TIME SERIES 501

Lesson 8: Time Series through Deep Learning

Learning Objectives

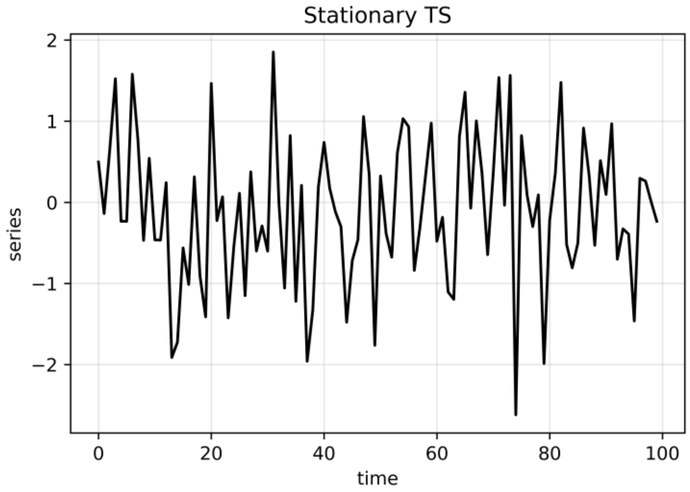
You will be able to do the following:

- Explain why deep learning is useful for time-series forecasting.
- Identify pros and cons of the deep-learning approach.
- Describe how time series can be modeled using recurrent neural networks (RNNs).
- Describe how long short-term memory units (LSTM) can improve on simple RNNs.
- Use Python* and Keras to create deep-learning models for time series.

Why Deep Learning?

Neural networks offer several benefits over traditional time series forecasting models:

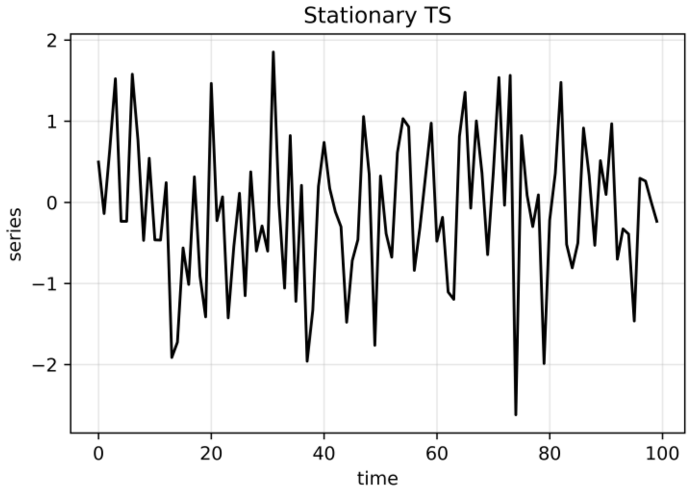
- **Automatically** learn how to incorporate series characteristics like trend, seasonality, and autocorrelation into predictions.
- Able to capture very complex patterns.
- Can simultaneously model many related series instead of treating each separately.



Why Not Deep Learning?

Neural network benefits don’t come for free:

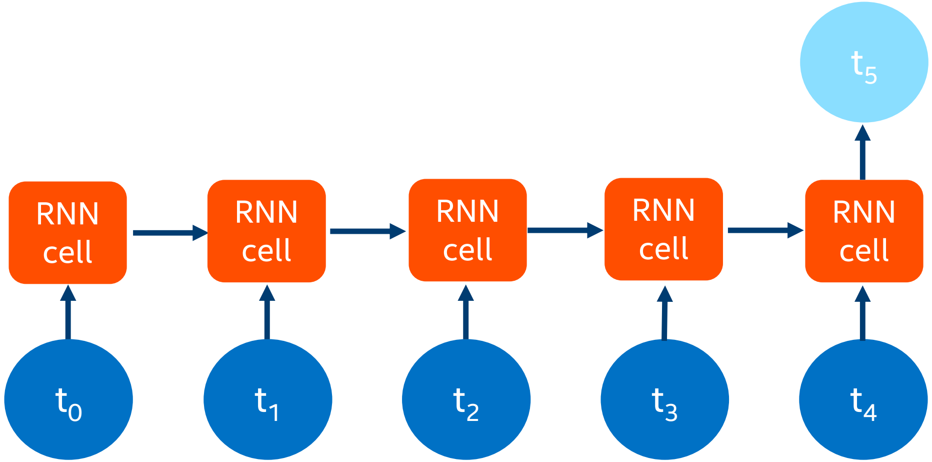
- Models can be complicated to build.
- Models are computationally expensive to build (GPUs help accelerate training).
- It is very challenging to explain / interpret the predictions made by the model (“black box”).
- Tends to perform best with large training datasets.



