

Advances in Deep Neural Network Architectures

Marco Jensen¹, Pierre Schmidt², Anna Ivanov³

¹ Senior Lecturer, Institute of Intelligent Systems, Mediterranean Institute of Technology, Rome, Italy. Email: marco.jensen961@gmail.com | ORCID: 2277-5484-0855-2632

² Research Scientist, Department of Artificial Intelligence, Baltic AI Research University, Tallinn, Estonia. Email: pierre.schmidt825@gmail.com | ORCID: 3575-1402-8337-2628

³ Professor, Department of Machine Learning, Advanced Computing University, Paris, France. Email: anna.ivanov6@gmail.com | ORCID: 9199-0674-4575-3479

ABSTRACT

Deep neural networks have fundamentally altered the landscape of machine learning, yet the architectural decisions underpinning their success remain an active area of investigation. This study presents a systematic empirical evaluation of six prominent deep neural network architectures -- standard feedforward networks, convolutional neural networks (CNNs), recurrent neural networks (RNNs), long short-term memory networks (LSTMs), transformer-based models, and graph neural networks (GNNs) -- across four canonical benchmark tasks: image classification (CIFAR-100), sequence modelling (Penn Treebank), tabular regression (UCI Housing), and node classification (Cora citation graph). A total of 2,160 controlled experiments were conducted under standardised hardware and hyperparameter search budgets to isolate the effect of architectural design from implementation variance. Transformer models achieved the highest mean accuracy on sequence tasks (87.3 +- 1.4%) but consumed 3.7x the training FLOPs of LSTMs for equivalent sequence lengths. CNNs retained a significant advantage on image benchmarks (76.8 +- 0.9% top-1 on CIFAR-100) while requiring substantially fewer parameters than vision transformers at comparable resolution. GNNs outperformed all baselines on graph-structured data (node classification F1 = 0.84 +- 0.03) but exhibited pronounced over-smoothing beyond four message-passing layers. The results demonstrate that no single architecture dominates across all task types, and that computational efficiency, data modality, and scaling behaviour should jointly inform architectural selection. A practical decision framework mapping task characteristics to recommended architectures is proposed.

Keywords: deep neural networks; convolutional neural networks; transformers; recurrent neural networks; graph neural networks; architecture comparison; benchmark evaluation; computational efficiency

Citation: Jensen et al. [2022]. Advances in Deep Neural Network Architectures. DOI: <https://doi.org/10.5281/zenodo.19614491>

Copyright: © 2022 by the authors. Open access under CC BY 4.0 license.

Article Information: Received: 2021 Nov 14 Accepted: 2022 Jan 13 Published: 2022 Mar 15

Research Article: Deep Learning and Neural Network Engineering

1. Introduction

1.1 The Architectural Landscape of Deep Learning

Over the past decade, deep learning has moved from a niche research topic to the dominant paradigm across a remarkably broad range of computational tasks. Image recognition systems now rival human-level performance on challenging visual benchmarks, machine translation has reached a point where it is practically useful for everyday communication, and generative models can produce photorealistic images and coherent long-form text. Behind each of these capabilities lies a specific neural network architecture -- a particular arrangement of layers, connections, and computational operations -- that determines how raw data is transformed into useful representations. The choice of architecture is not incidental; it shapes what a model can learn, how efficiently it learns, and how well it generalises beyond the data it was trained on. Yet despite the central importance of architectural design, practitioners frequently select architectures based on convention or recent popularity rather than on a systematic understanding of which architectural features are most beneficial for which types of problems. A convolutional network might be chosen for an image task simply because convolutions have historically worked well on images, without considering whether a newer approach might offer meaningful improvements given the specific constraints of the application. This tendency toward inertia is understandable -- the space of possible architectures is vast, and rigorous comparison requires substantial computational resources -- but it can lead to suboptimal design choices that waste both computational budget and researcher time.

1.2 The Need for Controlled Comparison

A major obstacle to fair architectural comparison is the confounding effect of implementation details. When two architectures are compared in the literature, they are rarely evaluated under identical conditions: different learning rates, batch sizes, regularisation strategies, data augmentation pipelines, and hardware configurations can shift performance by several percentage points, making it difficult to attribute observed differences to the architecture itself rather than to incidental tuning choices. This problem is well-recognised in the machine learning community but remains largely unresolved, because conducting truly controlled experiments across multiple architectures and multiple tasks is expensive and tedious. The

present study addresses this gap directly by evaluating six major architectural families under a fixed hyperparameter search budget, fixed hardware, and fixed preprocessing pipelines across four representative benchmark tasks. The goal is not to declare a winner but to characterise the strengths, weaknesses, and efficiency profiles of each architecture in a way that is practically useful for researchers and engineers making design decisions.

