Learning Deep Learning with PyTorch

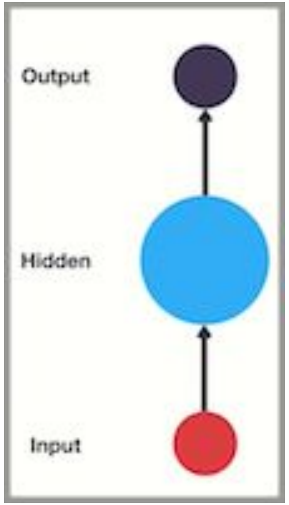
(6) RNNs and LSTM

Qiyang Hu
IDRE

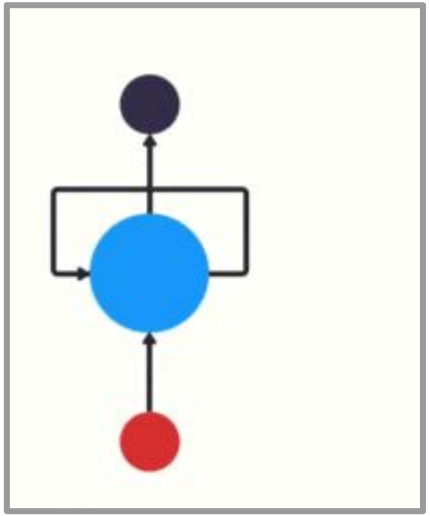
May 7th, 2020

RNNs = Recurrent Neural Networks

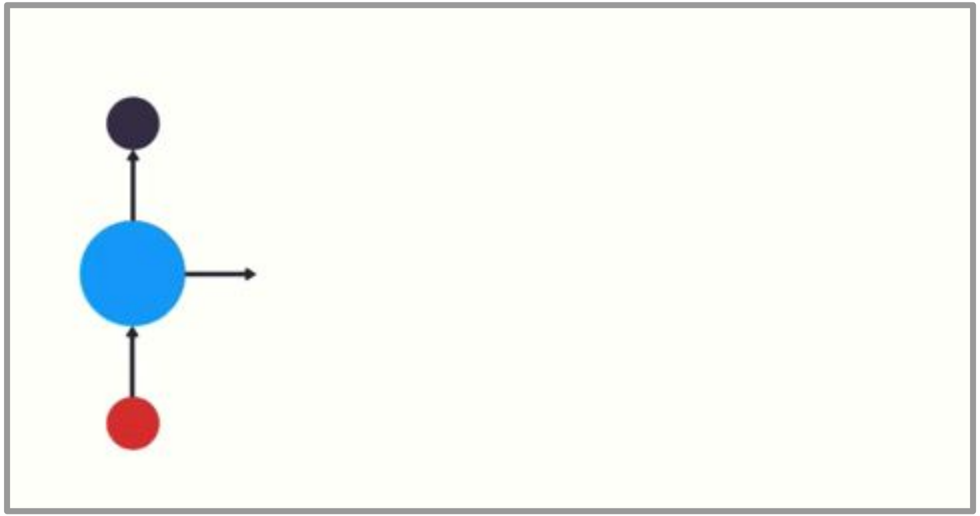
Feed Forward



Recurrent



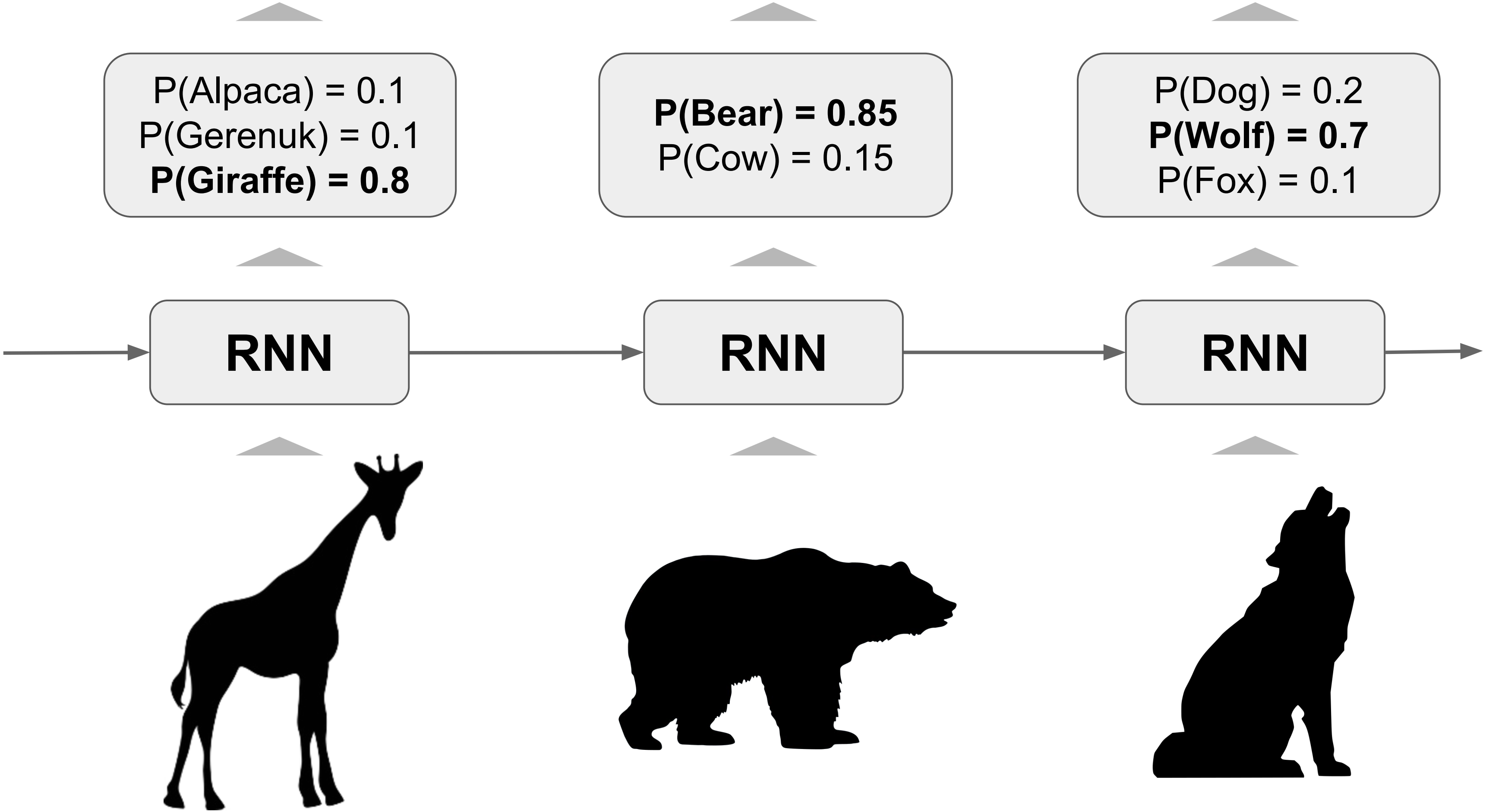
Unrolling RNN to Feed-Forward Networks



Fixed Input
Fixed Output
Independent Samples

Dynamic Input
Dynamic Output
Dependent Samples

Qiyang Hu



P(Alpaca) = 0.1
P(Gerenuk) = 0.1
P(Giraffe) = 0.8

P(Bear) = 0.85
P(Cow) = 0.15

P(Dog) = 0.2
P(Wolf) = 0.7
P(Fox) = 0.1

RNN

RNN

