# CPE 695: Applied Machine Learning

## *Lecture 13: Introduction to Deep Learning (2)*

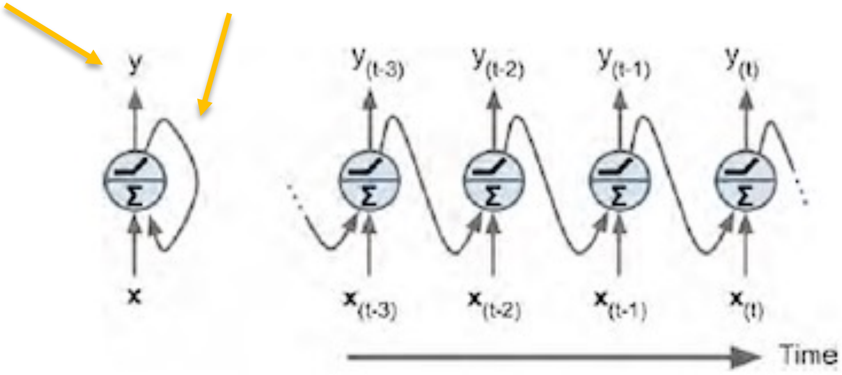
Dr. Shucheng Yu, Associate Professor
Department of Electrical and Computer Engineering
Stevens Institute of Technology

# Predicting the Future

- Network models introduced in previous modules do not consider sequential data.
- There are many real-world applications requiring to *predict the future* based on time-series data.
    - E.g., stock price prediction, car trajectory prediction, signal estimation.
- Recurrent neurons can process time-series data as previously discussed

Output

State



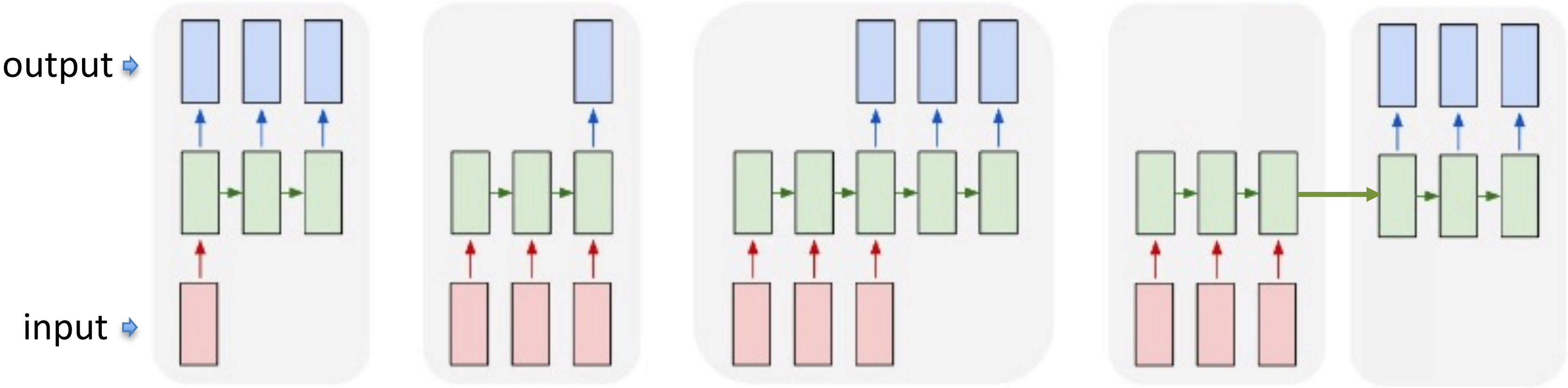
One recurrent neuron (left) unrolled through time (right)

[Geron Textbook]

- Training is like regular backpropagation, but over unrolled model.
- The strategy is called **backpropagation through time (BPTT)**

