THE UNIVERSITY OF CHICAGO

PRIVACY AND SECURITY IN RESPONSIVE ENVIRONMENTS

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY
CHRISTIAN ROBERTO CIANFARANI

CHICAGO, ILLINOIS
AUGUST 2026

ORCID iD: https://orcid.org/0000-0002-7318-7764

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I owe an immense debt of gratitude to my advisor, Aloni Cohen, without whom I would not be the researcher I am today. I also thank the members of my thesis committee, Blase Ur and Tian Li, and all those who mentored me throughout my PhD, particularly Arjun Bhagoji and Aravindan Vijayaraghavan. Lastly, I thank my friends and family who supported me and kept me company during my years in Chicago.

ABSTRACT

Private and secure systems require thorough evaluations to ensure that they are providing adequate protection while maintaining the utility of the underlying system. However, these evaluations frequently neglect to account for the ways in which the defense itself affects the environment in which it is deployed. In this thesis, we investigate this pitfall in the study of three defense techniques: adversarial training, machine unlearning, and differential privacy. In one work, we show the limitations of assuming a fixed adversary when training adversarially robust models and design a theoretically-motivated training scheme that provides protection when adversaries are allowed to modify their attacks to evade existing defenses. In another, we relax the assumption that deletion requests are independent when individuals are allowed to delete their data from AI models and explore how different user behaviors can impact the utility of downstream models. Finally, we examine the US Census Bureau's implementation of differential privacy in the 2020 census and ask whether the addition of privacy preserving noise will make it more difficult for states to comply with federal redistricting law. These contributions highlight the subtle ways in which private and secure systems alter the environments in which they are deployed.

CHAPTER 1

INTRODUCTION

The design of private and secure systems requires understanding threats as they exist in the real world. When designing these systems, we model the behaviors of adversaries are modeled and conduct thorough evaluations to ensure that realistic threats are adequately defended against. However, strong defenses rarely last forever. A system that is resilient to even the strongest adversaries may begin to fail soon after it is publicly deployed. This is sometimes due to technological or theoretical advances that enable new attacks that could not have been anticipated. However, in other cases, a defense itself creates the conditions that lead to its failure.

Take, for example, a lock on a door. Locks are typically installed to prevent those without keys from entering. In an ideal world, the lock would have no other effect. However, consider the unintended consequences that might result from installing a lock:

- Someone without a key might develop new lock picking techniques in an attempt to gain unauthorized access.
- The lock may be seen as an indication of something valuable behind the door, making it a target for thieves.
- People with keys may find themselves inconvenienced by having to unlock the door and instead opt to leave it propped open, giving access to those without keys.

In this scenario, the security provided by the lock was weakened because we failed to properly model how those who interacted with the lock would respond to its presence.

What a simplistic approach to threat modeling often ignores is the fact that environments can be *responsive* to the presence of privacy and security interventions. This responsiveness can lead to unintended consequences of deploying a defense. Adversaries could become

stronger, incentive structures could realign, and users might circumvent new systems. These effects, along with countless others, have the potential to negate the expected benefits of improved privacy and security. This thesis argues that, while responsiveness is a serious issue for privacy and security, careful evaluations and thoughtful designs can be used to anticipate and mitigate the environmental changes induced by privacy and security interventions to ensure long-term resilience.

These phenomena are related to a concept called *performativity*, which originally has its roots in the philosophy of language [Austin, 1975, Searle, 1975]. In this context, performativity refers to how utterances can not only describe reality but also be drivers of change. The notion of performativity has been extended to describe how scientific theories can themselves change the systems that they are attempting to model. This concept has been studied in fields such as economics [MacKenzie et al., 2020], network theory [Healy, 2015], and machine learning [Perdomo et al., 2020]. In the areas of privacy and security, we propose the following definition for the performativity of a defense technique, taking particular inspiration from the concept of performative prediction introduced by Perdomo et al. [2020]:

Definition 1 (Performativity of Defenses)**.** *A defense technique is* performative *if it induces changes in the environment in which it is deployed.*

We leave this definition rather broad in recognition of the complicated and unpredictable nature of the social and technical systems within which defenses are deployed. Although this language is rarely used specifically in the context of privacy and security, we posit that a large majority of defenses deployed in the real world could be considered performative under this definition. This thesis studies cases in which the performativity of a defense causes a degradation in the effectiveness of that defense. This kind of feedback can arise in unexpected ways, which complicates its study. For instance, a defense can induce non-adversarial changes in behavior can negatively impact a system.

This thesis analyzes the performative aspects of privacy and security through three case

studies:

Adversarial Machine Learning. Adversarial training has shown promise in defending ML models against adversarially perturbed inputs. However, adversaries frequently try to evade defenses, and it has been shown that adversarial training provides little protection when assumptions about the capabilities of the adversary are violated. We use data-driven methods to better understand how threat models impact internal representations of data that are learned by neural networks. Building on these results, we design a theoretically-motivated regularization approach for adversarial training which can adapt to new and evolving threats as they are encountered after deployment. This chapter is based on work published in Cianfarani et al. [2022] and Dai et al. [2025].

Machine Unlearning. Machine unlearning has been proposed as a remedy for the use of unauthorized data in the training of AI models. However, the behavior of the public when they are given the ability to request their data be deleted is underexplored in the literature. For example, people may strategically coordinate their decisions to unlearn in order to influence an AI model, or they may decide to unlearn only after seeing a new model update. We study how different models of individual and group behavior might affect the final model after unlearning. For instance, groups that collectively make decisions to remove subsets of the training data may have different impacts than individuals independently deciding whether or not to request unlearning. We show that, in certain settings, there are fundamental gaps between the capabilities of data subjects to influence the final model under these different behavioral assumptions. This chapter is based on work currently in preparation.

Redistricting with Private Data. The accuracy of official statistics is vital for many use cases, but releasing overly precise data can expose individuals to privacy harms. After the US Census Bureau adopted differential privacy in 2020, it drew criticism from many

who argued that the new data aren't be fit for use in many applications. We study the fear that differential privacy will alter the a way that redistricters draw legislative maps and lead states to inadvertently enact maps that are in violation of federal redistricting law. We run a large scale empirical study to evaluate this claim and clarify the effect that differential privacy has on redistricting. Using these empirical findings, we develop statistical models that help us understand the impacts of differential privacy and show how these impacts can be minimized using simple mitigation techniques. This chapter is based on work published in Cianfarani and Cohen [2026].

CHAPTER 2

MULTI-ATTACK ROBUSTNESS

2.1 Introduction

The rise of AI has brought with it a host of new security concerns. While much attention is focused on the threats posed by sophisticated attackers to cutting edge AI systems, it remains difficult to defend relatively simple models against relatively simple adversaries. One foundational task in ML security that remains unsolved is defending against *adversarial examples*: test-time inputs which have been modified in such a way that cause disagreement between humans and ML classifiers.

The defense technique that has shown the most promise in addressing this threat is *adversarial training* [Madry et al., 2018, Goodfellow et al., 2014]. Adversarial training reformulates the minimization objective of the standard training process as a min-max objective. Whereas standard training minimizes loss over a set of training samples, adversarial training minimizes the maximum loss of any point within a neighborhood of each training sample. This neighborhood around a sample is traditionally intended to define the set of points that are perceptually indistinguishable from that sample (see Section 2.2.1 for a more thorough explanation). We refer to the adversarially chosen point within this neighborhood as a *perturbed sample* or an *adversarial example* interchangeably, and the difference between the perturbed sample and the original sample as an *adversarial perturbation*. In our discussion of adversarial training, we refer to models trained using methods that account for adversarial examples as *robust* models and those trained without these methods as *non-robust* models. Similarly, we refer to accuracy on non-perturbed input as *benign accuracy* and on adversarial examples as *robust accuracy*.

While simple in principle, we run into issues when we attempt to formalize the underlying threat model. One issue is the capabilities of the adversary, particularly what types of adver-

sarial neighborhoods we define around input data. As stated above, prior works have tended to focus on imperceptible perturbations, opting to define adversaries that are restricted to adding perturbations with bounded ℓ_p norm (typically for $p \in \{1, 2, \infty\}$). In the image domain, this represents low magnitude, high frequency noise which doesn't meaningfully change the image when viewed from a moderate distance.

The standard formulation of adversarial training has proved useful for defending models against prespecified adversaries. Standardized evaluation suites such as AutoAttack [Croce and Hein, 2020b] have allowed comparisons between robust models and have shown a clear improvement in the efficacy of new adversarial training methods. However, these improved defense mechanisms belie some of the more subtle difficulties raised by the threat of adversarial examples in the real world. For one, the capabilities of an adversary are not necessarily known in advance. In the literature, adversaries are often modeled as being able to add perturbations to inputs with bounded ℓ_p norms in a way that is imperceptible to the human eye. In reality, perturbations may not be bounded have bounded ℓ_p norms, and may instead recolor images, introduce compression artifacts, or spatially modify images in minimal ways. These perturbations need not even be imperceptible, as long as they do not change the semantic meaning of an image. Furthermore, the capabilities of the attacker may change over time. An adversary might notice that their attacks are unsuccessful and change their approach to evade existing defenses. A meaningful defense must account for both uncertainty about the adversary's capabilities and the possibility of adaptive adversaries.

Broadening the scope of adversarial training in this manner brings with it new questions. For one, is it even possible to defend against arbitrary unconstrained adversaries? Some adversaries may simply be too strong to defend against in any meaningful sense. There may be other sets of adversaries against which it may be possible to defend individually but not all simultaneously. Our work does not attempt to fully solve the problem of adversarial training. Instead, we focus on a more constrained problem called *multi-robustness*: a model's

ability to be robust to multiple types of adversarial perturbations (as defined by different neighborhoods). We seek to better understand the complexities of multi-robustness and make improvements on existing methods.

Our work addresses these problems from the perspective of *representation learning*. By *representation*, we mean the neuron activations within a model induced by a given input (or batch of inputs). Under this framework, we characterize adversaries not by their inherent properties but by the patterns they induce in model representations. The main hypothesis we explore in this chapter is: *if a model has learned similar representations of two attacks, then it will display similar levels of robustness to those attacks.* Our work consists of two parts. First, we study the validity of this hypothesis from both a data-driven and theoretical perspective. Then, we use insights about robust representations in order to design a new adversarial training method that accounts for uncertain and changing adversaries. Our main research questions are described below.

2.1.1 Research Questions

Understanding Adversarial Examples Our first task was to better understand how learned representations of adversarially-perturbed samples relate to downstream adversarial robustness. We approached this through two analytical frameworks: one data-driven, and one theoretical in nature. Our work was the first to apply representation similarity techniques to the problem of adversarial robustness. Our empirical results demonstrate the usefulness of this approach for understanding certain properties of adversarially-trained models, like transferability, model capacity, and robust overfitting. However, when applied to the problem of multi-attack robustness, similarities between attacks don't correlate strongly with robustness in models adversarially trained against multiple attacks.

In a second attempt, we look towards learning theory to better understand robust

representations. Our theoretical results directly tie a model's robust accuracy against multiple different attacks to the model's internal representations of those attacks. Put simply, if the representations of two attacks are close in parameter space, then the model will have a similar level of robustness to both attacks. Not only does this result provide formal guarantees of robustness, it also suggests regularization techniques that can be used to enhance the multi-attack robustness.

Defending Against Multiple Adversaries We apply our insights about multi-attack robustness to a setting called *continual adaptive robustness* (CAR), first introduced by Dai et al. [2024]. Using insights from our theoretical results, we design a regularization-based adversarial training approach which can efficiently update models to be robust to new adversaries as they are encountered in the wild. We demonstrate the improved performance of our technique over baselines and highlight the need for additional research in this area.

2.2 Background: Adversarial Machine Learning and Neural Network Representations

2.2.1 Adversarial Machine Learning

Adversarial Examples: Adversarial examples are perturbed inputs to machine learning models that intentionally induce misclassification [Biggio et al., 2014, Szegedy et al., 2014]. For an input $x \in \mathcal{X}$, an untargeted adversarial example with respect to a hypothesis $h \in \mathcal{H}$ and loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is generated by adding a perturbation δ such that

$$\delta = \underset{x+\delta \in C(x)}{\arg\max}\ \ell(h(x+\delta), y),$$

Network	Benign accuracy			Robust accuracy		
	Train	Test	Δ	Train	Test	Δ
Non-robust	100.0%	94.9%	5.1%	0.0%	0.0%	0.0%
Robust	100.0%	87.6%	12.4%	98.2%	44.2%	54.0%

Table 2.1: **Baseline Model Performance.** Performance of non-robust and robust WRN-28-10 networks trained on the CIFAR-10 dataset. In particular, the robust network exhibits a large gap between benign and robust test accuracy, as well as high robust generalization gap (the difference between robust accuracy on the train and test set, as captured by Δ).

where y is the label of the original class and $C : \mathcal{X} \to \mathcal{X}$ represents the adversarial constraint or neighborhood. These constraints are often formulated as a bound on the ℓ_p-norm of the perturbation [Madry et al., 2018, Carlini and Wagner, 2017], although other types of constraints have been studied in the literature, such as spatial transformations [Xiao et al., 2018], color shifts [Laidlaw and Feizi, 2019], JPEG compression and weather changes [Kaufmann et al., 2019], bounded Wasserstein distance [Wong et al., 2019, Wu et al., 2020] as well as attacks based on distances that are more aligned with human perception such as SSIM [Gragnaniello et al., 2021] and LPIPS distances [Laidlaw et al., 2021, Ghazanfari et al., 2023].

Robust Learning: Adversarial training [Madry et al., 2018] has emerged as the most effective methods to train robust classifiers. It modifies the empirical risk minimization procedure in non-robust training to a min-max optimization, where the inner maximization focuses on finding the strongest adversarial example for each input during training:

$$\min_{h \in \mathcal{H}} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\max_{x' \in C(x)} \ell(h(x'), y) \right], \tag{2.1}$$

where $\mathcal{D}$ is a data distribution and h is a model in hypothesis class $\mathcal{H}$.

Adversarial training induces a trade-off between robustness to adversarial examples and accuracy on unperturbed samples, as we show in Table 2.1. Though adversarially trained models are more robust to adversarial examples than models trained with standard methods, they usually have significantly lower accuracy on benign data. A more advanced method,

TRADES [Zhang et al., 2019], attempts to better balance this tradeoff by directly minimizing loss at the original training sample while incorporating a term which encourages smoothness within the adversarial neighborhood. The optimization problem becomes the following:

$$\min_{h \in \mathcal{H}} \mathop{\mathbb{E}}_{(x,y)\sim D} \left[\ell(h(x), y) + \max_{x' \in C(x)} \ell(h(x), h(x'))/\lambda \right], \tag{2.2}$$

where λ is a regularization parameter. Various other improvements to adversarial training have been proposed, including architectural improvements [Wang et al., 2023b, Peng et al., 2023], ensembling techniques [Bai et al., 2024, Pang et al., 2019], and utilizing synthetic training data [Sehwag et al., 2022, Gowal et al., 2021].

Multi-Robustness. The property that this chapter is primarily concerned with is called *multi-robustness*: a model being robust to multiple types of adversarial perturbations (as defined by distinct constraints on adversarial neighborhoods). Despite the wide variety of attacks that have been introduced, defenses against adversarial examples focus mainly on ℓ_∞ or ℓ_2-norm bounded perturbations [Cohen et al., 2019, Zhang et al., 2020, Madry et al., 2018, Zhang et al., 2019, Croce et al., 2020].

A few prior works have studied the problem of achieving robustness against multiple attacks, under the assumption that all attacks are known a priori. These include training based approaches [Maini et al., 2020, Tramèr and Boneh, 2019a, Madaan et al., 2020] which incorporate adversarial examples from the threat models of interest (usually a combination of ℓ_1, ℓ_2, and ℓ_∞ norm bounded attacks) during training. Croce and Hein [2020a] provides a robustness certificate of all ℓ_p norms given certified robustness against ℓ_∞ and ℓ_1 attacks.

Another line of work has looked at defending against attacks that are not known by the defender, a problem known as *unforeseen robustness*. These techniques are all training-based and include Laidlaw et al. [2021] which proposes training based on LPIPS [Zhang et al., 2018], a metric more aligned with human perception than ℓ_p distances, and Dai et al. [2022], Jin and Rinard [2020] which use regularization during training in order to obtain better

generalization to unforeseen attacks. Dai et al. [2023] provides a comprehensive leaderboard for the performance of existing defenses against a large variety of attacks at different attack strengths.

Our work differs from these lines of works since we assume that while the defender may not know all attacks a priori, they are allowed to adjust their defense when they become aware of new attacks. The work most similar to ours is Croce and Hein [2022], which proposes fine-tuning a model robust against one ℓ_p attack to be robust against the union of ℓ_p attacks for $p \in \{1, 2, \infty\}$. Specifically, they demonstrate that one can achieve simultaneous multi-attack robustness for the union of ℓ_p attacks by obtaining robustness against ℓ_1 and ℓ_∞ attacks, and thus propose fine-tuning with ℓ_1 and ℓ_∞ attacks to achieve this efficiently. Our work differs from this work since we explore adapting to attacks outside of ℓ_p attacks, investigate ways of improving the initial state of the model prior to fine-tuning, and consider adapting to sequences of attacks.

2.2.2 Probing Neural Network Representations

Throughout this chapter, we refer to *neural network representations*: the activations of neurons at a specified layer of a network when given a batch of data as input. These representations take the form of n by p matrices, where n is the number of data samples in the batch and p is the number of neurons in the layer, and represent the output of the the layer after the activation function has been applied. We characterize activations of internal layer in a network as *benign representations* if the input batch is not adversarially perturbed and *adversarial representations* if the input batch is an adversarially perturbed.

Representation similarity (RS) metrics allow for the quantitative comparison of the activations of different sets of neurons on a given dataset. Comparing learned, internal representations within and between neural networks of different architectures is important for understanding the performance and behavior of these networks under changing conditions.

Several metrics have been proposed for this purpose: Canonical Correlation Analysis (CCA) [Hardoon et al., 2004] and its extensions, Singular Vector CCA (SVCCA) [Raghu et al., 2017] and Projection Weighted CCA (PWCCA) [Morcos et al., 2018], Centered Kernel Alignment (CKA) [Kornblith et al., 2019], the Orthogonal Procrustes distance [Ding et al., 2021] and model stitching [Bansal et al., 2021, Csiszárik et al., 2021]. These metrics can all handle different sized representations, but require that the number of datapoints be the same.

We choose CKA to measure representation similarity due to its desirable properties such as orthogonal and isotropic invariance, and wide use in previous work [Neyshabur et al., 2020, Nguyen et al., 2020, Raghu et al., 2020, Vulić et al., 2020]. Ding et al. [2021] have shown that CKA is nevertheless insensitive to deletions of principal components of representations that strongly correlate with model accuracy. However, this property does not impact the manner in which we use CKA. Appendix A.2.1 contains experiments with other RS metrics, yielding similar discoveries.

Centered Kernel Alignment (CKA): Let $X \in \mathbb{R}^{n \times p_1}$ with rows x_i, and $Y \in \mathbb{R}^{n \times p_2}$ with rows y_i be the activation matrices of layers with p_1 and p_2 neurons respectively, usually defined on the same set of n points.[1] These activation matrices are then the representations induced by a particular layer. We also define a $n \times n$ centering matrix $H = I_n - \mathbf{1}\mathbf{1}^\top$. The CKA between these activation matrices is then

$$\mathrm{CKA}(K, L) = \frac{\mathrm{tr}(KHLH)}{\sqrt{\mathrm{tr}(KHKH)\mathrm{tr}(LHLH)}}, \tag{2.3}$$

where $K_{ij} = k(x_i, x_j)$, $L_{ij} = l(y_i, y_j)$, with k and l being kernels. When the kernels are linear and the activation matrices are centered, we get the Linear CKA between activation

1. In our experiments, we sometimes extract activations using two different sets of data. Additional details of this approach are in Section 2.2.3.

matrices to be

$$\text{Linear CKA} = \frac{\|Y^\top X\|_F^2}{\|X^\top X\|_F \|Y^\top Y\|_F} = \frac{\langle \text{vec}(XX^\top), \text{vec}(YY^\top) \rangle}{\|X^\top X\|_F \|Y^\top Y\|_F}. \tag{2.4}$$

Since Kornblith et al. [2019] find the use of a linear kernel to be as accurate as and much faster than other standard kernels, we focus only on linear kernels. Going forward, all references to CKA will refer specificaly to linear CKA. To reduce memory usage, we use online CKA which uses batching to get an unbiased estimate of CKA [Nguyen et al., 2020].

2.2.3 Using CKA with Neural Network Representations

A visualization technique we frequently utilize is the cross-layer similarity plot, as illustrated in Figure 2.2. These plots show the CKA similarity between every pairing of layer activations between two deep neural networks h_1 and h_2. Within one of these plots, the pixel at row i and column j therefore represents the CKA similarity between the activations of the i^{th} layer of h_1 and the j^{th} layer of h_2. This technique allows for a fair amount of flexibility in the types of comparison that we can make. For instance, the models h_1 and h_2 might be the same model, in which case the plot would be displaying self similarity. We may also alter the input data that is being fed in to either model. In our experiments, we often display similarities between benign and robust representations, as well as between robust representations that have been perturbed by different types of adversary. To disambiguate the forms this plot can take, we define the following categories:

- **Self Similarity Plot:** Both axes represent the same activations.
- **Architecture Similarity Plot:** The two axes represent activations of models with different architectures. In our experiments, these are typically different widths of the WideResNet architecture (i.e. WRN-28-1 compared to WRN-28-10).
- **Training Similarity Plot:** The two axes represent activations of models that were

trained using different training methods. Oftentimes, one model will be subject to standard training and the other to robust training, but both could be adversarially trained with respect to different attacks. In some plots, we compare representations from a network at different points in the training process (i.e. comparing representations after the first epoch of training to representations after the last epoch of training).

- **Attack Similarity Plot:** The two axes represent activations of models whose input data represent different types of adversarial perturbations. Oftentimes, one of the axes will represent benign activations and the other adversarially perturbed activations, but both could be attacked with different types of adversarial perturbations.

Occasionally our experiments will compare models that are different in more than one of the qualities described above. In this case, we will describe the resulting plots as a combination of the categories (i.e. an Architecture-Training similarity plot). Additionally, we will refer to representations that correspond to unperturbed input samples as *benign representations*, and those that correspond to adversarially perturbed input samples as *adversarial representations*. Unless specifically noted, representations being compared in our experiments can be assumed to be benign.

2.3 Research Questions and Methods

Our overarching goal is to understand how models defend against different types of adversaries. As explained above, our initial approach involves the use of representation similarity metrics, in particular Linear CKA. While this approach is straightforward in application, no prior work had examined the usefulness of RS metrics in adversarial settings. Our study of this topic thus involved answering two main questions:

- Are RS techniques useful for studying robust learning *in general*?
- Can RS techniques guide us towards defenses in multi-robust settings *in particular*?

In this section, we describe our approach for analyzing robust models, inspired mainly by the experiments of Kornblith et al. [2019]. We then introduce the architectures, datasets, training methods, and threat models we will be studying in later sections.

2.3.1 Experimental Setup

Models and datasets. We consider three commonly used image datasets: CIFAR-10 [Krizhevsky et al., 2009], ImageNette [Howard, 2020], and ImageWoof [Howard, 2020], where latter two are subsets of ImageNet dataset [Deng et al., 2009], which we use due to the high cost of adversarial training on the full ImageNet dataset. We use the Wide Resnet architecture [Zagoruyko and Komodakis, 2016] of models on CIFAR-10 dataset and the ResNet architecture [He et al., 2016] for ImageNette and Imagewoof datasets.

Adversarial training. We follow standard convention [Croce et al., 2020] with ℓ_∞ perturbations and use $\epsilon = \frac{8}{255}$ for CIFAR-10 and $\epsilon = \frac{4}{255}$ for ImageNet based datasets, where ϵ represents the bound on norm of the adversarial perturbation. We use a 10-step projected gradient descent (PGD) attack to generate adversarial examples during training and use 20 steps when generating adversarial examples at test time, similar to the approach of Madry et al. [2018].

Threat models. Most of our robust models were trained and evaluated on ℓ_∞ adversarial images. To supplement our results, we also ran experiments using conventional ℓ_2 attacks, as well as three adversarial attacks included in the Imagenet-UA framework [Kaufmann et al., 2019]: JPEG, Snow, and Gabor. The JPEG attack adds ℓ_p bounded adversarial noise to the JPEG-encoded space of images, rather than the pixel space of images. The Snow attack adds adversarially optimized visual occlusions to images that simulate snowfall. The Gabor attack optimizes the parameters of a Gabor kernel that is used to noise the image. Visual examples of these attacks are shown in Figure 2.1. Unless otherwise noted, for our CIFAR-10 experiments we run the ℓ_2 attack with $\epsilon = 1$, the JPEG attack with $\epsilon = 25$, the Snow attack

with $\epsilon = 2$, and the Gabor attack with $\epsilon = 75$. Table 2.2 has the training and validation accuracies for models robustly trained using these threat models.

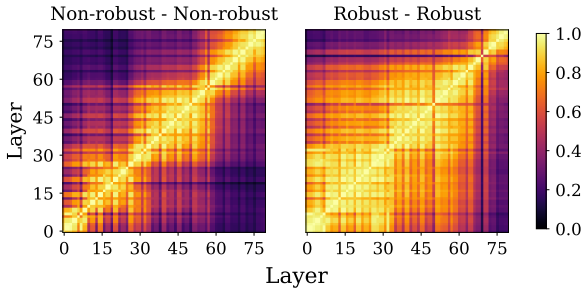
Figure 2.1: **Visualization of Adversarial Examples.** Adversarial examples generated by the threat models studied in this section.

	Benign Images		Adversarial Images	
Threat Model	Train Accuracy	Val. Accuracy	Train Accuracy	Val. Accuracy
Base	100.00	94.92	-	-
ℓ_∞ ($\epsilon = \frac{8}{255}$)	100.00	86.36	99.03	46.34
ℓ_2 ($\epsilon = 1$)	100.00	86.49	99.40	45.76
JPEG ($\epsilon = 25$)	100.00	84.65	91.98	45.52
Gabor ($\epsilon = 75$)	100.00	93.12	99.83	92.47
Snow ($\epsilon = 2$)	100.00	90.23	95.72	67.35

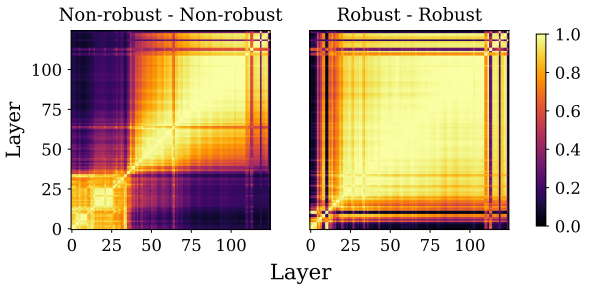
Table 2.2: **Baseline Model Robustness.** Training and validation accuracies of the WRN-28-10 models used in this paper's experiments. The threat model represents the attack that each model was trained to be robust against, and adversarial images were generated using the same threat model and attack strength as was used in training.

2.4 Observing the Effects of Robust Training on Representations

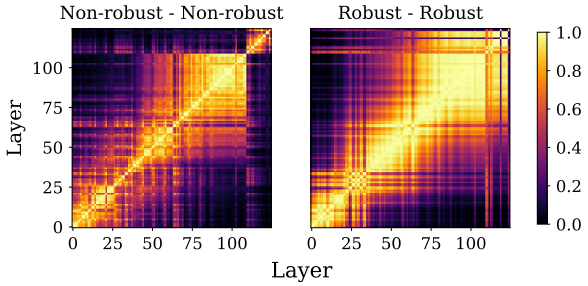
Robust training is expected to be a harder training objective than non-robust training, since it optimizes for both accuracy and robustness. The resulting trade-off is also reflected in final performance, where the benign accuracy of robust networks is much lower than non-robust networks [Madry et al., 2018]. In this section, we aim to understand the impact of robust training on internal representations, in particular by comparing benign representations in non-robust and robust networks. Further experiments and results from this section are included in Appendix A.2. We take the following two-fold approach:



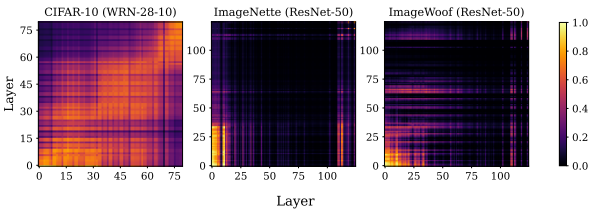
(a) Self Similarity (CIFAR-10)



(b) Self Similarity (ImageNette)



(c) Self Similarity (ImageWoof)



(d) Training Similarity: Non-Robust vs Robust Networks (CIFAR-10)

Figure 2.2: **Differences in Cross-Layer Similarity and Block Structure Between Non-Robust and Robust Networks.** We measure the similarity of benign representations across all pairs of layers in both non-robust and robust networks. *(a, b, c):* Robust networks have an amplified degree of self similarity, even even between distant layers, which leads to vanishing of the well known block structure characteristic observed in non-robust networks [Nguyen et al., 2020]. *(d)* We observe low similarity between representations of benign and robust models, when experimental setup is identical for both.

1) Comparing characteristics: Absence of block structure. When visualizing self similarity in non-robust networks, numerous works have revealed a 'block structure', with high similarity between the layers comprising a block, and low similarity outside it [Kornblith et al., 2019, Nguyen et al., 2020, Raghu et al., 2021] (Non-robust network plots in Figure 2.2). Nguyen et al. [2020] observe that this block structure arises when the model is overparamaterized with respect to the task being learned. In other words, when a network has much more capacity (as measured by number of paramaters) than is necessary to perform well on a certain task, a block structure will emerge in its self similarity plot. We observe that when moving from benign to robust training, while keeping everything else constant, the *block structure all but disappears.* The lack of a block structure in robustly trained networks

is accompanied by the presence of *long-range similarities* between layers, with layers having high similarity to those much farther away in the network (Figure 2.2).

2) Direct comparison: Poor similarity between representations in non-robust and robust networks. To further investigate the unique characteristic of robust networks, we conduct a cross-layer comparison of benign representations in non-robust and robust networks (Figure 2.2d). We find a very low degree of similarity across both network representations, which further shows that robust networks had indeed learned strikingly different representations than non-robust networks.

Our additional results in Appendix A.2 show that both of these observations generalize across architectures, datasets and threat models. We further explore the cross-layer similarity structure of robust networks representations.

Impact of robust training strength. We investigate this disappearance of block structure in robustly trained networks further by varying both the adversarial budget (ϵ) used for training and the width of the network (Figure 2.3). As the training ϵ increases, the block structure becomes fainter, eventually disappearing entirely. The block structure is also impacted by network width, with it being marginally more pronounced in larger networks. However, this distinction is largely absent at larger values of ϵ. This establishes that increasing the value of ϵ is making the task more complex, so maintaining the same model capacity leads to a decrease in relative capacity (indicated by high similarity across representations).

Impact of architecture. Previously Nguyen et al. [2020] observed that increasing network width, thus capacity, leads to the emergence of block structure in non-robust networks. Thus we incrementally increase the width of robust Wide-ResNet architecture on CIFAR-10 dataset and measure the cross-layer representation similarity of benign features (Figure 2.3, expanded on in Appendix A.2). Even after increasing network capacity by an order of magnitude, we don't see significant variation in the characteristics of robust network representations. These results suggest that increasing width in robust networks doesn't lead a

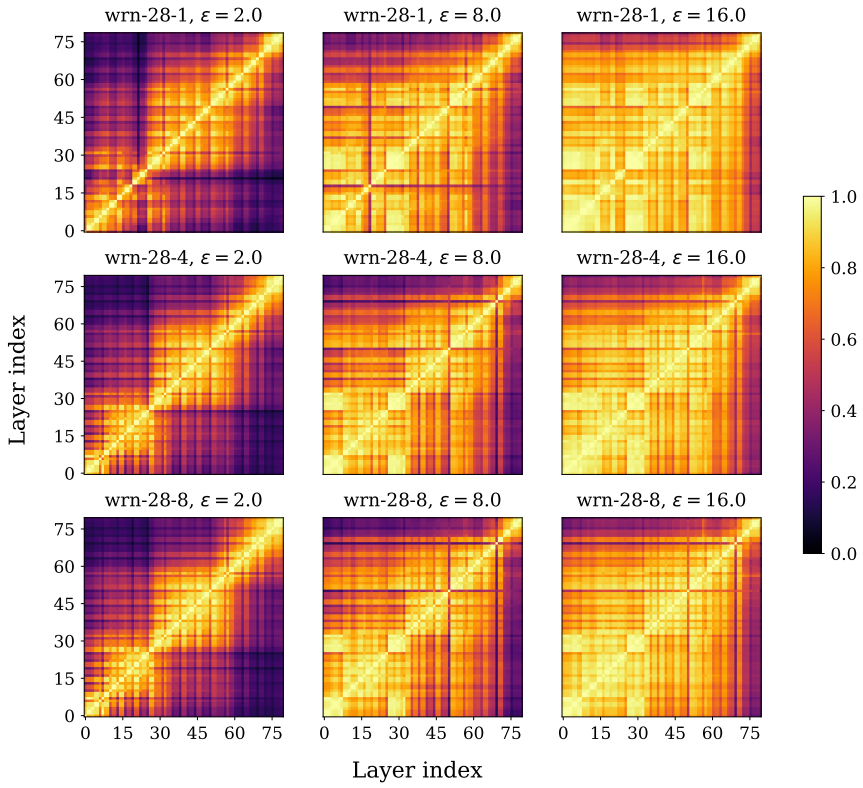


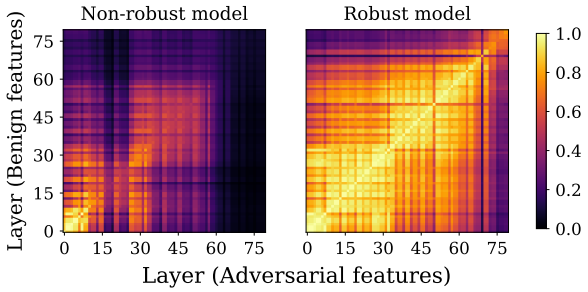
Figure 2.3: **Self Similarity of Different Architectures and Training Strengths.** To better understand the unique cross-layer characteristics in robust networks (Figure 2.2), we ablate along two main design choices: 1) Strength of adversarial attacks in robust training (by increasing perturbation budget (ϵ)) 2) Changing network size (by increasing network width). We observe that the strength of adversarial perturbations predominantly leads to amplification of cross-layer similarity (i.e. a higher degree of brightness in the plots). Increasing network capacity, even by an order of magnitude, doesn't substantially change this phenomenon.

drastic shift in similarity of internal layer representations.

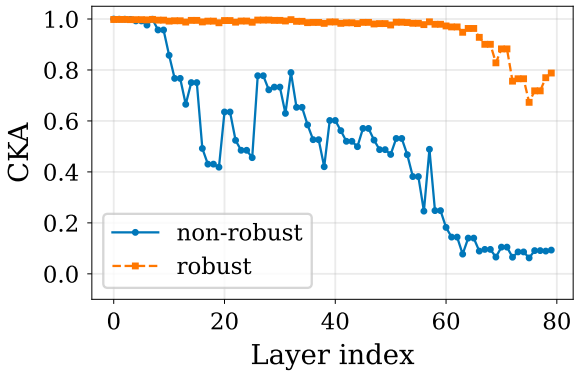
2.5 Explaining Robust Training with Representation Similarity

While it is clear that representations differ between robust and benign networks, it is not immediately obvious that these differences yield any meaningful insights about robust training. To establish the utility of our analytical framework, we show that the representational structure of robust networks can help us better understand well-documented properties of these networks. In particular, we show how our representation-based approach can rein-

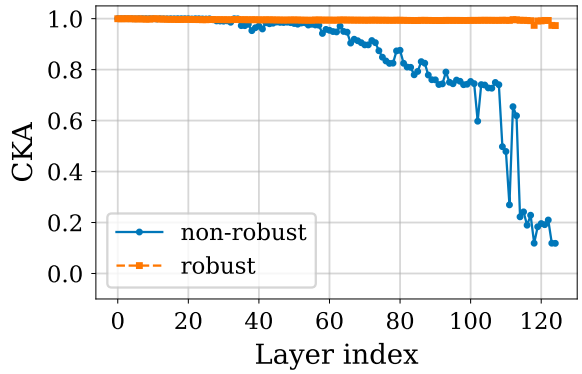
force and help explain properties of the internal subspaces of adversarial representations, the transferability of adversarial examples between models, and robust overfitting. Detailed results, as well as ablations on architecture and threat model, are available in Appendix A.3.



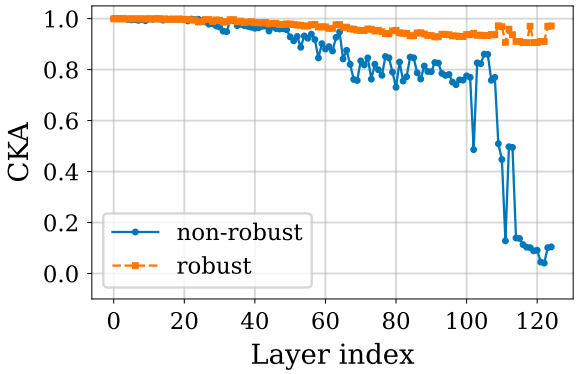
(a) Attack Similarity (Benign vs. ℓ_∞, CIFAR-10)



(b) CIFAR-10



(c) ImageNette



(d) ImageWoof

Figure 2.4: **Divergence of Benign and Adversarial Representations.** We compare how adversarial perturbations distort internal representations with respect to benign representations. In *(a)* we analyze attack similarity across all layers (comparing benign and ℓ_∞ representations), while in *(b, c, d)* we plot attack similarity of identical layers (i.e. just the diagonals of the corresponding cross-layer similarity plots) for different datasets. As expected, adversarial representations in non-robust networks are highly dissimilar from benign ones near the end of network. Unexpectedly, such divergence is very small for a non-trivial fraction of early layers (up to one-third). Robust training successfully reduces the impact of adversarial perturbations, thus leading to similar benign and adversarial representations in robust networks.

2.5.1 Complementing Findings on Adversarial Subspaces

Prior work on understanding robust learning has studied internal network activations from different perspectives than our RS approach. In particular, a number of works [Sabour et al., 2015, Wójcik et al., 2021, Ma et al., 2018] characterize how internal representations of adversarial examples progress through subsequent layers of a deep neural network. These works observed that the impacts of adversarial examples are more pronounced in later layers of a network than earlier layers. Below, we report similar results found using our approach.

A fraction of early layers are largely unaffected. In Figure 2.4, we present attack similarity plots comparing benign and adversarial representations for both non-robust and robust networks. If we narrow our focus to representations from the final layer of the networks, the CKA similarity between adversarial and benign representations decays to near zero. This is expected, as adversarial perturbation are crafted specifically to corrupt final layer representations. However, when we look at earlier layers, we observe that similarity between adversarial and benign representations is high for a significant fraction of the network (up to one-third for ImageNette and Imagewoof datasets). This observation holds across architectures and budgets, indicating that adversarial perturbations impart a stronger influence over later layers. While it would be tempting to claim that early layers of benign networks are therefore adversarially robust themselves, there is reason to doubt that the connection is so straightforward. Other works have been written which suggest that adversarial examples crafted in different ways can yield different activation patterns [Athalye et al., 2018] and that early layers are actually more important to a network's robustness than later layers [Bakiskan et al., 2022]. The results we present here are observational, and future work would be required to better understand how layers of different depths are impacted by adversarial training.

Network capacity influences the distinctiveness of robust representations. Within robust networks, benign and robust representations maintain a high degree of similarity for

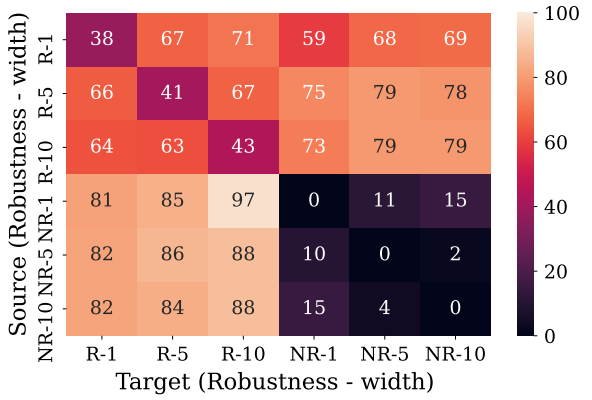
most of the depth, with divergence only occurring in the last 10 layers or so (see the orange lines in Figure 2.4). In our ablation on network size (Appendix A.2.4), we see that increasing the size of the network leads to increased differentiation, implying sufficient capacity for increased separability between benign and perturbed representations. This reinforces prior research finding that, more so than standard training, the task of adversarial training benefits from higher model capacity [Madry et al., 2018, Xie and Yuille, 2020, Wu et al., 2021].

2.5.2 Predicting Transferability of Adversarial Examples

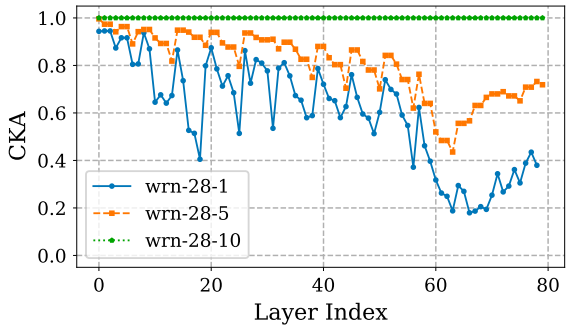
It is well-known that adversarial examples are frequently *transferable*: an adversarial examples crafted to fool one network can reliably fool others that were trained on the same task (but not necessarily the same dataset). In Figure 2.5a, we measure the transferability among non-robust and robust networks. Intriguingly, as previously observed by Croce et al. [2020], transferability is higher among some networks (non-robust to non-robust) while much poorer for other (robust to non-robust). High transferability points towards similarities in the decision boundaries learned by different models. It is therefore clear that robust training yields a meaningfully different decision boundary, particularly in areas where adversarial examples are clustered. We aim to better understand this phenomenon by comparing the internal representations of source and target networks. We first generate adversarial examples, and their corresponding internal representations, from a source network (WRN-28-10 in this case). We compare these with internal layer representations of these same adversarial examples on target network. We consider transfer from non-robust to robust, robust to robust, and robust to non-robust networks [2] (Figure 2.5).

For transfer from non-robust to non-robust or robust to robust, where the highest transferability occurs, we observe a high degree of similarity between representations at early layers. It indicates that the transferred adversarial examples distort the internal represen-

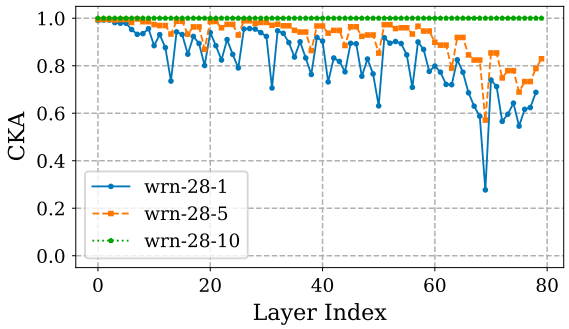
2. Robust networks are known to be highly resilient to adversarial examples generated from non-robust networks (see Figure 2.5a).



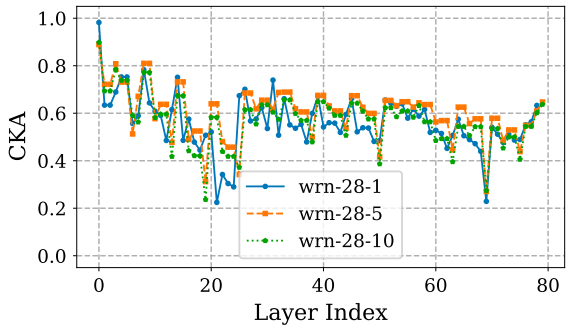
(a) Transfer accuracy of adversarial examples.



(b) Architecture-Training Similarity: Non-robust source (WRN-28-10), non-robust target.



(c) Architecture-Training Similarity: Robust source (WRN-28-10), robust target.



(d) Architecture-Training Similarity: Robust source (WRN-28-10), non-robust target.

Figure 2.5: **Visualizing Transferability with Architecture-Training Similarity.** *(a)* Measuring the robust accuracy of adversarial examples transferred among robust (R) and non-robust (NR) Wide-ResNet networks of varying width (R-x refers to Robust WRN-28-x). Low robust accuracy implies high transferability. *(b, c, d)* We measure RS between source and target networks, where these representations are extracted using adversarial examples generated for the source network. When both networks are trained in the same fashion (non-robust or robust), the layer wise similarity is generally high at earlier layers but drops off towards the end. In contrast, it drops right after the first layer from robust to non-robust network, a set of networks which also have least transferability.

tations in a similar fashion as source network. The most intriguing aspect in Figure 2.5a is poor transferability from robust to non-robust networks. This is reflected in significant differences in representations where, even between identical architectures, we observe a large drop in CKA right after the input layer. This suggests that adversarial perturbations generated from robust networks distort the internal representations of non-robust network in a strikingly different manner (even at very early layers). The change in the transfered adver-

sarial representations in non-robust networks may help explain why they have some degree of robustness to these adversarial examples despite not undergoing robust training.

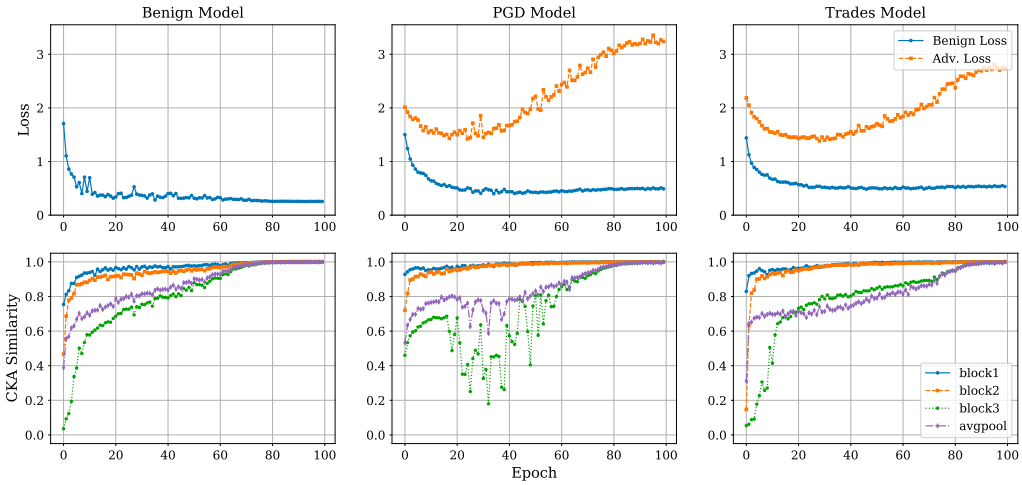


Figure 2.6: **Variation in the Convergence of Different Layers over the Course of Training.** We measure the training similarity between representations of different layers after each epoch of training with the representations of the same layers after 100 epochs of training. We plot results for non-robust and robust (both PGD-based and TRADES) training techniques. We observe that representations obtained after the first two blocks converge to their final state very early in the training process and have very little variation in both non-robust and robust network. We also find that later layers exhibit more complex behavior, with signs of overfitting when early stopping is not used, and a lack of stability from one epoch to the next.

2.5.3 Understanding Robust Overfitting

Previous research has shown that adversarial training is much more susceptible to *overfitting* than standard training [Rice et al., 2020, Kim et al., 2021, Yu et al., 2022]. During training, there will often be a point where robust loss on the test set will begin to increase while robust loss on the training set will continue to decrease. In this section, we study this phenomenon by tracking the evolution of robust representations learned over the course of training epochs in the optimization process. We study two convergence properties: 1) Rate of convergence at different layers, and 2) Stability of different layers throughout learning. As in earlier sections,

we use CKA to measure similarity, but in this section we *compare the benign representations of the same layer across time.*

We present our results for the WideResNet architecture (specifically WRN-28-10) trained for 100 epochs in Figure 2.6. This architecture consists of three groupings of residual blocks followed by a final average pool layer. To study representation evolution, we compare the representations at the final layer of each grouping after each epoch to those of the final epoch (epoch 100). We further investigate this evolution across architectures, different layers, and attack budgets in Appendix A.4.

Early layers converge faster than later layers. Benign representations after the first two blocks of WRN-28-10 are highly similar to the representation obtained at the end of training, after just ten epochs (Figure 2.6). This is true for all the training methods considered. In Appendix A.4, we show this largely holds true across architectures and smaller budgets. Our observation relates to the intuition that early layers learn simpler features [Krizhevsky et al., 2012, Yosinski et al., 2015, Zeiler and Fergus, 2014], thus are learned much faster than complex feature learned by later layers. Only with stronger attacks ($\epsilon = \frac{16}{255}$) do we observe a deviation from this trend where second block convergence is slower.

Overfitting is predominantly visible in later layers. From the cross-epoch heatmap of similarities between adversarial representations at different epochs (Appendix A.4), we observe that final layer representations change slowly at the beginning (until around epoch 30) and end of training, and change more quickly in the middle epochs. This clearly shows that multiple local minima are explored during training. This is true to a less pronounced degree for benign representations. In Fig. 2.6, we also observe the adversarial validation loss increasing after around 30 epochs, while the training loss keeps decreasing, indicating overfitting. After this point in training, the CKA similarity between the epoch 30 and final representation drops significantly in later layers, while the earlier layers remain unaffected. This suggests that the phenomenon of robust overfitting is localized to later layers.

Adversarial training can reduce stability of internal representations during training. As shown in Figure 2.6, while representations from subsequent epochs tend to be very similar to each other in non-robust training, in adversarial training there can be very large jumps in similarity between one epoch and the next, indicating a less stable convergence. This effect is further accentuated when training with a higher perturbation budget or when training a larger network (Appendix A.4). Intriguingly, we find that epoch to epoch stability of representations is significantly higher for TRADES training than standard adversarial training, which may help to explain the performance improvements associated with the method.

2.6 How Do Threat Models Impact Representations?

Up until this point, our experiments have been restricted to a single threat model: ℓ_∞-bounded perturbations. In this section, we begin to address the question of how different threat models affect internal representations. To this end, we consider *four* additional threat models: ℓ_2, JPEG, Gabor and Snow [Kang et al., 2019] (visualized in Figure 2.1). We also examine the generalizability of our earlier results across these threat models.

Do visually similar attacks lead to similar robust representations? RS gives us a method by which we can compare how different classes of adversarial perturbations affect learned representations. Using CKA, we plotted the training similarity of robust WRN-28-10 networks adversarially trained against five different threat models (ℓ_2, ℓ_∞, JPEG, Gabor, and Snow) (Figure 2.7). These comparisons lend credence to conventional knowledge about these threat models. The training similarity plot between l_2 and l_∞ robust models is strikingly similar to CKA plots between robust models trained on the same threat model. This mirrors theoretical results on correspondences between ℓ_p threat models [Tramèr and Boneh, 2019b]. We also observe a higher average similarity when comparing an ℓ_∞ model with a JPEG model than with Snow or Gabor models. This makes intuitive sense, given the

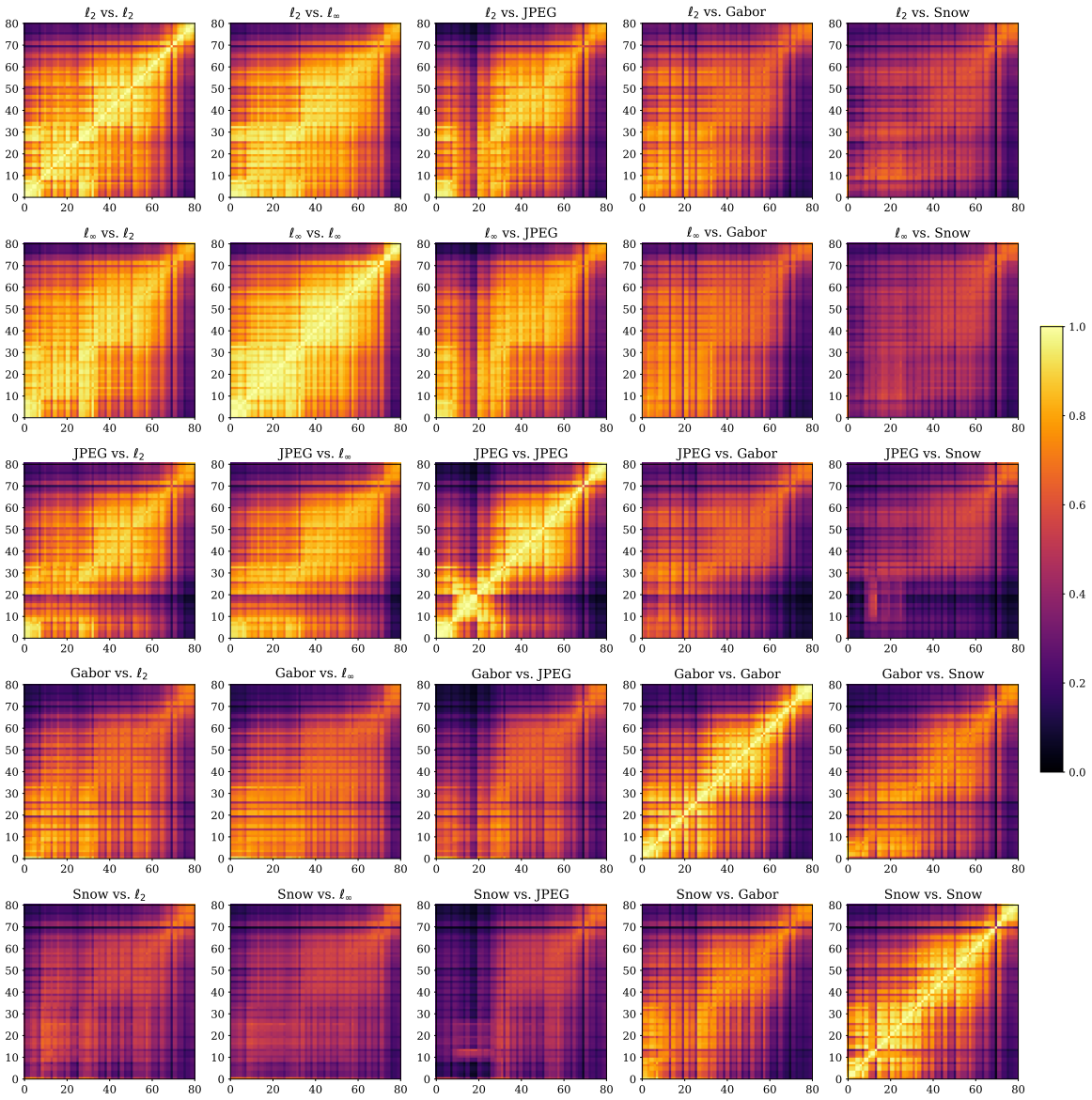


Figure 2.7: **Visualizing Relationships Between Threat Models.** Training similarity plots for robust WRN-28-10 models trained against different threat models. As suggested by prior research [Croce and Hein, 2021], ℓ_p robust networks are shown to be highly similar. JPEG, whose perturbation is ℓ_p bounded in the compressed space, also displays more structural similarities to the ℓ_p bounded attacks than to the others. Interestingly, Snow and Gabor trained models are noticeably more similar to each other than to the other models, implying an unexpected similarity between the two classes of attack.

ℓ_∞ constraint inherent in the JPEG attack. When using CKA to compare against the Snow threat model, we observe that the highest average similarity is achieved with Gabor. This

represents a novel insight into these threat classes, as correspondence between the two was not previously known.

Alignment of budgets across different attacks. CKA can also give insight into the varying effects of changing perturbation strengths within different threat models. Expanding on our experiments above, we compared representations between different threat models while varying perturbation strengths. Between ℓ_∞ and Gabor models (Figure 2.8), the layerwise similarity structure is much more sensitive to changes in the strength used in ℓ_∞-robust training than in Gabor-robust training. For more closely related attacks, like ℓ_∞ and JPEG (Figure 2.9), we see similar degradations of the structure when changing the strength of either attack. Furthermore, the structure is largely preserved when the attack strengths are changed in tandem. These results may be useful for developing techniques for training models to be jointly-robust against different classes of attack.

Layerwise structure across threat models. We find that across all threat models except Snow, robust and benign activations exhibit high similarity (figures in Appendix A.5). Further, as we change budgets across the same threat model, we find that ℓ_p threat models have representations that evolve more gradually.

2.6.1 Does Attack Similarity Predict Joint Robustness?

Returning to our hypothesis about joint robustness, we wish to understand whether CKA similarity between attack representations is correlated with a model's robustness to unseen attacks. In Figure 2.10, we present the robust accuracy of models trained on each attack with respect to all other attacks. Several patterns are present. For one, robust training against any of the ℓ_2, ℓ_∞, and JPEG attacks confers some degree of robustness against all of the attacks. Training against Snow or Gabor, on the other hand, confers negligible robustness against ℓ_2, ℓ_∞, and JPEG.

