We Can (and Should) Design Neural Networks with a Systematic Dimensional Approach

Anonymous authors
Paper under double-blind review

Abstract

The design of neural network architectures, despite remarkable empirical successes, resembles an architecture zoo characterized by chance innovations and reliance on intuition rather than systematic thinking. This approach limits our ability to deeply understand why architectures succeed, efficiently explore the vast design space, and transfer knowledge across different paradigms. We argue for a shift in how the machine learning community approaches neural architecture design: moving from an architecture-centric cataloging to a dimensional-centric understanding. Building on prior taxonomic work and integrating insights from recent architecture search approaches, we introduce a framework comprising 10 quasi-orthogonal structural dimensions that govern the capabilities of neural networks. This dimensional approach facilitates deeper understanding by enabling the deconstruction of complex architectures into their core design choices and their associated inductive biases. This aims to enable more principled innovation by providing a modern map for systematic exploration of the design space and targeted design for specific problem characteristics. We demonstrate the framework's utility by mapping diverse, prominent architectures onto these dimensions and call upon the community to adopt such systematic frameworks for more principled and efficient advancement in neural network design.

1 Introduction

Neural networks have achieved transformative successes across diverse domains, from computer vision (Krizhevsky et al., 2012) and natural language processing (Vaswani et al., 2017) to scientific discovery (Jumper et al., 2021). This progress has been propelled by a Cambrian explosion of novel architectures, with models like Convolutional Neural Networks (CNNs) (LeCun et al., 1989a), Recurrent Neural Networks (RNNs) with LSTMs (Hochreiter & Schmidhuber, 1997), Transformers (Vaswani et al., 2017), and more recently, Kolmogorov-Arnold Networks (Liu et al., 2025) as a distinct new class of architecture.

However, the process of architectural innovation often resembles a form of modern alchemy or artistry. While frequently brilliant, it tends to rely on individual insight, extensive empirical trial-and-error, and incremental modifications to successful precedents. The field currently lacks a modern, systematic language for articulating and reasoning about core architectural decisions. This neural network architecture zoo phenomenon (Leijnen & Veen, 2020) makes it challenging to transfer knowledge effectively, teach architectural concepts systematically, or ensure comprehensive exploration of the design space. We often treat not only the learned parameters but also the architectures themselves as opaque black boxes, making it difficult to rigorously attribute observed performance gains to specific structural choices or to generalize design principles beyond narrow contexts.

This prevailing ad-hoc approach presents several limitations to sustainable progress:

Inefficient Exploration: The vast space of possible neural architectures is explored somewhat opportunistically, potentially leading to diminishing returns from tweaking dominant paradigms (e.g. the proliferation of transformer variants) while neglecting potentially fruitful, uncharted territories.

Obscured Understanding: Without a foundational framework to dissect and compare architectures, it is difficult to build a cumulative science of why certain structural features confer advantages for particular tasks or data modalities.

Barriers to Education and Communication: The lack of a standardized vocabulary for architectural fundamentals complicates both the education of new researchers and clear communication about architectural innovations.

We argue that for the field to mature and continue its trajectory of impactful innovation, neural network design demands a more systematic, dimensional approach. This paper advocates for a shift: **moving from an architecture-centric view, which often focuses on specific named models, to a dimensional-centric view, which emphasizes the underlying fundamental design choices that define what any neural network can compute and represent.**

Our Contribution: by formalizing implicit design dimensions, we introduce a framework comprised of 10 structural dimensions of neural network design, summarized in Table 1 (details in Section 3). These dimensions are chosen to be quasi-orthogonal and to capture the core architectural decisions a designer makes, explicitly or implicitly. This framework is not intended to be prescriptive of which specific dimensional choices are optimal for a given task, but rather to offer a conceptual map of the architectural design space itself. While individual dimensions have been identified across disparate literature, our novel contribution is their unification into a coherent framework and demonstration of its analytical power. We argue that adopting such a dimensional perspective can catalyze progress by: **(1) enabling deeper, more nuanced understanding of existing and novel architectures; (2) facilitating more principled and efficient innovation through systematic exploration and targeted design; and (3) fostering accelerated, cumulative scientific advancement in the field.**

Table 1: Dimensions of Neural Network Design

Dimension Name	**Brief Definition**	**Example Variations (Illustrative)**
Edge Transformation	How information is transformed as it passes along connections between nodes.	Linear, Nonlinear, Identity
Node Activation	How aggregated inputs are processed at each node after aggregation.	Linear, Nonlinear, Identity/None
Bias Mechanism	How an offset or bias term is determined and applied to shift a node's operating point.	Static Learned, Conditional/Dynamic, Structured
Normalization Strategy	How activations within a layer are rescaled and re-centered.	Batch-wise, Layer-wise, Instance-wise, Group-wise
Aggregation Mechanism	How multiple inputs to a node are combined.	Additive, Order-Sensitive, Statistical, Attentional
Connectivity Pattern	Which nodes connect to which other nodes in the network.	Dense, Sparse, Hierarchical, Dynamic
Information Flow Direction	The overall directionality of information propagation (acyclic vs. cyclic).	Acyclic (Feedforward), Cyclic (Recurrent)
Shortcut Strategy	How direct alternative pathways are established, bypassing intermediate transformations.	No Shortcuts, Additive (Residual), Concatenative, Gated
Memory Structure	How the network retains and accesses information across processing steps.	Stateless, Stateful, External Memory
Parameter Sharing Pattern	How parameters are shared or reused across different parts of the network.	Independent, Shared, Conditional

2 A Dimensional View

A dimensional perspective offers an organizing lens for both reasoning through existing architectures and guiding the creation of novel ones. By building on existing taxonomic approaches to neural networks and moving beyond monolithic model names to a set of fundamental design choices, we can cultivate a more nuanced and systematic approach.

2.1 Deconstructing Complexity

The proposed framework (Table 1 guides the deconstruction of even highly complex architectures into a more manageable set of explicit choices along these 10 fundamental axes.

Consider, for example, the Transformer (Vaswani et al., 2017). An encoder block within a transformer can be understood through its specific configuration of dimensional choices: Information Flow is Acyclic (Feedforward) enhanced by an Additive Shortcut Strategy (Residual connections); its core Aggregation Mechanism is Attentional (self-attention); Normalization is Layer-wise; Edge Transformations are primarily Linear (for query, key, value projections and within feed-forward networks), which are then followed by Nonlinear Activations (Softmax for attention weights, GELU/ReLU in the FFN sub-layer). Bias Mechanisms often include Static Learned biases in linear layers, and may incorporate Structured/Computed biases such as positional encodings at the input stage. Connectivity within the self-attention mechanism is typically Dense (all-to-all within the context window), while Parameter Sharing is employed for the feed-forward network across positions.

This decomposition, illustrated for several prominent architectures in Table 2 (with more detailed analyses in Appendix A), clarifies which fundamental components and their interactions contribute to an architecture's observed capabilities. For transformers, this helps attribute its success in modeling long-range dependencies to the Attentional Aggregation and unhindered information flow, and its scalability to great depths to the effective use of Additive Shortcuts and Layer Normalization. Such analysis fosters a more principled understanding of *why* specific architectures work, moving beyond isolated empirical success towards a structural rationale.

Table 2: Illustrative Dimensional Decomposition of Popular Model Architectures

Dimension	**MLP**	**ResNet**	**LSTM**	**Transformer (Enc)**	**KAN**
Edge Transform.	Linear	Linear (Conv)	Linear	Linear	**Nonlinear (Spline)**
Node Activation	Nonlinear	Nonlinear	Nonlinear	Nonlinear	**Linear/None**
Bias Mechanism	Static Learned	Static Learned	Static Learned	Static/Structural	Implicit/None
Norm. Strategy	Often None	Batch	Often None	Layer	Often None
Aggregation	Additive	Additive	Additive/**Gated***	**Attentional**	Additive
Connectivity	Dense	Sparse (Local)	Specific (Gated)	**Dense (Attention)**	Dense
Info. Flow Dir.	Acyclic	Acyclic	**Cyclic**	Acyclic	Acyclic
Shortcut Strat.	No Shortcuts	**Additive (Res.)**	(Internal Gating)	**Additive (Res.)**	No Shortcuts
Memory Struct.	Stateless	Stateless	**Stateful**	Stateless	Stateless
Param. Sharing	Independent	**Shared (Kernels)**	Shared (Time)	Shared (Pos-wise FFN)	Independent

**LSTM internal gating involves multiplicative interactions more complex than basic aggregation categories.*
Bold indicates particularly notable or contrasting dimensional choices for the architecture listed.

2.2 Guiding Principled Architectural Discovery

Beyond enhancing understanding, a dimensional perspective fundamentally reshapes the process of architectural innovation from serendipitous discovery to a more systematic and principled endeavor.

The value of systematic frameworks in neural network design has historical precedent. Earlier taxonomic descriptions of neural networks (Gardner & Dorling, 1998; Jain et al., 1996; Judd, 1990; Goodfellow et al., 2016; Raghu et al., 2016) created a common vocabulary that helped researchers understand relationships between seemingly disparate architectures. Similar systematic thinking has led to significant architectural breakthroughs: Hochreiter's analysis of gradient flow led to LSTMs (Hochreiter & Schmidhuber, 1997; Hochreiter, 1991); structured understanding of layer degradation gave us ResNets (He et al., 2015); and neuroscientific categorization of visual processing directly influenced CNN development alongside mathematical insights and engineering solutions. Most recently, revisiting the Kolmogorov-Arnold theorem through a systematic lens produced KANs with nonlinear edge transformations (Liu et al., 2025). While these advances often targeted specific limitations rather than following a comprehensive framework, they demonstrate how structured thinking about fundamental design choices can catalyze architectural innovation. Our proposed dimensional framework builds on this tradition but offers a more comprehensive and principled approach to the entire design space.