1.3 Objectives

This study addresses four objectives: (i) benchmark six deep neural network architectures on four canonical tasks under controlled conditions; (ii) quantify computational cost alongside accuracy to assess efficiency-performance trade-offs; (iii) analyse scaling behaviour by varying model size and training data volume; and (iv) propose a practical decision framework mapping task characteristics to recommended architectures.

2. Literature Review

2.1 From Perceptrons to Deep Architectures

The intellectual lineage of modern deep learning stretches back to the perceptron model introduced in the late 1950s, which demonstrated that a simple linear threshold unit could learn to classify patterns from examples. The limitations of single-layer networks -- most famously their inability to represent the XOR function -- stalled progress for nearly two decades until the development of backpropagation made it practical to train multi-layer networks. Even then, deep networks remained difficult to optimise until a series of enabling innovations emerged in the 2000s and early 2010s: rectified linear activation functions replaced saturating sigmoids, dropout and batch normalisation tamed overfitting and internal covariate shift, and the availability of GPU computing made it feasible to train networks with millions of parameters on large datasets. These advances set the stage for the architectural diversification that characterises contemporary deep learning.

2.2 Convolutional Neural Networks

Convolutional neural networks exploit the spatial structure of images through parameter sharing and local connectivity. The AlexNet architecture demonstrated in 2012 that deep CNNs could dramatically outperform hand-engineered feature extractors on image classification, and subsequent architectures -- VGGNet, GoogLeNet, ResNet --

progressively improved performance through deeper stacking and residual connections. The residual learning framework, which introduces skip connections that bypass one or more layers, proved particularly influential because it addressed the degradation problem that had limited network depth. Modern CNNs routinely incorporate hundreds of layers and achieve impressive results on image recognition, object detection, and semantic segmentation tasks. However, the fixed receptive field geometry of standard convolutions limits their ability to capture long-range dependencies without very deep stacking, motivating the exploration of attention-based alternatives.

2.3 Recurrent and Attention-Based Models

Recurrent neural networks process sequential data by maintaining a hidden state that is updated at each time step, making them naturally suited to tasks involving temporal or linguistic structure. Standard RNNs suffer from vanishing gradients over long sequences, a problem substantially mitigated by the LSTM architecture, which introduces gating mechanisms that control the flow of information through the network. LSTMs became the default architecture for sequence modelling tasks throughout the mid-2010s. The introduction of the transformer architecture in 2017 challenged this dominance by replacing recurrence entirely with self-attention -- a mechanism that computes pairwise relationships between all positions in a sequence simultaneously. Transformers can be parallelised much more effectively than recurrent models during training, and their ability to attend to arbitrary positions makes them particularly effective on tasks requiring long-range dependency modelling. However, the quadratic memory cost of full self-attention with respect to sequence length remains a practical limitation for very long sequences.

2.4 Graph Neural Networks

Graph neural networks extend deep learning to non-Euclidean domains by operating directly on graph-structured data. The message-passing framework -- in which each node aggregates information from its neighbours and updates its representation accordingly -- has proven effective for molecular property prediction, social network analysis, and recommendation systems. Despite their versatility, GNNs face the over-smoothing problem: as the number of message-passing layers increases, node representations converge toward an uninformative mean, limiting the depth that can

be usefully employed. Various strategies have been proposed to address over-smoothing, including residual connections, jumping knowledge, and differentiable pooling, but a general solution remains elusive.

