



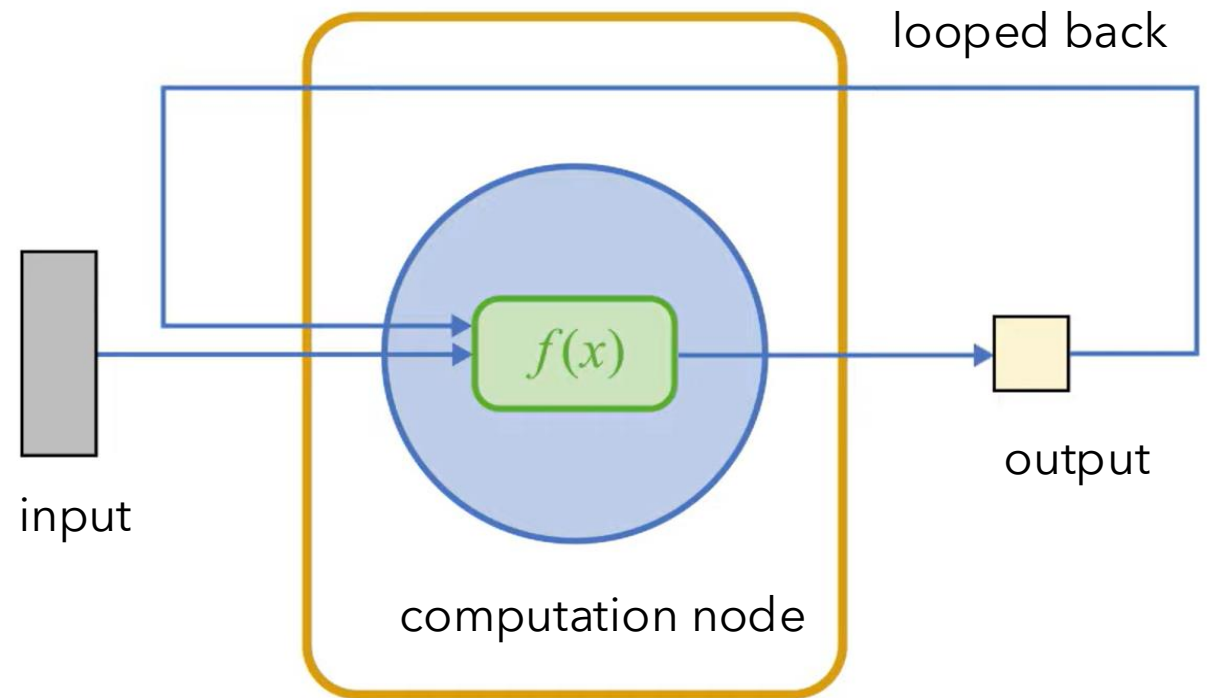
Time series forecasting using LSTM

Week 12 - Advanced Topic 2

Introduction

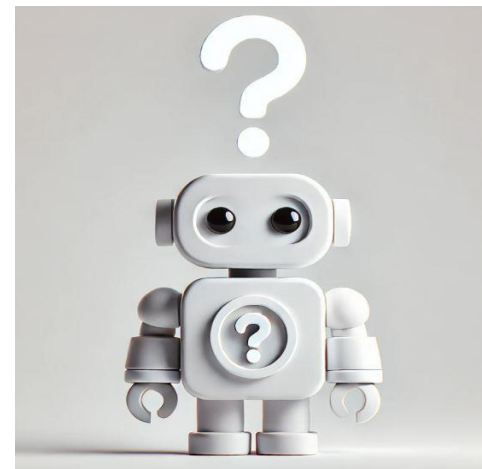
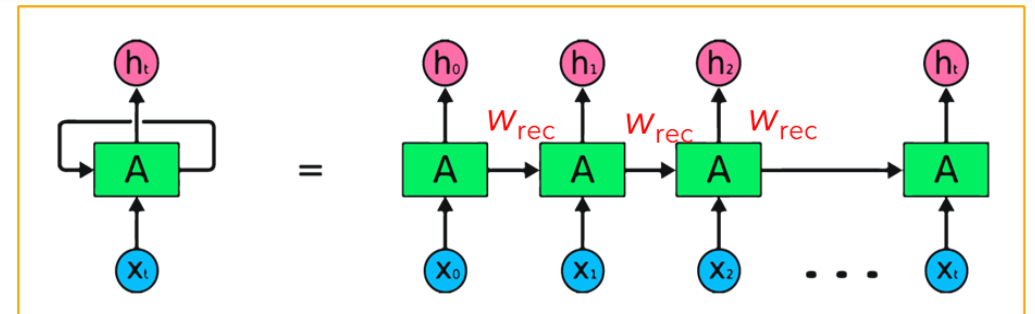
RNN: The Foundation of LSTM

- In an RNN, each step in a sequence feeds its output back into the network to influence the next step.
- This looping allows the RNN to remember information from previous steps, helping it recognize patterns over time.



