"NEURAL NETWORK – BASED PREDICTIVE MODELLING FOR STOCK MARKET FORECASTING ANALYSIS: A COMPARATIVE STUDY"

Research Intern project report submitted

by the student

Student Name: Ankit Kumar

(Student at IIT Patna)

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY GUWAHATI

(IIIG), Assam, India - 781015

Date: - 18th July 2024

Declaration

I hereby declare that this submission is my own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material that, to a substantial extent, has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Date: 18th July, 2024

Student Name, Roll No, Signature

Ankit Kumar, 2312res127, 

Summary of the Project

This project focuses on developing and evaluating neural network-based models for predictive analytics in critical domains: stock market forecasting data analysis. The primary aim is to leverage advanced machine learning algorithms to derive actionable insights from large-scale, real-world datasets.

The methodology involved several key steps. Data preprocessing was crucial, involving the cleaning and preparation of datasets to remove inconsistencies and missing values. Feature engineering was performed to enhance the predictive power of the models. Various neural network models, including ANN, RNN, CNN, LSTM, Transformers, and KNN, were developed and fine-tuned for stock market data. These models were trained using historical data and evaluated using appropriate metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and accuracy scores. A detailed comparative analysis was conducted to identify the most effective approach for each dataset, highlighting the strengths and weaknesses of each model in terms of predictive accuracy, computational efficiency, and robustness.

Table of Contents

Chapt 01: Artificial Neural network (ANN)

27th June'24 (Thursday)

Q. How to improve the performance of deep (ML) neural networks?

Ans: Take multiple hidden layers with fewer neurons because deep learning the techniques called Representation learning (I) Hidden layers, (ii) Neurons per layers, (iii) Learning rate, (iv) Optimizer, (v) Batch size, (vi) activation function and (vii) number of epochs

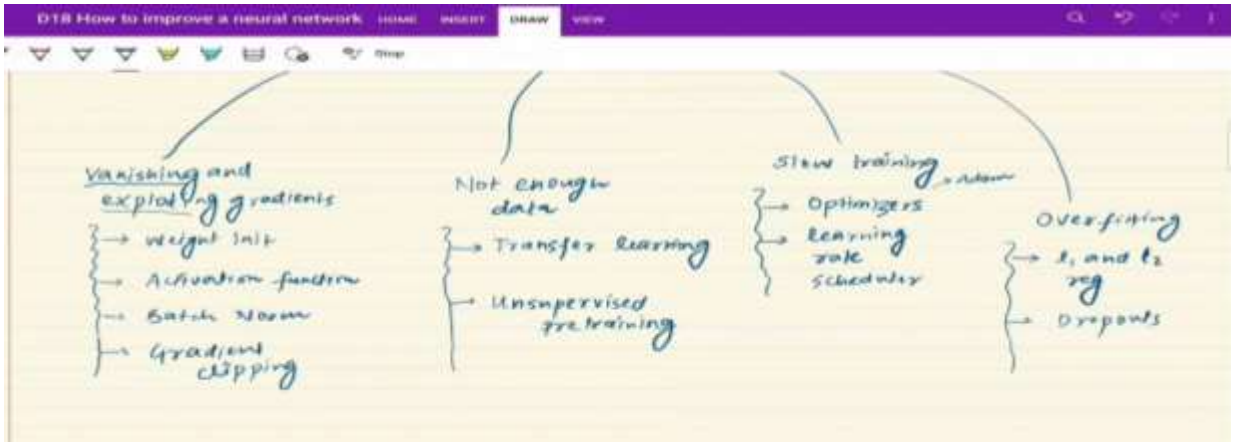
By solving problems: (I) Finishing or exploiting gradient, (ii) Not enough data, (iii) Slow training, and (iv) overfitting

Fine tuning Hyper Parameters

Transform learning (in CNN): you make a NN on a single problem and apply on the other problem, can handle complex problems.

No of neuron in each layer: Input layer = no of data as input, Hidden layer = generally in pyramid structure can capture primitive feature or same no. of neurons i.e. more than what is req.

Warming Up the learning rate: when we increased learning epoch if small epochs and decease if large. Epochs – 100, 500, 1000 (angry man take) Early stopping read in Keras: stable + callback concept Problems with solution in models



Tips

Early stopping in NN:

```
callback = EarlyStopping(
    monitor="val_loss",
    min_delta=0.00001,
    patience=20,
    verbose=1,
    mode="auto",
    baseline=None,
    restore_best_weights=False
)
```

```
history = model.fit(X_train, y_train, validation_data=(X_test, y_test),
epochs=3500, callbacks=callback)
```

https://keras.io/api/callbacks/early_stopping/

Scaling = Standard (-1 to +1) + Normalization (0 to 1) use X-MAX

Dropout problems:

Explain dropout is similar to Random Forest? Because Ensemble learning. Use dropout at training not in testing time.

Paper link on dropout: https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

28th June'24 (Friday) Regulation

Regulation is a way to reduce overfitting

Sparce modal: in which most of weight become zero.

Activation function: that calculates the output of the node based on its individual inputs and their weights.

https://en.wikipedia.org/wiki/Activation_function

Check right activation function – (I) Non-linear, (ii) Differentiable, (iii) Computationally expensive, (iv) Zero centred, and (v) non-saturating

Dying RELU problem: dead neuron – not learn on the given data.

Weight Initialize value https://www.deeplearning.ai/ai-notes/initialization/index.html

Enjoy the initialization animation: https://www.deeplearning.ai/ai-notes/initialization/index.html

Don't initialize the weight with: (i) zero initialization, (ii) non-zero constant, (iii) Random initialize with small weights, (iv) Random initialize with large weights.

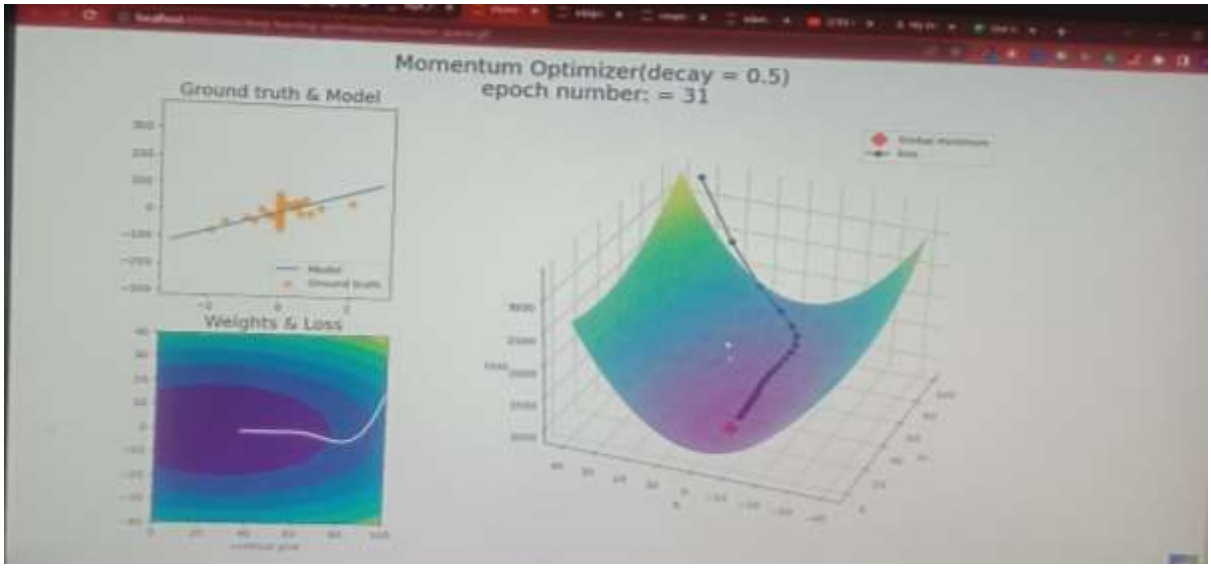
Batch **Norm**? is an algorithmic method which makes the training of DNN faster and more stable. It consists of normalizing activation vectors from hidden layers using the mean and variance of the current batch. This normalization step is applied right before (or right after) the nonlinear function. Mean = 0, SD = 1

Internal covariant shift: the change in the distribution of network activations due to the change in network parameters during training.

