Modeling Sequential Data with 3D CNNs and Recurrent Neural Networks

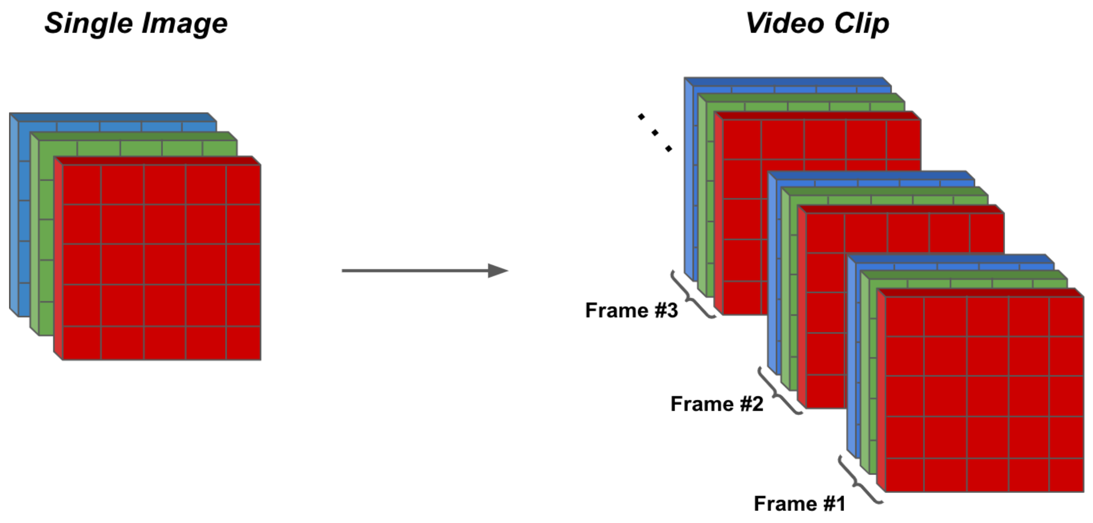
A Transition From Feedforward Neural Networks

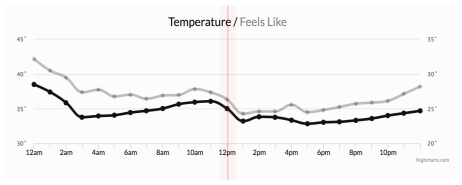
Outline

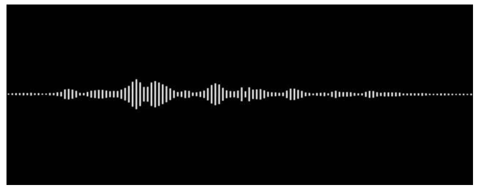
- What is Sequential Data?
- Modeling Sequential Data with Feedforward Neural Networks and 2D CNNs
- 3D Convolutional Neural Networks
- Recurrent Neural Networks
- Long Short-term Memory, LSTM
- Gated Recurrent Unit, GRU
- Combining CNNs and RNNs
- Conclusion

What is Sequential Data?

- Data points have **ordered structure**.
- Data points have **temporal dependency**.
- **Examples:**
  - Video data is a sequence of images (frames).
  - The number of frames that are displayed every second is refered to as FPS.

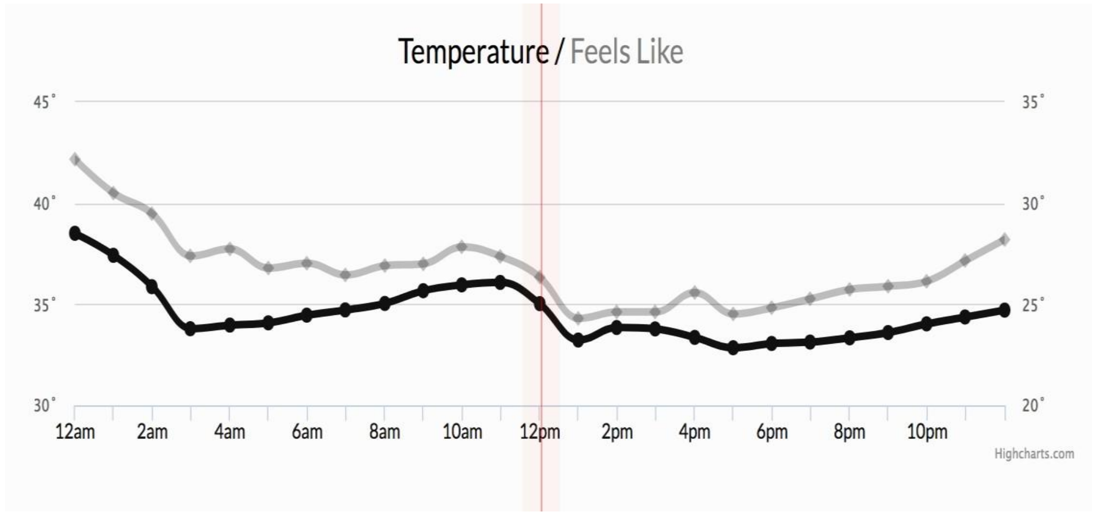


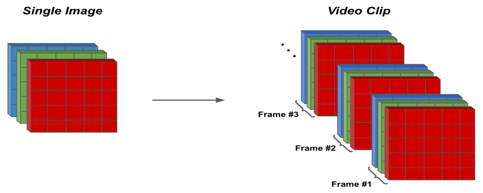


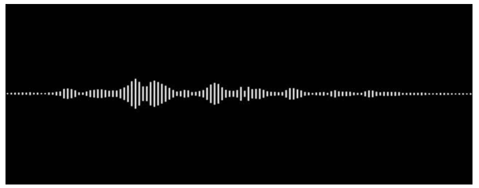


What is Sequential Data?

- Data points have **ordered structure**.
- Data points have **temporal dependency**.
- **Examples:**
  - Time series data is collection o f measurements recorded at specific points in time.
  - Measurements are ordered chronologically.