The correspondence between these patterns and those observed in Figure 2.7 is murky at

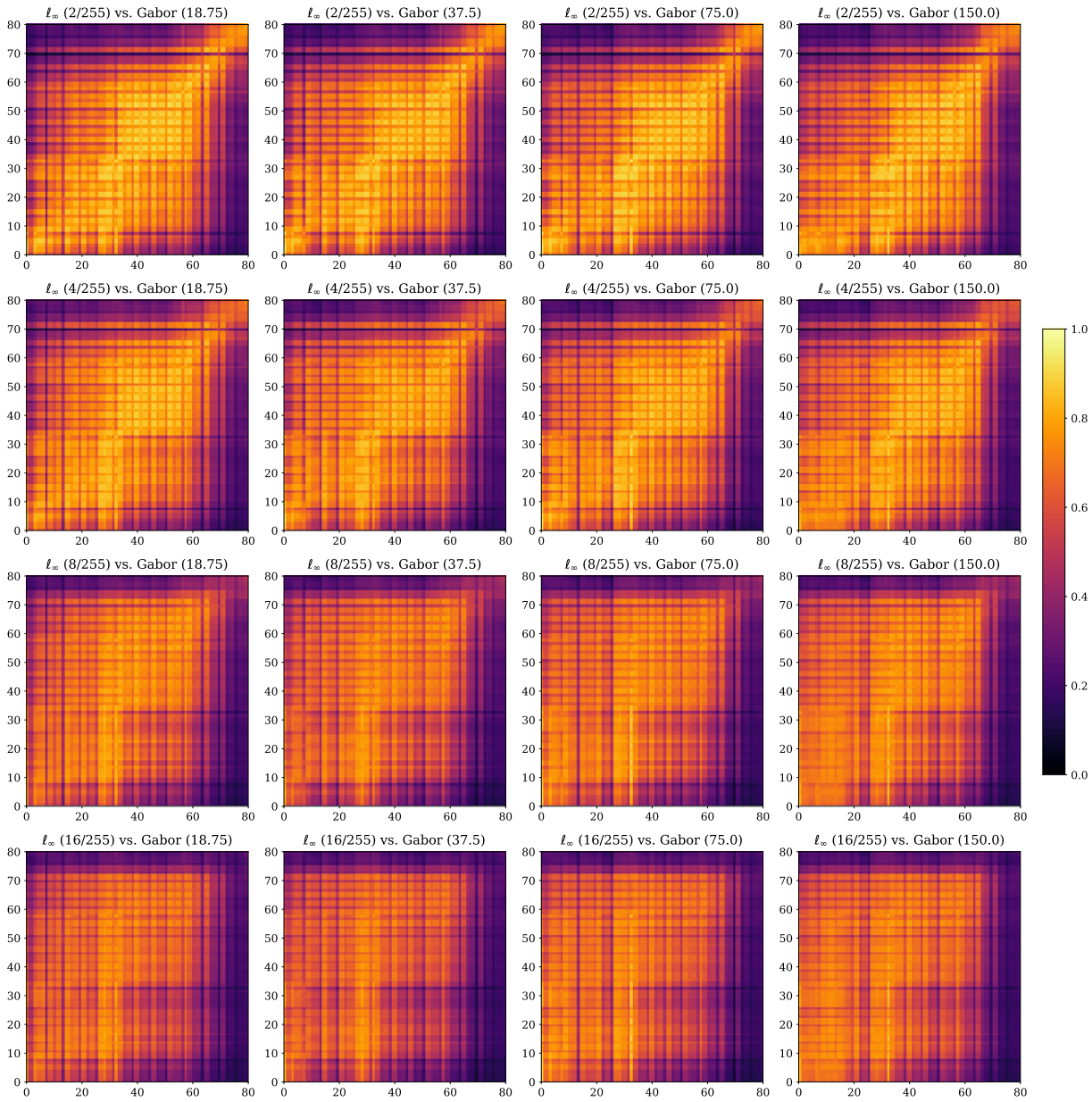


Figure 2.8: **Similarities Between ℓ_∞ and Gabor Threat Models.** Training similarity between ℓ_∞ and Gabor robust models trained against different attack strengths.

best. On one hand, the internal representations of the Snow attack appear more similar to those of Gabor than those of the other attacks, and that is mirrored in the robust accuracies that we measure. However, our CKA analysis would suggest that the ℓ_∞ is much more similar to ℓ_2 than Gabor, and therefore we would hypothesize that an ℓ_∞-robust model should be

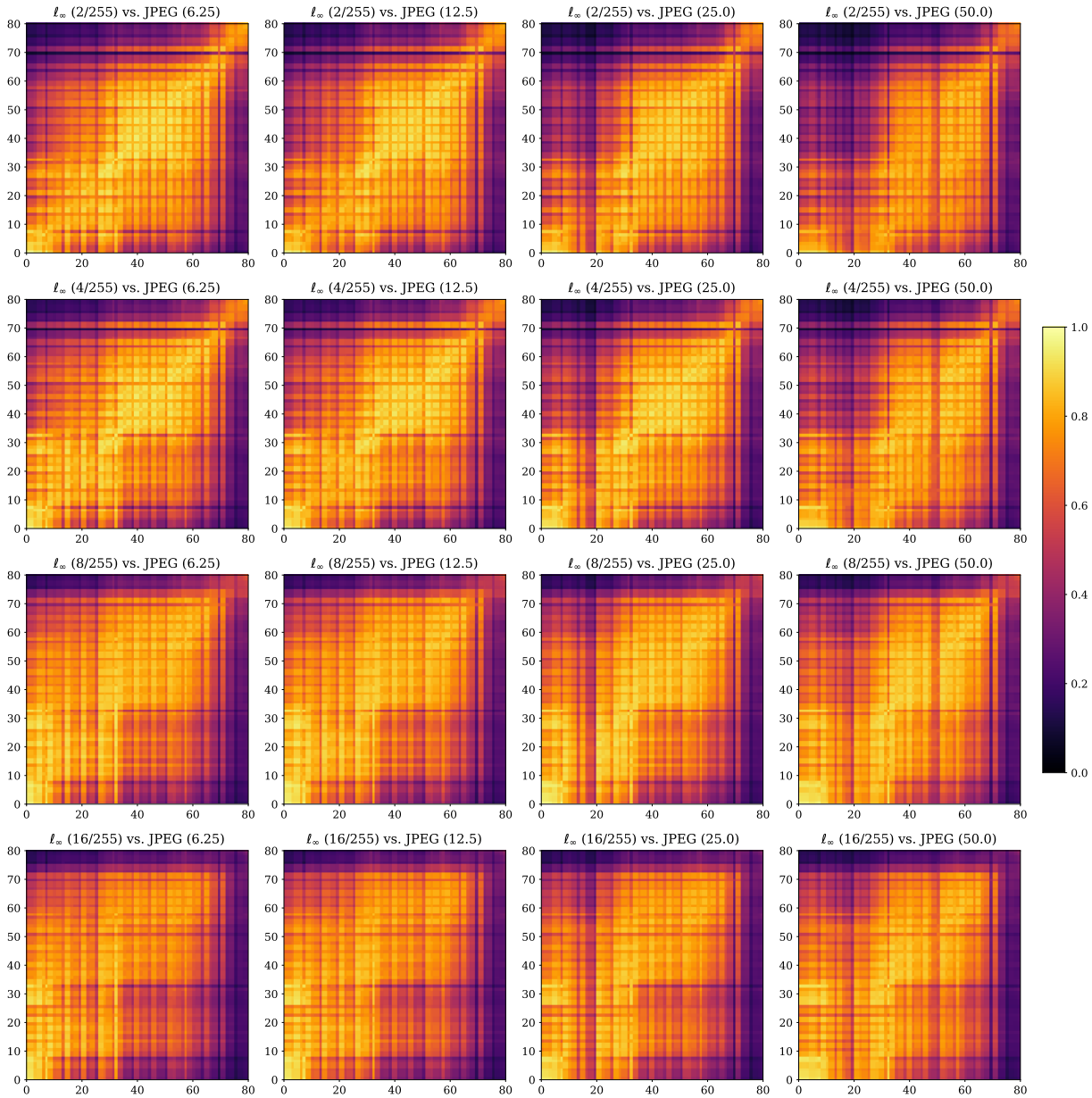


Figure 2.9: **Similarities Between ℓ_∞ and JPEG Threat Models.**Training similarity between l_∞ and JPEG robust models trained against different attack strengths.

inherently more robust to ℓ_2 attacks than Gabor attacks. When we measure robust accuracy directly, the opposite turns out to be true. Based off of these observations, we hypothesize that there must be additional structure in representations that is not captured by our RS analysis which contribute to the robustness of deep neural networks.

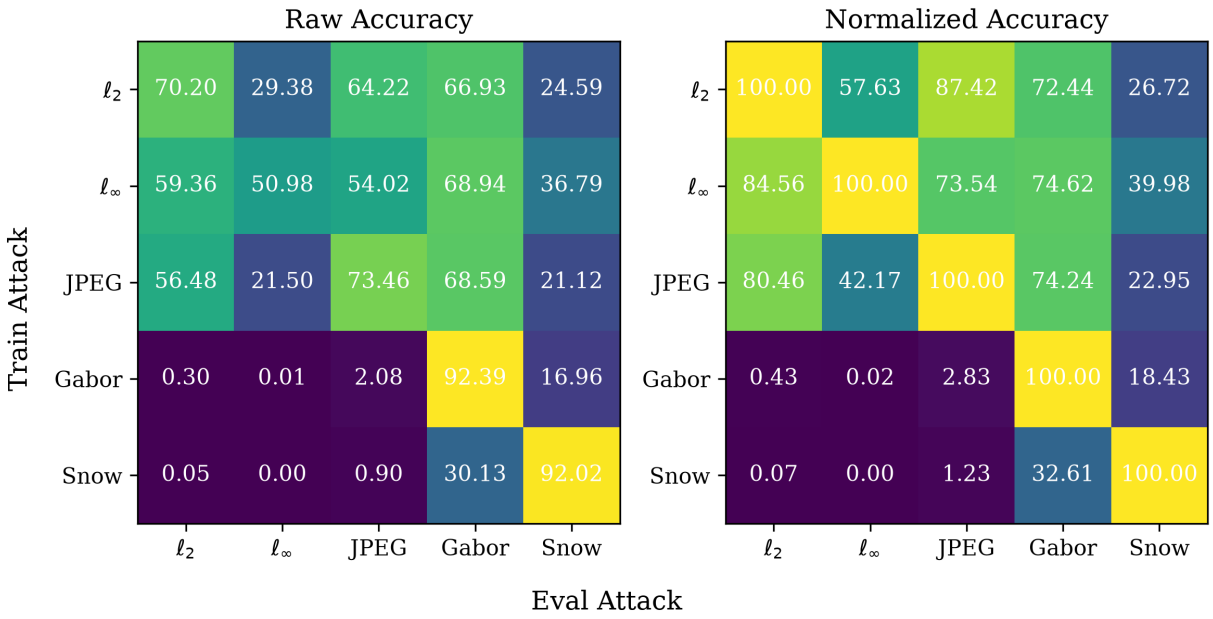


Figure 2.10: **Robustness of Models to Unseen Attacks.** The rows of each heatmap represent the attacks that were used to adversarially train a robust model, and the columns represent the attacks that were used to evaluate each model. Accuracies against the evaluation attacks are displayed in each cell of the left heatmap, while the right heatmap is normalized columnwise such the highest observed robustness to each attack is given a score of 100.

2.7 A Theoretical Approach

The data-driven approach explored above, while powerful in its own right, doesn't have strong explanatory power as to how multi-attack robustness arises in models. Two attacks that yield similar internal representations in a given model, as measured using CKA, may have significantly different attack success rates. We now turn to an approach grounded in learning theory to characterize the relationship between multi-attack robustness and representations.

Many prior works have used theoretical techniques to explain and certify the robustness of different types of models. Of particular relevance to our work is Nern et al. [2023], which proves that the gap between a model's benign and robust loss with respect to any arbitrary attack is bounded by the ℓ_2 distance between the model's benign and robust representations. We adopt techniques from this work and extend them to the multi-attack

setting to demonstrate how a model's robustness to a pair of attacks is related to their distance in the representation space.

Let $h : \mathbb{R}^d \to \mathbb{R}^k$ be a k class neural network classification model. We focus on the a model's robust loss with respect to attacks defined by adversarial neighborhoods C_1 and C_2. Consider the following two adversarial loss functions:

$$\mathcal{L}_1(h) := \mathop{\mathbb{E}}_{(x,y)\in\mathcal{D}} \left[\max_{x'\in C_1(x)} \ell(h(x'), y) \right]$$

and

$$\mathcal{L}_2(h) := \mathop{\mathbb{E}}_{(x,y)\in\mathcal{D}} \left[\max_{x'\in C_2(x)} \ell(h(x'), y) \right].$$

Without loss of generality, assume that $\mathcal{L}_1(h) \geq \mathcal{L}_2(h)$. We additionally define *Union loss*, the expected loss when we target our network with the union of both attacks:

$$\mathcal{L}_{1,2}(h) := \mathop{\mathbb{E}}_{(x,y)\in\mathcal{D}} \left[\max_{i\in\{1,2\}} \left[\max_{x'\in C_i(x)} \ell(h(x'), y) \right] \right].$$

We can then bound the difference between $\mathcal{L}_1(h)$ and $\mathcal{L}_2(h)$, adapting a result from Nern et al. [2023], as follows[3]:

Theorem 1. *Assume that loss function $\ell(\hat{y}, y)$ is M_1-Lipschitz in $\|\cdot\|_2$ with $M_1 > 0$ and bounded by $M_2 > 0$,* [4] *i.e. $0 \leq \ell(\hat{y}, y) \leq M_2 \ \forall \hat{y} \in h(X)$. Then, for a set of n samples*

3. As stated, these results hold for loss functions that are Lipschitz with respect to the ℓ_2 norm. We note that similar bounds can be derived for other norms by applying a constant scaling factor to the first term of the bound (i.e. for losses Lipschitz with respect to the ℓ_1 norm, the scaling factor would be $\sqrt{c}$).

4. We note that surrogate losses such as the cross-entropy used during training are not bounded, but the $0-1$ loss which is often the key quantity of interest *is bounded*.

$x_1, \ldots, x_n$ independently drawn from $\mathcal{D}$, the following holds with probability at least $1 - \rho$:

$$\mathcal{L}_1(h) - \mathcal{L}_2(h) \leq M_1 \frac{1}{n} \sum_{i=1}^{n} \Bigg(\max_{x' \in C_1(x_i)} \|h(x') - h(x_i)\|_2 + \max_{x' \in C_2(x_i)} \|h(x') - h(x_i)\|_2 \Bigg) + J,$$

where $J = M_2 \sqrt{\frac{\log(\rho/2)}{-2n}}$.

Proof. We begin by defining independent random variables $D_1, \ldots, D_n$ which represent the *pointwise robust loss gap*. This gap represents difference in robust loss exhibited by a model on a single point with respect to the two attacks:

$$D_i = \max_{x_i' \in C_1(x_i)} \ell(h(x_i'), y_i) - \max_{x_i'' \in C_2(x_i)} \ell(h(x_i''), y_i)$$

We can use Hoeffding's inequality to show that the average pointwise robust loss gap we observe will be close to the expected pointwise robust loss gap with high probability:

$$\begin{aligned}
&\mathbb{P}\left(\left| \sum_{i=1}^{n} D_i - n\mathbb{E}[D] \right| \geq t \right) \leq 2 \cdot \exp\left(\frac{-2t^2}{nM_2^2} \right) \\
\implies &\mathbb{P}\left(\left| \sum_{i=1}^{n} D_i - n\mathbb{E}[D] \right| \leq M_2 \sqrt{\frac{n \log(\rho/2)}{-2}} \right) \geq 1 - \rho \\
\implies &\mathbb{P}\left(\left| \frac{1}{n} \sum_{i=1}^{n} D_i - \mathbb{E}[D] \right| \leq M_2 \sqrt{\frac{\log(\rho/2)}{-2n}} \right) \geq 1 - \rho.
\end{aligned}$$

Making use of linearity of expectation, the following holds with probability at least $1 - \rho$:

$$\begin{aligned}
\mathbb{E}[D] &= |\mathcal{L}_1(h) - \mathcal{L}_2(h)| \\
&= \left| \mathbb{E}_{(x,y)} \left[\max_{x' \in C_1(x)} \ell(h(x'), y) - \max_{x'' \in C_2(x)} \ell(h(x''), y) \right] \right| \\
&\leq \left| \frac{1}{n} \sum_{i=1}^{n} \max_{x' \in C_1(x)} \ell(h(x'), y_i) - \max_{x'' \in C_2(x)} \ell(h(x''), y_i) \right| + M_2 \sqrt{\frac{\log(\rho/2)}{-2n}}. \qquad (2.5)
\end{aligned}$$

We can further this final term since the loss function $\ell(r, y)$ is M_1-Lipschitz in $\|\cdot\|_2$. This allows us to relate the gap in robust loss with respect to the two attacks to distances in the logit space between perturbed samples.

$$\begin{aligned}
&\left| \frac{1}{n} \sum_{i=1}^{n} \max_{x' \in C_1(x)} \ell(h(x'), y_i) - \max_{x'' \in C_2(x)} \ell(h(x''), y_i) \right| \\
&\leq \left| \frac{1}{n} \sum_{i=1}^{n} |\ell(h(x_i'), y_i) - \ell(h(x_i''), y_i)| \right| \\
&\leq M_1 \frac{1}{n} \sum_{i=1}^{n} \|h(x_i') - h(x_i'')\|_2, \qquad (2.6)
\end{aligned}$$

where $x_1', \ldots, x_n'$ with $x_i' \in C_1(x_i)$ and $x_1'', \ldots, x_n''$ with $x_i'' \in C_2(x_i)$ are chosen to maximize $\ell(h(\cdot), y_i)$ for each i. The perturbed samples represented in this inequality might not maximize the distance between the logits, but that distance can be bounded by the maximally distant perturbations within each neighborhood. Making use of the triangle inequality, we

obtain:

$$\begin{aligned}
&\sum_{i=1}^{n} \|h(x_i') - h(x_i'')\|_2 \\
&= \sum_{i=1}^{n} \|(h(x_i') - h(x_i)) - (h(x_i'') - h(x_i))\|_2 \\
&\leq \sum_{i=1}^{n} \|h(x_i') - h(x_i)\|_2 + \|h(x_i'') - h(x_i)\|_2 \\
&\leq \sum_{i=1}^{n} \max_{x' \in C_1(x_i)} \|h(x') - h(x_i)\|_2 + \max_{x'' \in C_2(x_i)} \|h(x'') - h(x_i)\|_2. \qquad (2.7)
\end{aligned}$$

We then achieve our final result, recalling the assumption that $\mathcal{L}_1(h) \geq \mathcal{L}_2(h)$:

$$\begin{aligned}
\mathcal{L}_1(h) - \mathcal{L}_2(h) &= |\mathcal{L}_1(h) - \mathcal{L}_2(h)| \\
&\leq M_1 \frac{1}{n} \sum_{i=1}^{n} \Big(\max_{x' \in C_1(x_i)} \|h(x') - h(x_i)\|_2 + \max_{x'' \in C_2(x_i)} \|h(x'') - h(x_i)\|_2 \Big) + J, \qquad (2.8)
\end{aligned}$$

where $J = M_2 \sqrt{\frac{\log(\rho/2)}{-2n}}$. □

Put simply, this result implies that a model will have similar levels of robustness to two attacks if they are close in ℓ_2 distance in the model's the representation space. This points towards regularization techniques as an approach for enhancing multi-robustness during adversarial training. Our bound suggests that incorporating a regularization term which penalizes the distance between robust and benign representations with respect to a single attack will give the model greater resiliency against unforeseen attacks. Using regularization when fine-tuning on a new attack could also prevent degradations in robustness against previously seen attacks. Using similar reasoning, we can also bound the gap between Union and clean loss:

Corollary 1. *Let*

$$\mathcal{L}_{1,2}(h) := \mathop{\mathbb{E}}_{\mathcal{D}}\left[\max\left(\ell(h(P_{C_1}(x,y,h)),y),\ell(h(P_{C_2}(x,y,h)),y)\right)\right].$$

Then, with probability at least $1-\rho$*,*

$$\begin{aligned}\mathcal{L}_{1,2}(h) - \mathcal{L}(h) \leq M_1 \frac{1}{n}\sum_{i=1}^{n}\Bigg(&\max_{x'\in C_1(x_i)}\|h(x')-h(x_i)\|_2 \\ &+ \max_{x'\in C_2(x_i)}\|h(x')-h(x_i)\|_2\Bigg) + J.\end{aligned}$$

Proof. Similar to above, define $D_1,\ldots,D_n$ as

$$D_i = \max_{x_i'\in C_1(x_i)\cup C_2(x_i)} \ell(h(x_i'),y_i) - \ell(h(x_i),y_i).$$

We then derive the same bound using Hoeffding's inequality:

$$\begin{aligned}&\mathbb{P}\left(\left|\sum_{i=1}^{n} D_i - n\mathbb{E}[D]\right| \geq t\right) \leq 2\cdot\exp\left(\frac{-2t^2}{nM_2^2}\right) \\ \implies &\mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n} D_i - \mathbb{E}[D]\right| \leq M_2\sqrt{\frac{\log(\rho/2)}{-2n}}\right) \geq 1-\rho.\end{aligned}$$

Then, with probability at least $1-\rho$,

$$\begin{aligned}\mathbb{E}[D] &= \left|\mathcal{L}_{1,2}(h) - \mathcal{L}(h)\right| \\ &= \left|\mathbb{E}_{(x,y)}\left[\max_{x'\in C_1(x)\cup C_2(x)} \ell(h(x'),y) - \ell(h(x),y)\right]\right| \\ &\leq \left|\frac{1}{n}\sum_{i=1}^{n}\max_{x'\in C_1(x_i)\cup C_2(x_i)} \ell(h(x'),y_i) - \ell(h(x_i),y_i)\right| + M_2\sqrt{\frac{\log(\rho/2)}{-2n}}. \qquad (2.9)\end{aligned}$$

Using the fact that the loss function $\ell(r,y)$ is M_1-Lipschitz in $\|\cdot\|_2$ to bound the above term

by ℓ_2 distance in the logit space:

$$\begin{aligned}
&\left|\frac{1}{n}\sum_{i=1}^{n}\max_{x'\in C_1(x_i)\cup C_2(x_i)}\ell(h(x'),y_i)-\ell(h(x_i),y_i)\right| \\
&=\left|\frac{1}{n}\sum_{i=1}^{n}|\ell(h(x_i'),y_i)-\ell(h(x_i),y_i)|\right| \\
&\leq M_1\frac{1}{n}\sum_{i=1}^{n}\|h(x_i')-h(x_i)\|_2,
\end{aligned} \tag{2.10}$$

where $x_1', \ldots, x_n'$ with $x_i' \in C_1(x_i) \cup C_2(x_i)$ are chosen to maximize $\ell(h(\cdot), y_i)$ for each i. The perturbed samples represented in this inequality might not maximize the distance between the logits, but that distance can be bounded by the maximally distant perturbations within each neighborhood.

$$\begin{aligned}
\sum_{i=1}^{n}\|h(x_i')-h(x_i)\|_2 &\leq \sum_{i=1}^{n}\max_{x'\in C_1(x_i)\cup C_2(x_i)}\|h(x')-h(x_i)\|_2 \\
&=\sum_{i=1}^{n}\max_{C\in\{C_1,C_2\}}\max_{x'\in C(x_i)}\|h(x')-h(x_i)\|_2
\end{aligned} \tag{2.11}$$

We then achieve our final result :

$$\begin{aligned}
\mathcal{L}_{1,2}(h)-\mathcal{L}(h) &= \left|\mathcal{L}_{1,2}(h)-\mathcal{L}(h)\right| \\
&\leq M_1\frac{1}{n}\sum_{i=1}^{n}\Big(\max_{x'\in C_1(x_i)}\|h(x')-h(x_i)\|_2+\max_{x''\in C_2(x_i)}\|h(x'')-h(x_i)\|_2\Big)+J,
\end{aligned} \tag{2.12}$$

where $J = M_2\sqrt{\frac{\log(\rho/2)}{-2n}}$. □

This corollary helps characterize the trade-off between clean and worst-case robust loss in our setting. In fact, the gap between Union and benign loss is bounded by the same value as

the gap between robust loss on attack 1 and attack 2. This also supports the regularization approach described above, implying that the model's performance on benign data can be controlled for.

We note that the results above measure distances between representations at the final layer of the network (i.e. the logits). We can derive similar results that apply to internal representations as well. In this setting, we decompose our network into a deep feature extractor and a final linear layer. The bound is similar to above, except it includes a multiplicative factor proportional to the magnitude of the final layer weights.

Corollary 2. *Let $h : \mathbb{R}^d \to \mathbb{R}^c$ be a c class neural network classification model with a final linear layer (i.e. $h(x) = Wg(x)$, where $g : \mathbb{R}^d \to \mathbb{R}^r$, and $W \in \mathbb{R}^{c \times r}$). Assume that loss $\ell(\hat{y}, y)$ is M_1-Lipschitz in $\|\cdot\|_\alpha$ for $\alpha \in \{1, 2, \infty\}$, for $\hat{y} \in h(X)$ with $M_1 > 0$ and bounded by $M_2 > 0$, i.e. $0 \leq \ell(\hat{y}, y) \leq M_2 \ \forall \hat{y} \in h(X)$. Then, for a set of n samples $x_i \dots x_n$ independently drawn from $\mathcal{D}$, the following holds with probability at least $1 - \rho$:*

$$\mathcal{L}_1(h) - \mathcal{L}_2(h) \leq L_\alpha(W) M_1 \frac{1}{n} \sum_{i=1}^{n} \left(\max_{x' \in C_1(x_i)} \|g(x') - g(x_i)\|_2 + \max_{x' \in C_2(x_i)} \|g(x') - g(x_i)\|_2 \right) + J,$$

where $J = M_2 \sqrt{\frac{\log(\rho/2)}{-2n}}$ *and*

$$L_\alpha(W) := \begin{cases} \|W\|_2 & , \text{if } \|\cdot\|_\alpha = \|\cdot\|_2, \\ \sum_i \|W_i\|_2 & , \text{if } \|\cdot\|_\alpha = \|\cdot\|_1, \\ \max_i \|W_i\|_2 & , \text{if } \|\cdot\|_\alpha = \|\cdot\|_\infty. \end{cases}$$

Proof. From (2.5), (2.6), and the definition of h, we have that

$$|\mathcal{L}_1 - \mathcal{L}_2| \leq M_1 \frac{1}{n} \sum_{i=1}^{n} \|Wg(x_i') - h(x_i'')\|_2 + M_2 \sqrt{\frac{\log \rho/2}{-2n}}.$$

We then apply Lemma 2 from Nern et al. [2023] and the definition of L_α:

$$\begin{aligned} M_1 \frac{1}{n} \sum_{i=1}^{n} \|Wg(x_i') - Wg(x_i'')\|_\alpha \\ \leq L_\alpha(W) M_1 \frac{1}{n} \sum_{i=1}^{n} \|g(x_i') - g(x_i'')\|_2. \end{aligned}$$

As in the proof for Theorem 1, the perturbed samples represented in this inequality might not maximize the distance between the representations, but that distance can be bounded by the maximally distant perturbations within each neighborhood. Making use of the triangle inequality, we obtain:

$$\begin{aligned} &\sum_{i=1}^{n} \|g(x_i') - g(x_i'')\|_2 \\ &= \sum_{i=1}^{n} \|(g(x_i') - g(x_i)) - (g(x_i'') - g(x_i))\|_2 \\ &\leq \sum_{i=1}^{n} \|g(x_i') - g(x_i)\|_2 + \|g(x_i'') - g(x_i)\|_2 \\ &\leq \sum_{i=1}^{n} \max_{x' \in C_1(x_i)} \|g(x') - g(x_i)\|_2 + \max_{x'' \in C_2(x_i)} \|g(x'') - g(x_i)\|_2. \end{aligned}$$

We then achieve our final result:

$$\begin{aligned}
&\mathcal{L}_1(h) - \mathcal{L}_2(h) \\
&\qquad = |\mathcal{L}_1(h) - \mathcal{L}_2(h)| \\
&\qquad \leq L_\alpha(W) M_1 \frac{1}{n} \sum_{i=1}^{n} \Big(\max_{x' \in C_1(x_i)} \|g(x') - g(x_i)\|_2 + \max_{x'' \in C_2(x_i)} \|g(x'') - g(x_i)\|_2 \Big) + J,
\end{aligned}$$

where $J = M_2 \sqrt{\frac{\log(\rho/2)}{-2n}}$. □

When we situate our study of representations within a theoretical framework, we can paint a much clearer picture of the relationship between representations and robustness. In our case, the relationship is simple: representations of attacks close in ℓ_2 distance imply that multi-robustness is achievable. In the remainder of this chapter, we will use insights from these theoretical results to guide the design of regularization-based approach for training multi-robust models.

2.8 Case Study: Continual Adaptive Robustness

To motivate the design of our multi-robust training approach, we will begin by introducing the adversarial context in which we anticipate our models being deployed. We adopt the problem setting of continual adaptive robustness (CAR), first defined by Dai et al. [2024], which aims to achieve robustness against new attacks as they are sequentially discovered. We survey existing approaches to this problem and propose a novel defense technique. CAR is visualized in Figure 2.11.

At a high level, CAR describes a setting in which an entity trying to train and deploy an adversarially robust model does not have full knowledge of the types of adversaries that will try to attack the model. For instance, the model deployer may only be aware of ℓ_p bounded attacks at the time of training and therefore attempt to train a model that is robust to

those attacks. After deployment, they may encounter Gabor-bounded adversaries trying to attack their model. The question becomes how the model deployer can best defend against adversaries when the set of known attacks is constantly updating. This setup creates two main goals for the model deployer:

1. Be able to train a model that has some amount of inherent robustness against attacks that were unknown during training.

2. Be able to efficiently update the weights of the model when a new attack is encountered so as to enhance robustness against the new attack without degrading robustness to previously known attacks.

We propose and analyze the first dedicated defense for CAR in this work.

2.8.1 Problem Formulation

Attack sequences: In CAR [Dai et al., 2024], different test-time attacks are introduced sequentially (Figure 2.11). Each attack (perturbed input) can be formulated as the maximizer of the loss $P_C(x, y, h) = \arg\max_{x' \in C(x)} \ell(h(x), y)$ (within the constraint C) and has a corresponding time $T(P_C)$ at which it is discovered by the defender. We call the set of attacks known by the defender at a given time t the *knowledge set* at time t: $K(t) = \{P \,|T(P) \leq t\}$. The expansion of K over time models settings such as research groups or a security team discovering new attack types.

Goals in CAR: A defender in CAR uses a defense algorithm $\mathcal{A}_{\text{CAR}}$ to deploy a model $h_t = \mathcal{A}_{\text{CAR}}(\mathcal{D}, K(t), \mathcal{H})$ at each time step t. Performance at time t is measured by Union robust loss across the knowledge set: $\mathcal{L}(h, t) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \max_{P \in K(t)} [\ell(P(x, y, h), y)]$.

Definition 2 (Continual Adaptive Robustness [Dai et al., 2024])**.** *Given loss tolerances δ_{known} and $\delta_{unknown}$ with $0 < \delta_{known} < \delta_{unknown}$ and grace period Δt for recovering from a new attack, a defense algorithm $\mathcal{A}_{CAR}$ achieves CAR if for all $t > 0$:*

- *When $t - T(P) < \Delta t$ for any attack P and $T(P) < t$, h_t satisfies $\mathcal{L}(h_t, t) \leq \delta_{unknown}$*
- *Otherwise, $\mathcal{L}(h_t, t) \leq \delta_{known}$.*

These criteria capture 3 distinct goals for the defender: (1) The model at time t must *achieve good robustness if no attacks have been introduced recently* (within Δt time). This is due to the δ_{known} threshold on the robust loss in the second criterion. (2) If a new attack has occurred within Δt period before the current time t, the model at time t must *achieve some robustness against the new attack*. This is modeled by the δ_{unknown} threshold in the first criterion. Since $0 < \delta_{\text{known}} < \delta_{\text{unknown}}$, CAR tolerates a degradation in robustness between the 2 cases; (3) The defense is expected to *recover robustness quickly after new attacks*. This is modeled by the Δt time window; Δt time after the introduction of a new attack, the loss threshold changes from δ_{unknown} to δ_{known}.

2.8.2 Baseline Approaches to CAR

CAR through multiattack robustness (MAR). Multiattack robustness aims to defend against a set of attacks by that are known to the defender at training time. Existing approaches for achieving multiattack robustness often involve training with multiple attacks simultaneously [Tramèr and Boneh, 2019a, Maini et al., 2020]. However, this type of approach can be computationally expensive. A trivial (but expensive) defense algorithm for CAR is to use these training-based techniques on the initial set of known attacks $K(0)$ and retrain a model from scratch on $K(t)$ every time new attacks are revealed. A significant drawback of this approach is that the grace period Δt would increase as the size of the knowledge set increases.

CAR through unforeseen attack robustness (UAR). Unforeseen attack robustness (UAR) aims to defend against attacks aims to defend against attacks that were not necessarily known at training time. Examples of this approach include Laidlaw et al. [2021] which proposes training based on LPIPS [Zhang et al., 2018], a metric more aligned with human

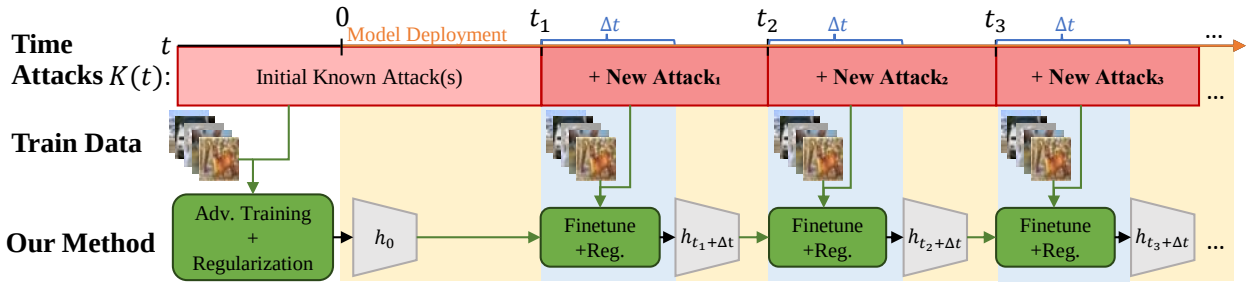


Figure 2.11: **An Overview of the Problem of Adapting to New Adversaries (Continual Adaptive Robustness) and Our Solution Framework (Regularized Continual Robust Training).** The defender learns about the existence of new attacks sequentially, and at time t aims to achieve robustness against $K(t)$, the set of attacks known at times $\leq t$. A model h_0 is deployed at time 0 to be robust against an initial set of known attacks, and new attacks are introduced at times t_1, t_2, and t_3. We propose performing regularized initial robust training on the initially known attack(s) and then using regularized fine-tuning to adapt the model against future attacks within time Δt, leading to a sequence of models $h_0, h_{t_1+\Delta t}, h_{t_3+\Delta t}, h_{t_3+\Delta t}$.

perception than ℓ_p distances, as well as Dai et al. [2022] and Jin and Rinard [2020] which use regularization during training in order to obtain better generalization to unforeseen attacks. An effective UAR defense could trivially serve as an effective CAR defense by training an initial model which is robust to unforeseen attacks. This approach is efficient as the model would not need to be retrained as the knowledge set grows. However, it would require much higher values of δ_{known} as in practice these methods do not obtain state of the art robustness across attacks, especially those not seen during training [Dai et al., 2023].

2.8.3 Our Approach: Continual Robust Training

We now introduce continual robust training (CRT). CRT consists of 2 parts, *initial training* and *iterative fine-tuning* (Figure 2.11). The output of initial training is deployed at $t = 0$ while fine-tuning is used as new attacks are introduced.

At time $t = 0$, the goal of the defender is to minimize the initial training objective: $\mathcal{L}(h, 0) = \frac{1}{m}\sum_{i=1}^{m} \ell(h(P_{C_{\text{init}}}(x_i, y_i, h)), y_i)$ where $\{(x_i, y_i)\}_{i=1}^{m}$ is the training dataset and $P_{C_{\text{init}}}$ is the initial attack. Notably, using standard training in this stage yields a high

δ_{unknown}.

At $t > 0$, as new attacks are introduced, we use a fine-tuning strategy F to select the attack from $K(t)$ to use for each example. Specifically, we formulate this as: $\mathcal{L}(h, t) = \frac{1}{m}\sum_{i=1}^{m} \ell(h(P_C(x_i, y_i, h)), y_i)$ where $P_C = F(K(t), (x_i, y_i))$. Fine-tuning strategies include picking the attack that maximizes $\ell(x_i, y_i)$, randomly sampling from $K(t)$, and using the newest attack. A good fine-tuning strategy would be able to quickly adapt the model to new attacks, allowing it to satisfy a small Δt threshold. However, naive fine-tuning does not guarantee good performance across all attacks and may require large values of δ_{known}. As illustrated in 2.12, a model may lose robustness to the initial attack after the fine-tuning stage. We now discuss how such degradation can be addressed through regularization.

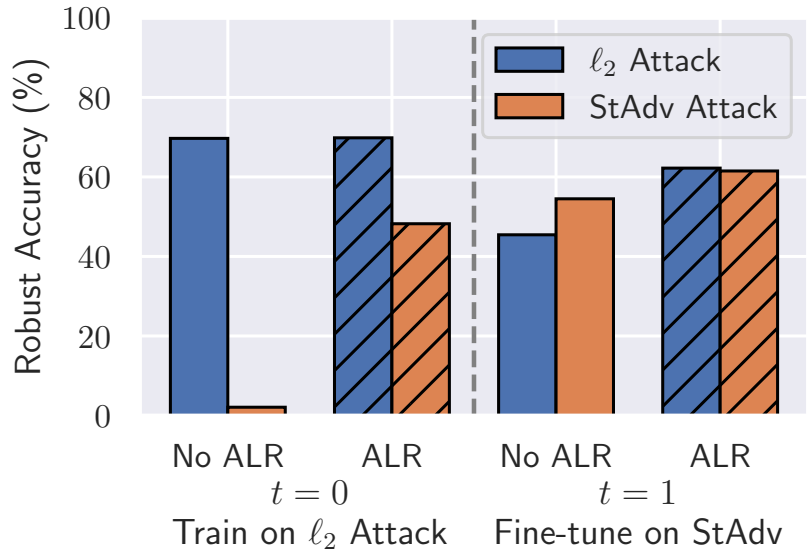


Figure 2.12: **Impact of Our Proposed Regularization Term (ALR) in Both Training and Fine-Tuning on Cifar-10.** Adversarial ℓ_2 regularization (ALR) significantly improves generalization to the unforeseen StAdv attack when performing adversarial training for ℓ_2 robustness. Using ALR when subsequently fine-tuning with only StAdv attack also decreases the drop in ℓ_2 robustness.

2.8.4 Regularization Methods

Theorem 1 suggests that reducing the sensitivity of logits to *either* attack has the potential to reduce the performance gap between attacks (see 2.13 in the Appendix for an empirical validation of this effect). To this end, we propose incorporating regularization into both training stages. Specifically, we adopt modified training objective $\mathcal{L}_{\text{reg}}(h,t) = \mathcal{L}(h,t) + \lambda R(h, K(t))$, where λ is the regularization strength and $R(h)$ is the regularization term used. We will now discuss several forms of regularization.

Adversarial ℓ_2 regularization. (ALR) Driven by our theoretical results, we first introduce adversarial ℓ_2 regularization:

$$R_{\text{ALR}}(h, K(t)) = \frac{1}{m}\sum_{i=1}^{m} \max_{x' \in C(x_i)} \|h(x') - h(x_i)\|_2,$$

where $C = C_{\text{init}}$ in initial training and corresponds to attack $P_C = F(K(t), (x_i, y_i))$ chosen by the fine-tuning strategy. ℓ_2 regularization penalizes the maximum distance between a sample's logits and the furthest adversarially perturbed logits within that sample's neighborhood. Using this regularization term would directly minimize the upper bounds in Theorem 1 and Corollary 1. We note that while ALR is structurally similar TRADES [Zhang et al., 2019], it uses a Euclidean distance instead of the KL-divergence. This work is the first to show that this form of regularization is beneficial for multirobustness.

Efficiently approximating ALR. Computing ALR uses multi-step optimization which can be costly to compute in practice. To improve efficiency in experiments, we consider (1) using single step optimization for ALR and (2) using randomly sampled, unoptimized perturbations can help with CAR. For (2), we consider Gaussian noise regularization (GR) and Uniform noise regularization (UR), specifically:

$$R_{\text{GR}}(h, K(t)) = \frac{1}{m}\sum_{i=1}^{m} \|h(x') - h(x_i)\|_2$$

where $x' \sim \mathcal{N}(0, \sigma^2)$ and

$$R_{\text{UR}}(h, K(t)) = \frac{1}{m} \sum_{i=1}^{m} \|h(x') - h(x_i)\|_2$$

where $x' \sim \mathcal{U}(-\sigma, \sigma)$.

Other Baselines. We compare to variation regularization (VR), which has been shown to improve generalization to unforeseen attacks [Dai et al., 2022]. VR is defined as:

$$R_{\text{VR}}(h, K(t)) = \frac{1}{m} \sum_{i=1}^{m} \max_{x', x'' \in C(x_i)} \|h(x') - h(x'')\|_2$$

where $C = C_{\text{init}}$ in initial training. We also consider VR in finetuning with C corresponding to attack $P_C = F(K(t), (x_i, y_i))$.

We note that ALR and VR are inherently related terms. Since VR optimizes over 2 perturbations x' and x'' for each example while ALR optimizes only for x', it is clear that $R_{\text{ALR}} \leq R_{\text{VR}}$. Additionally, we note that:

$$R_{\text{VR}}(h, K(t)) = \frac{1}{m} \sum_{i=1}^{m} \max_{x', x'' \in C(x_i)} \|h(x') - h(x) + h(x) - h(x'')\|_2$$

$$\leq \frac{1}{m} \sum_{i=1}^{m} \max_{x', x'' \in C(x_i)} \|h(x') - h(x)\|_2 + \|h(x) - h(x'')\|_2$$

$$= \frac{2}{m} \sum_{i=1}^{m} \max_{x' \in C(x_i)} \|h(x') - h(x)\|_2$$

$$= 2R_{\text{ALR}}$$

Thus, ALR and VR are related in the sense that $R_{\text{ALR}} \leq R_{\text{VR}} \leq 2R_{\text{ALR}}$.

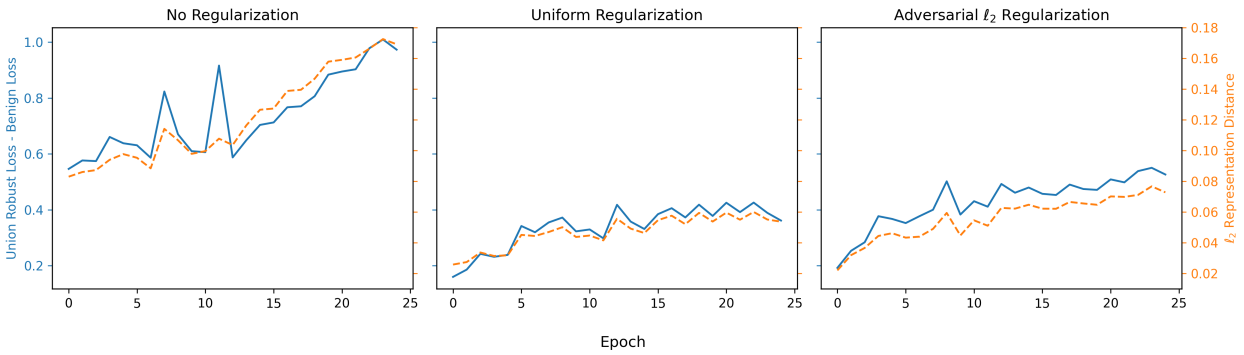


Figure 2.13: **Comparing Representation Distances and Robust Loss Gap.** Adversarial loss gap $\mathcal{L}_{1,2}(h) - \mathcal{L}(h)$ (solid blue lines) and average ℓ_2 distance between logits of ℓ_2 ($\epsilon = 0.5$, representing P_{C_1}) and StAdv ($\epsilon = 0.05$, representing P_{C_2}) attacked samples over 25 epochs of fine-tuning using Croce and Hein [2022]'s fine-tuning method (dashed orange lines), both with and without regularization. Each model is fine-tuned starting from a model that is adversarially trained against an ℓ_2 adversary, as described in Section 2.9.1. In all training scenarios, there is a visible correlation between the loss gap and the logit distance, aligning with the theoretical result in Corollary 1.

2.8.5 Experimental Validation of Theoretical Results

We now briefly demonstrate that our chosen regularization terms align with our theoretical results. In Figure 2.13, we start with WRN-28-10 models that were adversarially trained to be robust against ℓ_2-bounded attacks, and fine-tune them to increase their robustness against StAdv attacks using either no regularization, uniform regularization, or adversarial ℓ_2 regularization. We observe a number of trends:

Sensitivity correlates with loss gap. Whether or not regularization is used, there is a clear correlation between total adversarial sensitivity across both attacks (defined as $\max_{x' \in C_1(x)} \|h(x') - h(x)\| + \max_{x' \in C_2(x)} \|h(x') - h(x)\|$) and the loss gap between the union robust loss and the benign loss (i.e. $\mathcal{L}_{1,2}(h) - \mathcal{L}(h)$).

Regularization reduces sensitivity and loss gap. Both metrics are significantly lower throughout fine-tuning when regularization is used, indicating that regularization is successfully targeting our theoretical bounds.

Loss gap increases over time. Across all three models there is an increase in both loss gap and adversarial sensitivity over the course of fine-tuning. While this may seem like a

failure of regularization, the benefit is more apparent when further analyzing what is causing the loss gap to increase. In the regularized fine-tuning runs, both benign and robust losses are decreasing, with benign loss decreasing more quickly. This is likely influenced by an initial increase in benign loss at the very beginning of fine-tuning which is not captured in Figure 2.13. However, without regularization, benign loss decreases while union robust loss increases. This shows us that despite theoretically targeting the gap between union robust loss and benign loss, the use of regularization still aids in individually reducing both losses in absolute terms.

2.9 Experimental Results

2.9.1 Experimental Setup

Datasets and Architectures. We largely use the same datasets and architectures discussed in Section 2.3.1. For datasets, we largely focus on CIFAR-10, with some experiments on CIFAR-100 and ImageNette. For CIFAR-10 and CIFAR-100 we use the WideResnet-28-10 (WRN-28-10) architecture and for ImageNette we use ResNet-18.

Attacks. To study CRT with a diverse range of attackers, we include results for 12 types of adversaries. These include the five attacks described in Section 2.3.1 (ℓ_2, ℓ_∞, JPEG, Snow, and Gabor), StAdv [Xiao et al., 2018], ReColor attacks [Laidlaw and Feizi, 2019], and an additional five attacks from Imagenet-UA (Pixel, Elastic, Wood, Glitch, and Kaleidoscope) [Kaufmann et al., 2019]. For ℓ_2 attacks, we use a bound $\epsilon = 0.5$ for CIFAR datasets and $\epsilon = 1$ for ImageNette. For ℓ_2 attacks, we use $\epsilon = \frac{8}{255}$, and for StAdv and ReColor attacks, we use the same bounds as used in their original papers Xiao et al. [2018] ($\epsilon = 0.05$) and Laidlaw and Feizi [2019] ($\epsilon = 0.06$) respectively. For ImageNet-UA attacks, we use the *medium distortion* strength bounds used by Kaufmann et al. [2019].

Training from scratch baselines. We consider the following baselines for training from

scratch:

- *Training with AVG and MAX objectives* [Tramèr and Boneh, 2019a]: Tramèr and Boneh [2019a] propose two different training objectives, AVG:

$$L_{\text{AVG}}(h,t) = \frac{1}{m|K(t)|}\sum_{i=1}^{m}\sum_{P_C\in K(t)}\ell(h(P_C(x_i,y_i)),y_i),$$

  and MAX:

$$L_{\text{MAX}}(h,t) = \frac{1}{m}\sum_{i=1}^{m}\max_{P_C\in K(t)}\ell(h(P_C(x_i,y_i)),y_i)$$

  for robustness against multiple known attacks.

- *Randomly sampling attacks* [Madaan et al., 2020]: AVG and MAX require generating adversarial examples with all attacks for each image. For a more efficient baseline, we consider randomly sampling an attack for each batch for use in adversarial training.

CRT Baselines. For CRT, we use PGD adversarial training (AT) [Madry et al., 2018] for initial training and then fine-tune the model using several different fine-tuning strategies:

- *MAX objective fine-tuning* (FT-MAX) [Tramèr and Boneh, 2019a]: We use the MAX objective for fine-tuning when a new attack is introduced.

- *Croce and Hein [2022] fine-tuning* (FT Croce): Croce and Hein [2022] introduce a fine-tuning technique for use with ℓ_∞ and ℓ_1 attacks which we *generalize to training with arbitrary attacks*. This approach samples a single attack per batch. The probability that an attack P_C is sampled is given by $\frac{\text{err}(P_C)}{\sum_{P\in K(t)}\text{err}(P)}$ where $\text{err}(P)$ denotes the running average of robust loss with respect to attack P computed across batches of each attack.

- *Single attack fine-tuning* (FT Single): We also consider fine-tuning with *only the newly introduced attack*, allowing us to determine the extent to which previous attacks are

forgotten. The previous two fine-tuning techniques involve replaying previous attacks.

We then investigate incorporating regularization into the initial training and fine-tuning phases of CRT.

Training and Fine-tuning Procedures. During training, we use 10-step Projected Gradient Descent [Madry et al., 2018] to generate adversarial examples. For the regularization terms (2.8.4), VR and ALR use single step optimization to reduce time overhead, while UR and GR use $\sigma = 2$ and $\sigma = 0.2$, respectively. We train models for 100 epochs for initial training and 10 epochs for fine-tuning (results with 25 epochs in A.7). We include additional details about the training procedure in A.6.

Evaluation Attacks and Metrics. Our main results in Table 2.3 and additional ones in Appendix A.7 use full AutoAttack [Croce and Hein, 2020b] for evaluating ℓ_p robustness. For ablations, we restrict to APGD-T and FAB-T from AutoAttack to reduce evaluation time. We use 20-step optimization when evaluating StAdv and ReColor attacks and the default evaluation hyperparameters for ImageNet-UA attacks in Kaufmann et al. [2019]. We report *accuracy on each attack*, *Union accuracy* (overall accuracy when the worst case attack is chosen for each test example), *Average accuracy* (average over accuracy on each attack), and *training time* (in hours). Metrics are reported for the epoch E^* with best performance on the set of known attacks. For training from scratch, the reported training time is scaled by fraction of training for the best epoch (*i.e.* we report $\frac{E^*}{100} \times$ training time for 100 epochs). For fine-tuning we report training time for the full 10 epochs. This allows us to see how much faster fine-tuning is to optimal early stopping when re-training from scratch.

2.9.2 Improving CRT with Regularization

We now analyze the robustness of models trained using CRT with and without regularization. To model a CAR setting, we consider a sequence of 4 attacks: $\ell_2 \to$ StAdv $\to \ell_\infty \to$ Recolor. The first attack is the initially known attack while other attacks are introduced at later time

Time Step		Procedure	Threat Models	Clean	ℓ_2	StAdv	ℓ_∞	Recolor	Avg (known)	Union (known)	Avg (all)	Union (all)	Time (hrs)
0	Init	AT	ℓ_2	**91.17**	69.7	2.08	28.41	44.94	69.7	69.7	36.28	1.24	8.68
		AT + ALR ($\lambda = 1$)	ℓ_2	89.43	**69.84**	**48.23**	**34.00**	**65.46**	**69.84**	**69.84**	**54.38**	**31.27**	17.17
1	Finetune	FT MAX	ℓ_2, StAdv	83.73	57.07	58.67	12.51	49.03	57.87	51.32	44.32	12.36	4.00
		FT Single	ℓ_2, StAdv	80.89	45.45	54.5	6.09	41.98	49.98	41.05	37.0	5.87	2.78
		FT Croce	ℓ_2, StAdv	84.7	57.88	54.27	14.38	51.08	56.07	48.13	44.4	13.8	2.40
		FT Single + ALR	ℓ_2, StAdv	**87.24**	**62.22**	61.5	**21.4**	**70.87**	61.86	55.04	**54.0**	**21.14**	4.24
		FT Croce + ALR	ℓ_2, StAdv	86.03	59.18	**65.14**	15.36	63.31	**62.16**	**55.83**	50.75	15.29	3.47
2	Finetune	FT MAX	ℓ_2, StAdv, ℓ_∞	83.16	65.63	56.68	36.9	65.69	53.07	35.18	56.23	34.83	5.62
		FT Single	ℓ_2, StAdv, ℓ_∞	87.99	**70.53**	11.17	41.63	63.46	41.11	7.95	46.7	7.74	1.57
		FT Croce	ℓ_2, StAdv, ℓ_∞	85.05	67.3	48.07	33.38	62.52	49.58	28.96	52.82	28.63	2.27
		FT Single + ALR	ℓ_2, StAdv, ℓ_∞	**88.74**	69.15	47.33	**42.08**	68.62	52.85	**36.66**	56.8	**36.62**	2.26
		FT Croce + ALR	ℓ_2, StAdv, ℓ_∞	86.57	67.99	**61.55**	36.59	**72.16**	**55.38**	35.68	**59.57**	35.52	2.87
3	Finetune	FT MAX	ℓ_2, StAdv, ℓ_∞, Recolor	83.64	66.21	57.53	**37.77**	69.32	57.71	**36.02**	57.71	**36.02**	8.45
		FT Single	ℓ_2, StAdv, ℓ_∞, Recolor	90.41	66.47	3.93	29.6	69.03	42.26	2.49	42.26	2.49	3.11
		FT Croce	ℓ_2, StAdv, ℓ_∞, Recolor	86.64	**68.76**	44.81	36.02	68.05	54.41	29.44	54.41	29.44	2.34
		FT Single + ALR	ℓ_2, StAdv, ℓ_∞, Recolor	**90.45**	61.58	25.77	27.43	69.26	46.01	19.2	46.01	19.2	4.24
		FT Croce + ALR	ℓ_2, StAdv, ℓ_∞, Recolor	87.62	68.14	**58.5**	36.39	**72.35**	**58.85**	34.92	**58.85**	34.92	3.35

Table 2.3: **Continual Robust Training on CIFAR-10.** Best performance for each time step are **bolded**. The defender initially knows about ℓ_2 attacks and over time, is sequentially introduced to StAdv, ℓ_∞, and ReColor attacks. We report clean accuracy, accuracy on individual attacks, and average and union accuracies. The "Threat Models" column specifies known attacks at the current time step, and accuracies on these attacks are in green cells. Initial adversarial training occurs at time step 0, and the model is updated through fine-tuning the model from the previous time step. "Avg (known)" and "Union (known)" columns represent average and union accuracies on known attacks while "Avg (all)" and "Union (all)" columns report performance across all four attacks. We report training time for each time step in the "Time" column.

steps. We present results for CIFAR-10 in Table 2.3. We include results in Appendix A.7 for Imagenette and CIFAR-100 as well as additional results for longer duration of fine-tuning (25 epochs) and a separate sequence of attacks: $\ell_\infty \to$ StAdv $\to$ Recolor $\to \ell_2$. For these experiments, we use $\lambda = 0.5$ unless specified otherwise.

Regularization reduces degradation on previous attacks. From Table 2.3, we observe that fine-tuning with only the new attack (FT Single) can lead to degradation of robustness against previous attacks. The incorporation of ALR significantly decreases this drop in robustness. For example, when fine-tuning from an ℓ_2 robust model with StAdv attacks (time step 1 in Table 2.3), FT Single incurs a 24.25% drop (from 69.7% to 45.45%) in ℓ_2 accuracy from the initial checkpoint (AT at time step 0). Meanwhile FT Single + ALR only experiences a 7.62% drop (from 69.84% to 62.22%) in ℓ_2 accuracy from the initial checkpoint (AT + ALR at time step 0). Similarly, after the introduction of ℓ_∞ attack at time

Procedure	Clean	Avg	Union	Time
MAX	84.3	54.18	37.44	61.09
AVG	87.77	54.5	30.39	51.55
Random	86.32	54.27	30.76	13.15
CRT + ALR	87.62	58.85	34.92	26.86

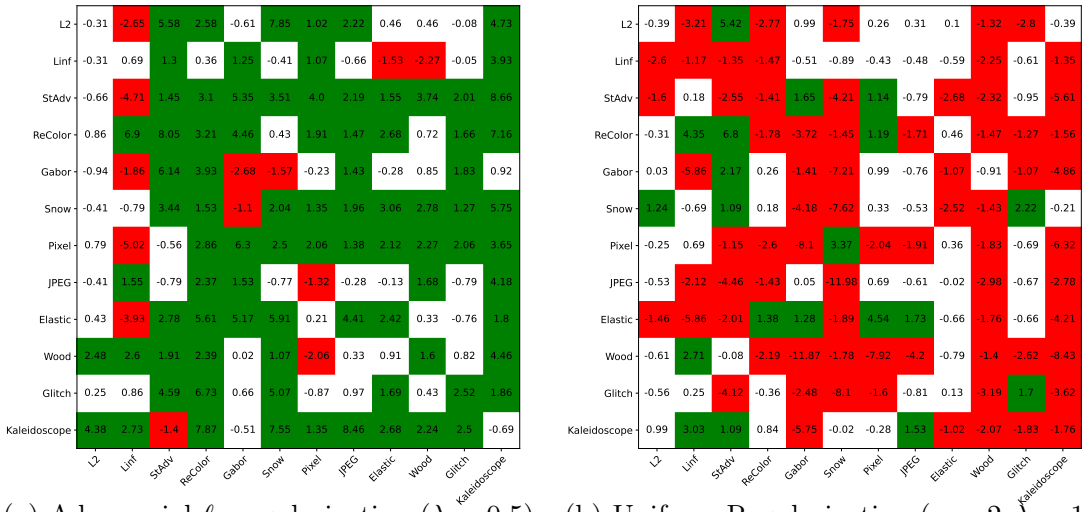
Table 2.4: **Comparing Regularized CRT to Retraining from Scratch.** Regularized CRT (using Croce and Hein [2020b] fine-tuning strategy) compared to training from scratch on ℓ_2, StAdv, ℓ_∞, and Recolor attacks on CIFAR-10.

Initial Attack	Reg Type	λ	Clean	ℓ_2	ℓ_∞	StAdv	ReColor	Gabor	Snow	Pixel	JPEG	Elastic	Wood	Glitch	Kaleid-oscope	Avg	Union
ℓ_2	None	0	91.08	70.02	29.38	0.79	33.69	66.93	24.59	14.99	64.22	45.13	70.85	80.30	30.08	44.25	0.10
ℓ_2	VR	0.2	89.99	70.38	34.56	13.41	48.99	67.64	29.09	22.57	66.64	48.38	73.31	80.07	32.33	48.94	5.40
ℓ_2	ALR	0.5	89.57	70.29	34.16	17.44	51.04	65.63	28.71	22.50	66.76	48.80	73.24	79.66	28.83	48.92	5.94
ℓ_2	UR	5	88.34	66.66	27.41	26.22	60.22	69.16	26.67	22.57	64.08	46.83	71.14	77.60	31.36	49.16	6.23
ℓ_2	GR	0.5	86.89	68.19	32.02	16.54	58.32	74.85	25.69	21.26	65.32	46.82	74.08	76.99	31.93	49.33	4.18
ℓ_∞	None	0	85.53	59.36	50.98	6.34	56.27	68.94	36.79	20.57	54.02	51.00	64.24	75.94	39.44	48.66	1.31
ℓ_∞	VR	0.2	82.58	58.36	51.53	18.98	62.12	67.18	39.22	23.62	54.73	52	63.35	71.72	43.18	50.50	5.08
ℓ_∞	ALR	0.5	83.18	58.21	51.47	19.50	61.02	68.75	37.94	22.78	53.89	49.82	63.47	73.57	39.88	50.02	5.52
ℓ_∞	UR	5	78.04	60.28	40.59	42.25	70.00	67.06	33.40	26.57	60.07	49.21	64.61	67.08	38.43	51.63	8.36
ℓ_∞	GR	0.5	80.65	59.74	46.12	34.57	70.49	68.33	35.80	26.04	57.28	51.98	65.46	70.73	38.21	52.06	6.28

Table 2.5: **Impact of Regularization on Unforeseen Robustness.** We consider the setting where the defender is only aware of a single attack and performs training with and without different types of regularization: variation regularization (VR), adversarial ℓ_2 regularization (ALR), uniform regularization (UR), and Gaussian regularization (GR) at regularization strength λ. We report clean accuracy and robust accuracies on a range of attacks. Green cells represent an improvement of at least 1% while red cells represent a drop of at least 1% in comparison to no regularization.

step 2, the accuracy of FT Single on StAdv attacks drops 43.42% (from 54.5% to 11.17%) while FT Single + ALR only experiences a 14.17% drop (from 61.5% to 47.33%). These results align with Theorem 1: when incorporating ALR into training, the gap in loss on the two attacks is lessened.

Regularization improves performance on held out (unforeseen) attacks. We observe that regularized CRT leads to higher robustness on attacks held out from training. For example, at time step 1 in Table 2.3, which trains with ℓ_2 and StAdv attacks, the best accuracy on Recolor attacks out of unregularized CRT methods is 51.08%, while FT Single + ALR achieves 70.87% accuracy on Recolor attacks and FT Croce + ALR achieves 63.31% accuracy on Recolor attacks. The improvement in robustness on unforeseen attacks aligns with 1 as regularization helps decrease the drop in accuracy between clean inputs and



(a) Adversarial ℓ_2 regularization ($\lambda = 0.5$) (b) Uniform Regularization ($\sigma = 2, \lambda = 1$)

Figure 2.14: **Change in Union Robust Accuracy After Fine-Tuning with Regularization.** We fine-tune models on Imagenette across 144 pairs of initial attack and new attack. The initial attack corresponds to the row of each grid and new attack corresponds to each column. Values represent differences between the accuracy measured on a model *fine-tuned with and without regularization*. Gains in accuracy of at least 1% are highlighted in green, while drops in accuracy of at least 1% in red. Regularization was not used during the initial training phase, only during fine-tuning.

perturbed inputs. This also aligns with CAR's goal of having a small δ_{unknown}.

Regularization balances performance and efficiency. Our proposed regularization term adds a small computational overhead over other FT approaches but generally improves union performance on the set of known attacks. For example, when considering the sequence of ℓ_2 and StAdv attacks (time step 1 in Table 2.3), FT Croce + ALR improves union accuracy over FT Croce by 7.7% while adding a time overhead of 1.07 hours. Additionally, when considering the sequence of 3 attacks (ℓ_2, StAdv, and ℓ_∞ attacks), FT Croce + ALR improves union accuracy over FT Croce by 6.72% while adding a time overhead of 0.6 hours. This increase in time complexity is much smaller than FT MAX which takes 1.6 hours longer than FT Croce for ℓ_2 and StAdv and 3.35 hours longer for ℓ_2, StAdv, and ℓ_∞. With respect to goals in CAR, regularization balances δ_{known} and Δt.

Comparison to training from scratch. In Table 2.4, we report clean, average, and union accuracies along with total training times for using training from scratch on all 4 attacks

compared to training sequentially with regularized CRT on CIFAR-10. We observe that regularized CRT is significantly more efficient than MAX and AVG training (taking a total of 26.86 hours while AVG and MAX take over 50 hours of training time). Surprisingly, we find that on CIFAR-10, regularized CRT can outperform training from scratch methods, achieving 4.35% higher average accuracy compared to the best achieved by training from scratch. This suggests that transferable robustness between carefully chosen attacks can improve MAR as a whole. However, we note that the ability to outperform training from scratch seems to be specific to CIFAR-10; for ImageNette and CIFAR-100 (Appendix A.7) training from scratch outperforms using fine-tuning in CAR.

Impact of dataset and attack sequence. In Appendix A.7, we provide results on ImageNette and CIFAR-100 as well as for attack sequence $\ell_\infty \to$ StAdv $\to$ Recolor $\to \ell_2$. Overall, we observe that trends such as improved robustness to unforeseen and the union of attacks are generally consistent. However, the extent to which regularization improves performance over FT Croce varies. The choice of the initial attack seems to play a role in subsequent robustness, and if defenders are aware of multiple attacks, choosing the right one to start with is an interesting open question.