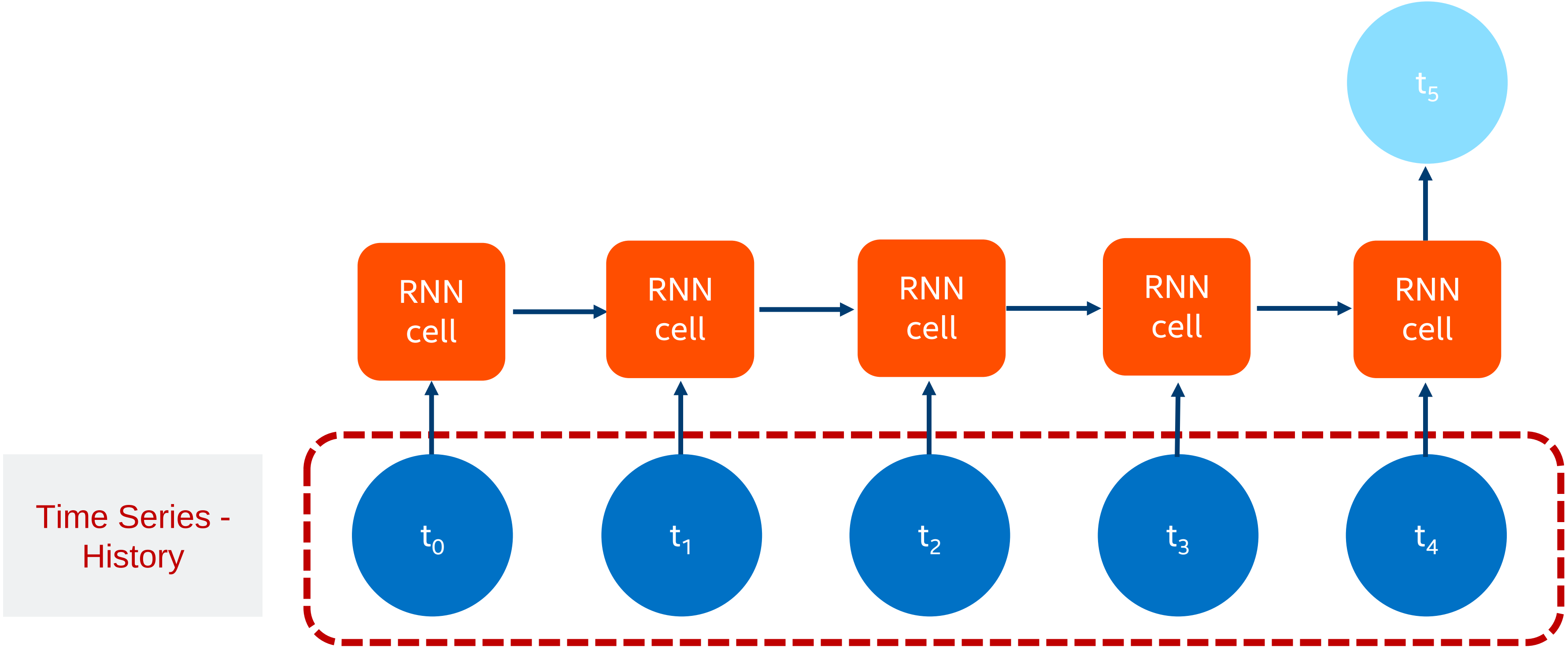
What Is an RNN?

Recurrent neural networks map a sequence of inputs to predicted output(s).

- Most common format is **many-to-one,** which maps an input sequence to one output value
- Input at each time step is used to sequentially update the RNN cell's **hidden state** or **memory**.
- After processing of the input sequence, hidden state information is used to predict the output.



