WARP-LUTs: Walsh-Assisted Relaxation for Probabilistic Look Up Tables

Lino Gerlach
Princeton University
lg0508@princeton.edu

Liv Våge
Princeton University
lv7805@princeton.edu

Thore Gerlach
University of Bonn
tgerlac1@uni-bonn.de

Elliott Kauffman
Princeton University
ek8842@princeton.edu

Isobel Ojalvo
Princeton University
iojalvo@princeton.edu

Abstract

Fast machine learning is of growing interest to the scientific community and has spurred significant research into novel model architectures and hardware-aware design. Recent hard- and software co-design approaches have demonstrated impressive results with entirely multiplication-free models, such as Differentiable Logic Gate Networks (DLGNs). However, these models suffer from high computational cost during training and do not generalize well to logic blocks with more inputs. In this work, we introduce Walsh-Assisted Relaxation for Probabilistic Look-Up Tables (WARP-LUTs)—a novel gradient-based method that efficiently learns combinations of logic gates with substantially fewer trainable parameters. We demonstrate that WARP-LUTs achieve significantly faster convergence on CIFAR-10 compared to DLGNs, while maintaining comparable accuracy. Furthermore, our approach suggests potential for extension to higher-input logic blocks, motivating future research on extremely efficient deployment on modern FPGAs and its real-time science applications.

1 Introduction

Deep learning [1] has become the standard for a wide range of tasks in science, but its success comes with heavy computational costs in both training and inference. This restricts deployability in many real-world settings—particularly in domains where ultra-fast inference is critical, such as particle physics [2], gravitational wave astronomy [3], and quantum computing [4]. These constraints have motivated substantial research into the development of models that maintain predictive accuracy while improving computational efficiency. Prominent among such efforts are model compression techniques, which encompass approaches for inducing sparsity [5, 6], pruning [7, 8], and reducing numerical precision via parameter quantization [9–11].

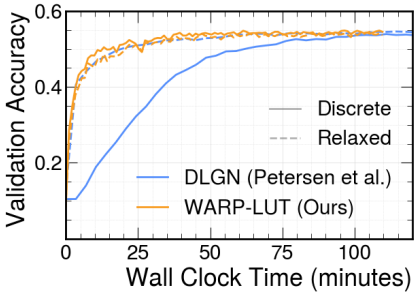

Figure 1: Validation accuracy on CIFAR-10. Logic gate activations are represented as continuous numbers (relaxed) during training and discrete values during inference. Our implementation shows faster convergence compared to the baseline.

Nevertheless, such approaches do not directly address the intrinsic computational cost of numerical multiplication. To overcome this limitation, multiplication-free architectures have been proposed, including binary neural networks [12, 13] and other bit-level

1st Workshop on Differentiable Systems and Scientific Machine Learning @ EurIPS 2025.

summation models [14–16]. However, efficient inference on logic gate–based hardware requires mapping such abstract computations into executable logic, which incurs significant overhead.

Within the domain of multiplication-free models, Weightless Neural Networks (WNNs) [17] represent a distinct class that overcomes this limitation. Instead of relying on weighted connections, WNNs employ look-up tables (LUTs) with binary values to drive neural activity during inference, allowing them to capture highly nonlinear behaviors while avoiding arithmetic operations altogether. Although state-of-the-art models achieve remarkable performance on small-scale tasks, their scalability remains constrained—whether due to limited expressiveness [18], reliance on gradient approximations [19], double-exponential growth in parameter requirements [20, 21], or large discretization gaps [22, 23], describing an accuracy mismatch between the training using relaxations and discrete inference.

We overcome these issues by proposing **Walsh-Assisted Relaxation for Probabilistic LUTs (WARP-LUTs)**, based on differentiable relaxation similar to Differentiable Logic Gate Networks (DLGN) [20]. Publishing our code will bridge the gap created by the lack of publicly available convolutional DLGN implementations for the community [23]. Our contributions can be summarized as follows:

- We propose representing Boolean functions with the Walsh–Hadamard (WH) transform [24], providing a compact and differentiable parameterization of a deep WNN enabling an exponential reduction in the number of parameters compared to DLGNs.
- Through single-exponential parameter scaling in LUT-input size, this not only increases expressiveness [25], but also maps directly to modern FPGA primitives, such as LUT-6 blocks, promising more efficient deployment on such hardware.
- A Gumbel reparameterization scheme is presented. This yields a smoother loss landscape, better alignment between training and inference, faster convergence, and a reduced discretization gap.
- Experiments show the versatility and efficacy of our approach compared to previous methods.

