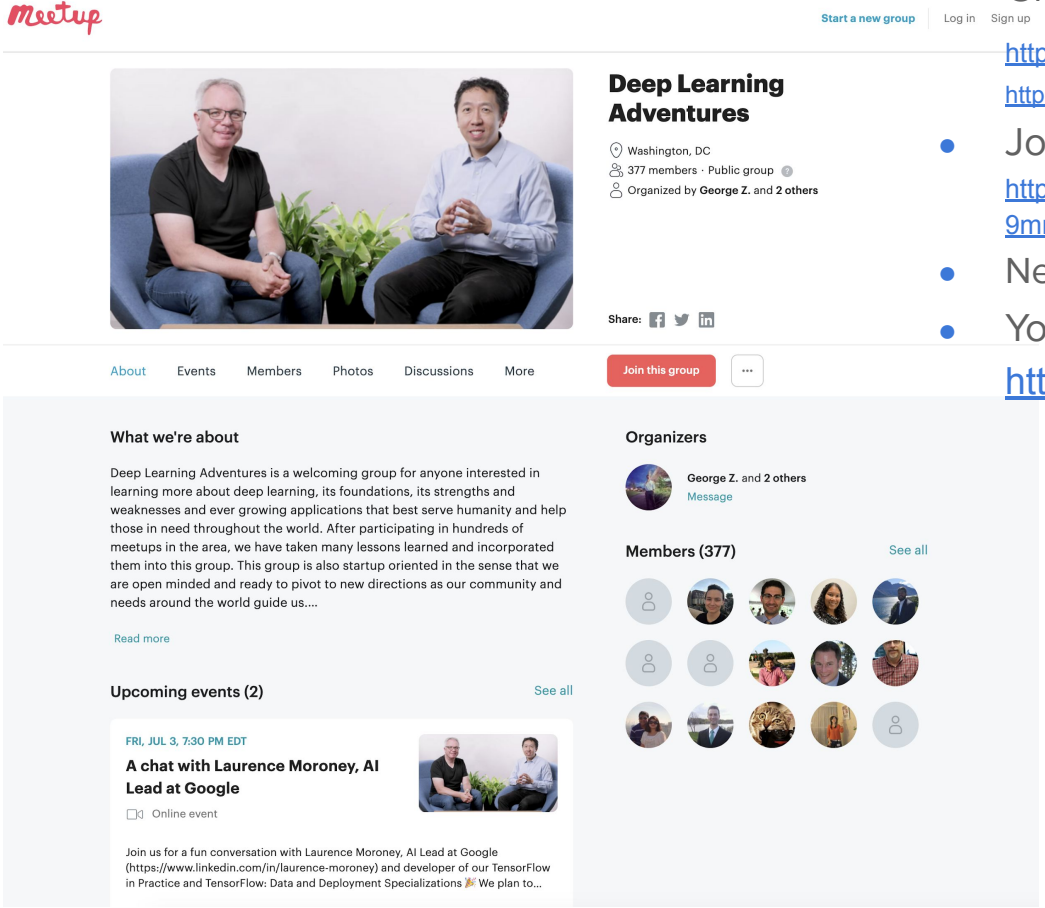
Deep Learning Adventures TensorFlow In Practice - Presentation 5

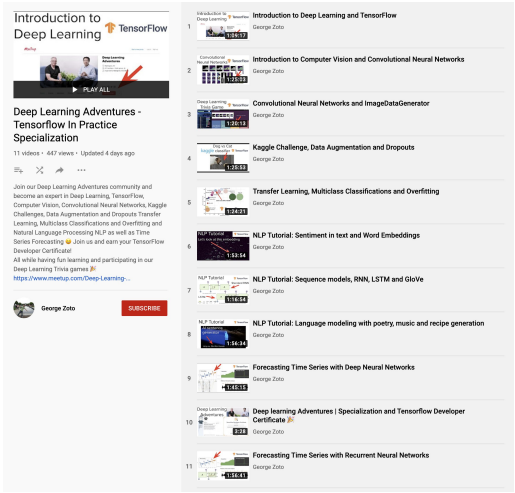
A quick overview of Coursera's Tensorflow in Practice specialization course

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

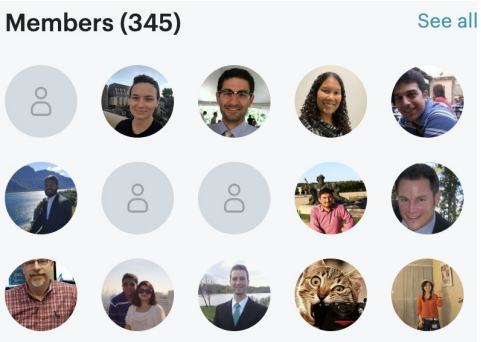
In the beginning...

- 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





Get to know our community



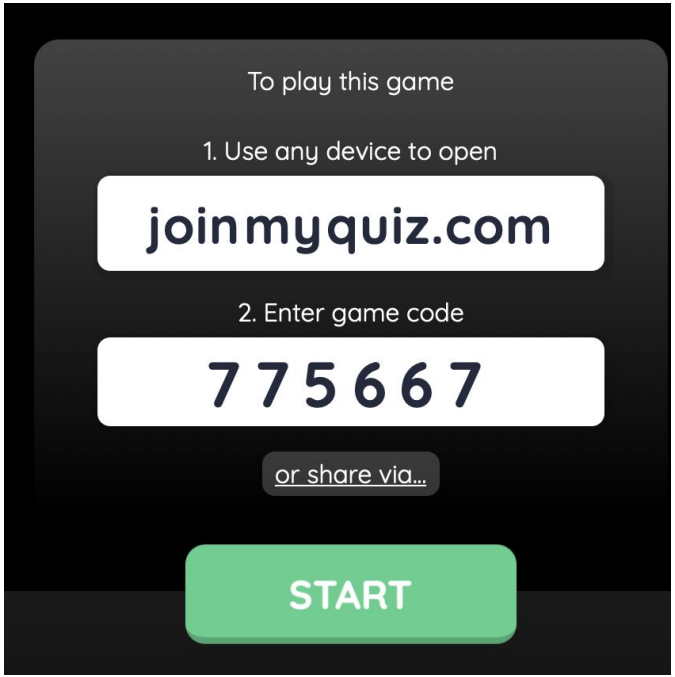
Hello, my name is ____ and I am a new/existing/regular member of this meetup.

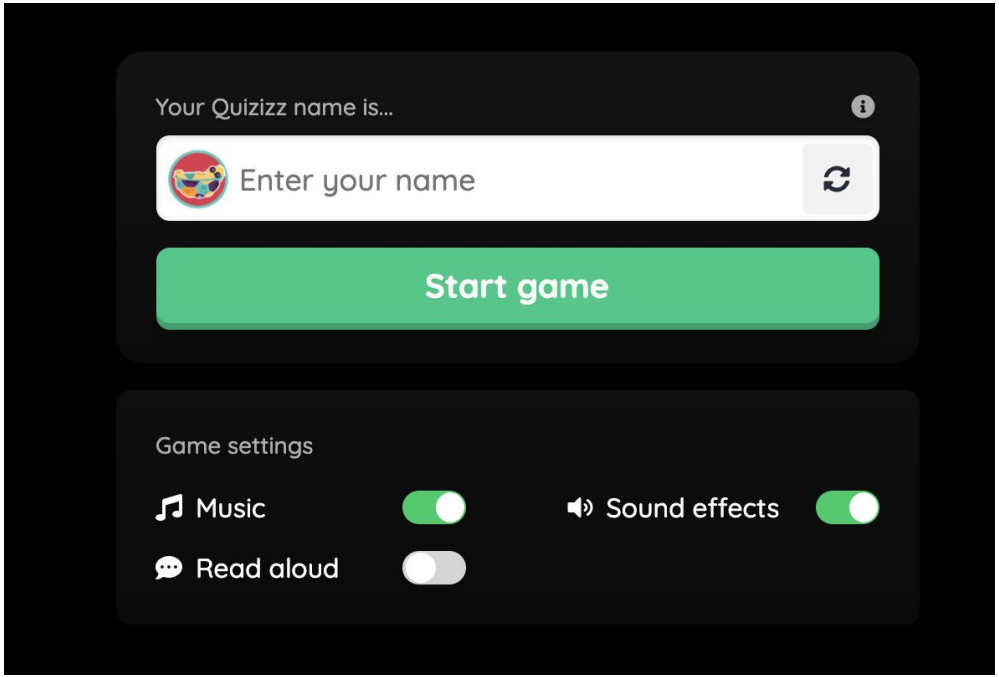
I enjoy ____ and I am interested in learning more about ____

Training set 😀

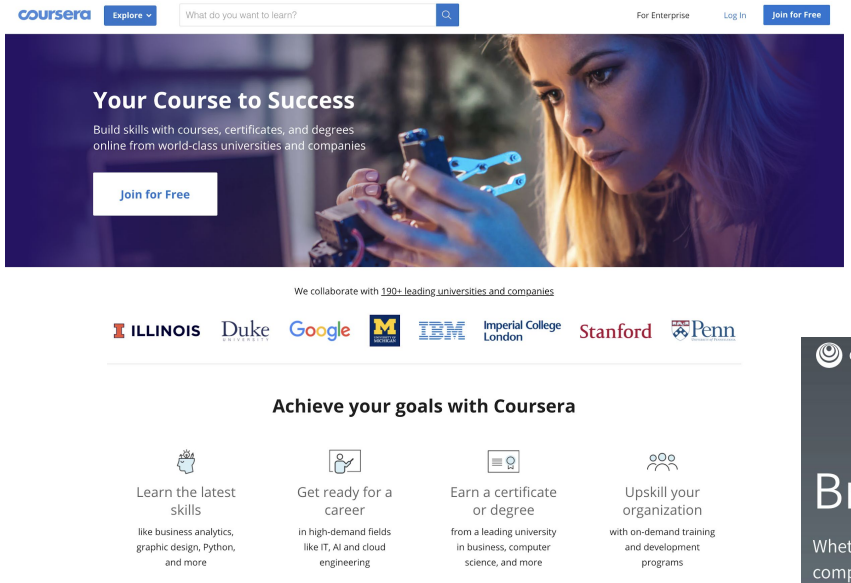
Hello, my name is **George** and I am a **regular** member of this meetup. I enjoy **applying deep learning to solve interesting problems** and I am interested in learning more about **time series and forecasting**.

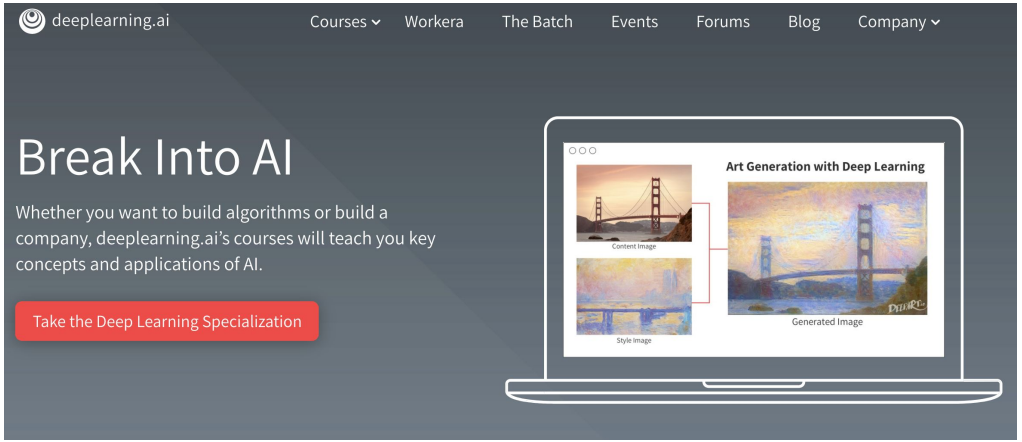
Not a typical Meetup... Get ready for a fun game on 7/3 😀🎉





Attribution to Coursera and deeplearning.ai





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

Chapter 1 - TensorFlow in Practice Specialization

About this Specialization

199,621 recent views

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

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

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

Looking for more advanced TensorFlow content? Check out the new **TensorFlow: Data and Deployment Specialization**.

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

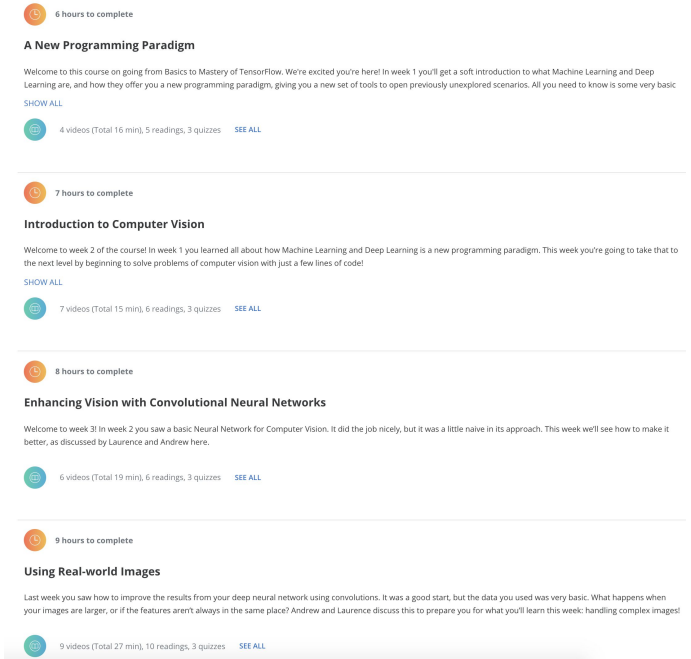
COURSE 4

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

Setup

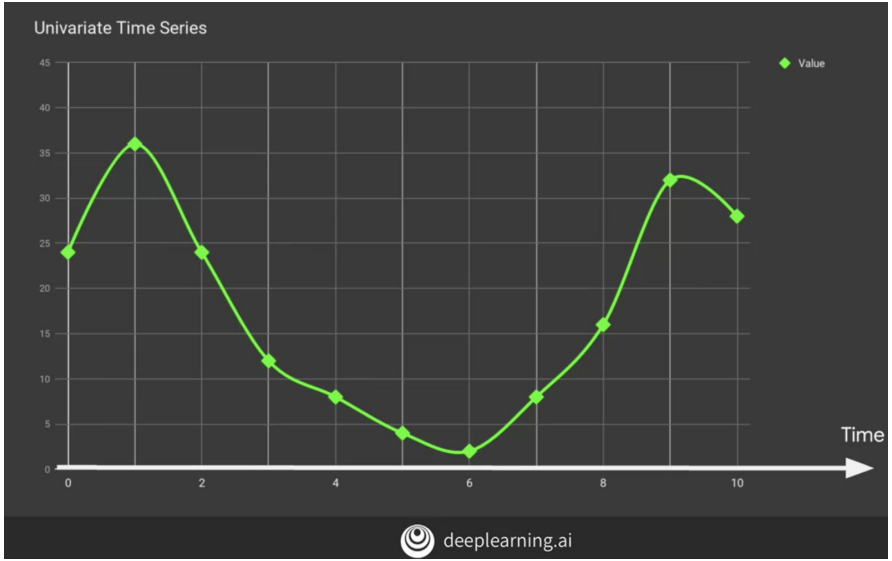
Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. You can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser.

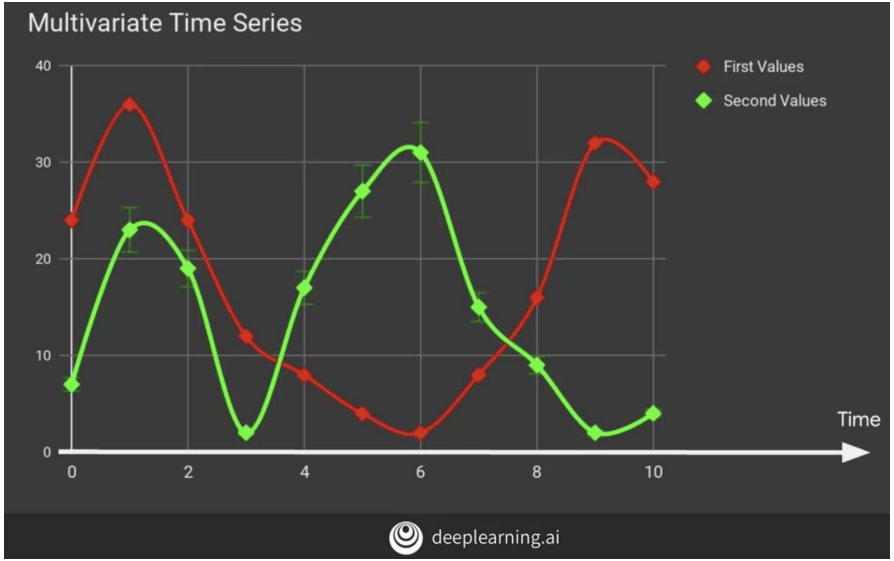
https://colab.research.google.com

Course 4: Sequences, Time Series and Prediction

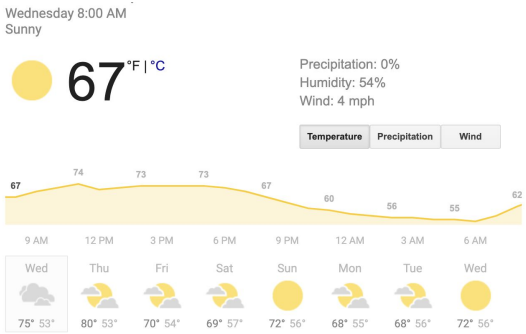
Week 1: Sequences and Prediction
Week 2: Deep Neural Networks for Time Series
Week 3: Recurrent Neural Networks for Time Series
Week 4: Real-world time series data

Sequences and Prediction



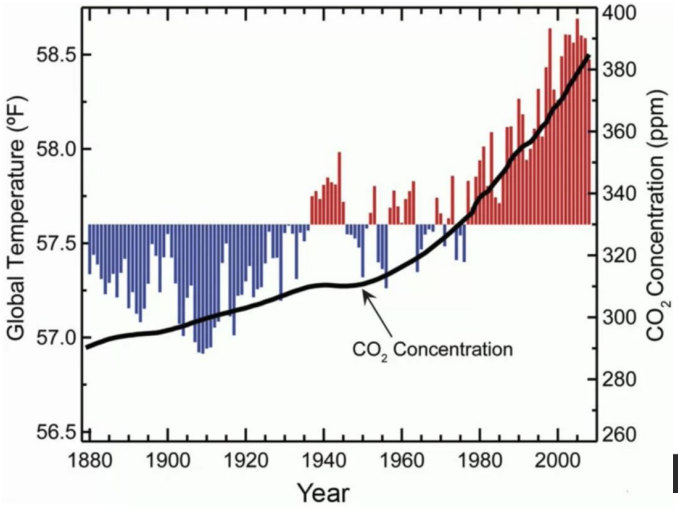


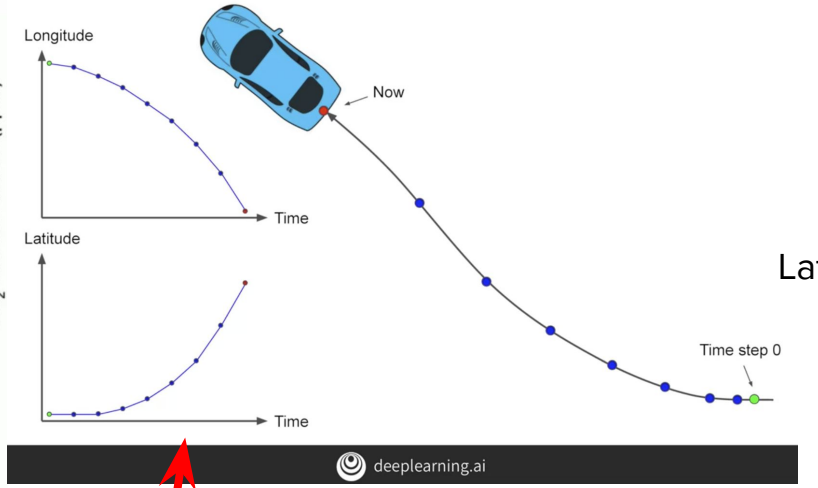
Time series: An ordered sequence of values that are usually equally spaced over time e.g. every day in the weather forecast
Single value at each time step: **uni**variate time series
Multiple values at each time step: **multi**variate time series



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

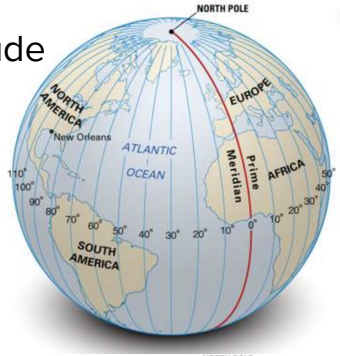
Sequences and Prediction



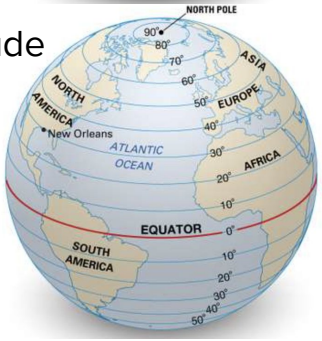


Movement of a body can also be plotted as a series of **univariates** or as a combined **multivariate**

