MACHINE INTELLIGENCE

UNIT-5

RNNs

feedback/corrections: vibha@pesu.pes.edu

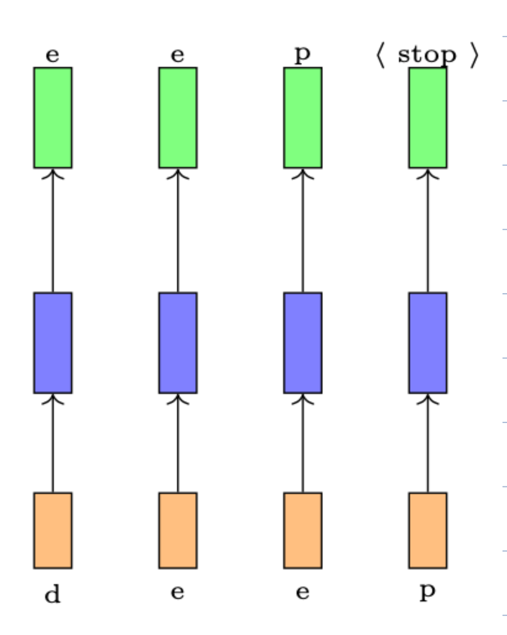
VIBHA MASTI

Why RNNs?

- Feedforward NNs : fixed input size, each input independent of prev/future inputs

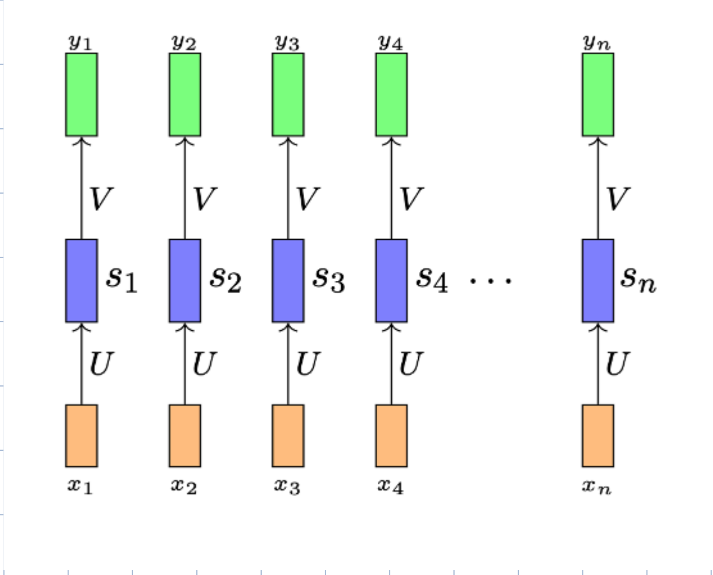
- Image classification: each image independent of prev

- Auto-completion: successive inputs not independent
  - predicting next letter depends on prev & cur inputs
  - length of input, no. of predictions not fixed
  - each orange-blue-green network performs same task



