VibeChecker MID-EVALS

"Deciphering Patterns: An Implementation-First Approach to Deep Learning."

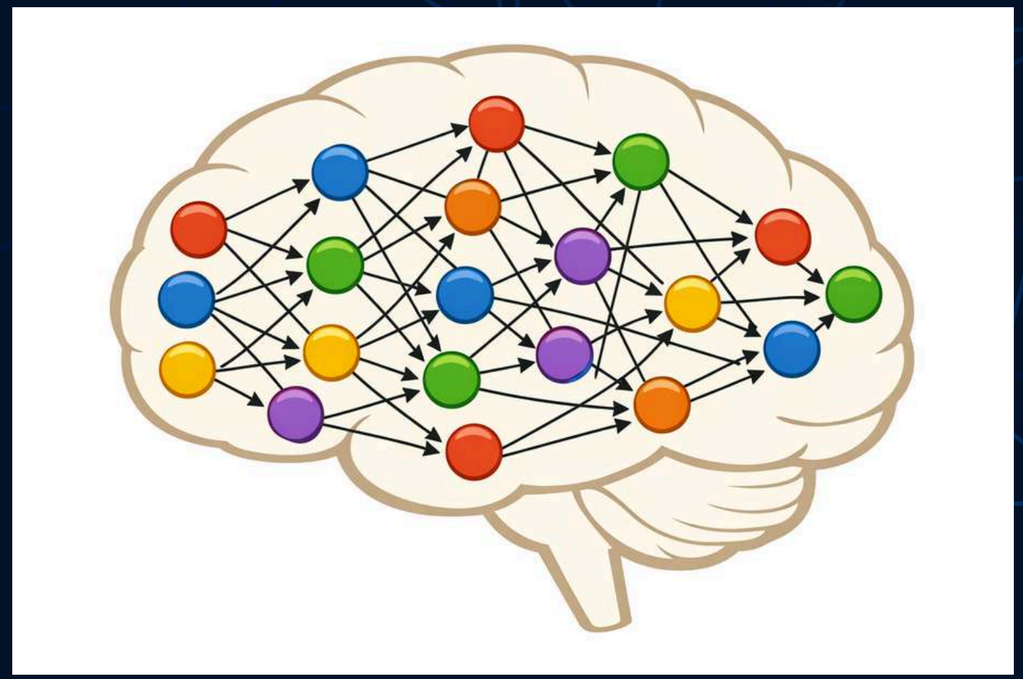
Turing’s Enigma

Team Members:-
Archi
Shubh
Bhawansh
Neha
Shailendra
Samridhi
Moksh
Hariish

Artificial Neural Network (ANN)

What it is:

Inspired by the human brain, ANN is used in machine learning to help computers recognize patterns, make predictions, and learn from data.

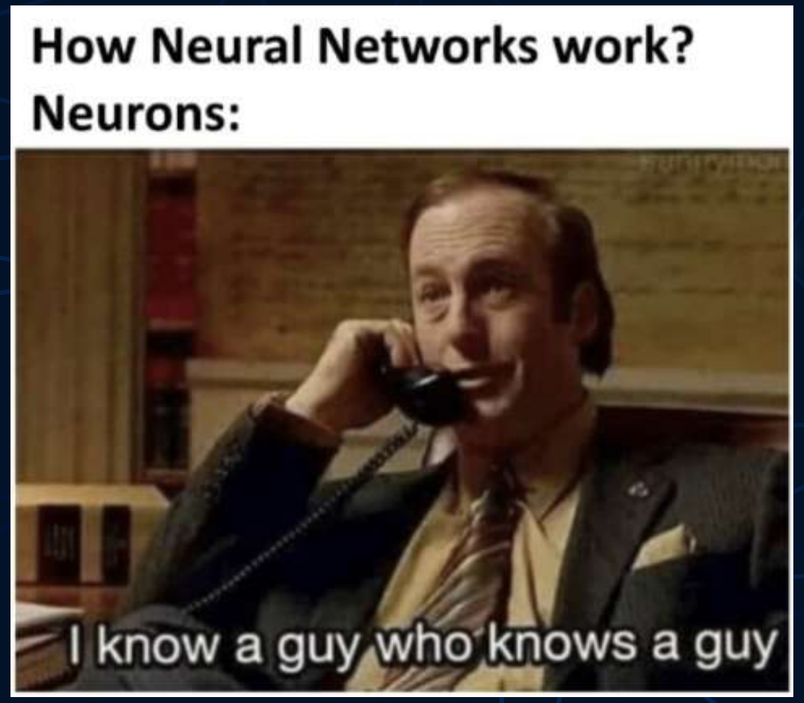


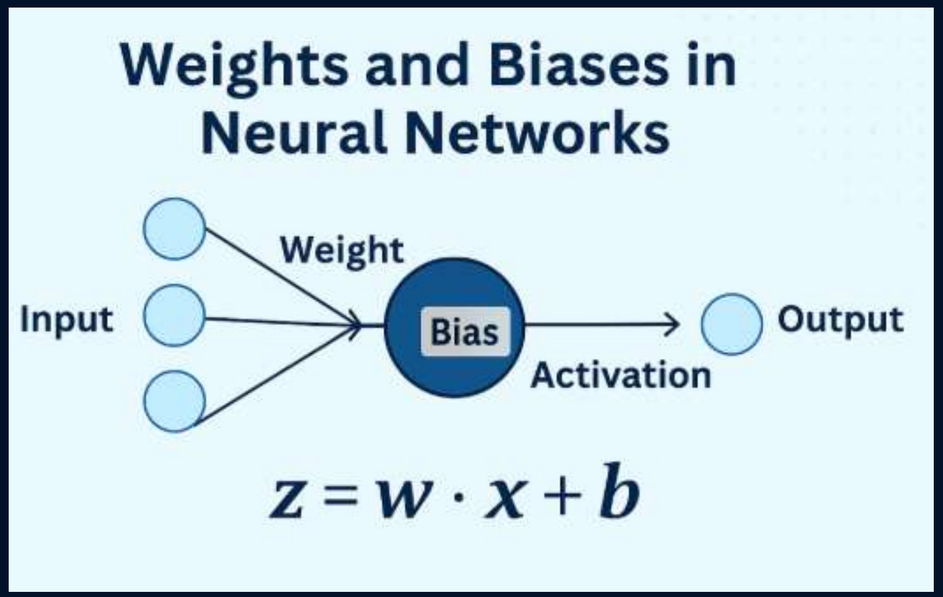
Where it’s used:

- 📷 Image Generation
- 🌍 Language Translation
- 🎤 Speech Recognition

- Some data is given to the network. This is the input.
- The input is then multiplied by weights. Weights determine the importance of the input feature.
- A bias is added, allowing the model to be more flexible.
- This sum is then passed through an activation function, which gives us the output. This function helps the network learn complex patterns. Ex: sigmoid, tanh, and ReLU.

This process is called **Forward Propogation**





This output is then compared with the correct(target) output. Error is calculated using the loss function

Backpropagation

Backpropagation is a machine learning technique essential to the optimization of artificial neural networks.

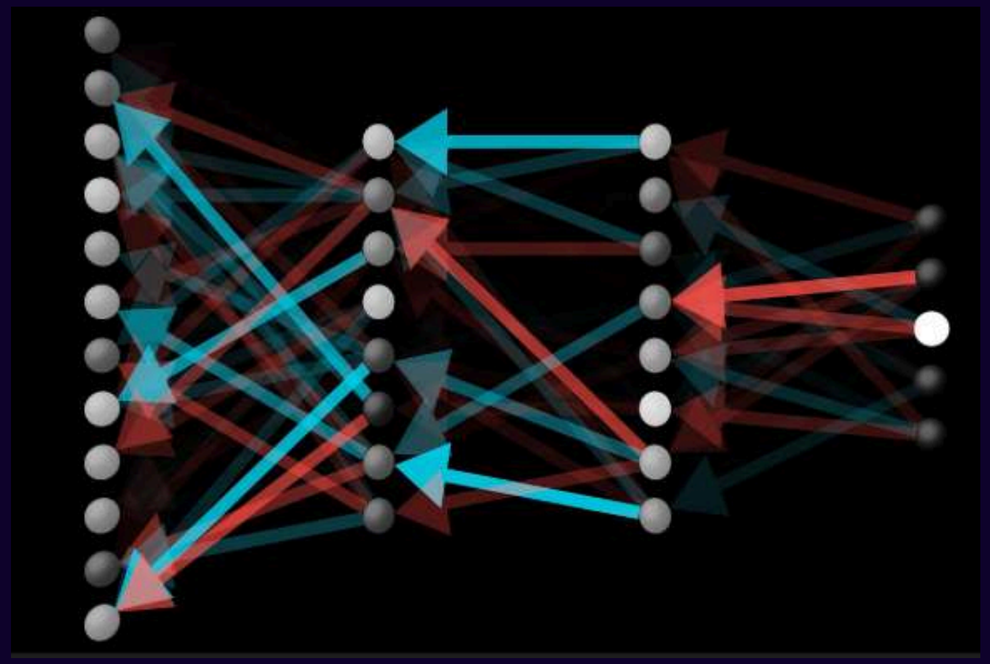
Backpropagation is an algorithm for determining how a single training example would like to nudge the weights and biases not just in terms they should increase or decrease but in terms of what relative proportions to those changes cause the most rapid decrease to the cost

