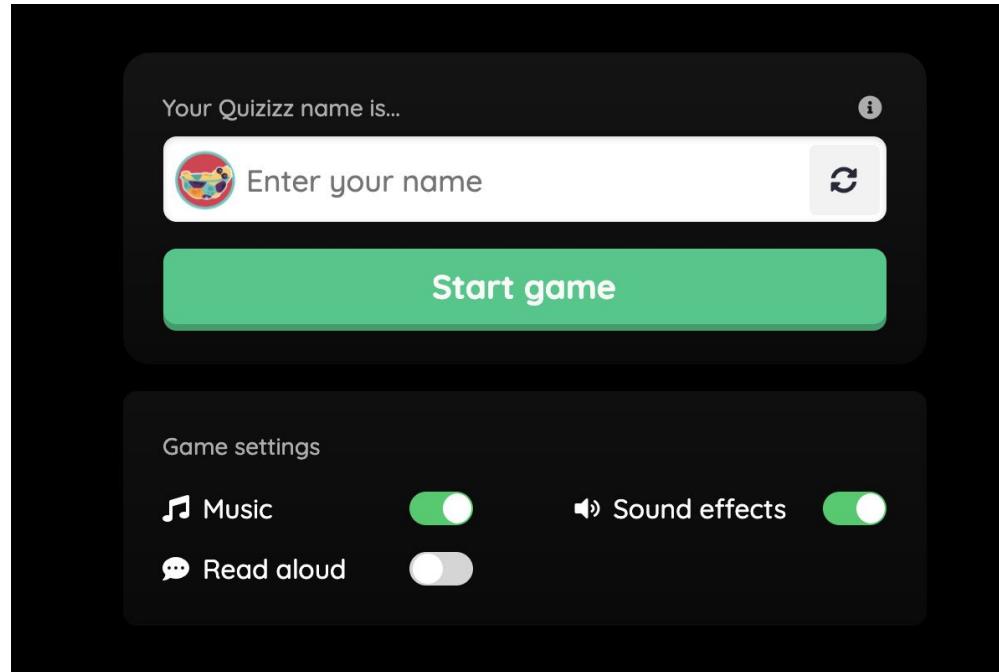
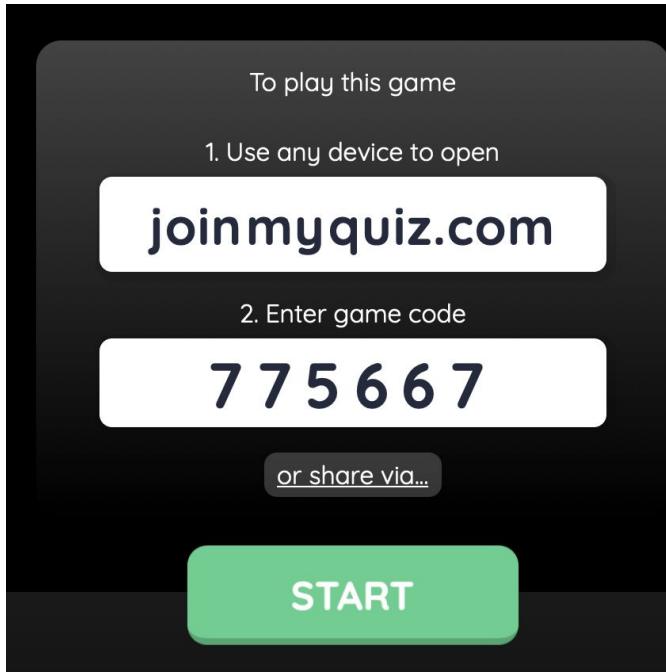
Online event



Join us for a fun conversation 🍻 🍹 🍹 No slides or recording this time, just getting to know each other better and forge a stronger community 😊 We are taking a break from our TensorFlow in Practice Specialization available at:...

