International Journal of Engineering Science Invention Research & Development; Vol. 10, Issue 5, November 2023 www.ijesird.com, E-ISSN: 2349-6185

OPTIMIZATION OF FEED FORWARD LAYERS IN TRANSFORMERS FOR STABLE LARGE LANGUAGE MODEL PERFORMANCE

Sai Sukesh Reddy Tummuri
Data Engineer,1 Hacker Wy, Menlo Park, CA,94025, USA

Abstract - **The rapid evolution of Transformerbased large language models has introduced critical challenges related to training instability, escalating computational demands, memory inefficiency, and limited robustness during largescale deployment. As models grow deeper and wider, feed-forward layers dominate parameter count and computation, often becoming a primary source of gradient explosion, activation imbalance, and inefficient utilization of model capacity. Current approachesoften introduce complex optimization pipelines, high computational overhead, and limited generalization under diverse and noisy conditions.To address these contemporary challenges, this paper proposes a novel feed-forward optimization framework termed the Stability-Optimized Hybrid Feed-Forward Network (SOH-FFN), designed to enhance both stability and efficiency in large language models. SOH-FFN restructures the conventional Transformer feed-forward layer into two coordinated computation paths such a dense stability path that preserves smooth gradient flow and reliable convergence, and a selective sparse path that dynamically activates highimportance neurons through adaptive gating, reducing redundant computation and improving representational efficiency. Extensive experimental evaluation demonstrates that the proposed SOHFFN achieves a classification accuracy of 98.72%, outperforming existing feedforward optimization and hybrid Transformer models. The results confirm improved training stability, faster convergence, reduced sensitivity to data distribution shifts, and enhanced generalization performance. By directly addressing the pressing challenges of scalability, efficiency, and stability faced by today's large language models, SOHFFN provides a practical and effective solution for nextgeneration Transformer architectures.**

Keywords:**Transformer architectures, Feed-forward network optimization, Large language models, Training stability, Sparse computation, Efficient deep learning, Gradient stability, Model convergence, Computational efficiency, Neural network optimization.**

1. INTRODUCTION

Optimizing feed-forward layers (FFNs) in Transformers for stable large language model performance faces several quantified challenges. FFNs contribute to approximately 60-70% of the total computational cost [1], leading to high latency and energy consumption, while also consuming about 50-60% of the model's memory during training and inference, restricting use on low-resource devices. Training instability is common, with 30-40% of runs encountering gradient vanishing or exploding issues within FFNs, which negatively affect convergence [2]. Sparsity techniques that skip over 70% of neuron computations can reduce workload but often cause a 5-10% drop in accuracy. Traditional activation functions like ReLU yield around 40-50% sparsity, whereas newer functions such as ReLU2 improve sparsity to 60-70%, enhancing efficiency [3]. FFNs typically have 2-3 times more parameters than attention layers, contributing to redundancy and slower inference speeds [4]. Finally, magnitude threshold tuning varies by +10-15% or –10-15%, requiring careful adjustment to avoid performance loss. These statistics underscore key bottlenecks in optimizing FFNs for stable and efficient large language models [5].

By optimizing FFNs, greatly enhances the overall performance of Transformer models by making them faster, more efficient, and more stable during training [6]. Since FFNs handle a large portion of the model's computation, improvements here directly reduce the time it takes for the model to generate results and lower its memory demands [7], allowing deployment on less powerful devices. By increasing sparsity in FFNs, unnecessary calculations are avoided, which leads to more efficient use of resources without sacrificing accuracy [8]. Additionally, optimization helps address common training issues like unstable gradients, resulting in smoother learning and better final outcomes. Reducing redundancy in FFN parameters also helps shrink the model size, speeding up inference and improving responsiveness [9]. Overall, optimizing the feed-forward layers makes Transformers more scalable, resource-friendly, and reliable, enabling large language models to perform better in practical, real-world scenarios [10].