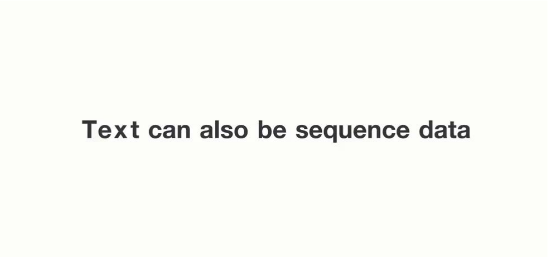
RNN

Qiyang Hu

Strength of RNNs: sequence data predictions

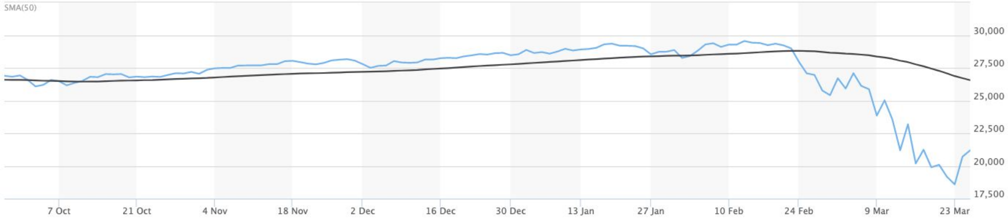
- Audio content in speech

- Text content in articles

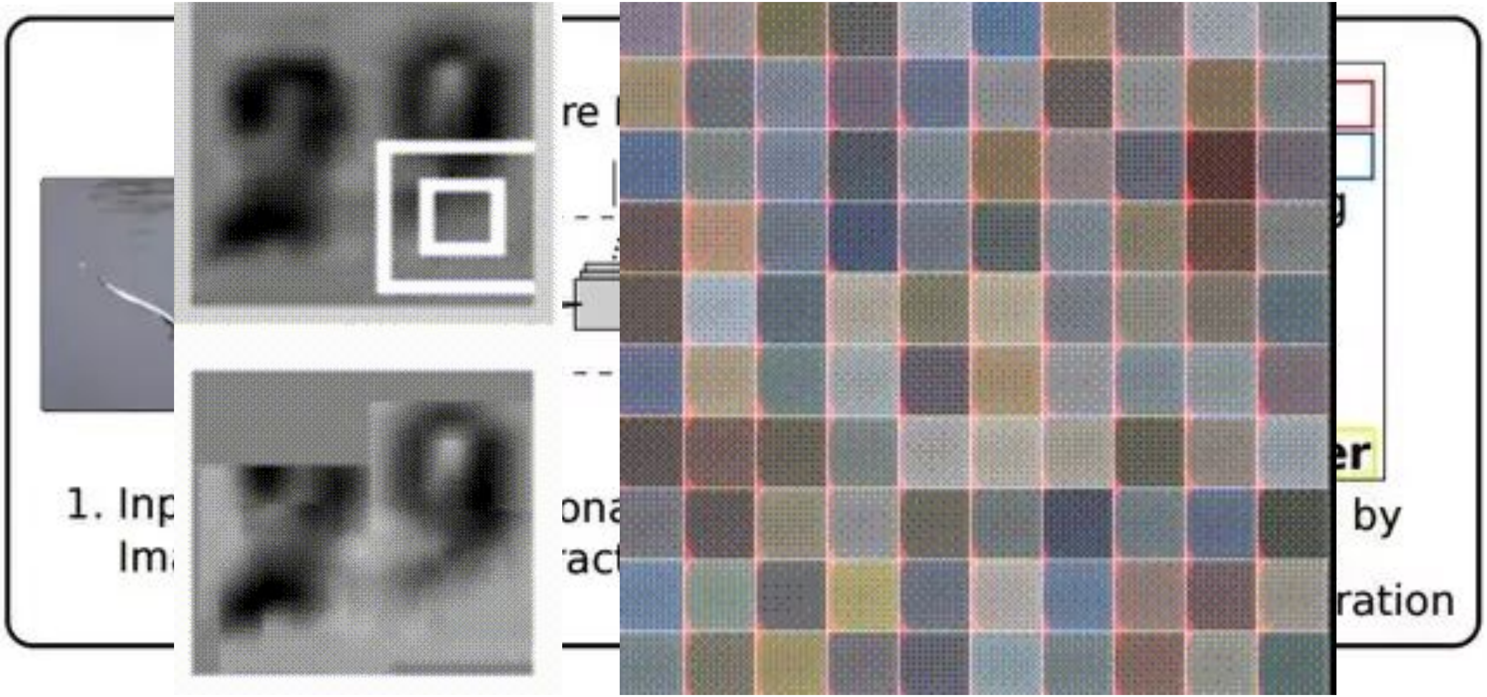


Text can also be sequence data

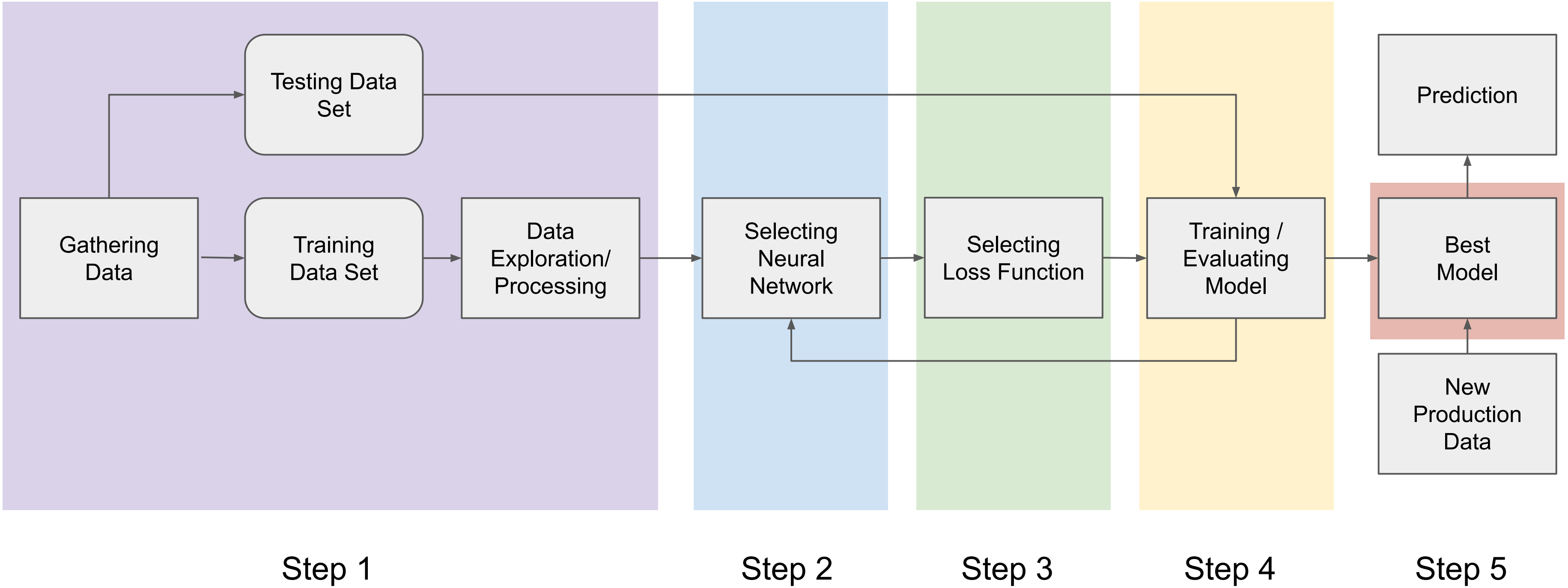
- Stock price prediction



Sequential processing in absence of sequences

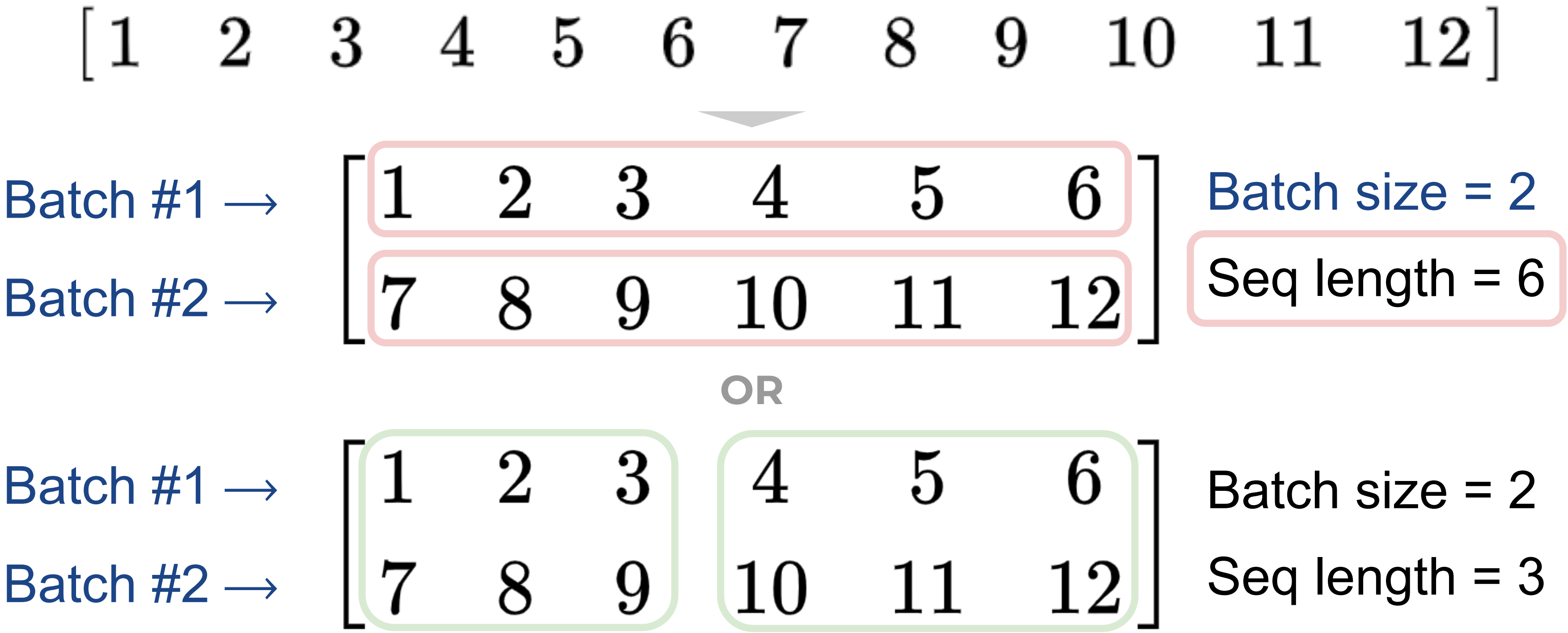


Workflow for a deep learning project



Qiyang Hu

Sequence Batching



$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{bmatrix}$$

