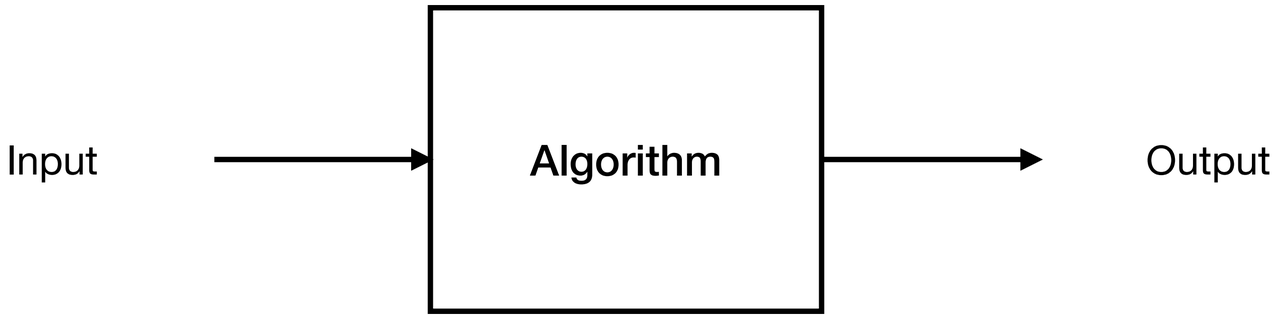
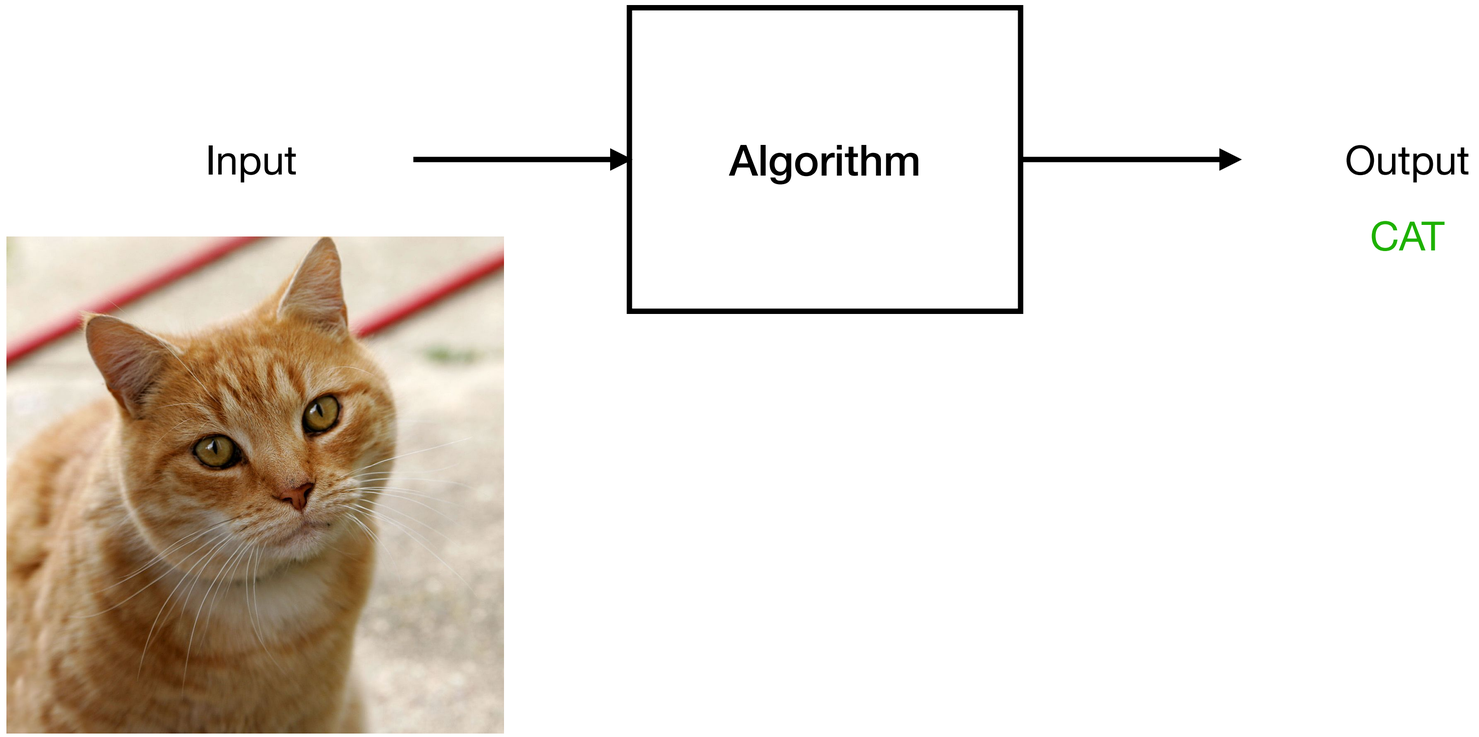
Sequence Prediction