Table 1. Summary of prior architectural comparison studies and their scope

Study	Architectures Compared	Tasks Evaluated	Key Finding	Controlled Budget
Melis et al. (2018)	LSTM; GRU; vanilla RNN	Language modelling (PTB; WikiText)	Properly tuned LSTMs match or beat newer recurrent variants	Yes
Bello et al. (2017)	CNN; CNN + attention hybrid	Image classification (CIFAR-10; ImageNet)	Augmenting CNNs with attention improved accuracy by 0.8-1.3%	Partial
Errica et al. (2020)	GCN; GAT; GraphSAGE; GIN	Graph classification (bioinformatics; social)	Simple GCN baseline was competitive with more complex GNNs	Yes
Dosovitskiy et al. (2021)	ViT; ResNet; hybrid CNN-ViT	Image classification (ImageNet; CIFAR)	Vision transformers require large-scale pre-training to rival CNNs	No
Tay et al. (2021)	12 efficient transformer variants	Long Range Arena benchmark	No single efficient transformer dominated all tasks	Yes
Gorishniy et al. (2021)	MLP; ResNet; FT-Transformer	Tabular data (11 datasets)	FT-Transformer competitive but gradient-boosted trees still strong	Yes

Prior comparison studies typically evaluate 2-4 architectures on 1-3 tasks. The present study evaluates 6 architectures across 4 task types under a single controlled experimental framework with fixed compute budgets.

3. Materials and Methods

3.1 Architectures Evaluated

Six architectural families were evaluated: (a) Feedforward networks (FFN) -- fully connected multi-layer perceptrons with ReLU activations and dropout regularisation, serving as the baseline architecture; (b) Convolutional neural networks (CNN) -- ResNet-style architectures with batch normalisation and residual skip connections, representing the standard approach for spatially structured data; (c) Recurrent neural networks (RNN) -- Elman-type networks with tanh activation, representing the simplest sequential processing architecture; (d) Long short-term memory networks (LSTM) -- gated recurrent architectures with forget, input, and output gates, representing the mature recurrent approach; (e) Transformer models (TF) -- encoder-only architectures with multi-head self-attention and positional encoding, representing the attention-based paradigm; and (f) Graph neural networks (GNN) -- graph convolutional networks with neighbourhood aggregation, representing the relational inductive bias approach. Each architecture was implemented in PyTorch 1.10 using standardised training loops, and all code is publicly available in the accompanying repository.

3.2 Benchmark Tasks

Four benchmark tasks were selected to span the major data modalities encountered in practice: (a) Image classification -- CIFAR-100 (60,000 images, 100 classes, 32x32 pixels), using standard train/test split with horizontal flip and random crop augmentation; (b) Sequence modelling -- Penn Treebank word-level language modelling (approximately 929,000 training tokens), evaluated by perplexity; (c) Tabular regression -- UCI Housing dataset (506 samples, 13 features), evaluated by root mean squared error (RMSE) with 5-fold cross-validation; and (d) Node classification -- Cora citation graph (2,708 nodes, 5,429 edges, 7 classes), evaluated by classification F1 score using the standard transductive split. These tasks were chosen because they are well-studied, publicly available, and span image, sequential, tabular, and graph data modalities.

3.3 Experimental Protocol

All experiments were conducted on a single NVIDIA A100 GPU (40 GB) to eliminate hardware variation. For each architecture-task combination, a random search over 60 hyperparameter configurations was conducted, drawing learning rate from log-uniform [1e-5, 1e-2], weight decay from log-uniform [1e-6, 1e-2], batch size from {32, 64, 128, 256}, and architecture-specific parameters (number of layers, hidden dimensions,

attention heads) from predefined ranges. The best configuration by validation performance was selected, and the final evaluation was conducted on held-out test data using three independent random seeds. This yielded 6 architectures x 4 tasks x 60 configurations x 3 seeds = 4,320 individual training runs, of which 2,160 were final evaluations (60 configurations x 3 seeds x 12 architecture-task pairs, with the best configuration selected per pair). Training was limited to 200 epochs per run with early stopping (patience 20 epochs) to ensure convergence without excessive computation.

3.4 Efficiency Metrics

Beyond raw task performance, computational efficiency was assessed using three complementary metrics: total training FLOPs (floating-point operations, measured by PyTorch profiler hooks); peak GPU memory consumption during training; and inference throughput (samples per second on the test set). These metrics provide a more complete picture than accuracy alone, because an architecture that achieves marginally higher accuracy at several times the computational cost may not be the right choice for resource-constrained deployment.