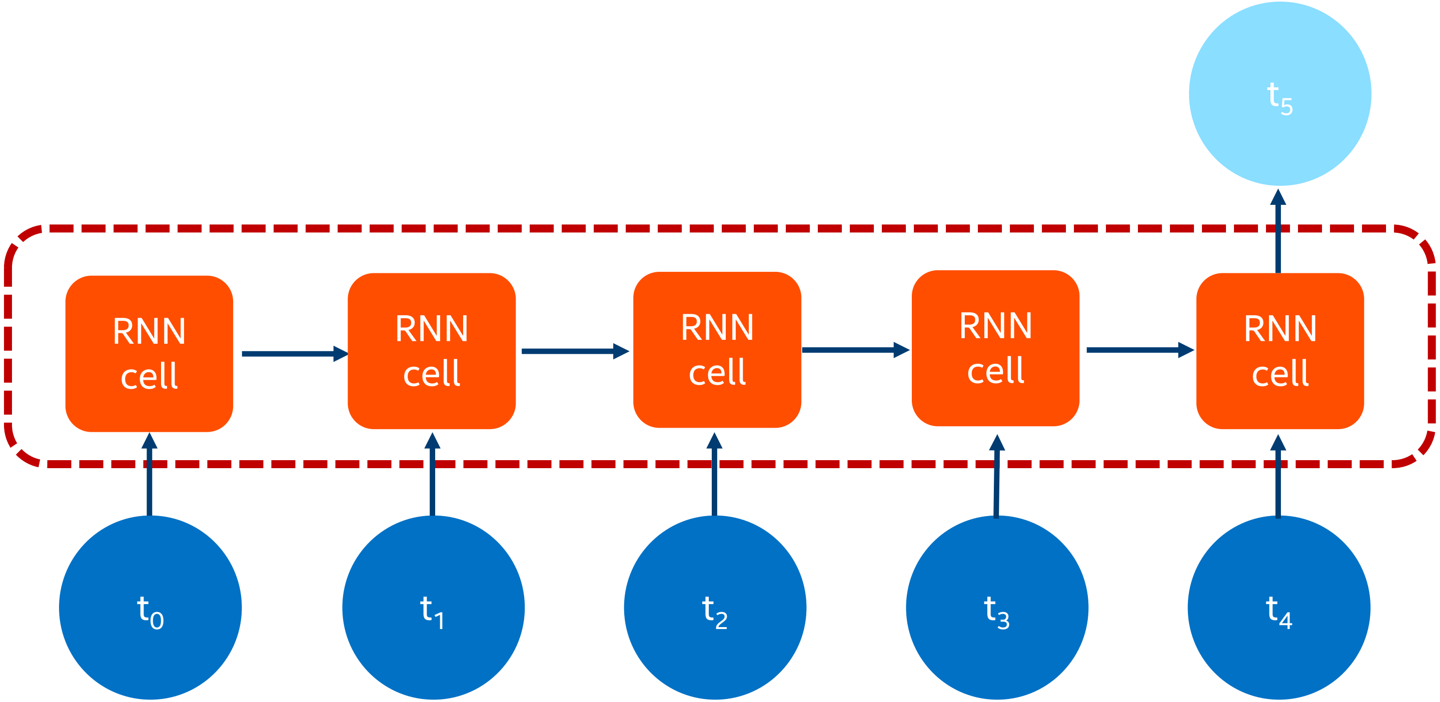
Applying RNN to Time-Series Forecasting



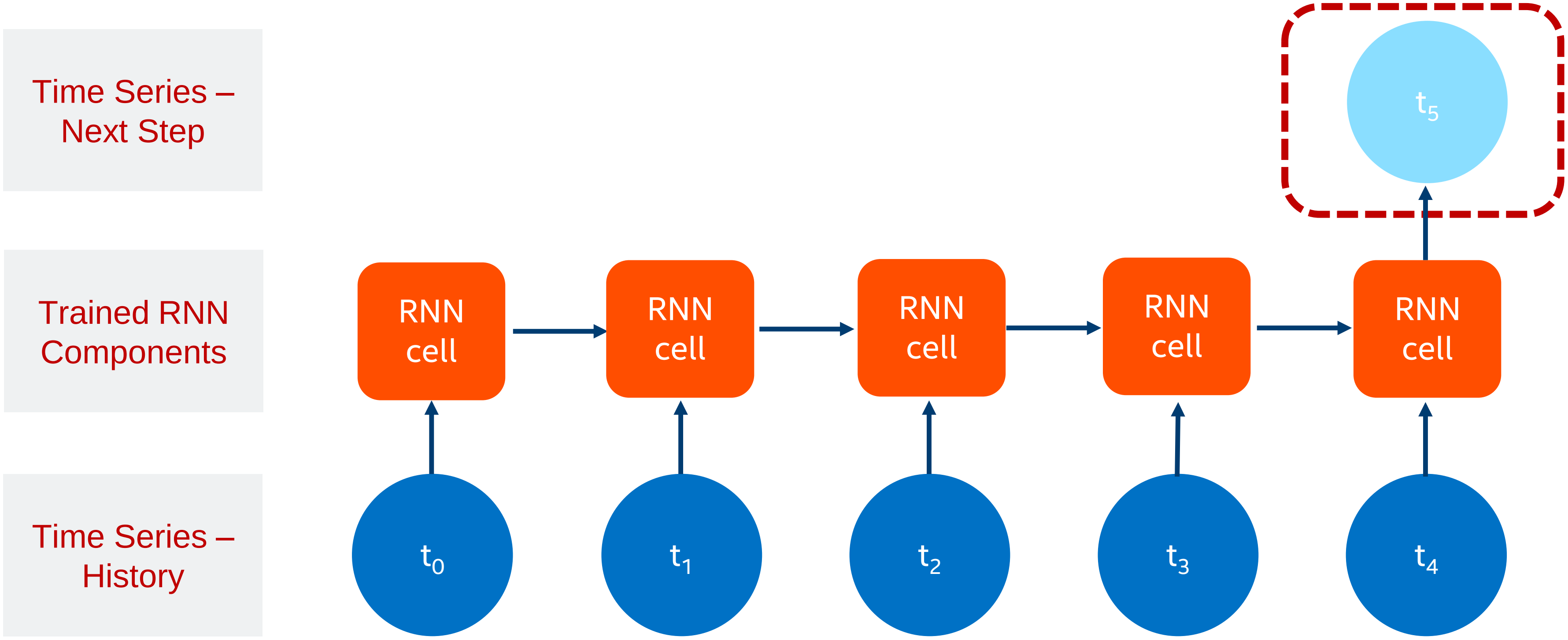
Applying RNN to Time-Series Forecasting

Trained RNN Components

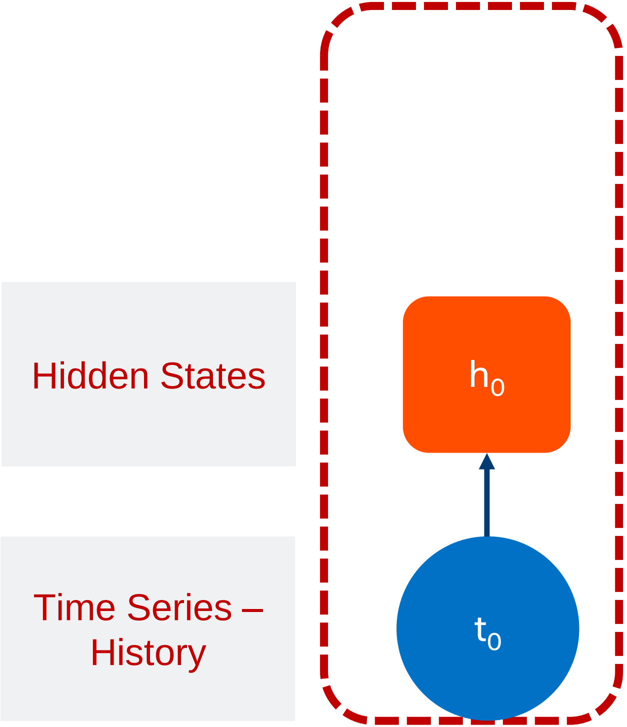
