Neurosymbolic Learning With Random Forest

Training Random Forests on Neural Network Layer Activations

Michal Barnišin[1,*], Lubomír Popelínský[1]

[1]Faculty of Informatics, Masaryk University, Czech Republic

Abstract

Neurosymbolic learning combines the representational power of neural networks with the data efficiency and interpretability of symbolic methods. In this thesis, we investigate a hybrid approach using GoogLeNet as a feature extractor and a random forest classifier trained on intermediate neuron activations. Focusing on moderately small image datasets, we show that this combination can improve classification accuracy compared to the neural network alone. Furthermore, we analyze the training time and find that the hybrid model can reach the neural network's peak accuracy in less time, depending on the layer used for feature extraction.

Keywords

neurosymbolic learning, random forest, GoogLeNet, feature extraction, image classification, training efficiency

1. Introduction

Neural networks (NNs) have driven major advances in machine learning across domains such as computer vision, natural language processing, and bioinformatics. Their success stems from their ability to learn high-level representations directly from raw data. However, this comes at a cost: training deep networks often requires large datasets, significant computational resources, and dedicated hardware.

In contrast, traditional models like decision trees and random forests (RFs) offer an efficient training procedure and strong interpretability. RFs, in particular, are competitive on small or tabular datasets, often outperforming neural networks [1, 2]. Despite their simplicity, these models struggle with high-dimensional unstructured data, such as images or text, where deep networks excel.

This motivates neurosymbolic learning, a paradigm that combines the representational power of NNs with the efficiency of symbolic or rule-based methods. In this work, we investigate a hybrid architecture where a random forest classifier is trained on intermediate activations of a pretrained neural network. We focus on small-scale image datasets (1,000 and 10,000 samples).

It is known that an appropriate combination of a neural network and a symbolic classifier leads to increased accuracy. In this work, we take it a step further and aim to explore how to best construct such a combination. Specifically, we aim to answer the following research questions:

1. Which layer is the best for increasing accuracy or reducing training time without a significant loss in accuracy?
2. How many epochs are needed to achieve the goal?
3. Do we need the activations of all neurons in a layer?

We evaluate this hybrid setup using GoogLeNet [3] as a feature extractor and train RFs on selected subsets of activation layers. Our findings suggest that combining lightweight neural feature extraction with fast tree-based classifiers is a promising direction, particularly in low-data scenarios.

This article is based on research conducted as part of the Master's thesis [4].

ITAT'25: Information Technologies – Applications and Theory, September 26–30, 2025, Telgárt, Slovakia

*Corresponding author.

michal.barnisin14@gmail.com (M. Barnišin); lubospopelinsky@gmail.com (L. Popelínský)

2. Related Work

The relation between neural networks and decision trees (and random forests) has been studied multiple times [5, 6]. The main motivation to combine these methods is to increase the interpretability of the resulting model, increase accuracy or adapt NNs for problems with small datasets. [7] presents a well organised survey of neural trees—the different combinations of NNs and DTs.

One common direction uses NNs as feature extractors, feeding intermediate activations into tree-based models. This allows the NN to learn high-level representations, while the tree model performs final classification. For example, [8, 9] trained RFs on the final layer of convolutional neural networks (CNNs) for medical and aerial image classification, reporting higher accuracy and better robustness, especially on small datasets.

Neural-backed Decision Trees [10] integrate soft decision trees into the architecture by reshaping the NN's final layer into a WordNet-aligned hierarchy, yielding interpretable misclassification paths without sacrificing performance.

A setup closer to ours is presented in [11], where RFs are trained on full-layer activations from multiple levels of a CNN, and their outputs are aggregated via voting. This leverages both low-level and high-level features.

In contrast to prior work, we systematically evaluate which layers are most effective for hybrid NN–RF classifiers, aiming for faster training and high accuracy with limited data. We also explore the possibility of combining different neuron activations from various layers and epochs.

3. Hybrid Classifier

We study a hybrid classification model that combines deep neural networks with random forests. The NN serves as a feature extractor, while the RF acts as the final classifier.

To train the model (see Algorithm 1), we fine-tune a pretrained GoogLeNet on an image classification task. During training, we periodically extract activations from five selected layers. These span from early convolutional outputs to the final logits. The collected activations serve as input features for a random forest classifier. The training alternates between updating the NN and training the RF on the neural network's activations.

