LILogic Net: Compact Logic Gate Networks with Learnable Connectivity for Efficient Hardware Deployment

Supplementary Material

6. Full Experimental Results on MNIST

Table 7 presents the complete set of results from our experiments on the MNIST dataset, expanding on the summary shown in the main paper.

We systematically varied the **layer width**, **network depth** (1 to 4 layers), and **logical connectivity strategy**, including:

- **F**: Fixed, non-learnable connectome,
- **Top-K**: Top-K sparse, learnable connectome with $K \in \{2, 4, 8, 16, 32, 64, 128\}$,
- **L**: Dense, fully learnable connectome.

For each architecture, we report:

- **Test accuracy** (mean $\pm$ std. deviation - in %),
- **Training time** (mean $\pm$ std. deviation - in minutes).

6.1. Experimental Setup

- Each configuration was run 10 times for 1- and 2-layer models, and 5 times for 3- and 4-layer networks.
- Training time was measured on a single NVIDIA A4000 GPU and includes model validation every 25 epochs (200 in total).
- Entries with "–" indicate configurations that were infeasible due to memory constraints.

6.2. Key Observations

Learned sparse Top-K connectivity consistently outperforms fixed random connectivity, especially at smaller model widths, with the performance gap narrowing as the width increases. At lower widths (e.g., 2K), the accuracy difference between Top-16 and fixed random (F) connectivity is more pronounced (over 2% points difference at 4 layers). As width increases to 32K, both methods achieve high accuracy, with Top-K maintaining a modest advantage (around 0.3–0.4% point). This indicates that while fixed random wiring can suffice at large scales, learned sparse wiring helps especially in smaller or medium-width models by optimizing logical connections more effectively.

Sparse Top-K connectivity improves accuracy across widths, with depth effects depending on K. Increasing the number of inputs K per gate generally improves accuracy across all widths (2K to 32K). For smaller K values (e.g., Top-2, Top-4), increasing the number of layers consistently enhances performance, reflecting the benefit of deeper feature integration. However, for larger K values (e.g., Top-64, Top-128), the best accuracy is often achieved at 2 or 3 layers, with 4-layer models showing little or no further improvement—or even slight degradation. This suggests that when gates have access to more inputs, very deep architectures may lead to overfitting or diminishing returns in this setting.

Top-K offers an efficient trade-off between accuracy and computational cost. Top-K connectivity sparsifies each gate's inputs, significantly reducing the number of active computations during both training and inference, while retaining high accuracy. For instance, at 8K width and 2 layers, a Top-16 model achieves **98.66%** accuracy — slightly exceeding fully learnable (L) models — while using fewer connections and benefiting from reduced computational complexity. This makes Top-K particularly attractive for efficient, scalable architectures in resource-constrained settings.

Comparison of fully learned (L) and Top-K connectivity within the same width and depth. At shallow depths (e.g., 1 layer), learned connectivity (L) consistently achieves the highest accuracy, outperforming both fixed random (F) and Top-K sparsity patterns. However, as the network depth increases, Top-K connectivity increasingly surpasses learned connectivity, demonstrating stronger scalability and better generalization in deeper models. Interestingly, for deeper architectures, smaller values of K in Top-K (e.g., Top-8, Top-16) tend to yield the best accuracy, suggesting that moderate sparse wiring is sufficient for effective information integration while controlling complexity and overfitting.

At fixed gate budgets, shallow and wide architectures often outperform deeper ones — until a certain scale is reached. As discussed in Section 4.3, under constrained gate budgets (2K–8K), shallow fully learnable (L) models achieve surprisingly strong performance. For instance, a 1-layer 4K-width model (1L–4K) reaches **97.96%** accuracy, outperforming deeper or sparse Top-K models of similar budget (e.g., 2Top128–2K with **97.66%**). This suggests that when capacity is limited, wider layers and dense training provide more effective use of resources than depth or sparsity. However, at higher budgets (e.g., 8K+), Top-K architectures begin to outperform learned ones. For example, 2Top64–8K reaches **98.69%**, exceeding 1L–16K (**98.58%**). Depth becomes beneficial beyond a threshold, but results show that 2-layer configurations often outperform their 4-layer counterparts (e.g., 2Top64–8K vs. 4Top16–4K), indicating diminishing returns from excessive depth once sparsity and width are well-balanced.