Time Series - History



Applying RNN to Time-Series Forecasting



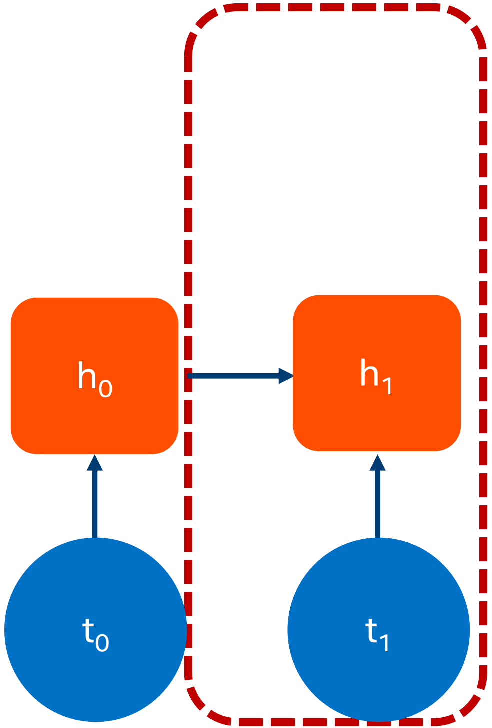
Sequential Information Flow Sequence: Step 0



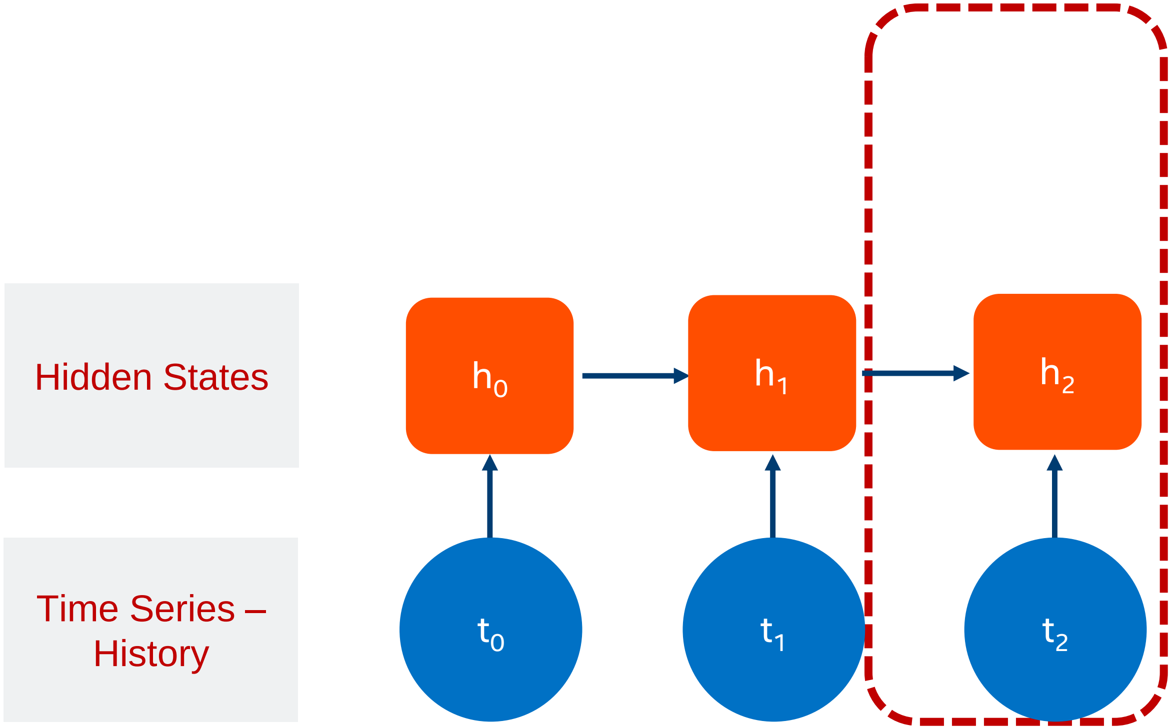
Sequential Information Flow Sequence: Step 1

