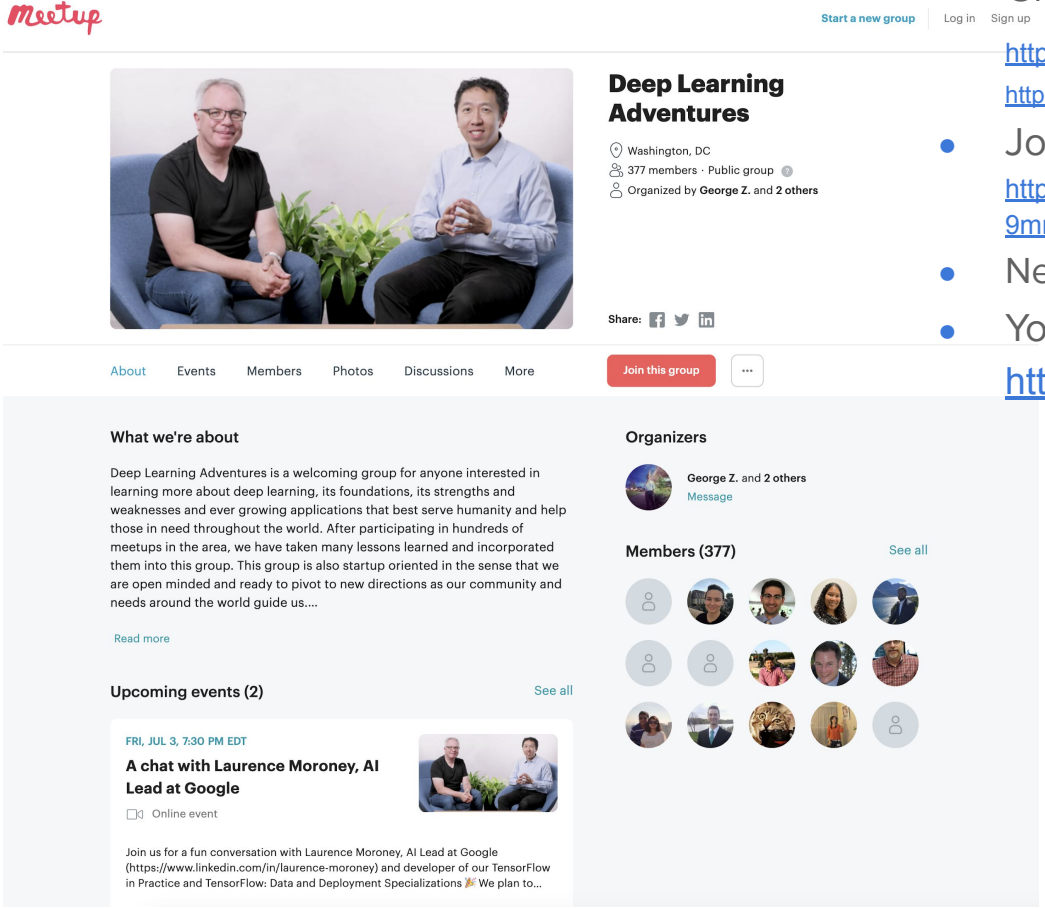
Deep Learning Adventures TensorFlow In Practice - Presentation 6

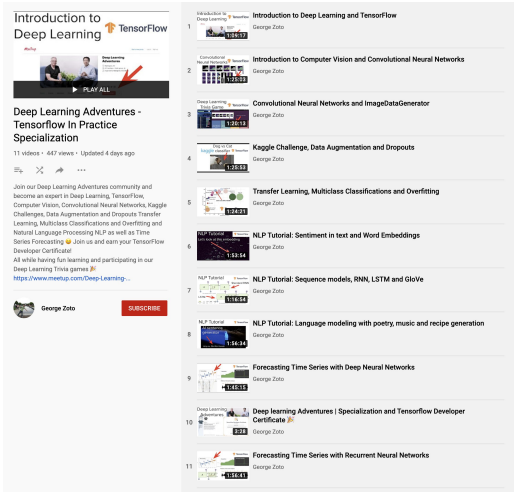
A chat with Laurence Moroney, AI Lead at Google, after completing Coursera's Tensorflow in Practice Specialization

Robert Kraig, David Patton, George Zoto
https://www.meetup.com/Deep-Learning-Adventures

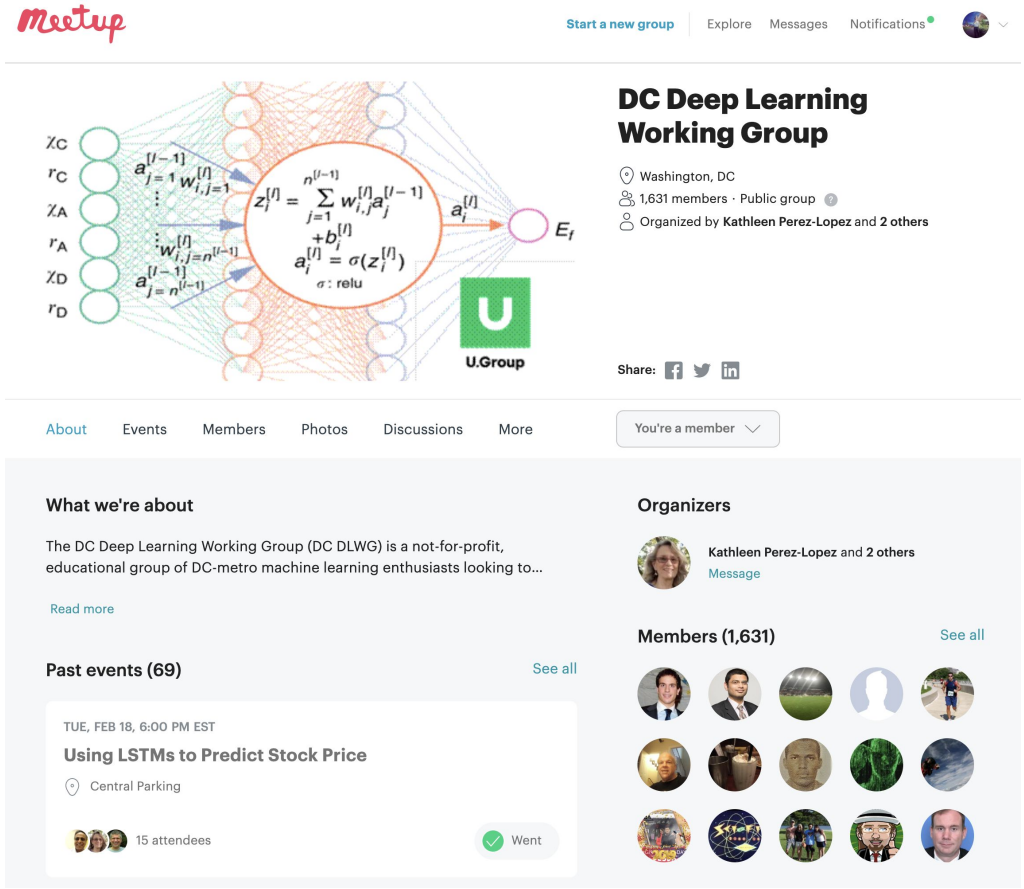
In the beginning...

- 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
- Need a refresher or new to Deep Learning?
- YouTube recordings of all our Meetups 😀
  https://bit.ly/deep-learning-tf



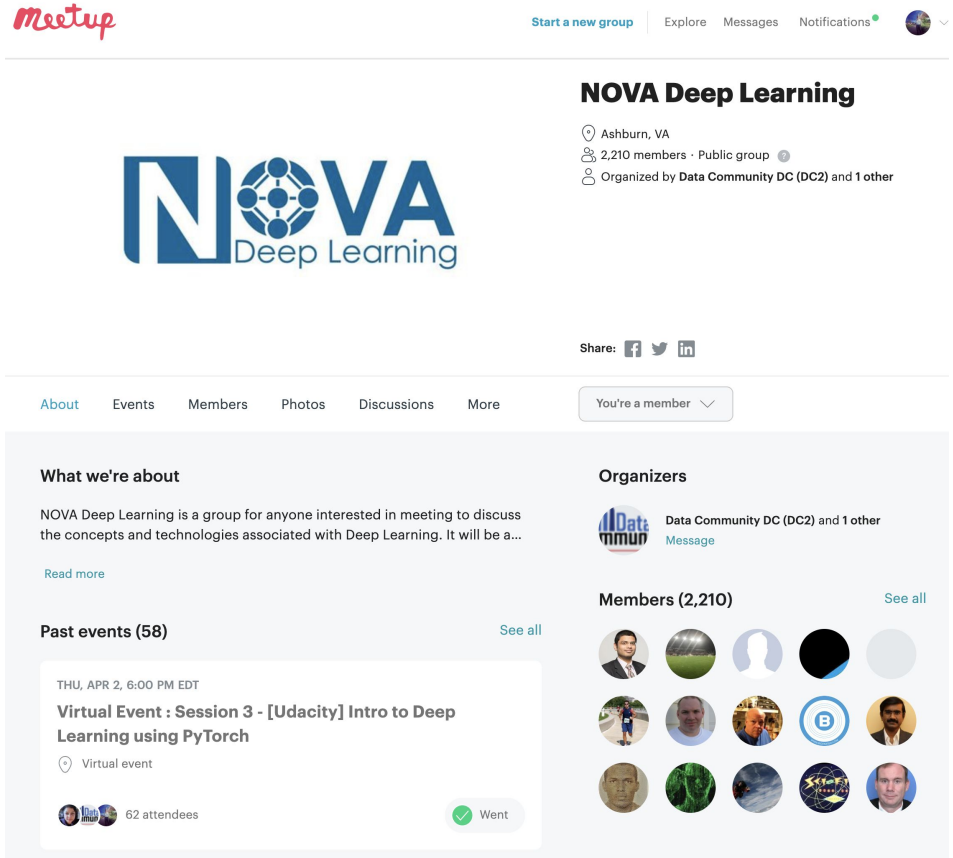


Deep Learning and Data Science Partner Communities



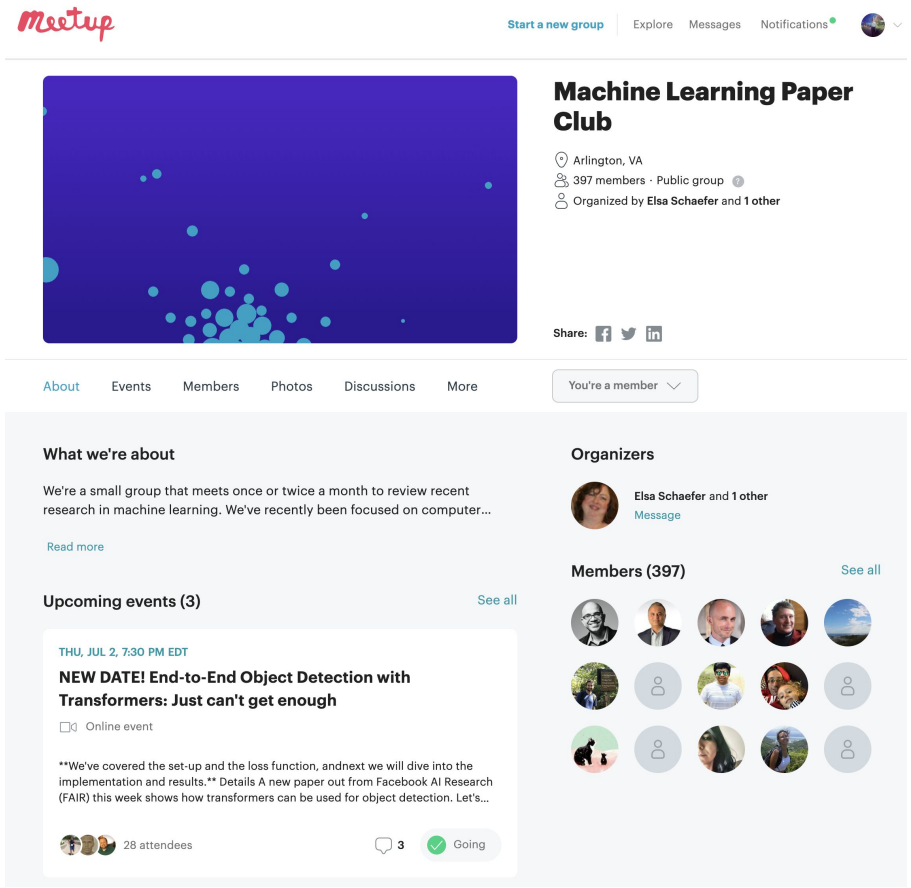
Source: https://www.meetup.com/DC-Deep-Learning-Working-Group

Deep Learning and Data Science Partner Communities



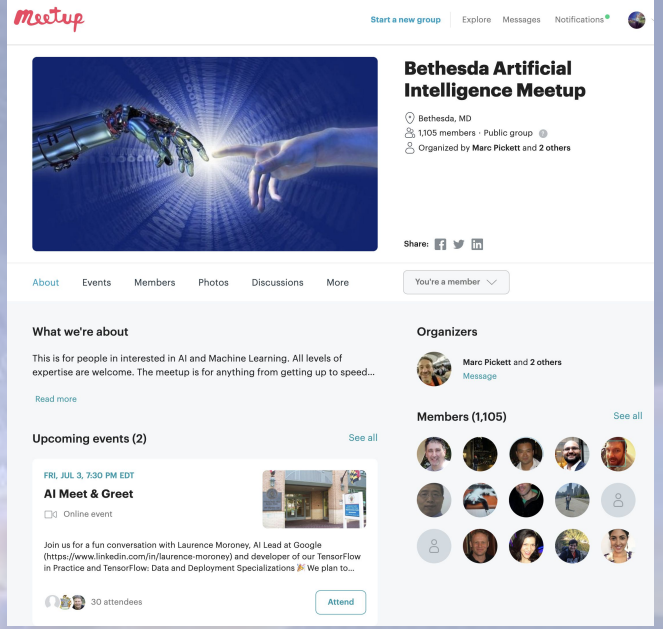
Source: https://www.meetup.com/novadeeplearning

Deep Learning and Data Science Partner Communities



Source: https://www.meetup.com/Machine-Learning-Paper-Club

Bethesda Artificial Intelligence Meetup



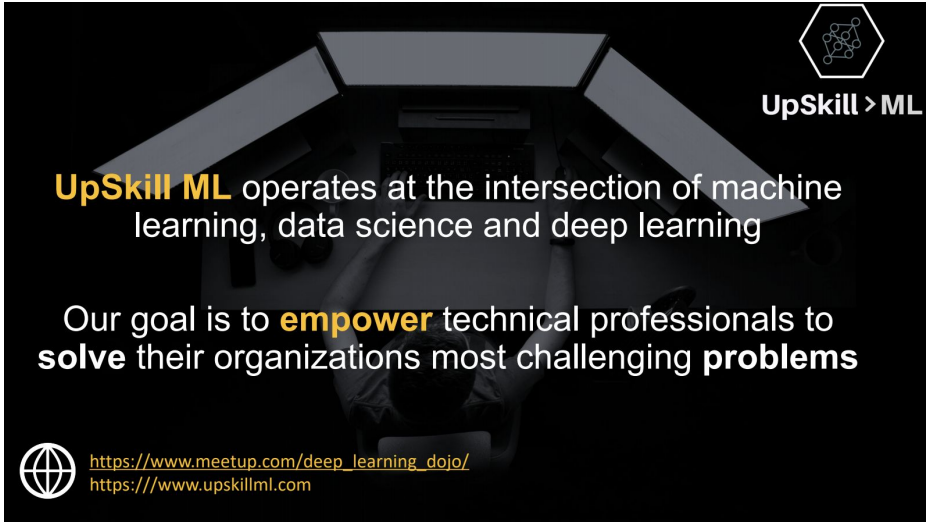
A meetup for all levels of people interested in AI, Machine Learning, and related fields.

Centered in the Bethesda area, and meet biweekly Saturdays from 10AM to noon.
(We have free coffee when we're not virtual.)

https://www.meetup.com/Bethesda-Artificial-Intelligence-Meetup/

Source: https://www.meetup.com/Bethesda-Artificial-Intelligence-Meetup

Deep Learning and Data Science Partner Communities

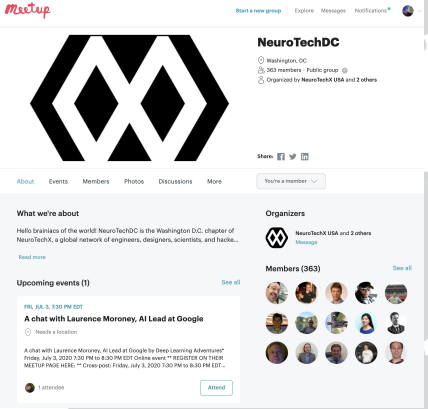


UpSkill ML is a meetup that teaches applied machine learning concepts to data science and machine learning practitioners in the DMV area. Our sessions are in-person workshops that focus on applications and advances in the field, presenting them in a digestible way. Our goal is to present you with the math and intuition behind the ideas along with starter code for your projects.

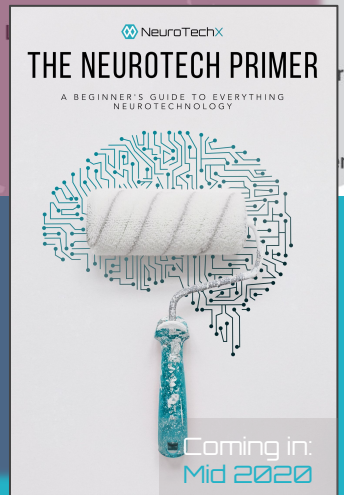
Data Education DC is a non-profit professional and educational group that meets bi-monthly to learn and discuss Data Science concepts and applications. Members learn to use Machine Learning, Data Science and Data Visualization tools that they can apply immediately following our meetups!

Source: https://www.meetup.com/deep_learning_dojo
https://www.meetup.com/Data-Education-DC

NeuroTechX is an international community that facilitates the advancement of neurotechnology



NEUROTECHX



20 CHAPTERS WORLDWIDE

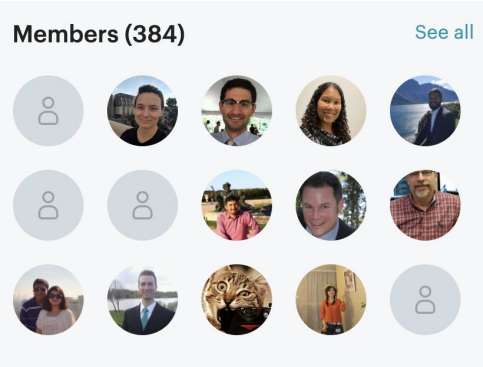
200 MEETUP EVENTS

13,000 MEETUP NEUROHACKERS

3500 ONLINE COMMUNITY MEMBERS

Source: https://www.meetup.com/NeuroTechDC

Get to know our community



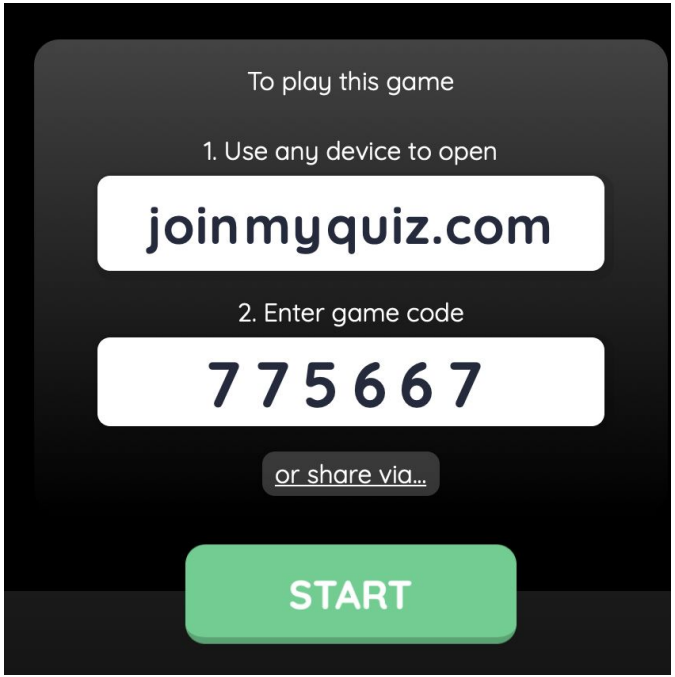
Hello, my name is ____ and I am a new/existing/regular member of this meetup.

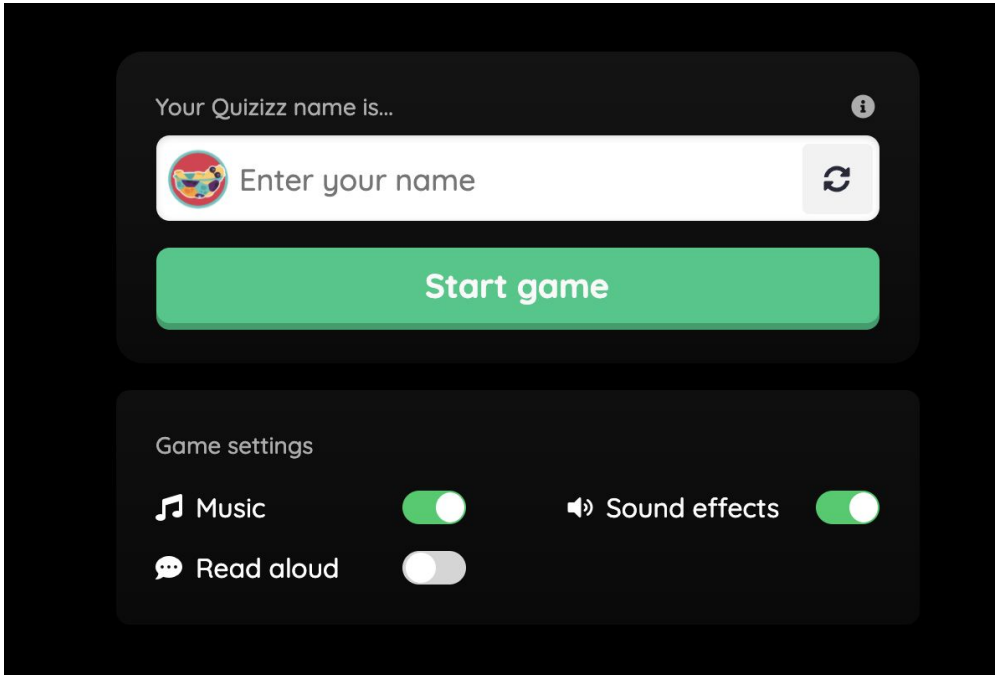
I enjoy ____ and I am interested in learning more about ____

Training set 😀

Hello, my name is **George** and I am a **regular** member of this meetup. I enjoy **applying deep learning to solve interesting problems** and I am interested in learning more about **time series and forecasting**.

