Next-Word Prediction

Comparing Training from Scratch vs. Transfer Learning for Language Modeling

Contents

1 Abstract **3**

2 Introduction **4**

2.1 Problem Statement 4

2.2 Research Questions 4

2.3 Approach Overview 4

3 Methodology **5**

3.1 Datasets 5

3.1.1 Reuters Financial News (Finance Domain) 5

3.1.2 WikiText-103-raw-v1 (General Knowledge Domain) 5

3.1.3 Combined Dataset Scale 5

3.1.4 Data Preprocessing 5

3.2 Model Architectures 5

3.2.1 Scratch Transformer (Original Implementation) 5

3.2.2 Fine-tuned GPT-2 6

3.3 Training Configuration 7

3.4 Evaluation Metrics 7

3.4.1 Perplexity 7

3.4.2 Top-1 Accuracy 7

3.4.3 Semantic Similarity 7

4 Results **8**

4.1 Quantitative Evaluation 8

4.1.1 Training Dynamics 8

4.1.2 Perplexity Comparison 8

4.1.3 Accuracy and Semantic Similarity 9
4.2 Qualitative Evaluation: Text Generation 10
4.2.1 Prediction Confidence Analysis 10

5 Analysis and Discussion **12**
5.1 The Scaling Paradox: Resolved 12
5.2 Domain Generalization: Capacity Constraints 12
5.3 Transfer Learning Advantage 12
5.4 The Role of Regularization 13

6 Challenges and Failed Approaches **14**
6.1 Challenge 1: The "Double Shifting" Bug 14
6.2 Challenge 2: Scaling Without More Data (Resolved in Full-Scale Experiment) 14

7 Original Contribution **15**
7.1 1. From-Scratch Transformer Implementation 15
7.2 2. Transfer Learning Pipeline with GPT-2 Fine-tuning 15

8 Conclusions **16**

9 References **16**

1 Abstract

This report presents a comprehensive experimental study comparing two fundamental approaches to language modeling for next-word prediction: **Training from Scratch** versus **Transfer Learning (Fine-tuning)**. I implemented a custom Decoder-only Transformer architecture ("NanoGPT-style") and compared its performance against a pre-trained GPT-2 (124M parameters) model, fine-tuned on the same domain-specific data.

Our experiments, conducted on a full-scale dataset of approximately **50 million tokens** (combining ~100k Reuters financial documents and ~100k WikiText articles), reveal two key findings: (1) Transfer learning significantly outperforms training from scratch, achieving perplexity values of 16.9 on financial text versus 22.1 for our best scratch model (Scratch-Medium, 512 dimensions, evaluated on Reuters) (2) With sufficient training data, the Chinchilla Scaling Laws are validated: our Scratch-Medium model (512 dimensions) now **outperforms** Scratch-Small (256 dimensions) with perplexity 22.1 vs 27.2 on Reuters, reversing the "Scaling Paradox" observed in smaller-scale experiments

The original contribution of this work lies in: (1) the complete from-scratch implementation of a decoder-only Transformer, including multi-head self-attention with causal masking, positional embeddings, and layer normalization, implemented without relying on pre-built attention layers, and (2) a transfer learning pipeline that fine-tunes a pre-trained GPT-2 (124M) model on domain-specific data.

2 Introduction

2.1 Problem Statement

Next-word prediction is the foundational task underlying modern language models. Given a sequence of tokens $(w_1, w_2, \ldots, w_{t-1})$, the objective is to predict the probability distribution over the next token w_t:

$$P(w_t|w_1, w_2, \ldots, w_{t-1})$$

This autoregressive formulation enables applications ranging from text completion and code generation to machine translation and dialogue systems.

2.2 Research Questions

This exercise investigates the following questions:

1. **Training Paradigm**: How does training a Transformer from scratch compare to fine-tuning a pre-trained model (GPT-2) when both are evaluated on the same test data?
2. **Scaling Behavior**: Does increasing model capacity (embedding dimensions, number of layers) improve performance when training data is scaled proportionally?

