Security, Privacy, and Transparency Guarantees for Machine Learning Systems

Mathias Lécuyer

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2019

ABSTRACT

Security, Privacy, and Transparency Guarantees for Machine Learning Systems

Mathias Lécuyer

Machine learning (ML) is transforming a wide range of applications, promising to bring immense economic and social benefits. However, it also raises substantial security, privacy, and transparency challenges. ML workloads indeed push companies toward aggressive data collection and loose data access policies, placing troves of sensitive user information at risk if the company is hacked. ML also introduces new attack vectors, such as adversarial example attacks, which can completely nullify models' accuracy under attack. Finally, ML models make complex data-driven decisions, which are opaque to the end-users, and difficult to inspect for programmers. In this dissertation we describe three systems we developed. Each system addresses a dimension of the previous challenges, by combining new practical systems techniques with rigorous theory to achieve a guaranteed level of protection, and make systems easier to understand. First we present Sage, a differentially private ML platform that enforces a meaningful protection semantic for the troves of personal information amassed by today's companies. Second we describe PixelDP, a defense against adversarial examples that leverages differential privacy theory to provide a guaranteed level of accuracy under attack. Third we introduce Sunlight, a tool to enhance the transparency of opaque targeting services, using rigorous causal inference theory to explain targeting decisions to end-users.

Contents

Introduction **1**

The Shift to ML-Driven Code 2

Security, Privacy, and Transparency Implications 3

Focus of This Dissertation 6

Chapter 1 Data Protection and Privacy in Machine Learning Systems **9**

1.1 Motivation 10

1.2 The Sage System 11

1.3 Background 13

1.4 Design 18

1.5 Theory 38

1.6 Evaluation 47

1.7 Related Work 59

1.8 Summary 60

Chapter 2 Security Guarantees Against Adversarial Examples **62**

2.1 Motivation 63

2.2 The PixelDP Defense 64

2.3 Background 66

2.4 Design 71

2.5 Theory 77

2.6 Evaluation 81

2.7 Related Work 93

2.8 Summary 96

Chapter 3 Data Use Transparency in Opaque Targeting Services **98**

3.1 Motivation 99

3.2 The Sunlight System 100

3.3 Background 103

3.4 Design 106

3.5 Theory 113

3.6 Evaluation 122

3.7 Related Work 142

3.8 Summary 142

Conclusion **144**

Bibliography **147**

Acknowledgements

This dissertation would not have been possible without my advisors and mentors, Roxana Geambasu, Augustin Chaintreau, Daniel Hsu, and Siddhartha Sen. I thank you for your guidance and support, both human and intellectual. In particular my main advisor Roxana Geambasu created an environment supportive of family life, and relentlessly fostered my academic career. For this I am especially grateful.

I also thank my co-authors, Vaggelis Atlidakis, Jonathan Bell, Augustin Chaintreau, Guillaume Ducoffe, Roxana Geambasu, Daniel Hsu, Tzu-Kuo Huang Suman Jana, Francis Lan, Joshua Lockerman, Lamont Nelson, Jason Nieh, Andrei Papancea, Theofilos Petsios, Siddhartha Sen, Amit Sharma, Aleksandrs Slivkins, Riley Spahn, Giannis Spiliopoulos Max Tucker, Nicolas Viennot, and Kiran Vodrahalli, for their work and the dynamic research environment they helped create. I thank the members of my thesis committee, David Blei, Augustin Chaintreau, Roxana Geambasu, Daniel Hsu, and Siddhartha Sen, for valuable discussions and feedback. I also acknowledge the funding that made this work possible: NSF grants CNS-1514437 and CNS-1351089.

Last, but not least, I couldn't thank my wife and two daughters enough, for their love during this endeavor, their patience with my travels and research induced absent-mindedness, their support both moral and material. I am also obliged to my wife for numerous discussions that pushed my work forward, introduced me to new ideas, and brought clarity to my research and writing.

To Joséphine, Eléa, and Zélie.

Introduction

The recent machine learning (ML) revolution, powered by new systems supporting massive amounts of data and advanced algorithms to analyse them, is transforming a wide range of applications. These include search, personalized recommendations, and smart assistants, as well as security and safety-critical applications, such as self-driving cars, face recognition based access control, and fraud detection. However, ML-driven applications also introduce disruptive workloads and atypical semantics that do not fit existing system paradigms. Adversaries can manipulate model predictions without explicitly breaking into host machines. Traditional protection abstractions from operating systems are ill-equipped to handle the new data access patterns of ML workloads, hence being ineffective at safeguarding the vast troves of collected data. And ML models are often too opaque for end-users to understand and trust, and too brittle for system designers security requirements. These challenges pose serious barriers to reaching ML's full potential.

We propose that by combining techniques from systems and theory, we can *design ML systems that are both practical and provide provable guarantees of security, privacy, and transparency*. Three steps are involved. First, components requiring strong guarantees should be separated from those that can do with best-effort, heuristic operation. This separation enables designs that are practical but still guarantee clear semantics. Second, each component requiring rigorous guarantees is matched to the theoretical field best suited as a foundation for a solution, either within theoretical computer science or from other domains. Third, building blocks are assembled into a coherent architecture to implement a prototype, evaluated against multiple workloads and at scale to understand its strengths and limitations.

This thesis describes abstractions, tools, and systems that leverage this approach to enhance the security, robustness, and transparency in ML-powered systems. We start with background on the shift to ML-driven code and the challenges it raises in security, privacy, and transparency. We then focus on the specific aspects of these challenges we will address, before detailing each contribution in its dedicated chapter.

The Shift to ML-Driven Code

Three recent trends are combining to make Machine Learning (ML) ubiquitous. First, enormous amounts of data is constantly generated and collected. The internet started this trend by connected data generating devices and sensors to means of storage and analysis. More recent technological developments such as mobile phones, social media, or the internet of things accelerated this trend. Second, recent advances in ML, and in particular deep learning, leverage this data to enable great performance at many tasks, such as processing and understanding images, voice, or text, personalizing content, and performing data-driven optimization of decisions. Third, maturing systems support gives broad access to the tools and hardware necessary to process these troves of data, train state-of-the-art models, and deploy them around the world.

Because of its success, ML is now used in many domains and applications. The best known applications may be those of user facing online services, such as ad targeting, search, price optimization, content personalization, content recommendation, or voice based smart assistants. But these are only the tip of the iceberg. ML is also used in less visible applications, from cloud infrastructure optimization, to fraud detection, malware detection, or face recognition based authentication. ML is even used in decisions with important societal implications, for instance to sort applications in hiring, personalize education, prioritize patients for medical interventions, and make bail recommendations for prisoners.

This ubiquity of ML deployments is changing the origin and makeup of the code driving many of our applications, services, and devices. Traditional code is written by programmers and encodes algorithms that express business logic and configuration. While this code can be complex, its ideal behavior is specified in advance, and the implementation includes the entire logic. Traditional

code can thus be understood by humans, and audited for incorrect behavior. ML is a drastic shift from this paradigm: it is used when one cannot specify the desired behavior of the "code" in advance. This happens when there are too many tasks or variables to handle, such as in large scale personalization or price optimization, or when tasks are too complex and not well understood, such as in autonomous driving or speech recognition. To enable these applications part of the "code" is learned using: an ML model, an optimization algorithm, and *a lot of data*. Data is leveraged both at training time to learn the "code", and at prediction time to run the "code". Often, this data comes from users which means that it can be of personal nature (e.g. emails, searches, website visits, locations, heartbeats), and it can be manipulated.

Security, Privacy, and Transparency Implications

The shift to ML-driven "code" has profound security, privacy, and transparency implications.

Increased Data Exposure: A looming security and privacy threat posed by ML-driven applications is the vast troves of personal information that organizations routinely collect to fuel them. ML applications indeed have blurry data requirements: data collected to improve a news recommendation service may also be relevant for ad targeting, a smart assistant, or fraud detection. Because data requirements are unclear many organizations collect all the data they can even if no clear use-case exists yet. They also adopt wide-access policies for this data, such as the "data lake" policy: all data collected from the company's various products is integrated into one central repository, to which all employees and services get access. Improperly protected, this data can be grabbed by hackers or unethical employees – and frequently is, as countless data breach reports demonstrate.

Moreover, ML models increase exposure because of their relationship with the data. ML models indeed incorporate the sensitive data they train on, but are often handled as traditional, secret-free code would be. ML platforms, such as Google's Tensorflow-Extended (TFX), Facebook's FBLearner, and Uber's Michelangelo, routinely push models trained over sensitive data to servers all around the world[13;79;116;162] and sometimes to end-user devices[198;178] for faster predictions. Some companies also report pushing feature models – such as user embedding vectors and statis-

tics of user activity – into shared model stores that are widely accessible within the company, even though they may not enable such wide access for the data itself[79;116;172].

There is perhaps a sense that, because ML models aggregate data from multiple users, they "obfuscate" individuals' data sufficiently to warrant weaker protection than that of the data itself. However, that perception is succumbing to increasing evidence that ML models leak substantial information about their training sets. Carlini, et.al.[28] showed that language models over users' emails leak secrets – such as passwords, social security numbers, and credit card numbers – that users often include in their communications. Shokri, et.al.[173] showed that membership in a training set can be inferred from ML models even when the attacker only has access to the model's external predictions. Calandrino, et.al.[26] showed recommenders leak information across users. Finally, it has long been established both theoretically and empirically that access to too many linear statistics from a dataset – as an adversary might have due to periodic releases of ML code which often includes statistics used for featurization – is *fundamentally non-private*[9;48;65;83].

New Attack Vectors. In addition to leaking training set information, leaving logic decisions to trained ML models opens new attacks surfaces. In traditional appliactions, an adversary triggering behavior outside of the code's specification by crafting specific inputs is concidered a major vulnerability. Examples include gaining priviledged access to protected data, executing code of the adversary's chosing on remote machines, or triggering actions under another user's credentials with cross site scripting. In ML-driven applications, code is learned from data and inputs influence the program's behavior by design. Since these changes are part of the expected behavior, it is hard to defend against malicious users trying to influence the code through data, opening new attack surfaces in two broad phases of the applications' life cycle[147].

First, in many applications users have some control over the data used at prediction time (e.g tection, text in spam filtering, the binary in malware detection, part of the image is face recognition, sound in voice recognition). An adversary can manipulate these inputs at prediciton time, creating small perturbations to correctly classified inputs that cause a DNN to produce an erroneous prediction, possibly of the adversary's choosing[180]. These adversarial examples pose serious threats

to security-critical applications. Since the initial demonstration of adversarial examples, attack methodologies have been refined to become much more efficient[30;6], demonstrated on physical objects[7], and extended to other applications such as voice recognition[159] as well as other ML tasks such as reinforcement learning[101] and generative models[100].

Second, an attacker can sometimes influence the training data (e.g. through crowdsourcing, data sharing, or user actions) with the aim of corrupting the learned model. In this broad class of attacks, called data poisoning, the goal can be to degrade accuracy, or to ensure that targeted data features will be assigned a specific label of the attackers' chosing at prediction time. After being introduced for support verctor machines in[16], it has then been the focus of intense research with extensions to clustering[17], topic modeling[131], collaborative filtering[115], or neural networks[204].

Opaque Decisions Applied at Large Scale. Finally, ML applications introduce a new scale in the complexity of software systems. In traditional applications code is written by software developers. This requires significant efforts, which has two consequences. First, the logic typically follows specifications and is built to be as simple as possible and understandable by humans. Code can thus be audited to understand its behavior, and tested against the specifications. Second, the same logic is deployed for every user with minimal personalization, as manual fine grained personalization is not realistic. In ML applications on the other hand, the code in learned from large amounts of data. Because data is constantly updated, can be complex and noisy, and can be user generated, ML code is much more complex and hard to audit. And since the code can automatically be optimized for different users and population segments, it is possible to personalize the logic at scale, resulting in a context dependent behavior. This paradigm shift has three important technical and societal ramifications. First, the complexity of ML models and the data they are trained on makes it hard to interpret and explain their decisions. This poses difficulties both by making ML-driven applications increasingly hard to understand and debug[169;88;4], and by preventing applications from providing justifications and transparency for their decisions[209].

Second, this lack of transparency is turning the Web into an opaque and privacy-insensitive environment. As ML-driven applications exploit data for varied and undisclosed purposes, per-

sonalizing decisions to each user, it is increasingly hard for users and investigators (such as FTC agents, journalists, or researchers) to answer questions such as: Who has what data, and for what purposes is it used? Who shares data with whom? Does the service adhere to its own privacy policy? Examples of problematic practices abound, highlighting the need for more transparency. Google was found to have used institutional emails from ad-free Google Apps for Education to target ads in users' personal accounts [73;167]. MySpace was found to have violated its privacy policy by leaking personally identifiable information to advertisers[102]. Several consumer sites, such as Orbitz and Staples, were found to have adjusted their product pricing based on user location[122;187]. Large tech companies were found to enable questionable targeting[91], or participate in undisclosed data sharing deals[134].

Third, deploying hard to audit data driven code in increasingly impactful applications such as hiring[89], insurance pricing[92], or policing[156;140], can introduce and perpetuate discriminatory or otherwise unfair practices. These effects can be hard to measure, as they can affect specific subpopulations or individuals even when ML models have a great average performance over the entire population. These unfair behaviors can come from existing biases encoded in the data[21;27], result from optimizing in average over the population[37;25], be caused by the need for exploration in the data gathering process[18;160], or follow from strategic adaptation to the deployed algorithms[137].

Focus of This Dissertation

In this dissertation, we focus on one specific aspect of each of the previous challenges. In each case, we build systems and tools mixing both theory and systems mechanisms to alleviate the adverse impact of the shift to ML-driven code. Through this work, we hope to enable the promises of ML driven ecosystems without imposing undue risks.

First, Chapter 1 focuses on the increased data exposure challenge brought by the switch to ML code. Focusing on reducing the exposure of data through the release of ML models, we make the case that in ML systems, privacy and data protection are intertwined. To reduce and bound the data exposure coming from ML models leaking information about their training data, we leverage differential privacy (DP) theory, which offers strong guarantees against such leaks.

However, differential privacy is not directly applicable to entire ML workloads. We thus designed and built an ML platform[111] that enforces a global DP guarantee across all models produced from a constantly updating data stream. To make this possible, we use and extend DP theory, proposing a new model of interaction with the data more adapted to the requirements of ML workloads. To alleviate differential privacy's adversarial impact on models' utility, we devise a heuristic, but practical, privacy resource allocation mechanism, and leverage theory form statistics to ensure that only high quality models are released. We also leverage training set minimization techniques from the ML literature to create DP summaries of past data[109]. These summaries can be leveraged to provide high utility despite the privacy constraints.

Second, Chapter 2 considers the new attack vectors introduced by ML. Specifically, it describes a new defense strategy[106] against adversarial examples, an ML-specific class of attacks in which the adversary finds a small perturbation to an input which results in a prediction of the adversary's choosing. This defense leverages differential privacy, a seemingly unrelated theory from the privacy domain, to provide provable robustness guarantees against adversarial examples. To ensure scalability, a crucial property given the trend of using larger and larger models, we devise a training technique that empirically maximizes models' utility while maintaining the strong robustness guarantees. This approach yields the first certified defense that both scales to large networks and datasets (such as Google's Inception network for ImageNet) and applies broadly to arbitrary model types. Moreover, this approach enables a firewall-like security architecture, in which a small model is prepended to an existing, already trained one to make it more robust. Such an architecture is common in traditional software systems but unique for ML workloads.

Third, Chapter 3 concentrates on a facet of the challenge of ML's opaque decisions. More precisely, it presents transparency tools[108;110] that provide a new visibility into how ML-driven web services use end-users' data for targeting. These tools enable large scale randomized experiments that collect and analyse data from the studied services to deduce which user inputs cause which outputs (e.g. ads, recommendations). The data collection and analysis is decomposed in a multi-stage pipeline. Crucial stages leverage theory from causal inference and statistics to audit the impact of

specific pieces of user data on targeting, giving strong causal semantics and measures of statistical confidence. Other stages, such as the targeting detection stage, leverage sparsity assumptions and filtering heuristics to provide scalability without compromising the strong semantics.

The remainder of this dissertation details each contribution, an evaluation of the corresponding system, and the relevant related work.

Chapter 1

Data Protection and Privacy in Machine Learning Systems

This chapter focuses on the information that ML models leak about their training data, an aspect of the first challenge brought by the switch to ML, increased data exposure. It draws heavily on our paper describing Sage[111], an ML platform that enforces a global differential privacy guarantee over entire ML workloads, to bound the data exposure through all released models. It also incorporates parts of Pyramid[109], in which we introduced training set reduction techniques as a mechanism to reduce access to past data.

1.1 Motivation

Machine learning (ML) is changing the origin and makeup of the code driving many of our applications, services, and devices. Traditional code is written by programmers and encodes algorithms that express business logic, plus a bit of configuration data. We keep sensitive data – such as passwords, keys, and user data – out of our code, because we often ship this code to untrusted locations, such as end-user devices and app stores. We mediate accesses to sensitive data with access control. When we do include secrets in code, or when our code is responsible for leaking user information to an unauthorized entity (e.g., through an incorrect access control decision), it is considered a major vulnerability.

With ML, "code" – in the form of ML models – is learned by an ML platform from *a lot of training data*. Learning code enables large-scale personalization, as well as powerful new applications like autonomous cars. Often, the data used for learning comes from users and is of personal nature, including emails, searches, website visits, heartbeats, and driving behavior. And although ML "code" is derived from sensitive data, it is often handled as secret-free code. ML platforms, such as Google's Tensorflow-Extended (TFX), routinely push models trained over sensitive data to servers all around the world[13;79;116;162] and sometimes to end-user devices[198;178] for faster predictions. Some companies also push feature models – such as user embedding vectors and statistics of user activity – into model stores that are often times widely accessible within the companies[79;116;172]. Such exposure would be inconceivable in a traditional application. Think of a word processor: it might push *your* documents to *your* device for faster access, but it would be outrageous if it pushed *your* documents to *my* (and everyone else's) device!

There is perhaps a sense that because ML models aggregate data from multiple users, they "obfuscate" individuals' data and warrant weaker protection than the data itself. However, this perception is succumbing to growing evidence that ML models can leak specifics about individual entries in their training sets. Language models trained over users' emails for auto-complete have been shown to encode not only commonly used phrases but also social security numbers and credit card numbers that users may include in their communications[28]. Prediction APIs have been shown to be enable testing for membership of a particular user or entry within a training set[173]. Finally, it has long been established both theoretically and empirically that access to too many linear statistics from a dataset – as an adversary might have due to periodic releases of models, which often incorporate statistics used for featurization – is *fundamentally non-private*[9;48;65;83].

1.2 The Sage System

As companies continue to disseminate many versions of models into untrusted domains, controlling the risk of data exposure becomes critical. We present *Sage*, an ML platform based on TFX that uses differential privacy (DP)[55] to bound the cumulative exposure of individual entries in a company's sensitive data streams through all the models released from those streams. At a high level, DP randomizes a computation over a dataset (e.g. training one model) to bound the leakage of individual entries in the dataset through the output of the computation (the model). Each new DP computation increases the bound over data leakage, and can be seen as consuming part of a *privacy budget* that should not be exceeded; Sage makes the process that generates all models and statistics preserve a *global DP guarantee*.

Sage builds upon the rich literature on DP ML *algorithms* (e.g.,[2;207;125], see §2.3.2) and contributes pragmatic solutions two of the most pressing *systems challenges* of global DP: (1) running out of privacy budget and (2) the privacy-utility tradeoff. Sage expects to be given training pipelines explicitly programmed to individually satisfy a parameterized DP guarantee. It acts as a new access control layer in TFX that: mediates all accesses to the data by these DP training pipelines; instantiates their DP parameters at runtime; and accounts for the cumulative privacy loss from all pipelines to enforce the global DP guarantee against the stream. At the same time,

Sage provides the developers with: control over the quality of the models produced by the DP training pipelines and mechanisms to minimize privacy budget use (thereby addressing challenge (2)); and the ability to release models endlessly without running out of privacy budget for the stream (thereby addressing challenge (1)).

The key to addressing both challenges is the realization that ML workloads operate on *growing databases*, a model of interaction that has been explored very little in DP, and only with a purely theoretical and far from practical approach[44]. Most DP literature, largely focused on *individual algorithms*, assumes either static databases (which do not incorporate new data) or online streaming (where computations do not revisit old data). In contrast, *ML workloads* – which consist of many algorithms invoked periodically – operate on *growing databases*. Across invocations of different training algorithms, the workload both incorporates new data and reuses old data, often times adaptively. It is in that *adaptive reuse of old data coupled with new data* that our design of Sage finds the opportunity to address the preceding two challenges in ways that are practical and integrate well with TFX-like platforms.

To address the running out of privacy budget challenge, we develop *block composition*, the first privacy accounting method that both allows efficient training on growing databases and avoids running out of privacy budget as long as the database grows fast enough. Block composition splits the data stream into *blocks*, for example by time (e.g., one day's worth of data goes into one block) or by users (e.g., each user's data goes into one block), depending on the unit of protection (event- or user-level privacy). Block composition lets training pipelines combine available blocks into larger datasets to train models effectively, but accounts for the privacy loss of releasing a model at the level of the specific blocks used to train that model. When the privacy loss for a given block reaches a pre-configured ceiling, the block is *retired* and will not be used again. However, new blocks from the stream arrive with zero privacy loss and can be used to train future models. Thus, as long as the database adds new blocks fast enough relative to the rate at which models arrive, Sage will never run out of privacy budget for the stream. Finally, block composition allows

adaptivity in the choice of training computation, privacy parameters, and blocks to execute on, thereby modeling the most comprehensive form of adaptivity in DP literature.

To address the privacy-utility tradeoff we develop *privacy-adaptive training*, a training procedure that controls the utility of DP-trained models by repeatedly and adaptively training them on growing data and/or DP parameters available from the stream. Models retrain until, with high probability, they meet programmer-specified quality criteria (e.g. an accuracy target). Privacy-adaptive training uses block composition's support for adaptivity and integrates well with TFX's design, which includes a model validation stage in training pipelines. Another implication of Sage's privacy guarantees is that data is retired when its privacy budget is exhausted. This can lead to less data being available, reducing utility. Sage proposes a mechanism that leverages training set minimzation mechanisms to build *protected data summaries*. These DP summarizes can be used and re-used at will to create new features that minimize the need for training data, thereby conserving the privacy budget of available data and retaining some utility from retired data.

1.3 Background

Our effort builds upon an opportunity we observe in today's companies: the rise of *ML platforms*, trusted infrastructures that provide key services for ML workloads in production, plus strong library support for their development. They can be thought of as *operating systems* for ML workloads. Google has TensorFlow-Extended (TFX)[13]; Facebook has FBLearner[79]; Uber has Michelangelo[116]; and Twitter has DeepBird[113]. The opportunity is to *incorporate DP into these platforms as a new type of access control that constrains data leakage through the models a company disseminates*.

1.3.1 ML Platforms

Fig. 11 shows our view of an ML platform; it is based on[13;79;116]. The platform has several components: *Training Pipelines* (one for each model pushed into production), *Serving Infrastructure*, and a shared data store, which we call the *Growing Database* because it accumulates data from the company's various streams. The access control policies on the Growing Database are exercised through Stream-level ACLs and are typically restrictive for sensitive streams.

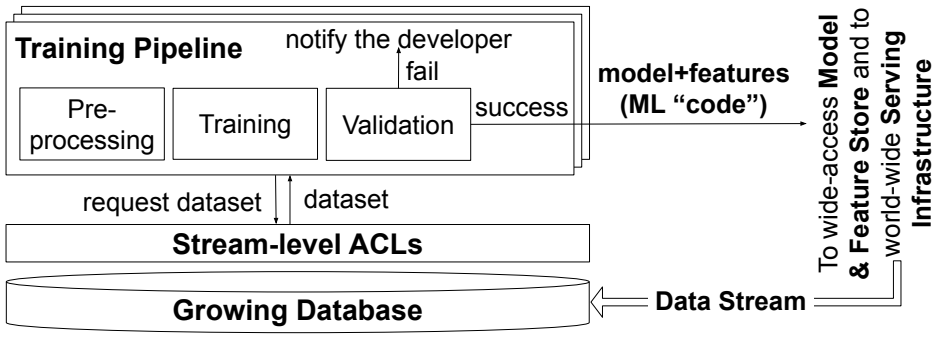


Fig. 11: **Typical Architecture of an ML Platform.**

The *Training Pipeline* trains a model on data from the Growing Database and verifies that it meets specific quality criteria before it is deployed for serving or shared with other teams. It is launched periodically (e.g., daily) on datasets containing samples from a representative time window (e.g., logs over the past month). It has three customizable modules: (1) *Pre-processing* loads the dataset from the Growing Database, transforms it into a format suitable for training and inference by applying feature transformation operators, and splits the transformed dataset into a training set and a testing set; (2) *Training* trains the model on a training set; and (3) *Validation* evaluates one or more *quality metrics* – such as accuracy for classification or mean squared error (MSE) for regression – on the testing set. It checks that the metrics reach specific *quality targets* to warrant the model's push into serving. The targets can be fixed by developers or can be values achieved by a previous model. If the model meets all quality criteria, it is bundled with its feature transformation operators (a.k.a. *features*) and pushed into the Serving Infrastructure. The model+features bundle is what we call *ML code*.

The *Serving Infrastructure* manages the online aspects of the model. It distributes the model+features to inference servers around the world and to end-user devices and continuously evaluates and partially updates it on new data. The model+features bundle is also often pushed into a company-wide *Model and Feature Store*, from where other teams within the company can discover it and integrate into their own models. Twitter and Uber report sharing embedding models[172] and tens of thousands of summary statistics[116] across teams through their Feature Stores. To enable such wide

sharing, companies sometimes enforce more permissive access control policies on the Model and Feature Store than on the raw data.

1.3.2 Threat Model

We are concerned with the increase in sensitive data exposure that is caused by applying looser access controls to data-carrying ML code –models+features– than are typically applied to the data. This includes placing models+features in company-wide Model and Feature Stores, where they can be accessed by developers not authorized to access the raw data. It includes pushing models+features, or their predictions, to end-user devices and prediction servers that could be compromised by hackers or oppressive governments. And it includes opening the models+features to the world through prediction APIs that can leak training data if queried sufficiently[186;173]. Our goal is to "neutralize" the wider exposure of ML codes by making the process of generating them DP across all models+features ever released from a sensitive stream.

We assume the following components are trusted and implemented correctly: Growing Database; Stream-level ACLs; the ML platform code running a Training Pipeline. We also *trust the developer* that instantiates the modules in each pipeline *as long as the developer is authorized* by Stream-level ACLs to access the data stream(s) used by the pipeline. However, we do not trust the wide-access Model and Feature Store or the locations to which the serving infrastructure disseminates the model+features or their predictions. Once a model/feature is pushed to those components, it is considered *released* to the untrusted domain and accessible to adversaries.

We focus on two classes of attacks against models and statistics (see Dwork[64]): (1) *membership inference*, in which the adversary infers whether a particular entry is in the training set based on either white-box or black-box access to the model, features, and/or predictions[9;65;83;173]; and (2) *reconstruction attacks*, in which the adversary infers unknown sensitive attributes about entries in the training set based on similar white-box or black-box access[28;48;64].

1.3.3 Differential Privacy

DP is concerned with whether the output of a computation over a dataset – such as training an ML model – can reveal information about individual entries in the dataset. To prevent such

information leakage, *randomness* is introduced into the computation to hide details of individual entries.

Definition 1 (Differential Privacy (DP)[61])**.** A randomized algorithm $\mathcal{Q} : \mathcal{D} \rightarrow \mathcal{V}$ is (ϵ, δ)-DP if for any $\mathcal{D}, \mathcal{D}'$ with $|D \oplus D'| \leq 1$ and for any $\mathcal{S} \subseteq \mathcal{V}$, we have: $P(\mathcal{Q}(\mathcal{D}) \in \mathcal{S}) \leq e^{\epsilon} P(\mathcal{Q}(\mathcal{D}') \in \mathcal{S}) + \delta$.

The $\epsilon > 0$ and $\delta \in [0, 1]$ parameters quantify the strength of the privacy guarantee: small values imply that one draw from such an algorithm's output gives little information about whether it ran on D or D'. The *privacy budget* ϵ upper bounds an (ϵ, δ)-DP computation's privacy loss with probability (1-δ). $\oplus$ is a dataset distance (e.g. the symmetric difference[130]). If $|D \oplus D'| \leq 1$, D and D' are *neighboring datasets*.

Multiple mechanisms exist to make a computation DP. They add noise to the computation scaled by its sensitivity s, the maximum change in the computation's output when run on any two neighboring datasets. Adding noise from a Laplace distribution with mean zero and scale $\frac{s}{\epsilon}$ (denoted $laplace(0, \frac{s}{\epsilon})$) gives $(\epsilon, 0)$-DP. Adding noise from a Gaussian distribution scaled by $\frac{s}{\epsilon}\sqrt{2\ln(\frac{1.25}{\delta})}$ gives (ϵ, δ)-DP.

DP is known to address the attacks in our threat model[173;64;28;90]. At a high level, membership and reconstruction attacks work by finding data points that make the observed model more likely: if those points were in the training set, the likelihood of the observed output increases. DP prevents these attacks, as no specific data point can drastically increase the likelihood of the model outputted by the training procedure.

DP literature is very rich and mature, including in ML. DP versions exist for almost every popular ML algorithm, including: stochastic gradient descent (SGD)[2;207]; various regressions[34;144;181;211;99]; collaborative filtering[128]; language models[127]; feature selection[36]; model selection[175]; evaluation[23]; and statistics, e.g. contingency tables[10], histograms[203]. The privacy module in TensorFlow v2 implements several SGD-based algorithms[125].

A key strength of DP is its *composition* property, which in its basic form, states that the process of running an (ϵ_1, δ_1)-DP and an (ϵ_2, δ_2)-DP computation on the same dataset is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DP. Composition enables the development of complex DP computations – such as DP Training

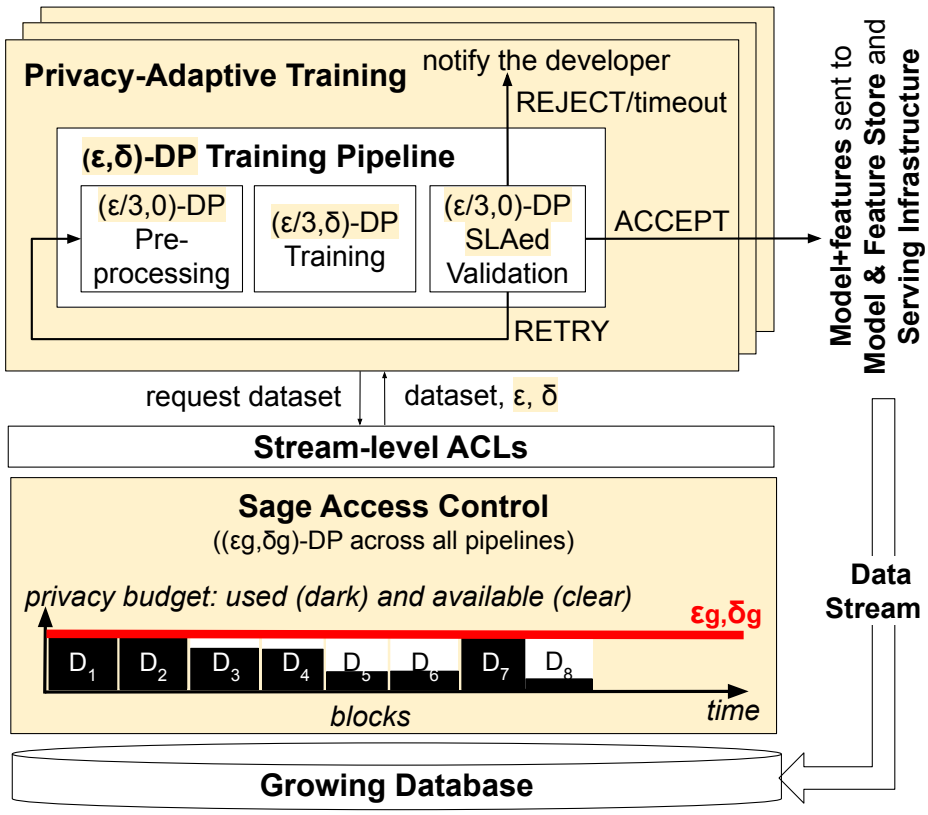


Fig. 12: **Sage DP ML Platform.** Highlights changes from non-DP version.

Pipelines – from piecemeal DP components, such as DP ML algorithms. Composition also lets one account for (and bound) the privacy loss resulting from a sequence of DP-computed outputs, such as the release of multiple models+features.

A distinction exists between *user-level* and *event-level* privacy. User-level privacy enforces DP on all data points contributed by a user toward a computation. Event-level privacy enforces DP on individual data points (e.g., individual clicks). User-level privacy is more meaningful than event-level privacy, but much more challenging to sustain on streams. Although Sage's design can in theory be applied to user-level privacy (§1.5.6), we focus most of the chapter on *event-level privacy*, which we deem practical enough to be deployed in big companies. §1.8 discusses the limitations of this semantic.

1.4 Design

The Sage training platform enforces a global (ϵ_g, δ_g)-DP semantic over all models+features that have been, or will ever be, released from each sensitive data stream. The highlighted portions in Fig. 12 show the changes Sage brings to a typical ML platform. First, each Training Pipeline must be made to individually satisfy (ϵ, δ)-DP for some privacy parameters given by Sage at runtime (box (ϵ, δ)-*DP Training Pipeline*, §1.4.1). The developer is responsible for making this switch to DP, and while research is needed to ease DP programming, this work leaves that challenge aside.

Second, Sage introduces an additional layer of access control beyond traditional stream-level ACLs (box *Sage Access Control*, §1.4.2). The new layer splits the data stream into *blocks* and accounts for the privacy loss of releasing a model+features bundle at the level of the specific blocks that were used to train that bundle. In theory, blocks can be defined by any insensitive attribute, with two attributes particularly relevant here: time (e.g., one day's worth of data goes into one block) and user ID (e.g., a user's data goes into the same block). Defining blocks by time provides event-level privacy; defining them by user ID accommodates user-level privacy. Because of our focus on the former, the remainder of this section assumes that blocks are defined by time; §1.5 discusses sharding by user ID and other attributes.

When the privacy loss for a block reaches the (ϵ_g, δ_g) ceiling, the block is *retired* (blocks D_1, D_2, D_7 are retired in Fig. 12). However, new blocks arrive with a clean budget and can be used to train future models: *as long as the database grows fast enough in new blocks*, the system will never run out of privacy budget for the stream. Perhaps surprisingly, this privacy loss accounting method, which we call *block composition*, is the first practical approach to avoid running out of privacy budget while enabling effective training of ML models on growing databases. §1.4.2 gives the intuition of block composition while §1.5 formalizes it and proves it (ϵ_g, δ_g)-DP.

Third, Sage provides developers with control over the quality of models produced by the DP Training Pipelines. Such pipelines can produce less accurate models that fail to meet their quality targets more often than without DP. They can also push in production low-quality models whose validations succeed by mere chance. Both situations lead to operational headaches: the former

```
def preprocessing_fn(inputs, epsilon):
  dist_01 = tft.scale_to_0_1(inputs["distance"],0,100)
  speed_01 = tft.scale_to_0_1(inputs["speed"],0,100)
  hour_of_day_speed = ~~group_by_mean~~
  sage.dp_group_by_mean(
    inputs["hour_of_day"], speed_01, 24, epsilon, 1.0)
  return {"dist_scaled": dist_01,
    "hour_of_day": inputs["hour_of_day"],
    "hour_of_day_speed": hour_of_day_speed,
    "duration": inputs["duration"]}

def trainer_fn(hparams, schema, epsilon, delta): [...]
  feature_columns = [numeric_column("dist_scaled"),
    numeric_column("hour_of_day_speed"),
    categorical_column("hour_of_day", num_buckets=24)]
  estimator = \
    ~~tf.estimator.DNNRegressor~~sage.DPDNNRegressor(
      config=run_config,
      feature_columns=feature_columns,
      dnn_hidden_units=hparams.hidden_units,
      privacy_budget=(epsilon, delta))
  return tfx.executors.TrainingSpec(estimator,...)

def validator_fn(epsilon):
  model_validator = \
    ~~tfx.components.ModelValidator~~sage.DPModelValidator(
      examples=examples_gen.outputs.output,
      model=trainer.outputs.output,
      metric_fn = _MSE_FN, target = _MSE_TARGET,
      epsilon=epsilon, confidence=0.95, B=1)
  return model_validator

def dp_group_by_mean(key_tensor, value_tensor, nkeys,
  epsilon, value_range):
  key_tensor = tf.dtypes.cast(key_tensor, tf.int64)
  ones = tf.fill(tf.shape(key_tensor), 1.0)
  dp_counts = group_by_sum(key_tensor, ones, nkeys)\
    + laplace(0.0, 2/epsilon, nkeys)
  dp_sums = group_by_sum(
    key_tensor,value_tensor,nkeys)\
    + laplace(0.0, value_range * 2/epsilon, nkeys)
  return tf.gather(dp_sums/dp_counts, key_tensor)
```

List. 1.1: **Example Training Pipeline.** Shows non-DP TFX (stricken through) and DP Sage (highlighted) versions. TFX API is simplified for exposition.

gives more frequent notifications of failed training, the latter gives dissatisfied users. The issue is often referred to as the *privacy-utility tradeoff* of running under a DP regime. Sage addresses this challenge by wrapping the (ϵ, δ)-DP Training Pipeline into an iterative process that reduces the effects of DP randomness on the quality of the models and the semantic of their validation by invoking training pipelines repeatedly on increasing amounts of data and/or privacy budgets (box *Privacy-Adaptive Training*, §1.4.3).

1.4.1 Example (ϵ, δ)-DP Training Pipeline

Sage expects each pipeline submitted by the ML developer to satisfy a parameterized (ϵ, δ)-DP. Acknowledging that DP programming abstractions warrant further research, L 1.1 illustrates the changes a developer would have to make at present to convert a non-DP training pipeline written for TFX to a DP training pipeline suitable for Sage. Removed/replaced code is stricken through and the added code is highlighted. The pipeline processes New York City Yellow Cab data[146] and trains a model to predict the duration of a ride.

To integrate with TFX (non-DP version), the developer implements three TFX callbacks. (1) `preprocessing_fn` uses the dataset to compute aggregate features and make user-specified feature transformations. The model has three features: the distance of the ride; the hour of the day; and an aggregate feature representing the average speed of cab rides each hour of the day. (2) `trainer_fn` specifies the model that is to be trained: it configures the columns to be modeled, defines hyperparameters, and specifies the dataset. The model trains with a neural network regressor. (3) `validator_fn` validates the model by comparing test set MSE to a constant.

To integrate with Sage (DP version), the developer: (a) switches library calls to DP versions of the functions (which ideally would be available in the ML platform) and (b) splits the (ϵ, δ) parameters, which are assigned by Sage at runtime, across the DP function calls. (1) `preprocessing_fn` replaces one call with a DP version that is implemented in Sage: the mean speed per day uses Sage's `dp_group_by_mean`. This function (lines 32-40) computes the number of times each key appears and the sum of the values associated with each key. It makes both DP by adding draws from appropriately-scaled Laplace distributions to each count. Each data point has exactly one key value so the privacy budget usage composes in parallel across keys[130]. The privacy budget is split across the sum and count queries. We envision common functions like this being available in the DP ML platform. (2) `trainer_fn` switches the call to the non-private regressor with the DP implementation, which in Sage is a simple wrapper around TensorFlow's DP SGD-based optimizer. (3) `validator_fn` invokes Sage's DP model validator (§1.4.3).

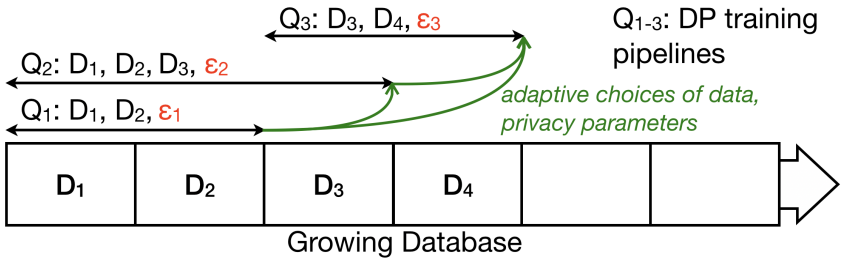


Fig. 13: **Characteristics of Data Interaction in ML.**

1.4.2 Sage Access Control

Sage uses the composition property of DP to rigorously account for (and bound) the cumulative leakage of data from sensitive user streams across multiple releases of models+features learned from these streams. Specifically, for each sensitive stream, Sage maintains a pre-specified event-level (ϵ_g, δ_g)-DP guarantee across all uses of the stream. Unfortunately, traditional DP composition theory considers either static databases, which leads to wasteful privacy accounting; or purely online streaming, which is inefficient for many ML workloads, including deep neural network training. We thus developed our own composition theory, called *block composition*, which leverages characteristics of ML workloads running on growing databases to permit both efficient privacy accounting and efficient learning. §1.5 formalizes the new theory. This section describes the limitations of existing DP composition for ML and gives the intuition for block composition and how Sage uses it as a new form of access control in ML platforms.

Data Interaction in ML on Growing Databases. Fig. 13 shows an example of a typical workload as seen by an ML platform. Each training pipeline, or *query* in DP parlance, is denoted Q_i. We note two characteristics. First, a typical ML workload consists of multiple training pipelines, training over time on a continuously growing database, and on data subsets of various sizes. For instance, Q_2 may train a large deep neural network requiring massive amounts of data to reach a good performance, while Q_3 may train a linear model with smaller data requirements, or even a simple statistic like the mean of a feature over the past day. All pipelines are typically updated or retrained periodically on new data, with old data eventually being deemed irrelevant and ignored.

Second, the data given to a training pipeline – and for a DP model its DP parameters – are typically chosen *adaptively*. For example, the model trained in Q_1 on data from $D_{1,2}$ with budget ϵ_1 may give unsatisfactory performance. After a new block D_3 is collected, a developer may decide to retrain the same model in query Q_2 on data from $D_{1,2,3}$, and with a higher DP budget ϵ_2. Adaptivity can also happen indirectly through the data. Suppose Q_2 successfully trained a recommendation model. Then, future data collected from the users (e.g. in D_4) may depend on the recommendations. Any subsequent query, such as Q_3, is potentially influenced by Q_2's output.

These characteristics imply three requirements for a composition theory suitable for ML. It must support:

R1 Queries on overlapping data subsets of diverse sizes.

R2 Adaptivity in the choice of: queries, DP parameters, and data subsets the queries process.

R3 Endless execution on a growing database.

Limitations of Existing Composition Theory for ML. No previous DP composition theory supports all three requirements. DP has mostly been studied for static databases, where (adaptively chosen) queries are assumed to compute over *the entire database*. Consequently, composition accounting is typically made at *query level*: each query consumes part of the total available privacy budget for the database. Query-level accounting has carried over even in extensions to DP theory that handle streaming databases[60] and partitioned queries[130]. There are multiple ways to apply query-level accounting to ML, but each trades off at least one of our requirements.

First, one can query overlapping data subsets (**R1**) adaptively across queries, the data used, and the DP parameters[163] (**R2**) by accounting for composition at the query level *against the entire stream*. Queries on Fig. 13 would thus result in a total privacy loss of $\epsilon_1 + \epsilon_2 + \epsilon_3$ over the whole stream. This approach wastes privacy budget and leads to the problem of "running out of budget". Once $\epsilon_g = \epsilon_1 + \epsilon_2 + \epsilon_3$, enforcing a global leakage bound of ϵ_g means that *one must stop using the stream* after query Q_3. This is true even though (1) not all queries run on all the

data and (2) there will be new data coming into the system in the future (e.g., D_5). This violates requirement (**R3**) of endless execution on streams.

