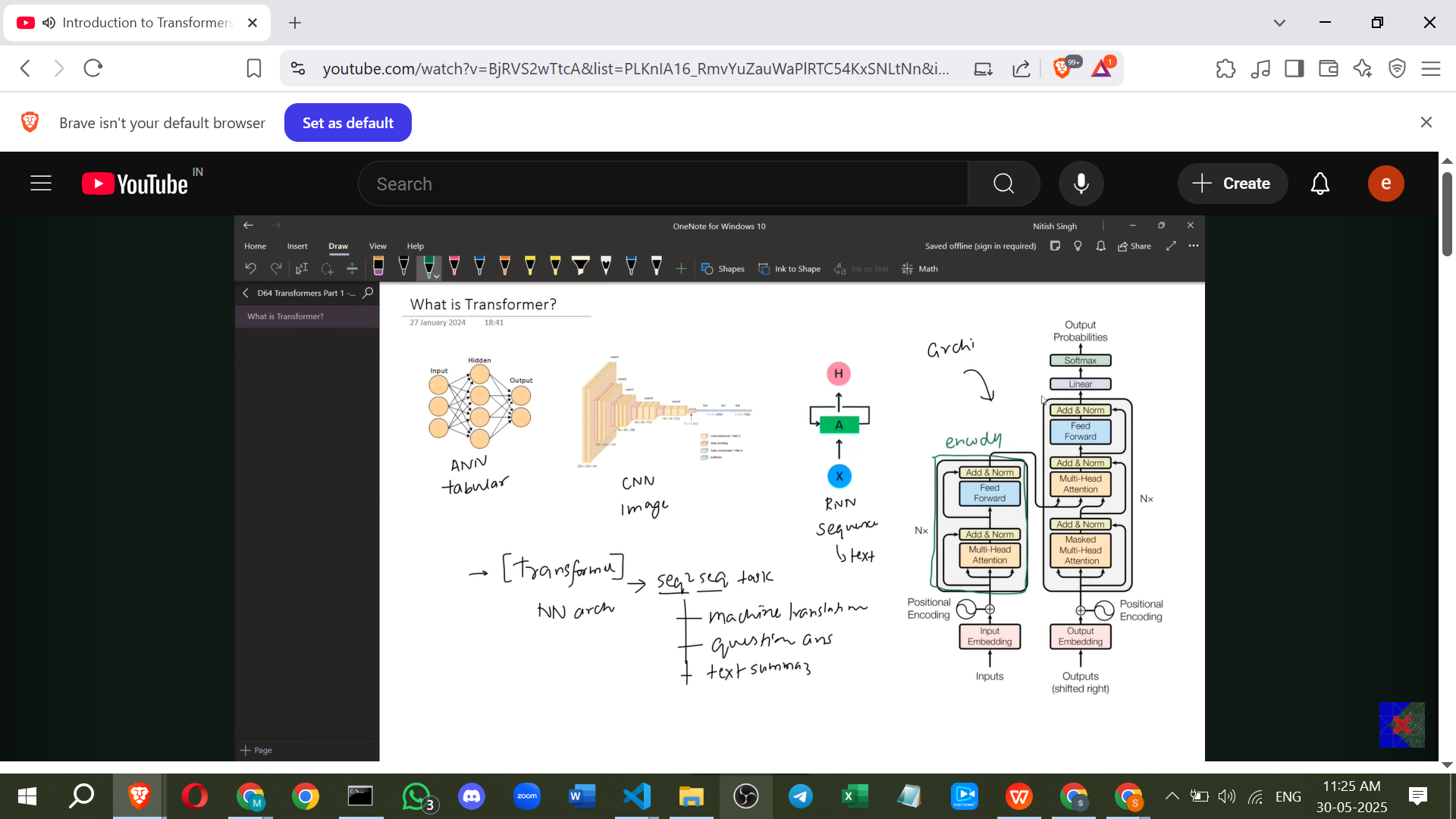
# **Class 1. Introduction to Transformers | Transformers Part 1**