Attend

# Time for a fun game 😊🎉



Practice here or use Flashcards:

<https://quizizz.com/join/quiz/5e90e8e631fb32001f63510b/start?from=soloLinkShare&referrer=5d921444d0fa99001a135336>

# Time for a fun game



Practice here or use Flashcards:

[quizizz.com/join/quiz/5e87bdb07fa7f001b120404/start?from=soloLinkShare&referrer=5d921444d0fa99001a135336](https://quizizz.com/join/quiz/5e87bdb07fa7f001b120404/start?from=soloLinkShare&referrer=5d921444d0fa99001a135336)

The image shows two side-by-side screenshots. On the left is the Quizizz game dashboard titled 'Join a Game - Quizizz' with a game code '432 236'. It displays a list of 22 players with their names, profile icons, scores, and a 'Leave' button. A video feed of a person is shown in the top right corner. On the right is a slide from a Google Slides presentation titled 'Deep Learning Adventures - Chapter 1 - Presentation 1'. The slide has a purple header with the text 'How do you import Tensorflow in your Python code'. Below the header are four numbered options: 1. 'import tf from ai' (blue), 2. 'import tensorflow from google' (teal), 3. 'expot tensorflow as tf' (yellow), and 4. 'import tensorflow as tf' (pink). The slide also includes navigation controls like a play button, a progress bar showing '4/12', and a timer '100min'.

| Player           | Score |
|------------------|-------|
| Anil             | 8630  |
| rek              | 8630  |
| Charles Stockman | 7450  |
| Martin           | 7290  |
| Angel            | 7230  |
| Chuba            | 7050  |
| Peter            | 6880  |
| Dean             | 6840  |
| Melissa          | 6505  |
| Sanjay           | 5710  |
| Hasson           | 4750  |
| AH               | 4470  |
| Kottie           | 4470  |
| v                | 4010  |
| KL               | 3910  |
| Tony             | 3240  |
| SS               | 2130  |
| George           | 2100  |
| Thomas           | 0     |

# Questions

# Discussion

## 2 Deep Learning representations

For representations:

- nodes represent inputs, activations or outputs
- edges represent weights or biases

Here are several examples of Standard deep learning representations

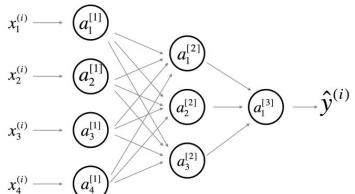


Figure 1: Comprehensive Network: representation commonly used for Neural Networks. For better aesthetic, we omitted the details on the parameters ( $w_{ij}^{[l]}$  and  $b_i^{[l]}$  etc...) that should appear on the edges

## Standard notations for Deep Learning

This document has the purpose of discussing a new standard for deep learning mathematical notations.

### 1 Neural Networks Notations.

General comments:

- superscript (i) will denote the  $i^{th}$  training example while superscript [l] will denote the  $l^{th}$  layer

Sizes:

$\cdot m$  : number of examples in the dataset

$\cdot n_x$  : input size

$\cdot n_y$  : output size (or number of classes)

$\cdot n_h^{[l]}$  : number of hidden units of the  $l^{th}$  layer

In a for loop, it is possible to denote  $n_x = n_h^{[0]}$  and  $n_y = n_h^{[\text{number of layers} + 1]}$ .

$\cdot L$  : number of layers in the network.

Objects:

$\cdot X \in \mathbb{R}^{n_x \times m}$  is the input matrix

$\cdot x^{(i)} \in \mathbb{R}^{n_x}$  is the  $i^{th}$  example represented as a column vector

$\cdot Y \in \mathbb{R}^{n_y \times m}$  is the label matrix

$\cdot y^{(i)} \in \mathbb{R}^{n_y}$  is the output label for the  $i^{th}$  example

$\cdot W^{[l]} \in \mathbb{R}^{\text{number of units in next layer} \times \text{number of units in the previous layer}}$  is the weight matrix, superscript [l] indicates the layer

$\cdot b^{[l]} \in \mathbb{R}^{\text{number of units in next layer}}$  is the bias vector in the  $l^{th}$  layer

$\cdot \hat{y} \in \mathbb{R}^{n_y}$  is the predicted output vector. It can also be denoted  $a^{[L]}$  where  $L$  is the number of layers in the network.

