# Recurrent Neural Networks

# So Far…

- Dense neural networks work like a typical machine learning algorithm. They can produce categorical or continuous predictions.
- Convolutional neural networks are primarily used in image recognition.
  - Not always—just **usually**
- Both types take input data all at once: All the inputs from a single sample are presented at once.
- But what about time or sequence-dependent data?
  - Language
  - Time series

# Recurrent Neural Networks (RNNs)

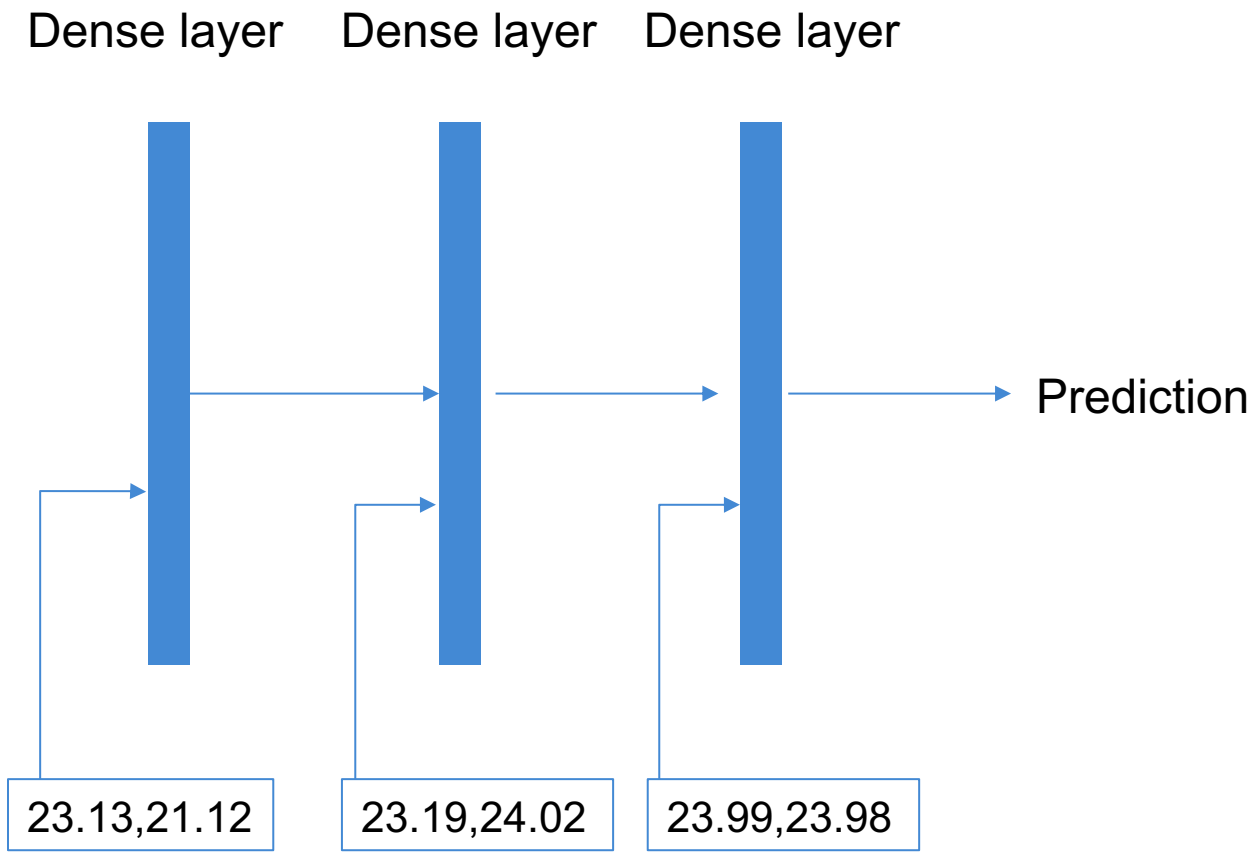
- RNNs deal with *sequential* data
  - Sequential: The order of the data is important!
- To contrast with CNN, CNN is positional—nearby inputs matter
- In RNNs, only the previous inputs matter
  - RNNs are sometimes called temporal (time-based).
  - Given what happens at time-1, time-2, etc., what happens at time-n?
- How do we remember the previous state?

# The RNN Layer

The basic RNN layer is a dense layer with 2 sets of inputs.

1. The first set of inputs is the traditional data inputs.
2. The second set of inputs is the **output** from the previous step.