RNN: The Foundation of LSTM (cont.)

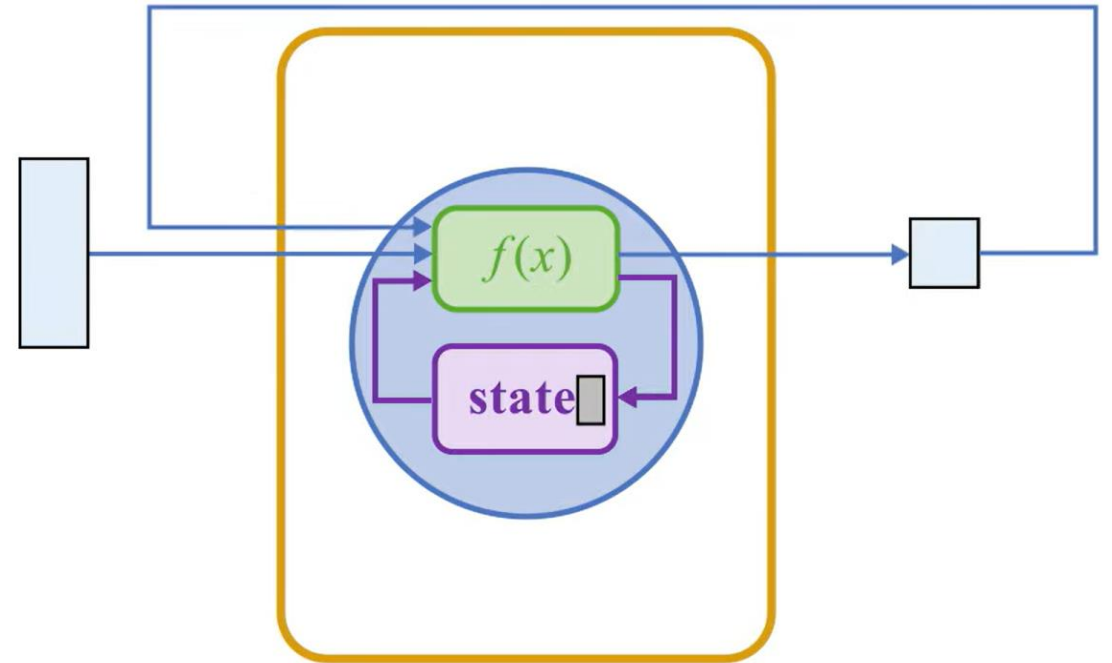
- However, RNNs struggle to retain long-term information due to issues like exploding / **vanishing gradients**, meaning early data points lose or accumulate influence over time.
- For example, in predicting stock prices, an RNN might exaggerate or even **"forget"** significant patterns or events from months ago, making it less accurate for long-term forecasts.



$w_{rec} \sim \text{small} \rightarrow \text{Vanishing}$
 $w_{rec} \sim \text{large} \rightarrow \text{Exploding}$