By conceptualizing the design space through fundamental dimensions such as those in Table 1, several avenues for more structured innovation emerge:

Mapping existing architectures (Table 2) onto the high dimensional space inevitably reveals combinations of dimensional variations that are currently under-explored or entirely uncharted. For instance, while Nonlinear Edge Transformations are gaining some initial traction with models like KANs (Liu et al., 2025) in predominantly Acyclic (Feedforward) settings without explicit Shortcuts, their systematic integration with Cyclic (Recurrent) information flow or advanced Shortcut Strategies like Gated Shortcuts represents a vast, largely unexplored frontier. Similarly, the combination of highly Dynamic Connectivity with sophisticated External Memory structures could unlock capabilities for adaptive, large-scale reasoning systems currently beyond our reach. Our framework explicitly enumerates such research opportunities for each dimension, effectively providing a roadmap for these "guided expeditions" into less-trodden regions of the design space, potentially yielding architectures with unique and valuable properties.

Instead of defaulting to monolithic, general-purpose architectures, designers can leverage a dimensional understanding to construct solutions. For a new task or dataset, one can reason about which dimensional choices and their specific variations best embody the required inductive biases or computational properties. For example, tasks requiring strict permutation invariance might favor specific Aggregation mechanisms with appropriate Parameter Sharing, while problems involving fine-grained hierarchical feature understanding would benefit from deliberate choices in Hierarchical Connectivity and suitable Normalization Strategies at different scales. This allows for more hypothesis-driven architectural development.

Many Neural Architecture Search (NAS) approaches (Elsken et al., 2019; White et al., 2023; Pham et al., 2018) operate by searching over predefined, often complex, cells or a limited set of operations derived from existing successful architectures. A dimensional framework offers a more fundamental and potentially much richer search space. NAS could be reimagined to explore combinations of choices directly along these 10 dimensions. Such an approach could not only discover more radically novel architectures but also yield discovered architectures that are inherently more interpretable, as their structure would be defined by a set of explicit, fundamental dimensional choices rather than opaque, monolithic cells.

2.3 Towards More Principled and Cumulative Progress

The adoption of a common, dimensional language for describing and reasoning about neural network architectures could accelerate the field's progress by addressing several systemic inefficiencies.

A shared vocabulary based on fundamental dimensions can help minimize the re-invention of the wheel under different terminologies and facilitate clearer, unambiguous communication of architectural innovations, thus reducing redundancy and fostering clarity.

By deconstructing architectures into comparable dimensional choices, insights gained from one architectural family (e.g. the importance of skip connections in CNNs) can be more readily understood for comparison and potentially transferred to others (e.g. Transformers). A structured, dimensional framework provides a more systematic and less daunting way to teach the fundamentals of neural network design, moving beyond a catalog of popular models.

Finally, isolating fundamental design dimensions creates a more tractable basis for theoretical analysis. Instead of attempting to theorize about monolithic architectures, researchers can investigate the specific impact of choices along each dimension, such as representational capacity, generalization properties, or optimization landscape characteristics and their interplay.

3 Framework Dimensions of Neural Network Design

Our proposed conceptual framework for systematic neural network design (summarized previously in Table 1) is built upon 10 fundamental, quasi-orthogonal structural dimensions. Each dimension encapsulates a core architectural decision point, profoundly influencing a network's computational properties, inductive biases, and overall representational capacity. Below, we provide a concise overview of each dimension and its Variations.

3.1 Information Transformation Type (Edge Level)

Definition: This dimension specifies how information is transformed as it passes along individual connections (edges) between nodes or aggregated representations.

Motivation: The nature of edge-level transformations fundamentally dictates the functional complexity the network can learn locally and how information is progressively refined or abstracted. Systematically considering this moves beyond treating layers as monolithic blocks.

Variations:

- **Linear Transformations:** Information is modified via linear functions (e.g. $y = Wx + b$). Examples include weight matrices in Multi-Layer Perceptrons (MLPs) and convolutional kernels in Convolutional Neural Networks (CNNs) (LeCun et al., 1989a).
- **Nonlinear Transformations:** Information is transformed directly via learnable nonlinear functions along edges. An example includes the B-spline activation functions on edges in Kolmogorov-Arnold Networks (KANs) (Liu et al., 2025).
- **Identity (No Transformation):** Information passes through unchanged. This is fundamental to various shortcut connections, such as those in Residual Networks (ResNets) (He et al., 2015).

3.2 Activation Function Type (Node Level)

Definition: This dimension describes how aggregated information, received from incoming edges, is processed (typically nonlinearly) at each node before being propagated.

Motivation: Node-level activations are critical for introducing nonlinearities that enable networks to learn complex patterns and break linear dependencies between layers, thus increasing expressive power.

Variations:

- **Linear Activation:** The output is a linear function of the aggregated input, often the identity (e.g. $f(x) = x$). Commonly used in output layers for regression tasks.
- **Nonlinear Activation:** The output is a nonlinear function of the aggregated input. Widespread examples include ReLU (Nair & Hinton, 2010), Sigmoid, Tanh, and more recent functions like GELU (Hendrycks & Gimpel, 2023).
- **No Activation (Identity Passthrough):** The aggregated input passes through entirely unchanged after aggregation. Distinct from a linear activation if aggregation is complex.

3.3 Bias Mechanism (Node/Transformation Level)

Definition: This dimension outlines how an offset or bias term is determined and applied to shift the operating point of a node or transformation.

Motivation: Biases provide learnable offsets, allowing activation functions to operate in different regimes independently of input magnitudes, thereby enhancing function approximation.

Variations:

- **Static Learned Bias:** A fixed, learnable parameter (scalar, vector, or tensor) added during transformation (e.g. the "b" in $y = Wx + b$).
- **Conditional/Dynamic Bias:** The bias term is generated dynamically, often by another part of the network or conditioned on external input (e.g. FiLM (Perez et al., 2018)).
- **Structured/Computed Bias:** Bias is determined by a predefined, non-learned rule, often based on input properties like position (e.g. additive sinusoidal positional encodings (Vaswani et al., 2017)).
- **No Bias (Zero Bias):** Explicit omission of a bias term, constraining the transformation.

3.4 Normalization Strategy (Intra-Layer Level)

Definition: This dimension specifies how activations or pre-activations within a layer (or for a set of features) are re-scaled and re-centered, typically based on their running statistics.

Motivation: Normalization techniques stabilize training dynamics in deep networks, mitigating internal covariate shift, enabling higher learning rates, and sometimes providing regularization.

Variations:

- **Batch-wise Normalization:** Normalizes activations across the batch dimension for each feature independently (e.g. Batch Normalization (Ioffe & Szegedy, 2015)).
- **Layer-wise Normalization:** Normalizes activations across all features for a single instance within a layer (e.g. Layer Normalization (Ba et al., 2016)).
- **Instance-wise Normalization:** Normalizes across spatial dimensions for each feature and instance independently (e.g. Instance Normalization (Ulyanov et al., 2016)).
- **Group-wise Normalization:** Divides channels into groups and normalizes within each group (e.g. Group Normalization (Wu & He, 2018)).
- **No Explicit Normalization:** Relies on other mechanisms (e.g. careful initialization, or simply capriciously good fortune) to manage training dynamics.

3.5 Aggregation Mechanism (Node Input Level)

Definition: This dimension describes how multiple inputs arriving at a node (or a conceptual aggregation point) are combined into a single representation.

Motivation: The choice of aggregation imposes strong inductive biases about how information from different sources should interact or be summarized, significantly impacting pattern learning.

Variations:

- **Additive (Sum, Weighted Sum):** Inputs are typically multiplied by weights and then summed. This is the standard operation in dense and convolutional layers.
- **Order-Sensitive (Sequential):** Inputs are processed in a specific order, with state carried forward. Fundamental to RNNs like LSTMs (Hochreiter & Schmidhuber, 1997).
- **Statistical (Mean, Max, Min):** Inputs are combined using a statistical function (e.g. MaxPooling in CNNs (Cireşan et al., 2011; Scherer et al., 2010)).
- **Attentional (Similarity-Based Weighting):** Inputs are weighted based on learned relevance or similarity metrics, then combined (e.g. Self-Attention in transformers (Vaswani et al., 2017; Niu et al., 2021)).

3.6 Connectivity Pattern (Inter-Node Topology)

Definition: This dimension specifies which nodes or groups of nodes connect to which others, defining the network's graph structure and information flow paths.

Motivation: Connectivity dictates parameter count, computational cost, receptive field properties, and imposes structural inductive biases reflecting assumed data properties (e.g. locality in images).

Variations:

- **Dense (Fully Connected):** Every node in one layer connects to every node in the next (e.g. layers in MLPs).
- **Sparse (Selected Connections):** Only certain connections exist, often with a regular pattern (e.g. local connectivity of kernels in CNNs).
- **Hierarchical (Structured Sparsity):** Connections form a multi-scale or hierarchical structure (e.g. U-Net (Ronneberger et al., 2015) encoder-decoder).
- **Dynamic (Input-Dependent Connections):** Connections are determined or modulated during execution based on the input (e.g. dynamic convolutions (Chen et al., 2019)).
- **Stochastic (Train-Time Regularization):** A subset of connections or entire units is randomly and temporarily deactivated or "dropped" during each training iteration (e.g. Dropout (Srivastava et al., 2014), DropConnect (Wan et al., 2013)).

3.7 Information Flow Direction (Overall Graph Structure)

Definition: This dimension describes the overall directionality of information propagation through the network graph, specifically whether cycles are permitted.

Motivation: The allowance of cycles fundamentally determines a network's ability to model temporal dependencies or processes requiring iterative refinement of internal state.

Variations:

- **Acyclic (Feedforward):** Information flows in one direction from input to output without forming cycles. Most common architectures like CNNs and standard transformers are acyclic.
- **Cyclic (Recurrent):** Information can flow in cycles, allowing the network to maintain an internal state updated based on current inputs and past states. Defining feature of RNNs (Elman, 1990; Rumelhart et al., 1986; Sherstinsky, 2018).

3.8 Shortcut Strategy (Intra-Network Pathway Modification)

Definition: This dimension describes how direct alternative pathways for information and gradients are established, allowing signals to bypass one or more intermediate transformations.

Motivation: Shortcuts are crucial for training very deep networks by mitigating vanishing gradients and enabling easier learning of identity mappings, enabling feature reuse and deeper hierarchies.

Variations:

- **No Shortcuts (Sequential Path):** Information flows strictly sequentially through layers without bypass connections (e.g. traditional CNNs like VGGNet (Simonyan & Zisserman, 2014)).
- **Additive Shortcuts (Residual):** Output of an earlier layer is added element-wise to that of a later layer (e.g. ResNets (He et al., 2015)).

