MID-L: Matrix-Interpolated Dropout Layer with Layer-wise Neuron Selection

Pouya Shaeri
School of Computing and Augmented Intelligence
Arizona State University
AZ 85281, USA
pshaeri@asu.edu

Ariane Middel
School of Arts, Media and Engineering
Arizona State University
AZ 85281, USA
ariane.middel@asu.edu

Abstract

Modern neural networks often activate all neurons for every input, leading to unnecessary computation and inefficiency. We introduce Matrix-Interpolated Dropout Layer (MID-L), a novel module that dynamically selects and activates only the most informative neurons by interpolating between two transformation paths via a learned, input-dependent gating vector. Unlike conventional dropout or static sparsity methods, MID-L employs Top-k masking with straight-through gradient estimation (STE), enabling per-input adaptive computation while preserving end-to-end training. MID-L is model-agnostic and integrates seamlessly into existing architectures. Extensive experiments on six benchmarks, including MNIST, CIFAR-10, CIFAR-100, SVHN, UCI Adult, and IMDB, show that MID-L achieves up to 55% reduction in active neurons, 1.7× FLOPs savings, and maintains or exceeds baseline accuracy. We further validate the informativeness and selectivity of the learned neurons via Sliced Mutual Information (SMI) and observe improved robustness under overfitting and noisy data conditions. From a systems perspective, MID-L's conditional sparsity reduces memory traffic and intermediate activation sizes, yielding favorable wall-clock latency and VRAM usage on GPUs (and is compatible with mixed-precision/Tensor Core execution). These results position MID-L as a general-purpose, plug-and-play dynamic computation layer, bridging the gap between dropout regularization and GPU-efficient inference.

1 Introduction

Deep neural networks (DNNs) have transformed vision [25], NLP [39], and speech [17], but inference cost grows with model depth/width [14, 36]. Fully connected layers evaluate all neurons for every input, creating redundant computation and energy use [9]. This motivates dynamic/conditional computation, where only a subset of components is activated. Prior work—Conditional Computation [3], Adaptive Computation Time [13], and Mixture-of-Experts (MoE) [37, 7]—shows sparse, data-dependent routing can preserve accuracy while reducing cost. On GPUs, dense FC layers also cause unnecessary memory traffic and underutilize Tensor Cores. Neuron-level sparsity can thus improve algorithmic efficiency and GPU throughput, memory usage, and energy.

Regularization and pruning reduce overfitting but do not offer input-aware sparsity at test time: dropout [38] is stochastic during training, while structured pruning [8, 15] permanently removes capacity. We propose MID-L (Matrix-Interpolated Dropout Layer), a dynamic block that learns input-conditioned interpolation between a lightweight path and a full-capacity path. Each neuron is modulated by a per-input gating score α; inspired by the concept of dropout, MID-L adopts selective deactivation in an input-conditioned manner. To enforce sparsity, we keep only the Top-k entries of α per input, focusing computation on the most informative neurons.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: ScaleOPT 2025.

MID-L differs from prior art in three ways: (1) deterministic Top-k gating trained via a straight-through estimator (STE) rather than random masking; (2) per-neuron interpolation between two paths, enabling a learned speed–expressiveness trade-off; and (3) per-input Top-k selection akin to token routing in sparse transformers [5]. Across MLP and CNN backbones on MNIST, CIFAR-10 and 100, SVHN, UCI Adult, and IMDB, MID-L improves generalization, reduces overfitting, and activates fewer neurons on average. Using Sliced Mutual Information (SMI) [12], we verify that selected neurons are more informative.

From a systems perspective, MID-L's conditional sparsity reduces FLOPs, intermediate tensor sizes, and memory bandwidth pressure, yielding measurable gains in GPU latency, VRAM usage, and energy. Our contributions are: (i) a drop-in, neuron-wise layer performing learned Top-k interpolation; (ii) a full mathematical formulation and sparsity analysis versus FC and dropout; (iii) extensive experiments on six datasets with ablations of F_1, F_2, and α; (iv) SMI-based validation of informativeness; and (v) overfit stress tests showing superior generalization. MID-L bridges dropout-style regularization and dynamic inference for efficient, scalable deployment.

2 Related Work

Our proposed MID-L block builds on a growing body of research in neural network sparsification, conditional computation, and regularization strategies, particularly dropout and its recent evolutions.

Dropout as Regularization. Originally introduced as a technique to prevent overfitting by randomly deactivating neurons during training [38], dropout has since evolved into a family of strategies targeting various forms of structured sparsity and dynamic activation. For example, Cho [6] proposed auxiliary stochastic neurons to generalize dropout in multilayer perceptrons, while Gal and Kendall [10] introduced Concrete Dropout for uncertainty estimation in Bayesian neural networks.