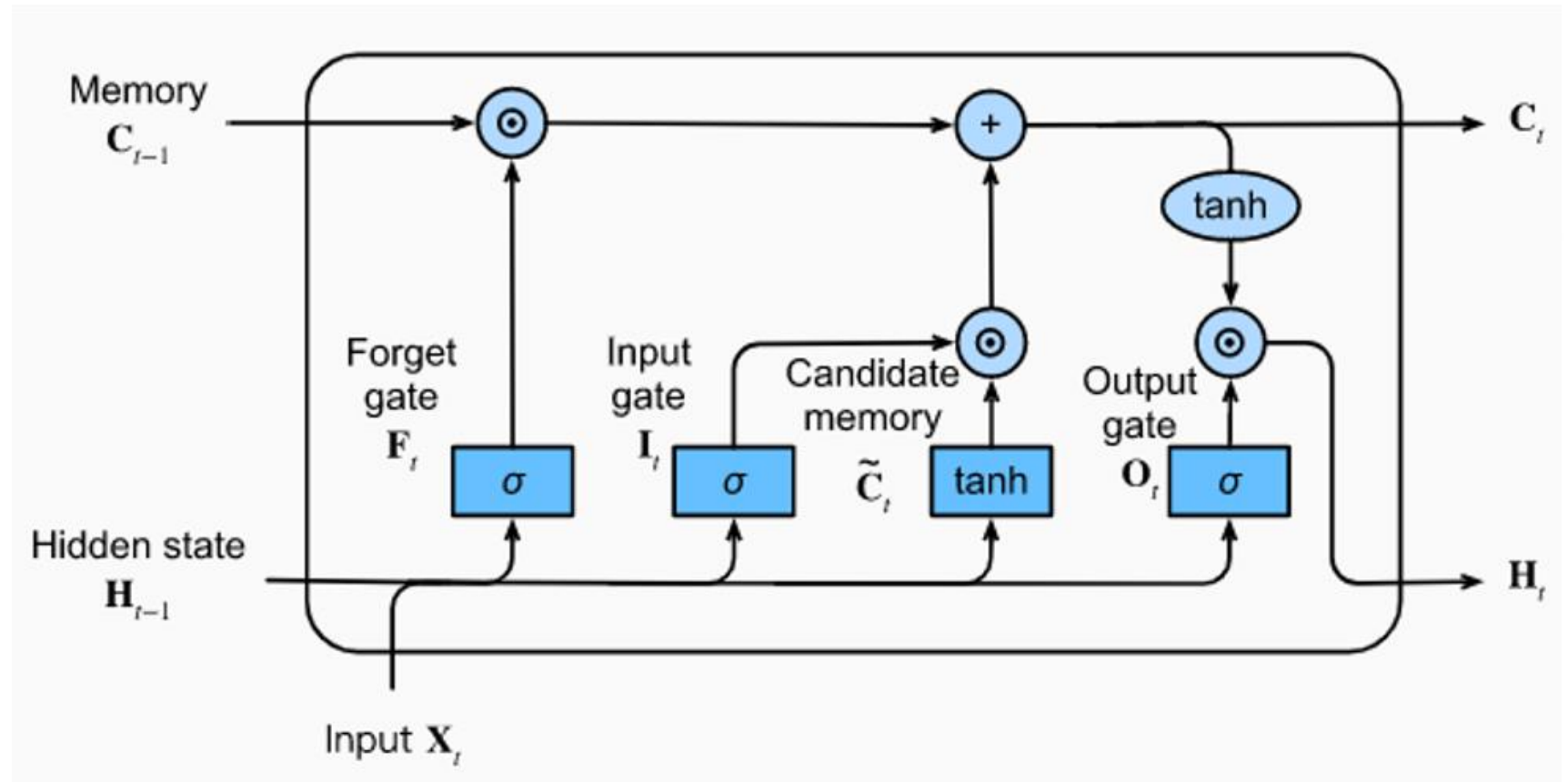
LSTM : an improved version of RNN

- Therefore, LSTM is an improved version of RNN that includes a special '**state**' node, known as the **cell state**, which helps it retain important information over longer sequences.
- This cell state acts as a memory that flows through the network, carrying relevant information across time steps and allowing the model to "remember" crucial details.



LSTM : an improved version of RNN (cont.)

- In addition to the cell state, LSTM also has **parameterized gates** that control the flow of information in and out of the cell state.



LSTM : an improved version of RNN (cont.)

- The cell state and parameterized gates in LSTM allow it to retain, update, and forget information, enabling it to manage long-term dependencies. This design makes LSTM highly effective for time series forecasting, where understanding past trends is crucial.