During inference, samples are passed through the NN, and the RF uses the selected activations to predict the final class label.

Algorithm 1 Training Hybrid NN–RF Classifier

```
Require: Training dataset $\mathcal{D}_{\text{train}}$, set of NN layers $L$, stopping criteria stopping_criteria
1: Initialize neural network $\mathcal{N}^{(0)}$
2: for epoch $t = 1$ to $\infty$ do
3:     Train $\mathcal{N}^{(t)}$ on $\mathcal{D}_{\text{train}}$ from $\mathcal{N}^{(t-1)}$ for one epoch
4:     Extract activations $\mathcal{A}^{(t)}$ from layers $L$ on $\mathcal{D}_{\text{train}}$
5:     Train a RF $\mathcal{F}$ on some subsets of $\bigcup_{i\in 0\ldots t} \mathcal{A}^{(i)}$
6:     if stopping_criteria($\mathcal{N}^{(t)}, \mathcal{F}$) then
7:         return $\mathcal{N}^{(t)}, \mathcal{F}$
8:     end if
9: end for
```

3.1. Neural Network Setup

We use the PyTorch implementation of GoogLeNet, selected for its moderate size and compatibility with freely available GPU environments. The original final layer is replaced with a new fully connected layer, initialized with *Kaiming Uniform*. All layers are fine-tuned using the *Adam optimizer* (learning rate is fixed at 0.001), batch size 64, and cross-entropy loss. Auxiliary classifier heads are removed.

We extract activations from five layers (Table 1) distributed throughout the network to study the effect of feature depth.

Table 1
Selected GoogLeNet layers on which we will study the performance gains. *Layer Name* refers to the name of the module in the implementation of GoogLeNet in PyTorch. *Layer Size* represents the total number of neurons in the selected layer.

Layer Name	Layer Size
maxpool2	150,528
maxpool3	94,080
maxpool4	40,768
dropout	1,024
fc	10

3.2. Random Forest Settings

The RF classifier is implemented using scikit-learn. We fix the number of trees to 100 and maximum tree depth to 20, based on preliminary tuning on Fashion-MNIST. These hyperparameters remain constant across all experiments to ensure consistency. RFs are trained on flattened layer activations stored during NN training.

3.3. Datasets

We evaluate the hybrid classifier on three small-to-moderate image classification datasets:

- Fashion-MNIST [12]: 10 grayscale clothing classes.
- EMNIST (letters) [13]: 26-class handwritten character grayscale dataset.
- CIFAR-100 [14]: 100-class colored object dataset.

All datasets are resized to 224×224 pixels. Grayscale images are converted to 3-channel RGB format. Inputs are normalized using ImageNet statistics [15]. Since we focus on moderately small datasets only, the accuracy scores and training times are reported for random 1,000- and 10,000-sample subsets, averaged across multiple random selections.

4. Results

This section presents the empirical findings from our experiments comparing the performance and training efficiency of hybrid models to those of a fully trained GoogLeNet and a conventional fully trained Random Forest.

We evaluate classification accuracy, training time[1], and performance stability using different neural network layers and their combinations. The experiments include two specializations of Algorithm 1, where *some subsets* refers to:

- **Single-layer evaluation**: A separate RF is trained on activations from a single layer of GoogLeNet from the most recent epoch to identify the most informative layers.
- **Multi-layer and cross-epoch evaluation**: Activations from multiple layers and epochs are aggregated to find minimal configurations achieving strong performance with minimal training time.

[1]Measured on Intel(R) Xeon(R) CPU @ 2.00GHz and Tesla T4 GPU.

4.1. Main Findings

Our results show that:

1. A random forest trained on activations from a single layer of a NN can exceed the NN's classification accuracy, with the effect being more pronounced for deeper layers—at the cost of longer training time.
2. This hybrid architecture can match the NN's performance in less time, showing that this approach is viable in dynamic environments or prototyping.
3. We have also found that full-layer activations are often unnecessary; subsets of layers are equally informative and allow training time savings without loss in performance.

4.2. Baseline

To establish a baseline, we evaluated GoogLeNet trained end-to-end using cross-entropy loss and an RF trained directly on flattened input images. The results are in Table 2.

