Natural Language Processing (NLP)

RNN Variants

Equipping You with Research Depth and Industry Skills

By:

Dr. Zohair Ahmed

www.youtube.com/@ZohairAI Subscribe

www.begindiscovery.com

NN uses only linear activations

- What happens if NN uses only linear activations
- Each layer looks like: $\mathrm{h} = \mathrm{W}\,\mathrm{x} + \mathrm{b}$
- Stack 3 layers: $h_3 = W_3(W_2(W_1 x + b_1) + b_2) + b_3$
- This just collapses into one big linear transformation: $h_3 = W'x + b'$
- So, a multi-layer “linear” network is no more powerful than a single linear layer.
- It can only model straight lines (or planes/hyperplanes).

Why that’s a limitation

- Real-world problems (images, text, speech, prices) are rarely linear.
- For example: a circle-shaped decision boundary can’t be drawn with a straight line.
- So, a purely linear NN couldn’t separate such classes.

Why that’s a limitation

- Real-world problems (images, text, speech, prices) are rarely linear.
- For example: a circle-shaped decision boundary can’t be drawn with a straight line.
- So, a purely linear NN couldn’t separate such classes.
- **Where linear is still used**
- **Last layer** of a NN is often linear, *before* applying something else:
  - Classification: linear → softmax.
  - Regression: linear output (predicting a real number).
- Inside the network, though, we almost always add **nonlinearities** (tanh, ReLU, etc.) to give it real power.

Why that's a limitation

- **Linear functions (straight-line type)**
- A function is linear if output is directly proportional to input (no curves, no bends). Examples:
- $f(x) = w \cdot x + b$ (basic linear regression)
- In neural nets, a linear layer is just a matrix multiply + bias.
- **Nonlinear functions (curved, bending)**
- These bend the input → allow neural nets to learn complex patterns. Common activation functions:

- **Sigmoid**:$\sigma(x) = \frac{1}{1+e^{-x}}$
- **tanh**: squashes to $(-1,1)$
- **ReLU**: $f(x) = \max(0, x)$
- **Leaky ReLU**: lets small negative slope through
- **Softmax**: turns logits into probabilities
- **ELU, GELU, Swish** (newer smooth activations)
- **Rule of thumb:**
- **Linear** = straight lines, planes, or flat scaling.
- **Nonlinear** = curves, bends, squashes, or anything you can't reduce to a straight line.

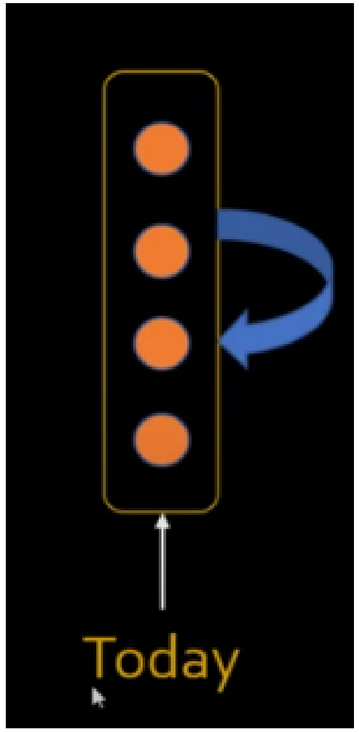
Why do we need nonlinearity at all?

- If we didn’t have it:
- Multiple layers of linear functions would collapse into just **one linear function**.
- That means the network could only learn straight lines/planes, too limited for real-world data.
- **With nonlinearity:**
- The model can “curve” and combine patterns.
- For example, in housing prices:
  - Up to a certain area, price increases slowly.
  - After 2000 sq. ft., price jumps much faster.

    That’s not a line: that’s a curve.

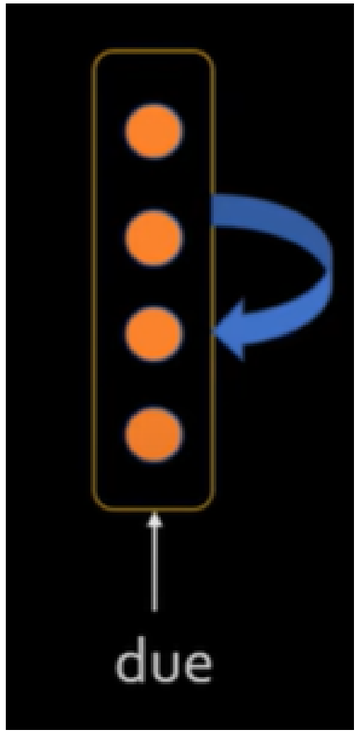
Traditional RNN (Vanilla)

Today, due to my current job situation and family conditions, I need to take a loan.



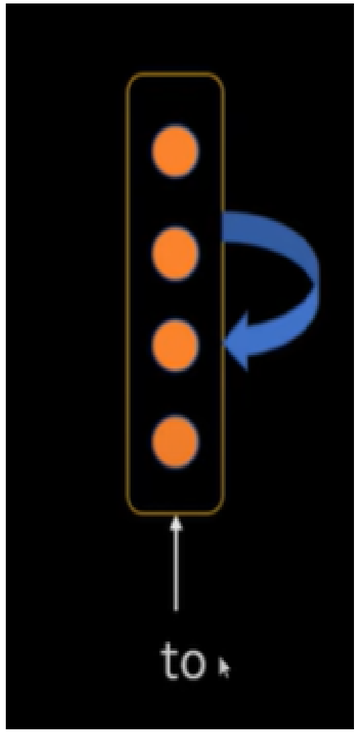
Traditional RNN (Vanilla)

Today, due to my current job situation and family conditions, I need to take a loan.



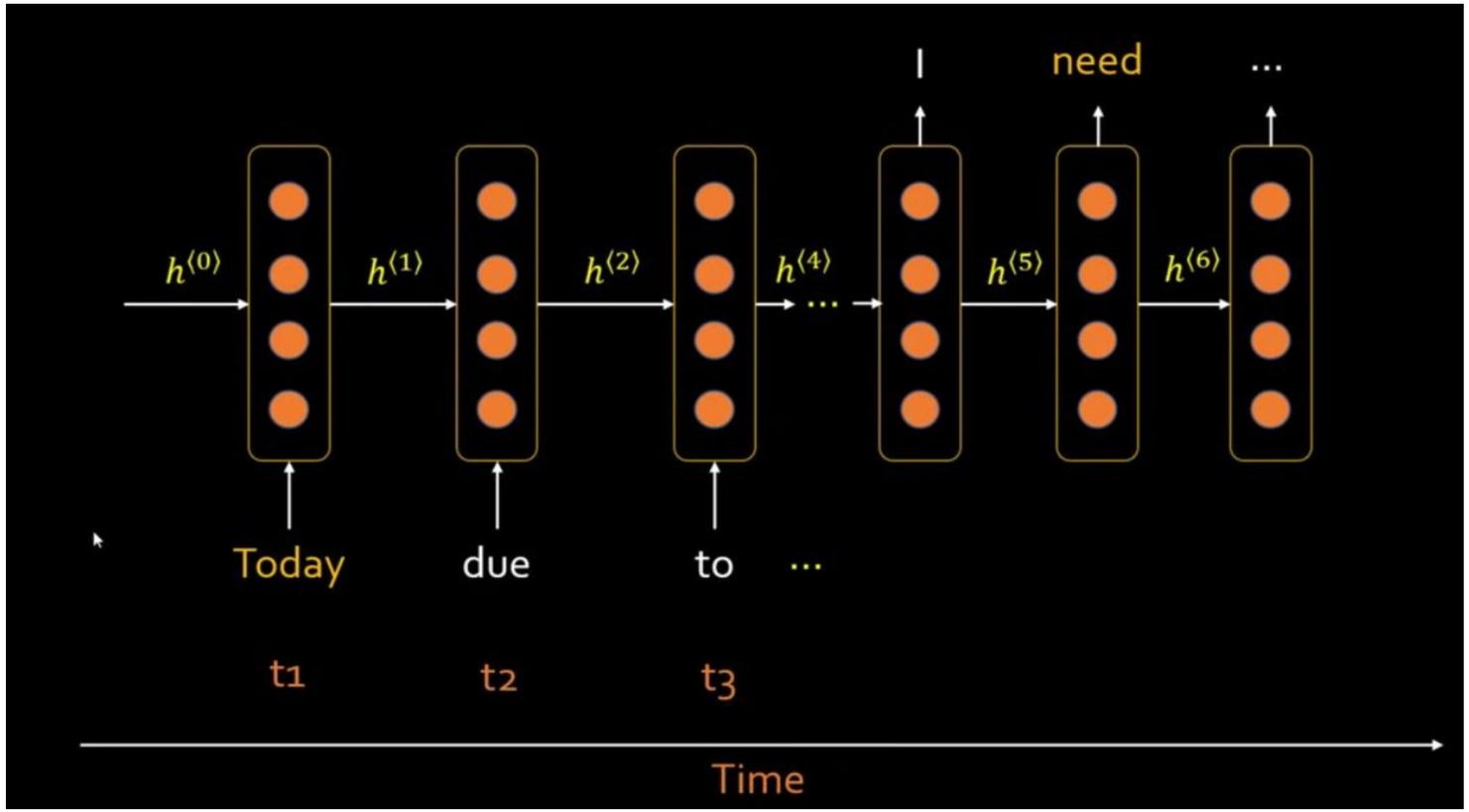
Traditional RNN (Vanilla)

Today, due to my current job situation and family conditions, I need to take a loan.