Training time scales non-uniformly with width and K, but grows roughly linearly with depth. Training time increases with model width, but the scaling is non-monotonic: from 2K to 8K gates, the time increase is modest, likely

Width	Type	1 Layer		2 Layers		3 Layers		4 Layers	
		Test Acc. [%]	Train Time [min]	Test Acc. [%]	Train Time [min]	Test Acc. [%]	Train Time [min]	Test Acc. [%]	Train Time [min]
2,000	F	87.09 ± 0.28	3.3 ± 0.3	90.82 ± 0.40	5.5 ± 0.6	93.69 ± 0.29	7.4 ± 0.4	95.38 ± 0.20	8.9 ± 0.7
	Top2	90.05 ± 0.18	4.3 ± 0.2	94.58 ± 0.21	1.9 ± 0.0	96.25 ± 0.12	2.5 ± 0.0	96.73 ± 0.12	3.1 ± 0.0
	Top4	92.41 ± 0.23	4.6 ± 0.3	96.39 ± 0.15	2.0 ± 0.0	97.19 ± 0.12	2.7 ± 0.1	97.38 ± 0.14	3.3 ± 0.1
	Top8	94.29 ± 0.12	4.4 ± 0.3	96.97 ± 0.12	2.0 ± 0.0	97.47 ± 0.16	2.7 ± 0.0	97.59 ± 0.24	3.4 ± 0.0
	Top16	95.63 ± 0.19	4.6 ± 0.3	97.35 ± 0.08	2.6 ± 0.0	97.72 ± 0.12	4.0 ± 0.0	**97.75 ± 0.12**	5.3 ± 0.0
	Top32	96.38 ± 0.15	4.6 ± 0.2	97.50 ± 0.13	4.8 ± 0.0	97.83 ± 0.15	7.8 ± 0.0	97.71 ± 0.12	10.8 ± 0.0
	Top64	96.86 ± 0.18	5.1 ± 0.3	97.65 ± 0.05	8.9 ± 0.0	**97.89 ± 0.06**	15.1 ± 0.0	97.45 ± 0.31	21.3 ± 0.0
	Top128	97.02 ± 0.12	5.6 ± 0.1	**97.66 ± 0.06**	17.6 ± 0.0	97.67 ± 0.17	23.3 ± 0.0	96.92 ± 0.14	37.5 ± 6.5
	L	**97.25 ± 0.13**	5.0 ± 0.4	97.37 ± 0.08	4.1 ± 0.0	96.13 ± 0.52	6.4 ± 0.0	92.00 ± 1.54	8.5 ± 0.1
4,000	F	90.16 ± 0.24	1.4 ± 0.6	93.28 ± 0.21	1.9 ± 0.7	95.77 ± 0.18	2.4 ± 1.0	96.90 ± 0.11	2.7 ± 0.0
	Top2	92.37 ± 0.14	1.5 ± 0.4	96.50 ± 0.15	2.0 ± 0.1	97.56 ± 0.13	2.6 ± 0.1	97.84 ± 0.15	3.2 ± 0.0
	Top4	94.66 ± 0.26	1.5 ± 0.0	97.61 ± 0.10	2.2 ± 0.1	97.99 ± 0.10	2.9 ± 0.0	98.04 ± 0.21	3.7 ± 0.0