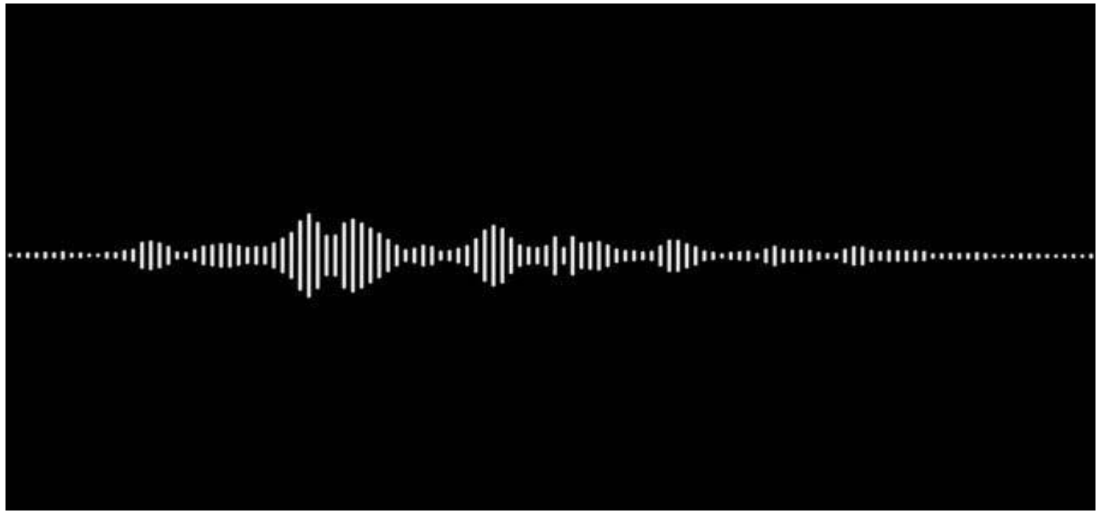


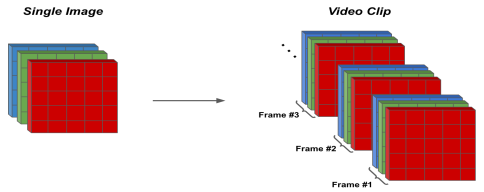


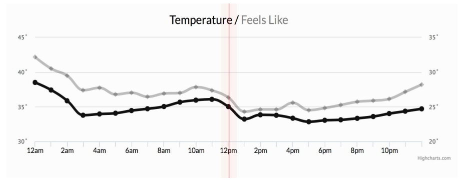


What is Sequential Data?

- Data points have **ordered structure**.
- Data points have **temporal dependency**.
- **Examples:**
  - Video data
  - Time series data
  - Text data
  - Audio data

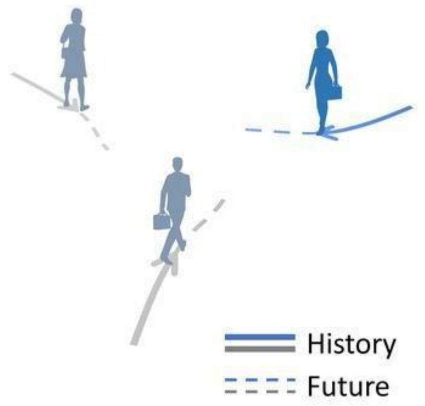






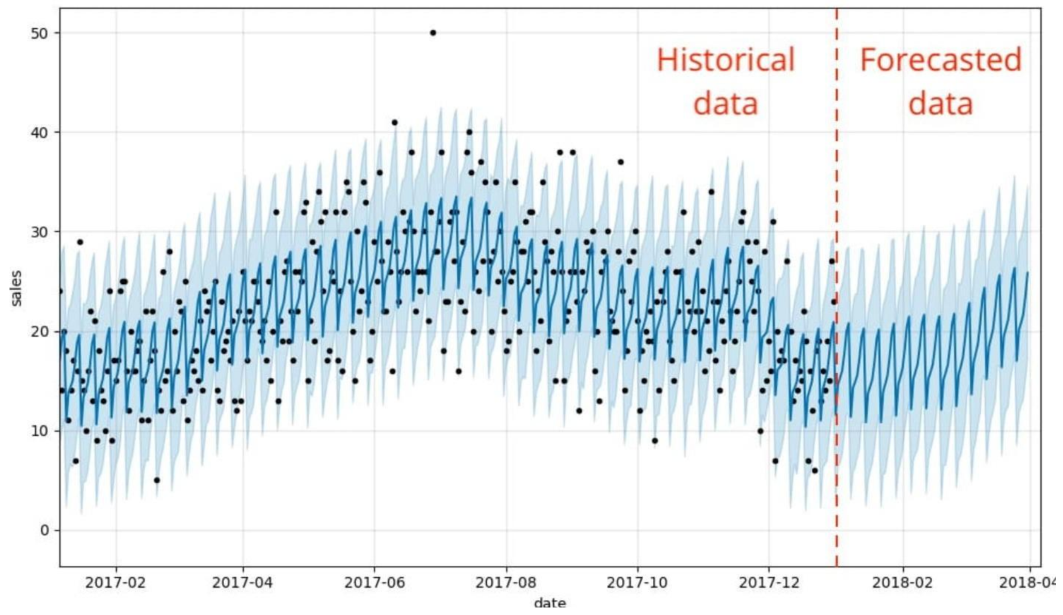
Applications of Modeling Sequential Data

- **Video analysis**
  - Human activity recognition
    - Surveillance systems to detect unusual behaviors
    - Analyzing athletes' movements for performance improvement
  - Trajectory prediction
    - Safe navigation in autonomous vehicles



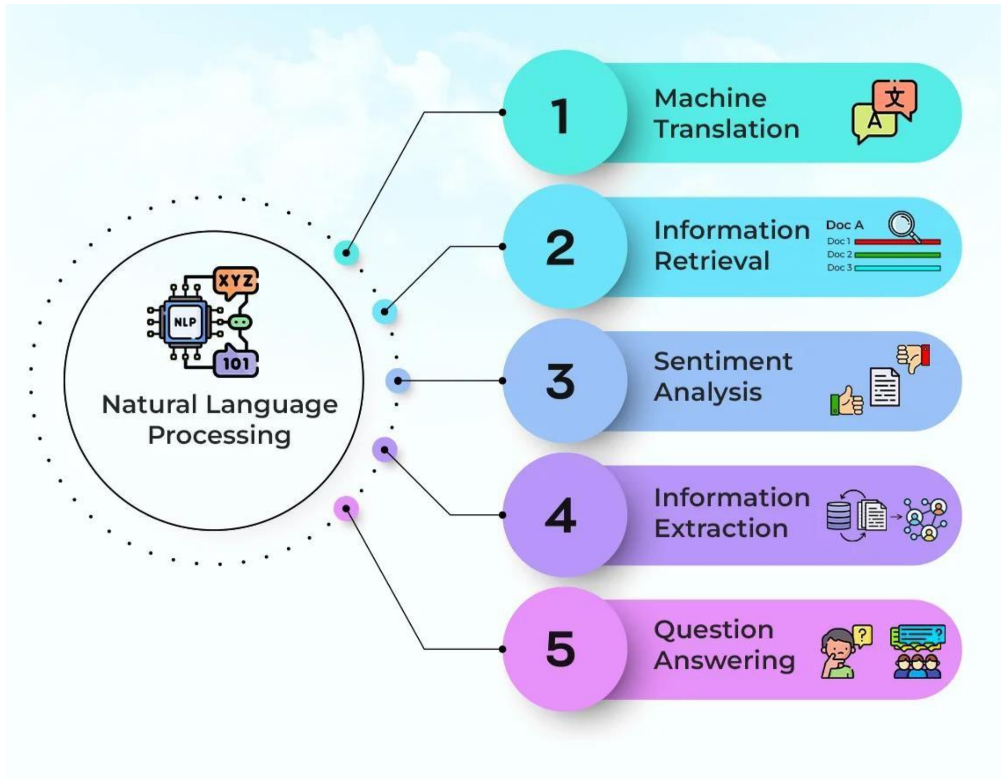
Applications of Modeling Sequential Data