2 Related Work

Early work on WNNs explored single-layer architectures that replace weighted connections with LUTs, enabling extremely efficient inference [18, 26]. However, their limited expressiveness and lack of effective training methods for deeper architectures have restricted their use in complex tasks.

To address these limitations, DLGNs represent Boolean functions as differentiable two-input gates, enabling gradient-based training of multi-layer architectures [20, 21]. Advances in connection learning [27–29] have improved flexibility, but training DLGN architectures introduces a discretization gap between continuous training and discrete inference. Recent work has sought to bridge this gap through stochastic relaxation [22] and forward-backward alignment [23].

However, DLGNs remain restricted to two-input gates and suffer from double-exponential parameter growth, limiting scalability. To account for higher-degree LUTs, the authors of [19] extend LUT-based models with differentiable training, but rely on gradient approximations that can affect performance. Finally, alternative approaches move beyond neural networks, such as TreeLUT [30], which combines gradient-boosted trees with LUT mappings for efficient inference, and Truth Table Net [31], which leverages Boolean circuit structures for scalability and verifiability.

3 Methodology

Any Boolean function $f : \{0,1\}^n \to \{0,1\}$ is uniquely represented by its truth-table. Since the input- and output space have size 2^n and 2, respectively, there are $2^{(2^n)}$ boolean functions. Each one can be uniquely represented in the WH basis by first re-encoding its inputs from $\{0,1\}$ into signed variables $\{-1,+1\}$, according to the mapping $B : \{0,1\} \to \{-1,+1\}$, $B(0) = -1$, $B(1) = +1$. The resulting function takes the form

$$f(\mathbf{x}) = \operatorname{sign}\left(l_{\mathbf{c},B}(\mathbf{x})\right), \quad l_{\mathbf{c},B}(\mathbf{x}) = \sum_{S \subseteq \{1,\ldots,n\}} c_S \prod_{i \in S} B(x_i), \tag{1}$$

with $B(x_i) \in \{-1, +1\}$ describing the transformed inputs. The 2^n-dimensional vector of all WH coefficients can be calculated from the orthogonal transform $\mathbf{c} = \frac{1}{2^n} H f_{\pm} \in \frac{1}{2^n} \mathbb{Z}^{2^n}$, where $f_{\pm} \in \{\pm 1\}^{2^n}$ is the $\pm 1-$valued truth vector and H is the $2^n \times 2^n$ Hadamard matrix. Intuitively, this expansion uses simple polynomial basis functions—individual variables, pairwise products, and higher-order interactions—to capture the structure of the Boolean function. Hence the WH representation provides a compressed and structured parametrization of Boolean functions, where the 2^{2^n} Boolean functions are in bijection with a finite subset of this 2^n-dimensional lattice.

Example: 2-input logic gates In the special case of two inputs (a, b), every binary logic gate admits a decomposition with only four coefficients:

$$f(a, b) = \text{sign}\big(c_0 + c_1 a + c_2 b + c_3 (a \cdot b)\big),$$

where c_0 encodes the constant bias (tendency toward 0 or 1), c_1 and c_2 encode dependence on the individual inputs, and c_3 encodes the interaction term between the inputs. For example, the coefficients $(c_0, c_1, c_2, c_3) = (0, 0, 0, -1)$ correspond to the XOR gate, while the AND gate can be expressed as $(c_0, c_1, c_2, c_3) = (-\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. This decomposition demonstrates that instead of enumerating all 16 binary gates explicitly, one can parameterize them compactly with just four WH coefficients (see Table 1 in Sec. B for the full list of gates and coefficients).