One Step Prediction



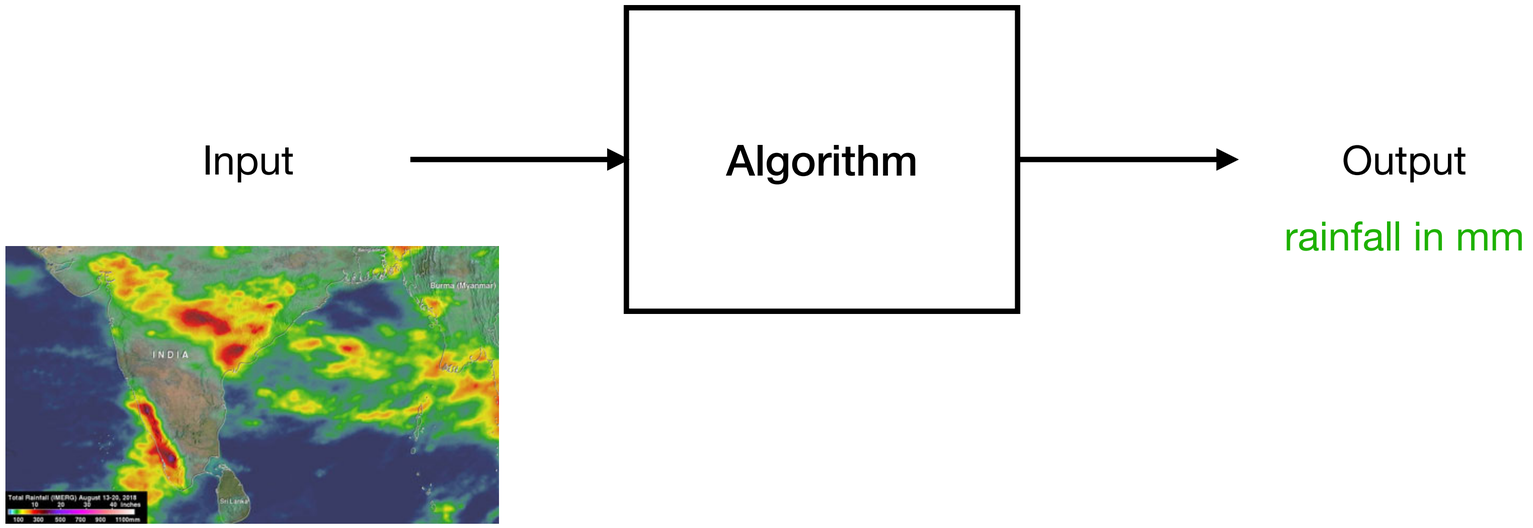
One Step Prediction

Image Classification



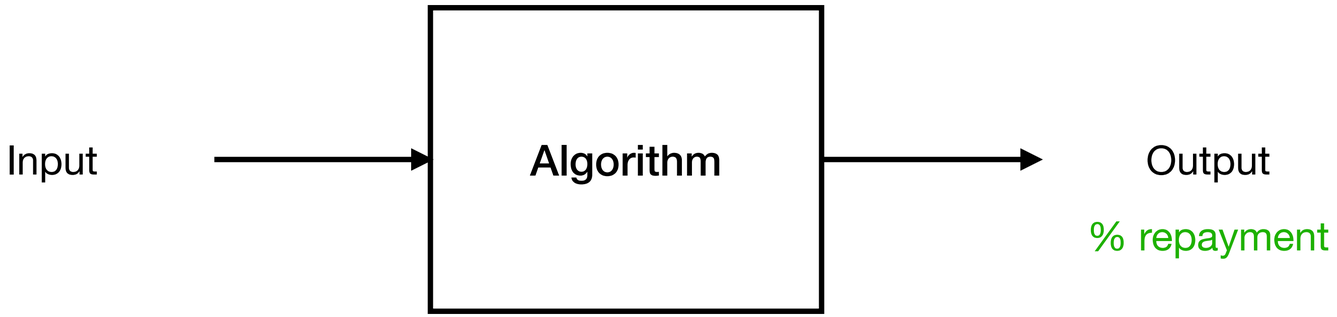
One Step Prediction

Rainfall Prediction



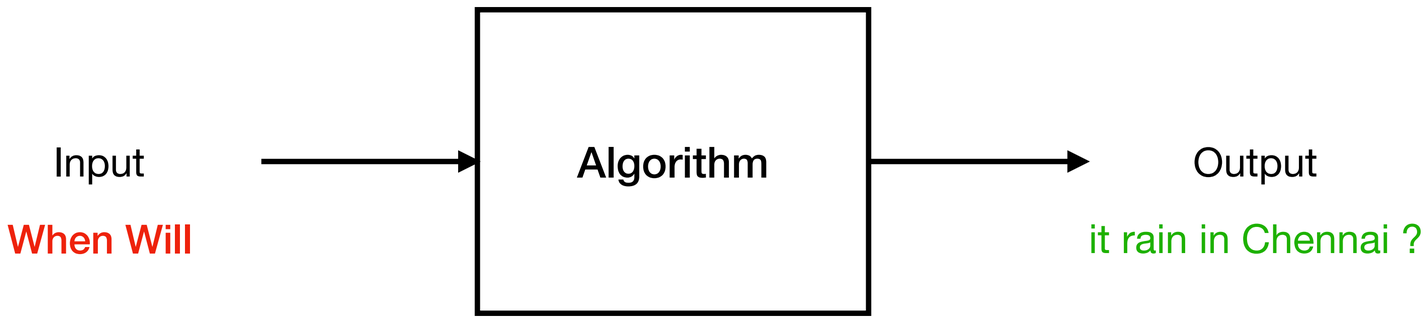
One Step Prediction

Loan Repayment



Sequence Prediction

Sentence Completion

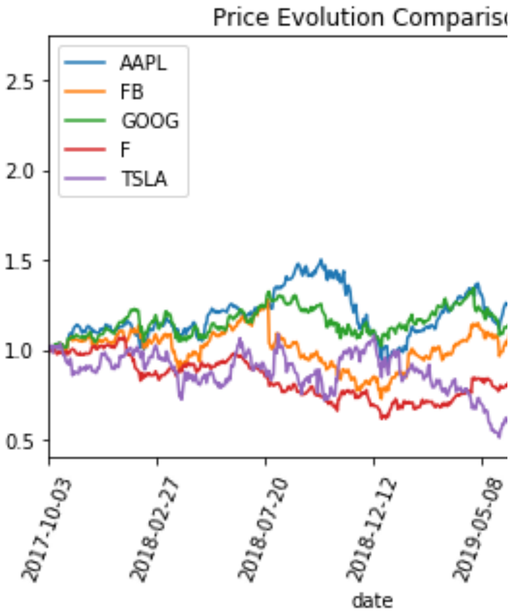


When will it rain in Chennai ?