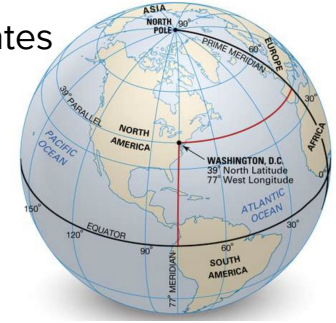
Longitude



Latitude

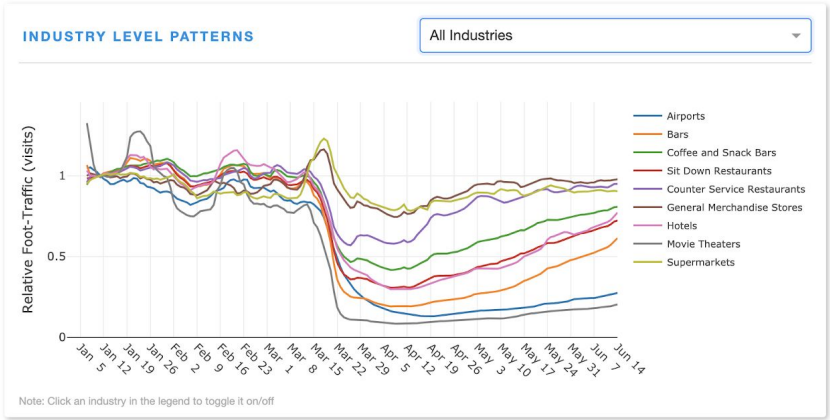


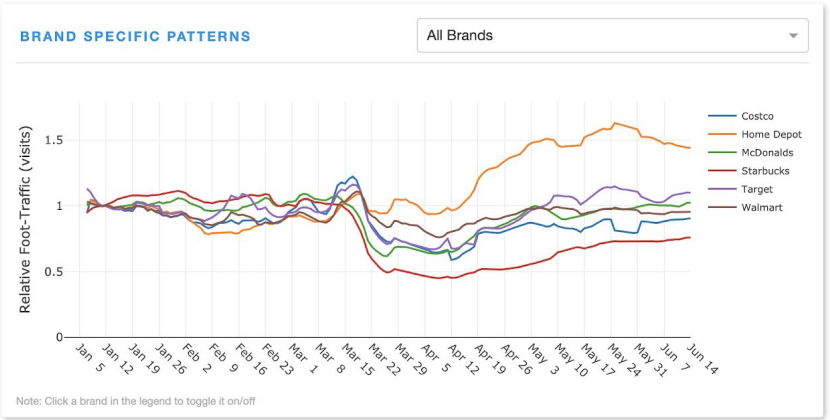
Coordinates

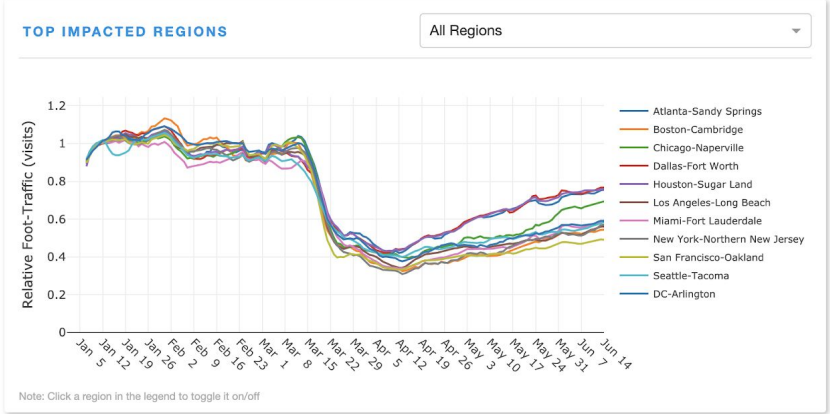


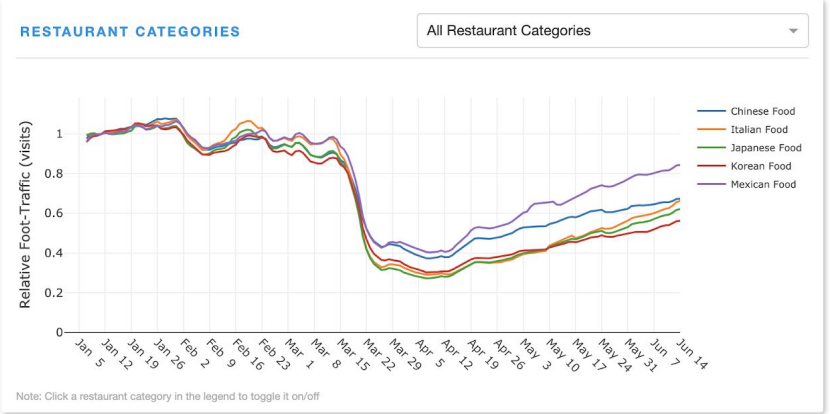
Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction
https://www.britannica.com/science/longitude

Sequences and Prediction



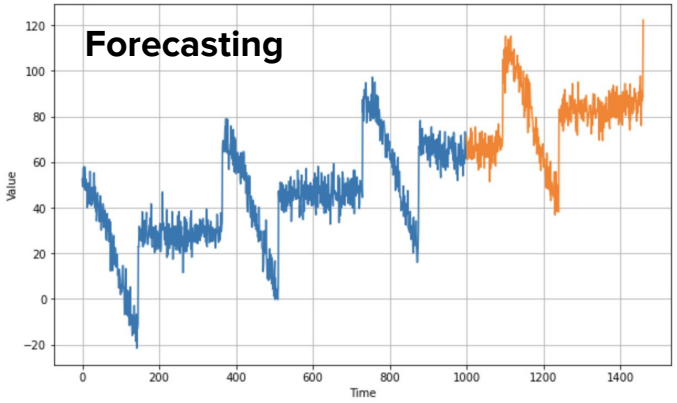






Source: https://www.safegraph.com/dashboard/covid19-commerce-patterns
https://finance.google.com/finance

Sequences and Prediction



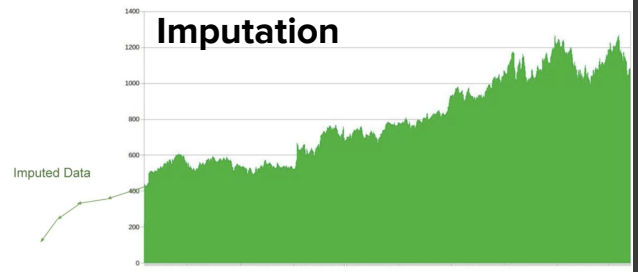
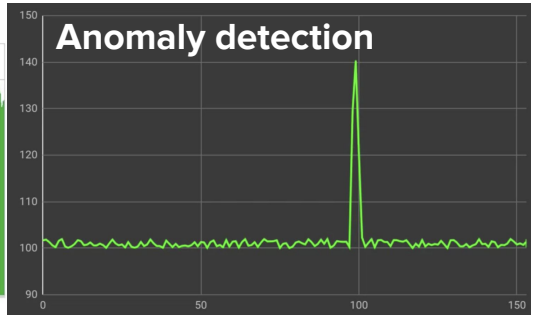


Chart Created by lmoroney@



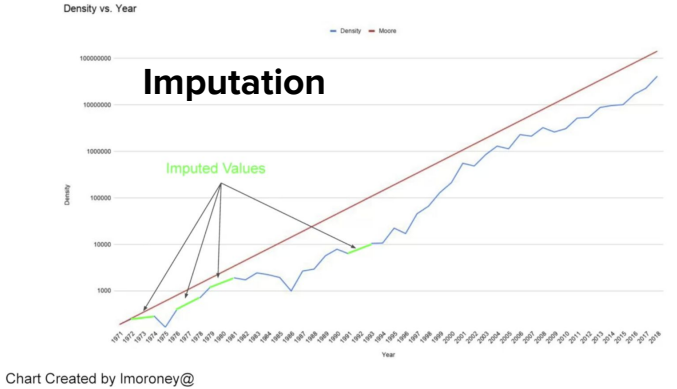
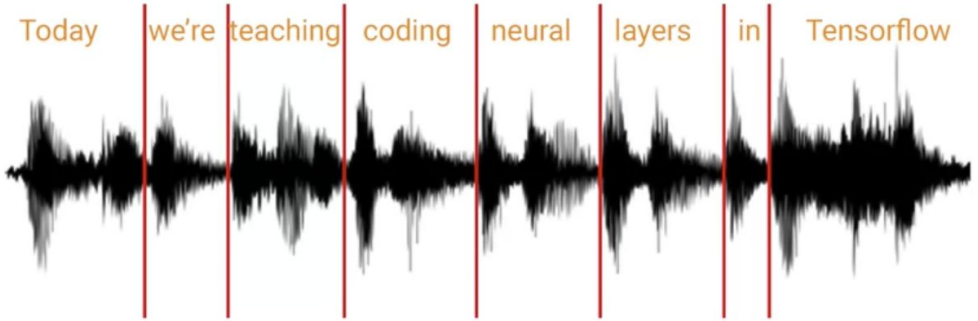


Chart Created by lmoroney@

Patterns / time series analysis



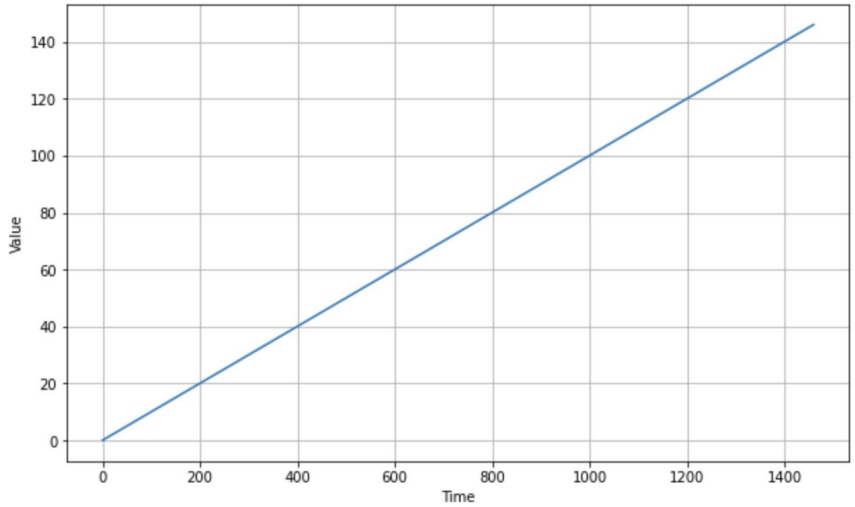
Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

Sequences and Prediction

```
def trend(time, slope=0):
    return slope * time
```

```
time = np.arange(4 * 365 + 1)
print(time)

baseline = 10
series = trend(time, 0.1)
print(series)

plt.figure(figsize=(10, 6))
plot_series(time, series)
plt.show()
```

```
[   0    1    2 ... 1458 1459 1460]
[0.000e+00 1.000e-01 2.000e-01 ... 1.458e+02 1.459e+02 1.460e+02]
```

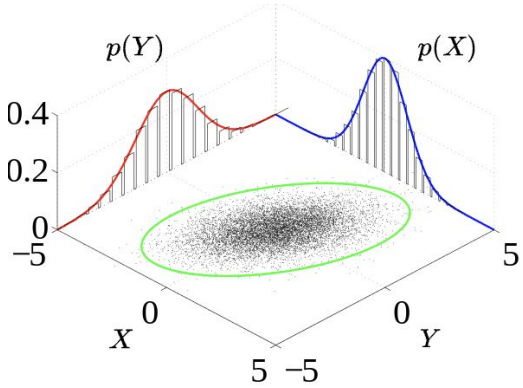


Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

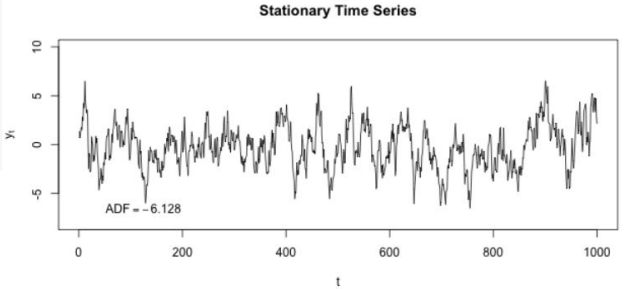
Sequences and Prediction

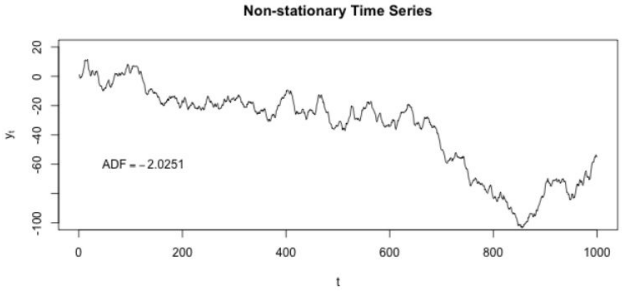
```
def seasonal_pattern(season_time):
    """Just an arbitrary pattern, you can change it if you wish"""
    return np.where(season_time < 0.4,
                    np.cos(season_time * 2 * np.pi),
                    1 / np.exp(3 * season_time))

def seasonality(time, period, amplitude=1, phase=0):
    """Repeats the same pattern at each period"""
    season_time = ((time + phase) % period) / period
    return amplitude * seasonal_pattern(season_time)
```

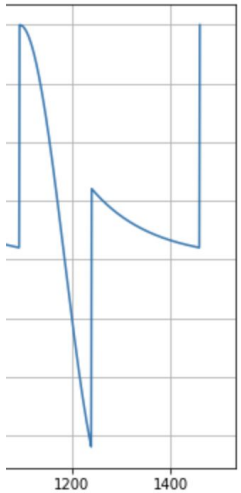
```
baseline = 10
amplitude = 40
series = seasonality(time, period=365, amplitude=amplitude)
```



Stationary Time Series / Process: Its unconditional joint probability distribution does not change when shifted in time. Consequently, parameters such as mean and variance also do not change over time.



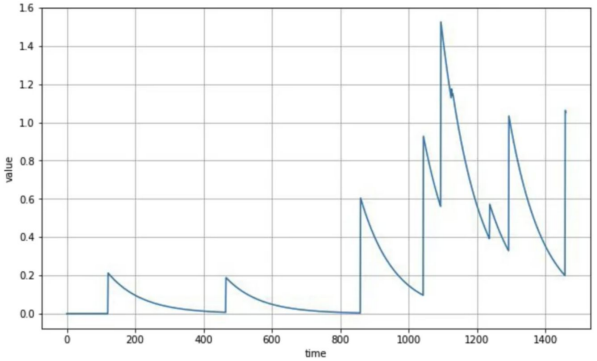




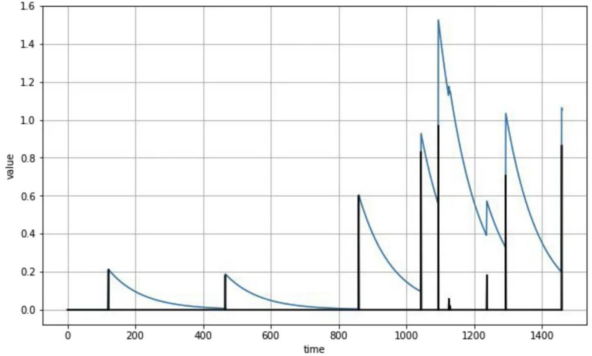
Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction
https://en.wikipedia.org/wiki/Stationary_process

Sequences and Prediction

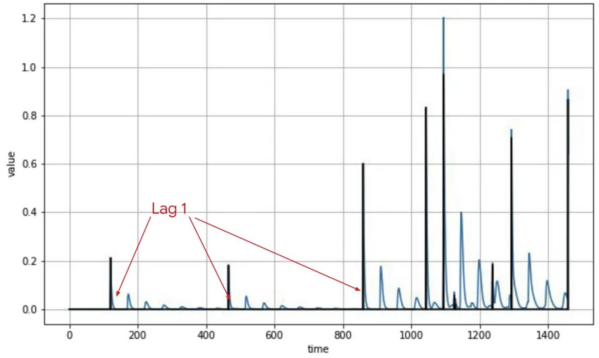
Autocorrelation



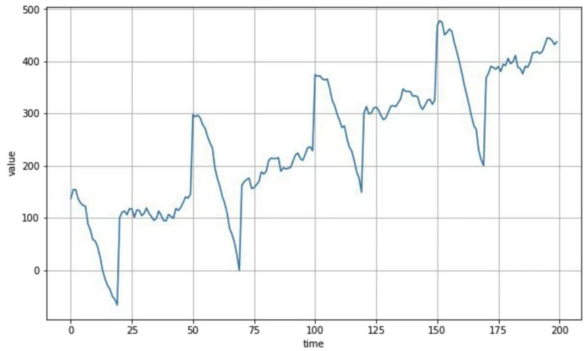
$v(t) = 0.99 \times v(t-1)$ + occasional spike



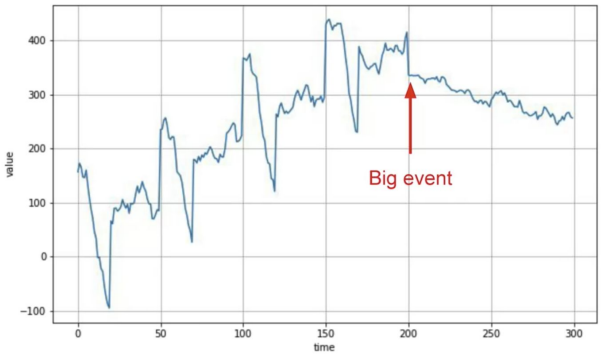
$v(t) = 0.7 \times v(t-1) + 0.2 \times v(t-50)$ + occasional spike



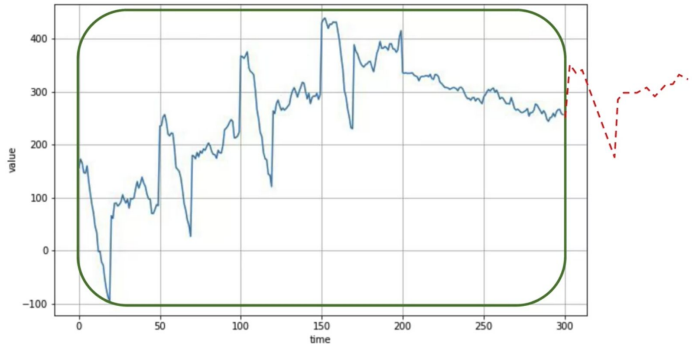
Trend + Seasonality + Autocorrelation + Noise



Non-Stationary Time Series



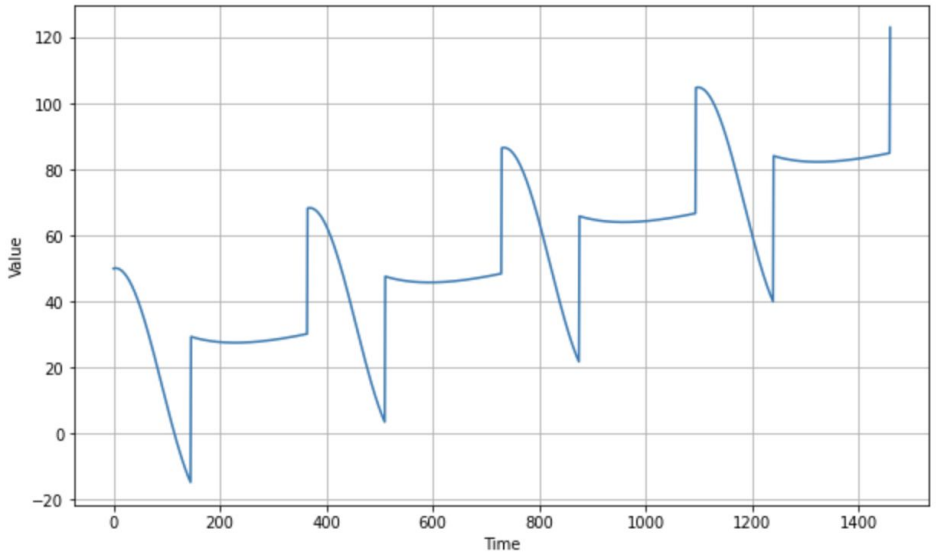
Non-Stationary Time Series



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

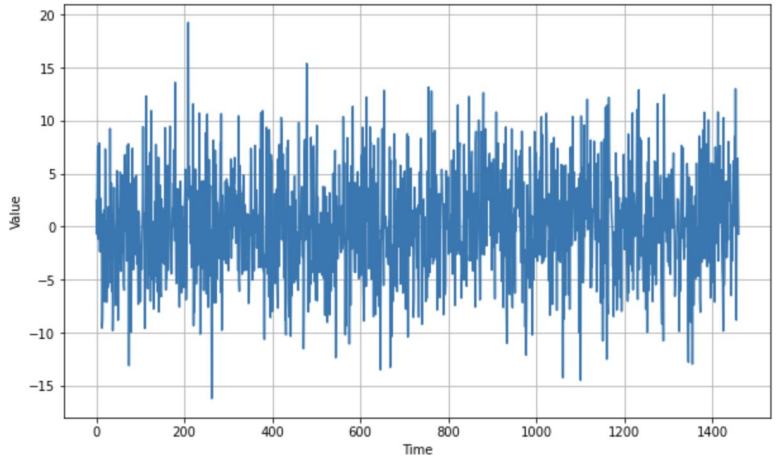
Sequences and Prediction

```
slope = 0.05
series = baseline + trend(time, slope) + seasonality(time, period=365, amplitude=amplitude)

plt.figure(figsize=(10, 6))
plot_series(time, series)
plt.show()
```




```
def white_noise(time, noise_level=1, seed=None):
    rnd = np.random.RandomState(seed)
    return rnd.randn(len(time)) * noise_level
```

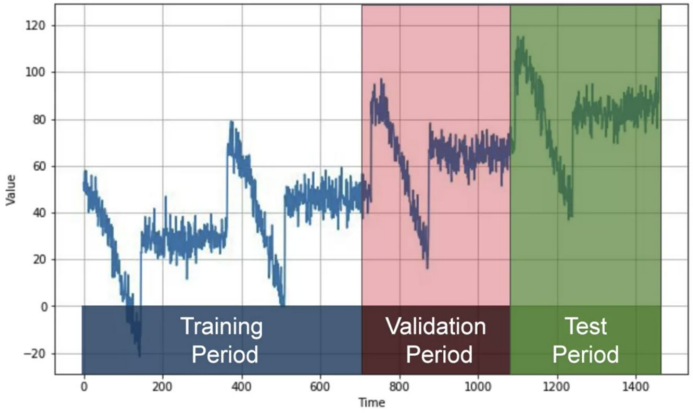
```
noise_level = 5
noise = white_noise(time, noise_level, seed=42)

plt.figure(figsize=(10, 6))
plot_series(time, noise)
plt.show()
```

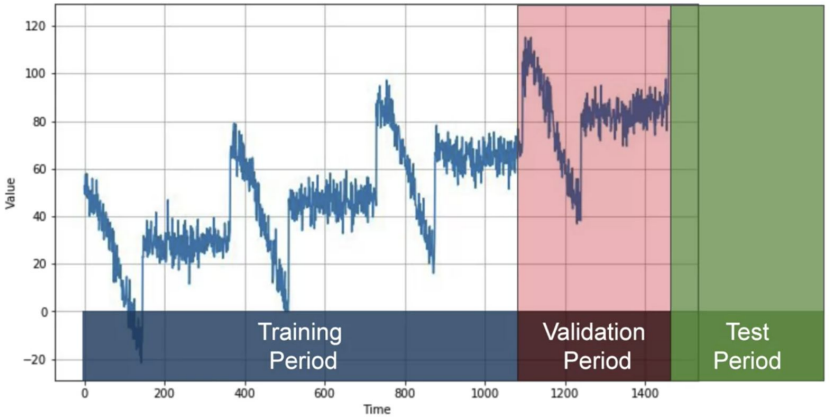


Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

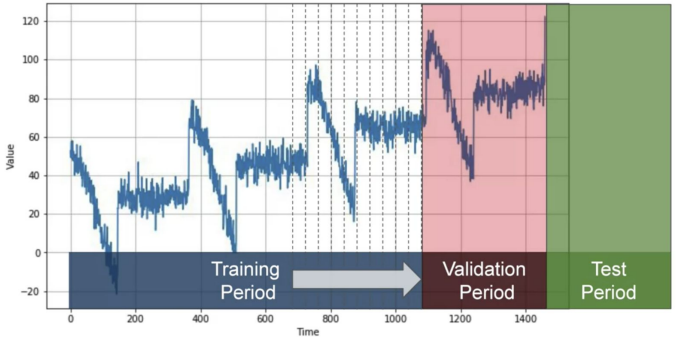
Sequences and Prediction

Fixed Partitioning





Roll-Forward Partitioning



Metrics

```
errors = forecasts - actual

mse = np.square(errors).mean()

rmse = np.sqrt(mse)

mae = np.abs(errors).mean()

mape = np.abs(errors / x_valid).mean()
```