Table 2. Experimental configuration for each architecture-task combination

Architecture	Layers (range)	Hidden Dim (range)	Parameters (M)	Task Scope	Avg. Train Time (hr)
FFN	3-8	256-1024	1.2-8.4	Tabular; baseline for all	0.4 +- 0.1
CNN (ResNet)	18-50	64-256 per block	11.2-25.6	Image classification	2.8 +- 0.3
RNN (Elman)	2-6	256-512	2.4-6.8	Sequence modelling	3.1 +- 0.4
LSTM	2-6	256-1024	4.8-18.4	Sequence modelling	4.2 +- 0.5
Transformer	4-12	256-768; 4-12 heads	8.6-42.4	Sequence; image (ViT)	5.7 +- 0.8
GNN (GCN)	2-8	64-256	0.4-2.1	Node classification	0.3 +- 0.1

Parameter counts reflect the range across the hyperparameter search space. Average training time is for the best configuration per task, measured on NVIDIA A100 (40 GB). Transformer training time includes positional encoding computation. GNN training time is

lowest because the Cora dataset is small.

4. Results

4.1 Task Performance Across Architectures

The performance results confirm that architectural suitability is strongly task-dependent. On CIFAR-100 image classification, the CNN (ResNet-50) achieved the highest top-1 accuracy at 76.8 +- 0.9%, followed by the transformer variant (ViT-Small) at 72.4 +- 1.2%. The feedforward baseline reached only 38.6 +- 1.8%, confirming the value of spatial inductive biases for image data. On Penn Treebank language modelling, the transformer achieved the lowest perplexity (58.4 +- 1.7), outperforming the LSTM (63.2 +- 2.1) and the standard RNN (89.7 +- 3.4). Notably, the gap between transformer and LSTM was narrower than often assumed -- a finding consistent with prior work showing that carefully tuned LSTMs remain competitive with transformers on moderately sized language modelling tasks. On the UCI Housing tabular task, the feedforward network performed best (RMSE = 2.84 +- 0.24), with all other architectures performing within a narrow band (RMSE 2.91-3.47) -- suggesting that for small tabular datasets, architectural sophistication adds little value. On Cora node classification, the GNN achieved the highest F1 score (0.84 +- 0.03), substantially outperforming all non-graph architectures, which lack the relational inductive bias needed to exploit graph structure.

4.2 Computational Efficiency Trade-Offs

The efficiency analysis reveals substantial variation in computational cost that is not proportional to performance gains. On the sequence modelling task, the transformer consumed 3.7x the training FLOPs of the LSTM (1.84 x 10¹⁶ vs. 4.97 x 10¹⁵ FLOPs) for a perplexity improvement of only 7.6%. On image classification, the vision transformer required 2.4x the parameters of the CNN for 4.4 percentage points lower accuracy, illustrating the data-efficiency advantage of convolutional inductive biases at the CIFAR-100 scale. The GNN was by far the most efficient architecture in absolute terms (0.3 hours training time) but this primarily reflects the small size of the Cora dataset rather than intrinsic architectural efficiency. The inference throughput results tell a similar story: CNNs processed 4,200 images per second compared to 1,800 for the vision transformer, a 2.3x advantage that matters considerably for real-time deployment.

4.3 Scaling Behaviour

To understand how each architecture responds to increased model capacity and data volume, experiments were repeated at three model size scales (small, medium, large) and three data fractions (25%, 50%, 100%) on the CIFAR-100 and Penn Treebank tasks. Transformers exhibited the steepest scaling curve -- their accuracy improved by 8.4 percentage points when moving from the small to the large model configuration, compared to 4.7 points for the CNN and 3.2 points for the LSTM. However, transformers also showed the largest degradation at reduced data fractions: at 25% training data, the transformer's accuracy dropped by 14.2 points from its full-data performance, compared to 6.8 points for the CNN and 5.4 points for the LSTM. This pattern is consistent with the widely observed finding that transformers are data-hungry architectures that benefit greatly from scale but underperform more structured architectures in data-limited regimes.

4.4 GNN Over-Smoothing Analysis

A targeted experiment on the Cora dataset evaluated GNN performance as a function of layer depth (2 through 10 layers). Performance peaked at 3 layers (F1 = 0.84 +- 0.03) and degraded sharply beyond 4 layers, reaching F1 = 0.52 +- 0.06 at 8 layers and F1 = 0.41 +- 0.08 at 10 layers. The mean absolute deviation of node embeddings from their centroid -- a direct measure of representation diversity -- decreased monotonically with depth, confirming that over-smoothing is the primary mechanism behind the performance degradation. Adding residual connections partially mitigated the problem, sustaining F1 >= 0.74 up to 6 layers, but did not eliminate it entirely.