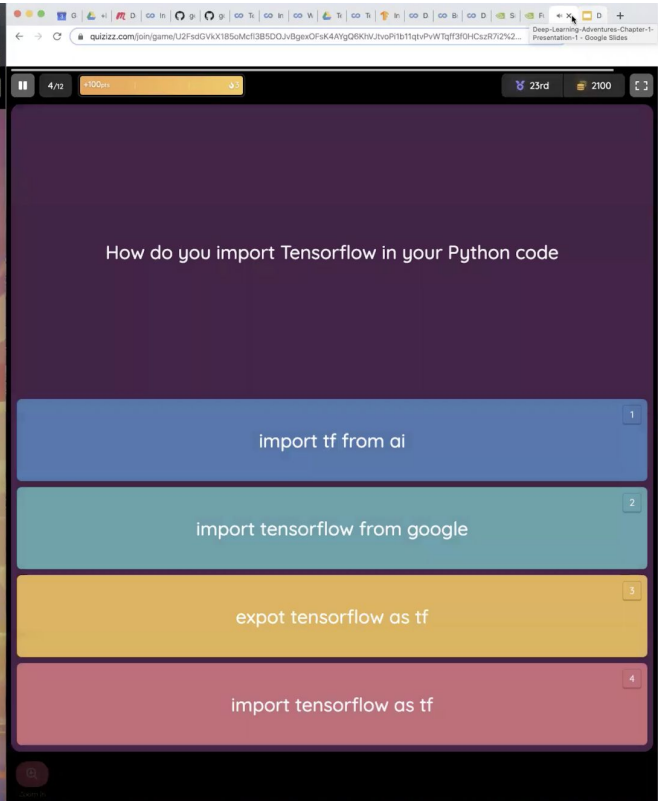
Not a typical Meetup... Get ready for a fun game 😀🎉



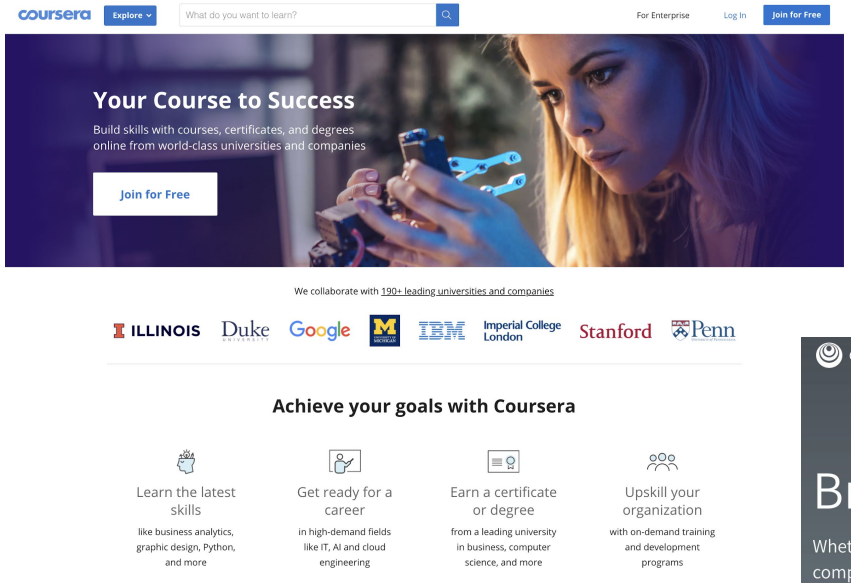


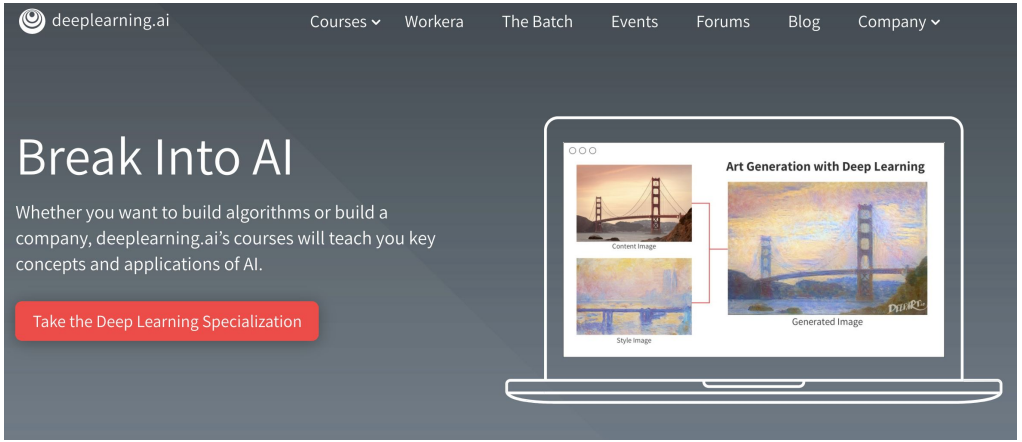
Time for a fun game 😀🎉

Practice here or use Flashcards:
https://quizizz.com/join/quiz/5e87bdbc07fa7f001b120404/start?from=soloLinkShare&referrer=5d921444d0fa99001a135336



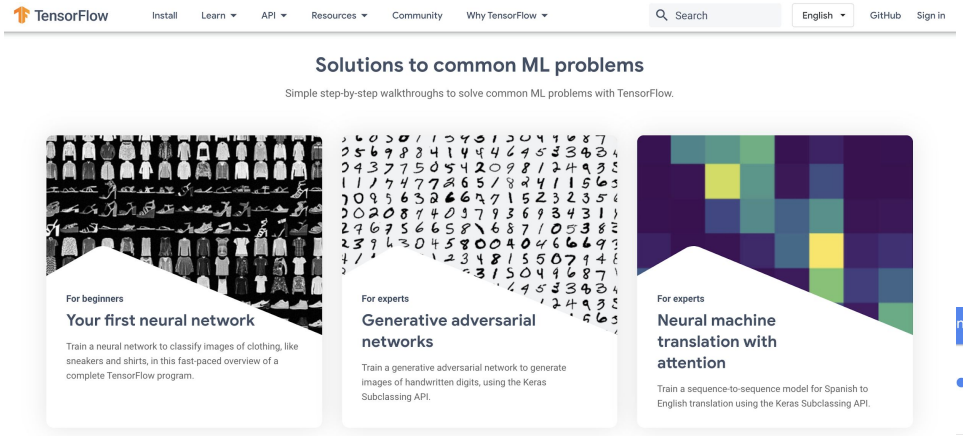
Attribution to Coursera and deeplearning.ai

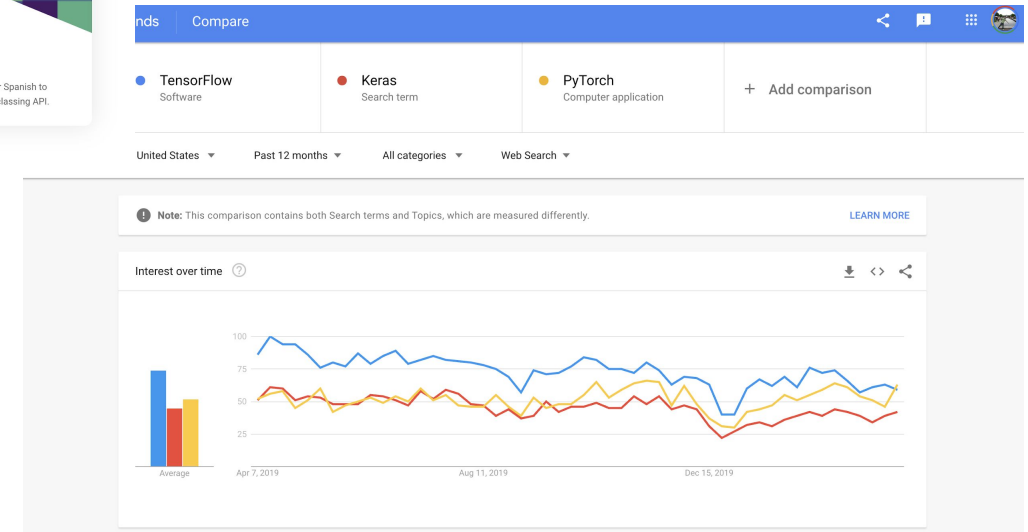




Source:
https://www.coursera.org/about/terms
https://www.coursera.org/
https://www.deeplearning.ai/

TensorFlow: An end-to-end open source machine learning platform





Source: https://www.tensorflow.org/

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**.

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

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

★★★★½ 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

COURSE 2

Convolutional Neural Networks in TensorFlow

★★★★½ 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

COURSE 3

Natural Language Processing in TensorFlow

★★★★½ 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

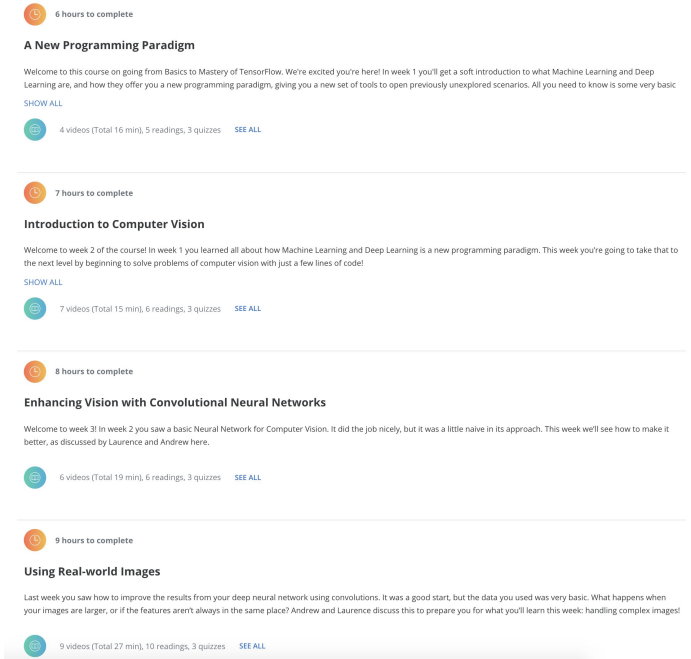
COURSE 4

Sequences, Time Series and Prediction

★★★★½ 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



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

TensorFlow in Practice Specialization

Prerequisites

- Programming in Python
- High school level math

Helpful but not required

- Basic linear algebra and calculus
- Basic deep learning concepts

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

Discussion with Laurence

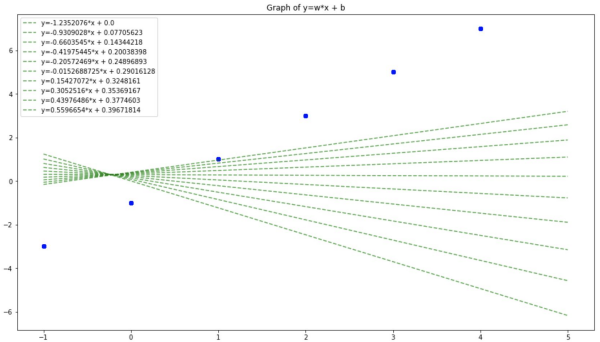
- Goals and challenges developing TensorFlow in Practice
- User engagement in this remote setup
- Other events you found interesting/cool
- Other deep learning frameworks
- Other specializations in mind - Data and Deployment
- TensorFlow Developer Exam: PyCharm vs Colab

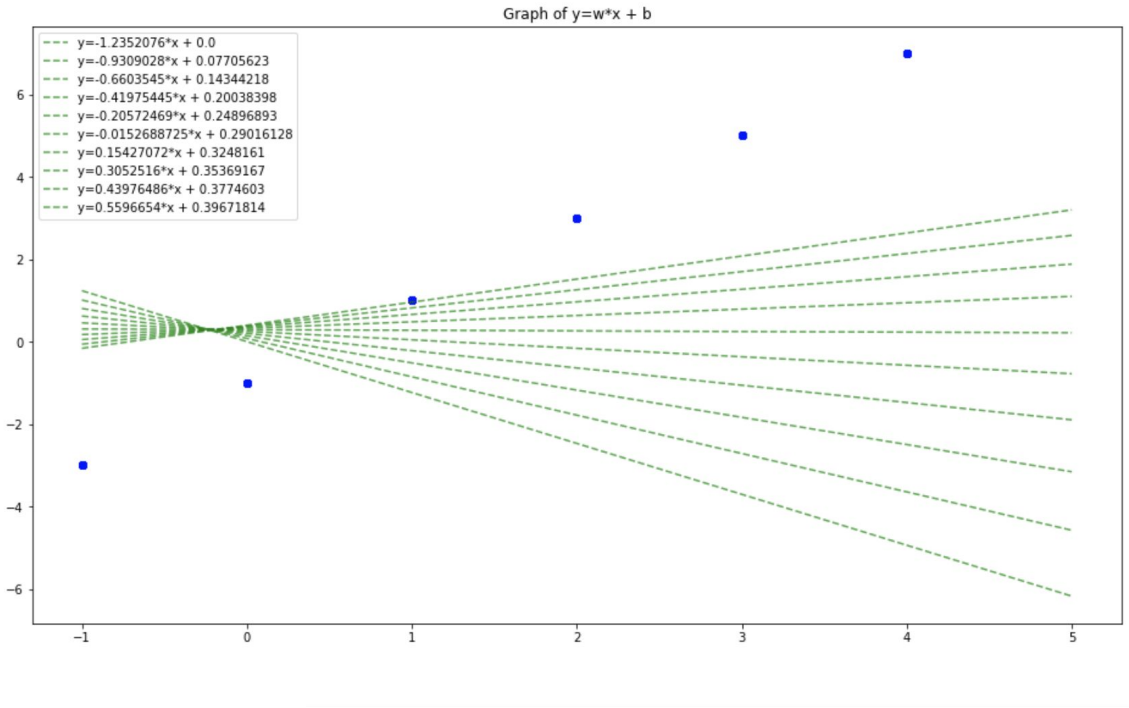
Course 1: Introduction to TensorFlow for Artificial Intelligence, Machine Learning, and Deep Learning

Week 1: A New Programming Paradigm
Week 2: Introduction to Computer Vision
Week 3: Enhancing Vision with Convolutional Neural Networks
Week 4: Using Real-world Images

Content added after our 1st session by George

```
model.fit(xs, ys, epochs=10, callbacks=[get_weight_and_bias()])

Epoch 1/10
-1.2352076 0.0
1/1 [==============================] - 0s 2ms/step - loss: 45.3716
Epoch 2/10
-0.9309028 0.07705623
1/1 [==============================] - 0s 2ms/step - loss: 36.0725
Epoch 3/10
-0.6603545 0.14344218
1/1 [==============================] - 0s 2ms/step - loss: 28.7486
Epoch 4/10
-0.41975445 0.20038398
1/1 [==============================] - 0s 1ms/step - loss: 22.9789
Epoch 5/10
-0.20572469 0.24896893
1/1 [==============================] - 0s 1ms/step - loss: 18.4323
Epoch 6/10
-0.0152688725 0.29016128
1/1 [==============================] - 0s 1ms/step - loss: 14.8479
Epoch 7/10
0.15427072 0.3248161
1/1 [==============================] - 0s 1ms/step - loss: 12.0208
Epoch 8/10
0.3052516 0.35369167
1/1 [==============================] - 0s 1ms/step - loss: 9.7895
Epoch 9/10
0.43976486 0.3774603
1/1 [==============================] - 0s 1ms/step - loss: 8.0273
Epoch 10/10
0.5596654 0.39671814
1/1 [==============================] - 0s 1ms/step - loss: 6.6342
<tensorflow.python.keras.callbacks.History at 0x7f77aee0da20>
```





Our code is available on GitHub and on our Slack channel:
https://github.com/georgezoto/TensorFlow-in-Practice/blob/master/Course-1-Introduction-to-TensorFlow/C1W1_A_new_programming_paradigm_1.ipynb

#tensorflow-in-practice-specialization

You created this channel on March 31st. This is the very beginning of the **#tensorflow-in-practice-specialization** channel.

An Introduction to computer vision

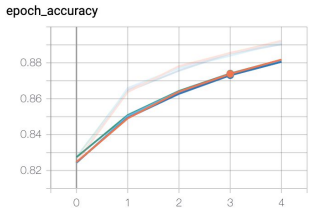
Extra Content - TensorBoard - issues in Colab

https://colab.research.google.com/github/tensorflow/tensorboard/blob/master/docs/scalars_and_keras.ipynb

```
# Define the Keras TensorBoard callback.
logdir = "logs/scalars/" + datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = keras.callbacks.TensorBoard(log_dir=logdir)
```

```
training_history = model.fit(training_images,
          training_labels,
          #batch_size = 64,
          #verbose = 0,
          epochs = 5,
          #validation_data = (test_images, test_labels),
          callbacks=[tensorboard_callback])
```

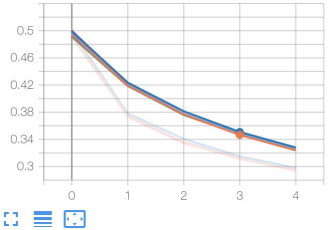
epoch_accuracy



Name	Smoothed	Value	Step	Time	Relative
20200410-165543/train	0.8738	0.8854	3	Fri Apr 10, 12:56:03	13s
20200410-171047/train	0.873	0.8852	3	Fri Apr 10, 13:11:06	12s
20200410-171729/train	0.8731	0.884	3	Fri Apr 10, 13:17:47	12s
20200410-175850/train	0.8738	0.8855	3	Fri Apr 10, 13:59:09	12s
20200410-180421/train	0.8741	0.8855	3	Fri Apr 10, 14:04:39	12s

Name	Smoothed	Value	Step	Time	Relative
20200410-165543/train	0.3466	0.3113	3	Fri Apr 10, 12:56:03	13s
20200410-171047/train	0.3509	0.3152	3	Fri Apr 10, 13:11:06	12s
20200410-171729/train	0.3489	0.3152	3	Fri Apr 10, 13:17:47	12s
20200410-175850/train	0.3475	0.3128	3	Fri Apr 10, 13:59:09	12s
20200410-180421/train	0.3474	0.3139	3	Fri Apr 10, 14:04:39	12s

epoch_loss

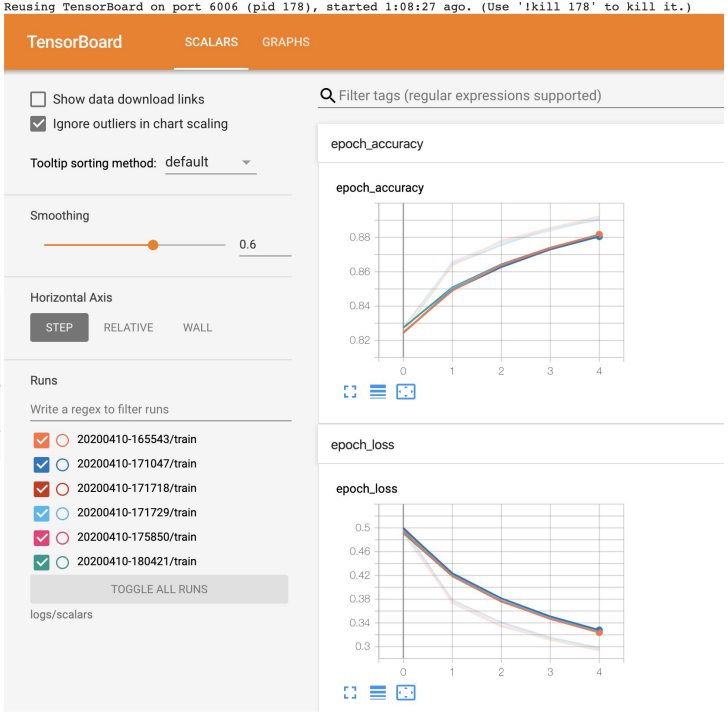


Op-level graph

Start TensorBoard and wait a few seconds for the UI to load. Select the Graphs dashboard by tapping "Graphs" at the top.

```
[15] %tensorboard --logdir logs/scalars
```

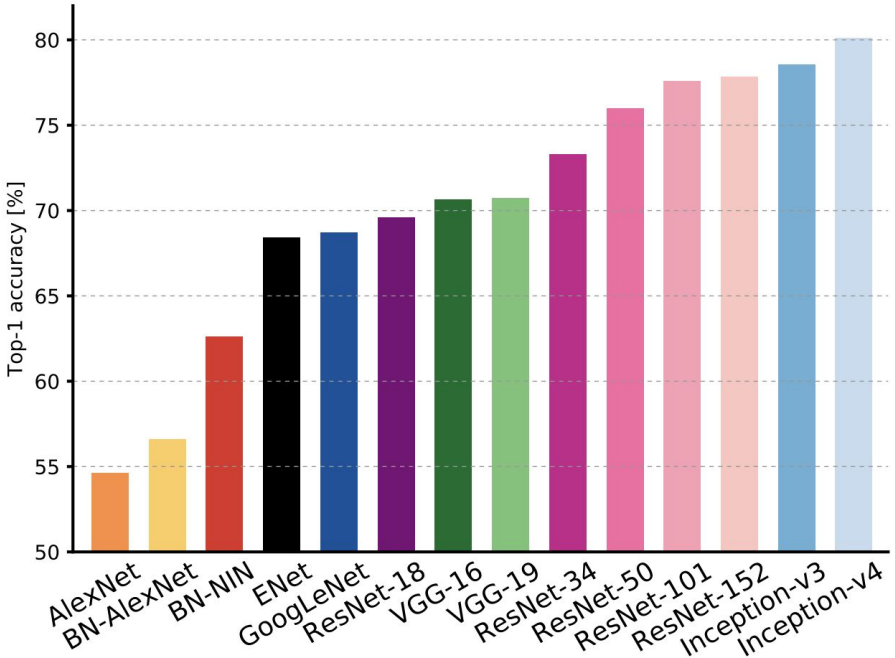
```
Reusing TensorBoard on port 6006 (pid 178), started 1:08:27 ago. (Use '!kill 178' to kill it.)
```

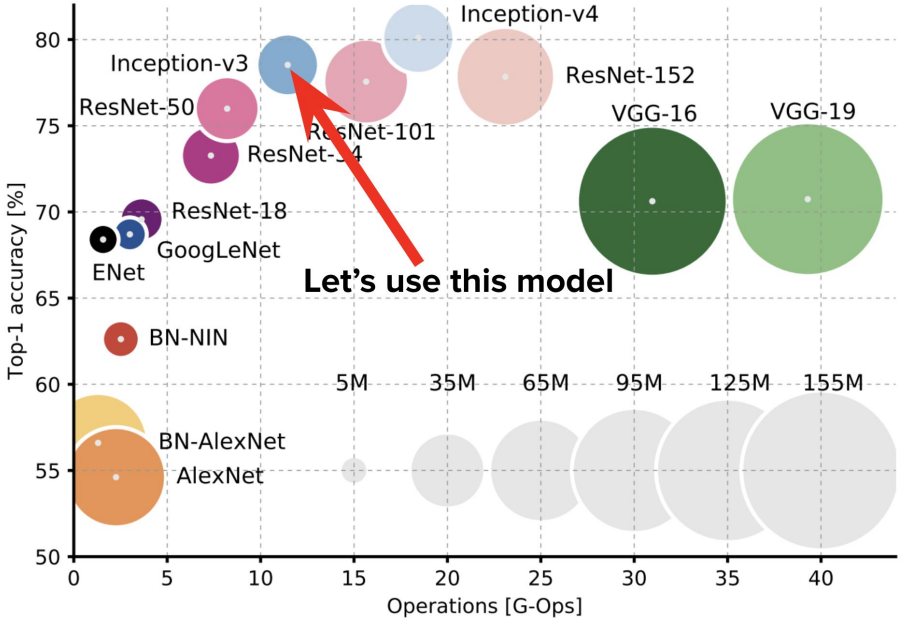


Course 2: Convolutional Neural Networks in TensorFlow

Week 1: Exploring a Larger Dataset
Week 2: Augmentation: A technique to avoid overfitting
Week 3: Transfer Learning
Week 4: Multiclass Classifications

Transfer Learning - Extra Content - Classic Networks by David





Source: https://arxiv.org/pdf/1605.07678.pdf

Extra Content - Regularization

Logistic regression

$$\min_{w,b} J(w,b)$$

$w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$

λ = regularization parameter (lambda / lambd)

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\|w\|_2^2$$

$+ \frac{\lambda}{2m} b^2$ (omit)

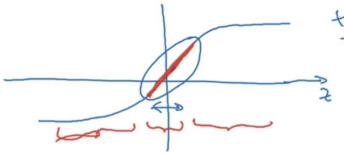
L2 regularization: $\|w\|_2^2 = \sum_{j=1}^{n_x} w_j^2 = w^T w$

L1 regularization: $\frac{\lambda}{2m}\sum_{j=1}^{n_x} |w_j| = \frac{\lambda}{2m}\|w\|_1$

w will be sparse

Andrew Ng

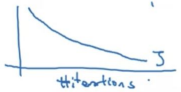
How does regularization prevent overfitting?

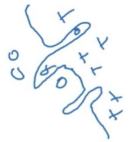


$\lambda \uparrow$ $w^{[l]} \downarrow$ $z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]}$

Every layer $\approx$ linear.

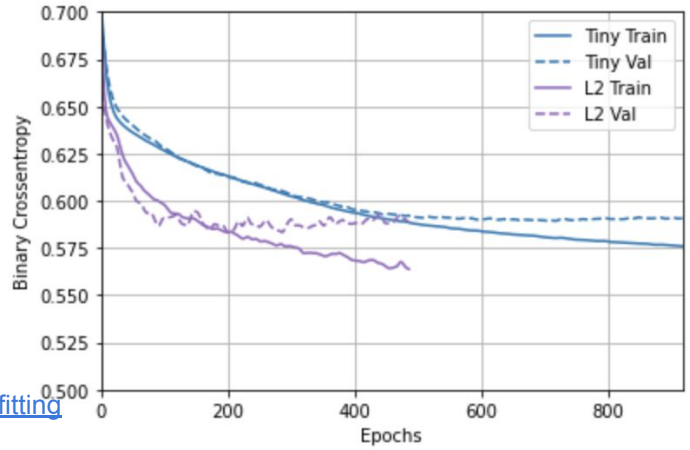
$$J(\dots) = \sum_i \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{1}{2m}\sum_l \|w^{[l]}\|_F^2$$





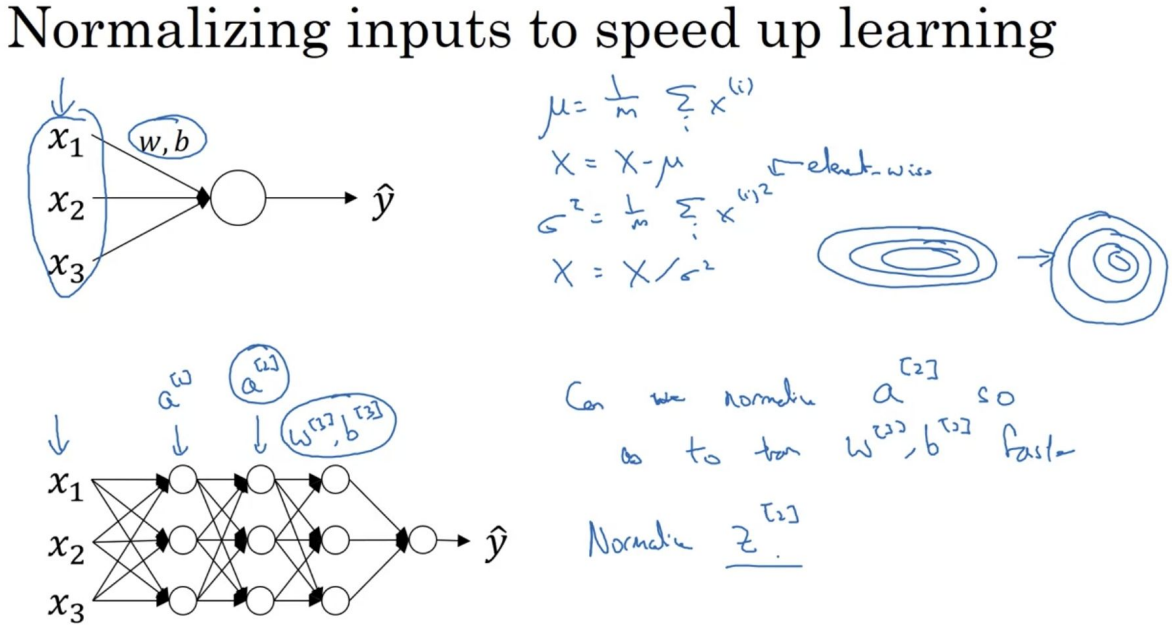
Andrew Ng

```
from tensorflow.keras import regularizers

l2_model = tf.keras.Sequential([
    layers.Dense(512, activation='elu',
                 kernel_regularizer=regularizers.l2(0.001),
                 input_shape=(FEATURES,)),
    layers.Dense(512, activation='elu',
                 kernel_regularizer=regularizers.l2(0.001)),
    layers.Dense(512, activation='elu',
                 kernel_regularizer=regularizers.l2(0.001)),
    layers.Dense(512, activation='elu',
                 kernel_regularizer=regularizers.l2(0.001)),
    layers.Dense(1)
])
```



Source: https://www.coursera.org/learn/deep-neural-network/lecture/Srsrc/regularization
https://www.coursera.org/learn/deep-neural-network/lecture/T6OJj/why-regularization-reduces-overfitting
https://www.tensorflow.org/tutorials/keras/overfit_and_underfit

Extra Content - Batch Normalization



TensorFlow > API > TensorFlow Core v2.1.0 > Python

tf.keras.layers.BatchNormalization

```
tf.keras.layers.BatchNormalization(
```

Normalize the activations of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1.