	Top8	96.27 ± 0.11	1.5 ± 0.0	97.91 ± 0.09	2.8 ± 0.0	98.13 ± 0.07	4.2 ± 0.0	98.22 ± 0.11	5.7 ± 0.0
	Top16	97.17 ± 0.10	1.9 ± 0.0	98.14 ± 0.15	4.6 ± 0.1	98.33 ± 0.09	7.3 ± 0.0	**98.37 ± 0.09**	10.1 ± 0.0
	Top32	97.54 ± 0.16	3.1 ± 0.9	98.17 ± 0.16	8.3 ± 0.1	**98.38 ± 0.05**	13.8 ± 0.0	98.29 ± 0.12	19.4 ± 0.1
	Top64	97.76 ± 0.09	6.3 ± 0.0	98.25 ± 0.07	19.0 ± 0.0	98.31 ± 0.09	31.5 ± 0.1	98.06 ± 0.18	44.1 ± 0.0
	Top128	97.94 ± 0.07	8.0 ± 0.0	**98.28 ± 0.08**	33.1 ± 0.2	98.30 ± 0.07	58.8 ± 0.0	97.56 ± 0.16	83.6 ± 0.1
	L	**97.96 ± 0.11**	2.5 ± 0.0	98.11 ± 0.06	10.8 ± 0.0	97.02 ± 0.53	19.2 ± 0.1	94.41 ± 1.49	27.4 ± 0.3
8,000	F	91.48 ± 0.18	1.1 ± 0.0	94.83 ± 0.12	1.6 ± 0.0	96.83 ± 0.11	2.2 ± 0.0	97.69 ± 0.12	2.8 ± 0.0
	Top2	93.72 ± 0.09	1.2 ± 0.0	97.41 ± 0.07	2.7 ± 0.0	98.20 ± 0.11	4.0 ± 0.0	98.28 ± 0.14	5.3 ± 0.0
	Top4	95.70 ± 0.12	1.4 ± 0.0	98.24 ± 0.08	3.4 ± 0.0	98.50 ± 0.04	5.4 ± 0.0	98.56 ± 0.05	7.3 ± 0.0
	Top8	97.17 ± 0.07	1.8 ± 0.0	98.43 ± 0.07	5.7 ± 0.0	98.60 ± 0.06	9.4 ± 0.0	98.63 ± 0.09	13.1 ± 0.0
	Top16	97.85 ± 0.08	2.8 ± 0.0	98.66 ± 0.06	10.3 ± 0.0	98.69 ± 0.04	17.6 ± 0.1	**98.79 ± 0.07**	24.9 ± 0.1
	Top32	98.15 ± 0.02	6.3 ± 0.0	98.63 ± 0.08	22.8 ± 0.1	**98.77 ± 0.13**	39.2 ± 0.1	98.72 ± 0.05	55.5 ± 0.2
	Top64	98.36 ± 0.07	11.6 ± 0.0	**98.69 ± 0.06**	45.3 ± 0.2	98.74 ± 0.09	78.5 ± 0.3	98.44 ± 0.10	111.5 ± 0.4
	Top128	98.39 ± 0.07	16.2 ± 0.1	98.67 ± 0.06	82.4 ± 0.4	98.72 ± 0.11	148.1 ± 0.3	97.94 ± 0.28	213.8 ± 0.6
	L	**98.45 ± 0.07**	4.3 ± 0.1	98.62 ± 0.12	4.4 ± 0.1	97.87 ± 0.22	69.2 ± 0.2	96.81 ± 0.77	101.4 ± 0.2
16,000	F	93.16 ± 0.16	1.3 ± 0.0	96.43 ± 0.12	2.6 ± 0.0	97.82 ± 0.13	3.8 ± 0.0	98.34 ± 0.08	5.1 ± 0.0