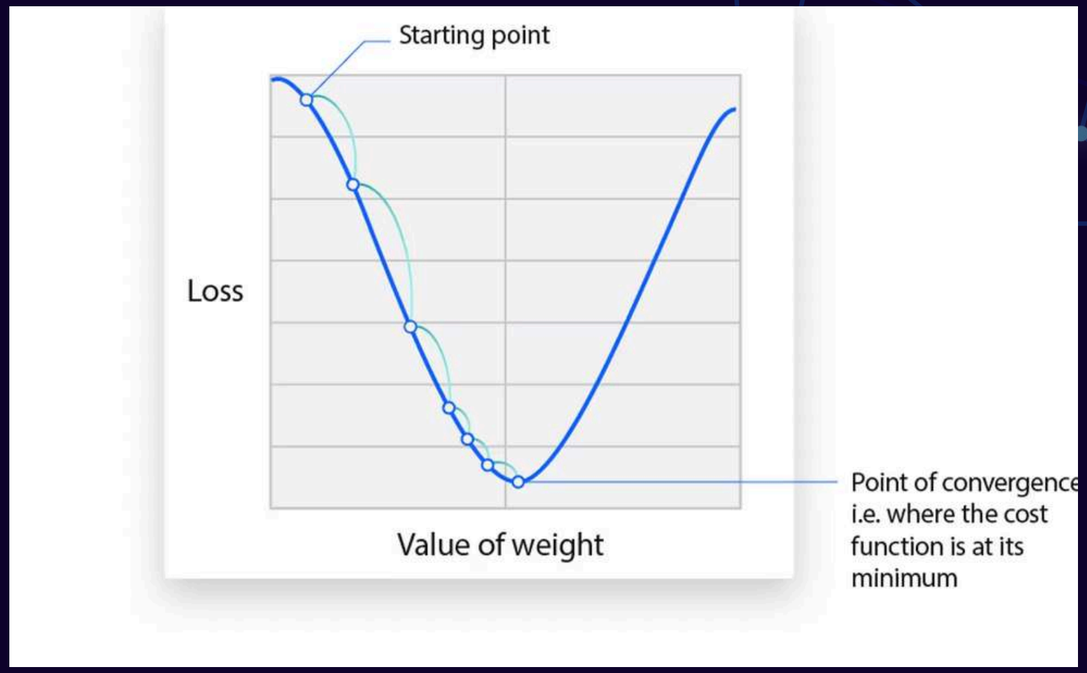
In a neural network, cost measures how wrong predictions are compared to true labels. It's a single number minimized during training to improve accuracy.

Pretty Confusing !!!

In simpler words the process of back propagation involves altering weights and biases of a neural network which helps increasing the accuracy of the model

Weights and biases are updated during training based on the prediction error.
After calculating the loss, backpropagation computes how each weight and bias contributed to the error.
Using gradient descent and a learning rate, they are adjusted to reduce the loss.
This process repeats over multiple iterations, allowing the network to learn better patterns.





Convolutional Neural Network (CNN)

Technique helps in pattern recognition within images.

Overcome the computational capabilities of ANN.

1D VS 2D/3D

SPATIAL INFORMATION LOSS IS HIGH

EVERY INPUT CONNECTED TO OUTPUT NEURON

LEADS TO OVERFITTING

SEARCH FOR IMAGE SPECIFIC VS EVERYWHERE

LOCATION PROBLEM

Used MNIST database
(Handwritten numbers database)

Applied 3 layers : Convolutional layer, pooling layers and fully connected layers.

- Convolutional Neural Networks (CNNs) are a specialized class of artificial neural networks designed for processing data with a grid-like topology, such as images.

- CNNs use a mathematical operation called "convolution" in at least one of their layers instead of general matrix multiplication.

- The primary goal of a CNN is to learn spatial hierarchies of features automatically and adaptively, from low-level patterns to high-level concepts.

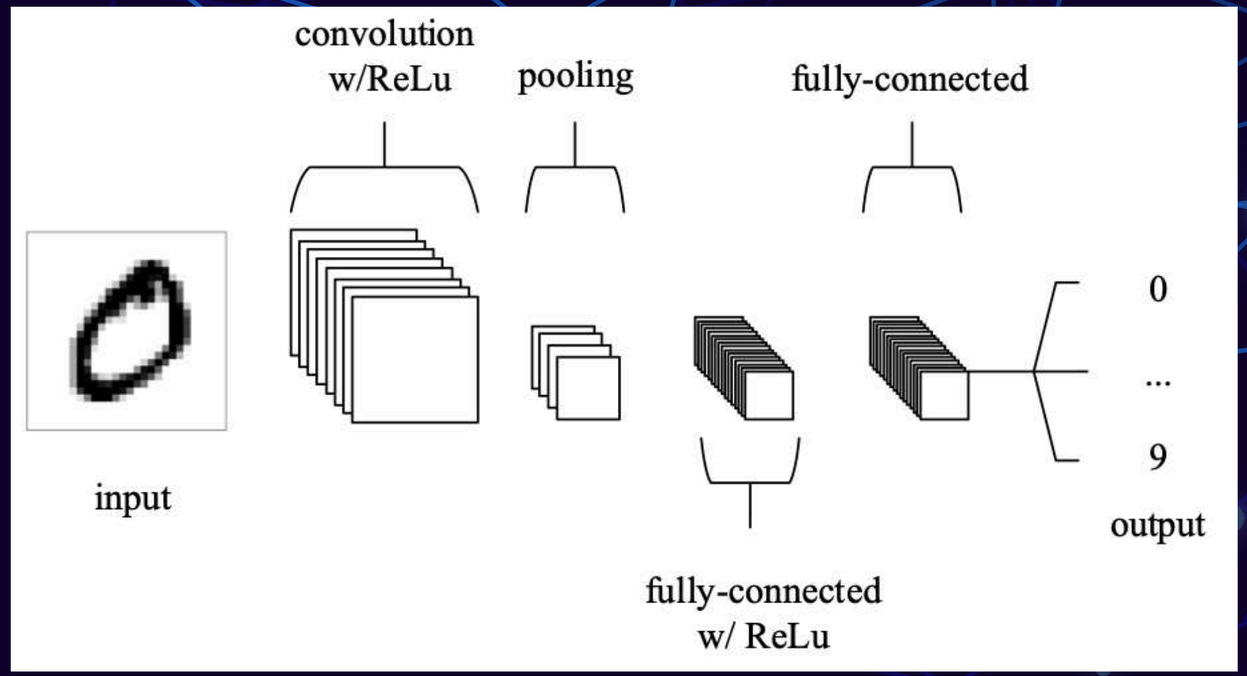
- They are particularly effective at overcoming the issue of "overfitting," where a model learns training data too well but fails to generalize to new, unseen data

Type of layers

Convolutional Layer → The fundamental building block that performs feature extraction by applying various filters to the input data

Pooling Layer → Used to reduce the spatial dimensions (width and height) of the input volume for the next convolutional layer.

Fully-Connected Layer → Usually located at the end of the network, these layers perform the final classification based on the features extracted by the previous layers



3*3 kernels (which represent features such as colors and edges)

ReLu : Rectified linear unit(ReLu)
Activations as Sigmoid

MaxPooling : No weights

Softmax : Probability function

Operation	Input Shape	Output Shape	What happens?
conv1	[1, 28, 28]	[8, 28, 28]	Look for 8 edge types
pool	[8, 28, 28]	[8, 14, 14]	Reduce size by half
conv2	[8, 14, 14]	[16, 14, 14]	Look for 16 shapes
pool	[16, 14, 14]	[16, 7, 7]	Reduce size by half
transition	[16, 7, 7]	[784]	Stretch into a line
fc1	[784]	[64]	Compress into 64 concepts
fc2	[64]	[10]	Final vote for digits 0-9