- **Concatenative Shortcuts (Dense):** Feature maps from earlier layers are concatenated with those from later layers (e.g. DenseNets (Huang et al., 2016)).
- **Gated Shortcuts (Highway):** A gating mechanism controls information flow through main versus shortcut paths (e.g. Highway Networks (Srivastava et al., 2015)).

3.9 Memory Structure (Information Retention Across Steps)

Definition: This dimension describes how the network retains, accesses, and utilizes information across multiple processing steps or over extended contexts.

Motivation: Explicit memory mechanisms determine a network's capacity for sequential reasoning, long-term dependency modeling, and tasks requiring algorithmic manipulation of stored information.

Variations:

- **Stateless (Pure Function of Current Input):** Output depends only on the current input batch, with no persistent internal state across batches or distinct processing steps (e.g. standard feedforward CNNs).
- **Stateful (Maintains Internal State):** The network maintains an internal state that persists and evolves between processing steps (e.g. hidden state in LSTMs (Hochreiter & Schmidhuber, 1997)).
- **External Memory:** The network can read from and write to an explicit, often differentiable, memory structure (e.g. Neural Turing Machines (Graves et al., 2014)).

3.10 Parameter Sharing Pattern (Weight Re-utilization)

Definition: This dimension describes how parameters (weights and biases) are shared or reused across different parts of the network.

Motivation: Parameter sharing drastically reduces model size, can enforce invariances (e.g. translation equivariance in CNNs), and improves generalization by allowing features learned in one part to be applied elsewhere.

Variations:

- **Independent (Unique Parameters):** Each connection or distinct processing unit uses its own unique set of parameters (e.g. fully connected layers in an MLP).
- **Shared (Parameters Reused):** The same parameters are used for multiple connections or operations (e.g. convolutional kernels shared across spatial locations in CNNs; weights shared across time steps in RNNs).
- **Conditional (Input-Dependent Sharing/Selection):** Parameter usage, sharing, or selection depends on the input (e.g. Mixture of Experts (Shazeer et al., 2017) where experts (sets of parameters) are conditionally activated).

4 Addressing Concerns

The proposition of a dimensional framework for neural network design undoubtedly invite understandable reservations, which we address here:

"Is this framework not overly complex itself?" While encompassing 10 dimensions might appear daunting, we argue that it brings explicit clarity to a complexity that is already inherent, yet often implicit, in modern neural architectures. The framework does not add new layers of complexity but rather provides a structured vocabulary to articulate and manage the existing multitude of design choices. While a map doesn't guarantee you'll find El Dorado (or the next thing after the transformer), it beats wandering aimlessly, does it not?

"Current methods, however messy, are achieving remarkable success; why poke the bear?" We fully acknowledge the empirical successes of current approaches. Our position is not at all to say that these methods are invalid, but that their underlying principles can be better understood, their successes potentially replicated more efficiently, and their limitations overcome more systematically with a dimensional lens. The goal is to make sustained progress less reliant on serendipity and more on principled exploration, particularly as low-hanging fruit becomes scarcer.

"Are these dimensions truly orthogonal when many co-occur in established architectures?" This is like assuming carrots can only be used in mirepoix (onions, carrots, and celery) because that's their traditional context. A chef understands that carrots can function equally well in a soup or a carrot cake; here, our framework recognizes that architectural dimensions that commonly co-occur can be separated and recombined in novel ways. We term the dimensions "quasi-orthogonal" because while they represent distinct axes of design, we recognize that choices along one dimension can influence the functional impact or even the sensible options along another (e.g. network depth often necessitates shortcut strategies). The power of the dimensional approach lies in deconstructing these complex interactions by first isolating the fundamental choices.

"Will such a systematic approach stifle creativity and intuitive leaps?" To the contrary, a well-defined map and a richer palette typically empower, rather than inhibit, creativity. Good science often arises from a deep understanding of fundamental components and their possible interactions. This framework aims to provide such an understanding, enabling researchers to make more informed intuitive leaps and to creatively combine dimensional variations in truly novel ways, rather than constraining them to minor variations of existing kitchen recipes.

5 Call to the Community

Realizing the full potential of a systematic, dimensional approach to neural network design requires a concerted effort from various stakeholders within the machine learning community. We call upon:

Researchers: To actively employ dimensional thinking not only when designing novel architectures but also when analyzing and reporting on existing ones. Explicitly articulating dimensional choices in publications can enhance clarity, reproducibility, and comparative understanding. Furthermore, we encourage the deliberate probing of under-explored regions of this high dimensional design space.

Educators: To integrate such foundational frameworks into curricula, thereby equipping the next generation of researchers with a deeper, more structural understanding of neural network design that transcends the cataloging of popular model kitchen recipes.

Builders of ML Frameworks and NAS Platforms: To consider how their software tools and search algorithms can better represent, support, and even automate the exploration of architectural design along these fundamental dimensions, moving beyond fixed cell structures or operator lists.

Reviewers and Program Committees: To recognize and value contributions that advance systematic understanding, propose well-justified novel dimensional choices, or thoroughly explore regions of the dimensional design space, alongside traditional metrics of benchmark performance. Such an emphasis fosters a research ecosystem that rewards foundational insights as well as empirical gains.

Adoption and refinement of such dimensional perspectives can transition our field towards a more robust, principled, and ultimately more impactful paradigm for neural architecture innovation.

6 Conclusion

Neural network alchemy, characterized by remarkable but often serendipitous discoveries, has undeniably propelled the field of deep learning to unprecedented heights. However, to sustain this momentum and tackle increasingly complex challenges, we contend that a shift towards a more principled science of architectural design is necessary. We have proposed an 10-dimensional framework as a step in this direction, aiming to equip researchers and practitioners with a systematic language and conceptual map for navigating the vast neural network design space.

Where the community as of now actively explores the vast capabilities of current paradigms like attention transformers, while also anticipating the next fundamental architectural shift, such a map becomes even more viable. Rather than passively awaiting the emergence of a successor through a spark of genius, the dimensional approach advocated here equips researchers with tools to proactively chart and explore the design space, potentially leading to these novel breakthroughs or, at minimum, to rapidly understand and integrate them into our cumulative knowledge.

Dimensional analysis can reveal potentially promising combinations that remain underexplored. For instance, while nonlinear edge transformations have been explored in feedforward networks (KANs), their integration with cyclic information flow presents interesting theoretical possibilities for modeling dynamic systems. Similarly, the combination of dynamic connectivity with conditional parameter sharing could theoretically allow for more adaptive computation paths without the overhead of fully dynamic architectures. We highlight these not as empirically validated directions but as illustrations of how a dimensional perspective can guide more systematic exploration of the design space.

By deconstructing architectures into their fundamental dimensional choices, this framework fosters deeper understanding of current successes. While no single framework can be exhaustive or final, we believe that the explicit consideration of these core structural dimensions and the collaborative effort to refine and expand such systematic understandings offers a route beyond the current "architecture zoo." It paves the way for a more mature, cumulative, and principled approach to neural network research and application, ultimately accelerating the development of more capable, interpretable, and reliable intelligent systems.

While our dimensional framework focuses specifically on neural networks, we note that many of these dimensions could potentially characterize any computational function approximator that can be abstracted to a graph structure (neural networks are, after all, computational graphs). The connection to graph theory is particularly relevant; dimensions like Connectivity Pattern, Information Flow Direction, and Shortcut Strategy have clear parallels in classical graph-theoretical concepts. However, other dimensions such as Edge Transformation and Node Activation extend beyond standard graph theory to capture the specific computational nature of neural networks.

The standard vocabulary from graph theory, while foundational, doesn't inherently capture many of the architectural design choices that make a ResNet a ResNet, a Transformer a Transformer, or a KAN a KAN. This suggests the potential of what we could call "neural graph theory" as a specialized extension of graph theory that incorporates the unique aspects of neural networks while maintaining connections to the mathematical foundations of graph structures. Such a theoretical bridge might reveal why certain combinations of these 10 dimensions are particularly powerful, or suggest entirely new dimensions based on first principles, and potentially offer insights into how useful complexity arises from information flow within discrete graph structures.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. 2016. URL `https://arxiv.org/abs/1607.06450`.

Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari S. Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Grégoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning. *ArXiv*, abs/2304.12210, 2023. URL `https://api.semanticscholar.org/CorpusID:258298825`.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=xm6YD62D1Ub`.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael K Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: Extended long short-term memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=ARAxPPIAhq`.

L. Bottou, C. Cortes, J.S. Denker, H. Drucker, I. Guyon, L.D. Jackel, Y. LeCun, U.A. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of classifier methods: a case study in handwritten digit recognition. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3 - Conference C: Signal Processing (Cat. No.94CH3440-5)*, volume 2, pp. 77–82 vol.2, 1994. doi: 10.1109/ICPR.1994.576879.

Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. *ArXiv*, abs/2001.10167, 2020a. URL `https://api.semanticscholar.org/CorpusID:210932292`.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, 2020b. URL `https://api.semanticscholar.org/CorpusID:220363476`.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Kristjanson Duvenaud. Neural ordinary differential equations. In *Neural Information Processing Systems*, 2018. URL `https://api.semanticscholar.org/CorpusID:49310446`.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *ArXiv*, abs/2002.05709, 2020c. URL `https://api.semanticscholar.org/CorpusID:211096730`.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 15750–15758. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.01549. URL `https://openaccess.thecvf.com/content/CVPR2021/html/Chen_Exploring_Simple_Siamese_Representation_Learning_CVPR_2021_paper.html`.

Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11027–11036, 2019. URL `https://api.semanticscholar.org/CorpusID:208910380`.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, 2014. URL `https://api.semanticscholar.org/CorpusID:5590763`.

Dan C. Cireşan, Ueli Meier, Jonathan Masci, Luca M. Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, pp. 1237–1242. AAAI Press, 2011. ISBN 9781577355144.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Annual Meeting of the Association for Computational Linguistics*, 2019. URL `https://api.semanticscholar.org/CorpusID:57759363`.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2020. URL `https://api.semanticscholar.org/CorpusID:225039882`.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. doi: 10.1207/s15516709cog1402_1.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: a survey. *J. Mach. Learn. Res.*, 20(1):1997–2017, January 2019. ISSN 1532-4435.