Batch normalization differs from other layers in several key aspects:

Source:
https://www.coursera.org/learn/deep-neural-network/lecture/4ptp2/normalizing-activations-in-a-network
https://www.tensorflow.org/api_docs/python/tf/keras/layers/BatchNormalization

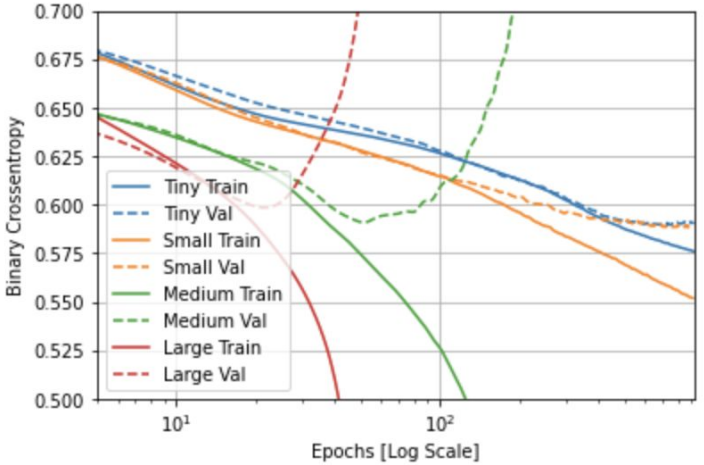
Extra Content - Overfitting in Neural Networks

To recap: here are the most common ways to prevent overfitting in neural networks:

- Get more training data.
- Reduce the capacity of the network.
- Add weight regularization.
- Add dropout.

The Higgs Dataset: 11M samples, 28 features, binary class

Total params:
Tiny: 0.5K
Small: 0.7K
Medium: 10K
Large: 800K

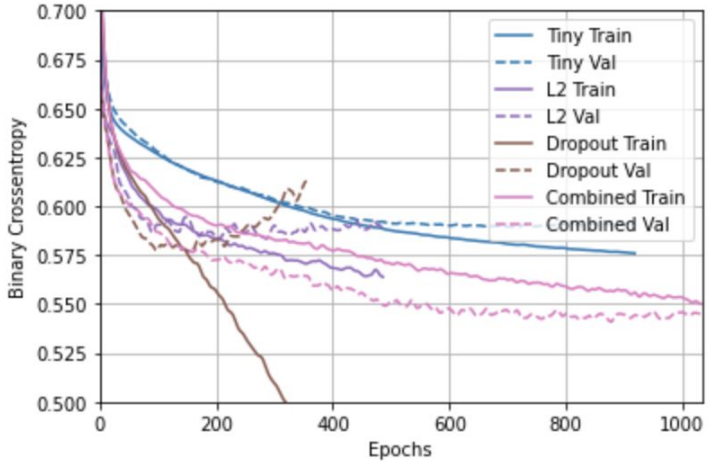


Two important approaches not covered in this guide are:

- data-augmentation
- batch normalization

Remember that each method can help on its own, but often combining them can be even more effective.

- Data Augmentation
- Dropout
- Early Stopping
- Ensembling
- Injecting Noise
- L1 Regularization
- L2 Regularization



Source: https://www.tensorflow.org/tutorials/keras/overfit_and_underfit
https://ml-cheatsheet.readthedocs.io/en/latest/regularization.html

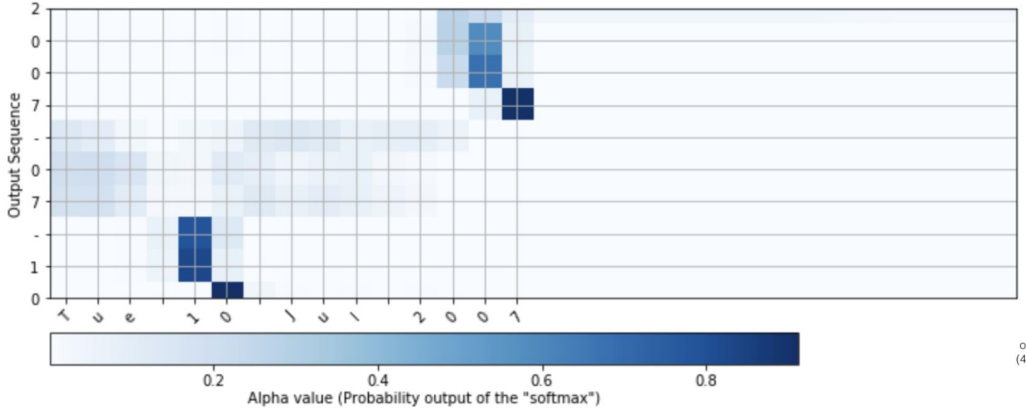
Course 3: Natural Language Processing in TensorFlow

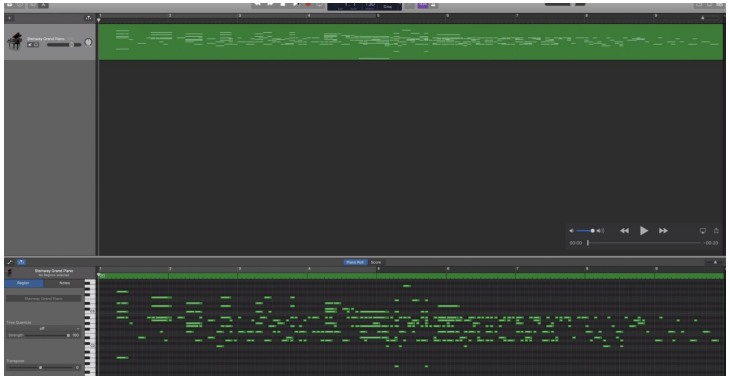
Week 1: Sentiment in text
Week 2: Word Embeddings
Week 3: Sequence models
Week 4: Sequence models and literature

Extra Content - cool ways to use RNNs, Bias, Attention, Text and Vision by George

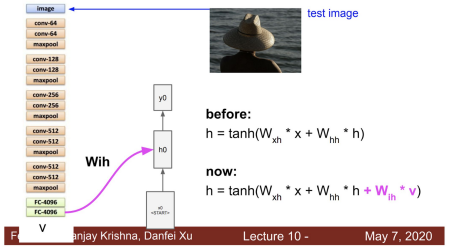
```
text = EXAMPLES[3]
pre_plot(model, human_vocab, inv_machine_vocab, text, num = 7, n_s = 64);
```

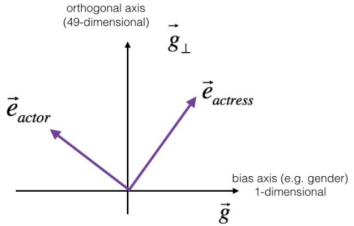
```
Input Tue 10 Jul 2007
Output 2007-07-10
```



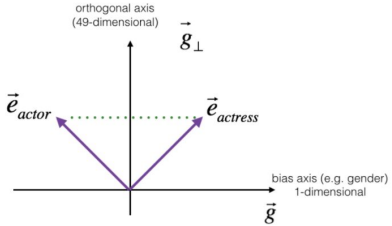


Sequence models and computer vision





before equalizing,
"actress" and "actor" differ in many ways beyond the direction of $\vec{g}$



after equalizing,
"actress" and "actor" differ only in the direction of $\vec{g}$, and further are equal in distance from $\vec{g}_\perp$

Source: https://github.com/georgezoto/RNN-LSTM-NLP-Sequence-Models

Extra Content - Deep Learning for Structured Data

TensorFlow Install Learn API Resources Community Why TensorFlow Search

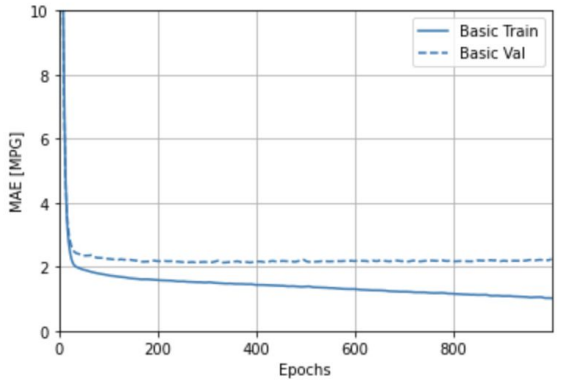
- TensorFlow tutorials
- Quickstart for beginners
- Quickstart for experts

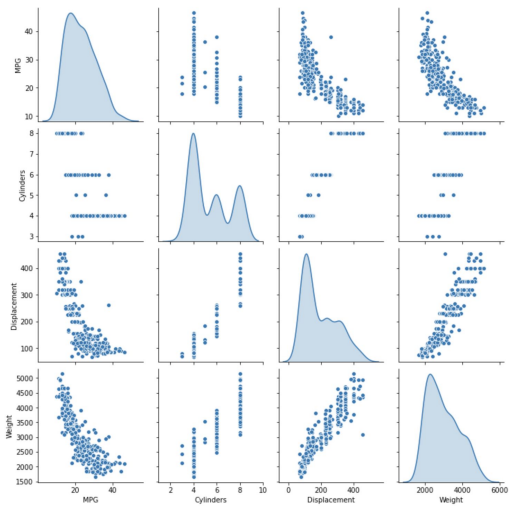
BEGINNER

ML basics with Keras
- Basic image classification
- Text classification with TF Hub
- Text classification with preprocessed text
- Regression
- Overfit and underfit

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin
393	27.0	4	140.0	86.0	2790.0	15.6	82	1
394	44.0	4	97.0	52.0	2130.0	24.6	82	2
395	32.0	4	135.0	84.0	2295.0	11.6	82	1
396	28.0	4	120.0	79.0	2625.0	18.6	82	1
397	31.0	4	119.0	82.0	2720.0	19.4	82	

```
def build_model():
  model = keras.Sequential([
    layers.Dense(64, activation='relu', input_shape=[len(tra
    layers.Dense(64, activation='relu'),
    layers.Dense(1)
  ])

  optimizer = tf.keras.optimizers.RMSprop(0.001)

  model.compile(loss='mse',
                optimizer=optimizer,
                metrics=['mae', 'mse'])
  return model
```





Source: https://www.tensorflow.org/tutorials/keras/regression

Course 4: Sequences, Time Series and Prediction

Week 1: Sequences and Prediction
Week 2: Deep Neural Networks for Time Series
Week 3: Recurrent Neural Networks for Time Series
Week 4: Real-world time series data

Laurence’s LR optimization? by David

For consistent results:

```
import random
seed = 51
tf.random.set_seed(seed)
random.seed = seed
```

TensorFlow > API > TensorFlow Core v2.2.0 > Python

tf.keras.callbacks.ReduceLROnPlateau

```
tf.keras.callbacks.ReduceLROnPlateau(
    monitor='val_loss', factor=0.1, patience=10, verbose=0, mode='auto',
    min_delta=0.0001, cooldown=0, min_lr=0, **kwargs
)
```

Models often benefit from reducing the learning rate by a factor of 2-10 once learning stagnates. This callback monitors a quantity and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced.

Set all runs to epochs=500

Restart runtime and run all each test.

Source: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ReduceLROnPlateau

Laurence's LR optimization results (seed=51)

1. `optimizer=tf.keras.optimizers.SGD(lr=1e-6, momentum=0.9)`

   `mae  4.500836`

2. `optimizer = tf.keras.optimizers.SGD(lr=4e-6, momentum=0.9)`

   `mae  4.8579106`

3. `optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)`

   `mae  5.3804765`

4. `ReduceLROnPlateau(monitor='loss', patience=25, verbose=1)`

   `optimizer = tf.keras.optimizers.Adam(learning_rate=0.01)`

   `mae  4.490887`

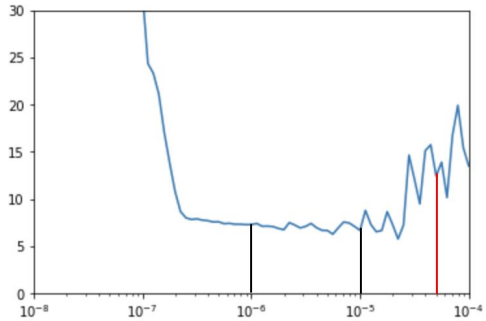
Recurrent Neural Networks for Time Series by Robert

```
# Create the series
series = baseline + trend(time, slope) + seasonality(time, period=365, amplit
# Update with noise
series += noise(time, noise_level, seed=42)

split_time = 1000
time_train = time[:split_time]
x_train = series[:split_time]
time_valid = time[split_time:]
x_valid = series[split_time:]

window_size = 20
batch_size = 32
shuffle_buffer_size = 1000

my_epoch = 100
```

```
time = np.arange(4 * 365 + 1,
baseline = 10
series = trend(time, 0.1)
baseline = 10
amplitude = 40
slope = 0.05
noise_level = 5
```

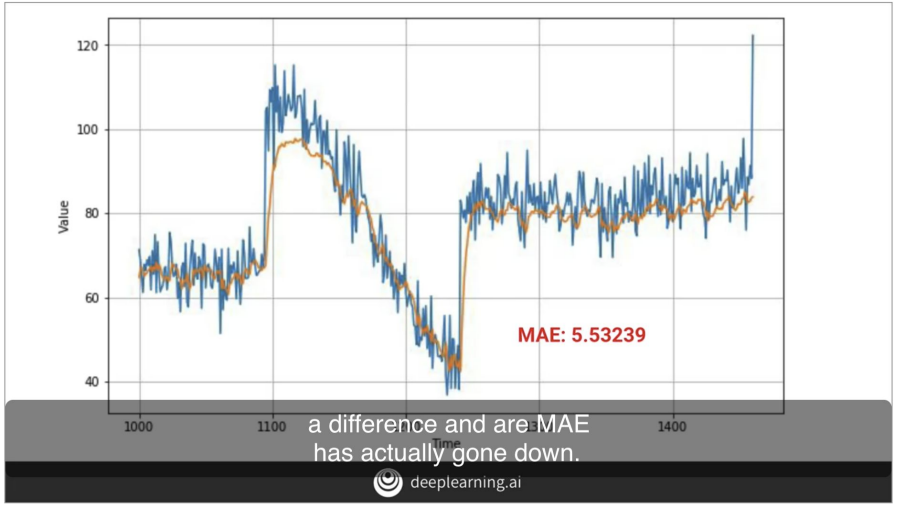
```
plt.semilogx(history.history["lr"], history.history["loss"])
plt.axis([1e-8, 1e-4, 0, 30])
```

```
(1e-08, 0.0001, 0.0, 30.0)
```




```
tf.keras.backer
tf.random.set_s
np.random.seed(

train_set = win

model = tf.kera
  tf.keras.laye

  tf.keras.laye
  tf.keras.laye
  tf.keras.laye
  tf.keras.laye
])

lr_schedule = t
    lambda epoc
optimizer = tf.
model.compile(l
                c
                m
history = model
```

Coding LSTMs



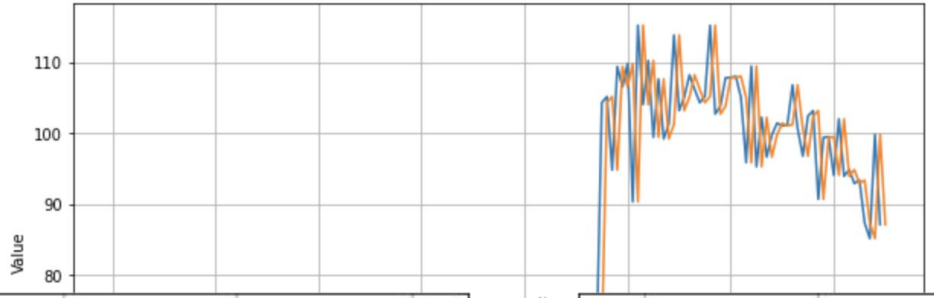

```
optimizer = tf.keras.optimizers.SGD(lr=5e-5, momentum=0.9)
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])
history = model.fit(dataset,epochs=my_epoch*4)
```

Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

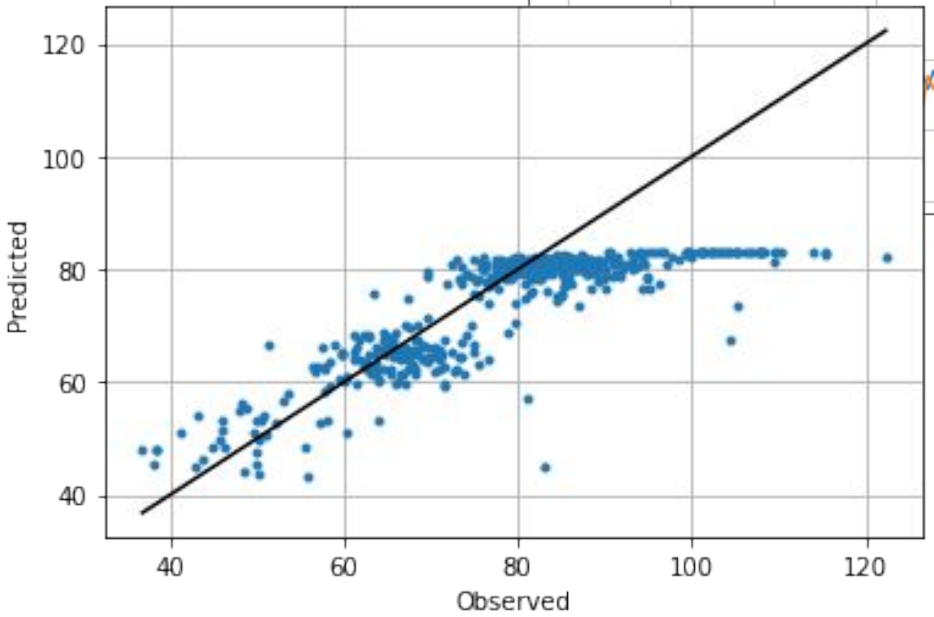
Errata 5*10-6: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction/lecture/5W1Rw/rnn

Errata MAE has gone down: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction/lecture/8u3wq/coding-lstms

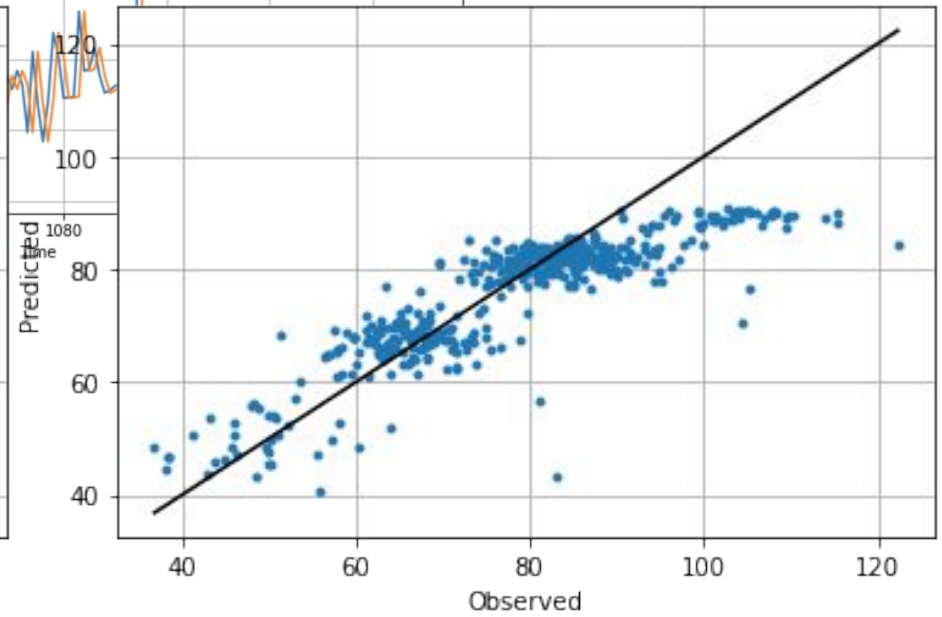
Recurrent Neural Networks for Time Series



LR: 5e-5



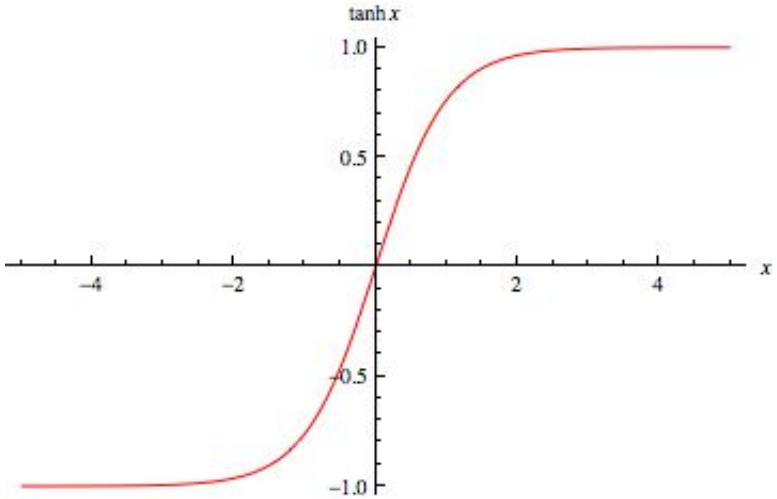
LR: 3e-6



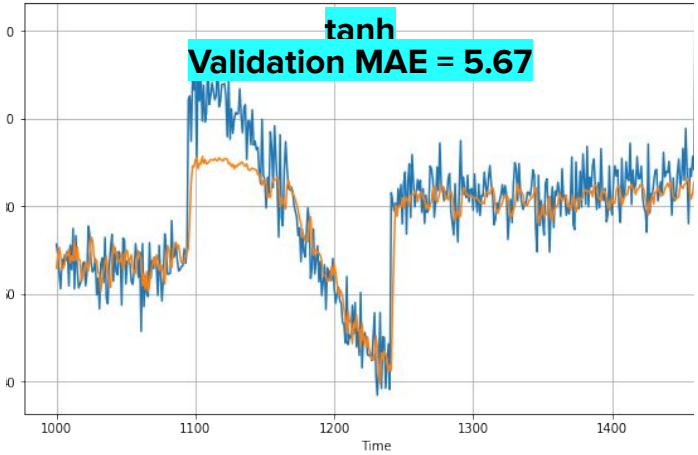
Recurrent Neural Networks for Time Series

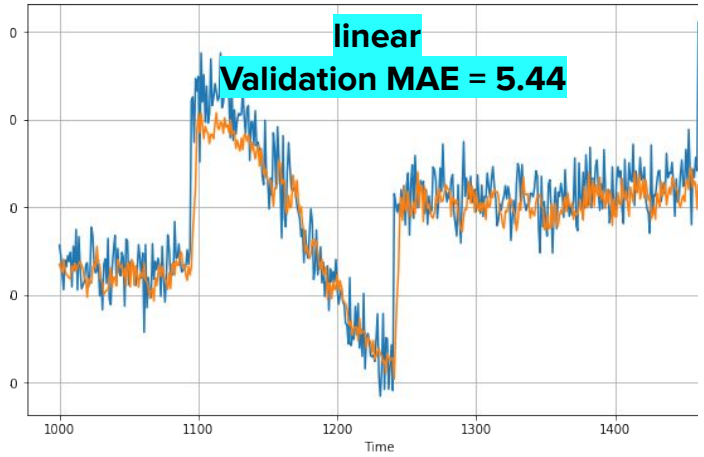
TensorFlow

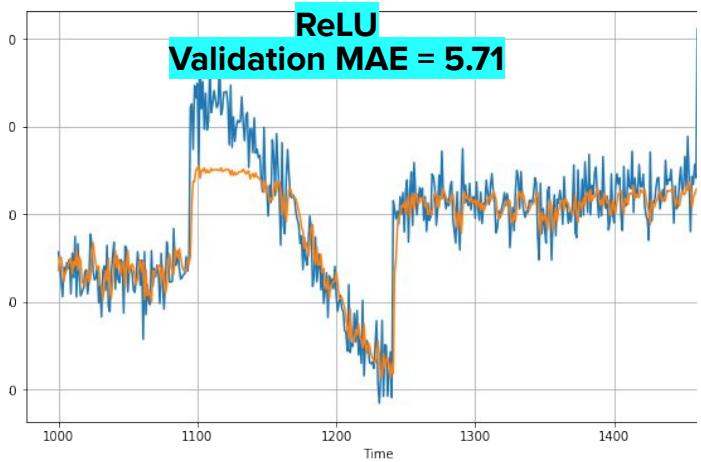
```
tf.keras.layers.SimpleRNN(
    units, activation='tanh', use_bias=True, kernel_initializer='glor
    recurrent_initializer='orthogonal', bias_initializer='zeros',
    kernel_regularizer=None, recurrent_regularizer=None, bias_regular
    activity_regularizer=None, kernel_constraint=None, recurrent_cons
```