	Top2	95.52 ± 0.16	1.8 ± 0.0	98.15 ± 0.09	6.0 ± 0.1	98.53 ± 0.03	10.0 ± 0.0	98.64 ± 0.09	14.0 ± 0.0
	Top4	97.24 ± 0.09	2.2 ± 0.0	98.59 ± 0.08	8.8 ± 0.0	98.68 ± 0.06	15.3 ± 0.0	98.77 ± 0.06	21.8 ± 0.1
	Top8	98.15 ± 0.04	3.1 ± 0.0	98.74 ± 0.10	14.7 ± 0.1	98.80 ± 0.08	25.9 ± 0.1	**98.87** ± 0.08	37.3 ± 0.4
	Top16	98.48 ± 0.10	6.9 ± 0.0	98.80 ± 0.05	30.2 ± 0.1	98.79 ± 0.08	53.4 ± 0.2	98.83 ± 0.10	76.7 ± 0.5
	Top32	98.50 ± 0.07	12.2 ± 0.0	**98.95 ± 0.09**	57.1 ± 0.2	**98.86 ± 0.07**	101.6 ± 0.3	98.73 ± 0.05	146.2 ± 0.5
	Top64	98.53 ± 0.11	22.5 ± 0.0	98.88 ± 0.06	111.9 ± 0.3	98.85 ± 0.10	200.7 ± 0.4	98.71 ± 0.04	289.1 ± 0.3
	Top128	98.55 ± 0.03	42.3 ± 0.0	98.89 ± 0.11	219.4 ± 0.3	98.84 ± 0.09	396.5 ± 0.3	98.23 ± 0.12	475.6 ± 0.5
	L	**98.58** ± 0.07	7.8 ± 0.0	98.82 ± 0.10	154.9 ± 0.1	–	–	–	–
32,000	F	94.74 ± 0.07	2.9 ± 0.0	97.41 ± 0.13	5.4 ± 0.0	98.27 ± 0.10	8.0 ± 0.1	98.52 ± 0.09	10.4 ± 0.2
	Top2	96.85 ± 0.06	3.1 ± 0.0	98.41 ± 0.12	15.8 ± 0.1	98.64 ± 0.11	28.7 ± 0.2	98.58 ± 0.06	41.0 ± 0.7
	Top4	98.05 ± 0.11	4.0 ± 0.0	98.67 ± 0.05	23.0 ± 0.1	98.81 ± 0.04	43.6 ± 0.3	98.79 ± 0.08	61.4 ± 0.6
	Top8	98.41 ± 0.11	7.7 ± 0.0	98.75 ± 0.07	40.5 ± 0.1	98.78 ± 0.01	69.1 ± 0.3	98.76 ± 0.06	105.5 ± 0.3
	Top16	98.56 ± 0.07	13.4 ± 0.1	98.83 ± 0.04	57.9 ± 1.0	98.83 ± 0.05	103.0 ± 0.5	98.81 ± 0.05	149.5 ± 0.5
	Top32	**98.58 ± 0.07**	24.2 ± 0.1	98.83 ± 0.09	134.3 ± 0.1	98.84 ± 0.10	240.7 ± 1.2	**98.89 ± 0.06**	348.4 ± 0.7
	Top64	98.55 ± 0.10	45.5 ± 0.1	98.87 ± 0.07	268.0 ± 0.8	**98.87 ± 0.05**	490.2 ± 6.6	97.04 ± 0.04	723.0 ± 1.2
	Top128	98.51 ± 0.08	84.0 ± 0.1	**98.91 ± 0.05**	524.1 ± 0.2	98.83 ± 0.04	884.5 ± 4.2	97.77 ± 0.20	1326.6 ± 5.1
	L	98.52 ± 0.09	15.0 ± 0.1	–	–	–	–	–	–