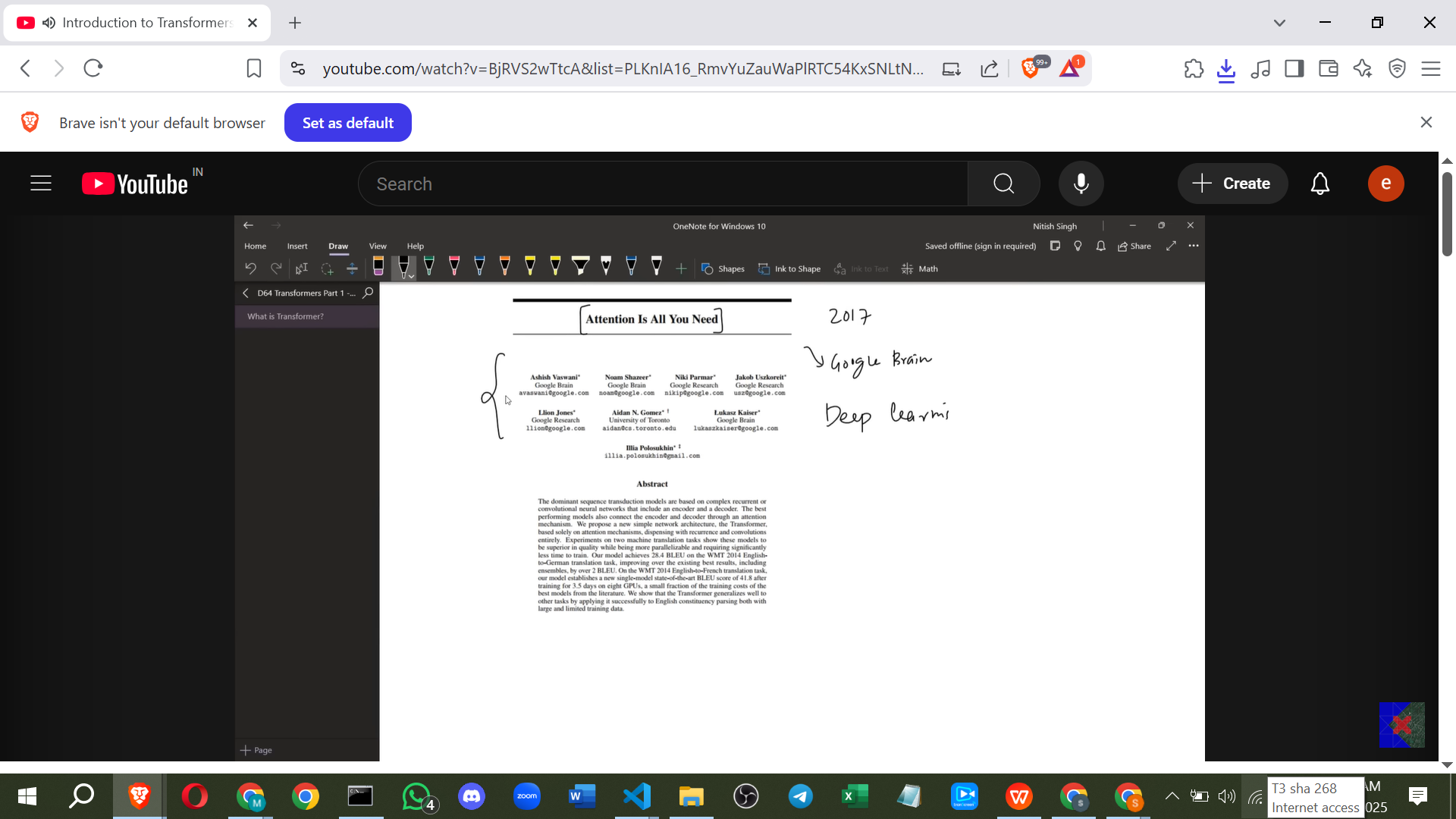


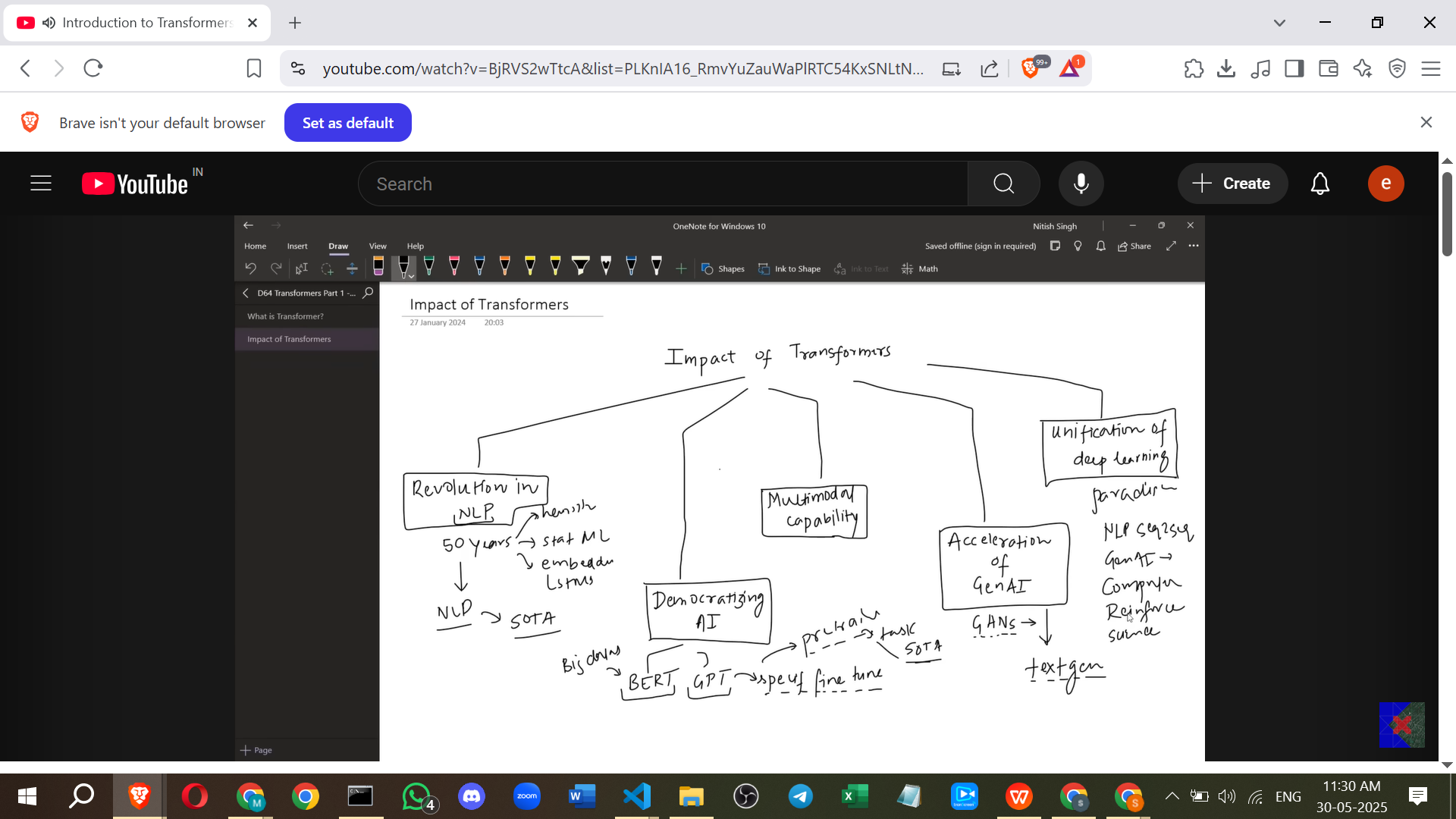
**Introduction to Transformers**

* It provides an **overview of Transformers**, their architecture, the factors behind their creation, advantages, disadvantages, and future prospects.
* The goal is to offer a solid overview to help understand more detailed discussions later.

**What are Transformers?**

* Transformers are a **neural network architecture**.
* Like other neural network architectures such as ANNs (for tabular data), CNNs (for image data), and RNNs (for sequential data like text), Transformers are designed for a specific type of data.
* Transformers were built to handle **sequence-to-sequence tasks**.





**Sequence-to-Sequence Tasks**

* A sequence-to-sequence task involves sequential data as both the **input and output**.
* **Examples** mentioned in the transcript include:
  + **Machine Translation**: Converting a sentence from one language to another.
  + **Question Answering Systems**: Taking a question (sequence) and generating an answer (sequence).
  + **Text Summarisation**: Generating a summary (sequence) from a given large text (sequence).
* The name "Transformer" comes from their function of **transforming one sequence into another**.

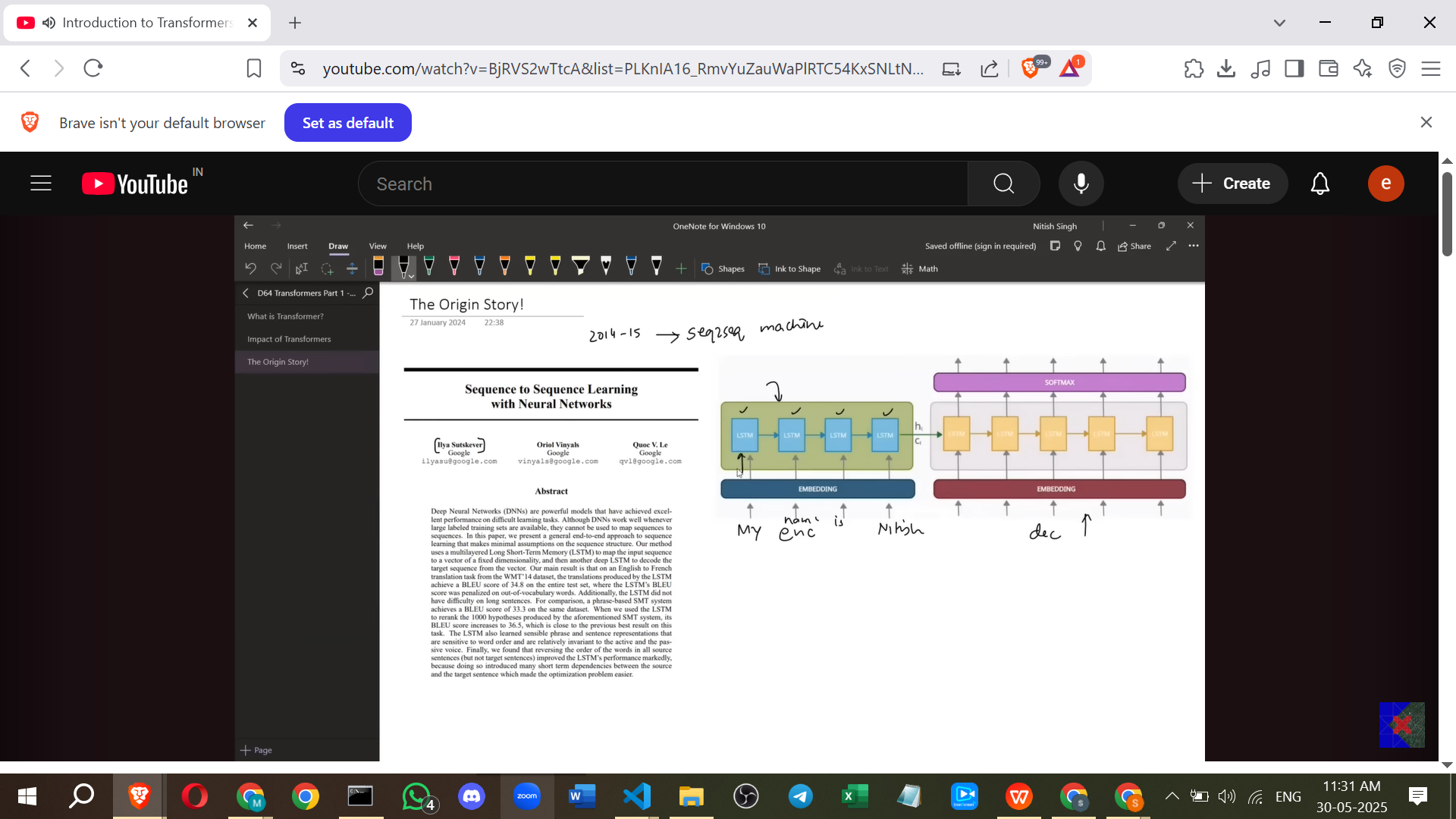
**Transformer Architecture**

* The architecture consists of an **encoder** and a **decoder**, similar to other sequence-to-sequence models.
* However, unlike previous architectures (which used LSTMs), Transformers **do not use LSTMs**.
* Instead, they use a form of **attention called Self-Attention**.
* Self-Attention allows the encoder to **process all words in a sentence simultaneously** (parallel processing).
* This parallel processing makes the architecture **highly scalable** and allows for very large datasets to be handled and trained efficiently.
* The complex architecture also involves other components like Residual Connections, Layer Normalisation, and Feed Forward Neural Networks.