# Input and Output Sequences

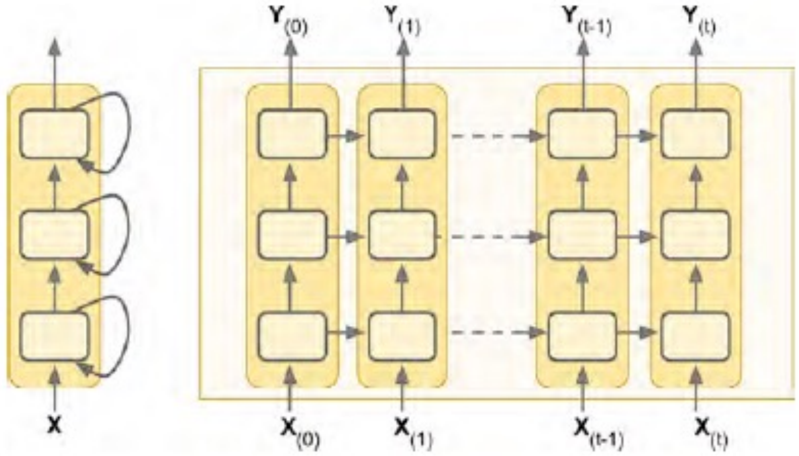
- An RNN can have different structure based on inputs-outputs
  - *Vector-to-sequence*: a vector of inputs output a sequence of outputs.
  - *Sequence-to-vector*: a sequence of inputs output a vector.
  - *Sequence-to-sequence*: a sequence of inputs produce a sequence of outputs.
  - *Encoder-Decoder*: sequence-to-vector followed by vector-to-sequence
  - Many other structures.



v-to-seq seq-to-v seq-to-seq Encoder-Decoder

# Deep RNNs

- For complex tasks, deep RNNs are needed.
  - Stack multiple layers of RNN neurons.
  - Training long sequences results in deep unrolled RNN (over time)



Deep RNN (left) unrolled through time (right)

[Geron Textbook]

Challenges:

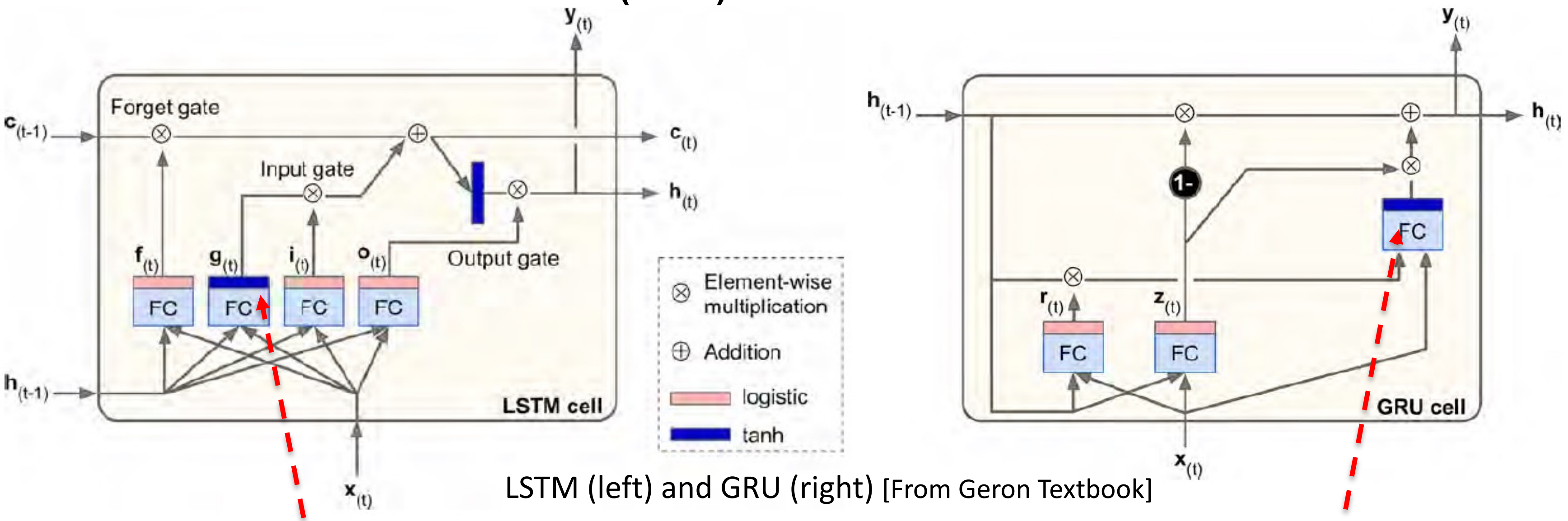
- **Unstable gradients**
- **Short-term memory**

# Unstable Gradients Problem

- Similar to feedforward DNNs, deep RNNs suffer from unstable gradients
  - Most tricks applied in DNNs are still applicable
    - Appropriate parameter initialization
    - Fast optimizers
    - Dropout
    - …
- However, some important techniques are not applicable
  - Non-saturating activation functions do not help
    - Repeatedly adding updates using the gradient descents of same weights unfolded over time will cause outputs to explode
    - On the contrary, deep RNNs favor saturating activation functions, e.g., sigmoid function, hyperbolic tangent function.
  - Batch normalization is not helpful either
    - Layer normalization is used in RNNs.
      - It normalizes across feature dimension instead of batch dimension.