$$\begin{array}{rc} \text{Batch \#1} \longrightarrow & \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ \text{Batch \#2} \longrightarrow & 7 & 8 & 9 & 10 & 11 & 12 \end{bmatrix} \end{array}$$

Batch size = 2

Seq length = 6

OR

$$\begin{array}{rc} \text{Batch \#1} \longrightarrow & \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ \text{Batch \#2} \longrightarrow & 7 & 8 & 9 & 10 & 11 & 12 \end{bmatrix} \end{array}$$

Batch size = 2

Seq length = 3

Qiyang Hu

In our demo

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{bmatrix}$$

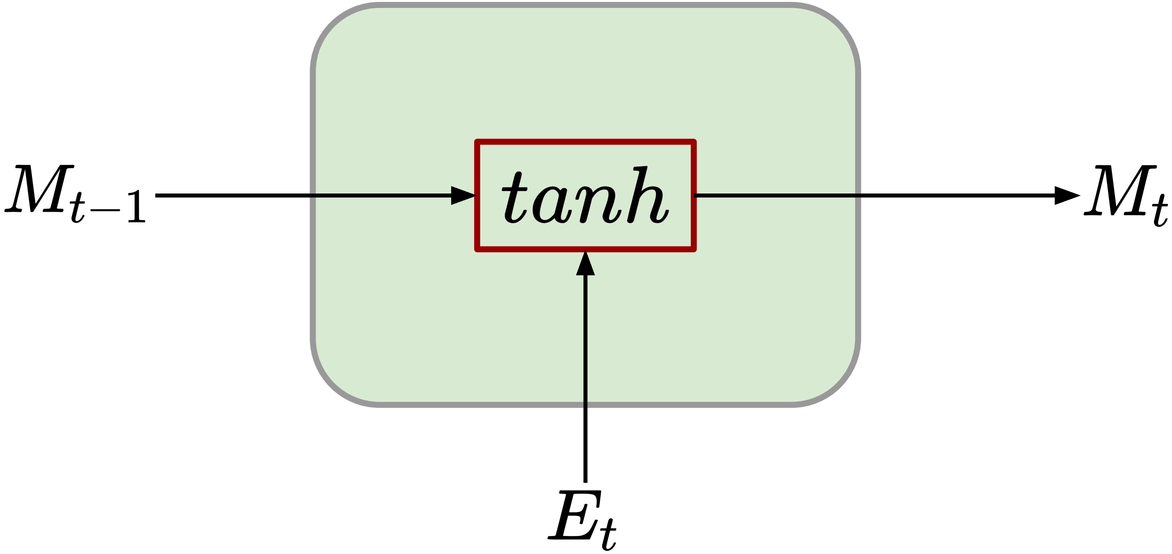
$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{bmatrix}$$

