RECURRENT NEURAL NETWORKS

INTRODUCTION

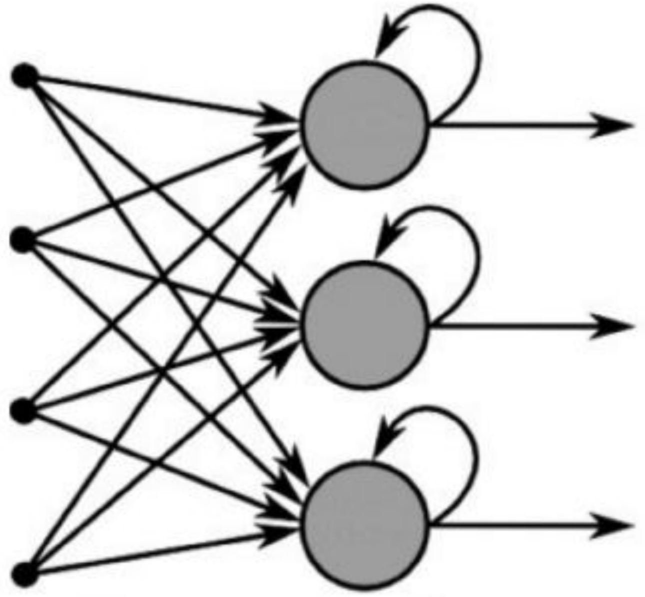
- Recurrent Neural Networks are in the family of feed-forward neural networks. They are **different from other feed-forward networks in their ability to send information over time-steps**.
- **Recurrent Neural Networks are similar to the human brain**, which is a large feedback network of connected neurons that somehow can learn to translate a lifelong sensory input stream into a sequence of useful motor outputs.
- **Recurrent Neural Networks take each vector from a sequence of input vectors and model them one at a time. This allows the network to retain state** while modeling each input vector across the window of input vectors.
- **Modeling the time dimension** is a hallmark of Recurrent Neural Networks.

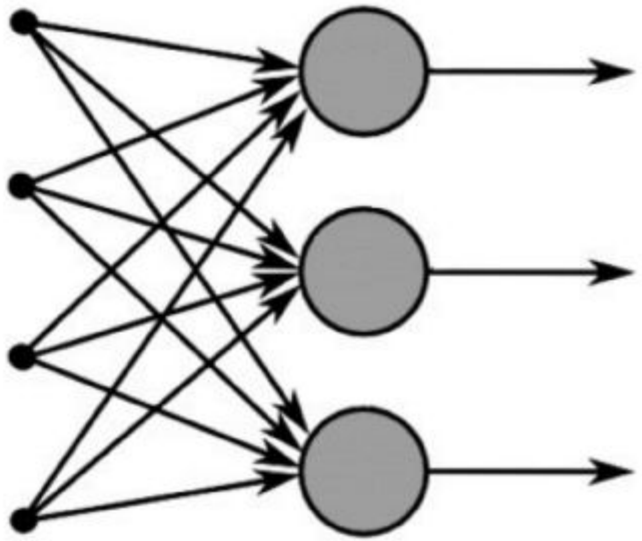
MODELLING THE TIME DIMENSION

- Many classification tools (support vector machines and regular feed-forward networks) have been applied **successfully without modeling the time dimension, assuming independence.**
- Other variations of these tools **capture the time dynamic by modeling a sliding window of the input** (e.g., the previous, current, and next input together as a single input vector).
- A drawback of these tools is that assuming independence in the time connection between model inputs **does not allow our model to capture long-range time dependencies**.
- **Sliding window techniques have a limited window width** and will fail to capture any effects larger than the fixed window size.
- **A well-trained Recurrent Neural Network could compete in Alan Turing's famed Turing Test**, for instance, which attempts to fool a human into thinking he is speaking with a real person.

MODELLING THE TIME DIMENSION

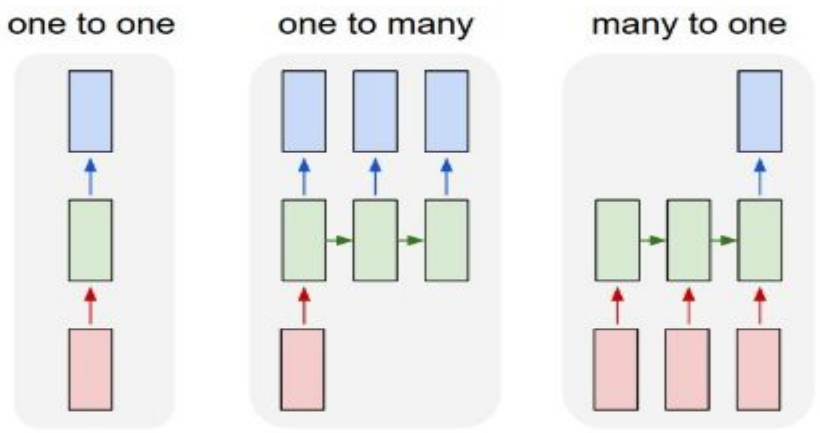
- Recurrent Neural Networks **can have loops in the connections.** This allows them to model temporal behavior gain accuracy in domains such as time-series, language, audio, and text.
- Recurrent Neural Networks **are trained to generate sequences**, in which the output at each time-step is based on both the current input and the input at all previous time steps.
- **Normal Recurrent Neural Networks compute a gradient with an algorithm called backpropagation through time (BPTT).**
- Recurrent Neural Networks **contrast with other deep networks** in what type of input they can model (non-fixed input):
  - Non-fixed computation steps
  - Non-fixed output size
  - It can operate over sequences of vectors, such as frames of video.


