PAPER • **OPEN ACCESS**

Task complexity shapes internal representations and robustness in neural networks

To cite this article: Robert Jankowski *et al* 2026 *Mach. Learn.: Sci. Technol.* **7** 025045

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
25 July 2025

REVISED
16 March 2026

ACCEPTED FOR PUBLICATION
19 March 2026

PUBLISHED
8 April 2026

PAPER

Task complexity shapes internal representations and robustness in neural networks

Robert Jankowski[1,2,5,*], **Filippo Radicchi**[3], **M Ángeles Serrano**[1,2,4], **Marián Boguñá**[1,2] and **SantoFortunato**[3]

[1] Departament de Física de la Matèria Condensada, Universitat de Barcelona, Martí i Franquès 1, Barcelona E-08028, Spain
[2] Universitat de Barcelona Institute of Complex Systems (UBICS), Barcelona, Spain
[3] Center for Complex Networks and Systems Research, Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN 47405, United States of America
[4] ICREA, Passeig Lluís Companys 23, Barcelona E-08010, Spain
[5] Current address: Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2628 CD Delft, The Netherlands
[*] Author to whom any correspondence should be addressed.

E-mail: robertjankowski.research@gmail.com and marian.serrano@ub.edu

Keywords: interpretability, network science, robustness

Abstract

Neural networks excel across a wide range of tasks, yet remain 'black boxes'. In particular, how their internal representations are shaped by the complexity of the input data and the problems they solve remains obscure. In this work, we introduce a suite of five data-agnostic probes -pruning, binarization, noise injection, sign flipping, and bipartite network randomization - to quantify how task difficulty influences the topology and robustness of representations in multilayer perceptrons (MLPs). MLPs are represented as signed, weighted bipartite graphs from a network science perspective. We contrast easy and hard classification tasks on the MNIST and Fashion-MNIST datasets. We show that binarizing weights in hard-task models collapses accuracy to chance, whereas easy-task models remain robust. We also find that pruning low-magnitude edges in binarized hard-task models reveals a sharp phase transition in performance. Moreover, moderate noise injection can enhance accuracy, resembling a stochastic-resonance effect linked to optimal sign flips of small-magnitude weights. Finally, preserving only the sign structure, instead of precise weight magnitudes, through bipartite network randomizations suffices to maintain high accuracy. These phenomena define a model- and modality-agnostic measure of task complexity: the performance gap between full-precision and binarized or shuffled neural network performance. Our findings highlight the crucial role of signed bipartite topology in learned representations and suggest practical strategies for model compression and interpretability that align with task complexity.

1. Introduction

Neural networks have achieved remarkable success across a wide range of applications and now underpin many aspects of our daily lives. However, their vast number of trainable parameters often renders them opaque 'black boxes' that, despite their effectiveness, sacrifice interpretability [1, 2]. To address this, the emerging field of mechanistic interpretability (MI) seeks to reverse-engineer the parameters and algorithms of trained networks in order to understand precisely how and why they produce their outputs [3, 4]. A common first step in MI is to decompose a network into simpler, more analyzable components. In the case of one of the simplest architectures-the multilayer perceptron (MLP)-each layer can be viewed, from a network-science perspective, as a signed, weighted bipartite graph. Such graphs are a central object of study in network science, which analyzes complex systems ranging from telecommunications and computer networks to biological and social networks [5–8]. Many real-world networks exhibit characteristic topological features-power-law degree distributions, the small-world property, community structure, and high clustering coefficient-that reflect underlying organizational principles. By treating

each MLP layer as a complex network, we can apply these same tools, such as network null models, laying the groundwork for a deeper understanding of how neural networks learn and generalize.

Neural network performance depends not only on its architecture and training procedure but also on the complexity of the tasks it must solve [9, 10]. Task difficulty shapes the representations a network learns-more challenging problems typically demand finer-grained or more abstract features [11]. For example, distinguishing between visually similar classes forces the network to encode subtle differences that are not required when classes are easily separable [12]. These differences should be observed by modeling each MLP layer as a signed bipartite graph, where positive and negative weights correspond to signed edges, and analyzing its structure. Understanding the difficulty of a task guides model selection, architecture design, and optimization strategy. Additionally, by understanding which parts of the task are more difficult to learn, we can gain insight into how the network processes information and identify potential biases or limitations.

In this work, we investigate the internal representations learned by a fully connected MLP through the lens of network science, contrasting an 'easy' task with a 'hard' task on MNIST and Fashion-MNIST datasets. Conceptually, we distinguish between two related notions of difficulty. In the image experiments on MNIST and Fashion-MNIST, we use the Structural Similarity Index Measure (SSIM) solely to select representative 'easy' and 'hard' class pairs based on visual similarity. Our central notion of task complexity, however, is probe-based: it is defined by how a trained network's performance degrades under controlled perturbations of its weights. To this end, we design five complementary experimental probes: pruning (progressively removing edges with the smallest absolute weights), binarization (reducing all weights to ± 1), noise injection (adding zero-mean noise of varying amplitude to the weights), flipping signs (changing the sign of the smallest-magnitude weights), and bipartite network randomization (shuffling connections while preserving given networks' properties). Our key findings are as follows.

- Binarizing an MLP trained on a hard task causes its accuracy to collapse to chance, whereas the easy-task model remains quite robust.
- As we prune low-magnitude edges, a binarized model trained on the hard task exhibits a sharp performance transition at a characteristic sparsity level.
- For the same model, injecting moderate noise can boost accuracy–a manifestation of stochastic resonance–while excessive noise degrades performance.
- The performance peak in the noise experiment arises from flipping the sign of the weights with the smallest absolute values.
- Randomizing the bipartite connectivity while preserving the sign of each weight leaves the network's accuracy on the easy task nearly unchanged, demonstrating that the learned representations depend more critically on the sign structure than on precise weight magnitudes.

These findings enable us to quantify task complexity in a data-agnostic manner. This means that our probes can be applied to any model and any modality as long as it contains an MLP layer. As a case study, we use these probes to evaluate the robustness of each layer in a DistilBERT model trained for named entity recognition (NER). We discover that the earliest layers are the least robust, but that simple pruning of the smallest-magnitude weights can improve their performance. In contrast, in the deeper layers, it is the sign of each weight that matters the most. Practically speaking, this means those deeper layers can be binarized at inference time without any loss in accuracy.

2. Related work

Network pruning and binary neural networks (BNNs). Model compression via pruning and quantization has been extensively explored to reduce inference cost while retaining accuracy [13–15]. Early work on optimal brain surgeon uses a second-order Taylor approximation of the training loss to identify and remove weights with minimal impact on performance [16]. First-order, data-driven methods such as Taylor pruning estimate the change in loss induced by removing individual filters or channels, achieving significant floating-point operations reductions on large convolutional neural networks with minimal retraining [17, 18]. More recently, single-shot techniques like SNIP perform connection saliency scoring at initialization-eliminating the need for any gradient-based fine-tuning to attain high sparsity levels [19]. Moreover, approaches based on spectral analysis, particularly those using tools from random matrix theory, have been applied to characterize the spectra of weight matrices and to motivate

pruning strategies [20–23]. Parallel to pruning, BNNs constrain weights and activations to $\{-1, +1\}$ to enable extreme compression and ultra-fast bitwise operations. BinaryConnect and BinaryNet introduced stochastic and deterministic binarization schemes, demonstrating that end-to-end training of 1-bit networks can achieve competitive accuracy on small benchmarks [24, 25]. Subsequent architectures, such as MeliusNet, employ dense feature propagation and learned scaling factors to close the gap to full-precision models, even on ImageNet-scale data [26]. However, these methods typically focus on worst-case accuracy drops and rarely analyze how task difficulty modulates robustness to quantization.

Similarity of neural network models. Advances in understanding how different networks or layers encode information have been driven by measures of representational and functional similarity [27]. Representational similarity measures assess how activations of intermediate layers differ, whereas functional similarity measures compare the outputs of neural networks with respect to their task. Representational similarity measures include singular vector canonical correlation analysis [28], which aligns the subspaces spanned by activations to compare layers or models, representational similarity analysis (RSA) [29], which assesses the geometry of activation patterns by correlating pairwise distance matrices and has been applied to assess the relationship between visual tasks and their task-specific models [30], and centered kernel alignment (CKA) [31], which uses kernel methods to produce robust, scale-invariant similarity scores. On the other hand, within the functional similarity measures class, we can highlight types such as: performance, hard prediction [32, 33], soft-prediction [34], or gradient-based [35] measures. In this work, since we use a simple MLP, we aim to compare representations through performance analysis and to probe the models' internals, which can be classified as a functional measure.

Task complexity. The difficulty of learning a task has been studied from both neuroscience and machine learning perspectives. Recent empirical studies show that neural networks trained on tasks with high intra-class similarity or fine-grained distinctions tend to learn deeper or more distributed feature hierarchies [11]. Additionally, Mukherjee *et al.* [36] demonstrated that the modality of the output task plays a crucial role in shaping interpretable object representations. It has also been shown that to ensure better learning outcomes, representations may need to be tailored to both task and model to align with the implicit distribution of model and task [37]. When visually assessing images, one might intuitively conclude that datasets composed of grayscale images-such as MNIST [38] or Fashion-MNIST [39]-are generally easier to classify than RGB-valued datasets like CIFAR [40]. Metrics such as SSIM [41] or learned perceptual image patch similarity [42] could serve as proxies to quantify the classification difficulty between image classes. However, in this work, we propose using neural network probes instead. These probes offer greater generalizability and can be extended beyond images to quantify task difficulty across various domains.

Network science in deep learning. Network science methods have been applied to neural network research in several ways. Custom loss functions based on graph-theoretic principles have been proposed for graph neural networks [43], and fully connected architectures have been analyzed in terms of classic centrality measures to link network structure with model performance [44]. Similarly, recurrent neural networks have been shown to exhibit universal patterns of signed motifs [45], and more generally, neural networks have been studied as dynamical systems to characterize their learning trajectories [46–48]. Network science insights have also informed the design and initialization of neural architectures. Sparse connectivity patterns inspired by scale-free graphs have been used to improve the efficiency of training large networks [49], graph-based initialization schemes have been developed to accelerate convergence [50], and random wiring schemes drawn from network models have been explored [51]. Graph-theoretic metrics-such as average shortest-path length and clustering coefficient-have been applied to characterize deep architectures, linking connectivity patterns to generalization performance [52], comparing artificial networks with biological neural circuits [53], and assessing model robustness under perturbations [54]. A recent position paper surveys many additional opportunities for network-science approaches in deep learning [55]. Despite these advances, little work has applied signed, weighted bipartite graph analysis to understand how task complexity drives emergent topological transitions in the weight space. Our work fills this gap by systematically probing MLP layers under pruning, binarization, noise injection, and randomization, revealing novel phase-transition-like behavior that depends on task difficulty.

MI. MI has been applied primarily to large-language models, where circuit-level analyses reveal functional subnetworks and token-wise attributions [56]. Extensions to Graph Transformers [57] and to bilinear MLPs [58] uncover attention-based motifs and feature-interaction circuits. However, these efforts

focus on local circuits or weight factorizations, overlooking the global connectivity patterns that a network science perspective can reveal.

3. Probing the internal representation of neural networks

Our primary benchmarks are the MNIST and Fashion-MNIST datasets. MNIST contains 70 000 28×28 grayscale images of handwritten digits (0–9), and Fashion-MNIST contains 70 000 28×28 grayscale images of Zalando clothing items across 10 categories. In this section, the SSIM [41] is used solely to select one easy and one hard binary classification problem for illustration; it does not enter our later definition of task complexity, which is based on the network's response to the probes introduced below. Initially, to define *easy* and *hard* tasks, we calculate the SSIM [41] distance, i.e. $1 - \text{SSIM}$, between all pairs of classes. The larger the SSIM distance, the greater the difference between the two images. Two identical images have zero SSIM distance. We then select the class pair with the highest SSIM distance as the easy task and the pair with the lowest SSIM distance as the hard task. On MNIST, the easiest pair is $\{0, 7\}$, and the hardest is $\{7, 9\}$. On Fashion-MNIST, the easiest pair is {`Dress`, `Pullover`}, while the hardest is {`Dress`, `Trousers`}. In figure 6 in the appendix, we show the SSIM distance heatmaps for both datasets. Let us now define the **E-model** (**H-model**) as a model trained on an easy (hard) task.

The input grayscale images are flattened, and we begin by training two MLPs with ReLU activation, each with a single hidden layer of dimension $d = 64$, on binary classification tasks of differing difficulty. Optimization is performed using the Adam optimizer together with a cosine-annealing learning-rate schedule with a maximum of 10 epochs. At each step, we measure test accuracy and stop training when it is maximized. For completeness, in the appendix, we report experiments using hidden-layer sizes $d = 32$ (figures 10 and 11) and $d = 128$ (figures 12 and 13) and a comparison across hidden-layer sizes in figure 14, showing quantitatively similar trends. We also report experiments using `tanh` activation in figure 15, and `hardsigmoid` in figure 16, which show qualitatively similar results.

All of our probing methods are applied to each trained network *without* any further fine-tuning or retraining. In the following sections, we provide a detailed description of each probe. The code for reproducing experiments is available at https://github.com/robertjankowski/probing-neural-networks.

3.1. Pruning and binarizing

There are many methods for pruning neural networks. In this work, we use a simple post-training strategy-iteratively removing the weights with the smallest absolute values, as in the Lottery Ticket Hypothesis [59]. After each pruning step, we measure test accuracy on both easy and hard tasks. We also evaluate binarized versions of these pruned models, where each remaining weight is replaced by its sign, and refer to them as the signed-E and signed-H models, respectively.

Figure 1(a) shows that the E-model retains higher accuracy as the smallest-magnitude weights are removed, whereas the H-model's performance drops much more rapidly. Interestingly, pruning can even increase the accuracy of the signed-E model. Most strikingly, the signed-H model, which initially performs at near random, exhibits a performance transition and even surpasses the H-model's accuracy for some sparsity levels. A similar behavior is present for the Fashion-MNIST dataset (see figure 1(b)).

Although the test accuracies of the E-model and the H-model are high, their internal representations differ. One could argue that this distinction can be measured through the distribution of weights. However, as shown in figure 7, the standard deviation of the weights, $\sigma(w)$, depends on the dataset. For MNIST, $\sigma(w)$ is narrower and smaller for the easy task, whereas it is wider and larger for Fashion-MNIST. Hence, weight statistics alone cannot serve as a reliable measure of task complexity.

In addition, in figure 17 in the appendix, we compare different pruning strategies. One can observe that removing only the weights with the smallest absolute values yields a performance increase for the signed-E and signed-H models. Neither pruning the largest-magnitude weights nor randomly pruning weights exhibits this phenomenon. We also compare the element-wise pruning to the spectral one (see appendix A.7). Typically, only a few of the top-k singular values are important for maintaining high test accuracy. Interestingly, for the signed-H-model, we observe that retaining only 2-3 singular values improves test accuracy (see figure 19).

3.2. Noise injection

As an additional probe, we inject noise into the network weights. Specifically, we perturb each weight w by adding a random variable drawn from the uniform distribution, $U(-a, a)$, where a controls the noise level.

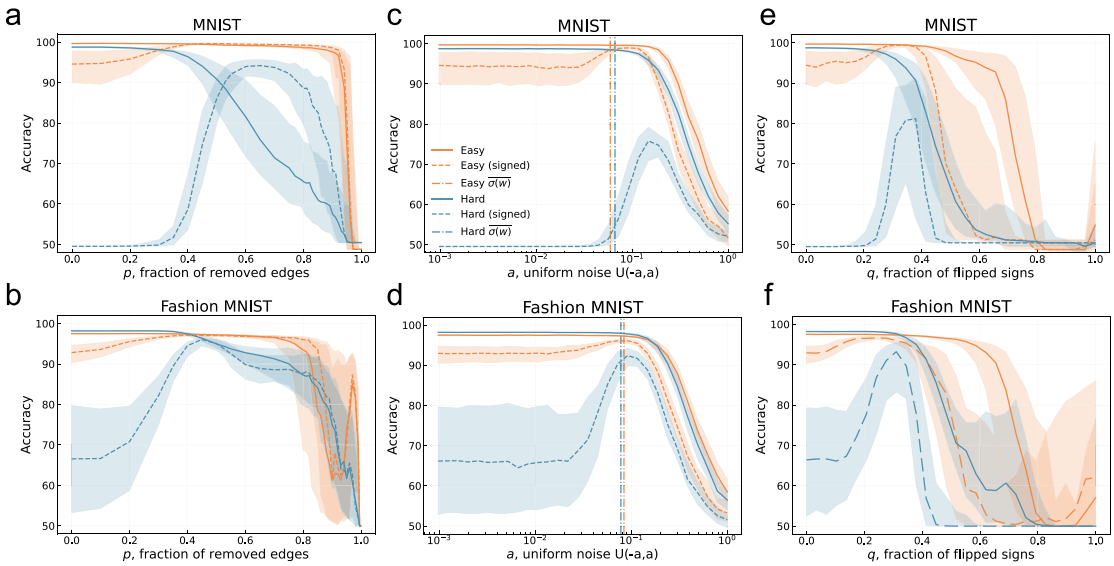

Figure 1. (a) and (b) Pruning experiment. The test accuracy as a function of the fraction of removed edges. (c) and (d) Noise injection experiment. The test accuracy as a function of the uniform noise level injected into the weights. The vertical lines show the average standard deviation of the weights. (e) and (f) Sign flipping experiment. The test accuracy as a function of the fraction of the smallest-magnitude sign flipped. All curves are averaged over 100 random initializations. Shaded regions denote the interquartile range (IQR), and the solid lines represent the median.

For each noise magnitude, we evaluate test accuracy on both the E- and H-models, as well as their binarized ('signed') counterparts. For the signed models, we first inject noise and then binarize the weights. Figures 1(c) and (d) show that the E-model remains substantially more robust under noise injection than the H-model. Moreover, adding a moderate amount of noise to the signed-E and signed-H models can actually improve their accuracy-a phenomenon akin to stochastic resonance [60, 61], which has been documented in a wide range of systems, including bistable ring lasers, semiconductor devices, chemical reactions, and climate dynamics [62–64].

In our context, this stochastic-resonance-like effect appears in the accuracy curves. When the noise standard deviation is much smaller than the average weight standard deviation $\bar{\sigma}(w)$ (indicated by the vertical dotted lines), we observe no improvement in model performance. Conversely, when the noise level significantly exceeds $\bar{\sigma}(w)$, accuracy degrades to near-random levels. Thus, there exists an optimal noise-level region that maximizes performance.

In addition, instead of additive noise, we also tested the impact of the multiplicative noise applied to a fraction of the smallest-magnitude weights (see appendix A.6), finding that signed models remain comparatively robust and can even slightly improve for moderate perturbations when only weak-to-intermediate weights are affected (see figure 18).

3.3. Flipping signs

To further investigate the stochastic resonance-like effect observed in the accuracy curves, we design a simple experiment in which we flip the signs of the smallest-magnitude weights. First, we sort all weights by their absolute values and then flip the sign of a fraction q of the smallest-magnitude weights. In figures 1(e) and (f), we plot the test accuracy as a function of q for the original models and their binarized counterparts. Consistent with our noise-injection findings, flipping a nonzero fraction of the smallest-magnitude weights in both the signed-E and signed-H models improves performance, yielding a higher-accuracy peak. These results indicate that the signs of the weights, rather than their exact values, are most critical to model performance. To test this hypothesis, we next apply a series of bipartite network randomizations.

3.4. Bipartite network randomization

Each MLP layer can be represented as a signed, weighted bipartite graph. The graph comprises two disjoint node sets-left L (inputs) and right R (outputs)-with edges only running between L and R. A forward signal propagates from L to R. In the unweighted case, each node $i \in L \cup R$ has two degree counts: k_i^+ (the number of positive-weight edges) and k_i^- (the number of negative-weight edges). In the weighted formulation, these become strengths-$s_i^+ = \sum_{j:w_{ij}>0} w_{ij}$ and $s_i^- = \sum_{j:w_{ij}<0} |w_{ij}|$-summing the magnitudes of the positive or negative edges incident on i. Finally, we denote the degree (or strength)

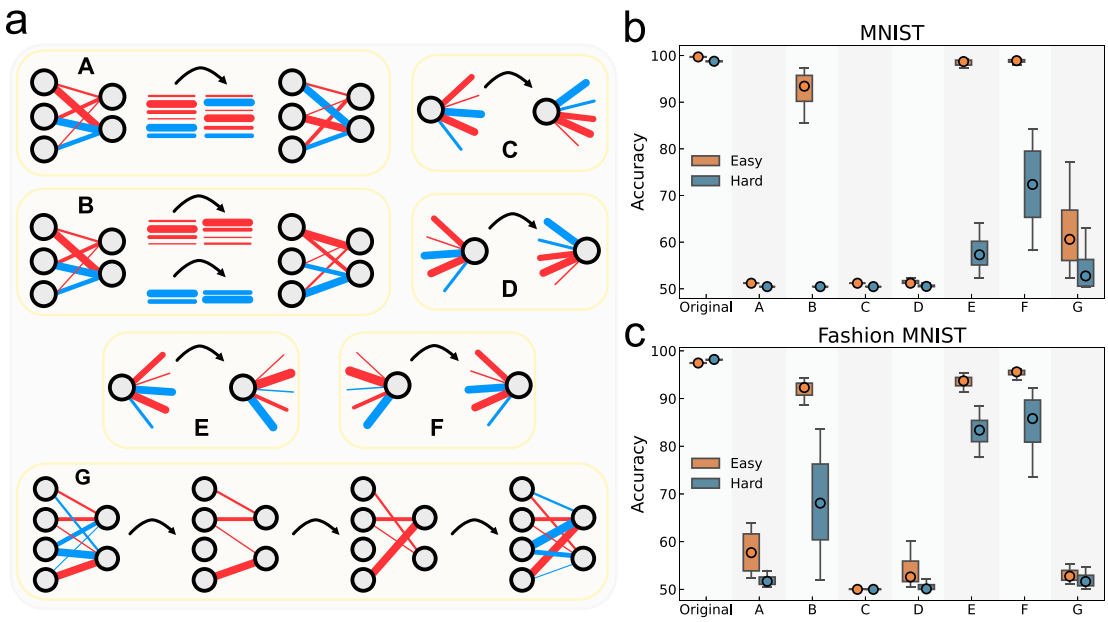

Figure 2. (a) Visualization of the seven types of bipartite randomizations. The accuracy of the neural network after applying each type of bipartite randomization for (b) MNIST and (c) Fashion MNIST. Boxplots show the distribution of test accuracies across 100 independent network trainings, whereas scatter markers denote the median accuracy.

Table 1. Types of bipartite randomizations and properties preserved after randomization where α is a fraction of the edges with a positive sign. A *checkmark* in the *Keeps original sign* column indicates that each edge retains its original positive or negative sign under that randomization.

Type	α	k_L^-	k_L^+	s_L^-	s_L^+	k_R^-	k_R^+	s_R^-	s_R^+	Keeps original sign
A	✓	—	—	—	—	—	—	—	—	—
B	✓	✓	✓	—	—	✓	✓	—	—	✓
C	✓	✓	✓	✓	✓	—	—	—	-	—
D	✓	—	—	—	—	✓	✓	✓	✓	—
E	✓	✓	✓	✓	✓	✓	✓	—	—	✓
F	✓	✓	✓	—	—	✓	✓	✓	✓	✓
G	✓	✓	✓	—	—	✓	✓	—	—	—

distributions over all positive and negative edges by $P(k^+)$ and $P(k^-)$ (or $P(s^+)$ and $P(s^-)$ in the weighted case).

We introduce seven distinct randomization strategies, each of which preserves different structural properties of these bipartite graphs. Figure 2(a) illustrates these methods, and table 1 summarizes their key characteristics. Moreover, we provide a pseudocode for each randomization in appendix A.9.

Using these randomization strategies, we evaluate the post-randomization accuracy of our trained MLPs. Figure 2(b) presents accuracy boxplots for the E- and H-models. We see that only those strategies that (1) preserve both the positive and negative degree distributions $P(k^+)$ and $P(k^-)$ and (2) retain each edge's original sign, maintain high performance. If either of these properties is altered, accuracy falls to chance. Notably, both randomizations B and G keep the degree distributions fixed, but only B preserves accuracy-demonstrating that the specific arrangement of positive versus negative weights is itself critical.

We further evaluate the randomization strategies that preserve high accuracy in the pruning experiment. As in section 3.1, we measure both the original and randomized models' performance at each fraction of removed edges. Figure 3 shows that randomization initially causes an accuracy drop for both the E- and H-models. However, as sparsity increases, the randomized E-model's accuracy steadily recovers and ultimately matches that of the original E-model. By contrast, the randomized H-model undergoes an abrupt transition-similar to the signed-H-model. These results confirm that preserving the learned edge signs, rather than the precise weight values, is essential for maintaining high performance under heavy pruning.

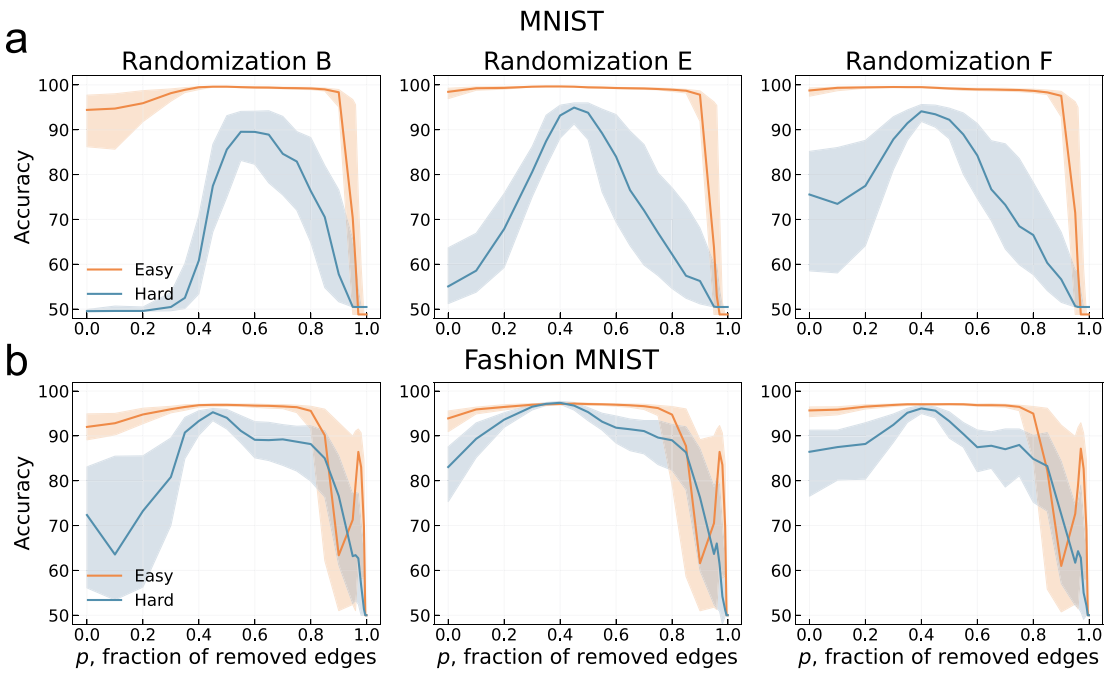

Figure 3. The test accuracy as a function of the fraction of removed edges after applying bipartite randomization B, E, and F for (a) MNIST and (b) Fashion-MNIST. All curves are averaged over 100 random initializations. Shaded regions denote the interquartile range (IQR), and the solid lines represent the median.

4. Defining task complexity

So far, we have compared models trained on easy and hard tasks. Binarizing or randomizing the H-model causes a large drop in accuracy, whereas the E-model's accuracy declines much less. This suggests a link between task difficulty and post-binarization (or randomization) performance. We therefore quantify task difficulty by measuring, for each image class pair in the MNIST dataset, the change in accuracy before versus after binarizing or randomizing.

We first note that the test accuracy for each digit class exceeds 98% (see figure 8). Next, we evaluate how much accuracy changes once we apply our probes. In figure 4(a), we plot the difference in accuracy between each original model and its signed version. A smaller gap means the signed model's performance remains close to the original, whereas a larger gap indicates a harder classification task. For example, digits 0 and 3 show very little change-these are easy to distinguish-while digits 1 and 7 fall to around 50% accuracy after binarization, producing a substantial drop compared to the original. Applying bipartite randomization yields similar patterns (figure 4(b)): the harder the digits are to classify, the greater the loss in accuracy. We further quantify this relation in figure 4(c). The Spearman correlation between the changes in accuracy is very high, with a coefficient of $\rho = 0.96$.

Initially, we defined *easy* and *hard* tasks using the SSIM, which quantifies visual similarity between image pairs. As shown in figure 4, SSIM-easy task (0 vs 7) exhibits only a small drop in accuracy, whereas SSIM-hard task (7 vs 9) suffers accuracy losses approaching 50%. However, SSIM requires image data. On the other hand, our approach is data-agnostic. This means that our probes can be applied to any model and any data modality, as long as it contains MLP components.

5. Measuring layer robustness in a language model

We emphasize that, unlike the MNIST and Fashion-MNIST experiments, we do not vary task complexity in this section; instead, we use a fixed downstream task (NER) to demonstrate that the same probes yield informative, layer-wise robustness profiles in a transformer. In this case study, we evaluated the robustness of individual layers in a pretrained DistilBERT model[5] [65], fine-tuned on the CoNLL-2003

[5] https://huggingface.co/dslim/distilbert-NER.

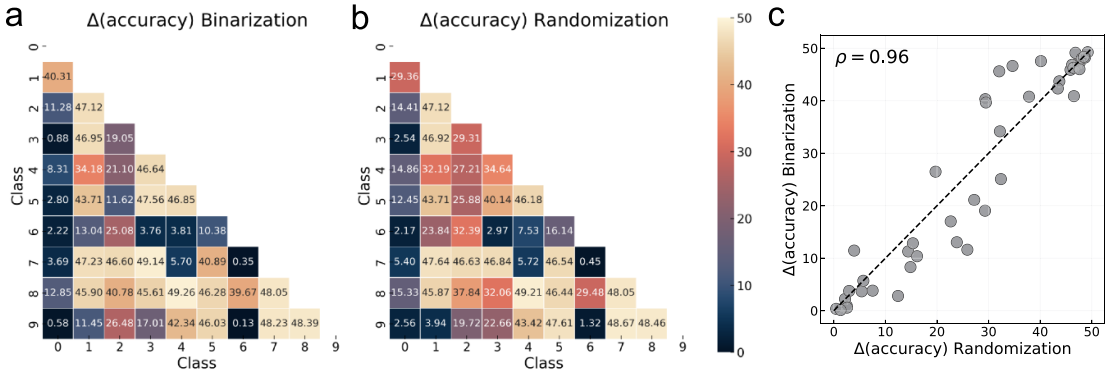

Figure 4. Difference in accuracy of the neural network for two-class discrimination under two modifications: (a) weight binarization, (b) application of bipartite randomization B. Each entry indicates the change in accuracy introduced by the modification, averaged over 10 realizations. (c) Scatter plot of the change in accuracy under randomization versus binarization. Each point represents a digit-class pair. In the top left corner, the Spearman correlation coefficient is reported.

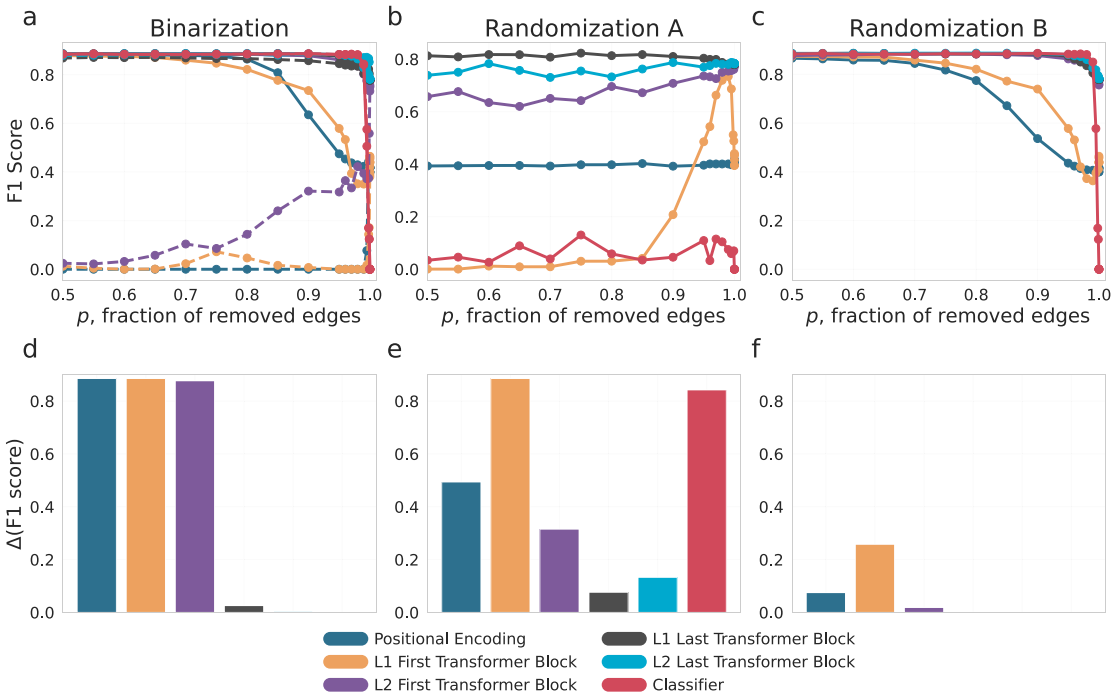

Figure 5. Case study on the DistilBERT model. $F1$ score as a function of the proportion of removed edges: (a) weight-binarization experiment-solid line: original weights; dotted line: binarized weights, (b) after applying randomization A, (c) after applying randomization B. In the bottom panel, for $p = 0$ (no edges removed), we plot the $F1$-score differences under (d) weight binarization, (e) randomization A, and (f) randomization B. Each color corresponds to a different probed layer of DistilBERT, evaluated on the NER task.

NER dataset [66]. DistilBERT, a distilled variant of bidirectional encoder representations from transformers (BERT) [67], contains approximately 65 million parameters. We focus on the NER task, which aims to identify and categorize entities within text. We apply our diagnostic probes independently to six layers of the model: (1) the positional embedding layer, (2) the first linear layer of the first transformer block, (3) the second linear layer of the first transformer block, (4) the first linear layer of the final transformer block, (5) the second linear layer of the final transformer block, and (6) the token-classification (output) layer. For each probe, we report the test $F1$-score on the NER task.

In figures 5(a)–(c), we plot the $F1$ score as a function of the fraction of removed edges for six transformer layers. First, consider the binarization experiment (figure 5(a)). The earliest layers suffer a rapid decline in $F1$ score as the edges are removed, with the signed model's predictions becoming nearly indistinguishable from random. In contrast, the deepest layers remain remarkably robust: even under extreme edge removal, their binarized counterparts sustain high $F1$ scores. To make this comparison explicit,

figure 5(d) shows the difference between the original and binarized $F1$ curves for each layer. This pattern aligns with the findings of Bai *et al* [68], who demonstrated that shallower transformer layers are more susceptible to quantization errors than deeper ones.

Next, we examine two bipartite randomization schemes, A and B. Under randomization A (figure 5(b)), which, in isolation, previously degraded accuracy to chance, the model still retains reasonable performance when edges are removed. We attribute this resilience to the residual connections in each linear sublayer, which effectively bypass the randomized weights. Randomization B (figure 5(c)) has virtually no impact on the $F1$ score in the later layers, underscoring the inherent robustness of these models.

Finally, by comparing the performance drops induced by binarization versus those induced by randomization B (figures 5(d) and (f)), we see that binarization causes a substantially larger performance drop in the early layers.

This might be unsurprising: perturbations at the network's input propagate through all subsequent layers, amplifying their effect on overall performance. However, the effect could be more complex and additional tools might be needed to quantify the dumping and amplification effect, which we discuss in appendix A.8. Randomization, by contrast, produces a more modest decline. Yet, it follows the same relative layer-wise pattern, with early layers more affected than later ones.

The same overall trends persist in the noise-injection and sign-flip experiments (figure 9). However, under sign flipping, the positional-encoding layer's performance curve becomes non-monotonic. We believe this arises from the interplay of residual connections and LayerNorm, which together render the network invariant to a global sign inversion. Specifically, when $q = 1$, we invert the entire positional-encoding vector, as the model contains six transformer blocks-an even number-each successive sign inversion is counteracted by the next, so that by the time the representation reaches the final classification head, the original encoding is effectively restored.

6. Conclusions

Understanding task complexity is essential for designing robust neural networks, guiding model selection, and improving training strategies. In this work, we investigated the internal representations of MLP layers by contrasting models trained on *easy* and *hard* tasks using five complementary probes: pruning, binarization, noise injection, sign flipping, and bipartite network randomization.

Our results show that task complexity fundamentally shapes the robustness of learned representations to perturbations. In particular, binarizing a model trained on a hard task leads to a marked loss of accuracy, whereas an easy-task model remains comparatively robust. Likewise, pruning the binarized hard-task model produces a sharp performance transition that is absent in the easy-task case. We also found that adding noise to binarized models can improve accuracy, consistent with a stochastic-resonance-like effect. Importantly, this performance gain from flipping a fraction of the weakest weights should not be interpreted as evidence that these individual weights are themselves the most critical to model function. Rather, the effect suggests that perturbing small weights can suppress noisy local sign assignments. More broadly, our results indicate that model performance depends less on the precise magnitudes of individual weights than on the overall organization of positive and negative connections. This interpretation is further supported by the bipartite network randomization experiments. In easier tasks, high accuracy is maintained only when randomization preserves both the positive/negative degree distributions and the original edge signs, confirming that the sign structure of the learned connectivity is more important than the exact floating-point weight values. These probes therefore suggest a data-agnostic way to quantify task difficulty, as the magnitude of the accuracy loss after binarization or randomization correlates with the hardness of the task or the separability of its classes.

Applying the same probes to DistilBERT on a NER task revealed a different layerwise pattern: early layers are less robust to binarization, randomization, and pruning than later layers. The greater resilience of later layers, even under randomization, may be partly explained by the presence of residual connections.

Overall, our study shows that task complexity leaves a clear signature in the robustness of learned representations and highlights the central role of sign structure and connectivity in neural network function. From a practical perspective, layers whose performance is governed primarily by sign structure may be promising candidates for binarization at inference time. Future work could explore how these probe-based robustness measures relate to representational similarity metrics such as CKA [31] or RSA [27].

Acknowledgments

We acknowledge the support of the AccelNet-MultiNet program, a project of the National Science Foundation (Awards #1927425 and #1927418). R J acknowledges support from the fellowship FI-SDUR funded by Generalitat de Catalunya. F R acknowledges support from the Air Force Office of Scientific Research (Grant No. FA9550-24-1-0039). M Á S and M B acknowledge support from Grant No. TED2021-129791B-I00 funded by MCIN/AEI/10.13039/501100011033 and by 'European Union NextGenerationEU/PRTR', and Grant No. PID2022-137505NB-C22 funded by MCIN/AEI/10.13039/501100011033 and by ERDF/EU.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/robertjankowski/probing-neural-networks [69].

Appendix

A.1. Structural similarity index distance between pairs of classes

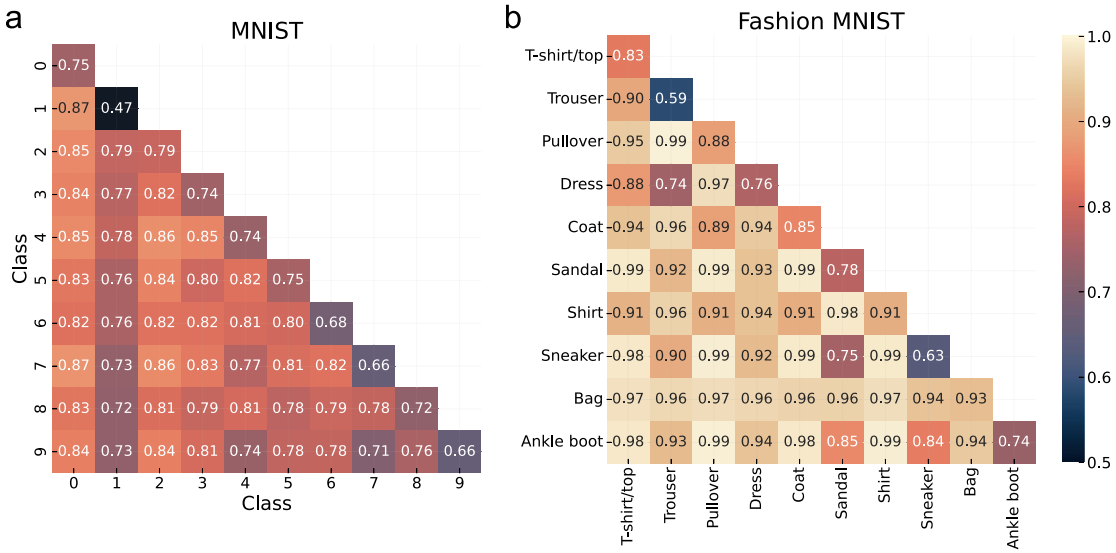

Figure 6. The Structural Similarity Index Measure (SSIM) distance between all pairs of classes for (a) MNIST and (b) Fashion MNIST. The lower the value, the more similar the two pairs of classes are.

A.2. Standard deviations of learned weights

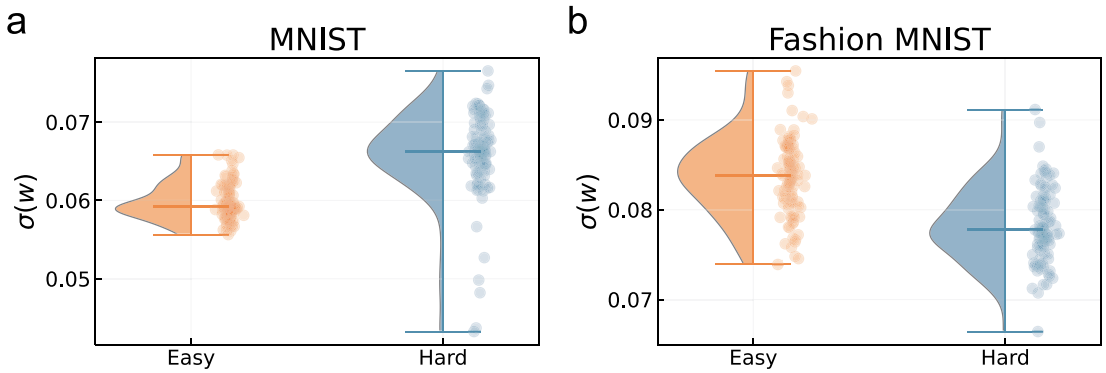

Figure 7. The distribution of weight standard deviations for (a) MNIST and (b) Fashion MNIST. Each point corresponds to a single trained neural network.

A.3. Accuracy heatmap for MNIST

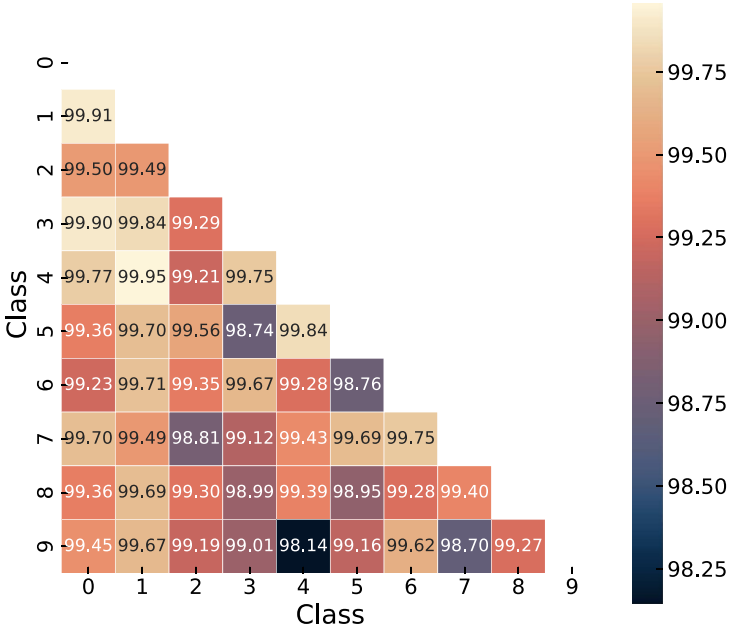

Figure 8. The accuracy heatmap for MNIST. Each entry shows the accuracy of the neural network trained to distinguish between two classes. The results are averaged over 10 realizations.

A.4. Additional experiments for DistilBERT

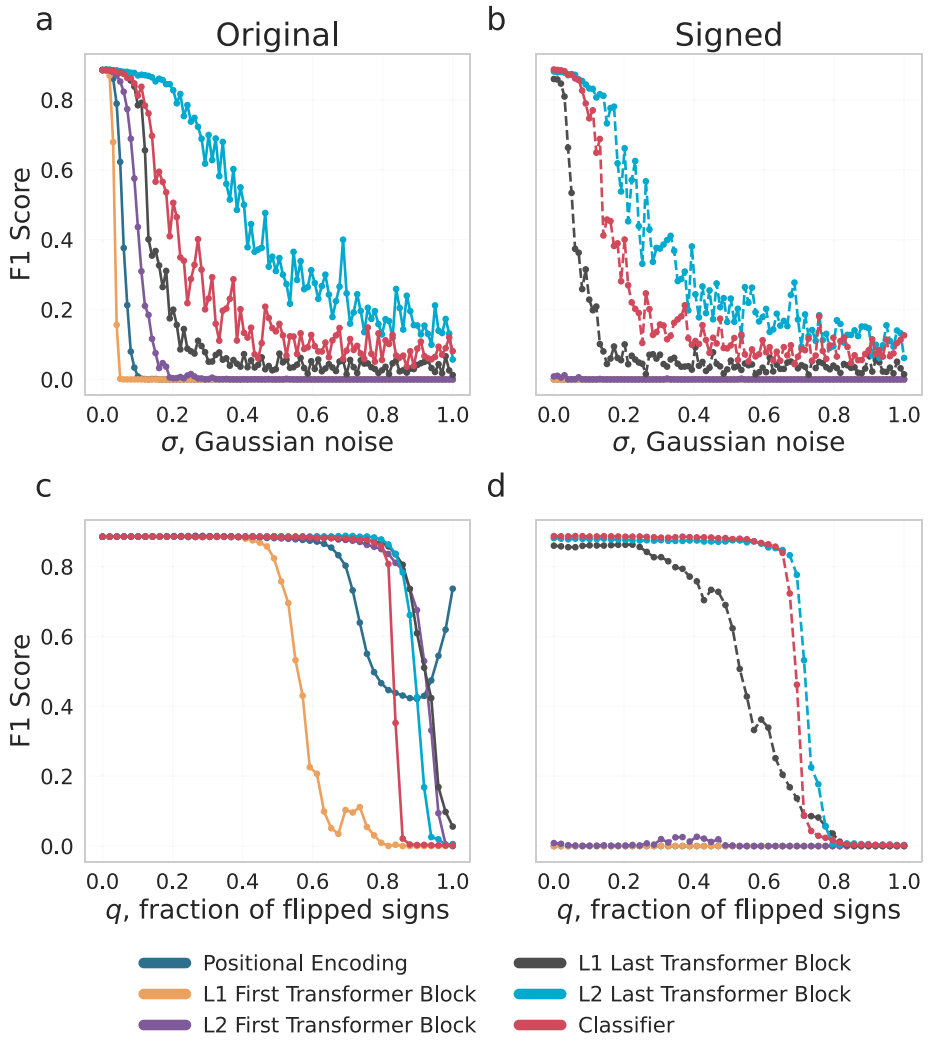

Figure 9. (a) and (b) The noise injection experiment. The $F1$ score as a function of the Gaussian noise injected into the weights. (c) and (d) Sign flip experiment. The $F1$ score as a function of the fraction of the smallest-magnitude sign flipped. Each color corresponds to a different probed layer of DistilBERT, evaluated on the NER task.

A.5. Additional experiments for MNIST and Fashion-MNIST datasets

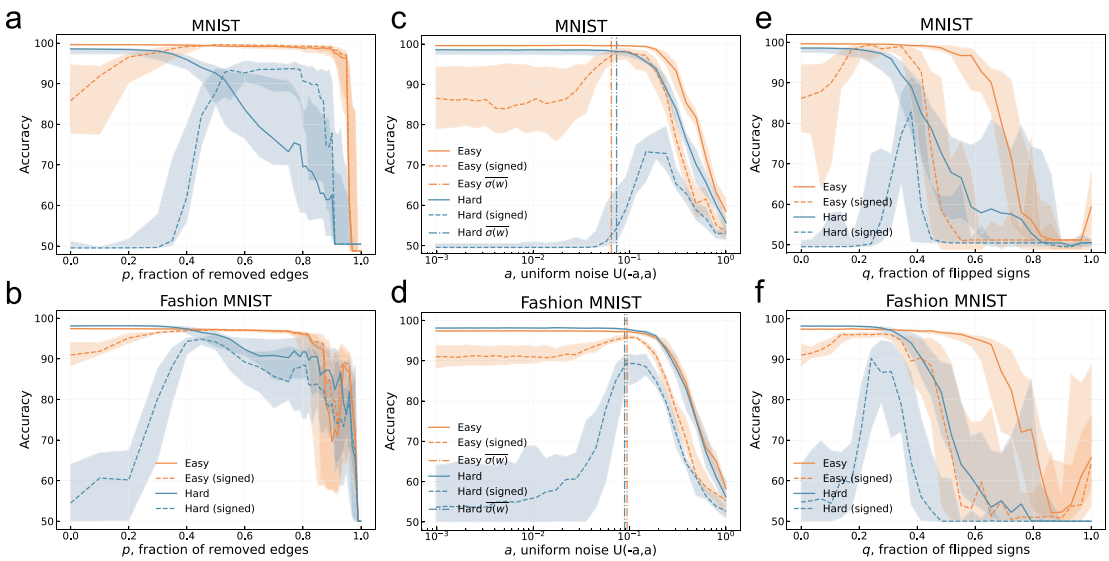

Figure 10. (a) and (b) Pruning experiment. The test accuracy as a function of the fraction of removed edges. (c) and (d) Noise injection experiment. The test accuracy as a function of the uniform noise level injected into the weights. The vertical lines show the average standard deviation of the weights. (e) and (f) Sign flipping experiment. The test accuracy as a function of the fraction of the smallest-magnitude sign flipped. All curves are averaged over 20 random initializations **for hidden layer size** $d = 32$. Shaded regions denote the interquartile range (IQR), and the solid lines represent the median.

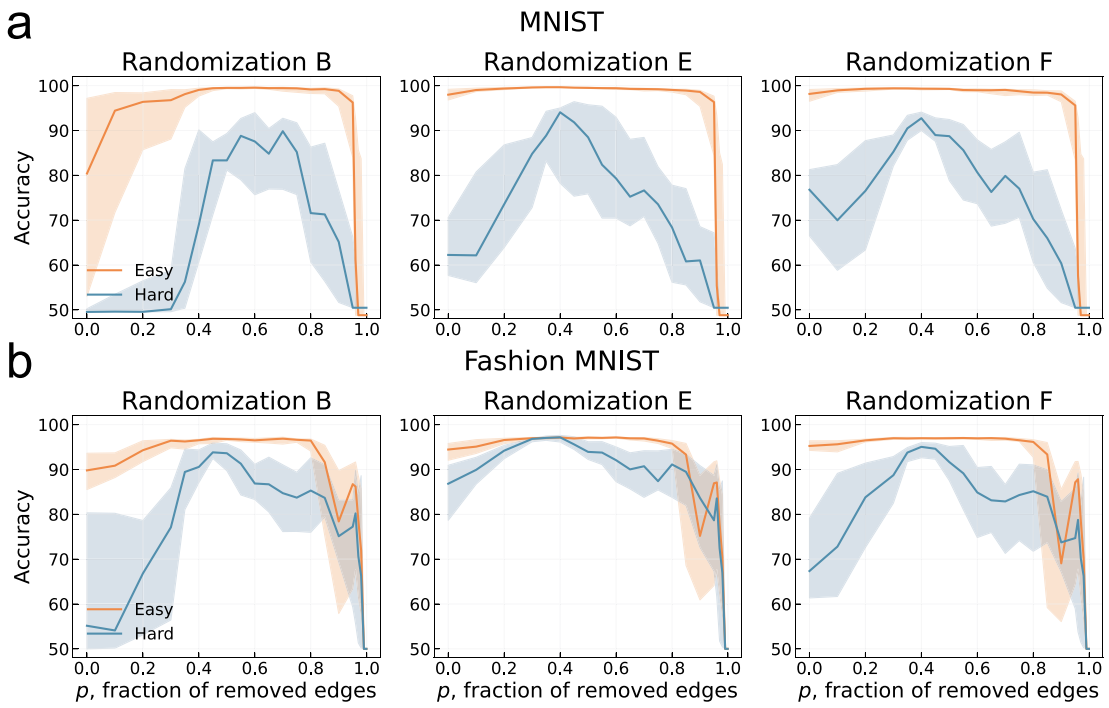

Figure 11. The test accuracy as a function of the fraction of removed edges after applying bipartite randomization B, E, and F for (a) MNIST and (b) Fashion-MNIST **for hidden layer size** $d = 32$. All curves are averaged over 100 random initializations. Shaded regions denote the interquartile range (IQR), and the solid lines represent the median.

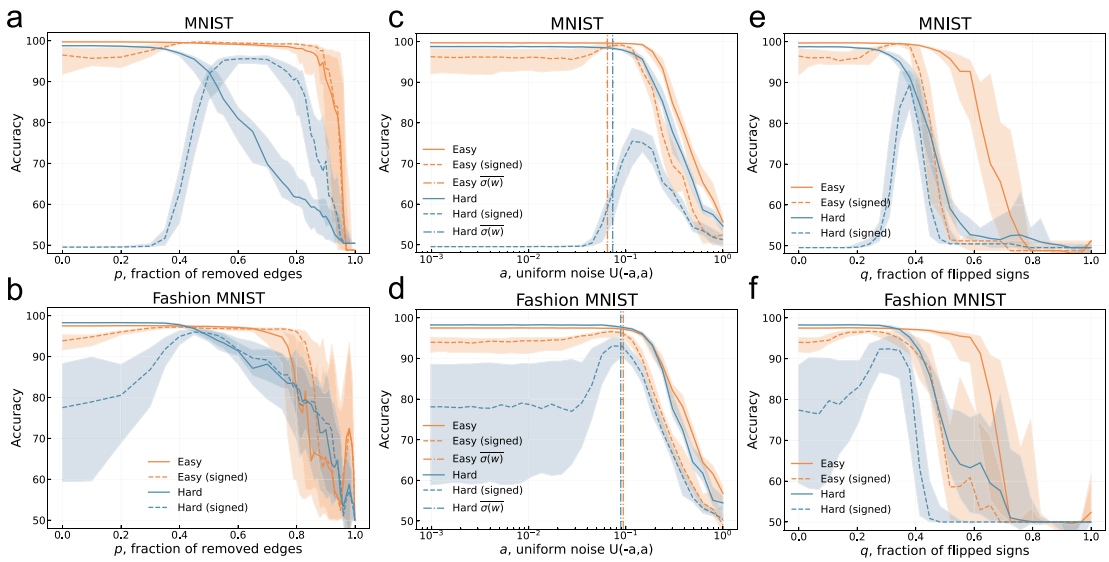

Figure 12. (a) and (b) Pruning experiment. The test accuracy as a function of the fraction of removed edges. (c) and (d) Noise injection experiment. The test accuracy as a function of the uniform noise level injected into the weights. The vertical lines show the average standard deviation of the weights. (e) and (f) Sign flipping experiment. The test accuracy as a function of the fraction of the smallest-magnitude sign flipped. All curves are averaged over 20 random initializations **for hidden layer size** $d = 128$. Shaded regions denote the interquartile range (IQR), and the solid lines represent the median.

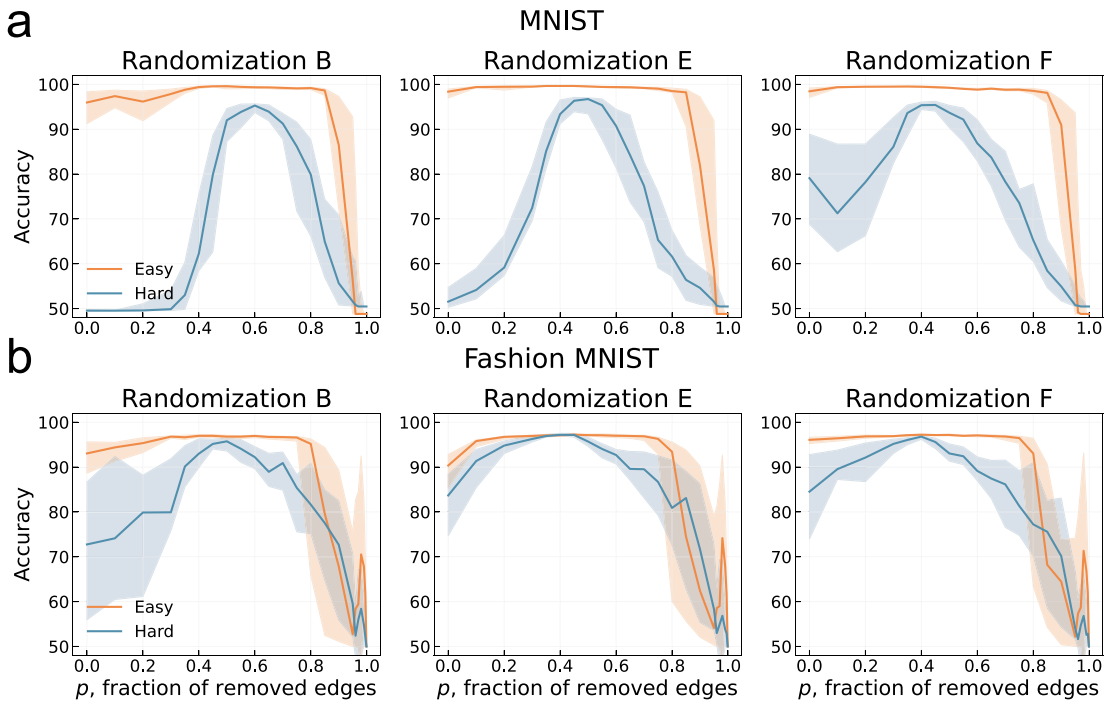

Figure 13. The test accuracy as a function of the fraction of removed edges after applying bipartite randomization B, E, and F for (a) MNIST and (b) Fashion-MNIST **for hidden layer size** $d = 128$. All curves are averaged over 100 random initializations. Shaded regions denote the interquartile range (IQR), and the solid lines represent the median.

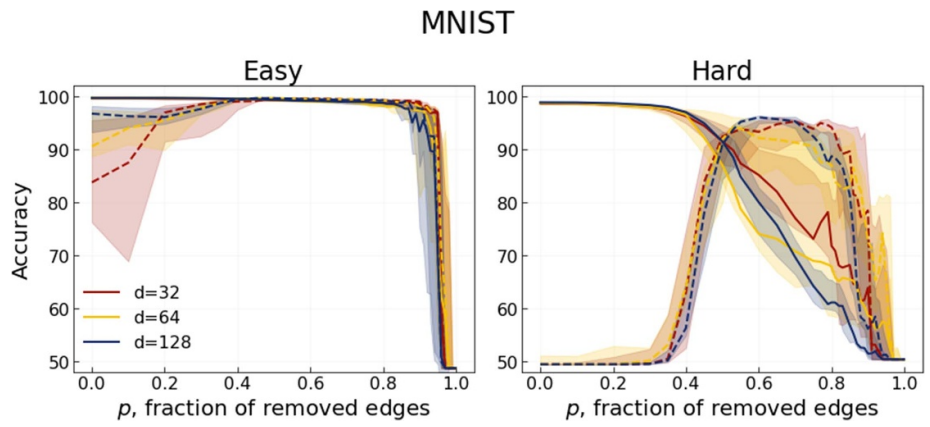

Figure 14. Comparison of different hidden dimensions in the pruning experiment for the MNIST dataset.

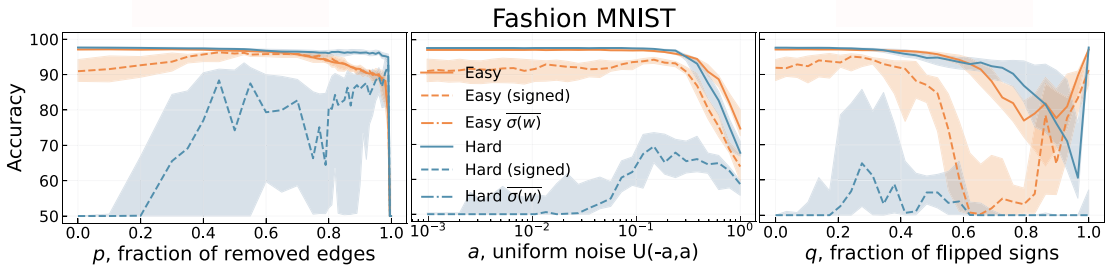

Figure 15. Experiments on Fashion-MNIST with `tanh` activation function.

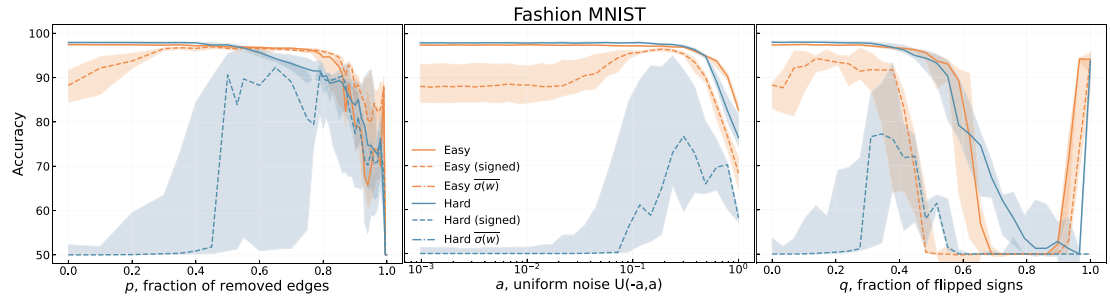

Figure 16. Experiments on Fashion-MNIST with `hardsigmoid` activation function.

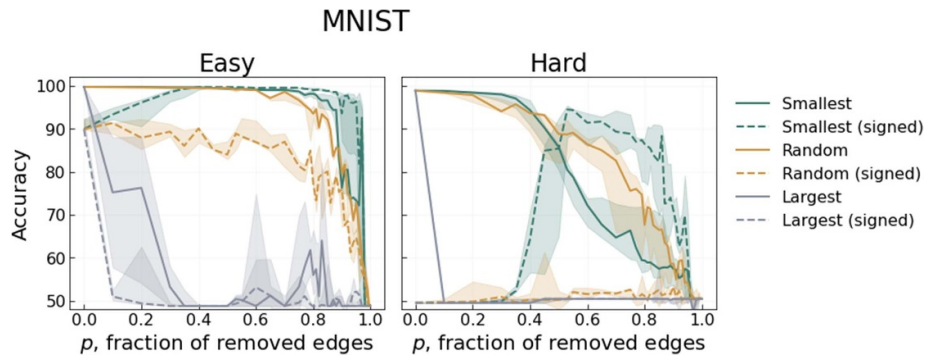

Figure 17. Pruning experiment. We apply three strategies: Smallest (removing edges with the smallest magnitude), Random (removing edges at random), and Largest (removing edges with the largest magnitude).

A.6. Multiplicative noise

Multiplicative noise is applied only to a selected fraction of the smallest-magnitude weights. Specifically, we first choose a fraction r of the smallest weights, and then perturb only those weights according to

$$w \leftarrow w \cdot x, \qquad x \sim U(-a, a), \tag{1}$$

where a controls the noise amplitude. We then sweep over different values of a and measure the test accuracy. The results are shown in figure 18 for three values of r. For $r = 0.3$, where only the smallest 30% of weights are perturbed, the test accuracy of the standard models on both the Easy and Hard tasks decreases as the noise magnitude increases. By contrast, the signed model on the Easy task shows a slight improvement and remains stable even for larger values of a. For $r = 0.7$, we observe a similar trend for the standard Easy and Hard models. However, in this regime, the signed model on the Hard task also improves, reaching a broad plateau. This indicates that signed models can tolerate, and in some cases benefit from, perturbations affecting a substantial fraction of weak and intermediate weights. For $r = 0.9$, where all but the largest 10% of weights are perturbed, the accuracy of all models decreases rapidly. In this case, the perturbation is no longer confined to clearly negligible weights and begins to affect increasingly important connections, which leads to a substantial degradation in performance.

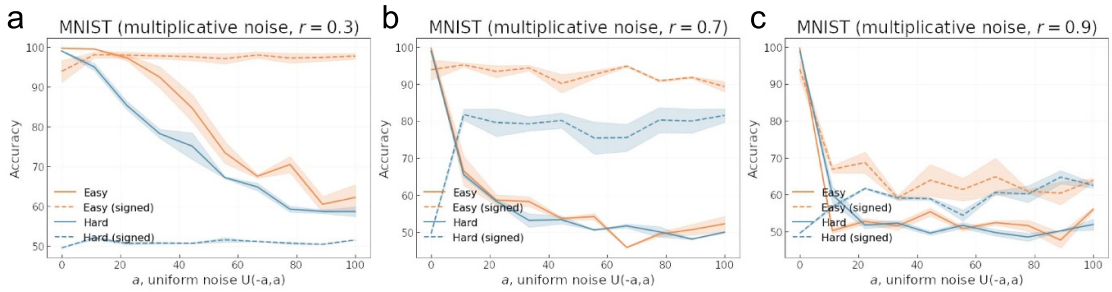

Figure 18. Multiplicative noise applied to a fraction of the smallest weights for the MNIST dataset. Panels (a)–(c) show test accuracy as a function of the noise magnitude a for three values of r, where r denotes the fraction of the smallest-magnitude weights that are perturbed.

A.7. Spectral pruning

For each weight matrix, we compute its singular value decomposition and construct a low-rank approximation by retaining only the top-k singular values. We then sweep k from the full rank down to $k=1$ and measure the test accuracy at each step. Figure 19 reports test accuracy as a function of the number of retained singular values for the MNIST dataset. For the Easy task, retaining only two singular values suffices to maintain high performance, whereas the Hard task requires more singular values. Interestingly, for the signed model on the Hard task, we observe a performance peak: keeping only $k=2$ or $k=3$ singular values improves test accuracy relative to higher ranks. This suggests that, in this variant, task-relevant information is concentrated in a small number of dominant singular modes, while intermediate spectral components become detrimental after binarization. A similar non-monotonic effect is observed with our element-wise pruning strategy that removes the smallest-magnitude weights (figure 1(a) and (b)), indicating that the improvement is not tied to a specific pruning basis but rather reflects broader redundancy in the representation. In contrast, for the Easy task, the signed model degrades gradually under spectral pruning, whereas under element-wise pruning, it shows a slight improvement.

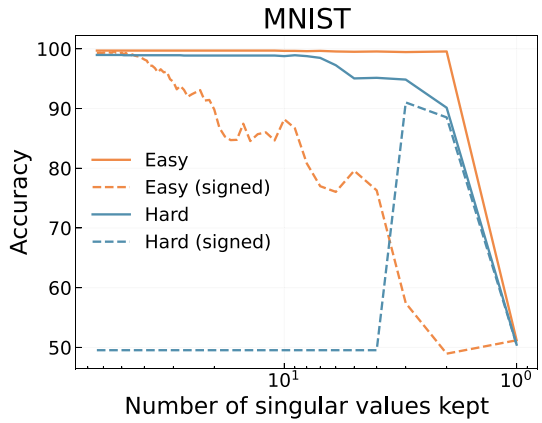

Figure 19. Spectral pruning on MNIST: test accuracy versus truncation rank k (number of singular values retained). For each layer, the weight matrix is replaced by its rank-k truncated SVD approximation, sweeping from the full-rank model down to $k=1$.

A.8. CKA similarity of the transformer layers under perturbation

Depending on the magnitude of the perturbation and other architectural factors, a signal can either be amplified or damped as it propagates forward. In our empirical analysis, we initially used the $F1$ score to quantify this effect and observed that the score deteriorates more rapidly when perturbations are applied to early layers. To study this phenomenon more directly, we propose measuring how perturbations alter the activations themselves. To this end, we use CKA, a metric commonly used to assess similarity between neural network layers [31].

In figure 20, we report the CKA similarity of the transformer layers under magnitude-based pruning of individual layers. In panel (a), for example, we perturb only the positional-encoding layer and measure how the activations of all downstream layers change. For pruning rates up to $p=0.4$, CKA remains high across all layers, indicating that the perturbation has little effect on the activations. This is consistent with our results in figure 5(a), where the $F1$ score also remains high for large values of p. At higher pruning rates, CKA similarity decreases, and different layers exhibit distinct sensitivity profiles. Interestingly, the first layer of the first transformer block maintains higher CKA similarity than the perturbed positional-encoding layer itself, suggesting that its activations are comparatively robust to perturbations in earlier components.

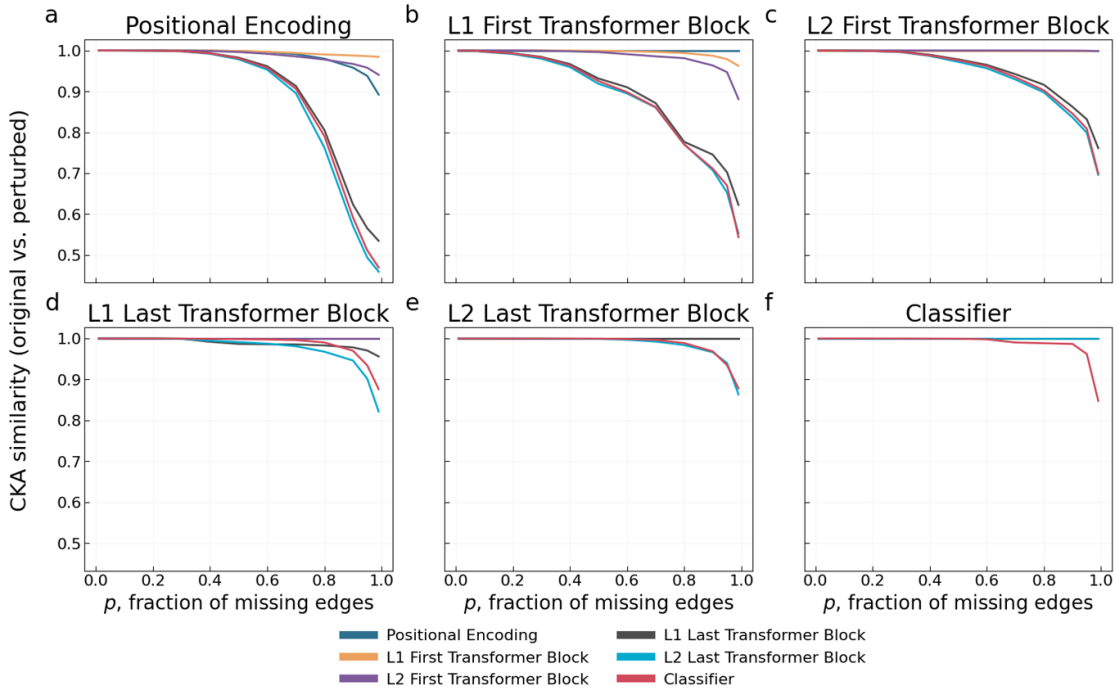

Figure 20. CKA similarity of the transformer layers under magnitude-based pruning of individual layers. Each panel (a)–(f) shows the linear CKA similarity between original and perturbed activations as a function of the pruning rate p (fraction of edges removed) when pruning is applied only to the layer named in the panel title. Within each panel, colored curves show CKA for the activations of the layers indicated in the legend.

A.9. Bipartite randomization algorithms

Algorithm 1. Type A randomization.

Require: Bipartite edge list $E = \{(u, v, w_{uv})\}$

1: Separate edges into $E_0 = \{w = 0\}$ and $E_{\neq 0} = \{w \neq 0\}$
2: Extract all nonzero weights $W = \{w : (u, v, w) \in E_{\neq 0}\}$
3: Randomly permute W
4: Reassign permuted weights back to edges in $E_{\neq 0}$ (same positions)
5: **return** $E' = E_{\neq 0} \cup E_0$

Algorithm 2. Type B randomization.

Require: Bipartite edge list $E = \{(u, v, w_{uv})\}$

1: $E^+ \leftarrow \{(u, v, w_{uv}) \in E : w_{uv} > 0\}$ ▷positive edges
2: $E^- \leftarrow \{(u, v, w_{uv}) \in E : w_{uv} < 0\}$ ▷negative edges
3: $E^0 \leftarrow \{(u, v, w_{uv}) \in E : w_{uv} = 0\}$ ▷zero edges
4: Extract weight lists W^+, W^-
5: Randomly permute W^+, W^- separately
6: Reassign permuted weights to E^+, E^- (same locations)
7: **return** $E' = E'^+ \cup E'^- \cup E^0$

Algorithm 3. Type C randomization.

Require: Bipartite edge list $E = \{(u, v, w_{uv})\}$
1: **for** each left node u **do**
2: Split edges of u into zeros and nonzeros
3: Permute nonzero weights only within node u
4: Reassign them to u's edges
5: **end for**
6: **return** all updated edges

Algorithm 4. Type D randomization.

Require: Bipartite edge list $E = \{(u, v, w_{uv})\}$
1: **for** each right node v **do**
2: Separate zeros and nonzeros of node v
3: Permute nonzero weights of v
4: Reassign within node v
5: **end for**
6: **return** all updated edges

Algorithm 5. Type E randomization.

Require: Bipartite edge list $E = \{(u, v, w_{uv})\}$
1: **for** each left node u **do**
2: Collect u's positive edges and negative edges separately
3: Permute positive weights of u
4: Permute negative weights of u
5: Reassign them to the same positive/negative positions
6: **end for**
7: **return** all edges plus zeros

Algorithm 6. Type F randomization.

Require: Bipartite edge list $E = \{(u, v, w_{uv})\}$
1: **for** each right node v **do**
2: Collect v's positive edges and negative edges separately
3: Permute positive weights of v
4: Permute negative weights of v
5: Reassign them to the same positive/negative positions
6: **end for**
7: **return** all edges plus zeros

Algorithm 7. Type G randomization.

Require: Bipartite edge list $E = \{(u, v, w_{uv})\}$
1: Extract positive edges E^{+} and zero/negative edges $E^{\leq 0}$
2: **for** $t = 1$ to number of swaps **do**
3: Pick two positive edges (u_1, v_1) and (u_2, v_2)
4: Propose swap to (u_1, v_2) and (u_2, v_1)
5: **if** swap does not create duplicates or zero-edges **then**
6: Apply the swap
7: **end if**
8: **end for**
9: Reattach all negative weights in original positions (shuffled if needed)
10: Return full edge list

ORCID iDs

Robert Jankowski 0000-0002-6288-509X
Filippo Radicchi 0000-0002-8352-1287
M Ángeles Serrano 0000-0001-8779-5931
Marián Boguñá 0000-0001-7833-3487
Santo Fortunato 0000-0002-9039-4730