Common forward propagation equation examples:

$a = g^{[l]}(W_x x^{(i)} + b_1) = g^{[l]}(z_1)$  where  $g^{[l]}$  denotes the  $l^{th}$  layer activation function

$\hat{y}^{(i)} = \text{softmax}(W_h h + b_2)$

· General Activation Formula:  $a_j^{[l]} = g^{[l]}(\sum_k w_{jk}^{[l]} a_k^{[l-1]} + b_j^{[l]}) = g^{[l]}(z_j^{[l]})$

·  $J(x, W, b, y)$  or  $J(\hat{y}, y)$  denote the cost function.

Examples of cost function:

$\cdot J_{CE}(\hat{y}, y) = -\sum_{i=0}^m y^{(i)} \log \hat{y}^{(i)}$

$\cdot J_1(\hat{y}, y) = \sum_{i=0}^m |y^{(i)} - \hat{y}^{(i)}|$

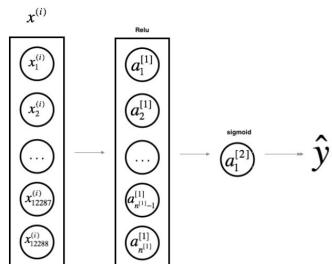


Figure 2: Simplified Network: a simpler representation of a two layer neural network, both are equivalent.

# Introduction to Deep Learning



TensorFlow

meetup

Start a new group Log in Sign up



**Deep Learning Adventures**

Washington, DC  
377 members · Public group  
Organized by George Z. and 2 others

Share: [f](#) [t](#) [in](#)

About Events Members Photos Discussions More

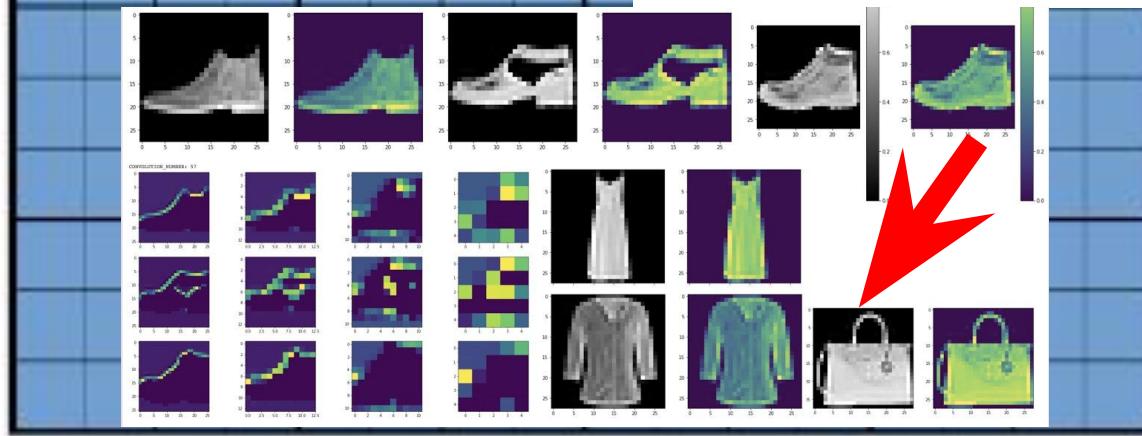
Join this group ...

A red arrow points from the bottom right towards the 'Join this group' button.

# Convolutional Neural Networks



TensorFlow



# Deep Learning Trivia Game



TensorFlow

The image shows a screenshot of the Deep Learning Trivia Game. On the left, there is a sidebar with a green progress bar at 68% and 73% Accuracy. Below it is a "Leaderboard" section showing the top 10 players:

| Rank | Name         | Score |
|------|--------------|-------|
| 1    | Robert Kroig | 11    |
| 2    | Peter        | 10    |
| 3    | George Zoto  | 9     |
| 4    | Martin       | 8     |
| 5    | Sanjoy       | 7     |
| 6    | Jesse        | 6     |
| 7    | Kathleen     | 5     |
| 8    | Kathy        | 4     |
| 9    | Pete         | 3     |
| 10   | melbottoast  | 2     |

On the right, there is a question card with the following content:

What do we see in this image?