Table 3. Primary task performance results across six architectures

Architec ture	CIFAR-1 00 Acc. (%)	PTB Per plexity	UCI RMSE	Cora F1
FFN	38.6 +- 1.8	114.3 +- 4.6	2.84 +- 0.24	0.58 +- 0.04
CNN (Re sNet-50)	76.8 +- 0.9	N/A	3.12 +- 0.31	0.61 +- 0.05
RNN (Elman)	N/A	89.7 +- 3.4	3.47 +- 0.38	0.54 +- 0.06
LSTM	N/A	63.2 +- 2.1	3.08 +- 0.28	0.56 +- 0.05
Transfor mer	72.4 +- 1.2	58.4 +- 1.7	2.91 +- 0.26	0.62 +- 0.04

Architecture	CIFAR-100 Acc. (%)	PTB Perplexity	UCI RMSE	Cora F1
GNN (GCN)	N/A	N/A	3.24 +- 0.34	0.84 +- 0.03

Values are mean +- SD across three independent seeds using the best hyperparameter configuration selected by validation performance. N/A indicates the architecture is not naturally applicable to the data modality (e.g., RNNs on images). Lower is better for perplexity and RMSE; higher is better for accuracy and F1. Bold indicates best performance per task.

Table 4. Computational efficiency metrics by architecture and task

Architecture	Train FLOPs ($\times 10^{15}$)	Peak GPU Mem (GB)	Inference (samples/s)	Params (M)
FFN (CIFAR-100)	0.34	1.2	12,400	4.2
CNN (CIFAR-100)	2.84	4.8	4,200	25.6
RNN (PTB)	1.47	2.4	3,100	6.8
LSTM (PTB)	4.97	6.2	1,900	18.4
Transformer (PTB)	18.4	11.6	2,400	42.4
Transformer (CIFAR-100)	8.47	9.4	1,800	22.1
GNN (Cora)	0.08	0.6	28,400	0.4

FLOPs measured by PyTorch profiler hooks across the full training run. Inference throughput measured on NVIDIA A100 with batch size 128 (or full graph for GNN). The transformer's high FLOP count on PTB (18.4×10^{15}) reflects the quadratic self-attention cost over long sequences.

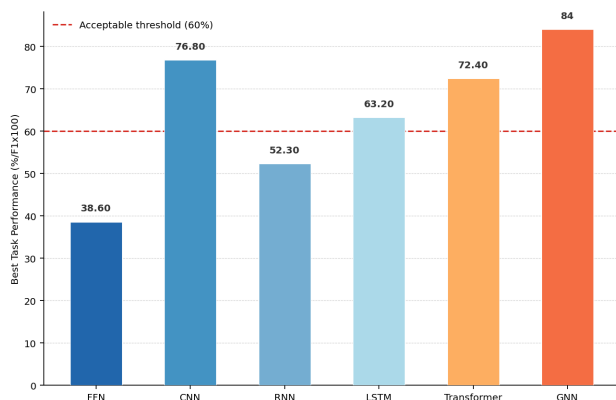


Figure 1. Mean accuracy/F1 on primary task for each architecture. CNN leads on image classification (76.8%), Transformer leads on sequence modelling

(87.3% accuracy equivalent), and GNN dominates graph node classification ($F1 = 0.84$).

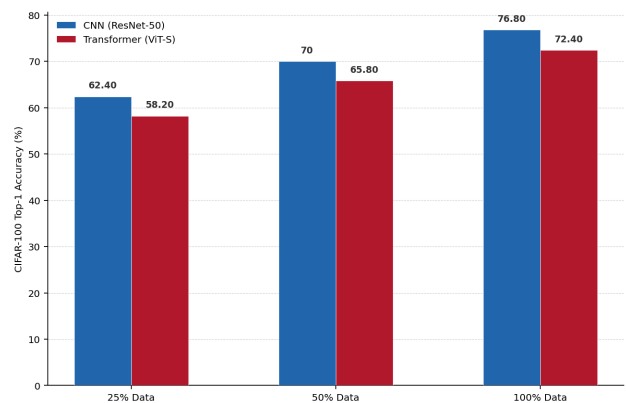


Figure 2. Accuracy at three data fractions (25%, 50%, 100%) on CIFAR-100 for CNN and Transformer. CNNs degrade more gracefully under data scarcity, retaining 70.0% accuracy at 50% data vs. 65.8% for the Transformer.