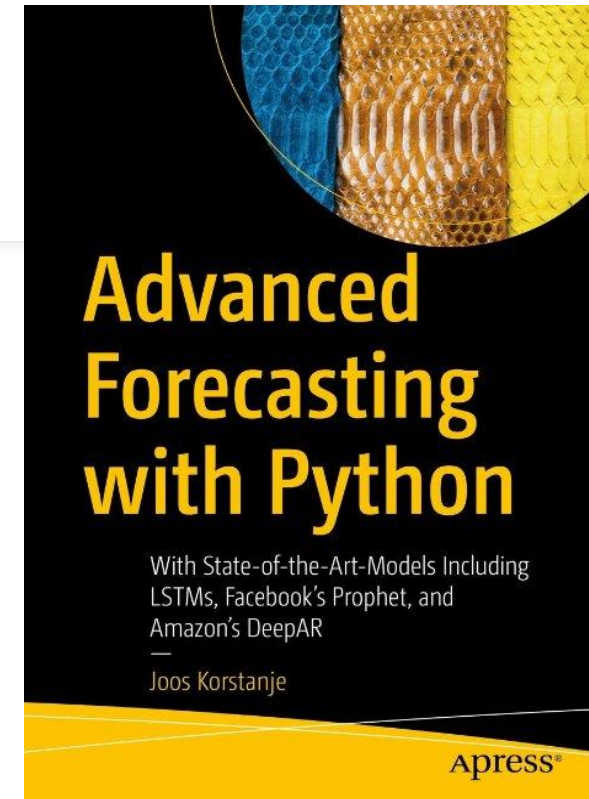


Code Examples



Weather forecast example

- Dataset from Max Planck Institute of Biochemistry in Jena, Germany 2009-2016.
- Time step of 10 minutes.
- Many great LSTM examples can be found on [Kaggle](#). This example is based on chapter 18 of the book "Advanced Forecasting with Python".
- We restricted the example to a single timeseries of temperature.
- Why can LSTM be employed for weather forecast?



Chapter 18, <https://doi.org/10.1007/978-1-4842-7150-6>

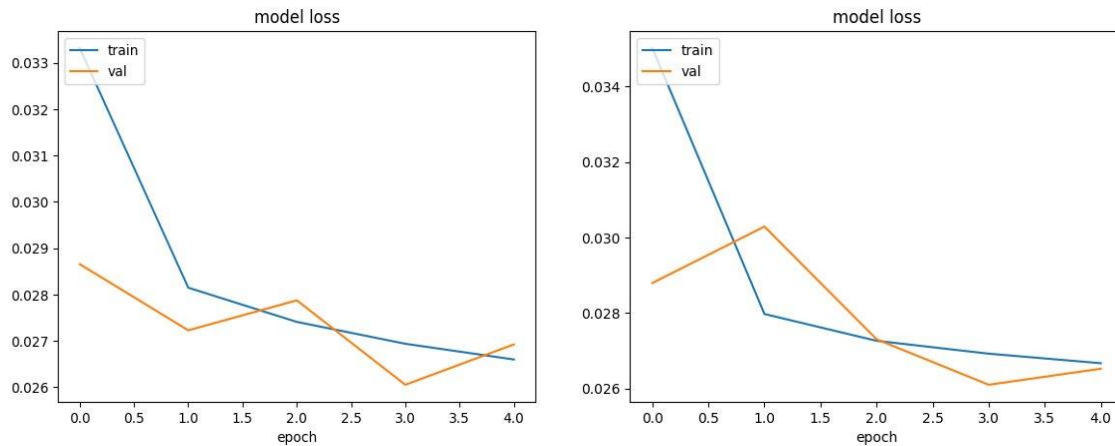
Setup

- LSTM is provided by open-source library Keras, based on paper by Hocreiter in 1997.
- For this setup, **a single layer of 8 neurons** was employed with tanh-activation.
- The training used **5 epochs**, which is the number of training cycles over the dataset.