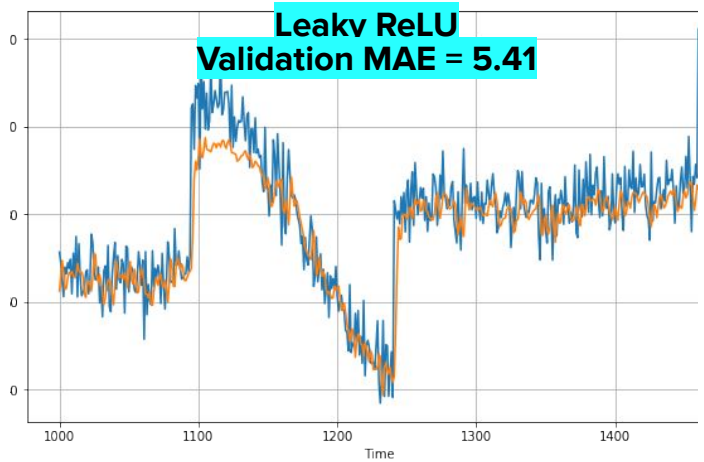


Recurrent Neural Networks for Time Series

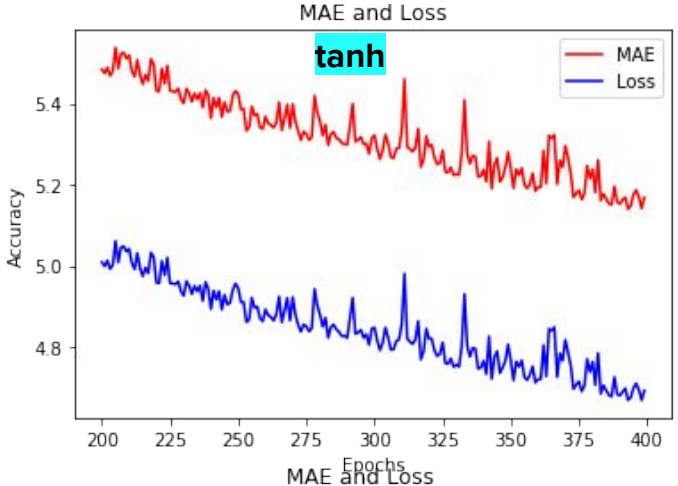


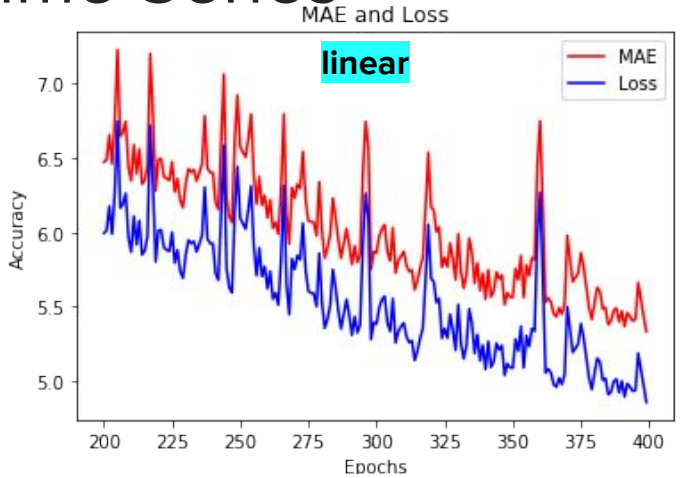


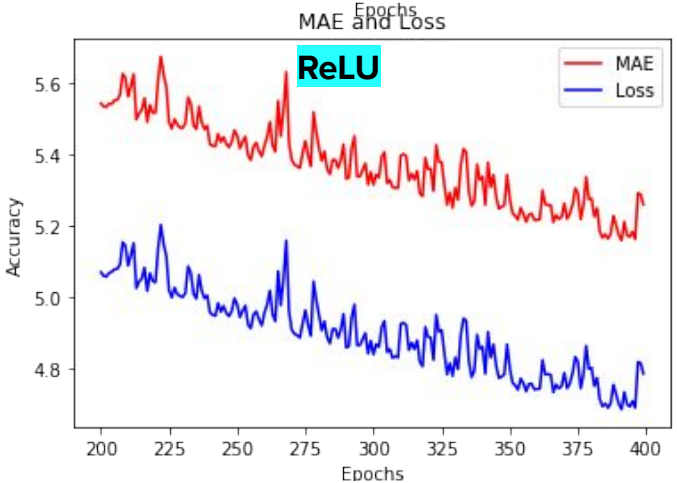


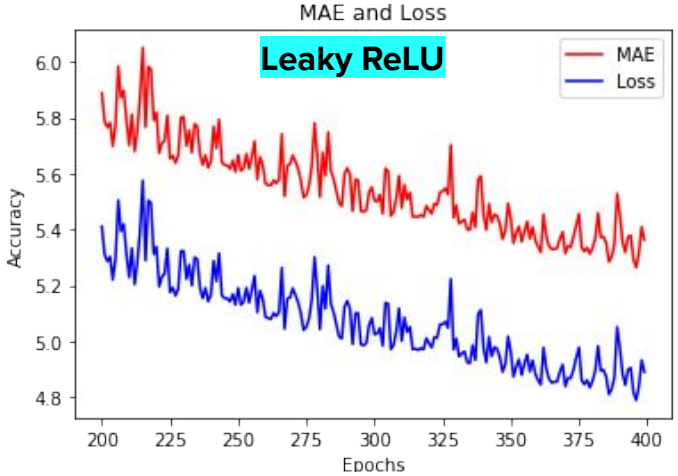


Recurrent Neural Networks for Time Series









Appendix:
A very quick overview of TensorFlow in Practice Specialization

Course 1: Introduction to TensorFlow for Artificial Intelligence, Machine Learning, and Deep Learning

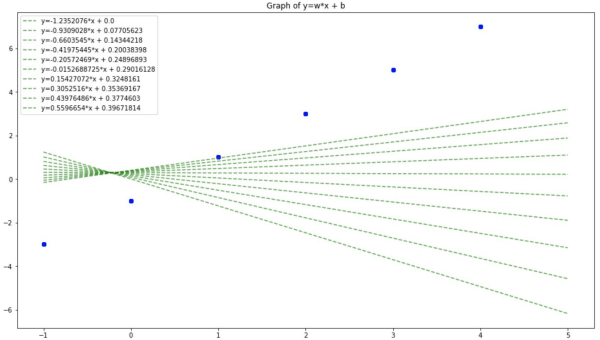
Week 1: A New Programming Paradigm
Week 2: Introduction to Computer Vision
Week 3: Enhancing Vision with Convolutional Neural Networks
Week 4: Using Real-world Images

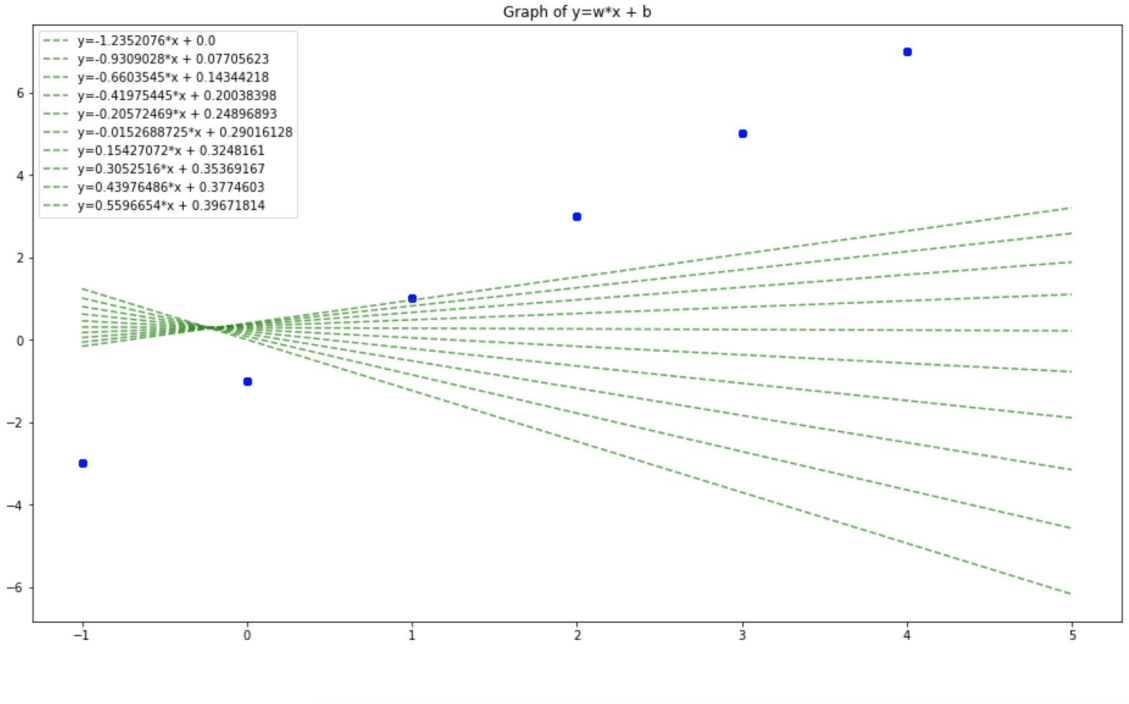
A New Programming Paradigm

```
model = keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)

model.fit(xs, ys, epochs=500)

print(model.predict([10.0]))
```

Content added after our 1st session, based on group feedback

```
model.fit(xs, ys, epochs=10, callbacks=[get_weight_and_bias()])

Epoch 1/10
-1.2352076 0.0
1/1 [==============================] - 0s 2ms/step - loss: 45.3716
Epoch 2/10
-0.9309028 0.07705623
1/1 [==============================] - 0s 2ms/step - loss: 36.0725
Epoch 3/10
-0.6603545 0.14344218
1/1 [==============================] - 0s 2ms/step - loss: 28.7486
Epoch 4/10
-0.41975445 0.20038398
1/1 [==============================] - 0s 1ms/step - loss: 22.9789
Epoch 5/10
-0.20572469 0.24896893
1/1 [==============================] - 0s 1ms/step - loss: 18.4323
Epoch 6/10
-0.0152688725 0.29016128
1/1 [==============================] - 0s 1ms/step - loss: 14.8479
Epoch 7/10
0.15427072 0.3248161
1/1 [==============================] - 0s 1ms/step - loss: 12.0208
Epoch 8/10
0.3052516 0.35369167
1/1 [==============================] - 0s 1ms/step - loss: 9.7895
Epoch 9/10
0.43976486 0.3774603
1/1 [==============================] - 0s 1ms/step - loss: 8.0273
Epoch 10/10
0.5596654 0.39671814
1/1 [==============================] - 0s 1ms/step - loss: 6.6342
<tensorflow.python.keras.callbacks.History at 0x7f77aee0da20>
```





Our code is available on GitHub and on our Slack channel:
https://github.com/georgezoto/TensorFlow-in-Practice/blob/master/Course-1-Introduction-to-TensorFlow/C1W1_A_new_programming_paradigm_1.ipynb

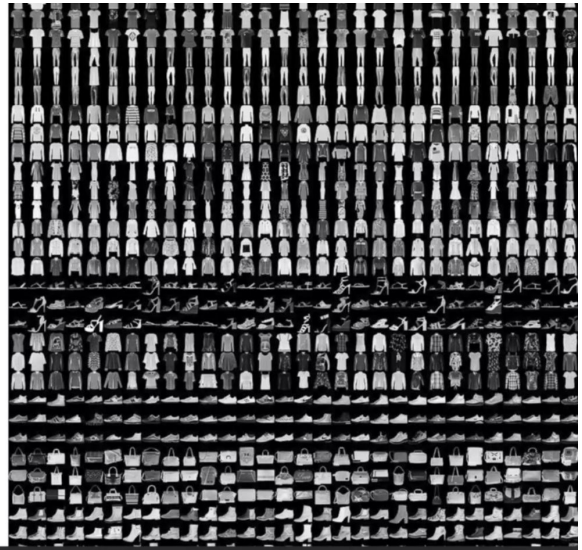
#tensorflow-in-practice-specialization

You created this channel on March 31st. This is the very beginning of the **#tensorflow-in-practice-specialization** channel.

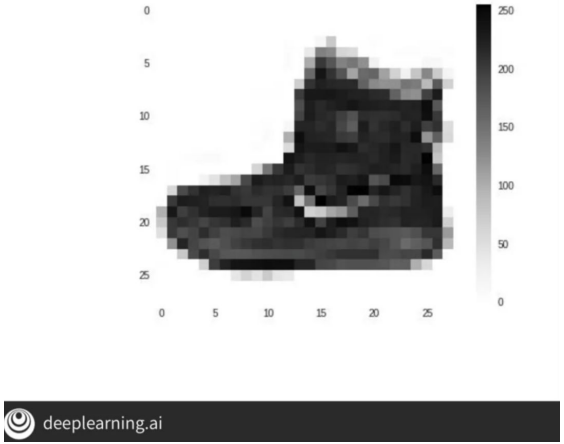
An Introduction to computer vision

Fashion MNIST

- 70k Images
- 10 Categories
- Images are 28x28
- Can train a neural net!

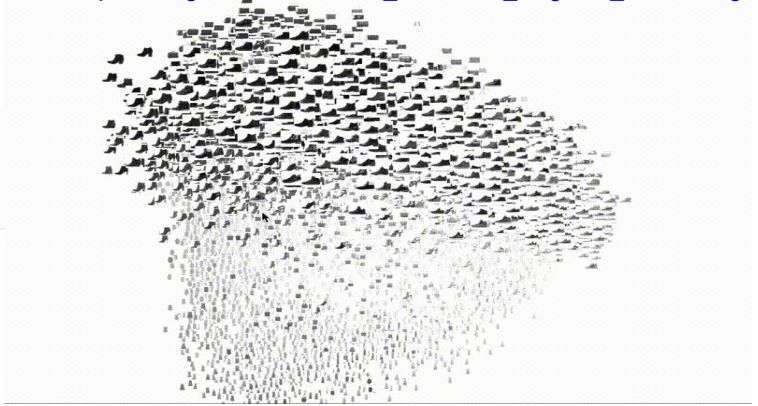


Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

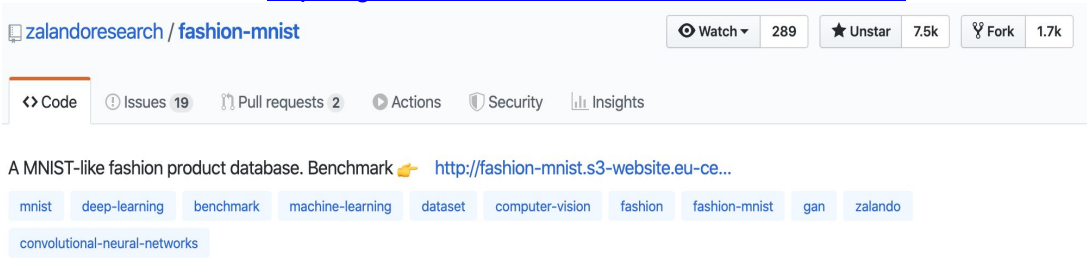


Visualization: t-SNE

https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding



Fashion MNIST: https://github.com/zalandoresearch/fashion-mnist



An Introduction to computer vision

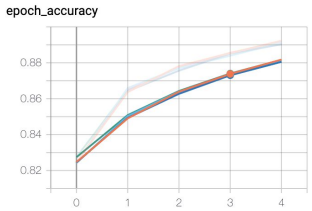
Extra Content - TensorBoard - issues in Colab

https://colab.research.google.com/github/tensorflow/tensorboard/blob/master/docs/scalars_and_keras.ipynb

```
# Define the Keras TensorBoard callback.
logdir = "logs/scalars/" + datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = keras.callbacks.TensorBoard(log_dir=logdir)
```

```
training_history = model.fit(training_images,
          training_labels,
          #batch_size = 64,
          #verbose = 0,
          epochs = 5,
          #validation_data = (test_images, test_labels),
          callbacks=[tensorboard_callback])
```

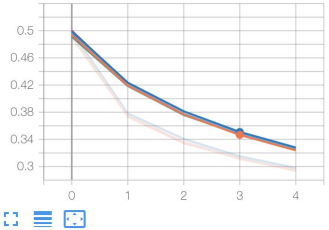
epoch_accuracy



Name	Smoothed	Value	Step	Time	Relative
20200410-165543/train	0.8738	0.8854	3	Fri Apr 10, 12:56:03	13s
20200410-171047/train	0.873	0.8852	3	Fri Apr 10, 13:11:06	12s
20200410-171729/train	0.8731	0.884	3	Fri Apr 10, 13:17:47	12s
20200410-175850/train	0.8738	0.8855	3	Fri Apr 10, 13:59:09	12s
20200410-180421/train	0.8741	0.8855	3	Fri Apr 10, 14:04:39	12s

Name	Smoothed	Value	Step	Time	Relative
20200410-165543/train	0.3466	0.3113	3	Fri Apr 10, 12:56:03	13s
20200410-171047/train	0.3509	0.3152	3	Fri Apr 10, 13:11:06	12s
20200410-171729/train	0.3489	0.3152	3	Fri Apr 10, 13:17:47	12s
20200410-175850/train	0.3475	0.3128	3	Fri Apr 10, 13:59:09	12s
20200410-180421/train	0.3474	0.3139	3	Fri Apr 10, 14:04:39	12s

epoch_loss

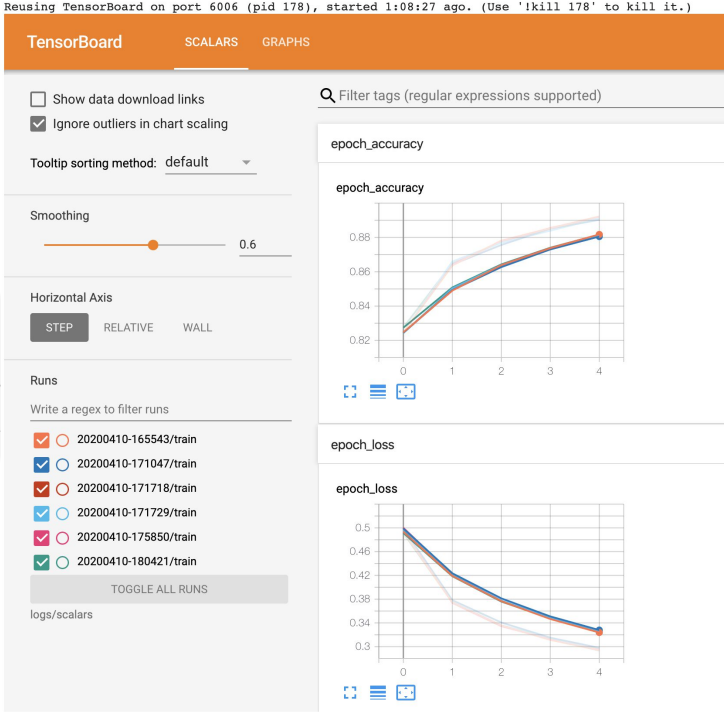


Op-level graph

Start TensorBoard and wait a few seconds for the UI to load. Select the Graphs dashboard by tapping "Graphs" at the top.

```
[15] %tensorboard --logdir logs/scalars
```

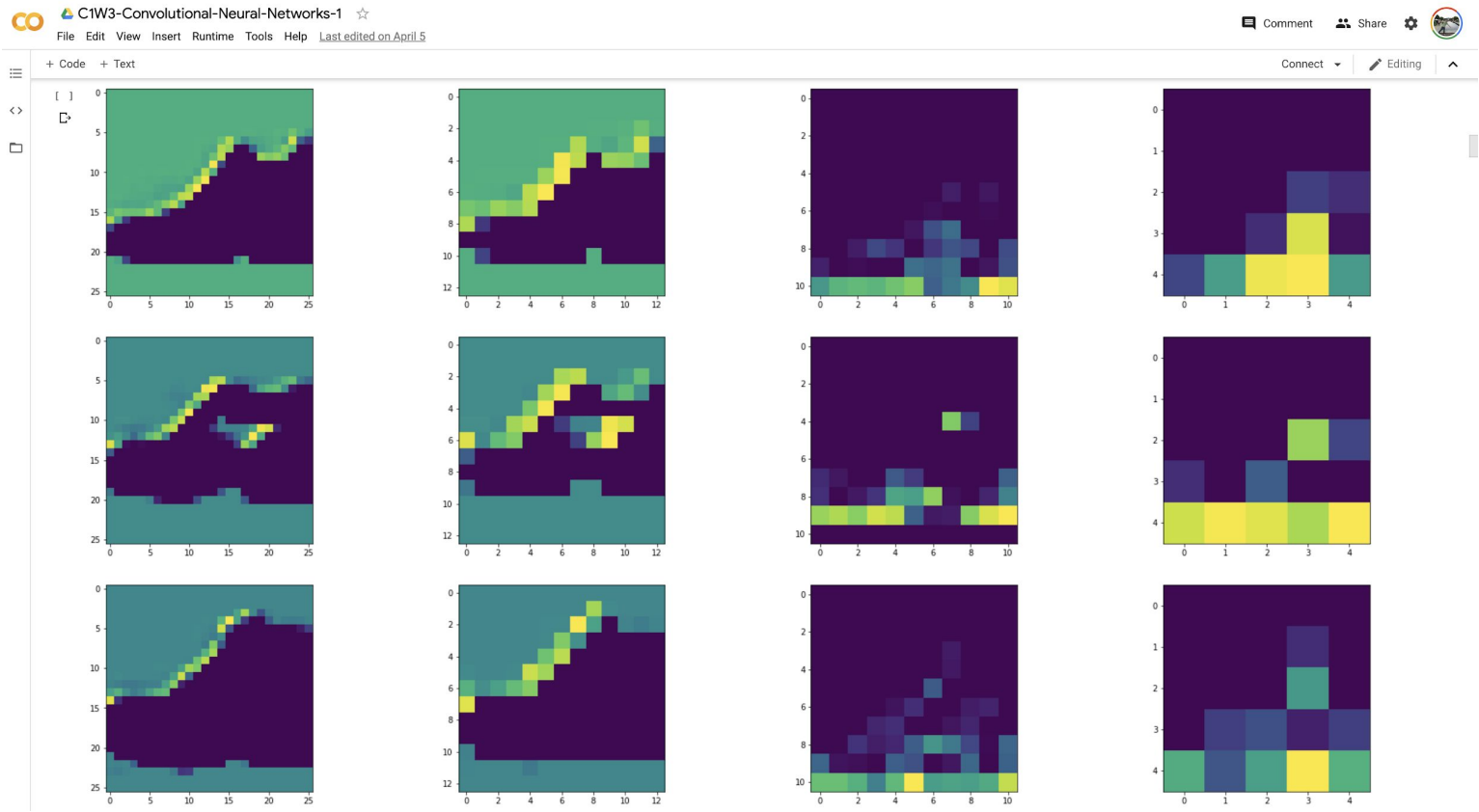
```
Reusing TensorBoard on port 6006 (pid 178), started 1:08:27 ago. (Use '!kill 178' to kill it.)
```



As you can see we go from the raw pixels of the images to increasingly abstract and compact representations. The representations downstream start highlighting what the network pays attention to, and they show fewer and fewer features being "activated"; most are set to zero. This is called "sparsity." Representation sparsity is a key feature of deep learning.

These representations carry increasingly less information about the original pixels of the image, but increasingly refined information about the class of the image. You can think of a convnet (or a deep network in general) as an information distillation pipeline.

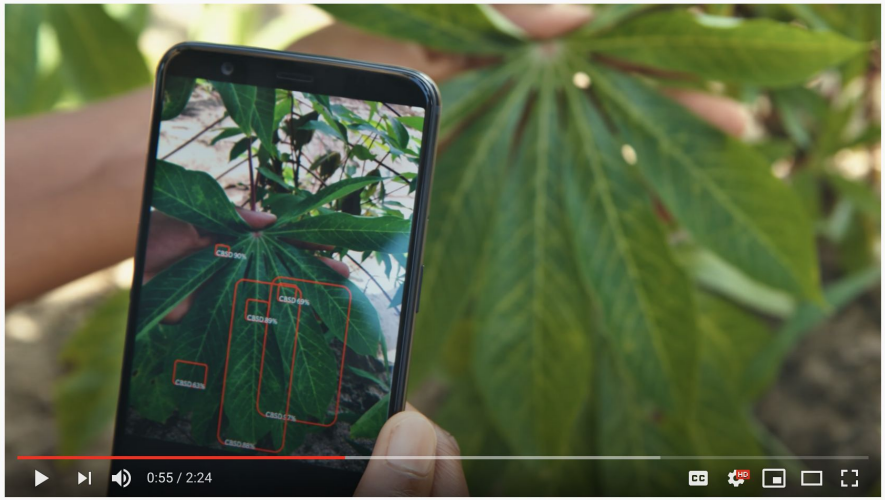
Enhancing Vision with Convolutional Neural Networks - Colab



Check out these resources

TensorFlow: an ML platform for solving impactful and challenging problems

105,225 views • Mar 30, 2018 2.3K 25 SHARE SAVE



TensorFlow: an ML platform for solving impactful and challenging problems

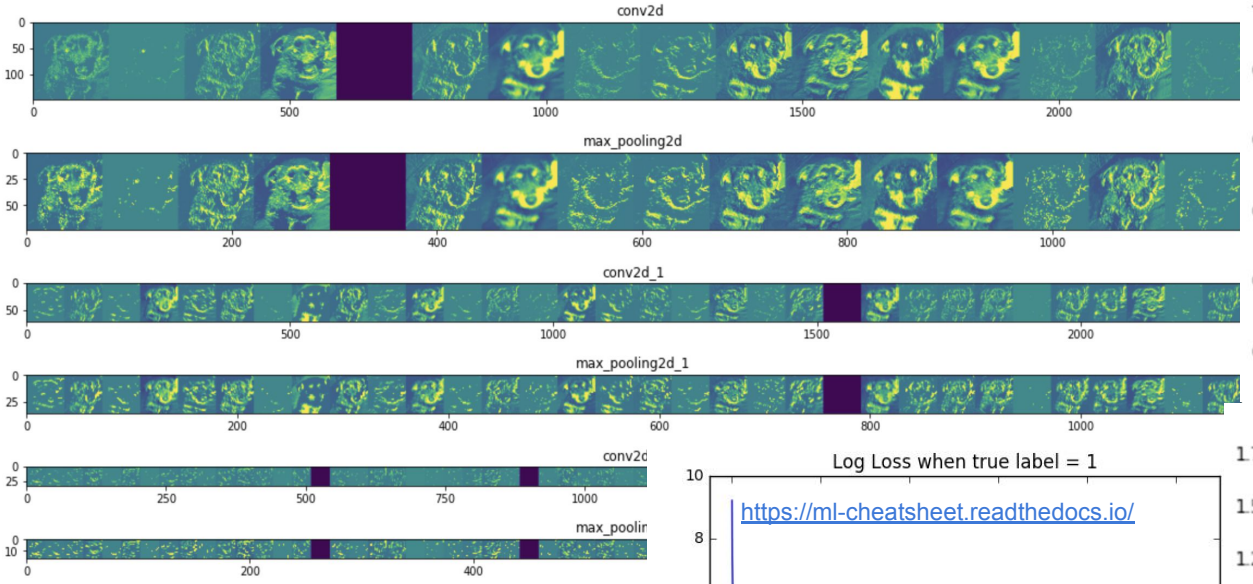
105,225 views • Mar 30, 2018 2.3K 25 SHARE SAVE

Source: https://www.youtube.com/watch?v=NIpS-DhayQA
https://plantvillage.psu.edu/
https://www.iita.org/
https://www.blog.google/technology/ai/ai-takes-root-helping-farmers-identity-diseased-plants/