Table 7. Test accuracy and training time on MNIST for different logic gate networks, layer widths, and depths. Each cell shows the mean ± standard deviation over multiple runs (10 runs for 1-2 layers, 5 runs for 3-4 layers). Dashes indicate inapplicable configurations. Bolded values indicate the best performance per block.

due to better GPU utilization. However, for larger widths (16K–32K), the increase becomes steeper, suggesting that memory or computational bottlenecks begin to dominate. The relationship between training time and the number of gate inputs K is not strictly linear: while larger K generally increases cost, the overhead is influenced by implementation details, such as sparse gather operations and input aggregation. Top-128 models are much slower than Top-16, but the scaling trend is irregular. In contrast, training time grows approximately linearly with depth — adding layers increases computation in a predictable way. Notably, fully learnable connectivity (L) often trains faster than Top-K models despite involving weight optimization. This is likely due to GPU-accelerated dense matrix operations (`matmul`) being more efficient than sparse input selection used in Top-K. Several anomalies are observed: (i) For width 2K, 1-layer model, training times are longer than for 4K or 8K models. This may stem from poor GPU utilization, kernel launch overhead, or small matrix sizes being inefficiently handled. (ii) Fixed connectivity (F) also shows unexpectedly long training times at low widths (e.g., 2K), possibly due to inefficient memory access or unoptimized computation of fixed connections.

Low variance in results confirms robustness. Across all settings, standard deviation in test accuracy remains under 0.2%, indicating stable convergence and reproducibility, even for models with high structural sparsity.

7. Full Experimental Results on CIFAR-10

Table 8 presents the complete set of results from our experiments on the CIFAR-10 dataset, expanding on the summary shown in the main paper.

Building on insights from MNIST, we focused on 1-layer variants for smaller architectures (8K gates) and 1–2-layer variants for wider layers. We evaluated the following logical connectivity strategies:

- **F**: Fixed, non-learnable connectome,
- **Top-K**: Top-K sparse, learnable connectome with $K = 32$, chosen based on MNIST experiments as a good trade-off between accuracy and training time,
- **L**: Dense, fully learnable connectome.

We also experimented with two sets of fixed thresholds to binarize each RGB channel:

- $[0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875]$,
- $[0.2, 0.4, 0.6, 0.8]$,

resulting in input vector sizes of 21,504 and 12,288, respectively.

For each architecture, we report:

- τ **value**,
- **Test accuracy** (mean ± std. deviation in
- **Training time** (mean ± std. deviation in minutes).

7.1. Experimental Setup

- Each configuration was run 5 times, and results are averaged.
- Training time was measured on a single NVIDIA H200 GPU and includes model validation every 25 epochs (200 in total).
- The entire training dataset was loaded at the beginning instead of subsequent batch loading.
- Entries with "–" indicate configurations that were infeasible due to memory constraints.

7.2. Key Observations

Most of the observations reported for MNIST also apply to CIFAR-10. Therefore, here we focus only on additional aspects specific to this dataset.

Threshold count has limited effect. The difference between using 4 or 7 thresholds is small, though the 7-threshold configuration shows a slight advantage in most cases.

Training time is largest for L-type architectures. For MNIST, L-type layers often trained faster than Top-K models, likely because GPU-accelerated dense matrix operations (`matmul`) are more efficient than sparse input selection used in Top-K. In CIFAR-10, however, L-type layers have the longest training times due to the much larger input size, which increases the number of connections in the first layer. Additionally, training time differences between the 4- and 7-threshold modes appear only for L-type architectures for the same reason; Top-K and F architectures are unaffected, as their number of connections does not depend on input size.

Accuracy standard deviation is larger than for MNIST. Accuracy across 5 runs varies more than for MNIST, with standard deviations reaching up to 0.3%. This is largely due to binarizing separate RGB channels with multiple thresholds, which increases variability in the input representation. Although CIFAR-10 has the same number of classes and slightly larger images than MNIST, this multi-threshold binarization amplifies sensitivity to initialization, data order, and training dynamics. Noise, labeling inconsistencies, and the small number of runs also contribute, whereas MNIST's simple grayscale images with a single threshold yield highly stable results.