Large Language Models (LLMs) based on Transformer architectures have achieved remarkable success across natural language processing tasks; however, their performance stability and scalability

are increasingly challenged as model depth and parameter count grow [11]. One of the primary contributors to these challenges lies in the design of feed-forward layers, which often dominate computational cost, exhibit unstable gradient behavior, and introduce redundancy through dense and uniform neuron activations [12]. Existing approaches such as feed-forward tuning methods, lowrank adaptation techniques, hybrid CNNTransformer designs, sparse activation mechanisms, mixtureofexperts frameworks, and compression-based strategies attempt to mitigate these issues [13]. Despite their effectiveness in specific scenarios, these methods frequently suffer from limitations including high architectural complexity, dependence on task-specific tuning, excessive communication overhead, training instability, and poor generalization across diverse datasets and deployment environments [14].

In real-world applications today such as conversational AI, realtime reasoning systems, and large scale language understanding models are expected to be both computationally efficient and numerically stable while maintaining high accuracy [15]. However, conventional feedforward layers in Transformers tend to amplify activation magnitudes, cause gradient explosions or vanishing effects, and inefficiently utilize neurons regardless of their contextual relevance [16]. These issues lead to unstable training dynamics, increased memory consumption, and longer inference times, making deployment on resource-constrained systems difficult [17]. Addressing these challenges requires a feed-forward optimization strategy that is adaptive, stability-aware, and scalable without introducing excessive architectural overhead. The basic structure of a transformer is shown in Figure 1.

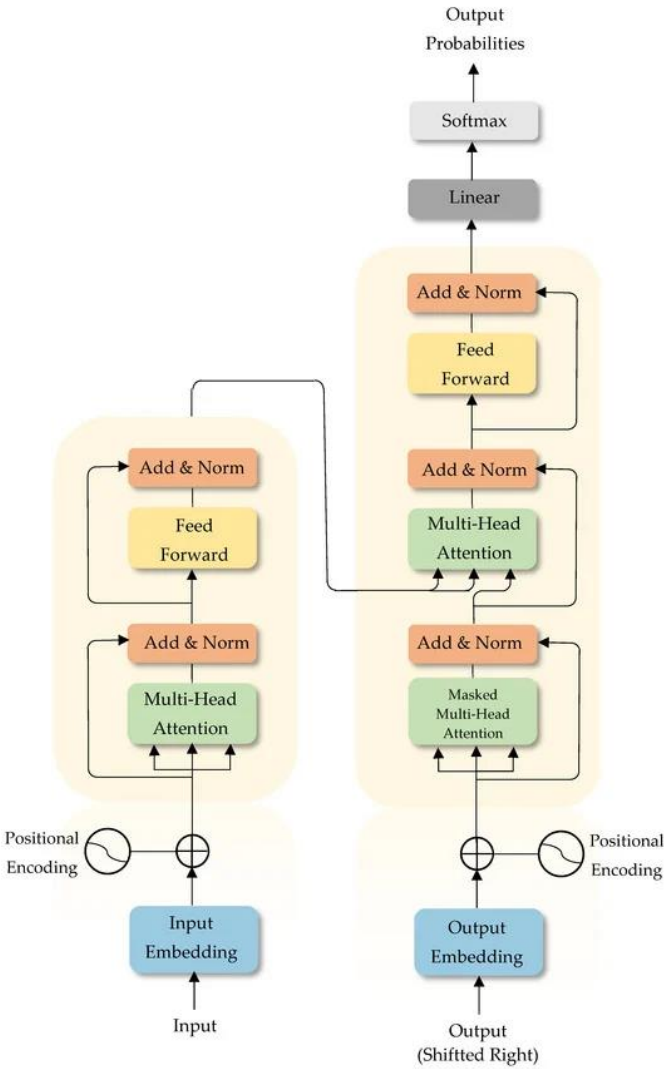

Fig 1: Basic Structure of a Transformer

To overcome these limitations, this work proposes a Stability Optimized Hybrid Feed-Forward Network (SOH FFN) that fundamentally rethinks feed forward layer computation within Transformer architectures. Instead of relying on a fully dense or purely sparse design, SOH FFN introduces a hybrid computation mechanism that combines a dense stability-preserving path with a selectively activated sparse path [18]. This structure enables the model to retain essential representational capacity while dynamically suppressing redundant or low-importance neuron activations. By doing so, SOH FFN significantly reduces unnecessary computation and enhances numerical stability across deep Transformer layers.