Sequence Prediction

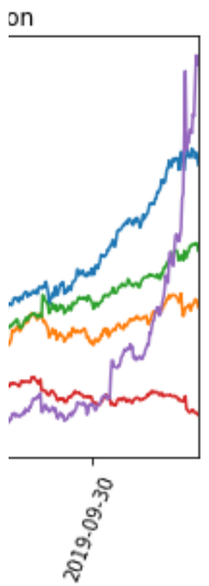
Stock Prediction

Input



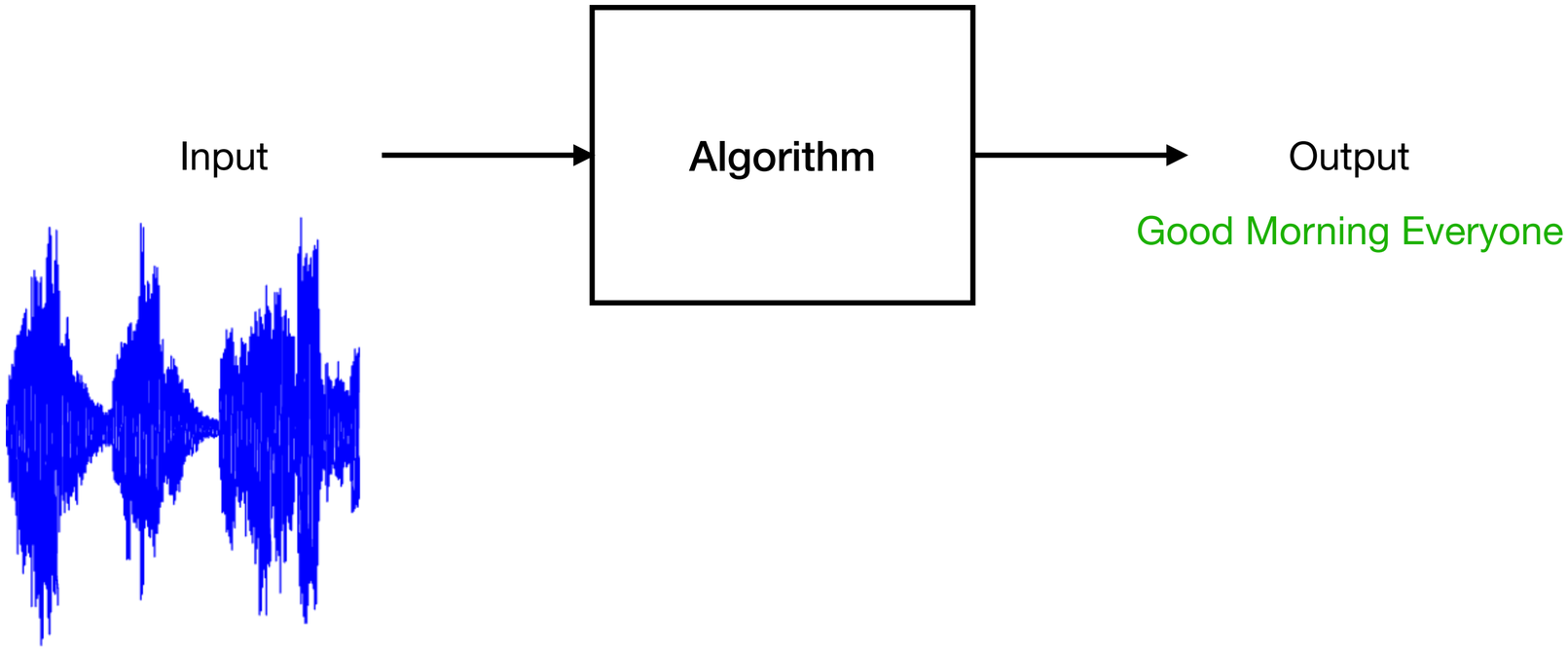
Algorithm

Output



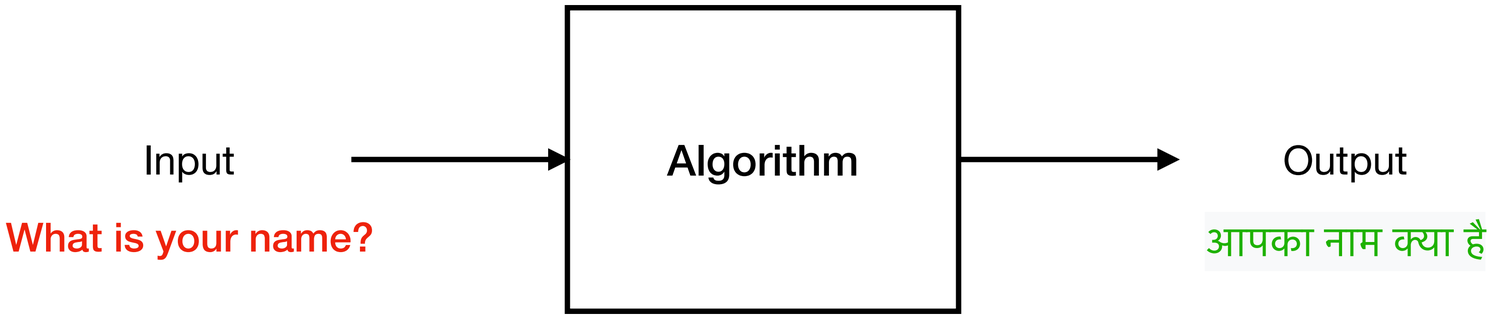
Sequence Prediction

Speech Recognition



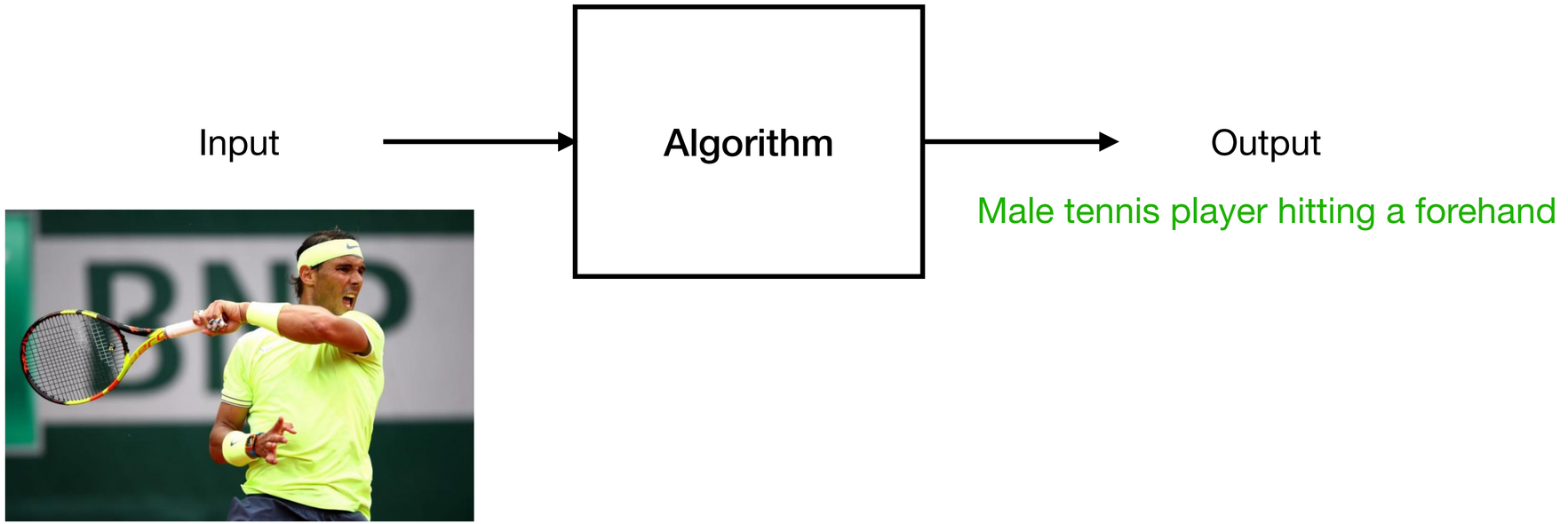
Sequence Prediction

Machine Translation



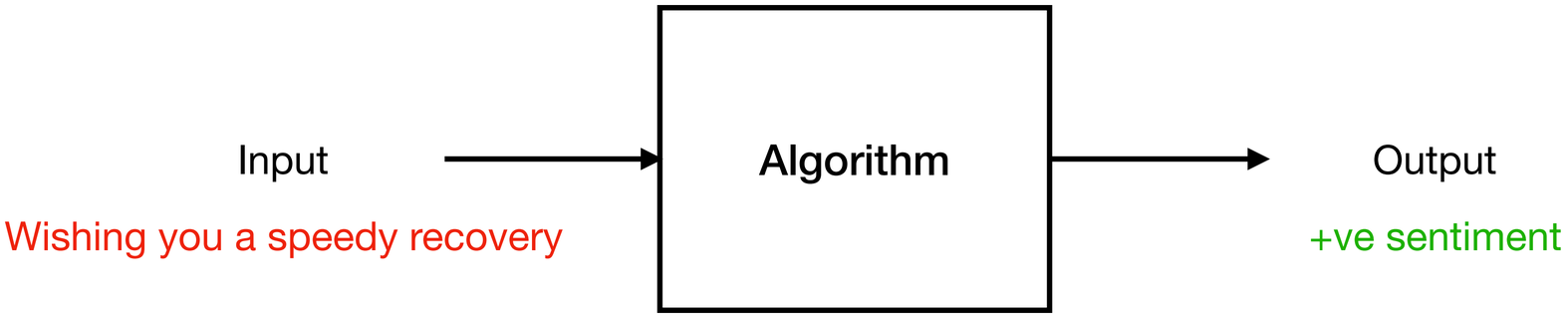
Sequence Prediction

Image Captioning



Sequence Prediction

Sentiment Analysis

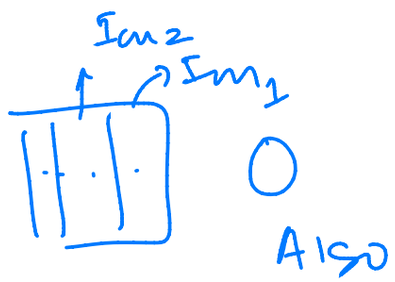


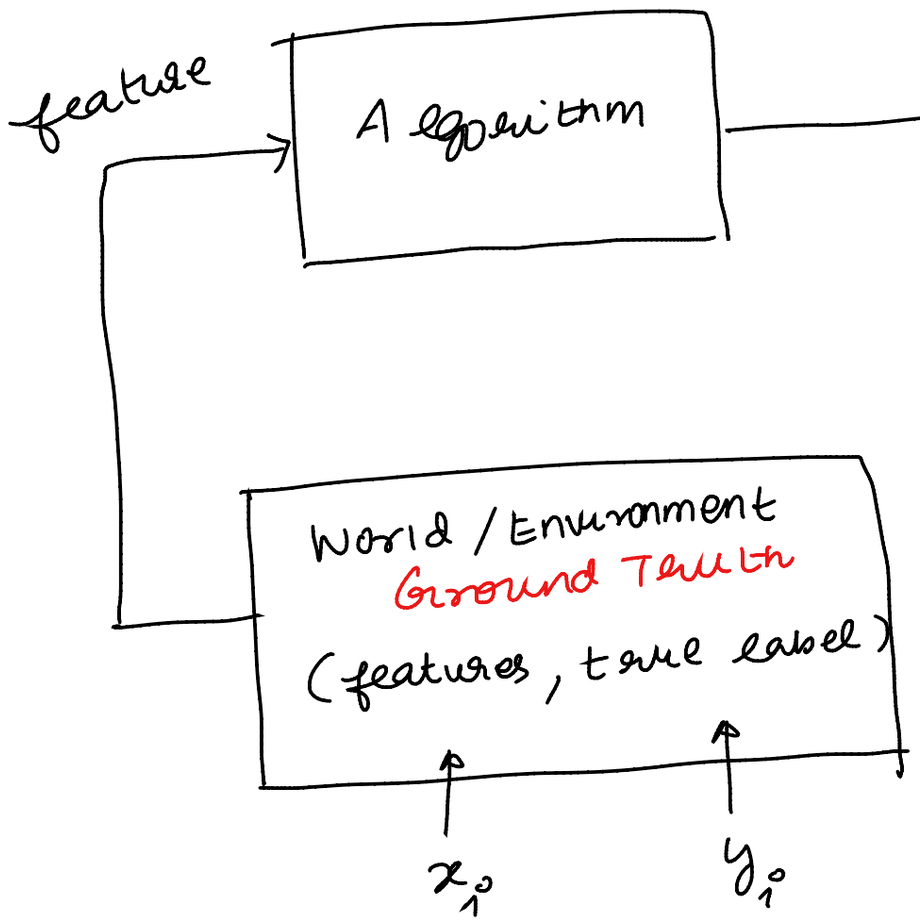
- Different types of sequence prediction problems
- Place RNNs, LSTMs and Transformers amongst various other deep neural network architectures
  (these are architectures for sequence prediction)
