Fall 10-16-2020

Flight Simulator Modeling Using Recurrent Neural Networks

Nickolas Sabatini
Portland State University

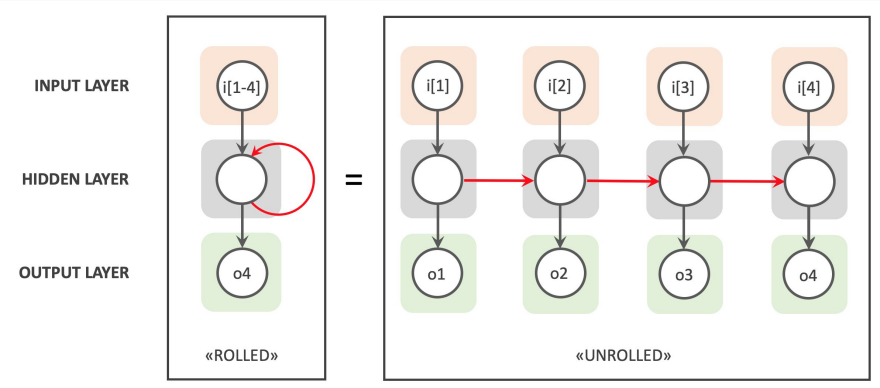
Andreas Natsis
Portland State University

Recurrent Neural Networks

Nick Sabatini

What are recurrent neural networks?

-Recurrent Neural Networks: Using past data values to predict future values



Flight Simulator

-Predict air pressure values

-Tensorflow and Keras

-LSTM: Predicting sequence values

https://www.curiousily.com/posts/demand-prediction-with-lstms-using-tensorflow-2-and-keras-in-python/

Flight Simulator

Tensorflow is a Python library used for machine learning and neural networks

Keras is a Python library for neural networks

Flight Simulator

-Training vs. validation loss

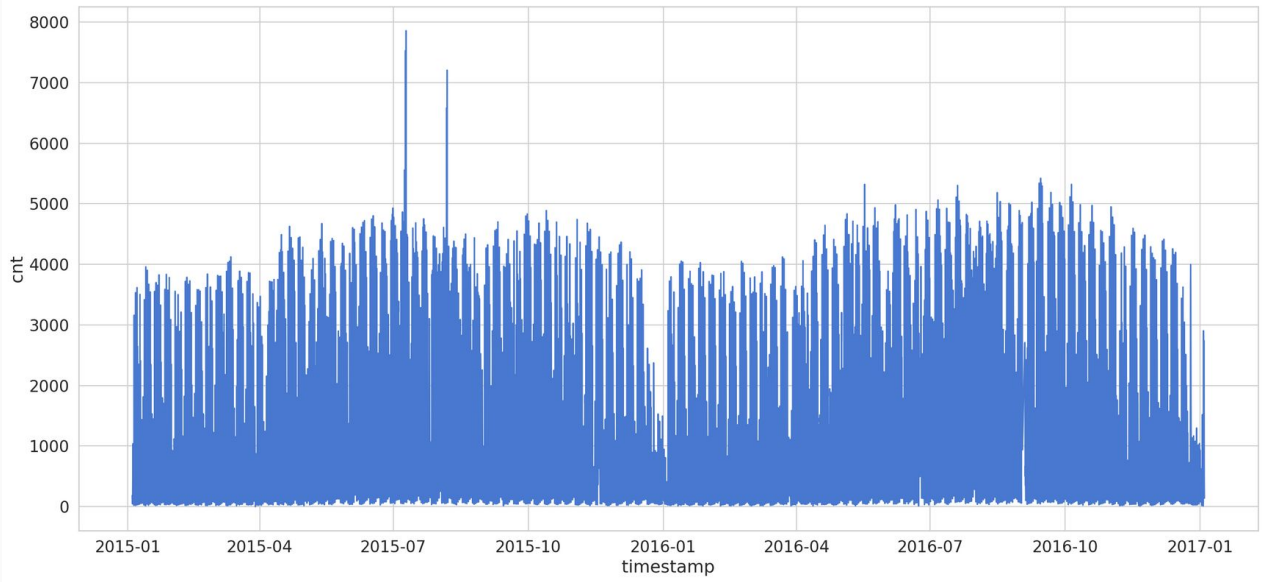
-Training loss: error when training the model