3 items, the grey one is the input of a CNN and the green one is the output of a Conv2D

3 items, the grey one is the input of a CNN and the green one is the output of a MaxPooling2D

3 items, the grey one and the green one are the same images on different color scales

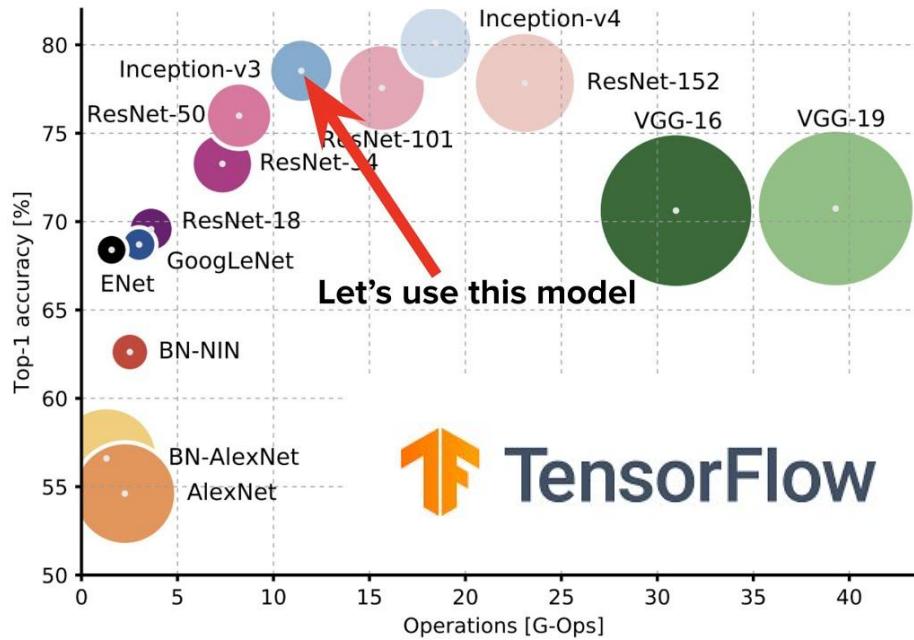
3 items, the grey one is the input of a CNN and the green one is the output of a AveragePooling2D

A large red arrow points to the third item in the list, which states: "3 items, the grey one and the green one are the same images on different color scales".