- **Forecasting:** Making predictions about the future based on information about the past and present
  - Weather
  - Sales



Applications of Modeling Sequential Data

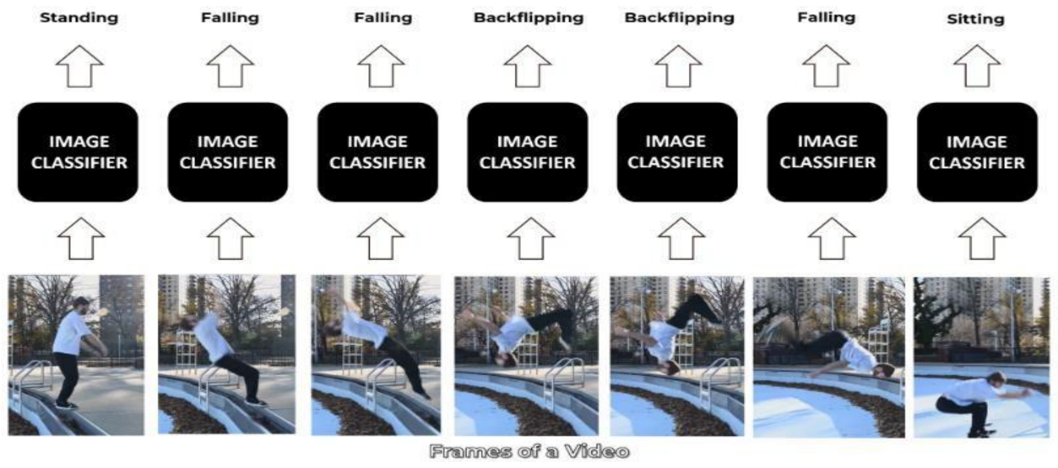
- **Natural language processing:** Getting computers to understand human language.
    - Language translation
    - Sentiment analysis
    - Text generation
    - Text summarization



Modeling Sequential Data with Feedforward Neural Networks and 2D CNNs

- **Frame by Frame Processing:**
    By extracting individual frames from the video and processing each frame independently.
- **Limitations:**
    Cannot model dynamic changes across different frames.

    Cannot capture information about previous frames which is crucial in modeling sequential data.



Modeling Sequential Data with Feedforward Neural Networks and 2D CNNs

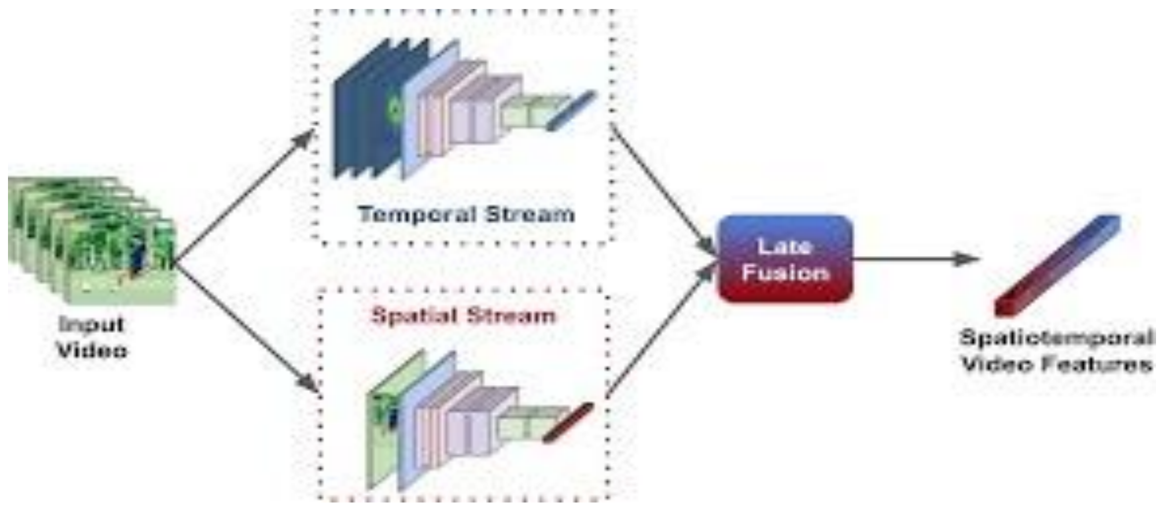
- **Two stream networks:**

  Uses two separate networks to analyze the video. One network captures spatial features and the other captures temporal features. Then both these features are fused together for downstream tasks.

- **Limitations:**

  It requires lots of data, video as well as optical flow images, and therefore is also computationally expensive.

  May struggle with scenes where motion alone isn't enough to describe what is happening.



Modeling Sequential Data with Feedforward Neural Networks and 2D CNNs

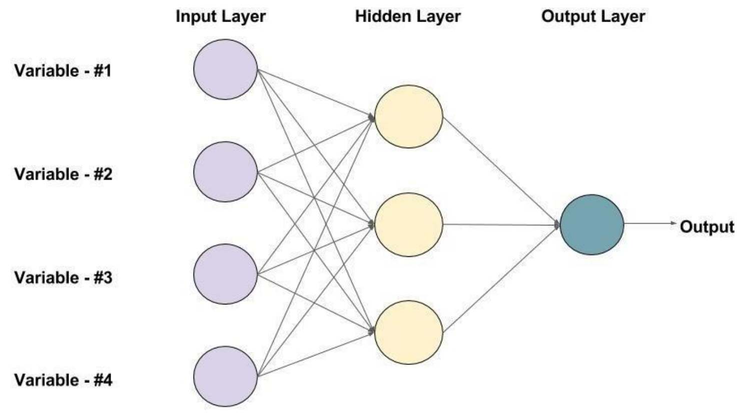
- **Sliding window approach:**

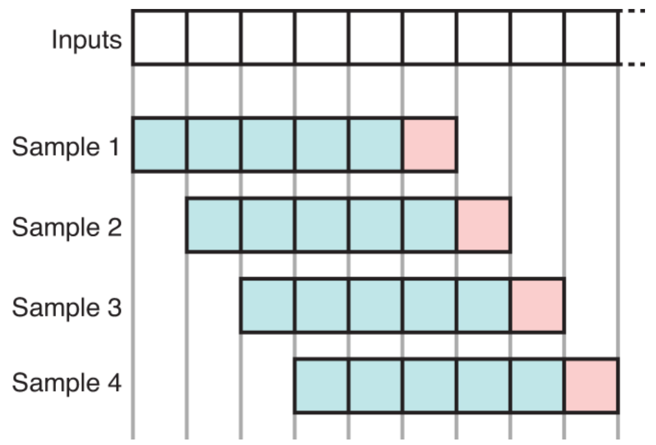
  It involves creating a window of fixed size that "slides" over the data set, one step at a time, to generate smaller segments of the data for analysis.