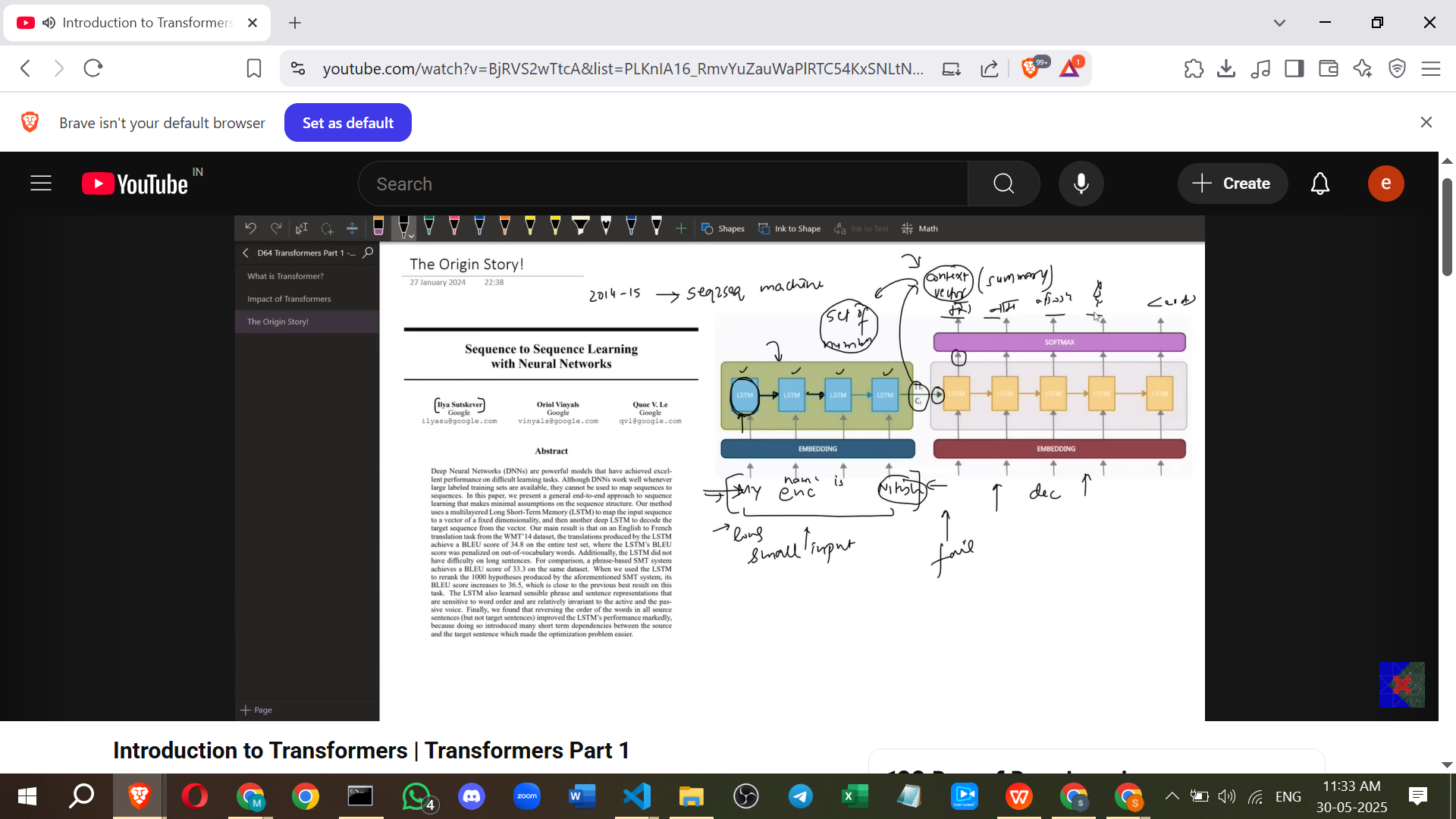
**Origin Story**

* The creation of Transformers was driven by the limitations of previous architectures.
* **Chapter 1: Sequence-to-Sequence with LSTMs (circa 2014-2015)**

**[[1409.3215] Sequence to Sequence Learning with Neural Networks](https://arxiv.org/abs/1409.3215)**

****

* + Based on the paper "Sequence to Sequence Learning with Neural Networks".
  + Proposed an **encoder-decoder architecture using LSTMs** for sequence-to-sequence tasks like machine translation.
  + The encoder processes the input sentence step-by-step and produces a **single context vector** representing the entire sentence.
  + The decoder then uses this context vector to generate the output sequence word by word.
  + **Problem**: This architecture **fails with long sentences** (e.g., 30+ words) because the single context vector cannot retain all the necessary information from a large input, leading to poor translation quality.