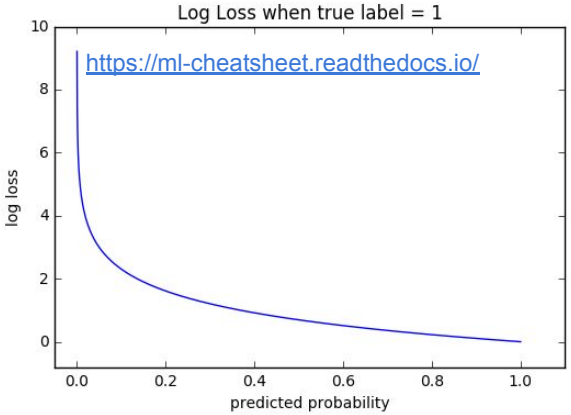
Course 2: Convolutional Neural Networks in TensorFlow

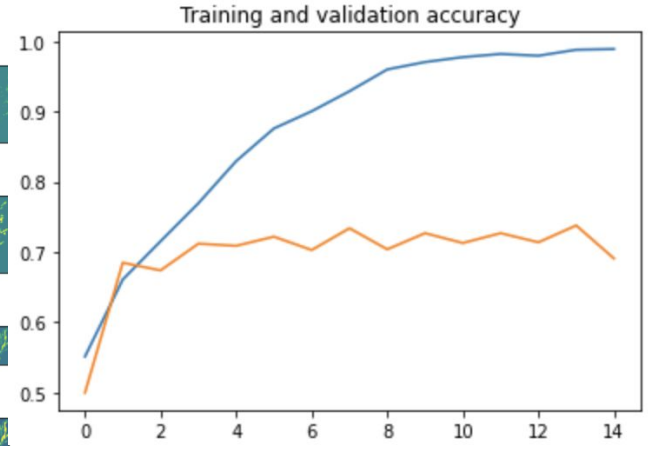
Week 1: Exploring a Larger Dataset
Week 2: Augmentation: A technique to avoid overfitting
Week 3: Transfer Learning
Week 4: Multiclass Classifications

Exploring a Larger Dataset



Bug fix: [layer.output for layer in model.layers[1:]]

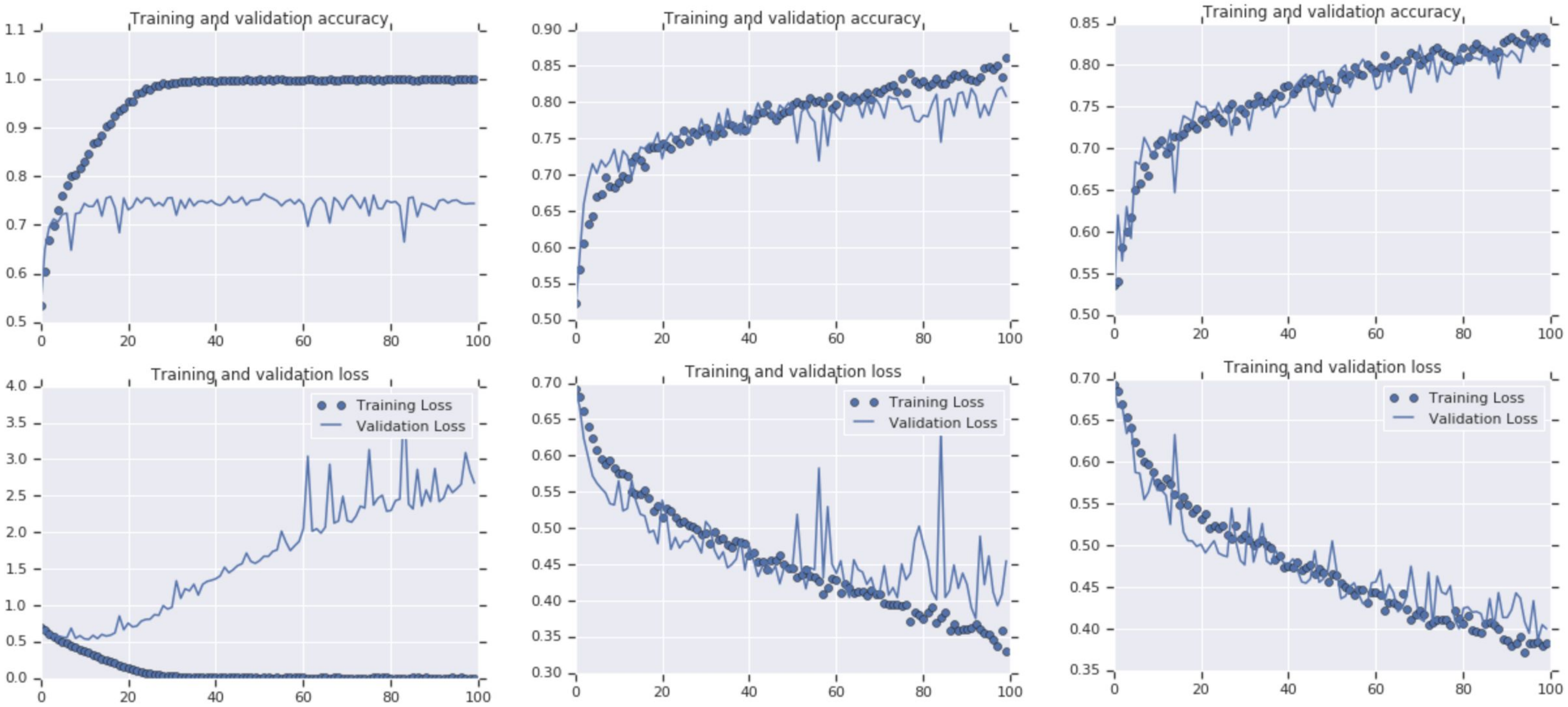




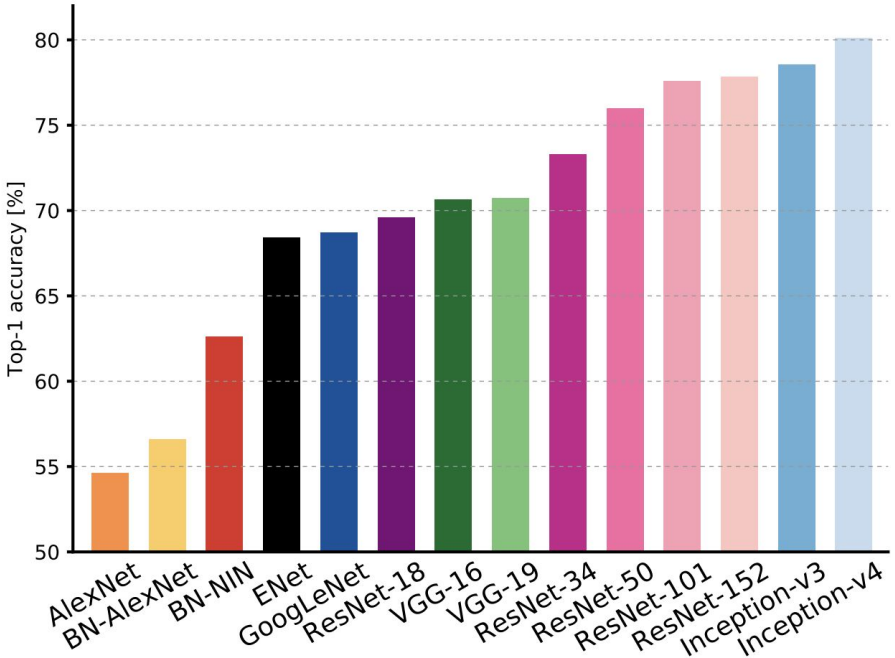
Model is badly overfitting!

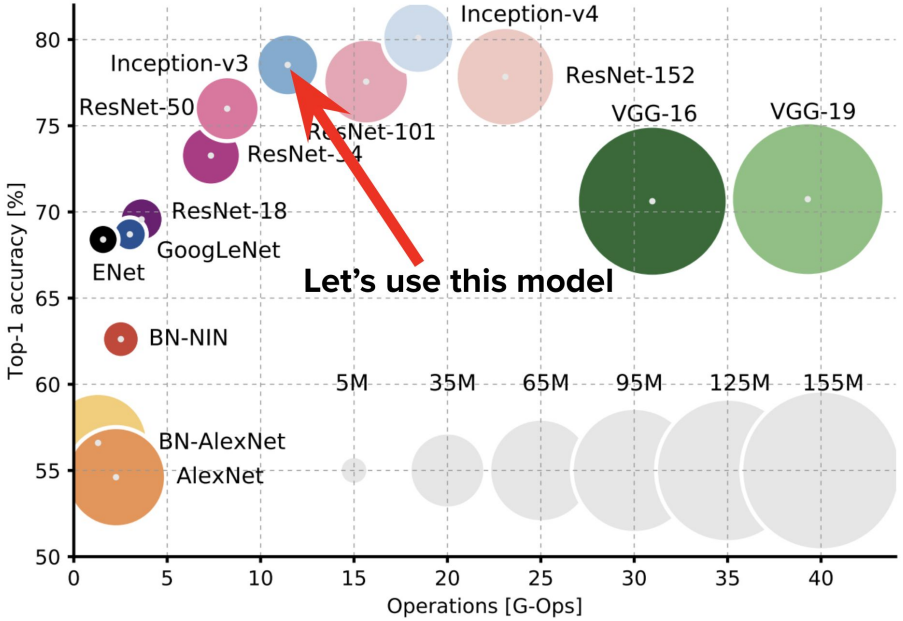
Training Loss Decreasing
Validation Loss Increasing

Augmentation and Dropout: Techniques to avoid overfitting



Transfer Learning - Extra Content - Classic Networks





Source: https://arxiv.org/pdf/1605.07678.pdf

Extra Content - Regularization

Logistic regression

$$\min_{w,b} J(w,b)$$

$w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$

λ = regularization parameter (lambda / lambd)

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\|w\|_2^2$$

$+ \frac{\lambda}{2m} b^2$ (omit)

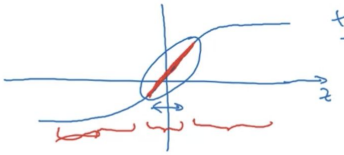
L2 regularization: $\|w\|_2^2 = \sum_{j=1}^{n_x} w_j^2 = w^T w$

L1 regularization: $\frac{\lambda}{2m}\sum_{j=1}^{n_x} |w_j| = \frac{\lambda}{2m}\|w\|_1$

w will be sparse

Andrew Ng

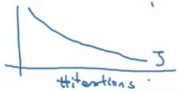
How does regularization prevent overfitting?

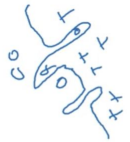


$\lambda \uparrow$ $w^{[l]} \downarrow$ $z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]}$

Every layer $\approx$ linear.

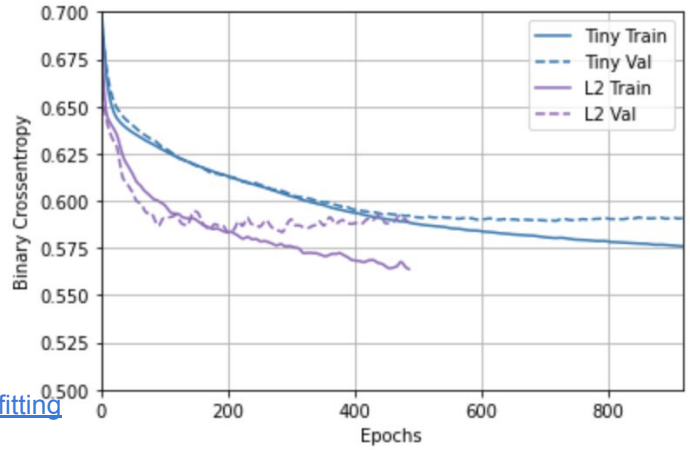
$$J(\dots) = \sum_i \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{1}{2m}\sum_l \|w^{[l]}\|_F^2$$





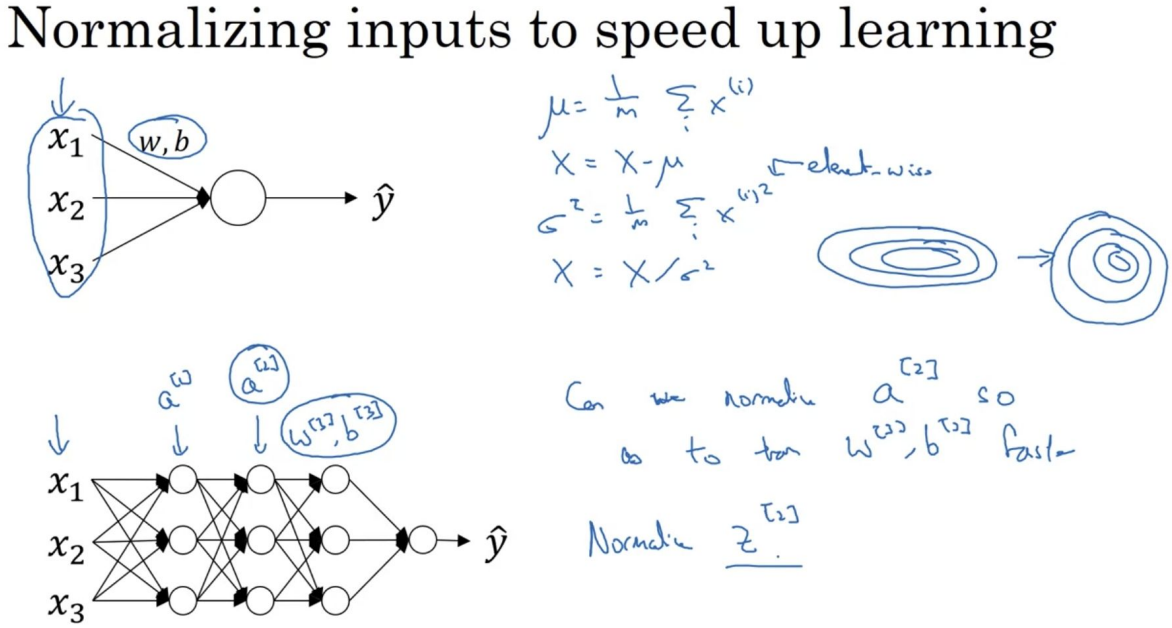
Andrew Ng

```
from tensorflow.keras import regularizers

l2_model = tf.keras.Sequential([
    layers.Dense(512, activation='elu',
                 kernel_regularizer=regularizers.l2(0.001),
                 input_shape=(FEATURES,)),
    layers.Dense(512, activation='elu',
                 kernel_regularizer=regularizers.l2(0.001)),
    layers.Dense(512, activation='elu',
                 kernel_regularizer=regularizers.l2(0.001)),
    layers.Dense(512, activation='elu',
                 kernel_regularizer=regularizers.l2(0.001)),
    layers.Dense(1)
])
```



Source: https://www.coursera.org/learn/deep-neural-network/lecture/Srsrc/regularization
https://www.coursera.org/learn/deep-neural-network/lecture/T6OJj/why-regularization-reduces-overfitting
https://www.tensorflow.org/tutorials/keras/overfit_and_underfit

Extra Content - Batch Normalization



TensorFlow > API > TensorFlow Core v2.1.0 > Python

tf.keras.layers.BatchNormalization

```
tf.keras.layers.BatchNormalization(
```

Normalize the activations of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1.

Batch normalization differs from other layers in several key aspects:

Source:
https://www.coursera.org/learn/deep-neural-network/lecture/4ptp2/normalizing-activations-in-a-network
https://www.tensorflow.org/api_docs/python/tf/keras/layers/BatchNormalization

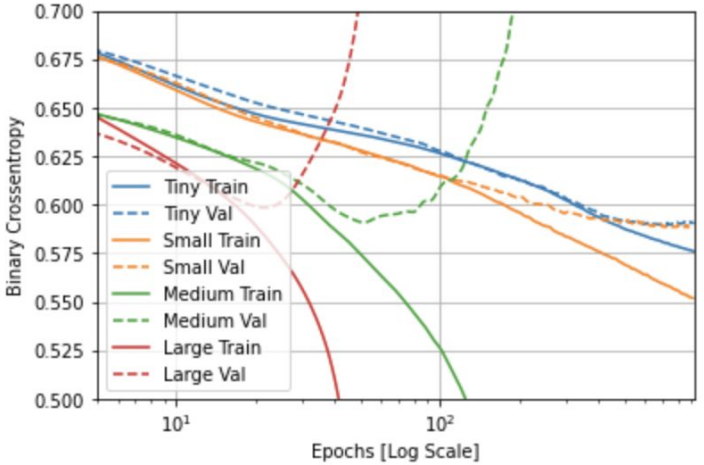
Extra Content - Overfitting in Neural Networks

To recap: here are the most common ways to prevent overfitting in neural networks:

- Get more training data.
- Reduce the capacity of the network.
- Add weight regularization.
- Add dropout.

The Higgs Dataset: 11M samples, 28 features, binary class

Total params:
Tiny: 0.5K
Small: 0.7K
Medium: 10K
Large: 800K

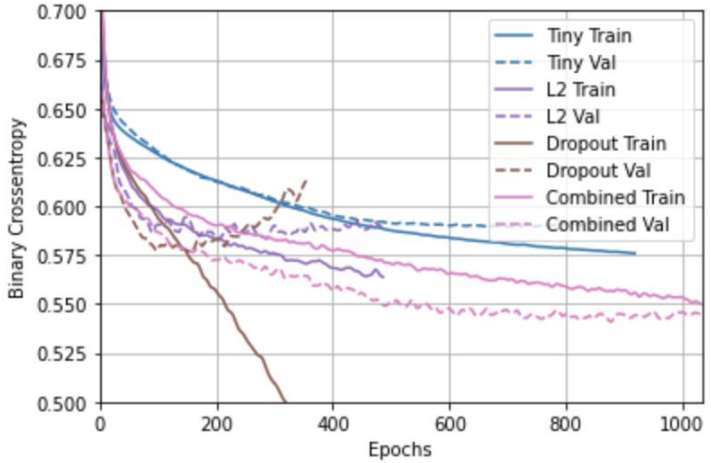


Two important approaches not covered in this guide are:

- data-augmentation
- batch normalization

Remember that each method can help on its own, but often combining them can be even more effective.

- Data Augmentation
- Dropout
- Early Stopping
- Ensembling
- Injecting Noise
- L1 Regularization
- L2 Regularization



Source: https://www.tensorflow.org/tutorials/keras/overfit_and_underfit
https://ml-cheatsheet.readthedocs.io/en/latest/regularization.html

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

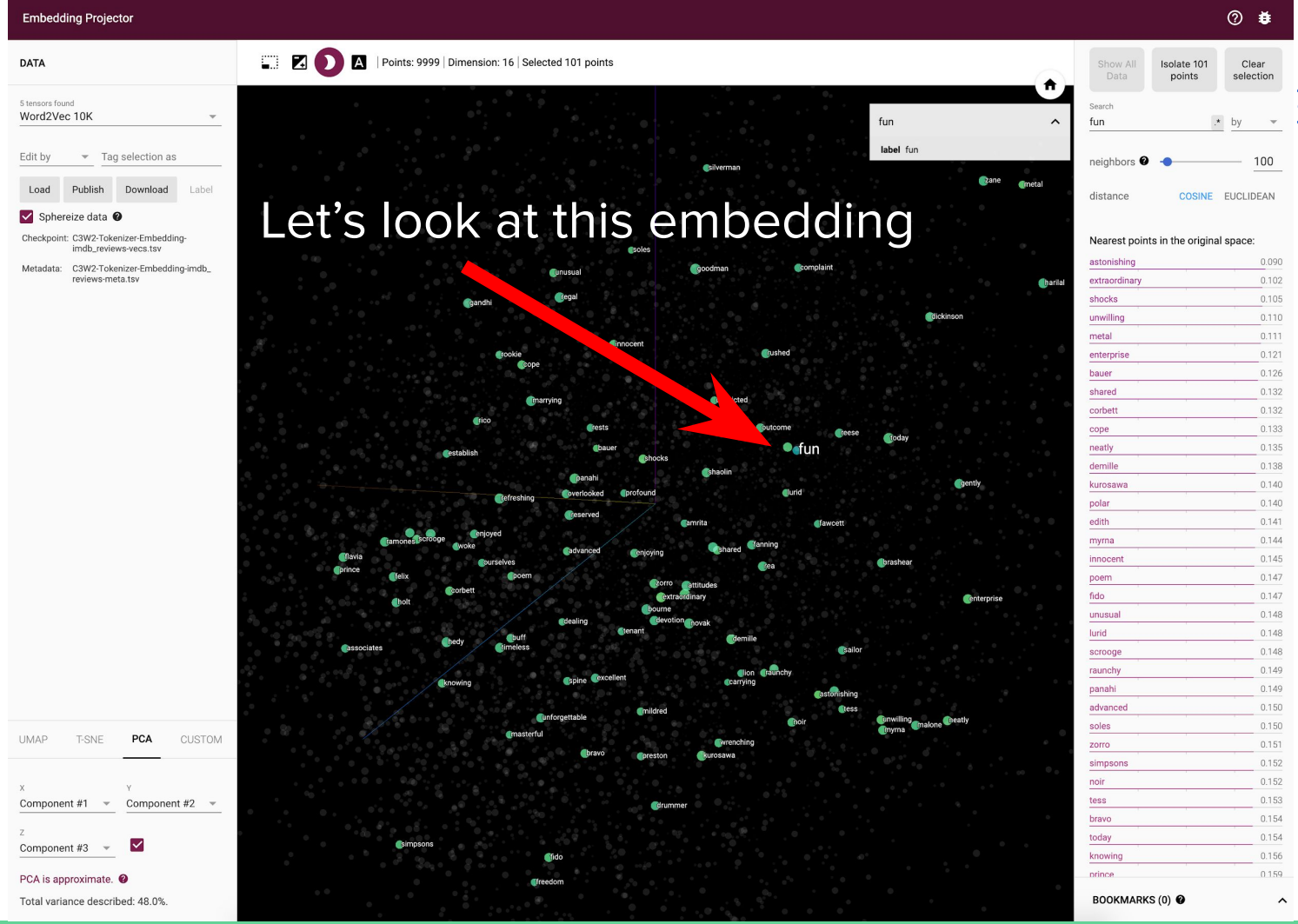
Word Embeddings

```
audio

"nsynth"

image

"abstract_reasoning"
"caltech101"
"cats_vs_dogs"
"celeb_a"
"celeb_a_hq"
"cifar10"
"cifar100"
"cifar10_corrupted"
"coco2014"
"colorectal_histology"
"cycle_gan"
"diabetic_retinopathy..."
"dsprites"
"dtd"
"emnist"
"fashion_mnist"
"horses_or_humans"
"image_label_folder"
"imagenet2012"
"imagenet2012_corrupted"
"kmnist"
"lsun"
"mnist"
"omniglot"
"open_images_v4"
"oxford_iiit_pet"
"quickdraw_bitmap"
"rock_paper_scissors"
"shapes3d"
"smallnorb"
"sun397"
"svhn_cropped"
"tf_flowers"

structured

"higgs"
"iris"
"titanic"

text

"cnn_dailymail"
"glue"
"imdb_reviews"
"lm1b"
"multi_nli"
"squad"
"wikipedia"
"xnli"

translate

"flores"
"para_crawl"
"ted_hrlr_translate"
"ted_multi_translate"
"wmt15_translate"
"wmt16_translate"
"wmt17_translate"
"wmt18_translate"
"wmt19_translate"
```

Source:
https://www.tensorflow.org/datasets/catalog/overview
https://github.com/tensorflow/datasets
https://www.tensorflow.org/datasets/catalog/imdb_reviews

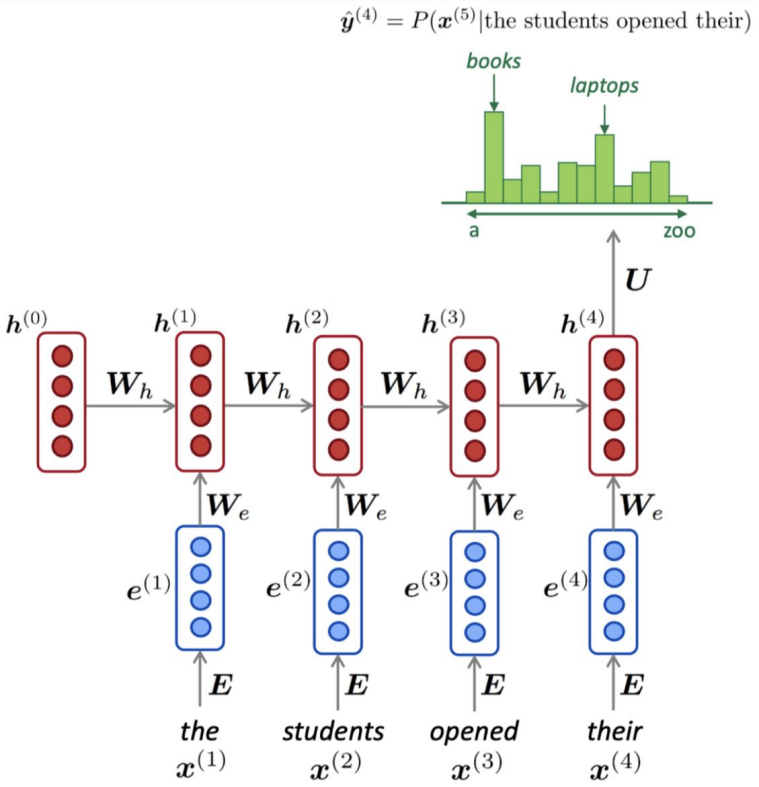
TensorFlow Datasets

- 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

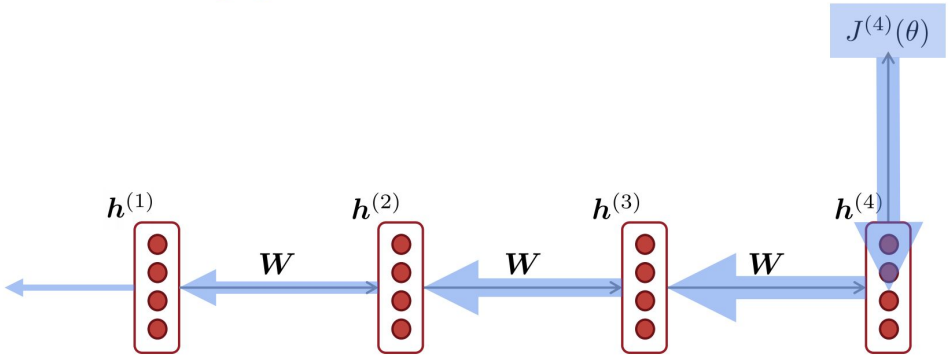


Source: http://projector.tensorflow.org/

Sequence models - new



Vanishing gradient intuition



What happens if these are small?