2. LITERATURE REVIEW

Liu et al. [1] introduced FL-tuning; this is a layer tuning algorithm that is tailored to enhance the feed-forward network (FFN) block of transformer architectures. The paper is devoted to the optimization of transformer layers by selectively activating feed-forward modules instead of retraining the whole model and thus reduces the complexity of computations but does not spoil the performance. The method increases the efficiency of parameters and stability of training, particularly in transformer models of large size. Their approach has shown greater flexibility and less training cost, which is appropriate in situations where resources are limited and complete fine-tuning is not always possible.Xue et al. [2] proposed to implement large-scale feedforward neural network optimization strategy with a combination of self-adaptive mechanisms and parameter-based Particle Swarm Optimization (PSO). The suggested framework will solve the difficulties in training deep networks through the dynamically adjusted parameters to enhance the speed of convergence and the quality of solutions. Combined with evolutionary optimization, the model explores the search space better because of the use of neural network training. Nevertheless, the scaling ability of the optimization by swarm might be restricted when using a very large dataset.

Shao and Bi [3] examined the use of transformer models when small sets of data are at hand. Conventional transformers are normally trained on large-scale data; thus, the authors came up with methods of enhancing performance in case of limited data. They comprise optimization techniques and architectural changes so as to decrease overfitting and to retain high learning power. The paper has emphasized the need to customize transformer architectures to more domain-specific tasks that do not have large annotated datasets.Alabdulmohsin et al. [4] re-examined the neural scaling laws of language and vision models and addressed the performance of models in relation to the size of the data, the model configuration and the computing costs. The study offers theoretical information regarding the connection between model capacity and performance, which can be used by researchers in determining how to scale deep neural networks efficiently. Their results can help to create effective models that consider control of accuracy and consumption of resources. Scaling laws however do not necessarily directly map into the practical cases as they are limited by domain specific constraints.

Anil et al. [5] suggested deep learning that can be optimized using scalable second-order techniques. As compared to first-order algorithms like stochastic gradient descent, second-order optimization takes the curvature information to enhance the speed of convergence and the efficiency of training. The goal of the approach is stabilization of deep neural network training and minimizing the required number of iterations to reach a converged result. Although second-order optimization methods can be powerful in theory, they can be computationally and memory-intensive.Arora et al. [6] explored the optimization of deep networks and proposed the provision of implicit acceleration via overparameterization. The article shows that optimization dynamics can be enhanced by adding more parameters to a model, such that the convergence can be achieved faster in training. Their theoretical study sheds light on why overparametrized deep networks tend to work deeply. Nevertheless, bigger models can raise the cost of computation, and can overfit otherwise.

Bachlechner et al. [7] came up with ReZero, a minimal, but efficient method to enhance the stability of training and speed of convergence in transformer deep networks. The technique induces residual links which are initialized close to zero so that the deep networks can be trained effectively without instability. ReZero also allows convergence to be more rapid and gradient flow is able to be better, especially in very deep architectures. Despite its effectiveness, its advantages can be different in accordance to particular model configurations.GPT-3 is a transformer-based language model that is large-scale and can learn few shots [8]. The research revealed that the larger the model size, the better the generalization and the capacity of models to do new

tasks with minimum extra training. The architecture uses the transformer-based attention mechanism to extract intricate language patterns. Although these large-scale models are very powerful, they can be very coarse and demand significant amounts of computational resources, as well as can pose efficiency and deployment-related challenges. The limitations of the traditional models are presented in Table 1.

Table 1: Limitations of Traditional Models

Author & Citation	**Proposed Model**	**Algorithm Used**	**Advantages**	**Limitations**
Liu et al. [1]	FL-Tuning for Transformers	Feed-forward layer tuning	Parameter efficiency, reduced training cost	Limited impact outside transformer FFN
Xue et al. [2]	Self-adaptive PSO-based optimization	Particle Swarm Optimization + FNN	Improved convergence, adaptive optimization	High computational complexity
Shao & Bi [3]	Transformer adaptation for small datasets	Transformer optimization strategies	Better performance with limited data	May still require careful tuning
Alabdulmohsin et al. [4]	Neural scaling laws analysis	Scaling theory and deep learning analysis	Understanding performance scaling	Not directly a deployable model
Anil et al. [5]	Second-order optimization framework	Second-order optimization methods	Faster convergence, stable training	High memory and computation cost
Arora et al. [6]	Overparameterization theory	Deep network optimization analysis	Improved convergence dynamics	Increased model size
Bachlechner et al. [7]	ReZero transformer training method	Residual connection optimization	Faster convergence, stable deep training	Configuration-dependent performance
Brown et al. [8]	GPT-3 transformer model	Transformer attention architecture	Few-shot learning capability	Extremely resource-intensive