# A Simple Example (Stock Data)




Let's try and look at the previous 3 days' data to predict tomorrow's price in stocks.

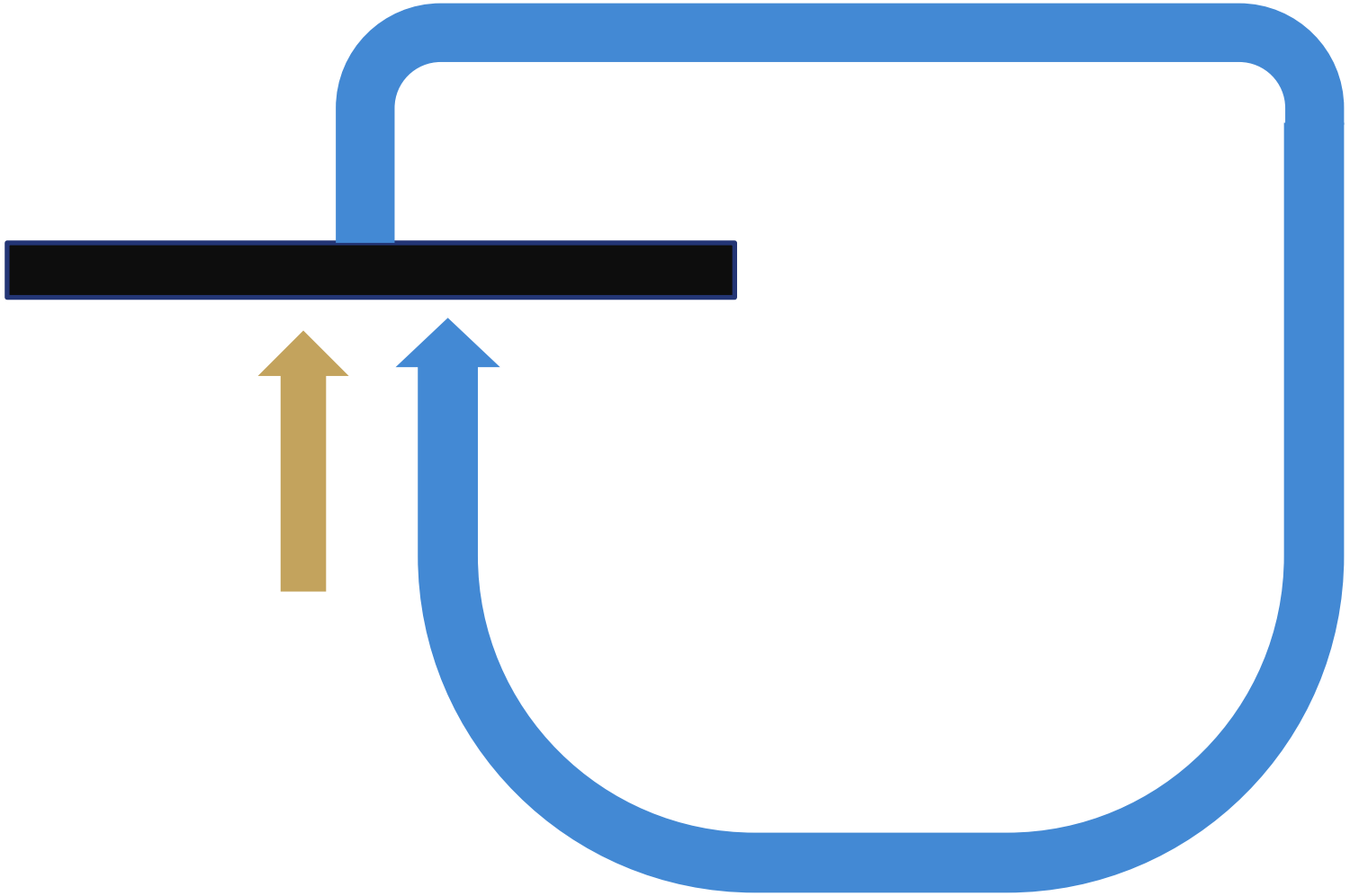
| Day | Open price | Close price |
|---|---|---|
| 1 | 23.13 | 21.12 |
| 2 | 21.19 | 24.02 |
| 3 | 23.99 | 23.98 |

# A Little Twist

What if, instead of 3 different layers, we reused the first layer?