Recurrent Neural Network


Feed-Forward Neural Network

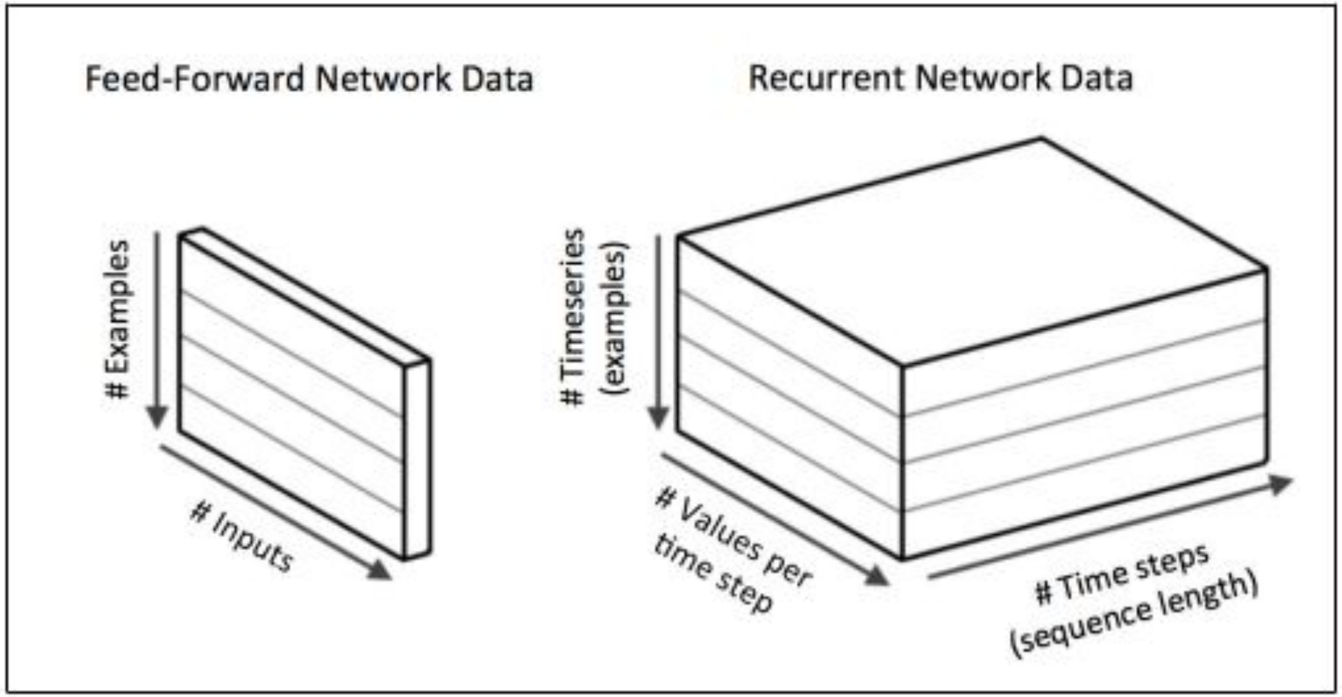
MODELLING THE TIME DIMENSION

- The following list gives examples of how Recurrent Neural Networks operate on sequences of input and output vectors:
  - **One-to-many:** sequence output. For example, **image captioning** takes an image and outputs a sequence of words.
  - **Many-to-one:** sequence input. For example, **sentiment analysis** where a given sentence is input.
  - **Many-to-many:** For example, **video classification: label each frame.**



REPRESENTATION OF INPUT DATA IN AN RNN

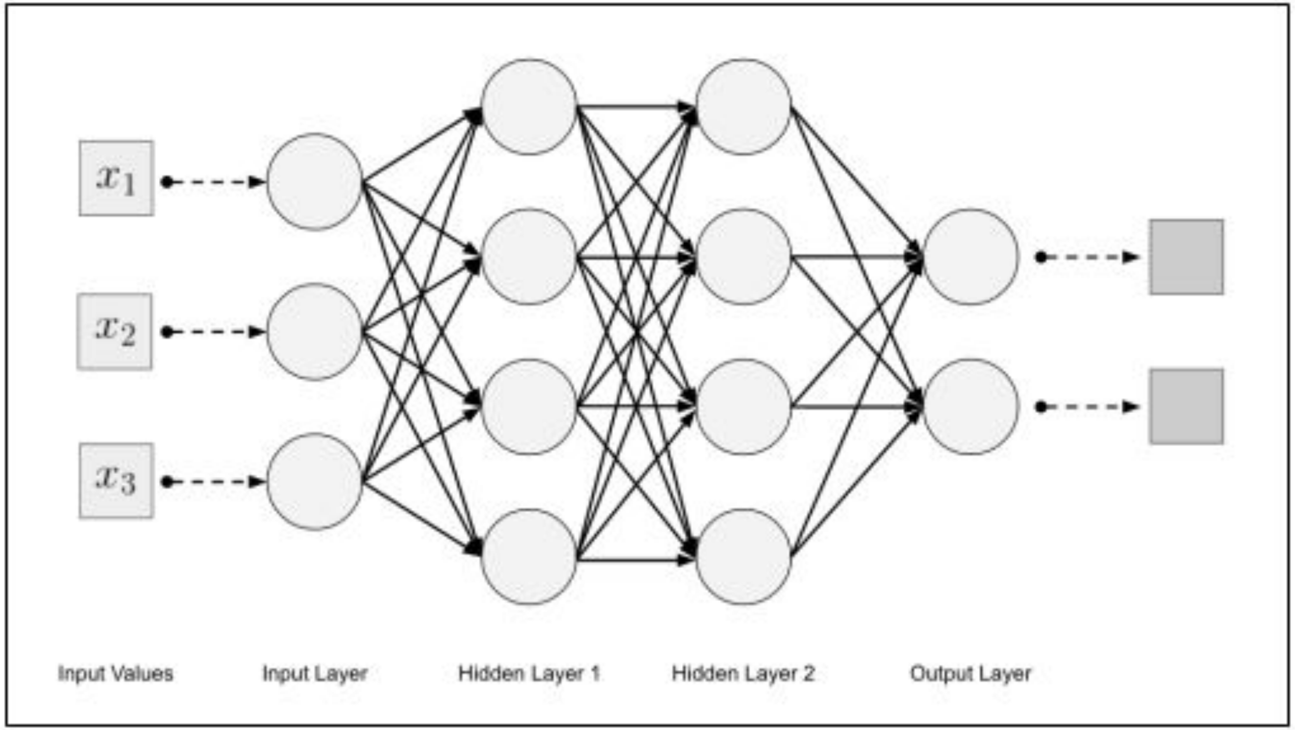
- We have **three dimensions** for the input:
    - Mini-batch size
    - Number of columns in our vector per time-step
    - Number of time-steps
- **Mini-batch size** is the number of input records (collections of time-series points for a single source entity) we want to model per batch.
- The **number of columns** matches up to the traditional feature column count found in a normal input vector.
- The **number of time-steps** is how we represent the change in the input vector over time. This is the time-series aspect to the input data.