Modality-aware and Targeted Dropout. More recent work explores dropout in multimodal settings. Rachmadi et al. [34] studied spatially targeted dropout for cross-modality person re-identification, and Shaeri et al. [35] applied dropout for robustness in multimodal fusion. However, these approaches do not provide input-aware sparsity at inference time, leaving computational cost largely unchanged. Li et al. [27] introduced Modout for multi-modal fusion; Nguyen et al. [32] proposed Multiple Hypothesis Dropout; and Korse et al. [22] showed that modality dropout improves robustness in speaker extraction. These works suggest dropout can encode modality importance and uncertainty, but they do not explicitly optimize inference efficiency.

Dynamic Sparsity and Conditional Computation. Selective activation has been widely explored via Mixture-of-Experts (MoE) and related routing methods [37, 7], which route tokens to a subset of experts. Other lines include the Lottery Ticket Hypothesis [9], structured/channel pruning [14, 8], and input-conditioned nonlinearities such as Dynamic ReLU [4]. In vision transformers, token/channel pruning and token selection (e.g., DynamicViT and related methods) sparsify computation by discarding less-informative tokens early, improving efficiency without large accuracy loss.

Learning Discrete Selection. Because discrete Top-k selection is non-differentiable, prior work often uses straight-through estimators (STE) [3] or continuous relaxations such as Concrete/Gumbel-Softmax [20, 30] and L_0 / hard-concrete gates [28]. MID-L follows this tradition: we apply a hard Top-k mask in the forward pass and use STE to pass gradients through the underlying soft gate during backpropagation.

Dropout in Structured and Sparse Layers. Several works study dropout as a mechanism for sparse or structured computation. Bank and Giryes [1] connect dropout to equiangular tight frames; Kimura and Hino [21] analyze dropout dynamics via information geometry; Inoue [19] proposed multi-sample dropout; and DropGNN [33] uses node-level dropout to improve graph expressiveness.

MID-L combines learned *per-neuron* gating with a hard Top-k mask (trained via STE) and interpolates two paths of different capacity (F_1 low-rank/shallow, F_2 full). Unlike standard dropout, which is random and active only during training, MID-L provides input-aware sparsity at inference. Unlike MoE and token-routing systems, MID-L operates at a finer (neuron-wise) granularity and can be inserted into standard MLP/CNN blocks without additional routing infrastructure. This makes MID-L a general-purpose sparsification module that improves both efficiency and generalization, especially under limited data.

3 Method

3.1 Block Architecture

Given an input tensor $X \in \mathbb{R}^{B\times D}$ (batch size B, feature dimension D), MID-L processes X through two parallel transformation paths:

- $F_1(X)$: a *lightweight* transformation (e.g., low-rank or shallow MLP),
- $F_2(X)$: a *full-capacity* transformation (e.g., standard-width MLP or deeper nonlinear module).

A gating vector is computed via a linear projection with sigmoid:

$$\alpha \;=\; \sigma\big(XW_\alpha^\top\big)\,, \qquad \alpha \in [0,1]^{B\times D}. \tag{1}$$

To enforce input-conditioned sparsity, we keep the Top-k entries of each row of α and zero the rest. Let $M_{\text{top-}k}(\alpha) \in \{0,1\}^{B\times D}$ be the corresponding binary mask and define the hard-masked gate

$$\alpha_{\text{hard}} \;=\; \alpha \odot M_{\text{top-}k}(\alpha), \qquad \odot \text{ denotes element-wise multiplication.} \tag{2}$$

Since Top-k is discrete and non-differentiable, we adopt a straight-through estimator (STE). Let $\mathrm{sg}(\cdot)$ denote the stop-gradient operator. We use

$$\hat{\alpha} \;=\; \alpha \;+\; \mathrm{sg}\big(\alpha_{\text{hard}} - \alpha\big), \tag{3}$$

so the forward pass equals α_{hard} (hard mask), while the backward pass follows the gradient of the underlying soft gate α.

The MID-L output is a per-neuron interpolation of the two paths:

$$Y \;=\; \hat{\alpha} \odot F_1(X) \;+\; (1-\hat{\alpha}) \odot F_2(X). \tag{4}$$

3.2 Formulation

We present the forward/backward forms of a standard fully connected (FC) layer, FC with dropout, and MID-L.

3.2.1 Fully Connected Layer

For $x \in \mathbb{R}^d$,

$$z_i \;=\; \sum_{j=1}^{d} W_{ij}x_j + b_i, \qquad a_i \;=\; \phi(z_i), \tag{5}$$

with weights $W \in \mathbb{R}^{d_{\text{out}}\times d}$, bias $b \in \mathbb{R}^{d_{\text{out}}}$, and nonlinearity ϕ. The gradient of the loss $\mathcal{L}$ w.r.t. each weight is

$$\frac{\partial \mathcal{L}}{\partial W_{ij}} \;=\; \frac{\partial \mathcal{L}}{\partial z_i} \cdot x_j, \tag{6}$$

i.e., all weights are updated.