Relaxation We propose a relaxation of the WH transform-based decomposition that allows to differentiate w.r.t. its coefficients using the following three components: (i) Extend the component-wise basis transform B to real-valued inputs, $\tilde{B} : [0, 1] \to [-1, 1]$, $\tilde{B}(x) = 2x - 1$, (ii) generalize the discrete WH coefficients c_S to continuous, real-valued parameters $\tilde{c}_S$. This breaks bijectivity, creating a surjective mapping from the parameter space to all binary functions, as any truth table can now be represented by infinite combinations of generalized WH coefficients and (iii) Approximate the sign function with the sigmoid function in Eq. (1), leading to $\tilde{f}(\mathbf{x}) = \sigma\left(l_{\mathbf{c}, \tilde{B}}(\mathbf{x})/\tau\right)$, where τ is the *temperature* parameter that controls the smoothness of the relaxation. This decomposition provides a compact and differentiable parameterization with 2^n parameters, while still allowing to collapse back into one of the 2^{2^n} exact Boolean gates at inference time by choosing the closest truth table.

Reducing Discretization Gap To bridge the discretization gap between the continuous relaxation used during training and the hard decisions required during inference, we adopt Gumbel–Sigmoid reparameterization, a binary variant of the Gumbel–Softmax (Concrete) distribution, enabling gradient-based training with discrete variables [32, 33]. Given a logit $l(x)$ and Gumbel noises $g_1, g_2 \sim \text{Gumbel}(0, 1)$, relaxed samples are computed as $\tilde{f}_{\text{Gumbel}}(x) = \sigma\left((l(x) + g_1 - g_2)/\tau\right)$. Discretization to a specific LUT can be obtained by deciding whether $\tilde{f}(x), \tilde{f}_{\text{Gumbel}}(x) < 0.5$ which leads to a binary output. In our experiments, we also evaluated the performance of using this discretization already during training in the forward pass as a straight-through estimator.

This approach is motivated by two key observations. First, injecting Gumbel noise during the forward pass induces stochastic smoothing, implicitly penalizing curvature and favoring flatter minima, which is known to improve generalization [23]. Second, unlike DLGNs, which optimize a surrogate objective misaligned with inference, Gumbel-based training mitigates this discrepancy.

4 Experiments

For our experiments, we restrict our study to the case of binary logic gates with two inputs, consistent with the setup considered in prior work. We evaluate model performance on the CIFAR-10 dataset, following standard pre-processing and an 80/20 split in training / validation data. The architectures of the models used are shown in Sec. C.

In Fig. 1, we adopt the "large" model architecture introduced in the original DLGN paper [20] as the baseline. This MLP-style model contains 20,480,000 trainable parameters, while our WARP-LUT model based on the same architecture variant reduces this to 5,120,000 parameters. Both models were trained for 50,000 steps, albeit each WARP-LUT training step takes roughly one third of the wall clock time on an A100 GPU. After every 500 steps, we evaluate the models' performance on the validation set, both in the differentiable approximation ('Relaxed') used for training, as well as in

