# Memory-based Self-Explainable Early Exit Networks

**Evan Kaiden**
University of California, Santa Cruz

**Leilani H. Gilpin**
University of California, Santa Cruz

**Biagio La Rosa**
University of California, Santa Cruz
bilarosa@ucsc.edu

## Abstract

Early Exit Neural Networks are networks that accelerate inference by allowing samples to exit through internal classifiers. One of the understudied areas related to these networks is whether early exits can provide interpretability alongside the prediction. In this work, we propose a memory-based self-interpretable design to make the internal classifiers more interpretable. Our design uses a sparse content-based attention mechanism to select relevant examples from a memory set and uses them to aid and explain the decision process. We show that, compared to alternative self-explainable designs, our design better preserves the performance of black-box baselines while providing means to inspect its inner mechanisms.

## 1 Introduction

Advances in deep neural networks have enabled their widespread use across many domains. This progression comes from larger models and entails significant computational overhead, which limits their applicability in compute-constrained settings, such as edge devices. In order to balance computational overhead and model performance, Early Exit Neural Networks (EENN) provide an approach for accelerating inference by allowing samples to exit early through *internal classifiers* (ICs). Despite the recent progress in performance for these methods [10, 7, 8, 35], an understudied question is whether these early exits can provide interpretability alongside the predictions.

This question has been extensively studied in the field of eXplainable AI (XAI), where several Self-Explainable (SENN) methods have been proposed for standard classification settings [2, 12, 16]. SENNs incorporate explainable elements into the classification process, grounding their predictions in these elements. However, they typically rely on specific assumptions, such as that the interpretable layer is attached to the final layer and that the backbone features are representative and well learned. These two assumptions do not typically hold for EENNs, since ICs are distributed across intermediate layers, where features may be noisy or underdeveloped.

In this work, we propose Memory-based Self-Explainable Early Exit Networks (M-SEEEN), a design to make the ICs of exit networks self-interpretable. Inspired by [16], M-SEEEN uses a sparse content-based attention mechanism to select relevant examples from a memory set to aid and explain the decision process. We show that, compared to alternative SENN designs, M-SEEEN better preserves the performance of black-box baselines while providing means to inspect its inner mechanisms.

## 2 Related Work

**Early Exit Neural Networks** Teerapittayanon et al. [31] introduced the concept of EENNs, an architecture equipped with ICs that allow sufficiently processed samples to exit before the network completes its full computation. This initial design has been improved through alternative EENN

Preprint.

architectures [8], exit criteria [15, 9], and training paradigms [14, 25, 6, 4, 1, 18]. One of the key challenges of EENNs is the reliability of features in early layers. Several approaches have been explored to address this issue. From the training perspective, approaches include training ICs independently on top of a pre-trained frozen backbone [10], joint training with the backbone [10, 18, 8], and mixed [14] training strategies. Other approaches propose to enrich the information available by cascading the predictions of earlier ICs to subsequent exits [35], maintaining fine-to-coarse feature maps throughout the network [8], or separating classification and feature extraction into independent branches [7]. Despite this progress, obtaining informative representations at early layers remains a fundamental challenge. In this context, interpretability represents another underexplored aspect of EENNs. In particular, to the best of our knowledge, existing work has used interpretability techniques to determine which features to keep or compress [18], to identify the features relevant to specific exits [26] or predictions [10], and to improve exit strategies [26]. Our work complements these directions by introducing a design specialized to work with noisy and underdeveloped features, making it particularly suited to EENNs. At the same time, the proposed design exposes information about its decision process and the evolution of decisions across exits, thus improving the interpretability of early-exit networks.

**Self-Explainable Neural Networks** SENNs are models that provide explanations alongside their predictions. Among the most popular families of SENNs, we can distinguish between Concept Bottleneck Models (CBMs) and Prototypical-Part Networks (ProtoPNets). CBMs [12] introduce an intermediate layer between the feature extractor and the classifier, trained so that each dimension corresponds to an interpretable concept and its value encodes the degree of the presence of a concept. The final classifier then predicts the class by combining the resulting concept scores. This framework has been improved by reducing its dependence on annotated datasets [23, 30, 27, 37] and by introducing alternative architectural and training designs [38, 33]. ProtoPNets [2] share the principle of introducing an interpretable bottleneck, but provide spatially grounded explanations by comparing regions of the input representation with a fixed set of learned prototypes. The literature has explored several directions, such as variations in the number of prototypes [28, 29, 22] and the improvement of the semantic quality of the explanations [5, 21, 34]. Beyond these two main families, a variety of SENNs ([20, 36, 16]) have been proposed for specific architectures, tasks, or application domains. Among these, our work is inspired by Memory Wrap [16], a lightweight memory-based module designed for small-data settings that can improve the predictive performance of deep neural networks while exposing information about their decision process. Compared with other SENNs, our approach does not require a strong pre-trained backbone, specialized training procedures or metric-learning objectives, and, more generally, does not assume a particular structure in the input or task, such as the availability of predefined concepts or fine-grained visual categories.

## 3 M-SEEEN

M-SEEEN is an early exit network that uses a memory module to augment ICs and provide interpretability. Specifically, let $F$ be a neural network with $n$ layers, $x$ the input, $f_j$ the $j$-th layer, and $F_j(\cdot) = f_j(f_{j-1}(\dots f_1(\cdot)))$ the composition of the first $j$ layers. M-SEEEN attaches a set of $E$ ICs to this structure, where each IC is attached to a distinct layer $f_j$. Each IC $E_j$ compares and combines the latent representation of an input $x$ at its previous layer $f_j$ and the **cached** latent representations at the same layer of a set of $k$ memory samples $\mathcal{M} = \{m_1, \dots, m_k\}$ randomly sampled from a pool of $N$ training representations. Specifically, M-SEEEN applies sparsemax [3, 19] to the cosine similarity between these representations and uses the resulting scores $\alpha^{(e)}$ as weights to compute a combination of the memory representations: $\mathbf{v}_m^{(e)} = \sum_{m_r \in \mathcal{M}} \alpha_r^{(e)} m_r^{(e)}$ . Finally, a linear classifier predicts the probability distribution $\hat{y}^{(e)} = \mathrm{softmax}(f_{\mathrm{linear}}^{e}\left([F_{j_e}(x), \mathbf{v}_m^{(e)}]\right))$ from the concatenation of the input representation and the weighted memory vector. As common in literature [10], given a confidence threshold $\tau$, early exiting is performed by evaluating the exits in order $e = 1, \dots, E$ and returning $\hat{y}^{(e)}$ at the first exit satisfying $\max \hat{y}^{(e)} > \tau$. If no exit exceeds $\tau$, then the prediction with the highest confidence over all $E$ exits is returned.

To **explain** its predictions, M-SEEEN exposes the memory samples assigned a non-zero weight by sparsemax. Inspecting the distribution of these samples across exits can provide insights into how the decision evolves and how concentrated or uncertain the model is at each stage. Among them, we identify two types of explanations. The *best explanation by example* is the memory sample with

Table 1: Accuracy (%) reached by competitors across inference budgets under **joint** training, averaged over 3 runs. **Bold** indicates the best SENN method, underline the best overall result.