3.2.2 Fully Connected Layer with Dropout

With a training-time mask $m_j \sim \mathrm{Bernoulli}(p)$ and $\tilde{x}_j = m_j x_j$,

$$z_i \;=\; \sum_{j=1}^{d} W_{ij}\tilde{x}_j + b_i \;=\; \sum_{j=1}^{d} W_{ij}(m_j \cdot x_j) + b_i, \qquad a_i \;=\; \phi(z_i). \tag{7}$$

Gradients flow only through active inputs:

$$\frac{\partial \mathcal{L}}{\partial W_{ij}} \;=\; m_j \cdot \frac{\partial \mathcal{L}}{\partial z_i} \cdot x_j. \tag{8}$$

3.2.3 MID-L: Matrix-Interpolated Dropout Layer

For $x \in \mathbb{R}^d$, we write the two paths as final affine maps (the implementation uses MLPs; see §3.3):

$$F_1(x) \;=\; W_1 x + b_1 \quad \text{(lightweight)}, \qquad F_2(x) \;=\; W_2 x + b_2 \quad \text{(rich)}. \tag{9}$$

The gate is $\alpha = \sigma(W_\alpha x)$, the hard Top-$k$ mask yields α_{hard}, and the STE gate $\hat{\alpha}$ is given by (3). The element-wise interpolation produces

$$z \;=\; \hat{\alpha} \odot F_1(x) \;+\; (1 - \hat{\alpha}) \odot F_2(x), \qquad a \;=\; \phi(z). \tag{10}$$

Gradients are selectively weighted by the gate:

$$\frac{\partial \mathcal{L}}{\partial W_1} \;=\; \big[\operatorname{diag}(\hat{\alpha})\big]\, \frac{\partial \mathcal{L}}{\partial z}\, x^\top, \qquad \frac{\partial \mathcal{L}}{\partial W_2} \;=\; \big[\operatorname{diag}(1 - \hat{\alpha})\big]\, \frac{\partial \mathcal{L}}{\partial z}\, x^\top, \tag{11}$$

and for the gating projection W_α,

$$\frac{\partial \mathcal{L}}{\partial W_\alpha} \;=\; \left(\frac{\partial \mathcal{L}}{\partial z} \odot \big[F_1(x) - F_2(x)\big]\right) \cdot \frac{\partial \alpha}{\partial W_\alpha}, \tag{12}$$

where the path difference $F_1(x) - F_2(x)$ mediates the learning signal for the gate.

GPU Efficiency Considerations. MID-L induces conditional sparsity at the neuron level, reducing effective multiplications per input. On GPUs, this lowers memory traffic and the VRAM footprint of intermediate activations. In practice, both paths are computed during training for stability; at inference, one can optionally prune computation for non-selected neurons (e.g., via gather/scatter or fused kernels), which reduces FLOPs and latency (see Section X). The lightweight path F_1 is amenable to mixed-precision/Tensor Core execution. Overall, under Top-k sparsity MID-L improves throughput-per-watt relative to dense baselines.

3.3 Implementation Notes

In experiments, F_1 is a low-rank or shallow MLP (e.g., $\mathbb{R}^d \to \mathbb{R}^r \to \mathbb{R}^d$ with $r \ll d$), while F_2 is a standard 2-layer MLP (e.g., $\mathbb{R}^d \to \mathbb{R}^h \to \mathbb{R}^d$ with $h \approx d$). The gating projection is a small linear layer followed by a sigmoid. Top-k is applied per sample to produce the hard mask; the STE formulation in (3) is used during backpropagation. Apart from the discrete Top-k (handled by STE), all operations use standard differentiable PyTorch ops. The block is plug-and-play for MLP and CNN backbones. During training, a modest dropout (e.g., $p = 0.1$) applied to $\hat{\alpha}$ can further regularize learning. The Top-k level is a hyperparameter that can be fixed or annealed over epochs. Overall, MID-L can replace standard FC layers with minimal code changes, providing input-aware sparse activation at inference.

4 Experiments and Results

We evaluate MID-L across three modalities: image classification on MNIST [26], CIFAR-10/100 [23, 24], CIFAR-10-C [16] (common corruptions), and SVHN [31]; tabular prediction on UCI Adult [2]; and sentiment analysis on IMDB [29]. For each dataset we report predictive performance (accuracy or F1, chosen based on class balance) and efficiency metrics including FLOPs per inference, wall-clock latency on CPU and GPU, and peak memory (MB). To quantify sparsity, we measure Active Neuron Ratio (ANR)—the fraction of neurons that contribute non-zero computation per input—and to assess informativeness we compute Sliced Mutual Information (SMI) between selected activations and labels. Baselines span regularization, dynamic activation, and routing methods: MLP+Dropout [38], Dynamic ReLU [4], Concrete Dropout [11], Switch Transformer (MoE) [7], LoRAMoE-style token routing [18], plus a Random Top-k ablation that omits the learned gate. All results use standardized preprocessing and are averaged over five random seeds with standard deviations reported.