Optimizers: Gradient descent (I) Momentum, (ii) Adagrad, (iii) NAG, (iv) RMS prop. (iv) Adam == Exponentially weighted moving average (EWMA)

SGD with momentum, NAG (Nesterov Accelerated Gradient): when sparse data is given

AdaGrad (Adaptive Gradient) when complex NN dealing, RMS prop (propagations), ADAM (adaptive moment estimation)



IIITG

Hyperparameters Tuner (means Change, *JAISE AC KO AAP LOG CHANGE TEMP***) of NN**: is the process of selecting the optimal values for a machine learning model's hyperparameters. Hyperparameters are settings that control the learning process of the model, such as the learning rate, the number of neurons in a neural network, or the kernel size in a support vector machine. The goal of hyperparameter tuning is to find the values that lead to the best performance on a given task.

https://www.geeksforgeeks.org/hyperparameter-tuning/

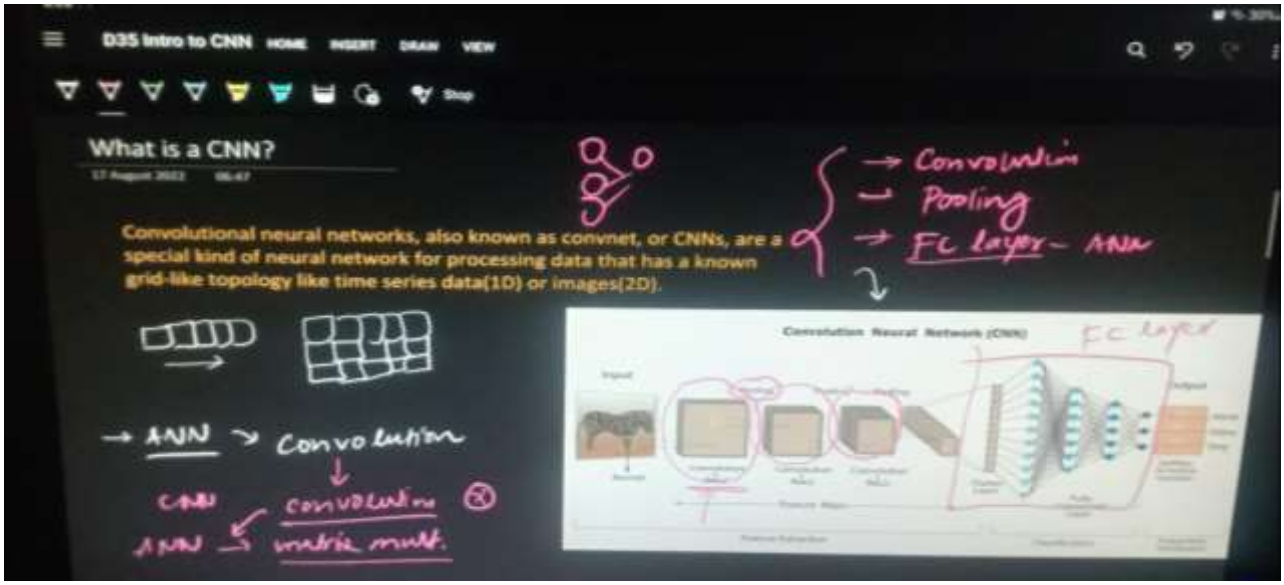
Models can have many hyperparameters and finding the best combination of parameters can be treated as a search problem. The two best strategies for Hyperparameter tuning are:

- ✓ Grid Search CV
- ✓ Randomized Search CV
- ✓ Bayesian Optimization

Chapt 02: Convolutional Neural Network (CNN):

Also known as Convnet, are a special kind of neutral network for processing data that has known gride-like topology like time series data (1D) or images (2D)
How differ from ANN? By convolution layer i.e., which works is convolutional operation use but in ANN (we use matrix-multiplication)

CNN have 3 layers: Convolutional, pooling, FC layer (fully connected, like used in ANN)



CNN > ANN more perform on image data because of efficiency and widely or mostly used in real life e.g., bank cheque, OCR, handwritten, face lock / finger lock on phone.

Image = pixels of 2D grid

Why not ANN? (I) High computation cost, (ii) overfitting, (iii) Loss of imp. Info. Like spatial arrangements of pixels if you can the dimes ion of image i.e., 3D to 2D try to image the image by closing your eyes. Features extract of primitive features

Roadmap to go in this chapter: (I) Biological connects, (ii) Convolution-operation (padding + stride), (iii) pooling, ANN architecture and CNN v/s ANN, dog v/s cat classifier

Conclusion from experiment on cat:

- Simple cell: orientation cell, feature detector (edge detection), smaller receptive field,
- Complex cell: bigger receptive field or preferred stimulated

Padding & Strides = change the dimension of image

https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

Pooling operation:

Animation enjoying https://deeplizard.com/resource/pavq7noze3

CNN architecture (Let Net 5)

ImageNet is world competition happen for CNN. The annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is a platform that allows researchers to evaluate their algorithms and models. It brings the development of deep learning models for image classification, object detection, and other computer vision tasks.

Q. Diff b/w CNN and ANN?

Mini-Project: Cat-Dog classification

```
# folder direct data ko lo apna env mai

!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

```
# run command api command

!kaggle datasets download -d salader/dogs-vs-cats
```

Unzipping

```
# unzip this folder

import zipfile

zip_ref = zipfile.ZipFile('/content/dogs-vs-cats.zip', 'r')

# kaha pe rakhana h
zip_ref.extractall('/content')
zip_ref.close()


# !unzip dogs-vs-cats.zip
```

Concept of Generators in CNN: when the data set is given very big then it divides then generators is working i.e., the dataset into the Batches

Data Augmentation in deep NN: building powerful image classification using very little data.

Why use Pretrained models? ImageNET dataset, ILSVRC – imageNET challenges, AlexNET: CNN with GPU and relu as activation

https://www.image-net.org/challenges/LSVRC/

https://keras.io/api/applications/

```
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input,
decode_predictions
```


1st July'24 (Monday)

Transfer Learning: is a technique where you can train a model on the dataset and run on another new dataset. the reuse of a pre-trained model on a new problem.

Why Keras-**Sequential model is failed** for creating non-linear NN? But smoothly work on the linear.
Ans: because sequential API can't properly work on non-linear. E.g., take human in which you want to predicate their age (a Regression problem) and another to classify emotion, happy, sad (classification problems). So, we need Keras - Functional model.

Blog for functional API https://machinelearningmastery.com/keras-functional-api-deep-learning/

Chapt 03: Recurrent Neural Network (RNN)

Now we have read that ANN works on tabular data and CNN for image data where as RNN is for sequential data (given text i.e., NLP area).

RNN is a type of sequential model to work on sequential data (textual, or time series data, speech, audio in the form of wave, DNA sequential).

Why RNN? Ans: we give sentence in English & convert into binary (0 or 1) vectorize by One-Hot encoding,

Word Embeddings: represent words as dense vectors in a continuous vector space, where each word's vector captures semantic relationships and contextual meanings based on its usage in a large corpus of text., Word2Vec and GloVe are popular algorithms for generating word embeddings. They learn to map words that appear in similar contexts to vectors that are close together in the embedding space.

TF-IDF (Term Frequency-Inverse Document Frequency): evaluates how important a word is to a document in a collection or corpus. It is often used for feature extraction in text mining and information retrieval. Word Vectors via Neural Networks:

Reason: (I) Text input-varying image, (ii) zero padding-unnecessary computation, (iii) Prediction problem, (iv) totally disregarding the sequence

Application: (I) sentiment analysis – given a sentence and you have to say positive or negative review e.g., Flipkart, (ii) Image caption, (iii) sentence completing, (iv) Google translator, (v) forecasting, and speech recognitions

Roadmap: simple RNN, Backpropagation, problems and solution (LSTM + GRU), Types of RNN, Deep RNN, Bi-directional RNN, Seq2Seq

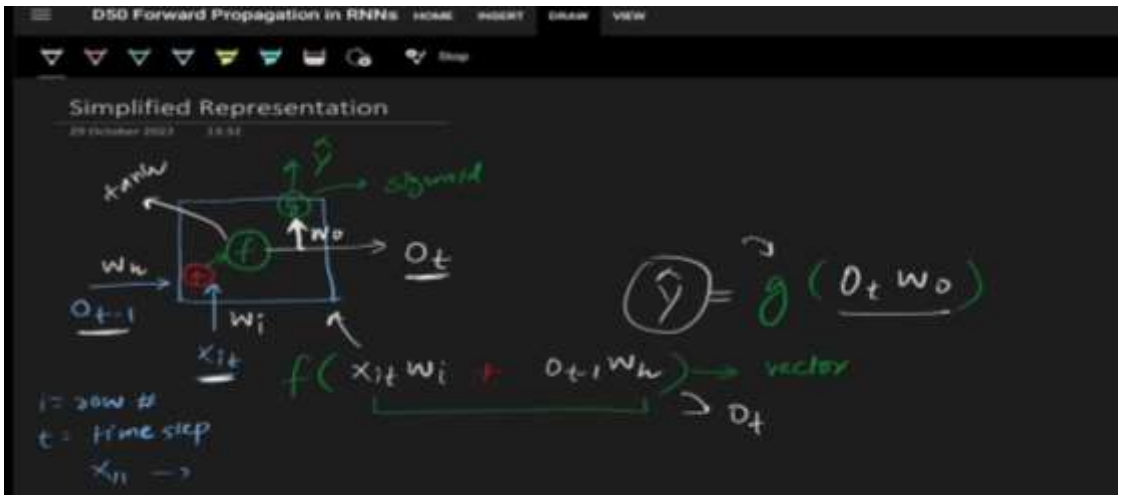
Data for RNN is in form of timesteps & input-features

Working of RNN: How differ? RNN is similar to ANN, but two difference in RNN on each time (t1, t2…etc) one input is passed (X11, X12, X13) and second is back feed happen in RNN (this makes RNN) not in ANN

How ***forward connection*** happen in RNN? By unfolding three times.

- Because your hidden layer again and again occurs as input change, that's why you can said Recurrent NN
- Taking weights + bias as same on each new input, which is called SHARING-Parameters occurs.
- RNN can process the sequence of hidden occurs.
- A simple RNN can process a sequence of 10 times timestamps.

Structure of RNN



Tokenization = (sentence into words, Capital letter to small and remove special characters or symbol)
Padding = take all in same range
Sentiment analysis from RNN:

- Step1: convert sentence into vector using (integer encoding and embeddings), can form vocabulary from unique word from document, and form index of each i.e., Tokenization
- Step2: replace sentence with their index value
- Step3: padding-take all in same range, and get vector
- Step4: once the vector is found and pass to RNN model and find your answer.