- List down aspects of sequence prediction
  (gives an idea of how to derived architecture for RNN, LSTM and Transformers)
- Build RNNs from scratch
- Recurrences (linear)
- Backpropagation Through time
- Issues with RNN
- LSTMs
- Transformers

One Shot Prediction

Think about the algorithm that tags images

1) Snapchat
2) Instagram
3) Facebook





Dataset: $(x_i, y_i)_{i=1}^{n}$

Historical Data

Training Data

Image Recognition:

x_i = image $\subseteq \mathbb{R}^d$

(Mega Pixel Camera)

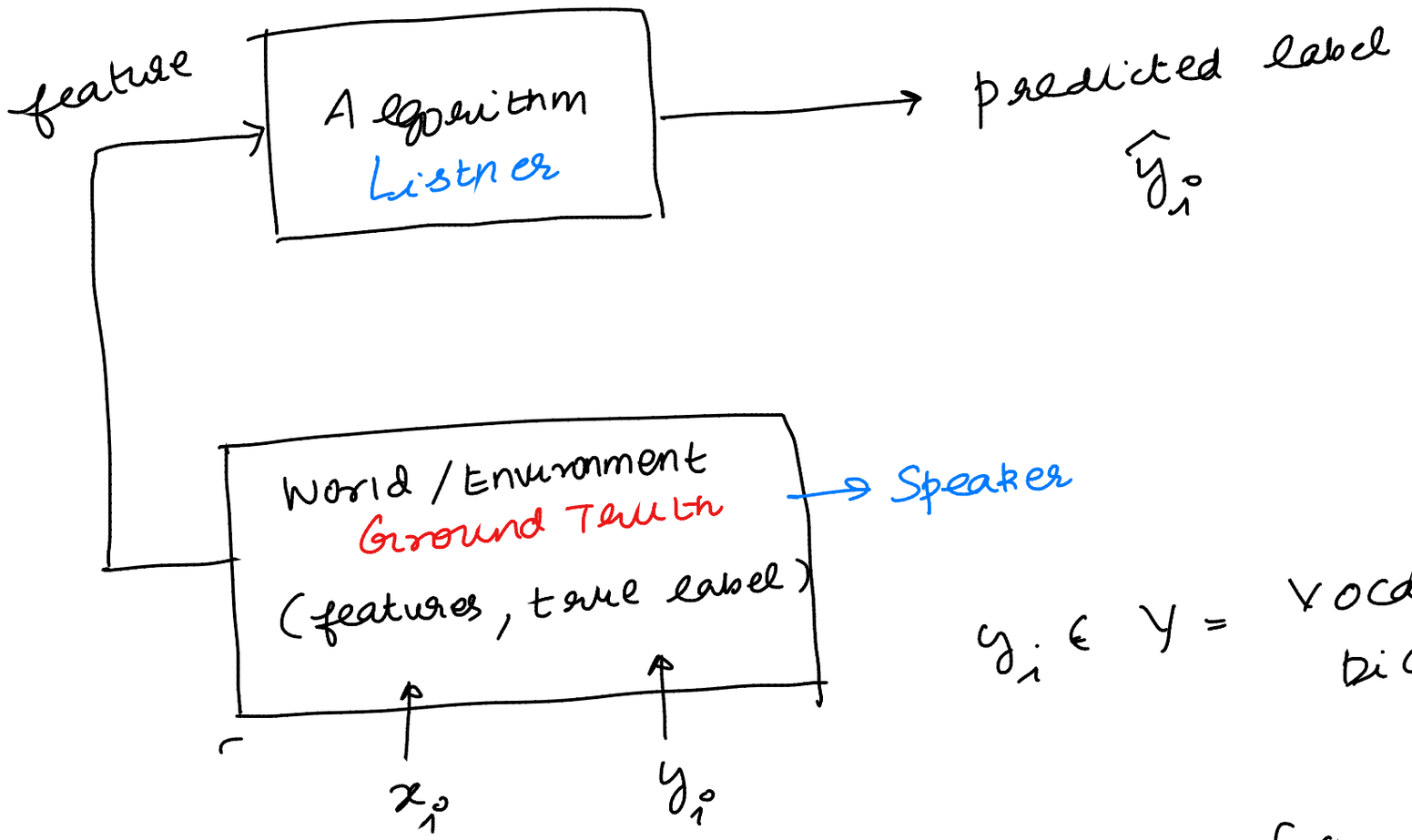
$x_i \in \mathbb{R}^{3 \times 10^6}$

$\mathbb{R}^{3 \times 1000 \times 1000}$

$y_i \in Y = \{$ cat, dog, human, house, ..., train, bus $\}$

ImageNet : 1000 categories classes.