| | WideResNet32-4-SDN | | | | | MSDNet | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Classifier | 25% | 50% | 75% | 100% | Max | 25% | 50% | 75% | 100% | Max |
| **CIFAR-10** | | | | | | | | | | |
| Black-box | 87.8 | 94.3 | 94.4 | 94.4 | 94.4 | 92.2 | 93.2 | 93.2 | 93.2 | 93.2 |
| M-SEEEN | **86.2** | **93.9** | 94.1 | 94.1 | 94.1 | 91.7 | 92.9 | 92.9 | 92.9 | 92.9 |
| LF-CBM | 74.6 | 91.8 | 94.0 | 94.1 | 94.1 | 91.6 | 92.5 | 92.5 | 92.5 | 92.6 |
| VLG-CBM | 72.8 | 91.7 | **94.4** | **94.4** | **94.4** | **92.1** | **93.2** | **93.2** | **93.2** | **93.2** |
| PIP-Net | 73.5 | 87.4 | 89.6 | 89.6 | 89.7 | 84.7 | 87.8 | 88.0 | 88.0 | 88.0 |
| ProtoPNet | 79.3 | 81.4 | 81.5 | 81.5 | 81.6 | 81.1 | 81.1 | 81.1 | 81.1 | 82.0 |
| **CIFAR-100** | | | | | | | | | | |
| Black-box | 60.9 | 74.0 | 76.1 | 75.9 | 76.2 | 64.2 | 72.0 | 72.8 | 72.7 | 72.9 |
| M-SEEEN | **59.0** | **73.2** | **76.4** | **76.2** | **76.4** | **62.8** | **71.1** | **72.6** | **72.4** | **72.6** |
| LF-CBM | 35.5 | 57.7 | 72.5 | 74.7 | 74.7 | 61.3 | 69.3 | 70.4 | 70.1 | 70.5 |
| VLG-CBM | 30.7 | 53.8 | 72.2 | 75.8 | 75.8 | 62.4 | 70.3 | 71.5 | 71.5 | 71.7 |
| PIP-Net | 34.7 | 47.1 | 51.8 | 51.8 | 52.2 | 46.0 | 53.2 | 55.1 | 55.0 | 55.3 |
| ProtoPNet | 41.7 | 44.9 | 45.7 | 44.9 | 45.8 | 39.9 | 43.5 | 44.1 | 44.0 | 44.2 |
| **TinyImageNet** | | | | | | | | | | |
| Black-box | 40.3 | 55.7 | 62.4 | 62.2 | 62.7 | 45.8 | 56.6 | 60.2 | 59.9 | 60.3 |
| M-SEEEN | **38.7** | **54.2** | **61.9** | **61.8** | **62.4** | **44.4** | **56.0** | **60.3** | **60.1** | **60.5** |
| LF-CBM | 15.6 | 36.2 | 53.7 | 60.0 | 60.0 | 40.4 | 50.3 | 55.3 | 55.1 | 55.5 |
| VLG-CBM | 18.6 | 36.1 | 54.7 | 61.6 | 61.7 | 42.2 | 52.3 | 57.2 | 57.2 | 57.6 |
| PIP-Net | 24.4 | 36.0 | 41.5 | 41.8 | 42.6 | 30.3 | 36.1 | 39.7 | 40.5 | 40.6 |
| ProtoPNet | 26.8 | 29.3 | 30.6 | 30.6 | 31.0 | 30.6 | 32.4 | 32.9 | 32.1 | 33.2 |

the highest weight among those predicted in the same class as the input. The *best counterfactual candidate* is instead the highest-weight memory sample predicted in a class different from that assigned to the input. Together, these explanations provide a view of the evidence considered at each exit and can help characterize both the prediction and its evolution throughout the network.

# 4 Experiments

## 4.1 Performance

**Setup** To validate our method, we compare its performance against the black-box baseline IC, implemented as either SDN [10] or MSDNet [8], two representative CBMs (LF-CBM [23] and VLG-CBM [30]) and two ProtoPNets (ProtoPNet [2] and PIP-Net [22]). We select these SENN competitors because they are among the most popular approaches and are relatively compatible with exit-network settings (see Section B for an extended discussion of their implementation and training). We use SDN and MSDNet as representative designs of EENN due to their flexibility in terms of combination with additional interpretable layers. We leave for future work the evaluation of additional recent EENNs. We evaluate all methods on CIFAR-10 [13], CIFAR-100 [13], and TinyImageNet [17], using WideResNet32-4 [39] and MSDNet as backbones. We set the size $k$ of the memory sets to 100 for CIFAR-10, and 1000 for both CIFAR-100 and TinyImageNet. The reader can find additional experiments with alternative backbones in Section A of the appendix. We report the computational overhead introduced by each interpretable classifier in Section D.

**Results** We follow [10] and quantify the performance of the exit network by sweeping over candidate exit thresholds and reporting the achieved accuracy at the closest computation budget in % of the full network as well as the maximum accuracy achieved with all considered thresholds.


Figure 1: Two sample trajectories when using an exit confidence threshold of 0.7 on CIFAR-10. For each exit, both the input and the top weighted memory images are shown with their predicted classes. The scores correspond to the confidence for the input and the weights for the memory images.

Table 1 shows that the performance of SENNs tends to degrade at lower computational budgets, with the largest gaps relative to the black-box baselines occurring at the earliest exits. This behavior is consistent with the limited semantics encoded in shallow representations, which may not yet support the complex concepts or prototypes required by these architectures. As depth increases and representations become more informative, SENNs progressively reduce or close the performance gap. In contrast, M-SEEEN exhibits substantially more stable performance across computational budgets and, overall, the smallest average gap relative to the black-box model. The difference is especially pronounced at the lowest computational budgets, where M-SEEEN outperforms the other SENN approaches by large margins. Similar trends are observed with alternative backbones (Section A).

### 4.2 Explanations

This section analyzes, discusses, and showcases the quality of explanations retrievable by M-SEEEN. Specifically, we will showcase how M-SEEEN can help users understand the prediction flow through the EENN and the extent to which M-SEEEN preserves the explanation quality typically associated with memory-based explanations. We additionally show how the memory can be used to determine the reliability of predictions in section C of the appendix.

**Visual Interpretation** Here, we show how M-SEEEN can help users analyze and understand the trajectory of samples across the ICs by analyzing the samples in fig. 1. The horse sample (top) corresponds to an easy sample, as the model exits at the first IC. In this case, all the selected memories strongly agree with the input in terms of both visual appearance and class. In the other sample (bottom), the model is uncertain, and this is reflected by the conflicting memory samples selected at the first two exits. Note that, in the first exit, the relatively high confidence may incorrectly suggest that the model is sufficiently confident in its prediction. At the third exit, the representation has become sufficiently rich and the model predicts the correct class. Accordingly, the selected memory samples mostly agree with the predicted class.

**Quantitative Analysis** A crucial question in the context of EENNs is whether the properties of memory-based explanations are preserved at intermediate exits. To investigate this, we follow the protocol of La Rosa et al. [16] and study the quality of explanations by example and counterfactuals retrieved from the memory across the exits. As metrics, we use *input non-representativeness* and

Table 2: Input and prediction non-representativeness per exit for MobileNet-SDN, averaged over 3 runs. Ours denotes M-SEEEN; Rand denotes the random baseline. Lower is better.

| | *Input non-representativeness* | | | | | | *Prediction non-representativeness* | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | 1 | 2 | 3 | 4 | 5 | 6 | 1 | 2 | 3 | 4 | 5 | 6 |
| **C100** | | | | | | | | | | | | |
| Ours | 2.29 | 2.83 | 2.64 | 2.63 | 2.06 | 1.97 | 0.69 | 0.34 | 0.21 | 0.13 | 0.05 | 0.03 |
| CHP | 2.28 | 2.82 | 2.60 | 2.60 | 2.11 | 2.03 | 0.75 | 0.44 | 0.32 | 0.20 | 0.09 | 0.05 |
| KNN$*$ | 2.24 | 2.75 | 2.54 | 2.55 | 2.04 | 1.95 | 0.74 | 0.42 | 0.28 | 0.18 | 0.07 | 0.04 |
| Rand | 2.65 | 3.22 | 3.07 | 3.17 | 2.57 | 2.50 | 0.78 | 0.44 | 0.30 | 0.19 | 0.09 | 0.04 |
| **TIM** | | | | | | | | | | | | |
| Ours | 2.15 | 2.66 | 3.40 | 4.11 | 3.49 | 2.79 | 1.21 | 0.67 | 0.34 | 0.16 | 0.04 | 0.03 |
| CHP | 2.13 | 2.62 | 3.37 | 4.06 | 3.51 | 2.82 | 1.25 | 0.76 | 0.43 | 0.24 | 0.08 | 0.05 |
| KNN$*$ | 2.10 | 2.58 | 3.33 | 4.01 | 3.44 | 2.74 | 1.24 | 0.72 | 0.40 | 0.22 | 0.07 | 0.04 |
| Rand | 2.36 | 2.88 | 3.65 | 4.43 | 3.81 | 3.13 | 1.27 | 0.77 | 0.40 | 0.21 | 0.06 | 0.04 |

Table 3: Results of the IM1, IIM1, and IM2 scores at different exits on MobileNet-SDN, averaged over 3 runs. Lower is better.

| | Exit | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| **C100** | IM1 | 0.978 | 0.979 | 0.983 | 0.988 | 0.991 | 0.991 |
| | IIM1 | 1.040 | 1.040 | 1.036 | 1.030 | 1.026 | 1.026 |
| | IM2 | 0.479 | 0.467 | 0.458 | 0.454 | 0.454 | 0.457 |
| **TIM** | IM1 | 0.983 | 0.983 | 0.985 | 0.984 | 0.985 | 0.992 |
| | IIM1 | 1.026 | 1.027 | 1.026 | 1.027 | 1.024 | 1.016 |
| | IM2 | 0.451 | 0.448 | 0.440 | 0.439 | 0.435 | 0.431 |

*prediction non-representativeness* for explanations by examples, and the IM1 [32], IIM1 [16], and IM2 [32] scores for counterfactuals. In this section, we report the results computed over all exits with MobileNet as a backbone architecture. However, similar results can be observed in other models, as discussed in Section E, together with a description of the metrics used for evaluation.

