FIRA: A Transformer-Based Language Model with Mixture-of-Experts for Enhanced Capacity and Efficiency

M.M. Tahmeed Thoky (Co-CEO and Founder)
Magentabits

M.D. Ibne Jihan (CEO and Founder)
Magentabits

Abstract

This paper introduces FIRA, a transformer-based language model designed to enhance model capacity and computational efficiency through the integration of a Mixture-of-Experts (MoE) layer. Unlike traditional transformer models that rely on static feed-forward networks, FIRA employs a dynamic MoE layer that selectively activates a subset of expert sub-networks for each input, enabling efficient scaling. We detail FIRA's architecture, provide a mathematical formulation of its components focusing on the MoE layer and analyze its workflow during training and inference. Furthermore, we compare FIRA with models like GPT-2 (124M), Google Gemma (2B), and LLaMA (7B), highlighting the advantages of its MoE-based approach for large-scale language modeling tasks.

Note: This model has not yet been fully evaluated on its benchmark due to hardware and computational resource limitations.

1 Introduction

The transformer architecture has revolutionized large language models (LLMs), powering models like GPT-2, LLaMA, and Google Gemma. However, scaling these models to handle increasingly complex tasks often results in significant computational overhead. To address this challenge, we present FIRA, a transformer-based LLM that incorporates a Mixture-of-Experts (MoE) layer. By dynamically routing inputs to specialized expert networks, FIRA achieves greater capacity without a proportional increase in computation. This paper explores FIRA's architecture, workflow, and compares it with other prominent models.

2 Architecture of FIRA

FIRA builds upon the transformer decoder framework, augmenting it with an MoE layer. Its key components are:

- Token and Position Embeddings
- Transformer Blocks with Multi-Head Self-Attention (MHSA) and MoE
- Layer Normalization
- Language Modeling Head

2.1 Token and Position Embeddings

FIRA maps input tokens and their positions into a shared embedding space:

- **Token Embedding**: $\mathbf{E}_{\text{token}} \in \mathbb{R}^{\text{vocab\_size} \times d_{\text{model}}}$
- **Position Embedding**: $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{\text{max\_seq\_len} \times d_{\text{model}}}$

The input embedding is computed as:

$$\mathbf{x} = \mathbf{E}_{\text{token}}(\text{input\_ids}) + \mathbf{E}_{\text{pos}}(\text{pos\_ids})$$

2.2 Transformer Blocks

Each transformer block comprises:

- Layer Normalization (LN1)
- Multi-Head Self-Attention (MHSA)
- Layer Normalization (LN2)
- Mixture-of-Experts (MoE) Layer
- Dropout

The forward pass is:

$$\mathbf{x} \leftarrow \mathbf{x} + \text{Dropout}(\text{MHSA}(\text{LN1}(\mathbf{x})))$$

$$\mathbf{x} \leftarrow \mathbf{x} + \text{Dropout}(\text{MoE}(\text{LN2}(\mathbf{x})))$$

2.2.1 Multi-Head Self-Attention (MHSA)

MHSA computes attention across multiple heads:

- **Query, Key, Value Projections**:

$$\mathbf{q} = \mathbf{W}_q\mathbf{x}, \quad \mathbf{k} = \mathbf{W}_k\mathbf{x}, \quad \mathbf{v} = \mathbf{W}_v\mathbf{x}$$

  where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$.

- **Attention Scores**:

$$\text{att} = \frac{\mathbf{q}\mathbf{k}^T}{\sqrt{d_{\text{head}}}}$$

- **Masked Softmax**: For causal attention, a mask is applied:

$$\text{att} = \text{softmax}(\text{att} + \text{mask})$$

- **Output**:

$$\text{out} = \text{att}\mathbf{v}, \quad \text{out} = \mathbf{W}_o\text{out}$$

  where $\mathbf{W}_o \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$.

