Large Small Language Model

Paridhi Lohani

Deep Learning 14x050

Autumn 2024

Aim

To implement character-based, character-level language model and train it on the Shakespeare dataset with the objective of generating Shakespearean like text given a seed string

In addition to this I also aim to make the model compatible with running on a GPU for faster training times than those possible on a CPU

Transformer Decoder-Only Architecture

The below is the transformer architecture (Encoder and Decoder) from the paper Attention is all you need. Following it we discuss what happens when the encoder is removed from this architecture to give us decoder only architecture.

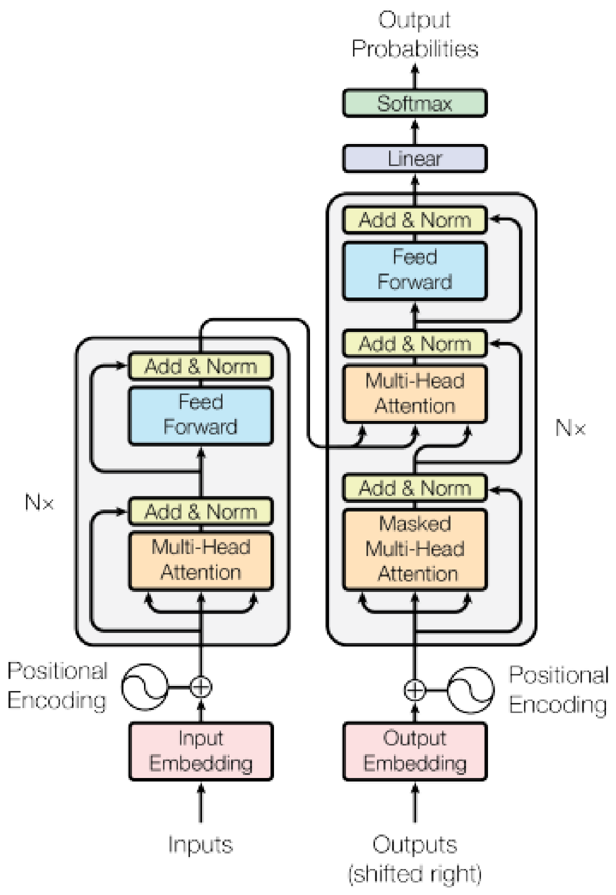


Fig 1: Transformer model Architecture

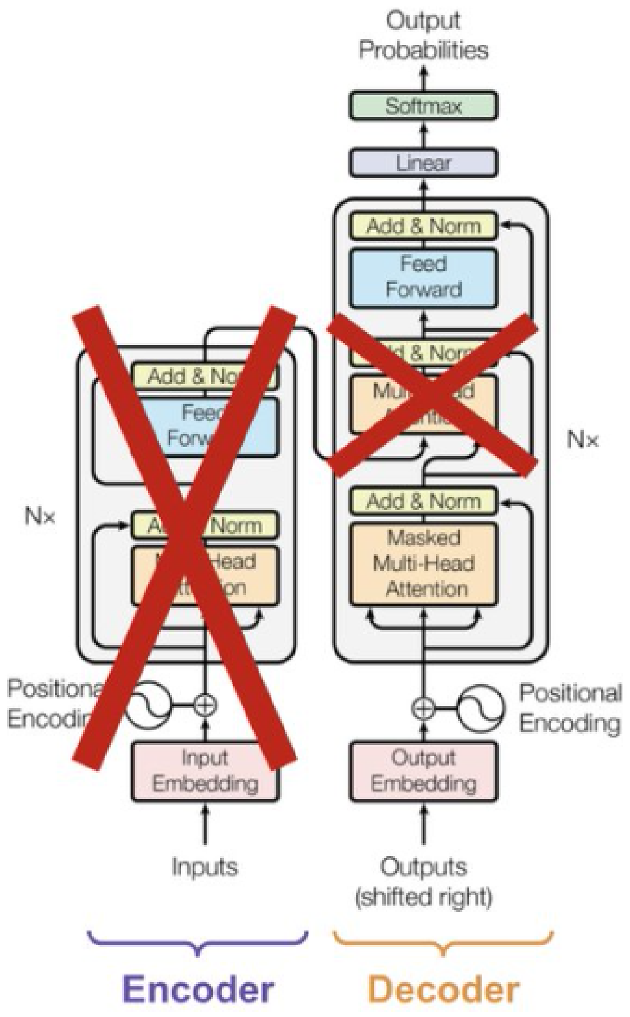


Fig 2: Removing the Encoder

In a decoder-only architecture only the decoder part works on the inputs and encoder is removed (including the cross attention portion of the decoder because the encoder does not exist so cross-attention is not required)

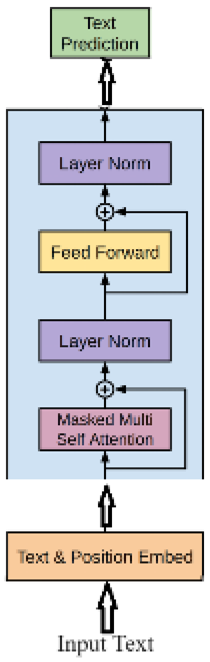


Fig 3: Decoder only Architecture

The Code

Input Preparation and CharDataset Class

From the url that is the source of the Shakespeare dataset, I extract the text using the urllib library. Then I pass the results to the CharDataset class which takes care of tokenization and emits batches of characters. It also gives us other parameters of the dataset like vocabulary size (number of characters in dataset), length of the dataset in terms of number of sliding windows of given length (block_size) that can be extracted from it, stoi, the dictionary that maps characters to integer indices and itos which is the reverse.