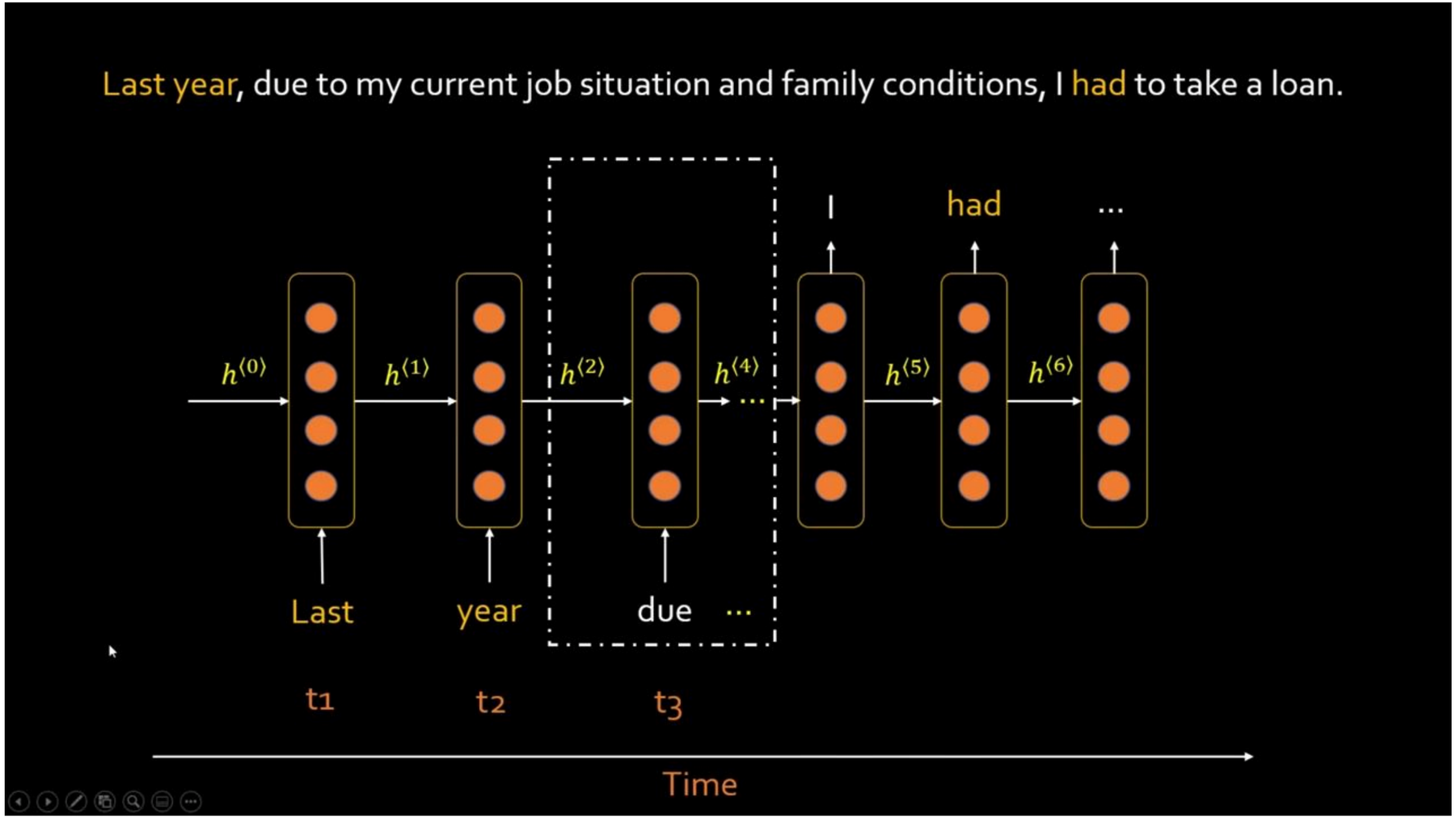


Traditional RNN (Vanilla)

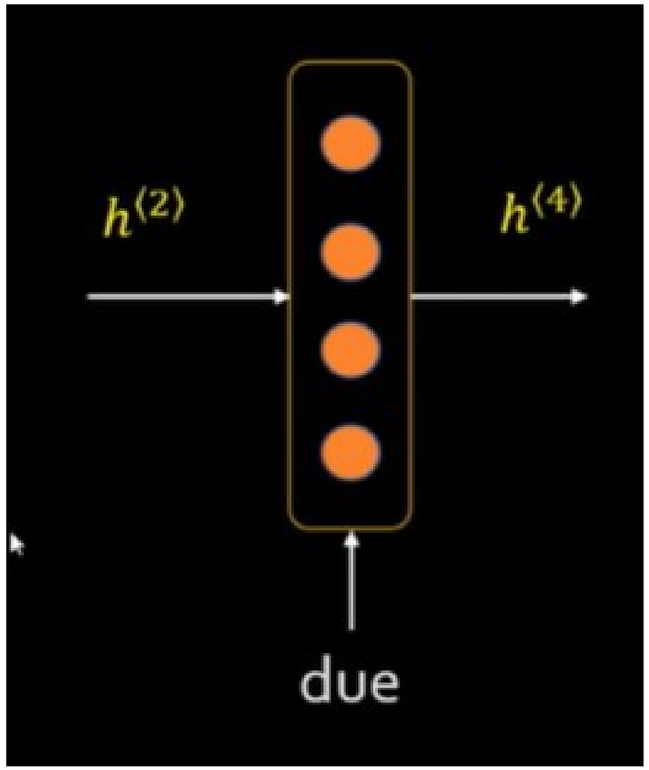
Today, due to my current job situation and family conditions, I need to take a loan.

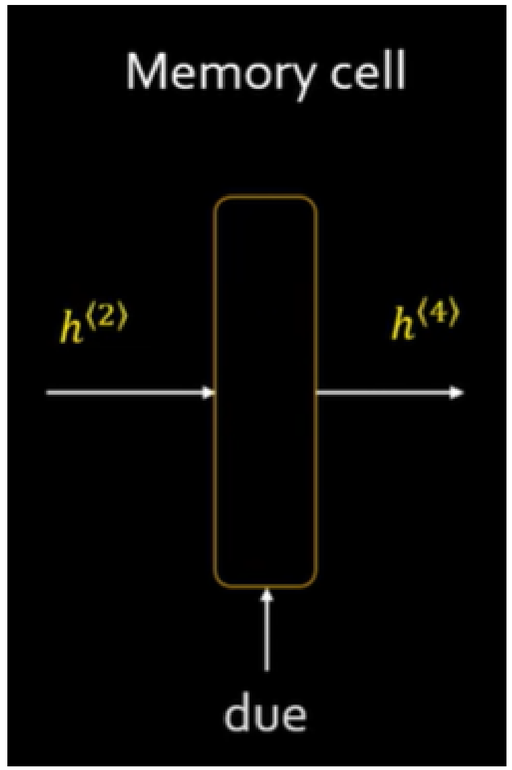


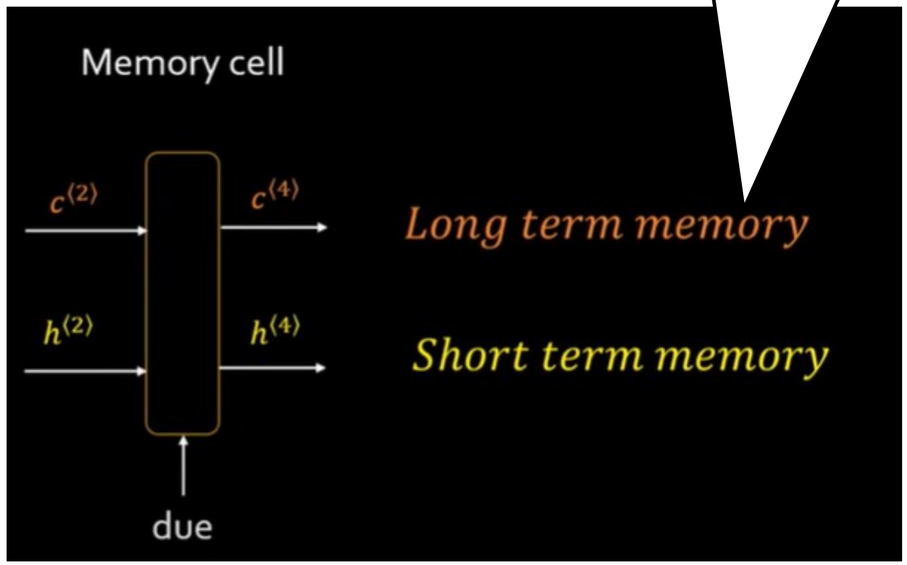
Traditional RNN (Vanilla)