2.10 Discussion

The performative nature of adversarial training presents difficulties in designing meaningful defenses for neural networks. Our exploration of robust networks using representation similarity explores how traditional defense techniques leave networks vulnerable to new and evolving adversaries. Our theoretical results inspire the design of a new defense technique, continual robust training, which can quickly adapt to these changing adversaries. However, it is by no means a perfect solution. For some combinations of attacks our approach fails to outperform the baseline, and our framework still makes assumptions about the adversary (namely that the perturbations they introduce must be bounded by some norm).

Future work could focus on tightening the bounds provided by our theoretical results in order to provide stronger protections against disparate sets of attacks. Another direction might be to try to anticipate the ways in which adversaries might attempt to evade existing defenses. This idea echos the framework of performative prediction proposed by Perdomo et al. [2020], who develop a method for training a model to predict a target variable which may be influenced by the model itself.

CHAPTER 3

MODELING DELETION REQUESTS IN MACHINE UNLEARNING

3.1 Introduction

In recent years, the right to erasure has gained recognition from an increasing number of legislative and regulatory bodies. Legislation like the General Data Protection Regulation (GDPR) [European Parliament and Council of the European Union, 2016] in the EU and the California Consumer Privacy Act (CCPA) [California State Legislature, 2018] in California give individuals the right to request their data be deleted from databases operated by private individuals or organizations.

The specific rights conferred by these laws are often textually limited: restrictions are placed on the types of information that are covered and distinctions are drawn between public and private information. However, any attempt to confer such a right is inherently limited by the technical difficulty of erasing data. In the current age of AI, personal data is aggregated and transformed into the parameters of enormous machine learning models, where determining the exact contribution of an individual's data on those parameters is oftentimes intractable. This has spurred the development of a line of research known as *machine unlearning*, where the goal is to design efficient methods to remove the influence of an individual's data on a model's weights.

Machine unlearning is often viewed as a privacy-enhancing technology. However, this thesis claims that the implications of unlearning in combination with the right to erasure are not limited to privacy. We argue that *machine unlearning facilitates a collective right to dataset curation.*

This thesis demonstrates how collectives can exert influence over the behavior of machine learning models when individuals are granted the right to remove their data from those

models. Our theoretical results explore how different types of collective behavior exhibit different levels of influence over models.

3.1.1 Chapter Summary

Characterizing Deletion Requesters

Inspired by prior work, we present definitions of machine unlearning and deletion requesters. We then present two qualities that can characterize the behavior of requesters: *adaptivity*, or the ability of a requester to respond to previous requests and model updates, and *collectivity*, or the ability to act on knowledge of the full training dataset. We also define what it means for a requester to be optimal, tying it to a quantity we call requester utility.

Understanding Gaps in Performance

We define a simple learning setting within which we study the optimal requesters of different types. Our main theoretical result is a reduction from the behavior of an optimal unlearning requester to a solution to an instance of the stochastic knapsack problem. We show separations between optimal collective and individual requesters, as well as between adaptive and nonadaptive requesters. These results imply that (1) personal information shared between individuals and (2) information leaked by the model itself could alter the impacts of deletion requests on model behavior.

Designing Strong Requesters

Since solving the knapsack problem is often intractable, we show how one can design requesters that outperform random requests in terms of utility. Drawing parallels between unlearning and data valuation, we design a requester based on the Shapley value, as well as adaptive and nonadaptive variants. We show that although the Shapley requester is

suboptimal in certain settings, it shows a clear advantage over random requests in image classification experiments. We also empirically study how adaptivity and collectivity affect the performance of Shapley-based requesters.

3.2 Related Work

3.2.1 Machine Unlearning

Machine Unlearning was first defined in its modern form by Cao and Yang [2015], who propose modifications to existing unlearning systems to allow for unlearning. Ginart et al. [2019] and Bourtoule et al. [2021] provide more formal definitions and propose unlearning algorithms for more general learning algorithms. Guo et al. [2019] provide definitions for certified machine unlearning, which formalizes the indistinguishability between models after unlearning and models that are retrained from scratch.

3.2.2 Evaluations of Unlearning

A simple framework for evaluating an unlearning method is as follows. First, a model is trained on a training dataset. That dataset is then split into a forget set (the set of points to be unlearned) and a retain set (the set of points to be retained). The unlearning method is applied to the original model to remove the influence of the forget set, and a new model is trained from scratch on the retain set. These models are then used to measure the performance of the unlearning method.

Within the machine unlearning literature, there is little standardization in how performance is measured. Li et al. [2025] provide a taxonomy of existing approaches. The metrics used to measure unlearning performance are divided into verification metrics, which can be used to test whether data in the forget set has been fully removed, and evaluation metrics, which measure how well an unlearning method approximates retraining from scratch.

Another aspect of machine unlearning evaluations is the selection method for the forget set A common approach is random forgetting: the forget set is sampled uniformly from the training set. Another approach is class unlearning, where the entirety of a given class is deleted from the training set. Other papers seek to unlearn a collection of points that share certain qualities. For example, unlearning all of the points that have been targeted by a data poisoning attack, or the set of points that encode a certain behavior in a generative model.

A distinct line of work has studied how the choice of forget set can impact the *fairness* of a model. Several papers Oesterling et al. [2024], Zhang et al. [2024], Liu and Shen [2025] investigate fairness impacts when deletion requests are biased towards minority or majority groups. Surve and Pradhan [2025] study the use of machine unlearning as an explainability technique for understanding fairness violations. They propose optimization techniques to identify the subset of the training data that contributes the most to the unfairness of the model.

Common definitions of machine unlearning fail to provide privacy guarantees when requests are allowed to be dependent on previous deletions and model updates. Gupta et al. [2021] were the first to formalize privacy guarantees in this setting, which they called adaptive machine unlearning. Follow up works [Chourasia and Shah, 2023, Cohen et al., 2023, 2026] further refine and strengthen these guarantees, although non-adaptive definitions continue to be more popular in the literature.

3.2.3 Data Valuation

Our approach to evaluating the impact of deletion requests is inherently tied to the topic of data valuation. A sizable body of work supports the use of the Shapley value to assign importance scores to individual data points, especially in machine learning tasks. Other papers have argued for the use of other valuation algorithms, like those based on the least core (Yan and Procaccia [2021]) and influence functions (Choe et al. [2024]).

3.3 Modeling Deletion Requests

3.3.1 Notation and Problem Setup

We will work with an arbitrary data domain $\mathcal{X}$ with labels in $\mathcal{Y}$ and a learning algorithm $\mathcal{A} : (\mathcal{X} \times \mathcal{Y})^* \to \Theta$. Let $\mathcal{R}_{\mathcal{A}} : \Theta \times (\mathcal{X} \times \mathcal{Y}) \to \Theta$ be an unlearning algorithm. We work with datasets $D \in (\mathcal{X} \times \mathcal{Y})^*$ sampled from some data distribution $\mathcal{D}$ and labeled according to a binary labeling function $f : \mathcal{X} \to \{0, 1\}$, sampled from some distribution $\mathcal{F}$ over binary functions. For a generic set $S = \{s_1, \ldots, s_n\}$, we use the notation S_{-i} to denote the set containing all but the i^{th} item in S.

Adopting the conventions from Ginart et al. [2019], we present definitions of *exact machine unlearning* and *approximate machine unlearning*.

Definition 3 (Exact Machine Unlearning [Ginart et al., 2019])**.** *An* unlearning algorithm *for a learning algorithm $\mathcal{A}$, $R_{\mathcal{A}} : (\mathcal{X} \times \mathcal{Y})^* \times \Theta \times \mathbb{N} \to \Theta$, is a function that maps a labeled dataset $D \in (\mathcal{X}, \mathcal{Y})^*$, a model θ, and an index $i \in \{1, \ldots, |D|\}$ to some new model $\theta' \in \Theta$. $R_{\mathcal{A}}$ satifies* exact machine unlearning *if, for all $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ and $i \in \{1, \ldots, |D|\}$, the random variables $\mathcal{A}(D_{-i})$ and $R_{\mathcal{A}}\big(D, \mathcal{A}(D), i\big)$ are equal in distribution:*

$$\mathcal{A}\big(D \setminus \{(x_i, y_i)\}\big) =_d R_{\mathcal{A}}\big(D, \mathcal{A}(D), i\big).$$

Definition 4 (Approximate Machine Unlearning [Ginart et al., 2019])**.** *Unlearning algorithm $R_{\mathcal{A}}$ satifies* δ-approximate machine unlearning *if, for all $D' \in (\mathcal{X} \times \mathcal{Y})^*$ and every measurable subset $S \subseteq \Theta$:*

$$\Pr_{D \sim \mathcal{D}}\Big[\mathcal{A}\big(D_{-i}\big) \in S \mid D_{-i} = D'_{-i}\Big] \geq \delta \cdot \Pr_{D \sim \mathcal{D}}\Big[R_{\mathcal{A}}\big(D, \mathcal{A}(D), i\big) \in S \mid D_{-i} = D'_{-i}\Big].$$

3.3.2 Deletion Requesters

The object of study in this paper is *deletion requesters*. Requesters have been objects of study in prior work, particularly in research on adaptive machine unlearning [Gupta et al., 2021, Chourasia and Shah, 2023, Cohen et al., 2026]. A deletion requester chooses which points are deleted from a model and the order in which those points are deleted. The requester and the unlearning algorithm take part in a multi-round exchange. At each round, indexed by t, the requester outputs a deletion request $u^t \in \mathcal{X} \times \mathcal{Y}$ consisting of a labeled training sample to be deleted. We denote the state of the training dataset after round t as D^t, where $D^0 = D$ and $D^t = D^{t-1} \setminus \{u^t\}$. After receiving the unlearning request, the unlearning algorithm outputs the updated state of the model θ^t, such that $\theta^t = R_{\mathcal{A}}(D^{t-1}, \theta^{t-1}, u^t)$ and $\theta^0 = \mathcal{A}(D^0)$. We now formally define deletion requesters. This definition encompasses all deletion requesters; later, we will define specific subclasses of requesters, specifically those exhibiting adaptivity and collectivity.

Definition 5 (Deletion Requesters)**.** *A deletion requester is a (possibly randomized) stateful mapping* $\texttt{Req} : (\mathcal{X}\times\mathcal{Y})^* \times \Theta \times ((\mathcal{X}\times\mathcal{Y})^* \times \Theta)^* \to (\mathcal{X}\times\mathcal{Y})$ *that takes as input an initial training dataset D and the history of interaction between itself and the algorithms, and outputs a new datapoint to be deleted in the current round. Given a deletion requester* $\texttt{Req}$ *and algorithms $(\mathcal{A}, \mathcal{R}_{\mathcal{A}})$, the deletion sequence $U = \{u^t\}_t$ can be written as*

$$
\begin{aligned}
u^1 &= \texttt{Req}(D, \theta^0), \\
u^2 &= \texttt{Req}(D, \theta^0, u^1, \theta^1), \\
&\dots \\
u^t &= \texttt{Req}(D, \theta^0, u^1, \theta^1, \dots, u^{t-1}, \theta^{t-1}).
\end{aligned}
$$

One quality of requesters we study in this chapter is *adaptivity*. Adaptive requesters are able to make decisions about which deletion requests to issue based off of previous model

updates. We motivate this concept with the following examples:

> **Example 1 (Adaptivite Requester).** An online querying interface for a new large language model is made public in a jurisdiction that allows individuals to request their data be deleted from model weights. The deployer of the model promised to unlearn each request and update the model weights immediately after the request is received. Alice interacts with the model soon after it is released and decides against requesting her data be deleted. However, after a few weeks during which the weights of the model were updated several times, Alice learns that the model is now regurgitating her personal information and she decides to request the removal of her data.

> **Example 2 (Nonadaptive Requester).** A company is developing an open-weight model but wants to give the public the opportunity to request their data be deleted from the model before the weights are published. An online querying interface for the initial state of the model is made public for a certain period of time, and individuals are allowed to request their data be deleted. After that time period, all of the requests that the company received will be unlearned together, and the final weights of the model will be published. During the querying phase, Bob interacts with the model and finds that the model does not regurgitate his personal information so he does not request deletion. After the deletion requests are unlearned and the final model is published, however, the model does regurgitate Bob's personal information.

These examples reflect how an individual might come to different conclusions about whether to delete their data depending on whether they can act adaptively. Both Alice and Bob will request deletion if the model leaks information about them. Both Alice and Bob were comfortable with their data being present in the initial state of the model, but only Alice

was able to delete her data after the model started leaking it because she was able to react to the effects of other deletion requests. We present a formal definition of adaptive and nonadaptive deletion requesters below.

Definition 6 (Nonadaptive/Adaptive Requesters)**.** *A deletion requester* `Req` *is nonadaptive if it is independent of all models after time step 0, i.e., if there exists a mapping* $\texttt{Req}' : (\mathcal{X} \times \mathcal{Y})^* \times \Theta \times \mathbb{N} \to (\mathcal{X} \times \mathcal{Y})$ *such that for all* $t \geq 1$*,*

$$u^t = \texttt{Req}\big(D, \theta^0, u^1, \theta^1, u^2 \ldots, u^{t-1}, \theta^{t-1}\big) = \texttt{Req}'(D, \theta_0, t).$$

We refer to requesters that are not nonadaptive as adaptive.

This definition of nonadaptive requesters equivalent to saying that the deletion sequence is fixed before the the first deletion request is sent.

The formulations we present here are adapted from the work of Gupta et al. [2021], but makes some simplifications. In particular, Gupta et al. [2021] allows for the deletion and addition of samples, while we focus specifically on deletions. They also consider a publishing function that passes information about the model to requesters. For simplicity, our requesters are instead able to observe the full state of the model at each time step. The corresponding definition from Gupta et al. [2021] of nonadaptive requesters requires that the output of `Req` is independent of all models, including θ_0. However, we are interested in requesters that are able to see the initial state of the model, but must fix the order of deletion requests before the model is updated.

We now introduce a new property of requesters which we term *collectivity*. We say a requester is *collective* if it can make decisions about which samples to delete using full knowledge of the training dataset. We say a requesters is *individual* if it must independently assign a value to each sample individually and only request the deletion of the samples whose value exceeds a certain threshold. We illustrate collective and individual requesters with the

following examples.

> **Example 3 (Individual Requester).** A model is trained on data collected from participants in a medical study. Participants know that their data was used in training, but for privacy reasons do not know each other's identities and are unable to communicate with one another. When making the decision to request the deletion of their data, participants therefore have no knowledge of the training dataset beyond what is revealed by the model itself.

> **Example 4 (Collective Requester).** A company releases a language model trained on books from a certain literary genre. Authors of books in that genre are represented by a guild. Some of the individual authors are concerned about the presence of their work in the training data, but defer to the guild on whether or not to request their data be deleted. The guild holds a vote among its members and decides that they will all request the deletion of their data.

In Example 3, participants in the study are incapable of exchanging information with one another, so each participant's decision to remove their data is independent of the training dataset conditioned on the initial state of the model. We model this behavior by defining a scoring function which takes in only a training sample and a model and outputs a real number. We then send deletion requests in decreasing order by score. For collective requesters, no such restrictions are imposed. As in example 4, these requesters may use full knowledge of the training dataset to make its decisions. A formal definition of individual and collective requesters is presented below.

Definition 7 (Individual/Collective Requesters)**.** *A deletion requester is* individual *if it can be modeled as taking the highest value datapoints according to a deterministic scoring function* $s : (\mathcal{X} \times \mathcal{Y}) \times \Theta \to \mathbb{R}$ *without ties. Concretely, for a given time step* t *in the adaptive*

setting,

$$u^t = \underset{(x,y)\in D^{t-1}}{\arg\min}\ s\Big((x,y),\theta^{t-1}\Big),$$

where $D^{t-1} = D \setminus \{u^1,\ldots,u^{t-1}\}$. *In the nonadaptive setting* $u_t = s_t$, *where* $s_1,\ldots s_{m'}$ *are the pairs* $(x_i,y_i) \in D$ *sorted in ascending order by* $s\big((x,y),\theta^0\big)$. *We refer to requesters that are not* individual *as* collective *requesters.*

The intention of prohibiting ties in the scoring function of individual requesters was to enforce a meaningful distinction between individual and collective requesters. If ties could be broken arbitrarily and s was a constant function, then an individual requester could implement a collective requester through its tie-breaking behavior. A more natural definition might be to allow for ties but require that they be broken randomly. However, this makes the analysis of the optimal individual requester much more difficult for reasons we explain in Section 3.4.4. An equivalent definition to our would be to allow for ties in s but require that ties be broken lexicographically. We note that given our definition of individual requesters, identical samples in the training dataset will always be removed sequentially.

It is worth noting that in our definitions, adaptivity and collectivity are binary properties; a requester is either adaptive (individual) or it is nonadaptive (collective). An interesting future direction would be to look at the *degree* to which a requester exhibits adaptivity or collectivity. For example, if requests came from small groups of individuals who knew each other's data but not the data of others outside of their group, or if individuals had incomplete or coarse knowledge of the training dataset. These requesters are not individual, but the classes they represent may be meaningfully different than the class of collective requesters that utilize full knowledge of the training data. We leave the study of these *semi-adaptive* and *semi-collective* requesters to future work.

3.3.3 Requester Utility

One object of our study is the interaction between requesters and *model performance*. We define a model performance metric as a function $g : \Theta \to \mathbb{R}$ which gives a real valued measurement of some property of a model. We intentionally keep the notion of model performance broad in order to capture the many possible objectives of model training. Works in the machine unlearning literature tend to measure aspects of model performance before and after unlearning as a way to evaluate the suitability of proposed methods. The most common metric of this sort is test accuracy, which is largely standard across evaluations, but other metrics like group fairness [Oesterling et al., 2024, Chen et al., 2023] or empirical privacy [Carlini et al., 2022b, Chen et al., 2021] have also been the focus of study.

What is often not considered in prior work is the impact that the deletion requester itself can have on measurements of model performance after unlearning. Different distributions of unlearning requests can lead to vastly different performance in downstream models. For example, unlearning the entirety of a class will likely degrade test accuracy much more than unlearning uniformly distributed test samples. It would therefore be natural to characterize deletion requesters by the degree to which they alter model performance. We define a requester's utility after t rounds as the difference in this measurement between θ^0 and θ^t, where θ^t is the state of the model after t rounds of unlearning.

Definition 8 (Requester Utility)**.** *Let D be a training dataset, $\theta^0 := \mathcal{A}(D)$ be an initial model. Let θ^t be the state of the model after t rounds of interaction between a deletion requester and a model trainer. A requester* `Req`*'s utility at time step t with respect to $g : \Theta \to \mathbb{R}$ (and implicitly with respect to $\mathcal{A}$ and $R_{\mathcal{A}}$) is defined as*

$$\texttt{util}_{\texttt{Req},g}(\theta^0, D, t) = g(\theta^t) - g(\theta^0).$$

Requester utility is defined as a random variable with randomness over $\mathcal{A}$, $R_{\mathcal{A}}$, and `Req`.

At times, we will also reason about the *expected requester utility*:

Definition 9 (Expected Requester Utility)**.** *Let $\mathcal{D}$ be a be a distribution on training datasets in $(\mathcal{X} \times \mathcal{Y})^*$. A requester* Req*'s expected utility at time step t with respect to $g : \Theta \to \mathbb{R}$ is defined as*

$$\overline{\texttt{util}}_{\texttt{Req},g}(\mathcal{D}, t) = \mathop{\mathbb{E}}_{\substack{f \sim \mathcal{F} \\ D \sim \mathcal{D}}} \texttt{util}(\mathcal{A}(D), D, t).$$

3.3.4 Optimal Requesters

We seek to characterize the abilities of *optimal requesters*: those that maximize expected requester utility. For a given performance function g, we define the optimality of a deletion requester with respect to the data distribution $\mathcal{D}$, and number of deletions k:

$$\texttt{opt}_g(\mathcal{D}, k) = \sup_{\texttt{Req}} \overline{\texttt{util}}_{\texttt{Req},g}(\mathcal{D}, k). \tag{3.1}$$

This optimum is also dependent on $\mathcal{A}$ and $R_{\mathcal{A}}$, although we omit them from the notation for the sake of brevity.

3.4 Characterizing Unlearning Requesters

Our goal is to show whether the behavior of a deletion requester (i.e. adaptive vs. non-adaptive, individual vs. collective) has any influence over its maximum achievable utility. To do so, we instantiate a simple learning task for which it is easier to reason about and characterize the optimal learning requester which possesses certain qualities. Our model is loosely based on that of Feldman [2020], which studies the impact of data memorization on generalization. We will show that, in the setting we consider, we can determine lower bounds on the gaps between the utilities of the optimal adaptive and nonadaptive requesters, as well as the optimal collective and individual requesters.

3.4.1 A Simplified Learning Setting

We instantiate a data domain $\mathcal{X}$ consisting of m subpopulations (i.e. $\mathcal{X} = [m]$). We define a distribution of datasets $\mathcal{D}$ on $(\mathcal{X} \times \mathcal{Y})^*$ parameterized by m values $\lambda_1, \lambda_2, \ldots, \lambda_m$ such that $\lambda_i \in \mathbb{R}^+$. To sample a dataset D from $\mathcal{D}$, we first sample a labeling function $f \sim \mathcal{F}$, where $\mathcal{F}$ is the uniform distribution over all binary functions on $\mathcal{X}$. For each $i \in \mathcal{X}$ we sample $n_i \sim \text{Poisson}(\lambda_i)$ and add n_i copies of item $(i, f(i))$ to D. Owing to properties of the Poisson distribution, the expected size of D is N, where $N = \sum_{i \in \mathcal{X}} \lambda_i$. Additionally, since the expectation of n_i is λ_i, the expected proportion of $D \sim \mathcal{D}$ made up of copies of subpopulation i is $p_i = \frac{\lambda_i}{N}$. We refer to p_i as the *population frequency* of subpopulation i.[1] We emphasize that D is a multiset, in which a given point i can (and likely will, for large enough N) occur multiple times. There is no label noise in this model; every sample from the same subpopulation will have the same label.

The fundamental learning task is to learn the function f from labeled data. We measure the performance of a model $\theta : (\mathcal{X} \times \mathcal{Y}) \to \{0, 1\}$ through population error:

$$\texttt{err}(\theta) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \mathbb{1}[\theta(x) \neq y].$$

Given a distribution $\mathcal{D}$, the goal of an optimal learning algorithm is to minimize population error:

$$\texttt{err}^*(\mathcal{D}) := \inf_{\mathcal{A}} \mathop{\mathbb{E}}_{D\sim\mathcal{D}} \texttt{err}(\mathcal{A}(D)).$$

We now show that in our learning setting, the learning algorithm that simply memorizes the labels of the samples included in the training dataset and randomly guesses on the rest is optimal. A similar result is proved in Feldman [2020, Theorem 2.3]. We provide a simplified justification below.

1. For the sake of clarity, we tend to parameterize $\mathcal{D}$ by N and $p_1, \ldots, p_m$ rather than $\lambda_1, \ldots, \lambda_m$. These formulations are equivalent.

Claim 1. *In this setting, the minimum population error* $\texttt{err}^*(\mathcal{D})$ *is achieved by the learning algorithm* $\mathcal{A}^*$ *which yields a model* $\theta^* := \mathcal{A}^*(D)$ *of the form*

$$\theta^*(x) = \begin{cases} y & \text{if } (x,y) \in D \\ \mathbb{1}[H=1] & \text{otherwise,} \end{cases}$$

where $H \sim$ *Bernoulli* $\left(\frac{1}{2}\right)$.

Proof. In order for $\mathcal{A}^*$ to be optimal, it must minimize the following objective:

$$\inf_{\mathcal{A}} \mathop{\mathbb{E}}_{D\sim\mathcal{D}} \texttt{err}(\mathcal{A}(D)).$$

Expanding this expression, we get

$$\inf_{\mathcal{A}} \mathop{\mathbb{E}}_{\substack{D\sim\mathcal{D} \\ \theta\leftarrow\mathcal{A}(D)}} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \mathbb{1}[\theta(x) \neq y]$$

We can decompose the inner expectation into error on points that are in D and error on points that are not in D.

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\substack{D\sim\mathcal{D} \\ \theta\leftarrow\mathcal{A}(D)}} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \mathbb{1}[\theta(x) \neq y] \\
&\quad= \mathop{\mathbb{E}}_{\substack{D\sim\mathcal{D} \\ \theta\leftarrow\mathcal{A}(D)}} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \Big[\mathbb{1}[(x,y)\in D]\cdot\mathbb{1}[\theta(x)\neq y] + \mathbb{1}[(x,y)\notin D]\cdot\mathbb{1}[\theta(x)\neq y]\Big] \\
&\quad= \mathop{\mathbb{E}}_{\substack{D\sim\mathcal{D} \\ \theta\leftarrow\mathcal{A}(D)}} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \mathbb{1}[\theta(x)\neq y \wedge (x,y)\in D] + \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \mathbb{1}[\theta(x)\neq y \wedge (x,y)\notin D] \\
&\quad= \mathop{\mathbb{E}}_{\substack{D\sim\mathcal{D} \\ \theta\leftarrow\mathcal{A}(D)}} \Pr_{(x,y)\sim\mathcal{D}}[\theta(x)\neq y \wedge (x,y)\in D] + \Pr_{(x,y)\sim\mathcal{D}}[\theta(x)\neq y \wedge (x,y)\notin D].
\end{aligned}$$

The left inner probability is minimized by $\mathcal{A}^*$. Letting $\theta^* = \mathcal{A}^*(D)$ $\theta^*(x) = y$ for all

$(x, y) \in D$, so this probability will be 0. By Bayes' rule, the right inner probability is equivalent to

$$\Pr_{(x,y)\sim\mathcal{D}}[\theta(x) \neq y \mid (x, y) \notin D] \cdot \Pr[(x, y) \notin D].$$

$\Pr[(x, y) \notin D]$ is independent of $\mathcal{A}$, since $\mathcal{A}$ and $\mathcal{D}$ are fixed before the dataset is sampled. Let f be the labeling function used to label the points in D. Now, consider a point $x \in \mathcal{X}$ such that $(x, f(x)) \notin D$. Since the labeling function for D was sampled from the uniform distribution over all binary labeling functions, its label is independent of the label of any subpopulation in D. This implies that $\theta = \mathcal{A}(D)$ is also independent of $f(x)$ for all learning algorithms $\mathcal{A}$ that are independent of f conditioned on the training dataset D. Therefore, the right probability above is a constant which is minimized by any learning algorithm, so $\mathcal{A}^*$ minimizes the objective.

□

Since the learning algorithm is memorizing training data and there is no label noise, the resulting model will be correct on all the subpopulations it has seen during training and must guess with 50% accuracy on all the subpopulations it hasn't seen. We can therefore characterize expected population error of a model output by this learning algorithm:

$$\texttt{err}(\mathcal{A}^*(D)) = \sum_{i\in[m]} \frac{p_i \cdot \mathbb{1}[(i, f(i)) \notin D]}{2}.$$

3.4.2 Instantiating the Unlearning Problem

The unlearning algorithm we focus on is retraining from scratch:

$$R_{\mathcal{A}}(\theta, D, i) = \texttt{Retrain}_{\mathcal{A}}(\theta, D, i) = \mathcal{A}(D_{-i}).$$

Seeing as retraining is the "gold standard" for unlearning, this choice allows us to abstract away any failures of the unlearning algorithm and focus on the impact of deletion requests

on the downstream models.

As we allude to above, the measurement function $g(\theta)$ in this setting is $\texttt{err}(\theta)$. Therefore, the goal of the optimal unlearning requester is to maximize the population error of the model output after t rounds of unlearning. Within this setting, we can intuitively understand how a requester would achieve this goal. In an interaction with the optimal learning algorithm $\mathcal{A}^*$, a deletion requester will only change the model if every instance of a subpopulation is deleted from the model. Consider the utility of a requester $\texttt{Req}$ after k rounds of unlearning, letting U be the multiset of k points that are unlearned. Below, we use the notation S_i to denote the number of times subpopulation i occurs in a given multiset S.

$$\begin{aligned}
\texttt{util}_{\texttt{Req},\texttt{err}}(\theta^0, D, k) &= \texttt{err}(\theta^k) - \texttt{err}(\theta^0) \\
&= \frac{1}{2} \sum_{i \in [m]} p_i (\mathbb{1}[(i, f(i)) \notin D \setminus U] - \mathbb{1}[(i, f(i)) \notin D]) \\
&= \frac{1}{2} \sum_{i \in [m]} p_i \cdot \mathbb{1}[U_i > 0 \text{ and } U_i = D_i]).
\end{aligned}$$

This shows us that the utility of a requester in this setting is completely determined by the total population frequency of the subpopulations that are completely removed from D. The optimal requester will therefore maximize this value.

3.4.3 Optimal Requesters and the Knapsack Problem

We now ask: if a requester is constrained to a certain number of deletion requests, what is the maximum utility the requester can expect to achieve? In this learning setting, it turns out that an optimal deletion requester (either adaptive or nonadaptive) must solve an instance of the knapsack problem. In the knapsack problem, we are given a list of *items* $1, \ldots, m$, corresponding *weights* $w_1, \ldots, w_m \in \mathbb{R}^+$ and *values* $v_1, \ldots, v_m \in \mathbb{R}^+$ for each item, and a knapsack with a given *capacity* $k \in \mathbb{N}$. The goal is to find the subset of items $S \subseteq [m]$

maximizing $\sum_{i \in S} v_i$ subject to $\sum_{i \in S} w_i \leq k$.

We reiterate that the task of the optimal requester is to increase the population error (thereby decreasing the accuracy) of a model as much as possible. In this setting, error will only increase if all samples from a subpopulation are removed from D. In particular, once a subpopulation is fully removed, the accuracy of a model will decrease by that subpopulation's population frequency. We can therefore restate the task of the unlearning requester: Given a number of unlearning requests k (*capacity*), find the subset of subpopulations for whom the number of times they appear in D (*weights*) sum to no more than k and the sum of whose population frequencies (*values*) is maximized.

It is straightforward to show that an optimal *collective* unlearning requester is solving an instance of the knapsack problem. Collective requesters can act on full knowledge of the training dataset and therefore are able to compute the weights corresponding to each subpopulation ($w_i = n_i \ \forall \ i \in [m]$). Subpopulation frequencies are properties of the data distribution and accessible to any requester ($v_i = p_i \ \forall \ i \in [m]$). Since we are not placing any restrictions on running time, a collective requester can simply compute the optimal solution to the knapsack problem and request the deletion of training samples represented by the subpopulations in the solution.

The connection between *individual* requesters and knapsack is more complex. As stated above, all requesters can make use of subpopulation frequencies, but individual requesters must fix their scoring function s before observing D, meaning they cannot act on full knowledge of the weights. Instead, individual requesters can only act on knowledge of the distribution that the weights were drawn from: in this case, a Poisson centered at λ_i.

We note that while an individual requester does not a priori know the exact multiplicity of each subpopulation in the dataset, it is able to sequentially request the deletion of complete subpopulations. In fact, since we require scoring functions be deterministic and without ties, individual requesters are only capable of removing whole subpopulations sequentially.

(We further discuss this restriction on individual requesters in Section 3.4.4.) Similarly, an individual requester can determine when a complete subpopulation has been removed by observing the point when the accuracy of the model decreases.

The optimal individual requester turns out to be solving an instance of the stochastic knapsack problem, a setting which has been studied extensively in algorithms literature [Rothkopf, 1966, Dean et al., 2008, Bhalgat et al., 2011]. Instead of being fixed, item weights in the stochastic knapsack problem are independent random variables drawn from known distributions. An algorithm placing items into the knapsack is only able to see an item's realized weight after it is inserted; once an item overflows the knapsack, execution is terminated. The requirement of the stochastic knapsack problem that item sizes be independently sampled motivated our choice of data distribution. We note that the distribution we define is equivalent to sampling from a multinomial distribution with N trials and event probabilities $(p_1, \ldots, p_m)$, where N is sampled from a Poisson distribution.

A difference between unlearning in our setting and the standard formulation of the stochastic knapsack problem is the correlation between item values and sizes. In stochastic knapsack, item values must be independent of weights, whereas the deletion of a subpopulation increases requester utility by its population frequency unless its multiplicity in the training dataset is 0. This corresponds to a variant of the stochastic knapsack problem known as the correlated stochastic knapsack problem [Gupta et al., 2011], in which the value of an item is correlated with its realized weight. Specifically, each subpopulation has an associated probability distribution over (size, reward) pairs $(\pi_{i,t}, R_{i,t})$, where $\pi_{i,t}$ represents the probability that subpopulation i has instantiated weight t and $R_{i,t}$ is the value of subpopulation i if it has instantiated size t. In our setting, $\pi_{i,t}$ reflects the distributions our weights are sampled from $(\pi_{i,t} = \Pr[\text{Poisson}(\lambda_i) = t])$ and $R_{i,t}$ is equal to 0 if $t = 0$ and p_i otherwise.

Gaps and Existing Results

Dean et al. [2008] study the *adaptivity gap* of the stochastic knapsack problem.

Definition 10 (Adaptivity Gap [Dean et al., 2008])**.** *Let* $\mathsf{Adapt}(I)$ *be the supremal expected value achieved by any adaptive policy (one which can change it's strategy after observing a realized item weight) on a problem instance* I*. Let* $\mathsf{Nonadapt}(I)$ *be the supremal expected value achieved by any nonadaptive policy (one which must fix insertion order before any items are inserted) on a problem instance* I*. The* adaptivity gap *is defined as*

$$\sup_I \frac{\mathsf{Adapt}(I)}{\mathsf{Nonadapt}(I)},$$

where the supremum is taken over all problem instances.

Dean et al. [2008] define the adaptivity gap over all instances of the stochastic knapsack problem. In this work, we study the adaptivity gap over the subset of instances induced by unlearning problems, making use of the definitions of adaptive and nonadaptive requesters in Definition 6. We also introduce what we call the *collectivity gap*: the ratio between the expected utility achieved by the optimal collective requester and optimal individual requester.

Definition 11 (Collectivity Gap)**.** *Let* $\mathsf{Coll}(I)$ *be the supremal expected value achieved by any collective policy on a problem instance* I*. Let* $\mathsf{Ind}(I)$ *be the supremal xpected value achieved by any individual policy on a problem instance* I*. The* collectivity gap *is defined as*

$$\sup_I \frac{\mathsf{Coll}(I)}{\mathsf{Ind}(I)},$$

where the supremum is taken over all problem instances.

Again, in this work we define the collectivity gap over all instances of the optimal unlearning problem, making use of the definitions of collective and individual requesters we present in Definition 7.

Gupta et al. [2011] give an upper bound for the adaptivity gap for correlated stochastic knapsack, constructing a nonadaptive algorithm which is an 8-approximation of the optimal adaptive algorithm. Thus, there will always exist a nonadaptive unlearning requester whose utility is at least $\frac{1}{8}$ that of optimal adaptive unlearning requester.

Dean et al. [2008] give a lower bound of 1.25 for the adaptivity gap for the standard stochastic knapsack, but this result does not apply to our instantiation of the correlated stochastic knapsack problem. Dean et al. [2008] arrive at this lower bound by formulating an instance of stochastic knapsack where the item weights are distributed according to carefully crafted distributions (which are not Poisson) and showing that the expected value of the optimal adaptive algorithm is at least 1.25 times greater than that of the optimal nonadaptive algorithm. It does not follow from this result that the optimal adaptive algorithm has a higher expected value than the optimal nonadaptive algorithm in the setting where item weights are distributed as Poissons, as they are in our instantiation of the problem. Barak and Talgam-Cohen [2026] improve the lower bound on the adaptivity gap for stochastic knapsack to 1.37, but they also make use of carefully crafted non-Poisson item weight distributions.

3.4.4 Randomized Individual Requesters

One restriction we are making on the abilities of individual requesters is that their associated scoring functions must be deterministic. Ideally, we would want to be able to model requesters characterized by randomized scoring functions as well. Under the following conjecture, which states that there exists an optimal randomized individual requester of a simple form, the problem of finding an optimal individual requester would also reduce to correlated stochastic knapsack as above.

Conjecture 1. *An optimal randomized individual deletion requester will never* preempt *a subpopulation. That is to say, such a requester will never begin removing samples from a certain population and switch to removing samples from a different subpopulation before the*

first subpopulation was completely removed.

We hypothesize that this conjecture is true for both adaptive and nonadaptive requesters when n_i is sampled from a Poisson distribution, as is the case in our setting. When a copy of a subpopulation is deleted, the expected number of copies off that subpopulation that remain in the training data will strictly decrease, while the utility gained by removing that subpopulation in its entirety will remain the same. In that sense, the *expected utility per deletion* (the ratio of utility gained by removing a complete subpopulation to number of remaining elements of that subpopulation) will increase for a given subpopulation after one of its elements is deleted and remain the same for all other subpopulations. It would make sense that a requester would not preempt a subpopulation whose expected utility per deletion is increasing in this way. However, this analysis overlooks the requester's total budget of k deletions, which may induce non-monotonicity in expected utility per deletion for a given subpopulation. In our research, we have failed to either prove this conjecture or find a counterexample. Gupta et al. [2011] demonstrate that there are instances of the stochastic knapsack problem where the *cancellation* of jobs is necessary, which is a stronger notion than preemption that would correspond to a requester preempting a subpopulation and never returning to it. However, like the prior lower bounds on the adaptivity gap of stochastic knapsack, this result requires that item sizes be sampled from carefully crafted non-Poisson distributions.

3.4.5 Understanding Gaps in Requester Utility

Here, we demonstrate the existence gaps between individual and collective requesters and nonadaptive and adaptive requesters in our learning model. The lower bounds we present show how requesters that take advantage of collective and adaptive behaviors can have more influence over models after unlearning.

Individual vs. Collective. The separation between individual nonadaptive[2] and collective requesters can be shown through a simple example. Let N be large. Consider the case where there are two subpopulations, each of which has a population frequency $p_i = \frac{1}{2}$ (i.e. $\mathcal{X} = \{1, 2\}$, $\mathcal{D} = \left(\frac{N}{2}, \frac{N}{2}\right)$). The counts of each subpopulation in our training dataset will be distributed as Poissons, as described in Section 3.4:

$$n_i \sim \text{Poisson}\left(\frac{N}{2}\right) \ \forall\, i \in \{1, 2\}.$$

Let $k = \frac{N}{2}$. Since n_1 and n_2 are independent and identically distributed, then for large N we would expect one to be greater than k and one to be less than k.

Theorem 2. *The collectivity gap in this learning model is at least* $\frac{3}{2}$.

Proof. Take $N \to \infty$. We know by the central limit theorem that Poisson $\left(\frac{N}{2}\right)$ approaches a Gaussian centered at $\frac{N}{2}$, so $\lim_{N\to\infty} \Pr[n_1 > k] = \lim_{N\to\infty} \Pr[n_2 > k] = 0.5$. Also, $\lim_{N\to\infty} \Pr[n_1 = 0] = \lim_{N\to\infty} \Pr[n_2 = 0] = 0$ and $\lim_{N\to\infty} \Pr[n_1 + n_2 \leq k] = 0$.

To determine a lower bound on the adaptivity gap, we will compute the values of both $\overline{\texttt{util}}_{\texttt{Coll},\texttt{err}}(\mathcal{D}, k)$ and $\overline{\texttt{util}}_{\texttt{Ind},\texttt{err}}(\mathcal{D}, k)$, where `Coll` is the optimal collective requester and `Ind` is the optimal individual requester. For a requester `Req`, the expected utility is defined as

$$\begin{aligned}\overline{\texttt{util}}_{\texttt{Req},\texttt{err}}(\mathcal{D}, k) &= \mathop{\mathbb{E}}_{D\sim\mathcal{D}}[\texttt{err}(\theta^k) - \texttt{err}(\theta^0)] \\ &= \mathop{\mathbb{E}}_{D\sim\mathcal{D}} \texttt{err}(\mathcal{A}(D \setminus U_{\texttt{Req}})) - \mathop{\mathbb{E}}_{D\sim\mathcal{D}} \texttt{err}(\mathcal{A}(D)),\end{aligned}$$

where $U_{\texttt{Req}}$ is the set of k deletion requests issued by `Req`. We therefore need to compute $\mathbb{E}_{D\sim\mathcal{D}}\, \texttt{err}(\mathcal{A}(D))$, $\mathbb{E}_{D\sim\mathcal{D}}\, \texttt{err}(\mathcal{A}(D \setminus U_{\texttt{Coll}}))$, and $\mathbb{E}_{D\sim\mathcal{D}}\, \texttt{err}(\mathcal{A}(D \setminus U_{\texttt{Ind}}))$

2. In this example, the set of individual adaptive and individual nonadaptive requesters are identical due to there only being two subpopulations. Since individual requesters are only capable of removing entire subpopulation sequentially, the entire unlearning sequence is determined by the first point that is unlearned.

First, we look at the expected error of the optimal learning algorithm on the original training dataset. Since it is very likely that both subpopulations appear in D, and since the optimal learning algorithm memorizes labels in the training dataset, we can conclude that

$$\mathop{\mathbb{E}}_{D\sim\mathcal{D}}[\texttt{err}(\mathcal{A}^*(D))] = 0.$$

A collective requester `Coll`, having full knowledge of n_1 and n_2, would be able to choose the subpopulation corresponding to the smaller n_i and request its deletion first, followed by the other subpopulation. The probability that `Coll` will be able to delete the entirety of the first subpopulation it removes is equal to the probability that $n_1 \leq k$ or $n_2 \leq k$, which is $\Pr[n_1 \leq k \text{ or } n_2 \leq k] = 1 - \Pr[n_1 > k \text{ and } n_2 > k] \approx 1 - \frac{1}{4} = \frac{3}{4}$. The probability that `Coll` will be able to delete the entirety of both subpopulations is equal to the probability that $n_1 + n_2 \leq k$, which as stated above is about 0. If `Coll` is able to delete a subpopulation, then the expected error will be $\frac{1}{4}$, since the optimal model will now have to guess with 50% accuracy on that subpopulation, which has a population freqency of $\frac{1}{2}$. If it is not, then the expected error will be 0 since the correct labels for both subpopulations are still present in the updated training dataset. Let A be the event that $n_1 \leq k$ or $n_2 \leq k$, and let B be the event that $n_1 > k$ and $n_2 > k$. By the law of total expectation,

$$\begin{aligned}
\mathop{\mathbb{E}}_{D\sim\mathcal{D}}[\texttt{err}(\mathcal{A}^*(D \setminus U_{\texttt{Coll}}))] &= \sum_{E\in\{A,B\}} \Pr_{D\sim\mathcal{D}}[E] \cdot \mathop{\mathbb{E}}_{D\sim\mathcal{D}}[\texttt{err}(\mathcal{A}(D \setminus U_{\texttt{Coll}})) \mid E] \\
&= \frac{3}{4} \cdot \frac{1}{4} + \frac{1}{4} \cdot 0 \\
&= \frac{3}{16}.
\end{aligned}$$

An individual requester `Ind`, on the other hand, would not know which of n_1 and n_2 is smaller and can therefore do no better than guessing. In the case where n_1 and n_2 are both greater than k (occurring with probability $\frac{1}{4}$), the expected error of the model after

unlearning will be 0. In the case where n_1 and n_2 are both less than or equal to k (occurring with probability $\frac{1}{4}$), the expected error of the model after unlearning will be $\frac{1}{4}$. In the case where one of n_1 and n_2 is less than or equal to k and the other is greater (occurring with probability $\frac{1}{2}$), `Ind` will choose a random subpopulation to delete and be correct with probability $\frac{1}{2}$, so the expected error of the model after unlearning will be $\frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8}$. Let A be the event that $n_1 \leq k$ and $n_2 \leq k$, let B be the event that $n_1 > k$ XOR $n_2 > k$, and let C be the event that $n_1 > k$ and $n_2 > k$. Then,

$$\begin{aligned}
\mathop{\mathbb{E}}_{D\sim\mathcal{D}}[\texttt{err}(\mathcal{A}^*(D \setminus U_{\texttt{Ind}}))] &= \sum_{E\in\{A,B,C\}} \Pr_{D\sim\mathcal{D}}[E] \cdot \mathop{\mathbb{E}}_{D\sim\mathcal{D}}[\texttt{err}(\mathcal{A}(D \setminus U_{\texttt{Ind}})) \mid E] \\
&= \frac{1}{4} \cdot \frac{1}{4} + \frac{1}{2} \cdot \frac{1}{8} + \frac{1}{4} \cdot 0 \\
&= \frac{1}{8}.
\end{aligned}$$

Using these values, we can compute that the expected utility of `Coll` is

$$\begin{aligned}
\overline{\texttt{util}}_{\texttt{Coll},\texttt{err}}(\mathcal{D}, k) &= \mathop{\mathbb{E}}_{D\sim\mathcal{D}}[\texttt{err}(\mathcal{A}^*(D \setminus U_{\texttt{Coll}}))] - \mathop{\mathbb{E}}_{D\sim\mathcal{D}}[\texttt{err}(\mathcal{A}^*(D))] \\
&= \frac{3}{16} - 0 \\
&= \frac{3}{16},
\end{aligned}$$

and the expected utility of `Ind` is

$$\begin{aligned}
\overline{\texttt{util}}_{\texttt{Ind},\texttt{err}}(\mathcal{D}, k) &= \mathop{\mathbb{E}}_{D\sim\mathcal{D}}[\texttt{err}(\mathcal{A}^*(D \setminus U_{\texttt{Ind}}))] - \mathop{\mathbb{E}}_{D\sim\mathcal{D}}[\texttt{err}(\mathcal{A}^*(D))] \\
&= \frac{1}{8} - 0 \\
&= \frac{1}{8}.
\end{aligned}$$

Therefore, the collectivity gap for unlearning is as least $\frac{\overline{\texttt{util}}_{\texttt{Coll},\texttt{err}}(\mathcal{D},k)}{\overline{\texttt{util}}_{\texttt{Ind},\texttt{err}}(\mathcal{D},k)} = \frac{3}{2}$. □

Nonadaptive vs. Adaptive. The intuition behind the existence of an adaptivity gap in this setting is simple: having observed the sizes of previously deleted subpopulations, a requester has more information to determine the optimal next subpopulation to remove. However, demonstrating an adaptivity gap in this model is not as straightforward as demonstrating a collectivity gap.

We note that in our setting, the adaptivity gap for collective requesters is 1 (i.e., there is no gap). This is due to the fact that there is no randomness involved in retraining from scratch in this learning setting, so an optimal collective requester is able to simulate the the complete interaction with the unlearning algorithm. An adaptivity gap may appear for collective requesters if we were to use an unlearning method other than retraining from scratch.

For simplicity, we will focus on the case of three subpopulations ($\mathcal{X} = \{1, 2, 3\}$). In this setting, the optimal nonadaptive requester `Nonadapt` outputs an ordering of the subpopulations a, b, c that maximizes expected utility. The utility of this sequence is defined by the following expression:

$$\begin{aligned}\overline{\texttt{util}}_{\texttt{Nonadapt},\texttt{err}}(\mathcal{D}, k) = p_a \cdot \Pr[n_a > 0 \text{ and } n_a \leq k] \\ + p_b \cdot \Pr[n_b > 0 \text{ and } n_a + n_b \leq k] \\ + p_c \cdot \Pr[n_c > 0 \text{ and } n_a + n_b + n_c \leq k].\end{aligned} \tag{3.2}$$

Our goal is to find an adaptive requester which has higher utility than the optimal nonadaptive requester. We will consider a restricted class of adaptive requesters which we call *thresholded adaptive requesters*.

Definition 12 (Thesholded Adaptive Requesters)**.** *Given an integer t,* thresholded adaptive requesters *are a class of requesters denoted by $\langle a, b, c \mid t\rangle$, where a, b, and c are distinct subpopulations in $\mathcal{X}$. Such a requester would first request the deletion of all instances of subpopulation a. If $n_a > t$, the requester will then request all instances of subpopulation b,*

followed by all instances of subpopulation c. Otherwise, if $n_a \leq t$, the requester will then request all instances of subpopulation c, followed by all instances of subpopulation b.

We now present the expected utility of $\langle a, b, c \mid t \rangle$:

$$\begin{aligned}
\overline{\texttt{util}}_{\langle a,b,c|t\rangle,\texttt{err}}(\mathcal{D}, k) = p_a \cdot \Pr[n_a > 0 \text{ and } n_a \leq k] \\
+ \Pr[n_a > t] \cdot (p_b \cdot \Pr[n_b > 0 \text{ and } n_a + n_b \leq k \mid n_a > t] \\
+ p_c \cdot \Pr[n_c > 0 \text{ and } n_a + n_b + n_c \leq k \mid n_a > t]) \\
+ \Pr[n_a \leq t] \cdot (p_c \cdot \Pr[n_c > 0 \text{ and } n_a + n_c \leq k \mid n_a \leq t] \\
+ p_b \cdot \Pr[n_b > 0 \text{ and } n_a + n_b + n_c \leq k \mid n_a \leq t]).
\end{aligned} \tag{3.3}$$

Theorem 3. *The adaptivity gap in this learning model is at least 1.064.*

Proof. Let $N = 10$ and $k = 8$. We perform a grid search over parameters p_1, p_2, and p_3 constrained such that $p_1 + p_2 + p_3 = 1$ with a step size of 0.01. For each choice of (p_1, p_2, p_3), we explicitly compute the expected utility of the optimal nonadaptive requester (Equation 3.2), as well as the expected utility of the all thresholded adaptive requesters of the form $\langle a, b, c \mid t \rangle$ (Equation 3.3) where (a, b, c) is a permutation of $(1, 2, 3)$ and $0 \leq t \leq k$. We define the *thresholded adaptivity gap* to be the gap between the optimal nonadaptive requester and the optimal thresholded adaptive requester under a certain set of parameters. Figure 3.1 plots the thresholded adaptivity gap over the parameters of our grid search. We observe a gap of approximately 1.011 when $p_1 = p_2 = 0.4$ and $p_3 = 0.2$. With these parameters, the optimal nonadaptive requester deletes subpopulations in the order $(1, 2, 3)$ and the optimal thresholded adaptive requester is $\langle 1, 3, 2 \mid 4 \rangle$.

In Figure 3.1, we observe that the highest values are achieved when two of the subpopulation frequencies are approximately equal and larger than the third probability. Based on this, we parameterize the subpopulation frequencies with a single parameter ε and expand

our search to include expected dataset size N. In Figure 3.2, we plot the result of a grid search over ε and N. We set $p_1 = p_2 = \epsilon$, $p_3 = 1 - 2\epsilon$, and $k = \lfloor 0.8N \rfloor$. In our grid search, we step over integer values of n between 10 and 50 and values of ϵ between 0.33 and 0.50 with a step size of 0.01. We observe a maximum thresholded adaptivity gap of approximately 1.064, significantly improving the result above. □

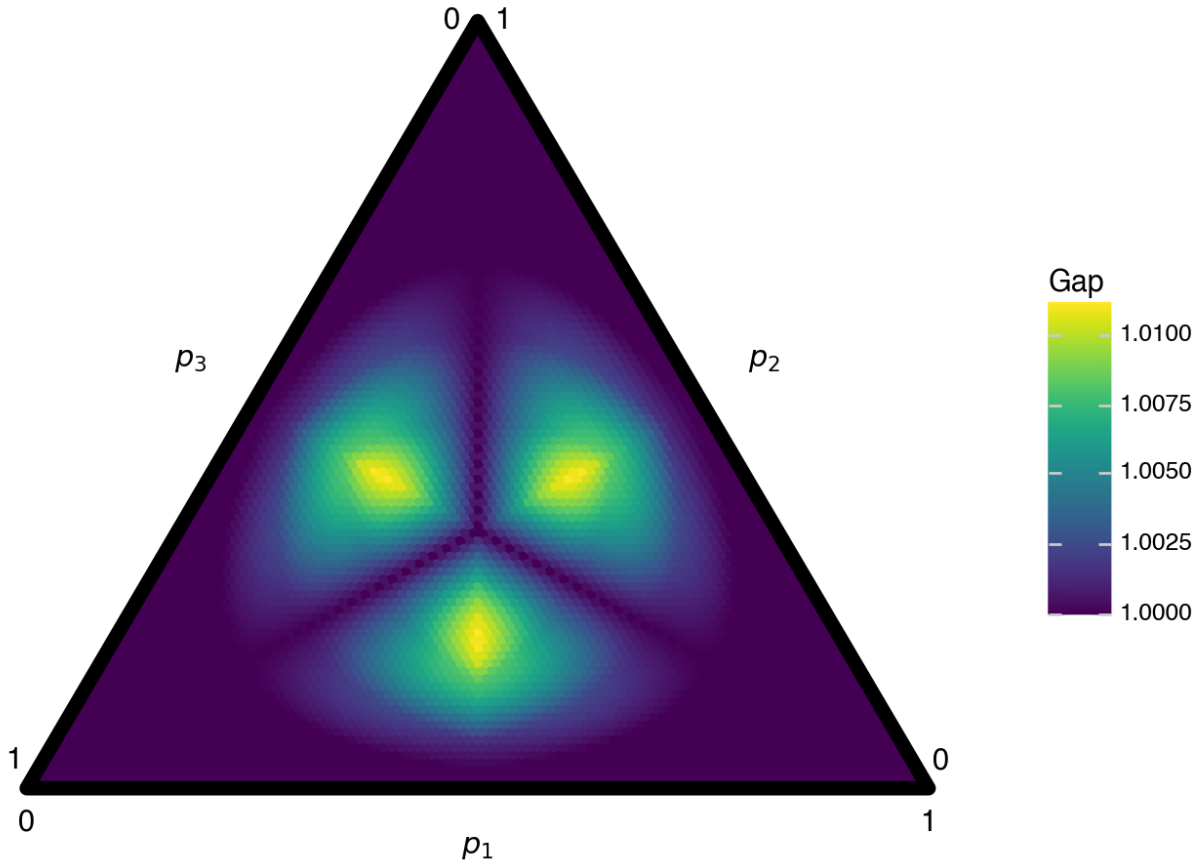


Figure 3.1: **Thresholded Adaptivity Gaps.** A ternary plot of the thresholded adaptivity gap for three subpopulations plotted over values of (p_1, p_2, p_3). In this plot, N is fixed to 10 and k is fixed to 8.

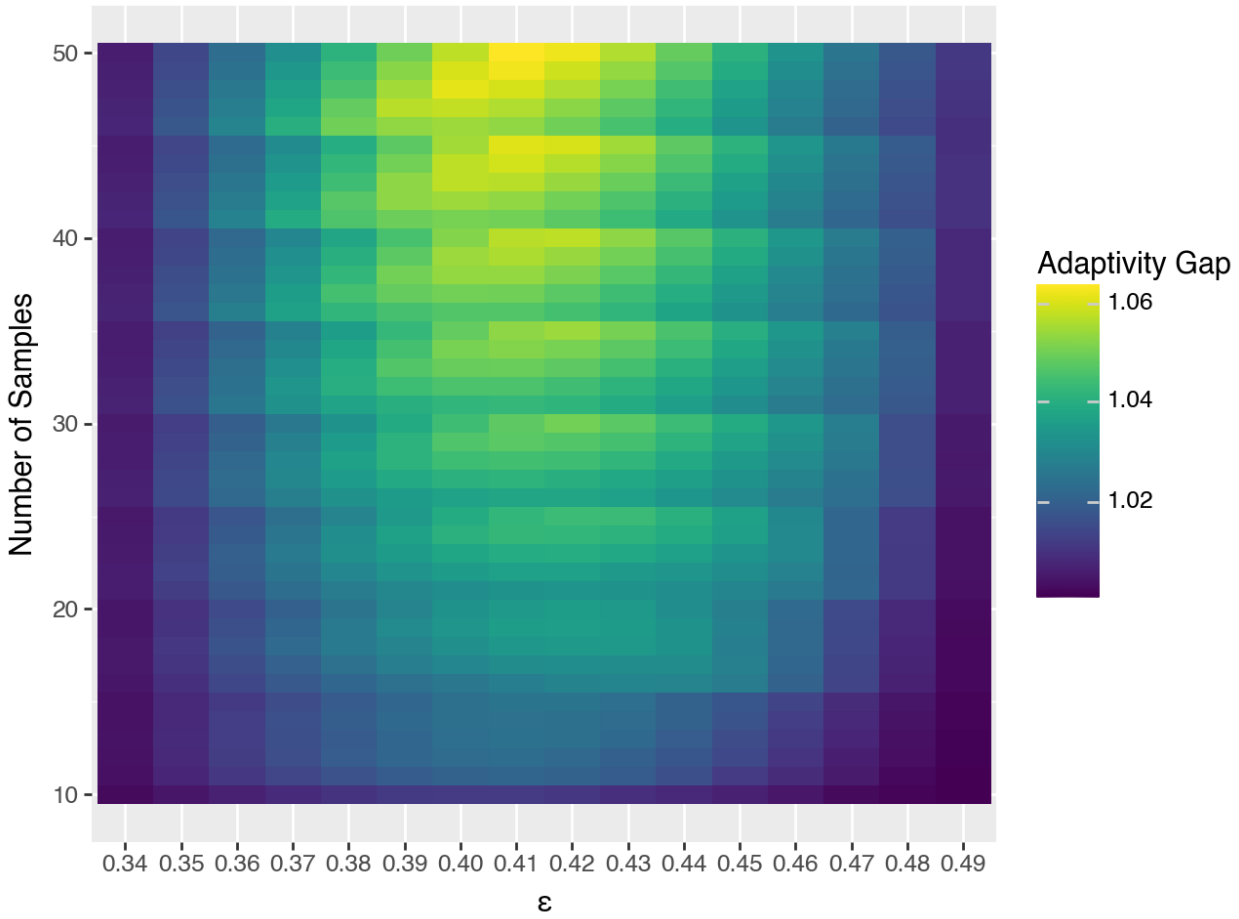


Figure 3.2: **Thresholded Adaptivity Gaps in a Single Parameter Setting.** Plotting the thresholded adaptivity gap over values of ε and number of samples N, such that $p_1 = p_2 = \varepsilon$ and $p_3 = 1 - 2\varepsilon$. In this plot, we fix k to $\lfloor 0.8N \rfloor$.

3.5 Towards Optimality in Practice

The learning setting for which we demonstrate theoretical gaps in Section 3.4 does not account for many of the properties that make realistic learning tasks difficult, like complex data distributions and label noise. While designing *optimal* unlearning requesters in a realistic learning setting is likely intractable, there are a variety of heuristic methods that can be used to outperform random requesters. In this section, we adopt methods from the data valuation literature to design requesters that cause significant degradation in image classification models. The requesters we design help explain how and when requesters will outperform

random deletion requests, as well as help clarify the impacts of adaptivity and collectivity.

3.5.1 Data Valuation-Based Requesters

In an attempt to design realistic requesters that approach worst-case performance, we look towards the data valuation literature. Data valuation attempts to quantify the importance of each training sample for a model's performance on a given task by assigning each sample a value. We specifically consider the data Shapley value, a valuation method that has seen varied applications in machine learning in recent years (Ghorbani and Zou [2019]).

Definition 13 (Data Shapley Value)**.** *Let* $D = \{(x_j, y_j)\}_{j\in[n]}$ *be a dataset and* $u : (\mathcal{X} \times \mathcal{Y})^* \to \mathbb{R}$ *be an arbitrary utility function mapping datasets to scores. The data Shapley value for the* j^{th} *point in* D *is defined as*

$$\phi_j(D, u) = \frac{1}{n} \cdot \sum_{S\subseteq D\setminus\{(x_j,y_j)\}} \frac{u(S \cup \{(x_j, y_j)\}) - u(S)}{\binom{n-1}{|S|}}.$$

When it is clear from context, we will use ϕ_j to refer to $\phi_j(D, u)$. Much like a performance metric g used to compute requester utility, which can represent arbitrary functions of a model, the function u can represent arbitrary functions of a dataset. To translate the Shapley value into our unlearning framework, we define the nonadaptive Shapley requester.

Definition 14 (Shapley Requester)**.** *For each index* $i \in [n]$*, the nonadaptive Shapley requester* `Shap` *computes* $\phi_i(D, u)$ *and returns the samples in* D *ordered by decreasing values of* ϕ_i*, with ties being broken randomly.*

As we have defined it, the Shapley requester is *nonadaptive* and *collective*; the ordering of deletions is determined based only on the initial dataset, and computing a Shapley value with respect to a dataset D requires full knowledge of D. We make minor modifications to this requester in order to isolate the effects of adaptivity and collectivity on requester performance. First, we define the *adaptive Shapley requester*:

Definition 15 (Adaptive Shapley Requester)**.** *The order of deletions requested by the adaptive Shapley requester $u_1, \ldots, u_n$ is captured by the following recurrence relation:*

$$u_1 = \arg\max_{j \in [n]} \phi_j(D, u)$$

$$u_t = \arg\max_{j \in [n+1-t]} \phi_j(D^t, u),$$

where $D^t = D \setminus \{x_1, \ldots, x_{t-1}\}$.