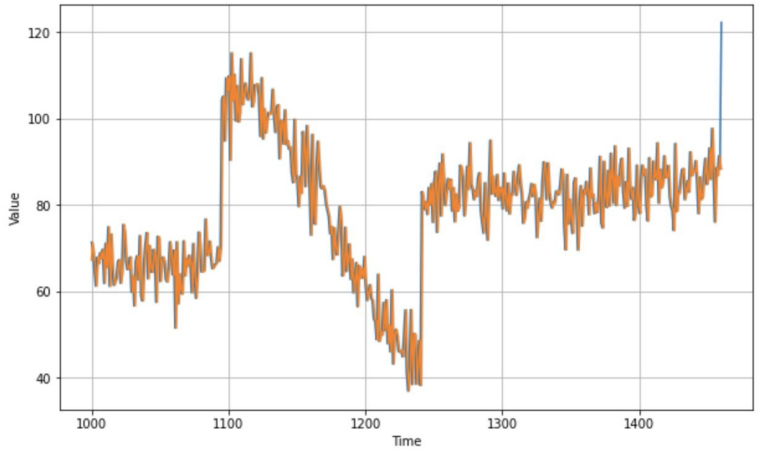
Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

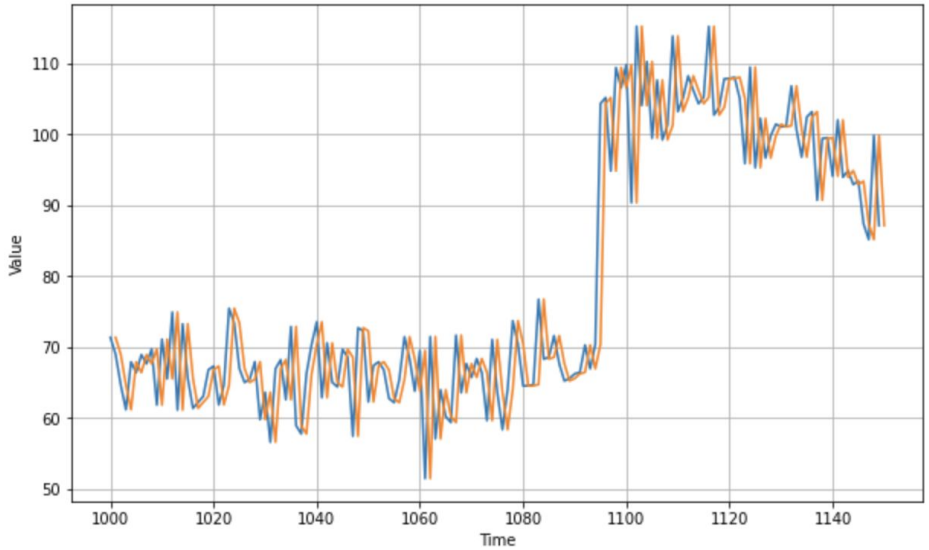
Sequences and Prediction

```
# Create the series
series = baseline + trend(time, slope) + seasonality(time, period=365, amplitude=amplitude)
# 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:]
```

```
naive_forecast = series[split_time - 1:-1]
```






```
print(keras.metrics.mean_squared_error(x_valid, naive_forecast).numpy())
print(keras.metrics.mean_absolute_error(x_valid, naive_forecast).numpy())
```

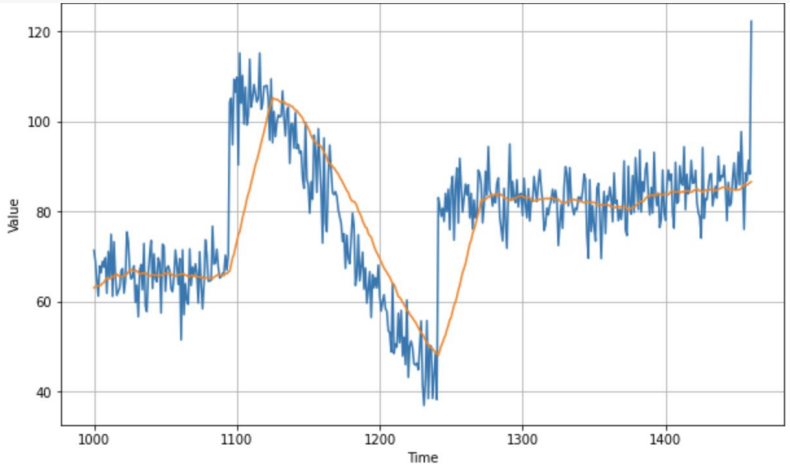
```
61.827538
5.9379086
```

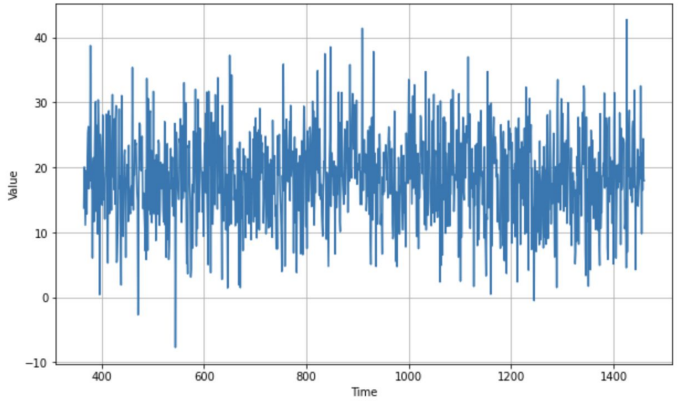
Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

Sequences and Prediction

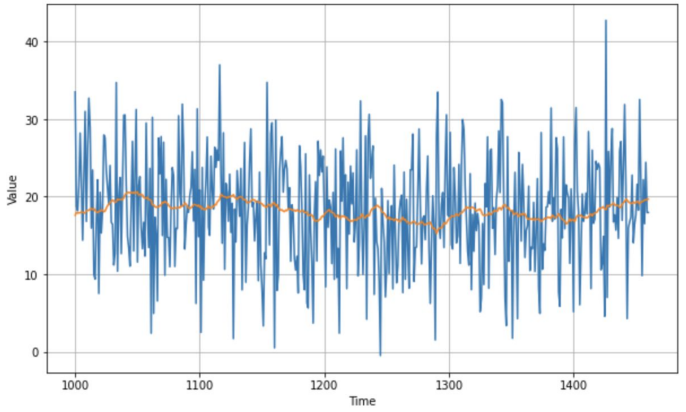
```
diff_series = (series[365:] - series[:-365])
diff_time = time[365:]
```

```
def moving_average_forecast(series, window_size):
  """Forecasts the mean of the last few values.
     If window_size=1, then this is equivalent to naive forecast"""
  forecast = []
  for time in range(len(series) - window_size):
    forecast.append(series[time:time + window_size].mean())
  return np.array(forecast)
moving_avg = moving_average_forecast(series, 30)[split_time - 30:]
```






```
diff_moving_avg = moving_average_forecast(diff_series, 50)[split_time - 365 - 50:]
```




```
print(keras.metrics.mean_squared_error(x_valid, moving_avg).numpy())
print(keras.metrics.mean_absolute_error(x_valid, moving_avg).numpy())
```

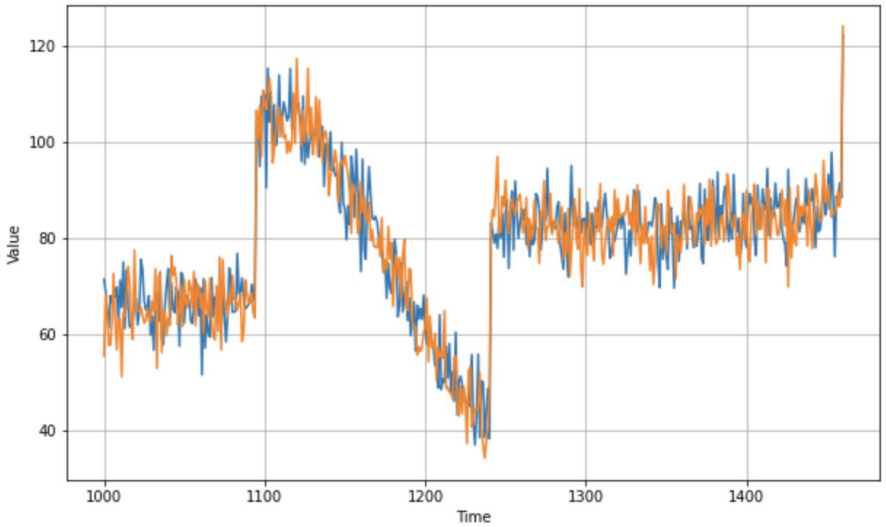
```
106.674576
7.142419
```

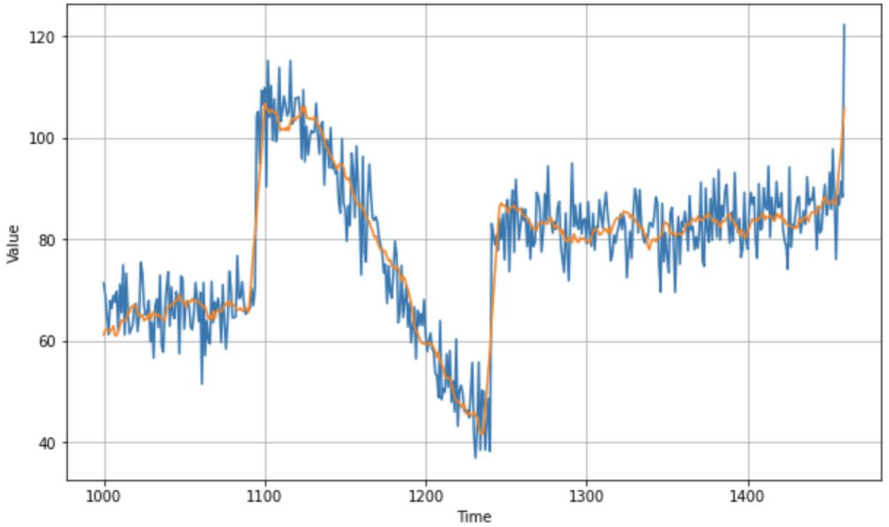
Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

Sequences and Prediction

```
diff_moving_avg_plus_past = series[split_time - 365:-365] + diff_moving_avg
```

```
diff_moving_avg_plus_smooth_past = moving_average_forecast(series[split_time - 370:-360], 10) + diff_moving_avg
```






```
print(keras.metrics.mean_squared_error(x_valid, diff_moving_avg_plu
print(keras.metrics.mean_absolute_error(x_valid, diff_moving_avg_p
```

```
52.973656
5.8393106
```

```
print(keras.metrics.mean_squared_error(x_valid, diff_moving_avg_plus_smooth_past).numpy())
print(keras.metrics.mean_absolute_error(x_valid, diff_moving_avg_plus_smooth_past).numpy())
```

```
33.45226
4.569442
```

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

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

Deep Neural Networks for Time Series

```
dataset = tf.data.Dataset.range(10)
for val in dataset:
    print(val.numpy())
```

```
0
1
2
3
4
5
6
7
8
9
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1)
for window_dataset in dataset:
  for val in window_dataset:
    print(val.numpy(), end=" ")
  print()
```

```
0 1 2 3 4
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
6 7 8 9
7 8 9
8 9
9
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
for window_dataset in dataset:
  for val in window_dataset:
    print(val.numpy(), end=" ")
  print()
```

```
0 1 2 3 4
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
for window in dataset:
  print(window.numpy())
```

```
[0 1 2 3 4]
[1 2 3 4 5]
[2 3 4 5 6]
[3 4 5 6 7]
[4 5 6 7 8]
[5 6 7 8 9]
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
dataset = dataset.map(lambda window: (window[:-1], window[-1:]))
for x,y in dataset:
  print(x.numpy(), y.numpy())
```

```
[0 1 2 3] [4]
[1 2 3 4] [5]
[2 3 4 5] [6]
[3 4 5 6] [7]
[4 5 6 7] [8]
[5 6 7 8] [9]
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=Tru
dataset = dataset.flat_map(lambda window: window.batch(
dataset = dataset.map(lambda window: (window[:-1], winc
dataset = dataset.shuffle(buffer_size=10)
for x,y in dataset:
  print(x.numpy(), y.numpy())
```

```
[0 1 2 3] [4]
[2 3 4 5] [6]
[5 6 7 8] [9]
[1 2 3 4] [5]
[4 5 6 7] [8]
[3 4 5 6] [7]
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
dataset = dataset.map(lambda window: (window[:-1], window[-1:]))
dataset = dataset.shuffle(buffer_size=10)
dataset = dataset.batch(2).prefetch(1)
for x,y in dataset:
  print("x = ", x.numpy())
  print("y = ", y.numpy())
```

```
x =  [[0 1 2 3]
 [5 6 7 8]]
y =  [[4]
 [9]]
x =  [[1 2 3 4]
 [2 3 4 5]]
y =  [[5]
 [6]]
x =  [[3 4 5 6]
 [4 5 6 7]]
y =  [[7]
 [8]]
```

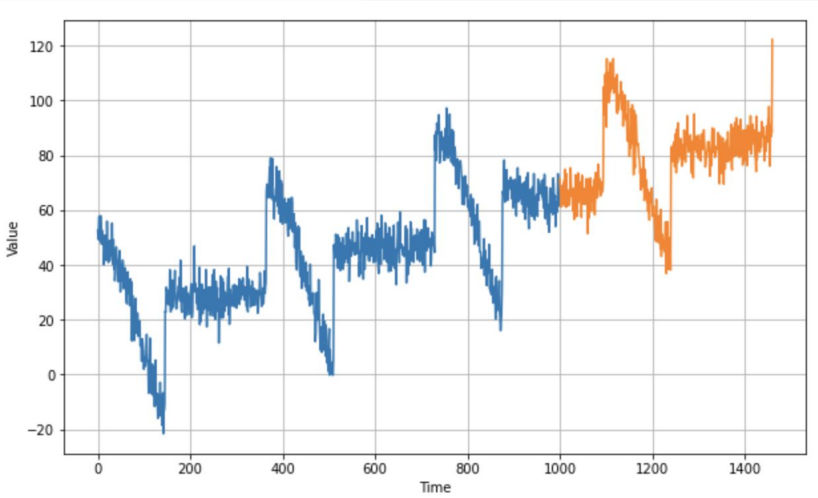
Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

Deep Neural Networks for Time Series

```
time = np.arange(4 * 365 + 1, dtype="float32")
baseline = 10
series = trend(time, 0.1)
baseline = 10
amplitude = 40
slope = 0.05
noise_level = 5
```

```
# Create the series
series = baseline + trend(time, slope) + seasonality(time, period=365, amplitude=amplitude)
# 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
```

```
plot_series(time_train, x_train)
plot_series(time_valid, x_valid)
```




```
def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
  dataset = tf.data.Dataset.from_tensor_slices(series)
  dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)
  dataset = dataset.flat_map(lambda window: window.batch(window_size + 1))
  dataset = dataset.shuffle(shuffle_buffer).map(lambda window: (window[:-1], window[-1]))
  dataset = dataset.batch(batch_size).prefetch(1)
  return dataset
```

```
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)
print(dataset)
l0 = tf.keras.layers.Dense(1, input_shape=[window_size])
model = tf.keras.models.Sequential([l0])


model.compile(loss="mse", optimizer=tf.keras.optimizers.SGD(lr=1e-6, momentum=0.9), metrics=['accuracy'])
model.fit(dataset,epochs=50,verbose=1)
```

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

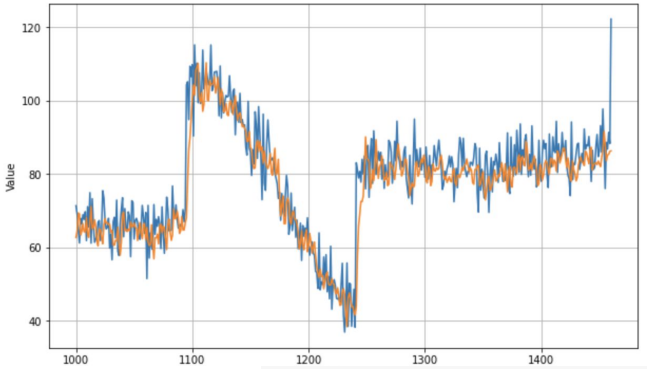
Deep Neural Networks for Time Series

```
Epoch 40/50
31/31 [==============================] - 0s 2ms/step - loss: 54.7502 - accuracy: 0.0000e+00
Epoch 41/50
31/31 [==============================] - 0s 2ms/step - loss: 52.2573 - accuracy: 0.0000e+00
Epoch 42/50
31/31 [==============================] - 0s 2ms/step - loss: 52.1144 - accuracy: 0.0000e+00
Epoch 43/50
31/31 [==============================] - 0s 3ms/step - loss: 51.1316 - accuracy: 0.0000e+00
Epoch 44/50
31/31 [==============================] - 0s 3ms/step - loss: 51.8674 - accuracy: 0.0000e+00
Epoch 45/50
31/31 [==============================] - 0s 2ms/step - loss: 50.2164 - accuracy: 0.0000e+00
Epoch 46/50
31/31 [==============================] - 0s 2ms/step - loss: 49.6189 - accuracy: 0.0000e+00
Epoch 47/50
31/31 [==============================] - 0s 2ms/step - loss: 50.3166 - accuracy: 0.0000e+00
Epoch 48/50
31/31 [==============================] - 0s 2ms/step - loss: 49.0882 - accuracy: 0.0000e+00
Epoch 49/50
31/31 [==============================] - 0s 2ms/step - loss: 49.3388 - accuracy: 0.0000e+00
Epoch 50/50
31/31 [==============================] - 0s 2ms/step - loss: 50.4229 - accuracy: 0.0000e+00
```

```
print("Layer weights {}".format(l0.get_weights()))
```

```
Layer weights [array([[ 0.04292529],
       [-0.06003034],
       [-0.12568147],
       [ 0.12413348],
       [ 0.13928801],
       [-0.04836606],
       [-0.05267965],
       [ 0.05041023],
       [-0.05249092],
       [-0.17136675],
       [-0.03027576],
       [ 0.01343512],
       [ 0.09165959],
       [ 0.08489882],
       [ 0.13626154],
       [ 0.08202708],
       [-0.00438521],
       [ 0.12374628],
       [ 0.29035616],
       [ 0.34197554]], dtype=float32), array([0.01085013], dtype=float32)]
```

```
forecast = []

for time in range(len(series) - window_size):
  forecast.append(model.predict(series[time:time + window_size][np.newaxis]))

forecast = forecast[split_time-window_size:]
results = np.array(forecast)[:, 0, 0]


plt.figure(figsize=(10, 6))

plot_series(time_valid, x_valid)
plot_series(time_valid, results)
```

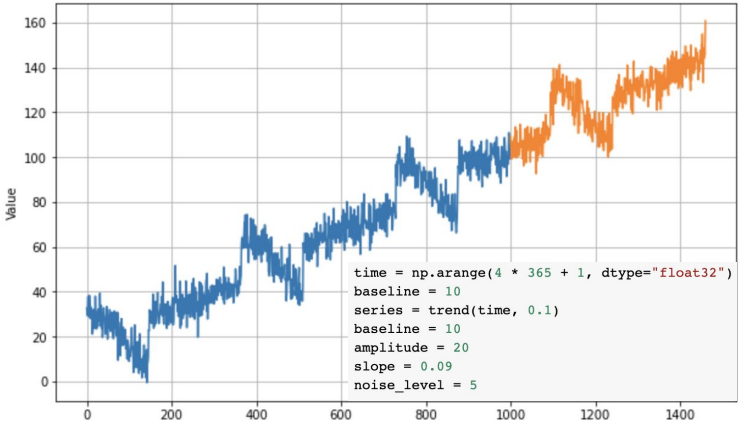


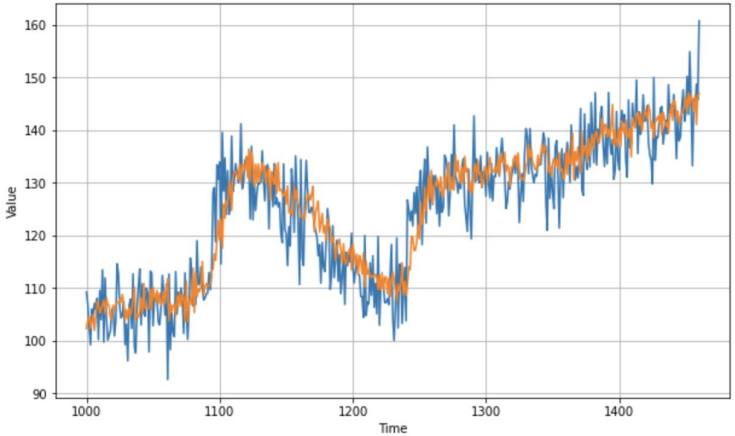

```
tf.keras.metrics.mean_absolute_error(x_valid, results).numpy()
```

```
5.2385697
```

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

Deep Neural Networks for Time Series






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


model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(10, 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-6, momentum=0.9))
model.fit(dataset,epochs=my_epochs,verbose=1)
```

```
Epoch 97/100
31/31 [==============================] - 0s 2ms/step - loss: 38.6709
Epoch 98/100
31/31 [==============================] - 0s 2ms/step - loss: 38.8540
Epoch 99/100
31/31 [==============================] - 0s 2ms/step - loss: 39.5055
Epoch 100/100
31/31 [==============================] - 0s 2ms/step - loss: 38.8792
```

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

```
4.678847
```

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

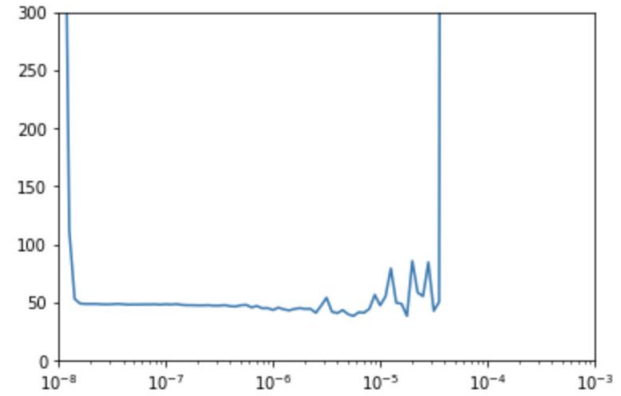
Deep Neural Networks for Time Series

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

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

lr_schedule = tf.keras.callbacks.LearningRateScheduler(lambda epoch: 1e-8 * 10**(epoch / 20))

optimizer = tf.keras.optimizers.SGD(lr=1e-8, momentum=0.9)
model.compile(loss="mse", optimizer=optimizer)

history = model.fit(dataset, epochs=my_epochs, callbacks=[lr_schedule], verbose=1)
```

```
Epoch 97/100
31/31 [==============================] - 0s 2ms/step - loss: 742.2804 - lr: 6.3096e-04
Epoch 98/100
31/31 [==============================] - 0s 2ms/step - loss: 725.2363 - lr: 7.0795e-04
Epoch 99/100
31/31 [==============================] - 0s 2ms/step - loss: 727.0015 - lr: 7.9433e-04
Epoch 100/100
31/31 [==============================] - 0s 3ms/step - loss: 739.7311 - lr: 8.9125e-04
```

```
lrs = 1e-8 * (10 ** (np.arange(my_epochs) / 20))
plt.semilogx(lrs, history.history["loss"])
plt.axis([1e-8, 1e-3, 0, 300])
```