- Sequence learning problems

- Functions at each layer



$$s_i = \sigma(Ux_i + b)$$

or

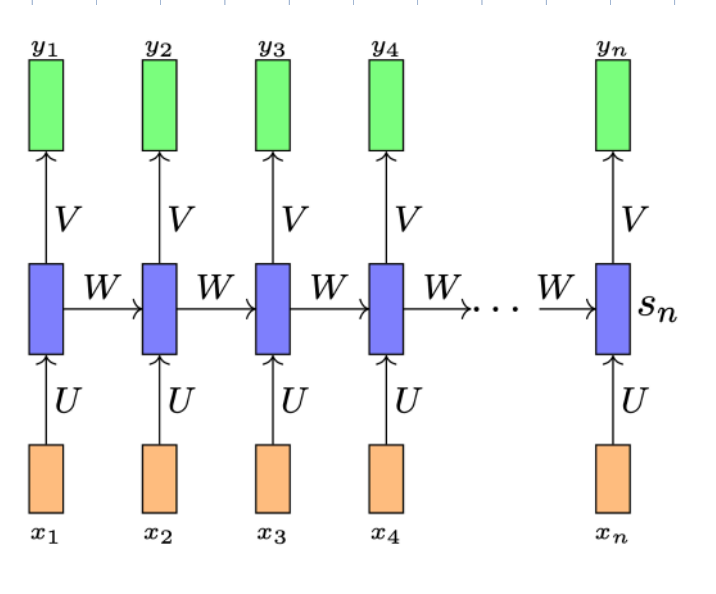
$$s_i = \tanh(Ux_i + b)$$

$$y_i = O(Vs_i + c)$$

or

$$y_i = \text{softmax}(Vs_i + c)$$

RNNs



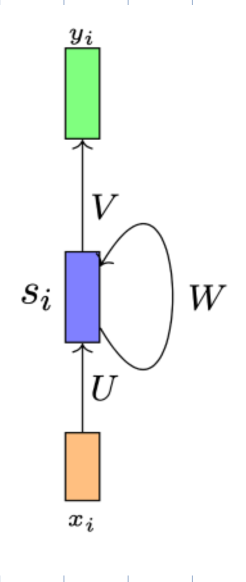
parameters shared across timestamps

$$s_i = \sigma(Ux_i + Ws_{i-1} + b)$$

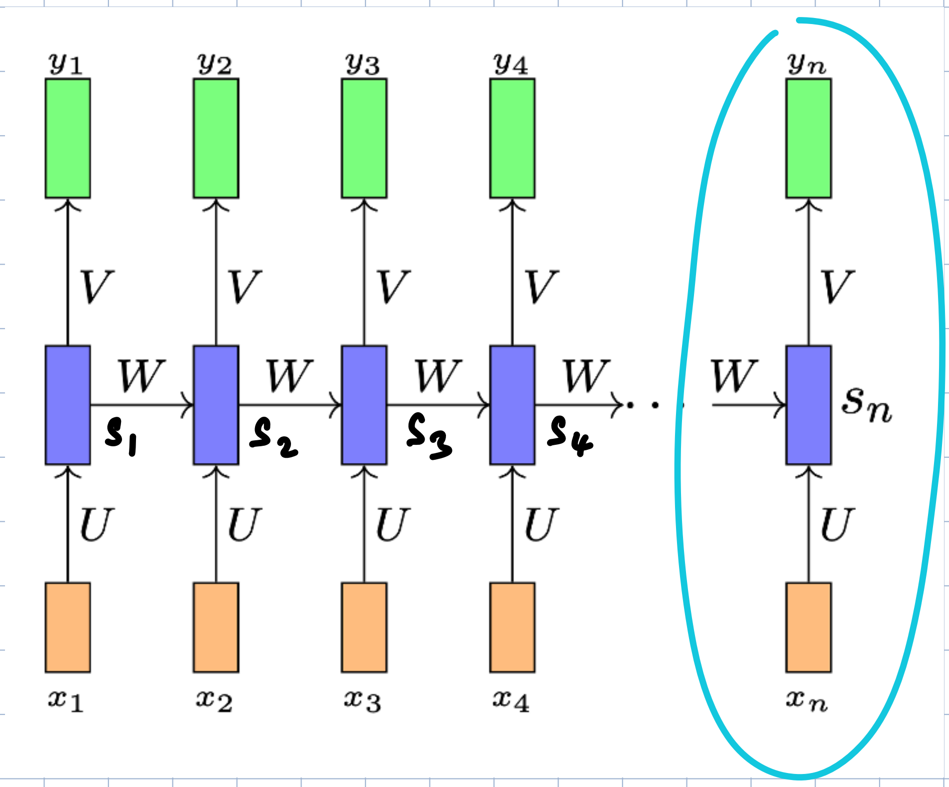
$$y_i = O(Vs_i + c)$$

or

$$y_i = f(x_i, s_{i-1}, U, V, W, b, c)$$



Dimensions



→ each is a fully connected ANN

$x_i \in R^n$ (n-d input)

$s_i \in R^d$ (d-d state)

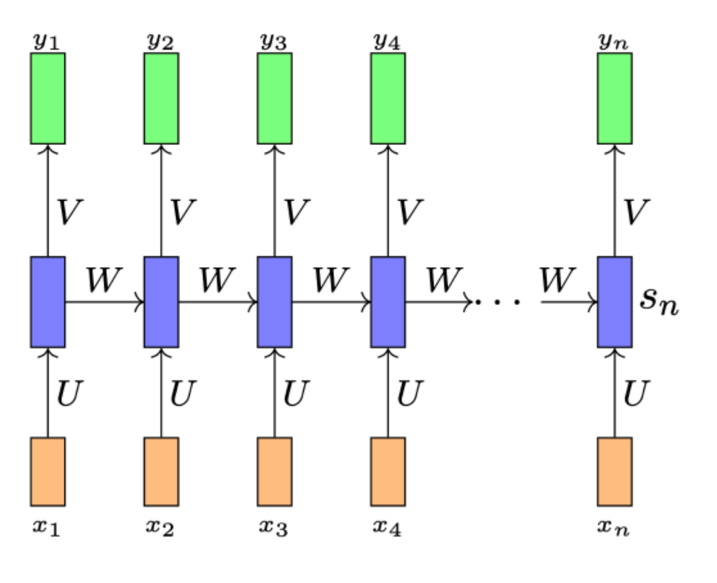
$y_i \in R^k$ (k classes)