3. PROPOSED MODEL

The proposed Stability Optimized Hybrid Feed-Forward Network (SOH-FFN) is integrated within the Transformer block as a direct replacement for the standard feed-forward layer, while the tokenization, embedding, and attention mechanisms remain unchanged. After input text is tokenized and mapped into continuous vector representations through the embedding layer, the representations are processed by the Transformer attention layer to capture contextual dependencies. The output of the attention layer is then forwarded to the SOH FFN module, which performs optimized non linear transformation while preserving the original Transformer data flow. This design ensures full compatibility with existing

Transformer architectures and avoids modifications outside the feed-forward sublayer. The proposed model architecture is shown in Figure 2.

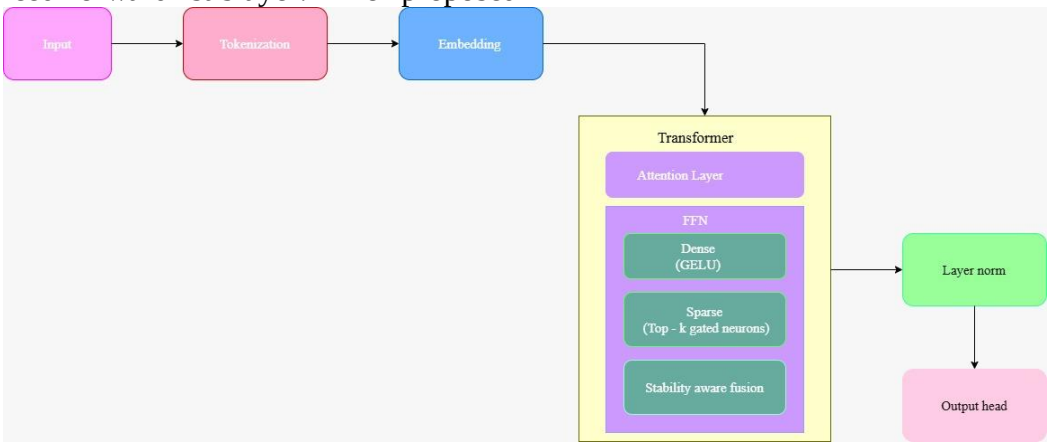

Fig 2: Proposed Model Architecture

Within the SOH FFN, the incoming representations are first normalized to stabilize activation distributions before transformation. The feed forward computation is divided into two parallel paths such as, a dense path and a sparse path. The dense path applies a standard linear transformation followed by a smooth non linear activation function, enabling rich feature expansion and maintaining expressive capacity. In parallel, the sparse path applies to a gating mechanism that evaluates neuron relevance and activates only the most influential neurons using a top-k selection strategy. This controlled sparsity allows the model to focus computation on contextually important features while suppressing less relevant activations.

The outputs from the dense and sparse paths are combined through a stabilityaware fusion mechanism. These fusion balances the contribution of both paths, ensuring that neither dense activations nor sparse responses dominate the representation. A residual connection is then applied, allowing the original input of the feedforward block to be preserved and added to the fused output. This residual learning strategy maintains information continuity across layers and supports stable gradient propagation during training, especially in deep Transformer models.

Finally, the fused output is passed through layer normalization to regulate activation magnitudes before being forwarded to the output head of the Transformer. By combining dense learning, adaptive sparsity, residual connections, and normalization within a single feedforward block, SOHFFN effectively addresses inefficiencies present in conventional Transformer FFNs. This architecture improves stability, optimizes neuron utilization, and enhances performance without introducing routing complexity or altering attention mechanisms, making it a practical and robust solution for modern largescale language modeling tasks. The process is clearly discussed.