Batch size = 5

Seq length = 8

RNN Inside

$$M_t = \tanh(W_{iM} E_t + b_{iM} + W_{MM} M_{t-1} + b_{MM})$$



Different Applications of RNNs

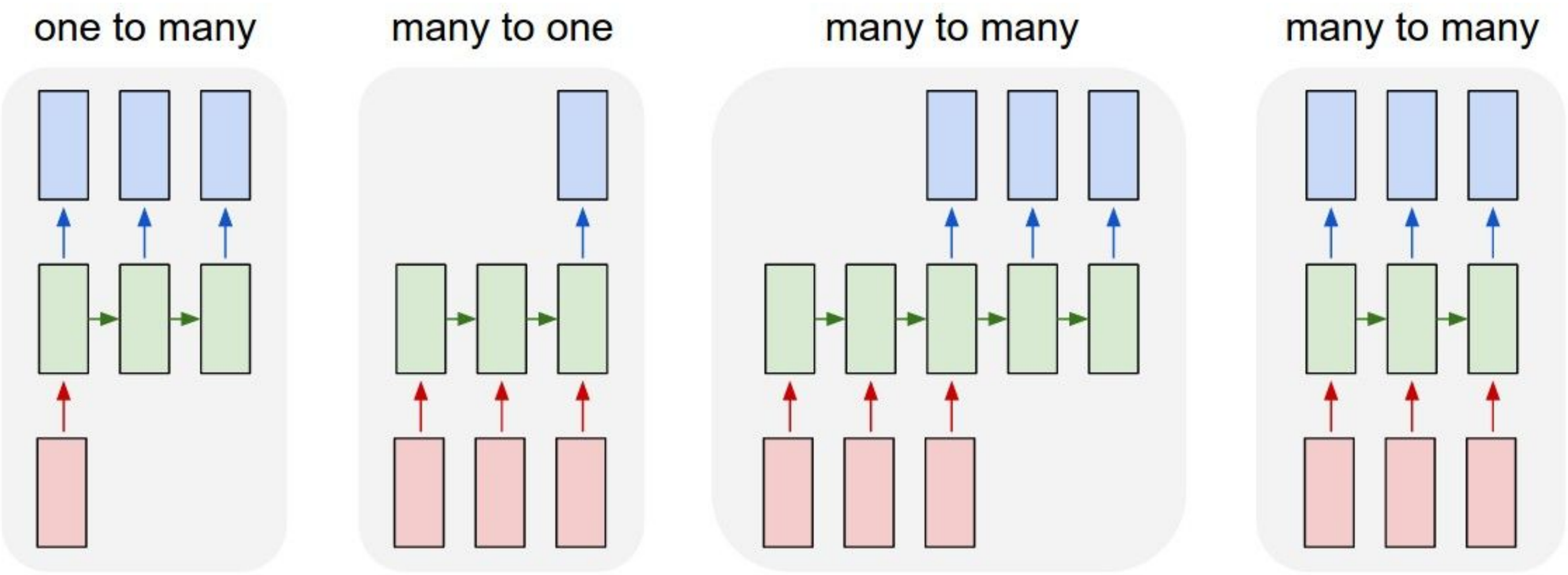


Image Captioning

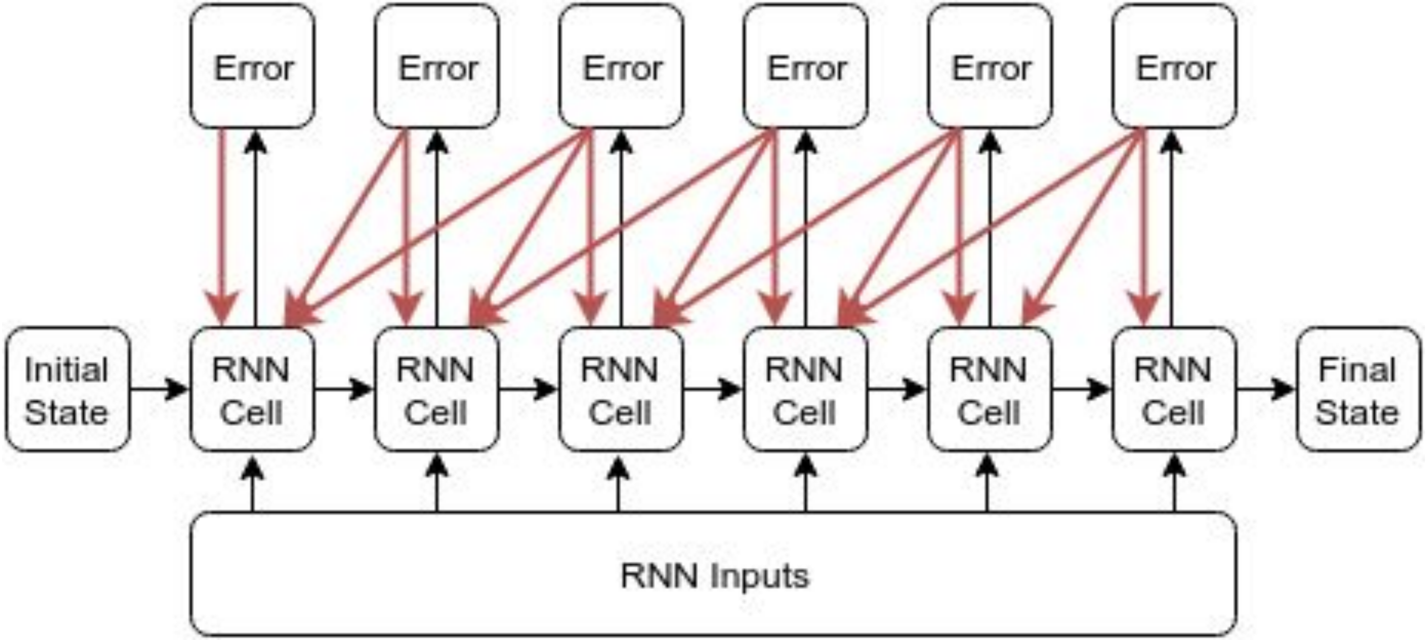
Sentiment Analysis

Machine Translation

Video Classification

Source

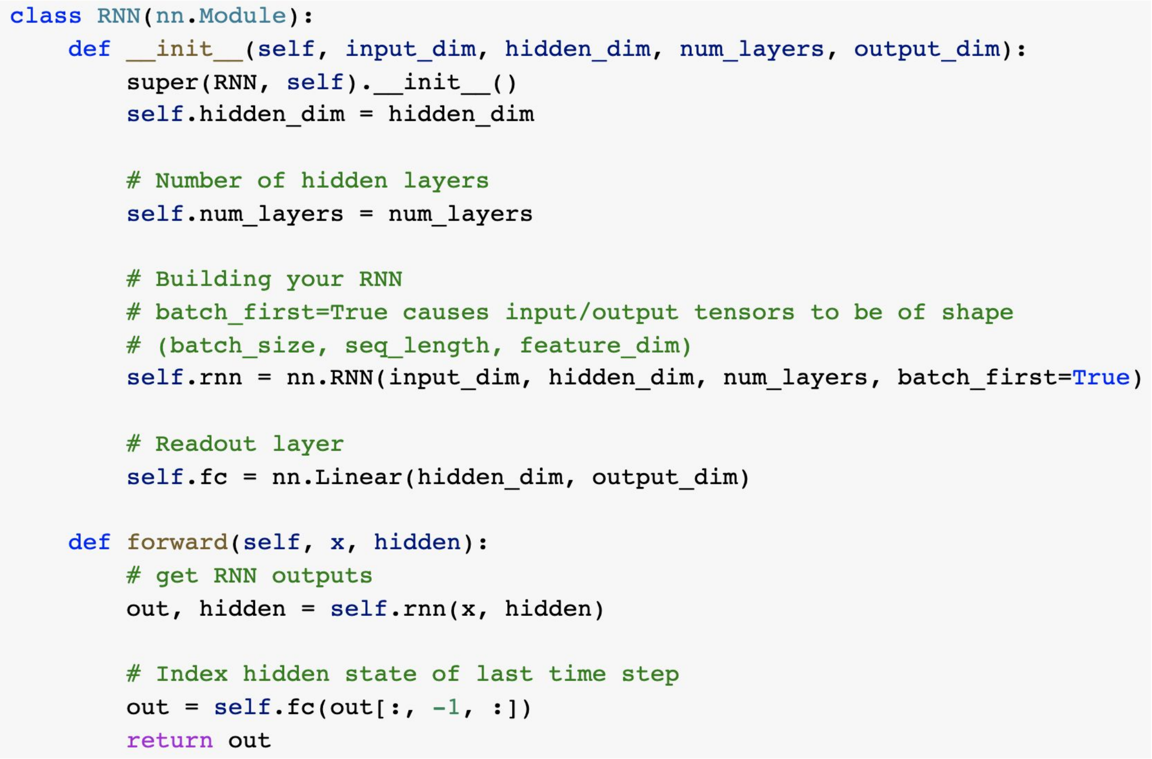
Backpropagation Through Time (BPTT)



Source

RNN Computation in PyTorch

- Defining an RNN
  - Create an RNN layer
  - Add a fully-connected layer
- torch.nn.RNN
  - input_dim
  - hidden_dim
  - output_dim
  - num_layers
  - Batch_first
- In forward function
  - Initialize hidden state
  - BPTT
  - Shape output