M.W Gardner and S.R Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14):2627–2636, 1998. ISSN 1352-2310. doi: https://doi.org/10.1016/S1352-2310(97)00447-0. URL `https://www.sciencedirect.com/science/article/pii/S1352231097004470`.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, October 2020. ISSN 0001-0782. doi: 10.1145/3422622. URL `https://doi.org/10.1145/3422622`.

Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks : the official journal of the International Neural Network Society*, 18 5-6:602–10, 2005. URL `https://api.semanticscholar.org/CorpusID:1856462`.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *ArXiv*, abs/1410.5401, 2014. URL `https://api.semanticscholar.org/CorpusID:15299054`.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21271–21284. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/f3ada80d5c4ee70142b17b8192b2958e-Paper.pdf`.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper_files/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf`.

Tuomas Haarnoja, Aurick Zhou, P. Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *ArXiv*, abs/1801.01290, 2018. URL `https://api.semanticscholar.org/CorpusID:28202810`.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015. URL `https://api.semanticscholar.org/CorpusID:206594692`.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, 2016. URL `https://api.semanticscholar.org/CorpusID:6447277`.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning . In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9726–9735, Los Alamitos, CA, USA, June 2020. IEEE Computer Society. doi: 10.1109/CVPR42600.2020.00975. URL `https://doi.ieeecomputersociety.org/10.1109/CVPR42600.2020.00975`.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). 2023. URL `https://arxiv.org/abs/1606.08415`.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Sepp Hochreiter. Untersuchungen zu dynaarticlehen neuronalen netzen. 1991. URL `https://api.semanticscholar.org/CorpusID:60091947`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997. URL `https://api.semanticscholar.org/CorpusID:1915014`.

Kurt Hornik, Maxwell B. Stinchcombe, and Halbert L. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989. URL `https://api.semanticscholar.org/CorpusID:2757547`.

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2016. URL `https://api.semanticscholar.org/CorpusID:9433631`.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/ioffe15.html`.

A.K. Jain, Jianchang Mao, and K.M. Mohiuddin. Artificial neural networks: a tutorial. *Computer*, 29(3): 31–44, 1996. doi: 10.1109/2.485891.

J. Stephen Judd. Neural network design and the complexity of learning. In *Neural network modeling and connectionism*, 1990. URL `https://api.semanticscholar.org/CorpusID:261292752`.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021. doi: 10.1038/s41586-021-03819-2.

Patrick Kidger. On neural differential equations. *ArXiv*, abs/2202.02435, 2022. URL `https://api.semanticscholar.org/CorpusID:246634262`.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013. URL `https://api.semanticscholar.org/CorpusID:216078090`.

Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907, 2016. URL `https://api.semanticscholar.org/CorpusID:3144218`.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In S. Solla, T. Leen, and K. Müller (eds.), *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL `https://proceedings.neurips.cc/paper_files/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf`.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

Věra Kůrková. Kolmogorov's theorem is relevant. *Neural Computation*, 3:617–622, 1991. URL `https://api.semanticscholar.org/CorpusID:123557782`.

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*, 1(4):541–551, 1989a. doi: 10.1162/neco.1989.1.4.541.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In *Neural Information Processing Systems*, 1989b. URL https://api.semanticscholar.org/CorpusID:2542741.

Yann LeCun, Sumit Chopra, Raia Hadsell, Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. 2006. URL https://api.semanticscholar.org/CorpusID:8531544.

Stefan Leijnen and Fjodor van Veen. The neural network zoo. *Proceedings*, 47(1), 2020. ISSN 2504-3900. doi: 10.3390/proceedings2020047009. URL https://www.mdpi.com/2504-3900/47/1/9.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljacic, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov–arnold networks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Ozo7qJ5vZi.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Kirkeby Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015. URL https://api.semanticscholar.org/CorpusID:205242740.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016. URL https://api.semanticscholar.org/CorpusID:6875312.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pp. 807–814, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021. URL https://api.semanticscholar.org/CorpusID:233562906.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: visual reasoning with a general conditioning layer. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *ArXiv*, abs/1802.03268, 2018. URL https://api.semanticscholar.org/CorpusID:3638969.

Sébastien Racanière, Théophane Weber, David P. Reichert, Lars Buesing, Arthur Guez, Danilo Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter Battaglia, Demis Hassabis, David Silver, and Daan Wierstra. Imagination-augmented agents for deep reinforcement learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 5694–5705, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Maithra Raghu, Ben Poole, Jon M. Kleinberg, Surya Ganguli, and Jascha Narain Sohl-Dickstein. On the expressive power of deep neural networks. In *International Conference on Machine Learning*, 2016. URL https://api.semanticscholar.org/CorpusID:2838204.

Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10674–10685, 2021. URL https://api.semanticscholar.org/CorpusID:245335280.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *ArXiv*, abs/1505.04597, 2015. URL `https://api.semanticscholar.org/CorpusID:3719281`.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. 1986. URL `https://api.semanticscholar.org/CorpusID:62245742`.

Dominik Scherer, Andreas C. Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *International Conference on Artificial Neural Networks*, 2010. URL `https://api.semanticscholar.org/CorpusID:18388506`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017. URL `https://api.semanticscholar.org/CorpusID:28695052`.

Noam M. Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *ArXiv*, abs/1701.06538, 2017. URL `https://api.semanticscholar.org/CorpusID:12462234`.

Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *ArXiv*, abs/1808.03314, 2018. URL `https://api.semanticscholar.org/CorpusID:51968707`.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL `https://api.semanticscholar.org/CorpusID:14124313`.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56): 1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *ArXiv*, abs/1505.00387, 2015. URL `https://api.semanticscholar.org/CorpusID:14786967`.

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. *IEEE Trans. Neural Networks*, 9:1054–1054, 1998. URL `https://api.semanticscholar.org/CorpusID:60035920`.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'99, pp. 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

Yee Whye Teh, Max Welling, Simon Osindero, and Geoffrey E. Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4:1235–1260, 2003. URL `https://api.semanticscholar.org/CorpusID:272555194`.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *ArXiv*, abs/1607.08022, 2016. URL `https://api.semanticscholar.org/CorpusID:16516553`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In Sanjoy Dasgupta and David McAllester (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1058–1066, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL `https://proceedings.mlr.press/v28/wan13.html`.

Christopher Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992. URL `https://api.semanticscholar.org/CorpusID:208910339`.

Colin White, Mahmoud Safari, Rhea Sanjay Sukthanker, Binxin Ru, Thomas Elsken, Arber Zela, Debadeepta Dey, and Frank Hutter. Neural architecture search: Insights from 1000 papers. *ArXiv*, abs/2301.08727, 2023. URL `https://api.semanticscholar.org/CorpusID:256080480`.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. A theory of generative convnet. *ArXiv*, abs/1602.03264, 2016. URL `https://api.semanticscholar.org/CorpusID:7252503`.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *ArXiv*, abs/1605.07146, 2016. URL `https://api.semanticscholar.org/CorpusID:15276198`.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12310–12320. PMLR, 2021. URL `http://proceedings.mlr.press/v139/zbontar21a.html`.

A Appendix: Architectural Case Studies

Here we survey and provide detailed dimensional breakdowns for a handful of neural network architectures, illustrating how the 10-dimensional framework (defined in Section 3 of the paper) can be used for analysis and understanding existing models as a descriptive framework.

A.1 Ordinary Least Squares (OLS) Regression as a "Neural Network"

For an illustrative, perhaps even funny, baseline and to demonstrate the framework's ability to explain even simple models, we can consider the Ordinary Least Squares (OLS) regression. An OLS model predicting y from features X with $y = X\beta + \epsilon$ can be viewed as a very simple, single layer feedforward "neural network" without hidden units.

- **Dimension 1 (Information Transformation Type - Edge Level):** *Linear Transformations.* The core operation is the matrix multiplication $X\beta$, which is a linear transformation of the input features X by the learned coefficients (weights) β.
- **Dimension 2 (Activation Function Type - Node Level):** *Linear Activation (Identity).* The output $X\beta$ is used directly; there is no nonlinear activation function applied. $f(z) = z$.
- **Dimension 3 (Bias Mechanism - Node/Transformation Level):** If an intercept term is included in the OLS model (one column of X is all ones, or an explicit intercept β_0 is fit), this corresponds to a *Static Learned Bias.* If no intercept is fit, it's *No Bias.*
- **Dimension 4 (Normalization Strategy - Intra-Layer Level):** *No Explicit Normalization* within the model itself. (Feature scaling/normalization is a preprocessing step, not part of the OLS model's architecture).

- **Dimension 5 (Aggregation Mechanism - Node Input Level):** *Additive (Weighted Sum).* The prediction for each output (if y is multivariate) or the single output y is a weighted sum of the input features X_j with weights β_j.
- **Dimension 6 (Connectivity Pattern - Inter-Node Topology):** *Dense (Fully Connected)* if considering inputs to a single output node. Each input feature X_j connects to the output.
- **Dimension 7 (Information Flow Direction - Overall Graph Structure):** *Acyclic (Feedforward).* A direct, one-step computation from input to output.
- **Dimension 8 (Shortcut Strategy - Intra-Network Pathway Modification):** *No Shortcuts.*
- **Dimension 9 (Memory Structure - Information Retention Across Steps):** *Stateless.*
- **Dimension 10 (Parameter Sharing Pattern - Weight Re-utilization):** *Independent* parameters (each β_j is unique for each feature X_j).

Discussion: Viewing OLS through this dimensional lens reveals it as a minimal instance of a feedforward neural network: it employs only linear edge transformations, linear activation, additive aggregation, and no hidden layers or complex strategies for memory, shortcuts, or parameter sharing. Its "architecture" is about as simple as possible while still structured as a learnable mapping.

This helps to ground the dimensional framework and illustrates that many more complex neural network architectures build upon these very fundamental choices by introducing nonlinearities, depth, structural priors (like sparsity or weight sharing), and more sophisticated mechanisms for information flow and memory. Here, we highlight that our framework isn't just for deep or complex networks but attempts to provide a language for all function approximators structured as interconnected nodes and transformations.

