Intro do Deep Learning

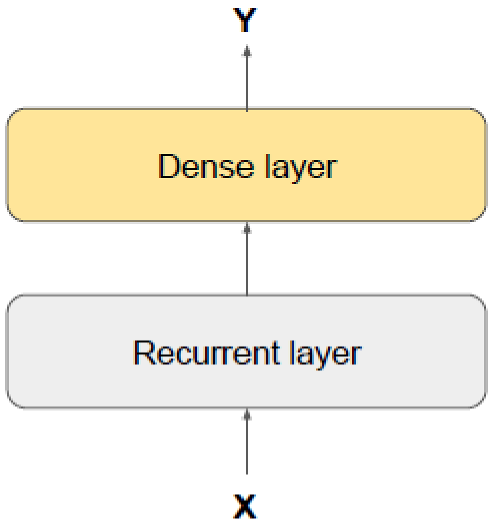
RNN
LSTM
Autoencoder

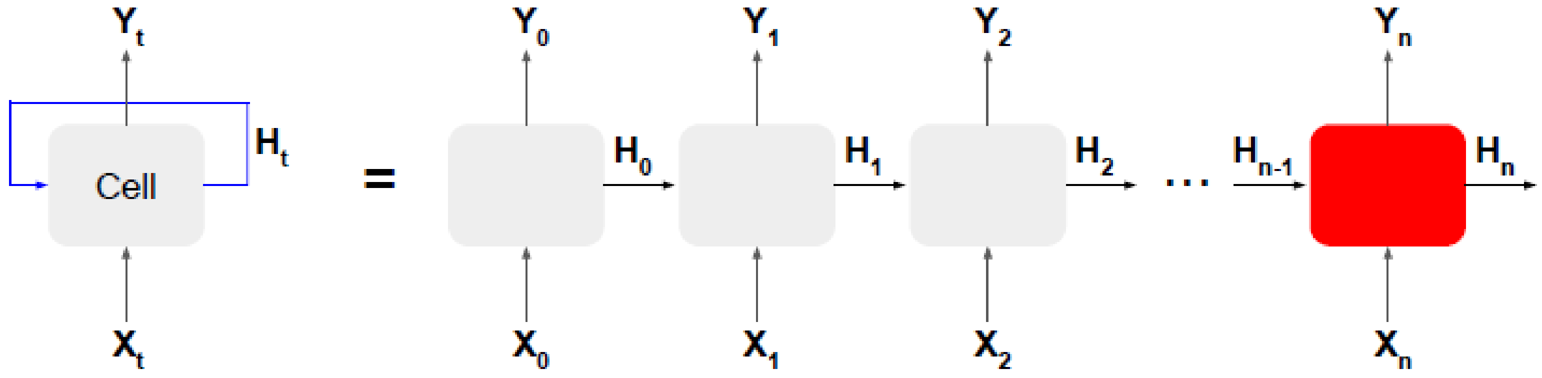
HUAWEI

RNN (Recurrent Neural Network)

RNNs is often used for sequential data, each item is processed in context.

Input Data points one at a time, then predict the next step. Prediction depends on previous data points.





Dense layer: classification
Recurrent layer: Process sequential data in sequential way

Use 1 shell recurrently.
Xt: input
Yt: output
Ht: hidden state (memory, used for next step)
Expand to n-step in a linear way

RNN Data Shape

Shape of X = [batch_size, # steps, # dimensions] = [2, 9, 1]

Input at each step = [batch_size, # dimensions] = [2, 1]