2.2.2 Mixture-of-Experts (MoE) Layer

The MoE layer replaces the traditional feed-forward network, introducing dynamic routing:

- **Experts**: Each expert is a feed-forward network:

$$\text{expert}_i(\mathbf{x}) = \mathbf{W}_{i,2} \cdot \text{GELU}(\mathbf{W}_{i,1}\mathbf{x} + \mathbf{b}_{i,1}) + \mathbf{b}_{i,2}$$

  where $\mathbf{W}_{i,1} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$, $\mathbf{W}_{i,2} \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}$.

- **Gating Mechanism**:
  - Compute gate logits:

$$\mathbf{g} = \frac{\mathbf{W}_g\mathbf{x}}{\tau}$$

    where $\mathbf{W}_g \in \mathbb{R}^{d_{\text{model}} \times \text{num\_experts}}$, τ is the gate temperature.
  - Select top-k experts:

$$\mathbf{g}_{\text{topk}} = \text{topk}(\mathbf{g}, k)$$

  - Compute gate probabilities:

$$\mathbf{p} = \text{softmax}(\mathbf{g}_{\text{topk}})$$

- **Weighted Output**:

$$\text{output} = \sum_{i=1}^{\text{num\_experts}} p_i \cdot \text{expert}_i(\mathbf{x})$$

- **Load Balancing Loss**: Ensures even expert utilization:

$$\mathbf{a} = \frac{1}{B \cdot T} \sum_{b=1}^{B} \sum_{t=1}^{T} \mathbf{p}_{b,t}$$

$$L_{\text{balance}} = \left(\mathbf{a} - \frac{1}{\text{num\_experts}}\right)^2 \cdot \lambda$$

2.3 Layer Normalization and Language Modeling Head

- **Layer Normalization**:

$$\text{LN}(\mathbf{x}) = \frac{\mathbf{x} - \mu}{\sigma} \cdot \gamma + \beta$$

- **Language Modeling Head**:

$$\text{logits} = \mathbf{W}_{\text{lm}} \cdot \text{LN}_f(\mathbf{x})$$

  where $\mathbf{W}_{\text{lm}} \in \mathbb{R}^{d_{\text{model}} \times \text{vocab\_size}}$.

3 Workflow

FIRA's workflow during training and inference is illustrated in Figure 1.The diagram has been adjusted to a vertical layout to fit within the page.

The detailed steps are:

1. **Input Processing**: Tokenize input text into token IDs and generate position IDs.
2. **Embedding**: Compute $\mathbf{x} = \mathbf{E}_{\text{token}}(\text{input\_ids}) + \mathbf{E}_{\text{pos}}(\text{pos\_ids})$.
3. **Transformer Blocks**: For each block:
   - Apply LN1 and MHSA with residual connection.
   - Apply LN2 and MoE with residual connection.
4. **Final Normalization**: Apply LN_f to the output.
5. **Language Modeling**: Compute logits via lm_head.
6. **Training**: Minimize cross-entropy loss plus L_{balance}.
7. **Inference**: Generate text using the generate method with sampling.

4 Architecture Diagram

Figure 2 provides a visual representation of FIRA's architecture, highlighting the integration of the MoE layer within the transformer blocks.

5 Comparison with Other Models

FIRA's architecture, particularly its use of the Mixture-of-Experts (MoE) layer, offers several advantages over traditional transformer models. Below, we analyze FIRA's use cases and explain why it can be better than other models like GPT-2 (124M), Google Gemma (2B), and LLaMA (7B).

5.1 Use Cases of FIRA

FIRA is designed for tasks that benefit from large-scale language modeling but require efficient computation. Its MoE layer allows it to handle complex tasks with fewer active parameters per input, making it ideal for:

- **Large-scale text generation**: FIRA can generate coherent and contextually relevant text while using fewer computational resources compared to models with similar parameter counts.
- **Fine-tuning on specialized tasks**: The MoE layer enables FIRA to adapt to specific domains by routing inputs to relevant experts, improving performance on niche tasks.
- **Efficient inference**: By activating only a subset of experts per input, FIRA reduces the computational cost during inference, making it suitable for real-time applications.
- **Scaling to larger models**: FIRA's architecture allows for scaling to billions of parameters without a proportional increase in computation, making it a cost-effective solution for large models.