A.2 Multi-Layer Perceptron (MLP)

The Multi-Layer Perceptron is one of the earliest neural network architectures (if you don't count OLS), consisting of a series of fully connected layers. It is a universal approximator for continuous functions under certain conditions (Hornik et al., 1989).

- **Dimension 1 (Edge Transformation Type - Edge Level):** Predominantly *Linear Transformations.* Each neuron's input is a weighted sum of outputs from the previous layer, $y = Wx + b$, which is a linear transformation of the previous layer's activations x.
- **Dimension 2 (Activation Function Type - Node Level):** Typically *Nonlinear Activation.* After the linear transformation and aggregation, a nonlinear function (e.g. Sigmoid, Tanh, ReLU (Nair & Hinton, 2010)) is applied element-wise. The output layer might use a Linear activation (for regression) or a Softmax activation (for multi-class classification, often considered part of the activation/output mapping).
- **Dimension 3 (Bias Mechanism - Node/Transformation Level):** *Static Learned Bias.* The 'b' term in $Wx + b$ is a learnable vector added to the weighted inputs of each layer.
- **Dimension 4 (Normalization Strategy - Intra-Layer Level):** Classic MLPs often used *No Explicit Normalization.* However, modern deep MLPs may incorporate *Batch-wise Normalization* (Ioffe & Szegedy, 2015) or *Layer-wise Normalization* (Ba et al., 2016) between layers to aid training.
- **Dimension 5 (Aggregation Mechanism - Node Input Level):** *Additive (Weighted Sum).* Each neuron computes a weighted sum of its inputs from the preceding layer.
- **Dimension 6 (Connectivity Pattern - Inter-Node Topology):** *Dense (Fully Connected).* Every neuron in one layer is connected to every neuron in the subsequent layer.

- **Dimension 7 (Information Flow Direction - Overall Graph Structure):** *Acyclic (Feedforward).* Information propagates strictly from input layers, through hidden layers, to output layers without cycles.
- **Dimension 8 (Shortcut Strategy - Intra-Network Pathway Modification):** Typically *No Shortcuts.* Standard MLPs lack direct connections that bypass layers.
- **Dimension 9 (Memory Structure - Information Retention Across Steps):** *Stateless.* The output for a given input depends solely on that input and the learned weights; no state is carried over between distinct input samples.
- **Dimension 10 (Parameter Sharing Pattern - Weight Reutilization):** *Independent.* Each layer has its own unique weight matrix (W) and bias vector (b). There is no sharing of parameters across layers or within different parts of a layer (unlike CNNs or RNNs).

Discussion: The MLP's dimensional choices (Dense Connectivity, Independent Parameters, No Shortcuts, Stateless Memory) make it a general-purpose function approximator but limit its efficiency and effectiveness for data with inherent structure like spatial locality (images) or sequential dependencies (time series), where architectures with Sparse/Shared parameters and specialized flow/memory structures excel. Its simplicity, however, illustrates the interplay of core linear transformations and nonlinear activations compared to the Ordinary Least Squares regression.

A.3 Convolutional Neural Networks (LeNet, AlexNet, VGGNet)

LeNet (Lecun et al., 1998; Bottou et al., 1994; LeCun et al., 1989b), AlexNet (Krizhevsky et al., 2012), and VGGNet (Simonyan & Zisserman, 2014) are early Convolutional Neural Networks designed for image classification tasks, laying the groundwork for modern deep learning in computer vision.

- **Dimension 1 (Edge Transformation):** Primarily *Linear Transformations.* This occurs through the convolutional operations (discrete convolutions, which are linear) and the linear transformations in its fully connected layers.
- **Dimension 2 (Activation):** *Nonlinear Activation.* LeNet-5 originally used Sigmoid or Tanh activation functions. AlexNet's key innovation was popularizing ReLU activations, which enabled training deeper networks by addressing vanishing gradient problems. VGGNet continued with ReLU activation. Modern counterparts would typically use ReLU.
- **Dimension 3 (Bias Mechanism):** *Static Learned Bias* terms are added in both convolutional and fully connected layers.
- **Dimension 4 (Normalization):** The original LeNet-5 had *No Explicit Normalization* of feature maps. AlexNet introduced *Local Response Normalization*, while later CNNs after VGGNet moved toward *Batch Normalization*, which became ubiquitous in modern CNNs.
- **Dimension 5 (Aggregation):** *Additive (Weighted Sum)* in convolutional layers. For pooling, LeNet-5 used *Average Pooling*, while AlexNet shifted to *Max Pooling*, which became standard in subsequent architectures like VGGNet.
- **Dimension 6 (Connectivity):** *Sparse (Selected Connections)* due to the local receptive fields of convolutional layers. Progresses to *Dense* connectivity in the final MLP-style layers. Also, *Hierarchical* structure emerges from the stacking of convolution/pooling layers, creating increasingly abstract features.
- **Dimension 7 (Info. Flow Dir.):** *Acyclic (Feedforward).*
- **Dimension 8 (Shortcut Strategy):** *No Shortcuts* in the original LeNet-5 design.
- **Dimension 9 (Memory Structure):** *Stateless.*

- **Dimension 10 (Param. Sharing):** Critically, *Shared* parameters are used via weight sharing in convolutional kernels across spatial locations. The fully connected layers use *Independent* parameters.

Discussion: The CNN family's success stems from dimensional choices tailored for image data: Sparse local connectivity and Shared parameters capture local features while providing translation equivariance efficiently.

The trajectory from LeNet to AlexNet to VGGNet primarily involved deeper networks (stack more layers), improved activations (ReLU), and refined kernel sizes, while maintaining the same fundamental dimensional architecture. AlexNet demonstrated that deeper networks with appropriate activations could significantly improve performance, while VGGNet showed that simpler, more uniform architectures with smaller stacked kernels could achieve better results with greater computational efficiency.

Notably, the core dimensional choices (Shared parameters, Sparse connectivity, Hierarchical structure) remained consistent through CNN evolution until ResNets introduced a fundamental change along the Shortcut Strategy dimension.

A.4 Residual Networks (ResNet, Wide ResNet)

ResNets (He et al., 2015; 2016; Zagoruyko & Komodakis, 2016) sparked a paradigm shift in deep learning by training much deeper networks than previously was thought feasible, primarily through residual (skip) connections.

- **Dimension 1 (Edge Transformation):** Predominantly *Linear Transformations* (convolutions within residual blocks). The skip connection itself is often an *Identity* transformation if channel dimensions match, or a *Linear Transformation* (1x1 convolution) if they need to be aligned.
- **Dimension 2 (Activation):** *Nonlinear Activation* (typically ReLU) is applied after the addition of the main path's output and the skip connection's output, and also within the main path (e.g. before a convolution if using pre-activation ResNet variant).
- **Dimension 3 (Bias Mechanism):** *Static Learned Bias* is typically used in the convolutional layers. Often, biases are omitted in convolutional layers immediately followed by Batch Normalization as the BN's beta parameter can serve a similar role.
- **Dimension 4 (Normalization):** *Batch-wise Normalization* is a ubiquitous component within ResNet blocks, typically applied after each convolution and before the activation.
- **Dimension 5 (Aggregation):** *Additive (Sum).* The core of the residual block is the element-wise addition of the shortcut path's output to the main transformation path's output. Convolutional layers also perform Additive (Weighted Sum) aggregation.
- **Dimension 6 (Connectivity):** *Sparse* (local convolutional receptive fields). Also *Hierarchical* as layers are stacked and feature map dimensions change through strided convolutions or pooling.
- **Dimension 7 (Info. Flow Dir.):** *Acyclic (Feedforward).* Despite the "skip," the overall graph remains feedforward.
- **Dimension 8 (Shortcut Strategy):** *Additive Shortcuts (Residual).* This is the defining characteristic of ResNets.
- **Dimension 9 (Memory Structure):** *Stateless.*
- **Dimension 10 (Param. Sharing):** *Shared* parameters (convolutional kernels are shared across spatial locations).

Discussion: ResNet's key contribution comes from its *Shortcut Strategy* (Additive Residuals), which, combined with appropriate *Normalization* (Batch Norm), drastically improves trainability of very deep acyclic networks. This allows for richer hierarchical feature learning without falling prey to vanishing gradients.

A.5 LSTMs (Long Short-Term Memory units in RNNs)

LSTMs (Hochreiter & Schmidhuber, 1997; Graves & Schmidhuber, 2005; Cho et al., 2014; Beck et al., 2024) are a type of Recurrent Neural Network (RNN) cell designed to overcome the vanishing gradient problem in simple RNNs, allowing them to learn long-range temporal dependencies.

- **Dimension 1 (Edge Transformation):** *Linear Transformations* are applied to the concatenation of the current input and the previous hidden state to compute values for the input, forget, output gates, and the candidate cell state.
- **Dimension 2 (Activation):** *Nonlinear Activation.* Sigmoid functions are typically used for the three gates (input, forget, output) to produce values between 0 and 1. Tanh is often used for the candidate cell state $\tilde{c}_t$ and for the final output h_t (after gating the cell state c_t).
- **Dimension 3 (Bias Mechanism):** *Static Learned Bias* terms are included in the linear transformations for gates and cell state computation.
- **Dimension 4 (Normalization):** Traditionally, LSTMs had *No Explicit Normalization* within the cell structure itself, though techniques like Layer Normalization can be applied to the outputs of an LSTM layer. Some research explores normalizing recurrent connections.
- **Dimension 5 (Aggregation):** Quite complicated. Input for gates/cell candidates uses *Additive (Weighted Sum).* The cell state update $c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$ involves element-wise multiplication (gating) followed by addition. The output $h_t = o_t \odot \tanh(c_t)$ also uses element-wise multiplication. These gating operations can be seen as a form of *Learned Modulatory Aggregation* not perfectly captured by basic categories but essential to LSTM function. It also implies *Order-Sensitive* processing due to the sequential nature of RNNs.
- **Dimension 6 (Connectivity):** Specific, fixed internal connections define the gate structures. The overall connectivity is driven by the sequential processing inherent in RNNs.
- **Dimension 7 (Info. Flow Dir.):** *Cyclic (Recurrent)* due to the feedback of the previous hidden state h_{t-1} and cell state c_{t-1} into the computation for the current time step.
- **Dimension 8 (Shortcut Strategy):** The cell state c_t provides a largely uninterrupted "information highway" across time steps, modulated by the forget and input gates. This resembles a *Gated Shortcut* through time for the cell state, mitigating vanishing gradients for long sequences. There are typically no explicit shortcuts bypassing multiple unrolled time steps in a single leap (as in deep feedforward nets).
- **Dimension 9 (Memory Structure):** Critically *Stateful*, maintaining both the hidden state h_t and the cell state c_t across time steps. The cell state acts as the primary long-term memory.
- **Dimension 10 (Param. Sharing):** *Shared* parameters (the weight matrices for gates and cell state computation are reused across all time steps).