Hidden States

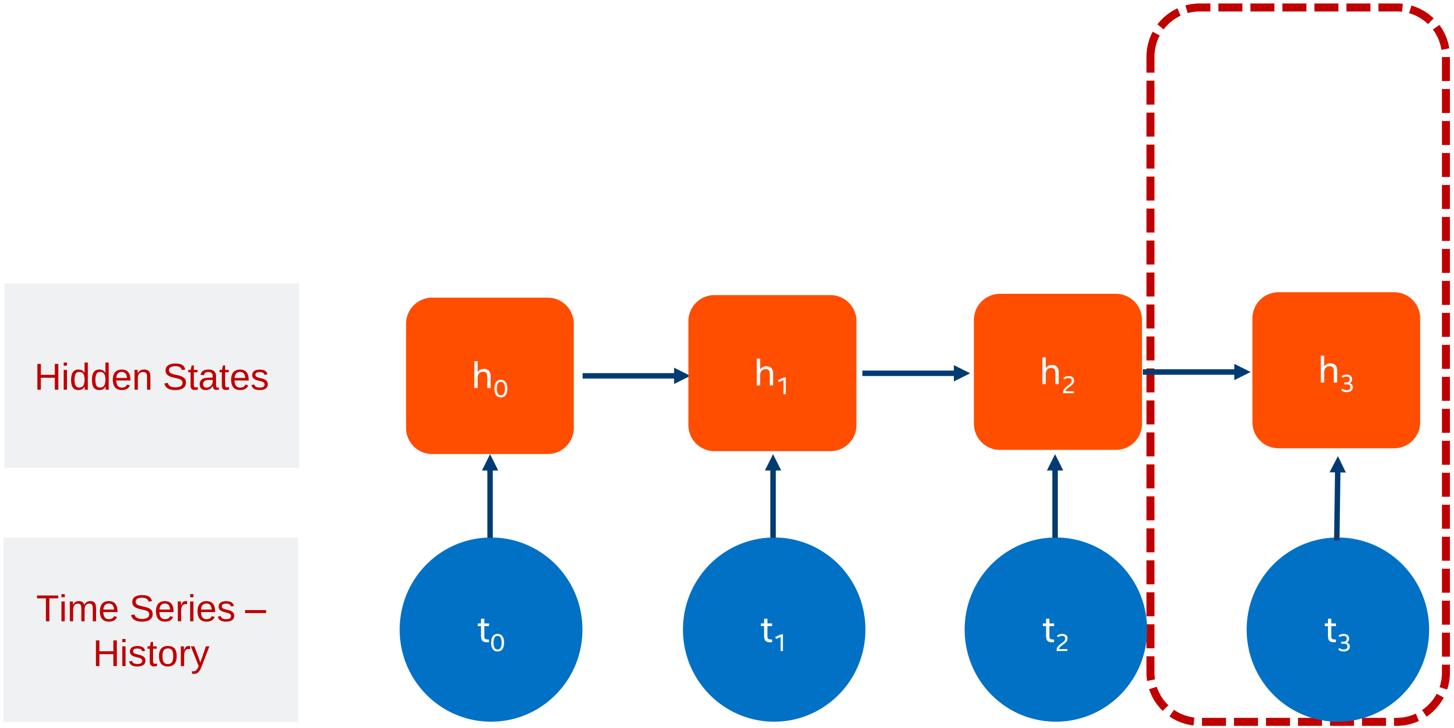
Time Series – History



Sequential Information Flow Sequence: Step 2



Sequential Information Flow Sequence: Step 3

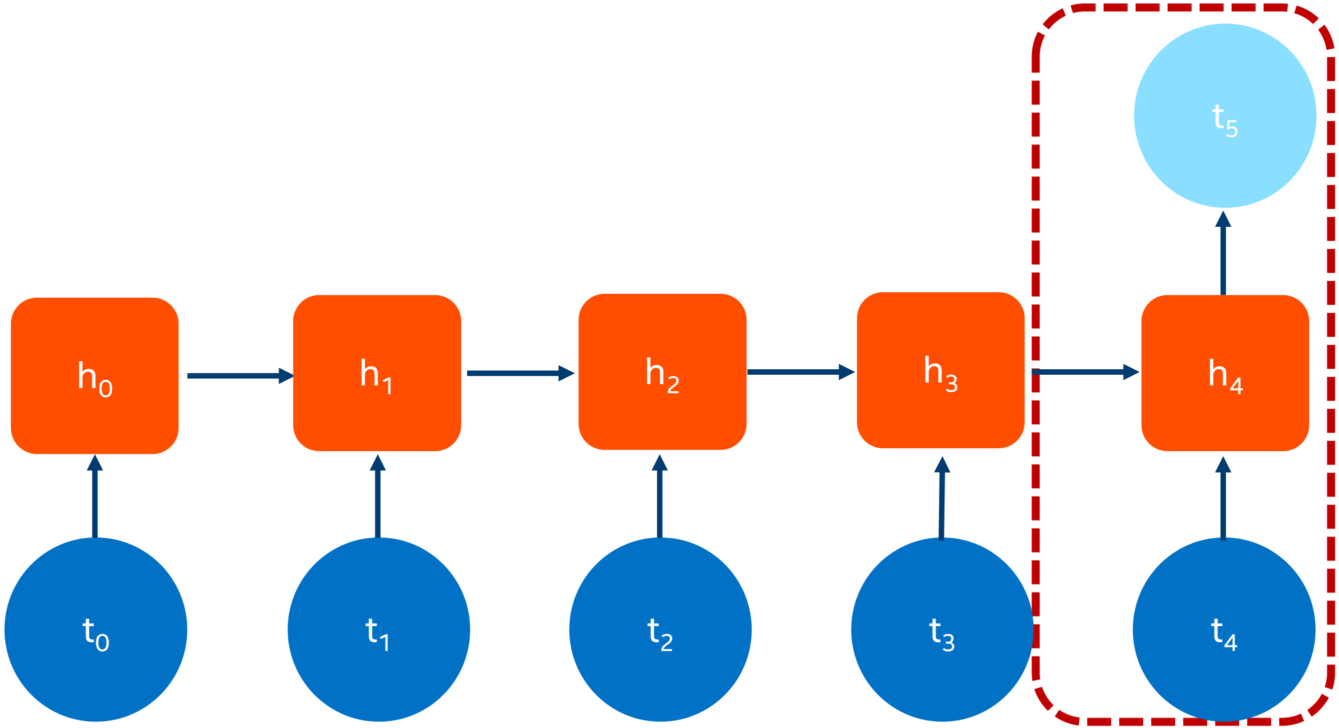


Sequential Information Flow Sequence: Step 4

Time Series – Next Step

Hidden States

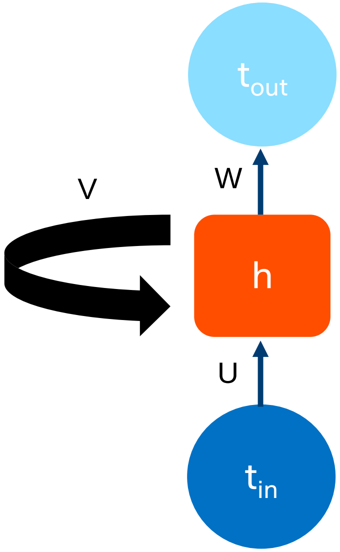
Time Series – History



What's Going on under the Hood? (cont.)

RNNs are often represented as a cycle, simplifying the diagram

- The same U and V are applied repeatedly to sequentially update the hidden state, using the previous hidden state and the new input at each time step



$$t_{out} = Wh_{out\text{-}1}$$

$$h_i = \sigma(Ut_{i-1} + Vh_{i-1})$$

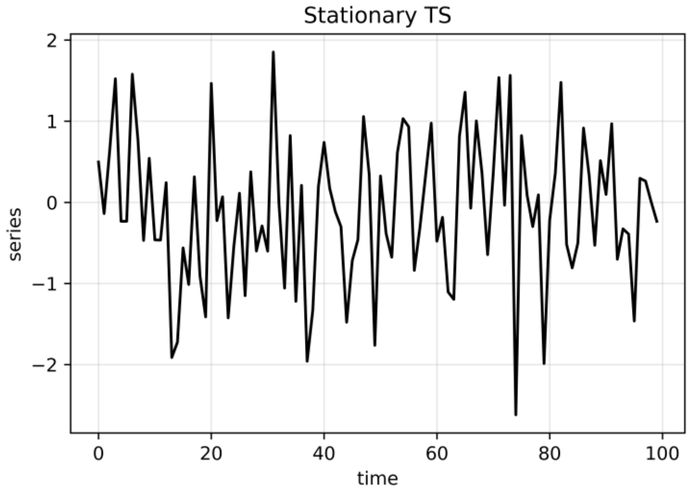