Traditional RNN (Vanilla)

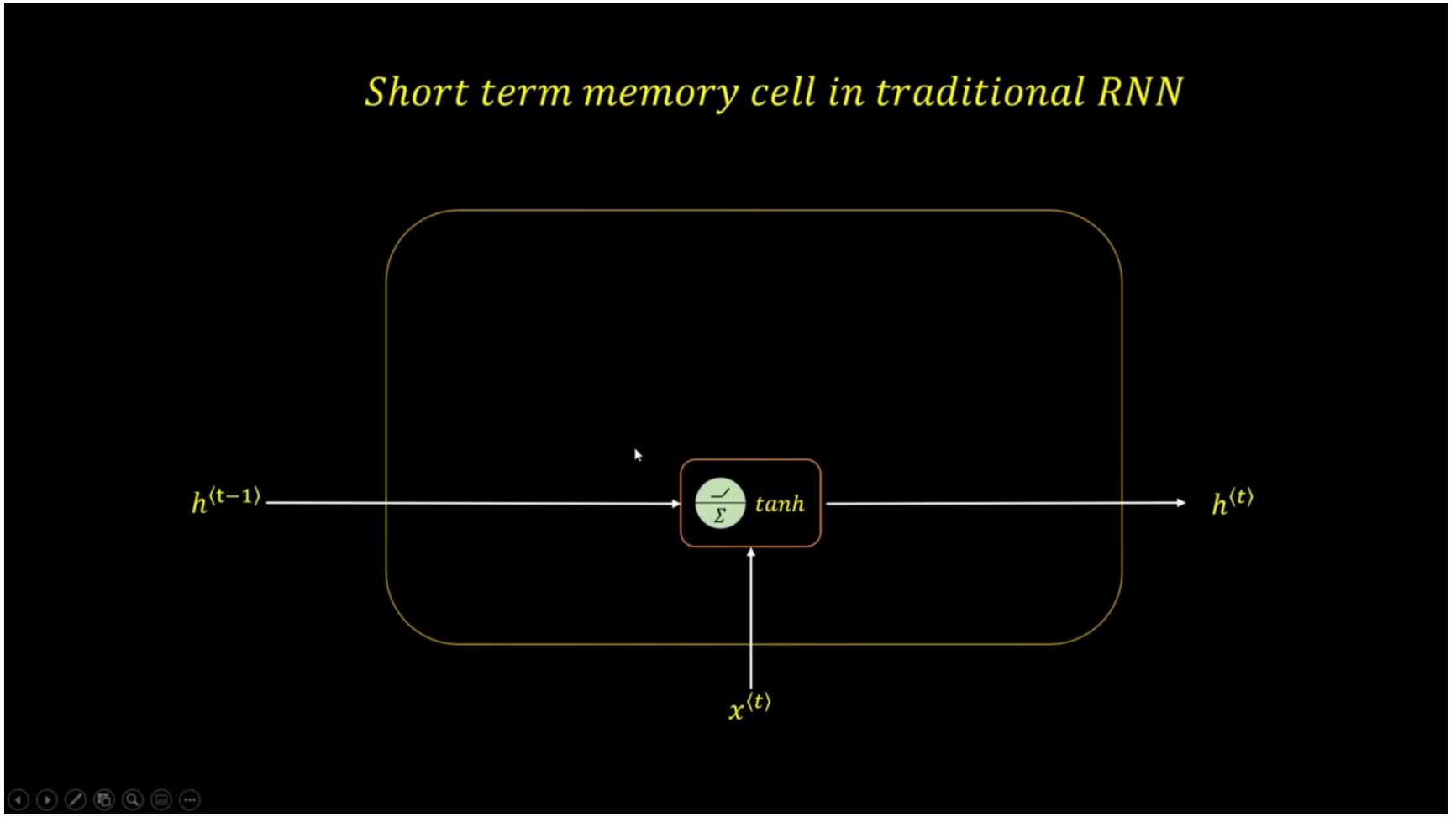




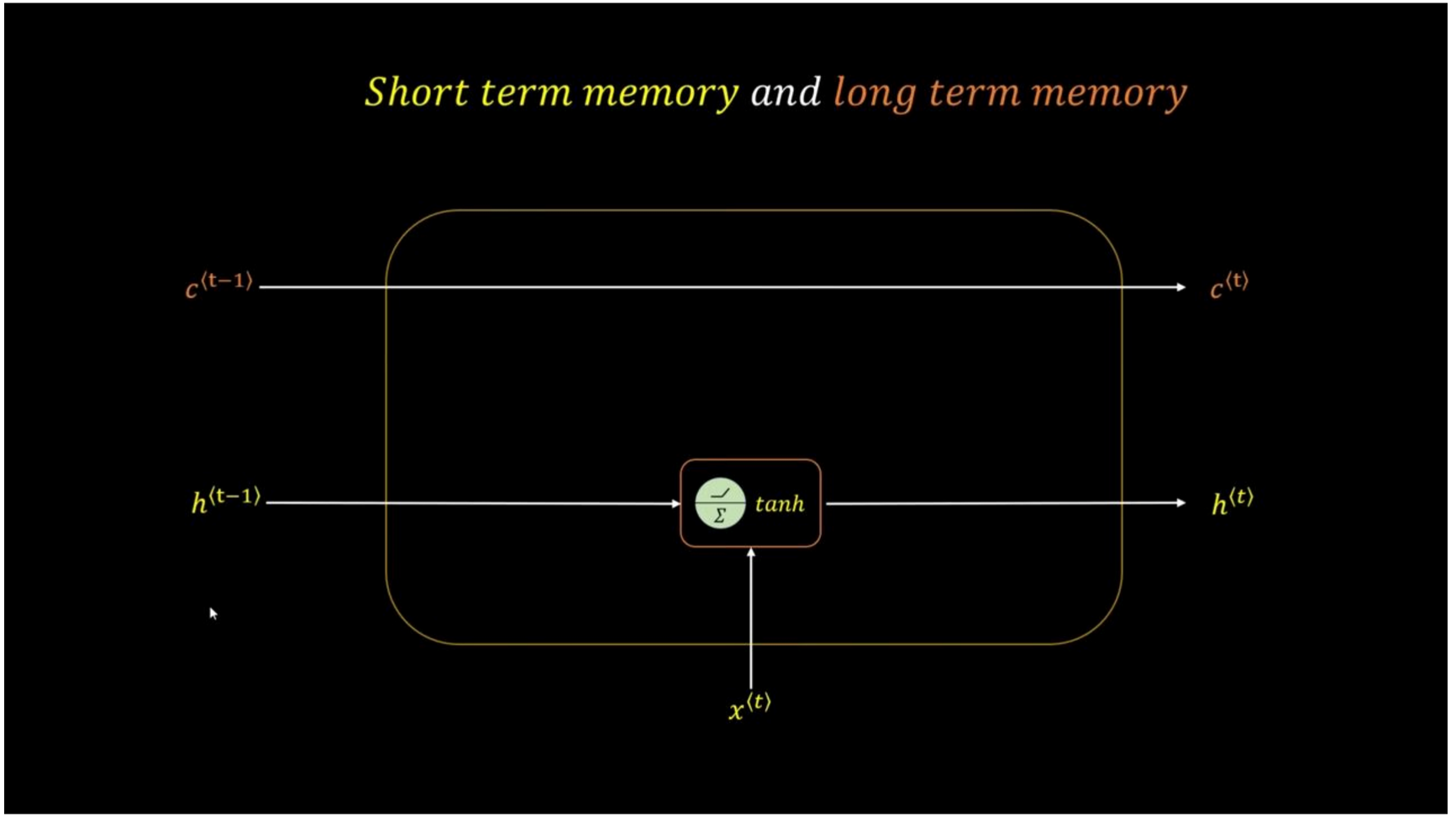


Introduce New State

Traditional RNN



Traditional RNN → LSTM



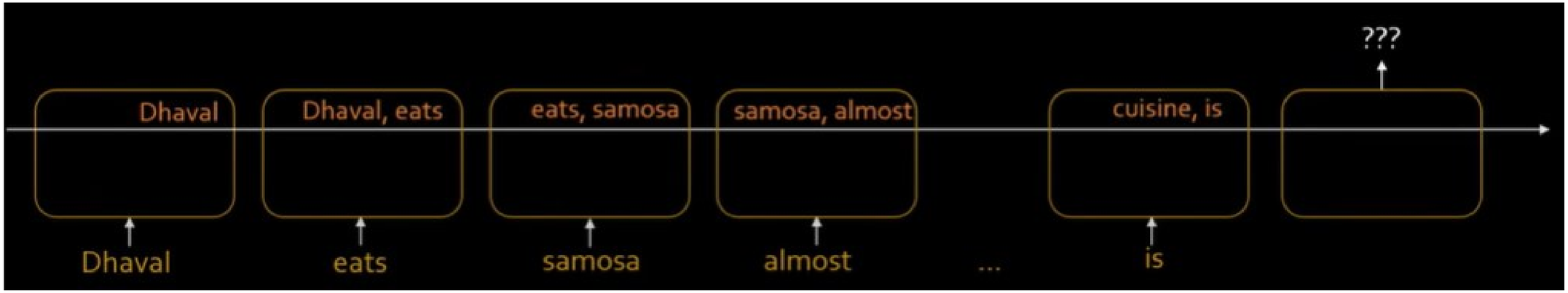
Traditional RNN

Dhaval eats samosa almost everyday, it shouldn't be hard to guess that his favorite cuisine is ???