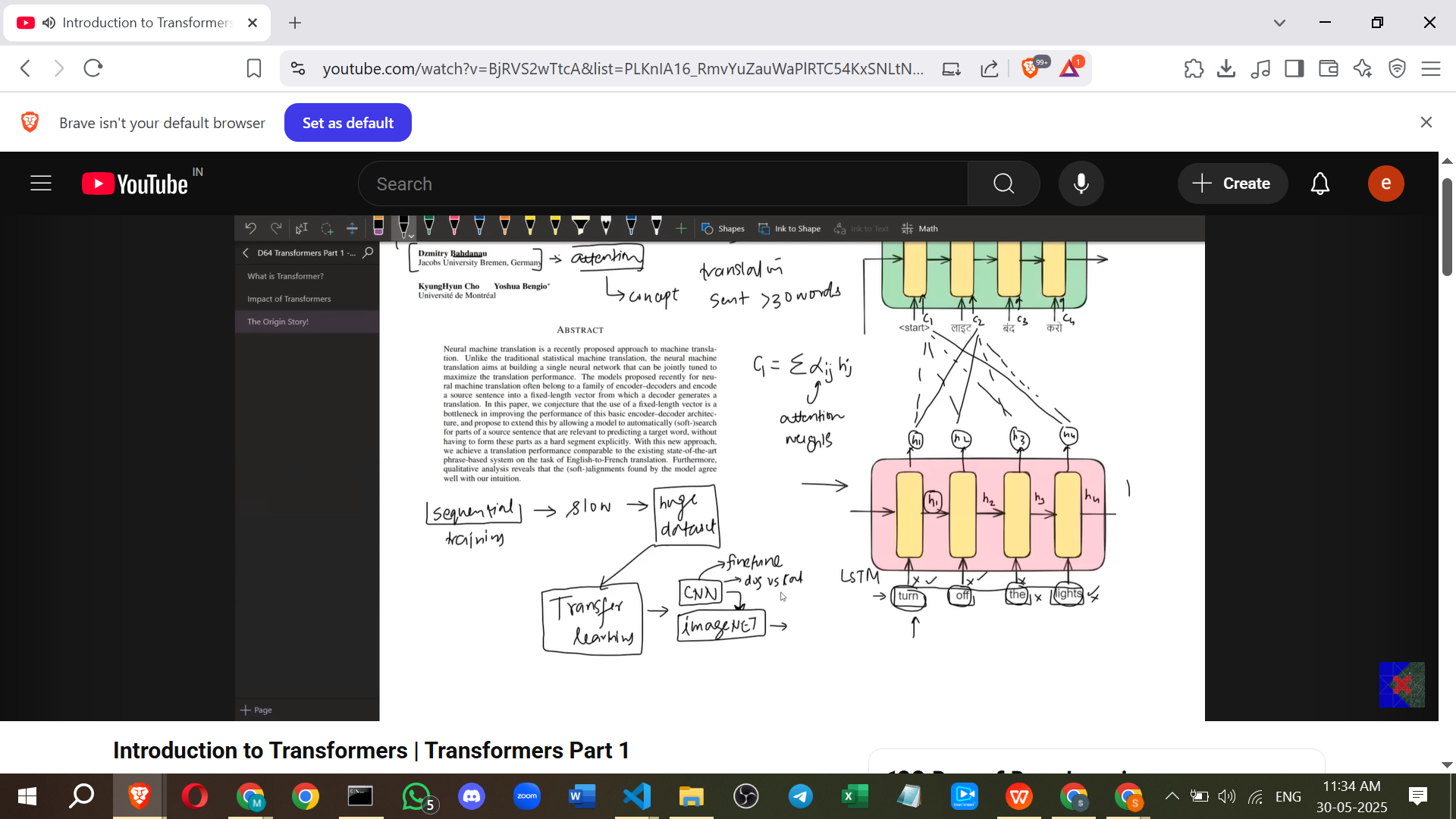


* **Chapter 2: Neural Machine Translation by Jointly Learning to Align and Translate (introducing Attention)**

**[[1409.0473] Neural Machine Translation by Jointly Learning to Align and Translate](https://arxiv.org/abs/1409.0473)**

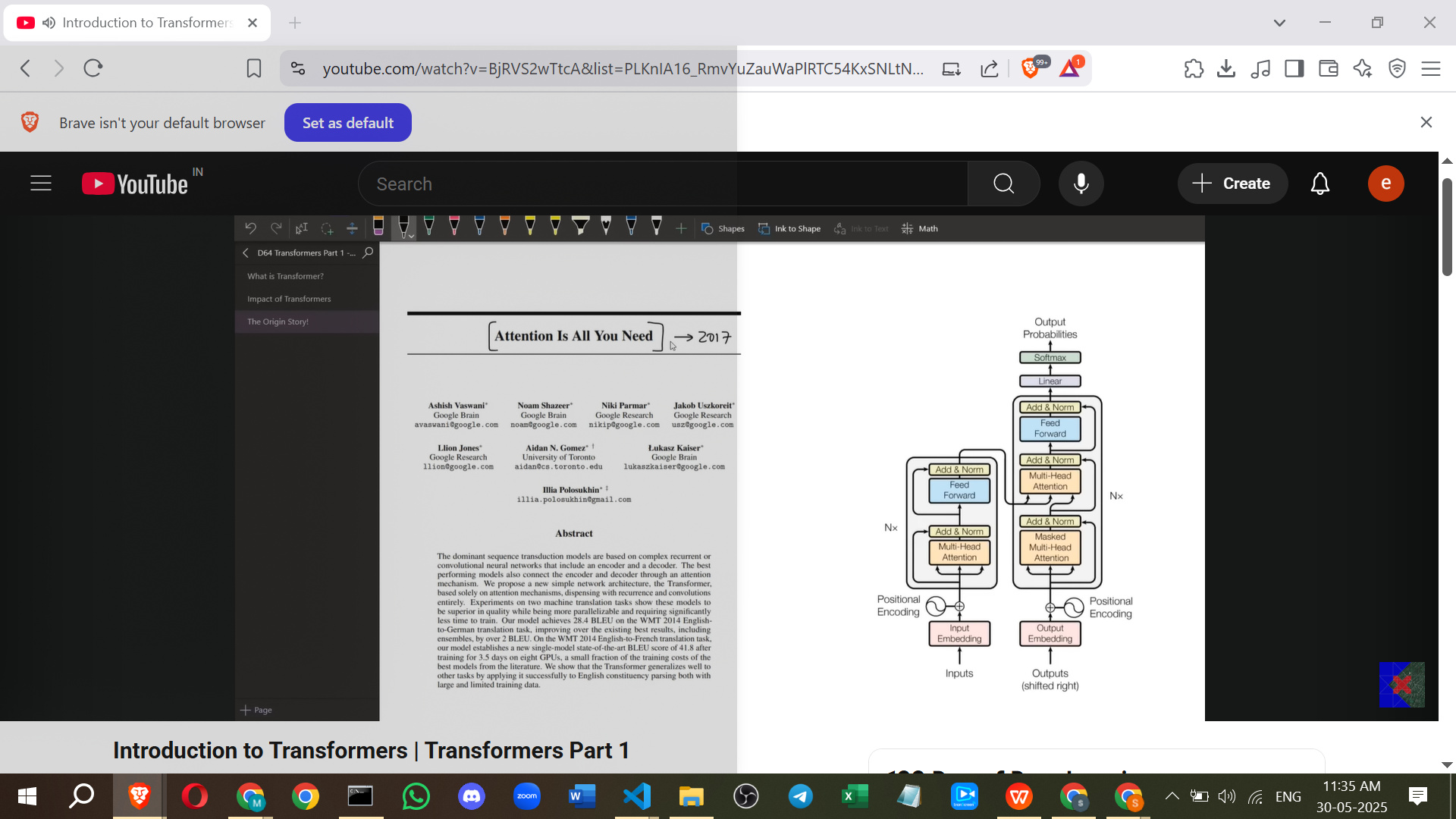
****

* + This paper (by Bahdanau et al.) introduced the **concept of Attention** to solve the problem of losing context in long sentences.
  + **How Attention works**: Instead of a single context vector, the decoder receives a **dynamic context vector at each time step**.
  + This context vector is a **weighted sum of the encoder's hidden states**, where the weights (attention weights) determine which input words are most relevant for generating the current output word.
  + This improved translation quality, especially for longer sentences.
  + **Remaining Problem**: The architecture still relied on LSTMs, meaning training was **inherently sequential**. This made training slow and prevented it from being trained on very large datasets.
  + This sequential limitation hindered the application of **Transfer Learning** in NLP. Transfer learning, previously successful in Computer Vision (e.g., using CNNs pre-trained on ImageNet and fine-tuning for specific tasks like dog/cat classification), requires training on massive datasets first. Since the LSTM-based models couldn't train on such scale, transfer learning wasn't feasible. This meant building any new NLP application required training a model from scratch, which was time-consuming, costly (requiring lots of data and labelling), and often resulted in suboptimal performance.