-Validation loss: error after training

-Overfitting

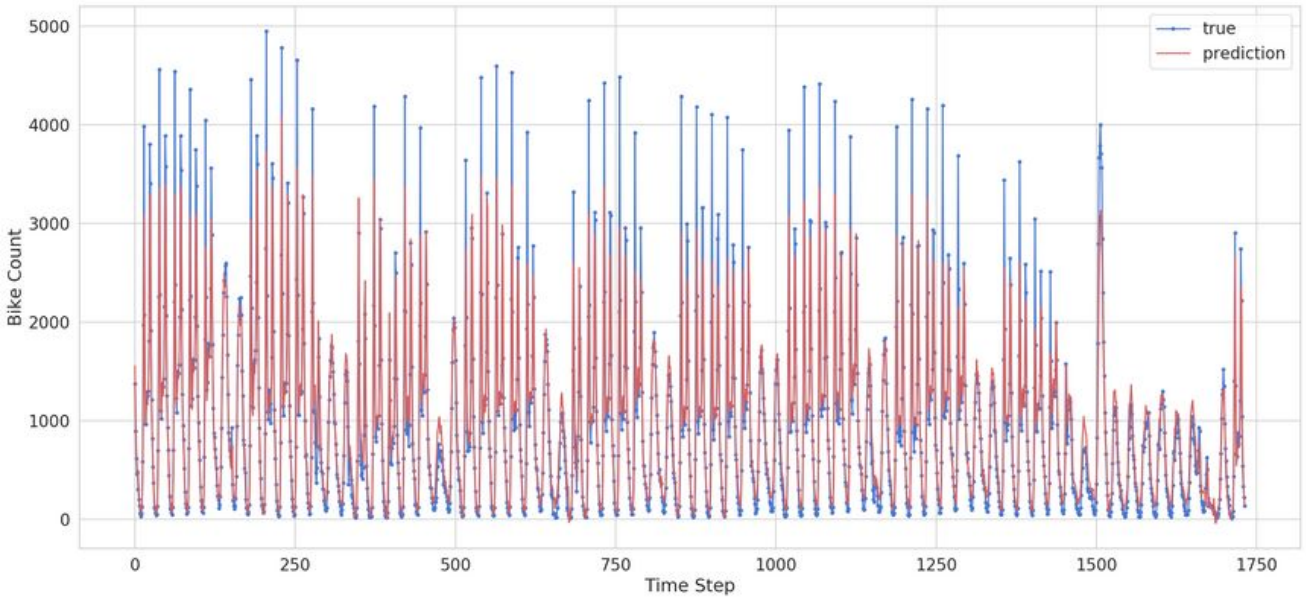
Time Series Forecasting

Bike sharing in London

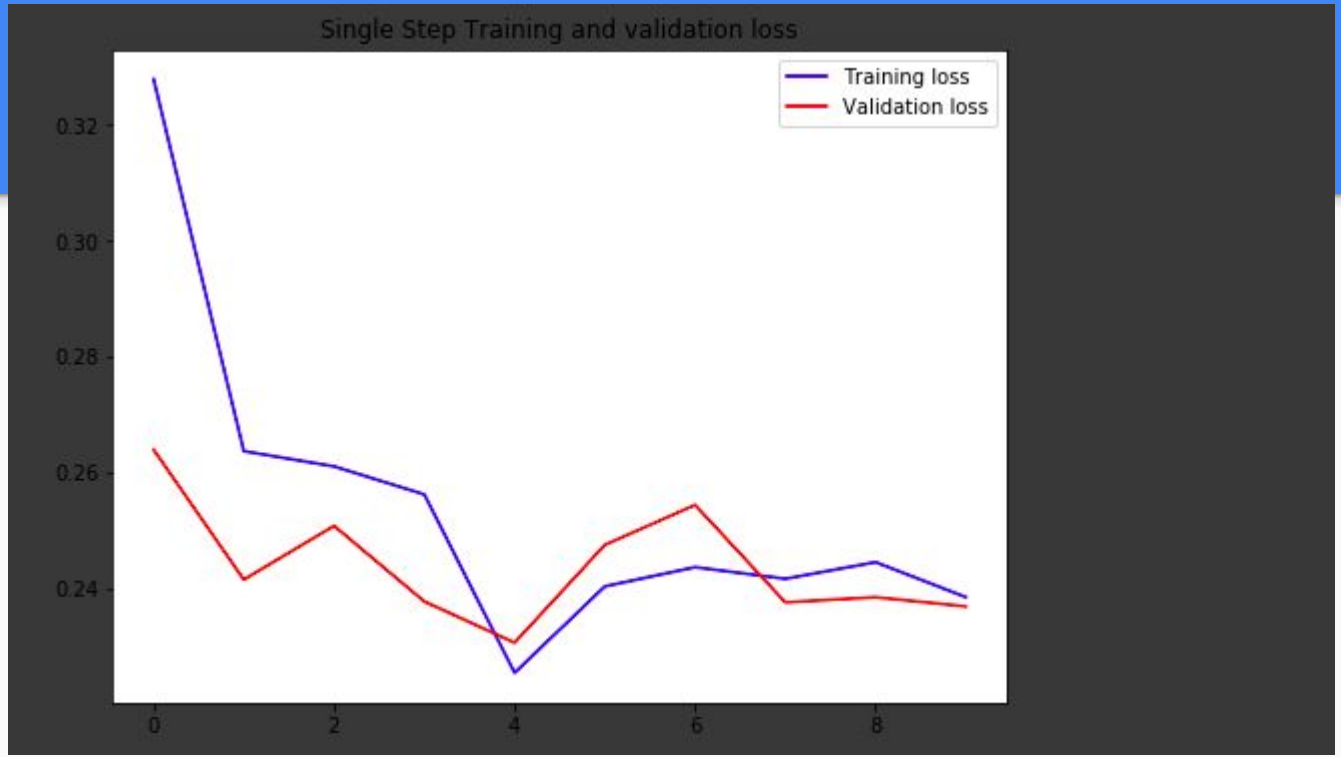


Time Series Forecasting

Bike sharing in London



```
Single window of past history : (120, 3)
Train on 200 steps, validate on 50 steps
Epoch 1/10
200/200 [==============================] - 28s 140ms/step - loss: 0.3278 - val_loss: 0.2639
Epoch 2/10
200/200 [==============================] - 27s 133ms/step - loss: 0.2636 - val_loss: 0.2414
Epoch 3/10
200/200 [==============================] - 27s 135ms/step - loss: 0.2610 - val_loss: 0.2507
Epoch 4/10
200/200 [==============================] - 26s 132ms/step - loss: 0.2561 - val_loss: 0.2376
Epoch 5/10
200/200 [==============================] - 26s 131ms/step - loss: 0.2253 - val_loss: 0.2305
Epoch 6/10
200/200 [==============================] - 26s 129ms/step - loss: 0.2403 - val_loss: 0.2474
Epoch 7/10
200/200 [==============================] - 27s 133ms/step - loss: 0.2436 - val_loss: 0.2543
Epoch 8/10
200/200 [==============================] - 27s 133ms/step - loss: 0.2416 - val_loss: 0.2375
Epoch 9/10
200/200 [==============================] - 27s 133ms/step - loss: 0.2444 - val_loss: 0.2384
Epoch 10/10
200/200 [==============================] - 26s 131ms/step - loss: 0.2384 - val_loss: 0.2368
```


Single Step Training and validation loss
Training loss
Validation loss
0.32
0.30
0.28
0.26
0.24
0
2
4
6
8