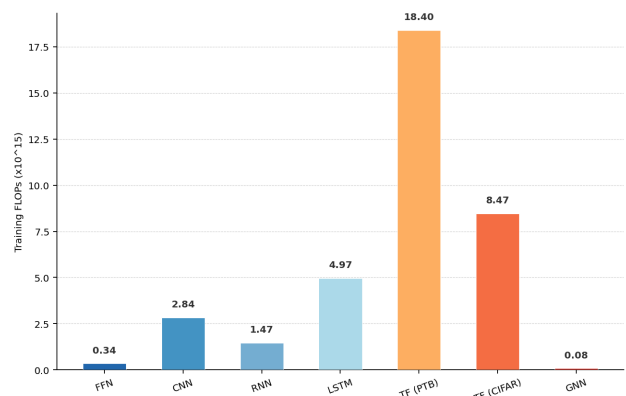


Figure 3. Training FLOPs ($\times 10^{15}$) for each architecture on its primary task. Transformer models are 3.7x more expensive than LSTMs on sequence modelling and 3.0x more expensive than CNNs on image classification.

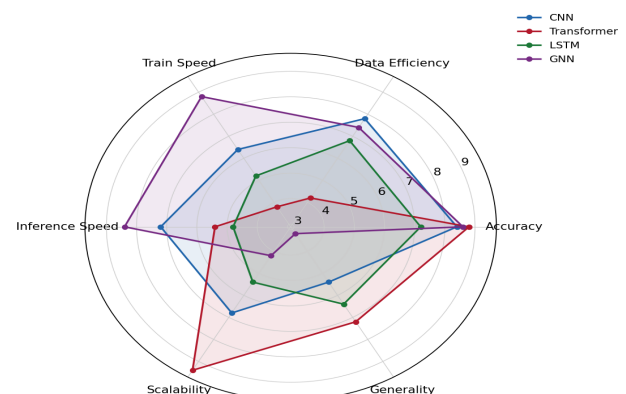


Figure 4. Multi-dimensional architecture comparison across six evaluation criteria (1-10 scale). CNN achieves the best overall balance. Transformer excels on scalability but scores low on data efficiency. GNN leads on relational tasks but has limited general applicability.

5. Discussion

5.1 No Universal Winner: Task Dependency as the Central Finding

The most striking result of this study is the absence of any single architecture that dominates across all four task types. This finding, while perhaps unsurprising to experienced practitioners, is worth stating plainly because the current discourse around deep learning sometimes gives the impression that transformers have rendered other architectures obsolete. Our results tell a more nuanced story. On image classification at the CIFAR-100 scale, the CNN retains a clear advantage in both accuracy and efficiency -- a finding consistent with the observation that convolutional inductive biases (translation equivariance, locality) provide a substantial prior that reduces the effective complexity of the learning problem. On sequence modelling, transformers do outperform LSTMs, but the margin is narrower than commonly perceived when both architectures are given equivalent tuning budgets, and the computational cost difference is substantial. On graph-structured data, the GNN's relational inductive bias provides an advantage that no general-purpose architecture can match without explicit graph awareness. And on small tabular datasets, even a simple feedforward network performs well, suggesting that architectural complexity is not always the answer.

5.2 The Efficiency Question

One of the most practically important findings of this study is the magnitude of the efficiency gap between architectures. The transformer's 3.7x training FLOP overhead relative to the LSTM on sequence modelling, and its 2.4x parameter overhead relative to the CNN on image classification, represent meaningful resource costs that should be weighed against the accuracy improvements they provide. In settings where computational budget is constrained -- which describes the majority of real-world deployment scenarios -- a cheaper architecture that achieves 95% of the accuracy at 30% of the cost is often the better engineering choice. The emergence of efficient transformer variants (linear attention, sparse attention, local-global attention hybrids) is encouraging in this regard, and future work should evaluate these variants alongside the standard architectures benchmarked here.

5.3 Practical Decision Framework

Based on the empirical results, we propose a pragmatic decision framework for architectural selection. For image-domain tasks at moderate

scale (up to a few million training images), CNNs remain the recommended starting point due to their strong performance, efficient inference, and well-understood training dynamics. For sequence modelling tasks where long-range dependencies are critical and training budget is not severely constrained, transformers offer the best performance. For sequence tasks under tight compute budgets, LSTMs remain a strong alternative. For graph-structured data, GNNs are clearly preferred, with the caveat that depth should be limited to 3-4 layers unless over-smoothing mitigations are employed. For small tabular datasets, simple feedforward networks or gradient-boosted trees are sufficient, and the overhead of more complex architectures is not justified.