The adaptive Shapley requester recomputes the Shapley values after each deletion with respect to all of the remaining points in the training dataset. Next, we define the *individual Shapley requester*:

Definition 16 (Individual Shapley Requester)**.** *Let $D' \in (\mathcal{X} \times \mathcal{Y})^n$ be a dataset drawn from the same distribution as D. For each index $j \in [n]$, the individual Shapley requester computes $\phi_j(D' \cup \{(x_j, y_j)\}, u)$ and returns the samples in D ordered by decreasing values of $\phi_i(D' \cup \{(x_j, y_j)\}, u)$.*

The individual Shapley requester operates similar to the original Shapley requester, except it computes Shapley values with respect to a dataset sampled independently from D, but from the same distribution. The individual Shapley requester is inherently *nonadaptive*: the computation of $\phi_j(D', \cup\{x_j, y_j\}, u)$ depends neither on prior unlearning requests nor the state of the model. This formulation is similar in principle to the distributional Shapley value introduced by Ghorbani et al. [2020].

Definition 17 (Distributional Shapley Value [Ghorbani et al., 2020])**.** *Given a data distribution $\mathcal{D}$ supported on $\mathcal{X} \times \mathcal{Y}$, a utility function $u : (\mathcal{X} \times \mathcal{Y})^* \to \mathbb{R}$, and some $m \in \mathbb{N}$, the distributional Shapley value for a point $(x, y) \in \mathcal{X} \times \mathcal{Y}$ is the expected data Shapley value*

over data sets of size m containing i:

$$\nu((x,y);u,\mathcal{D},m) = \mathop{\mathbb{E}}_{B\sim\mathcal{D}^{m-1}} \phi_{(x,y)}(B\cup\{(x,y)\},u).$$

Since the individual Shapley requester only uses a single sample from the distribution to determine its ordering, it is implicitly computing an unbiased (but high variance) estimator of the distributional Shapley value. We made this design decision for a number of reasons. For one, the ordering computed by the individual Shapley requester needs to be independent of the training dataset D. In our experiments we work with the CIFAR-10 dataset, which is of modest size (50,000 training examples). In order to sample datasets from the same distribution, the requester must train on a subset of CIFAR-10 and compute the ordering using subsets of the same size. If we train on a large subset of CIFAR-10, there will be significant overlap between the training sets and the subsets used for Shapley computation, increasing the dependence between the distributional Shapley computation and the true Shapley value with regards to D. If we train on a small subset of CIFAR-10, there will be less overlap but the performance of the trained classifiers will be poor. Training on half of the training set and computing the deletion ordering with respect to the held out set allows us to ensure that the requester is truly individual.

In this work, we will generally take $u(S)$ to be the test accuracy of a model trained from scratch on S. With this choice of u, the goal of the Shapley requester and its variants can be thought of as maximizing with each request the expected decrease in test accuracy after that sample is deleted. Observe how this corresponds to the requester's utility when the performance metric g is chosen to be test error. Ordering deletion requests by decreasing Shapley value can thus be seen as a greedy heuristic for assembling a set of training samples which minimizes test accuracy after being unlearned.

We note, however, that the optimization problem that the Shapley requester is solving is distinct from that of an optimal unlearning requesters. It is straightforward to show that

the Shapley requester is sometimes suboptimal.

Claim 2. *The nonadaptive Shapley requester is not an optimal deletion requester in the learning setting defined in Section 3.4.*

Proof. We will demonstrate a realized dataset in which the Shapley requester does not behave optimally. Let $\mathcal{X} = \{a, b\}$, and let $p_a = \frac{1}{3}$ and $p_b = \frac{2}{3}$. Suppose that a dataset D^{ref} is sampled such that it contains 1 item from subpopulation a and 2 items from subpopulation b. Furthermore, let $k = 1$.

In this setting, the test accuracy of a model trained on S using learning algorithm $\mathcal{A}^*$ is

$$u(S) = \sum_{i \in \mathcal{X}} p_i \cdot \mathbb{1}[i \in S] + \frac{p_i}{2} \cdot \mathbb{1}[i \notin S].$$

It is clear from this expression that $u(S \cup \{(i, f(i))\}) > u(S)$ if and only if $(i, f(i)) \notin S$. In fact, if $(i, f(i)) \notin S$, then $u(S \cup \{(i, f(i))\}) - u(S) = \frac{p_i}{2}$. We can therefore simplify the Shapley value in this scenario:

$$\begin{aligned}
\phi_i &= \frac{1}{n} \cdot \sum_{S \subseteq D^{\mathsf{ref}}_{-i}} \frac{u(S \cup \{(x_i, y_i)\}) - u(S)}{\binom{n-1}{|S|}} \\
&= \frac{1}{n} \cdot \sum_{S \subseteq D^{\mathsf{ref}}_{-i}} \frac{p_{x_i}/2 \cdot \mathbb{1}[(x_i, y_i) \notin S]}{\binom{n-1}{|S|}}.
\end{aligned}$$

Let j_a denote the index in D^{ref} of the item from subpopulation a. None of the subsets

of $D^{\mathsf{ref}}_{-j_a}$ contain subpopulation a. Therefore, the Shapley value for this item is

$$\begin{aligned}
\phi_{j_a} &= \frac{1}{n} \cdot \sum_{S \subseteq D^{\mathsf{ref}}_{-j_a}} \frac{p_a/2 \cdot \mathbb{1}[(x_{j_a}, y_{j_a}) \notin S]}{\binom{n-1}{|S|}} \\
&= \frac{1}{3} \cdot \sum_{S \subseteq D^{\mathsf{ref}}_{-j_a}} \frac{p_a/2}{\binom{2}{|S|}} \\
&= \frac{p_a}{6} \left(\frac{1}{\binom{2}{|\{\}|}} + \frac{1}{\binom{2}{|\{b\}|}} + \frac{1}{\binom{2}{|\{b\}|}} + \frac{1}{\binom{2}{|\{b,b\}|}} \right) \\
&= \frac{p_a}{6} \left(\frac{1}{1} + \frac{1}{2} + \frac{1}{2} + \frac{1}{1} \right) \\
&= \frac{p_a}{2} = \frac{1}{6}.
\end{aligned}$$

Let j_b be the index in D^{ref} of one of the items representing subpopulation b. The subsets of $D^{\mathsf{ref}}_{-j_b}$ that don't contain subpopulation b are $\{\}$ and $\{a\}$. Therefore, the Shapley value for an element of subpopulation b is

$$\begin{aligned}
\phi_{j_b} &= \frac{1}{n} \cdot \sum_{S \subseteq D^{\mathsf{ref}}_{-j_b}} \frac{p_b/2 \cdot \mathbb{1}[(x_{j_b}, y_{j_b}) \notin S]}{\binom{n-1}{|S|}} \\
&= \frac{1}{3} \cdot \sum_{S \in \{\{\},\{a\}\}} \frac{p_b/2}{\binom{2}{|S|}} \\
&= \frac{p_b}{6} \cdot \left(\frac{1}{\binom{2}{|\{\}|}} + \frac{1}{\binom{2}{|\{a\}|}} \right) \\
&= \frac{p_b}{6} \cdot \left(\frac{1}{1} + \frac{1}{2} \right) \\
&= \frac{p_b}{4} = \frac{1}{6}.
\end{aligned}$$

Therefore, $\phi_{j_a} = \phi_{j_b}$. The Shapley requester, which breaks ties randomly, will request

the deletion of a random sample from D^{ref} and expect to achieve a utility of

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathtt{Shap}} \mathtt{util}_{\mathtt{Shap},\mathtt{err}}(\theta^0, D^{\mathsf{ref}}, 1) &= \mathop{\mathbb{E}}_{\mathtt{Shap}} \mathtt{err}(\theta^1) - \mathtt{err}(\theta^0) \\
&= \mathop{\mathbb{E}}_{\mathtt{Shap}} \mathtt{err}(\theta_1) \\
&= \Pr[u^1 = a] \cdot \frac{p_a}{2} + \Pr[u^1 = b] \cdot 0 \\
&= \frac{p_a}{6} = \frac{1}{18}.
\end{aligned}$$

Now, let $\mathtt{Shap}'$ be a requester that behaves identically to $\mathtt{Shap}$ unless $D = D^{\mathsf{ref}}$, in which case it will request deletions in the optimal order (subpopulation a, then subpopulation b). On dataset D^{ref}, $\mathtt{Shap}'$ would deterministically achieve a utility of

$$\begin{aligned}
\mathtt{util}_{\mathtt{Shap}',\mathtt{err}}(\theta^0, D^{\mathsf{ref}}, 1) &= \mathtt{err}(\theta_1) - \mathtt{err}(\theta^0) \\
&= \frac{p_a}{2} = \frac{1}{6}.
\end{aligned}$$

This implies that

$$\mathop{\mathbb{E}}_{D\sim\mathcal{D}}[\mathtt{util}_{\mathtt{Shap},\mathtt{err}}(\theta, D, k) \mid D \neq D^{\mathsf{ref}}] = \mathop{\mathbb{E}}_{D\sim\mathcal{D}}[\mathtt{util}_{\mathtt{Shap}',\mathtt{err}}(\theta, D, k) \mid D \neq D^{\mathsf{ref}}]$$

and

$$\mathop{\mathbb{E}}_{D\sim\mathcal{D}}[\mathtt{util}_{\mathtt{Shap},\mathtt{err}}(\theta, D, k) \mid D = D^{\mathsf{ref}}] < \mathop{\mathbb{E}}_{D\sim\mathcal{D}}[\mathtt{util}_{\mathtt{Shap}',\mathtt{err}}(\theta, D, k) \mid D = D^{\mathsf{ref}}],$$

Therefore, $\overline{\mathtt{util}}_{\mathtt{Shap},\mathtt{err}}(\mathcal{D}, k) < \overline{\mathtt{util}}_{\mathtt{Shap}',\mathtt{err}}(\mathcal{D}, k)$ so $\mathtt{Shap}$ is subpotimal. □

Despite this suboptimality, there is reason to believe that the Shapley requester is an effective unlearning requester in practice. In the data valuation literature, techniques are often evaluated by removing points from a dataset ordered by valuation and retraining

models on the progressively smaller datasets (see Ghorbani and Zou [2019, Figure 2], Yan and Procaccia [2021, Figure 1]). These evaluations show that ordering deletions according to valuation metrics like the Shapley value can resulting a more rapid degradation in model performance than random orderings. This indicates a positive correlation between a data point's value and its contribution to unlearning utility.

3.5.2 Other Deletion Requesters

The design of these Shapley-based requesters is meant to reflect how a hypothetical individual might act after learning that their data was used to train an AI model. In this model of human behavior, an individual whose data is deemed valuable to the model trainer might feel that they should have been compensated for their data, and therefore they would be more likely to request deletion. On the other hand, an individual whose data was not deemed valuable may not feel that they were harmed and therefore be less likely to request deletion. This, of course, is an overly simplistic behavioral model which does not account for many of the myriad ways in which people think about the use of their data. However, our framework for studying requesters allows for flexibility in the types of behaviors that we are modeling. In our experiments, we work with two additional requesters which are intended to model different ways individuals might prioritize deletion.

The Loss Requester: An individual may simply want a model to not regurgitate correct information about them. In this case, the individual would be more likely to request deletion if the model confidently classifies their data correctly, indicated by a low cross-entropy loss. Therefore, the *loss requester* issues deletion requests ordered by decreasing cross-entropy loss (which we denote by $\ell(\cdot, \cdot)$).

Definition 18 (Loss Requester)**.** *The* nonadaptive loss requester *computes the loss* $\ell(\theta^0(x), y)$ *for each point* $(x, y) \in D$ *and returns the samples in* D *ordered by increasing values of loss.*

The adaptive loss requester *issues deletions requests* $u_1, \ldots, u_n$ *such that*

$$u_1 = \arg\max_{(x,y)\in D} \ell(\theta^0(x), y)$$

$$u_j = \arg\max_{(x,y)\in D^j} \ell(\theta^{j-1}(x), y),$$

where $D^j = D \setminus \{x_1, \ldots, x_{j-1}\}$.

The MIA Requester: An individual may be motivated by privacy concerns presented by a model more broadly. Rather than their data being regurgitated, this individual is worried that their data has any sort of measurable influence on the model in such a way that might be detected by an attacker. This is the type of privacy leakage that is typically measured using membership inference attacks (MIAs). Therefore, the *MIA requester* runs a membership inference attack on each training sample and issues requests by decreasing MIA confidence.

Definition 19 (MIA Requester)**.** *Let* $\texttt{MIA} : \Theta \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}$ *be a* membership inference attack *mapping a model and a data sample to a real number. The* nonadaptive MIA requester *computes the value* $\texttt{MIA}(\theta^0, (x, y))$ *for each point* $(x, y) \in D$ *and returns the samples in* D *ordered by increasing values of MIA values.*

The adaptive MIA requester *issues deletions requests* $u_1, \ldots, u_n$ *such that*

$$u_1 = \arg\max_{(x,y)\in D} \texttt{MIA}(\theta^0, (x, y))$$

$$u_j = \arg\max_{(x,y)\in D^j} \texttt{MIA}(\theta^{j-1}, (x, y)),$$

where $D^j = D \setminus \{x_1, \ldots, x_{j-1}\}$.

In our experiments, we adopt LiRA [Carlini et al., 2022a] as a membership inference attack for this requester.

The Random Requester: As a baseline, we implement a requester which issues deletion requests in a random order. This reflects the evaluation methods used for many proposed machine unlearning methods, which sample forget sets uniformly at random from a model's training set. This requester thus reflects the most common approach for evaluating unlearning algorithms and will serve as the baseline in our exploration of requester utility.

Definition 20 (Random Requester)**.** *The random requester* `Rand` *returns the simplaes in* D *in the order* $u_1, \ldots, u_n$*, where* $u_1, \ldots, u_n$ *is a uniformly sampled permutation of* $[n]$*.*

We note that all three of these requesters are *individual*. Neither computing the cross-entropy loss for a datapoint nor computing a LiRA score require knowledge of the particular samples in the training dataset beyond the point being tested. Additionally, we can easily modify either of these requesters to be adaptive by recomputing loss/LiRA scores after each deletion.

3.5.3 Experimental Setup

Evaluation methods: We borrow a technique from the data valuation literature to evaluate the performance of different requesters. As mentioned in Section 3.5.1, a common evaluation technique for data valuation methods is to sort points in the training dataset in decreasing order by valuation and, for some subset of $k \leq |D|$, computing the performance of a model trained on the $|D| - k$ least valuable datapoints. These are plotted as series for different valuation metrics to allow for comparisons between metrics [Ghorbani and Zou, 2019, Ghorbani et al., 2020, Yan and Procaccia, 2021].

These plots can be easily adapted to evaluate unlearning requesters. Instead of sorting by valuation according to a metric, we sort training samples according to the order in which a requester requests their deletion. This applies to both nonadaptive requesters, where the request order is fixed at the beginning of the interaction, and adaptive requesters, which recompute orderings in response to each model update. The choice of unlearning algorithm

can also be varied. In the valuation literature, retraining from scratch on the $|D| - k$ least valuable points is evaluated, while we can alter this to use any algorithm to unlearn the first k points that are requested. We call these visualizations *unlearning utility plots*. In our experiments, we typically run 10 rounds of unlearning, requesting points in batches of 1,000. We report average accuracies over 4 trials and give error bars representing one standard deviation above and below the mean.

Datasets and Architectures. All of our experiments make use of the ResNet18 architecture trained on the CIFAR-10 dataset. Rather than train on the full set of 50,000 training samples, we train our initial models on a randomly sampled 50% subset of CIFAR-10's training set. We use the held-out training samples for the computation of LiRA scores in the MIA requester and the computation of Shapley values in the individual Shapley Requester.

Requesters. As described above, we run experiments on the Shapley requester, the loss requester, the MIA requester, and the random requester. Additionallly, we run experiments on the adaptive and individual variants of the Shapley requester, as well as the adaptive variants of the loss and MIA requesters.

Since it is computationally infeasible to compute exact Shapley values for a dataset as large as CIFAR-10, we make use of a method for computing approximate Shapley values. In particular, we implement a k-nearest neighbors based approach for computing Shapley values first proposed by Jia et al. [2019]. This involves training a KNN model on the activations induced by each training sample at the final fully connected layer of the network (i.e. the input to the final softmax layer). We then compute exact Shapley values for the KNN model, which can be computed efficiently using dynamic programming. In our experiments, we set $k = 10$, which we found yielded KNN classifiers with similar performance to the ResNet18 networks being approximated.

For computing LiRA scores, we adopt the methodology presented by Carlini et al. [2022a].

We train 500 shadow models, each of which is trained on a random subset of 25,000 CIFAR-10 training samples.

Unlearning Methods. We make use of three unlearning methods which have been proposed in the literature: retraining, relabeling, and SCRUB. retraining is an exact unlearning method, while relabeling and scrub are approximate unlearning methods. We do not claim that these algorithms necessarily represent the state of the art as it currently stands. Instead, we use them as examples to demonstrate how the utility of an unlearning algorithm can change depending on the choice of unlearning algorithm

Retraining serves as the "gold standard" baseline for machine unlearning and simply involves training a new model from scratch on the set of points being retained.

Relabeling [Golatkar et al., 2020] fine-tunes the model on the original training set, except the set of points being forgotten are relabeled with random labels.

SCRUB [Kurmanji et al., 2023] uses a student-teacher knowledge distillation approach. The unlearned model (student) is trained in such a way that it behaves similarly to the original model (teacher) but is uncorrelated with the original model on the forget set.

3.5.4 Results

Comparing Deletion Requesters

We begin by analyzing the nonadaptive variants of the deletion requesters described above (random, loss, MIA, and Shapley). An deletion performance plot is shown in Figure 3.3, using retraining as the unlearning method. We observe several trends. Perhaps unsurprisingly, all requesters degrade the performance of the model. After the removal of 9,000 data points (36% of the original training dataset), all methods on average reduce test accuracy to below 84%, a reduction of over 4 percentage points below the original model. While all of the requesters are relatively close in performance, the Shapley performs the best in the end,

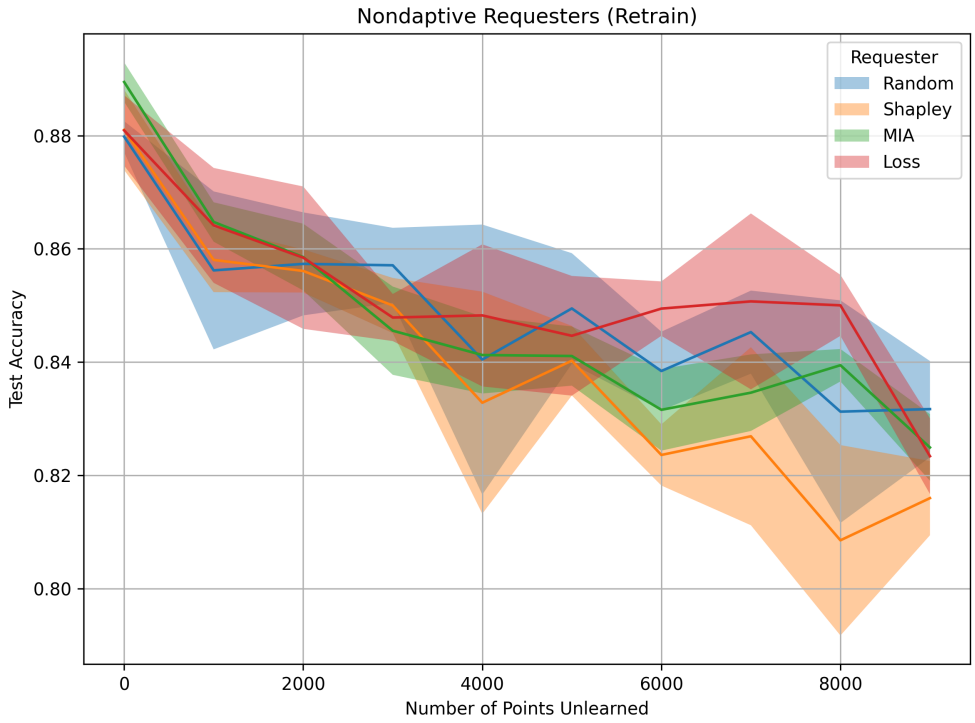


Figure 3.3: **Nonadaptive Requester Unlearning Performance.** We plot the performance of the loss, MIA, and Shapley requesters compared to the random requester. For each requester, we train an initial model for 50 epochs on a random subset of 25,000 CIFAR-10 training points and use that model to compute an ordering in which the training sampled will be unlearned. Points are then unlearned in this order over 9 iterations of 1,000 points per iteration. Solid lines represent averages over 4 trials, and one standard deviation is filled above and below the mean.

achieving a final test accuracy of 81.6%. For comparison, the random requester achieves a final test accuracy of 83.2%. This is directionally in line with results from the data valuation literature Ghorbani and Zou [2019], although those results would suggest a larger difference between the Shapley and random requesters. We believe this discrepancy is due to our use of an approximation of the Shapley value (as described in 3.5.3).

Disentangling Adaptivity and Collectivity

We now measure the performance of the adaptive variants of the requesters we analyzed above. Results are presented in Figure 3.4. We note that the random requester is inherently

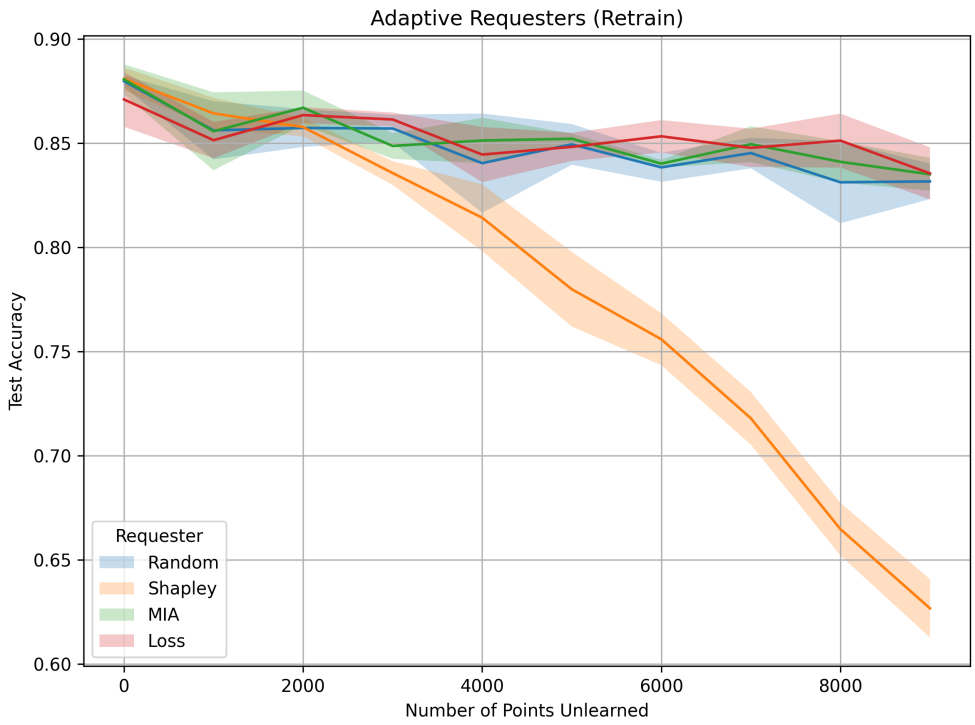


Figure 3.4: **Adaptive Requester Unlearning Performance.** We plot the performance of the adaptive variants of the loss, MIA, and Shapley requesters compared to the random requester.

nonadaptive and included in this figure as a frame of reference. We observe that ehat the adaptive loss and MIA requesters perform nearly identically to their nonadaptive variants. However, the adaptive Shapley requester is much more powerful than its nonadaptive variant, reducing test accuracy to 62.7% on average. This shows us that adaptive requesters are not inherently more powerful than nonadaptive requesters. We hypothesize that the Shapley requester benefits from adaptivity because it attempts to rank points by how important they are to the accuracy of the model, which is exactly the metric that we are measuring here.

Additionally, we plot the results for the variants of the Shapley requester in Figure 3.5. Unsurprisingly, the individual Shapley requester is less powerful than both the adaptive and nonadaptive variants of the (collective) Shapley requester. Oddly, the individual Shapley requester also has less utility than the random requester. This is likely due to the high-

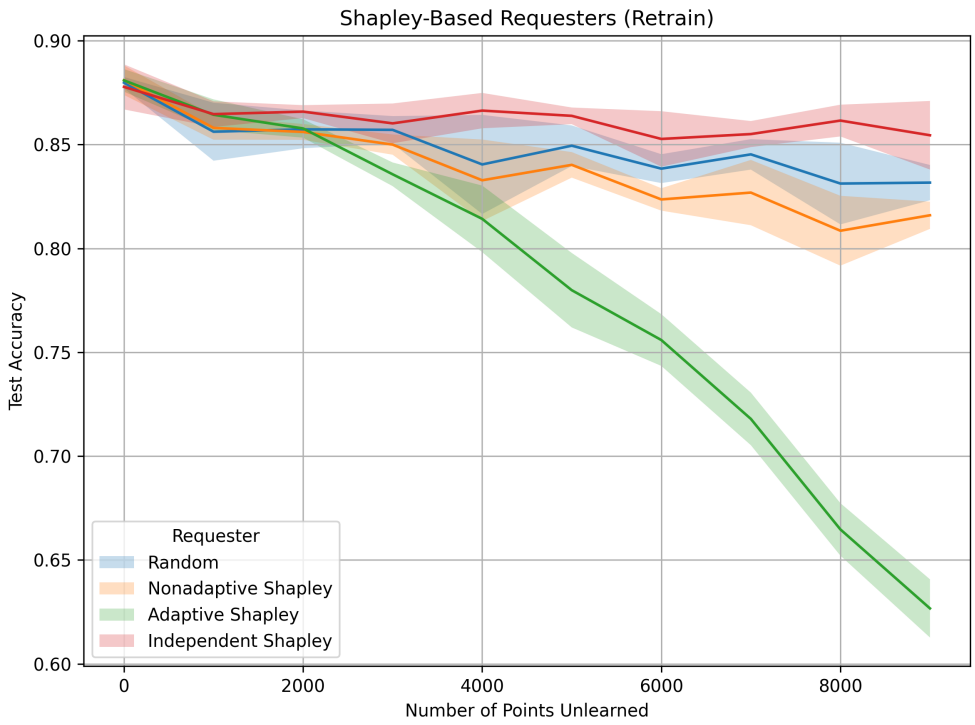


Figure 3.5: **Shapley-Based Requester Unlearning Performance.** We plot the performance of variants of the Shapley requester (nonadaptive, adaptive, and individual) compared to the random requester.

variance estimator of the distributional Shapley value that we utilize. Results from Ghorbani et al. [2020] suggest that the individual Shapley requester should be able to outperform the random requester in at least some settings.

Comparing Unlearning Algorithms

We recreate the plots in Figures 3.3 and 3.4 (which used Retrain as an unlearning method) using both Relabel (Figure 3.6) and SCRUB (Figure 3.7). One thing we observe is that, for all requesters interacting with either of these approximate methods, the accuracy of the model increases after the first round of unlearning. We believe this is because both of these methods involve fine-tuning the weights of the original model, resulting in a model that has been trained for longer, and only after the first step have enough points been removed to for

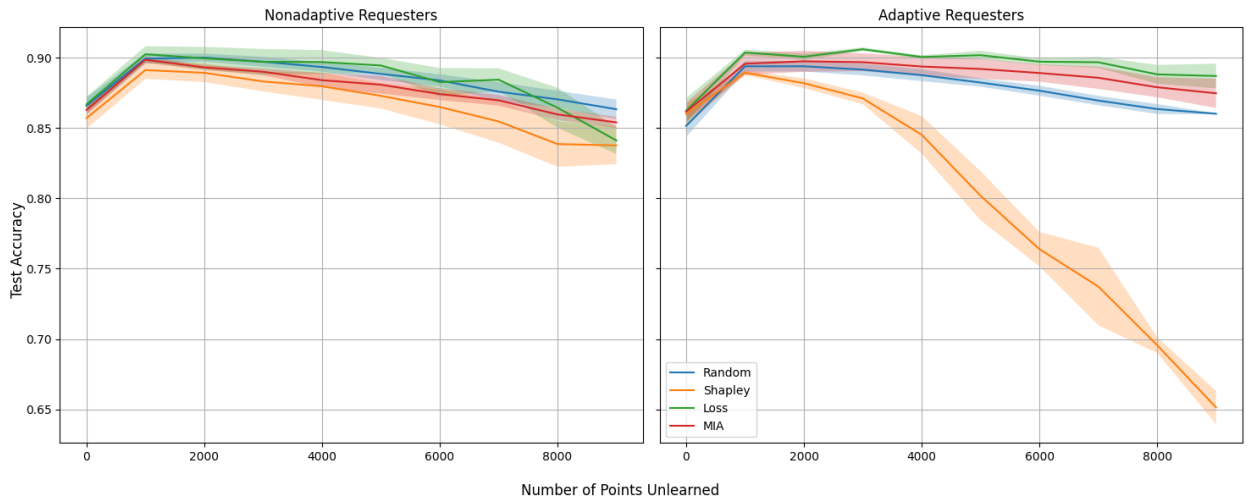


Figure 3.6: **Relabel Unlearning Utility.** We plot the performance of nonadaptive and adaptive unlearning requesters compared to the random requester when the unlearning algorithm is Relabel.

accuracy to start decreasing again.

Looking at the results for Relabel, patterns look broadly similar to those of Retrain. In particular, the nonadaptive Shapley requester shows slightly higher utility than the other requesters, and the adaptive Shapley requester displays significantly higher utility. Interestingly, the adaptive Loss and MIA requesters appear to have lower utility than the random requester, a pattern that is not clearly visible in their nonadaptive variants or when using Retrain.

SCRUB yields significantly different results. After the initial rise in accuracy, the accuracy of the unlearned models tends to decrease only marginally or not at all. In fact, the nonadaptive Shapley requester does not appear to reduce the accuracy of the model after 9,000 deletions. We do observe an gap between the adaptive and nonadaptive Shapley requester after $\tilde{3}$,000 points are unlearned, but the effects are much more muted than with Retrain or Relabel.

These plots illustrate how the utility of a requester is dependent on the unlearning method being used. Our results indicate that some approximate unlearning algorithms (like SCRUB)

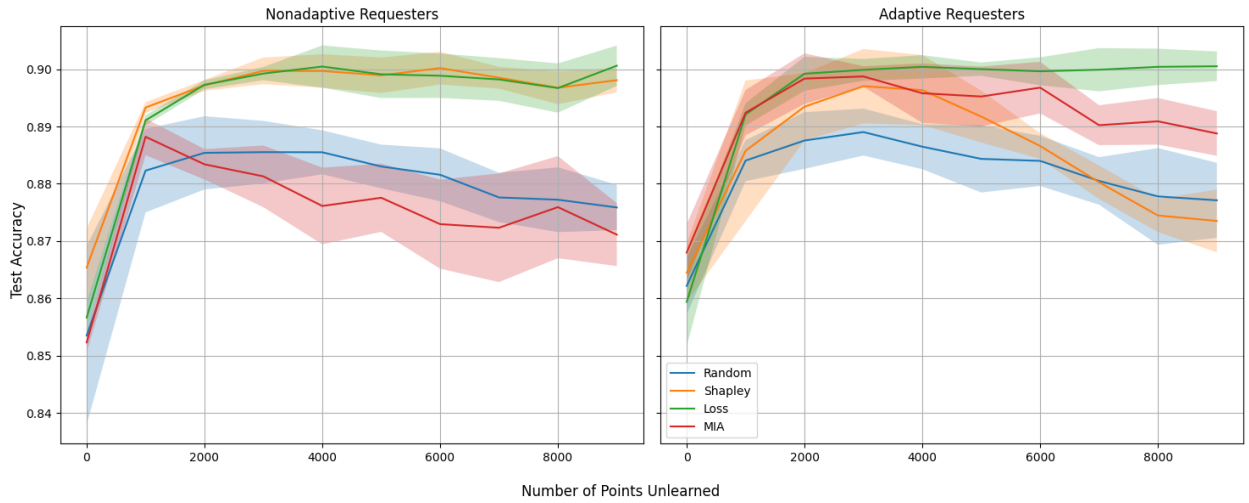


Figure 3.7: **SCRUB Unlearning Utility.** We plot the performance of nonadaptive and adaptive unlearning requesters compared to the random requester when the unlearning algorithm is SCRUB.

may be more resistant to the effects of requester adaptivity than others (like Relabel). More work is necessary to better understand this phenomenon.

3.6 Limitations and Future Work

One significant limitation of our framework is that we don't account for mitigation strategies that can be used by an unlearning algorithm to lessen the impacts of deletion requesters. For example, the algorithm could attempt to sample new data from the same distribution as the deleted data that can be used for fine-tuning the model after unlearning. It is not immediately clear from our results whether the effects of request distribution would be difficult to account for. One approach might take inspiration from concepts in performative prediction [Perdomo et al., 2020]. For example, one might attempt to train a model that induces distributions of requests from certain requesters that are close to random.

Another limitation is the binary nature of our definitions of adaptivity and collectivity. More granular definitions might allow us to better understand how different properties of adaptive and collective requesters emerge. Taking inspiration from ideas in data governance,

one might ask how large a data cooperative would have to be in order to influence models more than individuals could on their own. Recent theoretical work on semi-adaptivity gaps in the stochastic knapsack problem Barak and Talgam-Cohen [2026] might be useful in analysis of these requesters.

CHAPTER 4

REDISTRICTING AND DIFFERENTIAL PRIVACY

4.1 Introduction

In 2021, the state of Alabama sued the US Department of Commerce. Alabama sought to enjoin the use of a new disclosure avoidance system for the forthcoming release of 2020 decennial census data.

> "Forcing the State to redistrict with intentionally flawed data will impede Alabama's ability to draw representative districts with near-equal populations, which is what the Constitution and one-person, one-vote jurisprudence require. This will also impede Alabama's ability to draw districts to protect minority voting rights as required by the Voting Rights Act." (*Alabama v. Department of Commerce*, 2021)

Concerns persisted even after the data were published. In early 2022, one of us was contacted by lawyers at the ACLU in connection with racial gerrymandering cases in Georgia and Arkansas. They wondered whether the new disclosure avoidance system could affect the reported number of majority-Black districts in a demonstration redistricting plan.[1]

Every ten years, the US Census Bureau conducts a decennial census and produces the *Redistricting Data Summary Files*—the basis for all redistricting for the following decade. These Redistricting Data are created by first aggregating data from the *Census Edited File* (CEF)—the confidential dataset that reflects the results of the completed census—and then applying the *Disclosure Avoidance System* (DAS). Through 2010, the DAS comprised traditional disclosure limitation techniques, primarily swapping [McKenna, 2018]. These techniques are now understood as lacking formal guarantees against re-identification and individualized data disclosures [Garfinkel et al., 2018, Dick et al., 2023].

1. Ultimately, this issue was not raised by defendants. No expert testimony was needed.

In contrast, the 2020 DAS is based on differential privacy, a modern framework for formally quantifying and limiting such disclosure risks [Dwork et al., 2006, Abowd et al., 2022]. The 2020 DAS synthesizes and aggregates a new set of microdata census responses that have no one-to-one correspondence with the actual microdata in the CEF but closely approximate many of its statistics. While previous DASes have also introduced error into census tabulations, the 2020 DAS is the first to perturb total population counts in every geography smaller than states. This is necessary for the formal guarantees sought by the Census Bureau. Relative to the population, these errors can be quite large in very small geographies (24.79% in urban census blocks) but are much smaller even in moderately-sized geographies (0.31% for counties of less than 1,000 people).

Many viewed the 2020 DAS as undermining the legitimacy of the data's use for redistricting and called for a rethinking of disclosure avoidance in the census [Kenny et al., 2021, Schneider, 2021, Wezerek and Van Riper, 2020, Banks, 2025]. Others disagreed [Cohen et al., 2022, Jarmin et al., 2023]. The argument against the new DAS was made most forcefully by Kenny et al. [2021]. They analyze the impact of the 2020 DAS using an ensemble-based redistricting analysis: a modern, computationally-intensive approach to exploring properties of the space of reasonable redistricting plans in a given geography. They conclude that "nonrandom local errors can aggregate into substantively large and unpredictable biases at district levels" and claim "the added noise makes it impossible to follow the principle of One Person, One Vote, as it is currently interpreted by courts and policy-makers" [Kenny et al., 2021].

The concern is easy to understand: two districts whose populations are equal according to the official 2020 Redistricting Data will very likely have different populations according to the confidential 2020 CEF. If such discrepancies are large or biased, they could impact electoral power.

Sharp legal thresholds can amplify discrepancies, turning a continuous source of error into

a discrete category change. For example, 10% population deviation is the cutoff between a state legislative plan being presumed discriminatory or not. Prior work found that about 5% of plans generated for the Louisiana state senate using the 2020 Redistricting Data would have exceeded that population deviation threshold according to the 2020 CEF Kenny et al. [2021].

This is what Alabama and Kenny et al. [2021] are worried about: that plans created using the Redistricting Data could turn out to be illegal—potentially "impeding" or even "inhibiting" the ability to "draw fair lines," as Alabama put it. (We question the underlying legal theory, as discussed in Section 4.9.) Despite prior work quantifying the magnitude and characteristics of the perturbations, little is known about the interaction with threshold redistricting requirements.

4.1.1 Research Questions

The goal of this paper is to better understand how perturbations from disclosure avoidance combine with sharp legal thresholds to impact redistricting. What sorts of discrepancies arise and why? Can they be mitigated?

To that end, we consider a conscientious redistricter who wants to minimize the impact of disclosure avoidance on their map, even if not legally required. Our redistricter only has access to the officially published 2020 Redistricting Data (after disclosure avoidance), but wants their maps to comply with threshold redistricting requirements as measured using the confidential CEF data (before disclosure avoidance).[2]

We ask whether there exists methods by which the redistricter is able to achieve two goals, measured against the confidential CEF. First, to produce plans balanced within *One Person, One Vote* (OPOV) population tolerances. Second, to produce plans with as many *majority-minority districts* (MMDs) as possible.

2. As explained in Section 4.3.1, we do not have access to CEF and must use a different dataset SWAP as a stand in. See Section 4.8 for a discussion of how this does and does not affect our results.

We study both questions at the state legislative level, where the legal standing of such plans depend on sharp thresholds (see Section 4.2). For OPOV, population deviations less than 10% are presumptively constitutional; larger deviations are presumptively unconstitutional. And showing that a political geography admits some number of majority-minority districts—districts with at least 50% of the population in the relevant minority group—is necessary to successfully overturn racially gerrymandered plans. We focus our examination of MMDs on majority-Black districts, reflecting a common scenario in voting rights litigation and the largest racial minority in the USA.

4.1.2 Overview of Methods

This paper employs *ensemble analysis*. We algorithmically sample large collections of possible districting plans and observe how statistics of our ensembles differ between our two data sources (see Section 4.3 for details).

To study the effect of the DAS, one would ideally compare the data with and without disclosure avoidance, namely the 2020 CEF and 2020 Redistricting Data. However, the CEF is confidential to those outside of the Census Bureau. We instead use two Census datasets which we call SWAP and DEMO as stand-ins for the 2020 CEF and Redistricting Data, respectively. We are using these datasets exactly as intended: The Census Bureau created DEMO to help stakeholders study the impacts of the 2020 DAS by comparing it to SWAP. See Section 4.3 for more information and Section 4.8 for further discussion on the limitations of using these datasets.

Our high-level approach is simple, and is adapted from Kenny et al. [2021]. First, we generate an ensemble of many redistricting plans. These plans are drawn to satisfy some constraint—or to maximize some objective—on the DEMO dataset (standing in for 2020 Redistricting Data). Second, across plans in the ensemble, we see how often the constraint is violated—or how the value of the objective compares—on the SWAP dataset (standing in

for 2020 CEF).

We call such disagreements in measurements between DEMO and SWAP *discrepancies*. We quantify discrepancies in two important quantities of redistricting plans: the population-deviation, and the number of majority-minority districts.

4.1.3 Contributions and Main Findings

We investigate the discrepancies that occur when maps created using data perturbed by the 2020 disclosure avoidance system are measured against sharp threshold requirements using the unseen, unperturbed dataset. Overall, we find that thresholding can amplify the impact of disclosure avoidance noise, in ways that are well-captured by simple models and possible to account for. In particular, there exists a method to satisfy population balance requirements on SWAP using only the DEMO data with confidence. We also provide simple probabilistic models that help explain our empirical observations.

Below is a summary of our contributions and some of our main findings. Our quantitative findings are specific to our experimental design (e.g., state legislative districts, using precincts as building blocks); drawing quantitative conclusions for other redistricting settings (e.g., smaller sub-state districts, incorporating different constraints or objectives) would require adapting our techniques. We discuss these and other important limitations in Section 4.8.

For One Person, One Vote (Section 4.5) We extend the ensemble analysis in prior work [Kenny et al., 2021] from one state legislative geography to 93. As in the prior work, we observe significant discrepancies. For example, among plans sampled with at most 5% population deviation on DEMO, the *discrepancy rate*—the fraction of plans exceeding 5% deviation on SWAP—was at least 40% in half of the legislative geographies we examined.[3]

3. 5% deviation as measured using the method of Kenny et al. [2021] and others, which approximates the 10% deviation threshold as measured by courts (see Section 4.3.3).

At the same time, there exists a method to draw districts that can substantially reduce or eliminate observed threshold violations within our district-generation procedure. The countermeasure is simple: use a population deviation threshold in the redistricting sampling algorithm that is slightly tighter than the 5% policy target (i.e., applying an offset). Sampling with a 4.5% limit yields a discrepancy rate of 0% for over half of geographies examined. Sampling with a 4% limit does so for 90% geographies examined.

For majority-minority districts (Section 4.6) We extend the ensemble analysis in prior work [Kenny et al., 2021] from one state legislative geography to 52. We observe significant discrepancies, but in the opposite direction than prior work (see Section 4.4). For example, among Georgia state house plans sampled using DEMO, 9% have a different number of majority-Black districts on SWAP—almost always fewer. When sampling plans using a method that maximizes the number of majority-Black districts (as is common in Voting Rights Act litigation), this jumps to 66%.

Simple models for understanding discrepancies In both cases, we give simple probabilistic models that help explain many of the observed phenomena, including the effectiveness of the mitigation proposed for achieving population balance and the impact of maximizing the number of majority-Black districts on the discrepancies that result. Our hope is that these models prove useful for thinking about the impact of disclosure avoidance noise in settings beyond our specific experimental design.

4.1.4 A Note on Relevance

In the spring of 2026, there were two major events the affected the relevance of our findings. In April, the Supreme Court's decision in *Louisiana v. Callais* significantly weakened Section 2 of the Voting Rights Act, making it more difficult for plaintiffs to argue that a map is an

unconstitutional racial gerrymander. In June, the US Department of Commerce issued an order prohibiting the use of differential privacy (along with any other disclosure avoidance technique that makes use of noise infusion) in statistical products released by the Census Bureau and the Bureau of Economic Analysis [US Department of Commerce, 2026]. The research this chapter is based on was completed and published before these events, when differential privacy was expected to be used in the 2030 census and the VRA could compel states to draw maps with more majority-minority districts. Given the current state of the legal landscape, it is unlikely that the results we present will be directly relevant to redistricters after the release of the 2030 Census. However, we still believe that the research framework we present for understanding and mitigating the effects of differential privacy can still be useful for practitioners seeking to better account for the effects of noise in their data analyses.

We present our research below in its original form, demonstrating how redistricters might have reckoned with differential privacy in 2030 had these legal changes not taken place. In Section 4.8, we address further how our results can be interpreted in the light of recent events.

4.2 Legal context

4.2.1 One Person, One Vote

The One-Person, One-Vote principle holds that any two people living in the same state should have about the same voting power. This means that the total population of different districts must be balanced according to the decennial census data (despite its imperfections, see Sec. 4.2.3).

For Congressional districts, "districts [must] be apportioned to achieve population equality as nearly as is practicable" (*Karcher v. Daggett*). No amount of population deviation is

considered de minimis. In practice, many states balance congressional districts to within a single person in total population. This is one reason that Congressional redistricting is not the focus of our study: there is no policy-relevant numerical target that a redistricter can hope to achieve after accounting for noise.

State-level legislative districts must also be balanced in total population, although the requirement is much less strict. For state legislative districts, population deviations below 10% "will ordinarily be considered de minimis," while "disparities larger than 10% creates a prima facie case of discrimination" (*Brown v. Thomson* 1983). In practice, state legislative population deviations are often much smaller than 10%.

Note that courts define population deviation as the relative difference between the largest and smallest districts. We instead adopt the convention of Kenny et al. [2021] and others, using the maximum relative difference from the ideal district population. A 10% deviation under the former definition is approximately equal to a 5% deviation under the definition used in this paper (see Sec. 4.3.3).

4.2.2 Voting Rights Act

Section 2 of the Voting Rights Act of 1965 outlaws vote dilution on the basis of race, including racial gerrymandering. While the Court's decision in *Louisiana v. Callais* (2026) will fundamentally change how racial gerrymandering cases are litigated, we out how the law was interpreted before 2026. The Supreme Court laid out a framework for judging racial gerrymandering claims in *Thornburg v. Gingles* (1986), and recently reaffirmed the framework in *Allen v. Milligan* (2023).

To successfully challenge an enacted map, a minority group needed to first pass a three-part threshold test: the Gingles preconditions. This chapter focuses on the first Gingles precondition, *Gingles 1.* (The second and third preconditions together require voting to be polarized to such an extent that the minority group is prevented from electing its candidates

of choice.)

Gingles 1 requires the minority group to be "sufficiently large and geographically compact to constitute a majority in a reasonably configured district" (*Allen v. Milligan*, 2023). To provide evidence of this, the plaintiff typically produced for the court one or more demonstration maps (eleven in *Allen*), each with more *majority-minority districts* than the enacted map being challenged. Gingles 1 is satisfied if at least one demonstration map is reasonably configured ("if it comports with traditional districting criteria").

Throughout this paper, we focus on Black voters. This reflects a common scenario in voting rights litigation and the largest racial minority in the USA according to Census data. Hence, districts will be considered majority-minority if a majority of the *voting age population* is Black (*Bartlett v. Strickland*, 2009), excluding those who selected multiple races.

Gingles claims have also been used to allege vote dilution against other minority groups. We present corresponding results for claims related to majority-Hispanic districts in Table B.5 in Appendix B.1.

4.2.3 Redistricting with Imperfect Data

It has long been recognized that the Decennial Census is far from perfect [Zalesin, 2020]. For example, coverage errors are large and vary by race: estimates for 2020 are 3.30% Black undercount and 4.99% Hispanic undercount, compared to 0.66% White overcount [Khubba et al., 2022]. Even if the Decennial Census was a perfect record of the population on Census Day, populations can change significantly in the 10 years between censuses when districts have historically remained unchanged.

The Supreme Court recognizes these issues. Generally, the Census's official Redistricting Data is considered "the only basis for good-faith attempts to achieve population equality" despite being "less than perfect" (*Karcher v. Daggett* 1983). But the Court has also acknowledged that deviating from the Redistricting Data can be appropriate. In *Mahan v. Howell*

(1973), a plan was considered malapportioned when about 18,000 people were known to live off of the naval base in which they were counted in the 1970 Census.

Alabama's lawsuit against the 2020 DAS raises an important question. What data should redistricting law treat as the ground truth: the data as collected, edited, and imputed (the CEF) or as published (the Redistricting Data)? No court has ever faced this precise question.

Kenny et al. [2021] consider Alabama's view (treating the CEF as ground truth for redistricting) to be the law as "currently interpreted by courts and policy-makers." Under this view, discrepancies between the Redistricting Data and the CEF could plausibly *impede* redistricting by forcing mapmakers to account for measurement error, and might even be argued to *inhibit* it if no plan could satisfy threshold requirements when judged against the CEF. (Looking ahead, our results suggest that for the state-legislative setting and plan-generation procedures we study, the "impede" concern is real but manageable; see Section 4.9 for further discussion).

We agree instead with Cohen et al. [2022]: the Redistricting Data should remain the only benchmark for redistricting requirements, considering the overall scale of the noise from disclosure avoidance in the face of other sources of imperfection that the Supreme Court has long countenanced, and the infeasibility of enforcing requirements measured against a confidential data source. Under this view, redistricters would not be required to account for data discrepancies to meet legal threshold requirements, though they may still wish to do so as a prudential matter.

4.3 Methods and Data

4.3.1 Data Sources

We use data from the 2010 Census P.L. 94-171 Redistricting Data Summary Files (henceforth SWAP) and the Privacy-Protected 2010 Census Demonstration Data Vintage 2021-06-08

(DEMO) [Van Riper et al., 2020].

SWAP is the official redistricting data from the 2010 Census, produced by applying the swapping-based 2010 DAS to the 2010 CEF. DEMO was produced by applying the 2020 DAS to the same 2010 CEF. Both DEMO and SWAP contain tabulations of population by voting age, sex, race, ethnicity for each geographic unit on the census geographic hierarchy—census blocks, block groups, tracts, counties, states, and the nation as a whole.

SWAP and CEF agree on the total and voting age populations of every census geography, and agree on all statistics at the state level. In contrast, DEMO and CEF agree on the total population only at the state level, and all sub-state counts may differ. Neither SWAP nor DEMO agree with the CEF on population by race for census blocks.

To create the input data for our redistricting algorithms, we link these data with TIGER/-Line Shapefiles [US Census Bureau, 2023b] and block equivalency files for voting precincts and legislative districts enacted after the 2010 Census [US Census Bureau, 2023a]. For precincts, we additionally joined our data with dual graphs provided by the maintainers of the GerryChain software package [Rule et al., 2021] for simpler integration into our algorithm.

4.3.2 Ensemble Analysis

In the past decade, algorithmic methods for studying gerrymandering and redistricting have made their way from academia [Engstrom and Wildgen, 1977, Cirincione et al., 2000, Chen et al., 2013, Duchin et al., 2019, Fifield et al., 2020b] to the Supreme Court (*Gill v. Whitford* (2018), *Rucho v. Common Cause* (2019), *Harper v. Moore* (2022), *Allen v. Milligan* (2023)).[4]

Markov Chain Monte Carlo (MCMC) methods enable sampling ensembles of thousands or millions of redistricting plans from predefined distributions for analysis [DeFord and Duchin,

4. Ensemble-based evidence has been more persuasive at trial courts than at the Supreme Court. It was received positively by the dissent in *Rucho v. Common Cause*, but less so in *Allen v. Milligan*: "[C]ourts should exercise caution before treating results produced by algorithms as all but dispositive of a §2 claim."

2022, Autry et al., 2023a, Fifield et al., 2020a]. Our paper makes use of the ReCom algorithm, a merge-split MCMC algorithm which has been used by expert witnesses in gerrymandering cases to generate ensembles [DeFord et al., 2021, *Harper v. Hall* 2022, *Allen v. Milligan* 2023]. We use the implementation from the GerryChain software package [Rule et al., 2021] (with some modifications, as described in Section 4.6).

We use voting precincts as the smallest unit of geography for drawing districts.[5] Keeping precincts intact is often considered desirable in practice by state legislatures and previous academic work on redistricting ensembles has used precincts to compose districts [Kenny et al., 2021]. For a few states, precinct data was incompatible with our experimental setup (either due to especially large precincts or incomplete geographical data). In those cases, we used block groups as the building block of districts in our analyses. See Section 4.8 for a discussion of how these choices may have affected our results.

Details for population balance (Section 4.5) Except where otherwise noted, each ensemble contains 100,000 plans generated by running ReCom for 1,000,000 steps and selecting every tenth plan. We impose various population deviation limits on the MCMC sampler, as described in Section 4.5.

We generate ensembles for each of 93 state legislative geographies. These include the upper and lower geographies for 45 bicameral states and one geography for Nebraska's unicameral legislature. It excludes lower houses in 4 states (Hawaii, New Hampshire, North Dakota, and Vermont) and upper houses in 2 states (Hawaii and North Dakota) where our MCMC chains do not find any plans that are population balanced to within 5% in DEMO. This limitation is due to our use of voting precincts rather than Census blocks as the unit of geography in our chains. Since the state legislative districts in these states are relatively small in population, it is computationally inefficient to find valid districts composed of precincts

5. We use the term precinct instead of the more correct "voting district" or VTD to avoid confusion with the much larger state and Congressional districts that are the main subject of this paper [United States Census Bureau, 1994].

using MCMC methods.

Details for majority-minority districts (Section 4.6) In Section 4.6, we generate plans using both ReCom and a modified version of ReCom optimized to select plans with more majority-minority districts (MMDs). The latter uses a technique called *short bursts*, which has been shown to outperform other techniques including biased random walks and simulated annealing [Cannon et al., 2023]. Details are deferred to Section 4.6.1.

Ensembles using unmodified ReCom contain 100,000 plans generated as above. Ensembles using short-bursts optimization contain 1,000 plans each (except for the Georgia state house ensemble which contains 5,000 plans).

Of the 93 state legislative geographies described above, our ensembles found at least one majority-Black district in 52 of them (27 lower house, 25 upper house). We restrict our analysis of MMD redistricting to these 52 geographies. For each geography, we impose a population deviation limit of 5% minus the *critical offset* found in Section 4.5.

Convergence Table B.1 in the Appendix summarizes results for Markov chain convergence tests for our main ensemble analyses. Using the Gelman-Rubin split-$\hat{R}$ diagnostic Gelman et al. [2013], we find that estimates of the metrics we study in Section 4.5 show clear signs of convergence in the large majority of the geographies we study ($\hat{R} \leq 1.01$, $\mathsf{ESS} \geq 400$). Compared to the analyses in Section 4.5, fewer ensembles in Section 4.6 exhibit clear signs of convergence. In that context, we view the convergence tests as less important: we are using MCMC sampling more as a way to optimize quantity of interest (the number of MMDs) rather than to estimate one (the MMD discrepancy), in line with the *Gingles 1* test.

4.3.3 Notation

Let data denote a reference dataset, either DEMO or SWAP. A state legislative redistricting plan $\mathcal{P}$ is a partition of a state into k districts: $D_1, \ldots, D_k$. Each district D_i is a contiguous

collection of census blocks.[6] We denote a district's population in data as $\mathsf{pop}_{\mathsf{data}}(D)$, and its voting-age population $\mathsf{VAP}_{\mathsf{data}}(D)$.

The ideal population of each district in a plan is $\overline{p}_{\mathsf{data}} = \frac{1}{k} \cdot \sum_{i=1}^{k} \mathsf{pop}_{\mathsf{data}}(D_i)$. It depends only on the total population of the geography and the number of districts. Because the 2010 and 2020 DASes do not affect a state's total population, $\overline{p}_{\mathsf{demo}} = \overline{p}_{\mathsf{swap}}$ for state legislative redistricting. As this setting is our focus, we will use $\overline{p}$ throughout. Fixing $\overline{p}$, we define the *population deviation* of a district and of a plan, respectively, as $\mathsf{dev}_{\mathsf{data}}(D) = |\mathsf{pop}_{\mathsf{data}}(D) - \overline{p}|/\overline{p}$ and $\mathsf{dev}_{\mathsf{data}}(\mathcal{P}) = \max_{i=1,\ldots,k} \mathsf{dev}_{\mathsf{data}}(D_i)$. This follows the convention of Kenny et al. [2021] of measuring deviation relative to the ideal population, and is a measure directly supported by our MCMC sampler.[7]

We denote by $\mathsf{pop}^{\mathsf{Black}}_{\mathsf{data}}(D)$ and $\mathsf{BVAP}_{\mathsf{data}}(D)$ the total and voting-age Black populations, respectively. We say a district is *majority-Black* if $\mathsf{BVAP}_{\mathsf{data}}(D)/\mathsf{VAP}_{\mathsf{data}}(D) > 0.5$. The number of majority-Black districts in a plan $\mathcal{P}$ according to data is denoted $\mathsf{MMD}_{\mathsf{data}}(\mathcal{P})$.

4.4 Related Work

Two prior studies have used ensemble methods to study the effects of the 2020 DAS on redistricting, specifically on population balance and the analysis of racial gerrymandering [Kenny et al., 2021, Cohen et al., 2022]. We discuss them below. Other research on the 2020 DAS include studies of the impacts on different policy issues, including minority representation [Christ et al., 2022], misallocation of federal funds [Steed et al., 2022], and public health monitoring [Krieger et al., 2021], for example.

The Census Bureau has also published its own analyses of the 2020 DAS. The most rele-

6. In our main experiments, districts are constructed from larger units called *precincts* (each precinct aggregates many census blocks); we use "blocks" here only to align with the Census geographic hierarchy. See Section 4.8 for discussion of how using precincts (rather than blocks) may affect the observed impact of DAS noise.

7. Courts typically instead measure deviation as the largest population minus the smallest, divided by the smallest. To guarantee that this latter measure is at most τ^*, it suffices that $\mathsf{dev}(\mathcal{P}) \leq 2\tau^*/(2+\tau^*)$ (e.g., for $\tau^* = 10\%$, $\mathsf{dev}(\mathcal{P}) \leq 0.094$ suffices).

vant examines the impact of the 2020 DAS on the reliability and consistency of measurements of demographic characteristics in various geographies [Wright and Irimata, 2021]. Among other findings, it shows that the 2020 DAS changes the apparent fraction of the population belonging to the largest demographic group by at most 5% in 95% of block groups with total population 450 to 499. Such reliability guarantees are important for Voting Rights Act enforcement, but don't do away with issues presented by sharp thresholds, as is the focus of our work.

4.4.1 Kenny et al.

Kenny et al. [2021] study the effects of the 2020 DAS on redistricting through a variety of case studies using ensemble-based comparisons of DEMO and SWAP.[8] We adopt their basic setup: generate many maps using the DEMO data, and measure their characteristics on the SWAP data.

Population balance Kenny et al. [2021] study of the impact on population balance for state legislative districts, with the Louisiana state senate as a case study [Kenny et al., 2021]. They generate ensembles of 5,000 plans using a merge-split MCMC algorithm (the same algorithm we use, but a different implementation [Kenny et al., 2022]). They find that a significant fraction of redistricting plans satisfying a given maximum population deviation τ on DEMO exceed the τ limit on SWAP [Kenny et al., 2021, Figure S4.4]. For example, for $\tau = 1\%$, about 75% of plans exceed the deviation limit on SWAP; for $\tau = 5\%$, about 5% of plans do. We include these results in Figure 4.1.

In Section 4.5, we greatly extend this analysis. We extend it in scale: using 93 state legislative geographies (instead of 1) and ensembles of 100,000 plans (instead of 5,000).

8. Only a non-final version of DEMO (called "DAS-12.2 data" in Kenny et al. [2021]) was available at the time of their initial experiments. Results using the more recent DEMO data ("DAS-19.61 data") are in [Kenny et al., 2021, Supplementary Materials].

We also extend it in scope: introducing *offsets* to mitgate discrepancies between DEMO and SWAP and measuring their effect and also proposing a model to explain the empirical observations.

Majority-minority districts Kenny et al. [2021] consider how the DAS affects the apparent number of majority-minority districts that can be drawn [Kenny et al., 2021, Table S4.2], through a case study reexamining *NAACP of Spring Valley v. East Ramapo Central School District* (2020). They sample 10,000 plans (5% max deviation) for the school district using DEMO, and count the number of MMDs in each plan based on DEMO and SWAP. The find fewer majority-minority districts in DEMO versus SWAP.

We find the opposite, across many state legislative geographies (Section 4.6). As we explain next, we believe the disagreement stems from differences in how our two works (i) define and count MMDs, and (ii) generate ensembles of redistricting plans.

We define an MMD as a district having more than 50% of *voting age population* in the minority group. This is the measure used in Supreme Court cases including *Allen v. Milligan* (2023), *Bartlett v. Strickland* (2009), and others. Kenny *et al.* instead use *registered voters* as was done in the Ramapo case. Race data for registered voters in Ramapo are not available. To get around this, Kenny *et al.* count MMDs using inferred race derived from the surnames and addresses of registered voters using a technique called Bayesian Improved Surname Geocoding (BISG). BISG predicts the race or ethnicity of the registered voters in a district based on their surname and the demographic makeup of the district as measured using census data (i.e., DEMO or SWAP). Because of this, the effect of the 2020 DAS on counting MMDs as measured in Kenny et al. [2021] is mediated through its effect on BISG. The latter is a significant confounder: comparing DEMO and SWAP, the inferred "proportions of Black and Hispanic [registered voters] are much smaller, especially in the blocks where they form a majority group" [Kenny et al., 2021].

The second difference is our use of an optimization technique designed to sample plans

with many majority-Black MMDs. Compared to the race-blind sampler used in Kenny et al. [2021] and in Section 4.5, our MMD-optimizing sampler appears to exacerbate the discrepancies we observe.