- **Limitations:**

  The fixed window size cannot capture longer trends and pattern.

  It is computationally expensive for large datasets.





3D Convolutional Neural Networks

- **2D Convolution:**

  The convolutional kernel/filter is applied across two dimensions.

  Works very well with two dimensional input, like images, to detect spatial features.

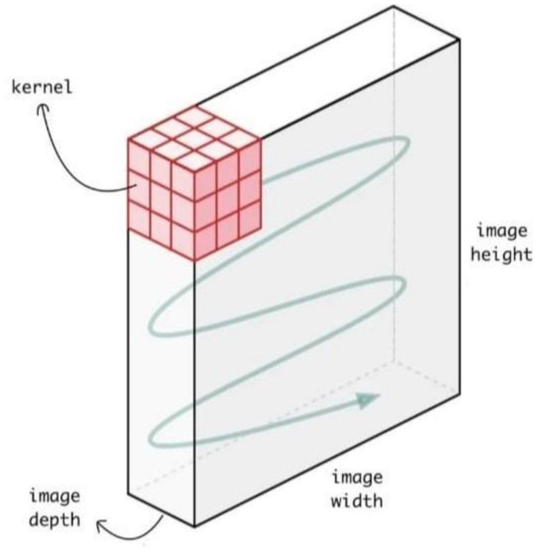
  Each kernel produces a 2D feature map.

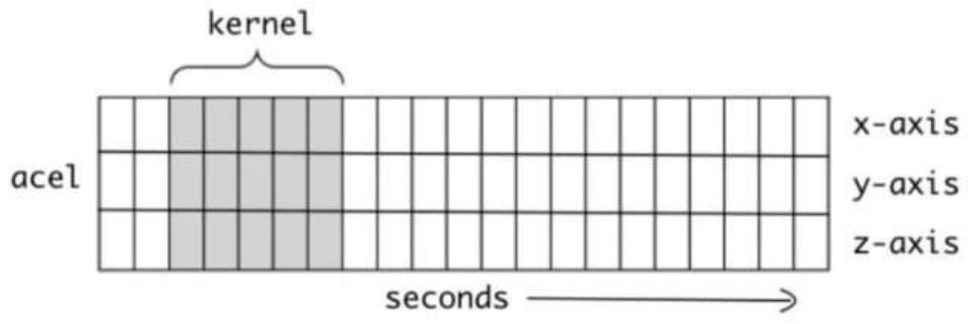
- **1D Convolution:**

  The convolutional kernel/filter is applied across a single dimension.

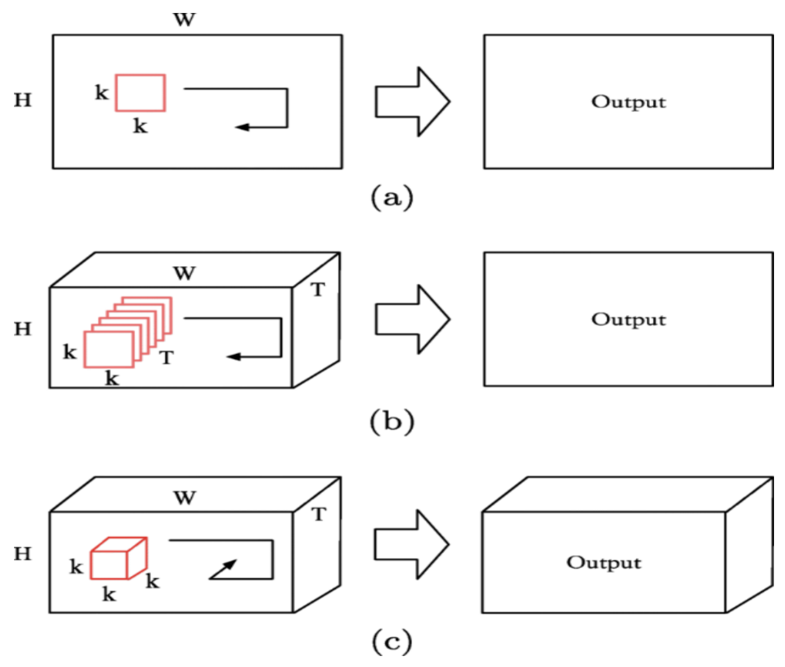
  Used when the input is one dimensional like time series or a sequential data to capture trends.

  Here, each kernel produces a 1D feature map.





3D Convolutional Neural Networks



- **3D Convolution:**

  The convolutional operation is performed along three dimensions which means that a kernel of size (*d, h, w*) slides along the temporal dimension *d*, in addition to the spatial dimensions *h* and *w*.

  It captures spatial as well as temporal features.

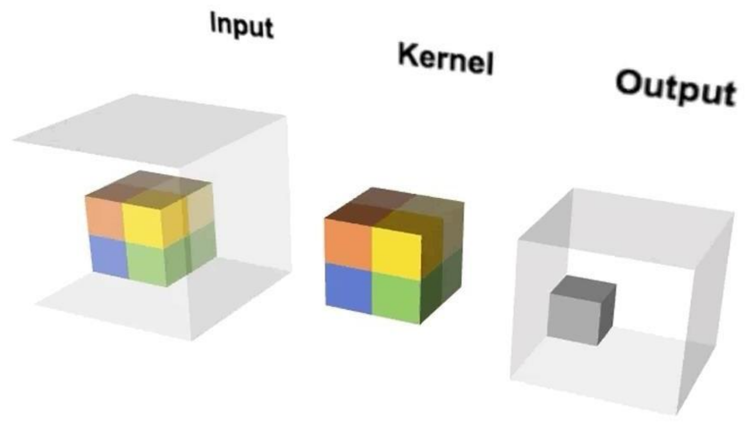
  Each kernel produces a 3D feature map.