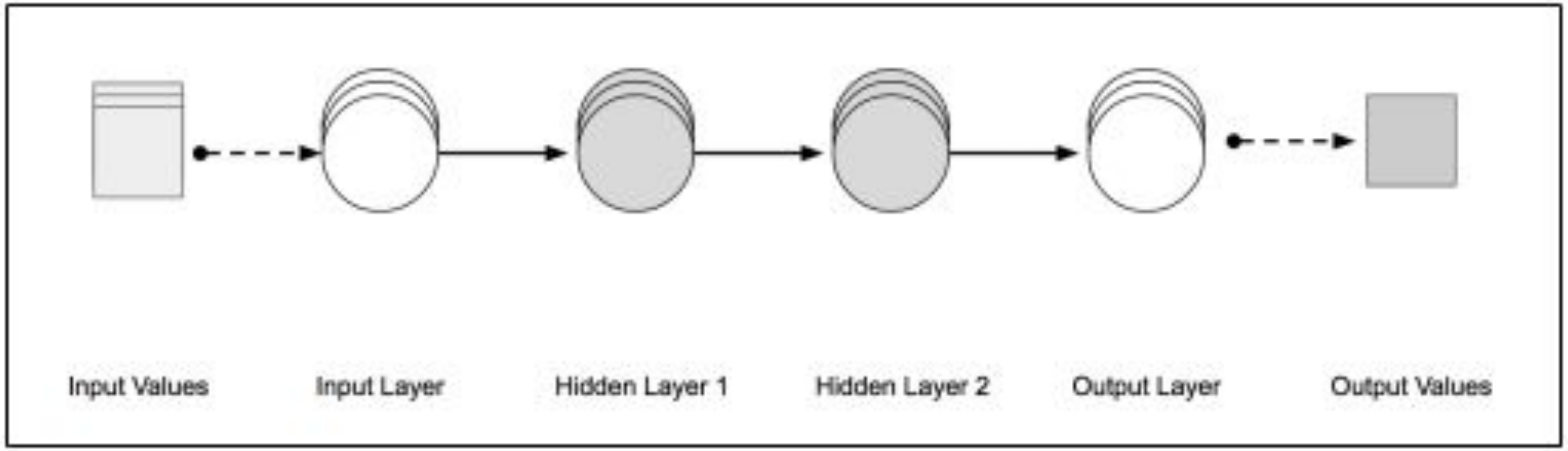
Normal input vectors compared to Recurrent Neural Networks input

GENERAL RECURRENT NEURAL NETWORK ARCHITECTURE

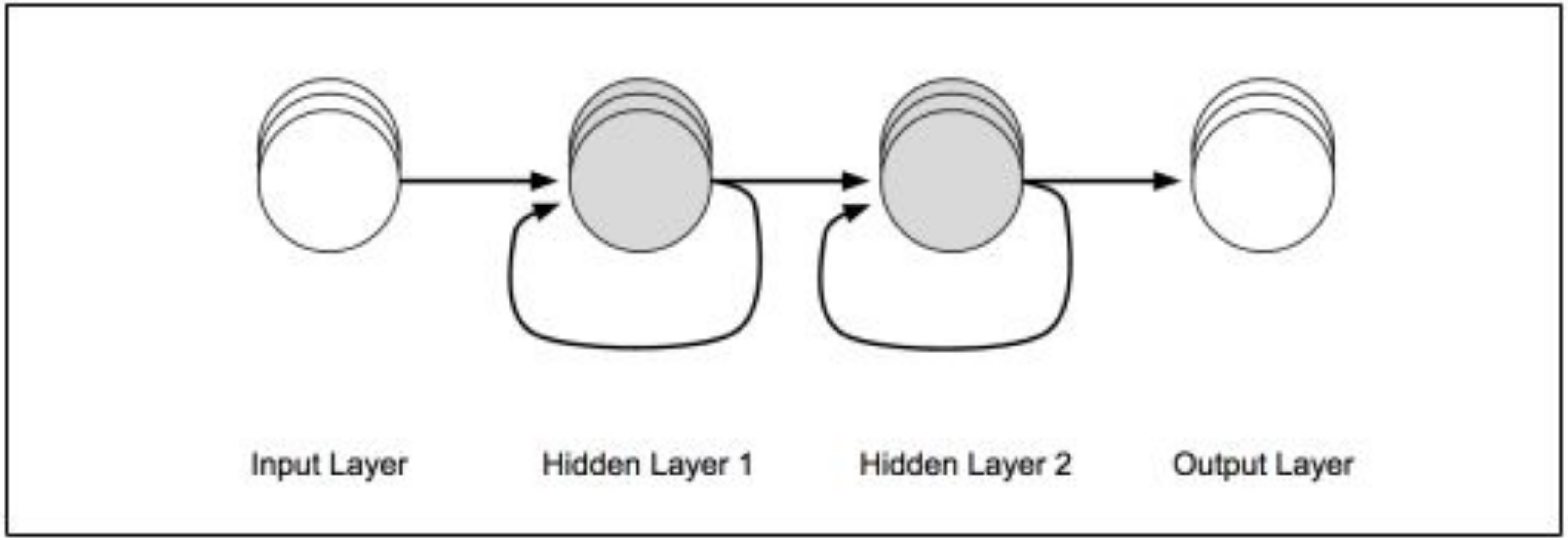
- Recurrent Neural Networks are **a superset of feed-forward neural networks** but they **add the concept of recurrent connections**.
- These **connections (or recurrent edges) span adjacent time-steps** (e.g., a previous time-step), giving the model the concept of time.
- The conventional connections do not contain cycles in recurrent neural networks. However, **recurrent connections can form cycles** including connections back to the original neurons themselves at future time-steps.
- The **previous input vector at the previous time step can influence the current output at the current time-step** through the recurrent connections.