4.4.2 Cohen et al.

Cohen et al. [2022] also use ensemble methods to study the effects of the 2020 DAS on redistricting. (But as we elaborate below, the methods used in that paper make it less directly relevant to the present work than it may seem.) Like Kenny et al. [2021], Cohen et al. [2022] offer a detailed study of a small handful of case studies. Our study of offsets in Section 4.5 was in part inspired by the proposal in Cohen et al. [2022] to tweak a standard method for measuring racially-polarized voting (the 2nd and 3rd *Gingles* preconditions).

While it goes a long way to understanding the scale and characteristics of the noise introduced by disclosure avoidance, the prior work does not study the interaction between the noise from disclosure avoidance and the sharp thresholds. Perhaps this is one reason that Cohen et al. [2022] and Kenny et al. [2021] arrive at opposite conclusions about whether the official Redistricting Data should be treated as ground truth for legal tests.

Finally, Cohen *et al.* overlook the possibility that redistricting with the perturbed data might induce additional effects. Whereas our MMD analysis creates plans using the perturbed data, the closest case study in Cohen *et al.* Cohen et al. [2022] only examines the demographic makeup of four existing districts in Navajo County, Arizona.

Methodology Cohen *et al.* do not use the Census demonstration data whatsoever—unlike the majority of independent analyses of the 2020 DAS's effects, including Kenny et al. [2021] and ours. Instead, they analyze samples of noised data generated using an early version of the 2020 DAS, which they call *TopDown18* and that they run themselves. In brief, they run TopDown18 with various parameter settings and, separately, sample redistricting plans

using various map drawing methods. Then they analyze the scale and characteristics of the noise from TopDown18, aggregated up to the sampled districts, for the parameters and map drawing methods tested.

One limitation of this approach is that TopDown18's algorithm, implementation, and parameters are very different than the final version of the 2020 DAS. The other important limitation is the need to resort to using less-than-perfect data as input to the DAS, the real data being confidential. Cohen *et al.* used a reconstructed version of the 2010 Census data constructed to be consistent with the official tabulations published in SWAP [Cohen et al., 2022]. There is no way to know how the reconstructed dataset differs from the real data, or how those differences might have affected the analyses in Cohen et al. [2022].

4.5 Balancing District Populations with Noise

This section investigates how perturbations from disclosure avoidance interact with population balance requirements. We consider a data user's ability to use the public DEMO dataset (after disclosure avoidance) to generate redistricting plans satisfying a maximum population deviation target under the confidential CEF dataset (before disclosure avoidance). Because SWAP and CEF agree on population totals, we can use SWAP as a perfect stand-in for CEF.

A significant fraction of redistricting plans satisfying a given population deviation limit on DEMO exceeds that limit on SWAP, as first shown by Kenny *et al.* [Kenny et al., 2021]. They conclude that it is impossible to produce redistricting maps that adhere to One Person, One Vote requirements on the unseen SWAP dataset, pointing to state legislative districts as particularly challenging.

This section challenges that conclusion. We propose a straightforward method for ensuring that state legislative maps meet population balance thresholds under both datasets. Our method—requiring a smaller population deviation on the public DEMO dataset—is very

effective, and does not greatly increase the difficulty of finding valid plans[9] nor require the adoption of new redistricting techniques. See Section 4.8 for a discussion of limitations.

We focus on state legislative districts for a few reasons. First, some population deviation is allowed but precise thresholds are still observed. Our techniques would be ill-suited for studying Congressional maps, where population deviations greater than 1 person must be justified, because our algorithms are unable to generate ensembles of maps such strict requirements. Second, states offer enough diversity to observe meaningful variation, while also being few enough to cover exhaustively. Third, we are building on the existing case study of the LA state senate in Kenny et al. [2021].

4.5.1 Meeting Population Deviation Limits Using Offsets

We begin with a case study of the Louisiana state senate—where the sort of population discrepancies we study were first observed by Kenny *et al.* [Kenny et al., 2021, Figure S4.4]—then extend the analysis to 93 state legislative geographies.

We propose a simple approach: draw the plan to satisfy a slightly tighter limit $\tau - \Delta$ on DEMO, for some value $0 \leq \Delta \leq \tau$. We call Δ the *offset*. To evaluate this approach, we measure the *discrepancy at τ with offset Δ*: the fraction of plans that exceed τ deviation under SWAP among an ensemble of plans with deviation at most $\tau - \Delta$ under DEMO. When $\Delta = 0$, we call this the *discrepancy at τ* or the *no-offset discrepancy rate*. Finally, we also consider what we call the *critical offset*: the smallest offset with 0 discrepancies observed in our ensembles of 100,000 plans (see Appendix B.1 for more information on the computation of the critical offset).

Conceptually, this is an existence exercise: for our district-construction procedure, does there exist an offset Δ such that plans drawn to satisfy $\tau - \Delta$ on DEMO are empirically

9. As evidence of the fact that decreasing this threshold does not greatly increase the difficulty of finding valid plans, all of the ensembles analyzed in Section 4.6 are sampled using population tolerance offsets derived in Section 4.5.

likely (in our ensembles) to satisfy τ on SWAP. The resulting critical offsets are therefore specific to our simulation method and choice of geographic building blocks (e.g., precincts versus blocks), and should not be treated as universal constants for all redistricting processes (Section 4.8). Still, this approach can provide a useful framework for reasoning about the effects of disclosure avoidance on OPOV concerns.

A Case Study: the Louisiana State Senate

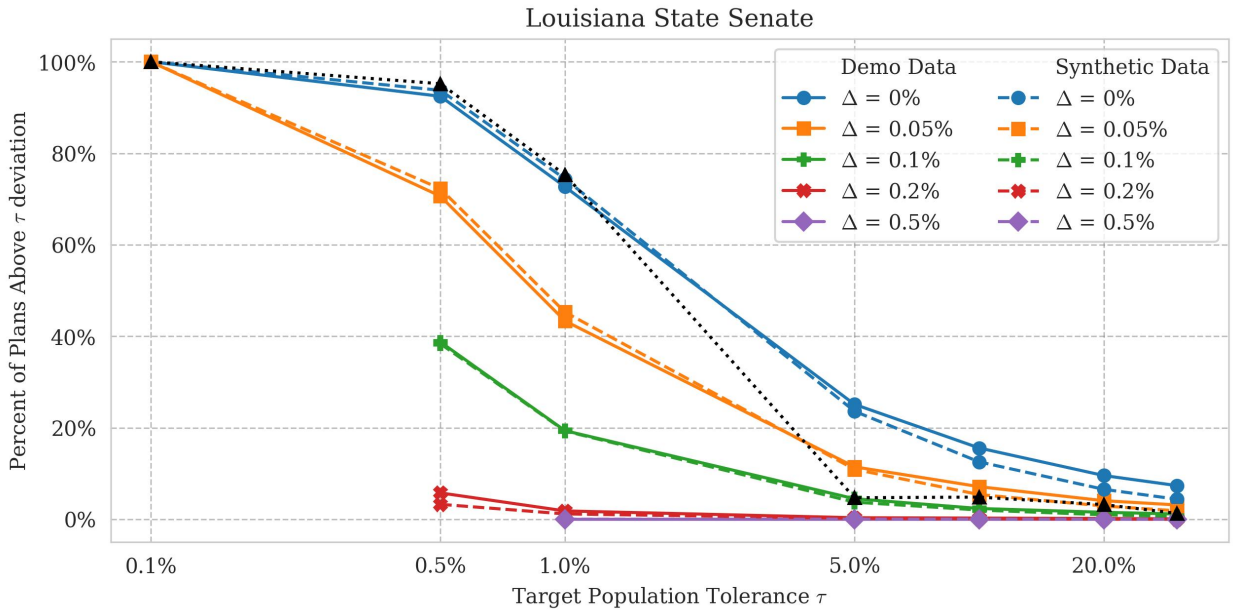


Figure 4.1: **Discrepancy with Offsets in the Louisiana State Senate.** We plot the fraction of plans exceeding intended population tolerance limit τ, with various offsets Δ for the Louisiana state senate (i.e., discrepancy at τ with offset Δ). Solid lines are computed using the DEMO and SWAP datasets with ensembles of 100,000 plans for each τ and Δ (see Sec. 4.5.1). Dashed lines are computed from 100,000 samples from the statistical model of district populations and disclosure avoidance noise described in Sec. 4.5.2. Black triangles (dotted lines) represent the results obtained by [Kenny et al., 2021, Figure S4.4].

Replicating Kenny et al. [2021], we measure the no-offset discrepancy for each of six population deviation thresholds τ between 0.1% and 30%. Namely, we generate an ensemble of 100,000 plans for the LA state senate with population deviation at most τ according to DEMO, and count what fraction of those plans exceed τ deviation according to SWAP.

The results are summarized by the solid blue line in Figure 4.1. The overall conclusion remains the same as in Kenny et al. [2021]: many plans drawn to a satisfy τ population deviation using DEMO exceed that threshold on SWAP. The exact numbers differ, perhaps due to differences in our experimental setups, such as a different MCMC sampler implementation and larger ensembles.

To test the effectiveness of offsets as a mitigation, we measure the discrepancy at τ and varying the offset parameter Δ, for the same six values of τ as before. Additionally, we compute the critical offset for the Louisiana State Senate for $\tau = 5\%$ to be $\Delta = 0.33\%$ (averaged over 10 runs).

The solid lines in Figure 4.1 illustrate the results for $\Delta = 0.05\%$, 0.1%, 0.2%, and 0.5%. Using $\Delta = 0.1\%$ reduces the discrepancy at $\tau = 5\%$ from 23.4% to 4.0%. Using $\Delta = 0.2\%$ reduces the discrepancy at $\tau = 1\%$ from 74.2% to 1.1%. Notably, the discrepancy at $\tau = 5\%$ with offset $\Delta = 0.5\%$ was 0%. That is, of 100,000 plans generated using a 4.5% tolerance on DEMO, none exceeded a 5% tolerance on SWAP.

Offsets for State Legislative Redistricting

We perform a similar analysis for 93 state legislative geographies. For each geography, we measure the discrepancy at $\tau = 5\%$ as we increase Δ in steps of 0.05% (generating an ensemble of 100,000 plans for each offset). For each geography we record the no-offset discrepancy ($\Delta = 0\%$) and the critical offset (where the discrepancy first falls to 0%).

Table 4.1 gives a coarse summary of the results (Table B.2 in Appendix B.1 gives much more detail). Observe that the no-offset discrepancy rates vary widely. In the worst case (Kansas State House of Representatives) 96% of plans exceeded 5% population deviation under SWAP. But offsets provide a powerful mitigation: for most geographies, $\Delta \leq 0.75\%$ suffices to eliminate plans that exceed a 5% population tolerance from our ensembles.

We do not know why different geographies exhibit different discrepancies and critical

	Min	25^{th}	50^{th}	75^{th}	Max
Discrepancy Rate	4.73%	22.65%	40.59%	66.13%	100.00%
Critical Offset	0.10%	0.28%	0.44%	0.74%	2.18%

Table 4.1: **Discrepancy Rates and Critical Offsets at $\tau = 5\%$.** Summary of discrepancy rates ($\Delta = 0$) and critical offsets for 93 state legislative ensembles and $\tau = 5\%$ (extrema and quartiles). Note the ensemble maximizing discrepancy does not maximize the critical offset; likewise for the other order statistics.

offsets, although the size of districts appears to play an important role. In Figure 4.2, we examine the standard deviations of the observed district population errors in our state lower and upper legislative district ensembles. Notably, we observe a strong positive correlation between the logarithm of average district population and error standard deviation in both state lower houses ($r = 0.87$) and state upper houses ($r = 0.83$). We also observe weaker correlations between the logarithm of total state population and error standard deviation (lower houses: $r = 0.64$, upper houses: $r = 0.82$) and between the logarithm of the average number of census blocks per district and error standard deviation (lower houses: $r = 0.59$, upper houses: $r = 0.66$).

We ran additional experiments on the state of Louisiana to understand the link between district size and discrepancy rate. For each value of Δ we examined in Figure 4.1, we measure the discrepancy rate at $\tau = 5\%$ for plans with different numbers of districts, ranging from 5 to 160. This corresponds to average district populations between 28,334 and 906,674 people.[10] Results are shown in Figure 4.3. We observe that for all values of $\tau - \Delta$, the discrepancy rate is monotonically decreasing as the ideal district size increases.

4.5.2 Why Offsets Work: Most Deviation Isn't from Disclosure Avoidance

The effectiveness of our mitigation can be understood by disentangling the population deviation caused by disclosure avoidance from the deviation caused by other factors. The

10. For comparison: Louisiana Congressional districts contain about 766,000 people; New Orleans City Council districts contain about 72,500 people.

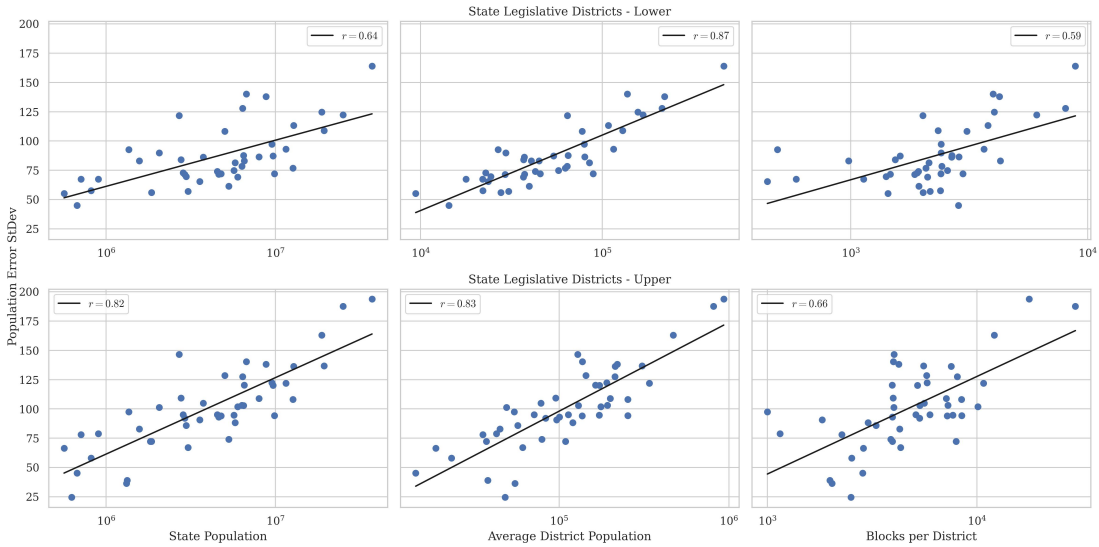


Figure 4.2: **Standard Deviation of Population Error Across State Legislative Geographies.** Standard deviations of the population error ($\mathsf{pop}_{\mathsf{demo}}(D) - \mathsf{pop}_{\mathsf{swap}}(D)$) over districts in ensembles generated for state upper and lower legislative districting plans. Each point represents a state legislative geography. The left column plots the logarithm of each state's population on the horizontal axis. The center column plots the logarithm of each legislative geography's average district population on the horizontal axis. The right column plots the logarithm of the total number of blocks in each state divided by the number of districts in each respective legislative geography on the horizontal axis.

population deviation of an individual district $\mathsf{dev}_{\mathsf{swap}}(D)$ consists of two components:[11]

$$\mathsf{dev}_{\mathsf{swap}}(D) = \Bigg| \underbrace{\frac{\mathsf{pop}_{\mathsf{demo}}(D) - \overline{p}}{\overline{p}}}_{\text{signed } \mathsf{dev}_{\mathsf{demo}}(D)} + \underbrace{\frac{\mathsf{pop}_{\mathsf{swap}}(D) - \mathsf{pop}_{\mathsf{demo}}(D)}{\overline{p}}}_{\mathsf{err}_{\mathsf{das}}(D)} \Bigg|.$$

The first component is a signed version of $\mathsf{dev}_{\mathsf{demo}}(D)$—the apparent deviation in DEMO. This is entirely under the control of the mapmaker. The second component is the additional error from disclosure avoidance, which we denote by $\mathsf{err}_{\mathsf{das}}(D)$.

Plotting these two terms side-by-side clarifies their relative import. Figure 4.4 plots histograms of signed $\mathsf{dev}_{\mathsf{demo}}(D)$ and $\mathsf{err}_{\mathsf{das}}(D)$ for all the LA state senate districts in our $\tau = 5\%$ ensembles. Note the different scales on the horizontal axes. The signed $\mathsf{dev}_{\mathsf{demo}}(D)$ is

11. This identity uses the fact that $\overline{p}_{\mathsf{demo}} = \overline{p}_{\mathsf{swap}}$ for state legislatures. Analyzing sub-state redistricting requires more care.

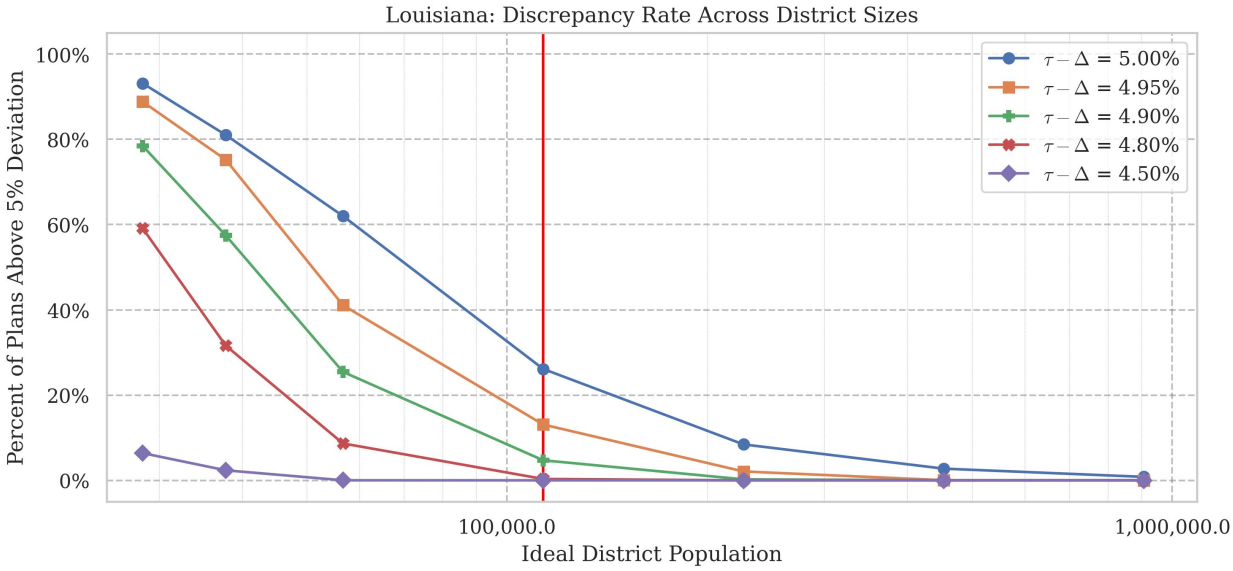


Figure 4.3: **Discrepancies by District Size in Louisiana.** Measuring the discrepancy rate of Louisiana redistricting plans over various district sizes and population tolerance offsets. Each point represents an ensemble that was generated using a population tolerance threshold of $5\% - \Delta$ on DEMO (for several values of Δ) and evaluated with a $\tau = 5\%$ threshold on SWAP. The ideal district populations correspond to plans with 5, 10, 20, 40, 80, 120, and 160 plans. The vertical red line corresponds with the actual district size of the Louisiana State Senate.

roughly uniform across the acceptable range $[-5\%, 5\%]$. The DAS noise $\mathsf{err}_{\mathsf{das}}(D)$ is highly concentrated, with mean 0.0% and standard deviation 0.080%. Only a small fraction of districts have $\mathsf{dev}_{\mathsf{demo}}(D)$ close enough to 5% for the DAS noise to matter. Using an offset moves $\mathsf{dev}_{\mathsf{demo}}(D)$ away from the 5% threshold, further lowering the number of districts where the DAS noise matters.

Figure 4.4 suggests a very simple probabilistic model for population deviations under SWAP. The deviation of a plan is sampled as the maximum over district-level deviations: $\mathsf{dev}_{\mathsf{swap}}(\mathcal{P}) = \max_i \mathsf{dev}_{\mathsf{swap}}(D_i)$. Each $\mathsf{dev}_{\mathsf{swap}}(D_i)$ is independently sampled as $|X + E|$, where X is uniform over $[-(\tau - \Delta), \tau - \Delta]$ and $E \sim \mathrm{N}(\mu, \sigma^2)$. The parameters $\mu = 0.0\%$ and $\sigma = 0.080\%$ are the empirical mean and standard deviation of $\mathsf{err}_{\mathsf{das}}(D)$ across districts in our LA state senate ensembles.

The dashed lines in Figure 4.1 show the fraction of 100,000 values of $\mathsf{dev}_{\mathsf{swap}}(\mathcal{P})$ sampled

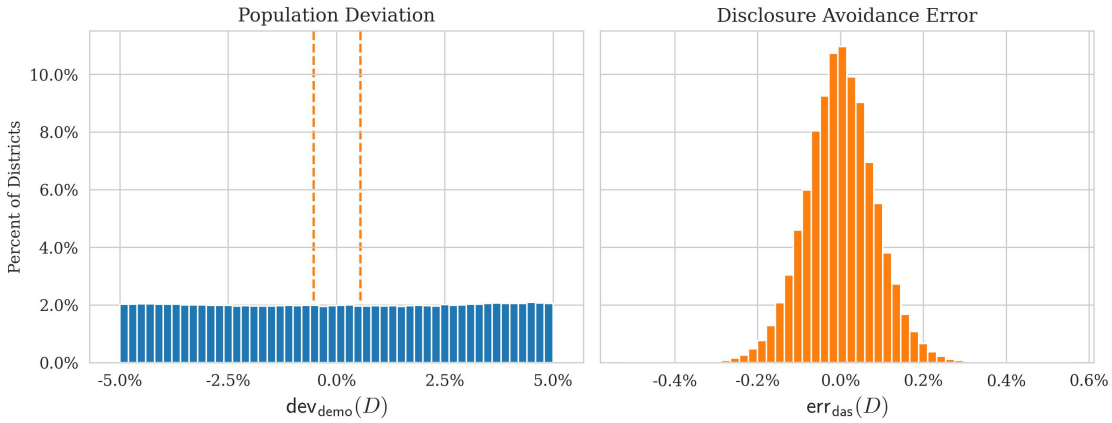


Figure 4.4: **The Two Components of Population Deviation in Districts.** Histograms of apparent population deviation in DEMO ($\mathsf{dev}_{\mathsf{demo}}(D) = (\mathsf{pop}_{\mathsf{demo}}(D) - \overline{p}/\overline{p})$ and the additional error from disclosure avoidance ($\mathsf{err}_{\mathsf{das}}(D) = (\mathsf{pop}_{\mathsf{swap}}(D) - \mathsf{pop}_{\mathsf{demo}}(D))/\overline{p}$) for each unique district included in an ensemble of Louisiana state senate plans sampled with a 5% population tolerance on the DEMO data. Note the very different scales on the horizontal axis. Together, these terms make up the population of a state legislative district's population deviation. The dashed vertical lines behind the left histogram represent the maximum and minimum error values observed in the right histogram.

as above that exceeded τ, for each τ and Δ. Qualitatively, the model closely approximates the empirical data for the LA senate, suggesting this is a useful mental model for understanding the effect of offsets.

Note that this is meant only as a simple, usable model of the phenomenon. In particular, the observed values of $\mathsf{err}_{\mathsf{das}}(D)$ do not appear normally distributed, failing the parametric bootstrap goodness of fit test implemented in `SciPy` Virtanen et al. [2020].)

4.5.3 A Closer Look at Errors from Disclosure Avoidance

To understand the effect of the 2020 DAS on OPOV questions, it appears that we must understand $|\mathsf{err}_{\mathsf{das}}(D)|$: the magnitude of population error in a district D due to disclosure avoidance, as a fraction of the ideal district population. This is perhaps unsurprising.

We briefly discuss this error in enacted districts, as compared to other sources of error, and in relation to racial biases in the 2020 DAS.

Ideal population $\overline{p}$	<8k	8-16k	16-32k	32-64k	64-128k	128-256k	256-512k	≥512k
Count	104	641	1,057	2,010	1,392	1,232	216	506
Max	0.0168	0.0214	0.0137	0.0069	0.0048	0.0051	0.0041	0.0011
98^{th} pct	0.0088	0.0111	0.0063	0.0039	0.0026	0.0017	0.0015	0.0006
90^{th} pct	0.0047	0.0064	0.0039	0.0023	0.0015	0.0010	0.0006	0.0003

Table 4.2: **Population Error in Enacted Districts.** Error from disclosure avoidance in state legislative districts and Congressional districts enacted after the release of the 2010 census redistricting data, as a fraction of ideal population: $|\mathsf{err}_{\mathsf{das}}(D)| = |\mathsf{pop}_{\mathsf{demo}}(D) - \mathsf{pop}_{\mathsf{swap}}(D)|/\overline{p}$. Districts are grouped by ideal population.

Errors in enacted districts Real redistricting plans are drawn by people, not sampled from ensembles. We do not know how offsets fare in a real redistricting setting. But the distribution of $|\mathsf{err}_{\mathsf{das}}(D)|$ on districts created by the political process can give us some initial idea. Using DEMO and SWAP, we compute $|\mathsf{err}_{\mathsf{das}}(D)|$ for every congressional and state legislative district enacted after the 2010 Decennial Census (rather than for algorithmic ensembles of districts). We group the districts by ideal population, which doesn't depend on the redistricting plan. Table 4.2 reports the maximum, 98th, and 90th percentile values in each group. For the overwhelming majority of geographies—including all with ideal population above $32,000$—the relative error magnitude from disclosure avoidance was well under 1%. A Rhode Island State House of Representatives district saw the largest observed relative error: 2.14%, corresponding to just 301 people.

A redistricter can use these numbers to guide the use of offsets where population deviation under the CEF is a concern, though more research would be warranted. If the ideal district population is $\overline{p} = 200,000$, an offset of $\Delta = 0.51\%$ might be a conservative choice; if $\overline{p} = 20,000$, an offset of $\Delta = 1.37\%$ appears more appropriate.

Magnitude relative to population drift As a point of comparison for the population deviations so far, we analyze district population imbalances arising from population shifts in the 10 years between decennial censuses. This is only one acknowledged source of population deviation among districts, and ignores systematic undercounts and overcounts that differ by

race and ethnicity [US Census Bureau, 2022]. Using data from IPUMS that standardizes congressional district geographies between decennial censuses, we compare total populations of congressional districts of the 110th-112th Congresses in both the 2000 and 2010 censuses [Manson et al., 2022].[12] When measured with the 2000 data used to draw the districts, the deviations are relatively small. Of states with more than one Congressional district, the average deviation is just 613 people (∼0.1% of the ideal population), while the maximum is 6,698 (∼1% of the ideal). By the 2010 Census, the average increased to 137,974 people (∼19% of the ideal), while the maximum increased to 340,948 people (Texas, ∼43% of the ideal).

Biases in population discrepancies Prior work found that noise from the 2020 DAS is correlated with racial/ethnic homogeneity [Kenny et al., 2021, Petti and Flaxman, 2019]. Specifically, precincts with higher Herfindahl–Hirschman index—a measure of racial and ethnic homogeneity—tend to have their populations inflated, and vice versa. [Kenny et al., 2021]. This motivates the following question: Does the distribution of district-level errors in our ensembles vary by the racial and ethnic makeup of the districts? If so, it might systematically shift voting power among groups even if district populations are still within the acceptable bounds. Fortunately, we do not observe any meaningful effect. For the 876,800 unique districts we sampled for the Georgia state house, a geography for which small changes in voting power of Black residents could have a meaningful impact on the makeup of the legislature, we find that the Pearson correlation between per-district population error and Herfindahl-Hirschman index is minute: $r = 0.07$, 95% CI $(0.068, 0.072)$.

12. Because Census geographies (i.e. block boundaries) are modified every 10 years, districts drawn using 2000 census data may split 2010 census blocks, complicating comparisons between years.

4.6 Counting Majority-Minority Districts with Noise

To challenge an enacted electoral map for racial vote dilution, plaintiffs must meet the Gingles 1 precondition. Gingles 1 requires showing that one can draw more majority-minority districts (MMDs) than exist in the enacted map being challenged. This is usually done by submitting expert reports with one or more such illustrative plans.

This section investigates how perturbations from disclosure avoidance interact with Gingles 1. We consider a data user's ability to use the public DEMO dataset (after disclosure avoidance) to produce illustrative plans with many majority-minority districts under the confidential CEF dataset (before disclosure avoidance). Notice that these illustrative plans may be quite different than those produced in the normal course of redistricting: they must have more majority-minority districts to satisfy Gingles 1. For this reason, we also investigate the additional impact of optimizing the number of majority-minority districts using a technique called *short-burt optimization* [Cannon et al., 2023] described below.

Unfortunately, the data needed to study these questions directly is unavailable: SWAP and CEF do not agree on population by race, though they do agree on voting age population. Therefore, we will use SWAP as an imperfect stand-in for CEF. As a result, the discrepancies we observe reflect the combined impact of perturbations from the 2010 DAS (CEF→SWAP) and the 2020 DAS (CEF→DEMO). We discuss this limitation further in Section 4.8, and take a first step towards overcoming it in Appendix 4.7.

This section specifically focuses on majority-Black districts and uses MMD to mean a district where a majority of the voting age population is Black, excluding those who select multiple races. As before, we restrict our attention to state legislative redistricting.

4.6.1 Short-Burst Optimization

Gingles 1 incentivizes plaintiffs to maximize the number of MMDs in the illustrative plan, shifting the distribution of plans. To study the effect of this shift, we generate ensembles of

plans using a technique called *short-burst optimization*, which is designed to output plans with many majority-Black MMDs [Cannon et al., 2023]. It is generally intractable to determine the maximum possible number of MMDs in a plan at the scale of a US state. Therefore, we use the optimized chains as a proxy for a redistricter who is trying to draw a plan with a near-maximal number of majority-minority districts.

We use short-burst optimization in a manner suggested to us by Duchin (personal communication, August 2025). We run 100 chains (500 for Georgia) with short burst optimization for 100,000 steps each in parallel. From each chain, we randomly subsample 10 plans from among those with highest MMD count in DEMO, resulting in 1,000-plan ensembles (5,000 for Georgia).

This approach is somewhat heuristic, but it allows us to observe the effects of incentivizing maps with certain characteristics, as Gingles 1 does. The ensembles can also be seen as containing plans that might have been selected as a Gingles 1 illustrative plan by a redistricter who used short-burst optimization to find maps with many MMDs. While the human and the algorithm differ in their methods for selecting plans, this methodology sheds light on the qualities of plans that are drawn from a nearly-maximal distribution.

This approach is computationally intensive, requiring hundreds of CPU-hours to generate an ensembles of 5,000 plans for the state of Georgia. We ran fewer chains for other states in order to allow us to complete our experiments in a reasonable timeframe. We observed that analyzing the first 1,000 plans of the Georgia ensemble yielded similar results to the full 5,000 plan ensemble.

4.6.2 Measuring MMD Discrepancy

We begin with a case study of Georgia state house plans, then extend the analysis to 50 state legislative geographies. The Georgia house serves as a good illustrative geography due to its significant Black population and large legislature.

We measure and examine the *MMD discrepancy* of plans generated using the DEMO dataset: $\mathsf{MMD}_{\mathsf{demo}}(\mathcal{P}) - \mathsf{MMD}_{\mathsf{swap}}(\mathcal{P})$. We measure the *mean MMD discrepancy* and the *non-zero discrepancy rate*—the fraction of plans with non-zero MMD discrepancy.

Georgia State House Plans

Figure 4.5 illustrates the MMD discrepancies that arise in 100,000 plans sampled using our base merge-split MCMC algorithm (Section 4.3.2). The histogram breaks down the plans by number of majority-Black MMDs according to the DEMO dataset. Each histogram bin is further broken down by MMD discrepancy: the difference between the number of MMDs as measured by DEMO and SWAP. Across the whole ensemble, the mean MMD discrepancy is 0.03, and 9% of plans have a non-zero MMD discrepancy. Furthermore, plans with more MMDs have slightly greater discrepancies (Pearson's $r = 0.08$).

Figure 4.6 illustrates the results of the same analysis for 5,000 plans generated using short-bursts optimization These numbers are markedly higher: the mean MMD discrepancy is 1.02, and 66% of plans have a non-zero MMD discrepancy. The correlation between number of MMDs and MMD discrepancy is also higher (Pearson's $r = 0.16$).

Other State Legislative Geographies

We examine the MMD discrepancies in the 52 geographies where our short burst ensembles found at least one MMD (27 lower house, 25 upper house). Full results are included in Table B.6 in Appendix B.1. Different geographies experience very different levels of MMD discrepancy. Some geographies experience very minor discrepancies; others very significant (e.g., 0.9% of CA lower house plans have non-zero discrepancies, compared to 78.9% of MS lower house plans).

Still, some patterns we see in the GA house generalize to almost all geographies. First, positive MMD discrepancies—more MMDs according to DEMO than SWAP—are both more

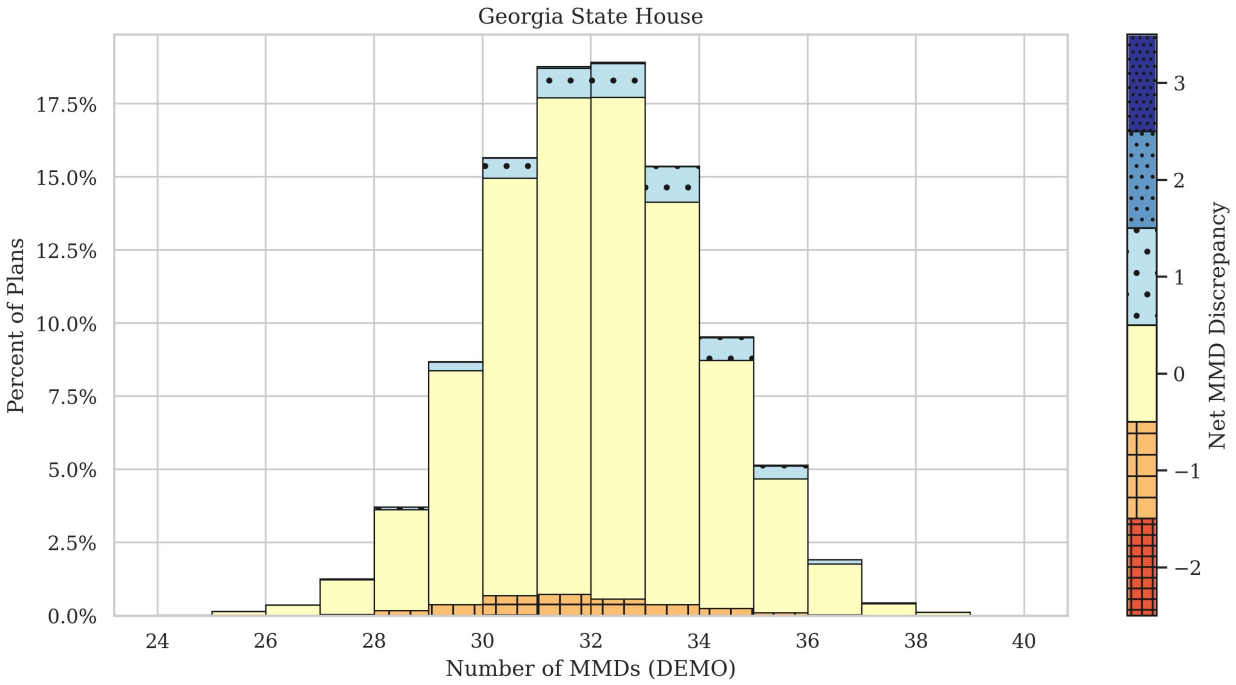


Figure 4.5: **MMD Discrepancies in the Georgia State House base Ensemble.** We plot the number of majority-Black districts in plans sampled in our base ensemble. Plans are grouped according to the number of majority-Black districts as measured using DEMO. Bars are colored to indicate the fraction of those plans which have the corresponding MMD discrepancy. Each ensemble contains 100,000 plans sampled with a 4.4% population deviation limit (utilizing $\Delta = 0.6\%$) using the DEMO data.

frequent and larger in magnitude than negative discrepancies. Second, plans with more MMDs have greater discrepancies. Finally, we observe that MMD discrepancies are generally greater in lower houses than in the (typically smaller) upper houses within the same state.

DEMO vs SWAP vs CEF

Recall that the discrepancies we observe reflect the combined impact of perturbations from the 2010 DAS (CEF→SWAP) and the 2020 DAS (CEF→DEMO). This is inherent to any analysis that compares the DEMO data to the 2010 Redistricting Data, including most analyses of the effects of the 2020 DAS. (Additional results in Appendix 4.7 suggest discrepancies might be similar if we were to use the 2010 CEF as a baseline, though these results require

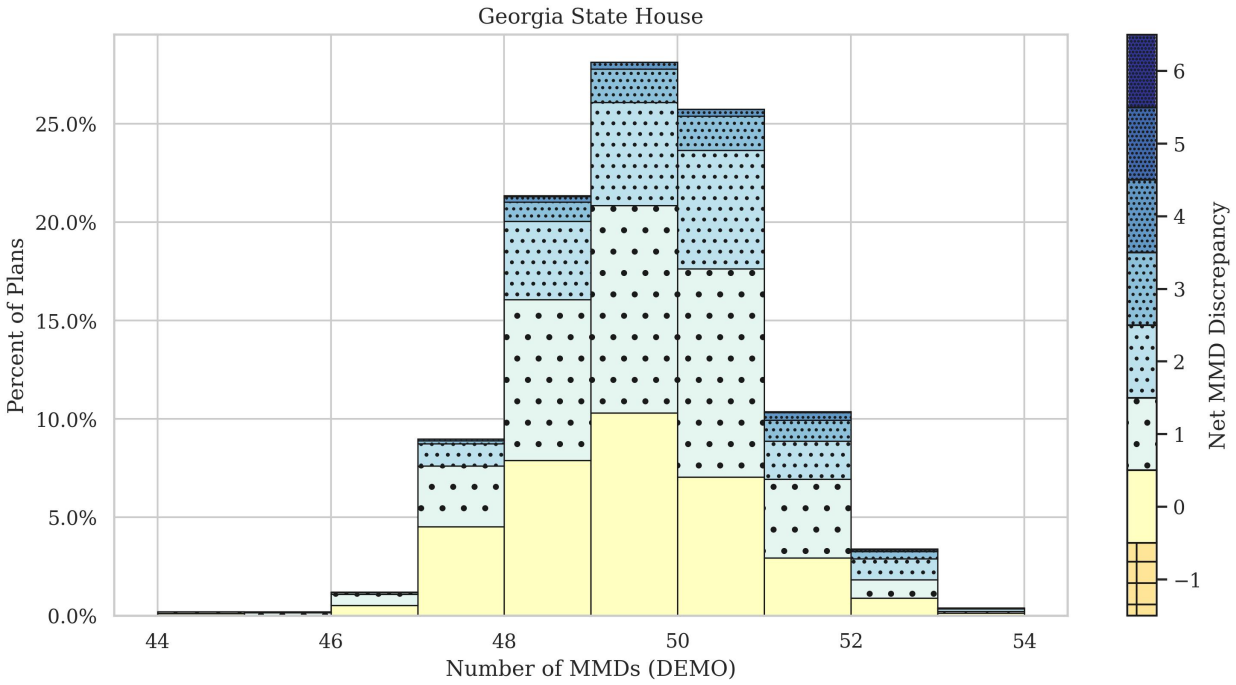


Figure 4.6: **MMD Discrepancies in the Georgia State House Optimized Ensemble.** We plot the number of majority-Black districts in plans sampled in our short bursts ensemble, which is designed to produce plans with many MMDs. Plans are grouped according to the number of majority-Black districts as measured using DEMO. Bars are colored to indicate the fraction of those plans which have the corresponding MMD discrepancy. Each ensemble contains 100,000 plans sampled with a 4.4% population deviation limit (utilizing $\Delta = 0.6\%$ offset listed in Table B.2) using the DEMO data.

strong assumptions on the re-implementation of the swapping algorithm in Ballesteros et al. [2025].)

Still, characterizing the expected discrepancies between DEMO and SWAP can be useful for redistricters in practice. If one accepts the need for disclosure avoidance in census tabulations but is concerned that the new DAS yields data that is less fit for use than the previous DAS, our results should provide some reassurance. Our results can be seen as useful for understanding whether DEMO is sufficiently close to SWAP for use in redistricting, setting aside whether SWAP is sufficiently close to the 2010 CEF to be fit for use. If it is, and if SWAP was fit for use for redistricting (which was never under question), then that lends support for DEMO's fitness for use as well.

4.6.3 The Connection Between BVAP Margin and MMD Discrepancy

Above, when drawing districts with DEMO and measuring against SWAP we see: (1) a bias towards more MMDs in DEMO than SWAP, and (2) the bias increases as the number of MMDs increases within a geography. This section describes a simple probabilistic model that helps explain these observations. We do so by connecting plan-level discrepancies to a district-level quantity (BVAP margin) whose sign determines whether a district is counted as majority-Black.

BVAP margin The key quantity is *BVAP margin*: the size of a district's Black voting age majority (i.e., Black voting age population minus half of the total voting age population). We denote a district's BVAP margins under DEMO and SWAP as:

$$B_{\mathsf{demo}}(D) = \mathsf{BVAP}_{\mathsf{demo}}(D) - 0.5 \cdot \mathsf{VAP}_{\mathsf{demo}}(D) \tag{4.1}$$
$$B_{\mathsf{swap}}(D) = \mathsf{BVAP}_{\mathsf{swap}}(D) - 0.5 \cdot \mathsf{VAP}_{\mathsf{demo}}(D)$$

By definition, a district is an MMD according to DEMO if and only if its BVAP margin under DEMO is positive: $B_{\mathsf{demo}}(D) > 0$. Likewise, for SWAP.

Given only the DEMO dataset, a redistricter can reason about MMDs under SWAP by observing the following:

$$B_{\mathsf{swap}}(D) = B_{\mathsf{demo}}(D) + \underbrace{\big(B_{\mathsf{swap}}(D) - B_{\mathsf{demo}}(D)\big)}_{\mathsf{err}^{\mathsf{MMD}}_{\mathsf{das}}(D)} \tag{4.2}$$

This decomposition separates what is observable by the redistricter from what is not (as Section 4.5.2 did for population balance). The first component $B_{\mathsf{demo}}(D)$ is entirely under the control of the mapmaker. The second component is the change in BVAP margin from disclosure avoidance, which we denote $\mathsf{err}^{\mathsf{MMD}}_{\mathsf{das}}(D)$.

By equation equation 4.2, a district contributes to plan-level MMD discrepancy precisely when the disclosure-avoidance term $\mathsf{err}^{\mathsf{MMD}}_{\mathsf{das}}(D)$ causes a *sign flip* between $B_{\mathsf{demo}}(D)$ and $B_{\mathsf{swap}}(D)$. Districts with $B_{\mathsf{demo}}(D) \approx 0$ are the most likely to flip. Therefore, we might hope to explain characteristics of the plan-level MMD discrepancies by looking at the characteristics of $B_{\mathsf{demo}}(D)$ and $\mathsf{err}^{\mathsf{MMD}}_{\mathsf{das}}(D)$, and focusing specifically on *near-threshold* districts where $B_{\mathsf{demo}}(D) \approx 0$.

Understanding biases in MMD discrepancy In our experimental setup, we tend to see a bias towards more MMDs in DEMO than in SWAP. This corresponds to plans having more districts D satisfying $B_{\mathsf{swap}}(D) < 0 < B_{\mathsf{demo}}(D)$ than the reverse. The more MMDs a plan has, the greater the observed bias tends to be. Why might this be?

The reasoning above suggests that we find an explanation by looking at biases in $B_{\mathsf{demo}}(D)$ and $\mathsf{err}^{\mathsf{MMD}}_{\mathsf{das}}(D)$, focusing on near-threshold districts where the BVAP margin is close to 0 (i.e., where sign flips / discrepancies are plausible).

Hence, we consider two possible sources for the observed bias. First, if $\mathsf{err}^{\mathsf{MMD}}_{\mathsf{das}}(D)$ is negatively biased. Second, if plans tend to have more districts D with BVAP just over 50% ($B_{\mathsf{demo}}(D) > 0$) than just under 50% ($B_{\mathsf{demo}}(D) < 0$). The latter would also help explain the correlation between the number of MMDs and the bias, as maximizing the number of MMDs requires smaller Black majorities on average.

Both explanations are borne out in our data for the Georgia state house. Figure 4.7 plots $\mathsf{err}^{\mathsf{MMD}}_{\mathsf{das}}(D)$ for the districts in the short-burst ensemble. For districts with BVAP margin close to 0, the mean $\mathsf{err}^{\mathsf{MMD}}_{\mathsf{das}}(D)$ is -22.303 (standard deviation 6.3).[13]

Figure B.1 plots histograms of $B_{\mathsf{demo}}(D)$ for all districts in both the base and short-burst ensembles. In the short bursts ensemble, districts with small Black majorities are much more common than districts with small non-Black majorities. This is not true in the base ensemble which ignores race, helping explain the smaller biases seen in those plans.

13. Across all districts, the mean $\mathsf{err}^{\mathsf{MMD}}_{\mathsf{das}}(D)$ is 0.030.

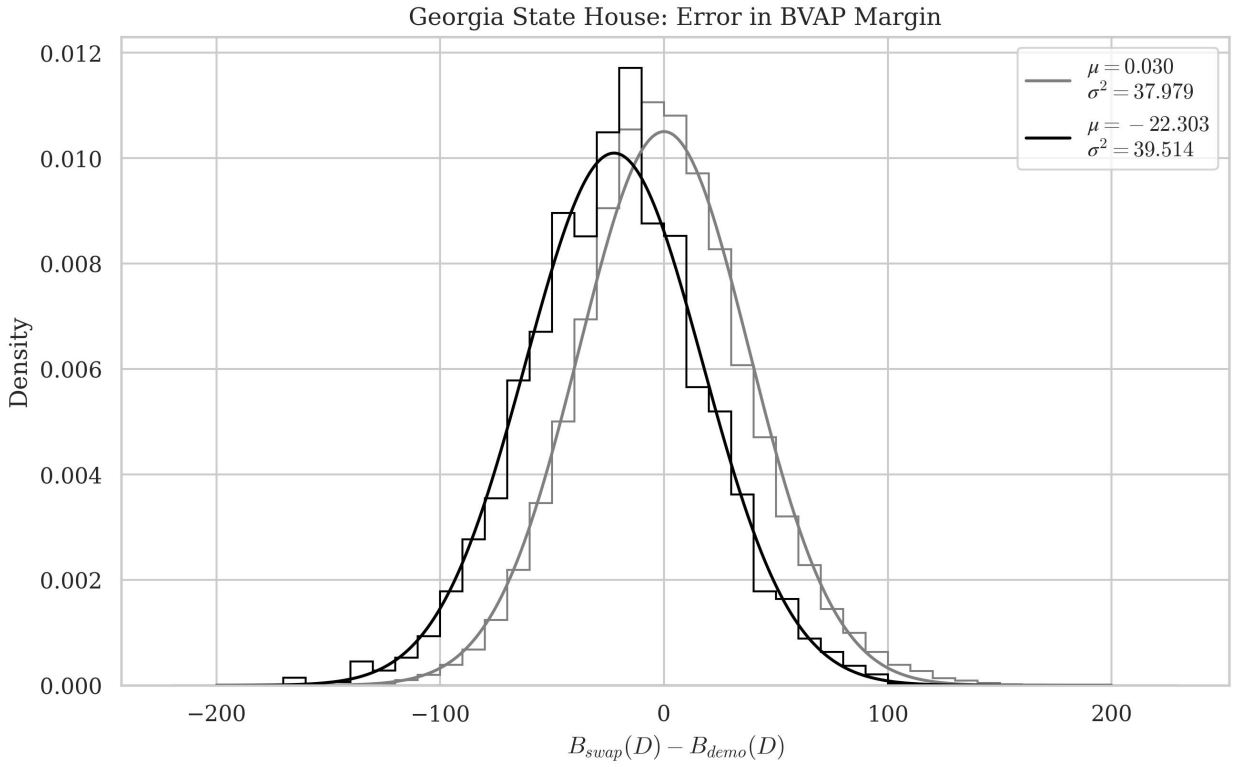


Figure 4.7: **Empirical Change in BVAP Margin.** Change in BVAP margin ($\mathsf{err}^{\mathsf{MMD}}_{\mathsf{das}}(D) = B_{\mathsf{swap}}(D) - B_{\mathsf{demo}}(D)$) among districts in our short bursts ensemble for the Georgia State House of Representatives. The gray histogram includes all districts that were contained within plans in the ensemble. The black histogram is restricted to districts with Black voting age populations within 196 individuals of half of the total VAP. The threshold of 196 was the maximum error in BVAP margin observed in the ensemble. Note that the noise in BVAP margin for districts that have close to 50% BVAP is biased in the negative direction.

A model of MMD discrepancies from BVAP margins The discussion above suggests a simple probabilistic model for MMD discrepancies using only BVAP margins, ignoring all other effects (e.g., correlations between errors across districts). The model aims to capture the behavior of near-threshold districts whose BVAP is close enough to 50% that disclosure-avoidance error could change their majority status.

Given a plan with districts D_i, sample $B_{\mathsf{swap}}(D_i) = B_{\mathsf{demo}}(D_i) + E_i$, where $E_i \sim \mathrm{N}(\mu, \sigma^2)$ is normally distributed. The parameters μ and σ are estimated from the ensemble by looking at districts whose BVAP margins are small enough that a sign flip is possible (i.e.,

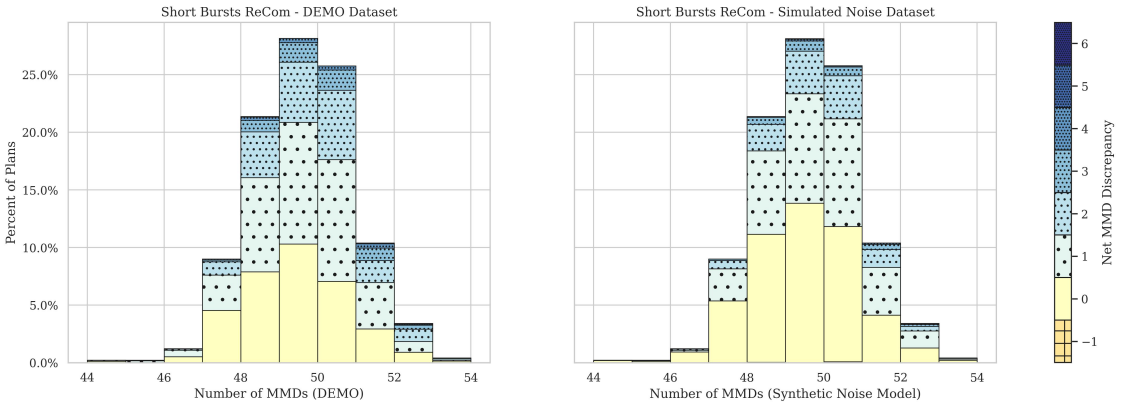


Figure 4.8: **Modeling MMD Discrepancies in the Georgia State House.** We compare the MMD discrepancies previously observed in Figure 4.6 to the MMD discrepancies predicted by our synthetic noise model for the Georgia state house.

$|B_{\mathsf{demo}}(D)| \leq \max_{D'} |\mathsf{err}^{\mathsf{MMD}}_{\mathsf{das}}(D')|$, the largest observed change in BVAP margin). Intuitively, this conditions on the at-risk districts for which noise can change MMD classification, and avoids letting the many districts with very large margins dominate the fit.

Figure 4.8 plots the results of applying this simulated MMD discrepancy model to the 5,000 Georgia state house plans in our short-bursts ensemble ($\mu = -22.303$, $\sigma^2 = 39.514$), side-by-side with the MMD discrepancies observed when comparing DEMO and SWAP. The two plots contain exactly the same plans each with the same number of MMDs in DEMO. The only difference is whether MMDs in SWAP are measured against the real data or sampled from the model above.

Qualitatively, the model appears to explain much of the observed biases in MMD discrepancies, but not all. The model is deliberately simple and does not capture all features of the observed discrepancies (e.g., non-normality and correlations in $\mathsf{err}^{\mathsf{MMD}}_{\mathsf{das}}(D)$), so it should be interpreted as a clarifying approximation rather than a full predictive account. More work would be needed to quantify the result and to understand what characteristics of the data are driving the remaining bias.

Offsets for Gingles 1 plans As in Section 4.5, a natural approach to mitigating MMD discrepancies in the context of Gingles 1 would be to draw illustrative plans whose majority-Black districts have large BVAP margins. Instead of targeting 50% VAP + 1 person, target 50% + 1 + *offset*, for some value *offset*> 0.

We leave a detailed study of the effectiveness of offsets for MMD discrepancies to future work. For now, we remark that this is already common practice. For example, across the eleven illustrative plans used in the recent Supreme Court case *Allen v. Milligan (2023)*, the smallest BVAP margin in a majority-Black district was 256 people (50.05% BVAP) Merrill v. Milligan [2022], larger than any $\mathsf{err}^{\mathsf{MMD}}_{\mathsf{das}}(D)$ observed in the Georgia ensemble.

4.7 Towards Disentangling the Impact of the 2010 and 2020 Disclosure Avoidance Systems

Our goal is to understand the impact of the 2020 DAS on MMD discrepancies. Unfortunately, comparing DEMO and SWAP reflects the combined effects of the 2010 DAS and the 2020 DAS. This is because DEMO is the result of applying the 2020 DAS to the confidential 2010 CEF data, while SWAP is the result of applying the 2010 DAS to the confidential 2010 CEF data.

One way to try to isolate the effect of the 2020 DAS would be to run it ourselves using reconstructed 2010 Census microdata derived from SWAP (as in Cohen et al. [2022]). But running the 2020 DAS reliably is difficult [Ballesteros et al., 2025] and beyond the scope of the present work.

Instead, we try to isolate the effect of the 2010 DAS using a dataset provided to us by the authors of Ballesteros et al. [2025] (personal communication, September 2025). The dataset is the result of applying an implementation of *swapping* (designed to approximate the 2010 DAS as closely as possible using only publicly-available information) to reconstructed 2010 Census microdata for Georgia, as described in Ballesteros et al. [2025]. Hence, this dataset—

which we call *SWAP-SWAP*—is the result of applying the swapping-based 2010 DAS to the 2010 CEF, then reconstructing it and applying an independent implementation of swapping.

We generate a new ensembles of plans using the SWAP-SWAP and DEMO datasets (using short-bursts optimization for ReCom with 100 chains in parallel). For each ensemble, we measure the MMD discrepancies relative to SWAP. (For this analysis only, we count majority-Black districts using the total population rather than the voting age population, because voting age data was not included in the SWAP-SWAP dataset.) The results are plotted in Figure 4.9. The similarities are clear. In both ensembles, the number of MMDs according the dataset used to generate the plans (DEMO, SWAP-SWAP) tends to be greater than the number of MMDs according to the unseen dataset (SWAP). Moreover, the bias grows with the number of MMDs.

Going a step further, SWAP-SWAP population data makes it possible to estimate what Black population margins might look like in the CEF. This holds under a strong assumption on the fidelity of the SWAP-SWAP data.[14] Namely, the analysis below requires the differences between tabulations in SWAP-SWAP and SWAP be distributed similarly to the differences between tabulations in SWAP and the 2010 CEF. Adapting the notation introduced in Section 4.6.3,

$$B_{\mathsf{swap\text{-}swap}}(D) - B_{\mathsf{swap}}(D) \approx B_{\mathsf{swap}}(D) - B_{\mathsf{cef}}(D), \tag{4.3}$$

where $B_{\mathsf{data}}(D)$ is the Black population margin (i.e., $\mathsf{pop}^{\mathsf{Black}}_{\mathsf{data}}(D) - 0.5 \cdot \mathsf{pop}^{\mathsf{Black}}_{\mathsf{dara}}(D)$) in district D as measured in dataset DATA. This differs slightly from the definition of $B_{\mathsf{data}}(D)$ in Section 4.6.3, which considers voting age population rather than total population. This is due to the lack of voting age information in SWAP-SWAP, as mentioned above.

Assuming the above, we can approximate Black population margins in the 2010 CEF

14. It appears difficult or impossible to validate this assumption for the same reasons that measuring the effects the 2010 swapping-based DAS is difficult: the swapping algorithm and the CEF input data is secret, known only to the Census Bureau.

directly from the margins we measured using SWAP and SWAP-SWAP:

$$\begin{aligned} B_{\mathsf{cef}}(D) &= B_{\mathsf{swap}}(D) + \big(B_{\mathsf{cef}}(D) - B_{\mathsf{swap}}(D)\big) \\ &\approx B_{\mathsf{swap}}(D) + \big(B_{\mathsf{swap}}(D) - B_{\mathsf{swap\text{-}swap}}(D)\big) \\ &= 2B_{\mathsf{swap}}(D) - B_{\mathsf{swap\text{-}swap}}(D) \end{aligned} \tag{4.4}$$

Using these estimates, we plot MMD discrepancies between DEMO and the above estimate of CEF in Figure 4.10. Discrepancies between DEMO and CEF are qualitatively similar to discrepancies between DEMO and SWAP, although slightly smaller in magnitude. This result suggests that our observations in Section 4.6 would have been similar had we been using the 2010 CEF rather than SWAP.

To reiterate, the analysis above assumes that the implementation of swapping used to create our SWAP-SWAP dataset closely approximates the 2010 DAS. More work would be needed to fully disentangle the effects of the 2020 and 2010 disclosure avoidance systems, even setting aside the extent to which comparing SWAP-SWAP to SWAP is a faithful representation of the effects of the 2010 DAS. For example, it is not clear whether the similarities between the two plots in Figure 4.9 mean that the effect of the 2020 DAS is smaller, larger, or similar to the effects of swapping. Instead, we remark only that any differences in the MMD discrepancies arising from the two systems appear more likely differences in degree than in kind.

4.8 Limitations

Methods An important limitation is that we study algorithmically sampled plans, incorporating only a few redistricting criteria (contiguity, population balance, and compactness). But real districts are drawn by people as part of a political process, taking account of many factors. One should not read too much into the specific numbers we observe: changing the

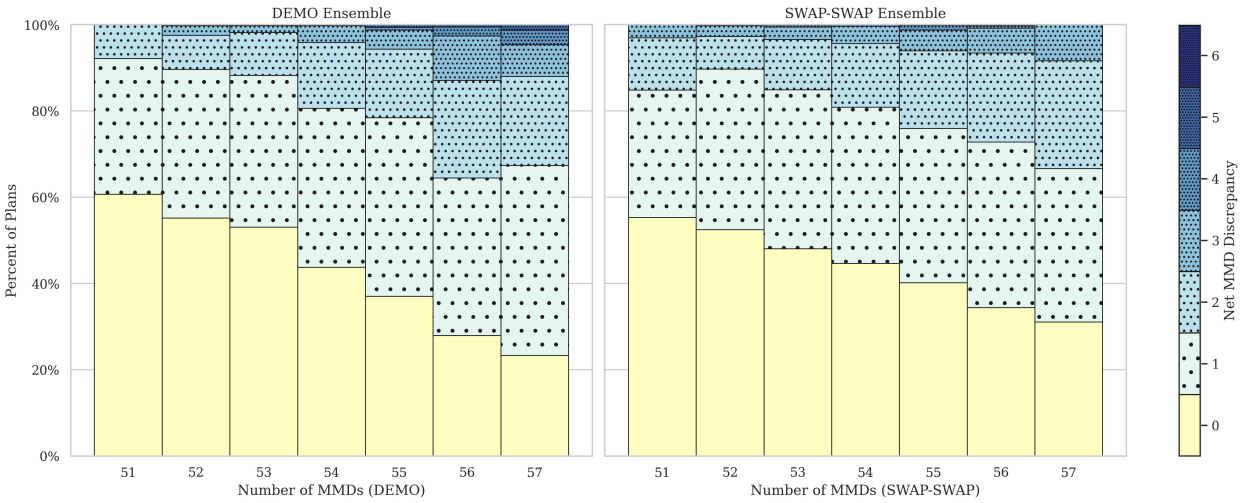


Figure 4.9: **MMD Discrepancies in SWAP-SWAP.** Distribution of MMD discrepancies (relative to SWAP) for Georgia state house using the DEMO and SWAP-SWAP datasets. The number of MMDs on the horizontal axis is measured using the dataset used to generate the ensembles (DEMO or SWAP-SWAP), and the percent of plans normalized to 100% for each bin is displayed on the vertical axis. We generate ensembles using the method described in Section 4.6.1, running 100 chains for each dataset.

way we generate districts would likely change the quantitative results.