Table 2 shows a progressive degradation in the scores from the last exit toward the earliest one. However, because these metrics are computed from the logits, differences in magnitude across layers may affect their values. To determine whether this trend reflects an actual degradation in the quality of explanations, we compare them against explanations by example produced by the post-hoc methods CHP [11] and KNN$*$ [24], as well as against a sample randomly selected from the memory, following the evaluation protocol of La Rosa et al. [16]. We observe a similar degradation across all methods, including the random baseline. This suggests that the trend may be driven by layer-dependent differences rather than by a deterioration specific to memory-based explanations. Importantly, the explanations remain consistently better than the random baseline across exits, indicating that the retrieved examples continue to provide meaningful explanations even at earlier layers. Moreover, M-SEEEN achieves scores comparable to those of the post-hoc methods across all settings, while requiring less time to compute the explanations. These results are consistent with those previously observed in small data settings [16]. Conversely, the scores associated with counterfactual explanations (Table 3) remain more stable across exits, thus confirming their independence from the specific layer for which they are computed. Overall, these results suggest that memory-based explanations preserve their relative quality when attached to different stages of an EENN, supporting memory-based mechanisms as a promising approach for self-interpretable early-exit architectures.

## 5 Conclusion

In this work, we introduced M-SEEEN, a self-explainable architecture for Early Exit Neural Networks. We showed that M-SEEEN better preserves the predictive performance of black-box EENN baselines than alternative self-explainable approaches, while providing meaningful explanations at each exit.

Future work will investigate explainable exit strategies and architectural extensions aimed at further improving both predictive performance and explanation quality.

## References


[1] Mohammed Ayyat, Tamer Nadeem, and Bartosz Krawczyk. ClassyNet: Class-Aware Early-Exit Neural Networks for Edge Devices. *IEEE Internet of Things Journal*, 11(9):15113–15127, May 2024. ISSN 2327-4662. doi: 10.1109/JIOT.2023.3344120. URL https://ieeexplore.ieee.org/abstract/document/10365527.

[2] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: Deep learning for interpretable image recognition. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/adf7ee2dcf142b0e11888e72b43fcb75-Paper.pdf.

[3] Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. Adaptively sparse transformers. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2174–2184, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1223. URL https://aclanthology.org/D19-1223/.

[4] Yonghao Dong, Qiang He, Penghong Rui, Zhenzhe Zheng, Zhao Li, Feifei Chen, Hai Jin, and Yun Yang. EnViT: Enhancing the performance of early-exit vision transformers via exit-aware structured dropout-enabled self-distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 20852–20860, 2026.

[5] Jon Donnelly, Alina Jade Barnett, and Chaofan Chen. Deformable ProtoPNet: An Interpretable Image Classifier Using Deformable Prototypes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10265–10275, 2022.

[6] Yizeng Han, Yifan Pu, Zihang Lai, Chaofei Wang, Shiji Song, Junfeng Cao, Wenhui Huang, Chao Deng, and Gao Huang. Learning to Weight Samples for Dynamic Early-Exiting Networks. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 362–378, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-20083-0. doi: 10.1007/978-3-031-20083-0_22.

[7] Yizeng Han, Dongchen Han, Zeyu Liu, Yulin Wang, Xuran Pan, Yifan Pu, Chao Deng, Junlan Feng, Shiji Song, and Gao Huang. Dynamic perceiver for efficient visual recognition. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5969–5979. IEEE, 2023.

[8] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hk2aImxAb.

[9] Metod Jazbec, Alexander Timans, Tin Hadži Veljković, Kaspar Sakmann, Dan Zhang, Christian A. Naesseth, and Eric Nalisnick. Fast yet safe: Early-exiting with risk control. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=bbFjpasRgs.

[10] Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3301–3310. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/kaya19a.html.

[11] Eoin M. Kenny and Mark T. Keane. Explaining deep learning using examples: Optimal feature weighting methods for twin systems using post-hoc, explanation-by-example in xai. *Know.-Based Syst.*, 233(C), December 2021. ISSN 0950-7051. doi: 10.1016/j.knosys.2021.107530. URL https://doi.org/10.1016/j.knosys.2021.107530.

[12] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5338–5348. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/koh20a.html.

[13] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[14] Piotr Kubaty, Bartosz Wójcik, Bartłomiej Tomasz Krzepkowski, Monika Michaluk, Tomasz Trzcinski, Jary Pomponi, and Kamil Adamczewski. How to train your multi-exit model? analyzing the impact of training strategies. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=vhTPfOdwyQ.

[15] Piotr Kubaty, Filip Szatkowski, Grzegorz Choczyński, Eric Nalisnick, and Bartosz Wójcik. Rethinking calibration for early-exit neural networks. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=jTtfB7ur5m.

[16] Biagio La Rosa, Roberto Capobianco, and Daniele Nardi. A self-interpretable module for deep image classification on small data. *Applied Intelligence*, aug 2022. doi: 10.1007/s10489-022-03886-6.

[17] Y. Le and X. Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[18] Changyao Lin, Zhenming Chen, Ziyang Zhang, and Jie Liu. E3: Early exiting with explainable ai for real-time and accurate dnn inference in edge-cloud systems. In *Proceedings of the 23rd ACM Conference on Embedded Networked Sensor Systems*, SenSys '25, page 385–397, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400714795. doi: 10.1145/3715014.3722076. URL https://doi.org/10.1145/3715014.3722076.

[19] Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1614–1623, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/martins16.html.

[20] David Mascharka, Philip Tran, Ryan Soklaski, and Arjun Majumdar. Transparency by design: Closing the gap between performance and interpretability in visual reasoning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[21] Meike Nauta, Ron Van Bree, and Christin Seifert. Neural Prototype Trees for Interpretable Fine-grained Image Recognition. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14928–14938, Nashville, TN, USA, June 2021. IEEE. ISBN 978-1-6654-4509-2. doi: 10.1109/CVPR46437.2021.01469. URL https://ieeexplore.ieee.org/document/9577335/.

[22] Meike Nauta, Jörg Schlötterer, Maurice van Keulen, and Christin Seifert. Pip-net: Patch-based intuitive prototypes for interpretable image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2744–2753, June 2023.

[23] Tuomas Oikarinen, Subhro Das, Lam M. Nguyen, and Tsui-Wei Weng. Label-free concept bottleneck models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=FlCg47MNvBA.

[24] Nicolas Papernot and Patrick McDaniel. Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning. 2018.

[25] Mary Phuong and Christoph H Lampert. Distillation-based training for multi-exit architectures. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1355–1364, 2019.

[26] Haseena Rahmath P, Ajith Abraham, and Kuldeep Chaurasia. XAI-Exit: Interpretability-driven dynamic early exits for efficient and transparent dnn inference. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–12, 2026. doi: 10.1109/TNNLS.2026.3685408.

[27] Sukrut Rao, Sweta Mahajan, Moritz Böhle, and Bernt Schiele. Discover-then-name: Task-agnostic concept bottlenecks via automated concept discovery. In *European Conference on Computer Vision*, 2024.

[28] Dawid Rymarczyk, Łukasz Struski, Jacek Tabor, and Bartosz Zieliński. Protopshare: Prototypical parts sharing for similarity discovery in interpretable image classification. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 1420–1430, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383325. doi: 10.1145/3447548.3467245. URL https://doi.org/10.1145/3447548.3467245.