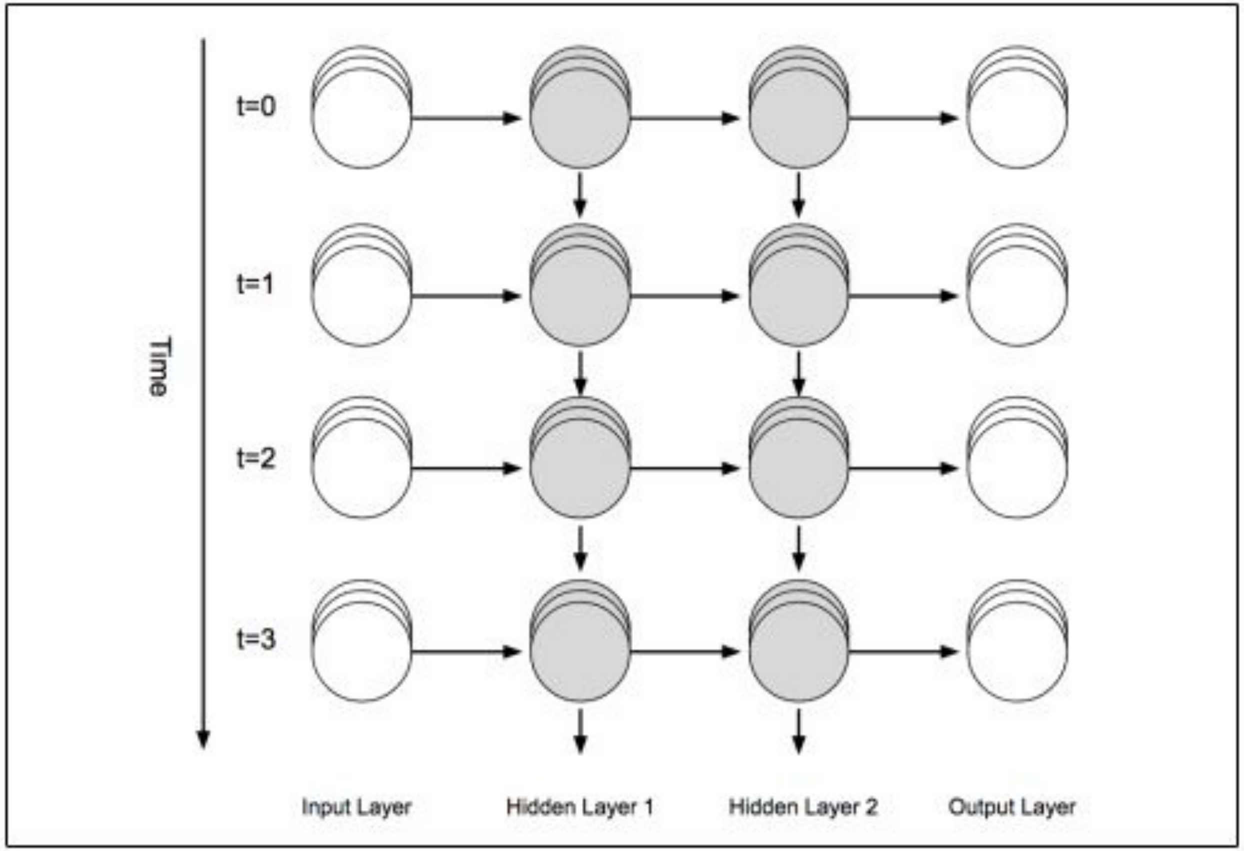
Feed-forward multilayer neural network architecture



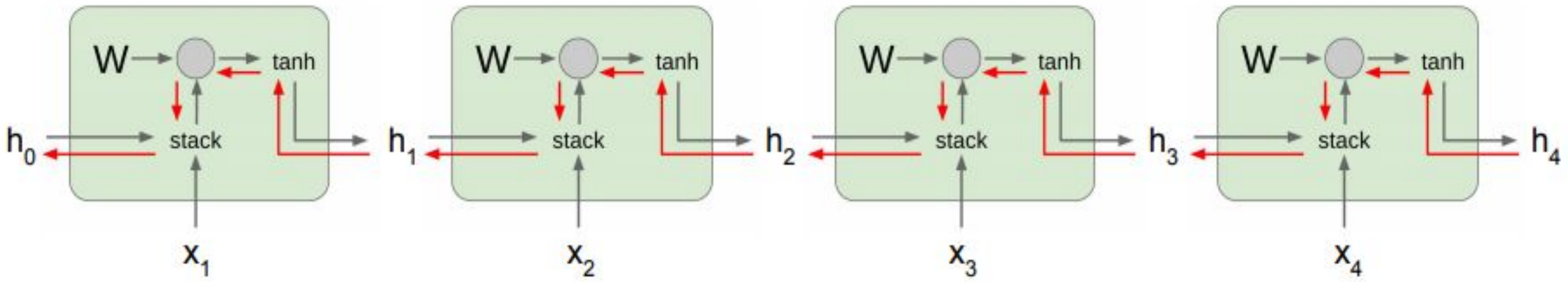
Flattened representation of a feed-forward multilayer network



Showing recurrent connections on hidden-layer nodes



Recurrent Neural Network unrolled along the time axis



Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1: **Exploding gradients** → **Gradient clipping**: Scale gradient if its norm is too big

```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
  grad *= (threshold / grad_norm)
```

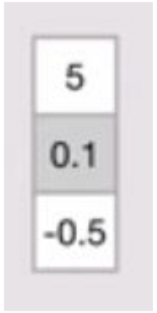
Largest singular value < 1: **Vanishing gradients**

LSTM NETWORKS

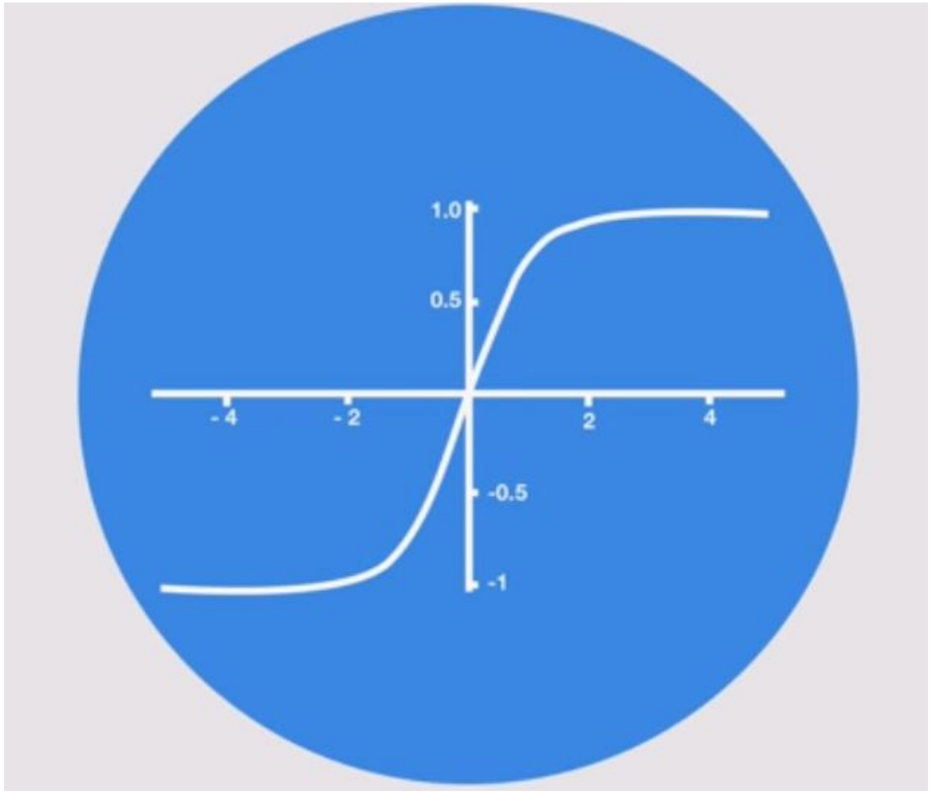
- LSTM networks are the most commonly used **variation of Recurrent Neural Networks.**
- The critical **components of the LSTM are the memory cell and the gates.**
- **The contents of the memory cell are modulated by the input gates and forget gates**. Assuming that both of these gates are closed, the contents of the memory cell will remain unmodified between one time-step and the next.
- The **gating structure allows information to be retained across many time-steps**, and consequently also **allows gradients to flow across many time-steps**. This **allows the LSTM model to overcome the vanishing gradient problem** that occurs with most Recurrent Neural Network models.
- LSTMs are known for the following:
  - Better update equations
  - Better backpropagation