- **Limitations:**

  3D CNNs are very computationally expensive due to large number of parameters.

  They are prone to overfitting therefore they require large training set.

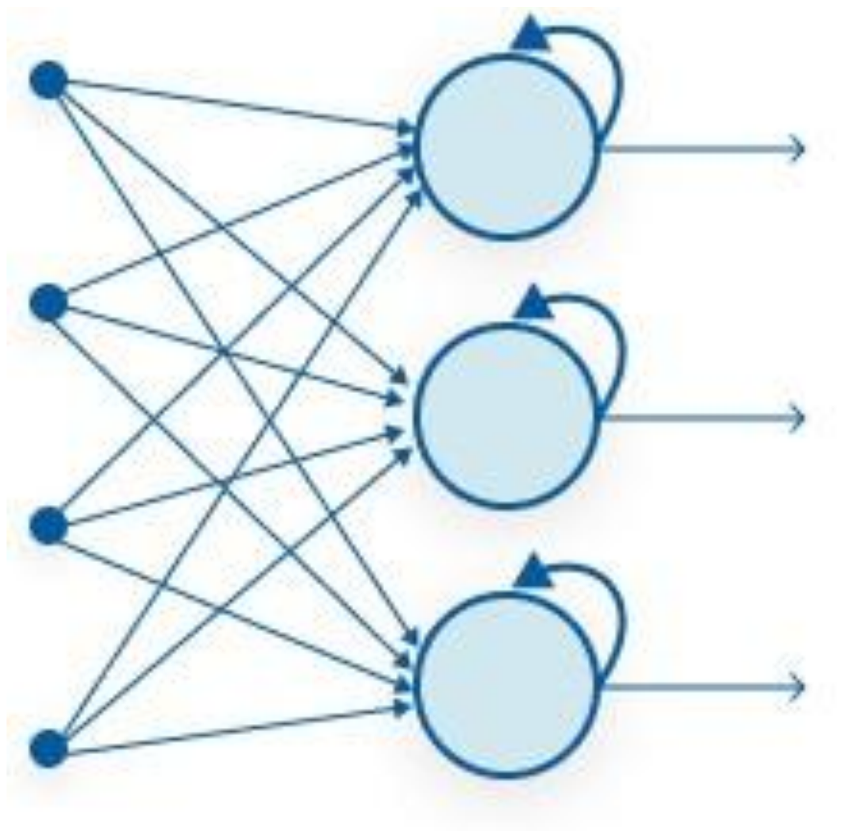
  They cannot capture long term dependencies.



Recurrent Neural Networks

- A class of neural networks that is designed to use with sequential data.
- They can process sequence of arbitrary length, rather than fixed-size inputs.

- This section covers:
  - ✓ Recurrent neurons
  - ✓ Memory cells
  - ✓ Types of RNNs
  - ✓ Training RNNs
  - ✓ Problems with RNNs



Recurrent Neural Networks

- **Recurrent Neurons:**

  It is just like a simple neuron except it has connection pointing backwards.

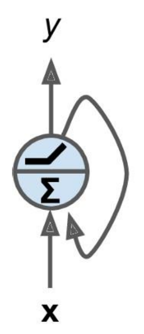
  At each time step *t*, the neuron receives input **x**, and the output **y**, from the previous time step *(t-1)*.

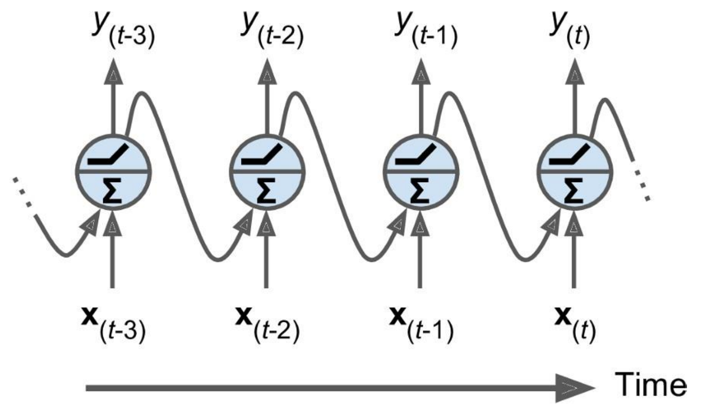
  By placing a bunch of these neurons together we can create a recurrent layer.

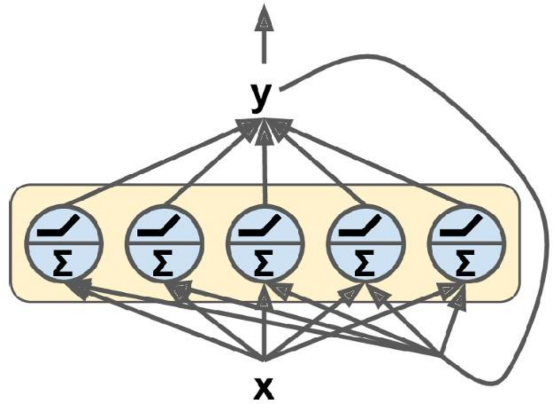
  Each recurrent neuron has two sets of weight **Wx** and **Wy** for the current input **x** and previo output **y**, respectively.

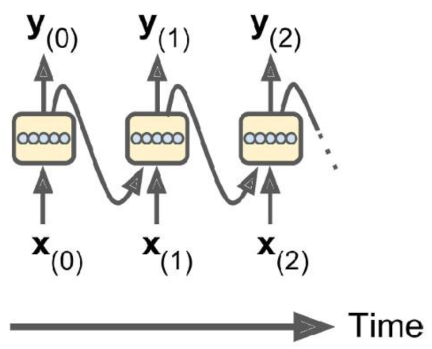
  Output for a recurrent layer is calculated as

$$\mathbf{y}_{(t)} = \phi\left(\mathbf{W}_x{}^\top \mathbf{x}_{(t)} + \mathbf{W}_y{}^\top \mathbf{y}_{(t-1)} + \mathbf{b}\right)$$