[29] Dawid Rymarczyk, Łukasz Struski, Michał Górszczak, Koryna Lewandowska, Jacek Tabor, and Bartosz Zieliński. Interpretable image classification with differentiable prototypes assignment. In *European Conference on Computer Vision*, pages 351–368. Springer, 2022.

[30] Divyansh Srivastava, Ge Yan, and Tsui-Wei Weng. VLG-CBM: Training concept bottleneck models with vision-language guidance. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=Jm2aK3sDJD.

[31] Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. BranchyNet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469, December 2016. doi: 10.1109/ICPR.2016.7900006. URL https://ieeexplore.ieee.org/abstract/document/7900006.

[32] Arnaud Van Looveren and Janis Klaise. Interpretable counterfactual explanations guided by prototypes. In Nuria Oliver, Fernando Pérez-Cruz, Stefan Kramer, Jesse Read, and Jose A. Lozano, editors, *Machine Learning and Knowledge Discovery in Databases. Research Track*, pages 650–665. Springer International Publishing, 2021. ISBN 978-3-030-86520-7.

[33] Moritz Vandenhirtz, Sonia Laguna, Ričards Marcinkevičs, and Julia E Vogt. Stochastic concept bottleneck models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=iSjqTQ5S1f.

[34] Jiaqi Wang, Huafeng Liu, Xinyue Wang, and Liping Jing. Interpretable Image Recognition by Constructing Transparent Embedding Space. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 875–884. IEEE, 2021. ISBN 978-1-6654-2812-5. doi: 10.1109/ICCV48922.2021.00093. URL https://ieeexplore.ieee.org/document/9709918/.

[35] Maciej Wołczyk, Bartosz Wójcik, Klaudia Bałazy, Igor T Podolak, Jacek Tabor, Marek Śmieja, and Tomasz Trzcinski. Zero time waste: Recycling predictions in early exit neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 2516–2528. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/149ef6419512be56a93169cd5e6fa8fd-Paper.pdf.

[36] Kai Yang, Yi Yang, Qiang Gao, Ting Zhong, Yong Wang, and Fan Zhou. Self-explainable next poi recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 2619–2623, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657967. URL https://doi.org/10.1145/3626772.3657967.

[37] Yue Yang, Artemis Panagopoulou, Shenghao Zhou, Daniel Jin, Chris Callison-Burch, and Mark Yatskar. Language in a bottle: Language model guided concept bottlenecks for interpretable image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19187–19197, 2023.

[38] Mert Yuksekgonul, Maggie Wang, and James Zou. Post-hoc concept bottleneck models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=nA5AZ8CEyow.

[39] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016. ISBN 1-901725-59-6. doi: 10.5244/C.30.87. URL https://dx.doi.org/10.5244/C.30.87.

# A Performance

In this section, we evaluate performance using the same protocol as in Section 4.1, with MobileNet and ResNet56 as additional SDN backbones. Table 4 reports these results, where M-SEEEN again performs within a small margin of the corresponding black-box classifier. On MobileNet, several methods are unable to operate at the 25% computational budget: M-SEEEN and ProtoPNet on CIFAR-100 and TinyImageNet, and PIP-Net on TinyImageNet. As shown in fig. 2, the additional computational overhead introduced by these methods causes the minimum average inference cost to exceed 25% of the base network. This is also true in the case of the ProtoPNet classifier on ResNet56-SDN for TinyImageNet. We also observe that VLG-CBM outperforms M-SEEEN on CIFAR-10; however, this does not persist on the more complex CIFAR-100 and TinyImageNet datasets.

# B Extended Setup

In this section, we provide details about changes made to the training or evaluation procedures and how the competitor SENN methods are incorporated with the exit network backbones.

**ICs Placement.** In the case of MSDNet [8] we follow their standard approach for IC placement. Otherwise, we follow SDN [10] to place the ICs and to train the backbone and the ICs jointly.

**PIP-Net** Adapting PIP-Net [22] as an internal classifier follows its original implementation.

We use a $1 \times 1$ convolution at each IC to project the varying number of backbone channels to a fixed dimensionality across exits. This ensures that all ICs have access to a sufficiently large number of prototypes. Specifically, we project the feature maps to 128 channels for CIFAR-10, 512 for CIFAR-100, and 1024 for TinyImageNet. The effect of this design choice is evaluated in Table 5, which reports performance with and without the additional convolution. The spatial resolution ($H \times W$) of the intermediate feature maps also varies across layers. To obtain more consistent inputs across ICs, we apply average pooling to a $4 \times 4$ grid for CIFAR-10 and CIFAR-100 and an $8 \times 8$ grid for TinyImageNet. When the backbone produces a smaller feature map, we retain its original resolution rather than upsampling it.

For a fair comparison with the other methods, we do not initialize the PIP-Net ICs with a pre-trained backbone. Instead, we extend the initial joint pre-training phase to 50 epochs, allowing the backbone and the prototypes of all ICs to be learned jointly. We then perform 100 additional epochs of joint training. During the first three epochs of this phase, all parameters except the classifiers are frozen, following the training procedure of PIP-Net. Since this optimization deviates from the original PIP-Net setting, we additionally evaluate a two-stage training procedure that more closely follows the original training protocol. In the first stage, we initialize the model with a backbone pre-trained on the target task and replace its final classifier with the PIP-Net architecture. We first pre-train the backbone and final classifier for 50 epochs, followed by 100 epochs of joint optimization of the backbone and PIP-Net classifier. During the first three epochs of this second phase, only the classifier is fine-tuned. In the second stage, we attach the PIP-Net ICs and freeze the backbone. We first train the ICs for 50 epochs to optimize their prototypes, followed by 50 epochs of training of all parameters within the PIP-Net ICs. During the first three epochs of this phase, only the classifiers are trained.

As shown in Table 6, performance remains poor under this training procedure, suggesting that the results observed with PIP-Net are not primarily caused by our joint-training adaptation.

During evaluation, we apply the same transformation used during training to the raw outputs of the non-negative linear layer before computing the softmax. This preserves the scale of the classifier inputs between training and evaluation. It also prevents PIP-Net from abstaining from making a prediction, which is necessary in our setting because each IC must produce a prediction for every input.

**ProtoPNet** Similar to PIP-Net, the implementation and training procedure of ProtoPNet [2] require minor adaptations for the early-exit setting. We preserve the original ProtoPNet architecture and use 10 prototypes per class for all datasets.

As with PIP-Net, we apply average pooling so that the feature grid is $4 \times 4$ for CIFAR-10 and CIFAR-100 and $8 \times 8$ for TinyImageNet. In our default setting, the backbone is randomly initialized. We therefore remove ProtoPNet's warm-up phase, whose purpose is to adapt the prototype vectors and add-on layers to a pre-trained backbone, and instead train the network jointly for 100 epochs. Because the backbone is trained from scratch, we increase its learning rate from $1 \cdot 10^{-4}$ to $1 \cdot 10^{-3}$. We perform prototype pushing at epochs 60, 70, and 100 and, after each push, optimize only the classifiers for 20 epochs. Following the original ProtoPNet setup, we do not apply data augmentation during training.

Since this training procedure differs from the original ProtoPNet protocol, we additionally evaluate a two-stage setting that more closely reproduces its original training conditions. In the first stage, we initialize the model with a backbone pre-trained on the target task and replace its classifier with the ProtoPNet classifier. During the first 5 of 100 training epochs, we optimize only the prototype vectors and add-on layers, while keeping the backbone and classifier frozen. For the remaining epochs, we jointly optimize the backbone, prototypes, and add-on layers while keeping the classifier frozen. At epochs 60, 70, and 100, we perform a prototype push followed by 20 epochs of classifier-only optimization. In the second stage, we freeze the trained backbone and attach the ProtoPNet ICs. We optimize the prototypes and add-on layers of the ICs for 50 epochs, perform a prototype push after the final epoch, and then train only the classifiers for 20 additional epochs.

As shown in Table 6, this procedure improves ProtoPNet's performance, but it remains below that of M-SEEEN trained using the corresponding two-stage setting.

**CBM** The architectures of LF-CBM [23] and VLG-CBM [30] follow their original implementations. Since both methods are trained on top of a frozen backbone, our "jointly trained" CBM setting refers to the backbone rather than to the CBM training procedure itself. Specifically, we first train the backbone jointly on the target task, replace the original ICs with the corresponding CBM-based ICs, freeze the backbone, and then train the CBM ICs on the resulting intermediate representations.