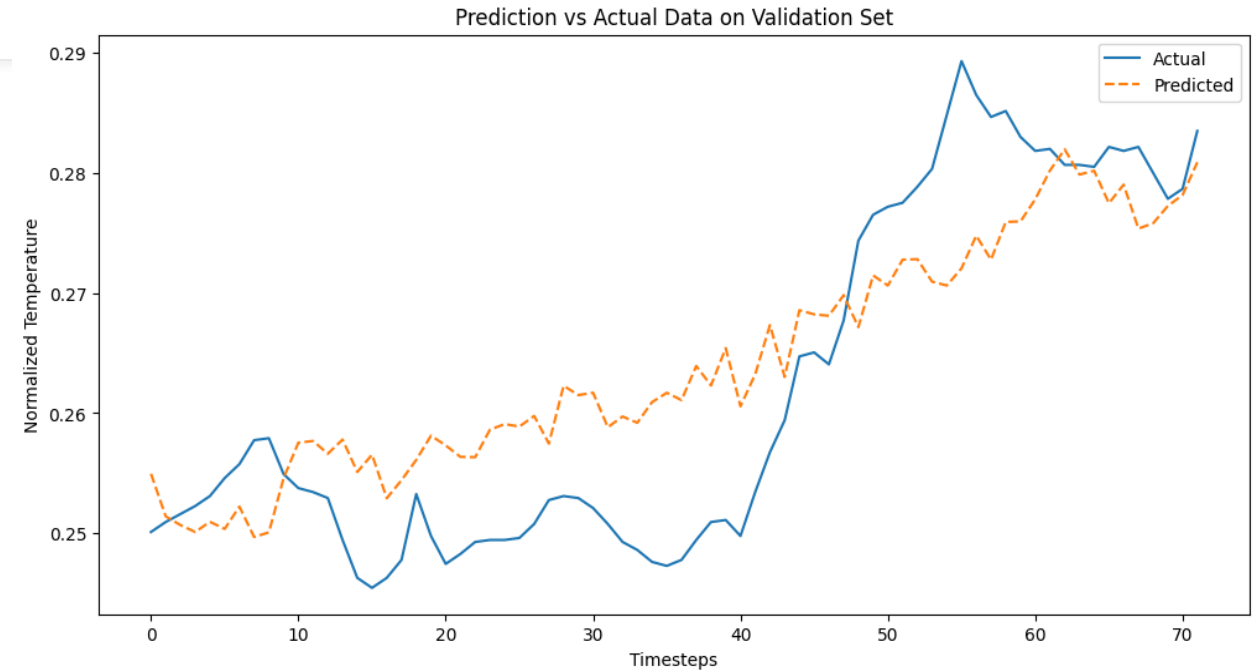
```
import random
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
random.seed(42)
batch_size = 32
simple_model = Sequential([
    LSTM(8, activation='tanh', input_shape=(n_timesteps, n_features)),
    Dense(y_train.shape[1]),
])
simple_model.summary()
simple_model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=0.01),
    loss='mean_absolute_error',
    metrics=['mean_absolute_error'],
)
smod_history = simple_model.fit(X_train_rs, y_train,
    validation_split=0.2,
    epochs=5,
    batch_size=batch_size,
    shuffle = True
)
```

Prediction vs Actual Data

- LSTM receives an **R2 score of 0.93**, better than both Gated Recurrent Unit (GRU) of 0.88 and a simple RNN of 0.66 on the same example.