Some algorithmic methods for sampling plans have the ability to generate representative samples of districting plans under relevant probability distributions for redistricting [McCartan and Imai, 2023, Autry et al., 2023b]. For the purposes of this paper, we make no such claims and ignore many of the complexities of practical redistricting. At the same time, from a technical point of view, we believe it is reasonable to offer statistics over a set of interest as a descriptive matter without a need for a distributional claim about the sample. One goal of this paper is to replicate and extend findings from prior work, and to explain the phenomena we observe. For such uses, having a clear, well-defined sample frame may be more useful than strict adherence to human redistricting practice.

Our sample frame does not contain all permissible plans: it excludes plans that subdivide precincts, for instance. Nor does it necessarily contain only permissible plans: ignoring political subdivisions and communities of interest in states where required, for example.

It is hard to predict the net effect of these differences on our findings. As we explain next,

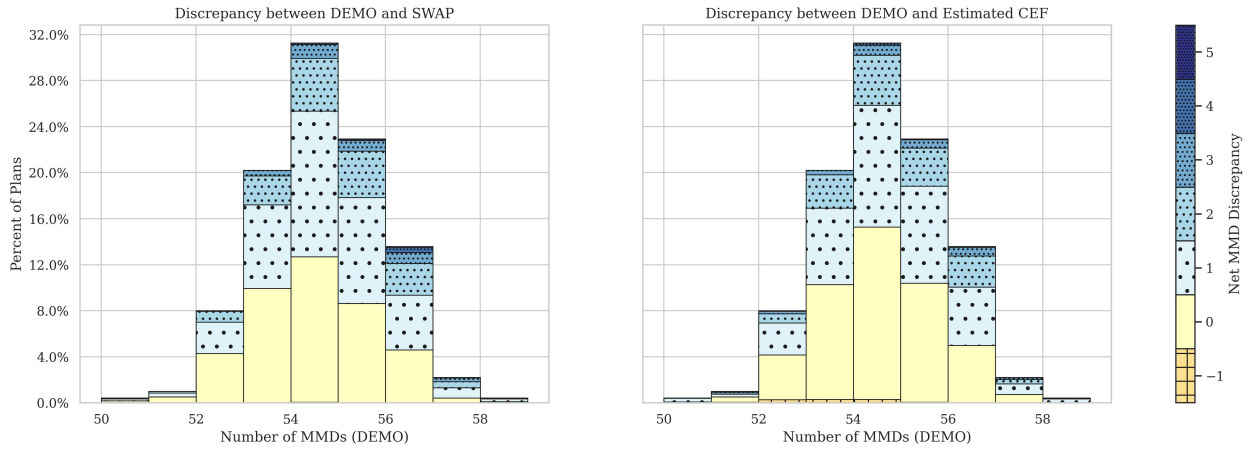


Figure 4.10: **MMD Discrepancies in Reconstructed CEF Data.** Distribution of MMD discrepancies for an ensemble using short bursts optimization to sample plans with many MMDs in DEMO. On the left, we display the MMD discrepancy between DEMO and SWAP. On the right, we display the MMD discrepancy between DEMO and our estimation of the 2010 CEF, as described in Appendix 4.7.

breaking up precincts would tend to strengthen the effects of the noise. On the other hand, the common practice of keeping cities and counties intact would tend to weaken those effects. Future work could use more sophisticated ensemble analyses to account for individual states' redistricting requirements [Becker et al., 2021, McCartan et al., 2022], or perhaps a dataset of hand-drawn plans.

In order to make the most of our limited computational resources, we chose to use precincts as the geounit from which our districts were composed, rather than blocks. This greatly reduced the computational complexity, allowing us to run more chains and study larger geographies. In practice, redistricters may split precincts between neighboring districts (though they often prefer not to).

To get some indication as to how using blocks would have affected our results, we computed the critical offset and no offset discrepancy rate for three small-enough state geographies using blocks as our base geounit (Table B.3 in Appendix B.1). We observe a notable increase in both metrics in all three geographies, indicating that our results may underestimate the impact of the new DAS on redistricting in practice. (On the other hand, keeping

cities and counties whole would have the opposite effect.) We hypothesize the difference is greatest where districts are very small, as in the three examined geographies. This aligns with guidance from the Census Bureau suggesting that data users should aggregate block-level data to increase the accuracy of their analyses.

Data Comparing the DEMO and SWAP datasets does not isolate the impacts of the 2020 DAS [boyd and Sarathy, 2022]. It reflects the impact of both the 2010 and 2020 DASes. This is inherent. Outside the Census Bureau, it is impossible to directly compare the public 2020 Redistricting Data and the confidential 2020 CEF. DEMO was created to aid stakeholders in understanding the effects of the 2020 DAS by comparing it to SWAP.

If we were somehow able to run our analyses using DEMO and the 2010 CEF, nothing in Section 4.5 would change except possibly in the last paragraph. By using only total population—unaffected by the swapping—we observe the impact of the 2020 DAS alone [Kenny et al., 2023].

On the other hand, Section 4.6 examines the BVAP as a fraction of the VAP in districts. In 2010, swapping did not affect VAP. But it could affect the BVAP if households with different BVAP were swapped between districts. This is an inherent limitation any work based on comparing the 2010 demonstration and redistricting data [United States Census Bureau, 2021].

One way to try to isolate the effect of the 2020 DAS would be to run it ourselves using an alternate dataset (e.g., reconstructed 2010 Census microdata derived from SWAP as in Cohen et al. [2022]). But running the 2020 DAS reliably is difficult [Ballesteros et al., 2025] and beyond the scope of the present work.

In Appendix 4.7, we instead take a step towards isolating the effect of the 2010 DAS, using data provided to us by the authors of Ballesteros et al. [2025]. More work would be needed to fully isolate the effects of the two disclosure avoidance systems. Still, based on our analysis in Section 4.6.3 and Appendix 4.7, we believe that any differences in the MMD

discrepancies arising from the two systems are more likely differences in degree than in kind.

Scope Other limitations stem from our scope, rather than our methods. Most obviously, we focused on state legislative redistricting, excluding both Congressional and sub-state redistricting, and on *Gingles 1*, excluding the other parts of the *Gingles* framework and racial-gerrymandering litigation more broadly. So while our results give us hope that discrepancies from disclosure avoidance are tolerable, more serious discrepancies may still lurk elsewhere. As a point of comparison, we provide a limited set of results for experiments on Congressional district geographies in Table B.4 in Appendix B.1. Future work could further study Congressional and sub-state redistricting, or extend Cohen et al. [2022] by quantifying discrepancies for the second and third *Gingles* tests.

Finally, we do not contend with the potential use of the Noisy Measurement Files (NMF) for redistricting (see below). While recent work has leveraged the NMF to analyze the error introduced by the 2020 DAS Kenny et al. [2024], it introduces novel difficulties when applied to redistricting. For example, since hierarchical consistency is not maintained, the population of a district as measured by the NMF can change depending on how its constituent geounits are aggregated. Addressing this issue and evaluating whether the NMF is suitable for redistricting uses is an avenue for future work.

Timeliness and Relevance The results presented in this chapter were published just months before the Supreme Court released its decision in *Louisiana v. Callais* and the Department of Commerce released its administrative order on noise infusion in the census. As a result of *Louisiana v. Callais*, claims of racial gerrymandering will be held to a much higher standard of evidence, likely resulting in significantly fewer cases being brought before federal courts. As a result of the administrative order, the disclosure avoidance system for the 2030 census will no longer be based on differential privacy, and the released tabulations will likely be consistent with the CEF (although it remains unclear what form the 2030 DAS

will take). We therefore do not expect the arguments we present to have much influence on relevant case law in the near future.

On the other hand, the evaluation framework we adopt can still be useful for researchers trying to better understand the performative aspects of differential privacy. In particular, it is necessary to model how differential privacy will alter the environment in which it is deployed in order to properly evaluate it. We also emphasize that the experimental framework we adopt is not only useful for the study of differential privacy, but also reasoning about other sources of noise and uncertainty. As we have explained, there are many sources of noise in census data, and similar questions to those raised in Alabama's lawsuit could be asked about phenomenon such as undercounts and overcounts.

4.9 Discussion

In its lawsuit challenging the 2020 DAS, Alabama argued that One-Person, One-Vote requires balancing populations according to the confidential CEF data, rather than the official Redistricting Data (*Alabama v. Department of Commerce*, 2021). In the face of discrepancies, Alabama concluded that redistricting plans created using the Redistricting Data could be illegal. As Alabama put it, this could potentially "inhibit" or "impede" the redistricting process (*Alabama v. Department of Commerce*, 2021).

What data should redistricting law treat as the ground truth: the data as collected, edited, and imputed (the CEF) or as published (the Redistricting Data)? Our experiments can't answer the question. Instead, they suggest that treating the CEF as the ground truth may not greatly affect a redistricter's ability to comply with the law using only the Redistricting data. At the very least, there exist countermeasures for our specific redistricting procedure. To the extent that the finding generalizes, data discrepancies may "impede" redistricters by giving them something else to account for, but appears unlikely to "inhibit" lawful redistricting altogether.

At least for state legislative redistricting, it appears that one can use the Redistricting Data to meet OPOV and MMD threshold requirements on the unseen data. Discrepancies do arise, but can be understood and accounted for. For OPOV requirements, using offsets does not make generating state legislative districts more difficult. In particular, all our ensembles in Section 4.6 used the critical offsets found in Section 4.5. And for counting MMDs, the common practice for producing illustrative maps is already to avoid very small margins.

We caution against over-interpreting our results as saying that disclosure avoidance can have no practical impacts on redistricting. Prior work highlights many plausible interactions Kenny et al. [2021], Cohen et al. [2022], Wright and Irimata [2021], only a few of which we explore. The perturbations from disclosure avoidance (or other sources of error) could conceivably make it harder to challenge a racially gerrymandered plan if, for example, a cracked plan in a racially polarized state appears to have more majority-minority districts in the DEMO data than it has "effective majority" districts in reality. Perhaps our probabilistic model of MMD discrepancies might help us reason about this possibility, but we leave that to future work.

Though we are not experts in this area of law, we think it is unlikely that changes in disclosure avoidance would cause courts to reject the long acknowledged "legal fiction" that the Decennial Census exactly reflects the underlying population. On the other hand, the case for statistical adjustment has only strengthened with the Census Bureau's recent release of the so-called Noisy Measurement File (NMF) United States Census Bureau [2023]. These are the raw, noisy statistics produced by the 2020 DAS before being post-processed into the required tabular form. By definition, the NMF is statistically more informative than any derived data products, including the Redistricting Data itself. It is conceivable that NMF-based statistical adjustments will find their place redistricting or policymaking. For the moment, this is mere speculation. The official Redistricting Data remains the only basis for redistricting today, to the best of our knowledge.

APPENDIX A

MULTI-ATTACK ROBUSTNESS

A.1 Additional Background and Setup Details

A.1.1 Representation Similarity Metrics

Canonical Correlation Analysis (CCA): Given activation matrices $X \in \mathbb{R}^{n \times p_1}, Y \in \mathbb{R}^{n \times p_2}$, canonical correlation analysis involves finding the orthogonal bases w_X^i, w_Y^i such that, after the matrices are projected onto their respective bases, correlation between the matrices is maximized [Kornblith et al., 2019]. The relevant summary statistic for CCA is $\bar{\rho}_{CCA}$, which is defined to be the mean of the canonical correlation coefficients ρ_i, for $1 \leq i \leq p_1$. ρ_i can be computed by

$$\begin{aligned} \rho_i = \max_{w_X^i, w_Y^i} & \operatorname{corr}(Xw_X^i, Yw_Y^i) \\ s.t. \quad & \forall_{j<i} Xw_X^i \perp Xw_X^j \\ & \forall_{j<i} Yw_Y^i \perp Yw_Y^j \end{aligned} \tag{A.1}$$

In order to improve the robustness of CCA, **Singular Vector Canonical Correlation Analysis (SVCCA)** has been proposed by Raghu et al. [2017]. Rather than compute CCA on X and Y, SVCCA first computes the singular vector decomposition of X and Y to find representations X' and Y' whose principle components explain a fixed proportion of the variance of X and Y. Standard CCA is then performed on X' and Y'.

Orthogonal Procrustes: The solution to the Orthogonal Procrustes problem is the left rotation of X that minimizes the Frobenius norm between the rotated X and Y. Formally, the problem is defined as

$$\min_R \|Y - RX\|_F^2, \; s.t. \; R^\intercal R = I \tag{A.2}$$

	Non-Robust Model		ℓ_∞-Robust Model			
	Benign Images		Benign Images		Adversarial Images	
Width	Train Acc.	Val. Acc.	Train Acc.	Val. Acc.	Train Acc.	Val. Acc.
1	99.68	91.23	89.21	82.39	50.90	42.92
2	99.97	93.44	97.73	84.91	66.91	44.02
3	99.99	94.18	99.72	85.53	80.78	43.74
4	100.00	94.33	99.98	85.46	89.64	44.13
5	100.00	94.66	100.00	85.97	93.42	45.38
6	100.00	94.86	100.00	86.40	95.79	45.70
7	100.00	94.59	100.00	86.87	97.38	46.79
8	100.00	94.50	100.00	86.90	98.24	47.15
9	100.00	94.98	100.00	87.39	98.68	48.11
10	100.00	94.92	100.00	87.59	98.90	48.89

Table A.1: **Performance of WideResNet Models.** Training and validation accuracies of the WRN-28-N models used in this chapter's experiments, where N is the width factor. Robust models were trained on the ℓ_∞ threat model with $\epsilon = \frac{8}{255}$.

The Orthogonal Procrustes distance between X and Y is defined as

$$d_{\text{Proc}}(X, Y) = \|X\|_F^2 + \|Y\|_F^2 - 2\|X^\intercal Y\|_*, \tag{A.3}$$

where $\|\cdot\|_*$ is the nuclear norm [Ding et al., 2021]. When X and Y are normalized, this distance metric lies in the range $[0, 2]$. For our purposes, we compute the Orthogonal Procrustes similarity between normalized matrices X and Y, defined to be

$$S_{\text{Proc}}(X, Y) = 2 - d_{\text{Proc}}(X, Y) \tag{A.4}$$

A.1.2 Related Work

Tools to probe neural networks. These can be sorted broadly into two categories: i) *sample-based* and ii) *distribution-based* tools. Sample-based tools include techniques like input saliency maps [Simonyan et al., 2013], Grad-CAM [Selvaraju et al., 2017], integrated

gradients [Sundararajan et al., 2017] etc. which are focused on interpreting the output of neural networks for specific inputs. On the other hand, loss surface visualizations [Yosinski et al., 2015, Nguyen and Hein, 2018, Yang et al., 2021] and representation similarity metrics [Dwivedi et al., 2020, Kornblith et al., 2019, Bansal et al., 2021, Raghu et al., 2017, Csiszárik et al., 2021] consider aggregate network properties derived from a set of samples.

Visualizing and validating network properties: Gilboa and Gur-Ari [2019] use activation atlases to argue that the learned representations of wider networks are 'more informative' than shallower ones. To validate this assertion, they fine-tune a linear layer for a new task and find the wider representations perform better. Raghu et al. [2021] study how transformers and CNNs learn features differently, Grigg et al. [2021] analyze the differences between supervised and self-supservised representations and Kornblith et al. [2021] consider the link between loss functions and feature transfer. Gavrikov and Keuper [2022] use convolutional filters to understand the impact of robust training while Ilyas et al. [2019] visualize what input features are needed for robust training.

A.2 Studying Benign Representations of Robust Networks?

In this section, we provide additional details on how the benign representations from robustly trained networks differ from those obtained from non-robust networks. We ablate across RS metrics (A.2.1), datasets (A.2.2), robust training methods (A.2.3) and architectures (A.2.4). We also delve deeper into the link between RS metrics and classification accuracy in A.2.5.

A.2.1 Using Other Representation Similarity Metrics

In Figure A.1, we plot the layerwise similarity between the layers for benign representations from both a non-robust and robust model. Due to computational constraints when computing SVCCA and Orthogonal Procrustes, comparisons are shown using ResNet18 models, instead of the Wide ResNet models used throughout much of the rest of the chapter. We note

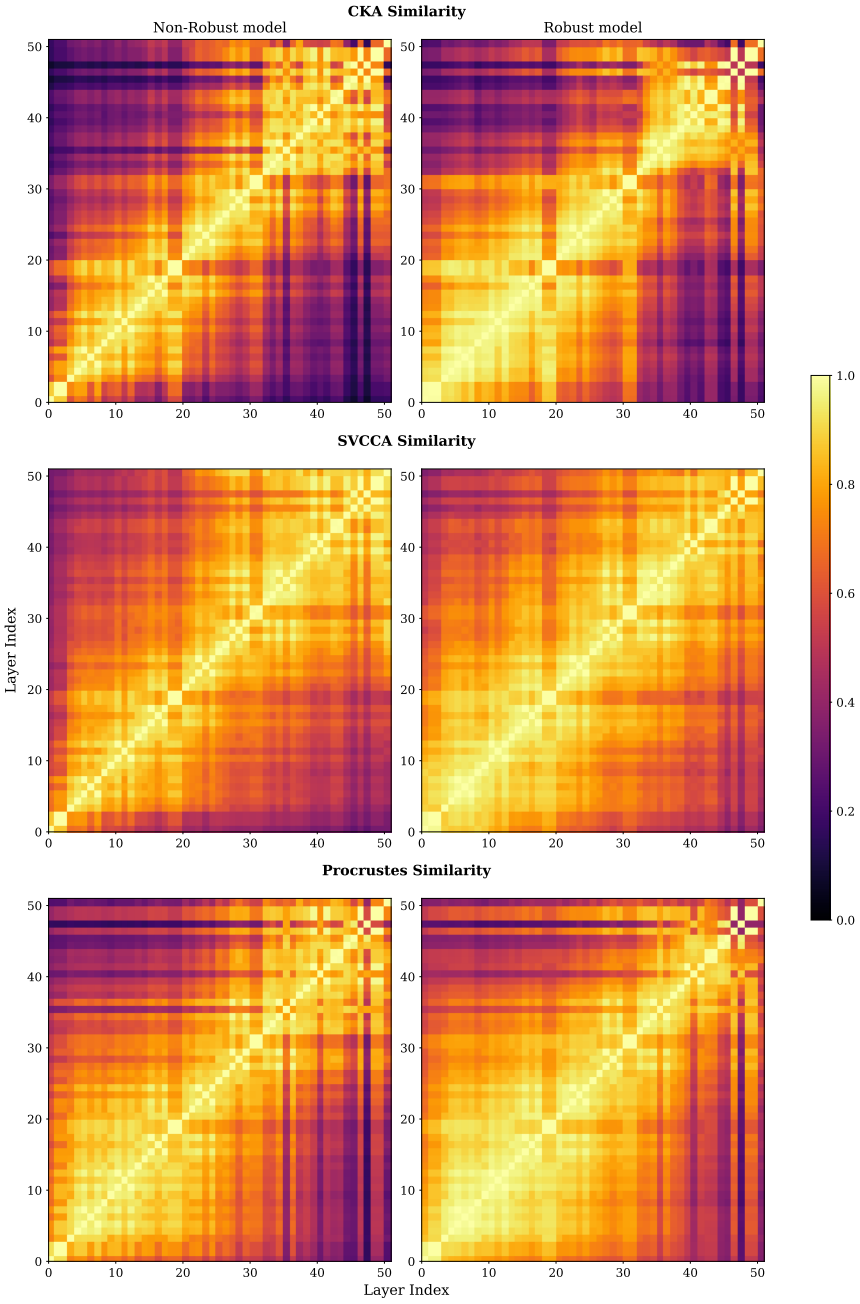
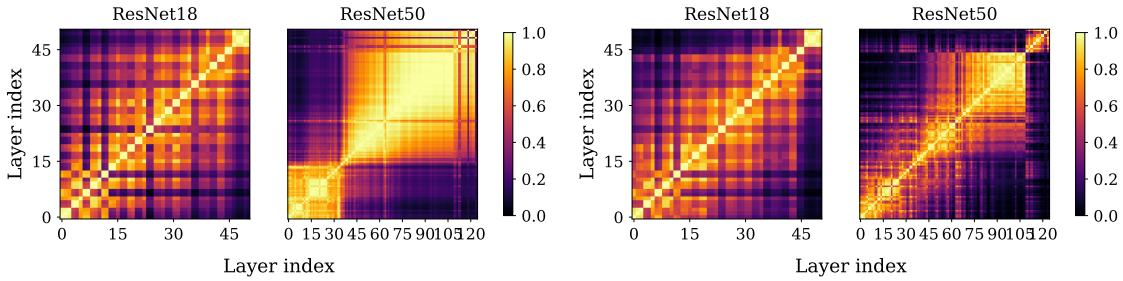


Figure A.1: **Layerwise Similarity Plots of Non-Robust and Robust ResNet18 Models**. 3 different RS metrics are used: CKA, SVCCA, and Orthogonal Procrustes similarity over benign representations.

that the trend we observe of increased cross-layer similarity for robustly trained networks appears regardless of the RS metric used. Further, CKA leads to the most visually distinct

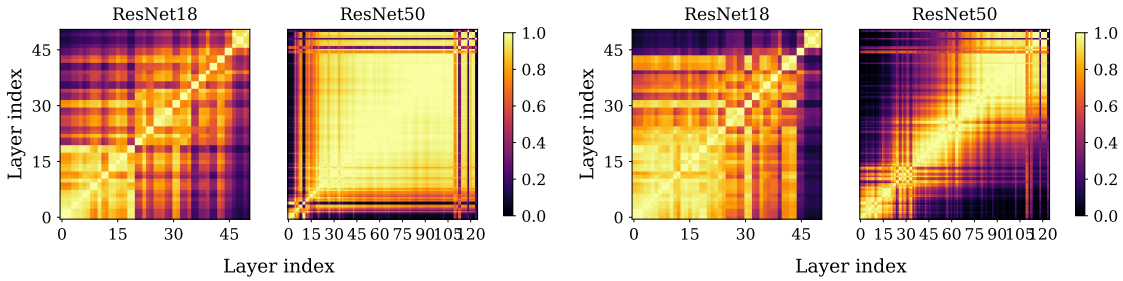
structure. The broad takeaways from the different RS metrics are aligned enough that we focus on CKA for the remainder of the results in this Appendix.

A.2.2 Using Other Datasets



(a) ImageNette (Easy) (b) ImageWoof (Hard)

Figure A.2: **CKA Similarity: Additional Datasets (Non-Robust Networks).** Cross layer cka for *non-robust* network (using benign features)



(a) ImageNette (Easy) (b) ImageWoof (Hard)

Figure A.3: **CKA Similarity: Additional Datasets (Robust Networks).** Cross layer cka for *robust* network (using benign features)

In Figures A.2 and A.3, we plot the layerwise similarities using CKA for ResNets trained on the ImageNette and Imagewoof datasets, both of which are subsets of the Imagenet dataset. These expand upon the results from Figure 2.2 of the main body, and show the stark differences between the representations obtained from non-robust and robust networks. The cross-layer similarities are far more pronounced for robust networks across datasets and architectures.

A.2.3 Comparing Across Robust Training Methods

In Fig. A.4, we compare the benign representations from models robustly trained using two different robust training techniques: PGD-based adversarial training and TRADES. Both methods lead to very similar robust representations, and exhibit the same long-range similarity across layers.

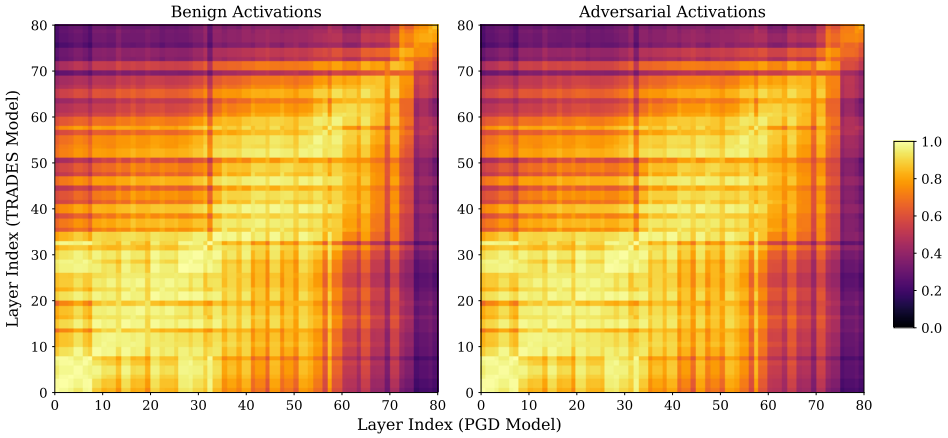
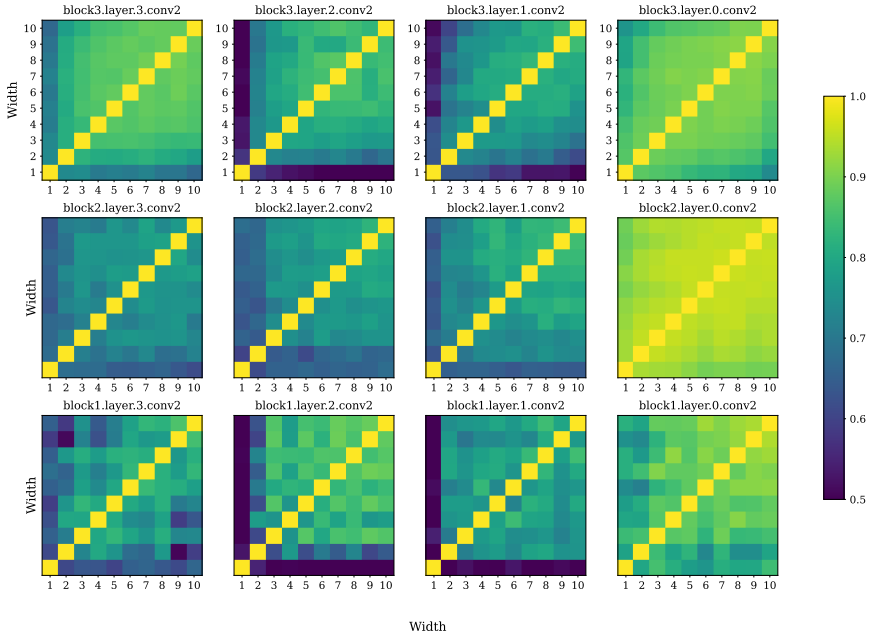


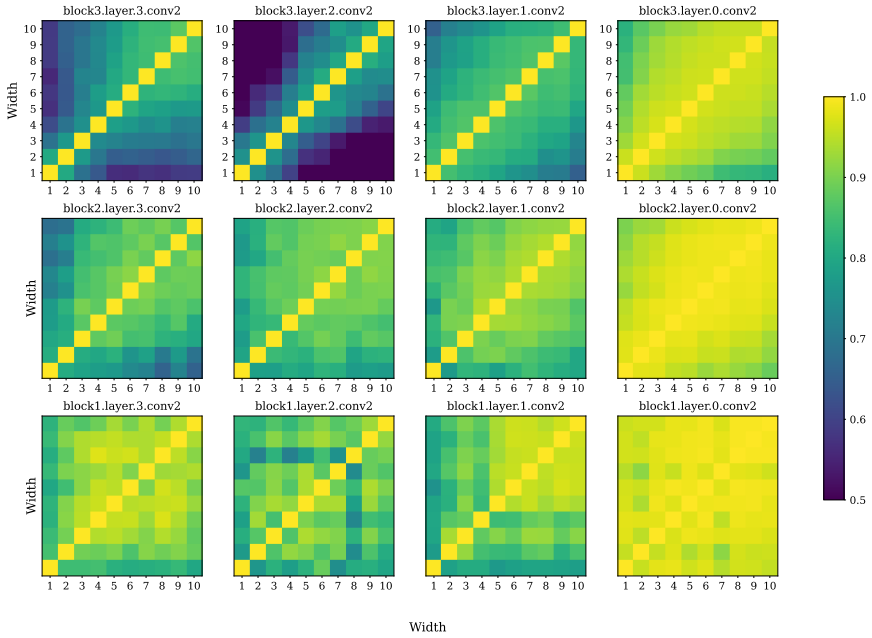
Figure A.4: **Adversarial Training and TRADES Produce Similar Final Representations.** Layerwise similarity between benign activations of a TRADES-trained WRN-28-5 model and a PGD-trained WRN-28-5 model. The structure of this plot is similar to that of Figure 2.2.

A.2.4 Comparing Across Architectures

Recent work [Huang et al., 2021] has studied the influence of depth and width on adversarial robustness. We extend this investigation by comparing the benign representations obtained from Wide-ResNet models trained with different widths in Figure A.5 with both non-robust and robust training. A few key observations stand out: 1) early layers in different blocks are very similar across different widths for robust training, as compared to benign training, 2) increasing width only leads to large variations in learned representations in the last block for robust training, and 3) benign networks of different widths are similar deeper in the network, and different early on, with the opposite trend for robust networks. Taken together, these



(a) *Non-robust* networks. Features extracted over *benign* images.



(b) *Robust* networks. Features extracted over *benign* images.

Figure A.5: **Cross-Model CKA Between Wide ResNet Models of Different Widths**. Data is presented across 12 convolutional layers, starting from the last convolutional layer, i.e., block3.layer3.conv2 and moving backward in the network. Robust networks have a high degree of similarity across widths in earlier layers but large variation in deeper layers, which is the opposite trend to that observed for non-robust networks.

results suggest that while increased width early on in the network may not be particularly useful, extra width deeper in the network allows for more expressive representations to be learned. This phenomenon could be used to aid in the design of robust architectures.

A.2.5 Delving Deeper Into CKA

Class-aware representation similarity. Standard RS metrics don't consider data labels, thus similarity scores doesn't provide insights in how individual class data impacts it. As an example, we consider Linear CKA (Equation 2.4), where the similarity is dependent on the dot-product of vectorized Gram matrices $(\boldsymbol{K}, \boldsymbol{L})$. Two factors contribute to the similarity of gram matrices: 1) Intra-class similarity (C_1): $\sum_{i,j,y_i=y_j} K_{ij}L_{ij}$, 2) Inter-class similarity (C_2): $\sum_{i,j,y_i\neq y_j} K_{ij}L_{ij}$. When measured between identical features, linear CKA is 1, i.e., $C_1/(C_1+C_2)+C_2/(C_1+C_2)$. Under a near uniform distribution of class labels, only a small fraction of entries in gram matrices contribute to intra-class similarity (C_1), as there are a lot more cross terms in the CKA computation. However, when experimentally measuring the influence of both terms, i.e., we find that the contribution of intra-class CKA is very high, and it increases as we go deeper into the network. This is likely because separability of feature improves deeper in the network with similar class features being clustered together, which naturally increases the magnitude of C_1 in the similarity score.

We hypothesized that robust networks would have lower intra-class similarity as compared to non-robust networks, which would explain their lower classification accuracy on benign data. While this holds for the CIFAR-10 and ImageNette, for Imagewoof, robust networks have a much higher intra-class contribution. We leave a detailed investigation for future work.

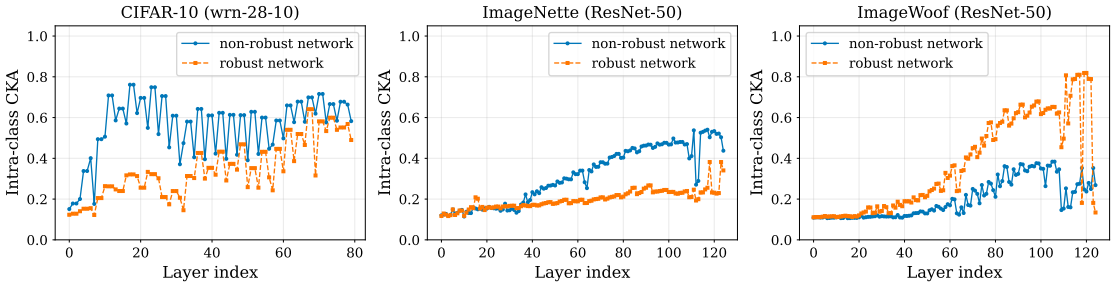


Figure A.6: **Intra-class Similarity.** We find that the contribution of intra-class similarity, measured by intra-class CKA, increases to non-trivial fraction, as we go deeper into the network.

A.3 Adversarially Perturbed Representations

In this section, we add further results to back up the insights about the impact of adversarially perturbed data on resulting representations from Section 2.4.

A.3.1 Comparing Benign and Perturbed Representations

In Figure A.7, we can see, across, architectures, benign and perturbed representations differ at all layers except the first few for non-robust networks. In addition, for robustly trained networks, increasing width increases the divergence between benign and perturbed representations. This may indicate the presence of increased capacity to learn different representations for benign and adversarial inputs.

Across different threat models (Figure A.9), we find robustly trained networks learn aligned benign and perturbed representations. This indicates that, in general, for most threat models, robust training is effective at ensuring that perturbed representations do not differ too significantly from benign ones, unlike for non-robust networks.

A.3.2 Adversarial Representations of Non-Robust Networks

When analyzing the architectural impacts on learned representations using benign data, Nguyen et al. [2020] note the emergence of a block structure with increased width and depth.

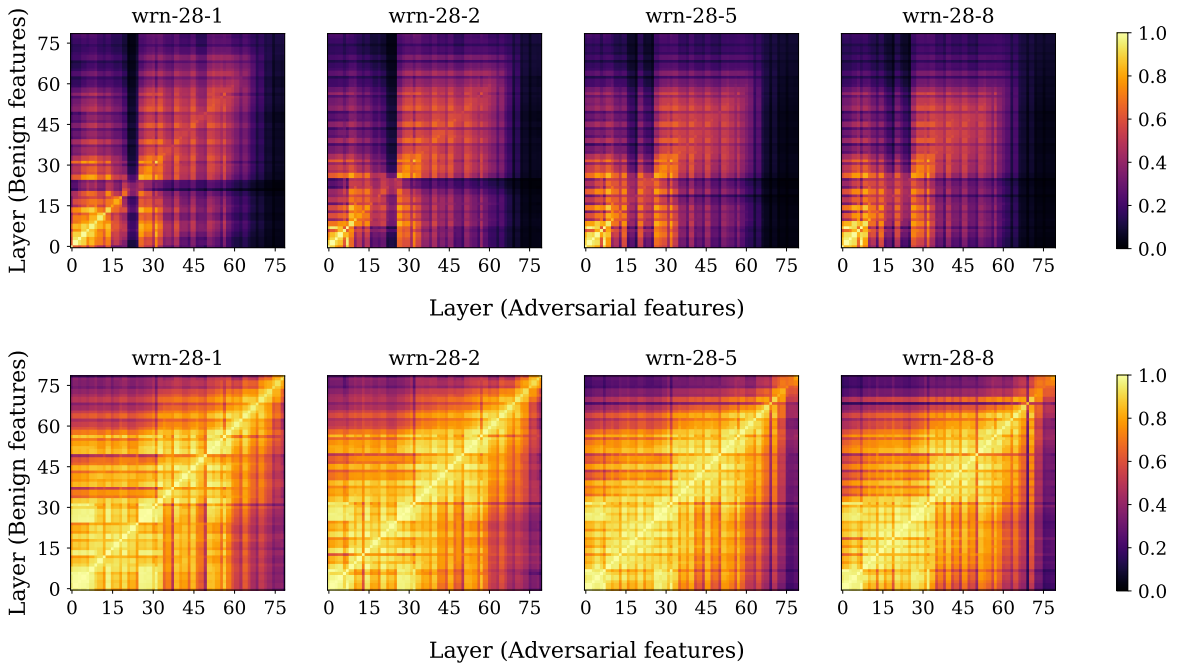


Figure A.7: **Adversarial Examples Cause Representation Drift in Later Layers.** In the similarity plots of non-robust network representations (*top row*), adversarial and benign features display a similar structure to plots comparing just benign representations of non-robust models only in the early layers. In later layers, adversarial representations are very dissimilar from the benign representation of all layers. The similarity plots of robust networks (*bottom row*) are quite similar to plots comparing just benign representations of robust models, suggesting that robust networks learn similar representations for benign and adversarial data.

We find that a much stronger block structure emerges even for low-width networks when using adversarial inputs (Fig. A.10). This implies that perturbations force representations within a residual block to have higher similarity than outside the block. The block structure has been linked to the capacity of models, with Nguyen et al. [2020] noting that the 'block structure arises in models that are heavily overparametrized relative to the training dataset'. In this light, adversarial examples have a peculiar impact on the representations of benign models, since the block structure is more clearly visible for adversarial inputs, but for models which do not classify the inputs correctly. This indicates that local similarity is being exhibited and is a different phenomenon compared to block structure emergence for well-trained models.

Figure A.11 shows the stark difference between perturbed features for robust and non-

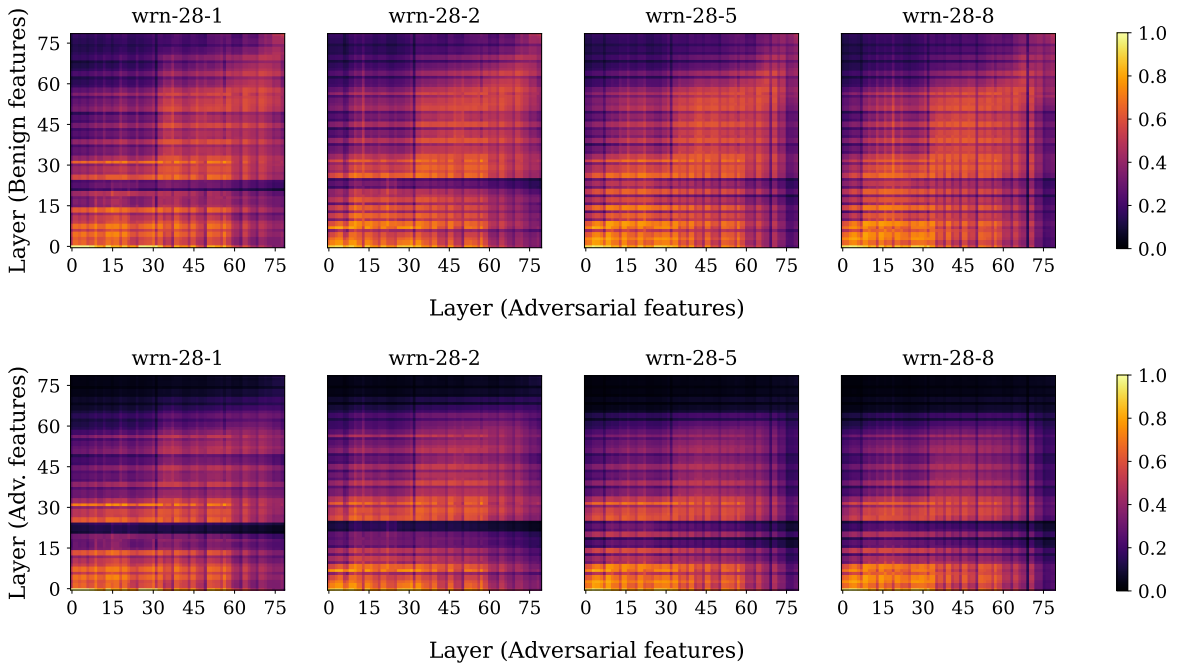


Figure A.8: **Non-Robust and Robust Models Differ Substantially Across Layers for Both Benign and Perturbed Inputs.** Cross layer CKA between a non-robust model (Y-axis) and robust model (X-axis) using perturbed and non-perturbed features.

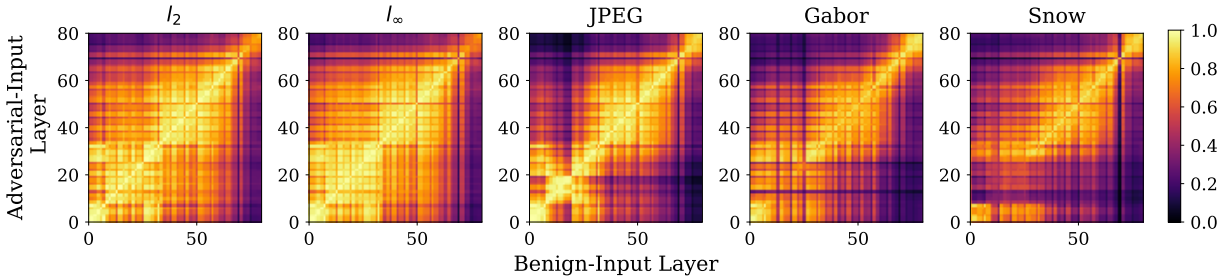


Figure A.9: **Robustly Trained Models Across Threat Models Have Aligned Benign and Perturbed Representations.** Except the 'Snow' threat model, all the others have identical benign and perturbed representations across layers.

robust networks. This is similar to the lack of similarity for benign inputs to these networks.

A.4 Evolution over Training

In this section, we study the similarity between the activations produced by models after each epoch of training. We present our results in two formats, time series and heatmaps.

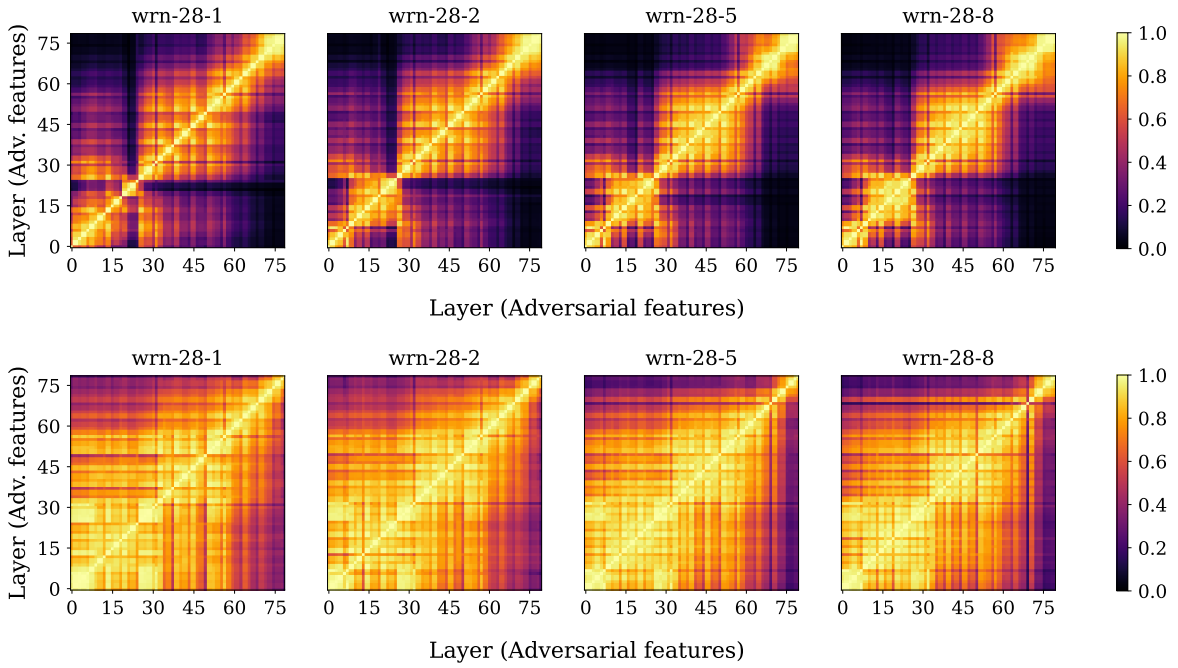


Figure A.10: **Adversarial Examples Have Different Impacts on the Representations of Benign and Robust Models.** In benign models (top row), the layerwise similarity between adversarial activations displays a similar structure to that of benign activations, except the layers with dissimilar benign activations appear to have even more dissimilar robust activations. This implies significant changes in adversarial activations are taking place in between the self similar layer "blocks". This pattern is largely absent from robust models (bottom row), which display similar patterns for both benign and adversarial inputs.

For ease of understanding, the time series corresponds to a single row of the heatmap, where the representation being compared to is held constant.

A.4.1 Impact of Model Capacity

In Figure A.12, we plot the pairwise similarity between the representations of a single layer after each epoch of training. Natural training behaves as we would expect it to: a layer's representations at a given epoch are most similar to those from nearby epochs, and similarity degrades as the distance between epochs increases. Higher average similarity between later epochs implies convergence to a final learned representation. This pattern is also observed in the robust training of low-capacity networks, but as capacity increases starkly different

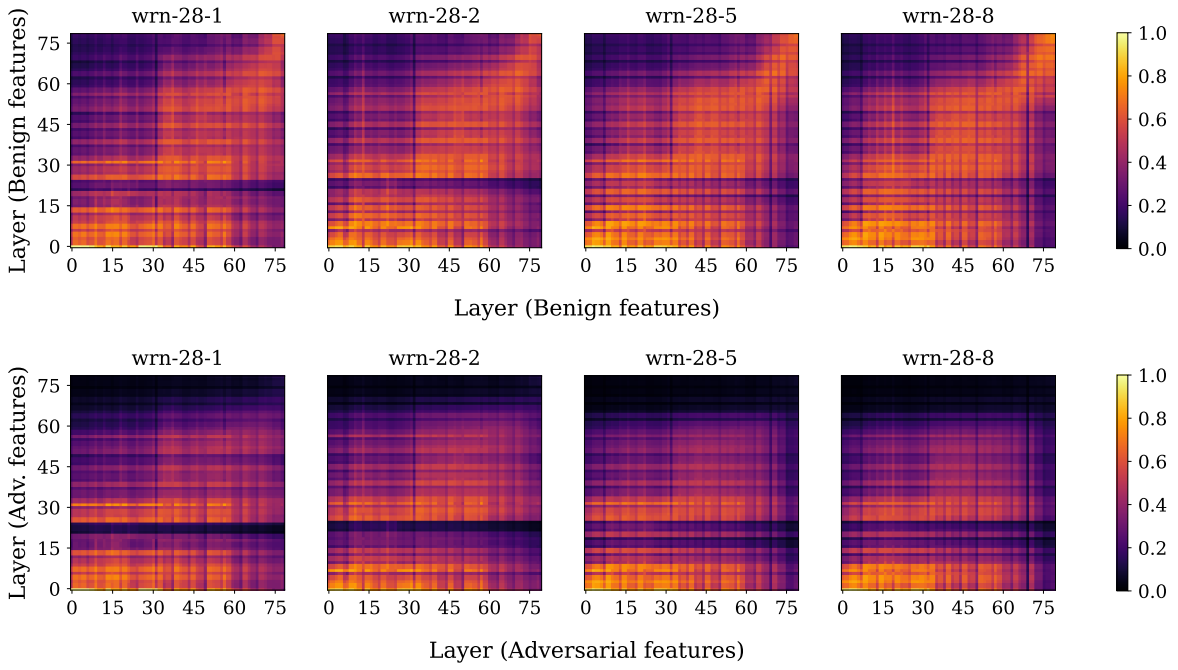


Figure A.11: **Perturbed Representations for Robust and Non-Robust Networks Differ Across All Layers.** Cross layer CKA between a non-robust model (y-axis) and robust model (x-axis) using perturbed features.

dynamics emerge. In high-capacity robust networks, we observe similarity start to decrease after about 30 epochs of training, only to increase again later on. This pattern is observed for both benign and adversarial activations.

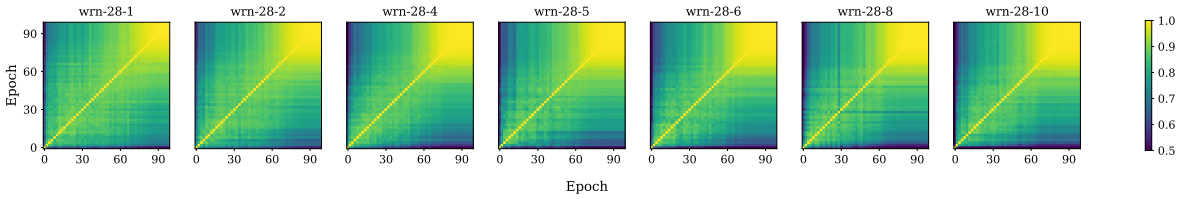
A.4.2 Adversarial Training and Overfitting

In Figure A.13, we plot the similarity to the final learned representations of the representations extracted from each of the three Wide ResNet blocks (block1, block2, block3) and the final average pooling layer before classification (avgpool) after each epoch of training. We also plot the validation loss of each model for comparison. We find that for models trained at $\epsilon = \frac{8}{255}$, instability in the later layers is not observed until the model begins to overfit on adversarial examples. At high perturbation strengths, training instability is observed at earlier layers and earlier in training. Training of the lowest capacity model is stable at all

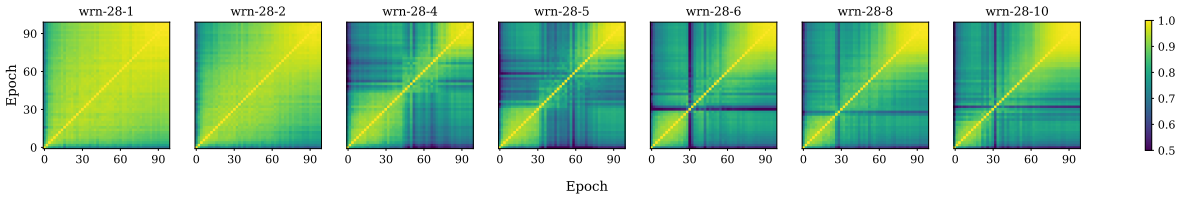
perturbation strengths, a trend also observed in Figure A.12.

A.4.3 Different Training Methods

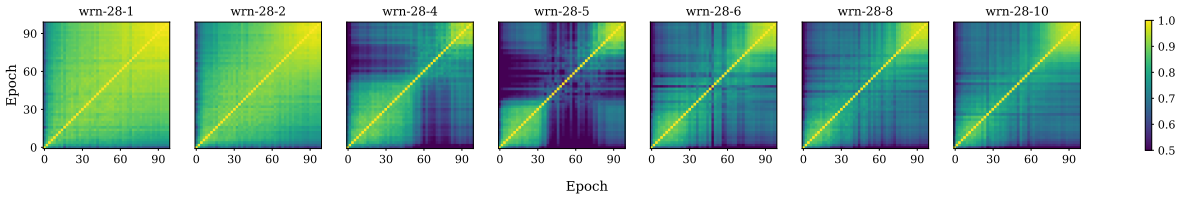
We also sought to test whether more principled robust training methods, such as those proposed by Zhang et al. [2019], would affect the patterns in similarity we observed in this section. In Figure A.13, we compare non-robust and robust training in a similar manner to Figure 2.6. The patterns we observed with PGD training disappeared when training with TRADES, highlighting the sensitivity of CKA to meaningful changes in training dynamics.



(a) *Non-robust* networks. Features extracted over *benign* images.

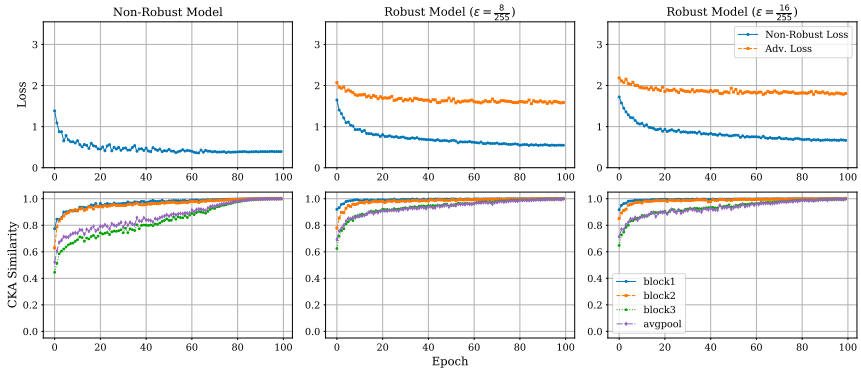


(b) *Robust* networks. Features extracted over *benign* images.

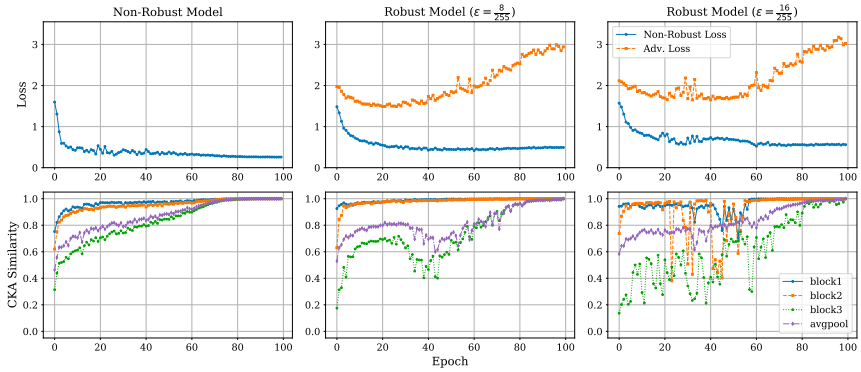


(c) *Robust* networks. Features extracted over *adversarial* images.

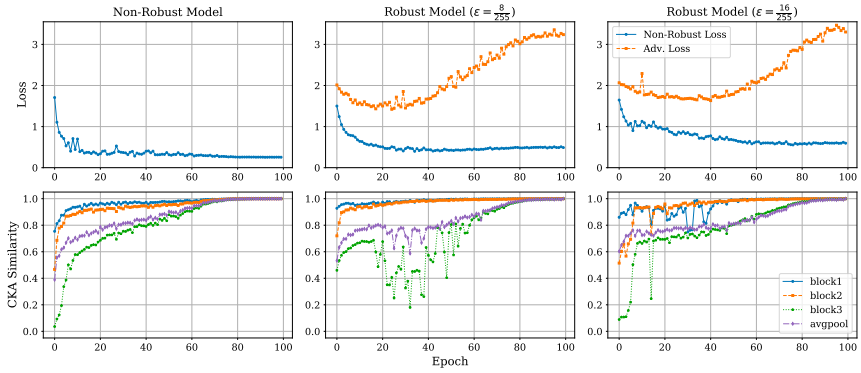
Figure A.12: **Stability of Robust Training Is Impacted Greatly by Model Capacity.** Each plot shows the CKA similarity between the activations of the final layer of a Wide ResNet feature extractor at each epoch of its 100-epoch training. While the stability of non-robust training is largely unaffected by capacity, the robust training of models above a certain capacity causes representations to be learned in the middle of training that are starkly different from the initial and final representations. This effect is even more pronounced when comparing adversarial representations.



(a) WRN-28-1



(b) WRN-28-5



(c) WRN-28-10

Figure A.13: **Overfitting and Training Instability in Robust Learning.** Each plot shows how benign representations of selected layers of Wide ResNet models compare to their final learned representations after each epoch of training. In larger robust models trained at $\epsilon = \frac{8}{255}$, training instability begins after the point of overfitting on adversarial data. However, when the perturbation strength is increased, there are large changes in the similarities of representations of subsequent epochs starting very early in training. High perturbation strengths also induce instability at earlier layers, which is not observed at lower strengths or in non-robust training.

A.5 Threat Models

In this section, we further investigate the impact of threat model on learned representations.

A.5.1 Impact of Budget Within a Threat Model

In Figure A.14, we show that increasing the budget and reducing width both lead to a decrease in relative capacity of the model, inducing higher redundancy of learned representations. In Figure A.15, we show that even within a single attack, changing the budget leads to drastic changes in the representations of deeper layers.

A.5.2 Aligning Representations Across Threat Models

Figures A.16, A.17, and A.18 ablate on the results of Figure 2.7 by varying the perturbation size of each threat model used during training. As expected, models at lower budgets tend to be better aligned. At larger budgets, even for visually similar attacks like JPEG and ℓ_2, the aligment dips, particularly at deeper layers. This indicates that to train models jointly robust to different threat models, explicit constraints on learned representations during training may be needed.

A.6 Additional CRT Experimental Setup Details

Additional training details. For initial training, we start with a learning rate of 0.1 and then use the multistep learning rate scheduling proposed by Gowal et al. [2020]; specifically, we scale the learning rate down by a factor of 10 halfway and 3/4 of the way through initial training or fine-tuning. For fine-tuning, we maintain a learning rate of 0.001. We train with SGD with momentum of 0.9 and weight decay of 0.0005.

Additional Attack parameters in training. Following other works on adversarial robustness, we use a step size of 0.075 for ℓ_2 attacks on CIFAR-10, 0.15 for ℓ_2 attacks on

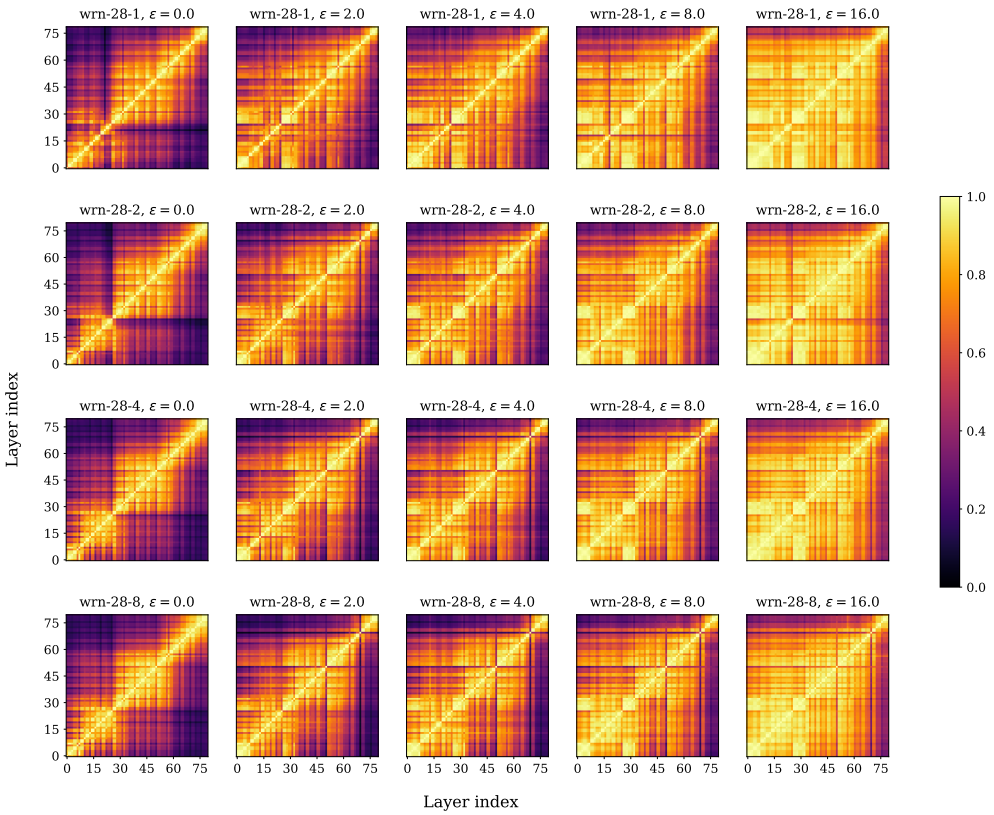
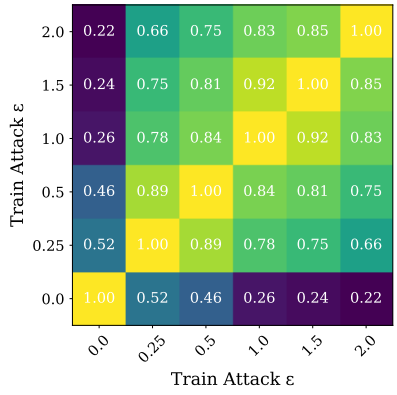


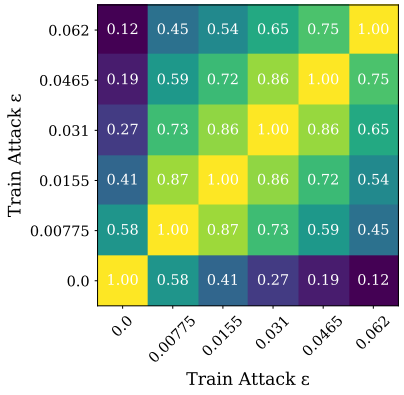
Figure A.14: **Delving Deeper Into Cross-Layer Similarity.** We observe that increasing the perturbation budget leads to an increase in cross-layer similarity, as does reducing the width. Both factors thus affect the relative capacity of the model.

ImageNette, and $\frac{2}{255}$ for ℓ_∞ attacks. For other attacks, we use $\frac{\epsilon}{8}$ where ϵ is the attack strength as the step size during training.