* **Chapter 3: Attention Is All You Need (2017)**

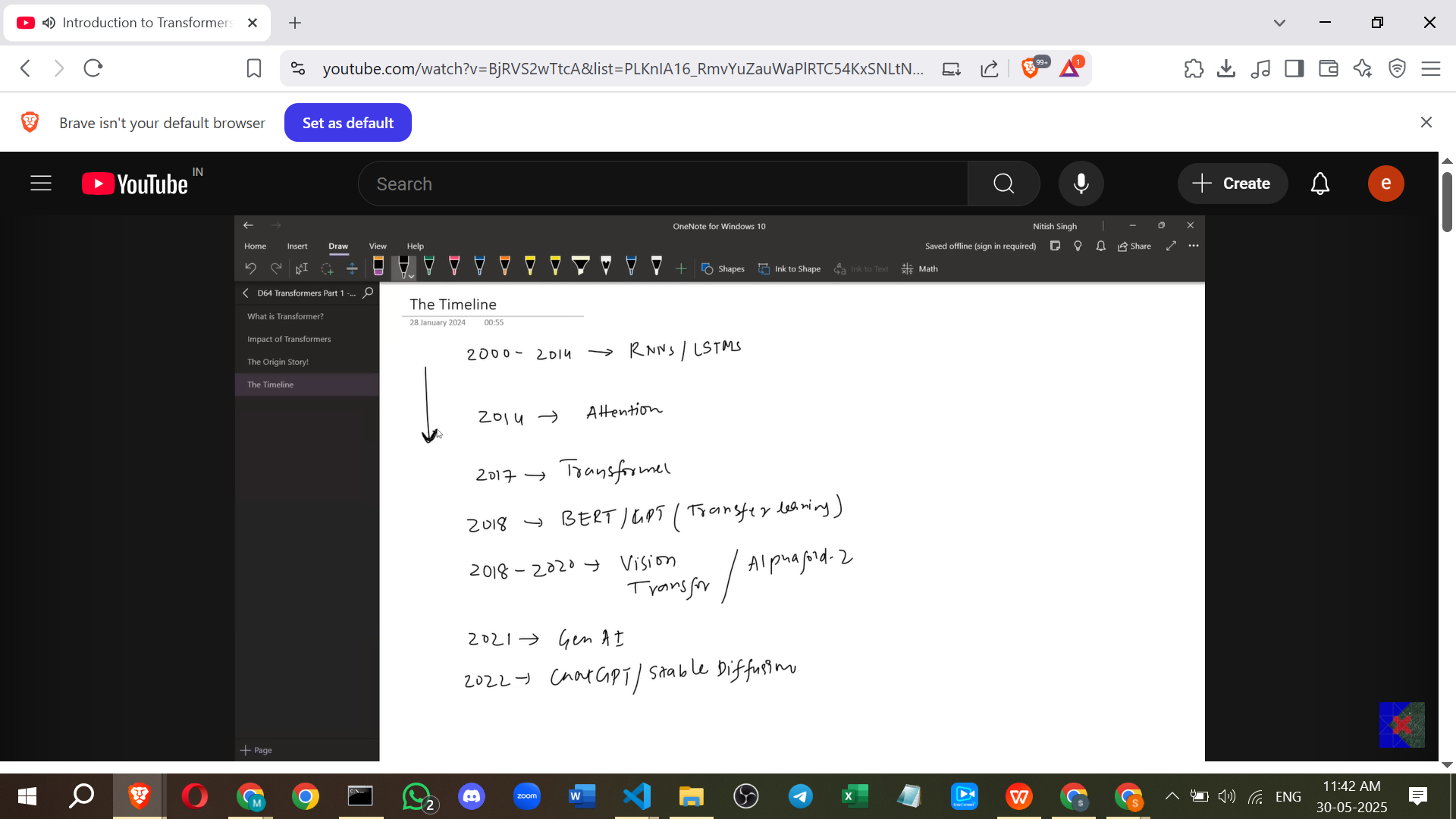
**[Attention Is All You Need](https://arxiv.org/html/1706.03762v7)**

****

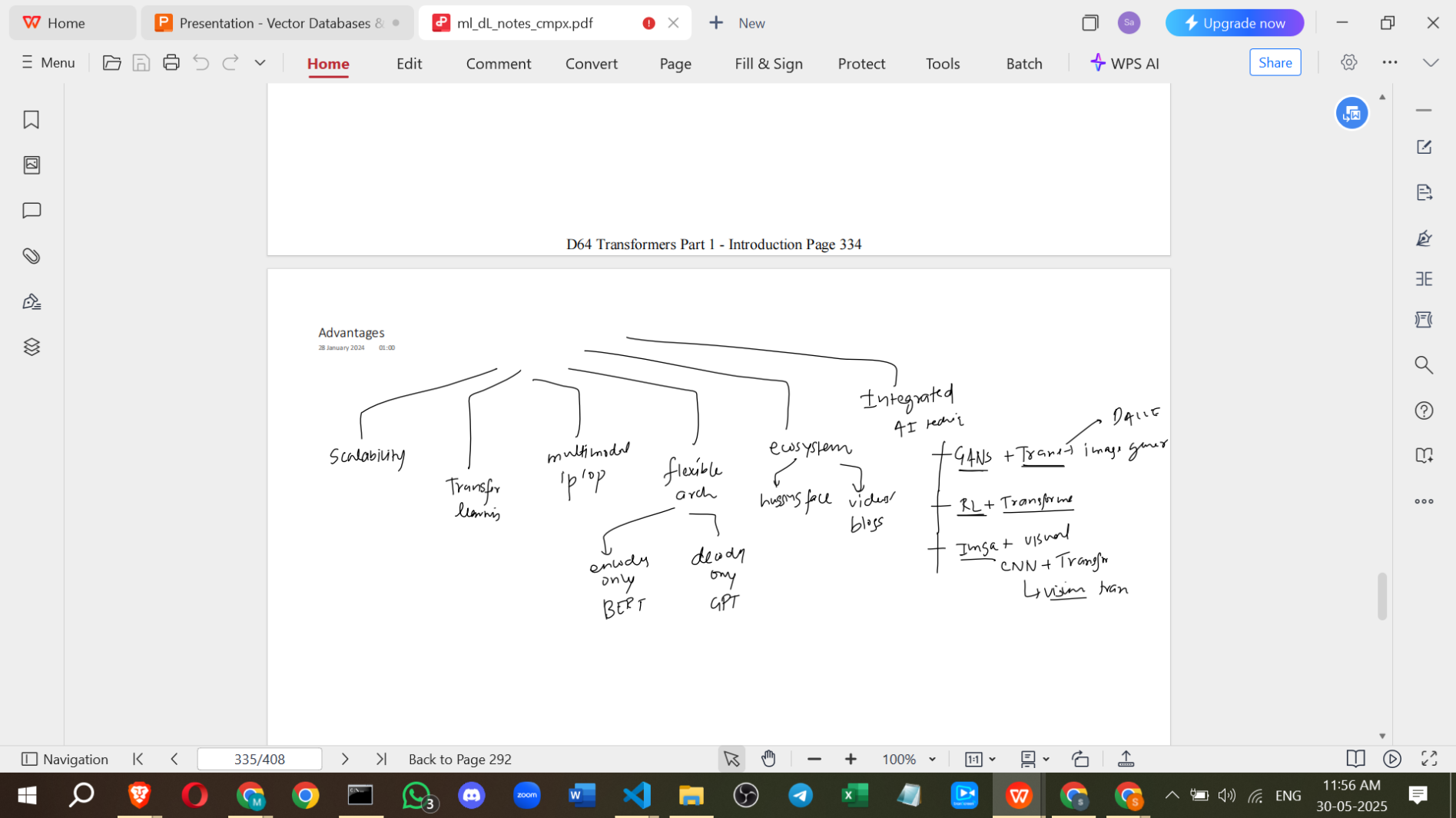
* + This landmark paper **introduced the Transformer architecture**.
  + It **completely solved the sequential training problem** by removing LSTMs and relying solely on Self-Attention.
  + This made the architecture trainable **in parallel**, leading to much faster training.
  + The ability to train in parallel made the architecture **highly scalable**, allowing it to be trained on huge datasets.
  + This scalability enabled the introduction of **Transfer Learning in NLP**. Pre-trained Transformer models like BERT and GPT could be trained on massive text corpora and then easily fine-tuned for specific tasks.
  + The paper is considered groundbreaking because it was **not incremental**; it introduced a completely new architecture from scratch.
  + The architecture is noted for being stable and robust due to the interplay of various components and stable hyperparameters.

**Timeline Overview**

* **2000-2014**: RNNs and LSTMs dominate NLP.
* **2014**: Encoder-Decoder architecture introduced, followed by the Attention mechanism.
* **2017**: **Attention Is All You Need paper** introduces the Transformer architecture.
* **2018**: Larger Transformers like BERT and GPT are trained, **starting the era of Transfer Learning in NLP**.
* **2018-2020**: Transformer concepts introduced to **different domains** like Computer Vision (Vision Transformers) and science (AlphaFold 2).
* **2021 onwards**: The era of **Generative AI** accelerates, with tools like GPT-3, DALL-E, and Codex emerging.
* **2022-Present**: Dominated by models like **ChatGPT** and Stable Diffusion, largely based on Transformers.