# Dog vs Cat

kaggle classifier  TensorFlow





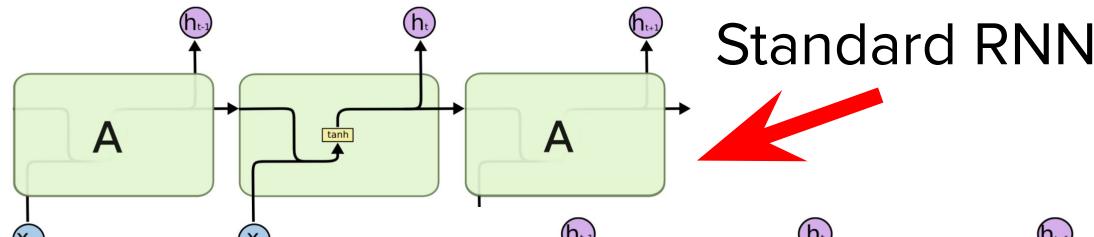
# NLP Tutorial



Let's look at this embedding

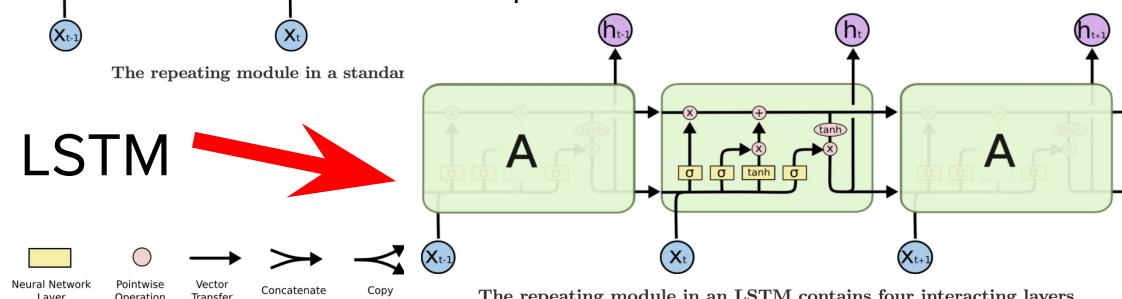


# NLP Tutorial



The repeating module in a standard RNN

LSTM



The repeating module in an LSTM contains four interacting layers.

Neural Network Layer

Pointwise Operation

Vector Transfer

Concatenate

Copy

# Sequence models and literature

## NLP Tutorial



Help me Obi Wan Kenobi, you're my only hope thyself away desired be cross afford twain date of men ' doth groan ' be free free live away away your days here brought of one one more must grow of free ride hits green thee burn pleasure pleasure me of days much brought of pleasure pleasure of pleasure green die burn'd hits dwells beard ward fall by days another rolling ' doth ride ride ride ride ' her view free grew to ride cherish plight grow thus days did tell away away night exchanged mad so meant free his senses can alive pleasure ride ride light decease translate must

# Deep Learning Adventures



4 Courses



04/26/2020

George Zoto

has successfully completed the online, non-credit Specialization

## TensorFlow in Practice

In this specialization, you got a grounding in what you need to get started with TensorFlow in Practice. The goal was to help you take the next steps, such as going deeper into understanding Machine Learning and the practice of understanding loss functions, optimizers and metrics, or perhaps you want to know more about neural network types and different types of layers, from convolutions to recurrent or LSTM. Now that you have used some of them and seen the impact of different layer types in practice, you can go forward equipped to go deeper!

The online specialization courses in this certificate may draw on material from courses taught on campus, but the included courses are not equivalent to our campus courses. The diploma in the online specialization does not count as evidence of having completed a degree. This certificate does not confer a University grade, the award of a degree, or a relevant certificate of the university.

View certificate at:  
[coursera.org/verify/8ZD2XAHSMNAM](https://coursera.org/verify/8ZD2XAHSMNAM)



## TensorFlow Developer Certificate

This certifies that

George Zoto

has passed the TensorFlow Developer Certificate exam covering foundational, practical machine learning skills through the building and training of models using TensorFlow.