Head Class

This class represents a single attention head in the multi-head attention mechanism. It computes the attention weights using the query, key and value matrices. It also applies causal masking so that the model pays attention to the right tokens.

Multi-Head Attention Mechanism Class

This class runs multiple head instances in parallel. It concatenates the output from each head instance and projects it to the original embedding dimension.

FeedForwardNetwork Class

This class is a Feed Forward Network that works on the output of the Multi-Head Attention Mechanism.
It applies a linear transformation, followed by a GeLu activation and another linear transformation and then applying a dropout

TransformerBlock Class

This class represents a single block in the transformer decoder. It combines all the previous classes to perform multi-head attention, feed-forward, layer normalisation and residual connections.
Thus this is a core building block of the model.

ShakespeareGPT Class

This class implements the GPT-like model
It initialises the token and position embeddings. It stacks multiple Transformer blocks to form the decoder. It has a final layer norm and a linear layer to predict/generate the next token. The forward method calculates the loss and returns the logits. The generate method gives the next token using the trained model by applying softmax function on the logits.

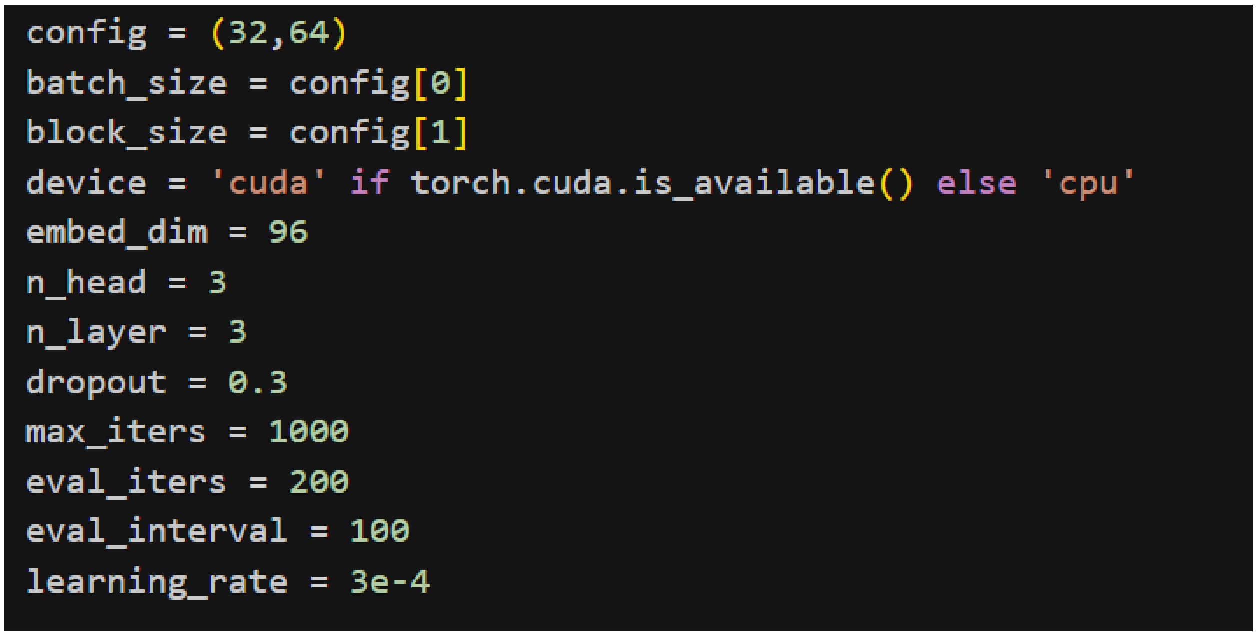
Overall Workflow

1. The dataset is loaded and preprocessed using CharDataset.
2. The ShakespeareGPT model is initialized and trained on the dataset.
3. The estimate_loss function evaluates the model's performance on the training and validation sets.
4. Finally, the generate_text function uses the trained model to generate new text given a prompt.