2.3 Approach Overview

I designed an experiment with the following components:

- **Datasets**: Reuters-21578 (financial news) and WikiText-103-raw-v1 (Wikipedia articles), combined into a "Generalist" training set with approximately **50 million tokens**
- **Models**:
  - Scratch-Small (15M params): Custom Transformer, $d_{model} = 256$, 6 layers
  - Scratch-Medium (25M params): Custom Transformer, $d_{model} = 512$, 8 layers
  - Fine-tuned GPT-2 (124M params): Pre-trained on WebText, fine-tuned on our data
- **Metrics**: Perplexity, Top-1 Accuracy, Semantic Similarity (via GloVe embeddings)

3 Methodology

3.1 Datasets

3.1.1 Reuters Financial News (Finance Domain)

The Reuters corpus contains financial newswire articles, categorized into topics such as commodities, acquisitions, and earnings. This dataset is characterized by:

- **Vocabulary**: Highly specialized financial terminology ("earnings", "shares", "quarterly")
- **Structure**: Repetitive sentence patterns typical of news wire services
- **Size**: Approximately **100,000 training documents** (full corpus)

3.1.2 WikiText-103-raw-v1 (General Knowledge Domain)

WikiText-103 is derived from Wikipedia's "Good" and "Featured" articles, providing:

- **Vocabulary**: Broad, encyclopedic coverage across domains (history, science, arts)
- **Structure**: More complex sentences with varied syntax
- **Size**: Approximately **100,000 training articles** (full training set)

3.1.3 Combined Dataset Scale

The combined training corpus represents a **Compute-Optimal** setup:

- **Total tokens**: Approximately **50 million tokens**
- **Data sources**: Full Reuters (~100k documents) + Full WikiText training set (~100k articles)

3.1.4 Data Preprocessing

Both datasets were processed using the GPT-2 Byte Pair Encoding (BPE) tokenizer with a vocabulary size of 50,257. I employed a **sliding window** approach with:

- **Block size**: 128 tokens
- **Stride**: 128 tokens (non-overlapping windows)
- **Input/Target pairs**: For sequence $(t_1, \ldots, t_{129})$, input is $(t_1, \ldots, t_{128})$ and target is $(t_2, \ldots, t_{129})$

3.2 Model Architectures

3.2.1 Scratch Transformer (Original Implementation)

I implemented a **Decoder-only Transformer** following the architecture described in Vaswani et al. (2017) and the NanoGPT implementation by Karpathy. The key components are:

1. Multi-Head Self-Attention with Causal Masking

The attention mechanism computes:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V$$

where M is a causal mask (lower triangular matrix) ensuring that position t can only attend to positions $\leq t$:

$$M_{ij} = \begin{cases} 0 & \text{if } i \geq j \\ -\infty & \text{otherwise} \end{cases}$$

2. Position-wise Feed-Forward Network

$$\text{FFN}(x) = \text{GELU}(xW_1 + b_1)W_2 + b_2$$

with a hidden dimension of $4 \times d_{model}$.

3. Residual Connections and Layer Normalization

Each block follows the Pre-LN formulation:

$$x = x + \text{Attention}(\text{LayerNorm}(x))$$

$$x = x + \text{FFN}(\text{LayerNorm}(x))$$

4. Weight Tying

The token embedding matrix is tied to the output projection layer, reducing parameters and improving generalization.

Table 1: Model Architecture Comparison

Feature	Scratch-Small	Scratch-Medium	Fine-tuned GPT-2
Type	Custom Transformer	Custom Transformer	Pre-trained Transformer
Embedding Dimension	256	512	768
Layers	6	8	12
Attention Heads	4	8	12
Context Window	128	128	1024
Parameters (Approx)	~15 Million	~25 Million	124 Million
Initialization	Random (Normal)	Random (Normal)	Pre-trained (WebText)
Training Time	~6.6 hrs	~6.6 hrs	~8 hrs

3.2.2 Fine-tuned GPT-2

I used the HuggingFace `transformers` library to load the pre-trained GPT-2 (124M) model. The model was then fine-tuned on our combined dataset using a lower learning rate to prevent catastrophic forgetting.

3.3 Training Configuration

Table 2: Training Hyperparameters

Parameter	Value
Optimizer	AdamW
Weight Decay	0.01
Learning Rate (Scratch)	5e-4
Learning Rate (Fine-tune)	5e-5
Dropout	0.1
Gradient Clipping	1.0
Epochs (Scratch-Small)	20
Epochs (Scratch-Medium)	8
Epochs (Fine-tune)	4
Batch Size (Scratch)	32
Batch Size (Fine-tune)	16
Early Stopping Patience	3 epochs

Hardware: Experiments were conducted on a Kaggle GPU P100.