```
# generate sequence to each sentence karo aap

sequences = tokenizer.texts_to_sequences(docs)
sequences
```

Embedding: can capture semantic meaning, dense dimensions, word-content, some technique (read in NLP like Word2VCC, Glovc). In deep learning – embedding layer use which create dense layer in RNN and give sentiment analysis.

Types of RNN: (I) Many to one, (ii) One to many, (iii) Many to many, (iv) one to one

I. Many to one: Input-sequence, output-non sequence
II. One to many: normal non sequential, output-sequential, e.g., image captioning
III. Many to many: Input-sequential data and output – sequential having two types same length (input seq == output sequence, used in making Chatbot), variable length: Machine translator in which 1 lang. to 2 lang. used in google translator, Encoder & Decoder
IV. One to One: this is not RNN but a simple ANN

2nd July'24 (Tuesday) **Backpropagation & LSTM**

RNN use for sequential data e.g., textual or time series data

Stagnant training = not properly training (exploding gradient problems)

Problems with RNN – two major (i) Problem of long-term dependencies i.e. vanishing gradient and (ii~~)~~ ~~unstable gradients~~, stagnant training: forget the short-term memory loss (*like Amir khan learn only for 15min in GAGANI movie* = Vanishing gradient problem)

Problem1 - Vanishing gradient problem: when your NN is very big, then the value become small. Solutions: different activation-relu/leaky relu, better weight limit, skip RNN, LSTM

Problem2 – Unstable training (exploding gradients): it's solution Gradient clipping, controlled learning rate, LSTM

3rd July'24 (Wednesday) **LSTM**

Why need LSTM? Because of two reason:

- Reason1: Architecture (str. wise) – RNN have only path to learn whereas LSTM have two; one for short terms (STM) & another for Long (LTM) to handle complex problem when the statement become very big!
- Reason2: Two state in LSTM (established the communication b/w the STM + LTM) and one state in RNN for input and output

Story is told by the sir i.e., Vikram v/s XYX, Vikram JR, and Vikram Super-JR of Pratagradh, short term contest and long term contest for 1000 years' old. Guys what's happen?
Movies: Ghajini by Amir Khan, forget the memory after 15 minutes, learn by write anything, Short term memory lost problems.

Scientist realize that we don't have any mechanism which can relate the LTM and STM that's how scientist take the action to develop the new network NN i.e., LSTM.

Core Idea behind LSTM:

On the basis of current input, we can take that forward to LTM which is happen in short term if required by established the connection b/w STM and LTM known for the cell state.

Blogs of LSTM link https://colah.github.io/posts/2015-08-Understanding-LSTMs/

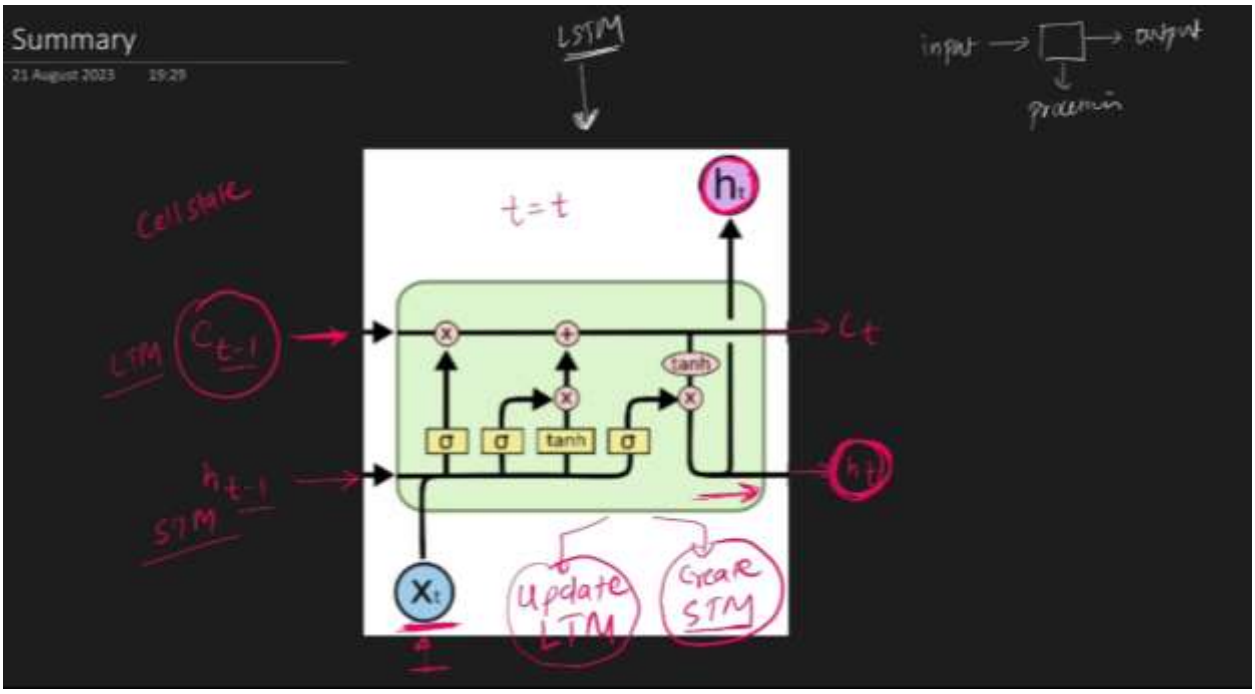
RNN (simple str.) is just mending the single state whereas the LSTM (complex str.) have two that's how both layer can communicate to each other.

Chapt 04: LSTM

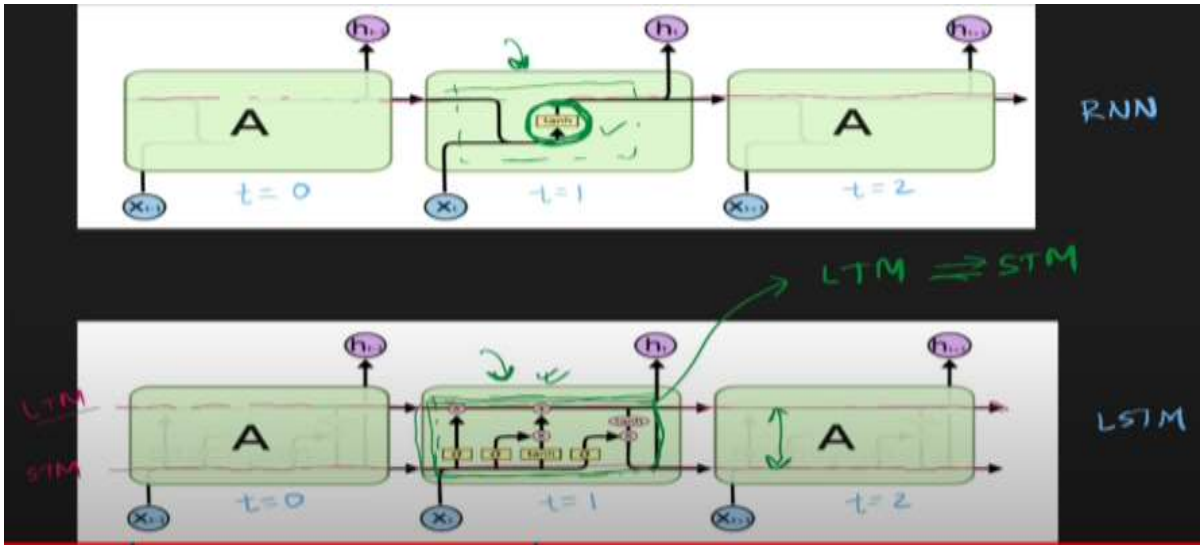
Architecture of LSTM:

Three gates – Forget gate, input and output gate. Their working:

- Forget gates: remove the thing from LTM
- Input: add to LTM
- Output: final outcome from the model, end the story and also create the short term memory for the next state



Summary for LSTM: it is a just like a computer definition (which you learn in the childhood) update the LTM and create STM.

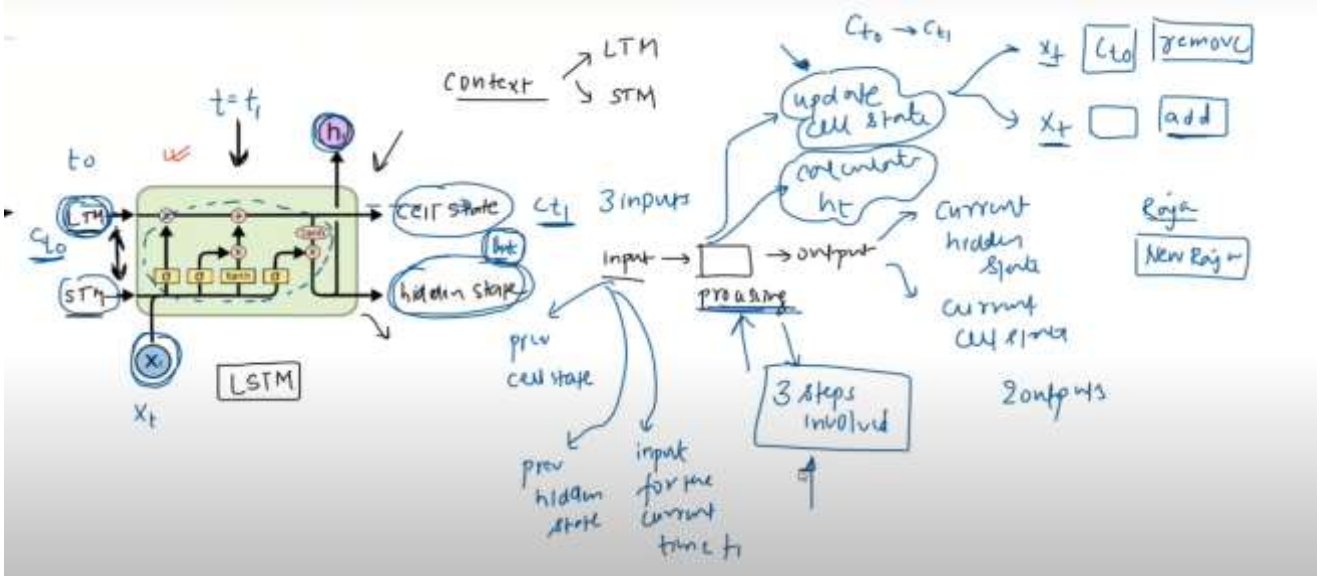


Diff b/w RNN and LSTM structure wise

LSTM (**working – principle**):

In the LSTM, long term path is known for Cell State and short term for Hidden state.

For input side, have previous cell state, hidden state & input for the current time t1 and for the outside, have current hidden state and current cell state. In processing part: update the cell state Ct1 and hidden state Ht1 involved three steps.



3 inputs are taken by LSTM and give 2 outputs

The **Gates** (only two work: update the cell state and to calculate the hidden state) in LSTM:

- Forget Gate: to remove something from C_t
- Input gate: adding
- Output gate: to calculate the hidden state h_t

Rule – never break in LSTM: dimension of Ct and Ht are never be different. Always be same. Both Ct and ht are 3d vector.

X1? answer: mathematically speaking X1 is also a vector (we do Vectorization).

What are ft, it, ot, and ct ?

Answer: ft is in forget gate, it and ct in input gate, ot in output gate. All must have same dimension.

Pointwise operations: (I) Pointwise multiplication, (ii) additions and (iii) pointwise tanh. Dimensions are same and the output is vector.

Neural Network Layers: Each layer has the activation function and have flexible no of node units; depend on you how much take (which is called Hyper-parameters, you can decide), but if you assume the 4 node in one layers then it must be 4 for rest all.

4th July'24 (Thursday) LSTM

Forget gate:

Input gate: works on 3 stages – (I) calculate the Ct candidate cell state, (ii) find out It, (iii) finally we get Ct current cell state

Embedding layer – give a dense layer give

Q. How to improve the performance of the model?
Answer: (I) more data, (ii) Hyper-parameter tuning, (iii) use the advance architecture (i.e., Stacked LSTM, Bi-directional LSTM, Transformer)

Gated Recurrent Unit (GRU in deep learning)

Coding on own laptop is known as development environment. Machine learning have two type:

- Batch learning – offline & Online learning (utilize the entire data for trading, you can go to the server but it is costly, that's why data scientist use the offline the laptop to train on the model, then go to online ***static modal***). Train the model frequently (test + upload on server.

Type of ML: (I) Supervised L = Regression + Clustering, (ii) Unsupervised L = clustering + dimensionally reduction + anemology detection + association, (iii) Semi – supervised, (iv) Reinforcement L

https://github.com/campusx-official/online-ml-sklearn-demo/blob/main/online_ml.ipynb

Online machine learning: train the model on the server directly e.g., Chabot, Alexa. On the click the data is generated, with mini-batch we can train the model.

Q. When to use Batch learning and online ML? Answer: when the nature of the product is change (= when there is **concept drift**) we online learning in e-commerce sell coming, cost effective, faster solution.

https://riverml.xyz/dev/

River is a python library for online machine learning. https://www.geeksforgeeks.org/difference-between-batch-processing-system-and-online-processing-system/

How machine learning learns? Answer: Learning (memorizing + generalize)

- **Instance based learning (LAZY – LEARNER)**: model (*RATANE – LAGATA HAI*) same thing, training data we don't learn anything; directly store the value, when the question is asked then I answered you. It doesn't build any model only on basis of similarity e.g. Kernel function, KNN, (less e.g.)

  https://vitalflux.com/instance-based-learning-model-based-learning-differences/

- **Model based learning**: algorithms try to analysis the behavior of data and classify the problem, training point, query point, e.g. Linear + logistic Regression, Decision tree (more e.g.)

5th July'24 (Friday) MLDLC

Challenge with ML: - in form of **data or Algorithms**

(I) Data collection (either from API or web – scrapping from google), (ii) Insufficient data / labelled data (NLP), (iii) Non – Representative data (sampling noise + sampling bias e.g. who win the world cup cricket match), (iv) Poor Quality data, (v) irrelative features (one dialog: Garbage IN Garbage OUT), (vi) Feature Engineering: mix two or three feature in single (e.g. Height + weight = BMI (body mass index), (vii) Overfitting & under fitting, (viii) Software Integration: which help the user in form of part of any software, convert into website or android apps. (ix) Offline learning & Deployment, (x) cost involved

Cost of AI: https://research.google/pubs/machine-learning-the-high-interest-credit-card-of-technical-debt/

MLOPS – new training scope field in ML

Real – life **application** of ML:

(I) Retails – amazon / big bazar, if you are paying for the product then you are the product in the internet contest, Arrange or Position of Oil box or any items by Associate based learning ML = find out Correlation of items (ii) Banking & Finance: compare your profile with past defaulter in loan, where bank branch open, stock, (iii) Transport OLA & Uber (user apps + driver apps): demand & supply forecasting, (iv) Manufacturing – Tesla: IOT based sensor or device, fault occurred, robotic arms, predicative maintaining, (v) consumer internet – Twitter: sentiment analysis, IMDB movie sentiment reviews .

MLDLC (Machine learning Development Life Cycle): 9 steps

It is a guideline for how the ML life cycle or software is developed end to end.

I. Frame the problems
II. Gathering data (.csv, API from JSON file, web-scrapping, database warehouse = DW, ETL = extract transform load, spark form and fetch)
III. Data Pre-processing: remove duplicate & misguide, outliers, scale
IV. EDA (Exploratory data analysis): try to learn the relation b/w input and output from the data, univariate & multi-variant, outliner detections, Impedances
V. Feature Eng. And Selection: create new features
VI. Model training, evaluation & selections: performance matrix, tune (or change) the best parameters i.e. Hyper-parameters tuning, Ensemble learning (combine two small model to make single)
VII. Model development
VIII. Testing: a/b testing
IX. Optimize: backups of model and data, load balancing, frequently retrain (rotting)

What is Tensor?

Answer: is a data-structure (which store the data), n – dimension array, tensor is a container where we store a number (vector, matrix)

Note: when 0-dimesion = scalar (or one number), when list of number = vector, when in 2D = matrix, general term for n-dimension = tensor, Axis = no of dimension

1D tensor = Vector, 2D tensor or matrices

Note: no of axis = rank = no of dimension, concept of shape

Tools for data-science:

- Anaconda (Syder: inbuild), Anaconda navigator, Virtual env need when deploy
- Jupyter notebook
- Kaggle: here no GPU
- Google Colab: online version of Jupyter: deep learning with GPU + TPU very imp step for deep L because take data from Kaggle and use with GPU of google Colab
  step1: go to Kaggle a/c and create new API token + download as .Json file and
  step2: upload to google drive as temporary
  Step3: select data on Kaggle and copy the API command, paste on google Colab code before exclamatory marks!