Table 2
Comparison of test accuracy score and training time of GoogLeNet (NN) and RF. Highlighted are the best accuracy scores for each dataset and sample size split.

Dataset	Size	Method	Acc	Time (s)
Fashion-MNIST	1,000	NN	0.760	40.8
	1,000	RF	**0.798**	0.9
	10,000	NN	**0.887**	612.0
	10,000	RF	0.849	11.1
EMNIST	1,000	NN	**0.811**	75.0
	1,000	RF	0.645	0.9
	10,000	NN	**0.917**	500.0
	10,000	RF	0.803	8.6
CIFAR-100	1,000	NN	0.071	100.0
	1,000	RF	**0.081**	11.2
	10,000	NN	**0.328**	882.0
	10,000	RF	0.162	96.3

RF trained on raw pixels generally underperforms GoogLeNet, with two exceptions: 1,000-sample Fashion-MNIST and 1,000-sample CIFAR-100, where GoogLeNet struggles due to insufficient data. However, RF trains substantially faster, 1–2 orders of magnitude faster in some cases.

4.3. Single Layer Evaluation

We evaluated RFs trained on the activations from individual layers of a pre-trained GoogLeNet. Specifically, for each pre-selected layer, we trained a separate RF to assess how informative different stages of neural network processing are for downstream tasks. This setup allows us to probe the representational quality of features extracted at various depths of the network. The performance of these models is summarized in Table 3.

Across all datasets, the hybrid NN+RF models consistently achieved **higher accuracy**, with improvements ranging from 2% to 6%. These results validate our core hypothesis and align with prior findings [8, 9, 11], where forests were shown to benefit from intermediate neural features.

Interestingly, the observed gains in accuracy and training efficiency were strongly **dependent on the selected layer**. Shallower layers tended to converge more quickly, but deeper—and often sparser—layers such as *dropout* or *maxpool4* ultimately delivered better predictive performance. This suggests that mid-to-deep layers strike a favorable balance between feature richness, dimensionality, and training time, especially when training a single-layer RF.

Table 3
Training times and achieved accuracy for the trained GoogLeNet (NN) and random forests trained on intermediate layer outputs. *Method* refers either to the NN or contains the name of the layer the RF was trained on, followed by the peak *Accuracy* by this method on the given dataset. *Time* contains the total training time for the method to achieve the peak accuracy. *Match NN* represents the time needed to match the performance of GoogLeNet.

Dataset	Size	Method	Accuracy	Time (s)	Match NN (s)
Fashion-MNIST	10,000	NN	0.887	612.0	–
		fc	0.900	786.0	235.8
		dropout	0.905	1030.0	309.0
		maxpool4	0.900	2110.0	633.0
		maxpool3	0.880	843.0	843.0
		maxpool2	0.860	225.0	–
	1,000	NN	0.760	40.8	–
		fc	0.800	66.4	24.9
		dropout	0.800	47.5	19.0
		maxpool4	0.820	73.0	14.6
		maxpool3	0.810	12.9	12.9
		maxpool2	0.800	13.8	13.8
EMNIST	10,000	NN	0.917	500.0	–
		fc	0.923	972.0	405.0
		dropout	0.935	1045.0	209.0
		maxpool4	0.923	874.0	437.0
	1,000	NN	0.811	75.0	–
		fc	0.848	113.4	56.7
		dropout	0.870	140.0	50.0
		maxpool4	0.849	245.0	70.0
CIFAR-100	10,000	NN	0.328	882.0	–
		fc	0.310	1827.0	–
		dropout	0.330	2790.0	2635.0
		maxpool4	0.210	5020.0	–
	1,000	NN	0.071	100.0	–
		fc	0.073	198.0	135.1
		dropout	0.080	280.0	144.9
		maxpool4	0.084	512.2	37.3

The hybrid approach also exhibited notably **lower variance across runs**, indicating more stable training dynamics. As illustrated in Figure 1, the NN+RF method consistently outperformed the NN baseline at nearly every epoch, maintaining a lead until convergence. This increased stability can be attributed to the RF's ability to mitigate the effects of noisy or suboptimal mini-batch selections during NN training—an issue especially pronounced in smaller datasets. By decoupling the prediction mechanism from stochastic gradient updates, the RF introduces an ensemble-based smoothing effect that dampens variance across training seeds.

Furthermore, the combined NN+RF approach maintains a consistent accuracy advantage across epochs. Although both the NN and hybrid methods tend to converge toward similar final performance, the RF-enhanced models often sustain a small but measurable lead. This suggests that RFs are particularly effective at leveraging the internal representations learned by NNs—sometimes even more so than linear classifiers typically used at the output layer.

Taken together, these results demonstrate that using RFs on top of neural features can provide both accuracy and stability gains, especially in scenarios with limited data or noisy optimization dynamics.

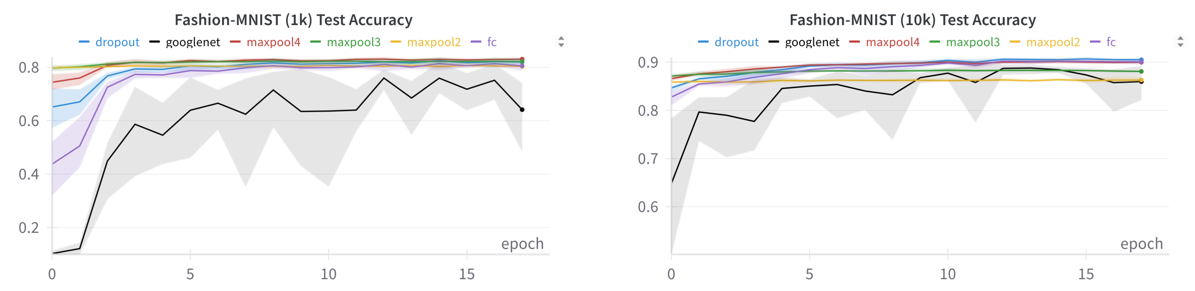

Figure 1: Test accuracy score for the FASHION-MNIST dataset with 1,000 (left) and 10,000 (right) samples. The x-axis corresponds to training epochs. The solid line represents the mean of five independent runs; the shaded region shows the min-max range. Epoch zero corresponds to the pre-trained GOOGLENET model with no fine-tuning. The black line indicates the NN performance, while the colored lines correspond to RFs trained on various layers. All models show improving accuracy, with the hybrid NN+RF setup consistently outperforming the pure NN baseline throughout training.

The approach requires no retraining of the neural network and can be flexibly applied to various intermediate layers, making it a lightweight and robust enhancement to existing NN pipelines.

4.4. Multi-Layer and Cross-Epoch Evaluation

We further examined whether the random forest could perform well when trained on only a subset of neuron activations from a pre-trained neural network. All experiments in this section were conducted on the FASHION-MNIST dataset, using a 1,000-sample subset due to memory constraints. We also adjusted the RF configuration by setting `min_samples_leaf = 5`, which encourages generalization by limiting the size of individual branches.

Surprisingly, training on as few as 1–10% of a layer's total neurons was often sufficient to match the accuracy obtained using the whole layer (Figure 2). This suggests a high degree of redundancy in the learned representation, and supports the idea that compact activation subsets can generalize effectively—particularly in low-data regimes.

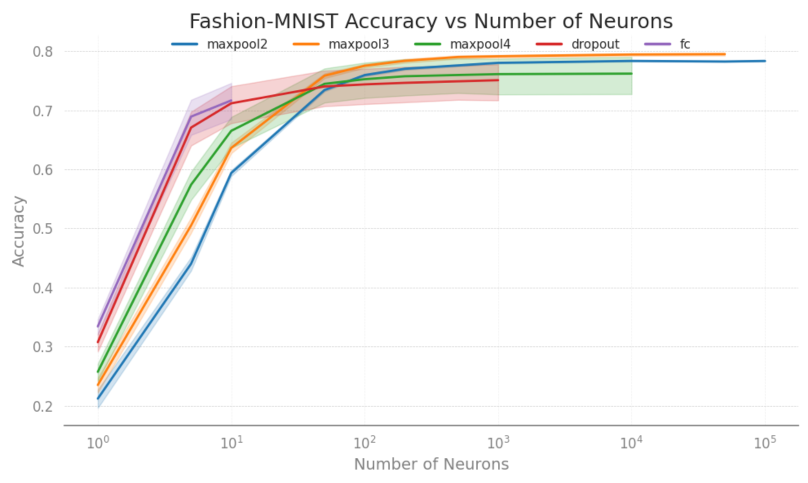

Figure 2: Accuracy score for different sizes of activation subsets from a fixed layer, on FASHION-MNIST (with 1,000 samples). Results are averaged across multiple random selections. While all ten neurons are needed for the final *fc* layer, for other layers, the accuracy reaches a plateau well before the full layer size is considered. E.g. for *maxpool3* layer, the 1,000 neuron subset behaves similarly to the full, 94,080 neurons wide layer.

We then explored randomly sampled configurations that draw neurons from multiple layers and epochs. As shown in Figure 3, most of these configurations outperformed the NN baseline in terms of

Table 4
Top RF configurations using subsets of neuron activations from GoogLeNet.

Epoch	Layer	Neurons	Accuracy	Time (s)
1	maxpool2	500	0.798	129.0
1	maxpool3	300		
5	maxpool3	50,000		
3	maxpool3	1,000	0.782	26.4
3	dropout	10		
1	maxpool3	10,000	0.768	22.2
2	maxpool3	10		
2	maxpool4	5		
2	fc	10		

accuracy, with training time primarily influenced by the number of epochs and the size of the feature set. Notably, high-performing configurations often included neurons from at least one later epoch, reinforcing the benefit of deeper or better-trained features. Poorer configurations, by contrast, tended to involve only very deep (but not fully trained) layers or very small feature sets (less than 100 neurons).

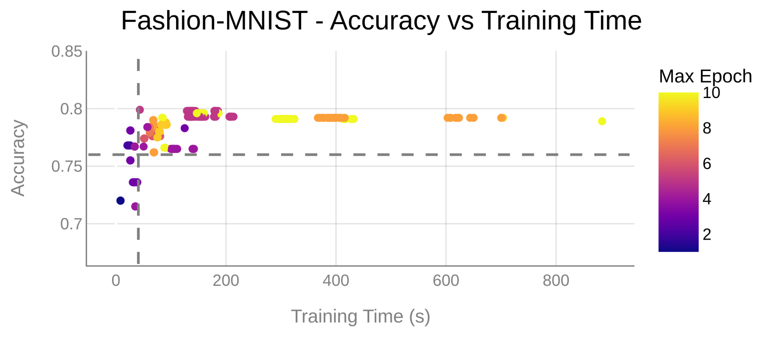

Figure 3: Maximal accuracy score for different combinations of layers and epochs, for Fashion-MNIST with 1,000 samples. The GoogLeNet 's performance is shown in the dashed line. The points are colour-coded based on the maximal epoch used in the combination—which is, alongside the feature set size, proportional to the training time.

Table 4 summarizes three top-performing configurations, balancing accuracy and training time. These results highlight the effectiveness of combining moderate numbers of neurons from well-trained intermediate layers.

Additionally, we found that the out-of-bag score of the RF strongly correlates with test accuracy across datasets, offering a practical alternative to held-out validation sets or a proxy for early stopping.

These findings suggest that with smart selection of activation subsets—preferably from mid-to-deep layers and later training epochs—it is possible to build compact, efficient, and competitive hybrid models
.

5. Discussion

We observed that deeper layers yielded better RF performance, in line with how NNs operate: early layers capture low-level features, while deeper layers encode task-relevant abstractions. Surprisingly, even randomly chosen subsets of activations provided sufficient signal, suggesting redundancy in representations and hinting at opportunities for feature selection and dimensionality reduction.

Several limitations remain. The RFs are trained offline, requiring all activations to be stored, thus limiting scalability. We also used random feature sampling rather than principled selection methods. Furthermore, we evaluated only one architecture (GoogLeNet), and did not explore how transferable the findings are to others. Finally, although RFs offer improved interpretability over NNs, we did not attempt to extract symbolic rules or explanations from them.

Importantly, our findings suggest that full convergence of a network may not be necessary if intermediate representations are already informative—opening the door to faster, hybrid learning pipelines in constrained environments.

It is worth mentioning that our approach parallels concept probing from XAI, where intermediate activations are used to detect specific concepts via simple classifiers [16]. Like concept probes, training a random forest on a layer's outputs can reveal which features or abstractions the network has learned, highlighting the informational content of each layer.

6. Conclusion

We presented a neurosymbolic approach that combines convolutional neural networks with random forests by training the latter on intermediate network activations. This hybrid method leverages the feature extraction capabilities of neural networks and the data efficiency of classical models.