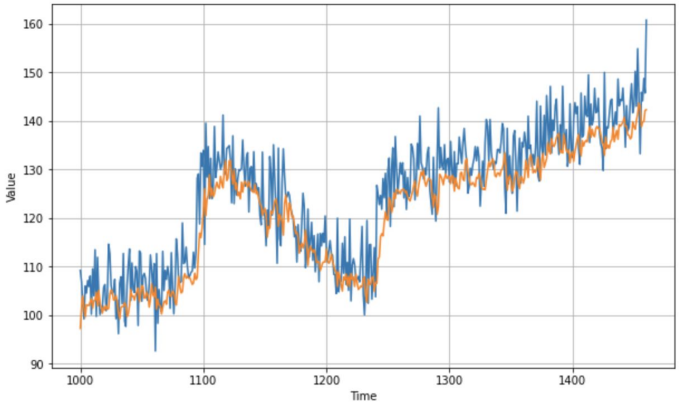
```
(1e-08, 0.001, 0.0, 300.0)
```



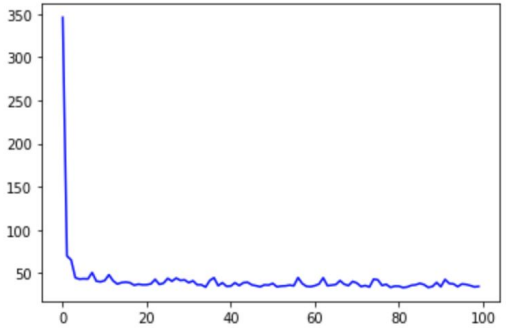
Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

Deep Neural Networks for Time Series

```
window_size = 30
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)

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

optimizer = tf.keras.optimizers.SGD(lr=8e-6, momentum=0.9)
model.compile(loss="mse", optimizer=optimizer)
history = model.fit(dataset, epochs=my_epochs, verbose=0)
```

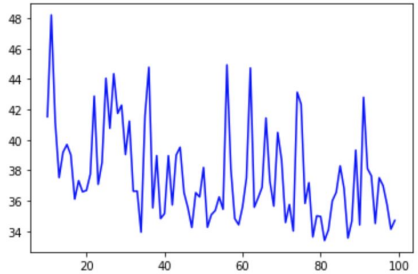



```
loss = history.history['loss']
epochs = range(len(loss))
plt.plot(epochs, loss, 'b', label='Training Loss')
plt.show()
```




```
# Plot all but the first 10
loss = history.history['loss']
epochs = range(10, len(loss))
plot_loss = loss[10:]
print(plot_loss)
plt.plot(epochs, plot_loss, 'b', label='Training Loss')
plt.show()
```

```
[41.522789001464844, 48.20726776123047, 41.1973762512207,
```

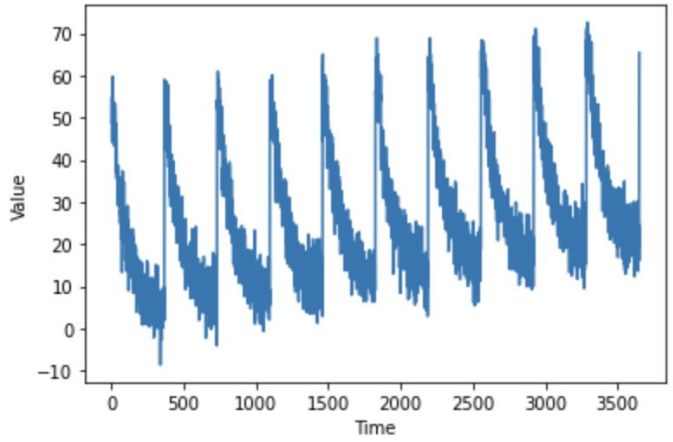


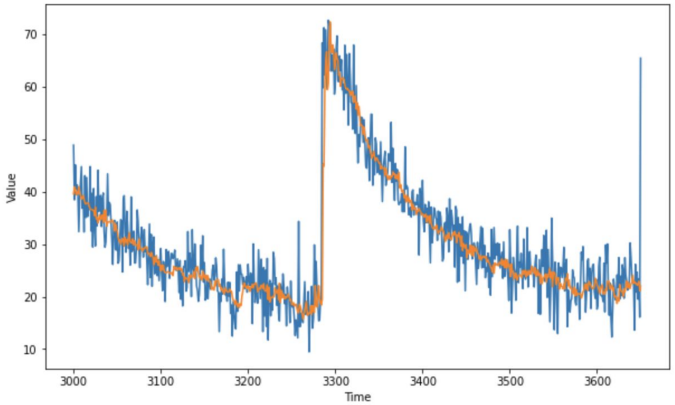

```
tf.keras.metrics.mean_absolute_error(x_valid, results).numpy()
```

```
5.301772
```

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

Deep Neural Networks for Time Series






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


model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(10, 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(learning_rate=1e-6, momentum=0.9))
model.fit(dataset,epochs=my_epochs,verbose=1)
```

```
time = np.arange(10 * 365 + 1,
series = trend(time, 0.1)
baseline = 0
amplitude = 50
slope = 0.005
noise_level = 4
```

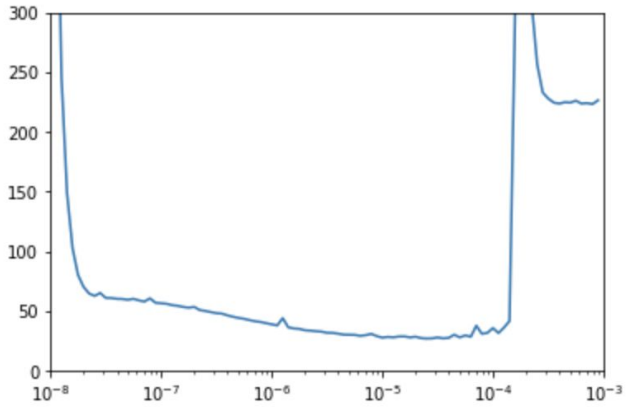
```
Total params: 331
Trainable params: 331
Non-trainable params: 0
```

```
tf.keras.metrics.mean_absolute_error(
# EXPECTED OUTPUT
# A Value less than 3
```

```
3.574421
```

```
lrs = 1e-8 * (10 ** (np.arange(my_epochs) / 20))
plt.semilogx(lrs, history.history["loss"])
plt.axis([1e-8, 1e-3, 0, 300])
```

```
(1e-08, 0.001, 0.0, 300.0)
```



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

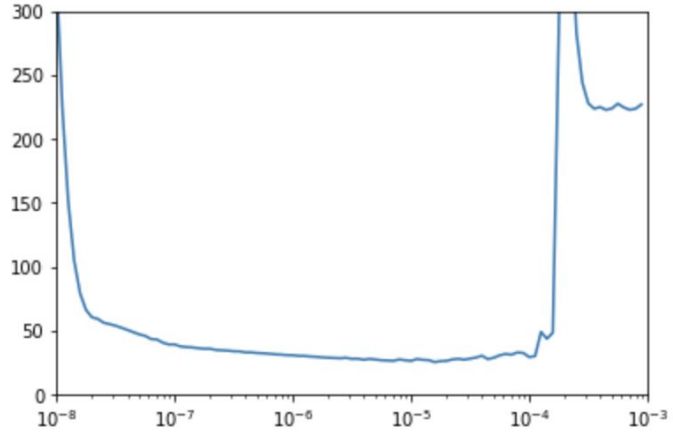
Deep Neural Networks for Time Series

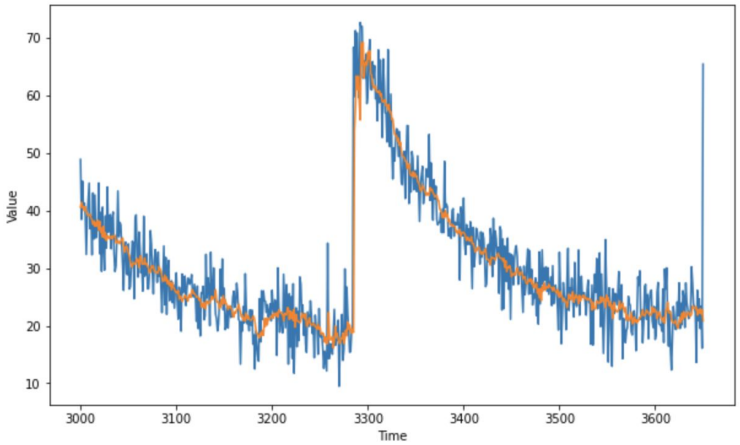
```
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(100, input_shape=[window_size], activation='relu'),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

model.compile(loss='mse', optimizer=tf.keras.optimizers.SGD(learning_rate=1e-6, momentum=0.9))
model.fit(dataset,epochs=my_epochs,verbose=1)
```

```
lrs = 1e-8 * (10 ** (np.arange(my_epochs) / 20))
plt.semilogx(lrs, history.history["loss"])
plt.axis([1e-8, 1e-3, 0, 300])
```

```
Total params: 3,121
Trainable params: 3,121
Non-trainable params: 0
```

```
(1e-08, 0.001, 0.0, 300.0)
```






```
tf.keras.metrics.mean_absolute_error
# EXPECTED OUTPUT
# A Value less than 3
```

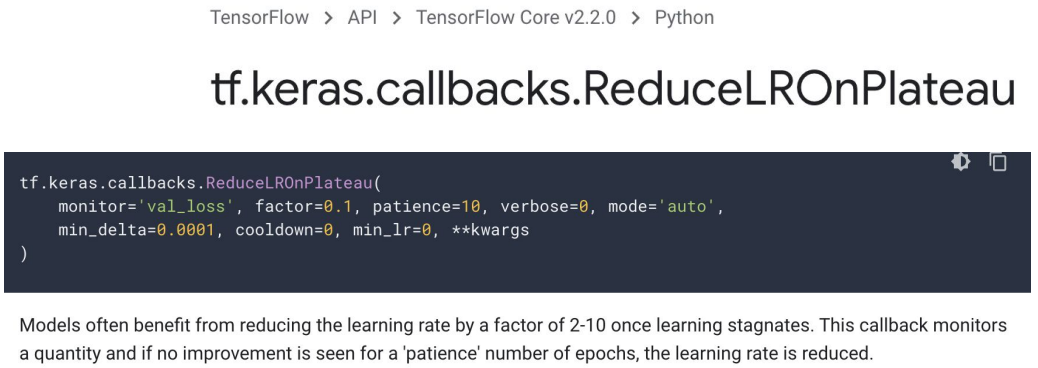
```
3.5475323
```

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`

Lawrence’s LR optimization?

For consistent results:

```
import random
seed = 52
tf.random.set_seed(seed)
random.seed = seed
```

Set all runs to epochs=500

Restart runtime and run all each test.

Lawrence's LR optimization results (seed=52)

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

   `mae  4.577156`

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

   `mae  4.5869045`

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

   `mae  4.632053`

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

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

   `mae  4.528657`

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...

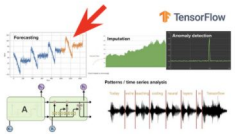
Online event

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...

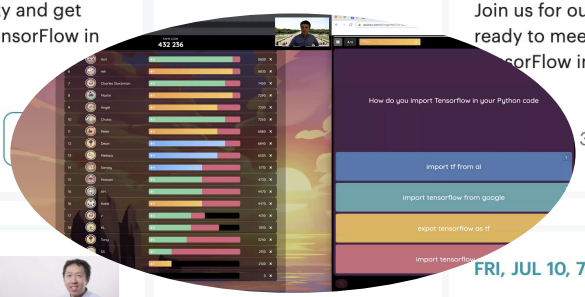


Online event

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...

9 attendees

1

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?...

14 attendees

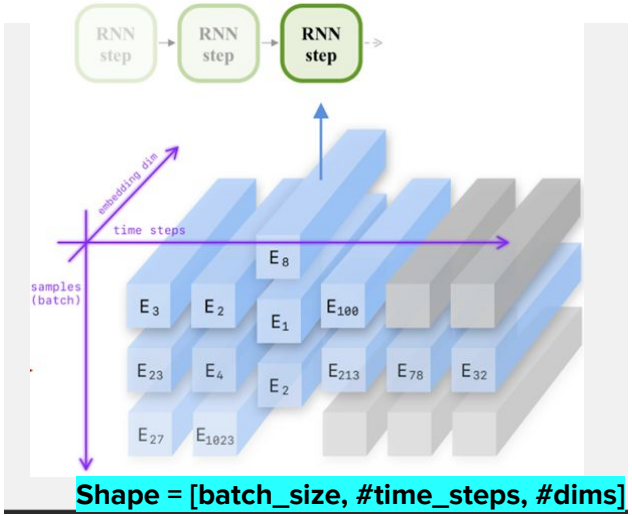
Attend

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

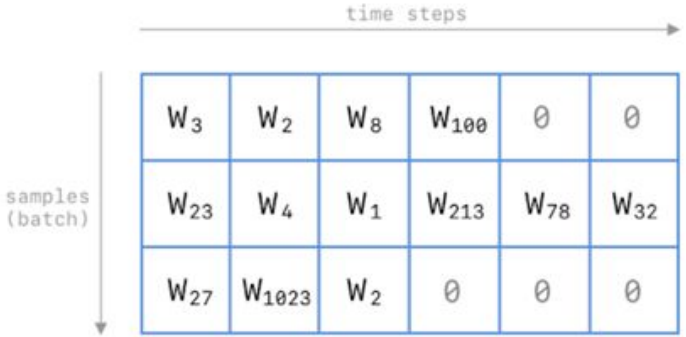
Course 4: Sequences, Time Series and Prediction

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

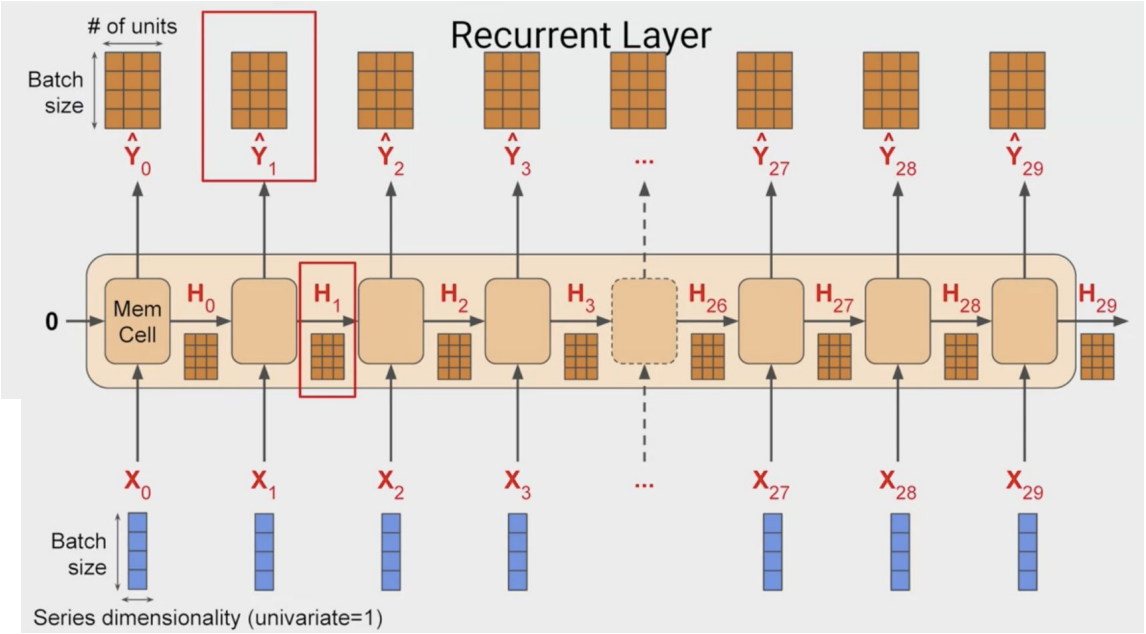
Recurrent Neural Networks for Time Series



Shape = [batch_size, #time_steps, #dims]



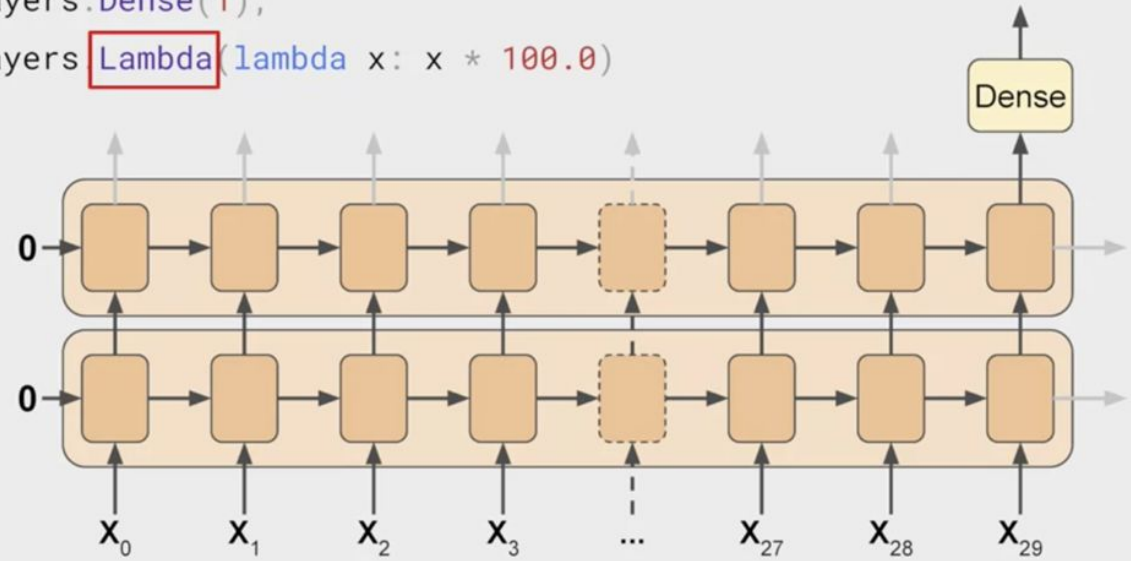
return_sequences=True



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)
])
```



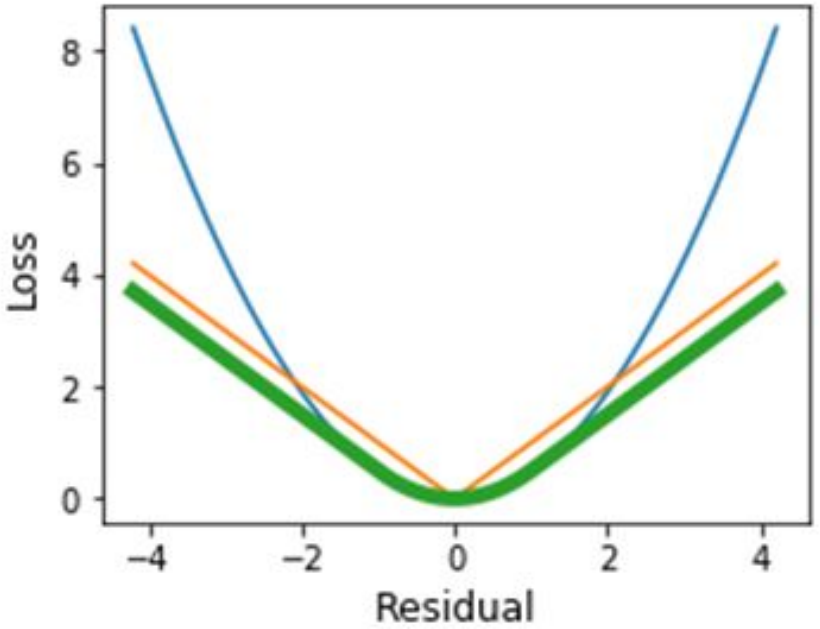
Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

Recurrent Neural Networks for Time Series

Loss Function: Huber

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta\,|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

```
from matplotlib import pyplot as plt
from matplotlib.pyplot import figure
import numpy as np
X=np.linspace(-4.2,4.2,841)
mse_loss = 0.5*X**2
mae_loss = np.abs(X)
huber = lambda x: (0.5*x**2 if (abs(x)<=1) else (abs(x)-0.5))
huber_loss = np.array(list(map(huber,X)))
```




```
model.compile(loss=tf.keras.losses.Huber(), ...
```

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

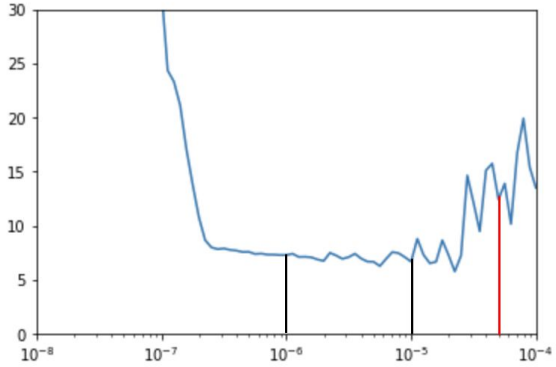
Recurrent Neural Networks for Time Series

```
# 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)
```

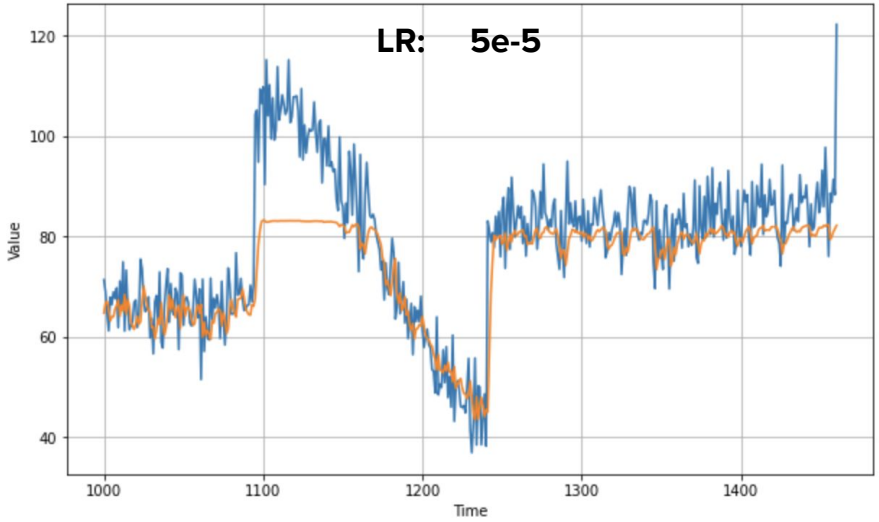


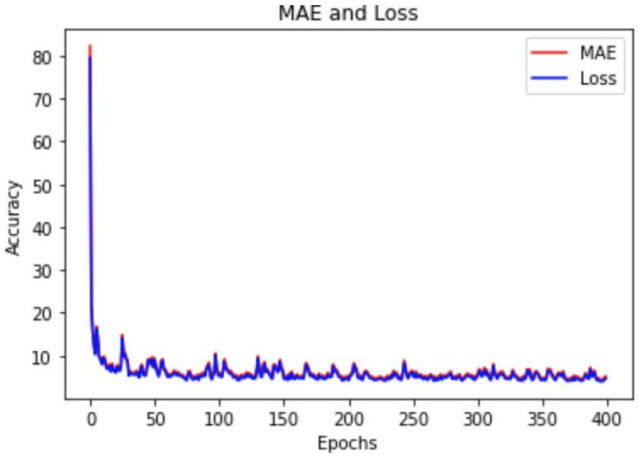

```
tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

train_set = windowed_dataset(x_train, window_size, batch_size=128, shuffle_buffer=shuffle_buffer_size)

model = tf.keras.models.Sequential([
  tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),
                      input_shape=[None]),
  tf.keras.layers.SimpleRNN(40, return_sequences=True),
  tf.keras.layers.SimpleRNN(40),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 100.0)
])

lr_schedule = tf.keras.callbacks.LearningRateScheduler(
    lambda epoch: 1e-8 * 10**(epoch / 20))
optimizer = tf.keras.optimizers.SGD(lr=1e-8, momentum=0.9)
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])
history = model.fit(train_set, epochs=my_epoch, callbacks=[lr_schedule])
```