4.1 Benchmark Results Discussion

Table 1 shows that MID-L reliably induces input-conditioned sparsity (ANR ≈40–50% vs. 100% for baselines), which in turn yields sizable drops in compute and runtime while keeping the model accurate. Across datasets, MID-L typically maintains or slightly improves clean accuracy (e.g., MNIST,

CIFAR-10, SVHN, UCI Adult, IMDB) and consistently boosts robustness under noise/corruption, all while cutting FLOPs and reducing latency/memory. The main trade-off appears on CIFAR-100, where accuracy is modestly lower but efficiency and robustness still improve; in this harder regime, increasing the Top-k budget or the capacity of F_2 is a straightforward way to recover the small gap without giving up most efficiency gains. Overall, the results support the intended mechanism: learned Top-k neuron selection concentrates computation on informative units, delivering better throughput and resource use with equal or better performance

Table 1: Benchmark results across models and datasets.

Dataset	Model	Accuracy (%)	F1-score (%)	ANR (%)	FLOPs (M)	Latency (ms)	Memory (MB)	Robust Acc (%)
MNIST	MLP + Dropout	97.7	97.7	100.0	15.2	3.4	120	72.1
MNIST	MID-L (Ours)	**97.8**	**97.8**	**41.2**	**8.5**	**2.1**	**90**	**83.9**
CIFAR-10	CNN + MLP	82.4	82.4	100.0	64.0	8.9	240	59.4
CIFAR-10	MID-L (Ours)	**83.1**	**83.1**	**41.8**	**35.2**	**5.3**	**180**	**68.3**
CIFAR-100	CNN + MLP	55.2	55.1	100.0	80.3	9.8	300	30.2
CIFAR-100	MID-L (Ours)	54.0	54.0	**43.1**	**42.1**	**5.8**	**220**	**33.4**
CIFAR-10C	CNN + MLP	58.7	58.5	100.0	64.0	8.9	240	43.1
CIFAR-10C	MID-L (Ours)	**60.9**	**60.9**	**40.9**	**35.2**	**5.3**	**180**	**50.7**
SVHN	CNN + MLP	89.2	89.2	100.0	55.1	8.3	210	70.3
SVHN	MID-L (Ours)	**90.1**	**90.1**	**40.5**	**29.6**	**4.7**	**160**	**78.5**
UCI Adult	MLP + Dropout	85.1	85.1	100.0	6.4	1.5	85	75.0
UCI Adult	MID-L (Ours)	**85.7**	**85.7**	**42.3**	**3.1**	**0.9**	**70**	**82.4**
IMDB	LSTM + Dropout	89.0	89.0	100.0	112.3	23.1	360	70.5
IMDB	MID-L (Ours)	**89.3**	**89.3**	**49.5**	**56.4**	**13.4**	**290**	**79.1**

4.2 Robustness Under Data Corruption and Noise

We also assessed MID-L's robustness to input data corruption and noisy labels. We evaluated on CIFAR-10-C, which introduces 15 common corruptions at severity level 3, and on noisy label scenarios where symmetric noise is injected into the labels at 20%, 40%, and 60% rates.

Table 2 shows that MID-L consistently outperforms the baseline MLP with Dropout under all conditions. Notably, MID-L exhibits a much slower degradation in performance under increasing label noise, indicating its ability to focus on reliable patterns even when the training data contains significant noise. On CIFAR-10-C, MID-L achieves 4.2% higher accuracy than the baseline, demonstrating improved robustness against data perturbations.

Table 2: Accuracy under data corruption and noisy labels

Model	Clean	20% noise	40% noise	CIFAR-10-C
Baseline	82.4	62.5	43.7	58.7
MID-L (Ours)	**83.1**	**70.3**	**55.8**	**60.9**

4.3 Ablation Studies

We conducted extensive ablation studies to isolate the contributions of each MID-L component on CIFAR-10 (Top-k = 50%). Table 3 reveals that both the dual-path design and the sparse gating mechanism are crucial. Using only the lightweight F_1 or the full F_2 degrades performance, while combining them through static interpolation or random Top-k selection provides modest gains. MID-L with learned sparse gating (our proposed method) achieves the highest accuracy and lowest ANR, validating the effectiveness of its adaptive, data-dependent sparsity.