Hyperparmeters

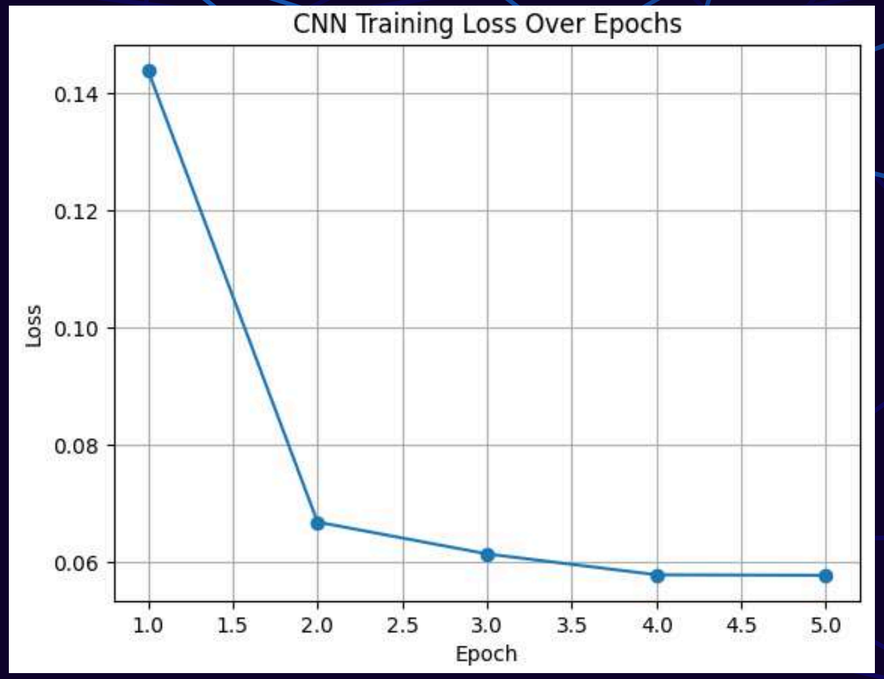
- Number of filters(8,16,32)
- Stride (1,2)
- Padding (0,1)

Others

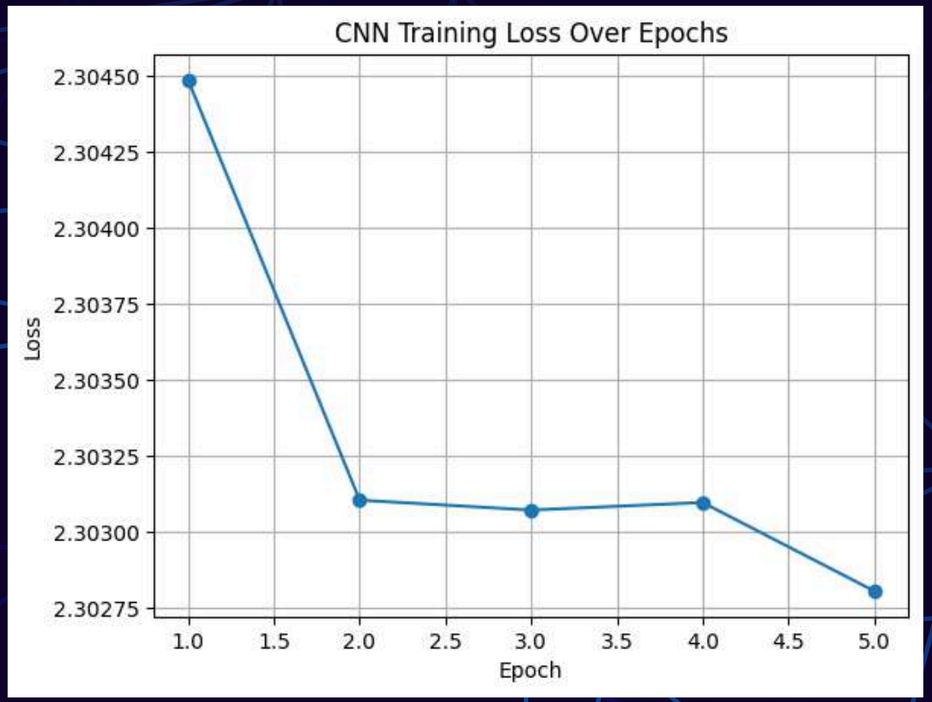
- Epochs (5, 10)
- Learning rate (0.01,0.02)

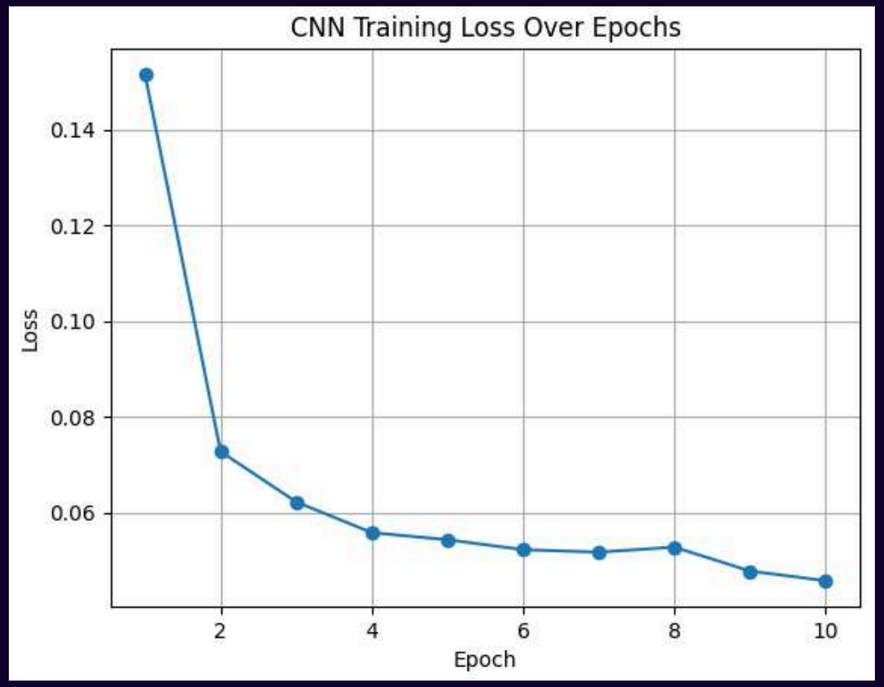
CNN Code

https://colab.research.google.com/drive/1yNZn2m8KnAsvpKPfsi0Dxk6bKfVh09P5?usp=sharing