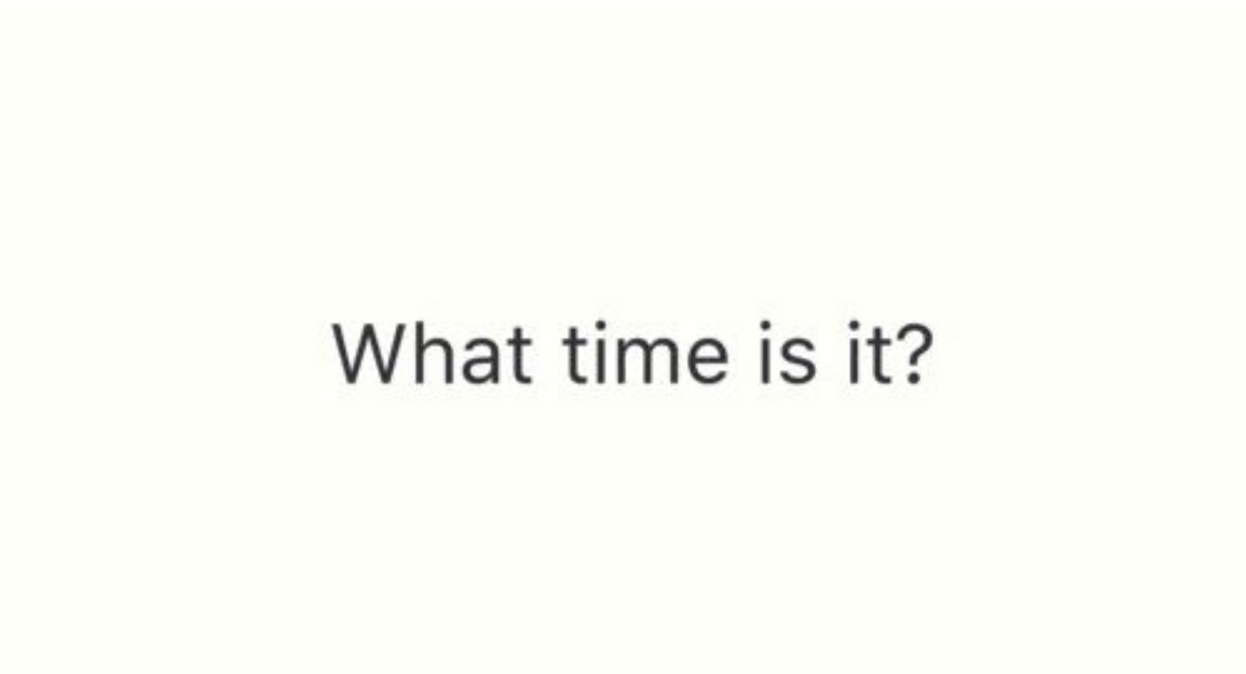


```python
class RNN(nn.Module):
    def __init__(self, input_dim, hidden_dim, num_layers, output_dim):
        super(RNN, self).__init__()
        self.hidden_dim = hidden_dim

        # Number of hidden layers
        self.num_layers = num_layers

        # Building your RNN
        # batch_first=True causes input/output tensors to be of shape
        # (batch_size, seq_length, feature_dim)
        self.rnn = nn.RNN(input_dim, hidden_dim, num_layers, batch_first=True)

        # Readout layer
        self.fc = nn.Linear(hidden_dim, output_dim)

    def forward(self, x, hidden):
        # get RNN outputs
        out, hidden = self.rnn(x, hidden)

        # Index hidden state of last time step
        out = self.fc(out[:, -1, :])
        return out
```

Colab Hands-on

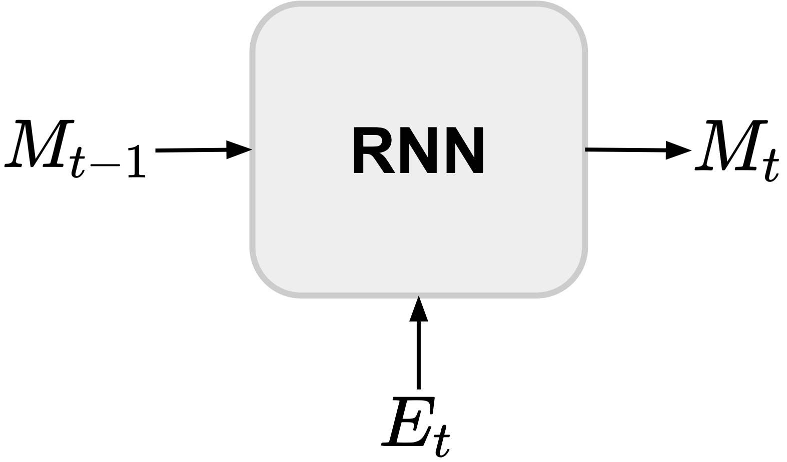
bit.ly/LDL_04

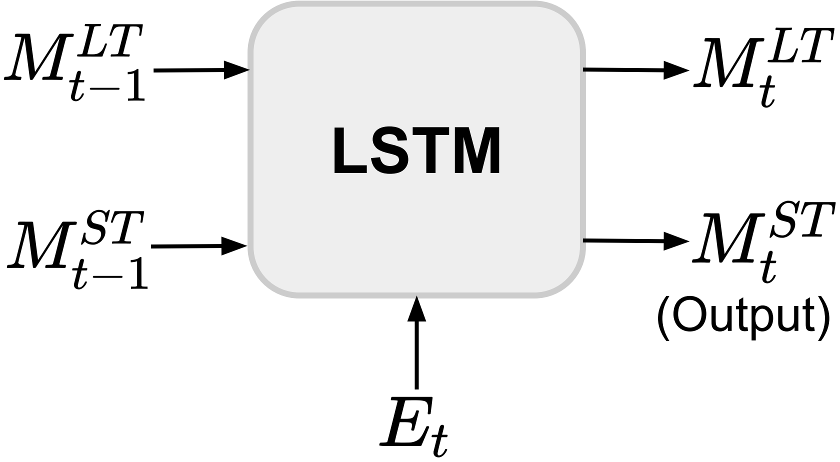
Problem in Vanilla RNN: short term of memory



What time is it?

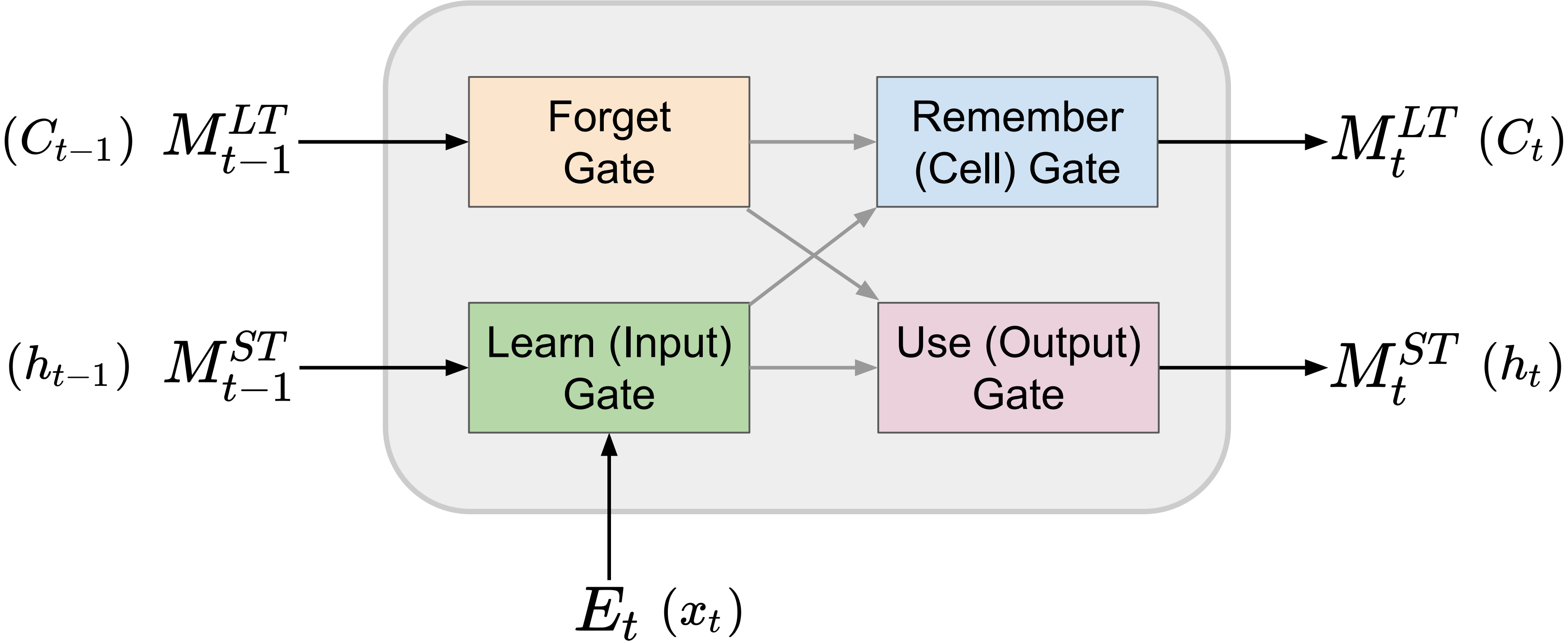
From RNN to Long Short-Term Memory (LSTM)