Sequence Prediction



Dataset

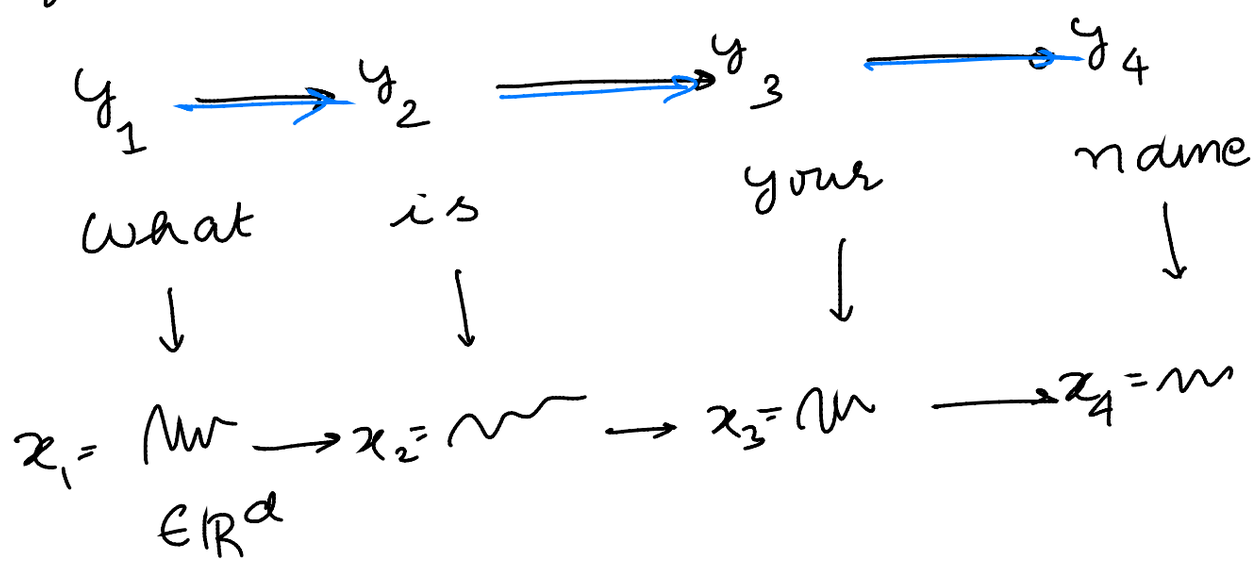
$(x_i, y_i)_{i=1}^{n}$

Historical Training Data

$y_i \in Y$ = Vocabulary / Dictionary

$y_{t+1} = f(y_1, \ldots, y_t)$

Speech Recognition:



NOT Speech Recognition:

once there was a lion

$y_1 \rightarrow y_2 \rightarrow y_3$ → a lion / a boy / a girl

} example shows that text is temporally connected

Stock Prediction:

All the information that dictates dynamics of finance

$y_1 \rightarrow y_2 \rightarrow \cdots \rightarrow y_t$

Price of 100 Stock: $x_1 \quad x_2 \quad \cdots \quad x_t$

$\in \mathbb{R}^{100}$

Observed / watched

Predicted: $\hat{y}_{t+1} = \hat{x}_{t+1}, \cdots, \hat{y}_{t+k} = \hat{x}_{t+k}$

Sentiment Analysis:

x_1 → x_2 → x_3 → x_4 → x_5

wishing you a speedy recovery

$\hat{y}$ = +ve / -ve.

Dataset:

x_1 . . . . x_5	+ve / -ve
x_1 . . - . x_{100}	+ve / -ve

ANN = MLP (Generic Word)

↓ Depth

Deep Neural Network (several variants based on the problem they solve)

One step prediction

Sequence Prediction

FeedForward

Feedback

No Feedback

Transformers

RNN

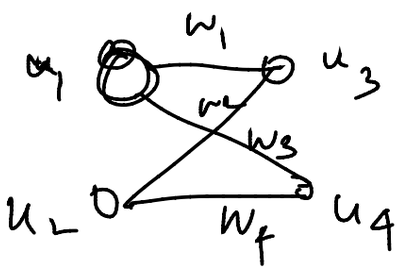
LSTM

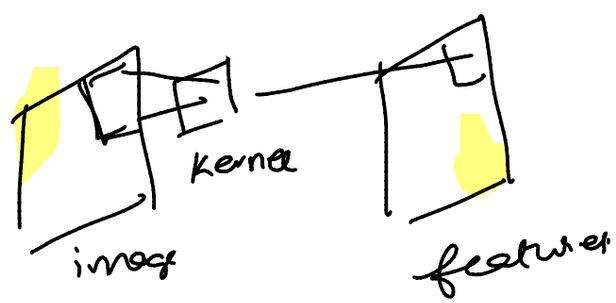
GRU

Dense
Fully connected

→ with weight sharing + Sparse weights → Convolutional (for images)

uses convolutions to extract local features from images.





One step prediction: 1 input, 1 output

Sequence Prediction

- Both input and output sequences are of different length
  - many ip, many op: machine translation, Speech recognition
  - many-ip; one op: Sentiment analysis
  - one-ip; many-op: image captioning
- Sequence prediction is always not a "time series"

  input domain ≠ output domain
- Inputs and outputs are temporally related

Part of Speech tagging

(domain = tag)

Output : noun verb article adjective verb noun

y_1 y_2 y_3 y_4 y_5 $\in \mathbb{R}^{d_{out}}$

v_1 v_2 v_3 v_4 v_5

s_1 s_2 s_3 s_4 s_5 $\in \mathbb{R}^{d}$

internal representation or intermediate representation

V_1 V_2 V_3 V_4 V_5

x_1 x_2 x_3 x_4 x_5 $\in \mathbb{R}^{d_{in}}$

(embedding)

Input : man is a social animal

(domain = vocabulary)

$V_i \in \mathbb{R}^{d \times d_{in}}$

$V_i \in \mathbb{R}^{d_{out} \times d}$

$$s_i = U_i x_i \quad \text{(input to internal)}$$

$$y_i = V_i s_i \quad \text{(internal to output)}$$

- U_i, V_i are different for each i $\Rightarrow$ no temporal connection yet

- $$s_i = U x_i$$

  $$y_i = V s_i$$

$\Rightarrow$ one step prediction applied at different times

A step back: How are sequences generated?

y_1 y_2 y_3 y_4 y_5 y_6

0, 0, 1, 0, 0, 1, 0, 0, 1, - - -

```
S = [ 0, 0, 1 ]

i = 0
while (true)
 {
    print S[ i % 3 ]

    i = i + 1

}
```

Matrix Equivalent

$$s_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$s_{t+1} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} s_t$$

$$s_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad s_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad s_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$y_t = [\,0 \;\; 0 \;\; 1\,]\, s_t$$