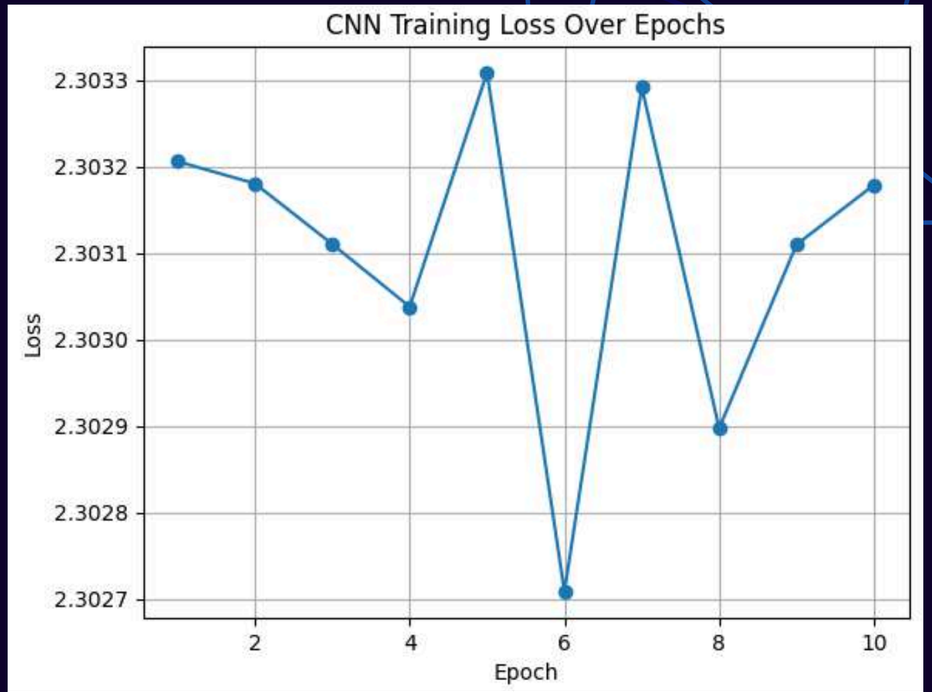


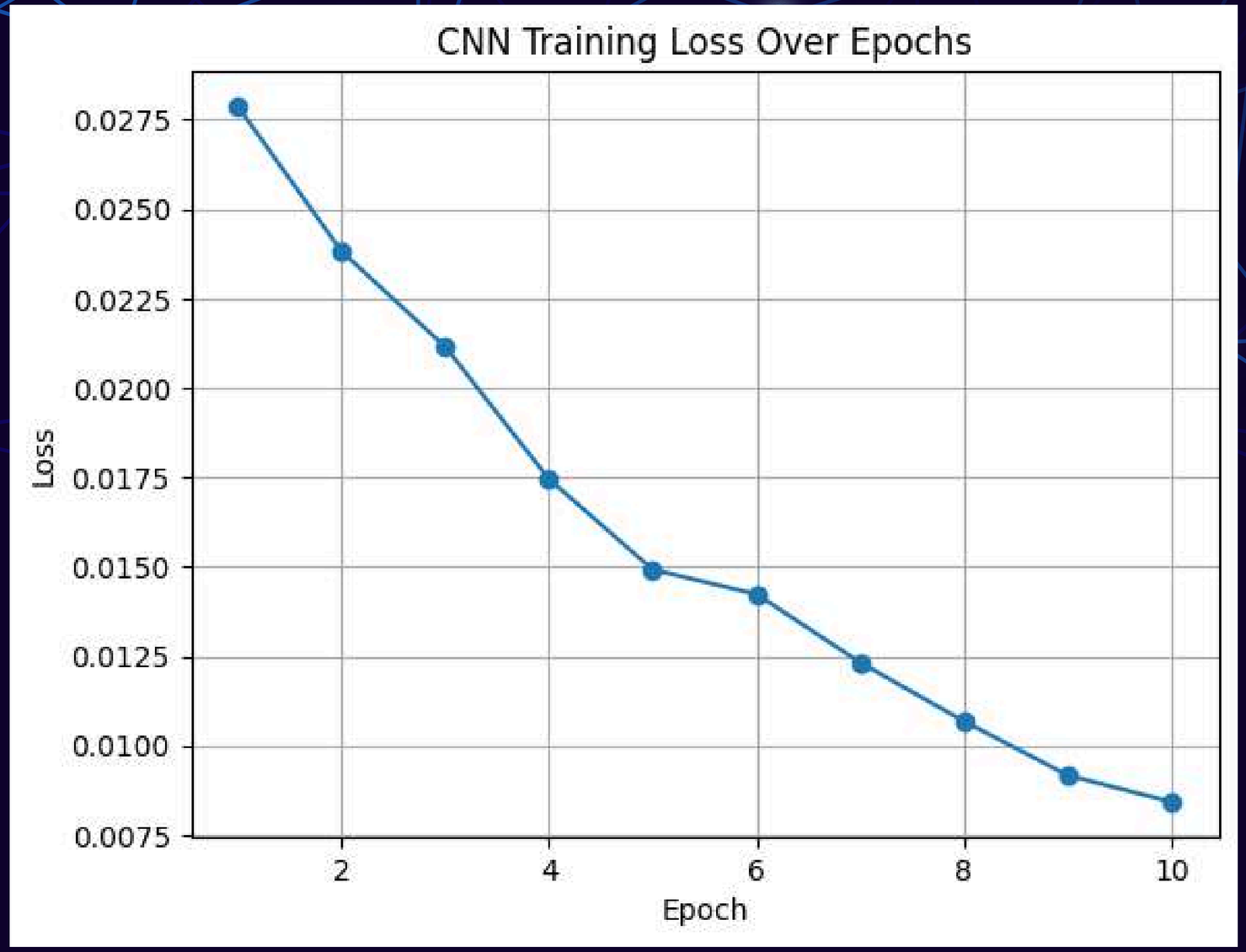
Learning rate : 0.01





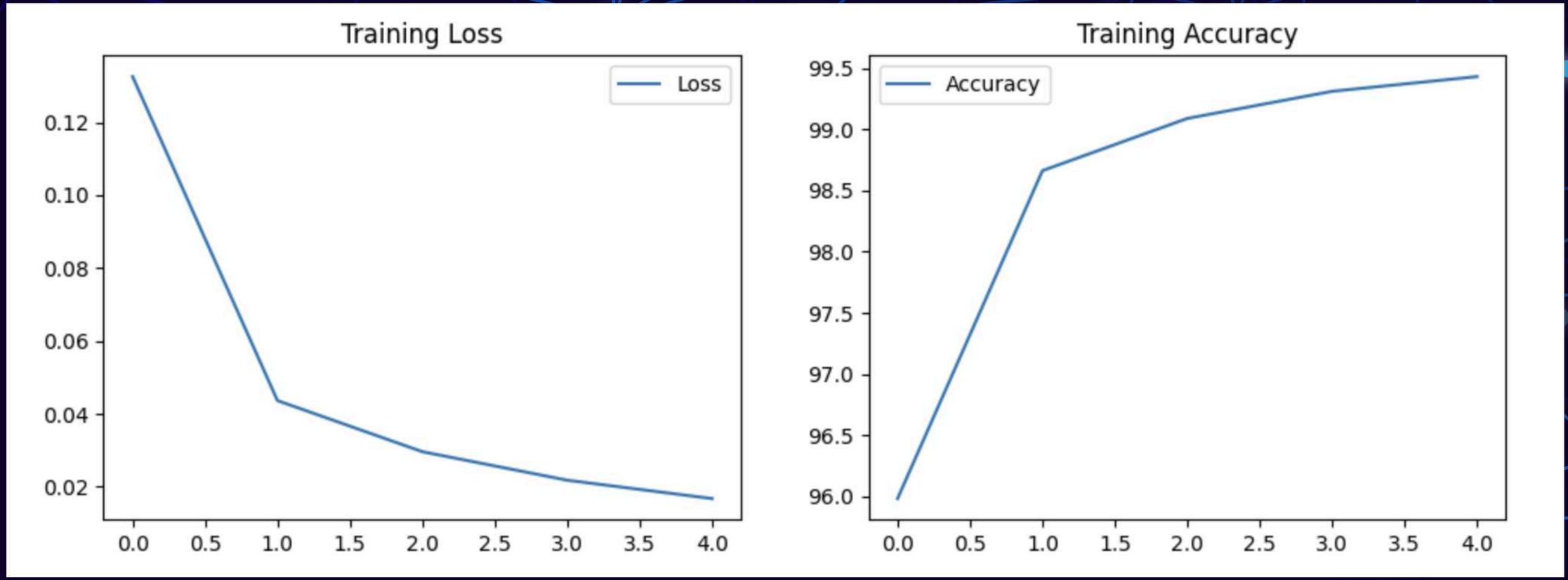
Learning rate : 0.02





Default learning rate : 0.001

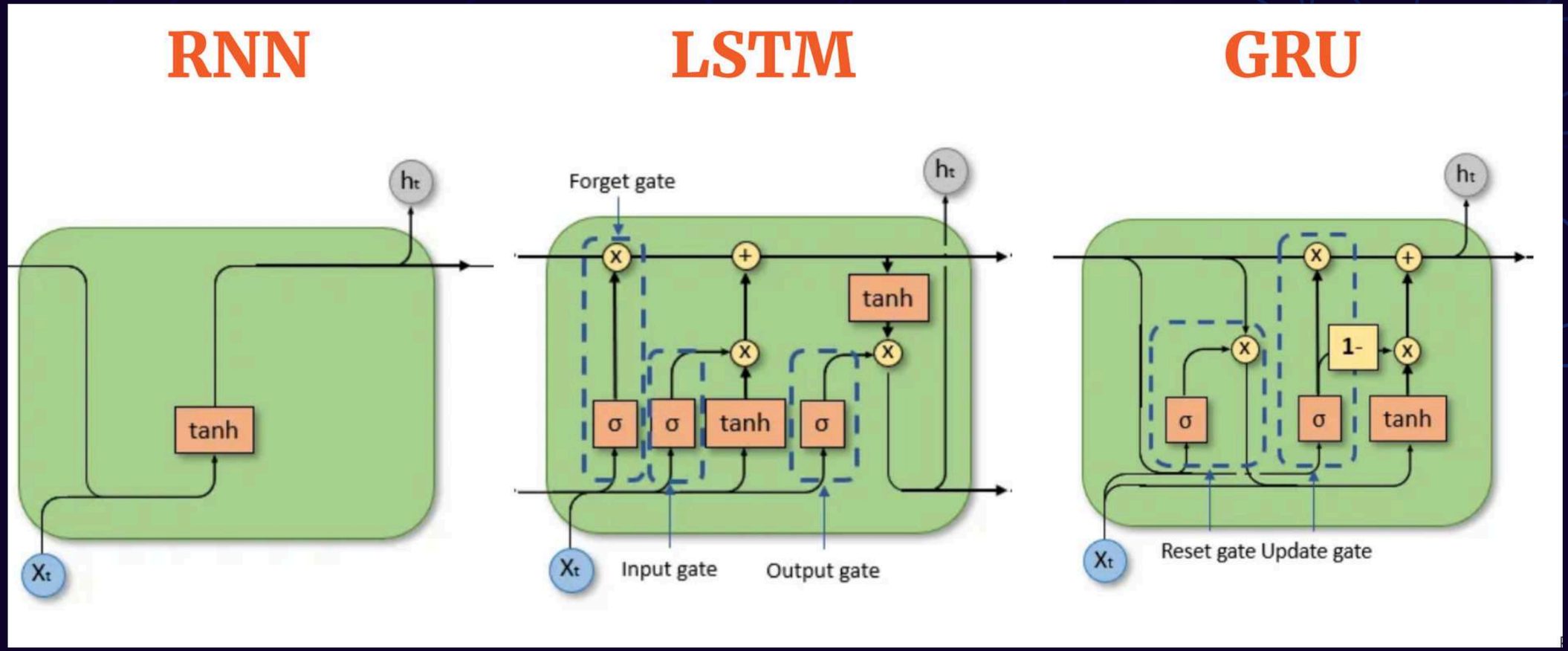
Accuracy : 98.79%

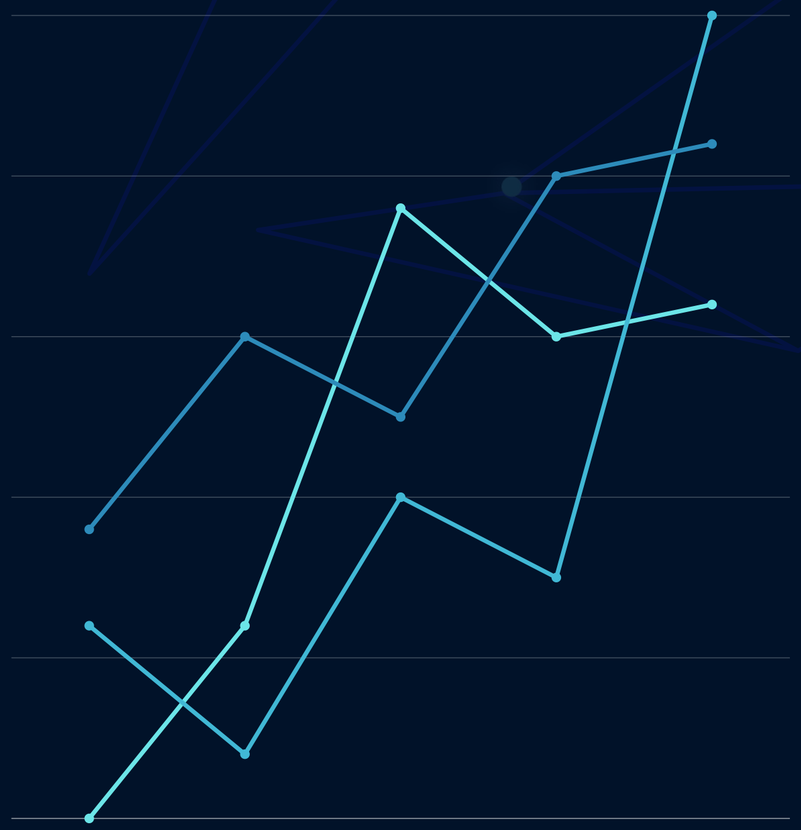


Training Data with batch size 1000

SEQUENTIAL MODELS

RNNs, LSTMs and GRUs





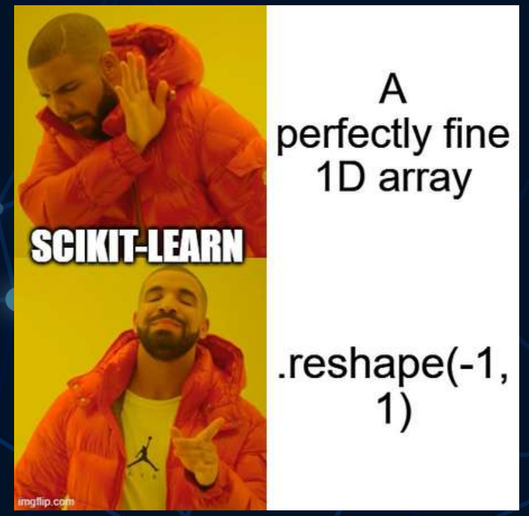
Comparative Analysis of Sequential Models for Tesla Stock Price Prediction

WHY SEQUENTIAL MODELS?

We used Sequential Models because they have built-in memory. They don't just look at today's price; they look at the patterns of the last 60 days to understand where the trend is going. By using gates to filter out important long-term trends and ignore daily random noise, these models are significantly better at handling the extreme fluctuations of the stock market.

Data collection and Preprocessing

Why are we reshaping?



DATA COLLECTION

We used the yfinance API to pull 6 years of daily historical data for Tesla (TSLA). We chose the 'Close' price as our primary target because it represents the market's overall sentiment for the day.