Dhaval eats samosa almost everyday, it shouldn't be hard to guess that his favorite cuisine is Indian

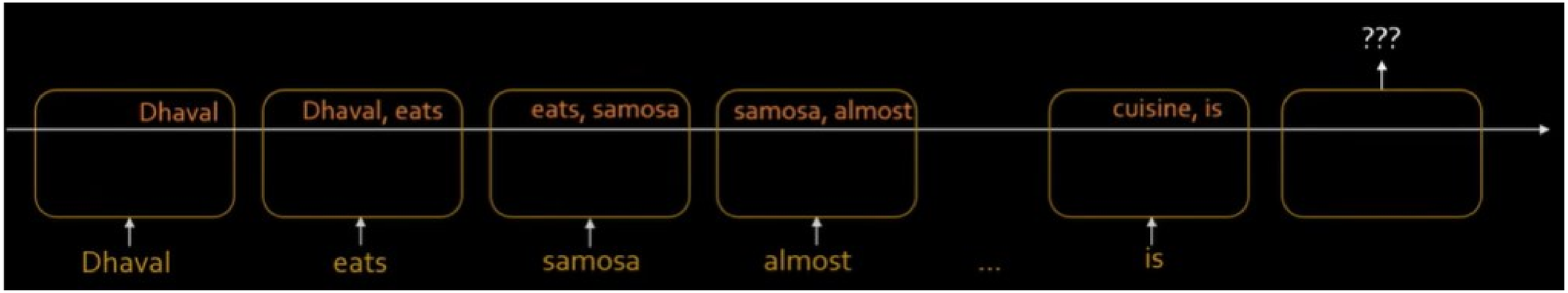
Traditional RNN

Dhaval eats samosa almost everyday, it shouldn't be hard to guess that his favorite cuisine is Indian



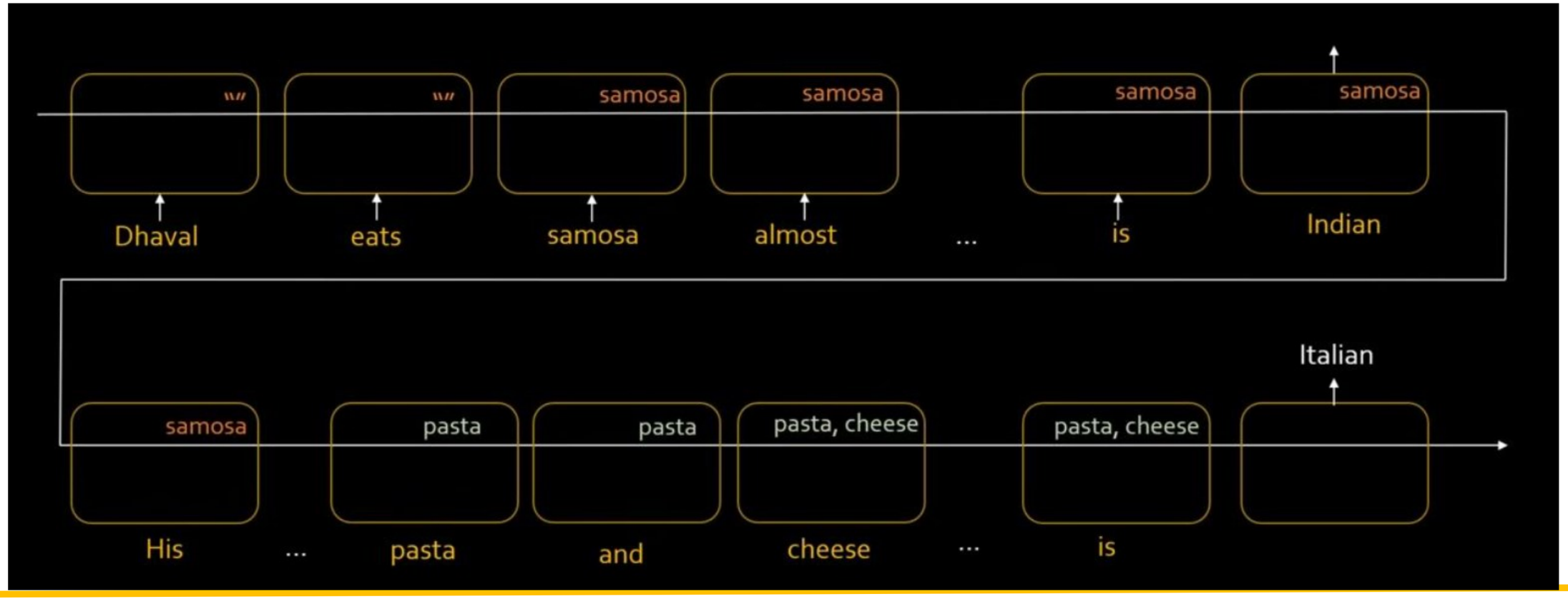
Traditional RNN

Dhaval eats samosa almost everyday, it shouldn't be hard to guess that his favorite cuisine is Indian



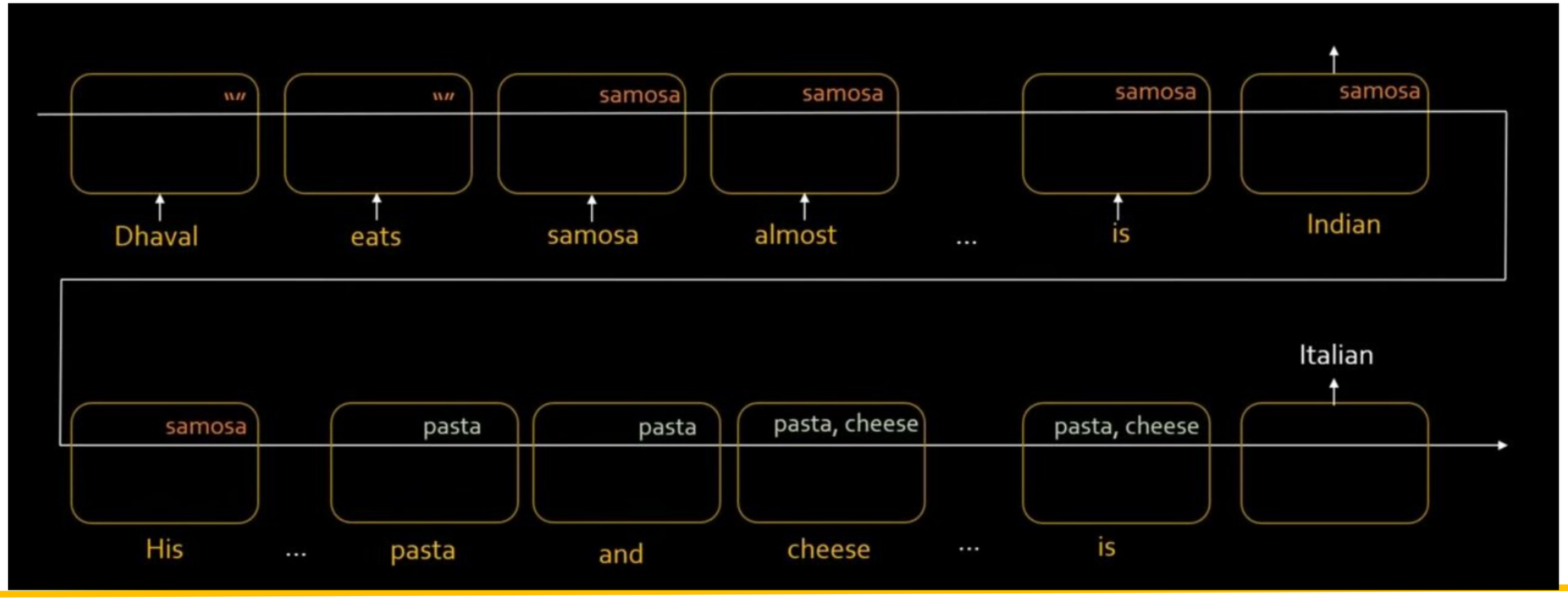
LSTM

Dhaval eats samosa almost everyday, it shouldn't be hard to guess that his favorite cuisine is Indian. His brother Bhavin however is a lover of pastas and cheese that means Bhavin's favorite cuisine is Italian