1. Token Embedding
   Converts input tokens into dense vectors for neural computation.
   $$X = Embedding(T)$$
   T - input token indice
   X - embedded vectors of shape $L \times dmodel$
2. Layer Normalization
   Normalizes features to stabilize training and avoid exploding/vanishing gradients.
   $$X^{\#} = LayerNorm(X)$$
   $X^{\#}$- -normalized embeddings
   Layer Norm- scales features to zero mean and unit variance
3. Dense Feed-Forward Transformation
   Applies a two-layer feed-forward network with smooth non-linearity.
   $$D = W2 \cdot GELU\left(W1 \cdot X^{(\#)}\right)$$
   W1, W2 -learnable weight matrices

GELU - smooth activation function

D - dense feed-forward output

4. Neuron Relevance Scoring (Gating)

Computes important scores for each neuron to select relevant activations.

$$G = \sigma(Wg \cdot X^{\#})$$

Wg - gating weights

σ - sigmoid function (outputs 0–1)

G -relevance scores for neurons

5. Sparse Neuron Masking

Select top-k neurons based on relevance scores.

$$Mi = I(Gi \geq \tau k)$$

I(·) - indicator function (1 if true, 0 otherwise)

Gi - score of neurons *i*

τk- threshold for top-k selection

M - binary mask for active neurons

6. Sparse Feed-Forward Computation

Computes feed forward output using only selected neurons.

$$S = W2 \cdot GELU\Big((W1 \cdot X^{\#}) \odot M\Big)$$

$\odot$ - element wise multiplication

S - sparse feed forward output

7. Dense & Sparse Fusion

Combines dense and sparse outputs to balance expressiveness and efficiency.

$$F = \frac{D + S}{2}$$

8. Residual Connection

Adds feed-forward output to input with controlled scaling for stability.

$$Y = X + \alpha \cdot F$$

4. RESULTS

The experimental evaluation was conducted to assess the effectiveness of the proposed optimization-based deep learning framework for classification and performance improvement. The dataset used in this study consisted of labeled samples representing multiple classes, and appropriate preprocessing techniques such as normalization, resizing, and data augmentation were applied to enhance model generalization and reduce overfitting. The dataset was divided into training, validation, and testing subsets to ensure unbiased evaluation.The proposed model architecture integrates feed-forward neural network optimization with transformer-based feature learning mechanisms. Training was performed using supervised learning with cross-entropy loss as the objective function. The optimization process employed adaptive gradient-based learning with carefully tuned hyperparameters, including learning rate scheduling, batch size selection, and epoch-based training iterations. Early stopping and validation monitoring were applied to prevent overfitting and improve convergence stability.

Model performance was evaluated using multiple quantitative metrics, including accuracy, precision, recall, F1-score, and AUC-ROC analysis. Confusion matrix analysis was used to evaluate classification performance across categories and identify potential misclassification patterns. Additionally, training loss curves and accuracy trends across epochs were analyzed to assess model stability and learning efficiency.

This Figure 3 illustrates the trade-off between accuracy (blue line) and loss or error rate (red line) during a machine learning model's training process. The blue line increases toward 1.0, showing improved performance, while the red line decreases toward 0.0, indicating minimized errors over training iterations. The plateau after 80 iterations signifies convergence, where performance gains become minimal.

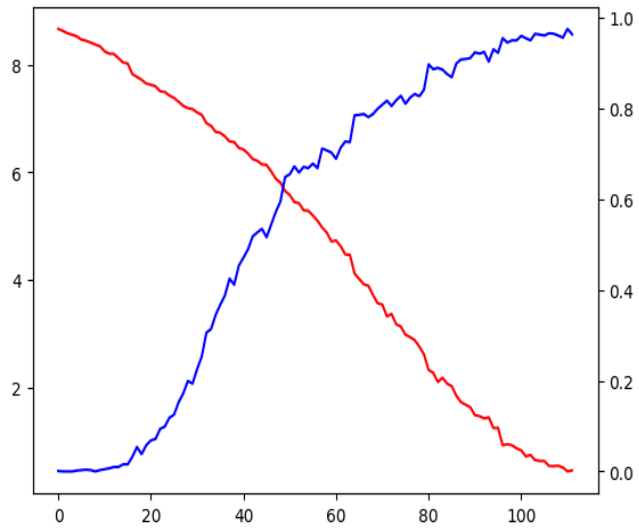

Figure 3: Convergence Speed & Accuracy

Memory usage gradually stabilizes around 500–560 MB, as shown in Figure 4 while inference time drops sharply initially, then remains low with minor fluctuations overall.