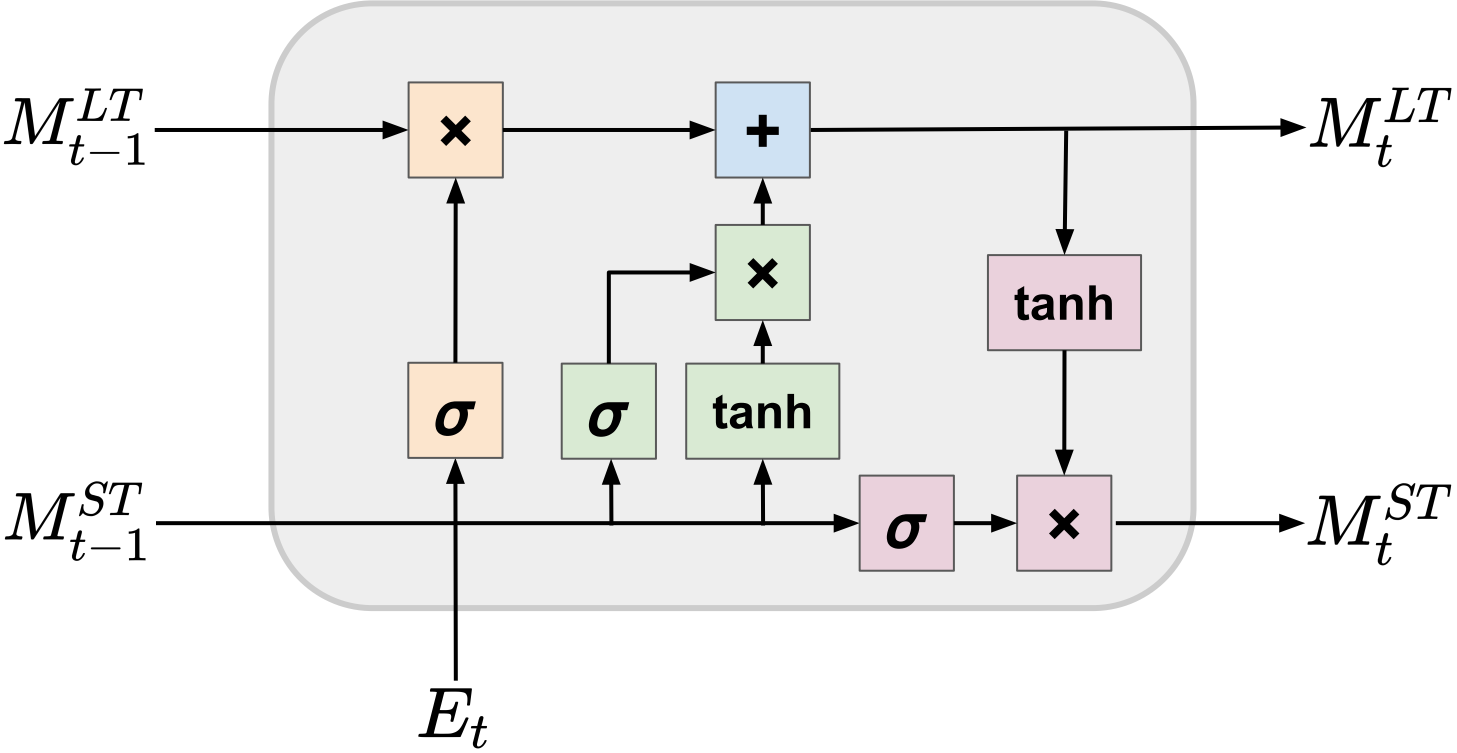
Qiyang Hu

Basic Ideas of LSTM



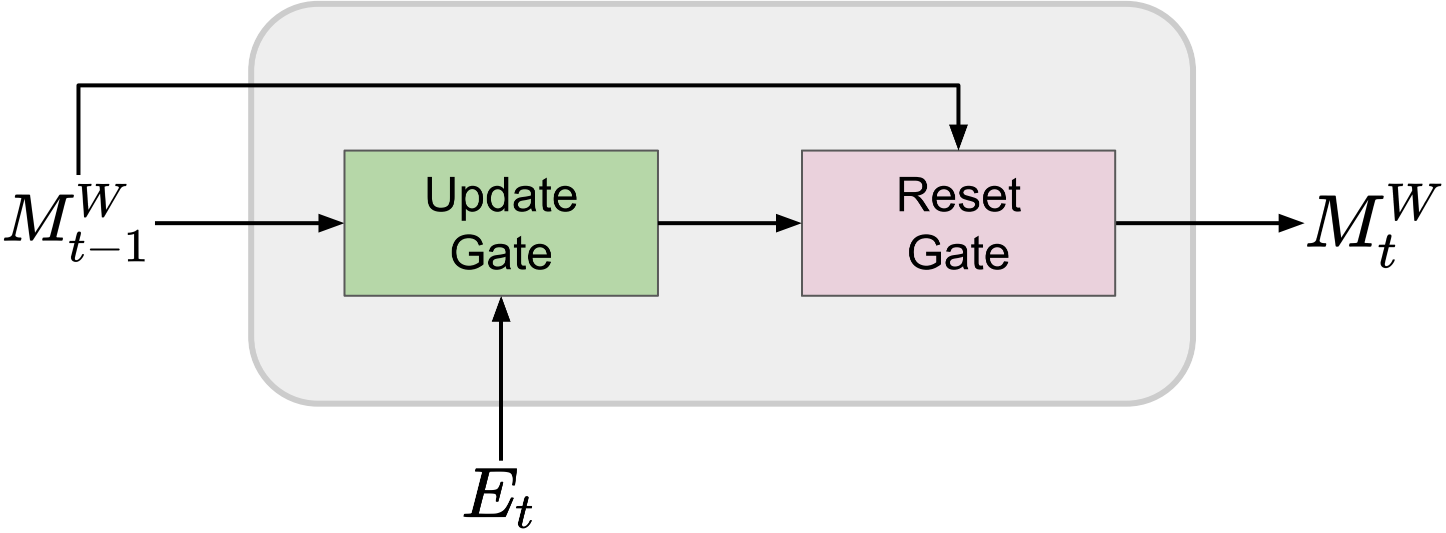
Qiyang Hu

Architectures of LSTM



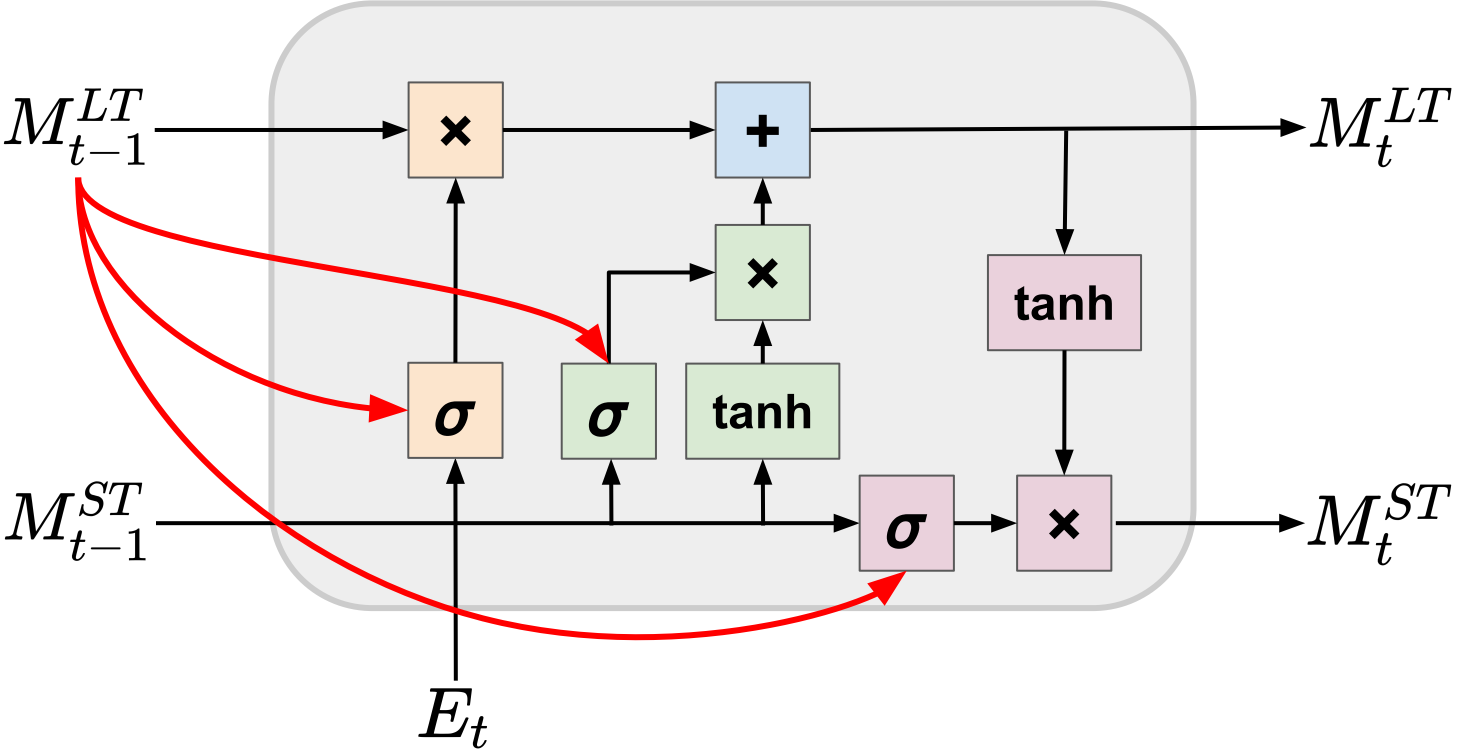
Qiyang Hu

Other architectures: Gated Recurrent Unit (GRU)



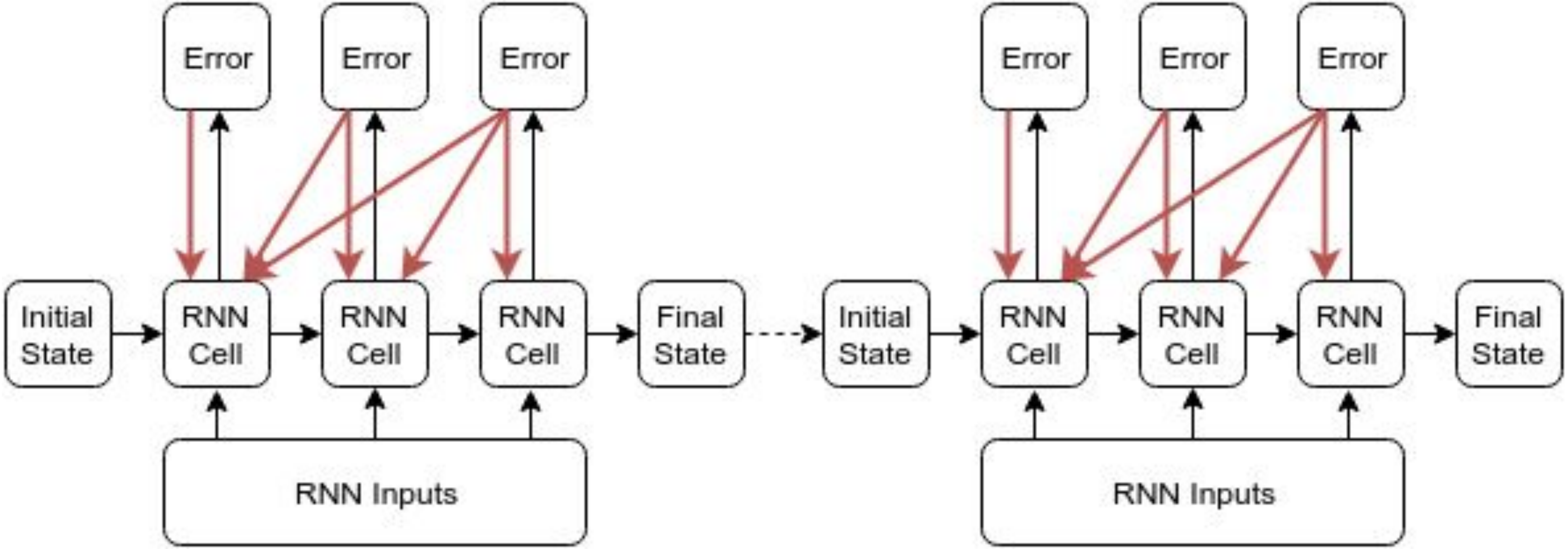
Qiyang Hu

Other architectures: Peephole LSTM



Qiyang Hu

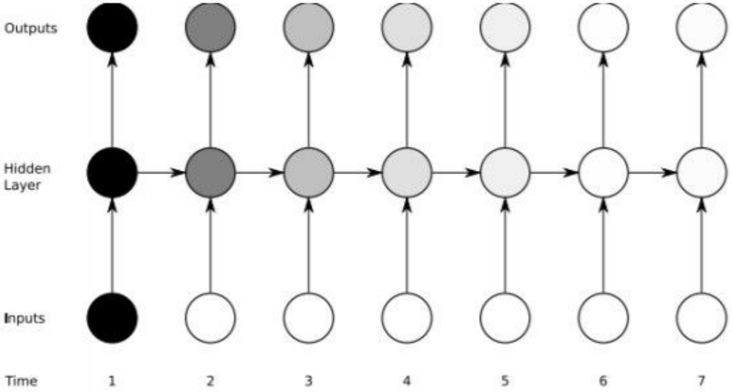
Truncated Backpropagation Through Time (TBPTT)



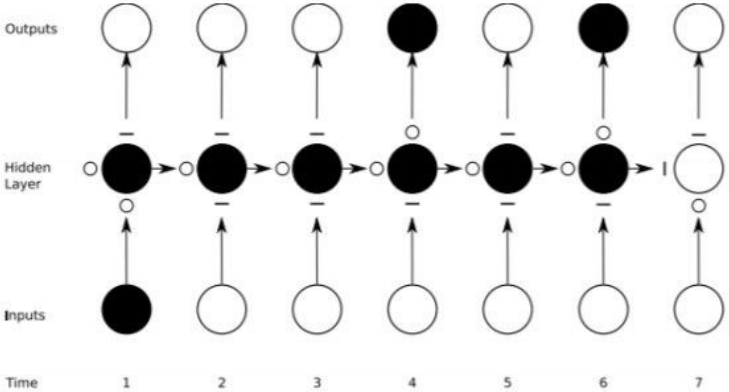
Source

LSTMs reduce vanishing gradient problem

Standard Recurrent Network



LSTM Network



Graves et al 2013

- The darker the shade, the greater the sensitivity
- The sensitivity decays exponentially over time as new inputs overwrite the activation of hidden unit and the network 'forgets' the first input