## C Prediction Reliability

Inspired by [16] we can flag the reliability of a model's prediction by simply looking at the sample in memory associated with the highest weight. We first define an **example** as a memory sample that was used in the memory set of the prediction and predicted as the same class as the input. Conversely, a **counterfactual** is a memory sample that was similarly used, however received a different prediction than the input.

We evaluate the performance of all classifiers in the network when the sample associated with the highest weight is a counterfactual or an example in Table 7. Across all classifiers, an example as the top-weighted memory sample clearly defines a set of more correct predictions as compared to a counterfactual. This follows from the retrieval mechanism: a prototypical sample sits well inside its own class cluster, so it retrieves memories predominantly from that cluster. A counterfactual explanation indicates the model is confused and likely the input sits near a decision boundary. This can quickly signal to a user that an otherwise confident prediction may be untrustworthy. It is logical that the example set generally grows with depth while the counterfactual set shrinks, as the more a sample is processed, the more closely it will sit within its class cluster.

## D Computational Cost

Here, we evaluate the additional computational cost introduced by the interpretable classifiers. Figure 2 reports the cumulative GFLOPs up to and including each classifier, together with the total GFLOPs required by the backbone. In most settings, the additional overhead introduced by the SENN classifiers remains small relative to the computational cost of the backbone.

MobileNet represents the main exception, as its comparatively low backbone cost makes the overhead of the additional classifiers more pronounced. The computational overhead varies across SENN architectures for different reasons. For ProtoPNet, the largest increase occurs on datasets with more classes, namely CIFAR-100 and TinyImageNet. This behavior follows from the number of prototypes scaling linearly with the number of classes, which consequently increases the cost of prototype matching. M-SEEEN also introduces substantial relative overhead when applied to

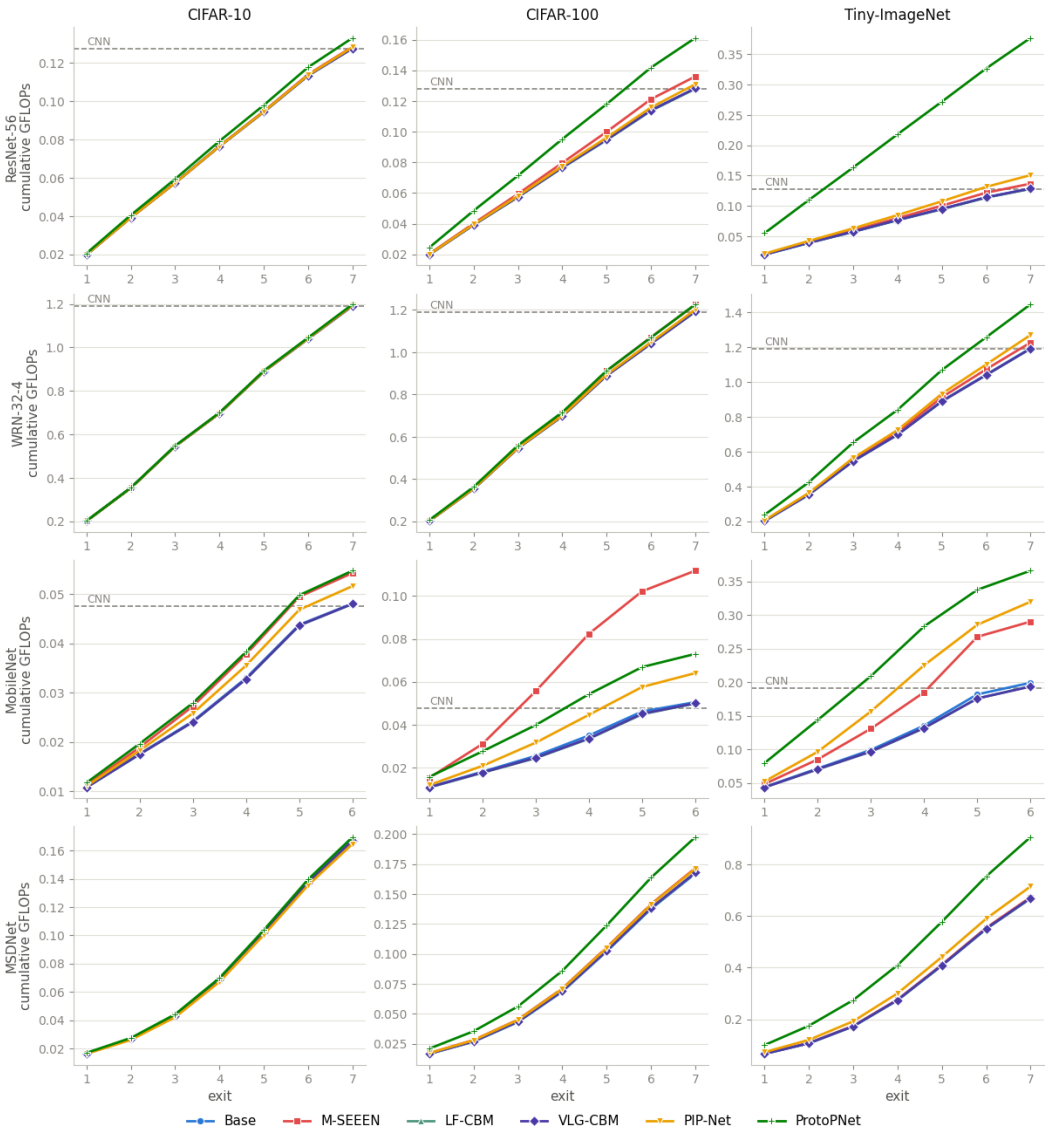


Figure 2: Cumulative compute (GFLOPs) at each internal classifier. The CNN column gives the GFLOPs of the unmodified backbone; MSDNet has no single-exit counterpart, so no baseline is reported.

MobileNet. The representations of the memory samples can be cached up to the distance-computation stage; therefore, at inference time, only the latent representation of the input must be compared against the cached representations of the memory set. However, M-SEEEN does not enforce a fixed feature dimensionality, and the cost of these comparisons consequently depends on the representations produced by the backbone at each exit. In the case of MobileNet, the backbone itself requires relatively few GFLOPs, while the cosine-similarity computation over the memory remains comparatively expensive. As a result, the relative computational overhead introduced by M-SEEEN is substantially larger at each IC.

# E Extended Explanation Quality

La Rosa et al. [16] showed that explanations by example and counterfactual explanations generated through the memory are comparable in quality to those produced by post-hoc baselines, while requiring lower computational cost. In this section, we provide more details about the metrics and the setup used to measure the quality of explanations, together with experiments on additional models.

## E.1 Metrics

For explanations by example, we use two metrics: *input non-representativeness*, which measures the $L_1$ distance between the logits of the current prediction and those of the selected explanation, and *prediction non-representativeness*, which measures the cross-entropy loss between the logits of the selected explanation and the predicted class. For counterfactual explanations, we use the IM1 and IM2 metrics introduced by Van Looveren and Klaise [32], together with the IIM1 score proposed by La Rosa et al. [16]. The **IM1** score is the ratio between the reconstruction error of the counterfactual under an autoencoder trained on samples from the input class and the reconstruction error under an autoencoder trained on samples from the counterfactual class. The **IM2** score measures the similarity between reconstructions of the counterfactual produced by an autoencoder trained on its predicted class and by an autoencoder trained on all classes. Finally, the **IIM1** score is the ratio between the reconstruction error of the counterfactual under an autoencoder trained on samples from the counterfactual class and that obtained using an autoencoder trained on samples from the input class.

## E.2 Explanation By Examples

By viewing Table 8, we can see that for the additional architectures the input non-representativeness of M-SEEEN is on-par with the post-hoc baselines. Similarly, Table 9 shows the prediction non-representativeness of M-SEEEN is almost always better than the post-hoc baselines, highlighting the quality of M-SEEEN's selected memory at all depths of the network.

## E.3 Counterfactuals

Table 10 reports the **IM1**, **IM2**, and **IIM1** scores on ResNet56-SDN, WideResNet32-4-SDN, and MS-DNet. We can again see that while an increase in depth will improve the selection of counterfactuals the scores stay within a small margin across classifiers of different depths.