Two learning processes shown above



Model

SimpleRNN

GRU

LSTM

One layer of 8

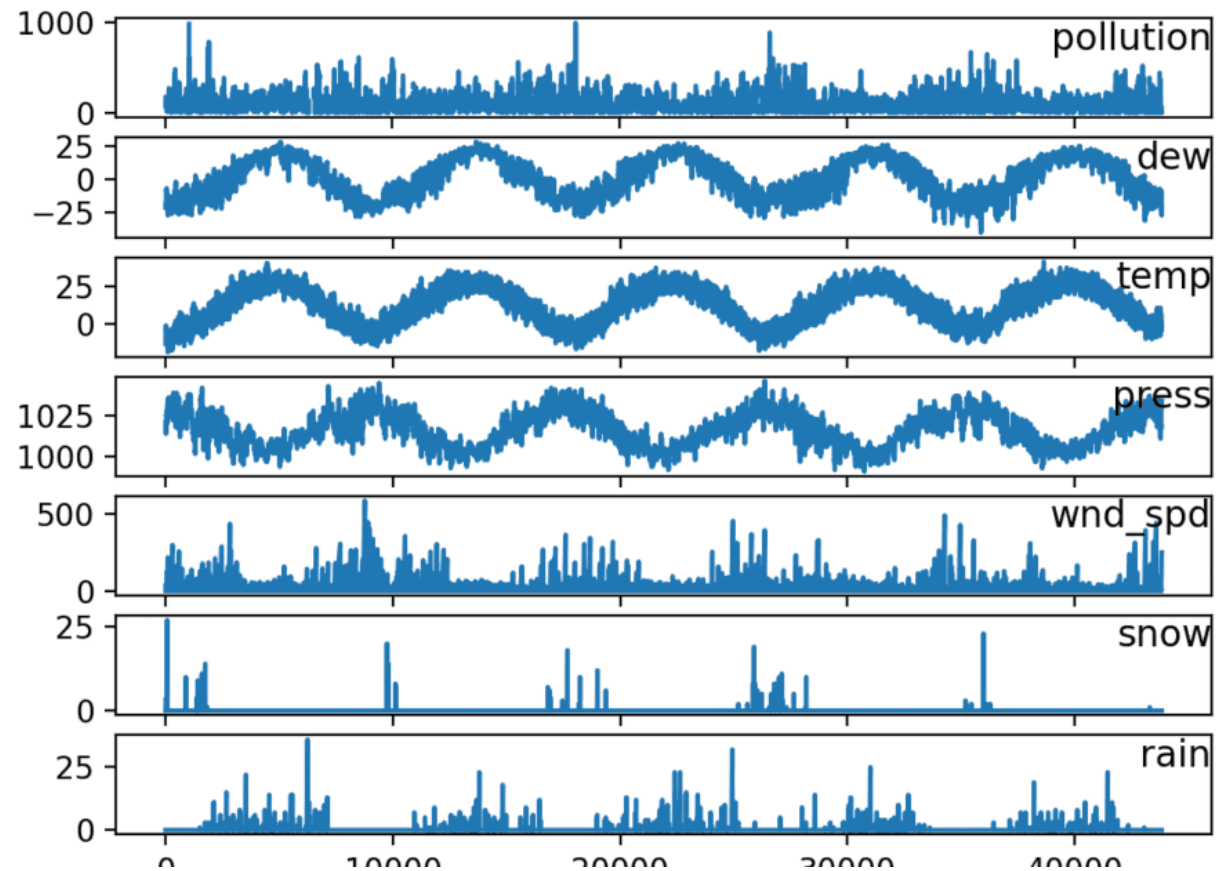
0.66

0.88

0.93

Pollution in Beijing LSTM

Predict the pollution based on historical pollution measurements and weather conditions



Preparing data for training

- Frame the dataset as a supervised learning problem and **normalize** the input variables
 - Predict the pollution at the current time (t) given the pollution measurements and weather conditions at the **prior time step**
- Label and **encode** features
 - i.e. encode wind direction
- Remove the weather variables for the time (t) to be predicted
 - Keep pollution

	pollu(t-1)	dew(t-1)	temp(t-1)	press(t-1)	wnd dir(t-1)	wnd spd(t-1)
1	0.129779	0.352941	0.245902	0.527273	0.666667	0.002290
2	0.148893	0.367647	0.245902	0.527273	0.666667	0.003811
3	0.159960	0.426471	0.229508	0.545454	0.666667	0.005332
4	0.182093	0.485294	0.229508	0.563637	0.666667	0.008391
5	0.138833	0.485294	0.229508	0.563637	0.666667	0.009912

	snow(t-1)	rain(t-1)	current Poll(t)
1	0.000000	0.0	0.148893
2	0.000000	0.0	0.159960
3	0.000000	0.0	0.182093
4	0.037037	0.0	0.138833
5	0.074074	0.0	0.109658

Training using keras

- Define and fit model
 - Split into train and test sets
 - In this example we will train on 1 year test on 4 years
 - 50 neurons in first hidden layer and 1 neuron in output for predicting pollution
 - Fit for 50 training epochs

```
2 # split into train and test sets
3 values = reframed.values
4 n_train_hours = 365 * 24
5 train = values[:n_train_hours, :]
6 test = values[n_train_hours:, :]
7 # split into input and outputs
8 train_X, train_y = train[:, :-1], train[:, -1]
9 test_X, test_y = test[:, :-1], test[:, -1]
10 # reshape input to be 3D [samples, timesteps, features]
11 train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1]))
12 test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))
13 print(train_X.shape, train_y.shape, test_X.shape, test_y.shape)
```

```
2 # design network
3 model = Sequential()
4 model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))
5 model.add(Dense(1))
6 model.compile(loss='mae', optimizer='adam')
7 # fit network
8 history = model.fit(train_X, train_y, epochs=50, batch_size=72, val
```