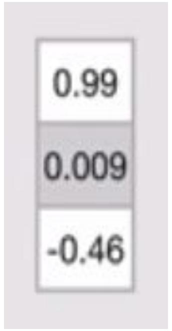
LSTM UNITS

- **Each LSTM unit has two types of connections**:
  - Connections from the previous time-step (outputs of those units)
  - Connections from the previous layer
- The memory cell in an LSTM network is the central concept that allows the network to maintain state over time. The main body of the LSTM unit is referred to as the **LSTM block.**
- The **3 gates** in an LSTM unit are:
  - The **input gate** protects the unit from irrelevant input events.
  - The **forget gate** helps the unit forget previous memory contents.
  - The **output gate** exposes the contents of the memory cell (or not) at the output of the LSTM unit.



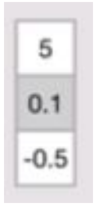
5
0.1
-0.5



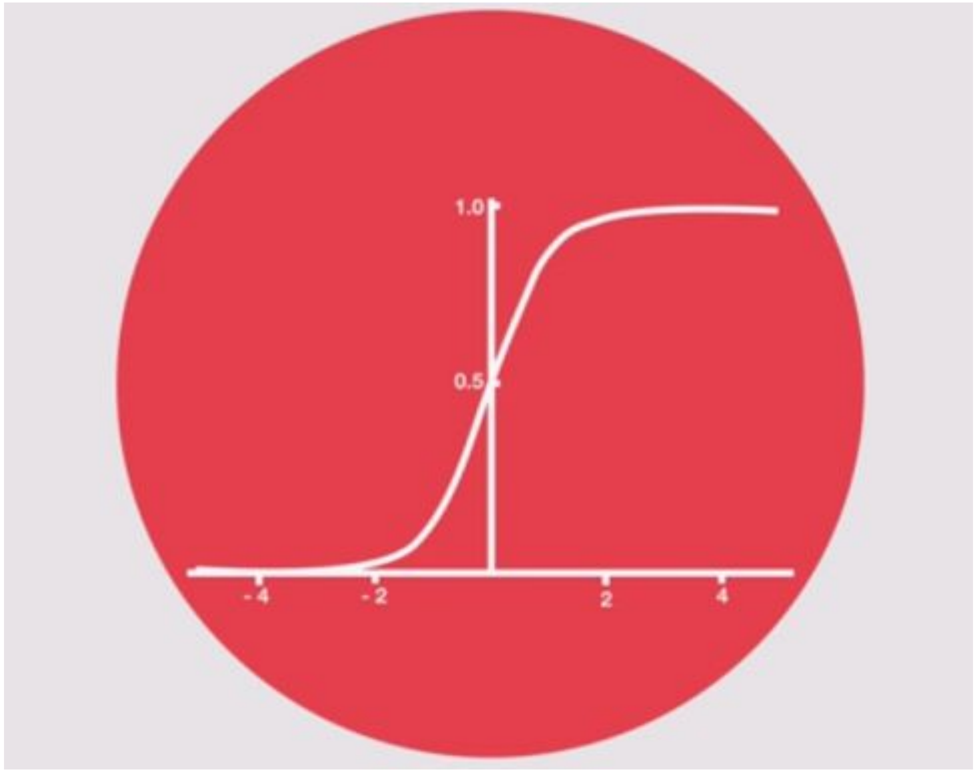


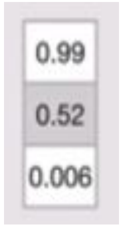
0.99
0.009
-0.46

tanh Activation function



5
0.1
-0.5



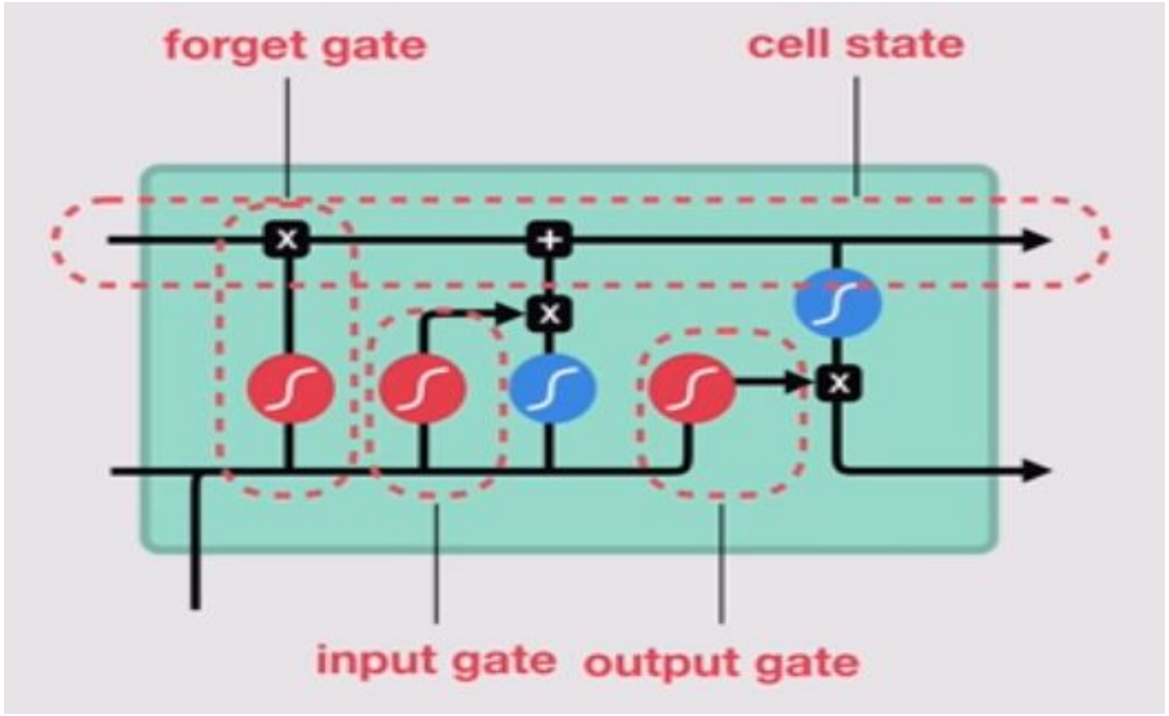


0.99
0.52
0.006

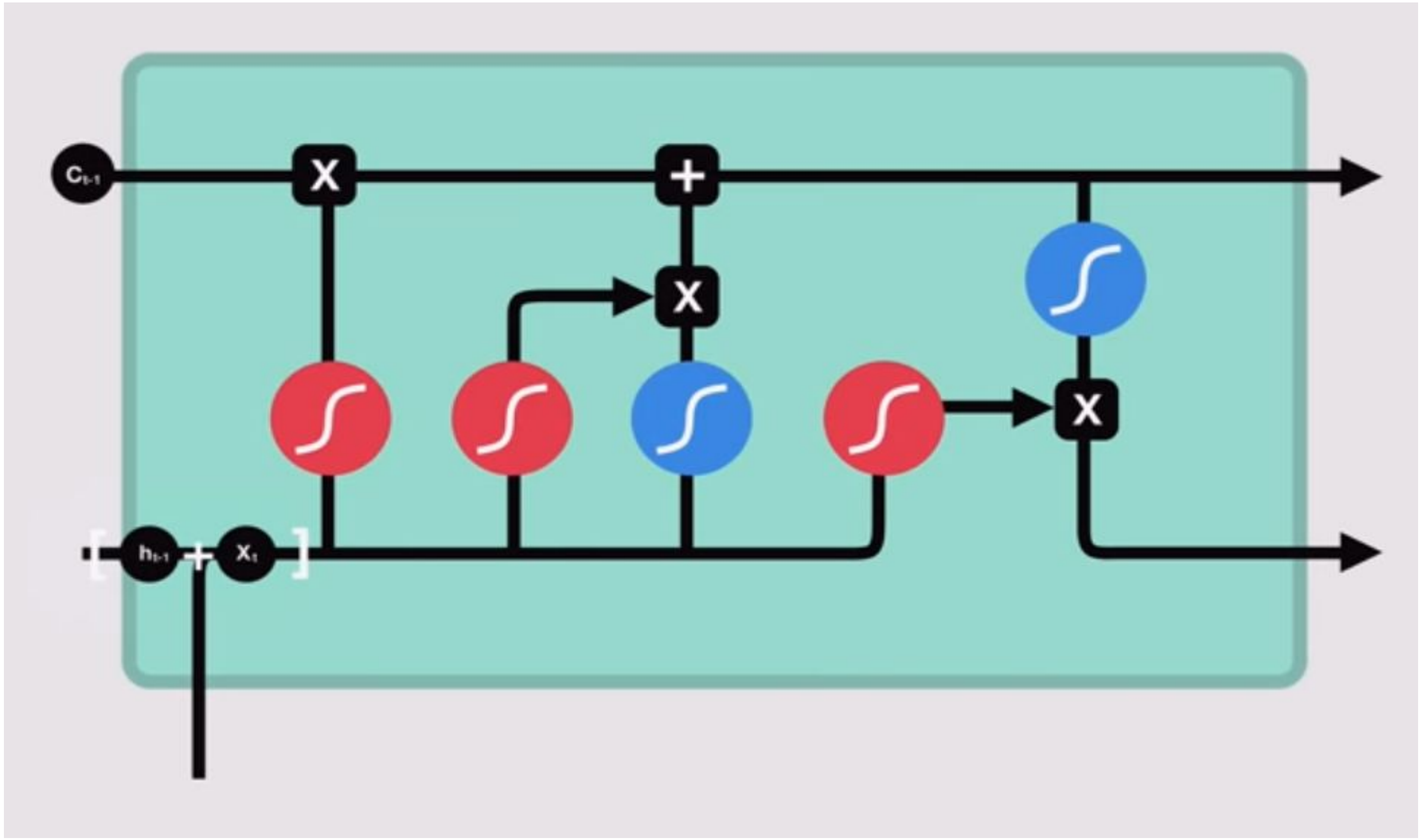
Sigmoid Activation function

LSTM UNITS

- The **output of the LSTM block is recurrent** and is hence connected back to the block input and all of the gates for the LSTM block.
- The **input, forget, and output gates in an LSTM unit have sigmoid activation functions** for [0, 1] restriction.