# Dealing with Short-Term Memory

- RNNs tend to forget information at early time steps when traversing
- Two important techniques to enable long-term memory
  - The **Long Short-Term Memory (LSTM)** cell
  - The **Gated Recurrent Unit (GRU)** cell



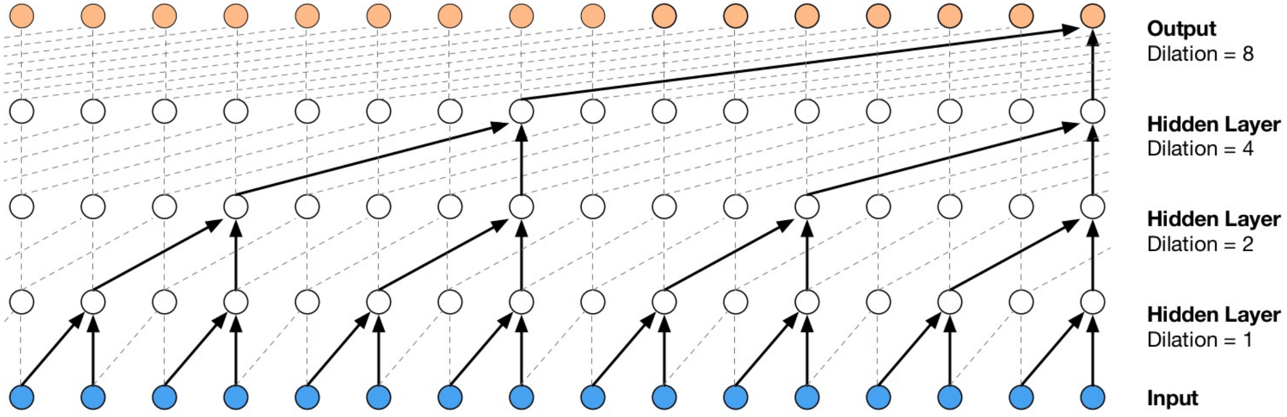
LSTM (left) and GRU (right) [From Geron Textbook]

- $g_{(t)}$ is the main layer for current inputs and previous state, using *tanh* as activation function.
- All other layers are gate controllers, using *logistic (sigmoid)* as activation function:

  0 to close the gate, 1 to open the gate

# Dealing with Long Sequences

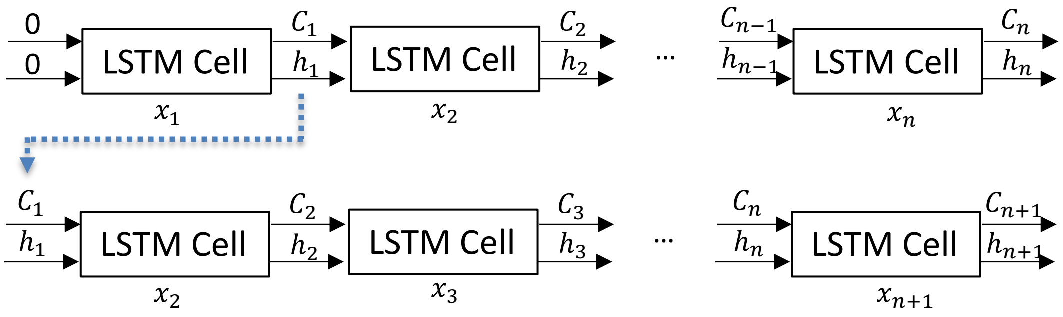
- LSTM and GRU still have quite limited short-term memory
- Techniques for very long sequences
  - **1D convolutional layer**
    - Detect short sequence patterns from long sequences like 2D convolutional layers
  - **WaveNet**
    - Stack 1D convolutional layers & double dilation rate at each layer
    - **Low layers** learn short-term patterns, **high layers** learn long-term patterns.



Source: http://www.gabormelli.com

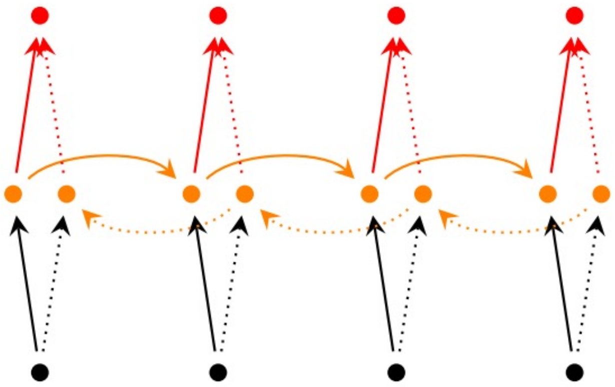
# Other RNN Techniques

- **Stateful RNN**
  - Depth of deep DNN determined by input sequence size
  - Long sequences (e.g., book) shall be chopped into shorter batches of sequences for RNN training
  - In stateless RNN (as discussed so far) does not preserve the state information of current iteration for the next iteration.
  - Stateful RNN preserves state of one batch and feeds it as input to the next training batch for the corresponding time step.