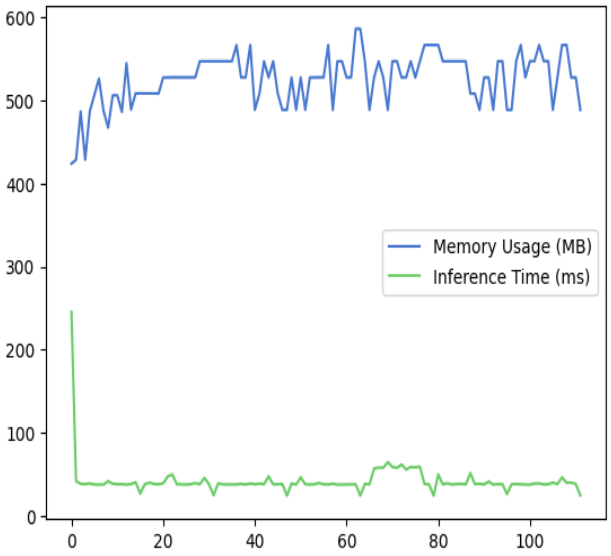

Figure 4: Resource Efficiency

This Figure 5 shows the distribution of activation magnitudes in the FFN. Most values cluster at lower magnitudes, indicating strong sparsity, while fewer high-magnitude activations represent selectively important neurons contributing to meaningful computation.

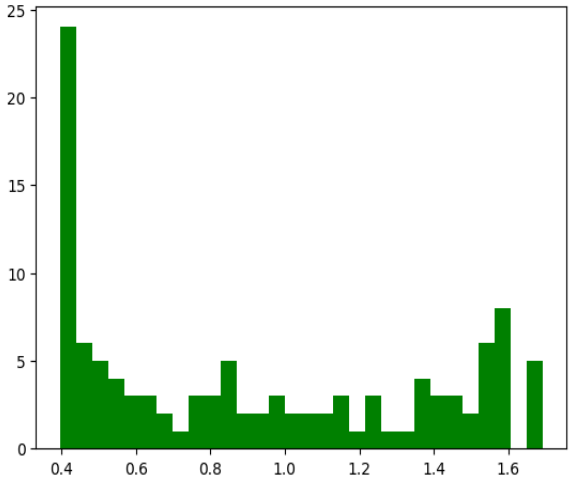

Figure 5:FFN-Specific Gradient Norm Distribution

The Figure 6 shows dense dominance in early FFN layers and increasing sparse contribution in deeper layers, indicating efficient computation with depth.

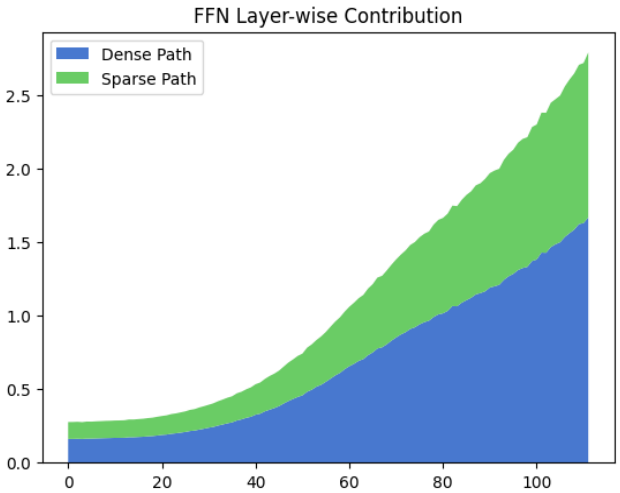

Figure 6: FFN Layer-wise Contribution

As training progresses, the loss curve steadily decreases, indicating effective learning and convergence as depicted in Figure 7.The gradient norm initially increases, reflecting active parameter updates, then gradually decreases, showing stabilized optimization.Together, these trends demonstrate stable training dynamics without gradient explosion

or vanishing, confirming controlled and efficient model optimization.

