GPT-2 from Scratch

Project Overview

This project presents a from-scratch implementation of a GPT-2-style transformer model using PyTorch, trained on the TinyStories dataset from Hugging Face. It serves as an educational tool to explore the inner workings of transformer-based language models by recreating all components manually, without relying on PyTorch's built-in transformer modules.

Model Architecture

Core Components

1. **Positional Encoding**
   - Learnable position embeddings of shape `(block_size, n_embd)` added to token embeddings.
2. **Multi-Head Self-Attention**
   - Implements scaled dot-product attention with causal masking for autoregressive generation.
   - 4 attention heads, each of dimension 16 (`n_embd // n_head`).
   - Dropout disabled (`p=0.0`) for final training.
3. **Feed-Forward Network**
   - Two linear layers with ReLU activation.
   - Hidden size: 256 (4× expansion of embedding dim).
   - Dropout: 0.0.
4. **Transformer Block**
   - Each block contains:
     - Multi-head self-attention + residual connection + LayerNorm
     - Feed-forward network + residual connection + LayerNorm
   - Total: 4 stacked transformer blocks.
5. **Output Head**
   - LayerNorm followed by a linear projection back to vocabulary size.

Model Specifications

Component	Value
Vocabulary size	Unique characters in dataset
Embedding size	64
Context length	32 tokens
Layers	4 transformer blocks
Attention heads	4
Dropout	0.0

Dataset and Preprocessing

- **Dataset**: TinyStories (211,971 stories — 75% subset used)
- **Steps**:
    1. Load dataset from Hugging Face Datasets.
    2. Concatenate text into a single string.
    3. Create character-level vocabulary.
    4. Encode characters as integer tokens.
    5. Split into 90% training, 10% validation.
    6. Store as memory-mapped `.bin` files for efficient IO.

Training Details

Hyperparameter	Value
Batch size	16
Context length	32 tokens
Embedding dim	64
Attention heads	4
Transformer layers	4
Optimizer	AdamW
Learning rate	1e-3
Dropout	0.0
Iterations	10,000
Hardware	CPU-only (no GPU)

Loss

- Initial loss: ~5.26
- Final training loss: ~1.25
- Final validation loss: ~1.25
- Estimated perplexity: ~3.5

Evaluation

Quantitative

- Minimal gap between training and validation loss
- No sign of overfitting
- Stable training convergence

Qualitative — Sample Generation

Seed: single space character

```
"let oway.Mom said the comes off treeze.""

On. Lily didn't very having.Lily been?"

Her smiled and sece creat it's chate for Lily. They everyone that day, she
the veasitly with he couldn't want to not things. The rain named Lily.

When you have to threw are renched, notwies watchester.

One day, he had a little britch found silong.
```

Observations:

- Displays basic sentence-like structure
- Some valid word generation and punctuation
- Limited narrative coherence due to short context and character-level granularity

---

Strengths

- Transformer built entirely from scratch — no high-level modules
- Efficient memory usage with binary dataset storage
- Successful training of a GPT-style model on character-level data
- Model convergence with consistent training/validation loss

Limitations

- Character-level modeling is less expressive than subword tokenization
- Small model capacity restricts generation quality
- Short context length (32 tokens) limits long-range coherence
- No dropout regularization or GPU acceleration used

Future Improvements

1. Switch to subword tokenization (e.g., Byte Pair Encoding)
2. Scale model depth and width (more layers, heads, and embeddings)
3. Increase context window size for better coherence
4. Enable dropout for better generalization
5. Leverage GPU for faster training
6. Train for more epochs
7. Implement beam search or top-k sampling for improved generation

---

Conclusion

This project demonstrates a faithful implementation of a GPT-2-style transformer entirely from scratch, achieving competent performance on character-level story generation. While limited by model size and data representation, it lays a solid foundation for future scaling and experimentation in transformer-based language modeling.