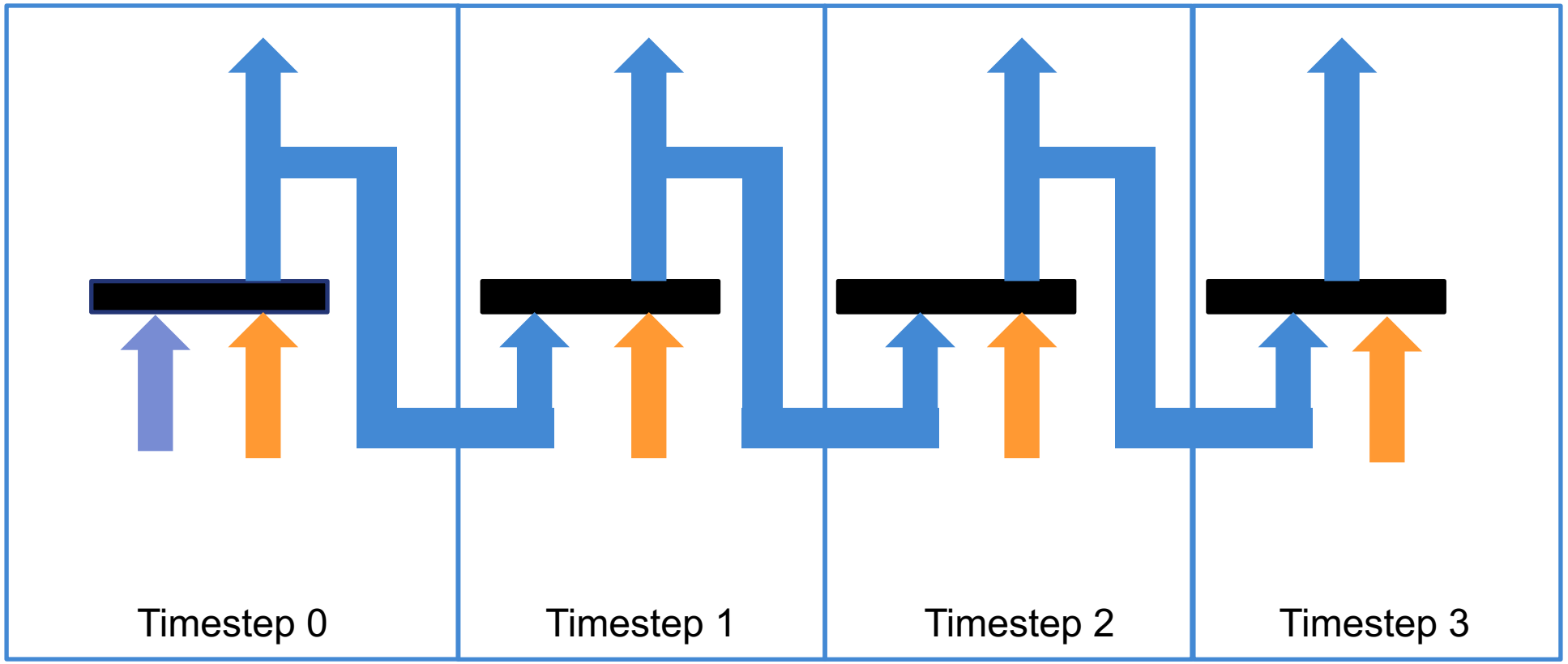
- Remember a “layer” is just the weights and activation function.
- The previous slide is an example of an “unrolled” RNN.
- RNNs reuse the same weights at each timestep.
- However, the output of the previous timestep is used as an input in the current timestep.
- This is the origin of the name “recurrent NN.”

# Rolled Up RNN



# Relook at the Unrolled Version

Instead of a single prediction, we will get a prediction at each step. We can choose to save or discard the intermediate predictions.




At timestep 0 there is no previous output, so the inputs are 0s.

# RNN Conventions

- The first “recurrent” input is all 0s.
- All the data must be the same length.
  - This presents challenges for language! Akin to saying all sentences must be the same length!
- Instead of a single weight matrix for an input that is a joint concatenation of the recurrent input and the data input, RNNs are represented with two weight matrices.
  - It is the same thing.
- RNN is the first step in the network having “memory.”

# 2 vs. 1 Matrix

- Look at just the first term for both equations and see how they are the same result.
- For some reason, the convention is to represent the RNN layer as 2 matrices—even in the libraries which implement neural networks.