Second, one can restructure the queries to enable finer granularity with query-level accounting. The data stream is partitioned in blocks, as in Fig. 13. Each query is split into multiple ones, each running with DP on an individual block. The DP results are then aggregated, for instance by averaging model updates as in federated learning[127]. Since each block is a separate dataset, traditional composition can account for privacy loss at the block level. This approach supports adaptivity (**R2**) and execution of the system on streams (**R3**) as new data blocks incur no privacy loss from past queries. However, it violates requirement (**R1**), resulting in unnecessarily noisy learning[50;51]. Consider computing a feature average. DP requires adding noise once, after summing all values on the combined blocks. But with independent queries over each block, we must add the same amount of noise to the sum over each block, yielding a more noisy total. Additionally, several DP training algorithms[2;127] fundamentally rely on sampling small training batches from large datasets to amplify privacy, which cannot be done without combining blocks.

Third, one can consume the data stream online using streaming DP. A new data point is allocated to one of the waiting queries, which consumes its entire privacy budget. Because each point is used by one query and discarded, DP holds over the entire stream. New data can be adaptively assigned to any query (**R2**) and arrives with a fresh budget (**R3**). However, queries cannot use past data or overlapping subsets, violating **R1** and rendering the approach impractical for large models.

Block Composition. Our new composition theory meets all three requirements. It splits the data stream into disjoint *blocks* (e.g., one day's worth of data for event-level privacy), forming a growing database on which queries can run on overlapping and adaptively chosen sets of blocks (**R1**, **R2**). This lets pipelines combine blocks with available privacy budgets to assemble large datasets. Despite running overlapping queries, we can still account for the privacy loss of each individual blocks, where each query impacts the blocks it actually uses, *not the entire data stream*. Unused blocks, including future ones, incur no privacy loss. In Fig. 13, the first three blocks each incur a privacy loss of $\epsilon_1 + \epsilon_2$ while the last block has $\epsilon_2 + \epsilon_3$. The privacy loss of these three

queries over the entire data stream will only be the maximum of these two values. Moreover, when the database grows (e.g. block D_5 arrives), the new blocks' privacy loss is zero. The system can thus run endlessly by training new models on new data (**R3**).

Sage Access Control. With block composition, Sage controls data leakage from a stream by enforcing DP on its blocks. The company configures a desirable (ϵ_g, δ_g) global policy for each sensitive stream. The Sage Access Control component tracks the available privacy budget for each data block. It allows access to a block until it runs out of budget, after which access to the block will never be granted again. When the Sage Iterator (described in §1.4.3) for a pipeline requests data, Sage Access Control only offers blocks with available privacy budget. The Iterator then determines the (ϵ, δ) privacy parameters it will use for its iteration and informs Sage Access Control, which deducts (ϵ, δ) from the available privacy budgets of those blocks. Finally, the Iterator invokes the developer-supplied DP Training Pipeline, trusting it to enforce the chosen (ϵ, δ) privacy parameters. §1.5 proves that this access control policy enforces (ϵ_g, δ_g)-DP for the stream.

The preceding operation is a DP-informed retention policy, but one can use block composition to define other access control policies. Suppose the company is willing to assume that its developers (or user devices and prediction servers in distinct geographies) will not collude to violate its customers' privacy. Then the company could enforce a separate (ϵ_g, δ_g) guarantee for each context (developer or geography) by maintaining separate lists of per-block available budgets.

1.4.3 Privacy-Adaptive Training

Sage's design adds reliability to the DP model training and validation processes, which are rendered imprecise by the DP randomness. We describe two novel techniques: (1) *SLAed validation*, which accounts for the effect of randomness in the validation process to give a high-probability guarantee of correctness for the validation (akin to a quality service level agreement, or SLA); and (2) *privacy-adaptive training*, which launches the (ϵ, δ)-DP Training Pipeline repeatedly on increasing amounts of data from the stream, and/or with increased privacy parameters, until the validation succeeds.

```
class DPLossValidator(sage.DPModelValidator):
  def validate(loss_fn, target, epsilon, confidence, B):
    if _ACCEPT_test(..., epsilon, (1-confidence)/2, B):
      return ACCEPT
    if _REJECT_test(..., epsilon, (1-confidence)/2, B):
      return REJECT
    return RETRY

  def _ACCEPT_test(test_labels, dp_test_predictions,
        loss_fn, target, epsilon, eta, B):
    n_test = dp_test_predictions.size()
    n_test_dp = n_test + laplace(2/epsilon)
    n_test_dp_min = n_test_dp - 2*log(3/(2*eta))/epsilon
    dp_test_loss = clip_by_value(loss_fn(test_labels,
      dp_test_predictions), 0, B) + laplace(2*B/epsilon)
    corrected_dp_test_loss = dp_test_loss +
      2*B*log(3/(2*eta))/epsilon
    return bernstein_upper_bound(
      corrected_dp_test_loss / n_test_dp_min,
      n_test_dp_min, eta/3, B) <= target

  def _REJECT_test(train_labels, dp_train_predictions,
        loss_fn, target, epsilon, eta, B):
    n_train = dp_train_predictions.size()
    n_train_dp = n_train + laplace(2/epsilon)
    n_train_dp_min = n_train_dp - 2*log(3/eta)/epsilon
    n_train_dp_max = n_train_dp + 2*log(3/eta)/epsilon
    dp_train_loss = clip_by_value(loss_fn(train_labels,
      dp_train_predictions), 0, B) + laplace(2*B/epsilon)
    corrected_dp_train_loss = dp_train_loss -
      2*B*log(3/(2*eta))/epsilon
    return hoeffding_lower_bound(
      corrected_dp_train_loss / n_train_dp_max,
      n_train_dp_min, eta/3, B) > target

  def bernstein_upper_bound(loss, n, eta, B):
    return loss+sqrt(2*B*loss*log(1/eta)/n)+4*log(1/eta)/n
  def hoeffding_lower_bound(loss, n, eta, B):
    return loss-B*sqrt(log(1/eta)/n)
```

List. 1.2: **Implementation of `sage.DPLossValidator` used in List. 1.1.**

SLAed DP Validation. Fig. 12 shows the three possible outcomes of SLAed validation: ACCEPT, REJECT, and RETRY. If SLAed validation returns ACCEPT, then with high probability (e.g. 95%) the model reaches its configured quality targets when predicting on new data from the same distribution. For certain metrics and under certain assumptions, it is also possible to give statistical guarantees of correct negative assessment. In such cases, if SLAed validation returns REJECT, then with high probability (e.g., 95%) this type of model *will never* reach the target, no matter how much data it is trained/validated on. Finally, if SLAed validation returns RETRY, it signals that more data is needed for an assessment.

We have implemented SLAed validators for three classes of metrics: loss metrics (e.g. MSE, log loss), accuracy metrics, and *absolute errors* of sum-based statistics such as mean and variance.

We give the intuition for these tests before detailing each test. All validators follow the same logic. First, we compute a DP version of the test quantity (e.g. MSE) on a testing set. Second, we compute the *worst-case impact of DP noise* on that quantity for a given confidence probability; we call this a *correction for DP impact*. For example, if we add Laplace noise with parameter $\frac{1}{\epsilon}$ to the sum of squared errors on n data points, assuming that the loss is in $[0, 1]$ we know that with probability $(1 - \eta)$ the sum is deflated by less than $-\frac{1}{\epsilon}\ln(\frac{1}{2\eta})$, because a draw from this Laplace distribution has just an η probability to be more negative than this value. Third, we use known statistical concentration inequalities, also made DP and corrected for worst case noise impact, to upper bound with high probability the loss on the entire distribution.

Loss SLAed Validator. A loss function is a measure of erroneous predictions on a dataset (so lower is better). Examples include: mean squared error for regression, log loss for classification, and minus log likelihood for Bayesian generative models. L 1.2 shows our loss validator. The validation function consists of two tests: ACCEPT and REJECT.

ACCEPT Test: Denote a loss function $l(f, x, y)$ with range $[0, B]$ measuring the quality of a prediction $f(x)$ with label y (lower is better), and a target loss τ_{loss} on the data distribution $\mathcal{D}$. Denote: the DP-trained model f^{dp}; the loss function range $[0, B]$; the target loss τ_{loss}. Given a the DP-trained model f^{dp}, we want to release f^{dp} only if $\mathcal{L}_{\mathcal{D}}(f^{\text{dp}}) \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}} l(f^{\text{dp}}, x, y) \leq \tau_{loss}$. The test works as follows. First, compute a DP estimate of the number of samples in the test set, corrected for the impact of DP noise to be a lower bound on the true value with probability $(1 - \frac{\eta}{3})$ (Lines 11-13 L 1.2):

$$\underline{n}_{\text{te}}^{\text{dp}} = n_{\text{te}} + \text{Laplace}(\frac{2}{\epsilon}) - \frac{2}{\epsilon}\ln(\frac{3}{2\eta}).$$

Then, compute a DP estimate of the loss corrected for DP impact (Lines 14-17 L 1.2) to be an upper bound on the true value, $\mathcal{L}_{te}(f^{\text{dp}}) \triangleq \frac{1}{n_{\text{te}}}\sum_{te} l(f^{\text{dp}}, x, y)$, with probability $(1 - \frac{\eta}{3})$:

$$\overline{\mathcal{L}}_{te}^{dp}(f^{\text{dp}}) = \frac{1}{\underline{n}_{\text{te}}^{\text{dp}}}\Big(\sum_{te} l(f^{\text{dp}}, x, y) + \text{Laplace}(\frac{2B}{\epsilon}) + \frac{2B}{\epsilon}\ln(\frac{3}{2\eta})\Big)$$

Lines 18-20, we see that Sage will ACCEPT when:

$$\overline{\mathcal{L}}_{te}^{dp}(f^{\text{dp}}) + \sqrt{\frac{2B\overline{\mathcal{L}}_{te}^{dp}(f^{\text{dp}})\ln(3/\eta)}{\underline{n}_{\text{te}}^{\text{dp}}}} + \frac{4B\ln(3/\eta)}{\underline{n}_{\text{te}}^{\text{dp}}} \leq \tau_{loss}.$$

This test gives the following guarantee:

Proposition 1 (Loss ACCEPT Test)**.** With probability at least $(1-\eta)$, the ACCEPT test returns true only if $\mathcal{L}_{\mathcal{D}}(f^{\mathrm{dp}}) \leq \tau_{loss}$.

Proof. The corrections for DP noise imply that $P(\underline{n}_{\mathrm{te}}^{\mathrm{dp}} > n_{\mathrm{te}}) \leq \frac{\eta}{3}$, and $P(\mathcal{L}_{te}^{dp}(f^{\mathrm{dp}}) > \mathcal{L}_{te}(f^{\mathrm{dp}})) \leq \frac{\eta}{3}$ (i.e. the lower bounds hold with probability at least (1-$\frac{\eta}{3}$)). Define $\mathrm{UB}^{dp} \triangleq \overline{\mathcal{L}}_{te}^{dp}(f^{\mathrm{dp}}) + \sqrt{\frac{2B\overline{\mathcal{L}}_{te}^{dp}(f^{\mathrm{dp}})\ln(3/\eta)}{\underline{n}_{\mathrm{te}}^{\mathrm{dp}}}} + \frac{4\ln(3/\eta)}{\underline{n}_{\mathrm{te}}^{\mathrm{dp}}}$, and $\mathrm{UB} \triangleq \mathcal{L}_{te}(f^{\mathrm{dp}}) + \sqrt{\frac{2B\mathcal{L}_{te}(f^{\mathrm{dp}})\ln(3/\eta)}{n_{\mathrm{te}}}} + \frac{4\ln(3/\eta)}{n_{\mathrm{te}}}$. Applying Bernstein's inequality[171] yields $P(\mathcal{L}_{\mathcal{D}}(f^{\mathrm{dp}}) > \mathrm{UB}) \leq \frac{\eta}{3}$. A union bound on those three inequalities gives that with probability at least $(1-\eta)$, $\mathcal{L}_{\mathcal{D}}(f^{\mathrm{dp}}) \leq \mathrm{UB} \leq \mathrm{UB}^{dp}$. The test ACCEPTs when $\mathrm{UB}^{dp} \leq \tau_{loss}$. □

This test uses Bernstein's concentration inequality to compute a upper bound for the loss over the entire distribution[171], which will give good bounds if the loss is small. If instead one expects the variance to be small, one can use empirical Bernstein bounds[123] as a drop-in replacement. Otherwise, one can fall back to Hoeffding's inequality[81].

REJECT Test: The ACCEPT test is already sufficient to ensure the quality of a model released by Sage's privacy-adaptive training. However, when the target is unachievable on the data distribution $\mathcal{D}$, these iterations waste valuable privacy resources. The REJECT test tries to detect unachievable targets so iteration can stop early. Assuming f^{dp} is obtained by a DP algorithm for approximately minimizing the training loss, the test detects when *no model* (DP or not) from the considered class $\mathcal{F}$ *can ever hope to achieve* τ_{loss} on $\mathcal{D}$. The reasoning behind the test uses the fact that the minimum training loss is no more than the training loss of any $f \in \mathcal{F}$; and this in particular holds for the $f^* \in \mathcal{F}$ with smallest expected loss over $\mathcal{D}$.

The REJECT test terminates a model when no model of the considered class $\mathcal{F}$ can hope to achieve the desired τ_{loss} performance. Noting the best possible model on the data distribution $f^\star \triangleq \arg\min_{f\in\mathcal{F}} \mathcal{L}_{\mathcal{D}}(f)$, we want to reject when $\mathcal{L}_{\mathcal{D}}(f^\star) > \tau_{loss}$. To do this, it considers the best model in $\mathcal{F}$ on the training set $\hat{f} \triangleq \arg\min_{f\in\mathcal{F}} \mathcal{L}_{tr}(f)$, and proceeds as follows. First, compute the DP upper and lower bounds for n_{tr}, holding together with probability at least $(1-\frac{\eta}{3})$ (Lines

24-27 L 1.2):

$$
\begin{aligned}
n_{\text{tr}}^{\text{dp}} &= n_{\text{tr}} + \text{Laplace}(\frac{2}{\epsilon}),\\
\underline{n}_{\text{tr}}^{\text{dp}} &= n_{\text{tr}}^{\text{dp}} - \frac{2}{\epsilon}\ln(\frac{3}{\eta}),\\
\overline{n}_{\text{tr}}^{\text{dp}} &= n_{\text{tr}}^{\text{dp}} + \frac{2}{\epsilon}\ln(\frac{3}{\eta}).
\end{aligned}
$$

Then, compute a DP estimate of the loss corrected for DP impact (Lines 14-17 L 1.2) to be a lower bound on the true value with probability $(1 - \frac{\eta}{3})$:

$$
\underline{\mathcal{L}}_{te}^{dp}(\hat{f}) = \frac{1}{\overline{n}_{\text{tr}}^{\text{dp}}}\Big(\sum_{tr} l(\hat{f}, x, y) + \text{Laplace}(\frac{2B}{\epsilon}) - \frac{2B}{\epsilon}\ln(\frac{3}{2\eta})\Big).
$$

Finally, Sage will REJECT when:

$$
\underline{\mathcal{L}}_{te}^{dp}(\hat{f}) - B\sqrt{\frac{\log(3/\eta)}{\underline{n}_{\text{tr}}^{\text{dp}}}} > \tau_{loss}.
$$

This test gives the following guarantee:

Proposition 2 (Loss REJECT Test)**.** With probability at least $(1-\eta)$, f^{dp} (or more accuratly $\mathcal{F}$) is rejected only if $\mathcal{L}_{\mathcal{D}}(f^\star) > \tau_{loss}$.

Proof. By definition, $\mathcal{L}_{tr}(\hat{f}) \leq \mathcal{L}_{tr}(f^\star)$. Applying Hoeffding's inequality[81] to $f^\star$ gives $P(\mathcal{L}_{\mathcal{D}}(f^\star) < \mathcal{L}_{tr}(f^\star) - B\sqrt{\frac{\log(3/\eta)}{n_{\text{tr}}}}) \leq \frac{\eta}{3}$. Similarly to the proof for Proposition (1), applying a union bounds over this inequality and the DP correction gives that with probability at least $(1-\eta)$, $\mathcal{L}_{\mathcal{D}}(f^\star) \geq \mathcal{L}_{tr}(f^\star) - B\sqrt{\frac{\log(3/\eta)}{n_{\text{tr}}}} \geq \mathcal{L}_{tr}^{dp}(\hat{f}) - B\sqrt{\frac{\log(3/\eta)}{\underline{n}_{\text{tr}}^{\text{dp}}}} > \tau_{loss}$, concluding the proof. □

Here we leverage Hoeffding's inequality[81] as we need to bound $\mathcal{L}_{\mathcal{D}}(f^\star)$: since we do not know $f^\star$, we cannot compute its variance or loss on the training set to use (empirical) Bernstein bounds.

We highlight that this result relies on $\hat{f}$, which may not always be available. In particular, is can be computed for convex problems, but not for DNNs for instance. When $\hat{f}$ is not available, Sage will not use the REJECT test.

DP Guarantee: We now need to prove that the ACCEPT and REJECT tests each are $(\epsilon, 0)$-DP. Since they each use a disjoint split of the dataset (training and testing) running both tests is $(\epsilon, 0)$-DP over the entire dataset.

Proposition 3 (DP ACCEPT)**.** ACCEPT is $(\epsilon, 0)$-DP.

Proof. The only data interaction for ACCEPT are to compute $\underline{n}_{\text{te}}^{\text{dp}}$ and $\overline{\mathcal{L}}_{te}^{dp}(f^{\text{dp}})$. The former is sensitivity 1, and is made $\frac{\epsilon}{2}$-DP with the Laplace mechanism. The latter uses data in the inner summer, which is sensitivity B and is made $\frac{\epsilon}{2}$-DP with the Laplace mechanism. Using basic composition, the test is $(\epsilon, 0)$-DP. □

The proof for REJECT is a bit more complicated as part of the procedure involves computing $\hat{f}$, which is not DP.

Proposition 4 (DP REJECT)**.** REJECT is $(\epsilon, 0)$-DP.

Proof. To apply the same argument as for Prop.3, we need to show that computing $\mathcal{L}_{tr}(\hat{f})$, which includes computing $\hat{f}$, has sensitivity B. Recall that $\hat{f}$ minimizes the training loss:

$$\hat{f}_{tr} = \arg\min_{f \in \mathcal{F}} \sum_{n_{\text{tr}}} l(f, x, y).$$

If we add a data point d, then $\hat{f}_{tr}$ has a loss at worst B on the new point. Because the best model with the new data point $\hat{f}_{tr+d}$ is a least as good as $\hat{f}_{tr}$ (otherwise it wouldn't be the $\arg\min$ model), $\mathcal{L}_{tr+d}(\hat{f}_{tr+d}) \leq \mathcal{L}_{tr}(\hat{f}_{tr}) + B$. Similarly, $\hat{f}_{tr+d}$ cannot be better than $\hat{f}_{tr}$ on the training set, so $\mathcal{L}_{tr+d}(\hat{f}_{tr+d}) \geq \mathcal{L}_{tr}(\hat{f}_{tr})$. Hence the sensitivty of computing $\mathcal{L}_{tr}(\hat{f})$ is at most B. □

Finding $\hat{f}$ is possible for some classes of problems (e.g. convex optimization), but not all (e.g. DNNs). As we see in the proof of Prop.4, getting the loss minimizer $\hat{f}$ is crucial to the DP property.

ACCEPT and REJECT are $(\epsilon, 0)$-DP. They use disjoint datasets, so `validate` is $(\epsilon, 0)$-DP.

Accuracy SLAed Validator. The accuracy metric applies to classification problems, where a model predicts one out of many classes. The target τ_{acc} is the proportion of correct predictions to reach on the data distribution. The accuracy validator follows the same logic as the loss validator,

with two small changes. First, the upper and lower bounds are reversed as losses are minimized while accuracy is maximized. Second, the result of classification predictions follows a binomial distribution, which yields tighter confidence intervals. Note $\mathbb{1}\{\text{predicate}\}$ the indicator function with value 1 if the predicate is true, and 0 otherwise, and $\overline{\text{Bin}}(k, n, \eta)$ and $\underline{\text{Bin}}(k, n, \eta)$ the upper and lower bounds on the probability parameter p of a binomial distribution such that k happen can happen with probability at least η out of n independent draws (both can be conservatively approximated using a Clopper-Pearson interval).

We compute $k_{te}^{dp} = \sum_{n_{\text{te}}} \mathbb{1}\{f(x) = y\} + Laplace(\frac{2}{\epsilon})$ and $n_{\text{te}}^{\text{dp}} = n_{\text{te}} + Laplace(\frac{2}{\epsilon})$. Similarly for the training set tr, $k_{tr}^{dp} = \sum_{n_{\text{te}}} \mathbb{1}\{\hat{f}(x) = y\} + Laplace(\frac{2}{\epsilon})$ and $n_{\text{tr}}^{\text{dp}} = n_{\text{tr}} + Laplace(\frac{2}{\epsilon})$. Sage will ACCEPT when:

$$\underline{\text{Bin}}\Big(k_{te}^{dp} - \frac{2}{\epsilon}\ln(\frac{3}{\eta}), n_{\text{te}}^{\text{dp}} + \frac{2}{\epsilon}\ln(\frac{3}{\eta}), \frac{\eta}{3}\Big) \geq \tau_{acc}.$$

And REJECT when:

$$\overline{\text{Bin}}\Big(k_{tr}^{dp} + \frac{2}{\epsilon}\ln(\frac{3}{\eta}), n_{\text{tr}}^{\text{dp}} - \frac{2}{\epsilon}\ln(\frac{3}{\eta}), \frac{\eta}{3}\Big) < \tau_{acc}.$$

Using the same reasoning as before, we can prove the equivalent four Propositions (1, 2, 3, 4) for the accuracy validator. Finding $\hat{f}$ for accuracy is computationally hard.

Sum-based Statistics SLAed Validator. This validator applies to computing sum based statistics (e.g. mean, variance). The target is defined as the maximum size of the absolute (additive) error τ_{err} for these summary statistics on the data distribution. This error can be computed directly on the training set, so there is no need to a test set. Second, because of the law of large numbers, we can always reach the target precision, so there is no rejection test. We next show the test using Hoeffding's inequality[81] (if we expect the variance to be small, empirical Bernstein bounds[123] will be better and are directly applicable).

Compute:

$$n_{\text{tr}}^{\text{dp}} = n_{\text{tr}} + Laplace(\frac{2}{\epsilon}) - \frac{2}{\epsilon}\ln(\frac{2}{\eta})$$

SageACCEPTs if:

$$\frac{1}{n_{\text{tr}}^{\text{dp}}}\frac{2}{\epsilon}\ln(\frac{2}{\eta}) + B\sqrt{\frac{\ln(2/\eta)}{n_{\text{tr}}^{\text{dp}}}} \leq \tau_{err},$$

in which case the absolute error is bellow τ_{err} with probability at least $(1-\eta)$, accounting for the statistical error, as well as the impact of DP noise on both the sum based summary statistic and the ACCEPT test. Once again, the same reasoning allows us to prove equivalent Propositions to 1 and 3.

Privacy-Adaptive Training. Sage attempts to improve the quality of the model and its validation by supplying them with more data or privacy budgets so the SLAed validator can either ACCEPT or REJECT the model. Several ways exist to improve a DP model's quality. First, we can increase the dataset's size: at least in theory, it has been proven that one can compensate for the loss in accuracy due to *any* (ϵ, δ)-DP guarantee by increasing the training set size[96]. Second, we can increase the privacy budget (ϵ, δ) to decrease the noise added to the computation: this must be done within the available budgets of the blocks involved in the training and *not too aggressively*, because wasting privacy budget on one pipeline can prevent other pipelines from using those blocks.

Privacy-adaptive training searches for a configuration that can be either ACCEPTed or REJECTed by the SLAed validator. We have investigated multiple strategies for how to do this search. Those that conserve privacy budget have proven the most efficient. We start out with a small privacy budget (ϵ_0, δ_0) and with developer-configured start time and minimum window size for the model's training. On RETRY from the validator, we make sure to double either the privacy budget or the number of samples available to the Training Pipeline by accepting new data from the stream. Evaluation §1.6.4 shows that compared to alternatives, this iteration strategy conserves privacy budgets and improves performance when multiple Training Pipelines contend for the same blocks.

1.4.4 Preserving Privacy Budget and Mitigating Block Retirement

With Sage's access control (§1.4.2) a data block is retired when its privacy budget is exhausted, and the blocks' data can never be queried again. To conserve privacy budget and retain some utility from retired blocks, Sage proposes an optional mechanism that leverages a training set minimization mechanism called count based featurization, together with differential privacy, to build *protected data summaries*. These summaries are used to enhance the training data with new

features, which reduces the volume of training data required to reach performance targets. This reduces the amount of data queried by a training pipeline, conserving DP budget. Because these protected data summaries are DP, they can remain in use even after a block is retired, retaining some utility from old data.

We next detail how Sage leverages *count featurization*, a known ML mechanism, to count featurize observations before feeding them to models for training and prediction. We then describe our *DP count tables* that enable DP count featurization with minimal impact on utility.

Count Featurization

Count-based featurization[177] is a popular approach to handling categorical variables of high cardinality. Rather than directly using the value of a categorical variable, this technique featurizes the data with the number of times a particular feature value (e.g., a user ID) was observed with each label and the conditional probability of the label given the feature value. This substantially reduces dimensionality: standard encoding of categorical variables[5] results in a feature space of dimension $O(dK)$, whereas with count featurization it is $O(dL)$. Count featurization can also be applied to continuous variables or continuous labels by first discretizing them; this increases dimensionality but only by a small factor. The dramatic dimensionality reduction yields important benefits. It is known that fewer dimensions permit more efficient learning, both statistically and computationally, potentially at the cost of reducing predictive accuracy. However, count featurization makes it feasible to apply advanced, nonlinear models, such as neural networks, boosted trees, and random forests. This combination of succinct data representation and powerful learning models enables substantial reduction of the training data with little loss in predictive performance. Quantified in §1.6.5, this is the insight behind our use of count-based featurization to limit data exposure.

Concretely, an observation is a pair $\langle \vec{x}, l \rangle$ with a feature vector $\vec{x} = \langle x_1, x_2, ..., x_d \rangle$ and a label l. When using protected data summaries, a new observation is incorporated into a *historical statistics store*, a data structures which consists of multiple *count tables* that maintain the number of occurrences of each feature with each label. Sage maintains count tables for all features in $\vec{x}$ and

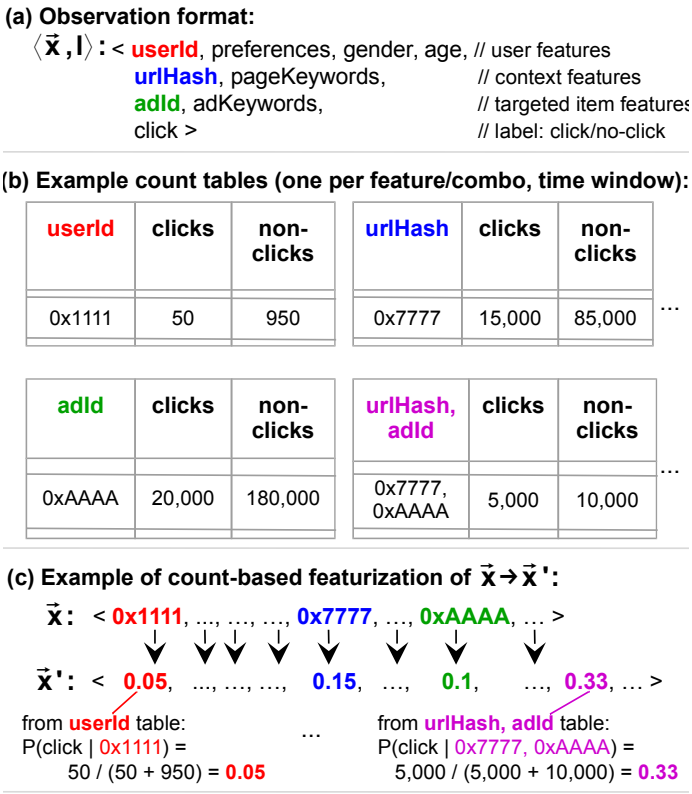


Fig. 14: **Count featurization example.**

for some feature combinations. A separate set of count tables is maintained for each data block (it will consume a pre-determined part of its privacy budget).

For learning and prediction, featurization transforms a feature vector $\vec{x}$ into a count-featurized feature vector $\vec{x'}$, by replacing each feature x_i with the conditional probabilities of each label value given x_i's value. The conditional probabilities are computed directly from the count tables as discussed below. To leverage protected data summaries, an ML pipeline featurizes its training and validation et using the historical counts. **Example.** Fig. 14 shows (a) a sample observation format, (b) some count tables used by Sage to count-featurize it, and (c) a sample count-featurized observation.

• *Observation format.* In targeting and personalization, an observation's feature vector $\vec{x}$ typically consists of *user features* (e.g., id, gender, age, and previously compiled preferences) and *contextual information* for the observation (e.g., the URL of the article or the ad shown to the user, plus any features of these). The label l might indicate whether the user clicked on the article/ad.

• *Count tables.* If a data stream of the preceding observation format is registered with Sage, the *userId* table maintains for each user the number of clicks the user has made on any ad shown and the number of non-clicks; it therefore encodes each user's propensity to click on ads. The *urlHash* table maintains for each URL the number of clicks that each user made on any ad shown on that page; it therefore encodes the page's inherent "ad-clickability." Sage maintains count tables for every feature in $\vec{x}$ and for some feature combinations with predictive potential, such as the $\langle urlHash, adId \rangle$ table, which encodes the joint probability of a particular ad being clicked when it is shown on a particular page.

• *Count featurization.* To count-featurize a feature vector $\vec{x} = \langle x_1, x_2, \ldots, x_d \rangle$, Sage first replaces each of its features with the conditional probabilities computed from the count tables, e.g., $\vec{x'} = \langle P(click|x_1), P(click|x_2), \ldots, P(click|x_d) \rangle$, where $P(click|x_i) = \frac{clicks}{clicks+non\text{-}clicks}$ from the row matching the value of x_i in the table corresponding to x_i. Sage also appends to $\vec{x'}$ the conditional probabilities for any feature combinations it maintains. Fig. 14(c) shows an example of feature vector $\vec{x}$ and its count-featurized version $\vec{x'}$. This is a simplified version of the count featurization function. We can also include the raw counts in $\vec{x'}$, and support non-binary categorical labels by including conditional probabilities for each label. To avoid featurizing with an effectively random probability when a given feature value has very few counts, we estimate the variance of our probability estimate and, if it is too high, featurize with a default probability $P(click)$.

Process. Sage count-featurizes all features x_i for each observation type. For categorical features, we featurize them as described above. Continuous features are first mapped to a discrete space, binning them by percentiles, and then count-featurized as categorical. We do the same with continuous labels. For low-cardinality and continuous features, we can additionally include the raw feature values in $\vec{x'}$ alongside the conditional probabilities.

Sage maintains count tables as follows. There is one count table per feature or feature group; it has a column for each label and a row for each value the feature can take. To support granular retention times, each count table is composed of count tables holding data for each data block, each such count table using a user defined part of the block's privacy budget (it is treated as a

query and fits directly in Sage's block composition approach). The complete count table is the sum of the associated windowed count tables. When a new observation arrives, it is added to the current block's count table; this count table is withheld when computing the complete count table until it is finished populating. At this point, Sage begins using it as part of the featurization process, phases out the oldest count table if it is past its retention period, and begins populating a new count table that has been initialized with differentially private noise. Once count tables are incorporated into the featurization process, they are never updated again.

Count-min sketches (CMSes). A key challenge with count featurization is its storage requirement. For a categorical variable of cardinality K and a label of cardinality L, the count table is of size $O(LK)$. A common solution, used in Azure[8], is to store each table in a Count-Min Sketch (CMS)[42], a data structure that approximates counts in sub-linear space. A CMS consists of a 2D array with an independent hash function for each row. When a new feature arrives, the CMS uses the hash function for each row to assign the feature to a column and increment the value in that cell.

We query the CMS for a feature count by hashing the feature into a column of each row and taking the minimum value. Despite overcounting from collisions, CMS provides sufficiently accurate count estimates to train ML models. With a CMS, we can maintain more and/or larger count tables with bounded storage overheads. This gives developers flexibility in the types of modeling they can do atop in-use data without tapping into the historical data store. The CMS poses challenges to our noise infusion process, as described next.

DP Count Tables

As all models and summary statistics in Sage, count tables need to be DP. They can then be used indefinitely without additional privacy loss. Upon creating a count table, we initialize each cell of the CMS storing that table with a random draw from a Laplace distribution. This noise is added only once: the count tables are updated as observations arrive and are sealed when the hot window rolls over. To determine the correct parameter for the Laplace distribution, b, we must account for three factors: (1) the internal structure of the CMS, (2) the number of observations we

want to hide simultaneously, and (3) the number of count tables (features or feature combinations) we are maintaining.

First, an exact count table has sensitivity 1 since adding or removing an observation can only change one count by 1. For a CMS, each observation is counted once per hash function; hence, the sensitivity is h, the number of hash functions. For a privacy budget of ϵ, then we make a count table ϵ-differentially private by adding noise from a Laplace distribution of parameter $b = \frac{h}{\epsilon}$ in every cell of the CMS. Second, we must maintain multiple count tables for the different features and feature groups. Since each observation affects every count table, we need to split the privacy budget ϵ among them, e.g., splitting it evenly by adding noise with $b = \frac{nh}{\epsilon}$ to each table.

This second consideration poses a significant challenge for Sage: the amount of noise we apply grows linearly with the number of count tables we keep. Since the amount of noise directly affects application accuracy, this yields a protection/accuracy tradeoff, which we address with weighted noise infusion.

Weighted noise infusion process. We note that count tables are not all equally susceptible to noise. For example in our movie recommender, the $user$ table most likely contains low values, since each user rates only a few movies (29 for the median user). Moreover, we do not expect this count to change significantly when adding more data, since single users will not rate significantly more movies. Each $genre$ table however contains higher values (1M or more), since each genre characterizes multiple movies, each rated by many users. Sharing noise equally between tables would pollute all counts by a standard deviation of 145 ($\epsilon = 1$, $h = 5$, and $k = 1$), a reasonable amount for $genre$s, but devastating for the $user$ feature, which essentially becomes random.

Sage's weighted noise infusion distributes the privacy budget unevenly across count tables, adding less noise to low-count features. This way, we retain more utility from those tables, and the composability property of differential privacy preserves our protection guarantees. Each table's share of noise is determined automatically, based on the count values observed in the hot window. Specifically, the user specifies a quantile, and the privacy budget is shared between each feature proportionally to this quantile of its counts. For instance we use the first percentile, so that 99% of

the counts for a feature will be less affected by the noise. Sharing the privacy budget proportionally to the counts is a heuristic that makes the noise's standard-deviation proportional to the typical counts of each feature. This scheme is also independent of the learning algorithm.

Section 1.6.5 shows that weighted noise infusion is vital for providing protection while preserving accuracy at scale: without it, the cost of hiding single observations is a 15% accuracy loss; with it, the loss is less than 5%.

The weight selection process must be made differentially private lest it may leak information about the hot window used to compute the weights, which we do as follows. We compute the weights every so often (e.g., every month) using a query to data blocks. We use a configurable portion of the blocks' privacy budget and leverage smooth sensitivity[145] to compute differentially private count percentiles, which we then use as feature weights. We use smooth sensitivity because the quantile function has a poor global sensitivity, but on most datasets, including those we tried, has a very small local sensitivity. We compute differentially private percentiles by adapting the J-List algorithm for the differentially private median described in[145] modified in a straight forward way for arbitrary quantiles. The maximum value of each count is the size of the window, and we use a Cauchy distribution to preserve (ϵ)-differential privacy.

Unbiased private count-median sketch. Another factor that degrades performance when adding differentially private noise is the interaction between the noise and the CMS. In the CMS, the final estimate for a count is $\min(h_i(\textit{key}))$ for each row i. The minimum makes sense here since collisions can only increase the counts. The Laplace distribution however is symmetric around zero, so we may add negative noise to the counts. Taking the minimum of multiple draws—each cell is initiated with a random draw from the distribution—thus selects the most extreme negative values, creating a downward bias that can be very large for a small ϵ.

We observe that because the mean of the Laplace distribution is 0, an unbiased estimator would not suffer from this drawback. For tables with large noise, we thus use a count-median sketch[32], which differs in two ways: 1) each row i has another hash function s_i that maps the key to a random sign $s_i(\textit{key}) \in \{+1, -1\}$, with each cell updated with $s_i(\textit{key})h_i(\textit{key})$; 2) the estimator

is the median of all counts multiplied by their sign, instead of the minimum. The signed update means that collisions have an expected impact of zero, since they have an equal chance of being negative or positive, making the cell an unbiased estimate of the true count. The median is a robust estimate that preserves the unbiased property.

Using this count-median sketch reduces the impact of noise, since values from the Laplace distribution are exponentially concentrated around the mean of zero. §1.6.5 shows that for small ϵ, or a large number of features, it is worth trading the CMS's better guarantees for reduced noise impact with the count-median sketch.

1.5 Theory

This section provides the theoretical backing for block composition, which we invent for Sage but which we believe has broader applications (§1.5.6). To analyze composition, one formalizes permissible interactions with the sensitive data in a *protocol* that facilitates the proof of the DP guarantee. This interaction protocol makes explicit the worst-case decisions that can be made by modeling them through an adversary. In the standard protocol (detailed shortly), an adversary $\mathcal{A}$ picks the neighboring data sets and supplies the DP queries that will compute over one of these data sets; the choice between the two data sets is exogenous to the interaction. To prove that the interaction satisfies DP, one must show that given the results of the protocol, it is impossible to determine with high confidence which of the two neighboring data sets was used.

Fig. 15 describes three different interaction protocols of increasing sophistication. Alg. (15a) is the basic DP composition protocol. Alg. (15b) is a block-level protocol we propose for static databases. Alg. (15c) is the protocol adopted in Sage; it extends Alg. (15b) by allowing a streaming database and adaptive choices of blocks and privacy parameters. Highlighted are changes made to the preceding protocol.

1.5.1 Traditional Query-Level Accounting

QueryCompose (Alg. (15a)) is the interaction protocol assumed in most analyses of composition of several DP interactions with a database. There are three important characteristics. First, the number of queries r and the DP parameters $(\epsilon_i, \delta_i)_{i=1}^r$ are fixed in advance. However the DP queries

(a) QueryCompose($\mathcal{A}$, b, r, $(\epsilon_i, \delta_i)_{i=1}^r$):

for i in $1, \ldots, r$ **do** ▷ ($\mathcal{A}$ depends on $V_1^b, \ldots, V_{i-1}^b$ in iter. i)
  $\mathcal{A}$ gives neighboring datasets $\mathcal{D}^{i,0}$ & $\mathcal{D}^{i,1}$
  $\mathcal{A}$ gives (ϵ_i, δ_i)-DP $\mathcal{Q}_i$
  $\mathcal{A}$ receives $V_i^b = \mathcal{Q}_i(\mathcal{D}^{i,b})$
return $V^b = (V_1^b, \ldots, V_r^b)$

(a) **Traditional Query-level Accounting.**

(b) BlockCompose($\mathcal{A}$, b, r, $(\epsilon_i, \delta_i)_{i=1}^r$, $(\text{blocks}_i)_{i=1}^r$):

$\mathcal{A}$ gives two neighboring block datasets $\mathcal{D}^0$ and $\mathcal{D}^1$
for i in $1, \ldots, r$ **do** ▷ ($\mathcal{A}$ depends on $V_1^b, \ldots, V_{i-1}^b$ in iter. i)
  $\mathcal{A}$ gives (ϵ_i, δ_i)-DP $\mathcal{Q}_i$
  $\mathcal{A}$ receives $V_i^b = \mathcal{Q}_i(\bigcup_{j \in \text{blocks}_i} \mathcal{D}_j^b)$
return $V^b = (V_1^b, \ldots, V_r^b)$

(b) **Block Composition for Static Datasets.** Change from query-level accounting shown in yellow background.

(c) AdaptiveStreamBlockCompose($\mathcal{A}$, b, r, ϵ_g, δ_g, $\mathcal{W}$):

1: $\mathcal{A}$ gives k, the index of the block with the adversarially chosen change
2: **for** i in $1, \ldots, r$ **do** ▷ ($\mathcal{A}$ depends on $V_1^b, \ldots, V_{i-1}^b$ in iter. i)
3: **if** create new block l and $l == k$ **then**
4: $\mathcal{A}$ gives neighboring blocks $\mathcal{D}_k^0$ and $\mathcal{D}_k^1$
5: **else if** create new block l and $l \neq k$ **then**
6: $\mathcal{D}_l^b = \mathcal{D}(\mathcal{W}, V_1^b, \ldots, V_{i-1}^b)$
7: $\mathcal{A}$ gives blocks_i, (ϵ_i, δ_i), and (ϵ_i, δ_i)-DP $\mathcal{Q}_i$
8: **if** $\bigwedge_{j \in \text{blocks}_i} \text{AccessControl}^j_{\epsilon_g, \delta_g}(\epsilon_1^j, \delta_1^j, ..., \epsilon_i^j, \delta_i^j, 0, ...)$ **then**
9: $\mathcal{A}$ receives $V_i^b = \mathcal{Q}_i(\bigcup_{j \in \text{blocks}_i} \mathcal{D}_j^b)$
10: **else** $\mathcal{A}$ receives no-op $V_i^b = \perp$
return $V^b = (V_1^b, \ldots, V_r^b)$

(c) **Sage Block Composition.** Adds support for streams (yellow lines 1-6) and adaptive choice of blocks, privacy parameters (green lines 7-8).

Fig. 15: **Interaction Protocols for Composition Analysis.** $\mathcal{A}$ is an algorithm defining the adversary's power; $b \in \{0, 1\}$ denotes two hypotheses the adversary aims to distinguish; r is the number of rounds; $(\epsilon_i, \delta_i)_{i=1}^r$ the DP parameters used at each round; $(\text{blocks}_i)_{i=1}^r$ the blocks used at each round. $\text{AccessControl}^j_{\epsilon_g, \delta_g}$ returns true if running (ϵ_i, δ_i)-DP query $\mathcal{Q}_i$ on block j ensures that with probability $\geq (1 - \delta_g)$ the privacy loss for block j is $\leq \epsilon_g$.

$\mathcal{Q}_i$ can be chosen adaptively. Second, the adversary adaptively chooses neighboring datasets $\mathcal{D}^{i,0}$ and $\mathcal{D}^{i,1}$ for each query. This flexibility lets the protocol readily support adaptively evolving data (such as with data streams) where future data collected may be impacted by the adversary's change to past data. Third, the adversary receives the results V^b of running the DP queries $\mathcal{Q}_i$ on $\mathcal{D}^{i,b}$;

here, $b \in \{0, 1\}$ is the exogenous choice of which database to use and is unknown to $\mathcal{A}$. DP is guaranteed if $\mathcal{A}$ cannot confidently learn b given V^b.

A common tool to analyze DP protocols is *privacy loss*:

Definition 2 (Privacy Loss)**.** Fix any outcome $v = (v_1, \ldots, v_r)$ and denote $v_{<i} = (v_1, \ldots, v_{i-1})$. The *privacy loss* of an algorithm Compose($\mathcal{A}$, b, r, $\cdot$) is:

$$\text{Loss}(v) = \ln\Big(\frac{P(V^0 = v)}{P(V^1 = v)}\Big) = \ln\Big(\prod_{i=1}^{r} \frac{P(V_i^0 = v_i | v_{<i})}{P(V_i^1 = v_i | v_{<i})}\Big)$$

Bounding the privacy loss for any adversary $\mathcal{A}$ with high probability implies DP[97]. Suppose that for any $\mathcal{A}$, with probability $\geq (1 - \delta)$ over draws from $v \sim V^0$, we have: $|\,\text{Loss}(v)| \leq \epsilon$. Then Compose($\mathcal{A}$, b, r, $\cdot$) is (ϵ, δ)-DP. This way, privacy loss and DP are defined in terms of distinguishing between two hypotheses indexed by $b \in \{0, 1\}$.