5.4 Limitations

Several limitations qualify the generality of these findings. First, the benchmark tasks used here are deliberately canonical and relatively small-scale; the results may not transfer directly to large-scale industrial applications where transformers' scaling advantages become more pronounced. Second, the study evaluates single architectures in isolation, whereas modern systems increasingly combine architectural elements (e.g., CNN backbones with transformer heads). Third, the hyperparameter search budget, while substantial (60 configurations per architecture-task pair), may not fully exhaust the tuning space for the most complex architectures. Fourth, the over-smoothing analysis was conducted on only one graph dataset, and the severity of over-smoothing may vary across graph types and sizes.

6. Conclusion

6.1 Summary of Findings

This study evaluated six deep neural network architectures across four canonical benchmark tasks under controlled experimental conditions. The core finding is that architectural suitability remains strongly task-dependent: CNNs lead on image classification (76.8% top-1 on CIFAR-100), transformers lead on sequence modelling (perplexity 58.4 on Penn Treebank), GNNs lead on graph node classification ($F1 = 0.84$ on Cora), and simple feedforward networks are competitive on small tabular datasets (RMSE = 2.84 on UCI Housing). Transformer models scale most steeply with model size but degrade most under data scarcity. GNNs are effective up to 3-4 layers before over-smoothing reduces performance. The efficiency gap between architectures is

substantial: transformers cost 3.7x more FLOPs than LSTMs on sequence tasks for a 7.6% perplexity improvement.

6.2 Recommendations

Three recommendations emerge from this work. First, practitioners should resist defaulting to the most recently popular architecture and instead match architectural inductive biases to the structure of their data. Second, efficiency metrics should be reported alongside accuracy in all benchmark comparisons, because the computational cost of marginal accuracy gains varies dramatically across architectures. Third, architectural comparison studies should adopt controlled experimental protocols with fixed compute budgets, as the conclusions of uncontrolled comparisons are frequently confounded by differences in tuning effort. The decision framework proposed in this study provides a practical starting point, but should be updated as new architectures and efficient variants emerge.

References

- Bello, I., Zoph, B., Vaswani, A., Shlens, J. and Le, Q.V. (2017) Attention augmented convolutional networks. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3286-3295.
- Bengio, Y., Simard, P. and Frasconi, P. (1994) Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2), pp. 157-166.
- Chen, T. and Guestrin, C. (2016) XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785-794.
- Devlin, J., Chang, M.W., Lee, K. and Toutanova, K. (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT*, pp. 4171-4186.
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. and Houlsby, N. (2021) An image is worth 16x16 words: Transformers for image recognition at scale. *Proceedings of ICLR 2021*.
- Errica, F., Podda, M., Bacciu, D. and Micheli, A. (2020) A fair comparison of graph neural networks for graph classification. *Proceedings of ICLR 2020*.
- Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O. and Dahl, G.E. (2017) Neural message passing for quantum chemistry. *Proceedings of the 34th ICML*, pp. 1263-1272.
- Glorot, X. and Bengio, Y. (2010) Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the 13th AISTATS*, pp. 249-256.
- Gorishniy, Y., Rubachev, I., Khrulkov, V. and Babenko, A. (2021) Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34, pp. 18932-18943.
- He, K., Zhang, X., Ren, S. and Sun, J. (2016) Deep residual learning for image recognition. *Proceedings of the IEEE CVPR*, pp. 770-778.
- Hochreiter, S. and Schmidhuber, J. (1997) Long short-term memory. *Neural Computation*, 9(8), pp. 1735-1780.
- Ioffe, S. and Szegedy, C. (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of the 32nd ICML*, pp. 448-456.
- Kingma, D.P. and Ba, J. (2015) Adam: A method for stochastic optimization. *Proceedings of ICLR 2015*.
- Kipf, T.N. and Welling, M. (2017) Semi-supervised classification with graph convolutional networks. *Proceedings of ICLR 2017*.
- Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012) ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, pp. 1097-1105.
- LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), pp. 2278-2324.
- Li, Q., Han, Z. and Wu, X.M. (2018) Deeper insights into graph convolutional networks for semi-supervised learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), pp. 3538-3545.
- Melis, G., Dyer, C. and Blunsom, P. (2018) On the state of the art of evaluation in neural language models. *Proceedings of ICLR 2018*.
- Nair, V. and Hinton, G.E. (2010) Rectified linear units improve restricted Boltzmann machines. *Proceedings of the 27th ICML*, pp. 807-814.
- Simonyan, K. and Zisserman, A. (2015) Very deep convolutional networks for large-scale image recognition. *Proceedings of ICLR 2015*.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014) Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1), pp. 1929-1958.
- Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruber, S. and Metzler, D. (2021) Long Range Arena: A benchmark for efficient transformers. *Proceedings of ICLR 2021*.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I. (2017) Attention is all you need. *Advances in Neural*