```
!kaggle datasets download -d wobotintelligence/face-mask-detection-dataset
```

```
# command

!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

```
# unzip

import zipfile
zip_ref = zipfile.ZipFile('face-mask-detection-dataset.zip', 'r')
zip_ref.extractall('/content')
zip_ref.close()
```

EDA = Exploratory data analysis, or data preprocessing,

Framing the problems:

1. Business prob to ML prob
2. Types of prob
3. Current solution
4. Getting data – data Engineering to build DW (data warehouse)
5. Matrices to measure
6. Online v/s Batch learning
7. Check assumptions
8. At Ankit ok use your own through process on pen and paper like data mining concepts applying

6th June'24 (Saturday) Data Handling, **Scrapping**

Working with .CSV (commas speared value) & .TSV file (tab separated value)

pd_csv('', sep = '\t')

working with JSON / SQL and fetching data from an API: e.g. railway ticket booking

movies API https://developers.themoviedb.org/

login on this site and generate own API key

https://jsonviewer.stack.hu/

Rapid API is taking https://rapidapi.com/collection/list-of-free-apis

Campus X YouTube channel https://github.com/campusx-official/100-days-of-machine-learning/tree/main/day17-api-to-dataframe

Web-Scrapping: taking information about site from their URL documentation src link https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Web scraping is an automatic method to obtain large amounts of data from websites. Most of this data is unstructured data in an HTML format which is then converted into structured data in a spreadsheet or a database so that it can be used in various applications.

Types of Webs Scrapers

Web Scrapers can be divided on the basis of many different criteria, including Self-built or Pre-built Web Scrapers, Browser extension or Software Web Scrapers, and Cloud or Local Web Scrapers.

Web scraping requires two parts, namely the crawler and the scraper.