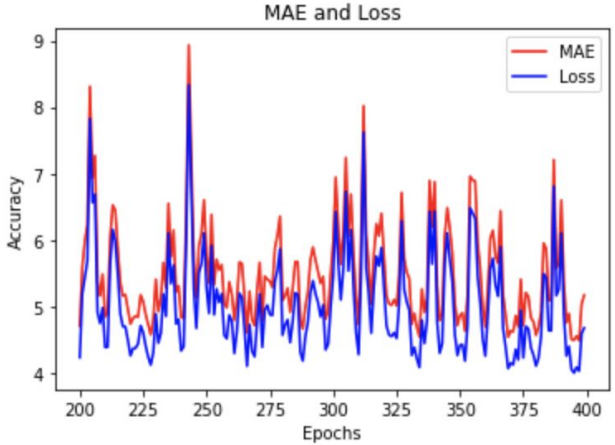
```
optimizer = tf.keras.optimizers.SGD(lr=5e-5, momentum=0.9)
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])
history = model.fit(dataset,epochs=my_epoch*4)
```

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

Recurrent Neural Networks for Time Series





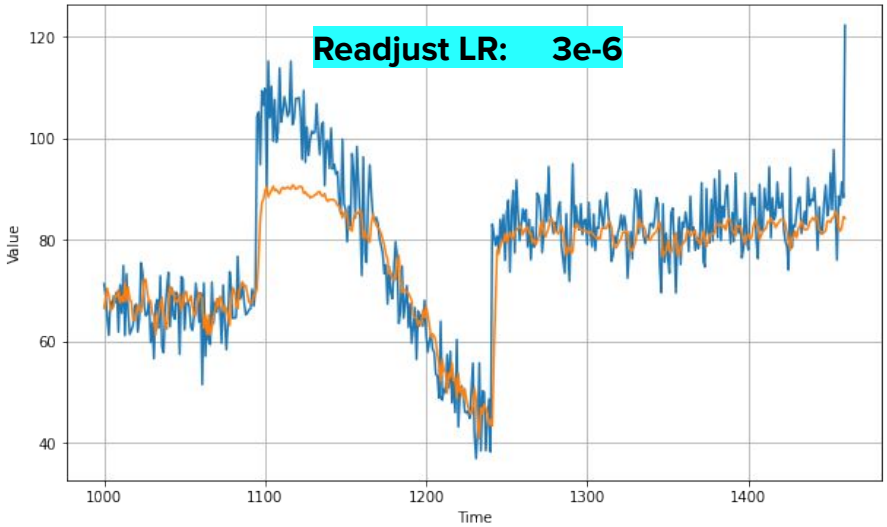


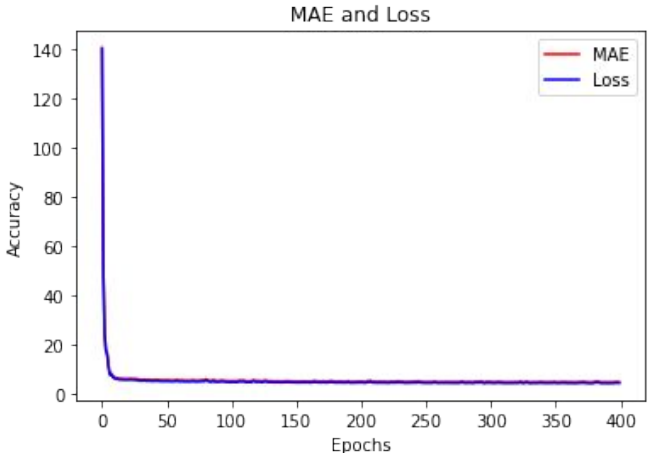

```
tf.keras.metrics.mean_absolute_error(x_valid, results).numpy()
```

```
6.824173
```

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

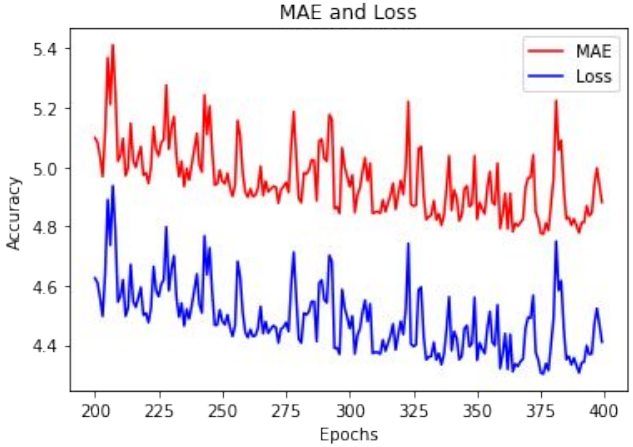
Recurrent Neural Networks for Time Series



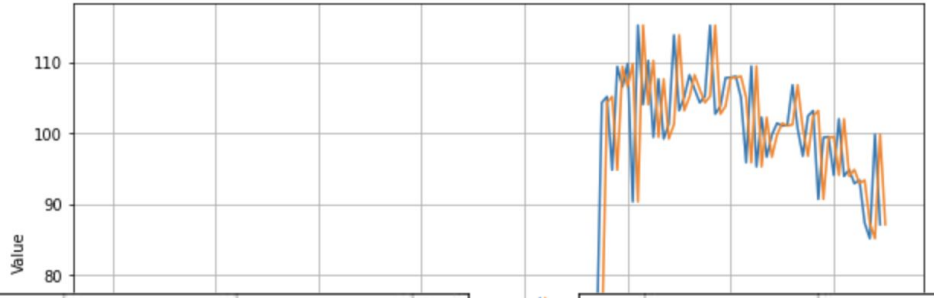



```
tf.keras.metrics.mean_absolute_error(x_valid, results).numpy()
```

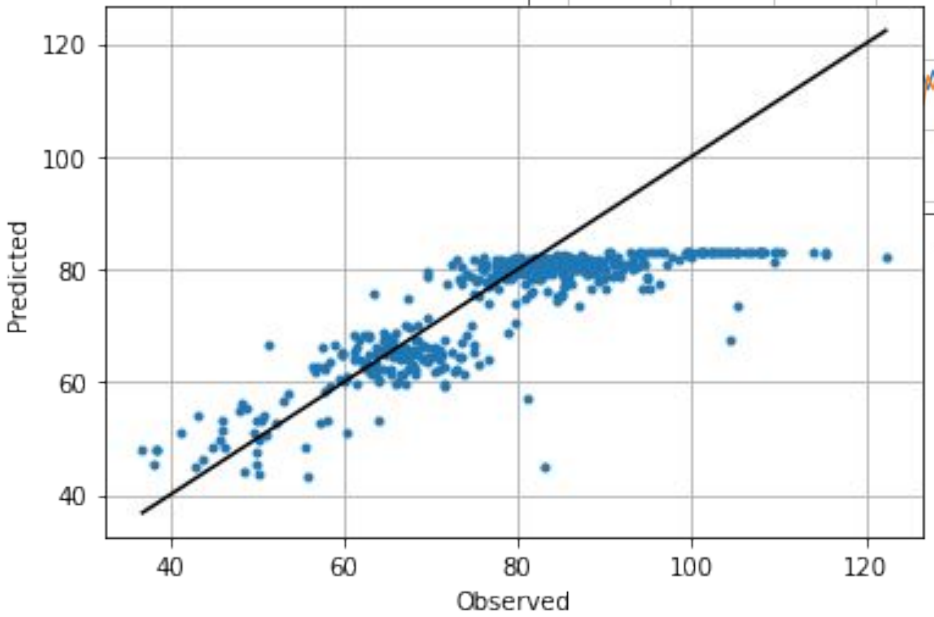
```
5.80845
```



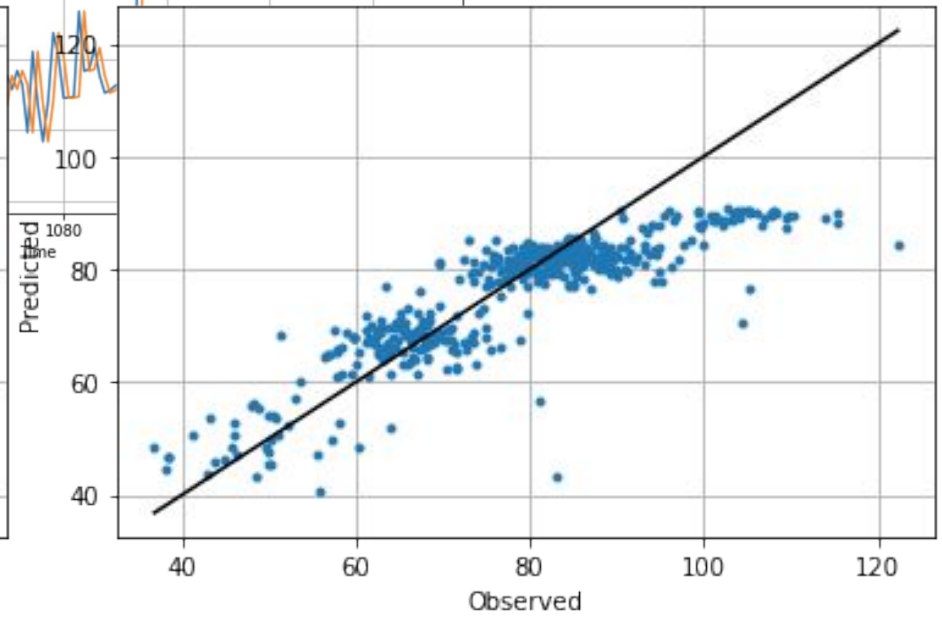
Recurrent Neural Networks for Time Series



LR: 5e-5



LR: 3e-6



Recurrent Neural Networks for Time Series

```
# Create the series
series = baseline + trend(time, slope) + seasonality(time, period=365, amplitude=amplitude)
# 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
```

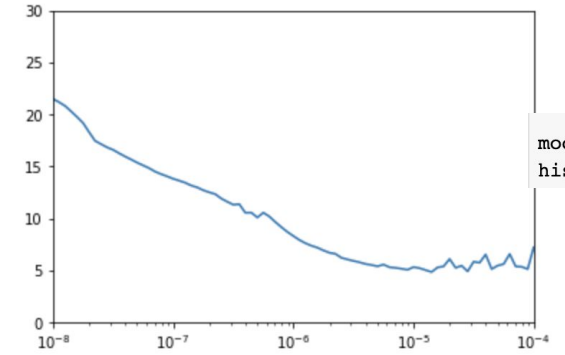
```
time = np.arange(4 * 365 + 1,
baseline = 10
series = trend(time, 0.1)
baseline = 10
amplitude = 40
slope = 0.05
noise_level = 5
```

```
tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

tf.keras.backend.clear_session()
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
  tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1), input_shape=[None]),
  tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32, return_sequences=True)),
  tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 100.0)
])

lr_schedule = tf.keras.callbacks.LearningRateScheduler(
    lambda epoch: 1e-8 * 10**(epoch / 20))
optimizer = tf.keras.optimizers.SGD(lr=1e-8, momentum=0.9)
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])
history = model.fit(dataset, epochs=100, callbacks=[lr_schedule])
```

https

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

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



Forget
cell cor

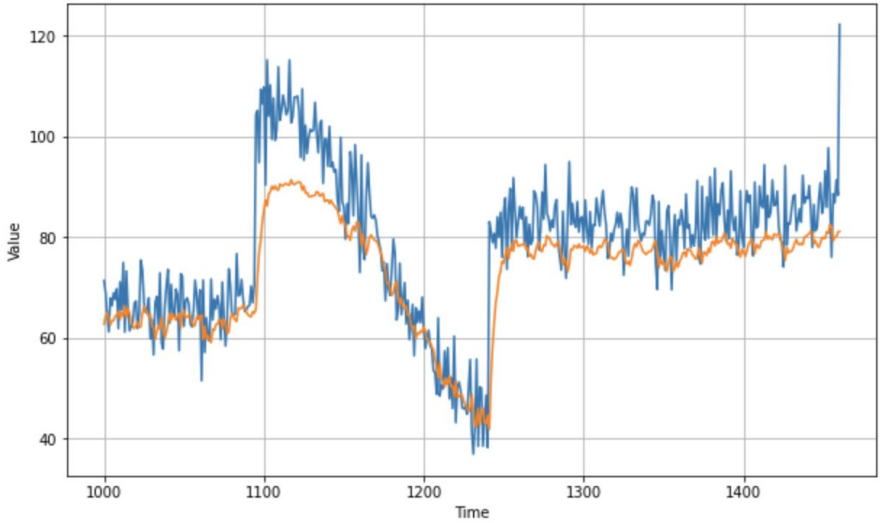
Comput
forget

Cor
in

```
model.compile(loss="mae", optimizer=tf.keras.optimizers.SGD(lr=1e-5, momentum=0.9),metrics=["mae"])
history = model.fit(dataset,epochs=400,verbose=0)
```

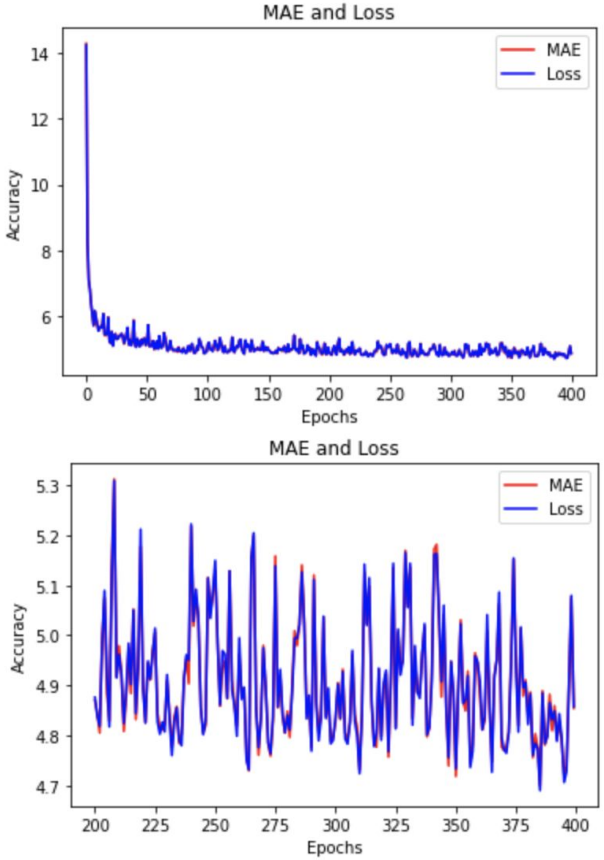
Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

Recurrent Neural Networks for Time Series




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

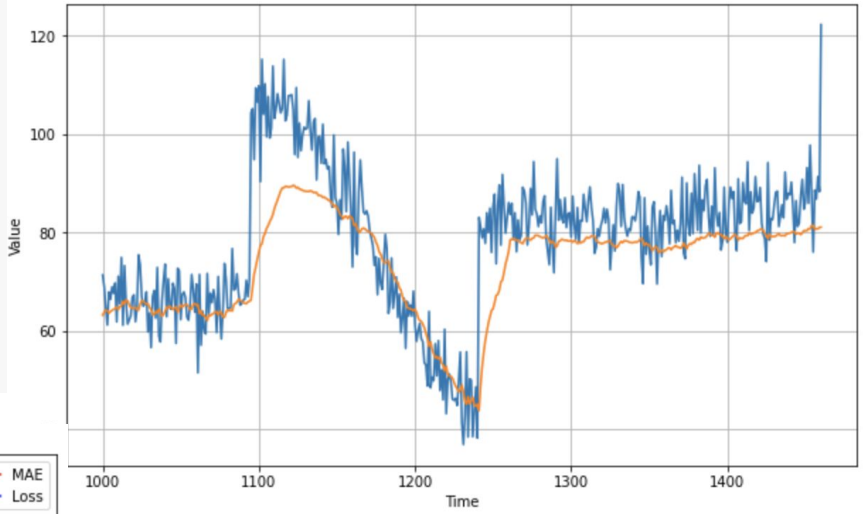
```
6.95316466.9531646
```

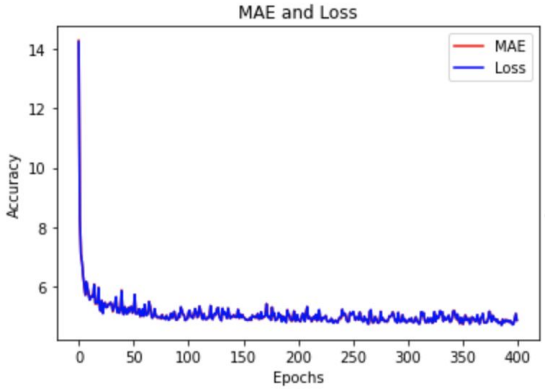


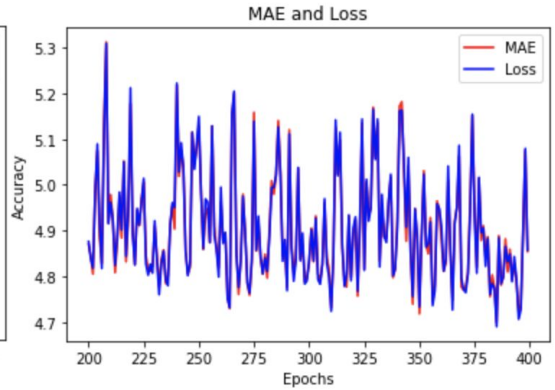
Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

Recurrent Neural Networks for Time Series

```
tf.keras.backend.clear_session()
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
  tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),
                      input_shape=[None]),
  tf.keras.layers.LSTM(32, return_sequences=True),
  tf.keras.layers.LSTM(32),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 100.0)
])


model.compile(loss="mae", optimizer=tf.keras.optimizers.SGD(lr=1e-6, momentum=0.9))
history = model.fit(dataset,epochs=100, verbose=0)
```







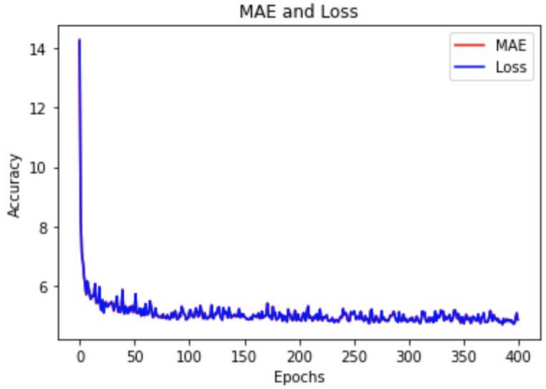

```
tf.keras.metrics.mean_absolute_error(x_valid, results).numpy()
```

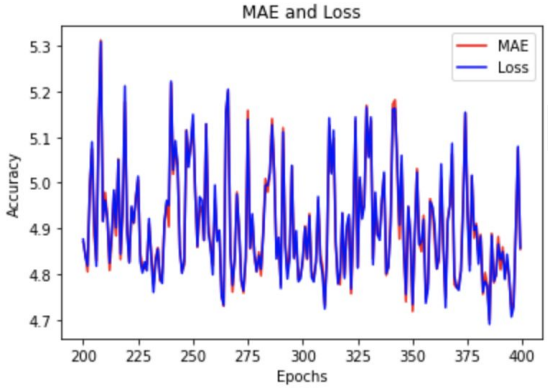
```
7.2298956
```

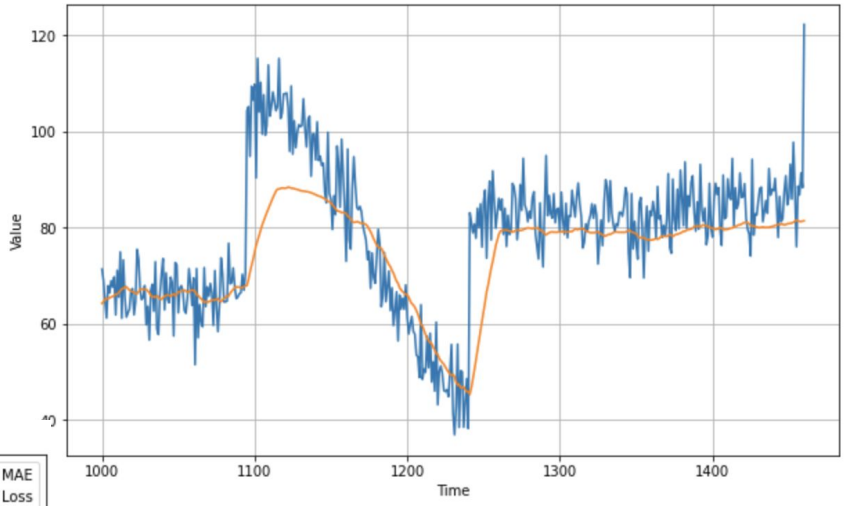
Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

Recurrent Neural Networks for Time Series

```
tf.keras.backend.clear_session()
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
  tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),
                      input_shape=[None]),
  tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32, return_sequences=True)),
  tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32, return_sequences=True)),
  tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 100.0)
])


model.compile(loss="mae", optimizer=tf.keras.optimizers.SGD(lr=1e-6, momentum=0.9))
model.fit(dataset,epochs=100)
```







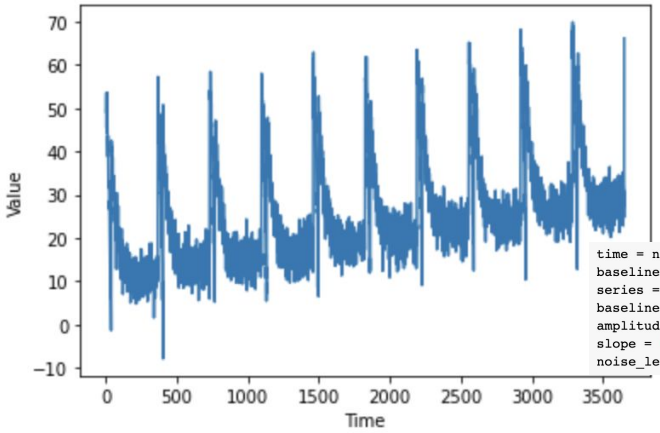

```
tf.keras.metrics.mean_absolute_error(x_valid, results).numpy()
```

```
7.212102
```

```
epochs_comment = """
Lambda, 2X Bi LSTM(32), 400 epochs
6.7509785
Lambda, 2X LSTM(32), 100 epochs
7.2298923
Lambda, 3X Bi LSTM(32), 100 epochs
7.242644
"""
```