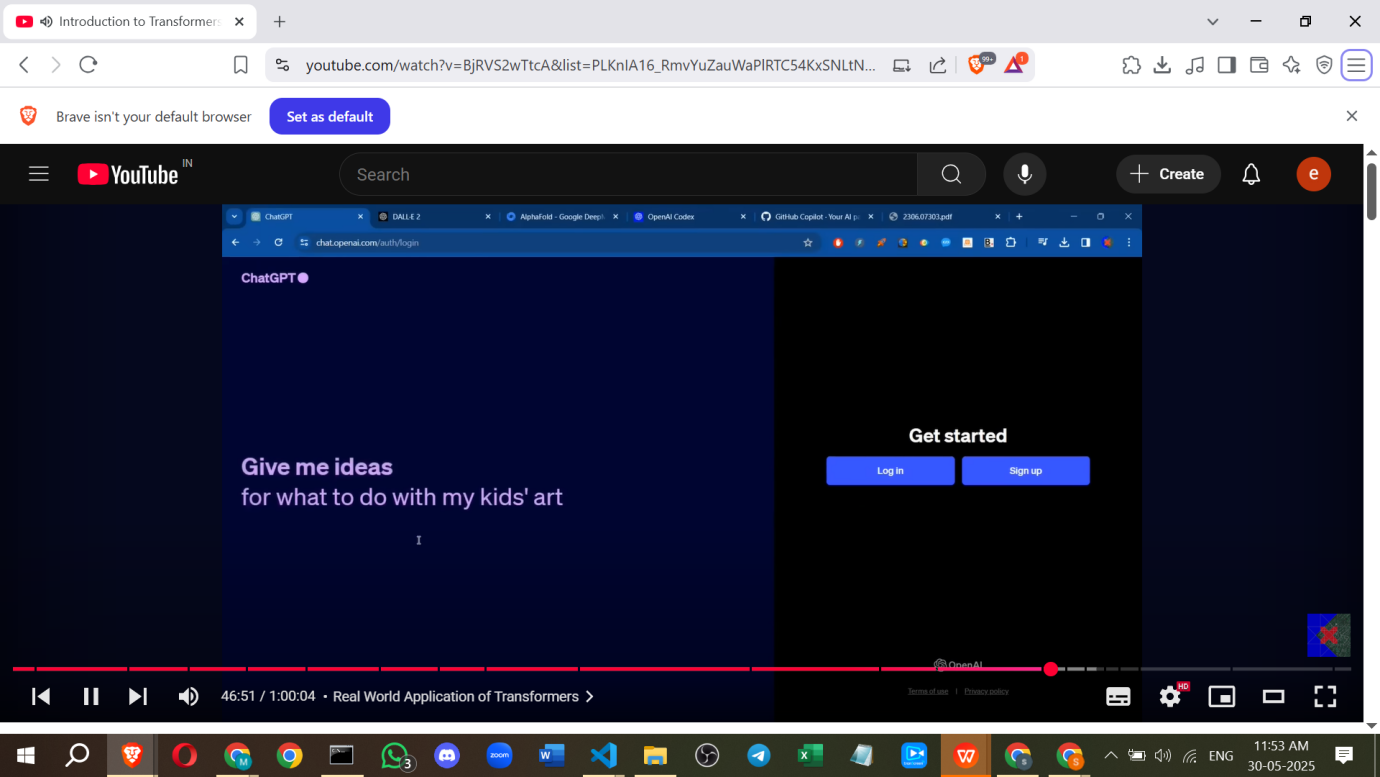
**Advantages of Transformers**

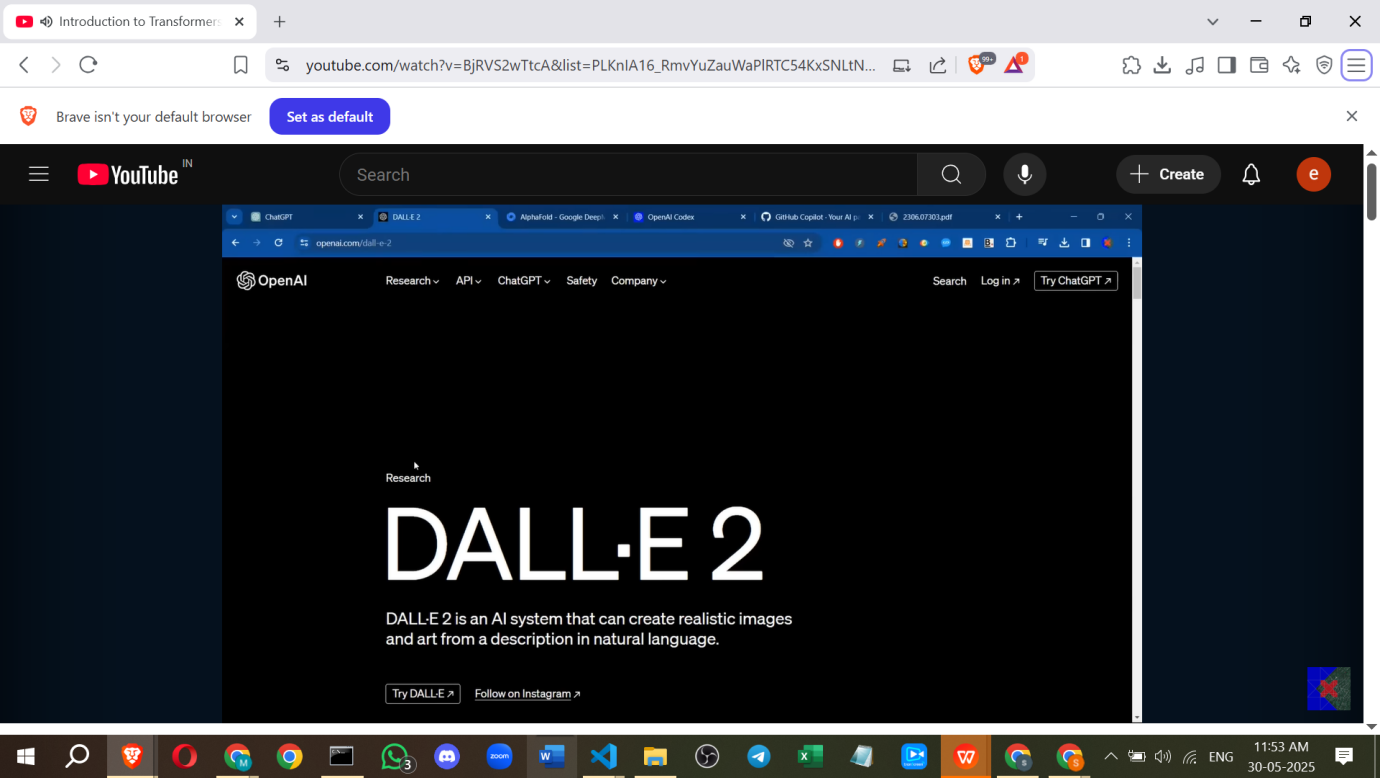
****

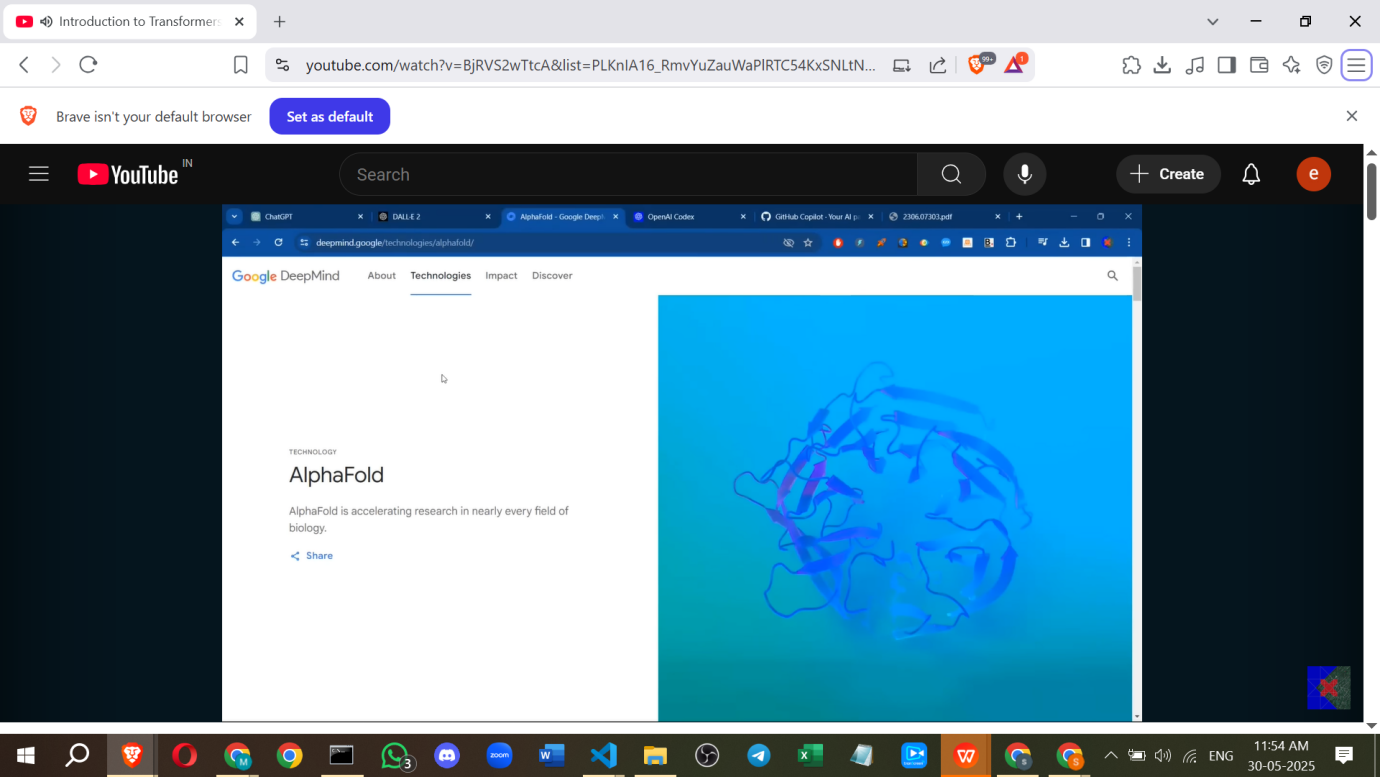
* **Scalability**: Ability to train parallelly, resulting in faster training and handling of large datasets.
* **Transfer Learning**: Can be pre-trained on massive datasets and fine-tuned for custom tasks with minimal effort.
* **Multimodal Input/Output**: Flexible architecture allows handling different data types (text, images, speech, etc.) if appropriate representations are created. This enables diverse applications.
* **Flexible Architecture**: Can be adapted for different needs (e.g., Encoder-only like BERT, Decoder-only like GPT).
* **Vibrant Ecosystem/Community**: Highly active research and development leads to many libraries (like Hugging Face), tools, videos, and blogs.