- An **activation output of 1.0 means "remember everything" and activation output of 0.0 means "forget everything."**
- With this in mind we usually **initialize the forget gate bias to a large value, such as 1.0 to enable learning of long-term dependencies**.
- The **LSTM block input and output activation function (usually) is a tanh activation function**.



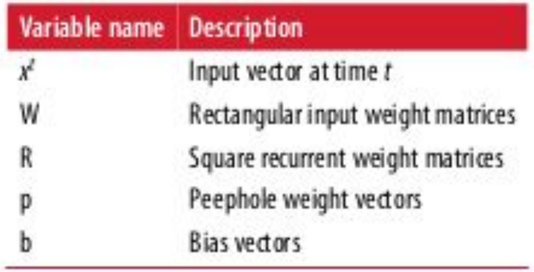
Long Short-Term Memory (LSTM)



Working of LSTM

$$
\begin{aligned}
\mathbf{z}^t &= g(\mathbf{W}_z\mathbf{x}^t + \mathbf{R}_z\mathbf{y}^{t-1} + \mathbf{b}_z) && \textit{block input}\\
\mathbf{i}^t &= \sigma(\mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i) && \textit{input gate}\\
\mathbf{f}^t &= \sigma(\mathbf{W}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{y}^{t-1} + \mathbf{p}_f \odot \mathbf{c}^{t-1} + \mathbf{b}_f) && \textit{forget gate}\\
\mathbf{c}^t &= \mathbf{i}^t \odot \mathbf{z}^t + \mathbf{f}^t \odot \mathbf{c}^{t-1} && \textit{cell state}\\
\mathbf{o}^t &= \sigma(\mathbf{W}_o\mathbf{x}^t + \mathbf{R}_o\mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^{t} + \mathbf{b}_o) && \textit{output gate}\\
\mathbf{y}^t &= \mathbf{o}^t \odot h(\mathbf{c}^t) && \textit{block output}
\end{aligned}
$$