Why are we scaling?

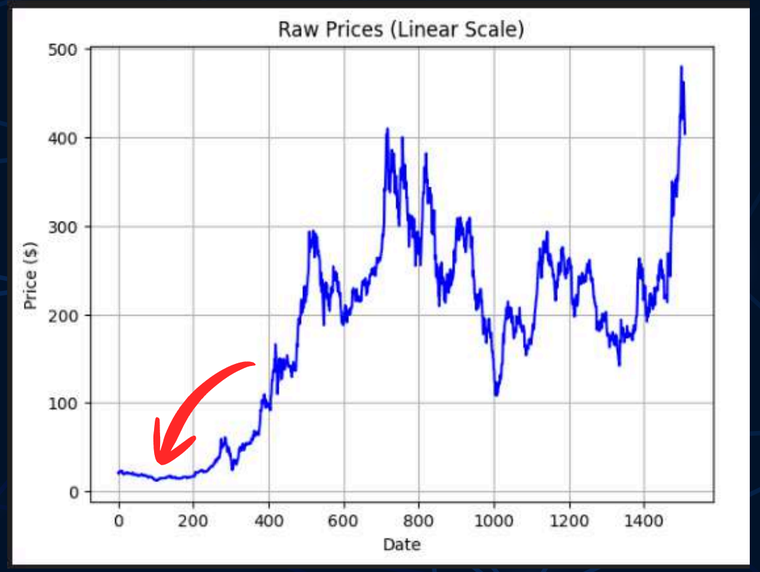
PREPROCESSING

It can be divided into three steps.

- Reshaping
- Scaling (Normalization)
- Sequencing

THE CHALLENGE: Handling exponential growth

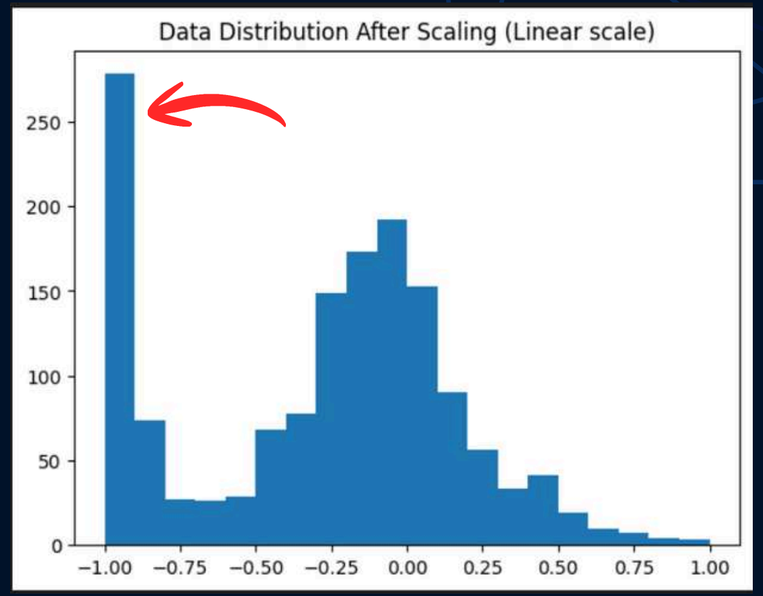
It was at the stage of scaling that we noticed a major problem: Tesla's growth was so explosive that standard scaling was hiding all the early historical patterns from the 2019-2020 time period.



The distribution plot (histogram) confirms that standard scaling crushes our historical signal into a single 'peak,' leaving the model with no meaningful variance to learn from.

We were facing two major issues when only using the MinMaxScaler:

- Vanishing Gradient problem: Almost all of the data in the 2019-2020 range was converted into values near -1, confusing the model into thinking that there was no activity during that time, which is false.
- The model does not understand the workings of the stock market properly. It thinks that every 10-dollar drop is the same, which is not the case when it comes to stock prices.