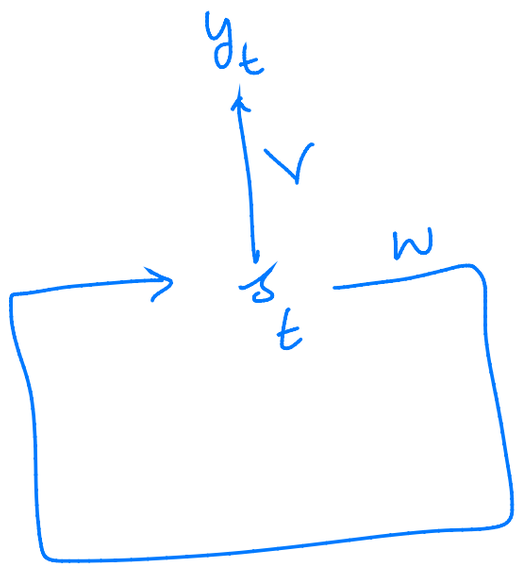
$$y_1 = [\,0 \;\; 0 \;\; 1\,] \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = 0$$

$$y_2 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = 0, \quad y_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 1$$

$$0, 0, 1, 0, 0, 1 \; - \; - \; .$$

$$W: \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

$$\left.\begin{aligned} s_{t+1} &= W s_t \\ y_t &= V s_t \end{aligned}\right\}$$ Linear Reaurrence without input