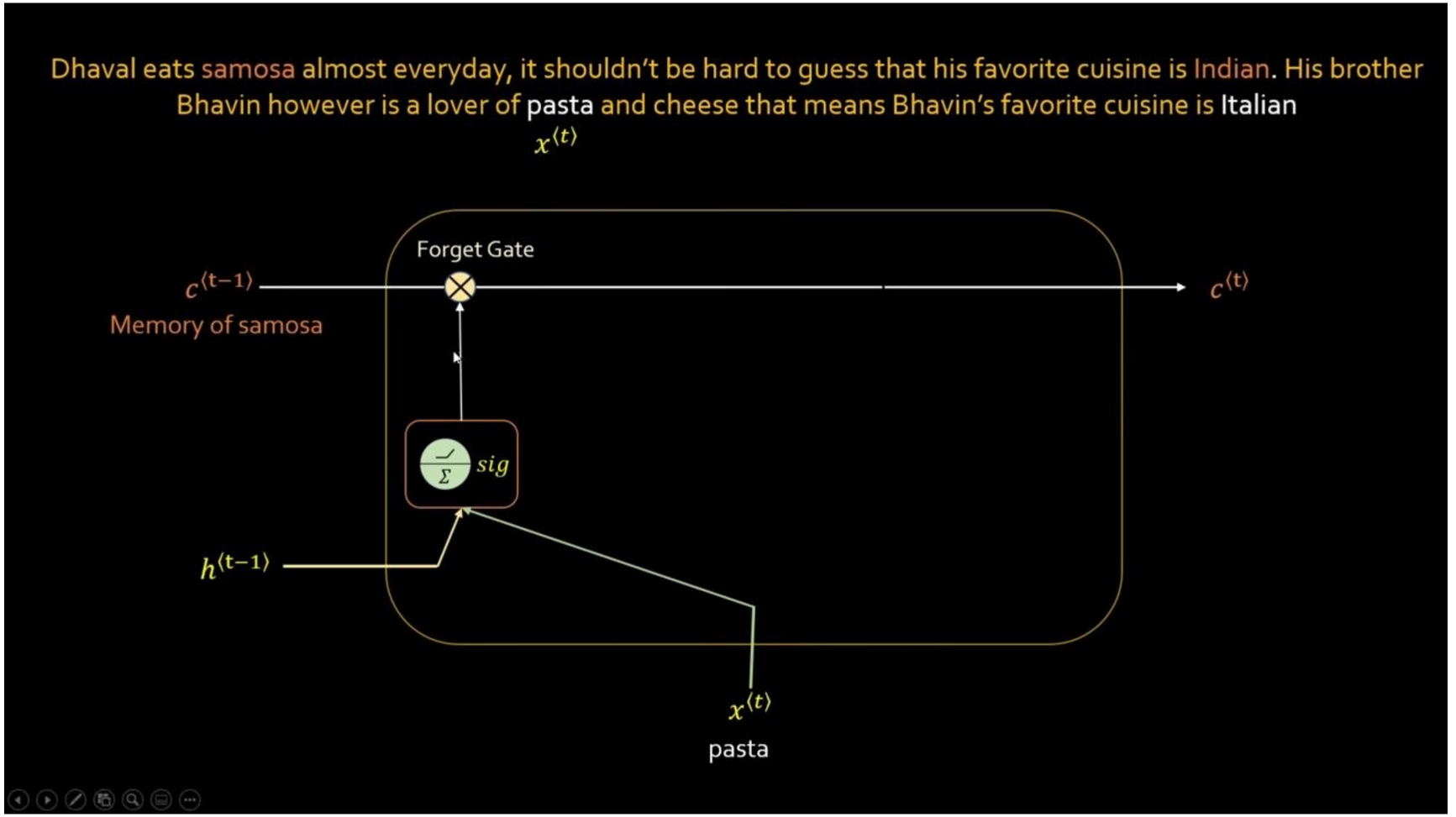


LSTM Long Term Memory

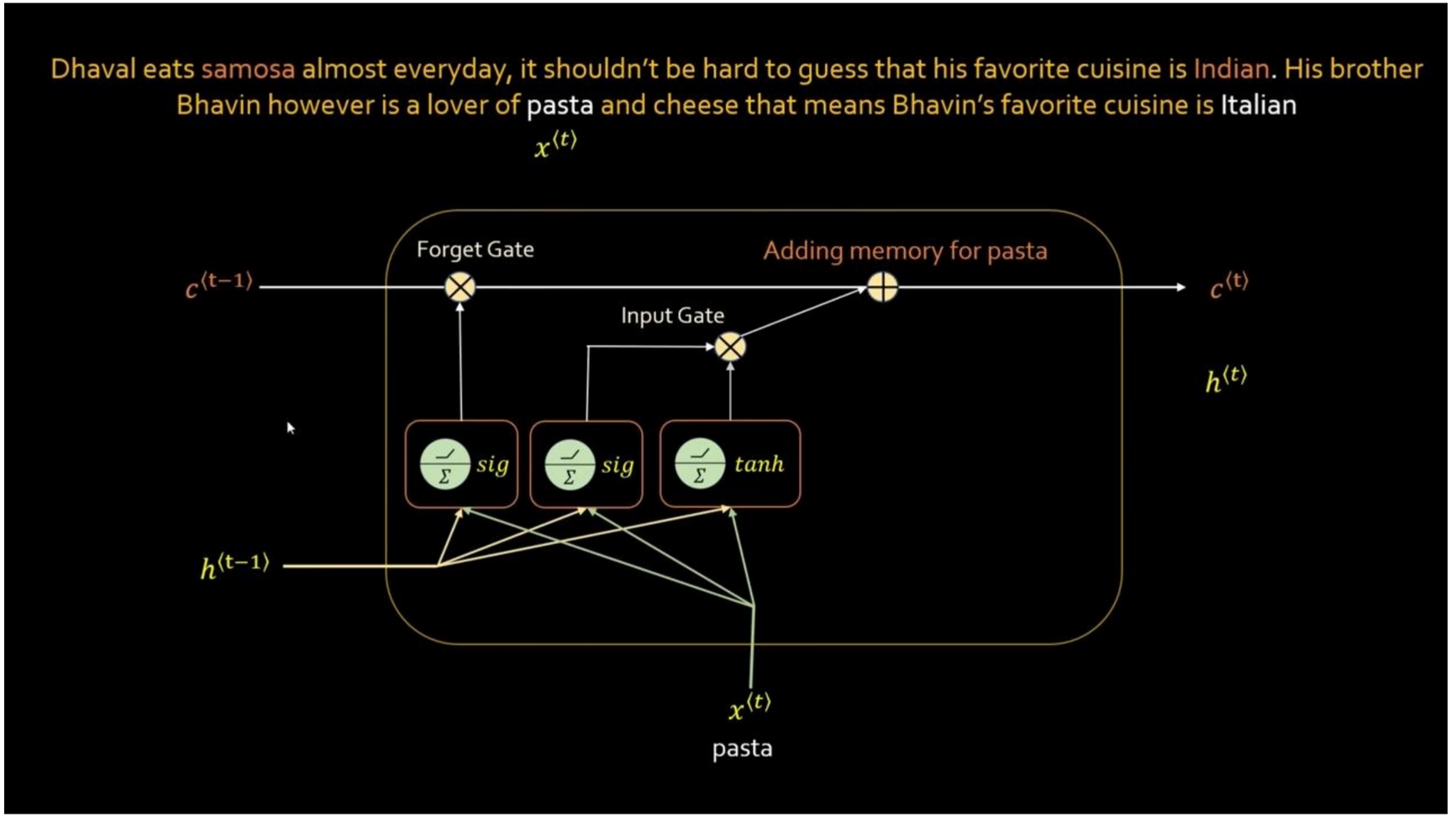
Dhaval eats samosa almost everyday, it shouldn't be hard to guess that his favorite cuisine is Indian. His brother Bhavin however is a lover of pastas and cheese that means Bhavin's favorite cuisine is Italian



