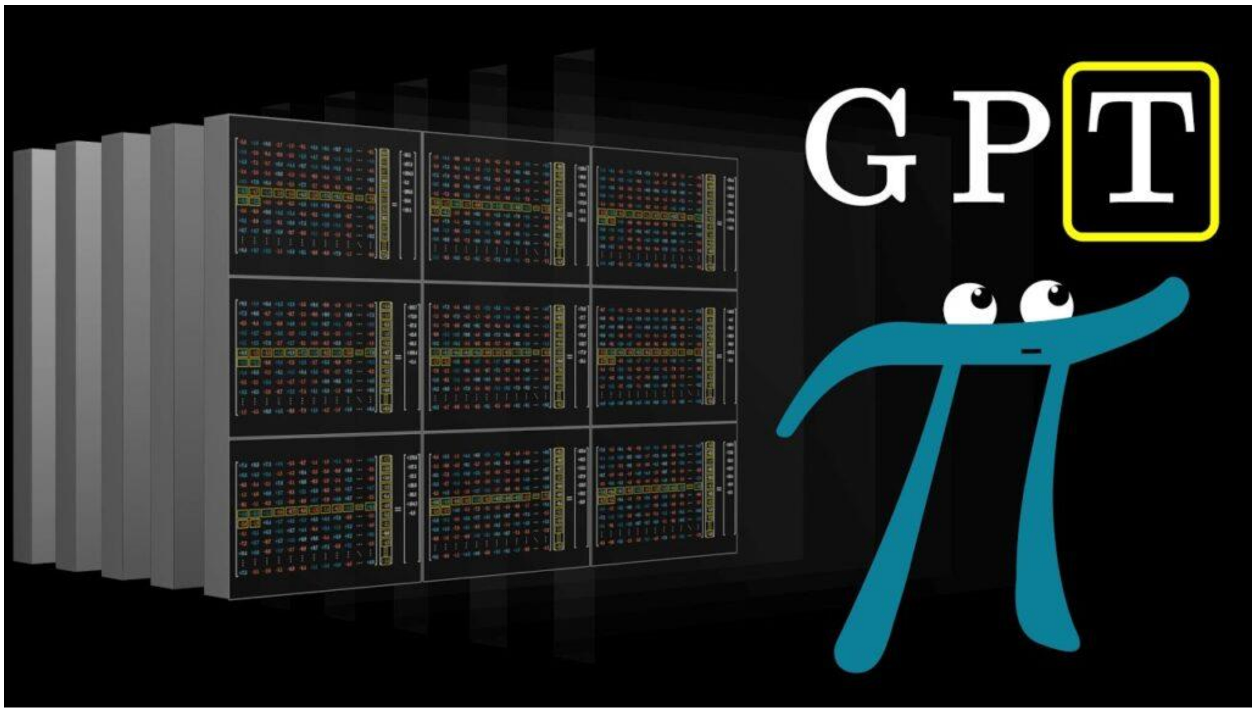
WIDS Report

Coding A ChatGPT Like Transformer From Scratch



Bhavya Patel (25b2477)

Mentor: Sandipan & Parth

Abstract:

The WiDS 5.0 (Winter in Data Science) project aimed to create a solid foundational knowledge of today’s Deep Learning architectures by building a GPT-like transformer model from the ground using PyTorch. Rather than relying on high-level APIs, the purpose of this project was to have a comprehensive understanding of the inner workings of NN's (neural networks) and the attention mechanisms of transformer architectures that are responsible for powering LLMs (Large Language Models) such as ChatGPT.

Over 5 weeks, the project progressed from basic Python and Neural Network concepts, to implementing a Basic Bigram Language Model, through to implementing a full transformer model with masked self-attention, multi-head attention, feed-forward layers, and positional embeddings. This report outlines the learning path taken, the conceptual understanding gained, the implementation details, experiments conducted, challenges encountered, and lessons learnt during the WiDS 5.0.

1) Introduction about Project:

The advent of Large Language Models (LLMs) has transformed how Artificial Intelligence performs by enabling machines to read and write as humans. Examples of these types of models include GPT, Claude, and Gemini, which use an architecture called the Transformer model, in particular a decoder only transformer for autoregressive text generation.

The goal of the WiDS 5.0 (Winter in Data Science) project is to create a step by step implementation of a GPT like model from scratch so that we can see how the underlying mathematical, data flows and training dynamics lead to the modern language models we see today.

The completed product of the project is a fully functioning GPT style language model trained on character data, capable of producing coherent text that appears as if it was produced in English language format.

2) Week 1: Python Refresher and Neural Network Basics

2.1 Python Fundamentals

The first week focused on refreshing Python fundamentals. Since Python is the dominant language for machine learning and deep learning, it was important to revisit core concepts such as:

- Their uses and types of variables and how to declare them

- How and when to use the main types of control structures (i.e., loops and conditional statements)

- How to organize your logic into separate functions and modules

- How to navigate file systems and manipulate files within them

The use of common Python libraries (e.g., NumPy, Pandas, and Matplotlib) was also pointed out so that participants could gain a beginning understanding of them as they will not be utilized to any great extent throughout the course of the program. However, they are fundamental libraries that are widely used in the quantitative analysis of data.

2.1 Neural Network Overview

An introductory overview of what a neural network is, how neurons are connected in multiple layers, forward propagation, loss functions, gradient descent, and backpropagation was presented during the week. The focus was not on memorizing formulas or definitions but rather developing an intuitive range of understanding as to how neural networks are able to learn from the data they utilize to determine the relationships that exist in the data.

