U-4 (DL)
(Recurrent Neural Networks)

* RNN-

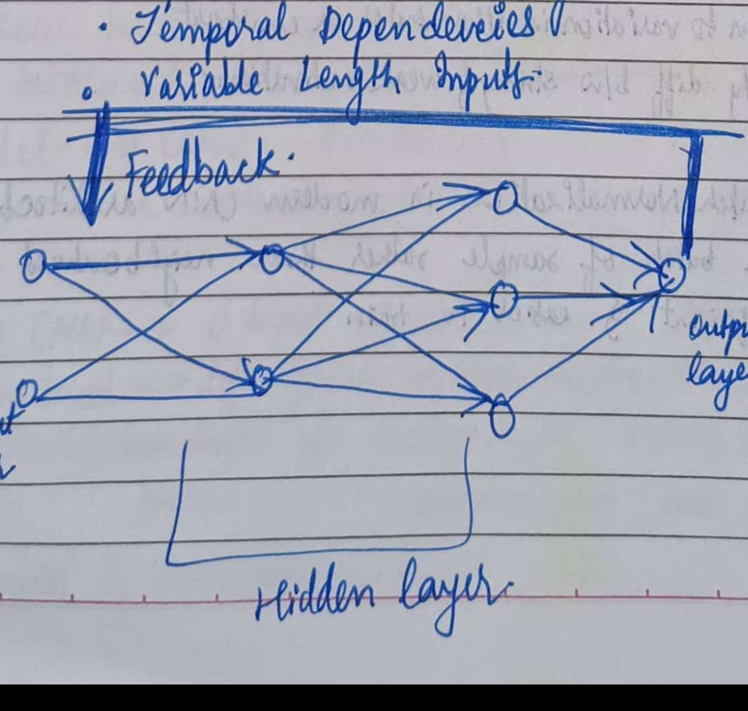
- Recurrent Neural Networks.
- Type of artifical neural network architecture
- Handles sequential data
- Internal memory that allows them remember past inputs & use context current input.
  - remember each info through time

Advantages → • Memory (past inputs & use info to process current one)
- useful in time series. • Variable Length Inputs
- extend effective pixel neighbourhood. • efficient processing.
- process sequence of data as an output & receives a sequence of information as input.
- even used in convolutional layers to extend neighbourhood.

Disadvantages → • Training challenges
- Gradient Vanishing & exploding problems.
- Can't process long sequence.
- slower computation.

- Language transl^n
- text to speech.

Need → • Sequential Data Processing
- Temporal Dependencies
- Variable Length Inputs.



Page No.
Date

Working RNN -

- Input - an element from a sequence as input.
- Hidden Layer Magic - Holds info from past elements.
- Information Mix - current input & hidden layer's memory are combined.
- Activation & Output - processed by activation funⁿ to generate an output.
- Memory Update - Its memory based on processed information.

Bi-directional Recurrent Neural Network -

- to process sequential data
- Type of RNN
- Capture contextual dependencies in input data by processing.
- one process data → forward direction (Forward RNN)
- 2^{nd} process data → reverse direction. (Backward RNN)

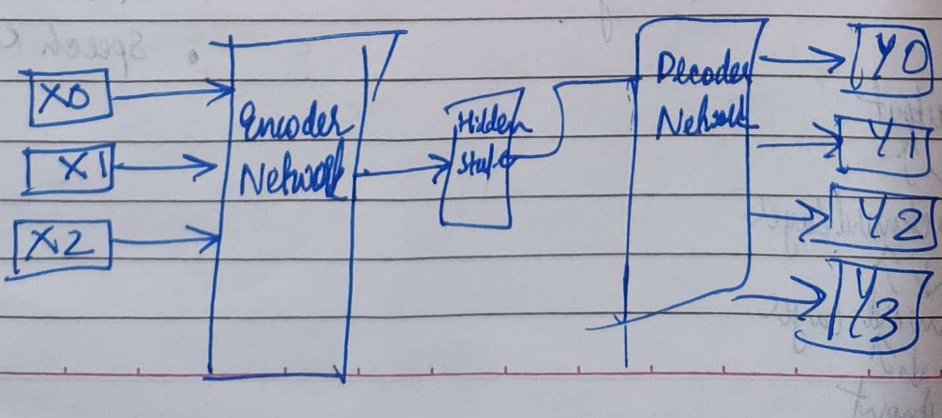
- Adv →
  - Improved Accuracy
  - efficient handling of variable length sequences
  - Better Long-Term Dependency Capture
  - context from both past & future.
- Disadv -
  - Increased computation Complexity
  - more Parameters to Train
  - Limited to Offline Processing

- working →
  - Inputting a sequence
  - Dual Processing
  - Computing hiddenstate
  - Determining output
  - Training
- Applⁿ -
  - Sentiment Analysis
  - Named Entity Recognition
  - Part of Speech Tagging
  - Machine Translation
  - Speech Recognition.