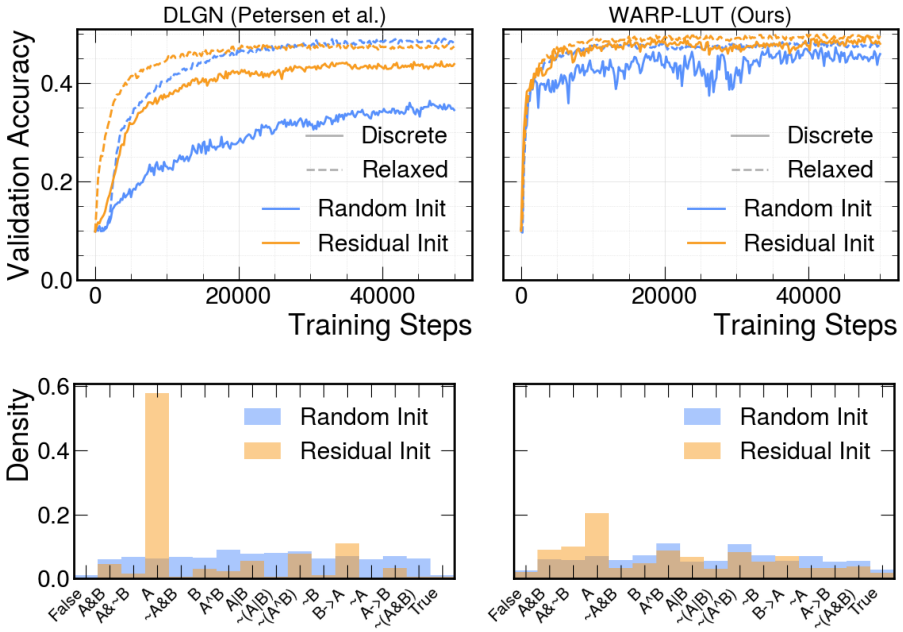

Figure 2: Results of a very small (35968 gates) convolutional model on CIFAR-10 for DLGN (left) and WARP-LUT (right). We compare the validation accuracy, where the solid line shows the discrete mode and the dashed line shows the relaxed mode (top) along with the distribution of logic gates after training (bottom). Blue depicts random weight initialization, orange residual weight initialization.

its binarized mode ('Discrete'), where the most likely binary logic gate is plugged into each node. Overall, our implementation achieves a comparable final accuracy to the baseline while requiring only a quarter of the parameters and exhibiting significantly reduced resource utilization during training.

In Fig. 2, we show results for a very small convolutional model with only 575,488 and 143,872 trainable parameters in the DLGN-based and WARP-LUT implementation, respectively. We also compare random- and residual initialization of weights, where the latter gives stronger initial probability for the identity gate, which helps propagating gradients through many layers [21]. It can be seen that residual initialization helps both, DLGN and WARP-LUTs converge faster. In any case, WARP-LUTs converge in significantly fewer training steps than the baseline. Note that each training is also computationally less expensive (not factored into this plot). Unlike for WARP-LUTs, residual initialization for DLGNs also results in significantly more identity gates after the training, which should lead to a smaller model when deployed on hardware. However, this effect might disappear when running the training long enough for DLGN and WARP-LUT accuracy to match.

5 Limitations and Future Work

With access to larger computational resources, it would be valuable to investigate how WARP-LUTs perform for substantially larger models (on the order of one billion parameters) and across different architectures. All results in this paper were obtained using PyTorch's default GPU support, without any dedicated CUDA kernels. Exploring optimized kernels for both DLGNs and WARP-LUTs will provide a more realistic assessment of their respective training efficiencies.

A key motivation for this work lies in scalability. While DLGNs scale doubly exponentially with the number of gate inputs, WARP-LUTs scale only exponentially, making it possible to learn four-input logic blocks with only 16 trainable parameters per node instead of roughly 65,000. If successful, this would open the door to even more efficient deployment on modern FPGAs, where multi-input look-up tables represent the fundamental computational units. Our method will be compared to other LUT-based state-of-the-art methods, such as TreeLUT [30] or Differentiable Weightless Neural Networks [19].

6 Acknowledgements

This work was supported by the National Science Foundation under Cooperative Agreements OAC-1836650 and PHY-2323298.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[2] Thea Aarrestad et al. "Fast convolutional neural networks on FPGAs with hls4ml". In: *Machine Learning: Science and Technology* 2.4 (July 2021), p. 045015. ISSN: 2632-2153. DOI: 10.1088/2632-2153/ac0ea1. URL: http://dx.doi.org/10.1088/2632-2153/ac0ea1.

[3] Ana Martins et al. *Improving early detection of gravitational waves from binary neutron stars using CNNs and FPGAs*. 2025. arXiv: 2409.05068 [astro-ph.IM]. URL: https://arxiv.org/abs/2409.05068.

[4] Madhav Narayan Bhat et al. *Machine Learning for Arbitrary Single-Qubit Rotations on an Embedded Device*. 2024. arXiv: 2411.13037 [quant-ph]. URL: https://arxiv.org/abs/2411.13037.

[5] Yi-Lin Sung, Varun Nair, and Colin A Raffel. "Training neural networks with fixed sparse masks". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 24193–24205.

[6] Torsten Hoefler et al. "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks". In: *Journal of Machine Learning Research* 22.241 (2021), pp. 1–124.

[7] Shaohui Lin et al. "Accelerating Convolutional Networks via Global & Dynamic Filter Pruning." In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*. Vol. 2. 7. Stockholm. 2018, p. 8.

[8] Shiwei Liu et al. "The Unreasonable Effectiveness of Random Pruning: Return of the Most Naive Baseline for Sparse Training". In: *Proceedings of the 10th International Conference on Learning Representations (ICLR)*. OpenReview. 2022.