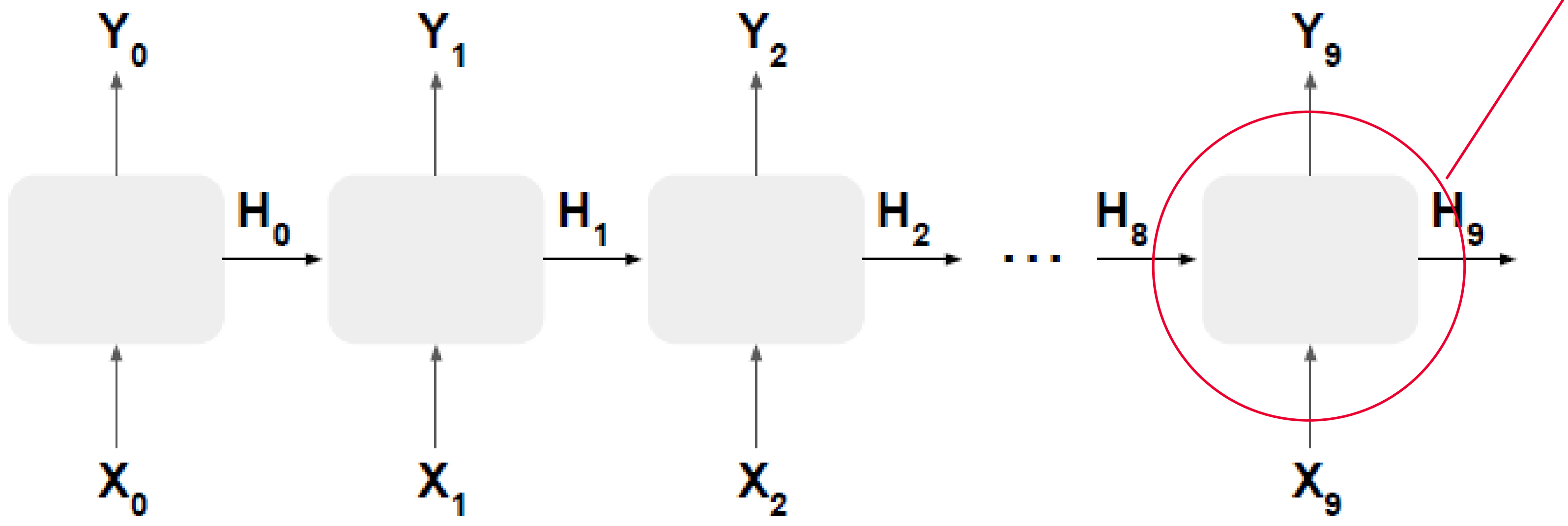
Output at each step = [batch size, # units] = [2, 3]

Output Shape = [batch size, #steps, # units] = [2, 9, 3]

Shape of Yt = Shape of Ht

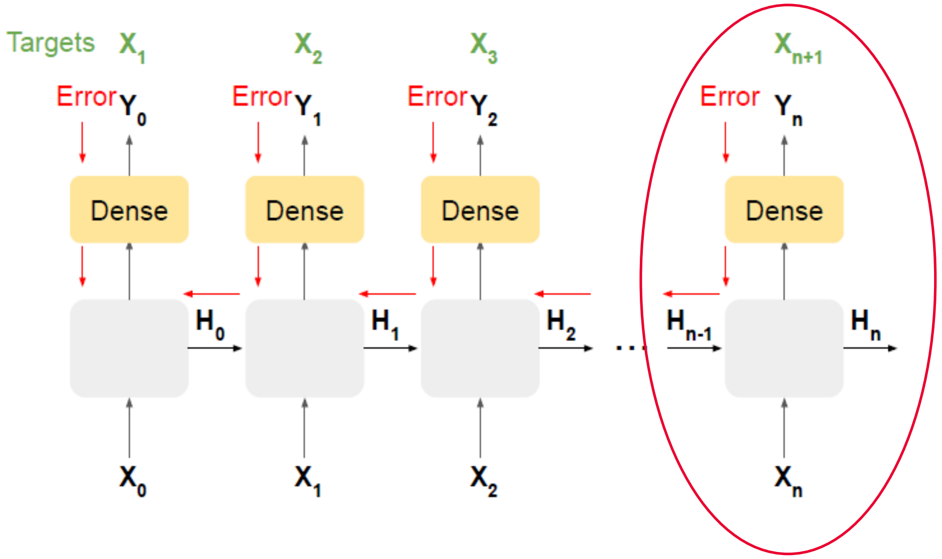
Memory Cell contains:

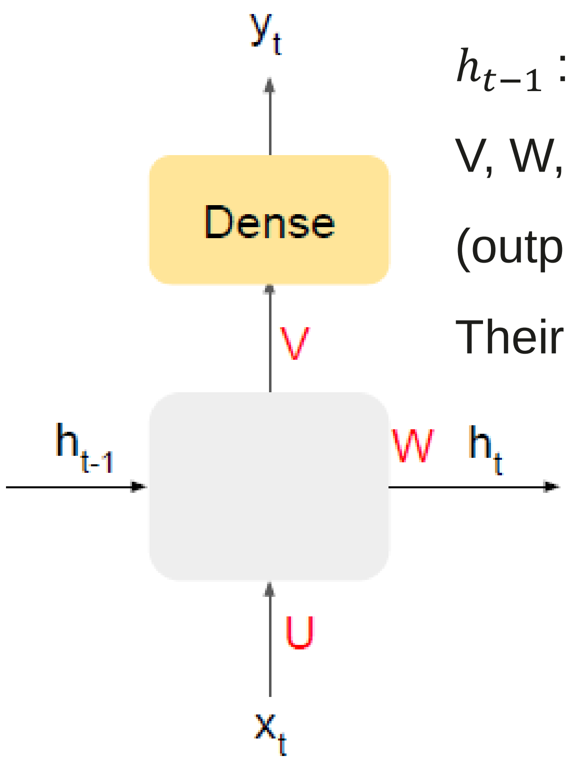
- Dense layer
- Input = state vector + input data
- Activation function = tanh