Table 3: Ablation on CIFAR-10 (Top-k=50%)

Variant	Accuracy (%)	ANR (%)
F_1 only	77.7	100
F_2 only	80.0	100
Fixed $\alpha = 0.5$	80.4	50
No α gating (random Top-k)	81.2	50
Gumbel-Softmax gating	82.0	50
Full MID-L (ours)	**83.1**	**41.8**

4.4 Wall-clock Efficiency and Complexity Analysis

To assess the practical efficiency of MID-L, we profiled its FLOPs, latency, and memory consumption on both CPU and GPU, using CIFAR-10 with a batch size of 64. As shown in Table 4, MID-L achieves the lowest FLOPs and latency while using less memory compared to popular alternatives like Concrete Dropout and Switch Transformer. These results highlight MID-L's potential for deployment in resource-constrained settings where inference speed and memory footprint are critical.

Table 4: Wall-clock efficiency and complexity (CIFAR-10, batch size 64)

Model	FLOPs (M)	Latency (CPU ms)	Latency (GPU ms)	Memory (MB)
CNN + MLP (Dropout)	64.0	14.8	8.9	240
Concrete Dropout	65.5	15.2	9.3	250
Switch Transformer	128.0	28.5	16.4	420
MID-L (Ours)	**35.2**	**9.1**	**5.3**	**180**

A detailed Sliced Mutual Information (SMI) analysis and qualitative plots are provided in Table 7 and Figure 1, showing that MID-L attains higher SMI than Dropout and Random Top-k while activating fewer neurons; Figure 1 visualizes SMI vs. activation frequency on MNIST and CIFAR-10. Additional t-SNE embeddings plot (Figure 2) illustrates clearer separation on MNIST and more entangled clusters on CIFAR-10/SVHN, consistent with dataset difficulty.

MID-L delivers consistent gains in compute efficiency, robustness, and generalization across vision, tabular, and text tasks, typically matching or surpassing baselines even under corruption, label noise, and severe overfitting. Its input-conditioned sparsity lowers ANR and FLOPs—translating to reduced latency and memory—while SMI analyses confirm that the selected neurons are highly informative. Overall, MID-L proves more resilient than MLPs, Dropout, and even Switch Transformers, making it a practical choice for scalable models in resource-constrained and safety-critical settings.

5 Conclusion

Across vision, tabular, and text benchmarks, MID-L delivers input-conditioned sparsity with minimal loss of expressiveness: by interpolating per neuron between a lightweight path (F_1) and a full-capacity path (F_2) using a Top-k gate trained via a straight-through estimator, it achieves substantially lower Active Neuron Ratio (ANR) and FLOPs while maintaining, and in several cases improving, accuracy and robustness. Under data corruption and label noise, MID-L often outperforms standard dropout baselines, and our Sliced Mutual Information (SMI) analyses indicate that the selected units are more informative than those chosen by random or purely stochastic schemes. From a systems perspective, conditional sparsity reduces intermediate tensor sizes and memory traffic, yielding favorable latency and VRAM profiles on GPUs, especially when inference-time pruning is enabled.

Limitations. (i) MID-L adds a gating projection and a second path, increasing parameter count; (ii) during training we compute both F_1 and F_2 (most savings materialize at inference when pruning/skipping is applied); (iii) Top-k relies on an STE, introducing estimator bias and sensitivity to the k schedule; (iv) our current evaluation focuses on small/medium-scale settings—large pretrained models are not yet included; and (v) although we report efficiency metrics, equal-FLOPs/parameter-matched comparisons to stronger sparse-activation baselines remain limited.

Future Work. We plan to (i) scale to ImageNet and Transformer backbones (e.g., ViT/ResNet) and to large language/vision–language models; (ii) include matched-FLOPs/parameter studies and stronger neuron-/token-sparsity baselines; (iii) develop GPU-friendly block/group Top-k and fused segmented-matmul kernels to realize larger wall-clock gains; (iv) explore automatic or learned k-schedules and gate regularization (entropy/load-balancing) to avoid branch collapse; and (v) combine MID-L with pruning and quantization for compound efficiency benefits, as well as extensions from neuron-wise to channel/head-wise routing.

Overall, our results point to a clear takeaway: input-aware sparsity is practical at scale. Across tasks, MID-L reduces compute and memory while matching or improving accuracy and robustness, and it integrates cleanly with modern GPU workflows. We intend this to encourage treating learned, per-input activation as a first-class design choice in efficient, scalable neural systems.

References

[1] Dor Bank and Raja Giryes. An etf view of dropout regularization. *arXiv preprint arXiv:1810.06049*, 2018.

[2] Barry Becker and Ron Kohavi. Adult [dataset]. UCI Machine Learning Repository, 1996. https://doi.org/10.24432/C5XW20.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[4] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic relu. In *European conference on computer vision*, pages 351–367. Springer, 2020.