Recurrent Neural Network

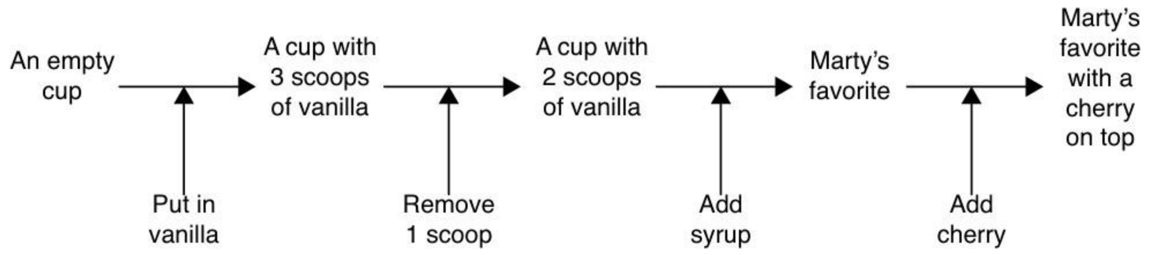
- **Memory Cells or State:**

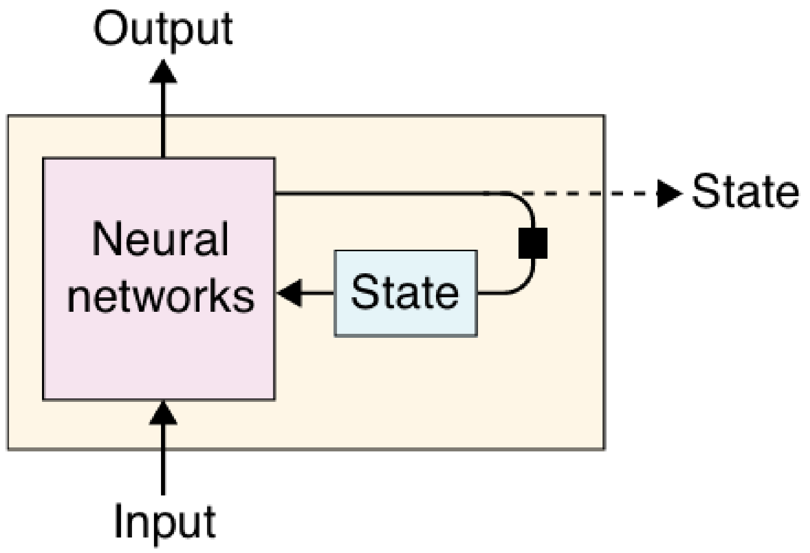
  The output of a recurrent neuron is a function of all inputs from previous time steps. Hence it has a form of memory or state.

  Thus it is capable of learning patterns (short ones)

  It is denoted by **h**, at any time step *t:*

$$\mathbf{h}_{(t)} = f(\mathbf{h}_{(t-1)}, \mathbf{x}_{(t)})$$


An empty cup
Put in vanilla
A cup with 3 scoops of vanilla
Remove 1 scoop
A cup with 2 scoops of vanilla
Add syrup
Marty's favorite
Add cherry
Marty's favorite with a cherry on top



Recurrent Neural Networks

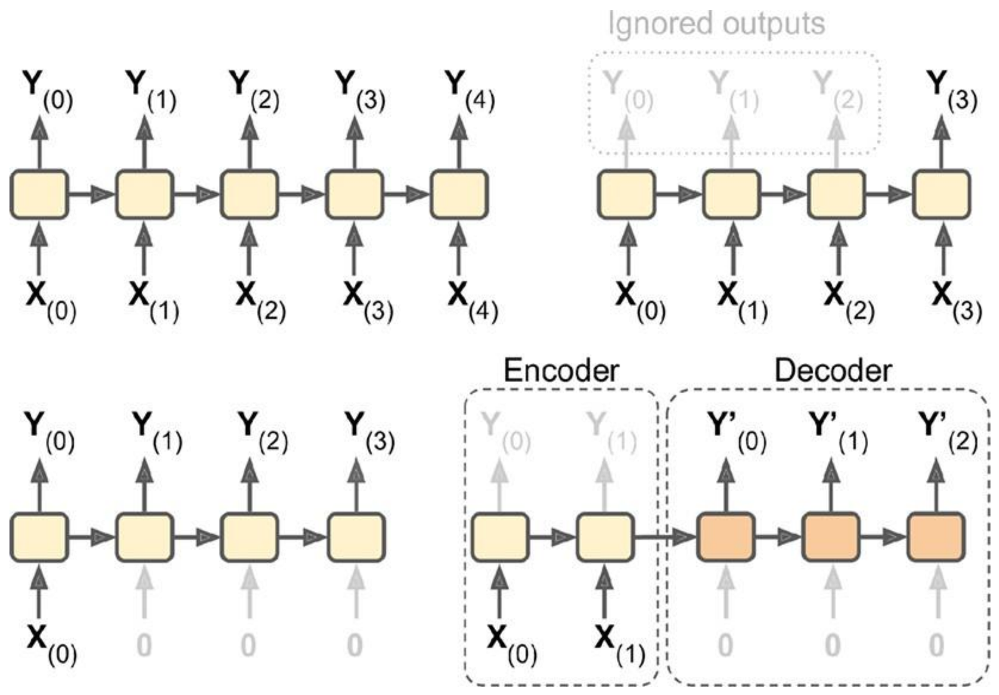
- **Types of RNNs:**

  An RNN that takes a sequence of inputs and provides a sequence of outputs is called ***sequence-to-sequence*** network.

  An RNN that takes a sequence of inputs and produce a single output is called a ***sequence-to-vector*** network.

  Conversely, we also have a ***vector-to-sequence*** network.

  We can have a ***sequence-to-vector*** network followed by a ***vector-to-sequence*** network, this architecture is called ***encoder-decoder***.



Ignored outputs
$Y_{(0)}$ $Y_{(1)}$ $Y_{(2)}$ $Y_{(3)}$ $Y_{(4)}$
$X_{(0)}$ $X_{(1)}$ $X_{(2)}$ $X_{(3)}$ $X_{(4)}$
$Y_{(0)}$ $Y_{(1)}$ $Y_{(2)}$ $Y_{(3)}$
$X_{(0)}$ $X_{(1)}$ $X_{(2)}$ $X_{(3)}$
$Y_{(0)}$ $Y_{(1)}$ $Y_{(2)}$ $Y_{(3)}$
$X_{(0)}$ 0 0 0
Encoder
Decoder
$Y_{(0)}$ $Y_{(1)}$ $Y'_{(0)}$ $Y'_{(1)}$ $Y'_{(2)}$
$X_{(0)}$ $X_{(1)}$ 0 0 0