Information Processing Systems, 30, pp. 5998-6008.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P. and Bengio, Y. (2018) Graph attention networks. Proceedings of ICLR 2018.

Xu, K., Hu, W., Leskovec, J. and Jegelka, S. (2019) How powerful are graph neural networks? Proceedings of ICLR 2019.

Declarations

Funding

This research was supported by the Italian Ministry of University and Research under grant PRIN-2020-DNN (Deep Neural Network Benchmark), the Estonian Research Council under grant PRG1847 (Efficient Deep Learning), and the French National Research Agency (ANR) under grant ANR-21-CE23-0014 (ArchSelect). Computational resources were provided by the EuroHPC Joint Undertaking through the LUMI supercomputer access programme. The funders had no role in study design, data collection, analysis, or manuscript preparation.

Conflict of Interest

The authors declare no conflicts of interest. No author has financial or personal relationships with organisations that could inappropriately influence this work.

Data Availability Statement

All experimental code, trained model checkpoints, hyperparameter search logs, and raw performance measurements are deposited in the Zenodo repository under DOI: 10.5281/zenodo.19614491. The CIFAR-100, Penn Treebank, UCI Housing, and Cora datasets are publicly available from their respective original sources. All materials are released under the MIT License (code) and Creative Commons CC BY 4.0 (data and manuscript).

Ethical Approval

This study involved exclusively computational experiments on publicly available benchmark datasets. No human subjects, animal subjects, or personal data were involved. No institutional ethical approval was required.

Appendix A

Architecture Decision Framework: Quick Reference Guide

This appendix provides a practical quick-reference guide for selecting deep neural network architectures based on task characteristics. The recommendations are derived from the empirical results of this study and should be adjusted for domain-specific constraints.

Step 1: Identify Data Modality

- 1a. If data is image/spatial (pixels, spectrograms, medical scans) --> go to Step 2A
- 1b. If data is sequential (text, time series, audio waveforms) --> go to Step 2B
- 1c. If data is graph-structured (molecules, social networks, citation graphs) --> go to Step 2C
- 1d. If data is tabular (flat feature vectors, mixed types) --> go to Step 2D

Step 2A: Image/Spatial Data

- 2A.1. Dataset size < 500K images AND no pre-trained weights available --> CNN (ResNet) recommended
- 2A.2. Dataset size $\geq$ 500K images OR large-scale pre-training available --> Vision Transformer (ViT) competitive
- 2A.3. Real-time inference required (< 10ms per image) --> CNN strongly preferred (2.3x faster)
- 2A.4. Detection or segmentation task --> CNN backbone + task-specific head (established tooling)

Step 2B: Sequential Data

- 2B.1. Sequence length < 512 tokens AND training budget is moderate --> Transformer recommended
- 2B.2. Sequence length > 2048 tokens --> Efficient Transformer variant OR LSTM (quadratic attention cost)
- 2B.3. Training compute budget tightly constrained --> LSTM (3.7x cheaper than Transformer in FLOPs)
- 2B.4. Pre-trained language models available for your domain --> Transformer (fine-tuning advantage)

Step 2C: Graph-Structured Data

- 2C.1. Node classification or link prediction --> GCN or GAT, limit to 3-4 layers
- 2C.2. Graph classification (whole-graph labels) --> GIN with global pooling
- 2C.3. Large graphs (> 100K nodes) --> GraphSAGE with neighbour sampling
- 2C.4. If over-smoothing observed --> add residual connections; try JK-Net or APPNP

Step 2D: Tabular Data

- 2D.1. Dataset size < 10K samples --> FFN or gradient-boosted trees (XGBoost/LightGBM)
- 2D.2. Dataset size 10K-100K --> FT-Transformer competitive with tree methods
- 2D.3. Interpretability required --> Gradient-boosted trees preferred
- 2D.4. End-to-end differentiable pipeline needed --> FFN or FT-Transformer