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

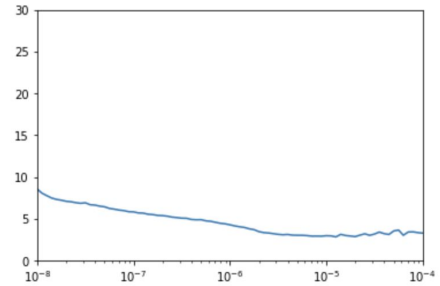
Recurrent Neural Networks for Time Series




```
time = np.arange(10 * 365 + 1,
baseline = 10
series = trend(time, 0.1)
baseline = 10
amplitude = 40
slope = 0.005
noise_level = 3
```

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

# FROM THIS PICK A LEARNING RATE
```

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

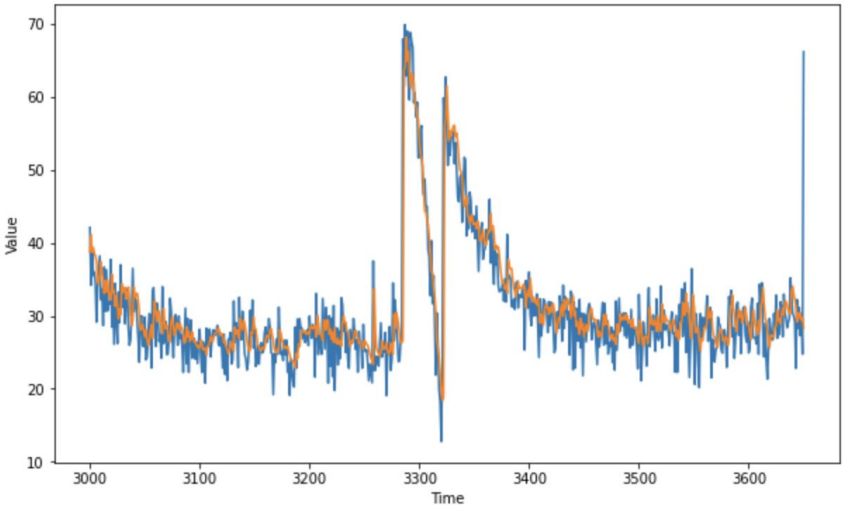



```
tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

tf.keras.backend.clear_session()
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
  tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1), input_shape=[None]),
  tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32, return_sequences=True)),
  tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 100.0)
])

lr_schedule = tf.keras.callbacks.LearningRateScheduler(lambda epoch: 1e-8 * 10**(epoch / 20))

optimizer = tf.keras.optimizers.SGD(lr=1e-8, momentum=0.9)

model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])

history = model.fit(dataset, epochs=100, callbacks=[lr_schedule])
```

```
model.compile(loss="mse", optimizer=tf.keras.optimizers.SGD(lr=1e-5, momentum=0.9),metrics=["mae"])
history = model.fit(dataset,epochs=400,verbose=1)
```

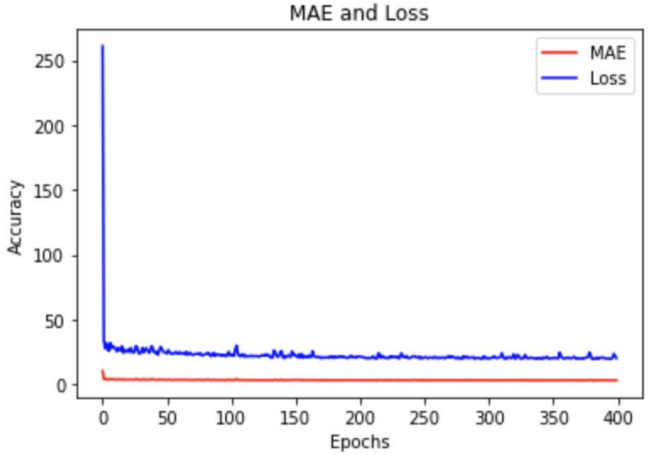
Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

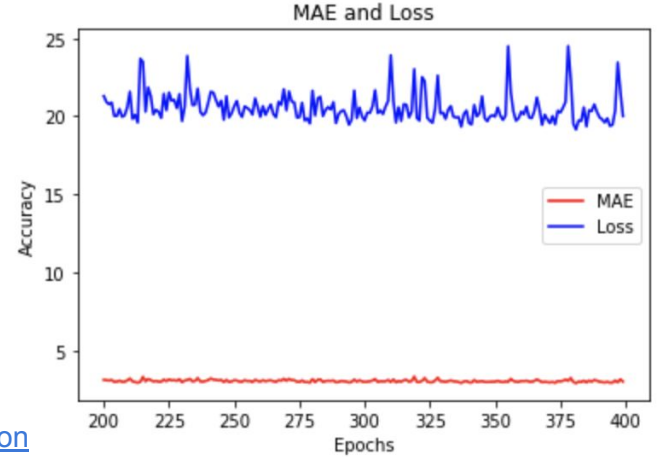
Recurrent Neural Networks for Time Series




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

# YOUR RESULT HERE SHOULD BE LESS THAN 4
```

```
3.051421
```



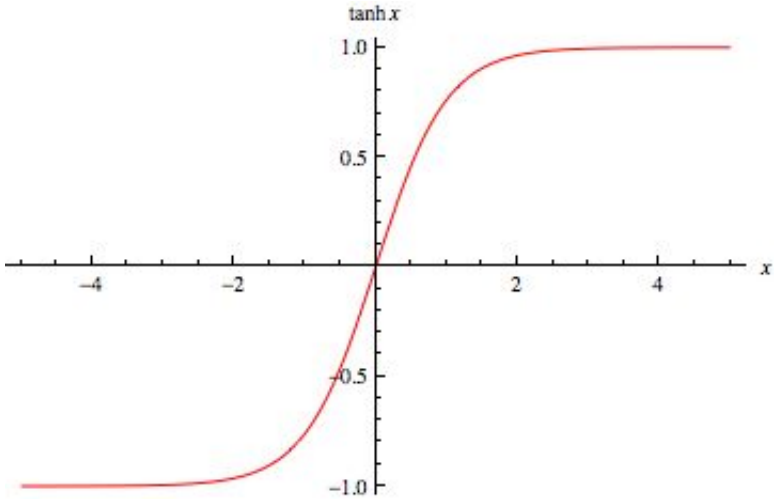


Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

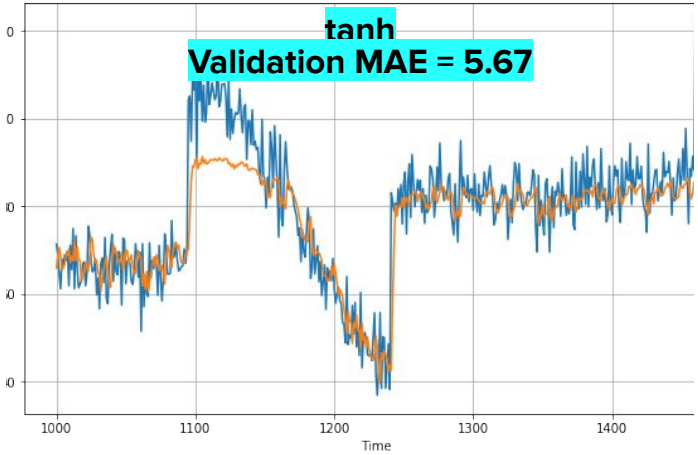
Recurrent Neural Networks for Time Series

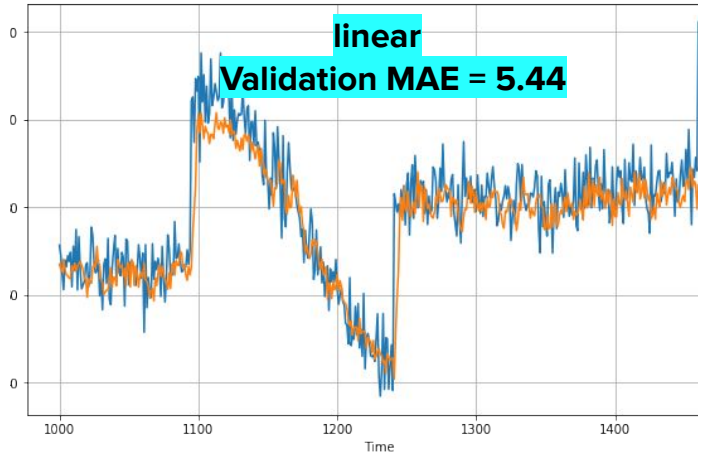
TensorFlow

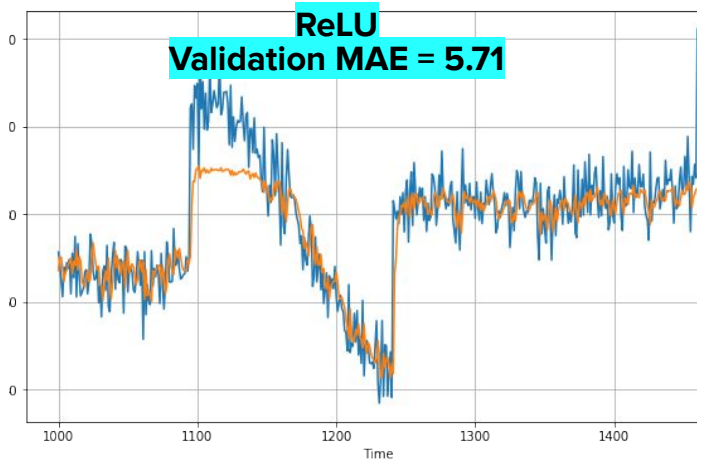
```
tf.keras.layers.SimpleRNN(
    units, activation='tanh', use_bias=True, kernel_initializer='glor
    recurrent_initializer='orthogonal', bias_initializer='zeros',
    kernel_regularizer=None, recurrent_regularizer=None, bias_regular
    activity_regularizer=None, kernel_constraint=None, recurrent_cons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

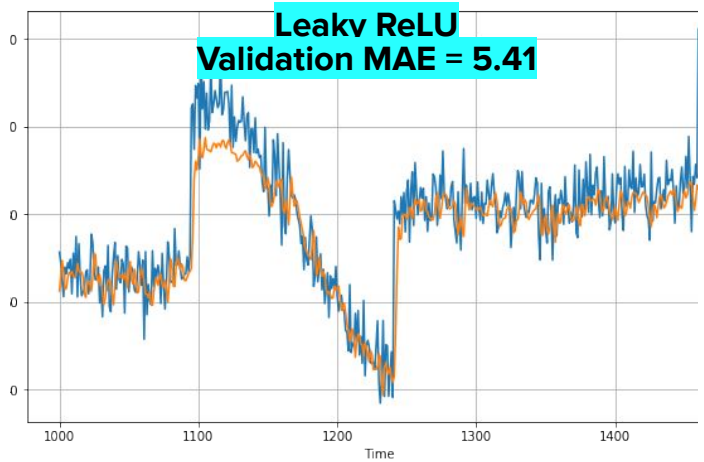


Recurrent Neural Networks for Time Series

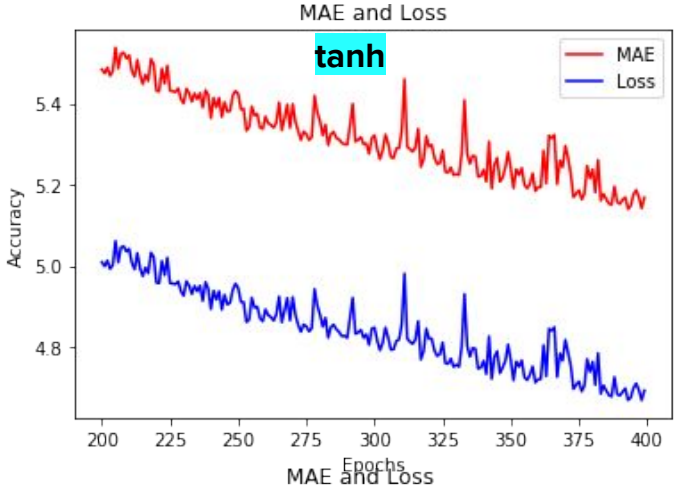


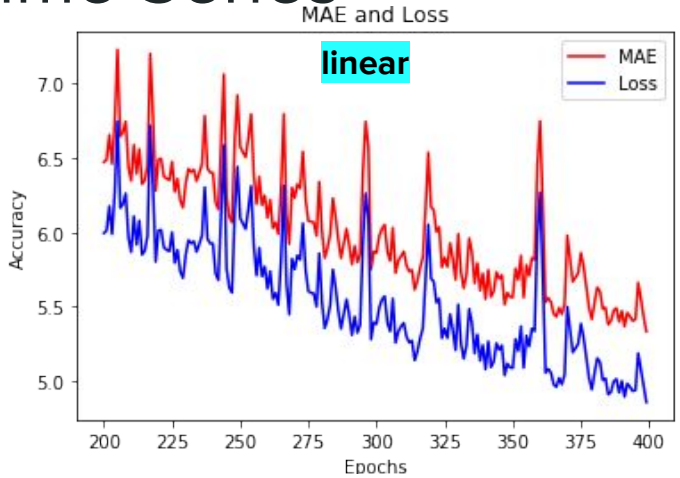


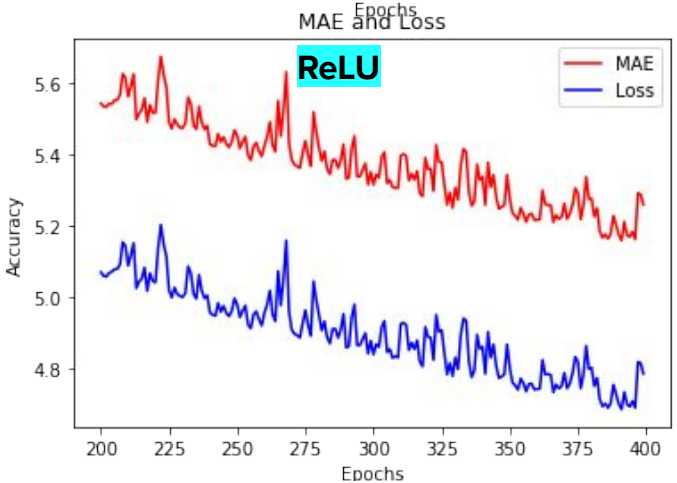


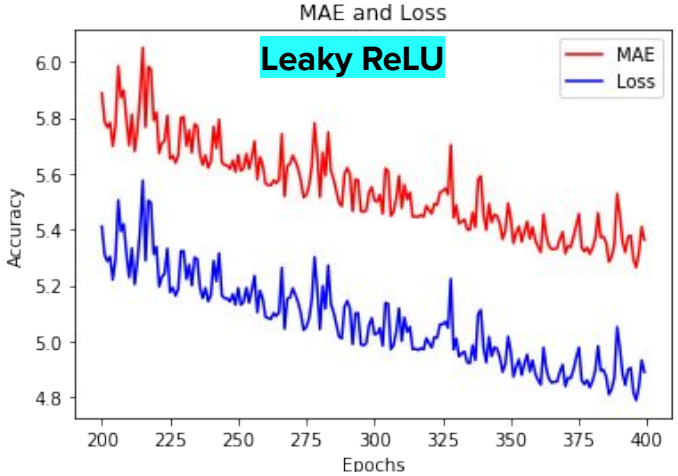


Recurrent Neural Networks for Time Series









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...

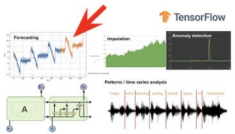
Online event

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...

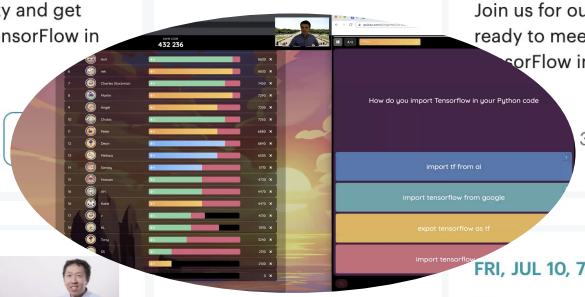


Online event

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...

9 attendees

1

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?...

14 attendees

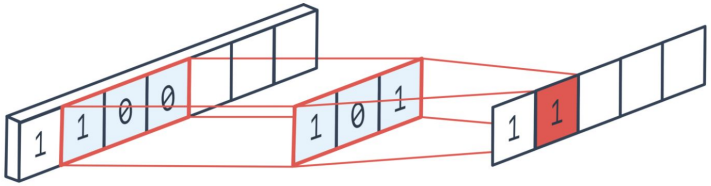
Attend

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

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

Real-world time series data



Conv1d-Input1d Example [Image [12]]

tf.keras.layers.Conv1D

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

```
tf.keras.layers.Conv1D(
    filters, kernel_size, strides=1, padding='valid', data_format='channels_last',
    dilation_rate=1, activation=None, use_bias=True,
    kernel_initializer='glorot_uniform', bias_initializer='zeros',
    kernel_regularizer=None, bias_regularizer=None, activity_regularizer=None,
    kernel_constraint=None, bias_constraint=None, **kwargs
)
```

```
>>> # The inputs are 128-length vectors with 10 timesteps, and the batch size
>>> # is 4.
>>> input_shape = (4, 10, 128)
>>> x = tf.random.normal(input_shape)
>>> y = tf.keras.layers.Conv1D(
... 32, 3, activation='relu',input_shape=input_shape)(x)
>>> print(y.shape)
(4, 8, 32)
```

`filters`	Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
`kernel_size`	An integer or tuple/list of a single integer, specifying the length of the 1D convolution window.
`strides`	An integer or tuple/list of a single integer, specifying the stride length of the convolution. Specifying any stride value != 1 is incompatible with specifying any `dilation_rate` value != 1.

Input shape:

3D tensor with shape: `(batch_size, steps, input_dim)`

Output shape:

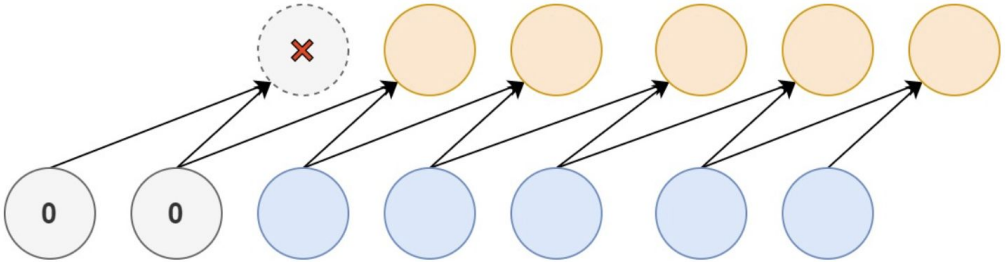
3D tensor with shape: `(batch_size, new_steps, filters)` `steps` value might have changed due to padding or strides.

Source: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv1D
https://towardsdatascience.com/pytorch-basics-how-to-train-your-neural-net-intro-to-cnn-26a14c2ea29

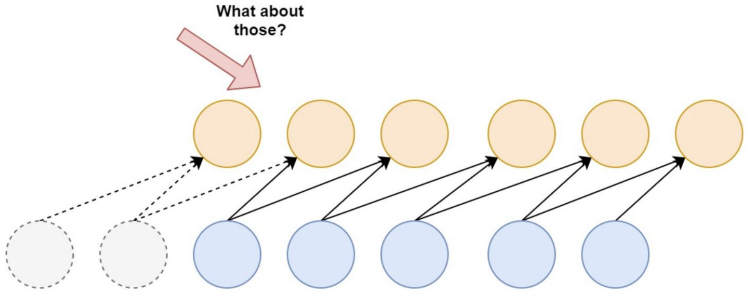
Real-world time series data

layer using "causal" padding: this ensures that the convolutional layer does not peek into the future when making predictions (it is equivalent to padding the inputs with the right amount of zeros on the left and using "valid" padding). We then add similar

Causal padding on the Conv1D layer allows you to include the partial information in your training process. By padding your input dataset with zeros at the front, a causal mapping to the first, missed-out targets can be made (Keras, n.d.; The Blog, n.d.). While the first target will be useless for training, the second can now be used based on the partial information that we have:



But what about the first two targets?



Although they are valid targets, the *inputs* are incomplete – that is, there is insufficient input data available in order to successfully use them in the training process (The Blog, n.d.). For the second target, *one* input – visible in gray – is missing (whereas the second is actually there), while for the first target both aren't there.

For the first target, there is no real hope for success (as we don't have any input at all and hence do not know which values produce the target value), but for the second, we have a partial picture: we've got half the inputs that produce the target.

Source: https://books.google.com/books?id=HnetDwAAQBAJ
https://www.machinecurve.com/index.php/2020/02/07/what-is-padding-in-a-neural-network/#causal-padding

Real-world time series d

Dilated and causal convolution

As discussed in the section on backtesting, we have to make sure that our model doe
bias:

As the convolu
ensures that th

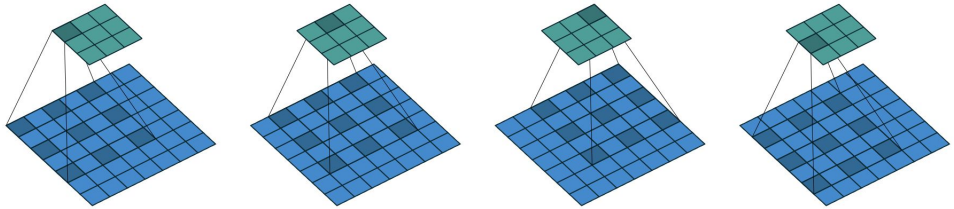
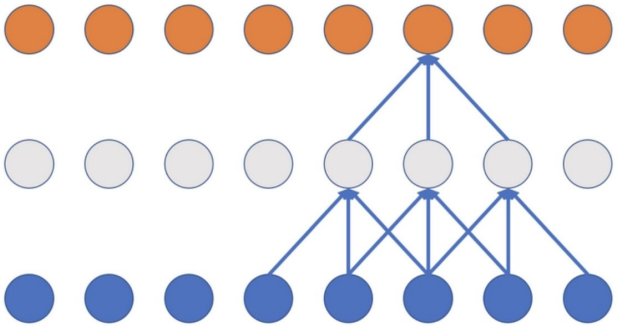


Figure 5.1: (Dilated convolution) Convolving a 3×3 kernel over a 7×7 input with a dilation factor of 2 (i.e., $i = 7$, $k = 3$, $d = 2$, $s = 1$ and $p = 0$).

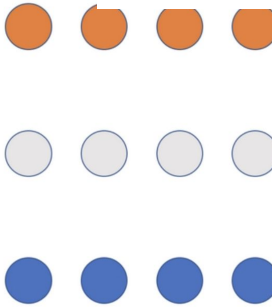
element, as we can see in the image below.

Standard Convolution



Standard convolution does not take the direction of convolution into account

As the convolutional filter slides over the data, it looks into the future as well as the p

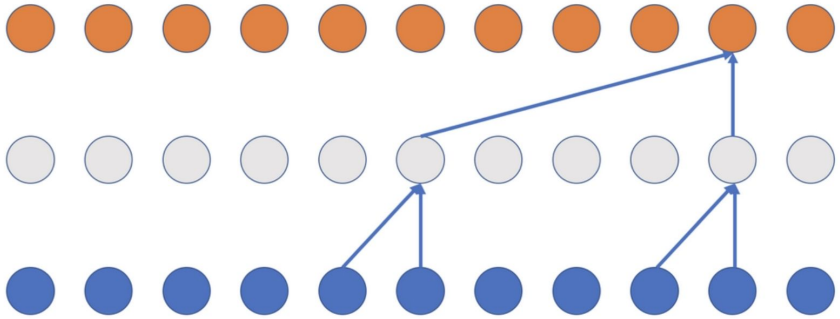


Causal convolution shifts the filter in th

In Keras, all we have to do is set the p
following code:

```
model.add(Conv1D(16,5, padding='ca
```

Causal Dilated Convolution



Dilated convolution skips over inputs while convolving

Source: https://subscription.packtpub.com/book/machine_learning/9781789136364/4/ch04lvl1sec59/dilated-and-causal-convolution
https://theblog.github.io/post/convolution-in-autoregressive-neural-networks
https://arxiv.org/pdf/1603.07285.pdf

Real-world time series data

```
time = np.arange(4 * 365 + 1, dtype="float32")
baseline = 10
series = trend(time, 0.1)
baseline = 10
amplitude = 40
slope = 0.05
noise_level = 5

# Create the series
series = baseline + trend(time, slope) + seasonality(time, period=365, amplitude=amplitude)
# 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
```

```
tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

window_size = 30
train_set = windowed_dataset(x_train, window_size, batch_size=128, shuffle_buffer=shuffle_buffer_size)

model = tf.keras.models.Sequential([
  tf.keras.layers.Conv1D(filters=32, kernel_size=5,
                         strides=1, padding="causal",
                         activation="relu",
                         input_shape=[None, 1]),
  tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32, return_sequences=True)),
  tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32, return_sequences=True)),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 200)
])
lr_schedule = tf.keras.callbacks.LearningRateScheduler(lambda epoch: 1e-8 * 10**(epoch / 20))

optimizer = tf.keras.optimizers.SGD(lr=1e-8, momentum=0.9)

model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])
```

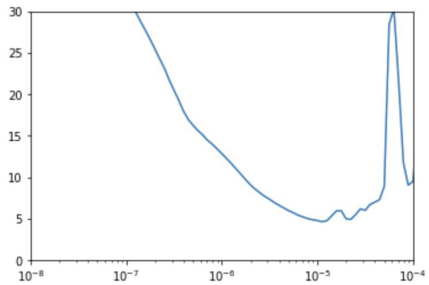
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, None, 32)          192
_________________________________________________________________
bidirectional (Bidirectional (None, None, 64)          16640
_________________________________________________________________
bidirectional_1 (Bidirection (None, None, 64)          24832
_________________________________________________________________
dense (Dense)                (None, None, 1)           65
_________________________________________________________________
lambda (Lambda)              (None, None, 1)           0
=================================================================
Total params: 41,729
Trainable params: 41,729
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(train_set, epochs=100, callbacks=[lr_schedule])
```

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

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



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

Real-world time series data

```
tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)
#batch_size = 16
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
  tf.keras.layers.Conv1D(filters=32, kernel_size=3,
                         strides=1, padding="causal",
                         activation="relu",
                         input_shape=[None, 1]),
  tf.keras.layers.LSTM(32, return_sequences=True),
  tf.keras.layers.LSTM(32, return_sequences=True),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 200)
])

optimizer = tf.keras.optimizers.SGD(lr=1e-5, momentum=0.9)

model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])
```

```
rnn_forecast = model_forecast(model, series[..., np.newaxis], window_size)
```

```
rnn_forecast.shape
```

```
(1432, 30, 1)
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, None, 32)          128
_________________________________________________________________
lstm (LSTM)                  (None, None, 32)          8320
_________________________________________________________________
lstm_1 (LSTM)                (None, None, 32)          8320
_________________________________________________________________
dense (Dense)                (None, None, 1)           33
_________________________________________________________________
lambda (Lambda)              (None, None, 1)           0
=================================================================
Total params: 16,801
Trainable params: 16,801
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(dataset,epochs=500)
```

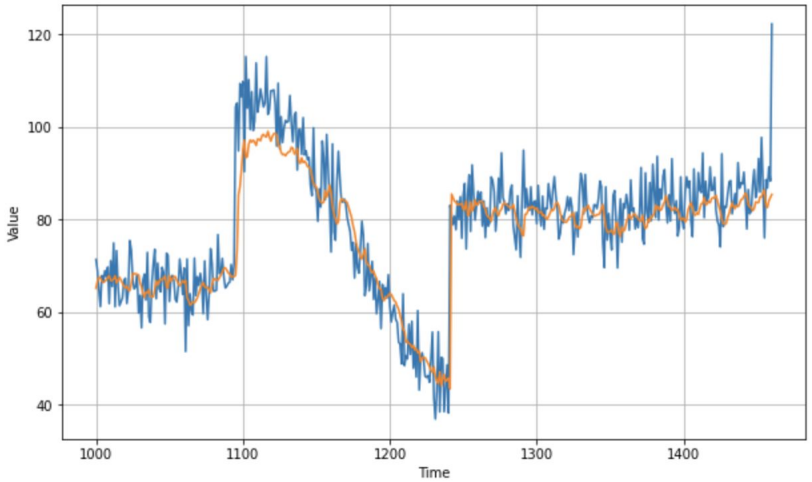
```
rnn_forecast = rnn_forecast[split_time - window_size:-1, -1, 0]
```

```
rnn_forecast.shape
```

```
(461,)
```

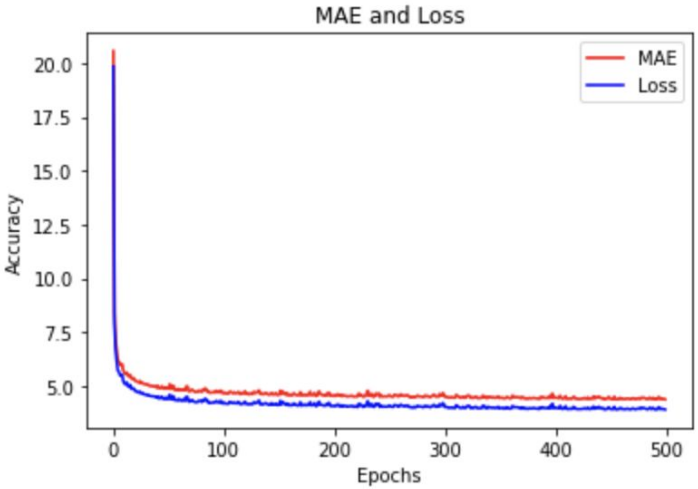
Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

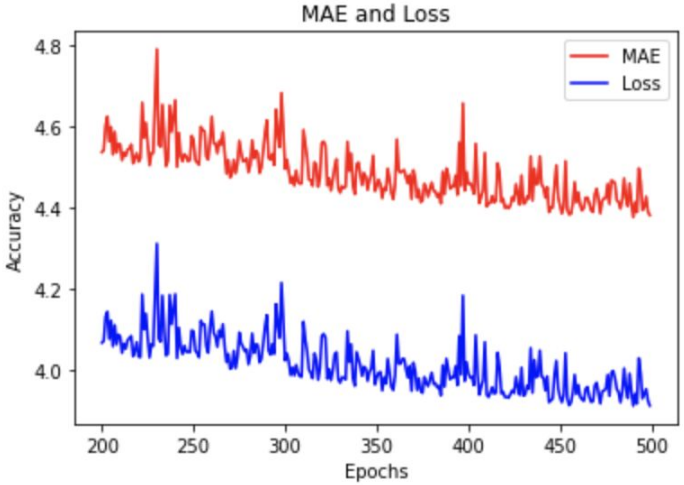
Real-world time series data




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

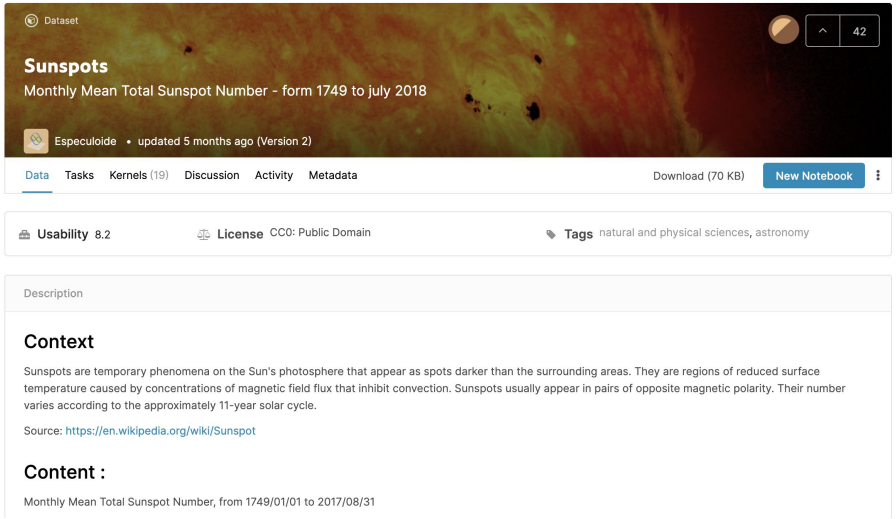
```
5.091009
```





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

Real-world time series data



Sunspots.csv (69.52 KB)

Detail Compact Column

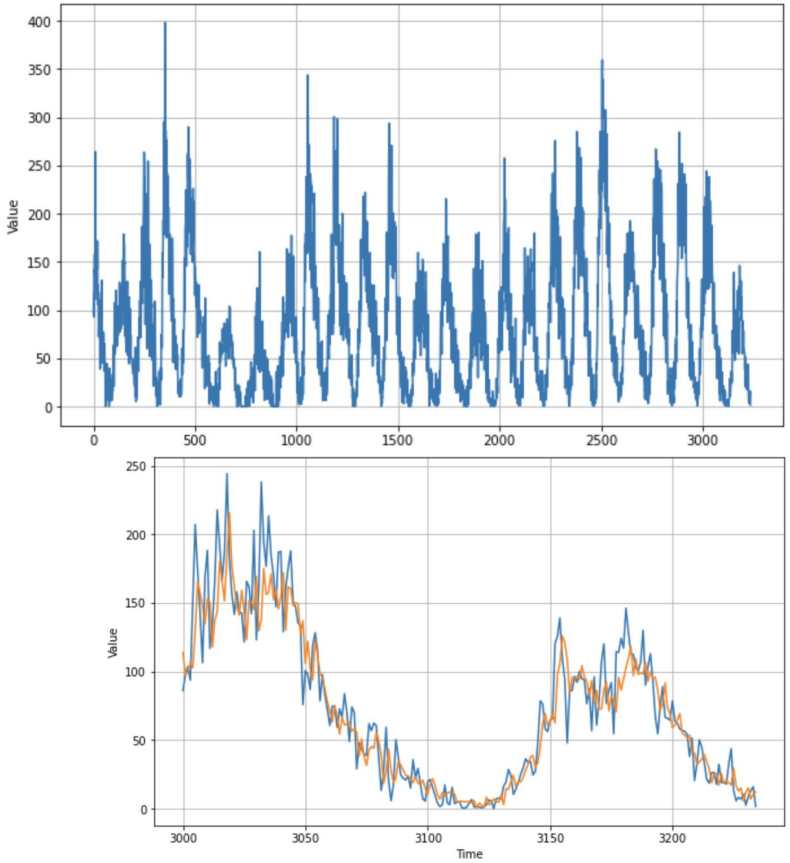
About this file

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
```

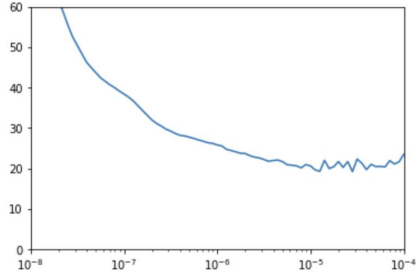
Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

Real-world time series data

```
tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)
window_size = 64
batch_size = 256
train_set = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)
print(train_set)
print(x_train.shape)

model = tf.keras.models.Sequential([
  tf.keras.layers.Conv1D(filters=32, kernel_size=5,
                         strides=1, padding="causal",
                         activation="relu",
                         input_shape=[None, 1]),
  tf.keras.layers.LSTM(64, return_sequences=True),
  tf.keras.layers.LSTM(64, return_sequences=True),
  tf.keras.layers.Dense(30, activation="relu"),
  tf.keras.layers.Dense(10, activation="relu"),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 400)
])

lr_schedule = tf.keras.callbacks.LearningRateScheduler(lambda epoch: 1e-8 * 10**(epoch / 20))

optimizer = tf.keras.optimizers.SGD(lr=1e-8, momentum=0.9)

model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])
```

```
plt.semilogx(history.history["lr"], history.history["loss"])
plt.axis([1e-8, 1e-4, 0, 60])
```

```
(1e-08, 0.0001, 0.0, 60.0)
```




```
model.summary()
```

```
<PrefetchDataset shapes: ((None, None, 1), (None, None, 1)), types:
(3000,)
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, None, 32)	192
lstm (LSTM)	(None, None, 64)	24832
lstm_1 (LSTM)	(None, None, 64)	33024
dense (Dense)	(None, None, 30)	1950
dense_1 (Dense)	(None, None, 10)	310
dense_2 (Dense)	(None, None, 1)	11
lambda (Lambda)	(None, None, 1)	0

```
Total params: 60,319
Trainable params: 60,319
Non-trainable params: 0
```

```
history = model.fit(train_set, epochs=100, callbacks=[lr_schedule])
```

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

Real-world time series data

```
tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)
train_set = windowed_dataset(x_train, window_size=60, batch_size=100, shuffle_buffer=shuffle_buffer_size)
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv1D(filters=60, kernel_size=5,
                      strides=1, padding="causal",
                      activation="relu",
                      input_shape=[None, 1]),
  tf.keras.layers.LSTM(60, return_sequences=True),
  tf.keras.layers.LSTM(60, return_sequences=True),
  tf.keras.layers.Dense(30, activation="relu"),
  tf.keras.layers.Dense(10, activation="relu"),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 400)
])


optimizer = tf.keras.optimizers.SGD(lr=1e-5, momentum=0.9)

model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])
```

```
rnn_forecast = model_forecast(model, series[..., np.newaxis], window_size)
rnn_forecast = rnn_forecast[split_time - window_size:-1, -1, 0]
```

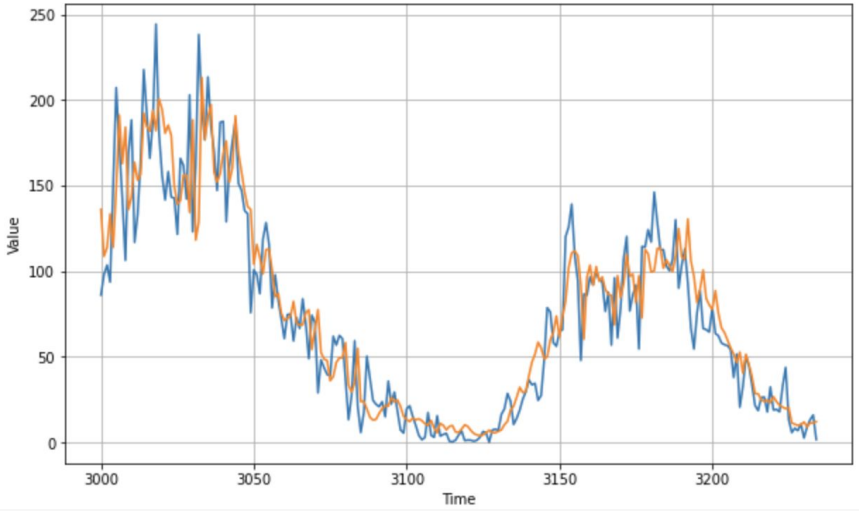
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, None, 60)          360
_________________________________________________________________
lstm (LSTM)                  (None, None, 60)          29040
_________________________________________________________________
lstm_1 (LSTM)                (None, None, 60)          29040
_________________________________________________________________
dense (Dense)                (None, None, 30)          1830
_________________________________________________________________
dense_1 (Dense)              (None, None, 10)          310
_________________________________________________________________
dense_2 (Dense)              (None, None, 1)           11
_________________________________________________________________
lambda (Lambda)              (None, None, 1)           0
=================================================================
Total params: 60,591
Trainable params: 60,591
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(train_set,epochs=500)
```

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

Real-world time series data

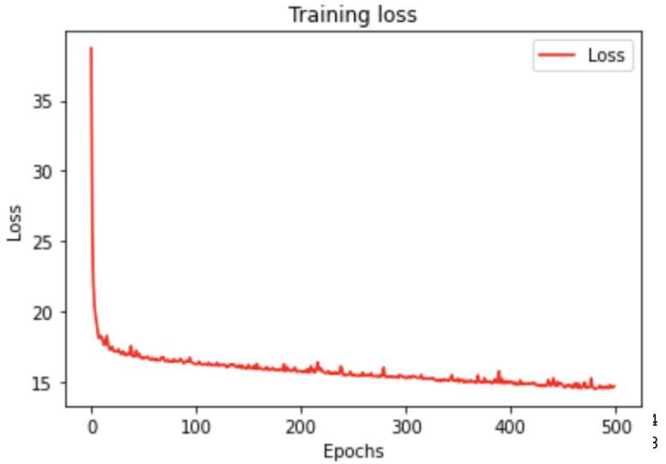


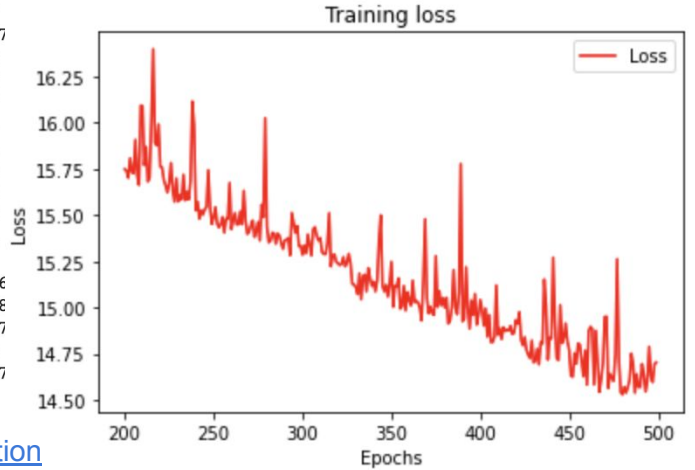

```
tf.keras.metrics.mean_absolute_error(x_valid, rnn_forecast).numpy()
```

```
15.171835
```

```
print(rnn_forecast)
```

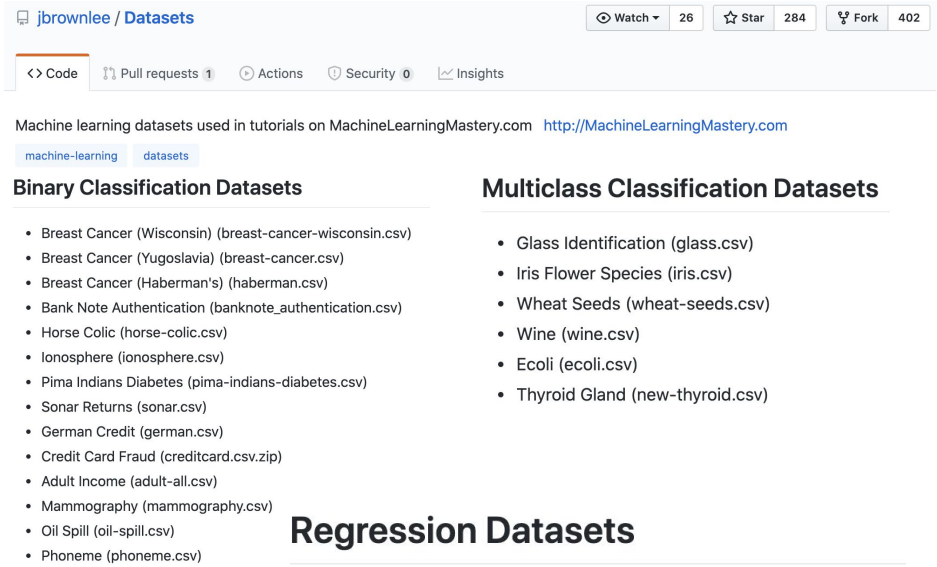
```
[136.03911   108.64892   113.79592   133.36797   113.831924  148.42674
 191.09799   162.74585   184.07158   135.99425   143.39186   163.61166
 153.16757
 182.03194
 149.59795
 188.31035
 197.25131
 151.96443
 137.62968
 112.5301
  71.117546
  68.58975
  52.379192
  49.27989
  54.85332
  13.381341
  23.775818
  14.10166
  12.787093
   9.287236
   9.28528
   5.521899
  10.305211
  29.03817
  55.202557
  61.49422
 108.99533
 102.60377
  68.64034
  98.18201
  99.39812
 103.24613
 130.41716
  83.96074
  64.0178
  40.443546
  24.747498
  21.727987
   9.88898
  12.069787
```





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

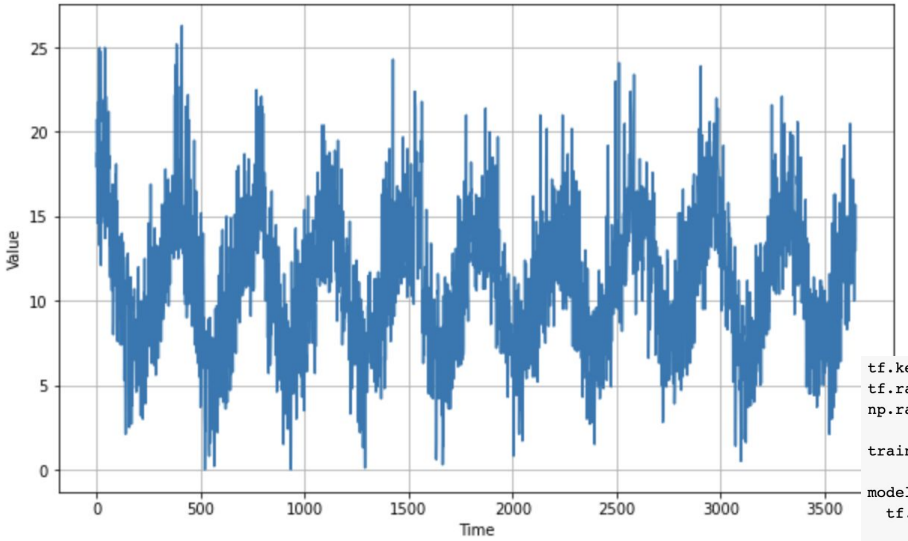
- Daily Minimum Temperatures in Melbourne (daily-min-temperatures.csv)
- Daily Maximum Temperatures in Melbourne (daily-max-temperatures.csv)
- Daily Female Births in California (daily-total-female-births.csv)
- Monthly International Airline Passengers (monthly-airline-passengers.csv)
- Monthly Armed 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

Real-world time series data



Branch: master ▾ Datasets / daily-min-temperatures.csv Find file Copy path

Jason Brownlee Added more time series datastes. 5224989 on Mar 10, 2018

0 contributors

3651 lines (3651 sloc) 66.3 KB Raw Blame History

Search this file...

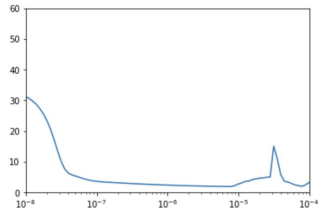
	Date	Temp
1	1981-01-01	20.7
2	1981-01-02	17.9
3	1981-01-03	18.8
4	1981-01-04	14.6
5	1981-01-05	15.8
6	1981-01-06	15.8
7	1981-01-07	15.8