* **Easy Integration**: Can be combined with other AI techniques (like GANs for image generation or CNNs for visual tasks) to create interesting applications.

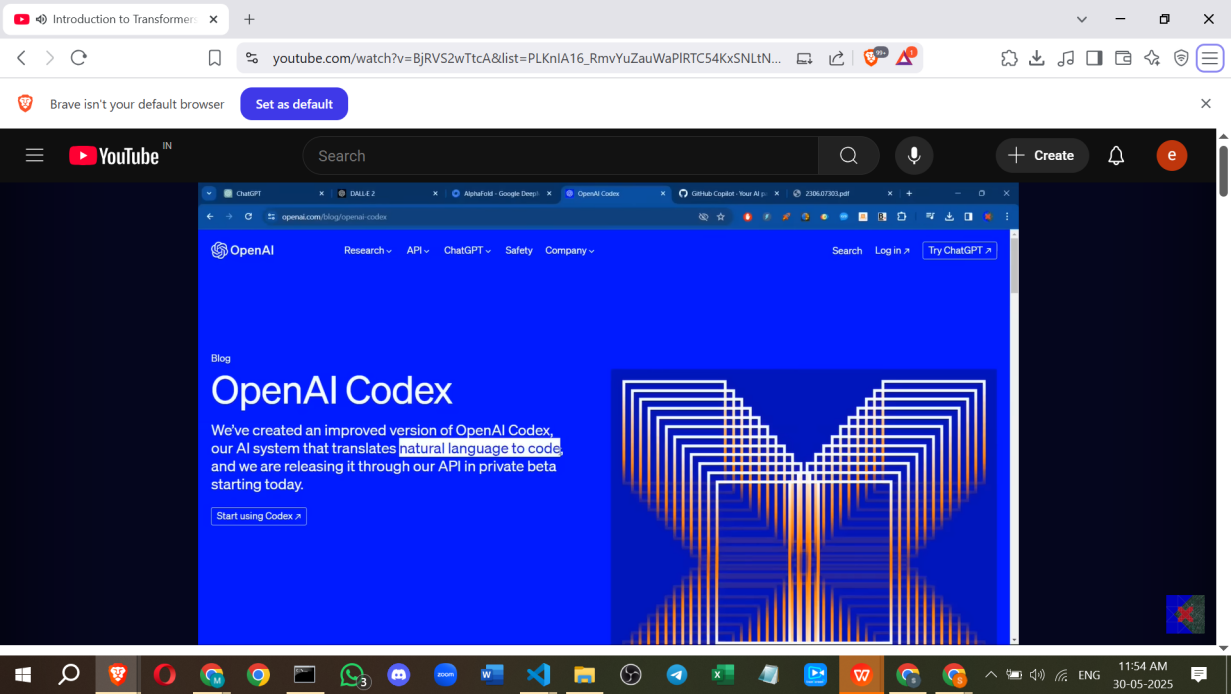
**Major Applications of Transformers**

* **ChatGPT**: A chatbot built on GPT-3 (a Generative Pre-trained Transformer), capable of generating human-like text for various purposes.
* **DALL-E 2**: A tool by OpenAI that generates images from text prompts.
* **AlphaFold 2**: Developed by DeepMind, uses Transformers to predict the 3D structure of proteins, a significant scientific breakthrough.
* **OpenAI Codex / GitHub Copilot**: A tool (based on OpenAI Codex) that converts natural language descriptions into code and provides code recommendations.
* Many other Transformer-based applications exist across different categories.