Hyperparameters

- config: A tuple containing batch size and block size (config = (batch size, block size))

- batch_size: number of sequences that are processed in parallel during training
- block_size: the maximum context length used by model for predictions i.e how many characters the model will look at
- device: This is the computing device the model uses for training
- embed_dim: Number of dimensions of the token and the positional embeddings. The size of the vector for each token. This vector is processed by the model
- n_head: The number of heads used for the multi-head attention mechanism
- n_layer: The number of layers or transformer blocks that are used in the model
- dropout: Dropout is a regularization technique to prevent overfitting by ignoring a number of randomly selected neurons. This hyperparameter is the dropout rate used in the model. Dropout rate is the fraction of neurons that are set to zero or ignored.
- max_iters: The number of iterations after which the training loop will stop
- eval_inters: The number of iterations during which the model runs in evaluation mode to evaluate the losses.
- eval_interval: Number of steps after which the model's performance is evaluated
- learning_rate: This is the learning rate for the optimizer ie the step size at which the model moves towards the minimum of the loss function in each iteration.

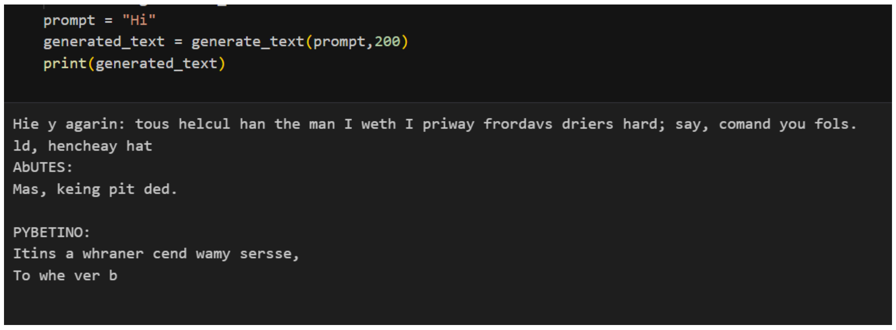


```
config = (32,64)
batch_size = config[0]
block_size = config[1]
device = 'cuda' if torch.cuda.is_available() else 'cpu'
embed_dim = 96
n_head = 3
n_layer = 3
dropout = 0.3
max_iters = 1000
eval_iters = 200
eval_interval = 100
learning_rate = 3e-4
```

Fig 4: Hyperparameters used in the model

Results

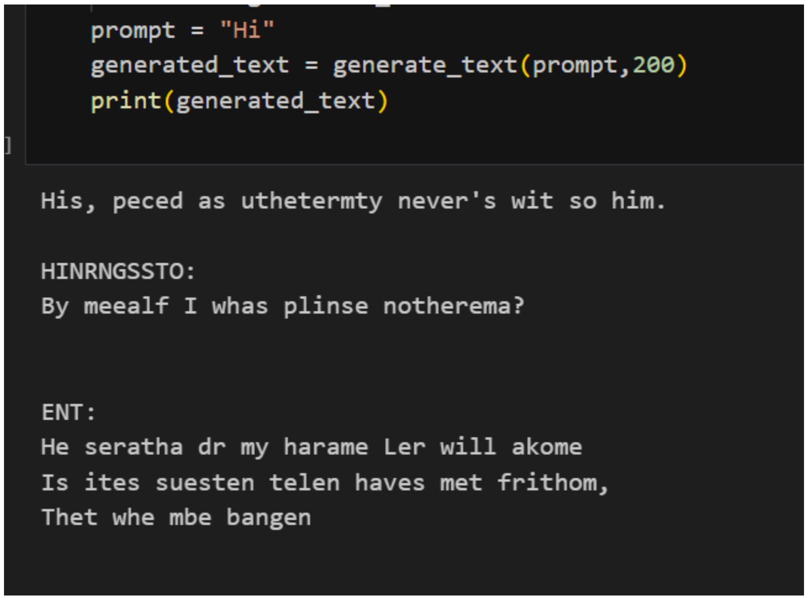
The following are my results for the seed string “Hi”. These are Shakespearean-looking but not necessarily meaningful.



```
prompt = "Hi"
generated_text = generate_text(prompt,200)
print(generated_text)
```

```
Hie y agarin: tous helcul han the man I weth I priway frordavs driers hard; say, comand you fols.
ld, hencheay hat
AbUTES:
Mas, keing pit ded.

PYBETINO:
Itins a whraner cend wamy sersse,
To whe ver b
```



```
prompt = "Hi"
generated_text = generate_text(prompt,200)
print(generated_text)
```

```
His, peced as uthetermty never's wit so him.

HINRNGSSTO:
By meealf I whas plinse notherema?


ENT:
He seratha dr my harame Ler will akome
Is ites suesten telen haves met frithom,
Thet whe mbe bangen
```

Fig 5: Results from the prompt “Hi”

Sources

- Shakespeare dataset https://raw.githubusercontent.com/karpathy/char-rnn/master/data/tinyshakespeare/input.txt
- Lectures and other resources for Deep Learning by Prof. Fleuret https://fleuret.org/dlc/
- Attention is All You Need Vaswani et al https://arxiv.org/abs/1706.03762
- How to Build an LLM from Scratch | An Overview https://www.youtube.com/watch?v=ZLbVdvOoTKM
- https://github.com/karpathy/minGPT