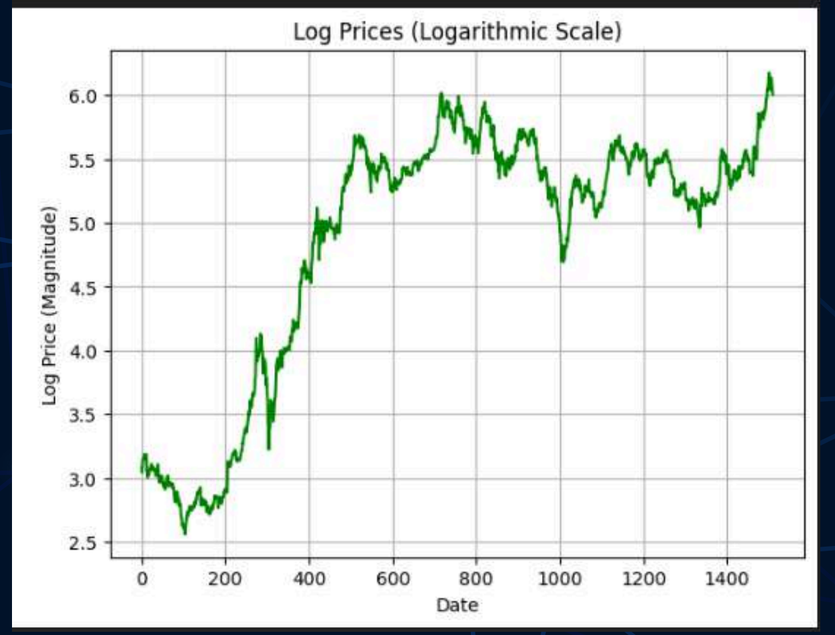


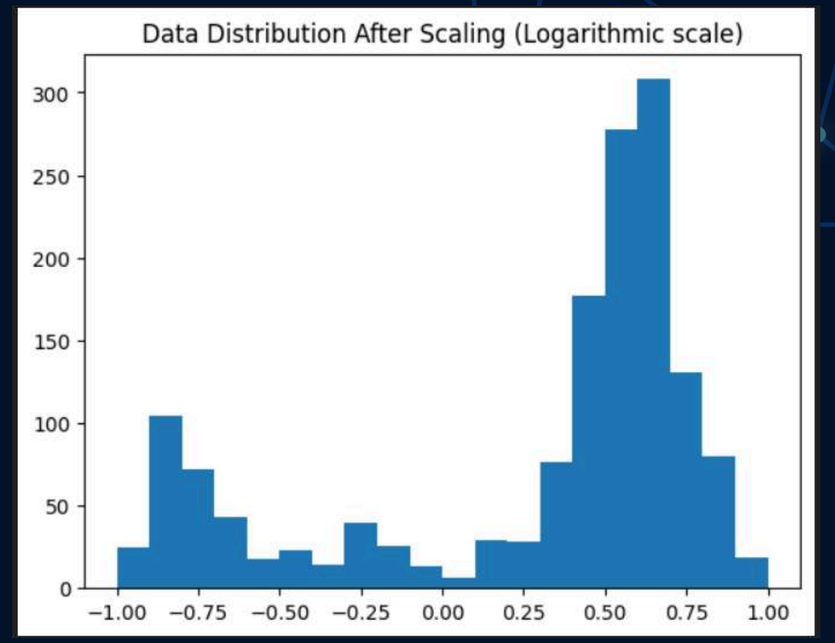
THE SOLUTION: Log scaling

Instead of looking at raw dollar amounts, we applied a Logarithmic Transformation ln(1+x).

How did this help us?

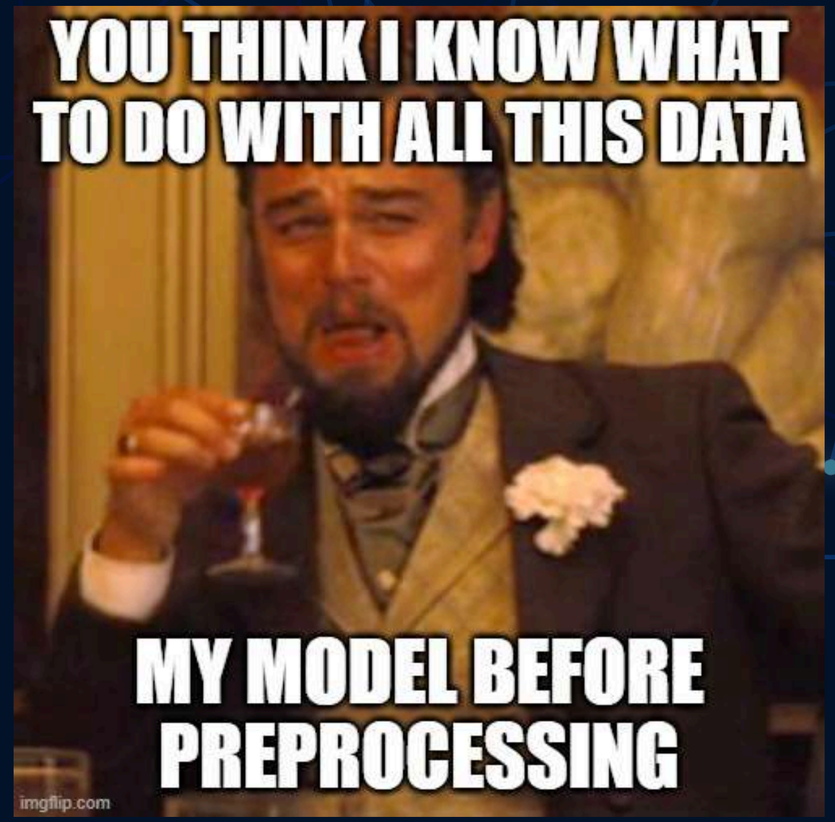
- We were able to convert the flat line in the 2019-2020 time period into meaningful activity from which the model can learn.
- The gradient is no longer zero, so the vanishing gradient problem is solved.
- Because logarithms focus more on ratios rather than pure differences in values, the model can now differentiate between a 10-dollar drop in 2019, which is a huge crash in the market, and a 10-dollar drop in 2024, which is just a normal dip.





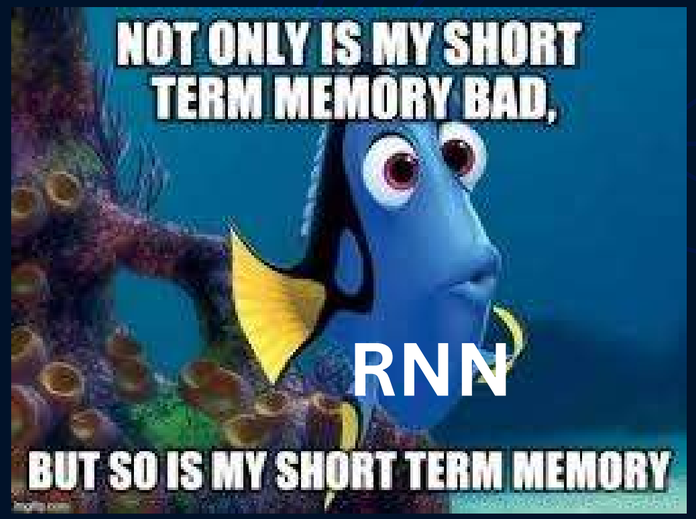
PREPROCESSING - Sequencing

- By using the 60-Day Sliding Window, we transformed our data into a question-and-answer format. The model looks at a window of 60 consecutive days (the question) to predict what happens on the 61st day (the answer).
- As the window advances one day at a time, it creates thousands of overlapping training examples, enabling the model to learn about various market cycles.
- This step converts our 1D list into a 3D Tensor of the format (Samples, 60, 1), ready to use the PyTorch libraries.



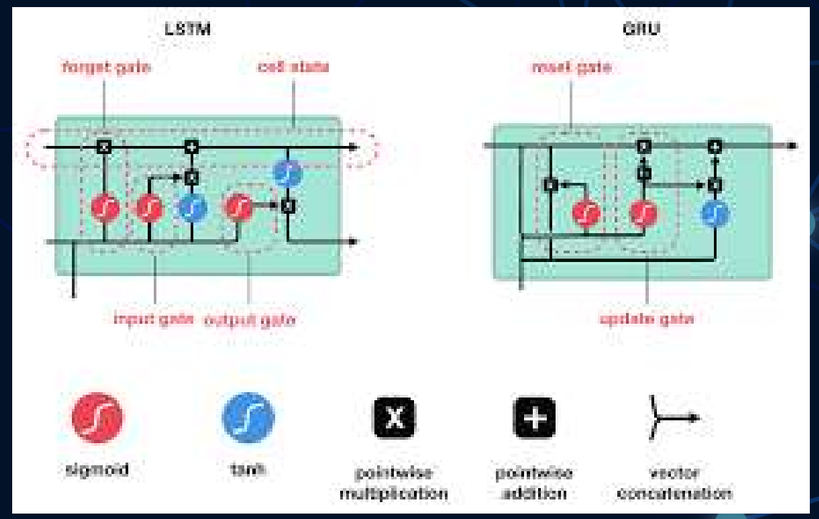
MODEL ARCHITECTURE

WHY NOT RNN?? 🤔🤔



We skipped standard RNNs because they suffer from "Goldfish Memory" (Vanishing Gradients)—they forget the start of the 60-day window by the time they reach the end.