Best-performing architectures. Across all settings, 1-layer L-type variants achieved the highest accuracy. For the largest budget architectures (e.g., 256K gates), memory constraints prevented evaluating L-type networks; in these cases, the best results were obtained using 2-layer Top-32 networks.

#gates	Model	Layer Width	τ	7 thresholds		4 thresholds	
				Test Acc. [%]	Train Time [min]	Test Acc. [%]	Train Time [min]
8,000	1F	8,000	20	44.08 ± 0.12	2.5 ± 0.0	44.18 ± 0.27	1.3 ± 0.0
	1Top32	8,000	20	52.27 ± 0.20	2.6 ± 0.0	51.86 ± 0.26	2.5 ± 0.0
	1L	8,000	20	**55.11** ± 0.17	45.4 ± 0.1	54.51 ± 0.25	26.1 ± 0.0
64,000	1F	64,000	90	49.17 ± 0.14	2.9 ± 0.0	49.06 ± 0.25	2.0 ± 0.1
	1Top32	64,000	90	57.28 ± 0.30	15.3 ± 0.0	56.81 ± 0.19	16.9 ± 0.3
	1L	64,000	90	**57.66** ± 0.17	139.7 ± 0.1	57.64 ± 0.23	80.9 ± 0.1
	2Top32	32,000	70	57.12 ± 0.11	72.9 ± 0.0	56.40 ± 0.17	72.7 ± 0.1
	2L	32,000	70	55.75 ± 0.18	224.1 ± 0.1	55.58 ± 0.42	188.3 ± 0.4
128,000	1F	128,000	100	50.92 ± 0.24	3.8 ± 0.0	50.87 ± 0.30	3.7 ± 0.1
	1Top32	128,000	100	59.43 ± 0.22	29.3 ± 0.0	58.95 ± 0.14	30.7 ± 0.1
	1L	128,000	100	–	–	60.13 ± 0.14	157.3 ± 0.3
	2Top32	64,000	90	58.88 ± 0.26	146.7 ± 0.0	58.36 ± 0.22	146.5 ± 0.0
256,000	1F	256,000	110	52.48 ± 0.28	5.0 ± 0.0	52.05 ± 0.09	5.3 ± 0.0
	2F	128,000	100	54.76 ± 0.27	16.3 ± 0.0	54.18 ± 0.24	16.1 ± 0.0
	2Top32	128,000	100	**60.98** ± 0.19	297.2 ± 6.6	59.86 ± 0.23	298.7 ± 0.3

Table 8. Test accuracy and training time on CIFAR-10 for different logic gate networks, layer widths, depths and input binarization thresholds. Each cell shows the mean ± standard deviation over 5 runs. Dashes indicate inapplicable configurations. Bolded values indicate the chosen models (LILogic Net-S, -M, -L) with best performance.

8. Training Accuracy and Effect of Discretization

We analyze the influence of discretization on network performance by measuring the difference between the accuracy in inference mode and the accuracy during differentiable training. Early in training, this binarization effect can noticeably affect accuracy as the network starts by learning a probabilistic, differentiable LGN, with high uncertainty in the selection of individual logic gates. The discretization step—selecting the most probable gate for inference—can therefore produce substantial changes in network behavior during this phase.

As training progresses, the gate selection probabilities become more confident, and the effect of discretization on accuracy diminishes (e.g. Fig. 7). By the end of training, the difference between inference-mode and training-mode accuracy is negligible, with a final binarization error **below 0.1 percentage points**.

This pattern demonstrates that the network first explores a smooth, probabilistic logic representation and gradually converges to stable discrete logic choices, with minimal impact on final performance.