Discussion: LSTMs are defined by their *Stateful Memory* (cell state), *Cyclic Information Flow*, and sophisticated internal *Gating Aggregation* (often described as having input, forget, and output gates). This allows them to selectively read, write, and forget information, making them effective (if hard/slow to train) for modeling long-range dependencies in sequential data. The cell state's pathway behaves like a Gated Shortcut across time.

A.6 Transformers (Encoder Block)

The Transformer architecture (Vaswani et al., 2017), particularly its encoder block, became the dominant model in NLP and is increasingly applied in other domains (Dosovitskiy et al., 2020).

- **Dimension 1 (Edge Transformation):** Primarily *Linear Transformations*. This includes the linear projections used to create Query (Q), Key (K), and Value (V) matrices in the self-attention mechanism, the output projection after attention, and the linear transformations within the position-wise Feed-Forward Network (FFN) sub-layer.
- **Dimension 2 (Activation):** *Nonlinear Activation*. Softmax is applied to the scaled dot-product scores in self-attention to obtain attention weights. The FFN sub-layer typically uses a nonlinear activation like ReLU or GELU (Hendrycks & Gimpel, 2023) between its two linear transformations.
- **Dimension 3 (Bias Mechanism):** Typically *Static Learned Bias* in the linear layers of the FFN and in the Q,K,V projections and output projection of self-attention (though sometimes omitted if normalization follows closely). Positional encodings (sinusoidal or learned) are often added to input embeddings, acting as a form of *Structured/Computed Bias* to provide positional information.
- **Dimension 4 (Normalization):** *Layer-wise Normalization* (Ba et al., 2016) is a key component, typically applied after the self-attention sub-layer and after the FFN sub-layer (i.e., before adding the residual connection's output).
- **Dimension 5 (Aggregation):** Critically employs *Attentional (Similarity-Based Weighting)* via the self-attention mechanism, where input token representations are aggregated based on learned, context-dependent attention weights. The residual connections use *Additive (Sum)* aggregation.
- **Dimension 6 (Connectivity):** *Dense (Fully Connected)* within the self-attention mechanism, as each token can attend to every other token in the input sequence (unless sparse attention variants are used). The FFN is applied position-wise, meaning it doesn't create new cross-token connections beyond what attention established. Stacking encoder layers creates a *Hierarchical* representation.
- **Dimension 7 (Info. Flow Dir.):** *Acyclic (Feedforward)*. Each encoder block processes information without cycles.
- **Dimension 8 (Shortcut Strategy):** Employs *Additive Shortcuts (Residual)* around both the self-attention sub-layer and the FFN sub-layer. These are vital for training deep Transformers.
- **Dimension 9 (Memory Structure):** *Stateless* for a single pass over a fixed-length input sequence. Information about the sequence is contained within the activations and attention patterns for that pass. Variants like Transformer-XL (Dai et al., 2019) introduce a notion of recurrence/state across segments to handle longer contexts, which would then be classified as *Stateful* at that inter-segment level.
- **Dimension 10 (Param. Sharing):** *Shared* parameters are used for the FFN, which is applied independently at each position. Within multi-head attention, parameters are shared across positions but are distinct per head before aggregation. Some research explores sharing parameters across encoder layers.

Discussion: The Transformer's usefulness comes from its unique combination of *Attentional Aggregation* allowing global context modeling, *Additive Shortcuts* enabling great depth, and efficient *Parameter Sharing* in its FFNs. *Layer Normalization* is crucial for stable training. Its *Stateless* nature for fixed contexts makes it highly parallelizable for training but requires specific additions like positional encodings for sequence order.

A.7 Graph Convolutional Networks (GCN)

Graph Convolutional Networks are designed to perform learning directly on graph-structured data, extending the concept of convolution to non-Euclidean domains. Here, we examine the variant by Kipf & Welling (Kipf & Welling, 2016) as a common baseline.

- **Dimension 1 (Edge Transformation):** *Linear Transformations*. Each GCN layer applies a shared linear transformation (weight matrix W) to the node features after they have been aggregated.

- **Dimension 2 (Activation):** *Nonlinear Activation* (e.g. ReLU) is typically applied after the linear transformation in each GCN layer, except possibly the final layer.
- **Dimension 3 (Bias Mechanism):** *Static Learned Bias* can be included with the linear transformation, although it's sometimes omitted in simpler GCN formulations.
- **Dimension 4 (Normalization):** This is a nuanced point. Feature normalization like Batch Norm is not standard in basic GCNs. However, a crucial aspect is the normalization of the graph structure itself, typically by pre-multiplying the adjacency matrix A with powers of the degree matrix D (e.g. $\hat{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$). This graph normalization acts as a form of "structural normalization" that directly influences the aggregation step rather than being a feature-wise normalization post-aggregation. So, for feature normalization: often *No Explicit Normalization*. For structural influence on aggregation: specific graph normalization.
- **Dimension 5 (Aggregation):** *Additive (Sum)*. In a GCN layer, features from neighboring nodes (including the node itself, often by adding self-loops to A) are summed, weighted implicitly by the normalized adjacency matrix, effectively performing a localized smoothing or message passing. The formula $\hat{A}HW$ can be seen as neighbors' features H being aggregated according to $\hat{A}$, then transformed by W. Graph Attention Networks (Veličković et al., 2018), as a different example here, use *Attentional (Similarity-Based Weighting)*.
- **Dimension 6 (Connectivity):** *Sparse (Selected Connections)*. The connectivity is explicitly defined by the input graph's adjacency matrix. Each node only connects to its direct neighbors as defined by the graph edges.
- **Dimension 7 (Info. Flow Dir.):** *Acyclic (Feedforward)* across GCN layers.
- **Dimension 8 (Shortcut Strategy):** Basic GCNs typically have *No Shortcuts*, although GCN variants (ResGCNs (Chen et al., 2020a), GCNII (Chen et al., 2020b)) incorporate residual or identity connections to deal with over-smoothing.
- **Dimension 9 (Memory Structure):** *Stateless.*
- **Dimension 10 (Param. Sharing):** *Shared.* The weight matrix W for the linear transformation is shared across all nodes within a single GCN layer.

Discussion: GCNs leverage *Sparse Connectivity* (from the graph) and *Shared Parameters* to efficiently learn from graph data. The aggregation based on the (normalized) graph structure is key to their message-passing behavior. The lack of explicit shortcuts in basic GCNs can lead to over-smoothing in deeper models, motivating extensions.

A.8 Kolmogorov-Arnold Networks (KAN)

Drawing inspiration from the Kolmogorov-Arnold representation theorem (Kůrková, 1991), KANs (Liu et al., 2025) place learnable activation functions (splines) on the edges rather than fixed activations on nodes.

- **Dimension 1 (Edge Transformation):** *Nonlinear Transformations.* Each edge in a KAN implements a learnable univariate function, typically parameterized as a B-spline. This is the defining feature of KANs.
- **Dimension 2 (Activation Function Type - Node Level):** Often effectively *Linear Activation (Identity)* or even *No Activation (Passthrough)* at the node if the node's sole purpose is to sum the outputs of the incoming nonlinear edge transformations. Some KAN interpretations might also place a learnable univariate spline at the node after summation, which would make node activation also *Nonlinear.* This dimension depends on the specific KAN layer formulation. For clarity, we consider the primary learnable functions on edges, making node function minimal (summation).

- **Dimension 3 (Bias Mechanism):** The expressiveness of the learnable spline functions on edges can implicitly handle offsets, potentially making separate traditional bias terms unnecessary. Bias can be considered *Embedded within the edge transformation* or effectively *No (explicit) Bias*.
- **Dimension 4 (Normalization Strategy):** Not an intrinsic part of the core KAN definition proposed so far. Standard normalization techniques could potentially be applied to the outputs of KAN layers if deemed beneficial for training deeper KANs. Thus, often *No Explicit Normalization* as a core KAN component.
- **Dimension 5 (Aggregation Mechanism):** *Additive (Sum)*. Nodes in a KAN layer sum the outputs from all incoming edges, each of which has already applied its learnable nonlinear spline function.
- **Dimension 6 (Connectivity Pattern):** Typically presented as *Dense (Fully Connected)* between layers, where each connection is a learnable spline function. KANs also discuss making these splines sparse dynamically.
- **Dimension 7 (Info. Flow Dir.):** *Acyclic (Feedforward)*.
- **Dimension 8 (Shortcut Strategy):** The original KAN proposal does not focus on explicit shortcuts, so typically *No Shortcuts*. However, shortcut strategies could be incorporated into KAN architectures.
- **Dimension 9 (Memory Structure):** *Stateless*.
- **Dimension 10 (Param. Sharing):** *Independent*. Each edge has its own independently learnable spline function, leading to a potentially large number of parameters compared to MLPs if the spline grid is fine.

Discussion: KANs shift complexity by using *Nonlinear Edge Transformations* (learnable splines) while simplifying node operations to mere summation. This design choice, contrasting with MLPs that have linear edges and nonlinear nodes, offers potential advantages in interpretability and accuracy for certain function classes. Their *Independent* parameterization of edge splines is a key characteristic affecting parameter count and learning dynamics.

A.9 U-Net

U-Net (Ronneberger et al., 2015) is a convolutional network architecture designed originally for biomedical image segmentation, characterized by its symmetric encoder-decoder structure and skip connections. Its architecture has proven highly effective both in and outside of segmentation and is now a foundational component in many diffusion models (Ho et al., 2020; Rombach et al., 2021) for generative tasks.