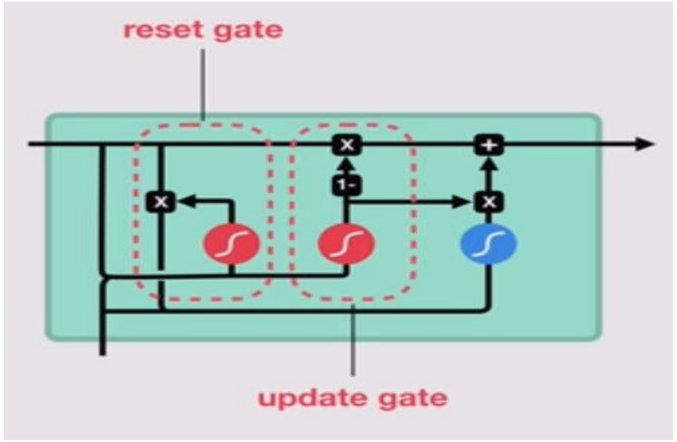
Vector formulas for LSTM layer forward pass



Variable name	Description
x^t	Input vector at time t
W	Rectangular input weight matrices
R	Square recurrent weight matrices
p	Peephole weight vectors
b	Bias vectors

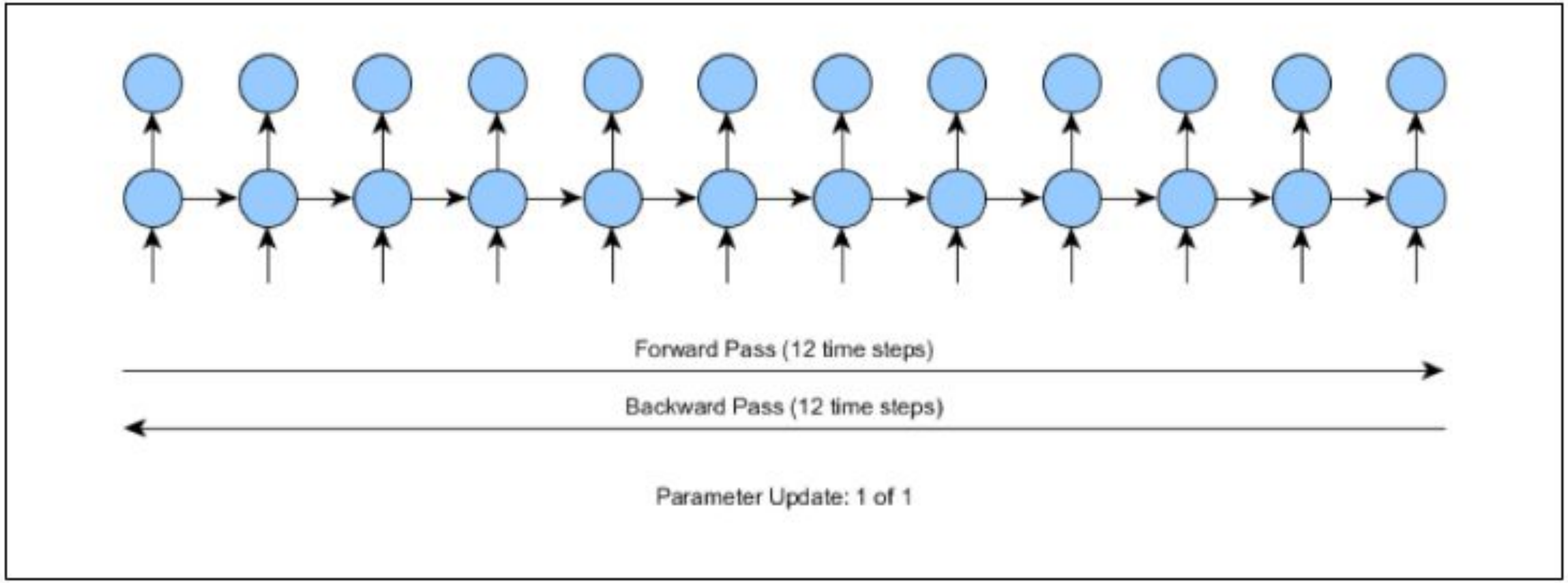
GATED RECURRENT UNITS (GRU)

- Another recurrent unit **similar to the LSTM** is the Gated Recurrent Unit (GRU). The GRU has **a reset gate and an update gate**, similar to the forget/input gates in the LSTM unit.
- The GRU was inspired by the LSTM unit but is considered **simpler to compute and implement.**



BACKPROPAGATION THROUGH TIME (BPTT)

- Recurrent Neural Network **training can be computationally expensive. The traditional option is to use BPTT.**
- BPTT is fundamentally the same as standard backpropagation: we apply the chain rule to work out the derivatives (gradients) based on the connection structure of the network. **It's through time in the sense that some of those gradients/error signals will also flow backward from future time-steps to current time-steps**, not just from the layer above (as occurs in standard backpropagation).