Table 4: Results for additional SDN models. We report the accuracy (%) reached by competitors across inference budgets under **joint** training, averaged over 3 runs. **Bold** indicates the best SENN method, underline the best overall result. "—" denotes configurations that are unable to reach that compute budget. Conventions as in Table 1.

| **MobileNet-SDN** | | | | | | |
|---|---|---|---|---|---|---|
| Data | Classifier | 25% | 50% | 75% | 100% | Max |
| **C10** | Black-box | 84.7 | 90.4 | 90.4 | 90.4 | 90.4 |
| | M-SEEEN | **79.0** | **89.6** | 89.7 | 89.7 | 89.8 |
| | LF-CBM | 73.6 | 89.5 | **90.3** | **90.3** | **90.3** |
| | VLG-CBM | 68.0 | 89.2 | **90.3** | **90.3** | **90.3** |
| | PIP-Net | 47.6 | 77.5 | 71.9 | 67.6 | 77.8 |
| | ProtoPNet | 72.1 | 84.3 | 84.4 | 84.4 | 84.5 |
| **C100** | Black-box | 57.6 | 66.5 | 65.6 | 64.8 | 66.7 |
| | M-SEEEN | — | **59.6** | **64.1** | **66.1** | **66.5** |
| | LF-CBM | **37.9** | 56.9 | 62.4 | 62.7 | 62.9 |
| | VLG-CBM | 29.2 | 52.5 | 63.0 | 64.4 | 64.4 |
| | PIP-Net | 26.4 | 37.8 | 43.8 | 46.4 | 46.6 |
| | ProtoPNet | — | 47.8 | 51.2 | 51.3 | 51.8 |
| **TIM** | Black-box | 40.6 | 55.9 | 59.6 | 58.9 | 59.8 |
| | M-SEEEN | — | **50.0** | **56.8** | **59.6** | **59.8** |
| | LF-CBM | **21.3** | 40.4 | 51.6 | 54.2 | 54.3 |
| | VLG-CBM | 16.5 | 35.0 | 50.7 | 57.0 | 57.1 |
| | PIP-Net | — | 26.8 | 34.8 | 39.9 | 43.6 |
| | ProtoPNet | — | 32.4 | 35.3 | 36.9 | 37.5 |
| **ResNet56-SDN** | | | | | | |
| Data | Classifier | 25% | 50% | 75% | 100% | Max |
| **C10** | Black-box | 80.1 | 90.4 | 91.0 | 91.0 | 91.0 |
| | M-SEEEN | **77.1** | **89.6** | 90.5 | 90.5 | 90.5 |
| | LF-CBM | 43.5 | 78.8 | 90.1 | 90.1 | 90.8 |
| | VLG-CBM | 49.7 | 80.1 | **90.6** | **90.6** | **91.0** |
| | PIP-Net | 41.3 | 72.9 | 79.2 | 79.3 | 79.3 |
| | ProtoPNet | 75.6 | 82.4 | 82.4 | 82.4 | 82.4 |
| **C100** | Black-box | 47.7 | 64.2 | 68.9 | 67.9 | 69.0 |
| | M-SEEEN | **44.1** | **60.3** | **67.2** | **66.5** | **67.4** |
| | LF-CBM | 12.2 | 31.6 | 54.0 | 65.9 | 66.0 |
| | VLG-CBM | 14.0 | 32.9 | 54.9 | 65.8 | 66.7 |
| | PIP-Net | 20.6 | 31.8 | 39.3 | 41.7 | 41.9 |
| | ProtoPNet | 40.1 | 45.3 | 46.2 | 44.1 | 46.4 |
| **TIM** | Black-box | 30.2 | 44.1 | 52.7 | 52.1 | 53.2 |
| | M-SEEEN | **29.1** | **42.0** | **50.4** | **51.3** | **52.1** |
| | LF-CBM | 7.6 | 19.1 | 35.7 | 48.6 | 48.6 |
| | VLG-CBM | 12.3 | 23.2 | 38.5 | 49.6 | 49.6 |
| | PIP-Net | 12.8 | 19.1 | 24.5 | 28.1 | 28.8 |
| | ProtoPNet | — | 28.3 | 29.9 | 31.4 | 33.7 |

Table 5: Accuracy (%) at fractions of the base network's compute budget for a WideResNet32-4-SDN trained jointly with PIP-Net ICs, with and without the $1 \times 1$ convolution at each IC. Removing the $1 \times 1$ conv decreases the number of prototypes at each IC. Results are mean $\pm$ std; "Max" is the highest accuracy reached at any budget.

| Data | Conv | 25% | 50% | 75% | 100% | Max |
|---|---|---|---|---|---|---|
| **C10** | ✓ | $73.55 \pm 0.76$ | $87.36 \pm 0.11$ | $89.61 \pm 0.57$ | $89.61 \pm 0.57$ | $89.74 \pm 0.56$ |
| | ✗ | $61.04 \pm 6.41$ | $84.51 \pm 0.41$ | $88.69 \pm 0.48$ | $88.69 \pm 0.48$ | $88.99 \pm 0.60$ |
| **C100** | ✓ | $34.72 \pm 0.71$ | $47.13 \pm 1.71$ | $51.77 \pm 2.67$ | $51.83 \pm 2.97$ | $52.17 \pm 2.94$ |
| | ✗ | $13.18 \pm 2.51$ | $34.09 \pm 1.68$ | $48.68 \pm 2.19$ | $50.80 \pm 2.02$ | $51.13 \pm 2.14$ |

Table 6: Performance of two-stage training on prototype competitors and M-SEEEN, averaged over 3 runs.

| **ResNet56-SDN** | | | | | | |
|---|---|---|---|---|---|---|
| Data | Classifier | 25% | 50% | 75% | 100% | Max |
| **C10** | M-SEEEN | 65.7 | 84.6 | 90.4 | 90.5 | 90.6 |
| | PIP-Net | 22.7 | 66.1 | 87.1 | 89.1 | 89.1 |
| | ProtoPNet | 59.8 | 79.1 | 88.0 | 88.9 | 89.4 |
| **C100** | M-SEEEN | 35.9 | 52.8 | 64.5 | 65.5 | 66.2 |
| | PIP-Net | 15.0 | 29.4 | 46.5 | 54.2 | 54.2 |
| | ProtoPNet | 26.6 | 37.8 | 49.9 | 59.2 | 62.1 |

Table 7: Accuracy, averaged over 3 runs, split by whether the top-weighted memory sample is an example (predicted the same class as the input) or a counterfactual (predicted a different class). Coverage is the percentage of evaluation samples whose top-weighted memory is an example; the remainder fall in the counterfactual row. MobileNet-SDN has six exits.