Backpropagation through time (BPTT)

Output prediction at each time step, then **compare** with target and calculate error, then back propagate the **error** to update the **weight vectors**.
We only care about **the last prediction**.





h_{t-1} : state factor at t-1

V, W, H are weight vectors to calculate y_t (output)

Their relationship is:

$$h_t = f(\boxed{U}x_t + \boxed{W}h_{t-1})$$

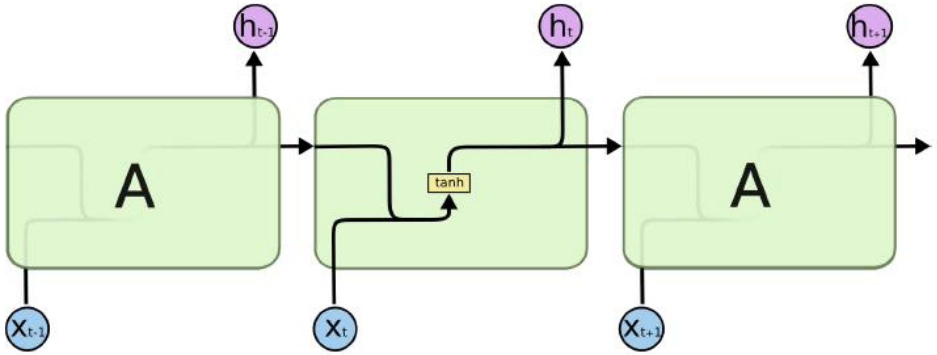
$$y_t = softmax(\boxed{V}h_t)$$

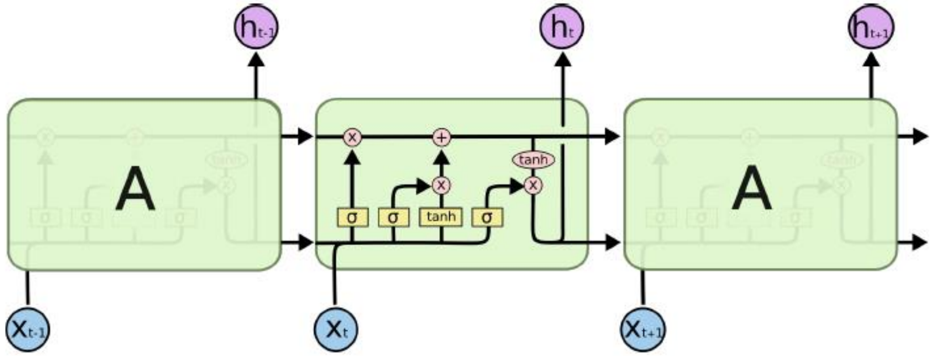
Shortcomings -> LSTM

- No long-term memory
- Network can’t use information from distant past
- Can’t learn patterns with long dependencies

Use **LSTM** to overcome these issues:

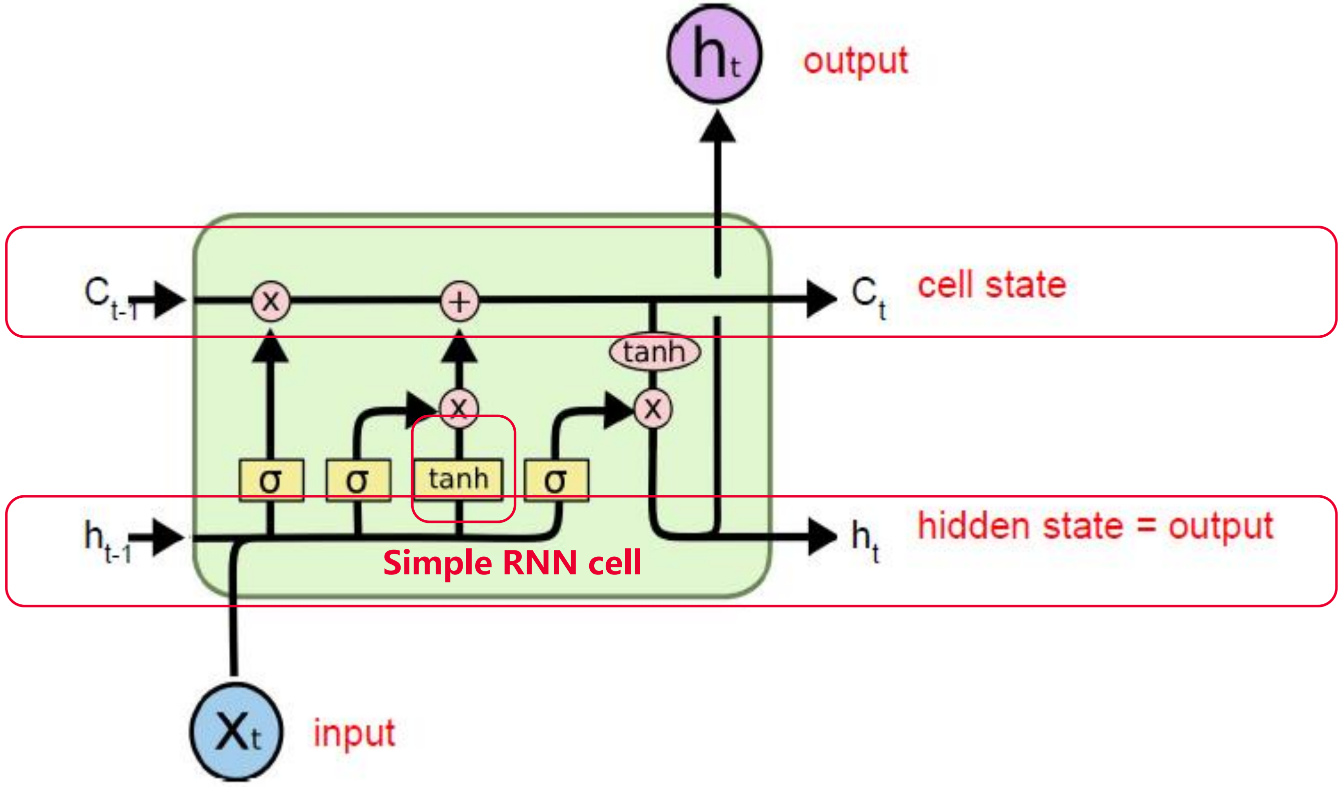
- Special type of RNN
- Can learn long-term patterns



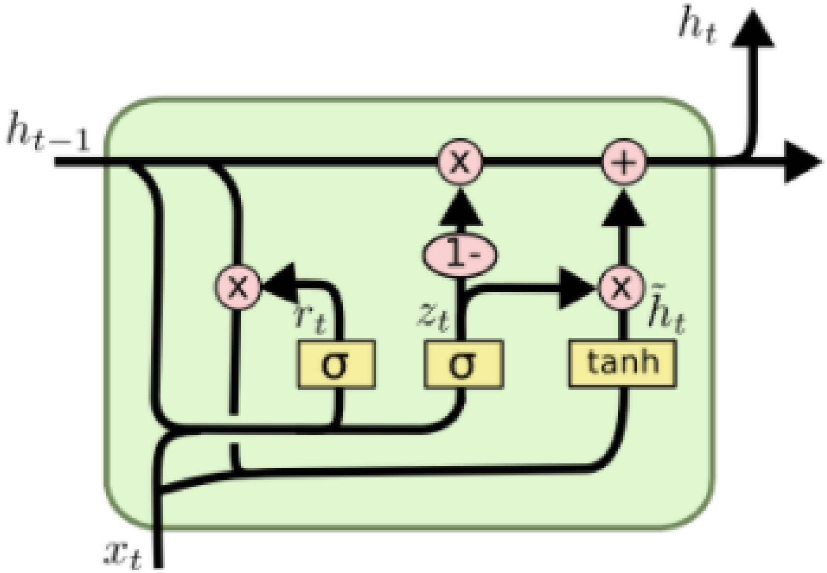


LSTM Cells

- Contains a simple RNN cell
- 1st State vector: **hidden state**: short-term memory
- 2nd state vector: **cell state**: contains information of long-term memory
  - Updated twice
    - **X** : decide what to forget in long terms
    - **+ :** decide what new information to remember
  - Few computation, can stabilize the gradients
- 3 Gates work as filters (make decisions)
  - **Forget Gate**
  - **Input Gate**
  - **Output Gate**



GRU (Gate-Recurrent Unit)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Autoencoder