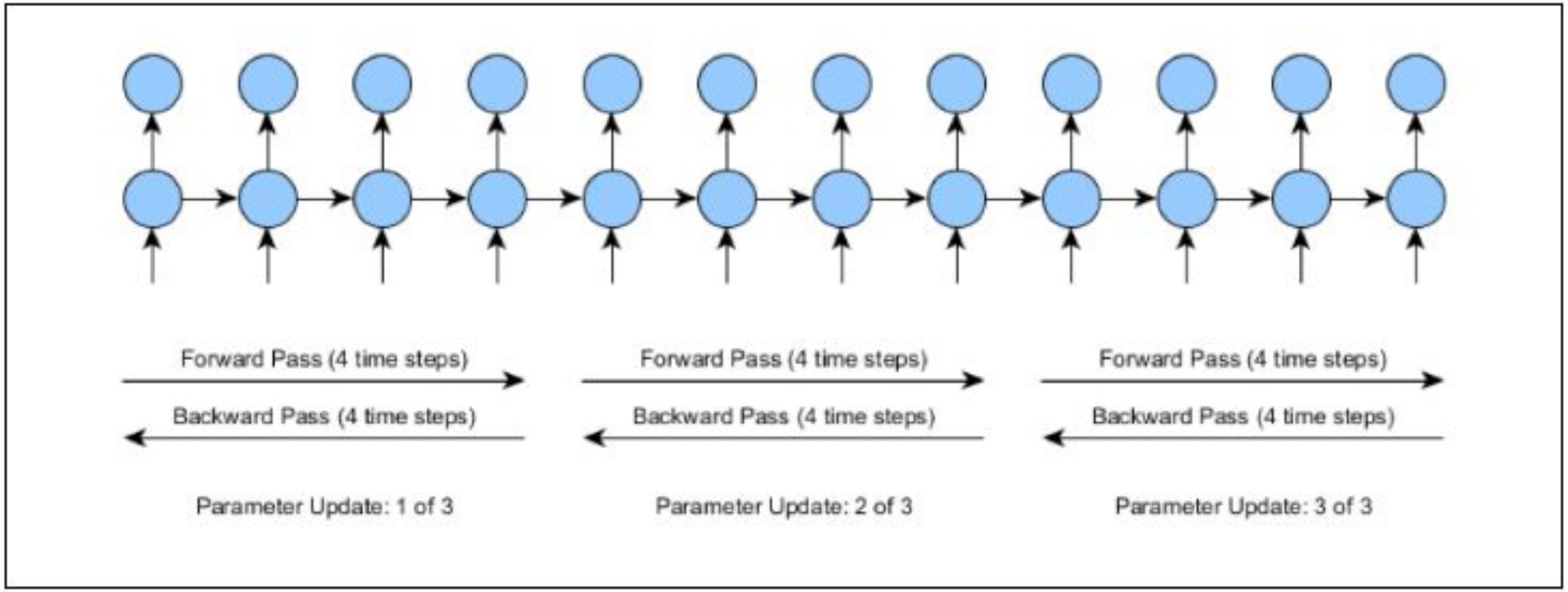
Standard BPTT

TRUNCATED BPTT

- When our recurrent network is **dealing with long sequences with many time-steps**, we recommend alternatively using **truncated BPTT**.
- Truncated BPTT **reduces the computational complexity of each parameter update** in a Recurrent Neural Network. Performing more frequent parameter updates speeds up recurrent neural network training.
- Truncated BPTT is recommended when the **input sequences are more than a few hundred time-steps.**
- Truncated BPTT is considered the current most practical method for training Recurrent Neural Networks.
- With truncated BPTT, we can capture long-term dependencies with less computational burden than regular BPTT. The overall complexity for standard BPTT and truncated BPTT is similar and requires the same number of time-steps during training. However, **we get more parameter updates for about the same amount of computational effort** (although there is some overhead for each parameter update).

TRUNCATED BPTT

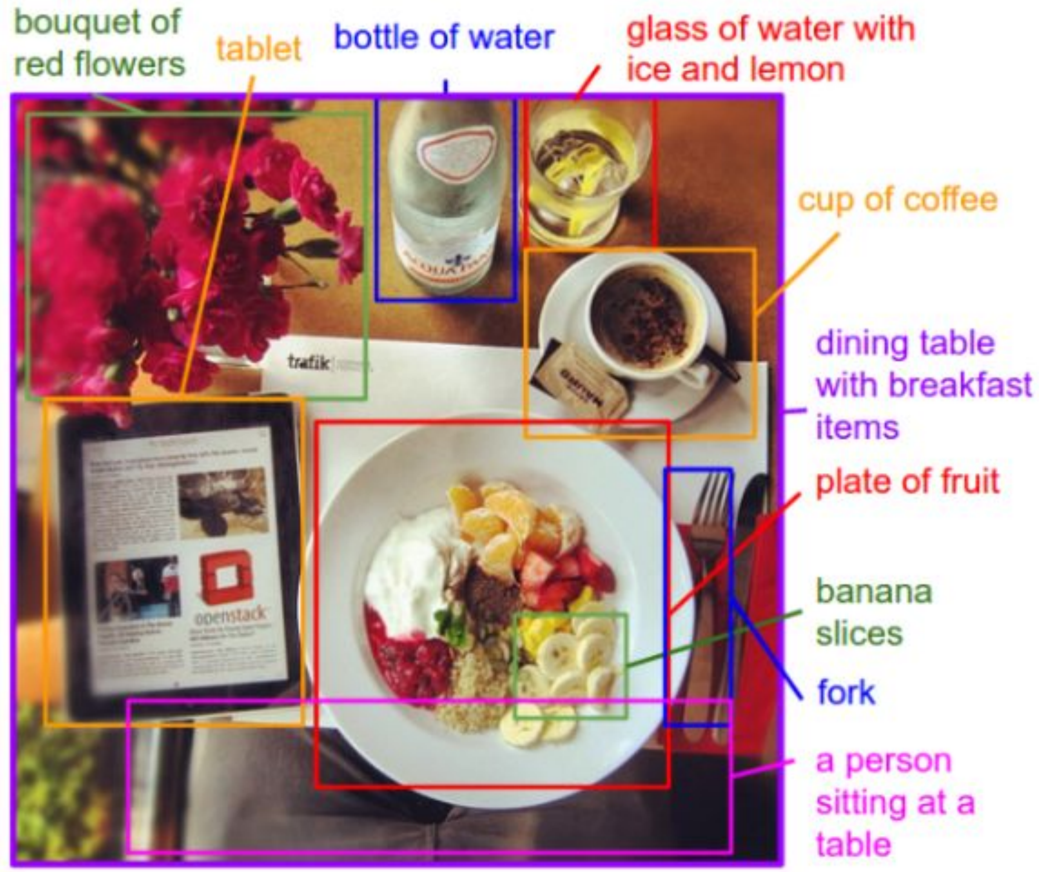
- As with all approximations, we see a slight downside when using truncated BPTT; **the length of the dependencies learned in truncated BPTT can end up being shorter than the full BPTT algorithm results.**
- **In practice, this speed trade-off is generally worth it as long as truncated BPTT lengths are set correctly.**


Forward Pass (4 time steps)
Backward Pass (4 time steps)
Parameter Update: 1 of 3
Forward Pass (4 time steps)
Backward Pass (4 time steps)
Parameter Update: 2 of 3
Forward Pass (4 time steps)
Backward Pass (4 time steps)
Parameter Update: 3 of 3

Truncated BPTT

BLENDED NETWORKS

- An emerging area of research in computer vision with Recurrent Neural Networks is a **network that can extract information from an image by processing only a small region at a time and is called Recurrent Models of Visual Attention.**
- These models are e**ffective at working with images that are cluttered with multiple objects** and are more difficult for CNNs to classify.
- Another hybrid CNN + RNN network of note is one in which **the network generates natural-language descriptions of images and their regions.** The model is able to generate captions for images by using datasets of images and their corresponding sentence.


bouquet of red flowers
tablet
bottle of water
glass of water with ice and lemon
cup of coffee
dining table with breakfast items
plate of fruit
banana slices
fork
a person sitting at a table

Labeling images with a blended CNN/Recurrent Neural Network