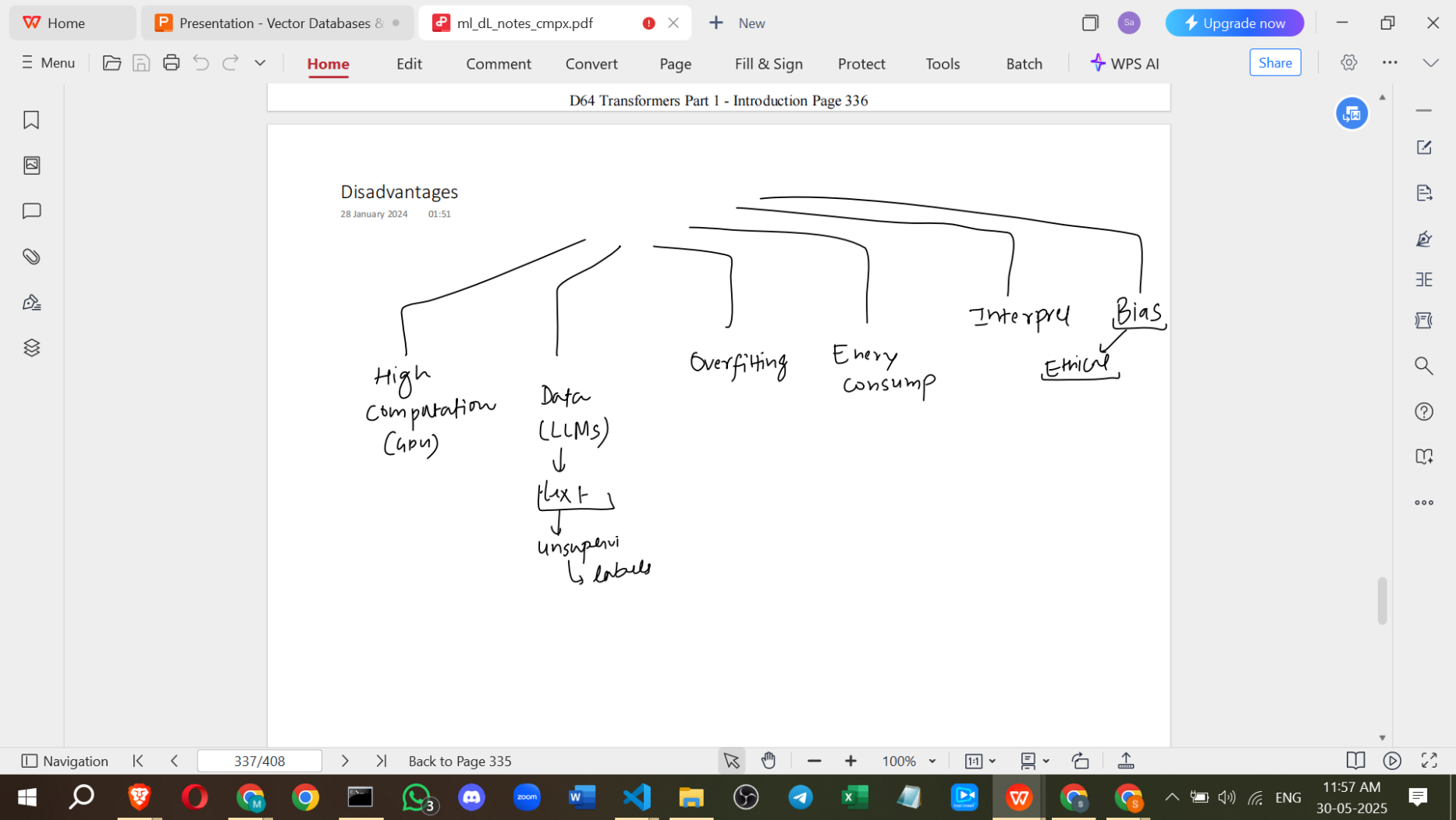






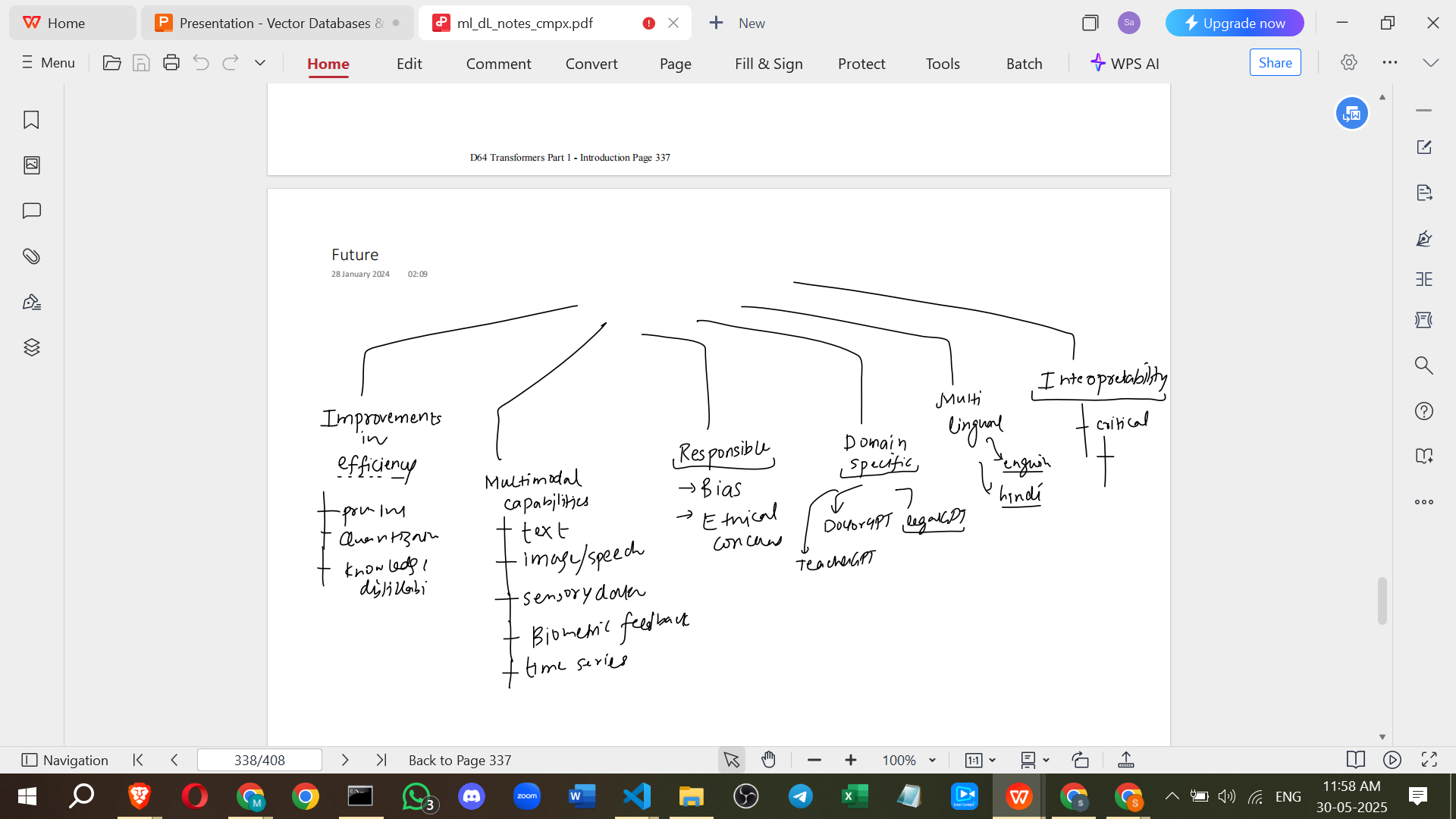
[[2306.07303] A Comprehensive Survey on Applications of Transformers for Deep Learning Tasks](https://arxiv.org/abs/2306.07303)

**Disadvantages of Transformers**

****

* **High Computational Resources**: Training requires significant processing power, typically GPUs, which are costly.
* **Data Requirements**: Like other deep learning models, Transformers generally require a lot of data for effective training, especially for domain-specific tasks.
* **Overfitting Risk**: With many parameters, there's a strong chance of overfitting if data lacks variety.
* **Energy Consumption**: Training large models requires substantial hardware and electricity, raising environmental concerns.
* **Interpretability**: Transformers are largely **black box models**; while results are good, it's difficult to understand *why* a specific output was generated. This can be a problem in critical sectors like banking or healthcare where explainability is important.
* **Bias and Ethical Concerns**: Models trained on internet-scale data can inherit biases present in the data. Using vast amounts of data scraped from the internet also raises ethical and legal issues.

**Future of Transformers**

****

* **Improving Efficiency**: Research is ongoing to reduce model size and training costs through techniques like pruning, quantisation, and knowledge distillation.
* **Enhanced Multimodal Capabilities**: Increased focus on handling modalities beyond text, such as images, speech, sensory data, and time series data, to create applications that can handle multiple types of input simultaneously.
* **Responsible Development**: Efforts to eliminate bias and address ethical concerns will be crucial.
* **Domain-Specific Models**: Development of specialised Transformers trained on data from specific fields (e.g., "Doctor GPT", "Legal GPT").
* **Multilingual Development**: Training Transformers from scratch or adapting existing ones for various regional languages beyond English.
* **Improved Interpretability**: Research aims to make Transformers less of a black box by trying to understand *why* they produce certain results, which is necessary for their adoption in critical domains.

======================================

=

1. Basic + Intermediate Python
2. Basic of ML/Deep Learning
3. NLP basic + Intermediate
4. Transformers
   1. Encoder BERT
   2. Decoder GPT
5. LLM