Previous composition theorems (e.g. basic composition[58], strong composition[63], and variations thereof[95]) analyze Alg. (15a) to derive various arithmetics for computing the overall DP semantic of interactions adhering to that protocol. In particular, the basic composition theorem[58] proves that QueryCompose($\mathcal{A}$, b, r, $(\epsilon_i, \delta_i)_{i=1}^r$) is $(\sum_{i=1}^r \epsilon_i, \sum_{i=1}^r \delta_i)$-DP. These theorems form the basis of most ML DP work. However, because composition is accounted for at the query level, imposing a fixed global privacy budget means that one will "run out" of it and stop training models even on new data.

1.5.2 Block Composition for Static Datasets

Block composition improves privacy accounting for workloads where interaction consists of queries that run on *overlapping data subsets of diverse sizes*. This is one of the characteristics we posit for ML workloads (requirement **R1** in §1.4.2). Alg. (15b), *BlockCompose*, formalizes this type of interaction for a *static dataset setting* as a springboard to formalizing the full ML interaction. We make two changes to QueryCompose. First (line 1), the neighboring datasets are defined once and for all before interacting. This way, training pipelines accessing non-overlapping parts of the dataset cannot all be impacted by one entry's change. Second (line 4), the data is split in blocks, and each DP query runs on a subset of the blocks.

We prove that the privacy loss over the entire dataset is the same as the maximum privacy loss on each block, accounting only for queries using this block:

Theorem 1 (Reduction to Block-level Composition)**.** The privacy loss of BlockCompose($\mathcal{A}$, b, r, $(\epsilon_i, \delta_i)_{i=1}^r$, $(\text{blocks}_i)_{i=1}^r$) is upper-bounded by the maximum privacy loss for any block:

$$|\operatorname{Loss}(v)| \leq \max_k \Big|\ln\Big(\prod_{\substack{i=1\\ k\in\text{blocks}_i}}^{r} \frac{P(V_i^0 = v_i|v_{<i})}{P(V_i^1 = v_i|v_{<i})}\Big)\Big|.$$

Proof. Let $\mathcal{D}^0$ and $\mathcal{D}^1$ be the neighboring datasets picked by adversary $\mathcal{A}$, and let k be the block index s.t. $\mathcal{D}_l^0 = \mathcal{D}_l^1$ for all $l \neq k$, and $|\mathcal{D}_k^0 \oplus \mathcal{D}_k^1| \leq 1$. For any result v of Alg. (15b):

$$\begin{aligned}
|\operatorname{Loss}(v)| &= \Big|\ln\Big(\prod_{i=1}^{r} \frac{P(V_i^0 = v_i|v_{<i})}{P(V_i^1 = v_i|v_{<i})}\Big)\Big| \\
&= \Big|\ln\Big(\prod_{\substack{i=1\\ k\in\text{blocks}_i}}^{r} \frac{P(V_i^0 = v_i|v_{<i})}{P(V_i^1 = v_i|v_{<i})}\Big) + \cancel{\ln\Big(\prod_{\substack{i=1\\ k\notin\text{blocks}_i}}^{r} \frac{P(V_i^0 = v_i|v_{<i})}{P(V_i^1 = v_i|v_{<i})}\Big)}\Big| \\
&\leq \max_k \Big|\ln\Big(\prod_{\substack{i=1\\ k\in\text{blocks}_i}}^{r} \frac{P(V_i^0 = v_i|v_{<i})}{P(V_i^1 = v_i|v_{<i})}\Big)\Big|
\end{aligned}$$

The slashed term is zero because if $k \notin \text{blocks}_i$, then $\bigcup_{j\in\text{blocks}_i} \mathcal{D}_j^0 = \bigcup_{j\in\text{blocks}_i} \mathcal{D}_j^1$, hence $\frac{P(V_i^0=v_i|v_{<i})}{P(V_i^1=v_i|v_{<i})} = 1$. □

Hence, unused data blocks allow training of other (adaptively chosen) ML models, and exhausting the DP budget of a block means we retire that block of data, and not the entire data set. This result, which can be extended to strong composition (see tech. report[112]), can be used to do tighter accounting than query-level accounting when the workload consists of queries on overlapping sets of data blocks (requirement **R1**). However, it does not support adaptivity in block choice or a streaming setting, violating **R2** and **R3**.

1.5.3 Sage Block Composition

Alg. (15c), *AdaptiveStreamBlockCompose*, addresses the preceding limitations with two changes. First, supporting streams requires that datasets not be fixed before interacting, because future data depends on prior models trained and pushed into production. The highlighted portions of lines 1-10 in Alg. (15c) formalize the dynamic nature of data collection by having new data blocks explicitly depend on previously trained models, which are chosen by the adversary, in addition to

other mechanisms of the world $\mathcal{W}$ that are not impacted by the adversary. Fortunately, Theorem 1 still applies, because model training can only use blocks that existed at the time of training, which in turn only depend on prior blocks through DP trained models. Therefore, new data blocks can train new ML models, enabling endless operation on streams (**R3**).

Second, interestingly, supporting adaptive choices in the data blocks implies supporting adaptive choices in the queries' DP budgets. For a given block, one can express query i's choice to use block j as using a privacy budget of either (ϵ_i, δ_i) or $(0, 0)$. Lines 7-8 in Alg. (15c) formalize the adaptive choice of both privacy budgets and blocks (requirement **R2**). It does so by leveraging recent work on DP composition under adaptive DP budgets[163]. At each round, $\mathcal{A}$ requests access to a group of blocks blocks_i, on which to run an (ϵ_i, δ_i)-DP query. Sage's Access Control permits the query only if the privacy loss of each block in blocks_i will remain below (ϵ_g, δ_g). Applying our Theorem 1 and[163]'s Theorem 3.3, we prove the following result (proof in[112]):

Theorem 2 (Composition for Sage Block Composition)**.** AdaptiveStreamBlockCompose($\mathcal{A}$,b,r,ϵ_g,δ_g,$\mathcal{W}$) is (ϵ_g, δ_g)-DP if for all k, AccessControl$^k_{\epsilon_g,\delta_g}$ enforces:

$$\Big(\sum_{\substack{i=1 \\ k\in\text{blocks}_i}}^{r} \epsilon_i(v_{<i}) \Big) \leq \epsilon_g \text{ and } \Big(\sum_{\substack{i=1 \\ k\in\text{blocks}_i}}^{r} \delta_i(v_{<i}) \Big) \leq \delta_g.$$

Proof. Denote l_i the highest block index that existed when query i was run. Denote $D^b_{\leq l_i}$ the data blocks that existed at that time. Recall $v_{<i}$ denotes the results from all queries released previous to i. Query i depends on both $v_{<i}$ and $D^b_{\leq l_i}$; the latter is a random variable that is fully determined by $v_{<i}$. Hence, the privacy loss for Alg. (15c) is: $\text{Loss}(v) = \ln\Big(\prod_{i=1}^{r} \frac{P(V_i^0=v_i|v_{<i}, D^0_{\leq l_i})}{P(V_i^1=v_i|v_{<i}, D^1_{\leq l_i})} \Big) = \ln\Big(\prod_{i=1}^{r} \frac{P(V_i^0=v_i|v_{<i})}{P(V_i^1=v_i|v_{<i})} \Big)$,

After applying Theorem 1, we must show that $\forall k$,
$\Big|\ln\Big(\prod_{\substack{i=1 \\ k\in\text{blocks}_i}}^{r} \frac{P(V_i^0=v_i|v_{<i})}{P(V_i^1=v_i|v_{<i})} \Big)\Big| \leq \sum_{\substack{i=1 \\ k\in\text{blocks}_i}}^{r} \epsilon_i$ with probability $\geq (1 - \sum_{\substack{i=1 \\ k\in\text{blocks}_i}}^{r} \delta_i)$, which is the result of Theorem 3.3 in[163]. Since $\mathcal{Q}_i$ is $\epsilon_i(v_{<i}), \delta_i(v_{<i}))$-DP, $\Big|\ln\Big(\frac{P(V_i^0=v_i|v_{<i})}{P(V_i^1=v_i|v_{<i})}\Big)\Big| \leq \epsilon_i(v_{<i})$ with probability $\geq 1 - \delta_i(v_{<i})$. Applying a union bound over all queries for which $k \in \text{blocks}_i$ concludes the proof. □

The implication of the preceding theorem is that under the access control scheme described in §1.4.2, Sage achieves *event-level* (ϵ_g, δ_g)*-DP over the sensitive data stream.*

1.5.4 Strong Composition with Block-Level Accounting.

We now prove strong composition results for Algorithms (15b) and (15c).

Fixed Blocks and DP Parameters: We now show how to use advanced composition results (e.g.[63]) is the context of block composition. This approach requires that both the blocks used by each query and the DP parameters are fixed in advanced, and correspond to Algorithms (15b).

Theorem 3 (Strong Block Composition (fixed DP parameters))**.** BlockCompose($\mathcal{A}$, b, r, $(\epsilon_i, \delta_i)_{i=1}^r$, $\text{blocks}_{i=1}^r$) is (ϵ_g, δ_g)-DP, with:

$$\epsilon_g = \max_k \Big(\sum_{\substack{i=1 \\ k \in blocks_i}}^{r} (e^{\epsilon_i} - 1)\epsilon_i + \sqrt{\sum_{\substack{i=1 \\ k \in blocks_i}}^{r} 2\epsilon_i^2 \log(\frac{1}{\tilde{\delta}})} \Big),$$

$$\delta_g = \tilde{\delta} + \max_k \Big(\sum_{\substack{i=1 \\ k \in blocks_i}}^{r} \delta_i \Big)$$

Proof. After applying Theorem 1, what remains to be shown is that $\forall k$, $\Big| \ln \Big(\prod_{\substack{i=1 \\ i \in \text{blocks}_k}}^{r} \frac{P(V_i^0 = v_i | v_{<i})}{P(V_i^1 = v_i | v_{<i})} \Big) \Big| \leq \sum_{\substack{i=1 \\ k \in blocks_i}}^{r} (e^{\epsilon_i} - 1)\epsilon_i + \sqrt{\sum_{\substack{i=1 \\ k \in blocks_i}}^{r} 2\epsilon_i^2 \log(\frac{1}{\tilde{\delta}})}$, with probability at least $(1 - \tilde{\delta} \sum_{\substack{i=1 \\ k \in \text{blocks}_i}}^{r} \delta_i)$. Using the fact $\mathcal{Q}_i$'s privacy loss is bounded by ϵ_i with probability at least $(1 - \delta_i)$, we know that there exists events E_i and E_i' with joint probability at least $(1 - \delta_i)$ such that for all v_i, $\Big| \ln \big(\frac{P(V_i^0 = v_i | E_i)}{P(V_i^1 = v_i | E_i')} \big) \Big| \leq e^{\epsilon}$. We can now condition the analysis on E_i and E_i', and use Theorem 3.20 of[62] to get that with probability at least $(1 - \tilde{\delta})$, $\Big| \ln \big(\frac{P(V^0 = v | E_i)}{P(V^1 = v | E_i')} \big) \Big| \leq e^{\epsilon_k}$, where $\epsilon_k = \sum_{\substack{i=1 \\ k \in blocks_i}}^{r} (e^{\epsilon_i} - 1)\epsilon_i + \sqrt{\sum_{\substack{i=1 \\ k \in blocks_i}}^{r} 2\epsilon_i^2 \log(\frac{1}{\tilde{\delta}})}$. A union bound on the E_i and E_i' for all $\{i, k \in blocks_i\}$ completes the proof. □

The proof directly extends to the stream setting (yellow parts of Alg. (15b) in the same way as in the proof of Theorem 2.

Adaptive Blocks and DP Parameters: Recall that with either adaptive blocks or DP parameters (or both), Note the fact that DP parameters $(\epsilon_i(v_{<i}, \delta_i(v_{<i})$ depend on history, which is why traditional composition theorems do not apply to the adaptive parameter case. For basic composition, the DP parameters still "sum" under composition. However, as showed in[163], strong composition yields a different formula: while the privacy loss still scales as the square root of the number of queries, the constant is worse than with parameters fixed in advance.

Theorem 4 (Strong Adaptive Stream Block Composition)**.** AdaptiveStreamBlockCompose($\mathcal{A}$, b, r, ϵ_g, δ_g, $\mathcal{W}$) is (ϵ_g, δ_g)-DP, and:

$$\max_k \Big(\sum_{\substack{i=1 \\ k \in \text{blocks}_i}}^{r} \frac{(e^{\epsilon_i} - 1)\epsilon_i}{2} + \sqrt{2\Big(\sum_{\substack{i=1 \\ k \in \text{blocks}_i}}^{r} \epsilon_i^2 + \frac{\epsilon_g^2}{28.04 \log(1/\tilde{\delta})}\Big)} \sqrt{\Big(1 + \frac{1}{2}\log\Big(\frac{28.04 \log(1/\tilde{\delta}) \sum_{\substack{i=1 \\ k \in \text{blocks}_i}}^{r} \epsilon_i^2}{\epsilon_g^2} + 1\Big) \log\Big(\frac{1}{\tilde{\delta}}\Big)\Big)\Big)} \leq \epsilon_g,$$

$$\tilde{\delta} + \max_k \Big(\sum_{\substack{i=1 \\ k \in \text{blocks}_i}}^{r} \delta_i \Big) \leq \delta_g$$

Proof. Similarly to the proof of Theorem 2, we apply Theorem 1, and bound the privacy loss of any block k using Theorem 5.1 of[163]. □

1.5.5 Analysis and Precisions.

Neighboring Datasets. We measure the distance between datasets using the symmetric difference: viewing a dataset as a multiset, twp datasets D and D' are neighboring if their disjunctive union, the elements which are in one of the sets but not in their intersection, is at most one. This definition is not the most standard: most DP work uses the Hamming distance which counts the number of records to change to go from D to D'. Intuitively, under the symmetric difference an attacker can add or remove a record in the dataset. Changing a record corresponds to a symmetric difference of size 2, but a Hamming distance of 1. Changing a record can still be supported under the symmetric difference using group privacy[61]: it is thus slightly weaker but as general.

We chose to use the symmetric difference following PINQ[130], as this definition is better suited to the analysis of DP composition over splits of the dataset (our blocks). Changing one record can indeed affect two blocks (e.g. the timestamp is changed) while adding or removing records can only affect one block.

Neighboring Streams. This notion of neighboring dataset extends pretty directly to streams of data[31;56;59;60]. Two streams D and D' indexed by t are neighboring if there exists an index T such that: for $t < T$ the streams are identical (i.e. $|D_{t<T} \oplus D'_{t<T}| = 0$), and for all $t \geq T$ the streams up to t form neighboring datasets (i.e. $|D_{t\geq T} \oplus D'_{t\geq T}| \leq 1$). This is equivalent to our Algorithm (15b) where the data, though unknown, is fixed in advance with only one record being changed between the two streams.

This definition however is restrictive, because a change in a stream's data will typically affect future data. This is especially true in an ML context, where a record changed in the stream will change the targeting or recommendation algorithms that are trained on this data, which in turn will impact the data collected in the future. Because of this, D and D' will probably differ in a large number of observations following the adversary's change of one record. Interactions described in our Algorithm (15c) model these dependencies. We show in Theorem (2) that if the data change impacts future data only through DP results (e.g. the targeting and recommendation models are DP) and mechanisms outside of the adversary's control (our world variable W), composition results are not affected.

Privacy Loss Semantics. Recall that bounding the privacy loss (Definition 2) with high probability implies DP[97]: if with probability $(1 - \delta)$ over draws from $v \sim V^0$ (or $v \sim V^1$) $|\operatorname{Loss}(v)| \leq \epsilon$, then the interaction generating V^b is (ϵ, δ)-DP.

In this exposition, we implicitly treated DP and bounded loss as equivalent by declaring $\mathcal{Q}_i$s as (ϵ, δ)-DP, but proving composition results using a bound on $\mathcal{Q}_i$'s privacy loss. However, this is not exactly true in general, as (ϵ, δ)-DP implies a bound on privacy loss with weaker parameters, namely that with probability at least $(1 - \frac{2\delta}{\epsilon e^\epsilon})$ the loss is bounded by 2ϵ. In practice, this small difference in not crucial, as the typical Laplace and Gaussian DP mechanisms (and those we use

in Sage) do have the same (ϵ, δ)-DP parameters for their bounds on privacy loss[62]: the Laplace mechanism for $(\epsilon, 0)$-DP implies that the privacy loss is bounded by ϵ and achieving (ϵ, δ)-DP with the Gaussian mechanism implies that the privacy loss is bounded by ϵ with probability at least $(1 - \delta)$.

1.5.6 Blocks Defined by User ID and Other Attributes

Block composition theory can be extended to accommodate user-level privacy and other use cases. The theory shows that one can split a static dataset (Theorem 1) or a data stream (Theorem 2) into disjoint blocks, and run DP queries adaptively on overlapping subsets of the blocks while accounting for privacy at the block level. The theory focused on *time* splits, but the same theorems can be written for splits based on *any attribute whose possible values can be made public*, such as geography, demographics, or user IDs. Consider a workload on a static dataset in which queries combine data from diverse and overlapping subsets of countries, e.g., they compute average salary in each country separately, but also at the level of continents and GDP-based country groups. For such a workload, block composition gives tighter privacy accounting across these queries than traditional composition, though the system will still run out of privacy budget eventually because no new blocks appear in the static database.

As another example, splitting a stream by user ID enables querying or ignoring all observations from a given user, adding support for user-level privacy. Splitting data over user ID requires extra care. If existing user IDs are not knows, each query might select user IDs that do not exist yet, spending their DP budget without adding data. However, making user IDs public can leak information. One approach is to use incrementing user IDs (with this fact public), and periodically run a DP query computing the maximum user ID is use. This would ensure DP, while giving an estimate of the range of user IDs that can be queried. In such a setting, block composition enables fine-grain DP accounting over queries on any subset of the users. While our block theory supports this use case, it suffers from a major practical challenge. New blocks are now created only when new users join the system, so new users must be added at a high rate relative to the model release rate to avoid running out of budget. This is unlikely to happen for mature companies, but may be

Taxi Regression Task		
Pipelines:	**Configuration:**	
Linear Regression (**LR**)	DP Alg.	AdaSSP from[193], (ϵ, δ)-DP
	Config.	Regularization param ρ : 0.1
	Budgets	$(\epsilon, \delta) \in \{(1.0, 10^{-6}), (0.05, 10^{-6})\}$
	Targets	MSE $\in [2.4 \times 10^{-3}, 7 \times 10^{-3}]$
Neural Network (**NN**)	DP Alg.	DP SGD from[2], (ϵ, δ)-DP
	Config.	ReLU, 2 hidden layers (5000/100 nodes) Learning rate: 0.01, Epochs: 3 Batch: 1024, Momentum: 0.9
	Budgets	$(\epsilon, \delta) \in \{(1.0, 10^{-6}), (0.5, 10^{-6})\}$
	Targets	MSE $\in [2 \times 10^{-3}, 7 \times 10^{-3}]$
Avg.Speed x3*	Targets	Absolute error $\in \{1, 5, 7.5, 10, 15\}$ km/h
Criteo Classification Task		
Pipelines:	**Configuration:**	
Logistic Regression (**LG**)	DP Alg.	DP SGD from[126], (ϵ, δ)-DP
	Config.	Learning rate: 0.1, Epochs: 3 Batch: 512
	Budgets	$(\epsilon, \delta) \in \{(1.0, 10^{-6}), (0.25, 10^{-6})\}$
	Targets	Accuracy $\in [0.74, 0.78]$
Neural Network (**NN**)	DP Alg.	DP SGD from[126], (ϵ, δ)-DP
	Config.	ReLU, 2 hidden layers (1024/32 nodes) Learning rate: 0.01, Epochs: 5 Batch: 1024
	Budgets	$(\epsilon, \delta) \in \{(1.0, 10^{-6}), (0.25, 10^{-6})\}$
	Targets	Accuracy $\in [0.74, 0.78]$
Counts x26**	Targets	Absolute error $\in \{0.01, 0.05, 0.10\}$

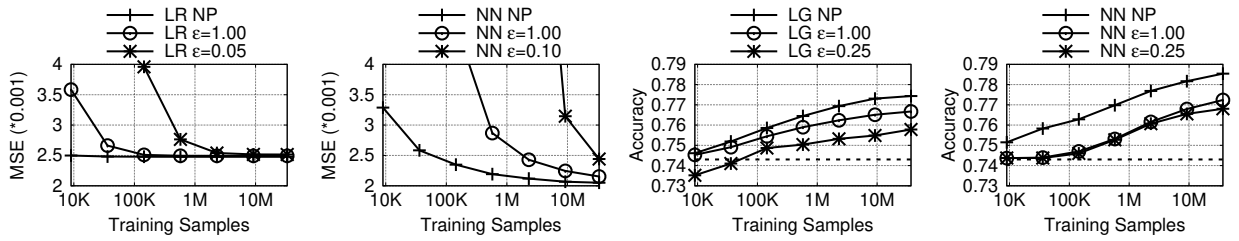
Tab. 11: **Experimental Training Pipelines.** *Three time granularities: hour of day, day of week, week of month. **Histogram of each categorical feature.

possible for emerging startups or hospitals, where the stream of incoming users/patients may be high enough to sustain modest workloads.

1.6 Evaluation

We ask five questions: (**Q1**) Does DP impact the reliability of TFX Training Pipelines? (**Q2**) Does Sage's privacy-adaptive training increase reliability of DP Training Pipelines? (**Q3**) Does block composition improve training over traditional composition? (**Q4**) How do workloads of multiple pipelines perform under Sage's (ϵ_g, δ_g)-DP regime? (**Q5**) How can protected data summaries help alleviate block retirements?

Methodology. We consider two datasets: 37M-samples from three months of NYC taxi rides[146] and 45M ad impressions from Criteo[94]. On the Taxi dataset we define a regression task to predict the duration of each ride using 61 binary features derived from 10 contextual features. We

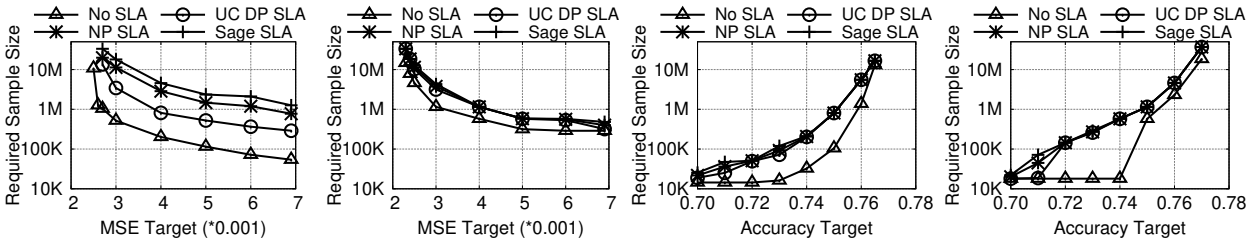


(a) **Taxi LR MSE** (b) **Taxi NN MSE** (c) **Criteo LG Accuracy** (d) **Criteo NN Accuracy**

Fig. 16: **Impacts on TFX Training Pipelines.** Impact of DP on the overall performance of training pipelines. 16a, and 16b show the MSE loss on the Taxi regression task (lower is better). 16c; 16d show the accuracy on the Criteo classification task (higher is better). The dotted lines are naïve model performance.

implement pipelines for a *linear regression (LR)*, a *neural network (NN)*, and three statistics (average speeds at three time granularities). On the Criteo dataset we formulate a binary classification task predicting ad clicks from 13 numeric and 26 categorical features. We implement a *logistic regression (LG)*, a *neural network (NN)*, and histogram pipelines. T 11 shows details.

Training: We make each pipeline DP using known algorithms, shown in T 11. *Validation:* We use the loss, accuracy, and absolute error SLAed validators on the regression, classification, and statistics respectively. *Experiments:* Each model is assigned a quality target from a range of possible values, chosen between the best achievable model, and the performance of a naïve model (predicting the label mean on Taxi, with MSE 0.0069, and the most common label on Criteo, with accuracy 74.3%). Most evaluation uses privacy-adaptive training, so privacy budgets are chosen by Sage, with an upper-bound of $\epsilon = 1$. While no consensus exists on what a reasonable DP budget is, this value is in line with practical prior work[2;127]. Where DP budgets must be fixed, we use values indicated in T 11 which correspond to a large budget ($\epsilon = 1$), and a small budget that varies across tasks and models. Other defaults: 90%::10% train::test ratio; $\eta = 0.05$; $\delta = 10^{-6}$. *Comparisons:* We compare Sage's performance to existing DP composition approaches described in §1.4.2. We ignore the first alternative, which violates the endless execution requirement **R3** and cannot support ML workloads. We compare with the second and third alternatives, which we call *query composition* and *streaming composition*, respectively.



(a) **Taxi LR ACCEPT** (b) **Taxi NN ACCEPT** (c) **Criteo LG ACCEPT** (d) **Criteo NN ACCEPT**

Fig. 17: **Number of Samples Required to ACCEPT** models at achievable quality targets. For MSE targets (Taxi regression 17a, and 17b) small targets are harder to achieve and require more samples. For accuracy targets (Criteo classification 17c, and 17d) high targets are harder and require more samples.

1.6.1 Unreliability of DP Training Pipelines in TFX (Q1)

We first evaluate DP's impact on model training. Fig. 16 shows the loss or accuracy of each model when trained on increasing amounts data and evaluated on 100K held-out samples from their respective datasets. Three versions are shown for each model: the non-DP version (NP), a large DP budget version ($\epsilon = 1$), and a small DP budget configuration with ϵ values that vary across the model and task. For both tasks, the NN requires the most data but outperforms the linear model in the private and non-private settings. The DP LRs catch up to the non-DP version with the full dataset, but the other models trained with SGD require more data. Thus, model quality is impacted by DP but the impact diminishes with more training data. This motivates privacy-adaptive training.

To evaluate DP's impact on validation, we train and validate our models for both tasks, with and without DP. We use TFX's vanilla validators, which compare the model's performance on a test set to the quality metric (MSE for taxi, accuracy for Criteo). We then re-evaluate the models' quality metrics on a separate, 100K-sample held-out set and measure the fraction of models accepted by TFX that violate their targets on the re-evaluation set. With non-DP pipelines (non-DP training and validation), the false acceptance rate is 5.1% and 8.2% for the Taxi and Criteo tasks respectively. With DP pipelines (DP training, DP validation), false acceptance rates hike to 37.9% and 25.2%, motivating SLAed validation.

Dataset	η	No SLA	NP SLA	UC DP SLA	Sage SLA
Taxi	0.01	0.379	0.0019	0.0172	0.0027
	0.05	0.379	0.0034	0.0224	0.0051
Criteo	0.01	0.2515	0.0052	0.0544	0.0018
	0.05	0.2515	0.0065	0.0556	0.0023

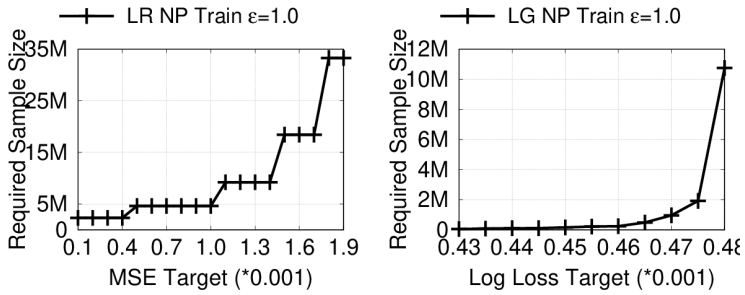
Tab. 12: **Target Violation Rate of ACCEPTed Models.** Violations are across all models separately trained with privacy-adaptive training.

1.6.2 Reliability of DP Training Pipelines in Sage (Q2)

Sage's privacy-adaptive training and SLAed validation are designed to add reliability to DP model training and validation. However, they may come at a cost of increased data requirements over a non-DP test. We evaluate reliability and sample complexity for the SLAed validation ACCEPT test.

T 12 shows the fraction of ACCEPTed models that violate their quality targets when re-evaluated on the 100K-sample held-out set. For two confidences η, we show: (1) *No SLA*, the vanilla TFX validation with no statistical rigor, but where a model's quality is computed with DP. (2) *NP SLA*, a non-DP but statistically rigorous validation. This is the best we can achieve with statistical confidence. (3) *UC DP SLA*, a DP SLAed validation without the correction for DP impact. (4) *Sage SLA*, our DP SLAed validator, with correction. We make three observations. First, the NP SLA violation rates are much lower than the configured η values because we use conservative statistical tests. Second, Sage's DP-corrected validation accepts models with violation rates close to the NP SLA. Slightly higher for the loss SLA and slightly lower for the accuracy SLA, but *well below the configured error rates*. Third, removing the correction increases the violation rate by 5x for the loss SLA and 20x for the accuracy SLA, violating the confidence thresholds in both cases, at least for low η. These results confirm that Sage's SLAed validation is reliable, and that correction for DP is critical to this reliability.

The increased reliability of SLAed validation comes at a cost: SLAed validation requires more data compared to a non-DP test. This new data is supplemented by Sage's privacy-adaptive training. Fig. 17a and 17b show the amount of train+test data required to ACCEPT a model under various loss targets for the Taxi regression task. Fig. 17c and 17d show the same for accuracy targets

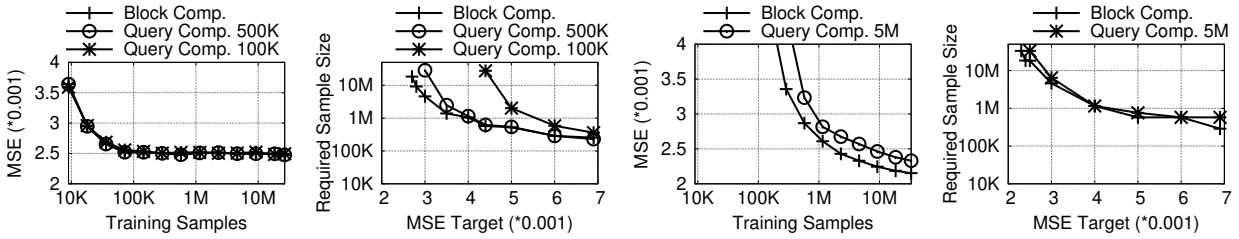


(a) **Taxi LR REJECT** (b) **Criteo LG REJECT**

Fig. 18: **Number of Samples Required to REJECT** models at unachievable quality targets for MSE targets (Taxi regression 18a) and accuracy targets (Criteo classification 18b).

for the Criteo classification task. We make three observations. First, unsurprisingly, non-rigorous validation (No SLA) requires the least data but has a high failure rate because it erroneously accepts models on small sample sizes. Second, the best model accepted by Sage's SLA validation are close to the best model accepted by No SLA. We observe the largest difference in Taxi LR where No SLA accepts MSE targets of 0.0025 while the Sage SLA accepts as low as 0.0027. The best achievable model is slightly impacted by DP, although more data is required. Third, adding a statistical guarantee but no privacy to the validation (NP SLA) already substantially increases sample complexity. Adding DP to the statistical guarantee and applying the DP correction incurs limited additional overhead. The distinction between Sage and NP SLA is barely visible for all but the Taxi LR. For Taxi LR, adding DP accounts for half of the increase over No SLA requiring twice as much data (one data growth step in privacy-adaptive training). Thus, privacy-adaptive training increases reliability of DP training pipelines for reasonable increase in sample complexity.

REJECT Test: We evaluate whether our validators can reject unachievable targets and stop wasteful iterations. Recall from §1.4.3 that the REJECT test requires the models minimizing a loss target on the training set. We use MSE for Taxi and log loss for Criteo which is the metric optimized during training. Fig. 18 shows the sample complexity of REJECT on unachievable targets for LR and LG respectively, with privacy budget $\epsilon = 1.0$ and $\eta = 0.05$. On the Taxi regression task 18a, LR can achieve a loss of 0.0025. The REJECT test is able to reject very low targets, up to 0.001 using less than 5M points, and can reject targets bellow 0.0019. On the Criteo classification task, LG can reach a log loss of 0.492. We can see on 18b that any target lower than 0.475 can



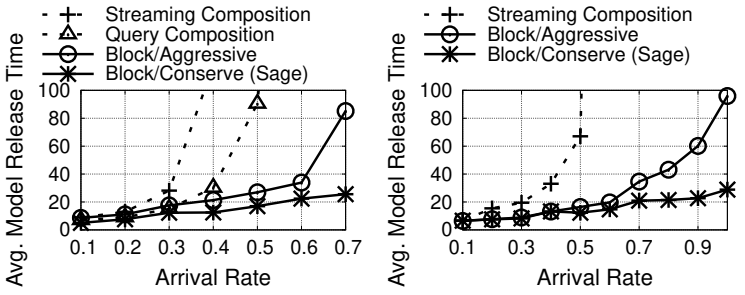
(a) **Taxi LR MSE** (b) **Taxi LR ACCEPT** (c) **Taxi DNN MSE** (d) **Taxi DNN ACCEPT**

Fig. 19: **Block-level vs. Query-level Accounting.** Block-level query accounting provides benefits to model quality and validation.

be rejected with less than 2M points, and targets bellow 0.48 with 11M points. We also evaluate whether our validators incorrectly reject pipelines with achievable targets. For LR and LG models, false rejections are guaranteed to be bellow η. In practice, we observe no false rejections.

1.6.3 Benefit of Block Composition (Q3)

Block composition lets us combine multiple blocks into a dataset, such that each DP query runs over all used blocks with only one noise draw. Without block composition a DP query is split into multiple queries, each operating on a single block, and receiving independent noise. The combined results are more noisy. Fig. 19a and 19c show the model quality of the LR and NN models on the Taxi dataset, when operating on blocks of different sizes, 100K and 500K for the LR, and 5M for the NN. Fig. 19b and 19d show the SLAed validation sample complexity of the same models. We compare these configurations against Sage's combined-block training that allows ML training and validation to operate on their full relevance windows. We can see that block composition helps both the training and validation stages. While LR training (Fig. 19a) performs nearly identically for Sage and block sizes of 100K or 500K (6h of data to a bit more than a day), validation is significantly impacted. The LR cannot be validated with any MSE better than 0.003 with block sizes of 500K, and 0.0044 for blocks of size 100K. Additionally, those targets that can be validated require significantly more data without Sage's block composition: 10x for blocks of size 500K, and almost 100x for blocks of 100K. The NN is more affected at training time. With blocks smaller than 1M points, it cannot even be trained. Even with an enormous block size of 5M, more than ten days of data (Fig. 19c), the partitioned model performs 8% worse than with



(a) **Taxi Dataset** (b) **Criteo Dataset**

Fig. 110: **Average Model Release Time Under Load.**

Sage's block composition. Although on such large blocks validation itself is not much affected, the worse performance means that models can be validated up to an MSE target of 0.0025 (against Sage's 0.0023), and requires twice as much data as with block composition.

1.6.4 Multi-pipeline Workload Performance (Q4)

Last is an end-to-end evaluation of Sage with a workload consisting of a data stream and ML pipelines arriving over discrete time steps. At each step, a number of new data points corresponding approximately to 1 hour of data arrives (16K for Taxi, 267K for Criteo). The time between new pipelines is drawn from a Gamma distribution. When a new pipeline arrives, its sample complexity (number of data points required to reach the target) is drawn from a power law distribution, and a pipeline with the relevant sample complexity is chosen uniformly among our configurations and targets (T 11). Under this workload, we compare the average model release in steady state for four different strategies. This first two leverage *Query Composition* and *Streaming Composition* from prior work, as explained in methodology and § 1.4.3. The other two take advantage of Sage's Block Composition. Both strategies uniformly divide the privacy budget of new blocks among all incomplete pipelines, but differ in how each pipeline uses its budget. *Block/Aggressive* uses as much privacy budget as is available when a pipeline is invoked. *Block/Conserve (Sage)* uses the Privacy-Adaptive Training strategy defined in § 1.4.3.

Fig. 110 shows each strategy's average model release time under increasing load (higher model arrival rate), as the system enforces $(\epsilon_g, \delta_g) = (1.0, 10^{-6})$-DP over the entire stream. We make two observations. First, Sage's block composition is crucial. Query Composition and Streaming

App	Dataset	Obs.	Feat.	Baseline
Ad targeting (classification)	Criteo Kaggle [94]	45M	39	neural net in Kaggle [93]
Ad targeting (classification)	Criteo Full [43]	1.2B	39	regularized linear model
Movie recommendation (classification)	MovieLens [78]	22M	21	matrix factorization [105]

Tab. 13: **Workloads.** Apps and datasets; number of observations and features in each dataset; and baselines used for comparison. All baselines are trained using VW [105].

Composition quickly degrade to off-the-charts release times: supporting more than one model every two hours is not possible and yields release times above three days. On the other hand, strategies leveraging Sage's block composition both provide lower release times, and can support up to 0.7 model arrivals per hour (more than 15 new models per day) and release them within a day. Second, we observe consistently lower release times under the privacy budget conserving strategy. At higher rates, such as 0.7 new models per hour, the difference starts to grow: Block/Conserve has a release time 4x and 2x smaller than Block/Aggressive on Taxi (Fig. 110a) and Criteo (Fig. 110b) respectively. Privacy budget conservation reduces the amount of budget consumed by an individual pipeline, thus allowing new pipelines to use the remaining budget when they arrive.

1.6.5 Benefits of Training Set Minimization As Protected Data Summaries (Q5)

We now show, using different versions of two data-driven applications based on two datasets (see Table 13).

For each application, we use a variety of ML models, including linear or logistic regression, neural networks, and boosted trees (see Table 14).. We split each dataset by time into a training set (80%) and testing set (20%). On the training set, we compute the counts and train our models on windows of growing sizes (to simulate using more and more data blocks), where all windows contain the most recent training data and grow backwards to include older data. This ensures that training occurs on the most recent data (closest to the testing set), and that count tables only include observations from the hot window or the past. We use the testing set to compare the performance of our count algorithms to their raw data counterparts and to the baseline algorithms, using each models' average logistic loss. That is, algorithms predict a probability for each class and are penalized by the logarithm of the probability predicted for the true class: $-\log(p_{true\_class})$. Models are

Dataset	Model	Parameters
Criteo-Kaggle	**B:** neural net (nn)	VW. One 35 nodes hidden layer with tanh activation. LR: 0.15. BP: 25. Passes: 20. Early Terminate: 1.
	logistic regression (log. reg.)	VW. LR: 0.5. BP: 26.
	gradient boosted trees (gbt)	Sklearn. 100 trees with 8 leaves. Subsample: 0.5. LR: 0.1. BP: 8.
Criteo-Full	**B:** ridge regression (rdg. reg.)	VW. L2 penalty: $1.5e^{-8}$. LR: 0.5. BP: 26.
MovieLens Classification	**B:** singular value decomposition (svd)	VW. Rank 10. L2 penalty: 0.001. LR: 0.015. BP: 18. Passes: 20. LR decay: 0.97. PowerT: 0.
	logistic regression (log. reg.)	VW. LR: 0.5. BP: 22. Passes: 5. Early Terminate: 1.
	gradient boosted trees (gbt)	Sklearn. 100 trees with 8 leaves. Subsample: 0.5. LR: 0.1. BP: 8.

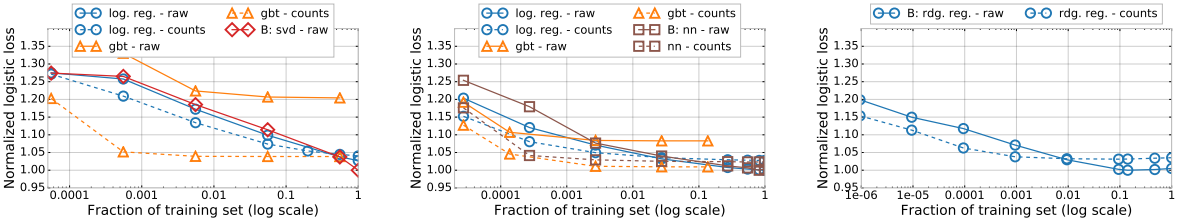
Tab. 14: **Model parameters.** The libraries and parameters used to train each model. The parameters not noted use library defaults. "LR" indicates the learning rate. "BP" indicates the hash featurization's bit precision (only applicable to raw models). "PowerT" exponent controls learning learning rate decay per step. "**B:**" indicates that the model will be used as a baseline. VW and Sklearn denote that the model was trained with Vowpal Wabbit[105] and scikit-learn[154], respectively.

penalized less for incorrect, low-confidence predictions and more for incorrect, high-confidence predictions. Logistic loss is better suited than accuracy for classification problems with imbalanced classes because a model cannot perform well simply by returning the most common class. For all baselines, we apply any dimensionality reduction mechanisms (e.g., hash featurization[194]) that those models typically apply to strengthen them.

All graphs report loss normalized by the baseline model trained on the *entire training data*. Lower values are better in all graphs: a value of 1 or less means that we beat the baseline's best performance; and a value > 1 means that we do worse than the baseline. For completeness, we specify our baselines' performance: MovieLens classification matrix factorization has a logistic loss of 0.537; Criteo-Kaggle neural network has a logistic loss of 0.467; and Criteo-Full ridge regression has a logistic loss of 0.136.

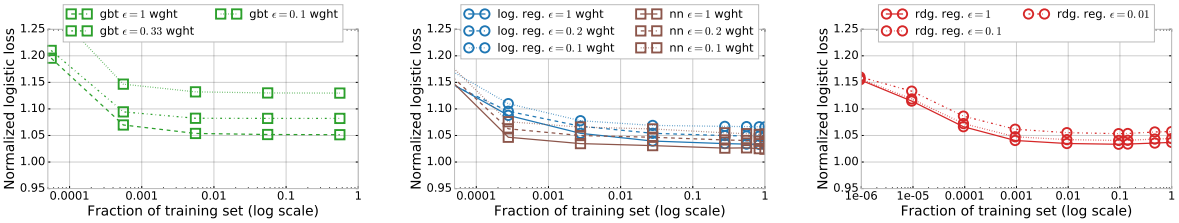
We next show how training with counts requires accessing less data (thus saving privacy budget), and how our mechanisms reduce the impact of noise to retain good utility.

Training with count reduces data requirements To reduce data needs, the count models must converge faster than raw-data models (reach their best performance with less data), and perform on par with state-of-the-art baselines. Fig. 111 shows the performance of several linear and nonlinear



(a) **MovieLens classification** (b) **Criteo-Kaggle classification** (c) **Criteo-Full classification**

Fig. 111: **Normalized losses for raw and count algorithms.** "B:" denotes the baseline model. Count algorithms converge faster than raw data algorithms, to results that are within 4% on MovieLens, and within 2% and 4% on Criteo Kaggle and full respectively.



(a) **MovieLens boosted tree** (b) **Criteo-Kaggle algorithms** (c) **Criteo-Full ridge regression**

Fig. 112: **Impact of data protection.** Results are normalized by the baselines. We fix $k = 1$ and vary ϵ, the privacy budget. Fig. 112a and Fig. 112b show results using the weighted noise (denoted wght). On MovieLens our weighting scheme is crucial to hide 1 observation. On Criteo we can easily hide 1 observation with little performance degradation and can hide up to 100 observations while remaining within 5% of the baseline.