# Other RNN Techniques

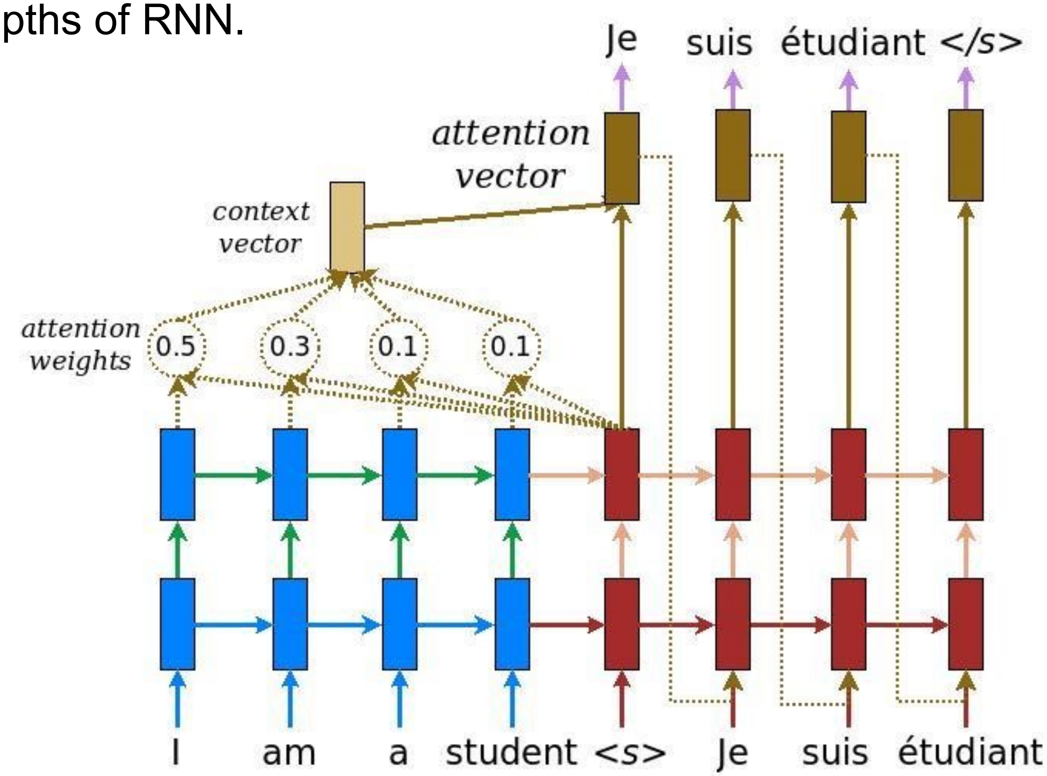
- **Bidirectional RNN**
  - In many tasks (e.g., NLP), encoding of an input depends on not only past and current inputs, but also future inputs.
  - This can be achieved by bidirectional RNN
    - Simple bidirectional RNN can be constructed by two “copies” of two identical RNNs, one reading from past to current (right to left in text), the other from future to current (left to right in text).
    - Outputs of the two RNNs at each time step are simply concatenated.



*http://www.wildml.com*

# Other RNN Techniques

- **Attention Mechanism**
  - RNNs usually need to preserve representation of inputs for long time steps for tasks like NLP.
  - Long time steps increases depths of RNN.
  - Attention mechanism can let RNN cells to focus on most related inputs instead of all and improves its performance
  - It becomes a very powerful tool in many applications.
  - Two attention mechanisms
    - Concatenative attention
    - Multiplicative attention



*Attention Mechanism in Encoder-Decoder*

*http://www.medium.com*

# Other Models for NLP

- **Transformer**
  - By Google, “Attention Is All You Need”, 2017
  - Attention-only architecture without any recurrent or convolutional layers
  - Faster and easier to parallelize than previous models
- **ELMo**
  - By M. Peters et al., “Deep Contextualized Word Representation”, 2018
- **ULMFiT**
  - By J. Howard and S. Ruder, “Universal Language Model Fine-Tuning for Text Classification”, 2018
- **GPT**
  - By OpenAI, “Improving Language Understanding by Generative Pre-Training”, 2018
- **BERT**
  - By Google, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, 2019
- More to come …