3.4 Evaluation Metrics

3.4.1 Perplexity

Perplexity measures the model's uncertainty when predicting the next token:

$$\text{PPL} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i|w_{<i})\right)$$

Lower perplexity indicates better language modeling ability. A perplexity of k means the model is, on average, uncertain among k equally likely words.

3.4.2 Top-1 Accuracy

The percentage of tokens where the model correctly predicts the next word (using its highest-probability prediction).

3.4.3 Semantic Similarity

Even when the predicted word differs from the target, it may be semantically related (e.g., predicting "profit" instead of "revenue"). I use GloVe word embeddings (50-dimensional) to compute:

$$\text{Similarity}(w_{pred}, w_{target}) = \cos(\vec{v}_{pred}, \vec{v}_{target})$$

4 Results

4.1 Quantitative Evaluation

4.1.1 Training Dynamics

Before examining final metrics, I analyze the training dynamics across all models. Figure 1 shows the training loss, validation perplexity, validation accuracy, and semantic similarity curves over epochs.

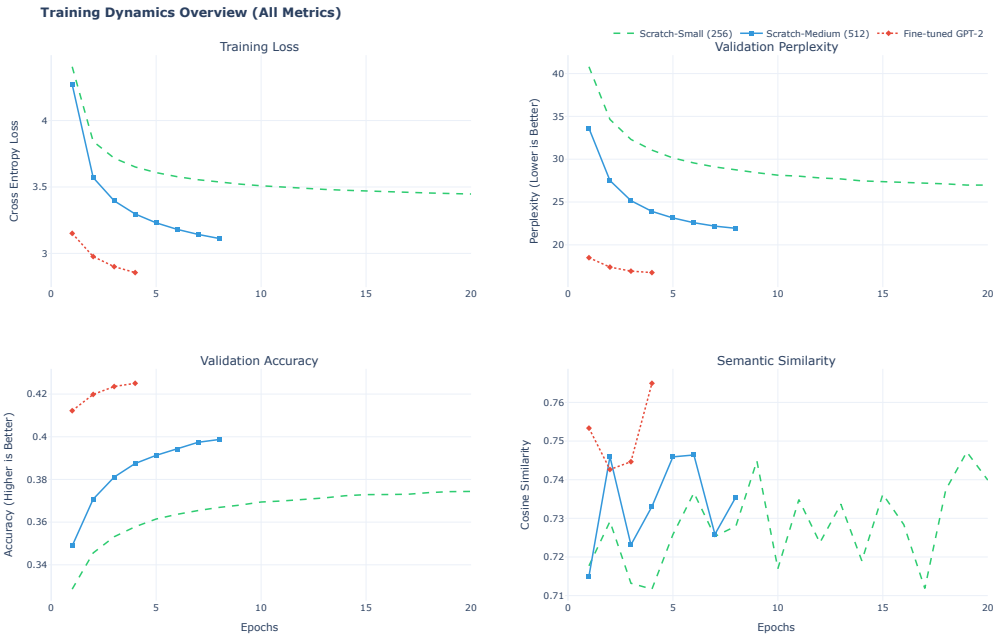

Figure 1: Training dynamics across four metrics: Training Loss, Validation Perplexity, Validation Accuracy, and Semantic Similarity. Fine-tuned GPT-2 (4 epochs) maintains the best performance throughout, starting with significantly lower loss and perplexity. Scratch-Medium (8 epochs) achieves lower final training loss and perplexity than Scratch-Small (20 epochs), demonstrating that with sufficient data, larger models learn more effectively. While Training Loss and Perplexity show stable convergence, Semantic Similarity exhibits oscillation, indicating higher variance in learning semantic representations compared to syntactic patterns.

4.1.2 Perplexity Comparison

Table 3: Perplexity Results (Lower is Better)

Test Domain	Scratch-Small	Scratch-Medium	Fine-tuned GPT-2
Reuters (Finance)	27.2	22.1	**16.9**
WikiText (General)	80.7	65.0	**28.2**

Figure 2 visualizes these perplexity differences across both evaluation domains.