Evaluate

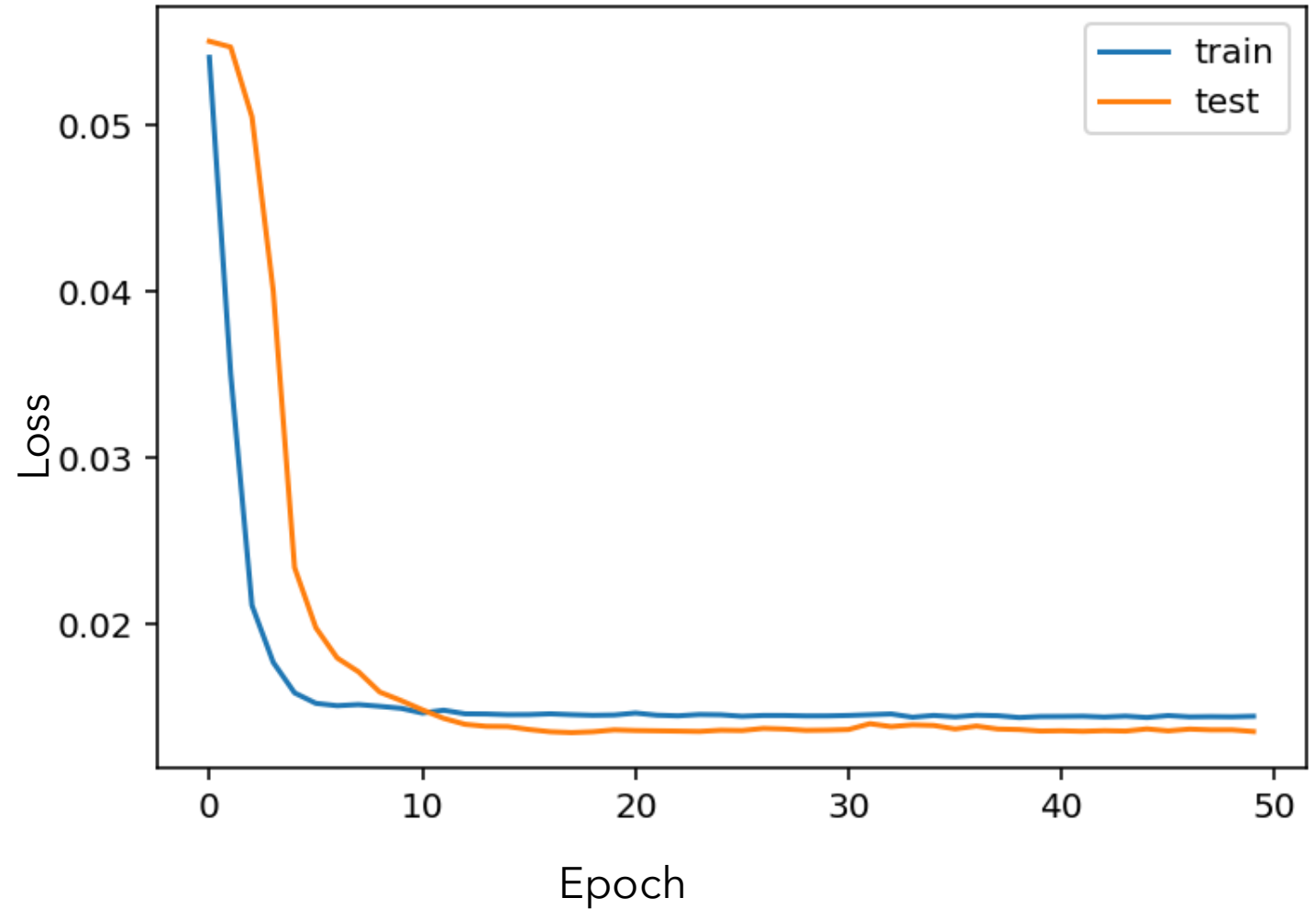
RMSE= 26.496 (pollution values
around hundreds)

Lower than persistence model

RMSE of 30

What is implied by our test loss
dropping below our training loss?

Train and test loss



Pros and Cons

Pros

- Captures patterns over long sequences.
- Suitable for nonlinear time series (despite ARIMA)
- Gated structure improves training on long sequences.

Cons

- Demands more processing power.
- Takes longer to train on large datasets.
- Complex architecture requires careful parameter tuning.

Conclusions


- LSTM and GRU (a simplified LSTM) are two RNN models that could perform for forecasting tasks.
- LSTMs are specifically designed to overcome the vanishing gradient problem of RNN, making them effective in predicting long-term data

Key takeaways:

The LSTM is advanced type of RNN as it has a large number of components that allow it to remember longer-term trends

Sources

1. <https://medium.com/@CallMeTwitch/building-a-neural-network-zoo-from-scratch-the-long-short-term-memory-network-1cec5cf31b7>
2. <https://mohameddhaoui.github.io/deeplearning/LSTM/>
3. <https://arxiv.org/pdf/1909.09586>
4. <https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/>



Thank you!