```
split_time = 2500
time_train = time[:split_time]
x_train = series[:split_time]
time_valid = time[split_time:]
x_valid = series[split_time:]

window_size = 30
batch_size = 32
shuffle_buffer_size = 1000
```

```
tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

train_set = windowed_dataset(x_train, window_size=60, batch_size=100, shuffle_buffer=shuffle_buffer_size)

model = tf.keras.models.Sequential([
  tf.keras.layers.Conv1D(filters=32, kernel_size=5,
                         strides=1, padding="causal",
                         activation="relu",
                         input_shape=[None, 1]),
  tf.keras.layers.LSTM(64, return_sequences=True),
  tf.keras.layers.LSTM(64, return_sequences=True),
  tf.keras.layers.Dense(30, activation="relu"),
  tf.keras.layers.Dense(10, activation="relu"),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 400)
])

optimizer = tf.keras.optimizers.SGD(lr=1e-5, momentum=0.9)

model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])
```

```
plt.semilogx(history.history["lr"], history.history["loss"])
plt.axis([1e-8, 1e-4, 0, 60])
```

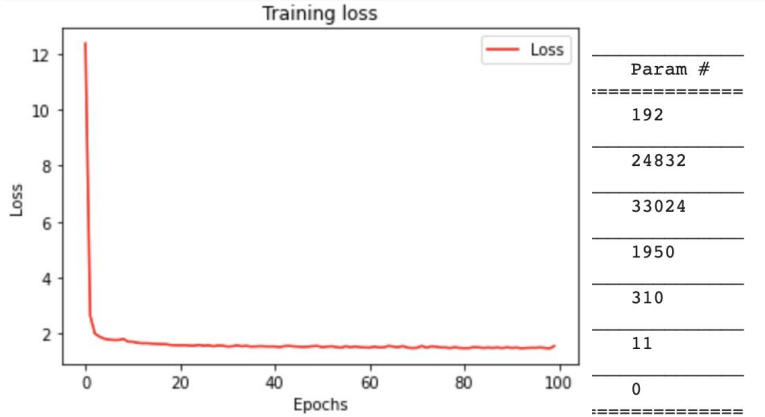
```
(1e-08, 0.0001, 0.0, 60.0)
```



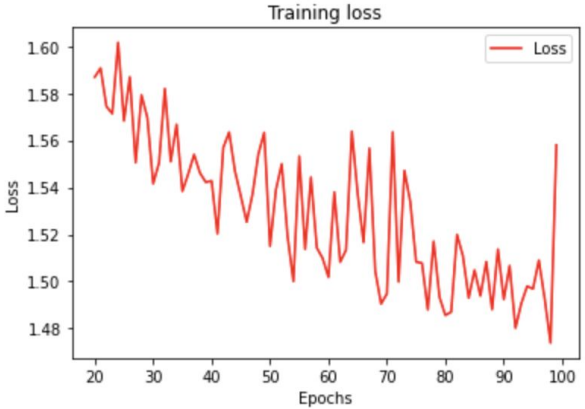
Source: https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction

Real-world time series data

```
model.summary()
```

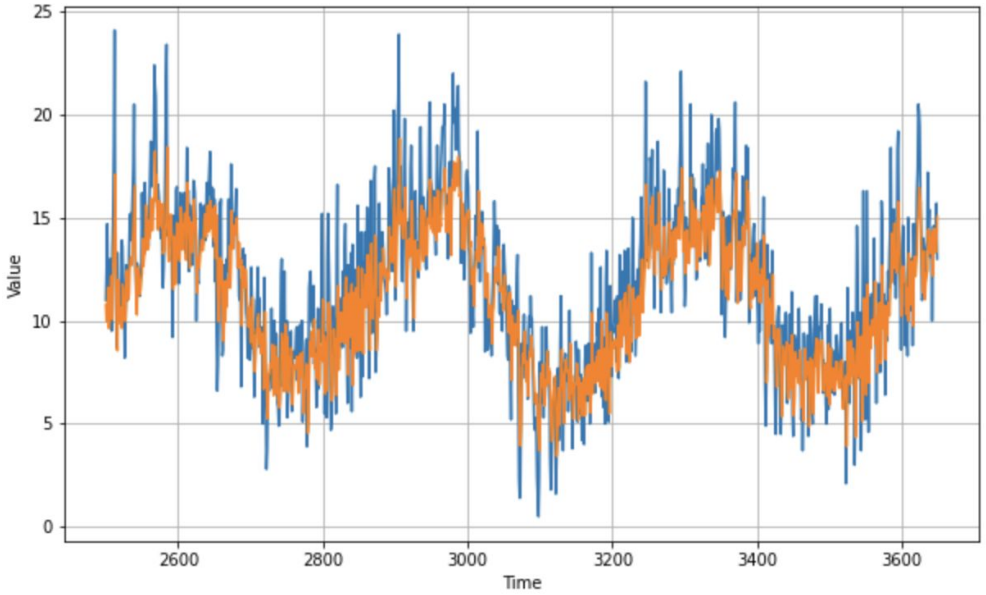



```
_________________________________________________________________
Param #
=================================================================
192
_________________________________________________________________
24832
_________________________________________________________________
33024
_________________________________________________________________
1950
_________________________________________________________________
310
_________________________________________________________________
11
_________________________________________________________________
0
=================================================================
```




```
OUT 30 EPOCHS
```

```
rnn_forecast = model_forecast(model, series[..., np.newaxis], window_size)
rnn_forecast = rnn_forecast[split_time - window_size:-1, -1, 0]
```




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

# EXPECTED OUTPUT MAE < 2 -- I GOT 1.789626
```

```
1.8799866
```

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

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

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...

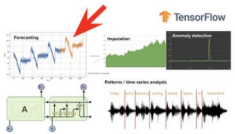
Online event

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...

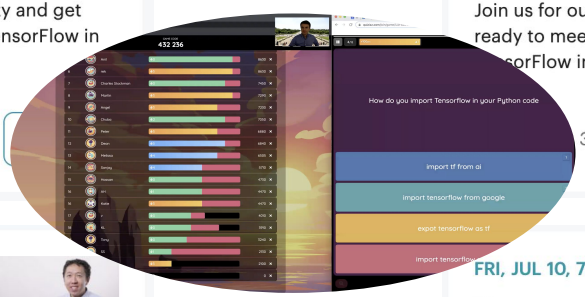


Online event

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...

9 attendees

1

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?...

14 attendees

Attend

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

Check out these resources

A DISCIPLINED APPROACH TO NEURAL NETWORK HYPER-PARAMETERS: PART 1 – LEARNING RATE, BATCH SIZE, MOMENTUM, AND WEIGHT DECAY

Leslie N. Smith
US Naval Research Laboratory
Washington, DC, USA
leslie.smith@nrl.navy.mil

Interested Add to Calendar Share

Tue Jun 16, 9:00 AM - 12:00 PM EDT (3 Hours)

Training Session: Beyond AutoPilot

How to level up your modeling after running Autopilot. Learning about new techniques, algorithms, and features DataRobot makes available to improve your models.

MOFC
Learn. Forecast. Compete. Disseminate. Excel.

ABOUT M COMPETITIONS M CONFERENCES M PUBLICATIONS M COURSES CONTACT

Home / Dataset

The Dataset

The M5 consists of 10,000 time series of Yearly, Quarterly, Monthlyand Other (Weekly, Daily and Hourly) data.

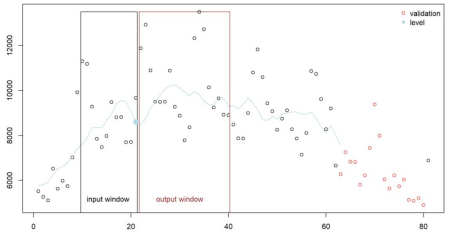
The minimum number of observations is 13 for yearly, 16 for quarterly, 42 for monthly, 80 for weekly, 93 for daily and 700 for hourly series.

The 10,000 time series of the dataset come mainly from the Economic, Finance, Demographics and Industry areas, while also including data from Tourism, Trade, Labor and Wage, Real Estate, Transportation, Natural Resources and the Environment.

The M5 Competition series, as those of the M1 and M3, aim at representing the real world as much as possible. The series were selected randomly from a database of 900,000 ones on December 28, 2017. Professor Makridakis chose the seed number for generating the random sample that determined the M4 Competition data. Some pre-defined filters were applied beforehand to achieve some desired characteristics, such as the length of the series, the percentage of Yearly, Quarterly, Monthly, Weekly, Daily, and Hourly data, as well as their type (Micro, Macro, Finance, Industry, Demographic, Other).

M4 Forecasting Competition: Introducing a New Hybrid ES-RNN Model

Slawek Smyl, Jai Ranganathan, and Andrea Pasqua June 25, 2018



Sign up for Uber Engineering updates:

Your email address

Subscribe

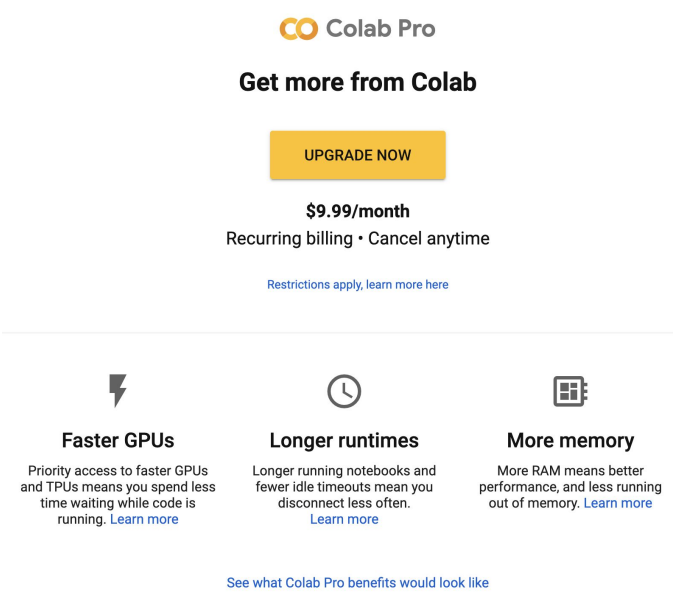
Popular Articles

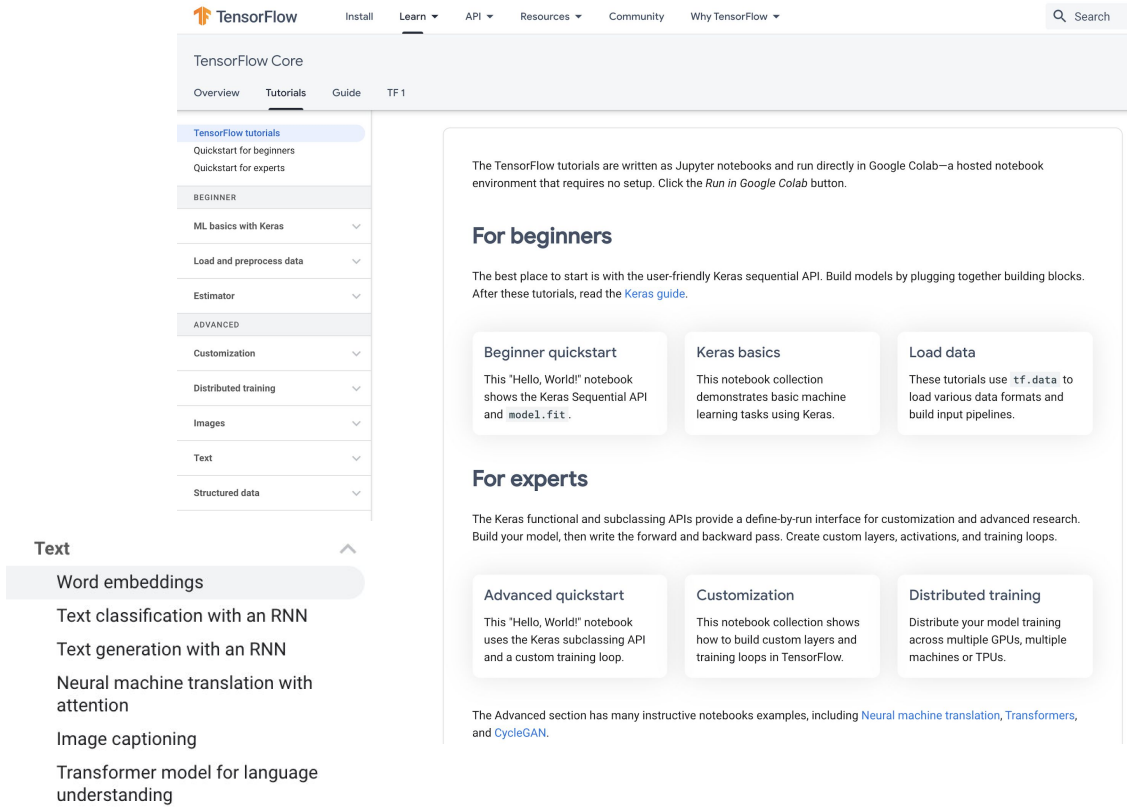
Meet Michelangelo: Uber's Machine Learning Platform
September 5, 2017

Uber's Big Data Platform: 100+ Petabytes with Minute Latency
October 17, 2018

Source: https://arxiv.org/pdf/1803.09820.pdf
https://aiworldwide.datarobot.com/agenda/session/259782
https://mofc.unic.ac.cy/the-dataset/
https://eng.uber.com/m4-forecasting-competition/

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

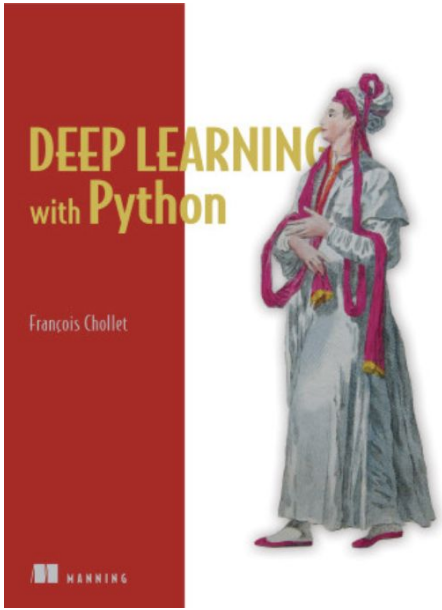
TensorFlow Core

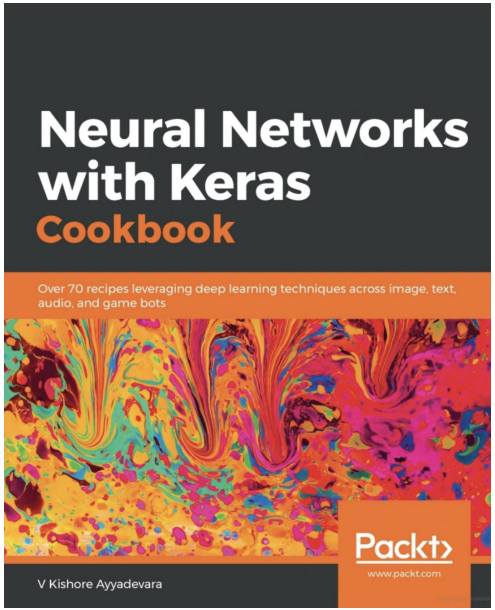
Introducing the TensorFlow Developer Certificate!

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...

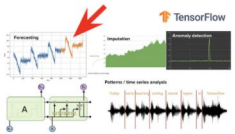
Online event

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...

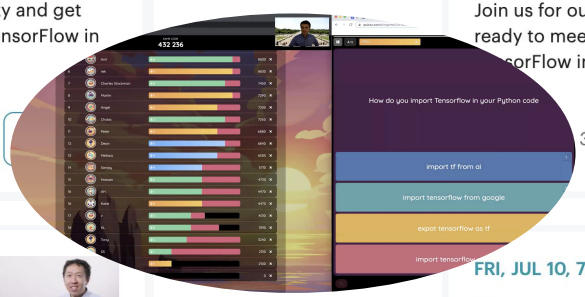


Online event

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...

9 attendees

1

Attend

FRI, JUL 10, 7:30 PM EDT

How to prepare for and pass the TensorFlow Developer Certificate 🎉

Online event

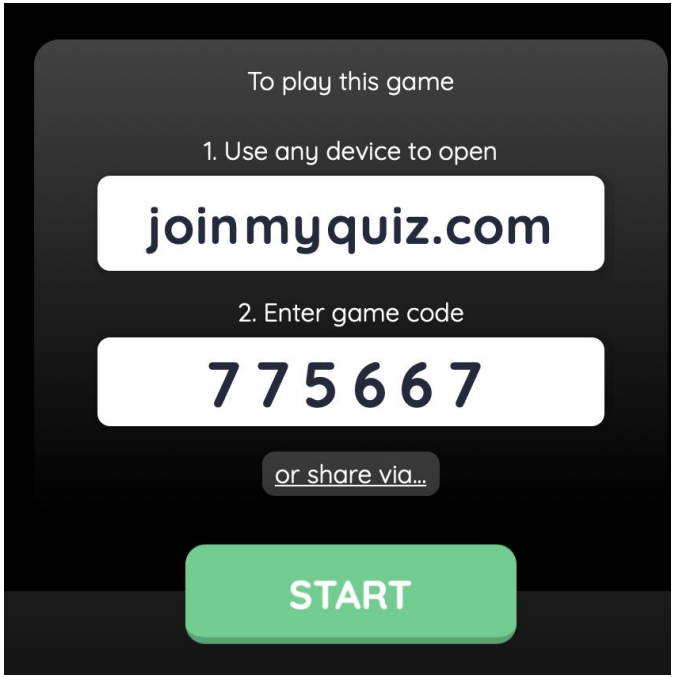
Join us for our 11th adventure in Deep Learning! Just bring your curiosity and get ready to meet our growing community 😀 Join Zoom Meeting: https://us02web.zoom.us/j/84402592502?...

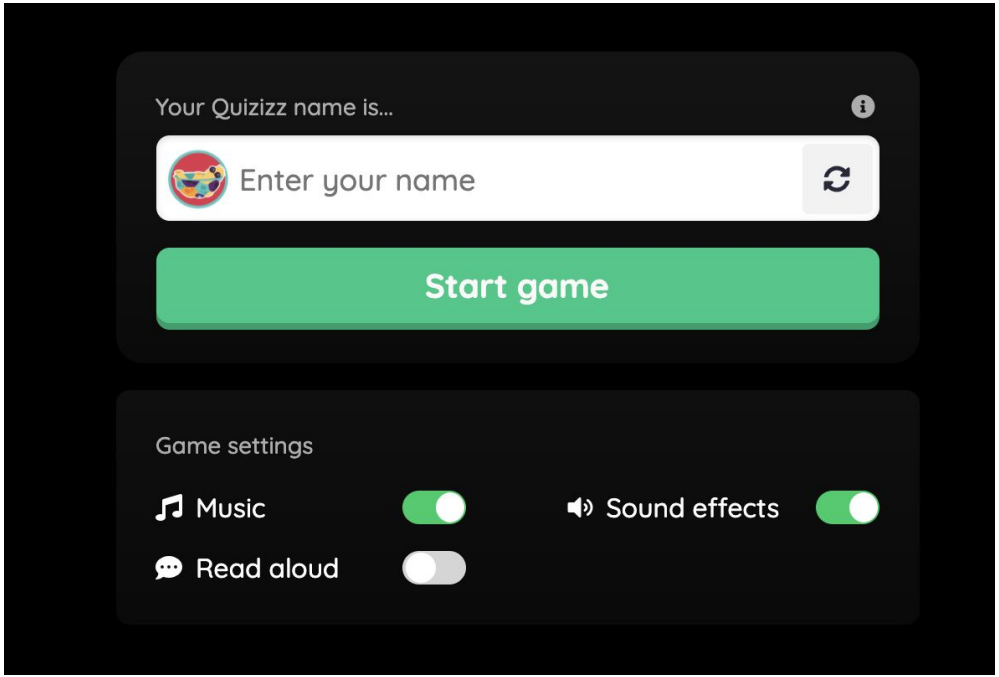
14 attendees

Attend

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

Time for a fun game 😀🎉

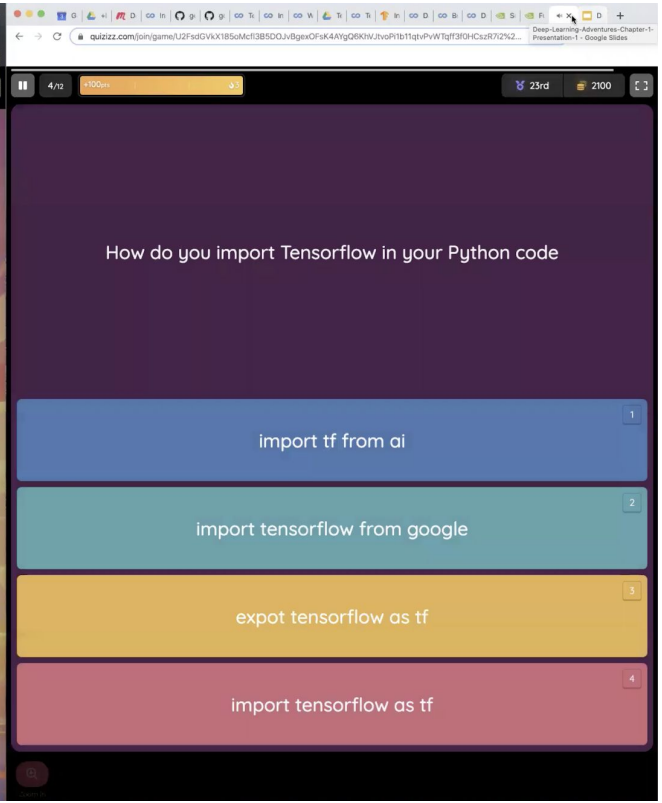




Practice here or use Flashcards:
https://quizizz.com/join/quiz/5eefd13d3db4b8001bb6d72b/start?from=soloLinkShare&referrer=5d921444d0fa99001a135336

Time for a fun game 😀🎉

Practice here or use Flashcards:
https://quizizz.com/join/quiz/5eefd13d3db4b8001bb6d72b/start?from=soloLinkShare&referrer=5d921444d0fa99001a135336



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

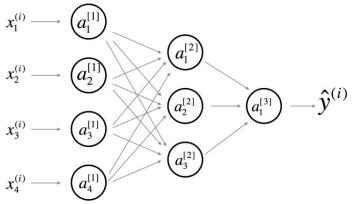


Figure 1: Comprehensive Network: representation commonly used for Neural Networks. For better aesthetic, we omitted the details on the parameters ($w_{ij}^{[l]}$ and $b_i^{[l]}$ etc...) that should appear on the edges

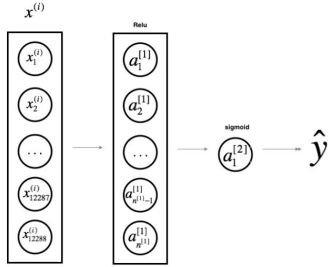


Figure 2: Simplified Network: a simpler representation of a two layer neural network, both are equivalent.