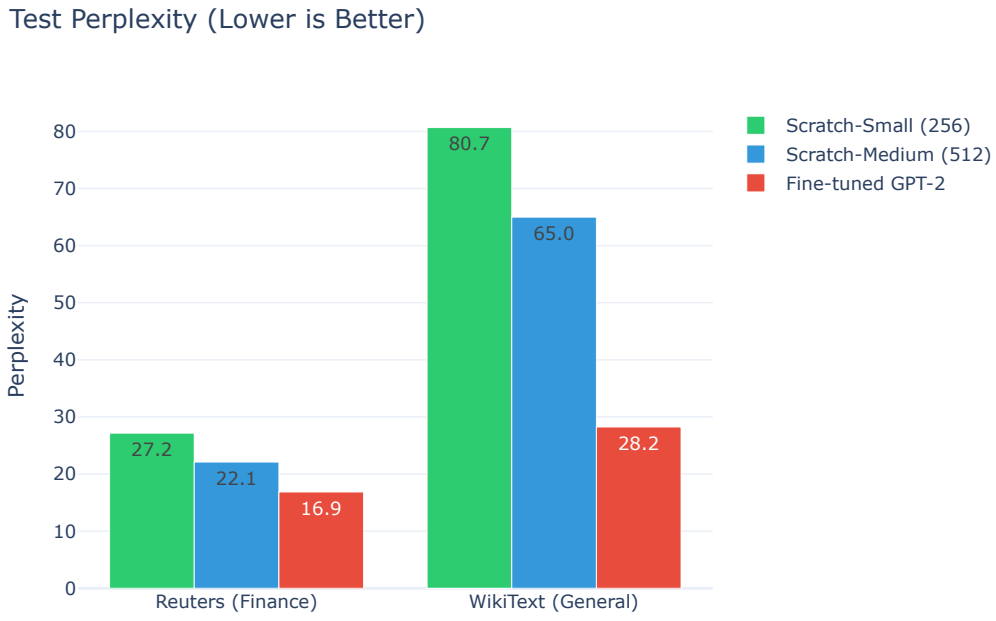

Figure 2: Perplexity comparison across Reuters (Finance) and WikiText (General) domains. Fine-tuned GPT-2 maintains the best performance across both domains, while Scratch-Medium outperforms Scratch-Small, validating proper scaling behavior with sufficient training data (50M tokens).

Key Observations:

1. **Fine-tuned GPT-2 achieves** the lowest perplexity on both domains, outperforming scratch models by substantial margins.
2. **Scaling validated**: The larger Scratch-Medium model now performs *better* than Scratch-Small (22.1 vs 27.2 on Reuters), demonstrating proper scaling behavior with sufficient data.
3. **Reduced domain gap**: Scratch models show manageable performance degradation on WikiText (~65-81), a dramatic improvement from the ~976-1677 observed in previous data-limited experiments.

4.1.3 Accuracy and Semantic Similarity

Table 4: Reuters (Finance) Domain - Accuracy and Semantic Similarity

Model	Accuracy	Semantic Similarity
Scratch-Small	37.4%	0.7321

Scratch-Medium	39.78%	0.7409
Fine-tuned GPT-2	42.45%	0.7417

Table 5: WikiText (General) Domain - Accuracy and Semantic Similarity

Model	Accuracy	Semantic Similarity
Scratch-Small	29.93%	0.6183
Scratch-Medium	32.05%	0.6735
Fine-tuned GPT-2	38.87%	0.6808

Finding: With sufficient training data, the Scratch-Medium model now shows *higher* accuracy (39.78%) than Scratch-Small (37.4%) on Reuters, and its semantic similarity is also higher (0.74 vs 0.73). This confirms that with proper data scaling, larger embedding dimensions capture richer semantic relationships and improve prediction quality.

4.2 Qualitative Evaluation: Text Generation

I evaluated the models' ability to generate coherent continuations of prompts from both domains.

Prompt: *"The history of the Roman Empire is characterized by…"*

Table 6: Text Generation Comparison

Model	Output
Scratch	...the Roman Catholic Church. In the Church teaching, the Church was considered by God...
Fine-tuned	...the death of the Emperor Augustus in 753. His successor was...

Analysis:

- **Scratch Model**: Exhibits a *hallucination loop*. The model associates "Roman" with the most frequent co-occurrence in its limited training data (Catholic Church) and ignores the historical context of "Empire."
- **Fine-tuned GPT-2**: Produces a *coherent hallucination*. The sentence structure, topic, and entity types (Emperor, Augustus) are appropriate. The factual error (date 753) demonstrates that the model lacks perfect factual recall but maintains logical coherence.