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \\ w_{41} & w_{42} & w_{43} & w_{44} & w_{45} \\ w_{51} & w_{52} & w_{53} & w_{54} & w_{55} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \\ w_{51} & w_{52} & w_{53} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} w_{14} & w_{15} \\ w_{24} & w_{25} \\ w_{34} & w_{35} \\ w_{44} & w_{45} \\ w_{54} & w_{55} \end{bmatrix} \begin{bmatrix} x_4 \\ x_5 \end{bmatrix}$$

DataScience@SMU

# Brief NLP Introduction

# Natural Language Processing

- Natural language processing (NLP) is the primary problem solved by RNNs (and their descendants).
- Why are RNNs so used in NLP?
- But before we start building a network, we have a problem: How do I convert text to numbers?
  - Traditionally, words are treated as categories.
    - One-hot encoded
    - That gives **a lot** of categories!
- How do we deal with sentences of different lengths?
- How do we determine if things are the same? Or different?

# NLP and RNN

Predict the next word in the sentence:

- The cow jumped over the _____
- Since English is read left to right, if we feed a representation for the word “The,” then “cow,” etc., this falls naturally into the same idea as an RNN!
- (What if your language is left to right: put in the leftmost word first!)
- (What if you want to determine the first word of the sentence by the words that come after: run the sequence **backwards**)

# Words as Numbers

- Rather than a category, let's try to represent our words as vectors.
  - Rather than a giant sparse one-hot encoded vector, can we find dense, or non-sparse vectors for words?
  - This is an active area of research—but the answer is **yes**, let us represent words as vectors and better yet—let our network **find them**!
  - Let the network build representations of words as features
- Word vectors are learned representations of words.
  - They can be learned for your specific problem
  - They can be generic or pretrained word vectors: trained on a large corpus

# Input Data

- Our input data has to always be the same length.
- Language is flexible, some sentences are long, some are short.
- To solve the problem, we decide on an input length, and any input longer we cut off at the end (or the beginning).
- If an input is too short, we add a special word called “<pad>.”
  - Yes, brackets are there on purpose.
- We will put the <pad> characters at the front of the sentence so we do not lose information.

# Similarity

- Previously, if we looked for structure, we looked for things that were “nearby” in terms of distance.
- With word vectors, the basic idea is that words that are alike or “nearby” are pointing in the same direction in vector space.
- Thus, if we look at the angle between vectors, this is a measure of their similarity.
  - Cosine similarity: 1—vectors are very similar. (If there is no angle between them, then cosine 0 is 1.)
  - Cosine distance: Because cosine can be negative and the concept of a negative distance is “illegal,” the cosine distance is 1 – cosine similarity.
    - Be aware of which is being used—they are both in common usage!

DataScience@SMU

# Building RNN for NLP

# Steps to Solve an NLP Problem

- Get our data and convert text to numbers
- Make all the data the same length
- Build the network
- Train and predict

# Text to Numbers

- A typical NLP problem is sentiment analysis. For instance, can we predict if a movie review is positive or negative?
- Start with movie reviews that have labels
  - Still supervised learning
  - Siskel and Ebert: thumbs up or down
  - Thumbs up 1, thumbs down 0 (typical class)
- Switch words to numbers
  - Each word gets a unique integer

# Text to Numbers (cont.)

- Examples
  - “The movie was the best I have seen”
  - “This movie was the worst”
- As numbers
  - “1 2 3 1 4 5 6 7”
  - “8 2 3 1 9”
- Typical schemes give popular words lower numbers
  - The scheme doesn’t matter
- Always good practice to include a word that represents an “unknown” word, in case during prediction a word not in the training set appears!

| Word | Number |
|---|---|
| the | 1 |
| movie | 2 |
| was | 3 |
| best | 4 |
| i | 5 |
| have | 6 |
| seen | 7 |
| this | 8 |
| worst | 9 |

# Make Data the Same Length

Make all the input data the same length

- “1 2 3 1 4 5 6 7”—length 8
- “8 2 3 1 9”—length 6!
  - “0 0 8 2 3 1 9”—length 8!
    - 0 is the number representing the pad
      - Usually the default
      - Can be any number: just make sure you are consistent!