$U \in R^{n \times d}$

$W \in R^{d \times d}$

$V \in R^{d \times k}$

BACKPROPAGATION (for GD)



$$\mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta)$$

$$\mathcal{L}_t(\theta) = -\log(y_{tc})$$

y_{tc} : predicted prob of true character at timestep t

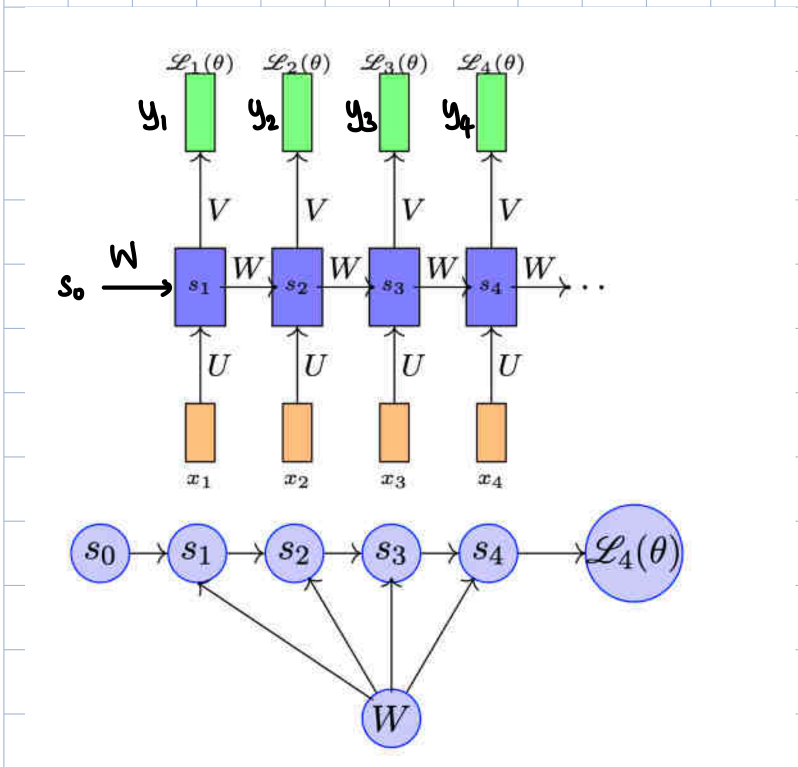
(1) Derivative of loss wrt outer layer weights V (matrix)

$$\frac{\partial \mathcal{L}(\theta)}{\partial V} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}_t(\theta)}{\partial V}$$

(2) Derivative of loss wrt recurrent weights W (matrix)

$$\frac{\partial \mathcal{L}(\theta)}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}_t(\theta)}{\partial W}$$

Consider $\mathcal{L}_4(\theta)$



$\mathcal{L}_4(\theta)$ depends on s_4

s_4 depends on s_3 & W

s_3 depends on s_2 & W

s_2 depends on s_1 & W

s_1 depends on s_0

Timestamp 1

$$\frac{\partial \mathcal{L}_1(\theta)}{\partial W} = \frac{\partial \mathcal{L}_1(\theta)}{\partial y_1} \frac{\partial y_1}{\partial s_1} \frac{\partial s_1}{\partial W}$$

Timestamp 2

$$\frac{\partial \mathcal{L}_2(\theta)}{\partial W} = \frac{\partial \mathcal{L}_2(\theta)}{\partial y_2} \frac{\partial y_2}{\partial s_2} \underbrace{\frac{\partial s_2}{\partial W}}_{\text{treat } s_2 \text{ as independent}} + \frac{\partial \mathcal{L}_2(\theta)}{\partial y_2} \frac{\partial y_2}{\partial s_2} \underbrace{\frac{\partial s_2}{\partial s_1} \frac{\partial s_1}{\partial W}}_{s_2 \text{ depends on } s_1}$$