- **Dimension 1 (Edge Transformation):** Primarily *Linear Transformations* from convolutional operations in both the encoder (contracting path) and decoder (expansive path). Upsampling operations in the decoder (e.g. transposed convolutions) are also linear.
- **Dimension 2 (Activation):** *Nonlinear Activation* (typically ReLU) applied after each convolution, except possibly the final output layer which might use Sigmoid for binary segmentation or Softmax for multi-class segmentation.
- **Dimension 3 (Bias Mechanism):** *Static Learned Bias* is typically used in convolutional layers, subject to Batch Normalization.
- **Dimension 4 (Normalization):** Commonly uses *Batch-wise Normalization* after convolutional layers in modern implementations to stabilize training. The original U-Net did not explicitly use it.

- **Dimension 5 (Aggregation):** *Additive (Weighted Sum)* within convolutional operations. Feature maps from the encoder path are often concatenated with upsampled feature maps in the decoder path before further convolutions; this concatenation itself is a form of bringing features together, though the subsequent convolution does the weighted sum.
- **Dimension 6 (Connectivity):** *Sparse* (local convolutions) and distinctly *Hierarchical* due to the encoder's downsampling (pooling or strided convolutions) and the decoder's upsampling. The skip connections provide direct links between corresponding levels of the encoder and decoder.
- **Dimension 7 (Info. Flow Dir.):** *Acyclic (Feedforward).*
- **Dimension 8 (Shortcut Strategy):** Critically employs *Concatenative Shortcuts* (though sometimes Additive after channel alignment) via its skip connections. These link feature maps from the encoder path to the corresponding level in the decoder path, allowing the decoder to access high-resolution features from the encoder that are lost during downsampling.
- **Dimension 9 (Memory Structure):** *Stateless.*
- **Dimension 10 (Param. Sharing):** *Shared* parameters (convolutional kernels are shared across spatial locations within their respective layers).

Discussion: U-Net's effectiveness in segmentation stems from its *Hierarchical* encoder-decoder structure combined with *Concatenative Shortcuts*. These shortcuts allow the network to combine coarse, semantic information from deeper layers with fine-grained, spatial information from shallower layers for precise localization. These same structural properties that lets the U-Net progressively refine representations are applied in diffusion models, where it iteratively denoises an input by effectively predicting and removing noise across multiple feature scales.

A.10 Self-Supervised Learning

Models like SimCLR (Chen et al., 2020c), MoCo (He et al., 2020), and other self-supervised learning (SSL) methods (Chen & He, 2021; Bardes et al., 2022; Zbontar et al., 2021; Grill et al., 2020; Balestriero et al., 2023) have achieved remarkable success in learning visual representations without human-provided labels. Here, we also analyze these through our dimensional framework to clarify the scope of architectural choices versus training paradigms.

Consider that a typical SimCLR setup involves:

1. A base encoder network (e.g. a ResNet).
2. A small projection head (an MLP) attached to the encoder.
3. A specific data augmentation pipeline that creates two augmented "views" of each image.
4. A contrastive loss function that pulls representations of positive pairs (two views of the same image) together and pushes representations of negative pairs (views from different images) apart.

We can analyze the **base encoder network and projection head** using our dimensions, assuming a ResNet encoder and a 2-layer MLP projector, as is common:

- **Dimension 1-10 (for the ResNet encoder):** The dimensional choices are identical to those detailed in Section A.4 for a standard ResNet.
- **Dimension 1-10 (for the MLP projection head):** The dimensional choices are identical to those detailed in Section A.2 for a standard MLP.

Discussion: The insight from this dimensional analysis is that the core *architectural components* of SimCLR (the encoder and projector) are constructed from well-understood architectural choices already covered by our framework. The ResNet encoder's capacity to learn good features is vital, as is the projector's role in mapping features to a space suitable for the contrastive loss.

However, the novelty and power of SimCLR do not primarily arise from a new fundamental *architectural dimension* within the encoder or projector itself. Instead, they stem from:

- **Data Processing and Augmentation:** The specific augmentations used to create positive pairs are critical. This is an input-level consideration, external to the core network structure that processes those inputs.
- **Training Objective and Loss Function:** The contrastive loss (e.g. NT-Xent in SimCLR) is what drives the representation learning. This defines the optimization goal, not an intrinsic structural property of the processing network.
- **Overall Training Framework:** The setup where two views are processed (usually by the same shared-weight encoder), and the mechanisms for defining positive/negative pairs (e.g. large batch sizes for negatives in SimCLR, or a memory bank/momentum encoder in MoCo).

These aspects like data augmentation, loss functions, and specific training procedures (like momentum updates for a key encoder in MoCo) are part of the broader *learning paradigm* rather than core, reusable structural dimensions of the network module that transforms input x to representation $h(x)$. Our framework aims to systematically characterize the $h(x)$ function's structure itself.

Therefore, while SSL methods like SimCLR might employ networks with specific *configurations* of our dimensions (e.g. a deep ResNet for good feature capacity), the principles of SSL itself (contrastive learning, data augmentation policies) are distinct considerations. This clarifies that our framework focuses on the intrinsic building blocks and patterns of the neural network performing the information processing, separate from, though often designed to be complementary to, the overarching training methodology and learning objectives. This distinction is important for maintaining a focused and coherent taxonomy of architectural design.

A.11 Neural Ordinary Differential Equations (Neural ODEs) (Chen et al., 2018; Kidger, 2022)

Neural ODEs introduce a continuous-depth perspective to neural networks, where the hidden state dynamics are defined by an ordinary differential equation (ODE) parameterized by a neural network. Instead of a discrete sequence of layers, a Neural ODE layer defines a vector field, and the output is computed by integrating this vector field over a specified time interval.

Let $h(t)$ be the hidden state at "depth" or "time" t. A Neural ODE layer is defined by:

$$\frac{dh(t)}{dt} = f(h(t), t, \theta)$$

where f is a neural network (often an MLP) with parameters θ. The output of the layer, $h(t_1)$, is obtained by integrating this equation from an initial state $h(t_0)$ (which could be the input to the layer) to t_1:

$$h(t_1) = h(t_0) + \int_{t_0}^{t_1} f(h(t), t, \theta)dt$$

This integration is typically performed by an adaptive numerical ODE solver.

Let's analyze the **dynamics function** $f(h(t), t, \theta)$ **itself**, and then consider the overall Neural ODE layer.

A.11.1 Dimensional Analysis of the Dynamics Function f

Assuming f is a standard MLP (as is common):

- **Dimension 1-6, 10 (for the MLP f):** The dimensional choices for f are typically those of a standard MLP (Section A.2).
- *Within the context of parameterizing f, the choices are for a standard, discrete-depth network component.*

A.11.2 Dimensional Analysis of the Overall Neural ODE Layer

Now, we consider the entire construct $h(t_1) = \text{ODESolve}(f, h(t_0), [t_0, t_1], \theta)$:

- **Dimension 1 (Edge Transformation):** This is abstract. The "transformation" from $h(t_0)$ to $h(t_1)$ is via the ODE integration, driven by the learned vector field f. One could argue it's a highly *Nonlinear Transformation*, as ODE solutions are generally nonlinear functions of their initial conditions and parameters, even if f has linear components. There are no discrete "edges" in the traditional sense for the ODE solve itself.
- **Dimension 2 (Activation):** Not directly applicable to the ODE solve as a monolithic "node." The nonlinearities are within the dynamics function f. The integration process itself is not a simple node activation.
- **Dimension 3 (Bias Mechanism):** Biases are present within f. No overarching bias is typically added after the ODE solve.
- **Dimension 4 (Normalization):** Normalization techniques are applied within f if anywhere, or to $h(t_0)$ before the solve, or to $h(t_1)$ after. The ODE solve itself doesn't have an intrinsic normalization strategy.
- **Dimension 5 (Aggregation):** Not directly applicable in the standard sense for the ODE integration. The "aggregation" of infinitesimal changes dh happens via the mathematical process of integration. Another angle is that ODEs smooth out changes across the entire interval. This continuous nature is different from discrete aggregation (like pooling or attention), though related at a high level.
- **Dimension 6 (Connectivity):** This applies to the structure of f. For the overall ODE layer, connectivity is conceptual: the entire state $h(t)$ is evolved. One might consider it *Dense* in the sense that all components of $h(t)$ influence all components of dh/dt through f.
- **Dimension 7 (Info. Flow Dir.):** This is interesting. The integration from t_0 to t_1 is a continuous, directed evolution. If f does not depend on an external sequence input but only $h(t)$ and t, the "depth" t acts like a continuous feedforward direction. However, Neural ODEs can be viewed as the continuous limit of a ResNet (where f is the residual block) or an RNN (if f also takes an external input $x(t)$ and describes dh/dt). This really depends on what the f is set up as. (w.r.t. the intended use case and the structure of the arguments to f)
  - If viewed as a continuous-depth feedforward network: *Acyclic (Feedforward)* (in continuous "depth" t).
  - If $h(t)$ is an evolving state meant to model sequential data (like an RNN): then the "time" of the ODE evolution maps to sequence time, and if it models $h(t)$ based on $h(t - \delta t)$ implicitly, it is effectively *Cyclic (Recurrent)* in continuous time.
- **Dimension 8 (Shortcut Strategy):** The formulation $h(t_1) = h(t_0) + \int ...$ is inherently residual. The output is the initial state plus the integral of changes. Thus, it embodies a continuous form of an *Additive Shortcut (Residual)*. This is a reason for their good gradient properties.
- **Dimension 9 (Memory Structure):** If processing a single input $h(t_0)$ to get $h(t_1)$ in one go: *Stateless* (the parameters θ are fixed). If used to model a sequence where $h(t_i)$ becomes the input for the next segment evolution $h(t_{i+1})$: *Stateful*.

- **Dimension 10 (Param. Sharing):** The parameters θ of the function f are *Shared* across the entire continuous "depth" or integration interval $[t_0, t_1]$. This is analogous to weight sharing across layers in very deep ResNets or across time in RNNs.

Discussion: Neural ODEs provide a fascinating case for our dimensional framework. While the dynamics function f itself is a standard neural network analyzable by our dimensions, the overall Neural ODE layer reinterprets several dimensions in a continuous context:

- The discrete notions of "layers," "edges," and "node activations" become blurred into the continuous evolution defined by the ODE.
- *Information Flow Direction* and *Memory Structure* depend on how the Neural ODE is utilized (as a continuous-depth ResNet analogue vs. a continuous-time RNN analogue).
- The *Shortcut Strategy* (Additive/Residual) becomes an intrinsic property of the ODE formulation itself ($h(t_0) + \int ...$).
- *Parameter Sharing* across the continuous "depth" (integration path) is fundamental.