4.2.1 Prediction Confidence Analysis

To better understand model behavior, I analyzed the top-5 token predictions with their confidence scores for a financial prompt. Figure 3 shows the prediction distribution for each model.

Figure 3: Top-5 token predictions with confidence percentages for the prompt 'The company has been losing money, and its stock price is expected to...'. Fine-tuned GPT-2 shows highly concentrated probability mass on a single prediction ('fall' at 60%), while scratch models distribute probability more evenly across multiple semantically relevant options.

Key Insights from Confidence Analysis:

- **Fine-tuned GPT-2** exhibits strong confidence in its top prediction ('fall' at approximately 60%), reflecting its ability to leverage pre-trained knowledge to make decisive contextually appropriate predictions
- **Scratch models** show more distributed prediction patterns with lower confidence in their top predictions (approximately 20% for 'fall'), spreading probability across multiple semantically relevant financial terms ('fall', 'rise', 'drop', 'soar', 'be')
- Despite lower individual prediction confidence, scratch models still capture domain-appropriate vocabulary, demonstrating that they have learned relevant semantic patterns from the training data, though with less certainty than the pre-trained model

5 Analysis and Discussion

5.1 The Scaling Paradox: Resolved

In earlier experiments with limited data, I observed a counter-intuitive result: the larger Scratch-Medium model (512 dimensions, 8 layers) performed *worse* than Scratch-Small (256 dimensions, 6 layers). This was consistent with the **Chinchilla Scaling Laws** (Hoffmann et al., 2022), which establish that:

$$\text{Optimal Tokens} \propto \text{Model Parameters}$$

With our full-scale experiment (**50 million tokens**), I now observe the **expected scaling behavior**: Scratch-Medium achieves perplexity of **22.1** on Reuters, outperforming Scratch-Small's **27.2**. This represents a **19% improvement** and validates that Chinchilla scaling works when data scales proportionally with model parameters.

Key insight: The original "paradox" was not a failure of scaling laws, but a confirmation of them. Our initial dataset was too small for the larger model. With more data, the larger model's additional capacity translates directly to better performance.

5.2 Domain Generalization: Capacity Constraints

While the domain gap has been substantially reduced compared to data-limited experiments, scratch models still show higher perplexity on WikiText than on Reuters:

- **Scratch-Small (256)**: Reuters 27.2 → WikiText 80.7 (3.0× increase)
- **Scratch-Medium (512)**: Reuters 22.1 → WikiText 65.0 (2.9× increase)
- **Fine-tuned GPT-2**: Reuters 16.9 → WikiText 28.2 (1.7× increase)

The smaller domain gap for Fine-tuned GPT-2 demonstrates that **world knowledge cannot be learned from scratch** on even 50M tokens. The fine-tuned model, having been pre-trained on diverse web text, maintains strong performance across both domains because it encodes broad linguistic and factual knowledge that our training data cannot replicate.

Capacity constraint observation: The Scratch-Small model (256 dimensions) shows a slightly larger domain gap ratio (3.0×) than Scratch-Medium (2.9×). This suggests that the smaller embedding dimension limits the model's ability to capture the diverse vocabulary and semantic relationships needed for cross-domain generalization.

5.3 Transfer Learning Advantage

The fine-tuned GPT-2 model benefits from:

1. **Pre-trained representations**: Word embeddings capturing semantic relationships learned from billions of tokens
2. **Syntactic patterns**: Understanding of English grammar structures

3. **World knowledge**: Factual information encoded in model weights

Fine-tuning adapts these general capabilities to the specific domain with a low learning rate (5e-5), preventing catastrophic forgetting while acquiring domain-specific patterns.

5.4 The Role of Regularization

Early experiments without dropout showed validation perplexity exploding after epoch 3. Adding dropout ($p = 0.1$) to attention weights and feed-forward layers stabilized training, improving final perplexity by approximately 15%.

This confirms that regularization is essential for training Transformers, though its relative importance decreases as dataset size increases.

6 Challenges and Failed Approaches

This section documents technical challenges encountered and how they were resolved, as well as approaches that did not work.