$$\sigma(x) = \frac{e^x}{e^x + 1}$$

The Limitations of Simple RNNs

Time series image – large number of timesteps to process

Basic RNNs often struggle when processing long input sequences

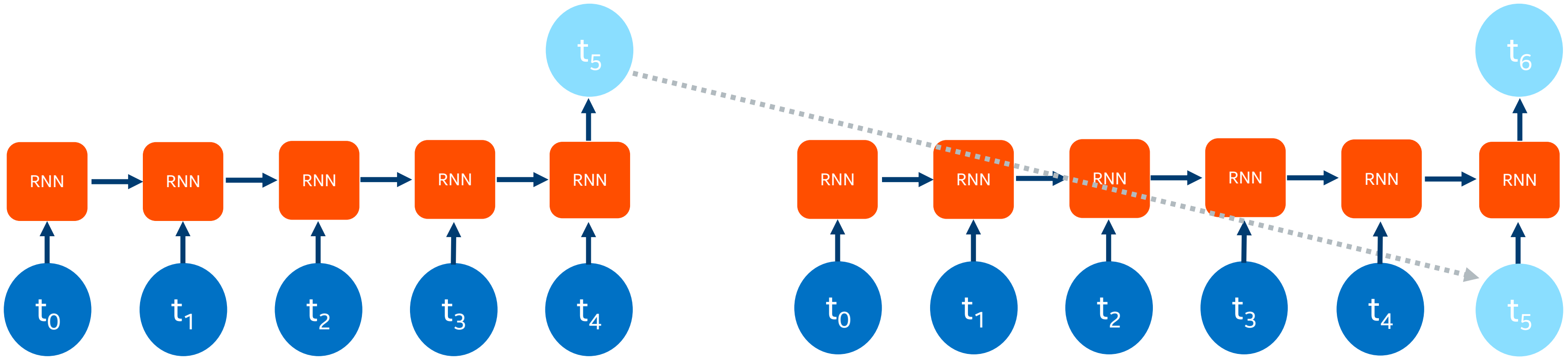
- Mathematically difficult for RNNs to capture long-term dependencies over many time steps
- Problem for time series, where sequences are often hundreds of steps or more
- **Long short-term memory networks** (LSTMs) can mitigate these issues with a better memory system



Recap: Time Series Many-to-One RNN

Use past time steps to forecast future time steps

- Input: time series' historical steps
- Output: time series' next step
- Can forecast multiple time steps by adding previous predicted step to input sequence



