Comparative Analysis of Modular, Sequential, and Parallel Neural Architectures on Inference Accuracy and Latency

Aryan Kulkarni

aryanamol10@gmail.com

Bellarmine Symposium on STEM and Society (2026) • Extended Abstract

1. Introduction

1.1 Background

Large language models (LLMs) and neural networks more broadly have become foundational tools across scientific and commercial domains. As datasets grow in volume and complexity, the choice of architectural framework increasingly determines a model's ability to extract meaningful patterns within acceptable computational budgets. While the literature covers individual architectures, direct comparisons under controlled conditions—holding data, hardware, and optimizer fixed—remain relatively sparse.

A motivating example is the binary classification of speaker loudness. Given plotted data points for large and small speakers, an LLM must scan each data point, recognize patterns in increase and decrease, and output a confident prediction. Figure 1 illustrates this simple linear scenario, where the model outputs a high-confidence prediction (Large: 0.95, Small: 0.05) from a linear decision boundary.

Figure 1: Illustrative example of binary classification using a linear decision boundary. The LLM identifies speaker size (large vs. small) from loudness-vs-instances scatter data.

As data complexity increases for additional categories, overlapping distributions, or nonlinear boundaries, the model must produce multiple ranked predictions. Figure 2 shows this scenario, where three simultaneous predictions (Prediction 1: 0.56, Prediction 2: 0.39, Prediction N: 0.05) are required, and the line of best fit curves to accommodate the more complex structure.

Figure 2: Multi-class prediction scenario with a curved decision boundary. Increased data complexity requires the LLM to maintain multiple ranked probability estimates simultaneously.

The central challenge is the latency-accuracy trade-off, where layer-by-layer processing introduces substantial latency and can cause increased likelihood of error. Architectural decisions about how computation is organized fundamentally affect both the speed and accuracy of inference.

1.2 Problem

This paper investigates three neural network paradigms: (1) sequential (hierarchical layer stacking), (2) parallel (dual-branch processing), and (3) modular (decomposition via neural chains). How do these architectural paradigms compare in final regression accuracy (MSE) and training time across tasks of increasing nonlinear complexity, when all other training variables are held constant?

2. Investigation

2.1 Methodology

To isolate the effect of architecture, all experimental variables except network structure were held constant. Three models were evaluated across three synthetic regression environments, each defined by a distinct generating function of escalating complexity. Common parameters: training samples (N) = 5,000; neurons per layer = 64; optimizer = Adam (default learning rate); training epochs = 100; hardware = A100/L4 GPUs (identical across all runs).

Three synthetic data environments were generated over the domain $X \in [0, 10]$ with N = 5,000 uniformly spaced samples. Environment 1—Linear: $y = 2X + \varepsilon$, $\varepsilon \sim N(0, 1)$, a standard baseline. Environment 2—Curved: $y = 0.5X^2 + \sin(X) + \varepsilon$, $\varepsilon \sim N(0, 2)$, combining polynomial and sinusoidal components. Environment 3—Chained: y = X

$\cos(0.5X) + \sin X \cos(0.5X) + 0.5\ \mathrm{sgn}(\sin(X)) + \varepsilon$, $\varepsilon \sim N(0, 0.5)$, a multi-stage nonlinear function designed to test subtask decomposition. Figure 3 shows representative scatter plots for each environment.

Figure 3: Scatter plots of the three synthetic regression datasets. Left: Linear ($y = 2X + \varepsilon$). Center: Curved ($y = 0.5X^2 + \sin(X) + \varepsilon$). Right: Chained (multi-stage nonlinear composition with noise).

Sequential model: Implemented in Keras/TensorFlow. Fully connected feed-forward network with decreasing layer widths ($64 \rightarrow 32 \rightarrow 16 \rightarrow 1$).

Parallel model: Implemented in TensorFlow. Dual-branch architecture, each branch receiving the full input; outputs concatenated before a final linear projection.

Modular model: Implemented in PyTorch. Chain of three sub-networks, each receiving the output of the previous stage, targeting a different challenge of the regression problem.

Two metrics were recorded for each model-environment pair: (1) final training MSE at epoch 100, and (2) total wall-clock training time in seconds. Both metrics were collected across two identical independent runs to verify consistency.