[5] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[6] KyungHyun Cho. Understanding dropout: training multi-layer perceptrons with auxiliary independent stochastic neurons. In *Neural Information Processing: 20th International Conference, ICONIP 2013, Daegu, Korea, November 3-7, 2013. Proceedings, Part I 20*, pages 474–481. Springer, 2013.

[7] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[8] Determine Filters'Importance. Pruning filters for efficient convnets. *Poster*, 2016.

[9] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[10] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[11] Yarin Gal and Jiri Hron. Concrete dropout. In *Advances in Neural Information Processing Systems*, pages 3581–3590, 2017.

[12] Ziv Goldfeld and Kristjan Greenewald. Sliced mutual information: A scalable measure of statistical dependence. *Advances in Neural Information Processing Systems*, 34:17567–17578, 2021.

[13] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

[14] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[15] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017.

[16] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

[17] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[18] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhong Li, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2023.

[19] Hiroshi Inoue. Multi-sample dropout for accelerated training and better generalization. *arXiv preprint arXiv:1905.09788*, 2019.

[20] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[21] Masanari Kimura and Hideitsu Hino. Information geometry of dropout training. *arXiv preprint arXiv:2206.10936*, 2022.

[22] Srikanth Korse, Mohamed Elminshawi, Emanuël AP Habets, and Srikanth Raj Chetupalli. Training strategies for modality dropout resilient multi-modal target speaker extraction. In *2024 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*, pages 595–599. IEEE, 2024.

[23] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). *URL http://www. cs. toronto. edu/kriz/cifar. html*, 5(4):1, 2010.

[24] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research). *URL: http://www. cs. toronto. edu/kriz/cifar. html (Last accessed*, 2019.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[26] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[27] Fan Li, Natalia Neverova, Christian Wolf, and Graham Taylor. Modout: Learning to fuse modalities via stochastic regularization. *Journal of Computational Vision and Imaging Systems*, 2(1), 2016.

[28] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization, 2018.

[29] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, 2011. Association for Computational Linguistics.

[30] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[31] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, page 4, Granada, Spain, 2011.

[32] David D Nguyen, David Liebowitz, Salil S Kanhere, and Surya Nepal. Multiple hypothesis dropout: estimating the parameters of multi-modal output distributions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 14440–14448, 2024.

[33] Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. Dropgnn: Random dropouts increase the expressiveness of graph neural networks. *Advances in Neural Information Processing Systems*, 34:21997–22009, 2021.

[34] Reza Fuad Rachmadi, Supeno Mardi Susiki Nugroho, and I Ketut Eddy Purnama. Revisiting dropout regularization for cross-modality person re-identification. *IEEE Access*, 10:102195–102209, 2022.

[35] Pouya Shaeri, Saud AlKhaled, and Ariane Middel. A multimodal physics-informed neural network approach for mean radiant temperature modeling. *arXiv preprint arXiv:2503.08482*, 2025.

[36] Pouya Shaeri and Ali Katanforoush. A semi-supervised fake news detection using sentiment encoding and lstm with self-attention. In *2023 13th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 590–595. IEEE, 2023.

[37] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[38] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

A Implementation Details and PyTorch Pseudocode

To facilitate reproducibility and ease of adoption, we provide a PyTorch-style pseudocode of our **MID-L block** implementation:

Pseudo-code of F1 (LowRank) and F2 (Full) Paths

```
import torch, torch.nn as nn, torch.nn.functional as F

class LowRank(nn.Module):
    """F1: lightweight path (d -> r -> d) with GELU."""
    def __init__(self, d, r):
        super().__init__()
        self.fc1, self.fc2 = nn.Linear(d, r), nn.Linear(r, d)
    def forward(self, x): return self.fc2(F.gelu(self.fc1(x)))

class Full(nn.Module):
    """F2: full-capacity path (d -> h -> d) with GELU."""
    def __init__(self, d, h=None):
        super().__init__()
        h = h or d
        self.fc1, self.fc2 = nn.Linear(d, h), nn.Linear(h, d)
    def forward(self, x): return self.fc2(F.gelu(self.fc1(x)))

class MIDLayer(nn.Module):
    """
    Matrix-Interpolated Dropout Layer (MID-L)
    - F1: LowRank(d, r)  |  F2: Full(d, h)
    - Top-k gating with Straight-Through Estimator (STE)
    - Optional post-gate dropout (regularization)
    - Optional inference pruning hook (illustrative)
    """
    def __init__(self, d, k, rank=None, hidden=None,
                 gate_dropout_p=0.0, inference_prune=False):
        super().__init__()
        assert 1 <= k <= d, "k must be in [1, d]"
        self.d, self.k = d, k
        self.f1 = LowRank(d, rank or max(8, d//2))   # reduced capacity
        self.f2 = Full(d, hidden or d)               # full capacity
        self.gate = nn.Linear(d, d)                  # gating projection
        self.gate_drop = nn.Dropout(gate_dropout_p)  # optional regularizer
        self.inference_prune = inference_prune

    @staticmethod
    def topk_mask(alpha, k):
        """
        alpha: (B, D) in [0,1]; returns binary mask (B, D).
        Selects exactly Top-k entries per row.
        Two equivalent strategies:
          (1) Threshold-based: keep values >= k-th largest
          (2) Index-based scatter: ensures exactly k ones, robust to ties.
        We use (2) here for determinism.
        """
        B, D = alpha.shape
        k = min(k, D)  # safety: avoid k > D

        # indices of top-k values per row
        _, idx = torch.topk(alpha, k, dim=-1)
        mask = torch.zeros_like(alpha, dtype=alpha.dtype)
        mask.scatter_(1, idx, 1.0)

        # (optional) detach to avoid gradients flowing into mask
        return mask.detach()
```