[9] Chang Sun et al. "Gradient-based automatic mixed precision quantization for neural networks on-chip". In: *arXiv preprint arXiv:2405.00645* (2024).

[10] Amir Gholami et al. "A survey of quantization methods for efficient neural network inference". In: *Low-Power Computer Vision*. Chapman and Hall/CRC, 2022, pp. 291–326.

[11] Brian Chmiel et al. "Neural gradients are near-lognormal: improved quantized and sparse training". In: *Proceedings of the 9th International Conference on Learning Representations (ICLR)*. 2021.

[12] Itay Hubara et al. "Binarized neural networks". In: *Advances in neural information processing systems* 29 (2016).

[13] Haotong Qin et al. "Binary neural networks: A survey". In: *Pattern Recognition* 105 (2020), p. 107281.

[14] Hanting Chen et al. "AdderNet: Do we really need multiplications in deep learning?" In: *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 1468–1477.

[15] Mostafa Elhoushi et al. "Deepshift: Towards multiplication-less neural networks". In: *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 2359–2368.

[16] Van Minh Nguyen et al. "BOLD: Boolean Logic Deep Learning". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 61912–61962.

[17] Igor Aleksander et al. "A brief introduction to weightless neural systems." In: *Proceedings of the 17th European Symposium on Artificial Neural Networks (ESANN)*. 2009, pp. 299–305.

[18] Zachary Susskind et al. "Uleen: A novel architecture for ultra-low-energy edge neural networks". In: *Transactions on Architecture and Code Optimization (TACO)* 20.4 (2023), pp. 1–24.

[19] Alan TL Bacellar et al. "Differentiable weightless neural networks". In: *Proceedings of the 41st International Conference on Machine Learning, PMLR*. Vol. 235. 2024, pp. 2277–2295.

[20] Felix Petersen et al. “Deep differentiable logic gate networks”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 2006–2018.

[21] Felix Petersen et al. “Convolutional differentiable logic gate networks”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 121185–121203.

[22] Youngsung Kim. “Deep stochastic logic gate networks”. In: *IEEE Access* 11 (2023), pp. 122488–122501.

[23] Shakir Yousefi et al. “Mind the Gap: Removing the Discretization Gap in Differentiable Logic Gate Networks”. In: *arXiv preprint arXiv:2506.07500* (2025).

[24] Kunz. “On the equivalence between one-dimensional discrete Walsh-Hadamard and multidimensional discrete Fourier transforms”. In: *IEEE Transactions on Computers* 100.3 (1979), pp. 267–268.

[25] Hugo CC Carneiro et al. “The exact VC dimension of the WiSARD n-tuple classifier”. In: *Neural computation* 31.1 (2019), pp. 176–207.

[26] Zachary Susskind et al. “Weightless neural networks for efficient edge inference”. In: *Proceedings of the 31st International Conference on Parallel Architectures and Compilation Techniques (PACT)*. 2022, pp. 279–290.

[27] Wout Mommen et al. “A Method for Optimizing Connections in Differentiable Logic Gate Networks”. In: *arXiv preprint arXiv:2507.06173* (2025).

[28] Chang Yue and Niraj K Jha. “Learning interpretable differentiable logic networks”. In: *IEEE Transactions on Circuits and Systems for Artificial Intelligence* (2024).

[29] Fabian Kresse, Emily Yu, and Christoph H Lampert. “Scalable Interconnect Learning in Boolean Networks”. In: *arXiv preprint arXiv:2507.02585* (2025).

[30] Alireza Khataei and Kia Bazargan. “TreeLUT: An Efficient Alternative to Deep Neural Networks for Inference Acceleration Using Gradient Boosted Decision Trees”. In: *Proceedings of the 2025 ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA)*. 2025, pp. 14–24.

[31] Adrien Benamira et al. “Truth table net: Scalable, compact & verifiable neural networks with a dual convolutional small boolean circuit networks form”. In: *Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI)*. 2024.

[32] C Maddison, A Mnih, and Y Teh. “The concrete distribution: A continuous relaxation of discrete random variables”. In: *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. International Conference on Learning Representations. 2017.