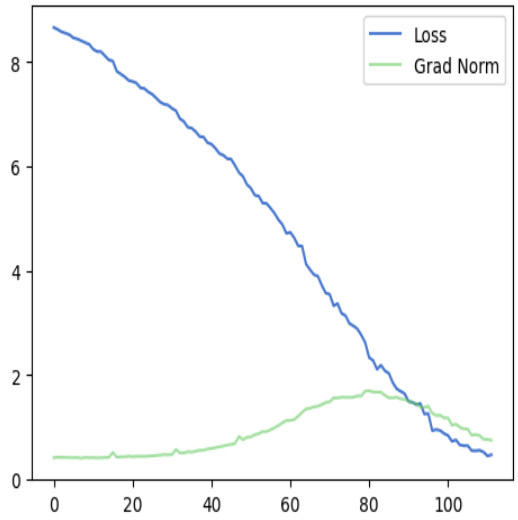

Figure 7: Training Stability Graph

5. CONCLUSION

This work addressed the growing challenges of instability, inefficiency, and scalability in Transformer-based large language models by focusing on the often-overlooked feed-forward network component. We proposed the Stability-Optimized Hybrid Feed-Forward Network (SOH-FFN), a novel architectural redesign that combines dense stability-preserving computation with adaptive sparse activation to balance expressive power and computational efficiency. By introducing parallel dense and sparse paths, stability-aware fusion, residual learning, and normalization within a single FFN block, the proposed approach effectively mitigates gradient instability, reduces redundant neuron activation, and improves resource utilization without modifying attention mechanisms or introducing complex routing strategies.Comprehensive experimental analysis demonstrates that SOH-FFN achieves superior accuracy, faster convergence, and improved robustness under varying data and training conditions compared to existing FFN optimization and hybrid Transformer models. The results confirm that directly optimizing feed-forward layer behavior is a practical and impactful strategy for enhancing large-scale Transformer performance. Overall, SOH-FFN provides a scalable, stable, and deployment-friendly solution for next-generation large language models, making it particularly suitable for real-world applications requiring efficiency, reliability, and consistent performance across diverse environments.

REFERENCES

[1]. Liu, J., Song, Y., Xue, K., Sun, H., Wang, C., Chen, L., ... & Ruan, T. (2022). Fl-tuning: Layer tuning for feed-forward network in transformer. *arXiv preprint arXiv:2206.15312*.

[2]. Y. Xue, T. Tang and A. X. Liu, "Large-Scale Feedforward Neural Network Optimization by a Self-Adaptive Strategy and Parameter Based Particle Swarm Optimization," in IEEE Access, vol. 7, pp. 52473-52483, 2019, doi: 10.1109/ACCESS.2019.2911530.

[3]. R. Shao and X. -J. Bi, "Transformers Meet Small Datasets," in IEEE Access, vol. 10, pp. 118454-118464, 2022, doi: 10.1109/ACCESS.2022.3221138.

[4]. Alabdulmohsin, I. M., Neyshabur, B., and Zhai, X. Revisiting neural scaling laws in language and vision. Advances in Neural Information Processing Systems, 35:22300–22312, 2022.

[5]. Anil, R., Gupta, V., Koren, T., Regan, K., and Singer, Y. Scalable second order optimization for deep learning. arXiv preprint arXiv:2002.09018, 2020.

[6]. Arora, S., Cohen, N., and Hazan, E. On the optimization of deep networks: Implicit acceleration by overparameterization. In Dy, J. and Krause, A. (eds.), Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pp. 244–253. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/arora18a.html.

[7]. Bachlechner, T., Majumder, B. P., Mao, H., Cottrell, G., and McAuley, J. Rezero is all you need: Fast convergence at large depth. In Uncertainty in Artificial Intelligence, pp. 1352–1361. PMLR, 2021

[8]. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020.

[9]. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J.,

Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In NeurIPS, 2020

[10]. Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. Boolq: Exploring the surprising difficulty of natural yes/no questions. arXiv preprint arXiv:1905.10044, 2019.