Colab Hands-on

bit.ly/LDL_04

More words on time-series predictions

- Inspect the uncertainty in the series
  - Stochastic (truly random)
  - Epistemic (process too complex, looks like random)

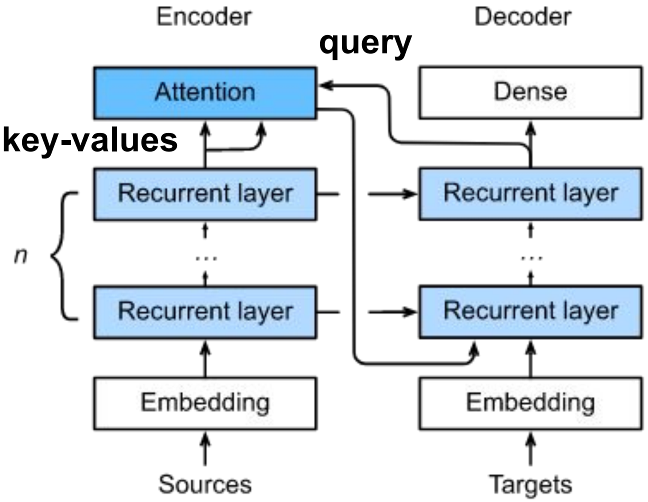
- ARIMA-type model:
  - Implicit Gaussian assumptions
  - Parametric: to predict an individual time series (e.g. *p*, *d* & *q*)
  - Better for small dataset
  - Better results in forecasting short term
  - Python module statsmodels: `from statsmodels.tsa.arima_model import ARIMA`

- RNN-based model:
  - “Non”-parametric: possibly for unknown series
  - Better for large dataset
  - LSTM: better results in forecasting long term

Attention is all you need!