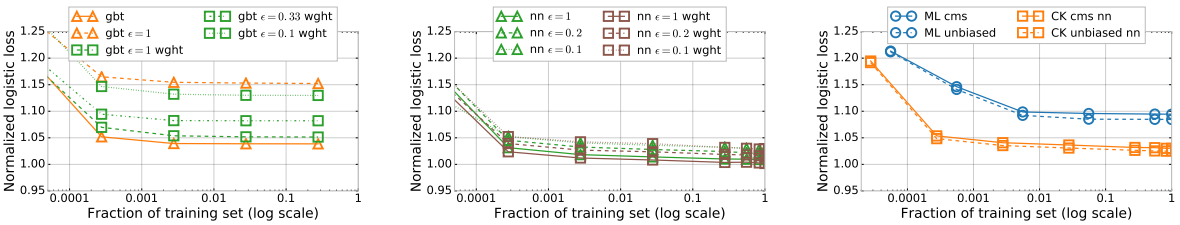
models, on raw and count-featurized data. We see that training with counts requires less data. On both Criteo and MovieLens the best count-featurized algorithm approaches the best raw-data algorithm by training on *1% of the data or less*. On Criteo-Kaggle (Fig. 111b), the count-featurized neural network comes within 3% of the baseline when trained on 0.4% of the data and performs within 1.7% of the baseline with 28% of the training data. On Criteo-Full (Fig. 111c), the count-featurized ridge regression model comes within 3.3% of the baseline with only 0.1% of the data, and within 2.5% when trained on 15% of the data. These results show that models trained on count-featurized data can perform close to raw models in both balanced and very imbalanced datasets (Criteo Full and Kaggle's respective click rates are 3% and 25%). On MovieLens (Fig. 111a), the count-featurized boosted tree needs only 0.8% of the data to get within 4% of the baseline, or match the raw data logistic regression. Because counts summarize history and reduce dimensionality,

they allow algorithms to perform well with very little data. We say that they *converge faster* than raw data algorithms.

Impact of DP noise. We have shown that count-featurized algorithms converge faster than models trained on raw data. However the count tables have to be differentially private. The amount of noise to add depends on the desired privacy guarantee, parameterized by ϵ (smaller is more private), but also on the number of features (see Table 13) and CMS hash functions (five here), through the formula from §1.4.4. In this section we evaluate the noise's impact on performance, as well as Sage's two mechanisms that increase data utility: automatic weighted noise infusion and the use of private count-median sketches.

Fig. 112 shows the performance of different algorithms and datasets with different privacy budgets ϵ. We find that Sage can protect observations with minimal performance loss. When $\epsilon = 1$, the boosted tree model on the MovieLens dataset remains within 5% of the baseline with only 1% of the training data. The logistic regression and neural network models on the Criteo-Kaggle dataset perform within 2.7% and 1.8% of the baseline respectively, and the Criteo-Full ridge regression is within 3%. All Criteo models also come within 5% of their respective baseline with a privacy budget as small as $\epsilon = 0.2$. The Criteo-Full ridge regression performance degrades less than models on other datasets when the noise increases. For instance, it degrades by less than 1% with ϵ going from 1 to 0.1, while the Criteo-Kaggle neural network loses 6.5%. This is explained by the fact that the amount of noise required to make a query differentially private is not related to the size of the dataset. The Criteo-Full dataset is much larger, so the additional noise is much smaller relative to the counts.

Weighted noise infusion. Weighted noise infusion is integral to the protection of past observations with minimal performance cost. Fig. 113a shows the impact of noise on the boosted tree for the MovieLens dataset. Without weighting the privacy budget of different features, the model performs 15% worse than the baseline even for $\epsilon = 1$. With non-private weighting, the MovieLens model performs at 5% of the baseline. The weighted noise infusion technique is thus critical to maintaining performance on the MovieLens dataset. Intuitively, this is because the users making



(a) **MovieLens boosted tree** (b) **Criteo-Kaggle neural network** (c) **Sketch comparison**

Fig. 113: **Impact of data protection (continued).** Results are normalized to the baselines. We fix $k = 1$ and vary ϵ, the privacy budget. (a) Without the feature weighting trick the gradient boosted trees perform unacceptably poorly. (b) The weighting trick marginally improves the performance of Criteo-Kaggle models over equally distributing the privacy budget. (c) Private count-median sketch improves performance in both MovieLens (ML) and Criteo-Kaggle (CK) models with $\epsilon = 1$.

the rating and the movie being rated are the most important features when predicting ratings. Most users rate relatively few movies, and a long tail of movies are rarely rated, so their respective counts are quickly overwhelmed by the noise when the privacy budget is equally distributed among all features. The Criteo models do not depend as much on the weighting trick, since they do not rely on a few features with small counts. Noise weighting is still beneficial, though: e.g., the Criteo-Kaggle neural network gains about 0.5% of performance, as shown in Fig. 113b.

Private count-median sketch. Another technique that Sage uses to reduce the impact of noise is to switch to a private count-median sketch. As noted in §1.4.4, the count-min sketch will exhibit a strong downward bias when initialized with differentially private noise, because taking the minimum of multiple observations will select the most extreme negative noise values. The count-median sketch uses the median instead of the minimum and does not suffer from this effect. Fig. 113c shows that when noise is added, the count-median sketch improves performance over the count-min sketch by around 0.5%, on MovieLens and Criteo-Kaggle. When combined with weighted noise infusion, the private count-median sketch is less useful at first, since the noise is small on features with small counts. However, it provides an improvement for lower ϵ. For instance, the MovieLens boosted tree improves by 0.5% even after noise weighting for $\epsilon = 0.10$.

1.7 Related Work

Sage's main contribution – block composition – is related to *DP composition theory*. Basic[58] and strong[63;95] composition theorems give the DP guarantee for multiple queries with adaptively chosen computation. McSherry[130] and Zhang, et.al.[210] show that non-adaptive queries over non-overlapping subsets of data can share the DP budget. Rogers, et.al.[164] analyze composition under adaptive DP parameters, which is crucial to our block composition. These works all consider fixed datasets and query-level accounting.

Compared to all these works, our main contribution is to formalize the new block-level DP interaction model, which supports ML workloads on growing databases while enforcing a global DP semantic without running out of budget. This model sits between traditional DP interaction with static data, and streaming DP working only on current data. In proving our interaction model DP we leverage prior theoretical results and analysis methods. However, the most comprehensive prior interaction model[164] did not support all our requirements, such as interactions with adaptively chosen data subsets, or future data being impacted by previous queries.

Streaming DP[31;56;59;60] extends DP to data streams but is restrictive for ML. Data is consumed once and assumed to never be used again. This enables stronger guarantees, as data need not even be kept internally. However, training ML models often requires multiple passes over the data.

Cummings, et.al.[44] consider *DP over growing databases*. They focus on theoretical analysis and study two setups. In the fist setup, they also run DP workloads on exponentially growing data sizes. However, their approach only supports linear queries, with a runtime exponential in the data dimension and hence impractical. In the second setup, they focus on training a single convex ML model and show that it can use new data to keep improving. Supporting ML workloads would require splitting the privacy budget for the whole stream among models, creating a running out of privacy budget challenge.

A few *DP systems* exist, but none focuses on streams or ML. PINQ[130] and its generalization wPINQ[158] give a SQL-like interface to perform DP queries. They introduce the partition operator allowing *parallel composition*, which resembles Sage's block composition. However, this

operator only supports non-adaptive parallel computations on non-overlapping partitions, which is insufficient for ML. Airavat[165] provides a MapReduce interface and supports a strong threat model against actively malicious developers. They adopt a perspective similar to ours, integrating DP with access control. GUPT[139] supports automatic privacy budget allocation and lets programmers specify accuracy targets for arbitrary DP programs with a real-valued output; it is hence applicable to computing summary statistics but not to training ML models. All these works focus on static datasets and adopt a generic, query-level accounting approach that applies to any workload. Query-level accounting would force them to run out of privacy budget if unused data were available. Block-level accounting avoids this limitation but applies to workloads with specific data interaction characteristics (§1.4.2).

1.8 Summary

As companies disseminate ML models trained over sensitive data to untrusted domains, it is crucial to start controlling data leakage through these models. We presented *Sage*, the first ML platform that enforces a global DP guarantee across all models released from sensitive data streams. Its main contributions are its *block-level accounting* that permits endless operation on streams and its *privacy-adaptive training* that lets developers control DP model quality. The key enabler of both techniques is our systems focus on ML training workloads rather than DP ML's typical focus on individual training algorithms. While individual algorithms see either a static dataset or an online training regime, workloads interact with *growing databases*. Across executions of multiple algorithms, new data becomes available (helping to renew privacy budgets and allow endless operation) and old data is reused (allowing training of models on increasingly large datasets to appease the effect of DP noise on model quality).

We believe that this systems perspective on DP ML presents further opportunities worth pursuing in the future. Chief among them is how to allocate data, privacy parameters, and compute resources to conserve privacy budget while training models efficiently to their quality targets. Sage proposes a specific heuristic for allocating the first two resources (§1.4.3), but leaves unexplored tradeoffs between data and compute resources. To conserve budgets, we use as much data as is

available in the database when a model is invoked, with the lowest privacy budget. This gives us the best utilization of the privacy resource. But training on more data consumes more compute resources. Identifying principled approaches to perform these allocations is an open problem.

A key limitation of this work is its focus on event-level privacy, a semantic that is insufficient when groups of correlated observations can reveal sensitive information. The best known example of such correlation happens when a user contributes multiple observations, but other examples include repeated measurements of a phenomenon over time, or users and their friends on a social network. In such cases, observations are all correlated and can reveal sensitive information, such as a user's demographic attributes, despite event-level DP. It should be noted that even in the face of correlated data DP holds for each individual observation: other correlated observations constitute side information, to which DP is known to be resilient. Still, to increase protection, an exciting area of future work is to add support for and evaluate user-level privacy. Our block accounting theory is amenable to this semantic (§1.5.6), but finding settings where the semantic can be sustained without running out of budget is an open challenge.

Chapter 2

Security Guarantees Against Adversarial Examples

In addition to leaking information about sensitive training data, deploying ML models in applications also introduces new attack surfaces, the second challenge we described. In this chapter we present a defense against one such ML specific attack, adversarial examples. This chapter is based on our PixelDP paper[106], introducing the first defense to provide a guaranteed level of accuracy under adversarial examples attack, and also scale to large deep learning models and datasets.

2.1 Motivation

Deep neural networks (DNNs) perform exceptionally well on many machine learning tasks, including safety- and security-sensitive applications such as self-driving cars[20], malware classification[151], face recognition[150], and critical infrastructure[212]. Robustness against malicious behavior is important in many of these applications, yet in recent years it has become clear that DNNs are vulnerable to a broad range of attacks. Among these attacks – broadly surveyed in[149] – are *adversarial examples*: the adversary finds small perturbations to correctly classified inputs that cause a DNN to produce an erroneous prediction, possibly of the adversary's choosing[180]. Adversarial examples pose serious threats to security-critical applications. A classic example is an adversary attaching a small, human-imperceptible sticker onto a stop sign that causes a self-driving car to recognize it as a yield sign. Adversarial examples have also been demonstrated in domains such as reinforcement learning[101] and generative models[100].

Since the initial demonstration of adversarial examples[180], numerous attacks and defenses have been proposed, each building on one another. Initially, most defenses used *best-effort* approaches and were broken soon after introduction. Model distillation, proposed as a robust defense in[148], was subsequently broken in[30]. Other work[119] claimed that adversarial examples are unlikely to fool machine learning (ML) models in the real-world, due to the rotation and scaling introduced by even the slightest camera movements. However,[7] demonstrated a new attack strategy that is robust to rotation and scaling. While this back-and-forth has advanced the state of the art, recently the community has started to recognize that rigorous, theory-backed, defensive approaches are required to put us off this arms race.

Accordingly, a new set of *certified defenses* have emerged over the past year, that provide rigorous guarantees of robustness against norm-bounded attacks[40;161;197]. These works alter the learning methods to both optimize for robustness against attack at training time and permit provable robustness checks at inference time. At present, these methods tend to be tied to internal network details, such as the type of activation functions and the network architecture. They struggle to generalize across different types of DNNs and have only been evaluated on small networks and datasets.

2.2 The PixelDP Defense

We propose a new and orthogonal approach to certified robustness against adversarial examples that is *broadly applicable, generic, and scalable*. We observe for the first time a connection between *differential privacy* (DP), a cryptography-inspired formalism, and a definition of robustness against norm-bounded adversarial examples in ML. We leverage this connection to develop *PixelDP*, the first certified defense we are aware of that both scales to large networks and datasets (such as Google's Inception network trained on ImageNet) and can be adapted broadly to arbitrary DNN architectures. Our approach can even be incorporated with no structural changes in the target network (e.g., through a separate auto-encoder as described in Section 2.4.1). We provide a brief overview of our approach below along with the section references that detail the corresponding parts.

§2.3 establishes the DP-robustness connection formally (our first contribution). To give the intuition, DP is a framework for randomizing computations running on databases such that a small change in the database (removing or altering one row or a small set of rows) is guaranteed to result in a bounded change in the distribution over the algorithm's outputs. Separately, robustness against adversarial examples can be defined as ensuring that small changes in the input of an ML predictor (such as changing a few pixels in an image in the case of an l_0-norm attack) will not result in drastic changes to its predictions (such as changing its label from a stop to a yield sign). Thus, if we think of a DNN's inputs (e.g., images) as databases in DP parlance, and individual features (e.g., pixels) as rows in DP, we observe that randomizing the outputs of a DNN's prediction function to

enforce DP on a small number of pixels in an image *guarantees* robustness of predictions against adversarial examples that can change up to that number of pixels. The connection can be expanded to standard attack norms, including l_1, l_2, and l_∞ norms.

§2.4 describes *PixelDP*, the first certified defense against norm-bounded adversarial examples based on differential privacy (our second contribution). Incorporating DP into the learning procedure to increase robustness to adversarial examples requires is completely different and orthogonal to using DP to preserve the privacy of the training set, the focus of prior DP ML literature[129;1;35] (as § 2.7 explains). A PixelDP DNN includes in its architecture a *DP noise layer* that randomizes the network's computation, to enforce DP bounds on how much the distribution over its predictions can change with small, norm-bounded changes in the input. At inference time, we leverage these DP bounds to implement a certified robustness check for individual predictions. Passing the check for a given input *guarantees* that no perturbation exists up to a particular size that causes the network to change its prediction. The robustness certificate can be used to either act exclusively on robust predictions, or to lower-bound the network's accuracy under attack on a test set.

§2.6 presents the first experimental evaluation of a certified adversarial-examples defense for the Inception network trained on the ImageNet dataset (our third contribution). We additionally evaluate PixelDP on various network architectures for four other datasets (CIFAR-10, CIFAR-100, SVHN, MNIST), on which previous defenses – both best effort and certified – are usually evaluated. Our results indicate that PixelDP is (1) as effective at defending against attacks as today's state-of-the-art, best-effort defense[120] and (2) more scalable and broadly applicable than a prior certified defense.

Our experience points to DP as a uniquely generic, broadly applicable, and flexible foundation for certified defense against norm-bounded adversarial examples (§2.5.2, §2.7). We credit these properties to the *post-processing property of DP*, which lets us incorporate the certified defense in a network-agnostic way.

2.3 Background

2.3.1 Adversarial ML Background

An ML model can be viewed as a function mapping inputs – typically a vector of numerical feature values – to an output (a label for multiclass classification and a real number for regression). Focusing on multiclass classification, we define a *model* as a function $f\colon \mathbb{R}^n \to \mathcal{K}$ that maps n-dimensional inputs to a label in the set $\mathcal{K} = \{1, \ldots, K\}$ of all possible labels. Such models typically map an input x to a vector of scores $y(x) = (y_1(x), \ldots, y_K(x))$, such that $y_k(x) \in [0, 1]$ and $\sum_{k=1}^{K} y_k(x) = 1$. These scores are interpreted as a probability distribution over the labels, and the model returns the label with highest probability, i.e., $f(x) = \arg\max_{k \in \mathcal{K}} y_k(x)$. We denote the function that maps input x to y as Q and call it the *scoring function*; we denote the function that gives the ultimate prediction for input x as f and call it the *prediction procedure*.

Adversarial Examples. Adversarial examples are a class of attack against ML models, studied particularly on deep neural networks for multiclass image classification. The attacker constructs a small change to a given, fixed input, that wildly changes the predicted output. Notationally, if the input is x, we denote an adversarial version of that input by $x + \alpha$, where α is the change or perturbation introduced by the attacker. When x is a vector of pixels (for images), then x_i is the i'th pixel in the image and α_i is the change to the i'th pixel.

It is natural to constrain the amount of change an attacker is allowed to make to the input, and often this is measured by the p-norm of the change, denoted by $\|\alpha\|_p$. For $1 \leq p < \infty$, the p-norm of α is defined by $\|\alpha\|_p = (\sum_{i=1}^{n} |\alpha_i|^p)^{1/p}$; for $p = \infty$, it is $\|\alpha\|_\infty = \max_i |\alpha_i|$. Also commonly used is the 0-norm (which is technically not a norm): $\|\alpha\|_0 = |\{i : \alpha_i \neq 0\}|$. A small 0-norm attack is permitted to arbitrarily change a few entries of the input; for example, an attack on the image recognition system for self-driving cars based on putting a sticker in the field of vision is such an attack[66]. Small p-norm attacks for larger values of p (including $p = \infty$) require the changes to the pixels to be small in an aggregate sense, but the changes may be spread out over many or all features. A change in the lighting condition of an image may correspond to such an attack[104;155].

The latter attacks are generally considered more powerful, as they can easily remain invisible to human observers. Other attacks that are not amenable to norm bounding exist[201;176;200], but in this chapter we deal exclusively with norm-bounded attacks.

Let $B_p(r) := \{\alpha \in \mathbb{R}^n : \|\alpha\|_p \leq r\}$ be the p-norm ball of radius r. For a given classification model, f, and a fixed input, $x \in \mathbb{R}^n$, an attacker is able to craft a successful adversarial example of size L for a given p-norm if they find $\alpha \in B_p(L)$ such that $f(x+\alpha) \neq f(x)$. The attacker thus tries to find a small change to x that will change the predicted label.

Robustness Definition. Intuitively, a predictive model may be regarded as *robust* to adversarial examples if its output is *insensitive* to small changes to any *plausible* input that may be encountered in deployment. To formalize this notion, we must first establish what qualifies as a *plausible* input. This is difficult: the adversarial examples literature has not settled on such a definition. Instead, model robustness is typically assessed on inputs from a test set that are not used in model training – similar to how accuracy is assessed on a test set and not a property on all plausible inputs. We adopt this view of robustness.

Next, given an input, we must establish a definition for *insensitivity* to small changes to the input. We say a model f is insensitive, or *robust*, to attacks of p-norm L on a given input x if $f(x) = f(x+\alpha)$ for all $\alpha \in B_p(L)$. If f is a multiclass classification model based on label scores (as in §2.3.1), this is equivalent to:

$$\forall \alpha \in B_p(L) \;.\; y_k(x + \alpha) > \max_{i:i\neq k} y_i(x + \alpha), \quad (2.1)$$

where $k := f(x)$. A small change in the input does not alter the scores so much as to change the predicted label.

2.3.2 DP Background

DP is concerned with whether the output of a computation over a database can reveal information about individual records in the database. To prevent such information leakage, randomness is introduced into the computation to hide details of individual records.

A randomized algorithm A that takes as input a database d and outputs a value in a space O is said to satisfy (ϵ, δ)-*DP with respect to a metric ρ over databases* if, for any databases d and d' with $\rho(d, d') \leq 1$, and for any subset of possible outputs $S \subseteq O$, we have

$$P(A(d) \in S) \leq e^{\epsilon} P(A(d') \in S) + \delta. \tag{2.2}$$

Here, $\epsilon > 0$ and $\delta \in [0, 1]$ are parameters that quantify the strength of the privacy guarantee. In the standard DP definition, the metric ρ is the Hamming metric, which simply counts the number of entries that differ in the two databases. For small ϵ and δ, the standard (ϵ, δ)-DP guarantee implies that changing a single entry in the database cannot change the output distribution very much. DP also applies to general metrics ρ[33], including p-norms relevant to norm-based adversarial examples.

Our approach relies on two key properties of DP. First is the well-known *post-processing property*: any computation applied to the output of an (ϵ, δ)-DP algorithm remains (ϵ, δ)-DP. Second is the *expected output stability property*, a rather obvious but not previously enunciated property that we prove in Lemma 1: the expected value of an (ϵ, δ)-DP algorithm with bounded output is not sensitive to small changes in the input.

Lemma 1. (General Expected Output Stability Bound) Suppose a randomized function A, with bounded output $A(x) \in [a, b],\ a, b \in \mathbb{R}$, with $a \leq 0 \leq b$, satisfies (ϵ, δ)-DP. Let $A_+(x) = \max(0, A(x))$ and $A_-(x) = -\min(0, A(x))$, so that $A(x) = A_+(x) - A_-(x)$. Then the expected value of its output meets the following property: for all $\alpha \in B_p(1)$,

$$\mathbb{E}(A(x + \alpha)) \leq e^{\epsilon}\mathbb{E}(A_+(x)) - e^{-\epsilon}\mathbb{E}(A_-(x)) + b\delta - e^{-\epsilon}a\delta,$$
$$\mathbb{E}(A(x + \alpha)) \geq e^{-\epsilon}\mathbb{E}(A_+(x)) - e^{\epsilon}\mathbb{E}(A_-(x)) - e^{-\epsilon}b\delta + a\delta.$$

The expectation is taken over the randomness in A.

Proof. Consider any $\alpha \in B_p(1)$, and let $x' := x + \alpha$. Observe that $\mathbb{E}(A_+(x)) = \int_0^b P(A(x) > t)dt$, so by the (ϵ, δ)-DP property of A via Equation (2.2), we have $\mathbb{E}(A_+(x')) \leq e^{\epsilon}\mathbb{E}(A_+(x)) + b\delta$ and $\mathbb{E}(A_+(x')) \geq e^{-\epsilon}\mathbb{E}(A_+(x)) - e^{-\epsilon}b\delta$. Similarly, $\mathbb{E}(A_-(x')) \leq e^{\epsilon}\mathbb{E}(A_-(x)) - a\delta$ and $\mathbb{E}(A_-(x')) \geq e^{-\epsilon}\mathbb{E}(A_+(x)) + e^{-\epsilon}a\delta$. Putting these four inequalities together concludes the proof. □

2.3.3 DP-Robustness Connection

The intuition behind using DP to provide robustness to adversarial examples is to create a *DP scoring function* such that, given an input example, the predictions are DP with regards to the features of the input (e.g. the pixels of an image). In this setting, we can derive stability bounds for the expected output of the DP function using Lemma 1. The bounds, combined with Equation (2.1), give a rigorous condition (or *certification*) for robustness to adversarial examples.

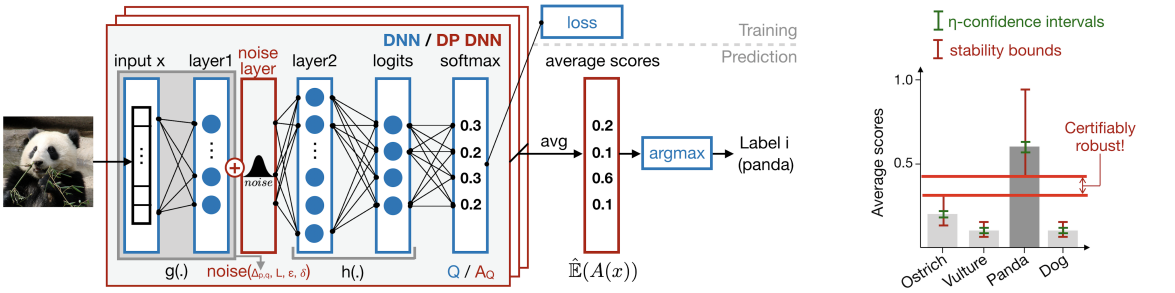
Formally, regard the feature values (e.g., pixels) of an input x as the records in a database, and consider a randomized scoring function A that, on input x, outputs scores $(y_1(x), \ldots, y_K(x))$ (with $y_k(x) \in [0, 1]$ and $\sum_{k=1}^{K} y_k(x) = 1$). We say that A is an *(ϵ, δ)-pixel-level differentially private* (or *(ϵ, δ)-PixelDP*) function if it satisfies (ϵ, δ)-DP (for a given metric). This is formally equivalent to the standard definition of DP, but we use this terminology to emphasize the context in which we apply the definition, which is fundamentally different than the context in which DP is traditionally applied in ML (see §2.7 for distinction).

Lemma 1 directly implies bounds on the expected outcome on an (ϵ, δ)-PixelDP scoring function:

Corollary 1. Suppose a randomized function A satisfies (ϵ, δ)-PixelDP with respect to a p-norm metric, and where $A(x) = (y_1(x), \ldots, y_K(x)),\ y_k(x) \in [0, 1]$:

$$\forall k, \forall \alpha \in B_p(1) \,.\, \mathbb{E}(y_k(x)) \leq e^{\epsilon} \mathbb{E}(y_k(x + \alpha)) + \delta. \tag{2.3}$$

Proof. For any k apply Lemma 1 with $a = 0$ and $b = 1$. □



(a) PixelDP DNN Architecture (b) Robustness Test Example

Fig. 21: **Architecture.** (a) In blue, the original DNN. In red, the noise layer that provides the (ϵ, δ)-DP guarantees. The noise can be added to the inputs *or* any of the following layers, but the distribution is rescaled by the sensitivity $\Delta_{p,q}$ of the computation performed by each layer before the noise layer. The DNN is trained with the original loss and optimizer (e.g., Momentum stochastic gradient descent). Predictions repeatedly call the (ϵ, δ)-DP DNN to measure its empirical expectation over the scores. (b) After adding the bounds for the measurement error between the empirical and true expectation (green) and the stability bounds from Lemma 1 for a given attack size L_{attack} (red), the prediction is certified robust to this attack size if the lower bound of the $\arg\max$ label does not overlap with the upper bound of any other labels.

Our approach is to transform a model's scoring function into a *randomized* (ϵ, δ)*-PixelDP scoring function*, $A(x)$, and then have the model's prediction procedure, f, use A's *expected output over the DP noise*, $\mathbb{E}(A(x))$, as the label probability vector from which to pick the $\arg\max$. I.e., $f(x) = \arg\max_{k \in \mathcal{K}} \mathbb{E}(A_k(x))$. We prove that a model constructed this way allows the following robustness certification to adversarial examples:

Proposition 5. (Robustness Condition) Suppose A satisfies (ϵ, δ)-PixelDP with respect to a p-norm metric. For any input x, if for some $k \in \mathcal{K}$,

$$\mathbb{E}(A_k(x)) > e^{2\epsilon} \max_{i:i \neq k} \mathbb{E}(A_i(x)) + (1 + e^{\epsilon})\delta, \tag{2.4}$$

then the multiclass classification model based on label probability vector $y(x) = (\mathbb{E}(A_1(x)), \ldots, \mathbb{E}(A_K(x)))$ is robust to attacks α of size $\|\alpha\|_p \leq 1$ on input x.

Proof. Consider any $\alpha \in B_p(1)$, and let $x' := x + \alpha$. From Equation (2.3), we have:

$$\mathbb{E}(A_k(x)) \leq e^{\epsilon}\mathbb{E}(A_k(x')) + \delta, \qquad (a)$$

$$\mathbb{E}(A_i(x')) \leq e^{\epsilon}\mathbb{E}(A_i(x)) + \delta, \quad i \neq k. \qquad (b)$$

Equation (a) gives a lower-bound on $\mathbb{E}(A_k(x'))$; Equation (b) gives an upper-bound on $\max_{i \neq k} \mathbb{E}(A_i(x'))$. The hypothesis in the proposition statement (Equation (2.4)) implies that the lower-bound of the

expected score for label k is strictly higher than the upper-bound for the expected score for any other label, which in turn implies the condition from Equation (2.1) for robustness at x. To spell it out:

$$\begin{aligned}
\mathbb{E}(A_k(x')) &\overset{\text{Eq(a)}}{\geq} \frac{\mathbb{E}(A_k(x)) - \delta}{e^\epsilon} \\
&\overset{\text{Eq(4)}}{>} \frac{e^{2\epsilon} \max_{i:i\neq k} \mathbb{E}(A_i(x)) + (1 + e^\epsilon)\delta - \delta}{e^\epsilon} \\
&= e^\epsilon \max_{i:i\neq k} \mathbb{E}(A_i(x)) + \delta \\
&\overset{\text{Eq(b)}}{\geq} \max_{i:i\neq k} \mathbb{E}(A_i(x')) \\
\implies \mathbb{E}(A_k(x')) &> \max_{i:i\neq k} \mathbb{E}(A_i(x+\alpha)) \; \forall \alpha \in B_p(1),
\end{aligned}$$

the very definition of robustness at x (Equation (2.1)). □

The preceding certification test is *exact* regardless of the value of the δ parameter of differential privacy: there is no failure probability in this test. The test applies only to attacks of p-norm size of 1, however all preceding results generalize to *attacks of p-norm size L*, i.e., when $\|\alpha\|_p \leq L$, by applying group privacy[61]. The next section shows how to apply group privacy (§2.4.1) and generalize the certification test to make it practical (§2.5.1).

2.4 Design

PixelDP is a certified defense against p-norm bounded adversarial example attacks built on the preceding DP-robustness connection. Fig. 21a shows an example PixelDP DNN architecture for multi-class image classification. The original architecture is shown in blue; the changes introduced to make it PixelDP are shown in red. Denote Q the original DNN's scoring function; it is a deterministic map from images x to a probability distribution over the K labels $Q(x) = (y_1(x), \ldots, y_K(x))$. The vulnerability to adversarial examples stems from the unbounded sensitivity of Q with respect to p-norm changes in the input. Making the DNN (ϵ, δ)-PixelDP involves adding *calibrated noise* to turn Q into an (ϵ, δ)-DP randomized function A_Q; the expected output of that function will have bounded sensitivity to p-norm changes in the input. We achieve this by introducing a *noise layer* (shown in red in Fig. 21a) that adds zero-mean noise to the out-

put of the layer preceding it (layer1 in Fig. 21a). The noise is drawn from a Laplace or Gaussian distribution and its standard deviation is proportional to: (1) L, the p-norm attack bound for which we are constructing the network and (2) Δ, the sensitivity of the pre-noise computation (the grey box in Fig. 21a) with respect to p-norm input changes.

Training an (ϵ, δ)-PixelDP network is similar to training the original network: we use the original loss and optimizer, such as stochastic gradient descent. The major difference is that we alter the pre-noise computation to *constrain its sensitivity* with regards to p-norm input changes. Denote $Q(x) = h(g(x))$, where g is the pre-noise computation and h is the subsequent computation that produces $Q(x)$ in the original network. We leverage known techniques, reviewed in §2.4.2, to transform g into another function, $\tilde{g}$, that has a fixed sensitivity (Δ) to p-norm input changes. We then add the noise layer to the output of $\tilde{g}$, with a standard deviation scaled by Δ and L to ensure (ϵ, δ)-PixelDP for p-norm changes of size L. Denote the resulting scoring function of the PixelDP network: $A_Q(x) = h(\tilde{g}(x) + noise(\Delta, L, \epsilon, \delta))$, where $noise(.)$ is the function implementing the Laplace/Gaussian draw. Assuming that the noise layer is placed such that h only processes the DP output of $\tilde{g}(x)$ without accessing x again (i.e., no skip layers exist from pre-noise to post-noise computation), the post-processing property of DP ensures that $A_Q(x)$ also satisfies (ϵ, δ)-PixelDP for p-norm changes of size L.

Prediction on the (ϵ, δ)-PixelDP scoring function, $A_Q(x)$, affords the robustness certification in Proposition 5 if the prediction procedure uses the *expected scores*, $\mathbb{E}(A_Q(x))$, to select the winning label for any input x. Unfortunately, due to the potentially complex nature of the post-noise computation, h, we cannot compute this output expectation analytically. We therefore resort to Monte Carlo methods to estimate it at prediction time and develop an approximate version of the robustness certification in Proposition 5 that uses standard techniques from probability theory to account for the estimation error (§2.5.1). Specifically, given input x, PixelDP's prediction procedure invokes $A_Q(x)$ multiple times with new draws of the noise layer. It then averages the results for each label, thereby computing an estimation $\hat{\mathbb{E}}(A_Q(x))$ of the expected score $\mathbb{E}(A_Q(x))$. It then computes an η-confidence interval for $\hat{\mathbb{E}}(A_Q(x))$ that holds with probability η. Finally, it integrates

this confidence interval into the stability bound for the expectation of a DP computation (Lemma 1) to obtain η-confidence upper and lower bounds on the change an adversary can make to the average score of any label with a p-norm input change of size up to L. Fig. 21b illustrates the upper and lower bounds applied to the average score of each label by the PixelDP prediction procedure. If the lower bound for the label with the top average score is strictly greater than the upper bound for every other label, then, with probability η, the PixelDP network's prediction for input x is *robust* to arbitrary attacks of p-norm size L. The failure probability of this robustness certification, $1 - \eta$, can be made arbitrarily small by increasing the number of invocations of $A_Q(x)$.

One can use PixelDP's certification check in two ways: (1) one can decide only to actuate on predictions that are deemed robust to attacks of a particular size; or (2) one can compute, on a test set, a lower bound of a PixelDP network's accuracy under p-norm bounded attack, independent of how the attack is implemented. This bound, called *certified accuracy*, will hold no matter how effective future generations of the attack are.

The remainder of this section details the noise layer, training, and certified prediction procedures. To simplify notation, we will henceforth use A instead of A_Q.

2.4.1 DP Noise Layer

The noise layer enforces (ϵ, δ)-PixelDP by inserting noise inside the DNN using one of two well-known DP mechanisms: the Laplacian and Gaussian mechanisms. Both rely upon the *sensitivity* of the pre-noise layers (function g). The sensitivity of a function g is defined as the maximum change in output that can be produced by a change in the input, given some distance metrics for the input and output (p-norm and q-norm, respectively):

$$\Delta_{p,q} = \Delta^g_{p,q} = \max_{x,x':x\neq x'} \frac{\|g(x) - g(x')\|_q}{\|x - x'\|_p}.$$

Assuming we can compute the sensitivity of the pre-noise layers (addressed shortly), the noise layer leverages the Laplace and Gaussian mechanisms as follows. On every invocation of the network on an input x (whether for training or prediction) the noise layer computes $g(x)+Z$, where the coordinates $Z = (Z_1, \ldots, Z_m)$ are independent random variables from a noise distribution defined by the function $noise(\Delta, L, \epsilon, \delta)$.

- Laplacian mechanism: $noise(\Delta, L, \epsilon, \delta)$ uses the Laplace distribution with mean zero and standard deviation $\sigma = \sqrt{2}\Delta_{p,1}L/\epsilon$; it gives $(\epsilon, 0)$-DP.
- Gaussian mechanism: $noise(\Delta, L, \epsilon, \delta)$ uses the Gaussian distribution with mean zero and standard deviation $\sigma = \sqrt{2\ln(\frac{1.25}{\delta})}\Delta_{p,2}L/\epsilon$; it gives (ϵ, δ)-DP for $\epsilon \leq 1$.

Here, L denotes the p-norm size of the attack against which the PixelDP network provides (ϵ, δ)-DP; we call it the *construction attack bound*. The noise formulas show that for a fixed noise standard deviation σ, the guarantee degrades gracefully: attacks twice as big halve the ϵ in the DP guarantee ($L \leftarrow 2L \Rightarrow \epsilon \leftarrow 2\epsilon$). This property is often referred as group privacy in the DP literature[61].

Computing the sensitivity of the pre-noise function g depends on where we choose to place the noise layer in the DNN. Because the post-processing property of DP carries the (ϵ, δ)-PixelDP guarantee from the noise layer through the end of the network, a DNN designer has great flexibility in placing the noise layer anywhere in the DNN, as long as no skip connection exists from pre-noise to post-noise layers. We discuss here several options for noise layer placement and how to compute sensitivity for each. Our methods are not closely tied to particular network architectures and can therefore be applied on a wide variety of networks.

Option 1: Noise in the Image. The most straightforward placement of the noise layer is right after the input layer, which is equivalent to adding noise to individual pixels of the image. This case makes sensitivity analysis trivial: g is the identity function, $\Delta_{1,1} = 1$, and $\Delta_{2,2} = 1$.

Option 2: Noise after First Layer. Another option is to place the noise after the first hidden layer, which is usually simple and standard for many DNNs. For example, in image classification, networks often start with a convolution layer. In other cases, DNNs start with fully connected layer. These linear initial layers can be analyzed and their sensitivity computed as follows.

For linear layers, which consist of a linear operator with matrix form $W \in \mathbb{R}^{m,n}$, the sensitivity is the matrix norm, defined as: $\|W\|_{p,q} = \sup_{x:\|x\|_p \leq 1} \|Wx\|_q$. Indeed, the definition and linearity of W directly imply that $\frac{\|Wx\|_q}{\|x\|_p} \leq \|W\|_{p,q}$, which means that: $\Delta_{p,q} = \|W\|_{p,q}$. We use the following matrix norms[196]: $\|W\|_{1,1}$ is the maximum 1-norm of W's columns; $\|W\|_{1,2}$ is the maximum

2-norm of W's columns; and $\|W\|_{2,2}$ is the maximum singular value of W. For ∞-norm attacks, we need to bound $\|W\|_{\infty,1}$ or $\|W\|_{\infty,2}$, as our DP mechanisms require $q \in \{1, 2\}$. However, tight bounds are computationally hard, so we currently use the following bounds: $\sqrt{n}\|W\|_{2,2}$ or $\sqrt{m}\|W\|_{\infty,\infty}$ where $\|W\|_{\infty,\infty}$ is the maximum 1-norm of W's rows. While these bounds are sub-optimal and lead to results that are not as good as for 1-norm or 2-norm attacks, they allow us to include ∞-norm attacks in our frameworks. We leave the study of better approximate bounds to future work.

For a convolution layer, which is linear but usually not expressed in matrix form, we reshape the input (e.g. the image) as an $\mathbb{R}^{nd_{in}}$ vector, where n is the input size (e.g. number of pixels) and d_{in} the number of input channels (e.g. 3 for the RGB channels of an image). We write the convolution as an $\mathbb{R}^{nd_{out} \times nd_{in}}$ matrix where each column has all filter maps corresponding to a given input channel, and zero values. This way, a "column" of a convolution consists of all coefficients in the kernel that correspond to a single input channel. Reshaping the input does not change sensitivity.

Option 3: Noise Deeper in the Network. One can consider adding noise later in the network using the fact that when applying two functions in a row $f_1(f_2(x))$ we have: $\Delta_{p,q}^{(f_1 \circ f_2)} \leq \Delta_{p,r}^{(f_2)} \Delta_{r,q}^{(f_1)}$. For instance, ReLU has a sensitivity of 1 for $p, q \in \{1, 2, \infty\}$, hence a linear layer followed by a ReLU has the same bound on the sensitivity as the linear layer alone. However, we find that this approach for sensitivity analysis is difficult to generalize. Combining bounds in this way leads to looser and looser approximations. Moreover, layers such as batch normalization[86], which are popular in image classification networks, do not appear amenable to such bounds (indeed, they are assumed away by some previous defenses[40]). Thus, our general recommendation is to add the DP noise layer early in the network – where bounding the sensitivity is easy – and taking advantage of DP's post-processing property to carry the sensitivity bound through the end of the network.

Option 4: Noise in Auto-encoder. Pushing this reasoning further, we uncover an interesting placement possibility that underscores the broad applicability and flexibility of our approach: adding noise "before" the DNN in a *separately trained auto-encoder*. An auto-encoder is a special form of DNN trained to predict its own input, essentially learning the identity function $f(x) = x$. Auto-

encoders are typically used to de-noise inputs[188], and are thus a good fit for PixelDP. Given an image dataset, we can train a (ϵ, δ)-PixelDP auto-encoder using the previous noise layer options. We stack it before the predictive DNN doing the classification and fine-tune the predictive DNN by running a few training steps on the combined auto-encoder and DNN. Thanks to the decidedly useful post-processing property of DP, the stacked DNN and auto-encoder are (ϵ, δ)-PixelDP.

This approach has two advantages. First, the auto-encoder can be developed independently of the DNN, separating the concerns of learning a good PixelDP model and a good predictive DNN. Second, PixelDP auto-encoders are much smaller than predictive DNNs, and are thus much faster to train. We leverage this property to train the first certified model for the large ImageNet dataset, using an auto-encoder and the *pre-trained* Inception-v3 model, a substantial relief in terms of experimental work (§2.6.1).

2.4.2 Training Procedure

The soundness of PixelDP's certifications rely only on enforcing DP at prediction time. Theoretically, one could remove the noise layer during training. However, doing so results in near-zero certified accuracy in our experience. Unfortunately, training with noise anywhere except in the image itself raises a new challenge: left unchecked the training procedure will scale up the sensitivity of the pre-noise layers, voiding the DP guarantees.

To avoid this, we alter the pre-noise computation to keep its sensitivity constant (e.g. $\Delta_{p,q} \leq 1$) during training. The specific technique we use depends on the type of sensitivity we need to bound, i.e. on the values of p and q. For $\Delta_{1,1}$, $\Delta_{1,2}$, or $\Delta_{\infty,\infty}$, we normalize the columns, or rows, of linear layers and use the regular optimization process with fixed noise variance. For $\Delta_{2,2}$, we run the projection step described in[40] after each gradient step from the stochastic gradient descent (SGD). This makes the pre-noise layers Parseval tight frames, enforcing $\Delta_{2,2} = 1$. For the pre-noise layers, we thus alternate between an SGD step with fixed noise variance and a projection step. Subsequent layers from the original DNN are left unchanged.

It is important to note that during training, we optimize for *a single draw of noise* to predict the true label for a training example x. We estimate $\mathbb{E}(A(x))$ using multiple draws of noise only

at prediction time. We can interpret this as pushing the DNN to increase the margin between the expected score for the true label versus others. Recall from Equation (2.4) that the bounds on predicted outputs give robustness only when the true label has a large enough margin compared to other labels. By pushing the DNN to give high scores to the true label k at points around x likely under the noise distribution, we increase $\mathbb{E}(A_k(x))$ and decrease $\mathbb{E}(A_{i\neq k}(x))$.

2.4.3 Prediction Procedure

For a given input x, the prediction procedure in a traditional DNN chooses the $\arg\max$ label based on the score vector obtained from a single execution of the DNN's deterministic scoring function, $Q(x)$. In a PixelDP network, the prediction procedure differs in two ways. First, it chooses the $\arg\max$ label based on a Monte Carlo estimation of the expected value of the randomized DNN's scoring function, $\hat{\mathbb{E}}(A(x))$. This estimation is obtained by invoking $A(x)$ multiple times with independent draws in the noise layer. Denote $a_{k,n}(x)$ the n^{th} draw from the distribution of the randomized function A on the k^{th} label, given x (so $a_{k,n}(x) \sim A_k(x)$). In Lemma 1 we replace $\mathbb{E}(A_k(x))$ with $\hat{\mathbb{E}}(A_k(x)) = \frac{1}{n}\sum_n a_{k,n}(x)$, where n is the number of invocations of $A(x)$. We compute η-confidence error bounds to account for the estimation error in our robustness bounds, treating each label's score as a random variable in $[0, 1]$. We use Hoeffding's inequality[81] or Empirical Bernstein bounds[124] to bound the error in $\hat{\mathbb{E}}(A(x))$. We then apply a union bound so that the bounds for each label are all valid together. For instance, using Hoeffding's inequality, with probability η, $\hat{\mathbb{E}}^{lb}(A(x)) \triangleq \hat{\mathbb{E}}(A(x)) - \sqrt{\frac{1}{2n}ln(\frac{2k}{1-\eta})} \leq \mathbb{E}(A(x)) \leq \hat{\mathbb{E}}(A(x)) + \sqrt{\frac{1}{2n}ln(\frac{2k}{1-\eta})} \triangleq \hat{\mathbb{E}}^{ub}(A(x))$.

Second, PixelDP returns not only the prediction for x ($\arg\max(\hat{\mathbb{E}}(A(x)))$) but also a *robustness size certificate* for that prediction, which theory we describe next.

2.5 Theory

2.5.1 Certifying Predictions

To compute the certificate, we extend Proposition 5 to account for the measurement error:

Proposition 6. (Generalized Robustness Condition) Suppose A satisfies (ϵ, δ)-PixelDP with respect to changes of size L in p-norm metric. Using the notation from Proposition 5 further let $\hat{\mathbb{E}}^{ub}(A_i(x))$ and $\hat{\mathbb{E}}^{lb}(A_i(x))$ be the η-confidence upper and lower bound, respectively, for the Monte Carlo estimate $\hat{\mathbb{E}}(A_i(x))$. For any input x, if for some $k \in \mathcal{K}$,

$$\hat{\mathbb{E}}^{lb}(A_k(x)) > e^{2\epsilon} \max_{i:i\neq k} \hat{\mathbb{E}}^{ub}(A_i(x)) + (1 + e^{\epsilon})\delta,$$

then the multiclass classification model based on label probabilities $(\hat{\mathbb{E}}(A_1(x)), \ldots, \hat{\mathbb{E}}(A_K(x)))$ is robust to attacks of p-norm L on input x with probability $\geq \eta$.