[33] Eric Jang, Shixiang Gu, and Ben Poole. “Categorical Reparameterization with Gumbel-Softmax”. In: *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. 2017.

A Source Code

Our source code is available in GitHub: https://github.com/ligerlac/torchlogix

B WH-Transform-based Decompositions for Two-Input Boolean Functions

Table 1: The 16 binary Boolean functions of two inputs, their truth tables (ordered as 00,01,10,11), and the corresponding WH coefficients, (c_0, c_1, c_2, c_3).

#	Gate	Formula	Truth table	WH coeffs
0	CONST0	0	0000	$(-1,\ 0,\ 0,\ 0)$
1	CONST1	1	1111	$(+1,\ 0,\ 0,\ 0)$
2	AND	$a \wedge b$	0001	$\left(-\frac{1}{2},\ \frac{1}{2},\ \frac{1}{2},\ \frac{1}{2}\right)$
3	OR	$a \vee b$	0111	$\left(\frac{1}{2},\ \frac{1}{2},\ \frac{1}{2},\ -\frac{1}{2}\right)$
4	XOR	$a \oplus b$	0110	$(0,\ 0,\ 0,\ -1)$
5	XNOR	$\neg(a \oplus b)$	1001	$(0,\ 0,\ 0,\ 1)$
6	NAND	$\neg(a \wedge b)$	1110	$\left(\frac{1}{2},\ -\frac{1}{2},\ -\frac{1}{2},\ -\frac{1}{2}\right)$
7	NOR	$\neg(a \vee b)$	1000	$\left(-\frac{1}{2},\ -\frac{1}{2},\ -\frac{1}{2},\ \frac{1}{2}\right)$
8	$a \wedge \neg b$	$a \wedge \neg b$	0010	$\left(-\frac{1}{2},\ \frac{1}{2},\ -\frac{1}{2},\ -\frac{1}{2}\right)$
9	$\neg a \wedge b$	$\neg a \wedge b$	0100	$\left(-\frac{1}{2},\ -\frac{1}{2},\ \frac{1}{2},\ -\frac{1}{2}\right)$
10	ID(A)	a	0011	$(0,\ 1,\ 0,\ 0)$
11	NOT(A)	$\neg a$	1100	$(0,\ -1,\ 0,\ 0)$
12	ID(B)	b	0101	$(0,\ 0,\ 1,\ 0)$
13	NOT(B)	$\neg b$	1010	$(0,\ 0,\ -1,\ 0)$
14	IMP $(a \to b)$	$\neg a \vee b$	1101	$\left(\frac{1}{2},\ -\frac{1}{2},\ \frac{1}{2},\ \frac{1}{2}\right)$
15	IMP $(b \to a)$	$\neg b \vee a$	1011	$\left(\frac{1}{2},\ \frac{1}{2},\ -\frac{1}{2},\ \frac{1}{2}\right)$

C Model Architectures

The architecture of the model used in Fig. 2 is described in Table 2. The `ResidualLogicBlock` performs logic-based convolutional processing as described in [21]. The only difference is the inclusion of a residual connection, which was found to increase training performance. Each `LogicDense` layer applies differentiable logic transformations as described in [20]. The final `GroupSum` layer aggregates logical activations into class scores for CIFAR-10. **Model parameters:** For the *Small* variant, $n_{\text{bits}} = 3$, $k_{\text{num}} = 32$, and $\tau = 20$.

Table 2: Architecture of the **CLGN CIFAR-10 Res** model.

Layer	Input dimension	Output dimension	Description
ResidualLogicBlock	$3n_{\text{bits}} \times 32 \times 32$	$2k_{\text{num}} \times 16 \times 16$	Depth $= 3$, receptive field 3×3, padding 1
Flatten	$2k_{\text{num}} \times 16 \times 16$	$256 \times 2k_{\text{num}}$	Spatial flattening of logic feature maps
LogicDense_1	$256 \times 2k_{\text{num}}$	$512k_{\text{num}}$	Fully connected logic layer
LogicDense_2	$512k_{\text{num}}$	$256k_{\text{num}}$	Fully connected logic layer
LogicDense_3	$256k_{\text{num}}$	$320k_{\text{num}}$	Fully connected logic layer
GroupSum	$320k_{\text{num}}$	10	Grouped logic aggregation with temperature τ