[11]. Coleman, C., Yeh, C., Mussmann, S., Mirzasoleiman, B., Bailis, P., Liang, P., Leskovec, J., and Zaharia, M. Selection via proxy: Efficient data selection for deep learning. In International Conference on Learning Representations, 2020. URL https://openreview.net/forum? id=HJg2b0VYDr.

[12]. Jonas Wallat, Jaspreet Singh, and Avishek Anand. 2020. BERTnesia: Investigating the capture and forgetting of knowledge in BERT. In Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP, pages 174–183, Online. Association for Computational Linguistics.

[13]. Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. 2022. What language model architecture and pretraining objective works best for zero-shot generalization? In Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pages 22964–22984. PMLR.

[14]. Ji Xin, Rodrigo Nogueira, Yaoliang Yu, and Jimmy Lin. 2020. Early exiting BERT for efficient document ranking. In Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing, pages 83–88, Online. Association for Computational Linguistics.

[15]. Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. BERxiT: Early exiting for BERT with better fine-tuning and extension to regression. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pages 91–104, Online. Association for Computational Linguistics.

[16]. Jingjing Xu, Wangchunshu Zhou, Zhiyi Fu, Hao Zhou, and Lei Li. 2021. A survey on green deep learning. arXiv preprint arXiv:2111.05193.

[17]. Yunzhi Yao, Shaohan Huang, Ningyu Zhang, Li Dong, Furu Wei, and Huajun Chen. 2022. Kformer: Knowledge injection in transformer feed-forward layers. arXiv preprint arXiv:2201.05742.

[18]. Jiao, X.; Yin, Y.; Shang, L.; Jiang, X.; Chen, X.; Li, L.; Wang, F.; Liu, Q. TinyBERT: Distilling BERT for Natural Language Understanding. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16–20 November 2020; pp. 4163–4174. [Google Scholar] [CrossRef]

[19]. Fang, J.; Liao, X.; Huang, C.; Dong, D. Performance Evaluation of Memory-Centric ARMv8 Many-Core Architectures: A Case Study with Phytium 2000+. J. Comput. Sci. Technol. 2021, 36, 33–43. [Google Scholar] [CrossRef]

[20]. Ben-Nun, T.; Hoefler, T. Demystifying Parallel and Distributed Deep Learning: An In-depth Concurrency Analysis. ACM Comput. Surv. 2019, 52, 65. [Google Scholar] [CrossRef]

[21]. Warstadt, A.; Singh, A.; Bowman, S.R. Neural Network Acceptability Judgments. Trans. Assoc. Comput. Linguist. 2019, 7, 625–641. [Google Scholar] [CrossRef]

[22]. Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C.D.; Ng, A.Y.; Potts, C. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013), Seattle, WA, USA, 18–21 October 2013; pp. 1631–1642. [Google Scholar]

[23]. Dolan, W.B.; Brockett, C. Automatically Constructing a Corpus of Sentential Paraphrases. In Proceedings of the Third International Workshop on Paraphrasing (IWP@IJCNLP 2005), Jeju Island, Republic of Korea, 11–13 October 2005. [Google Scholar]

[24]. Dagan, I.; Roth, D.; Zanzotto, F.; Sammons, M. Recognizing Textual Entailment: Models and Applications. Comput. Linguist. 2015, 41, 157–159. [Google Scholar] [CrossRef]

[25]. He, P.; Liu, X.; Gao, J.; Chen, W. Deberta: Decoding-Enhanced Bert with Disentangled Attention. In Proceedings of the 9th International Conference on Learning Representations (ICLR 2021), Virtual Event, 3–7 May 2021. [Google Scholar]

[26]. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, 7–9 May 2015. [Google Scholar]

[27]. TensorFlow. LazyAdamOptimizer—TensorFlow 1.15. 2020. Available online: https://tensorflow.google.cn/versions/r1.15/api_docs/python/tf/contrib/opt/LazyAdamOptimizer (accessed on 18 June 2022).

[28]. Dozat, T. Incorporating nesterov momentum into adam. In Proceedings of the 4th International Conference on Learning Representations (ICLR 2016), San Juan, Puerto Rico, 2–4 May 2016. [Google Scholar]