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

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

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

$$\boldsymbol{f}^{(t)} = \sigma\left(\boldsymbol{W}_f\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_f\boldsymbol{x}^{(t)} + \boldsymbol{b}_f\right)$$

$$\boldsymbol{i}^{(t)} = \sigma\left(\boldsymbol{W}_i\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_i\boldsymbol{x}^{(t)} + \boldsymbol{b}_i\right)$$

$$\boldsymbol{o}^{(t)} = \sigma\left(\boldsymbol{W}_o\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_o\boldsymbol{x}^{(t)} + \boldsymbol{b}_o\right)$$

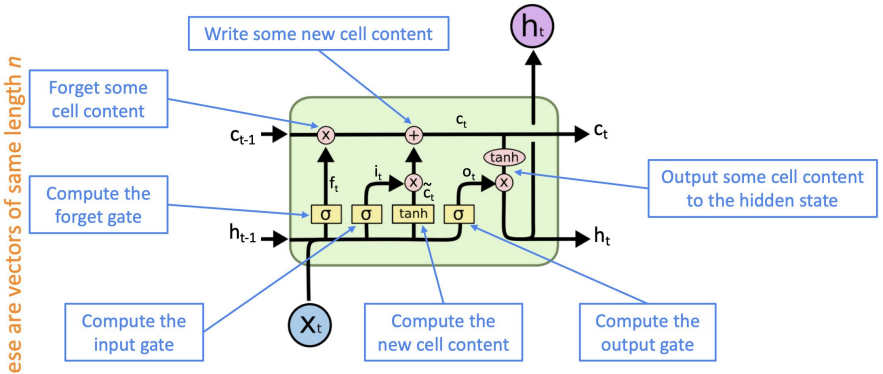
$$\tilde{\boldsymbol{c}}^{(t)} = \tanh\left(\boldsymbol{W}_c\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_c\boldsymbol{x}^{(t)} + \boldsymbol{b}_c\right)$$

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

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

Gates are applied using element-wise product

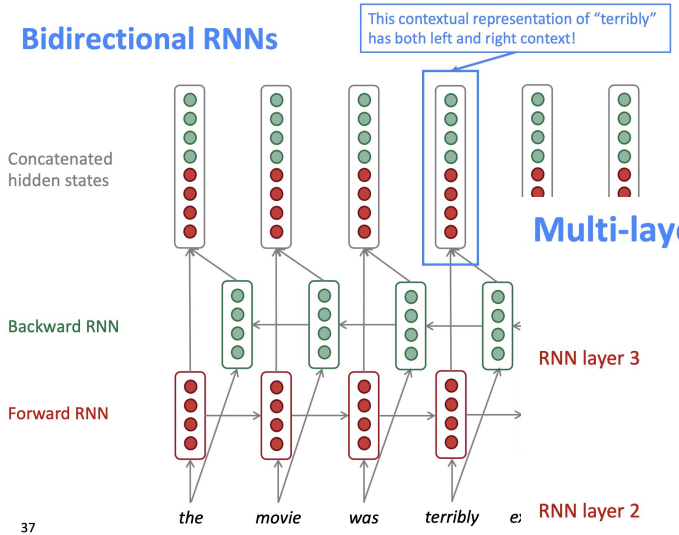
All these are vectors of same length n



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

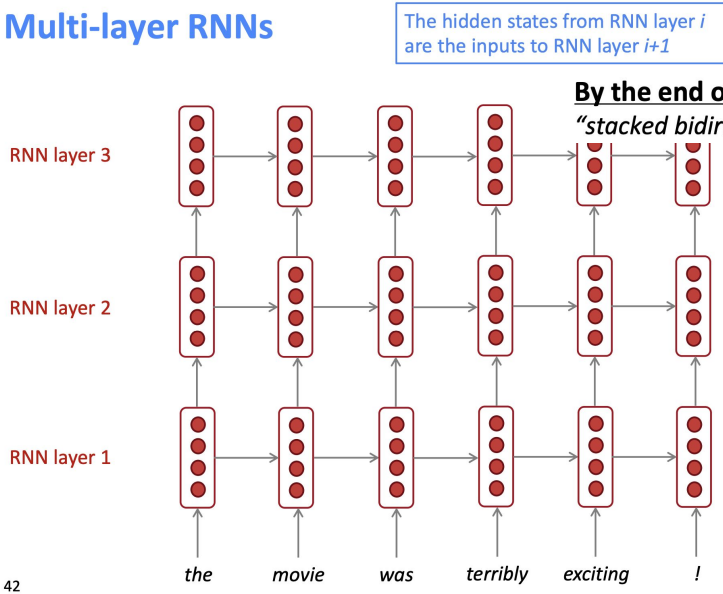
Sequence models - new

Bidirectional RNNs



37

Multi-layer RNNs



42

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"*

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

Sequence models - new

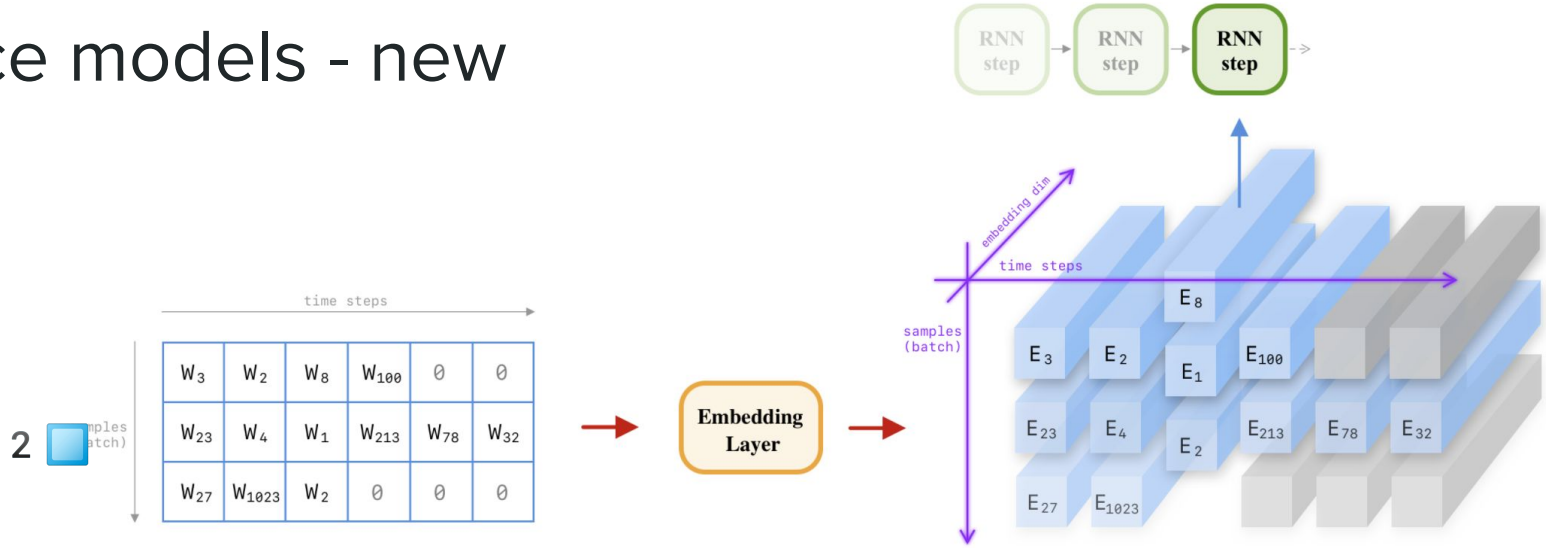


Figure 2. E_i is the embedding vector for word W_i. This tensor has [batch = 3, max_len = 6, features = 300] shape, and is ready to be fed into an RNN network directly.

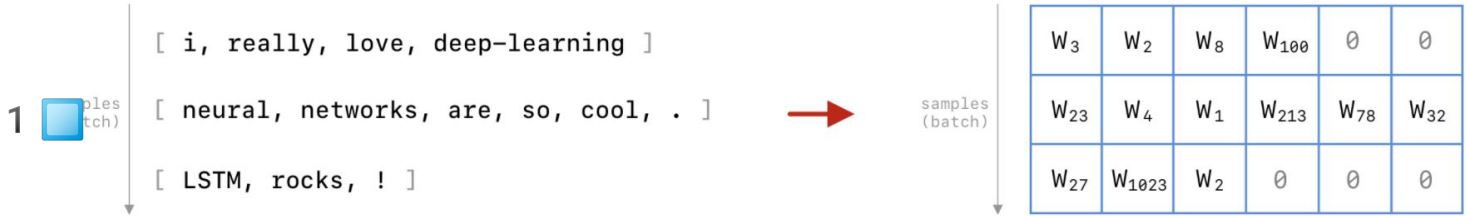
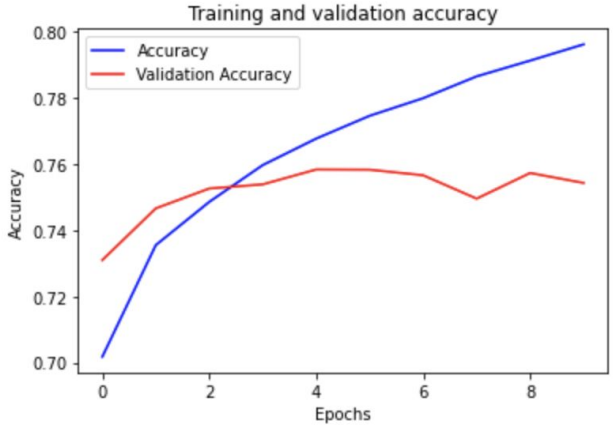


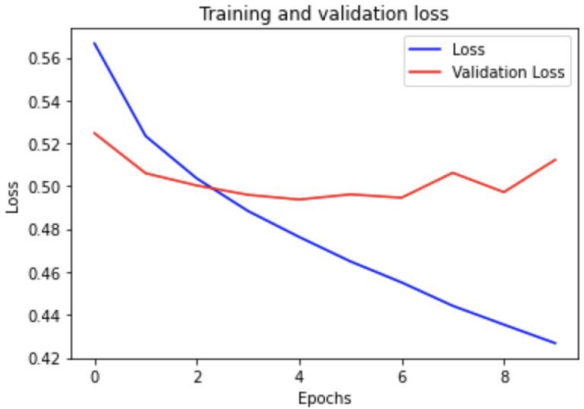
Figure 1. Batch size = 3, and W_i = Word ID assigned by the vocabulary, and 0 is reserved for padding; Here max_len = 6, and sentence 1 and 3 are padded to match up with the longest sentence

Source: https://iust-deep-learning.github.io/972/static_files/assignments/assignment_05_preview.html

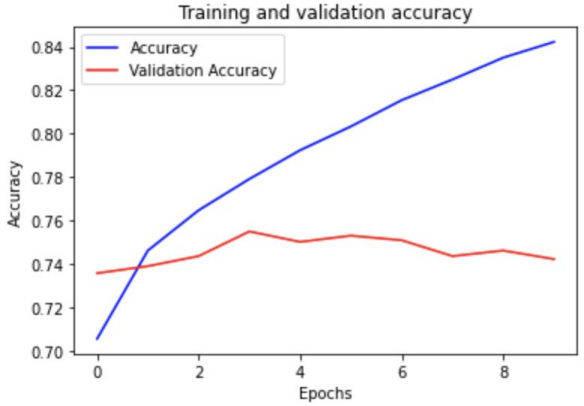
Sequence models

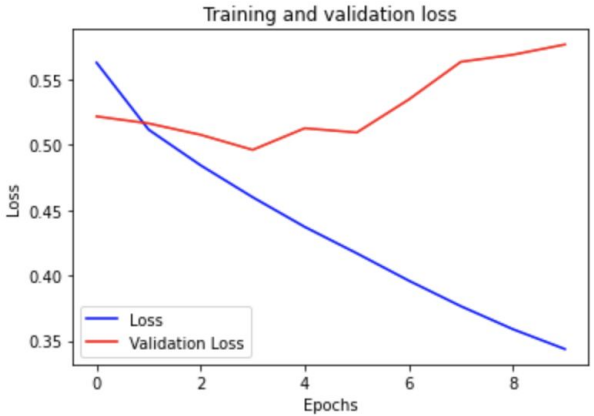
Conv1D, GlobalAveragePooling1D()



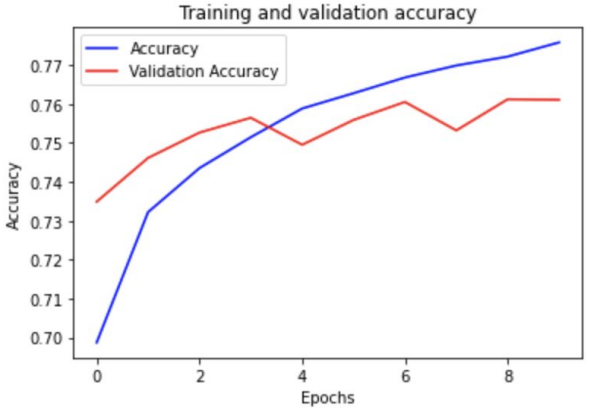


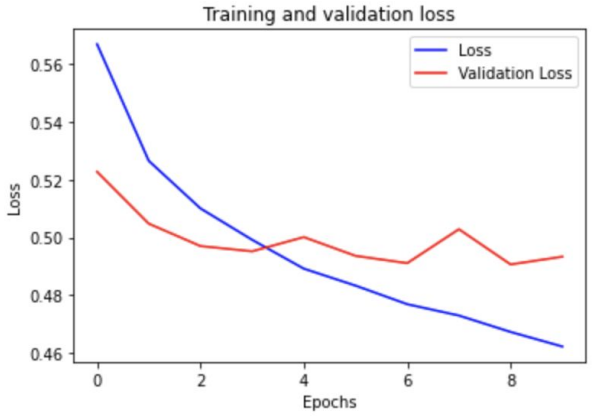
2 x Conv1D, GAP1D(), Dropout





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

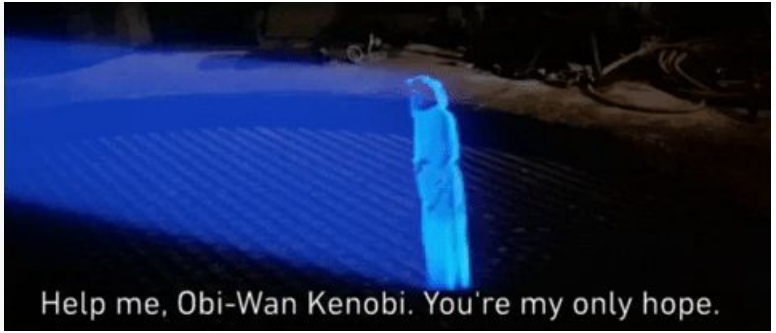


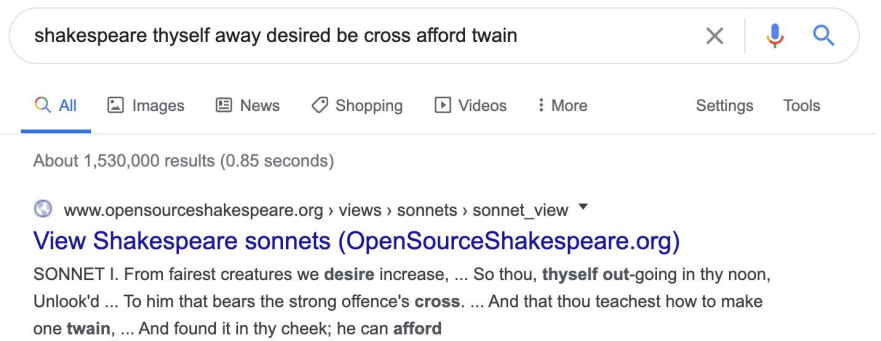


Sequence models and literature

```
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
)
```



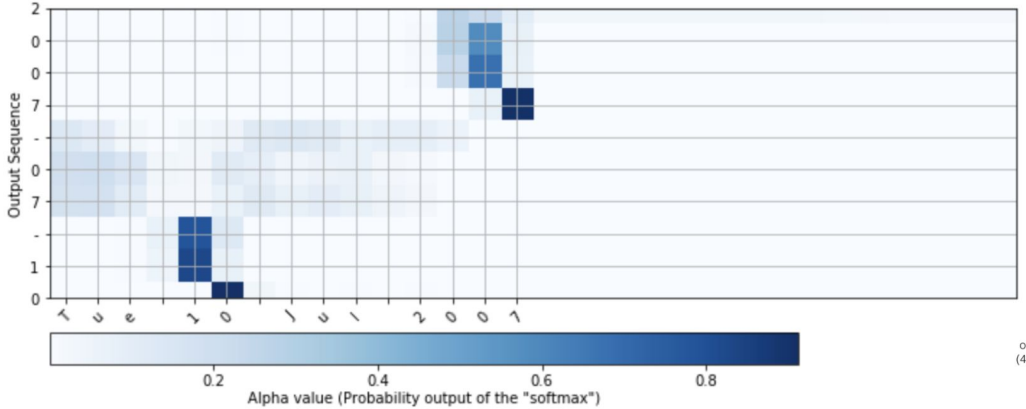


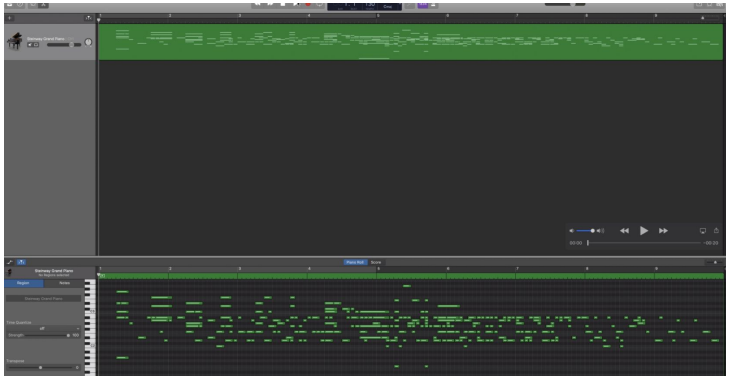
Help me Obi Wan Kenobi, you're my only hope thyself away desired be cross afford twain date of men ' doth groan ' be free free live away away your days here brought of one one more must grow of free ride hits green thee burn pleasure pleasure me of days much brought of pleasure pleasure of pleasure green 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

Extra Content - cool ways to use RNNs, Bias, Attention, Text and Vision

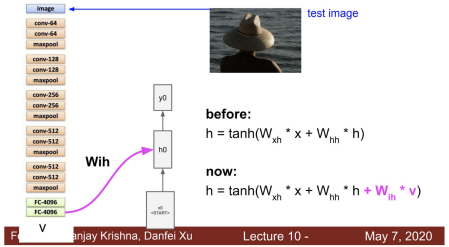
```
text = EXAMPLES[3]
pre_plot(model, human_vocab, inv_machine_vocab, text, num = 7, n_s = 64);
```

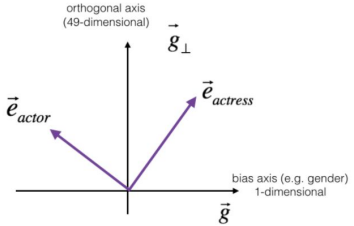
```
Input Tue 10 Jul 2007
Output 2007-07-10
```



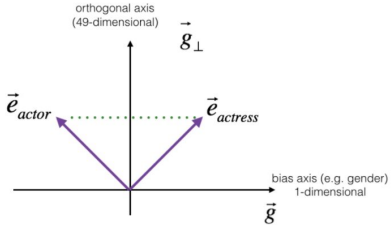


Sequence models and computer vision





before equalizing,
"actress" and "actor" differ in many ways beyond the direction of $\vec{g}$



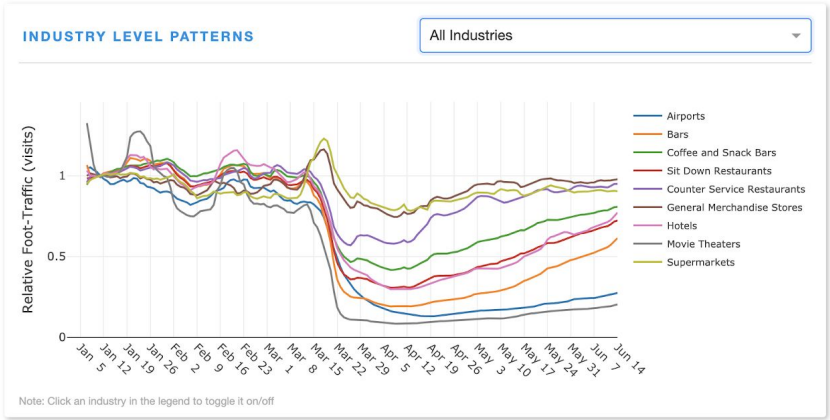
after equalizing,
"actress" and "actor" differ only in the direction of $\vec{g}$, and further are equal in distance from $\vec{g}_\perp$

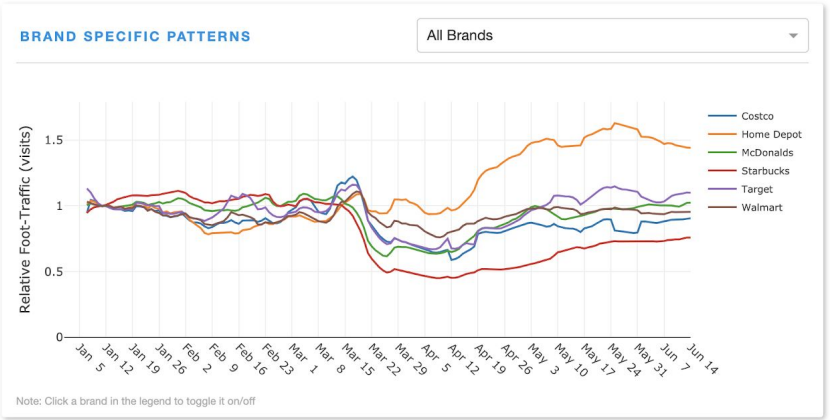
Source: https://github.com/georgezoto/RNN-LSTM-NLP-Sequence-Models

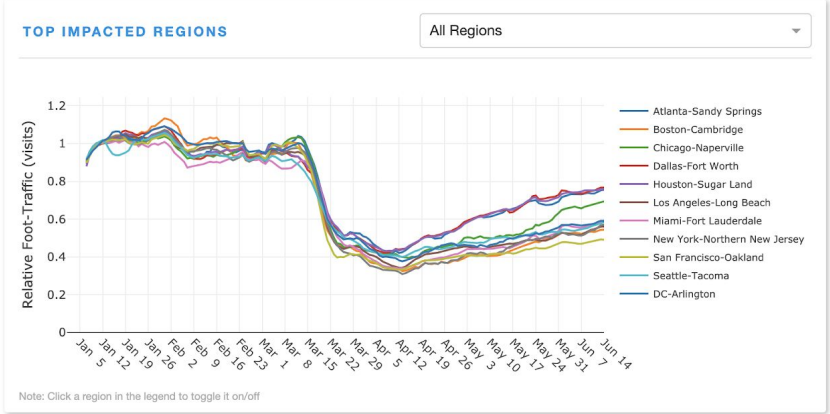
Course 4: Sequences, Time Series and Prediction

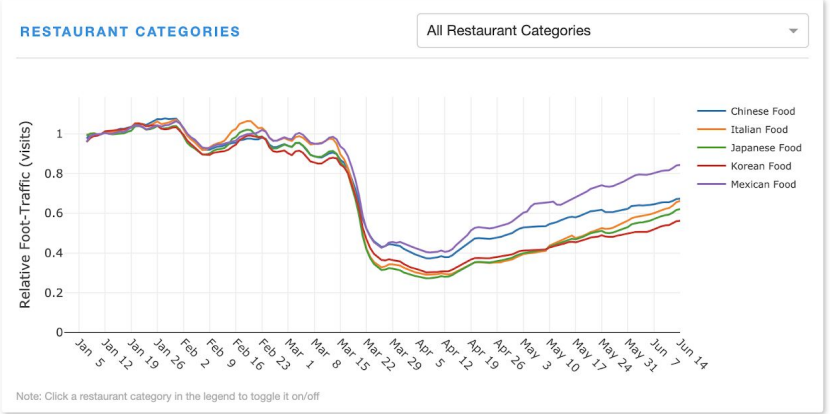
Week 1: Sequences and Prediction
Week 2: Deep Neural Networks for Time Series
Week 3: Recurrent Neural Networks for Time Series
Week 4: Real-world time series data

Sequences and Prediction



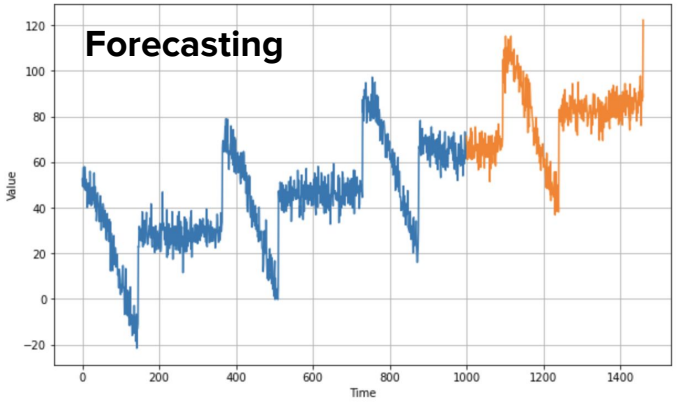






Source: https://www.safegraph.com/dashboard/covid19-commerce-patterns
https://finance.google.com/finance

Sequences and Prediction



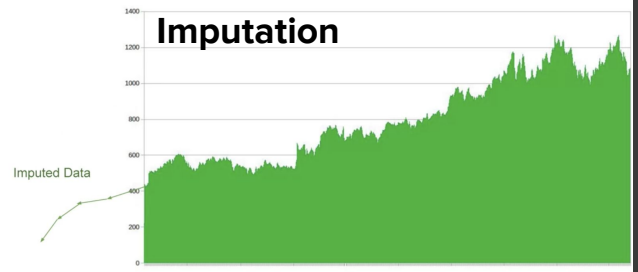
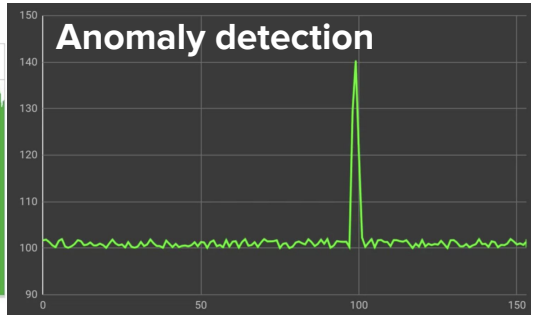


Chart Created by lmoroney@



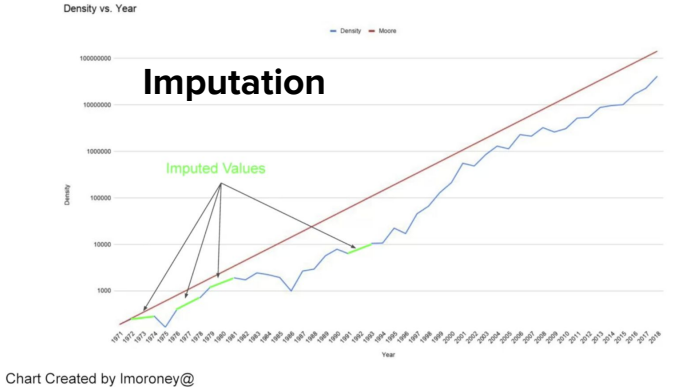
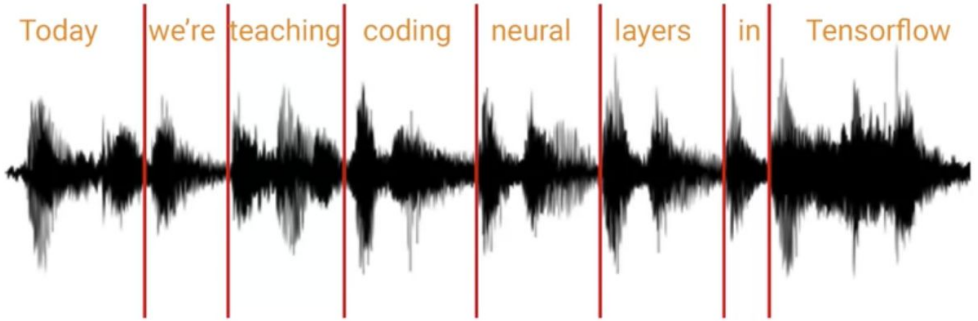


Chart Created by lmoroney@

Patterns / time series analysis

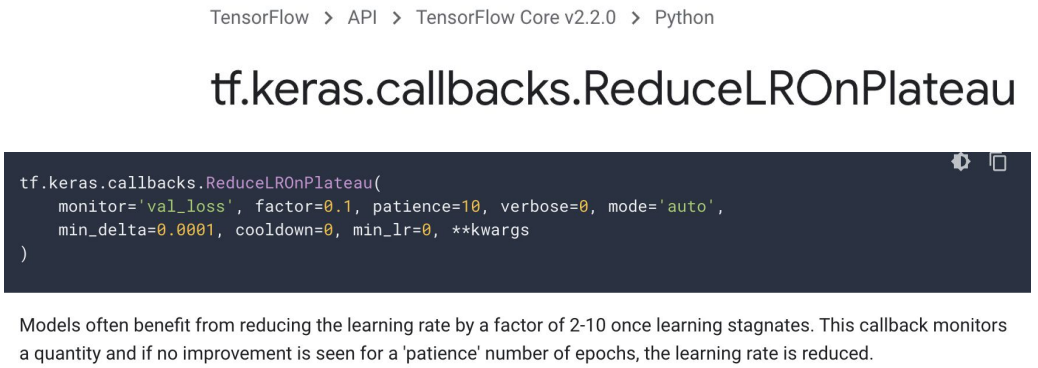


Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

Lawrence's LR optimization?