Output
↓↑
Backward Layer
↓↑
Forward Layer
↓↑
Input

* Encoder - Decoder Architecture -

	Encoder	Decoder =
	• compresses input sequence into a fixed length representation	• fixed length repsn into a input sequence
Input →	• variable length sequence (eg - sentence)	• fixed length representan (contex vector)
output →	• fixed length represntation. (Context vector)	• variable length sequence (translated sentence)
Processing ↳	• step by step using RNNs (LSTMs GRUs) to capture long term dependencies.	• step by step using RNNs. considering previously generated output & contentvec
complexity →	simpler operation	• more complex opern due to considering history.
Location →	Typically at trasmitting end	• Typically at receiving end.
eg →	• Text summarization. encodes a long document, generate a concise summary	• Machine translation. decoded source languages generates target language text



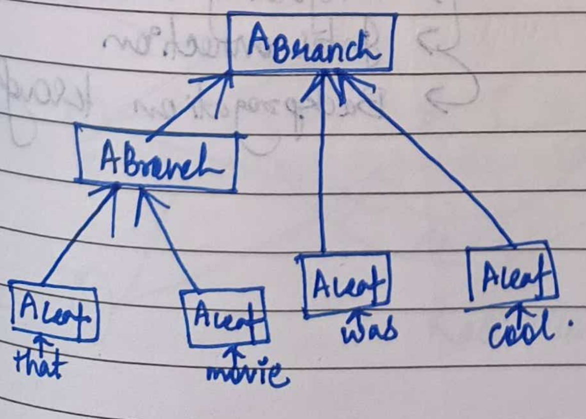
* Recursive Neural Network. (RvNNs).

- another generalization of recurrent network with diff compu^n graphs.
- structured as a deep tree.
- variable input sequence can be mapped to a fixed size output
- more suited for time-series data → linguistic structure aside.
- born to model by NLP syntax parse trees.
- don't take token sequentially, recursive models combine neighbor
- RVNNs aka TreeNets.

Adv →
- Handles complex structure
- Compositionality
- Expressive Power
- Reduced Network Depth
- Theoritical Underpinning
- Emerging Appl^n

Disadv →
- Computationaly Expensive
- Limited applicability
- Parallelization Challenges.

Steps →
- Start at root node of hiearchy
- Process elements at root node
- Recursively approach each child node
  - * Apply same processing logic to child
  - * combine processed child's output with parents output
- Move the hierachy & repeat steps until 2-3 until all nodes are processed
- Final combined output represent entire hierachial structure.



Page No.
Date

* Deep Recurrent Neural Network —

- Type of ANN architecture → handle sequential data where order of elements matter
- stacking multiple RNNs
- captures more complex relⁿ & dependencies within sequential data compared to regular RNNs.

Adv → • enchanced learning of Long Term dependencies.
• improved representation power
• Flexibility of various task

Disadv → • Increased Computational Cost
• Vanishing / Exploading Gradient Problem
• Potential for overfitting -

* Unfolding Computational Graph ⇒

- way to formulize structure of set of computations-
- consist of nodes that represent mathematical operⁿ or variables, edges → flow of data b/w these operations.

Adv → • Clarity
• Traning
• Analysis
• use same funⁿ with same parameters.

steps → Single Time Step
→ Unfolding across time
→ Interconnection
→ Backpropagation through time.

* Long Term Dependencies - (Challenges).

- Vanishing gradient → which direction to move to reduce cost. (difficult to known)
  - * RRN tries to learn long range dependecies → gradient are small.
  - * RRN not able to learn rel^n b/w input & output sequences.
  - * becoms small that they are efficiently zero.
- Exploding gradient → make learning unstable
  - → gradient → large make RNN to diverge.
  - → RNN can't learn rel^n.
- RNN struggle as they forget past information as they process sequen