Proof. Consider any $\alpha \in B_p(L)$, and let $x' := x + \alpha$. From Equation (2.2), we have with $p > \eta$ that

$$\begin{aligned}
\hat{\mathbb{E}}(A_k(x')) &\geq (\hat{\mathbb{E}}(A_k(x)) - \delta)/e^{\epsilon} \\
&\geq (\hat{\mathbb{E}}^{lb}(A_k(x)) - \delta)/e^{\epsilon}, \\
\hat{\mathbb{E}}(A_{i:i\neq k}(x')) &\leq e^{\epsilon} \max_{i:i\neq k} \hat{\mathbb{E}}^{ub}(A_i(x)) + \delta, \quad i \neq k.
\end{aligned}$$

Starting from the first inequality, and using the hypothesis, followed by the second inequality, we get

$$\begin{aligned}
\hat{\mathbb{E}}^{lb}(A_k(x)) > \; & e^{2\epsilon} \max_{i:i\neq k} \hat{\mathbb{E}}^{ub}(A_i(x)) + (1 + e^{\epsilon})\delta \Rightarrow \\
\hat{\mathbb{E}}(A_k(x')) &\geq (\hat{\mathbb{E}}^{lb}(A_k(x)) - \delta)/e^{\epsilon} \\
&> e^{\epsilon} \max_{i:i\neq k} \hat{\mathbb{E}}^{ub}(A_i(x)) + \delta \\
&> \hat{\mathbb{E}}(A_{i:i\neq k}(x'))
\end{aligned}$$

which is the robustness condition from Equation (2.1). □

Note that the DP bounds are not probabilistic even for $\delta > 0$; the failure probability $1 - \eta$ comes from the Monte Carlo estimate and can be made arbitrarily small with more invocations of $A(x)$.

Thus far, we have described PixelDP certificates as binary with respect to a fixed attack bound, L: we either meet or do not meet a robustness check for L. In fact, our formalism allows for a

more nuanced certificate, which gives the *maximum attack size* L_{max} (measured in p-norm) against which the prediction on input x is guaranteed to be robust: no attack within this size from x will be able to change the highest probability. L_{max} can differ for different inputs. We compute the robustness size certificate for input x as follows. Recall from 2.4.1 that the DP mechanisms have a noise standard deviation σ that grows in $\frac{\Delta_{p,q} L}{\epsilon}$. For a given σ used at prediction time, we solve for the maximum L for which the robustness condition in Proposition 6 checks out:

$L_{max} = \max_{L \in \mathbb{R}^+} L$ such that

$\hat{\mathbb{E}}^{lb}(A_k(x)) > e^{2\epsilon} \hat{\mathbb{E}}^{ub}(A_{i:i \neq k}(x)) + (1 + e^{\epsilon})\delta$ AND either

- $\sigma = \Delta_{p,1} L / \epsilon$ and $\delta = 0$ (for Laplace) OR
- $\sigma = \sqrt{2 \ln(1.25/\delta)} \Delta_{p,2} L / \epsilon$ and $\epsilon \leq 1$ (for Gaussian).

The prediction on x is robust to attacks up to L_{max}, so we award a robustness size certificate of L_{max} for x.

We envision two ways of using robustness size certifications. First, when it makes sense to only take actions on the subset of robust predictions (e.g., a human can intervene for the rest), an application can use PixelDP's certified robustness on each prediction. Second, when all points must be classified, PixelDP gives a lower bound on the accuracy under attack. Like in regular ML, the testing set is used as a proxy for the accuracy on new examples. We can certify the minimum accuracy under attacks up to a threshold size T, that we call the *prediction robustness threshold*. T is an inference-time parameter that can differ from the *construction attack bound* parameter, L, that is used to configure the standard deviation of the DP noise. In this setting the certification is computed only on the testing set, and is not required for each prediction. We only need the highest probability label, which requires fewer noise draws. §2.6.5 shows that in practice a few hundred draws are sufficient to retain a large fraction of the certified predictions, while a few dozen are needed for simple predictions.

2.5.2 Analysis

We make three points about PixelDP's guarantees and applicability. First, we emphasize that our Monte Carlo approximation of the function $x \mapsto \mathbb{E}(A(x))$ is *not* intended to be a DP proce-

dure. Hence, there is no need to apply composition rules from DP, because we do not need this randomized procedure to be DP. Rather, the Monte Carlo approximation $x \mapsto \hat{\mathbb{E}}(A(x))$ is just that: an approximation to a function $x \mapsto \mathbb{E}(A(x))$ whose robustness guarantees come from Lemma 1. The function $x \mapsto \hat{\mathbb{E}}(A(x))$ does not satisfy DP, but because we can control the Monte Carlo estimation error using standard tools from probability theory, it is also robust to small changes in the input, just like $x \mapsto \mathbb{E}(A(x))$.

Second, Proposition 5 is not a high probability result; it is valid with probability 1 even when A is $(\epsilon, \delta > 0)$-DP. The δ parameter can be thought of as a "failure probability" of an (ϵ, δ)-DP mechanism: a chance that a small change in input will cause a big change in the probability of some of its outputs. However, since we know that $A_k(x) \in [0, 1]$, the worst-case impact of such failures on the expectation of the output of the (ϵ, δ)-DP mechanism is *at most* δ, as proven in Lemma 1. Proposition 5 explicitly accounts for this worst-case impact (term $(1 + e^{\epsilon})\delta$ in Equation (2.4)).

Were we able to compute $\mathbb{E}(A(x))$ analytically, PixelDP would output deterministic robustness certificates. In practice however, the exact value is too complex to compute, and hence we approximate it using a Monte Carlo method. This adds probabilistic measurement error bounds, making the final certification (Proposition 6) a high probability result. However, the uncertainty comes exclusively from the Monte Carlo integration – and can be made arbitrarily small with more runs of the PixelDP DNN – and not from the underlying (ϵ, δ)-DP mechanism A. Making the uncertainty small gives an adversary a small chance to fool a PixelDP network into thinking that its prediction is robust when it is not. The only ways an attacker can increase that chance is by either submitting the same attack payload many times or gaining control over PixelDP's source of randomness.

Third, PixelDP applies to any task for which we can measure changes to input in a meaningful p-norm, and bound the sensitivity to such changes at a given layer in the DNN (e.g. sensitivity to a bounded change in a word frequency vector, or a change of class for categorical attributes). PixelDP also applies to other types of optimization problems. It applies to multiclass classification where the prediction procedure returns several top-scoring labels. PixelDP also applies to regression problems (i.e. predicting a real value instead of a category) with three simple steps.

First, if the output is unbounded, one must use $(\epsilon, 0)$-DP (e.g. with the Laplace mechanism). If the output is bounded, one may use (ϵ, δ)-DP. The output may be bounded either naturally, because the specific task has inherent output bounds, or by truncating the results to a large range of values and using a comparatively small δ. Second, instead of estimating the expected value of the randomized prediction function, we estimate both $A_+(x)$ and $A_-(x)$. We can use Hoeffding's inequality[81] or Empirical Bernstein bounds[124] to bound the error. Third, following Lemma 1, we bound $A_+(x)$ and $A_-(x)$ separately using the DP Expected Output Stability Bound, to obtain a bound on $\mathbb{E}(A(x)) = \mathbb{E}(A_+(x)) - \mathbb{E}(A_-(x))$.

2.6 Evaluation

We evaluate PixelDP by answering four key questions:

Q1: How does DP noise affect model accuracy?

Q2: What accuracy can PixelDP certify?

Q3: What is PixelDP's accuracy under attack and how does it compare to that of other best-effort and certified defenses?

Q4: What is PixelDP's computational overhead?

We answer these questions by evaluating PixelDP on five standard image classification datasets and networks – both large and small – and comparing it with one prior certified defense[197] and one best-effort defense[120]. §2.6.1 describes the datasets, prior defenses, and our evaluation methodology; subsequent sections address each question in turn.

Evaluation highlights: PixelDP provides meaningful certified robustness bounds for reasonable degradation in model accuracy on all datasets and DNNs. To the best of our knowledge, these include the first certified bounds for large, complex datasets/networks such as the Inception network on ImageNet and Residual Networks on CIFAR-10. There, PixelDP gives 60% certified accuracy for 2-norm attacks up to 0.1 at the cost of 8.5 and 9.2 percentage-point accuracy degradation respectively. Comparing PixelDP to the prior certified defense on smaller datasets, PixelDP models give higher accuracy on clean examples (e.g., 92.9% vs. 79.6% accuracy SVHN dataset),

Dataset	Image size	Training set size	Testing set size	Target labels	Classifier architecture	Baseline accuracy
ImageNet[46]	299x299x3	1.4M	50K	1000	Inception V3	77.5%
CIFAR-100[103]	32x32x3	50K	10K	100	ResNet	78.6%
CIFAR-10[103]	32x32x3	50K	10K	10	ResNet	95.5%
SVHN[142]	32x32x3	73K	26K	10	ResNet	96.3%
MNIST[206]	28x28x1	60K	10K	10	CNN	99.2%

Tab. 21: **Evaluation datasets and baseline models.** Last column shows the accuracy of the baseline, undefended models. The datasets are sorted based on descending order of scale or complexity.

p-norm used	DP mechanism	Noise location	Sensitivity approach
1-norm	Laplace	1^{st} conv.	$\Delta_{1,1} = 1$
1-norm	Gaussian	1^{st} conv.	$\Delta_{1,2} = 1$
2-norm	Gaussian	1^{st} conv.	$\Delta_{2,2} \leq 1$
1-norm	Laplace	Autoencoder	$\Delta_{1,1} = 1$
2-norm	Gaussian	Autoencoder	$\Delta_{2,2} \leq 1$

Tab. 22: **Noise layers in PixelDP DNNs.** For each DNN, we implement defenses for different attack bound norms and DP mechanisms.

and higher robustness to 2-norm attacks (e.g., 55% vs. 17% accuracy on SVHN for 2-norm attacks of 0.5), thanks to the ability to scale to larger models. Comparing PixelDP to the best-effort defense on larger models and datasets, PixelDP matches its accuracy (e.g., 87% for PixelDP vs. 87.3% on CIFAR-10) and robustness to 2-norm bounded attacks.

2.6.1 Methodology

Datasets. We evaluate PixelDP on image classification tasks from five pubic datasets listed in Table 21. The datasets are listed in descending order of size and complexity for classification tasks. MNIST[206] consists of greyscale handwritten digits and is the easiest to classify. SVHN[142] contains small, real-world digit images cropped from Google Street View photos of house numbers. CIFAR-10 and CIFAR-100[103] consist of small color images that are each centered on one object of one of 10 or 100 classes, respectively. ImageNet[46] is a large, production-scale image dataset with over 1 million images spread across 1,000 classes.

Models: Baselines and PixelDP. We use existing DNN architectures to train a high-performing baseline model for each dataset. Table 21 shows the accuracy of the baseline models. We then make each of these networks PixelDP with regards to 1-norm and 2-norm bounded attacks. We also did rudimentary evaluation of ∞-norm bounded attacks. While the PixelDP formalism can support ∞-norm attacks, our results show that tighter bounds are needed to achieve a practical defense. We leave the development and evaluation of these bounds for future work.

Table 22 shows the PixelDP configurations we used for the 1-norm and 2-norm defenses. The code is available at `https://github.com/columbia/pixeldp`. Since most of this section focuses on models with 2-norm attack bounds, we detail only those configurations here.

ImageNet: We use as baseline a pre-trained version of Inception-v3[179] available in Tensorflow[72]. To make it PixelDP, we use the autoencoder approach from §2.4.1, which does not require a full retraining of Inception and was instrumental in our support of ImageNet. The encoder has three convolutional layers and tied encoder/decoder weights. The convolution kernels are $10 \times 10 \times 32$, $8 \times 8 \times 32$, and $5 \times 5 \times 64$, with stride 2. We make the autoencoder PixelDP by adding the DP noise after the first convolution. We then stack the baseline Inception-v3 on the PixelDP autoencoder and fine-tune it for $20k$ steps, keeping the autoencoder weights constant.

CIFAR-10, *CIFAR-100*, *SVHN*: We use the same baseline architecture, a state-of-the-art Residual Network (ResNet)[208]. Specifically we use the Tensorflow implementation of a 28-10 wide ResNet[182], with the default parameters. To make it PixelDP, we slightly alter the architecture to remove the image standardization step. This step makes sensitivity input dependent, which is harder to deal with in PixelDP. Interestingly, removing this step also increases the baseline's own accuracy for all three datasets. In this section, we therefore report the accuracy of the changed networks as baselines.

MNIST: We train a Convolutional Neural Network (CNN) with two 5×5 convolutions (stride 2, 32 and 64 filters) followed by a 1024 nodes fully connected layer.

Evaluation Metrics. We use two accuracy metrics to evaluate PixelDP models: *conventional accuracy* and *certified accuracy*. Conventional accuracy (or simply accuracy) denotes the fraction of

a testing set on which a model is correct; it is the standard accuracy metric used to evaluate any DNN, defended or not. Certified accuracy denotes the fraction of the testing set on which a certified model's predictions are both *correct* and *certified robust* for a given prediction robustness threshold; it has become a standard metric to evaluate models trained with *certified defenses*[197;161;54]. We also use *precision on certified examples*, which measures the number of correct predictions exclusively on examples that are certified robust for a given prediction robustness threshold. Formally, the metrics are defined as follows:

1. *Conventional accuracy* $\frac{\sum_{i=1}^{n} isCorrect(x_i)}{n}$, where n is the testing set size and $isCorrect(x_i)$ denotes a function returning 1 if the prediction on test sample x_i returns the correct label, and 0 otherwise.
2. *Certified accuracy* $\frac{\sum_{i=1}^{n}(isCorrect(x_i)\&robustSize(scores,\epsilon,\delta,L)\geq T)}{n}$, where $robustSize(scores, \epsilon, \delta, L)$ returns the certified robustness size, which is then compared to the prediction robustness threshold T.
3. *Precision on certified examples* $\frac{\sum_{i=1}^{n}(isCorrect(x_i)\&robustSize(p_i,\epsilon,\delta,L)\geq T))}{\sum_{i=1}^{n} robustSize(p_i,\epsilon,\delta,L)\geq T)}$.

For $T = 0$ all predictions are robust, so certified accuracy is equivalent to conventional accuracy. Each time we report L or T, we use a $[0, 1]$ pixel range.

Attack Methodology. Certified accuracy – as provided by PixelDP and other certified defense – constitutes a guaranteed lower-bound on accuracy under *any* norm-bounded attack. However, the accuracy obtained in practice when faced with a specific attack can be much better. How much better depends on the attack, which we evaluate in two steps. We first perform an attack on 1,000 randomly picked samples (as is customary in defense evaluation[120]) from the testing set. We then measure conventional accuracy on the attacked test examples.

Main 2-norm Attack: For our evaluation, we use the state-of-the art attack from Carlini and Wagner[30], that we run for 9 iterations of binary search, 100 gradient steps without early stopping (which we empirically validated to be sufficient), and learning rate 0.01. We also adapt the attack to our specific defense following[6]: since PixelDP adds noise to the DNN, attacks based

on optimization may fail due to the high variance of gradients, which would not be a sign of the absence of adversarial examples, but merely of them being harder to find. We address this concern by averaging the gradients over 20 noise draws at each gradient step.

We also implemented variants of the iterative Projected Gradient Descent (PGD) attack described in[120], modified to average gradients over 15 noise draws per step, and performing each attack 15 times with a small random initialization. We implemented two version of this PGD attack.

2-norm PGD Attack: The gradients are normalized before applying the step size, to ensure progress even when gradients are close to flat. We perform $k = 100$ gradient steps and select a step size of $\frac{2.5L}{k}$. This heuristic ensures that all feasible points within the 2-norm ball can be reached after k steps. After each step, if the attack is larger than L, we project it on the 2-norm ball by normalizing it. Under this attack, results were qualitatively identical for all experiments. The raw accuracy numbers were a few percentage points higher (i.e. the attack was slightly less efficient), so we kept the results for the Carlini and Wagner attack.

∞-norm PGD Attack: We perform $max(L + 8, 1.5L)$ gradient steps and maintain a constant step of size of 0.003 (which corresponds to the minimum pixel increment in a discrete $[0, 255]$ pixel range). At the end of each gradient step we clip the size of the perturbation to enforce a perturbation within the ∞-norm ball of the given attack size. We used this attack to compare PixelDP with models from Madry and RobustOpt.

Finally, we performed sanity checks suggested in[6]. The authors observe that several heuristic defenses do not ensure the absence of adversarial examples, but merely make them harder to find by obfuscating gradients. This phenomenon, also referred to as gradient masking[149;185], makes the defense susceptible to new attacks crafted to circumvent that obfuscation[6]. Although PixelDP provides certified accuracy bounds that are *guaranteed* to hold regardless of the attack used, we followed guidelines from[6], to to rule out obfuscated gradients in our empirical results. We verified three properties that can be symptomatic of problematic attack behavior. First, when growing T, the accuracy drops to 0 on all models and datasets. Second, our attack significantly outperforms

Dataset	Baseline	$L = 0.03$	$L = 0.1$	$L = 0.3$	$L = 1.0$
ImageNet	77.5%	–	68.3%	57.7%	37.7%
CIFAR-10	95.5%	93.3%	87.0%	70.9%	44.3%
CIFAR-100	78.6%	73.4%	62.4%	44.3%	22.1%
SVHN	96.3%	96.1%	93.1%	79.6%	28.2%
MNIST	99.2%	99.1%	99.1%	98.2%	11%

Tab. 23: **Impact of PixelDP noise on conventional accuracy.** For each DNN, we show different levels of construction attack size L. Conventional accuracy degrades with noise level.

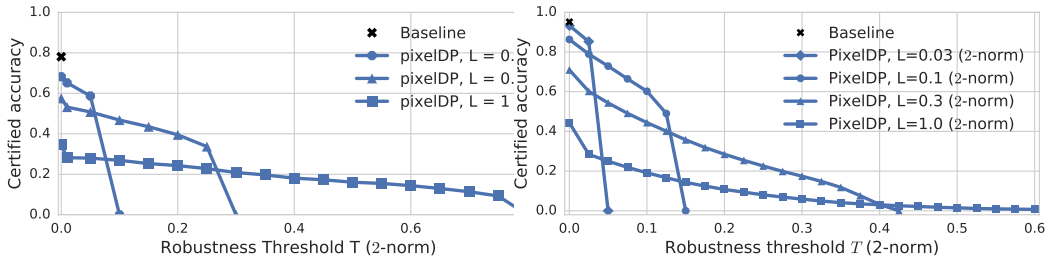
random sampling. Third, our iterative attack is more powerful than the respective single-step attack.

Prior Defenses for Comparison. We use two state-of-art defenses as comparisons. First, we use the empirical defense model provided by the Madry Lab for CIFAR-10[121]. This model is developed in the context of ∞-norm attacks. It uses an adversarial training strategy to approximately minimize the worst case error under malicious samples[120]. While inspired by robust optmization theory, this methodology is best effort (see §2.7) and supports no formal notion of robustness for individual predictions, as we do in PixelDP. However, the Madry model performs better under the latest attacks than other best-effort defenses (it is in fact the only one not yet broken)[6], and represents a good comparison point.

Second, we compare with another approach for certified robustness against ∞-norm attacks[197], based on robust optimization. This method does not yet scale to the largest datasets (e.g. ImageNet), or the more complex DNNs (e.g. ResNet, Inception) both for computational reasons and because not all necessary layers are yet supported (e.g. BatchNorm). We thus use their largest released model/dataset, namely a CNN with two convolutions and a 100 nodes fully connected layer for the SVHN dataset, and compare their robustness guarantees with our own networks' robustness guarantees. We call this SVHN CNN model *RobustOpt*.

2.6.2 Impact of Noise (Q1)

Q1: How does DP noise affect the conventional accuracy of our models? To answer, for each dataset we train up to four $(1.0, 0.05)$-PixelDP DNN, for construction attack bound $L \in \{0.03, 0.1, 0.3, 1\}$. Higher values of L correspond to robustness against larger attacks and larger noise standard deviation σ.



(a) ImageNet Certified Accuracy (b) CIFAR-10 Certified Accuracy

Fig. 22: **Certified accuracy, varying the construction attack bound (L) and prediction robustness threshold (T)**, on ImageNet auto-encoder/Inception and CIFAR-10 ResNet, 2-norm bounds. Robust accuracy at high Robustness thresholds (high T) increases with high-noise networks (high L). Low noise networks are both more accurate and more certifiably robust for low T.

Table 23 shows the conventional accuracy of these networks and highlights two parts of an answer to Q1. First, at fairly low but meaningful construction attack bound (e.g., $L = 0.1$), all of our DNNs exhibit reasonable accuracy loss – *even on ImageNet*, a dataset on which no guarantees have been made to date! ImageNet: The Inception-v3 model stacked on the PixelDP auto-encoder has an accuracy of 68.3% for $L = 0.1$, which is reasonable degradation compared to the baseline of 77.5% for the unprotected network. CIFAR-10: Accuracy goes from 95.5% without defense to 87% with the $L = 0.1$ defense. For comparison, the Madry model has an accuracy of 87.3% on CIFAR-10. SVHN: our $L = 0.1$ PixelDP network achieves 93.1% conventional accuracy, down from 96.3% for the unprotected network. For comparison, the $L = 0.1$ RobustOpt network has an accuracy of 79.6%, although they use a smaller DNN due to the computationally intensive method.

Second, as expected, constructing the network for larger attacks (higher L) progressively degrades accuracy. ImageNet: Increasing L to 0.3 and then 1.0 drops the accuracy to 57.7% and 37.7%, respectively. CIFAR-10: The ResNet with the least noise ($L = 0.03$) reaches 93.3% accuracy, close to the baseline of 95.5%; increasing noise levels ($L = (0.1, 0.3, 1.0)$) yields 87%, 70.9%, and 37.7%, respectively. Yet, as shown in §2.6.4, PixelDP networks trained with fairly low L values (such as $L = 0.1$) already provide meaningful empirical protection against larger attacks.

2.6.3 Certified Accuracy (Q2)

Q2: What accuracy can PixelDP certify on a test set? Fig. 22 shows the certified robust accuracy bounds for ImageNet and CIFAR-10 models, trained with various values of the construction

attack bound L. The certified accuracy is shown as a function of the prediction robustness threshold, T. We make two observations. First, PixelDP yields meaningful robust accuracy bounds even on large networks for ImageNet (see Fig. 22a), attesting the scalability of our approach. The $L = 0.1$ network has a certified accuracy of 59% for attacks smaller than 0.09 in 2-norm. The $L = 0.3$ network has a certified accuracy of 40% to attacks up to size 0.2. To our knowledge, PixelDP is the first defense to yield DNNs with certified bounds on accuracy under 2-norm attacks on datasets of ImageNet's size and for large networks like Inception.

Second, PixelDP networks constructed for larger attacks (higher L, hence higher noise) tend to yield higher certified accuracy for high thresholds T. For example, the ResNet on CIFAR-10 (see Fig. 22b) constructed with $L = 0.03$ has the highest robust accuracy up to $T = 0.03$, but the ResNet constructed with $L = 0.1$ becomes better past that threshold. Similarly, the $L = 0.3$ ResNet has higher robust accuracy than the $L = 0.1$ ResNet above the 0.14 2-norm prediction robustness threshold.

We ran the same experiments on SVHN, CIFAR-100 and MNIST models but omit the graphs for space reasons. Our main conclusion – that adding more noise (higher L) hurts both conventional and low T certified accuracy, but enhances the quality of its high T predictions – holds in all cases.

As far as ∞-norm attacks are concerned, we acknowledge that the size of the attacks against which our current PixelDP defense can certify accuracy is substantially lower than that of previous certified defenses. Although previous defenses have been demonstrated on MNIST and SVHN only, and for smaller DNNs, they achieve ∞-norm defenses of $T_\infty = 0.1$ with robust accuracy 91.6%[197] and 65%[161] on MNIST. On SVHN,[197] uses $T_\infty = 0.01$, achieving 59.3% of certified accuracy. Using the crude bounds we have between p-norms makes a comparison difficult in both directions. Mapping ∞-norm bounds in 2-norm gives $T_2 \geq T_\infty$, also yielding very small bounds. On the other hand, translating 2-norm guarantees into ∞-norm ones (using that $\|x\|_2 \leq \sqrt{n}\|x\|_\infty$ with n the size of the image) would require a 2-norm defense of size $T_2 = 2.8$ to match the $T_\infty = 0.1$ bound from MNIST, an order of magnitude higher than what we can achieve. As

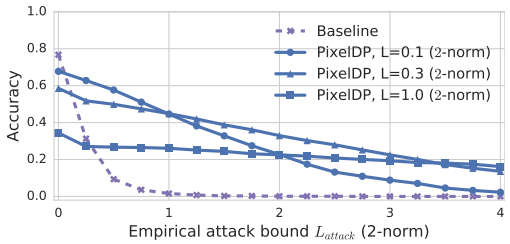


Fig. 23: **Accuracy under attack on ImageNet.** For the ImageNet auto-encoder plus Inception-v3, $L \in \{0.1, 0.3, 1.0\}$ 2-norm attacks. The PixelDP auto-encoder increases the robustness of Inception against 2-norm attacks.

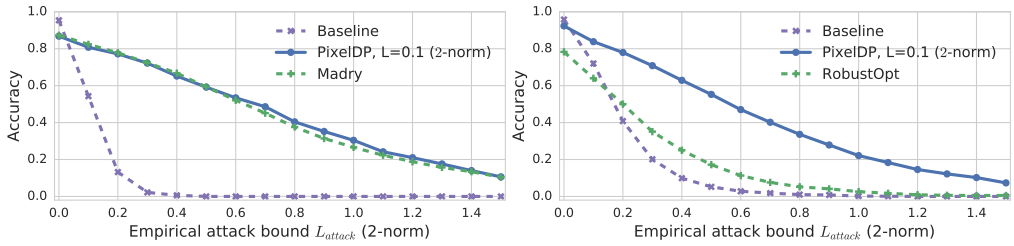
comparison points, our $L = 0.3$ CNN has a robust accuracy of 91.6% at $T = 0.19$ and 65% at $T = 0.39$. We make the same observation on SVHN, where we would need a bound at $T_2 = 0.56$ to match the $T_\infty = 0.01$ bound, but our ResNet with $L = 0.1$ reaches a similar robust accuracy as RobustOpt for $T_2 = 0.1$. This calls for the design ∞-norm specific PixelDP mechanisms that could also scale to larger DNNs and datasets.

While PixelDP does not yet yield strong ∞-norm bounds, it provides meaningful certified accuracy bounds for 2-norm attacks, including on much larger and more complex datasets and networks than those supported by previous approaches.

2.6.4 Accuracy Under Attack (Q3)

A standard method to evaluate the strength of a defense is to measure the conventional accuracy of a defended model on malicious samples obtained by running a state-of-the-art attack against samples in a held-out testing set[120]. We apply this method to answer three aspects of question *Q3: (1) Can PixelDP help defend complex models on large datasets in practice? (2) How does PixelDP's accuracy under attack compare to state-of-the-art defenses? (3) How does the accuracy under attack change for certified predictions?*

Accuracy under Attack on ImageNet. We first study conventional accuracy under attack for PixelDP models on ImageNet. Fig. 23 shows this metric for 2-norm attacks on the baseline Inception-v3 model, as well as three defended versions, with a stacked PixelDP auto-encoder trained with construction attack bound $L \in \{0.1, 0.3, 1.0\}$. PixelDP makes the model significantly more robust to attacks. For attacks of size $L_{attack} = 0.5$, the baseline model's accuracy drops to 11%, whereas the $L = 0.1$ PixelDP model's accuracy remains above 60%. At $L_{attack} = 1.5$, the baseline model



(a) CIFAR-10 (b) SVHN

Fig. 24: **Accuracy under 2-norm attack for PixelDP vs. Madry and RobustOpt**, CIFAR-10 and SVHN. For 2-norm attacks, PixelDP is on par with Madry until $L_{attack} \geq 1.2$; RobustOpt support only small models, and has lower accuracy.

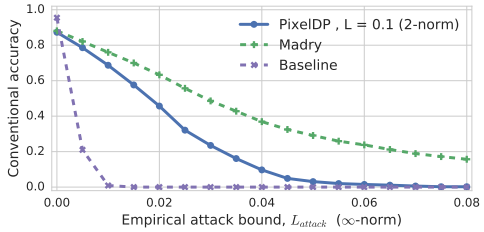


Fig. 25: **Accuracy under ∞-norm attacks for PixelDP and Madry.** The Madry model, explicitly trained against ∞-norm attacks, outperforms PixelDP. The difference increases with the size of the attack.

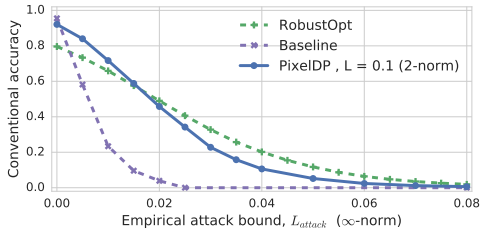


Fig. 26: **Accuracy under ∞-norm attacks for PixelDP and RobustOpt.** PixelDP is better up to $L_\infty = 0.015$, due to its support of larger ResNet models. For attacks of ∞-norm above this value, RobustOpt is more robust.

has an accuracy of 0, but the $L = 0.1$ PixelDP is still at 30%, while the $L = 0.3$ PixelDP model have more that 39% accuracy.

Accuracy under Attack Compared to Madry. Fig. 24a compares conventional accuracy of a PixelDP model to that of a Madry model on CIFAR-10, as the empirical attack bound increases for 2-norm attacks. For 2-norm attacks, our model achieves conventional accuracy on par with, or slightly higher than, that of the Madry model. Both models are dramatically more robust under this attack compared to the baseline (undefended) model. For ∞-norm attacks our model does not fare as well, which is expected as the PixelDP model is trained to defend against 2-norm attacks,

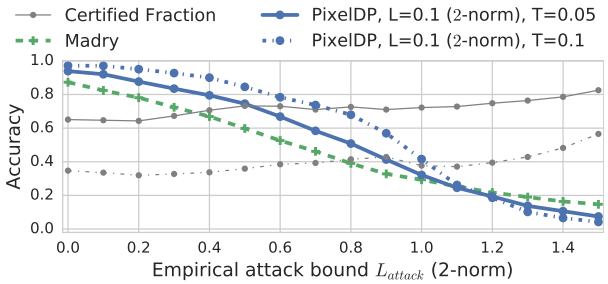


Fig. 27: **PixelDP certified predictions vs. Madry accuracy, under attack**, CIFAR-10 ResNets, 2-norm attack. PixelDP makes fewer but more correct predictions up to $L_{attack} = 1.0$.

while the Madry model is optimized for ∞-norm attacks. For $L_{attack} = 0.01$, PixelDP's accuracy is 69%, 8 percentage points lower than Madry's. The gap increases until PixelDP arrives at 0 accuracy for $L_{attack} = 0.06$, with Madry still having 22%.

Figure 25 shows PixelDP's accuracy under ∞-norm attacks compared to the Madry model, trained specifically against this type of attacks. On CIFAR-10, the Madry model outperforms PixelDP: for $L_{attack} = 0.01$, PixelDP's accuracy is 69%, 8 percentage points lower than Madry's. The gap increases until PixelDP arrives at 0 accuracy for $L_{attack} = 0.06$, with Madry still having 22%.

Accuracy under Attack Compared to RobustOpt. Fig. 24b shows a similar comparison with the RobustOpt defense[197], which provides certified accuracy bounds for ∞-norm attacks. We use the SVHN dataset for the comparison as the RobustOpt defense has not yet been applied to larger datasets. Due to our support of larger DNN (ResNet), PixelDP starts with higher accuracy, which it maintains under 2-norm attacks. For attacks of $L_{attack} = 0.5$, RobustOpt is bellow 20% accuracy, and PixelDP above 55%. Under ∞-norm attacks, the behavior is different: PixelDP has the advantage up to $L_{attack} = 0.015$ (58.8% to 57.1%), and RobustOpt is better thereafter. For instance, at $L_{attack} = 0.03$, PixelDP has 22.8% accuracy, to RobustOpt's 32.7%.

On Figure 26 we show PixelDP's accuracy under ∞-norm attacks compared to the RobustOpt model, trained specifically against this type of attacks. On SVHN, against the RobustOpt model, trained with robust optimization against ∞-norm attacks, PixelDP is better up to $L_\infty = 0.015$, due to its support of larger ResNet models. For attacks of ∞-norm above this value, RobustOpt is more robust.

Precision on Certified Predictions Under Attack. Another interesting feature of PixelDP is its ability to make certifiably robust predictions. We compute the accuracy of these certified predictions under attack – which we term *robust precision* – and compare them to predictions of the Madry network that do not provide such a certification. Fig. 27 shows the results of considering only predictions with a certified robustness above 0.05 and 0.1. It reflects the benefit to be gained by applications that can leverage our theoretical guarantees to filter out non-robust predictions. We observe that PixelDP's robust predictions are *substantially more correct* than Madry's predictions up to an empirical attack bound of 1.1. For $T = 0.05$ PixelDP's robust predictions are 93.9% accurate, and up to 10 percentage points more correct under attack for $L_{attack} \leq 1.1$. A robust prediction is given for above 60% of the data points. The more conservative the robustness test is (higher T), the more correct PixelDP's predictions are, although it makes fewer of them (Certified Fraction lines).

Thus, for applications that can afford to not act on a minority of the predictions, PixelDP's robust predictions under 2-norm attack are substantially more precise than Madry's. For applications that need to act on every prediction, PixelDP offers on-par accuracy under 2-norm attack to Madry's. Interestingly, although our defense is trained for 2-norm attacks, the first conclusion still holds for ∞-norm attacks; the second (as we saw) does not.

2.6.5 Computational Overhead (Q4)

Q4: What is PixelDP's computational overhead? We evaluate overheads for training and prediction. PixelDP adds little overhead for *training*, as the only additions are a random noise tensor and sensitivity computations. On our GPU, the CIFAR-10 ResNet baseline takes on average $0.65s$ per training step. PixelDP versions take at most $0.66s$ per training step (1.5% overhead). This represents a significant benefit over adversarial training (e.g. Madry) that requires finding good adversarial attacks for each image in the mini-batch at each gradient step, and over robust optimization (e.g. RobustOpt) that requires solving a constrained optimization problem at each gradient step. The low training overhead is instrumental to our support of large models and datasets.

PixelDP impacts *prediction* more substantially, since it uses multiple noise draws to estimate the label scores. Making a prediction for a single image with 1 noise draw takes $0.01s$ on average. Making 10 draws brings it only to $0.02s$, but 100 requires $0.13s$, and 1000, $1.23s$. It is possible to use Hoeffding's inequality[81] to bound the number of draws necessary to distinguish the highest score with probability at least η, given the difference between the top two scores $y_{max} - y_{second-max}$. Empirically, we found that 300 draws were typically necessary to properly certify a prediction, implying a prediction time of $0.42s$ seconds, a $42\times$ overhead. This is parallelizable, but resource consumption is still substantial. To make simple predictions – distinguish the top label when we must make a prediction on all inputs – 25 draws are enough in practice, reducing the overhead to $3\times$.

2.7 Related Work

Our work relates to a significant body of work in adversarial examples and beyond. Our main contribution to this space is to introduce a new and very different direction for building *certified defenses*. Previous attempts have built on robust optimization theory. In PixelDP we propose a new approach built on differential privacy theory which exhibits a level of flexibility, broad applicability, and scalability that exceeds what robust optimization-based certified defenses have demonstrated. While the most promising way to defend against adversarial examples is still an open question, we observe undebatable benefits unique to our DP based approach, such as the post-processing guarantee of our defense. In particular, the ability to prepend a defense to unmodified networks via a PixelDP auto-encoder, as we did to defend Inception with *no structural changes*, is unique among certified (and best-effort) defenses.

Best-effort Defenses. Defenders have used multiple heuristics to empirically increase DNNs' robustness. These defenses include model distillation[148], automated detection of adversarial examples[80;133;132], application of various input transformations[87;38], randomization[75;39], and generative models[157;85;205]. Most of these defenses have been broken, sometimes months after their publication[30;29;6].

The main empirical defense that still holds is Madry et al.[120], based on adversarial training[69]. Madry et al. motivate their approach with robust optimization, a rigorous theory. However not all the assumptions are met, as this approach runs a best-effort attack on each image in the minibatch at each gradient step, when the theory requires finding the best possible adversarial attack. And indeed, finding this worst case adversarial example for ReLU DNNs, used in[120], was proven to be NP-hard in[174]. Therefore, while this defense works well in practice, it gives no theoretical guarantees for individual predictions or for the model's accuracy under attack. PixelDP leverages DP theory to provide guarantees of robustness to arbitrary, norm-based attacks for individual predictions.

Randomization-based defenses are closest in method to our work[75;39;118]. For example, Liu et al.[118] randomizes the entire DNN and predicts using an ensemble of multiple copies of the DNN, essentially using draws to roughly estimate the expected $\arg\max$ prediction. They observe empirically that randomization smoothens the prediction function, improving robustness to adversarial examples. However, randomization-based prior work provides limited formalism that is insufficient to answer important defense design questions: where to add noise, in what quantities, and what formal guarantees can be obtained from randomization? The lack of formalism has caused some works[75;39] to add insufficient amounts of noise (e.g., noise not calibrated to pre-noise sensitivity), which makes them vulnerable to attack[29]. On the contrary,[118] inserts randomness into every layer of the DNN: our work shows that adding the right amount of calibrated noise at a single layer is sufficient to leverage DP's post-processing guarantee and carry the bounds through the end of the network. Our work formalizes randomization-based defenses using DP theory, and in doing so helps answer many of these design questions. Our formalism also lets us reason about the guarantees obtained through randomization and enables us to elevate randomization-based approaches from the class of best-effort defenses to that of *certified defenses*.

Certified Defenses and Robustness Evaluations. PixelDP offers two functions: (1) a strategy for learning robust models and (2) a method for evaluating the robustness of these models against adversarial examples. Both of these approaches have been explored in the literature. First, several

certified defenses modify the neural network training process to minimize the number of robustness violations[197;161;40]. These approaches, though promising, do not yet scale to larger networks like Google Inception[197;161]. In fact, all published certified defenses have been evaluated on small models and datasets[197;161;40;138], and at least in one case, the authors directly acknowledge that some components of their defense would be "completely infeasible" on ImageNet[197]. A recent paper[54] presents a certified defense evaluated on the CIFAR-10 dataset[103] for multi-layer DNNs (but smaller than ResNets). Their approach is completely different from ours and, based on the current results we see no evidence that it can readily scale to large datasets like ImageNet.

Another approach[174] combines robust optimization and adversarial training in a way that gives formal guarantees and has lower computational complexity than previous robust optimization work, hence it has the potential to scale better. This approach requires smooth DNNs (e.g., no ReLU or max pooling) and robustness guarantees are over the expected loss (e.g., log loss), whereas PixelDP can certify each specific prediction, and also provides intuitive metrics like robust accuracy, which is not supported by[174]. Finally, unlike PixelDP, which we evaluated on five datasets of increasing size and complexity, this technique was evaluated only on MNIST, a small dataset that is notoriously amenable to robust optimization (due to being almost black and white). Since the effectiveness of all defenses depends on the model and dataset, it is hard to conclude anything about how well it will work on more complex datasets.

Second, several works seek to formally verify[84;98;190;191;53;68;184] or lower bound[153;195] the robustness of pre-trained ML models against adversarial attacks. Some of these works scale to large networks[153;195], but they are insufficient from a defense perspective as they provide no scalable way to train robust models.

Differentially Private ML. Significant work focuses on making ML algorithms DP to preserve the privacy of training sets[129;1;35]. PixelDP is orthogonal to these works, differing in goals, semantic, and algorithms. The only thing we share with DP ML (and most other applied DP literature) are DP theory and mechanisms. The goal of DP ML is to learn the parameters of a model while ensuring DP with respect to the training data. Public release of model parameters trained using a DP

learning algorithm (such as DP empirical risk minimization or ERM) is guaranteed to not reveal much information about individual training examples. PixelDP's goal is to create a robust predictive model where a small change to any input example does not drastically change the model's prediction on that example. We achieve this by ensuring that the model's scoring function is a DP function with respect to the features of an input example (eg, pixels). DP ML algorithms (e.g., DP ERM) do not necessarily produce models that satisfy PixelDP's semantic, and our training algorithm for producing PixelDP models does not ensure DP of training data.

Previous DP-Robustness Connections. Previous work studies generalization properties of DP[12]. It is shown that *learning algorithms* that satisfy DP with respect to the training data have statistical benefits in terms of out-of-sample performance; or that DP has a deep connection to robustness at the dataset level[47;57]. Our work is rather different. Our learning algorithm is not DP; rather, the predictor we learn satisfies DP with respect to the atomic units (e.g., pixels) of a given test point.

2.8 Summary

We demonstrated a connection between robustness against adversarial examples and differential privacy theory. We showed how the connection can be leveraged to develop a certified defense against such attacks that is (1) as effective at defending against 2-norm attacks as today's state-of-the-art best-effort defense and (2) more scalable and broadly applicable to large networks compared to any prior certified defense. Finally, we presented the first evaluation of a certified 2-norm defense on the large-scale ImageNet dataset. In addition to offering encouraging results, the evaluation highlighted the substantial flexibility of our approach by leveraging a convenient autoencoder-based architecture to make the experiments possible with limited resources.

Since we first published this work[106], multiple other groups have built, most notably to improve the robustness bounds[114;41] or the training methodology[168], yielding new state of the art theoretical and empirical robustness guarantees. The approach described in this chapter thus seems promising as a defense against adversarial examples. We believe that a third avenue for improvement that hasn't been explored in depth yet is to design DNNs specifically for PixelDP. In our

opinion, a particularly promising approach is to perform multiple queries and leverage composition results from DP, maybe using recurrent neural networks or attention mechanisms.

Chapter 3

Data Use Transparency in Opaque Targeting Services

The last challenge triggered by the switch to ML that we described is the application of opaque decisions at large scale. This chapter addresses a particular facet of this challenge, namely the lack of transparency in data use for large scale targeting and personalization in web services. We present tools that provide a new visibility into how ML-driven web services use end-users' data, as well as a measurement study that reveals violations of the servcies' own privacy policies. This chapter is based in large parts on our Sunlight paper[110]. It also includes work we published in a prior paper, xRay[108], in particular on algorithms to detect targeting and an empirical evalution of the scalability of such algorithms.

3.1 Motivation

In a 1913 paper[24], Louis Brandeis, the proponent of modern views of individual rights to privacy, stated: *"Sunlight is said to be the best of disinfectants; electric light the most efficient policeman."* Unfortunately, today's Web is a very dark and complex ecosystem driven to a large extent by the massive collection and monetization of personal data. Myriad of Web services, mobile applications, and third parties are collecting large amounts of information from our daily online interactions, such as our website visits, clicks, emails, documents, and pictures. At a surface level, end-users and researchers alike largely understand that these companies may be using this information to target advertisements, customize recommendations, personalize news feeds, and even fine-tune prices. Indeed, the companies' own terms of service often stipulate such uses. But at a concrete level, neither end-users nor researchers – nor, as we argue, individual companies – understand how specific personal data flows through the complex web ecosystem, how it is being used (or abused) in practice by parties that interact with it, and how those uses affect the users.

Questions about the targeting on the web abound: Are our children's online activities being targeted, and if so what kinds of products are they being offered? Are people being targeted because their browsing patterns suggest that they might be vulnerable (e.g., sick, depressed, or in financial difficulty)? Are such inferences being used to increase insurance premiums, deny housing, or place potentially damaging products, such as alcoholic products or risky mortgage

deals? In other words, is our data being used without our knowledge or consent in ways that affect us? Today, we lack believable, at-scale answers to such questions.