5.2 Why FIRA is Better

FIRA’s key innovation is the MoE layer, which provides several advantages over traditional transformer models:

- **Dynamic Expert Routing**: Unlike static feed-forward networks in models like GPT-2, FIRA’s MoE layer dynamically selects a subset of experts for each input. This allows FIRA to leverage specialized knowledge from different experts, improving its ability to handle diverse inputs.
- **Efficient Computation**: By activating only a few experts per input, FIRA reduces the computational cost compared to models like Google Gemma (2B) and LLaMA (7B), which use all parameters for every input. This makes FIRA more efficient, especially for large models.
- **Scalability**: FIRA’s MoE architecture allows it to scale to larger parameter counts without a linear increase in computation. This is particularly beneficial for tasks that require massive models but are constrained by hardware limitations.
- **Load Balancing**: The load balancing loss ensures that all experts are utilized evenly, preventing any single expert from becoming a bottleneck. This improves training stability and model performance.

In summary, FIRA’s MoE layer makes it more efficient, scalable, and adaptable than traditional transformer models, particularly for tasks that require large-scale language modeling with limited computational resources.

Model	Parameters	Architecture	Scaling Approach	Use Case
FIRA	334.74M	Transformer with MoE	Dynamic expert routing	Efficient language modeling
GPT-2 (124M)	124M	Transformer decoder	Static FFN	Text generation
Google Gemma (2B)	2B	Transformer-based	Traditional scaling	Multimodal tasks
LLaMA (7B)	7B	Optimized transformer	Efficient training	Research and development

Table 1: Comparison of FIRA with other language models.

5.3 GPT-2 (124M)

- **Architecture**: Transformer decoder with MHSA and static FFN.
- **Differences**: FIRA uses MoE instead of FFN, enabling dynamic routing and greater capacity. GPT-2 lacks load balancing mechanisms.

5.4 Google Gemma (2B)

- **Architecture**: Transformer-based model.
- **Differences**: FIRA’s MoE layer provides efficient scaling, potentially outperforming traditional methods in Google Gemma’s design.

5.5 LLaMA (7B)

- **Architecture**: Optimized transformer decoder.

- **Differences**: FIRA's MoE offers an alternative scaling strategy, potentially reducing computational overhead compared to LLaMA's optimizations.

6 Mathematical Formulation of MoE

The MoE layer is defined as:

$$\text{output} = \sum_{i=1}^{\text{num\_experts}} p_i \cdot \text{expert}_i(\mathbf{x})$$

where:

$$\mathbf{p} = \text{softmax}(\text{topk}(\frac{\mathbf{W}_g \mathbf{x}}{\tau}, k))$$

The load balancing loss is:

$$L_{\text{balance}} = \lambda \cdot \sum_{i=1}^{\text{num\_experts}} \left(a_i - \frac{1}{\text{num\_experts}} \right)^2$$

where a_i is the average gating probability for expert i.

7 Conclusion

FIRA enhances the transformer architecture with a Mixture-of-Experts layer, enabling dynamic expert routing that boosts capacity and efficiency. Compared to models like GPT-2 (124M), Google Gemma (2B), and LLaMA (7B), FIRA's MoE approach offers a novel scaling paradigm, making it a compelling choice for advanced language modeling tasks.