# Build the Network

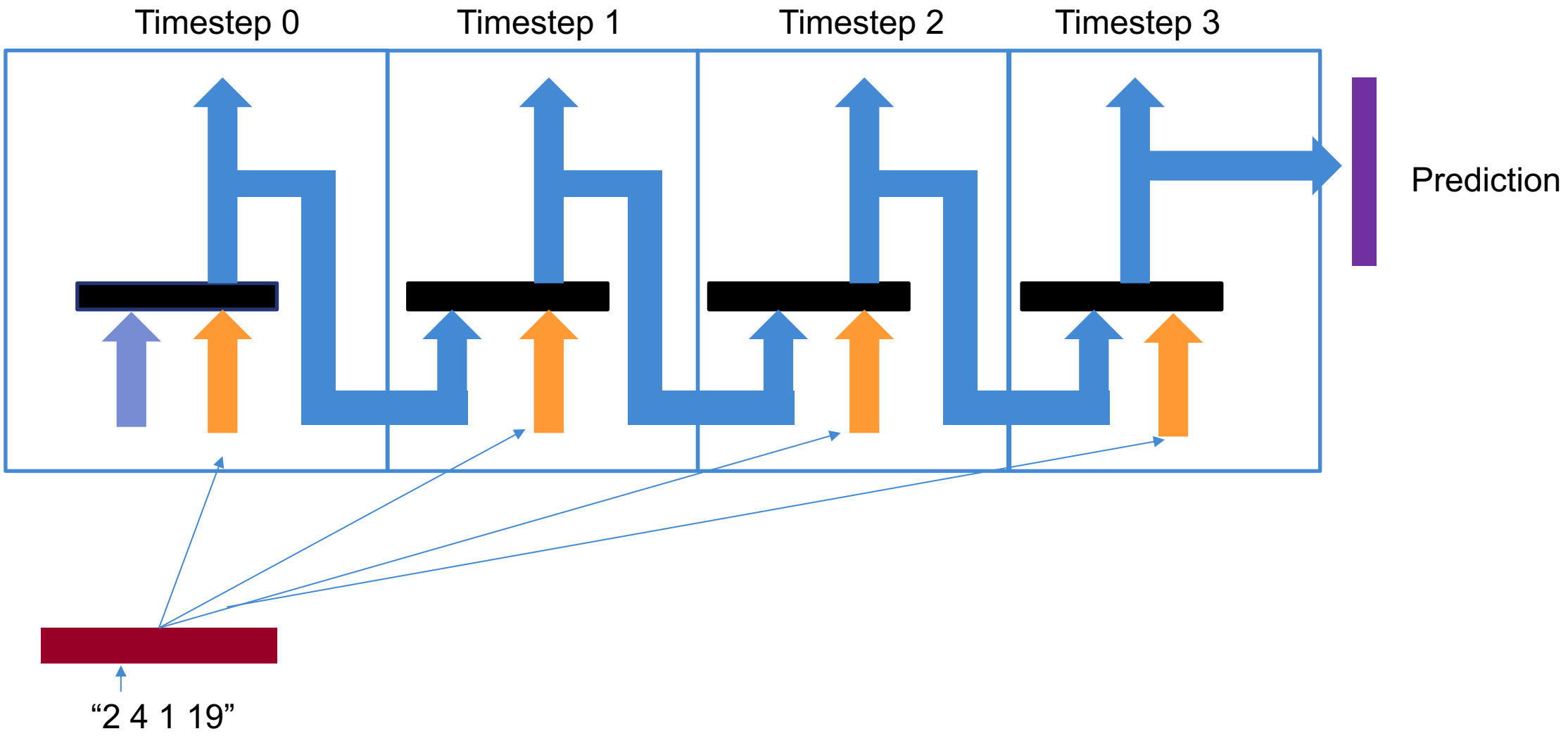
The network really consists of only 3 (yes 3!) layers.

- A layer that stores word representations (word vectors) called the embedding layer
  - The embedding layer **must** be the first layer.
- The recurrent layer
- A dense layer that makes a prediction

# Embedding Layer

- The embedding layer is a lookup layer.
  - It **could** be done as a traditional dense layer
  - It's **faster** to be done as a lookup
- Remember the word to numbers?
  - The number represents a row in a lookup table
  - 9—"get the weights in row 9"
  - The sequence "2 4 1 19" means get the numbers (vector) in row 2, then row 4, then row 1, then row 19
  - This is how the network learns word representations

# Remember This?



# RNN Flavors

- We just introduced the vanilla RNN.
- There are many types of RNN layers.
  - Most try and have more memory
  - Most popular are “long short-term memory” (LSTM) or “gated recurrence unit” (GRU)
  - They behave similar to RNNs, but are able to support longer “memory”
- RNNs have a very short memory.
  - They consist of about 5 steps
- Just like CNNs, the architecture of RNNs is an active area of study and rapidly changing. This is just a basic primer.
  - RNNs, in particular, get crazy **fast**
    - Lots of different related layers
  - CNNs are deep—but keep the basic layers

DataScience@SMU