- The **crawler** is an artificial intelligence algorithm that browses the web to search for the particular data required by following the links across the internet.
- The **scraper**, on the other hand, is a specific tool created to extract data from the website. The design of the scraper can vary greatly according to the complexity and scope of the project so that it can quickly and accurately extract the data.

e.g. This is a portal where the company listed
https://www.ambitionbox.com/list-of-companies?page=1

Understanding your data: (I) Asking basic ques, (ii) EDA (univariant + multi-variant), (iii) Pandas profiler

Asking Ques:

1. How data is big? Ans: df.shape()
2. Data looks like? Ans: df.sample()
3. Datatypes of column? Ans: df.info ()
4. Is any missing value? Ans: df.isnull().sum()
5. How data mathematically looks like? Ans: df.describe()
6. Any duplicate value? Ans: df.duplicated().sum()
7. How are the correlations b/w columns? Ans: df.corr()['ANKIT']

Uni – Variate analysis: independence analysis of each column in data
Bi-Variate: at a time, two columns analysis, more than 2 called multi
Data-Types: Numerical (e.g. height, age,) + Categorical (e.g. country,)

EDA USING UNI-VARIANT

1. CATEGORICAL DATA
   use Count plot (seaborn library): sns.countplot(df[' '])

```
sns.countplot(df['Embarked'])
#df['Survived'].value_counts().plot(kind='bar')
```

note: we can plot any Uni-variant any time

Pie-Chat: for better visualization in term of percentage

2. NUMERICAL DATA
   - Histogram: help in know the distribution in dataset (by creating a range)
   - Dist plot: called PDF (probability density function)

```
Dist-plot

sns.distplot(df['Age'])
```

   - Box-plot: used in statistics as 5 – Number Summary concept, easily identify the potential outliers and eliminate from dataset, in future, when work with noisy data, then Ankit uses box plot – how many data is noisy or outliers

Quartiles are values that divide a dataset into four equal parts, each containing approximately 25% of the data, SKEWNESS + Normal distribution.
A five-number summary is especially useful in descriptive analyses or during the preliminary investigation of a large data set. A summary consists of five values: the most extreme values in the data set (the maximum and minimum values), the lower and upper quartiles, and the median.

EDA USING Bi and Multi-VARIANT

NUMERICAL DATA

- Scatter plot: used to show relation b/w numeric and numeric data

```
sns.scatterplot(tips['total_bill'],tips['tip'],hue=df['sex'],style=df['smoke
r'],size=df['size'])
```

- Bar plot: when Numerical – Categorical data relation
- Box plot: Numerical – Categorical
- Dist plot: Numerical – categorical using seaborn library
- Heat map: on seeing the colour of grid, we can clearly visualize the data of categorical – categorical
- Cluster map: categorial – categorical dataset, tell the closeness b/w 2 values
- Pair plot: for numerical – numerical

Pandas Profiler: provides a solution to this by generating comprehensive reports for datasets that have numerous features. https://pypi.org/project/pandas-profiling/

YDATA – a python library for manipulation the data in form of web-pages *https://ydata.ai/*

```
# import ydata python library instead of pandas-profiling

!pip install ydata-profiling

from ydata_profiling import ProfileReport

# create profile
profile = ProfileReport(df)

# save report
profile.to_file(output_file='output.html')
```

Note: We explore 2 PY – library for data scientists very amazing Ankit (pandas-profiling and ydata-profiling) helps much, but some things pandas-profiling not working properly, then use ydata-profiling (in Ankit case).

https://github.com/ydataai/ydata-profiling

Data **Wrangling**: process of transforming and structuring data from one raw form into a desired format (GUI) with the intent of improving data quality and making it more consumable and useful for analytics or machine learning. Using BAMBOOLIB

https://docs.databricks.com/en/notebooks/bamboolib.html

EDA with SWEETVIZ https://pypi.org/project/sweetviz

Sweetviz is an open-source Python library that generates beautiful, high-density visualizations to kickstart EDA (Exploratory Data Analysis) with just two lines of code. Output is a fully self-contained HTML application.

```
# anlaysis

import sweetviz as sv
analyze_report = sv.analyze(df)

analyze_report.show_html('analyze.html', open_browser=False)
```

application.

https://pypi.org/project/pandasgui

https://github.com/adamerose/pandasgui

8th July'24 (Monday) **Search** (Random, Grid, Bayesian)

Skikit learn Pipelines: Pipelines chains together multiple steps so that the output of each step is used as input to the next step.

Pipelines makes it easy to apply the same pre-processing to train and test.

Cross Validation (CV): *JAISE MUMMY ASKED, CHECH DOOR IS OPEN / CLOSED 2 – 4 TIMES?*

What mummy try to observe? Validate the work is done properly or not, something is applied on the deep learning that train & test the modal 2-3 times by CV, and have freedom to give more options RELU / SIGMOID, no of epoch runs etc.

- Grid Search CV: use when the no of record is small or around 10,000 and imitated no of column.
- When the dataset is very big then randomize that CV i.e. Random Search CV if CV = 5, hyper-parameter tuning, give no of iteration, distribution, MD = max. depth, and N. tree = no of tree + best model or params.

  Note: Random search CV (when dataset is very big): is faster and give output quickly but not best answer because it doesn't consider all. So, for best we should try first Grid search, then random, then Bayesian search while hyper-parameter tuning.

https://www.npci.org.in/

9th July'24 (Tuesday) Dropout

Benefits of **Dropout** (Ensemble Learning):
Reduces Overfitting: By preventing complex co-adaptations on training data, dropout helps the model to generalize better on unseen data.
e.g. Dropout (0.3) similarly means that each neuron in the previous layer has a 30% chance of being ignored during training.
Keras Wrappers submodules for hyper-parameters tuning https://pypi.org/project/scikeras/0.1.8/

10th July'24 (Wednesday) **Hyper-parameter**s tuning (Grid + Random Search CV) for time series data LSTM

Two big questions?
What is model-parameters and model-hyperparameter?
A **model hyperparameter** is a configuration that is external to the model and whose value cannot be estimated from the data and whereas a **model parameter** is a configuration variable that is internal to the model and whose value can be estimated from the given data.

In the other words, a hyperparameter is used to construct the structure of the model and **cannot be learned from the data** and its value is set before the learning process begins. Therefore, hyperparameters are like the settings of an algorithm that can be adjusted to optimize performance and prevent overfitting. This is exactly what we do in the hyperparameter tuning. We try to choose a set of optimal hyperparameters for a learning algorithm to enhance the performance of the model.

Blog src https://www.datacamp.com/tutorial/parameter-optimization-machine-learning-models
There are two frequently used methods to perform hyperparameter tunning called 1) Grid Search and 2) Random Search.

12th July'24 (Friday) Bi-directional, ML model **ROLLOUT**

By Prof Hiranya Nath (USE, summer bootcamp IIITG) suggest me ROLLOUT concept to improve the performance of model.

Q. Can we plot the graph like the live stock market candle-stick; our intuition is that predicated close price (which is next 5days) to see that exact value when I hove the mouse cursor on the graph?

Ans: Yes, by PLOTLY

```
!pip install plotly

import plotly.graph_objects as go # Import the graph_objects module from
plotly
```

```
# Plot the actual and forecasted closing prices like live stock market view
fig = go.Figure()

# Add candlestick chart for the actual data
fig.add_trace(go.Candlestick(x=data['Date'],
                open=data['Open'],
                high=data['High'],
                low=data['Low'],
                close=data['Close'],
                name='Actual Close Price'))
```

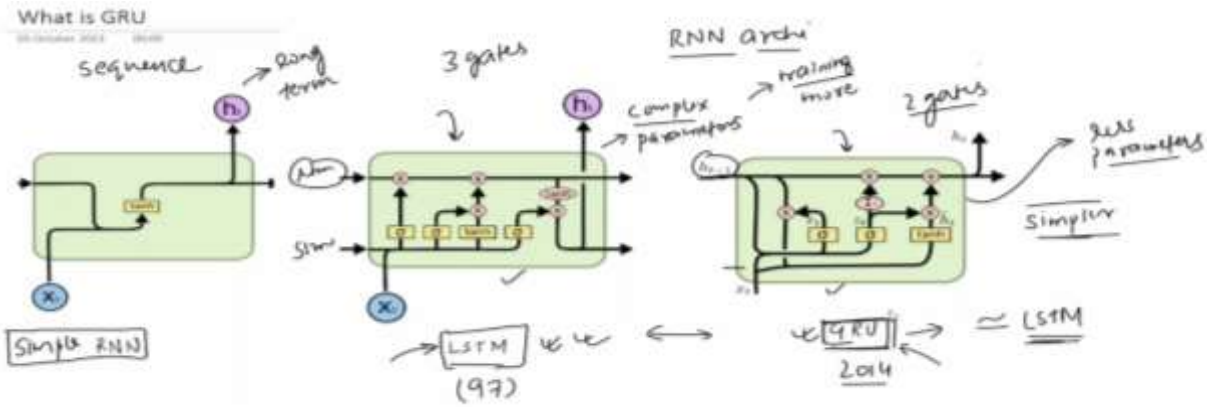
```
# Plot the actual vs predicted closing prices (2002 to 2024) for the entire
dataset
fig2 = go.Figure()

fig2.add_trace(go.Scatter(x=pd.to_datetime(comparison_df['Date']),
y=comparison_df['Actual_Close'],
                          mode='lines', name='Actual Close Price'))
fig2.add_trace(go.Scatter(x=pd.to_datetime(comparison_df['Date']),
y=comparison_df['Predicted_Close'],
                          mode='lines', name='Predicted Close Price'))

fig2.update_layout(
    title='Actual vs Predicted Close Prices (Entire Dataset: 2002 to 2024)
using simple RNN Multi-Variant',
    xaxis_title='Date',
    yaxis_title='Close Price',
    legend_title='Legend',
    hovermode='x unified'
)

fig2.show()
```


Chapt 05: Bi-Directional LSTM

13th July'24 (Saturday) GRU, Bi-Directional LSTM, **Deep** RNN (Stacked RNN, **Stacked** LSTM, Stacked GRU)

Why need Gated Recurrent Network (GRU)?