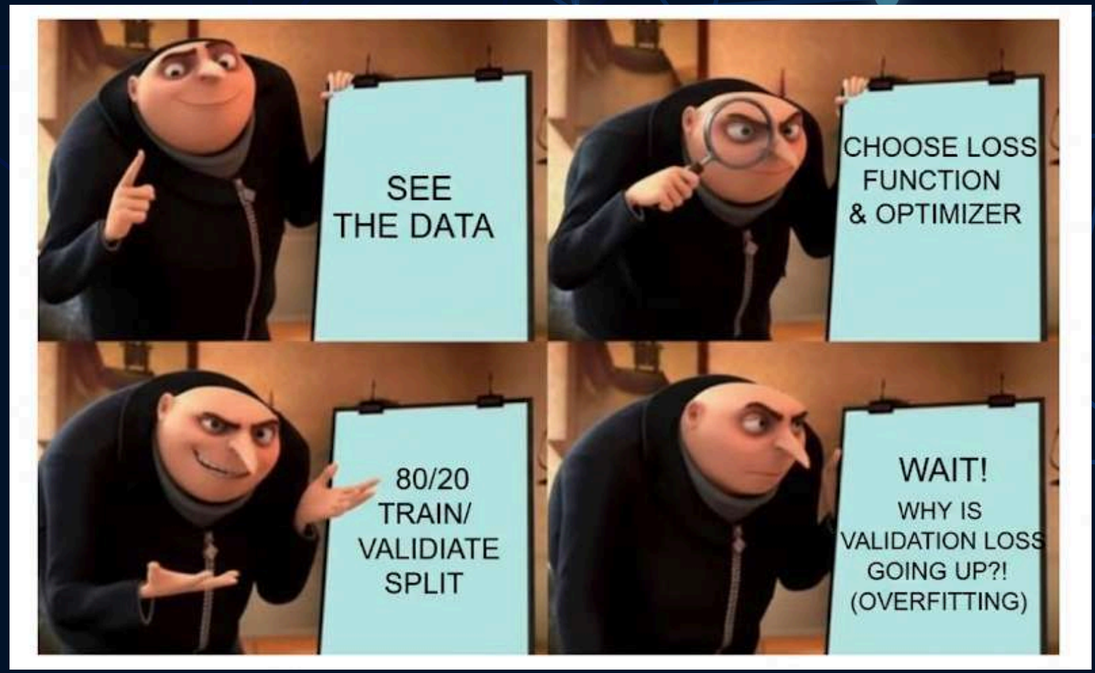
LSTM VS GRU



- LSTM: Uses three internal Gates (Forget, Input, Output) to surgically decide what history to keep and what to toss.
- GRU: A less complex version of the LSTM with just two gates instead of three. The input and forget gates are merged into one update gate.

TRAINING AND OPTIMIZATION

- Loss Function (MSE): We used Mean Squared Error. This punishes the model heavily for big misses, which is vital when you're dealing with high-value stock prices.
- The Optimizer (Adam): We chose Adam because it’s the most reliable for deep learning. It automatically manages the learning speed so the model doesn't get stuck or overshoot the solution.
- The 80/20 Rule: We kept 20% of the data completely hidden during training so that it can be used for testing to make sure that the model didn’t just memorize the dataset we provided (overfitting).



CONCLUSION - I

LSTM

Training time: 59.600841760635376

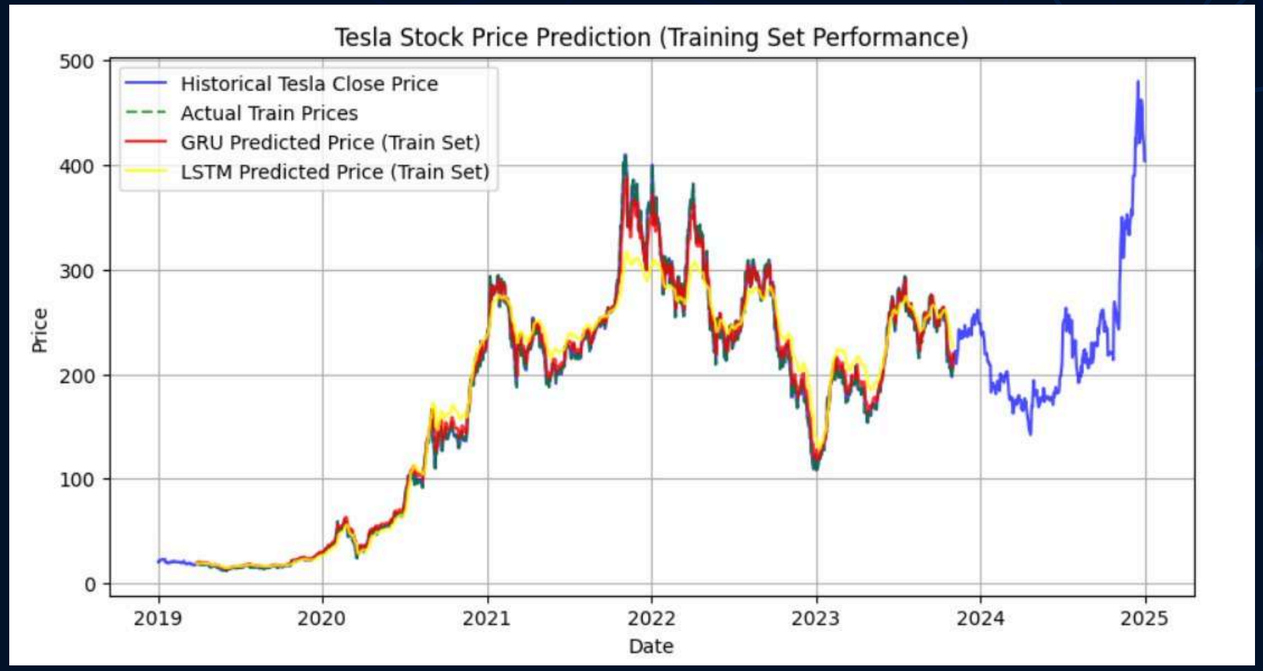
GRU

Training time: 42.862799406051636

WHEN GIVEN A SMALL DATASET LIKE THIS ONE, GRUs PERFORM BETTER AND FASTER THAN LSTMs.

The training times mentioned above are for 50 epochs.