A good way to shed light on large, complex systems is to measure them at scale using scientific methods and tools. Indeed, a number of measurement studies have attempted to answer questions about how personal data is being used on the web[76;202;135;136;189;77;11;117;45;141;22]. We reviewed 12 of these studies and found a significant gap in scalable experimental methodologies. Generally speaking, prior studies conduct tightly controlled experiments that vary personal data inputs (such as location, search terms, or profile interests) *one at a time* and observe the effect on service outputs (such as ads, recommendations, or prices) compared to a control group. Unfortunately, we find the methodologies employed by prior studies either difficult to scale or lacking formal notions of confidence, which make their results difficult to trust, interpret, and generalize. Consequently, the scale and scope of what we can assert today about data use on the web is limited, and our capacity to exert oversight on this large, complex, and ever-changing ecosystem is virtually non-existent.

In this chapter, we argue that shedding light into the web's complex data ecosystem requires the development of robust experimental methodologies, as well as infrastructures that implement them, that can be used to answer broad classes of questions at large scale and with interpretable, trustworthy, statistical justification of the results. We present *Sunlight*, a new methodology, plus a system that implements it, that achieves these goals in the context of one important class of questions: those that require a fine-grained measurement of the causes of targeting phenomena on the web. All the questions raised at the start of this section can be addressed with Sunlight.

3.2 The Sunlight System

The Sunlight methodology builds upon robust statistical methods to support scalable, trustworthy, and interpretable results. Our key innovation is to formally separate various operations into multiple interacting stages organized in a pipeline and identifying the right building blocks from statistics and machine learning to leverage at each stage of the pipeline. The Sunlight pipeline analyzes the data collected from an experiment that tries many different inputs *at once*, placing them at random in a small number of user accounts (logarithmic in the number of inputs) and col-

lecting outputs from each account. The Sunlight pipeline analyzes the data to reveal which specific input likely caused which output. The first stage, *scalable hypothesis generation*, creates a set of plausible targeting hypotheses regarding which specific inputs correlate with which outputs. It leverages sparsity properties to support the *simultaneous* estimation of the effect of multiple inputs on the outputs, a consequence of the same phenomenon that underlies compressed sensing[49]. If needed, the second stage, *interpretable hypothesis formation*, converts the targeting hypotheses to an interpretable form that Sunlight users (such as auditors or researchers) can readily understand. The third stage, *hypothesis testing*, establishes the statistical significance of the interpretable, plausible targeting hypotheses by testing their veracity in a separate, testing dataset initially carved out from the collected data but never used until this stage. In some circumstances, specialized tests can establish causation and not just correlation between the inputs and the outputs. Finally, the fourth stage, *multiple testing correction*, accounts for the testing of many hypotheses on the same dataset, which increases the chance of any individual hypothesis being wrong. The end result are validated, interpretable hypotheses about which inputs are targeted by each output, along with a statistical significance score (a *p-value*) for each hypothesis.

Sunlight implements this methodology in a modular way, which supports both the instantiation and the evaluation of each stage based on multiple building blocks from statistics and machine learning. We find that different mechanisms lead to different trade-offs between the scale of and the confidence in the results, hence Sunlight lets its users choose end-to-end pipelines that best fit their needs. Development of effective such pipelines from existing building blocks is surprisingly challenging, as different mechanisms interact in unexpected ways in the pipeline. For example, our detailed evaluation of various Sunlight pipelines reveals counterintuitive inversions of recall near the start of the pipeline and at the end. Indeed, substituting a mechanism for generating hypotheses in Stage 1 with one that has higher recall but lower precision, may ultimately lower the recall at the end of the pipeline. The reason is that the multiple testing correction at Stage 4 tends to favor those mechanisms that generate fewer but more accurate hypotheses.

This chapter discusses and evaluates the inherent trade-offs in web transparency measurement designs, bringing the following contributions to this emerging research topic:

1. A review of 12 recent articles on web transparency measurement and tools, which highlights the need for new, principled methodologies for scalable and trustworthy web transparency measurements. (§3.3.2)
2. The first methodology for detecting targeting in large-scale experiments with interpretable and statistically justifiable results. While our methodology focuses on our specific problem – fine-grained targeting detection – we believe that its conceptual bearings are relevant to other web transparency problems (e.g., price discrimination studies at scale). (§3.4)
3. The first system that implements this methodology to detect targeting at fine granularity, at scale, and with solid statistical justification of its results. Sunlight is modular, allows broad design space explorations, and customization of its pipeline to strike varied trade-offs of confidence and scale. (§3.5)
4. A detailed evaluation of Sunlight with comparisons of multiple design options and prior art. Our evaluation methodology is new in itself, and (we believe) a useful starting point for future transparency infrastructures, an area that currently lacks rigorous evaluations. Our results reveal a trade-off between the statistical confidence and number of targeting hypotheses that can be made. They also show that favoring high precision algorithms can yield better recall at high confidence, and that scaling output numbers may require to accept lower statistical guarantees to find sufficient hypotheses. (§3.6)
5. Results from analyzing targeting of tens of thousands of ads in two ecosystems: Gmail and the broader Web. Results reveal a large and diverse collection of ads targeting websites across many categories, including ads that appear to contradict explicit statements made by Google about targeting on sensitive topics, as well as advertising network policies about ads facilitating recreational drug use. (§3.6.10 and §3.6.10).
6. Sunlight's source code and datasets. (`https://columbia.github.io/sunlight/`)

3.3 Background

At an abstract level, our work is motivated by our desire to understand how to build principled, scalable infrastructures that can bring visibility to today's dark data-driven web. Such infrastructures must be able to detect data flows at great scale and in complex, heterogeneous environments, and provide trustworthy assessments about these data flows. We believe there is urgent need for such infrastructures (which we term generically *web transparency infrastructures*), yet we find limited progress in the related literature.

At a concrete level, we describe our experience building one such scalable and trustworthy[1] infrastructure, *Sunlight*, which aims to discover data flows in a specific context: detecting the causes of targeting phenomena at fine granularity from controlled experiments with differentiated inputs. We next motivate the need for targeting detection systems in particular, before motivating the broader need for scale and confidence in web transparency infrastructures.

3.3.1 The Targeting Detection Problem

Targeting is a pervasive phenomenon on the web and involves the use of a user's personal data (*inputs*) to tailor some content (*output*), such as an ad, a recommendation, a price, search results, or news. Sunlight aims to identify the likely causes of each targeted output in the context of controlled experiments that test many inputs at once. Numerous use cases exist that could leverage such functionality. For example, researchers could use it to study targeting at larger scale than was possible before. We provide results from our own case studies of ad targeting in Gmail and on the web in §3.6.10. Following are two other example use cases that broaden the scope and help underscore Sunlight's design requirements.

Example 1: Ann, a federal trade commission researcher specializing in COPPA enforcement, plans to investigate whether and how advertisers target children. She hypothesizes that advertisers leverage information amassed by web trackers to bid for users with browsing histories characteristic of children. Ann wants to run a *large-scale study* to both quantify the amount of children-

[1] In this chapter, the term *trustworthy* refers strictly to the level of confidence (in a statistical sense) one can have in the results of an investigation assuming the non-malicious service model in §3.4.3.

oriented targeting, and find specific instances of what might be deemed as inappropriate or illegal targeting (e.g., targeting pornographic movies at teenagers or promoting unhealthy eating habits to young children). The number of websites dedicated to children is large, and there are even more neutral websites frequented by both children and adults on which targeted ads might appear. Ann fully expects that child-based targeting will be rare events, hence running her experiment at large scale is vital. For any case of inappropriate or illegal targeting, Ann plans to investigate through legal means (e.g., interview the advertiser) to determine whether the targeting was intentional or purely algorithmic. Such investigations are expensive, so Ann requires *high confidence* in an experimental finding to justify her investigative effort.

Example 2: Bob, a tech-savvy investigative journalist, wishes to investigate how coupons are targeted at users. Coupon services aim to provide product discounts to users who are likely to be interested in a particular product but may need some incentive to do so. A wide variety of types of information could feed into the targeting decision, including web history, tweets, and Facebook likes. Bob would like to try many different activities and see which ones are targeted by coupons. In addition to requiring high confidence in the results, Bob needs the results to also be easily *interpretable* so that he and his readers can understand and reason about the implications of the targeting. Ideally, whatever statements he makes in his articles should be directly validated on the datasets and have an associated confidence level that he can understand and potentially communicate to his audience. For example, he imagines statements such as the following to be appropriate for communication with his readers: *"In our experiments, profiles that tweeted about weight loss or diets were much more likely to be offered McDonald's coupons than those without such tweets. This result was very unlikely (0.01% chance) to have been observed if such tweets were not targeted by McDonald's."*

3.3.2 Limitations of Prior Approaches

The preceding examples illustrate the need for a *generic* system that supports targeting investigations by identifying not only the fact of targeting but also the likely cause of each targeted output at fine granularity (specific inputs). The experiments must run at *large scale* and any re-

sults must be *statistically justifiable* and *interpretable*. We know of no prior system that satisfies all these properties. Indeed, when turning to prior literature on measurements and tools for web transparency to guide our own design, we discovered significant mismatches at all levels.

We examined the methodologies used by 12 web transparency measurements and tools to study various aspects of data use, including: personalization in search engines[76;202], news and product recommendations[77], and online pricing[135;136;189;77]; targeting in advertising on the web[11;117;45;108] and in mobile apps[141;22]. We make several observations.

• *Generic, reusable methodologies are scarce:* Until 2014 the approach was to investigate specific questions about web targeting and personalization by developing purpose-specific, small-scale experimental methodologies. This resulted in much redundancy between investigations, and typically in small-scale, one-off experiments. In 2014, our team developed XRay[108], the first generic and scalable system design that provides reusable algorithmic building blocks in support of many targeting investigations. Following XRay, AdFisher[45] introduced in 2015 a generic, statistically sound methodology for small-scale targeting investigations. (See §3.7 for further discussion of XRay and AdFisher.)

• *Scalability is often disregarded:* Most prior works[76;202;77;135;136;189;11;117;45;141;22] disregard scalability as a core design goal. Generally speaking, the approach is to observe data flows by varying *one input at a time* in successive experiments. This independent treatment of inputs limits the forms of personalization (e.g. based on location, cookie profile, or some system-specific attributes) that can be detected by the approach. Extending such approaches to scale to many inputs and hypotheses appears difficult. For example, AdFisher builds a separate classifier for each input and validates its effect with experimental data. To investigate targeting on combinations of multiple inputs, one must build a classifier and run a separate experiment for each such combination – an exponential approach that does not scale. XRay is the only prior system that incorporates scalability with many inputs into its design.

• *Confidence assessments are often missing:* Most prior works lack robust statistical justification for their results[135;136;189;76;77;202;11;141;22;117;108]. Many works use case-by-case, comparative

metrics, where the variation in different conditions is compared to that of a control group (e.g. observed price differences[135;136;189], fraction of inconsistent search results[202], Jaccard Index and edit distance[76], normalized discount cumulative gain[77]), but do not report any assessments of statistical confidence or reliability. Other works detect targeting by running basic statistical tests, typically to reject that a given input seen conditionally on a given output is distributed uniformly[11;141;22]. Running these tests multiple times requires a careful correction step, an aspect that is usually ignored. Finally, our own prior system, XRay[108], provides no confidence on an individual finding basis; its predictions are only shown to become asymptotically accurate overall as XRay is applied to larger and larger systems. This makes individual results hard to trust and interpret. In terms of statistical rigor, the most mature approach is AdFisher[45] which, for a given input, builds a specific classifier and validates its effect with statistical confidence.

• *Limited evaluation and design space exploration:* Most prior work lack a rigorous evaluation of the proposed tools and associated design space. In web transparency, evaluation is extremely challenging because ground truth of targeting effects is unknown. Manual assessment is sometimes used in prior work[108;45], but it is, in our experience, extremely prone to error (see §3.6.8). Inability to quantify the accuracy (precision, recall) of an algorithm makes it difficult to explore the design space and understand its trade-offs.

This work seeks to fill in the preceding gaps by presenting: (1) The first generic web transparency methodology that provides both scalability and robust statistical confidence for individual inferences. (2) An implementation of this methodology in Sunlight. Sunlight's design is inspired by XRay and AdFisher, but improves both in significant ways (see §3.7 for detailed comparison). (3) An approach for evaluating the design space of a transparency system like Sunlight. We next begin by describing Sunlight's methodology.

3.4 Design

A core contribution in Sunlight is the development of a principled methodology for web targeting investigations, which follows what we believe are important principles to follow when building infrastructures for other types of web transparency investigations.

3.4.1 Design Principles

• *Design for scale and generality.* The web is big; the number of services and third-parties that could be targeting the users is gigantic. The kinds of personal data they could be using as inputs of their targeting are many and diverse. The number of service outputs that could be targeted at the users is immense. Reliability and trust in an investigation's results depend not only on the methodologies used but also on the scale at which conclusions are reached. Sunlight must thus support large-scale investigations, both in terms of the number of inputs being tracked and in terms of the number of outputs, as well as – to the extent possible – in terms of the services to which it is applied. This last goal requires us to minimize the assumptions we make about the inspected service(s).

• *Provide robust statistical justification for all inferences.* Trustworthiness in the results is key to an investigation, hence Sunlight must provide robust confidence assessments for its inferences. The metrics must be understandable and broadly accepted by the community. Where possible, Sunlight should be able to make causal claims about its inferences, and not simply correlations, which are more difficult to reason about. Enforcing high confidence for all findings may result in missing some. We believe that correct findings are preferable to complete findings. Sunlight hence attempts to limit such effects, but given a choice favors precision over recall.

• *Ensure interpretability of inferences.* A key challenge with many machine learning or statistics mechanisms is that their inferences are often not easily interpretable, and *post hoc* interpretations may not have been statistically validated. Interpretability is a critical aspect of a transparency system as people are the consumers of the system's output. Sunlight explicitly integrates a rudimentary but effective technique to ensure that its inferences are interpretable and statistically validated in these interpretable forms.

To the best of our knowledge, Sunlight is the first web transparency system to closely follow all these principles. It can currently run on hundreds of virtual machines to process data from targeting experiments, and precisely detects targeting of tens of thousands of Gmail and web ads, testing

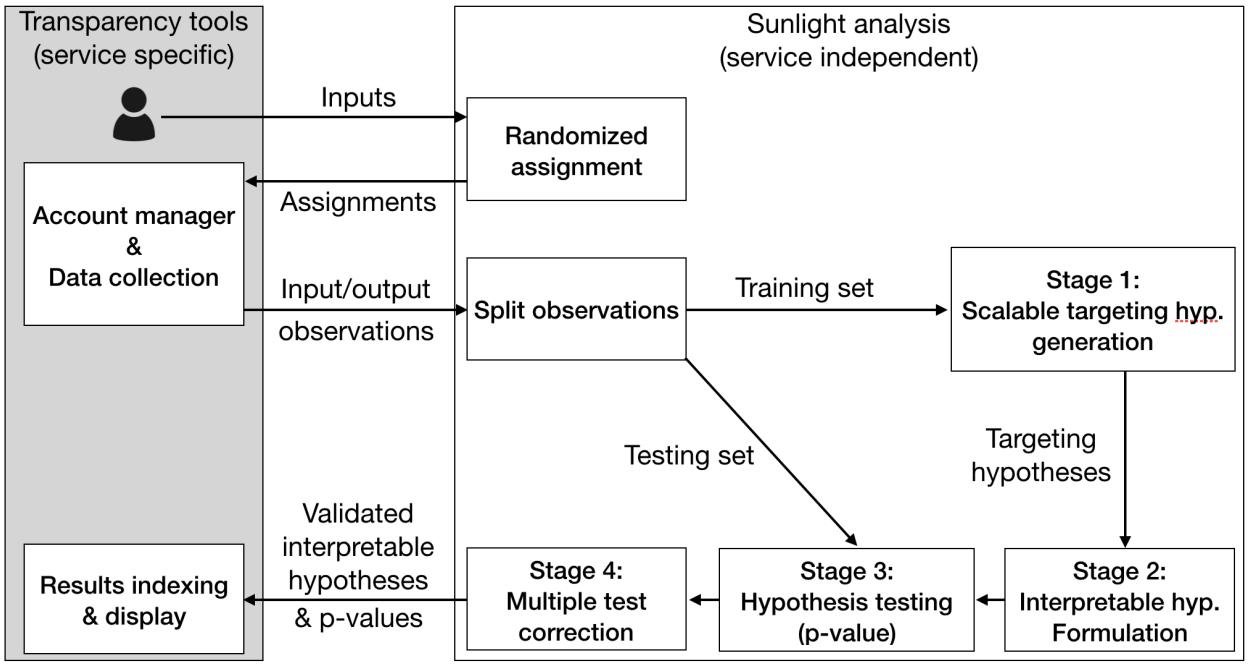


Fig. 31: **The Sunlight methodology.** Relies on randomized input assignments for causality and a train/test set split for statistical soundness. The analysis consists of a four-phase pipeline: (1) scalable hypothesis generation, (2) interpretable hypothesis formation, (3) hypothesis testing, and (4) multiple test correction. **The Sunlight system.** Adds: service specific account managers responsible for populating and managing the accounts used in experiments, perform data collection, and store observations; an index of results with a search interface to explore findings.

hundreds of inputs simultaneously. It minimizes the assumptions it makes about the services and provides statistically significant and interpretable results.

3.4.2 Methodology

Fig. 31 shows the Sunlight methodology. It consists of a pipeline with four data analysis stages. Those stages depend on experimental data from an initial data collection step determined by the investigator. Data collection begins with the creation of several fictitious user profiles, each with randomly-assigned input attributes (called *inputs*) which are potentially visible to an ad targeting mechanism. For instance, an input may indicate the presence of a particular e-mail in the user profile's webmail inbox, or that the user profile was used to visit a particular website. Then, each user profile is used to measure several potential effects of targeting mechanisms (*outputs*), such as specific ad displays shown on browser visits to a news website or webmail service. The inputs should be specified *a priori*, and for various reasons which we discuss later, it will be desirable

that the random assignment of input values be statistically independent (across different inputs and across different user profiles); the outputs may be specified generically (e.g., all possible ads displayed in ten refreshes of `cnn.com`), so that the set of outputs is only determined *post hoc*. The end result is a data set comprised of inputs and output measurements for each profile.

At its core, the Sunlight methodology analyzes the collected data set using a sample-splitting approach (sometimes called the "holdout method" in machine learning) to generate and evaluate targeting hypotheses. The profiles in the data set are randomly split into a training set and a testing set. In **Stage 1 (Scalable Hypothesis Generation)**, we apply scalable classification and regression methods to the training set to generate prediction functions that can explain the output measurements for a user profile (e.g., indicator of whether a particular ad was displayed to the user) using the profile's input attributes. We focus on scalable methods that generate *simple functions* of a *small number of inputs* so that they are readily interpretable as targeting hypotheses, and take explicit measures in **Stage 2 (Interpretable Hypothesis Formation)** to ensure this if necessary. In addition, we discard any prediction functions that fail some simple sanity checks so as to reduce the number of targeting hypotheses; this again is performed just using the training set. At the end of Stage 2, we have a filtered collection of interpretable targeting hypotheses generated using only the training set.

In **Stage 3 (Hypothesis Testing)**, each such hypothesis is then evaluated on the testing set using a statistical test to generate a measure of confidence in the targeting hypothesis—specifically, a p-value. The p-value computations may make use of the known probability distributions used to assign input values in the test set profiles, and each targeting hypothesis' p-value should be valid under minimal assumptions. Because such statistical tests may be conducted for many targeting hypotheses (e.g., possibly several targeted ads), we finally apply a correction to these confidence scores in **Stage 4 (Multiple Testing Correction)** so that they are *simultaneously* valid. We may then filter the targeting hypotheses to just those with sufficiently high confidence scores, so that the end result is a statistically-validated set of interpretable targeting hypotheses.

3.4.3 Threat Model and Assumptions

Like all prior transparency systems of which we are aware, we assume that Web services, advertisers, trackers, and any other parties involved in the web data ecosystem, do *not* attempt to frustrate Sunlight's targeting detection. In the future, we believe that robustness against malicious adversaries should become a core design principle, but this work does not provide such progress. Moreover, we assume that the users leveraging Sunlight are tech-savvy and capable of developing the measurement data collection necessary to collect the data. Sunlight enables targeting detection given the experimental datasets obtained through independent means.

While Sunlight can establish correlation and even causation in some circumstances between particular inputs and targeted outputs (within some confidence level), it *cannot* attribute targeting decisions to particular parties (e.g., advertisers, ad networks, trackers, etc.), nor can it distinguish between *intentional targeting* (e.g., advertisers choosing to target users in a particular category) versus algorithmic decisions (e.g. an unsupervised algorithm decides to target a particular population of users based on patterns of prior ad clicks). Moreover, because Sunlight's correlations and causations are obtained from controlled experiments with synthetic user profiles, its findings are not guaranteed to be representative of the targeting on the real population. Finally, while Sunlight can detect certain combined-input targeting, it cannot detect all forms of targeting, but rather only targeting on disjunctive (OR) combinations of a limited number of controlled inputs.

Given all of these constraints, Sunlight is best used in contexts where its results inform and provide believable justification for subsequent investigations through independent means aimed at establishing the "truth." Our scenarios in §3.3.1 fall into this category.

3.4.4 Architecture

Sunlight instantiates the preceding methodology to detect, validate, and report the likely causes of targeting phenomena on the web. This raises three significant challenges. First, at each stage, unique aspects of our domain require careful modeling of the problem to map them onto appropriate statistical mechanisms. Second, across stages, mechanisms may interact in subtle ways and require careful and challenging co-designs. For example, as §3.6 shows, a design choice to use

a permissive classification at Stage 1 (high recall but low precision) results in significant penalty due to correction in Stage 4 and failure to validate many true hypotheses (i.e., poor recall at the end of the Sunlight pipeline). In contrast, a stricter Stage 1 method that we developed for Sunlight (§3.5.1), which has comparatively lower recall but higher precision in it of itself results in better recall at the end of the Sunlight pipeline. Thus, a second key contribution in Sunlight is to identify the key requirements that must be met by the mechanisms we use at each stage of the pipeline and combine them in ways that provide scalability, confidence, and interpretability.

To address these challenges, we have designed Sunlight to be modular. It allows both the instantiation of multiple pipelines and the evaluation and comparison at different levels of the pipeline. This provides two benefits. First, it lets us explore the design space and choose the best combination of mechanisms for our problem. Second, it lets our users – researchers and investigators – adapt Sunlight to their own needs. For example, some mechanisms provide confidence at scale while others provide superior statistical guarantees; with Sunlight users can make the choices they prefer. Fig. 32 lists the mechanisms we currently support at each stage, some of them which we imported from prior literature, others we developed to address limitations of prior mechanisms (see §3.5).

3.4.5 Prototype

We implemented Sunlight in Ruby using statistical routines (e.g., Lasso) from R, a programming environment for statistical computing. The analysis is built around a modular pipeline that lists the algorithms to use for each stage, and each algorithm implements a basic protocol to communicate with the next stage.

Service Specific Transparency Tools. The Account Manager (Fig. 31) is responsible for: (1) populating accounts on the target Web service with its assigned inputs, and (2) periodically retrieving outputs from the audited service for each shadow account. Both functions are service specific. For Gmail, they involve sending emails with SMTP and invoking the ad API to request ads. For YouTube, they involve streaming a video and scraping recommendations, and for Amazon, they involve placing products in wish lists and scraping recommendations. The complexity of these tasks

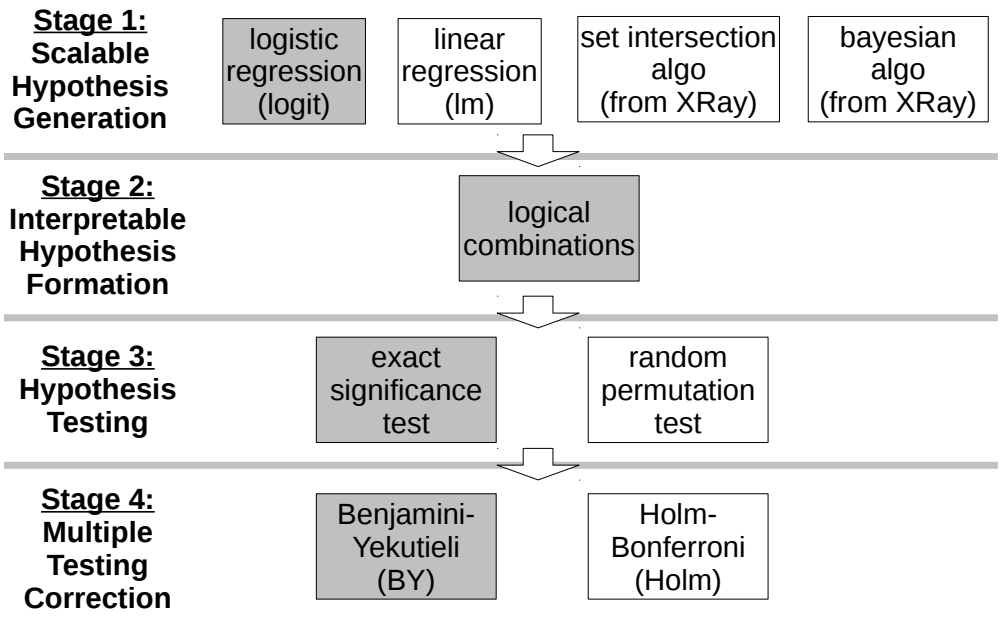


Fig. 32: **The Sunlight modular pipeline.** Grey boxes show the default pipeline, which in our experience strikes a good balance between results confidence and interpretability versus support for large-scale investigations.

depends on the availability of APIs or the stability of a service's page formats. Outputs collected from the Web service are placed into a database (Cassandra), which maps shadow accounts to their input sets, output observations, and various metadata (e.g. context input, observation timestamp). After running the analysis pipeline, results are indexed in Elasticsearch to enable convenient access and analysis.

Default Analysis Pipeline. Fig. 32 shows the default pipeline used by Sunlight. In Stage 1, we use sparse logistic regression (Logit) to estimate regression coefficients that give an ordering over the inputs. In Stage 2, we select a disjunction (i.e., an "OR" combination) with the best predictive accuracy from ordered inputs from Stage 1, and discard inaccurate hypotheses as determined using heuristic p-value computations. Stage 3 computes the p-values for the statistical test of independence on the test data. Finally, our Stage 4 implementation computes both the Benjamini-Yekutieli (BY) and Holm-Bonferroni (Holm) corrections, though our default recommendation is the BY correction. Finally, we recommend p-values < 0.05 for good confidence.

In §3.6, we show that these defaults strike a good balance between the scalability and the confidence of these hypotheses. Using these defaults, our targeting experiments on Gmail and on the web produced the largest number of high confidence hypotheses, and we have manually inspected many of these hypotheses to validate the results. We describe these measurements next.

3.5 Theory

We next describe the mechanisms we developed for each stage of Sunlight, and detail the semantics and guarantees they offer.

3.5.1 Stage 1: Scalable Hypothesis Generation

We generate *interpretable* targeting hypotheses by applying classification and regression methods to the training set; the hypotheses are formulated as interpretable functions of a profiles' input attributes that can explain the corresponding output measurements. In machine learning parlance, we train predictors of the outputs based on the inputs (as input variables). To ensure that the hypotheses are interpretable, we explicitly seek predictors that only depend on a few inputs, and that only have a simple functional form. To this end, we restrict attention to hypotheses about the output that can be represented as a disjunction formula over at most k inputs for some small integer number k (here we also assume for simplicity that inputs and outputs are binary-valued). This class of small disjunction formulae is one of the simplest classes of natural and interpretable hypotheses; our focus on this class here serves as a starting point for building up techniques for other hypothesis classes.

Even with the restriction to the simple class of disjunction formulae, we face formidable computational challenges. Finding the most accurate such disjunction on an arbitrary training set is generally computationally intractable in very strong senses[67], with brute force enumeration requiring $\Omega(d^k)$ time for d inputs. Therefore, we can only hope to find accurate small disjunction hypotheses that have additional special structure. In some cases, we can use a two-step greedy approach to find such disjunction hypotheses: (i) we first use scalable classification and regression methods to order the inputs by some measure of relevance—e.g., by their average correlation with the output across user profiles in the training set (this stage); then (ii) we use this ordering over

inputs to greedily construct a disjunction with sufficiently high accuracy on the training set (Stage 2). Under other conditions, it may be possible to directly form small disjunction hypotheses. We discuss three approaches we implemented in more detail below.

Threshold Set Intersection

First, we present the *Set Intersection Algorithm* (Fig. 33) for which we can prove that, under some assumptions, infering targeting requires no more than a *logarithmic* number of accounts as a function of the number of inputs.

Specifically, we assume that the studied service decides targeting using a disjunctive combination of the inputs (e.g. show O_k if D_5 *or* D_8 are in the account) with a size bounded by a maximum *input size*, r. Additionally, we define the following three probabilities: p_{in}, the probability that an account containing the input targeted by a particular ad, will see that ad at least once; p_{out} the probability that an account lacking the targeted input D_i will see the ad; and $p_\emptyset$ the probability that an ad that is nottargeted will appear in a given account. We assume that $p_{\text{in}}, p_\emptyset, p_{\text{out}}$ are constant across all emails, ads, and time, and that p_{out} is strictly smaller than p_{in} (bounded noise hypothesis). Under these assumptions, we prove the following theorem:

Theorem 5. Under the previous assumptions, for any $\varepsilon > 0$ there exists an algorithm that requires $C \times \ln(N)$ accounts to correctly identify the inputs of a targeted ad with probability $(1 - \varepsilon)$. The constant C depends on ε and the maximum size of combinations r ($O(r2^r \log(\frac{1}{\varepsilon}))$).

Given that outputs will appear more often in accounts containing the targeting inputs, the core of the algorithm is to determine the set of inputs appearing in the highest number of accounts that also see a given ad. We prove that it has the correctness and scaling properties specified in the theorem. Here we describes a basic version of the algorithm that makes some simplifying assumptions and provides a brief proof sketch. The detailed proof and complete algorithm are described in our technical report[107].

Algorithm. The algorithm relies on a randomized placement of inputs into shadow accounts, with some redundancy to cope with imperfect coverage. We thus pick a probability, $0 < \alpha < 1$, create

```
// Set Intersection Algo:
// Runs with each collected ad.
In: Output O_k (e.g. an ad).
Params: MIN_ACTIVE_ACCTS, THRESHOLD.
Out: Targeted input combination.
// Step 1: Compute active accounts.
A_k = the accounts that see ad O_k.
if |A_k| < MIN_ACTIVE_ACCTS
  return ∅
end
// Step 2: Create input combination hypothesis.
targeted_set = ∅
foreach input D_i do
  if (number of A_k containing D_i) / |A_k| >THRESHOLD
    targeted_set += D_i
  end
end
// Step 3: Verify it is a real combination.
if (number of A_k containing entire targeted_set) / |A_k| <THRESHOLD
  return ∅
end
// targeted_set triggered the output.
return targeted_set
```

Fig. 33: **The Set Intersection Algorithm.** Can be proven to predict targeting correctly under certain assumptions with a logarithmic number of accounts.

$C\ln(N)$ shadow accounts, and place each input D_i randomly into each account with probability α. Fig. 33 shows the Set Intersection algorithm for a set of observations, $\vec{x}$. Given an output O_k collected from the user account, we compute the set of *active accounts*, A_k, as those shadow accounts that have seen the output (Step 1). We then compute the set of inputs that appear in at least a threshold fraction of active accounts; this set is our candidate for the combination being targeted by the ad (Step 2). Finally, we check that the entire combination is in a threshold fraction of the active accounts (Step 3). Theoretically, we prove that there exists a threshold for which the algorithm is arbitrarily correct with the available $C\ln(N)$ accounts. Practically, this threshold must be tuned experimentally to achieve good accuracy on every service – a key reason for our Bayesian enhancement in §3.5.1.

Correctness Proof Sketch. The proof shows that if there were targeting, every non-targeting input would have a vanishingly small probability to be in a significant fraction of the active accounts. Let us call *S* the set of inputs contained in a significant fraction of the active accounts. Without targeting, these inputs would be present in the accounts by mere chance. Since inputs are independently distributed into the accounts, we show that the probability of *S* not being empty decreases

```
// Bayesian Prediction Alg:
// Runs with each collected ad.
In: Output O_k (e.g. an ad).
Out: Targeted input.
// Compute probabilities.
foreach input D_i do
  P[D_i | x⃗] = bayes(P[x⃗ | D_i])
end
// Compute untargeted prob.
P[D_∅ | x⃗] = bayes(P[x⃗ | D_∅])
// Return event with max prob.
return D_i with max P[D_i | x⃗]
```

```
// Parameter Learning Alg:
// Runs periodically.
// Initialize params (arbitrary).
p_in = .7, p_out = .01, p_∅ = .1
do
 foreach output O_k do
  Run Bayesian Prediction.
 end
 Update p_in, p_out, p_∅
  from predictions.
until p_in, p_out, p_∅ converge
end
```

Fig. 34: **Bayesian Algorithm.** Left: Bayesian prediction algorithm for behavioral targeting. Right: Expectation Maximization algorithm to learn parameters.

exponentially with the number of active accounts (through Chernoff bounds). With targeting, we show that with high probability no other input than the explaining combination is in *S*, because of the bounded noise hypothesis.

Self-Tuning Bayesian Algorithm

The Set Intersection algorithm provides a good theoretical foundation; however, it requires parameters be tuned and applies only to behavioral targeting, not contextual targeting. Thus, we include a more robust, self-tuning version that leverages a Bayesian model and en expectation maximization (EM) algorithm to automatically adjust parameters.

The Bayesian model leverages the same information from the shadow account observations, $\vec{x}$. It counts the observations x_j of ad O_k in an account j as a binary signal: if the ad has appeared at least once in account j, we count it once; otherwise we do not count it. Briefly, the Bayesian model is a simple generative model that simulates the audited service given some targeting associations (e.g., D_i triggers O_k). It computes the probability for this model to generate the outputs we do observe for every targeting association. The most likely association will be the one Sunlight returns.

In more detail if the ad were targeted towards D_i, then an account j containing D_i would see this ad at least once with a *coverage* probability p_{in}; otherwise, it would miss it with probability $(1 - p_{\text{in}})$. An account j' without input D_i would see the ad with a smaller probability, p_{out}, missing it with probability $(1 - p_{\text{out}})$. If the ad were not behaviorally targeted, it would appear in each

account with the same probability, $p_\emptyset$. If we define A_k as the set of active accounts that have seen the ad, and A_i as the set of accounts that contain email D_i, then we have the following definitions for the probabilities:

$$\begin{aligned}\mathbb{P}\left[\vec{x}\middle|\, D_i\right] = &\quad (p_{\text{in}})^{|A_i \cap A_k|}\,(1-p_{\text{in}})^{|A_i \cap \bar{A}_k|} \times (p_{\text{out}})^{|\bar{A}_i \cap A_k|}\,(1-p_{\text{out}})^{|\bar{A}_i \cap \bar{A}_k|}\ ,\\ \mathbb{P}\left[\vec{x}\middle|\, D_\emptyset\right] = &\quad (p_\emptyset)^{|A_k|}\,(1-p_\emptyset)^{|\bar{A}_k|}\ ,\end{aligned}$$

where $D_\emptyset$ designates the untargeted prediction.

The preceding formula has an interesting interpretation that is visible if placed in the equivalent form:

$$\mathbb{P}\left[\vec{x}\middle|\, D_i\right] = \quad (p_{\text{in}})^{|A_k|}\,(1-p_{\text{out}})^{|\bar{A}_k|} \times \left(\frac{1-p_{\text{in}}}{1-p_{\text{out}}}\right)^{|A_i \cap \bar{A}_k|} \left(\frac{p_{\text{out}}}{p_{\text{in}}}\right)^{|\bar{A}_i \cap A_k|}$$

From the point of view of the event D_i, an account found in $A_i \cap \bar{A}_k$ is a false positive (an ad was expected but not shown). This should lower the probability, especially when the coverage p_{in} is close to 1. Inversely, an account found in $\bar{A}_i \cap A_k$ acts as a false negative (we observed an ad where we did not expect it), which should decrease the probability, especially when p_{out} is close to 0.

These formulas let us infer the likelihood of event D_i according to Bayes' rule: $\mathbb{P}\left[A\middle|\, B\right] = \frac{\mathbb{P}[B|\,A]\times\mathbb{P}[A]}{\mathbb{P}[B]}$. Figure 34 shows two algorithms. First, the prediction algorithm (left) predicts the targeting of O_k by computing the probabilities defined above, applying Bayes' rule, and returning the input with the maximum probability. Second, the EM parameter learning algorithm (right) computes the variables that those probabilities depend upon (p_{in}, p_{out}, and $p_\emptyset$) using an iterative process. It repeatedly runs the prediction algorithm for all outputs and re-computes p_{in}, p_{out}, and $p_\emptyset$ based on the predictions. It stops when the variables converge (i.e., their variation from one iteration to another is small).

Sparse regression.

Our last technique for ordering inputs, and the one we will use as default, is based on linear regression. In our application, each of the d inputs is regarded as a (boolean) predictive variable, and our goal is to find a linear combination of these d variables $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ that predicts an associated output measurement. The coefficients $\mathbf{w} = (w_1, w_2, \ldots, w_d)$ used to form the linear combination $\langle \mathbf{w}, \mathbf{x} \rangle = \sum_{i=1}^{d} w_i x_i$ are called the *regression coefficients*, and these can be regarded

as a measure of association between the inputs and the output. These coefficients are estimated from a collection of d-dimensional data vectors, which in our setting are the vectors of input attributes for each profile in the training set.

We use a sparse linear regression method called Lasso[183] to estimate the regression coefficients $\mathbf{w}$. Lasso is specifically designed to handle the setting where the number of inputs may exceed the number n of data vectors (i.e., user profiles) in the training set, as long as the number of non-zero regression coefficients is expected to be small—i.e., the coefficient vector is sparse. This sparsity assumption entails that only a few inputs are, in combination, correlated with the output. Under certain conditions on the n data vectors (which we ensure are likely to be satisfied *by construction* of our user profiles), Lasso accurately estimates the coefficients as long as $n \geq O(k \log d)$, where k is the number of non-zero coefficients[15]—i.e., the number of input variables potentially correlated with the output. In fact, this collection of $O(k \log d)$ input vectors supports the *simultaneous* estimation of multiple coefficient vectors for different outputs (e.g., different ads), a consequence of the same phenomenon that underlies compressed sensing[49].

Linear regression permits the use of additional variables for uncontrolled factors (e.g., time-of-day, IP address of machine used to collect data) to help guard against erroneous input/output associations that could otherwise be explained by these factors. For instance, some ads may be shown more during work hours to target office workers, but the time-of-day in which data is collected for certain profiles could inadvertently be correlated with some inputs. Including time-of-day as a variable in the regression model helps suppress these unwanted associations.

We also consider the use of a generalized linear model called *logistic regression*, which is especially suited for binary outputs. This model posits that $\Pr[\text{output} = 1] = g(\langle \mathbf{x}, \mathbf{w} \rangle)$, where $g(z) = 1/(1 + e^{-z})$. To estimate regression coefficients in this model, we use a variant of Lasso called L_1-regularized logistic regression[143], whose effectiveness has been established in several empirical studies across multiple domains (e.g.,[199;19]). As in Lasso, we are able to regard the inputs with large estimated coefficients as likely to be relevant in predicting the output (and this

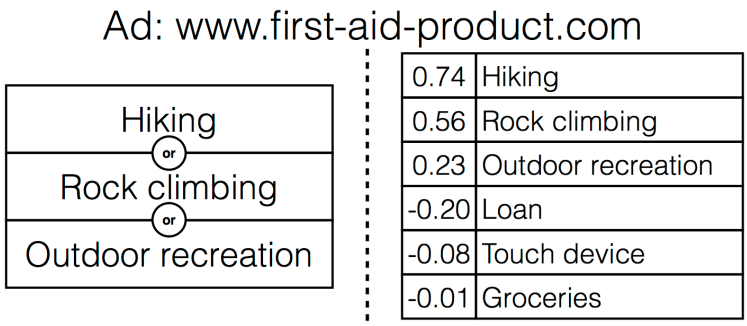


Fig. 35: **Interpretable vs. raw hypothesis.** (Left) Targeting hypothesis formulated as disjunction of inputs. (Right) Raw hypothesis based on logistic regression parameters.

would not be the case if we used unregularized or L_2-regularized logistic regression, at least in this $n \ll d$ regime).

3.5.2 Stage 2: Interpretable Hypothesis Formation

Given an ordering over the inputs, we form disjunctions of inputs in a greedy fashion. Specifically, we first consider a singleton disjunction with the first input on the list, then a disjunction of the first two inputs on the list, and so on. We proceed as long as the training accuracy of the disjunction (in predicting the output for profiles in the training set) is sufficiently high; the criterion we use to determine this threshold is just a heuristic, but is similar to the hypothesis test used in the testing stage (§3.5.3). The largest sufficiently accurate disjunction is then taken (together with the associated output) as a targeting hypothesis. For some outputs, it is possible that even the singleton disjunction lacks high-enough accuracy; in such cases no hypothesis is formed.

Fig. 35 shows an example of this transformation for a hypothesis based on logistic regression. The leftside hypothesis says a profile is targeted if it has at least one of the shown inputs; the rightside hypothesis says a profile is targeted if the sum of coefficients for the inputs in the profile is greater than zero. The latter appears more difficult to interpret than the former.

An alternative is to forgo interpretability and seek out out any kind of potential association between inputs and outputs. For instance, we could look for associations between arbitrary functions of inputs and outputs by using richer classes of prediction functions beyond simple disjunctions (e.g. arbitrary linear threshold functions), as well as by using other flexible measures of associa-

tion (e.g.[74]). Such associations may be much easier to detect and statistically validate, but they may not be readily interpretable nor easy to reason about in a follow-up studies.

An important and subtle note is that if interpretability is important, then any transformation needed for interpretation should be applied at this stage. For example, an intuitive but *incorrect* way of interpreting a result from Sunlight would be to generate raw hypotheses, validate them in Stages 3 and 4, and then "interpret" those with low enough p-values. That would result in potentially misleading conclusions. For example, just because a hypothesis based on a logistic model can be validated with low p-values, it does not follow that the corresponding disjunctive version of that hypothesis is also statistically significant. For this reason, the Sunlight methodology critically includes this explicitly interpretability stage, which reminds a developer to transform her hypothesis early for interpretability so the p-values can be computed for *that* hypothesis.

3.5.3 Stage 3: Hypothesis Testing

The second stage of the analysis considers the targeting hypotheses (disjunctions of inputs and an associated output) generated from the first stage and provides a confidence score for each hypothesis. The score, a p-value, comes from an exact statistical test that decides between a null hypothesis H_0 that the disjunction of inputs is independent of the associated output, and an alternative hypothesis H_1 that the disjunction of inputs is positively correlated with the output. A small p-value—≤ 0.05 by convention—implies that our observations on the data (discussed below) are unlikely to be seen under H_0; it lends confidence in rejecting H_0 and accepting H_1 and the validity of the targeting hypothesis.

Computing p-values. The test is based on computing a test statistic on the testing set (i.e., the subset of profiles that were not used to generate targeting hypotheses). A critical assumption here is that the profiles (and specifically, the outputs associated with each profile) are statistically independent, and hence the selected disjunction is also independent of the profiles in the testing set. The specific test statistic T we use is an association measure based on Pearson's correlation: we compute T using the profiles from the testing set, and then determine the probability mass of the interval $\{t \in \mathbb{R} : t \geq T\}$ under H_0. This probability is precisely the p-value we seek.

Because the distribution of the inputs for each profile is known (and, in fact, controlled by us), it is straightforward to determine the exact distribution of T under H_0. For example, when the inputs for each profile are determined with independent but identically distributed coin tosses, the p-value computation boils down to a simple binomial tail calculation.