Timestamp t

$$\frac{\partial \mathcal{L}_t(\theta)}{\partial W} = \sum_{k=1}^{t} \frac{\partial \mathcal{L}_t(\theta)}{\partial y_t} \frac{\partial y_t}{\partial s_t} \left(\prod_{i=k+1}^{t} \frac{\partial s_i}{\partial s_{i-1}} \right) \frac{\partial s_k}{\partial W}$$

$$= \frac{\partial \mathcal{L}_t(\theta)}{\partial y_t} \sum_{k=1}^{t} \frac{\partial y_t}{\partial s_t} \left(\prod_{i=k+1}^{t} \frac{\partial s_i}{\partial s_{i-1}} \right) \frac{\partial s_k}{\partial W}$$

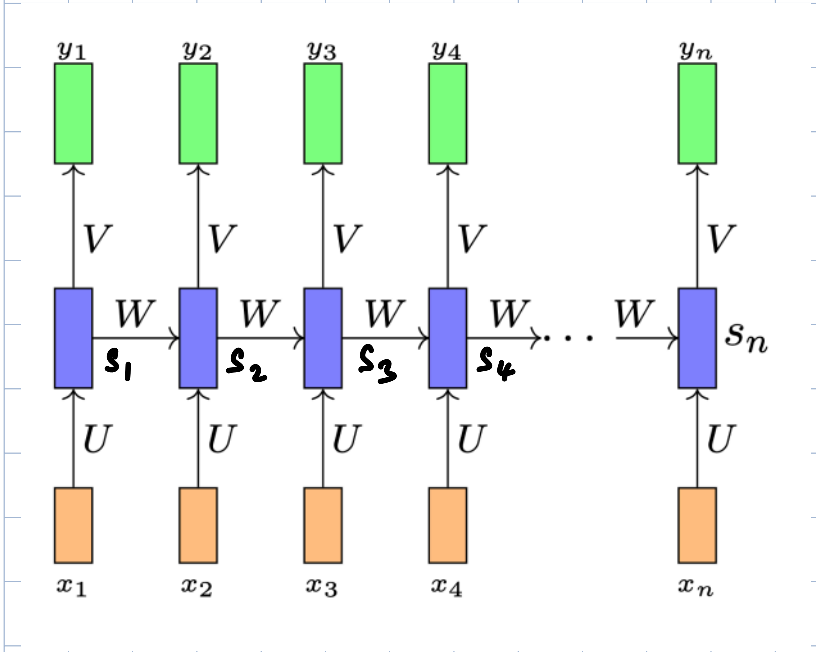
Exploding / Vanishing Gradient

$$\frac{\partial s_i}{\partial s_{i-1}}$$

$$a_i = W s_{i-1} + b + U x_i$$

$$s_i = \sigma(a_i)$$

$$\frac{\partial s_i}{\partial s_{i-1}} = \underbrace{\frac{\partial s_i}{\partial a_i}}_{(a)} \underbrace{\frac{\partial a_i}{\partial s_{i-1}}}_{(b)}$$



$$a_i = [a_{i1}, a_{i2}, \dots, a_{id}]$$

$$s_i = [\sigma(a_{i1}), \sigma(a_{i2}), \dots, \sigma(a_{id})]$$

ⓐ $\dfrac{\partial s_i}{\partial a_i}$ is a d×d matrix

$$\frac{\partial s_i}{\partial a_i} = \begin{bmatrix} \frac{\partial s_{i1}}{\partial a_{i1}} & \frac{\partial s_{i2}}{\partial a_{i1}} & \cdots & \frac{\partial s_{id}}{\partial a_{i1}} \\ \frac{\partial s_{i1}}{\partial a_{i2}} & \frac{\partial s_{i2}}{\partial a_{i2}} & & \vdots \\ \vdots & & \ddots & \\ \frac{\partial s_{i1}}{\partial a_{id}} & \cdots & & \frac{\partial s_{id}}{\partial a_{id}} \end{bmatrix}$$

$$\frac{\partial s_i}{\partial a_i} = \begin{bmatrix} \sigma'(a_{i1}) & 0 & \cdots & 0 \\ 0 & \sigma'(a_{i2}) & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma'(a_{id}) \end{bmatrix}$$

$$= \mathrm{diag}(\sigma'(a_i))$$

ⓑ $\frac{\partial a_i}{\partial s_{i-1}}$ $a_i = W s_{i-1} + b$

$$\frac{\partial a_i}{\partial s_{i-1}} = W$$

Magnitude of $\frac{\partial s_i}{\partial s_{i-1}}$

$\sigma'(a_i) \leq \frac{1}{4} = \gamma$ (sigmoid)