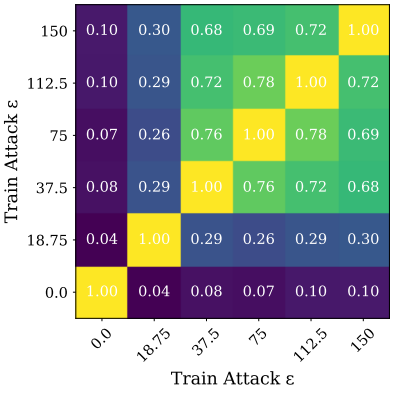
Model selection. In the experimental results we present, we perform evaluation using the epoch at which the model has the best performance measured across known attack types. Specifically, after each epoch of training, we evaluate the performance of each model against the attacks used during training (with the same attack parameters as used during training). For training with AVG, we use the best performing model with respect to the AVG objective (which is the model with the best performance measured as an average over individual attack accuracies). Meanwhile for MAX and FT MAX, we use the best performing model with respect to the MAX objective (which is the best performing model across the union of



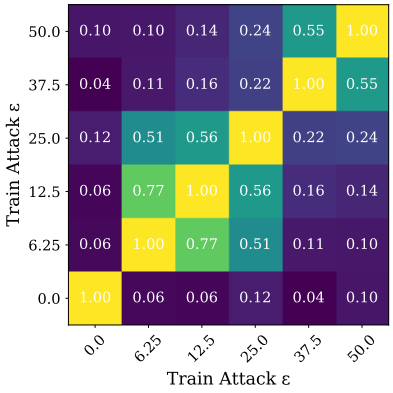
(a) ℓ_2



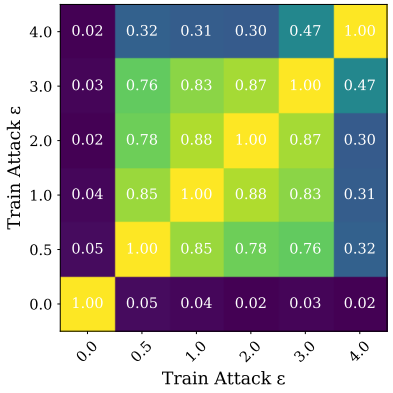
(b) ℓ_∞



(c) Gabor



(d) JPEG



(e) Snow

Figure A.15: **Adversarial Training Induces Very Different Representations, Even at Low Attack Strengths.** Each plot displays the CKA similarity between activations of the final layer of WRN-28-10 feature extractors adversarially trained on different strengths of each threat model. In each plot, even the representations of the lowest strength robust model are quite dissimilar from those of the benign model, with only the ℓ_p bounded models having that similarity be relatively close to the lowest similarities between robust models.

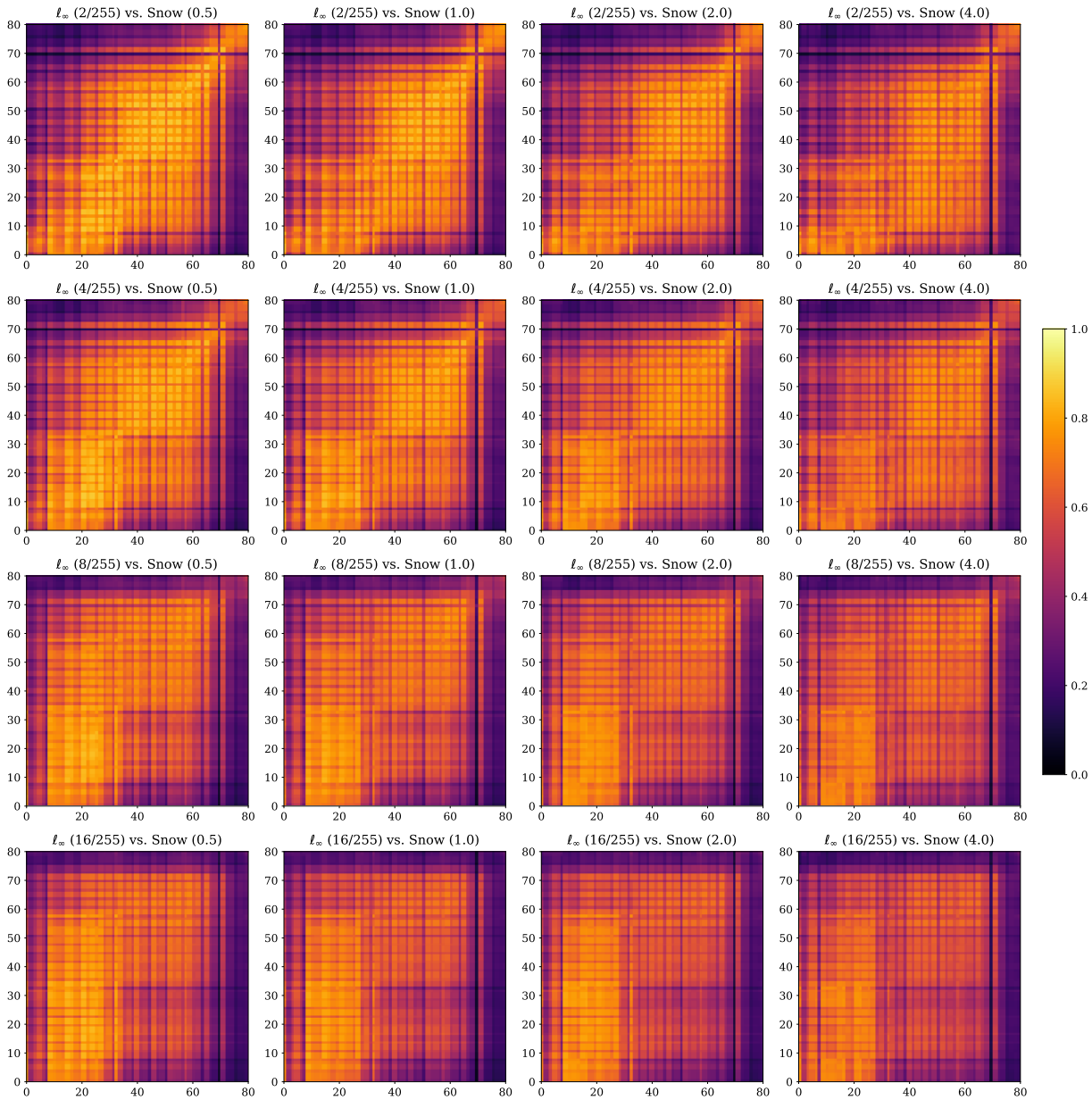


Figure A.16: ℓ_∞ **vs. Snow Representations.** Comparing representations of l_∞ and Snow robust models at different attack strengths.

all attacks). For procedures that only use a single attack per batch during training (Random, FT Single, FT Croce, and our procedure), we use the best performing model measured by sampling attacks per batch randomly.

Regularization setup. We note that all attacks used in this chapter use a gradient

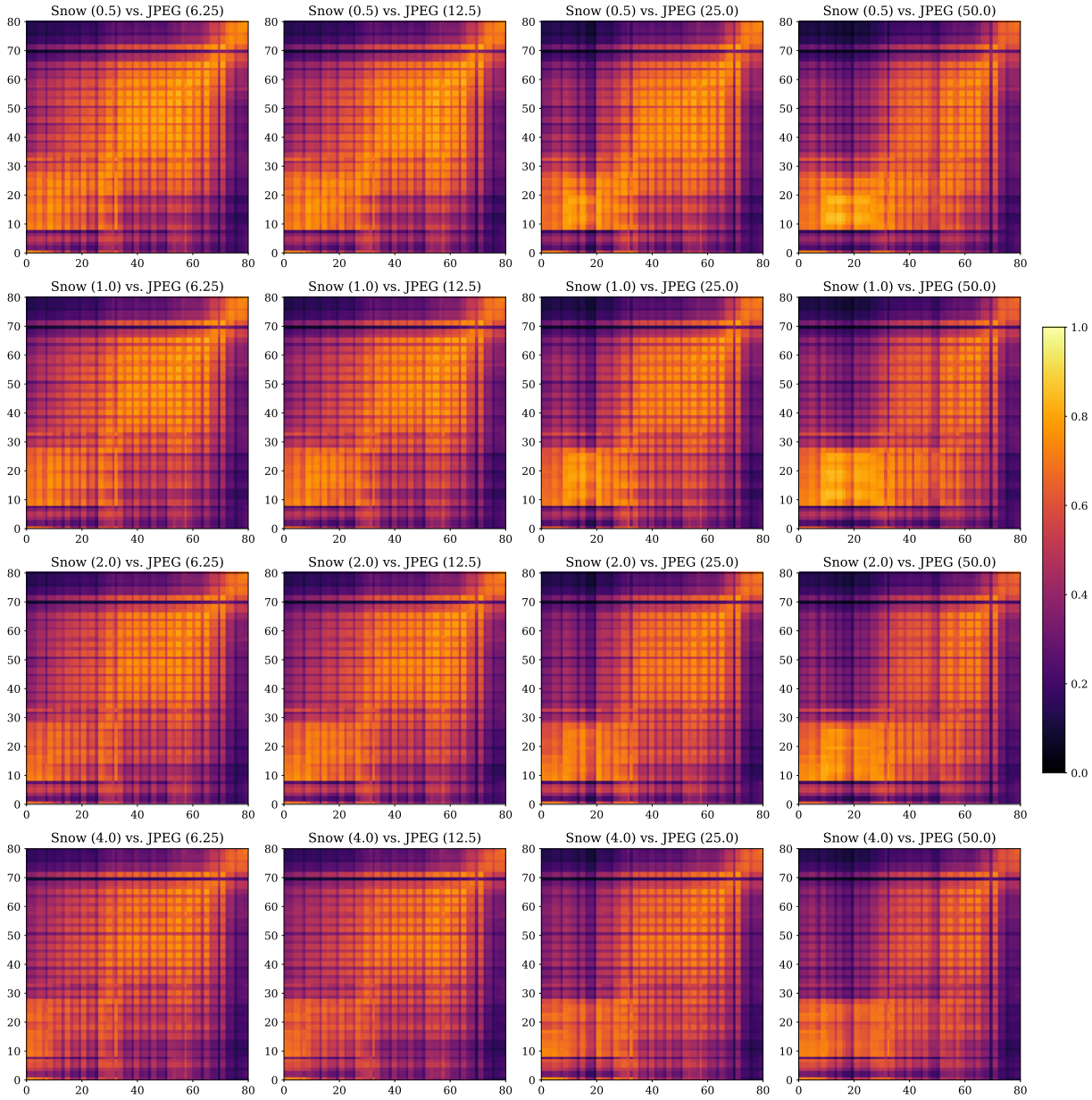


Figure A.17: **Snow vs. JPEG Representations.** Comparing representations of Snow and JPEG robust models at different attack strengths.

based optimization scheme for finding the attack. In order to compute regularization for non-ℓ_p threat models, we follow the same optimization scheme used by the attack [Xiao et al., 2018, Laidlaw and Feizi, 2019, Kaufmann et al., 2019] but replace the classification loss portion of the optimization objective to be the ℓ_2 distance between features/logits between

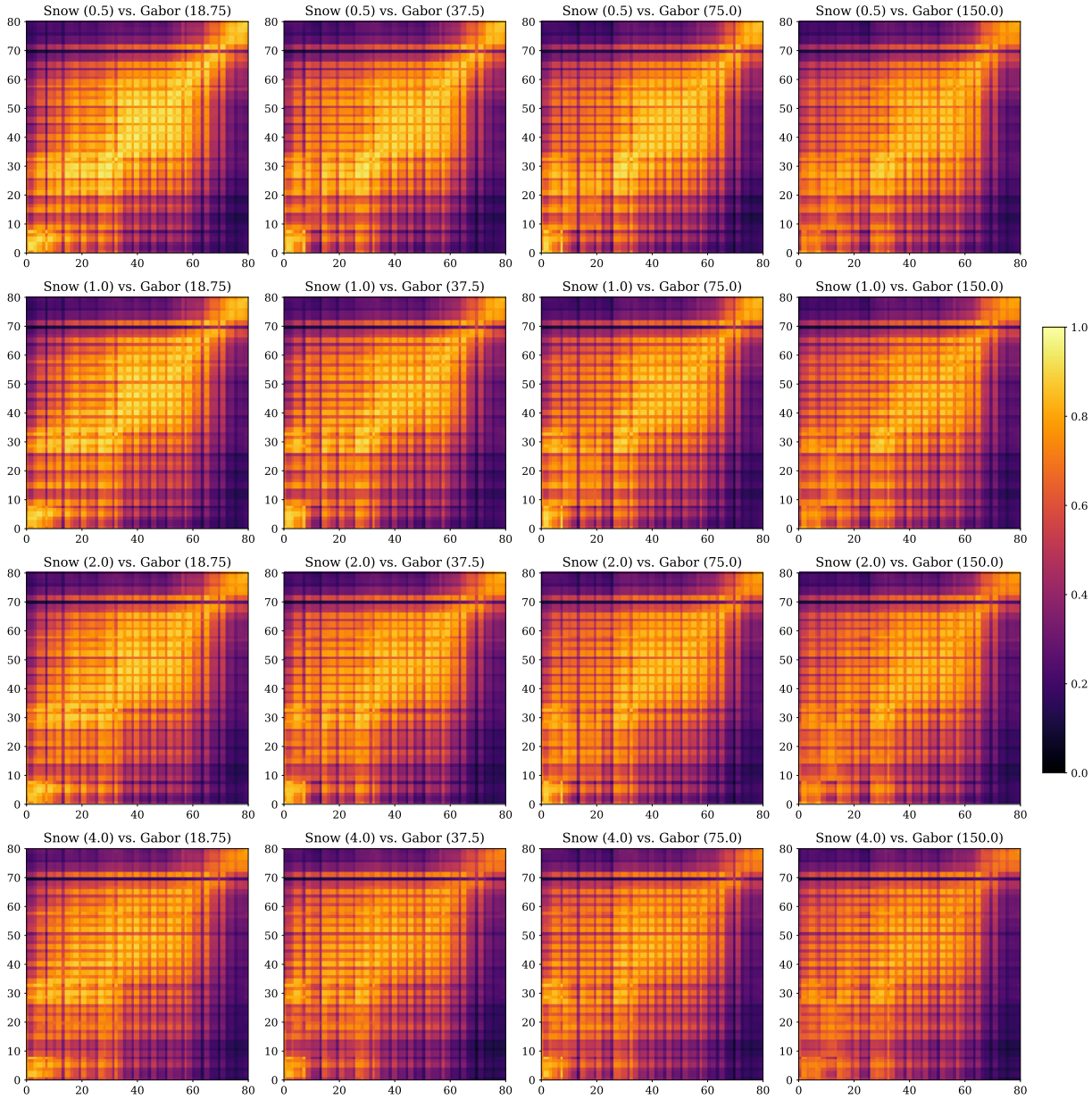


Figure A.18: **Snow vs. Gabor Representations.** Comparing representations of Snow and Gabor robust models at different attack strengths.

the perturbed and unperturbed input. For fine-tuning with regularization, since Croce and Hein [2022]'s fine-tuning approach only uses a single attack per batch, we structure the regularization to mimic Croce and Hein [2022]'s fine-tuning procedure. Specifically, for each batch, the regularization is for a single attack type (the same one which is selected to use

with adversarial training by Croce and Hein [2022]'s fine-tuning approach). This helps to reduce the overhead from regularization.

A.7 Additional Experimental Results for CAR

We present additional results for CAR on CIFAR-10 in Tables A.2 and A.3, results for CAR on Imagenette in Table A.4 and A.5 and results for CIFAR-100 in Tables A.6 and A.7. We also compare different fine-tuning approaches in the absense of regularization.

Training time and robust performance. We find that fine-tuning with MAX objective (FT MAX) or Croce and Hein [2022] (FT Croce) can generally achieve robustness across previous attacks and the new attack in the sequence comparable to training from scratch. For example, in Table A.3, when fine-tuning to gain robustness against StAdv attack starting from a model initially trained with adversarial training on ℓ_∞ attacks on CIFAR-10, we find that FT MAX achieves 50.75% average robustness across the two attacks and 41.57% union robustness across the two attacks, and FT Croce achieves 49.48% average robustness and 29.69% union robustness. These values lie within (or even above) the range obtained through training from scratch (42.23%-49.61% average robustness and 28.03%-40.8% union robustness). We find that this trend generally holds as well across time steps when new attacks are introduced, when using a different sequence ordering (Table A.2).

Of these two techniques, we find that FT MAX generally achieves higher average and union accuracies across the set of known attacks, but is less efficient when used in fine-tuning. For example, In Table A.3, FT MAX takes 3.99 hours for 10 epochs of fine-tuning from an ℓ_∞ robust model while FT Croce takes 2.31 hours. The time complexity of FT MAX also scales as the number of attacks increases, leading to 7.9 hours of fine-tuning for 10 epochs when there are 4 known attacks while FT Croce maintains approximately the same training time.

In comparison to naively training from scratch, we also find that these fine-tuning tech-

niques can be much more efficient. For example, a model robust to a sequence of 4 attacks in Table A.2 can be found in roughly 17 hours using CRT, but training from scratch each time would require 44 hours cumulatively.

Importance of replay. We find that replay of previous attacks is important for achieving good robustness across the set of known attacks when training with CRT. Fine-tuning with only the new attack (FT Single) usually leads to rapid forgetting of the previous attack. For example, in Table A.3 we observe that the accuracy of robustness on the initial attack (ℓ_∞) drops to 31.14% robust accuracy at time step 1 (from the initial accuracy of 51.49% at time step 0) and then further drops to 25.27% at time step 2 when the third attack (Recolor) is introduced. This forgetting is independent from tradeoffs between attacks as we find that training from scratch and FT MAX and FT Croce techniques can all achieve at least 40% ℓ_∞ accuracy at time step 1 and at least 35% ℓ_∞ accuracy at time step 2. The forgetting of previous attacks is also analogous to catastrophic forgetting of previous tasks in continual learning [Wang et al., 2023a, McCloskey and Cohen, 1989]. We note however that forgetting is less of a limitation in CAR than in continual learning since the defender's knowledge set only grows over time; they do not forget the formulation of previous attacks and can thus can always use methods such as replay.

ALR applied on logits vs features. In Table A.2, we also provide results for using regularization based on distances in the feature space (before the final linear layer), which are labelled with "+ ALR feature". Overall we observe that using regularization in the feature space can also help improve performance on average and union robustness across known attacks as well as improve unforeseen robustness over baselines. However, we observe that feature space regularization leads to larger tradeoffs in clean accuracy than regularization on the logits ("+ ALR" rows) while robust performance is comparable to regularization applied on the logits.

Training durations. Across all tables we also provide experiments for fine-tuning with

25 epochs (as opposed to 10 epochs reported in the main body). We find that increasing the number of fine-tuning epochs can help methods such as FT Croce achieve robustness closer to that of training from scratch, but at the cost of increased time for updating the model.

Performance on other datasets. We find that the gain in performance through using ALR varies across datasets. For Imagenette the gain in performance is generally much smaller than on CIFAR-10 (ALR closes the gap between fine-tuning based updates and training from scratch rather than surpassing training from scratch as in CIFAR-10. On CIFAR-100 ALR generally does not improve performance over fine-tuning. We believe that this is because achieving robustness on multiple attacks is quite hard on CIFAR-10; clean accuracy is between 60-70% and robust accuracies are even lower with StAdv and ℓ_∞ robustness only achieving up to 32% robust accuracy and 25% robust accuracy respectively.

Time Step	Procedure	Threat Models	Clean	ℓ_2	StAdv	ℓ_∞	Recolor	Avg (known)	Union (known)	Avg (all)	Union (all)	Time (hrs)
	AT	ℓ_2	**91.17**	69.7	2.08	28.41	44.94	69.7	69.7	36.28	1.24	8.35
0	AT + ALR ($\lambda = 1$)	ℓ_2	89.43	**69.84**	**48.23**	**34.00**	**65.46**	**69.84**	**69.84**	**54.38**	**31.27**	11.15
	AT + ALR feature ($\lambda = 5$)	ℓ_2	83.7	63.1	26.57	31.6	62.53	63.1	63.1	45.95	20.16	11.13
	AVG	ℓ_2, StAdv	87.74	62.17	50.92	17.17	45.47	56.55	47.55	43.93	15.92	23.72
	MAX	ℓ_2, StAdv	86.18	58.65	57.21	11.21	43.07	57.93	51.72	42.54	11.03	23.69
	Random	ℓ_2, StAdv	84.91	57.77	59.74	14.05	44.88	58.76	52.15	44.11	13.68	10.92
	FT MAX (10 ep)	ℓ_2, StAdv	83.73	57.07	58.67	12.51	49.03	57.87	51.32	44.32	12.36	4
	FT MAX (25 ep)	ℓ_2, StAdv	84.85	56.44	61.34	10.35	48.08	58.89	52.52	44.05	10.24	10
	FT Croce (10 ep)	ℓ_2, StAdv	84.7	57.88	54.27	14.38	51.08	56.07	48.13	44.4	13.8	2.4
	FT Croce (25 ep)	ℓ_2, StAdv	86.24	58.94	57.37	13.26	50.36	58.16	50.89	44.98	13	5.98
	FT Single (10 ep)	ℓ_2, StAdv	80.89	45.45	54.5	6.09	41.98	49.98	41.05	37	5.87	2.78
1	FT Single (25 ep)	ℓ_2, StAdv	81.21	44.17	54.6	5.56	40.95	49.38	39.76	36.32	5.36	6.92
	FT Single + ALR (10 ep)	ℓ_2, StAdv	87.24	62.22	61.5	21.4	**70.87**	61.86	55.04	54	21.14	4.24
	FT Single + ALR (25 ep)	ℓ_2, StAdv	87.54	61.21	60.38	20.81	69.49	60.8	54.22	52.97	20.48	8.77
	FT Single + ALR feature ($\lambda = 2$, 10 ep)	ℓ_2, StAdv	81.79	56.98	60.28	20.59	63.64	58.63	51.65	50.37	20.21	3.52
	FT Single + ALR feature ($\lambda = 5$, 10 ep)	ℓ_2, StAdv	81.26	60.43	57.61	**28.17**	67.95	59.02	51.99	53.54	**27.24**	3.53
	FT Croce + ALR (10 ep)	ℓ_2, StAdv	86.03	59.18	**65.14**	15.36	63.31	**62.16**	**55.83**	50.75	15.29	3.47
	FT Croce + ALR (25 ep)	ℓ_2, StAdv	**88.5**	**64.88**	58.98	23.9	70.79	61.93	55.03	**54.64**	23.33	7.96
	FT Croce + ALR feature ($\lambda = 2$, 10 ep)	ℓ_2, StAdv	83.19	61.28	59.04	23.98	62.69	60.16	53.25	51.75	23.2	2.97
	FT Croce + ALR feature ($\lambda = 5$, 10 ep)	ℓ_2, StAdv	83.51	61.69	61.76	23.33	62.48	61.73	55.25	52.31	22.77	3.13
	AVG	ℓ_2, StAdv, ℓ_∞	85.98	67.60	45.81	42.39	62.43	51.93	34.05	54.56	33.39	33.12
	MAX	ℓ_2, StAdv, ℓ_∞	84.54	54.87	52.33	38.23	55.90	48.48	35.25	50.33	34.08	79.04
	Random	ℓ_2, StAdv, ℓ_∞	39.52	67.46	47.35	42.12	63.61	52.31	35.46	55.13	34.79	10.92
	FT MAX (10 ep)	ℓ_2, StAdv, ℓ_∞	83.16	65.63	56.68	36.9	65.69	53.07	35.18	56.23	34.83	5.62
	FT MAX (25 ep)	ℓ_2, StAdv, ℓ_∞	83.99	65.69	58.16	37.21	65.52	53.69	35.76	56.65	35.31	12.88
	FT Croce (10 ep)	ℓ_2, StAdv, ℓ_∞	85.05	67.3	48.07	33.38	62.52	49.58	28.96	52.82	28.63	2.27
	FT Croce (25 ep)	ℓ_2, StAdv, ℓ_∞	86.14	67.3	52.47	35.86	63.43	51.88	32.54	54.77	32.08	5.01
	FT Single (10 ep)	ℓ_2, StAdv, ℓ_∞	87.99	**70.53**	11.17	41.63	63.46	41.11	7.95	46.7	7.74	1.57
2	FT Single (25 ep)	ℓ_2, StAdv, ℓ_∞	88.67	70.23	8.79	43.4	63.03	40.81	6.19	46.36	6.05	3.91
	FT Single + ALR (10 ep)	ℓ_2, StAdv, ℓ_∞	**88.74**	69.15	47.33	42.08	68.62	52.85	36.66	56.8	36.62	2.26
	FT Single + ALR (25 ep)	ℓ_2, StAdv, ℓ_∞	88.14	68.26	49.1	41.48	66.73	52.95	**37.55**	56.39	37.5	5.4
	FT Single + ALR feature ($\lambda = 2$, 10 ep)	ℓ_2, StAdv, ℓ_∞	85.69	67.62	29.42	43.68	68.75	46.91	24.44	52.37	24.38	2.16
	FT Single + ALR feature ($\lambda = 5$, 10 ep)	ℓ_2, StAdv, ℓ_∞	84.03	67.64	42.03	**44.36**	71.36	51.34	32.54	56.35	32.48	2.29
	FT Croce + ALR (10 ep)	ℓ_2, StAdv, ℓ_∞	86.57	67.99	**61.55**	36.59	72.16	**55.38**	35.68	59.57	35.52	2.87
	FT Croce + ALR (25 ep)	ℓ_2, StAdv, ℓ_∞	86.96	68.91	57.21	39.65	**72.22**	55.26	37.25	**59.5**	37.14	6.87
	FT Croce + ALR feature ($\lambda = 2$, 10 ep)	ℓ_2, StAdv, ℓ_∞	83.13	66.91	56.76	38.66	68.57	54.11	35.95	57.73	35.76	2.82
	FT Croce + ALR feature ($\lambda = 5$, 10 ep)	ℓ_2, StAdv, ℓ_∞	84.25	68.14	57.7	39.8	70.29	55.21	37.4	58.98	**37.21**	2.79
	AVG	ℓ_2, StAdv, ℓ_∞, Recolor	87.77	68.55	39.55	**41.97**	67.93	54.5	30.39	54.5	30.39	50.54
	MAX	ℓ_2, StAdv, ℓ_∞, Recolor	84.3	57.62	52.3	41.69	65.1	54.18	**37.44**	54.18	**37.44**	55.54
	Random	ℓ_2, StAdv, ℓ_∞, Recolor	86.32	65.87	47.82	35.04	68.35	54.27	30.76	54.27	30.76	12.41
	FT MAX (10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	83.64	66.21	57.53	37.77	69.32	57.71	36.02	57.71	36.02	8.45
	FT MAX (25 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	83.9	65.72	57.84	38.37	68.84	57.69	36.87	57.69	36.87	21.44
	FT Croce (10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	86.64	**68.76**	44.81	36.02	68.05	54.41	29.44	54.41	29.44	2.34
	FT Croce (25 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	87.11	67.89	49.57	35.58	67.05	55.02	31.21	55.02	31.21	5.9
	FT Single (10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	90.41	66.47	3.93	29.6	69.03	42.26	2.49	42.26	2.49	3.11
3	FT Single (25 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	**90.89**	65.14	3.02	30.32	68.54	41.75	1.92	41.75	1.92	7.41
	FT Single + ALR (10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	90.45	61.58	25.77	27.43	69.26	46.01	19.2	46.01	19.2	4.24
	FT Single + ALR (25 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	90.4	57.07	24.91	22.91	67.39	43.07	17.21	43.07	17.21	9.79
	FT Single + ALR feature ($\lambda = 2$, 10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	90.15	57.89	8.75	22.86	72.27	40.44	6.61	40.44	6.61	3.94
	FT Single + ALR feature ($\lambda = 5$, 10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	88.44	66.03	18.88	34.17	69.35	47.11	16.1	47.11	16.1	3.76
	FT Croce + ALR (10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	87.62	68.14	58.5	36.39	72.35	58.85	34.92	58.85	34.92	3.35
	FT Croce + ALR (25 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	87.05	68.05	**59.26**	38.38	**73.42**	**59.78**	36.83	**59.78**	36.83	7.78
	FT Croce + ALR feature ($\lambda = 2$, 10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	84.78	67.67	53.13	40.25	69.99	57.76	36.3	57.76	36.3	3.04
	FT Croce + ALR feature ($\lambda = 5$, 10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	83.94	67.28	59.21	39.38	71.67	59.38	37.15	59.38	37.15	2.91

Table A.2: **Continual Robust Training on CIFAR-10 (Sequence of 4 Attacks Starting with ℓ_2).** The learner initially has knowledge of ℓ_2 attacks and over time, we are sequentially introduced to StAdv, ℓ_∞, and ReColor attacks. We report clean accuracy, accuracy on different attack types, and average and union accuracies. The threat models column represents the set of attacks known to the defender and accuracies on known attacks are highlighted with in green cells. "FT" procedures are fine-tuning approaches starting from adversarially trained to ℓ_2 model (AT) and then sequentially fine-tuning with new attacks for 25 epochs. AVG, MAX, and Random strategies train models from scratch with all attacks for 100 epochs. The "Avg (known)" and "Union (known)" columns represent average and union accuracies on attacks known to the defender while "Avg (all)" and "Union (all)" columns represent average and union accuracies on all four attacks. Additionally, we report training times for the procedure (non-cumulative) in the "Time" column. Best performance out of both training from scratch and fine-tuning are bolded, while best performance when only comparing fine-tuning approaches is underlined.

Time Step	Procedure	Threat Models	Clean	ℓ_∞	StAdv	Recolor	ℓ_2	Avg (known)	Union (known)	Avg (all)	Union (all)	Time
0	AT	ℓ_∞	**85.93**	51.44	14.87	62.48	**59.48**	51.44	51.44	47.07	11.9	7.52
	AT + ALR	ℓ_∞	83.18	**51.49**	**34.78**	**58.15**	58.15	**51.49**	**51.49**	**53.27**	**29.87**	11.12
1	AVG	ℓ_∞, StAdv	**86.44**	30.05	54.4	46.71	52.1	42.23	28.03	45.81	26.75	23.68
	MAX	ℓ_∞, StAdv	82.62	44.96	53.68	64.24	**60.85**	49.32	40.8	55.93	39.81	23.68
	Random	ℓ_∞, StAdv	83.15	40.86	58.37	60.53	58.17	49.61	38.95	54.48	37.64	11.70
	FT MAX (10 ep)	ℓ_∞, StAdv	81.63	44.13	57.38	66.66	60.27	50.75	41.57	57.11	40.96	3.99
	FT MAX (25 ep)	ℓ_∞, StAdv	81.99	44.32	57.8	66.25	60.29	51.06	41.98	57.16	41.25	9.93
	FT Croce (10 ep)	ℓ_∞, StAdv	82.66	44.75	54.2	65.99	60.27	49.48	39.69	56.3	39.01	2.31
	FT Croce (25 ep)	ℓ_∞, StAdv	83.55	45.12	53.25	66.44	60.65	49.19	39.43	56.36	38.74	5.44
	FT Single (10 ep)	ℓ_∞, StAdv	80.39	31.14	55.88	59.13	51.58	43.51	29.01	49.43	28.67	2.77
	FT Single (25 ep)	ℓ_∞, StAdv	79.85	31.34	54.86	58.69	51.43	43.1	29.01	49.08	28.66	6.6
	FT Single + ALR (10 ep)	ℓ_∞, StAdv	82.77	35.67	57.92	68.38	54.91	46.8	33.69	54.22	33.65	3.51
	FT Single + ALR (25 ep)	ℓ_∞, StAdv	81.81	35.4	59.47	68.63	54.34	47.44	33.72	54.46	33.66	8.73
	FT Croce + ALR (10 ep)	ℓ_∞, StAdv	82.94	**46.39**	**64.13**	**73.58**	59.41	**55.26**	**44.47**	**60.88**	**44.03**	2.99
	FT Croce + ALR (25 ep)	ℓ_∞, StAdv	82.3	45.89	63.76	72.8	59.56	54.82	44	60.5	43.54	7.5
2	AVG	ℓ_∞, StAdv, Recolor	88.67	39.46	47.1	66.87	57.16	51.14	32.61	52.65	32.55	39.72
	MAX	ℓ_∞, StAdv, Recolor	83.42	44.54	53.06	67.56	60.71	55.05	40.23	56.47	40.17	47.21
	Random	ℓ_∞, StAdv, Recolor	83.23	35.01	54.7	68.68	**62.92**	52.8	32.83	55.33	32.83	13.81
	FT MAX (10 ep)	ℓ_∞, StAdv, Recolor	81.97	44.1	57.36	68.68	60.37	56.71	41.21	57.63	41.2	6.72
	FT MAX (25 ep)	ℓ_∞, StAdv, Recolor	82.24	44.36	58.52	68.87	60.23	57.25	**41.73**	57.99	**41.67**	16.69
	FT Croce (10 ep)	ℓ_∞, StAdv, Recolor	84.98	43.32	52.45	69.46	61.04	55.08	37.05	56.57	37.01	2.53
	FT Croce (25 ep)	ℓ_∞, StAdv, Recolor	84.89	44.66	51.6	68.86	61.59	55.04	38.02	56.68	37.96	6.3
	FT Single (10 ep)	ℓ_∞, StAdv, Recolor	90.55	25.27	12.77	**74.01**	48.99	37.35	10.85	40.26	10.85	4.35
	FT Single (25 ep)	ℓ_∞, StAdv, Recolor	**90.24**	33.94	13.43	73.23	53.51	40.2	10.67	43.53	10.64	7.83
	FT Single + ALR (10 ep)	ℓ_∞, StAdv, Recolor	88.38	38.62	24.87	72.69	56.66	45.39	19.2	48.21	19.19	3.41
	FT Single + ALR (25 ep)	ℓ_∞, StAdv, Recolor	89.38	33.64	20.91	73.52	53.36	42.69	17.39	45.36	17.38	9.87
	FT Croce + ALR (10 ep)	ℓ_∞, StAdv, Recolor	84.3	44.39	58.86	71.67	60.42	58.31	40.82	58.84	40.69	3.52
	FT Croce + ALR (25 ep)	ℓ_∞, StAdv, Recolor	84.69	**44.96**	**59.53**	73.54	61.73	**59.34**	41.39	**59.94**	41.22	8.21
3	AVG	ℓ_∞, StAdv, Recolor, ℓ_2	87.77	41.97	39.55	67.93	68.55	54.5	30.39	54.5	30.39	50.54
	MAX	ℓ_∞, StAdv, Recolor, ℓ_2	84.3	41.69	52.3	65.1	57.62	54.18	37.44	54.18	37.44	55.54
	Random	ℓ_∞, StAdv, Recolor, ℓ_2	86.32	35.04	47.82	68.35	65.87	54.27	30.76	54.27	30.76	12.41
	FT MAX (10 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	82.27	44.21	58.13	69.08	60.7	58.03	**41.48**	58.03	**41.48**	7.9
	FT MAX (25 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	82.6	43.84	57.75	68.84	60.23	57.66	41.19	57.66	41.19	19.74
	FT Croce (10 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	85.11	44.71	50.32	68.39	63.29	56.68	37.23	56.68	37.23	2.37
	FT Croce (25 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	85.33	43.8	50.28	68.77	63.17	56.51	36.77	56.51	36.77	5.95
	FT Single (10 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	88.49	**44.93**	18.06	65.96	67.56	49.13	15.78	49.13	15.78	1.63
	FT Single (25 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	**89.3**	42.72	11.85	60.27	**69.12**	45.99	10.71	45.99	10.71	4.07
	FT Single + ALR (10 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	88.14	41.52	26.06	61.97	68.77	49.58	24.19	49.58	24.19	2.52
	FT Single + ALR (25 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	87.8	40.78	28.34	59.47	68.32	49.23	25.92	49.23	25.92	5.89
	FT Croce + ALR (10 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	84.56	42.19	55.55	69.95	60.69	57.1	38.24	57.1	38.24	3.4
	FT Croce + ALR (25 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	84.1	43.32	**58.2**	**72.09**	61.96	**58.89**	39.97	**58.89**	39.97	8.28

Table A.3: **Continual Robust Training on CIFAR-10 (Sequence of 4 Attacks Starting with ℓ_∞).** The learner initially has knowledge of ℓ_∞ attacks and over time, we are sequentially introduced to StAdv, ReColor, and ℓ_2 attacks. We report clean accuracy, accuracy on different attack types, and average and union accuracies. The threat models column represents the set of attacks known to the defender and accuracies on known attacks are highlighted with in green cells. "FT" procedures are fine-tuning approaches starting from adversarially trained to ℓ_∞ model (AT) and then sequentially fine-tuning with new attacks for 25 epochs. AVG, MAX, and Random strategies train models from scratch with all attacks for 100 epochs. The "Avg (known)" and "Union (known)" columns represent average and union accuracies on attacks known to the defender while "Avg (all)" and "Union (all)" columns represent average and union accuracies on all four attacks. Additionally, we report training times for the procedure (non-cumulative) in the "Time" column. Best performance out of both training from scratch and fine-tuning are bolded, while best performance when only comparing fine-tuning approaches is underlined.

Time Step	Procedure	Threat Models	Clean	ℓ_2	StAdv	ℓ_∞	Recolor	Avg (known)	Union (known)	Avg (all)	Union (all)	Time (hrs)
0	AT	ℓ_2	**90.22**	83.95	10.65	7.67	49.22	83.95	83.95	37.87	3.16	1.71
	AT + ALR	ℓ_2	89.76	**84.41**	**28.23**	**25.22**	**54.70**	**84.41**	**84.41**	**48.14**	**18.01**	2.15
1	AVG	ℓ_2, StAdv	84.56	77.68	74.32	7.57	31.33	76	73.68	47.73	7.44	3.58
	MAX	ℓ_2, StAdv	85.22	76.87	**77.63**	4.94	27.61	**77.25**	**75.57**	46.76	4.76	3.52
	Random	ℓ_2, StAdv	85.71	77.55	74.32	5.78	29.61	75.94	73.55	46.82	5.53	2.58
	FT MAX (10 ep)	ℓ_2, StAdv	83.92	77.5	69.02	10.78	35.77	73.26	68.89	48.27	10.45	0.61
	FT MAX (25 ep)	ℓ_2, StAdv	84.56	77.73	69.35	9.76	36.15	73.54	69.1	48.25	9.43	1.44
	FT Croce (10 ep)	ℓ_2, StAdv	85.07	78.62	67.52	10.57	38.34	73.07	67.31	48.76	10.29	0.4
	FT Croce (25 ep)	ℓ_2, StAdv	**86.37**	**79.67**	69.32	9.81	38.27	74.5	69.17	49.27	9.63	0.98
	FT Single (10 ep)	ℓ_2, StAdv	84.08	77.86	68.31	10.83	36.97	73.08	68.13	48.49	10.45	0.51
	FT Single (25 ep)	ℓ_2, StAdv	85.63	78.39	72.31	7.57	35.31	75.35	72.08	48.39	7.36	1.15
	FT Single + ALR (10 ep)	ℓ_2, StAdv	83.8	77.94	71.62	**20.71**	43.13	74.78	71.34	**53.35**	**20.13**	0.58
	FT Single + ALR (25 ep)	ℓ_2, StAdv	83.9	77.78	71.97	17.35	38.39	74.88	71.59	51.38	16.76	1.44
	FT Croce + ALR (10 ep)	ℓ_2, StAdv	85.04	79.54	69.99	18.68	42.93	74.76	69.89	52.78	18.09	0.51
	FT Croce + ALR (25 ep)	ℓ_2, StAdv	85.07	79.39	68	19.57	**43.67**	73.69	67.97	52.66	19.16	1.24
2	AVG	ℓ_2, StAdv, ℓ_∞	**86.62**	**84.92**	68.89	50.57	66.98	**68.13**	49.17	**67.84**	47.82	10.51
	MAX	ℓ_2, StAdv, ℓ_∞	80.36	78.09	68.38	**52.61**	67.29	66.36	**51.77**	66.59	**50.37**	11.96
	Random	ℓ_2, StAdv, ℓ_∞	84.92	83.06	68.76	49.50	66.11	67.11	48.15	66.86	46.60	4.29
	FT MAX (10 ep)	ℓ_2, StAdv, ℓ_∞	81.76	76.69	71.03	28.31	54.32	58.68	28.31	57.59	27.69	0.67
	FT MAX (25 ep)	ℓ_2, StAdv, ℓ_∞	82.04	77.86	69.02	42.83	66.9	63.24	42.37	64.15	41.86	1.71
	FT Croce (10 ep)	ℓ_2, StAdv, ℓ_∞	83.59	78.8	69.53	34.17	61.5	60.83	34.06	61	33.61	0.3
	FT Croce (25 ep)	ℓ_2, StAdv, ℓ_∞	85.22	81.02	69.58	39.92	64.79	63.51	39.59	63.83	39.03	0.73
	FT Single (10 ep)	ℓ_2, StAdv, ℓ_∞	82.06	77.25	73.1	27.21	57.4	59.18	27.21	58.74	26.9	0.22
	FT Single (25 ep)	ℓ_2, StAdv, ℓ_∞	82.04	77.96	70.42	41.15	66.09	63.18	40.92	63.9	40.46	0.54
	FT Single + ALR (10 ep)	ℓ_2, StAdv, ℓ_∞	81.38	77.89	71.8	46.68	**72.13**	65.45	46.5	67.12	46.14	0.31
	FT Single + ALR (25 ep)	ℓ_2, StAdv, ℓ_∞	80.92	77.43	70.78	47.16	70.6	65.12	46.96	66.49	46.62	0.79
	FT Croce + ALR (10 ep)	ℓ_2, StAdv, ℓ_∞	83.95	79.57	69.22	37.96	59.77	62.25	37.86	61.63	36.99	0.40
	FT Croce + ALR (25 ep)	ℓ_2, StAdv, ℓ_∞	83.11	79.24	**72.38**	36.61	60.18	62.74	36.59	62.1	36.15	1.01
3	AVG	ℓ_2, StAdv, ℓ_∞, Recolor	**87.67**	**85.66**	66.06	50.42	75.90	69.51	47.90	69.51	47.90	13.79
	MAX	ℓ_2, StAdv, ℓ_∞, Recolor	83.26	81.22	70.70	**56.94**	74.80	**70.92**	**55.31**	**70.92**	**55.31**	14.60
	Random	ℓ_2, StAdv, ℓ_∞, Recolor	86.55	84.64	66.52	47.29	74.93	68.34	45.71	68.34	45.71	9.61
	FT MAX (10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	81.99	77.78	68.28	41.83	69.91	64.45	41.4	64.45	41.4	1.31
	FT MAX (25 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	82.78	79.21	**70.83**	45.15	71.39	66.64	44.76	66.64	44.76	3.6
	FT Croce (10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	84.87	80.38	66.68	36.82	68.61	63.12	36.31	63.12	36.31	0.45
	FT Croce (25 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	86.32	82.11	68.79	41.27	72.41	66.15	40.69	66.15	40.69	1.2
	FT Single (10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	86.27	81.35	54.73	23.59	70.17	57.46	22.55	57.46	22.55	0.71
	FT Single (25 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	85.1	80.48	58.17	36.38	70.62	61.41	34.45	61.41	34.45	2.03
	FT Single + ALR (10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	85.3	81.04	49.35	40.48	74.8	61.42	35.62	61.42	35.62	0.85
	FT Single + ALR (25 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	86.78	82.8	47.82	33.12	**77.58**	60.33	29.17	60.33	29.17	2.38
	FT Croce + ALR (10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	85.3	81.3	69.35	43.13	70.85	66.16	42.62	66.16	42.62	0.53
	FT Croce + ALR (25 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	85.81	81.76	67.13	45.38	73.02	66.82	44.56	66.82	44.56	1.36

Table A.4: **Continual Robust Training on ImageNette (Sequence of 4 Attacks Starting with ℓ_2).**

Time Step	Procedure	Threat Models	Clean	ℓ_∞	StAdv	Recolor	ℓ_2	Avg (known)	Union (known)	Avg (all)	Union (all)	Time (hrs)
0	AT	ℓ_∞	**82.52**	56.94	**61.32**	71.62	**78.39**	56.94	56.94	**67.07**	50.32	1.70
	AT + ALR	ℓ_∞	81.52	**59.62**	60.51	**73.50**	72.69	**59.62**	**59.62**	66.58	**52.92**	2.67
1	AVG	ℓ_∞, StAdv	**85.78**	53.30	**75.69**	67.69	**81.96**	64.5	53.02	69.66	51.11	5.87
	MAX	ℓ_∞, StAdv	83.77	**58.04**	70.04	**72.76**	80.38	64.04	**56.54**	**70.31**	55.26	6.11
	Random	ℓ_∞, StAdv	83.34	52.23	73.76	67.85	79.87	62.99	51.77	68.43	50.39	2.44
	FT MAX (10 ep)	ℓ_∞, StAdv	82.27	55.03	70.52	69.78	78.27	62.78	54.17	68.4	52.66	0.62
	FT MAX (25 ep)	ℓ_∞, StAdv	82.57	55.46	71.75	69.94	78.73	63.61	54.85	68.97	53.22	1.48
	FT Croce (10 ep)	ℓ_∞, StAdv	82.29	54.62	69.2	68.87	78.32	61.91	53.35	67.75	51.9	0.37
	FT Croce (25 ep)	ℓ_∞, StAdv	83.67	54.27	71.57	69.07	79.54	62.92	53.58	68.61	52.2	0.86
	FT Single (10 ep)	ℓ_∞, StAdv	83.06	50.52	71.52	65.43	78.78	61.02	49.96	66.56	48.18	0.51
	FT Single (25 ep)	ℓ_∞, StAdv	84.00	43.59	73.68	58.14	78.85	58.64	43.46	63.57	41.27	1.16
	FT Single + ALR (10 ep)	ℓ_∞, StAdv	82.19	42.7	73.17	60.97	77.55	57.94	42.6	63.6	41.27	0.58
	FT Single + ALR (25 ep)	ℓ_∞, StAdv	81.4	51.8	69.35	66.32	76.54	60.57	51.08	66	50.04	1.47
	FT Croce + ALR (10 ep)	ℓ_∞, StAdv	82.62	57.71	70.11	72.41	77.89	63.91	**56.54**	69.53	55.62	0.49
	FT Croce + ALR (25 ep)	ℓ_∞, StAdv	82.37	57.55	71.64	70.93	78.14	**64.6**	**56.54**	69.57	**55.64**	1.11
2	AVG	ℓ_∞, StAdv, Recolor	**86.39**	51.80	**73.81**	**77.99**	**83.13**	**67.86**	51.31	**71.68**	51.31	11.57
	MAX	ℓ_∞, StAdv, Recolor	81.20	54.55	68.64	72.82	78.01	65.33	53.12	68.5	53.12	13.55
	Random	ℓ_∞, StAdv, Recolor	86.29	50.96	72.28	76.59	82.96	66.61	50.27	70.69	50.27	4.90
	FT MAX (10 ep)	ℓ_∞, StAdv, Recolor	82.34	55.34	71.34	72.87	78.22	66.51	54.04	69.44	54.04	1.21
	FT MAX (25 ep)	ℓ_∞, StAdv, Recolor	83.75	55.29	72.56	74.83	79.97	67.56	54.27	70.66	54.27	3.03
	FT Croce (10 ep)	ℓ_∞, StAdv, Recolor	84.28	54.01	69.96	72.56	79.72	65.51	52.13	69.06	52.13	0.5
	FT Croce (25 ep)	ℓ_∞, StAdv, Recolor	84.05	52.99	70.47	73.07	80.33	65.51	51.92	69.22	51.92	1.23
	FT Single (10 ep)	ℓ_∞, StAdv, Recolor	83.77	53.61	65.38	73.35	79.36	64.11	50.7	67.92	50.7	0.75
	FT Single (25 ep)	ℓ_∞, StAdv, Recolor	85.07	48.2	65.86	75.41	80.41	63.16	46.34	67.47	46.34	1.88
	FT Single + ALR (10 ep)	ℓ_∞, StAdv, Recolor	84.94	50.8	65.71	76.33	80.15	64.28	48.31	68.25	48.31	0.88
	FT Single + ALR (25 ep)	ℓ_∞, StAdv, Recolor	82.09	55.46	66.27	73.63	77.76	65.12	52.89	68.28	52.89	2.14
	FT Croce + ALR (10 ep)	ℓ_∞, StAdv, Recolor	82.42	**55.75**	65.83	73.3	78.06	64.96	52.94	68.24	52.94	0.61
	FT Croce + ALR (25 ep)	ℓ_∞, StAdv, Recolor	83.9	55.52	71.21	75.29	79.82	67.34	**54.32**	70.46	**54.32**	1.49
3	AVG	ℓ_∞, StAdv, Recolor, ℓ_2	**87.67**	50.42	66.06	**75.90**	**85.66**	69.51	47.90	69.51	47.90	13.79
	MAX	ℓ_∞, StAdv, Recolor, ℓ_2	83.26	56.94	**70.70**	74.80	81.22	**70.92**	**55.31**	**70.92**	**55.31**	14.60
	Random	ℓ_∞, StAdv, Recolor, ℓ_2	86.55	47.29	66.52	74.93	84.64	68.34	45.71	68.34	45.71	4.58
	FT MAX (10 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	82.73	**55.08**	71.36	73.5	78.98	69.73	54.19	69.73	54.19	1.3
	FT MAX (25 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	83.72	54.93	72.18	74.42	79.9	70.36	53.94	70.36	53.94	3.26
	FT Croce (10 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	84.33	52.18	69.5	72.74	79.97	68.6	50.55	68.6	50.55	0.44
	FT Croce (25 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	84.84	52.59	68.74	73.27	81.45	69.01	50.96	69.01	50.96	1.1
	FT Single (10 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	84.79	53.02	65.43	72.82	80.08	67.83	50.29	67.83	50.29	0.26
	FT Single (25 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	85.5	49.12	64.79	74.27	80.76	67.24	47.21	67.24	47.21	0.63
	FT Single + ALR (10 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	85.1	45.4	63.44	67.06	80.59	64.12	42.96	64.12	42.96	0.35
	FT Single + ALR (25 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	83.64	54.37	64.48	72.41	79.69	67.74	50.96	67.74	50.96	0.92
	FT Croce + ALR (10 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	83.03	53.96	67.9	72.38	79.08	68.33	51.95	68.33	51.95	0.55
	FT Croce + ALR (25 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	84.84	53.94	68.51	75.11	81.32	69.72	52.23	69.72	52.23	1.36

Table A.5: **Continual Robust Training on ImageNette (Sequence of 4 Attacks Starting with ℓ_∞).**

Time Step	Procedure	Threat Models	Clean	ℓ_2	StAdv	ℓ_∞	Recolor	Avg (known)	Union (known)	Avg (all)	Union (all)	Time (hrs)
0	AT	ℓ_2	**67.75**	41.65	4.21	14.28	22.46	41.65	41.65	20.65	2.34	14.85
	Ours	ℓ_2	63.53	**42.88**	**6.16**	**19.8**	**23.36**	**42.88**	**42.88**	**23.05**	**4.37**	21.98
1	AVG	ℓ_2, StAdv	**64.48**	**35.72**	28.57	9.18	19.51	32.15	25.25	**23.24**	7.53	47.24
	MAX	ℓ_2, StAdv	61.80	33.82	31.77	7.58	17.76	**32.79**	**27.68**	22.73	6.51	47.24
	Random	ℓ_2, StAdv	62.7	31.75	**32.25**	6.50	17.62	32.00	26.28	22.03	5.79	23.56
	FT MAX (10 ep)	ℓ_2, StAdv	60.3	30.9	29.32	5.96	17.04	30.11	23.97	20.8	5.15	4
	FT MAX (25 ep)	ℓ_2, StAdv	61.14	31.69	29.93	6.21	17.84	30.81	24.84	21.42	5.3	10.71
	FT Croce (10 ep)	ℓ_2, StAdv	62.68	35.2	23.7	8.9	**19.93**	29.45	21.02	21.93	6.77	2.6
	FT Croce (25 ep)	ℓ_2, StAdv	63.39	31.38	28.95	5.7	17.98	30.16	23.87	21	5.04	5.96
	FT Single (10 ep)	ℓ_2, StAdv	60.48	18.29	30.83	2.24	13.28	24.56	16.19	16.16	1.84	2.77
	FT Single (25 ep)	ℓ_2, StAdv	60.78	18.22	31.79	1.99	13.32	25	16.34	16.33	1.6	6.91
	FT Single + ALR (10 ep)	ℓ_2, StAdv	53	29.23	24	8.45	17.18	26.61	20.12	19.71	6.71	2.77
	FT Single + ALR (25 ep)	ℓ_2, StAdv	54.44	22.03	29.1	4.46	13.6	25.56	19.21	17.3	3.78	8.73
	FT Croce + ALR (10 ep)	ℓ_2, StAdv	59.09	34.09	26.59	9.84	19.2	30.34	23.62	22.43	8.28	3.11
	FT Croce + ALR (25 ep)	ℓ_2, StAdv	58.59	34.65	26.45	**10.55**	19.65	30.55	23.67	22.82	**8.64**	7.67
2	AVG	ℓ_2, StAdv, ℓ_∞	62.09	40.34	25.13	20.88	28.84	**28.78**	16.18	28.80	14.71	69.49
	MAX	ℓ_2, StAdv, ℓ_∞	56.94	34.74	**28.46**	22.72	**30.35**	28.64	**19.66**	**29.04**	**17.55**	69.36
	Random	ℓ_2, StAdv, ℓ_∞	60.98	38.01	25.21	17.25	25.76	26.82	14.26	26.56	12.97	19.58
	FT MAX (10 ep)	ℓ_2, StAdv, ℓ_∞	59.9	38.94	26.55	16.95	25.72	27.48	14.78	27.04	13.07	5.2
	FT MAX (25 ep)	ℓ_2, StAdv, ℓ_∞	61.01	38.56	27.54	17.05	25.91	27.72	15.34	27.27	13.66	12.25
	FT Croce (10 ep)	ℓ_2, StAdv, ℓ_∞	62.78	39.87	20.17	14.59	23.7	24.88	10.46	24.58	9.2	2.29
	FT Croce (25 ep)	ℓ_2, StAdv, ℓ_∞	65.85	42.51	11.05	19.67	26.55	24.41	8.06	24.95	7.3	5.06
	FT Single (10 ep)	ℓ_2, StAdv, ℓ_∞	65.62	42.95	7.55	22.03	28.06	24.18	5.57	25.15	4.98	1.49
	FT Single (25 ep)	ℓ_2, StAdv, ℓ_∞	**65.93**	42.82	7.62	21.83	28.2	24.09	5.65	25.12	5.12	3.71
	FT Single + ALR (10 ep)	ℓ_2, StAdv, ℓ_∞	62.35	**43.56**	8.76	23.72	27.77	25.35	6.98	25.95	6.29	2.16
	FT Single + ALR (25 ep)	ℓ_2, StAdv, ℓ_∞	62.56	42.33	7.7	**25.06**	26.57	25.03	6.67	25.41	6.02	5.39
	FT Croce + ALR (10 ep)	ℓ_2, StAdv, ℓ_∞	60.67	42.06	16.66	21.18	25.89	26.63	12.59	26.45	11.21	3.05
	FT Croce + ALR (25 ep)	ℓ_2, StAdv, ℓ_∞	63.43	42.92	10.14	23.16	26.37	25.41	8.29	25.65	7.64	6.7
3	AVG	ℓ_2, StAdv, ℓ_∞, Recolor	65.61	40.86	22.4	20.45	37.27	30.25	14.09	30.25	14.09	101.43
	MAX	ℓ_2, StAdv, ℓ_∞, Recolor	59.12	33.89	**28.02**	**22.20**	35.00	29.78	**18.74**	29.78	**18.74**	101.43
	Random	ℓ_2, StAdv, ℓ_∞, Recolor	63.1	39.47	24.79	19.04	38.15	**30.36**	14.57	**30.36**	14.57	22.87
	FT MAX (10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	61.5	39.34	26.97	17.25	33.56	29.28	14.55	29.28	14.55	8.61
	FT MAX (25 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	62.14	38.68	27.51	17.13	33.06	29.09	14.84	29.09	14.84	19.67
	FT Croce (10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	64.82	41.09	19.78	16.55	32.26	27.42	10.57	27.42	10.57	2.42
	FT Croce (25 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	66.31	41.02	13.42	17.34	31.02	25.7	8.4	25.7	8.4	6.03
	FT Single (10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	**69.82**	32.63	4.07	9.38	40.07	21.54	1.42	21.54	1.42	3.06
	FT Single (25 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	68.63	37.06	5.57	13.28	37.66	23.39	2.76	23.39	2.76	7.78
	FT Single + ALR (10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	66.58	37.98	6.65	16.37	39.23	25.06	3.83	25.06	3.83	3.91
	FT Single + ALR (25 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	68.15	32.05	5.08	11.82	**41.5**	22.61	2.46	22.61	2.46	9.72
	FT Croce + ALR (10 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	64.11	**42.52**	10.89	21.36	34.05	27.21	8.11	27.21	8.11	3.42
	FT Croce + ALR (25 ep)	ℓ_2, StAdv, ℓ_∞, Recolor	65.33	39.4	11.41	16.84	34.15	25.45	7.35	25.45	7.35	7.41

Table A.6: **Continual Robust Training on CIFAR-100 (Sequence of 4 Attacks Starting with ℓ_2).**

Time Step	Procedure	Threat Models	Clean	ℓ_∞	StAdv	Recolor	ℓ_2	Avg (known)	Union (known)	Avg (all)	Union (all)	Time (hrs)
0	AT	ℓ_∞	**60.95**	27.61	9.92	**33.2**	**35.6**	27.61	27.61	**26.58**	7.45	16.33
	AT + ALR	ℓ_∞	55.36	**28.01**	**11.25**	31.01	33.95	**28.01**	**28.01**	26.05	8.62	23.75
1	AVG	ℓ_∞, StAdv	**66.09**	8.65	33.19	18.42	21.58	20.92	8.18	20.46	7.47	47.59
	MAX	ℓ_∞, StAdv	57.01	22.8	29.23	30.54	**33.58**	**26.02**	**20.28**	**29.04**	**17.96**	46.97
	Random	ℓ_∞, StAdv	47.14	16.62	25.61	25.35	27.25	21.12	14.63	23.71	13.37	23.92
	FT MAX (10 ep)	ℓ_∞, StAdv	58.05	22.48	28.66	30.43	33.15	25.57	18.95	28.68	17.05	4.03
	FT MAX (25 ep)	ℓ_∞, StAdv	58.56	22.36	29.38	**30.97**	33.16	25.87	19.38	28.97	17.43	10.02
	FT Croce (10 ep)	ℓ_∞, StAdv	60.17	22.75	26.81	31.22	33.68	24.78	17.54	28.62	15.95	2.49
	FT Croce (25 ep)	ℓ_∞, StAdv	60.27	22.18	28.17	30.38	33.25	25.18	17.81	28.5	16.21	5.69
	FT Single (10 ep)	ℓ_∞, StAdv	55.87	15.43	24.29	24.34	27.54	19.86	11.9	22.9	10.95	2.77
	FT Single (25 ep)	ℓ_∞, StAdv	56.11	16.09	24.46	24.84	28.65	20.27	12.5	23.51	11.33	6.95
	FT Single + ALR (10 ep)	ℓ_∞, StAdv	56.03	3.81	35.21	18.52	17.31	19.51	3.77	18.71	3.51	75.49
	FT Single + ALR (25 ep)	ℓ_∞, StAdv	59.65	2.6	**37.96**	18.51	13.52	20.28	2.58	18.15	2.41	8.34
	FT Croce + ALR (10 ep)	ℓ_∞, StAdv	54.86	**23.33**	27.19	30.15	31.54	25.26	18.75	28.05	16.86	2.97
	FT Croce + ALR (25 ep)	ℓ_∞, StAdv	54.27	23.27	26.27	29.4	31.79	24.77	18.25	27.68	16.06	7.4
2	AVG	ℓ_∞, StAdv, Recolor	**68.19**	16.88	**29.53**	38.12	30.75	28.18	14.14	28.82	14.11	79.47
	MAX	ℓ_∞, StAdv, Recolor	57.96	22.38	28.92	35.24	33.9	28.85	**19.27**	30.11	**19.17**	79.5
	Random	ℓ_∞, StAdv, Recolor	47.14	16.62	25.61	25.35	27.25	21.12	14.63	23.71	13.37	23.92
	FT MAX (10 ep)	ℓ_∞, StAdv, Recolor	58.96	22.04	29	36.35	34.55	29.13	18.37	30.48	18.31	6.75
	FT MAX (25 ep)	ℓ_∞, StAdv, Recolor	59.41	21.7	29.2	35.57	33.24	28.82	18.21	29.93	18.09	16.75
	FT Croce (10 ep)	ℓ_∞, StAdv, Recolor	62.27	22.42	26.16	37.25	34.5	28.61	16.34	30.08	16.24	2.77
	FT Croce (25 ep)	ℓ_∞, StAdv, Recolor	62.09	23.24	25.6	36.91	35.89	28.58	16.63	30.41	16.52	6.46
	FT Single (10 ep)	ℓ_∞, StAdv, Recolor	64.02	20.72	14.72	41.7	32.89	25.71	9.61	27.51	9.57	3.01
	FT Single (25 ep)	ℓ_∞, StAdv, Recolor	67.39	13.25	6.94	45.1	27.71	21.76	3.59	23.25	3.57	7.86
	FT Single + ALR (10 ep)	ℓ_∞, StAdv, Recolor	64.86	8.76	9.34	46.4	25.62	21.5	3.91	22.53	3.91	3.8
	FT Single + ALR (25 ep)	ℓ_∞, StAdv, Recolor	66.87	3.74	8.44	**50.81**	19.81	21	2.03	20.7	2.03	9.89
	FT Croce + ALR (10 ep)	ℓ_∞, StAdv, Recolor	56.41	24.28	25.62	36.21	34.43	28.7	17.6	30.14	17.41	3.6
	FT Croce + ALR (25 ep)	ℓ_∞, StAdv, Recolor	56.83	22.43	25.9	38.27	33.5	**28.87**	16.98	30.03	16.76	8.28
3	AVG	ℓ_∞, StAdv, Recolor, ℓ_2	64.8	20.9	22.46	37.27	41.05	30.42	14.56	30.42	14.56	101.39
	MAX	ℓ_∞, StAdv, Recolor, ℓ_2	57.9	22.39	28.72	35.96	35.65	**30.68**	**19.15**	**30.68**	**19.15**	101.25
	Random	ℓ_∞, StAdv, Recolor, ℓ_2	63.23	19.52	21.29	**39.71**	39.95	30.12	13.6	30.12	13.6	24.88
	FT MAX (10 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	59.61	22.14	29.13	36.17	34.35	30.45	18.61	30.45	18.61	8.58
	FT MAX (25 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	59.42	22.02	**29.28**	35.96	34.45	30.43	18.64	30.43	18.64	21.36
	FT Croce (10 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	62.44	20.96	26.06	35.91	36.77	29.93	15.83	29.93	15.83	2.38
	FT Croce (25 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	62.17	21.84	26.14	36.92	36.69	30.4	16.08	30.4	16.08	5.81
	FT Single (10 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	63.94	23.86	13.73	37.22	41.47	29.07	9.92	29.07	9.92	1.61
	FT Single (25 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	**66.44**	21.17	7.72	31.83	**42.5**	25.8	5.67	25.8	5.67	4.07
	FT Single + ALR (10 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	60.76	22.36	10.35	31.33	41.91	26.49	7.99	26.49	7.99	2.35
	FT Single + ALR (25 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	62.25	20.56	7.92	30.69	41.42	25.15	6.25	25.15	6.25	6.35
	FT Croce + ALR (10 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	57.56	24.64	22.52	35.77	37.55	30.12	15.96	30.12	15.96	3.36
	FT Croce + ALR (25 ep)	ℓ_∞, StAdv, Recolor, ℓ_2	58.14	**24.85**	18.69	36.75	39.16	29.86	13.87	29.86	13.87	7.64

Table A.7: **Continual Robust Training on CIFAR-100 (Sequence of 4 Attacks Starting with ℓ_∞).**

APPENDIX B

REDISTRICTING AND DIFFERENTIAL PRIVACY

B.1 Additional Results

MCMC Convergence tests To provide evidence of the convergence of our chains, we report the Gelman-Rubin split-$\hat{R}$ metric and the effective sample size (ESS) for each legislative geography we study in Table B.1. Traditionally, an $\hat{R}$ of below 1.1 has been accepted as evidence of convergence, although more recent work recommends that using a threshold of 1.01 for $\hat{R}$ along with a rank-normalized ESS of at least 400 for more robust results. We test for the convergence of our measurement of population balance (i.e. whether our estimate of the mean of the function $f(\mathcal{P}) = \mathbb{1}\left[\mathsf{dev}_{\mathsf{swap}}(\mathcal{P}) > 0.1\right]$ has converged, where $\mathcal{P}$ is a plan sampled from the stationary distribution of our Markov chain) and our measurement of MMD discrepancy (i.e. $f(\mathcal{P}) = \mathsf{MMD}_{\mathsf{demo}}(\mathcal{P}) - \mathsf{MMD}_{\mathsf{swap}}(\mathcal{P})$). For our population balance measurment, we tested convergence using 4 chains of length 1,000,000, subsampling every 10^{th} plan to yield an ensemble of 100,000 plans. For the MMD discrepancy measurement, we tested convergence using the 100 ensembles that we generated for each legislative geography as described in Section 4.6. We say that a chain shows signs of convergence after a certain number of steps when $\hat{R} < 1.01$ and $\texttt{ESS} > 400$.