CONCLUSION - II

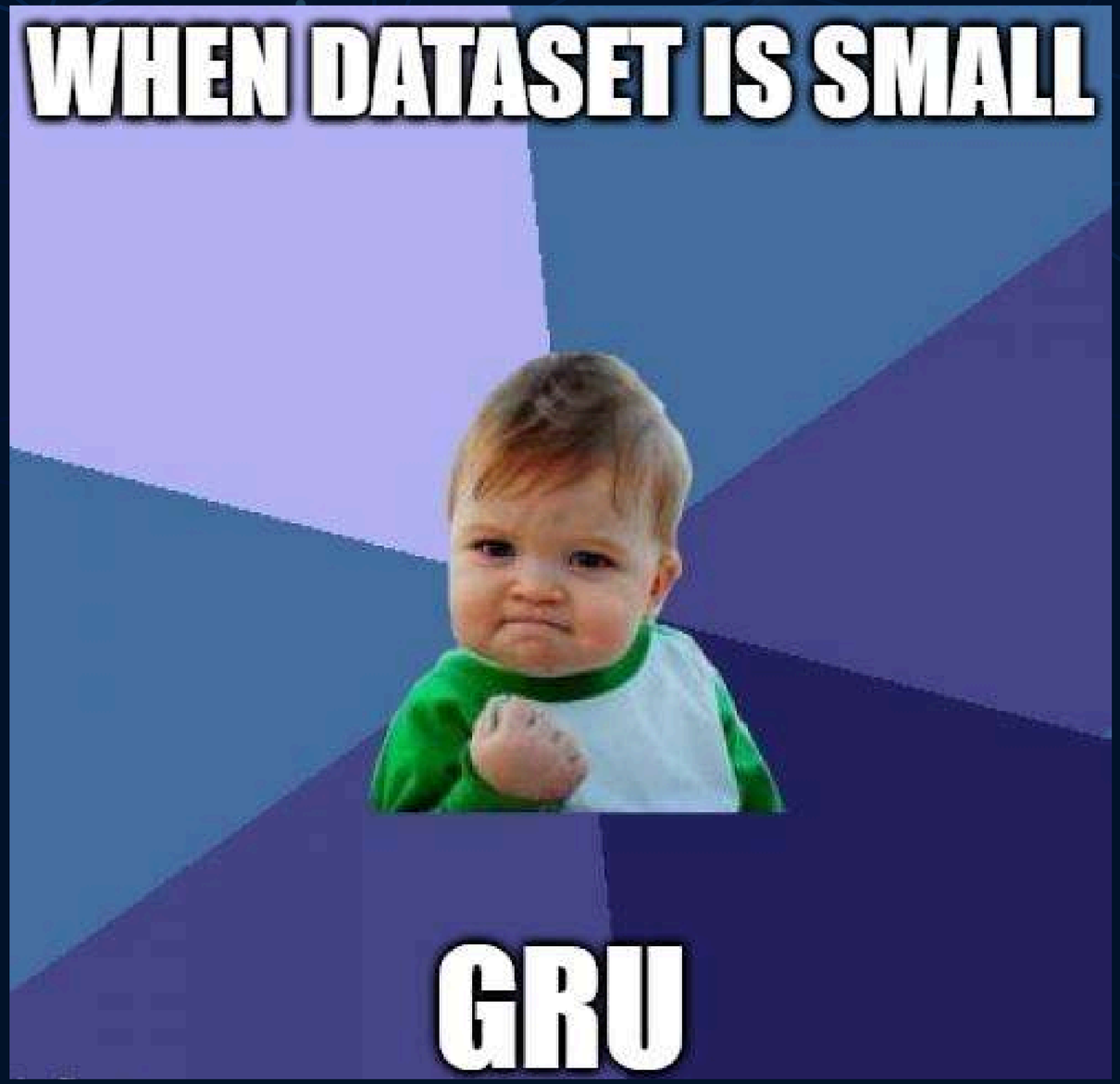


Number of epochs = 50

Number of epochs = 100

GRU WAS ALREADY PERFORMING PRETTY WELL IN JUST 50 EPOCHS, SO CHANGING IT TO 100 HAD VERY LITTLE EFFECT ON THE GRU PREDICTIONS. HOWEVER, LSTM PREDICTIONS IMPROVED SIGNIFICANTLY WHEN WE INCREASED THE NUMBER OF EPOCHS.

CLAIM: If we had implemented early stopping GRU would have stopped way earlier than LSTM.


WHEN DATASET IS SMALL
GRU

Transformers and Architecture

Core Technology:

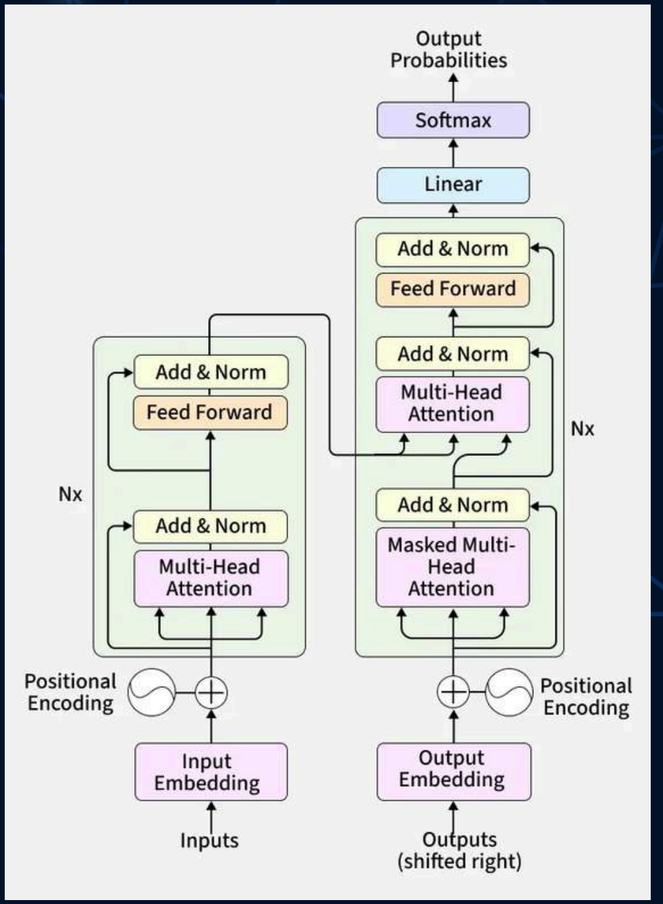
Parallel Processing: Unlike previous models (RNNs/LSTMs) that read data sequentially (word-by-word), Transformers process the entire sequence of data simultaneously.

Self-Attention Mechanism: The "brain" of the model. It calculates the relevance of every word in a sentence to every other word, enabling the model to understand complex context and long-range dependencies .

Positional Encodings: Since the model reads everything at once, it uses mathematical vectors to tag the order of words so the sequence isn't lost.

Architecture Breakdown:

Computer Vision: Vision Transformers (ViTs) used for object detection and image segmentation in medical imaging and autonomous driving.



Large Language Models(LLMs)

How They Work:

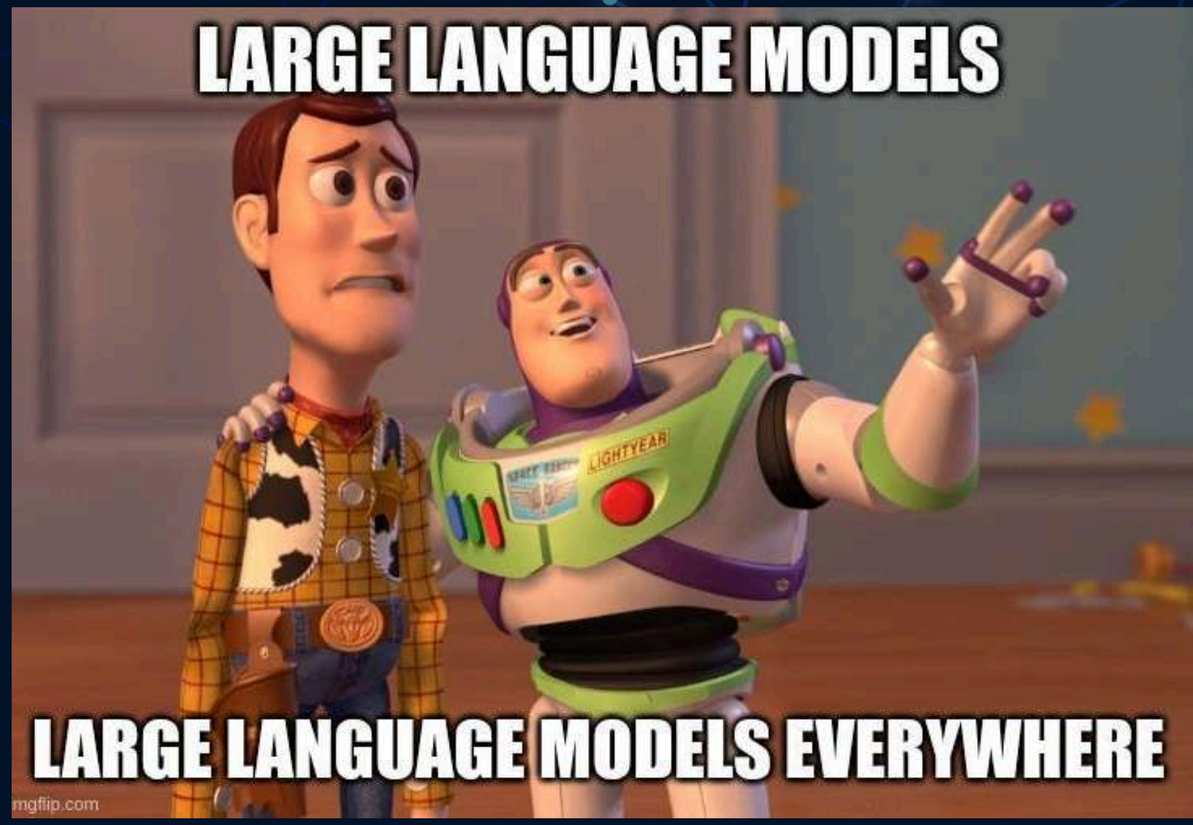
Generative Pre-training: LLMs are trained on massive unlabeled datasets (the internet) to learn language patterns, followed by "Fine-Tuning" on specific data for specialized tasks.

Probabilistic Prediction: They function as advanced prediction engines, calculating the statistical probability of the next "token" (word/part of word) to generate coherent text.

Types of LLMs:

Decoder-Only (e.g., GPT): Best for creative text and code generation.

Encoder-Only (e.g., BERT): Best for analyzing sentiment and classifying documents.



Artificial General Inteligence(AGI)

Technological Advancements:
Multimodality: New models (like GPT-4o) can process text, audio, video, and images simultaneously in real-time, creating a more holistic "human-like" perception.

Chain-of-Thought Reasoning: Advanced techniques where models break down complex logic problems into steps, improving their ability to "reason" rather than just predict.
The Path to AGI (Artificial General Intelligence):
Definition: AI that possesses the ability to understand, learn, and apply knowledge across a wide variety of tasks, indistinguishable from a human.
e.


"BUT...I THOUGHT MY PRESENTATION WAS OK.
YESS. BUT WE STILL HAVE TO ASK
THE MOST DIFFICULT QUESTION POSSIBLE.
Please be easy on us.
imgflip.com