$\sigma'(a_i) \leq 1 = \gamma$ (tanh)

$$\begin{aligned}
\left\| \frac{\partial s_i}{\partial s_{i-1}} \right\| &= \left\| \mathrm{diag}(\sigma'(a_i))\, W \right\| \\
&\leq \left\| \mathrm{diag}(\sigma'(a_i)) \right\| \, \|W\| \\
&\leq \gamma \, \|W\| \\
&\leq \gamma \, \lambda
\end{aligned}$$

$$\left\| \frac{\partial s_t}{\partial s_{t-1}} \frac{\partial s_{t-1}}{\partial s_{t-2}} \cdots \frac{\partial s_{k+1}}{\partial s_k} \right\| = \left\| \prod_{j=k+1}^{t} \frac{\partial s_j}{\partial s_{j-1}} \right\|$$

$$\le \left(\prod_{j=k+1}^{t} \gamma \lambda \right)$$

$$\le (\gamma\lambda)^{t-k}$$

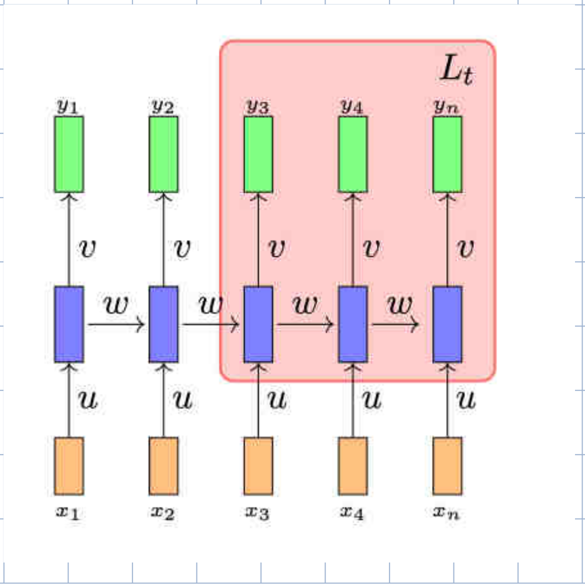
if $\gamma\lambda < 1$: vanishing gradient

if $\gamma\lambda > 1$: exploding gradient

How to avoid?

1. Truncated backpropagation

- only $t-k$ timestep to t timestep



2. Gradient clipping

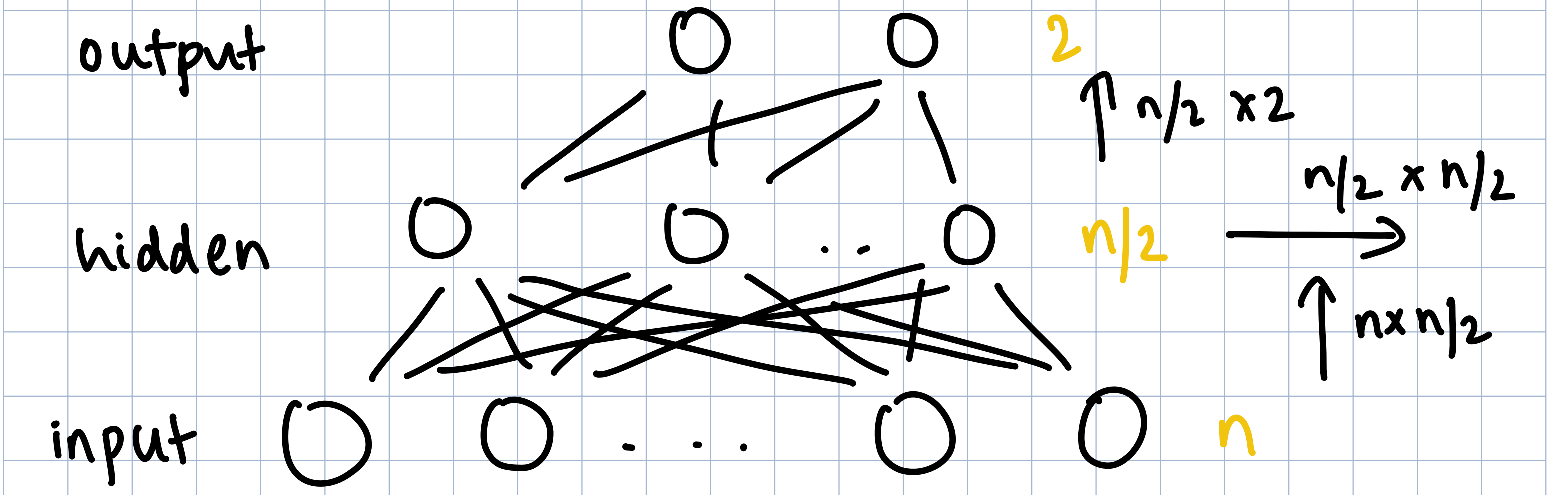
- Normalise gradients according to a vector norm (eg: L2)