More specifically, suppose the inputs are independent and identically distributed binary random variables with mean $\alpha \in (0, 1)$. Consider a disjunction of k inputs and a particular (binary-valued) output. Let N be the number of profiles for which the output is 1, and let B be the number of profiles for which both the disjunction and the output are 1. If $N = 0$, then the p-value is 1. Otherwise, the p-value is $\sum_{i=B}^{N} \binom{N}{i} \alpha_k^i (1 - \alpha_k)^{N-i}$ where $\alpha_k = 1 - (1 - \alpha)^k$.

Use of p-value in Stage 2. As previously mentioned, we also use this p-value computation in Stage 2 *as a rough heuristic* for deciding which disjunctions to keep and pass on to Stage 3. However, we stress that these Stage 2 p-value computations are not valid because the disjunctions are formed using the profiles from the training set, and hence are not independent of these same training set profiles. The validity of the Stage 3 p-values, which are based on the testing set, relies on the independence of the disjunction formulae and the testing set profiles themselves.

Independence assumption. It is possible to weaken the independence assumption by using different non-parametric statistical tests, as is done in[45]. Such tests are often highly computationally intensive and have lower statistical power to detect associations. We opt to admit the assumption of independence in favor of obtaining more interpretable results under the assumption; gross violations may be identified in follow-up studies by an investigator, which we anyway recommend in all cases.

Causal effects. Under the alternative hypothesis of a positive correlation between a disjunction of inputs and an output, it is possible to draw a conclusion about the *causal effect* of the inputs on the output. Specifically, if the input values are independently assigned for each user profile, then a positive correlation between a given disjunction of inputs and an output translates to a positive *average causal effect*[166]. This independent assignment of input values can be ensured in the creation of the user profiles.

3.5.4 Stage 4: Multiple Testing Correction

In the final stage of our analysis methodology, we appropriately adjust the p-values for each of our targeting hypotheses to correct for the *multiple testing problem*. As one simultaneously considers more and more statistical tests, it becomes more and more likely that the p-value for some test will be small just by chance alone even when the null hypothesis H_0 is true. If one simply rejects H_0 whenever the stated p-value is below 0.05 (say), then this effect often leads to erroneous rejections of H_0 (*false rejections*).

This multiple testing problem is well-known and ubiquitous in high-throughput sciences (e.g., genomics[52]), and several statistical methods have been developed to address it. A very conservative correction is the Holm-Bonferroni method[82], which adjusts the p-values (generally making them larger them by some amount) in a way so that the probability of *any* false rejection of H_0 (based on comparing adjusted p-values to 0.05) is indeed bounded above by 0.05. While this strict criterion offers a very strong guarantee on the resulting set of discoveries, it is often overly conservative and has low statistical power to make any discoveries at all. A less conservative correction is the Benjamini-Yekutieli procedure[14], which guarantees that among the adjusted p-values that are less than 0.05, the expected fraction that correspond to false discoveries (i.e., false rejections of H_0) is at most 0.05. Although this guarantee on the expected false discovery rate is weaker than what is provided by the Holm-Bonferroni method, it is widely accepted in applied statistics as an appropriate and preferred correction for exploratory studies.

With either correction method, the adjusted p-values provide a more accurate and calibrated measure of confidence relative to the nominal 0.05 cut-off. We can either return the set of targeting hypotheses whose p-values fall below the cut-off, or simply return the list of targeting hypotheses ordered by the p-values. Either way, the overall analysis produced by this methodology is highly-interpretable and statistically justified.

3.6 Evaluation

We evaluate Sunlight by answering the following questions: (Q0) Sanity check: can Sunlight detect our own advertizing? (Q1) How accurate is Sunlight against ground truth, where it is avail-

able? (Q2) How does Sunlight's Stage 1 account requirements scale with input size? (Q3) How do different Stage 1 algorithm's hypotheses compare? (Q4) What is the influence of p-value correction on Sunlight? (Q5) How does scale affect confidence? As foreshadowing, we show that Sunlight's high-confidence hypotheses are precise, and that the Logit (logistic regression) method is best suited among those we evaluated for maximizing hypothesis recall after p-value correction. Somewhat surprisingly, we show that the "winning" inference algorithm at Stage 1 (the Bayesian algorithm) is *not* the winner at the end of the pipeline, after correction is applied. Finally, we show that the same effect in also responsible for a trade-off between confidence and scalability in the number of outputs.

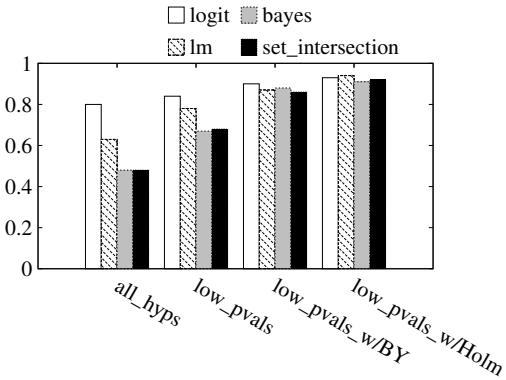
Finally, we conclude with a measurement study of Gmail ads and ads on the web, in which we find contradictions of statements from Google policies or official FAQs.

3.6.1 Methodology

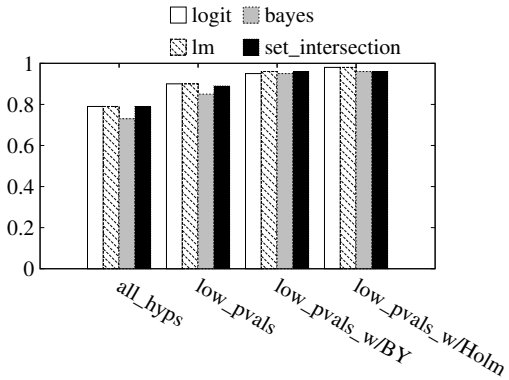
We evaluate Sunlight using the system's split of observations into a training and a testing set, and leveraging the modularity of Sunlight to measure the effectiveness of targeting detection at different stages of the analysis pipeline. We believe that our evaluation methodology, along with the metrics that we developed for it, represents a significant contribution and a useful starting point for the evaluation of future transparency infrastructures, an area that currently lacks rigorous evaluations (see §3.3.2).

A critical challenge in evaluating Sunlight and its design space is the lack of ground truth for targeting for most experiments. For example, in Gmail, we do not know how ads are targeted; we can take guesses, but that is extremely error prone (see §3.6.8). In other cases (such as for Amazon and Youtube recommendations), we can obtain the ground truth from the services. For a thorough evaluation, we thus decided to use a multitude of metrics, each designed for a different situation and goal. They are:

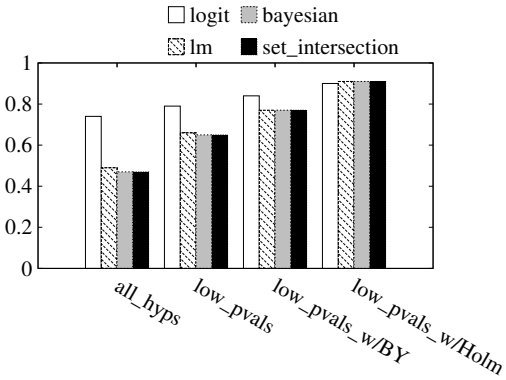
1. *hypothesis precision*: proportion of high-confidence hypotheses that are true given some ground truth assessment.



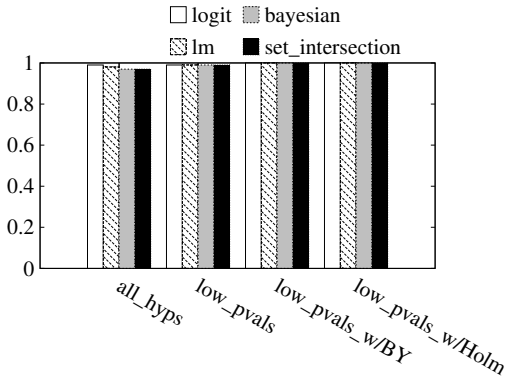
(a) **Ad prediction precision, Gmail**



(b) **Ad prediction recall, Gmail**



(c) **Ad prediction precision, Websites**



(d) **Ad prediction recall, Websites**

Fig. 36: **Ad prediction precision and recall.** The x-axis shows different p-value correction methods, and the y-axis shows the proportion of precision and recall. (a) and (b) show ad prediction precision and recall on the Gmail dataset, (c) and (d) on the Websites dataset, for all algorithms. Both metrics increase when using stricter p-values, indicating better hypotheses.

2. *hypothesis recall*: proportion of true hypotheses that are found from some ground truth assessment.
3. *ad prediction precision*, the proportion of success in predicting if an ad will be present in a training set account.[2]
4. *ad prediction recall*: proportion of ads appearances that were correctly guessed when predicting if an ad will be present in a training set account.
5. *algorithm coverage*: proportion of low p-value hypotheses found by an algorithm, out of all low p-value hypotheses found by any of the algorithms.

[2]"Ad" in this section is short for the more generic output.

Workload	Profiles	Inputs	Outputs
Gmail (one day)	119	327	4099
Website	200	84	4867
Website-large	798	263	19808
Gmail-simple	35	51	193
YouTube	45	64	308
Amazon	51	61	2593

Tab. 31: **Workloads used to evaluate Sunlight**

We use the first two metrics in cases where ground truth is available (§3.6.3) and with manual assessments (§3.6.2, §3.6.4, §3.6.8). These are typically small scale experiments. We use the next two metrics in cases where ground truth is unavailable; this lets us evaluate at full scale and on interesting targeting. Finally, we use the last metric for comparison of various pipeline instantiations.

Table 31 shows the datasets on which we apply these metrics. The first three datasets come from the experiments described in the preceding section. The Gmail dataset corresponds to one day's worth of ads in the middle of our 33-day experiment. The Gmail-simple, YouTube, and Amazon datasets are from our early work on XRay[108]. They contain Gmail ads, and targeting observations for the recommendation systems of YouTube and Amazon, for videos and products respectively. They are small (about 60 inputs), and with inputs on very distinct topics, minimizing the chances for targeting on input combinations. We created the inputs workload for each service by selecting topics from well-defined categories relevant for that service. For Gmail and YouTube, we crafted emails and selected videos based on AdSense categories; for Amazon, we selected products from its own product categories. On the other hand the Gmail and Websites datasets are larger scale, with up to 327 inputs and thousands outputs. Moreover their inputs are not distinct, containing some redundancy because they include emails or websites on the same topics that are more likely to attract similar outputs. They are thus more representative of experiments that would be conducted by investigators.

To assess Sunlight's hypothesis precision and recall on small experiments (last three), we needed the ground truth for associations. Amazon and YouTube provide it for their recommendations. For instance, Amazon provides a link "Why recommended?" which explicitly explains

Ad Keyword	Targeted Email	Detected by Sunlight?	# Accounts & Displays
Chaldean Poetry	Like Chaldean Poetry?	Yes	13/13, 1588/1622
Steampunk	Fan of Steampunk?	Yes	13/13, 888/912
Cosplay	Discover Cosplay.	Yes	13/13, 440/442
Falconry	Learn about Falconry.	Yes	13/13, 1569/1608

Fig. 37: **Self-Targeted Ads.** The last column shows raw behavioral and contextual data for interpretation: X/Y, Z/T means that the ad was seen in X active accounts that contain the targeted email out of a total of Y active accounts; the ad was shown Z times in the context of the targeted email out of a total of T times.

the recommendation ; when clicked, it shows an explanation of the form "The [Coloring Book] is recommended because your wish list includes [Crayola Crayons Set]." YouTube explains its video recommendations similarly. For Gmail, we manually labeled ads based on our personal assessment. The ads for different experiments were labeled by different people, generally project members. A non-computer scientist labeled the largest experiment (51 emails).

3.6.2 Q0: Sanity-check experiment: detecting our own advertizing

To build intuition into Sunlight's functioning, we ran a simple sanity-check experiment on Gmail. Recall that, unlike Amazon and YouTube, Gmail does not provide any ground truth, requiring us to manually label associations, a process that can be itself faulty. Before measuring Sunlight's accuracy against labeled associations, we checked that Sunlight can detect associations for our own ads, whose targeting we control. For this, we strayed away from the aforementioned methodology to create a highly controlled experiment. We posted four Google AdWords campaigns targeted on very specific keywords (Chaldean Poetry, Steampunk, Cosplay, and Falconry), crafted an inbox that included one email per keyword, and used Sunlight to recover the associations between our ads and those emails. In total, we saw our ads 1622, 912, 442, and 1608 times, respectively, across all accounts (shadows and master). Fig. 37 shows our results. After one round of ad collection (which involved 50 refreshes per email), Sunlight correctly associated all four ads with the targeted email. It did so with very high confidence: the p-value $<< 0.05$ in all cases. The figure also shows some of the raw contextual/behavioral data, which provides intuition into

Sunlight's perfect precision and recall in this controlled experiment. We next turn to evaluating Sunlight in less controlled environments.

3.6.3 Q1: Precision and recall on ground truth

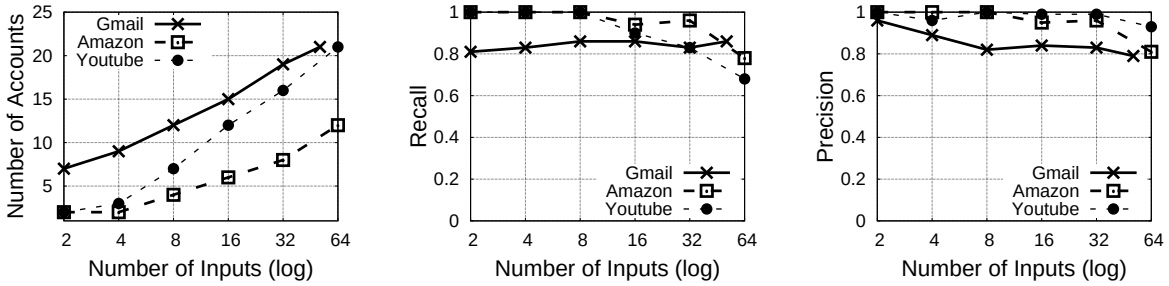
Dataset	Precision		Recall		Hyp. count
	Sunlight	XRay	Sunlight	XRay	
Amazon	100%	81%	46%	78%	142
YouTube	100%	93%	52%	68%	1349

Tab. 32: **Sunlight's hypothesis precision & recall**

Sunlight favors finding reliable, validated targeting hypotheses over finding every potential targeting, so that investigators do not waste time on dead ends. This strategy is characterized by *hypothesis precision* that should be very high, and *hypothesis recall* that we try to keep high without lowering precision. We measure these two metrics on two datasets from YouTube and Amazon, both containing ground truth (Amazon and YouTube inform users why they are shown certain recommendations). This gives us a direct comparison with prior art, as well as an assessment of Sunlight's hypothesis precision and recall on services provided ground truth for recommendation targeting. Table 32 describes the results. We make two observations. First Sunlight's hypothesis precision against ground truth is 100% (with a Logit Stage 1) on both Amazon and YouTube, while XRay's best algorithm reaches only 81% and 93% respectively. This confirms Sunlight's high hypothesis precision that makes a difference even on simple cases.

Second hypothesis recall is higher for XRay. The Bayesian algorithm reaches 68% on YouTube and 78% on Amazon while Logit yields 46% and 52% respectively. This can be explained by the small size of these datasets: when faced with little evidence, Sunlight will return no hypothesis or low confidence hypotheses, favoring precision over recall compared to XRay's algorithms. We believe this is a valuable trade-off when performing large scale experiments. In the absence of ground truth, we need to be able to trust targeting hypotheses even at the cost of some recall.

This confirms Sunlight's focus on precision over recall on datasets with ground truth. We next study more complex targeting with inputs on redundant topics, but that do not provide ground truth.



(a) **Scalability with Input Size** (b) **Recall with Input Size** (c) **Precision with Input Size**

Fig. 38: **Scalability.** (a) Number of accounts required to achieve the knee accuracy for varied numbers of inputs. (b), (c) Recall/precision achievable with the number of accounts in (a).

3.6.4 Q2: Scalability with input size

A main contribution of this work is the realization that, under certain assumptions, the number of accounts needed to achieve high accuracy for Sunlight scales logarithmically with the number of tracked inputs. We have proven that under certain assumptions, the Set Intersection and sparse regression algorithms scale logarithmically. This theoretical result is hard to extend to the Bayesian algorithm, so we evaluated it experimentally by studying three metrics with growing input size: the number of accounts required to reach the recall knee and the value of recall/precision at this knee. Figures 38a, 38b and 38c show the corresponding results for Gmail-simple, YouTube and Amazon. For Gmail-simple, the number of accounts necessary to reach the knee increased less than 3-fold (from 8 to 21) as input size increased more than 25-fold (from 2 to 51). For Amazon and YouTube, the increases in accounts were 6- and 8-fold respectively, for a 32-fold increase in input size. In general, the roughly linear shapes of the log-x-scale graphs in Fig. 38a confirm the logarithmic increase in the number of accounts required to handle different inputs. Fig. 38b and 38c confirm that the "knee number" of accounts (number of accounts above which precision and recall are flat) achieved high recall and precision (over 80%).

What accounts for the large gap between the number of accounts needed for high accuracy in Gmail-simple versus Amazon? For example, tracking a mere two emails in Gmail-simple required 8 accounts, while tracking two viewed products in Amazon needed 2 accounts. The distinction corresponds to the difference in coverage exhibited by the two services. In Gmail-simple, a targeted ad was typically seen in a smaller fraction of the relevant accounts compared to a recommended

product in Amazon. Sunlight adapted its parameters to lower coverage automatically, but it needed more accounts to do so.

Overall, these results confirm that our theoretical scalability results hold for real-world systems, at least for carefully crafted, non-overlapping input workloads. We next study more complex scenarios with more realistic workloads and measures of statistical significance. In this case, because we cannot get ground truth, we use our proxy metrics of ad prediction precision/recall and algorithm coverage.

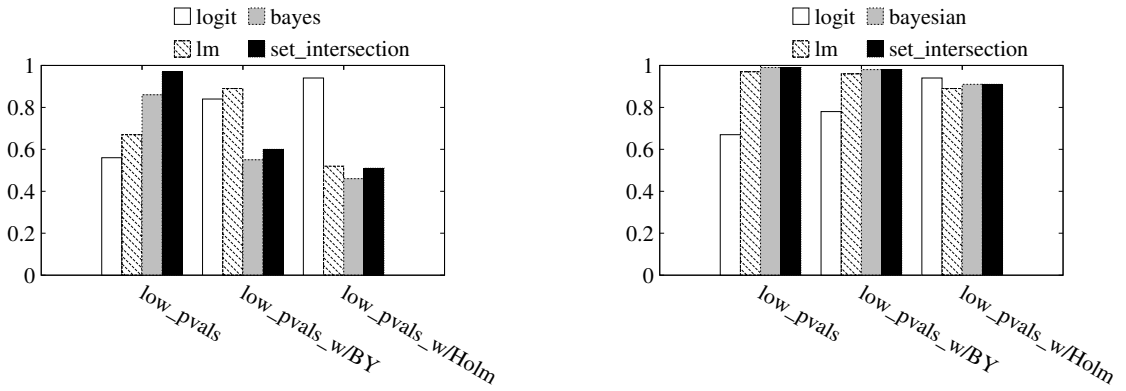
3.6.5 Q3: Evaluating the analysis pipeline

We now look inside the analysis pipeline to measure the effects of its different stages and to compare stage 1 algorithms. In order to measure each algorithm's performance we use its *ad prediction precision and recall* described in §3.6.1. Intuitively if the algorithms detect targeting, they can predict where the ads will be seen in the testing set. Because ads do not always appear in *all* accounts that have the targeted inputs, we do not expect precision to always be 100%. On the other hand, a targeting hypothesis formed using many inputs may easily yield high recall.

Fig. 36 shows the precision and recall of those predictions on the Gmail and Website datasets, first on all hypotheses and then after selecting higher and higher confidence hypotheses. We make three observations. First, the precision is poor if we take every hypotheses into account (see group labeled all_hyps). Precision is below 80% for both datasets, and even less than 60% for most algorithms. Restricting to just the low p-value hypotheses (without correction) somewhat increases ad presence precision (low_pvals group).

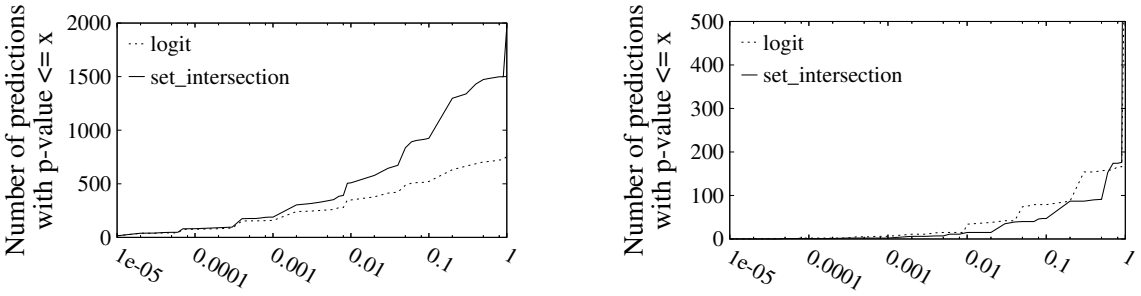
Second, correcting the p-values for multiple testing increases precision as well as recall. The best algorithms on the Gmail and Website datasets, respectively, reach a precision of 90% and 84% after BY correction, and 93% and 91% after Holm correction (low_pvals_w/BY and low_pvals_w/Holm groups). The precision is higher when with Holm because it is more conservative than BY.

Third, the differences introduced by Stage 1 algorithms are reduced by filtering out low-confidence hypotheses. While the precision with all hypotheses (all_hyps group) can vary of up to



(a) **Coverage, Gmail** (b) **Coverage, Websites**

Fig. 39: **Coverage.** Proportion of ads each algorithm found, out of all ads found by at least one algorithm.

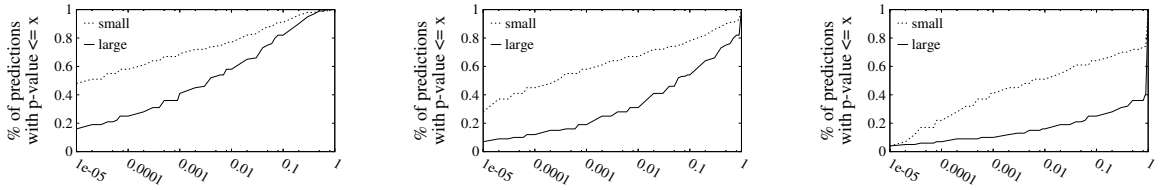


(a) **P-value CDF (Gmail)** (b) **Same as (a) with Holm**

Fig. 310: **Effect of p-value correction on the distribution.** The Set Intersection algorithm makes much more hypothesis, and thus has more low p-value hypothesis. After Holm's correction however, the Logit algorithm has more low p-value hypothesis. X is log scale.

40 percentage points, different Stage 1 algorithms vary only by 1 or 2 percentage points after Holm correction (low_pvals_w/Holm group). The exception is with the BY correction (low_pvals_w/BY group), where the precision of Logit is noticeably higher than that of the other algorithms on the Website dataset.

Thus, when selecting only high-confidence hypotheses, Sunlight is able to predict the presence of an ad with high precision and recall. Moreover, all Stage 1 algorithms generally yield accurate high-confidence hypotheses, which suggests that we should maximize the number of hypotheses. We next compare the number of high-confidence hypotheses and how it is affected by correction.



(a) **P-value distribution** (b) **P-value dist. after BY correction** (c) **P-value dist. after Holm correction**

Fig. 311: **Effect of p-value correction on the distribution, at different scales.** In this graph, the scale is regarding the number of ads. For our small and large scale Website datasets, for the Logit Stage 1, (a) shows the p-value distribution, (b) the p-value distribution after BY correction, and (c) the p-value distribution after Holm correction. The higher number of hypotheses in the large experiment widens the difference in distributions after correction.

3.6.6 Q4: The effect of p-value correction

Maximizing the number of high-confidence hypotheses is maximizing *coverage* (see §3.6.1), the proportion of all high-confidence hypotheses found by a given algorithm. Fig. 39 shows for each Stage 1 algorithm the coverage on the Gmail and Website datasets for low p-values, before and after correction. Set Intersection outputs the most low p-value hypotheses (low_pvals group), but we saw in Fig. 36a-36c that these hypotheses are poor predictors of the ad presence, with a precision below 50%. After the strictest correction (low_pvals_w/Holm), when all hypotheses have similar predictive power, the Logit Stage 1 gives the best coverage, with 93% on Gmail and 94% on Website, beating Lm, the Bayesian algorithm, and Set Intersection. We can make the same conclusion on Gmail after BY correction, but the picture is not as clear on the Websites dataset, where Logit has a lower coverage (about 80%) but makes hypotheses with a better ad prediction precision (see Fig. 36c).

It is interesting to understand why Set Intersection has a much better coverage before correction, but loses this edge to Logit after p-value correction. This can be explained by the fact that the number of hypotheses, and the proportion of high p-value hypotheses play an important role in the correction, both increasing the penalty applied to each p-value. To further demonstrate this effect, Fig. 310 shows the CDF for the distribution of the absolute number of hypotheses per p-value for Logit and Set Intersection. On Fig. 310a we observe that the Set Intersection Stage 1 algo-

rithm makes more low p-value hypotheses with 836 hypothesis below 5%, and only 486 for Logit. However we can also see that the total number of hypothesis is much higher (3282 compared to 928), and that a lot of these hypotheses have a high p-value. After Holm correction however, the Logit algorithm retains 80% more low p-value hypotheses, as shown on Fig. 310b. BY correction shows the same trend, although with less extreme results, explaining why on the Websites dataset Set Intersection keeps a higher coverage.

We show that making many hypotheses hurts coverage after p-value correction, particularly with the Holm method. This calls for algorithms that favor small number of high quality hypotheses, such as Logit, over algorithms that make hypotheses even on weak signal, such as Set Intersection. Making only a small number of low p-value hypotheses is not always possible however, especially when scaling experiments, which we evaluate next.

3.6.7 Q5: Confidence at scale

We saw in §3.6.6 that favoring fewer, higher quality hypothesis is a winning strategy to retain low p-value hypothesis after correction. Unfortunately this is not always possible when scaling the number of outputs, for instance when experiments have more inputs, more profiles, or just when we collected data for a longer period. In these cases analysis may also be harder, leading to fewer good p-value hypotheses. Fig. 311 shows a CDF of the proportion of inputs below a given p-value, and the same after the two p-value correction techniques we consider. We make three observations. First, it appears that hypotheses are indeed harder to make on the large dataset. Fig. 311a shows that the proportion of low p-values is lower even before correction, with 75% of hypotheses with a p-value below 5% for the Website-large, and 88% for Website.

Second, Fig. 311c shows that the Holm correction greatly reduces the proportion of low p-values, with the Website experiment going from 88% to 61%. The effect on the many hypotheses of Website-large is much stronger however, with the low p-values dropping from 75% to 21%. We conclude that the Holm correction is very hard on experiments with a lot hypotheses. The larger the proverbial haystack, the harder the needles will be to find.

Third, the BY correction method is milder than Holm's. Even though we can see the large experiment's distribution caving more than the small's, the distinction at scale is smaller. Website-large still has 46% of its p-values below 5%, while the small one has 74%. Despite the weaker guarantees of the BY correction, it can be a useful trade-off to make when dealing with large numbers of outputs. Indeed, it is a well-accepted correction in statistics and machine learning. We hence include it as a default in our system.

3.6.8 Anecdotal experience with the data

We already saw that high confidence hypotheses give good predictors of the profiles in which an ad will appear. While this *ad prediction precision and recall* reveals that our algorithms are indeed detecting a correlation, we also manually looked at many hypotheses to understand the strengths and weaknesses of our methods. We next describe the results of this experience on large, complex datasets from Gmail and Websites experiments. These services do not provide ground truth at this granularity, so we manually assessed the hypotheses' validity. For this manual assessment we looked at low p-value hypotheses, visited the website targeted by the ad, and decided if it made sense for the website to target the ad. If we did not find a connection for at least one email in the targeting combination, we declared the hypothesis wrong. For instance an ad for a ski resort targeting the "Ski" email was considered right, but the same ad targeting "Ski" and "Cars" was considered a mistake. This labelling is very error prone. On the one hand some associations can be non obvious but still be a real targeting. On the other hand it can be easy to convince ourselves that an association makes sense even when there is no targeting. It is thus an anecdotal experience of Sunlight's performance.

Hypothesis Precision. We examined 100 ads with high-confidence hypotheses (p-value < 0.05 after Holm) from the Gmail and Website experiments, and counted instances where we could not explain the input/output association with high certainty. We found precisions of 95% and 96%, respectively. Hypotheses we classified as false positives were associations of ads and emails that we just could not explain from the topics, such as the luggage company "www.eaglecreek.com" targeting the "Cheap drugs online order" email from Figure 312. In this specific case the ad appears

for 3 consecutive days, multiple times (a total of 437 impressions in 15 to 19 different accounts), and almost only in the context of the email, so there does seem to be some targeting, although we cannot semantically explain it. However, in other cases we have less data to gain confidence, and we classify them as a mistake. This example highlights the difficulty and the risk of bias of manual labelling.

Many errors were also combinations with one input that seemed relevant, and other inputs that did not. Intuitively this happens if the targeting detection algorithm adds any other input to a heavily targeted input, because in the specific *training set* this other input correlates with the output (this is a case of over-fitting in the training set). If for instance the ad appears in 10 profiles in the *testing set*, all with the relevant input, and the inputs are assigned to profiles with a 20% probability, the p-value should be $1.0e^{-7}$ with only the relevant input. With the second input added the p-value becomes higher since a combination is more likely to cover profiles. However the new p-value is still $3.6e^{-5}$ for two inputs, which will remain below the 5% accepted error rate after correction.

Hypothesis Recall. Assessing hypothesis recall based on manual inspection is even more challenging than assessing hypothesis precision. First, there are many more ads to analyze. Second, finding an input among the hundreds we have that is very likely to be targeted is challenging, and the many possibilities make it very easy to invent connections where there is none. For this reason we did not try to quantify hypothesis recall. Instead, we studied low p-value hypotheses that are rejected after correction, a more amenable method that gives information into how many hypotheses we lose due to correction. In our experience, this happens mostly if an ad does not appear enough: the p-value cannot be low enough to be below 5% after correction. For instance if the ad appears in 10 profiles, it will be in about 3 profiles of the testing set, and the p-value cannot be below 0.008 if the inputs are assigned with a 20% probability. After correction this will most likely be over the 5% threshold on big experiments.

This anecdotal experience qualitatively confirms Sunlight's high hypothesis precision on sizeable datasets. It also confirms that manual labelling is unreliable. This is why we conducted our

rigorous evaluation with the five objective metrics described in §3.6.1. More importantly this experience emphasizes the importance of focusing on quality hypotheses when analyzing a large number of outputs. Indeed, the correction will reject all reasonable hypotheses without a lot of data when the number of hypotheses is too high.

3.6.9 Evaluation Summary

We show that Sunlight's high-confidence hypotheses have a good ad prediction precision and recall after p-value correction. This empirically confirms the need to correct for multiple hypothesis testing. The BY correction seems to reach a good trade-off between statistical guarantees and number of low p-value hypotheses.

More surprisingly, we also show an inversion of recall after correction, where algorithms that make fewer, more precise hypotheses end up with better coverage after correction. This makes the case for algorithms that favor precision even at the cost of some recall. Even with such algorithms, recall can become lower after correction when scaling the number of outputs. This represents a fundamental scale/confidence trade-off in targeting detection.

To showcase Sunlight, we also ran larger measurement experiments, which we describe next.

3.6.10 (Q6) Measurement

To showcase Sunlight, we explored targeting in two ad ecosystems with two experiments, on Gmail ads and ads on the web respectively. We used the datasets generated from these experiments for two purposes: (1) to evaluate Sunlight and compare its performance to prior art's (§3.6) and (2) to study a number of interesting aspects about targeting in these ecosystems (§3.6.10). As foreshadowing for our results, both experiments revealed contradictions of separate statements from Google policies or official FAQs. While our use cases refer exclusively to ad targeting detection, we stress that our method is general and (intuitively) should be applicable to other forms of targeting and personalization (e.g., §3.6.3 shows its effectiveness on Amazon's and YouTube's recommendation systems).

	email subject & text	ads Title, url & text	Results
Race	**Dominican** dominican [...] **(OR)** **Hair** hair cut hair cut	**Shampoo JC** www.shampoojc.com Professional Coloring, Highlights. Make your appointment now!	p-value = 0.0004 1427 impressions in 44 profiles 93% in context
Religion & Spirituality	**Mormon** mormon mormon	**Family History Search** genealogy.com/Family+History 1) Simply enter their name. 2) View their family history now!	p-value = 0.001 237 impressions in 18 profiles 74% in context
	Muslim muslim muslim	**Fine Models & Miniatures Shop** www.1stdibs.com/Modern Our Unique Modern Collection Rare Items Added Every Week.	p-value = 0.007 59 impressions in 11 profiles 100% in context
	Jewish jewish jewish	**Free Ancestor Search** archives.com Looking for Your Family Ancestry? Search for Free [...]	p-value = 0.0007 287 impressions in 15 profiles 100% in context
	Buddhist buddhist buddhist	**Berkshire Retreat** www.eastover.com Holistic Retreat Center Personal Retreat	p-value = 0.008 190 impressions in 17 profiles 43% in context
	Guru guru spiritual guide **(OR)** **Astrology** astrology psychic mystical	**What is Quantum Jumping?** www.quantumjumping.com Discover How Thousands of People are Jumping to Change Their Life [...]	p-value = 0.001 244 impressions In 34 profiles 76% in context
Sexual Orient.	**Gay** gay homosexual lesbian gay [...]	**Men Underwear/Workout** www.gosoftwear.com Underwear, Swimwear Go Natural American [...]	p-value = 0.05 54 impressions in 19 profiles 74% in context
General Health	**Affordable** afforable care [...] **(OR)** **Nursing** nursing home [...]	**Illinois Senior Living** www.cottagesofnewlenox.com Assisted Living for Seniors in New Lenox [...]	p-value = 0.03 103 impressions in 36 profiles 28% in context
	Alzheimer Alzheimer Alzheimer	**1/3 of Seniors 65+ Fall** jacuzzi-walk-in-tubs.com/Safety Help Eliminate the Fear of Falling in the Bathroom [...]	p-value = 0.01 21 impressions in 8 profiles 100% in context
	Depressed depression **(OR)** **Anxious** anxious anxiety	**Is He A Cheater?** spokeo.com/Cheating-Spouse-Search Enter His Email Address. Find Pics & Profiles From 70+ Social Networks.	p-value = 0.03 1179 impressions in 52 profiles 20% in context
	Cancer advice How did you cope with cancer in your familly? What an aweful disease!	**The Business of Wellness** healthmediagroup.blogspot.com What my doctor can learn from my Shoe Shine Man [...]	p-value = 0.04 380 impressions in 28 profiles 91% in context
Prohibited	**counterfeit, counterfeit** counterfeit counterfeit	**A&A Global Industries** aaglobal.com Largest Supplier to Bulk Industry Toys, Equipment, Candy, Supplies	p-value = 0.002 66 impressions in 17 profiles 100% in context
	drugs drugs cheap online order	**Eagle Creek Luggage** www.eaglecreek.com/ Extremely Tough & Durable Gear. Luggage, Organizers, Duffels & More	p-value = 0.03 214 impressions in 19 profiles 99% in context
Misc.	**Deregulation** deregulation [...] **(OR)** **Financial Reform** financial reform [...]	**Compliance Audit** unifiedcompliance.com Checklist All IT Compliance You Need to Track In [...]	p-value = 0.0008 1582 impressions in 36 accounts 61% in context
	Unemployed lazy unemployed	**Easy Auto Financing** www.midsouthautoloans.com Need a quick car loan? We work with credit issues	p-value = 0.006 161 impressions in 24 profiles 8% in context
	Payday payday loan	**Fast Cash Loan Online.** www.checkintocash.com Apply Now. Takes Only 5 Minutes. It's as Easy as 1,2,3.	p-value = 0.007 198 impressions in 10 profiles 6% in context
	Veterans war veteran veterans	**Veterans Care Costa Rica** www.veteranscarecostarica.com Receive your proper medical care Tricare, VA, Champ VA	p-value = 0.0006 490 impressions in 15 profiles 84% in context

Fig. 312: **Sample targeted ads from the 33-day Gmail experiment.**

Gmail Ads

As a first example of personal data use, we turn to Gmail which, until November last year, offered personalized advertisements tailored to a user's email content. We selectively placed more

than 300 emails containing single keywords or short phrases to encode a variety of topics, including commercial products (e.g. TV, cars, clothes) and sensitive topics (e.g., religion, sexual orientation, health) into 119 profiles. The emails were manually written by us by selecting topics and writing keywords related to this topic. The first column of Figure 312 shows examples of emails we used. The topics were selected from the AdSense categories [70], with other sensitive forbidden by the AdWords policies [71].

The profiles were Gmail accounts created specifically to study Gmail targeting. Because creating Gmail accounts is costly, some accounts were reused from previous studies, and already contained some emails. The emails relevant to this study were different, and assigned independently from previous emails, so our statistical guaranties still hold. To perform the independent assignment each email was sent to each account with a given probability (in this case 0.2). Emails were sent from 30 other Gmail accounts that did not otherwise take part in the study. No account from the study sent an email to another account of the study. Finally we collected targeted ads by calling Google's advertising endpoints the same way Gmail does, looping over each email and account ten times.

Our goal was to study (1) various aspects related to targeted advertisements, such as how frequent they are and how often they appear in the context of the email being targeted (a more obvious form of targeting) versus in the context of another email (a more obscure form of targeting) and (2) whether advertisers are able to target their ads to sensitive situations or special groups defined by race, religion etc. We collected targeting data for 33 days, from Oct. 8 to Nov. 10, 2014 when Google abruptly shut down Gmail ads. One might say that we have the last month of Gmail ads.[3]

Before Google disabled Gmail ads, we collected 24,961,698 impressions created collectively by 19,543 unique ads. As expected, the distribution of impressions per ad is skewed: the median ads were observed 22 times, while the top 25/5/1% of ads were observed 217/4,417/20,516 times. We classify an ad as *targeted* if its statistical confidence is high (corrected p-value< 0.05 with

[3]Gmail now has email "promotions;" we did not study those.

Sunlight's default pipeline). In our experiment, 2890 unique ads (15% of all) were classified as targeted. While we observe that ads classified as targeted are seen more often (1159 impressions for the median targeted ads), this could be an artifact as most ads seen only occasionally present insufficient evidence to form hypotheses.

Figure 312 shows some examples of ads Sunlight identified as targeted, along with the content of the emails they targeted, the corrected p-values, and information about the context where the impressions appeared. Some ads show targeting on single inputs while others show targeting on combinations of emails. We selected these examples by looking at all ads that were detected as targeting the sensitive emails we constructed, and choosing representative ones. When multiple interesting examples were available, we chose those with a lot of data, or that we detected across multiple days, as we are more confident in them.

Notably, the examples show that information about a user's health, race, religious affiliation or religious interest, sexual orientation, or difficult financial situation, all generate targeted advertisements. Our system cannot assign intention of either advertisers or Google for the targeting we found, but this appears to contradict a statement in an official-looking Gmail Help page: [4]

> Only ads classified as Family-Safe are displayed in Gmail. We are careful about the types of content we serve ads against. For example, Google may block certain ads from running next to an email about catastrophic news. We will also not target ads based on sensitive information, such as *race*, *religion*, *sexual orientation*, *health*, or *sensitive financial categories*.
>
> – `support.google.com/mail/answer/6603`.

While our results do *not* imply that this targeting was intentional or explicitly chosen by any party involved (Google, advertisers, etc.), we believe they demonstrate the need for investigations like the ones Sunlight supports. We also point out that those violations are needles in a haystack. Several topics we included in our experiment (e.g., fatal diseases and loss) generated not a single ad classified as targeted.

[4]The page containing this statement used to be accessible through a user's own account (Gmail - Help - Security & privacy - Privacy policies) and its look and feel until 12/24/2014 was more official than it currently is. The 2014 version is available on archive.org (`https://web.archive.org/web/20141224113252/https://support.google.com/mail/answer/6603`).

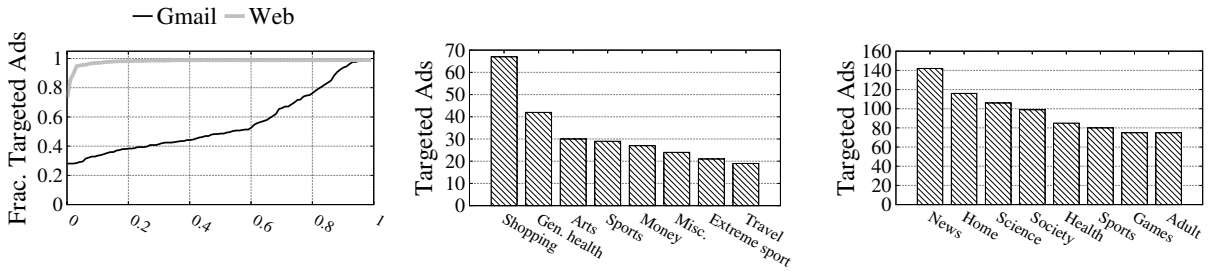
	Targeted website	ads Title & text	Results
Drugs	drugs.com	Nasalcrom Proven to Prevent Nasal Allergy Symptoms	p-value = 2.5e-5 374 impressions in 73 profiles 41% in context
	hightimes.com	AquaLab Technologies Bongs, Pipes, and Smoke Accessories	p-value = 2.6e-13 1714 impressions in 76 profiles 99% in context
News	foxnews.com	IsraelBonds.com Invest in Israel	p-value = 0.0041 71 impression in 45 accounts 100% in context
	huffingtonpost.com	Stop The Tea Party Support Patrick Murphy	p-value = 0.010 97 impressions in 37 profiles 100% in context
	economist.com	The Economist Great Minds Like a Think - Introductory Offer	p-value = 0.00066 151 impressions in 77 profiles 0% in context
Misc.	pcgamer.com (games)	Advanced PCs Digital Storm Starting at $699	p-value = 0.035 575 impressions in 129 profiles 66% in context
	soberrecovery.com (rehab)	Elite Rehab Speak w/ a counselor now	p-value = 6.8e-6 5486 impressions 82 profiles 99% in context

Fig. 313: **Sample targeted ads from the display-ads experiment (also called Website experiment).**

§3.6.10 presents further results about targeting on Gmail.

Display Ads on the Web

As a second example of personal data use, we look at targeting of arbitrary ads on the web on users' browsing histories. This experiment is not specifically related to Google, though Google is one of the major ad networks that serve the ads we collect. Similar to the Gmail use case, our goal is to study aspects such as frequency of targeted ad impressions, how often they appear in the context of the website being targeted versus outside, and whether evidence of targeting on sensitive websites (e.g., health, support groups, etc.) exists. We populate 200 browsing profiles with 200 input sites chosen randomly from the top 40 sites across 16 different Alexa categories, such as News, Home, Science, Health, and Children/Teens. Each website is randomly assigned to each profile with a probability 0.5. For each site, we visit the top 10 pages returned from a site-specific search on Google. We use Selenium [170] for browsing automation. We collect ads from the visited pages using a modified version of AdBlockPlus[3] that detects ads instead of blocking them. After collecting data, we use Sunlight's default pipeline and a p-value < 0.05 to assess targeting.



(a) **Targeted Ads Out of Context CDF** (b) **Gmail Category Targeting** (c) **Web Category Targeting**

Fig. 314: **Measurement Results.** (a) Shows what fraction of targeted ads appear how often in their targeted context in both Gmail and Web experiments. (b) Shows the number of ads are found to target emails in the eight most popular categories in the Gmail experiment. (c) Shows the number of ads are found to target sites in the eight most popular categories in the web experiment.