For consistent results:

```
import random
seed = 51
tf.random.set_seed(seed)
random.seed = seed
```



Set all runs to epochs=500

Restart runtime and run all each test.

Source: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ReduceLROnPlateau

Lawrence's LR optimization results (seed=51)

1. `optimizer=tf.keras.optimizers.SGD(lr=1e-6, momentum=0.9)`

   `mae  4.500836`

2. `optimizer = tf.keras.optimizers.SGD(lr=4e-6, momentum=0.9)`

   `mae  4.8579106`

3. `optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)`

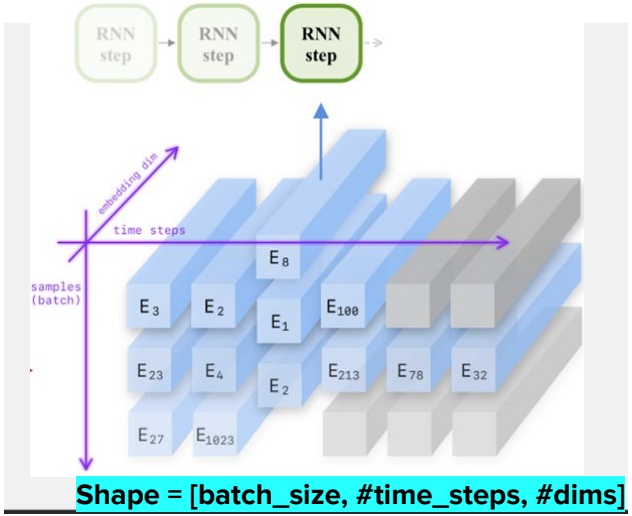
   `mae  5.3804765`

4. `ReduceLROnPlateau(monitor='loss', patience=25, verbose=1)`

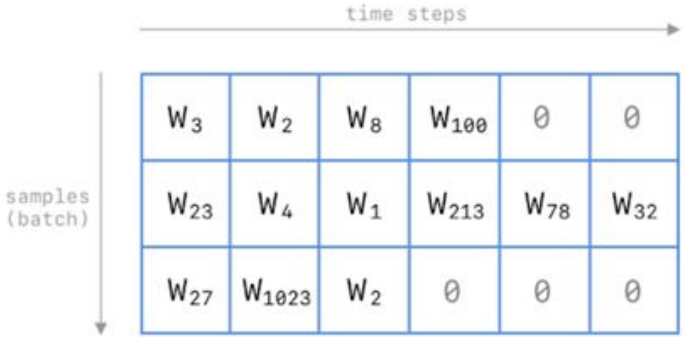
   `optimizer = tf.keras.optimizers.Adam(learning_rate=0.01)`

   `mae  4.490887`

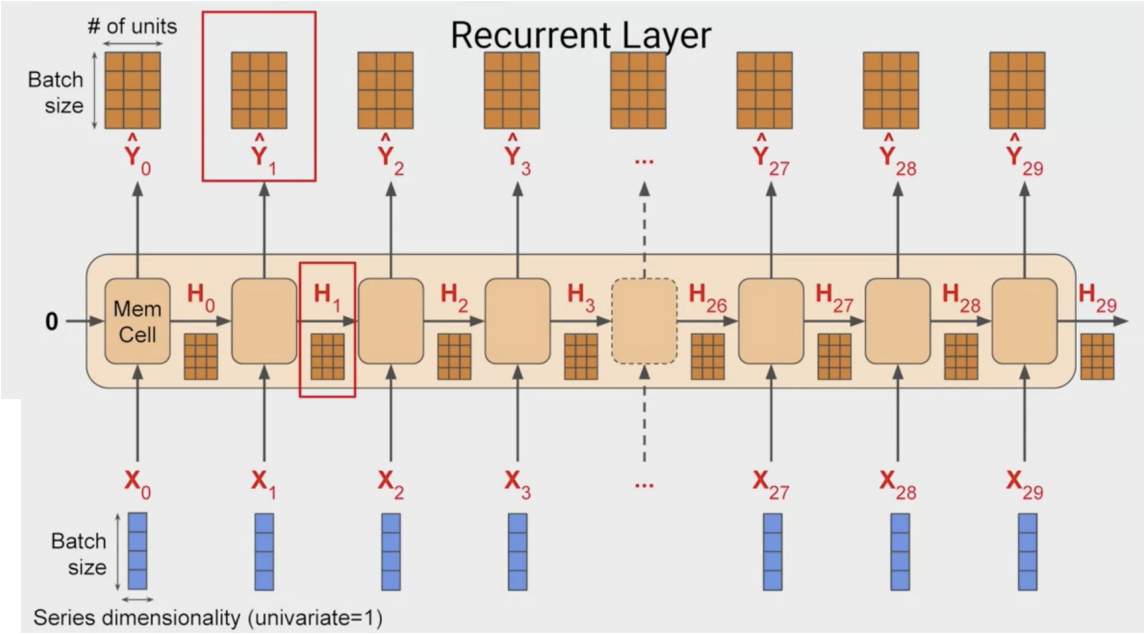
Recurrent Neural Networks for Time Series



Shape = [batch_size, #time_steps, #dims]



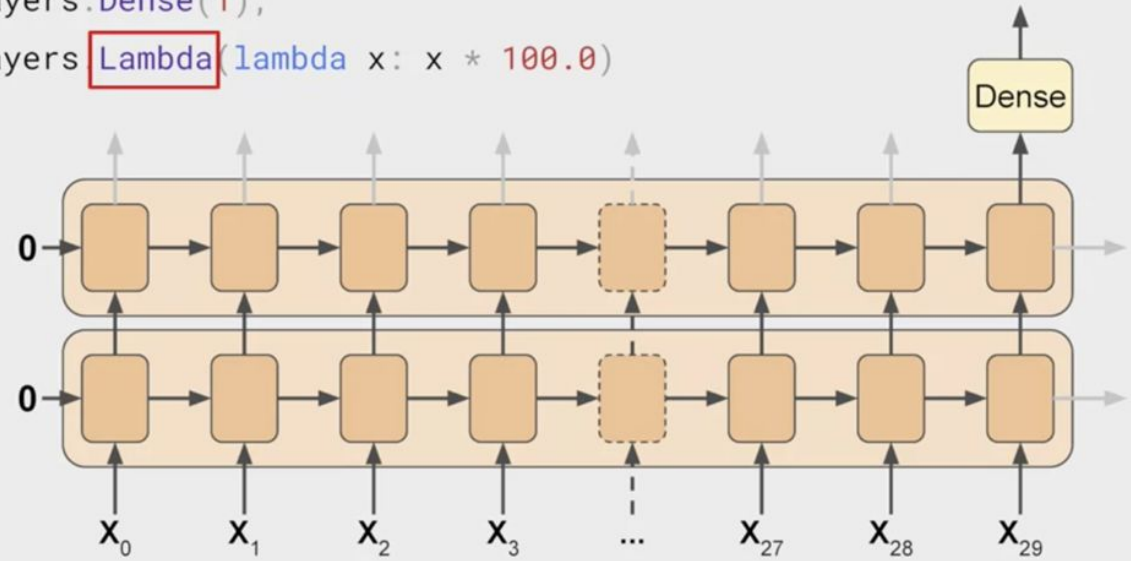
return_sequences=True



Sources: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction
http://karpathy.github.io/2015/05/21/rnn-effectiveness
https://iust-deep-learning.github.io/972/static_files/assignments/assignment_05_preview.html

Recurrent Neural Networks for Time Series

```
model = keras.models.Sequential([
  keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),
                      input_shape=[None]),
  keras.layers.SimpleRNN(20, return_sequences=True),
  keras.layers.SimpleRNN(20),
  keras.layers.Dense(1),
  keras.layers.Lambda(lambda x: x * 100.0)
])
```



Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

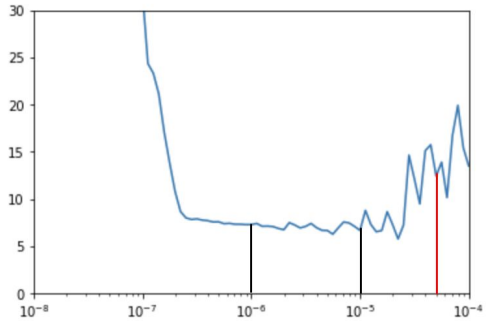
Recurrent Neural Networks for Time Series

```
# Create the series
series = baseline + trend(time, slope) + seasonality(time, period=365, amplit
# Update with noise
series += noise(time, noise_level, seed=42)

split_time = 1000
time_train = time[:split_time]
x_train = series[:split_time]
time_valid = time[split_time:]
x_valid = series[split_time:]

window_size = 20
batch_size = 32
shuffle_buffer_size = 1000

my_epoch = 100
```

```
time = np.arange(4 * 365 + 1,
baseline = 10
series = trend(time, 0.1)
baseline = 10
amplitude = 40
slope = 0.05
noise_level = 5
```

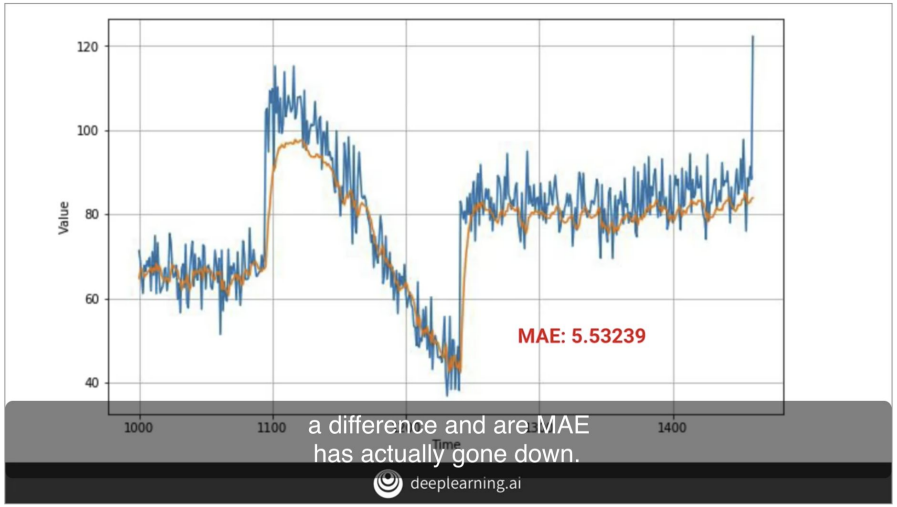
```
plt.semilogx(history.history["lr"], history.history["loss"])
plt.axis([1e-8, 1e-4, 0, 30])
```

```
(1e-08, 0.0001, 0.0, 30.0)
```




```
tf.keras.backer
tf.random.set_s
np.random.seed(

train_set = win

model = tf.kera
  tf.keras.laye

  tf.keras.laye
  tf.keras.laye
  tf.keras.laye
  tf.keras.laye
])

lr_schedule = t
    lambda epoc
optimizer = tf.
model.compile(l
                c
                m
history = model
```

Coding LSTMs



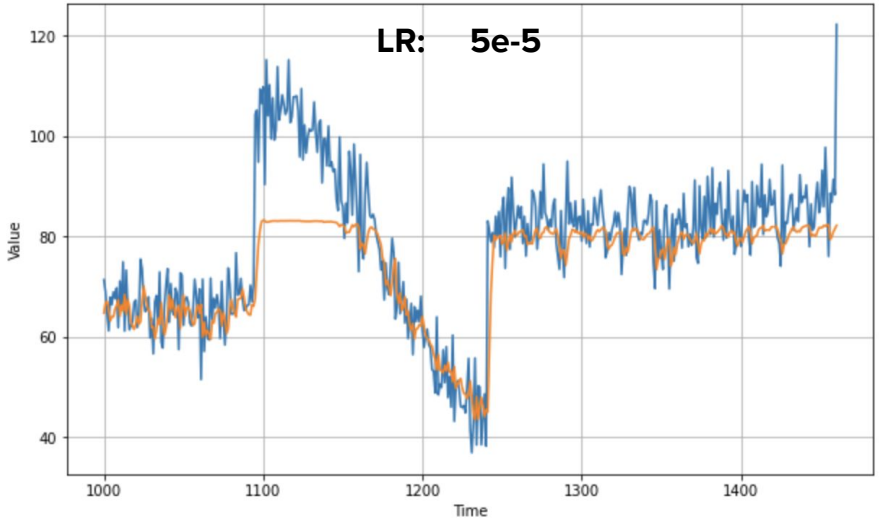

```
optimizer = tf.keras.optimizers.SGD(lr=5e-5, momentum=0.9)
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])
history = model.fit(dataset,epochs=my_epoch*4)
```

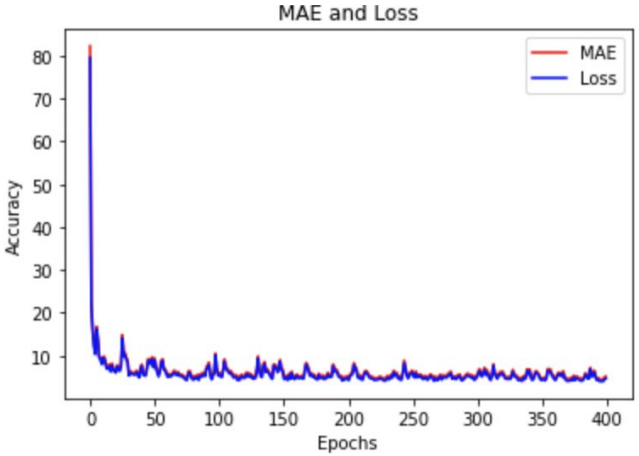
Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

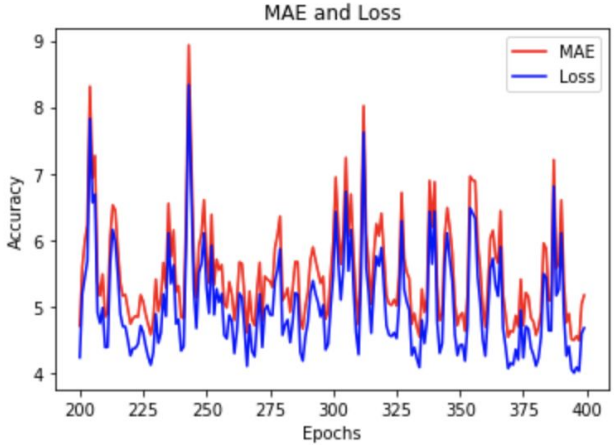
Errata 5*10-6: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction/lecture/5W1Rw/rnn

Errata MAE has gone down: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction/lecture/8u3wq/coding-lstms

Recurrent Neural Networks for Time Series





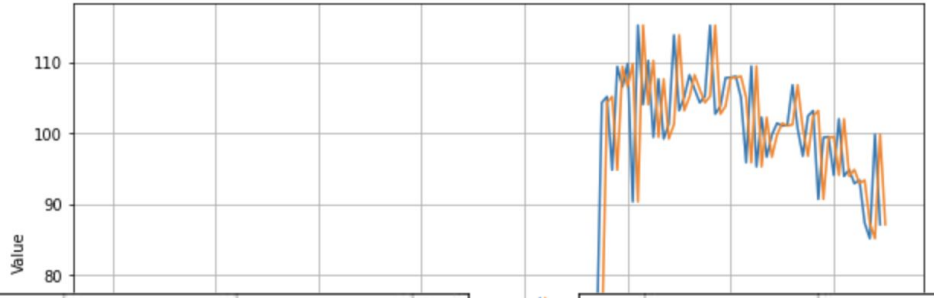



```
tf.keras.metrics.mean_absolute_error(x_valid, results).numpy()
```

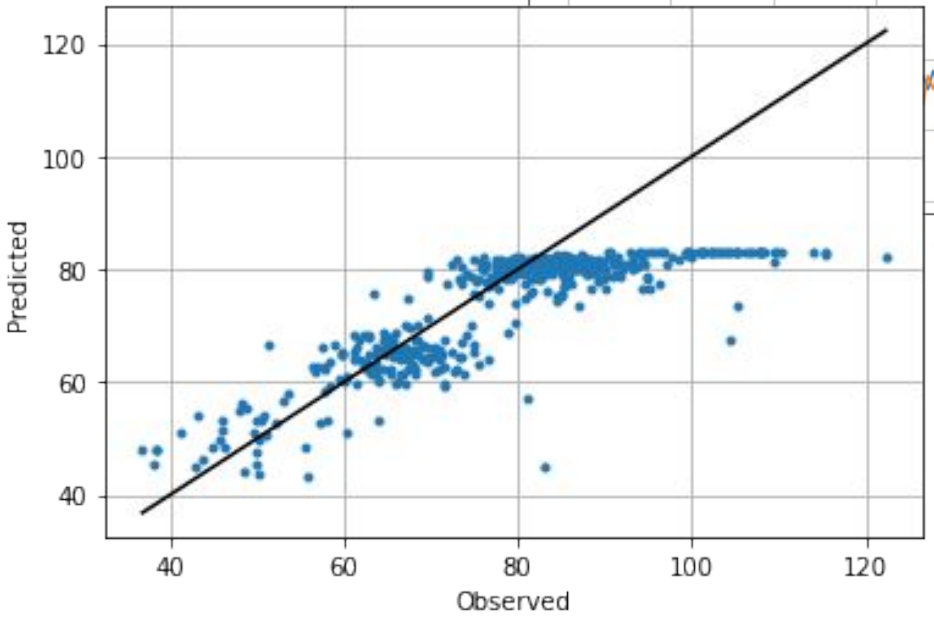
```
6.824173
```

Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

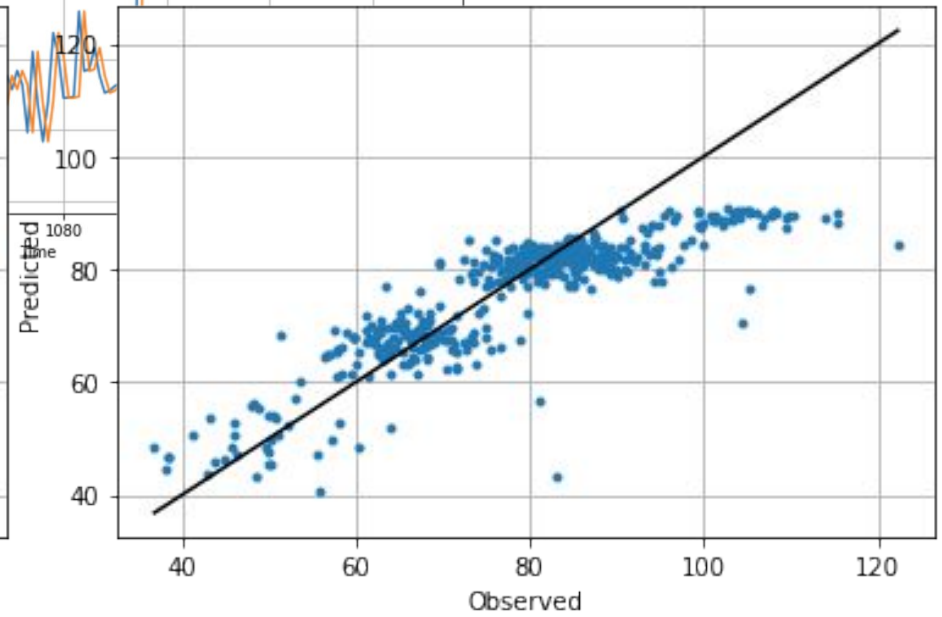
Recurrent Neural Networks for Time Series



LR: 5e-5



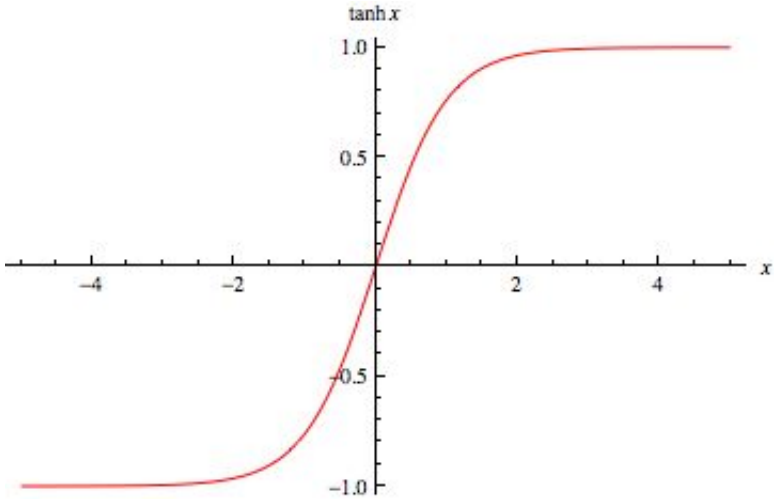
LR: 3e-6



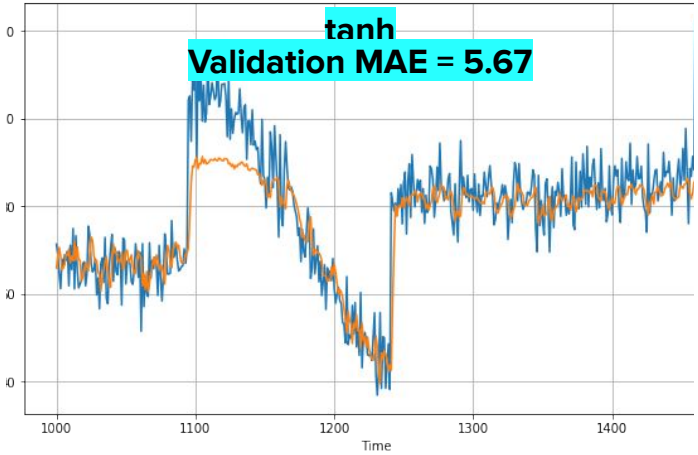
Recurrent Neural Networks for Time Series