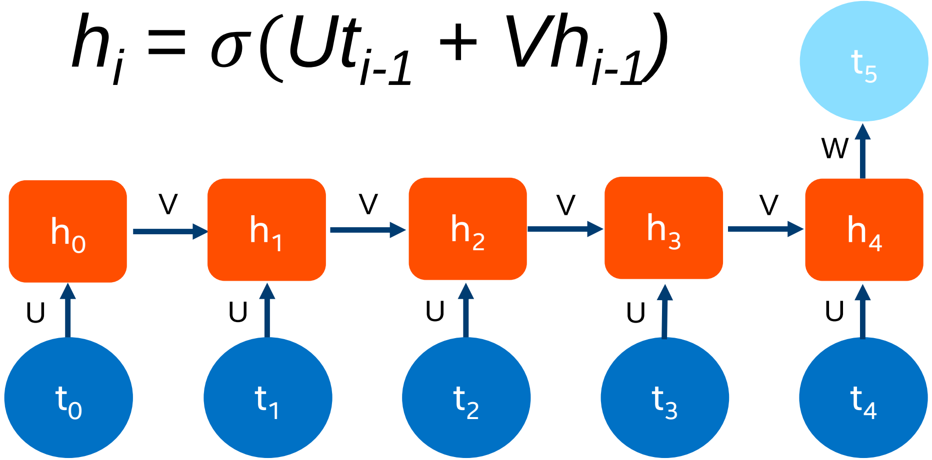
What's Going on under the Hood?

RNN unit math:

- *U*, *V*, and *W* are **trainable weight matrices,** the h_i are **hidden states**
- *σ* is the **sigmoid activation function**, and the weight matrices are applied as linear transformations

$$h_i = \sigma(Ut_{i-1} + Vh_{i-1})$$

$$t_{out} = Wh_{out-1}$$

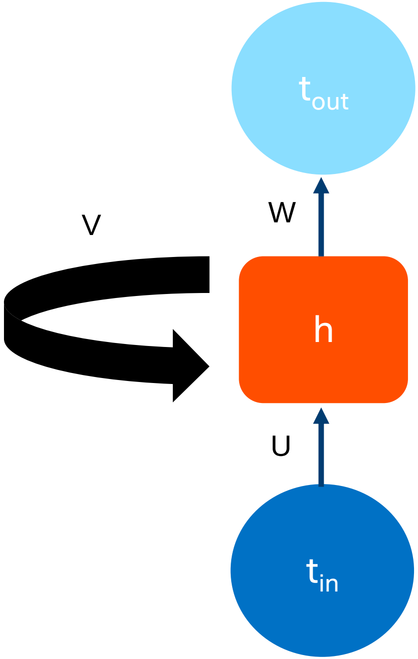


$$\sigma(x) = \frac{e^x}{e^x+1}$$

What's Going on under the Hood? (cont. 2)

How do we obtain the weight matrices *U*, *V*, and *W*?

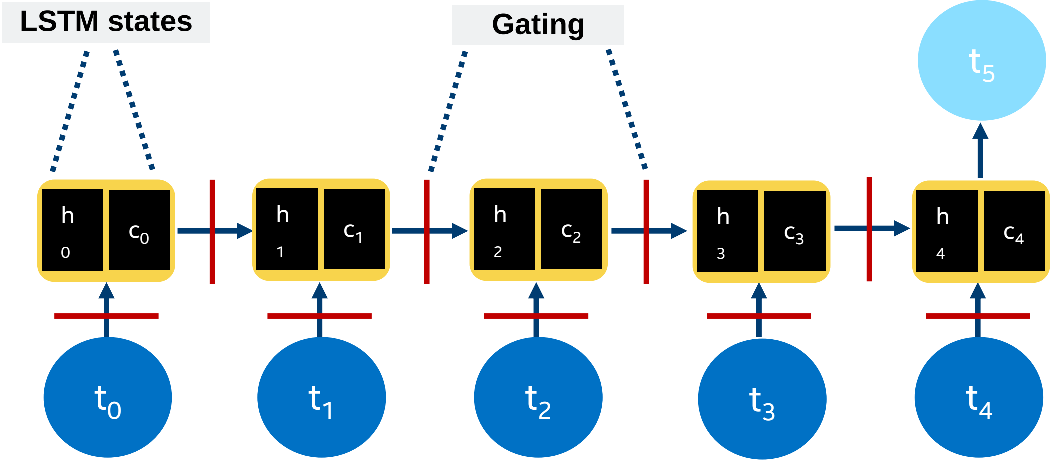
- When we train an RNN, we are actually finding weights via the **backpropagation algorithm.**
- In backpropagation, we repeatedly process the training data, updating the weights in order to minimize a **cost function.**
- For time series forecasting, a typical cost function would be **mean squared error** or a similar metric.
- Intuitively, we find values for *U*, *V*, and *W* that cause our predicted outputs t_{out} to be as close to the true target values as possible.



What Is a LSTM? (cont.)

Long short-term memory networks regulate information flow and memory storage.