| | | Exit | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **MobileNet-SDN** | | | | | | | | |
| C10 | Example | 83.6 | 91.8 | 93.7 | 92.7 | 91.6 | 91.4 | N/A |
| | Counterfactual | 71.7 | 80.0 | 78.5 | 62.6 | 54.5 | 54.4 | N/A |
| | Coverage | 40.9 | 53.3 | 62.0 | 90.3 | 94.9 | 95.4 | N/A |
| C100 | Example | 71.2 | 81.0 | 82.2 | 81.2 | 78.7 | 76.6 | N/A |
| | Counterfactual | 47.6 | 55.9 | 57.8 | 53.1 | 45.5 | 39.0 | N/A |
| | Coverage | 20.8 | 27.5 | 37.1 | 47.2 | 59.5 | 67.8 | N/A |
| TIM | Example | 52.1 | 67.7 | 75.8 | 78.5 | 78.6 | 76.3 | N/A |
| | Counterfactual | 34.6 | 44.5 | 49.7 | 50.3 | 49.5 | 44.2 | N/A |
| | Coverage | 9.8 | 12.7 | 15.9 | 20.6 | 30.4 | 44.8 | N/A |
| **ResNet56-SDN** | | | | | | | | |
| C10 | Example | 73.0 | 81.1 | 90.5 | 92.7 | 94.1 | 94.1 | 92.5 |
| | Counterfactual | 60.4 | 66.5 | 76.3 | 77.1 | 75.6 | 67.4 | 52.3 |
| | Coverage | 38.7 | 43.4 | 44.2 | 65.0 | 76.1 | 86.8 | 94.8 |
| C100 | Example | 53.5 | 60.1 | 74.4 | 77.5 | 82.7 | 83.4 | 79.0 |
| | Counterfactual | 33.7 | 38.1 | 49.6 | 52.3 | 56.5 | 54.0 | 46.3 |
| | Coverage | 19.1 | 21.7 | 25.9 | 30.2 | 36.5 | 44.9 | 61.3 |
| TIM | Example | 34.8 | 41.9 | 57.2 | 63.1 | 70.5 | 73.2 | 70.0 |
| | Counterfactual | 23.2 | 26.5 | 37.8 | 40.6 | 45.9 | 45.2 | 40.2 |
| | Coverage | 13.8 | 12.9 | 14.6 | 15.8 | 18.1 | 22.0 | 36.7 |
| **WideResNet32-4-SDN** | | | | | | | | |
| C10 | Example | 84.2 | 86.6 | 93.5 | 95.6 | 95.2 | 95.5 | 95.1 |
| | Counterfactual | 70.6 | 72.8 | 82.6 | 81.8 | 75.6 | 63.1 | 48.1 |
| | Coverage | 46.5 | 50.4 | 56.6 | 73.6 | 87.9 | 95.4 | 98.0 |
| C100 | Example | 70.9 | 73.8 | 82.3 | 84.8 | 86.8 | 87.7 | 81.9 |
| | Counterfactual | 48.0 | 50.2 | 59.3 | 61.6 | 62.1 | 59.2 | 38.6 |
| | Coverage | 22.1 | 24.8 | 28.5 | 34.1 | 43.7 | 59.3 | 86.4 |
| TIM | Example | 48.2 | 52.8 | 70.1 | 74.2 | 78.4 | 80.2 | 74.0 |
| | Counterfactual | 32.0 | 34.0 | 45.1 | 49.3 | 51.9 | 52.0 | 41.5 |
| | Coverage | 11.9 | 14.1 | 15.8 | 18.7 | 24.9 | 34.7 | 62.5 |
| **MSDNet** | | | | | | | | |
| C10 | Example | 86.4 | 88.6 | 90.8 | 91.9 | 92.8 | 93.8 | 94.0 |
| | Counterfactual | 57.4 | 58.8 | 57.9 | 55.6 | 55.2 | 54.9 | 56.4 |
| | Coverage | 85.9 | 88.2 | 91.2 | 93.4 | 95.7 | 96.8 | 97.3 |
| C100 | Example | 60.9 | 68.3 | 73.5 | 76.7 | 78.5 | 80.4 | 81.8 |
| | Counterfactual | 37.3 | 42.7 | 47.0 | 49.4 | 49.6 | 48.3 | 49.0 |
| | Coverage | 49.8 | 50.6 | 54.3 | 58.4 | 65.0 | 71.3 | 71.5 |
| TIM | Example | 43.0 | 54.2 | 61.5 | 67.1 | 70.6 | 73.9 | 77.1 |
| | Counterfactual | 25.7 | 32.9 | 38.5 | 42.3 | 44.7 | 46.5 | 46.6 |
| | Coverage | 35.5 | 32.8 | 33.4 | 34.4 | 38.3 | 42.7 | 44.4 |

Table 8: Input non-representativeness per exit for additional backbones, mean over 3 runs. Lower is better.

| **ResNet56-SDN** | | Exit | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Dataset | Method | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **C10** | M-SEEEN | 1.60 | 1.87 | 2.44 | 2.94 | 3.37 | 3.73 | 1.73 |
| | CHP | 1.60 | 1.87 | 2.38 | 2.82 | 3.20 | 3.64 | 1.78 |
| | KNN∗ | 1.57 | 1.84 | 2.32 | 2.75 | 3.12 | 3.51 | 1.64 |
| | Random | 1.89 | 2.17 | 2.76 | 3.34 | 3.77 | 4.28 | 2.51 |
| **C100** | M-SEEEN | 1.76 | 1.98 | 2.45 | 2.82 | 3.20 | 3.84 | 2.74 |
| | CHP | 1.76 | 1.99 | 2.44 | 2.81 | 3.20 | 3.88 | 2.87 |
| | KNN∗ | 1.71 | 1.94 | 2.37 | 2.72 | 3.09 | 3.72 | 2.67 |
| | Random | 2.13 | 2.38 | 2.87 | 3.28 | 3.67 | 4.38 | 3.48 |
| **TIM** | M-SEEEN | 1.59 | 1.88 | 2.31 | 2.56 | 2.81 | 3.18 | 2.62 |
| | CHP | 1.60 | 1.86 | 2.31 | 2.57 | 2.79 | 3.17 | 2.66 |
| | KNN∗ | 1.56 | 1.81 | 2.24 | 2.49 | 2.72 | 3.08 | 2.54 |
| | Random | 1.85 | 2.11 | 2.58 | 2.84 | 3.06 | 3.43 | 3.04 |

| **WideResNet32-4-SDN** | | Exit | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Dataset | Method | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **C10** | M-SEEEN | 2.15 | 2.37 | 3.20 | 3.80 | 3.79 | 3.91 | 0.90 |
| | CHP | 2.10 | 2.33 | 3.08 | 3.64 | 3.69 | 3.71 | 0.97 |
| | KNN∗ | 2.08 | 2.29 | 3.03 | 3.56 | 3.52 | 3.60 | 0.86 |
| | Random | 2.45 | 2.73 | 3.54 | 4.29 | 4.26 | 4.46 | 1.30 |
| **C100** | M-SEEEN | 2.31 | 2.49 | 3.13 | 3.48 | 3.94 | 4.43 | 1.19 |
| | CHP | 2.28 | 2.48 | 3.05 | 3.38 | 3.92 | 4.40 | 1.32 |
| | KNN∗ | 2.22 | 2.41 | 2.99 | 3.31 | 3.78 | 4.24 | 1.18 |
| | Random | 2.67 | 2.89 | 3.50 | 3.88 | 4.41 | 4.98 | 1.50 |
| **TIM** | M-SEEEN | 2.17 | 2.30 | 2.91 | 3.24 | 3.84 | 4.44 | 1.66 |
| | CHP | 2.13 | 2.28 | 2.87 | 3.20 | 3.82 | 4.43 | 1.78 |
| | KNN∗ | 2.09 | 2.21 | 2.80 | 3.13 | 3.70 | 4.28 | 1.64 |
| | Random | 2.39 | 2.56 | 3.14 | 3.49 | 4.14 | 4.78 | 1.94 |

| **MSDNet** | | Exit | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Dataset | Method | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **C10** | M-SEEEN | 0.96 | 1.15 | 1.24 | 1.29 | 1.24 | 1.15 | 1.61 |
| | CHP | 0.98 | 1.15 | 1.22 | 1.24 | 1.17 | 1.07 | 1.49 |
| | KNN∗ | 0.88 | 1.05 | 1.12 | 1.15 | 1.09 | 1.01 | 1.42 |
| | Random | 1.38 | 1.60 | 1.72 | 1.77 | 1.71 | 1.59 | 2.19 |
| **C100** | M-SEEEN | 1.09 | 1.36 | 1.54 | 1.68 | 1.76 | 1.75 | 2.46 |
| | CHP | 1.13 | 1.42 | 1.60 | 1.74 | 1.83 | 1.81 | 2.54 |
| | KNN∗ | 1.04 | 1.30 | 1.47 | 1.60 | 1.68 | 1.68 | 2.38 |
| | Random | 1.43 | 1.75 | 1.95 | 2.11 | 2.19 | 2.18 | 3.04 |
| **TIM** | M-SEEEN | 1.07 | 1.42 | 1.65 | 1.83 | 2.02 | 2.20 | 3.01 |
| | CHP | 1.10 | 1.47 | 1.70 | 1.88 | 2.07 | 2.26 | 3.06 |
| | KNN∗ | 1.04 | 1.38 | 1.60 | 1.78 | 1.96 | 2.14 | 2.91 |
| | Random | 1.34 | 1.72 | 1.97 | 2.17 | 2.37 | 2.56 | 3.43 |

Table 9: Prediction non-representativeness per exit, mean over 3 runs. Lower is better.