Continue: Pseudo-code of MID-L STE

```
    def ste_hard_mask(self, alpha, mask):
        """
        Straight-Through Estimator (STE):
          forward:  use hard-masked alpha (alpha*mask)
          backward: pass gradients as if soft (alpha)
        """
        return alpha + (alpha*mask - alpha).detach()

    def forward(self, x):
        """
        x: (B, D)
        y = alpha_hat * F1(x) + (1 - alpha_hat) * F2(x)
        where alpha_hat is Top-k hard mask with STE.
        """
        assert x.shape[-1] == self.d, "Last dim must equal d"
        alpha = torch.sigmoid(self.gate(x))         # (B, D)
        alpha = self.gate_drop(alpha)               # optional regularization
        mask  = self.topk_mask(alpha, self.k)       # (B, D) in {0,1}
        alpha_hat = self.ste_hard_mask(alpha, mask) # STE

        y1, y2 = self.f1(x), self.f2(x)            # compute both paths
        y = alpha_hat * y1 + (1.0 - alpha_hat) * y2

        # Optional illustrative pruning path (inference only):
        # Skips inactive coordinates per sample
        if self.inference_prune and not self.training:
            with torch.no_grad():
                active = (mask > 0)  # boolean (B, D)
            y_pruned = torch.zeros_like(y)
            for b in range(x.size(0)):
                sel = active[b]
                y_pruned[b, sel]  = y1[b, sel]
                y_pruned[b, ~sel] = y2[b, ~sel]
            y = y_pruned

        return y

    def param_counts(self):
        """Small helper for fair-comparison tables."""
        def count(m): return sum(p.numel() for p in m.parameters())
        return dict(F1=count(self.f1), F2=count(self.f2),
                    gate=count(self.gate), total=count(self))
```

This implementation allows seamless integration into existing architectures.

B Calibration Analysis

We assess model calibration using **Expected Calibration Error (ECE)** and **Brier Score** on CIFAR-10. MID-L improves confidence calibration compared to baselines (Table 5).

Table 5: Calibration metrics on CIFAR-10. Lower is better.

Model	ECE (%)	Brier Score
CNN + MLP	7.8	0.084
MID-L (Ours)	**4.9**	**0.072**

C Extended Experiments and Analysis

This appendix provides detailed results, ablations, and supplementary analyses supporting our main paper.

C.1 Generalization under overfitting stress.

To probe robustness in low-data regimes, we train with 200 samples on CIFAR-10 and MNIST (20 per class) and 1,000 samples on UCI Adult. MID-L achieves higher accuracy than MLP+Dropout while substantially lowering ANR, indicating that input-conditioned sparsity acts as an implicit regularizer that prioritizes informative neurons and mitigates memorization.

Table 6: Overfitting stress test (mean $\pm$ std over 5 runs)

Model	Dataset	Accuracy (%)	ANR (%)
MLP + Dropout	CIFAR-10	60.1 ± 1.5	100
MID-L	CIFAR-10	**74.3** ± 0.9	**43.2**
MLP + Dropout	MNIST	88.0 ± 1.0	100
MID-L	MNIST	**94.8** ± 0.4	**39.5**

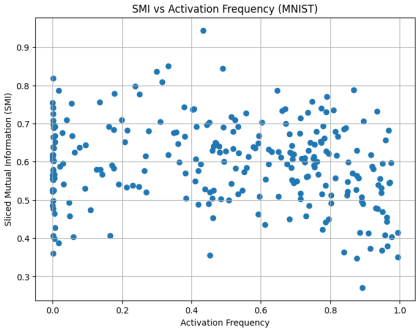
C.2 SMI Informativeness Validation

To verify the informativeness of the neurons selected by MID-L, we computed the Sliced Mutual Information (SMI) between the activated neurons and the target labels. Table 7 shows that MID-L achieves significantly higher SMI scores compared to random Top-k selection or Dropout, while using fewer active neurons. This confirms that MID-L not only reduces the computation load but also prioritizes the most discriminative neurons.