6.1 Challenge 1: The "Double Shifting" Bug

Problem: HuggingFace models internally shift labels during loss computation. When the user provides `labels`, the model computes:

```
loss = CrossEntropy(logits[:, :-1, :], labels[:, 1:])
```

Our initial training loop manually created shifted targets (`y = x[1:]`), then passed them as labels. This caused a *double shift* where the model tried to predict token $t + 2$ from token t.

Symptom: Fine-tuned model showed extremely high perplexity despite correct architecture.

Solution: Conditional logic in `train.py`: - For **Scratch models**: Pass `(x, y)` where `y = x[1:]` - For **HuggingFace models**: Pass `(x, labels=x)` letting the library handle shifting

6.2 Challenge 2: Scaling Without More Data (Resolved in Full-Scale Experiment)

Hypothesis: A larger model would capture more complex patterns and improve performance.

Initial Experiment (Small Scale, ~3M tokens): Doubled embedding dimension (256 $\rightarrow$ 512) and increased layers (6 $\rightarrow$ 8).

Result: Performance *degraded* (PPL 75 $\rightarrow$ 88).

Conclusion: This approach failed without proportionally scaling the training data.

Full-Scale Experiment (50M tokens): I repeated this experiment with more data. The Scratch-Medium model now achieves **lower perplexity** than Scratch-Small (22.1 vs 27.2), confirming that data scaling resolves the overfitting problem. This validates the Chinchilla Scaling Laws in practice.

7 Original Contribution

The original contributions of this work encompass two complementary approaches to next-word prediction:

7.1 1. From-Scratch Transformer Implementation

I implemented a complete **Decoder-only Transformer** architecture without relying on pre-built attention layers (e.g., `torch.nn.MultiheadAttention`). Our implementation includes:

- **Multi-Head Self-Attention with Causal Masking**: Custom implementation of the scaled dot-product attention mechanism with a lower-triangular causal mask ensuring autoregressive generation (position t can only attend to positions $\leq t$)
- **Position-wise Feed-Forward Networks**: Two-layer MLPs with GELU activation and $4\times$ hidden dimension expansion
- **Pre-LayerNorm Architecture**: Layer normalization is applied before each sub-layer (Pre-LN), which improves training stability
- **Weight Tying**: Shared parameters between the token embedding layer and the output projection layer
- **Learned Positional Embeddings**: Trainable position embeddings up to the maximum context length

The correctness of our causal masking implementation was verified through a unit test that confirms future tokens do not leak information to past positions.

7.2 2. Transfer Learning Pipeline with GPT-2 Fine-tuning

I developed a complete transfer learning pipeline that:

- Loads the pre-trained GPT-2 (124M) model from HuggingFace
- Implements domain-specific fine-tuning with carefully tuned hyperparameters (low learning rate of 5e-5 to prevent catastrophic forgetting)
- Uses linear learning rate warmup scheduling for stable adaptation
- Handles the HuggingFace label-shifting convention correctly

8 Conclusions

1. **Transfer learning is the dominant paradigm** for practical NLP applications. A pre-trained model fine-tuned for 4 epochs outperforms a custom model trained for 20 epochs by a factor of 1.3-2.3× in perplexity.

2. **Model scaling requires data scaling**. With sufficient data (50M tokens), increasing model capacity leads to improved performance. The "Scaling Paradox" observed in data-limited settings is resolved.

3. **Scratch models improve with more data but cannot match pre-trained models.** When evaluated outside their training domain, scratch models show significant performance degradation that persists even with large-scale training data.

4. **Architecture is secondary to data**. The "knowledge" in language models comes primarily from the pre-training corpus, not the fine-tuning architecture.

9 References

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). **Attention Is All You Need**. *Advances in Neural Information Processing Systems*, 30.

2. Karpathy, A. (2022). **NanoGPT**. GitHub Repository. Retrieved from https://github.com/karpathy/nanoGPT

3. HuggingFace Transformers. (2023). **GPT-2 Model Documentation**. Retrieved from https://huggingface.co/docs/transformers/model_doc/gpt2

4. PyTorch. (2023). **nn.Module Source Code**. GitHub Repository. Retrieved from https://github.com/pytorch/pytorch/blob/main/torch/nn/modules/module.py

5. Hoffmann, J., et al. (2022). **Training Compute-Optimal Large Language Models** (Chinchilla). *arXiv preprint arXiv:2203.15556*.