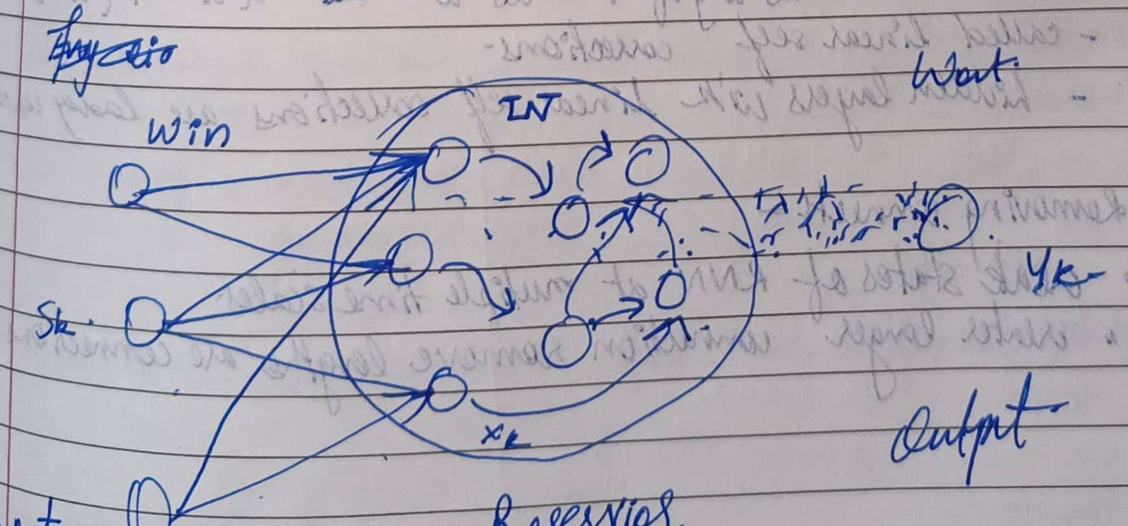
* Echo statenetworks (ESN) -

- type of RNN → easy to train & able to learn long term depende
- ability to handle sequencial data.
- series prediction, speech recognition & control systems.
- Hidden layer or Reservoir is sparsely connected.

Adv → • easy to train
• learn long range dependecies
• relatively fast to run.

Disadv → sensetive to choice of hyperpar
• difficult to interpret
• prone to overfitting.

~~Eng-ctio~~



* Leaky Units & strategies for mutiple time scales-

* Leaky units-

- type of RNN unit has linear self connection-
- fraction of output of unit is fed back into input of unit-
- ~~fraction of output is fed~~ back → leak rate-
- learn mutiple time scales in RNNs-
- leak rate → how much info from past is forgotten-
- high leak rate → more info from past is forgotten-
- low leak rate → less info " " "

Strategies -

- Adding Skip Time - connection through
  - Gradient explode wrt time steps (τ)
  - Introduces time delay (d) now gradient dimension as a func^n of τ/d rather than τ.
  - helps capture long term dependencies
- Leaky units of f spectrum of time scales -
  - Don't use integer skip of d instead use real valued α.
  - Consider $u^{(t)}$ as avg of $v^{(t)}$ as $u^{(t)} \leftarrow \alpha u^{(t)} + (1 0-\alpha) v^{(t)}$
  - called linear self connections-
  - hidden layers with linear self connections are leaky units.
- Removing connects -
  - create states of RNN at mutiple time scales.
  - create longer connection remove length are connections.

Page No.
Date

* Long Short Term Memory (LSTM) -

- Type of RNN.
- designed to address vanishing gradient problem.
- handles long term dependencies in sequential data.

* Adv → • captures long time dependencies.
  • addresses vanishing gradient problem
  • In various applications. → Machine Translation
    → speech recognition
    → Time series forecasting
    → Text generation.

* Disadv → • computational complexity
  • Prone to overfitting.

* Working → Forget Gate, Input Gate, Cell state, Output Gate.

* Gated Recurrent Unit (GRUs)

- Type of RNN
- process sequential data.
- handle long term dependency with data.

Adv → • simpler architecture
  • effective performance
  • eg → NLP, Speech Recog, Time Series Forecasting

Disadv → limited capability
  → Less control over inform^n flow

Working → Input, Reset Gate, Update Gate, Hidden State Update, Output.

* Optimization

Page No.
Date

* Optimization for Long Term Dependencies -

- Using larger batch size -
- Using lower learning rate -
- using gradient clipping
- Using dropout -
- Using 1, 2 regularization -
- Using gated RNN -

- weight initialization
- skip connection
- gated architecture
- regularization technique

* Explicit Memory

- use of seprate memory unit to store info from previous timestep
- contrast to implicit memory, info from prev steps is stored implicitly in weights of RNN
- Adv → • improve performance (Long range depencies)
  - store info from previous steps - (even if not relevant)
  - Reasoning & Problem solving.
- Disadv → • Increased complexity
  - Interpretability

ways of explicit memory implemented in RNNs -
- → Using seperate memory unit
- → Using a gated RNN.

* Performance Metrics

- Accuracy = $\frac{\text{No of correct pred}}{\text{Total no of pred.}}$
- Precision = $\frac{TP}{TP+FP}$
- Recall = $\frac{TP}{TP+FN}$
- Fscore = $2 \times \frac{\text{Precision} * \text{recall}}{\text{Prec} + \text{recall}}$

$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$.

* Default Baseline Model

- → Simple RNN
- → LSTM
- → GRU
- → BiRNN
- → encoder Decoder (S2S) Model

* Determing weather gather more data

- → Training Performance
- → Data Distribution
- → ~~Resource~~ Task Complexity
- → Expert Knowledge
- → Valid^n Perf.
- → Task Complexity
- → Resource Constraints.

* Selecting HyperParameters -

- Learning Rate
- Batch size.
- Activation fun^n
- No. of epochs
- No. of neurons