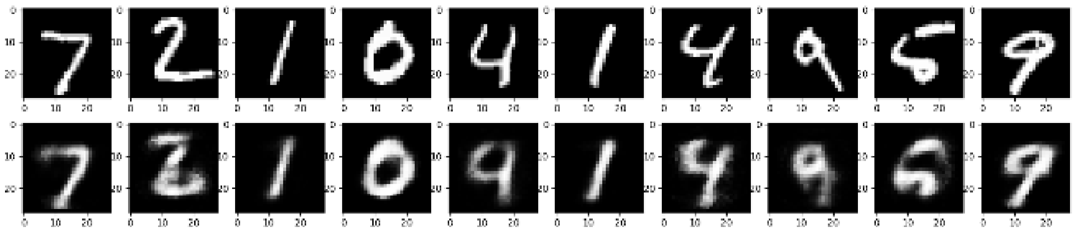
Autoencoder is a data compression algorithm.

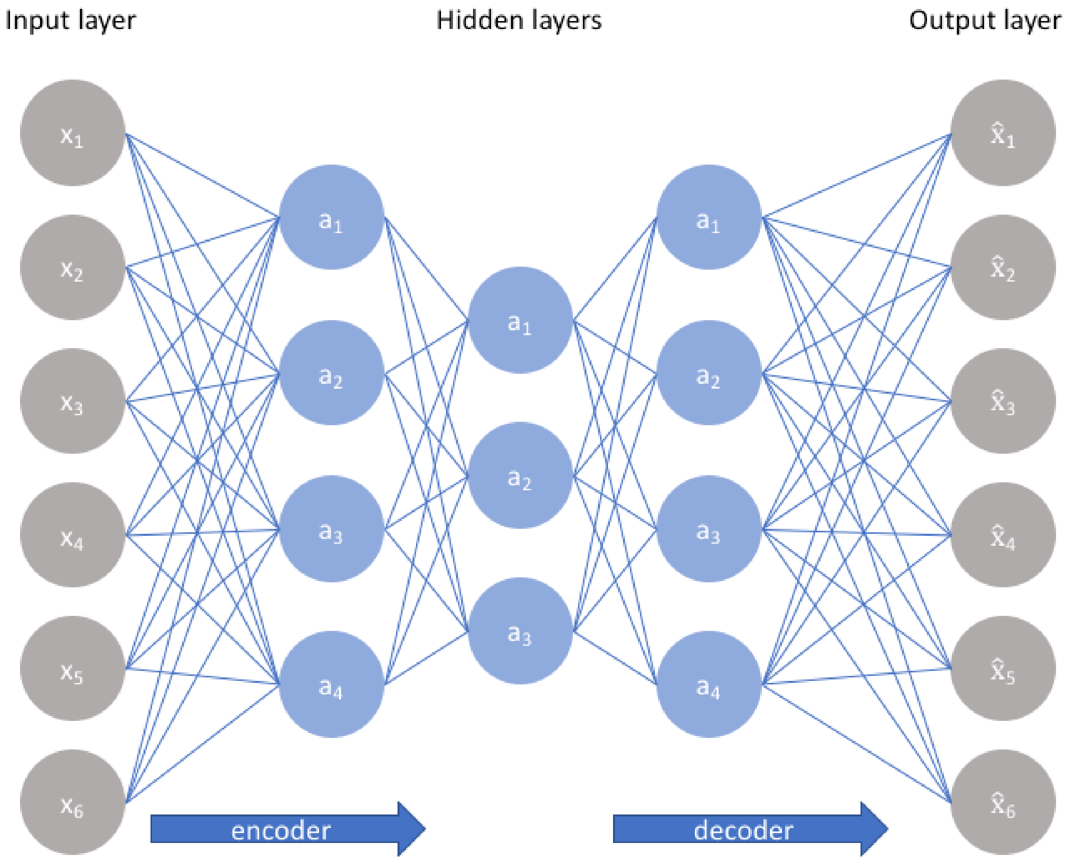
The compression and decompression functions are implemented with neural networks.

- Data Specific: Only able to compress test data similar to trained data
- Lossy: Decompressed outputs will be degraded compared to original inputs

Application

- Data denoising
- Dimensionality reduction




Input layer
Hidden layers
Output layer
x_1
x_2
x_3
x_4
x_5
x_6
a_1
a_2
a_3
a_4
a_1
a_2
a_3
a_1
a_2
a_3
a_4
$\hat{x}_1$
$\hat{x}_2$
$\hat{x}_3$
$\hat{x}_4$
$\hat{x}_5$
$\hat{x}_6$
encoder
decoder