Experiments across several image classification tasks demonstrated that random forests trained on deeper neural activations consistently outperformed those trained on raw inputs and, in low-data regimes, even rivaled the neural networks themselves. Moreover, using only a subset of activations was often sufficient, reducing training time without sacrificing accuracy.

These findings highlight that intermediate neural representations carry a significant signal and that symbolic models like random forests can effectively exploit them. Our results support the broader vision of neurosymbolic learning: combining neural and symbolic methods can offer favorable trade-offs between accuracy, interpretability, and computational efficiency.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT in order to: Improve writing style. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?, Journal of Machine Learning Research 15 (2014) 3133–3181. URL: https://jmlr.org/papers/v15/delgado14a.html.

[2] L. Grinsztajn, E. Oyallon, G. Varoquaux, Why do tree-based models still outperform deep learning on typical tabular data?, in: Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22, Curran Associates Inc., Red Hook, NY, USA, 2022.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1–9. doi:10.1109/CVPR.2015.7298594.

[4] M. Barnišin, Neurosymbolic Learning with Random Forest, Master's thesis, Masaryk University, Faculty of Informatics, Brno, 2025. URL: https://is.muni.cz/th/ja28l/.

[5] O. Boz, Extracting decision trees from trained neural networks, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '02, Association for Computing Machinery, New York, NY, USA, 2002, p. 456–461. URL: https://doi.org/10.1145/775047.775113. doi:10.1145/775047.775113.

[6] N. Vasilev, Z. Mincheva, V. Nikolov, Decision Tree Extraction Using Trained Neural Network, in: Proceedings of the 9th International Conference on Smart Cities and Green ICT Systems (SMARTGREENS), SCITEPRESS, 2020, pp. 194–200. doi:10.5220/0009351801940200.

[7] H. Li, J. Song, M. Xue, H. Zhang, J. Ye, L. Cheng, M. Song, A Survey of Neural Trees, arXiv e-prints (2022) arXiv:2209.03415. doi:10.48550/arXiv.2209.03415. arXiv:2209.03415.

[8] G.-H. Kwak, C.-w. Park, K.-d. Lee, S.-i. Na, H.-y. Ahn, N.-W. Park, Potential of Hybrid CNN-RF Model for Early Crop Mapping with Limited Input Data, Remote Sensing 13 (2021). URL: https://www.mdpi.com/2072-4292/13/9/1629. doi:10.3390/rs13091629.

[9] F. Khozeimeh, D. Sharifrazi, N. H. Izadi, et al., RF-CNN-F: Random Forest with Convolutional Neural Network Features for Coronary Artery Disease Diagnosis Based on Cardiac Magnetic Resonance, Scientific Reports 12 (2022) 1–12. doi:10.1038/s41598-022-15374-5.

[10] A. Wan, L. Dunlap, D. Ho, J. Yin, S. Lee, S. Petryk, S. A. Bargal, J. E. Gonzalez, NBDT: Neural-Backed Decision Tree, in: International Conference on Learning Representations (ICLR), 2021. URL: https://openreview.net/forum?id=mCLVeEpplNE.

[11] G. Xu, M. Liu, Z. Jiang, D. Söffker, W. Shen, Bearing Fault Diagnosis Method Based on Deep Convolutional Neural Network and Random Forest Ensemble Learning, Sensors 19 (2019) 1088. doi:10.3390/s19051088.

[12] H. Xiao, K. Rasul, R. Vollgraf, Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, 2017. URL: https://arxiv.org/abs/1708.07747. arXiv:1708.07747.

[13] G. Cohen, S. Afshar, J. Tapson, A. van Schaik, EMNIST: an extension of MNIST to handwritten letters, 2017. arXiv:1702.05373.

[14] A. Krizhevsky, Learning Multiple Layers of Features from Tiny Images, Technical Report TR-2009, University of Toronto, Toronto, Canada, 2009.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255. doi:10.1109/CVPR.2009.5206848.

[16] G. Alain, Y. Bengio, Understanding Intermediate Layers Using Linear Classifier Probes, in: Proceedings of the International Conference on Learning Representations (ICLR) Workshop Track, 2017. URL: https://arxiv.org/abs/1610.01644, originally published as arXiv:1610.01644 (2016).