Laurence Moroney  
Head of TensorFlow Advocacy

Issue Date  
May 24, 2020

Expiration Date  
May 24, 2023

Certificate ID  
18625727



## Deep Learning Adventures Chapter 1 - Presentation 1

A quick overview of Coursera's Tensorflow in Practice specialization course

George Zoto

<https://www.meetup.com/Deep-Learning-Adventures>

## Deep Learning Adventures TensorFlow In Practice - Presentation 3

A quick overview of Coursera's Tensorflow in Practice specialization course

Robert Kraig, David Patton, George Zoto

<https://www.meetup.com/Deep-Learning-Adventures>

## Deep Learning Adv TensorFlow In Pract Presentation 2

A quick overview of Coursera's Tensorflow in Practice specialization course

Robert Kraig, David Patton, George Zoto

<https://www.meetup.com/Deep-Learning-Adventures>

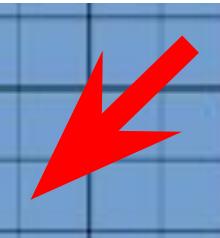


## Deep Learning Adventures TensorFlow In Practice - Presentation 4

A quick overview of Coursera's  
specialization course

Robert Kraig, David Patton, Geor

<https://www.meetup.com/Deep-Learn>

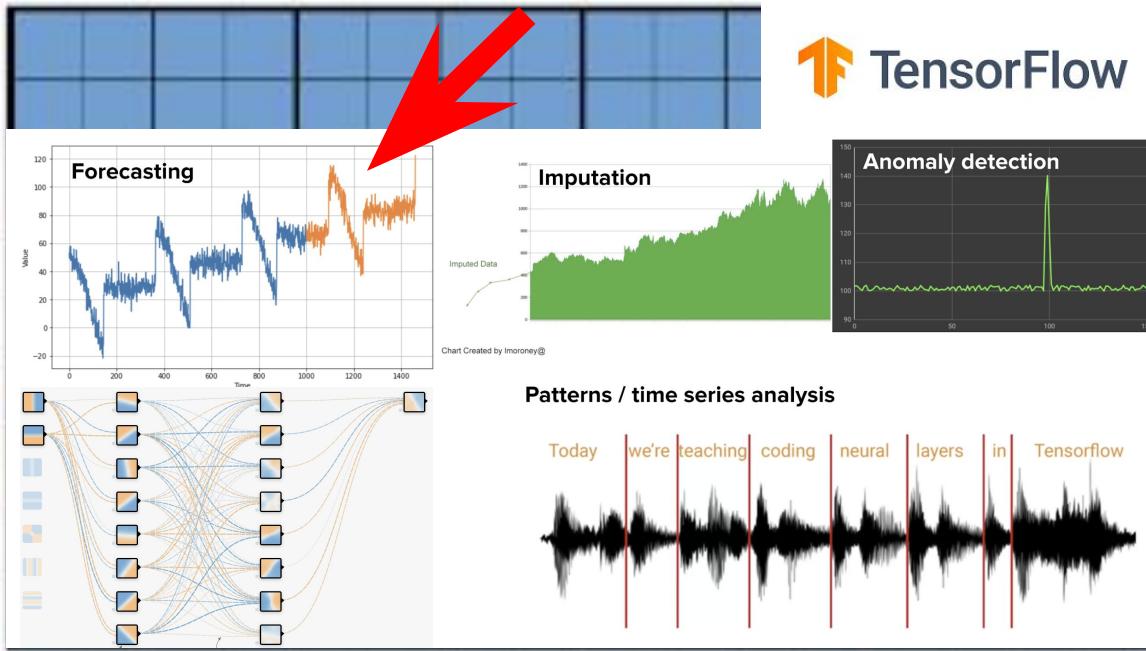


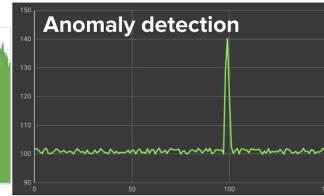
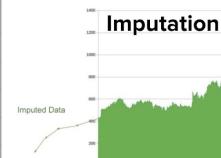
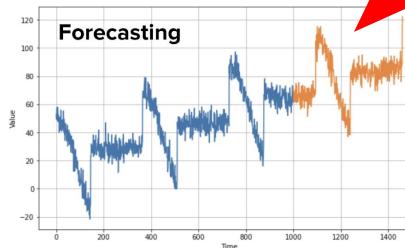
## Deep Learning Adventures TensorFlow In Practice - Presentation 5

A quick overview of Coursera's Tensorflow in Practice specialization course

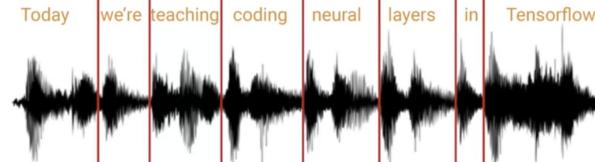
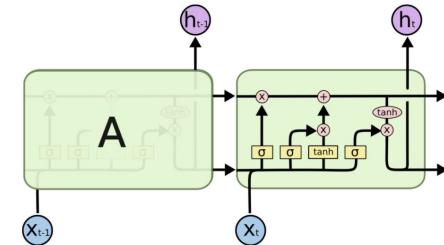
Robert Kraig, David Patton, George Zoto

<https://www.meetup.com/Deep-Learning-Adventures>





### Patterns / time series analysis





## Differentiate yourself with the TensorFlow Developer Certificate

Demonstrate your proficiency in using TensorFlow to solve deep learning and ML problems. Get recognized for your skills and join [our Certificate Network](#).

[Review Candidate Handbook](#)



a

a