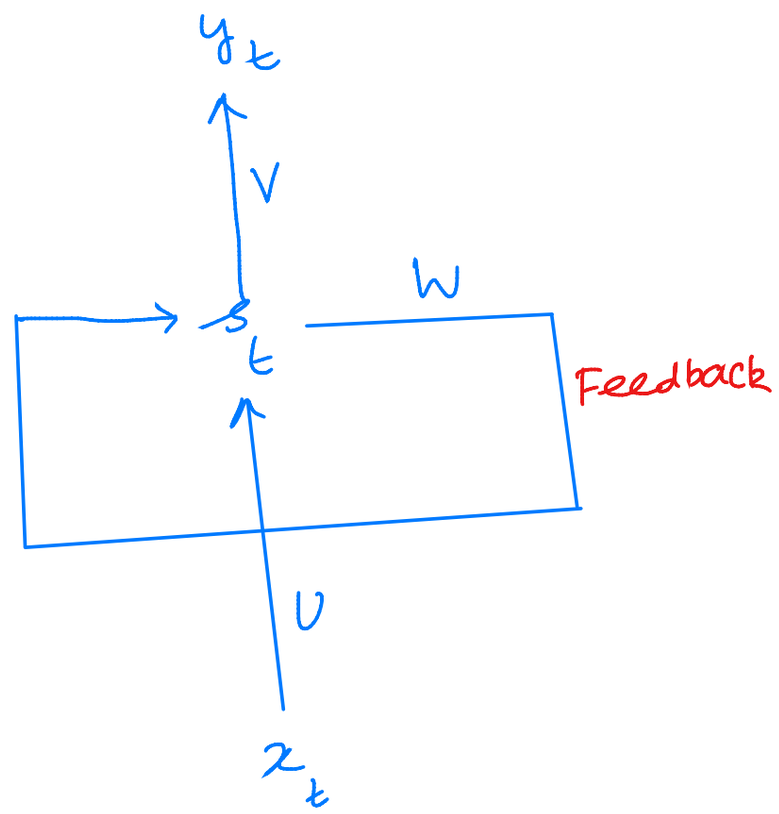
Feedback

input

$$s_{t+1} = W s_t + U x_t$$

$$\text{output} \rightarrow y_t = V s_t$$

} Linear Recurrence with input



unfurling the recurrence

$$s_{t+1} = W s_t + U x_t$$

$$= W (W s_{t-1} + U x_{t-1}) + U x_t$$

$$= W^2 s_{t-1} + W U x_{t-1} + U x_t$$

$$= W^2 (W s_{t-2} + U x_{t-2}) + W U x_{t-1} + U x_t$$

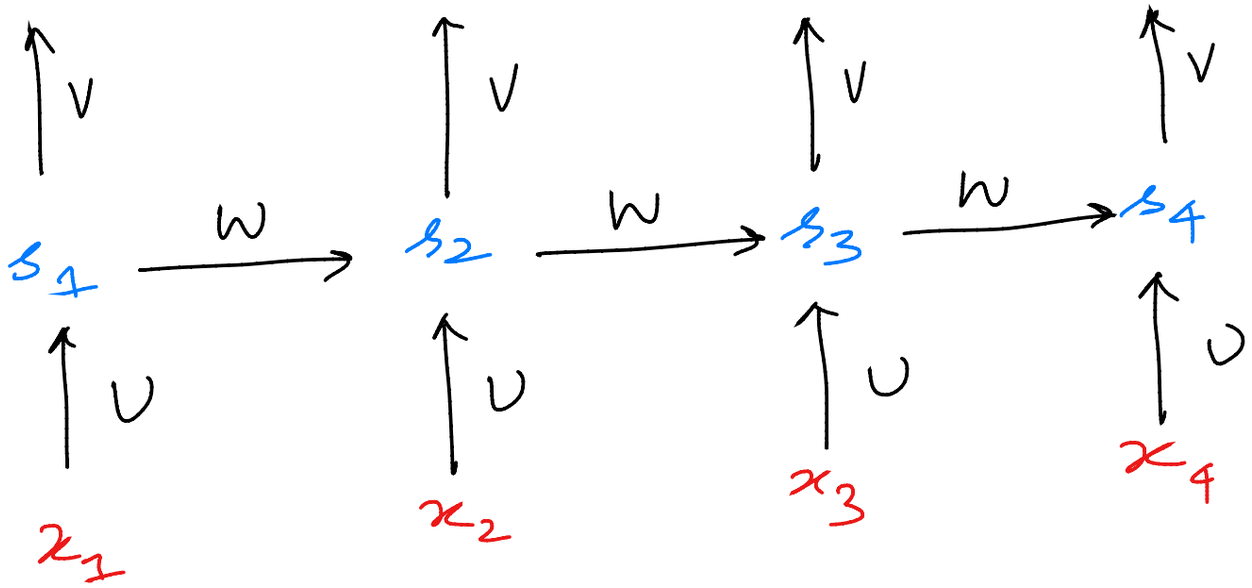
$$\vdots$$

$$= W^t s_1 + W^{t-1} U x_1 + W^{t-2} U x_2 + \cdots + U x_t$$

summary of x_2

$W^t s_1$: how much of initial memory 's_1' is remaining

$W^{t-1} U x_1$: summary input at $t=1$, ie x_1


V
V
V
V
W
W
W
s_1
s_2
s_3
s_4
U
U
U
U
x_1
x_2
x_3
x_4

Recurrent Neural Network

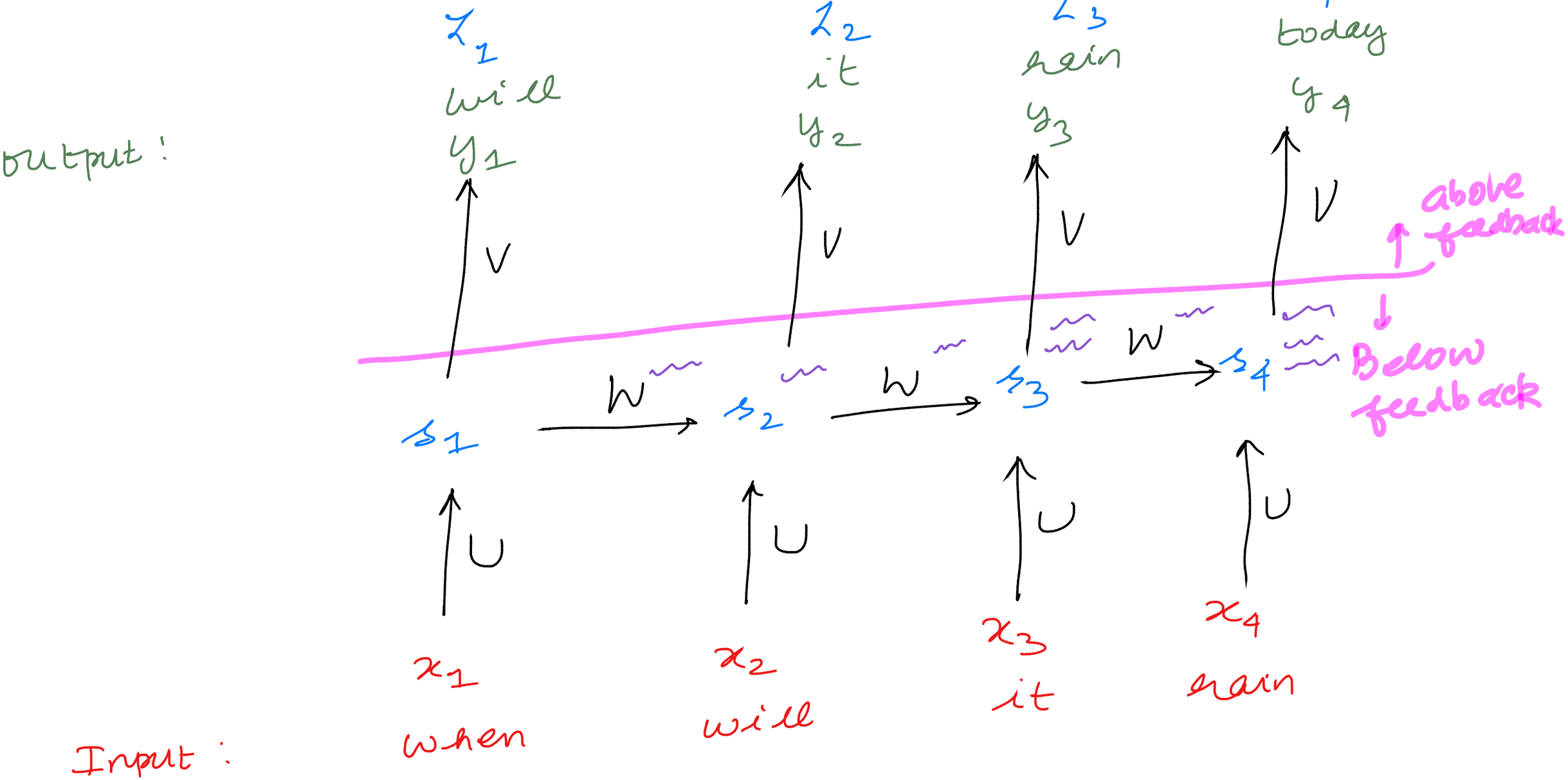
$$s_t = \sigma_s(W s_{t-1} + U x_t + bias)$$

$$y_t = \sigma_y(V s_t + bias)$$

- $W \in \mathbb{R}^{d \times d}$
- $U \in \mathbb{R}^{d \times d_{in}}$
- $V \in \mathbb{R}^{d_{out} \times d}$

RNN Training

Backpropagation Through Time



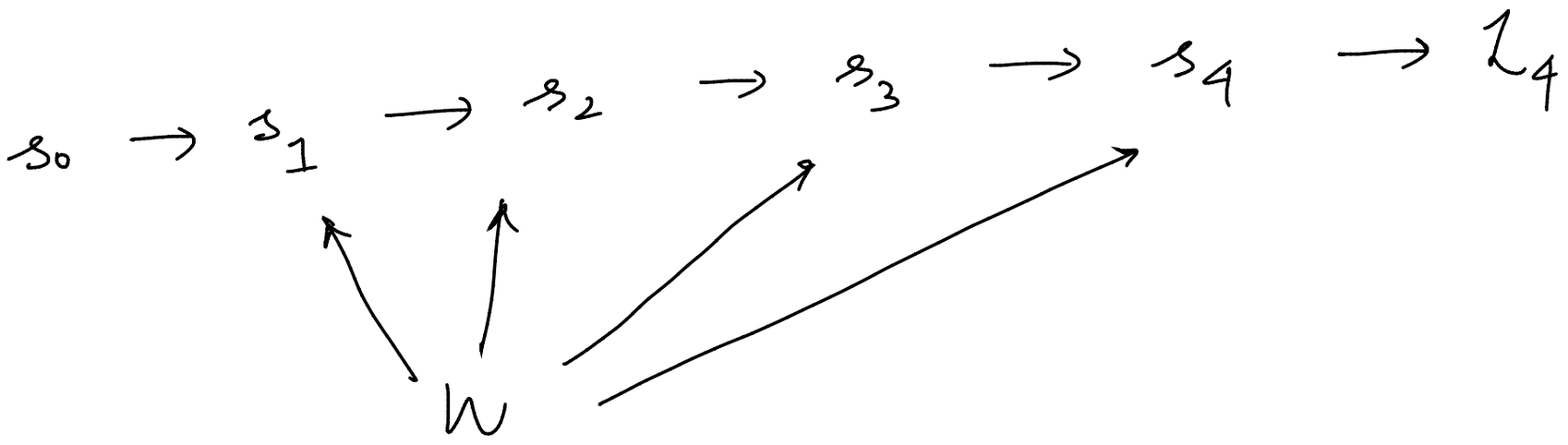
y_1 can be the probability over vocabulary

L_i : can be cross entropy loss

$$L = L_1 + L_2 + L_3 + L_4$$

$\frac{\partial L}{\partial V}$ calculation is standard back propagation because V is above the feedback

$\frac{\partial L}{\partial W}$ involves time



Moral: change in W causes a ripple effect and capturing this is called Backpropagation Through Time

$$\frac{\partial L_4}{\partial w} = \frac{\partial L_4}{\partial s_4} \cdot \frac{\partial s_4}{\partial w}$$

to compute $\frac{\partial s_4}{\partial w}$

- direct part : $\frac{\partial^+ s_4}{\partial w}$ = keeps s_3 a constant
- indirect part : effect of changing w on s_3

$$\frac{\partial s_4}{\partial w} = \frac{\partial^+ s_4}{\partial w} + \frac{\partial s_4}{\partial s_3} \frac{\partial s_3}{\partial w}$$

direct

indirect

$$= \frac{\partial^+ s_4}{\partial W} + \frac{\partial s_4}{\partial s_3}\left(\frac{\partial^+ s_3}{\partial W} + \frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial W}\right)$$

$$\frac{\partial s_4}{\partial W} = \frac{\partial^+ s_4}{\partial W} + \frac{\partial s_4}{\partial s_3}\frac{\partial^+ s_3}{\partial W} + \frac{\partial s_4}{\partial s_3}\frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial W}^{+} + \frac{\partial s_4}{\partial s_3}\cdot\frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial s_1}\frac{\partial^+ s_1}{\partial W}$$

$$= \frac{\partial^+ s_4}{\partial W} + \frac{\partial s_4}{\partial s_3}\frac{\partial^+ s_3}{\partial W} + \frac{\partial s_4}{\partial s_2}\frac{\partial^+ s_2}{\partial W} + \frac{\partial s_4}{\partial s_1}\frac{\partial^+ s_1}{\partial W}$$

$$\frac{\partial s_4}{\partial W} = \sum_{k=1}^{4}\frac{\partial s_4}{\partial s_k}\frac{\partial^+ s_k}{\partial W}$$