2.2 Results

Figure 4 presents the MSE training curves across all three environments and models over 100 epochs. All models exhibit rapid initial loss reduction followed by stabilization, though the rate of descent and final plateau differ meaningfully across architectures. In the linear environment, the modular model (green) descends to the lowest plateau, while sequential and parallel converge to similar but higher loss values. In the curved environment, a similar pattern holds. In the chain environment, the ranking inverts: the sequential model (blue) converges to the lowest final loss ($\sim 10.4 \times 10^1$), while the modular model converges more slowly to a marginally higher peak ($\sim 10.6 \times 10^1$).

Figure 4: MSE training loss curves over 100 epochs for Sequential (blue), Parallel (orange), and Modular (green) architectures across Linear (left), Curved (center), and Chain (right) environments.

Table 1 reports the final MSE and training time for each model-environment combination.

Table 1. Final MSE and Training Time (s) by Model and Environment

Model	Linear		Curved		Chain	
	MSE	Time (s)	MSE	Time (s)	MSE	Time (s)
Sequential	29.38	37.80	29.46	239.64	30.08	10.39
Parallel	30.31	37.83	30.93	240.53	30.38	10.44
Modular	18.67	38.16	17.72	245.68	18.60	10.55

Note: Lower MSE indicates higher accuracy.

The modular model achieved the best accuracy in two of three environments. In the linear case, it reduced MSE to 18.67—approximately 37% lower than sequential (29.38) and parallel (30.31). In the curved case, it achieved an MSE of 17.72 versus 29.46 (sequential) and 30.93 (parallel)—again roughly 40% lower. The exception is the chain environment, where the modular model's MSE (18.60) was the highest of the three despite training for the longest time (10.55 s). The sequential model achieved its best relative outcome in the chain environment: MSE of 30.08 at 10.39 seconds—the fastest training time across that scenario. The parallel model demonstrated consistent but middling performance across all environments, with the smallest variance in both MSE and training time.

3. Conclusion

3.1 Related Work

Jacobs et al. [1] introduced the Mixture-of-Experts paradigm, which motivates modular decomposition by showing that dedicated sub-networks can specialize effectively on distinct subtasks. LeCun et al. [3] demonstrated that hierarchical feed-forward architectures suffer from error propagation across layers—a limitation the sequential model exhibits in our curved environment. Szegedy et al. [5] showed that parallel multi-branch architectures improve representational efficiency in deep networks; our parallel model's consistency across environments supports this finding, though it does not achieve peak accuracy. Our work differs by directly comparing all three paradigms under identical training conditions on synthetic regression tasks of

controlled complexity, isolating the architectural variable.

3.2 Limitations

All experiments used synthetic data with known generating functions; generalization to real-world datasets with unknown structure is not guaranteed. Modular subtask definitions were hand-specified, and an automated decomposition strategy might produce different results. All models were evaluated at a fixed scale (64 neurons, 5,000 samples); architectural rankings may shift at different parameter counts or dataset sizes.

3.3 Societal Impact

This research benefits practitioners who must select neural architectures under compute constraints—particularly in resource-limited settings such as edge devices or under-resourced research labs. By providing concrete accuracy-latency trade-off data, it enables more informed decisions that reduce unnecessary computation and energy use. The primary risk of harm is over-generalization: practitioners may apply these findings to domains (e.g., image classification, NLP) where the synthetic regression results do not transfer, leading to poorly performing or inefficient deployed models. The findings should be treated as directional guidance rather than definitive architectural prescriptions.

4. References and Additional Details

4.1 References

[1] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, “Adaptive mixtures of local experts,” Neural Computation, vol. 3, no. 1, pp. 79–87, 1991.

[2] A. Kulkarni, “Neural architecture comparison experiments [Source code],” GitHub, 2025. [Online]. Available: https://github.com/aryanamol10/Research-Paper-Comparative-Analysis-of-Neural-Models

[3] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” Neural Computation, vol. 1, no. 4, pp. 541–551, 1989.

[4] A. Stöffelbauer, “How large language models work,” Data Science at Microsoft, Medium, Oct. 24, 2023. [Online]. Available: https://medium.com/data-science-at-microsoft/how-large-language-models-work-91c362f5b78f

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1–9.

4.2 Disclosures & Acknowledgements

AI Use: [Disclose any AI use for brainstorming, editing, or code assistance per paper guidelines if applicable.]

Mentor/Funding Acknowledgements: [Add acknowledgements during camera-ready.]