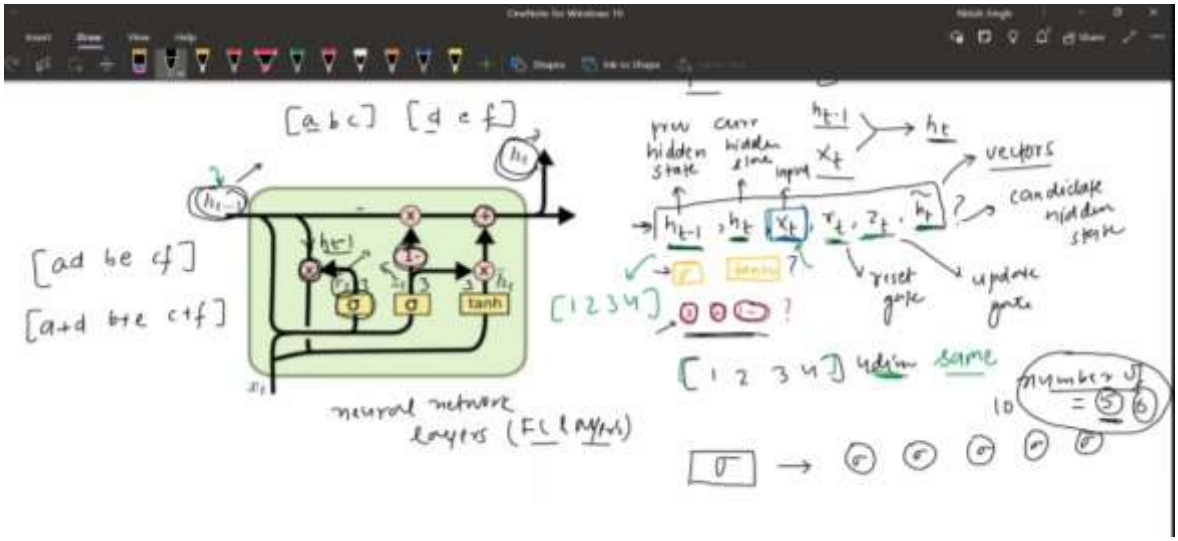
Ans: Problem with RNN: long term, LSTM: have architecture is complex + no of parameters more + training time more, but GRU have 2 gates, less training, not in every case this is the best.



The Big Idea Behind GRU

GRU don't have cell state and only have hidden state which can handle Long + short term memory, 2 gates = Reset gate and update gate.

The **Setup**



[1, 2, 3, 4] = 4D-Vectors = previous hidden state, current hidden state, input, reset gate, update gate, candidate hidden state

F = fully connected NN

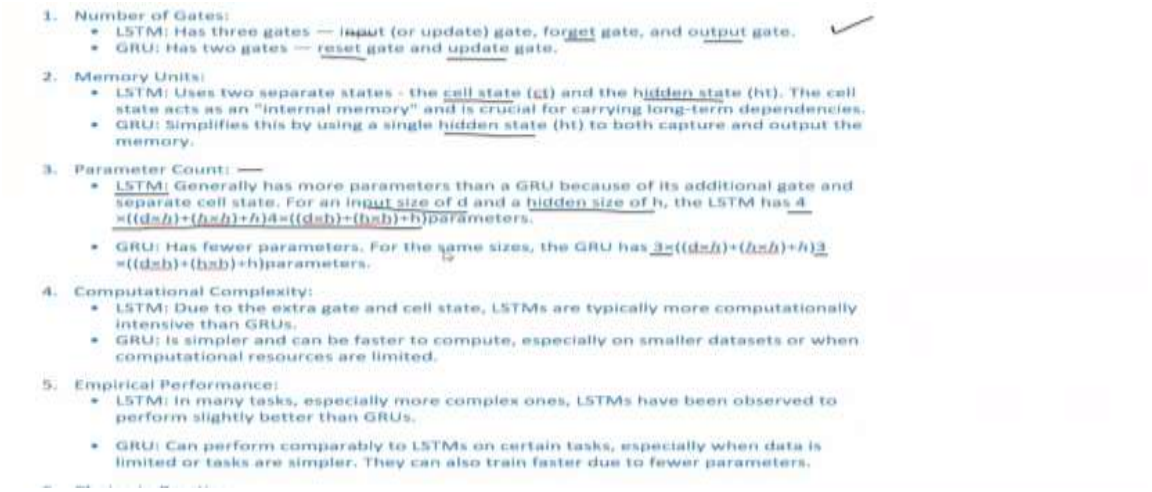
Neural Network layers (F – layers), eliminate – wise operations

Architecture of GRU

4 steps: ht-1 and Xt and ht

Step1: calculate rt (reset gate)
Step2: calculate **ht** (candidate hidden state)
Step3: calculate Zt (update gate)
step4: calculate ht (current hidden state)

Difference b/w LSTM & GRU



1. Number of Gates:
   - LSTM: Has three gates — input (or update) gate, forget gate, and output gate.
   - GRU: Has two gates — reset gate and update gate.
2. Memory Units:
   - LSTM: Uses two separate states - the cell state (ct) and the hidden state (ht). The cell state acts as an "internal memory" and is crucial for carrying long-term dependencies.
   - GRU: Simplifies this by using a single hidden state (ht) to both capture and output the memory.
3. Parameter Count:
   - LSTM: Generally has more parameters than a GRU because of its additional gate and separate cell state. For an input size of d and a hidden size of h, the LSTM has 4 ×((d×h)+(h×h)+h)4=((d×h)+(h×h)+h)parameters.
   - GRU: Has fewer parameters. For the same sizes, the GRU has 3×((d×h)+(h×h)+h)3 ×((d×h)+(h×h)+h)parameters.
4. Computational Complexity:
   - LSTM: Due to the extra gate and cell state, LSTMs are typically more computationally intensive than GRUs.
   - GRU: Is simpler and can be faster to compute, especially on smaller datasets or when computational resources are limited.
5. Empirical Performance:
   - LSTM: In many tasks, especially more complex ones, LSTMs have been observed to perform slightly better than GRUs.
   - GRU: Can perform comparably to LSTMs on certain tasks, especially when data is limited or tasks are simpler. They can also train faster due to fewer parameters.

And

- GRU: Has fewer parameters. For the same sizes, the GRU has 3×((d×h)+(h×h)+h)3 ×((d×h)+(h×h)+h)parameters.

4. Computational Complexity:
   - LSTM: Due to the extra gate and cell state, LSTMs are typically more computationally intensive than GRUs.
   - GRU: Is simpler and can be faster to compute, especially on smaller datasets or when computational resources are limited.
5. Empirical Performance:
   - LSTM: In many tasks, especially more complex ones, LSTMs have been observed to perform slightly better than GRUs.
   - GRU: Can perform comparably to LSTMs on certain tasks, especially when data is limited or tasks are simpler. They can also train faster due to fewer parameters.
6. Choice in Practice:
   - The choice between LSTM and GRU often comes down to empirical testing. Depending on the dataset and task, one might outperform the other. However, GRUs, due to their simplicity, are often the first choice when starting out.

Deep RNN (Stacked RNN):

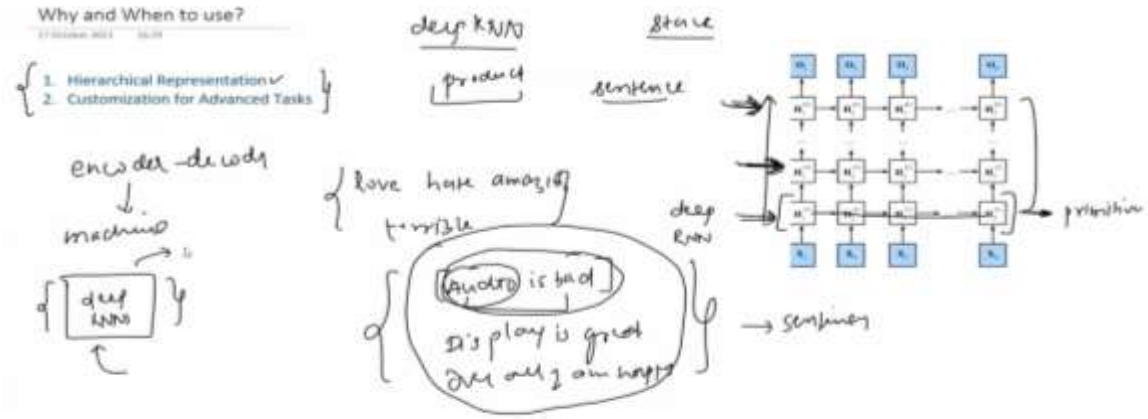
Deep RNN, Stacked LSTM. Stacked GRU, how related