Recurrent Neural Networks

- **Backpropagation through time:**

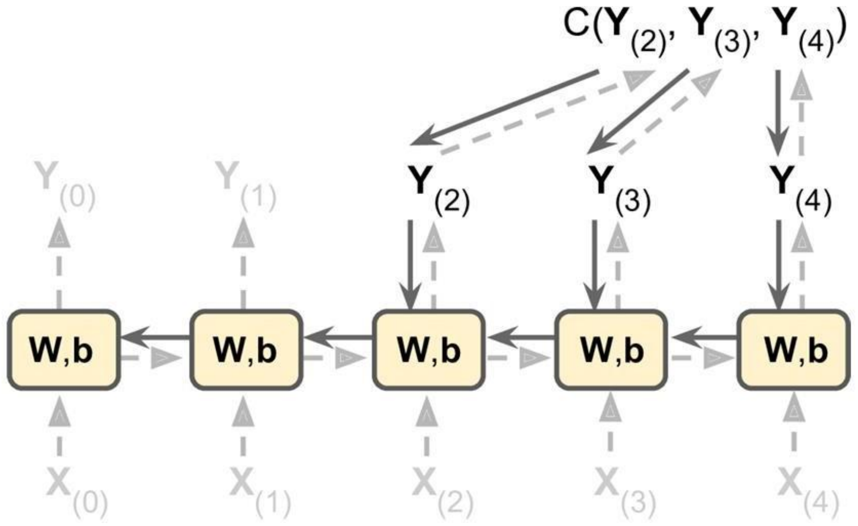
  First we unroll the network through time.

  The loss for the output at each time step (or only at final time step) is calculated.

  The loss is propagated backward through each time step.

  The gradient of the loss is calculated by summing the gradient at each time step.

  Finally, the weights and biases are updated based on this gradient



Recurrent Neural Networks

- **Unstable gradients:**

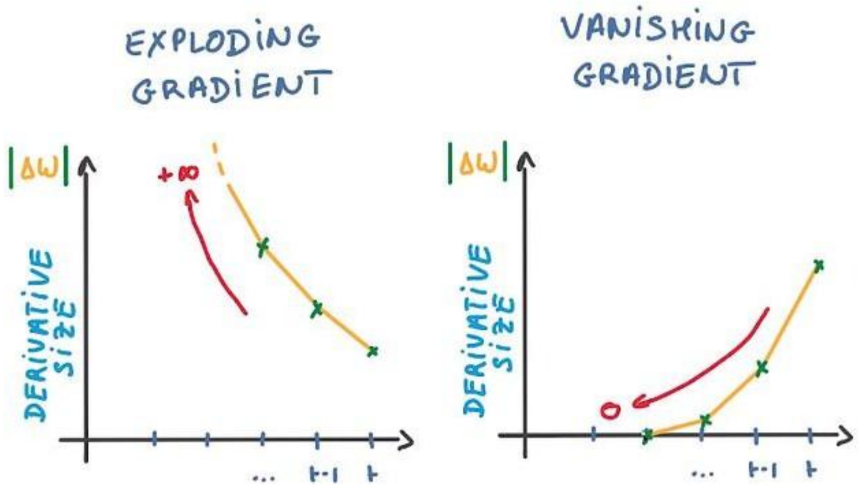
  RNNs suffer from unstable gradients due to the nature of backprop through time.

  Vanishing gradient occurs when the gradient gets smaller and smaller with each time step and exploding gradient occurs when the opposite happens

- **Short-term memory:**

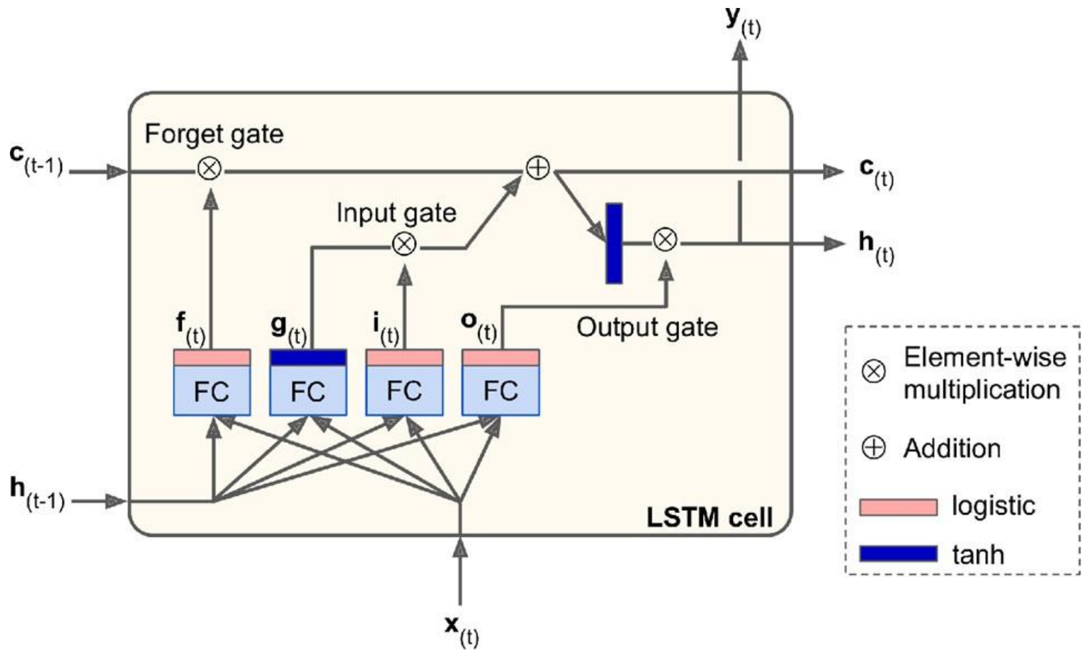
  As RNN processes a long sequence, it will gradually forget the first inputs, hence we say that they have short-term memory.

  As the hidden state saves information from all previous time steps, but due to the fix size of the hidden state there is a limit to how much information can be retained.



Long Short-term Memory

- **LSTM cells:**
  - It is designed to capture long term dependencies.
  - The state is split into two states, short-term state $\mathbf{h}_{(t)}$ and long-term state $\mathbf{c}_{(t)}$.
  - There are four fully connected layers which serve different purposes:
    - The main layers is the one with the **tanh** activation. It analyzes the current $\mathbf{x}_{(t)}$ and previous short-term state $\mathbf{h}_{(t-1)}$ and outputs the most important parts.
    - The other three layers are gate controllers.