| **ResNet56-SDN** | | Exit | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Dataset | Method | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **C10** | M-SEEEN | 0.45 | 0.31 | 0.12 | 0.06 | 0.02 | 0.01 | 0.01 |
| | CHP | 0.47 | 0.32 | 0.17 | 0.10 | 0.05 | 0.02 | 0.01 |
| | KNN∗ | 0.47 | 0.32 | 0.15 | 0.09 | 0.04 | 0.01 | 0.01 |
| | Random | 0.49 | 0.36 | 0.17 | 0.10 | 0.05 | 0.02 | 0.01 |
| **C100** | M-SEEEN | 1.16 | 0.92 | 0.55 | 0.39 | 0.21 | 0.09 | 0.10 |
| | CHP | 1.20 | 0.95 | 0.62 | 0.47 | 0.30 | 0.16 | 0.15 |
| | KNN∗ | 1.20 | 0.95 | 0.61 | 0.45 | 0.28 | 0.14 | 0.13 |
| | Random | 1.22 | 1.01 | 0.66 | 0.50 | 0.30 | 0.14 | 0.15 |
| **TIM** | M-SEEEN | 1.69 | 1.44 | 0.94 | 0.78 | 0.55 | 0.37 | 0.39 |
| | CHP | 1.72 | 1.49 | 1.01 | 0.88 | 0.68 | 0.52 | 0.50 |
| | KNN∗ | 1.72 | 1.48 | 0.98 | 0.84 | 0.62 | 0.47 | 0.48 |
| | Random | 1.75 | 1.52 | 1.06 | 0.90 | 0.66 | 0.46 | 0.49 |

| **WideResNet32-4-SDN** | | Exit | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Dataset | Method | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **C10** | M-SEEEN | 0.22 | 0.17 | 0.05 | 0.01 | 0.00 | 0.00 | 0.00 |
| | CHP | 0.25 | 0.20 | 0.08 | 0.03 | 0.01 | 0.00 | 0.00 |
| | KNN∗ | 0.25 | 0.20 | 0.07 | 0.03 | 0.01 | 0.00 | 0.00 |
| | Random | 0.27 | 0.22 | 0.08 | 0.02 | 0.01 | 0.00 | 0.00 |
| **C100** | M-SEEEN | 0.64 | 0.52 | 0.25 | 0.15 | 0.06 | 0.01 | 0.01 |
| | CHP | 0.73 | 0.60 | 0.33 | 0.23 | 0.12 | 0.02 | 0.02 |
| | KNN∗ | 0.72 | 0.58 | 0.31 | 0.21 | 0.10 | 0.02 | 0.02 |
| | Random | 0.75 | 0.63 | 0.34 | 0.23 | 0.10 | 0.02 | 0.02 |
| **TIM** | M-SEEEN | 1.11 | 0.95 | 0.52 | 0.38 | 0.19 | 0.05 | 0.08 |
| | CHP | 1.20 | 1.01 | 0.62 | 0.46 | 0.30 | 0.11 | 0.14 |
| | KNN∗ | 1.18 | 1.00 | 0.58 | 0.44 | 0.26 | 0.10 | 0.10 |
| | Random | 1.22 | 1.05 | 0.62 | 0.47 | 0.26 | 0.09 | 0.13 |

| **MSDNet** | | Exit | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Dataset | Method | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **C10** | M-SEEEN | 0.09 | 0.06 | 0.03 | 0.01 | 0.00 | 0.00 | 0.00 |
| | CHP | 0.15 | 0.10 | 0.05 | 0.03 | 0.01 | 0.00 | 0.00 |
| | KNN∗ | 0.14 | 0.09 | 0.05 | 0.02 | 0.01 | 0.00 | 0.00 |
| | Random | 0.18 | 0.11 | 0.06 | 0.03 | 0.01 | 0.00 | 0.00 |
| **C100** | M-SEEEN | 0.68 | 0.45 | 0.30 | 0.20 | 0.11 | 0.06 | 0.02 |
| | CHP | 0.86 | 0.61 | 0.42 | 0.30 | 0.18 | 0.10 | 0.04 |
| | KNN∗ | 0.82 | 0.56 | 0.38 | 0.27 | 0.16 | 0.08 | 0.04 |
| | Random | 0.89 | 0.63 | 0.44 | 0.31 | 0.19 | 0.10 | 0.04 |
| **TIM** | M-SEEEN | 1.43 | 1.04 | 0.81 | 0.62 | 0.46 | 0.29 | 0.12 |
| | CHP | 1.58 | 1.18 | 0.94 | 0.75 | 0.57 | 0.38 | 0.17 |
| | KNN∗ | 1.54 | 1.14 | 0.91 | 0.72 | 0.54 | 0.36 | 0.16 |
| | Random | 1.57 | 1.19 | 0.94 | 0.75 | 0.57 | 0.39 | 0.17 |

Table 10: Results of the IM1, IIM1, and IM2 scores at different exits on ResNet56-SDN, WideResNet32-4-SDN, and MSDNet, averaged over 3 runs. Lower is better.

| **ResNet56-SDN** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Exit | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **C10** | IM1 | 0.989 | 0.987 | 0.984 | 0.984 | 0.983 | 0.982 | 0.994 |
| | IIM1 | 1.035 | 1.043 | 1.040 | 1.040 | 1.044 | 1.051 | 1.033 |
| | IM2 | 0.255 | 0.260 | 0.254 | 0.252 | 0.254 | 0.254 | 0.249 |
| **C100** | IM1 | 0.982 | 0.981 | 0.977 | 0.979 | 0.979 | 0.981 | 0.992 |
| | IIM1 | 1.035 | 1.037 | 1.040 | 1.039 | 1.039 | 1.039 | 1.023 |
| | IM2 | 0.471 | 0.470 | 0.467 | 0.463 | 0.463 | 0.463 | 0.460 |
| **TIM** | IM1 | 0.985 | 0.983 | 0.985 | 0.985 | 0.985 | 0.985 | 0.992 |
| | IIM1 | 1.025 | 1.032 | 1.024 | 1.025 | 1.025 | 1.026 | 1.016 |
| | IM2 | 0.441 | 0.439 | 0.440 | 0.440 | 0.438 | 0.436 | 0.430 |

| **WideResNet32-4-SDN** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Exit | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **C10** | IM1 | 0.986 | 0.987 | 0.986 | 0.988 | 0.986 | 0.983 | 0.997 |
| | IIM1 | 1.032 | 1.033 | 1.032 | 1.032 | 1.036 | 1.050 | 1.029 |
| | IM2 | 0.246 | 0.249 | 0.246 | 0.247 | 0.246 | 0.252 | 0.247 |
| **C100** | IM1 | 0.975 | 0.977 | 0.979 | 0.980 | 0.979 | 0.980 | 0.993 |
| | IIM1 | 1.041 | 1.039 | 1.037 | 1.037 | 1.039 | 1.038 | 1.022 |
| | IM2 | 0.476 | 0.479 | 0.476 | 0.469 | 0.462 | 0.458 | 0.459 |
| **TIM** | IM1 | 0.983 | 0.985 | 0.986 | 0.986 | 0.987 | 0.988 | 0.995 |
| | IIM1 | 1.025 | 1.024 | 1.022 | 1.023 | 1.022 | 1.021 | 1.012 |
| | IM2 | 0.443 | 0.444 | 0.442 | 0.440 | 0.436 | 0.435 | 0.430 |

| **MSDNet** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Exit | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **C10** | IM1 | 0.996 | 0.998 | 0.996 | 0.999 | 0.998 | 0.997 | 0.996 |
| | IIM1 | 1.030 | 1.026 | 1.029 | 1.026 | 1.027 | 1.026 | 1.030 |
| | IM2 | 0.250 | 0.247 | 0.247 | 0.244 | 0.246 | 0.245 | 0.246 |
| **C100** | IM1 | 0.988 | 0.987 | 0.988 | 0.990 | 0.990 | 0.990 | 0.990 |
| | IIM1 | 1.026 | 1.027 | 1.025 | 1.025 | 1.025 | 1.025 | 1.025 |
| | IM2 | 0.453 | 0.455 | 0.455 | 0.455 | 0.456 | 0.457 | 0.455 |
| **TIM** | IM1 | 0.993 | 0.992 | 0.992 | 0.992 | 0.992 | 0.994 | 0.993 |
| | IIM1 | 1.015 | 1.016 | 1.016 | 1.015 | 1.015 | 1.013 | 1.014 |
| | IM2 | 0.423 | 0.425 | 0.426 | 0.426 | 0.426 | 0.429 | 0.430 |