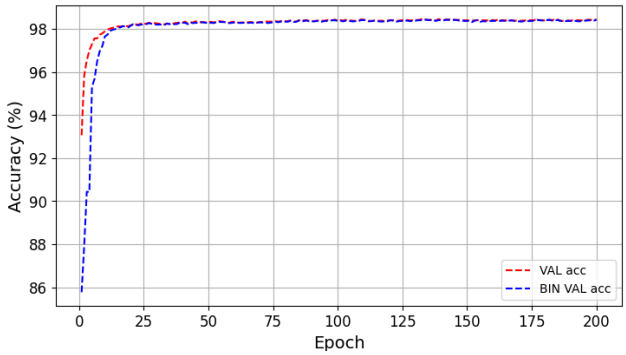

Figure 7. MNIST LILogic Net-S training and validation accuracy curves before (VAL acc) and after (BIN VAL acc) discretization. The discretization error is very small during late training.

9. LGN Model FPGA Resource Usage

Table 9 summarizes the accuracy and FPGA resource utilization of different LGN configurations across MNIST, FashionMNIST, and CIFAR-10 datasets, including LUT usage for generic and target devices.

Tables 9 and 10 summarize the hardware characteristics of various LGN configurations across MNIST, FashionMNIST, and CIFAR-10 datasets. Table 9 reports accuracy and FPGA resource utilization (LUT usage for generic and Xilinx devices), while Table 10 reports Xilinx frequency and logic/routing latency. Architectures marked with $*$ correspond to DiffLogicNet variants. Dashes (-) indicate missing or not applicable data.

Table 9. Accuracy of single runs and FPGA resources per dataset. Most of the achitectures are binarized LILogicNet-S,-M,-L. $*$ Architectures of DiffLogicNet-S and DiffLogiCNet reported for MNIST. Given two values of LUTs means total LUTS and LUTs without population count (popcount) part.

Dataset	Architecture	Acc [%]	Generic LUT4	ICE40 LUT4	Generic LUT6	Xilinx LUT6
MNIST	1x4000	97.63	11987 / 3637	11188	8760 / 3637	7103 / 1791
	1x8000	98.47	21677 / 7047	20696	16811 / 7047	14076 / 3491
	2x16000	98.94	47869 / 14940	46379	41258 / 14940	37373 / 11833
	4x8000	98.79	24676 / 8681	23697	20806 / 8065	16004 / 4637
FMNIST	1x8000	90.03	32436 / 7695	27163	-	14321 / 3736
	2x16000	90.03	51311 / 14891	50893	-	37075 / 11535
	2x32000	90.39	95342 / 29111	93926	-	73938 / 22308
	2x64000	90.66	176882 / 55302	172196	-	143226 / 39769
	6x8000*	88.56	39347 / 17562	39417	-	22710 / 10377
	6x64000*	90.27	264873 / 131992	264982	-	172787 / 72807
CIFAR10	1x8000	54.89	43237 / 7834	-	38225 / 7834	14415 / 3830
	1x64000	58.06	186010 / 57329	187468	-	104853 / 27070
	2x128000	60.76	387156 / 117164	-	339376 / 117164	293285 / 92353

Table 10. Xilinx XC7A200T frequency and latency per dataset and model architecture. Most of the achitectures are binarized LILogicNet-S,-M,-L. $*$ Architectures of DiffLogicNet-S and DiffLogicNet reported for MNIST. Dashes (-) indicate missing or not applicable data.

Dataset	LGN	Xilinx Freq	Logic Latency (ns)	Routing Latency (ns)
MNIST	1x4000	40.15 MHz	5.1	19.8
	1x8000	32.89 MHz	6.0	24.4
	2x16000	29.71 MHz	6.2	27.4
	4x8000	26.69 MHz	5.0	32.4
FMNIST	1x8000	33.76 MHz	5.9	23.8
	2x16000	20.13 MHz	6.1	43.6
	2x32000	18.16 MHz	-	-
	2x64000	-	-	-
	6x8000*	19.11 MHz	-	-
	6x64000*	-	-	-
CIFAR10	1x8000	31.18 MHz	5.8	26.2
	1x64000	-	-	-
	2x128000	-	-	-