Deep RNN = stacking one RNN cell on top of another RNN, and then unfolding these on the time

https://www.linkedin.com/pulse/deep-rnn-dhiraj-patra

When use deep RNN? (I) Complex tasks (for speech recognitions in machine translation), (ii) Large details: overfitting, (iii) computational, (iv) simpler models: deep RNN

https://d2l.ai/chapter_recurrent-modern/deep-rnn.html

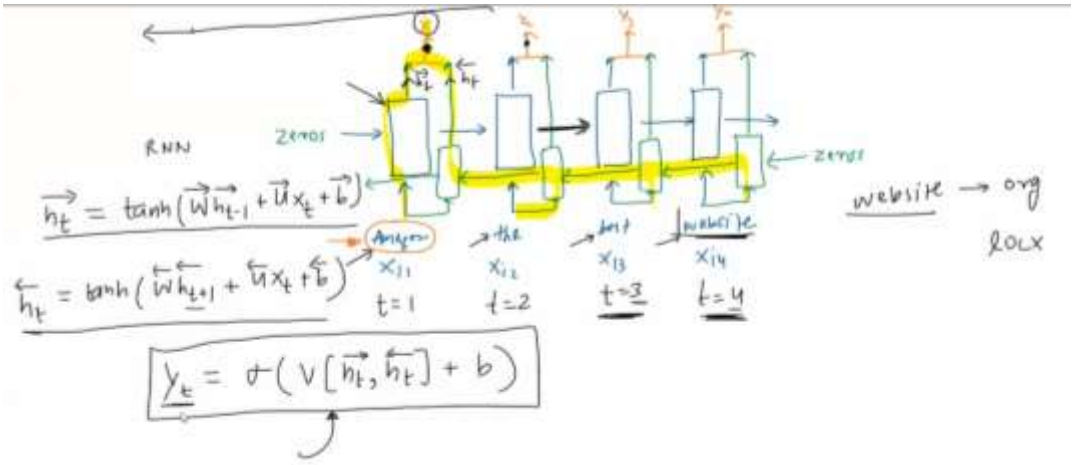


14th July'24 (Sunday) Bi-Directional LSTM (RNN or GRU), LLM

Bi-Directional LSTM

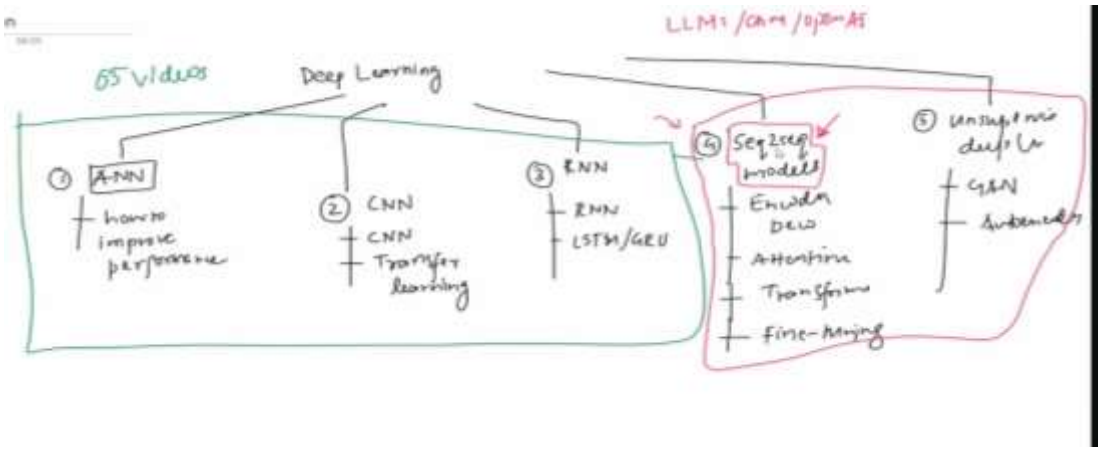
NER: Name entity recognition, used in making chatbot, by NLP,

Architecture of Bi-Direction RNN:
It has 2 RNN – one is forward & another one is backward and concatenate the output of each other



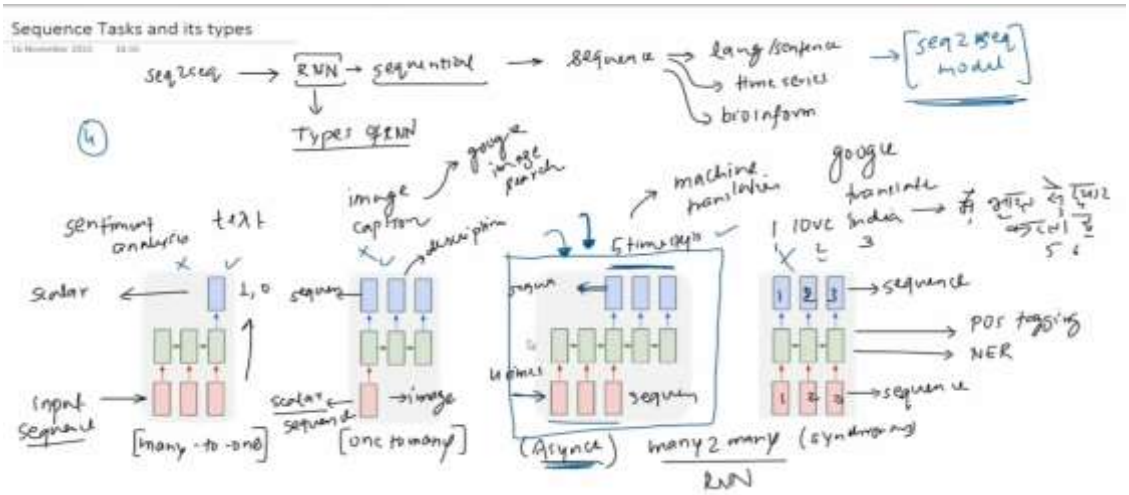
Application + drawback: (I) NER, (ii) POS tagging, (iii) machine translations, (iv) sentiment analysis, also (v) apply on time-series data in finance economics – which is Ankit, Research: Field of Interest at IIITG
Drawbacks: (I) Complexity, (ii) real-time recognitions, cause the late issue

Modules of campusX YouTube channel



Chapt 06: LLMs (Transformers & Transfer Learning)

Story of LLMs: (best e.g. Chat GPT history, Ankit just enjoys the story)
5-parts: The Epics History of Large Language Models (LLMs), from LSTM to ChatGPT

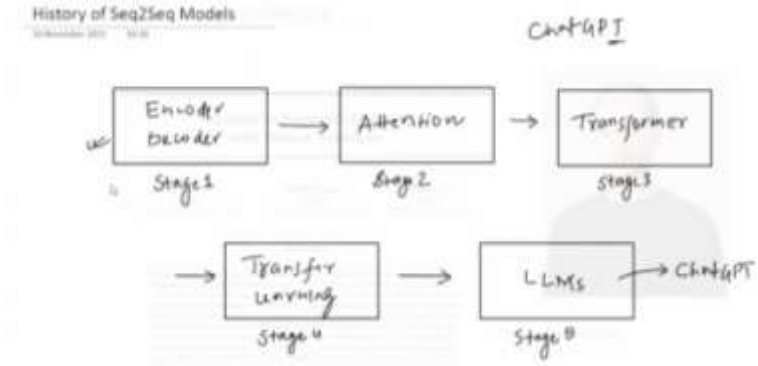


Sequence Tasks and its types (**seq2seq model)**:

Asynchronous - 4 times, you provide input output have in 5 times steps (best e.g. Machine Translation, or google translation) – this problem is solved by seq2seq model

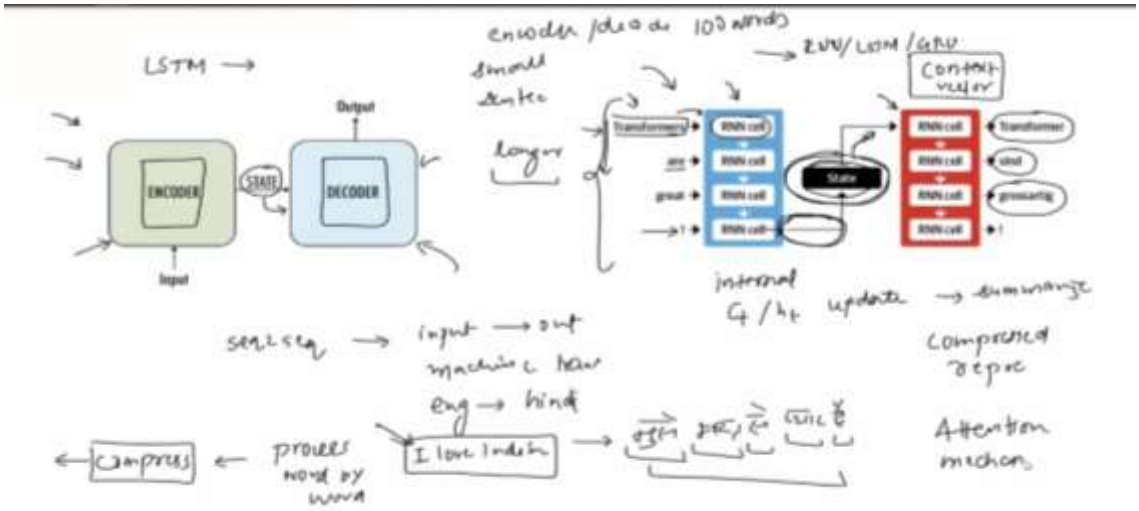
Uses of seq2seq model is: (I) text summarization, (ii) Q&A, (iii) Chatbot – input: text and output: text form, (iv) speech-to-text: seq2seq used by YouTube for subtitle