This shows the framework's adaptability: it can describe the components parameterizing unconventional structures (like f) and also offer a lens to interpret the behavior of the larger, more abstract computational block (the ODE solve), even if some dimensions apply more conceptually. The core idea is that f learns a vector field, and the overall "layer" is an operation (ODE integration) on that field; the framework describes f precisely, and the ODESolve implicitly defines others like the residual shortcut.

A.12 Energy-Based Models (EBMs)

Energy-Based Models (LeCun et al., 2006; Teh et al., 2003; Xie et al., 2016) flexibly represents dependencies by associating a scalar energy $E_\theta(X)$ to a configuration of variables X. Learning involves shaping this energy landscape such that desirable configurations (e.g. valid data points) have low energy and undesirable ones have high energy. Inference often requires optimization or sampling procedures (e.g. MCMC) to find low-energy configurations.

Our dimensional analysis will focus on the **neural network that parameterizes the energy function $E_\theta(X)$ itself**, which we refer to as the "Energy Network". This network takes X (which could be an image x, a pair (x, y), etc.) as input and outputs a single scalar energy value. The architecture of this Energy Network can vary greatly (e.g. a CNN for images, an MLP for structured data).

Assume for this study that the Energy Network is a ResNet-style CNN when X is an image x:

- **Dimension 1-8, 10 (for the ResNet-style Energy Network, excluding final output):** If the core of the Energy Network is a ResNet that processes an input X to produce a high-dimensional feature vector, its dimensional choices up to that point would be similar to those detailed in Section A.4.
- **Transformation to Scalar Energy:** To produce the final scalar energy value, the feature vector from the ResNet core is typically passed through:
  - An *Aggregation Mechanism (Dimension 5)* like Global Average Pooling (a *Statistical* aggregation) if the features are spatial.
  - Followed by one or more *Dense (Fully Connected) Layers (Dimension 6)* with *Linear Edge Transformations (Dimension 1)* and possibly *Nonlinear Activations (Dimension 2)* in hidden FC layers.
  - The final output layer mapping to the scalar energy uses a *Linear Edge Transformation* and critically, a *Linear Activation (Dimension 2)* or no activation, as energy is an unconstrained scalar.

  - *Parameter Sharing (Dimension 10)* for these final FC layers is typically *Independent*.

- **Dimension 9 (Memory Structure):** The Energy Network itself, when computing $E_\theta(X)$ for a given X, is typically *Stateless*. The "memory" in EBMs might arise during inference if sampling methods like MCMC maintain a state (the current sample), but this is part of the inference algorithm, not the Energy Network's intrinsic architecture for computing $E_\theta(X)$.

Discussion: The dimensional analysis of an EBM primarily concerns the architecture of the "Energy Network" that defines $E_\theta(X)$. This network itself can be any standard (or unconventional) architecture describable by our framework. For instance, an EBM for images often uses a CNN (like a ResNet without the final classification head, adapted to output a scalar).

Key points related to EBMs and the dimensional framework:

- **Architecture of $E_\theta(X)$:** The dimensions fully apply to the design of the neural network component that computes the energy function. The choice of this architecture (e.g. depth, width, connectivity, normalization within the Energy Network) is crucial for the EBM's ability to model a complex energy landscape.
- **Output is a Scalar:** The final layers of the Energy Network are constrained to produce a single, unnormalized scalar value. This typically means a final Linear Edge Transformation with a Linear (Identity) Activation at the output node.
- **Separation from Inference/Learning Algorithm:** The architectural dimensions describe $E_\theta(X)$. The methods used to *learn* θ (e.g. contrastive divergence, score matching) or to *sample* from $p(X) \propto \exp(-E_\theta(X))$ (e.g. Langevin dynamics, MCMC) are algorithms that use the Energy Network but are not part of its intrinsic architectural structure as defined by our dimensions. For instance, Langevin dynamics uses $\nabla_X E_\theta(X)$, which requires $E_\theta(X)$ to be differentiable with respect to its inputs X, a property influenced by choices like Activation Function Type within the Energy Network.
- **No Direct "Flow" to Output Data (for unconditional EBMs):** Unlike generative models like GANs (Goodfellow et al., 2020) or VAEs (Kingma & Welling, 2013) which have a network that directly generates a sample, an unconditional EBM defines a landscape. "Generating" a sample involves finding a low-energy configuration, often an iterative process. This contrasts with architectures defined primarily by their forward data transformation path from input to a structured output.

Thus, the dimensional framework is effective in characterizing the structure of the neural network component within EBMs, while also clarifying that the broader EBM paradigm involves distinct learning and inference mechanisms that operate upon the energy function defined by this network.

A.13 Neural Networks in Reinforcement Learning Agents

Reinforcement Learning (RL) encompasses a broad class of problems and algorithms where an agent learns to make decisions in an environment to maximize a cumulative reward signal (Sutton & Barto, 1998). Neural networks are widely used in Deep RL to approximate various components of an RL agent, such as the policy, value function, or environment model. Our dimensional framework applies to these neural network components themselves, rather than to the RL problem or learning algorithm (e.g. Q-learning (Watkins & Dayan, 1992), Policy Gradients (Sutton et al., 1999), Actor-Critic methods (Konda & Tsitsiklis, 1999)) as a whole.

We can consider common neural network components in Deep RL:

A.13.1 1. Policy Networks (e.g. in A2C/A3C (Mnih et al., 2016), PPO (Schulman et al., 2017))

A policy network $\pi_\theta(a|s)$ maps an environment state s to a distribution over actions a (for discrete actions) or to parameters of a continuous action distribution.

- **Input Processing (State Representation):** If the state s is an image, the initial layers are often a *CNN* (analyzable as per Section A.3 or A.4). If s is a feature vector, an *MLP* (Section A.2) is common. Thus, Dimensions 1-6, 8, 10 specific to CNNs/MLPs apply.
- **Output Head:** The final layers typically transform the learned state representation into action logits or distribution parameters.
  - *Dimension 1 (Edge Transformation): Linear Transformation* for the final layer.
  - *Dimension 2 (Activation):* For discrete actions, often *No Activation (Identity Passthrough)* before a Softmax function (which normalizes logits to probabilities; Softmax itself can be seen as a special activation or a post-processing step for distributions). For continuous actions, activations like Tanh might be used to bound actions, or Linear if outputting means/std. devs.
  - Other MLP-like dimensions apply for hidden layers in the output head.
- **Dimension 7 (Info. Flow Dir.):** Typically *Acyclic (Feedforward)* from state to action/action-distribution parameters.
- **Dimension 9 (Memory Structure):** Often *Stateless* if operating on the current state. If recurrent layers (e.g. LSTMs, Section A.5) are used to process a sequence of observations to form the current effective state (e.g. in Partially observable Markov decision processes (POMDPs)), then it becomes *Stateful.*
- **Dimensions 3, 4** would be chosen based on the specific MLP/CNN backbone.

A.13.2 2. Value Networks / Q-Networks (e.g. in DQN (Mnih et al., 2015), A2C/A3C, SAC (Haarnoja et al., 2018))

A Q-network $Q_\theta(s, a)$ estimates the expected cumulative reward for taking action a in state s. A value network $V_\theta(s)$ estimates the expected cumulative reward from state s.

- **Input Processing:**
  - For $V_\theta(s)$: Similar to policy networks, using a CNN/MLP backbone for state s.
  - For $Q_\theta(s, a)$: If a is discrete, a common architecture processes s with a CNN/MLP backbone, then has separate output heads (Linear Transformations) for each action's Q-value. If a is continuous, s and a might be concatenated and fed into an MLP.
- **Output Head (for $V(s)$ or each $Q(s, a_i)$):** A final *Linear Edge Transformation* with a *Linear Activation* (Identity), as Q-values and state values are unconstrained scalars.
- Other dimensions for the backbone and any intermediate layers align with standard CNN/MLP choices. For example, DQN often uses CNNs similar to LeNet/AlexNet for processing game screens.

A.13.3 3. Model-Based RL: Environment Models (Ha & Schmidhuber, 2018; Racanière et al., 2017)

Some RL agents learn a model of the environment $p(s', r|s, a)$, which predicts the next state s' and reward r given the current state s and action a. The network parameterizing this model can again be an MLP, CNN (if states are images), or a recurrent network if history is important. Its dimensional choices would align with these architectures, with specific output heads for predicting s' and r.

Discussion: The neural networks utilized within RL agents are generally constructed from standard architectural building blocks described by our dimensions. The choices for these dimensions (e.g. using a CNN for image states, an LSTM for partial observability) are motivated by the nature of the state and action spaces and the complexity of the function being approximated (policy, value, or model).

Key points regarding RL and the dimensional framework:

- **Architectural Components are Analyzable:** The policy network, value network, and learned environment models are all neural networks whose specific structures can be fully characterized by the 10 dimensions.
- **RL Algorithm is External:** The learning algorithm (e.g. REINFORCE, DQN's Bellman error minimization, TRPO's constrained policy updates) dictates how the network parameters θ are updated based on experience (trajectories of states, actions, rewards). This learning process is external to the intrinsic static architecture of the network component.
- **Input-Output Structure Driven by RL Task:** The specific nature of inputs (states, state-action pairs) and outputs (action probabilities, Q-values, next-state predictions) is determined by the RL component being modeled. The network architecture is then chosen to effectively map these inputs to outputs.
- **Exploration vs. Exploitation:** Strategies for exploration (e.g. epsilon-greedy, adding noise to actions) are part of the agent's decision-making process, not an architectural dimension of the policy or value network itself (though some works explore architectural methods to intrinsically promote exploration, these would still be describable components).

Therefore, our framework describes the "neural machinery" an RL agent employs, while the principles of reinforcement learning itself govern how this "machine" is trained and used to interact with an environment. For instance, an actor-critic method uses two networks (actor/policy, critic/value), each of which is a specific configuration of our dimensions, and the actor-critic algorithm orchestrates their interaction and learning.