- Let $\hat{g} = \frac{\partial \mathcal{L}(\theta)}{\partial W}$

$$\hat{g} = \frac{\text{threshold}}{\|\hat{g}\|} \cdot \hat{g}$$

3. LSTM

Q: An RNN takes input of words each as a vector of length n, one hidden layer with n/2 neurons and one output layer with 2 neurons. If total no. of weights =161, find n.



$$\text{total} = n \times \frac{n}{2} + \frac{n}{2} \times \frac{n}{2} + \frac{n}{2} \times 2 = 161$$

$$\frac{3n^2}{4} + n = 161$$

$$n = 14$$

(3) Derivative of loss wrt input weights U (matrix)

$$\frac{\partial \mathcal{L}(\theta)}{\partial U} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}_t(\theta)}{\partial U}$$

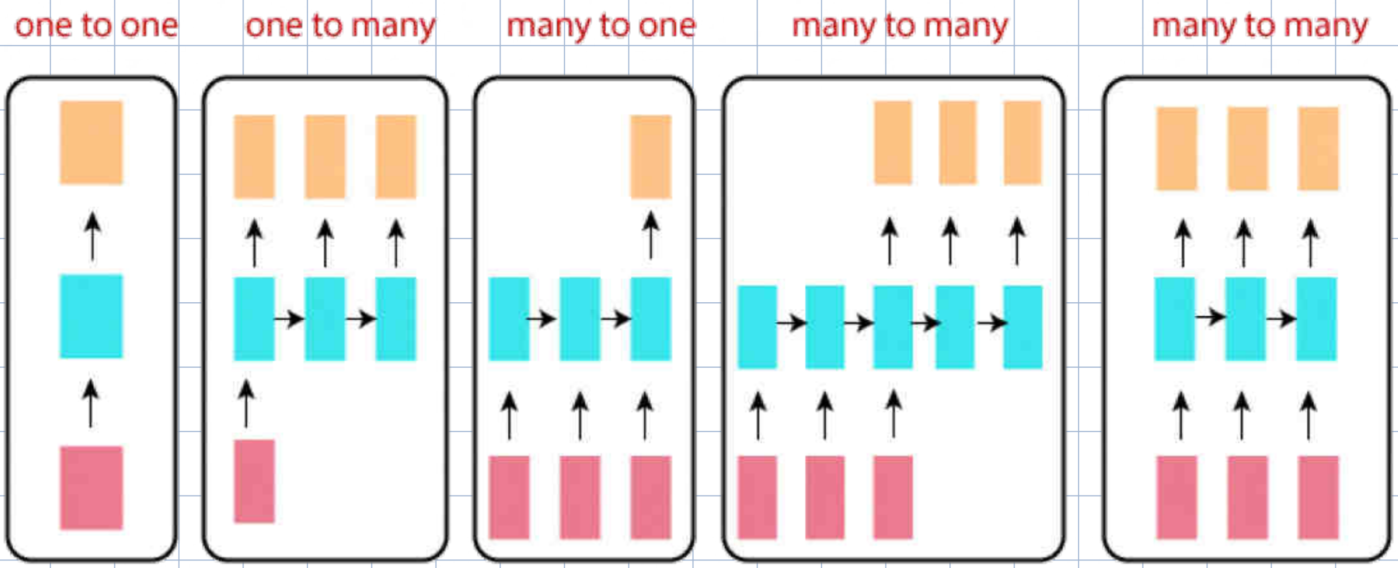
RNN Architectures

(1) One-to-one

- Single input to single output
- Output from t fed as input to t+1
- Eg: predictive text

(2) One-to-Many

- Single input to multiple hidden states and multiple output values
- Share hidden states across timesteps
- Eg: image captioning, music generation



© vibhas notes 2021

(3) Many-to-One

- Sequence of inputs mapped to single output
- Eg: sentiment analysis

(4) Many-to-Many

- Input of arbitrary length, output of arbitrary length
- Eg: language translation