3) Week 2: Deep Dive into Neural Network Components

3.1 Understanding Network Internals

In Week 2, we built on the basics from Week 1 by learning more about the following topics:

- Linear layers and weight matrices
- Activation functions like ReLU and Softmax
- Model capacity and overfitting
- The difference between training and validation loss

This phase helped us see how minor design choices affect learning performance.

3.2 Optional Advanced Learning

We were encouraged to seek out and explore other learning opportunities, such as courses offered by Stanford University or DeepLearning.ai. These additional resources provided theoretical context and solidified concepts of optimization, regularization, and representation learning.

3.3 Moving Towards Transformers

By the end of Week 2, the focus started to shift to understanding how we represent sequences, as this will become crucial to performing the language tasks we are going to learn in the following weeks. This led up to our going to learn about attention mechanisms and Transformers.

4) Week 3: PyTorch and the Transformer Architecture

4.1 Introduction to PyTorch

The third week transitioned from theory-based learning to actually applying theory. The main deep learning framework covered was PyTorch. Some key takeaways were

Tensors are multi-dimensional arrays

GPU

Autograd & automatic differentiation.

defining neural network modules

Tensors also represent a key concept to understand as Transformers rely heavily on matrix multiplication and reshaping of tensors.

4.2 Transformer Fundamentals

Participants were introduced to The Illustrated Transformer, which provided visual/conceptual representations of the Transformer architecture(s). Some key topics discussed were:

Self-Attention

Query/Key/Value

Sequence Processing in Parallel

Differences between Encoder/Decoder Architectures

The week focused heavily on developing an understanding of how well Transformers perform for language modeling purposes.

5) Mid-Term Assignment: Bigram Language Model

5.1 Introduction:

The first model used in the project prior to developing a complex transformer was a simple bigram language model, which predicts the next character in a sequence based solely on the preceding character with no memory of previous context.

5.2 Implementation of the Bigram Model

The implementation of the bigram model was done by:

Encoding each character into an integer

Learning to generate a probability distribution of the next possible character

Using cross-entropy loss for training

The bigram model was used as a baseline to establish:

Tokenization methodologies

Suggestions for loss calculations

Suggestions for training loops

Logic behind producing textual output

5.3 Results

The final outcome produced from this model was purposely meaningless; it looked like broken-english. However through these results we were able to see that this particular model has some level of learning as it was able to develop its own statistical patterns through the provided data.

Through this specific experiment it was established that models built without context will not perform well when trying to make accurate predictions or learn to be accurate.

6) Week 4: Masked Self-Attention and GPT Upgrade

6.1 Why Masking Is Necessary

In Week 4, a significant idea was introduced: masked self-attention. Without masking, a model would be able to look forward to future tokens when training, violating the autoregressive feature of modeling language.

By adding masking to self-attention, each token can only see previous tokens. As a result, each token has an incomplete depiction of what the future contains.

6.2 Decoder-only Transformer

We used a decoder-only Transformer in this project (like the GPT architectures). Each of the following features were necessary parts of the project:

- Causal masking
- Self-Attention blocks
- Residual connections
- Layer normalization

Studying these individual components allowed students to see how a Transformer can be powerful and still stable for training.

6.3 Karpathy's implementation of GPT

We picked up where they left off on the tutorial and implemented a series of pieces so that they could continue improving their existing Bigram model from Week 2 into an initial Transformer that contained information from a context of anything.

7) Final Project: GPT-Like Transformer Implementation

7.1 Model Components

The final model included the following core components:

Self-Attention Head

Calculates attention weights using Query, Key, and Value projections.

Multi-Head Attention

Runs multiple attention heads in parallel, allowing the model to capture different types of relationships simultaneously.

Feed-Forward Network

Applies non-linear transformations to enhance representation learning.

Transformer Block

Combines attention and feed-forward layers with residual connections.

Positional Embeddings

Adds information about token order, enabling the model to understand sequence structure.

8) Training and Experiments

8.1 Training Setup

- Training iterations: 3,000–5,000
- Optimizer: Adam
- Learning rate: Tuned to avoid overfitting
- Batch size and block size adjusted for memory constraints

8.2 Evaluation Metrics

- Training loss
- Validation loss
- Sample text generation quality

The target validation loss range of **1.5–2.0** was achieved, indicating effective learning.

8.3 Generated Output

The final model generated text with:

- Correct spelling
- Basic grammar
- Coherent sentence structures

While still limited in complexity, the output demonstrated the power of self-attention and contextual learning.

9) Challenges and Debugging

Several challenges were encountered during implementation:

- Tensor dimension mismatches
- GPU memory limitations
- Overfitting on small datasets

Debugging strategies included printing tensor shapes, reducing model size, and adjusting learning rates. These challenges significantly improved practical debugging skills.

10) Key Learning(s) and Learning's:

Transformers rely heavily on matrix operations and attention mechanism(s).

Masked self-attention is critical in autoregressive models.

Building models from scratch provides much more insights than using API.

Training dynamics and hyper parameter have a tremendous effect on model performance.

More importantly, this project built a substantial understanding from abstract concepts.

11) Conclusion Of The Project:

The WiDS 5.0 project successfully guided participants through building a GPT-like Transformer from scratch which is among the most challenging and rewarding tasks available to today's AI. The project has created a significant understanding of the architecture that underlies state-of-art language models; it provides both an improved theoretical foundation to apply techniques in ML and has contributed to building the concrete competence necessary to work and know this field.

The completion of this project represents not only a technical accomplishment but also a fundamental understanding of how to build modern AI systems. This understanding lays the groundwork for continued exploration into advanced NLP, large-scale model training, and AI research.

References

- Andrej Karpathy – *Let's Build GPT from Scratch*
- Jay Alammar – *The Illustrated Transformer*
- Harvard NLP – *The Annotated Transformer*