History of Seq2Seq Models: initial uses for encoder-decoder



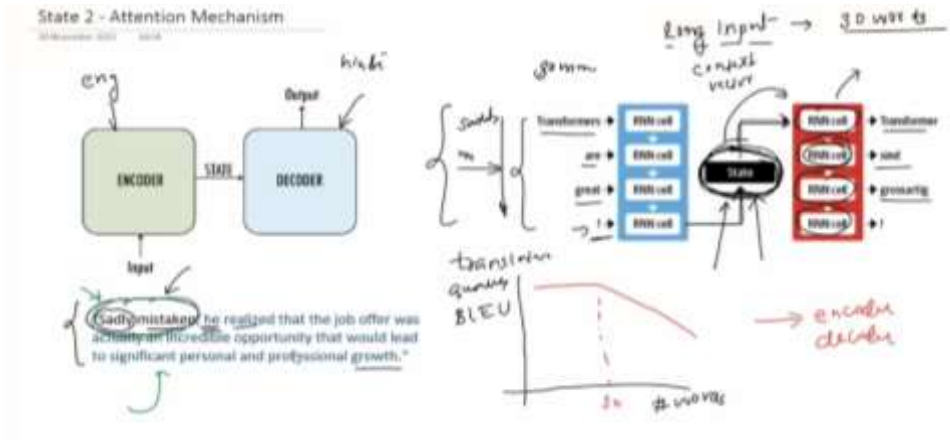
ChatGPT uses the transformer, Computer vision,

#Stage1: **Encoder-Decoder** works good on the small sentence, but not on large sentence; contact vector – short term memory lost, then develop a new mechanism i.e. ATTENTION-mechanism.

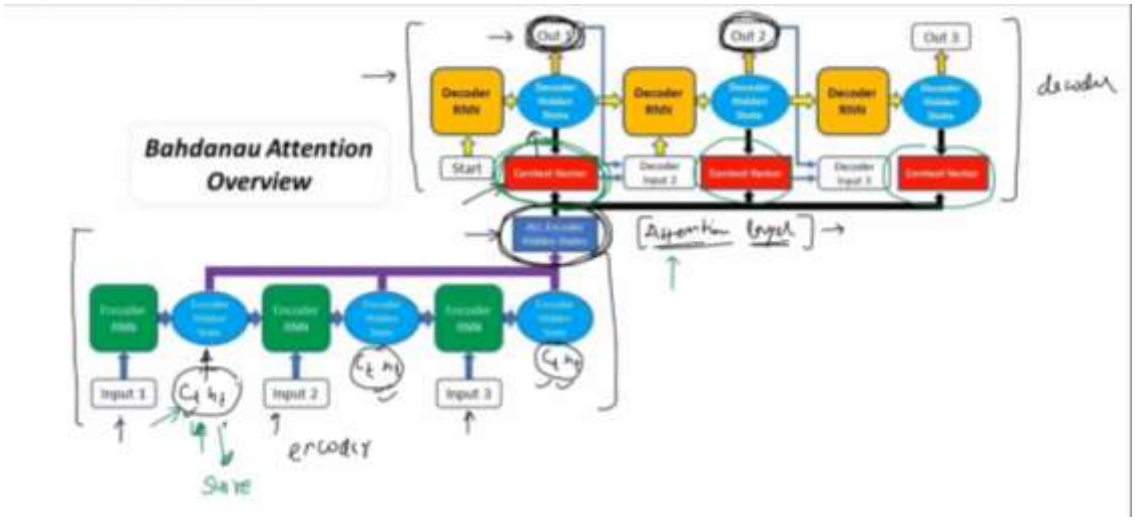


#Stage2: **Attention – Mechanism**

https://sh-tsang.medium.com/review-neural-machine-translation-by-jointly-learning-to-align-and-translate-3b381fc032e3



mechanism



#Stage3: **Transformers**: one step at a time (sequential nature remove) then come PARALLEL process which training time reduce in NLP landscape totally changed.

He said: "Our model doesn't need any RNN or LSTM cells, only ATTENSION IS ALL YOU NEED" very popular paper in 2012

https://arxiv.org/abs/1409.0473

#Stage4: **Transfer Learning**

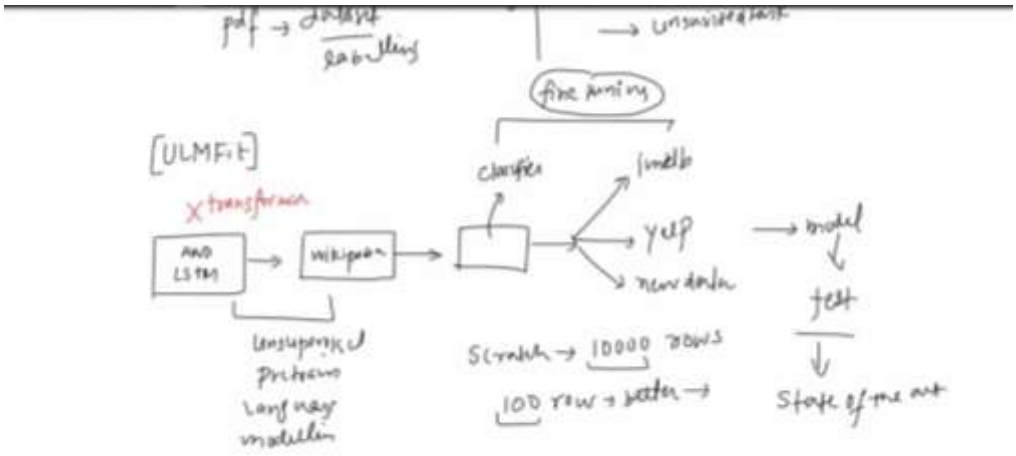
Problems: (I) Hardware: GPU which is costly, (ii) training time, (iii) data – a lot
Solution comes is Transfer learning

https://arxiv.org/abs/1801.06146

In NLP domain, transfer learning is not more applicable: (I) task specificity (Sentiment analysis, NER, POS, ML, summary), (ii) lack of data in ML translation,

ULMFiT research paper: for pre-training: he uses LANGUAGE MODELLING (a NLP task where try to learn next word predication)

Language modelling as a UNSUPERVISED pre-training task: (I) Rich feature learning (text classification, questing, text summarization, NEP), (ii) Huge available of data



ULMFiT paper comes in January 2018
We have best option: for Architure – we have transformer and training – we have transfer learning

#Stage5: LLMs (***JISKA DAR THA WAHI HUWA****: training by transfer learning + archit. made by transformer)*

October 2018: realized 2 papers: both is language, fine tune

- Google – BERT (encoder-based model only),
- OpenAI – GPT (is a decoder model only)

Data used is very big while training process, GPT2, and V3, because train on huge data person said LLM.

Q. When any LM called LLMs?
Ans: qualities of LLM:

1) data is diversity, 45TB
2) Hardware: cluster of GPUS, GPT3-superpower – 1000 NVIDA GPU

3) Training – days to weeks
4) Cost – hardware + electricity bills + infrastructure + experts scientists + big company or govt. or institutes IIT (still not any IIT do this kind)
5) Energy consumption

The Grand Finale – ChatGPT

Q. diff b/w ChatGPT and GPT? GPT is a model but ChatGPT is a application chatbot
e.g. HP laptop with intel processor, del, bard also uses GPT

Journey of GPT3 to ChatGPT

1) RLHF: reinforcement learning from human feedback, in two steps (I) supervised-fine tuning labelled dataset, (ii) apply the concept of reinforcement learning – response rank
2) In-corporate: safety and ethical guidelines (how to make atom bomb), minimize the biases and improve
3) Improve in context point
4) Dialogue specific training – conversion, better under: dialogues language
5) ChatGPT: continues improve – from user feedback

IIITG

Conclusion

The project successfully showcased the application of advanced neural network models in predictive analytics for stock market forecasting and diabetes data analysis. The findings highlight the importance of model selection and data preprocessing in achieving accurate and reliable predictions. The insights gained from this research can be utilized to develop more sophisticated models and drive future advancements in the fields of finance and healthcare.

This research was conducted under the supervision of Dr. Hari K. Chaudhary, with guidance from Sanjib Singha at the Indian Institute of Information Technology, Guwahati.

References

[1] User-images, "Venn Diagram of ML, DL, DS and Neural Network," Available: https://user-images.githubusercontent.com/7995307/64065525-153d9580-cbfe-11e9-9236-d9ffc3f34a43.png.

[2] GeeksforGeeks, "Artificial Neural Networks and its Applications," Available: https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/.

[3] YouTube, "Krish Naik," Available: https://www.youtube.com/channel/UCNU_lfiiWBdtULKOw6X0Dig.

[4] YouTube, "Five-Minute Engineering," Available: https://www.youtube.com/@5MinutesEngineering.

[5] YouTube, "CampusX," Available: https://www.youtube.com/@campusx-official.

[6] GeeksforGeeks, "Machine Learning," Available: https://www.geeksforgeeks.org/machine-learning/.

[7] GeeksforGeeks, "Deep Learning Tutorial," Available: https://www.geeksforgeeks.org/deep-learning-tutorial/.

[8] Advanced neural network
https://d2l.ai/chapter_recurrent-modern/deep-rnn.html
https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm