# Feature Contamination: Neural Networks Learn Uncorrelated Features and Fail to Generalize

Tianren Zhang [*1] Chujie Zhao [*1] Guanyu Chen [1] Yizhou Jiang [1] Feng Chen [1 2]

## Abstract

Learning representations that generalize under distribution shifts is critical for building robust machine learning models. However, despite significant efforts in recent years, algorithmic advances in this direction have been limited. In this work, we seek to understand the fundamental difficulty of out-of-distribution generalization with deep neural networks. We first empirically show that perhaps surprisingly, even allowing a neural network to *explicitly* fit the representations obtained from a teacher network that *can* generalize out-of-distribution is insufficient for the generalization of the student network. Then, by a theoretical study of two-layer ReLU networks optimized by stochastic gradient descent (SGD) under a structured feature model, we identify a fundamental yet unexplored feature learning proclivity of neural networks, *feature contamination*: neural networks can learn *uncorrelated* features together with predictive features, resulting in generalization failure under distribution shifts. Notably, this mechanism essentially differs from the prevailing narrative in the literature that attributes the generalization failure to spurious correlations. Overall, our results offer new insights into the non-linear feature learning dynamics of neural networks and highlight the necessity of considering inductive biases in out-of-distribution generalization.[1]

## 1. Introduction

The capability of generalizing under distribution shifts is crucial for machine learning systems to be deployed in the wild (Amodei et al., 2016; Beery et al., 2018; Koh et al., 2021). In the last decade, it has proved that the conventional principle of empirical risk minimization (ERM), when combined with deep neural networks, can lead to remarkable in-distribution (ID) generalization performance given sufficient training data. Nevertheless, this powerful paradigm can often fail in *out-of-distribution (OOD) generalization*, where distribution shifts occur due to data variations that are not well-covered in training (Torralba & Efros, 2011; Beery et al., 2018; Geirhos et al., 2018; DeGrave et al., 2021).

In response, recent years have witnessed a surge of developing algorithms that promote OOD generalization. However, the effectiveness of many proposed algorithms has been called into question by recent work (Gulrajani & Lopez-Paz, 2021; Koh et al., 2021), in which no tested algorithm exhibits a significant advantage over ERM under fair comparisons. On the other hand, it turns out that the most effective means of improving OOD generalization to date is pre-training on a more diverse dataset (Taori et al., 2020; Wiles et al., 2022), with notable examples including CLIP (Radford et al., 2021) and GPT (Brown et al., 2020). Yet, using additional pre-training data also blurs the notion of "OOD" itself since it essentially expands the training distribution. Moreover, it has been observed that when the test distribution differs from the pre-training distribution, pre-trained models can also suffer from performance degradation (Bommasani et al., 2022; Liu et al., 2023; Li & Flanigan, 2023).

The limited algorithmic success underlines the necessity of identifying and understanding the fundamental factors behind OOD generalization. In particular, a prevailing narrative in the literature attributes the OOD generalization failure to *spurious correlations* (Arjovsky et al., 2019; Nagarajan et al., 2021; Schölkopf et al., 2021). This explanation is inspired by the observation that the representations learned by ERM can absorb features that have *correlational* yet non-causal relationships with the output (Beery et al., 2018; Geirhos et al., 2020), and it has motivated a main line of algorithmic endeavor of designing better representation learning objectives in recent years (Arjovsky et al., 2019; Krueger et al., 2021; Mitrovic et al., 2021; Chen et al., 2022; Shi et al., 2022). However, despite being intuitive, it remains elusive how much this failure mode actually contributes to the OOD generalization failure in practice—as we will elab-

[*]Equal contribution [1]Department of Automation, Tsinghua University, Beijing, China [2]LSBDPA Beijing Key Laboratory, Beijing, China. Correspondence to: Feng Chen <chenfeng@mail.tsinghua.edu.cn>.

*Proceedings of the $41^{st}$ International Conference on Machine Learning*, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1] Code is available at `https://github.com/trzhang0116/feature-contamination`.

orate in the following sections, there exists a major OOD generalization gap in many tasks that *cannot* be straightforwardly explained by spurious correlations, implying that there must exist some more dominant factors.

On the theoretical side, a series of work has been devoted to analyzing the failure modes of OOD generalization. However, existing analysis has two major limitations: (i) conceptually, most studies only consider the failure mode due to spurious correlations; (ii) technically, most studies either only consider *linear* models such as linear classification over prescribed features or neural tangent kernels (Arjovsky et al., 2019; Sagawa et al., 2020b; Nagarajan et al., 2021; Xu et al., 2021; Ahuja et al., 2021b;a; Pezeshki et al., 2021; Chen et al., 2022; Wang et al., 2022; Rosenfeld et al., 2022; Abbe et al., 2023; Chen et al., 2023), or only consider arbitrary *unstructured* models without taking into the account the role of optimization (Rosenfeld et al., 2021; Kamath et al., 2021; Ye et al., 2021)—this makes them unable to capture the inductive biases of today's most widely used model class, i.e., *neural networks*. As a result, it has been observed that many OOD generalization algorithms that enjoy provable guarantees in their theoretical models do not excel in practice (Gulrajani & Lopez-Paz, 2021).

Overall, the above results imply that current explanations and theoretical models on OOD generalization may *not* faithfully reflect real-world distribution shifts. Motivated by the gap between theory and practice, we argue that taking into account the inductive biases of neural networks is not only important but also *necessary* for understanding OOD generalization in the era of deep learning.

### 1.1. Our Results and Implications

In this work, we set out to understand the fundamental difficulty of OOD generalization with deep neural networks:

**Empirically,** inspired by the ongoing trend of designing specific representation learning objectives for OOD generalization (Arjovsky et al., 2019; Gulrajani & Lopez-Paz, 2021), we investigate what will happen in an "ideal" setting where good representations are *explicitly given* during training. Concretely, we show on a range of distribution shift benchmarks that perhaps surprisingly, even if we allow a neural network to explicitly fit the representations obtained from a teacher network that *can* generalize out-of-distribution, the performance of the student network can still significantly deteriorate under distribution shifts. Our results thus imply that only considering the effect of the representation learning objective is *insufficient* for understanding OOD generalization in practice without considering the inductive biases in optimization. Moreover, we show that the above generalization failure *cannot* be simply explained by spurious correlations or other existing explanations in the literature on OOD generalization.

**Theoretically,** we prove that in certain structured binary classification tasks where the data is generated from generalizable *core features* and other *background features* (formal definitions in Section 3), a randomly initialized two-layer ReLU neural network trained by SGD can achieve ID generalization given sufficient iterations, yet fails to generalize OOD. In particular, we show that the above failure mode differs fundamentally from prior work as it holds even when:

- Background features are *uncorrelated* with the label (this excludes the failure mode due to spurious correlations).
- Ground-truth labels can be perfectly predicted by core features (this excludes the failure mode due to lacking informative features for prediction).
- Core features and background features are distributed in orthogonal subspaces (this excludes the failure mode due to non-linearly entangled features in the input that may be hard to disentangle for the neural network in training).

Instead, we identify that the above failure stems from a fundamental yet unexplored feature learning proclivity, which we name *feature contamination*, of neural networks. In brief, feature contamination indicates that during the learning of core features, SGD-trained neural networks also learn background features simultaneously, even when background features are *uncorrelated* with the label and in the presence of weight decay. The reason for this phenomenon is that the neurons in the network tend to have *asymmetric activation* for different classes, resulting in non-zero expected gradient projections onto *both* the core feature subspace and the background feature subspace. This eventually leads to additional risks under distribution shifts due to the coupling of core and background features in the neurons' pre-activation. Moreover, we formally show that ReLU networks and linear networks are *provably different* in our setting with the latter exhibiting no such behavior, suggesting a separation between linear and non-linear models. Finally, we present empirical evidence on deep neural networks that connects feature contamination to the empirical OOD generalization failure observed in our experiments.

At a high level, we expect that feature contamination as a *novel inductive bias of SGD-trained neural networks* may also be used in more general contexts. For example, it may serve as a new perspective for understanding the *feature learning* process of (deep) neural networks, complementing other known inductive biases of neural networks such as the simplicity bias (Arpit et al., 2017; Shah et al., 2020).

## 2. Good Representations Are Hard to Learn Even when Explicitly Given in Training

We begin our analysis by an empirical study inspired by recent algorithmic explorations in OOD generalization.

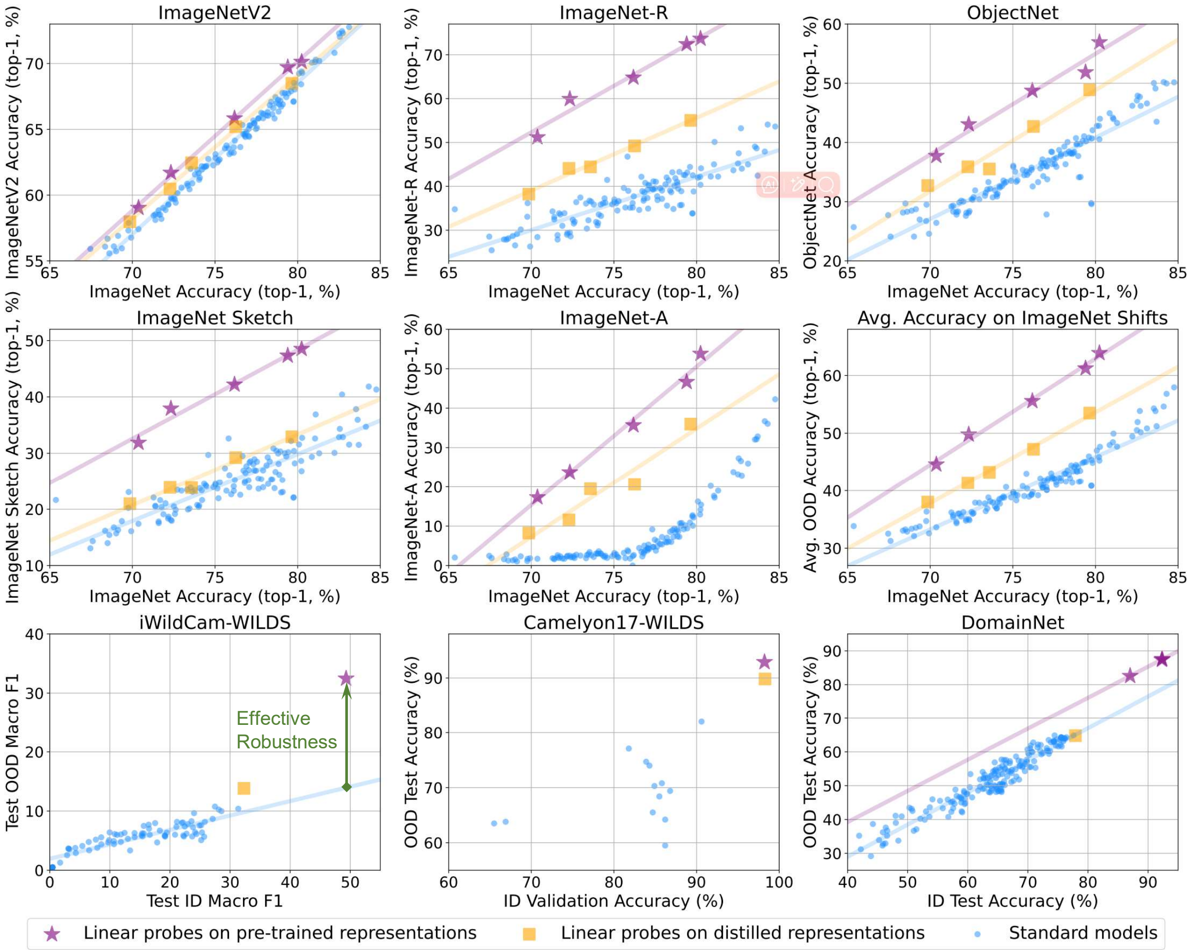


Figure 1: OOD performance ($y$-axes) v.s. ID performance ($x$-axes) for three model families including (i) linear probes on pre-trained representations (purple stars), (ii) linear probes on distilled representations (orange squares), and (iii) standard models trained on ID data (blue circles). The $y$-axis of the sixth panel stands for the average accuracy on ImageNet-based OOD test sets, averaged from the first five panels. Please refer to Section E for more details on each model family.

Existing work has made various attempts to learn OOD-generalizable models by designing auxiliary *representation learning* objectives beyond minimizing the prediction risk (see the baseline models in Section E.4 for some examples). Typically, those objectives reflect the premise of the properties of "good" representations. For example, a major line of work focuses on learning invariant representations across multiple training domains (Arjovsky et al., 2019; Chen et al., 2022; Shi et al., 2022), with the aim of removing domain-specific spurious correlations. Given the limited success of existing algorithms, here we would like to investigate the fundamental limitations of such representation learning methods. However, a main confounder in our study is that it is often unclear whether optimizing certain objectives is indeed effective for shaping the representation to satisfy the ideal properties—for example, it has been shown that optimizing some invariant representation learning objectives may lead to representations that are not truly invariant (Kamath et al., 2021; Rosenfeld et al., 2021).

To ablate the potential sub-optimality in the representation learning objective, we focus on an "ideal" scenario where the model has *explicit access to good representations* in training. Concretely, we leverage large-scale pre-trained models such as CLIP (Radford et al., 2021), which has shown remarkable robustness against distribution shifts, to extract good representations for each input: given a pre-trained CLIP encoder as a *teacher encoder*, we randomly initialize another *student encoder* with the *same* architecture. We then train the student encoder by minimizing the Euclidean distance between its output representations and the representations extracted by the teacher encoder, a process known as representation distillation (Hinton et al., 2014; Tian et al., 2020). Finally, we evaluate the ID and the OOD performance of both models. In total, our experiments span six different pre-trained models and eight extensively benchmarked distribution shift datasets, including five ImageNet-based natural distribution shift datasets (Taori et al., 2020), two in-the-wild distribution shift datasets from

WILDS (Koh et al., 2021), and a domain generalization dataset DomainNet (Peng et al., 2019). Please see Section E for more experimental details.

**Evaluation protocol.** We evaluate the ID and the OOD performance of pre-trained and distilled encoders by training linear probes on top of their output representations on the ID training set and then evaluate those linear probes on both ID and OOD test sets. Note that under our protocol, the linear probes still face OOD generalization tasks on the OOD test set, albeit with representations instead of raw images as input. To compare the OOD generalization ability of different models, we follow the evaluation protocol of *effective robustness* (Taori et al., 2020), which quantifies a model's distribution shift robustness as its OOD performance advantage over a baseline representing the OOD performance of standard models trained on ID data. Following Taori et al. (2020), we illustrate the effective robustness of our models using scatter plots, with $x$-axes representing ID performance and $y$-axes representing OOD performance.

**Results.** As shown in Figure 1, linear probes on distilled representations exhibit consistent effective robustness gains over standard models. This is not very surprising given that the distilled models have additional supervision provided by the representations obtained from the teacher models in training, while standard models do not. However, the upshot is that *even with explicit access to good representations, the OOD generalization performance of distilled models still lags far behind their pre-trained counterparts*. For example, distilled models only close about half of the average effective robustness gap between standard models and pre-trained models in ImageNet-based datasets, with even worse performance on iWildCam and DomainNet. Note that this is not due to the failure in distillation, as distilled models do achieve similar *ID performance* to that of the pre-trained models. Our results thus suggest that the limited algorithmic success in OOD generalization cannot be simply explained by not having a "good enough" representation learning objective.

**What is the cause of the above failure?** First, one may argue that spurious correlations can still play a role here, as the representations extracted by pre-trained models may also contain spurious correlations to the label even if they achieve generally good OOD performance. While we do acknowledge this possibility, we emphasize that spurious correlations *cannot* explain the large *OOD performance gap* between the distilled and the pre-trained models since we would expect them to be similarly impacted by the spurious correlations in their representations. Another plausible explanation is data leakage, i.e., CLIP may have "seen" many OOD examples in its pre-training stage and thus can extract richer predictive features for OOD examples (Zhang & Bottou, 2023). However, this possibility is nullified by a recent study (Mayilvahanan et al., 2024), which shows that CLIP's distribution shift robustness persists even when OOD examples are pruned from its pre-training dataset.

In a nutshell, we argue that *existing explanations are insufficient to account for the above OOD generalization gap*. This suggests that taking into the account the inductive biases of SGD-trained neural networks are necessary for understanding the OOD generalization failure in practice. In the following sections, we will formally identify feature contamination as a novel OOD generalization failure mode and further connect it to the results in this section.

## 3. A Theoretical Model of OOD Generalization

**Notation.** We use $[d]$ to denote the set $\{1, \ldots, d\}$ for positive integers $d$. For a set $\mathcal{S}$, we denote its cardinality by $|\mathcal{S}|$. For a vector $\mathbf{u}$, we denote its $\ell^2$-norm by $\|\mathbf{u}\|_2$. We denote the inner product of two vectors $\mathbf{u}$ and $\mathbf{v}$ by $\langle \mathbf{u}, \mathbf{v} \rangle$. We use the standard big-O notation: $O(\cdot)$, $\Omega(\cdot)$, $\Theta(\cdot)$, $o(\cdot)$, as well as their soft-O variants such as $\widetilde{\Theta}(\cdot)$ to hide logarithmic factors. For some parameter $d$, we use $\mathsf{poly}(d)$ to denote $\Theta(d^C)$ with some unspecified large constant $C$. We use $\mathbf{1}_E$ to denote the indicator function for an event $E$.

### 3.1. OOD Generalization Problem Setup

**Task and data.** We consider a binary classification task with an input space $\mathcal{X} \subseteq \mathbb{R}^d$, a label space $\mathcal{Y} = \{-1, 1\}$, a model class $\mathcal{H} : \mathcal{X} \to \mathbb{R}$, and a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. For every distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ and model $h \in \mathcal{H}$, the expected risk of $h$ on $\mathcal{D}$ is given by $\mathcal{R}_{\mathcal{D}}(h) := \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\ell(h(\mathbf{x}), \mathbf{y})$. In an OOD generalization problem, there exist a set of distributions $\mathbb{D}$ that consists of all possible distributions to which we would like our model to generalize. In training, we have access to a training distribution set $\mathbb{D}_{\mathrm{train}} \subsetneq \mathbb{D}$, where $\mathbb{D}_{\mathrm{train}}$ may contain one or multiple training distributions. Following prior work (Arjovsky et al., 2019; Sagawa et al., 2020a; Nagarajan et al., 2021; Rosenfeld et al., 2021), we aim to select a model $h \in \mathcal{H}$ to minimize the *OOD risk*, defined as the worst-case expected risk on $\mathbb{D}$:

$$\mathcal{R}_{\mathrm{OOD}}(h) := \max_{\mathcal{D} \in \mathbb{D}} \mathcal{R}_{\mathcal{D}}(h). \tag{1}$$

It is clear that without further assumptions on $\mathbb{D}_{\mathrm{train}}$ and $\mathbb{D}$, OOD generalization is impossible since no model can generalize to an arbitrary distribution. Fortunately, real-world distribution shifts are often *structured* with some structural similarities shared by different distributions. We can thus hope that such structures can be captured by certain algorithms to train models that can generalize OOD.

To formalize this, in this work we assume that both ID and OOD data are generated by a dictionary $\boldsymbol{M} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{d_0}) \in \mathbb{R}^{d \times d_0}$ consisting of $d_0$ features with each feature $\boldsymbol{m}_i \in \mathbb{R}^d$. Throughout the paper, we work with the case where $d_0$ is sufficiently large and $d \in [\Omega(d_0^{2.01}), \mathsf{poly}(d_0)]$. For simplicity, we assume that every feature satisfies $\|\boldsymbol{m}_i\|_2 = 1$ and different features are orthogonal: $\forall i \neq j \in [d_0], \langle \boldsymbol{m}_i, \boldsymbol{m}_j \rangle = 0$.[2]

Among all features in $\boldsymbol{M}$, we assume that there are $d_{\text{core}}$ features consistently correlating with the label in all distributions in $\mathbb{D}$. We denote the index set of those features by $\mathcal{S}_{\text{core}} \subsetneq [d_0]$ and refer to them as **core features** since they are consistently predictive of the label in all distributions. We refer to the remaining features as **background features** and denote their index set by $\mathcal{S}_{\text{bg}} = [d_0] \setminus \mathcal{S}_{\text{core}}$ with $d_{\text{bg}} := |\mathcal{S}_{\text{bg}}| = d_0 - d_{\text{core}}$. We assume that $d_{\text{core}} = \Theta(d_0)$ and $d_{\text{bg}} = \Theta(\frac{d_0}{\log d_0})$, so that the number of both core features and background features is non-negligible. With the above definitions, we introduce our ID and OOD data generation model in Definition 3.1.

**Definition 3.1** (ID and OOD data generation)**.** Under the feature model stated above, consider a training distribution (ID data distribution) $\mathcal{D}_{\text{train}} \in \mathbb{D}_{\text{train}}$ and a test distribution (OOD data distribution) $\mathcal{D}_{\text{test}} \in \mathbb{D} \setminus \mathbb{D}_{\text{train}}$.[3] Each example $(\mathbf{x}, \mathbf{y}) \sim \mathcal{D} \in \{\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}}\}$ is generated as follows:

1. Sample a label $\mathbf{y}$ from the uniform distribution over $\mathcal{Y}$.
2. Sample a weight vector $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_{d_0}) \in \mathbb{R}^{d_0}$ where different coordinates of $\mathbf{z}$ are independent random variables generated as follows:
   - **ID data (**$\mathcal{D} = \mathcal{D}_{\text{train}}$**):** for every $j \in [d_0]$, sample $\mathbf{z}_j$ from some distribution $\mathcal{D}_j$ over $[0, 1]$ such that its moments satisfy $\mu_{jp} := \mathbb{E}_{\mathcal{D}_j} \mathbf{z}_j^p = \Theta(1)$ for $p \in [3]$ and its variance satisfies $\sigma_j^2 = \Theta(1)$.
   - **OOD data (**$\mathcal{D} = \mathcal{D}_{\text{test}}$**):** for every $j \in [d_0]$, if $j \in \mathcal{S}_{\text{core}}$, sample $\mathbf{z}_j$ from $\mathcal{D}_j$ over $[0, 1]$; if $j \in \mathcal{S}_{\text{bg}}$, sample $\mathbf{z}_j$ from some distribution $\mathcal{D}_j'$ over $[-1, 0]$ such that $\mathbb{E}_{\mathcal{D}_j'} \mathbf{z}_j = -\Theta(1)$.
3. Generate $\mathbf{x} = \sum_{j \in \mathcal{S}_{\text{core}}} \mathbf{y}\mathbf{z}_j \boldsymbol{m}_j + \sum_{j \in \mathcal{S}_{\text{bg}}} \mathbf{z}_j \boldsymbol{m}_j$.

**Remarks on data generation.** Our data model formalizes a structured OOD generalization setup reflecting several facets of real-world OOD generalization problems:

- The explicit separation of core and background features captures structural assumptions that make OOD generalization tractable: under the distribution shifts on background features, there still exists a set of core features that enable robust classification. Hence, a model that discards background features and retains core features can generalize OOD. This rules out the ill-posed case where the ID data is not informative enough to learn a generalizable model (Tripuraneni et al., 2020; Xu et al., 2021; Kumar et al., 2022), and is also the key intuition of many OOD generalization algorithms aiming to learn invariant representations (Gulrajani & Lopez-Paz, 2021).
- The weights of background features are assumed to be independent of the label, rendering background features and labels *uncorrelated*. This differs from prior OOD generalization analysis (Arjovsky et al., 2019; Sagawa et al., 2020b; Nagarajan et al., 2021; Rosenfeld et al., 2021) where background features are assumed to be *spuriously correlated* with the label and hence useful for prediction. We intentionally make this assumption to "ablate" the effect of spurious correlations in feature learning.[4]

### 3.2. Model and Training

**Model.** We consider a model class $\mathcal{H}$ representing width-$m$ two-layer neural networks with ReLU activation. Formally, given hidden-layer weights $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_m) \in \mathbb{R}^{d \times m}$ and output-layer weights $\mathbf{a} = (a_1, \ldots, a_m)^\top \in \mathbb{R}^m$, the output of a model $h \in \mathcal{H}$ given an input $\mathbf{x} \in \mathcal{X}$ is given by

$$h(\mathbf{x}) = \sum\nolimits_{k \in [m]} a_k \cdot \mathsf{ReLU}(\langle \mathbf{w}_k, \mathbf{x} \rangle), \tag{2}$$

where $\mathsf{ReLU}(u) = \max\{u, 0\}, u \in \mathbb{R}$. Similar to practical design choices, we consider an *overparameterized* setting where $m \in [\Theta(d_0), \Theta(d)]$. We initialize each weight vector $\mathbf{w}_k, k \in [m]$ by sampling $\mathbf{w}_i^{(0)} \sim \mathcal{N}(\mathbf{0}, \sigma_0^2 \boldsymbol{I}_d)$ with $\sigma_0^2 = \frac{1}{d}$. We randomly initialize output-layer weights $\mathbf{a}$ by sampling $a_k \sim \mathsf{Uniform}\{-\frac{1}{m}, \frac{1}{m}\}$ independently for each $k \in [m]$. To simplify our analysis, we keep output-layer weights $\mathbf{a}$ fixed during training, which is a common assumption in analyzing two-layer neural networks (Allen-Zhu & Li, 2021; Karp et al., 2021; Allen-Zhu & Li, 2023b).

**Training.** We train the network using SGD to minimize a standard hinge loss $\ell(y, y') = \max\{1 - yy', 0\}$ with step size $\eta > 0$ for $T$ iterations. We also include a weight decay with strength $\lambda = O(\frac{d_0}{m^{1.01}})$ for regularization. At each iteration $t \in \{0, \ldots, T\}$, we i.i.d. sample a batch of examples $\{(\mathbf{x}_i^{(t)}, \mathbf{y}_i^{(t)})\}_{i \in [N]} \sim \mathcal{D}_{\text{train}}^N$ with batch size $N = \mathsf{poly}(d)$ and consider the following empirical loss:

$$\widehat{\mathcal{L}}(h^{(t)}) = \frac{1}{N} \sum_{i \in [N]} \ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right) + \frac{\lambda}{2} \sum_{k \in [m]} \|\mathbf{w}_k^{(t)}\|_2^2, \tag{3}$$

where we use $h^{(t)}$ to denote the model at iteration $t$, with weights $\mathbf{W}^{(t)} = (\mathbf{w}_1^{(t)}, \ldots, \mathbf{w}_m^{(t)})$. The SGD update for each weight vector $\mathbf{w}_k, k \in [m]$ is then given by

$$\mathbf{w}_k^{(t+1)} = \mathbf{w}_k^{(t)} - \eta \nabla_{\mathbf{w}_k^{(t)}} \widehat{\mathcal{L}}(h^{(t)}). \tag{4}$$

[2] Another advantage of assuming orthogonal features is that this prevents the network from learning background features due to their correlations with core features. We note that our results can be extended to more general settings without orthogonality.

[3] Note that $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ can also be (weighted) *mixtures* of multiple distributions in $\mathbb{D}_{\text{train}}$ and $\mathbb{D} \setminus \mathbb{D}_{\text{train}}$, respectively.

[4] On the other hand, our results can be extended to the settings where some background features have spurious correlations.

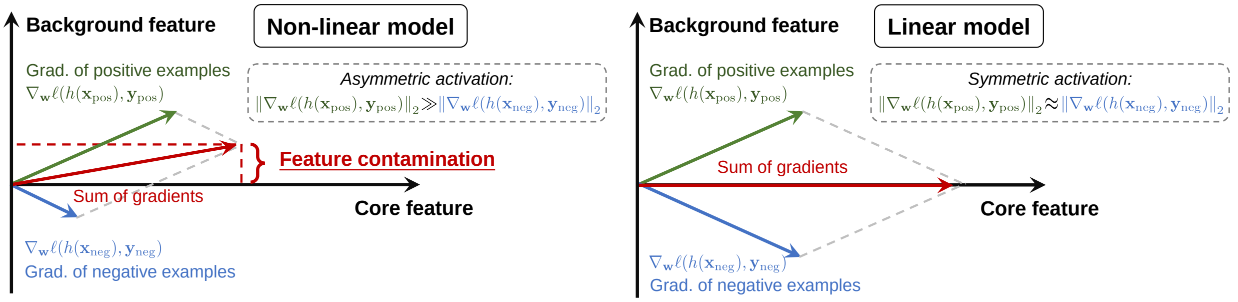

Background feature
Non-linear model
Grad. of positive examples
$\nabla_{\mathbf{w}}\ell(h(\mathbf{x}_{\text{pos}}),\mathbf{y}_{\text{pos}})$
Asymmetric activation:
$\|\nabla_{\mathbf{w}}\ell(h(\mathbf{x}_{\text{pos}}),\mathbf{y}_{\text{pos}})\|_2 \gg \|\nabla_{\mathbf{w}}\ell(h(\mathbf{x}_{\text{neg}}),\mathbf{y}_{\text{neg}})\|_2$
Feature contamination
Sum of gradients
Core feature
$\nabla_{\mathbf{w}}\ell(h(\mathbf{x}_{\text{neg}}),\mathbf{y}_{\text{neg}})$
Grad. of negative examples
Background feature
Linear model
Grad. of positive examples
$\nabla_{\mathbf{w}}\ell(h(\mathbf{x}_{\text{pos}}),\mathbf{y}_{\text{pos}})$
Symmetric activation:
$\|\nabla_{\mathbf{w}}\ell(h(\mathbf{x}_{\text{pos}}),\mathbf{y}_{\text{pos}})\|_2 \approx \|\nabla_{\mathbf{w}}\ell(h(\mathbf{x}_{\text{neg}}),\mathbf{y}_{\text{neg}})\|_2$
Sum of gradients
Core feature
$\nabla_{\mathbf{w}}\ell(h(\mathbf{x}_{\text{neg}}),\mathbf{y}_{\text{neg}})$
Grad. of negative examples

Figure 2: A diagram of feature contamination in our binary classification setting. **Left:** for models with non-linear activation functions such as ReLU, activation asymmetry leads to non-zero gradient projections onto background features. **Right:** for linear models, background features are cancelled out in the gradients, exhibiting no feature contamination.

## 4. Main Theoretical Results

In this section, we present our main theoretical results, provide mathematical reasoning of why feature contamination happens, and discuss its impact on generalization. We also include numerical results and show that our findings can be extended to more general data and neural network models.

**Technical challenges.** As we have discussed in Section 1, most existing theoretical work on OOD generalization *separates generalization and optimization* and directly studies the *global minimizers* of their training objeves without considering optimization dynamics. By contrast, without a unique global minimizer, our setup requires an explicit analysis on the SGD optimization trajectory, which is known to be challenging due to its *non-convex* and *non-linear* nature. Prior work has studied fine-tuning pre-trained models for OOD generalization in the context of two-layer *linear networks* (Kumar et al., 2022; Lee et al., 2023). Analyzing non-linear networks further requires a careful treatment on the activation property of the neurons, which results in SGD dynamics that essentially deviate from linear networks.

**Our approach.** At a high level, our analysis is based on the construction of two neuron subsets $\mathcal{N}_y^{(t)}$ (see Definition A.4) for $y \in \mathcal{Y} = \{-1, 1\}$ at iteration $t \in \{0, \dots, T\}$ so that each subset has cardinality $\Theta(m)$ and its neurons are randomly initialized to have large enough expected correlations with the examples from the class $y$ (i.e., "winning the lottery tickets" (Frankle & Carbin, 2019; Allen-Zhu & Li, 2021)). We then apply the Berry-Esseen theorem to bound the class-conditional activation probabilities of ReLU for the neurons in those subsets. By a careful treatment of the activation probabilities as the neurons evolve during training, we can bound the expected gradients for each neuron in $\mathcal{N}_y^{(t)}$ at every step $t$, hence iteratively tracking its weight updates throughout training. This treatment allows us to characterize the output of the network up to constant factors while avoiding the nuisance of analyzing the activation probability of *every* neuron in the network, which turns out to be very challenging. For ease of presentation, in the sequel we separate our main results into four parts and introduce them progressively, with an illustration of our key ideas in Figure 2. Complete proofs of all theoretical results are deferred to Appendix I.

***1. Neuron activation is asymmetric.*** Our key insight is that during training, every neuron in $\mathcal{N}_y^{(t)}$ has the incentive to be positively correlated with the examples from at most one class $\mathbf{y}_{\text{pos}} = y$ (whether $y = 1$ or $y = -1$ depends on the random initialization of the neuron); we refer to those examples as *positive examples* $(\mathbf{x}_{\text{pos}}, \mathbf{y}_{\text{pos}}) \sim \mathcal{D}_{\text{train}}|\mathbf{y} = \mathbf{y}_{\text{pos}}$ for that neuron. Correspondingly, we refer to examples from the other class $\mathbf{y}_{\text{neg}} = -y$ as *negative examples* $(\mathbf{x}_{\text{neg}}, \mathbf{y}_{\text{neg}}) \sim \mathcal{D}_{\text{train}}|\mathbf{y} = \mathbf{y}_{\text{neg}}$ for the neuron. Due to randomness at initialization, we can show that $|\mathcal{N}_y^{(0)}| = \Theta(m)$ for both $y \in \{-1, 1\}$ and, after sufficient SGD iterations, all neurons in $\mathcal{N}_y^{(t)}$ will accumulate (in expectation) positive correlations with examples from $y$ and negative correlations with examples from $-y$, resulting in class-wise asymmetry in their activation as shown by Theorem 4.1.

**Theorem 4.1** (Activation asymmetry)**.** *For every $\eta \le \frac{1}{\mathsf{poly}(d_0)}$ and every $y \in \mathcal{Y}$, there exists $T_0 = \widetilde{\Theta}(\frac{m}{\eta\sqrt{d}})$ such that w.h.p., for every $t \ge T_0$, there exist $\Theta(m)$ neurons in which the weight $\mathbf{w}_k^{(t)}$ for each neuron satisfies:*

$$\begin{aligned} &\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \ge 0] = 1 - O\left(d_0^{-\frac{1}{2}}\right), \\ &\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \ge 0] = o(1). \end{aligned} \tag{5}$$

***2. Activation asymmetry leads to feature contamination.*** Note that for every $k \in [m]$, the weight vector of the $k$-th neuron (we will also refer to it as the *learned feature* of the neuron) after $t$ iterations can be written as

$$\mathbf{w}_k^{(t)} = \sum_{j\in\mathcal{S}_{\text{core}}} \langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle \boldsymbol{m}_j + \sum_{j\in\mathcal{S}_{\text{bg}}} \langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle \boldsymbol{m}_j + \text{res}, \tag{6}$$

where the residual term satisfies $\langle \text{res}, \boldsymbol{m}_j\rangle = 0$ for every $j \in [d_0]$ and thus can be neglected. Intuitively, Eq. (6) indicates that the learned feature can be decomposed into

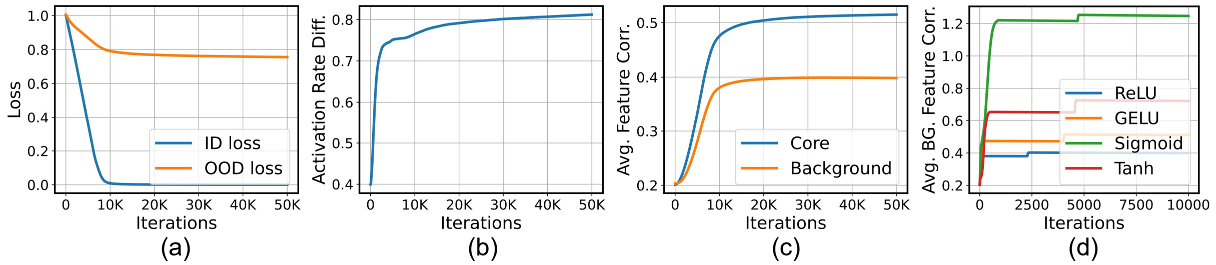


Figure 3: Numerical results. **(a)** *ID and OOD risks:* During training, ID loss quickly approaches zero, while OOD loss stays high. **(b)** *Activation asymmetry:* the difference of average neuron activation rates for different classes largely increases during training. **(c)** *Feature contamination:* the average correlations between neuron weights and both core features and *uncorrelated background features* increase in training. **(d)** Feature contamination also occurs in more general settings with different activation functions. Please refer to Section F.1 for more details and results.

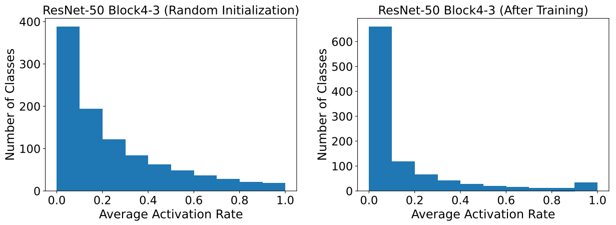


Figure 4: *Class-averaged* activation rate histograms of a randomly initialized CLIP-RN50 **(left)** and a distilled CLIP-RN50 **(right)**. After training, more classes have smaller average activation rates close to zero and only a small number of classes have large average activation rates.

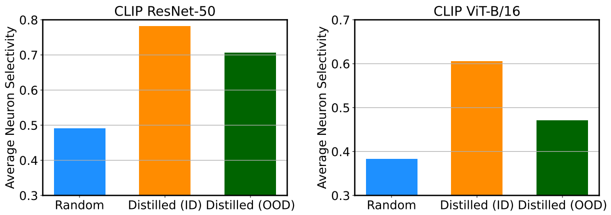


Figure 5: Average *neuron selectivity* of random and distilled CLIP-RN50 **(left)** and CLIP-ViT-B/16 **(right)** models. Distilled models have larger selectivity compared with random models and exhibit a selectivity drop in OOD data. Please refer to Section F.3 for more details.

its projections onto different feature vectors. Meanwhile, as we will prove in Lemma A.3, at iteration $t$, the gradient projection onto background features for every neuron $k \in \mathcal{N}_y^{(t)}$ satisfies: for every $j \in \mathcal{S}_{\text{bg}}$,

$$\begin{aligned}&\langle -\nabla_{\mathbf{w}_k^{(t)}} \widehat{\mathcal{L}}(h^{(t)}), \boldsymbol{m}_j \rangle \propto \\ &\qquad \mathbb{E}_{(\mathbf{x},\mathbf{y})}(\mathbf{1}_{\mathbf{y}=\mathbf{y}_{\text{pos}}} - \mathbf{1}_{\mathbf{y}=\mathbf{y}_{\text{neg}}})\mathbf{1}_{\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \ge 0} \mathbf{z}_j.\end{aligned} \tag{7}$$

By Theorem 4.1, we then have that for at least $\Theta(m)$ neurons in $\mathcal{N}_y^{(t)}$, $\mathbb{E}_{\mathbf{x}|\mathbf{y}=\mathbf{y}_{\text{pos}}} \mathbf{1}_{\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \ge 0}$ would be much larger than $\mathbb{E}_{\mathbf{x}|\mathbf{y}=\mathbf{y}_{\text{neg}}} \mathbf{1}_{\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \ge 0}$, resulting in a positive gradient projection onto *every* background feature $\boldsymbol{m}_j$ *regardless of its correlation with the label*. We refer to this feature learning proclivity of neural networks as **feature contamination**. Formally, Theorem 4.2 shows that this will result in the neurons' learned features accumulating both correlated core features and *uncorrelated* background features.

**Theorem 4.2** (Learned features)**.** *For every $\eta \le \frac{1}{\mathsf{poly}(d_0)}$ and every $y \in \mathcal{Y}$, there exists $T_1 = \Theta(\frac{m}{\eta d_0})$ such that w.h.p., after $T_1$ iterations, there exist $\Theta(m)$ neurons in which the weight $\mathbf{w}_k^{(T_1)}$ for each neuron satisfies the following:*

$$\begin{aligned}&\sum\nolimits_{j \in \mathcal{S}_{\text{core}}} \mu_{j1} \langle \mathbf{w}_k^{(T_1)}, \boldsymbol{m}_j \rangle = y \cdot \Theta(1), \\ &\sum\nolimits_{j \in \mathcal{S}_{\text{bg}}} \mu_{j1} \langle \mathbf{w}_k^{(T_1)}, \boldsymbol{m}_j \rangle = \widetilde{\Theta}(1).\end{aligned} \tag{8}$$

***3. Feature contamination induces large OOD risk.*** Intuitively, this result is a direct consequence of the *coupling* of core features and background features in the neurons' pre-activation as shown by Theorem 4.2. With this coupling, negative shifts of background features can *reduce the activation of the neuron*, resulting in OOD risk. In extreme cases, when the pre-activation of a neuron is reduced to negative, the contribution of the core features extracted by the neuron will also diminish. Formally, Theorem 4.3 quantifies such impact of feature contamination on ID and OOD risks.

**Theorem 4.3** (ID and OOD risks)**.** *For every $\eta \le \frac{1}{\mathsf{poly}(d_0)}$, there exists $T_2 = \widetilde{\Theta}(\frac{m}{\eta d_0})$ such that w.h.p., after $T_2$ iterations, the trained model $h^{(T_2)}$ satisfies the following:*

$$\mathcal{R}_{\mathcal{D}_{\text{train}}}(h^{(T_2)}) \le o(1), \ \mathcal{R}_{\text{OOD}}(h^{(T_2)}) = \widetilde{\Theta}(1). \tag{9}$$

***4. Linear networks are provably free from feature contamination.*** Finally, to further understand the role of non-linearity, we prove that if we "remove" the non-linearity in the network by replacing each ReLU with the identity function, then feature contamination will no longer occur.

**Theorem 4.4** (Linear networks)**.** *If we replace the ReLU functions in the network with identity functions and keep other conditions the same as in Theorem 4.2, then with high probability, we have $|\langle \mathbf{w}_k^{(T_1)}, \boldsymbol{m}_j \rangle| \le \widetilde{O}(\frac{1}{\sqrt{d}})$ for every $k \in [m]$ and every $j \in \mathcal{S}_{\text{bg}}$.*

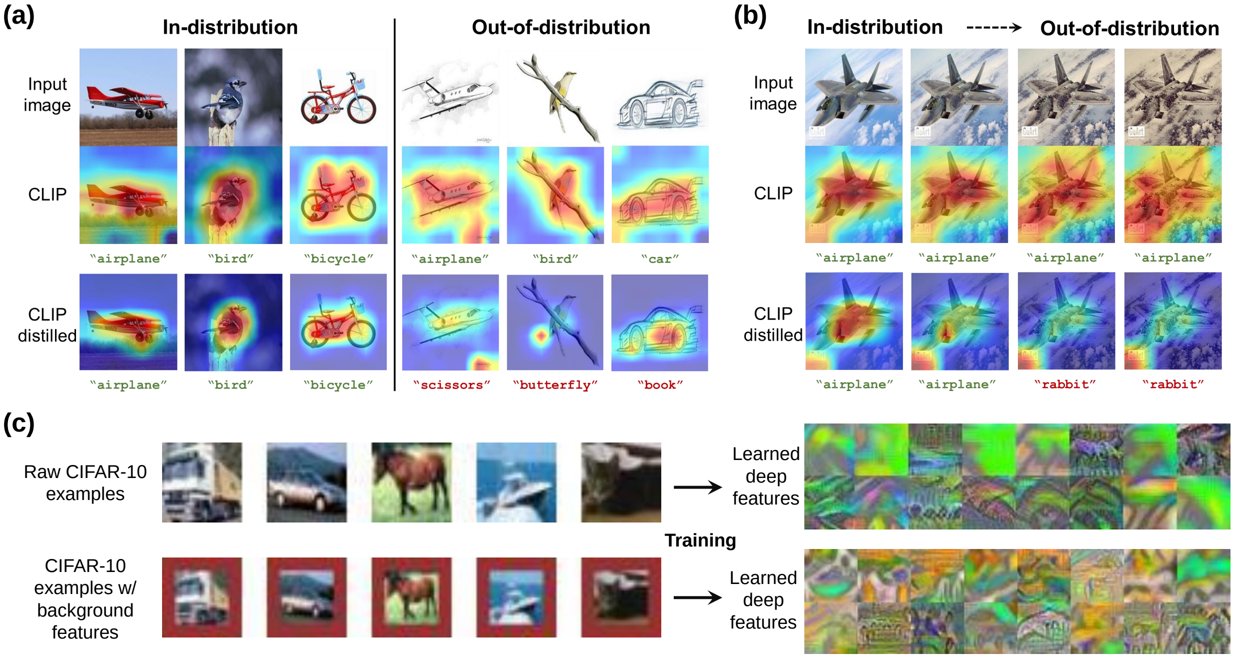


Figure 6: Empirical evidence for feature contamination in *deep* neural networks. **(a)** Grad-CAM results for CLIP-RN50 and a distilled CLIP-RN50 on DomainNet. **(b)** Grad-CAM results for CLIP-RN50 and a distilled CLIP-RN50 on controlled distribution shifts. For the distilled model, while the weights of core objects are dominant for ID images, they are *reduced* under distribution shifts. **(c)** Example images of the raw CIFAR-10 dataset and our modified version with background color features that are *uncorrelated* with the label, and the visualization of deep features learned by a ResNet on both datasets.

The main intuition of Theorem 4.4 is that without non-linearity, the activation magnitude for the examples from different classes would be no longer asymmetric: for two-layer linear networks, the gradient projection onto background features is akin to Eq. (7) but without the activation derivative $\mathbf{1}_{\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \geq 0}$. We then have $\langle -\nabla_{\mathbf{w}_k^{(t)}} \widehat{\mathcal{L}}(h^{(t)}), \boldsymbol{m}_j \rangle \approx 0$ for every $j \in \mathcal{S}_{\text{bg}}$ since positive gradients and negative gradients on $\mathbf{z}_j$ will now cancel out. As a result, the background features will not be accumulated during SGD.

**Numerical results.** As empirical evidence that corroborates our theory, we provide numerical results in Figure 3 that demonstrate the existence of activation asymmetry and feature contamination under our setting.

**Extensions to more general settings.** To show that feature contamination also occurs *beyond* our theoretical setting, we also conduct numerical experiments with several relaxations from our setting, including (i) *non-linear* relationships between core features and the label, (ii) more activation functions other than ReLU, and (iii) optimizers with adaptive step sizes other than vanilla SGD. As shown by Figure 3(d), feature contamination consistently occurs in more general settings. In Section F.1, we also provide numerical results in regression (representation distillation) tasks to show that feature contamination also occurs beyond classification.

## 5. Feature Contamination in Practice

In this section, we present empirical evidence that connects our theoretical results to practical OOD generalization failure and discuss possible solutions.

**Activation asymmetry in deep networks.** In our theoretical model, feature contamination stems from the asymmetric activation of the neurons. To examine whether deep networks trained on real-world data also exhibit this behavior, we compute the *class-averaged* activation rate histograms of the ResNet and ViT models trained in our experiments in Section 2. As shown in Figure 4 and Section F.2, both models exhibit activation asymmetry after training, with low average activation rates that are close to zero for most classes and high average activation rates for only a small number of classes. Moreover, we also adopt a more quantitative metric termed *neuron selectivity* that measures the difference of neuron activation magnitudes of different classes. As shown in Figure 5, distilled models have considerably larger average neuron selectivity than random models, which corroborates our theory. Please refer to Section F.3 for the implementation details of computing neuron selectivity.

**Prediction heatmap visualization.** In Figure 6(a), we visualize the prediction heatmaps of CLIP-RN50 and a distilled CLIP-RN50 in the DomainNet dataset using Grad-

CAM (Chattopadhay et al., 2018). An intriguing phenomenon revealed by the heatmaps is that for the distilled model, while the weights of core objects are dominant for ID images, they are *reduced* under distribution shifts, resulting in OOD generalization failure. Quantitatively, a similar observation is shown in Figure 5, where the selectivity of neurons drops in OOD data. Here we argue that *feature contamination explains this phenomenon*: the Grad-CAM score of each feature map is calculated by differentiating the classification score with respect to the feature map, thus being proportional to the corresponding neurons' activation. Hence, when the core features and the background features are coupled in the neurons' pre-activation (Theorem 4.2), the shift of background features can reduce the activation, which in turn reduces the Grad-CAM score of the feature map as visually observed. The reduction in neuron selectivity in OOD data can also be explained in a similar way.

**Prediction heatmaps under controlled distribution shifts.** In Figure 6(b), we consider a synthetic OOD generalization task based on image style transfer where we can manually control the "degree" of distribution shifts by controlling the amount of style change. This setting closely matches our data model in Definition 3.1 by keeping core features intact while *only* changing background features in the ID images. As shown in the figure, the prediction heatmaps exhibit visually similar patterns as the heatmaps of natural OOD images. This implies that our data model indeed captures some key characteristics of real-world distribution shifts and backs the explanation of feature contamination.

**Visualizing contaminated deep features.** To qualitatively show the impact of feature contamination on the learned features of *deep* neural networks, we visualize the features learned by a ResNet on a modified CIFAR-10 dataset in Figure 6(c), where the original images are padded with background red pixels *uncorrelated* with labels. Compared to the original CIFAR-10 dataset, the learned features on the modified CIFAR-10 dataset exhibits evident color differences, indicating that feature contamination also occurs in *deep features*. See Section F.4 for more details and results.

**Discussion on possible solutions.** In our theoretical model, although gradient descent accumulates both core and background features in the weight space due to feature contamination, there still exists a *subspace* (corresponding to the span of core features) where background features are not accumulated. Hence, constraining the SGD updates to this subspace would possibly lead to ideal generalization. This is consistent with prior results (Idnani et al., 2023) showing that projecting the network's intermediate representations onto certain subspaces may improve OOD generalization. However, how to effectively find the correct subspace for projection without the explicit access to the core and background feature subspaces remains an open problem.

# 6. Conclusion

In this section, we discuss potential implications of our results and list important future directions.

**Takeaway 1: OOD generalization algorithms need to consider inductive biases.** Many existing studies on OOD generalization motivate and analyze their algorithms using linear or unstructured models that do not capture the inductive biases of neural networks. Our results imply that OOD generalization may not be feasible without considering such inductive biases, calling for explicitly incorporating them into principled algorithm design.

**Takeaway 2: Non-linearity in neural networks elicits new OOD generalization challenges.** As we formally show in Section 4, feature contamination essentially stems from the gradient descent optimization process of non-linear neural networks even with *uncorrelated* background features, thus being orthogonal to the prevailing narrative of spurious correlations. This provides a new perspective on OOD generalization and may inspire new algorithmic design.

**Takeaway 3: Learned features may behave very differently from prescribed ones.** Many existing studies on OOD generalization explicitly or implicitly assume that we can directly work on a set of *well-separated* core/spurious features. While this assumption helps build intuitions, our results highlight that it can also be misleading since the features *learned* by neural networks may manifest in a *non-linearly coupled* manner, thus often diverging from the intuitions for prescribed, well-separated features.

## 6.1. Limitations, a Conjecture, and Future Work

While this work takes a step towards fully understanding OOD generalization in practice, our results still leave much room for improvement such as extensions to more general data distributions, multi-class classification, and more complicated network architectures. Meanwhile, while our current setup focuses on training from scratch, we envision that the viewpoint of feature contamination may also be helpful in analyzing the effect of *pre-training* on OOD generalization. In particular, we have the following conjecture:

**Conjecture.** Pre-training on a sufficiently diverse dataset does not remove uncorrelated features, but *linearizes* those features in the model's representations, hence mitigating feature contamination and improving OOD generalization.

We provide preliminary empirical evidence that supports the above conjecture in Section G, as well as more discussion on related empirical observations in recent work (Gandelsman et al., 2024; Mayilvahanan et al., 2024). Yet, we believe that rigorously proving this conjecture requires a more fine-grained treatment for the pre-training data distribution and the SGD dynamics and thus leave it as future work.

## Acknowledgements

This work was supported in part by the National Key Research and Development Program of China under STI 2030-Major Projects 2021ZD0200300, and in part by the National Natural Science Foundation of China under Grant 62176133, and in part by the Tsinghua-Guoqiang research program under Grant 2019GQG0006.

## Impact Statement

This paper presents work that aims to advance our understanding of the feature learning process of neural networks and its impact on generalization under distribution shifts, which may benefit building machine learning models that are more generalizable, robust, and trustworthy.

## References

Abbe, E., Bengio, S., Lotfi, A., and Rizk, K. Generalization on the unseen, logic reasoning and degree curriculum. In *International Conference on Machine Learning*, 2023.

Ahuja, K., Caballero, E., Zhang, D., Bengio, Y., Mitliagkas, I., and Rish, I. Invariance principle meets information bottleneck for out-of-distribution generalization. In *Advances in Neural Information Processing Systems*, pp. 3438–3450, 2021a.

Ahuja, K., Wang, J., Dhurandhar, A., Shanmugam, K., and Varshney, K. R. Empirical or invariant risk minimization? A sample complexity perspective. In *ICLR*, 2021b.

Allen-Zhu, Z. and Li, Y. Feature purification: How adversarial training performs robust deep learning. *arXiv preprint arXiv:2005.10190*, 2021.

Allen-Zhu, Z. and Li, Y. Physics of language models: Part 3.1, knowledge storage and extraction. *arXiv preprint arXiv:2309.14316*, 2023a.

Allen-Zhu, Z. and Li, Y. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. In *International Conference on Learning Representations*, 2023b.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., and Lacoste-Julien, S. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pp. 233–242, 2017.

Barbu, A., Mayo, D., Alverio, J., Luo, W., Wang, C., Gutfreund, D., Tenenbaum, J., and Katz, B. ObjectNet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *Advances in Neural Information Processing Systems*, pp. 9448–9458, 2019.

Beery, S., Van Horn, G., and Perona, P. Recognition in terra incognita. In *ECCV*, volume 11220, pp. 472–489, 2018.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P. W., Krass, M., Krishna, R., Kuditipudi, R., Kumar, A., Ladhak, F., Lee, M., Lee, T., Leskovec, J., Levent, I., Li, X. L., Li, X., Ma, T., Malik, A., Manning, C. D., Mirchandani, S., Mitchell, E., Munyikwa, Z., Nair, S., Narayan, A., Narayanan, D., Newman, B., Nie, A., Niebles, J. C., Nilforoshan, H., Nyarko, J., Ogut, G., Orr, L., Papadimitriou, I., Park, J. S., Piech, C., Portelance, E., Potts, C., Raghunathan, A., Reich, R., Ren, H., Rong, F., Roohani, Y., Ruiz, C., Ryan, J., Ré, C., Sadigh, D., Sagawa, S., Santhanam, K., Shih, A., Srinivasan, K., Tamkin, A., Taori, R., Thomas, A. W., Tramèr, F., Wang, R. E., Wang, W., Wu, B., Wu, J., Wu, Y., Xie, S. M., Yasunaga, M., You, J., Zaharia, M., Zhang, M., Zhang, T., Zhang, X., Zhang, Y., Zheng, L., Zhou, K., and Liang, P. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2022.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

Chattopadhay, A., Sarkar, A., Howlader, P., and Balasubramanian, V. N. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pp. 839–847. IEEE, 2018.

Chen, Y., Rosenfeld, E., Sellke, M., Ma, T., and Risteski, A. Iterative feature matching: Toward provable domain generalization with logarithmic environments. In *Advances in Neural Information Processing Systems*, 2022.

Chen, Y., Huang, W., Zhou, K., Bian, Y., Han, B., and Cheng, J. Understanding and improving feature learning for out-of-distribution generalization. In *Advances in Neural Information Processing Systems*, 2023.

DeGrave, A. J., Janizek, J. D., and Lee, S.-I. AI for radiographic COVID-19 detection selects shortcuts over signal. *Nature Machine Intelligence*, 3(7):610–619, 2021. ISSN 2522-5839.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255, 2009.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., and Chen, C. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*, 2019.

Gandelsman, Y., Efros, A. A., and Steinhardt, J. Interpreting CLIP's image representation via text-based decomposition. In *International Conference on Learning Representations*, 2024.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016.

Geirhos, R., Temme, C. R. M., Rauber, J., Schütt, H. H., Bethge, M., and Wichmann, F. A. Generalisation in humans and deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 7549–7561, 2018.

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F. A. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020. ISSN 2522-5839.

Gould, R., Ong, E., Ogden, G., and Conmy, A. Successor heads: Recurring, interpretable attention heads in the wild. *arXiv preprint arXiv:2312.09230*, 2023.

Gulrajani, I. and Lopez-Paz, D. In search of lost domain generalization. In *ICLR*, 2021.

Gurnee, W. and Tegmark, M. Language models represent space and time. In *International Conference on Learning Representations*, 2024.

HaoChen, J. Z., Wei, C., Kumar, A., and Ma, T. Beyond separability: Analyzing the linear transferability of contrastive representations to related subpopulations. *arXiv preprint arXiv:2204.02683*, 2022.

Heinzerling, B. and Inui, K. Monotonic representation of numeric properties in language models. *arXiv preprint arXiv:2403.10381*, 2024.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., Song, D., Steinhardt, J., and Gilmer, J. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *ICCV*, 2021a.

Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., and Song, D. Natural adversarial examples. In *CVPR*, 2021b.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. In *Advances in Neural Information Processing Systems Deep Learning Workshop*, 2014.

Huang, Z., Wang, H., Xing, E. P., and Huang, D. Self-challenging improves cross-domain generalization. In *European Conference on Computer Vision*, pp. 124–140, 2020.

Idnani, D., Madan, V., Goyal, N., Schwab, D. J., and Vedantam, S. R. Don't forget the nullspace! Nullspace occupancy as a mechanism for out of distribution failure. 2023.

Kamath, P., Tangella, A., Sutherland, D. J., and Srebro, N. Does invariant risk minimization capture invariance? In *AISTATS*, 2021.

Karp, S., Winston, E., Li, Y., and Singh, A. Local signal adaptivity: Provable feature learning in neural networks beyond kernels. In *Advances in Neural Information Processing Systems*, pp. 24883–24897, 2021.

Kim, D., Yoo, Y., Park, S., Kim, J., and Lee, J. SelfReg: Self-supervised contrastive regularization for domain generalization. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9599–9608, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., Lee, T., David, E., Stavness, I., Guo, W., Earnshaw, B. A., Haque, I. S., Beery, S., Leskovec, J., Kundaje, A., Pierson, E., Levine, S., Finn, C., and Liang, P. Wilds: A benchmark of in-the-wild distribution shifts. In *ICML*, 2021.

Krueger, D., Caballero, E., Jacobsen, J.-H., Zhang, A., Binas, J., Zhang, D., Priol, R. L., and Courville, A. Out-of-distribution generalization via risk extrapolation (rex). In *ICML*, 2021.

Kumar, A., Raghunathan, A., Jones, R., Ma, T., and Liang, P. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022.

Lee, Y., Chen, A. S., Tajwar, F., Kumar, A., Yao, H., Liang, P., and Finn, C. Surgical fine-tuning improves adaptation to distribution shifts. In *International Conference on Learning Representations*, 2023.

Li, C. and Flanigan, J. Task contamination: Language models may not be few-shot anymore. *arXiv preprint arXiv:2312.16337*, 2023.

Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. M. Deeper, broader and artier domain generalization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5543–5551, 2017.

Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. M. Learning to generalize: Meta-learning for domain generalization. In *AAAI*, 2018.

Liu, H., Ning, R., Teng, Z., Liu, J., Zhou, Q., and Zhang, Y. Evaluating the logical reasoning ability of chatgpt and gpt-4. *arXiv preprint arXiv:2304.03439*, 2023.

Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Marks, S. and Tegmark, M. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. *arXiv preprint arXiv:2310.06824*, 2023.

Mayilvahanan, P., Wiedemer, T., Rusak, E., Bethge, M., and Brendel, W. Does CLIP's generalization performance mainly stem from high train-test similarity? In *International Conference on Learning Representations*, 2024.

Miller, J., Taori, R., Raghunathan, A., Sagawa, S., Koh, P. W., Shankar, V., Liang, P., Carmon, Y., and Schmidt, L. Accuracy on the line: On the strong correlation between out-of-distribution and in-distribution generalization. In *ICML*, 2021.

Mitrovic, J., McWilliams, B., Walker, J. C., Buesing, L. H., and Blundell, C. Representation learning via invariant causal mechanisms. In *ICLR*, 2021.

Morcos, A. S., Barrett, D. G. T., Rabinowitz, N. C., and Botvinick, M. On the importance of single directions for generalization. In *International Conference on Learning Representations*, 2018. URL http://arxiv.org/abs/1803.06959.

Nagarajan, V., Andreassen, A., and Neyshabur, B. Understanding the failure modes of out-of-distribution generalization. In *ICLR*, 2021.

Nam, H., Lee, H., Park, J., Yoon, W., and Yoo, D. Reducing domain gap by reducing style bias. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8690–8699, 2021.

Oliveira, R. I. Concentration of the adjacency matrix and of the laplacian in random graphs with independent edges. *arXiv preprint arXiv:0911.0600*, 2010.

Park, K., Choe, Y. J., and Veitch, V. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Wang, B. Moment matching for multi-source domain adaptation. In *ICCV*, pp. 1406–1415, 2019.

Pezeshki, M., Kaba, S.-O., Bengio, Y., Courville, A., Precup, D., and Lajoie, G. Gradient starvation: A learning proclivity in neural networks. In *Advances in Neural Information Processing Systems*, pp. 1256–1272, 2021.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763, 2021.

Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. Do imagenet classifiers generalize to imagenet? In *ICML*, 2019.

Rosenfeld, E., Ravikumar, P., and Risteski, A. The risks of invariant risk minimization. In *ICLR*, 2021.

Rosenfeld, E., Ravikumar, P., and Risteski, A. Domain-adjusted regression or: ERM may already learn features sufficient for out-of-distribution generalization. *arXiv preprint arXiv:2202.06856*, 2022.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *ICLR*, 2020a.

Sagawa, S., Raghunathan, A., Koh, P. W., and Liang, P. An investigation of why overparameterization exacerbates spurious correlations. In *ICML*, 2020b.

Salman, H., Li, J., Razenshteyn, I., Zhang, P., Zhang, H., Bubeck, S., and Yang, G. Provably robust deep learning via adversarially trained smoothed classifiers. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Schölkopf, B., Locatello, F., Bauer, S., Ke, N. R., Kalchbrenner, N., Goyal, A., and Bengio, Y. Toward causal representation learning. *Proceedings of the IEEE*, 109(5): 612–634, 2021. ISSN 1558-2256.

Shah, H., Tamuly, K., and Raghunathan, A. The pitfalls of simplicity bias in neural networks. In *Advances in Neural Information Processing Systems*, 2020.

Shen, K., Jones, R., Kumar, A., Xie, S. M., HaoChen, J. Z., Ma, T., and Liang, P. Connect, not collapse: Explaining contrastive learning for unsupervised domain adaptation. In *International Conference on Machine Learning*, volume 19847-19878, 2022.

Shi, Y., Seely, J., Torr, P. H. S., Siddharth, N., Hannun, A., Usunier, N., and Synnaeve, G. Gradient matching for domain generalization. In *International Conference on Learning Representations*, 2022.

Singh, M., Gustafson, L., Adcock, A., De Freitas Reis, V., Gedik, B., Kosaraju, R. P., Mahajan, D., Girshick, R., Dollar, P., and Van Der Maaten, L. Revisiting weakly supervised pre-training of visual perception models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 794–804, 2022.

Sun, B. and Saenko, K. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pp. 443–450, 2016.

Tan, S., Peng, X., and Saenko, K. Class-imbalanced domain adaptation: An empirical odyssey. *arXiv preprint arXiv:1910.10320*, 2020.

Taori, R., Dave, A., Shankar, V., Carlini, N., Recht, B., and Schmidt, L. Measuring robustness to natural distribution shifts in image classification. In *Advances in Neural Information Processing Systems*, pp. 18583–18599, 2020.

Tian, Y., Krishnan, D., and Isola, P. Contrastive representation distillation. In *International Conference on Learning Representations*, 2020.

Torralba, A. and Efros, A. A. Unbiased look at dataset bias. In *CVPR*, pp. 1521–1528, 2011.

Tripuraneni, N., Jordan, M. I., and Jin, C. On the theory of transfer learning: The importance of task diversity. In *Advances in Neural Information Processing Systems*, pp. 7852–7862, 2020.

Tropp, J. A. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12: 389–434, 2012.

Vapnik, V. *The nature of statistical learning theory*. 1999.

Wang, H., Ge, S., Xing, E. P., and Lipton, Z. C. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems*, pp. 10506–10518, 2019.

Wang, H., Si, H., Li, B., and Zhao, H. Provable domain generalization via invariant-feature subspace recovery. In *International Conference on Machine Learning*, pp. 23018–23033, 2022.

Wiles, O., Gowal, S., Stimberg, F., Rebuffi, S.-A., Ktena, I., Dvijotham, K., and Cemgil, T. A fine-grained analysis on distribution shift. In *ICLR*, 2022.

Wortsman, M., Ilharco, G., Kim, J. W., Li, M., Kornblith, S., Roelofs, R., Lopes, R. G., Hajishirzi, H., Farhadi, A., Namkoong, H., and Schmidt, L. Robust fine-tuning of zero-shot models. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7949–7961, 2022.

Xu, K., Li, J., Zhang, M., Du, S. S., Kawarabayashi, K.-i., and Jegelka, S. How neural networks extrapolate: From feedforward to graph neural networks. In *ICLR*, 2021.

Ye, H., Xie, C., Cai, T., Li, R., Li, Z., and Wang, L. Towards a theoretical framework of out-of-distribution generalization. In *Advances in Neural Information Processing Systems*, 2021.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. Mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

Zhang, J. and Bottou, L. Learning useful representations for shifting tasks and distributions. In *International Conference on Machine Learning*, 2023.

Zhang, M., Marklund, H., Dhawan, N., Gupta, A., Levine, S., and Finn, C. Adaptive risk minimization: Learning to adapt to domain shift. In *Advances in Neural Information Processing Systems*, pp. 23664–23678, 2021.

# APPENDIX

The appendix is divided into two parts for readability. In Appendix I, we provide complete proofs of our theoretical results. In Appendix II, we present experimental details and additional empirical results.

# APPENDIX I: PROOFS OF THEORETICAL RESULTS

In this part of the appendix, we provide complete proofs of our theorems in the main text. A quick overview of the structure of this part is as follows:

- In Section A, we introduce the preliminaries and some lemmas that characterize the neuron properties at random initialization.
- In Section B, we provide the proofs of our main theorems on activation asymmetry (Theorem 4.1), feature contamination (Theorem 4.2), and ID/OOD risks (Theorem 4.3).
- In Section C, we provide the proof of Theorem 4.4 on linear neural networks.
- In Section D, we provide the basic probability theory lemmas used in our proofs for completeness.

## A. Preliminaries

**Notation.** Throughout the appendix, we overload $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ to allow them to denote (joint) training and test distributions on both $\mathcal{X} \times \mathcal{Y}$ and $\mathcal{Z} \times \mathcal{Y}$. We also use $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ to denote the corresponding marginal distributions on $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$. For presentation brevity, unless otherwise stated, we use the shorthand $\mathbb{E}_{(\cdot)}$ and $\mathbf{Pr}_{(\cdot)}$ to denote $\mathbb{E}_{(\cdot)\sim\mathcal{D}_{\text{train}}}$ and $\mathbf{Pr}_{(\cdot)\sim\mathcal{D}_{\text{train}}}$, respectively, and use the shorthand $h$ to denote $h^{(t)}$ when it is clear from the context. As in Definition 3.1, we denote the moments of each $\mathbf{z}_j$ on the training distribution by $\mu_{jp} := \mathbb{E}_{\mathbf{z}\sim\mathcal{D}_{\text{train}}}\mathbf{z}_j^p$ for every $j \in [d_0]$ and $p \in [3]$, and use the shorthand $\mu_j$ to denote $\mu_{j1}$ when it is clear from the context.

### A.1. Weight Decomposition and Gradient Calculations

We begin by recalling that each weight vector $\mathbf{w}_k \in \mathbb{R}^d, k \in [m]$ (i.e., the learned feature of the $k$-th neuron) in the network can be decomposed into the sum of its projections to different feature vectors:

$$\mathbf{w}_k^{(t)} = \sum_{j\in\mathcal{S}_{\text{core}}} \langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j \rangle \boldsymbol{m}_j + \sum_{j\in\mathcal{S}_{\text{bg}}} \langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j \rangle \boldsymbol{m}_j + \sum_{j\in[d]\setminus[d_0]} \langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j \rangle \boldsymbol{m}_j, \tag{10}$$

where $(\boldsymbol{m}_{d_0+1}, \dots, \boldsymbol{m}_d)$ are an orthogonal complement of $\boldsymbol{M}$. Since all possible inputs are generated to be in $\operatorname{span}\{\boldsymbol{m}_1, \dots, \boldsymbol{m}_{d_0}\}$ as in Definition 3.1, the last term in the RHS of Eq. (10) (i.e., the residual term in Eq. (6) in the main text) can be neglected due to the orthogonality of different feature vector $\boldsymbol{m}_j$s. Therefore, throughout the following analysis, we will overload the notation $\mathbf{w}_k^{(t)}$ and let

$$\mathbf{w}_k^{(t)} = \sum_{j\in\mathcal{S}_{\text{core}}} \langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j \rangle \boldsymbol{m}_j + \sum_{j\in\mathcal{S}_{\text{bg}}} \langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j \rangle \boldsymbol{m}_j. \tag{11}$$

A direct consequence of Eq. (10) is that we can analyze the feature learning process of the network by tracking the correlations between each weight vector $\mathbf{w}_k^{(t)}$ and different feature vectors $\boldsymbol{m}_j, j \in [d]$ as the training proceeds. To this end, we need to first analyze the gradient of each neuron at every iteration.

**Gradient of each neuron.** Recall that at each iteration $t = 0, \dots, T-1$, the SGD update for each weight vector $\mathbf{w}_k, k \in [m]$ is given by

$$\mathbf{w}_k^{(t+1)} \leftarrow (1-\eta\lambda)\mathbf{w}_k^{(t)} - \eta\nabla_{\mathbf{w}_k^{(t)}} \frac{1}{N} \sum_{i\in[N]} \ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right),$$

where

$$h^{(t)}(\mathbf{x}_i^{(t)}) = \sum_{k\in[m]} a_k \cdot \mathsf{ReLU}\big(\langle \mathbf{w}_k^{(t)}, \mathbf{x}_i^{(t)}\rangle\big)$$

and $\ell(y, y') = \max\{1 - yy', 0\}$. We can then calculate the gradient of each neuron $\mathbf{w}_k^{(t)}$ with regard to a certain example $(\mathbf{x}, \mathbf{y})$:

**Lemma A.1** (Gradient). *For every example $(x, y) \in \mathcal{X} \times \mathcal{Y}$, every $k \in [m]$, and every iteration $t$, the following holds:*

$$\nabla_{\mathbf{w}_k^{(t)}} \ell\left(h(x), y\right) = -a_k y \mathbf{1}_{h(x)\le 1} \mathbf{1}_{\langle \mathbf{w}_k^{(t)}, x\rangle \ge 0} x. \tag{12}$$

*Proof.* The proof follows from simple calculation. □

We then introduce a lemma that bounds the empirical growth of the correlation between each neuron $\mathbf{w}_k^{(t)}$ and each feature vector $\boldsymbol{m}_j$ after an SGD update using population gradients.

**Lemma A.2** (Gap between empirical and population gradients). *For every $k \in [m]$, every $j \in [d]$, and every iteration $t$, if the batch size $N = \mathsf{poly}(d)$ for some sufficiently large polynomial, then the following holds with probability at least $1 - e^{-\Omega(d)}$:*

$$\left|\left\langle \nabla_{\mathbf{w}_k^{(t)}} \frac{1}{N} \sum_{i\in[N]} \ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right), \boldsymbol{m}_j \right\rangle - \left\langle \nabla_{\mathbf{w}_k^{(t)}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\text{train}}} \ell\left(h(\mathbf{x}), \mathbf{y}\right), \boldsymbol{m}_j \right\rangle\right| \le \frac{1}{\mathsf{poly}(d)}. \tag{13}$$

*Proof.* Recall that $\|\boldsymbol{m}_j\|_2 = 1$. Applying Cauchy-Schwarz inequality gives

$$\begin{aligned}&\left|\left\langle \nabla_{\mathbf{w}_k^{(t)}} \frac{1}{N} \sum_{i\in[N]} \ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right), \boldsymbol{m}_j \right\rangle - \left\langle \nabla_{\mathbf{w}_k^{(t)}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\text{train}}} \ell\left(h(\mathbf{x}), \mathbf{y}\right), \boldsymbol{m}_j \right\rangle\right| \\ &\le \underbrace{\left\| \nabla_{\mathbf{w}_k^{(t)}} \frac{1}{N} \sum_{i\in[N]} \ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right) - \nabla_{\mathbf{w}_k^{(t)}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\text{train}}} \ell\left(h(\mathbf{x}), \mathbf{y}\right) \right\|_2}_{\mathbf{S}^{(t)}}.\end{aligned}$$

We define

$$\mathbf{Z}_i^{(t)} := \frac{1}{N} \nabla_{\mathbf{w}_k^{(t)}} \ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right) - \frac{1}{N} \nabla_{\mathbf{w}_k^{(t)}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\text{train}}} \ell\left(h(\mathbf{x}), \mathbf{y}\right),\ \forall i \in [N].$$

It is easy to see that $\mathbf{S}^{(t)} = \sum_{i\in[N]} \mathbf{Z}_i^{(t)}$, $\mathbb{E}\mathbf{Z}_i^{(t)} = 0$ for every $i \in [N]$, and $\forall i \ne j \in [N]$, $\mathbf{Z}_i^{(t)}$ and $\mathbf{Z}_j^{(t)}$ are independent. By Lemma A.1, we have

$$\mathbf{Z}_i^{(t)} = \frac{1}{N} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\text{train}}} a_k \mathbf{y} \mathbf{1}_{h(\mathbf{x})\le 1} \mathbf{1}_{\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \ge 0} \cdot \mathbf{x} - \frac{1}{N} a_k \mathbf{y}_i^{(t)} \mathbf{1}_{h^{(t)}(\mathbf{x}_i^{(t)})\le 1} \mathbf{1}_{\langle \mathbf{w}_k^{(t)}, \mathbf{x}_i^{(t)}\rangle \ge 0} \cdot \mathbf{x}_i^{(t)}.$$

Recall that $a_k \in \{-\frac{1}{m}, \frac{1}{m}\}$ and $\mathbf{x}$ is generated by $\mathbf{x} = \sum_{j\in\mathcal{S}_{\text{core}}} \mathbf{y}\mathbf{z}_j \boldsymbol{m}_j + \sum_{j\in\mathcal{S}_{\text{bg}}} \mathbf{z}_j \boldsymbol{m}_j$ according to Definition 3.1. We then have $\|\mathbf{Z}_i^{(t)}\|_2 \le \frac{2\sqrt{d_0}}{mN}$, which also indicates that $\mathbb{E}\langle \mathbf{Z}_i^{(t)}, \mathbf{Z}_i^{(t)}\rangle \le \frac{4d_0}{m^2N^2}$. This gives

$$\mathbb{E}\langle \mathbf{S}^{(t)}, \mathbf{S}^{(t)}\rangle = \sum_{i\in[N]} \mathbb{E}\langle \mathbf{Z}_i^{(t)}, \mathbf{Z}_i^{(t)}\rangle \le \frac{4d_0}{m^2 N}.$$

Applying matrix Bernstein's inequality (Lemma D.2), we have

$$\mathbf{Pr}\left[\|\mathbf{S}^{(t)}\|_2 \ge \delta\right] \le (d+1)\exp\left(-\frac{3m^2N^2\delta^2}{24d_0 + 4\sqrt{d_0}\delta m N}\right)$$

hold with every $\delta = \frac{1}{\mathsf{poly}(d)}$. Therefore, we have that for $N = \mathsf{poly}(d)$ with some sufficiently large polynomial, the following holds with probability at least $1 - e^{-\Omega(d)}$:

$$\|\mathbf{S}^{(t)}\|_2 \le \frac{1}{\mathsf{poly}(d)}.$$

This gives the desired result. □

Lemma A.2 directly leads to the following corollary:

**Lemma A.3.** *For every $k \in [m]$, every $j \in [d_0]$, and every iteration $t$, the following holds:*

$$\begin{aligned}
&\left\langle \nabla_{\mathbf{w}_k^{(t)}} \frac{1}{N} \sum_{i\in[N]} \ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right), \boldsymbol{m}_j \right\rangle \\
&= \left\langle \nabla_{\mathbf{w}_k^{(t)}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\text{train}}} \ell\left(h(\mathbf{x}), \mathbf{y}\right), \boldsymbol{m}_j \right\rangle \pm \frac{1}{\mathsf{poly}(d)} \\
&= -a_k \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\text{train}}} \mathbf{y} \mathbf{1}_{h(\mathbf{x})\le 1} \mathbf{1}_{\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \ge 0} \cdot \mathbf{z}_j \pm \frac{1}{\mathsf{poly}(d)}, \quad j \in [d_0].
\end{aligned} \tag{14}$$

*Proof.* The proof directly follows from combining Lemma A.1 and the generation process of $\mathbf{x}$ in Definition 3.1. □

Lemma A.3 allows us to directly work with population gradients instead of empirical gradients when analyzing the trajectory of SGD iterations in the subsequent sections.

### A.2. Neuron Characterization

In this section, we define two subsets of neurons that will be used throughout our proofs.

**Definition A.4** (Neuron characterization)**.** For each label $y \in \mathcal{Y} = \{-1, 1\}$ and every iteration $t$, we define the set $\mathcal{N}_y^{(t)} \subseteq [m]$ as:

$$\begin{aligned}
\mathcal{N}_y^{(t)} := \Bigg\{ k \in [m] : &\sum_{j\in\mathcal{S}_{\text{core}}} y\mu_j \langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle + \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle \ge \Theta\left(\sqrt{\frac{d_0}{d}}\right), \\
&\text{sign}\left(a_k\right) = y \Bigg\}.
\end{aligned} \tag{15}$$

**Intuition.** For each label $y \in \mathcal{Y}$ and iteration $t$, Definition A.4 characterizes a subset of neurons $\mathcal{N}_y^{(t)}$ in which

- each neuron has (in expectation) enough positive correlations with the examples from class $y$ (recall that $\mathbf{x} = \sum_{j\in\mathcal{S}_{\text{core}}} \mathbf{y}\mathbf{z}_j \boldsymbol{m}_j + \sum_{j\in\mathcal{S}_{\text{bg}}} \mathbf{z}_j \boldsymbol{m}_j$);
- each neuron positively contributes to the classification of examples from class $y$ (i.e., sign $(a_k) = y$).

In our main proof, we will show in an iterative fashion that each neuron in $\mathcal{N}_y^{(t)}$ will accumulate either positive (if random initialization gives $a_k = \frac{1}{m}$) or negative (if random initialization gives $a_k = -\frac{1}{m}$) correlations with features in $\mathcal{S}_{\text{core}}$ (*core feature learning*), while also accumulating positive correlations with features in $\mathcal{S}_{\text{bg}}$ (*feature contamination*).

For each neuron, we formally define the notion of *positive examples* and *negative examples* which are informally mentioned in Section 4:

**Definition A.5** (Positive examples and negative examples)**.** Let $(x, y) \in \mathcal{X} \times \mathcal{Y}$ be an example. For every $k \in [m]$, we say that $(x, y)$ is a **positive example** of neuron $k$ if $\text{sign}(a_k) = y$, and say that $(x, y)$ is a **negative example** of neuron $k$ if $\text{sign}(a_k) = -y$.

### A.3. Properties at Initialization

In this section, we introduce some useful properties of the neurons at initialization $t = 0$, which serve as a basis for our inductive proofs in the subsequent sections.

**Lemma A.6.** *For every $j \in [d_0]$, every $\mathcal{S} \subseteq [d_0]$, and every $\{y_j\}_{j\in\mathcal{S}} \in \{-1, 1\}^{|\mathcal{S}|}$, the following holds for every $\delta > 0$*

*over random initialization:*

$$\mathbf{Pr}_{\mathbf{W}^{(0)}}\Big[\sum_{j\in\mathcal{S}} y_j\mu_j\langle \mathbf{w}_k^{(0)}, \boldsymbol{m}_j\rangle \geq \frac{\delta}{\sqrt{d}}\Big] \geq \frac{1}{\sqrt{2\pi}}\frac{\delta\sqrt{\sum_{j\in\mathcal{S}}\mu_j^2}}{\delta^2+\sum_{j\in\mathcal{S}}\mu_j^2}\exp\left(-\frac{\delta^2}{2\sum_{j\in\mathcal{S}}\mu_j^2}\right),$$
$$\mathbf{Pr}_{\mathbf{W}^{(0)}}\Big[\sum_{j\in\mathcal{S}} y_j\mu_j\langle \mathbf{w}_k^{(0)}, \boldsymbol{m}_j\rangle \geq \frac{\delta}{\sqrt{d}}\Big] \leq \frac{1}{\sqrt{2\pi}}\frac{\sqrt{\sum_{j\in\mathcal{S}}\mu_j^2}}{\delta}\exp\left(-\frac{\delta^2}{2\sum_{j\in\mathcal{S}}\mu_j^2}\right). \tag{16}$$

*Proof.* Recall that different neurons are independently initialized by $\mathbf{w}_k^{(0)} \sim \mathcal{N}(\mathbf{0}, \sigma_0^2\boldsymbol{I}_d), \forall k\in[m]$ with $\sigma_0^2 = \frac{1}{d}$. Using the fact that $\|\boldsymbol{m}_j\|_2 = 1, \forall j\in[d_0]$ and $y_j^2 = 1, \forall j\in\mathcal{S}$, we have

$$\sum_{j\in\mathcal{S}} y_j\mu_j\langle \mathbf{w}_k^{(0)}, \boldsymbol{m}_j\rangle \sim \mathcal{N}\Big(0,\ \frac{1}{d}\sum_{j\in\mathcal{S}}\mu_j^2\Big)$$

Applying standard bounds for the Gaussian distribution function (Lemma D.3) gives that for every $\delta > 0$,

$$\frac{1}{\sqrt{2\pi}}\frac{\delta}{\delta^2+1}\exp\left(-\frac{\delta^2}{2}\right) \leq \mathbf{Pr}_{\mathbf{W}^{(0)}}\left[\frac{\sqrt{d}\sum_{j\in\mathcal{S}} y_j\mu_j\langle \mathbf{w}_k^{(0)}, \boldsymbol{m}_j\rangle}{\sqrt{\sum_{j\in\mathcal{S}}\mu_j^2}} \geq \delta\right] \leq \frac{1}{\sqrt{2\pi}}\frac{1}{\delta}\exp\left(-\frac{\delta^2}{2}\right).$$

A simple transformation completes the proof. □

**Lemma A.7** (Neuron properties at initialization)**.** *For each label $y \in \mathcal{Y}$, the following holds with probability at least $1 - e^{-\Omega(m)}$ over random initialization:*

$$\big|\mathcal{N}_y^{(0)}\big| = \Theta(m). \tag{17}$$

*Proof.* For each neuron $k\in[m]$, define events $E_{k1}$ and $E_{k2}$ to be

$$E_{k1} := \left\{\sum_{j\in\mathcal{S}_{\text{core}}} y\mu_j\langle \mathbf{w}_k^{(0)}, \boldsymbol{m}_j\rangle + \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j\langle \mathbf{w}_k^{(0)}, \boldsymbol{m}_j\rangle \geq \Theta\left(\sqrt{\frac{d_0}{d}}\right)\right\},$$
$$E_{k2} := \Big\{\operatorname{sign}(a_k) = y\Big\}.$$

By $a_k \sim \mathsf{Uniform}\{-\frac{1}{m}, \frac{1}{m}\}$, we immediately have $\mathbf{Pr}[E_{k2}] = \frac{1}{2}$ for every $k\in[m]$. For $E_{k1}$, by applying Lemma A.6 with $\delta = \Theta(\sqrt{d_0})$ we obtain

$$\mathbf{Pr}_{\mathbf{W}^{(0)}}[E_{k1}] \geq \frac{1}{\sqrt{2\pi}}\frac{\Theta\left(\sqrt{d_0\sum_{j\in[d_0]}\mu_j^2}\right)}{\Theta(d_0)+\sum_{j\in[d_0]}\mu_j^2}\exp\left(-\Theta\left(\frac{d_0}{\sum_{j\in[d_0]}\mu_j^2}\right)\right),$$
$$\mathbf{Pr}_{\mathbf{W}^{(0)}}[E_{k1}] \leq \frac{1}{\sqrt{2\pi}}\Theta\left(\sqrt{\frac{\sum_{j\in[d_0]}\mu_j^2}{d_0}}\right)\exp\left(-\Theta\left(\frac{d_0}{\sum_{j\in[d_0]}\mu_j^2}\right)\right).$$

Together with $\mu_j^2 = \Theta(1)$ for every $j\in[d_0]$, we have that $\mathbf{Pr}_{\mathbf{W}^{(0)}}[E_{k1}] = \Theta(1)$ for every $k\in[m]$. Since events $E_{k1}$ and $E_{k2}$ are independent, we have that for each neuron $k\in[m]$, the probability of it belonging to $\mathcal{N}_y^{(0)}$ is given by $\mathbf{Pr}(k\in\mathcal{N}_y^{(0)}) = \mathbf{Pr}(E_{k1}\cap E_{k2}) = \Theta(1)$.

Since different neurons are independently initialized, $|\mathcal{N}_y^{(0)}|$ follows a binomial distribution with trial number $m$ and some success probability $\Theta(1)$. Applying the standard tail bound for binomial variables (Lemma D.4) then gives $|\mathcal{N}_y^{(0)}| \geq \Theta(m)$ with probability at least $1-e^{-\Omega(m)}$. Together with the trivial upper bound that $|\mathcal{N}_y^{(0)}| \leq m$, we have that $|\mathcal{N}_y^{(0)}| = \Theta(m)$ with probability at least $1-e^{-\Omega(m)}$. □

**Lemma A.8** (Neuron properties at initialization, continued)**.** *With probability at least $1 - O(\frac{1}{m})$ over random initialization, for every $y \in \mathcal{Y}$, the following holds:*

$$\max_{k\in[m]} \left|\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}\langle \mathbf{w}_k^{(0)}, \mathbf{x}\rangle\right| \le O\left(\sqrt{\frac{d_0 \log m}{d}}\right).$$

*Proof.* Recall that different neurons are independently initialized by $\mathbf{w}_k^{(0)} \sim \mathcal{N}(\mathbf{0}, \sigma_0^2 \boldsymbol{I}_d), \forall k \in [m]$ with $\sigma_0^2 = \frac{1}{d}$. By $\|\boldsymbol{m}_j\|_2 = 1$, we have

$$\begin{aligned}\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}\langle \mathbf{w}_k^{(0)}, \mathbf{x}\rangle &= \sum_{j\in\mathcal{S}_{\text{core}}} y\mu_j \langle \mathbf{w}_k^{(0)}, \boldsymbol{m}_j\rangle + \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \langle \mathbf{w}_k^{(0)}, \boldsymbol{m}_j\rangle \\ &\sim \mathcal{N}\Big(0, \ \frac{1}{d}\sum_{j\in[d_0]} \mu_j^2\Big).\end{aligned}$$

Applying Lemma D.5 over the i.i.d. random variables $\langle \mathbf{w}_1^{(0)}, \mathbf{x}\rangle, \dots, \langle \mathbf{w}_m^{(0)}, \mathbf{x}\rangle$ gives

$$\mathbf{Pr}_{\mathbf{W}^{(0)}}\left[\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}\langle \mathbf{w}_k^{(0)}, \mathbf{x}\rangle \ge 2\sqrt{\frac{\sum_{j\in[d_0]}\mu_j^2}{d}\log m}\right] \le \frac{1}{m}.$$

Finally, using $\sum_{j\in[d_0]}\mu_j^2 = \Theta(d_0)$ and $m \in [\Theta(d_0), \Theta(d)]$ completes the proof. □

**Lemma A.9** (Output magnitude at initialization)**.** *For every $x \in \mathcal{X}$, the following holds with probability at least $1 - e^{-\Omega(d_0)}$ over random initialization:*

$$\left|h^{(0)}(x)\right| = O\left(\frac{1}{\sqrt{d_0}}\right). \tag{18}$$

*Proof.* By $\mathbf{w}_k^{(0)} \sim \mathcal{N}(\mathbf{0}, \sigma_0^2 \boldsymbol{I}_d)$ with $\sigma_0^2 = \frac{1}{d}$ and $\|\boldsymbol{m}_j\|_2 = 1$, we have

$$\sum_{k\in[m]} \frac{1}{m} \sum_{j\in[d_0]} \langle \mathbf{w}_k^{(0)}, \boldsymbol{m}_j\rangle \sim \mathcal{N}\Big(0, \frac{d_0}{md}\Big).$$

Applying standard bounds for the Gaussian distribution function (Lemma D.3) gives

$$\frac{1}{\sqrt{2\pi}}\frac{\delta}{\delta^2+1}\exp\left(-\frac{\delta^2}{2}\right) \le \mathbf{Pr}_{\mathbf{W}^{(0)}}\left[\sum_{k\in[m]} \frac{1}{m} \sum_{j\in[d_0]} \langle \mathbf{w}_k^{(0)}, \boldsymbol{m}_j\rangle \ge \delta\sqrt{\frac{d_0}{md}}\right] \le \frac{1}{\sqrt{2\pi}}\frac{1}{\delta}\exp\left(-\frac{\delta^2}{2}\right)$$

for every $\delta > 0$. Substituting $\delta$ by $\Theta(\sqrt{d_0})$ and using the symmetry of Gaussian then yield

$$\mathbf{Pr}_{\mathbf{W}^{(0)}}\left[\left|\sum_{k\in[m]} \frac{1}{m} \sum_{j\in[d_0]} \langle \mathbf{w}_k^{(0)}, \boldsymbol{m}_j\rangle\right| \ge \frac{\Theta(d_0)}{\sqrt{md}}\right] \le \exp(-\Omega(d_0)).$$

We then have

$$\begin{aligned}\left|h^{(0)}(x)\right| &= \left|\sum_{k\in[m]} a_k \cdot \mathsf{ReLU}\big(\langle \mathbf{w}_k^{(0)}, \mathbf{x}\rangle\big)\right| \\ &\le \left|\sum_{k\in[m]} \frac{1}{m}\langle \mathbf{w}_k^{(0)}, \mathbf{x}\rangle\right| \\ &\le \left|\sum_{k\in[m]} \frac{1}{m} \sum_{j\in[d_0]} \langle \mathbf{w}_k^{(0)}, \boldsymbol{m}_j\rangle\right| \\ &\le \frac{\Theta(d_0)}{\sqrt{md}} = O\left(\frac{1}{\sqrt{d_0}}\right).\end{aligned}$$

holds with probability at least $1 - e^{-\Omega(d_0)}$, where in the last equality we use the fact that $m = \Omega(d_0)$ and $d = \Omega(d_0^{2.5})$. □

In what follows, we will always assume that the high-probability events at initialization in Lemma A.7, Lemma A.8, and Lemma A.9 hold—by a union bound argument and the fact that $m = \Omega(d_0)$, the probability that they all hold is at least $1 - O(\frac{1}{m}) - e^{-\Omega(d_0)}$.

## B. Activation Asymmetry, Feature Contamination, and OOD Failure: Proofs of Theorem 4.1, Theorem 4.2, and Theorem 4.3

Before we delve into the main proofs, we first introduce some technical lemmas that characterize the gradient updates starting from random initialization. We begin by introducing an important lemma that characterizes the activation probability of the ReLU function using the Berry-Esseen theorem:

**Lemma B.1** (Activation probability)**.** *Assume that the training (ID) data is generated according to Definition 3.1 and* $\left|\frac{\langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_i\rangle}{\langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle}\right| = \Theta(1)$ *for every* $k \in [m]$ *and for every* $i, j \in [d_0]$*. Then, for every label* $y \in \mathcal{Y}$*, every* $k \in [m]$*, and every iteration* $t$*, the following holds:*

$$\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \geq 0] = \Phi\left(\frac{\mathbb{E}_{\mathbf{x}|\mathbf{y}=y}\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle}{\Theta(1)\sqrt{\sum_{j\in[d_0]}\langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle^2}}\right) \pm O\left(\frac{1}{\sqrt{d_0}}\right), \tag{19}$$

*where* $\Phi$ *denotes the cumulative distribution function of* $\mathcal{N}(0, 1)$*.*

*Proof.* Recall Definition 3.1 that given label $y \in \mathcal{Y}$, $\mathbf{x}$ is generated by

$$\mathbf{x} = \sum_{j\in\mathcal{S}_{\text{core}}} y\mathbf{z}_j\boldsymbol{m}_j + \sum_{j\in\mathcal{S}_{\text{bg}}} \mathbf{z}_j\boldsymbol{m}_j.$$

Therefore,

$$\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle = \sum_{j\in\mathcal{S}_{\text{core}}} y\mathbf{z}_j\langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle + \sum_{j\in\mathcal{S}_{\text{bg}}} \mathbf{z}_j\langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle.$$

For every $j \in [d_0]$, define the random variable

$$\mathbf{r}_j := y_j(\mathbf{z}_j - \mu_j)\langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle,$$

where $y_j := \begin{cases} y, \ j \in \mathcal{S}_{\text{core}} \\ 1, \ j \in \mathcal{S}_{\text{bg}} \end{cases}$. Recall that $\mu_j := \mathbb{E}_{\mathbf{z}\sim\mathcal{D}_{\text{train}}}\mathbf{z}_j$. It is then easy to derive that $\mathbb{E}\mathbf{r}_j = 0$ and $\mathbb{E}\mathbf{r}_j^2 = \Theta(1)\langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle^2$ (recall that $\mathbb{E}(\mathbf{z}_j - \mu_j)^2 = \Theta(1)$). We now upper bound $\mathbb{E}|\mathbf{r}_j^3|$: first recall that by Definition 3.1 we have $\mathbb{E}\mathbf{z}_j^3 = \Theta(1)$. For every $p \geq 1$, denote the $\ell_p$ norm of the random variable $\mathbf{z}_j$ by $\|\mathbf{z}_j\|_p := (\mathbb{E}|\mathbf{z}_j|^p)^{\frac{1}{p}}$. Applying Minkowsky's inequality gives

$$\begin{aligned}\|\mathbf{z}_j - \mu_j\|_p &\leq \|\mathbf{z}_j\|_p + \|\mu_j\|_p \\ &\overset{(a)}{=} \|\mathbf{z}_j\|_p + \|\mathbf{z}_j\|_1 \\ &\overset{(b)}{\leq} 2\|\mathbf{z}_j\|_p,\end{aligned}$$

where $(a)$ is due to the fact that $\|\mu_j\|_p = |\mu_j| = \|\mathbf{z}_j\|_1$ and $(b)$ is due to the power norm inequality indicating that $\|\cdot\|_p$ is non-decreasing with regard to $p$. Letting $p = 3$ and cubing the above inequality gives

$$\mathbb{E}|\mathbf{z}_j - \mu_j|^3 \leq 8\mathbb{E}|\mathbf{z}_j^3| = 8\mathbb{E}\mathbf{z}_j^3 = \Theta(1),$$

from which we obtain $\mathbb{E}|\mathbf{r}_j^3| = \Theta(1) \cdot |\langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle|^3$.

We then define the normalized sum of $\mathbf{r}_j$:

$$\mathbf{s}_{d_0} := \frac{\sum_{j\in[d_0]}\mathbf{r}_j}{\sqrt{\sum_{j\in[d_0]}\mathbb{E}\mathbf{r}_j^2}}.$$

Since $\mathbf{r}_i$ and $\mathbf{r}_j$ are independent and zero-mean for every $i \neq j \in [d_0]$, we can apply the Berry-Esseen theorem (Lemma D.6) to the normalized sum $\mathbf{s}_{d_0}$ and obtain

$$\begin{aligned}\sup_{\delta \in \mathbb{R}} \left| \mathbf{Pr}_{\mathbf{x}|\mathbf{y}=y}[\mathbf{s}_{d_0} < \delta] - \Phi(\delta) \right| &\le C_0 \left( \sum_{j \in [d_0]} \mathbb{E}\mathbf{r}_j^2 \right)^{-\frac{3}{2}} \sum_{j \in [d_0]} \mathbb{E}|\mathbf{r}_j^3| \\ &= C_0 \left( \sum_{j \in [d_0]} \Theta(1) \langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j \rangle^2 \right)^{-\frac{3}{2}} \sum_{j \in [d_0]} \Theta(1) \left| \langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j \rangle \right|^3 \\ &\overset{(c)}{=} O\left( \frac{1}{\sqrt{d_0}} \right),\end{aligned}$$

where $(c)$ is due to the assumption that $\left| \frac{\langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_i \rangle}{\langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j \rangle} \right| = \Theta(1)$. Note that $\sum_{j \in [d_0]} \mathbf{r}_j = \langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle - \mathbb{E}_{\mathbf{x}|\mathbf{y}=y} \langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle$. We then have for every $\delta \in \mathbb{R}$,

$$\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=y} \left[ \langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle \ge \mathbb{E}_{\mathbf{x}|\mathbf{y}=y} \langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle + \delta \sqrt{\sum_{j \in [d_0]} \mathbb{E}\mathbf{r}_j^2} \right] = 1 - \Phi(\delta) \pm O\left( \frac{1}{\sqrt{d_0}} \right).$$

Finally, letting $\delta = -\frac{\mathbb{E}_{\mathbf{x}|\mathbf{y}=y} \langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle}{\sqrt{\sum_{j \in [d_0]} \mathbb{E}\mathbf{r}_j^2}}$ and using the symmetry of unit Gaussian $1 - \Phi(\delta) = \Phi(-\delta)$ give the desired result. □

We then define two terms that will be frequently used when analyzing gradients.

**Definition B.2.** For each label $y \in \mathcal{Y}$, every $k \in [m]$, every feature vector $\boldsymbol{m}_j, j \in [d_0]$, every iteration $t$, and every subset $\mathcal{S} \subseteq [d_0]$, define

$$\begin{aligned} g_{k,y,j}^{(t)} &:= \frac{1}{m} \mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim \mathcal{D}_{\text{train}}} \mathbf{1}_{h^{(t)}(\mathbf{x}) \le 1} \mathbf{1}_{\mathbf{y}=y} \mathbf{1}_{\langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle \ge 0} \mu_j \mathbf{z}_j, \\ g_{k,y,\mathcal{S}}^{(t)} &:= \sum_{j \in \mathcal{S}} g_{k,y,j}^{(t)}. \end{aligned} \tag{20}$$

Given the above notation, we now introduce two lemmas that separately bound the gradient projection onto the core features and the gradient projection onto the background features for neurons in $\mathcal{N}_y^{(t)}$, which will be helpful for us to track the trajectory of SGD starting from network initialization.

**Lemma B.3** (Gradient projection onto core features, neurons in $\mathcal{N}_y^{(t)}$)**.** *For every iteration $t \le O(\frac{m}{\eta d_0})$, the following holds for every $y \in \mathcal{Y}$ and every neuron $k \in \mathcal{N}_y^{(t)}$:*

$$-\left\langle \nabla_{\mathbf{w}_k^{(t)}} \frac{1}{N} \sum_{i \in [N]} \ell\left( h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)} \right), \sum_{j \in \mathcal{S}_{\text{core}}} \mu_j \boldsymbol{m}_j \right\rangle = y \left( g_{k,y,\mathcal{S}_{\text{core}}}^{(t)} + g_{k,-y,\mathcal{S}_{\text{core}}}^{(t)} \right), \tag{21}$$

*where*

$$g_{k,y,\mathcal{S}_{\text{core}}}^{(t)} = \Theta\left( \frac{d_0}{m} \right). \tag{22}$$

*Proof.* Recall Definition 3.1 that given a label $\mathbf{y} \in \mathcal{Y}$, $\mathbf{x}$ is generated by

$$\mathbf{x} = \sum_{j \in \mathcal{S}_{\text{core}}} \mathbf{y}\mathbf{z}_j \boldsymbol{m}_j + \sum_{j \in \mathcal{S}_{\text{bg}}} \mathbf{z}_j \boldsymbol{m}_j.$$

Then, applying Lemma A.3 to the LHS of Eq. (21) and using $\mathrm{sign}(a_k) = y$ for every $k \in \mathcal{N}_y^{(t)}$, we obtain

$$\begin{aligned}
&-\left\langle \nabla_{\mathbf{w}_k^{(t)}} \frac{1}{N} \sum_{i\in[N]} \ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right), \sum_{j\in\mathcal{S}_{\mathrm{core}}} \mu_j \boldsymbol{m}_j \right\rangle \\
&\quad= -\left\langle \nabla_{\mathbf{w}_k^{(t)}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\mathrm{train}}} \ell\left(h^{(t)}(\mathbf{x}), \mathbf{y}\right), \sum_{j\in\mathcal{S}_{\mathrm{core}}} \mu_j \boldsymbol{m}_j \right\rangle \pm \frac{O(d_0)}{\mathsf{poly}(d)} \\
&\quad= a_k \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\mathrm{train}}} \mathbf{y} \mathbf{1}_{h^{(t)}(\mathbf{x})\le 1} \mathbf{1}_{\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \ge 0} \sum_{j\in\mathcal{S}_{\mathrm{core}}} \mathbf{y}\mu_j \mathbf{z}_j \pm \frac{O(d_0)}{\mathsf{poly}(d)} \\
&\quad= a_k \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\mathrm{train}}} \mathbf{1}_{h^{(t)}(\mathbf{x})\le 1} \mathbf{1}_{\mathbf{y}=y} \mathbf{1}_{\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \ge 0} \sum_{j\in\mathcal{S}_{\mathrm{core}}} \mu_j \mathbf{z}_j \\
&\qquad\quad + a_k \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\mathrm{train}}} \mathbf{1}_{h^{(t)}(\mathbf{x})\le 1} \mathbf{1}_{\mathbf{y}=-y} \mathbf{1}_{\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \ge 0} \sum_{j\in\mathcal{S}_{\mathrm{core}}} \mu_j \mathbf{z}_j \pm \frac{O(d_0)}{\mathsf{poly}(d)} \\
&\quad= y\left(g_{k,y,\mathcal{S}_{\mathrm{core}}}^{(t)} + g_{k,-y,\mathcal{S}_{\mathrm{core}}}^{(t)}\right) \pm \frac{O(d_0)}{\mathsf{poly}(d)}.
\end{aligned} \tag{23}$$

For $g_{k,y,\mathcal{S}_{\mathrm{core}}}^{(t)}$, by the law of total expectation we have

$$\begin{aligned}
g_{k,y,\mathcal{S}_{\mathrm{core}}}^{(t)} &= \frac{1}{m} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\mathrm{train}}} \mathbf{1}_{h^{(t)}(\mathbf{x})\le 1} \mathbf{1}_{\mathbf{y}=y} \mathbf{1}_{\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \ge 0} \sum_{j\in\mathcal{S}_{\mathrm{core}}} \mu_j \mathbf{z}_j \\
&= \frac{1}{2m} \mathbb{E}_{\mathbf{x}|\mathbf{y}=y} \left[ \mathbf{1}_{h^{(t)}(\mathbf{x})\le 1} \sum_{j\in\mathcal{S}_{\mathrm{core}}} \mu_j \mathbf{z}_j \,\middle|\, \langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \ge 0 \right] \mathbf{Pr}_{\mathbf{x}|\mathbf{y}=y}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \ge 0] \\
&= \frac{1}{2m} \mathbb{E}_{\mathbf{x}|\mathbf{y}=y} \left[ \mathbf{1}_{h^{(t)}(\mathbf{x})\le 1} \sum_{j\in\mathcal{S}_{\mathrm{core}}} \mu_j \mathbf{z}_j \right] \\
&\qquad - \frac{1}{2m} \mathbb{E}_{\mathbf{x}|\mathbf{y}=y} \left[ \mathbf{1}_{h^{(t)}(\mathbf{x})\le 1} \sum_{j\in\mathcal{S}_{\mathrm{core}}} \mu_j \mathbf{z}_j \,\middle|\, \langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle < 0 \right] \mathbf{Pr}_{\mathbf{x}|\mathbf{y}=y}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle < 0].
\end{aligned}$$

Applying Lemma B.1 gives

$$\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=y}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle < 0] = \Phi\left( -\frac{\mathbb{E}_{\mathbf{x}|\mathbf{y}=y} \langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle}{\Theta(1)\sqrt{\sum_{j\in[d_0]} \langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle^2}} \right) \pm O\left(\frac{1}{\sqrt{d_0}}\right).$$

Recall that for $\mathbf{x} \sim \mathcal{D}_{\mathrm{train}}|\mathbf{y} = y$,

$$\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle = \sum_{j\in\mathcal{S}_{\mathrm{core}}} y\mathbf{z}_j \langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle + \sum_{j\in\mathcal{S}_{\mathrm{bg}}} \mathbf{z}_j \langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle.$$

By Definition A.4, we have for every $k \in \mathcal{N}_y^{(t)}$, $\mathbb{E}_{\mathbf{x}|\mathbf{y}=y} \langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \ge \Theta\left(\sqrt{\frac{d_0}{d}}\right) > 0$, which indicates that

$\Phi\left(-\frac{\mathbb{E}_{\mathbf{x}|\mathbf{y}=y}\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle}{\Theta(1)\sqrt{\sum_{j\in[d_0]}\langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle^2}}\right) < \frac{1}{2}$. Together with $h^{(t)}(\mathbf{x}) \le 1$, this gives

$$\begin{aligned} g_{k,y,\mathcal{S}_{\text{core}}}^{(t)} &\ge \frac{1}{2m}\mathbb{E}_{\mathbf{x}|\mathbf{y}=y}\left[\mathbf{1}_{h^{(t)}(\mathbf{x})\le 1}\sum_{j\in\mathcal{S}_{\text{core}}}\mu_j \mathbf{z}_j\right] \\ &\quad - \frac{1}{2m}\mathbb{E}_{\mathbf{x}|\mathbf{y}=y}\left[\mathbf{1}_{h^{(t)}(\mathbf{x})\le 1}\sum_{j\in\mathcal{S}_{\text{core}}}\mu_j \mathbf{z}_j \;\middle|\; \langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle < 0\right]\cdot\left(\frac{1}{2}\pm O\left(\frac{1}{\sqrt{d_0}}\right)\right) \\ &\ge \sum_{j\in\mathcal{S}_{\text{core}}}\frac{\mu_j^2}{2m} - \sum_{j\in\mathcal{S}_{\text{core}}}\frac{\mu_j^2}{4m} \pm \sum_{j\in\mathcal{S}_{\text{core}}}\frac{\mu_j^2}{\Theta(m\sqrt{d_0})} \\ &= \Theta\left(\frac{d_0}{m}\right). \end{aligned} \tag{24}$$

Meanwhile, we also have the upper bound

$$\begin{aligned} g_{k,y,\mathcal{S}_{\text{core}}}^{(t)} &= \frac{1}{m}\mathbb{E}_{(\mathbf{x},\mathbf{y})}\mathbf{1}_{h^{(t)}(\mathbf{x})\le 1}\mathbf{1}_{\mathbf{y}=y}\mathbf{1}_{\langle \mathbf{w}_k^{(t)},\mathbf{x}\rangle\ge 0}\sum_{j\in\mathcal{S}_{\text{core}}}\mu_j\mathbf{z}_j \\ &\le \frac{1}{2m}\mathbb{E}_{\mathbf{x}|\mathbf{y}=y}\sum_{j\in\mathcal{S}_{\text{core}}}\mu_j\mathbf{z}_j \\ &= \Theta\left(\frac{d_0}{m}\right). \end{aligned} \tag{25}$$

Combining Eqs. (24) and (25) gives

$$g_{k,y,\mathcal{S}_{\text{core}}}^{(t)} = \Theta\left(\frac{d_0}{m}\right).$$

Finally, plugging the above equation and $m = O(d)$ into Eq. (23) completes the proof. □

**Lemma B.4** (Gradient projection onto background features, neurons in $\mathcal{N}_y^{(t)}$)**.** *For every iteration $t \le O(\frac{m}{\eta d_0})$, the following holds for every $y \in \mathcal{Y}$ and every neuron $k \in \mathcal{N}_y^{(t)}$:*

$$-\left\langle \nabla_{\mathbf{w}_k^{(t)}}\frac{1}{N}\sum_{i\in[N]}\ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right), \sum_{j\in\mathcal{S}_{\text{bg}}}\mu_j\boldsymbol{m}_j\right\rangle = g_{k,y,\mathcal{S}_{\text{bg}}}^{(t)} - g_{k,-y,\mathcal{S}_{\text{bg}}}^{(t)}, \tag{26}$$

*where*

$$g_{k,y,\mathcal{S}_{\text{bg}}}^{(t)} = \widetilde{\Theta}\left(\frac{d_0}{m}\right). \tag{27}$$

*Proof.* Similar to the proof of Lemma B.3, we apply Lemma A.3 to the LHS of Eq. (26) and using $\operatorname{sign}(a_k) = y$ for every

$k \in \mathcal{N}_y^{(t)}$, which gives

$$\begin{aligned}
&-\left\langle \nabla_{\mathbf{w}_k^{(t)}} \frac{1}{N} \sum_{i \in [N]} \ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right), \sum_{j \in \mathcal{S}_{\mathrm{bg}}} \mu_j \boldsymbol{m}_j \right\rangle \\
&= -\left\langle \nabla_{\mathbf{w}_k^{(t)}} \mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim \mathcal{D}_{\mathrm{train}}} \ell\left(h^{(t)}(\mathbf{x}), \mathbf{y}\right), \sum_{j \in \mathcal{S}_{\mathrm{bg}}} \mu_j \boldsymbol{m}_j \right\rangle \pm \frac{O(d_0)}{\mathsf{poly}(d)} \\
&= a_k \mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim \mathcal{D}_{\mathrm{train}}} \mathbf{y} \mathbf{1}_{h^{(t)}(\mathbf{x}) \le 1} \mathbf{1}_{\langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle \ge 0} \sum_{j \in \mathcal{S}_{\mathrm{bg}}} \mu_j \mathbf{z}_j \pm \frac{O(d_0)}{\mathsf{poly}(d)} \\
&= \frac{1}{m} \mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim \mathcal{D}_{\mathrm{train}}} \mathbf{1}_{h^{(t)}(\mathbf{x}) \le 1} \mathbf{1}_{\mathbf{y}=y} \mathbf{1}_{\langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle \ge 0} \sum_{j \in \mathcal{S}_{\mathrm{bg}}} \mu_j \mathbf{z}_j \\
&\qquad - \frac{1}{m} \mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim \mathcal{D}_{\mathrm{train}}} \mathbf{1}_{h^{(t)}(\mathbf{x}) \le 1} \mathbf{1}_{\mathbf{y}=-y} \mathbf{1}_{\langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle \ge 0} \sum_{j \in \mathcal{S}_{\mathrm{bg}}} \mu_j \mathbf{z}_j \pm \frac{O(d_0)}{\mathsf{poly}(d)} \\
&= g_{k,y,\mathcal{S}_{\mathrm{bg}}}^{(t)} - g_{k,-y,\mathcal{S}_{\mathrm{bg}}}^{(t)} \pm \frac{O(d_0)}{\mathsf{poly}(d)}.
\end{aligned} \tag{28}$$

Also, by a nearly identical argument to Lemma B.3 and $d_{\mathrm{bg}} = \Theta\left(\frac{d_0}{\log d_0}\right)$, we have

$$g_{k,y,\mathcal{S}_{\mathrm{bg}}}^{(t)} = \widetilde{\Theta}\left(\frac{d_0}{m}\right). \tag{29}$$

This completes the proof. □

Next, we also introduce a lemma that bound the gradient projection onto core features for all neurons:

**Lemma B.5** (Gradient projection onto core features, all neurons)**.** *For every iteration $t \le O(\frac{m}{\eta d_0})$, the following holds for every $y \in \mathcal{Y}$ and every neuron $k \in [m]$ with* $\operatorname{sign}(a_k) = y$*:*

$$-\left\langle \nabla_{\mathbf{w}_k^{(t)}} \frac{1}{N} \sum_{i \in [N]} \ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right), \sum_{j \in \mathcal{S}_{\mathrm{core}}} \mu_j \boldsymbol{m}_j \right\rangle = y \cdot O\left(\frac{d_0}{m}\right). \tag{30}$$

*Proof.* By an identical proof to Lemma B.3, we have

$$-\left\langle \nabla_{\mathbf{w}_k^{(t)}} \frac{1}{N} \sum_{i \in [N]} \ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right), \sum_{j \in \mathcal{S}_{\mathrm{core}}} \mu_j \boldsymbol{m}_j \right\rangle = y\left(g_{k,y,\mathcal{S}_{\mathrm{core}}}^{(t)} + g_{k,-y,\mathcal{S}_{\mathrm{core}}}^{(t)}\right) \pm \frac{\Theta(d_0)}{\mathsf{poly}(d)}.$$

By Eq. (25), we have the upper bound $g_{k,y,\mathcal{S}_{\mathrm{core}}}^{(t)} \le \Theta\left(\frac{d_0}{m}\right)$. By a similar argument, we also have $g_{k,-y,\mathcal{S}_{\mathrm{core}}}^{(t)} \le \Theta\left(\frac{d_0}{m}\right)$. Plugging those upper bounds and $m = O(d)$ into the above equation completes the proof. □

We then introduce a lemma that bounds the expected correlation between each neuron in $\mathcal{N}_y^{(t)}$ and its positive examples.

**Lemma B.6** (Correlation with positive examples, neurons in $\mathcal{N}_y^{(t)}$)**.** *For every iteration $t \le O(\frac{m}{\eta d_0})$, every $y \in \mathcal{Y}$, and every $k \in \mathcal{N}_y^{(t)}$, the following holds:*

$$\mathbb{E}_{\mathbf{x}|\mathbf{y}=y \sim \mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t+1)}, \mathbf{x} \rangle] \ge (1 - \lambda\eta) \mathbb{E}_{\mathbf{x}|\mathbf{y}=y \sim \mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle] + \Theta\left(\frac{\eta d_0}{m}\right). \tag{31}$$

*Proof.* Recall Definition 3.1 that given the label $y \in \mathcal{Y}$, $\mathbf{x}$ is generated by

$$\mathbf{x} = \sum_{j \in \mathcal{S}_{\mathrm{core}}} y \mathbf{z}_j \boldsymbol{m}_j + \sum_{j \in \mathcal{S}_{\mathrm{bg}}} \mathbf{z}_j \boldsymbol{m}_j.$$

We can thus obtain

$$\begin{aligned}
&\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}_k^{(t+1)}, \mathbf{x}\rangle] - \mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] \\
&\quad = \underbrace{y\Big(\big\langle \mathbf{w}_k^{(t+1)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j \boldsymbol{m}_j\big\rangle - \big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j \boldsymbol{m}_j\big\rangle\Big)}_{\Delta_{\text{core}}^{(t)}} \\
&\qquad + \underbrace{\left(\big\langle \mathbf{w}_k^{(t+1)}, \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j\big\rangle - \big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j\big\rangle\right)}_{\Delta_{\text{bg}}^{(t)}}.
\end{aligned}$$

For $\Delta_{\text{core}}^{(t)}$, by the SGD iteration (4) we have

$$\Delta_{\text{core}}^{(t)} = -\eta y\Big\langle \nabla_{\mathbf{w}_k^{(t)}} \frac{1}{N}\sum_{i\in[N]} \ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right), \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j \boldsymbol{m}_j\Big\rangle - \lambda\eta y\big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j \boldsymbol{m}_j\big\rangle.$$

Applying Lemma B.3 gives

$$\Delta_{\text{core}}^{(t)} = \eta\left(g_{k,y,\mathcal{S}_{\text{core}}}^{(t)} + g_{k,-y,\mathcal{S}_{\text{core}}}^{(t)}\right) - \lambda\eta y\big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j \boldsymbol{m}_j\big\rangle,$$

which results in the iterative expression

$$y\big\langle \mathbf{w}_k^{(t+1)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j \boldsymbol{m}_j\big\rangle = y(1-\lambda\eta)\big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j \boldsymbol{m}_j\big\rangle + \eta\left(g_{k,y,\mathcal{S}_{\text{core}}}^{(t)} + g_{k,-y,\mathcal{S}_{\text{core}}}^{(t)}\right). \tag{32}$$

For $\Delta_{\text{bg}}^{(t)}$, by the SGD iteration (4) we have

$$\Delta_{\text{bg}}^{(t)} = -\eta\Big\langle \nabla_{\mathbf{w}_k^{(t)}} \frac{1}{N}\sum_{i\in[N]} \ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right), \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j\Big\rangle - \lambda\eta\big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j\big\rangle.$$

Applying Lemma B.4 gives

$$\Delta_{\text{bg}}^{(t)} = \eta\left(g_{k,y,\mathcal{S}_{\text{bg}}}^{(t)} - g_{k,-y,\mathcal{S}_{\text{bg}}}^{(t)}\right) - \lambda\eta\big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j\big\rangle,$$

which results in the iterative expression

$$\big\langle \mathbf{w}_k^{(t+1)}, \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j\big\rangle = (1-\lambda\eta)\big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j\big\rangle + \eta\left(g_{k,y,\mathcal{S}_{\text{bg}}}^{(t)} - g_{k,-y,\mathcal{S}_{\text{bg}}}^{(t)}\right). \tag{33}$$

Combining Eqs. (36) and (37) gives

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}_k^{(t+1)}, \mathbf{x}\rangle] &= y\big\langle \mathbf{w}_k^{(t+1)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j \boldsymbol{m}_j\big\rangle + \big\langle \mathbf{w}_k^{(t+1)}, \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j\big\rangle \\
&= y(1-\lambda\eta)\big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j \boldsymbol{m}_j\big\rangle + \eta\left(g_{k,y,\mathcal{S}_{\text{core}}}^{(t)} + g_{k,-y,\mathcal{S}_{\text{core}}}^{(t)}\right) \\
&\qquad + (1-\lambda\eta)\big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j\big\rangle + \eta\left(g_{k,y,\mathcal{S}_{\text{bg}}}^{(t)} - g_{k,-y,\mathcal{S}_{\text{bg}}}^{(t)}\right) \\
&= y(1-\lambda\eta)\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] \\
&\qquad + \eta\left(g_{k,y,\mathcal{S}_{\text{core}}}^{(t)} + g_{k,-y,\mathcal{S}_{\text{core}}}^{(t)} + g_{k,y,\mathcal{S}_{\text{bg}}}^{(t)} - g_{k,-y,\mathcal{S}_{\text{bg}}}^{(t)}\right) \\
&\overset{(a)}{\geq} (1-\lambda\eta)\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] + \Theta\left(\frac{\eta d_0}{m}\right),
\end{aligned}$$

where $(a)$ is due to $g^{(t)}_{k,y,\mathcal{S}_{\text{core}}} = \Theta\left(\frac{d_0}{m}\right)$ (Lemma B.3), $g^{(t)}_{k,y,\mathcal{S}_{\text{bg}}} = O\left(\frac{d_0}{m}\right)$ (Lemma B.4), and

$$\begin{aligned}
g^{(t)}_{k,-y,\mathcal{S}_{\text{core}}} - g^{(t)}_{k,-y,\mathcal{S}_{\text{bg}}} &= \frac{1}{m}\mathbb{E}_{(\mathbf{x},\mathbf{y})}\mathbf{1}_{h^{(t)}(\mathbf{x})\le 1}\mathbf{1}_{\mathbf{y}=-y}\mathbf{1}_{\langle \mathbf{w}^{(t)}_k,\mathbf{x}\rangle\ge 0}\left(\sum_{j\in\mathcal{S}_{\text{core}}}\mu_j\mathbf{z}_j - \sum_{j\in\mathcal{S}_{\text{bg}}}\mu_j\mathbf{z}_j\right)\\
&\overset{(b)}{\ge} \frac{1}{m}\mathbb{E}_{(\mathbf{x},\mathbf{y})}\mathbf{1}_{h^{(t)}(\mathbf{x})\le 1}\mathbf{1}_{\mathbf{y}=-y}\mathbf{1}_{\langle \mathbf{w}^{(t)}_k,\mathbf{x}\rangle\ge 0}\left(\Theta(d_0) - \Theta\left(\frac{d_0}{\log d_0}\right)\right)\\
&\ge 0,
\end{aligned}$$

where $(b)$ is due to the fact that $d_{\text{core}} = \Theta(d_0)$ and $d_{\text{bg}} = \Theta\left(\frac{d_0}{\log d_0}\right)$. □

We also introduce a general upper bound on the expected correlation between every neuron in the network and its positive examples.

**Lemma B.7** (Correlation with positive examples, all neurons)**.** *For every iteration $t \le O(\frac{m}{\eta d_0})$, the following holds for every $y \in \mathcal{Y}$ and every $k \in \mathcal{N}^{(t)}_y$ with* $\text{sign}(a_k) = y$*:*

$$\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}^{(t+1)}_k,\mathbf{x}\rangle] \le (1-\lambda\eta)\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}^{(t)}_k,\mathbf{x}\rangle] + O\left(\frac{\eta d_0}{m}\right). \tag{34}$$

*Proof.* By an identical proof to Lemma B.6, we have

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}^{(t+1)}_k,\mathbf{x}\rangle] &= y(1-\lambda\eta)\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}^{(t)}_k,\mathbf{x}\rangle]\\
&\quad + \eta\left(g^{(t)}_{k,y,\mathcal{S}_{\text{core}}} + g^{(t)}_{k,-y,\mathcal{S}_{\text{core}}} + g^{(t)}_{k,y,\mathcal{S}_{\text{bg}}} - g^{(t)}_{k,-y,\mathcal{S}_{\text{bg}}}\right)
\end{aligned}$$

By Eq. (25), we have the upper bound $g^{(t)}_{k,y,\mathcal{S}_{\text{core}}} \le \Theta\left(\frac{d_0}{m}\right)$. By a similar argument, we also have the upper bounds $g^{(t)}_{k,-y,\mathcal{S}_{\text{core}}} \le \Theta\left(\frac{d_0}{m}\right)$ and $g^{(t)}_{k,y,\mathcal{S}_{\text{bg}}} \le O\left(\frac{d_0}{m}\right)$. Plugging those upper bounds and the trivial lower bound $g^{(t)}_{k,-y,\mathcal{S}_{\text{bg}}} \ge 0$ into the above equation completes the proof. □

The above two lemmas directly lead to the following result saying that if a neuron is initialized to have large enough correlation to its positive examples (i.e., belonging to $\mathcal{N}^{(0)}_y$), then this large enough correlation will be retained during training.

**Lemma B.8** (Neuron properties during training)**.** *For every label $y \in \mathcal{Y}$, every iteration $t \le \widetilde{O}(\frac{m}{\eta d_0})$, and every step size $\eta \le \frac{1}{\mathsf{poly}(d_0)}$, we have $\mathcal{N}^{(t+1)}_y \supseteq \mathcal{N}^{(t)}_y$.*

*Proof.* By Lemma A.8, we have at initialization

$$\max_{k\in[m]}\left|\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}\langle \mathbf{w}^{(0)}_k,\mathbf{x}\rangle\right| \le \widetilde{O}\left(\sqrt{\frac{d_0}{d}}\right),\ \forall y\in\mathcal{Y}.$$

By Lemma B.7 and our choice of $T = \Theta(\frac{m}{\eta d_0})$, we have

$$\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}^{(t)}_k,\mathbf{x}\rangle] \le O\left(\frac{\eta d_0 T}{m}\right) + \max_{k\in[m]}\left|\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}\langle \mathbf{w}^{(0)}_k,\mathbf{x}\rangle\right| = O(1).$$

By Lemma B.6, we have

$$\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}^{(t+1)}_k,\mathbf{x}\rangle] \ge (1-\lambda\eta)\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}^{(t)}_k,\mathbf{x}\rangle] + \Theta\left(\frac{\eta d_0}{m}\right).$$

Recall that $\lambda = O(\frac{d_0}{m^{1.01}})$. Therefore, as long as $\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] = \widetilde{O}(1)$,

$$\begin{aligned}\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}_k^{(t+1)}, \mathbf{x}\rangle] - \mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] &\geq \Theta\left(\frac{\eta d_0}{m}\right) - \lambda\eta\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] \\ &= \Theta\left(\frac{\eta d_0}{m}\right) > 0.\end{aligned}$$

Finally, recall that $\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] = \sum_{j\in\mathcal{S}_{\text{core}}} y\mu_j\langle \mathbf{w}_k^{(0)}, \boldsymbol{m}_j\rangle + \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j\langle \mathbf{w}_k^{(0)}, \boldsymbol{m}_j\rangle$. By Definition A.4, we immediately have $\mathcal{N}_y^{(t+1)} \supseteq \mathcal{N}_y^{(t)}$. □

Finally, we introduce a lemma that bounds the expected correlation between every neuron in the network and its negative examples.

**Lemma B.9** (Correlation with negative examples, all neurons)**.** *For every iteration $t$, every $y \in \mathcal{Y}$, and every $k \in [m]$ such that* $\operatorname{sign}(a_k) = y$*, the following holds:*

$$\begin{aligned}\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}_k^{(t+1)}, \mathbf{x}\rangle] &= (1-\lambda\eta)\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] \\ &\quad - \Theta\left(\frac{\eta d_0}{m}\right)\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-y}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \geq 0].\end{aligned} \tag{35}$$

*Proof.* Similar to the proof of Lemma B.6, we have

$$\begin{aligned}&\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}_k^{(t+1)}, \mathbf{x}\rangle] - \mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\text{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] \\ &\quad = \underbrace{-y\Big(\big\langle \mathbf{w}_k^{(t+1)}, \sum_{j\in\mathcal{S}_{\text{core}}}\mu_j\boldsymbol{m}_j\big\rangle - \big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{core}}}\mu_j\boldsymbol{m}_j\big\rangle\Big)}_{\Delta_{\text{core}}^{(t)}} \\ &\qquad + \underbrace{\Big(\big\langle \mathbf{w}_k^{(t+1)}, \sum_{j\in\mathcal{S}_{\text{bg}}}\mu_j\boldsymbol{m}_j\big\rangle - \big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{bg}}}\mu_j\boldsymbol{m}_j\big\rangle\Big)}_{\Delta_{\text{bg}}^{(t)}}.\end{aligned}$$

For $\Delta_{\text{core}}^{(t)}$, by the SGD iteration (4) we have

$$\Delta_{\text{core}}^{(t)} = \eta y\Big\langle \nabla_{\mathbf{w}_k^{(t)}}\frac{1}{N}\sum_{i\in[N]}\ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right), \sum_{j\in\mathcal{S}_{\text{core}}}\mu_j\boldsymbol{m}_j\Big\rangle + \lambda\eta y\big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{core}}}\mu_j\boldsymbol{m}_j\big\rangle.$$

Applying Lemma B.3 gives

$$\Delta_{\text{core}}^{(t)} = -\eta\left(g_{k,y,\mathcal{S}_{\text{core}}}^{(t)} + g_{k,-y,\mathcal{S}_{\text{core}}}^{(t)}\right) + \lambda\eta y\big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{core}}}\mu_j\boldsymbol{m}_j\big\rangle,$$

which results in the iterative expression

$$-y\big\langle \mathbf{w}_k^{(t+1)}, \sum_{j\in\mathcal{S}_{\text{core}}}\mu_j\boldsymbol{m}_j\big\rangle = -y(1-\lambda\eta)\big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{core}}}\mu_j\boldsymbol{m}_j\big\rangle - \eta\left(g_{k,y,\mathcal{S}_{\text{core}}}^{(t)} + g_{k,-y,\mathcal{S}_{\text{core}}}^{(t)}\right). \tag{36}$$

For $\Delta_{\text{bg}}^{(t)}$, by the SGD iteration (4) we have

$$\Delta_{\text{bg}}^{(t)} = -\eta\Big\langle \nabla_{\mathbf{w}_k^{(t)}}\frac{1}{N}\sum_{i\in[N]}\ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right), \sum_{j\in\mathcal{S}_{\text{bg}}}\mu_j\boldsymbol{m}_j\Big\rangle - \lambda\eta\big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{bg}}}\mu_j\boldsymbol{m}_j\big\rangle.$$

Applying Lemma B.4 gives

$$\Delta_{\text{bg}}^{(t)} = \eta\left(g_{k,y,\mathcal{S}_{\text{bg}}}^{(t)} - g_{k,-y,\mathcal{S}_{\text{bg}}}^{(t)}\right) - \lambda\eta\big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{bg}}}\mu_j\boldsymbol{m}_j\big\rangle,$$

which results in the iterative expression

$$\Big\langle \mathbf{w}_k^{(t+1)}, \sum_{j\in\mathcal{S}_{\mathrm{bg}}} \mu_j \boldsymbol{m}_j \Big\rangle = (1-\lambda\eta)\Big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\mathrm{bg}}} \mu_j \boldsymbol{m}_j \Big\rangle + \eta\left(g^{(t)}_{k,y,\mathcal{S}_{\mathrm{bg}}} - g^{(t)}_{k,-y,\mathcal{S}_{\mathrm{bg}}}\right). \tag{37}$$

Combining Eqs. (36) and (37) gives

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t+1)}, \mathbf{x}\rangle] &= -y\Big\langle \mathbf{w}_k^{(t+1)}, \sum_{j\in\mathcal{S}_{\mathrm{core}}} \mu_j \boldsymbol{m}_j \Big\rangle + \Big\langle \mathbf{w}_k^{(t+1)}, \sum_{j\in\mathcal{S}_{\mathrm{bg}}} \mu_j \boldsymbol{m}_j \Big\rangle \\
&= -y(1-\lambda\eta)\Big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\mathrm{core}}} \mu_j \boldsymbol{m}_j \Big\rangle - \eta\left(g^{(t)}_{k,y,\mathcal{S}_{\mathrm{core}}} + g^{(t)}_{k,-y,\mathcal{S}_{\mathrm{core}}}\right) \\
&\quad + (1-\lambda\eta)\Big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\mathrm{bg}}} \mu_j \boldsymbol{m}_j \Big\rangle + \eta\left(g^{(t)}_{k,y,\mathcal{S}_{\mathrm{bg}}} - g^{(t)}_{k,-y,\mathcal{S}_{\mathrm{bg}}}\right) \\
&= (1-\lambda\eta)\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] \\
&\quad + \eta\left(g^{(t)}_{k,y,\mathcal{S}_{\mathrm{bg}}} - g^{(t)}_{k,y,\mathcal{S}_{\mathrm{core}}}\right) - \eta\left(g^{(t)}_{k,-y,\mathcal{S}_{\mathrm{core}}} + g^{(t)}_{k,-y,\mathcal{S}_{\mathrm{bg}}}\right).
\end{aligned} \tag{38}$$

For $g^{(t)}_{k,y,\mathcal{S}_{\mathrm{bg}}} - g^{(t)}_{k,y,\mathcal{S}_{\mathrm{core}}}$, we have

$$\begin{aligned}
g^{(t)}_{k,y,\mathcal{S}_{\mathrm{bg}}} - g^{(t)}_{k,y,\mathcal{S}_{\mathrm{core}}} &= \frac{1}{m}\mathbb{E}_{(\mathbf{x},\mathbf{y})}\mathbf{1}_{h^{(t)}(\mathbf{x})\le 1}\mathbf{1}_{\mathbf{y}=y}\mathbf{1}_{\langle \mathbf{w}_k^{(t)},\mathbf{x}\rangle\ge 0}\left(\sum_{j\in\mathcal{S}_{\mathrm{bg}}}\mu_j\mathbf{z}_j - \sum_{j\in\mathcal{S}_{\mathrm{core}}}\mu_j\mathbf{z}_j\right) \\
&= \frac{1}{m}\mathbb{E}_{(\mathbf{x},\mathbf{y})}\mathbf{1}_{h^{(t)}(\mathbf{x})\le 1}\mathbf{1}_{\mathbf{y}=y}\mathbf{1}_{\langle \mathbf{w}_k^{(t)},\mathbf{x}\rangle\ge 0}\left(\Theta\left(\frac{d_0}{\log d_0}\right) - \Theta(d_0)\right) \\
&\le 0.
\end{aligned} \tag{39}$$

For $g^{(t)}_{k,-y,\mathcal{S}_{\mathrm{core}}} + g^{(t)}_{k,-y,\mathcal{S}_{\mathrm{bg}}}$, we have

$$\begin{aligned}
g^{(t)}_{k,-y,\mathcal{S}_{\mathrm{core}}} + g^{(t)}_{k,-y,\mathcal{S}_{\mathrm{bg}}} &= \frac{1}{m}\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\mathrm{train}}}\mathbf{1}_{h^{(t)}(\mathbf{x})\le 1}\mathbf{1}_{\mathbf{y}=-y}\mathbf{1}_{\langle \mathbf{w}_k^{(t)},\mathbf{x}\rangle\ge 0}\left(\sum_{j\in\mathcal{S}_{\mathrm{bg}}}\mu_j\mathbf{z}_j + \sum_{j\in\mathcal{S}_{\mathrm{core}}}\mu_j\mathbf{z}_j\right) \\
&= \frac{1}{m}\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\mathrm{train}}}\mathbf{1}_{h^{(t)}(\mathbf{x})\le 1}\mathbf{1}_{\mathbf{y}=-y}\mathbf{1}_{\langle \mathbf{w}_k^{(t)},\mathbf{x}\rangle\ge 0}\sum_{j\in[d_0]}\mu_j\mathbf{z}_j \\
&= \frac{1}{2m}\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y}\left[\mathbf{1}_{h^{(t)}(\mathbf{x})\le 1}\sum_{j\in[d_0]}\mu_j\mathbf{z}_j \,\middle|\, \langle \mathbf{w}_k^{(t)},\mathbf{x}\rangle\ge 0\right]\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-y}[\langle \mathbf{w}_k^{(t)},\mathbf{x}\rangle\ge 0] \\
&= \Theta\left(\frac{d_0}{m}\right)\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-y}[\langle \mathbf{w}_k^{(t)},\mathbf{x}\rangle\ge 0]
\end{aligned} \tag{40}$$

Finally, plugging Eqs. (39) and (40) into Eq. (38) gives the desired result. □

We are now ready to introduce the proofs of our main theoretical results.

### B.1. Proof of Theorem 4.1

For ease of presentation, we first restate the theorem and then introduce its proof.

**Theorem B.1** (Activation asymmetry)**.** *For every $\eta \le \frac{1}{\mathsf{poly}(d_0)}$ and every $y\in\mathcal{Y}$, there exists $T_0 = \widetilde{\Theta}(\frac{m}{\eta\sqrt{d}})$ such that with high probability, for every $t\ge T_0$, there exist $\Theta(m)$ neurons in which the weight $\mathbf{w}_k^{(t)}$ for each neuron satisfies:*

$$\begin{aligned}
&\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)},\mathbf{x}\rangle\ge 0] = 1 - O\left(d_0^{-\frac{1}{2}}\right), \\
&\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)},\mathbf{x}\rangle\ge 0] = o(1).
\end{aligned} \tag{41}$$

*Proof.* For every $y \in \mathcal{Y}$, consider the neuron set $\mathcal{N}_y^{(t)}$ defined in Definition A.4. By Lemma A.7 and Lemma B.8, we have $|\mathcal{N}_y^{(t)}| = \Theta(m)$ for every iteration $t \leq \Theta(\frac{m}{\eta d_0})$. We then prove that after at most $T_0$ iterations, for every neuron $k \in \mathcal{N}_y^{(T_0)}$ we have $\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(T_0)}, \mathbf{x}\rangle \geq 0] = 1 - O\left(\frac{1}{\sqrt{d_0}}\right)$ and $\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(T_0)}, \mathbf{x}\rangle \geq 0] = o(1)$.

**Part 1: proving $\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(T_0)}, \mathbf{x}\rangle \geq 0] = 1 - O\left(\frac{1}{\sqrt{d_0}}\right)$.**

Let $T_0 = \Theta(\frac{m\sqrt{\log md_0}}{\eta\sqrt{d}}) = \widetilde{\Theta}(\frac{m}{\eta\sqrt{d}})$. By Lemma B.6 and Lemma B.7 we have

$$\begin{aligned}\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t+1)}, \mathbf{x}\rangle] &= (1-\lambda\eta)\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] + \Theta\left(\frac{\eta d_0}{m}\right)\\ &\leq \mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] + \Theta\left(\frac{\eta d_0}{m}\right),\end{aligned}$$

which gives $\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] \leq \widetilde{\Theta}(\frac{d_0}{\sqrt{d}}) = O(1)$. Recall that $\lambda = o(\frac{d_0}{m})$, we then have

$$\begin{aligned}\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t+1)}, \mathbf{x}\rangle] &= (1-\lambda\eta)\mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] + \Theta\left(\frac{\eta d_0}{m}\right)\\ &\geq \mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] + \Theta\left(\frac{\eta d_0}{m}\right) - o\left(\frac{\eta d_0}{m}\right)\\ &= \mathbb{E}_{\mathbf{x}|\mathbf{y}=y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] + \Theta\left(\frac{\eta d_0}{m}\right),\end{aligned}$$

which gives $\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(T_0)}, \mathbf{x}\rangle] \geq \Theta\left(\frac{d_0\sqrt{\log md_0}}{\sqrt{d}}\right)$.

By Lemma B.1, we have

$$\begin{aligned}\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=y}[\langle \mathbf{w}_k^{(T_0)}, \mathbf{x}\rangle \geq 0] &= \Phi\left(\frac{\mathbb{E}_{\mathbf{x}|\mathbf{y}=y}\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle}{\Theta(1)\sqrt{\sum_{j\in[d_0]}\langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle^2}}\right) \pm O\left(\frac{1}{\sqrt{d_0}}\right)\\ &\geq \Phi\left(\frac{\Theta\left(\frac{d_0\sqrt{\log md_0}}{\sqrt{d}}\right)}{\Theta(\sqrt{d_0})\max_{j\in[d_0]}|\langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle|}\right) \pm O\left(\frac{1}{\sqrt{d_0}}\right)\\ &\geq \Phi\left(\frac{\Theta\left(\frac{d_0\sqrt{\log md_0}}{\sqrt{d}}\right)}{\Theta(\sqrt{d_0})\left(O\left(\sqrt{\frac{d_0\log m}{d}}\right) + \Theta\left(\frac{\sqrt{\log md_0}}{\sqrt{d}}\right)\right)}\right) \pm O\left(\frac{1}{\sqrt{d_0}}\right)\\ &= \Phi\left(\Theta\left(\sqrt{\log d_0}\right)\right) \pm O\left(\frac{1}{\sqrt{d_0}}\right).\end{aligned}$$

Applying Lemma D.3 gives $\Phi\left(\Theta\left(\sqrt{\log d_0}\right)\right) = 1 - \Theta(\frac{1}{\sqrt{d_0}})$. We then have

$$\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=y}[\langle \mathbf{w}_k^{(T_0)}, \mathbf{x}\rangle \geq 0] = 1 - O\left(\frac{1}{\sqrt{d_0}}\right).$$

**Part 2: proving $\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(T_0)}, \mathbf{x}\rangle \geq 0] = o(1)$.**

By Lemma B.9, we have for every $t$ and $k \in \mathcal{N}_y^{(t)}$:

$$\begin{aligned}\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t+1)}, \mathbf{x}\rangle] = (1-\lambda\eta)&\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle]\\ &- \Theta\left(\frac{\eta d_0}{m}\right)\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-y}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \geq 0].\end{aligned} \tag{42}$$

By Lemma B.1, we have

$$\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-y}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \geq 0] = \Phi\left(\frac{\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y}\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle}{\Theta(1)\sqrt{\sum_{j\in[d_0]}\langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle^2}}\right) \pm O\left(\frac{1}{\sqrt{d_0}}\right). \tag{43}$$

Assume that a neuron $k \in \mathcal{N}_y^{(0)}$ satisfies $\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y}\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \geq 0$. Then by Eq. (43), we have $\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-y}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \geq 0] \geq \frac{1}{2} \pm O(\frac{1}{\sqrt{d_0}})$, which gives

$$\begin{aligned}\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t+1)}, \mathbf{x}\rangle] &= (1-\lambda\eta)\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] - \Theta\left(\frac{\eta d_0}{m}\right)\\ &\leq \mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] - \Theta\left(\frac{\eta d_0}{m}\right).\end{aligned}$$

By Lemma A.8, we have at initialization $t = 0$:

$$\max_{k\in[m]} \left|\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}\langle \mathbf{w}_k^{(0)}, \mathbf{x}\rangle\right| \leq \widetilde{O}\left(\sqrt{\frac{d_0}{d}}\right). \tag{44}$$

Therefore, for any step size $\eta = \frac{1}{\mathsf{poly}(d_0)}$, after at most $T_{01} := \widetilde{\Theta}(\frac{m}{\eta\sqrt{d_0 d}})$ iterations, we must have $\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y}\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \leq 0$ for every $k \in \mathcal{N}_y^{(t)}$.

Now, let $T_{02} := \Theta(\frac{m\sqrt{\log md_0}}{\eta\sqrt{d}})$. Suppose that $\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-y}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \geq 0] \geq \Theta(1)$ after $t = T_{01} + T_{02} = \widetilde{\Theta}(\frac{m}{\eta\sqrt{d}})$ steps. We then have for $t = T_{01}, \ldots, T_{01} + T_{02} - 1$ that

$$\begin{aligned}\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t+1)}, \mathbf{x}\rangle] &= (1-\lambda\eta)\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] - \Theta\left(\frac{\eta d_0}{m}\right)\\ &\geq \mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] - \Theta\left(\frac{\eta d_0}{m}\right),\end{aligned}$$

which gives $\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] \geq -\widetilde{O}\left(\sqrt{\frac{d_0}{d}}\right) - \Theta\left(\frac{d_0\sqrt{\log md_0}}{\sqrt{d}}\right) \geq -O(1)$. Since $\lambda = o(\frac{d_0}{m})$, we then have

$$\begin{aligned}\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t+1)}, \mathbf{x}\rangle] &= (1-\lambda\eta)\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] - \Theta\left(\frac{\eta d_0}{m}\right)\\ &\leq \mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] - \Theta\left(\frac{\eta d_0}{m}\right) + o\left(\frac{\eta d_0}{m}\right)\\ &= \mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle] - \Theta\left(\frac{\eta d_0}{m}\right),\end{aligned}$$

which gives $\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y\sim\mathcal{D}_{\mathrm{train}}}[\langle \mathbf{w}_k^{(T_{01}+T_{02})}, \mathbf{x}\rangle] \leq -\Theta\left(\frac{d_0\sqrt{\log md_0}}{\sqrt{d}}\right)$. Plugging this into Eq. (43), we obtain

$$\begin{aligned}\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-y}[\langle \mathbf{w}_k^{(T_{01}+T_{02})}, \mathbf{x}\rangle \geq 0] &= \Phi\left(\frac{\mathbb{E}_{\mathbf{x}|\mathbf{y}=-y}\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle}{\Theta(1)\sqrt{\sum_{j\in[d_0]}\langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle^2}}\right) \pm O\left(\frac{1}{\sqrt{d_0}}\right)\\ &\leq \Phi\left(-\frac{\Theta\left(\frac{d_0\sqrt{\log md_0}}{\sqrt{d}}\right)}{\Theta(\sqrt{d_0})\max_{j\in[d_0]}|\langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle|}\right) \pm O\left(\frac{1}{\sqrt{d_0}}\right)\\ &\leq \Phi\left(-\frac{\Theta\left(\frac{d_0\sqrt{\log md_0}}{\sqrt{d}}\right)}{\Theta(\sqrt{d_0})\left(O\left(\sqrt{\frac{d_0\log m}{d}}\right) + \Theta\left(\frac{\sqrt{\log d_0}}{\sqrt{d}}\right)\right)}\right) \pm O\left(\frac{1}{\sqrt{d_0}}\right)\\ &= \Phi\left(-\Theta\left(\sqrt{\log d_0}\right)\right) \pm O\left(\frac{1}{\sqrt{d_0}}\right).\end{aligned}$$

Applying Lemma D.3 gives $\Phi\left(-\Theta\left(\sqrt{\log d_0}\right)\right) = \Theta(\frac{1}{\sqrt{d_0}})$, which leads to

$$\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-y}[\langle \mathbf{w}_k^{(T_{01}+T_{02})}, \mathbf{x}\rangle \geq 0] = O\left(\frac{1}{\sqrt{d_0}}\right).$$

This contradicts with our assumption that $\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-y}[\langle \mathbf{w}_k^{(T_{01}+T_{02})}, \mathbf{x}\rangle \geq 0] \geq \Theta(1)$. Hence, we must have $\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-y}[\langle \mathbf{w}_k^{(T_{01}+T_{02})}, \mathbf{x}\rangle \geq 0] = o(1)$.

Finally, combining **Part 1** and **Part 2** finishes the proof. □

### B.2. Proof of Theorem 4.2

For ease of presentation, we first restate the theorem and then introduce its proof.

**Theorem B.2** (Learned features)**.** *For every $\eta \leq \frac{1}{\mathsf{poly}(d_0)}$ and every $y \in \mathcal{Y}$, there exists $T_1 = \Theta(\frac{m}{\eta d_0})$ such that with high probability, after $T_1$ iterations, there exist $\Theta(m)$ neurons in which the weight $\mathbf{w}_k^{(T_1)}$ for each neuron satisfies the following:*

$$\begin{aligned}\sum\nolimits_{j\in\mathcal{S}_{\text{core}}} \mu_{j1}\langle \mathbf{w}_k^{(T_1)}, \boldsymbol{m}_j\rangle &= y\cdot\Theta(1),\\ \sum\nolimits_{j\in\mathcal{S}_{\text{bg}}} \mu_{j1}\langle \mathbf{w}_k^{(T_1)}, \boldsymbol{m}_j\rangle &= \widetilde{\Theta}(1).\end{aligned} \tag{45}$$

*Proof.* For every $y \in \mathcal{Y}$, consider the neuron set $\mathcal{N}_y^{(t)}$ defined in Definition A.4. By Lemma A.7 and Lemma B.8, we have $|\mathcal{N}_y^{(t)}| = \Theta(m)$ for every iteration $t \leq T_1$. We break the subsequent proof into two parts: in the first part we prove the desired result for core features $\mathcal{S}_{\text{core}}$ for all neurons $k \in \mathcal{N}_y^{(T_1)}$; in the second part we prove the desired result for background features $\mathcal{S}_{\text{bg}}$ for all neurons $k \in \mathcal{N}_y^{(T_1)}$. Recall that we use the shorthand $\mu_j$ to denote $\mu_{j1} = \mathbb{E}_{\mathbf{z}\sim\mathcal{D}_{\text{train}}}\mathbf{z}_j$.

**Part 1: proving $\sum_{j\in\mathcal{S}_{\text{core}}} \mu_j\langle \mathbf{w}_k^{(T_1)}, \boldsymbol{m}_j\rangle = \langle \mathbf{w}_k^{(T_1)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j\boldsymbol{m}_j\rangle = \Theta(1)$.**

The SGD update (4) gives

$$\begin{aligned}&\big\langle \mathbf{w}_k^{(t+1)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j\boldsymbol{m}_j\big\rangle - \big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j\boldsymbol{m}_j\big\rangle\\ &= -\eta\Big\langle \nabla_{\mathbf{w}_k^{(t)}}\frac{1}{N}\sum_{i\in[N]}\ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right), \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j\boldsymbol{m}_j\Big\rangle - \lambda\eta\big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j\boldsymbol{m}_j\big\rangle\end{aligned}$$

for every $t = 0, \ldots, T_1 - 1$.

Applying Lemma B.3, we obtain

$$\begin{aligned}&\big\langle \mathbf{w}_k^{(t+1)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j\boldsymbol{m}_j\big\rangle - \big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j\boldsymbol{m}_j\big\rangle\\ &= y\cdot\Theta\left(\frac{\eta d_0}{m}\right) + y g_{k,-y,\mathcal{S}_{\text{core}}}^{(t)} - \lambda\eta\big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j\boldsymbol{m}_j\big\rangle\\ &= y\cdot\Theta\left(\frac{\eta d_0}{m}\right) - \lambda\eta\big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j\boldsymbol{m}_j\big\rangle.\end{aligned}$$

Without loss of generality, assume that $y = 1$ (the case of $y = -1$ is similar). By the choice of $T_1 = \Theta(\frac{m}{\eta d_0})$, we have

$$\begin{aligned}\big\langle \mathbf{w}_k^{(T_1)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j\boldsymbol{m}_j\big\rangle &\leq \Theta\left(\frac{\eta T_1 d_0}{m}\right) + \big\langle \mathbf{w}_k^{(0)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j\boldsymbol{m}_j\big\rangle\\ &\leq \Theta(1) + \widetilde{O}\left(\frac{d_0}{d}\right) = \Theta(1),\end{aligned}$$

where in the second inequality we apply the concentration inequality of the maximum absolute Gaussian (Lemma D.5). By our choice of $\lambda = o(\frac{d_0}{m})$, we have

$$\begin{aligned}
&\big\langle \mathbf{w}_k^{(t+1)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j \boldsymbol{m}_j \big\rangle - \big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j \boldsymbol{m}_j \big\rangle \\
&= \Theta\left(\frac{\eta d_0}{m}\right) - \lambda\eta \big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j \boldsymbol{m}_j \big\rangle \\
&\geq \Theta\left(\frac{\eta d_0}{m}\right) - o\left(\frac{\eta d_0}{m}\right) = \Theta\left(\frac{\eta d_0}{m}\right).
\end{aligned}$$

Summing the above inequality from $t = 0$ to $t = T_1 - 1$ yields

$$\big\langle \mathbf{w}_k^{(T_1)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j \boldsymbol{m}_j \big\rangle = \Theta(1).$$

Similarly, for $y = -1$ we have $\big\langle \mathbf{w}_k^{(T_1)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j \boldsymbol{m}_j \big\rangle = -\Theta(1)$.

**Part 2: proving $\sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \langle \mathbf{w}_k^{(T_1)}, \boldsymbol{m}_j \rangle = \langle \mathbf{w}_k^{(T_1)}, \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j \rangle = \widetilde{\Theta}(1)$.**

Similar to the first part of the proof, we have the SGD update

$$\begin{aligned}
&\big\langle \mathbf{w}_k^{(t+1)}, \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j \big\rangle - \big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j \big\rangle \\
&= -\eta \Big\langle \nabla_{\mathbf{w}_k^{(t)}} \frac{1}{N} \sum_{i\in[N]} \ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right), \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j \Big\rangle - \lambda\eta \big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j \big\rangle.
\end{aligned}$$

Applying Lemma B.4, we obtain

$$\begin{aligned}
&\big\langle \mathbf{w}_k^{(t+1)}, \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j \big\rangle - \big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j \big\rangle \\
&= \widetilde{\Theta}\left(\frac{\eta d_0}{m}\right) - \eta g_{k,-y,\mathcal{S}_{\text{bg}}}^{(t)} - \lambda\eta \big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j \big\rangle,
\end{aligned}$$

where

$$\begin{aligned}
g_{k,-y,\mathcal{S}_{\text{bg}}}^{(t)} &= \frac{1}{m} \sum_{j\in\mathcal{S}_{\text{bg}}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\text{train}}} \mathbf{1}_{h^{(t)}(\mathbf{x})\leq 1} \mathbf{1}_{\mathbf{y}=-y} \mathbf{1}_{\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \geq 0} \mu_j \mathbf{z}_j \\
&= \frac{1}{2m} \mathbb{E}_{\mathbf{x}|\mathbf{y}=-y}\left[\mathbf{1}_{h^{(t)}(\mathbf{x})\leq 1} \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \mathbf{z}_j \,\middle|\, \langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \geq 0\right] \mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-y}\left[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \geq 0\right] \\
&\leq \Theta\left(\frac{d_0}{m}\right) \mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-y}\left[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \geq 0\right].
\end{aligned}$$

Using Theorem 4.1, we have after at most $T_0 = \widetilde{\Theta}(\frac{m}{\eta\sqrt{d}})$ iterations, $\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-y}\left[\langle \mathbf{w}_k^{(t)}, \mathbf{x}\rangle \geq 0\right] = o(1)$. We thus have

$$\big\langle \mathbf{w}_k^{(t+1)}, \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j \big\rangle = (1-\lambda\eta) \big\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j \big\rangle + \widetilde{\Theta}\left(\frac{\eta d_0}{m}\right)$$

for every $t \geq T_0$. By a similar argument as in the first part of the proof, we have $\langle \mathbf{w}_k^{(T)}, \sum_{j \in \mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j \rangle \leq \Theta(1)$ and

$$\begin{aligned}
\left\langle \mathbf{w}_k^{(T_1)}, \sum_{j \in \mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j \right\rangle &= (T_1 - T_0)\widetilde{\Theta}\left(\frac{\eta d_0}{m}\right) + \left\langle \mathbf{w}_k^{(T_0)}, \sum_{j \in \mathcal{S}_{\text{bg}}} \mu_j \boldsymbol{m}_j \right\rangle \\
&\geq \widetilde{\Theta}(1) - T_0 \cdot \Theta\left(\frac{\eta d_0}{m}\right) - \widetilde{O}\left(\sqrt{\frac{d_0}{d}}\right) \\
&\overset{(a)}{=} \widetilde{\Theta}(1) - \widetilde{\Theta}\left(\frac{m}{\eta\sqrt{d}}\right)\Theta\left(\frac{\eta d_0}{m}\right) \\
&= \widetilde{\Theta}(1),
\end{aligned}$$

where $(a)$ is due to $d \in [\Omega(d_0^{2.01}), \mathsf{poly}(d_0)]$.

Finally, combining **Part 1** and **Part 2** completes the proof. □

### B.3. Proof of Theorem 4.3

For ease of presentation, we first restate the theorem and then introduce its proof.

**Theorem B.3** (ID and OOD risks)**.** *For every $\eta \leq \frac{1}{\mathsf{poly}(d_0)}$, there exists $T_2 = \widetilde{\Theta}(\frac{m}{\eta d_0})$ such that with high probability, after $T_2$ iterations, the trained model $h^{(T_2)}$ satisfies the following:*

$$\begin{aligned}
&\mathcal{R}_{\mathcal{D}_{\text{train}}}(h^{(T_2)}) \leq o(1), \\
&\mathcal{R}_{\text{OOD}}(h^{(T_2)}) = \widetilde{\Theta}(1).
\end{aligned} \tag{46}$$

*Proof.* We break the subsequent proof into two parts: in the first part we prove the desired result for the ID risk; in the second part we prove the desired result for the OOD risk.

**Part 1: proving $\mathcal{R}_{\mathcal{D}_{\text{train}}}(h^{(T_2)}) \leq o(1)$.**

By definition, we have

$$\begin{aligned}
\mathcal{R}_{\mathcal{D}_{\text{train}}}(h^{(T_2)}) &= \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\text{train}}} \max\left\{1 - \mathbf{y}h^{(T_2)}(\mathbf{x}), 0\right\} \\
&= \frac{1}{2}\underbrace{\mathbb{E}_{\mathbf{x}|\mathbf{y}=1}\left[1 - h^{(T_2)}(\mathbf{x})\middle| h^{(T_2)}(\mathbf{x}) \leq 1\right]\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=1}\left[h^{(T_2)}(\mathbf{x}) \leq 1\right]}_{\mathcal{R}_1} \\
&\quad + \frac{1}{2}\underbrace{\mathbb{E}_{\mathbf{x}|\mathbf{y}=-1}\left[1 + h^{(T_2)}(\mathbf{x})\middle| h^{(T_2)}(\mathbf{x}) \geq -1\right]\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=-1}\left[h^{(T_2)}(\mathbf{x}) \geq -1\right]}_{\mathcal{R}_{-1}}.
\end{aligned} \tag{47}$$

We first consider $\mathcal{R}_1$. By Theorem 4.2, we have that after $T_1 = \Theta(\frac{m}{\eta d_0})$ iterations, for every neuron $k \in \mathcal{N}_1^{(t)}$ with $t \geq T_1$, we have

$$\sum_{j \in \mathcal{S}_{\text{core}}} \mu_j \mathbf{z}_j \langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j \rangle = \Theta(1), \quad \sum_{j \in \mathcal{S}_{\text{bg}}} \mu_j \mathbf{z}_j \langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j \rangle = \widetilde{\Theta}(1).$$

We can then obtain

$$\mathbb{E}_{\mathbf{x}|\mathbf{y}=1} \langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle = \sum_{j \in [d_0]} \mu_j \mathbf{z}_j \langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle = \Theta(1).$$

On the other hand, by Lemma B.7, we know that for every neuron $k$ satisfying $\text{sign}(a_k) = y$, its correlation grow rate is asymptotically not larger than the correlation grow rate of neurons in $\mathcal{N}_y^{(t)}$. Denoting the set of those neurons as $\mathcal{M}_y := \{k \in [m] : \text{sign}(a_k) = y\}$, $\forall y \in \mathcal{Y}$, we then have

$$\mathbb{E}_{\mathbf{x}|\mathbf{y}=1} \langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle = O(1), \, \forall k \in \mathcal{M}_1, t \geq T_1.$$

Meanwhile, for all neurons $k \in \mathcal{M}_{-1}$, by Lemma B.9 and Theorem 4.1 we have for all $t \geq T_0 = \widetilde{\Theta}(\frac{m}{\eta\sqrt{d}})$,

$$\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=1}[\langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle \geq 0] = o(1).$$

Therefore, we have

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}|\mathbf{y}=1} h^{(T_1)}(\mathbf{x}) &= \frac{1}{m} \sum_{k \in \mathcal{M}_1} \mathbb{E}_{\mathbf{x}|\mathbf{y}=1}\Big[\mathsf{ReLU}\left(\langle \mathbf{w}_k^{(T_1)}, \mathbf{x} \rangle\right)\Big] - \frac{1}{m} \sum_{k \in \mathcal{M}_{-1}} \mathbb{E}_{\mathbf{x}|\mathbf{y}=1}\Big[\mathsf{ReLU}\left(\langle \mathbf{w}_k^{(T_1)}, \mathbf{x} \rangle\right)\Big] \\
&= \frac{1}{m} \sum_{k \in \mathcal{M}_1} \Theta(1) - \frac{1}{m} \sum_{k \in \mathcal{M}_{-1}} o(1) \\
&= \Theta(1).
\end{aligned}$$

Now, suppose that $\mathcal{R}_1 \geq \Theta(1)$. Choose $T_2 = \Theta(\frac{m\sqrt{\log m}}{\eta d_0}) = \widetilde{\Theta}(\frac{m}{\eta d_0})$. Then, for every $t = T_1, \ldots, T_2 - 1$ we have

$$\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=1}\Big[h^{(T_2)}(\mathbf{x}) \leq 1\Big] = \Theta(1).$$

This further leads to

$$\begin{aligned}
&\frac{1}{m} \sum_{k \in \mathcal{M}_1} \mathbb{E}_{\mathbf{x}|\mathbf{y}=1}\Big[\mathsf{ReLU}\left(\langle \mathbf{w}_k^{(t+1)}, \mathbf{x} \rangle\right)\Big] - \frac{1}{m} \sum_{k \in \mathcal{M}_1} \mathbb{E}_{\mathbf{x}|\mathbf{y}=1}\Big[\mathsf{ReLU}\left(\langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle\right)\Big] \\
&\quad = \frac{1}{m} \sum_{k \in \mathcal{M}_1} \mathbb{E}_{\mathbf{x}|\mathbf{y}=1}\Big[\langle \mathbf{w}_k^{(t+1)}, \mathbf{x} \rangle - \langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle\Big] \\
&\quad = \frac{1}{m} \sum_{k \in \mathcal{N}_1^{(t)}} \mathbb{E}_{\mathbf{x}|\mathbf{y}=1}\Big[\langle \mathbf{w}_k^{(t+1)}, \mathbf{x} \rangle - \langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle\Big] + \frac{1}{m} \sum_{k \in \mathcal{M}_1 \setminus \mathcal{N}_1^{(t)}} \mathbb{E}_{\mathbf{x}|\mathbf{y}=1}\Big[\langle \mathbf{w}_k^{(t+1)}, \mathbf{x} \rangle - \langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle\Big]
\end{aligned} \tag{48}$$

For the first term in the RHS of the last equality in (48), by Lemma B.6 we have

$$\begin{aligned}
&\frac{1}{m} \sum_{k \in \mathcal{N}_1^{(t)}} \mathbb{E}_{\mathbf{x}|\mathbf{y}=1}\Big[\langle \mathbf{w}_k^{(t+1)}, \mathbf{x} \rangle - \langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle\Big] \\
&\quad = \frac{1}{m} \sum_{k \in \mathcal{N}_1^{(t)}} \left(\Theta\left(\frac{\eta d_0}{m}\right) - \lambda\eta \mathbb{E}_{\mathbf{x}|\mathbf{y}=1} \langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle\right) \\
&\quad = \Theta\left(\frac{\eta d_0}{m}\right),
\end{aligned}$$

where in the last equality we use $|\mathcal{N}_1^{(t)}| = \Theta(m)$, $\lambda = O(\frac{d_0}{m^{0.01}})$ and $\mathbb{E}_{\mathbf{x}|\mathbf{y}=1}\langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle = \widetilde{O}(1)$ for $t \leq T_2$.

For the second term in the RHS of the last equality in (48), by Lemma B.7 we have

$$\frac{1}{m} \sum_{k \in \mathcal{M}_1 \setminus \mathcal{N}_1^{(t)}} \mathbb{E}_{\mathbf{x}|\mathbf{y}=1}\Big[\langle \mathbf{w}_k^{(t+1)}, \mathbf{x} \rangle - \langle \mathbf{w}_k^{(t)}, \mathbf{x} \rangle\Big] \leq O\left(\frac{\eta d_0}{m}\right).$$

Therefore,

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}|\mathbf{y}=1} h^{(T_2)}(\mathbf{x}) &= \mathbb{E}_{\mathbf{x}|\mathbf{y}=1} h^{(T_1)}(\mathbf{x}) + \Theta\left(\frac{\eta d_0 (T_2 - T_1)}{m}\right) \pm o(1) \\
&= \Theta(1) + \Theta(\sqrt{\log m}) \pm o(1) = \Theta(\sqrt{\log m}).
\end{aligned}$$

Applying one-sided Bernstein's inequality (Lemma D.1) then gives

$$\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=1}\Big[h^{(T_2)}(\mathbf{x}) \leq 1\Big] = O\left(\frac{1}{\sqrt{m}}\right),$$

which contradicts with $\mathbf{Pr}_{\mathbf{x}|\mathbf{y}=1}\left[h^{(T_2)}(\mathbf{x}) \leq 1\right] = \Theta(1)$. Hence, we must have $\mathcal{R}_1 = o(1)$. By a similar argument, we also have $\mathcal{R}_{-1} = o(1)$. We then have that $\mathcal{R}_{\mathcal{D}_{\text{train}}}(h^{(T_2)}) = o(1)$ holds.

**Part 2: proving $\mathcal{R}_{\text{OOD}}(h^{(T)}) = \widetilde{\Theta}(1)$.**

This part of the proof directly follows from Theorem 4.2. Since after $t = T_1$ iterations we have $\sum_{j \in \mathcal{S}_{\text{bg}}} \mu_j \langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j \rangle = \widetilde{\Theta}(1)$ for every neuron $k \in \mathcal{N}_y^{(t)}$, it can be shown that perturbing each $\mu_j$ from $\Theta(1)$ to $-\Theta(1)$ for $j \in \mathcal{S}_{\text{bg}}$ (recall the generation process of the OOD data in Definition 3.1) changes the output of the network by at least $-\frac{1}{m}\sum_{k \in \mathcal{N}_y^{(t)}} \widetilde{\Theta}(1) = -\widetilde{\Theta}(1)$ using the fact that $|\mathcal{N}_y^{(t)}| = \Theta(m)$ for every $t$ (using Lemma A.7 and Lemma B.8). By the definition of the OOD risk we then arrive at the desired result.

Finally, combining **Part 1** and **Part 2** completes the proof. □

## C. Separation between Linear Networks and Non-Linear Networks: Proof of Theorem 4.4

Before providing the main proof, we first introduce some lemmas that characterize the gradients of the two-layer linear network. In general, the gradients of two-layer linear networks take a similar form to those of two-layer ReLU networks except for not having the ReLU derivative. We can thus reuse some of our lemmas in Section A and Section B in the analysis of the gradients.

**Notation.** In this section, we overload the notation from the previous sections such as $h^{(t)}(\mathbf{x})$ and $\mathbf{w}_k^{(t)}$ and let them also denote the linear network model/weights.

**Lemma C.1** (Gradient of linear networks)**.** *For every example $(x, y) \in \mathcal{X} \times \mathcal{Y}$, every $k \in [m]$, and every iteration $t$, the following holds:*

$$\nabla_{\mathbf{w}_k^{(t)}} \ell\left(h(x), y\right) = -a_k y \mathbf{1}_{h(x) \leq 1} x. \tag{49}$$

*Proof.* The proof follows from simple calculation. □

**Lemma C.2** (Gap between empirical and population gradients)**.** *For every $k \in [m]$, every $j \in [d]$, and every iteration $t$, if the batch size $N = \mathsf{poly}(d)$ for some sufficiently large polynomial, then the following holds with probability at least $1 - e^{-\Omega(d)}$:*

$$\left| \left\langle \nabla_{\mathbf{w}_k^{(t)}} \frac{1}{N} \sum_{i \in [N]} \ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right), \boldsymbol{m}_j \right\rangle - \left\langle \nabla_{\mathbf{w}_k^{(t)}} \mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim \mathcal{D}_{\text{train}}} \ell\left(h(\mathbf{x}), \mathbf{y}\right), \boldsymbol{m}_j \right\rangle \right| \leq \frac{1}{\mathsf{poly}(d)}. \tag{50}$$

*Proof.* The proof is nearly identical to Lemma A.2, hence we omit here. □

Since in linear models we do not need to consider the activation probability (equivalently, this can be viewed as each neuron being fully activated for every example), we can analyze the gradient projections for all neurons without resorting to characterizing a subset of neurons as in Definition A.4.

**Lemma C.3** (Gradient projection onto background features, linear networks)**.** *For every iteration $t \leq O(\frac{m}{\eta d_0})$, every $k \in [m]$, and every $j \in \mathcal{S}_{\text{bg}}$, the following holds:*

$$\left| \left\langle \nabla_{\mathbf{w}_k^{(t)}} \frac{1}{N} \sum_{i \in [N]} \ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right), \boldsymbol{m}_j \right\rangle \right| = \frac{1}{\mathsf{poly}(d)}, \tag{51}$$

*Proof.* Applying Lemma C.1 and Lemma C.2, we obtain

$$\begin{aligned}
&-\Big\langle \nabla_{\mathbf{w}_k^{(t)}} \frac{1}{N}\sum_{i\in[N]} \ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right), \boldsymbol{m}_j \Big\rangle \\
&\quad = -\langle \nabla_{\mathbf{w}_k^{(t)}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\text{train}}} \ell\left(h^{(t)}(\mathbf{x}), \mathbf{y}\right), \boldsymbol{m}_j\rangle \pm \frac{1}{\mathsf{poly}(d)} \\
&\quad = a_k \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\text{train}}} \mathbf{y}\mathbf{1}_{h^{(t)}(\mathbf{x})\le 1}\mathbf{z}_j \pm \frac{1}{\mathsf{poly}(d)} \\
&\quad = a_k \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\text{train}}} \mathbf{1}_{h^{(t)}(\mathbf{x})\le 1}\mathbf{1}_{\mathbf{y}=1}\mathbf{z}_j \\
&\qquad\quad - a_k \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\text{train}}} \mathbf{1}_{h^{(t)}(\mathbf{x})\le 1}\mathbf{1}_{\mathbf{y}=-1}\mathbf{z}_j \pm \frac{1}{\mathsf{poly}(d)} \\
&\quad = \pm\frac{1}{\mathsf{poly}(d)}.
\end{aligned} \tag{52}$$

This gives the desired result. □

We are now ready to prove Theorem 4.4. For ease of presentation, we first restate the theorem and then introduce its proof.

**Theorem C.4** (Linear networks)**.** *If we replace the ReLU functions in the network with identity functions and keep other conditions the same as in Theorem 4.2, then with high probability, we have* $|\langle \mathbf{w}_k^{(T_1)}, \boldsymbol{m}_j\rangle| \le \widetilde{O}(\frac{1}{\sqrt{d}})$ *for every* $k\in[m]$ *and every* $j\in\mathcal{S}_{\text{bg}}$.

*Proof.* For every $k\in[m]$ and every $j\in\mathcal{S}_{\text{bg}}$, by the SGD update (4) we have

$$\langle \mathbf{w}_k^{(t+1)}, \boldsymbol{m}_j\rangle = -\eta\Big\langle \nabla_{\mathbf{w}_k^{(t)}} \frac{1}{N}\sum_{i\in[N]} \ell\left(h^{(t)}(\mathbf{x}_i^{(t)}), \mathbf{y}_i^{(t)}\right), \boldsymbol{m}_j\Big\rangle + (1-\lambda\eta)\langle \mathbf{w}_k^{(t)}, \sum_{j\in\mathcal{S}_{\text{core}}} \mu_j \boldsymbol{m}_j\rangle.$$

By Lemma C.3, we obtain

$$\langle \mathbf{w}_k^{(t+1)}, \boldsymbol{m}_j\rangle = (1-\lambda\eta)\langle \mathbf{w}_k^{(t)}, \boldsymbol{m}_j\rangle \pm \frac{\eta}{\mathsf{poly}(d)}.$$

By Lemma D.5, with probability at least $1-O(\frac{1}{m})$, we have at initialization

$$\max_{k\in[m]} |\langle \mathbf{w}_k^{(0)}, \boldsymbol{m}_j\rangle| \le 2\sqrt{\frac{\log m}{d}}.$$

Recall that $\lambda = O(\frac{d_0}{m^{1.01}})$. Combining the above equations gives the desired result. □

**Remark.** Similar to our analysis of two-layer ReLU networks, for two-layer linear networks we can also analyze the correlation growth between every neuron and the core features and show that SGD can converge to a solution with small ID risk. Since Theorem 4.4 indicates that linear networks do not have feature contamination (i.e., background features do not accumulate in the weights), we can show that the network would also have small OOD risk at convergence. Since this analysis has a similar procedure to (and is also much simpler than) our analysis on two-layer ReLU networks we do not include it here.

## D. Probability Theory Lemmas

In this section, we provide the probability theory lemmas used in our proofs for completeness. Since those lemmas are standard results in the probability theory we omit the proofs of them.

We first state an one-sided form of Bernstein's inequality.

**Lemma D.1** (One-sided Bernstein's inequality)**.** *Given* $n$ *independent random variables* $\{\mathbf{x}_i\}_{i\in[n]}$ *with* $\mathbf{x}_i \le b$ *almost surely for every* $i\in[n]$*, the following holds for every* $\delta\ge 0$*:*

$$\mathbf{Pr}\Big[\sum_{i\in[n]}(\mathbf{x}_i - \mathbb{E}\mathbf{x}_i) \ge n\delta\Big] \le \exp\left(-\frac{n\delta^2}{\frac{1}{n}\sum_{i\in[n]}\mathbb{E}\mathbf{x}_i^2 + \frac{b\delta}{3}}\right). \tag{53}$$

Note that the above result can also be used to derive bounds on the lower tail by applying it to the random variables $\{-\mathbf{x}_i\}_{i\in[n]}$ if each $\mathbf{x}_i$ is bounded from below.

We then state a matrix extension of Bernstein's inequality; such type of inequalities is useful for bounding the gradients of the network in our proofs.

**Lemma D.2** (Matrix Bernstein's inequality (Oliveira, 2010; Tropp, 2012))**.** *Given $n$ independent random matrices $\{\mathbf{X}_i\}_{i\in[n]}$ with dimension $d_1 \times d_2$ and $\mathbb{E}\mathbf{X}_i = \mathbf{0}$, $\|\mathbf{X}_i\|_2 \le b$ almost surely for every $i \in [n]$, define the sum $\mathbf{S} := \sum_{i\in[n]} \mathbf{X}_i$ and let $v(\mathbf{S})$ denote the matrix variance statistic of the sum:*

$$v(\mathbf{S}) := \max\{\|\mathbb{E}[\mathbf{S}\mathbf{S}^*]\|_2, \|\mathbb{E}[\mathbf{S}^*\mathbf{S}]\|_2\}, \tag{54}$$

*where $\|\cdot\|_2$ denotes the spectral norm a matrix or the $\ell_2$ norm of a vector (when $d_1 = 1$ or $d_2 = 1$). Then, the following holds for every $\delta \ge 0$:*

$$\mathbf{Pr}\left[\|\mathbf{S}\|_2 \ge \delta\right] \le (d_1 + d_2) \cdot \exp\left(-\frac{\delta^2}{2v(\mathbf{S}) + \frac{2b\delta}{3}}\right). \tag{55}$$

We then state a basic result for bounding the cumulative distribution function of the unit Gaussian distribution that is repeatedly used in deriving neuron properties in initialization.

**Lemma D.3** (Bounds for unit Gaussian variables)**.** *Let $\mathbf{x} \sim \mathcal{N}(0,1)$ be a unit Gaussian random variable. Then, the following holds for every $\delta > 0$:*

$$\frac{1}{\sqrt{2\pi}} \frac{\delta}{\delta^2 + 1} \exp\left(-\frac{\delta^2}{2}\right) \le \mathbf{Pr}[\mathbf{x} \ge \delta] \le \frac{1}{\sqrt{2\pi}} \frac{1}{\delta} \exp\left(-\frac{\delta^2}{2}\right). \tag{56}$$

Finally, we state a result for lower bounding the upper tail of the cumulative distribution function for binomial variables using Hoeffding's inequality:

**Lemma D.4** (Tail bound for binomial variables)**.** *Let $\mathbf{x} \sim \mathcal{B}(n,p)$ be a binomial random variable with trial number $n$ and success probability $p \in [0,1]$. Then, the following holds for every $n, p$ and integer $k \le np$:*

$$\mathbf{Pr}[\mathbf{x} \ge k] \ge 1 - \exp\left(-2n\left(p - \frac{k-1}{n}\right)^2\right). \tag{57}$$

**Lemma D.5** (Concentration of the maximum of absolute Gaussian)**.** *Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be i.i.d. random variables that follow the zero-mean Gaussian distribution $\mathcal{N}(0, \sigma^2)$. Then, the following holds for every positive integer $n$:*

$$\mathbf{Pr}\left[\max_{i\in[n]} |\mathbf{x}_i| \ge 2\sigma\sqrt{\log n}\right] \le \frac{2}{n}. \tag{58}$$

**Lemma D.6** (Berry–Esseen theorem)**.** *Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be independent random variables with $\mathbb{E}\mathbf{x}_i = 0$, $\mathbb{E}\mathbf{x}_i^2 = \sigma_i^2 > 0$, and $\rho_i := \mathbb{E}|\mathbf{x}_i^3| < \infty$. Also, define the normalized sum*

$$\mathbf{s}_n := \frac{\sum_{i\in[n]} \mathbf{x}_i}{\sqrt{\sum_{i\in[n]} \sigma_i^2}}. \tag{59}$$

*Denote $\Phi$ the cumulative distribution function of $\mathcal{N}(0,1)$. Then, there exists a constant $C_0 \in [0.40, 0.56]$ such that*

$$\sup_{\delta\in\mathbb{R}} |\mathbf{Pr}[\mathbf{s}_n < \delta] - \Phi(\delta)| \le C_0 \left(\sum_{i=1}^n \sigma_i^2\right)^{-\frac{3}{2}} \sum_{i=1}^n \rho_i. \tag{60}$$

# Appendix II: Experimental Details and Additional Results

In this part of the appendix, we present the details of the experiments in the main text and include additional empirical results in both real-world datasets and synthetic distribution shift settings. A quick overview of the structure of this part is as follows:

- In Section E, we provide the implementation details and more results of the representation distillation experiments in Section 2.
- In Section F, we present more numerical results, implementation details, more results of class activation histograms, and additional feature visualization for deep neural networks.
- In Section G, we provide empirical evidence that supports the conjecture in Section 6 and more discussion on related work.

## E. Representation Distillation Details

### E.1. Natural Distribution Shifts of ImageNet

**Datasets.** Following (Taori et al., 2020; Radford et al., 2021; Wortsman et al., 2022), we consider 5 natural distribution shift test sets of ImageNet that are representative of real-world distribution shifts without artificial perturbations to images, including ImageNetV2 (Recht et al., 2019), ImageNet-R (Hendrycks et al., 2021a), ObjectNet (Barbu et al., 2019), ImageNet Sketch (Wang et al., 2019), and ImageNet-A (Hendrycks et al., 2021b). Compared to the original training and validation (ID test) sets of ImageNet, those test sets are reflective of changes in data distribution due to natural variations in the data collection process such as lighting, geographic location, image background, and styles.

**Pre-trained models.** We used pre-trained checkpoints provided by CLIP (Radford et al., 2021), which is reported to exhibit remarkable robustness to distribution shifts of ImageNet. The official CLIP repository provide CLIP models pre-trained on the same dataset with varying sizes and architectures (ResNets and ViTs). In our experiments, we used five different CLIP models, including four ResNets and one ViT: CLIP-ResNet-50 (CLIP-RN50), CLIP-ResNet-101 (CLIP-RN101), CLIP-ResNet-50x4 (CLIP-RN50x4), CLIP-ResNet-50x16 (CLIP-RN50x16), and CLIP-ViT-B/16. For linear probing, we freezed the weights of the pre-trained models and trained randomly-initialized linear classification heads on top of the extracted representations on the ImageNet training set for 10 epochs. Following the hyperparameters used by Wortsman et al. (2022), we used the AdamW optimizer (Loshchilov & Hutter, 2019) with learning rate 0.001, $\ell_2$ weight decay 0.1, batch size 256, and a cosine learning rate scheduler (Loshchilov & Hutter, 2017). The results are reported based on the model with the best ID validation accuracy.

**Representation distillation.** For each pre-trained CLIP model (teacher model), we freezed its weights and randomly initialized another model with identical architecture to the teacher model. We used the Mean Squared Error (MSE) loss to train the student model on the ImageNet training set, minimizing the mean Euclidean distance between the representations extracted by the student model and the representations extracted by the teacher model. We did not perform extensive grid search on the distillation hyperparameters and sticked to the following hyperparameter choices based on our preliminary experiments:

- CLIP-RN50: AdamW optimizer with learning rate 0.001, $\ell_2$ weight decay 0.05, batch size 256, and a cosine learning rate schedular with warmup for 10000 steps; 100 distillation epochs.
- CLIP-RN101: AdamW optimizer with learning rate 0.001, $\ell_2$ weight decay 0.1, batch size 256, and a cosine learning rate scheduler with warmup for 10000 steps; 100 distillation epochs.
- CLIP-RN50x4 and CLIP-RN50x16: AdamW optimizer with learning rate 0.0001, $\ell_2$ weight decay 0.5, batch size 256, and a cosine learning rate scheduler with warmup for 10000 steps; 100 distillation epochs.
- CLIP-ViT-B/16: AdamW optimizer with learning rate 0.0001, $\ell_2$ weight decay 0.1, batch size 256, and a cosine learning rate scheduler with warmup for 10000 steps; 200 distillation epochs. Besides minimizing the difference

between final representations (i.e., the output of the last layer of the networks) of student and teacher networks, we also minimized the difference between student and teacher network's intermediate representations of each residual attention block with a weighting coefficient 0.1.

In the linear probing stage, we freezed the parameters of the student models and trained a randomly initialized linear classification head for each student model on the ImageNet training set for 10 epochs. We used the AdamW optimizer with learning rate 0.001, $\ell_2$ weight decay of 0.001, batch size 256, and a cosine learning rate scheduler. The results are reported based on the model with the best ID validation accuracy.

**Baseline models.** We reported the results of baseline models provided by the testbed of Taori et al. (2020). In their testbed, Taori et al. (2020) catogory the models into different types, where some type of models are trained with more data than the original ImageNet training set. Since our aim is to explore the limit of representation learning using only ID data, we omit the results of those models trained with more data. We also omit the results of models with significantly lower accuracy than common ImageNet models, such as linear classifier on pixels or random features, classifiers based on nearest neighbors, and low accuracy CNNs. Concretely, we reported the results of the following two types of models defined by Taori et al. (2020):

- `STANDARD`: models obtained by standard training (i.e., ERM) on the ImageNet training set.
- `ROBUST_INTV`: models trained with existing robust intervention techniques on the ImageNet training set.

**Detailed results.** We list detailed OOD generalization performance of linear probes on pre-trained and distilled representations on all 5 distribution shift test sets as well as the ID generalization results on the original ImageNet validation set in Table 1.

Table 1: Detailed ID and OOD top-1 accuracy (%) of linear probes on pre-trained and distilled representations on ImageNet-based test sets. “Im” refers to “ImageNet”.

| | Im (ID) | OOD Avg. | ImV2 | Im-R | ObjectNet | Im Sketch | Im-A |
|---|---|---|---|---|---|---|---|
| CLIP-RN50 | 70.37 | 39.42 | 59.03 | 51.18 | 37.72 | 31.87 | 17.31 |
| Distilled RN50 | 69.85 | 31.63 | 57.97 | 38.22 | 32.72 | 20.97 | 8.25 |
| CLIP-RN101 | 72.33 | 45.27 | 61.70 | 59.92 | 43.07 | 37.93 | 23.73 |
| Distilled RN101 | 72.28 | 35.18 | 60.46 | 44.09 | 35.89 | 23.88 | 11.56 |
| CLIP-ViT-B/16 | 79.40 | 57.59 | 69.72 | 72.42 | 51.85 | 47.33 | 46.64 |
| Distilled ViT-B/16 | 73.58 | 37.14 | 62.45 | 44.43 | 35.52 | 23.83 | 19.47 |
| CLIP-RN50x4 | 76.18 | 51.45 | 65.83 | 64.80 | 48.74 | 42.19 | 35.67 |
| Distilled RN50x4 | 76.25 | 41.40 | 65.20 | 49.22 | 42.71 | 29.23 | 20.64 |
| CLIP-RN50x16 | 80.24 | 60.61 | 70.13 | 73.67 | 56.92 | 48.52 | 53.79 |
| Distilled RN50x16 | 79.65 | 48.26 | 68.49 | 55.03 | 48.90 | 32.93 | 35.97 |

### E.2. iWildCam-WILDS

**Dataset.** We used the official version of the dataset provided by WILDS (Koh et al., 2021).

**Pre-trained models.** In order to obtain a feature extractor that exhibits sufficient generalization ability on the dataset, we explored different pre-trained models including ViTs in CLIP (Radford et al., 2021), RegNets in SWAG (Singh et al., 2022) as well as ResNets pre-trained on ImageNet (Deng et al., 2009). In the end, we chose a fine-tuned ResNet-50 (RN50) that is pre-trained on ImageNet as the teacher model since we observed that ImageNet-scale pre-training already leads to considerable robustness improvements compared to models trained from scratch on this dataset (also reported by Miller et al. (2021)), while being consistent to the network architecture used in the official WILDS repository. For linear probing, we freezed the parameters of the pre-trained model and trained a randomly initialized linear classification head using the hyperparameters provided by the official WILDS repository. The results are reported based on the model with the best OOD validation accuracy, following the protocol used by the WILDS paper (Koh et al., 2021).

**Representation distillation.** We freezed the weights of the teacher model and randomly initialized a ResNet-50 as the student model. We trained the student model by minimizing the Euclidean distance between its extracted representations and the representations extracted by the teacher model using the MSE loss on the training domains of iWildCam-WILDS. The student model was trained for 150 epochs using AdamW with batch size 128, learning rate 0.0001, and $\ell_2$ weight decay 0.1. In the linear probing stage, we freezed the parameters of the student model and trained a randomly initialized linear classification head using the hyperparameters provided by the official WILDS repository. The results are reported based on the model with the best OOD validation accuracy, following the protocol used by the WILDS paper.

**Baseline models.** We reported the results of baseline models provided by (Miller et al., 2021). In their result file, Miller et al. (2021) report both results for ImageNet-pre-trained neural networks (corresponding to models with `model_type` as "Neural Network" in the result file) and results for neural networks trained from scratch (corresponding to models with `model_type` as "ImageNet Pretrained Network"). Since our aim is to explore the limit of representation learning using only ID data, we omit the results of the models with pre-training.

**Detailed results.** We list detailed ID and OOD generalization performance of linear probes on pre-trained and distilled representations on iWildCam-WILDS in Table 2.

Table 2: Detailed ID and OOD Macro F1 of linear probes on pre-trained and distilled representations on iWildCam-WILDS.

| | ID Macro F1 | OOD Macro F1 |
|---|---|---|
| ImageNet RN50 | 49.30 | 32.46 |
| Distilled RN50 | 32.32 | 13.83 |

### E.3. Camelyon17-WILDS

**Dataset.** We used the official version of the dataset provided by WILDS (Koh et al., 2021).

**Pre-trained models.** After preliminary experiments, we chose a ViT-B/16 pre-trained by CLIP as our teacher model. For linear probing, we freezed the parameters of the pre-trained model and trained a randomly initialized linear classification head using the hyperparameters provided by the official WILDS repository. The results are reported based on the model with the best OOD validation accuracy, following the protocol used by the WILDS paper (Koh et al., 2021).

**Representation distillation.** We freezed the weights of the teacher model and randomly initialized a ViT-B/16 with identical architecture to the teacher model as the student model. We trained the student model by minimizing the Euclidean distance between its extracted representations and the representations extracted by the teacher model using the MSE loss on the training domains of Camelyon17-WILDS. The student model was trained for 120 epochs with batch size 128, learning rate 0.0001 and $\ell_2$ weight decay 0.1 using AdamW. For linear probing, we freezed the parameters of the student model and trained a randomly initialized linear classification head using the hyperparameters provided by the official WILDS repository. The results are reported based on the model with the best OOD validation accuracy, following the protocol used by the WILDS paper.

**Baseline models.** We reported the results of all algorithms from the offcial WILDS leaderboard (accessed at September 26th, 2023) that do not use custom data augmentation or pre-training (including "SGD (Freeze-Embed)" that uses CLIP pre-training and "ContriMix", "MBDG", "ERM w/ targeted aug" and "ERM w/ H&E jitter" that use custom, task-specific data augmentations) as baseline results.

**Detailed results.** We list detailed ID and OOD generalization performance of linear probes on pre-trained and distilled representations on Camelyon17-WILDS in Table 3.

### E.4. DomainNet

**Dataset.** Following the setup of Tan et al. (2020); Kumar et al. (2022), we used a pruned version of the original DomainNet dataset (Peng et al., 2019). The pruned dataset consists of 4 domains {Clipart, Painting, Real, Sketch} and 40 commonly occurring classes, selected from the original DomainNet which consists of 6 domains and 345 classes.

Table 3: Detailed ID validation and OOD test accuracy (%) of linear probes on pre-trained and distilled representations on Camelyon17-WILDS.

| | ID Validation Accuracy | OOD Test Accuracy |
|---|---|---|
| CLIP-ViT-B/16 | 98.22 | 92.88 |
| Distilled ViT-B/16 | 98.28 | 89.83 |

**Implementation details.** We adhered to the experimental settings as in DomainBed (Gulrajani & Lopez-Paz, 2021), which encompassed protocols for data augmentation, dataset partitioning, and hyperparameter search strategies. We opted for the widely adopted training domain validation for the model selection criterion. To reduce the computational cost, without loss of generality, we chose the Sketch domain with the largest distributional shifts as the test domain (OOD), while training on the other three domains (ID). For both our model and baseline models, we performed random searches on the hyperparameters with three different random seeds, each involving 5 trials.

**Pre-trained models.** We used the official ResNet-50 (RN50), ResNet-101 (RN101), and ViT-B/32 pre-trained checkpoints provided by CLIP.

**Representation distillation.** Due to limitations imposed by the scale of the dataset, we exclusively employed the CLIP-RN50 as the teacher model—it turns out in our preliminary experiments that distilling the other two pre-trained models results in *worse* performance both ID and OOD, which we believe is because the scale of the dataset is too small for distilling larger models. In the distillation stage, we freezed the pre-trained CLIP-RN50 as the teacher model and used the MSE loss to train the student RN50 model with the exact same structure as CLIP-RN50 but with random initialization. We used the AdamW optimizer with a cosine scheduler and learning rate 0.0003, $\ell_2$ weight decay 5e-5, batch size 32, and trained the student model for 95000 iterations. In the linear probe stage, we freezed the parameters of the student model and add a randomly initialized single-layer linear classifier. We trained the linear probe on the training sets of the three training domains and performed zero-shot evaluation on the test domain. We ultimately select the checkpoints with the highest accuracy on the validation set from the training domain. During this stage, we used the Adam optimizer (Kingma & Ba, 2015) with a cosine scheduler and learning rate 0.003, $\ell_2$ weight decay 1e-6, batch size 32, and trained the linear probe for 5000 iterations.

**Baseline models.** We generally followed the settings of DomainBed, with the exception of using a modified RN50 model with the same structure as CLIP-RN50 but randomly initialized. Additionally, we introduced a cosine scheduler with a warmup to enhance the convergence of models trained from scratch. We conducted extensive experiments with 15 representative domain generalization algorithms, including ERM (Vapnik, 1999), IRM (Arjovsky et al., 2019), GroupDRO (Sagawa et al., 2020a), Mixup (Zhang et al., 2018), MLDG (Li et al., 2018), Deep CORAL (Sun & Saenko, 2016), DANN (Ganin et al., 2016), SagNet (Nam et al., 2021), ARM (Zhang et al., 2021), VREx (Krueger et al., 2021), RSC (Huang et al., 2020), SelfReg (Kim et al., 2021), IB-ERM (Ahuja et al., 2021a), and IB-IRM (Ahuja et al., 2021a), and Fish (Shi et al., 2022). We increased the training iterations from the default 5000 to 20000 to ensure the convergence of all methods.

**Detailed results.** We list detailed ID and OOD generalization performance of linear probes on pre-trained and distilled representations on DomainNet in Table 4.

Table 4: Detailed ID test and OOD test accuracy (%) of linear probes on pre-trained and distilled representations on DomainNet.

| | ID Test Accuracy | OOD Test Accuracy |
|---|---|---|
| CLIP-RN101 | 92.30 | 87.34 |
| CLIP-ViT-B/32 | 92.35 | 87.60 |
| CLIP-RN50 | 87.02 | 82.58 |
| Distilled RN50 | 77.91 | 64.78 |

# F. Additional Experiments and Results

## F.1. Numerical Experiments

In this section, we present the results of our numerical experiments. The numerical experiments were conducted with parameters $d_{\text{core}} = d_{\text{bg}} = 32$, $d = 256$, $m = 256$, and $N = 1000$. During training, each $\mathbf{z}_i$, $i \in [d_0]$ was sampled from the uniform distribution on its support $[0, 1]$; during testing, each $\mathbf{z}_i$, $i \in \mathcal{S}_{\text{core}}$ was sampled from the same distribution as in training, while each $\mathbf{z}_i$, $i \in \mathcal{S}_{\text{bg}}$ was sampled from the uniform distribution on $[-1, 0]$. We considered two experimental settings:

- **Classification:** We trained a two-layer ReLU network to predict the binary label for each input, which matches our theoretical setting in Section 4. As an ablation, we also trained a two-layer linear network for the same task, replacing the ReLU functions in the network by identity functions.
- **Regression (representation distillation):** We trained a two-layer ReLU network to predict the vector $(\mathbf{z}_i)_{i \in \mathcal{S}_{\text{core}}}$ for each input—note that this is an optimal representation for OOD generalization, which matches the setting as our real-world representation distillation experiments in Section 2. As an ablation, we also trained a two-layer linear network.

In both settings, we trained the network using SGD with a learning rate 0.001 and a weight decay $\lambda = 0.001$. The results are in Figure 7, Figure 8, and Figure 9, which corroborate our theoretical results on

- **Activation asymmetry:** as shown by Figure 7, each neuron evolves to have positive correlations with at most one class of examples during training.
- **Feature contamination happens for non-linear networks:** as shown by Figure 8a (classification) and Figure 9a (regression), two-layer ReLU networks indeed accumulate weight projections onto the background features during training, leading to small ID risk yet large OOD risk.
- **Feature contamination does *not* happen for linear networks:** as shown by Figure 8b (classification) and Figure 9b (regression), two-layer linear networks does not accumulate weight projections onto the background features during training, leading to both small ID risk and small OOD risk when the concept class is linearly separable.

**Extensions to more general settings:** in Figure 3(d) in the main text, we provide numerical results of the average correlations between weights and background features for different activation functions. Here we detail our experimental settings and provide complete results in Figure 10. Concretely, we consider a *non-linear* relationship between core features and the label where core features for the two classes are distributed in a hyperball in $\mathbb{R}^{d_{\text{core}}}$ with radii 1.0 and 2.0, respectively. We consider four different activation functions, namely ReLU, GELU (Hendrycks & Gimpel, 2016), Sigmoid, and Tanh. We consider a two-layer network where both layers have trainable weights and biases. We use the AdamW optimizer (Loshchilov & Hutter, 2019) instead of SGD.

## F.2. Class Activation Histograms

In this section, we include average activation rate histograms for all blocks of ResNet-50 and ViT-B/16 as described in Section 5 in the main text. For every block in ResNet, we compute the mean activation rate for every class averaged over all channels in the final ReLU layer; for every block in ViT, we compute the mean activation rate for every class averaged over all channels in the GELU layer in its MLP. For every channel with the activation function $\sigma$ and pre-activation input $x$, the activation rate is defined by $\mathtt{rate}(x) := \begin{cases} 1, \text{ if } \sigma(x) \geq 0 \\ 0, \text{ otherwise} \end{cases}$, where $\sigma$ is ReLU for ResNet and GELU for ViT.

See Figure 11 for histograms of ResNet-50 blocks and Figure 12 for histograms of ViT-B/16 blocks. For earlier blocks in ResNet-50, activation asymmetry is not exhibited at random initalization but exhibited after training; for later blocks in ResNet-50 and all blocks in ViT, activation asymmetry is exhibited both at random initialization and after training.

## F.3. Neuron Selectivity

In this section, we detail the *neuron selectivity* metric that we adopted to produce the results in Figure 5 in the main text. Concretely, to examine whether the property of activation asymmetry also holds for deep neural networks, we adopt a similar selectivity metric as in (Morcos et al., 2018) to quantify the difference of neuron activation magnitudes between different classes. For a $C$-way multi-class classification problem and for a neuron, we denote the class-conditional mean activation after the nonlinearity for each class by $\mu_1, \dots, \mu_C$. In other words, each $\mu_i \in \mathbb{R}$ is calculated by averaging the activation of all inputs that belong to class $i$. Then, the *selectivity* of this neuron is defined as

$$\mathsf{Selectivity} := \frac{\mu_{\max} - \mu}{|\mu_{\max}| + |\mu| + \epsilon}, \tag{61}$$

where $\mu_{\max} = \max\{\mu_1, \dots, \mu_C\}$ and $\mu = \frac{1}{C}\sum_{i\in[C]} \mu_i$ denote the largest class-conditional mean activation and the mean activation of all classes, respectively. $\epsilon > 0$ is a small constant for numerical stability and we set $\epsilon = 1e^{-6}$ in our experiments. In practice, we compute each $\mu_i, i \in [C]$ by averaging over examples in mini-batches with a batch size of 1024 on the ImageNet validation set, and then averaging over all mini-batches. For CLIP-RN50, the reported neuron selectivity is averaged over all dimensions of the output of the last attention pooling layer. For CLIP-ViT-B/16, the reported neuron selectivity is averaged over all dimensions of the output of every GELU layer in the last attention block.

### F.4. Feature Visualization on CIFAR-10

To investigate the presence of feature contamination in real-world datasets, we conducted a experiment based on a variant of the CIFAR-10 dataset that is explicitly modified to incorporate background features that have *no correlation* with the label. Concretely, we augmented the CIFAR-10 training set by padding brick red pixels to the original images from CIFAR-10 and resized the padded images to the size of the original images, as shown in Figure 6(c). Since our padding does not impact the original image contents, it follows the "orthogonal" setting in our theoretical model where the core features (the original image contents) and the background features (the padded pixels) are independent—there exists a *linear* transformation in the *input space* that can fully recover core features and remove background features.

We then visualize the learned features of a ResNet-32 network trained on the original CIAFR-10 dataset and another ResNet-32 trained on our modified dataset. Following the visualization technique in Allen-Zhu & Li (2021), we performed adversarial training using the method proposed by Salman et al. (2019) and visualized the features learned by the network's convolutional kernels in the 31st layer using the same hyperparameters as described in Allen-Zhu & Li (2021). As shown by Figure 13, we observe notable differences in the learned color information between models trained on the original CIFAR-10 dataset and its modified variant. Meanwhile, we note that there are no obvious geometric patterns in the red areas, which we conjecture is due to the augmentations used during training such as random cropping and flipping. In general, the visualization results suggest that background features are indeed learned by deep neural networks despite having no correlation with the label, which corroborates our theory and indicates that *feature contamination also happens in* ***deep features*** *learned from real-world image data*.

## G. Empirical Evidence that Supports the Conjecture in Section 6

In this section, we provide preliminary empirical evidence that supports the conjecture stated in Section 6 in the main text and discuss its relation with related observations in recent work. For ease of presentation, here we restate this conjecture:

**Conjecture.** Pre-training on a sufficiently diverse dataset does not remove uncorrelated features, but *linearizes* those features in the model's representations, hence mitigating feature contamination and improving OOD generalization.