We collect 19,807 distinct ads through 932,612 total impressions. The web display ads we collected skewed to fewer impressions than those we collected in the Gmail experiment. The median ad appears 3 times and we recorded 12/126/584 impressions for the top 25/5/1% of display ads. In this experiment, 931 unique ads (5% of all) were classified as targeted, and collectively they are responsible from 37% of all impressions.

Figure 313 shows a selection of ads from the study, chosen similarly as the ads from the Gmail study. Among the examples are ads targeted on marijuana sites and drug use sites. Many of the ads targeted on drug use sites we saw, such as the "Aqua Lab Technologies" ad, advertise drug paraphernalia and are served from `googlesyndication.com`. This appears to contradict Google's advertising policy, which bans "Products or services marketed as facilitating recreational drug use." – `https://support.google.com/adwordspolicy/answer/6014299`.

We next presents further results about targeting on the web.

Other Targeting Results

Using Sunlight, we found several other interesting aspects about ad targeting in Gmail (when ads were shown next to emails) and on the web. We next describe those aspects as examples of the kinds of things that could be learned with Sunlight. As before, we consider an ad *targeted* if its corrected p-value is < 0.05 under the Sunlight default pipeline. For the results in this section we use the entire website display ad dataset and focus on one day from the Gmail study.

In Context vs. Outside Context. One interesting question one might wonder is how often are ads shown out of the context of the targeted input. Intuitively, if an ad is shown in the email (or on the page) that it targets, its should be more obvious to a user compared to an ad shown with one email (or page) but targeting another email (or page). Figure 314a shows a CDF of how often targeted ads appear in their target context for the Gmail and Website-large datasets. The Y axis represents the fraction of all targeted ads in each experiment.

In Gmail, ads are frequently out of context (i.e., alongside emails that they do not target). Approximately 28% of the Gmail ads labeled as targeted appear only in their targeted context and half of targeted Gmail ads appear out of context 48% of the time or more. Thus there is (or rather, was) heavy behavioral targeting in Gmail.

On the web, display ads are rarely shown outside of their targeted context. 73% of ads are only ever shown on the site targeted by the ad. Of the targeted ads that do appear out of context, the majority of them appear on only 1 or 2 other sites. This suggests a very heavy contextual targeting for display ads. That said, we have found convincing examples of behaviorally targeted ads that appear entirely outside of their targeted context. Included in Figure 313 is an ad for *The Economist* encouraging viewers to subscribe to the publication. That ad *never* appeared on the targeted site. We found similar examples for *The New York Times*.

Targeting Per Category. Figures 314b and 314c show the number of ads targeting emails and websites, respectively in a particular category. For emails, we classify them based on their content. For websites, we use the Alexa categories. It is possible, and common, for Sunlight to detect that an ad targets multiple emails so the cumulative number of guesses represented in the figure may be larger than the total number of ads.

In Gmail, by far the most targeted category (topic) in our dataset was shopping (e.g., emails containing keywords such as clothes, antiques, furniture etc.). The second most popular targeted category was General health (i.e., emails with keywords such as vitamins, yoga, etc.). On the web, we did not observe a single dominant category as we did in Gmail. The News category, containing

sites like The Economist and Market, was targeted by the most ads in the study but with only slightly more ads then the Home category.

Overall, these results demonstrate that Sunlight is valuable not only for investigators but also for researchers interested in broader aspects of targeting.

3.7 Related Work

§3.3.2 already discusses web transparency measurements[76;202;77;135;136;189;11;117;141;22;45;108]. These works aim to quantify various data uses on the web, including targeting, personalization, price tuning, or discrimination. Sunlight is the first system to detect targeting at fine grain (individual inputs), at scale, and with solid statistical justification.

The work closest in spirit to ours[108;110] is AdFisher[45] which aims, like us, to create generic, broadly applicable methodologies for various web transparency goals. AdFisher shares our goal of providing solid statistical justification for its findings, but, because of scale limitations, makes it hard to simultaneously track many inputs. So far it was applied to relatively coarse targeting (e.g., gender, a specific interest). Since Ad-Fisher grounds its confidence in all outputs simultaneously, its results should be carefully interpreted: it rigorously proved that some targeting is taking place, but does not exhaustively and separately single out the output subject to this targeting. Finally this design disregards scalability with the number of inputs: the effect of each input and each possible combination of inputs needs to be tested separately.

Our methods for statistical experimental design and analysis draw from the subjects of *compressed sensing*[49] and *sparse regression*[183;15]. The experimental setups we consider correspond to sensing matrices that satisfy certain analytic properties that permit robust recovery of sparse signals. In Sunlight, these signals correspond to the hypothesized targeting effects we subsequently test and validate, and they are sparse when the targeting effects only depend on a few variables.

3.8 Summary

This chapter argues for the need for scalable and statistically rigorous methodologies, plus infrastructures that implement them, to shed light over today's opaque online data ecosystem. We

have presented one such methodology, developed in the context of *Sunlight*, a system designed to detect targeting at fine granularity, at scale, and with statistical justification for all its inferences. The Sunlight methodology consists of a four-stage pipeline, which gradually generates, refines, and validates hypotheses to reveal the likely causes of observed targeting. Sunlight implements this methodology in a modular way, allowing for broad explorations and evaluation of the design space. Our own exploration reveals an interesting trade-off between the statistical confidence and the number of targeting hypotheses that can be made. Our empirical study of this effect suggests that favoring high precision hypothesis generation can yield better recall at high confidence at the end of the Sunlight pipeline, and that scaling the number of outputs of an experiment may require to accept lower statistical semantics.

In our opinion, a crucial avenue for future work will be to leverage causal inference methods (see[166;152] for different approaches), and in particular recent advances on causal inference on observational data, such as[192]. This would break two major limitations of current measurements. First, causal results from randomized experiments are valid only for the studied population. Formulating and testing causal targeting hypotheses on real user accounts would be more broadly applicable, and would probably detect more complex targeting. Second, it would break the scaling barrier by opening a much larger pool of data, which would also be much harder for services to tamper with as the data would come from real user accounts.

Conclusion

In this dissertation we considered the implications of the recent shift to ML-driven applications. This recent ML revolution, powered by new systems supporting massive amounts of data and advanced algorithms to analyse them, is transforming a wide range of applications. However, it also poses substantial security, privacy, and transparency concerns, creating serious barriers to reaching ML's full potential. ML workloads push companies toward aggressive data collection and loose data access policies, placing troves of user information in danger if the company is hacked. ML also introduces new attack vectors, which can completely nullify models' accuracy under attack. Finally, ML models make complex, data-driven decisions that are opaque for the end-users and difficult to inspect for programmers.

Through examples of tools and systems we developed to address one aspect of each challenge, we showed how an approach combining systems design and theory from different domains, from privacy to statistics and causal inference, can enable ML systems with strong semantics of security, privacy, and transparency, offering provable guarantees while remaining practical. Components requiring strong guarantees, typically those enforcing security, privacy, or interpretability semantics, are separated from those that can do with best-effort operation. Components enforcing rigorous guarantees are matched to the theoretical field best suited as a foundation for a solution. Best-effort components leverage heuristics approaches, enabling practical and scalable designs.

In Chapter 1 we argued that disseminating ML models trained over sensitive data to untrusted domains, a common practice in ML applications, is an access control issue. Indeed, those models leak the data they were trained on, in a sense giving an irrevocable capability to the training data. Based on the fact that differential privacy (DP), a theory from the privacy domain, protects

each single observation in the training set from leakage we built Sage, the first ML platform that enforces a global event level DP guarantee across all models released from sensitive data streams. The components enforcing DP and testing the accuracy of the trained models before release require strong guarantees for their semantics. To enable these semantics, we leveraged and adapted existing theory from differential privacy and statistics, and proposed a block-level accounting technique that permits endless operation of ML workloads on growing databases. Best-effort components include resource allocation for data and privacy budget, an iterative training method to find good performing models while conserving these resources, and a component leveraging training set minimization techniques to compute DP summaries of past data. These summaries can be used as surrogates to the data, to conserve their privacy budget or benefit from their information even after their privacy budget is exhausted.

In Chapter 2 we proposed a new defense strategy against adversarial examples, an ML-specific class of attacks in which the adversary finds a small perturbation to an input which results in a prediction of the adversary's choosing. To enforce strong security semantics, namely give for each prediction a size bellow which no adversarial attack exists, we leverage the stability properties of differential privacy in a novel way. We also use theory from statistics. Training the model on the other hand is a best effort approach, which empirically maximizes the accuracy of our robust model on a training set. This approach yields the first certified defense that both scales to large networks and datasets (such as Google's Inception network for ImageNet) and applies broadly to arbitrary model types. It not only introduces scalable algorithms for learning more robust models with probable guarantees, but also enables a firewall-like security architecture, where a small model is prepended to an existing, already trained one to make it more robust. Such an architecture is common in traditional software systems but unique for ML workloads.

In Chapter 3 we presented transparency tools that provide a new visibility into how ML-driven web services use end-users' data for targeting. The methodology relies on a multi-stage pipeline handling the experiment design and the data analysis. This pipeline has three components enforcing important semantics. First, the experiment design component leverages theory from causal

inference and randomized experiments to provide causal semantics: when we detect a reaction to user data (e.g. targeting, personalization), the data caused the change, and was indeed directly used by the Web service. Second, the targeting hypothesis formulations component provides algorithms with strong theoretical scalability properties, at least under sparsity assumptions. Third, the hypothesis testing component uses theory form statistics to give sound measures of statistical significance, which express how (un)likely a result is to have happened by mere chance. Integrating these components in a practical architecture raises significant challenges, that we address with heuristic components. For instance, a best-effort hypothesis filtering component ensures that we have enough statistical power to validate hypotheses, giving output size scalability without compromising other components' guarantees.

To conclude, we presented three research directions to build guarantees of security, privacy, and transparency in ML systems. More generally, we hope that this work will show the value of designing mechanisms that leverage theory domains not traditionally associated with systems, such as statistics, causal inference, or differential privacy. With this approach, we can integrate ML into systems while enforcing strong semantics system designers can rely on. Other applications, such as learning decision policies or resource allocation, could benefit from such semantics.

Bibliography

[1] Abadi, M., Chu, A., Goodfellow, I., Brendan McMahan, H., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep Learning with Differential Privacy. *ArXiv*.

[2] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)*.

[3] AdBlockPlus (2015). `https://adblockplus.org/`.

[4] Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. (2018). Sanity checks for saliency maps. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*.

[5] Agresti, A. (2013). *Categorical Data Analysis*. Wiley Series in Probability and Statistics. Wiley.

[6] Athalye, A., Carlini, N., and Wagner, D. (2018a). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proc. of the International Conference on Machine Learning Research (ICML)*.

[7] Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. (2018b). Synthesizing robust adversarial examples. In *Proc. of the International Conference on Machine Learning Research (ICML)*.

[8] AzureML (2016). Build counting transform. `https://msdn.microsoft.com/en-us/library/azure/mt243845.aspx`.

[9] Backes, M., Berrang, P., Humbert, M., and Manoharan, P. (2016). Membership privacy in microRNA-based studies. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)*.

[10] Barak, B., Chaudhuri, K., Dwork, C., Kale, S., McSherry, F., and Talwar, K. (2007). Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proc. of the ACM SIGMOD International Conference on Management of Data*.

[11] Barford, P., Canadi, I., Krushevskaja, D., Ma, Q., and Muthukrishnan, S. (2014). Adscape: Harvesting and Analyzing Online Display Ads. *WWW*.

[12] Bassily, R., Nissim, K., Smith, A., Steinke, T., Stemmer, U., and Ullman, J. (2016). Algorithmic stability for adaptive data analysis. In *Proc. of the ACM Symposium on Theory of Computing (STOC)*.

[13] Baylor, D., Breck, E., Cheng, H.-T., Fiedel, N., Foo, C. Y., Haque, Z., Haykal, S., Ispir, M., Jain, V., Koc, L., Koo, C. Y., Lew, L., Mewald, C., Modi, A. N., Polyzotis, N., Ramesh, S., Roy, S., Whang, S. E., Wicke, M., Wilkiewicz, J., Zhang, X., and Zinkevich, M. (2017). TFX: A Tensorflow-based production-scale machine learning platform. In *Proc. of the International Conference on Knowledge Discovery and Data Mining (KDD)*.

[14] Benjamini, Y. and Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. *Annals of statistics*.

[15] Bickel, P. J., Ritov, Y., and Tsybakov, A. B. (2009). Simultaneous analysis of lasso and dantzig selector. *The Annals of Statistics*.

[16] Biggio, B., Nelson, B., and Laskov, P. (2012). Poisoning attacks against support vector machines. In *Proc. of the International Conference on Machine Learning Research (ICML)*.

[17] Biggio, B., Pillai, I., Rota Bulò, S., Ariu, D., Pelillo, M., and Roli, F. (2013). Is data clustering in adversarial settings secure? In *Proc. of the ACM Workshop on Artificial Intelligence and Security*.

[18] Bird, S., Barocas, S., Crawford, K., Diaz, F., and Wallach, H. (2016). Exploring or exploiting? social and ethical implications of autonomous experimentation in ai. In *Workshop on Fairness, Accountability, and Transparency in Machine Learning*.

[19] Bodik, P., Goldszmidt, M., Fox, A., Woodard, D. B., and Andersen, H. (2010). Fingerprinting the datacenter: Automated classification of performance crises. In *Proc. of the ACM European Conference on Computer Systems (EuroSys)*.

[20] Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. (2016). End to end learning for self-driving cars. *arXiv*.

[21] Bolukbasi, T., Chang, K.-W., Zou, J. Y., Saligrama, V., and Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*.

[22] Book, T. and Wallach, D. S. (2015). An Empirical Study of Mobile Ad Targeting. *arXiv.org*.

[23] Boyd, K., Lantz, E., and Page, D. (2015). Differential privacy for classifier evaluation. In *Proc. of the ACM Workshop on Artificial Intelligence and Security*.

[24] Brandeis, L. (1913). What Publicity Can Do. *Harper's Weekly*.

[25] Buolamwini, J. and Gebru, T. (2018). Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*.

[26] Calandrino, J. A., Kilzer, A., Narayanan, A., Felten, E. W., and Shmatikov, V. (2011). "You Might Also Like:" Privacy risks of collaborative filtering. In *Proc. of IEEE Symposium on Security and Privacy (S&P)*.

[27] Caliskan, A., Bryson, J. J., and Narayanan, A. (2017). Semantics derived automatically from language corpora contain human-like biases. *Science*.

[28] Carlini, N., Liu, C., Erlingsson, U., Kos, J., and Song, D. (2018). The secret sharer: Evaluating and testing unintended memorization in neural networks. arXiv.

[29] Carlini, N. and Wagner, D. (2017a). Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proc. of the ACM Workshop on Artificial Intelligence and Security*.

[30] Carlini, N. and Wagner, D. A. (2017b). Towards evaluating the robustness of neural networks. In *Proc. of IEEE Symposium on Security and Privacy (S&P)*.

[31] Chan, T.-H. H., Shi, E., and Song, D. (2011). Private and continual release of statistics. *ACM Transactions on Information Systems Security*.

[32] Charikar, M., Chen, K., and Farach-Colton, M. (2002). Finding frequent items in data streams. In *Proc. of the International Colloquium on Automata, Languages and Programming*.

[33] Chatzikokolakis, K., Andrés, M. E., Bordenabe, N. E., and Palamidessi, C. (2013). Broadening the scope of differential privacy using metrics. In *Proc. of the International Symposium on Privacy Enhancing Technologies Symposium*.

[34] Chaudhuri, K. and Monteleoni, C. (2008). Privacy-preserving logistic regression. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*.

[35] Chaudhuri, K., Monteleoni, C., and Sarwate, A. D. (2011). Differentially private empirical risk minimization. *Journal of Machine Learning Reserach*.

[36] Chaudhuri, K., Sarwate, A. D., and Sinha, K. (2013). A near-optimal algorithm for differentially-private principal components. *Journal of Machine Learning Reserach*.

[37] Chen, I., Johansson, F. D., and Sontag, D. (2018). Why is my classifier discriminatory? In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*.

[38] Chuan Guo, Mayank Rana, Moustapha Cisse, Laurens van der Maaten (2018). Countering adversarial images using input transformations. *Proc. of the International Conference on Learning Representations (ICLR)*.

[39] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, Alan Yuille (2018). Mitigating adversarial effects through randomization. *Proc. of the International Conference on Learning Representations (ICLR)*.

[40] Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. (2017). Parseval networks: Improving robustness to adversarial examples. In *Proc. of the International Conference on Machine Learning Research (ICML)*.

[41] Cohen, J. M., Rosenfeld, E., and Kolter, J. Z. (2019). Certified adversarial robustness via randomized smoothing. In *Proc. of the International Conference on Machine Learning Research (ICML)*.

[42] Cormode, G. and Muthukrishnan, S. (2005). An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*.

[43] Criteo (2015). Criteo releases its new dataset. `http://labs.criteo.com/2015/03/criteo-releases-its-new-dataset/`.

[44] Cummings, R., Krehbiel, S., Lai, K. A., and Tantipongpipat, U. (2018). Differential privacy for growing databases. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*.

[45] Datta, A., Tschantz, M. C., and Datta, A. (2015). Automated Experiments on Ad Privacy Settings. In *Proceedings of Privacy Enhancing Technologies*.

[46] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[47] Dimitrakakis, C., Nelson, B., Mitrokotsa, A., and Rubinstein, B. (2016). Bayesian Differential Privacy through Posterior Sampling. *arXiv*.

[48] Dinur, I. and Nissim, K. (2003). Revealing information while preserving privacy. In *Proc. of the International Conference on Principles of Database Systems (PODS)*.

[49] Donoho, D. L. (2006). Compressed sensing. *IEEE Transactions on Information Theory*.

[50] Duchi, J. and Rogers, R. (2019). Lower bounds for locally private estimation via communication complexity. *arXiv*.

[51] Duchi, J. C., Jordan, M. I., and Wainwright, M. J. (2018). Minimax optimal procedures for locally private estimation. *Journal of the American Statistical Association*.

[52] Dudoit, S. and van der Laan, M. (2008). *Multiple testing procedures with applications to genomics*. Springer.

[53] Dutta, S., Jha, S., Sankaranarayanan, S., and Tiwari, A. (2018). Output range analysis for deep feedforward neural networks. In *NASA Formal Methods Symposium*.

[54] Dvijotham, K., Gowal, S., Stanforth, R., Arandjelovic, R., O'Donoghue, B., Uesato, J., and Kohli, P. (2018). Training verified learners with learned verifiers. *ArXiv*.

[55] Dwork, C. (2006). Differential privacy. In *Automata, languages and programming*.

[56] Dwork, C. (2010). Differential privacy in new settings. In *Proc. of the ACM Symposium on Discrete Algorithms (SODA)*.

[57] Dwork, C. and Lei, J. (2009). Differential privacy and robust statistics. In *Proc. of the ACM Symposium on Theory of Computing (STOC)*.

[58] Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *Proc. of the Conference on Theory of Cryptography (TCC)*.

[59] Dwork, C., Naor, M., Pitassi, T., , and Yekhanin, S. (2010a). Pan-private streaming algorithms. In *Proc. of The Symposium on Innovations in Computer Science*.

[60] Dwork, C., Naor, M., Pitassi, T., and Rothblum, G. N. (2010b). Differential privacy under continual observation. In *Proc. of the ACM Symposium on Theory of Computing (STOC)*.

[61] Dwork, C. and Roth, A. (2014a). The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*.

[62] Dwork, C. and Roth, A. (2014b). The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407.

[63] Dwork, C., Rothblum, G. N., and Vadhan, S. (2010c). Boosting and differential privacy. In *Proc. of the IEEE Symposium on Foundations of Computer Science (FOCS)*.

[64] Dwork, C., Smith, A., Steinke, T., and Ullman, J. (2017). Exposed! A survey of attacks on private data. *Annual Review of Statistics and Its Application*.

[65] Dwork, C., Smith, A., Steinke, T., Ullman, J., and Vadhan, S. (2015). Robust traceability from trace amounts. In *Proc. of the IEEE Symposium on Foundations of Computer Science (FOCS)*.

[66] Evtimov, I., Eykholt, K., Fernandes, E., Kohno, T., Li, B., Prakash, A., Rahmati, A., and Song, D. (2017). Robust physical-world attacks on machine learning models. *arXiv*.

[67] Feldman, V. (2009). Optimal hardness results for maximizing agreement with monomials. *SIAM Journal on Computing*.

[68] Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., and Vechev, M. (2018). Ai 2: Safety and robustness certification of neural networks with abstract interpretation. In *Proc. of IEEE Symposium on Security and Privacy (S&P)*.

[69] Goodfellow, I., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *Proc. of the International Conference on Learning Representations (ICLR)*.

[70] Google (2015a). AdSense policy. `https://support.google.com/adsense/answer/3016459?hl=en`.

[71] Google (2015b). AdWords policy. `https://support.google.com/adwordspolicy/answer/6008942?hl=en`.

[72] Google (Accessed: 2018). Inception v3. `https://github.com/tensorflow/models/tree/master/research/inception`.

[73] Gould, J. (2014). SafeGov.org - Google admits data mining student emails in its free education apps.

[74] Gretton, A., Bousquet, O., Smola, A., , and Schölkopf, B. (2005). Measuring statistical dependence with Hilbert-Schmidt norms. In *Algorithmic Learning Theory*.

[75] Guneet S. Dhillon, Kamyar Azizzadenesheli, Jeremy D. Bernstein, Jean Kossaifi, Aran Khanna, Zachary C. Lipton, Animashree Anandkumar (2018). Stochastic activation pruning for robust adversarial defense. *Proc. of the International Conference on Learning Representations (ICLR)*.

[76] Hannak, A., Sapiezynski, P., Kakhki, A. M., Krishnamurthy, B., Lazer, D., Mislove, A., and Wilson, C. (2013). Measuring personalization of web search. In *Proc. of the International World Wide Web Conference (WWW)*.

[77] Hannak, A., Soeller, G., Lazer, D., Mislove, A., and Wilson, C. (2014). Measuring Price Discrimination and Steering on E-commerce Web Sites. In *Proc. of the ACM Internet Measurement Conference (IMC)*.

[78] Harper, F. M. and Konstan, J. A. (2015). The MovieLens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*

[79] Hazelwood, K., Bird, S., Brooks, D., Chintala, S., Diril, U., Dzhulgakov, D., Fawzy, M., Jia, B., Jia, Y., Kalro, A., Law, J., Lee, K., Lu, J., Noordhuis, P., Smelyanskiy, M., Xiong, L., and Wang, X. (2018). Applied machine learning at Facebook: A datacenter infrastructure perspective. In *Proc. of International Symposium on High-Performance Computer Architecture (HPCA)*.

[80] Hendrycks, D. and Gimpel, K. (2017). Early methods for detecting adversarial images. In *ICLR (Workshop Track)*.

[81] Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*.

[82] Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*.

[83] Homer, N., Szelinger, S., Redman, M., Duggan, D., Tembe, W., Muehling, J., Pearson, J. V., Stephan, D. A., Nelson, S. F., and Craig, D. W. (2008). Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS Genetics*.

[84] Huang, X., Kwiatkowska, M., Wang, S., and Wu, M. (2017). Safety verification of deep neural networks. In *Proc. of the International Conference on Computer Aided Verification*.

[85] Ilyas, A., Jalal, A., Asteri, E., Daskalakis, C., and Dimakis, A. G. (2017). The robust manifold defense: Adversarial training using generative models. *arXiv*.

[86] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. of the International Conference on Machine Learning Research (ICML)*.

[87] Jacob Buckman, Aurko Roy, Colin Raffel, Ian Goodfellow (2018). Thermometer encoding: One hot way to resist adversarial examples. *Proc. of the International Conference on Learning Representations (ICLR)*.

[88] jan Kindermans, P., Hooker, S., Adebayo, J., Alber, M., Schütt, K. T., Dähne, S., Erhan, D., and Kim, B. (2017). The (un)reliability of saliency methods. In *NeurIPS Workshop - Interpreting, Explaining and Visualizing Deep Learning - Now what?*

[89] Javier Sánchez-Monedero and Lina Dencik (2019). The datafication of the workplace. https://datajusticeproject.net/wp-content/uploads/sites/30/2019/05/Report-The-datafication-of-the-workplace.pdf.

[90] Jayaraman, B. and Evans, D. (2019). Evaluating differentially private machine learning in practice. In *Proc. of USENIX Security*.

[91] Julia Angwin and Terry Parris Jr. (2016). Facebook Lets Advertisers Exclude Users by Race. https://www.propublica.org/article/facebook-lets-advertisers-exclude-users-by-race.

[92] Justin Volz (2018). Health Insurers Are Vacuuming Up Details About You — And It Could Raise Your Rates. https://www.propublica.org/article/health-insurers-are-vacuuming-up-details-about-you-and-it-could-raise-you

[93] Kaggle (2014a). https://www.kaggle.com/c/criteo-display-ad-challenge/discussion/10429#54591.

[94] Kaggle (2014b). Kaggle display advertising challenge dataset. https://www.kaggle.com/c/criteo-display-ad-challenge.

[95] Kairouz, P., Oh, S., and Viswanath, P. (2015). The composition theorem for differential privacy. In *International Conference on Machine Learning (ICML)*.

[96] Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., and Smith, A. (2011). What can we learn privately? *SIAM Journal on Computing*.

[97] Kasiviswanathan, S. P. and Smith, A. (2014). On the'semantics' of differential privacy: A bayesian formulation. *Journal of Privacy and Confidentiality*.

[98] Katz, G., Barrett, C. W., Dill, D. L., Julian, K., and Kochenderfer, M. J. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. *arXiv*.

[99] Kifer, D., Smith, A., and Thakurta, A. (2012). Private convex empirical risk minimization and high-dimensional regression. In *Proc. of the ACM Conference on Learning Theory (COLT)*.

[100] Kos, J., Fischer, I., and Song, D. (2017). Adversarial examples for generative models. *arXiv*.

[101] Kos, Jernej and Song, Dawn (2017). Delving into adversarial attacks on deep policies. *arXiv*.

[102] Krishnamurthy, B. and Wills, C. E. (2009). On the leakage of personally identifiable information via online social networks. In *Proc. of the ACM Workshop on Online Social Networks*.

[103] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images.

[104] Kurakin, A., Goodfellow, I. J., and Bengio, S. (2016). Adversarial examples in the physical world. *arXiv*.

[105] Langford, J., Li, L., and Strehl, A. (2007). Vowpal Wabbit online learning project.

[106] Lécuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. (2019a). Certified robustness to adversarial examples with differential privacy. In *Proc. of IEEE Symposium on Security and Privacy (S&P)*.

[107] Lécuyer, M., Ducoffe, G., Lan, F., Papancea, A., Petsios, T., Spahn, R., Chaintreau, A., and Geambasu, R. (2014a). XRay: Enhancing the Web's Transparency with Differential Correlation. Technical report, CS Department, Columbia University.

[108] Lécuyer, M., Ducoffe, G., Lan, F., Papancea, A., Petsios, T., Spahn, R., Chaintreau, A., and Geambasu, R. (2014b). XRay: Increasing the web's transparency with differential correlation. In *Proc. of USENIX Security*.

[109] Lécuyer, M., Spahn, R., Geambasu, R., Huang, T.-K., and Sen, S. (2017). Pyramid: Enhancing selectivity in big data protection with count featurization. In *Proc. of IEEE Symposium on Security and Privacy (S&P)*.

[110] Lécuyer, M., Spahn, R., Spiliopoulos, Y., Chaintreau, A., Geambasu, R., and Hsu, D. (2015). Sunlight: Fine-grained targeting detection at scale with statistical confidence. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)*.

[111] Lécuyer, M., Spahn, R., Vodrahalli, K., Geambasu, R., and Hsu, D. (2019b). Privacy accounting and quality control in the sage differentially private ML platform. In *Proc. of the ACM Symposium on Operating Systems Principles (SOSP)*.

[112] Lécuyer, M., Spahn, R., Vodrahalli, K., Geambasu, R., and Hsu, D. (2019c). Privacy accounting and quality control in the sage differentially private ML platform. Online Supplements (also available on arXiv).

[113] Leonard, N. and Halasz, C. M. (2018). Twitter meets tensorflow. `https://blog.twitter.com/engineering/en_us/topics/insights/2018/twittertensorflow.html`.

[114] Li, B., Chen, C., Wang, W., and Carin, L. (2018). Second-order adversarial attack and certifiable robustness. *arXiv*.

[115] Li, B., Wang, Y., Singh, A., and Vorobeychik, Y. (2016). Data poisoning attacks on factorization-based collaborative filtering. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*.

[116] Li, L. E., Chen, E., Hermann, J., Zhang, P., and Wang, L. (2017). Scaling machine learning as a service. In *Proc. of The International Conference on Predictive Applications and APIs*.

[117] Liu, B., Sheth, A., Weinsberg, U., Chandrashekar, J., and Govindan, R. (2013). AdReveal: improving transparency into online targeted advertising. In *Proc. of the Workshop on Hot Topics in Networks (HotNets)*.

[118] Liu, X., Cheng, M., Zhang, H., and Hsieh, C. (2017). Towards robust neural networks via random self-ensemble. Technical report.

[119] Lu, J., Sibai, H., Fabry, E., and Forsyth, D. (2017). No need to worry about adversarial examples in object detection in autonomous vehicles. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[120] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *arXiv*.

[121] Madry Lab (Accessed: 1/22/2017). CIFAR-10 Adversarial Examples Challenge. `https://github.com/MadryLab/cifar10_challenge`.

[122] Mattioli, D. (2012). WSJ.com - On Orbitz, Mac Users Steered to Pricier Hotels.

[123] Maurer, A. and Pontil, M. (2009). Empirical Bernstein Bounds and Sample Variance Penalization.

[124] Maurer, A. and Pontil, M. (2009). Empirical bernstein bounds and sample-variance penalization. In *Proc. of the ACM Conference on Learning Theory (COLT)*.

[125] McMahan, H. B. and Andrew, G. (2018a). A general approach to adding differential privacy to iterative training procedures. *arXiv*.

[126] McMahan, H. B. and Andrew, G. (2018b). A general approach to adding differential privacy to iterative training procedures. *arXiv*.

[127] McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. (2018). Learning differentially private recurrent language models. In *Proc. of the International Conference on Learning Representations (ICLR)*.

[128] McSherry, F. and Mironov, I. (2009a). Differentially private recommender systems: Building privacy into the Netflix prize contenders. In *Proc. of the International Conference on Knowledge Discovery and Data Mining (KDD)*.

[129] McSherry, F. and Mironov, I. (2009b). Differentially private recommender systems: Building privacy into the netflix prize contenders. In *ACM's Special Interest Group on Knowledge Discovery and Data Mining*.

[130] McSherry, F. D. (2009). Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proc. of the ACM SIGMOD International Conference on Management of Data*.

[131] Mei, S. and Zhu, X. (2015). The security of latent dirichlet allocation. In *Proc. of the International Conference on Artificial Intelligence and Statistics*.

[132] Meng, D. and Chen, H. (2017). Magnet: A two-pronged defense against adversarial examples. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)*.

[133] Metzen, J. H., Genewein, T., Fischer, V., and Bischoff, B. (2017). On detecting adversarial perturbations. In *Proc. of the International Conference on Learning Representations (ICLR)*.

[134] Michael LaForgia and Nicholas Confessore and Gabriel J.X. Dance (2018). Facebook Rebuked for Failing to Disclose Data-Sharing Deals. `https://www.nytimes.com/2018/12/19/technology/facebook-data-privacy-criticism.html`.

[135] Mikians, J., Gyarmati, L., Erramilli, V., and Laoutaris, N. (2012). Detecting price and search discrimination on the internet. In *Proc. of the Workshop on Hot Topics in Networks (HotNets)*.

[136] Mikians, J., Gyarmati, L., Erramilli, V., and Laoutaris, N. (2013). Crowd-assisted Search for Price Discrimination in E-Commerce: First results. *arXiv*.

[137] Milli, S., Miller, J., Dragan, A. D., and Hardt, M. (2019). The social cost of strategic classification. In *Proc. of the Conference on Fairness, Accountability, and Transparency*.

[138] Mirman, M., Gehr, T., and Vechev, M. (2018). Differentiable abstract interpretation for provably robust neural networks. In *Proc. of the International Conference on Machine Learning Research (ICML)*.

[139] Mohan, P., Thakurta, A., Shi, E., Song, D., and Culler, D. (2012). GUPT: Privacy preserving data analysis made easy. In *Proc. of the 2012 ACM SIGMOD International Conference on Management of Data*.

[140] Mohler, G. O., Short, M. B., Malinowski, S., Johnson, M., Tita, G. E., Bertozzi, A. L., and Brantingham, P. J. (2015). Randomized controlled field trials of predictive policing. *Journal of the American statistical association*.

[141] Nath, S. (2015). MAdScope: Characterizing Mobile In-App Targeted Ads. *Proceedings of ACM Mobisys*.

[142] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

[143] Ng, A. Y. (2004). Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proc. of the International Conference on Machine Learning Research (ICML)*.

[144] Nikolaenko, V., Weinsberg, U., Ioannidis, S., Joye, M., Boneh, D., and Taft, N. (2013). Privacy-preserving ridge regression on hundreds of millions of records. In *Proc. of IEEE Symposium on Security and Privacy (S&P)*.

[145] Nissim, K., Raskhodnikova, S., and Smith, A. (2007). Smooth sensitivity and sampling in private data analysis. In *Proc. of the ACM Symposium on Theory of Computing (STOC)*.

[146] NYC (2018). NYC Taxi & Limousine Commission - trip record data. `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`.

[147] Papernot, N., McDaniel, P., Sinha, A., and Wellman, M. P. (2018). Sok: Security and privacy in machine learning. In *Proc. of the IEEE European Symposium on Security and Privacy (EuroS&P)*.

[148] Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. (2016a). Distillation as a defense to adversarial perturbations against deep neural networks. In *Proc. of IEEE Symposium on Security and Privacy (S&P)*.

[149] Papernot, N., McDaniel, P. D., Sinha, A., and Wellman, M. P. (2016b). Towards the science of security and privacy in machine learning. *arXiv*.

[150] Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2015). Deep face recognition. In *Proc. of the British Machine Vision Conference (BMVC)*.

[151] Pascanu, R., Stokes, J. W., Sanossian, H., Marinescu, M., and Thomas, A. (2015). Malware classification with recurrent networks. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[152] Pearl, J. et al. (2009). Causal inference in statistics: An overview. *Statistics surveys*.

[153] Peck, J., Roels, J., Goossens, B., and Saeys, Y. (2017). Lower bounds on the robustness to adversarial perturbations. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*.

[154] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Reserach*.

[155] Pei, K., Cao, Y., Yang, J., and Jana, S. (2017). Deepxplore: Automated whitebox testing of deep learning systems. In *Proc. of the ACM Symposium on Operating Systems Principles (SOSP)*.

[156] Perry, W. L. (2013). *Predictive policing: The role of crime forecasting in law enforcement operations*. Rand Corporation.

[157] Pouya Samangouei, Maya Kabkab, Rama Chellappa (2018). Defense-GAN: Protecting classifiers against adversarial attacks using generative models. *Proc. of the International Conference on Learning Representations (ICLR)*.

[158] Proserpio, D., Goldberg, S., and McSherry, F. (2014). Calibrating data to sensitivity in private data analysis: a platform for differentially-private analysis of weighted datasets. *Proc. of the International Conference on Very Large Data Bases (VLDB)*.

[159] Qin, Y., Carlini, N., Goodfellow, I., Cottrell, G., and Raffel, C. (2019). Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. *arXiv*.

[160] Raghavan, M., Slivkins, A., Vaughan, J. W., and Wu, Z. S. (2018). The externalities of exploration and how data diversity helps exploitation. *Proc. of the ACM Conference on Learning Theory (COLT)*.

[161] Raghunathan, A., Steinhardt, J., and Liang, P. (2018). Certified defenses against adversarial examples. *arXiv*.

[162] Ravi, S. (2017). On-device machine intelligence. `https://ai.googleblog.com/2017/02/on-device-machine-intelligence.html`.

[163] Rogers, R., Roth, A., Ullman, J., and Vadhan, S. (2018). Privacy odometers and filters: Pay-as-you-go composition. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*.

[164] Rogers, R. M., Roth, A., Ullman, J., and Vadhan, S. (2016). Privacy odometers and filters: Pay-as-you-go composition. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*.

[165] Roy, I., Setty, S. T., Kilzer, A., Shmatikov, V., and Witchel, E. (2010). Airavat: Security and privacy for MapReduce. In *Proc. of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.

[166] Rubin, D. B. (1974). Estimating the causal effects of treatments in randomized and non-randomized studies. *Journal of Educational Psychology*.

[167] SafeGov.org (2013). Declaration of Kyle C. Wong in Support of Google Inc.'s Opposition to Plaintiffs' Motion for Class Certification.

[168] Salman, H., Yang, G., Li, J., Zhang, P., Zhang, H., Razenshteyn, I. P., and Bubeck, S. (2019). Provably robust deep learning via adversarially trained smoothed classifiers. *arXiv*.

[169] Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., and Dennison, D. (2015). Hidden technical debt in machine learning systems. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*.

[170] Selenium (2015). `http://www.seleniumhq.org/`.

[171] Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms. Appendix B.* Cambridge University Press.

[172] Shiebler, D. and Tayal, A. (2010). Making machine learning easy with embeddings. SysML `http://www.sysml.cc/doc/115.pdf`.

[173] Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *Proc. of IEEE Symposium on Security and Privacy (S&P)*.

[174] Sinha, A., Namkoong, H., and Duchi, J. (2018). Certifying Some Distributional Robustness with Principled Adversarial Training. In *Proc. of the International Conference on Learning Representations (ICLR)*.

[175] Smith, A. and Thakurta, A. (2013). Differentially private model selection via stability arguments and the robustness of lasso. *Journal of Machine Learning Reserach*.

[176] Song, Y., Shu, R., Kushman, N., and Ermon, S. (2018). Generative adversarial examples. *arXiv*.

[177] Srivastava, A., König, A. C., and Bilenko, M. (2016). Time adaptive sketches (ada-sketches) for summarizing data streams. In *Proc. of the ACM SIGMOD International Conference on Management of Data*.

[178] Strimel, G. P., Sathyendra, K. M., and Peshterliev, S. (2018). Statistical model compression for small-footprint natural language understanding. arXiv.

[179] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[180] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. In *Proc. of the International Conference on Learning Representations (ICLR)*.

[181] Talwar, K., Thakurta, A., and Zhang, L. (2015). Nearly-optimal private LASSO. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*.

[182] Tensorflow r1.5 (2017). Resnet models. `https://github.com/tensorflow/models/tree/r1.5/research/resnet`.

[183] Tibshirani, R. (1994). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B*.

[184] Tjeng, V., Xiao, K., and Tedrake, R. (2017). Evaluating robustness of neural networks with mixed integer programming. *arXiv*.

[185] Tramèr, F., Kurakin, A., Papernot, N., Boneh, D., and McDaniel, P. D. (2017). Ensemble adversarial training: Attacks and defenses. *arXiv*.

[186] Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., and Ristenpart, T. (2016). Stealing machine learning models via prediction apis. In *Proc. of USENIX Security*.

[187] Valentino-Devries, J., Singer-Vine, J., and Soltani, A. (2012). WSJ.com - Websites Vary Prices, Deals Based on Users' Information.

[188] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Reserach*.

[189] Vissers, T., Nikiforakis, N., Bielova, N., and Joosen, W. (2014). Crying Wolf? On the Price Discrimination of Online Airline Tickets. *Hot Topics in Privacy Enhancing Technologies*.

[190] Wang, S., Pei, K., Justin, W., Yang, J., and Jana, S. (2018a). Efficient formal safety analysis of neural networks. *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*.

[191] Wang, S., Pei, K., Justin, W., Yang, J., and Jana, S. (2018b). Formal security analysis of neural networks using symbolic intervals. *Proc. of USENIX Security*.

[192] Wang, Y. and Blei, D. M. (2018). The blessings of multiple causes. *arXiv*.

[193] Wang, Y.-X. (2018). Revisiting differentially private linear regression: optimal and adaptive prediction & estimation in unbounded domain. *arXiv*.

[194] Weinberger, K., Dasgupta, A., Langford, J., Smola, A., and Attenberg, J. (2009). Feature hashing for large scale multitask learning. In *Proc. of the International Conference on Machine Learning Research (ICML)*.

[195] Weng, T.-W., Zhang, H., Chen, P.-Y., Yi, J., Su, D., Gao, Y., Hsieh, C.-J., and Daniel, L. (2018). Evaluating the robustness of neural networks: An extreme value theory approach. *arXiv*.

[196] Wikipedia (Accessed: 2017). Operator norm. `https://en.wikipedia.org/wiki/Operator_norm`.

[197] Wong, E. and Kolter, Z. (2018). Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proc. of the International Conference on Machine Learning Research (ICML)*.

[198] Wu, C.-J., Brooks, D., Chen, K., Chen, D., Choudhury, S., Dukhan, M., Hazelwood, K., Isaac, E., Jia, Y., Jia, B., Leyvand, T., Lu, H., Lu, Y., Qiao, L., Reagen, B., Spisak, J., Sun, F., Tulloch, A., Vajda, P., Wang, X., Wang, Y., Wasti, B., Wu, Y., Xian, R., Yoo, S., and Zhang, P. (2019). Machine learning at Facebook: Understanding inference at the edge. In *Proc. of the IEEE International Symposium on High-Performance Computer Architecture (HPCA)*.

[199] Wu, T. T., Chen, Y. F., Hastie, T., Sobel, E., and Lange, K. (2009). Genome-wide association analysis by lasso penalized logistic regression. *Bioinformatics*.

[200] Xiao, C., Li, B., Zhu, J., He, W., Liu, M., and Song, D. (2018a). Generating adversarial examples with adversarial networks. *arXiv*.

[201] Xiao, C., Zhu, J., Li, B., He, W., Liu, M., and Song, D. (2018b). Spatially transformed adversarial examples. *arXiv*.

[202] Xing, X., Meng, W., Doozan, D., Feamster, N., Lee, W., and Snoeren, A. C. (2014). Exposing Inconsistent Web Search Results with Bobble. *Passive and Active Measurements Conference*.

[203] Xu, J., Zhang, Z., Xiao, X., Yang, Y., Yu, G., and Winslett, M. (2012). Differentially private histogram publication. In *Proc. of the IEEE International Conference on Data Engineering (ICDE)*.

[204] Yang, C., Wu, Q., Li, H., and Chen, Y. (2017). Generative poisoning attack method against neural networks. *arXiv*.

[205] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, Nate Kushman (2018). Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *Proc. of the International Conference on Learning Representations (ICLR)*.

[206] Yann LeCun, Corinna Cortes, Christopher J.C. Burges (Accessed: 2017). The mnist database of handwritten digits.

[207] Yu, L., Liu, L., Pu, C., Gursoy, M. E., and Truex, S. (2019). Differentially private model publishing for deep learning. In *Proc. of IEEE Symposium on Security and Privacy (S&P)*.

[208] Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv*.

[209] Zeng, J., Ustun, B., and Rudin, C. (2017). Interpretable classification models for recidivism prediction. *Journal of the Royal Statistical Society Series A*.

[210] Zhang, D., McKenna, R., Kotsogiannis, I., Hay, M., Machanavajjhala, A., and Miklau, G. (2018). Ektelo: A framework for defining differentially-private computations. In *Proc. of the ACM SIGMOD International Conference on Management of Data*.

[211] Zhang, J., Zhang, Z., Xiao, X., Yang, Y., and Winslett, M. (2012). Functional mechanism: Regression analysis under differential privacy. In *Proc. of the International Conference on Very Large Data Bases (VLDB)*.

[212] Zohrevand, Z., Glässer, U., Tayebi, M. A., Shahir, H. Y., Shirmaleki, M., and Shahir, A. Y. (2017). Deep learning based forecasting of critical infrastructure data. In *Proc. of the Conference on Information and Knowledge Management*.