All geographies except for the Alabama, Connecticut, West Virginia, and Wyoming state lower houses showed signs of convergence after 1,000,000 steps when measuring population balance. Many geographies did not show signs of converging after 100,000 steps under the MMD discrepancy measurement. However, given our method for creating ensembles with lots of MMDs (sampling maximal plans from each of the 100 chains that we ran), the convergence of the individual chains doesn't have a straightforward relationship to that of the final ensemble. Further work is necessary to characterize the distribution of plans that maximize majority-minority districts.

Population Balance and Critical Offsets In Table B.2 we compute no offset discrepancy rates and the critical offsets for state lower and upper house geographies. The critical offsets are computed by repeatedly running chains with a higher offset Δ from the base acceptable τ. For each geography, starting at $\Delta = 0$, we sample an ensemble using a population tolerance threshold in `demo` of $5\% - \Delta$, and check whether any plan exceeds a 5% population tolerance. If this there does not exist such a plan in the ensemble, the critical offset is Δ, otherwise we increase Δ by 0.05% and repeat the procedure. The average and standard deviation are computed over 10 experimental runs. We default to $\tau = 5\%$, and Δ is increased in increments of 0.05%. We repeat this computation ten times for each geography to observe the variation in critical offset between runs. The largest deviation we observed between the smallest and largest critical offset for a geography was 0.55% for the Montana state lower house, with all other geographies having deviations no greater than 0.5%. We also report what are synthetic model would predict for no offset discrepancy rate and critical offset for each geography, as described in Section 4.5.2.

Robustness to Geounit Type All results in this chapter were reported for chains using either precincts or census block groups as the base unit of district composition. In Table B.3 we report results for our critical offset computations using chains that are composed from census blocks, the smallest unit of geography defined in the census hierarchy which is sometimes used as the base unit in redistricting. Due to computational constraints, we only report results for three legislative geographies: RI state lower house, RI state upper house, and CT state lower house. In all three geographies, we found that using blocks instead of block groups led to an increase in both no-offset discrepancy rate and critical offset. See Section 4.8 for additional discussion about the possible effects of the choice of geounit.

Robustness to District Type We provide metrics from chains run on Congressional districting plans for each state that was apportioned more than one seat in the House of

Representatives after the 2010 census. While there is no population deviation that is generally considered de minimis in Congressional redistricting, we adopt the same $\tau = 5\%$ used in our other experiments for the sake of consistency. We find that in every state, the average critical offset over 10 runs, as defined in Section 4.5, is less than 0.2%.

Robustness to Minority Group We compare our results from Section 5 to state lower house chains optimized to increase the number of majority-Hispanic districts in a given plan, rather than majority-Black districts. Each optimized ensemble is constructed by running 100 ReCom chains with short bursts optimization to generate plans with many majority Hispanic districts, and sampling 10 plans from each chain with the highest number of majority Hispanic districts to be included in the ensemble. Each chain uses a population tolerance threshold of $5\% - \Delta$, where Δ is the critical offset for that legislative geography, as computed in Section 4.5. The magnitudes of discrepancies we observed for majority-Hispanic districts are similar in magnitude to those of majority-Black districts. For example, the largest mean discrepancy we observed for majority-Hispanic districts was 1.093 in the New Mexico State Lower House, while the larges we observed for majority-Black districts was 1.431 in the Mississippi State Lower House.

MMD Discrepancies Across Geographies Table B.6 contains the mean MMD discrepancy and non-zero discrepancy rate for optimized ensembles across 52 legislative geographies. Each optimized ensemble is constructed by running 100 ReCom chains with short bursts optimization to generate plans with many majority-Black districts, and sampling 10 plans from each chain with the highest number of majority-Black districts to be included in the ensemble. Each chain uses a population tolerance threshold of $5\% - \Delta$, where Δ is the critical offset for that legislative geography, as computed in Section 4.5. The legislative geographies included are those for which our chains found at least one plan with at least one majority-minority district.

Distribution of BVAP Population in Standard and Optimized Ensembles. Figure B.1 plots histograms of the %BVAP for all plans sampled in the base ensemble and the short bursts ensemble. The short burst optimization technique samples heavily from plans that are just over 50% BVAP.

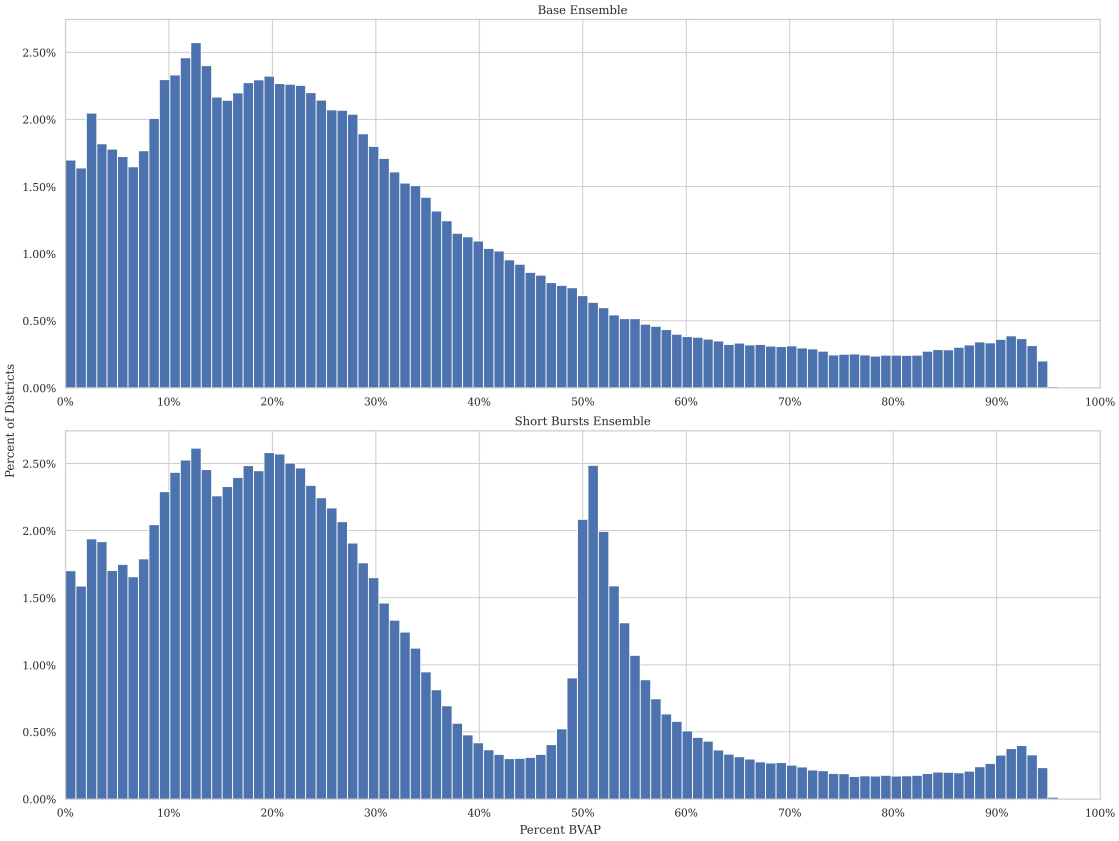


Figure B.1: **Histograms of % BVAP Population for Districts in Two Georgia State House Ensembles.** The data include all distinct districts in our base (top) and short bursts (bottom) ensembles. The short bursts ensemble, which is designed to produce plans with many MMDs, samples districts with small Black majorities much more often than the base ensemble which ignores race.

	State Lower House				State Upper House			
	Population Balance		MMD		Population Balance		MMD	
	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS	$\hat{R}$	ESS
AL	1.41348	16.2	1.03605	1665.2	1.00004	115308	1.01460	4993.0
AK	1.00053	9602.7			1.00003	95286.1		
AZ	1	144229			1.00001	135120		
AR	1.00006	18532.2	1.09077	712.0	1.00005	99684.9	1.07575	798.1
CA	1.00007	48475.2	1.02305	2671.2	1	131413		
CO	1.00005	50360.1			1.00008	118716		
CT	1.04848	63.8	1.77493	149.5	1.00001	73049.9	1.00271	30996.1
DE	1.00028	13237.5	1.01678	5751.8	1.00003	115724	1.02261	2979.9
FL	1.00015	25729	1.03677	1722.2	1.00001	115492	1.08056	765.7
GA	1.00039	11449.4	1.09305	680.1	1.00008	68435.2	1.09921	617.7
HI								
ID	1.00009	58208.3			1.00005	56707.5		
IL	1.00016	32520.9	1.02456	4013.5	1.00004	73011.1	1.00655	16931.1
IN	1.00029	31532.7	1.05866	1061.5	1.00002	79920.8	1.08445	713.4
IA	1.00012	19643.4			1.00002	56643.6		
KS	1.00022	19338	1.23501	310.5	1.00001	82320.3		
KY	1.00021	29848.4	1.02158	1703.8	1.00002	124136.5		
LA	1.00004	25932.2	1.08096	776.0	1.00004	97275.8	1.04140	1498.4
ME	1.00151	3354.3			1.00061	11535.5		
MD	1.00004	40961	1.05367	1167.7	1.00003	73312.8	1.05580	1104.4
MA	1.00076	10113.5	1.04999	1335.7	1.00009	81589.6	1.09748	648.4
MI	1.00008	28161.2	1.00857	13478.6	0.999999	124932	1.02615	2555.4
MN	1.00034	12959.6			1.00011	48035.2		
MS	1.00096	7232.6	1.09317	668.6	1.00008	53867.2	1.04285	1481.9
MO	1.00011	17791.9	1.04485	1430.1	0.999997	135122	1.14201	467.1
MT	1.00070	3922.1			1.00005	29807.9		
NE					1.00014	53859.5	1.31089	249.0
NV	1.00011	64538.9			1	179202		
NH					1.00969	528.9		
NJ	1.00002	113039	1.02834	2232.8	1.00004	113333	1.03377	1875.2
NM	1.00019	25091.2			1.00005	48720.1		
NY	1.00015	25506.5	1.08593	696.3	1.00007	72752.3	1.02001	3121.1
NC	1.00012	20428.8	1.04465	1463.2	1.00008	77151.6	1.03267	1938.9
ND								
OH	1	40349.8	1.04475	1415.6	0.999999	143535	1.01950	3294.4
OK	1.00027	22396.2	1.12866	488.9	1.00013	66426.5	1.00780	12631.9
OR	1.00015	50394.5			1.00000	137612.0		
PA	1.00016	16996.6	1.06915	892.7	1.00007	93319.3	1.01360	6143.2
RI	1.00014	14566.3			1.00006	58658.8		
SC	1.00032	16261.9	1.08331	769.2	1.00001	80159.9	1.04219	1449.7
SD	1.00012	43806.1			1.00004	50200.9		
TN	1.00011	19834.9	1.11814	543.6	1.00001	140019	1.05512	1069.4
TX	1.00011	23232.5	1.01942	4100.9	1.00003	190915		
UT	1.0002	29949.4			1.00001	110132		
VT					1.00001	171445		
VA	1.00014	23230.8	1.04422	1397.1	1	107197	1.07987	755.4
WA	1.00004	77180.7			1.00003	73462.3		
WV	1.20887	13.0			1	237573		
WI	1.00003	29893.9	1.02260	3215.7	1.00004	134723	1.10688	580.7
WY	1.08295	31.8			1.00018	37894		

Table B.1: **Convergence Tests for State Legislative District Ensembles.** Cells that are highlighted in red used block groups as the base unit of geography due, all others used precincts. Cells that are highlighted in gray represent ensembles for which the $\hat{R}$ computation is undefined, either because no plans were found that were balanced in population in our experimental setup, or because every plan within an ensemble contained the same number of MMDs. Cells that are highlighted in black represent legislative geographies that do not exist.

	State Lower House					State Upper House				
	Discrepancy ($\tau = 5\%$)		Critical Offset (Δ)			Discrepancy ($\tau = 5\%$)		Critical Offset (Δ)		
State	Observed	Predicted	Mean Δ	(StDev)	Predicted Δ	Observed	Predicted	Mean Δ	(StDev)	Predicted Δ
AL	63.280%	68.802%	0.535%	(0.045%)	0.540%	19.088%	17.617%	0.270%	(0.024%)	0.310%
AK	65.634%	70.735%	1.580%	(0.100%)	1.670%	26.436%	29.790%	0.940%	(0.044%)	0.890%
AZ	13.917%	13.464%	0.235%	(0.039%)	0.265%	14.068%	13.366%	0.240%	(0.020%)	0.255%
AR	87.448%	86.045%	0.955%	(0.057%)	1.090%	29.745%	26.677%	0.435%	(0.032%)	0.460%
CA	23.843%	20.163%	0.155%	(0.015%)	0.150%	8.144%	6.425%	0.100%	(0.000%)	0.100%
CO	53.002%	51.869%	0.550%	(0.059%)	0.600%	24.416%	22.247%	0.350%	(0.032%)	0.385%
CT	89.200%	96.267%	0.717%	(0.062%)	1.215%	22.958%	24.228%	0.370%	(0.040%)	0.400%
DE	58.165%	63.885%	0.970%	(0.060%)	1.320%	24.496%	26.760%	0.670%	(0.033%)	0.755%
FL	58.247%	53.510%	0.350%	(0.045%)	0.350%	12.441%	10.501%	0.140%	(0.020%)	0.150%
GA	90.185%	90.446%	0.600%	(0.050%)	0.735%	26.967%	26.758%	0.275%	(0.025%)	0.310%
HI										
ID	39.807%	40.598%	0.725%	(0.072%)	0.785%	39.992%	40.581%	0.725%	(0.075%)	0.765%
IL	70.243%	63.297%	0.415%	(0.039%)	0.475%	32.335%	26.399%	0.260%	(0.030%)	0.270%
IN	67.875%	66.219%	0.550%	(0.050%)	0.600%	30.481%	27.136%	0.320%	(0.033%)	0.335%
IA	79.268%	77.879%	0.765%	(0.050%)	0.865%	38.509%	35.739%	0.450%	(0.055%)	0.475%
KS	95.661%	96.008%	1.190%	(0.070%)	1.440%	37.253%	34.863%	0.555%	(0.035%)	0.565%
KY	56.490%	59.686%	0.480%	(0.046%)	0.500%	16.484%	15.380%	0.260%	(0.037%)	0.230%
LA	76.667%	77.106%	0.685%	(0.059%)	0.775%	24.970%	23.534%	0.330%	(0.033%)	0.350%
ME	95.432%	97.177%	1.060%	(0.162%)	1.315%	22.543%	26.362%	0.420%	(0.075%)	0.435%
MD	39.455%	40.802%	0.380%	(0.046%)	0.420%	23.880%	24.559%	0.280%	(0.024%)	0.320%
MA	91.418%	92.903%	0.760%	(0.070%)	0.915%	20.662%	20.909%	0.275%	(0.034%)	0.315%
MI	51.753%	51.264%	0.355%	(0.027%)	0.380%	11.829%	10.398%	0.150%	(0.000%)	0.160%
MN	77.355%	81.167%	0.655%	(0.061%)	0.690%	36.776%	39.524%	0.405%	(0.057%)	0.410%
MS	96.175%	94.029%	1.040%	(0.058%)	1.270%	47.674%	46.542%	0.600%	(0.032%)	0.640%
MO	92.414%	91.548%	0.705%	(0.057%)	0.845%	17.515%	14.543%	0.245%	(0.015%)	0.250%
MT	88.118%	91.207%	1.175%	(0.150%)	1.355%	47.250%	50.706%	0.820%	(0.046%)	0.740%
NE						51.177%	53.399%	0.735%	(0.059%)	0.830%
NV	47.556%	47.248%	0.750%	(0.045%)	0.825%	18.552%	17.459%	0.435%	(0.039%)	0.480%
NH						6.909%	11.953%	0.240%	(0.030%)	0.285%
NJ	21.662%	18.250%	0.265%	(0.039%)	0.265%	21.368%	18.262%	0.255%	(0.035%)	0.270%
NM	81.610%	82.258%	1.330%	(0.084%)	1.370%	52.029%	50.167%	0.835%	(0.045%)	0.860%
NY	69.468%	63.668%	0.345%	(0.027%)	0.380%	25.293%	20.091%	0.175%	(0.025%)	0.205%
NC	67.835%	69.229%	0.445%	(0.027%)	0.555%	22.654%	22.640%	0.260%	(0.037%)	0.285%
ND										
OH	52.874%	47.655%	0.325%	(0.040%)	0.365%	12.567%	9.318%	0.155%	(0.015%)	0.155%
OK	84.469%	84.903%	0.970%	(0.071%)	1.050%	40.591%	40.346%	0.540%	(0.037%)	0.570%
OR	41.871%	43.524%	0.530%	(0.051%)	0.535%	15.693%	15.504%	0.300%	(0.022%)	0.305%
PA	85.595%	86.451%	0.545%	(0.057%)	0.550%	18.750%	15.653%	0.170%	(0.024%)	0.195%
RI	86.942%	90.707%	1.795%	(0.117%)	1.730%	48.021%	52.211%	1.225%	(0.093%)	1.040%
SC	84.431%	85.243%	0.690%	(0.049%)	0.845%	29.123%	28.846%	0.350%	(0.032%)	0.400%
SD	54.947%	54.113%	1.020%	(0.064%)	1.110%	46.376%	50.573%	0.995%	(0.072%)	1.045%
TN	64.002%	62.194%	0.485%	(0.071%)	0.540%	14.440%	13.176%	0.225%	(0.025%)	0.230%
TX	62.132%	58.322%	0.310%	(0.030%)	0.350%	8.014%	5.564%	0.105%	(0.015%)	0.100%
UT	75.406%	74.807%	0.880%	(0.051%)	1.005%	25.152%	23.305%	0.425%	(0.040%)	0.460%
VT						4.730%	5.160%	0.290%	(0.020%)	0.200%
VA	57.649%	57.934%	0.405%	(0.042%)	0.485%	16.420%	15.990%	0.225%	(0.025%)	0.240%
WA	37.025%	32.980%	0.400%	(0.039%)	0.440%	38.125%	33.107%	0.400%	(0.000%)	0.445%
WV	66.127%	66.311%	0.750%	(0.047%)	0.880%	8.815%	8.598%	0.250%	(0.032%)	0.265%
WI	65.347%	64.443%	0.555%	(0.052%)	0.570%	16.543%	13.467%	0.230%	(0.033%)	0.225%
WY	100.000%	94.393%	2.175%	(0.144%)	2.645%	56.729%	57.540%	1.440%	(0.073%)	1.485%

Table B.2: **Empirical and Predicted Discrepancies and Critical Offsets for 93 Legislative Geographies.** Critical offsets are computed as described in Appendix B.1 (averaged over 10 runs). Columns labeled "Predicted" are computed by sampling discrepancies (with and without offsets) from the probabilistic model described in Section 4.5.2 using the empirical means and standard deviations of the corresponding legislative redistricting ensembles. Cells that are highlighted in gray represent states where no plans were found that were balanced in population in our experimental setup. Cells that are highlighted in red had precinct data that was incompatible with our experimental setup (either due to large precincts or a lack of complete precinct data) and were computed using block groups as the building blocks of districts instead. Cells that are highlighted in black represent legislative geographies that do not exist.

	Block Group/Precinct		Block	
	Discrepancy Rate	Critical Offset (%)	Discrepancy Rate	Critical Offset (%)
CT SH	89.20%	0.72%	98.58%	>2.0%
RI SH	86.94%	1.80%	97.62%	>2.0%
RI SS	48.02%	1.23%	66.14%	1.60%

Table B.3: **Using Blocks as the Base-Level Geounit.** Comparison between no offset discrepancy rate and critical offset for three geographies using precincts (Connecticut) or block groups (for Rhode Island legislative geographies) versus blocks as the smallest unit of district construction. We halted the computation of critical offsets at 2.0%.

	Population Balance		Majority-Minority Districts	
	Standard Constraints		Short Bursts	
State	Δ	DR ($\Delta = 0$)	Mean Discrep.	Discrep. Rate
AL	0.10%	1.67%	0.18	17.20%
AK				
AZ	0.10%	2.02%		
AR	0.10%	0.74%		
CA	0.13%	12.81%		
CO	0.11%	1.84%		
CT	0.10%	0.77%		
DE				
FL	0.12%	6.79%	0.10	10.10%
GA	0.10%	3.23%	0.47	44.40%
HI	0.10%	0.26%		
ID	0.10%	0.18%		
IL	0.12%	5.93%	0.15	13.70%
IN	0.10%	2.12%		
IA	0.10%	0.53%		
KS	0.10%	1.00%		
KY	0.10%	0.88%		
LA	0.10%	1.18%	0.15	15.30%
ME	0.10%	0.01%		
MD	0.10%	1.38%	0.19	18.40%
MA	0.10%	1.78%		
MI	0.10%	2.48%	0.32	27.90%
MN	0.10%	1.48%		
MS	0.10%	0.85%	0.01	0.90%
MO	0.10%	1.67%		
MT				
NE	0.10%	0.46%		
NV	0.12%	1.04%		
NH	0.10%	0.04%		
NJ	0.11%	3.51%	0.17	16.60%
NM	0.10%	0.64%		
NY	0.11%	6.80%	0.23	21.80%
NC	0.10%	3.25%	0.05	5.30%
ND				
OH	0.10%	3.95%	0.02	2.40%
OK	0.10%	1.22%		
OR	0.10%	0.78%		
PA	0.10%	4.28%	0.25	25.00%
RI	0.10%	0.31%		
SC	0.10%	1.54%	0.26	25.70%
SD				
TN	0.10%	1.98%	0.14	14.30%
TX	0.14%	9.08%	0.01	1.40%
UT	0.13%	0.86%		
VT				
VA	0.10%	2.14%	0.28	27.50%
WA	0.19%	3.16%		
WV	0.10%	0.42%		
WI	0.10%	1.80%		
WY				

Table B.4: **Results for Ensembles of Congressional District Plans.** Cells highlighted in black represent states that were apportioned a single congressional representative after the 2010 census, cells highlighted in gray represent geographies for which no majority-Black districts were found by our chains, and cells highlighted in red represent chains that were run using block groups as the base unit of geography due to missing precinct data. For each ensemble sampled with standard constraints, we calculate the critical offset (Δ) and the no-offset discrepancy rate (i.e. the proportion of plans exceeding a 5% population balance threshold under SWAP). Critical offsets are averaged over 10 runs. For each ensemble sampled using short burst optimization, we calculate the mean MMD discrepancy and the discrepancy rate (i.e. the proportion of plans with a non-zero MMD discrepancy).

	State Lower House		State Upper House	
State	Non-Zero Discrep. Rate	Mean Discrep.	Non-Zero Discrep. Rate	Mean Discrep.
AL				
AK				
AZ	32.5%	0.387	27.4%	0.336
AR				
CA	47.4%	0.601	40.1%	0.523
CO	58.0%	0.717	22.2%	0.234
CT	1.1%	0.011	0.0%	0.000
DE				
FL	27.1%	0.286	6.5%	0.066
GA	0.3%	0.003		
HI				
ID				
IL	32.2%	0.344	3.4%	0.034
IN				
IA				
KS	7.5%	0.076	5.2%	0.052
KY				
LA				
ME				
MD	0.8%	0.008	0.0%	0.000
MA	29.2%	0.312		
MI				
MN				
MS				
MO				
MT				
NE			0.0%	0.000
NV	35.1%	0.430	55.9%	0.664
NH				
NJ	20.6%	0.226	19.1%	0.206
NM	67.3%	1.093	56.4%	0.811
NY	26.7%	0.322	22.2%	0.238
NC				
ND				
OH				
OK	10.3%	0.087	7.9%	-0.079
OR				
PA	23.7%	0.253		
RI				
SC				
SD				
TN				
TX	57.2%	0.743	11.4%	0.118
UT				
VT				
VA				
WA	60.8%	0.719	55.7%	0.648
WV				
WI	16.9%	0.172		
WY				

Table B.5: **Results for Ensembles Optimized to Increase the Number of Majority-Hispanic Districts.** We show the mean majority-Hispanic district discrepancy and the discrepancy rate (i.e. the proportion of plans with a non-zero MMD discrepancy). Gray cells represent geographies for which our ensembles did not contain any majority-Hispanic districts in either DEMO or SWAP. Black cells represent legislative geographies that do not exist.

	Short Bursts			
	State Lower House		State Upper House	
State	Non-Zero Discrep. Rate	Mean Discrep.	Non-Zero Discrep. Rate	Mean Discrep.
AL	57.5%	0.822	25.3%	0.293
AK				
AZ				
AR	37.2%	0.470	54.2%	0.715
CA	0.9%	0.009		
CO				
CT	11.6%	0.126	6.1%	0.061
DE	11.7%	0.120	4.3%	0.045
FL	23.4%	0.256	7.6%	0.076
GA	65.6%	1.027	34.7%	0.433
HI				
ID				
IL	23.3%	0.270	8.9%	0.095
IN	31.5%	0.364	28.2%	0.291
IA				
KS	39.2%	0.415		
KY	2.4%	0.024		
LA	68.3%	1.122	43.7%	0.553
ME				
MD	17.6%	0.191	16.6%	0.181
MA	17.6%	0.181	5.4%	0.054
MI	11.3%	0.120	6.4%	0.067
MN				
MS	78.9%	1.431	53.7%	0.747
MO	32.9%	0.405	58.3%	0.790
MT				
NE			38.9%	0.389
NV				
NH				
NJ	11.0%	0.114	10.1%	0.107
NM				
NY	42.7%	0.533	17.5%	0.187
NC	49.2%	0.649	27.8%	0.315
ND				
OH	24.8%	0.289	24.3%	0.261
OK	34.7%	0.404	15.0%	0.150
OR				
PA	20.2%	0.220	14.3%	0.149
RI				
SC	71.3%	1.203	59.3%	0.847
SD				
TN	30.5%	0.325	26.0%	0.291
TX	9.2%	0.096		
UT				
VT				
VA	31.6%	0.367	53.1%	0.724
WA				
WV				
WI	20.3%	0.218	46.3%	0.493
WY				

Table B.6: **MMD Discrepancy Rates Across Geographies.** Mean majority-Black district discrepancy and non-zero discrepancy rate for short-burst optimized ensembles across 52 state legislative geographies. Gray cells represent geographies for which our ensembles did not contain any majority-Black districts in either DEMO or SWAP. Black cells represent legislative geographies that do not exist. Red cells represent chains that were run using block groups as the base unit of geography due to missing precinct data.

REFERENCES

John Abowd, Robert Ashmead, Ryan Cumings-Menon, Simson Garfinkel, Micah Heineck, Christine Heiss, Robert Johns, Daniel Kifer, Philip Leclerc, Ashwin Machanavajjhala, Brett Moran, William Sexton, Matthew Spence, and Pavel Zhuravlev. The 2020 Census Disclosure Avoidance System TopDown Algorithm. *Harvard Data Science Review*, Special Issue 2, June 24 2022. https://hdsr.mitpress.mit.edu/pub/7evz361i.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018.

John Langshaw Austin. *How to do things with words*. Harvard university press, 1975.

Eric Autry, Daniel Carter, Gregory J. Herschlag, Zach Hunter, and Jonathan C. Mattingly. Metropolized forest recombination for monte carlo sampling of graph partitions. *SIAM Journal on Applied Mathematics*, 83(4):1366–1391, 2023a. doi:10.1137/21M1418010. URL `https://doi.org/10.1137/21M1418010`.

Eric Autry, Daniel Carter, Gregory J Herschlag, Zach Hunter, and Jonathan C Mattingly. Metropolized forest recombination for monte carlo sampling of graph partitions. *SIAM Journal on Applied Mathematics*, 83(4):1366–1391, 2023b.

Yatong Bai, Mo Zhou, Vishal M. Patel, and Somayeh Sojoudi. MixedNUTS: Training-free accuracy-robustness balance via nonlinearly mixed classifiers. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL `https://openreview.net/forum?id=pyD6cujUmL`.

Can Bakiskan, Metehan Cekic, and Upamanyu Madhow. Early layers are more important for adversarial robustness. In *ICLR 2022 workshop on new frontiers in adversarial machine learning*, 2022.

María Ballesteros, Cynthia Dwork, Gary King, Conlan Olson, and Manish Raghavan. Evaluating the impacts of swapping on the us decennial census. In *Proceedings of the Symposium on Computer Science and Law*, CSLAW '25, page 64–76. ACM, March 2025. doi:10.1145/3709025.3712210. URL `http://dx.doi.org/10.1145/3709025.3712210`.

Jim Banks. Senator Banks Calls for Investigation into 2020 Census Miscounts and Data Integrity, October 2025. `https://perma.cc/2BE2-YGHK`.

Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. *Advances in Neural Information Processing Systems*, 34, 2021.

Zohar Barak and Inbal Talgam-Cohen. Stochastic knapsack: Semi-adaptivity gaps and improved approximation. *arXiv preprint arXiv:2602.24042*, 2026.

Amariah Becker, Moon Duchin, Dara Gold, and Sam Hirsch. Computational redistricting and the voting rights act. *Election Law Journal: Rules, Politics, and Policy*, 20(4):407–441, 2021.

Anand Bhalgat, Ashish Goel, and Sanjeev Khanna. Improved approximation results for stochastic knapsack problems. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1647–1665. SIAM, 2011.

Battista Biggio, Igino Corona, Blaine Nelson, Benjamin IP Rubinstein, Davide Maiorca, Giorgio Fumera, Giorgio Giacinto, and Fabio Roli. Security evaluation of support vector machines in adversarial environments. In *Support Vector Machines Applications*, pages 105–153. Springer, 2014.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE symposium on security and privacy (SP)*, pages 141–159. IEEE, 2021.

danah boyd and Jayshree Sarathy. Differential Perspectives: Epistemic Disconnects Surrounding the U.S. Census Bureau's Use of Differential Privacy. *Harvard Data Science Review*, Special Issue 2, June 24 2022. https://hdsr.mitpress.mit.edu/pub/3vj5j6i0.

California State Legislature. California Consumer Privacy Act of 2018, 2018. URL `https://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?division=3.&part=4.&lawCode=CIV&title=1.81.5`.

Sarah Cannon, Ari Goldbloom-Helzner, Varun Gupta, JN Matthews, and Bhushan Suwal. Voting rights, Markov chains, and optimization by short bursts. *Methodology and Computing in Applied Probability*, 25(1):36, 2023.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pages 463–480. IEEE, 2015.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE symposium on security and privacy (SP)*, pages 1897–1914. IEEE, 2022a.

Nicholas Carlini, Matthew Jagielski, Chiyuan Zhang, Nicolas Papernot, Andreas Terzis, and Florian Tramer. The privacy onion effect: Memorization is relative. *Advances in Neural Information Processing Systems*, 35:13263–13276, 2022b.

Jowei Chen, Jonathan Rodden, et al. Unintentional gerrymandering: Political geography and electoral bias in legislatures. *Quarterly Journal of Political Science*, 8(3):239–269, 2013.

Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. When machine unlearning jeopardizes privacy. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, pages 896–911, 2021.

Ruizhe Chen, Jianfei Yang, Huimin Xiong, Jianhong Bai, Tianxiang Hu, Jin Hao, Yang Feng, Joey Tianyi Zhou, Jian Wu, and Zuozhu Liu. Fast model debias with machine unlearning. *Advances in Neural Information Processing Systems*, 36:14516–14539, 2023.

Sang Keun Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Minsoo Kang, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, et al. What is your data worth to gpt? llm-scale data valuation with influence functions. *arXiv preprint arXiv:2405.13954*, 2024.

Rishav Chourasia and Neil Shah. Forget unlearning: Towards true data-deletion in machine learning. In *International conference on machine learning*, pages 6028–6073. PMLR, 2023.

Miranda Christ, Sarah Radway, and Steven M Bellovin. Differential privacy and swapping: Examining de-identification's impact on minority representation and privacy preservation in the us census. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 457–472. IEEE, 2022.

Christian Cianfarani and Aloni Cohen. Understanding and mitigating the impacts of differentially private census data on state-level redistricting. *Election Law Journal: Rules, Politics, and Policy*, page 15331296261434381, 2026.

Christian Cianfarani, Arjun Nitin Bhagoji, Vikash Sehwag, Ben Zhao, Heather Zheng, and Prateek Mittal. Understanding robust learning through the lens of representation similarities. *Advances in Neural Information Processing Systems*, 35:34912–34925, 2022.

Carmen Cirincione, Thomas A Darling, and Timothy G O'Rourke. Assessing South Carolina's 1990s congressional districting. *Political Geography*, 19(2):189–211, 2000. ISSN 0962-6298. doi:https://doi.org/10.1016/S0962-6298(99)00047-5. URL `https://www.sciencedirect.com/science/article/pii/S0962629899000475`.

Aloni Cohen, Moon Duchin, JN Matthews, and Bhushan Suwal. Private numbers in public policy: Census, differential privacy, and redistricting. *Harvard Data Science Review*, Special Issue 2, 2022.

Aloni Cohen, Adam Smith, Marika Swanberg, and Prashant Nalini Vasudevan. Control, confidentiality, and the right to be forgotten. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 3358–3372, 2023.

Aloni Cohen, Refael Kohen, Kobbi Nissim, and Uri Stemmer. Protecting the undeleted in machine unlearning. *arXiv preprint arXiv:2602.16697*, 2026.

Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors,

Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of *Proceedings of Machine Learning Research*, pages 1310–1320. PMLR, 2019. URL http://proceedings.mlr.press/v97/cohen19c.html.

Francesco Croce and Matthias Hein. Provable robustness against all adversarial l_p-perturbations for $p\geq 1$. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020a. URL https://openreview.net/forum?id=rklk_ySYPB.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020b.

Francesco Croce and Matthias Hein. Adversarial robustness against multiple l_p-threat models at the price of one and how to quickly fine-tune robust models to another threat model. *arXiv:2105.12508 [cs]*, May 2021. URL http://arxiv.org/abs/2105.12508.

Francesco Croce and Matthias Hein. Adversarial robustness against multiple and single l_p-threat models via quick fine-tuning of robust classifiers. In *International Conference on Machine Learning*, pages 4436–4454. PMLR, 2022.

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

Adrián Csiszárik, Péter Kőrösi-Szabó, Ákos Matszangosz, Gergely Papp, and Dániel Varga. Similarity and matching of neural network representations. *Advances in Neural Information Processing Systems*, 34, 2021.

Sihui Dai, Saeed Mahloujifar, and Prateek Mittal. Formulating robustness against unforeseen attacks. *arXiv preprint arXiv:2204.13779*, 2022.

Sihui Dai, Saeed Mahloujifar, Chong Xiang, Vikash Sehwag, Pin-Yu Chen, and Prateek Mittal. MultiRobustBench: Benchmarking robustness against multiple attacks. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 6760–6785. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/dai23c.html.

Sihui Dai, Chong Xiang, Tong Wu, and Prateek Mittal. Position paper: Beyond robustness against single attack types. *arXiv preprint arXiv:2405.01349*, 2024.

Sihui Dai, Christian Cianfarani, Vikash Sehwag, Prateek Mittal, and Arjun Bhagoji. Adapting to evolving adversaries with regularized continual robust training. In *International Conference on Machine Learning*, pages 11954–12000. PMLR, 2025.

Brian C Dean, Michel X Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of Operations Research*, 33(4):945–964, 2008.

Daryl DeFord and Moon Duchin. Random walks and the universe of districting plans. In *Political Geometry: Rethinking Redistricting in the US with Math, Law, and Everything In Between*, pages 341–381. Springer, 2022.

Daryl DeFord, Moon Duchin, and Justin Solomon. Recombination: A Family of Markov Chains for Redistricting. *Harvard Data Science Review*, 3(1), March 31 2021. https://hdsr.mitpress.mit.edu/pub/1ds8ptxu.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Travis Dick, Cynthia Dwork, Michael Kearns, Terrance Liu, Aaron Roth, Giuseppe Vietri, and Zhiwei Steven Wu. Confidence-ranked reconstruction of census microdata from published statistics. *Proceedings of the National Academy of Sciences*, 120(8):e2218605120, 2023. doi:10.1073/pnas.2218605120. URL `https://www.pnas.org/doi/abs/10.1073/pnas.2218605120`.

Frances Ding, Jean-Stanislas Denain, and Jacob Steinhardt. Grounding representation similarity through statistical testing. *Advances in Neural Information Processing Systems*, 34, 2021.

Moon Duchin, Taissa Gladkova, Eugene Henninger-Voss, Ben Klingensmith, Heather Newman, and Hannah Wheelen. Locating the representational baseline: Republicans in massachusetts. *Election Law Journal: Rules, Politics, and Policy*, 18(4):388–401, 2019.

Kshitij Dwivedi, Jiahui Huang, Radoslaw Martin Cichy, and Gemma Roig. Duality diagram similarity: a generic framework for initialization selection in task transfer learning. In *European Conference on Computer Vision*, pages 497–513. Springer, 2020.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pages 265–284. Springer, 2006.

Richard L. Engstrom and John K. Wildgen. Pruning thorns from the thicket: An empirical test of the existence of racial gerrymandering. *Legislative Studies Quarterly*, 2(4):465–479, 1977. ISSN 03629805. URL `http://www.jstor.org/stable/439420`.

European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council, 2016. URL `https://data.europa.eu/eli/reg/2016/679/oj`.

Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, page 954–959, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450369794. doi:10.1145/3357713.3384290. URL https://doi.org/10.1145/3357713.3384290.

Benjamin Fifield, Michael Higgins, Kosuke Imai, and Alexander Tarr. Automated redistricting simulation using markov chain monte carlo. *Journal of Computational and Graphical Statistics*, 29(4):715–728, 2020a. doi:10.1080/10618600.2020.1739532. URL https://doi.org/10.1080/10618600.2020.1739532.

Benjamin Fifield, Kosuke Imai, Jun Kawahara, and Christopher T Kenny. The essential role of empirical validation in legislative redistricting simulation. *Statistics and Public Policy*, 7(1):52–68, 2020b.

Simson Garfinkel, John M Abowd, and Christian Martindale. Understanding database reconstruction attacks on public data: These attacks on statistical databases are no longer a theoretical danger. *Queue*, 16(5):28–53, 2018.

Paul Gavrikov and Janis Keuper. Adversarial robustness through the lens of convolutional filters. *arXiv preprint arXiv:2204.02481*, 2022.

Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian Data Analysis, Third Edition*. CRC Press, 2013.

Sara Ghazanfari, Siddharth Garg, Prashanth Krishnamurthy, Farshad Khorrami, and Alexandre Araujo. R-lpips: An adversarially robust perceptual similarity metric. *arXiv preprint arXiv:2307.15157*, 2023.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pages 2242–2251. PMLR, 2019.

Amirata Ghorbani, Michael Kim, and James Zou. A distributional framework for data valuation. In *International Conference on Machine Learning*, pages 3535–3544. PMLR, 2020.

Dar Gilboa and Guy Gur-Ari. Wider networks learn better features. *arXiv preprint arXiv:1909.11572*, 2019.

Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making ai forget you: Data deletion in machine learning. *Advances in neural information processing systems*, 32, 2019.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9304–9312, 2020.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020.

Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy A Mann. Improving robustness using generated data. *Advances in Neural Information Processing Systems*, 34:4218–4233, 2021.

Diego Gragnaniello, Francesco Marra, Luisa Verdoliva, and Giovanni Poggi. Perceptual quality-preserving black-box attack against deep learning image classifiers. *Pattern Recognition Letters*, 147:142–149, 2021. ISSN 0167-8655. doi:https://doi.org/10.1016/j.patrec.2021.03.033. URL `https://www.sciencedirect.com/science/article/pii/S0167865521001288`.

Tom George Grigg, Dan Busbridge, Jason Ramapuram, and Russ Webb. Do self-supervised and supervised methods learn similar visual representations? *arXiv preprint arXiv:2110.00528*, 2021.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.

Anupam Gupta, Ravishankar Krishnaswamy, Marco Molinaro, and Ramamoorthi Ravi. Approximation algorithms for correlated knapsacks and non-martingale bandits. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 827–836. IEEE, 2011.

Varun Gupta, Christopher Jung, Seth Neel, Aaron Roth, Saeed Sharifi-Malvajerdi, and Chris Waites. Adaptive machine unlearning. *Advances in Neural Information Processing Systems*, 34:16319–16330, 2021.

David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 2004.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Kieran Healy. The performativity of networks. *European Journal of Sociology / Archives Européennes de Sociologie / Europäisches Archiv für Soziologie*, 56(2):175–205, 2015. ISSN 00039756, 14740583. URL `https://www.jstor.org/stable/26573206`.

Jeremy Howard. imagenette. `https://github.com/fastai/imagenette`, 2020.

Hanxun Huang, Yisen Wang, Sarah Erfani, Quanquan Gu, James Bailey, and Xingjun Ma. Exploring architectural ingredients of adversarially robust deep neural networks. *Advances in Neural Information Processing Systems*, 34:5545–5559, 2021.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32, 2019.

Ron S. Jarmin, John M. Abowd, Robert Ashmead, Ryan Cumings-Menon, Nathan Goldschlag, Michael B. Hawes, Sallie Ann Keller, Daniel Kifer, Philip Leclerc, Jerome P. Reiter, Rolando A. Rodríguez, Ian Schmutte, Victoria A. Velkoff, and Pavel Zhuravlev. An in-depth examination of requirements for disclosure risk assessment. *Proceedings of the National Academy of Sciences*, 120(43):e2220558120, 2023. doi:10.1073/pnas.2220558120. URL `https://www.pnas.org/doi/abs/10.1073/pnas.2220558120`.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas J Spanos, and Dawn Song. Efficient task-specific data valuation for nearest neighbor algorithms. *arXiv preprint arXiv:1908.08619*, 2019.

Charles Jin and Martin Rinard. Manifold regularization for locally stable deep neural networks. *arXiv preprint arXiv:2003.04286*, 2020.

Daniel Kang, Yi Sun, Dan Hendrycks, Tom Brown, and Jacob Steinhardt. Testing robustness against unforeseen adversaries. *arXiv preprint arXiv:1908.08016*, 2019.

Max Kaufmann, Daniel Kang, Yi Sun, Steven Basart, Xuwang Yin, Mantas Mazeika, Akul Arora, Adam Dziedzic, Franziska Boenisch, Tom Brown, et al. Testing robustness against unforeseen adversaries. *arXiv preprint arXiv:1908.08016*, 2019.

Christopher T Kenny, Shiro Kuriwaki, Cory McCartan, Evan TR Rosenman, Tyler Simko, and Kosuke Imai. The use of differential privacy for census data and its impact on redistricting: The case of the 2020 us census. *Science advances*, 7(41):eabk3283, 2021.

Christopher T. Kenny, Cory McCartan, Ben Fifield, and Kosuke Imai. *redist: Simulation Methods for Legislative Redistricting*, 2022. URL `https://alarm-redist.org/redist/`. R package.

Christopher T. Kenny, Shiro Kuriwaki, Cory McCartan, Evan T. R. Rosenman, Tyler Simko, and Kosuke Imai. Comment: The Essential Role of Policy Evaluation for the 2020 Census DisclosureAvoidance System. *Harvard Data Science Review*, Special Issue 2, January 31 2023. https://hdsr.mitpress.mit.edu/pub/6ffzuq19.

Christopher T Kenny, Cory McCartan, Shiro Kuriwaki, Tyler Simko, and Kosuke Imai. Evaluating bias and noise induced by the us census bureau's privacy protection methods. *Science Advances*, 10(18):eadl2524, 2024.

Shadie Khubba, Krista Heim, and Jinhee Hong. *National Census Coverage Estimates for People in the United States by Demographic Characteristics: 2020 Post-Enumeration Survey Estimation Report*. US Department of Commerce, US Census Bureau, 2022.

Hoki Kim, Woojin Lee, and Jaewook Lee. Understanding catastrophic overfitting in single-step adversarial training. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 8119–8127, 2021.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

Simon Kornblith, Ting Chen, Honglak Lee, and Mohammad Norouzi. Why do better loss functions lead to less transferable features? *Advances in Neural Information Processing Systems*, 34, 2021.

Nancy Krieger, Rachel C. Nethery, Jarvis T. Chen, Pamela D. Waterman, Emily Wright, Tamara Rushovich, and Brent A. Coull. Impact of differential privacy and census tract data source (decennial census versus american community survey) for monitoring health inequities. *American Journal of Public Health*, 111(2):265–268, 2021. doi:10.2105/AJPH.2020.305989. URL `https://doi.org/10.2105/AJPH.2020.305989`. PMID: 33351654.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009.

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

Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. Towards unbounded machine unlearning. *Advances in neural information processing systems*, 36: 1957–1987, 2023.

Cassidy Laidlaw and Soheil Feizi. Functional adversarial attacks. *Advances in neural information processing systems*, 32, 2019.

Cassidy Laidlaw, Sahil Singla, and Soheil Feizi. Perceptual adversarial robustness: Defense against unseen threat models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL `https://openreview.net/forum?id=dFwBosAcJkN`.

Na Li, Chunyi Zhou, Yansong Gao, Hui Chen, Zhi Zhang, Boyu Kuang, and Anmin Fu. Machine unlearning: Taxonomy, metrics, applications, challenges, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

Yezi Liu and Yanning Shen. Enabling group fairness in machine unlearning via distribution correction. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 1925–1935, 2025.

Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*, 2018.

Donald MacKenzie, Siu Leung-Sea, and Fabian Muniesa. *Do economists make markets?: on the performativity of economics*. Princeton University Press, 2020.

Divyam Madaan, Jinwoo Shin, and Sung Ju Hwang. Learning to generate noise for robustness against multiple perturbations. *CoRR*, abs/2006.12135, 2020. URL `https://arxiv.org/abs/2006.12135`.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL `https://openreview.net/forum?id=rJzIBfZAb`.

Pratyush Maini, Eric Wong, and J. Zico Kolter. Adversarial robustness against the union of multiple perturbation models. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 6640–6650. PMLR, 2020. URL `http://proceedings.mlr.press/v119/maini20a.html`.

Steven Manson, Jonathan Schroeder, David Van Riper, Tracy Kugler, and Steven Ruggles. IPUMS National Historical Geographic Information System: Version 17.0 [dataset]. Minneapolis, MN: IPUMS, 2022.

Cory McCartan and Kosuke Imai. Sequential monte carlo for sampling balanced and compact redistricting plans. *The Annals of Applied Statistics*, 17(4):3300–3323, 2023.

Cory McCartan, Christopher T Kenny, Tyler Simko, George Garcia III, Kevin Wang, Melissa Wu, Shiro Kuriwaki, and Kosuke Imai. Simulated redistricting plans for the analysis and evaluation of redistricting in the united states. *Scientific Data*, 9(1):689, 2022.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In Gordon H. Bower, editor, *Psychology of Learning and Motivation*, volume 24, pages 109–165. Academic Press, 1989. doi:https://doi.org/10.1016/S0079-7421(08)60536-8. URL `https://www.sciencedirect.com/science/article/pii/S0079742108605368`.

Laura McKenna. Disclosure avoidance techniques used for the 1970 through 2010 decennial censuses of population and housing. Technical report, U.S. Census Bureau, 2018. URL `https://www2.census.gov/ces/wp/2018/CES-WP-18-47.pdf`.

Merrill v. Milligan. Declarations of Moon Duchin (ECF 68-5) and William S. Cooper (ECF 48). in Supplemental Joint Appendix, On appeal from and writ of certiorari to the United States District Court for the Northern District of Alabama, Supreme Court of the United States, 2022. https://www.supremecourt.gov/DocketPDF/21/21-1086/221826/20220425150837756_42140%20pdf%20Bowdre%20IV%20Supplemental%20JA.pdf.

Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. *Advances in Neural Information Processing Systems*, 31, 2018.

Laura Fee Nern, Harsh Raj, Maurice Georgi, and Yash Sharma. On transfer of adversarial robustness from pretraining to downstream tasks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *Advances in neural information processing systems*, 33:512–523, 2020.

Quynh Nguyen and Matthias Hein. Optimization landscape and expressivity of deep cnns. In *International conference on machine learning*, pages 3730–3739. PMLR, 2018.

Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. *arXiv preprint arXiv:2010.15327*, 2020.

Alex Oesterling, Jiaqi Ma, Flavio Calmon, and Himabindu Lakkaraju. Fair machine unlearning: Data removal while mitigating disparities. In *International Conference on Artificial Intelligence and Statistics*, pages 3736–3744. PMLR, 2024.

Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In *International Conference on Machine Learning*, pages 4970–4979. PMLR, 2019.

ShengYun Peng, Weilin Xu, Cory Cornelius, Matthew Hull, Kevin Li, Rahul Duggal, Mansi Phute, Jason Martin, and Duen Horng Chau. Robust principles: Architectural design principles for adversarially robust cnns. In *34th British Machine Vision Conference 2023, BMVC 2023, Aberdeen, UK, November 20-24, 2023*. BMVA, 2023. URL https://papers.bmvc2023.org/0739.pdf.

Juan C. Perdomo, Tijana Zrnic, Celestine Mendler-Dünner, and Moritz Hardt. Performative prediction. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Samantha Petti and Abraham Flaxman. Differential privacy in the 2020 us census: what will it do? quantifying the accuracy/privacy tradeoff. *Gates open research*, 3, 2019.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgMkCEtPB.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in neural information processing systems*, 30, 2017.

Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34, 2021.

Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pages 8093–8104. PMLR, 2020.

Michael H Rothkopf. Scheduling with random service times. *Management Science*, 12(9): 707–713, 1966.

Parker Rule, Matthew Sun, and Bhushan Suwal. GerryChainJulia, March 2021. `https://doi.org/10.5281/zenodo.4649464`.

Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J Fleet. Adversarial manipulation of deep representations. *arXiv preprint arXiv:1511.05122*, 2015.

Mike Schneider. People, homes vanish due to 2020 census' new privacy method. *Associated Press*, Oct 2021. `https://apnews.com/article/religion-wisconsin-new-york-tampa-florida-68c96e7eb701da74ae7c8df3c3476705`.

J. R. Searle. A taxonomy of illocutionary acts. *Language Mind, and Knowledge*, 1975. URL `https://cir.nii.ac.jp/crid/1573105974577618048`.

Vikash Sehwag, Saeed Mahloujifar, Tinashe Handina, Sihui Dai, Chong Xiang, Mung Chiang, and Prateek Mittal. Robust learning meets generative models: Can proxy distributions improve adversarial robustness? In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=WVX0NNVBBkV`.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Ryan Steed, Terrance Liu, Zhiwei Steven Wu, and Alessandro Acquisti. Policy impacts of statistical uncertainty and privacy. *Science*, 377(6609):928–931, 2022.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

Tanmay Surve and Romila Pradhan. Explaining fairness violations using machine unlearning. In *Proceedings 28th International Conference on Extending Database Technology, EDBT*, pages 25–28, 2025.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6199.

Florian Tramèr and Dan Boneh. Adversarial training and robustness for multiple perturbations. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019a. URL https://arxiv.org/abs/1904.13000.

Florian Tramèr and Dan Boneh. Adversarial training and robustness for multiple perturbations. In *Advances in Neural Information Processing Systems*, pages 5866–5876, 2019b.

United States Census Bureau. *Geographic Areas Reference Manual*, chapter Voting Districts. United States Department of Commerce Washington DC, 1994. URL https://www2.census.gov/geo/pdfs/reference/GARM/Ch14GARM.pdf.

United States Census Bureau. Disclosure Avoidance for the 2020 Census: An Introduction, 2021. US Government Publishing Office Washington, DC.

United States Census Bureau. 2020 Census Redistricting Noisy Measurement File (NMF), June 2023. https://www.census.gov/newsroom/press-releases/2023/2020-redistricting-noisy-measurement-file.html.

US Census Bureau. Census Bureau Releases Estimates of Undercount and Overcount in the 2020 Census, March 2022. URL https://www.census.gov/newsroom/press-releases/2022/2020-census-estimates-of-undercount-and-overcount.html.

US Census Bureau. Redistricting Data Program, 2023a. https://www.census.gov/programs-surveys/decennial-census/about/rdo.html.

US Census Bureau. TIGER/Line Shapefiles, 2023b. https://www.census.gov/geo/maps-data/data/tiger-line.html.

US Department of Commerce. Disclosure Avoidance for Statistical Products (DAO 216-26). https://www.commerce.gov/opog/disclosure-avoidance-statistical-products, June 4 2026.

David Van Riper, Tracy Kugler, and Jonathan Schroeder. IPUMS NHGIS Privacy-Protected 2010 Census Demonstration Data, version 20210608 [Database]. Minneapolis, MN: IPUMS, 2020.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi:10.1038/s41592-019-0686-2.

Ivan Vulić, Edoardo Maria Ponti, Robert Litschko, Goran Glavaš, and Anna Korhonen. Probing pretrained language models for lexical semantics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7222–7240, Online, November 2020. Association for Computational Linguistics. doi:10.18653/v1/2020.emnlp-main.586. URL `https://aclanthology.org/2020.emnlp-main.586`.

Qian Wang, Yaoyao Liu, Hefei Ling, Yingwei Li, Qihao Liu, Ping Li, Jiazhong Chen, Alan Yuille, and Ning Yu. Continual adversarial defense. *arXiv preprint arXiv:2312.09481*, 2023a.

Zekai Wang, Tianyu Pang, Chao Du, Min Lin, Weiwei Liu, and Shuicheng Yan. Better diffusion models further improve adversarial training. In *International conference on machine learning*, pages 36246–36263. PMLR, 2023b.

Gus Wezerek and David Van Riper. Changes to the Census Could Make Small Towns Disappear. *The New York Times*, Feb 2020. URL `https://www.nytimes.com/interactive/2020/02/06/opinion/census-algorithm-privacy.html`.

Bartosz Wójcik, Paweł Morawiecki, Marek Śmieja, Tomasz Krzyżek, Przemysław Spurek, and Jacek Tabor. Adversarial examples detection and analysis with layer-wise autoencoders. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1322–1326. IEEE, 2021.

Eric Wong, Frank R. Schmidt, and J. Zico Kolter. Wasserstein adversarial examples via projected sinkhorn iterations. *CoRR*, abs/1902.07906, 2019. URL `http://arxiv.org/abs/1902.07906`.

Tommy Wright and Kyle Irimata. Empirical study of two aspects of the TopDown algorithm output for redistricting: Reliability & variability. Technical report, U.S. Census Bureau, 2021.

Boxi Wu, Jinghui Chen, Deng Cai, Xiaofei He, and Quanquan Gu. Do wider neural networks really help adversarial robustness? *Advances in Neural Information Processing Systems*, 34:7054–7067, 2021.

Kaiwen Wu, Allen Wang, and Yaoliang Yu. Stronger and faster wasserstein adversarial attacks. In *International Conference on Machine Learning*, pages 10377–10387. PMLR, 2020.

Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=HyydRMZC-.

Cihang Xie and Alan Yuille. Intriguing properties of adversarial training at scale. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HyxJhCEFDS.

Tom Yan and Ariel D Procaccia. If you like shapley then you'll love the core. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5751–5759, 2021.

Yaoqing Yang, Liam Hodgkinson, Ryan Theisen, Joe Zou, Joseph E Gonzalez, Kannan Ramchandran, and Michael W Mahoney. Taxonomizing local versus global structure in neural network loss landscapes. *Advances in Neural Information Processing Systems*, 34, 2021.

Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

Chaojian Yu, Bo Han, Li Shen, Jun Yu, Chen Gong, Mingming Gong, and Tongliang Liu. Understanding robust overfitting of adversarial training and beyond. In *International Conference on Machine Learning*, pages 25595–25610. PMLR, 2022.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Richard C. Wilson, Edwin R. Hancock, and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press, 2016. URL http://www.bmva.org/bmvc/2016/papers/paper087/index.html.

Jeff Zalesin. Beyond the adjustment wars: Dealing with uncertainty and bias in restistricting data. *Yale Law Journal Forum*, 130:186, 2020.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

Dawen Zhang, Shidong Pan, Thong Hoang, Zhenchang Xing, Mark Staples, Xiwei Xu, Lina Yao, Qinghua Lu, and Liming Zhu. To be forgotten or to be fair: Unveiling fairness implications of machine unlearning methods. *AI and Ethics*, 4(1):83–93, 2024.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In Kamalika

Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482. PMLR, 2019. URL http://proceedings.mlr.press/v97/zhang19p.html.

Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane S. Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=Skxuk1rFwB.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 586–595. Computer Vision Foundation / IEEE Computer Society, 2018. doi:10.1109/CVPR.2018.00068. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_The_Unreasonable_Effectiveness_CVPR_2018_paper.html.