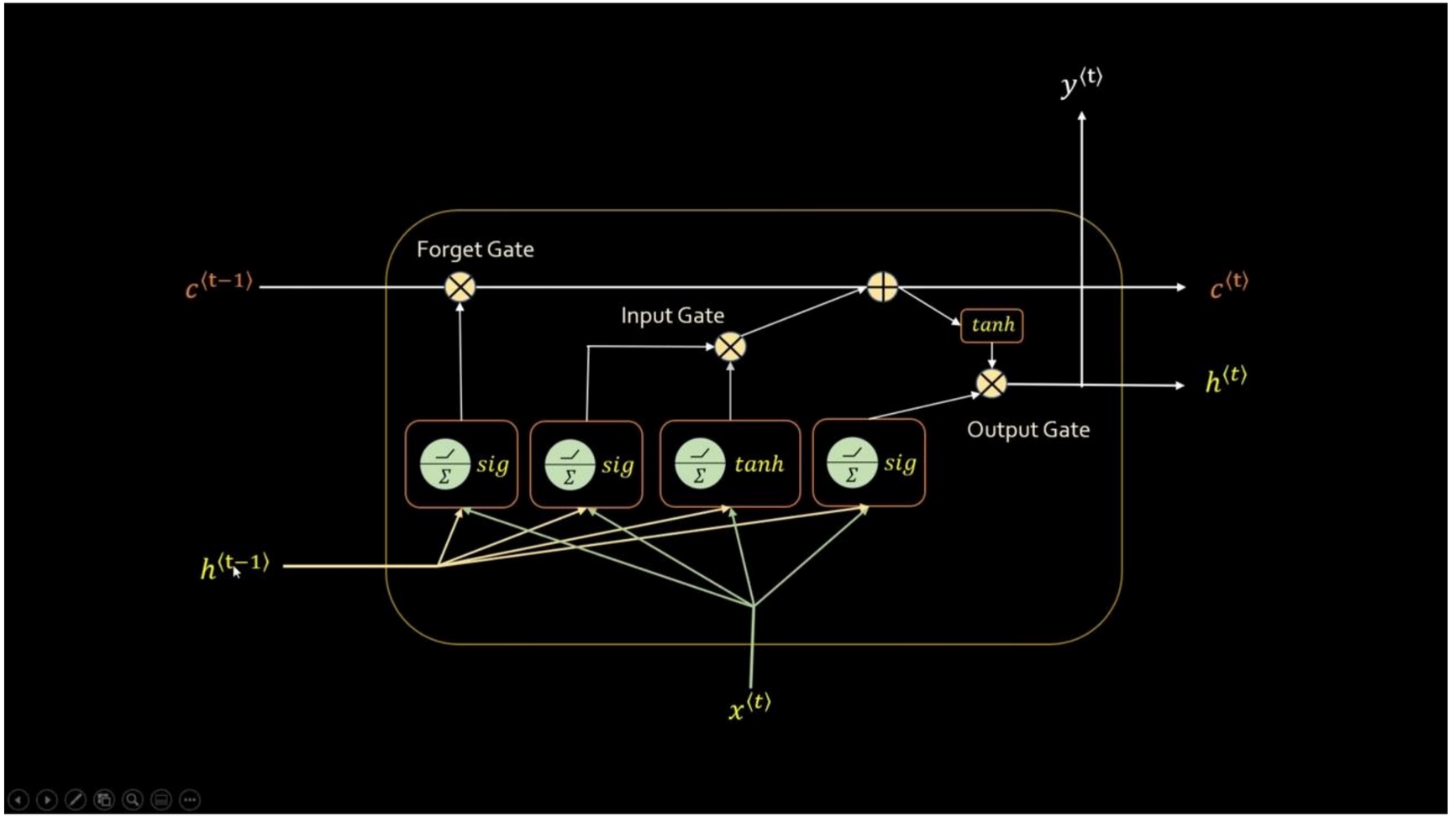
LSTM Long Term Memory



LSTM Long Term Memory



LSTM Long Term Memory



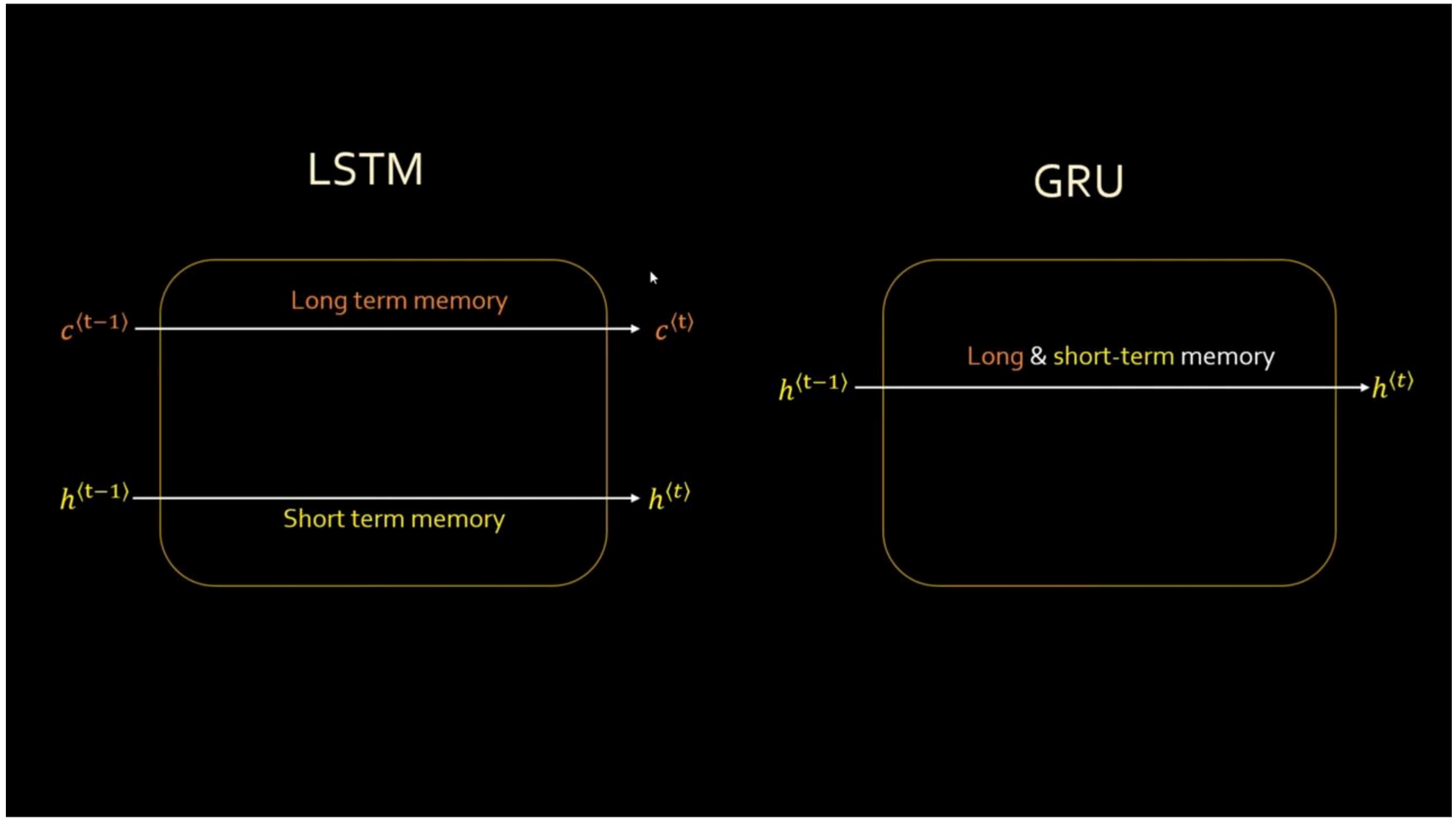
LSTM Long Term Memory

- https://colah.github.io/posts/2015-08-Understanding-LSTMs/

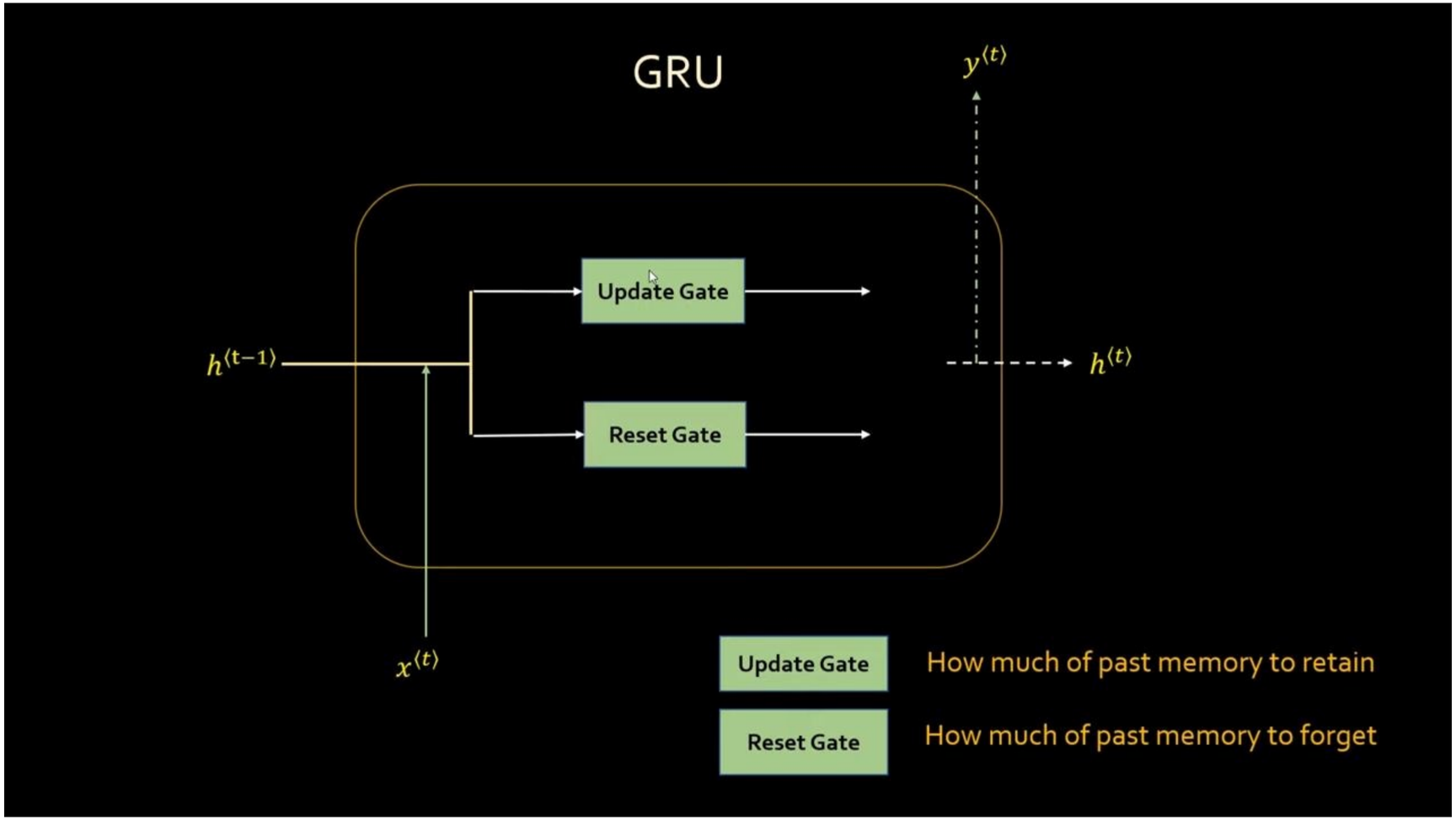
GRU

- Gated Recurrent Units (GRUs) are a type of RNN introduced by Cho et al. in 2014.
- The core idea behind GRUs is to use gating mechanisms to selectively update the hidden state at each time step
- It allowing them to remember important information while discarding irrelevant details.