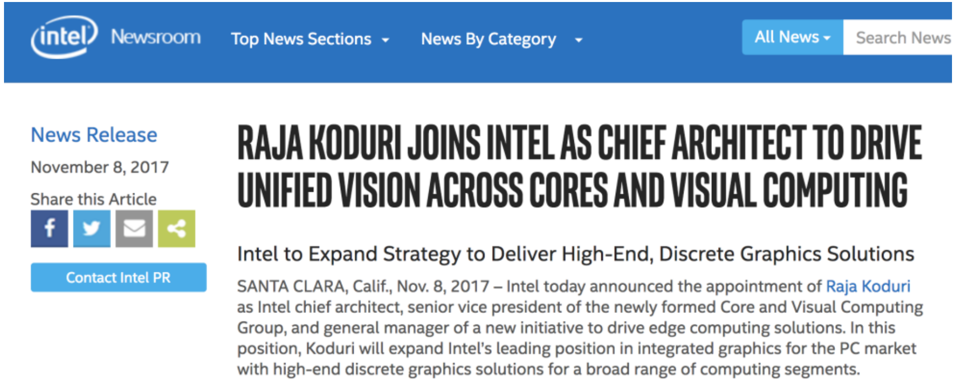
- LSTM cells share **forget, input,** and **output gates** that control how memory states are updated and information is passed forward.
- At each time step, the input and current states determine the gate computations.



Choosing LSTM vs. RNN?

Always consider the problem at hand.

- If sequences are many time steps long, an RNN may perform poorly.
- If training time is an issue, using a LSTM may be too cumbersome.
- Graphics processing units (**GPUs**) speed up all neural network training but are especially recommended when training LSTMs on large datasets.



Use Python and Keras to Construct RNNs and LSTMs for Time-Series Forecasting

Next up is a look at building these neural networks in Python.

- See notebook entitled *Introduction_to_Deep_Learning_student.ipynb*

Legal Notices and Disclaimers

This presentation is for informational purposes only. INTEL MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

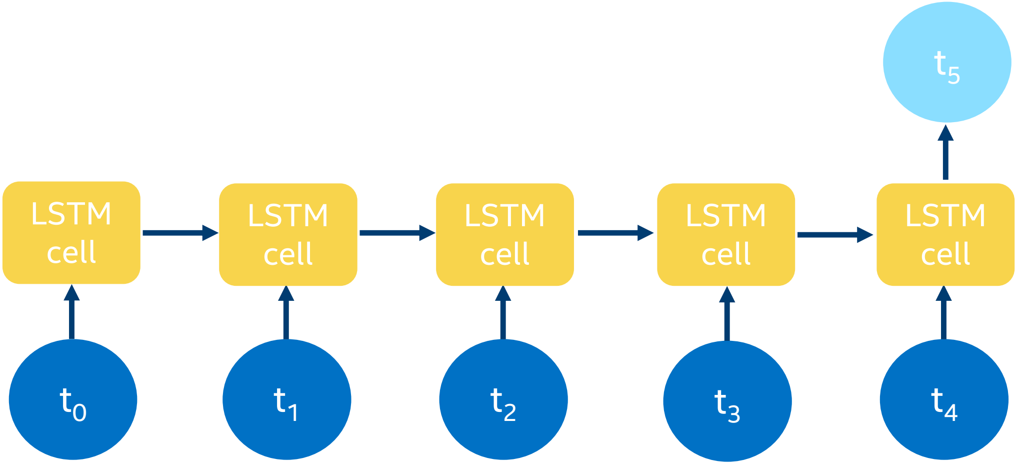
Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at intel.com.

Sample source code is released under the Intel Sample Source Code License Agreement.

What Is a LSTM?

Long short-term memory networks share RNNs' conceptual structure.

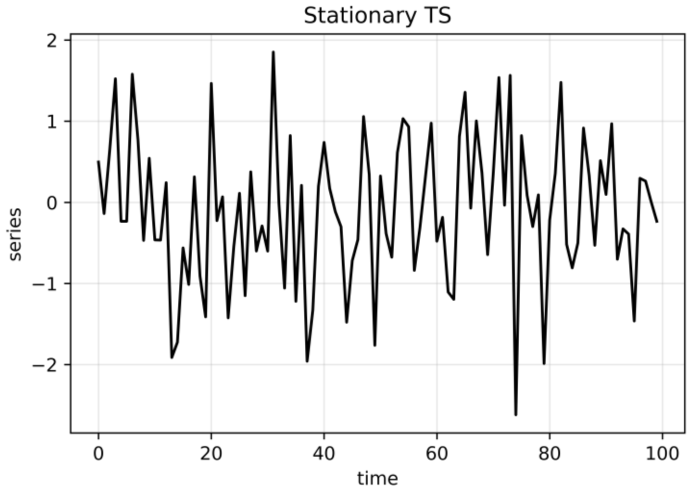
- LSTM cells have the same role as RNN cells in sequential processing of the input sequence.
- LSTM cells are internally more complex, with **gating** mechanisms and **two states** – a "hidden state" and a "cell state."



LSTM vs. RNN?

Are LSTMs always better than simple RNNs?

- LSTMs are better suited for handling long-term dependencies than RNNs
- However, they are much more complex, requiring many more trainable weights
- The result is that they take longer to train (slower backpropagation) and can be more prone to overfitting



APPLICATIONS IN PYTHON

Learning Objectives Recap

In this session you learned how to do the following:

- Explain why deep learning is used for time-series forecasting and some pros/cons.
- Describe how time series can be modeled using RNNs and LSTMs.
- Use Python to create these models.