Backpropagation Through Time

$$\frac{\partial L_4}{\partial W} = \frac{\partial L}{\partial s_4}\left(\sum_{k=1}^{4}\frac{\partial s_4}{\partial s_k}\frac{\partial^+ s_k}{\partial W}\right)$$

Taken care under the hood by Tensorflow / PyTorch / Caffine

Issues with RNNs

$$s_t = \sigma_s(Ws_{t-1} + Ux_t + b_s)$$

$$y_t = \sigma_y(Vs_t + b_y)$$

Simplify by removing σ_s, σ_y, b_s, b_y

$$s_t = Ws_{t-1} + Ux_t$$

$$y_t = Vs_t$$

Further simplify by making dimension = 1

$$s_t = ws_{t-1} + ux_t$$

$$y_t = vs_t$$

Further simplify, let $u = 0$

$$s_t = w\, s_{t-1}$$

$$y_t = v\, s_t$$

$$s_1 = w s_0, \quad s_2 = w s_1 = w^2 s_0, \quad s_3 = w^3 s_0, \ \ldots, \quad s_t = w^t s_0$$

$|w| > 1$: for $s_0 \neq 0$, $s_t \longrightarrow \infty$ (exploding memory)

$|w| < 1$: $s_t \longrightarrow 0$ (forgetting / fading memory)

$w = 1$: s_t remains the same

$w = -1$: s_t flips sign

Moral: Because of feedback memory either explodes or fades

Long - Short Term Memory Networks

explicit control of read, writing and forgetting

We will use the white board analogy by Prof. Mitesh

Idea: Think of s_t to be whiteboard

Toy example: computation of $ac(bd+a) + ad$

$a = 1,\ b = 3,\ c = 5,\ d = 11$

Constraint: we can only have a summary of 3 statements at a time

- $ac = 5$
- $bd = 33$
- $bd + a = 34$

selective writings

$ac,\ bd,\ ad,\ a,\ bd+a$

selective reading of input

$ad \times (bd+a)$

- $ac = 5$
- $ac(bd+a) = 170$
- $bd+a = 34$

selectively forgetting bd

- $ac = 5$
- $ac(bd+a) = 170$
- $ad = 11$

select $bd+a = 34$ for forgetting

- $ad + ac(bd+a) = 181$
- $ac(bd+a) = 170$
- $ad = 11$

push $ac = 5$ out

Moral: Selective Reading, Write, Forget

LSTM

Selective Write

$$\begin{bmatrix} 1.2 \\ -3 \\ \vdots \\ 5 \end{bmatrix} \odot \begin{bmatrix} 0.2 \\ 0.5 \\ \vdots \\ 0.3 \end{bmatrix} = \begin{bmatrix} 1.2 \times 0.2 \\ -3 \times 0.5 \\ \vdots \\ 5 \times 0.3 \end{bmatrix}$$

s_{t-1} — element wise product — O_{t-1} — h_{t-1}

$$h_{t-1} = s_{t-1} \odot O_{t-1}$$

Output Gate

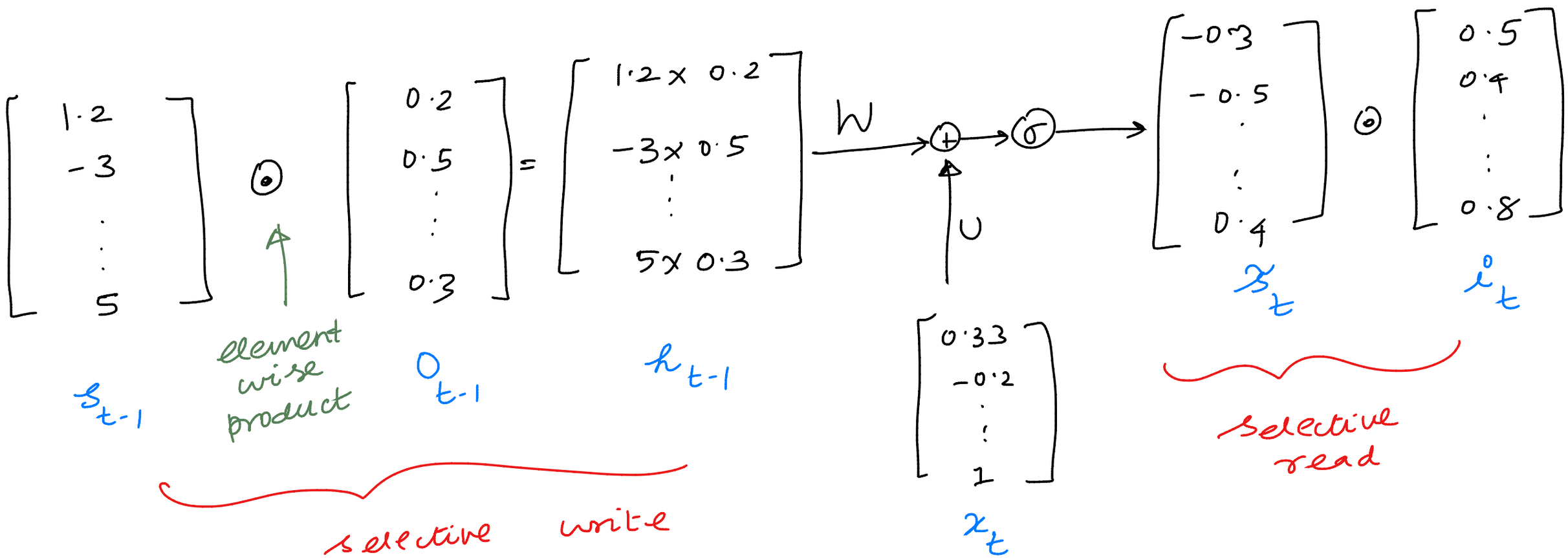
$$O_{t-1} = \sigma\left(W_o h_{t-2} + U_o x_{t-1} + b_o\right)$$

$$\tilde{s}_t = \sigma_s\left(W h_{t-1} + U x_t + b\right)$$

Input Gate

$$i_t = \sigma\left(W_i h_{t-1} + U_i x_t + b_i\right)$$

$$s_t = \tilde{s}_t \odot i_t$$



Selective Forget

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$

$$s_t = f_t \odot s_t + i_t \odot \tilde{s}_t$$

↑ summary — forget — selective reading

LSTM

Gates

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$

States

$$\tilde{s}_t = \sigma(W h_{t-1} + U x_t + b)$$

$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$$

$$h_t = o_t \odot \sigma(s_t)$$