References

[1] Castelvecchi D 2016 Can we open the black box of ai? *Nat. News* **538** 20
[2] Guidotti R, Monreale A, Ruggieri S, Turini F, Giannotti F and Pedreschi D 2018 A survey of methods for explaining black box models *ACM Comput. Surv.* **51** 1–42
[3] Sharkey L *et al* 2025 Open problems in mechanistic interpretability (arXiv:2501.16496)
[4] Bereska L and Gavves E 2024 Mechanistic interpretability for ai safety–a review (arXiv:2404.14082)
[5] Pósfai M and Barabási A-L'o 2016 *Network Science* vol 3 (Citeseer)
[6] Newman M E J 2003 The structure and function of complex networks *SIAM Rev.* **45** 167–256
[7] Newman M E J 2001 The structure of scientific collaboration networks *Proc. Natl Acad. Sci.* **98** 404–9
[8] Girvan M and Newman M E J 2002 Community structure in social and biological networks *Proc. Natl Acad. Sci.* **99** 7821–6
[9] Bengio Y, Courville A and Vincent P 2013 Representation learning: a review and new perspectives *IEEE Trans. Pattern Anal. Mach. Intell.* **35** 1798–828
[10] Ho T K 2022 Complexity of representations in deep learning *2022 26th Int. Conf. on Pattern Recognition (ICPR)* (IEEE) pp 2657–63
[11] Lampinen A K, Chan S C Y and Hermann K 2024 Learned feature representations are biased by complexity, learning order, position, and more (arXiv:2405.05847)
[12] Lin T-Y, RoyChowdhury A and Maji S 2015 Bilinear CNN models for fine-grained visual recognition *Proc. IEEE Int. Conf. on Computer Vision* pp 1449–57
[13] ONeill J 2020 An overview of neural network compression (arXiv:2006.03669)
[14] Cheng H, Zhang M and Shi J Q 2024 A survey on deep neural network pruning: Taxonomy, comparison, analysis and recommendations *IEEE Trans. Pattern Anal. Mach. Intell.* **46** 10558–78
[15] Liang T, Glossner J, Wang L, Shi S and Zhang X 2021 Pruning and quantization for deep neural network acceleration: a survey *Neurocomputing* **461** 370–403
[16] Hassibi B, Stork D G and Wolff G J 1993 Optimal Brain Surgeon and General Network Pruning *Proc. IEEE Int. Conf. on Neural Networks*
[17] Molchanov P, Tyree S, Karras T, Aila T and Kautz J 2017 Pruning convolutional neural networks for resource efficient inference *Int. Conf. on Learning Representations*
[18] He Y, Zhang X and Sun J 2017 Channel pruning for accelerating very deep neural networks *Proc. IEEE Int. Conf. on Computer Vision* pp 1389–97
[19] Lee N, Ajanthan T and Torr P H S 2019 SNIP: single-shot network pruning based on connection sensitivity *Int. Conf. on Learning Representations*
[20] Thamm M, Staats M and Rosenow B 2022 Random matrix analysis of deep neural network weight matrices *Phys. Rev. E* **106** 054124
[21] Staats M, Thamm M and Rosenow B 2023 Boundary between noise and information applied to filtering neural network weight matrices *Phys. Rev. E* **108** L022302
[22] Staats M, Thamm M and Rosenow B 2024 Small singular values matter: a random matrix analysis of transformer models (arXiv:2410.17770)
[23] Garrod C and Keating J P 2024 Unifying low dimensional observations in deep learning through the deep linear unconstrained feature model (arXiv:2404.06106)
[24] Courbariaux M, Bengio Y and David J-P 2015 Binaryconnect: training deep neural networks with binary weights during propagations *Advances in Neural Information Processing Systems* vol 28 pp 3123–31
[25] Rastegari M, Ordonez V, Redmon J and Farhadi A 2016 XNOR-Net: ImageNet classification using binary convolutional neural networks *European Conf. on Computer Vision* pp 525–42
[26] Bethge J, Bartz C, Yang H, Chen Y and Meinel C 2020 MeliusNet: can binary neural networks achieve mobilenet-level accuracy? (arXiv:2001.05936)
[27] Klabunde M, Schumacher T, Strohmaier M and Lemmerich F 2023 Similarity of neural network models: a survey of functional and representational measures *ACM Comput. Surv.* **57** 1–52
[28] Raghu M, Gilmer J, Yosinski J and Sohl-Dickstein J 2017 Svcca: singular vector canonical correlation analysis for deep representations *Advances in Neural Information Processing Systems* vol 30 pp 6076–85
[29] Kriegeskorte N, Mur M and Bandettini P A 2008 Representational similarity analysis-connecting the branches of systems neuroscience *Front. Syst. Neurosci.* **2** 4
[30] Dwivedi K and Roig G 2019 Representation similarity analysis for efficient task taxonomy and transfer learning *2019 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)* pp 12379–88
[31] Kornblith S, Norouzi M, Lee H and Hinton G 2019 Similarity of neural network representations revisited *Proc. 36th Int. Conf. on Machine Learning* pp 3519–29
[32] Madani O, Pennock D and Flake G 2004 Co-validation: using model disagreement on unlabeled data to validate classification algorithms *Advances in Neural Information Processing Systems* vol 17 (MIT Press)
[33] Marx C, Calmon F and Ustun B 2020 Predictive multiplicity in classification *Proc. 37th Int. Conf. on Machine Learning* (*Proc. of Machine Learning Research* vol 119) (PMLR) pp 6765–74
[34] Ba L J and Caruana R 2014 Do deep nets really need to be deep? *Advances in Neural Information Processing Systems* vol 27 (Curran Associates, Inc.)

[35] Li Y, Zhang Z, Liu B, Yang Z and Liu Y 2021 Modeldiff: testing-based dnn similarity comparison for model reuse detection *Proc. 30th ACM SIGSOFT Int. Symp. on Software Testing and Analysis* (*ISSTA'21*) (ACM) pp 139–51
[36] Mukherjee K and Rogers T T 2020 How does task structure shape representations in deep neural networks? *NeurIPS 2020 Workshop SVRHM*
[37] Zilly J, Hetzel L, Censi A and Frazzoli E 2019 Quantifying the effect of representations on task complexity (arXiv:1912.09399)
[38] Lecun Y, Bottou L, Bengio Y and Haffner P 1998 Gradient-based learning applied to document recognition *Proc. IEEE* **86** 2278–324
[39] Xiao H, Rasul K and Vollgraf R 2017 Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (arXiv:1708.07747)
[40] Krizhevsky A 2012 *Learning Multiple Layers of Features From Tiny Images* (University of Toronto)
[41] Wang Z, Bovik A C, Sheikh H R and Simoncelli E P 2004 Image quality assessment: from error visibility to structural similarity *IEEE Trans. Image Process.* **13** 600–12
[42] Zhang R, Isola P, Efros A A, Shechtman E and Wang O 2018 The unreasonable effectiveness of deep features as a perceptual metric *2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition* pp 586–95
[43] Bonifazi G, Cauteruccio F, Corradini E, Marchetti M, Ursino D and Virgili L 2024 A network analysis-based framework to understand the representation dynamics of graph neural networks *Neural Comput. Appl.* **36** 1875–97
[44] Scabini L F S and Bruno O M 2023 Structure and performance of fully connected neural networks: Emerging complex network properties *Physica A* **615** 128585
[45] Zhang X-J, Moore J M, Yan G and Xiang Li 2023 universal structural patterns in sparse recurrent neural networks *Commun. Phys.* **6** 243
[46] Malfa E L, Malfa G L, Nicosia G and Latora V 2021 Characterizing learning dynamics of deep neural networks via complex networks *2021 IEEE 33rd Int. Conf. on Tools With Artificial Intelligence (ICTAI)* (IEEE) pp 344–51
[47] Jiang C, Huang Z, Pedapati T, Chen P-Y, Sun Y and Gao J 2024 Network properties determine neural network performance *Nat. Commun.* **15** 5718
[48] Lu Y, Yang W, Zhang Y, Chen Z, Chen J, Xuan Q, Wang Z and Yang X 2022 Understanding the dynamics of dnns using graph modularity *European Conf. on Computer Vision* (Springer) pp 225–42
[49] Mocanu D C, Mocanu E, Stone P, Nguyen P H, Gibescu M and Liotta A 2018 Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science *Nat. Commun.* **9** 2383
[50] Scabini L, De Baets B and Bruno O M 2024 Improving deep neural network random initialization through neuronal rewiring *Neurocomputing* **599** 128130
[51] Xie S, Kirillov A, Girshick R and Kaiming H 2019 Exploring randomly wired neural networks for image recognition *Proc. IEEE/CVF Int. Conf. on Computer Vision* pp 1284–93
[52] You J, Leskovec J, He K and Xie S 2020 Graph structure of neural networks *Int. Conf. on Machine Learning* (PMLR) pp 10881–91
[53] Du Y, Wang L, Guo L, Han J, Liu T and Hu X 2023 Topological similarity between artificial and biological neural networks *2023 IEEE 20th Int. Symp. on Biomedical Imaging (ISBI)* (IEEE) pp 1–5
[54] Waqas A, Farooq H, Bouaynaya N C and Rasool G 2022 Exploring robust architectures for deep artificial neural networks *Commun. Eng.* **1** 46
[55] Blöcker C, Rosvall M, Scholtes I and West J D 2025 Insights from network science can advance deep graph learning (arXiv:2502.01177)
[56] Rai D, Zhou Y, Feng S, Saparov A and Yao Z 2024 A practical review of mechanistic interpretability for transformer-based language models (arXiv:2407.02646)
[57] El B, Choudhury D, Liò P and Joshi C K 2025 Towards mechanistic interpretability of graph transformers via attention graphs (arXiv:2502.12352)
[58] Pearce M T, Dooms T, Rigg A, Oramas J M and Sharkey L 2024 Bilinear mlps enable weight-based mechanistic interpretability (arXiv:2410.08417)
[59] Frankle J and Carbin M 2019 The lottery ticket hypothesis: Finding sparse, trainable neural networks *Int. Conf. on Learning Representations* (available at: https://openreview.net/forum?id = rJl-b3RcF7)
[60] Benzi R, Sutera A and Vulpiani A 1981 The mechanism of stochastic resonance *J. Phys. A: Math. Gen.* **14** L453
[61] Ludwig S 2025 Stochastic resonance improves the detection of low contrast images in deep learning models (arXiv:2502.14442)
[62] Gammaitoni L, Hänggi P, Jung P and Marchesoni F 1998 Stochastic resonance *Rev. Mod. Phys.* **70** 223–87
[63] Benzi R, Parisi G, Sutera A and Vulpiani A 1982 Stochastic resonance in climatic change *Tellus A* **34** 10
[64] Benzi R, Parisi G, Sutera A and Vulpiani A 1983 A theory of stochastic resonance in climatic change *SIAM J. Appl. Math.* **43** 565–78
[65] Sanh V, Debut L, Chaumond J and Wolf T 2019 Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter (arXiv:1910.01108)
[66] Sang E F T K and De Meulder F 2003 Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition *Proc. 7th Conf. on Natural Language Learning at HLT-NAACL 2003* pp 142–7
[67] Devlin J, Chang M-W, Lee K and Toutanova K 2019 BERT: Pre-training of deep bidirectional transformers for language understanding *Proc. 2019 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (*Long and Short Papers* vol 1) (Association for Computational Linguistics) pp 4171–86
[68] Bai H, Zhang W, Hou L, Shang L, Jin J, Jiang X, Liu Q, Lyu M and King I 2021 BinaryBERT: Pushing the limit of BERT quantization *Proc. 59th Annual Meeting of the Association for Computational Linguistics and the 11th Int. Joint Conf. on Natural Language Processing (Volume 1: Long Papers)* (Association for Computational Linguistics) pp 4334–48
[69] Jankowski R 2026 Code for "Task complexity shapes internal representations and robustness in neural networks" *GitHub* (available at: https://github.com/robertjankowski/probing-neural-networks)