GRU

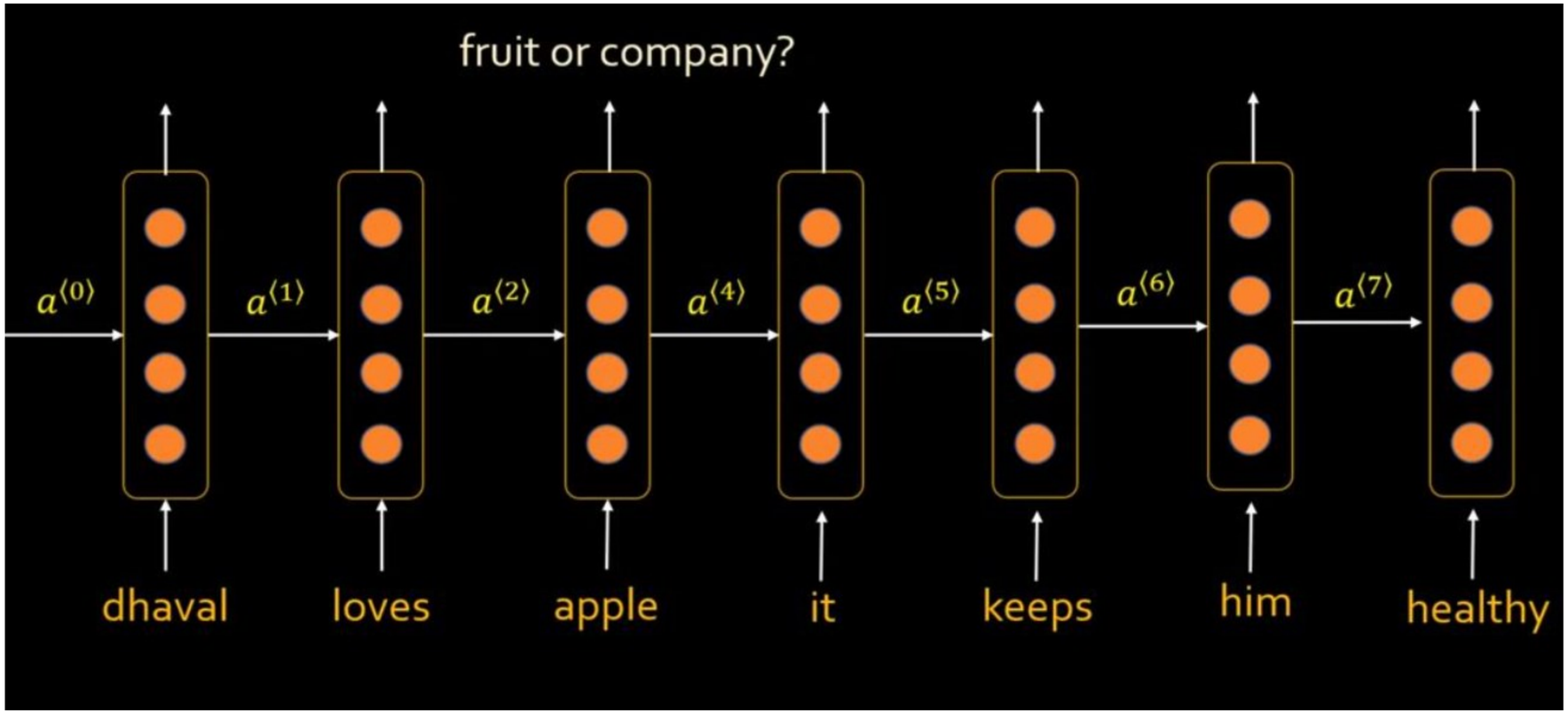

LSTM
Long term memory
$c^{\langle t-1 \rangle}$
$c^{\langle t \rangle}$
$h^{\langle t-1 \rangle}$
$h^{\langle t \rangle}$
Short term memory
GRU
Long & short-term memory
$h^{\langle t-1 \rangle}$
$h^{\langle t \rangle}$

GRU


GRU
$y^{\langle t \rangle}$
Update Gate
$h^{\langle t-1 \rangle}$
$h^{\langle t \rangle}$
Reset Gate
$x^{\langle t \rangle}$
Update Gate
How much of past memory to retain
Reset Gate
How much of past memory to forget