Table 5: Detailed ID test accuracy, OOD test accuracy, and domain classification error (%) of linear probes on pre-trained and distilled representations on PACS.

| | ID Test Acc. | OOD Test Acc. | Domain Classification Error |
|---|---|---|---|
| CLIP-ViT-B/16 | **99.68** | **91.59** | **0.06** |
| CLIP-RN50 | 97.35 | 85.67 | 0.19 |
| ERM-RN50 | 99.28 | 76.47 | 1.02 |

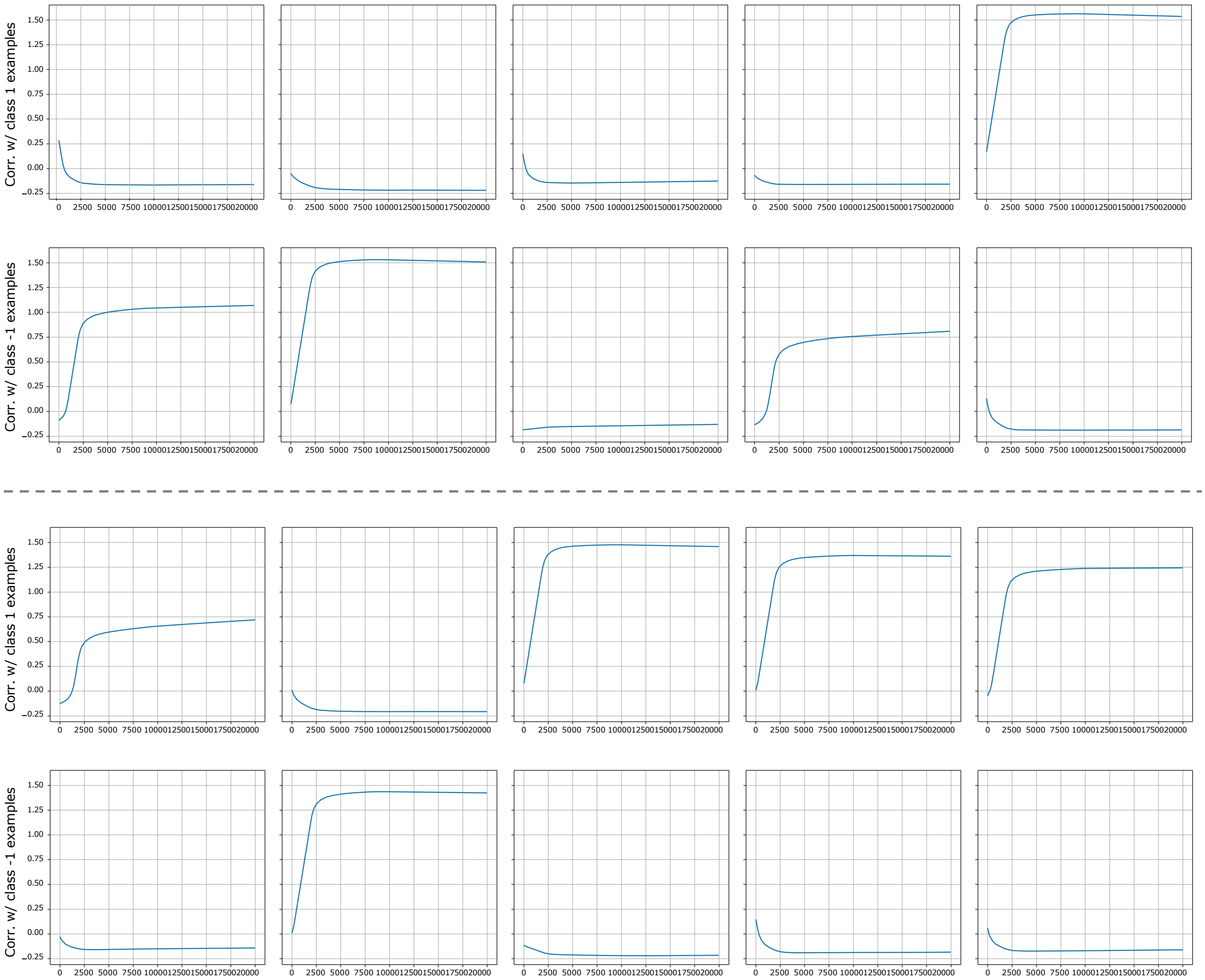
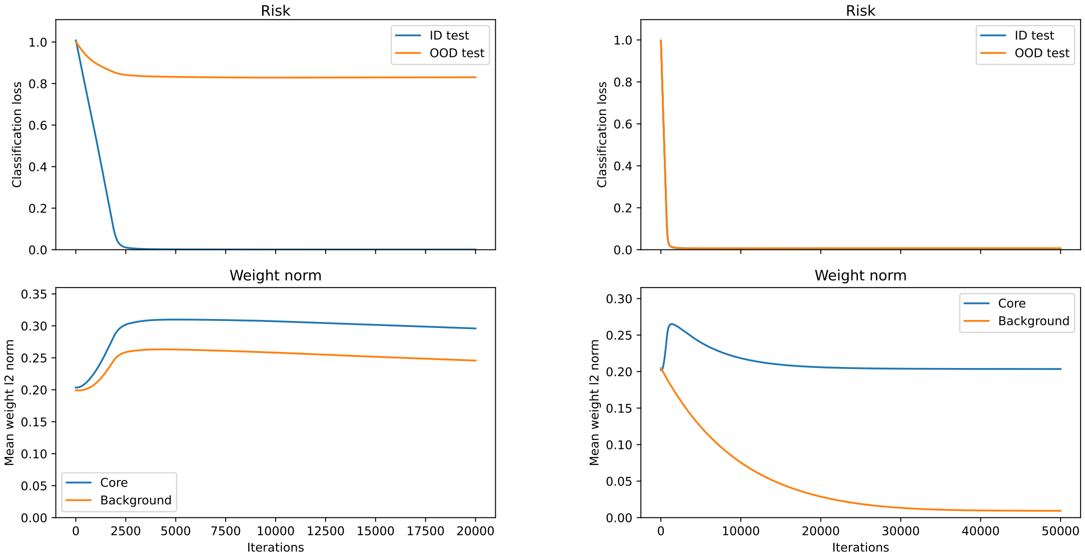


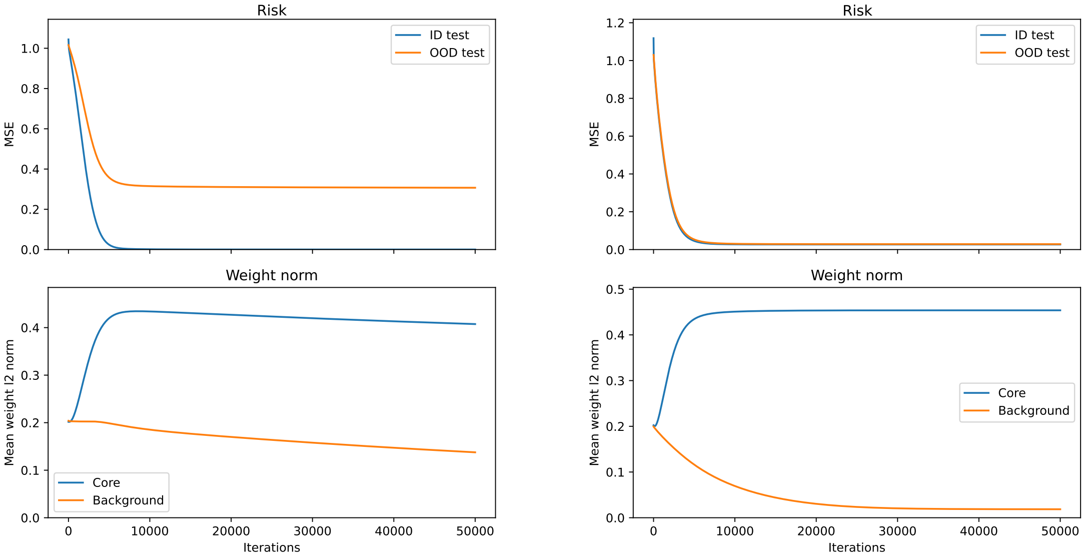
Figure 7: **(Activation asymmetry)** The average correlation between 10 random neurons and examples from both classes as a function of training iterations in the classification setting. In each column, the top plot above shows the average correlation between the weight (learned feature) of the neuron and the examples from class $y = 1$, while the bottom plot shows the average correlation between the weight (learned feature) of the neuron and the examples from class $y = -1$. As the training goes on, each neuron evolves to have positive correlation with at most one class of examples, resulting in activation asymmetry.



(a) Two-layer **ReLU** network

(b) Two-layer **linear** network

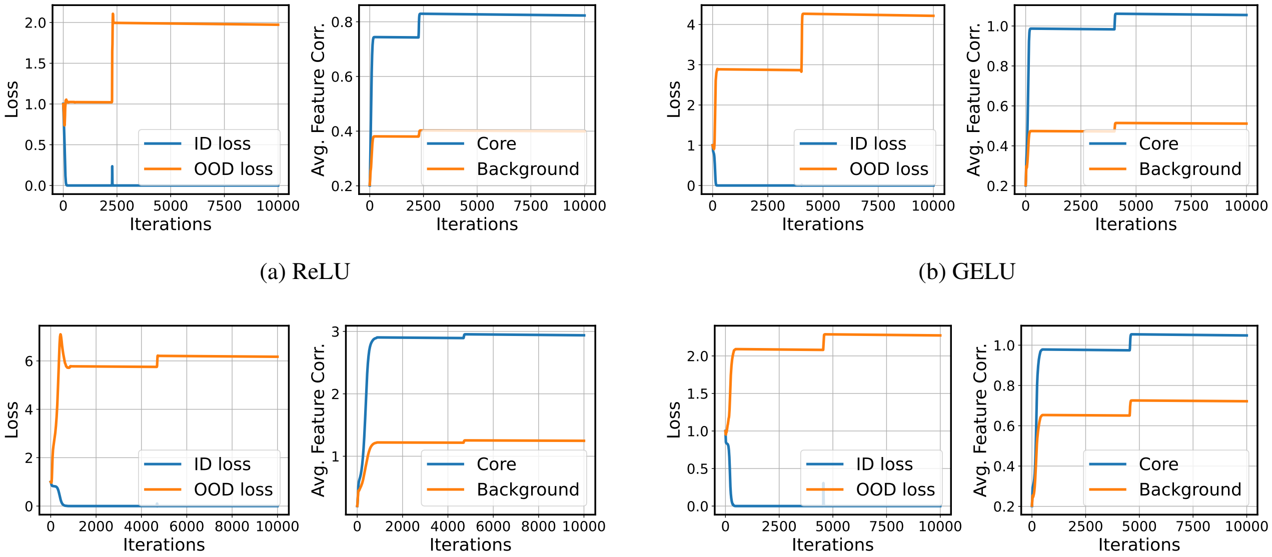
Figure 8: The ID and OOD risks (**top**) and the norm of weight projections onto core and background features (**bottom**) in the **classification** setting.



(a) Two-layer **ReLU** network

(b) Two-layer **linear** network

Figure 9: The ID and OOD risks (**top**) and the norm of weight projections onto core and background features (**bottom**) in the **regression** setting.



(a) ReLU

(b) GELU

(c) Sigmoid

(d) Tanh

Figure 10: Complete numerical results of the ID/OOD loss and average weight-feature correlations for **different activation functions**. Feature contamination occurs for all activation functions considered, resulting in large OOD loss.

### G.1. Large-Scale Pre-training Leads to Linear Separability of Domains

To empirically test this conjecture, we first examined the properties of the pre-trained representations from CLIP and the representations learned by ERM on a domain generalization dataset PACS (Li et al., 2017) for image classification. The images in PACS are divided into four domains, namely Photo, Art painting, Cartoon, and Sketch, with seven common categories. We trained a ResNet-50 ERM model using the examples from the first three domains (ID) and the Sketch domain was treated as the OOD domain. To evaluate the robustness of CLIP representations, we fitted a linear probe on top of freezed CLIP representations on ID domains and evaluated the learned linear probe on the OOD domain.

We begin by a 2-dimensional visualization of both the learned ERM representations and the CLIP representations using PCA dimensionality reduction. As shown in Figure 14, *ERM representations and CLIP representations exhibit quite different properties in terms of domain separability*: while examples from training and test domains are visually mixed in ERM representations, examples from training and test domains are *strongly linearly separable* in CLIP representations.

We then quantitatively examined this linear separability by fitting linear classifiers on top of ERM and CLIP representations for *domain classification*. Concretely, we trained linear classifiers with the original "class" label of each example substituted by its domain index. We then evalute the accuracy of this classifier on a hold-out validation set. As shown in Table 5, domain classifiers on CLIP representations have considerably smaller error than domain classifiers on ERM representations, which is consistent with visualization. This phenomenon is related to recent work on unsupervised domain adaptation based on contrastive learning (Shen et al., 2022; HaoChen et al., 2022), where it has been shown that contrastive learning can learn representations that disentangle domain and class information, enabling generalization that they refer to as "linear transferability" (HaoChen et al., 2022). However, their analysis requires that unlabeled examples from the target domain are seen by the contrastive learning algorithm during training, while large-scale pre-training in our context seems to achieve a similar disentangling effect even without explicitly trained on the target distribution. Further theoretical explanations of this phenomenon is an important future work.

In summary, the results in this section suggest that the representations learned by large-scale pre-training is highly linearized, with features representing different factors of variation not as non-linearly coupled as in our analysis on feature contamination. We believe that such high linearity of representations plays a critical role in the OOD capability of pre-trained models.

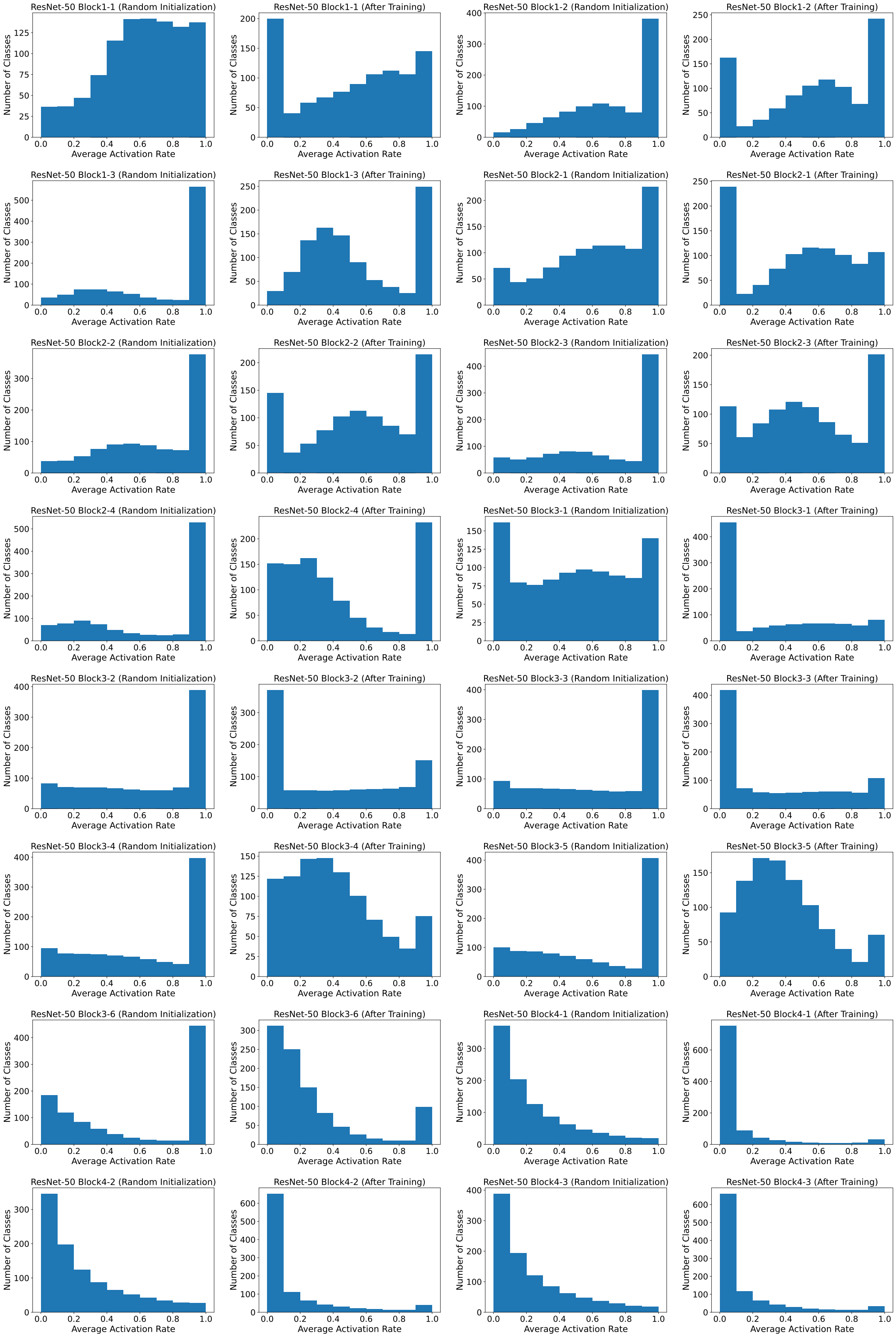


Figure 11: Activation rate histograms of all blocks in randomly initialized and trained ResNet-50 networks, computed from the ImageNet validation set.

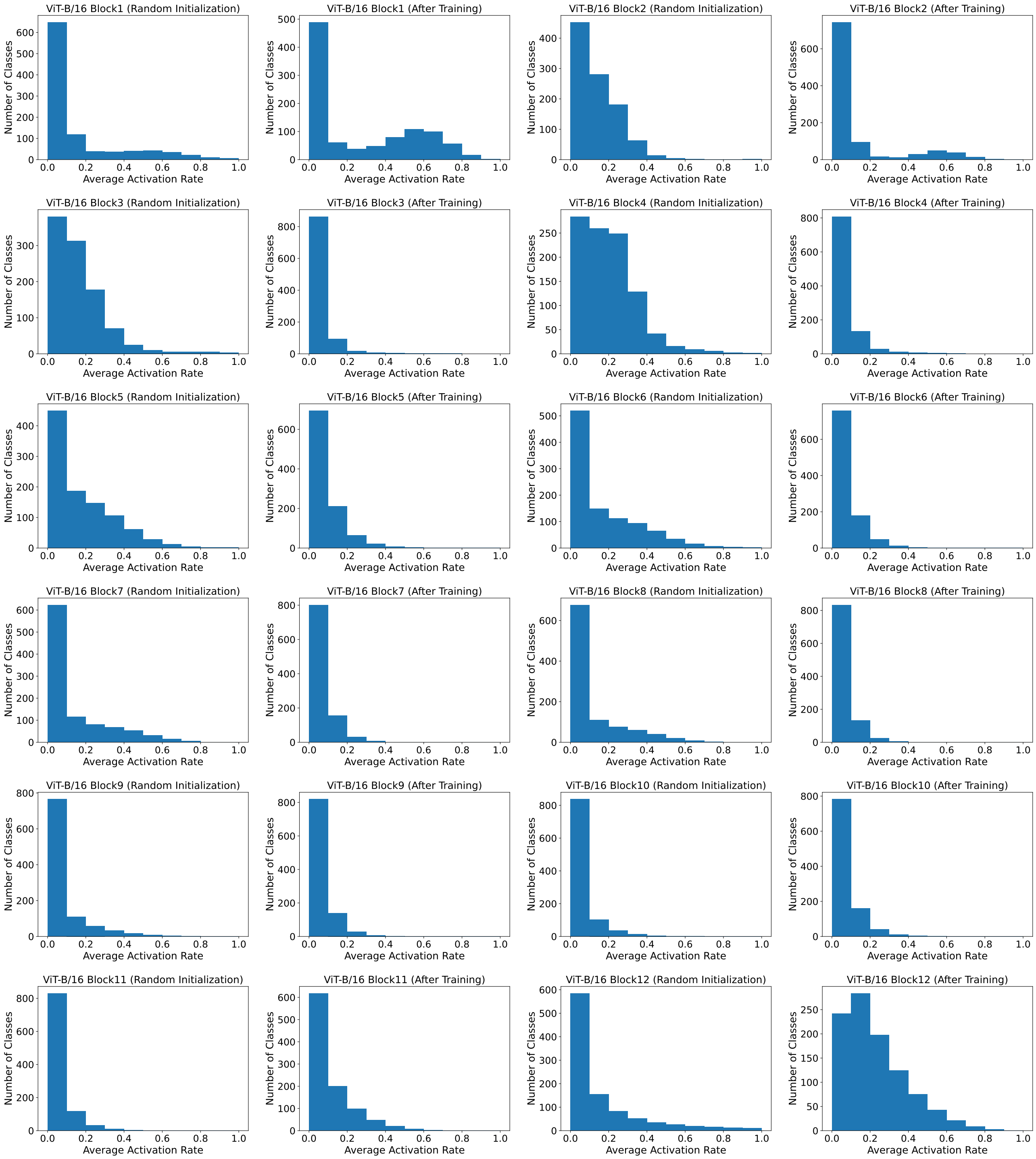
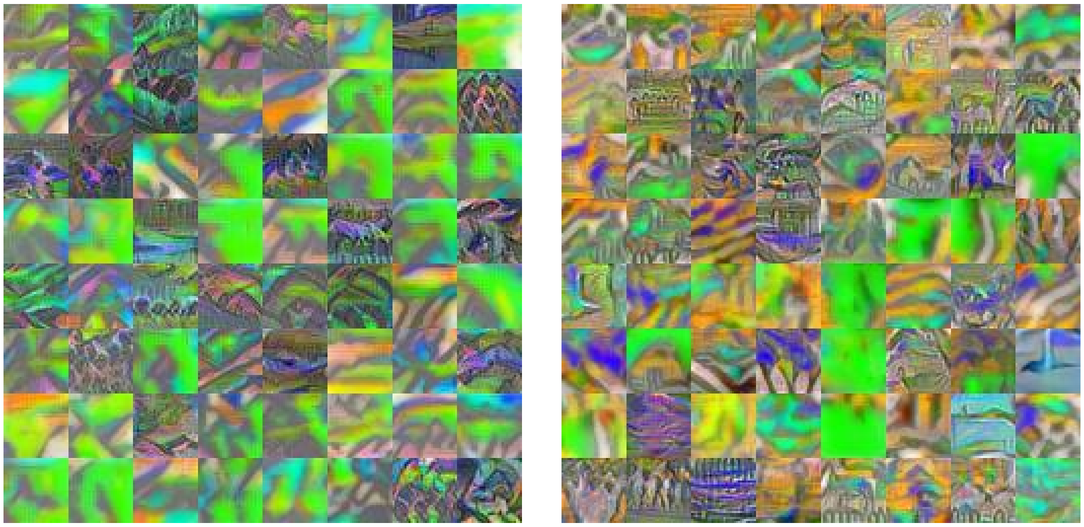


Figure 12: Activation rate histograms of all blocks in randomly initialized and trained ViT-B/16 networks, computed from the ImageNet validation set.



(a) Learned features of a ResNet-32 trained on the original CIFAR-10 dataset.

(b) Learned features of a ResNet-32 trained on the modified CIFAR-10 dataset with background features.

Figure 13: Additional visualization of the learned deep features in our CIFAR-10 experiment.

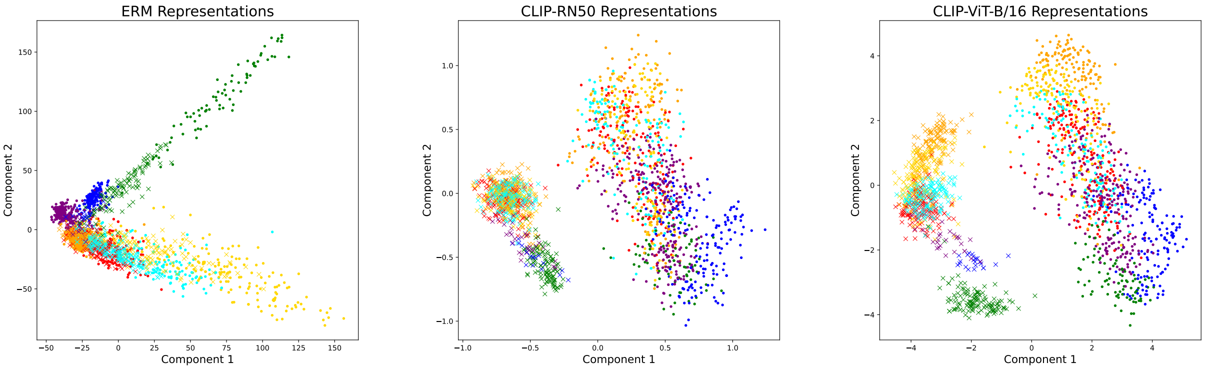
### G.2. Large-Scale Pre-training Leads to Denser Neuron Activation

In this section, we study property of pre-trained representations from another angle of neuron activation. As we have formally proved in Section 4, feature contamination causes the neurons to learn non-linearly coupled features. The activation of each neuron is thus likely to involve multiple feature vectors due to this coupling. By the above deduction, if pre-training alleviates feature contamination and learns more linearized features, then the activation of different feature vectors would be more likely to involve different neurons, resulting in an increase in the total number of activated neurons for each input.

Empirically, we confirmed the above hypothesis by calculating the histogram of the neuron's expected activation value in pre-trained and distilled models from the ImageNet experiments in Section 2. We considered the CLIP-RN50 teacher model and its corresponding student model obtained from representation distillation, and maintained an estimate of the average activation value for each output ReLU activation in the first residual block during one evaluation run. We plot the histogram of the neuron's average activation value in Figure 15. As shown by the figure, the pre-trained CLIP model indeed have considerably denser neuron activation than the distilled model, even on the ID ImageNet validation set where their top-1 accuracy is nearly the same (70.37% for the pre-trained CLIP model and 69.85% for the distilled model). This suggests that pre-trained models learn more "decoupled" features than models trained solely on the ID data.

### G.3. More Discussion on Related Work

**Explaining the distributional robustness of CLIP.** Understanding the remarkable distributional robustness of large-scale pre-trained models such as CLIP is an open research problem of its own. Due to the amount and diversity of pre-training data, a major confounder in this problem is that pre-trained models may have "seen" similar examples in standard distribution shift test sets during pre-training rather than be essentially robust to unseen distribution shifts. Recently, Mayilvahanan et al. (2024) conducted controlled experiments suggesting that even if we remove examples that are semantically similar to those in OOD test sets during pre-training, CLIP still remains a large portion of its distributional robustness. Therefore, CLIP must have achieved good OOD performance in a non-trivial way by learning OOD generalizable representations rather than simply memorize the test distribution. Gandelsman et al. (2024) shows that CLIP's image representations can be decomposed as sums across individual image patches, layers and attention heads that are often *property-specific*. Such "decomposability" in the representations implies that CLIP may represent different semantics in the input images in a decoupled way, which may be free from feature contamination. However, a rigorous connection between this observation and the linearity of features remains to be explored.



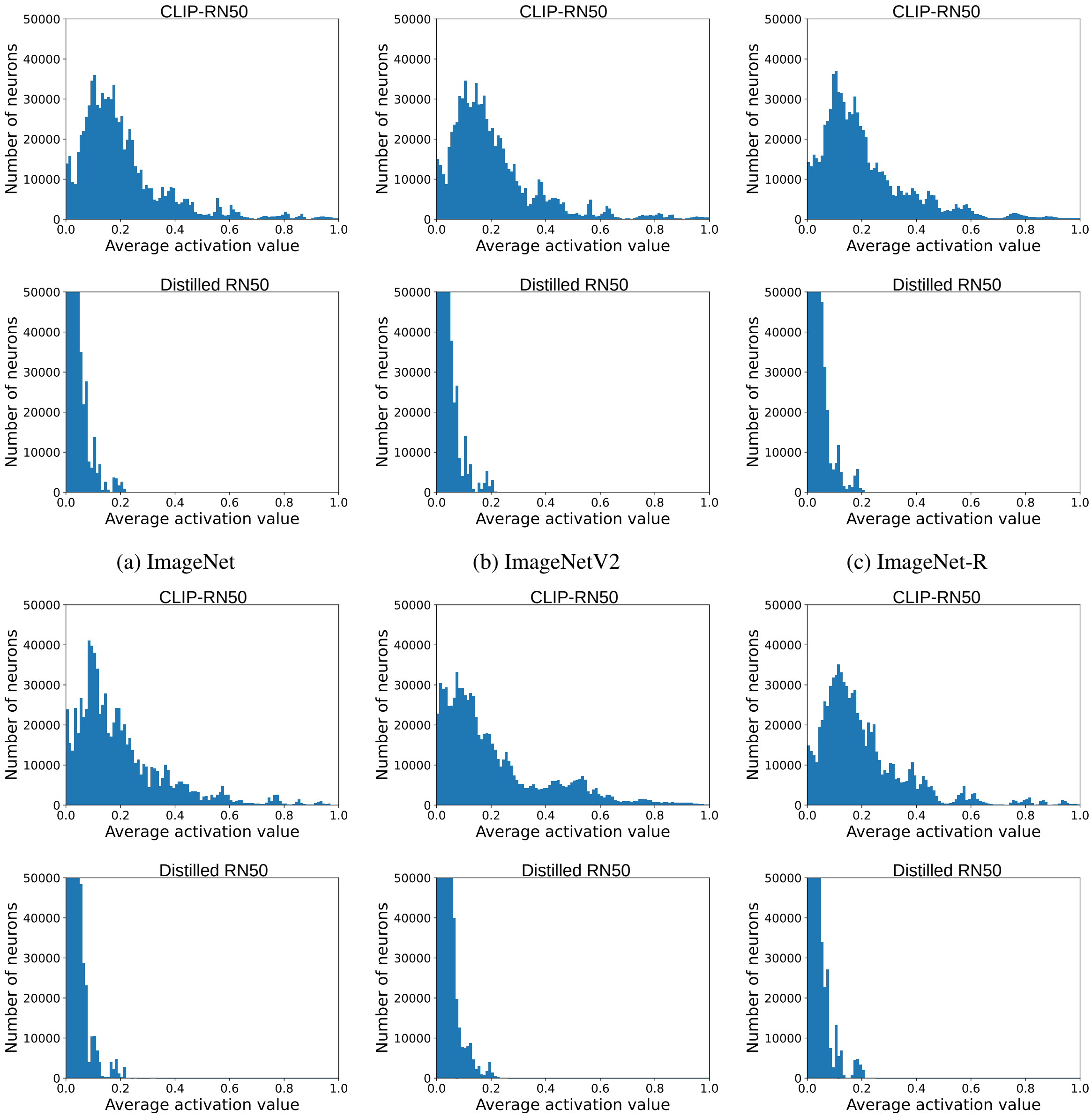
(a) ERM representations

(b) CLIP-RN50 representations

(c) CLIP-ViT-B/16 representations

Figure 14: Visualizations of ERM and CLIP representations after PCA dimensionality reduction to two dimensions. Circles refer to image representations in the training domains, while crosses refer to image representations in the test domain. Different colors represent different classes. Compared to ERM representations where the examples from training and test domains are visually mixed, CLIP representations exhibit strong *linear separability* of different domains.

**The linear representation hypothesis.** An important observation made by recent work on interpreting the representations of large language models (LLMs) is that many high-level, abstract concepts are *linearly* represented in the LLMs' intermediate activation spaces (Marks & Tegmark, 2023; Allen-Zhu & Li, 2023a; Gould et al., 2023; Park et al., 2023; Heinzerling & Inui, 2024; Gurnee & Tegmark, 2024). At a high level, those results are related to our conjecture in Section 6 on the *feature linearization* effect of pre-training. However, it remains an open problem how pre-training leads to such effects. Another closely related concept in the literature is *superposition* (Elhage et al., 2022), which hypothesizes that neural networks may represent more independent features than the number of neurons in the network by assigning different features to the same neuron when those features are sparse and correlated with the task. By contrast, we show that neural network can also learn *uncorrelated* features even when it has enough neurons to represent all features separately.



(a) ImageNet

(b) ImageNetV2

(c) ImageNet-R

(d) ObjectNet

(e) ImageNet Sketch

(f) ImageNet-A

Figure 15: Histograms of average neuron activations of a pre-trained CLIP-RN50 and a distilled CLIP-RN50 on ImageNet distribution shift datasets. In each subfigure, the top plot shows the histogram of CLIP, and the bottom plot shows the histogram of the distilled model.