- **Seq2Seq**
  - Introduce weights for encoding outputs

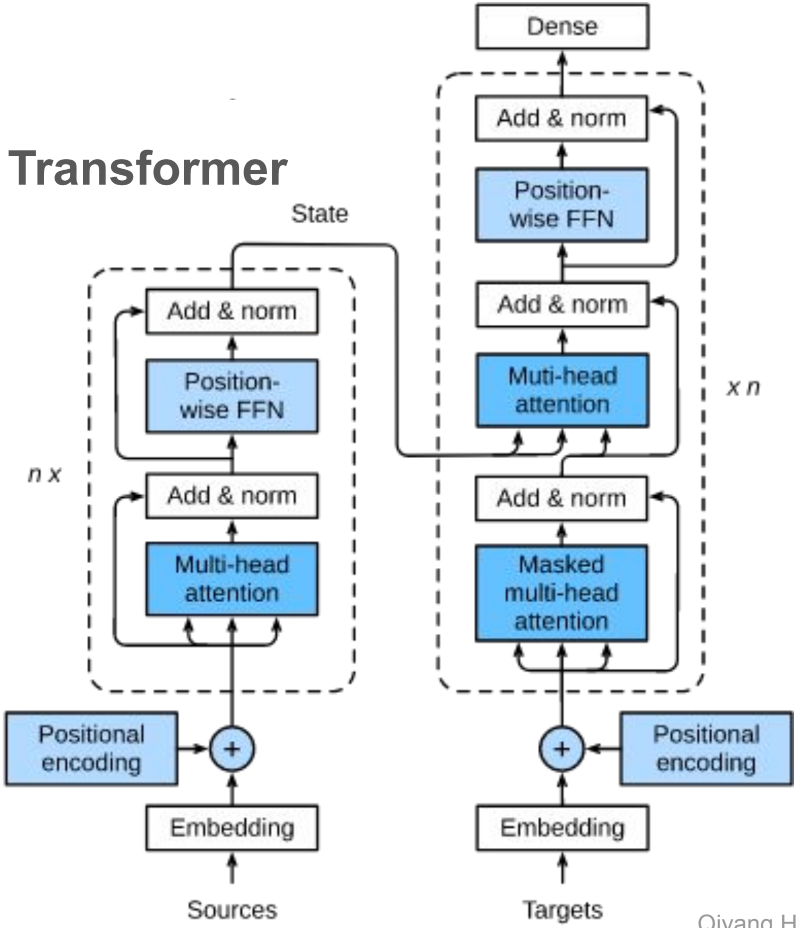
$$r = \sum_{i=1}^{n} \alpha(k_i, q) \cdot v_i$$

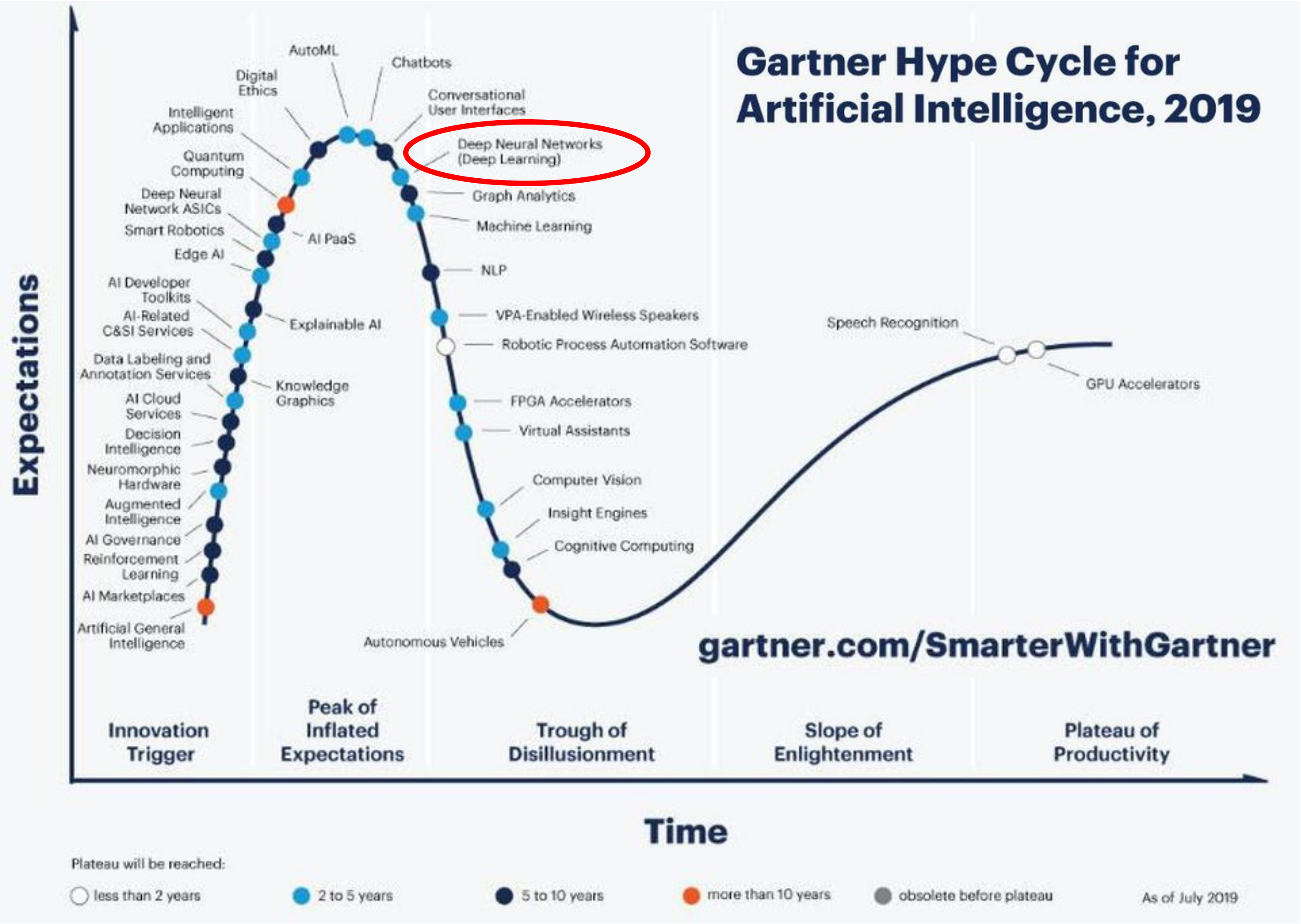


query

key-values

- **Transformer**




Gartner Hype Cycle for
Artificial Intelligence, 2019
Expectations
AutoML
Digital Ethics
Chatbots
Intelligent Applications
Conversational User Interfaces
Quantum Computing
Deep Neural Networks (Deep Learning)
Deep Neural Network ASICs
Graph Analytics
Smart Robotics
Machine Learning
AI PaaS
Edge AI
NLP
AI Developer Toolkits
AI-Related C&SI Services
Explainable AI
VPA-Enabled Wireless Speakers
Robotic Process Automation Software
Speech Recognition
Data Labeling and Annotation Services
GPU Accelerators
Knowledge Graphics
AI Cloud Services
FPGA Accelerators
Decision Intelligence
Virtual Assistants
Neuromorphic Hardware
Augmented Intelligence
Computer Vision
AI Governance
Insight Engines
Reinforcement Learning
Cognitive Computing
AI Marketplaces
Artificial General Intelligence
Autonomous Vehicles
gartner.com/SmarterWithGartner
Innovation Trigger
Peak of Inflated Expectations
Trough of Disillusionment
Slope of Enlightenment
Plateau of Productivity
Time
Plateau will be reached:
○ less than 2 years
● 2 to 5 years
● 5 to 10 years
● more than 10 years
● obsolete before plateau
As of July 2019

We want to hear from you!

bit.ly/2X2phyS

- Contact me for questions or discussions
  - huqy@idre.ucla.edu
  - Office: Math Sci #3330
  - Phone: 310-825-2011