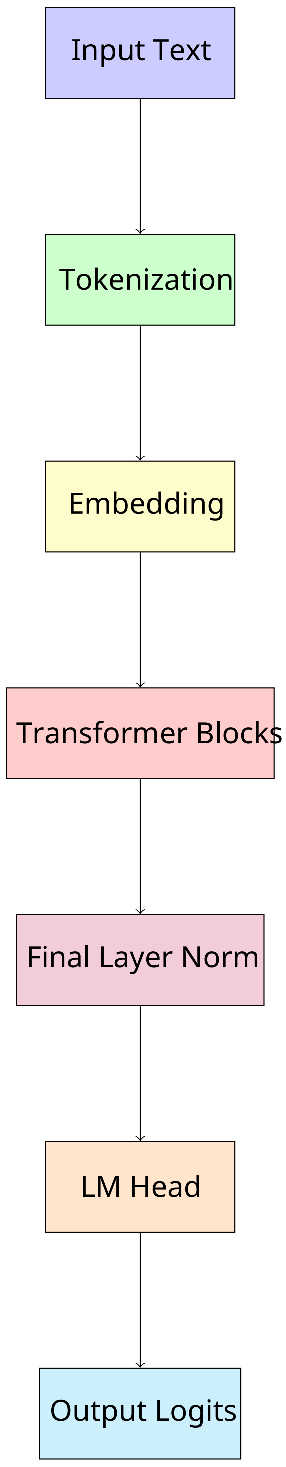

Figure 1: Workflow of FIRA during training and inference (vertical layout).

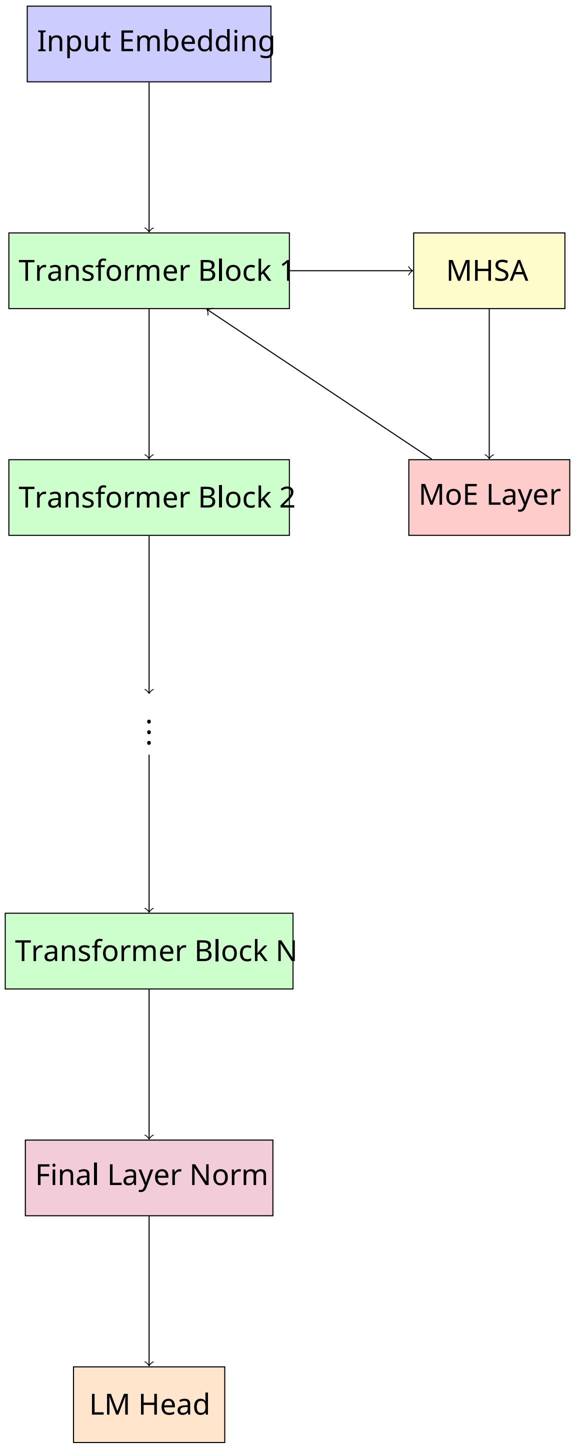

Figure 2: Architecture of FIRA, showing the transformer blocks with MHSA and MoE layers.