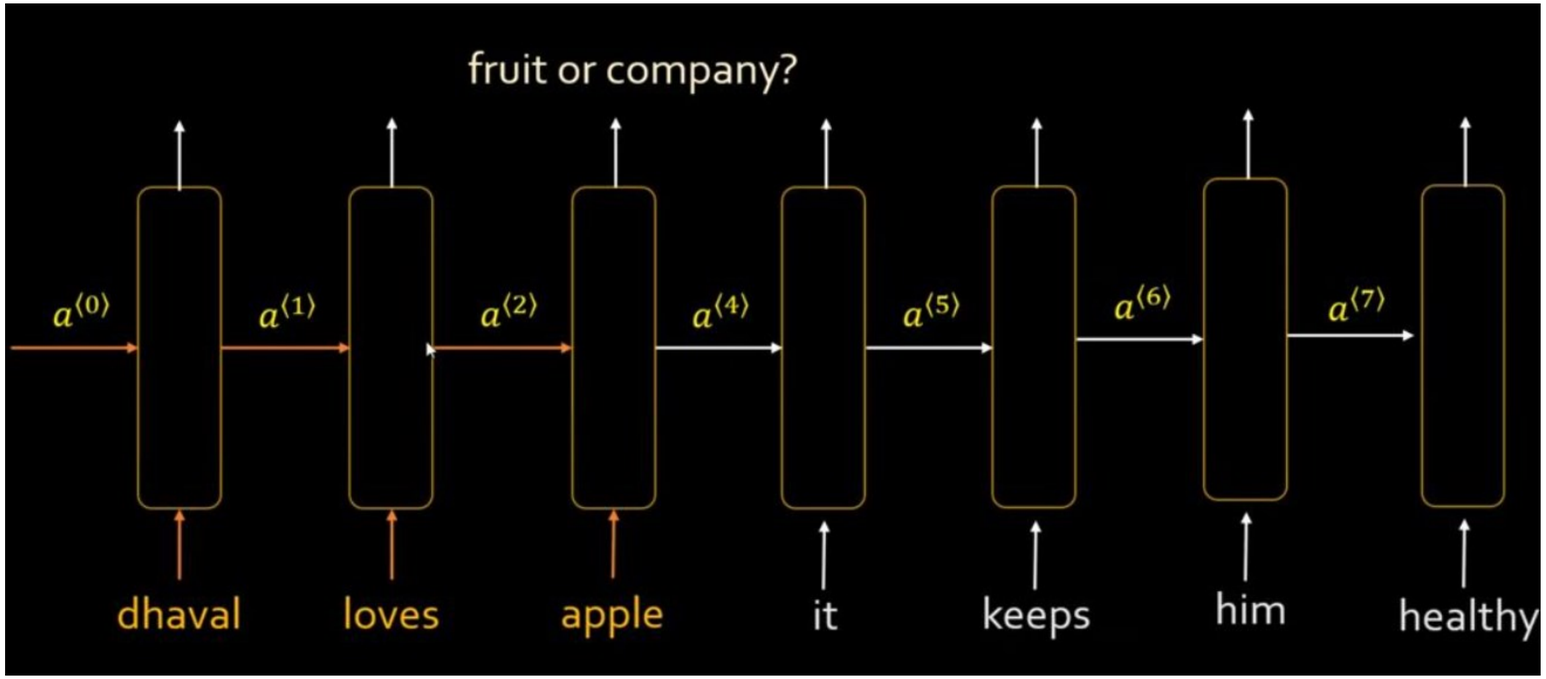
Bidirectional RNN

Bidirectional RNN



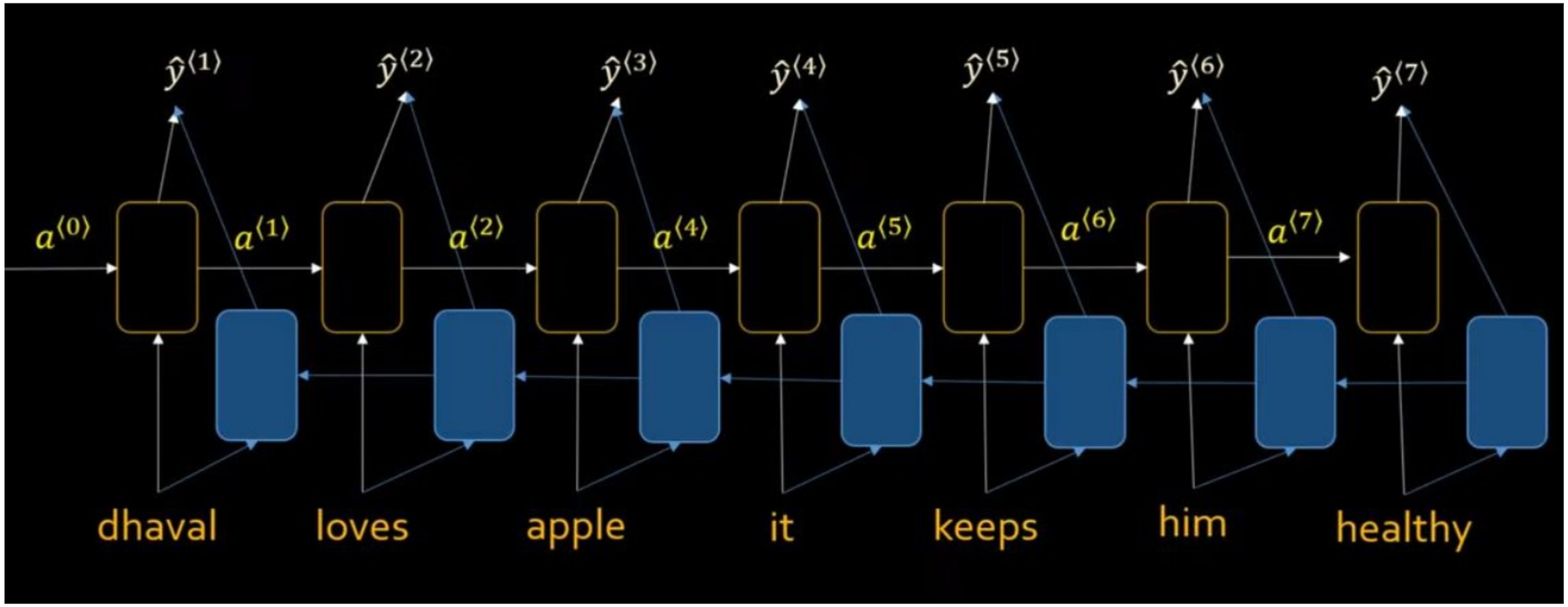
Bidirectional RNN

- The Word apple influence after the words appear.
- So, we Need one more Layer which is Right to Left.

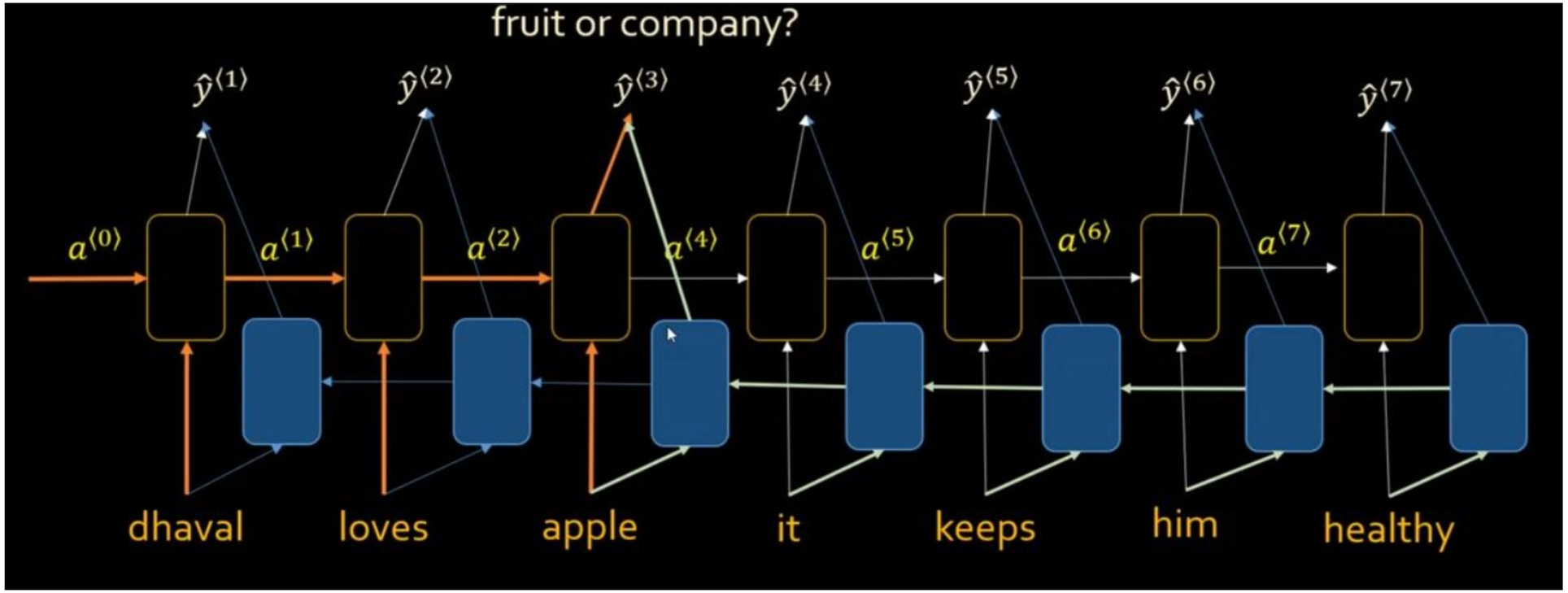


Bidirectional RNN

- We add a new Layer in Blue Color



Bidirectional RNN



TensorFlow Class Reference

- **RNN Class:** In TensorFlow, the tf.keras.layers.SimpleRNN class is used to implement a vanilla RNN. It can be used for tasks such as time-series forecasting or simple language modeling.
  - rnn_layer = tf.keras.layers.SimpleRNN(units=128)
- **LSTM Class:** The tf.keras.layers.LSTM class implements a Long Short-Term Memory network. This is used when you need to handle long-term dependencies in sequential data.
  - lstm_layer = tf.keras.layers.LSTM(units=128)
- **GRU Class:** The tf.keras.layers.GRU class implements a Gated Recurrent Unit (GRU), which is simpler and faster than LSTM but still handles long-term dependencies.
  - gru_layer = tf.keras.layers.GRU(units=128)
- **Bidirectional Layer:** The tf.keras.layers.Bidirectional wrapper can be used with RNN, LSTM, or GRU to process sequences in both forward and backward directions. This is especially useful for tasks where both past and future context are important (e.g., sentiment analysis, machine translation).
  - bi_lstm_layer = tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(units=128))