Long Short-term Memory

- **Forget gate:**

  The forget gate controls which part of long-term state should be erased

  $\boldsymbol{f}_t = \sigma(W_f \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f)$

- **Input gate:**

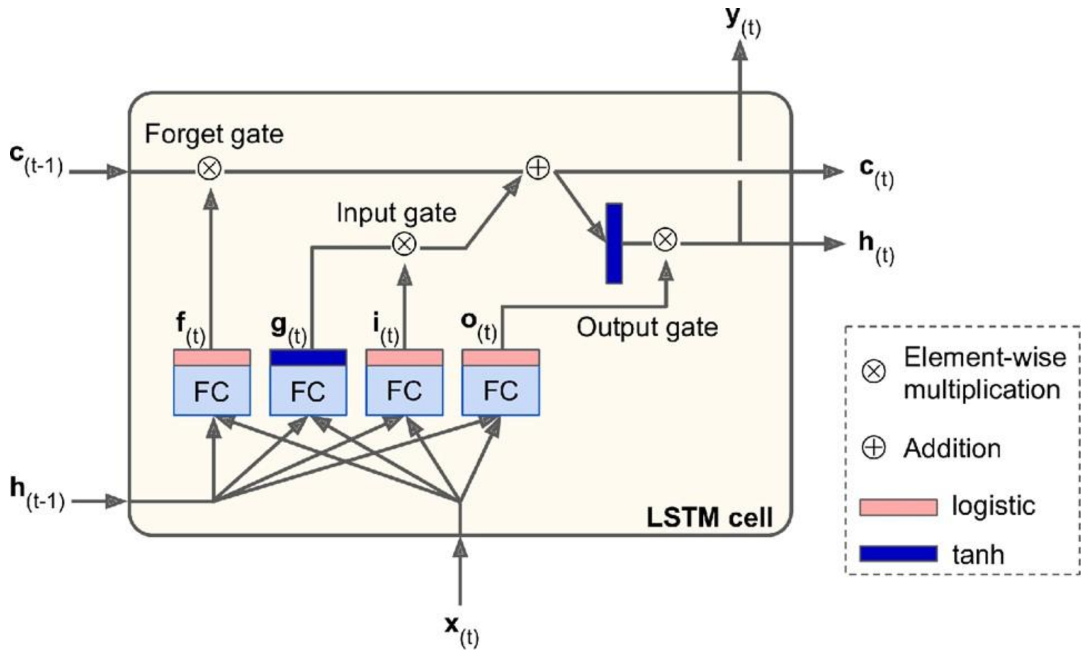
  The input gate controls which part of the input from the current time step should be added to the long-term state.

  $\boldsymbol{i}_t = \sigma(W_i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i)$

- **Output gate:**

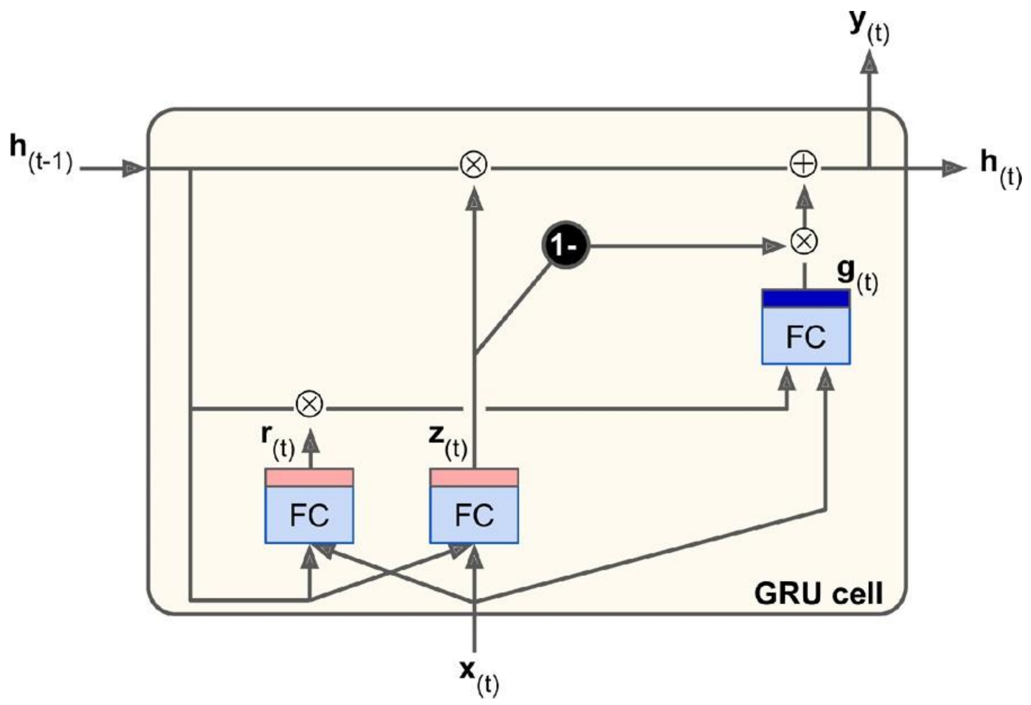
  The output gate controls which part of the long-term state should be read and output at this time step.

  $\boldsymbol{o}_t = \sigma(W_o \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o)$



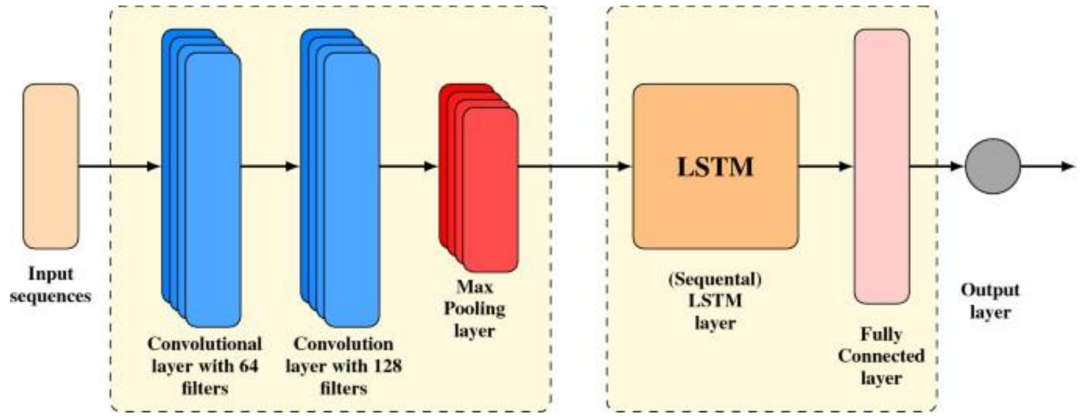
Gated Recurrent Unit

- The GRU cell is the simplified version of LSTM.
- Both state vectors are merged into a single vector.
- We have a main layer, and two gate controllers, update gate and reset gate.
- The update gate does the work of a forget gate and the input gate.
- The other gate controller decides which part of the state will be shown to the main layer.



Combining CNN and RNN

- We can leverage the strengths of both architectures by combining them.
- CNN will extract spatial features from the data, which are fed to RNN to capture temporal dependencies.
- Enable the model to learn effective representation of the input data
- **Applications:**
  - Image captioning
  - Video analysis
  - etc



Do you have any Questions?