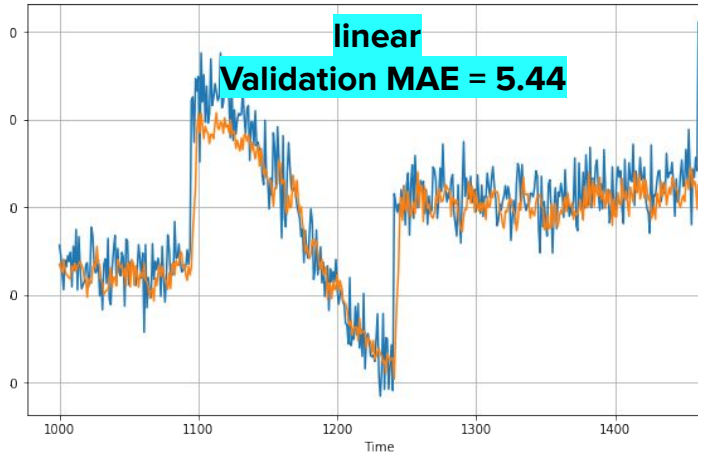
TensorFlow

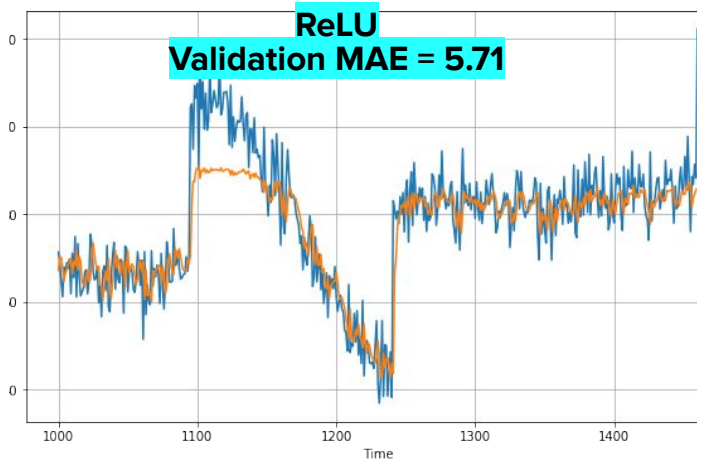
```
tf.keras.layers.SimpleRNN(
    units, activation='tanh', use_bias=True, kernel_initializer='glor
    recurrent_initializer='orthogonal', bias_initializer='zeros',
    kernel_regularizer=None, recurrent_regularizer=None, bias_regular
    activity_regularizer=None, kernel_constraint=None, recurrent_cons
```

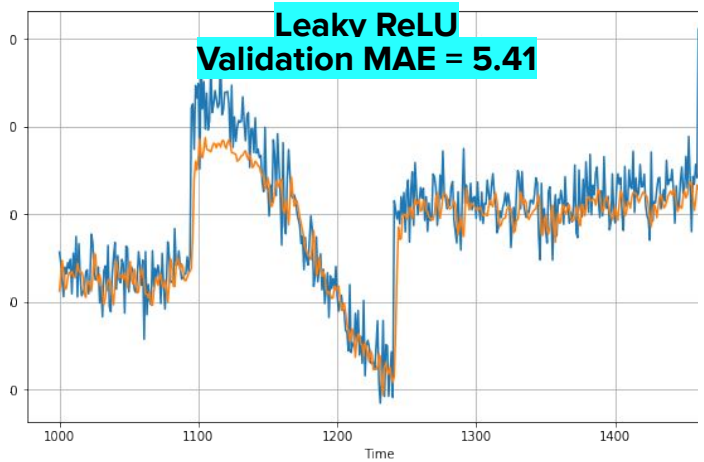


Recurrent Neural Networks for Time Series

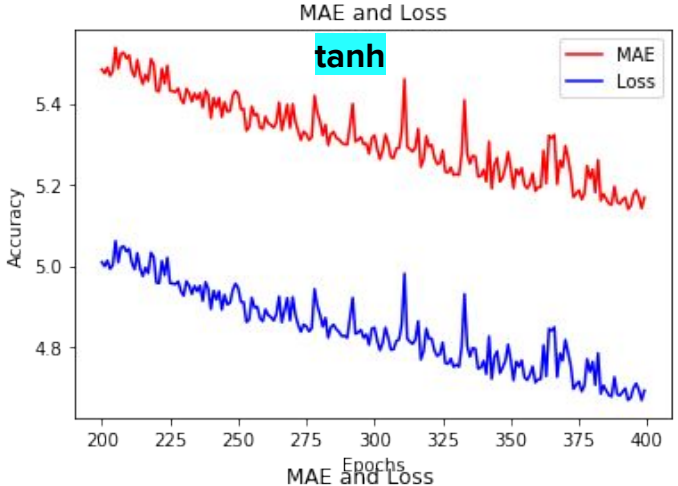


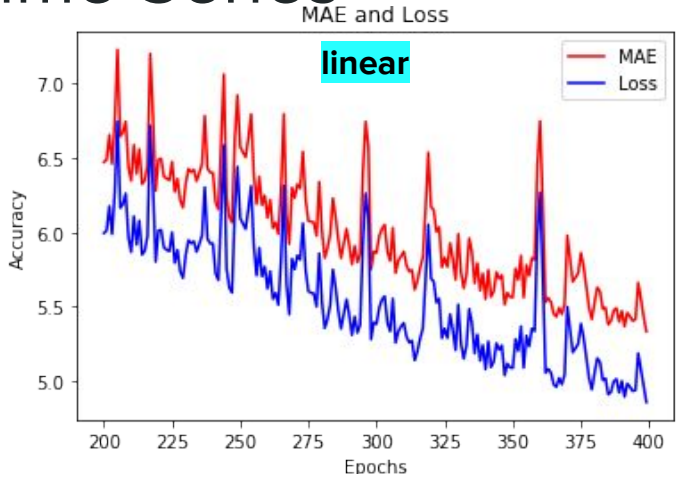


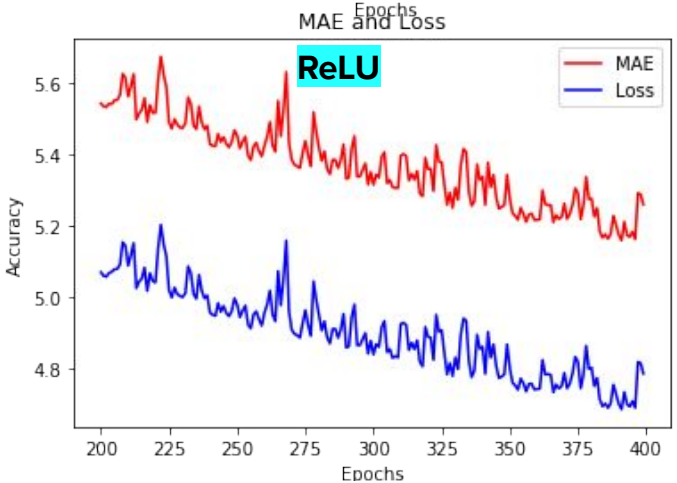


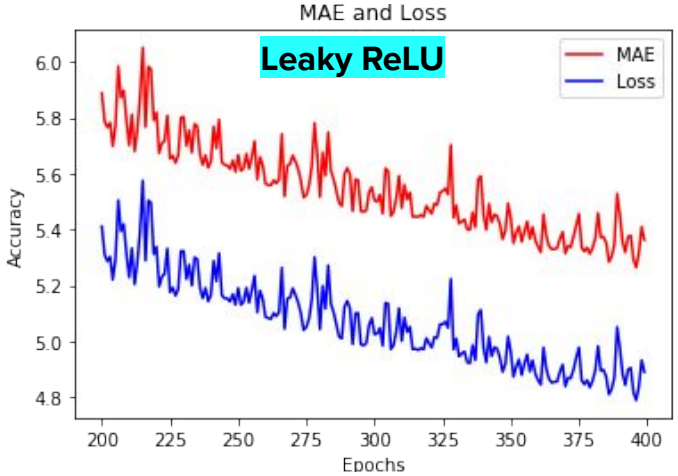


Recurrent Neural Networks for Time Series

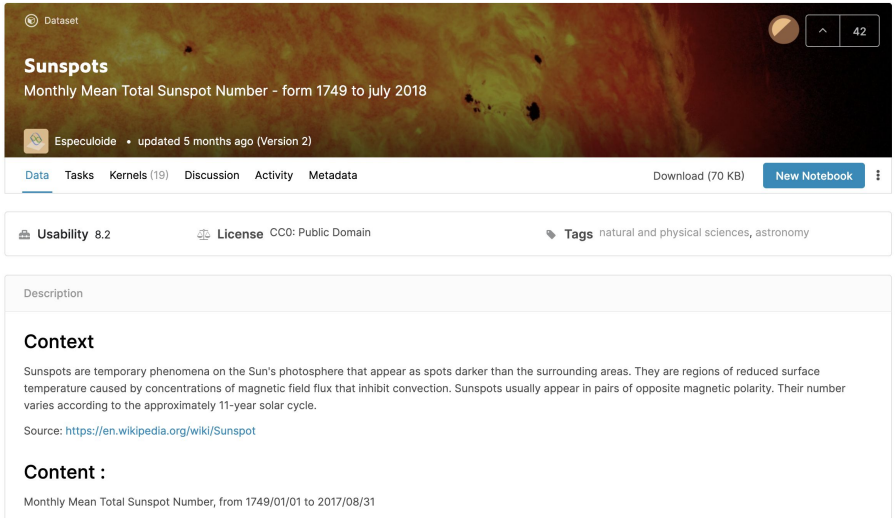








Real-world time series data



Sunspots.csv (69.52 KB)

Detail Compact Column

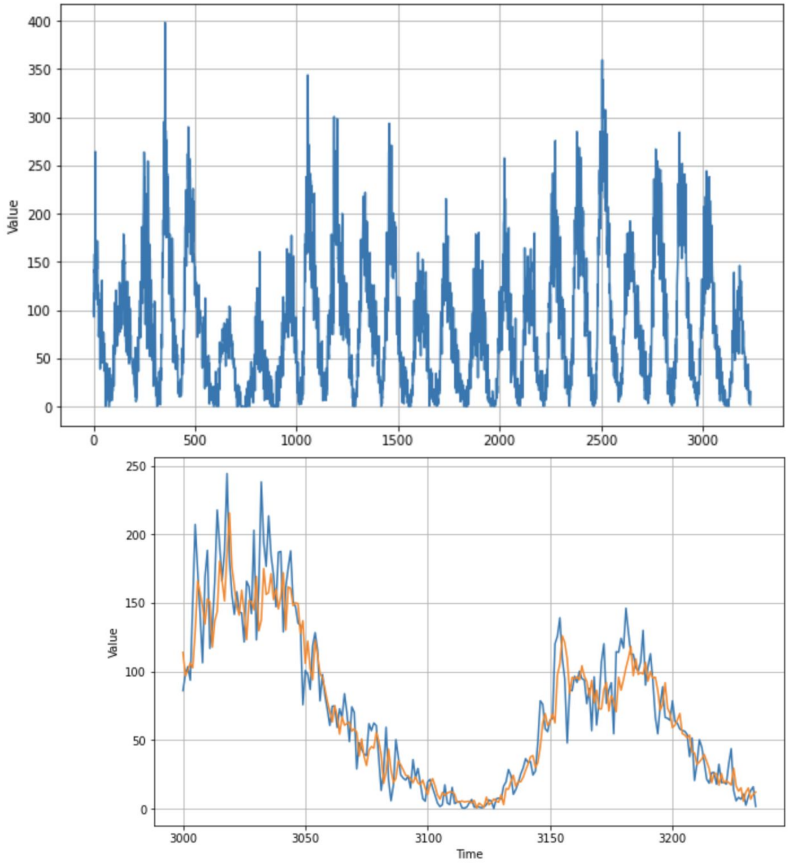
About this file

Sunspots - Monthly Mean Total Sunspot Number

	Date	Monthly Mean T...
0 – 3251	30Jan49 – 30Dec19	0 – 398
0	1749-01-31	96.7
1	1749-02-28	104.3
2	1749-03-31	116.7
3	1749-04-30	92.8
4	1749-05-31	141.7
5	1749-06-30	139.2
6	1749-07-31	158.0
7	1749-08-31	110.5
8	1749-09-30	126.5
9	1749-10-31	125.8
10	1749-11-30	264.3

Source: https://www.kaggle.com/robervalt/sunspots

Real-world time series data




```
dataset = windowed_dataset_dnn(x_train, window_size, batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(20, input_shape=[window_size], activation="relu"),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

model.compile(loss="mse", optimizer=tf.keras.optimizers.SGD(lr=1e-7, momentum=0.9))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 20)                620
_________________________________________________________________
dense_4 (Dense)              (None, 10)                210
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 11
=================================================================
Total params: 841
Trainable params: 841
Non-trainable params: 0
_________________________________________________________________
```

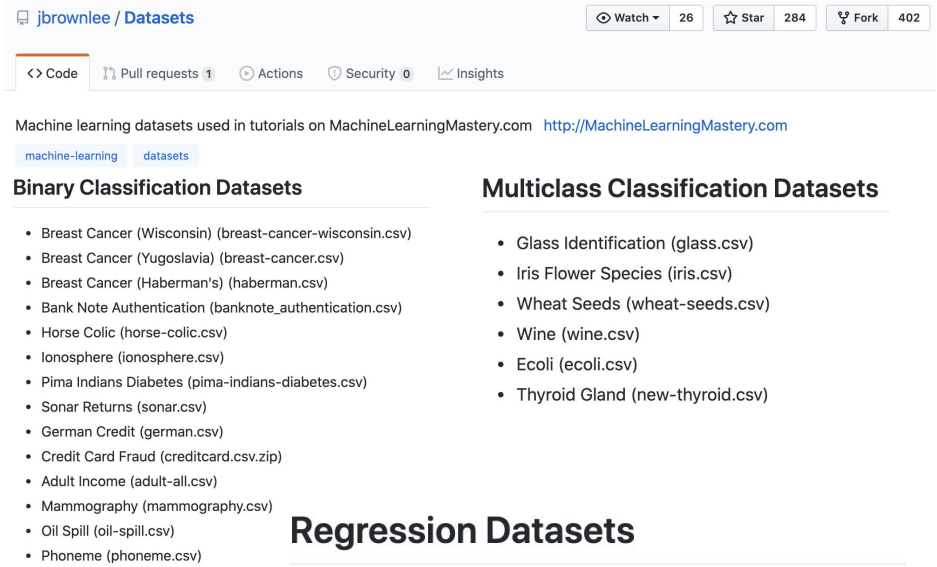
```
model.fit(dataset,epochs=100,verbose=1)
```

```
tf.keras.metrics.mean_absolute_error(x_valid, results).numpy()
```

```
14.931627
```

Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

Real-world time series data



jbrownlee / Datasets

Watch 26 | Star 284 | Fork 402

Code | Pull requests 1 | Actions | Security 0 | Insights

Machine learning datasets used in tutorials on MachineLearningMastery.com http://MachineLearningMastery.com

machine-learning datasets

Binary Classification Datasets

- Breast Cancer (Wisconsin) (breast-cancer-wisconsin.csv)
- Breast Cancer (Yugoslavia) (breast-cancer.csv)
- Breast Cancer (Haberman's) (haberman.csv)
- Bank Note Authentication (banknote_authentication.csv)
- Horse Colic (horse-colic.csv)
- Ionosphere (ionosphere.csv)
- Pima Indians Diabetes (pima-indians-diabetes.csv)
- Sonar Returns (sonar.csv)
- German Credit (german.csv)
- Credit Card Fraud (creditcard.csv.zip)
- Adult Income (adult-all.csv)
- Mammography (mammography.csv)
- Oil Spill (oil-spill.csv)
- Phoneme (phoneme.csv)

Multiclass Classification Datasets

- Glass Identification (glass.csv)
- Iris Flower Species (iris.csv)
- Wheat Seeds (wheat-seeds.csv)
- Wine (wine.csv)
- Ecoli (ecoli.csv)
- Thyroid Gland (new-thyroid.csv)

Regression Datasets

- Boston Housing (housing.csv)
- Auto Insurance Total Claims (auto-insurance.csv)
- Auto Imports Prices (auto_imports.csv)
- Abalone Age (abalone.csv)
- Wine Quality Red (winequality-red.csv)
- Wine Quality White (winequality-white.csv)

Univariate Time Series Datasets

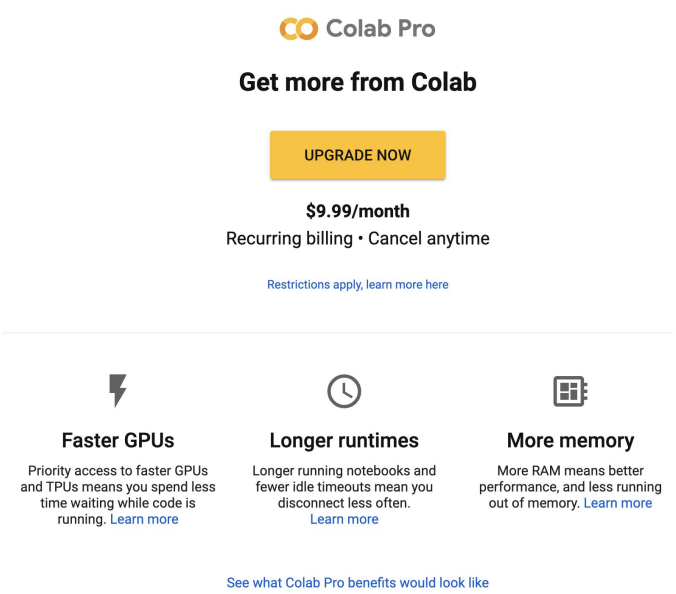
- Daily Minimum Temperatures in Melbourne (daily-min-temperatures.csv)
- Daily Maximum Temperatures in Melbourne (daily-max-temperatures.csv)
- Daily Female Births in California (daily-total-female-births.csv)
- Monthly International Airline Passengers (monthly-airline-passengers.csv)
- Monthly Armed Robberies in Boston (monthly-robberies.csv)
- Monthly Sunspots (monthly-sunspots.csv)
- Monthly Champagne Sales (monthly_champagne_sales.csv)
- Monthly Shampoo Sales (monthly-shampoo-sales.csv)
- Monthly Car Sales (monthly-car-sales.csv)
- Monthly Mean Temperatures in Nottingham Castle (monthly-mean-temp.csv)
- Monthly Specialty Writing Paper Sales (monthly-writing-paper-sales.csv)
- Yearly Water Usage in Baltimore (yearly-water-usage.csv)

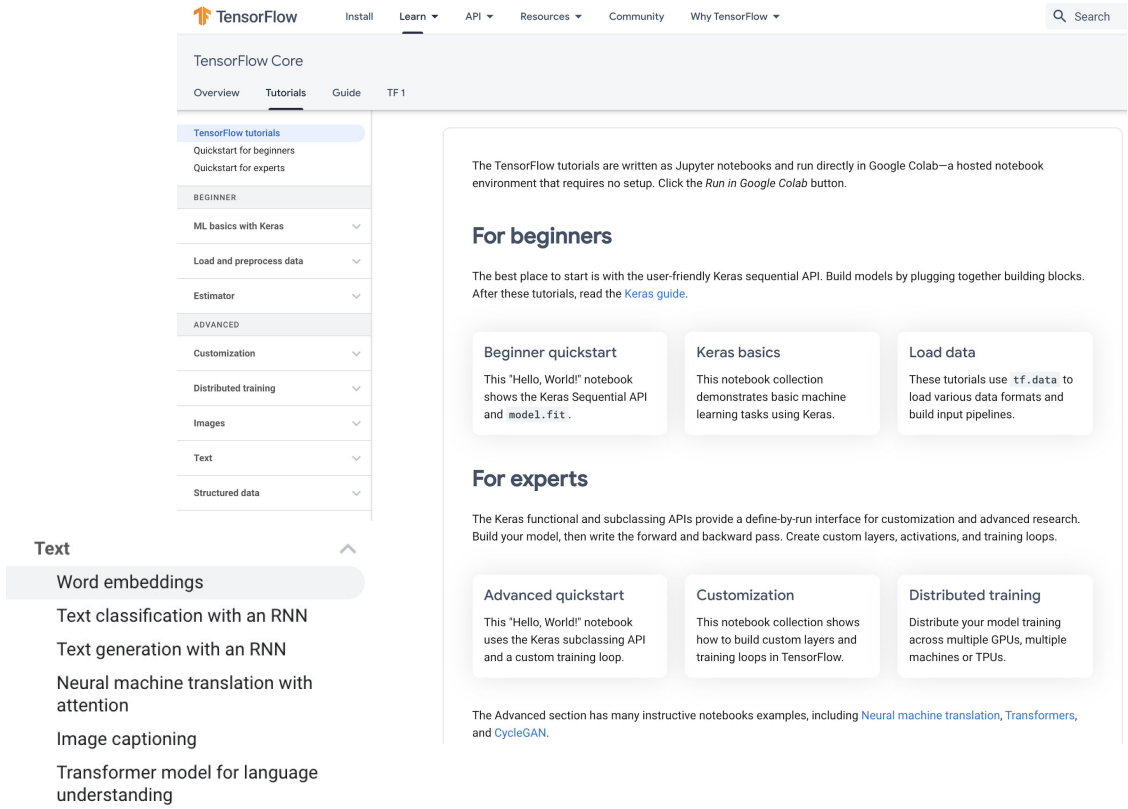
Multivariate Time Series Datasets

- Hourly Pollution Levels in Beijing
- Minutely Individual Household Electric Power Consumption
- Human Activity Recognition Using Smartphones
- Indoor Movement Prediction
- Yearly Longley Economic Employment

Source: https://github.com/jbrownlee/Datasets

Check out these resources





Source:
https://colab.research.google.com/signup
https://www.tensorflow.org/tutorials/

Check out these AI for Healthcare resources

deeplearning.ai **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 1

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

COURSE 2

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

COURSE 3

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

SYLLABUS

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

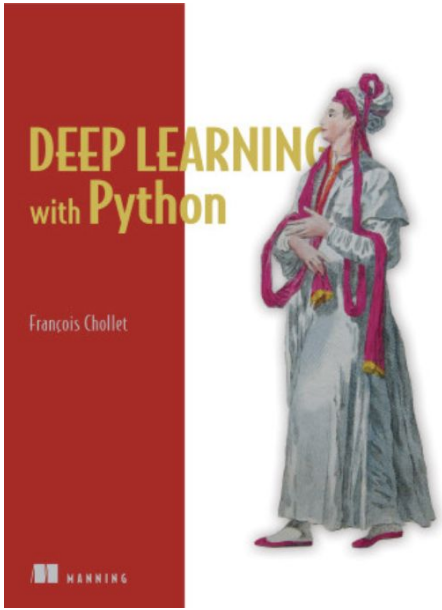
TensorFlow Core

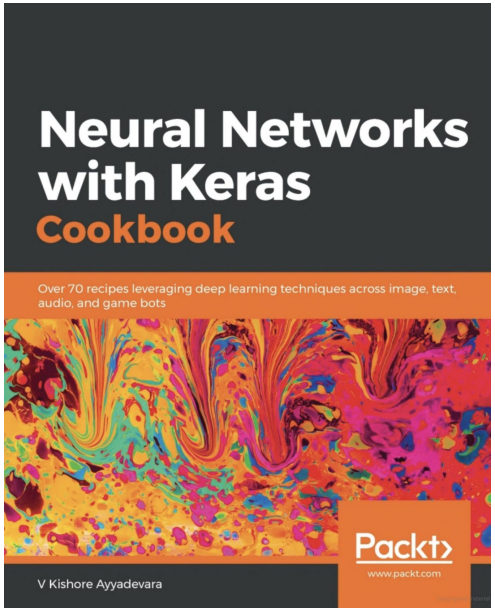
Introducing the TensorFlow Developer Certificate!

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=5guLDwAAQBAJ&printsec=frontcover&source=gbs_ge_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 Time Series adventure 😀

FRI, JUN 19, 7:30 PM EDT

TensorFlow in Practice - C4 Week 1,2 - Forecasting and DNN for Time...

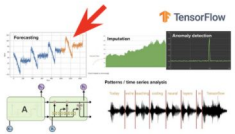
Online event

Join us for our 9th adventure in Deep Learning! Just bring your curiosity and get ready to meet our growing community 😀 We are taking Course 4 of TensorFlow in Practice Specialization available at:...

39 attendees

FRI, JUN 26, 7:30 PM EDT

TensorFlow in Practice - C4 Week 3,4 - Forecasting & RNN, Conv1D f...

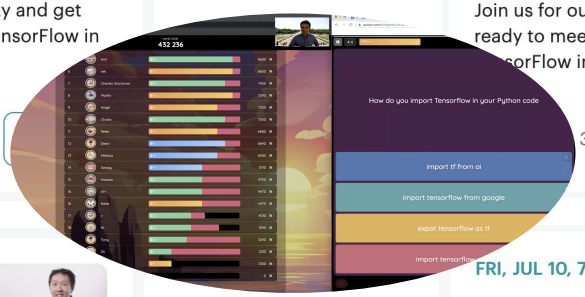


Online event

Join us for our 10th adventure in Deep Learning! Just bring your curiosity and get ready to meet our growing community 😀 We are taking our last Course 4 of TensorFlow in Practice Specialization available at:...

31 attendees

Attend



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...

71 attendees

2

Attend

FRI, JUL 10, 7:30 PM EDT

How to prepare for and pass the TensorFlow Developer Certificate 🎉

Online event

Join us for our 11th adventure in Deep Learning! Just bring your curiosity and get ready to meet our growing community 😀 Join Zoom Meeting: https://us02web.zoom.us/j/84402592502?...

21 attendees

Attend

Source: https://www.meetup.com/Deep-Learning-Adventures

Questions

Discussion

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:

·m : number of examples in the dataset

·n_x : input size

·n_y : output size (or number of classes)

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

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

·L : number of layers in the network.

Objects:

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

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

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

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

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

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

·$\hat{y} \in \mathbb{R}^{n_y}$ 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

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

· 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:

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

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

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

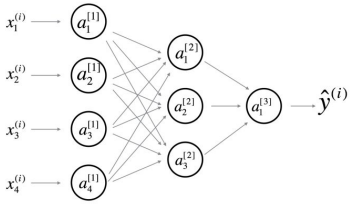


Figure 1: Comprehensive Network: representation commonly used for Neural Networks. For better aesthetic, we omitted the details on the parameters ($w_{ij}^{[l]}$ and $b_i^{[l]}$ etc...) that should appear on the edges

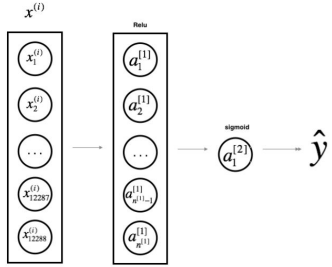


Figure 2: Simplified Network: a simpler representation of a two layer neural network, both are equivalent.