[7]

```
# Scikit-Learn ≥0.20 is required
import sklearn
#assert sklearn.__version__ >= "0.20"
#!pip install tensorflow
try:
    # %tensorflow_version only exists in Colab.
    %tensorflow_version 2.x
    IS_COLAB = True
except Exception:
    IS_COLAB = False

# TensorFlow ≥2.0 is required
import tensorflow as tf
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession

config = ConfigProto()
config.gpu_options.allow_growth = True
session = InteractiveSession(config=config)
from tensorflow import keras
#assert tf.__version__ >= "2.0"

if not tf.test.is_gpu_available():
    print("No GPU was detected. LSTMs and CNNs can be very slow without a GPU.")
    if IS_COLAB:
        print("Go to Runtime > Change runtime and select a GPU hardware accelerator.")

# Common imports
import numpy as np
import os
```

```
import pandas as pd
from google.colab import files
uploaded = files.upload()

data = pd.read_csv("data_sampled_10.csv", sep=",")
data.describe()
print(data.shape)
```

```
arr = data.to_numpy() #convert to numpy array
arr = arr[:,1:]  #I drop time from the dataset
print(arr.shape)
```

```
(98703, 21)
```

```
#!pip install -U scikit-learn
from pandas import Series
from sklearn.preprocessing import StandardScaler
from math import sqrt
scaler = StandardScaler()
scaler = scaler.fit(arr)
print('Mean: %f, StandardDeviation: %f' % (scaler.mean_[0], sqrt(scaler
# normalize the dataset and print
standardized = scaler.transform(arr)
print(standardized)
```

```
Mean: -50.588483, StandardDeviation: 290.559608
[[ 0.1568989   1.80215733 -1.65141996 ...  0.05406893  0.00454944
  -0.08534238]
 [ 0.1568989   1.80215733 -1.65141996 ...  0.0061874  -0.01504004
  -0.11624133]
 [ 0.1568989   1.80215733 -1.65141996 ... -0.01535929 -0.00355655
  -0.03899395]
 ...
 [ 0.14313236  1.57632452 -1.65141996 ...  0.01336963 -0.0092983
  -0.0698929 ]
 [ 0.14313236  1.80215733 -1.86305921 ...  0.02055186 -0.00220555
   0.1541245 ]
 [ 0.14313236  1.57632452 -1.65141996 ...  0.02534001  0.00387394
   0.05370291]]
```

```
# split into samples (e.g. 5000/200 = 25)

X_train = np.empty((97703,500,21))  # create an empty 3d array
n=98703-999

Y_train = np.empty(n)
length = 500  # length of every sample

for i in range(0,n-1,1):

  X_train[i,:,:] = standardized[i:i+length,:]

print(X_train.shape)


for i in range(0,n-2,1):

  Y_train[i] = standardized[i+length+20,0]

print(Y_train.shape)

X_train.shape, Y_train.shape
```

```
(97703, 500, 21)
(97704,)
((97703, 500, 21), (97704,))
```

```
#np.random.seed(15)
#tf.random.set_seed(15)

model = keras.models.Sequential([
    keras.layers.LSTM(200, return_sequences=True, input_shape=[500, 21]
    keras.layers.LSTM(150,return_sequences=False),
    #keras.layers.LSTM(100, return_sequences=True,dropout=0.2),
    #keras.layers.LSTM(80, return_sequences=True,dropout=0.2),
    #keras.layers.LSTM(20, return_sequences=False),
    keras.layers.Dense(1)
])

model.compile(loss="mse", optimizer="sgd")
history = model.fit(X_train, y_train, epochs=2, validation_data=(X_vali
```

```
Epoch 1/2
1719/1719 [==============================] - 117s 68ms/step - loss: 0.0
Epoch 2/2
 548/1719 [========>.....................] - ETA: 1:12 - loss: 0.0127
```

```
:f plot_learning_curves(loss, val_loss):
   plt.plot(np.arange(len(loss)) + 1, loss, "b.-", label="Training loss")
   plt.plot(np.arange(len(val_loss)) + 1, val_loss, "r.-", label="Validatic
   plt.gca().xaxis.set_major_locator(mpl.ticker.MaxNLocator(integer=True))
   plt.axis([1, 5, 0, 0.05])
   plt.legend(fontsize=14)
   plt.xlabel("Epochs")
   plt.ylabel("Loss")
   plt.grid(True)

.ot_learning_curves(history.history["loss"], history.history["val_loss"])
.t.show()
```