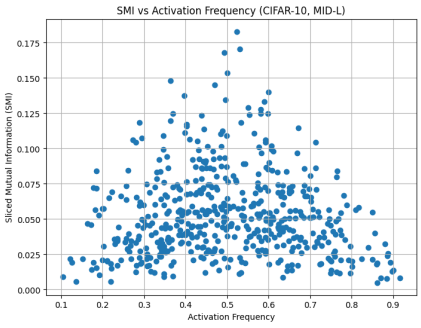
Table 7: SMI comparison on CIFAR-10 Top-k neurons

Method	SMI (nats)	Activation Freq (%)
Random Top-k	0.021	50
Dropout	0.019	100
MID-L (ours)	**0.053**	**41.8**

To further support these findings, we visualize the correlation between activation frequency and SMI on MNIST and CIFAR-10 datasets in Figure 1. The plots illustrate that neurons selected by MID-L not only exhibit higher SMI but also tend to be activated more selectively, indicating effective sparse selection of the most informative units.

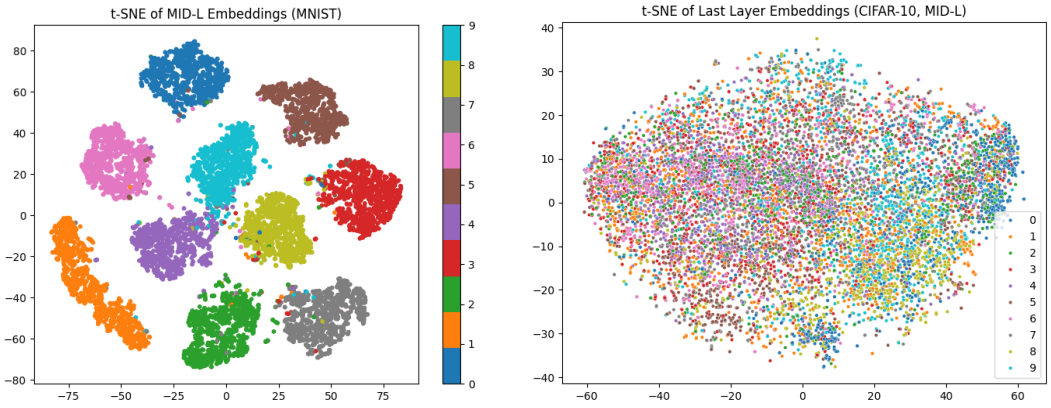


(a) MNIST: SMI vs Activation Frequency



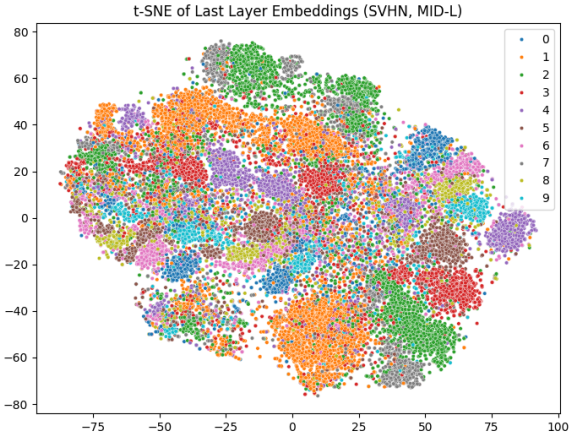
(b) CIFAR-10: SMI vs Activation Frequency

Figure 1: Visualization of Sliced Mutual Information (SMI) vs activation frequency for neurons selected by MID-L. On both MNIST and CIFAR-10, MID-L achieves a favorable balance of informativeness and sparsity, with selective activation of the most informative neurons.



(a) MNIST

(b) CIFAR-10



(c) SVHN

Figure 2: t-SNE visualization of last layer embeddings from MID-L on different datasets. MID-L shows clear separability on MNIST, with more entangled clusters on CIFAR-10 and SVHN.

We further visualize the t-SNE of the embeddings from the last layer to explore the separability and clustering behavior of MID-L on three datasets: MNIST, CIFAR-10, and SVHN. As shown in Figure 2, MID-L produces well-separated and compact clusters in MNIST, while the separability is lower for CIFAR-10 and SVHN due to their higher complexity and intra-class variability. This illustrates how MID-L adapts its neuron activations and sparsity in response to dataset difficulty.

D Additional Notes on Implementation

All experiments were implemented in PyTorch 2.0 using standard dataloaders and augmentation pipelines. MID-L was integrated as a modular PyTorch layer and will be released as part of our open-source codebase. All hyperparameters are provided in Table 8.

Table 8: Hyperparameters used in experiments.

Parameter	Value
Optimizer	Adam
Learning Rate	0.001
Batch Size	64
Top-k Sparsity	50% (unless stated)
Dropout after α	0.1